LaTeX

Marcin Szpyrka

Praktyczny przewodnik

Helion

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Autor: prof. dr hab. Marcin Szpyrka
Akademia Górniczo-Hutnicza
Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej
Katedra Informatyki Stosowanej
al. A. Mickiewicza 30, 30-059 Kraków

Redaktor prowadzący: Małgorzata Kulik

Recenzja naukowa: prof. dr hab. inż. Jacek Izydorczyk

Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Grafika na okładce została wykorzystana za zgodą AdobeStock.com.

Skład komputerowy w systemie LaTeX wykonał autor.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
helion.pl/user/opinie/latepp_ebook
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-289-3668-3

Copyright © Helion S.A. 2026

- Poleć książkę na Facebook.com
- Kup w wersji papierowej
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

Pamięci moich Rodziców

Spis treści

Od autora **ix**

1 Zaczynamy **1**
1.1 Pierwszy dokument 3
1.2 Środowisko pracy 10

2 Wprowadzamy i formatujemy tekst **15**
2.1 Znaki specjalne 15
2.2 Wybór czcionki 17
2.3 LaTeX na kolorowo 28
2.4 Łamanie linii i stron na własną rękę 31
2.5 Cudzysłowy i nie tylko 32
2.6 Znaki diakrytyczne i inne znaki specjalne 35
2.7 Ramki i odległości 36

3 Formatujemy akapity i organizujemy tekst **39**
3.1 Zarządzanie akapitami 39
3.2 Interlinia 44
3.3 Listy 45
3.4 Środowisko minipage 54
3.5 Pakiet tcolorbox 57

4 Zarządzanie dokumentem **63**
4.1 Standardowe klasy dokumentów 63
4.2 Etykiety 71
4.3 Duże projekty 72
4.4 Marginesy 75
4.5 Tytuły rozdziałów i sekcji 79
4.6 Spis treści 82
4.7 Pliki LaTeX-a 86

5 Tabele **87**
5.1 Składnia struktur tabelarycznych 87
5.2 Obramowania 90
5.3 Zarządzanie odstępem między kolumnami 96
5.4 Dane numeryczne w tabelach 98
5.5 Scalanie komórek 99
5.6 Komórki inne niż reszta 102
5.7 Formatowanie nagłówka tabeli 104
5.8 Zagnieżdżanie tabel 107
5.9 Tabele jako wstawki 108
5.10 Tabele wielostronicowe 112

6 Wzory matematyczne **117**
6.1 Tryby matematyczne 117
6.2 Czcionki w trybie matematycznym 121
6.3 Symbole matematyczne 123
6.4 Ograniczniki 126
6.5 Struktury tabelaryczne we wzorach 129
6.6 Serie wzorów 132
6.7 Wzory wieloliniowe 135
6.8 Ozdobniki i odstępy 138

7 Zarządzanie grafiką **145**
7.1 Osadzanie grafiki w dokumencie 145
7.2 Rysunki jako wstawki 148
7.3 Specjalne metody osadzania grafiki 153
7.4 Zarządzanie podpisami 158

8 Zaawansowane środowiska dla dokumentów naukowych **161**
8.1 Środowiska dla definicji, twierdzeń itp. 161
8.2 Listingi 166
8.3 Składanie algorytmów w pseudokodzie 176

9 Zarządzanie bibliografią **183**
9.1 Samodzielne formatowanie bibliografii 183
9.2 Rekordy w BibTEX-u 185
9.3 BibTEX 189
9.4 BibLATEX 193

10 Ipe **199**
10.1 Wprowadzenie . . . 199
10.2 Tryby rysowania . . . 201
10.3 Osadzanie obiektów LaTeX-owych . . . 207
10.4 Tryby przyciągania . . . 209
10.5 Transformacje obiektów . . . 215
10.6 Ipelety . . . 219
10.7 Wiele warstw . . . 222
10.8 Siatki do wspomagania rysowania . . . 226
10.9 Przycinanie grafiki do ścieżki . . . 229

11 Ti*k*Z **231**
11.1 Pierwszy przykład . . . 231
11.2 Określanie współrzędnych punktów . . . 233
11.3 Rysowanie ścieżek . . . 238
11.4 Węzły . . . 243
11.5 Rysowanie ze stylem . . . 250
11.6 Rysowanie w pętli . . . 253
11.7 Rysowanie w trzech wymiarach . . . 258
11.8 Osadzanie rysunków Ti*k*Z w dokumencie . . . 262
11.9 Nakładanie wzorów na rysunek . . . 267

12 PGFPlots **271**
12.1 Dwuwymiarowy układ współrzędnych . . . 271
12.2 Praca z danymi tabelarycznymi . . . 281
12.3 Wykresy słupkowe . . . 287
12.4 Rysowanie poza środowiskiem `axis` . . . 291
12.5 Zaznaczanie obszarów między krzywymi . . . 293
12.6 Kreślenie wykresów w układzie trójwymiarowym . . . 297

13 Przygotowanie prezentacji **307**
13.1 Wprowadzenie do Beamera . . . 307
13.2 Slajdy . . . 310
13.3 Slajd jako seria scen . . . 314
13.4 Zarządzanie strukturą prezentacji . . . 321
13.5 Motywy . . . 323
13.6 Przygotowanie własnego motywu . . . 324

14 Uzupełnienie dla dużych projektów **327**
14.1 Przypisy i notatki na marginesie 327
14.2 Style strony 330
14.3 Spisy 333
14.4 Skorowidz 337
14.5 Definiowanie poleceń przez użytkownika 342
14.6 Hiperłącza i metadane 345

15 Środowisko pracy **349**
15.1 Overleaf 349
15.2 TeXstudio 355

16 Użycie AI do pracy nad dokumentem **369**
16.1 Praca z ChatGPT 369
16.2 Wzory matematyczne 372
16.3 Tabele 376
16.4 Wykresy 379
16.5 Grafika w Ti*k*Z 383
16.6 Poprawianie tekstu 385
16.7 Wsparcie AI dla Overleaf 388
16.8 Wsparcie AI dla TeXstudio 391

17 Podsumowanie **393**

Bibliografia **399**

Od autora

Cześć! Pozwól, że na wstępie zaproponuję przejście na ty. Moje imię widziałeś już na okładce książki. Jeżeli po nią sięgnąłeś, aby uczyć się LaTeX-a, to z dużym prawdopodobieństwem mogę przyjąć, że jestem od Ciebie starszy, stąd pozwalam sobie złożyć taką propozycję – traktuj mnie jako starszego kolegę, który chciałby Ci pomóc opanować zasady składania dokumentów w systemie LaTeX.

Na początek kilka słów o mnie i moich związkach z LaTeX-em. Od ponad 25 lat jestem związany z Akademią Górniczo-Hutniczą im. Stanisława Staszica w Krakowie, gdzie aktualnie pracuję na stanowisku profesora w Katedrze Informatyki Stosowanej na Wydziale Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej. LaTeX-a używam również od ponad 25 lat i jest dla mnie podstawowym narzędziem zarówno do publikowania wyników prac naukowych, jak i w pracy dydaktyka. Te 25 lat z LaTeX-em to m.in. kilka napisanych książek, ponad 100 publikacji naukowych i kilka tysięcy slajdów do wykładów, które prowadziłem lub nadal prowadzę. Od 2012 roku nieprzerwanie uczę LaTeX-a. Prowadzę wykłady dla studentów studiów inżynierskich, a od kilku lat dla doktorantów Szkoły Doktorskiej AGH. Jestem ogromnym fanem LaTeX-a i dzielenie się wiedzą na jego temat nieprzerwanie sprawia mi dużą przyjemność.

Niniejsza książka powstała na bazie moich doświadczeń zawodowych. Można powiedzieć, że mój wykład, który ewoluował na przestrzeni lat, przyjął właśnie formę książki. Starałem się, aby był on jak najbardziej praktyczny. Jeżeli jakiś problem można w LaTeX-u rozwiązać na 5 sposobów, to nie opisuję ich wszystkich. W takich przypadkach starałem się wybrać metodę najbardziej uniwersalną i to ją przedstawiałem. Książka jest bogata w szczegółowo omówione przykłady i z założenia ma dostarczać gotowe rozwiązania, które będziesz mógł stosować w swoich dokumentach.

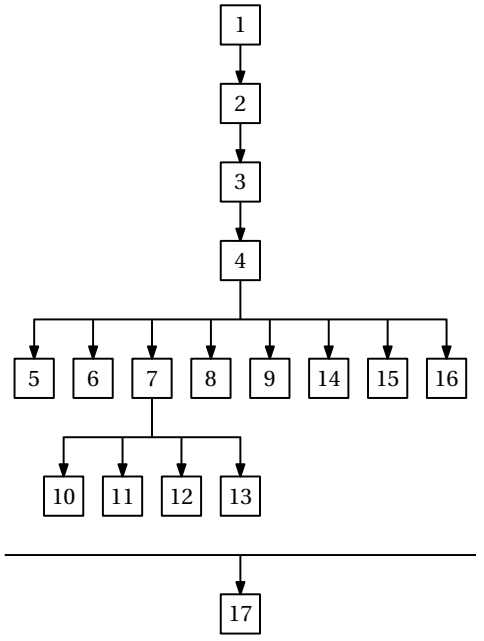
Zakładam, że w większości przypadków po tę książkę sięgną studenci, doktoranci lub pracownicy naukowi, którzy postanowili uzupełnić swój warsztat o umiejętności składu dokumentów w LaTeX-u. Nie jest wymagana żadna znajomość LaTeX-a, aby z niej korzystać. Zakładam, że potrafisz obsługiwać komputer

w podstawowym zakresie (kto dzisiaj nie potrafi?), i to wszystko. Nie ma znaczenia, czy reprezentujesz nauki inżynieryjno-techniczne, ścisłe i przyrodnicze, czy może humanistyczne. Jestem przekonany, że znajdziesz tutaj wiele przydatnych dla Ciebie informacji.

W tym miejscu chciałbym podziękować wydawnictwu Helion za złożenie mi propozycji napisania tej książki. Praca nad nią dała mi dużo satysfakcji. Serdecznie dziękuję również mojemu Koledze z Katedry, Filipowi Kamińskiemu, który był jej pierwszym czytelnikiem. Jego sugestie pozwoliły mi poprawić jakość treści.

Zawartość książki

Zawartość książki została podzielona na 17 rozdziałów. Niektóre z nich są od siebie zależne i powinny być czytane w odpowiedniej kolejności. Na rys. 1 przedstawiłem zalecaną kolejność. Rozdziały 1–4 potraktuj jako niezbędne wprowadzenie do LaTeX-a i przeczytaj je w kolejności zamieszczonej w książce. Później możesz wybierać rozdziały według tematu, który Cię interesuje, ale zalecam lekturę rozdziału 7 przed którymkolwiek z rozdziałów 10–13. Na koniec warto przeczytać podsumowanie (rozdział 17)!



Rys. 1. Zalecana kolejność czytania rozdziałów

Książka, którą oddaję do Twoich rąk, to nie tylko przewodnik po LaTeX-u. Szybko zauważysz, że kilka rozdziałów poświęciłem przygotowywaniu grafiki. Powiedzenie *jeden obraz wart jest tysiąca słów* doskonale pasuje do świata nauki. Dobrze przygotowany rysunek może być dużo więcej wart niż kilka stron tekstu. Znajdziesz w tej książce opis sprawdzonych narzędzi, które doskonale współpracują z LaTeX-em i pozwalają przygotować rewelacyjnej jakości grafikę. W skrócie zawartość poszczególnych rozdziałów prezentuje się następująco:

– W rozdziale 1 przedstawiłem krótki, ale kompletny dokument w LaTeX-u oraz wyjaśniłem, jak zamieścić go i skompilować w środowisku Overleaf. Jeżeli nigdy wcześniej nie spotkałeś się z LaTeX-em, to po lekturze tego rozdziału powinieneś być w stanie przygotować prosty dokument w darmowym środowisku online. Nie potrzebujesz tutaj niczego instalować. Dostęp do niezbędnych narzędzi masz przez przeglądarkę internetową.

– W rozdziale 2 skupiam się na formatowaniu tekstu. Zaczynam od omówienia roli znaków specjalnych w kodzie LaTeX-a. Jej zrozumienie jest kluczowe dla znacznego ograniczenia liczby błędów składniowych. Następnie omawiam polecenia pozwalające modyfikować różne składowe czcionki: rozmiar, rodzinę, serię i kształt. W rozdziale przedstawiłem również przegląd wybranych pakietów zawierających różne kroje czcionek. Mowa jest również m.in. o zastosowaniu kolorów w składzie treści, wymuszaniu łamania linii lub strony i wprowadzaniu szczególnych znaków, takich jak cudzysłowy, półpauzy, pauzy, znaki diakrytyczne itp.

– Rozdział 3 dotyczy zarządzania treścią na poziomie akapitu. Zaczynam od omówienia środowisk zmieniających domyślne wyrównanie do lewej i prawej strony. Następnie przedstawiam środowiska do zamieszczania cytatów i składania poezji, a także przegląd środowisk do składania list wraz z opisem metod zarządzania ich wyglądem. Ostatnie dwa podrozdziały dotyczą środowiska **`minipage`** i pakietu `tcolorbox`. Pozwalają one na uzyskanie ciekawych efektów w postaci opakowania treści w estetyczne ramki i zarządzania układem takich ramek.

– W rozdziale 4 znowu przechodzimy o poziom wyżej i tym razem analizujemy nasz projekt z poziomu całego dokumentu. Rozdział rozpoczyna się od omówienia czterech standardowych klas LaTeX-a. Następnie przedstawiam koncepcję etykiet i mechanizm odwołań kluczowe dla poprawnego odnoszenia się do elementów takich jak równania matematyczne, tabele, rysunki itp. Kolejne podrozdziały dotyczą podziału dużego projektu na wiele plików źródłowych oraz ustawiania rozmiaru marginesów, wyglądu tytułów rozdziałów, sekcji itp. oraz wstawiania i zarządzania wyglądem

spisu treści. Rozdział kończy się (niekompletnym) przeglądem plików, jakie mogą się pojawić w Twoim projekcie.

– Rozdział 5 dotyczy składania tabel. Niezależnie od tego, czy ich zawartością ma być zwykły tekst, liczby, czy grafika, znajdziesz tutaj przegląd kluczowych tematów dotyczących składania i formatowania wyglądu tabel, od najprostszych do bardzo złożonych (np. zagnieżdżanie tabel w tabelach). Rozdział kończą informacje na temat umieszczania tabel w postaci wstawek oraz składania tabel wielostronicowych.

– W rozdziale 6 dotykam serca LaTeX-a. Omawiam w nim zasady składania wzorów matematycznych. Zaczynając od przedstawienia trybów składania wzorów i różnic między nimi, omawiam typowe problemy, jakie mogą się pojawić przy składaniu wzorów, i ich rozwiązania. Jak już wcześniej wspomniałem, książka ta nie miała być encyklopedycznym przeglądem wszystkich możliwości LaTeX-a. Skupiłem się na wybranych, w mojej ocenie najbardziej uniwersalnych, środowiskach do składu matematyki.

– Rozdział 7 jest wprowadzeniem do kwestii zarządzania grafiką w dokumentach LaTeX-a. Rysunki, schematy, diagramy, fotografie itd. są nieodłącznym elementem prac naukowych, dyplomowych czy doktorskich. Rozdział ten dotyczy metod umieszczenia przygotowanej grafiki w dokumencie LaTeX-a. Poza koncepcją wstawek omawiam w nim również m.in. metody umieszczania kilku rysunków obok siebie, „oblewania" rysunku tekstem i zamieszczania dużych rysunków na oddzielnych stronach.

– W rozdziale 8 omawiam środowiska spotykane w dokumentach naukowych, ale już nie tak powszechne jak tabele czy wzory matematyczne. Zaczynam od środowisk typu definicje i twierdzenia, w kolejnym podrozdziale przedstawiam metody składania listingów (eleganckiego dołączania kawałków kodu źródłowego do dokumentu), a kończę na sposobach składania algorytmów w postaci pseudokodu.

– Rozdział 9 dotyczy zarządzania bibliografią. Tłumaczę w nim, jak przygotować bazę rekordów w postaci pliku `bib` oraz jak zarządzać taką bazą z użyciem BibTeX-a i BibLaTeX-a. Zawarłem w nim również przegląd najpopularniejszych stylów składania bibliografii.

– Jeżeli miałbym wskazać najlepszy (znany mi) edytor do przygotowywania grafiki wektorowej współpracujący z LaTeX-em, to odpowiedź może być tylko jedna – Ipe. Rozdział 10 to przegląd możliwości, ale jednocześnie tutorial dla tego niezwykłego narzędzia. Daj mu szansę (2–3 wieczory), a jestem przekonany, że będzie to również Twój ulubiony edytor.

– Rozdział 11 jest wprowadzeniem do pakietu `tikz` (i to skromnym jak na pakiet, którego dokumentacja ma ponad 1300 stron). Pakiet `tikz` prezentuje zupełnie inne podejście do przygotowywania grafiki niż Ipe. Zawiera bogaty zestaw poleceń LaTeX-owych i opcji, które służą do „programowania" grafiki. Nawet jeżeli to podejście dla Ciebie okaże się zbyt trudne, warto choć raz przeczytać ten rozdział. Wiele innych pakietów bazuje na `tikz` i dobrze jest znać choćby podstawowe koncepcje.

– Rozdział 12 dotyczy pakietu `pgfplots`. W połączeniu z pakietem `tikz` daje on olbrzymie możliwości kreślenia dwu- i trójwymiarowych wykresów z poziomu LaTeX-a, bez potrzeby uciekania się do zewnętrznych narzędzi. W rozdziale omówiłem możliwości współpracy pakietu z różnymi danymi wejściowymi oraz zasady tworzenia i formatowania wykresów (również parametrycznych) w przestrzeni $\mathbb{R}^2$ i $\mathbb{R}^3$.

– W rozdziale 13 omawiam zasady przygotowywania prezentacji z zastosowaniem pakietu i klasy `beamer`. Poza podstawowymi informacjami dotyczącymi komponowania zawartości slajdu znajdziesz tu informacje o sposobach dzielenia slajdów na wiele scen, odsłaniania slajdu etapami (również rysunków tworzonych z użyciem pakietu `tikz`) lub podmieniania części jego zawartości. W ostatnim podrozdziale pokazałem szybkie metody na przygotowanie własnego motywu prezentacji.

– Rozdział 14 stanowi w pewnym sensie uzupełnienie dla rozdziału 4. Zebrałem tutaj kilka tematów dotyczących większych opracowań, które są rzadziej używane. W rozdziale jest mowa m.in. o przypisach, stylach strony, generowaniu różnego rodzajów spisów (rysunki, tabele, algorytmy) i przygotowywaniu skorowidza. Ostatnie podrozdziały dotyczą definiowania nowych poleceń LaTeX-a przez użytkownika oraz wykorzystywania hiperłączy i metadanych w plikach `pdf`.

– Rozdział 15 poświęciłem omówieniu dwóch popularnych środowisk do pracy z LaTeX-em. Po pierwsze jest tu uzupełnienie opisu środowiska Overleaf z rozdziału 1. Po drugie przedstawiam środowisko TeXstudio. Jest to narzędzie, z którego korzystam na co dzień, i jeśli tylko zdecydujesz się na instalację oprogramowania do pracy z LaTeX-em na swoim komputerze, powinieneś poważnie rozważyć tę opcję.

– W rozdziale 16 zawarłem subiektywny przegląd możliwości wsparcia pracy nad dokumentem przez modele LLM, w szczególności ChatGPT. Zdaję sobie sprawę, jak szybko podejście określane jako AI się rozwija i jak bardzo rozdział ten może być niekompletny za kilka miesięcy, toteż liczę, że przedstawione przykłady będą dobrym wprowadzeniem,

w szczególności dla osób, dla których ta tematyka jest ciągle nowością. Rozdział kończy opis aktualnego wsparcia AI dla Overleaf i TeXstudio.

– Rozdział 17 zawiera krótkie podsumowanie. Starałem się, aby był on inspiracją do rozwijania Twojej wiedzy i umiejętności związanych z LaTeX-em. Wspomniałem w nim o kilku pakietach (niekoniecznie „naukowych"). Mam nadzieję, że będzie to intrygujący sygnał, że poza materiałem w tej książce jest jeszcze sporo do odkrycia.

Zasady notacji

Wszystkie przykłady kodu źródłowego składane są czcionką o stałej szerokości znaków. Dotyczy to zarówno kodu zamieszczonego na listingach, jak również małych fragmentów wplecionych w treść akapitów. Nazwy poleceń i środowisk LaTeX-a zostały wyróżnione pogrubioną czcionką. W większości przypadków bezpośrednio pod listingiem zamieszczany jest wynik kompilacji kodu z tego listingu. Ponieważ książka również została złożona w LaTeX-u i często korzystam w przykładach z tych samych czcionek co w książce, używam poziomych linii, aby wskazać miejsca, gdzie kończą się przykłady.

Jeżeli do kompilacji danego przykładu konieczne jest dołączenie jakiegoś pakietu lub biblioteki, to informacja taka podawana jest w następującej ramce:

Dodatkowo istotne uwagi, zazwyczaj dotyczące radzenia sobie z problemami, które mogą się pojawić w związku z omawianym aktualnie tematem, zamieszczam w takich ramkach:

Dołożyłem wszelkich starań, aby wyeliminować z treści książki wszelkie błędy, zwłaszcza merytoryczne. Mam nadzieję, że te, które zostały, nie są zbyt poważne. Jeżeli chcesz mi przesłać jakieś uwagi na temat treści, skieruj je na adres `mszpyrka@agh.edu.pl`. Życzę owocnej lektury!

Marcin Szpyrka
Kraków
Maj 2025

Rozdział 1

Zaczynamy

LaTeX jest systemem składu umożliwiającym tworzenie dowolnego typu dokumentów (w szczególności naukowych i technicznych) o wysokiej jakości typograficznej. Specyficznie zapisywana nazwa, którą tu widzisz, jest traktowana jednocześnie jako logo tego systemu. Wysoka jakość składu jest niezależna od rozmiaru dokumentu. Nie ma znaczenia, czy chcemy przygotować jednostronicowy list, czy liczącą ponad tysiąc stron książkę. LaTeX jest zarówno oprogramowaniem, jak i zestawem instrukcji umożliwiających autorom skład i wydruk ich prac na wysokim poziomie. Wynikiem końcowym jest zazwyczaj dokument w formacie `pdf`. Dokumenty złożone w LaTeX-u cechuje determinizm, tj. uzyskamy ten sam efekt niezależnie od systemu operacyjnego, w którym odbywa się kompilacja, czy też drukarki użytej do drukowania dokumentów (o ile nie spróbujesz drukować swojego dokumentu np. na drukarce igłowej).

Wszystko zaczęło się w roku 1977, gdy Donald E. Knuth pracował nad poprawionym wydaniem drugiego tomu swojej książki *The Art of Computer Programming*. Wydanie to złożono w nowym systemie składu drukarskiego używanym przed wydawnictwo, ale próbne odbitki pokazały, że jakość jest dużo gorsza niż w pierwszym wydaniu. Knuth porównał to z książką Patricka Winstona *Artificial Intelligence*, która ukazała się mniej więcej w tym samym czasie, a którą złożono cyfrowo, i uznał, że to jest przyszłość dla składu książek. Postanowił opracować system do cyfrowego składu swoich książek, który umożliwiłby mu pełną kontrolę nad ich wyglądem. Ponieważ system TeX, nad którym pracował, przeznaczony był do składania artykułów i książek zawierających dużą liczbę wzorów matematycznych, mniej więcej rok po rozpoczęciu prac przedstawił go na dorocznym zjeździe Amerykańskiego Towarzystwa Matematycznego (AMS). System, mimo początkowej fazy swojego rozwoju, wzbudził duże zainteresowanie, gdyż m.in. pozwalał autorom kontrolować jakość składu, był darmowy i szybko

stał się dostępny dla dowolnych komputerów i systemów operacyjnych. Prace nad systemem TEX zajęły prawie 10 lat, a sam Knuth był wspierany przez wiele osób w doprowadzeniu swojego dzieła do końca. Nazwa TEX zawiera 3 greckie litery: τ, ϵ i χ, więc w języku polskim wymawiamy ją „tech" i jest to jedyna poprawna wersja.

Na bazie TEX-a Leslie Lamport opracował w 1985 r. system LATEX, który pozwala się skupić na strukturze dokumentu zamiast na szczegółach formatowania. Nazwę tę wymawiamy w języku polskim jako „latech" i znowu można powiedzieć, że jest to w zasadzie jedyna poprawna wersja. W języku angielskim można spotkać cztery sposoby wymawiania tej nazwy, które możemy uznać za poprawne: „latech", „latek", „lejtech" i „lejtek". Ponieważ nazwa litery χ wymawiana jest jako „kai", stąd można zrozumieć pochodzenie wersji z literą k na końcu. Jeżeli przyjmiemy, że pierwsza sylaba w nazwie pochodzi od nazwiska twórcy, to możemy w obu językach przyjąć jako poprawną wersję „latech" i tego uzgodnienia będziemy się trzymać dalej (jest to jedna z dwóch wersji wskazanych przez samego Lamporta, druga to „lejtech"). Każdą inną wersję wymowy należy uznać za błędną, a w szczególności „lateks".

Grupa prowadzona przez Franka Mittelbacha opracowała w 1994 r. system w wersji LATEX 2ε. Jest to do dzisiaj najczęściej używany standard, co nie znaczy, że system nie jest rozwijany. Powstają nowe pakiety, rozwijane są istniejące i trwają intensywne prace nad wersją LATEX3, jednak obecnie jest to wersja wyłącznie do eksperymentowania.

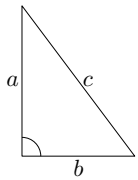
LATEX automatyzuje wiele prac związanych ze składaniem dokumentów, np. referencje, cytowania, generowanie spisów treści, rysunków, symboli itp. Autor ma się przeważnie skupić na przygotowaniu treści dokumentu, a nie na jego formatowaniu. Większość wiodących wydawnictw opracowuje własne *klasy dokumentów*, które udostępnia autorom. Klasa dokumentu zawiera mnóstwo szczegółów dotyczących składu, zaczynając od określenia podstawowej czcionki, formatu papieru, rozmiarów marginesów, ustalenia, czy skład ma być w jednej, czy dwóch kolumnach itd., a kończąc na takich szczegółach jak określenie, czy w podpisie pod rysunkiem po skrócie „Rys." ma być kropka, czy np. kropka i dwukropek. Zastosowanie tej samej klasy dokumentu w dużej mierze gwarantuje, że artykuł opublikowany w ostatnim numerze czasopisma *X* i artykuł opublikowany w nim 10 lat wcześniej wyglądają podobnie. Z tego powodu wiele czasopism naukowych przyjmuje do druku wyłącznie publikacje złożone w LATEX-u. Z drugiej strony, z perspektywy autora, zmiana klasy dokumentu wymaga zazwyczaj kosmetycznych poprawek w preambule dokumentu (zaraz sobie wyjaśnimy, co to takiego) i jego przekompilowania. W ten sposób niemal „magicznie" możemy w dwie minuty zmienić całkowicie wygląd naszego artykułu i dopasować go do oczekiwań innego wydawnictwa.

1.1 Pierwszy dokument

Planując tę książkę, postawiłem sobie za zadanie napisać ją tak, by każdy rozdział rozwijał Twoje umiejętności, a jednocześnie był w pewien sposób kompletny. Rozumiem to następująco: po lekturze pierwszego rozdziału powinieneś być w stanie przygotować prosty, ale kompletny dokument w LaTeX-u; w kolejnych rozdziałach umiejętności te poszerzysz, wzbogacając swoje dokumenty o nowe elementy, na przykład rozdział 2 dostarczy informacji o formatowaniu tekstu, rozdział 3 skupia się na formatowaniu akapitów, w rozdziale 4 jest mowa, jak zarządzać całym projektem, zwłaszcza gdy składa się z wielu plików, potem nauczymy się składać tabele, wzory matematyczne itd.

1 Twierdzenie Pitagorasa

Twierdzenie Pitagorasa jest zdecydowanie najpopularniejszym z tych, z jakimi spotykamy się w szkole. Jego nazwa została połączona z nazwiskiem wielkiego filozofa greckiego żyjącego od ok. 572 roku p.n.e. do ok. 497 roku p.n.e. Sam Pitagoras raczej nie był twórcą tego twierdzenia, które znane było długo przed jego narodzinami. Ponad tysiąc lat wcześniej wiedziano, że wykorzystując trójkąt, którego boki mają odpowiednio 3, 4 i 5 jednostek miary, możemy wyznaczyć kąt prosty. Bardziej prawdopodobne jest, że Pitagoras udowodnił znane twierdzenie i dlatego później jego uczniowie nazwali je na cześć mistrza.



Rysunek 1: Trójkąt prostokątny

Jedno z bardziej popularnych sformułowań słynnego twierdzenia brzmi: *kwadrat strony przeciwnej do kąta prostego jest równy sumie kwadratów boków sąsiadujących z kątem prostym*. Jeżeli przyjmiemy takie oznaczenia jak na rys. 1, gdzie a, b i c to długości boków w trójkącie prostokątnym, to twierdzenie możemy zapisać krótko w postaci $c^2 = a^2 + b^2$.

Rys. 1.1. Pierwszy dokument

Na rys. 1.1 pokazano nasz pierwszy cel. Zamiast tradycyjnego „Witaj, świecie!” podejmiemy się nieco ambitniejszego zadania. Dokument ten składa się z jednej sekcji o tytule „Twierdzenie Pitagorasa”, zawierającej dwa akapity tekstu

przedzielone rysunkiem (sam rysunek został przygotowany w zewnętrznym edytorze i tutaj skupimy się na jego osadzeniu w dokumencie). W drugim akapicie pojawia się fragment tekstu złożony kursywą i nasz pierwszy wzór matematyczny. Dodatkowo mamy jeszcze odwołanie do rysunku.

Zawartość rys. 1.1 to podgląd pliku `pdf`, który jest wynikiem kompilacji *pliku źródłowego*. Plik źródłowy ma zazwyczaj rozszerzenie `tex` i jest plikiem tekstowym, zawierającym treść naszego dokumentu, w której dodatkowo zamieszczono polecenia LaTeX-a. Można go sporządzić w dowolnym, nawet najprostszym edytorze tekstu, ale zdecydowanie wygodniej jest korzystać z narzędzi przygotowanych do współpracy z LaTeX-em. O narzędziach opowiem więcej w następnym podrozdziale. Teraz skupimy się wyłącznie na pliku źródłowym pokazanym na listingu 1.1.

Listing 1.1. Zawartość pliku źródłowego dla dokumentu z rys. 1.1

```
\documentclass[a4paper,12pt]{article}      % preambuła
\usepackage[polish]{babel}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{graphicx}
\usepackage{newtx}

\begin{document}                           % część główna

\section{Twierdzenie Pitagorasa}
\label{sec:twierdzenie:pitagorasa}

Twierdzenie Pitagorasa jest zdecydowanie najpopularniejszym z tych, z
jakimi spotykamy się w szkole. Jego nazwa została połączona z
nazwiskiem wielkiego filozofa greckiego żyjącego od ok. 572 roku p.n.e.
do ok. 497 roku p.n.e. Sam Pitagoras raczej nie był twórcą tego
twierdzenia, które znane było długo  przed jego narodzinami. Ponad
tysiąc lat wcześniej wiedziano, że wykorzystując trójkąt, którego boki
mają odpowiednio 3, 4 i 5 jednostek miary, możemy wyznaczyć kąt prosty.
Bardziej prawdopodobne jest, że Pitagoras udowodnił  znane twierdzenie
i~dlatego później jego uczniowie nazwali je na cześć mistrza.

\begin{figure}[h]
\centering
\includegraphics{trojkat_prostokatny.pdf}
\caption{Trójkąt prostokątny}
\label{fig:trojkat:prostokatny}
\end{figure}

```

```
30 Jedno z bardziej popularnych sformułowań słynnego twierdzenia brzmi:
31 \textit{kwadrat strony przeciwnej do kąta prostego jest równy sumie
32 kwadratów boków sąsiadujących z kątem prostym}. Jeżeli przyjmiemy
33 takie oznaczenia jak na rys.~\ref{fig:trojkat:prostokatny}, gdzie $a$,
34 $b$ i $c$ to długości boków w trójkącie prostokątnym, to twierdzenie
35 możemy zapisać krótko w~postaci $c^2 = a^2 + b^2$.
36
37 \end{document}
```

W pliku źródłowym możemy wyróżnić dwie części: *preambułę* i *część główną*. Pierwsze linie tekstu należące do nich zostały wyróżnione za pomocą stosownych komentarzy. W LaTeX-u symbol % oznacza początek komentarza, który rozciąga się do końca linii. Komentarze są ignorowane na etapie kompilacji, tj. generowania pliku wynikowego (tutaj `pdf`) na podstawie pliku źródłowego. Możemy w nich pozostawiać komentarze dla siebie lub współpracowników.

Zanim skasujesz fragment dokumentu, zastanów się, czy nie lepiej na razie go zakomentować. Dla kompilatora przestanie być widoczny, a Ty zawsze możesz do niego wrócić, gdyby zaszła taka potrzeba.

Wszystkie polecenia w LaTeX-u rozpoczynają się od ukośnika wstecznego (w języku informatyki najczęściej używa się angielskiej nazwy znaku – backslash) \. Możemy również spotkać nazwy: lewy ukośnik, odwrotny ukośnik i w-tył-ciach. Dla porównania terminem ukośnik lub prawy ukośnik określamy znak /. W dalszej części książki będziemy używać wyłącznie ukośnika wstecznego (z jednym małym wyjątkiem, co zostanie wyraźnie zaznaczone). Polecenie LaTeX-a może się składać z ukośnika wstecznego i nazwy zawierającej wyłącznie litery, której koniec wskazuje spacja, cyfra lub dowolny inny znak niebędący literą, bądź z ukośnika wstecznego i dokładnie jednego znaku niebędącego literą. Jeżeli polecenie przyjmuje obowiązkowe parametry, to są one umieszczane w nawiasach klamrowych { } (każdy parametr ma własną parę nawiasów), za to w przypadku opcjonalnych – w nawiasie kwadratowym (w przypadku niektórych poleceń parametry opcjonalne znajdują się w pojedynczym nawiasie i są oddzielone przecinkami). Przykładami poleceń w naszym pierwszym dokumencie są **`\documentclass`**, **`\usepackage`**, **`\section`** itd. Na podstawie typów nawiasów łatwo zidentyfikujesz, które parametry są obowiązkowe w tych poleceniach, a które opcjonalne.

Środowisko LaTeX-a jest strukturą złożoną z pary poleceń **`\begin`**`{name}` i **`\end`**`{name}`. Wewnątrz środowiska można używać innych poleceń lub środowisk. Argumenty dla środowiska podaje się po **`\begin`**`{name}`. W dokumencie

z listingu 1.1 mamy użyte dwa środowiska **document** i **figure**. To drugie przyjmuje parametr opcjonalny.

Pierwszym poleceniem w pliku źródłowym musi być **\documentclass** (przed nim mogą być tylko komentarze). Jego obowiązkowym parametrem jest nazwa klasy dokumentu. W naszym przypadku jest to klasa `article`, będąca jedną z czterech podstawowych klas LaTeX-a. Służy ona do przygotowywania niezbyt długich dokumentów, a najwyższym poziomem organizacji tekstu jest w niej sekcja. Więcej o klasach dokumentów opowiem w podrozdziale 4.1. W naszym przypadku polecenie **\documentclass** przyjmuje jeszcze dwa parametry opcjonalne: `a4paper` i `12pt`. Pierwszy ustala wielkość papieru, a drugi bazowy rozmiar czcionki w dokumencie. Gdybyśmy je pominęli, zostałyby użyte domyślne wartości – odpowiednio `letter` i `10pt`.

Dalej w preambule mamy pięciokrotnie użyte polecenie **\usepackage**. Dodaje ono do dokumentu dodatkowy pakiet, który jest plikiem bądź kolekcją plików zawierającą dodatkowe polecenia LaTeX-a lub modyfikującą te już istniejące. Jeżeli kiedykolwiek programowałeś, to dołączenie pakietu jest jak dodanie kolejnej biblioteki w języku programowania. W repozytoriach LaTeX-a dostępne są tysiące pakietów, z których możemy korzystać, jeżeli jest taka potrzeba. Bez pakietów mamy dostęp tylko do bazowego zestawu poleceń. Każdy pakiet rozszerza nasze możliwości. Rola użytych tutaj pakietów jest następująca:

- `babel` – pakiet ten dodaje reguły dzielenia słów dla wskazanych języków. W tym przypadku dodajemy zasady dzielenia wyrazów w języku polskim.
- `inputenc` – pakiet odpowiada za translację różnych wejściowych kodowań dokumentów na wewnętrzne kodowanie LaTeX-a. W naszym przypadku wskazano, że dokument źródłowy został zapisany z kodowaniem `utf8` (obecnie najczęściej spotykana sytuacja). Inne popularne opcje w przypadku języka polskiego to `latin2` (kodowanie zgodne ze standardem ISO 8859-2) lub `cp1250` (popularne w systemach z rodziny Windows).
- `fontenc` – pakiet określa sposób kodowania czcionki (dotyczy wyjścia). Opcja `OT1` wskazuje na 7-bitowe kodowanie TeXa. Dostępnych jest tylko 128 znaków (tzw. glifów). W kodowaniu tym na przykład litera `ś` nie ma swojego glifu i powstaje przez dodanie akcentu do litery `s`. Opcja `T1` wskazuje na kodowanie 8-bitowe, w którym `ś` jest pojedynczym glifem. Wiele nowszych czcionek dostępnych jest tylko w `T1`. Brak tego pakietu z opcją `T1` oznacza, że słowa zawierające znaki diakrytyczne nie będą poprawnie dzielone, czy też nie uda nam się poprawnie skopiować tekstu z pliku `pdf`.
- `graphicx` – pakiet umożliwia dodawanie do dokumentu zewnętrznych plików z grafiką. Bez tego pakietu nie moglibyśmy skorzystać z polecenia **\includegraphics**.

– `newtx` – pakiet ustawia czcionkę Times Roman (lub równoważną) jako domyślną. Ponadto dostarcza wsparcie dla składu wyrażeń matematycznych dla tej czcionki.

Jeżeli składasz dokument w języku polskim i plik źródłowy wygląda poprawnie, a po kompilacji w pliku `pdf` widzisz dziwne znaki w miejscu polskich liter ze znakami diakrytycznymi (ą, ę, ć itd.), wskazuje to na niezgodność między kodowaniem, w jakim zapisałeś plik, a opcją przekazaną do pakietu `inputenc`.

Przedstawioną tutaj preambułę potraktuj jako punkt wyjścia dla swoich dokumentów. W przyszłości będziesz dodawał do niej kolejne pakiety lub definicje niezbędne w dokumencie. Pracując z LaTeX-em, nigdy nie zaczynam od pustego pliku. Przeważnie sięgam po istniejący już dokument, zapisuję go pod nową nazwą (zazwyczaj w nowym katalogu) i modyfikuję jego zawartość. To pozwala mi zaoszczędzić mnóstwo czasu i zwalnia mnie z pamiętania o wszystkich pakietach, które powinienem dołączyć.
Zaproponowaną preambułę bardzo łatwo dostosujesz do innego języka, zmieniając opcję pakietu `babel` – jeżeli zamierzasz składać publikacje naukowe, to najprawdopodobniej będzie to język angielski. Jeżeli piszesz przede wszystkim w języku polskim, to warto sięgnąć po pakiet `polski`.

Zawartość dokumentu umieszczamy wewnątrz środowiska **`document`** (linie 8 i 37). Wszystko, co umieścimy poniżej środowiska **`document`**, będzie traktowane jako komentarz. Przy komponowaniu tekstu obowiązują dwie proste zasady:

1. Poszczególne słowa oddzielamy spacjami, przy czym liczba spacji nie ma znaczenia. Po kompilacji wielokrotne spacje i tak będą wyglądały jak pojedyncza spacja. Spacje na początku linii są ignorowane.
2. Znakiem końca akapitu jest co najmniej jedna pusta linia (liczba pustych linii nie ma znaczenia), a nie znaki przejścia do nowej linii.

   Linie od 13 do 21 tworzą zatem pierwszy akapit tekstu w naszym dokumencie, a linie od 30 do 35 drugi.

Polecenia LaTeX-a mogą służyć do wygenerowania czegoś nowego, np. polecenie **`\LaTeX`** generuje logo, które widziałeś już kilka razy w tej książce, albo do zmiany formatowania istniejącego elementu dokumentu. Pierwsze polecenie wewnątrz środowiska **`document`** to polecenie **`\section`**. Tworzy ono nową sekcję (w tym przypadku jedyną) i wykorzystuje podany parametr jako tytuł tej

sekcji. Jak już wspomniałem, jest to najwyższy poziom organizacji tekstu w klasie `article`. Dostępne są kolejne poziomy, np. podsekcje (`\subsection`). Więcej o organizacji struktury dokumentu znajdziesz w podrozdziale 4.1.

LaTeX dostarcza bardzo wygodny mechanizm odsyłaczy, dzięki któremu bezbłędnie możemy odwoływać się do elementów zamieszczonych w dokumencie: rozdziałów, sekcji, równań, rysunków, tabel, definicji itd. Mechanizm ten eliminuje możliwość popełnienia pomyłki, jaka może się zdarzyć przy ręcznym numerowaniu. Zachęcam, abyś od samego początku wyrobił w sobie nawyk definiowania etykiet do wszelkich elementów, do których możesz chcieć się odwoływać (zazwyczaj są to te wszystkie elementy, którym przypisywane są numery porządkowe). Etykieta jest tworzona z użyciem polecenia `\label`, a odwołania za pomocą polecenia `\ref`. Instrukcja `\label` umieszczona w linii 11 definiuje etykietę dla naszej sekcji. Polecenie to umieszczamy bezpośrednio po poleceniu `\section`. Przy definiowaniu etykiet lepiej unikać znaków diakrytycznych i spacji. Zamiast spacji możemy używać dwukropków. Użycie gdziekolwiek w dokumencie polecenia `\ref{sec:twierdzenie:pitagorasa}` spowoduje, że na etapie kompilacji zostanie ono zamienione na numer sekcji, do której się odwołujemy.

Środowisko `figure` (linie 23 i 28) służy do zdefiniowania nowego rysunku, a dokładniej – jego zawartość będzie traktowana jako rysunek (nawet jeśli do środka wrzucimy akapit tekstu). Zawartość środowiska jest tzw. *obiektem pływającym*. Element taki musi być umieszczony na jednej stronie. Opcja `h` (here) wskazuje, że chcemy go umieścić w tym samym miejscu w dokumencie wyjściowym, co w pliku źródłowym (czyli między naszymi dwoma akapitami). Jest to tylko nasze oczekiwanie. Jeżeli rysunek byłby za duży, by się zmieścić na tej samej stronie co pierwszy akapit, to zostałby on przesunięty na kolejną stronę. Jednocześnie LaTeX postara się uzupełnić stronę zawartością, która w pliku źródłowym znajduje się po środowisku `figure` (w naszym przypadku mógłby tylko przesunąć drugi akapit, co i tak prawdopodobnie nie starczyłoby do uzyskania kompletnej strony). Więcej na ten temat znajdziesz w podrozdziale 7.2.

Polecenie `\centering` powoduje, że występujące po nim elementy są wyśrodkowane na stronie. Zostało ono zamieszczone wewnątrz środowiska `figure`, więc jego działanie kończy się wraz z zamknięciem środowiska (linia 28) – nie musimy używać nawiasów klamrowych, aby ograniczyć obszar jego działania. Rysunek z trójkątem przygotowałem w edytorze grafiki Ipe (zobacz rozdział 10) i zapisałem w pliku `trojkat_prostokatny.pdf`. Plik ten został zapisany w tym samym katalogu co nasz przykładowy dokument (gdyby plik był w innym katalogu, to moglibyśmy podać ścieżkę dostępu). Polecenie `\includegraphics` pozwala na wstawienie zawartości tego pliku do naszego dokumentu. Ostatnie dwa polecenia w środowisku `figure` definiują podpis pod rysunkiem i etykietę rysunku. Muszą one wystąpić dokładnie w tej kolejności, tj. polecenie `\label`

umieszczamy bezpośrednio po poleceniu `\caption`. Odwołanie do tej etykiety znajdziesz wewnątrz drugiego akapitu. Sprawdź na rys. 1.1, że zostało ono zamienione na stosowny numer.

Wewnątrz drugiego akapitu znajduje się polecenie `\textit{}`. Jego rolą jest składanie tekstu przyjętego jako parametr z użyciem kursywy. W naszym przykładzie kursywą jest złożona treść twierdzenia Pitagorasa.

Ostatni element, jaki nam pozostał, to wzory matematyczne. Przez wzór rozumiemy tutaj dowolną zawartość dokumentu, którą chcemy złożyć w trybie matematycznym. Jeżeli jesteśmy wewnątrz akapitu tekstu, to aby przejść w tryb matematyczny, wystarczy użyć znaku `$`. Treść umieszczona pomiędzy znakami `$` zostanie potraktowana jako wzór matematyczny. Ostatni z wzorów to `$c^2 = a^2 + b^2$`. Symbol `^` odpowiada za umieszczenie kolejnego znaku w indeksie górnym (aby uzyskać a^{12}, należy użyć nawiasów klamrowych `$a^{12}$`). W efekcie po kompilacji dostajemy $c^2 = a^2 + b^2$. Przy składaniu wzorów obowiązują nieco inne zasady dotyczące czcionek lub odstępów między symbolami niż przy składzie w trybie tekstowym, na przykład kompilator ignoruje wszystkie odstępy, a poszczególne znaki traktuje jak zmienne i sam decyduje, gdzie wstawić odstępy. Pojedyncze litery a, b i c, wymienione nieco wcześniej w drugim akapicie, a będące częścią naszego głównego wzoru, również są w trybie matematycznym, co zachowuje spójność składu i wyróżnia je w zwykłym tekście. Więcej na temat składania wzorów znajdziesz w rozdziale 6.

Przeanalizowaliśmy zawartość listingu 1.1. Wiesz już, jakie jest znaczenie poszczególnych linii kodu. Wspomniałem także, że plik źródłowy można przygotować, używając dowolnego edytora tekstu (np. Gedit w Linuksie lub Notatnik w Windows). Pozostaje pytanie, jak skompilować przygotowany plik, by uzyskać dokument w formacie `pdf`. Potrzebujemy do tego jakiejś dystrybucji LaTeX-a, która między innymi dostarczy kompilator, który przetworzy przygotowany plik `tex` w plik `pdf`. Jest to drugi element oprogramowania, obok edytora, który jest niezbędny do pracy z LaTeX-em. Przyjmiemy tutaj założenie, że korzystamy z dystrybucji TeX Live i kompilatora `pdflatex` (najlepiej najnowszej stabilnej wersji). Więcej o tych narzędziach opowiem w podrozdziale 15.2. Jeżeli pracujesz na komputerze, na którym takie oprogramowanie jest już zainstalowane, to zakładając, że nasz dokument zapisaliśmy w pliku `pitagoras.tex`, kompilacja sprowadza się do jednego polecenia:

```
pdflatex pitagoras.tex
```

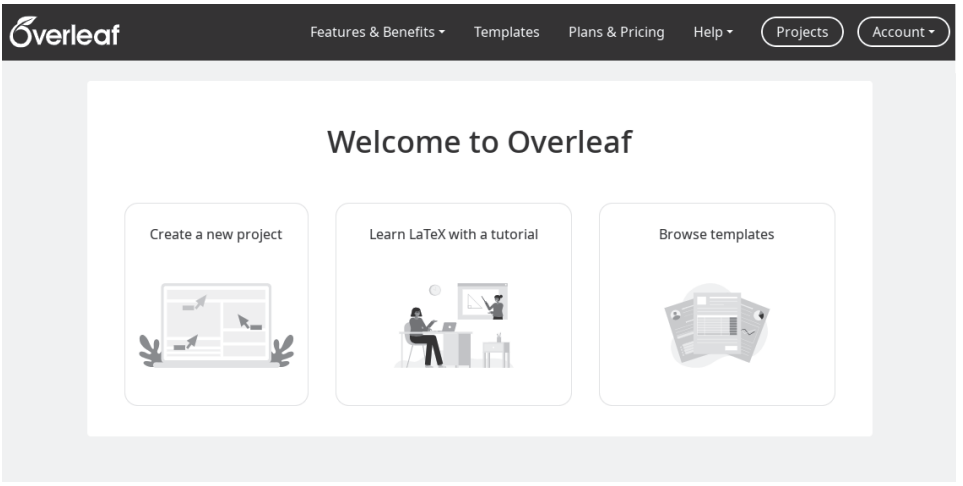
Polecenie takie wydajemy w terminalu z poziomu katalogu, w którym jest zapisany nasz plik. Jeżeli edytor, z którym pracujemy, wspiera używanie LaTeX-a,

to wykonanie takiego polecenia sprowadza się do kliknięcia przycisku lub wciśnięcia klawiszy skrótu. W wyniku kompilacji, jeśli nie było błędów w kodzie, w tym samym katalogu utworzony zostanie m.in. plik `pdf` z naszym dokumentem. W kolejnym podrozdziale pokażę, jak można korzystać z narzędzi oferujących edytor i zintegrowany kompilator (dystrybucja TeX Live) bez instalowania czegokolwiek.

1.2 Środowisko pracy

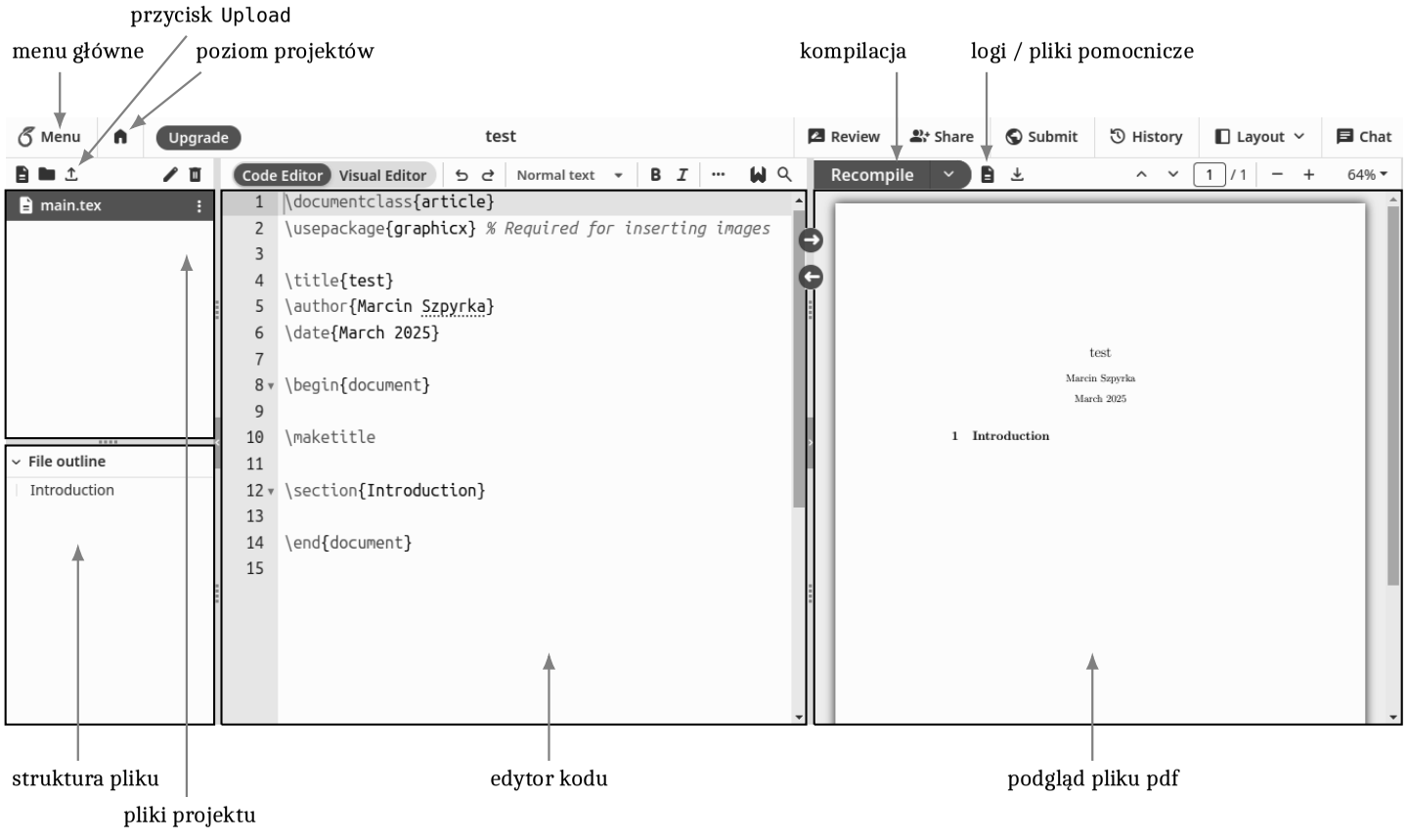
W tym podrozdziale nie będziemy mówili o instalacji jakichkolwiek narzędzi, ale skupimy się na wykorzystaniu darmowego środowiska online, które z poziomu przeglądarki internetowej (niezależnie od systemu operacyjnego, którego używasz) pozwoli Ci pracować z LaTeX-em. Środowisko to zawiera również edytor kodu źródłowego, dzięki czemu nie musimy się na razie zastanawiać nad tym, który edytor najlepiej wybrać.

Środowisko pracy znajduje się pod adresem `https://www.overleaf.com`. Jeżeli nigdy wcześniej go nie używałeś, to musisz zacząć od przycisku `Sign up`, podać swój adres e-mail i ustawić hasło. Możesz również się zalogować, korzystając z konta `Google`, jeśli je posiadasz. Gdy następnie zalogujesz się do systemu, powinieneś zobaczyć ekran startowy pokazany na rys. 1.2.



Rys. 1.2. Ekran startowy środowiska Overleaf

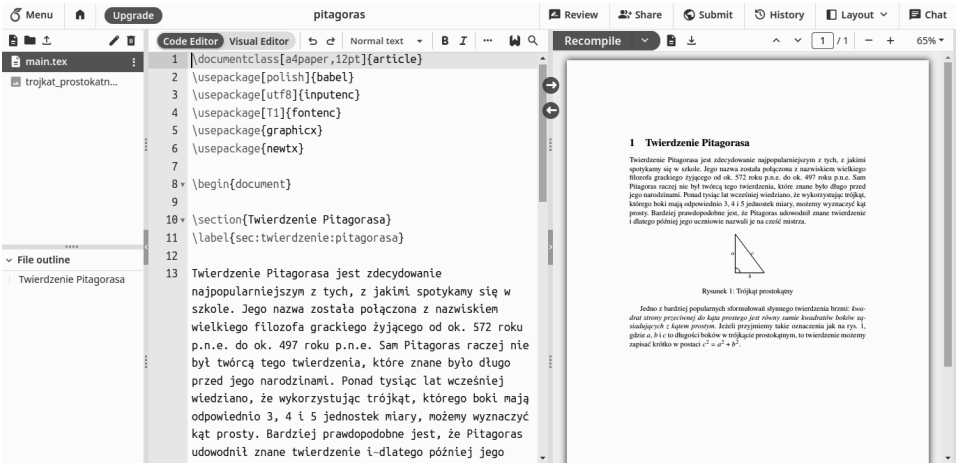
Wybierz kafelek z opcją `Create a new project`, a z rozwijanej listy, która się wyświetli, opcję `Blank Project`. Efekt naszych działań widoczny jest na rys. 1.3. Jak widać, projekt wcale nie jest „pusty”. Overleaf wygenerował szczątkowy



Rys. 1.3. Widok środowiska po wybraniu opcji `Blank Project`

dokument w LaTeX-u i go skompilował. Dwa główne obszary, które widzimy w oknie przeglądarki, to edytor kodu źródłowego po lewej stronie i podgląd pliku `pdf` po prawej. Po wprowadzeniu jakichkolwiek zmian w edytorze wystarczy kliknąć przycisk kompilacji (zob. rys. 1.3), by ponownie skompilować dokument i zaktualizować plik `pdf`. Jest to typowy zabieg, gdy pracujemy z LaTeX-em. Zazwyczaj po dopisaniu kilku linii tekstu kompilujemy dokument, aby zobaczyć efekt naszej pracy. Jeżeli klikniesz strzałkę w dół po prawej stronie przycisku `Recompile`, to w rozwiniętym menu możesz wybrać opcję automatycznej rekompilacji. Będzie się ona uruchamiać za każdym razem, gdy pojawi się przerwa we wprowadzaniu tekstu. Jeżeli piszesz powoli, to odradzam używanie tej opcji.

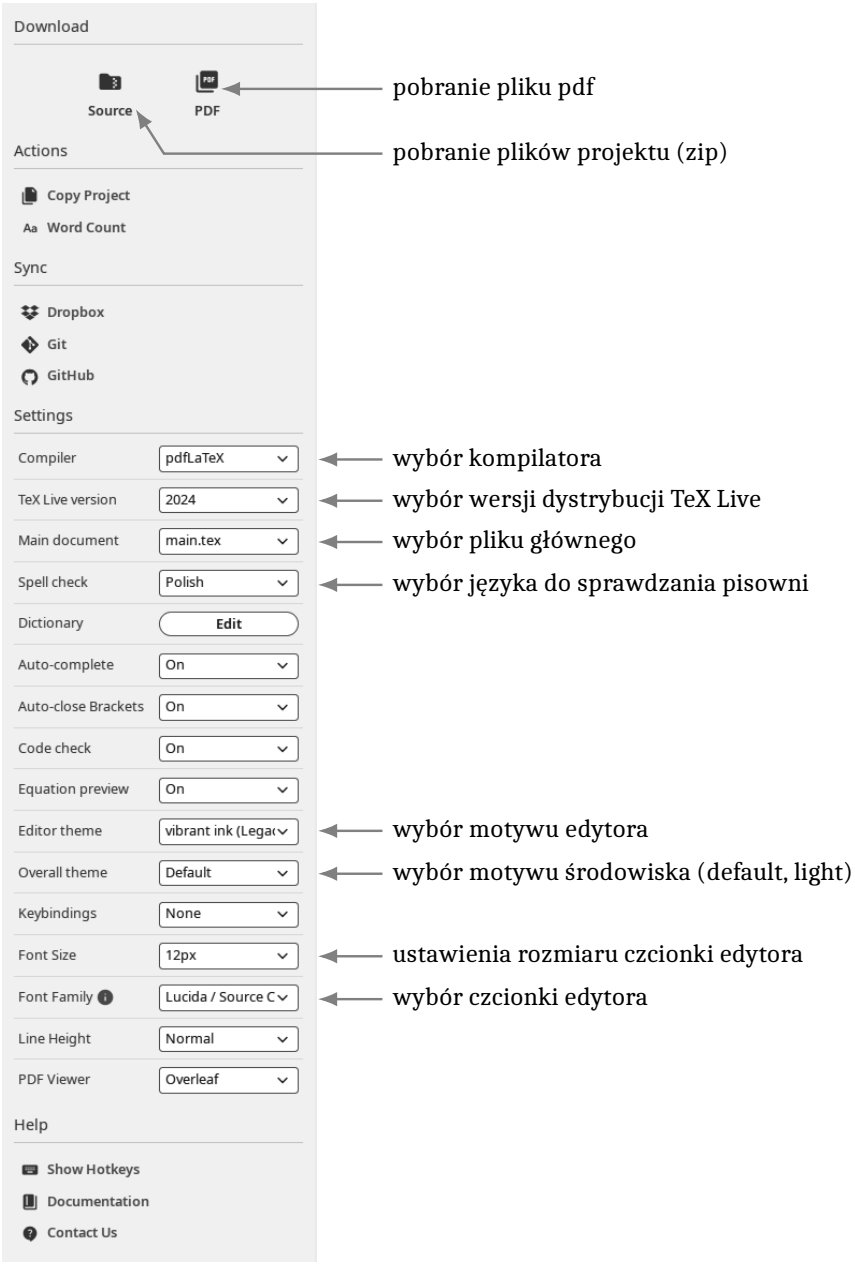
Zaczynamy od zastąpienia zawartości pliku `main.tex` kodem źródłowym widocznym na listingu 1.1. Do kompletu potrzebujemy dodać jeszcze nasz plik z rysunkiem. Klikamy przycisk `Upload` (zob. rys. 1.3) i dodajemy plik z grafiką do naszego projektu (dla testów możesz użyć dowolnego rysunku, ale musisz wstawić właściwą nazwę pliku do polecenia **`\includegraphics`**). Teraz możemy skompilować nasz dokument. Efekt końcowy naszych działań widoczny jest na rys. 1.4.



Rys. 1.4. Przykład z listingu 1.1 umieszczony w środowisku Overleaf

Na zakończenie tego rozdziału chciałbym zwrócić uwagę na kilka ważnych kwestii związanych z pracą w środowisku Overleaf. Panel po lewej stronie jest podzielony na dwie części. W górnej dostępna jest informacja dotycząca plików wchodzących w skład projektu, a w dolnej – struktury bieżącego pliku (spróbuj dodać drugą sekcję i zobacz, co się zmieni w panelu). Jeżeli chcemy zwiększyć obszar roboczy (edytor, podgląd pliku `pdf`), to możemy ukryć panel,

przeciągając w lewo jego lewą krawędź lub klikając szary przycisk dostępny w połowie krawędzi, służący także do jego ponownego wyświetlania.



Rys. 1.5. Główne menu środowiska Overleaf

Przycisk `Menu` (w lewym górnym rogu) pozwala wyświetlić główne menu środowiska, pokazane na rys. 1.5. Możemy w nim m.in.:

- pobrać pliki źródłowe projektu (w formie archiwum `zip`) lub tylko skompilowany dokument w formacie `pdf`;
- wskazać główny dokument projektu;
- wybrać wersję dystrybucji TeX Live – niektóre pakiety używane w książce wymagają wybrania najnowszej;
- wskazać język, który ma być używany do sprawdzania pisowni w kodzie źródłowym;
- wybrać schemat kolorystyczny dla edytora – przetestuj ciemne motywy, mniej męczą oczy;
- wybrać schemat kolorystyczny dla środowiska pracy – pokazane na rysunkach zrzuty ekranu wykonano po wyborze opcji `Light`;
- ustawić czcionkę używaną w edytorze i jej rozmiar;
- otworzyć dokumentację środowiska Overleaf, gdzie znajdziemy dużo więcej informacji na temat Overleaf, jak również na temat samego LaTeX-a.

Na etapie kompilacji generowanych jest mnóstwo informacji związanych z przetwarzanymi plikami i używanymi pakietami. Można je wyświetlić, klikając przycisk dostępu do logów i plików tymczasowych (zob. rys. 1.3). Jeżeli kompilacja kończy się bez błędu, to w zasadzie nie ma powodów, by tam zaglądać (chyba że chcemy pobrać pliki tymczasowe – zob. podrozdział 15.1). Jeżeli w kodzie znajdują się błędy, to nad ikoną przycisku na czerwonym tle wyświetlona zostanie informacja o ich liczbie. Oznacza to, że plik `pdf` nie został wygenerowany poprawnie. Należy zajrzeć do logów, przeczytać komunikaty o błędach, poprawić plik źródłowy i spróbować ponownie skompilować dokument. W ramach ćwiczenia spróbuj usunąć znak `$` na końcu linii 35 i skompiluj dokument, a następnie sprawdź, jaki komunikat o błędzie wygenerował kompilator.

Przycisk powrotu na poziom projektów (zob. rys. 1.3) umożliwia opuszczenie bieżącego projektu i przejście do listy wszystkich pozostałych. Z tego poziomu możemy łatwo dodać kolejny projekt, usunąć projekt, pobrać sam plik `pdf` lub kompletny projekt w postaci archiwum `zip`. Mamy też dostęp do szablonów dokumentów, z których można korzystać. Więcej na temat pracy z Overleaf znajdziesz w podrozdziale 15.1.

Rozdział 2

Wprowadzamy i formatujemy tekst

W rozdziale 1 przedstawiony został kompletny dokument w LaTeX-u. Będziemy go używać w większości rozdziałów jako punkt wyjścia do tworzenia kolejnych dokumentów. Kawałki kodu źródłowego, które będą prezentowane, w większości przypadków będą zawierały tylko fragmenty związane z omawianymi pojęciami, a nie kompletny kod źródłowy dokumentu. W tym rozdziale skupimy się na formatowaniu tekstu. Będziemy mówić m.in. o tym, jak zmienić fragment tekstu, zaczynając od rozmiaru czcionki, przez zmianę jego koloru, a kończąc na możliwościach sterowania rozmieszczeniem tekstu w poszczególnych liniach akapitu. Wyjaśnię, jak poprawnie uzyskać znaki takie jak cudzysłowy w języku polskim i angielskim, półpauzy, pauzy itp. Szczególnie ważny jest podrozdział 2.1. Przedstawię w nim znaki o specjalnym znaczeniu w kodzie źródłowym LaTeX-a. Zrozumienie ich roli i poprawne posługiwanie się nimi pozwala uniknąć wielu błędów kompilacji, a co za tym idzie nauka LaTeX-a staje się mniej stresująca.

2.1 Znaki specjalne

Dziesięć znaków ma w LaTeX-u znaczenie specjalne i ich bezpośrednie wpisanie z klawiatury nie powoduje, że pojawią się one w tekście. Mało tego, zazwyczaj skończy się to błędem kompilacji, a plik `pdf` będzie niepoprawny lub wcale nie zostanie wygenerowany. Niektóre z nich już pojawiły się w naszym pierwszym przykładzie, np. ukośnik wsteczny lub nawiasy klamrowe. Pełną ich listę przedstawiono w tabeli 2.1.

Znakiem szczególnym jest ukośnik wsteczny (\). Jak już wiesz, rozpoczyna on każde polecenie LaTeX-a. Jest on również wykorzystywany jako „znak ucieczki” dla pozostałych znaków specjalnych. Innymi słowy, jeżeli chcemy

Tabela 2.1. Lista znaków specjalnych

`$`	`&`	`%`	`#`	`_`	`{`	`}`	`~`	`^`	`\`

„uciec” od specjalnego znaczenia znaku `$`, wystarczy zapisać go jako `\$`, co w efekcie końcowym spowoduje wydrukowanie znaku `$` w dokumencie. Ukośnik wsteczny nie służy jako znak ucieczki dla siebie samego. Jeżeli chcemy uzyskać znak ukośnika wstecznego w trybie tekstowym, używamy polecenia **`\textbackslash`**, a w trybie matematycznym **`\backslash`**.

Nawiasy klamrowe (`{ }`) zostały omówione w rozdziale 1 jako „opakowanie” dla parametrów przekazywanych dla poleceń. Służą one również do ograniczania zakresu działania poleceń, które mają charakter „przełączników”, tj. działają od miejsca ich użycia do końca środowiska, w którym ich użyto (w szczególności może to być koniec dokumentu). Na przykład w kodzie z listingu 1.1 użyliśmy polecenie **`\textit`**`{kwadrat ...}`, aby za pomocą kursywy złożyć treść twierdzenia Pitagorasa (patrz rys. 1.1). LaTeX oferuje również polecenie/przełącznik **`\itshape`**, które również pozwala uzyskać tekst złożony kursywą. Polecenie to nie przyjmuje żadnych parametrów i użyte na przykład przed słowem „kwadrat” spowoduje, że nie tylko treść twierdzenia Pitagorasa, ale cała reszta dokumentu będzie złożona kursywą. W takim przypadku można ograniczyć zakres działania przełącznika za pomocą nawiasów klamrowych: `{`**`\itshape`**` kwadrat ...}`. Warto jeszcze wspomnieć o nawiasach klamrowych w trybie matematycznym. Jeżeli chcemy uzyskać zapis $A = \{a, b, c\}$, to poprawny kod źródłowy ma postać `$A = \{a,b,c\}$`. Jeżeli zapomnimy o ukośnikach wstecznych: `$A = {a,b,c}$`, to dostaniemy w efekcie $A = a, b, c$.

Znak dolara `$` pozwala nam przejść w tryb matematyczny wewnątrzakapitowy lub opuścić ten tryb, tj. służy do umieszczania wyrażeń (wzorów) matematycznych wewnątrz tekstu. W TeX-u funkcjonuje również podwójny znak dolara `$$`, który powoduje przejście lub opuszczenie trybu matematycznego eksponowanego (wzór składany w oddzielnej linii, najczęściej wyśrodkowany). Składni tej nie powinniśmy używać w LaTeX-u. Wzory eksponowane uzyskujemy, stosując odpowiednie środowiska – zob. rozdział 6.

Znaki podkreślenia (`_`) i daszek (`^`) służą w trybie matematycznym do przejścia odpowiednio do indeksu dolnego i górnego. Jeżeli chcemy uzyskać x_2, to wpisujemy w kodzie `$x_2$`, a jeśli x^2, to `$x^2$`. Przypominam, że działa to tylko na jeden kolejny znak (również generowany przez polecenie LaTeX-a). Jeżeli w indeksie dolnym lub górnym ma się znaleźć więcej znaków, to umieszczamy je w nawiasach klamrowych, np. kod `$x_{123}$` pozwoli uzyskać x_{123}. Oba symbole można połączyć w ramach jednego wzoru, np. x^3_{10} uzyskamy, pisząc `$x_{10}^{3}$` (drugi nawias klamrowy jest zbędny, bo mamy jeden znak,

dodano go dla poprawienia czytelności kodu). Symbole te można też zagnieżdżać, np. `$x_{y_2}$` pozwoli uzyskać x_{y_2}.

Znak tyldy (~) reprezentuje tzw. twardą (niełamliwą) spację. Użyty w kodzie źródłowym zamiast spacji zabrania łamania tekstu w danym miejscu. Na rys. 1.1 możesz znaleźć kilka przykładów użycia twardej spacji, np. `i~dlatego` gwarantuje nam, że linia nie zostanie złamana po `i`. Unikniemy w ten sposób tzw. „sierotki", czy bardzo krótkiego, zwłaszcza jednoznakowego słowa na końcu linii. Sierotki są błędem typograficznym. W innym przykładzie, `rys.~\ref{fig:trojkat:prostokatny}`, użycie twardej spacji gwarantuje nam, że łamanie nie nastąpi między `rys.` a numerem rysunku. Warto przyjąć za regułę, że przed poleceniem `\ref{}` zawsze wstawiamy twardą spację.

Znak procentu rozpoczyna komentarze w LaTeX-u. Przykłady jego użycia pokazano w naszym pierwszym dokumencie. Często możesz trafić na puste komentarze. Może być tak, że linia zaczyna się od znaku procentu i jest to jedyny znak w tej linii. Zabieg taki stosujemy, jeżeli chcemy dodać pustą linię dla zwiększenia czytelności kodu źródłowego, ale nie chcemy, by była to prawdziwa pusta linia, która oznacza koniec akapitu. Może się zdarzyć, że zobaczysz znak procentu na końcu linii. Zabieg ten służy wyeliminowaniu wstawienia przez kompilator jakiekolwiek znaku między dwoma liniami (nawet spacji lub znaku łamania linii).

Znak ampersand (&) służy jako separator komórek w wierszach struktur tabelarycznych. Więcej na jego temat powiem w rozdziale 5. Kratka (#) służy między innymi do odwoływania się do parametrów poleceń, gdy je definiujemy. Więcej na ten temat powiem w podrozdziale 14.5.

2.2 Wybór czcionki

Domyślna czcionka dla dokumentów LaTeX-a zdefiniowana jest jako *medium series, serif family, upright shape, 10 pt size*. Wstawione w nazwie przecinki mają za zadanie zwrócić uwagę na cztery składowe opisu czcionki: serię, rodzinę, kształt i rozmiar, które omówię w dalszej części podrozdziału. Zacznijmy od ostatniego elementu. Jak już pokazałem w przykładzie na rys. 1.1, bazowy rozmiar czcionki możemy zmienić, podając go jako parametr opcjonalny polecenia **`\documentclass`**. Rozmiary czcionki w różnych częściach dokumentu (tytuły rozdziałów, podrozdziałów, tekst w indeksie górnym i dolnym itd.) są wyliczane proporcjonalnie do ustalonego rozmiaru bazowego. W tabeli 2.2 podano polecenia, które pozwalają zmienić rozmiar czcionki dla fragmentu tekstu. Wszystkie te polecenia działają jako przełączniki.

Tabela 2.2. Przełączniki do zmiany wielkości czcionki

Kod źródłowy	Efekt
{\tiny tiny}	tiny
{\scriptsize script size}	script size
{\footnotesize footnote size}	footnote size
{\small small}	small
{\normalsize normal size}	normal size
{\large large}	large
{\Large Large}	Large
{\LARGE LARGE}	LARGE
{\huge huge}	huge
{\Huge Huge}	Huge

Przedstawione w tabeli 2.2 polecenia dadzą nieco inny efekt, gdy zmienimy czcionkę bazową (ta książka jest składana czcionką bazową o rozmiarze 11 pt). W tabeli 2.3 podano wymiary czcionek w klasach standardowych zależnie od czcionki bazowej [40].

Tabela 2.3. Rozmiar czcionek w klasach standardowych

Polecenie	10pt	11pt	12pt
\tiny	5pt	6pt	7pt
\scriptsize	7pt	8pt	8pt
\footnotesize	8pt	9pt	10pt
\small	9pt	10pt	11pt
\normalsize	10pt	11pt	12pt
\large	12pt	12pt	14pt
\Large	14pt	14pt	17pt
\LARGE	17pt	17pt	20pt
\huge	20pt	20pt	25pt
\Huge	25pt	25pt	25pt

W większości przypadków polecenia te są wystarczające do zarządzania wielkością czcionki. Jeżeli zajdzie taka konieczność, to można zdefiniować arbitralny rozmiar czcionki (nie będzie się skalował względem rozmiaru bazowego). Realizowane jest to przez sekwencję poleceń `\fontsize{}{}\selectfont`. Pierwszy parametr polecenia `\fontsize` definiuje rozmiar czcionki, a drugi

interlinię. Polecenia `\fontsize {30pt}{24pt}\selectfont` zadziałają jak przełącznik ustawiający wysokość tekstu na 30 pt, a interlinię na 24 pt, np. polecenie `{\fontsize{30pt}{24pt} \selectfont duża czcionka}` wygeneruje tekst:

duża czcionka

Współcześnie, gdy używamy terminu czcionka, mamy prawie zawsze na myśli krój pisma. Z perspektywy historii słowa drukowanego czcionka to po prostu obraz pojedynczego znaku drukowanego, czyli np. pojedyncza litera, cyfra lub znak interpunkcyjny. Kompletny zestaw znaków o jednolitym charakterze graficznym nazywamy krojem pisma, np. Courier, Helvetica, Times Roman. W dalszej części pozostaniemy przy współczesnej nomenklaturze i pod terminem czcionka będziemy rozumieć krój pisma. Wszystkie warianty czcionki funkcjonujące w ramach jednej nazwy nazywamy rodziną czcionki. Czcionki w rodzinie mogą różnić się grubością, pochyleniem, wielkością odstępu między znakami itp. LaTeX grupuje czcionki lub ich rodziny w trzy kategorie: `serif` (`roman`), `sans serif` i `monospaced` (`teletype`). Polecenia i odpowiadające im przełączniki, które pozwalają zmienić rodzinę czcionek, przedstawiono w tabeli 2.4.

Tabela 2.4. Polecenia i przełączniki do zmiany rodziny czcionki

Rodzina	**Kod źródłowy**	**Przełącznik**	**Efekt**
serif	`\textrm{serif}`	`\rmfamily`	serif
sans serif	`\textsf{sans serif}`	`\sffamily`	sans serif
monospaced	`\texttt{monospaced}`	`\ttfamily`	monospaced

W rodzinie `serif` mamy czcionki szeryfowe, a w rodzinie `sans serif` bezszeryfowe. Szeryfy to ozdobne, krótkie kreski stosowane w wielu czcionkach do zwiększenia ich dekoracyjności. Porównaj np. wygląd liter w słowie „serif” w ostatniej kolumnie tabeli 2.4. Trzecia kategoria to czcionki o stałej szerokości znaku (o dwóch poprzednich mówimy, że są proporcjonalne). Czcionki o stałej szerokości znaków są używane w tej książce do prezentacji fragmentów kodu źródłowego. Są też powszechnie stosowane w edytorach kodu źródłowego.

Jeżeli w danym miejscu możesz użyć polecenia bądź odpowiadającego mu przełącznika, to wybierz polecenie. Polecenia lepiej radzą sobie z odstępami po fragmencie, do którego stosujesz wybraną czcionkę.

Czcionki należące do jednej rodziny mogą mieć różne style. W LaTeX-u styl czcionki definiuje się jako kombinację serii i kształtu. Seria definiuje grubość czcionki i wielkość odstępów między znakami. Kształt określa pochylenie tekstu

lub wykorzystanie kapitalików. Polecenia i przełączniki do zmiany serii czcionki przedstawiono w tabeli 2.5, a do zmiany kształtu w tabeli 2.6.

Tabela 2.5. Polecenia i przełączniki do zmiany serii czcionki

Seria	Kod źródłowy	Przełącznik	Efekt
boldface	`\textbf{bold}`	`\bfseries`	**bold**
medium	`\textmd{medium}`	`\mdseries`	medium

Tabela 2.6. Polecenia i przełączniki do zmiany kształtu czcionki

Kształt	Kod źródłowy	Przełącznik	Efekt
prosta	`\textup{upright}`	`\upshape`	upright
kursywa	`\textit{italic}`	`\itshape`	*italic*
pochyła	`\textsl{slanted}`	`\slshape`	*slanted*
kapitaliki	`\textsc{small caps}`	`\scshape`	SMALL CAPS

Możesz natrafić na dwuliterowe polecenia związane z wyborem serii lub kształtu czcionki, np. `\bf`, `\it` itp. Są one uważane za przestarzałe i należy ich unikać.

Polecenia dotyczące wyboru rozmiaru, serii i kształtu można ze sobą łączyć, np. `\textbf{\textit{\Large ...}}` da w efekcie tekst, który jest jednocześnie pogrubiony, złożony kursywą i powiększony względem tekstu bazowego. Warto tutaj zwrócić uwagę na poprawne domknięcie wszystkich nawiasów klamrowych. Jest to szczególnie ważne, gdy zagnieżdżonych poleceń jest więcej. Na listingu 2.1 pokazano wybrane kombinacje tych poleceń oraz odpowiadających im przełączników. Poniżej kodu źródłowego zamieszczono wynik kompilacji. Porównaj efekty zmiany położenia nawiasów klamrowych oraz kolejności użycia poszczególnych poleceń.

Listing 2.1. Przykłady łączenia poleceń formatujących tekst

```
\textbf{\textit{\Large Twierdzenie Pitagorasa}}\\
\textbf{\textit{\Large Twierdzenie} Pitagorasa}\\
\textbf{\Large\textit{Twierdzenie} Pitagorasa}\\
\textit{\textbf{\Large Twierdzenie Pitagorasa}}\\
\textit{\textbf{\Large Twierdzenie} Pitagorasa}\\
\textit{\Large\textbf{Twierdzenie} Pitagorasa}\\
{\bfseries\itshape\Large Twierdzenie Pitagorasa}\\
```

```
{\itshape\bfseries\Large Twierdzenie Pitagorasa}\\
{\Large\bfseries\itshape Twierdzenie Pitagorasa}
```

Twierdzenie Pitagorasa
***Twierdzenie* Pitagorasa**
***Twierdzenie* Pitagorasa**
Twierdzenie Pitagorasa
Twierdzenie* *Pitagorasa
Twierdzenie* *Pitagorasa
Twierdzenie Pitagorasa
Twierdzenie Pitagorasa
Twierdzenie Pitagorasa

Przypominam, że użycie nawiasów klamrowych przy przełącznikach służy jedynie ograniczeniu zakresu ich działania. Nie jest ono niezbędne. Jeżeli użyjemy na przykład przełącznika `\itshape`, to od tego miejsca kursywa będzie stosowana do kolejnych fragmentów dokumentu. Powrót do ustawień standardowych uzyskamy za pomocą przełącznika `\normalfont`.

Najwygodniejszą formą zarządzania wyglądem całego dokumentu jest użycie pakietu zmieniającego całą kolekcję czcionek w tym dokumencie. Brak jakiegokolwiek pakietu oznacza użycie standardowej czcionki Computer Modern. Na rysunku 2.1 pokazano pangramy (serif, sans serif, monospaced, kursywa, pochyła) i przykład wzoru matematycznego dla trzech kompletnych kolekcji czcionek. Rodzina Latin Modern jest w zasadzie poprawioną wersją czcionki standardowej, więc powinna być używana, gdy chcemy pozostać przy domyślnym wyglądzie. Na prawo od nazwy czcionki podano nazwę pakietu, który należy dołączyć do pliku, aby uzyskać pokazany efekt. Pakiety mogą być dołączane z opcjami, które wpływają na wygląd czcionek, np. pakiet `kpfonts` z opcją `light`. Listę dostępnych opcji znajdziemy zawsze w dokumentacji danego pakietu.

Dołączenie pakietu `lmodern` lub `kpfonts` nie oznacza, że nagle mamy dostępne czcionki, których nie mogliśmy używać wcześniej. Możemy użyć tych czcionek bez dołączania wymienionych pakietów, ale sami musielibyśmy redefiniować pewne polecenia. Rolą pakietów jest m.in. redefinicja trzech poleceń, ustawiających domyślną czcionkę szeryfową, bezszeryfową i o stałej szerokości (zob. listing 2.2).

Listing 2.2. Redefinicje poleceń ustawiających domyślną czcionkę szeryfową, bezszeryfową i o stałej szerokości

```
\renewcommand{\rmdefault}{...}
\renewcommand{\sfdefault}{...}
\renewcommand{\ttdefault}{...}
```

Computer Modern —

Pójdź, kińże tę chmurność w głąb flaszy! 1234567890
Pójdź, kińże tę chmurność w głąb flaszy! 1234567890
Pójdź, kińże tę chmurność w głąb flaszy! 1234567890
Pójdź, kińże tę chmurność w głąb flaszy! 1234567890
Pójdź, kińże tę chmurność w głąb flaszy! 1234567890

$$\Delta f(X_0) = \sum_{i=1}^{n} \frac{\partial f(X_0)}{\partial x_i} \Delta x_i + \varepsilon\rho.$$

Latin Modern `\usepackage{lmodern}`

Pójdź, kińże tę chmurność w głąb flaszy! 1234567890
Pójdź, kińże tę chmurność w głąb flaszy! 1234567890
Pójdź, kińże tę chmurność w głąb flaszy! 1234567890
Pójdź, kińże tę chmurność w głąb flaszy! 1234567890
Pójdź, kińże tę chmurność w głąb flaszy! 1234567890

$$\Delta f(X_0) = \sum_{i=1}^{n} \frac{\partial f(X_0)}{\partial x_i} \Delta x_i + \varepsilon\rho.$$

Kepler Fonts `\usepackage{kpfonts}`

Pójdź, kińże tę chmurność w głąb flaszy! 1234567890
Pójdź, kińże tę chmurność w głąb flaszy! 1234567890
Pójdź, kińże tę chmurność w głąb flaszy! 1234567890
Pójdź, kińże tę chmurność w głąb flaszy! 1234567890
Pójdź, kińże tę chmurność w głąb flaszy! 1234567890

$$\Delta f(X_0) = \sum_{i=1}^{n} \frac{\partial f(X_0)}{\partial x_i} \Delta x_i + \varepsilon\rho.$$

Rys. 2.1. Pangramy czcionek: serif, sans serif, monospaced, kursywa, pochyła

Jeżeli chcemy sami ustawić czcionkę, domyślnie dla całego dokumentu lub dla wybranego fragmentu, to potrzebujemy jej nazwy kodowej. Zestawienie kodów dla wybranych czcionek przedstawiono w tabeli 2.7. Kolejne grupy w tabeli (oddzielone poziomą linią) to czcionki szeryfowe, bezszeryfowe, o stałej szerokości i czcionka kaligraficzna. Dla przykładu ustawienia podstawowych czcionek realizowane przez pakiet `lmodern` pokazano na listingu 2.3.

Listing 2.3. Redefinicje poleceń ustawiających domyślne czcionki w pakiecie `lmodern`

```
\renewcommand{\rmdefault}{lmr}
\renewcommand{\sfdefault}{lmss}
\renewcommand{\ttdefault}{lmtt}
```

Tabela 2.7. Nazwy kodowe czcionek

Czcionka	Pakiet	Kod czcionki
Computer Modern Roman		cmr
Kp-Fonts	kpfonts	jkp
Kerkis	kerkis	mak
Latin Modern Roman	lmodern	lmr
New Century Schoolbook	newcent	pnc
Palatino	newpx, newpxtext	zpltlf
TEX Gyre Bonum	tgbonum	qbk
TEX Gyre Pagella	tgpagella	qpl
TEX Gyre Schola	tgschola	qcs
TEX Gyre Termes	tgtermes	qtm
Times Roman	newtx, newtxtext	ptm
Utopia / Fourier	utopia, fourier	put
Arev	arev	fav
Computer Modern Sans Serif		cmss
Helvetica	helvet	phv
Kerkis Sans Serif	kerkis	maksf
Kp-Fonts Sans Serif	kpfonts	jkpss
Kurier	kurier	kurier
Latin Modern Sans Serif	lmodern	lmss
TEX Gyre Adventor	tgadventor	qag
TEX Gyre Heros	tgheros	qhv
Bera Mono	beramono	fvm
Computer Modern Typewriter		cmtt
Courier	courier	pcr
Inconsolata	inconsolata	zi4
Kp-Fonts Typewriter	kpfonts	jkptt
Latin Modern Typewriter	lmodern	lmtt
TEX Gyre Cursor	tgcursor	qcr
Times Roman Typewriter	newtx, newtxtext	ntxtt
TEX Gyre Chorus	tgchorus	qzc

Dodając trzy linie z listingu 2.3 do naszej preambuły (bez dołączania pakietu `lmodern`), wykonalibyśmy częściowo pracę za ten pakiet, tj. w odniesieniu do samego tekstu osiągnęlibyśmy ten sam efekt. Sam pakiet robi dużo więcej, bo definiuje jeszcze sporo ustawień dla składu matematyki.

Jeżeli znamy kody czcionek, to możemy łatwo zastosować wybraną czcionkę do fragmentu tekstu. Realizujemy to poprzez sekwencję poleceń

`\fontfamily{...}\selectfont`. Przykład wykorzystania tych poleceń pokazano na listingu 2.4. Zakres zastosowania danej czcionki możemy ograniczyć, stosując na przykład nawiasy klamrowe. W przykładzie z listingu nie ograniczono zastosowania czcionki Kp-Fonts Sans Serif, więc po zdaniu złożonym za pomocą TEX Gyre Chorus znowu używamy Kp-Fonts Sans Serif.

Listing 2.4. Przykład zastosowania poleceń `\fontfamily` i `\selectfont`

```
\fontfamily{jkpss}\selectfont Twierdzenie Pitagorasa jest zdecydowanie
najpopularniejszym z tych, z jakimi spotykamy się w szkole.
{\fontfamily{qzc}\selectfont Jego nazwa została połączona z nazwiskiem
wielkiego filozofa greckiego żyjącego od ok. 572 roku p.n.e. do
ok. 497 roku p.n.e.} Sam Pitagoras raczej nie był twórcą tego
twierdzenia, które znane było długo  przed jego narodzinami.
```

Twierdzenie Pitagorasa jest zdecydowanie najpopularniejszym z tych, z jakimi spotykamy się w szkole. *Jego nazwa została połączona z nazwiskiem wielkiego filozofa greckiego żyjącego od ok. 572 roku p.n.e. do ok. 497 roku p.n.e.* Sam Pitagoras raczej nie był twórcą tego twierdzenia, które znane było długo przed jego narodzinami.

LATEX oferuje sporo pakietów ustawiających określoną rodzinę czcionek (w przykładzie w rozdziale pierwszym użyliśmy pakietu `newtx` – zob. listing 1.1). Nie każdy pakiet oferuje wsparcie dla matematyki. Jeżeli skład wzorów matematycznych jest dla nas ważny, to konieczne jest dobranie drugiego pakietu, który dostarczy pasujące czcionki dla matematyki. Poza tym może zajść konieczność dobrania pakietów dla czcionki szeryfowej, bezszeryfowej i stałej szerokości, które będą do siebie pasowały. Katalog czcionek LATEX-a można znaleźć pod adresem `https://tug.org/FontCatalogue/`, przy czym nie każda czcionka, na którą trafimy, wspiera polskie litery. Nazwy czcionek są odnośnikami. Po wyborze interesującej nas czcionki zobaczymy, jakie pakiety trzeba włączyć do dokumentu i jak jej używamy.

Na rysunku 2.2 przedstawiono pangramy wybranych czcionek szeryfowych. W tabeli 2.7 mamy jeszcze czcionki szeryfowe: TEX Gyre Bonum – zamiennik czcionki Bookman, TEX Gyre Pagella – zamiennik czcionki Palatino, TEX Gyre Schola – zamiennik czcionki New Century Schoolbook i TEX Gyre Termes – zamiennik czcionki Times Roman.

Pakiety czcionek z rys. 2.2 ustawiają odpowiednią czcionkę szeryfową (i ewentualnie wsparcie dla matematyki), ale nie oznacza to, że pozostałe czcionki muszą być w wersjach domyślnych. Dla przykładu: użycie pakietu `newtx` (czyli m.in. pakietu `newtxtext`) powoduje załadowanie również pakietu `helvet` (odpowiednio wyskalowanego), czyli ustawienie czcionki Helvetica jako domyślnej czcionki bezszeryfowej. Skalowanie

Kerkis (Bookman) `\usepackage{kmath,kerkis}`

Pójdź, kińże tę chmurność w głąb flaszy! 1234567890

$$\Delta f(X_0) = \sum_{i=1}^{n} \frac{\partial f(X_0)}{\partial x_i} \Delta x_i + \varepsilon\rho.$$

New Century Schoolbook `\usepackage{newcent,fouriernc}`

Pójdź, kińże tę chmurność w głąb flaszy! 1234567890

$$\Delta f(X_0) = \sum_{i=1}^{n} \frac{\partial f(X_0)}{\partial x_i} \Delta x_i + \varepsilon\rho.$$

Palatino `\usepackage{newpx}`

Pójdź, kińże tę chmurność w głąb flaszy! 1234567890

$$\Delta f(X_0) = \sum_{i=1}^{n} \frac{\partial f(X_0)}{\partial x_i} \Delta x_i + \varepsilon\rho.$$

Times Roman `\usepackage{newtx}`

Pójdź, kińże tę chmurność w głąb flaszy! 1234567890

$$\Delta f(X_0) = \sum_{i=1}^{n} \frac{\partial f(X_0)}{\partial x_i} \Delta x_i + \varepsilon\rho.$$

Utopia / Fourier `\usepackage{fourier}`

Pójdź, kińże tę chmurność w głąb flaszy! 1234567890

$$\Delta f(X_0) = \sum_{i=1}^{n} \frac{\partial f(X_0)}{\partial x_i} \Delta x_i + \varepsilon\rho.$$

Rys. 2.2. Pangramy wybranych czcionek szeryfowych

pakietu realizowane jest przez parametr opcjonalny polecenia `\usepackage`, np. `\usepackage[scaled=0.95]{helvet}`. Używa się go, gdy czcionka bezszeryfowa wydaje się za duża przy używanej obok czcionce szeryfowej. Z kolei pakiety `kerkis` i `newcent` ustawiają czcionkę Courier jako domyślną czcionkę o stałej szerokości znaków. W przypadku pakietu `kerkis` ważne jest, aby wcześniej załadować pakiet `kmath`.

Na rysunku 2.3 przedstawiono pangramy wybranych czcionek bezszeryfowych. Dodatkowo czcionkę TEX Gyre Heros możemy traktować jako zamiennik dla czcionki Helvetica. Dodanie pakietu `arev` (czyli m.in. `arevtext`) skutkuje ustawieniem zarówno `\rmdefault`, jak i `\sfdefault` na czcionkę Arev. W efekcie dokument jest składany domyślnie czcionką bezszeryfową. Dodatkowo jako czcionka o stałej szerokości ustawiana jest Bera Mono.

Arev `\usepackage{arev}`

Pójdź, kińże tę chmurność w głąb flaszy! 1234567890

$$\Delta f(X_0) = \sum_{i=1}^{n} \frac{\partial f(X_0)}{\partial x_i} \Delta x_i + \varepsilon\rho.$$

Helvetica `\usepackage{helvet,sfmath}`

Pójdź, kińże tę chmurność w głąb flaszy! 1234567890

$$\Delta f(X_0) = \sum_{i=1}^{n} \frac{\partial f(X_0)}{\partial x_i} \Delta x_i + \varepsilon\rho.$$

Kurier `\usepackage[math]{kurier}`

Pójdź, kińże tę chmurność w głąb flaszy! 1234567890

$$\Delta f(X_0) = \sum_{i=1}^{n} \frac{\partial f(X_0)}{\partial x_i} \Delta x_i + \varepsilon\rho.$$

Rys. 2.3. Pangramy wybranych czcionek bezszeryfowych

Jeżeli chcemy używać w dokumencie jako domyślnej czcionki bezszeryfowej, to możemy ustawić to w preambule. Realizujemy to przez polecenie `\renewcommand{\familydefault}{\sfdefault}`. Jaka będzie to czcionka bezszeryfowa, zależy od załadowanego pakietu. Podobnie możemy ustawić jako domyślną dla dokumentu czcionkę o stałej szerokości znaków.

Na rysunku 2.4 przedstawiono pangramy wybranych czcionek o stałej szerokości, a na rys. 2.5 przykłady czcionek kaligraficznych. Możemy ich użyć przy składaniu eleganckiego listu, zaproszenia, prezentacji itp. Nie każdy pakiet modyfikuje ustawienia czcionek. Dla przykładu pakiet `miama` dodaje polecenie `\fmmfamily` i tekst złożony kaligraficznie uzyskujemy z użyciem tego polecenia, np. `{\fmmfamily\Large Miama Nueva}` daje *Miama Nueva*.

Bera Mono `\usepackage{beramono}`

Pójdź, kińże tę chmurność w głąb flaszy! 1234567890

Courier `\usepackage{courier}`

Pójdź, kińże tę chmurność w głąb flaszy! 1234567890

Inconsolata `\usepackage{inconsolata}`

Pójdź, kińże tę chmurność w głąb flaszy! 1234567890

Rys. 2.4. Pangramy wybranych czcionek o stałej szerokości

TEX Gyre Chorus `\usepackage{tgchorus}`

Pójdź, kiń że tę chmurność w głąb flaszy! 1234567890

LobsterTwo `\usepackage{LobsterTwo}`

Pójdź, kiń że tę chmurność w głąb flaszy! 1234567890

Miama Nueva `\usepackage{miama}`

Pójdź, kiń że tę chmurność w głąb flaszy! 1234567890

Rys. 2.5. Pangramy wybranych czcionek kaligraficznych

Co się stanie, jeśli dodamy kilka pakietów z czcionkami? To zależy od tego, co te pakiety ustawiają. Jeżeli do pakietu `newtx`, dodamy pakiet `beramono`, to dostaniemy całkiem ciekawy zestaw Times Roman + Helvetica + Bera Mono odpowiednio dla czcionki szeryfowej, bezszeryfowej i o stałej szerokości. Z drugiej strony, jeśli umieścimy razem pakiety, które ustawiają te same czcionki, to później dołączony pakiet nadpisze ustawienia wcześniej dołączanego pakietu. Dla przykładu: jeśli dołączymy pakiety `kpfonts` i `beramono` (w tej kolejności), to dostaniemy wszystkie ustawienia wprowadzone przez pakiet `kpfonts` z wyjątkiem czcionki o stałej szerokości znaku, którą będzie Bera Mono – te ustawienia zostaną nadpisane, inne nie, bo pakiet `beramono` ustawia tylko czcionkę o stałej szerokości znaku. Jeżeli zmienimy kolejność, to w ogóle nie zauważymy efektu dodania pakietu `beramono`, bo jego ustawienia zostaną nadpisane przez `kpfonts`.

Poza wyborem odpowiedniej czcionki możemy skorzystać z poleceń, które pozwalają nam dodać efekty wyróżniające fragment tekstu. Polecenie `\underline{}` pozwala podkreślić fragment tekstu podany jako parametr. Podstawowe ograniczenie tego polecenia to brak możliwości podkreślenia tekstu obejmującego więcej niż jedną linię. Rozwiązaniem jest użycie pakietu `ulem`, który oferuje szereg możliwości wyróżniania tekstu i nie ogranicza zakresu stosowania swoich poleceń do pojedynczych linii. Polecenia z tego pakietu pokazano na listingu 2.5. Pakiet `ulem` domyślnie zamienia wyróżnienia (efekt polecenia `\emph`) z kursywy na podkreślenia. Efekt ten możemy zobaczyć na przykład w składzie bibliografii. Zachowanie to możemy wyłączyć, dołączając pakiet z opcją `normalem`.

```
\usepackage{ulem}              % lub
\usepackage[normalem]{ulem}
```

Listing 2.5. Polecenia pakietu `ulem` do wyróżniania tekstu

```
Pitagoras był \uline{greckim filozofem}\\
Pitagoras był \uuline{greckim filozofem}\\
Pitagoras był \uwave{greckim filozofem}\\
Pitagoras był \sout{greckim filozofem}\\
Pitagoras był \xout{greckim filozofem}\\
Pitagoras był \dashuline{greckim filozofem}\\
Pitagoras był \dotuline{greckim filozofem}
```

Pitagoras był greckim filozofem
Pitagoras był greckim filozofem
Pitagoras był greckim filozofem
Pitagoras był ~~greckim filozofem~~
Pitagoras był greckim filozofem
Pitagoras był greckim filozofem
Pitagoras był greckim filozofem

Z każdym pakietem związana jest dokumentacja, która szczegółowo opisuje jego zastosowanie i dostarczane polecenia. Dokumentację dla poszczególnych pakietów możemy znaleźć na stronie `https://www.ctan.org` – w oknie wyszukiwarki wpisujemy nazwę szukanego pakietu. Wyszukana podstrona pakietu powinna zawierać plik `pdf` z dokumentacją.

2.3 LaTeX na kolorowo

Podstawowym pakietem, który dodaje obsługę kolorów w dokumentach LaTeX-a, jest `color`. Pozwala on m.in. na skorzystanie z poleceń umożliwiających zmianę koloru tekstu lub tła, a także na definiowanie własnych kolorów. Dużo większe możliwości oferuje natomiast pakiet `xcolor` i na nim skupimy się w tym podrozdziale.

```
\usepackage{xcolor}
```

Pakiet ten dostarcza predefiniowane kolory: *black, blue, brown, cyan, darkgray, gray, green, lightgray, lime, magenta, olive, orange, pink, purple, red, teal, white, violet, yellow*. Możemy używać tych nazw w poleceniach zmieniających kolor tekstu lub tła. Podstawowym poleceniem jest `\textcolor{}{}`. Pierwszym jego parametrem jest nazwa koloru (lub definicja, jak zobaczymy

później), a drugim tekst, do którego zostanie on zastosowany. Podobny efekt możemy uzyskać za pomocą przełącznika `\color{}`. Tutaj jedynym parametrem jest nazwa koloru, którego chcemy użyć. Dla przykładu: użycie polecenia `\color{olive}` spowoduje, że od tego miejsca tekst będzie składany w kolorze oliwkowym. Podobny efekt uzyskamy za pomocą `\textcolor{olive}{...}`, ale w tym przypadku kolor oliwkowy będzie użyty tylko do tekstu w drugim nawiasie klamrowym.

Ofertę predefiniowanych kolorów możemy wzbogacić, dodając własne. Do ich definiowania służy polecenie `\definecolor{}{}{}`. Poszczególne parametry to nazwa koloru, model koloru i argumenty numeryczne. W tabeli 2.8 przedstawiono najpopularniejsze modele kolorów wspierane przez pakiet `xcolor` oraz zakres parametrów numerycznych używanych w poszczególnych modelach. Użyta w tabeli 2.8 notacja $[0;1]$ oznacza liczbę rzeczywistą z przedziału od 0 do 1, a 0..255 liczbę całkowitą z zakresu od 0 do 255. W przypadku modelu HTML podajemy 3 liczby szesnastkowe kodujące dany kolor. Przykłady definicji kolorów pokazano na listingu 2.6. Definicje takie umieszczamy w preambule dokumentu. W liniach 3–6 i 7–10 mamy definicje zbliżonych kolorów, ale określone z użyciem różnych modeli.

Tabela 2.8. Wybrane modele kolorów wspierane przez pakiet `xcolor`

Nazwa	**Kolory bazowe**	**Zakres parametrów**
rgb	red, green, blue	$([0;1],[0;1],[0;1])$
cmyk	cyan, magenta, yellow, black	$([0;1],[0;1],[0;1],[0;1])$
gray	gray	$[0;1]$
RGB	red, green, blue	$(0..255, 0..255, 0..255)$
HTML	RRGGBB	$000000, .., FFFFFF$

Listing 2.6. Przykłady definicji kolorów w pakiecie `xcolor`

```
1  \definecolor{mylightgray}{gray}{0.4}
2  \definecolor{mydarkgray}{gray}{0.8}
3  \definecolor{darkred1}{rgb}{0.7,0,0}
4  \definecolor{darkred2}{RGB}{179,0,0}
5  \definecolor{darkred3}{cmyk}{0,1,1,0.3}
6  \definecolor{darkred4}{HTML}{b30000}
7  \definecolor{darkgreen1}{rgb}{0,0.6,0}
8  \definecolor{darkgreen2}{RGB}{0,153,0}
9  \definecolor{darkgreen3}{cmyk}{1,0,1,0.4}
10 \definecolor{darkgreen4}{HTML}{009900}
```

Definicję koloru można podać bezpośrednio w miejscu jego wykorzystania, np. w poleceniu `\textcolor[RGB]{200,100,50}{...}` wskazujemy używany model kolorów i zamiast nazwy podajemy parametry numeryczne używanego koloru. Analogicznie możemy postąpić z przełącznikiem `\color`, a także innymi poleceniami, które jako parametr oczekują nazwy koloru.

Pakiet `xcolor` zawiera też konstrukcję umożliwiającą mieszanie kolorów, np. `blue!20!red` oznacza mieszaninę 20% koloru niebieskiego z 80% koloru czerwonego. Oczywiście możemy używać nie tylko predefiniowanych nazw kolorów. Brak drugiej nazwy oznacza, że zostanie użyty domyślnie kolor biały, zatem `blue!50` oznacza niebieski zmieszany pół na pół z białym. Konstrukcja mieszania może być rozszerzona na więcej kolorów, ale ze względu na sposób wyznaczania koloru finalnego wydaje się mało użyteczna. Zainteresowanych odsyłam do manuala pakietu `xcolor` [28]. Przykład użycia omawianej konstrukcji pokazano na listingu 2.7.

Listing 2.7. Przykład mieszania kolorów

```
Liczba różnych dowodów twierdzenia Pitagorasa jest duża.
\textcolor{black!80}{W \textit{Elementach} Euklidesa znajdziemy ich
aż osiem.} \textcolor{black!60}{Znana dzisiaj liczba dowodów jest
trzycyfrowa.}
```

Liczba różnych dowodów twierdzenia Pitagorasa jest duża. W *Elementach* Euklidesa znajdziemy ich aż osiem. Znana dzisiaj liczba dowodów jest trzycyfrowa.

Polecenie `\pagecolor{}` pozwala zmienić kolor tła bieżącej i kolejnych stron. Parametrem jest nazwa koloru. Polecenie `\nopagecolor` przywraca domyślne ustawienie z przezroczystym tłem. Jeżeli chcemy zmienić tło dla fragmentu tekstu, to możemy zastosować polecenie `\colorbox{}{}`. Pierwszym parametrem jest nazwa koloru, a drugim tekst, którego tło ma być zmienione. Istotnym ograniczeniem tego polecenia jest to, że pracuje ono w specjalnym trybie wierszowym (tryb tekstowy LR – od lewej do prawej). W trybie tym nie jest stosowane łamanie linii, czyli nasz tekst musi zmieścić się w pojedynczym wierszu (w przeciwnym wypadku wejdzie na prawy margines). W tym samym trybie pracuje polecenie `\fcolorbox{}{}{}`, które dodatkowo rysuje ramkę wokół wskazanego tekstu w podanym kolorze. Kolejne parametry oznaczają: kolor ramki, kolor tła i tekst umieszczony w ramce. Przykłady użycia poleceń `\colorbox` i `\fcolorbox` pokazano na listingu 2.8.

Listing 2.8. Przykłady zmiany tła dla fragmentu tekstu

```
\colorbox{lightgray}{Istnieje ponad sto geometrycznych
dowodów twierdzenia Pitagorasa.}

```

```
\fcolorbox{black!90}{lightgray!50}{Istnieje ponad sto
geometrycznych dowodów twierdzenia Pitagorasa.}

\fcolorbox{black!90}{lightgray}{\color{white}Istnieje
ponad sto geometrycznych dowodów twierdzenia Pitagorasa.}
```

Istnieje ponad sto geometrycznych dowodów twierdzenia Pitagorasa.

Istnieje ponad sto geometrycznych dowodów twierdzenia Pitagorasa.

Istnieje ponad sto geometrycznych dowodów twierdzenia Pitagorasa.

Jednym ze sposobów na obejście ograniczenia związanego z trybem wierszowym w poleceniach **\colorbox** i **\fcolorbox** jest użycie środowiska **minipage**. Służy ono do zdefiniowania obszaru, w którym tekst jest składany w domyślnym trybie akapitowym. To tak jak strona zagnieżdżona w zwykłej stronie dokumentu (może zawierać akapity tekstu, rysunki, tabele itd.). W najprostszej swojej wersji środowisko przyjmuje jeden parametr definiujący szerokość ministrony. Wysokość jest wyznaczana automatycznie na podstawie zawartości. Przykład użycia środowiska pokazano na listingu 2.9.

Listing 2.9. Przykład użycia ministrony wewnątrz ramki

```
\fcolorbox{black!90}{gray}{\color{white}
\begin{minipage}{7cm}
Istnieje ponad sto geometrycznych dowodów twierdzenia Pitagorasa.
\end{minipage}}
```

Istnieje ponad sto geometrycznych dowodów twierdzenia Pitagorasa.

2.4 Łamanie linii i stron na własną rękę

Często powtarzam swoim studentom, że LaTeX dobrze „wie", jak składać tekst, i powinniśmy go poprawiać tylko wtedy, gdy naprawdę wiemy, co robimy. Kompilator LaTeX-a stara się łamać tekst między liniami lub stronami, tak aby zapewnić wysoką jakość produktu końcowego. Czasami jednak nawet LaTeX jest bezradny w sytuacji, gdy napotyka tekst, którego ani nie może złamać, ani nie powinien przenosić w całości do kolejnej linii. Przeanalizujmy poniższy akapit.

LaTeX-a możemy pobrać na wszystkie popularne systemy operacyjne (Linux, Mac OS, Windows), na przykład dystrybucję TeX Live: https://www.latex-project.org/get/

Adres strony internetowej nie mieści się w drugiej linii. Użyte w kodzie polecenie zabrania kompilatorowi łamania tego adresu, więc jest on przeniesiony do kolejnej linii. Ponieważ tekst w akapicie jest wyrównany do lewej i do prawej strony, odstępy między słowami w dwóch pierwszych liniach są nienaturalnej wielkości. Jeżeli nie chcemy modyfikować treści tego akapitu, to sensowne jest wymuszenie łamania linii po słowie „Live":

LaTeX-a możemy pobrać na wszystkie popularne systemy operacyjne (Linux, Mac OS, Windows), na przykład dystrybucję TeX Live:
https://www.latex-project.org/get/

Łamanie linii możemy wymusić poleceniem `\\` (dwa ukośniki wsteczne). Mogłeś je już zauważyć na listingu 2.1. Alternatywnie można użyć polecenia **`\newline`**. LaTeX oferuje również polecenie `\\*`, które jednocześnie wymusza łamanie linii i zabrania łamania strony w miejscu złamania linii. Poleceń tych nie powinniśmy stosować na końcu akapitu, bo dodajemy w ten sposób pustą linię. Warto podkreślić, że łamanie linii nie oznacza zakończenia akapitu. Do tego potrzebna jest pusta linia w kodzie źródłowym. Obu poleceń można użyć z opcjonalnym parametrem, który określa dodatkowy pionowy odstęp po złamanej linii, np. `\\[2mm]`.

Poza bezwzględnym wymuszeniem łamania linii możemy zasugerować miejsce, w którym warto złamać linię lub nie powinno się tego robić. Służą do tego polecenia **`\linebreak`**`[n]` i **`\nolinebreak`**`[n]`. Opcjonalny parametr `n` przyjmuje wartości od 0 do 4 (wartość domyślna). Wartość 4 oznacza bezwarunkowy zakaz lub nakaz złamania linii. Wartości mniejsze od 4 pozwalają zignorować naszą instrukcję, jeżeli w wyniku jej użycia otrzymalibyśmy skład niskiej jakości. Łamanie strony możemy wymusić poleceniem **`\newpage`**. Dodatkowo dostępne są polecenia **`\pagebreak`**`[n]` i **`\nopagebreak`**`[n]`. Zachowują się podobnie jak wyżej omówione polecenia **`\linebreak`**`[n]` i **`\nolinebreak`**`[n]`, ale stanowią zachętę lub niezgodę na łamanie strony.

2.5 Cudzysłowy i nie tylko

Cudzysłów jest szczególnym znakiem interpunkcyjnym, który składa się z dwóch znaków – otwierającego i zamykającego. W tabeli 2.9 pokazano, w jaki sposób poprawnie uzyskać różne rodzaje cudzysłowów w LaTeX-u.

Apostrof wsteczny (grawis) to znak, który znajdziesz na tym samym klawiszu co tyldę. Cudzysłów angielski w przedstawionej wersji jest typowy dla odmiany amerykańskiej. W odmianie brytyjskiej może być używany w podanej podwójnej wersji, jak i w pojedynczej (1 apostrof wsteczny, 1 apostrof). W języku polskim cudzysłowów francuskich i niemieckich używa się głównie do oznaczania

Tabela 2.9. Cudzysłowy

Nazwa	Otwarcie	Zamknięcie	Przykład
polski	`,,` (2 przecinki)	`''` (2 apostrofy)	„polski”
angielski	``` `` ``` (2 apostrofy wsteczne)	`''` (2 apostrofy)	“angielski”
francuski	`<<` (2 znaki mniejszości)	`>>` (2 znaki większości)	«francuski»
niemiecki	`>>` (2 znaki większości)	`<<` (2 znaki mniejszości)	»niemiecki«

cytatów drugiego stopnia (cytat w cytacie). Warto też przypomnieć, że w języku polskim kropkę stawia się za cudzysłowem, a w języku angielskim wewnątrz.

W tabeli 2.10 przedstawiono znaki mające postać różnej długości poziomych kresek i sposób poprawnego ich uzyskania w LaTeX-u. Najkrótszym z nich jest dywiz, który najczęściej wstawiany jest automatycznie dla oznaczenia słowa dzielonego między liniami tekstu. Dywizu używamy również, tworząc różnego rodzaju złożenia, np. niebiesko-czarny, Bielsko-Biała. Warto zapamiętać, że wokół dywizu nigdy nie stawiamy spacji.

Tabela 2.10. Poziome kreski

Nazwa	Kod	Przykład użycia
dywiz (łącznik)	`-` (1 minus)	niebiesko-czarny
półpauza	`--` (2 minusy)	str. 12–17
pauza (myślnik)	`---` (3 minusy)	— Uczymy się LaTeX-a — powiedział X.

Najdłuższym znakiem w tabeli 2.10 jest pauza (myślnik) stosowana najczęściej w dialogach powieściowych. Współcześnie, aby poprawić estetykę tekstu, powszechne jest zastępowanie pauzy półpauzą (w takim przypadku zawsze po obu stronach są spacje). Półpauzę stosuje się również przy określaniu przedziałów, str. 12–17, lata 2020–2022 itp. (bardziej poprawnie bez spacji wokół półpauzy). Jeżeli pauza (półpauza) pełni rolę znaku międzywyrazowego i znajdzie się na granicy między wierszami, to należy ją pozostawić na końcu.

Z rozdziału 1 wiesz, że za obsługę dzielenia wyrazów odpowiada pakiet `babel`. W niektórych przypadkach zachodzi konieczność poprawienia dzielenia słów, zwłaszcza jeżeli mamy do czynienia z nietypowymi wyrazami, np. nazwami własnymi. Rozwiązaniem jest umieszczenie w preambule polecenia `\hyphenation{}`, które jako parametr przyjmuje listę słów oddzielonych przecinkami. W każdym z tych słów za pomocą sekwencji znaków `\-` możemy wskazać poprawne miejsce dzielenia danego słowa. Jeżeli nie wskażemy takiego miejsca, będzie to oznaczać, że dane słowo nie może być dzielone między wierszami.

Przykład użycia polecenia `\hyphenation{}` pokazano na listingu 2.10. W pierwszej linii wskazano słowa, w których wykluczono niektóre teoretycznie możliwe miejsca dzielenia wyrazów. W drugiej linii mamy nazwisko słynnego polskiego matematyka oraz dwie nazwy własne (nazwa firmy i polskiego samolotu z okresu międzywojennego). W tym przypadku zabezpieczamy się przed jakimkolwiek dzieleniem tych słów.

Listing 2.10. Przykład określenia zasad dzielenia wyrazów

```
\hyphenation{anty\-ma\-te\-ria, między\-na\-ro\-dowy, paro\-statek,
Banach, Bytom, Czapla}
```

`\usepackage{url}`

Na koniec tego podrozdziału chciałbym zwrócić uwagę na wykorzystanie tyldy w adresach stron internetowych. Jak wiemy, tylda jest znakiem specjalnym (twarda spacja) i musimy zastosować ukośnik wsteczny jako znak ucieczki. Na przykład adres strony z materiałami dla moich studentów złożony kursywą i z sekwencją `\~{}` w środku wygląda tak: *http://home.agh.edu.pl/~mszpyrka*. Tylda jest prawie niewidoczna. Adresy internetowe najlepiej składać, stosując polecenie `\url{}` z pakietu `url` (parametrem jest adres). W efekcie jego użycia dostajemy następującą formę adresu: `http://home.agh.edu.pl/~mszpyrka`.

W tym miejscu warto jeszcze wytłumaczyć się z `\~{}`. Po co puste nawiasy klamrowe po znaku tyldy? Przeanalizujmy listing 2.11. Ukośnik wsteczny w połączeniu z niektórymi znakami pozwala uzyskać litery zawierające znaki diakrytyczne. W nazwie Iruña sekwencja znaków `\~n` generuje znak ñ. Tylda została umieszczona nad literą n. Jeżeli dodamy `{}` (zobacz linia 2), to jest ona umieszczana nad pustym miejscem. Stąd brak nawiasów klamrowych w adresie internetowym skończy się wygenerowaniem litery m̃. Podobny efekt obserwujemy w pozostałych słowach z listingu 2.11 z innymi znakami diakrytycznymi.

Listing 2.11. Znaki diakrytyczne uzyskiwane z użyciem ukośnika wstecznego

```
1 Iru\~na,  M\"unchen, \'Swiebodzin, \.Zarnowiec
2 Iru\~{}na,  M\"{}unchen, \'{}Swiebodzin, \.{}Zarnowiec
```

Iruña, München, Świebodzin, Żarnowiec
Iru˜na, M¨unchen, ´Swiebodzin, ˙Zarnowiec

2.6 Znaki diakrytyczne i inne znaki specjalne

Przedstawiona na listingu 2.11 metoda uzyskiwania znaków diakrytycznych pozwala uzyskać aż 6 liter diakrytyzowanych z polskiego alfabetu – ć, ń, ó, ś, ź, ż (również wielkie litery Ć, Ń, Ó, Ś, Ź, Ż). Metoda ta była niegdyś jedynym sposobem na uzyskiwanie liter ze znakami diakrytycznymi. Literę ł (Ł) uzyskujemy poleceniem `\l` (`\L`), często jest ono umieszczane w klamrach, np. `{\L}ukasiewicz` pozwala uzyskać słowo „Łukasiewicz". Jeżeli chcemy uniknąć klamer, to kod należy zapisać jako `\L ukasiewicz`. Ostatnie dwie litery uzyskujemy za pomocą polecenia `\k{}`. Argumentem jest litera, której należy dodać znak diakrytyczny „ogonek". Słowo to w polskim brzmieniu funkcjonuje również w anglojęzycznych książkach do LaTeX-a. Zestawienie poleceń pozwalających uzyskać litery diakrytyzowane (nie tylko polskiego alfabetu) przedstawiono w tabeli 2.11.

Może się zdarzyć, że będziesz składał artykuł naukowy i wymagane będzie przygotowanie tekstu w kodowaniu `latin1`. W takim przypadku metoda uzyskiwania liter diakrytyzowanych przedstawiona w tabeli 2.11 pozwala np. uzyskać poprawny zapis polskich nazwisk.

Tabela 2.11. Polecenia generujące litery ze znakami diakrytycznymi

Polecenie	Litera	Polecenie	Litera
`\k{a}, \k{A}`	ą, Ą	`\"a, \"A`	ä, Ä
`\'c, \'C`	ć, Ć	`\‘a, \‘A`	à, À
`\k{e}, \k{E}`	ę, Ę	`\^a, \^A`	â, Â
`\l, \L`	ł, Ł	`\~a, \~A`	ã, Ã
`\'n, \'N`	ń, Ń	`\c{a}, \c{A}`	ą, Ą
`\'o, \'O`	ó, Ó	`\r{a}, \r{A}`	å, Å
`\'s, \'S`	ś, Ś	`\u{a}, \u{A}`	ă, Ă
`\'z, \'Z`	ź, Ź	`\v{a}, \v{A}`	ǎ, Ǎ
`\.z, \.Z`	ż, Ż		

LaTeX pozwala wygenerować nie tylko znaki diakrytyczne, ale mnóstwo innych symboli. Symbole te są dostępne poprzez polecenia grupowane w dodatkowych pakietach. W dokumentacji [41] możemy znaleźć 686 tabel z symbolami. Zasada ich używania jest bardzo prosta – w nagłówku tabeli czcionką bezszeryfową wyróżnione jest słowo będące nazwą pakietu. W większości przypadków wystarczy dodać ten pakiet, aby skorzystać z zawartych w danej tabeli

poleceń. Wybrane polecenia z pakietu `pifont` pokazano w tabeli 2.12. Skorzystamy z nich w dalszej części książki. Zwróć uwagę, że symbole te skalują się, np. `{\small\ding{96}, {\Large\ding{96}}` wygeneruje: ❀, ❀.

```
\usepackage{pifont}
```

Tabela 2.12. Wybrane polecenia z pakietu `pifont` [41]

✢ `\ding{66}`	★ `\ding{72}`	✾ `\ding{94}`	❄ `\ding{100}`
✣ `\ding{67}`	☆ `\ding{73}`	✿ `\ding{95}`	❅ `\ding{101}`
✤ `\ding{68}`	✪ `\ding{74}`	❀ `\ding{96}`	❆ `\ding{102}`
✥ `\ding{69}`	✫ `\ding{75}`	❁ `\ding{97}`	❇ `\ding{103}`

Chcesz uzyskać symbole hieroglifów, znaki zodiaku, motywy roślinne, a może twarze postaci z kreskówki *The Simpsons*? Wystarczy, że sięgniesz po dokumentację [41] i odszukasz właściwą nazwę pakietu i zgrupowane w nim polecenia.

2.7 Ramki i odległości

Ramka jest częścią dokumentu, w której obowiązuje tryb wierszowy (brak możliwości łamania linii). Mówiliśmy już o przykładach ramek w podrozdziale 2.3. Podstawowym poleceniem do tworzenia ramek jest `\framebox[][]{}`. Jedynym obowiązkowym parametrem jest tekst, który zostanie umieszczony w ramce. Zarówno tekst, jak i ramka są składane z użyciem bieżącego koloru czcionki (kolor czcionki możemy zmienić odpowiednim poleceniem umieszczonym wewnątrz ramki). Rozmiar ramki jest ustalany na podstawie jej zawartości. Jeżeli użyjemy jednego z opcjonalnych parametrów, to podajemy w nim szerokość ramki. Tekst zostanie wówczas wyśrodkowany w ramce. Drugi z opcjonalnych parametrów służy do określenia sposobu wyrównania tekstu w ramce: `c` – wyśrodkowanie, `l` – do lewej strony, `r` – do prawej strony, `s` – rozciągnięcie tekstu na całą szerokość ramki (jeżeli jej zawartość pozwala na rozciąganie), np. polecenie `\framebox[4cm]{Stefan Banach}` wygeneruje Stefan Banach. Dostępna jest skrócona wersja tego polecenia `\fbox{}`, która nie obsługuje parametrów opcjonalnych.

Jeżeli chcemy uzyskać ramkę bez rysowanej krawędzi, to stosujemy polecenie `\makebox[][]{}` lub wersję skróconą `\mbox{}`. Znaczenie parametrów jest takie samo jak dla poprzednich dwóch poleceń. Szczególnie użyteczne jest

polecenie **\mbox**. Gwarantuje ono, że umieszczony w takiej ramce tekst znajdzie się w jednym wierszu. Rozważmy następującą sytuację użycia twardej spacji `W~domu`. Chcieliśmy uniknąć pozostawienia litery W na końcu wiersza, ale nie zabezpiecza nas to przed podziałem na pół słowa domu. Użycie **\mbox**`{W domu}` daje gwarancję, że całość będzie umieszczona w jednym wierszu.

Długość ramek (i nie tylko ramek) możemy definiować zależnie od szerokości tekstu na stronie. Wartość ta jest dostarczana przez polecenie **\textwidth**, np. konstrukcja `0.6`**\textwidth** oznacza 60% szerokości tekstu. Wysokość tekstu na stronie można uzyskać za pomocą polecenia **\textheight**.

Ciekawym poleceniem jest **\dotfill**. Wypełnia ono kropkami dostępną przestrzeń do końca linii lub końca pudełka, w którym tego polecenia użyjemy. Na listingu 2.12 pokazano fragment formularza, który używa poleceń **\makebox** i **\dotfill**. W pierwszych dwóch liniach obszar wypełniony kropkami jest ograniczony wielkością pudełka, w ostatniej szerokością wiersza tekstu.

Listing 2.12. Fragment formularza

```
1 \makebox[8cm]{Imię i nazwisko: \dotfill}\\
2 \makebox[6cm]{Data urodzenia: \dotfill}\\
3 Adres zamieszkania: \dotfill
```

Imię i nazwisko: .....
Data urodzenia: .....
Adres zamieszkania: .....

Ostatni temat, który w tym miejscu omówię, to wprowadzanie w tekście poziomych i pionowych odstępów. Z rozdziału 1 wiesz, że w zwykłym tekście nie można do tego celu użyć wielokrotnych spacji, bo na etapie kompilacji i tak zostaną one zamienione na jedną spację. Poziomy odstęp możemy wstawić za pomocą polecenia **\hspace**`{}`. Parametrem polecenia jest wielkość tego odstępu, np. `7mm`. Polecenie to ma również wersję z gwiazdką **\hspace**`*{}`, która może być użyta na końcu wiersza. Do wstawiania pionowych odstępów służą polecenia **\vspace**`{}` i **\vspace**`*{}`. To drugie może być użyte na granicy stron.

Określając odległości w poleceniach LaTeX-a, możemy korzystać z różnych jednostek. Ich zestawienie zapożyczone z pozycji [40] zawarto w tabeli 2.13.

Są przypadki, gdy nie jesteśmy w stanie dokładnie określić wielkości odstępu, ale potrzebujemy, aby wypełnił on całą dostępną w danym miejscu przestrzeń. W takim przypadku możemy użyć polecenia **\fill**. Działa ono jak sprężyna, która rozszerza się na tyle, na ile ma miejsce. Jeżeli użyjemy go jako parametru polecenia **\hspace**, to będzie to rozszerzanie w poziomie, a jeśli **\vspace**, to w pionie. Połączenie **\hspace**`{`**\fill**`}` jest reprezentowane przez dodatkowe

Tabela 2.13. Jednostki używane przez LaTeX-a

Jednostka	Opis
mm	milimetr ≈ 1/25 cala
cm	centymetr
in	cal = 25,4 mm
pt	punkt ≈ 1/72 cala ≈ 1/3 mm
em	≈ szerokość litery M w bieżącej czcionce
ex	≈ wysokość litery x w bieżącej czcionce

polecenie **\hfill**, zaś **\par\vspace{\fill}** przez **\vfill** (**\par** jest TeX-owym poleceniem oznaczającym przejście do nowego akapitu).

Listing 2.13. Wykorzystanie elastycznych odległości

```
1 \fbox{
2 \begin{minipage}[t][5cm][t]{0.7\textwidth}
3 Kraków\hspace{\fill} 01.01.2025\\
4 \vspace{\fill}\\
5 2025
6 \end{minipage}}
```

Kraków 01.01.2025

2025

Zastosowanie polecenia **\fill** pokazano na listingu 2.13. Zdefiniowana ministrona ma szerokość równą 70% szerokości tekstu i wysokość 5 cm. Polecenie **\hspace{\fill}** użyte w 3 linii powoduje przesunięcie daty do prawej krawędzi ministrony. Z kolei polecenie **\vspace{\fill}** użyte w linii 4 powoduje przesunięcie roku na dół ministrony. Znaczenie pozostałych parametrów opcjonalnych ministrony omówimy w podrozdziale 3.4. Ministronę umieszczono w ramce dla podkreślenia efektu działania polecenia **\fill**.

Rozdział 3

Formatujemy akapity i organizujemy tekst

Akapit jest najważniejszą jednostką podziału tekstu w LaTeX-u. Jak wiesz z rozdziału 1, koniec akapitu i przejście do nowego realizowane jest poprzez wstawienie co najmniej jednej pustej linii w pliku źródłowym. W rozdziale 2 przedstawiłem metody wymuszenia lub zachęcenia LaTeX-a do złamania linii we wskazanym miejscu. Tutaj zajmiemy się akapitami jako całością. Omówię, jak wprowadzić inne niż domyślne wyrównanie tekstu w akapitach, jak organizować tekst w listy, a na koniec przedstawię wszystkie opcjonalne parametry związane z korzystaniem ze środowiska **minipage** oraz pakiet `tcolorbox` umożliwiający składanie tekstu w kolorowych ramkach.

3.1 Zarządzanie akapitami

Domyślnie LaTeX wyrównuje tekst w akapicie do lewej i do prawej strony oraz stosuje wcięcia akapitowe. Zgodnie z typografią angielską LaTeX nie stosuje wcięcia dla pierwszego akapitu po tytułach rozdziałów, podrozdziałów itp. Ustawienia takie są jednak niepoprawne, jeżeli przygotowujemy dokument w języku polskim. W takim przypadku warto w preambule dodać pakiet `indentfirst`, który zagwarantuje dodawanie wcięć również w pierwszych akapitach po tytułach (jest to jedyna rola tego pakietu).

```
\usepackage{indentfirst}
```

Wyrównanie akapitu do obu marginesów wymusza stosowanie między słowami odstępów różnej wielkości. LaTeX dobiera je automatycznie, aby uzyskać jak najlepszą jakość składu. Domyślnie odstęp wstawiany na końcu zdania jest nieco większy (po kropce, pytajniku lub wykrzykniku) niż odstępy wstawiane między słowami. Zachowanie takie jest typowe dla krajów anglosaskich. Jeżeli składamy dokument w języku polskim, to możemy wyłączyć to zachowanie poleceniem **\frenchspacing** umieszczonym w preambule. Polecenie nie wymaga żadnych parametrów.

Listing 3.1. Ustawienia dla wcięć akapitowych i odstępów między akapitami

```
\setlength{\parindent}{0ex}
\setlength{\parskip}{1ex}
```

Domyślnie LaTeX nie wstawia odstępów między kolejnymi akapitami. Możemy łatwo zarządzać wielkością wcięcia pierwszego wiersza akapitu oraz takimi odstępami. Stosowne polecenia pokazano na listingu 3.1. W przypadku tych poleceń zalecane jest używanie jednostek zależnych od wielkości czcionki (zob. tabela 2.13). Polecenia te umieszczamy w preambule dokumentu.

W celu zademonstrowania środowisk do wyrównywania tekstu użyjemy ciekawostki na temat pierwszego wydruku dzieła Mikołaja Kopernika przedstawionej w książce [25]. Tekst został tutaj podzielony na dwa akapity.

Listing 3.2. Przykład użycia środowiska **flushleft**

```
1  \begin{flushleft}
2  Mikołaj Kopernik na podstawie obserwacji astronomicznych ruchu Słońca,
3  Księżyca i planet na tle nieba doszedł do wniosku, że to Ziemia i inne
4  planety krążą wokół Słońca.
5
6  Bojąc się prześladowań oraz nie będąc pewnym swoich racji, dopiero w
7  ostatnim okresie życia przesłał swoją pracę \textit{De Revolutionibus}
8  do wydrukowania. Pierwszą kopię słynnego opracowania otrzymał w dniu
9  śmierci 24 maja 1543 r.
10 \end{flushleft}
```

Mikołaj Kopernik na podstawie obserwacji astronomicznych ruchu Słońca, Księżyca i planet na tle nieba doszedł do wniosku, że to Ziemia i inne planety krążą wokół Słońca.
Bojąc się prześladowań oraz nie będąc pewnym swoich racji, dopiero w ostatnim okresie życia przesłał swoją pracę *De Revolutionibus* do wydrukowania. Pierwszą kopię słynnego opracowania otrzymał w dniu śmierci 24 maja 1543 r.

Efekt wyrównania do lewej strony możemy uzyskać, stosując środowisko **flushleft**, jak to pokazano na listingu 3.2. Wewnątrz środowiska może się znaleźć wiele akapitów, tabele, rysunki itd. O ile dla danego typu elementów nie są stosowane specjalne zasady składu (np. centrowane wzory matematyczne w trybie wystawowym), to będą one wyrównane do lewej strony. Zamiast środowiska **flushleft** możemy zastosować przełącznik **\raggedright** (nierówna prawa strona). Efekt będzie podobny. Istotna różnica dotyczy odstępów, bo zarówno środowisko **flushleft**, jak i dwa kolejne, które omówię, wstawiają pionowe odstępy przed i po środowisku, czego nie robią odpowiadające im przełączniki.

Listing 3.3. Przykład użycia środowiska **flushright**

```
\begin{flushright}
Mikołaj Kopernik ...

Bojąc się ...
\end{flushright}
```

Mikołaj Kopernik na podstawie obserwacji astronomicznych ruchu Słońca, Księżyca i planet na tle nieba doszedł do wniosku, że to Ziemia i inne planety krążą wokół Słońca.
Bojąc się prześladowań oraz nie będąc pewnym swoich racji, dopiero w ostatnim okresie życia przesłał swoją pracę *De Revolutionibus* do wydrukowania. Pierwszą kopię słynnego opracowania otrzymał w dniu śmierci 24 maja 1543 r.

Listing 3.4. Przykład użycia środowiska **center**

```
\begin{center}
Mikołaj Kopernik ...

Bojąc się ...
\end{center}
```

Mikołaj Kopernik na podstawie obserwacji astronomicznych ruchu Słońca, Księżyca i planet na tle nieba doszedł do wniosku, że to Ziemia i inne planety krążą wokół Słońca.
Bojąc się prześladowań oraz nie będąc pewnym swoich racji, dopiero w ostatnim okresie życia przesłał swoją pracę *De Revolutionibus* do wydrukowania. Pierwszą kopię słynnego opracowania otrzymał w dniu śmierci 24 maja 1543 r.

Analogicznie możemy wykorzystać środowisko **flushright** (przykład użycia pokazano na listingu 3.3) oraz **center** (zobacz listing 3.4). Odpowiadające im przełączniki to **\raggedleft** i **\centering**. Zakres stosowania przełączników możemy ograniczyć za pomocą nawiasów klamrowych (mogą obejmować naprawdę duży fragment dokumentu). Jeżeli zastosujemy na przykład przełącznik **\raggedright**, a gdzieś niżej w dokumencie **\centering**, to oczywiście działanie pierwszego przełącznika zostanie wyłączone w miejscu użycia kolejnego.

Gdybyśmy szukali przełącznika, która pozwoli nam wrócić do domyślnego wyrównania do prawej i lewej strony, to musimy sięgnąć po pakiet `ragged2e`. Zawiera on m.in. przełącznik **\justifying**, który realizuje to zadanie. Pakiet ten obejmuje również nieco zmienione wersje omawianych w tym podrozdziale środowisk i przełączników. Są to odpowiednio **FlushLeft**, **\RaggedRight**, **FlushRight**, **\RaggedLeft**, **Center** i **\Centering**. Podstawowa różnica dotyczy kwestii dzielenia wyrazów. Pakietowe środowiska i przełączniki „starają się", aby brzegi były mniej nierówne, i stosują dzielenie wyrazów, które w oryginalnych środowiskach prawie nie występuje. Innym zastosowaniem pakietu `ragged2e` jest możliwość włączenia wyrównania do lewej strony dla całego dokumentu. W takim przypadku dodajemy pakiet z opcją `document`.

```
\usepackage{ragged2e}              % lub
\usepackage[document]{ragged2e}
```

Jeżeli potrzebujemy umieścić w dokumencie krótki cytat, to możemy zastosować środowisko **quote**. Przykład użycia pokazano na listingu 3.5. Do dłuższych cytatów (więcej niż jeden akapit) lepsze jest środowisko **quotation**.

Listing 3.5. Przykład użycia środowiska **quote**

```
I zasada dynamiki:
\begin{quote}
Jeżeli na ciało nie działa żadna siła, to pozostaje ono w spoczynku
lub porusza się ruchem jednostajnym prostoliniowym.
\end{quote}
```

I zasada dynamiki:

> Jeżeli na ciało nie działa żadna siła, to pozostaje ono w spoczynku lub porusza się ruchem jednostajnym prostoliniowym.

Innym ciekawym środowiskiem jest **verse** służące do składania poezji. W środowisku tym koniec wersu oznaczamy poleceniem `\\`, a koniec strofy pustą linią. Na listingu 3.6 pokazano skład wiersza *Ty mi nie będziesz ciężką...* Wiktora Dzierżanowskiego.

Listing 3.6. Przykład użycia środowiska **verse**

```
\begin{verse}
Ty mi nie będziesz ciężką, gdy po moim zgonie\\
Na pierś moją upadniesz, ziemio-karmicielko!\\
Kryjesz wszak w swoim wnętrzu miłość stworzeń wielką,\\
Byt twój wraz z dżdżu kroplami i łzy nasze chłonie!

O! wiem, że ty masz serce, co ogniami płonie\\
Zdolne przetopić w sobie ból i nędzę wszelką --\\
A ja jestem rosy maleńką kropelką,\\
Na twem wiecznie zielonem i dziewiczem łonie!

Kiedyś -- gdy ciało moje ty nakryjesz sobą --\\
Kwiaty co wzejdą bujnie może z setną wiosną,\\
Stawszy się szczęściem źrenic i dolin ozdobą,

Śląc z wonią wieść o wiośnie po świecie radosną --\\
Zaświadczą, że mych soków w tobie krąży siła --\\
Żeś ty mi jako puchy kwietne, lekką była!\dots
\end{verse}
```

Ty mi nie będziesz ciężką, gdy po moim zgonie
Na pierś moją upadniesz, ziemio-karmicielko!
Kryjesz wszak w swoim wnętrzu miłość stworzeń wielką,
Byt twój wraz z dżdżu kroplami i łzy nasze chłonie!

O! wiem, że ty masz serce, co ogniami płonie
Zdolne przetopić w sobie ból i nędzę wszelką –
A ja jestem rosy maleńką kropelką,
Na twem wiecznie zielonem i dziewiczem łonie!

Kiedyś – gdy ciało moje ty nakryjesz sobą –
Kwiaty co wzejdą bujnie może z setną wiosną,
Stawszy się szczęściem źrenic i dolin ozdobą,

Śląc z wonią wieść o wiośnie po świecie radosną –
Zaświadczą, że mych soków w tobie krąży siła –
Żeś ty mi jako puchy kwietne, lekką była!...

Środowisko **verse** jest przykładem bardzo mało wymagającego nakładu pracy po stronie osoby składającej tekst, a daje bardzo eleganckie efekty. Złożenie w LaTeX-u tomiku wierszy wydaje się drobnostką. Do bardziej zaawansowanego składu poezji można użyć pakietu `poemscol`.

3.2 Interlinia

Ważnym elementem składania akapitów jest ustawienie odstępu między wierszami – interlinii. Domyślnie używany jest odstęp pojedynczy. Jeżeli potrzebujemy zmienić te ustawienia dla całego dokumentu lub dla wybranych fragmentów, to zalecane jest użycie pakietu `setspace`. Przede wszystkim pakiet ten zawiera trzy polecenia: **`\singlespacing`**, **`\onehalfspacing`** i **`\doublespacing`** do ustawienia dla całego dokumentu interlinii odpowiednio 1, 1,5 i 2. Ustawienie sprowadza się do użycia jednego z tych poleceń w preambule. Polecenia można używać również wewnątrz dokumentu, jeżeli chcemy zmienić interlinię od pewnego momentu.

```
\usepackage{setspace}
```

Warto zwrócić uwagę, że odstęp pojedynczy nie jest stałą wielkością wyrażoną na przykład w punktach. Zależy on od wielkości czcionki (jest przechowywany w zmiennej **`\baselineskip`**). W tabeli 3.1 przedstawiono wielkość odstępu w punktach zależnie od wielkości czcionki bazowej.

Tabela 3.1. Zależność odstępu między wierszami od wielkości czcionki

Rozmiar czcionki (f)	**`\baselineskip`** (bls)	$\frac{\textbf{\texttt{\textbackslash baselineskip}}}{\textbf{Rozmiar czcionki}} = \frac{bls}{f}$
10pt	12pt	1,2
11pt	13,6pt	1,24
12pt	14,5pt	1,21

Możemy przyjąć, że dla pojedynczej interlinii przedstawiony w ostatniej kolumnie współczynnik jest równy w przybliżeniu 1,2. Jeżeli chcemy ustawić dowolną interlinię, to możemy to zrobić za pomocą polecenia **`\setstretch{}`**. Jeżeli chcemy ustawić interlinię n (np. potrójną $n = 3$), to musimy wyznaczyć wartość wyrażenia:

$$n \cdot \frac{f}{bls} \approx \frac{n}{1,2}, \tag{3.1}$$

gdzie f to nasza bazowa czcionka, a bls odpowiednia wartość z tabeli 3.1, i użyć wyniku jako parametru polecenia **`\setstretch`**. Zatem przy czcionce 11 pt współczynnik dla potrójnej interlinii wynosi $3 \cdot \frac{11}{13,6} \approx 2,43$. Jeżeli nie potrzebujemy dokładnych obliczeń (nie mamy pod ręką tabeli), to wyznaczamy $\frac{3}{1,2} = 2,5$ i korzystamy z polecenia **`\setstretch{2.5}`**. Polecenia **`\setstretch`** możemy

użyć zarówno w preambule dokumentu, jak i w treści (nawet wielokrotnie zmieniając ustawienia od wybranego momentu).

Jeżeli potrzebujemy skorzystać ze zmienionej interlinii tylko dla fragmentu dokumentu, to pakiet `setspace` oferuje w tym celu trzy środowiska **`singlespace`**, **`onehalfspace`** i **`doublespace`**. Przykład użycia środowiska **`onehalfspace`** pokazano na listingu 3.7. W efekcie użyta strofa jest złożona z interlinią 1,5.

Listing 3.7. Przykład użycia środowiska **`onehalfspace`**

```
\begin{onehalfspace}
\begin{verse}
Ty mi nie będziesz ciężką, gdy po moim zgonie\\
...
\end{verse}
\end{onehalfspace}
```

Ty mi nie będziesz ciężką, gdy po moim zgonie
Na pierś moją upadniesz, ziemio-karmicielko!
Kryjesz wszak w swoim wnętrzu miłość stworzeń wielką,
Byt twój wraz z dżdżu kroplami i łzy nasze chłonie!

Jeżeli podane środowiska nie są wystarczające, to dostępne jest środowisko **`spacing`**. Środowisko to przyjmuje jeden obowiązkowy parametr – współczynnik wyznaczany tak jak dla polecenia **`\setstretch`**, np. **`\begin{spacing}{2.5}`**.

3.3 Listy

Wypunktowania (nienumerowane listy) możemy tworzyć z użyciem środowisk **`itemize`** i **`description`**, zaś wyliczenia (numerowane listy) z użyciem środowiska **`enumerate`**. Kluczowym poleceniem dla wszystkich tych środowisk jest polecenie **`\item`**. Tworzy ono kolejny element (punkt) listy. Po wystąpieniu tego polecenia wszystko, co zamieścimy w dokumencie (akapity tekstu, wzory matematyczne, rysunki itp.), aż do kolejnego polecenia **`\item`** (lub do końca środowiska, jeżeli mamy do czynienia z ostatnim elementem), należy do tego elementu. Każde z tych środowisk wymaga użycia co najmniej raz polecenia **`\item`** i to na początku zawartości (o ile nie wstawiamy np. poleceń modyfikujących ustawienia danego środowiska).

Listing 3.8. Przykład użycia środowiska **`itemize`**

```
Zasady dynamiki:
\begin{itemize}
```

```
3  \item Jeżeli na ciało nie działa żadna siła, to pozostaje ono w
4  spoczynku lub porusza się ruchem jednostajnym prostoliniowym.
5
6  \item Przyspieszenie ciała jest wprost proporcjonalne do wypadkowej
7  siły działającej na ciało oraz odwrotnie proporcjonalne do jego masy.
8
9  Kierunek przyspieszenia jest zgodny z kierunkiem tej siły.
10
11 \item Każda akcja spotyka się z równą jej (i przeciwnie skierowaną)
12 reakcją.
13 \end{itemize}
```

Zasady dynamiki:

- Jeżeli na ciało nie działa żadna siła, to pozostaje ono w spoczynku lub porusza się ruchem jednostajnym prostoliniowym.
- Przyspieszenie ciała jest wprost proporcjonalne do wypadkowej siły działającej na ciało oraz odwrotnie proporcjonalne do jego masy.

  Kierunek przyspieszenia jest zgodny z kierunkiem tej siły.
- Każda akcja spotyka się z równą jej (i przeciwnie skierowaną) reakcją.

Przykład użycia środowiska **itemize** pokazano na listingu 3.8. Wewnątrz środowiska trzykrotnie użyliśmy polecenie **\item**, stąd nasza lista ma trzy elementy. Początek każdego z nich oznaczony jest półpauzą. Drugi element złożony jest z dwóch akapitów. Zwróć uwagę na sposób złożenia drugiego akapitu. Nie zaczyna się on od półpauzy, bo nie użyliśmy polecenia **\item**, ale jest odpowiednio wsunięty. Łatwo dzięki temu zauważyć, że mieści się w danym elemencie listy.

Puste linie 5 i 10 nie są niezbędne, ale zwiększają czytelność kodu, więc warto je stosować. Jeżeli pracujemy nad dokumentem, to zapewne nieraz będziemy nanosić w nim poprawki. Warto dbać o przejrzystość kodu źródłowego. Łatwiej będzie nam z nim pracować.

Środowisko **itemize** możemy zagnieżdżać, tworząc wielopoziomowe wypunktowania (aż do czterech poziomów). Możemy mieć wtedy punkty z podpunktami, a te podpunkty mogą mieć swoje podpunkty itd. LaTeX na każdym poziomie domyślnie używa innego symbolu do oznaczania punktów. Możemy łatwo zarządzać tymi symbolami z użyciem pakietu `enumitem`. Można wprowadzać ustawienia globalnie (w preambule) lub lokalnie dla danego wypunktowania. W drugim przypadku pakiet umożliwia używanie parametrów opcjonalnych środowiska. Własności, które chcemy zmodyfikować,

podajemy na zasadzie `nazwa=wartość`. Jeżeli chcemy ustawić półpauzę jako symbol dla pierwszego poziomu, wystarczy dodać w preambule polecenie `\setlist[itemize,1]{label=--}` (liczba w nawiasie kwadratowym wskazuje na poziom, dla którego zmieniamy ustawienia). Jeżeli chcemy to zrobić dla danego wypunktowania, to dodajemy parametr opcjonalny `[label=--]` dla tego środowiska. Możliwa jest zmiana symbolu indywidualnie dla jednego punktu. Wtedy symbol, który ma zostać użyty, podajemy jako opcjonalny parametr polecenia `\item` (do tego nie potrzebujemy dodatkowego pakietu). Przykład wykorzystania indywidualnych symboli elementów pokazano na listingu 3.9.

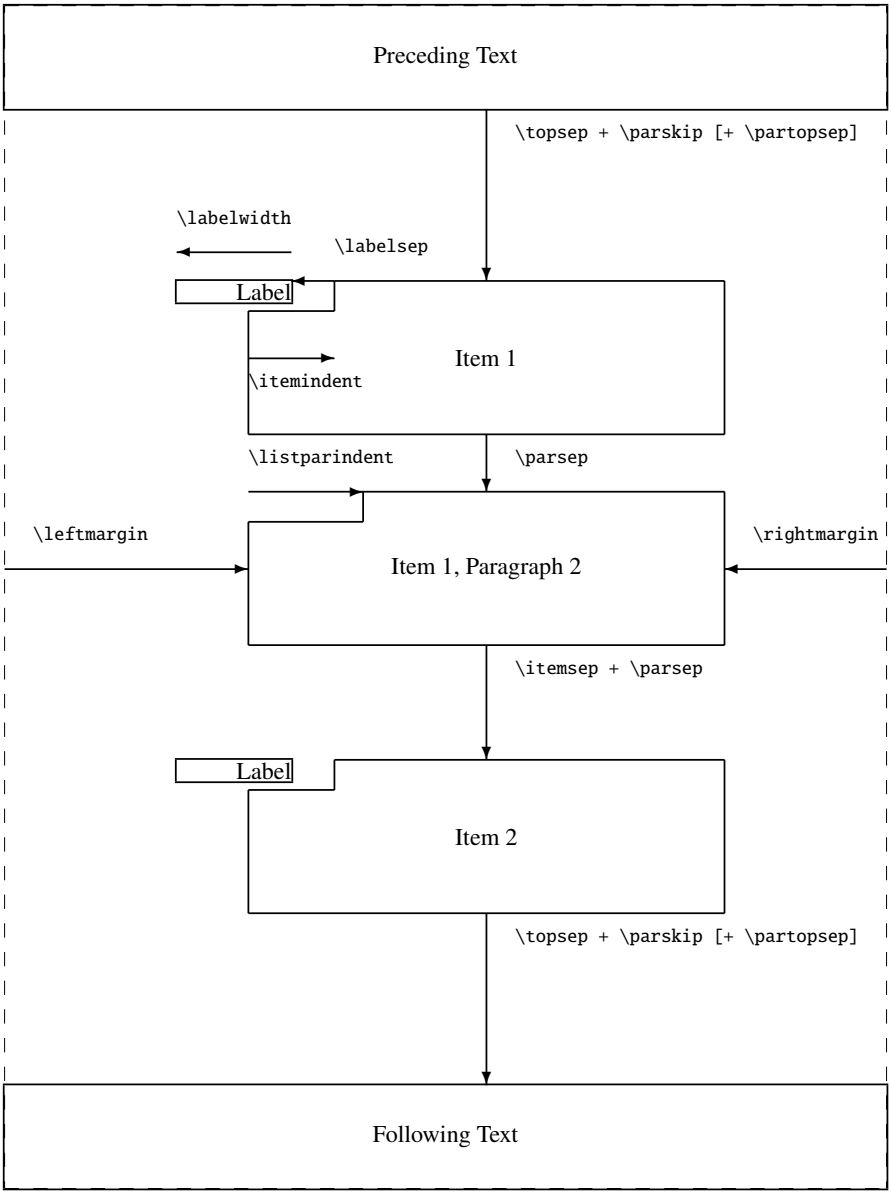
```
\usepackage{enumitem}
```

Listing 3.9. Przykład użycia indywidualnych symboli dla kolejnych punktów w środowisku **itemize**

```
1 \begin{itemize}
2 \item[\ding{94}] pierwszy element
3 \item[\ding{95}] drugi element
4 \item[\ding{96}] trzeci element
5 \end{itemize}
```

- ✾ pierwszy element
- ✿ drugi element
- ❀ trzeci element

Pakiet `enumitem` pozwala modyfikować wiele różnych parametrów wyświetlania list. Jak zapewne zauważyłeś, między poszczególnymi elementami listy stosowane są pionowe odstępy (dotyczy to wszystkich środowisk omawianych w tym podrozdziale). Często można spotkać komentarze, że LaTeX stosuje za duże odstępy w takich przypadkach. Na rys. 3.1 pokazano diagram wygenerowany przez pakiet `layouts`. Wizualizuje on polecenia LaTeX-a, które zarządzają odstępami w środowiskach składających listy. Jeżeli z nazw poleceń na rysunku usuniemy ukośniki wsteczne (z wyjątkiem `\parskip`), to dostaniemy nazwy opcji, których możemy używać przy ustawianiu własności np. środowiska **itemize**. Zatem aby zmniejszyć odstęp między punktami, wystarczy dodać opcję np. `itemsep=1mm`. Czasami drobna modyfikacja tego parametru wystarczy, by przy prawie niewidocznej zmianie jeden element więcej zmieścił się na stronie. Jeżeli chcemy całkowicie usunąć dodatkowe odstępy pionowe, to możemy użyć opcji `nosep`. W takim przypadku nie będzie nie tylko dodatkowych odstępów między punktami, ale również przed i po środowisku.



Rys. 3.1. Układ środowisk składających listy

Wróćmy do przykładu pokazanego na listingu 3.8. Załóżmy, że chcemy, aby zamiast półpauzy dla pierwszego elementu pojawił się tekst **I zasada dynamiki**. Możemy użyć polecenia `\item[\textbf{I zasada dynamiki}]`, ale będziemy niezadowoleni z efektów. Całość naszej etykiety zostanie umieszczona

na lewym marginesie środowiska (to dodatkowe względem marginesu strony wcięcie po lewej stronie), a ponieważ się tam nie zmieści, wejdzie daleko na lewy margines strony. Rozwiązaniem naszego problemu jest użycie środowiska **description**, jak pokazano na listingu 3.10.

Listing 3.10. Przykład użycia środowiska **description**

```
\begin{description}
\item[I zasada dynamiki] Jeżeli na ciało nie działa żadna siła, to
pozostaje ono w~spoczynku lub porusza się ruchem jednostajnym
prostoliniowym.

\item[II zasada dynamiki] Przyspieszenie ciała jest wprost
proporcjonalne do wypadkowej siły działającej na ciało oraz
odwrotnie proporcjonalne do jego masy.

Kierunek przyspieszenia jest zgodny z kierunkiem tej siły.

\item[III zasada dynamiki] Każda akcja spotyka się z równą
jej (i przeciwnie skierowaną) reakcją.
\end{description}
```

I zasada dynamiki Jeżeli na ciało nie działa żadna siła, to pozostaje ono w spoczynku lub porusza się ruchem jednostajnym prostoliniowym.

II zasada dynamiki Przyspieszenie ciała jest wprost proporcjonalne do wypadkowej siły działającej na ciało oraz odwrotnie proporcjonalne do jego masy.

Kierunek przyspieszenia jest zgodny z kierunkiem tej siły.

III zasada dynamiki Każda akcja spotyka się z równą jej (i przeciwnie skierowaną) reakcją.

Etykiety dla poszczególnych punktów są automatycznie składane pogrubioną czcionką. Możemy to zmienić, stosując odpowiednie polecenia zmiany czcionki wewnątrz nawiasów kwadratowych. Środowisko **description** możemy zagnieżdżać w nim samym, możemy też mieszać ze sobą omawiane w tym podrozdziale środowiska, tj. na przykład na pierwszym poziomie użyć **description**, a na drugim **itemize**.

Ostatnie omawiane środowisko służy do konstruowania wyliczeń. Domyślnie poszczególne elementy listy oznaczane są liczbami arabskimi z kropką po prawej stronie. Przykład użycia środowiska **enumerate** przedstawiono na listingu 3.11.

Listing 3.11. Przykład użycia środowiska **enumerate**

```
\begin{enumerate}
\item 1903 Maria Skłodowska-Curie -- fizyka
\item 1905 Henryk Sienkiewicz -- literatura
\item 1911 Maria Skłodowska-Curie -- chemia
\item 1924 Władysław Reymont -- literatura
\end{enumerate}
```

1. 1903 Maria Skłodowska-Curie – fizyka
2. 1905 Henryk Sienkiewicz – literatura
3. 1911 Maria Skłodowska-Curie – chemia
4. 1924 Władysław Reymont – literatura

Podobnie jak w przypadku środowiska **itemize**, możemy podać opcjonalne parametry dla polecenia **\item**, np. moglibyśmy użyć jako etykiety roku, w którym przyznano nagrodę Nobla. Przykład takiej modyfikacji pokazano na listingu 3.12.

Listing 3.12. Przykład użycia indywidualnych etykiet w środowisku **enumerate**

```
\begin{enumerate}
\item[1903] Maria Skłodowska-Curie -- fizyka
\item[1905] Henryk Sienkiewicz -- literatura
\item[1911] Maria Skłodowska-Curie -- chemia
\item[1924] Władysław Reymont -- literatura
\end{enumerate}
```

1903 Maria Skłodowska-Curie – fizyka

1905 Henryk Sienkiewicz – literatura

1911 Maria Skłodowska-Curie – chemia

1924 Władysław Reymont – literatura

LaTeX wykorzystuje 4 liczniki (zmienne przyjmujące wartości całkowite) do zliczania liczby elementów na poszczególnych poziomach: `enumi`, `enumii`, `enumiii` oraz `enumiv`. Oznaczenia punktów są generowane na podstawie aktualnej wartości tych liczników. Wystąpienie polecenia **\item** wewnątrz środowiska **enumerate** powoduje aktualizację licznika, o ile nie użyjemy parametru opcjonalnego. Wynika stąd, że jeśli decydujemy się na indywidualne etykietowanie, to już raczej dla wszystkich elementów. W innym przypadku efekt końcowy może nas nie zachwycić (np. użycie indywidualnych etykiet tylko przy Marii Skłodowskiej-Curie (listing 3.12) spowoduje, że przy Henryku Sienkiewiczu

zobaczymy etykietę `1.`, a przy Władysławie Reymoncie `2.`, czyli będziemy mieć sekwencję: 1903, 1, 1911, 2.

Sposób numerowania punktów możemy łatwo modyfikować z użyciem pakietu `enumitem`. Sprowadza się to do zdefiniowania opcji `label` dla środowiska, przy czym zamiast znaku wypunktowania musimy użyć poleceń określających styl numeracji, które pokazano w tabeli 3.2. Gwiazdka w poleceniach zastępuje aktualną wartość licznika.

Tabela 3.2. Polecenia definiujące styl numeracji

Polecenie	Symbol	Przykład
`\arabic*`	1	1,2,3...
`\Alph*`	A	A, B, C...
`\alph*`	a	a, b, c...
`\Roman*`	I	I, II, III...
`\roman*`	i	i, ii, iii...

Przykłady użycia tych poleceń pokazano na listingu 3.13. Zwróćmy uwagę na wybrane definicje. W drugim przypadku wielka litera A jest zwykłym znakiem, do którego następnie dołączona jest liczba arabska. W trzecim przypadku użycie dodatkowych nawiasów klamrowych pozwoliło na umieszczenie liczby w nawiasie kwadratowym – dzięki klamrom nie ma kolizji między znakiem `]` użytym jako część etykiety a zamknięciem nawiasu z opcjami. Jeżeli chcemy ustawić konkretne sposoby numerowania globalnie (zalecana opcja), to możemy dodać w preambule np. `\setlist[enumerate,1]{label=\Roman*.}`. Drugi parametr opcjonalny wskazuje na poziom, dla którego definiujemy sposób numerowania.

Listing 3.13. Przykłady definiowania sposobu numeracji elementów listy

```
\begin{enumerate}[label=\alph*)]
\item pierwszy element
\item drugi element
\item trzeci element
\end{enumerate}
```

a) pierwszy element
b) drugi element
c) trzeci element

```
\begin{enumerate}[label=A\arabic*:]
\item pierwszy element
\item drugi element
\item trzeci element
\end{enumerate}
```

A1: pierwszy element
A2: drugi element
A3: trzeci element

```
\begin{enumerate}[label={[\arabic*]}]
\item pierwszy element
\item drugi element
\item trzeci element
\end{enumerate}
```

[1] pierwszy element
[2] drugi element
[3] trzeci element

```
\begin{enumerate}[label={\bfseries etap (\alph*)}]
\item pierwszy element
\item drugi element
\item trzeci element
\end{enumerate}
```

etap (a) pierwszy element
etap (b) drugi element
etap (c) trzeci element

W tabeli 3.2 podano również jednoliterowe odpowiedniki poleceń. Są one dostępne, jeśli dodamy pakiet `enumitem` z opcją `[shortlabels]`. W takim przypadku możemy zdefiniować pierwszą metodę numerowania z listingu 3.13 w postaci `[a)]`.

Jak już powiedziałem, LaTeX aktualizuje swoje liczniki `enumi`, `enumii` itd. po dodaniu nowego punktu. Można wydrukować wartość takiego licznika. Robimy to, dodając w nazwie polecenia słowo „the", tj. **`\theenumi`**, **`\theenumii`** itd. (końcówki nazwy to liczby w systemie rzymskim). Pozwala to uzyskać ciekawe efekty w przypadku zagnieżdżenia środowisk **`enumerate`**. Przykład użycia licznika **`\theenumi`** przedstawiono na listingu 3.14. Przy definiowaniu sposobu numeracji na drugim poziomie korzystamy z aktualnej wartości licznika na poziomie pierwszym. W efekcie otrzymujemy etykiety 2.a, 2.b, 2.c, a w kolejnym punkcie 3.a, 3.b i 3.c.

Listing 3.14. Przykład użycia licznika przy definiowaniu sposobu numeracji

```
\begin{enumerate}[label=\arabic*]
\item pierwszy element
\item drugi element
\begin{enumerate}[label=\theenumi.\alph*]
\item podpunkt
\item podpunkt
\item podpunkt
\end{enumerate}
\item trzeci element
\begin{enumerate}[label=\theenumi.\alph*]
\item podpunkt
\item podpunkt
\item podpunkt
\end{enumerate}
\end{enumerate}
```

1 pierwszy element
2 drugi element
2.a podpunkt
2.b podpunkt
2.c podpunkt
3 trzeci element
3.a podpunkt
3.b podpunkt
3.c podpunkt

Pakiet `enumitem` niestety nie współpracuje najlepiej z pakietem `beamer` (do przygotowania prezentacji, zob. rozdział 13). Część opisanych w tym podrozdziale zmian można zrealizować bez jego pomocy, np. z użyciem polecenia **`\setlength`**, np.: **`\setlength`**`{`**`\itemsep`**`}{1mm}`. Polecenie takie umieszczamy na początku środowiska listowego. Do zarządzania numeracją można użyć pakietu `enumerate`, który stosuje symbole z drugiej kolumny tabeli 3.2.

Pakiet `enumitem` oferuje więcej ciekawych możliwości sterowania wyglądem numeracji. Jeżeli do środowiska **`enumerate`** dodamy opcję `start=x`, to numeracja rozpocznie się od wartości `x`. Jeżeli w kolejnym środowisku **`enumerate`** chcielibyśmy kontynuować numerację z poprzedniego, to wystarczy dodać opcję `resume*`. Jeżeli pakiet `enumitem` dołączymy z opcją `[inline]`, to dostępne będą środowiska **`itemize`**`*`, **`description`**`*` i **`enumerate`**`*`. Środowiska te składają listy w postaci pojedynczego akapitu. Przykład pokazano na listingu 3.15.

Listing 3.15. Przykład składania listy numerowanej w trybie `inline`

```
\begin{enumerate*}[label=\alph*)]
\item 1903 Maria Skłodowska-Curie -- fizyka
\item 1905 Henryk Sienkiewicz -- literatura
\item 1911 Maria Skłodowska-Curie -- chemia
\item 1924 Władysław Reymont -- literatura
\end{enumerate*}
```

a) 1903 Maria Skłodowska-Curie – fizyka b) 1905 Henryk Sienkiewicz – literatura c) 1911 Maria Skłodowska-Curie – chemia d) 1924 Władysław Reymont – literatura

Jeżeli chcemy globalnie ustawić opcje pakietu w preambule (o metodach zmiany liczników lub znaków wypunktowania mówiliśmy wcześniej), to stosujemy polecenie **`\setlist`**`{}`. Ogólne zasady pokazano na listingu 3.16.

Listing 3.16. Składnia określania opcji dla środowisk składających listy

```
\setlist[enumerate,poziom]{opcje}
\setlist[itemize,poziom]{opcje}
\setlist[description,poziom]{opcje}
\setlist[poziom]{opcje}
```

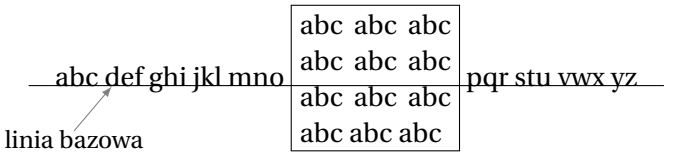
Pierwsze trzy linie wskazują, jak zdefiniować opcje dla określonych środowisk. W ostatniej linii mamy metodę określania opcji dla wszystkich środowisk jednocześnie. Jeżeli nie wskażemy poziomu, to ustawienia będą mieć zastosowanie do wszystkich poziomów, np. **`\setlist`**`{nosep}` spowoduje usunięcie

dodatkowych pionowych odstępów dla wszystkich środowisk i na wszystkich poziomach, zaś polecenie `\setlist[itemize,2]{leftmargin=2cm}` zmieni ustawienia lewego marginesu tylko dla środowiska **itemize** na drugim poziomie.

3.4 Środowisko minipage

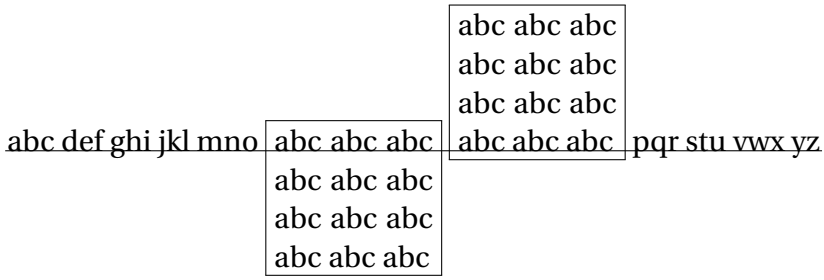
W podrozdziale 2.3 pokazano przykład wykorzystania środowiska **minipage** do umieszczenia większego fragmentu tekstu wewnątrz ramki. W przykładzie z listingu 2.9 skorzystaliśmy z ministrony z jedynym obowiązkowym parametrem, którym jest szerokość ministrony. Jej wysokość jest wówczas dobierana na podstawie zawartości. Na listingu 2.13 podano przykład z kompletem parametrów opcjonalnych, ale wspomniano o znaczeniu tylko jednego z nich. W tym podrozdziale omówimy ich znaczenie i przedstawimy dopuszczalne wartości.

Środowisko **minipage** definiuje obszar, w którym możemy składać tekst w zwykłym trybie tekstowym (akapitowym). Jest niejako stroną zagnieżdżoną w stronie naszego dokumentu, zazwyczaj o mniejszej szerokości. Środowisko może przyjąć trzy parametry opcjonalne. Zanim wyjaśnimy sobie ich znaczenie, warto przypomnieć, czym jest linia bazowa dla wiersza tekstu, bo względem niej są rozmieszczane ministrony. W typografii linia bazowa wiersza to linia, na której spoczywają wszystkie litery. Niektóre litery, np. „p", mają wydłużenie dolne wystające poniżej linii bazowej. Linię bazową dla wiersza tekstu pokazano na rys. 3.2. Jeżeli wewnątrz takiego wiersza umieścimy ministronę bez parametrów opcjonalnych, to zostanie ona wyśrodkowana w pionie względem tego wiersza, jak widać to na rysunku (oś symetrii jest w połowie wysokości liter a, c, e itd.).



Rys. 3.2. Linia bazowa – ministrona wyśrodkowana w pionie

Jeżeli skorzystamy z jednego parametru opcjonalnego, to będzie on określał właśnie sposób umieszczenia w pionie ministrony względem wiersza, w którym jest ona umieszczona. Wartość domyślna `c` daje efekt jak na rys. 3.2. Wartość `t` oznacza, że linia bazowa pierwszego wiersza ramki zostanie wyrównana z linią bazową wiersza, w którym umieszczono ministronę, zaś `b`, że linia bazowa ostatniego wiersza ramki zostanie wyrównana z linią bazową wiersza. Zostało to zilustrowane na rys. 3.3.



Rys. 3.3. Linia bazowa – ministrony z parametrami `t` i `b`

Dodanie drugiego parametru opcjonalnego oznacza, że sami definiujemy wysokość ministrony. W takim przypadku wysokość ministrony może być większa, niż jest to wymagane. Trzeci parametr służy do wskazania sposobu wyrównania w pionie zawartości ministrony. Dostępne opcje to: `t` – dosunięcie do góry, `b` – dosunięcie do dołu, `c` – wyśrodkowanie i `s` – rozciągnięcie w pionie (użycie tego parametru wymaga dodania do zawartości polecenia, które pozwoli na rozciągnięcie, np. `\vspace{\fill}`). Jeżeli trzeci parametr opcjonalny nie jest określony, to używana jest wartość pierwszego parametru.

Środowiska **minipage** używamy często, aby dany fragment strony złożyć w wielu kolumnach. Poszczególne kolumny definiujemy właśnie jako ministrony, określając ich szerokość i sposób wyrównania tekstu w środku, co pozwala uzyskać ciekawe kombinacje. Należy tutaj zwrócić uwagę, że ministrony są układane w wierszu podobnie jak słowa i podobnie jak między słowami LaTeX wstawi między ministronami spacje – jeżeli jest to niepożądane, to możemy się postarać, aby je wyeliminować. Podsumowując, musimy więc zadbać, aby nasza konstrukcja nie była szersza niż szerokość strony, bo LaTeX po prostu złamie wiersz i efekt końcowy nie będzie zgodny z naszym oczekiwaniem – na przykład ostatnia ministrona zostanie przesunięta do kolejnego wiersza.

Listing 3.17. Przykład użycia środowiska **minipage**

```
\noindent
\begin{minipage}[t][1.9cm][b]{0.32\textwidth}
Maria Skłodowska-Curie

1867--1934
\end{minipage}\hspace{\fill}
%
\begin{minipage}[t]{0.66\textwidth}
Dwukrotnie otrzymała Nagrodę Nobla: w 1903 r. z~fizyki za badania nad
zjawiskiem promieniotwórczości, a w 1911 r. z chemii za odkrycie
```

```
11 polonu i radu, wydzielenie czystego radu i badanie właściwości
12 chemicznych pierwiastków promieniotwórczych. Jest jedyną kobietą,
13 która otrzymała dwie nagrody w dwóch różnych dyscyplinach.
14 \end{minipage}
```

Maria Skłodowska-Curie 1867–1934	Dwukrotnie otrzymała Nagrodę Nobla: w 1903 r. z fizyki za badania nad zjawiskiem promieniotwórczości, a w 1911 r. z chemii za odkrycie polonu i radu, wydzielenie czystego radu i badanie właściwości chemicznych pierwiastków promieniotwórczych. Jest jedyną kobietą, która otrzymała dwie nagrody w dwóch różnych dyscyplinach.

Przykład wykorzystania środowiska **minipage** do dwukolumnowego składu pokazano na listingu 3.17. W tym przypadku zdecydowano się na podział szerokości kolumn w proporcji 1 do 2.

Polecenie **\noindent** użyte w linii 1 ma za zadanie usunąć wcięcie pierwszego wiersza dla tego akapitu (nasz akapit będzie miał jeden wiersz, w którym znajdą się dwie ministrony). Szerokość pierwszej kolumny ustawiono na 32% szerokości tekstu, a drugiej na 66%. Pozostałe 2% wykorzystano na odstęp między kolumnami. Jest on generowany przez polecenie **\hspace{\fill}** użyte w linii 6. Jeżeli w miejsce tego polecenia wstawilibyśmy znak komentarza (%), to wyeliminowalibyśmy jakikolwiek odstęp między ministronami – teksty z obu kolumn stykałyby się ze sobą. Pusty komentarz w linii 7 ma zadanie zwiększyć czytelność kodu. Na koniec zwróćmy uwagę na parametry obu środowisk **minipage**. Druga kolumna zawiera więcej tekstu, więc wskazaliśmy jedynie parametr t określający wyrównanie w pionie i jej szerokość. Wysokość tej kolumny jest wyznaczona na podstawie jej zawartości. W pierwszej kolumnie skorzystano z wyrównania tekstu do dołu, a wysokość ministrony została dobrana tak, aby imię i nazwisko uczonej znalazło się w połowie wysokości drugiej kolumny.

Innym ciekawym zastosowaniem ministron jest umieszczanie obok siebie tekstu i grafiki, na przykład przy podziale na dwie kolumny w jednej ministronie umieszczamy tekst, a w drugiej dołączamy grafikę. Do tematu składania obok siebie tekstu i grafiki wrócimy w podrozdziale 7.2.

Jeżeli umieszczamy środowisko **minipage** wewnątrz polecenia **\fbox** (aby uzyskać ramkę wokół ministrony), to polecenie **\fbox** dodaje margines o domyślnej szerokości 3 pt. Wielkość ta jest kontrolowana przez polecenie **\fboxsep**. Możemy ją modyfikować, np. **\setlength{\fboxsep}{0pt}**.

3.5 Pakiet tcolorbox

Pakiet `tcolorbox` zawiera środowisko oraz polecenia do umieszczania treści w eleganckich kolorowych ramkach. Wygląd ramki może być dostosowany do naszych potrzeb za pomocą opcji podawanych w układzie `klucz=wartość`. Ramki mogą być tworzone w trybie wierszowym, ale mogą zawierać również całe akapity tekstu, wzory matematyczne, tabele itp. W pakiecie znajduje się także kilka bibliotek z predefiniowanymi stylami rysowania ramek oraz biblioteka ułatwiająca przygotowywanie plakatów konferencyjnych (posterów).

```
\usepackage{tcolorbox}
\tcbuselibrary{theorems}
```

Kolorowa ramka mogąca obejmować wiele linii tekstu tworzona jest przez środowisko **`tcolorbox`**. Zawartość można podzielić na dwie części, górną i dolną, odseparowane poziomą linią, za pomocą polecenia **`\tcblower`**. Jeżeli nie użyjemy polecenia **`\tcblower`**, to cała zawartość składa się tylko z górnej części. Przykład dwuczęściowej ramki z domyślnymi ustawieniami pokazano na listingu 3.18. Gdybyśmy usunęli z kodu polecenie **`\tcblower`**, to uzyskalibyśmy dwa leżące jeden po drugim akapity bez oddzielającej je poziomej przerywanej linii i powiązanych z nią pionowych odstępów.

Listing 3.18. Przykład użycia środowiska **`tcolorbox`** z ustawieniami domyślnymi

```
\begin{tcolorbox}
Henryk Sienkiewicz urodził się 5 maja 1846 w Woli Okrzejskiej. Na
chrzcie otrzymał imiona Henryk, Adam, Aleksander, Pius. Zmarł 15
listopada 1916 r. w Vevey w Szwajcarii.
\tcblower
Był jednym z najpopularniejszych polskich pisarzy przełomu XIX i XX
wieku. Za całokształt twórczości otrzymał w 1905 r. Nagrodę Nobla
z literatury.
\end{tcolorbox}
```

Henryk Sienkiewicz urodził się 5 maja 1846 w Woli Okrzejskiej. Na chrzcie otrzymał imiona Henryk, Adam, Aleksander, Pius. Zmarł 15 listopada 1916 r. w Vevey w Szwajcarii.

Był jednym z najpopularniejszych polskich pisarzy przełomu XIX i XX wieku. Za całokształt twórczości otrzymał w 1905 r. Nagrodę Nobla z literatury.

Tabela 3.3. Wybrane opcje ramek z pakietu `tcolorbox`

Opcje	**Opis**
`title`	tworzy pasek tytułowy z podaną zawartością
`titlebox=invisible`	ukrywa tytuł (zostaje pusta przestrzeń)
`attach title to upper`	umieszcza tytuł na początku górnej części
`colframe`	kolor ramki
`colback`, `colbacktitle`	kolor tła, kolor tła dla tytułu
`coltitle`, `coltext`	kolor czcionki dla tytułu i tekstu (obie części)
`colupper`, `collower`	kolor czcionki dla części górnej i dolnej
`fontupper`, `fontlower`	tekst umieszczany przed zawartością części górnej (dolnej)
`fonttitle`	tekst umieszczany przed tytułem
`halign upper`, `halign lower`	sposób wyrównania tekstu w poziomie
`halign title`	sposób wyrównania tekstu w poziomie w tytule
`valign upper`, `valign lower`	sposób wyrównania tekstu w ponie
`width`, `height`	szerokość i wysokość ramki
`text width`	szerokość tekstu w górnej części
`text height`	wysokość tekstu w (całej) ramce
`toprule`, `bottomrule`	szerokość górnej i dolnej linii
`leftrule`, `rightrule`	szerokość lewej i prawej linii
`titlerule`	szerokość linii pod tytułem
`boxrule`	wspólna szerokość wszystkich linii
`arc`	wewnętrzny promień narożników
`boxsep`	dodatkowy odstęp wokół zawartości ramki
`left`, `lefttitle`, `leftupper` `leftlower`, `right`, `righttitle` `rightupper`, `rightlower`	lewy i prawy margines (całość i części ramki)
`top`, `bottom`	odległość między tekstem a ramką (góra i dół)
`toptitle`, `bottomtitle`	odległość między tytułem a ramką (góra i dół)
`middle`	odległość między tekstem a linią oddzielającą
`size`	rozmiar ramki, dopuszczalne wartości: `normal`, `title`, `small`, `fbox`, `tight`, `minimal`
`sharp corners`	ostre narożniki, wartości: `north`, `south`, `east`, `west`, `northwest`, `northeast`, `southwest`, `southeast`, `downhill`, `uphill`, `all`
`opacityframe`, `opacityback`	przezroczystość ramki i tła
`flush left` , `flush right` `center`	rozmieszczenie ramki w linii
`sidebyside=true`	poziomy układ ramki

W tabeli 3.3 przedstawiono najważniejsze opcje, które można wykorzystać do dopasowania wyglądu ramki. Opcje można umieszczać jako parametr opcjonalny środowiska **tcolorbox** (oddzielamy je przecinkami) lub jako parametr polecenia **\tcbset**. Opcje ustawione przez to polecenie zachowują ważność dla kolejnych ramek. Można je umieścić w preambule dokumentu, jeżeli chcemy stosować te same ustawienia dla wszystkich ramek. Przykład środowiska **tcolorbox** z ustawionymi wybranymi opcjami pokazano na listingu 3.19.

Listing 3.19. Przykład zastosowania opcji środowiska **tcolorbox** do zmiany wyglądu ramki

```
\begin{tcolorbox}[title=Henryk Sienkiewicz, halign title=center,
  fonttitle=\bfseries, colback=white, sharp corners=uphill]
Henryk Sienkiewicz urodził się 5 maja 1846 w Woli Okrzejskiej. Na
chrzcie otrzymał imiona Henryk, Adam, Aleksander, Pius.
Zmarł 15 listopada 1916 r. w Vevey w Szwajcarii.
\tcblower
Był jednym z najpopularniejszych polskich pisarzy przełomu XIX i XX
wieku. Za całokształt twórczości otrzymał w 1905 r. Nagrodę Nobla
z literatury.
\end{tcolorbox}
```

Henryk Sienkiewicz

Urodził się 5 maja 1846 w Woli Okrzejskiej. Na chrzcie otrzymał imiona Henryk, Adam, Aleksander, Pius. Zmarł 15 listopada 1916 r. w Vevey w Szwajcarii.

Był jednym z najpopularniejszych polskich pisarzy przełomu XIX i XX wieku. Za całokształt twórczości otrzymał w 1905 r. Nagrodę Nobla z literatury.

Użycie opcji `title` z niepustym tytułem skutkuje dodaniem belki tytułowej. Gdybyśmy chcieli uzyskać pustą belkę tytułową, należy dodać tytuł, a potem opcję `titlebox=invisible`. Opcja `fonttitle` została użyta, aby przed tytuł wstawić polecenie zmieniające formatowanie. Podobnie można użyć opcji `fontupper` i `fontlower` do zmiany formatowania górnej i dolnej części ramki. Wartość `uphill` dla opcji `sharp corners` spowodowała, że mamy ostry lewy dolny i prawy górny narożnik ramki. Użycie opcji `downhill` ustawia jako ostre lewy górny i prawy dolny narożnik. Pozostałe opcje z tej grupy mają raczej oczywiste znaczenie, ustawiają jako ostre narożniki po jednej stronie, tylko jeden z nich lub wszystkie (opcja `all`).

Ciekawe efekty można uzyskać, modyfikując szerokości linii ramki. Na listingu 3.20 ustawiono niezerową szerokość ramek tylko dla lewej i prawej strony. Zawartością pudełka jest używane już wcześniej wypunktowanie. W efekcie końcowym mamy dwa szare paski po obu stronach wypunktowania. Usunięcie opcji `width=10cm` spowoduje, że paski znajdą się przy marginesach.

Listing 3.20. Przykład zastosowania opcji środowiska **tcolorbox** do zmiany wyglądu ramki

```
\begin{tcolorbox}[colback=white, colframe=gray!70, boxrule=0pt,
  leftrule=8pt, rightrule=8pt, sharp corners=all, width=10cm]
\begin{enumerate}[itemsep=0pt]
\item 1903 Maria Skłodowska-Curie -- fizyka
\item 1905 Henryk Sienkiewicz -- literatura
\item 1911 Maria Skłodowska-Curie -- chemia
\item 1924 Władysław Reymont -- literatura
\end{enumerate}
\end{tcolorbox}
```

1. 1903 Maria Skłodowska-Curie – fizyka
2. 1905 Henryk Sienkiewicz – literatura
3. 1911 Maria Skłodowska-Curie – chemia
4. 1924 Władysław Reymont – literatura

W pakiecie dostępne jest również polecenie **\tcbsubtitle**`[]{}`, które dodaje podtytuł wewnątrz ramki. Jest to w rzeczywistości zagnieżdżona ramka, która dziedziczy część opcji związanych z kolorami i geometrią z otaczającej ramki, ale może być też modyfikowana poprzez opcje stylu `subtitle style`. Polecenia tego można używać wielokrotnie. Przykład wykorzystania tej możliwości pokazano na listingu 3.21. Styl podtytułu użyty został, aby wyśrodkować treść.

Listing 3.21. Przykład zastosowania polecenia **\tcbsubtitle**

```
\begin{tcolorbox}[title=Fizyka, colback=white, colframe=gray,
  halign title=center, fonttitle=\bfseries, width=7cm,
  subtitle style={halign=center}]
1903 Maria Skłodowska-Curie
\tcbsubtitle{Chemia}
1911 Maria Skłodowska-Curie
\tcbsubtitle{Literatura}
1905 Henryk Sienkiewicz\\
1924 Władysław Reymont
\end{tcolorbox}
```

Fizyka

1903 Maria Skłodowska-Curie

Chemia

1911 Maria Skłodowska-Curie

Literatura

1905 Henryk Sienkiewicz
1924 Władysław Reymont

Środowisko **tcolorbox** obsługuje opcję `sidebyside=true`. Jej użycie oznacza, że część górna i dolna są ułożone obok siebie jako lewa i prawa. Przykład pokazano na listingu 3.22. Domyślnie obie części mają taką samą szerokość. Opcja `lefthand width` została użyta, aby zmienić domyślny podział.

Listing 3.22. Przykład poziomego ułożenia części ramki

```
\begin{tcolorbox}[sidebyside=true, lefthand width=4cm, colback=white]
Maria Skłodowska-Curie\\
1867--1934
\tcblower
Dwukrotnie otrzymała Nagrodę Nobla: w 1903 r. z~fizyki za badania nad
zjawiskiem promieniotwórczości, a w 1911 r. z chemii za odkrycie
polonu i radu, wydzielenie czystego radu i badanie właściwości
chemicznych pierwiastków promieniotwórczych. Jest jedyną kobietą,
która otrzymała dwie nagrody w dwóch różnych dyscyplinach.
\end{tcolorbox}
```

Maria Skłodowska-Curie
1867–1934

Dwukrotnie otrzymała Nagrodę Nobla: w 1903 r. z fizyki za badania nad zjawiskiem promieniotwórczości, a w 1911 r. z chemii za odkrycie polonu i radu, wydzielenie czystego radu i badanie właściwości chemicznych pierwiastków promieniotwórczych. Jest jedyną kobietą, która otrzymała dwie nagrody w dwóch różnych dyscyplinach.

Pakiet `tcolorbox` zawiera również polecenie `\tcbox[]{}`. Polecenie to tworzy ramkę, której rozmiar zależny jest od wielkości parametru (zawartości ramki).

W poleceniu można użyć większości opcji z tabeli 3.3, poza opcjami związanymi z dolną częścią ramki, gdyż ta nie występuje przy ramkach tworzonych za pomocą **\tcbox**. Z poleceniem można używać opcji `on line`, która powoduje, że zachowuje się ono podobnie jak polecenie **\fbox**. Przykład takiego zastosowania polecenia **\tcbox** pokazano na listingu 3.23. Wszystkie ramki z listingu korzystają z tych samych opcji ustawionych za pomocą polecenia **\tcbset**.

Listing 3.23. Przykład zastosowania polecenia **\tcbox** z opcją `on line`

```
\tcbset{colback=white, boxsep=0pt, left=2pt, right=2pt,
  top=2pt, bottom=2pt, arc=1pt, on line}

Pierwsze Nagrody Nobla dla Polski otrzymali \tcbox{Maria
Skłodowska-Curie} w~1903 r. z fizyki, \tcbox{Henryk Sienkiewicz}
w 1905 r. z literatury, ponownie \tcbox{Maria Skłodowska-Curie}
w 1911 r. z chemii i \tcbox{Władysław Reymont} w 1924 r. z literatury.
```

Pierwsze Nagrody Nobla dla Polski otrzymali Maria Skłodowska-Curie w 1903 r. z fizyki, Henryk Sienkiewicz w 1905 r. z literatury, ponownie Maria Skłodowska-Curie w 1911 r. z chemii i Władysław Reymont w 1924 r. z literatury.

Po dołączeniu biblioteki `theorems` mamy do dyspozycji również polecenie **\tcboxmath**`[]{}`, którego można używać wewnątrz wzorów matematycznych. Możemy w ten sposób opakować w ramkę część wzoru, jak to pokazano na listingu 3.24 i we wzorze (3.2).

Listing 3.24. Przykład zastosowania polecenia **\tcboxmath**

```
\begin{equation}
\tcbset{boxsep=6pt}
\label{eq:tcboxmath}
\tcboxmath{p \Rightarrow q} \Longleftrightarrow
\tcboxmath{\neg p \vee q}
\end{equation}
```

$$\boxed{p \Rightarrow q} \Longleftrightarrow \boxed{\neg p \vee q} \tag{3.2}$$

Rozdział 4

Zarządzanie dokumentem

W rozdziałach 2 i 3 skupiliśmy się m.in. na kwestiach związanych najpierw z formatowaniem tekstu, a później akapitów. W tym rozdziale przejdziemy na wyższy poziom i spojrzymy na nasz dokument jak na cały projekt. Interesować nas będzie nie tylko struktura organizacyjna wewnątrz dokumentu (rozdziały, sekcje itp.), ale również organizacja plików projektu na dysku komputera, gdy ze względu na jego objętość zdecydujemy się podzielić go na części.

4.1 Standardowe klasy dokumentów

LaTeX zawiera cztery standardowe klasy dokumentów:

- `article` – artykuły, krótkie opracowania itp.,
- `report` – dłuższe opracowania, raporty techniczne, prace dyplomowe itp.,
- `book` – książki,
- `letter` – listy.

W zależności od wybranej klasy dokumentu inne są domyślne wartości parametrów opcjonalnych, a także występują różnice w zakresie organizacji struktury dokumentu. W tabeli 4.1 przedstawiono dostępne parametry opcjonalne klas. Wybrane parametry umieszcza się w nawiasie kwadratowym i oddziela przecinkami (zob. np. listing 1.1). Użycie parametrów opcjonalnych nie jest konieczne, bo dla każdej z klas zdefiniowano zestaw wartości domyślnych.

Klasy standardowe są dostępne w każdej instalacji LaTeX-a i możemy ich używać bez potrzeby pobierania dodatkowych plików. Za pomocą opcji z tabeli 4.1 oraz dodatkowych pakietów możemy je dostosować do własnych potrzeb, a nawet użyć ich do zdefiniowania własnej klasy.

Tabela 4.1. Opcje klas dokumentów

Opcje	**Znaczenie**
`10pt`, `11pt`, `12pt`	rozmiar podstawowego tekstu w dokumencie
`letterpaper`, `a4paper` `b5paper`,...	rozmiar papieru
`fleqn`	eksponowane wzory matematyczne dosuwane do lewej strony (zamiast domyślnego centrowania)
`leqno`	numery wzorów matematycznych po lewej stronie (zamiast domyślnie po prawej)
`titlepage`, `notitlepage`	umieszczanie tytułu (lub nie) na oddzielnej stronie
`oneside`, `twoside`	wydruk jedno- lub dwustronny
`onecolumn`, `twocolumn`	skład jedno- lub dwukolumnowy
`landscape`	poziomy układ strony
`openright`, `openany`	tytuły rozdziałów umieszczane na stronie nieparzystej lub dowolnej

Klasa article

W przypadku klasy `article` domyślnymi opcjami są `10pt`, `letterpaper`, `oneside`, `onecolumn` i `notitlepage`. Przykładową preambułę dokumentu składanego z użyciem klasy `article` wraz początkiem dokumentu pokazano na listingu 4.1.

Listing 4.1. Przykład dokumentu składanego z użyciem klasy `article`

```
\documentclass[a4paper]{article}
\usepackage[polish]{babel}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{graphicx}
\usepackage{newtx}

\title{Wprowadzenie do \LaTeX-a}

\author{Jan Techowski\\
\small Katedra Informatyki Stosowanej\\
\small AGH w Krakowie\\
\small \texttt{techowski@agh.edu.pl}}

\date{07.02.2025}

\begin{document}
```

```
18 \maketitle
19
20 \begin{abstract}
21 W artykule przedstawiono praktyczne wprowadzenie do przygotowania
22 artykułu naukowego z wykorzystaniem systemu \LaTeX\dots
23 \end{abstract}
```

Wprowadzenie do LaTeX-a

Jan Techowski
Katedra Informatyki Stosowanej
AGH w Krakowie
techowski@agh.edu.pl

07.02.2025

Streszczenie

W artykule przedstawiono praktyczne wprowadzenie do przygotowania artykułu naukowego z wykorzystaniem systemu LaTeX. . .

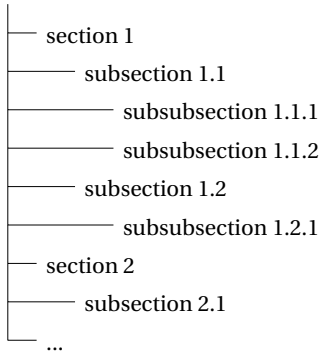
Jeżeli zamierzamy wygenerować tytuł dokumentu (polecenie **\maketitle** w linii 18), to w preambule powinniśmy dodać polecenia **\title** i **\author**. Jak pokazano na listingu (linie 10–13), polecenie **\author** można wykorzystać do podania, poza imieniem i nazwiskiem, afiliacji autora – polecenie dopuszcza łamanie linii z użyciem \\. W preambule możemy dodać również polecenie **\date** (linia 15). Jeżeli je pominiemy, to LaTeX użyje aktualnej daty systemowej. Jeżeli chcemy całkowicie pominąć datę, to dodajemy polecenie z pustym parametrem **\date**{}. Charakterystycznym elementem artykułu jest streszczenie. Przygotowujemy je z zastosowaniem środowiska **abstract** (zobacz linie 20–23). W omawianym przykładzie zamieszczono tylko pierwsze zdanie streszczenia. Efekt kompilacji tej części artykułu widzimy poniżej kodu źródłowego. Wyświetlana nazwa środowiska **abstract** – „Streszczenie", „Abstract" itd. – zależy od użytej opcji pakietu babel, wskazującej język dokumentu.

W tytule dokumentu możemy umieścić polecenie **\thanks**{}. Tworzy ono przypis do tytułu. W artykułach naukowych wykorzystuje się je zazwyczaj do wskazania projektu naukowego, w ramach którego powstał artykuł, lub wskazania źródła finansowania badań.

Poniżej środowiska **abstract** umieszczamy właściwą treść naszego artykułu. Warto zwrócić uwagę, że nie musimy generować tytułu, przygotowywać streszczenia itd. Jeżeli chcemy wykorzystać klasę article do przygotowania notatek

z ostatniej burzy mózgów w zespole, to być może te elementy nie będą potrzebne. Zobacz listing 1.1 – w naszym pierwszym przykładzie nie korzystaliśmy z tych możliwości.

Jeżeli przygotowujesz artykuł naukowy i korzystasz z klasy pobranej ze strony czasopisma, to sprawdź w jego przykładowym pliku `tex` (powinien być w pobranym przez Ciebie archiwum), w jaki sposób powinno się podać tytuł, nazwiska autorów i afiliacje. Czasopisma często stosują zmodyfikowane polecenia do wprowadzania takich danych.



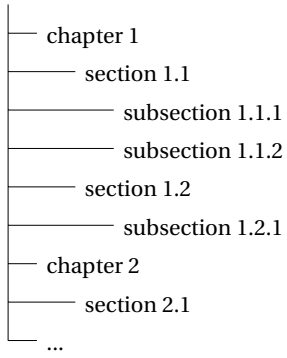
Rys. 4.1. Organizacja struktury artykułu

Zawartość artykułu dzielimy na sekcje (polecenie **\section**{}), podsekcje (**\subsection**{}) i podpodsekcje (**\subsubsection**{}). Każde z tych poleceń przyjmuje jako parametr tytuł danej części (może też przyjąć parametr opcjonalny zawierający wpis do spisu treści, jeżeli ma się w nim znaleźć coś innego niż tytuł danej części). Części te są numerowane – przykład pokazano na rys. 4.1 – i stanowią poziomy od 1 do 3 organizacji zawartości artykułu. Dopuszczalny jest jeszcze podział na części (polecenie **\part**{}). Jest to tzw. poziom 0 i jest rzadko używany. Z poleceniem **\section** związany jest licznik `section` (analogicznie z poleceniami **\subsection** i **\subsubsection**). Gdyby zaszła taka potrzeba, możemy pobrać jego aktualną wartość za pomocą polecenia **\thesection**. Liczniki są na wstępie zerowane. Dowolne z poleceń **\section**, **\subsection**, **\subsubsection** zwiększa o 1 swój licznik i zeruje liczniki na niższym poziomie. Polecenia te występują również w wersji z gwiazdką, np. **\section***{}. W takim przypadku sekcja jest nienumerowana, a liczniki nie są modyfikowane.

Jeżeli w klasie `article` umieścimy numerowane wzory, rysunki, tabele itd., to będą one numerowane w sposób ciągły, tj. będą oznaczane kolejnymi liczbami naturalnymi 1, 2, 3,.... Oczywiście każdy typ umieszczanych obiektów jest numerowany niezależnie od innych typów.

Klasa report

Klasa `report` jest bardzo dobrym wyborem przy przygotowywaniu pracy dyplomowej. Jej opcjami domyślnymi są `10pt`, `letterpaper`, `oneside`, `onecolumn`, `openany` i `titlepage`. W przypadku pracy dyplomowej warto wybrać opcje `11pt`, `a4paper` i `twoside`. Podstawowa różnica między klasą `article` i `report` to organizacja wewnętrznej struktury dokumentu. W klasie `report` (oraz `book`) trzy podstawowe numerowane poziomy organizacji tekstu tworzone są poleceniami: **`\chapter`**`{}`, **`\section`**`{}` i **`\subsection`**`{}`. Przykład organizacji struktury raportu/książki pokazano na rys. 4.2. Podobnie jak w przypadku klasy `article`, możemy stosować również polecenie **`\part`**, które bywa używane przy większych książkach. Możemy również korzystać z polecenia **`\subsubsection`**, ale nie generuje ono numerowanych części dokumentu.



Rys. 4.2. Organizacja struktury raportu/książki

W klasie `report` możemy skorzystać z poleceń **`\title`**, **`\author`**, **`\date`** oraz środowiska **`abstract`**. Ich użycie w kodzie źródłowym wygląda dokładnie tak, jak pokazano na listingu 4.1. Różnicę zobaczymy dopiero w pliku `pdf` – tytuł, autor i data zostaną umieszczone na pierwszej stronie, streszczenie na drugiej, a reszta dokumentu będzie składana od strony trzeciej. Jeżeli skorzystamy z opcji `notitlepage`, to tytuł, autor, data i streszczenie zostaną umieszczone na pierwszej stronie, a reszta dokumentu będzie składana od drugiej strony – o ile użyjemy polecenia **`\chapter`**, bo każdy rozdział jest składany od nowej strony.

W przypadku wydruku jednostronnego każda strona ma tak samo ustawiony prawy i lewy margines. Jeżeli wybierzemy opcję `twoside`, to mówimy

już o marginesach wewnętrznych i zewnętrznych – zazwyczaj margines wewnętrzny jest szerszy, jeżeli wydruk ma być w jakiś sposób scalony w formę książkową. Wybór wydruku dwustronnego wymusza podjęcie decyzji, czy chcemy użyć opcji `openany`, czy `openright`. Opcja `openright` oznacza, że nowy rozdział będzie zawsze zaczynał się na stronie prawej (nieparzystej) – jest to podejście często stosowane w drukowanych książkach. W efekcie mogą pojawić się puste strony (wakaty), gdy wcześniejszy rozdział kończy się na stronie nieparzystej. W przypadku opcji `openany` nowy rozdział zaczynany jest od kolejnej strony, nieważne, czy jest to strona parzysta, czy nieparzysta.

Jeżeli w klasie `report` umieścimy numerowane wzory, rysunki, tabele itd., to będą one numerowane w sposób ciągły w obrębie rozdziału, tj. np. pierwszy rysunek w rozdziale 3. otrzyma numer 3.1, kolejny 3.2 itd.

Klasa book

Opcjami domyślnymi dla klasy `book` są `10pt`, `letterpaper`, `twoside`, `onecolumn`, `openright` i `titlepage`. Widzimy więc, że w porównaniu z klasą `report` domyślnie mamy ustawiony druk dwustronny i rozpoczynanie rozdziałów od strony nieparzystej. Podobnie jak w poprzednich klasach, używamy tutaj poleceń **`\title`**, **`\author`** i **`\date`**, ale dla klasy `book` niedostępne jest środowisko **`abstract`**. Organizacja struktury dokumentu (rozdziały, sekcje itd.) jest taka jak dla klasy `report` (zob. rys. 4.2). Również numerowane wzory, rysunki, tabele itd. są domyślnie numerowane w sposób ciągły w obrębie rozdziału. Dodatkowo klasa `book` oferuje trzy polecenia, które pozwalają globalnie zarządzać zawartością książki, w szczególności pozwalają wyodrębnić wstępną część książki i zastosować dla niej oddzielną numerację. Przedstawiono je w tabeli 4.2. Przykład ich zastosowania omówimy w podrozdziale 4.3.

Tabela 4.2. Polecenia do zarządzania globalnym układem książki

Polecenie	Znaczenie
`\frontmatter`	umieszczone w tej części rozdziały nie mają numerów, ale są umieszczane w spisie treści, numeracja stron jest prowadzona małymi cyframi rzymskimi
`\mainmatter`	umieszczone w tej części rozdziały są numerowane, licznik stron `page` jest zerowany, numeracja stron jest prowadzona liczbami arabskimi
`\backmatter`	umieszczone w tej części rozdziały nie mają numerów, ale są umieszczane w spisie treści, numeracja stron jest kontynuowana liczbami arabskimi

Klasa letter

Ostatnią z podstawowych klas jest klasa **letter**. Zdecydowanie różni się ona od trzech wcześniej omawianych, na przykład nie zakłada struktury wewnętrznej dokumentu, tj. rozdziałów, sekcji itd. Klasa została zaprojektowana z myślą pisania pojedynczych listów, jak również obsługiwaniu korespondencji seryjnej. W jednym dokumencie możemy umieścić wiele listów. Każdy z nich składany jest w oddzielnym środowisku **letter**.

Na listingu 4.2 pokazano przykład z pojedynczym listem. W preambule użyto poleceń **\address**{}, **\signature**{} i **\date**{}, które będą współdzielone przez wszystkie listy, jeśli zdecydujemy się umieścić w dokumencie więcej niż jeden. Polecenie **\address** (linia 7) pozwala zdefiniować adres nadawcy. Dopuszcza ono łamanie linii z użyciem polecenia \\. Polecenie **\signature** (linia 10) służy do zdefiniowania podpisu, który zostanie umieszczony na końcu listu, zaś **\date** (linia 12) do określenia daty.

Argumentem obowiązkowym środowiska **letter** jest adresat listu. Wewnątrz środowiska można użyć polecenia **\opening**{} do złożenia nagłówka listu (linia 18), **\closing**{} do zakończenia listu (linia 22) – pod zakończeniem umieszczany jest argument polecenia **\signature**, **\cc**{} do wskazania osób otrzymujących kopię listu (linia 23) i **\ps**{} do wstawienia *post scriptum*. Wynik kompilacji rozważanego listu pokazano poniżej kodu źródłowego. Polecenie **\longindentation** (linia 15) zostało użyte do przesunięcie zakończenia i podpisu bardziej w lewo. Gdybyśmy chcieli, aby były one dosunięte do prawej strony, należy wybrać wartość 0.

Wewnątrz środowiska **letter** możemy umieścić dowolną liczbę akapitów z tekstem, tabele, rysunki, wzory matematyczne itd. Jeżeli chcemy wygenerować więcej niż jeden list z tym samym nadawcą, to wystarczy umieścić w dokumencie kolejne środowisko **letter**.

Listing 4.2. Przykład dokumentu składanego z użyciem klasy `letter`

```
1  \documentclass[a4paper]{letter}
2  \usepackage[polish]{babel}
3  \usepackage[utf8]{inputenc}
4  \usepackage[T1]{fontenc}
5  \usepackage{newtx}
6
7  \address{Jan Techowski\\AGH w Krakowie\\
8  Al. Mickiewicza 30\\30-059 Kraków}
9
10 \signature{Jan Techowski}
11
12 \date{Kraków, 11 lutego 2025 r.}
```

```
13
14 \begin{document}
15 \longindentation=9.5cm
16
17 \begin{letter}{Sz.P. Ewa Iksińska}
18 \opening{Szanowna Pani}
19 Z~radością informuję, że przyznano Pani stanowisko asystenta stażysty
20 w Katedrze Informatyki Stosowanej AGH. Staż będzie realizowany
21 w semestrze letnim pod moją opieką naukową.
22 \closing{Z poważaniem}
23 \cc{Kierownik KIS}
24 \end{letter}
25 \end{document}
```

Jan Techowski
AGH w Krakowie
Al. Mickiewicza 30
30-059 Kraków

Kraków, 11 lutego 2025 r.

Sz.P. Ewa Iksińska

Szanowna Pani

Z radością informuję, że przyznano Pani stanowisko asystenta stażysty w Katedrze Informatyki Stosowanej AGH. Staż będzie realizowany w semestrze letnim pod moją opieką naukową.

Z poważaniem

Jan Techowski

Do wiadomości: Kierownik KIS

Jeżeli potrzebujemy większej kontroli nad kompozycją listów składanych w LaTeX-u, to warto sięgnąć po pakiety oferujące alternatywne klasy opracowane w tym celu. Przykładem jest pakiet `newlfm`. Oferuje on kilka stylów składania listów i sporo opcji zarządzających składem tych dokumentów. Szczegóły można znaleźć w dokumentacji pakietu.

4.2 Etykiety

Jak już wspomniałem w rozdziale 1, LaTeX zawiera bardzo wygodny mechanizm odsyłaczy, dzięki któremu bezbłędnie możemy odwoływać się do obiektów umieszczonych w dokumencie, jak rozdziały, równania, rysunki itd. Etykieta jest definiowana z użyciem polecenia **\label{}**, którego parametrem jest definiowana nazwa. Z zasady etykietujemy wszystkie obiekty, do których można się odwoływać. Po pierwsze polecenie **\label** powinniśmy umieścić bezpośrednio po poleceniach definiujących numerowane części dokumentu, tj. po poleceniach **\chapter**, **\section** itd. W przypadku obiektów typu rysunek czy tabela polecenie **\label** umieszczamy bezpośrednio po poleceniu **\caption**. W przypadku środowisk składających definicje, twierdzenia, równania matematyczne itp. polecenie **\label** umieszczamy zazwyczaj bezpośrednio po poleceniu otwierającym środowisko (**\begin{}**).

Etykiety muszą być unikatowe w skali całego dokumentu. Przy większych dokumentach może to być kłopotliwe, np. w tej książce zdefiniowano kilkaset etykiet. Problem jest dwojakiego rodzaju, po pierwsze musimy wymyślać unikatowe etykiety, po drugie, gdy tworzymy odwołanie, środowisko pracy (np. Overleaf) wyświetli nam listę wszystkich zdefiniowanych etykiet. Przeglądanie listy 300 etykiet nie jest wygodne. Jedno z zalecanych rozwiązań to stosowanie prefiksów etykiet, np. możemy przyjąć, że wszystkie etykiety rysunków zaczynają się od prefiksu `fig:`. W przypadku wyboru etykiety wpisujemy ten prefiks i na liście pozostaną tylko etykiety rysunków. Ułatwia to też tworzenie etykiet. Możemy mieć na przykład etykiety `def:abc`, `fig:abc`, `eq:abc`, gdzie pierwsza dotyczy definicji obiektu `abc`, druga powiązanego z definicją rysunku, a ostatnia wzoru matematycznego umieszczonego w definicji. Przykładowy zestaw prefiksów zawarto w tabeli 4.3. Są one stosowane m.in. w przykładach w tej książce.

Tabela 4.3. Przykładowy zestaw prefiksów

Prefiks	Typ obiektu
`cha:`	rozdział (**\chapter**)
`sec:`	sekcja/podrozdział (**\section**)
`ssec:`	podsekcja (**\subsection**)
`fig:`	rysunek
`tab:`	tabela
`alg:`	algorytm
`lst:`	listing
`eq:`	wzór matematyczny
`def:`	definicja

W przypadku definiowania etykiet warto unikać znaków diakrytycznych i spacji. Ich użycie może generować problemy zależnie od używanego środowiska lub kodowania dokumentu. Dobrym zamiennikiem dla spacji są dwukropki. Do zdefiniowanych etykiet możemy odwoływać się na dwa sposoby. Polecenie `\ref{}` (parametrem jest etykieta) na etapie kompilacji jest zamieniane na oznaczenie elementu, do którego się odwołujemy, np. odwołanie się do pierwszego rysunku w trzecim rozdziale zwróci przy domyślnych ustawieniach klasy `book` wartość `3.1`, a do drugiego rysunku w dodatku A – `A.2`. Z kolei polecenie `\pageref{}` (parametrem jest etykieta) zwróci numer strony, na której dany element jest umieszczony. Jest to przydatne przy tzw. dalekich odwołaniach. Jeżeli chcemy odwołać się do definicji, którą zamieściliśmy kilka rozdziałów wcześniej, to aby ułatwić czytelnikowi jej odnalezienie, stosujemy konstrukcję, np. `zob. definicja~\ref{def:abc} na str.~\pageref{def:abc}`. Po skompilowaniu dokumentu wygeneruje to treść, np. *zob. definicja 3.1 na str. 62*.

Warto wspomnieć, że w przypadku pracy z lokalną dystrybucją LaTeX-a i zmian w definicjach lub rozmieszczeniu etykiet do uzyskania poprawnych wartości odwołań wymagana jest dwukrotna kompilacja dokumentu. Jeżeli używamy środowiska Overleaf, jest to robione automatycznie.

Na zakończenie tego podrozdziału rozważmy następującą sytuację. Przygotowujesz większy dokument w LaTeX-u (niech będzie to praca doktorska) i Twój promotor chce, abyś zmienił kolejność rozdziałów 4 i 5. Jeżeli stosowałeś system etykietowania, zadanie to zajmie chwilę. Wystarczy zmienić położenie tych rozdziałów w kodzie źródłowym i dwukrotnie skompilować dokument. Wszystko zostaje poprawnie ponumerowane bez żadnego wysiłku z Twojej strony. Wyobraź sobie teraz, że zamiast etykiet wpisywałeś numerki, a w tych rozdziałach jest sporo rysunków, tabel, równań matematycznych. Ile czasu zajmie korekta i ile błędów zrobisz przy poprawianiu odwołań?

4.3 Duże projekty

Jeżeli składamy artykuł naukowy albo przygotowujemy nieduży raport, to z dużym prawdopodobieństwem nasz projekt będzie zawierał jeden plik źródłowy LaTeX-a. Przy nieco większych projektach, jak praca dyplomowa, doktorska czy książka, warto podzielić projekt na kilka plików. Niezależnie od sytuacji warto przyjąć zasadę: *każdy projekt w oddzielnym katalogu.*

Jeżeli w naszym projekcie planujemy dodawać grafikę z plików zewnętrznych, to w katalogu projektu tworzymy podkatalog na grafikę – u mnie najczęściej jest to podkatalog `img`. Dodatkowo w preambule dodajemy wtedy polecenie `\graphicspath{{./img/}}`, aby wskazać ścieżkę do katalogu z grafiką.

W podanym przykładzie ścieżka ma notację uniksową. W razie potrzeby dostosuj ją do swojego systemu operacyjnego.

W projektach, w których pojawiają się rozdziały, osobiście stosuję zasadę: *każdy rozdział w oddzielnym pliku*. W takich przypadkach w oddzielnych plikach lądują nawet bardzo krótkie części projektu, jak np. przedmowa czy jednostronicowe dodatki. Dlaczego warto podzielić taki projekt na części?

- Optymalizujemy czas kompilacji – gdy pracujemy nad konkretnym rozdziałem, możemy ograniczyć się do kompilacji zmian w jednym pliku. Przy większych projektach to oszczędza naprawdę sporo czasu.
- Łatwiej pracuje się z plikiem źródłowym, który jest mniejszy. Przeszukiwanie pliku zawierającego kilka tysięcy linii kodu bywa uciążliwe.
- Łatwo reorganizować strukturę projektu, np. zmiana kolejności dwóch rozdziałów książki zajmie tylko kilka sekund.

Każdy projekt LaTeX-a musi mieć plik główny. Jest to ten dokument, w którym znajduje się preambuła i do którego w przypadku, gdy projekt jest podzielony na wiele plików, podłączamy inne pliki. Jeżeli generujemy nowy projekt w Overleaf, to domyślnie plik ten zostanie nazwany `main.tex`, przy czym możemy zmienić nazwę na taką, jaka nam odpowiada. Na listingu 4.3 pokazano przykład pliku głównego dla klasy `book`. Lista poleceń **`\usepackage`** została pokazana w minimalnym zestawieniu. Komentarz w linii 7 to taka sugestia, że tych pakietów dodajemy zazwyczaj dużo więcej – zależnie od tego, czego potrzebujemy. Preambuła zawiera też często dużo poleceń modyfikujących standardowe ustawienia. W dalszej części książki będą prezentowane fragmenty kodu, które są umieszczane w preambule. Przykład szkieletu dla pliku zawierającego pojedynczy rozdział pokazano na listingu 4.4.

Listing 4.3. Przykład pliku głównego dla klasy `book`

```
1  \documentclass[a4paper,11pt,fleqn]{book}
2  \usepackage[polish]{babel}
3  \usepackage[utf8]{inputenc}
4  \usepackage[T1]{fontenc}
5  \usepackage{graphicx}
6  \usepackage{newtx}
7  %...
8
9  \title{Wprowadzenie do \LaTeX-a}
10 \author{Jan Techowski}
11 \date{2025}
12
13 \begin{document}
14 \maketitle
```

```
15 \frontmatter
16 \setcounter{page}{5}
17 \tableofcontents
18 \cleardoublepage
19
20 \mainmatter
21 \include{wprowadzenie}
22 \include{tekst}
23 \include{tabele}
24 \include{matematyka}
25 \include{podsumowanie}
26 \appendix
27 \include{overleaf}
28 \include{texstudio}
29
30 \backmatter
31 \include{polecenia}
32 \bibliographystyle{plain}
33 \bibliography{latex}
34 \end{document}
```

Przeanalizujmy dokładnie, co znalazło się w środowisku **document**. W linii 14 mamy znane już polecenie, które generuje stronę tytułową na podstawie danych z preambuły (autor, tytuł i data). W kolejnej linii umieszczono polecenia **\frontmatter**, które wskazuje na początek wstępnej części książki. W tej części umieszcza się zazwyczaj podziękowania, przedmowę i różnego rodzaju spisy (treści, rysunków, tabel itp.). W podanym przykładzie (na razie) umieszczono tylko spis treści. Jest on generowany poleceniem **\tableofcontents** (linia 17). We wcześniejszej linii zmodyfikowaliśmy wartość licznika stron. Pierwsza strona spisu treści zostanie umieszczona na stronie V – pierwsza strona jest tytułowa, druga to wakat, na trzeciej będzie dedykacja i czwarta ponownie jest wakatem. W linii 18 użyto polecenia **\cleardoublepage**, którego zadaniem jest rozpoczęcie składu kolejnego „rozdziału" od strony nieparzystej. LaTeX oferuje również polecenie **\clearpage**, które rozpoczyna skład od kolejnej strony. Polecenie **\cleardoublepage** to przygotowanie pod dodanie po nim np. spisu rysunków. Jeżeli dopuszczamy, że taki spis może zaczynać się na stronie parzystej, to wystarczy użyć polecenia **\clearpage**. Warto tutaj dodać, że tytuły spisu treści, spisu rysunków, przedmowy itp. będą składane taką samą czcionką jak tytuły rozdziałów, ale nie będą numerowane (ale można je dołączyć do spisu treści).

Polecenie **\mainmatter** rozpoczyna główną część książki. Jeżeli przyjęliśmy zasadę, że każdy rozdział jest umieszczony w oddzielnym pliku, to część ta składa się przede wszystkim z serii poleceń **\include**{}. Polecenie to przyjmuje jako parametr nazwę pliku `tex` (bez rozszerzenia) i dołącza go we wskazanym

miejscu, wstawiając przed i po zawartości pliku łamanie strony. Ponieważ zostaliśmy przy domyślnych ustawienia klasy `book`, to każdy rozdział będzie rozpoczynał się od strony nieparzystej. Zatem linie 21–25 oznaczają dołączenie zawartości plików `wprowadzenie.tex`, `tekst.tex` itd.

Polecenie **`\appendix`** (linia 26) oznacza, że kolejne dołączane rozdziały (struktura dalej taka jak na listingu 4.4) będą traktowane jako dodatki. W naszym przykładzie będziemy więc mieć rozdziały od 1 do 5 i dodatki A i B.

Polecenie **`\backmatter`** (linia 30) rozpoczyna ostatnią część książki. Numeracja stron jest kontynuowana, ale dalsze rozdziały nie są już numerowane. W tym przykładzie przyjęto, że dołączamy jeszcze spis popularnych poleceń LaTeX-a (linia 31), ale nie będzie on już traktowany jako dodatek. Linie 32–33 odpowiadają za ustalenie stylu, w jakim składana będzie bibliografia, i wskazanie pliku z rekordami BibTeX-a. Składaniu bibliografii poświęcony jest rozdział 9. Na końcu często umieszcza się także indeks haseł.

Listing 4.4. Przykład pliku z rozdziałem – klasa `book` lub `report`

```
1 \chapter{Wprowadzenie}
2 \label{cha:wprowadzenie}
3
4 % treść rozdziału
```

Na wstępie do tego podrozdziału wskazałem kilka zalet podziału projektu na części. Weźmy za przykład listing 4.3. Jeżeli chcemy zmienić kolejność rozdziałów, wystarczy zmienić kolejność dołączania ich do pliku głównego. Jeżeli pracujemy w danym momencie z konkretnym rozdziałem lub rozdziałami, to możemy w preambule dodać polecenie **`\includeonly{}`**. Argumentem jest nazwa pliku lub plików, na których się skupiamy (nazwy bez rozszerzeń, oddzielone przecinkami, bez spacji). Przy wywołaniu kompilatora kompilowane będą tylko wskazane pliki. Znacznie przyspiesza to pracę w porównaniu z wielokrotnym kompilowaniem na przykład całej książki.

Poza poleceniem **`\include`** LaTeX zawiera polecenie **`\input{}`**. Również tutaj dołączana jest zawartość wskazanego pliku, ale bez dodawania łamania stron. Polecenie to jest przydatne, jeżeli chcemy np. w oddzielnym pliku złożyć długą tabelę lub rysunek z użyciem pakietu `tikz`.

4.4 Marginesy

Jeżeli przygotowujesz dokument, stosując klasę dostarczoną przez przyszłego wydawcę, to nie powinieneś ingerować w ustawienia tej klasy (często jest to wprost zaznaczone w przykładowym pliku `tex`). Jeżeli korzystasz ze

standardowych klas lub nie planujesz publikować dokumentu, to można dopasować go do indywidualnych oczekiwań. Wcześniej pisałem o wpływie opcji polecenia **\documentclass** na skład dokumentu oraz o możliwości doboru czcionek, które w nim wykorzystamy. W tym podrozdziale porozmawiamy o dodatkowych pakietach, które można wykorzystać, aby ustawić marginesy lub sprawdzić ich ustawienia. Zaczniemy od eksperymentu z klasą `article` pokazanego na listingu 4.5. Wynik kompilacji tego kodu pokazano na rys. 4.3.

```
\usepackage{layouts}
```

Listing 4.5. Zastosowanie pakietu `layouts` do pozyskania informacji o układzie strony

```
1  \documentclass[a4paper]{article}
2  \usepackage[utf8]{inputenc}
3  \usepackage[T1]{fontenc}
4  \usepackage[polish]{babel}
5  \usepackage{newtx}
6  \usepackage{layouts}
7
8  \begin{document}
9  \printinunitsof{mm}{\pagevalues}
10 \pagediagram
11 \end{document}
```

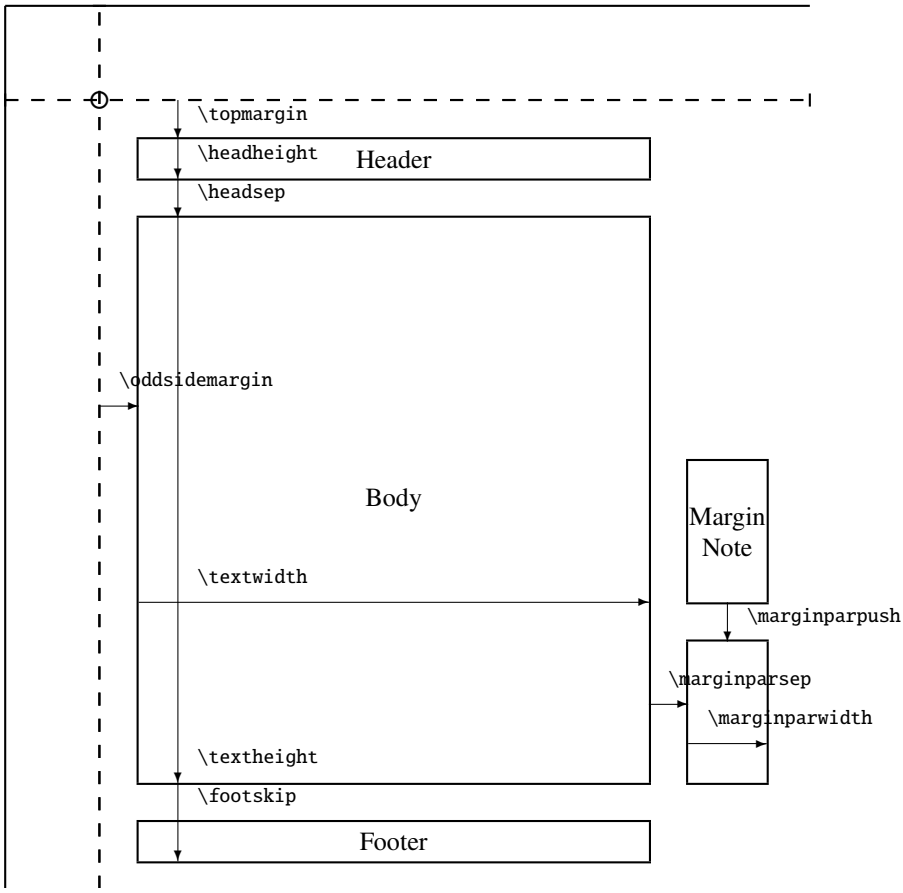
W przykładzie korzystamy z pakietu `layouts`, który umożliwia generowanie diagramów dotyczących składu (zob. też rys. 3.1). W tym przykładzie generujemy układ strony (linia 10, polecenie **\pagediagram**) Diagram pomoże nam zrozumieć informacje zwracane przez polecenie **\pagevalues** (linia 9), które dla ułatwienia skonwertowaliśmy na milimetry. Wybraliśmy rozmiar papieru A4 i informację tę widzimy w pierwszej linii wydruku – kartka papieru 210×297 mm. Wszystkie pozostałe dane to informacje o domyślnym układzie strony dla klasy `article`. Wartość widoczna przy poleceniu **\textheight** pokazuje, że nasz obszar do zapełnienia tekstem to „tylko" nieco ponad 210 mm. Poza nim zostaje prawie 87 mm. Z drugiej strony dostępna szerokość (**\textwidth**) to „tylko" nieco ponad 121 mm. Poza nią zostaje prawie 89 mm.

Ile w takim razie wynoszą poszczególne marginesy? To, co widzimy na obrazku, to albo wydruk jednostronny, albo prawa strona przy wydruku dwustronnym. Czyli wielkość **\oddsidemargin** związana jest z lewym lub wewnętrznym marginesem, a wielkość **\evensidemargin** z prawym lub zewnętrznym. Tutaj mamy domyślny wydruk jednostronny, w związku z czym lewy margines ma

Actual page layout values.

\paperheight = 296.9965mm	\paperwidth = 209.99753mm
\hoffset = 0mm	\voffset = 0mm
\evensidemargin = 18.9786mm	\oddsidemargin = 18.62715mm
\topmargin = 5.97475mm	\headheight = 4.21747mm
\headsep = 8.78639mm	\textheight = 210.1705mm
\textwidth = 121.25221mm	\footskip = 10.54367mm
\marginparsep = 3.86601mm	\marginparpush = 1.75728mm
\columnsep = 3.51456mm	\columnseprule = 0mm
1em = 3.51456mm	1ex = 1.58154mm

The circle is at 1 inch from the top and left of the page. Dashed lines represent (\hoffset + 1 inch) and (\voffset + 1 inch) from the top and left of the page.



Rys. 4.3. Wydruk informacji o ustawieniach strony

1 cal (25,4 mm) + **\oddsidemargin**, czyli około 44 mm, a prawy **\evensidemargin** + 1 cal, czyli około 44,4 mm. Górny margines to 1 cal + **\topmargin** + **\headheight** + **\headsep**, co daje około 44 mm. W końcu, aby obliczyć dolny margines, musimy od wysokości strony odjąć górny margines i **\textheight**, co daje około 43 mm. Widać, że przy ustawieniach domyślnych spora część strony nie jest wykorzystana. To wygląda zawile, ale chodzi tylko o to, aby wiadomo było, jak sprawdzić, czy wszystko jest w porządku z naszymi ustawieniami.

Jeżeli do swojego dokumentu dodasz kod z linii 6 (preambuła) oraz 9–10, to możesz zobaczyć, jak wyglądają bieżące ustawienia w Twoim dokumencie.

Do ustawienia własnych marginesów można wykorzystać pakiet `geometry`. Przykład ustawień pokazano na listingu 4.6. Polecenie **\geometry** umieszczamy w preambule.

```
\usepackage{geometry}
```

Listing 4.6. Zastosowanie pakietu `geometry` do ustawienia marginesów

```
\usepackage{geometry}
\geometry{left=2cm, right=2cm, top=2cm, bottom=2.5cm,
  bindingoffset=1cm}
```

Tym razem kod jest zdecydowanie bardziej zrozumiały niż w poprzednim przykładzie. Chcemy, aby marginesy lewy, prawy i górny miały po 2 cm, a margines dolny 2,5 cm. Dodatkowo chcemy przeznaczyć 1 cm na scalenie wydruku (zszycie, klejenie itp.). Jakie informacje uzyskamy z polecenia **\pagevalues** po dodaniu do naszego wcześniejszego przykładu kodu z listingu 4.6? Wartość **\evensidemargin** wynosi -5,4 mm. Jeżeli dodamy do tego cal, to otrzymamy w przybliżeniu 2 cm (prawy margines). Wartość **\oddsidemargin** wynosi 4,6 mm. Jak dodamy do tego cal, otrzymamy około 3 cm (lewy margines + rezerwa na scalenie). W podobny sposób można sprawdzić, że wszystko pasuje, jeśli chodzi o wysokość. Jak widzimy, pakiet `geometry` sprytnie przeliczył wielkości LaTeX-a, aby ustawienia strony pasowały do naszych oczekiwań.

Jeżeli założyliśmy wydruk dwustronny, to zamiast opcji `left` i `right` użyjemy `inner` i `outer`. Jeżeli chcemy ustawić wszystkie marginesy na tę samą wartość, wystarczy ustawić opcję `margin`, np. `margin=2cm`. Jeżeli z pary przeciwległych marginesów określimy tylko jeden, to drugi zostanie wyliczony z zachowaniem proporcji `top:bottom = 2:3`, `left:right = 1:1` i `inner:outer = 2:3`.

Pakiet przyjmuje również opcje `width` i `height` rozumiane jako szerokość i wysokość tekstu. Specyfikację układu strony możemy więc podać w postaci np. `\geometry{left=2cm, width=16cm, top=2cm, height=25cm}`, a pakiet automatycznie wyliczy wszystkie wielkości, przyjmując zasady:

- `paperwidth = left + width + right`
- `paperheight = top + height + bottom`

4.5 Tytuły rozdziałów i sekcji

Jeżeli potrzebujemy zmienić ustawienia dotyczące składania tytułów rozdziałów, podrozdziałów itd., to możemy w tym celu zastosować pakiet `titlesec`. Modyfikacja każdego z tytułów realizowana jest poprzez dwa polecenia: jedno ustawia szczegóły dotyczące formatowania tytułu, numeru, ewentualnej etykiety itp., a drugie wcięcia i odstępy przed i po tytule. Zmiany można ograniczyć do użycia tylko do jednego z nich.

Pierwszym z poleceń jest `\titleformat{}[]{}{}{}{}[]`. Znaczenie poszczególnych parametrów tego polecenia jest następujące:

- Pierwszy parametr wskazuje na polecenie, którego ustawienia chcemy modyfikować. Dopuszczalne parametry to `\part`, `\chapter`, `\section` itd.
- Drugi parametr (opcjonalny) wskazuje na organizację tytułu. Najciekawsze opcje to `hang` (wartość domyślna), `display` i `frame`. Przykłady zastosowania tych trzech opcji widoczne są na listingu 4.7.
- Trzeci parametr definiuje formatowanie, które będzie zastosowane do całego tytułu. Najczęściej umieszczamy w tym miejscu polecenia ustawiające wielkość czcionki i formatowanie tekstu.
- Czwarty parametr definiuje etykietę. Najczęściej jest to wartość odpowiedniego licznika, ale przy tytułach rozdziałów zazwyczaj dodajemy słowo „Rozdział” (LaTeX robi to domyślnie).
- Kolejny parametr definiuje wielkość separatora (liczba i jednostka). W przypadku opcji `hang` jest to wielkość poziomego odstępu między etykietą a właściwym tytułem. Dla pozostałych wspomnianych opcji jest to odstęp w pionie.
- Ostatnie dwa parametry mogą zawierać kod umieszczany przed i po tytule. Pierwszy z nich jest obowiązkowy, więc musimy zostawić co najmniej puste nawiasy klamrowe.

Drugie z poleceń to `\titlespacing{}{}{}{}[]`. Wszystkie parametry związane są z odległościami. Przy rozważanych opcjach organizacji tytułu (`hang`, `display`,

frame) pierwszy oznacza dodatkowy odstęp z lewej strony (przy ujemnej wartości tytuł wsunie się na margines), drugi oznacza pionowy odstęp przed tytułem, trzeci – pionowy odstęp po tytule, a ostatni (opcjonalny) – dodatkowy odstęp z prawej strony. Dodatkowo polecenie ma wersję z gwiazdką, która usuwa wcięcie z pierwszego akapitu po tytule. Więcej szczegółów na temat opcji i działania obu poleceń można znaleźć w dokumentacji pakietu [5].

```
\usepackage{titlesec}
\usepackage{pifont}
```

Listing 4.7. Przykłady ustawień dla tytułów rozdziałów

```
\titleformat{\chapter}[display]{\LARGE\bfseries}
{\textsc{\Huge Rozdział} \thechapter}{10mm}{\ding{103}~~}

\titleformat{\chapter}[frame]{\huge\bfseries}
{\textsc{Rozdział} \thechapter}{7mm}{\ding{103}~~}

\titleformat{\chapter}[hang]{\huge\bfseries}
{\Huge \thechapter}{10mm}{\ding{103}~~}

\titlespacing{\chapter}{0pt}{18mm}{18mm}
```

Rozdział 2

✳ Podstawy modelowania

Rozdział 2

✳ Podstawy modelowania

---

2 ✳ Podstawy modelowania

Na listingu 4.7 pokazano trzy przykłady specyfikacji wyglądu tytułu rozdziału. Wynik kompilacji dla tych ustawień pokazano na rysunku pod kodem źródłowym. Wszystkie ustawienia współdzielą polecenie **\titlespacing** z linii 10 (czego nie widać na rysunku, bo prezentuje on sam skład tytułu bez pierwszego akapitu). Z parametrów tego polecenia wynika, że tytuł jest dosunięty do lewej strony (do marginesu), a przed i po tytule mamy 18 mm odstępu.

Wszystkie polecenia **\titleformat** współdzielą pierwszy parametr (ustawiamy sposób składania tytułu rozdziału). Drugi parametr to trzy wspomniane wcześniej opcje. W przypadku opcji `display` (linie 1–2) etykieta z numerem jest składana w innej linii niż właściwy tytuł – piąty parametr decyduje o wielkości tego odstępu. Jako podstawowe formatowanie tytułu przyjęto czcionkę wielkości `LARGE` i pogrubioną. Etykieta składa się ze słowa „Rozdział” i numeru rozdziału. Samo słowo „Rozdział” składane jest czcionką wielkości `Huge` i z użyciem kapitalików. Ostatniego parametru użyto, aby dodać przed tytułem rozdziału symbol ✳.

Drugi przykład (linie 4–5) używa opcji `frame` – tytuł rozdziału umieszczony jest w ramce, a etykieta znajduje się na górnej krawędzi ramki. Parametr piąty decyduje tutaj o odległości od tekstu tytułu do ramki. W tym przypadku całość korzysta z czcionki pogrubionej o wielkości `huge`. Dodatkowo słowo „Rozdział” składane jest z użyciem kapitalików. W ostatnim przykładzie zastosowano opcję `hang`, która powoduje złożenie etykiety (tutaj ograniczonej do samego numeru) i tytułu w tej samej linii. Piąty parametr decyduje tutaj o odstępie poziomym między etykietą a tytułem.

Analogicznie używamy przedstawionych poleceń do dopasowania wyglądu części generowanych poleceniami **\section**, **\subsection** itd. W tych przypadkach najczęściej stosowana jest opcja `hang`, choć z opcją `display` również można uzyskać ciekawy efekt, jak pokazano na listingu 4.8.

Listing 4.8. Przykład ustawień dla tytułów sekcji (podrozdziałów)

```
1 \titleformat{\section}[display]
2 {\filcenter\Large\bfseries}{\filcenter\thesection}{0mm}{}
3 \titlespacing{\section}{0mm}
4 {8mm plus 0mm minus 1mm}{4mm plus 0mm minus 1mm}
```

2.1
Proces modelowania i weryfikacji systemów

Pakiet `titlesec` zawiera także kilka dodatkowych poleceń, które pozwalają dopasować na przykład umieszczenie etykiety dla opcji `display` (domyślnie dosunięta do lewej) i `frame` (domyślnie wycentrowana). Polecenie **\filright**

spowoduje dosunięcie etykiety do lewej strony (tak, do lewej, wstawiamy jakby sprężynę po prawej stronie), **\filleft** spowoduje dosunięcie do prawej strony, a **\filcenter** – wyśrodkowanie. W przykładzie z listingu 4.8 zarówno etykieta (tutaj tylko numer sekcji), jak i jej tytuł zostały wyśrodkowane (linia 2). Dodatkowo pokazano, jak można definiować elastyczne odstępy przed i po (linia 4). Pozwala to LaTeX-owi na dopasowanie, w pewnym zakresie, pionowych odstępów do potrzeb składu danej strony.

4.6 Spis treści

Spis treści jest nieodłącznym elementem większych dokumentów, choć możemy go dołączyć nawet, gdy używamy klasy `article`. W najprostszej wersji spis treści uzyskamy, stosując bezparametrowe polecenie **\tableofcontents** w miejscu, gdzie ma on zostać umieszczony. Jeżeli modyfikujemy strukturę dokumentu, to aby zmiany znalazły swoje odzwierciedlenie w spisie treści, potrzebna jest dwukrotna kompilacja. Spis treści generowany jest na podstawie argumentów poleceń takich jak **\part**, **\chapter**, **\section** itd. Domyślnie w spisie treści lądują tytuły do drugiego poziomu włącznie, przy czym numeracja poziomów jest następująca: `part` −1, `chapter` 0, `section` 1, `subsection` 2 itd. To domyślne ustawienie możemy zmienić za pomocą polecenia **\setcounter**`{tocdepth}{x}`, które umieszczamy w preambule. Jeżeli zamiast `x` wstawimy 1, to w przypadku klasy `book`, która nie ma wydzielonych części (`part`), w spisie treści znajdą się tylko tytuły rozdziałów i sekcji.

Polecenia **\part**, **\chapter**, **\section** itd. mogą poza obowiązkowym parametrem będącym tytułem danej części przyjmować również parametr opcjonalny. Stosujemy go, aby w spisie treści umieścić inny niż oryginalny tytuł. Korzystamy z tego głównie wtedy, gdy tytuł jest długi i źle wygląda w spisie treści (nie mieści się w jednej linii). Definiujemy wówczas jego skróconą wersję. Przykład pokazano na listingu 4.9. Tekst w nawiasie kwadratowym zostanie umieszczony w spisie treści, a także będzie używany w nagłówku strony, jeśli wybierzemy odpowiedni styl (zob. podrozdział 14.2).

Listing 4.9. Przykład definiowania skróconego tytułu do zamieszczenia w spisie treści

```
\section[Promieniotwórczość w środowisku naturalnym\dots]
{Promieniotwórczość w środowisku naturalnym wynikająca
z obecności śladowych ilości uranu}
```

Rozważane polecenia mają również wersję z gwiazdką, która odpowiada za generowanie nienumerowanych tytułów. Jeżeli więc w naszej książce pojawi się polecenie **\chapter***`{Od autora}`, to ten rozdział nie zostanie

uwzględniony w spisie treści. Możemy go dodać samodzielnie za pomocą polecenia `\addcontentsline{}{}{}`. Pierwszy parametr wskazuje na listę, do której dodajemy wpis (`toc` dla spisu treści), drugi wskazuje na typ tytułu (np. `chapter`, `section`, `figure`) i decyduje o tym, jak tytuł będzie formatowany w spisie treści, a ostatni zawiera tytuł, który ma być dodany. Wracając do naszego przykładu: po poleceniu `\chapter*{Od autora}` i `\label{...}` powinniśmy dodać linię `\addcontentsline{toc}{chapter}{Od autora}`.

Dostępne jest również polecenie `\addtocontents{}{}`. Pierwszy parametr jest taki jak poprzednio, a drugi to zawartość, która bez żadnego formatowania jest wstawiana do spisu treści. Przykłady użycia tego polecenia pokazano na listingu 4.10. Pierwsze z poleceń dodaje do spisu treści (w miejscu, gdzie je umieścimy) pionowy odstęp. Drugie z nich wymusza łamanie strony, a ostatnie ustawia styl strony na pusty (puste nagłówki i stopki – zob. podrozdział 14.2).

Listing 4.10. Przykład poleceń modyfikujących spis treści

```
1 \addtocontents{toc}{\vspace{2mm}}
2 \addtocontents{toc}{\protect\newpage}
3 \addtocontents{toc}{\protect\thispagestyle{empty}}
```

Wiele poleceń LaTeX-a określa się jako polecenia kruche. Należą do nich m.in. wszystkie polecenia przytrzymujące parametry opcjonalne. Poleceń kruchych nie można bezpośrednio używać wewnątrz parametrów wielu innych poleceń – wygeneruje to błąd na etapie kompilacji. Błędy kompilacji możemy wyeliminować poprzedzając kruche polecenia poleceniem `\protect`.

```
\usepackage{titlesec}
\usepackage{titletoc}
```

Jeżeli potrzebujemy bardziej ingerować w wygląd spisu treści, to z pomocą przyjdzie nam pakiet `titletoc` [5], który, co warto podkreślić, dobrze współpracuje z pakietem `titlesec`. Jeżeli zastosowaliśmy polecenia `\titleformat` i `\titlespacing` (zob. podrozdział 4.5) do ustawienia parametrów tytułów rozdziałów (przy założeniu, że używamy klasy `book` lub `report`), to ustawienia tytułu dla spisu treści będą spójne z ustawieniami tytułów innych rozdziałów.

Ze względu na dobrą współpracę między pakietami `titletoc` i `titlesec` dużo łatwiej jest dopasować wygląd spisu treści i innych spisów za pomocą pakietu `titletoc` niż pakietu `tocloft`.

Listing 4.11. Przykład poleceń modyfikujących spis treści – pakiet `titletoc`

```
\titlecontents{chapter}[0em]{\bfseries\addvspace{1em}}
  {\makebox[1.5em][l]{\contentslabel{0em}}}{}{\hfill\contentspage}
  [\titlerule\addvspace{0.25em}]
\titlecontents{section}[4em]{}{\contentslabel{2.5em}}{}
  {\titlerule*[6pt]{\tiny\color{gray!50}\ding{103}}\contentspage}
\titlecontents{subsection}[7em]{}{\contentslabel{3em}}{}
  {\titlerule*[6pt]{\tiny\color{gray!50}\ding{103}}\contentspage}
```

Spis treści

Od autora **2**

1 Podstawy tworzenia prezentacji **3**

1.1 Struktura slajdów 3
1.2 Używanie bloków 4
1.2.1 Bloki wewnątrz środowiska minipage 4

2 Nakładki **5**

2.1 Sceny 5
2.1.1 Polecenie *pause* 6
2.1.2 Specyfikacje odsłaniania scen 6
2.2 Sceny złożone z grafiki 7
2.2.1 Efekt symulacji 7

1,5em 2,5em 3em

Pakiet `titletoc` zawiera kilka poleceń do zarządzania wyglądem poszczególnych wpisów w spisie treści. W przykładzie z listingu 4.11 (kod do umieszczenia w preambule) użyto trzykrotnie polecenia **`\titlecontents`**`{}[]{}{}{}{}[]` do ustawienia kolejno wyglądu wpisów dotyczących rozdziałów, sekcji i podsekcji. Każdy wpis w spisie treści obejmuje kilka elementów: wcięcie z lewej strony (domyślnie im niższy poziom, tym większe), numer wpisywanego rozdziału, sekcji itp. (jeśli jest numerowany), właściwy tytuł, wypełnienie przestrzeni po tytule i numer strony. Znaczenie kolejnych parametrów polecenia **`\titlecontents`** jest następujące: 1) nazwa poziomu organizacyjnego, którego wpis modyfikujemy, np. `chapter`, `section` itd., 2) wielkość wcięcia po lewej stronie (nie można pominąć tego parametru), 3) kod umieszczany przed całym

wpisem (może zawierać polecenia definiujące odstępy w pionie, ale także polecenia dotyczące formatowania całego wpisu), 4) treść umieszczana przed tytułem numerowanym (zazwyczaj odpowiednio sformatowany numer), 5) treść umieszczana przed tytułem nienumerowanym, 6) sposób wypełnienia wolnej przestrzeni, 7) kod umieszczany po całym wpisie (może zawierać polecenia definiujące odstępy w pionie).

Do pomocy przy skonstruowaniu wszystkich parametrów polecenia **\titlecontents** mamy kilka dodatkowych poleceń dostępnych w pakiecie `titletoc`. Polecenie **\contentslabel**`{}` rezerwuje obszar o podanej szerokości przed wpisem i umieszcza w nim numer właściwego rozdziału, sekcji itd., a polecenie **\contentspage** rezerwuje przestrzeń na końcu wpisu (o wspólnej szerokości dla wszystkich wpisów) i umieszcza w niej numer strony. Dodatkowo w pakiecie zalecane jest stosowanie polecenia **\addvspace**`{}` zamiast **\vspace** do definiowania pionowych odstępów (parametrem jest wielkość odstępu) i używanie `\\*` zamiast `\\` do łamania linii.

W wizualizacji efektów kodu z listingu 4.11 dodane zostały w lewym dolnym rogu dodatkowe linie i oznaczone wielkości poszczególnych wcięć, aby ułatwić zrozumienie użytych parametrów. Linie 1–3 zawierają polecenie ustalające wygląd wpisów dla rozdziałów (wskazuje na to pierwszy parametr). Wpisy składane są bez wcięcia z lewej strony. W trzecim parametrze dodajemy polecenia **\bfseries** i **\addvspace** przed każdym wpisem dotyczącym rozdziału. Skutkuje to złożeniem całego takiego wpisu pogrubioną czcionką i dodaniem pionowego odstępu przed wpisem. Podobnie ostatni parametr skutkuje wyrysowaniem poziomej linii (polecenie **\titlerule**) i dodaniem małego odstępu po całym wpisie. Ponieważ mamy wpisy rozdziałów numerowane i nienumerowane i nie chcemy numerów wysuniętych w lewo, numer rozdziału został umieszczony w pudełku **\makebox** o szerokości 1,5 em. To spowoduje przesunięcie w prawo za pudełko tytułów numerowanych rozdziałów. Nie wstawiamy niczego (pusty parametr) przed nienumerowanym tytułem. Polecenie **\hfill** w szóstym parametrze oznacza pustą przestrzeń między tytułem i numerem strony.

Drugie polecenie (linie 4–5) ustala formatowanie dla tytułów sekcji. Wcięcie z lewej strony (4 em) jest wcięciem od marginesu do tytułu. W tym wcięciu mieści się też pudełko 2,5 em na numer sekcji (parametr polecenia **\contentslabel**). W tym przypadku nie umieszczamy żadnego kodu przed wpisem i po wpisie. Nie zmieniamy też formatowania wpisu. Nowym elementem jest tutaj użycie polecenia **\titlerule***`[]{}` z pakietu `titlesec`, które definiuje znak wypełniający przestrzeń między tytułem i numerem strony. Parametr opcjonalny decyduje o tym, jak gęsto umieszczone są znaki, a obowiązkowy określa używany znak. Ustawienia dla podsekcji są analogiczne, zmienia się tylko wielkość wcięcia i szerokość pudełka na numer podsekcji.

4.7 Pliki LaTeX-a

Na zakończenie tego rozdziału omówimy pliki, z jakimi możesz się spotkać, pracując z LaTeX-em. Jeżeli używasz systemu lokalnie, to sporo plików pojawia się automatycznie w katalogu projektu. Jeżeli pracujesz w środowisku Overleaf, to dobrze wiedzieć o istnieniu tych plików, gdybyś chciał pobrać któryś z nich.

Z podrozdziału 4.3 wiesz już, że w Twoim katalogu może być więcej niż jeden plik `tex`. Jeżeli pracujesz z szablonem pobranym z wydawnictwa, to od razu może pojawić się w katalogu kilka innych plików. Można je zidentyfikować na podstawie rozszerzeń:

- `cls` – plik z klasą dokumentu, np. `IEEEtran.cls`;
- `bst` – plik z definicją stylu dla BibTeX-a, np. `IEEEtran.bst`;
- `sty` – plik zawierający pakiet instrukcji LaTeX-a, możemy go dołączyć do dokumentu z użyciem polecenia **`\usepackage`**;
- `log` – komunikaty generowane podczas ostatniej kompilacji projektu;
- `aux` – plik pomocniczy służący między innymi do zbierania informacji o odwołaniach w dokumencie – jeśli podzielimy projekt na wiele plików, to dla każdego z nich oddzielnie generowany jest plik `aux`;
- `bib` – plik z bazą rekordów BibTeX-a (zob. rozdział 9);
- `bbl` – plik zawierający złożoną bibliografię, jeżeli korzystamy z BibTeX-a lub BibLaTeX-a;
- `blg` – komunikaty generowane przez BibTeX-a;
- `bcf` – tymczasowy plik BibLaTeX-a do pracy z biberem;
- `idx` – plik z danymi do wygenerowania skorowidza;
- `ilg` – komunikaty wyświetlane przy generowaniu skorowidza (`makeindex`);
- `ind` – plik zawierający wygenerowany skorowidz;
- `lof` – plik z danymi do wygenerowania spisu rysunków;
- `lot` – plik z danymi do wygenerowania spisu tabel;
- `toc` – plik z danymi do wygenerowania spisu treści.

Czasami zdarza się, że kompilator wskazuje na błędy w plikach tymczasowych. Są one pochodną błędów w naszych plikach źródłowych. Zdarza się, że po poprawieniu błędów w naszych plikach kompilator ciągle je „widzi" i wskazuje te same pliki tymczasowe jako ich źródło. W takich przypadkach warto usunąć pliki tymczasowe i od nowa skompilować projekt.

Rozdział 5

Tabele

Większość publikacji lub książek naukowych zawiera elementy o układzie tabelarycznym. Nie ma tutaj znaczenia, czy jest to tabela, jakich wiele spotkałeś już w tej książce, macierz liczbowa, czy grupa rysunków, do której organizacji wykorzystano układ tabelaryczny. Z perspektywy LaTeX-a zasady konstruowania tego typu elementów są wyjątkowo spójne. W niniejszym rozdziale omówimy te konstrukcje, a także pokażemy rozwiązania dla typowych problemów, z którymi spotykamy się, składając nieco bardziej skomplikowane tabele.

5.1 Składnia struktur tabelarycznych

Podstawowa zasada konstruowania struktur tabelarycznych jest wyjątkowo prosta. Każdy wiersz kończymy poleceniem `\\`, komórki w wierszu oddzielamy od siebie znakiem `&` i dbamy, aby każdy wiersz miał tyle samo komórek. Z perspektywy kompilatora zupełnie nie ma znaczenia, czy jeden wiersz jest umieszczony w jednej linii kodu źródłowego, czy jedna komórka zajmuje kilka linii. Ważne jest, abyśmy przestrzegali powyższej zasady.

Podstawowym środowiskiem do składania tabel jest **`tabular`**. Środowisko to przyjmuje jeden parametr obowiązkowy, którym jest definicja układu tabeli. Układ ten definiujemy za pomocą liter, które jednocześnie określają domyślny sposób wyrównania zawartości w odpowiadającej im kolumnie. Podstawowe trzy litery to: `l` – wyrównanie do lewej strony, `c` – wyśrodkowanie i `r` – wyrównanie do prawej strony. Argument w postaci `rcl` oznacza, że tabela będzie zawierała trzy kolumny, przy czym zawartość pierwszej będzie wyrównana do prawej strony, drugiej wyśrodkowana, a trzeciej wyrównana do lewej strony. Parametr w postaci `ccccc` oznacza, że tabela będzie złożona z pięciu kolumn i zawartość wszystkich będzie wyśrodkowana. Równoważnie można zastosować tutaj

zapis `*{5}{c}`. Ta druga wersja jest bardzo przydatna, gdy definiujemy tabele o dużej liczbie takich samych kolumn. Zapis ten może być bardziej złożony, np. `*{5}{cl}` oznacza, że mamy 10 kolumn – pięć par, w których zawartość pierwszej kolumny jest wyśrodkowana, a drugiej wyrównana do lewej strony.

Listing 5.1. Przykład użycia środowiska **tabular**

```
\begin{tabular}{rcl}
1903 & Maria Skłodowska-Curie & fizyka     \\
1905 & Henryk Sienkiewicz     & literatura \\
1911 & Maria Skłodowska-Curie & chemia     \\
1924 & Władysław Reymont      & literatura \\
\end{tabular}
```

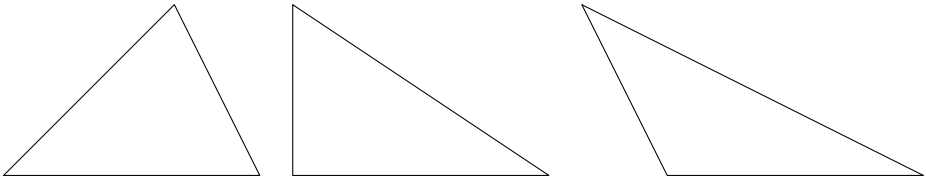
1903	Maria Skłodowska-Curie	fizyka
1905	Henryk Sienkiewicz	literatura
1911	Maria Skłodowska-Curie	chemia
1924	Władysław Reymont	literatura

Przykład użycia środowiska **tabular** przedstawiono na listingu 5.1. Od razu warto zauważyć, że nie musimy organizować kodu tak, aby znaki & tworzyły równe kolumny. W tym przypadku użyto polecenia środowiska TeXstudio, które automatycznie przeorganizowało kod źródłowy. Jest to przydatna funkcjonalność, zwłaszcza gdy przy większej tabeli zgubimy jakiś znak & i w pliku `pdf` widzimy zepsutą strukturę tabeli. Przeformatowanie kodu pozwala łatwiej znaleźć nasz błąd. Pokazana tabela nie zawiera wiersza nagłówkowego ani żadnych obramowań. Jedyne, co widzimy, to zorganizowany układ komórek.

Komórki w tabeli nie muszą zawierać wyłącznie tekstu. Na listingu 5.2 pokazano przykład wykorzystania środowiska **tabular** do organizacji grafiki. Rozważana tabela składa się z trzech kolumn o wyśrodkowanej zawartości. W pierwszym wierszu trzykrotnie użyliśmy polecenia **\includegraphics**, aby wstawić grafikę. W drugim wierszu mamy już tekstowe opisy poszczególnych rysunków.

Listing 5.2. Przykład użycia środowiska **tabular** do organizacji grafiki

```
\begin{tabular}{ccc}
\includegraphics{ostrokatny.pdf} &
\includegraphics{prostokatny.pdf} &
\includegraphics{rozwartokatny.pdf} \\
trójkąt ostrokątny &
trójkąt prostokątny &
trójkąt rozwartokątny \\
\end{tabular}
```



trójkąt ostrokątny trójkąt prostokątny trójkąt rozwartokątny

Jeżeli definiujemy kolumnę z użyciem liter `l`, `c` i `r`, to nie możemy ograniczyć szerokości kolumny. Jest ona określona na podstawie zawartości komórek. O ile nie zastosujemy jakichś specjalnych rozwiązań (zob. podrozdział 5.6), to zawartość komórki jest składana w pojedynczym wierszu. Czwartą podstawową opcją definiowania kolumn jest `p{}`. W takim przypadku w nawiasie klamrowym określamy szerokość kolumny, a w przypadku dłuższej zawartości komórek ich treść będzie składana w wielu liniach. Przykład użycia tej opcji pokazano na listingu 5.3. Uzyskany efekt jest bardzo podobny do tego z użyciem środowiska **`minipage`** (zob. listing 3.17).

Listing 5.3. Przykład definiowania kolumn z użyciem opcji `p`

```
\begin{tabular}{p{3.9cm}p{8cm}}
Maria Skłodowska-Curie 1867--1934&
Dwukrotnie otrzymała Nagrodę Nobla: w 1903 r. z~fizyki za badania nad
zjawiskiem promieniotwórczości, a w 1911 r. z chemii za odkrycie
polonu i~radu, wydzielenie czystego radu i~badanie właściwości
chemicznych pierwiastków promieniotwórczych. Jest jedyną kobietą,
która otrzymała dwie nagrody w~dwóch różnych dyscyplinach.\\
\end{tabular}
```

Maria Skłodowska-Curie 1867–1934	Dwukrotnie otrzymała Nagrodę Nobla: w 1903 r. z fizyki za badania nad zjawiskiem promieniotwórczości, a w 1911 r. z chemii za odkrycie polonu i radu, wydzielenie czystego radu i badanie właściwości chemicznych pierwiastków promieniotwórczych. Jest jedyną kobietą, która otrzymała dwie nagrody w dwóch różnych dyscyplinach.

`\usepackage{array}`

Warto dodać, że w przypadku kolumny typu `p{}` możemy używać wewnątrz komórek polecenia **`\newline`**, aby samodzielnie złamać linię. Jeżeli dołączymy pakiet `array`, to do dyspozycji mamy jeszcze dwa typy kolumn `m{}` i `b{}`, które

funkcjonują podobnie jak `p{}` (w ich przypadku też można używać polecenia **`\newline`**). Różnica między tymi trzema typami kolumn dotyczy wyrównania zawartości komórek w pionie. W przypadku `p{}` linia bazowa umieszczona jest w pierwszym wierszu, dla `b{}` w ostatnim, a w przypadku `m{}` zawartość jest wyśrodkowana w pionie. Zostało to zobrazowane na rys. 5.1. Pokazana tabela zawiera nazwy kolorów zdefiniowane w pakiecie `xcolor`.

`p{}`	`m{}`	`b{}`
red, green, blue, yellow, black, white	cyan, magenta, brown, olive, lime, teal	gray, lightgray, darkgray, orange, pink, purple, violet

Rys. 5.1. Wyrównanie zawartości komórki w pionie w zależności od typu kolumny

Zazwyczaj nie mieszamy różnych typów kolumn spomiędzy trzech prezentowanych na rys. 5.1. Jeżeli wszystkie kolumny będą typu `p{}`, to będziemy mieć efekt wyrównania zawartości do góry. Jeżeli wszystkie będą typu `b{}`, to do dołu. W trzecim przypadku będziemy mieć wyśrodkowanie. Pozornie dziwny efekt dostajemy tylko wtedy, gdy wymieszamy ze sobą te typy. Długa linia przerywana na rys. 5.1 jest linią bazową, względem której mamy wyrównanie w pionie. Leży na niej pierwszy wiersz w kolumnie `p{}` i ostatni w kolumnie `b{}`.

5.2 Obramowania

Podstawowe obramowania komórek tabeli uzyskujemy przez kombinację znaków `|` w parametrze definiującym strukturę tabeli i poleceń **`\hline`** w treści tabeli. Dla przykładu: parametr `r||c|l|` środowiska **`tabular`** oznacza, że definiujemy tabelę złożoną z trzech kolumn (o odpowiednim wyrównaniu zawartości) oraz wstawiamy podwójną pionową linię między kolumną 1 i 2, a także pojedynczą pionową linię między kolumną 2 i 3 oraz po kolumnie 3. Poziome linie na szerokość całej tabeli wstawiamy za pomocą polecenia **`\hline`**. Dwukrotne użycie tego polecenia wygeneruje podwójną poziomą linię. Przykład wykorzystania tych możliwości pokazano na listingu 5.4.

Listing 5.4. Przykład definiowania ramek w środowisku **`tabular`**

```
1 \begin{tabular}{r||c|l|}
2 \hline
```

```
3 \textbf{Rok} & \textbf{Osoba} & \textbf{Dyscyplina}\\
4 \hline\hline
5 1903 & Maria Skłodowska-Curie & fizyka     \\
6 1905 & Henryk Sienkiewicz     & literatura \\
7 1911 & Maria Skłodowska-Curie & chemia     \\
8 1924 & Władysław Reymont      & literatura \\\hline
9 \end{tabular}
```

Rok	Osoba	Dyscyplina
1903	Maria Skłodowska-Curie	fizyka
1905	Henryk Sienkiewicz	literatura
1911	Maria Skłodowska-Curie	chemia
1924	Władysław Reymont	literatura

Przeanalizujmy wybrane linie listingu 5.4. Polecenie **\hline** w linii 2 pozwala uzyskać poziomą linię nad pierwszym wierszem. Podwójne polecenie **\hline** w linii 4 generuje podwójną linię po wierszu nagłówkowym. Widzimy, że w miejscu podwójnej poziomej linii linie pionowe są przerwane. Mamy efekt oderwania wiersza nagłówkowego od reszty tabeli. Taki skład jest charakterystyczny dla LaTeX-a i dość często spotykany. Polecenie **\hline** po ostatnim wierszu (linia 8) generuje linię poziomą poniżej tego wiersza. Po wierszach w liniach 5–7 nie ma polecenia **\hline**, stąd brak linii poziomej między odpowiednimi wierszami tabeli. Gdybyśmy po ostatnim wierszu nie wstawiali polecenia **\hline**, to możemy pominąć również \\.

```
\usepackage{booktabs}
```

Warto w tym miejscu dodać, że niektórzy wydawcy odradzają stosowanie pionowych linii w tabelach i ograniczenie linii poziomych przede wszystkim do wyróżnienia wiersza nagłówkowego. W takim przypadku tabela zawiera poziomą linię nad i pod wierszem nagłówkowym oraz po ostatnim wierszu. Takie rozwiązanie jest preferowane w pakiecie `booktabs` [15]. Pakiet ten zawiera trzy polecenia do generowania tych poziomych linii (odpowiednio **\toprule**, **\midrule** i **\bottomrule**). Można też niezależnie ustawić grubość każdej z tych linii. Jeżeli robimy to okazjonalnie, to można użyć parametru opcjonalnego, np. **\midrule**`[3pt]`. Jeśli chcemy ustawić wspólne wartości dla całego dokumentu, to w preambule, po dodaniu pakietu `booktabs`, należy redefiniować polecenia **\heavyrulewidth** i **\lightrulewidth**. Pierwsze definiuje domyślną grubość linii dla **\toprule** i **\bottomrule**, a drugie dla **\midrule**. Przykład redefinicji tych poleceń i składu tabeli ze zmienionymi grubościami linii pokazano na listingu 5.5.

Listing 5.5. Przykład zastosowania pakietu `booktabs`

```
\setlength\heavyrulewidth{0.4ex}     % w preambule
\setlength\lightrulewidth{0.2ex}

\begin{tabular}{rcl}
\toprule
\textbf{Rok} & \textbf{Osoba} & \textbf{Dyscyplina}\\
\midrule
1903 & Maria Skłodowska-Curie & fizyka     \\
1905 & Henryk Sienkiewicz     & literatura \\
1911 & Maria Skłodowska-Curie & chemia     \\
1924 & Władysław Reymont      & literatura \\
\bottomrule
\end{tabular}
```

Rok	Osoba	Dyscyplina
1903	Maria Skłodowska-Curie	fizyka
1905	Henryk Sienkiewicz	literatura
1911	Maria Skłodowska-Curie	chemia
1924	Władysław Reymont	literatura

```
\usepackage{colortbl}
\usepackage{array}
```

Elegancki wygląd tabel możemy uzyskać, stosując różne kolory tła dla wierszy lub kolumn zamiast obramowań. W tym celu możemy posłużyć się pakietem `colortbl` [7]. Jeżeli chodzi o definiowanie tła dla wierszy, to w pakiecie zawarto w tym celu dwa polecenia. Polecenie `\rowcolor{}`, którego parametrem jest kolor tła, umieszczamy przed pierwszą komórką wiersza. Polecenie to zmieni kolor danego wiersza na wskazany. Polecenie `\rowcolors{}{}{}` umieszczamy przed środowiskiem **`tabular`**, a służy ono do określenia reguł naprzemiennego kolorowania tła wierszy w tabeli. Pierwszy parametr jest numerem wiersza, który ma być uznany za pierwszy wiersz nieparzysty (jeżeli chcemy pominąć pojedynczy wiersz nagłówkowy w schemacie kolorowania, to wybieramy wartość 2). Kolejne dwa parametry to kolor tła dla wierszy nieparzystych i parzystych.

Przykład zastosowania pakietu `colortbl` pokazano na listingu 5.6. Polecenie `\rowcolors` z pierwszej linii ustala naprzemienne stosowanie koloru `gray!20` i białego, począwszy od drugiego wiersza. Polecenie `\rowcolor` z trzeciej linii ustawia nieco ciemniejsze szare tło dla wiersza nagłówkowego.

Listing 5.6. Przykład naprzemiennego kolorowania tła wierszy – pakiet `colortbl`

```
1  \rowcolors{2}{gray!20}{white}
2  \begin{tabular}{>{\bfseries}c*{8}{c}}
3  \rowcolor{gray!40}
4  df~\textbackslash~p & \textbf{0,99} & \textbf{0,95} &
5  \textbf{0,90} & \textbf{0,80} & \textbf{0,50} & \textbf{0,30} &
6  \textbf{0,20} & \textbf{0,10}\\
7  1 & 0,000 & 0,004 & 0,016 & 0,064 & 0,455 & 1,074 & 1,642 & 2,706 \\
8  2 & 0,020 & 0,103 & 0,211 & 0,446 & 1,386 & 2,408 & 3,219 & 4,605 \\
9  3 & 0,115 & 0,352 & 0,584 & 1,005 & 2,366 & 3,665 & 4,642 & 6,251 \\
10 4 & 0,297 & 0,711 & 1,064 & 1,649 & 3,357 & 4,878 & 5,989 & 7,779 \\
11 5 & 0,554 & 1,145 & 1,610 & 2,343 & 4,351 & 6,064 & 7,289 & 9,236 \\
12 \end{tabular}
```

df \ p	0,99	0,95	0,90	0,80	0,50	0,30	0,20	0,10
1	0,000	0,004	0,016	0,064	0,455	1,074	1,642	2,706
2	0,020	0,103	0,211	0,446	1,386	2,408	3,219	4,605
3	0,115	0,352	0,584	1,005	2,366	3,665	4,642	6,251
4	0,297	0,711	1,064	1,649	3,357	4,878	5,989	7,779
5	0,554	1,145	1,610	2,343	4,351	6,064	7,289	9,236

Warto jeszcze wrócić do parametru środowiska **tabular**, tj. do konstrukcji `>{\bfseries}c*{8}{c}`. Definicja pierwszej kolumny została wzbogacona o polecenie `>{}` z pakietu `array` [39]. Polecenie to na początku każdej komórki w pierwszej kolumnie wstawia tekst przekazany mu jako parametr. W rozważanym przykładzie daje nam to efekt pogrubienia czcionki w pierwszej kolumnie. Pakiet `array` zawiera również polecenie `<{}`, które swój parametr wstawia na końcu każdej komórki w kolumnie, na przykład określenie kolumny jako `c<{~mm}` oznaczałoby, że po zawartości każdej komórki w tej kolumnie dodajemy twardą spację i jednostkę mm.

Pakiet `colortbl` zawiera również polecenie **\columncolor**`{}`, którego jedynym obowiązkowym parametrem jest kolor tła kolumny. Polecenie to powinno być używane jako parametr dla polecenia `>{}`. Przykład zastosowania pakietu `colortbl` do naprzemiennego kolorowania kolumn pokazano na listingu 5.7. Zwróćmy uwagę na parametr środowiska **tabular**. W definicji pierwszej kolumny (linia 1) parametrem polecenia `>{}` są aż dwa polecenia – pierwsze jak poprzednio odpowiada za pogrubioną czcionkę, drugie za ustawienie szarego koloru `gray!40` dla tej kolumny. Dalej mamy pięć razy powtórzoną definicję pary kolumn `c>{\columncolor{gray!20}}c`. Pierwsza z nich będzie miała domyślne białe tło, druga ma tło zmienione na jasnoszare. Gdybyśmy chcieli dodatkowo ustawić tło pierwszego wiersza na taki sam kolor jak tło pierwszej kolumny, to dodajemy polecenie **\rowcolor**`{gray!40}` na początku pierwszego wiersza.

Listing 5.7. Przykład naprzemiennego kolorowania tła kolumn – pakiet `colortbl`

```
\begin{tabular}{>{\bfseries\columncolor{gray!40}}c
*{4}{c>{\columncolor{gray!20}}c}}
df~\textbackslash~p & \textbf{0,99} & \textbf{0,95} &
\textbf{0,90} & \textbf{0,80} & \textbf{0,50} & \textbf{0,30} &
\textbf{0,20} & \textbf{0,10}\\
1 & 0,000 & 0,004 & 0,016 & 0,064 & 0,455 & 1,074 & 1,642 & 2,706 \\
2 & 0,020 & 0,103 & 0,211 & 0,446 & 1,386 & 2,408 & 3,219 & 4,605 \\
3 & 0,115 & 0,352 & 0,584 & 1,005 & 2,366 & 3,665 & 4,642 & 6,251 \\
4 & 0,297 & 0,711 & 1,064 & 1,649 & 3,357 & 4,878 & 5,989 & 7,779 \\
5 & 0,554 & 1,145 & 1,610 & 2,343 & 4,351 & 6,064 & 7,289 & 9,236 \\
\end{tabular}
```

df \ p	0,99	0,95	0,90	0,80	0,50	0,30	0,20	0,10
1	0,000	0,004	0,016	0,064	0,455	1,074	1,642	2,706
2	0,020	0,103	0,211	0,446	1,386	2,408	3,219	4,605
3	0,115	0,352	0,584	1,005	2,366	3,665	4,642	6,251
4	0,297	0,711	1,064	1,649	3,357	4,878	5,989	7,779
5	0,554	1,145	1,610	2,343	4,351	6,064	7,289	9,236

Jeżeli w naszym dokumencie planujemy często używać kolumn z określonym kolorem tła, to zamiast stosować polecenie **`\columncolor`** jako parametr polecenia `>{}`, prościej jest zdefiniować własny typ kolumny. Służy do tego polecenie **`\newcolumntype`**`{}{}`. Pierwszy parametr to jednoliterowe oznaczenie naszej kolumny, a drugi to jej definicja. W naszym przypadku (listing 5.7) może to być **`\newcolumntype`**`{g}{>{`**`\columncolor`**`{gray!20}}c}`. Wtedy kolumny od 2 do 9 mogą być zdefiniowane jako `*{4}{cg}`.

Zastosowanie kolumny typu `p{}`, `m{}` lub `b{}` skutkuje wyrównaniem tekstu do lewej i prawej strony. W efekcie często tekst jest nadmiernie rozciągnięty w niektórych liniach, co nie wygląda dobrze. Problem ten jest dobrze widoczny na rys. 5.1 w dwóch pierwszych kolumnach. Można wykorzystać przedstawioną konstrukcję `>{}` i przełączniki **`\raggedright`**, **`\centering`** i **`\raggedleft`**, aby zdefiniować własne wersje kolumn z odpowiednim wyrównaniem. Przykład pokazano na listingu 5.8.

Listing 5.8. Definiowanie własnych typów kolumn

```
\newcolumntype{L}[1]{>{\raggedright\let\newline\\\arraybackslash
  \hspace{0pt}}p{#1}}
\newcolumntype{C}[1]{>{\centering\let\newline\\\arraybackslash
  \hspace{0pt}}p{#1}}
\newcolumntype{R}[1]{>{\raggedleft\let\newline\\\arraybackslash
  \hspace{0pt}}p{#1}}

```

```
\begin{tabular}{|L{3.5cm}|C{3.5cm}|R{3.5cm}|}
\hline
\textbf{Nazwa} & \textbf{Nazwa} & \textbf{Nazwa} \\\hline
red, green, blue, yellow, black, white &
cyan, magenta, brown, olive, lime, teal &
gray, lightgray, darkgray, orange, pink, purple, violet \\\hline
\end{tabular}
```

Nazwa	**Nazwa**	**Nazwa**
red, green, blue, yellow, black, white	cyan, magenta, brown, olive, lime, teal	gray, lightgray, darkgray, orange, pink, purple, violet

Zdefiniowane typy kolumn `L`, `C` i `R` bazują na typie `p` i podobnie jak dla typu `p` musimy dostarczyć informację o szerokości kolumny (linia 8). Wszystkie parametry polecenia `>{}` poza pierwszym poleceniem związanym ze sposobem wyrównania tekstu zawierają sekwencję trzech innych poleceń. Po pierwsze **\let\newline**\\ pozwala na poprawne użycie polecenia **\newline** do samodzielnego łamania linii wewnątrz komórki. Ponieważ przełączniki **\raggedright**, **\centering** i **\raggedleft** redefiniują znaczenie polecenia \\, użyte jest polecenie **\arraybackslash** z pakietu `array`, aby przywrócić poleceniu \\ funkcję kończenia wiersza tabeli. Ostatnie polecenie jest sztuczką, która pozwala na dzielenie pierwszego słowa w komórce, co domyślnie nie jest stosowane przez LaTeX-a.

```
\usepackage{tcolorbox}
```

Ciekawe efekty formatowania dla tabel możemy uzyskać, używając pakietu `tcolorbox` (zob. podrozdział 3.5). Środowisko **tabular** jest tutaj zastąpione przez środowisko **tcolorbox**. Specyfikacja dotycząca struktury tabeli jest przekazywana jako wartość parametru `tabulars`. Ponadto możemy korzystać z opcji środowiska **tcolorbox**, aby uzyskać oczekiwany efekt wizualny (zob. tabela 3.3). Przykład tabeli złożonej w środowisku **tcolorbox** pokazano na listingu 5.9.

Listing 5.9. Użycie środowiska **tcolorbox** do formatowania wyglądu tabeli

```
\begin{tcolorbox}[tabulars={r|c|l}, width=8cm, colframe=gray,
  coltext=black!80, title=Nagroda Nobla, halign title=center,
  fonttitle=\bfseries, colback=white]
\textbf{Rok} & \textbf{Osoba} & \textbf{Dyscyplina}\\
\hline\hline
1903 & Maria Skłodowska-Curie & fizyka     \\
1905 & Henryk Sienkiewicz     & literatura \\
```

```
1911 & Maria Skłodowska-Curie & chemia     \\
1924 & Władysław Reymont      & literatura \\
\end{tcolorbox}
```

Nagroda Nobla		
Rok	**Osoba**	**Dyscyplina**
1903	Maria Skłodowska-Curie	fizyka
1905	Henryk Sienkiewicz	literatura
1911	Maria Skłodowska-Curie	chemia
1924	Władysław Reymont	literatura

Efekt wizualny jest ciekawszy, jeżeli używamy kolorów innych niż odcienie szarości. Nie tylko kolor tła, ale również kolory linii oddzielających komórki zależą od opcji środowiska.

5.3 Zarządzanie odstępem między kolumnami

LaTeX domyślnie wstawia przed i po zawartości każdej kolumny poziomy odstęp wielkości **\tabcolsep**. Gwarantuje nam to, że na przykład zawartość komórek nie styka się ze sobą lub z liniami oddzielającymi kolumny. Możemy łatwo redefiniować wielkość tego odstępu za pomocą polecenia **\setlength**{}, np. **\setlength**{**\tabcolsep**}{3pt}. Takie ustawienie powinno istotnie zredukować ten odstęp (domyślnie 6 pt). Manipulowanie odstępem ma najczęściej na celu zwężenie tabeli, która przy domyślnych ustawieniach nie mieści się na szerokości strony. Możemy to połączyć ze zmniejszeniem czcionki w tabeli, co również skutkuje jej zwężeniem.

Przykład redukcji odstępu międzykolumnowego pokazano na listingu 5.10. Dane numeryczne pokazane na listingach 5.6 i 5.7 to fragment tablicy rozkładu χ^2. Nie tylko ograniczyliśmy się do 5 wierszy (5 stopni swobody), ale zostały pominięte trzy kolumny, bo nie mieściły się przy standardowych ustawieniach. Komplet kolumn mamy na listingu 5.10. Tym razem tabela zmieściła się – wystarczyło zmniejszyć standardowy odstęp do 2,75 pt. (linia 1).

Listing 5.10. Tabela ze zredukowanym odstępem międzykolumnowym

```
1 \setlength{\tabcolsep}{2.75pt}
2 \begin{tabular}{>{\bfseries}c|*{12}{c}}
3 df~\textbackslash~p & \textbf{0,99} & \textbf{0,95} &
4 \textbf{0,90} & \textbf{0,80} & \textbf{0,50} & \textbf{0,30} &
5 \textbf{0,20} & \textbf{0,10} & \textbf{0,05} & \textbf{0,02} &
6 \textbf{0,01}\\\hline
```

```
1 & 0,000 & 0,004 & 0,016 & 0,064 & 0,455 & 1,074 & 1,642 & ...
\end{tabular}
```

df \ p	0,99	0,95	0,90	0,80	0,50	0,30	0,20	0,10	0,05	0,02	0,01
1	0,000	0,004	0,016	0,064	0,455	1,074	1,642	2,706	3,841	5,12	6,635
2	0,020	0,103	0,211	0,446	1,386	2,408	3,219	4,605	5,991	7,824	9,210
3	0,115	0,352	0,584	1,005	2,366	3,665	4,642	6,251	7,815	9,837	11,34
4	0,297	0,711	1,064	1,649	3,357	4,878	5,989	7,779	9,488	11,67	13,28
5	0,554	1,145	1,610	2,343	4,351	6,064	7,289	9,236	11,07	13,39	15,09

Odstęp między kolumnami w tabeli można zdefiniować lokalnie za pomocą polecenia `@{}`. Polecenie takie możemy umieści przed i po literze definiującej kolumnę (lub po zamykającym nawiasie klamrowym, jeśli mamy np. kolumnę typu `p{}`). Jeżeli przyjmiemy, że polecenie **`\tabcolsep`** definiuje domyślny lewy i prawy margines kolumny, to polecenie `@{}` umieszczone przed literą definiującą kolumnę zastępuje swoim parametrem lewy margines, a polecenie umieszczone po literze – prawy margines. W szczególności definicja postaci `@{}c@{}` ustawia oba marginesy na zero.

Listing 5.11. Indywidualne sterowanie odstępem międzykolumnowym

```
1 \begin{tabular}
2 {|c|@{\hspace{3pt}}c|@{\hspace{12pt}}c|@{}c|@{}c@{}|@{\dots}c@{abc}|}
3 X & X & X & X & X & X \\
4 X & X & X & X & X & X \\
5 X & X & X & X & X & X \\
6 X & X & X & X & X & X \\
7 \end{tabular}
```

X	X	X	X	X	…Xabc
X	X	X	X	X	…Xabc
X	X	X	X	X	…Xabc
X	X	X	X	X	…Xabc

Przeanalizujmy przykład pokazany na listingu 5.11. Dla lepszego zobrazowania efektu zmiany odstępów między kolumnami użyte zostały wyłącznie pionowe linie. Wszystkie kolumny w swoich komórkach zawierają pojedynczą literę X (zobacz linie 3–6). W pierwszej kolumnie zachowano domyślne odstępy, w drugiej lewy odstęp ustawiono na 3 pt (połowa domyślnego), w trzeciej kolumnie – na 12 pt. W czwartej kolumnie lewy odstęp jest zerowy – widać, że litera X dotyka pionowej linii, a w piątej kolumnie oba odstępy są zerowe. Szósta kolumna to tylko pokazanie, że polecenie `@{}` naprawdę zastępuje odstęp swoim parametrem. W tym przypadku lewy odstęp zamieniono na wielokropek, a prawy na tekst „abc”. Indywidualne sterowanie odstępami stosujemy przede

wszystkim wtedy, gdy zmianę chcemy wprowadzić tylko w niektórych miejscach lub zmiany są różne w różnych kolumnach.

5.4 Dane numeryczne w tabelach

Tabele są często stosowane do czytelnej reprezentacji danych numerycznych. Istotny problem, jaki napotykamy, to brak możliwości wyrównania liczb względem kropki dziesiętnej wśród standardowych typów kolumn. Jeżeli w jednej kolumnie występują liczby różnego rzędu wielkości, to żadna z opcji `l`, `c` oraz `r` nie nadaje się do czytelnego reprezentowania takich danych. Problem, o którym mówimy, pokazano w tabeli 5.1. W każdej kolumnie dane są ułożone w porządku malejącym. Gdyby były one pomieszane, a tabela zajmowałaby całą stronę, analiza danych (np. znalezienie największej wartości) byłaby kłopotliwa.

Tabela 5.1. Próba umieszczenia danych numerycznych w standardowych kolumnach

Do lewej	Wyśrodkowane	Do prawej
123.4	123.4	123.4
12.34	12.34	12.34
1.234	1.234	1.234
0.1234	0.1234	0.1234

```
\usepackage{dcolumn}
```

Rozwiązaniem tego problemu może być zastosowanie pakietu `dcolumn` [6]. Pakiet ten definiuje specjalny typ kolumny `D{}{}{}`. Kolumna ta służy do wyrównania danych numerycznych względem kropki dziesiętnej, a także pozwala na konwersję separatora części dziesiętnej. Pierwszy parametr wskazuje znak, który jest użyty w kodzie źródłowym jako separator, drugi – znak, który ma być użyty w pliku wynikowym jako separator, a trzeci określa maksymalną liczbę cyfr przed i po separatorze – ma on formę `m.n`. Przykład użycia pakietu `dcolumn` pokazano na listingu 5.12.

Listing 5.12. Kolumny wyrównane względem kropki dziesiętnej – pakiet `dcolumn`

```
\begin{tabular}
{D{,}{.}{3.4}|D{.}{.}{2.3}|D{;}{.}{4.3}}
123,4  & 5.06  & 7;809 \\
12,34  & 0.056 & 789;09 \\
```

```
1,234  & 56.01 & 78;09 \\
0,1234 & 5.006 & 7890;009 \\
\end{tabular}
```

123.4	5.06	7.809
12.34	0.056	789.09
1.234	56.01	78.09
0.1234	5.006	7890.009

5.5 Scalanie komórek

We wszystkich dotychczasowych przykładach mieliśmy układ, w którym dana komórka należała dokładnie do jednej kolumny i jednego wiersza. Czasami zachodzi potrzeba scalenia w poziomie lub pionie dwóch lub większej liczby komórek. LaTeX oferuje polecenie **\multicolumn**{}{}{}, które pozwala scalać komórki w poziomie. Pierwszy parametr wskazuje na liczbę łączonych komórek, drugi zawiera definicję określającą sposób złożenia zawartości scalonej komórki (podobnie do definiowania kolumny w tabeli), a trzeci zawartość powstałej komórki. Wewnątrz scalonej komórki nie obowiązują linie pionowe ani sposób wyrównania określony w specyfikacji środowiska **tabular**, stąd wszystkie potrzebne informacje muszą być umieszczone w drugim parametrze.

Przykład scalania komórek z użyciem polecenia **\multicolumn** przedstawiono na listingu 5.13. Zacznijmy od analizy wyniku kompilacji naszego kodu. Tabela w przykładzie zawiera dane o minimalnej, średniej i maksymalnej temperaturze w Polsce w poszczególnych miesiącach 2023 r. Miesiące są pogrupowane w kwartały, a kwartały w rok i te dane znajdują się w scalonych komórkach. Dodatkowo chcemy uzyskać efekt „braku pierwszej komórki” w trzech pierwszych wierszach.

Listing 5.13. Przykład scalania komórek w poziomie

```
\begin{tabular}{|r|*{12}{r|}}
\cline{2-13}
\multicolumn{1}{c|}{} & \multicolumn{12}{c|}{2023}\\\cline{2-13}
\multicolumn{1}{c|}{} & \multicolumn{3}{c|}{kwartał 1} &
\multicolumn{3}{c|}{kwartał 2} & \multicolumn{3}{c|}{kwartał 3} &
\multicolumn{3}{c|}{kwartał 4}\\\cline{2-13}
\multicolumn{1}{c|}{} & I & II & III & IV & V & VI &
VII & VIII & IX & X & XI & XII \\\hline
min & -9,8 & -14,1 & -11,6 & -7,4 & -2,5 & -1,5 &
5,6 &  5,9 &  3,7 & -3,7 & -15,8 & -16,7 \\\hline
śr. &  2,9 &   1,5 &   4,5 &  7,7 & 12,8 & 17,5 &
```

```
19,3 & 19,6 & 17,7 & 10,9 &   4,2 &   2,6 \\\hline
max & 18,9 &  13,5 &  22,1 & 24,4 & 27,0 & 32,2 &
35,4 & 35,7 & 31,3 & 29,3 &  17,9 &  13,6 \\\hline
\end{tabular}
```

	2023											
	kwartał 1			kwartał 2			kwartał 3			kwartał 4		
	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII
min	-9,8	-14,1	-11,6	-7,4	-2,5	-1,5	5,6	5,9	3,7	-3,7	-15,8	-16,7
śr.	2,9	1,5	4,5	7,7	12,8	17,5	19,3	19,6	17,7	10,9	4,2	2,6
max	18,9	13,5	22,1	24,4	27,0	32,2	35,4	35,7	31,3	29,3	17,9	13,6

Gdy przeanalizujemy parametr środowiska **tabular** (linia 1), to widać, że tabela składa się z 13 kolumn z zawartością wyrównaną do prawej strony i mamy 14 pionowych linii. Skoro chcemy uzyskać efekt brakującej komórki w pierwszym wierszu, to pierwsza pozioma linia nie może obejmować tej komórki. Jest ona generowana poleceniem **\cline**{}. Parametrem tego polecenia jest zakres kolumn, które ma obejmować pozioma linia – u nas 2–13. To samo polecenie jest użyte po pierwszym i po drugim wierszu (linie 3 i 6). Zwróćmy teraz uwagę na linię 3. Zaczyna się ona od polecenia **\multicolumn** obejmującego tylko jedną kolumnę i w dodatku komórka ta jest pusta. W tym przypadku celem tego polecenia nie jest tak naprawdę scalanie komórek, ale usunięcie pionowych linii (nie obowiązują dla scalonej komórki). W drugim parametrze tego polecenia dodajemy pionową linię po prawej stronie komórki. Drugie użycie polecenia **\multicolumn** jest już klasyczne – łączymy komórki z 12 kolumn, zawartość komórki to 2023 i jest ona wyśrodkowana. W drugim parametrze polecenia dodajemy jeszcze pionową linię po prawej stronie komórki – nie dodajemy po lewej, bo ta linia już została dodana w poprzednim poleceniu **\multicolumn**. Drugi i trzeci wiersz tabeli również zaczynają się od pustej scalonej komórki obejmującej jedną kolumną. W efekcie pierwsza pionowa linia widoczna jest od trzeciego wiersza w dół. W drugim wierszu (linie 4–6) mamy jeszcze czterokrotnie użyte polecenie **\multicolumn** łączące trzy komórki.

\usepackage{multirow}

Jeżeli chcemy scalać komórki w pionie, to możemy zrobić to za pomocą pakietu multirow [50], który zawiera polecenie **\multirow**{}{}[]{}. Pierwszym parametrem obowiązkowym polecenia jest liczba wierszy do scalenia, drugim szerokość powstałej komórki, a trzecim jej zawartość. W przypadku drugiego parametru możemy użyć symbolu $*$ dla automatycznego wyznaczenia szerokości. Zawartość scalonej komórki jest domyślnie wyśrodkowana w pionie. Jeżeli

chcemy przesunąć zawartość, to możemy użyć parametru opcjonalnego. Dodatnia wartość spowoduje przesunięcie w górę, a ujemna w dół.

Listing 5.14. Przykład scalania komórek w pionie

```
1  \setlength{\tabcolsep}{4pt}
2  \begin{tabular}{|*{5}{r|}}
3  \cline{2-5}
4  \multicolumn{1}{c|}{} & \multicolumn{4}{c|}{2023} \\\cline{2-5}
5  \multicolumn{1}{c|}{} & &  min & śr. & max \\\hline
6  \multirow{3}{*}{kwartał I} & I & -9,8 & 2,9 & 18,9 \\\cline{2-5}
7  & II & -14,1 & 1,5 & 13,5 \\\cline{2-5}
8  & III & -11,6 & 4,5 & 22,1 \\\hline
9  \multirow{3}{*}{kwartał II} & IV & -7,4 & 7,7 & 24,4 \\\cline{2-5}
10 & V & -2,5 & 12,8 & 27,0 \\\cline{2-5}
11 & VI & -1,5 & 17,5 & 32,2 \\\hline
12 \end{tabular}
13 \hspace{3mm}
14 \begin{tabular}{|*{5}{r|}}
15 %...
16 \end{tabular}
```

		2023		
		min	śr.	max
kwartał I	I	-9,8	2,9	18,9
	II	-14,1	1,5	13,5
	III	-11,6	4,5	22,1
kwartał II	IV	-7,4	7,7	24,4
	V	-2,5	12,8	27,0
	VI	-1,5	17,5	32,2

		2023		
		min	śr.	max
kwartał III	VII	5,6	19,3	35,4
	VIII	5,9	19,6	35,7
	IX	3,7	17,7	31,3
kwartał IV	X	-3,7	10,9	29,3
	XI	-15,8	4,2	17,9
	XII	-16,7	2,6	13,6

Przykład zastosowania polecenia **multirow** przedstawiono na listingu 5.14. W tym celu przeorganizowane zostały dane z listingu 5.13 i dodatkowo podzielone na dwie tabele. Podział ten wykorzystamy jeszcze w podrozdziale 5.9. W pierwszym i drugim wierszu (linie 4 i 5) zastosowano motyw z poprzedniego przykładu, aby uzyskać brakujące komórki w lewym górnym rogu tabeli. Polecenie **\multirow** jest użyte dla wiersza trzeciego i szóstego (linie 6 i 9). W obu przypadkach scalamy trzy wiersze, więc w dwóch kolejnych wierszach scalana komórka zostaje pusta. W przedstawionym przykładzie domyślne wyśrodkowanie w pionie jest zadowalające, więc nie użyto parametru opcjonalnego. Druga tabela jest złożona analogicznie jak pierwsza. Nie ma między nimi łamania wiersza, więc zostały umieszczone obok siebie.

5.6 Komórki inne niż reszta

Jeżeli stosujemy kolumny typu `l`, `c` oraz `r`, to ich szerokość zależy od zawartości komórek w tych kolumnach. To ta najszersza komórka decyduje o szerokości całej kolumny. Czasami złamanie linii w pojedynczej komórce może istotnie zmniejszyć szerokość kolumny, a co za tym idzie – szerokość całej tabeli. Pakiet `makecell` [34] zawiera m.in. polecenie `\makecell[]{}`, które pozwala daną komórkę złożyć niezależnie od ustawień dla całej kolumny. Parametrem obowiązkowym jest zawartość komórki, przy czym w środku klamer możemy użyć polecenia `\\`, aby złamać tekst w komórce (oczywiście skutkuje to zwiększeniem wysokości danego wiersza). Parametr opcjonalny pozwala określić sposób wyrównania w pionie i/lub w poziomie. W celu określenia wyrównania w pionie używamy jednej z liter: `t`, `c` lub `b`. Dla określenia wyrównania w poziomie dostępne są opcje: `l`, `c`, `r` i `p{}`. Przykładowo: wartość `tl` oznacza, że chcemy dosunąć zawartość do góry i do lewej strony. Jeżeli chcemy skorzystać z opcji `p{}`, to musi ona być umieszczona w nawiasach klamrowych. Dodatkowo konieczny jest format `{}{p{...}}`, gdy chcemy określić tylko sposób wyrównania w poziomie. Polecenie występuje również w wersji `\makecell*[]{}`. W przypadku wersji z gwiazdką dodawane są odstępy w pionie wokół komórki.

```
\usepackage{makecell}
```

Listing 5.15. Składanie komórek tabeli z użyciem pakietu `makecell`

```
\begin{tabular}{|l|l|}
\hline
\textbf{Imię i nazwisko} & \textbf{Języki obce}\\\hline
Jan Kowalski & \makecell[l]{angielski\\ niemiecki}\\\hline
\makecell[tl]{Ewa\\ Nowak} & \makecell*[l]{angielski\\
francuski\\ hiszpański\\ niemiecki\\ rosyjski}\\\hline
\end{tabular}
```

Imię i nazwisko	**Języki obce**
Jan Kowalski	angielski niemiecki
Ewa Nowak	angielski francuski hiszpański niemiecki rosyjski

Na listingu 5.15 pokazano przykład użycia polecenia **\makecell**. Zmiany przede wszystkim dotyczą prawej kolumny, gdzie wygodnie można teraz złamać tekst i umieścić po jednym słowie w każdej linii. Użycie opcjonalnego parametru `l` pozwoliło wyrównać tekst do lewej strony zamiast domyślnego wyśrodkowania. Dodatkowo w trzecim wierszu (linia 5) użyto polecenia **\makecell***, aby zilustrować różnicę między obiema wersjami. Wyraźnie widoczne są większe odstępy przed pierwszym i po ostatnim słowie w prawej dolnej komórce. W linii 5 widzimy również użycie polecenia **\makecell** z parametrem opcjonalnym `tl`. Złamanie tekstu „Ewa Nowak" służy tutaj wyłącznie ilustracji znaczenia opcji `t` – górny wiersz komórki został umieszczony w połowie wysokości. Jeżeli komórka zawiera tylko jedną linię tekstu, to nie ma różnicy między poszczególnymi opcjami wyrównania w pionie. Tekst będzie po prostu wyśrodkowany.

Tabela 5.2. Przykład umieszczenia wzorów matematycznych w tabeli

funkcja	$f(x) = x^2$
całka nieoznaczona	$\int x^2 dx$
całka oznaczona	$\int_2^5 x^2 dx$

W tabeli 5.2 pokazano typowy problem, jaki często napotykamy, gdy umieszczamy w tabeli wzory matematyczne, które są wyższe niż standardowy tekst. Widać wyraźnie, że w kilku miejscach mamy za małą przestrzeń między wzorem i ramką. Problem ten można rozwiązać, stosując polecenia **\gape**`[]{}` i **\Gape**`[][]{}` z pakietu `makecell`. Oba przyjmują jako obowiązkowy parametr zawartość komórki. Polecenie **\gape** bez parametru opcjonalnego wstawia dodatkowy odstęp od góry i od dołu (domyślnie 3 pt). Jeżeli chcemy dodać odstęp tylko od góry, dodajemy parametr opcjonalny o wartości `t`, jeśli tylko od dołu – o wartości `b`. Polecenie **\Gape** bez parametrów opcjonalnych jest równoważne **\gape**`{}`. Jeżeli podamy jeden parametr opcjonalny, to wskazuje on, ile przestrzeni należy dodać od góry i od dołu (lub odjąć, jeśli podamy ujemną wartość). Przy dwóch parametrach opcjonalnych pierwszy wskazuje zmiany od góry, drugi od dołu.

Listing 5.16. Modyfikacja odstępu w komórkach tabeli z użyciem pakietu `makecell`

```
1 \begin{tabular}{|l|l|}
2 \hline
3 funkcja & \gape[t]{$f(x) = x^2$} \\\hline
4 całka nieoznaczona & \gape{$\int x^2 dx$} \\\hline
5 całka oznaczona & \Gape[3pt][4pt]{$\int_2^5 x^2 dx$} \\\hline
6 \end{tabular}
```

funkcja	$f(x) = x^2$
całka nieoznaczona	$\int x^2 dx$
całka oznaczona	$\int_2^5 x^2 dx$

Przykład wykorzystania obu poleceń pokazano na listingu 5.16. W pierwszym wierszu (linia 3) dodano odstęp od góry. W pozostałych wierszach odstępy zostały dodane po obu stronach (polecenie **\gape** w linii 4 i **\Gape** w linii 5 – tutaj dodano różnej wielkości odstępy od góry i od dołu). Po przeprowadzonej korekcie tabela z wzorami prezentuje się zdecydowanie lepiej.

5.7 Formatowanie nagłówka tabeli

Pakiet `makecell` zawiera również polecenia pozwalające wygodnie zarządzać wyglądem nagłówka tabeli. Często zawartość nagłówka składana jest pogrubioną czcionką, co do tej pory osiągaliśmy, powtarzając polecenie **\textbf** w każdej komórce. Podejście takie jest uciążliwe i w razie konieczności zmiany wyglądu wielu nagłówków tabel w większym opracowaniu bardzo niewygodne. Pakiet zawiera polecenie **\thead** służące do oznaczenia komórek nagłówka i ustawienia dla nich wspólnego formatowania.

Listing 5.17. Przykład zarządzania nagłówkiem tabeli z użyciem pakietu `makecell`

```
1  \renewcommand\theadfont{\bfseries\small}
2  \renewcommand\theadalign{tc}
3  \renewcommand\theadgape{\Gape[4pt][2pt]}
4
5  \begin{tabular}{|l|*{3}{r|}}
6  \hline
7  \thead{Nazwa planety} & \thead{Odległość od Słońca\\ (w mln km)} &
8  \thead{Średnica równika\\ (w km)} &
9  \thead{Liczba znanych\\ księżyców} \\\hline
10 Merkury & 57,9 & 4878 & 0 \\
11 Wenus & 108,2 & 12102 & 0 \\
12 Ziemia & 149,6 & 12756 & 1 \\
13 Mars & 227,9 & 6794 & 2 \\
14 Jowisz & 778,4 & 142984 & 63 \\\hline
15 \end{tabular}
```

Nazwa planety	**Odległość od Słońca (w mln km)**	**Średnica równika (w km)**	**Liczba znanych księżyców**
Merkury	57,9	4878	0
Wenus	108,2	12102	0
Ziemia	149,6	12756	1
Mars	227,9	6794	2
Jowisz	778,4	142984	63

Przykład elastycznego zarządzania wyglądem nagłówka tabeli pokazano na listingu 5.17. Przede wszystkim w liniach 1–3 zdefiniowane zostały parametry składania komórek nagłówka: czcionka (linia 1), sposób wyrównania (linia 2) i pionowe odstępy od góry i dołu (linia 3). Teraz wystarczy zawartość komórki nagłówka podać jako parametr polecenia `\thead{}`, aby została ona złożona zgodnie z ustalonymi parametrami.

Ustawienia z linii 1–3 są trwałe, tj. dopóki ich nie zmienimy, będą stosowane do kolejnych tabel. Łatwo więc ustawić je na początku dokumentu, a uzyskamy jednolity wygląd wszystkich nagłówków tabel. Równie łatwo jest wówczas zmienić parametry składu dla wszystkich tabel. Dodatkowo zaletą polecenia `\thead` jest możliwość łamania tekstu w komórce za pomocą polecenia `\\` analogicznie jak w przypadku polecenia `makecell`. Z możliwości tej skorzystaliśmy w liniach 7–9. Polecenie występuje również w wersji z gwiazdką (`\thead*{}`). W tym wariancie polecenia wprowadzane są dodatkowe odstępy w pionie między zawartością komórki nagłówka a obramowaniem.

```
\usepackage{makecell}
\usepackage{rotating}
```

Czasami ze względu na długość treści w komórkach nagłówka tabeli warto obrócić tekst o 90°. Możemy to zrobić za pomocą polecenia `\rothead{}` z pakietu `makecell` (wymagane włączenie pakietu `rotating` [14]). Przykład pokazano na listingu 5.18. Polecenie użyte w linii 3 pozwala ustawić wysokość wiersza nagłówkowego na podstawie najdłuższej linii tekstu z nagłówka. W pozostałej części kodu zmieniamy tylko (w porównaniu z listingiem 5.17) polecenie `\thead` na `\rothead`. W pakiecie `makecell` znajdziemy także polecenie `\rotcell{}`, które umożliwia obrócenie tekstu we wskazanej komórce tabeli. Można również ustawić kąt obrotu w miejsce domyślnych 90°.

Listing 5.18. Nagłówek tabeli z obróconym tekstem – pakiet `makecell`

```
1 \renewcommand\theadfont{\bfseries\small}
2 \renewcommand\theadalign{tl}
```

```
\settowidth\rotheadsize{\theadfont Odległość od Słońca}

\begin{tabular}{|l|*{3}{r|}}
\hline
\rothead{Nazwa planety} & \rothead{Odległość od Słońca\\ (w mln km)}
& \rothead{Średnica równika (w km)} &
\rothead{Liczba znanych księżyców} \\\hline
%...
\end{tabular}
```

Nazwa planety	Odległość od Słońca (w mln km)	Średnica równika (w km)	Liczba znanych księżyców
Merkury	57,9	4878	0
Wenus	108,2	12102	0
Ziemia	149,6	12756	1
Mars	227,9	6794	2
Jowisz	778,4	142984	63

Jeżeli budujemy tabelę, w której chcemy etykietować zarówno kolumny, jak i wiersze, może pojawić się potrzeba umieszczenia dwóch etykiet w komórce w lewym górnym rogu i diagonalnego jej podziału. Przykład taki przedstawiono na listingu 5.19.

Listing 5.19. Podział diagonalny komórki – pakiet `makecell`

```
\renewcommand\theadfont{\bfseries\small}
\renewcommand\theadalign{cc}
\begin{tabular}{|>{\bfseries\small}l|*{3}{c|}}
\hline
\diaghead(2,-1){xxxxxxxxxxx}{Miesiąc}{Rok}
 & \thead{2022} & \thead{2023} & \thead{2024} \\\hline
Styczeń  & 31203 & 28754 & 30562 \\
Luty     & 28564 & 25675 & 27896 \\
Marzec   & 34612 & 31222 & 32454 \\
Kwiecień & 30896 & 29564 & 31112 \\\hline
\end{tabular}
```

Rok / Miesiąc	2022	2023	2024
Styczeń	31203	28754	30562
Luty	28564	25675	27896
Marzec	34612	31222	32454
Kwiecień	30896	29564	31112

Podział diagonalny komórki nagłówkowej realizowany jest przez polecenie **\diaghead** (linia 5). Liczby podane w nawiasie okrągłym (`(2,-1)`) określają proporcje dzielonej komórki (tutaj 2:1), zaś znak minus określa kierunek podziału (od lewego górnego rogu do prawego dolnego). Jeżeli usuniemy znak minus, to przekątna będzie łączyła prawy górny i lewy dolny róg. Na podstawie długości pierwszego parametru w nawiasach klamrowych (tutaj `xxxxxxxxxxx`) określana jest szerokość komórki. Pozostałe dwa parametry zawierają etykiety do umieszczenia w dzielonej komórce. Etykiety są składane zgodnie z ustawieniami, z których korzysta polecenie **\thead**.

5.8 Zagnieżdżanie tabel

W przykładzie na listingu 5.2 pokazaliśmy, że można użyć środowiska **tabular**, aby ułożyć obok siebie trzy obrazki. W komórkach tabeli możemy umieścić również kolejną tabelę. Powiedzmy, że chcemy wykorzystać efekt zagnieżdżania tabel do zobrazowania związku jeden-do-wielu w relacyjnych bazach danych. W naszym przykładzie będzie to układ: jedna faktura zawiera w sobie wiele pozycji. Zawartość takiej faktury umieścimy w zagnieżdżonej tabeli. Przykład przedstawiono na listingu 5.20. W linii 5 rozpoczyna się zawartość drugiego wiersza głównej tabel. Zawartością czwartej komórki (linie 6–11) jest kolejna tabela. Dodatkowo została umieszczona ona jako parametr polecenia **\gape**, aby uzyskać przestrzeń nad i pod tabelą w komórce. W ten sposób ramki obu tabel nie pokrywają się. Podobnie druga tabela jest zagnieżdżona w wierszu trzecim (linie 13–19).

Listing 5.20. Przykład zagnieżdżania tabel

```
1 \begin{tabular}{|c|l|l|l|}
2 \hline
3 \thead{Nr\_faktury} & \thead{Imię} &
4 \thead{Nazwisko} & \thead{Zamówienie} \\\hline
5 2025007 & Adam & Kowalski &
6 \gape{\begin{tabular}{|c|l|l|l|}
7 \hline
```

```
\thead{Nr} & \thead{Nazwa} & \thead{Cena} & \thead{Sztuk} \\\hline
1 & mleko & 4,50 & 12 \\
2 & mąka & 3,20 & 10 \\\hline
\end{tabular}} \\\hline
2025012 & Ewa & Nowak &
\gape{\begin{tabular}{|c|l|l|l|}
\hline
\thead{Nr} & \thead{Nazwa} & \thead{Cena} & \thead{Sztuk} \\\hline
1 & czekolada & 6,20 & 10 \\
2 & lody & 16,50 & 4 \\
3 & jogurt & 2,60 & 12 \\\hline
\end{tabular}} \\\hline
\end{tabular}
```

Nr_faktury	Imię	Nazwisko	Zamówienie
2025007	Adam	Kowalski	**Nr** **Nazwa** **Cena** **Sztuk** 1 mleko 4,50 12 2 mąka 3,20 10
2025012	Ewa	Nowak	**Nr** **Nazwa** **Cena** **Sztuk** 1 czekolada 6,20 10 2 lody 16,50 4 3 jogurt 2,60 12

5.9 Tabele jako wstawki

Do tej pory zajmowaliśmy się zagadnieniami konstruowania tabel. Żadna z nich (przykłady pokazane na listingach) nie zawierała definicji etykiety lub podpisu. Jeżeli użyjemy wyłącznie środowiska **tabular**, to tabela jest traktowana podobnie jak ministrony. Na listingu 5.14 mieliśmy przykład umieszczenia dwóch tabel w jednym wierszu. Podobnie jak w przykładzie z rys. 3.2, moglibyśmy w tym samym wierszu umieścić tekst i środowisko **tabular** zostałoby wyśrodkowane w pionie względem linii bazowej wiersza. Możemy również dodać parametr opcjonalny, jeżeli domyślnie rozmieszczenie w pionie nam nie odpowiada: `t` – na linii bazowej znajdzie się górna krawędź tabeli, `b` – na linii bazowej znajdzie się dolna krawędź tabeli.

W publikacjach tabele najczęściej występują z podpisem umieszczonym nad tabelą i są numerowane. Jeżeli chcemy uzyskać taki efekt, to tabelę, np. złożoną z użyciem środowiska **tabular**, musimy umieścić wewnątrz środowiska **table**.

To właśnie wewnątrz tego środowiska (podobnie jak w środowisku **figure** dla rysunków) umieszczamy również polecenia **\caption** i **\label** (koniecznie w tej kolejności i bezpośrednio po sobie), które definiują podpis i etykietę tabeli.

Listing 5.21. Przykład umieszczenia dwóch tabel w środowisku **table**

```
\begin{table}[ht]
\setlength{\tabcolsep}{4pt}
\caption{Temperatury w Polsce w 2023 r.}
\label{tab:temperatury}
\begin{tabular}{|*{5}{r|}}
%...
\end{tabular}\hspace{3mm}
\begin{tabular}{|*{5}{r|}}
%...
\end{tabular}
\end{table}
```

Tabela 5.3. Temperatury w Polsce w 2023 r.

	2023			
		min	śr.	max
kwartał I	I	-9,8	2,9	18,9
	II	-14,1	1,5	13,5
	III	-11,6	4,5	22,1
kwartał II	IV	-7,4	7,7	24,4
	V	-2,5	12,8	27,0
	VI	-1,5	17,5	32,2

	2023			
		min	śr.	max
kwartał III	VII	5,6	19,3	35,4
	VIII	5,9	19,6	35,7
	IX	3,7	17,7	31,3
kwartał IV	X	-3,7	10,9	29,3
	XI	-15,8	4,2	17,9
	XII	-16,7	2,6	13,6

Przykład użycia środowiska **table** pokazano na listingu 5.21, a efekt kompilacji tego kodu widzimy w tabeli 5.3. Podobnie jak w przypadku środowiska **figure**, nie ma znaczenia, co umieścimy wewnątrz środowiska **table** – zostanie to uznane za tabelę (nawet jeżeli będzie to akapit tekstu). W naszym przykładzie zamieściliśmy dwa środowiska **tabular** z listingu 5.14. Zostały umieszczone razem w środowisku **table**, więc są traktowana jako jedna tabela o numerze 5.3.

Zastosowanie środowiska **table** nie ogranicza się do możliwości zdefiniowania podpisu tabeli i nadania jej numeru. Środowisko to tworzy tzw. wstawkę (obiekt pływający). Jeżeli wstawka nie mieści się na danej stronie, LaTeX automatycznie przenosi ją na kolejną stronę, uzupełniając pozostałe miejsce tekstem. LaTeX przechowuje wstawki danego typu w kolejce FIFO (First In First Out), tzn. że być może zajdzie potrzeba przesuwania wstawek między stronami, ale zostaną one umieszczone w pliku `pdf` dokładnie w takiej samej kolejności jak w pliku źródłowym. Warto pamiętać, że kolejka dotyczy użycia środowiska

table. Tabele wielostronicowe z kolejnego podrozdziału nie są w niej umieszczane. Również tabela złożona z użyciem środowiska **tabular**, ale nieobudowana środowiskiem **table** nie znajdzie się w kolejce.

Jeżeli umieścimy w naszym dokumencie za dużą tabelę (nie mieści się na stronie), to LaTeX przesunie ją na koniec rozdziału (lub dokumentu). Za nią umieści wszystkie pozostałe tabele z tego rozdziału (dokumentu). Wynika to z faktu, że musi przestrzegać zasad funkcjonowania kolejki FIFO. W celu naprawy sytuacji musimy zmodyfikować kłopotliwą tabelę (zmniejszyć) i wtedy po rekompilacji sytuacja wróci do normy. Analogiczny problem dotyczy rysunków – mają one swoją kolejkę FIFO.

Środowisko **table** (podobnie jak **figure**) przyjmuje opcjonalny parametr, który określa, gdzie LaTeX może umieścić wstawkę. Parametr ten może zawierać kombinację poniższych znaków (kolejność nie jest istotna):

- `!` – pominięcie większości parametrów kontrolujących rozmieszczanie wstawek (np. maksymalna dopuszczalna liczba wstawek na stronie) i „próba wymuszenia" zastosowania którejś z pozostałych opcji;
- `h` – bez przemieszczania (dokładnie w miejscu użycia) – jeżeli nie jest to możliwe i nie dodano żadnych innych „literkowych" opcji, to LaTeX przyjmuje `t` jako alternatywę;
- `t` – na górze strony;
- `b` – na dole strony;
- `p` – na stronie zawierającej wyłącznie wstawki.

Dla przykładu: wartość `tb` oznacza, że wstawka zostanie umieszczona na górze lub na dole strony. Jeżeli nie jest możliwe umieszczenie na stronie bieżącej, to nastąpi przesunięcie na kolejną stronę.

Jeżeli dołączymy pakiet `float` [35], to będziemy mieć do dyspozycji jeszcze opcję `H` (musi ona wystąpić samotnie). Oznacza ona, że w przypadku, gdy nie można wstawki umieścić w miejscu, które wynika z kodu, to jest ona przenoszona na następną stronę, ale nie następuje przesunięcie treści – strona zostaje niekompletna.

W kilku przykładach w bieżącym rozdziale analizowaliśmy sytuacje, w których modyfikowany był odstęp między kolumnami, aby tabela zmieściła się między marginesami. Z perspektywy rozmieszczania wstawek większym problemem jest zbyt duża wysokość tabeli. Możliwa jest modyfikacja domyślnej wysokości wierszy. Realizujemy to poprzez redefinicję polecenia **\arraystretch**

– zazwyczaj w celu zwiększenia wysokości wiersza, jak to pokazano na listingu 5.22. Nowa wartość użyta w linii 1 oznacza, że wysokość wierszy będzie o 15% większa od domyślnej. Zmiana jest trwała (dla wszystkich kolejnych tabel). Jeżeli chcemy przywrócić domyślne ustawienia, zmieniamy wartość na 1 (linia 5). Jeżeli nasza tabela jest nieznacznie za wysoka, możemy ustawić wartość np. na 0.98. Wizualnie będzie to nie do zauważenia, a może wystarczy, aby tabela zmieściła się na stronie. Innym rozwiązaniem jest zmniejszenie czcionki dla tabeli.

Listing 5.22. Przykład zmiany domyślnej wysokości wierszy tabeli

```
1 \renewcommand{\arraystretch}{1.15}
2 \begin{tabular}{|*{5}{r|}}
3 %...
4 \end{tabular}
5 \renewcommand{\arraystretch}{1}
```

	2023			
		min	śr.	max
kwartał I	I	-9,8	2,9	18,9
	II	-14,1	1,5	13,5
	III	-11,6	4,5	22,1
kwartał II	IV	-7,4	7,7	24,4
	V	-2,5	12,8	27,0
	VI	-1,5	17,5	32,2

```
\usepackage{rotating}
```

Listing 5.23. Przykład wykorzystania środowiska **sidewaystable**

```
1 \begin{sidewaystable}[...]
2 \caption{...}
3 \label{tab:...}
4 \begin{tabular}
5 %...
6 \end{tabular}
7 \end{sidewaystable}
```

LaTeX umożliwia złożenie tabeli obróconej o 90 stopni. Tabela taka jest obracana wraz z podpisem. W takiej sytuacji możemy wykorzystać wysokość strony

jako maksymalną szerokość tabeli. Taki skład oferuje środowisko **sidewaystable** z pakietu `rotating` [14], którego używamy zamiast **table**. Przykład pokazano na listingu 5.23. Tabela taka jest obiektem pływającym i jest umieszczana w tej samej kolejce FIFO co tabele definiowane z użyciem środowiska **table**. Obrócona tabela jest składana na oddzielnej stronie (zob. tabela 9.1, str. 187).

5.10 Tabele wielostronicowe

Jeżeli przygotowujemy dłuższe zestawienie (np. dane statystyczne), to może się okazać, że nasza tabela będzie miała nawet kilka stron. Oczywiście w takim przypadku nie może być to wstawka. Rozwiązanie zawiera pakiet `longtable` [8], a jest nim środowisko o takiej samej nazwie. Środowiska **longtable** używamy zamiast pary środowisk **table** i **tabular**.

```
\usepackage{longtable}
```

Listing 5.24. Definiowanie nagłówków i stopek dla tabeli w środowisku **longtable**

```
\setlength{\tabcolsep}{4pt}
\begin{longtable}{|lc|lc|lc|}
\caption{Wybrane symbole używane w trybie matematycznym
\label{tab:symbole}}\\
\hline
\textbf{Polecenie} && \textbf{Polecenie} &&
\textbf{Polecenie} &\\\hline
1 & 2 & 3 & 4 & 5 & 6 \\\hline
\endfirsthead
\hline
1 & 2 & 3 & 4 & 5 & 6 \\\hline
\endhead
\hline
\multicolumn{6}{|c|}{\textit{\scriptsize --- Ciąg dalszy
na następnej stronie ---}}\\\hline
\endfoot
\hline
\multicolumn{6}{|c|}{\textit{\scriptsize --- Więcej
w rozdziale~\ref{cha:wzory:matematyczne} ---}}\\\hline
\endlastfoot
\multicolumn{6}{|c|}{Litery greckie}\\\hline
\lstinline!\alpha! & $\alpha$ & \lstinline!\pi! & $\pi$ &
\lstinline!\varsigma! & $\varsigma$ \\
```

```
24 \lstinline!\beta! & $\beta$ & \lstinline!\psi! & $\psi$ &
25 \lstinline!\vartheta! & $\vartheta$ \\
26 \lstinline!\chi! & $\chi$ & \lstinline!\rho! & $\rho$  &
27 \lstinline!\Delta! & $\Delta$ \\
28 %...
29 \end{longtable}
```

Na listingu 5.24 pokazano kod źródłowy tabeli 5.4, przy czym prezentację zawartości tabeli ograniczono do początkowych wierszy. W tabeli umieszczono wybrane polecenia, które służą do generowania symboli używanych w trybie matematycznym. Wzorom matematycznym poświęcony jest wprawdzie rozdział 6, ale ponieważ zależało mi na pokazaniu tabeli, która obejmuje choć częściowo trzy strony, takie wykorzystanie przestrzeni było bardziej racjonalne niż umieszczanie w niej np. losowych liczb. Sama tabela składa się z sześciu kolumn zawierających polecenia i generowane przez nie symbole, które dodatkowo podzielono na podzbiory, np. litery greckie, operatory binarne itd. – etykieta każdego podzbioru została umieszczona w scalonej komórce (zobacz linię 21). Liczbę kolumn i sposób wyrównania ich zawartości określamy analogicznie jak w przypadku środowiska **tabular** (linia 2). Występuje drobna różnica między środowiskami **longtable** i **table**, jeżeli chodzi o zdefiniowanie podpisu tabeli. W przypadku **longtable** po nawiasie klamrowym zamykającym parametr polecenia **\caption** umieszczamy polecenie \\ (podpis jest traktowany jak wiersz tabeli złożony w scalonej komórce), a definicję etykiety (polecenie **\label**) umieszczamy razem z treścią podpisu (linia 4).

Tabela 5.4. Wybrane symbole używane w trybie matematycznym

Polecenie		**Polecenie**		**Polecenie**	
1	2	3	4	5	6
Litery greckie					
\alpha	α	**\pi**	π	**\varsigma**	ς
\beta	β	**\psi**	ψ	**\vartheta**	ϑ
\chi	χ	**\rho**	ρ	**\Delta**	Δ
\delta	δ	**\sigma**	σ	**\Gamma**	Γ
\epsilon	ϵ	**\tau**	τ	**\Lambda**	Λ
\eta	η	**\theta**	θ	**\Omega**	Ω
\gamma	γ	**\upsilon**	υ	**\Phi**	Φ
\iota	ι	**\xi**	ξ	**\Pi**	Π
\kappa	κ	**\zeta**	ζ	**\Psi**	Ψ
\lambda	λ	**\digamma**	$\digamma$	**\Sigma**	Σ

— Ciąg dalszy na następnej stronie —

1	2	3	4	5	6
`\mu`	μ	`\varepsilon`	ε	`\Theta`	Θ
`\nu`	ν	`\varkappa`	$\varkappa$	`\Upsilon`	Υ
o	o	`\varphi`	φ	`\Xi`	Ξ
`\omega`	ω	`\varpi`	ϖ		
`\phi`	ϕ	`\varrho`	ϱ		
Operatory binarne					
`\ast`	$*$	`\oplus`	$\oplus$	`\uplus`	$\uplus$
`\cdot`	$\cdot$	`\ominus`	$\ominus$	`\sqcup`	$\sqcup$
`\circ`	$\circ$	`\otimes`	$\otimes$	`\sqcap`	$\sqcap$
`\bullet`	$\bullet$	`\oslash`	$\oslash$	`\setminus`	$\setminus$
`\diamond`	$\diamond$	`\odot`	$\odot$	`\wr`	$\wr$
`\times`	$\times$	`\boxplus`	$\boxplus$	`\triangleleft`	$\triangleleft$
`\div`	$\div$	`\boxminus`	$\boxminus$	`\triangleright`	$\triangleright$
`\pm`	$\pm$	`\boxtimes`	$\boxtimes$	`\vee`	$\vee$
`\mp`	$\mp$	`\boxdot`	$\boxdot$	`\wedge`	$\wedge$
`\star`	$\star$	`\cup`	$\cup$	`\veebar`	$\veebar$
`\centerdot`	$\centerdot$	`\cap`	$\cap$	`\barwedge`	$\barwedge$
Operatory relacyjne					
`\equiv`	$\equiv$	`\leq`	$\leq$	`\in`	$\in$
`\cong`	$\cong$	`\geq`	$\geq$	`\ni`	$\ni$
`\sim`	$\sim$	`\leqq`	$\leqq$	`\notin`	$\notin$
`\simeq`	$\simeq$	`\geqq`	$\geqq$	`\subset`	$\subset$
`\approx`	$\approx$	`\leqslant`	$\leqslant$	`\subseteq`	$\subseteq$
`\approxeq`	$\approxeq$	`\geqslant`	$\geqslant$	`\supset`	$\supset$
`\neq`	$\neq$	`\prec`	$\prec$	`\supseteq`	$\supseteq$
`\ncong`	$\ncong$	`\preceq`	$\preceq$	`\subsetneq`	$\subsetneq$
`\nsim`	$\nsim$	`\succ`	$\succ$	`\supsetneq`	$\supsetneq$
`\nprec`	$\nprec$	`\succeq`	$\succeq$	`\mid`	$\mid$
`\models`	$\models$	`\ll`	$\ll$	`\parallel`	$\parallel$
`\vdash`	$\vdash$	`\gg`	$\gg$	`\between`	$\between$
`\dashv`	$\dashv$	`\lll`	$\lll$	`\bowtie`	$\bowtie$
`\perp`	$\perp$	`\ggg`	$\ggg$	`\Join`	$\Join$
Symbole strzałek					
`\leftarrow`	$\leftarrow$	`\longleftarrow`	$\longleftarrow$	`\uparrow`	$\uparrow$
`\Leftarrow`	$\Leftarrow$	`\Longleftarrow`	$\Longleftarrow$	`\Uparrow`	$\Uparrow$
`\rightarrow`	$\rightarrow$	`\longrightarrow`	$\longrightarrow$	`\downarrow`	$\downarrow$
`\Rightarrow`	$\Rightarrow$	`\Longrightarrow`	$\Longrightarrow$	`\Downarrow`	$\Downarrow$
`\leftrightarrow`	$\leftrightarrow$	`\longleftrightarrow`	$\longleftrightarrow$	`\updownarrow`	$\updownarrow$

— Ciąg dalszy na następnej stronie —

1	2	3	4	5	6
`\Leftrightarrow`	$\Leftrightarrow$	`\Longleftrightarrow`	$\Longleftrightarrow$	`\Updownarrow`	$\Updownarrow$
`\mapsto`	$\mapsto$	`\leftrightarrows`	$\leftrightarrows$	`\nearrow`	$\nearrow$
`\longmapsto`	$\longmapsto$	`\rightleftarrows`	$\rightleftarrows$	`\searrow`	$\searrow$
`\leadsto`	$\leadsto$	`\rightrightarrows`	$\rightrightarrows$	`\swarrow`	$\swarrow$
`\nrightarrow`	$\nrightarrow$	`\leftleftarrows`	$\leftleftarrows$	`\nwarrow`	$\nwarrow$
`\nleftarrow`	$\nleftarrow$	`\twoheadrightarrow`	$\twoheadrightarrow$	`\dashrightarrow`	$\dashrightarrow$
`\nRightarrow`	$\nRightarrow$	`\twoheadleftarrow`	$\twoheadleftarrow$	`\dashleftarrow`	$\dashleftarrow$
`\nLeftarrow`	$\nLeftarrow$	`\downdownarrows`	$\downdownarrows$	`\upuparrows`	$\upuparrows$
Symbole z zmiennej wielkości					
`\sum`	$\sum$	`\bigcup`	$\bigcup$	`\bigoplus`	$\bigoplus$
`\prod`	$\prod$	`\bigcap`	$\bigcap$	`\bigotimes`	$\bigotimes$
`\coprod`	$\coprod$	`\bigvee`	$\bigvee$	`\bigodot`	$\bigodot$
`\biguplus`	$\biguplus$	`\bigwedge`	$\bigwedge$	`\int`	$\int$
Symbole różne					
`\forall`	$\forall$	`\cdots`	$\cdots$	`\clubsuit`	$\clubsuit$
`\exists`	$\exists$	`\vdots`	$\vdots$	`\diamondsuit`	$\diamondsuit$
`\partial`	∂	`\ldots`	$\ldots$	`\heartsuit`	$\heartsuit$
`\nabla`	∇	`\ddots`	$\ddots$	`\spadesuit`	$\spadesuit$
`\imath`	$\imath$	`\emptyset`	$\emptyset$	`\Diamond`	$\Diamond$
`\jmath`	$\jmath$	`\infty`	∞	`\square`	$\square$
— Więcej w rozdziale 6 —					

Zanim będziesz czytał kolejne zdanie, przeanalizuj dokładnie, jak zaczyna i kończy się tabela 5.4 na każdej stronie, na której jest zamieszczona. Znajdź elementy, które się powtarzają. Cechą charakterystyczną długich tabel są nagłówki i stopki, które pojawiają się na każdej stronie, na której znajduje się choćby fragment tabeli. Możemy wyróżnić cztery takie części:

– nagłówek początkowy – występuje raz na początku tabeli;

– nagłówek kontynuacji – występuje na każdej stronie, na której kontynuowana jest tabela;

– stopka kontynuacji – występuje na końcu każdej strony, która nie jest ostatnią;

– stopka końcowa – występuje raz na końcu tabeli.

Przeanalizujmy teraz, w jaki sposób te części zostały zdefiniowane na listingu 5.4. Definiowanie nagłówka początkowego rozpoczyna się bezpośrednio po poleceniu `\caption` (tutaj linia 5), a kończy się poleceniem `\endfirsthead`

(linia 9). W naszej tabeli nagłówek ten składa się z dwóch wierszy: pierwszy zawiera trzykrotnie powtórzone słowo „Polecenie", a drugi numery kolumn. Po poleceniu **\endfirsthead** rozpoczyna się definiowanie nagłówka kontynuacji, które kończy się poleceniem **\endhead** (linia 12). W naszym przypadku nagłówek ten zawiera wyłącznie wiersz z numerami kolumn i jest on użyty dwukrotnie na stronach, na których kontynuowana jest tabela. Po poleceniu **\endhead** rozpoczyna się definiowanie stopki kontynuacji, które kończy się poleceniem **\endfoot** (linia 16). Ta stopka została zdefiniowana jako scalona komórka z tekstem „Ciąg dalszy na następnej stronie". Jest ona użyta na końcu pierwszej i drugiej strony tabeli. Po poleceniu **\endfoot** definiowana jest stopka końcowa, która kończy się poleceniem **\endlastfoot** (linia 20). W naszym przypadku w stopce tej znajduje się odwołanie do rozdziału poświęconego wzorom matematycznym. Po poleceniu **\endlastfoot** zamieszczamy właściwą zawartość tabeli. Dla kompletu warto dodać, że polecenie **\lstinline** jest tu używane do eleganckiego składania kodu źródłowego. Więcej na ten temat znajdziesz w podrozdziale 8.2.

Zdefiniowane w powyższym przykładzie stopki miały na celu przede wszystkim demonstrację możliwości środowiska **longtable**. W przypadku danych numerycznych stopki kontynuacji można użyć do podsumowania danych na konkretnej stronie, a stopki końcowej do ich globalnego podsumowania.

Rozdział 6

Wzory matematyczne

Jak powiedziałem we wstępie, system TEX został opracowany przez Donalda E. Knutha z myślą o składzie prac naukowo-technicznych, których nieodłącznym elementem są wzory matematyczne. Kilka wzorów przewinęło się przez nasze wcześniejsze przykłady. W rozdziale 5 znajduje się tabela (tab. 5.4, str. 113) zawierająca blisko 200 poleceń generujących różne symbole w trybie matematycznym, co stanowi i tak tylko cząstkę możliwości LATEX-a w tym aspekcie. W tym rozdziale dokonamy systematycznego przeglądu, jak składać różnego rodzaju konstrukcje matematyczne, z którymi możemy się spotkać, przygotowując LATEX-owy dokument.

6.1 Tryby matematyczne

Możemy wyróżnić dwa tryby matematyczne, w których LATEX składa wzory: tryb matematyczny wewnątrzakapitowy i tryb matematyczny eksponowany. Z tym pierwszym trybem spotkaliśmy się już wcześniej. Wzór taki, składany między znakami `$`, jest najczęściej umieszczony wewnątrz akapitu tekstu. W trybie eksponowanym wzór składany jest zazwyczaj wewnątrz odpowiedniego środowiska i może mieć przypisany numer. Wzory eksponowane są domyślnie wyśrodkowane, ale możemy to zmienić, stosując w poleceniu **`\documentclass`** opcję `fleqn`. W takim przypadku wzory są przesunięte do lewej strony. Wielkość wcięcia dla wzorów możemy wówczas ustawić, zmieniając wartość polecenia **`\mathindent`**, np. ustawiając je w preambule na 1 cm: **`\setlength`**`{`**`\mathindent`**`}{1cm}`.

Zanim przejdziemy do omawiania dostępnych środowisk, kilka słów o dodatkowych pakietach, z których będziemy korzystać. W LATEX-u dostępnych jest kilka pakietów zawierających w nazwie skrót `ams` (American Mathematical

Society). Określane są one wspólnie jako AMS-LaTeX. Pakiety te rozszerzają możliwości LaTeX-a dotyczące składu wzorów matematycznych. Dalej będziemy zakładać, że pakiety te są dołączone do dokumentu, z którym pracujemy.

– `amsmath` – pakiet stanowi główną część dystrybucji AMS-LaTeX, zawiera środowiska i polecenia, wspomagające skład wzorów matematycznych.
– `amssymb` – pakiet zawiera dużą liczbę poleceń tworzących różnego rodzaju symbole matematyczne (wiele poleceń z tabeli 5.4 wymaga dołączenia tego pakietu do dokumentu).
– `amsfonts` – pakiet udostępnia polecenia `\mathbb{}` i `\mathfrak{}`. Jest ładowany automatycznie przez `amssymb`.

Jako uzupełnienie powyższych pakietów często dołączany jest pakiet `mathtools`, który nie jest częścią oficjalnej dystrybucji AMS-LaTeX, ale dodaje kilka poleceń przydatnych dla składu wzorów matematycznych. W dalszej części pokazany zostanie przykład zastosowania tego pakietu.

Niekiedy kolejność dołączania pakietów może generować informacje o błędach. Wynikają one np. z faktu redefiniowania pewnych nazw. Zgłaszany błąd o redefiniowaniu symboli `\lll` i `\LLL` można usunąć, zmieniając kolejność dołączania pakietów `amssymb` i `babel`.

```
\usepackage{amsmath}
\usepackage{amssymb}
\usepackage{mathtools}
```

Podstawowym środowiskiem do składania pojedynczych eksponowanych wzorów jest **`equation`**. Przeanalizujmy przykład pokazany na listingu 6.1.

Listing 6.1. Przykład użycia obu trybów matematycznych

```
1 Funkcją kwadratową nazywamy funkcję postaci $f(x) = ax^2 + bx + c$,
2 gdzie $a \ne 0$.
3
4 Funkcją kwadratową nazywamy funkcję postaci
5 \begin{equation}
6 \label{eq:funkcja:kwadratowa}
7 f(x) = ax^2 + bx + c,
8 \end{equation}
9 gdzie $a \ne 0$.
```

Funkcją kwadratową nazywamy funkcję postaci $f(x) = ax^2 + bx + c$, gdzie $a \neq 0$.

Funkcją kwadratową nazywamy funkcję postaci

$$f(x) = ax^2 + bx + c, \tag{6.1}$$

gdzie $a \neq 0$.

Na listingu mamy dwa akapity tekstu z tą samą treścią. W pierwszym akapicie wszystkie wzory matematyczne są w trybie matematycznym wewnątrzakapitowym. W drugim akapicie wzór na funkcję kwadratową został umieszczony w środowisku **equation** i jest złożony w trybie eksponowanym. Po prawej stronie w nawiasie okrągłym automatycznie umieszczany jest numer wzoru (w tym przypadku numeracja w ramach rozdziałów). W linii 6 zdefiniowana została również etykieta dla tego wzoru. Jeżeli powołamy się na nią z użyciem polecenia **\ref**, to zostanie zastąpiona tym numerem. Pakiet amsmath zawiera również polecenie **\eqref**{} (parametrem jest etykieta). Polecenie **\eqref** dodatkowo umieszcza numer wzoru w nawiasie okrągłym, tj. wynikiem **\ref**{eq:funkcja:kwadratowa} jest 6.1, a wynikiem **\eqref**{eq:funkcja:kwadratowa} – (6.1). Oczywiście możemy sami dodawać nawiasy przy poleceniu **\ref**, przy czym **\eqref** gwarantuje, że niezależnie od kontekstu nawiasy będą złożone prostą czcionką. Warto jeszcze dodać, że przed środowiskiem **equation** (linia 4) oraz pomiędzy zamknięciem środowiska **equation** (linia 7) a kontynuacją zdania (linia 8) nie powinniśmy zostawiać pustej linii (dla przejrzystości można dodać pusty komentarz). Ponieważ po wzorze powinien zostać wstawiony przecinek, to w przypadku trybu eksponowanego umieszczamy go wewnątrz środowiska **equation**.

Niepoprawne odwołania powstają, m.in. gdy odwołujemy się do nieistniejącej etykiety lub poprawnej etykiety, ale została ona zdefiniowana wielokrotnie. Kompilator informacje o tego typu usterkach przekazuje w formie ostrzeżeń, nie błędów, przez co może nam się wydawać, że wszystko jest w porządku, bo plik pdf jest generowany. Warto czasami przejrzeć log kompilacji, aby się upewnić, że system odwołań w dokumencie jest poprawny.

Jeżeli chcemy uzyskać nienumerowany wzór, należy użyć środowiska **equation***. W takim przypadku nie będzie można poprawnie odwołać się do wzoru. W wyniku kompilacji zamiast numeru wzoru zobaczymy dwa pytajniki – ??. Jest to sygnał, że odwołanie nie jest poprawne.

W przykładzie z listingu 6.1 nie widać różnicy między wzorem funkcji złożonym w trybie wewnątrzakapitowym i eksponowanym. W wielu przypadkach taką różnicę zauważymy. Tak jest na przykład we wzorach z listingu 6.2. Łatwo

zauważyć nie tylko różnice w wielkości czcionki, ale również sposobie rozmieszczenia elementów wzoru. W trybie wewnątrzakapitowym LaTeX stara się, aby wzór jak najmniej odróżniał się pod względem wysokości od reszty tekstu. W trybie eksponowanym nie ma tych ograniczeń. Wróćmy do samej zawartości wzoru. Polecenie **\sum** generuje znak sumy. Granice sumowania umieszczamy za pomocą symboli przeniesienia do indeksu dolnego i górnego w znany nam sposób. Przypominam, że symbole _ i ^ działają tylko na najbliższy znak. Jeżeli potrzebujemy zastosować je do większej liczby znaków, konieczne jest użycie nawiasów klamrowych. Polecenie **\frac**{}{} odpowiada za generowanie ułamków. Pierwszym parametrem jest zawartość licznika, a drugim mianownika.

Listing 6.2. Skład wzorów matematycznych zależnie od trybu

```
1  Sumę częściową dla szeregu geometrycznego wyznaczamy ze wzoru
2  $S_n = \sum_{k=0}^{n}a q^n = a \frac{1-q^{n+1}}{1-q}$, gdzie $a$ jest
3  pierwszym wyrazem szeregu geometrycznego, a $q \ne 1$ -- ilorazem
4  szeregu geometrycznego. Dla $q=1$ mamy $S_n = n a$.
5
6  Sumę częściową dla szeregu geometrycznego wyznaczamy ze wzoru
7  \begin{equation}
8  \label{eq:suma:szeregu}
9  S_n = \sum_{k=0}^{n}a q^n = a \frac{1-q^{n+1}}{1-q},
10 \end{equation}
11 gdzie $a$ jest pierwszym wyrazem szeregu geometrycznego, a $q \ne 1$
12 -- ilorazem szeregu geometrycznego. Dla $q=1$ mamy $S_n = n a$.
```

Sumę częściową dla szeregu geometrycznego wyznaczamy ze wzoru $S_n = \sum_{k=0}^{n} aq^n = a\frac{1-q^{n+1}}{1-q}$, gdzie a jest pierwszym wyrazem szeregu geometrycznego, a $q \neq 1$ – ilorazem szeregu geometrycznego. Dla $q = 1$ mamy $S_n = na$.

Sumę częściową dla szeregu geometrycznego wyznaczamy ze wzoru

$$S_n = \sum_{k=0}^{n} aq^n = a\frac{1-q^{n+1}}{1-q}, \tag{6.2}$$

gdzie a jest pierwszym wyrazem szeregu geometrycznego, a $q \neq 1$ – ilorazem szeregu geometrycznego. Dla $q = 1$ mamy $S_n = na$.

Istnieje możliwość wymuszenia nieco innego zachowania za pomocą poleceń **\limits** i **\nolimits**. Polecenia te umieszczamy bezpośrednio za poleceniem generującym symbol, na który mają one wpłynąć. Polecenie **\limits** umieszczone w linii 2 na listingu 6.3 powoduje, że granice sumowania są umieszczone pod i nad znakiem sumy, mimo że jesteśmy w trybie wewnątrzakapitowym. Z kolei polecenie **\nolimits** (linia 7) powoduje, że granice sumowania są umieszczone na prawo od symbolu sumy. Żadne z tych poleceń nie

zmienia wielkości symbolu sumy. Dalej widać wyraźną różnicę między oboma wzorami.

Listing 6.3. Polecenia modyfikujące ułożenie granic sumowania

```
Sumę częściową dla szeregu geometrycznego wyznaczamy ze wzoru
$S_n = \sum\limits_{k=0}^{n}a q^n = a \frac{1-q^{n+1}}{1-q}$, gdzie
$a$ jest pierwszym wyrazem szeregu geometrycznego, a $q \ne 1$ --
ilorazem szeregu geometrycznego. Dla $q=1$ mamy $S_n = n a$.

Sumę częściową dla szeregu geometrycznego wyznaczamy ze wzoru
\begin{equation}
\label{eq:suma:szeregu:nolimits}
S_n = \sum\nolimits_{k=0}^{n}a q^n = a \frac{1-q^{n+1}}{1-q},
\end{equation}
gdzie $a$ jest pierwszym wyrazem szeregu geometrycznego, a $q \ne 1$
-- ilorazem szeregu geometrycznego. Dla $q=1$ mamy $S_n = n a$.
```

Sumę częściową dla szeregu geometrycznego wyznaczamy ze wzoru $S_n = \sum\limits_{k=0}^{n} aq^n = a\frac{1-q^{n+1}}{1-q}$, gdzie a jest pierwszym wyrazem szeregu geometrycznego, a $q \neq 1$ – ilorazem szeregu geometrycznego. Dla $q = 1$ mamy $S_n = na$.

Sumę częściową dla szeregu geometrycznego wyznaczamy ze wzoru

$$S_n = \sum_{k=0}^{n} aq^n = a\frac{1-q^{n+1}}{1-q}, \tag{6.3}$$

gdzie a jest pierwszym wyrazem szeregu geometrycznego, a $q \neq 1$ – ilorazem szeregu geometrycznego. Dla $q = 1$ mamy $S_n = na$.

Pakiet `amsmath` zawiera polecenie `\sideset{}{}{}`, które umożliwia dodawanie indeksów górnych i dolnych po obu stronach symbolu. Pierwszy parametr określa indeksy po lewej stronie, drugi po prawej, a trzeci indeksowany symbol. Dla przykładu: na podstawie kodu `$\sideset{^a_b}{^c_d}{\sum}$` uzyskamy ${}_b^a\sum_d^c$, zaś kod `$\sideset{^a}{_d}{\sum}$` wygeneruje ${}^a\sum_d$.

6.2 Czcionki w trybie matematycznym

Znaki umieszczone w trybie matematycznym LaTeX traktuje inaczej niż te umieszczone w trybie tekstowym. Po pierwsze w trybie matematycznym ignorowane są niemal wszystkie odstępy i znaki końca linii. LaTeX na podstawie kontekstu decyduje, gdzie i jak duże należy umieścić odstępy. Wewnątrz wzorów nie możemy używać pustych linii – o metodach dzielenia wzoru na wiele linii będę mówić w dalszej części rozdziału. W końcu litery we wzorach domyślnie oznaczają zmienne (symbole) i są składane inaczej niż zwykły tekst.

Tabela 6.1. Polecenia modyfikujące wygląd czcionki w trybie matematycznym

Rodzina/seria/kształt	Polecenie	Efekt
serif	`\mathrm{}`	xyzXYZ
sans serif	`\mathsf{}`	xyzXYZ
monospaced	`\mathtt{}`	xyz$\mathtt{XYZ}$
boldface	`\mathbf{}`	$\mathbf{xyzXYZ}$
normalna	`\mathnormal{}`	$\mathnormal{xyzXYZ}$
kursywa	`\mathit{}`	xyzXYZ
open	`\mathbb{}`	$\mathbb{XYZ}$
kaligraficzna	`\mathcal{}`	$\mathcal{XYZ}$
fraktura	`\mathfrak{}`	$\mathfrak{xyzXYZ}$

W tabeli 6.1 zebrano polecenia modyfikujące wygląd czcionki w trybie matematycznym. Przede wszystkim kluczowe dla poprawnego składu jest polecenie `\mathit`. Zwróć uwagę na różnice w złożeniu tego samego tekstu dla poleceń `\mathnormal` i `\mathit`. W drugim przypadku litery ułożone są obok siebie bez żadnych dodatkowych odstępów. Weźmy na przykład taki wzór `$off \ne on$`: $off \ne on$. Użycie nazw wieloliterowych nie jest niczym niezwykłym, ale efekt, który tu widzimy, jest wręcz koszmarny. Porównajmy to z wersją `$\mathit{off} \ne \mathit{on}$`: $\mathit{off} \ne \mathit{on}$. Pamiętaj, aby zawsze, gdy we wzorze używasz wieloliterowej nazwy, umieszczać ją wewnątrz polecenia `\mathit` (wyjątek stanowią nazwy popularnych funkcji składane czcionką prostą).

Inne ciekawe polecenia to `\mathbb` i `\mathcal`. Ich parametrem powinny być wyłącznie wielkie litery. Pierwsze z nich jest używane m.in. do generowania oznaczeń zbiorów liczbowych, np. `$\mathbb{N}$`, `$\mathbb{R}$` itd. pozwalają uzyskać $\mathbb{N}$, $\mathbb{R}$ itd. Drugiego polecenia używamy zazwyczaj do generowania kaligraficznych oznaczeń, np. `$\mathcal{A}$`, `$\mathcal{B}$` wygenerują $\mathcal{A}$, $\mathcal{B}$ itd.

Polecenia z tabeli 6.1 różnią się istotnie od analogicznych używanych w trybie tekstowym. W trybie tekstowym (zob. podrozdział 2.2) na przykład polecenie `\textbf` zmienia serię czcionki, polecenie `\textit` jej kształt i możemy je nawzajem zagnieżdżać, kumulując obie zmiany. Polecenia z tabeli 6.1 nie zmieniają jednego aspektu czcionki, ale ustawiają specyficzną czcionkę, stąd ich zagnieżdżanie nie ma większego sensu. W efekcie użyta będzie czcionka z ostatniego polecenia, np. `$\mathbf{\mathit{xyz}}$` wygeneruje $\mathbf{\mathit{xyz}}$ – widzimy tylko efekt użycia kursywy.

Jeżeli wewnątrz wzoru matematycznego potrzebujemy przejść na chwilę do trybu tekstowego, to możemy użyć polecenia `\text{}`. Rozważmy taki wzór: `$f(x) = ax^2 + bx + c, \text{ gdzie } a \ne 0$`. W efekcie otrzymamy $f(x) = ax^2 + bx + c$, gdzie $a \ne 0$. Wszystko, co jest parametrem polecenia

`\text`, jest składane w trybie tekstowym (polecenie pochodzi z pakietu `amsmath`, ale założyliśmy, że domyślnie dołączamy pakiety `ams`). Zwróć uwagę, że słowo „gdzie" jest otoczone przez spacje, aby uzyskać odstęp między nim a resztą wzoru. Spacja umieszczona w trybie matematycznym zostanie zignorowana, więc spacje przed poleceniem `\text` i po zamykającym nawiasie klamrowych są pominięte.

6.3 Symbole matematyczne

Wstawianie symboli wymaga wyłącznie odnalezienia polecenia, które odpowiada za interesujący nas znak. Spory wybór poleceń znajduje się w tabeli 5.4. Więcej można znaleźć w pozycji [41]. Przy wyszukiwaniu symboli pożyteczna może być strona `https://detexify.kirelabs.org`. Wystarczy narysować w okienku symbol, którego szukamy, i po chwili dostaniemy listę propozycji. Choć liczba tych poleceń wydaje się przytłaczająca, to zazwyczaj okazuje się, że korzystamy z kilkudziesięciu i po jakimś czasie dobrze pamiętamy ich nazwy. Z drugiej strony środowiska przeznaczone do edycji dokumentów LaTeX-a zawierają zazwyczaj palety najczęściej używanych symboli i wstawienie właściwego polecenia sprowadza się do kliknięcia interesującego nas znaku.

Jednym z problemów, z jakimi możemy się spotkać, jest konieczność umieszczenia dwóch symboli nad sobą, np. spotykamy czasami w książkach oznaczenie w postaci znaku równości z literami „def" umieszczonymi nad nim, co ma symbolizować definiowanie znaczenia jakiegoś pojęcia. Problem taki możemy rozwiązać za pomocą polecenia `\stackrel[]{}{}` z pakietu `stackrel`. Jeżeli ograniczymy się tylko do parametrów obowiązkowych, to polecenie spowoduje umieszczenie pierwszego parametru nad drugim. Parametr umieszczany na górze jest składany tak, jakby znajdował się w indeksie górnym. Parametr podawany jako opcjonalny jest umieszczany pod symbolem głównym (drugim parametrem obowiązkowym). W efekcie możemy uzyskać różne kombinacje pokazane na listingu 6.4. W pakiecie dostępne jest również polecenie `\stackbin` o analogicznej składni, które generuje nieco mniejsze odstępy wokół symboli.

```
\usepackage{stackrel}
```

Listing 6.4. Rozmieszczanie symboli nad sobą z użyciem pakietu `stackrel`

```
1 $X \stackrel{\text{def}}{=} \{x\in \mathbb{N}\colon x^2 < 1000 \}$\\
2 $A \stackrel{\tau}{\longrightarrow} B$\\
```

```
3 $A_i \stackrel[i \ne j]{}{=} A_j$\\
4 $X \stackrel[a]{b}{\approx} Y$
```

$$X \stackrel{\text{def}}{=} \{x \in \mathbb{N}\colon x^2 < 1000\}$$

$$A \xrightarrow{\tau} B$$

$$A_i \underset{i \ne j}{=} A_j$$

$$X \overset{b}{\underset{a}{\approx}} Y$$

W pierwszym wzorze umieściliśmy wspomniane już „def” nad znakiem równości. Zostało ono potraktowane jak zwykły tekst, stąd skład prostą czcionką. We wzorze tym warto zwrócić uwagę na polecenie **\colon**, które generuje dwukropek o znaczeniu „takie, że”. Gdy przyjrzymy się wynikowi kompilacji, to łatwo zauważyć, że odstęp przed dwukropkiem jest wyraźnie mniejszy od odstępu po i w tym kontekście jest to prawidłowe. Zwykły dwukropek zostanie potraktowany jak operator binarny, np. znak plus – odstępy po obu stronach będą takie same, co nie jest tutaj poprawne. W drugim wzorze mamy literę τ umieszczoną nad symbolem strzałki. W trzecim przykładzie użyto parametru opcjonalnego, ale pierwszy parametr obowiązkowy jest pusty. W efekcie mamy tylko wyrażenie $i \ne j$ umieszczone pod znakiem równości. W ostatnim przykładzie skorzystaliśmy z wszystkich dopuszczalnych parametrów, co pozwoliło umieści litery nad i pod symbolem $\approx$.

W tabeli 5.4 znalazła się grupa poleceń generujących symbole zmiennej wielkości. Na listingu 6.2 mieliśmy przykład wykorzystania polecenia **\sum**. Zdarza się, że w indeksie dolnym takiego symbolu trzeba umieścić kilka warunków, tak jak to pokazano na listingu 6.5. Do użytego symbolu iloczynu chcemy dołączyć aż trzy warunki. Skład pokazany we wzorze (6.4) jest zdecydowanie nieelegancki. We wzorze (6.5) (linie 6–10) zastosowano polecenie **\substack{}**, które pozwala łamać tekst i złożyć te warunki w kilku liniach. Jeżeli rozwiązanie to nadal jest niewystarczające (uważamy, że odstęp między znakiem iloczynu i x_i jest za duży), to możemy sięgnąć po polecenie **\mathclap** z pakietu `mathtools`. We wzorze (6.6) (linie 11–15) obudowano nim polecenie **\substack**. W efekcie całkowicie wyeliminowaliśmy przesunięcie w prawo wyrażenia $x_i + y_j$. Oczywiście możliwe jest użycie **\mathclap** bez **\substack**, ale taka korekta wzoru (6.4) nie daje eleganckiego wyniku – warunki pod symbolem iloczynu będą rozciągać się za daleko.

Listing 6.5. Składanie granic w wielu liniach

```
1 \begin{equation}
2 \label{eq:prod:blad}
```

```
\prod_{0 \leqslant i \leqslant m, 0 \leqslant j \leqslant n, i \ne j}
 x_i + y_j
\end{equation}
\begin{equation}
\label{eq:prod:ok}
\prod_{\substack{0 \leqslant i \leqslant m,\\
  0 \leqslant j \leqslant n,\\ i \ne j}}x_i + y_j
\end{equation}
\begin{equation}
\label{eq:prod:lap}
\prod_{\mathclap{\substack{0 \leqslant i \leqslant m,\\
  0 \leqslant j \leqslant n,\\ i \ne j}}}x_i + y_j
\end{equation}
```

$$\prod_{0\leqslant i\leqslant m, 0\leqslant j\leqslant n, i\neq j} x_i + y_j \tag{6.4}$$

$$\prod_{\substack{0\leqslant i\leqslant m,\\ 0\leqslant j\leqslant n,\\ i\neq j}} x_i + y_j \tag{6.5}$$

$$\prod_{\substack{0\leqslant i\leqslant m,\\ 0\leqslant j\leqslant n,\\ i\neq j}} x_i + y_j \tag{6.6}$$

Zwróćmy uwagę na symbole słabej nierówności użyte w powyższych wzorach. Jeżeli składamy tekst w języku polskim, to powinniśmy używać znaków ⩽ i ⩾. Jeżeli składamy tekst w języku angielskim, to właściwe będą symbole ≤ i ≥.

Pakiet `mathtools` [26] oferuje więcej przydatnych poleceń, m.in. **`\mathllap`** i **`\mathrlap`** (we wszystkich trzech wspomnianych poleceniach środkowa litera wskazuje na sposób wyrównania). Polecenia te składają zawartość swoich parametrów tak, jakby nie zajmowały one w ogóle miejsca (fałszowanie rzeczywistości). Dla przykładu: polecenia **`\mathrlap`** można użyć, aby nałożyć na siebie dwa symbole `$\mathrlap{\vee}\wedge$` – ⩓.

Symbol dwukropka nie jest jedynym symbolem, który bardzo często jest niewłaściwie używany we wzorach matematycznych. Porównajmy te trzy próby uzyskania nawiasów ostrokątnych: `$<x,y>$` – $<x,y>$, `$\langle x,y \rangle$` – $\langle x,y \rangle$, `$\left< x,y \right>$` – $\left< x,y \right>$. Pierwsza próba jest zdecydowanie niepoprawna, zarówno kształt, jak i duże odstępy we wzorze są nie do przyjęcia. Taki skład wynika z faktu, że znaki < i > to symbole relacyjne dwuargumentowe i tak są domyślnie postrzegane. W trzecim podejściu występują w innym kontekście, o którym opowiem w dalszej części rozdziału.

W tabeli 6.2 podano przykłady często mylonych symboli. Analizując pary wierszy, możemy zauważyć, że same symbole wyglądają identycznie, ale stosowane są różne odstępy zależnie od ich semantyki. Jeżeli chcemy zapisać różnicę zbiorów A i B, to powinniśmy używać polecenia **\setminus**, jeżeli chcemy zaznaczyć, że płaszczyzny A i B są równoległe, to polecenia **\parallel** itd. Zazwyczaj opis tabeli, w której zebrano symbole, wskazuje na ich przeznaczenie.

Tabela 6.2. Przykłady podobnych wizualnie symboli o różnym przeznaczeniu

Polecenie	Opis	Przykład
\setminus	odejmowanie zbiorów	$A \setminus B$
\backslash	ukośnik wsteczny	$A \backslash B$
\mid	operator dwuargumentowy	$A \mid B$
\vert	pionowa kreska	$A \vert B$
\parallel	symbol relacji	$A \parallel B$
\Vert	podwójna pionowa kreska	$A \Vert B$

LaTeX zawiera polecenia generujące nazwy dla wielu popularnych funkcji, np. kod `$\sin x$` wygeneruje $\sin x$. Jak widzimy, nazwa funkcji sinus jest składana prostą czcionką i wstawiony jest odstęp między nazwą i argumentem. Warto to porównać z tym wzorem `$sin x$` – $sinx$. W tym przypadku wszystkie znaki skleiły się w jeden ciąg. Za poprawne uważa się pierwsze rozwiązanie. Jeżeli jakaś nazwa funkcji nie jest predefiniowana, np. potrzebujemy polskiego skrótu „tg" dla oznaczenia funkcji tangens, to możemy zdefiniować sobie taką nazwę w preambule **\DeclareMathOperator**{**\tg**}{tg}. Teraz wzór `$\tg x$` wygeneruje $\operatorname{tg} x$.

Pakiet `polski` redefiniuje kilka poleceń matematycznych, m.in. dodaje polecenie **\tg**.

6.4 Ograniczniki

Jeżeli konstruujemy bardziej złożone wzory, często zachodzi potrzeba stosowania różnego rodzaju nawiasów (okrągłe, kwadratowe, klamrowe). Są symbole, które wykorzystujemy w podobnym charakterze jak nawiasy, ale mają one specyficzne znaczenie, np. $\lfloor x \rfloor$ oznacza funkcję określaną jako „podłoga", zaś $|x|$ wartość bezwzględną. Wszystkie tego typu symbole nazywamy ogranicznikami. Ich cechą charakterystyczną jest to, że dopasowują swój rozmiar do zawartości (choć ogranicznik może mieć określony rozmiar maksymalny, np. **\langle**

i `\rangle`). Rozszerzające się ograniczniki uzyskujemy za pomocą poleceń `\left` i `\right`, po których bezpośrednio umieszczamy odpowiedni symbol ogranicznika. W tabeli 6.3 pokazano wybór dostępnych ograniczników.

Tabela 6.3. Wybrane ograniczniki

Ogranicznik	Przykład	Ogranicznik	Przykład
`()`	$\left(\sum_{i=0}^{10} i^2\right)$	`[]`	$\left[\sum_{i=0}^{10} i^2\right]$
`\{\}`	$\left\{\sum_{i=0}^{10} i^2\right\}$	`\langle \rangle`	$\left\langle\sum_{i=0}^{10} i^2\right\rangle$
`\lfloor $\rfloor$`	$\left\lfloor\sum_{i=0}^{10} i^2\right\rfloor$	`\lceil \rceil`	$\left\lceil\sum_{i=0}^{10} i^2\right\rceil$
`|` lub `\vert`	$\left\lvert\sum_{i=0}^{10} i^2\right\rvert$	`\\|` lub `\Vert`	$\left\lVert\sum_{i=0}^{10} i^2\right\rVert$
`\uparrow`	$\left\uparrow\sum_{i=0}^{10} i^2\right\uparrow$	`\Uparrow`	$\left\Uparrow\sum_{i=0}^{10} i^2\right\Uparrow$
`\downarrow`	$\left\downarrow\sum_{i=0}^{10} i^2\right\downarrow$	`\Downarrow`	$\left\Downarrow\sum_{i=0}^{10} i^2\right\Downarrow$
`\updownarrow`	$\left\updownarrow\sum_{i=0}^{10} i^2\right\updownarrow$	`\Updownarrow`	$\left\Updownarrow\sum_{i=0}^{10} i^2\right\Updownarrow$
`\lmoustache \rmoustache`	$\left\lmoustache\sum_{i=0}^{10} i^2\right\rmoustache$	`\lgroup \rgroup`	$\left\lgroup\sum_{i=0}^{10} i^2\right\rgroup$

Listing 6.6. Przykłady użycia elastycznych ograniczników

```
\begin{equation}
\label{eq:delimiters}
\left\}\sum_{i = 0}^{10} i^2 \right\{\qquad
\left[\sum_{i = 0}^{10} i^2 \right\updownarrow\qquad
\left\vert\sum_{i = 0}^{10} i^2 \right.\qquad
\left.\sum_{i = 0}^{10} i^2 \right\Vert
\end{equation}
```

$$\left\}\sum_{i=0}^{10} i^2\right\{ \qquad \left[\sum_{i=0}^{10} i^2\right\uparrow \qquad \left\lvert\sum_{i=0}^{10} i^2 \qquad \sum_{i=0}^{10} i^2\right\Vert \tag{6.7}$$

Ograniczniki możemy dowolnie ze sobą mieszać, ale polecenia **\left** i **\right** muszą wystąpić jako para. Możemy też ogranicznika zamykającego użyć w roli otwierającego i na odwrót. Jeżeli po jednej stronie nie potrzebujemy ogranicznika, to zastępujemy go kropką (pusty ogranicznik). Przykład użycia różnych ograniczników pokazano na listingu 6.7. W linii 3 mamy po lewej stronie zamykający, a po prawej otwierający nawias klamrowy (należy pamiętać o znakach ukośnika). W linii 4 mamy różne ograniczniki po prawej i lewej stronie. W kolejnych dwóch liniach użyto jednego pustego ogranicznika. Polecenie **\qquad** użyte w przykładzie generuje poziomy podwójny odstęp. Dostępne jest również polecenie **\quad** generujące pojedynczy odstęp. Można je stosować zarówno w trybie tekstowym, jak i matematycznym. Więcej o odstępach w trybie matematycznym powiem w podrozdziale 6.8.

Wszystkie ograniczniki z tabeli 6.3, z wyjątkiem ostatniego wiersza, funkcjonują jako samodzielne polecenia (znaki). Jeżeli nie potrzebujemy elastycznego ogranicznika, to możemy użyć ich bez poleceń **\left** i **\right**, np. `$\lceil x \rceil$` wygeneruje $\lceil x \rceil$ (wcześniej mieliśmy też przykład z nawiasem ostrokątnym). Warto dodać, że w dokumencie [41] możemy znaleźć informacje o kilku innych pakietach dostarczających dodatkowe elastyczne ograniczniki, np. podwojone lub potrojone nawiasy kwadratowe, ostrokątne itp.

Jeżeli nie jesteśmy zadowoleni z automatycznego dopasowania wielkości ograniczników, możemy zamiast poleceń **\left** i **\right** użyć poleceń z tabeli 6.4, za pomocą których na sztywno ustawiamy wielkość danego ogranicznika. Cechą charakterystyczną tych poleceń jest to, że nie muszą występować w parach. Daje to możliwość, przy łamaniu wzoru na wiele linii (zob. podrozdział 6.7), wstawienia ogranicznika otwierającego w jednej linii, a zamykającego w innej. Polecenia współpracują z dowolnym ogranicznikiem z tabeli 6.3. Przykład użycia takich ograniczników można znaleźć w listingu 6.18.

Tabela 6.4. Polecenia do ustawiania wielkości ograniczników

Polecenie	Przykład	Polecenie	Przykład
\bigl	$\bigl[$	**\bigr**	$\bigr]$
\Bigl	$\Bigl[$	**\Bigr**	$\Bigr]$
\biggl	$\biggl[$	**\biggr**	$\biggr]$
\Biggl	$\Biggl[$	**\Biggr**	$\Biggr]$

6.5 Struktury tabelaryczne we wzorach

Odpowiednikiem środowiska **tabular** w trybie matematycznym jest środowisko **array**. Podstawowa zasada jego stosowania pozostaje taka jak dla pierwszego z tych środowisk – każdy wiersz kończymy poleceniem \\, komórki w wierszu oddzielamy od siebie znakiem & i dbamy, aby każdy wiersz miał tyle samo komórek. Środowiska **array** używamy wyłącznie w trybie matematycznym. Zawartość każdej komórki jest składana w trybie matematycznym bez potrzeby używania np. znaków $. Środowisko **array** nie służy wyłącznie do składania macierzy. Można go użyć przy definiowaniu przez przypadki, składaniu układów równań itp.

Listing 6.7. Macierz uzupełniona układu równań

```
1  \begin{equation}
2  \label{eq:macierz:uzupelniona}
3  U = \left[ \begin{array}{cc} A & b \end{array} \right] =
4  \left[
5  \begin{array}{lllll}
6  a_{11} & a_{12} & \dots & a_{1n} & b_1\\
7  a_{21} & a_{22} & \dots & a_{2n} & b_2\\
8  \vdots & \vdots & \ddots & \vdots & \vdots\\
9  a_{m1} & a_{m2} & \dots & a_{mn} & b_m\\
10 \end{array}
11 \right]
12 \end{equation}
```

$$U = \left[\begin{array}{cc} A & b \end{array} \right] = \left[\begin{array}{lllll} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} & b_m \end{array} \right] \tag{6.8}$$

Zaczniemy jednak od najbardziej oczywistego zastosowania. Na listingu 6.7 przedstawiono tzw. macierz uzupełnioną układu równań liniowych. Zacznijmy od analizy dużej macierzy (linie 4–11). Jako ograniczniki zastosowano nawiasy kwadratowe (linie 4 i 11). Parametr obowiązkowy środowiska **array** (linia 5) wskazuje, że definiujemy 5 kolumn i każdą z zawartością wyrównaną do lewej strony. Jeżeli chodzi o zawartość komórek, to warto zwrócić uwagę na różne typy wielokropków użyte w macierzy.

Część reprezentowana przez współczynniki a_{ij} jest nazywana macierzą główną i oznaczana przez A, zaś ostatnia kolumna to kolumna wyrazów wolnych oznaczana przez b. Zostało to wyrażone w środkowej części wzoru, która też wykorzystuje środowisko **array** (linia 3). Zabieg ten pozwolił nam uzyskać

odstępy między A i b analogiczne jak w dużej macierzy. Dla porównania zapis `$[A\;b]$` wygeneruje nam $[A\;b]$ – polecenie `\;` wymusza dodanie odstępu między literami. Wersja ze środowiskiem **array** prezentuje się lepiej.

Analogicznie jak w środowisku **tabular** możemy wskazać za pomocą znaku `|`, gdzie chcemy dodać pionowe linie, lub użyć poleceń **\hline** i **\cline** do dodania linii poziomych. Możemy również stosować polecenia **\multicolumn** i **\multirow** do scalania komórek.

Jedną z metod rozwiązywania układów typu (6.8) jest wykonanie przekształceń elementarnych i sprowadzenie macierzy U do jednej z postaci bazowych, jak ta z listingu 6.8 (pełny opis metody można znaleźć w [44]). W linii 6 użyliśmy polecenia **\multicolumn**, aby scalić dwie komórki.

Listing 6.8. Postać bazowa

```
1 \begin{equation}
2 \label{eq:postac:bazowa4}
3 \left[
4 \begin{array}{ccc}
5 E & M & c_1\\
6 \multicolumn{2}{c}{0} & c_2
7 \end{array}
8 \right]
9 \end{equation}
```

$$\left[\begin{array}{ccc} E & M & c_1 \\ \multicolumn{2}{c}{0} & c_2 \end{array}\right] \tag{6.9}$$

Na listingu 6.9 mamy przykład macierzy bazowej zapożyczony z książki [44]. W tym przykładzie wykorzystano linie, aby wyróżnić poszczególne podmacierze zgodnie ze wzorem (6.9). Warto zwrócić uwagę na polecenie **\multicolumn** z linii 8, którego użyto, aby wyeliminować jedną z pionowych linii w ostatnim wierszu.

Listing 6.9. Postać bazowa – przykład

```
1  \begin{equation}
2  \label{eq:macierz:uzupelniona:przyklad}
3  U = \left[
4  \begin{array}{rrr|rr|r}
5  1 & 0 & 0 &  0 & -1 &  0\\
6  0 & 1 & 0 &  0 & -1 &  0\\
7  0 & 0 & 1 &  -3 & 0 & 0\\\hline
8  0 & 0 & \multicolumn{1}{r}{0} &  0 & 0 & 0\\
9  \end{array}
10 \right]
11 \end{equation}
```

$$U = \left[\begin{array}{ccc|cc|c} 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & -3 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \tag{6.10}$$

Środowisko **array** zachowuje się pod wieloma względami analogicznie jak **tabular**. Możemy użyć opisanych w rozdziale 5 metod zmiany tła dla wierszy i kolumn. Możemy stosować polecenia **\makecell**, **\gape**, **\Gape** oraz konstrukcję @{} do indywidualnej modyfikacji odstępu międzykolumnowego. Jeżeli chcemy ustawić jednocześnie wszystkie odstępy, to trzeba zmodyfikować polecenie **\arraycolsep**, np. **\setlength**{**\arraycolsep**}{4pt}.

Środowiska **array** możemy użyć do definiowania przez przypadki, jak pokazano na listingu 6.10, lub składania układów równań – listing 6.11. W tym drugim przypadku rola środowiska **array** ogranicza się w zasadzie wyłącznie do łamania linii wewnątrz ogranicznika.

Listing 6.10. Definiowanie funkcji przez przypadki

```
\begin{equation}
\label{eq:funkcjaf}
f(x) = \left\{
\begin{array}{rcl}
-x^2 & \text{dla} & x \leqslant 0,\\
\sqrt{x} + \sin x & \text{dla} & x > 0.
\end{array}
\right.
\end{equation}
```

$$f(x) = \begin{cases} -x^2 & \text{dla} \quad x \leqslant 0, \\ \sqrt{x} + \sin x & \text{dla} \quad x > 0. \end{cases} \tag{6.11}$$

Listing 6.11. Układ równań

```
\begin{equation}
\label{eq:uklad:rownan}
\left\{
\begin{array}{l}
x + y = 8\\
3x - y = 0
\end{array}
\right.
\end{equation}
```

$$\begin{cases} x + y = 8 \\ 3x - y = 0 \end{cases} \tag{6.12}$$

6.6 Serie wzorów

Jeżeli potrzebujemy złożyć kilka wzorów z trybie eksponowanym i mają być one położone bezpośrednio po sobie, nie powinniśmy do tego używać kolejnych środowisk **equation**. Powodem są zbyt duże odstępy, które powstają wówczas między kolejnymi wzorami. Sytuację taką możemy zaobserwować w przypadku wzorów z listingu 6.5. Możesz porównać pionowe odstępy w tamtych wzorach i przykładach prezentowanych w tym podrozdziale. LaTeX oferuje kilka środowisk, które pozwalają składać serie wzorów. Jednym z lepszych wyborów jest środowisko **align** lub jego nienumerowana wersja **align***. Pozwala ono nie tylko złożyć serię wzorów, ale również wyrównać je w pionie do lewej lub do prawej strony. Zacznijmy od przykładu, gdy umieszczamy jeden wzór w jednej linii – listing 6.12.

Listing 6.12. Seria wzorów złożona w środowisku **align**

```
1 \begin{align}
2 \left| z \right| = 0 \iff z = 0,\label{eq:zespolone1}\\
3 \left| z \right| \geqslant 0,\label{eq:zespolone2}\\
4 \left| \frac{z_1}{z_2} \right| =
5   \frac{\left| z_1 \right|}{\left| z_2 \right|},\label{eq:zespolone3}
6 \end{align}
```

$$|z| = 0 \iff z = 0, \tag{6.13}$$

$$|z| \geqslant 0, \tag{6.14}$$

$$\left|\frac{z_1}{z_2}\right| = \frac{|z_1|}{|z_2|}, \tag{6.15}$$

W najprostszej wersji mamy po kompilacji jeden wzór umieszczony w jednej linii (wzór może zajmować kilka linii kodu źródłowego), wiersze łamiemy za pomocą polecenia \\, każdy wzór jest numerowany i wzory są wyrównane do lewej strony. Ponieważ środowisko **align** obejmuje wiele wzorów, to etykiety dla poszczególnych wzorów umieszczamy tuż za ich kodem źródłowym.

W bardziej ogólnym przypadku jeden wiersz może obejmować wiele wzorów (części wzorów). Wówczas układ wewnątrz środowiska przypomina tabelę (komórki oddzielane znakiem &, wiersze łamane poleceniem \\), ale mamy domyślnie przypisany sposób wyrównania kolumn – kolumny nieparzyste są wyrównane do prawej, a parzyste do lewej strony (w przykładzie powyżej mieliśmy tylko jedną kolumnę). Numery równań są przypisywane do kolejnych wierszy.

W przykładzie z listingu 6.12 możemy łatwo uzyskać wyrównanie do prawej strony – wystarczy wstawić pustą pierwszą kolumnę, czyli dodać znaki & na początku linii 2–4. Żadne z tych rozwiązań nie oferuje wersji, w której seria równań

pozostaje wyśrodkowana. Jeżeli chcemy uzyskać taki wynik, to zamiast środowiska **align** można użyć **gather** (nie używamy znaków &, tylko łamiemy wiersze poleceniem \\) lub **gather*** dla serii wzorów nienumerowanych. Przykład złożenia powyższych wzorów z użyciem środowiska **gather** widzimy we wzorach (6.16)–(6.18). W porównaniu z kodem z listingu 6.12 zmieniono wyłącznie nazwę środowiska.

$$|z| = 0 \iff z = 0, \tag{6.16}$$

$$|z| \geqslant 0, \tag{6.17}$$

$$\left|\frac{z_1}{z_2}\right| = \frac{|z_1|}{|z_2|}, \tag{6.18}$$

Nieco inaczej wygląda wyrównanie w przykładzie z listingu 6.13. Tutaj korzystamy z dwóch kolumn, ale znaki & umieszczono tuż przed znakiem =. W efekcie otrzymujemy wyrównanie wzorów według tego znaku – wszystkie znaki równości są w pionie ustawione wzdłuż jednej linii.

Listing 6.13. Wyrównanie serii równań według znaku równości

```
\begin{align}
\lim_{x \to c} \left( f(x)\pm g(x)\right)
&= \lim_{x \to c}f(x)\pm \lim_{x \to c}g(x) ,\label{eq:granice1}\\
\lim_{x \to c} \left( f(x)\cdot g(x)\right)
&= \lim_{x \to c}f(x)\cdot \lim_{x \to c}g(x) ,\label{eq:granice2}\\
\lim_{x \to c} \frac{f(x)}{g(x)}
&= \frac{\lim_{x \to c}f(x)}{\lim_{x \to c}g(x)},
\text{\quad gdzie } \lim_{x \to c}g(x) \ne 0\label{eq:granice3}
\end{align}
```

$$\lim_{x\to c}\big(f(x)\pm g(x)\big) = \lim_{x\to c} f(x) \pm \lim_{x\to c} g(x), \tag{6.19}$$

$$\lim_{x\to c}\big(f(x)\cdot g(x)\big) = \lim_{x\to c} f(x) \cdot \lim_{x\to c} g(x), \tag{6.20}$$

$$\lim_{x\to c}\frac{f(x)}{g(x)} = \frac{\lim_{x\to c} f(x)}{\lim_{x\to c} g(x)}, \quad \text{gdzie } \lim_{x\to c} g(x) \neq 0 \tag{6.21}$$

W kolejnym przykładzie (listing 6.14) również wykorzystaliśmy efekt wyrównania w pionie względem znaków równości. W tym przypadku umieszczono cztery wzory w jednej linii (używamy 8 kolumn). Środowisko automatycznie rozmieszcza te wzory, wykorzystując dostępną szerokość strony. Użyliśmy środowiska bez numerowania linii, bo mamy po kilka wzorów w jednym wierszu.

Na zakończenie tej serii przykładów warto podkreślić, że w środowisku **align** nie deklarujemy, z ilu kolumn będziemy korzystać. Mało tego, w różnych liniach możemy użyć różnej liczby kolumn i nie wygeneruje to błędu kompilacji, np.

możemy usunąć dwa ostatnie wzory na pochodną (i odpowiednie znaki &) i nie będzie to stanowiło żadnego problemu.

Listing 6.14. Złożenie wzorów w wielu kolumnach

```
\begin{align*}
\left(C\right)' &=0 & \left(x^{\alpha}\right)' &= \alpha x^{\alpha-1}
& \left(\sin\right)' &=\cos & \left(\cos\right)' &= -\sin\\
\left(a^{x}\right)' &= a^{x}\ln a & \left(e^{x}\right)' &= e^{x} &
\left(\log_{a}x\right)' &= \frac{1}{x\ln a} & \left(\ln x\right)'
  &= \frac{1}{x}
\end{align*}
```

$$(C)' = 0 \qquad \left(x^{\alpha}\right)' = \alpha x^{\alpha-1} \qquad (\sin)' = \cos \qquad (\cos)' = -\sin$$
$$\left(a^{x}\right)' = a^{x}\ln a \qquad \left(e^{x}\right)' = e^{x} \qquad \left(\log_{a} x\right)' = \frac{1}{x\ln a} \qquad (\ln x)' = \frac{1}{x}$$

Na zakończenie tego podrozdziału skupimy się na kwestii odwoływania się do numerowanych wzorów, które jako seria stanowią pewną całość. Rozważmy przykład przedstawiony na listingu 6.15. Trzy równania zamieszczone w środowisku **align** stanowią definicję prostego modelu w algebrze procesów CCS. Opisując system, chcielibyśmy się odwoływać do wszystkich trzech równań jako całości, ale chcemy też mieć możliwość indywidualnego odniesienia się do poszczególnych równań. Problem możemy rozwiązać za pomocą środowiska **subequations**, które jako opakowanie dla środowiska **align** zmienia sposób numerowania równań. Wszystkie równania współdzielą numer, a różnią się dodaną do tego numeru literą.

Listing 6.15. Przykład zastosowania środowiska **subequations**

```
\begin{subequations}
\label{eq:css:system}
\begin{align}
A &= a.b.c.A \label{eq:agent:a} \\
B &= \bar{a}.B + \bar{b}.B \label{eq:agent:b} \\
C &= (A\mid B) \backslash \{a,b\} \label{eq:agent:c}
\end{align}
\end{subequations}
```

$$A = a.b.c.A \tag{6.22a}$$
$$B = \bar{a}.B + \bar{b}.B \tag{6.22b}$$
$$C = (A \mid B)\backslash\{a, b\} \tag{6.22c}$$

Zwróćmy uwagę, w jaki sposób zdefiniowane zostały etykiety w naszym przykładzie. Na końcu linii 4–6 mamy etykiety poszczególnych równań. Etykieta w linii 2 jest etykietą dla całego zestawu równań. Jeżeli teraz użyjemy odwołania `\eqref{eq:css:system}`, to otrzymamy po kompilacji (6.22), zaś w przypadku `\eqref{eq:agent:b}` dostaniemy (6.22b). Dla kompletu dodajmy, że polecenie `\bar` odpowiada za dodanie małej poziomej kreski nad swoim parametrem. Kreska nie jest elastyczna, więc parametrem powinien być raczej jeden znak.

Środowisko `align` automatycznie numeruje wszystkie wiersze. Jeżeli chcemy wyłączyć numerowanie danego wiersza, można umieścić w nim polecenie `\notag` (lub `\nonumber`, jeżeli nie używamy pakietów `ams`). W wierszu takim nie definiujemy etykiety. Z drugiej strony możemy użyć polecenia `\tag{}`, aby nadpisać domyślną etykietę – parametr polecenia będzie nową etykietą umieszczoną w nawiasach okrągłych. Pakiet `amsmath` zawiera również polecenie `\tag*{}`, które nie umieszcza nowej etykiety w nawiasach okrągłych.

Listing 6.16. Przedstawienie zmienionych definicji systemu (6.22)

```
1 \begin{align}
2 A &= a.b.c.A \tag*{powtórzenie \eqref{eq:agent:a}} \\
3 B &= \bar{a}.\bar{b}.B\label{eq:agent:b2}\tag{\ref{eq:agent:b}'} \\
4 C &= (A\mid B)\backslash\{a\}\label{eq:agent:c2}\tag{\ref{eq:agent:c}'}
5 \end{align}
```

$$A = a.b.c.A \qquad \text{powtórzenie (6.22a)}$$
$$B = \bar{a}.\bar{b}.B \qquad \text{(6.22b')}$$
$$C = (A \mid B)\backslash\{a\} \qquad \text{(6.22c')}$$

Rozważmy przykład przedstawiony na listingu 6.16. W pierwszym wierszu jedynie powtarzamy równanie (6.22a), co jest zaznaczone za pomocą polecenia `\tag*`. Kolejne równania przedstawiają zmienione definicje agentów B i C, które chcemy oznaczyć takim samym numerem jak poprzednio, ale uzupełnionym znakiem prim. Tutaj używamy polecenia `\tag`, ale nową etykietę definiujemy z użyciem polecenia `\ref`, aby powiązać ją z oryginalnym równaniem (nie używamy `\eqref`, bo nie chcemy dodatkowych nawiasów). Jeżeli teraz użyjemy odwołania `\eqref{eq:agent:c2}`, to dostaniemy (6.22c').

6.7 Wzory wieloliniowe

Środowiska matematyczne omawiane do tej pory nie pozwalają na łamanie wiersza, tzn. nie składaliśmy jednego wzoru w wielu liniach. Jeżeli jednak opisujemy na przykład rozwiązywanie zadań matematycznych, gdzie wyrażenie

przekształcane jest w wielu krokach, konieczne może być złożenie wzoru w więcej niż jednej linii. Możliwości takie daje środowisko **split**, którego używamy pomocniczo wewnątrz środowiska matematycznego.

Listing 6.17. Składanie wzoru matematycznego w wielu liniach

```
\begin{equation}
\label{eq:calka}
\begin{split}
\int x^2 e^x\,\mathrm{d}x & = x^2 e^x - 2 \int x e^x\,\mathrm{d}x = \\
& = x^2 e^x - 2 \left( x e^x - \int e^x\,\mathrm{d}x \right) = \\
& = x^2 e^x -2x e^x + 2 e^x + C
\end{split}
\end{equation}
```

$$\begin{aligned}\int x^2 e^x \,\mathrm{d}x &= x^2 e^x - 2\int x e^x \,\mathrm{d}x = \\ &= x^2 e^x - 2\left(x e^x - \int e^x \,\mathrm{d}x\right) = \\ &= x^2 e^x - 2x e^x + 2e^x + C\end{aligned} \tag{6.23}$$

Przykład użycia środowiska **split** pokazano na listingu 6.17. Każda linia zawiera symbol &, który wskazuje punkt wyrównania do lewej występujących po nim fragmentów linii. W tym przykładzie mamy wyrównanie w pionie według pierwszych znaków równości. W efekcie łatwo zauważyć, że kolejne linie stanowią kontynuację obliczeń. Zgodnie z konwencją typograficzną w matematyce, operator różniczkowy d składamy prostą czcionką z odstępem `\,` przed nim.

Jako drugi przykład przeanalizujemy wzór (6.24), który nie tylko został podzielony na wiele linii, ale wzbogacony również o objaśnienia tekstowe. Dodatkowo w kodzie wykorzystano ciekawe przykłady ograniczników, m.in. z koniecznością wymuszenia ich wielkości. Przykład pochodzi z książki [33]. Kod źródłowy dla tego przykładu znajduje się na listingu 6.18.

Listing 6.18. Wzór wieloliniowy ze wstawkami tekstowymi

```
\begin{equation}
\label{eq:calka:potrojna}
\begin{split}
\iiint\limits_V x\,\mathrm{d}v & = \iint\limits_D\left(x
\int\limits_{0}^{2-2x-2y}\,\mathrm{d}z \right)\,\mathrm{d}\sigma =
\iint\limits_D x \left(2-2x-2y \right)\,\mathrm{d}\sigma =\\
\intertext{określamy granice całkowania i obliczamy całkę względem $y$}
& = \iint\limits_D \left(2x-2x^2-2xy \right)\,\mathrm{d}\sigma =
\int\limits_0^1 \left(\int\limits_0^{1-x}(2x-2x^2-2xy)\,\mathrm{d}y
\right)\,\mathrm{d}x =\\
```

```
11 & = \int\limits_0^1 \biggl[2xy-2x^2y-xy^2 \biggr]_{y=0}^{y=1-x}\,
12 \mathrm{d}x =\\
13 & = \int\limits_0^1 \left(2x(1-x)-2x^2(1-x)-x(1-x)^2\right)\,
14 \mathrm{d}x =\\
15 \intertext{w końcu obliczamy całkę względem $x$}
16 & = \int\limits_0^1(x^3 -2x^2 +x)\,\mathrm{d}x =
17 \left[\frac{1}{4}x^4 -\frac{2}{3}x^3 +\frac{1}{2}x^2 \right]_0^1 =
18 \frac{1}{4} -\frac{2}{3} +\frac{1}{2} = \frac{1}{12}
19 \end{split}
20 \end{equation}
```

$$\iiint\limits_V x\,\mathrm{d}v = \iint\limits_D \left(x \int\limits_0^{2-2x-2y} \mathrm{d}z \right) \mathrm{d}\sigma = \iint\limits_D x\left(2-2x-2y\right)\mathrm{d}\sigma =$$

określamy granice całkowania i obliczamy całkę względem y

$$\begin{aligned} &= \iint\limits_D \left(2x-2x^2-2xy\right)\mathrm{d}\sigma = \int\limits_0^1 \left(\int\limits_0^{1-x} (2x-2x^2-2xy)\,\mathrm{d}y \right) \mathrm{d}x = \\ &= \int\limits_0^1 \left[2xy-2x^2y-xy^2\right]_{y=0}^{y=1-x} \mathrm{d}x = \\ &= \int\limits_0^1 \left(2x(1-x)-2x^2(1-x)-x(1-x)^2\right)\mathrm{d}x = \end{aligned} \tag{6.24}$$

w końcu obliczamy całkę względem x

$$= \int\limits_0^1 (x^3-2x^2+x)\,\mathrm{d}x = \left[\frac{1}{4}x^4-\frac{2}{3}x^3+\frac{1}{2}x^2\right]_0^1 = \frac{1}{4}-\frac{2}{3}+\frac{1}{2} = \frac{1}{12}$$

Symbole całek użyte w analizowanym przykładzie wstawiamy poleceniami **\iiint**, **\iint** i **\int**, które generują symbole odpowiednio całki potrójnej, podwójnej i pojedynczej (więcej poleceń do generowania symboli całek można znaleźć w [41]). We wszystkich całkach stosowane jest polecenie **\limits**, aby wymusić ustawianie granic całkowania pod i nad symbolem całki. Środowisko **align** domyślnie przesunie je na prawo od symbolu. W liniach 7 i 12 użyto polecenia **\intertext**. Polecenie powoduje wstrzymanie składania wzoru matematycznego i pozwala umieścić akapit tekstu, zazwyczaj w roli objaśnienia wykonywanych przekształceń. W linii 10 skorzystaliśmy z ustawienia wielkości nawiasów kwadratowych za pomocą poleceń **\biggl** i **\biggr**. Zawartość nawiasu

nie jest wysoka, więc automatycznie dobrany nawias był wyraźnie mniejszy od tego użytego w linii 14 (w tej linii wysoki nawias kwadratowy wymuszają ułamki). Jednocześnie przy obu nawiasach kwadratowych mamy w indeksach granice obliczeń. Dla zachowania spójnego wyglądu wzoru zostały wymuszone większe nawiasy w kodzie z linii 10.

6.8 Ozdobniki i odstępy

Z kilkoma ozdobnikami spotkaliśmy się już w niektórych przykładach w tym rozdziale, na przykład za pomocą znaku prim możemy wygenerować symbol pochodnej, a polecenie **\bar** doda małą kreskę nad swoim parametrem. Przypominają one swoim funkcjonowaniem znaki diakrytyczne. Najczęściej używane ozdobniki przedstawiono w tabeli 6.5.

Tabela 6.5. Wybrane ozdobniki dostępne w trybie matematycznym

Polecenie	Efekt użycia
\dot{}, **\ddot**{}, **\dddot**{}, **\ddddot**{}	$\dot{x}$, $\ddot{x}$, $\dddot{x}$, $\ddddot{x}$,
\bar{}	$\bar{x}$, $x\bar{y}z$
\hat{}	$\hat{x}$, $x\hat{y}z$
\tilde{}	$\tilde{x}$, $x\tilde{y}z$
\vec{}	$\vec{x}$, $x\vec{y}z$
\widehat{}	$\widehat{x}$, $\widehat{xyz}$
\widetilde{}	$\widetilde{x}$, $\widetilde{xyz}$
\overline{}	$\overline{x}$, $\overline{xyz}$
\underline{}	$\underline{x}$, $\underline{xyz}$
\overrightarrow{}	$\overrightarrow{x}$, $\overrightarrow{xyz}$
\underrightarrow{}	$\underrightarrow{x}$, $\underrightarrow{xyz}$
\overleftarrow{}	$\overleftarrow{x}$, $\overleftarrow{xyz}$
\underleftarrow{}	$\underleftarrow{x}$, $\underleftarrow{xyz}$
\overleftrightarrow{}	$\overleftrightarrow{x}$, $\overleftrightarrow{xyz}$
\underleftrightarrow{}	$\underleftrightarrow{x}$, $\underleftrightarrow{xyz}$

W przykładach (druga kolumna) pokazano ich zachowanie, gdy parametrem jest pojedynczy znak i ciąg znaków. Warto zwrócić uwagę, że długość znaków **\widehat** i **\widetilde** jest ograniczona, i jeśli parametrem będzie dłuższe wyrażenie, to znaki te nie będą rozciągać się nad całością. Ostatnich sześć poleceń

dostosowuje swoją długość do parametru. Pozwala to m.in. uzyskać dowolnie długie strzałki, jako alternatywę dla polecenia **\longrightarrow** z listingu 6.4. Przykład pokazano na listingu 6.19. Wyrażenie z poleceniem **\underrightarrow** jest przeniesione do indeksu górnego, w wyniku czego strzałka znalazła się na odpowiedniej wysokości.

Listing 6.19. Generowanie bardzo długich strzałek

```
$A^{\underrightarrow{\qquad\tau\qquad}} B$\\
$A^{\underrightarrow{f(x) + f'(x) + f''(x)}} B$
```

$$A \xrightarrow{\qquad\tau\qquad} B$$
$$A \xrightarrow{f(x)+f'(x)+f''(x)} B$$

Jeżeli potrzebujemy umieścić ozdobnik, np. symbol wektora nad literami i oraz j, to powinniśmy użyć poleceń generujących te znaki bez kropek. Są to odpowiednio **\imath** i **\jmath**. Na przykład polecenie `$\vec{\imath}$` wygeneruje $\vec{\imath}$. Wydaje się, że usuwanie kropki nad i przy dodawaniu symbolu wektora jest przesadą, ale dla porównania przeanalizujmy taki przypadek: `$\ddot{i}$` daje w efekcie $\ddot{i}$, zaś `$\ddot{\imath}$` – $\ddot{\imath}$.

Inny typ ozdobników zapewniają polecenia **\overbrace**{} i **\underbrace**{} generujące elastyczne nawiasy klamrowe. W pierwszym przypadku w indeksie górnym możemy podać opis, który znajdzie się nad klamrą. W drugim przypadku opis do umieszczenia pod klamrą podajemy w indeksie dolnym. Przykłady użycia tych poleceń pokazano na listingach 6.20 i 6.21.

Listing 6.20. Przykład zastosowania polecenia **\underbrace**

```
1 \begin{equation}
2 \label{eq:mnozenie}
3 \underbrace{a + a + \ldots + a}_{n}  = n \cdot a
4 \end{equation}
```

$$\underbrace{a+a+\ldots+a}_{n} = n \cdot a \tag{6.25}$$

Listing 6.21. Przykład zastosowania poleceń **\overbrace** i **\underbrace**

```
1 \begin{equation}
2 \label{eq:numu}
3 \overbrace{\neg a,a,a,\neg a,a,a,a,\neg a,\neg a,a,a,\neg a,
4 \underbrace{a,a,a,a,}_{\text{skończona liczba }a} \neg a,\dots}^
5 {\text{łączna liczba akcji } a \text{ może być nieskończona}}
6 \end{equation}
```

$$\overbrace{\neg a, a, a, \neg a, a, a, a, \neg a, \neg a, a, a, \neg a, \underbrace{a, a, a, a,}_{\text{skończona liczba } a} \neg a, \ldots}^{\text{łączna liczba akcji } a \text{ może być nieskończona}} \tag{6.26}$$

W drugim przypadku mamy zagnieżdżenie polecenia **\underbrace** wewnątrz parametru polecenia **\overbrace**. Jak widać, domyślnie przestrzeń zajmowana przez te polecenia zależy od tego, co jest dłuższe – treść parametru czy treść w indeksie. W równaniu (6.26) dłuższa treść w indeksie dolnym psuje skład. Rozwiązaniem jest opakowanie zawartości indeksu poleceniem **\mathclap**.

Listing 6.22. Korekta składu z użyciem polecenia **\mathclap**

```
\begin{equation}
\label{eq:numu2}
\overbrace{\neg a,a,a,\neg a,a,a,a,\neg a,\neg a,a,a,\neg a,
\underbrace{a,a,a,a,}_{\mathclap{\text{skończona liczba }a}} \neg a,
\dots}^{\text{łączna liczba akcji } a \text{ może być nieskończona}}
\end{equation}
```

$$\overbrace{\neg a, a, a, \neg a, a, a, a, \neg a, \neg a, a, a, \neg a, \underbrace{a, a, a, a,}_{\text{skończona liczba } a} \neg a, \ldots}^{\text{łączna liczba akcji } a \text{ może być nieskończona}} \tag{6.27}$$

Bliźniacze polecenie **\mathrlap** można zastosować, aby uzyskać ciekawy efekt częściowego nakładania się nawiasów klamrowych. Pomocniczo zastosujemy również polecenie **\phantom**{}. Polecenie to tworzy pudełko o rozmiarach dopasowanych do treści swojego parametru, ale nie wypełnia go treścią, np. we wzorze `$a + b + \phantom{c + d} = 0$` dostajemy puste miejsce, jakie zajmowałoby złożone wyrażenie $c + d$, tj. $a + b +$ $= 0$. Rozwiązanie przedstawimy etapami, aby dobrze zilustrować znaczenie poszczególnych poleceń.

Listing 6.23. Nakładające się nawiasy klamrowe – etap 1

```
\begin{equation}
\label{eq:rownowaznosc1}
p \Longleftrightarrow \overbrace{q \Longleftrightarrow r}^
{\mathclap{q \Rightarrow r \wedge r \Rightarrow q}}
\end{equation}
```

$$p \Longleftrightarrow \overbrace{q \Longleftrightarrow r}^{q \Rightarrow r \wedge r \Rightarrow q} \tag{6.28}$$

Zacznijmy od kodu przedstawionego na listingu 6.23 i wyniku kompilacji, który widzimy jako wzór (6.28). W tym kodzie nie mamy niczego nowego. Ponownie skorzystaliśmy z polecenia **\mathclap**, gdyż treść w indeksie górnym jest dłuższa niż wyrażenie pod nawiasem klamrowym. Naszym celem jest dodanie

drugiej klamry, która od dołu będzie obejmowała wyrażenie $p \Longleftrightarrow q$. Nie możemy tego zrobić bezpośrednio, stosując polecenie **\underbrace**, bo ze względu na częściowe nakładanie się klamer nie można poprawnie (z perspektywy kompilatora LaTeX-a) zagnieździć obu poleceń generujących klamry.

Listing 6.24. Nakładające się nawiasy klamrowe – etap 2

```
1 \begin{equation}
2 \label{eq:rownowaznosc2}
3 \underbrace{\phantom{p \Longleftrightarrow q}}_
4 {\mathclap{p \Rightarrow q \wedge q \Rightarrow p}}
5 p \Longleftrightarrow \overbrace{q \Longleftrightarrow r}^
6 {\mathclap{q \Rightarrow r \wedge r \Rightarrow q}}
7 \end{equation}
```

$$\underbrace{}_{p \Rightarrow q \wedge q \Rightarrow p} p \Longleftrightarrow \overbrace{q \Longleftrightarrow r}^{q \Rightarrow r \wedge r \Rightarrow q} \tag{6.29}$$

Wzór (6.29) przedstawia przygotowany drugi nawias klamrowy, który trzeba tylko „wsunąć” na właściwe miejsce. Właściwy rozmiar klamry uzyskaliśmy dzięki użyciu polecenia `phantom` (zob. listing 6.24, linia 3). Parametrem jest dokładnie treść, która powinna zostać objęta klamrą. Skoro chcemy uzyskać efekt „wsunięcia”, to potrzebujemy „przekonać” kompilator, że tego nawiasu klamrowego tam nie ma, a dokładniej, że jest, ale nie zajmuje on żadnego miejsca – aby kompilator rozpoczął składanie dalszej części wzoru w miejscu, na którym nam zależy (objętym klamrą). W tym celu użyjemy polecenia **\mathrlap**, bo teraz chcemy niejako przesunąć coś w prawo, a nie wyśrodkować jak nad i pod klamrami. Pełny kod jest na listingu 6.25, a efektem końcowym jest wzór (6.30).

Listing 6.25. Nakładające się nawiasy klamrowe – etap 3

```
1 \begin{equation}
2 \label{eq:rownowaznosc3}
3 \mathrlap{\underbrace{\phantom{p \Longleftrightarrow q}}_
4 {\mathclap{p \Rightarrow q \wedge q \Rightarrow p}}}
5 p \Longleftrightarrow \overbrace{q \Longleftrightarrow r}^
6 {\mathclap{q \Rightarrow r \wedge r \Rightarrow q}}
7 \end{equation}
```

$$\underbrace{p \Longleftrightarrow q}_{p \Rightarrow q \wedge q \Rightarrow p} \overbrace{\Longleftrightarrow r}^{q \Rightarrow r \wedge r \Rightarrow q} \tag{6.30}$$

Pakiet `mathtools` oferuje także polecenia **`\underbracket`** i **`\overbracket`**. Używamy ich analogicznie, ale zamiast klamer rysują poziomy nawias kwadratowy.

Jak już wcześniej wspomniałem, LaTeX „wie", co robi, i nie powinniśmy go poprawiać. Zdarza się jednak, że z jakiegoś powodu potrzebujemy zmodyfikować odstępy w trybie matematycznym. W tym celu możemy skorzystać z poleceń przedstawionych w tabeli 6.6. Ponadto można także użyć polecenia **`\hspace`**.

Tabela 6.6. Polecenia generujące odstępy w trybie matematycznym

Polecenie	Wielkość odstępu
`\!`	$-3/18$ em
`\,`	$3/18$ em
`\:`	$4/18$ em
`\;`	$5/18$ em
`\` (spacja po ukośniku)	wielkość spacji w normalnym tekście
`\quad`	1 em
`\qquad`	2 em

Przykładem składu, w którym zalecana jest korekta odstępów, jest definiowanie zbiorów przez warunek, np. we wzorze `$A = \{\,x^2\colon x < 7\,\}$`, co daje $A = \{x^2\colon x < 7\}$, zalecane jest dodanie małych poziomych odstępów po otwierającym i przed zamykającym nawiasem klamrowym.

Innym problemem, który możemy napotkać, jest skład liczb rzeczywistych. Jeżeli stosujemy jako separator części dziesiętnej kropkę, to efekt będzie raczej zgodny z naszym oczekiwaniem, np. wzór `$123.567$` wygeneruje 123.567. Jeżeli w miejsce kropki użyjemy przecinka, otrzymamy w wyniku 123,567. Wyraźnie widać dodatkowy odstęp umieszczony po przecinku. LaTeX nie traktuje tutaj przecinka jako separatora części dziesiętnej, a jako separator tysięcy. Jeżeli takich liczb mamy w tekście niewiele, to możemy samodzielnie poprawić skład, wstawiając ujemny odstęp, tj. kod `$123,\!567$` wygeneruje 123,567.

Jeżeli potrzebujemy bardziej kompleksowego rozwiązania, to możemy skorzystać z pakietu `siunitx` [52]. Pakiet zawiera polecenia pozwalające na jednolity skład liczb wraz z jednostkami fizycznymi w całym dokumencie. Dodatkowo mamy możliwość dopasowania funkcjonowania poszczególnych poleceń z pakietu za pomocą szeregu opcji. Wracając do problemu składania liczb rzeczywistych, jeżeli chcemy zastosować do ich składu pakiet `siunitx`, liczby umieszczamy jako parametry polecenia **`\num`**`[]{}`. Obowiązkowym parametrem polecenia jest liczba. Parametr opcjonalny jest używany do przekazywania dodatkowych opcji zdefiniowanych w pakiecie. Przykłady użycia polecenia **`\num`** i ich efekty przedstawiono na listingu 6.26.

```
\usepackage{siunitx}
```

Listing 6.26. Skład liczb z zastosowaniem pakietu `siunitx`

```
1 \sisetup{output-decimal-marker={,}}
2
3 \num{1234567890}\\
4 \num{12345.006789}\\
5 \num{1.23456e-7}\\
6 \num{1,23456e-7}\\
7 \num{.123456e-7}\\
8 \num{e6}
```

1 234 567 890
12 345,006 789
$1{,}234\,56 \times 10^{-7}$
$1{,}234\,56 \times 10^{-7}$
$0{,}123\,456 \times 10^{-7}$
10^{6}

Polecenie `\sisetup{}` w linii 1 ustawia przecinek jako separator części dziesiętnej (na wyjściu – w pliku `pdf`). Możemy je umieścić w preambule dokumentu. Jeżeli porównamy zapis w liniach 5 i 6, to widać, że nie ma znaczenia, czy w kodzie źródłowym użyjemy jako separatora kropki, czy przecinka. W obu przypadkach w skompilowanym kodzie widzimy przecinek, zgodnie z ustawieniami w linii 1. Przy domyślnym formatowaniu (zmieniliśmy tylko symbol separatora) pakiet grupuje cyfry w liczbie po trzy i wstawia między grupami małe odstępy, nie mamy odstępów przed i po separatorze części dziesiętnej, w elegancki sposób obsługiwany jest format wykładniczy.

Pakiet `siunitx` zawiera więcej poleceń. Na przykład polecenie `\ang[]{}` służy do formatowania miary kąta (rola parametrów jest analogiczna jak w poleceniu `\num`). W najprostszej wersji `\ang{30}` wygeneruje 30°, ale możemy podać nie tylko liczbę stopni, lecz również minut i sekund, np. `\ang{30;5;10}` wygeneruje 30°5′10″. Z kolei polecenie `\unit[]{}` służy do formatowania zapisu jednostek. Parametr obowiązkowy zawiera opis jednostek, a opcjonalny ewentualne dodatkowe opcji, np. polecenie `\unit{kg.m/s^2}` wygeneruje $\mathrm{kg\,m/s^2}$. Przedstawionych poleceń możemy używać w trybie tekstowym i matematycznym. Więcej informacji można znaleźć w dokumentacji pakietu [52].

Jeżeli koniecznie tego potrzebujemy, to możemy modyfikować nie tylko kwestię odstępów. LaTeX składa wzory matematyczne, stosując cztery style:

– `\textstyle` – litery i symbole składane jak w trybie wewnątrzakapitowym;

– **\displaystyle** – litery i symbole są składane jak w trybie wystawowym;

– **\scriptstyle** – stosowane są mniejsze czcionki, jak w indeksach;

– **\scriptscriptstyle** – stosowane są jeszcze mniejsze czcionki (w porównaniu z **\scriptstyle**), jak w zagnieżdżonych indeksach.

Style te są dobierane przez LaTeX-a automatycznie zależnie od części składanego wyrażenia. Możemy jednak wymusić stosowanie określonego stylu. Przykład pokazano na listingu 6.27.

Listing 6.27. Wymuszenie użycia stylu wystawowego wewnątrz akapitu

```
Suma nieskończona $\displaystyle\sum_{i=1}^{\infty}\frac{1}{2^i}$
jest wartością skończoną i wynosi 1.
```

Suma nieskończona $\displaystyle\sum_{i=1}^{\infty}\frac{1}{2^i}$ jest wartością skończoną i wynosi 1.

W przedstawionym przykładzie ta zmiana nie przeszkadza, bo cały akapit składa się z jednego wiersza i można uznać, że ten skład jest w porządku. Dla porównania rozważmy takie dwa proste przykłady `$\sqrt[3]{x+1}$` i `$\sqrt[\textstyle 3]{x+1}$`, które generują odpowiednio $\sqrt[3]{x+1}$ i $\sqrt[\textstyle 3]{x+1}$. Trudno uznać, że nietypowy skład w drugim przypadku jest elegancki.

Rozdział 7

Zarządzanie grafiką

W rozdziałach 10 i 11 omawiam dwa podejścia do przygotowania grafiki, które doskonale współpracują z LaTeX-em. W tym rozdziale zajmiemy się nie przygotowaniem grafiki, ale zarządzaniem nią z perspektywy dokumentu. Zakładamy, że grafika, którą chcemy osadzić w dokumencie, znajduje się w oddzielnych plikach. Nie ma tutaj znaczenia, w jakim narzędziu ją przygotowano ani czy jest to grafika bitowa, czy wektorowa. Jeżeli jest to tylko format akceptowany przez LaTeX-a, to możemy dołączyć ją do dokumentu.

7.1 Osadzanie grafiki w dokumencie

Zaczynając od pierwszego przykładu (zob. listing 1.1, str. 4), w kilku miejscach pojawiło się polecenie **\includegraphics**, które pozwala dołączyć do dokumentu zewnętrzny plik. Polecenie jest dostępne po dołączeniu pakietu `graphicx` [9]. Polecenie **\includegraphics**`[]{}` jako jedyny obowiązkowy parametr przyjmuje nazwę pliku. Jeżeli mamy ustawioną ścieżkę do katalogu z grafiką (polecenie **\graphicspath**) lub plik z grafiką znajduje się w tym samym katalogu co nasz dokument, to wystarczy sama nazwa pliku. W innym przypadku trzeba podać ścieżkę (format ścieżki zależy od systemu operacyjnego). Jeżeli nie ma problemu z jednoznacznością, to nie jest wymagane rozszerzenie pliku.

```
\usepackage{graphicx}
\graphicspath{{./img/}}  % przykładowa ścieżka
                         % (format uniksowy)
```

Jeżeli korzystamy z programu `pdflatex` do kompilacji plików `tex` (takie założenie przyjęliśmy w tej książce), to możemy dołączać grafikę w formatach `pdf`,

`jpeg`, `tiff` i `png`. Szczególnie preferowany jest pierwszy, gdyż pozwala zapisywać grafikę wektorową. Jako parametr opcjonalny polecenia **`\includegraphics`** można dołączyć szereg opcji (oddzielonych przecinkami), które wpływają na sposób osadzenia grafiki. Najważniejsze zostały przedstawione w tabeli 7.1.

Tabela 7.1. Wybrane opcje polecenia **`\includegraphics`**

Opcja	Przeznaczenie
`scale=x`	skalowanie rysunku z zachowaniem proporcji
`width=x`	ustawienie wskazanej szerokości rysunku
`height=x`	ustawienie wskazanej wysokości rysunku
`keepaspectratio`	wymuszenie utrzymania proporcji rysunku
`origin=x`	określenie środka obrotu – 1 lub 2 litery z `lrctb`
`angle=x`	kąt obrotu (w stopniach)
`trim l b r t`	zdefiniowanie widoku rysunku przyciętego o wskazane wymiary od lewej, od dołu, od prawej i od góry
`bb x1 y1 x2 y2`	zdefiniowanie widoku rysunku w postaci prostokąta, którego lewym dolnym narożnikiem jest punkt `(x1,y1)`, a prawym górnym – `(x2,y2)`
`clip`	przycięcie rysunku do zdefiniowanego widoku

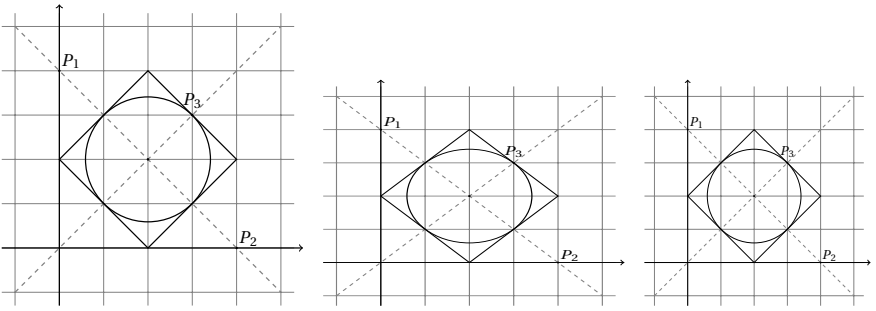
Jeżeli chcemy przeskalować rysunek, to można to zrobić na kilka sposobów. Możemy określić skalę, np. `scale=0.5`, i rysunek zostanie zmniejszony do 50% swoich oryginalnych wymiarów z zachowaniem proporcji. Możemy określić wyłącznie szerokość, np. `width=5cm`, rysunek zostanie przeskalowany z zachowaniem proporcji do zadanej szerokości. Często spotyka się wersję `width=\textwidth`, co pozwala wykorzystać pełną szerokość strony. Analogicznie możemy podać wyłącznie wysokość, a pakiet sam dobierze szerokość. Jeżeli określimy zarówno szerokość, jak i wysokość, to zachowanie zależy od tego, czy dodamy opcję `keepaspectratio`. Jeżeli tak, to skalowanie odbędzie się z zachowaniem proporcji, tak aby nie przekroczyć żadnego z podanych wymiarów. Jeżeli pominiemy tę opcję, to rysunek zostanie przeskalowany dokładnie do zadanych wymiarów, być może psując jego proporcje. Skalować możemy również tak, aby osadzony rysunek był większy od oryginału, np. `scale=1.2`. W przypadku grafiki bitowej może to jednak oznaczać słabą jakość, zwłaszcza jeśli znacznie powiększymy rysunek.

Przykład efektów skalowania tego samego rysunku przy różnym zestawie opcji pokazano na listingu 7.1. Nasz testowy rysunek ma taką samą szerokość i wysokość, więc w pierwszym przypadku dostajemy rysunek o wymiarach 4 cm × 4 cm. W drugim przypadku dostajemy zdeformowany rysunek dopasowany

do wymiarów 4 cm × 3 cm. W ostatnim przypadku, dostajemy rysunek w wymiarze 3 cm × 3 cm – największy możliwy rozmiar przy podanych ograniczeniach szerokości i wysokości, który nie psuje proporcji.

Listing 7.1. Przykłady skalowania rysunku

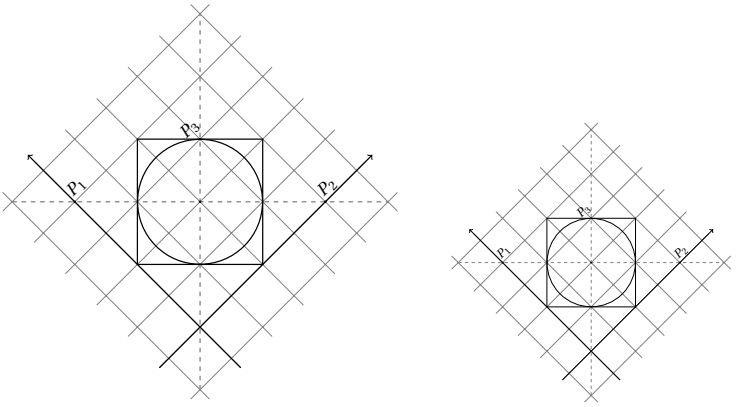
```
\includegraphics[width=4cm]{demo.pdf}\quad
\includegraphics[width=4cm, height=3cm]{demo.pdf}\quad
\includegraphics[width=4cm, height=3cm, keepaspectratio]{demo.pdf}
```



Jeżeli chcemy wykonać jednocześnie skalowanie i obrót, to istotna jest kolejność podawania opcji. Są one czytane od lewej do prawej i w tym przypadku w takiej kolejności będą wykonywane operacje na rysunku. Zestawy opcji `[width=5cm, angle=45]` i `[angle=45, width=5cm]` dadzą różne efekty, co pokazano na listingu 7.2.

Listing 7.2. Przykłady łączenia skalowania z obrotem

```
\includegraphics[width=4cm, angle=45]{demo.pdf}\quad
\includegraphics[angle=45, width=4cm]{demo.pdf}
```



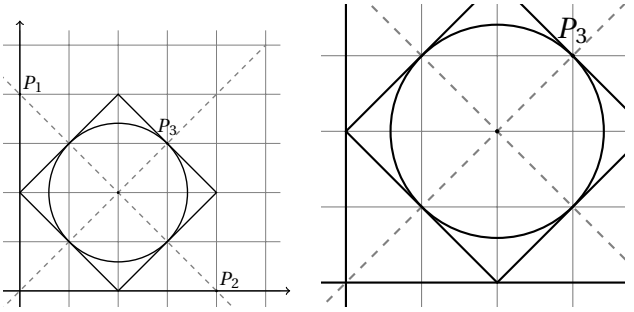
W pierwszym przypadku rysunek najpierw jest skalowany (bok ma wtedy 5 cm), a później realizowany jest obrót. W drugim przypadku, po wykonaniu obrotu, skalowanie powoduje, że to przekątna oryginalnego rysunku ma 5 cm.

Jeżeli obracamy rysunek, to domyślnie punktem obrotu jest środek rysunku. Za pomocą opcji `origin` możemy określić inny środek obrotu, np. `origin=tr` oznacza, że będziemy obracać względem prawego górnego narożnika.

Jeżeli chcemy przyciąć rysunek, to trzeba połączyć opcje `trim` i `clip` lub `bb` i `clip`. Załóżmy, że chcemy przyciąć nasz rysunek o 1 cm od lewej strony i od dołu. Opcja `trim=1cm 1cm 0 0` zdefiniuje nam dwa centymetrowe obszary do odcięcia. Opcja `clip` zrealizuje przycięcie. W przypadku opcji `bb` podajemy lewy dolny i prawy górny narożnik prostokąta, do którego chcemy przyciąć rysunek. Przykład użycia obu metod pokazano na listingu 7.3.

Listing 7.3. Przykłady przycinania rysunku

```
1 \includegraphics[trim=1cm 1cm 0 0, clip, scale=0.65]{demo.pdf}\quad
2 \includegraphics[bb=1cm 1cm 5cm 5cm, clip]{demo.pdf}
```



Metody przycinania rysunków pokazane na listingu 7.3 dają ciekawe możliwości. Załóżmy, że mamy dość duży diagram i przeskalowanie go tak, aby w całości zmieścił się na stronie, powoduje, że wiele szczegółów jest niewidocznych. Możemy w takim przypadku zamieścić w całości taki przeskalowany rysunek, aby pokazać strukturę całości, a potem na kolejnych rysunkach pokazać fragmenty diagramu, ale już w większej skali. To wszystko uzyskamy na podstawie jednego pliku z grafiką, który załączamy wielokrotnie do dokumentu, odpowiednio manewrując opcjami polecenia **`\includegraphics`**.

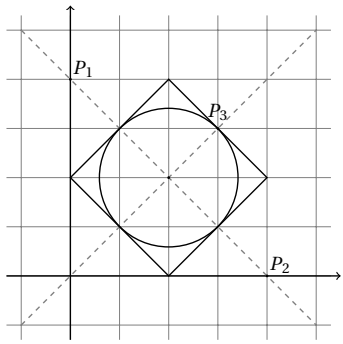
7.2 Rysunki jako wstawki

Polecenie **`\includegraphics`** możemy wykorzystać w dowolnym miejscu, gdzie tylko będzie to miało sens. Widzieliśmy przykład umieszczenia grafiki w komórkach tabeli (zob. listing 5.2, str. 88), w rozdziale 11 zawarto przykłady osadzania

grafiki jako węzeł w innym rysunku, możemy ją dodać jako logo w stopce strony czy wręcz w środku akapitu z tekstem. Jeżeli chcemy utworzyć z niej wstawkę, musimy użyć środowiska **figure**. Wewnątrz tego środowiska możemy umieścić polecenia **\caption** i **\label** (koniecznie w tej kolejności i bezpośrednio po sobie), które definiują podpis i etykietę rysunku. Zazwyczaj są one umieszczane pod rysunkiem. Możemy też dodać polecenie, które wyśrodkuje rysunek lub dosunie go do prawej strony. Przykład typowego kodu widzimy na listingu 7.4.

Listing 7.4. Przykład osadzenia rysunku wewnątrz środowiska **figure**

```
\begin{figure}[!ht]
\centering
\includegraphics[scale=0.65]{demo.pdf}
\caption{Rysunek demonstracyjny do osadzania grafiki}
\label{fig:demo}
\end{figure}
```



Rys. 7.1. Rysunek demonstracyjny do osadzania grafiki

Efekt kompilacji kodu z listingu 7.4 widzimy na rys. 7.1. Rysunki, a dokładniej środowiska **figure**, mają własną kolejkę FIFO. Mamy więc gwarancję, że zostaną one umieszczone w pliku `pdf` w tej kolejności w jakiej są w pliku źródłowym, ale niekoniecznie w tym samym miejscu. Stosujemy tutaj dokładnie te same opcje co w przypadku środowiska **table**: `h` – w miejscu użycia, `t` – na górze strony, `b` – na dole strony i `p` – na stronie zawierającej wyłącznie wstawki. Dodatkowo nasze oczekiwania możemy „wzmocnić” opcją `!`.

Jeżeli korzystamy z klasy `book` lub `report`, to wstawki nigdy nie będą przesuwane między rozdziałami. Ograniczenie to nie dotyczy sekcji, choć można to wymusić, dołączając pakiet `placeins` [2] w formie **\usepackage**`[section]{placeins}`. Jak zostało wspomniane przy okazji tabel, możemy dołączyć pakiet `float` [35] i korzystać z dodatkowej opcji `H` – gdy nie można wstawki umieścić

w miejscu, które wynika z kodu, to jest ona przenoszona na następną stronę, ale nie następuje przesunięcie treści – strona zostaje niekompletna.

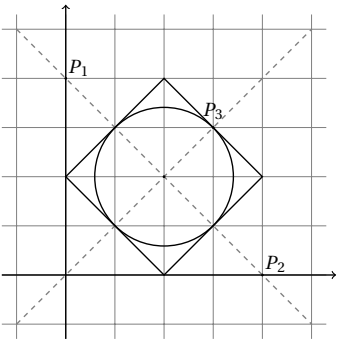
Środowisko **figure** nie musi w sobie zawierać polecenia **\includegraphics** – możemy użyć innej instrukcji osadzającej grafikę, możemy użyć środowiska, w którym tworzymy rysunek (np. **tikzpicture**, zob. rozdział 11), możemy połączyć rysunek z tekstem albo wcale nie umieszczać rysunku w środku. Niezależnie od tego, co zrobimy, LaTeX będzie postrzegał zawartość jako rysunek i umieści ją w odpowiedniej kolejce FIFO.

W jednym środowisku **figure** możemy umieścić kilka rysunków. Jest tutaj sytuacja analogiczna jak z dwoma środowiskami **tabular** umieszczonymi w jednym środowisku **table** (zob. listing 5.21, str. 109). Jeżeli kilka rysunków ma być umieszczonych obok siebie, to po prostu podajemy kilka instrukcji **\includegraphics** z ewentualnymi odstępami między nimi (zob. listing 7.1). O ile ich łączna szerokość (+ odstępy) nie przekroczy szerokości strony, to zostaną umieszczone w jednym wierszu. Skutkiem umieszczenia ich w jednym środowisku **figure** jest wspólny podpis i wspólny numer rysunku. Jeżeli chcemy, aby rysunki miały indywidualne podpisy i numery, to konieczne jest użycie środowisk **minipage** zagnieżdżonych w środowisku **figure**, jak pokazano na listingu 7.5 i rys. 7.2–7.4.

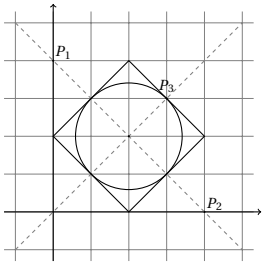
Listing 7.5. Przykład osadzenia kilku rysunków wewnątrz środowiska **figure**

```
\begin{figure}[!hbt]
\begin{minipage}[b]{0.3\textwidth}
\centering
\includegraphics[scale=0.65]{demo.pdf}
\caption{Skala 0.65}
\label{fig:demo:065}
\end{minipage}\hspace{\fill}
%
\begin{minipage}[b]{0.3\textwidth}
\centering
\includegraphics[scale=0.5]{demo.pdf}
\caption{Skala 0.5}
\label{fig:demo:05}
\end{minipage}\hspace{\fill}
%
\begin{minipage}[b]{0.3\textwidth}
\centering
\includegraphics[scale=0.35]{demo.pdf}
```

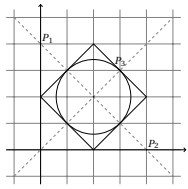
```
\caption{Skala 0.35}
\label{fig:demo:035}
\end{minipage}
\end{figure}
```



Rys. 7.2. Skala 0.65

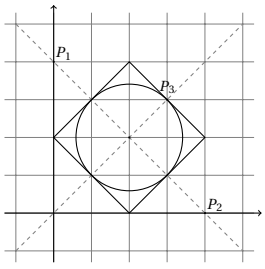


Rys. 7.3. Skala 0.5



Rys. 7.4. Skala 0.35

Wszystkie ministrony zostały zdefiniowane z tą samą szerokością (30% szerokości tekstu strony). Dodatkowo w liniach 7 i 14 zastosowaliśmy polecenie dodające elastyczny odstęp (`\hspace{\fill}`). To dało w efekcie równomierny rozkład ministron w wierszu. Istotna jest tutaj również opcja `b` odnosząca się do sposobu wyrównania ministron – gwarantuje ona umieszczenie wszystkich podpisów w jednej linii. W przypadku domyślnego wyśrodkowania w pionie z każdym kolejnym rysunkiem napis byłby nieco wyżej, gdyż rysunki są coraz mniejsze. W prezentowanym przykładzie użyto bardzo krótkich podpisów. Przy dłuższych tekst będzie zawijany w ramach szerokości danej ministrony i nie będzie to wyglądać zbyt estetycznie.



Rys. 7.5. Po lewej

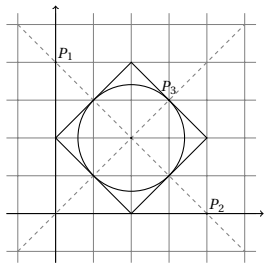
Nie wszystkie ministrony muszą zawierać rysunki. Jeżeli podzielimy obszar na dwie ministrony i jedną z nich uzupełnimy tekstem (zob. listing 7.6), to w efekcie uzyskamy tekst umieszczony obok rysunku, tak jak widzimy to w tym akapicie. Istotne jest tutaj, aby pamiętać, że zewnętrznym środowiskiem jest **`figure`**, a środowiska **`minipage`** są umieszczone w środku. Całość jest jedną wstawką, więc jeśli nie zmieści się na danej stronie, to zostanie przesunięta zgodnie z użytymi opcjami środowiska **`figure`**. Jest to o tyle istotne, że może nam zmienić kolejność akapitów.

Listing 7.6. Umieszczanie tekstu obok rysunku

```
\begin{figure}[!hbt]
\begin{minipage}[c]{0.3\textwidth}
\centering
\includegraphics[scale=0.5]{demo.pdf}
\caption{Po lewej}
\label{fig:demo:05x}
\end{minipage}\hspace{\fill}
%
\begin{minipage}[c]{0.68\textwidth}
Nie wszystkie ministrony ...
\end{minipage}
\end{figure}
```

```
\usepackage{wrapfig2}
```

Ograniczeniem tego podejścia jest konieczność dopasowania ilości tekstu do rozmiaru rysunku, aby efekt był estetyczny. Innym rozwiązaniem jest pozwolenie, aby tekst „opływał" rysunek, jak to widzimy w tym akapicie – tekst nad, obok i pod rysunkiem tworzy jeden akapit. W tym celu możemy użyć środowiska **wrapfigure**, które zdefiniowane jest w pakiecie `wrapfig2` [4] – pakiet zawiera również środowiska **wraptable** i **wraptext**. Układ możliwych parametrów jest następujący: `\begin{wrapfigure}[]{}[]{}`. Dwa obowiązkowe parametry to lokalizacja grafiki (`l` – lewa strona, `r` – prawa strona, `i` – wewnętrzna strona, `o` – zewnętrzna strona) i szerokość obszaru na grafikę. Pierwszy opcjonalny parametr pozwala na podanie, ile linii tekstu ma zostać zwężone, aby osadzić rysunek. Jeżeli go pominiemy, to pakiet policzy sam, ile trzeba miejsca w pionie, aby rysunek się zmieścił. Jeśli określimy za mało miejsca, to część tekstu będzie nałożona na rysunek. Przy zbyt dużej liczbie linii – pod rysunkiem będzie wolna przestrzeń. Drugi opcjonalny parametr pozwala na określenie, na ile rysunek może być przesunięty na margines. W szczególności bardzo mały rysunek możemy w całości wsunąć na margines. Przykład użycia środowiska **wrapfigure** (bieżący akapit) pokazano na listingu 7.7. Jak widzimy, środowisko jest włożone w środek akapitu tekstu. Jeżeli umieścimy je na początku, to tekst będzie opływał tylko od spodu, jeżeli na końcu akapitu – to tylko od góry.



Rys. 7.6. Po prawej

Listing 7.7. Tekst opływający rysunek

```
Ograniczeniem tego podejścia jest konieczność dopasowania ilości
tekstu do rozmiaru rysunku, aby efekt  był estetyczny. Innym
rozwiązaniem jest pozwolenie, aby tekst ,,opływał'' rysunek, jak
to widzimy w tym akapicie -- tekst nad, obok i pod rysunkiem
tworzy jeden akapit.
%
\begin{wrapfigure}{r}{4.2cm}
\centering
\includegraphics[scale=0.5]{demo.pdf}
\caption{Po prawej}
\label{fig:demo:05:wrap}
\end{wrapfigure}
%
W tym celu możemy użyć ...
```

Akapit ze środowiskiem **wrapfigure** nie jest obiektem pływającym. Nie dotyczy go zasada kolejki FIFO dla rysunków umieszczonych wewnątrz środowiska **figure**. Może się więc zdarzyć, że znajdzie się przed rysunkiem, który jest później wstawiony w kodzie.

7.3 Specjalne metody osadzania grafiki

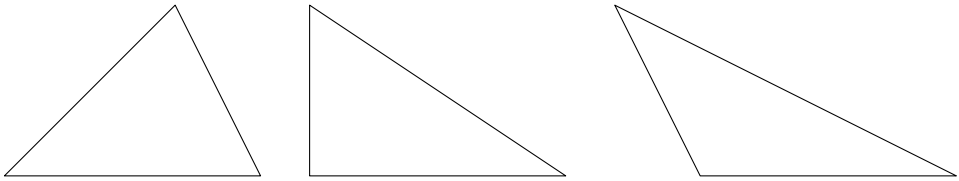
Na listingu 7.5 przedstawiono przykład połączenia środowisk **figure** i **minipage** do umieszczenia kilku rysunków obok siebie. Rysunki te miały niezależne numery i niezależne podpisy. Pakiet `subfig` [10] pozwala na nieco inne rozwiązanie – rysunki współdzielą numer i podpis, ale mają własne numerowanie drugiego rzędu i dodatkowe podpisy. Przykład takiego rozmieszczenia widzimy na rys. 7.7, a kod źródłowy pokazano na listingu 7.8.

```
\usepackage{subfig}
```

Listing 7.8. Przykład zastosowania pakietu `subfig`

```
\begin{figure}[!htb]
\centering
\subfloat[Trójkąt ostrokątny]
{\includegraphics{ostrokatny.pdf}
\label{fig:trojkat:ostrokatny}}\hspace{\fill}
\subfloat[Trójkąt prostokątny]
{\includegraphics{prostokatny.pdf}
\label{fig:trojkat:prostokatny}}\hspace{\fill}
```

```
\subfloat[Trójkąt rozwartokątny]
{\includegraphics{rozwartokatny.pdf}
\label{fig:trojkat:rozwartokatny}}
\caption{Trójkąty}
\label{fig:trojkaty}
\end{figure}
```



(a) Trójkąt ostrokątny **(b)** Trójkąt prostokątny **(c)** Trójkąt rozwartokątny

Rys. 7.7. Trójkąty

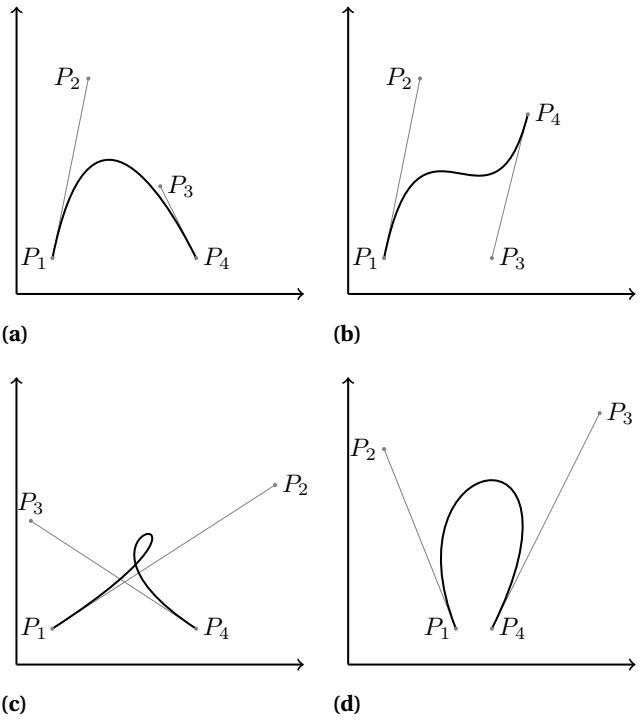
Pakiet `subfig` zawiera polecenie `\subfloat[]{}`, którego można użyć nie tylko wewnątrz środowiska **`figure`**, ale również wewnątrz środowiska **`table`**. Parametrem obowiązkowym jest zawartość jednego składnika. Na listingu 7.8 ta zawartość to polecenie `\includegraphics` i definicja etykiety. W przypadku tabel tym parametrem może być środowisko **`tabular`**. Parametr opcjonalny zawiera dodatkową etykietę dla tego składnika. Taka organizacja grafiki pozwala powoływać się na rysunek jak na całość, ale również odnosić się do składników, na przykład odwołanie `\ref{fig:trojkaty}` generuje nam 7.7, zaś odwołanie `\ref{fig:trojkat:rozwartokatny}` – 7.7c.

W rozważanym przykładzie trzy rysunki zostały umieszczone w jednej linii. Nie ma problemu, aby łamać linie i kolejne rysunki umieszczać w wielu wierszach. Możemy też wykorzystać środowisko **`tabular`**. Przykład taki pokazano na listingu 7.9, a efekt kompilacji na rys. 7.8. W rozważanych przykładzie polecenie `\subfloat` jest używane raz w każdej komórce tabeli. Warto zwrócić uwagę na puste podpisy. Jeżeli pominiemy nawiasy kwadratowe, to nie będziemy mieć wewnętrznej numeracji. Zostawienie pustych nawiasów powoduje, że pod rysunkami nie ma wprawdzie podpisów, ale mamy (a), (b) itd.

Listing 7.9. Przykład zastosowania pakietu `subfig` i układu tabelarycznego

```
\begin{figure}[!htb]
\centering
\begin{tabular}{cc}
\subfloat[]
{\includegraphics{bezier1.pdf} \label{fig:tab:bezier1}}&
\subfloat[]
{\includegraphics{bezier2.pdf} \label{fig:tab:bezier2}}\\
```

```
\subfloat[]
{\includegraphics{bezier3.pdf} \label{fig:tab:bezier3}}&
\subfloat[]
{\includegraphics{bezier4.pdf} \label{fig:tab:bezier4}}\\
\end{tabular}
\caption{Krzywe B\'eziera}
\label{fig:krzywe:beziera}
\end{figure}
```



Rys. 7.8. Krzywe Béziera

Na listingu 5.23 przedstawiony został kod pozwalający uzyskać tabelę obróconą o 90°. W przypadku rysunków do dyspozycji mamy środowisko **sidewaysfigure** z pakietu `rotating` [14], którego używamy zamiast **figure**. Środowisko składa rysunek na oddzielnej stronie, wykorzystując wysokość strony jako maksymalną szerokość rysunku. Rysunek jest umieszczony w kolejce FIFO, więc kolejność rysunków nie będzie zmieniona. Przykład użycia środowiska pokazano na listingu 7.10, a efekt kompilacji znajduje się na rys. 7.9. Rysunek pokazuje, że większość języków programowania, które powstały 40 i więcej lat temu, nadal funkcjonuje. Czarny znacznik po prawej stronie wskazuje rok

wydania ostatniej wersji danego języka. Próba umieszczenia rysunku na szerokości strony sprawia, że jest mało czytelny. W tym przypadku wykorzystano wysokość strony jako szerokość rysunku. Jeżeli wyłącznie obrócimy rysunek w ramach środowiska **figure**, to rysunek będzie w układzie poziomym, a podpis pionowym. Dodatkowo zmarnujemy część wysokości strony na podpis.

```
\usepackage{rotating}
```

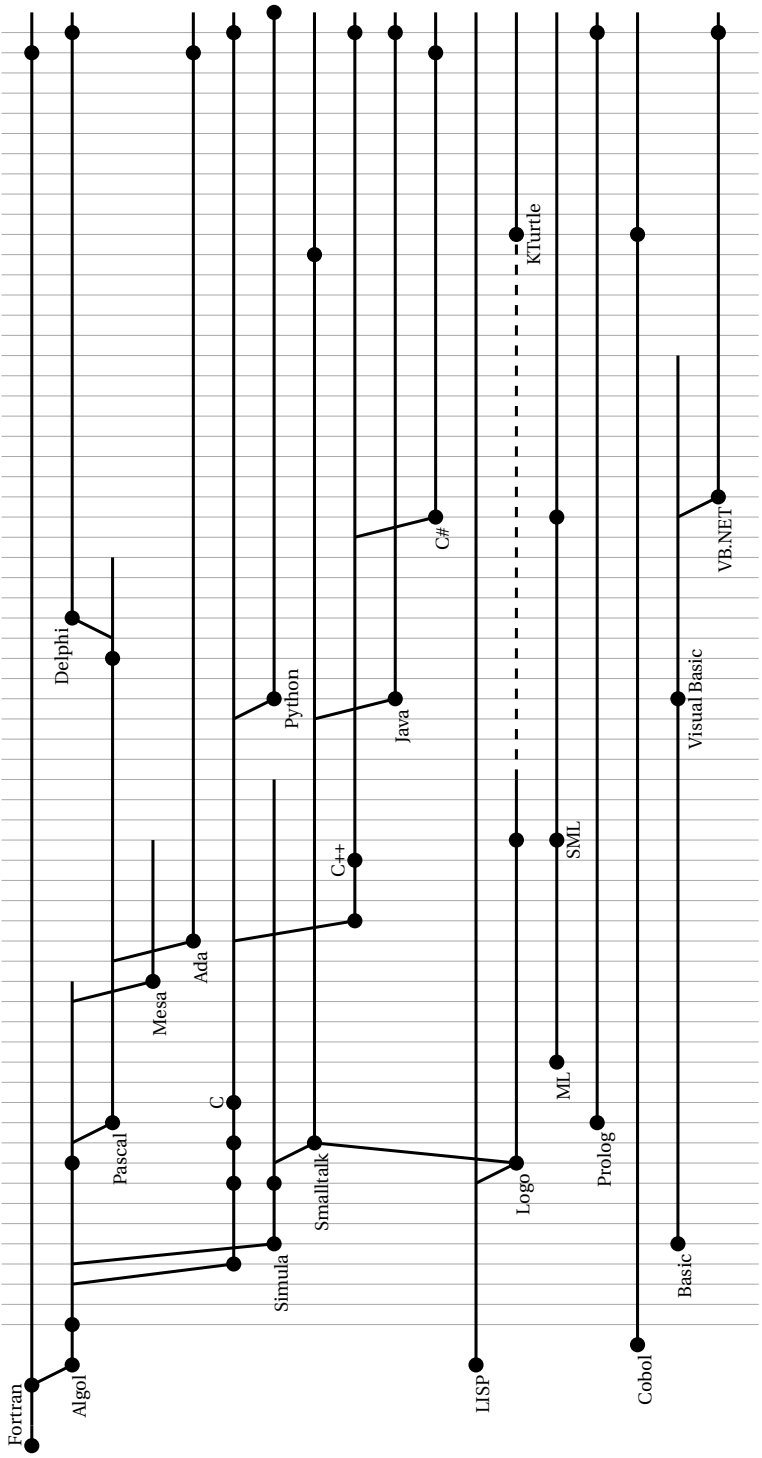
Listing 7.10. Przykład wykorzystania środowiska **sidewaysfigure**

```
\begin{sidewaysfigure}[p]
\centering
\includegraphics[width=\textwidth]{languages.pdf}
\caption{Języki programowania lat 70. i 80.}
\label{fig:jezyki:programowania}
\end{sidewaysfigure}
```

Jeżeli składamy na przykład artykuł do czasopisma, w którym obowiązuje skład dwukolumnowy, to użycie środowiska **figure** umieszcza rysunek w pojedynczej kolumnie. Przy większych rysunkach możemy skorzystać z możliwości składu na całej szerokości strony. Realizujemy to z użyciem środowiska **figure***. Pomimo gwiazdki w nazwie rysunek będzie numerowany i możemy się do niego odwoływać. Gwiazdka reprezentuje w tym przypadku specjalne rozmieszczenie rysunku. Analogicznie zachowuje się środowisko **table***, które pozwala złożyć tabelę na całej szerokości strony.

Na koniec rozważmy sytuację, w której chcemy wstawić do naszego dokumentu kompletną stronę, na przykład jest to broszurka w formacie A4, której nie chcemy skalować, ale dołączyć ją do dokumentu jako dodatkową stronę. Możliwości takie daje pakiet `pdfpages` [38]. Pakiet zawiera polecenie `\includepdf[]{}`. Parametrem obowiązkowym jest nazwa pliku `pdf`, z którego chcemy pobrać naszą stronę. Parametr opcjonalny może zawierać szereg opcji postaci `klucz=wartość`. Najważniejszą jest opcja `pages`, np. polecenie `\includepdf[pages={7}]{abc.pdf}` w miejscu swojego umieszczenia doda 7 stronę z dokumentu `abc.pdf`.

Parametr `pages` może przyjmować zakresy. W wersji `pages={5-12}` dołączone zostaną wszystkie strony z zakresu 7–12, a w wersji `pages={5-last}` – strony od piątej do końca dokumentu. Jeżeli zastosujemy wyłącznie znak minus, tj. `pages=-`, to dołączony zostanie cały wskazany dokument. Widzimy więc, że w praktyce daje nam to możliwość łączenia plików `pdf`. Pisząc raport w LaTeX-u, możemy bez problemu dołączyć do dokumentu załączniki z plików `pdf`.



Rys. 7.9. Języki programowania lat 70. i 80.

7.4 Zarządzanie podpisami

W przykładach dotyczących środowisk **table** i **figure** mogliśmy zauważyć, że położenie podpisu zależy od miejsca, gdzie umieścimy polecenie **\caption**. Dla przykładu przeniesienie podpisu pod tabelę sprowadza się do umieszczenia polecenia **\caption** poniżej środowiska **tabular**. Pakiet caption [43] udostępnia narzędzia do zarządzania wyglądem podpisów i odstępami przed i po podpisach. Jest to realizowane poprzez polecenie **\captionsetup**[]{}. Parametrem obowiązkowym jest lista opcji, parametrem opcjonalnym jest typ wstawki, do której odnoszą się ustawienia. Jeżeli pominiemy parametr opcjonalny, to ustawienia dotyczą wszystkich typów wstawek (np. zarówno tabel, jak i rysunków). Polecenie **\captionsetup** można umieścić w preambule, aby zdefiniować ustawienia dla całego dokumentu. Możemy go użyć również wewnątrz środowisk **table**, **figure** itd. i wtedy zmiany będą dotyczyły tylko konkretnej wstawki. Jeżeli polecenia umieścimy w obu miejscach, to ustawienia w środowisku będą nadpisywały ustawienia globalne, gdy dotyczą tych samych aspektów, i uzupełniały je, gdy odnoszą się do innych.

```
\usepackage{caption}
```

Weźmy pod uwagę podpis z ostatniego rysunku. „Rys. 7.9” jest etykietą, kropka występująca po numerze jest separatorem, a reszta tekstu jest treścią podpisu. W tabeli 7.2 przedstawiono najczęściej używane opcje stosowane do formatowania podpisów. Pełny przegląd znajduje się w dokumentacji pakietu.

Czcionki ustawiamy, podając w nawiasach klamrowych opcje, które mają być zastosowane (nawiasy są zbędne, jeśli używamy tylko jednej opcji). Dostępne opcje to m.in. `scriptsize`, `footnotesize`, `small`, `normalsize`, `large`, `Large`, `up` (czcionka prosta), `it` (kursywa), `sl` (czcionka pochyła), `sc` (kapitaliki), `md` (medium series), `bf` (bold series), `rm` (serif), `sf` (sans serif), `tt` (monospace), `color=x`, gdzie `x` jest nazwą koloru. Jeżeli w środowisku chcemy dodać następną opcję do ustawień czcionki, to można użyć opcji `font+=` zamiast `font=` itd.

Zazwyczaj wygląd podpisów ustawia się globalnie, a przy finalnym składzie koryguje się nieco odstępy. Dotyczy to szczególnie sytuacji, gdy mamy dwie wstawki bezpośrednio po sobie i daje to wrażenie zbyt dużych pionowych odstępów. Możemy je skorygować, modyfikując opcje `aboveskip` i `belowskip`. Warto zwrócić uwagę na fakt, że ich nazwy mogą być mylące. Pierwsza opcja oznacza odstęp pionowy między treścią wstawki a podpisem. Jeżeli więc użyjemy jej dla środowiska **table** i podpis jest na górze, to tak naprawdę zmieniamy

Tabela 7.2. Wybrane opcje polecenia `\captionsetup`

Opcja	Przeznaczenie
`format`	format, dostępne opcje `plain` – skład jak zwykły akapit i `hang` – wysunięta etykieta
`margin`	wielkość marginesu
`indention`	wcięcie drugiej linii i kolejnych
`aboveskip`	odstęp między podpisem i zawartością środowiska
`belowskip`	odstęp między podpisem i otaczającym tekstem
`font`	ustawienia czcionki dla całego podpisu
`justification`	sposób wyrównania, np. `justified`, `centering`, `raggedright`
`labelfont`	ustawienia czcionki dla etykiety
`labelformat`	format, dostępne opcje `empty` – brak, `simple` – nazwa i numer, `brace` – po numerze nawias, `parens` – numer w nawiasach
`labelsep`	separator, dostępne opcje `none`, `colon`, `period`, `space`, `quad`, `newline`, `endash` (półpauza)
`textformat`	opcja `period` powoduje automatyczne dodawanie kropki
`textfont`	ustawienia czcionki dla treści podpisu
`singlelinecheck`	opcja logiczna, dotyczy specjalnego traktowania podpisów, które mieszczą się w jednej linii

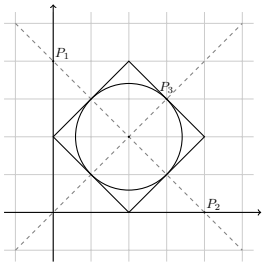
odstęp pod podpisem, który jest powyżej treści wstawki. Opcja `belowskip` dotyczy odstępu między podpisem i treścią otaczającą wstawkę. W przypadku wspomnianej tabeli będzie to odstęp nad podpisem. Dla stawek z podpisami pod spodem zachowanie pasuje do nazw.

Domyślnie LaTeX w klasach standardowych specjalnie traktuje podpisy, które mieszczą się w jednej linii – są one wyśrodkowane. Dla takich podpisów nie mają znaczenia opcje `justification`, `indention` ani nie zobaczymy efektu wysunięcia etykiety dla formatu `hang`. Zachowanie takie można zmienić, ustawiając opcję `singlelinecheck=false`. Wartością domyślną jest tutaj `true`. Przy wartości `false` krótkie podpisy nie będą traktowane w sposób specjalny.

Na listingu 7.11 pokazano przykład ustawień z preambuły dokumentu, a poniżej efekt ich zastosowania w przypadku dłuższego i krótkiego podpisu. Pierwszy podpis ma zawinięty tekst, ale zgodnie z opcją `hang` etykieta jest wysunięta w lewo. Drugi podpis to domyślne traktowanie krótkich tekstów. W trzeciej wersji zastosowano dodatkowo dla rysunku opcję `singlelinecheck=false`.

Listing 7.11. Przykład definiowania ustawień dla podpisów

```
1 \captionsetup{format=hang, margin=2mm, font=small,
2   labelsep=period, labelfont={color=gray,bf,sc}}
```



Rysunek 1. Rysunek demonstracyjny do osadzania grafiki – siatka oraz linie pomocnicze pozwalają łatwo ocenić deformację obrazu

Rysunek 2. Rysunek demonstracyjny do osadzania grafiki

Rysunek 3. Rysunek demonstracyjny do osadzania grafiki

Zdefiniowane zachowanie jest wspólne dla rysunków i tabel. Gdybyśmy chcieli, aby te ustawienia dotyczyły tylko rysunków, to polecenie z linii 1 przyjęłoby formę `\captionsetup[figure]{...}`. Wtedy powinniśmy dodać oddzielne ustawienia dla środowiska **table**.

Jeżeli chcemy zmienić domyślne słowo używane w etykiecie, to musimy redefiniować polecenie **\figurename** w przypadku rysunków lub **\tablename** w przypadku tabel. Dla przykładu: jeżeli zamiast skrótu „Rys.” chcemy mieć pełne słowo „Rysunek”, to należy dodać polecenie **\renewcommand\figurename** `{Rysunek}`. Jeżeli używamy pakietu `babel`, to polecenie to nie powinno być umieszczane w preambule, ale na początku środowiska **document**.

Rozdział 8

Zaawansowane środowiska dla dokumentów naukowych

W rozdziale 6 omówiliśmy metody składania wyrażeń matematycznych. W zależności od typu dokumentu, nad którym pracujemy, same wzory mogą być niewystarczające. Możemy potrzebować numerowanych środowisk do prezentowania definicji, twierdzeń, wniosków itp.; możemy chcieć dołączyć do dokumentu elegancko sformatowane fragmenty kodu źródłowego lub wykorzystać pseudokod do zaprezentowania procedur/algorytmów. W niniejszym rozdziale omówimy polecenia i środowiska, które służą do wzbogacenia dokumentu o wspomniane elementy.

8.1 Środowiska dla definicji, twierdzeń itp.

LaTeX udostępnia możliwość definiowania numerowanych środowisk do składania takich elementów jak definicje, twierdzenia, wnioski, przykłady itd. Środowisko takie nie musi wiązać się ściśle z terminologią matematyczną. Jeżeli chcemy opisywać serię doświadczeń, które mają być numerowane, i móc odwoływać się do nich z użyciem etykiet, to możemy zdefiniować środowisko, którego użycie wygeneruje etykiety: **Doświadczenie 1.**, **Doświadczenie 2.** itd. Za definiowanie środowisk odpowiada polecenie `\newtheorem{}{}`. Jego poprawioną wersję oferuje pakiet `amsthm`, który zawiera predefiniowane style do użycia przez nowe środowiska. Charakterystykę stylów i ich typowe przeznaczenie pokazano w tabeli 8.1. Polecenie `\newtheorem` przyjmuje dwa obowiązkowe parametry. Pierwszym z nich jest nazwa nowego środowiska, a drugim etykieta. Polecenie może też przyjmować parametr opcjonalny, który decyduje o sposobie numerowania. Różne warianty pokazano na listingu 8.1.

```
% koniecznie taka kolejność dołączania tych pakietów
\usepackage{amsmath}
\usepackage{amsthm}
```

Tabela 8.1. Predefiniowane style dla polecenia **\newtheorem**

Nazwa	Opis	Przykłady zastosowania
`plain`	etykieta – pogrubiona czcionka, treść – kursywa, dodatkowy odstęp przed i po środowisku	twierdzenia, lematy, wnioski
`definition`	etykieta – pogrubiona czcionka, treść – prosta czcionka, dodatkowy odstęp przed i po środowisku	definicje, problemy, przykłady, ćwiczenia, własności, hipotezy
`remark`	etykieta – kursywa, treść – czcionka prosta, bez dodatkowych odstępów przed i po środowisku	uwagi, przypadki, konkluzje

Jeżeli nie wskażemy stylu dla nowo definiowanego środowiska, to domyślnie zostanie użyty styl `plain`. Styl możemy wybrać za pomocą polecenia **\theoremstyle**{}, które umieszczamy przed poleceniem **\newtheorem**. Parametrem polecenia jest nazwa stylu. Może to być styl zdefiniowany przez nas. Jak to zrobić, zostanie pokazane w dalszej części. Dopóki nie zmienimy stylu przez ponowne użycie **\theoremstyle**{}, wszystkie definiowane środowiska będą stosować ten ostatnio ustalony. Przykłady definiowania środowisk pokazano na listingu 8.1. Mają one charakter demonstracyjny i raczej w jednym dokumencie nie powinniśmy mieszać tak różnych sposobów numerowania.

Listing 8.1. Definiowanie środowisk z użyciem polecenia **\newtheorem**

```
1 \theoremstyle{definition}
2 \newtheorem{definicja}{Definicja}
3 \theoremstyle{plain}
4 \newtheorem{twierdzenie}{Twierdzenie}[chapter]
5 \newtheorem{wniosek}{Wniosek}[chapter]
6 \newtheorem{lemat}[twierdzenie]{Lemat}
7 \theoremstyle{remark}
8 \newtheorem{eksperyment}{Eksperyment}[section]
9 \newtheorem{problem}{Problem}
```

Definiowanie środowisk zaczęliśmy od wybrania stylu `definition`. Korzysta z niego tylko środowisko **definicja** (linia 2). Ta najprostsza forma definiowania środowiska (bez parametru opcjonalnego) oznacza numerację ciągłą w całym

dokumencie. Następnie zmieniony został styl na `plain` i zdefiniowaliśmy środowiska **`twierdzenie`** i **`wniosek`**. Tutaj numeracja będzie zależna od rozdziału, tj. np. trzecie twierdzenie w rozdziale 2 będzie miało numer 2.3. Z tego samego stylu korzysta środowisko **`lemat`** (linia 6), ale parametr opcjonalny umieszczony między obowiązkowymi wskazuje, że środowisko to nie ma własnego licznika, ale będzie go współdzielić ze środowiskiem **`twierdzenie`**. Kolejna zmiana stylu (linia 7) oznacza, że dwa ostatnie środowiska będą korzystać ze stylu `remark`. Numeracja eksperymentów w ramach sekcji oznacza, że dla klas `book` i `report` ten numer będzie składał się z trzech liczb, np. 2.1.7 (drugi rozdział, pierwszy podrozdział, siódmy eksperyment). Ostatnie środowisko (linia 9) stosuje numerację ciągłą. Przykłady użycia zdefiniowanych środowisk pokazano na listingu 8.2.

Listing 8.2. Przykłady użycia środowisk z listingu 8.1

```
1  \begin{definicja}
2  \label{def:wektor}
3  \emph{Wektorem} o początku $P$ i końcu $Q$ nazywamy uporządkowaną parę
4  punktów $P$ i $Q$ i~oznaczamy $\overrightarrow{PQ}$.
5  \end{definicja}
6
7  \begin{definicja}
8  \label{def:wspolrzedne:wektora}
9  \emph{Współrzędnymi} wektora $\mathbf{v} = \overrightarrow{PQ}$ o
10 początku $P=(x_P,y_P,z_P)$ i końcu $Q=(x_Q,y_Q,z_Q)$ nazywamy liczby
11 $x_v = x_Q - x_P$, $y_v = y_Q - y_P$ i $z_v = z_Q - z_P$.
12 \end{definicja}
13
14 \begin{lemat}
15 \label{lm:iloczyn:skalary:wektor1}
16 $\alpha(\beta\mathbf{v}) = \beta(\alpha\mathbf{v})
17  = (\alpha\beta)\mathbf{v}$
18 \end{lemat}
19
20 \begin{lemat}
21 \label{lm:iloczyn:skalary:wektor2}
22 $(\alpha + \beta)\mathbf{v} = \alpha\mathbf{v} + \beta\mathbf{v}$
23 \end{lemat}
24
25 \begin{twierdzenie}
26 \label{tw:iloczyn:skalar:wektory}
27 Niech $\alpha \in \mathbb{R}$ i niech $\mathbf{u}, \mathbf{v}$ będą
28 \emph{wektorami}. Wtedy:\\$\alpha(\mathbf{u} + \mathbf{v})
29 = \alpha\mathbf{u} + \alpha\mathbf{v}$
30 \end{twierdzenie}
```

```
31
32 \begin{eksperyment}
33 Na podstawie lematu~\ref{lm:iloczyn:skalary:wektor1}:\\
34 $2 \cdot (3 \cdot [7,0,-1]) = 2 \cdot [21,0,-3] = [42,0,-6]$\\
35 $3 \cdot (2 \cdot [7,0,-1]) = 3 \cdot [14,0,-2] = [42,0,-6]$\\
36 $(2 \cdot 3) \cdot [7,0,-1] = 6 \cdot [7,0,-1] = [42,0,-6]$
37 \end{eksperyment}
```

Definicja 1. *Wektorem* o początku P i końcu Q nazywamy uporządkowaną parę punktów P i Q i oznaczamy $\overrightarrow{PQ}$.

Definicja 2. *Współrzędnymi* wektora $\mathbf{v} = \overrightarrow{PQ}$ o początku $P = (x_P, y_P, z_P)$ i końcu $Q = (x_Q, y_Q, z_Q)$ nazywamy liczby $x_v = x_Q - x_P$, $y_v = y_Q - y_P$ i $z_v = z_Q - z_P$.

Lemat 8.1. $\alpha(\beta\mathbf{v}) = \beta(\alpha\mathbf{v}) = (\alpha\beta)\mathbf{v}$

Lemat 8.2. $(\alpha + \beta)\mathbf{v} = \alpha\mathbf{v} + \beta\mathbf{v}$

Twierdzenie 8.3. *Niech* $\alpha \in \mathbb{R}$ *i niech* $\mathbf{u}, \mathbf{v}$ *będą* wektorami. *Wtedy:*
$\alpha(\mathbf{u} + \mathbf{v}) = \alpha\mathbf{u} + \alpha\mathbf{v}$

Eksperyment 8.1.1. Na podstawie lematu 8.1:
$2 \cdot (3 \cdot [7,0,-1]) = 2 \cdot [21,0,-3] = [42,0,-6]$
$3 \cdot (2 \cdot [7,0,-1]) = 3 \cdot [14,0,-2] = [42,0,-6]$
$(2 \cdot 3) \cdot [7,0,-1] = 6 \cdot [7,0,-1] = [42,0,-6]$

Analizując wynik kompilacji kodu z listingu 8.2, łatwo zauważyć, że lematy i twierdzenia rzeczywiście współdzielą licznik. Widać też różnice w składzie pomiędzy wszystkimi trzema stylami. Warto jeszcze zwrócić uwagę na polecenie `\emph{}`. Jego zadaniem jest wyróżnienie swojego parametru z otaczającego go tekstu. Użyte w liniach 3 i 9 powoduje złożenie słów „Wektorem" i „Współrzędnymi" kursywą, bo tekst wokół złożony jest prostą czcionką, zaś użyte w linii 28 powoduje złożenie słowa „wektorami" prostą czcionką, bo otaczający tekst jest złożony kursywą. Każde z podanych środowisk może przyjąć opcjonalny parametr, w którym możemy przekazać tekst do umieszczenia w nawiasie przy etykiecie, np. `\begin{twierdzenie}[Twierdzenie Banacha o punkcie stałym]` wygeneruje: **Twierdzenie 8.4** (Twierdzenie Banacha o punkcie stałym)**.**

Jeżeli żaden z predefiniowanych stylów nam nie odpowiada, możemy użyć polecenia `\newtheoremstyle`, aby zdefiniować własny. Polecenie to przyjmuje aż 9 parametrów. Ich znaczenie jest następujące: 1) nazwa stylu, 2) odstęp przed środowiskiem, 3) odstęp po środowisku, 4) sposób formatowania treści, 5) wielkość wcięcia, 6) sposób formatowania etykiety, 7) separator między etykietą i treścią, 8) odstęp między separatorem i treścią, 9) specyfikacja etykiety (można zostawić pustą, aby użyć domyślnej). Przykład definicji i użycia własnego stylu

pokazano na listingu 8.3. Podstawowa różnica względem stylów predefiniowanych to użycie kapitalików do złożenia etykiety środowiska.

Listing 8.3. Przykład definicji i zastosowania własnego stylu

```
\newtheoremstyle{scstyle}{3pt}{3pt}{\itshape}{0pt}{\scshape}{.}{ }{}
\theoremstyle{scstyle}
\newtheorem{wniosek}{Wniosek}[chapter]

\begin{wniosek}
\label{wn:dlugosc:wektora}
Jeśli $\mathbf{u} = \alpha\mathbf{v}$, to $|\mathbf{u}| =
|\alpha||\mathbf{v}|$.
\end{wniosek}
```

WNIOSEK 8.1. *Jeśli* $\mathbf{u} = \alpha\mathbf{v}$, *to* $|\mathbf{u}| = |\alpha||\mathbf{v}|$.

Pakiet `amsthm` udostępnia środowisko **`proof`** do składania dowodów. Automatycznie dodaje ono symbol końca dowodu (pusty kwadrat po prawej stronie). Przykład użycia środowiska pokazano na listingu 8.4 (dowód wniosku 8.1).

Listing 8.4. Przykład złożenia dowodu

```
\begin{proof}
Niech $\mathbf{v} = [x,y,z]$. Z założenia
$\mathbf{u} = \alpha\mathbf{v}$ wynika, że
$\mathbf{u} = [\alpha x,\alpha y,\alpha z]$.\\
Obliczamy długość wektora $\mathbf{u}$:
\begin{equation}
\begin{split}
|\mathbf{u}| &= \sqrt{(\alpha x)^2 + (\alpha y)^2 + (\alpha z)^2} =
\sqrt{\alpha^2(x^2 + y^2 + z^2)} = \\[1.5mm]
&= |\alpha|\sqrt{x^2 + y^2 + z^2} = |\alpha||\mathbf{v}|
\end{split}
\end{equation}
co należało udowodnić.
\end{proof}
```

Dowód. Niech $\mathbf{v} = [x, y, z]$. Z założenia $\mathbf{u} = \alpha\mathbf{v}$ wynika, że $\mathbf{u} = [\alpha x, \alpha y, \alpha z]$. Obliczamy długość wektora $\mathbf{u}$:

$$\begin{aligned} |\mathbf{u}| &= \sqrt{(\alpha x)^2 + (\alpha y)^2 + (\alpha z)^2} = \sqrt{\alpha^2(x^2 + y^2 + z^2)} = \\ &= |\alpha|\sqrt{x^2 + y^2 + z^2} = |\alpha||\mathbf{v}| \end{aligned} \tag{8.1}$$

co należało udowodnić. □

8.2 Listingi

Jeżeli potrzebujemy dołączyć do dokumentu fragment kodu źródłowego w jakimś języku programowania albo zapis sesji prowadzonej na terminalu komputerowym, to nie możemy użyć do tego zwykłego tekstu. Nawet użycie polecenia **\ttfamily**, aby uzyskać tekst o stałej szerokości znaków typowy dla terminali lub edytorów kodu źródłowego, nie rozwiązuje problemu, bo LaTeX interpretuje zawartość i nie traktuje takiego tekstu dosłownie. Rozwiązaniem tego problemu może być zastosowanie środowiska **verbatim**. Tekst umieszczony wewnątrz tego środowiska jest drukowany za pomocą ustawionej czcionki o stałej szerokości dokładnie tak, jak go wprowadziliśmy, tj. jeśli w kodzie znajduje się znak $, to jest drukowany znak $ i nie jest traktowany jak przejście do trybu matematycznego, jeżeli umieściliśmy dwie spacje, to są drukowane dwie spacje itd.

Uzupełnieniem dla środowiska **verbatim** jest polecenie **\verb**, które pozwala umieścić dosłownie mały fragment tekstu w trybie wierszowym (zawartość nie jest łamana). Polecenie to przyjmuje jeden parametr, ale ogranicznikiem dla parametru jest powtórzony ten sam symbol. Najczęściej jest to pionowa kreska (|) lub wykrzyknik, ważne, aby znak ten nie występował wewnątrz tekstu będącego parametrem. Przykład użycia środowiska **verbatim** i polecenia **\verb** pokazano na listingu 8.5. Załóżmy, że chcemy podać przykład i opisać działanie funkcji w języku Python.

Listing 8.5. Przykład dosłownego drukowania tekstu

```
\begin{verbatim}
def disjoint(a,b):
    for i in a:
        if i in b:
            return False
    return True
\end{verbatim}
Funkcja \verb|disjoint| sprawdza, czy dwie listy są rozłączne.
W ramach pętli \verb!for! przeglądamy wszystkie elementy z listy
\verb|a| i sprawdzamy, czy należą również do listy \verb|b|.
Jeśli znajdziemy choć jeden taki przypadek, zwracamy \verb!False!.
```

```
def disjoint(a,b):
    for i in a:
        if i in b:
            return False
    return True
```

Funkcja `disjoint` sprawdza, czy dwie listy są rozłączne. W ramach pętli `for` przeglądamy wszystkie elementy z listy `a` i sprawdzamy, czy należą również do listy `b`. Jeśli znajdziemy choć jeden taki przypadek, zwracamy `False`.

W podanym przykładzie widzimy liczne dla Pythona wcięcia, które zostały zachowane dokładnie tak, jak było w kodzie źródłowym. Jeśli zamiast środowiska **`verbatim`** użyjemy **`verbatim`**∗, to dodatkowo zobaczymy każdą spację wyświetloną jako znak ␣. Analogicznie zachowuje się polecenie `\verb`∗. Dodatkowo w podanym przykładzie kilka razy użyliśmy polecenia `\verb` z dwoma różnymi ogranicznikami, aby wyróżnić słowa pochodzące z listingu.

```
\usepackage{verbatim}
```

Pakiet `verbatim` dostarcza poprawioną wersję środowiska **`verbatim`**. Ma to znaczenie, szczególnie gdy chcemy użyć tego środowiska do dołączania długich listingów. Ciekawszą cechą tego pakietu jest środowisko **`comment`**, które pozwala na definiowanie blokowych komentarzy. Możemy dzięki niemu łatwo zamienić w komentarz nawet duży fragment dokumentu bez potrzeby wstawiania znaku % w każdej linii.

Pakiet `verbatim` jest wystarczający, jeżeli nie potrzebujemy w żaden sposób różnicować elementów wyświetlanego w sposób dosłowny tekstu. Jeżeli chcemy uzyskać efekt podobny do podświetlania składni w edytorach tekstu, mieć możliwość sterowania odstępami przed i po listingu, zamienić listingi we wstawki, to rozwiązaniem jest pakiet `listings` [24]. Przykłady kodu w LaTeX-u prezentowane w tej książce zostały złożone właśnie z użyciem tego pakietu.

```
\usepackage{listings}
\lstloadlanguages{Python}
```

Poza dołączeniem pakietu `listings` w preambule powinniśmy określić, z jakich języków chcemy korzystać. W prezentowanych przykładach ograniczymy się do języka Python. W dokumentacji pakietu [24] znajduje się tabela zawierająca nazwy prawie stu wspieranych języków programowania. Jeżeli nie znajdziemy w niej nazwy języka, której szukamy, to pakiet umożliwia również zdefiniowanie obsługi nowego języka. Wskazanie języka (lub języków – nazwy oddzielamy przecinkami) realizowane jest przez polecenie **`\lstloadlanguages`**. Jest to o tyle istotne, że podobnie jak system podświetlania składni w edytorach kodu, pakiet `listings` analizuje zawartość listingów i pozwala wyróżniać np. słowa kluczowe danego języka.

Pakiet oferuje środowisko **lstlisting** jako zamiennik dla środowiska **verbatim** i polecenie **\lstinline** jako zamiennik polecenia **\verb** (i na tej samej zasadzie funkcjonują ograniczniki dla tego polecenia). Dodatkowo dostępne jest polecenie **\lstinputlisting**, które pozwala na pobranie i złożenie kodu znajdującego się w zewnętrznym pliku.

Sposób, w jaki wyświetlane są listingi, zależy od naszych ustawień, które mogą być wielokrotnie zmieniane w dokumencie. Ustawienia są definiowane przez polecenie **\lstset**{}, którego parametrem jest lista opcji typu `klucz=wartość`. Przykład ustawień i ich wpływ na skład kodu pokazano na listingu 8.6 (porównaj efekt z listingiem 8.5).

Listing 8.6. Skład kodu źródłowego z użyciem pakietu `listings`

```
1  \lstset{language=Python,
2    basicstyle=\ttfamily\small,
3  }
4  \begin{lstlisting}
5  def disjoint(a,b):
6      for i in a:
7          if i in b:
8              return False
9      return True
10 \end{lstlisting}
11 Funkcja \lstinline|disjoint| sprawdza, czy dwie listy są rozłączne.
12 W ramach pętli \lstinline!for! przeglądamy wszystkie elementy z listy
13 \lstinline|a| i sprawdzamy, czy należą również do listy \lstinline|b|.
14 Jeśli znajdziemy choć jeden taki przypadek, zwracamy \lstinline!False!.
```

```
def disjoint(a,b):
    for i in a:
        if i in b:
            return False
    return True
```

Funkcja `disjoint` sprawdza, czy dwie listy są rozłączne. W ramach pętli **for** przeglądamy wszystkie elementy z listy `a` i sprawdzamy, czy należą również do listy `b`. Jeśli znajdziemy choć jeden taki przypadek, zwracamy `False`.

Kod zawarty w liniach 4–10 nie różni się niczym poza nazwą środowiska od kodu z listingu 8.5 (linie 1–7). W treści akapitu (linie 11–14) zmieniliśmy polecenie **\verb** na **\lstinline**. Efekt kompilacji tego kodu jest inny niż poprzednio. Kluczowe są ustawienia w liniach 1–2. Poza wskazaniem języka (linia 1) określiliśmy tu podstawową czcionkę dla listingu (linia 2). Ustawienia te są ważne dla tego listingu, ale również dla kolejnych i składu za pomocą polecenia

\lstinline (i **\lstloadlanguages**). W tym przypadku skorzystaliśmy z ustawień domyślnych opcji formatowania (poza `basicstyle`), np. pogrubione zostały rozpoznane słowa kluczowe. Możemy dopasować opcje do własnych potrzeb. Najważniejsze zostały zebrane w tabeli 8.2.

Tabela 8.2. Wybrane opcje polecenia **\lstset**

Opcja	Znaczenie
`language`	nazwa języka z ewentualnym dialektem, np. `[ANSI]C`
`basicstyle`	podstawowy styl do składania listingu
`identifierstyle`	styl dla identyfikatorów
`commentstyle`	styl dla komentarzy (domyślnie **\itshape**)
`keywordstyle`	styl dla słów kluczowych (domyślnie **\bfseries**)
`stringstyle`	styl dla łańcuchów znaków
`numberstyle`	styl dla numerowania linii
`numbers`	sposób umieszczenia: `none` (domyślnie), `left`, `right`
`stepnumber`	krok numerowania (domyślnie 1)
`numbersep`	odległość między numerami i kodem (domyślnie 10pt)
`captionpos`	sposób umieszczania podpisów (`t` – nad, `b` – pod)
`breaklines`	automatyczne zawijanie długich linii (`true` lub `false`)
`aboveskip`	odstęp przed listingiem
`belowskip`	odstęp po listingu
`abovecaptionskip`	odstęp przed podpisem
`belowcaptionskip`	odstęp po podpisie
`frame`	ramka (domyślnie brak)
`frameround`	zaokrąglanie narożników (domyślnie brak)
`escapechar`	znak ucieczki do LaTeX-a
`morekeywords`	lista dodatkowych słów kluczowych
`otherkeywords`	lista dodatkowych specjalnych słów kluczowych
`xleftmargin`	dodatkowy margines po lewej stronie
`xrightmargin`	dodatkowy margines po prawej stronie
`rulesep`	odległość między liniami podwójnej ramki
`framerule`	grubość linii ramki (tylko dla ostrych narożników)
`rulecolor`	kolor linii ramki

Wartość przypisana do opcji `basicstyle` nie może na końcu zawierać polecenia, które przetwarza występujące po nim znaki, np. nie można użyć **\underline**, co jest możliwe dla innych opcji.

Ramkę możemy zdefiniować, podając sekwencję liter określających, po której stronie ma być rysowana: `t` – góra, `r` – prawa strona, `b` – dół, `l` – lewa strona.

Jeżeli użyjemy wielkiej litery, to ramka będzie podwójna, np. `frame=TB` definiuje podwójną linię od góry i od dołu listingu. Zaokrąglenia narożników definiowane są jako sekwencja czterech liter `t` – zaokrąglony – lub `f` – ostry, które podajemy w kolejności: prawy górny, prawy dolny, lewy dolny i lewy górny, np. `frameround=tftf` spowoduje zaokrąglenie prawego górnego i lewego dolnego narożnika, przy założeniu, że kreślona jest kompletna ramka (np. dla ustawienia ramki `frame=TB` ustawienia narożników są ignorowane).

Nawet jeśli załadujemy potrzebny język, to może się okazać, że nie wszystkie słowa kluczowe są rozpoznawane, bo na przykład używamy rozszerzenia języka lub niestandardowych bibliotek. Za pomocą opcji `morekeywords` możemy dodać kolejne słowa do listy słów kluczowych. Z kolei opcja `otherkeywords` pozwala na dodanie słów kluczowych, których nazwy zaczynają się od cyfr lub zawierają niestandardowe znaki, np. możemy jako słowo kluczowe zdefiniować ciąg znaków `::=`.

Nie należy mylić opcji `morekeywords` z opcją `keywords`. Ta pierwsza dodaje słowa do listy, ta druga usuwa całą zawartość listy słów kluczowych i zastępuje swoim parametrem. Opcja `keywords` jest użyteczna, gdy definiujemy wsparcie nowego języka.

Na listingu 8.7 pokazano fragment kodu źródłowego, który zawiera słowa kluczowe, identyfikatory, łańcuchy znaków i komentarz. Można je łatwo zidentyfikować ze względu na różne formatowanie. Polecenie **`\lstset`** zostało zaplanowane tak, aby zademonstrować sporą część opcji z tabeli 8.2.

Listing 8.7. Przykład własnych ustawień składania kodu źródłowego

```
\lstset{language=Python,
  basicstyle=\ttfamily\small,
  identifierstyle=\underline,
  commentstyle=\itshape\color{gray},
  stringstyle=\color{black!70},
  numberstyle=\scriptsize,
  numbers=left,
  xleftmargin=5mm,
  xrightmargin=5mm,
  frame=tb,
  rulecolor=\color{gray},
  framerule=2pt,
}
\begin{lstlisting}
x = 'LaTeX'
print(x)
```

```
y = x[2:]                # y = 'TeX'
print(y)
print('I love ' + x)
\end{lstlisting}
```

```
x = 'LaTeX'
print(x)
y = x[2:]                # y = 'TeX'
print(y)
print('I␣love␣' + x)
```

Polecenia **\lstset** można użyć wielokrotnie w dokumencie. Nie zawsze trzeba od nowa ustawiać wszystkie opcje. Ustawiamy te opcje, których wartości chcemy nadpisać. Jeżeli robimy to na poziomie środowiska **document**, to ustawienia obowiązują dla wszystkich kolejnych listingów. Jeżeli chcemy wykonać zmianę ustawień dla danego listingu, to możemy to zrobić poprzez parametr opcjonalny środowiska **lstlisting**. W tabeli 8.3 zebrano przykładowe opcje, które mają zastosowanie wyłącznie dla konkretnego listingu. Ponadto możemy używać opcji z tabeli 8.2.

Tabela 8.3. Wybrane opcje do użycia z pojedynczymi listingami

Opcja	Znaczenie
`caption`	podpis dla numerowanych listingów
`title`	podpis dla nienumerowanych listingów
`label`	etykieta
`float`	listing jako obiekt pływający (podzbiór opcji `tbph`)
`firstline`	numer pierwszej linii do wyświetlenia
`lastline`	numer ostatniej linii do wyświetlenia
`emph`	lista słów do wyróżnienia
`emphstyle`	sposób wyróżniania słów wskazanych w opcji `emph`

Przykład ustawień z opcjami z tabeli 8.3 pokazano na listingu 8.8. Użycie opcji `float` oznacza, że ten listing będzie wstawką. Zdefiniowana etykieta pozwala na odwoływanie się do listingu za pomocą poleceń **\ref** i **\pageref**. Użycie opcji `caption` oznacza nie tylko zdefiniowanie podpisu dla listingu, ale również skutkuje nadaniem numeru. Jeżeli zamiast niej użyjemy opcji `title`, to listing będzie miał podpis, ale nie będzie miał przypisanego numeru. Jeżeli podpis jest wyjątkowo długi i dla spisu listingów chcemy zdefiniować krótką wersję, to stosujemy składnię `caption={[krótki podpis]oryginalny podpis}`.

Listing 8.8. Przykład użycia parametrów opcjonalnych dla listingu

```
\begin{lstlisting}[float=ht, label=lst:python, aboveskip=3mm,
belowskip=0mm, caption={Przykład definicji funkcji w języku Python}]
% ...
\end{lstlisting}
```

Wśród opcji dostępna jest możliwość zdefiniowania znaku ucieczki do LaTeX-a. Jeżeli użyjemy na przykład opcji `escapechar=!`, to tekst umieszczony między znakami `!` będzie interpretowany jak zwykły tekst LaTeX-a. Daje to nam możliwość umieszczenia w listingu na przykład wzoru matematycznego: `!$x^2$!`. Pakiet `listings` oferuje również opcję `literate`, która pozwala zdefiniować listę podmian, które będą wykonywane automatycznie. Wartością tej opcji jest ciąg trójek `{}{}{}` przedzielonych spacjami. Każda trójka to kolejno: *co zastępujemy*, *czym zastępujemy* i *ile znaków ma nowy tekst*. Tekst wstawiany może być dowolnym kodem LaTeX-a, w szczególności wzorem matematycznym. Przykład użycia opcji `literate` pokazano na listingu 8.9.

Listing 8.9. Przykład użycia automatycznych podmian – opcja `literate`

```
\begin{lstlisting}[literate={!=}{$\ne$}{1} {=}{$\gets$}{1}]
a = 278
b = 16
while a != b:
    if a > b:
        a = a-b
    else:
        b = b-a
print(a)
\end{lstlisting}

a ← 278
b ← 16
while a ≠ b:
    if a > b:
        a ← a-b
    else:
        b ← b-a
print(a)
```

W pokazanym przykładzie zdefiniowano dwie podmiany: para znaków `!=` zamieniana jest na symbol ≠ i operator przypisania `=` zamieniany jest na symbol ←. Gdybyśmy chcieli, aby takie podmiany wykonywane były we wszystkich listingach w dokumencie, to opcję `literate` należy umieścić wewnątrz polecenia `\lstset`. Opcja `literate` jest kluczowa, jeśli używamy kodowania `utf8` i chcemy używać na listingach liter ze znakami diakrytycznymi. Na dziś pakiet `listings`

ma problem z kodowaniem `utf8`. Niezależnie od języka rozwiązaniem jest zdefiniowanie listy podmian dla liter diakrytyzowanych. Lista podmian dla języka polskiego została pokazana na listingu 8.10. Dodanie jej w preambule pozwala bez problemu korzystać ze wszystkich polskich liter na listingach. Zobaczymy to w następnym przykładzie.

Listing 8.10. Lista podmian dla polskich liter diakrytyzowanych

```
\lstset{literate=
  {ą}{{\k{a}}}{1} {ć}{{\'c}}{1} {ę}{{\k{e}}}{1}
  {ó}{{\'o}}{1}   {ń}{{\'n}}{1} {ł}{{\l{}}}{1}
  {ś}{{\'s}}{1}   {ź}{{\'z}}{1} {ż}{{\.z}}{1}
  {Ą}{{\k{A}}}{1} {Ć}{{\'C}}{1} {Ę}{{\k{E}}}{1}
  {Ó}{{\'O}}{1}   {Ń}{{\'N}}{1} {Ł}{{\L{}}}{1}
  {Ś}{{\'S}}{1}   {Ź}{{\'Z}}{1} {Ż}{{\.Z}}{1},
}
```

Niezależnie od ustawień dotyczących wyświetlania słów kluczowych, identyfikatorów itd. możemy chcieć zwrócić uwagę na jedno czy drugie słowo, które z jakichś powodów jest kluczowe dla załączanego fragmentu kodu. Pakiet `listings` udostępnia w tym celu opcje `emph` i `emphstyle`. Pierwsza przyjmuje listę słów, które mają być wyróżnione, a druga definiuje sposób tego wyróżnienia. Dodatkowo możemy definiować wiele klas (grup) słów i każdą z klas wyróżnić w inny sposób, na przykład dla kodu z listingu 8.6 możemy określić ustawienia: `emph={disjoint}, emphstyle=\uline, emph={[2]True,False}, emphstyle={[2]\dashuline}`. W tej sytuacji słowo `disjoint` zostanie podkreślone linią ciągłą, a słowa `True` i `False` linią przerywaną (oba polecenia pochodzą z pakietu `ulem`). Efekt tych ustawień można zobaczyć na listingu 3.18. Jeżeli mamy taką możliwość, to warto wyróżniać słowa różnymi kolorami.

```
\usepackage{tcolorbox}
\usepackage{ulem}
\tcbuselibrary{listings}
```

Pakiet `listings` oferuje możliwość dodawania ramek do listingów (zob. listing 8.7), ale dużo większe możliwości mamy, gdy połączymy pakiet `listings` z pakietem `tcolorbox` (zob. podrozdział 3.5). Ten drugi ma m.in. zdefiniowane środowisko **`tcblisting`** przeznaczone do składania listingów (domyślnie jest to kod w LaTeX-u, gdzie jedna część ramki zawiera kod źródłowy, a druga jego wersję skompilowaną). Przy definiowaniu ustawień mamy do dyspozycji zarówno opcje związane z wyglądem ramek pakietu `tcolorbox`, jak i opcje dotyczące

formatowania kodu źródłowego z pakietu `listings`. Ważne są również wcześniejsze ustawienia zrealizowane za pomocą polecenia **`\lstset`** (oczywiście nie powinniśmy definiować ustawień dla ramek czy narożników, bo wizualizację realizujemy za pomocą ramki z pakietu `tcolorbox`).

Listing 8.11. Skład listingu w środowisku **`tcblisting`**

```
\begin{tcblisting}{colback=gray!10, colframe=gray, listing only,
title=Funkcja disjoint, fonttitle=\bfseries,
listing options={language=Python, basicstyle=\ttfamily\small,
emph={disjoint}, emphstyle=\uline, emph={[2]True,False},
emphstyle={[2]\dashuline}}}
def disjoint(a,b):
    for i in a:         # przeglądamy wszystkie elementy listy a
        if i in b:      # sprawdzamy, czy należą do listy b
            return False
    return True
\end{tcblisting}
```

Funkcja disjoint

```
def disjoint(a,b):
    for i in a:         # przeglądamy wszystkie elementy listy a
        if i in b:      # sprawdzamy, czy należą do listy b
            return False
    return True
```

W liniach 1–2 mamy ustawienia typowe dla ramek z pakietu `tcolorbox` z wyjątkiem nowej opcji `listing only`, która wskazuje, że ramka zawiera tylko część z kodem źródłowym. W liniach 3–5 mamy opcję `listing options`, która pozwala na przekazanie opcji związanych ze składaniem listingu. Zastosowane zostały opisane powyżej ustawienia dotyczące wyróżnień. W efekcie widzimy listing umieszczony wewnątrz eleganckiej ramki. Po załączeniu kodu z listingu 8.10 poprawnie są wyświetlane polskie litery użyte w komentarzach.

Drugi przykład użycia środowiska **`tcblisting`** pokazano na listingu 8.12. Przy odpowiednich ustawieniach kolorystycznych ramka wygląda jak typowy terminal. Dodatkowo opcja `every listing line` wstawia tutaj tekst wyglądający jak typowy znak zachęty z terminala.

Listing 8.12. Listing symulujący terminal

```
\begin{tcblisting}{colback=black!70,colframe=black!70,colupper=white,
listing only,every listing line={\textcolor{gray!50}
  {\small\ttfamily\bfseries \$> }}}
pdflatex main.tex
```

```
biblex main
pdflatex main.tex
pdflatex main.tex
\end{tcblisting}
```

```
$> pdflatex main.tex
$> biblex main
$> pdflatex main.tex
$> pdflatex main.tex
```

Jeżeli chcemy na listingach umieszczać kod w języku, który nie jest wspierany przez pakiet `listings`, to możemy łatwo takie wsparcie zdefiniować. Przykład takiej definicji opracowanej na potrzeby składania książki [45] pokazano na listingu 8.13.

Listing 8.13. Przykład definicji wsparcia dla nowego języka

```
1  \lstdefinelanguage{Alvis}{
2    keywords={agent, in, out, delay, jump, exec, alt, data, type,
3              critical, start, exit, far, loop, if, else, elseif,
4              select, cli, sti, proc, elseif, every, environment, null},
5    keywords=[2]{Char, Bool ,Int, Double, String, rem, sqrt, head, tail,
6                 signal, durable, queue},
7    sensitive=true,
8    morecomment=[l]{--},
9    morecomment=[s]{/*}{*/},
10   morestring=[b]",
11 }
```

Zadanie realizowane jest za pomocą polecenia **`\lstdefinelanguage`**. Zaczynamy od podania nazwy języka (linia 1) i zdefiniowania słów kluczowych (linie 2–6). W podanych przykładzie podzielono je na dwie klasy. Opcja `sensitive` ustawiona na `true` wskazuje, że język jest czuły na wielkość liter, tj. np. `int` nie zostanie uznane za słowo kluczowe, a `Int` tak. Linie 8 i 9 to definicje dwóch typów komentarzy, od znaków `--` zaczyna się komentarz rozciągający się do końca linii, a między znakami `/*` i `*/` możemy umieścić komentarz obejmujący wiele linii. Ostatnia opcja (linia 10) wskazuje, że łańcuchy znaków umieszczamy w podwójnych cudzysłowach. Na listingu 8.14 pokazano fragment kodu w języku Alvis. Widać, że pakiet `listings` bez problemu radzi sobie z rozpoznawaniem słów kluczowych i komentarzy.

Listing 8.14. Ustawienia składania listingów i fragment kodu w języku Alvis

```
1  \lstset{language=Alvis,
2    basicstyle=\ttfamily\small,
```

```
  keywordstyle=\bfseries,
  keywordstyle=[2]\bfseries\color{gray},
  commentstyle=\itshape\color{gray},
  numberstyle=\tiny\bfseries\color{gray},
  numbers=left,
}

\begin{lstlisting}
agent Buffer {
  i :: Int = 0;   -- parametr typu całkowitego
  proc (i == 0) put { in put; i = 1; }
  proc (i /= 0) get { out get; i = 0; }}

/* agent Receiver {
  loop { in get; }} */
\end{lstlisting}
```

```
agent Buffer {
  i :: Int = 0;   -- parametr typu całkowitego
  proc (i == 0) put { in put; i = 1; }
  proc (i /= 0) get { out get; i = 0; }}

/* agent Receiver {
  loop { in get; }} */
```

8.3 Składanie algorytmów w pseudokodzie

Termin pseudokod odnosimy do sposobu zapisu algorytmów, w którym zrezygnowano z niektórych elementów składniowych typowych dla języków programowania na rzecz przejrzystości i czytelności kodu, w tym możliwości korzystania z wyrażeń matematycznych opisujących realizowane przekształcenia oraz przedstawienia samej procedury na różnych poziomach abstrakcji w zależności od potrzeb. Przedstawione w tym rozdziale metody reprezentowania algorytmów nie są wyłącznie domeną informatyki. Wiele dyscyplin wykorzystuje obecnie obszary związane z eksploracją danych, uczeniem maszynowym itp., więc podstawowa znajomość możliwości LaTeX-a w tym temacie może być cenna.

```
\usepackage{algpseudocodex}
\usepackage[chapter]{algorithm}
```

Do składania algorytmów skorzystamy z pakietu `algpseudocodex` [36]. Algorytm składamy w środowisku **`algorithmic`**. Parametrem opcjonalnym środowiska jest liczba wskazująca sposób numerowania linii. Brak parametru lub podanie wartości 0 oznacza brak numeracji. Podanie wartości $n > 0$ oznacza, że numerowana będzie każda n-ta linia. Na listingu 8.15 przedstawiono popularny algorytm klasteryzacji metodą k-średnich.

Listing 8.15. Pseudokod dla klasteryzacji metodą k-średnich

```
1  \begin{algorithmic}[1]
2  \Require $k$ -- liczba klastrów, $n$ -- liczba obiektów
3  \Ensure zbiór $k$ klastrów $C_1,\dots,C_k$
4  \State wybierz losowo $k$ obiektów jako początkowe centroidy
5  $c_1,\dots,c_k$
6  \Repeat
7    \For{${x} \in \{{x}_1, {x}_2, \dots, {x}_n\}$}
8      \State przypisz ${x}$ do klastra najbliższego centroidu
9    \EndFor
10   \For{$i \in \{1,2,\dots,k\}$}
11     \State przelicz centroid ${c}_i$ klastra $C_i$
12   \EndFor
13 \Until{centroidy nie uległy zmianie}
14 \end{algorithmic}
```

Require: k – liczba klastrów, n – liczba obiektów
Ensure: zbiór k klastrów $C_1, \dots, C_k$
1: wybierz losowo k obiektów jako początkowe centroidy $c_1, \dots, c_k$
2: **repeat**
3: | **for** $x \in \{x_1, x_2, \dots, x_n\}$ **do**
4: | | przypisz x do klastra najbliższego centroidu
5: | **for** $i \in \{1, 2, \dots, k\}$ **do**
6: | | przelicz centroid c_i klastra C_i
7: **until** centroidy nie uległy zmianie

Polecenie **`\Require`** (linia 2) pozwala określić warunki początkowe algorytmu. Zostało użyte do wskazania, co dostarczamy na wejściu. Polecenie jest bezparametrowe. Bezpośrednio po nim podajemy po prostu tekst, który chcemy umieścić w linii **Require:**. Analogicznie działa polecenie **`\Ensure`** (linia 3), tyle że opisujemy warunki dla wyniku. Tutaj użyto go, aby określić, co jest wynikiem algorytmu. Najczęściej używanym poleceniem, które opisuje pojedynczy krok w algorytmie, jest **`\State`** (linie 4–5, 8, 11). Generuje ono pojedynczą linię w opisie algorytmu zawierającą podany tekst. Jeżeli jest użyte wewnątrz na przykład instrukcji pętli, to linia ta jest odpowiednio wcięta. Para poleceń

\Repeat, **\Until**{} (linie 6 i 13) odpowiada za wygenerowanie pętli typu *repeat until*. Parametrem polecenia **\Until** jest warunek zakończenia pętli. Para poleceń **\For**{}, **\EndFor** (linie 7 i 9 oraz 10 i 12) odpowiada za wygenerowanie pętli typu *for*. Parametrem polecenia **\For** jest opis, który powinien określać zmienną (iterator pętli) i zakres przyjmowanych przez nią wartości. Na koniec warto dodać, że wcięcia użyte w kodzie źródłowym nie są niezbędne, użyto ich dla zwiększenia czytelności kodu dla czytelnika. Pakiet sam zamieści odpowiednie wcięcia, niezależnie od tego, czy ich użyjemy, czy nie.

W tabeli 8.4 zebrano podstawowe polecenia, których możemy użyć do składania algorytmów. Dla polecenia **\Comment** parametrem jest komentarz, dla poleceń **\While** i **\Until** jest to warunek pętli, dla **\For**{} i **\ForAll**{} – iterator pętli i zakres przyjmowanych wartości, dla **\If** i **\ElsIf** – warunek logiczny, a dla **\Procedure**, **\Function** i **\Call** – kolejno nazwa i parametry procedury (funkcji).

Tabela 8.4. Podstawowe polecenia do składania algorytmów

Polecenia	Opis
\State	instrukcja
\Statex	kontynuacja instrukcji (brak numeru linii)
\Require	warunki początkowe (wejście)
\Ensure	warunki końcowe (wyjście)
\Comment{}	komentarz
\While{}, **\EndWhile**	pętla *while*
\For{}, **\EndFor**	pętla *for*
\ForAll{}, **\EndFor**	pętla *for all*
\Loop, **\EndLoop**	pętla (uniwersalna)
\Repeat, **\Until**{}	pętla *repeat until*
\If{}, **\ElsIf**{}, **Else**, **\EndIf**	instrukcja wyboru
\Procedure{}{}, **\EndProcedure**	procedura
\Function{}{}, **\EndFunction**	procedura
\State\Return	zwrócenie wartości
\State\Output	zwrócenie wartości
\State\Call{}{}	wywołanie procedury

Pakiet zawiera również konstrukcję **\BeginBox**, **\EndBox**, której rolą jest opakowanie fragmentu algorytmu w ramkę, którą może reprezentować obramowanie (linia ciągła, przerywana itd.) i/lub wybrany kolor tła. Przykład zastosowania tej konstrukcji pokazano na listingu 8.16, który przedstawia znany algorytm wyznaczania największego wspólnego dzielnika dla dwóch liczb. Ramki są rysowane z użyciem pakietu `tikz`, więc można używać typowych dla tego pakietu wartości opcji.

Listing 8.16. Pseudokod dla algorytmu Euklidesa

```
1  \begin{algorithmic}[1]
2  \Require $a, b \in \mathbb{N}$
3  \Ensure NWD $a,b$\Comment{Największy Wspólny Dzielnik a i b}
4  \While{$a \ne b$}
5    \BeginBox[fill=gray!30]
6      \If{$a>b$}
7        \State $a\gets a-b$
8      \Else
9        \State $b\gets b-a$
10     \EndIf
11   \EndBox
12 \EndWhile
13 \State\Return $a$
14 \end{algorithmic}
```

Require: $a, b \in \mathbb{N}$
Ensure: NWD a, b ▷ *Największy Wspólny Dzielnik a i b*
1: **while** $a \neq b$ **do**
2: **if** $a > b$ **then**
3: $a \leftarrow a - b$
4: **else**
5: $b \leftarrow b - a$
6: **return** a

W linii 3 skorzystaliśmy z polecenia `\Comment`. Komentarze są domyślnie dosuwane do prawej strony i składane w szarym kolorze. Na początek komentarza wskazuje mały trójkąt. Dodatkowo instrukcja wyboru została opakowana instrukcjami `\BeginBox` i `\EndBox` z opcją zamalowania tła szarym kolorem. W ten sposób wyróżniliśmy kluczowy fragment tego algorytmu.

Zdarza się, że po dodaniu poleceń `\BeginBox` i `\EndBox` i kompilacji ramka nie jest poprawnie wyświetlana. W takim przypadku trzeba jeszcze raz skompilować dokument.

Pakiet `algpseudocodex` pozwala na swobodną personalizację poleceń poprzez redefiniowanie słów kluczowych. Powiedzmy, że chcielibyśmy zmienić słowa generowane przez polecenia `\Require` i `\Ensure` na „Wejście” i „Wyjście”. W tym celu musimy zredefiniować polecenia `\algorithmicrequire` i `\algorithmicensure` (podobne polecenia definiują inne słowa kluczowe, pełne zestawienie znajduje się w dokumentacji pakietu [36]). Zmiana słowa

kluczowego jest realizowana za pomocą polecenia **\algrenewcommand**. Przykład pokazano na listingu 8.17. Zmiany słów kluczowych realizuje kod w liniach 1–2.

Listing 8.17. Pseudokod dla algorytmu Euklidesa

```
1 \algrenewcommand\algorithmicrequire{\bfseries Wejście:}
2 \algrenewcommand\algorithmicensure{\bfseries Wyjście:}
3
4 \begin{algorithmic}[1]
5 \Require $a, b \in \mathbb{N}$
6 %...
```

Wejście: $a, b \in \mathbb{N}$
Wyjście: NWD a, b ▷ *Największy Wspólny Dzielnik a i b*

1: **while** $a \neq b$ **do**
2: **if** $a > b$ **then**
3: $a \leftarrow a - b$
4: **else**
5: $b \leftarrow b - a$
6: **return** a

Algorytmy mogą być wstawkami, mieć przypisany podpis i etykietę, która pozwala się odwoływać do numeru algorytmu. W tym celu środowisko **algorithmic** należy obudować środowiskiem **algorithm** z pakietu `algorithm`. Przykładowy kod pokazano na listingu 8.18.

Listing 8.18. Przykład użycia środowiska **algorithm**

```
1  \floatname{algorithm}{Algorytm}     % w preambule
2
3  \begin{algorithm}[ht]
4  \caption{Hierarchiczny aglomeracyjny algorytm grupowania}
5  \label{alg:grupowanie:hierarchiczne}
6  \begin{algorithmic}[1]
7  \Require zbiór $n$ obiektów
8  \Ensure dendrogram
9  \State wyznacz macierz odległości między klastrami
10 \Repeat
11   \State scal dwa najbliższe klastry
12   \State zaktualizuj macierz
13   \Statex\Comment{aby odzwierciedlić odległości między nowym
14     klastrem\\ a oryginalnymi klastrami}
15 \Until{pozostał jeden klaster}
16 \end{algorithmic}
17 \end{algorithm}
```

Algorithm 8.1. Hierarchiczny aglomeracyjny algorytm grupowania

Require: zbiór n obiektów

Ensure: dendrogram

1: wyznacz macierz odległości między klastrami
2: **repeat**
3: scal dwa najbliższe klastry
4: zaktualizuj macierz
▷ *aby odzwierciedlić odległości między nowym klastrem a oryginalnymi klastrami*
5: **until** pozostał jeden klaster

Kod z linii 1 należy umieścić w preambule. Jest to redefinicja (tutaj spolszczenie) etykiety dla środowiska **`algorithm`**. Jeżeli składamy dokument w języku angielskim, to to polecenie pomijamy (chyba że chcemy zmienić nazwę etykiety środowiska na np. „Procedure”). Początek środowiska mamy w linii 3. Parametr opcjonalny jest analogiczny jak w przypadku środowiska **`figure`** i **`table`**. Wskazujemy potencjalne miejsca ulokowania naszej wstawki. Podobnie jak w przypadku tabel napisy dla algorytmów umieszcza się przed pseudokodem. Bezpośrednio po poleceniu **`\caption`** powinniśmy zdefiniować etykietę.

Dołączenie pakietu `algorithm` bez żadnych dodatkowych opcji powoduje ciągłą numerację algorytmów w dokumencie. Pakiet można dodać z opcją `chapter`, która spowoduje numerowanie algorytmów w ramach rozdziałów (można użyć innych opcji, jak `part`, `section` itd.). Jeżeli chcemy dodać kropkę po numerze algorytmu, to można skorzystać z pakietu `caption` (zob. podrozdział 7.4). W tym przypadku w preambule można dodać następujący kod (dopasowany do innych podpisów w tej książce): **`\captionsetup`**`[algorithm]{font=small,labelfont=bf,labelsep=period}`.

Rozdział 9

Zarządzanie bibliografią

Niezależnie czy przygotowujemy artykuł naukowy, czy pracę dyplomową, na pewno korzystamy z materiałów źródłowych, które należy wykazać w formie tzw. bibliografii. W rozdziale tym omówimy metody samodzielnego przygotowania bibliografii oraz programy, które pozwalają automatycznie ją wygenerować. W takim przypadku będziemy musieli co najwyżej nanieść drobne poprawki. To drugie podejście pozwala na opracowanie własnej bazy danych materiałów źródłowych, z której możemy potem korzystać wielokrotnie przy pracy nad kolejnymi projektami.

9.1 Samodzielne formatowanie bibliografii

Jeżeli przygotowujemy artykuł dla znanego wydawnictwa, to zapewne ustalony jest styl, w jakim złożona powinna zostać bibliografia, i zdecydowanie powinniśmy korzystać z jej automatycznego generowania. Nie zawsze tak jest. Czasami, na przykład przy referatach konferencyjnych, mamy tylko wymagania, jak bibliografia powinna wyglądać po złożeniu, ale nie jest podany styl (lub trudno znaleźć gotowy), który spełniałby te oczekiwania. W takich przypadkach możemy bibliografię przygotować samodzielnie bez wsparcia przez dodatkowe oprogramowanie. Znajomość zagadnień przedstawionych w tym podrozdziale przydaje się również wtedy, gdy chcemy samodzielnie poprawić wygenerowaną bibliografię.

Bibliografię umieszczamy w środowisku **`thebibliography`**, a samo środowisko w miejscu, gdzie chcemy, aby ta bibliografia się znalazła. Środowisko przyjmuje jeden obowiązkowy parametr, którego zadaniem jest określenie, ile maksymalnie znaków potrzeba na umieszczenie najdłuższego identyfikatora bibliografii. Na przykład, jeśli użyjemy numerowanej bibliografii, zawierającej

17 pozycji, to najdłuższy identyfikator to liczba dwucyfrowa, więc parametr powinien zawierać dwa znaki, np. aa. Wewnątrz środowiska każdą pozycję zaczynamy od polecenia `\bibitem[]{}`. Obowiązkowym parametrem polecenia jest etykieta pozycji bibliograficznej (analogicznie jak etykiety definiowane poleceniem `\label`). Opcjonalny parametr może zawierać nasz identyfikator, który zostanie użyty zamiast generowanego automatycznie. Po poleceniu `\bibitem` umieszczamy opis pozycji bibliograficznej. Przykład przygotowanej bibliografii pokazano na listingu 9.1.

Listing 9.1. Przykład środowiska `thebibliography`

```
\begin{thebibliography}{a}
\bibitem{Hewitt:Fizyka:2024} Hewitt P.G.:
\textit{Fizyka wokół nas}. PWN, Warszawa, 2024.

\bibitem{Lie:etal:soundness:2016} Liu G., Reisig W., Jiang C.,
Zhou, M.: \textit{A Branching-Process-Based Method to Check Soundness
of Workflow Systems}. IEEE Access, vol. 4, pp. 4104-4118, 2016.

\bibitem{latex:symbols:2024} Pakin S.: \textit{The Comprehensive
\LaTeX\ Symbol List}. 2024.\\ \url{https://ctan.org/pkg/comprehensive}
\end{thebibliography}
```

Literatura

[1] Hewitt P.G.: *Fizyka wokół nas*. PWN, Warszawa, 2024.

[2] Liu G., Reisig W., Jiang C., Zhou, M.: *A Branching-Process-Based Method to Check Soundness of Workflow Systems*. IEEE Access, vol. 4, pp. 4104-4118, 2016.

[3] Pakin S.: *The Comprehensive LaTeX Symbol List*. 2024. https://ctan.org/pkg/comprehensive

Przedstawione środowisko `thebibliography` zawiera trzy pozycje bibliograficzne, kolejno: książka, artykuł w czasopiśmie, dokumentacja techniczna dostępna w internecie. Poniżej kodu źródłowego pokazano efekt kompilacji dla klasy `article`. Ponieważ nie używaliśmy parametrów opcjonalnych poleceń `\bibitem`, pozycje bibliograficzne są oznaczone kolejnymi numerami umieszczonymi w nawiasach kwadratowych. Mamy tylko trzy pozycje, więc parametr obowiązkowy środowiska `thebibliography` zawiera tylko jeden znak.

Do pozycji bibliograficznych odwołujemy się, korzystając z polecenia `\cite[]{}`. Obowiązkowym parametrem jest tutaj etykieta lub etykiety oddzielone przecinkami. Jako parametr opcjonalny można dodać np. informację

o stronach, na przykład, jeżeli w kodzie użyjemy polecenia `\cite{Hewitt:Fizyka:2024}`, to po kompilacji zostanie ono zastąpione przez [1]. Jeżeli zaś użyjemy polecenia `\cite[str. 171]{Hewitt:Fizyka:2024}`, to dostaniemy [1, str. 171]. Jeżeli jako parametr podamy listę etykiet, to wszystkie odpowiednie identyfikatory znajdą się w jednym nawiasie kwadratowym. Jeżeli modyfikujemy bibliografię lub odwołania do niej, to jej poprawne wyświetlenie będzie wymagało podwójnej kompilacji.

Bibliografia złożona samodzielnie będzie wyglądała tak, jak ją przygotowaliśmy. Innymi słowy, jeżeli pomylimy się w sortowaniu (tutaj sortowano według nazwisk autorów), to ten błąd będzie widoczny. Jeżeli zapomnimy gdzieś o kursywie, kropce, roku publikacji itd., to tak już zostanie. Dodatkowo za błąd uważa się umieszczenie w bibliografii pozycji, na którą się w ogóle nie powołujemy. Przy omawianym podejściu, nawet jeżeli nie zacytujemy żadnej pozycji, to wygląd załączonej bibliografii się nie zmieni. Na koniec został najważniejszy argument, jeżeli otrzymamy informację, że musimy zmienić sortowanie lub sposób formatowania bibliografii liczącej 100 pozycji, to czeka nas naprawdę sporo pracy. Dlatego w dalszej części rozdziału będziemy mówić o automatyzacji procesu składania bibliografii.

9.2 Rekordy w BibTEX-u

Jednym z popularnych formatów używanych do przechowywania informacji o pozycjach bibliograficznych jest format BibTEX-a. Baza danych BibTEX-a to plik tekstowy zawierający co najmniej reden rekord. Plikowi nadajemy rozszerzenie `bib`. Możemy go edytować w dowolnym edytorze tekstu, ale środowiska wspierające LATEX-a zazwyczaj wspierają również przygotowanie plików `bib` (kolorowanie składni, podpowiedzi, generowanie szkieletu rekordu itp.). Na listingu 9.2 przedstawiono zawartość takiego pliku, który zawiera trzy omawiane w poprzednim podrozdziale pozycje bibliograficzne.

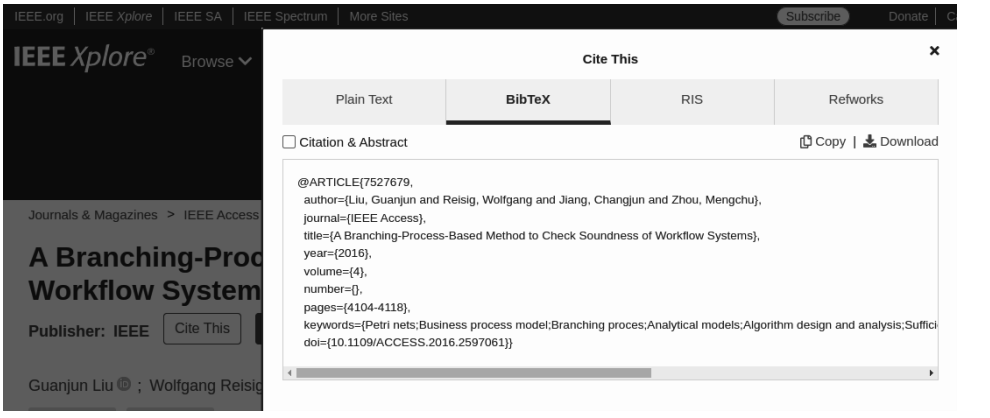
Listing 9.2. Przykład zawartości pliku `bib` z trzema rekordami

```
@book{Hewitt:Fizyka:2024,
author = {Hewitt, P.G.},
title = {Fizyka wokół nas},
publisher = {PWN},
year = {2024},
address = {Warszawa}}

@article{Lie:etal:soundness:2016,
author={G. Liu and W. Reisig and C. Jiang and M. Zhou},
journal={IEEE Access},
```

```
11 title={A Branching-Process-Based Method to Check Soundness
12   of Workflow Systems},
13 year={2016},
14 volume={4},
15 pages={4104-4118},
16 doi={10.1109/ACCESS.2016.2597061}}
17
18 @manual{latex:symbols:2024,
19 author = {Pakin, S.},
20 title = {The Comprehensive {LaTeX} Symbol List},
21 year = {2024},
22 note = {\url{https://ctan.org/pkg/comprehensive}}}
```

Każdy z rekordów w pliku `bib` rozpoczyna się od typu pozycji bibliograficznej, który umieszczamy po znaku `@` (np. `book`, `article`, `manual`). Następnie w nawiasie klamrowym umieszczamy etykietę i pola rekordu. Każde pole rekordu ma format `nazwa = {wartość}` (zamiast nawiasów klamrowych można też użyć podwójnych cudzysłowów). Etykieta i pola rekordu są oddzielone przecinkami. Znaczenie poszczególnych pól w przedstawionym przykładzie wynika jednoznacznie z nazwy pola. Pole `note` (linia 22) zostało użyte tutaj do przechowania informacji o adresie strony internetowej. Dla eleganckiego składu użyliśmy polecenia `\url`.

W tabeli 9.1 przedstawiono podstawowe typy rekordów używanych w bazach danych `bib`. Jest to zestawienie wspierane przez BibTEX-a. Jeżeli do składu bibliografii korzystamy z systemu BibLATEX, to mamy do dyspozycji dodatkowe typy rekordów.



Rys. 9.1. Pozyskiwanie rekordu `bib` z bazy IEEE Xplore

Tabela 9.1. Podstawowe typy rekordów dla plików `bib`

Nazwa	Znaczenie	Pola obowiązkowe	Ważniejsze pola opcjonalne
`Article`	artykuł w czasopiśmie	`author`, `title`, `journal`, `year`	`volume`, `number`, `pages`, `month`, `note`
`InProceedings`	artykuł w materiałach pokonferencyjnych	`author`, `title`, `booktitle`	`pages`, `year`, `editor`, `volume`, `number`, `series`, `publisher`, `note`
`InCollection`	artykuł w zbiorze	`author`, `title`, `booktitle`	`pages`, `year`, `editor`, `volume`, `number`, `series`, `publisher`, `note`
`InBook`	rozdział lub strony w książce	`author` albo `editor`, `title`, `chapter`, `publisher`, `year`	`volume`, `number`, `series`, `pages`, `note`
`Proceedings`	materiały pokonferencyjne	`title`, `year`	`editor`, `volume`, `number`, `series`, `publisher`, `note`
`Book`	książka	`author` albo `editor`, `title`, `publisher`, `year`	`volume`, `number`, `series`, `edition`, `note`
`Booklet`	broszura	`title`	`author`, `howpublished`, `year`, `note`
`PhdThesis`	praca doktorska	`author`, `title`, `school`, `year`	`address`, `note`
`MastersThesis`	praca magistersja	`author`, `title`, `school`, `year`	`address`, `note`
`TechReport`	raport techniczny	`author`, `title`, `institution`, `year`	`number`, `note`
`Manual`	instrukcja	`title`	`author`, `organization`, `year`, `note`
`Unpublished`	niepublikowane	`author`, `title`, `note`	`year`
`Misc`	różne		`author`, `title`, `howpublished`, `year`, `note`

Skąd brać rekordy z opisem bibliografii? Po pierwsze możemy przygotować je sami. Jeżeli potrzebujemy rekordu dla stojącej na półce książki, to bez trudu znajdziemy wymagane informacje. Po drugie wiele baz publikacji naukowych dostarcza takie rekordy. Załóżmy, że mamy tylko tytuł naszego artykułu [2]. Wpisujemy go w wyszukiwarce internetowej. Na liście wyników znajdziemy m.in. bazę IEEE Xplore pokazaną na rys. 9.1. Po kliknięciu przycisku „Cite This" i wybraniu zakładki „BibTeX" mamy gotowy rekord. Warto tylko zmienić etykietę.

Jeżeli samodzielnie przygotowujemy rekord dla bazy danych, to musimy zwrócić uwagę na kilka ważnych kwestii. Dla publikacji wieloautorskich separatorem między kolejnymi autorami jest słowo `and` (zobacz listing 9.2 linia 9). Najczęściej stosowany jest format zapisu, w którym najpierw podajemy nazwisko (może składać się z więcej niż jednego słowa), a potem po przecinku imiona lub inicjały. Dla przykładu rozważmy pełne nazwisko znanego holenderskiego profesora: *Willibrordus Martinus Pancratius van der Aalst*. Ograniczając się tylko do inicjałów imion, jego dane zapiszemy jako `van der Aalst, W.M.P.` Zwróćmy uwagę, że w tym przypadku nazwisko to aż trzy słowa. Jeżeli lista autorów jest bardzo długa, to można podać 2–3 pierwszych i dodać `and others`, co wygeneruje w bibliografii tekst „et al.". Nieeleganckie jest podanie na przykład trzech nazwisk i pominięcie tylko jednego.

Jeżeli chcemy mieć pewność, że polskie znaki diakrytyczne będą poprawnie obsługiwane, to możemy zastosować metodę ich wprowadzania pokazaną w tabeli 2.11 (str. 35).

W zależności od użytego stylu przy składaniu bibliografii system może zmienić wielkość liter na przykład w tytułach. Jeżeli chcemy, aby jakiś tekst nie podlegał tego typu modyfikacjom (np. skrót), to umieszczamy go dodatkowo w klamrach. Zostało to użyte w linii 20, aby zabezpieczyć się przed zmianą wielkości liter w słowie „LaTeX". Jeżeli chcemy, możemy w tytule użyć polecenia **`\LaTeX`**. Tytuł może zawierać również wzory matematyczne.

Nie ma znaczenia, w jakiej kolejności zostaną umieszczone pola rekordu. Może się też zdarzyć, że rekord zawiera dodatkowe pola (np. `keywords` na rys. 9.1), które nie zostaną obsłużone przez oprogramowanie składające bibliografię. Nie wpływają one jednak na poprawność pliku `bib`. Zostaną one pominięte na etapie kompilacji. Jeżeli w pliku `bib` chcemy dodać komentarz, to robimy to jako `@comment{}`. Taki komentarz może obejmować wiele linii. Komentarze możemy wykorzystać, aby dodać własne notatki obok rekordu, np. zapisać sobie, co ciekawego znaleźliśmy w dołączonym do bazy artykule.

Narzędzia przeznaczone do pracy z LaTeX-em zazwyczaj wspierają również edycję plików `bib`. Na listingu 9.3 pokazano przykład szkieletu rekordu typu

book wygenerowany przez polecenie z menu w TeXstudio. Przedrostki ALT i OPT pełnią rolę pomocniczą i są do usunięcia. Pola bez przedrostków to pola obowiązkowe. W przypadku przedrostka ALT powinniśmy wybrać jedno z dwóch pól. Wszystkie pola z przedrostkiem OPT są opcjonalne.

Listing 9.3. Szkielet rekordu dla opisu książki

```
@Book{ID,
ALTauthor = {author},
ALTeditor = {editor},
title = {title},
publisher = {publisher},
year = {year},
OPTkey = {key},
OPTvolume = {volume},
OPTnumber = {number},
OPTseries = {series},
OPTaddress = {address},
OPTedition = {edition},
OPTmonth = {month},
OPTnote = {note},
OPTannote = {annote},
}
```

9.3 BibTEX

Podstawowym narzędziem do generowania składu bibliografii na podstawie rekordów w plikach bib jest BibTEX. Użycie tego programu wymaga od nas przygotowania pliku lub plików bib z rekordami materiałów źródłowych, które chcemy cytować, i dodania w kodzie źródłowym naszego dokumentu tylko dwóch poleceń. Drugie z nich to polecenie **\bibliography**{}, którego parametrem jest nazwa pliku bib (bez rozszerzenia) lub plików bib – nazwy oddzielamy przecinkami. Polecenie **\bibliography** umieszczamy w miejscu, gdzie chcemy, aby znalazła się bibliografia. Pierwsze polecenie (pierwsze w kolejności ich użycia) to **\bibliographystyle**{}. Tutaj parametrem jest nazwa stylu składania bibliografii. Najczęściej jest ono umieszczane bezpośrednio przed poleceniem **\bibliography**. Jeżeli przyjmiemy, że nasza baza znajduje się w pliku latex.bib, to przykładowy kod generujący bibliografię będzie wyglądał jak na listingu 9.4.

Listing 9.4. Polecenia BibTEX-a generujące bibliografię z użyciem stylu plain

```
\bibliographystyle{plain}
\bibliography{latex}
```

Pozycje bibliograficzne cytujemy za pomocą polecenia **\cite**, dokładnie tak samo jak w przypadku samodzielnie poskładanej bibliografii. Środowisko pracy „wie", jakie pliki `bib` dodaliśmy do projektu, więc stara się nam podpowiadać, np. Overleaf lub Texstudio po wpisaniu polecenia **\cite** i otwierającego nawiasu klamrowego wyświetli listę etykiet. Ważne jest, aby wypracować taki styl konstruowania etykiet, aby jak najłatwiej można było znaleźć właściwą na liście.

Jeżeli korzystamy z polecenia `pdflatex` do kompilacji dokumentu, poprawna sekwencja działań to (zakładamy, że plik z preambułą to `main.tex`):

1) `pdflatex main.tex`
2) `bibtex main`
3) `pdflatex main.tex`
4) `pdflatex main.tex`

Środowiska takie jak TexStudio i Overleaf robią to automatycznie. BibTEX wygeneruje automatycznie środowisko **thebibliography**, które zostanie umieszczone w pliku o rozszerzeniu `bbl` – jeżeli nasz główny plik nazywa się `main.tex`, to będzie to plik `main.bbl`. W pliku `pdf` powinniśmy za to zobaczyć elegancko i jednolicie (jeśli rekordy były poprawne) złożoną bibliografię. „Jednolicie" oznacza tutaj dokładnie taką samą kolejność i formatowanie pól przy składaniu każdego artykułu, dokładnie taką samą kolejność i formatowanie pól przy składaniu każdej książki itd.

Najważniejszą cechą BibTEX-a (również BibLATEX-a, o czym w kolejnym podrozdziale) jest to, że środowisko **thebibliography** zostanie złożone wyłącznie z tych rekordów, które cytujemy. To oznacza, że w naszej bazie możemy mieć setki rekordów zbieranych przez lata, korzystać z tej bazy przy każdym projekcie i nie ma obawy, że w bibliografii znajdą się zbędne rekordy.

Jeżeli z jakiegoś powodu potrzebujemy dokonać jakiejkolwiek ręcznej korekty w bibliografii, to robimy to w momencie, gdy mamy pewność, że nie będziemy już dodawać nowych cytowań. W takim przypadku należy zakomentować linie z poleceniami **\thebibliography** i **\bibliographystyle**, a dodać polecenie **\input**`{main.bbl}` (z właściwą nazwą pliku `bbl`). W pliku `bbl` możemy wtedy wykonać wymagane korekty. Jeżeli nie chcemy trzymać bibliografii w oddzielnym pliku, to po prostu zawartość pliku `bbl` możemy skopiować w miejsce zakomentowanych dwóch poleceń.

Wróćmy do polecenia **\bibliographystyle** i kwestii jego parametru. Jeżeli pobraliśmy szablon dokumentu ze strony wydawnictwa, to z góry jest

określone, jakiego stylu bibliografii należy użyć. Możliwe, że sam plik ze stylem jest dołączony do pobranego archiwum – poznamy go po rozszerzeniu `bst`. Wówczas jako nazwy stylu używamy nazwy pliku (bez rozszerzenia `bst`). Jeżeli składamy pracę dyplomową, doktorską, piszemy książkę itp., to prawdopodobnie możemy sami wybrać styl bibliografii. Jeżeli styl, który nam się podoba, nie jest dostępny w systemie, którego używamy, to można pobrać z internetu plik stylu i umieścić go w tym samym katalogu, gdzie znajduje się główny plik projektu.

Na rys. 9.2 pokazano cztery wybrane standardowe style składania bibliografii dla wcześniej rozważanych trzech rekordów. Różnice między stylami dotyczą zazwyczaj kolejności sortowania, umieszczania inicjałów imion przed lub po nazwisku, formatowania wybranych pól (np. nazwiska autorów składane kapitalikami) czy też kolejności, w jakiej niektóre pola są umieszczane. W przypadku stylów, które same generują nienumeryczne identyfikatory, również forma identyfikatorów zależy od stylu (widać to wyraźnie między stylami `alpha` i `apalike`). Jeżeli wybieramy styl z nienumerycznymi identyfikatorami, to zmienia to także sposób wyświetlania cytowanych pozycji. Na przykład użycie stylu `apalike` i polecenia `\cite{Hewitt:Fizyka:2024}` wygeneruje [Hewitt, 2024], a w przypadku stylu `acm` – [Hew24].

Jeżeli w naszym dokumencie cytujemy co najmniej kilkadziesiąt pozycji bibliograficznych, to absolutnie nie powinniśmy używać stylów, które na przykład układają pozycje według kolejności cytowania i używają nienumerycznych identyfikatorów. Jeżeli czytelnik będzie chciał wtedy znaleźć pozycję opatrzoną np. kodem `LRJZ16`, to będzie musiał przejrzeć w pesymistycznym przypadku całą bibliografię.

Jeżeli chcemy zmienić nazwę dla sekcji (rozdziału), który zawiera bibliografię, to musimy redefiniować polecenie **`\refname`** (w przypadku klasy `article`) lub **`\bibname`** (dla klas `report` i `book`), np. **`\renewcommand`** `{`**`\bibname`**`}{Materiały źródłowe}`.

Poza poleceniem **`\cite`** dostępne jest też polecenie **`\nocite`**`{}`. Parametrem jest etykieta pozycji bibliograficznej. Polecenie dodaje daną pozycję do bibliografii, ale w dokumencie nie pojawia się cytowanie. Polecenie **`\nocite`** obsługuje również parametr w postaci gwiazdki (`*`). Użycie go powoduje dodanie wszystkich rekordów z pliku `bib`. Powinniśmy go używać tylko wtedy, gdy chcemy wygenerować plik `pdf` z całą naszą zebraną bibliografią. Tak jak wspomniałem już wcześniej, nie powinniśmy dodawać do bibliografii pozycji, na które ani razu się nie powołujemy.

Literatura

[1] P.G. Hewitt. *Fizyka wokół nas*. PWN, Warszawa, 2024.

[2] G. Liu, W. Reisig, C. Jiang, and M. Zhou. A branching-process-based method to check soundness of workflow systems. *IEEE Access*, 4:4104–4118, 2016.

[3] S. Pakin. *The Comprehensive LaTeX Symbol List*. `https://ctan.org/pkg/comprehensive`, 2024.

Literatura

[Hew24] P.G. Hewitt. *Fizyka wokół nas*. PWN, Warszawa, 2024.

[LRJZ16] G. Liu, W. Reisig, C. Jiang, and M. Zhou. A branching-process-based method to check soundness of workflow systems. *IEEE Access*, 4:4104–4118, 2016.

[Pak24] S. Pakin. *The Comprehensive LaTeX Symbol List*. `https://ctan.org/pkg/comprehensive`, 2024.

Literatura

[1] Hewitt, P. *Fizyka wokół nas*. PWN, Warszawa, 2024.

[2] Liu, G., Reisig, W., Jiang, C., and Zhou, M. A branching-process-based method to check soundness of workflow systems. *IEEE Access 4* (2016), 4104–4118.

[3] Pakin, S. *The Comprehensive LaTeX Symbol List*. `https://ctan.org/pkg/comprehensive`, 2024.

Literatura

[Hewitt, 2024] Hewitt, P. (2024). *Fizyka wokół nas*. PWN, Warszawa.

[Liu et al., 2016] Liu, G., Reisig, W., Jiang, C., and Zhou, M. (2016). A branching-process-based method to check soundness of workflow systems. *IEEE Access*, 4:4104–4118.

[Pakin, 2024] Pakin, S. (2024). *The Comprehensive LaTeX Symbol List*. `https://ctan.org/pkg/comprehensive`.

Rys. 9.2. Przykładowe style składania bibliografii: `plain`, `alpha`, `acm`, `apalike`

Warto dodać, że w przypadku BibTeX-a próba opracowania własnego pliku ze stylem lub nawet poprawienia istniejącego stylu to dosłowna droga przez mękę. Jest to kluczowy powód, dla którego opracowano nowy program do zarządzania bibliografią, któremu poświęcono kolejny podrozdział.

Niezależnie od tego, którego ze standardowych stylów użyjemy, cytowana pozycja będzie umieszczona w nawiasie kwadratowym. W niektórych przypadkach wydawnictwo oczekuje cytowania w układzie autor-rok, np.: Hewitt (2024). Efekt taki możemy uzyskać z użyciem pakietu `natbib` [11]. Jeżeli połączymy go ze stylem `apalike`, to uzyskamy dokładnie takie cytowania, a skład bibliografii pozostanie jak na rys. 9.2, ale z pominięciem identyfikatorów. Warto podkreślić, że pakiet nie współpracuje z każdym stylem. W niektórych przypadkach dostaniemy komunikat, że dany styl nie jest kompatybilny z cytowaniem w układzie autor-rok. Pakiet `natbib` oferuje również własne style, a także dodatkowe polecenia, których możemy użyć zamiast **`\cite`**, aby uzyskać określony styl wyświetlania cytowania, np. **`\citep`**`{Hewitt:Fizyka:2024}` wygeneruje (Hewitt, 2024). Więcej informacji można znaleźć w dokumentacji [11].

9.4 BibLaTeX

BibLaTeX jest uważany za następcę BibTeX-a. Oprogramowanie m.in. może korzystać z plików `bib`, o których mówiłem wcześniej. Z perspektywy użytkownika oznacza to niewielkie zmiany w sposobie dodania pliku `bib` do projektu i wygenerowania bibliografii. W zamian otrzymujemy duże możliwości w zakresie dopasowania stylu bibliografii do naszych potrzeb.

`\usepackage``{biblatex}`

Listing 9.5. Polecenia BibLaTeX-a generujące bibliografię

```
% w preambule
\usepackage{biblatex}
\addbibresource{latex.bib}
\addbibresource{mathematics.bib}

% w dokumencie, w miejscu, gdzie ma być umieszczona bibliografia
\printbibliography
```

Przykład minimalnego zestawu poleceń niezbędnych do skorzystania z BibLaTeX-a pokazano na listingu 9.5. Zaczynamy od dodania w preambule pakietu `biblatex` [29] oraz polecenia **`\addbibresource`**, którego parametrem

jest nazwa pliku wraz z rozszerzeniem `bib`. Jeżeli chcemy dołączyć więcej plików z bibliografią, to polecenia **`\addbibresource`** używamy wielokrotnie (linie 3–4). W miejscu, gdzie ma się znaleźć bibliografia, umieszczamy polecenie **`\printbibliography`**. Pozycje bibliograficzne możemy cytować za pomocą polecenia **`\cite`** dokładnie tak samo jak w przypadku BibTEX-a. Polecenie **`\printbibliography`** może przyjmować parametry opcjonalne, np. **`\printbibliography`**`[title={Materiały źródłowe}]` pozwala nam łatwo zmienić tytuł sekcji (rozdziału z bibliografią).

Jeżeli korzystamy ze środowiska Overleaf, to w zasadzie wszystkie niezbędne zmiany. Możemy skompilować dokument i powinniśmy zobaczyć domyślny wygląd numerowanej bibliografii. Jeżeli korzystamy z własnej instalacji systemu, to powinniśmy doinstalować oprogramowanie `biber` używane przez BibLATEX-a do składania bibliografii. W procesie kompilowania dokumentu `biber` jest uruchamiany zamiast polecenia `bibtex`.

Dopasowanie wyglądu bibliografii do naszych potrzeb jest realizowane poprzez parametry opcjonalne polecenia **`\usepackage`**, przy dołączaniu pakietu `biblatex`. Przede wszystkim możemy wybrać niezależne style do cytowania pozycji bibliograficznych i do ich wyświetlania w bibliografii. Jest to realizowane poprzez opcje `citestyle` i `bibstyle`. Jeżeli w obu przypadkach chcemy skorzystać ze stylów o tych samych nazwach, to po prostu używamy opcji `style`.

Listing 9.6. Kod źródłowy użyty do demonstarcji stylów cytowania

```
1 Moment siły definiuje się jako iloczyn siły
2 i jej ramienia~\cite[str. 171]{Hewitt:Fizyka:2024}.
3
4 W artykule~\cite{Lie:etal:soundness:2016} analizowana
5 jest poprawność sieci klasy WF.
```

Podstawowe style cytowania zostały zademonstrowane na rys. 9.3. W każdym przypadku użyto tego samego kodu źródłowego pokazanego na listingu 9.6 i podanego w podpisie stylu, np. `citestyle=numeric`. Styl `authoryear` powinien zostać użyty również z opcją `bibstyle` (podobnie `authortitle`). Ostatni styl (`verbose`) jest przeznaczony przede wszystkim do cytowania w przypisach. Jeżeli dodatkowo dołączymy opcję `natbib`, to włączamy tryb zgodności z pakietem `natbib`. Pozwala to na korzystanie z dodatkowych poleceń, np. **`\citep`**. W połączeniu ze stylem `authoryear` pozwala to uzyskać cytowania w postaci (Liu i in., 2016).

Prezentowane style występują często w kilku wersjach, na przykład dostępne są style `numeric-comp` i `numeric-verb`. Pierwszy z nich stosuje zakresy, tzn. jeżeli w ramach jednego polecenia **`\cite`** cytujemy kilka kolejnych pozycji, to zamiast np. [5, 6, 7] dostaniemy [5–7]. Drugi z kolei przedstawi wynik tego samego

polecenia jako [5]; [6]; [7]. Pełny przegląd wszystkich wariantów dostępny jest w obszernej dokumentacji pakietu [29].

Moment siły definiuje się jako iloczyn siły i jej ramienia [1, str. 171].
W artykule [2] analizowana jest poprawność sieci klasy WF.

Moment siły definiuje się jako iloczyn siły i jej ramienia [Hew24, str. 171].
W artykule [Liu+16] analizowana jest poprawność sieci klasy WF.

Moment siły definiuje się jako iloczyn siły i jej ramienia Hewitt 2024, str. 171.
W artykule Liu i in. 2016 analizowana jest poprawność sieci klasy WF.

Moment siły definiuje się jako iloczyn siły i jej ramienia Hewitt, *Fizyka wokół nas*, str. 171.
W artykule Liu i in., "A Branching-Process-Based Method to Check Soundness of Workflow Systems" analizowana jest poprawność sieci klasy WF.

Moment siły definiuje się jako iloczyn siły i jej ramienia P.G. Hewitt. *Fizyka wokół nas*. Warszawa: PWN, 2024, str. 171.
W artykule G. Liu i in. "A Branching-Process-Based Method to Check Soundness of Workflow Systems". W: *IEEE Access* 4 (2016), s. 4104–4118. DOI: `10.1109/ACCESS.2016.2597061` analizowana jest poprawność sieci klasy WF.

Rys. 9.3. Przykładowe style cytowania bibliografii: `numeric`, `alphabetic`, `authoryear`, `authortitle`, `verbose`

Podstawowe style wyświetlania bibliografii przedstawiono na rys. 9.4. Styl `verbose` generuje tutaj taki sam wynik jak styl `authortitle`. Wszystkie style skracają listę autorów do jednego, jeśli zawiera ona więcej niż trzech autorów. Z tego powodu w przypadku drugiej publikacji widzimy autorów w postaci „G. Liu i in.". Możemy nadpisać te ustawienia, zmieniając opcje `maxnames` (liczba autorów, powyżej której lista jest skracana) i `minnames` (liczba autorów na skróconej liście). Na przykład ustawienie opcji `minnames=2` przy pozostawieniu bez zmiany opcji `maxnames` spowoduje, że lista autorów drugiej pozycji będzie wyświetlana jako „G. Liu, W. Reisig i in.". Jeżeli zastosujemy opcję `maxnames=4` (lub większą wartość), to w przypadku rozważanej pozycji bibliograficznej zobaczymy wszystkie nazwiska.

Warto zwrócić uwagę na to, że nie każda kombinacja opcji ma sens. Jeżeli ustawimy `citestyle= numeric, bibstyle=alphabetic`, to cytowania będą w postaci [1], a w bibliografii zobaczymy identyfikator `Hew24`.

Bibliografia

[1] P.G. Hewitt. *Fizyka wokół nas*. Warszawa: PWN, 2024.

[2] G. Liu i in. "A Branching-Process-Based Method to Check Soundness of Workflow Systems". W: *IEEE Access* 4 (2016), s. 4104–4118. DOI: 10.1109/ACCESS.2016.2597061.

[3] S. Pakin. *The Comprehensive LaTeX Symbol List*. https://ctan.org/pkg/comprehensive. 2024.

Bibliografia

[Hew24] P.G. Hewitt. *Fizyka wokół nas*. Warszawa: PWN, 2024.

[Liu+16] G. Liu i in. "A Branching-Process-Based Method to Check Soundness of Workflow Systems". W: *IEEE Access* 4 (2016), s. 4104–4118. DOI: 10.1109/ACCESS.2016.2597061.

[Pak24] S. Pakin. *The Comprehensive LaTeX Symbol List*. https://ctan.org/pkg/comprehensive. 2024.

Bibliografia

Hewitt, P.G. (2024). *Fizyka wokół nas*. Warszawa: PWN.

Liu, G. i in. (2016). "A Branching-Process-Based Method to Check Soundness of Workflow Systems". W: *IEEE Access* 4, s. 4104–4118. DOI: 10.1109/ACCESS.2016.2597061.

Pakin, S. (2024). *The Comprehensive LaTeX Symbol List*. https://ctan.org/pkg/comprehensive.

Bibliografia

Hewitt, P.G. *Fizyka wokół nas*. Warszawa: PWN, 2024.

Liu, G. i in. "A Branching-Process-Based Method to Check Soundness of Workflow Systems". W: *IEEE Access* 4 (2016), s. 4104–4118. DOI: 10.1109/ACCESS.2016.2597061.

Pakin, S. *The Comprehensive LaTeX Symbol List*. https://ctan.org/pkg/comprehensive. 2024.

Rys. 9.4. Przykładowe style wyświetlania bibliografii: `numeric`, `alphabetic`, `authoryear`, `authortitle`

Pakiet oferuje również opcję `autocite` i polecenie **`\autocite`**. Polecenie stosujemy analogicznie jak **`\cite`**, ale możemy z poziomu opcji pakietu sterować sposobem działania tego polecenia. Na rys. 9.5 pokazano przykład użycia opcji `autocite=footnote` dla kodu źródłowego z listingu 9.6, przy czym zmieniono polecenia **`\cite`** na **`\autocite`** (druga użyta opcja to `style=authortitle`). W efekcie tych ustawień każde cytowanie zrealizowane z użyciem polecenia **`\autocite`** powoduje wygenerowanie przypisu, w którym umieszczane jest to cytowanie.

Moment siły definiuje się jako iloczyn siły i jej ramienia[1].
W artykule[2] analizowana jest poprawność sieci klasy WF.

[1] Hewitt, *Fizyka wokół nas*, str. 171.
[2] Liu i in., "A Branching-Process-Based Method to Check Soundness of Workflow Systems".

Rys. 9.5. Przykład użycia opcji `autocite=footnote`

Pakiet `biblatex` umożliwia bardzo elastyczne sterowanie opcjami sortowania bibliografii poprzez opcję `sorting`. Domyślny sposób sortowania oznaczony jest skrótem `nty` (name, title, year). Dostępne są m.in. opcje `nyt`, `ynt` i `ydnt`. W tym ostatnim przypadku mamy sortowanie od najnowszych pozycji do najstarszych. Jeżeli chcemy, aby pozycje pojawiały się w kolejności cytowania, to ustawiamy wartość `none` dla opcji `sorting`.

Polecenie **`\cite`** zostało w pakiecie `biblatex` rozbudowane do wersji **`\cite`**`[][]{}`. Parametry opcjonalne są określane w dokumentacji pakietu jako *prenote* i *postnote*. W pierwszym z nich możemy podać tekst, który zostanie umieszczony na początku cytowania, a w drugim – na końcu. Jeżeli przy ustawieniach `style=authoryear` zastosujemy polecenie w postaci: **`\cite`**`[zob.][str. 25]{Hewitt:Fizyka:2024}`, to otrzymamy: zob. Hewitt 2024, str. 25. Jeżeli podamy tylko jeden parametr opcjonalny, to jego zawartość zostanie umieszczona na końcu cytowania. Jeżeli chcemy dodać tekst tylko na początku, to musimy użyć obu parametrów opcjonalnych i zostawić drugi pusty.

BibLaTeX bezproblemowo obsługuje dużo więcej pól w rekordach niż BibTeX, na przykład nie ma problemu z obsługą adresów internetowych, radzi sobie ze składaniem wieloczłonowych nazwisk itd. Pakiet pozwala podzielić bibliografię na części według różnych filtrów.

Listing 9.7. Przykłady filtrowania bibliografii

```
\printbibliography[type=manual, title={Dokumentacja pakietów \LaTeX-a}]
\printbibliography[keyword={latex}, title={Bibliografia \LaTeX}]
```

Na listingu 9.7 pokazano dwa przykłady filtrowania bibliografii. W pierwszym przypadku wydrukowane zostaną tylko cytowane rekordy typu `manual`. W drugim przypadku wydruk będzie zawierał rekordy, które w polu `keywords` mają słowo „latex” (mogą być też inne słowa kluczowe). Pozycje na listach nie będą numerowane niezależnie, ale będą używać identyfikatorów z jednej kompletnej bibliografii. Jeżeli zamieścimy w dokumencie kilka takich odfiltrowanych zestawów, to może się zdarzyć, że ta sama pozycja znajdzie się na więcej niż jednej liście.

BibTEX jest ciągle powszechniej stosowanym systemem obsługi bibliografii w LATEX-u. Zapewne jeszcze przez długi czas będzie używany przez wiele wydawnictw. Biorąc jednak pod uwagę zalety BibLATEX-a, należy się liczyć, że z czasem przejmie dominującą pozycję na rynku. Na dziś warto jednak znać oba systemy.

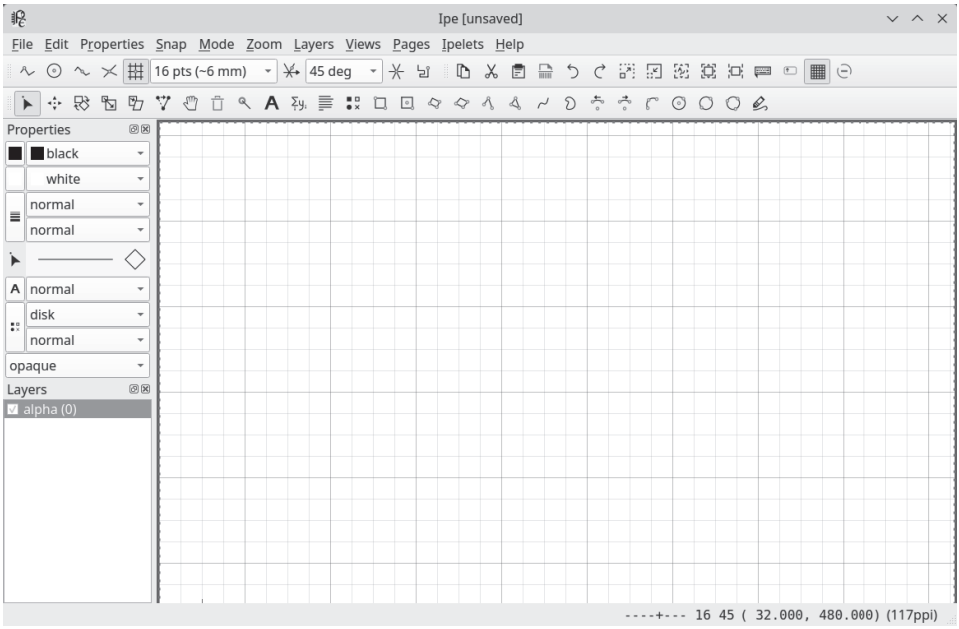
Rozdział 10

Ipe

Ipe jest wyjątkowym edytorem grafiki wektorowej. Jego cechą szczególną jest to, że jego naturalną częścią jest LaTeX. Po pierwsze dla dokumentu mamy możliwość zdefiniowania preambuły, co m.in. pozwala używać na rysunkach tych samych czcionek co w dokumencie. Po drugie Ipe traktuje etykiety tekstowe dodawane do rysunku jako tekst w LaTeX-u. W efekcie tego założenia, dodając opisy do rysunków, możemy korzystać z poleceń LaTeX-a, w szczególności bez problemu dodamy do rysunku nawet bardzo złożone wzory matematyczne. Etykieta po dodaniu jest kompilowana i na rysunku widzimy od razu efekt tej kompilacji. Ponadto edytor oferuje zaawansowane mechanizmy przyciągania, ułatwiające bardzo precyzyjne wskazywanie interesujących nas punktów (np. przecięcia dwóch linii). Zawiera również mechanizm rozszerzania go o nowe funkcjonalności, tzw. ipelety. Aktualnie dostępnych jest kilkadziesiąt takich dodatkowych poleceń. W rozdziale tym przyjrzymy się wybranym możliwościom edytora Ipe, które czynią go tak wyjątkowym.

10.1 Wprowadzenie

Edytor Ipe (czyt. *ajp*) jest darmowym oprogramowaniem dostępnym dla większości dystrybucji Linuksa, macOS i systemów z rodziny Windows. Wersję instalacyjną można pobrać ze strony projektu `https://ipe.otfried.org`, która zawiera również m.in. oficjalny podręcznik użytkownika [51]. Ipe zapisuje swoje pliki w formatach `xml` i `pdf`. To drugie rozwiązanie jest o tyle wygodne, że rysunek od razu jest gotowy do wstawienia do dokumentu LaTeX-a. Rysunek zapisany w formacie `pdf` można ponownie edytować. Należy jednak pamiętać, że Ipe może edytować tylko własne pliki `pdf`, które nie zostały zmodyfikowane przez inne oprogramowanie.



Rys. 10.1. Widok edytora Ipe bezpośrednio po uruchomieniu

Widok edytora Ipe bezpośrednio po uruchomieniu pokazano na rys. 10.1. Główny obszar okna wypełnia kanwa pokryta siatką ułatwiającą rysowanie. Kanwa jest domyślnie rozmiaru A4 (można zmienić preferencje), ale jeżeli podczas rysowania wykorzystamy tylko jej część, to rozmiar zapisanego w pliku `pdf` rysunku będzie odpowiadał rozmiarowi wykorzystanej części kanwy. W menu głównym i paskach narzędziowych powyżej kanwy zebrane zostały m.in. polecenia tworzące i modyfikujące obiekty na rysunku. Uzupełnieniem tych elementów jest menu kontekstowe, które zależnie od kontekstu może być bardzo bogate w oferowane polecenia.

Panele widoczne po lewej stronie służą do ustawiania własności zaznaczonego obiektu lub bieżącego rysowania oraz do zarządzania warstwami rysunku. Ustawienie kursora myszy nad dowolnym przyciskiem na paskach narzędziowych lub panelu powoduje wyświetlenie krótkiej podpowiedzi wskazującej przeznaczenie przycisku. Kliknięcie prawym przyciskiem myszy w pustym obszarze na dowolnym pasku narzędziowym lub tytułowej belki panelu wyświetla menu kontekstowe, które pozwala włączać i wyłączać widok poszczególnych pasków narzędziowych i paneli. Możemy dodatkowo wyświetlić niewidoczne domyślnie panele z zakładkami i notatkami. Panele możemy ustawiać w dowolnej kolejności, przy prawej lub lewej krawędzi okna. Wystarczy je przeciągnąć za belkę tytułową.

Pracę nad nowym rysunkiem powinniśmy zacząć od ustawienia preambuły, zwłaszcza jeśli chcemy korzystać z języka polskiego lub chcemy używać innych czcionek niż domyślne. W tym celu wybieramy polecenie `Edit→Document properties`. W oknie, które się wyświetli, możemy wprowadzić metadane dotyczące autora, tytułu, słów kluczowych itp. Dla prac z edytorem istotne jest pole `LaTeX preamble`. Dodajemy tam pakiety, z których chcemy korzystać. Przy założeniu, że chcemy korzystać z czcionek Times Roman oraz pakietów AMS, taka minimalna zawartość preambuły pokazana jest na listingu 10.1 (nie dodajemy polecenia `\documentclass`).

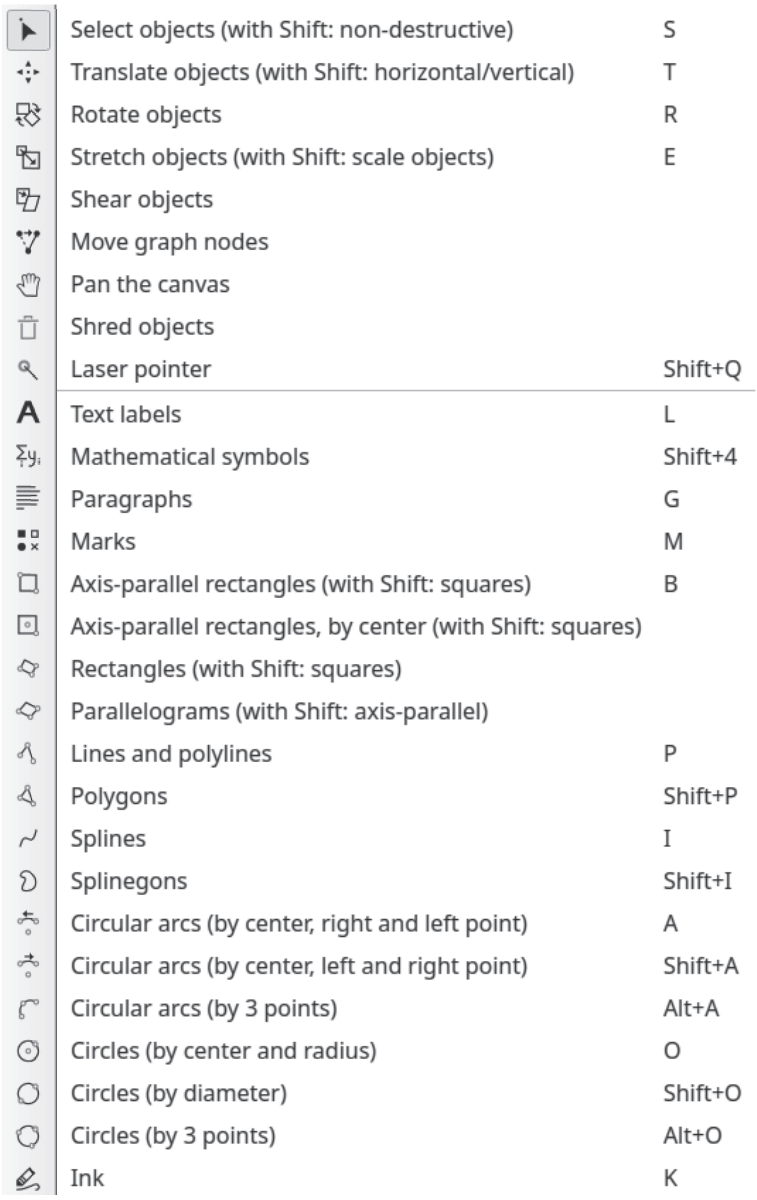
Listing 10.1. Przykładowa preambuła dla rysunku w Ipe

```
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[polish]{babel}
\usepackage{newtx}
\usepackage{amssymb}
\usepackage{amsmath}
```

10.2 Tryby rysowania

Polecenia związane z trybami rysowania i modyfikowania obiektów znajdują się w menu `Mode` i dokładnie w tej samej kolejności na pasku narzędziowym `Objects` (zob. rys. 10.2). Wybór trybu jest trwały, tj. jeżeli wybierzemy tryb `Marks`, to dopóki nie wybierzemy innego trybu, każde kliknięcie lewym przyciskiem myszy będzie dodawało kolejny znacznik do rysunku. W tym podrozdziale skupimy się na trybach od 13 do 28 (przy czym 13 omówimy na końcu). Przy opisie poszczególnych trybów będziemy używać skrótów: `LP` i `PP` do oznaczenia kliknięcia odpowiednio lewym i prawym przyciskiem myszy. Niektóre polecenia mają zmienione działanie, jeśli jednocześnie wciśnięty jest klawisz `Shift`. Można to symulować przyciskiem `Shift` na pasku narzędziowym `Edit`.

14. Prostokąt o bokach równoległych do krawędzi okna – wskazujemy dwa przeciwległe punkty (`LP`). Przy rysowaniu z wciśniętym klawiszem `Shift` otrzymujemy kwadrat.
15. Prostokąt jak wyżej, ale wskazujemy środek i jeden z narożników.
16. Prostokąt – wskazujemy trzy kolejne narożniki (`LP`). Przy rysowaniu z klawiszem `Shift` otrzymujemy kwadrat (wskazujemy dwa narożniki).
17. Równoległobok – wskazujemy trzy kolejne narożniki (`LP`). Przy rysowaniu z klawiszem `Shift` pierwszy bok jest równoległy do krawędzi okna.



Rys. 10.2. Pasek narzędziowy `Objects` i menu `Mode`

18. Odcinek lub łamana – pierwszy i pośrednie węzły dodajemy za pomocą `LP`, ostatni węzeł za pomocą `PP`.
19. Wielokąt – pierwszy i kolejne wierzchołki dodajemy za pomocą `LP`, ostatni za pomocą `PP`.

20. Splajn – pierwszy i pośrednie węzły dodajemy za pomocą `LP`, ostatni za pomocą `PP`.
21. Zamknięty splajn – rysujemy jak wyżej.
22. Łuk okręgu – wskazujemy środek i dwa punkty na okręgu, najpierw prawy, potem lewy (`LP`).
23. Łuk okręgu – wskazujemy środek i dwa punkty na okręgu, najpierw lewy, potem prawy (`LP`).
24. Łuk okręgu – wskazujemy trzy punkty na okręgu (`LP`).
25. Okrąg – wskazujemy środek i promień (`LP`).
26. Okrąg – wskazujemy dwa punkty tworzące średnicę (`LP`).
27. Okrąg – wskazujemy trzy punkty, przez które przechodzi okrąg (`LP`).
28. Swobodne rysowanie kursorem myszy.

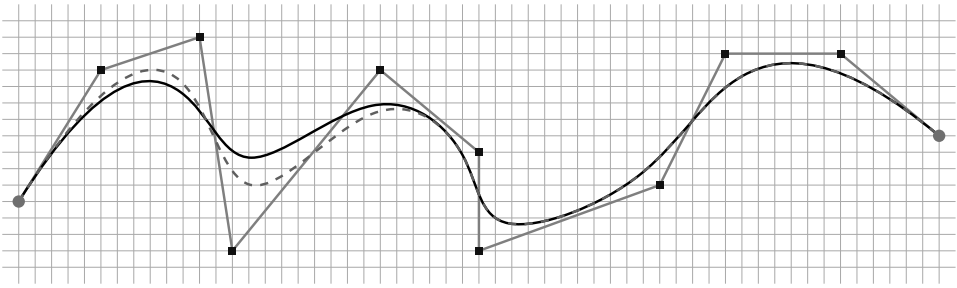
Z powyższych trybów wyjaśnienia wymaga tylko temat splajnów. Splajny, zwane inaczej krzywymi sklejanymi, są złożone z wielu łuków wielomianowych stosunkowo niskiego stopnia. Ipe domyślnie stosuje jednorodne B-splajny. Jeżeli zastosujemy tylko trzy punkty kontrolne P_1, P_2 i P_3, to taki splajn jest kwadratową krzywą Béziera – zaczyna się w punkcie P_1, w którym jest styczna do prostej przechodzącej przez punkty P_1 i P_2, oraz kończy się w punkcie P_3, w którym jest styczna do prostej przechodzącej przez punkty P_2 i P_3. Dowolny fragment paraboli możemy przedstawić za pomocą kwadratowej krzywej Béziera. Przy czterech punktach kontrolnych P_1, P_2, P_3 i P_4 splajn jest kubiczną krzywą Béziera – zaczyna się w punkcie P_1, w którym jest styczna do prostej przechodzącej przez punkty P_1 i P_2, oraz kończy się w punkcie P_4, w którym jest styczna do prostej przechodzącej przez punkty P_3 i P_4 (zob. podrozdział 11.3). Przy większej liczbie punktów kontrolnych mamy sklejenie serii kubicznych krzywych Béziera. Krzywa taka zaczyna się w pierwszym punkcie, kończy w ostatnim i aproksymuje pozostałe. Jedną z istotnych cech takich krzywych jest to, że są definiowane lokalnie, tzn. poszczególne fragmenty zależą tylko od kilku punktów kontrolnych. Zmiana punktu wpływa tylko na kilka segmentów.

Jako uzupełnienie dla jednorodnych B-splajnów Ipe oferuje splajny kardynalne, które interpolują punkty kontrolne, n punktów kontrolnych wyznacza $n-1$ krzywych Béziera, dla których para sąsiednich punktów jest początkiem i końcem, a styczna w każdym punkcie kontrolnym jest równoległa do odcinka łączącego poprzedni i następny punkt kontrolny. Trzeci typ splajnów to splajny klotoidalne (splajny typu spiro). Krzywe te są wyznaczane przez rozwiązanie problemu optymalizacji, który próbuje zminimalizować zmianę krzywizny. W efekcie mamy wrażenie, że w dużej mierze stanowią połączenie łuków okręgu. Typ splajnu można ustalić w menu: `Properties→Spline type`.

Polecenie 1 z rys. 10.2 pozwala na zaznaczenie obiektu lub obiektów. Ipe wyróżnia zaznaczone obiekty przez obrysowanie ich kolorową linią. Wśród zaznaczonych obiektów jeden jest wyróżniony i stanowi podstawowe zaznaczenie. Niektóre operacje, niezależnie od liczby zaznaczonych obiektów, dotyczą tylko tego podstawowego. Jeżeli jesteśmy w trybie zaznaczania, to kliknięcie lewym przyciskiem myszy w pobliżu brzegu danego obiektu zaznacza obiekt, odznaczając inne. Przy wciśniętym klawiszu `Shift` możemy dodawać lub usuwać obiekty z bieżącego zaznaczenia. Można również zaznaczyć obiekty należące do prostokątnego obszaru, rysując taki prostokąt w trybie zaznaczania przy wciśniętym lewym przycisku myszy. Jeżeli robimy to przy dodatkowo wciśniętym klawiszu `Shift`, to status zaznaczenia obiektów wewnątrz prostokąta zostanie zmieniony na przeciwny.

Jeżeli próbujemy zaznaczyć obiekt i wciskamy lewy klawisz myszy, to Ipe wyznacza listę obiektów, które są wystarczająco blisko wskaźnika i sortuje je według dystansu od wskaźnika myszy i głębokości ułożenia, po czym wybiera pierwszy z nich. Jeżeli przytrzymamy wciśnięty lewy klawisz myszy i będziemy naciskać spację, to możemy wybrać inny obiekt niż pierwszy z listy. Operacja ta jest użyteczna, gdy kilka obiektów jest blisko siebie i mamy problem z zaznaczeniem tego, który nas interesuje. Obiekt, który aktualnie jest „bieżący", jest wyróżniany cienką czerwoną linią. Jeżeli zwolnimy klawisz myszy, to ten bieżący obiekt zostanie zaznaczony. W przypadku bardziej złożonych rysunków warto umieszczać elementy na różnych warstwach, co również ułatwia zaznaczanie elementów. Temat warstw omówię w podrozdziale 10.7.

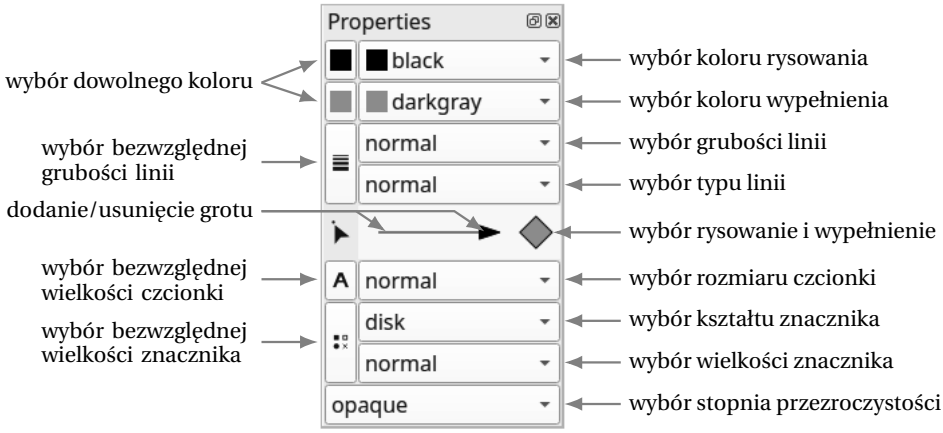
Polecenie `Edit→Edit object` (skrót `Ctrl+E`) pozwala przejść do trybu edycji obiektu stanowiącego podstawowe zaznaczenie. Na rys. 10.3 pokazano przykład splajnu w trybie edycji.



Rys. 10.3. Splajn w trybie edycji

W trybie tym możemy między innymi przesuwać węzły kontrolne (na rysunku widać efekt zmiany położenia dwóch punktów kontrolnych, co lokalnie odkształciło krzywą). Możemy również dodawać kolejne węzły lub je usuwać.

Dostępne w menu podręcznym operacje zależą od typu ścieżki, na przykład ścieżkę będącą łamaną możemy rozciąć w danym węźle, zamieniając ją na dwie podścieżki. Edytować w podobny sposób można dowolny z obiektów opisywanych w tym podrozdziale, np. możemy przesunąć jeden narożnik prostokąta, zmieniając go w czworokąt wklęsły. Tryb edycji opuszczamy, wybierając polecenie `Accept` z menu podręcznego lub naciskając klawisz spacji. Jeżeli chcemy anulować zmiany, to możemy opuścić tryb edycji, naciskając klawisz `Esc`.



Rys. 10.4. Przeznaczenie poszczególnych przycisków w panelu `Properties`

Poza modyfikacją kształtu ścieżki możemy ustalać wiele innych jej parametrów, takich jak kolor brzegu, kolor wypełnienia, typ i grubość linii, przezroczystość, a w przypadku ścieżek otwartych możemy dodać grot strzałki na początku i na końcu ścieżki, a także ustawić ich kształt i wielkość. Ustawienia te w większości realizowane są przez panel `Properties`. Przeznaczenie poszczególnych przycisków w tym panelu opisano na rys. 10.4. W przypadku przełączników w centralnej części panelu kliknięcie końców widocznej linii powoduje dodanie lub usunięcie grotu na początku i na końcu ścieżki, zaś klikanie rombu (etykieta „wybór rysowanie i wypełnienie”) powoduje przełączanie cykliczne między trybami: rysowanie, rysowanie i wypełnienie, wypełnienie. W przypadku pozostałych przycisków wyświetlana jest lista z wartościami do wyboru lub okno dialogowe do wskazania wartości definiowanej przez użytkownika. Listy wartości predefiniowanych pokazano na rys. 10.5. Jak widać, określenia są typowe dla LaTeX-a (porównaj rozdział 11).

Polecenia ustawiające własności obiektów są dostępne również poprzez menu podręczne. Zawiera ono wszystkie polecenia związane z danym typem obiektów. W przypadku poleceń dublujących przyciski z panelu `Properties` oferują one tylko predefiniowane wartości. Uzupełnieniem panelu są polecenia:

Rys. 10.5. Listy wartości predefiniowanych w panelu `Properties`

- `Line cap` – wybór kształtu zakończenia linii: `butt` – brak, `round` – dodane zaokrąglenie, `square` – dodane kwadratowe zakończenie;
- `Line join` – wybór sposobu łączenia segmentów ścieżki: `miter` – ostry, `round` – zaokrąglony, `bevel` – ścięty;
- `Stroke opacity` – poziom przezroczystości krawędzi;
- `Tiling pattern` – wzór wypełnienia: `normal` – wypełnienie kolorem, `falling` – opadające kreski, `rising` – wznoszące się kreski.

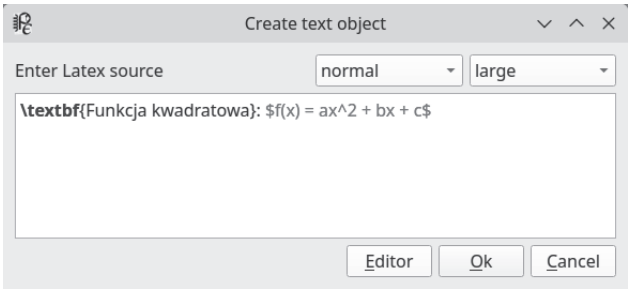
Ustawione właściwości obiektu można przenosić na inny obiekt. Służą do tego polecenia `Pick properties` i `Apply properties` dostępne w menu `Edit` i w menu podręcznym. Oba polecenia pracują na obiekcie będącym podstawowym zaznaczeniem.

Dodając kolejne obiekty do rysunku, układamy je jeden na drugim. Później dodany obiekt przykryje wcześniejsze, jeżeli zostanie umieszczony nad nimi. Polecenia z menu `Edit`: `Front` i `Back` pozwalają przesuwać obiekty na górę lub pod spód. Kolejna para `Group` i `Ungroup` pozwala na łączenie obiektów w grupy i rozdzielenie grupy na indywidualne obiekty. Zgrupowane obiekty możemy później przekształcać jako całość (zob. podrozdział 10.5). Poza grupowaniem kilka otwartych ścieżek możemy połączyć w jedną za pomocą polecenia `Join paths` z menu podręcznego. Wszystkie ścieżki, które chcemy połączyć, muszą być zaznaczone.

Na zakończenie tego podrozdziału wróćmy do polecenia 13 z rys 10.2. Pozwala ono na dodanie znacznika o określonym kształcie i rozmiarze (zob. rys. 10.5) do rysunku. Znaczniki wykorzystujemy przede wszystkim do wyróżnienia istotnych punktów.

10.3 Osadzanie obiektów LaTeX-owych

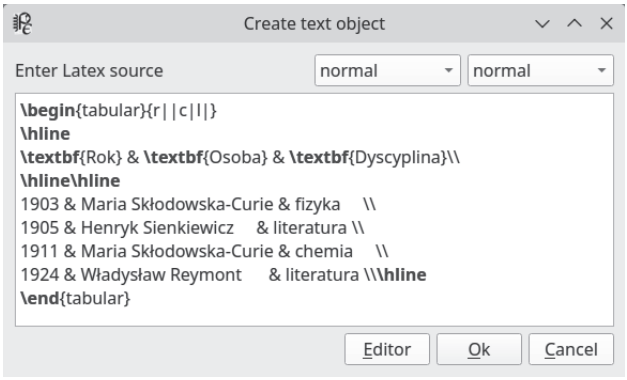
W tym podrozdziale omówimy tryby związane z osadzaniem obiektów tekstowych na rysunkach, czyli skupimy się na trybach 10–12 z rys. 10.2. Jeżeli wybierzemy tryb `Text labels` i klikniemy w dowolnym miejscu obszaru roboczego, to wyświetlone zostanie okno dialogowe pokazane na rys. 10.6, które pozwala na edycję etykiety. Przyciski w górnej części okna pozwalają wybrać tryb tekstowy (`normal`) lub matematyczny (`math`) oraz wielkość czcionki.



Rys. 10.6. Edycja etykiety

Edytując etykietę, możemy korzystać z dowolnych poleceń formatujących tekst, dołączać wzory matematyczne, wstawiać symbole itp. (ewentualnie może

to wymagać dodania kolejnych pakietów do preambuły). Jeżeli nie korzystamy z poleceń ustawiających kolor, to etykieta zostanie dodana w bieżącym kolorze rysowania. Etykieta jest składana w trybie wierszowym, więc możemy dodać cokolwiek, co w tym trybie może się znaleźć. Na rys. 10.7 pokazano przykład etykiety zawierającej środowisko **tabular**.



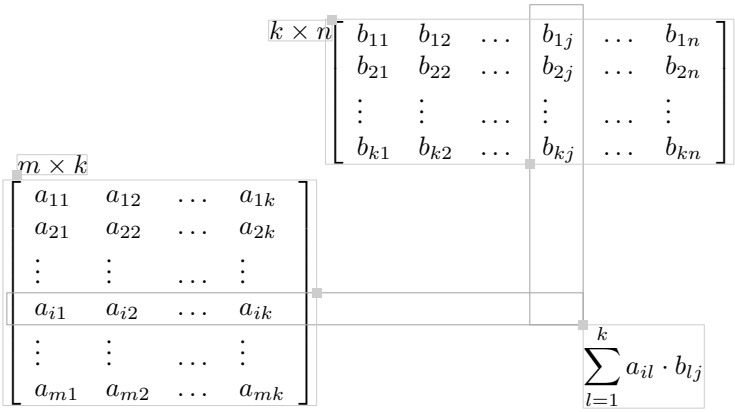
Rys. 10.7. Edycja etykiety zawierającej środowisko **tabular**

Domyślnie etykieta wyrównana jest w lewo w poziomie i do linii bazowej w pionie. Edytor obrazuje to, pokazując dla każdej etykiety jej punkt zaczepienia w postaci zielonego kwadratowego znacznika (nie jest on widoczny w pliku `pdf`). Wyrównanie zaznaczonej etykiety (lub etykiet) można zmienić, korzystając z poleceń `Horizontal alignment` i `Vertical alignment` z menu podręcznego lub menu `Properties`. Wartości wyrównania ustawione w menu `Properties` stają się domyślnymi dla kolejnych dodawanych etykiet. Sposób wyrównania etykiet jest istotny przy ich przesuwaniu (tryb 2 z rys. 10.2). Przesuwając etykietę, tak naprawdę przesuwamy jej punktu zaczepienia.

Tryb `Mathematical symbols` (tryb 11 z rys. 10.2) działa analogicznie jak tryb `Text labels`. Wyświetlane jest to samo okno dialogowe, ale z wybraną opcją `Math`, której nie można zmienić. Edycja odbywa się w trybie matematycznym wewnątrzakapitowym – nie używamy znaków `$`. Jeżeli chcemy uzyskać efekt stylu wystawowego, to możemy użyć polecenia **\displaystyle**.

Na rys. 10.8 przedstawiono ideę mnożenia macierzy. Wszystkie etykiety dodano, używając polecenia `Mathematical symbols`. Na rysunku widać stosowane przez edytor obrysy etykiet tekstowych oraz ich punkty zaczepienia. Różne punkty zakotwiczenia kolejnych etykiet pozwoliły łatwo rozmieścić etykiety zgodnie z oczekiwaniem.

Tryb `Paragraphs` (tryb 12 z rys. 10.2) wstawia obiekt będący ministroną (zob. podrozdział 3.4). Wstawienie obiektu rozpoczynamy od narysowania



Rys. 10.8. Punkty zaczepienia etykiet tekstowych

prostokąta o predefiniowanej wysokości, którego celem jest zdefiniowanie szerokości ministrony i pozycji jej górnej krawędzi. Jej wysokość zostanie ustalona na podstawie zawartości. Szerokość ministrony po wstawieniu można zmienić za pomocą polecenia `Edit→Change text width`. Wewnątrz ministrony możemy umieścić dowolną treść akceptowaną przez to środowisko, możemy mieć akapity tekstu, wypunktowania, tabele (środowisko **`tabular`**, ale nie **`table`**), środowiska matematyczne z numerowanymi wzorami, przypisy itd.

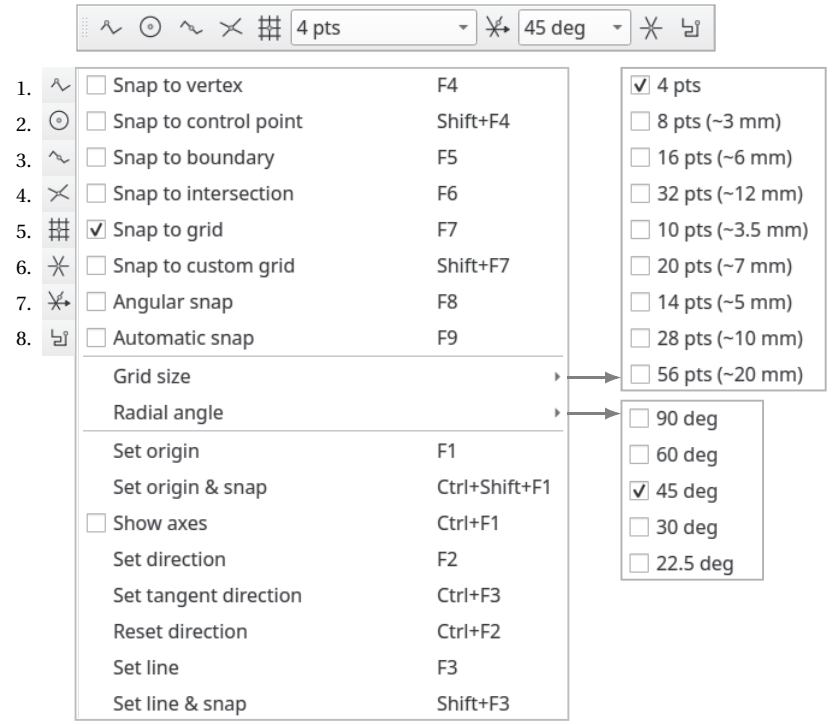
Każdą z wstawionych etykiet (niezależnie od użytego trybu) można ponownie edytować. Wystarczy zaznaczyć odpowiedni obiekt i wybrać polecenie `Edit→Edit object` lub polecenie `Edit text` z menu podręcznego.

10.4 Tryby przyciągania

Ipe oferuje bardzo rozbudowany i różnorodny mechanizm przyciągania, który ułatwia właściwe pozycjonowanie obiektów lub punktów kontrolnych podczas rysowania. Polecenia związane z przełączeniem trybów przyciągania oraz ustawianiem wielkości siatki zebrano w menu `Snap` oraz na pasku narzędziowym `Snap`. Zestawienie obu elementów interfejsu przedstawiono na rys. 10.9. W przypadku przełączników przyciągania istotna jest znajomość klawiszy skrótu, gdyż przyspiesza to pracę, jeśli potrzebujemy zmienić stan przełączników w trakcie rysowania, co jest dość częstą potrzebą. Same przełączniki są od siebie niezależne i możliwe jest ich włączanie i wyłączanie niezależnie od stanu pozostałych przełączników na pasku `Snap`. Jeżeli włączony jest co najmniej jeden tryb przyciągania, to wyświetlany jest dodatkowy znacznik w postaci znaku + w pobliżu kursora myszy. Ten dodatkowy kursor, zwany *Fifi*, wskazuje punkt, do którego

kursor myszy jest w danym momencie przyciągany. Brak kursora *Fifi* oznacza, że jesteśmy za daleko od punktów „magnetycznych”.

Jeżeli jakiś skrót nie działa w Twoim edytorze, to sprawdź, czy dany skrót nie jest używany przez system operacyjny w innym celu, i jeśli to możliwe, zmień te ustawienia.

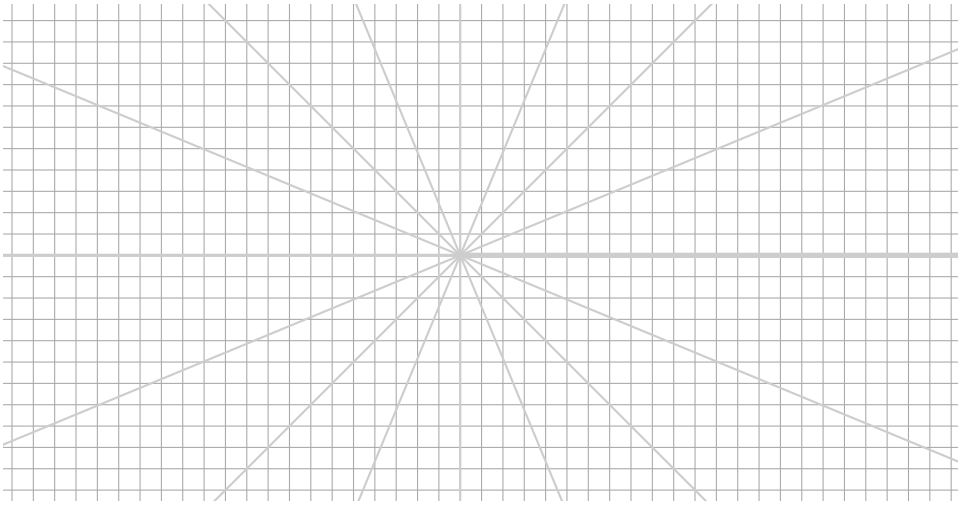


Rys. 10.9. Pasek narzędziowy `Snap` i część menu `Snap`

Podstawowy tryb przyciągania to przyciąganie do siatki (tryb 5 z rys. 10.9). Gęstość siatki możemy ustawić za pomocą polecenia `Snap→Grid size` lub przycisku na pasku `Snap`. Na rys. 10.9 pokazano wszystkie dostępne wielkości siatki. Przyciąganie do siatki zależy wyłącznie od włączenia trybu 5, a nie od widoku siatki. Widok siatki można wyłączyć (`Zoom→Grid visible` lub odpowiadający poleceniu przycisk na pasku `Edit`), ale nie wpływa to na działanie trybu przyciągania do siatki.

Tryby przyciągania 1–4 uruchamiają tzw. przyciąganie kontekstowe. W zależności od trybu różne punkty charakterystyczne obiektów uzyskują „magnetyczne własności”:

– tryb 1 (węzły): wierzchołki wielokątów, węzły łamanych, punty początkowe i końcowe łuków okręgu i splajnów;
– tryb 2 (punkty kontrolne): środki boków wielokątów i segmentów łamanych, narożniki obramowań i punkty zaczepienia obiektów tekstowych, punkty kontrolne splajnów (z wyjątkiem punktów końcowych), środki okręgów i łuków okręgów;
– tryb 3 (brzegi): dowolny punkt brzegowy wielokąta, dowolny punkt łamanej, splajnu, okręgu, łuku okręgu;
– tryb 4 (punkty przecięcia): wierzchołki wielokątów, węzły łamanych z wyjątkiem punktów początkowych, punkt przecięcia dowolnych dwóch linii.



Rys. 10.10. Widok układu współrzędnych przy ustawieniach `Radial angle = 22.5 deg`

Tryby 7 i 8 powiązane są z układem współrzędnych, który używany jest przy realizacji niektórych z transformacji obiektów. W nowym oknie edytora początek układu współrzędnych ulokowany jest w lewym dolnym narożniku kanwy, a jego osie są równoległe do krawędzi kanwy. Układ możemy wyświetlić, korzystając z polecenia `Snap→Show axes` (zob. rys. 10.9) lub klawisza skrótu `Ctrl+F1` (ta sama kombinacja ukrywa widok układu współrzędnych). Jest on wyświetlany w postaci pęku prostych przechodzących przez początek układu współrzędnych. Są one wyświetlane w zielonym kolorze, przy czym półprosta reprezentująca oś x jest wyróżniona pogrubioną linią. Liczba wyświetlanych prostych zależy od ustawień opcji `Snap→Radial angle` (opcja dostępna jest również na pasku narzędziowym `Snap`). Początek układu współrzędnych możemy

ustawić w miejscu kursora myszy (lub punktu, do którego jest przyciągany, jeśli włączony jest tryb przyciągania) za pomocą klawisza skrótu `F1`. Widok układu współrzędnych przy ustawieniach `Radial angle = 22.5 deg` pokazano na rys. 10.10. Proste są rozmieszczone co 22,5 stopnia, licząc od osi x przeciwnie do ruchu wskazówek zegara. Przy ustawieniach `Radial angle = 90 deg` zobaczymy tylko dwie prostopadłe proste.

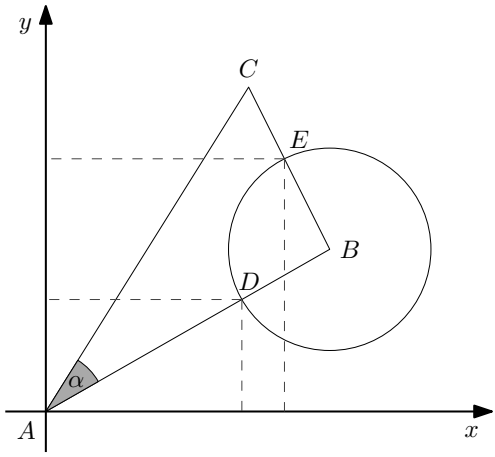
Po ustawieniu początku układu współrzędnych możemy ustawić również oś x. W tym celu wskazujemy punkt, przez który ma przechodzić oś, i korzystamy z klawisza skrótu `F2`. Oś x zostanie poprowadzona od początku układu współrzędnych przez wskazany punkt. Zmiana wyłącznie położenia początku układu współrzędnych (skrót `F1`) powoduje przesunięcie układu o wektor wyznaczony przez stary i nowy początek układu współrzędnych (oś x po zmianie jest równoległa do wcześniejszej osi x). Polecenie `Snap→Reset direction` powoduje ustawienie osi x poziomo bez zmiany początku układu współrzędnych.

Włączenie trybu przyciągania 7 (zob. rys. 10.9) powoduje, że kursor jest przyciągany przez proste z pęku prostych reprezentujących układ współrzędnych. Przy korzystaniu z tego trybu wygodnie jest mieć włączone wyświetlanie układu (jego ukrycie powoduje również wyłączenie trybu 7). Drugi z trybów wykorzystujących układ współrzędnych (tryb 8) jest aktywny tylko w trybach rysowania 18–20 (zob. rys. 10.2), tj. rysowaniu linii i łamanych, wielokątów i splajnów (otwartych). W takim przypadku rysowanie kolejnego segmentu łamanej, boku wielokąta lub umieszczenie kolejnego punktu kontrolnego splajnu odbywa się tylko zgodnie z kierunkami wyznaczonymi przez pęk prostych układu współrzędnych. Innymi słowy, w efekcie rysowania z włączonym trybem przyciągania 8 wszystkie segmenty łamanej, wszystkie boki wielokąta i wszystkie linie pomocnicze splajnu (odcinki łączące kolejne punkty kontrolne widoczne w trybie edycji) będą równoległe do jednej z prostych z pęku reprezentującego układ współrzędnych. Tryb przyciągania 8 jest szczególnie przydatny przy ustawieniu `Radial angle = 90 deg` i rysowaniu linii lub łamanych. W takim przypadku kreślone są wyłącznie pionowe lub poziome segmenty. Tryb przyciągania 6 wymaga przygotowania własnej siatki. Temat ten omówię w podrozdziale 10.7.

Tryby przyciągania mogą być włączane niezależnie od siebie. Możemy włączyć nawet wszystkie jednocześnie, ale nie oznacza to, że wszystkie będą aktywne. Zachodzą między nimi pewne interakcje, które dezaktywują działanie niektórych trybów w towarzystwie innych. Jeżeli włączony jest tryb 7, to nieaktywne jest przyciąganie do siatki, a przyciąganie do punktów brzegowych dotyczy tylko punktów brzegowych leżących na prostych z pęku prostych układu współrzędnych.

Przy włączonych wszystkich trybach 1–5 (i tylko tych) Ipe działa według następujących zasad. W pierwszej kolejności szukany jest wierzchołek, punkt

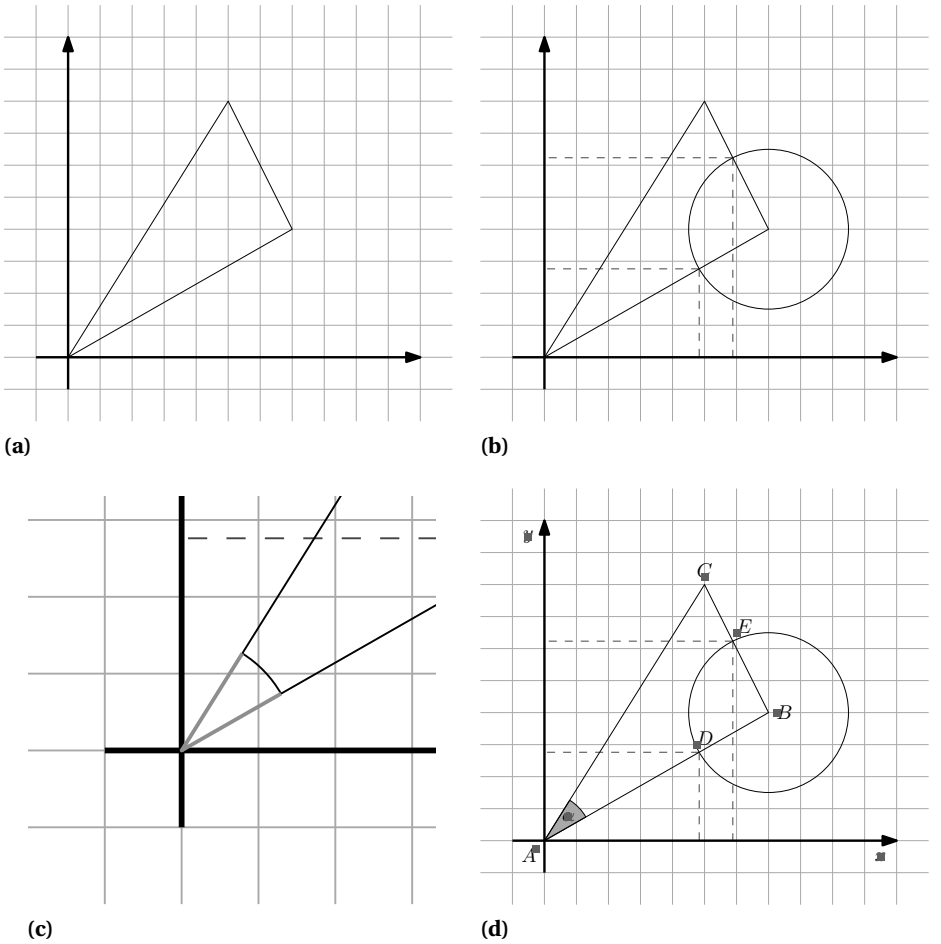
kontrolny lub punkt przecięcia, który jest wystarczająco blisko. Jeżeli nie ma takiego punktu, wyszukiwany jest najbliższy punkt na krawędzi. Jeżeli również takiego punktu nie ma wystarczająco blisko, to stosowane jest przyciąganie do siatki. Może to prowadzić do sytuacji, gdy kursor myszy przyciągany jest do punktu przecięcia dwóch linii, który jest wystarczająco blisko, ale dalej niż najbliższy węzeł siatki lub punkt brzegowy.



Rys. 10.11. Finalna wersja grafiki, którą chcemy przygotować

W celu zademonstrowania użyteczności poszczególnych trybów przyciągania przeanalizujemy krok po kroku przygotowanie grafiki pokazanej na rys. 10.11. Zaczynamy w trybie przyciągania do siatki (tryb 5) przy domyślnych ustawieniach (siatka 16 punktów). Rysujemy odcinek (od lewej do prawej), który będzie osią x. Następnie w panelu `Properties` dodajemy grot i ustawiamy grubość linii na `heavier`. Przy tych ustawieniach rysujemy oś y. Po narysowaniu drugiej osi jest ona zaznaczona. Jeżeli chcemy zmienić ustawienia przed rysowaniem trójkąta, to musimy wyłączyć to zaznaczenie. Można to zrobić za pomocą polecenia `Edit→Deselect all` lub klikając pusty obszar na kanwie przy wciśniętym klawiszu `Ctrl`. Wybieramy normalną grubość linii i wyłączamy grot. Następnie rysujemy nasz trójkąt. Efekt dotychczasowych prac powinien być taki jak na rys. 10.12a.

Wyłączamy przyciąganie do siatki i włączamy tryb przyciągania 1 (wierzchołki). Wybieramy tryb rysowania okręgu poprzez wskazanie środka i promienia (tryb 25 z rys. 10.2). Wybieramy jako środek wierzchołek trójkąta i rysujemy okrąg. Wyłączone przyciąganie do siatki powoduje, że mamy pełną swobodę w doborze promienia. Został on tutaj wybrany tak, aby punkty przecięcia okręgu i trójkąta nie znajdowały się w węzłach siatki. Następnie ustawiamy



Rys. 10.12. Etapy pracy nad rysunkiem

opcję `Radial angle` na 90 stopni i włączamy tryby przyciągania 3, 4 i 8. Ustawiamy tryb rysowania linii na wartość `dashed` i ustawiamy kolor rysowania na ciemny szary. Wybieramy dowolny kolor (zob. rys. 10.4) i ustawiamy taki odcień, jaki nam odpowiada. Rysujemy teraz cztery odcinki wskazujące rzuty punktów przecięcia na osie układu. Gdy przesuniemy kursor w pobliże punktu przecięcia, zostanie do niego przyciągnięty. Klikamy (`LP`) i przesuwamy kursor w kierunku osi x. Ze względu na tryb 8 i ustawienia `Radial angle` odcinek jest kreślony pionowo. Gdy kursor zostanie przyciągnięty do linii reprezentującej oś x, dodajemy drugi koniec odcinka (`PP`). Podobnie rysujemy pozostałe trzy odcinki. Efekt prac w tym etapie widzimy na rys. 10.12b.

Usuwamy ewentualne zaznaczenia i ustawiamy kolor rysowania na czarny, a tryb rysowania linii na `normal`. Włączamy tryby przyciągania 1 i 3 (wierzchołki i krawędzie). Wybieramy tryb 22 rysowania łuków okręgu (zob. rys. 10.2). Wskazujemy początek naszego układu współrzędnych jako środek okręgu, wybieramy punkt na jednym boku (działa tutaj przyciąganie do krawędzi) i na drugim boku. Ponieważ chcemy zamalować „kąt" szarym kolorem, to nie wystarczy sam łuk. Dodajemy łamaną złożoną z dwóch segmentów, tak jak to pokazano na rys. 10.12c. Wybieramy tryb zaznaczania (tryb 1) i z wciśniętym klawiszem `Shift` dodajemy do zaznaczenia wcześniej narysowany łuk. Z menu podręcznego wybieramy polecenie `Join paths`, co połączy obie ścieżki w wycinek koła. Teraz możemy wybrać kolor wypełnienia i tryb rysowania z wypełnieniem.

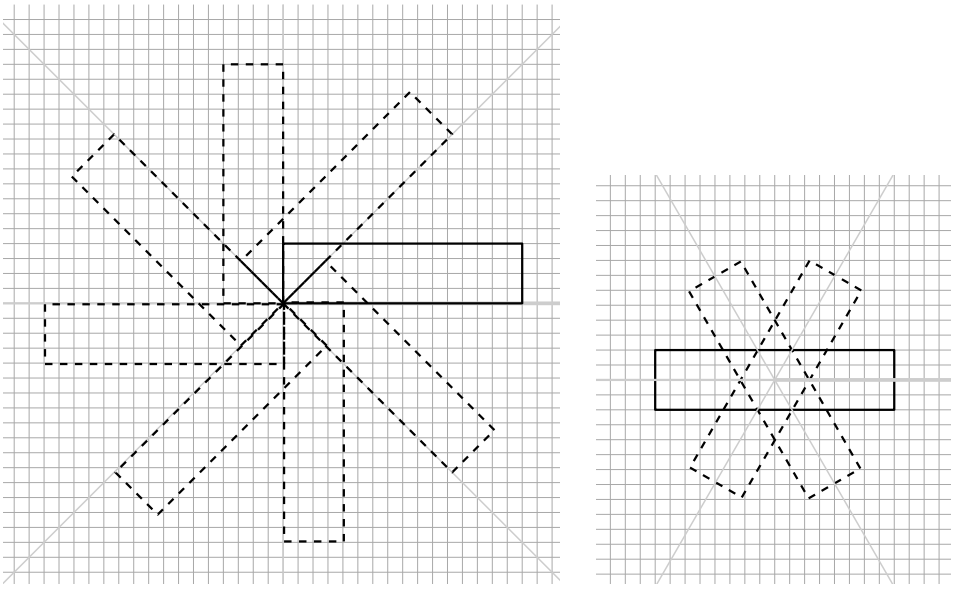
Włączamy tryb przyciągania do siatki (tryb 5) i ustawiamy siatkę na 4 punkty. Korzystając z etykiet matematycznych, dodajemy wszystkie etykiety. Można je dodać w dowolnym miejscu, a po ustawieniu właściwego punktu zaczepienia możemy je przesunąć na miejsce docelowe. Efekt ostatniego etapu prac widzimy na rys. 10.12d. Na rysunku umieszczono dodatkowe szare znaczniki, wskazujące punkty zaczepienia etykiet. Nie są one częścią finalnego rysunku.

10.5 Transformacje obiektów

Tryby pracy 2–5 z rys. 10.2 dotyczą transformacji istniejących obiektów. Domyślnie operacje transformacji wykonywane są na zaznaczonych obiektach. Ipe wprowadza udogodnienie, które pozwala przyspieszyć pracę. Jeżeli żaden obiekt nie jest zaznaczony, to operacja transformacji będzie dotyczyć obiektu najbliższego kursorowi myszy w momencie, w którym wciśnięty został lewy klawisz. Dodatkowo edytor udostępnia kilka kombinacji klawiszy z operacjami myszą dotyczących zaznaczania i przesuwania obiektów. Niezależnie od wybranego trybu pracy obiekty możemy przesunąć myszą przy wciśniętym klawiszu `Alt` (lewy), bez potrzeby przełączania w tryb 2. Obiekty możemy zaznaczać, klikając je lewym przyciskiem myszy przy wciśniętym klawiszu `Ctrl`. Kliknięcie obiektu przy wciśniętej kombinacji `Shift+Ctrl` dodaje obiekt do bieżącego zaznaczenia. Przy wciśniętym środkowym przycisku myszy (ewentualnie wciśnięta rolka) możemy przesuwać całą kanwę, aby wyświetlić inny jej obszar. Do przesuwania kawny służy również tryb 7 (zob. rys. 10.2).

Tryb 2 służy do przesuwania obiektów. Jeżeli chcemy jednocześnie przesuwać więcej niż jeden obiekt, to muszą one być zaznaczone. Jeżeli przy przesuwaniu przytrzymamy dodatkowo wciśnięty klawisz `Shift`, to obiekty przesuwane będą wyłącznie w pionie lub w poziomie. Bardzo precyzyjne przesuwanie można realizować za pomocą ipeletów. Omówię je w podrozdziale 10.6.

Tryb 3 służy do obracania obiektów. Jeżeli nie jest ustawiony układ współrzędnych (nie jest wyświetlany), to zaznaczone obiekty są obracane względem środka obszaru ograniczającego te obiekty. Jeżeli układ współrzędnych jest ustawiony, to obiekty są obracane względem początku układu współrzędnych. Przy ustawieniach domyślnych nie jest możliwe obracanie etykiet tekstowych. Jeżeli chcemy to zmienić, to po zaznaczeniu etykiety należy wybrać polecenie `Properties→Transformations` i wybrać wartość `affine`. Polecenie `Transformations` dostępne jest również w menu podręcznym etykiet.



(a) **(b)**

Rys. 10.13. Wykorzystanie przyciągania `Angular snap` do realizacji obrotów

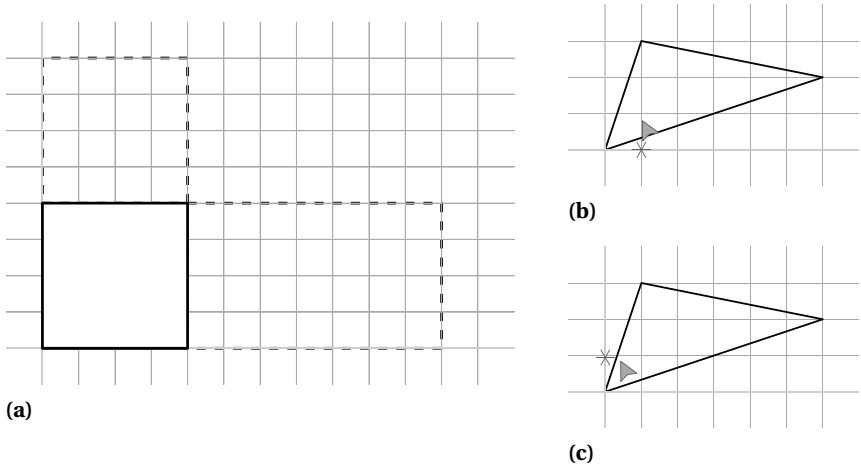
Do realizacji precyzyjnych obrotów możemy wykorzystać przyciąganie do linii z pęku prostych reprezentujących układ współrzędnych (tryb 7, zob. rys. 10.9). Przykład taki pokazano na rys. 10.13a. Początek układu współrzędnych został ustawiony w lewym dolnym narożniku oryginalnego prostokąta (rysowany linią ciągłą). Przy ustawieniu opcji `Radial angle` na 45 stopni i włączeniu tylko trybu przyciągania 7 obracanie prostokąta prowadzi do jednego z położeń przedstawionych za pomocą przerywanej linii. Na rys. 10.13b pokazano możliwość obracania tego samego prostokąta, jeśli początek układu został umieszczony w jego środku, a opcja `Radial angle` ustawiona na 60 stopni.

Przedstawiony sposób obracania ze wsparciem układu współrzędnych możemy realizować, gdy odpowiada nam jeden z kątów będących wynikiem

ustawienia opcji `Radial angle`. Edytor udostępnia również ipelet `Ipelets→ Goodies→Precise rotate`, który pozwala obrócić obiekt (obiekty) o podaną liczbę stopni. Ipelet można uruchomić za pomocą skrótu `Ctrl+R`.

Za pomocą trybu 4 możemy realizować skalowanie. Jeżeli operacja jest realizowana z wciśniętym klawiszem `Shift`, to uzyskujemy jednokładność (skala względem osi x i y jest taka sama). Jeżeli nie jest ustawiony układ współrzędnych, to operacja jest realizowana z wykorzystaniem jednego z narożników obszaru ograniczającego zaznaczone obiekty jako punktu stałego przekształcenia (kierunek przekształcania decyduje o tym, który to narożnik). W przeciwnym przypadku punktem stałym jest początek układu współrzędnych.

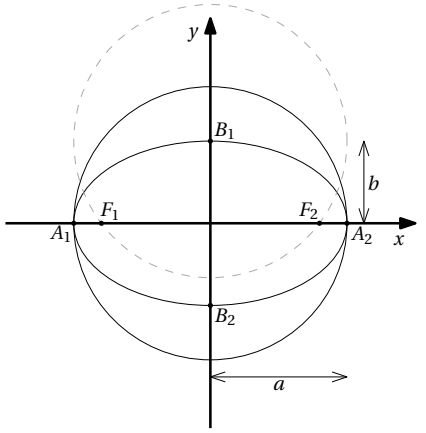
Jeżeli chcemy uzyskać skalowanie wyłącznie względem jednej osi, to możemy to zrealizować przy ustawionym układzie współrzędnych, opcji `Radial angle` o wartości 90 stopni i trybie przyciągania 7. Przy realizacji przekształcenia możliwe będzie wówczas przesuwanie kursora wyłącznie względem jednej z prostopadłych prostych. Skalowanie będzie realizowane względem tej prostej, do której przyciągany jest kursor przy rozpoczęciu przekształcenia: rys. 10.14b – skalowanie względem prostej poziomej, 10.14c – skalowanie względem prostej pionowej. Efekty przekształcenia kwadratu względem obu prostych pokazano na rys. 10.14a (prostokąty kreślone przerywaną linią).



Rys. 10.14. Rozciąganie względem jednej osi

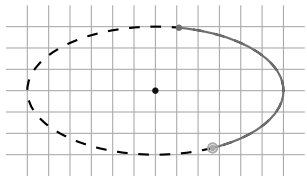
Skalowanie możemy wykorzystać, aby uzyskać elipsę w Ipe, co pokazano na rys. 10.15. Jedna z możliwych procedur wygląda następująco. Kreślimy okrąg o promieniu równym długiej osi elipsy (a). Ustawiamy początek układu współrzędnych w środku okręgu. Skalujemy okrąg w pionie, tak aby odległość od

punktu B_1 do początku układu współrzędnych wynosiła b. Jeżeli chcemy wyznaczyć ogniska elipsy, to wystarczy wykreślić okrąg o środku B_1 i promieniu a (na rysunku wykreślony przerywaną linią). W punktach przecięcia okręgu i osi x znajdują się ogniska elipsy (punkty F_1 i F_2). Analogiczną procedurę można zrealizować, skalując okrąg względem osi x. Ponadto uzyskaną elipsę możemy obrócić i przesunąć na docelowe miejsce.



Rys. 10.15. Wykorzystanie skalowania względem jednej osi do uzyskania elipsy

Edytor udostępnia również ipelet `Ipelets→Goodies→Precise stretch`, który pozwala przeskalować obiekt według podanej liczbowo skali. Ipelet można uruchomić za pomocą skrótu `Ctrl+K`. Jeżeli w oknie dialogowym podamy jedną liczbę, to zrealizowana zostanie jednokładność o podanej skali. Jeżeli podamy dwie liczby oddzielone spacją, to będą one traktowane jako współczynniki skalowania odpowiednio względem osi x i y. Ipeletu można użyć, aby szybko uzyskać elipsę o zadanej proporcji osi b do osi a. Zakładając, że oś a jest ułożona poziomo jak na rys. 10.15 i oś b ma być o połowę mniejsza, to po narysowaniu okręgu o promieniu a wystarczy wywołać ipelet z parametrami `1 0.5`.

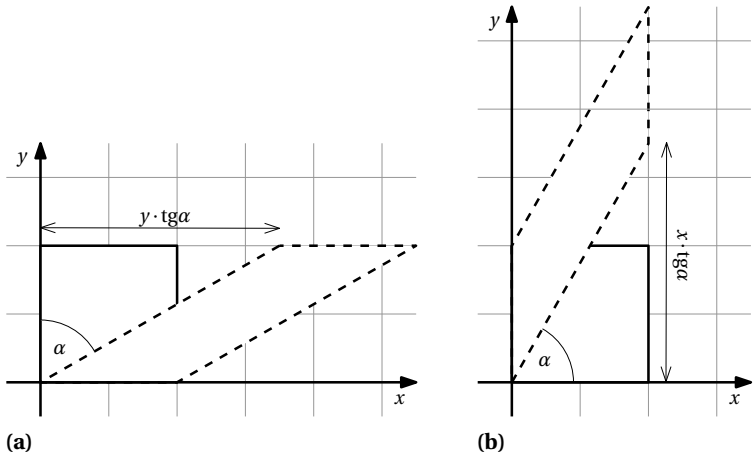


Rys. 10.16. Przycinanie elipsy do łuku eliptycznego o ustalonych końcach

Jeżeli chcemy uzyskać łuk elipsy, to po jej narysowaniu zaznaczamy tryb przyciągania 3 i przechodzimy w tryb edycji (`Ctrl+E`). Z menu podręcznego

wybieramy opcję `Cut ellipse` i klikamy lewym przyciskiem myszy. Po tej operacji wyświetlane są dwa różowe znaczniki, które możemy przesuwać, ustawiając końce łuku, który chcemy pozostawić (zob. rys. 10.16). Po zatwierdzeniu operacji z elipsy pozostanie tylko wskazany łuk. Analogiczną operację możemy wykonać na okręgu, który jest szczególnym przypadkiem elipsy.

Tryb 5 służy do nachylania obiektów. Ideę nachylania przedstawiono na rys. 10.17. Ipe realizuje tylko nachylenie względem osi x. Jeżeli chcemy uzyskać efekt nachylenia względem osi y (zob. rys. 10.17b), należy ustawić oś x pionowo.

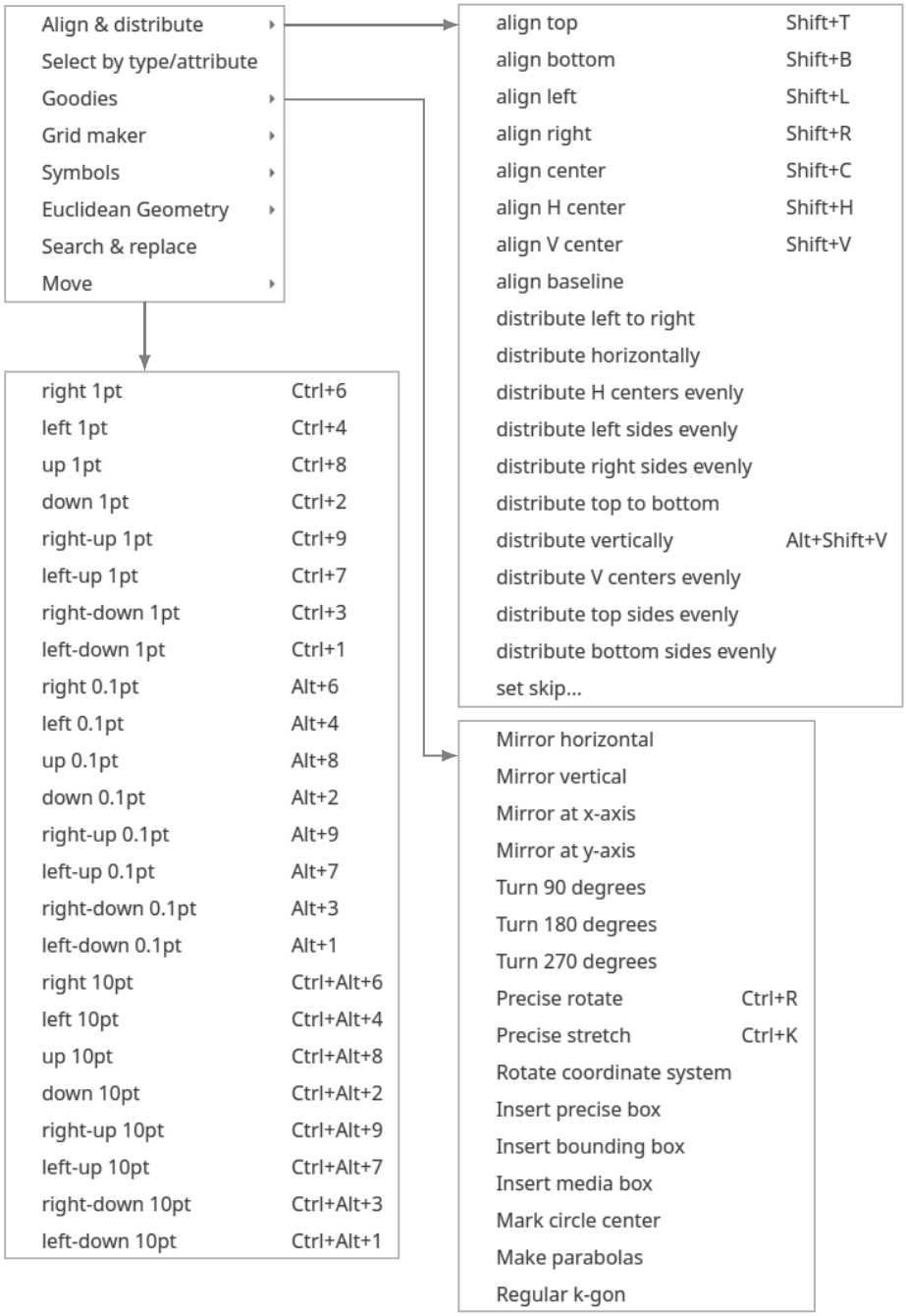


Rys. 10.17. Nachylenie **(a)** względem osi x, **(b)** względem osi y

10.6 Ipelety

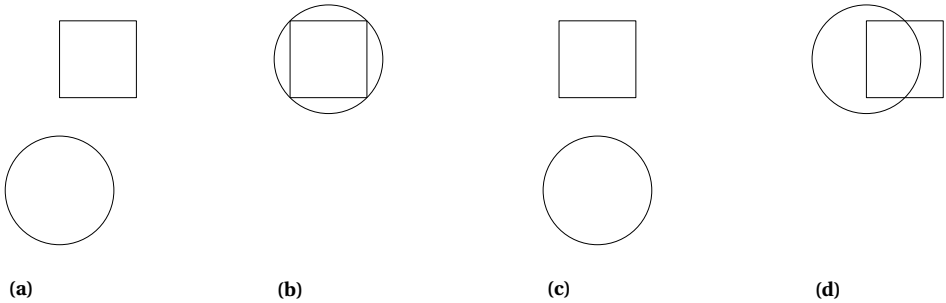
Edytor Ipe jest rozszerzalny. Możliwe jest implementowanie dodatkowych funkcjonalności i dołączanie ich do edytora. Takie dodatkowe funkcje nazywane są *ipeletami*. Edytor dopuszcza implementowanie ipeletów w językach Lua i C++. Więcej informacji na ten temat można znaleźć pod adresem `https://ipe.otfried.org/ipelib/`. Znaczną liczbę zarejestrowanych ipeletów zebrano w menu `Ipelets`. Wybrane listy poleceń przedstawiono na rys. 10.18.

Ipelety zebrane w menu `Align & distribute` pozwalają na szybkie wyrównanie położenia wielu zaznaczonych obiektów. Wyrównanie realizowane jest względem obiektu, który stanowi podstawowe zaznaczenie. Jeżeli więc wybierzemy wyrównanie do górnej krawędzi, to górna krawędź obszaru ograniczającego obiekt, który stanowi podstawowe zaznaczenie, będzie linią, do której wyrównane będą górne krawędzie obszarów ograniczających zaznaczone obiekty.



Rys. 10.18. Wybrane listy ipeletów

Należy pamiętać, że w przypadku splajnów obszar ograniczający jest zazwyczaj wyznaczony przez punkty kontrolne.



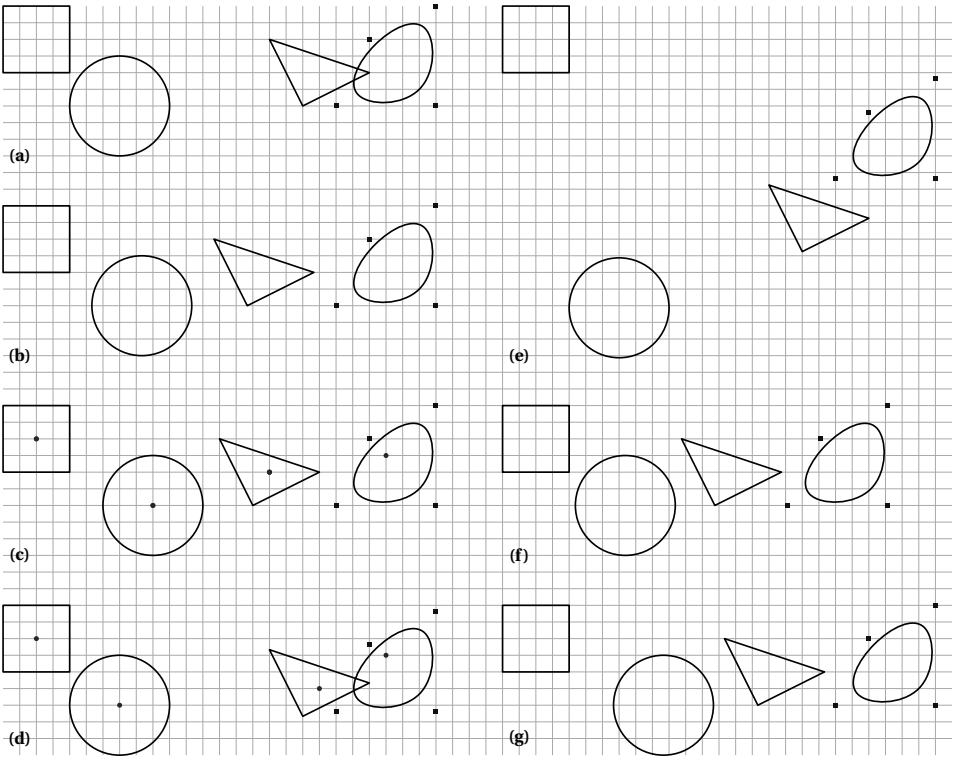
Rys. 10.19. Wyrównanie: **(a)** wyjściowe położenie obiektów, **(b)** `align center`, **(c)** `align H center`, **(d)** `align V center`

Na rys. 10.19 pokazano różnice między różnymi ipeletami związanymi z wyrównaniem do środka. W każdym przypadku podstawowe zaznaczanie stanowił kwadrat. Wyrównanie do linii bazowej (`aline baseline`) jest stosowane do wyrównywania etykiet tekstowych. Warto zwrócić uwagę, że ipelet zaprezentowany na rys. 10.19b pozwala łatwo umieścić etykietą tekstową w środku obiektu stanowiącego podstawowe zaznaczenie.

Na rys. 10.20 pokazano różnice między wybranymi ipeletami rozmieszczającymi elementy. Na wszystkich rysunkach zaznaczono punkty kontrolne użytego splajnu. Dodatkowo na rysunkach 10.20c i 10.20d dodano znaczniki wskazujące środki obiektów. Użyta w tle siatka pozwala dokładnie przeanalizować działanie poszczególnych poleceń.

Menu `Goodies` zawiera m.in. ipelety realizujące: odbicia symetryczne względem prostej (polecenia `Mirror...`), obroty o wielokrotność kąta 90 stopni (polecenia `Turn...`), omawiane już polecenia precyzyjnego obrotu i skalowania. Wśród pozostałych poleceń warto zwrócić uwagę na: `Rotate coordinate system` – obrót układ współrzędnych o zadaną liczbę stopni, `Insert bounding box` – dodanie prostokąta stanowiącego granice obszaru ograniczającego dany obiekt, `Regular k-gon` – rysowanie wielokątów foremnych. W przypadku ostatniego z poleceń należy zaznaczyć okrąg i wywołać polecenie. W okrąg zostanie wpisany wielokąt foremny o podanej liczbie wierzchołków.

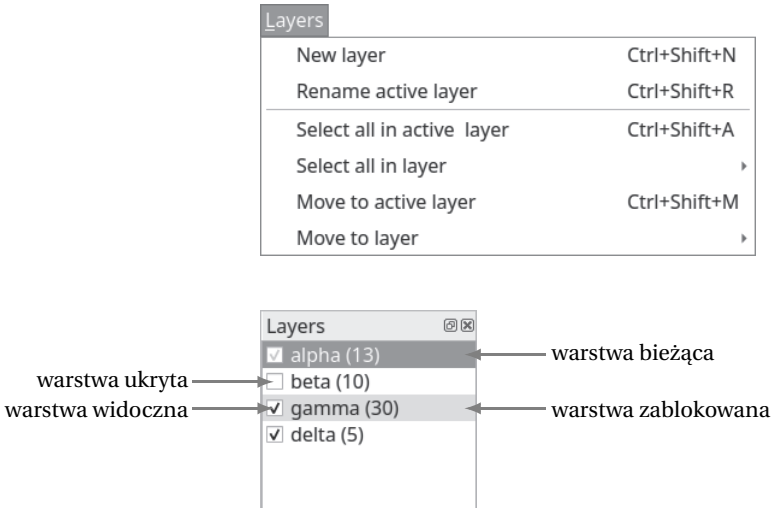
Menu `Move` nie wymaga komentarza. Mamy tutaj zebrane polecenia pozwalające na bardzo precyzyjne przesuwanie obiektów. Więcej ipeletów, które można doinstalować, znajduje się na stronie `https://github.com/otfried/ipe-wiki/wiki/Ipelets`.



Rys. 10.20. Rozmieszczenie obiektów: **(a)** wyjściowe położenie obiektów, **(b)** `distribute horizontally`, **(c)** `distribute H centers evenly`, **(d)** `distribute V centers evenly`, **(e)** `distribute top to bottom`, **(f)** `distribute left to right`, **(g)** `distribute left sides evenly`

10.7 Wiele warstw

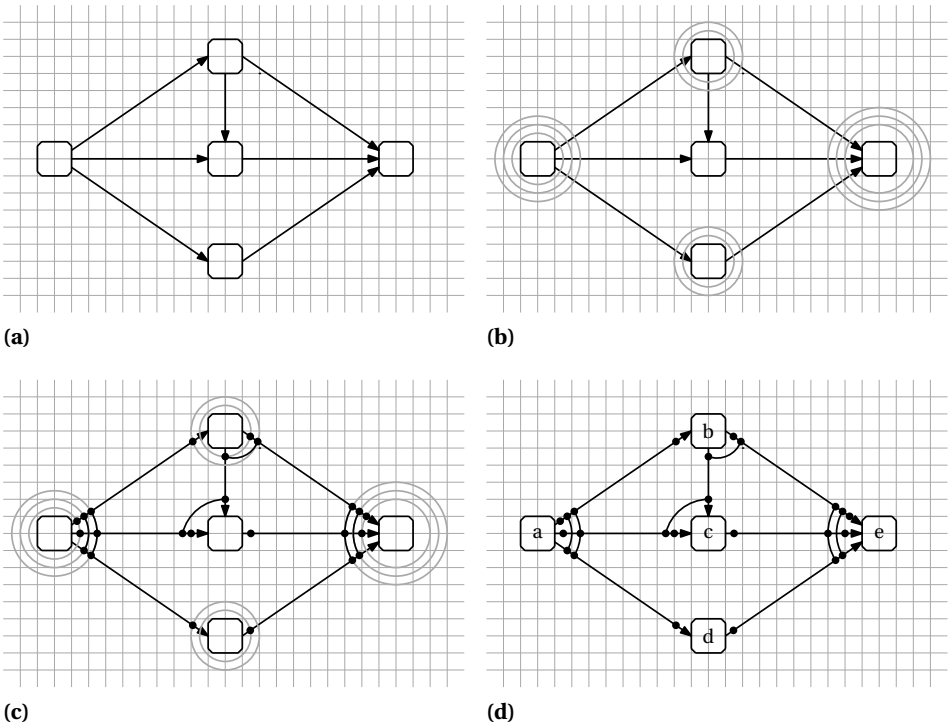
Tworząc rysunek w Ipe, możemy korzystać z wielu warstw. Każdy dodawany obiekt należy do jakiejś warstwy. Domyślnie rysunek składa się z jednej warstwy o nazwie `alpha` (nazwę możemy zmienić). Przynależność do warstwy jest całkowicie niezależna od „głębokości", na której znajduje się obiekt (ile obiektów leży nad nim). Obiekty możemy łatwo przenosić między warstwami. Kolejność warstw w panelu `Layers` nie ma większego znaczenia. Dla wyglądu finalnego rysunku istotne jest, czy warstwa jest zaznaczona jako widoczna. Pozwala to na dodawanie pomocniczych obiektów, które możemy umieścić na warstwie, którą później ukryjemy. Warstwę możemy zablokować, co chroni obiekty do niej należące przed przypadkową zmianą. Poza tym warstwy możemy wyłączać tymczasowo, aby ułatwić sobie pracę, jeżeli rysunek zawiera dużo obiektów. Łatwo również zaznaczać wszystkie obiekty z wybranej warstwy.



Rys. 10.21. Menu `Layers` i panel `Layers`

Na rys. 10.21 przedstawiono menu `Layers`, które m.in. zawiera polecenie pozwalające dodać nową warstwę do rysunku oraz panel `Layers`, służący do bieżącego zarządzania warstwami. Warstwę możemy usunąć, korzystając z jej menu podręcznego. Aktywną warstwę wybieramy, klikając jej nazwę w panelu. Jest ona podświetlona (kolor podświetlenia zależy od ustawień systemowych). Nowe obiekty są dodawane na bieżącej warstwie. Widoczność warstwy jest zależna od obecności znacznika w polu na lewo od jej nazwy. Warstwę możemy zablokować, korzystając z polecenia `Lock...` w menu podręcznym, a odblokować za pomocą polecenia `Unlock...`. Nie można zablokować aktywnej warstwy i nie można warstwy zablokowanej uczynić aktywną. Zablokowane warstwy również są wyróżnione przez podświetlenie (inny kolor niż podświetlenie warstwy bieżącej). Liczba w nawiasie okrągłym obok nazwy warstwy wskazuje liczbę obiektów na tej warstwie.

Rozważmy przykład przedstawiony na rys. 10.22. Pierwsza warstwa (rys. 10.22a) zawiera obiekty tworzące strukturę grafu (węzły i krawędzie). Na drugiej warstwie zostały dodane pomocnicze okręgi, które w dalszej części wykorzystano do ulokowania znaczników w punktach przecięcia krawędzi grafu i okręgów. Łączny widok dwóch pierwszych warstw pokazano na rys. 10.22b. W trzeciej warstwie dodane zostały znaczniki i łączące je łuki okręgów (trzy pierwsze warstwy widoczne są na rys. 10.22c). W ostatniej warstwie umieszczono etykiety tekstowe. Taki zabieg pozwala tymczasowo ukryć wszystkie inne warstwy i łatwo edytować etykiety, gdyby zaszła taka konieczność. Finalny model, po ukryciu warstwy `beta`, pokazano na rys. 10.22d.



Rys. 10.22. Wykorzystanie warstw przy konstruowaniu modelu

Z warstwami Ipe połączona jest koncepcja widoków, które odpowiadają scenom z pakietu `tikz` (zob. rozdział 11). Domyślnie mamy tylko jeden widok, tj. finalny rysunek jest traktowany jako jedna strona w pliku `pdf`. Jeżeli przygotowujemy grafikę na potrzeby prezentacji, to widoki pozwalają na odsłanianie rysunku w wielu krokach (widokach). W pliku `pdf` kolejne widoki są wyświetlane jako kolejne strony. Jeżeli chcemy dodać do dokumentu daną stronę rysunku, to korzystamy z opcji `page` polecenia **`\includegraphics`**, np. **`\includegraphics`**`[page=2]{model.pdf}`.

Nowy widok możemy dodać, korzystając z polecenia `Views→New view`. Jeżeli mamy zdefiniowany więcej niż jeden widok, to informacja o bieżącym widoku wyświetlana jest na dole panelu `Properties`. Menu `Views` zawiera polecenia pozwalające przemieszczać się między widokami, przy czym łatwo to realizować, korzystając z klawiszy skrótu `Home` (pierwszy widok), `End` (ostatni widok), `PgUp` (poprzedni widok) i `PgDn` (następny widok). Po przełączeniu się na dany widok możemy w panelu `Layers` ustawić, które warstwy są widoczne w danym widoku (ukrywamy warstwy, które nie mają być wyświetlane). Widzimy tutaj jeszcze

jedno zastosowanie warstw – obiekty, które mają być odsłaniane w kolejnych widokach, umieszczamy na kolejnych warstwach.

Warto umieszczać na oddzielnych warstwach małe elementy, jak znaczniki i krótkie (np. jednoliterowe) etykiety tekstowe. Jeżeli zachodzi potrzeba zmiany ustawień takich obiektów, to łatwo zaznaczyć elementy należące do oddzielnej warstwy.

Ipe udostępnia specjalne podejście do pracy z grupami. Często grupa powstaje jako połączenie jakiegoś kształtu i etykiety tekstowej, np. nazwy umieszczonej wewnątrz węzła grafu. Jeżeli zastosujemy polecenie `Edit→Edit object` (`Ctrl+E`) do grupy zawierającej co najmniej jedną etykietę tekstową, to edytowana będzie etykieta znajdująca się najwyżej w tej grupie.

Z drugiej strony możliwa jest edycja obiektów wewnątrz grupy bez konieczności rozgrupowania obiektów. Pracę wewnątrz grupy rozpoczynamy, wybierając polecenie `Edit→Edit group` (to samo polecenie dostępne jest w menu kontekstowym grupy). Edycja grupy realizowana jest na tymczasowej warstwie `EDIT-GROUP`, na której obiekty z edytowanej grupy są tymczasowo rozgrupowane, a jednocześnie wszystkie inne warstwy zostają zablokowane. W tym momencie możemy zaznaczyć dowolny obiekt z naszej grupy i go edytować. Pracę z grupą kończymy, wybierając polecenie `Edit→End group edit`. W efekcie wszystkie obiekty zostają ponownie zgrupowane i przeniesione na oryginalną warstwę, a warstwa tymczasowa zostaje usunięta. Jeżeli grupa zawiera w sobie inną grupę, to możemy edytować grupy rekurencyjnie. Po wydaniu polecenia `Edit group` możemy zaznaczyć grupę wewnętrzną i ponownie wybrać `Edit group` itd. Z każdym kolejnym użyciem polecenia `Edit group` dodana zostanie kolejna tymczasowa warstwa. Każde polecenie `End group edit` zamyka jedną taką tymczasową warstwę (najgłębszą).

Grupy mają jeszcze jedną ciekawą właściwość. Ipe pozwala zdefiniować akcję powiązaną z daną grupą. Jest to realizowane przez polecenie `Set link action` z menu podręcznego grupy. Jeżeli później otworzymy nasz rysunek w przeglądarce plików `pdf`, to kliknięcie wewnątrz obszaru ograniczającego grupę wyzwoli tę akcję. Użycie polecenia `Set link action` otwiera okno dialogowe, w którym możemy wprowadzić definicję naszej akcji. Ipe oferuje m.in. następujące możliwości:

– adres strony internetowej, np. `https://ctan.org/pkg/latex` – otwarcie adresu w domyślnej przeglądarce;

– uruchomienie aplikacji i otworzenie pliku, np. `launch:movie.mp4`, `launch:music.mp3`, `launch:kwiaty.png`, `launch:dokument.txt` – uruchomienie domyślnej aplikacji i odtworzenie pliku multimedialnego lub

wyświetlenie pliku z grafiką, dokumentu `pdf`, otwarcie pliku tekstowego w edytorze itp. (tutaj zakładamy, że pliki, których dotyczy akcja, są w tym samym katalogu co nasz rysunek `pdf`);

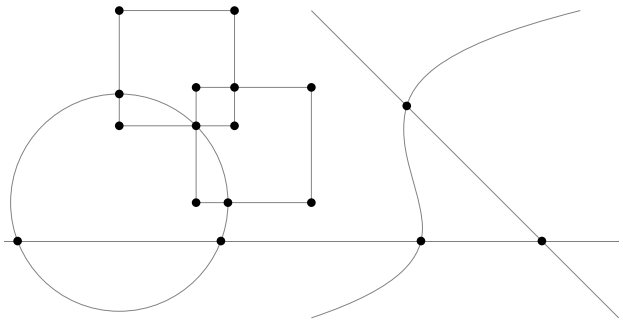
– poruszanie się między stronami, np. `named:NextPage` (inne: `PrevPage`, `FirstPage`. `LastPage`).

Jeżeli chcemy zmienić akcję, to wybieramy polecenie `Link action...` z menu podręcznego. Wyświetlone zostanie okno z bieżącą wartością, którą możemy zmienić.

Ipe umożliwia poza tworzeniem grafiki również przygotowanie prezentacji. W takim przypadku uzyskujemy wielostronicowy plik `pdf` (każdy slajd jest na oddzielnej stronie, dodatkowo strona może mieć wiele widoków), w którym opisane powyżej akcje będą działały. Jeżeli przygotowany dokument `pdf` osadzimy w innym dokumencie LaTeX-a, to niestety zdefiniowane akcje przestaną działać.

10.8 Siatki do wspomagania rysowania

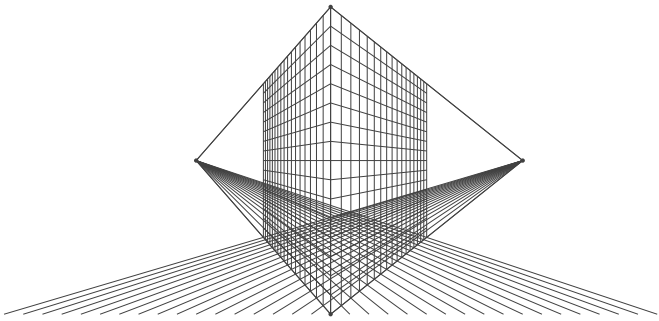
Ipe domyślnie oferuje siatkę prostokątną z różną gęstością oczek zależnie od ustawień opcji `Snap→Grid size`. Jeżeli potrzebujemy innej siatki, to możemy ją przygotować na warstwie o nazwie `GRID`. Warstwę taką możemy dodać za pomocą ipeletu `Ipelets→Grid maker→Create a GRID layer` lub po prostu dodać nową warstwę i zmienić jej nazwę. Jeżeli na warstwie tej umieścimy jakiekolwiek obiekty, to punkty przecięcia linii będą węzłami siatki (punktami magnetycznymi). Na rys. 10.23 pokazano skutek umieszczenia kilku obiektów na warstwie `GRID`. Wszystkie węzły zaznaczono za pomocą znaczników.



Rys. 10.23. Węzły (punkty magnetyczne) w siatce

Przyciąganie do węzłów powstałych na warstwie `GRID` włączamy za pomocą trybu 6 `Snap to custom grid` (zob. rys. 10.9). Jeżeli nie chcemy, aby obiekty z warstwy `GRID` były zaznaczane przy operowaniu myszką, to warstwę należy zablokować.

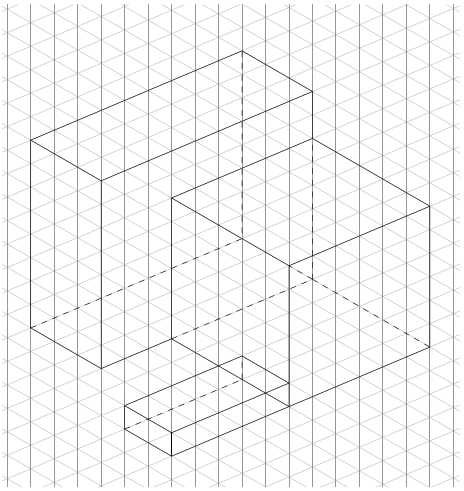
Domyślnie Ipe oferuje dwa ipelety do generowania siatek: `Ipelets→Grid maker→Two-point perspective grid` i `Triangular grid`. Efekt użycia pierwszego z ipeletów pokazano na rys. 10.24. Puntem wyjścia do wygenerowania siatki jest narysowanie czworokąta o pionowej i poziomej przekątnej. Na rysunku wierzchołki użytego czworokąta zaznaczono znacznikami. Drugi ipelet generuje siatkę trójkątną, która jest obrócona o 90 stopni względem siatki pokazanej na rys. 10.25a.



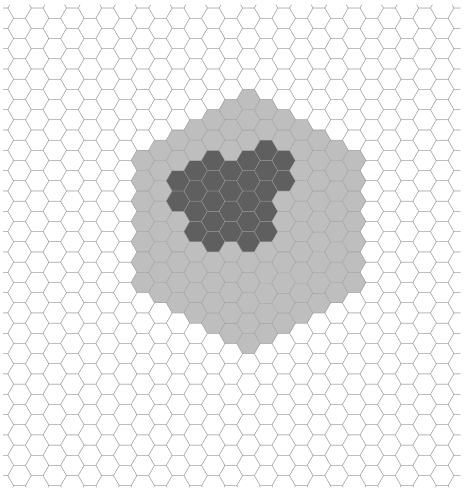
Rys. 10.24. Siatka wygenerowana dla dwupunktowej perspektywy

Na rys. 10.25 pokazano przykłady samodzielnie przygotowanych siatek wraz z przykładowymi rysunkami utworzonymi z ich zastosowaniem. Siatka izometryczna z rys. 10.25a powstała jako zbiór linii pionowych oraz nachylonych pod kątem 30 i 150 stopni. Do równomiernego rozłożenia linii użyto domyślnej siatki prostokątnej. Po narysowaniu pojedynczych linii w każdym kierunku były one kopiowane (również grupowo), aż do pokrycia całej kanwy. Siatka doskonale sprawdza się przy rysowaniu brył w rzucie izometrycznym.

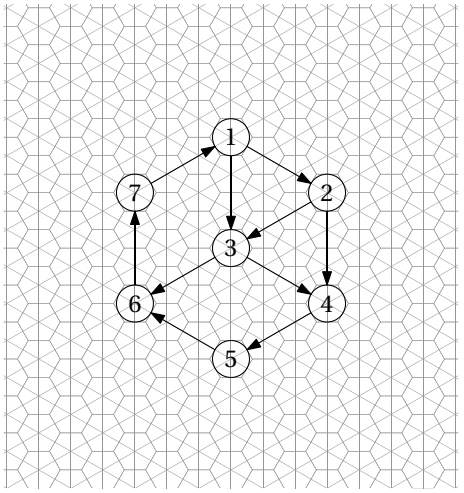
Siatka heksagonalna z rys. 10.25b powstała przez ułożenie obok siebie jednakowych sześciokątów. Pierwszy sześciokąt został wygenerowany za pomocą ipeletu `Ipelets→Goodies→Regular k-gon` (przed wywołaniem polecenia należy zaznaczyć okrąg, w który zostanie wpisany wielokąt). Następnie kopiujemy sześciokąt kilkukrotnie i umieszczamy sześciokąty obok siebie, korzystając z trybu przyciągania 1 (wierzchołki) – chwytamy jeden sześciokąt za wierzchołek i przesuwamy do wierzchołka, z którym ma się stykać. W tej sposób możemy idealnie ułożyć je obok siebie. Przy kilku ułożonych sześciokątach możemy kopiować grupy z czasem coraz większe, co szybko pozwala wypełnić kanwę sześciokątami. Siatki użyto do zobrazowania automatu komórkowego.



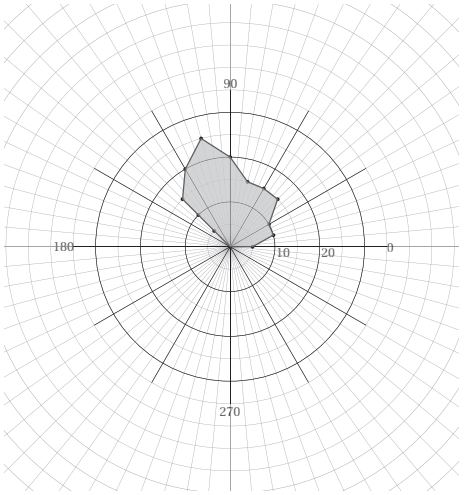
(a) Siatka izometryczna



(b) Siatka heksagonalna



(c) Siatka bazująca na mozaice



(d) Siatka do kreślenia w układzie biegunowym

Rys. 10.25. Wykorzystanie własnych siatek do rysowania

Siatka z rys. 10.25c powstała podobnie jak siatka heksagonalna przez wielokrotne kopiowanie sześciokąta, przy czym pierwszy sześciokąt został podzielony na 6 części. Przy włączonym trybie przyciągania 2 (punkty kontrolne) narysowano trzy odcinki łączące środki przeciwległych boków. Sześciokąt wraz z dodatkowymi odcinkami połączono w grupę, aby ułatwić sobie dalszą pracę. Siatka rewelacyjnie sprawdza się przy ręcznym rozmieszczaniu węzłów grafu

i dodawaniu krawędzi między węzłami. Dodatkowy węzeł siatki, który wypada w centrum wierzchołka grafu (punkt przecięcia trzech odcinków) jest idealny do pozycjonowania etykiet (warto wcześniej w menu `Properties` ustawić domyślne wyrównanie w poziomie i w pionie do środka). Warto tutaj zwrócić uwagę na tryb pracy 6 (zob. rys. 10.2). Jeżeli mamy połączone krawędziami węzły grafu, to tryb ten pozwala przesunąć węzeł bez zepsucia połączeń.

Ostatnia z prezentowanych siatek, pokazana na rys. 10.25d, powstała na potrzeby ręcznego rysowania w biegunowym układzie współrzędnych. Siatka składa się ze współśrodkowych okręgów o promieniach 5, 10, 15,... milimetrów (kreślono je przy domyślnej siatce prostokątnej o gęstości oczek 14 punktów) oraz pęku prostych ustawionych co 5 stopni (obracanie z użyciem ipeletu `Ipelets→Goodies→Precise rotate`).

Przy sprawnym operowaniu myszką każdą z prezentowanych siatek można przygotować w około 5 minut. Na koniec warto zaznaczyć wszystkie obiekty na warstwie `GRID`, zgrupować je, ustawić preferowany kolor i bezwzględną grubość na 0,2 pkt. Przy takich ustawieniach siatka spełnia swoją rolę, jednocześnie nie przysłania nadmiernie rysunku.

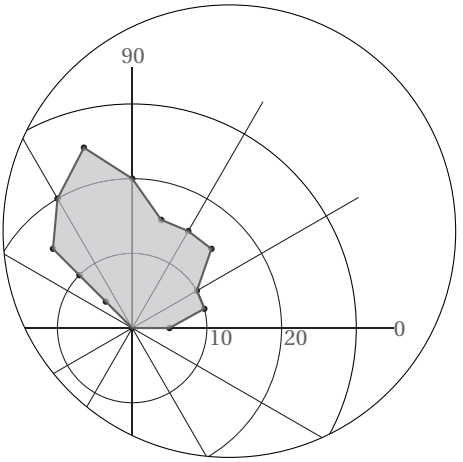
10.9 Przycinanie grafiki do ścieżki

Przy tworzeniu siatki niektóre z jej elementów wystają poza kanwę. Możemy przyciąć je do rozmiaru kanwy. W tym celu rysujemy prostokąt o bokach leżących na krawędzi kanwy, zostawiamy jako zaznaczony, dodajemy jako podstawowe zaznaczenie naszą siatkę (już zgrupowaną w jeden obiekt) i z menu podręcznego wybieramy polecenie `Add clipping path`. Po wyłączeniu zaznaczenia zobaczymy, że siatka została przycięta do rozmiarów prostokąta. W ten sposób możemy przyciąć rysunek do dowolnej ścieżki. Operacja pozostawia te punkty, które są wewnętrznymi punktami ścieżki (najlepiej użyć krzywej zamkniętej). Na rys. 10.26 pokazano wykres z rys. 10.25d przycięty do okręgu (wyłączono również wyświetlanie siatki). Jeżeli przycinamy do ścieżki, to ścieżka nie jest widoczną częścią rysunku. W tym przypadku, aby pozostawić okrąg, skopiowano go i jedna z kopii została użyta do przycięcia, a druga pozostała częścią finalnego rysunku.

Polecenie `File→Insert image` pozwala wstawiać grafikę z zewnętrznych plików jako element naszego rysunku. Obsługiwane są formaty `jpeg` i `png`. Obrazki takie mogą być następnie skalowane, nachylane lub obracane. Możemy również przyciąć obrazek do ścieżki.

Na rys. 10.27a pokazano zdjęcie osadzone jako obiekt w Ipe. Ponieważ polecenie `Add clipping path` stosuje się do zgrupowanych obiektów, to musimy

utworzyć grupę złożoną wyłącznie z naszego zdjęcia. Na rys. 10.27b widzimy zdjęcie przycięte do zamkniętego splajnu.



Rys. 10.26. Wykres w układzie biegunowym przycięty do okręgu

(a) Zdjęcie osadzone w Ipe

(b) Przycięcie do splajnu

Rys. 10.27. Przycinanie zdjęcia do ścieżki

Dodawanie grafiki pozwala dołączyć do niej nowe elementy. Możemy na przykład załadować mapę jakiegoś obszaru, aby dorysować do niej dodatkowe linie, oznaczyć punkty czy nawet dodać wzory matematyczne. Jeżeli chcemy dołączyć grafikę wektorową, to możemy to zrobić poprzez polecenie `\includegraphics{}`, które umieszczamy wewnątrz etykiety tekstowej.

Rozdział 11

Ti*k*Z

Pakiet `tikz` (Ti*k*Z – TikZ ist kein Zeichenprogramm) oferuje bogaty zestaw poleceń o składni nawiązującej do zwykłych poleceń LaTeX-a, które służą do tworzenia grafiki wektorowej. Kod przygotowany z użyciem tych poleceń jest przekształcany do języka PGF (Portable Graphics Format), którego zadaniem jest niskopoziomowe generowanie rysunku. Omawiane w tym rozdziale podejście jest zdecydowane inne niż korzystanie z edytora grafiki. Stajemy tutaj przed problemem „zaprogramowania" rysunku. Rozdział ten stanowi tylko wprowadzenie do możliwości pakietu `tikz`. Sam manual do tego pakietu [46] liczy ponad 1300 stron, a dochodzi do tego jeszcze dokumentacja pakietów współpracujących. Dotkniemy czubka góry lodowej, ale w mojej ocenie warto. Dlaczego? Dostajemy narzędzie do bardzo precyzyjnej kompozycji rysunku (położenie elementów zależy od podanych współrzędnych, a nie dokładności obsługi myszki), pakiet w pełni współpracuje z LaTeX-em, jeżeli konstruujemy podobne rysunki, możemy łatwo wykorzystać kawałki kodu, mamy pełną zgodność użytych czcionek między rysunkami i dokumentem. Dodatkowo na bazie pakietu `tikz` rozwinięto kolejne, które w łatwy sposób pozwalają tworzyć różnego rodzaju diagramy, wykresy itp., co zobaczymy w kolejnych rozdziałach.

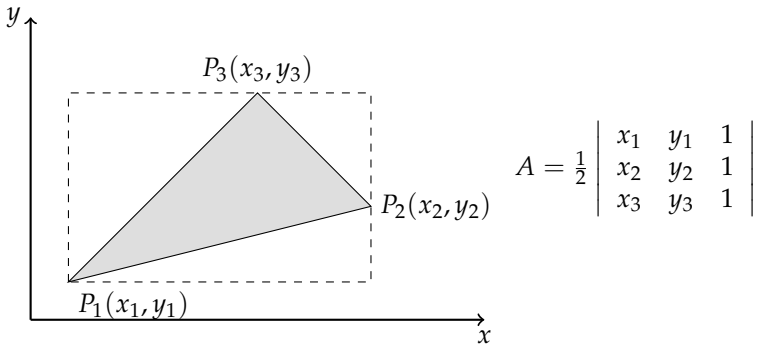
11.1 Pierwszy przykład

Podstawowe elementy, z których składają się rysunki projektowane z użyciem pakietu `tikz`, to węzły i ścieżki. Węzły są małymi fragmentami rysunku, które mają określone swoje położenie, kształt i mogą w sobie coś zawierać, na przykład tekst. Istnieje możliwość nadania węzłom nazw, aby łatwiej się do nich odnosić, jeśli korzystamy z danego węzła w wielu poleceniach. Ścieżki są ciągiem połączonych ze sobą odcinków i/lub fragmentów krzywych.

```
\usepackage{tikz}
```

Listing 11.1. Przykład rysunku przygotowanego z użyciem pakietu `tikz`

```
\begin{tikzpicture}
\draw[thick, ->] (0,0) -- (6,0) node [below] {$x$};
\draw[thick, ->] (0,0) -- (0,4) node [left] {$y$};
\draw[dashed] (0.5,0.5) -- (0.5,3) -- (4.5,3) -- (4.5,0.5) -- cycle;
\draw[fill=gray!20!] (0.5,0.5) -- (3,3) -- (4.5,1.5) -- cycle;
\draw (0.5,0.5) node [below right] {$P_1(x_1,y_1)$};
\draw (4.5,1.5) node [right] {$P_2(x_2,y_2)$};
\draw (3,3) node [above] {$P_3(x_3,y_3)$};
\draw (8,2) node {$A = \frac{1}{2}
  \left|\begin{array}{ccc}
  x_1 & y_1 & 1\\
  x_2 & y_2 & 1\\
  x_3 & y_3 & 1
  \end{array}\right|$};
\end{tikzpicture}
```



Przeanalizujmy pierwszy przykład pokazany na listingu 11.1. Rysunki tworzone z użyciem pakietu `tikz` umieszczamy wewnątrz środowiska **`tikzpicture`**. Podstawą tworzenia rysunków jest polecenie **`\draw`**. W ramach jednego polecenia **`\draw`** mogą występować zarówno segmenty ścieżek, jak i węzły. W składni polecenia obowiązują następujące zasady: współrzędne punktów podajemy w nawiasach okrągłych, opcje rysowania podajemy w nawiasach kwadratowych, a zawartość węzłów w nawiasach klamrowych. Domyślną jednostką przy podawaniu współrzędnych kartezjańskich są centymetry. Punkt `(0,0)` zlokalizowany jest w lewym dolnym rogu (tutaj jest to dokładnie początek układu współrzędnych), o ile nie użyjemy współrzędnych ujemnych. Każde polecenie

tikz musimy zakończyć średnikiem. Może ono obejmować wiele linii i dopiero znak średnika jest sygnałem, że jest ono kompletne.

Pierwsze polecenie **\draw** rysuje ścieżkę złożoną z jednego odcinka, który łączy punkty (0,0) i (6,0) i umieszcza węzeł o treści x poniżej punktu (6,0). Dwa minusy -- oznaczają połączenie sąsiadujących punktów odcinkiem. Opcje thick, -> odnoszą się do rysowanej ścieżki – pogrubiona linia i dodanie grotu na końcu ścieżki. Opcja below dotyczy rysowania węzła i oznacza, że znajdzie się on poniżej punktu (6,0) – jest to bieżący punkt w momencie dodawania węzła. Drugie polecenie jest analogiczne i odpowiada za narysowanie osi y. Polecenie **\draw** w linii 4 rysuje wyłącznie ścieżkę złożoną z czterech odcinków. Opcja cycle oznacza powrót do punktu wyjściowego (łamana zamknięta). Równie dobrze można w tym miejscu powtórzyć współrzędne pierwszego punktu. W nawiasie kwadratowym podana została opcja dashed, co skutkuje rysowaniem linią przerywaną. W efekcie to polecenie **\draw** kreśli widoczny na rysunku prostokąt. Kolejne polecenie rysuje trójkąt, ale ze względu na opcję fill wnętrze zamkniętej ścieżki zostało zamalowane wskazanym kolorem.

Kolejne trzy polecenia (linie 6–8) rysują węzły, które zawierają etykiety opisujące wierzchołki trójkąta. Węzeł domyślnie ma kształt prostokąta, ale ramka wokół węzła nie jest rysowana, o ile tego nie zażądamy. W tym przypadku lokalizacja węzłów wypada dokładnie w wierzchołkach trójkąta, ale zawartość rysowana jest kolejno: poniżej w prawo, na prawo i powyżej wskazanego punktu. Ostatnie polecenie **\draw** (linie 9–14) tworzy węzeł w punkcie (8,2). Punkt (8,2) stanowi w tym przypadku środek węzła. Zawartością węzła jest wzór matematyczny zawierający w sobie macierz. Równie dobrze mogłaby tam być ministrona, tabela złożona w środowisku **tabular**, polecenie **\includegraphics** wstawiające zewnętrzny plik z grafiką lub po prostu akapit tekstu. Na koniec warto dodać, że istotna jest kolejność, w jakiej dodajemy kolejne elementy rysunku. Jeżeli jakieś elementy na siebie nachodzą, to ten dodany później będzie rysowany na górze.

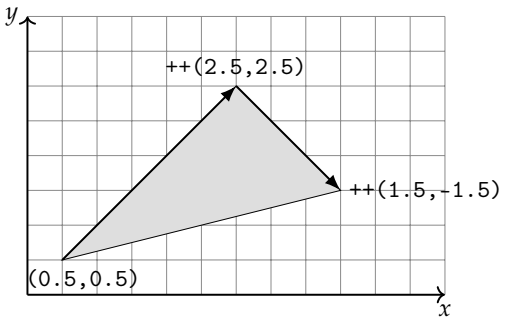
Teraz, gdy już mamy za sobą pierwszy przykład, proponuję, abyś zajrzał na stronę https://texample.net. Znajdziesz tam sporo przykładów przygotowanych z użyciem pakietu tikz z różnych dyscyplin naukowych. Pozwoli Ci to lepiej zorientować się w możliwościach pakietu. Dla przykładów udostępniono również kod źródłowy, więc mogą być inspiracją dla własnych prac.

11.2 Określanie współrzędnych punktów

Jak można było zauważyć w przykładzie z listingu 11.1, jedną z metod określania współrzędnych punktów jest użycie współrzędnych kartezjańskich. Kluczowym elementem jest tutaj przecinek jako separator między współrzędnymi.

W niektórych poleceniach opuszczamy nawiasy okrągłe i to właśnie przecinek wskazuje, że mamy do czynienia z punktami określonymi we współrzędnych kartezjańskich. Pakiet obsługuje również współrzędne kartezjańskie w przestrzeni trójwymiarowej. To zagadnienie omówię w podrozdziale 11.7. Drugą metodą jest użycie współrzędnych biegunowych. W tym przypadku separatorem jest dwukropek, np. `(30:1cm)` oznacza kąt 30° (od osi x, przeciwnie do ruchu wskazówek zegara) i promień 1 cm.

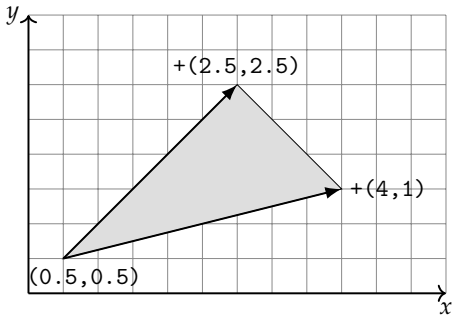
W przykładzie z listingu 11.1 wszystkie punkty były określane bezwzględnie. Możliwe jest definiowanie nowego punktu względem punktu bieżącego (podajemy wektor przesunięcia), a system sam wyznacza nowy punkt. Gdy rysujemy ścieżkę, możemy mówić o punkcie bieżącym, przy czym rozumiemy go dwojako: punkt bieżący dla obliczeń (oznaczmy go sobie O) i bieżące położenie „pisaka” (oznaczmy je sobie przez P). Zazwyczaj te dwa punkty się pokrywają. Jest tak, gdy podajemy bezwzględnie określone punkty. Na przykład, gdy rysujemy trójkąt (linia 5 listing 11.1), zaczynamy od punktu bieżącego $O = P = (0.5, 0.5)$, później mamy $O = P = (3,3)$ itd. Jeżeli chcemy przesunąć punkt bieżący o zadany wektor, to możemy go określić jako `++(..,..)`. Wyznaczamy go względem punktu O. Taki sam trójkąt moglibyśmy uzyskać, określając kolejne wierzchołki trójkąta jako: `(0.5,0.5) -- ++(2.5,2.5) -- ++(1.5,-1.5) -- cycle`. W tym przypadku również punkty O i P cały czas się pokrywają. Metoda ta została zobrazowana na rys. 11.1. Użyta w tle siatka pozwala łatwo sprawdzić poprawność takich przesunięć.



Rys. 11.1. Określanie pozycji względem punktu bieżącego – konstrukcja ++

Jeżeli użyjemy pojedynczego znaku +, to dalej będziemy mieć do czynienia z przesunięciem o wektor względem punktu O, ale ten punkt nie będzie się zmieniał. Wracając do naszego trójkąta, możemy jego wierzchołki zdefiniować jako: `(0.5,0.5) -- +(2.5,2.5) -- +(4,1) -- cycle`. Zaczynamy od $O = P = (0.5, 0.5)$, po narysowaniu pierwszego odcinka mamy $O = (0.5, 0.5)$, $P = (3,3)$, po narysowaniu drugiego odcinka – $O = (0.5, 0.5)$, $P = (4.5, 1.5)$.

Metoda ta została zobrazowana na rys. 11.2. Możemy w ramach jednej ścieżki użyć obu metod przesunięcia o wektor, o ile nie pogubimy się z położeniem punktów *O* i *P*.

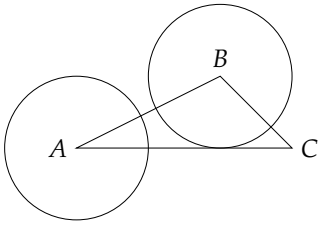


Rys. 11.2. Określanie pozycji względem punktu bieżącego – konstrukcja +

Jeżeli danego punktu używamy wielokrotnie, warto nadać mu nazwę. Przykłady pokazano na listingu 11.2. Nazwę możemy nadać, wykorzystując polecenie `\coordinate` (linie 1 i 2) lub „w locie" stosując opcję `coordinate` (linia 3). Definiując nazwę punktu, możemy przypisać mu etykietę i określić jej położenie (linia 2). Można to zrobić również, definiując nazwę „w locie". Przypisanie etykiety skutkuje natychmiastowym dodaniem jej do rysunku. Etykiety można dodawać również w formie węzłów analogicznie jak w pierwszym przykładzie (linie 6–7).

Listing 11.2. Przykład definiowania nazw dla punktów

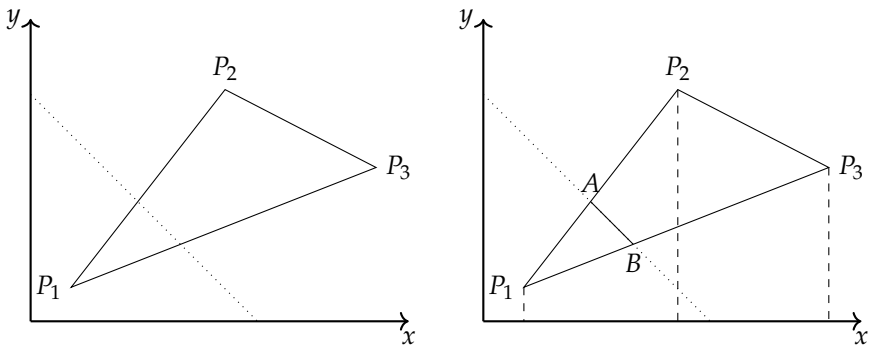
```
1 \begin{tikzpicture}
2 \coordinate (A) at (1,1);
3 \coordinate [label=above:{$B$}] (B) at (3,2);
4 \draw (A) -- (B) -- ++(1,-1) coordinate (C) -- cycle;
5 \draw (A) circle (1);
6 \draw (B) circle (1);
7 \draw (A) node [left] {$A$};
8 \draw (C) node [right] {$C$};
9 \end{tikzpicture}
```



Pakiet oferuje dwie proste metody definiowania współrzędnych jako punktu przecięcia się dwóch prostych. Konstrukcja `(A |- B)` oznacza punkt przecięcia pionowej prostej przechodzącej przez punkt `A` z poziomą prostą przechodzącą przez punkt `B`. W zapisie tym współrzędne punktów podajemy bez nawiasów okrągłych. Drugie wyrażenie ma postać `(intersection of A--B and C--D)` i oznacza punkt przecięcia odcinka `AB` z odcinkiem `CD`. Również w tym przypadku współrzędne punktów podajemy bez nawiasów okrągłych.

Listing 11.3. Wyznaczanie współrzędnych punktów przecięcia prostych

```
1  \begin{tikzpicture}
2  \draw[thick] [->] (0,0) -- (5,0) node [below] {$x$};
3  \draw[thick] [->] (0,0) -- (0,4) node [left] {$y$};
4  \coordinate [label=left:{$P_1$}] (P1) at (40:0.7);
5  \coordinate [label=above:{$P_2$}] (P2) at (50:4);
6  \coordinate [label=right:{$P_3$}] (P3) at (24:5);
7  \draw (P1) -- (P2) -- (P3) -- cycle;
8  \draw[dotted] (0,3) -- (3,0);
9
10 % uzupełnienie rysunku
11 \draw[dashed] (P1) -- (P1 |- 0,0);
12 \draw[dashed] (P2) -- (P2 |- 0,0);
13 \draw[dashed] (P3) -- (P3 |- 0,0);
14
15 \coordinate [label=above:{$A$}] (A)
16   at (intersection of 0,3--3,0 and P1--P2);
17 \coordinate [label=below:{$B$}] (B)
18   at (intersection of 0,3--3,0 and P1--P3);
19 \draw (A) -- (B);
20 \end{tikzpicture}
```



Rozważmy przykład przedstawiony na listingu 11.3. Zacznijmy od lewego rysunku. Powstaje on, jeżeli skorzystamy tylko z kodu z linii 2–8. Chcemy go

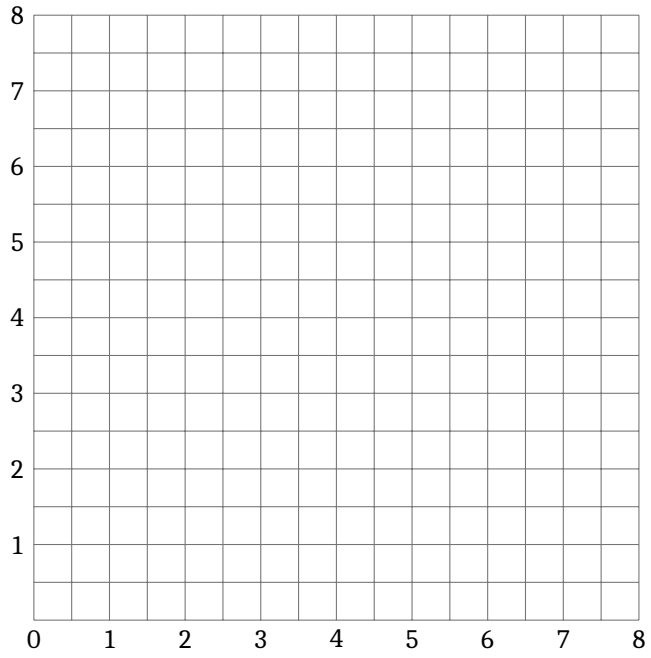
uzupełnić o odcinki wskazujące rzuty wierzchołków trójkąta na oś x oraz narysować ciągłą linią odcinek wykropkowanej prostej, który jest zawarty w trójkącie. Finalny obrazek mamy po prawej stronie. Rysunek przypomina wcześniej analizowany przykład (listing 11.1), ale tym razem punkty `P1`, `P2` i `P3` (zobacz linie 4–6) zostały zdefiniowane za pomocą współrzędnych biegunowych i wyliczenie współrzędnych szukanych rzutów przestaje być zadaniem łatwym. Zadanie jest trywialne, gdy zostawimy je pakietowi `tikz`. W liniach 11–13 mamy trzy polecenia kreślące odcinki łączące wierzchołki trójkąta z ich rzutami na oś x. Współrzędne punktów `A` i `B` zostały wyznaczone w liniach 15–18 przy użyciu drugiej z prezentowanych konstrukcji. Nadaliśmy im nazwy ze względu na etykiety, które zostały dodane do rysunku. W przypadku rzutów na oś x nie było takiej potrzeby. Na koniec rysujemy odcinek łączący punkty `A` i `B`.

Pakiet `tikz` zawiera sporo dodatkowych bibliotek, które można dołączyć do dokumentu. Realizujemy to poleceniem `\usetikzlibrary{}`. Parametrem jest lista nazw bibliotek oddzielonych przecinkami. Polecenie można też powtarzać w preambule, np. dodając za każdym razem jedną bibliotekę. Jedną z nich jest biblioteka `intersections`, która zawiera metody wyznaczania punktów przecięcia między dowolnymi ścieżkami (np. wykresami funkcji). Przykład jej użycia pokażę w kolejnym podrozdziale. Konstrukcje pokazane na listingu 11.3 nie wymagają żadnej dodatkowej biblioteki.

Przygotowując rysunek z użyciem pakietu `tikz`, zdarza się nam pracować metodą prób i błędów, zwłaszcza gdy chcemy idealnie ustawić wszystkie etykiety na rysunku. Możemy ułatwić sobie pracę, zamieniając kanwę, na której rysujemy, w „kartkę w kratkę". Jak to zrobić, pokazano na listingu 11.4. Gęstość naszej kratki (siatki) należy dopasować do własnych potrzeb, choć pokazany krok 5 mm wydaje się bardzo wygodny do pracy. Gdy rysunek będzie gotowy, wystarczy zakomentować trzy linie generujące siatkę z opisem (jest to lepsze rozwiązanie niż usunięcie tych linii – może będziemy kiedyś poprawiać rysunek). Wracając do listingu, polecenie `\draw` (linia 2) rysuje siatkę z krokiem 5 mm, w szarym kolorze, używając cienkiej kreski (jeżeli chcemy linii wykropkowanych, wystarczy dodać opcję `dotted`, jeżeli przerywanych – opcję `dashed`). Rozmiar siatki określamy, podając lewy dolny i prawy górny narożnik. Dwie kolejne linie dodają w pętli liczby na osi x i y. O wykorzystaniu pętli będę jeszcze mówić w podrozdziale 11.6. W tym momencie kluczowy jest efekt w postaci opisanej siatki. Oczywiście rozmiar siatki możemy dopasować do potrzeb. Wystarczy zmienić 4 liczby w podanym kodzie. Również można odpowiednio dobrać kolor. Pokazany przykład opracowano z myślą o druku. Siatkę warto ustawić w kontrastującym kolorze. Jaskrawy czerwony lub różowy kolor może okazać się w większości przypadków najlepszym wyborem. Dodanie siatki pozwala prawidłowo ustawić elementy bardzo szybko, często przy pierwszej próbie.

Listing 11.4. Przygotowanie kanwy do pracy z rysunkiem

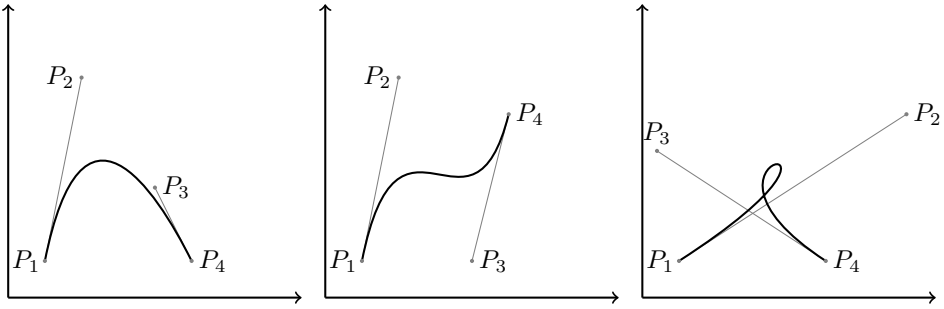
```
\begin{tikzpicture}
\draw[step=5mm,draw=gray!50,very thin] (0,0) grid (8,8);
\foreach \i in {0,...,8} \node[below] at (\i, 0) {\i};
\foreach \i in {1,...,8} \node[left] at (0, \i) {\i};

% ... polecenia
\end{tikzpicture}
```



11.3 Rysowanie ścieżek

Jak wspomniałem przy okazji pierwszego przykładu, ścieżka jest ciągiem połączonych ze sobą odcinków i/lub fragmentów krzywych. Jeżeli chcemy wykreślić prosty odcinek między dwoma punktami, korzystamy z konstrukcji `--`. Bardziej złożona konstrukcja ma postać `(P1) .. controls (P2) and (P3) .. (P4)` i kreśli krzywą Béziera o punkcie początkowym `P1`, punktach kontrolnych `P2` i `P3` oraz punkcie końcowym `P4` (zob. też rys. 10.3). Przykłady krzywych Béziera pokazano na rys. 11.3. Zależnie od ułożenia punktów kontrolnych uzyskujemy inne efekty. Przez punkty `P1` i `P2` przechodzi prosta styczna do krzywej w jej początku. Odległość między tymi punktami decyduje o tym, jak bardzo i jak szybko krzywa odsuwa się od stycznej. Analogicznie druga para punktów

wyznacza styczną w punkcie końcowym. Segmenty krzywej Béziera możemy dowolnie przeplatać z odcinkami lub innymi predefiniowanymi kształtami. Podstawowe kształty przedstawiono w tabeli 11.1. W każdym przypadku rysowanie zaczyna się w bieżącym punkcie. Jeżeli polecenie `\draw` ma narysować wyłącznie pojedynczy kształt, to przed nazwą kształtu musimy podać punkt startowy, np. `\draw (1,1) circle (0.5);`



Rys. 11.3. Krzywe Béziera

Tabela 11.1. Podstawowe kształty dostępne w pakiecie `tikz`

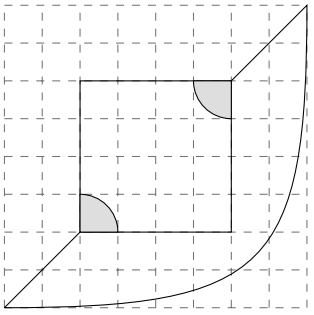
Polecenie	Opis
`rectangle (x,y)`	prostokąt, którego lewy dolny róg znajduje się w punkcie bieżącym, a prawym górnym jest punkt `(x,y)`, który jest nowym punktem bieżącym po zakończeniu rysowania
`grid (x,y)`	j.w., ale wnętrze wypełnione siatką o określonych parametrach
`circle (r)`	okrąg o środku w punkcie bieżącym i promieniu `r`, punkt bieżący nie ulega zmianie
`ellipse (rx and ry)`	elipsa o środku symetrii w punkcie bieżącym i promieniach `xr` i `yr`, punkt bieżący nie ulega zmianie
`arc (a:b:r)`	wycinek okręgu o promieniu `r`, o kącie początkowym `a` i końcowym `b`, rysowanie zaczyna się w punkcie bieżącym, nowym punktem bieżącym jest punkt końcowy łuku
`arc (a:b:rx and ry)`	j.w., ale wycinek elipsy

Na listingu 11.5 pokazano przykłady rysowania ścieżek z różnych segmentów. Dodana siatka powinna ułatwić analizę efektów rysowania poszczególnych

segmentów. We wszystkich punktach bieżących mamy $O = P$. Polecenie `\draw` w linii 3 zaczyna od punktu bieżącego `(0,0)` i rysuje odcinek do punktu `(1,1)`. Ten punkt jest bieżącym, gdy zaczynamy rysowanie prostokąta. Kończymy je w nowym punkcie bieżącym `(3,3)`. Dalej rysujemy odcinek do punktu `(4,4)` i krzywą Béziera do punktu `(0,0)`. Rysowanie w linii 5 zaczynamy od punktu bieżącego `(1,1)`. Następnie przesuwamy się 0.5 cm w prawo i rysujemy łuk o promieniu 0.5 cm w zakresie od 0 do 90 stopni. W efekcie kończymy z nowym punktem bieżącym `(1,1.5)`, skąd wracamy do punktu wyjścia, zamykając ścieżkę, i zamalowujemy jej wnętrze. Analogicznie powstaje drugi zaznaczony kąt.

Listing 11.5. Przykład rysowania ścieżek z różnych segmentów

```
1 \begin{tikzpicture}
2 \draw[step=5mm,draw=gray,dashed] (0,0) grid (4,4);
3 \draw (0,0) -- (1,1) rectangle (3,3) -- (4,4) ..
4   controls (4,0.5) and (3.5,0) .. (0,0);
5 \draw[fill=gray!20!] (1,1) -- ++(0.5,0) arc (0:90:0.5) -- cycle;
6 \draw[fill=gray!20!] (3,3) -- ++(-0.5,0) arc (180:270:0.5) -- cycle;
7 \end{tikzpicture}
```



O sposobie rysowania ścieżki decydują parametry, które umieścimy w nawiasie kwadratowym. Najważniejsze parametry, których możemy użyć, przedstawiono w tabeli 11.2.

Rysowanie ścieżki może rozpoczynać się i/lub kończyć grotem. Przegląd wybranych kształtów grotów pokazano w tabeli 11.3. Groty można łączyć ze sobą, np. `\draw[{Circle}{Turned Square}-{Triangle}{Square}] (0,0) -- (1,0);` wygeneruje ●◆⟶■. Ponadto pakiet oferuje możliwość ustawiania parametrów grotu. Najciekawsze opcje przedstawiono w tabeli 11.4. Więcej informacji można znaleźć w dokumentacji pakietu.

```
\usetikzlibrary{arrows.meta}
```

Tabela 11.2. Wybrane parametry rysowania ścieżek

Opcja	Opis
`color=c`	określenie koloru rysowania
`fill=c`	określenie koloru zamalowania wnętrza
`ultra thin`, `very thin`, `thin`, `semi thick`, `very thick`, `ultra thick`	określenie grubości linii
`line width=x`	określenie grubości linii
`loosely dotted`, `dotted`, `densely dotted`, `loosely dashed`, `dashed`, `densely dashed`, `loosely dash dot`, `dash dot`, `densely dash dot`, `loosely dash dot dot`, `dash dot dot`, `densely dash dot dot`	sposób rysowania linii
`rounded corners=x`	zaokrąglenie połączeń segmentów ścieżki (definiujemy promień)
`line cap=x`	sposób rysowania zakończeń segmentów (`round`, `rect`, `butt`)

Tabela 11.3. Wybrane kształty grotów – wymagana biblioteka `arrows.meta`

Opcja	Kształt	Opcja	Kształt
`->`	⟶	`->>`	⟶
`-Circle`	—●	`-{Circle[open]}`	—○
`-Diamond`	—◆	`-{Diamond[open]}`	—◇
`-Ellipse`	—⬬	`-{Ellipse[open]}`	—⬭
`-Kite`	—◆	`-{Kite[open]}`	—◇
`-LaTeX`	—▶	`-{LaTeX[open]}`	—▷
`-{LaTeX[round]}`	—▶	`-{LaTeX[round,open]}`	—▷
`-Rectangle`	—▬	`-{Rectangle[open]}`	—▭
`-Square`	—■	`-{Square[open]}`	—□
`-Stealth`	—➤	`-{Stealth[open]}`	—➤
`-{Stealth[round]}`	—➤	`-{Stealth[round,open]}`	—➤
`-Triangle`	—▶	`-{Triangle[open]}`	—▷
`-Turned Square`	—◆	`-{Turned Square[open]}`	—◇

Jeżeli to ma sens, to oczywiście opcje możemy łączyć ze sobą. Na przykład w ostatnich dwóch przykładach w tabeli 11.4 przeskalowano groty, aby uwydatnić efekt dotyczący zmiany koloru.

Tabela 11.4. Wybrane opcje modyfikacji wyglądu grotów

Polecenie	Efekt
`\draw[-Stealth] (0,0) -- (1,0);`	
`\draw[-{Stealth[scale=2]}] (0,0) -- (1,0);`	
`\draw[-{Stealth[width=6pt]}] (0,0) -- (1,0);`	
`\draw[-{Stealth[length=8pt]}] (0,0) -- (1,0);`	
`\draw[-{Stealth[reversed]}] (0,0) -- (1,0);`	
`\draw[-{Stealth[harpoon]}] (0,0) -- (1,0);`	
`\draw[-{Stealth[color=gray!20,scale=2]}] (0,0) -- (1,0);`	
`\draw[-{Stealth[fill=gray!20,scale=2]}] (0,0) -- (1,0);`	

`\usetikzlibrary{intersections}`

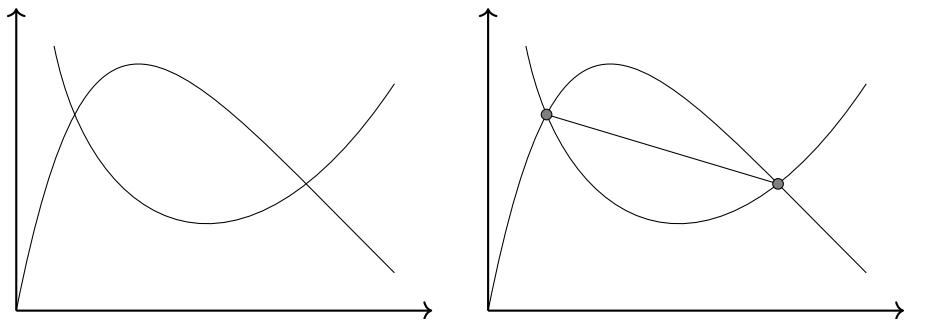
Jeżeli definiujemy ścieżkę, możemy przypisać jej nazwę. Jest to niezbędne, gdy chcemy wyznaczyć punkty przecięcia dwóch ścieżek i nie są to odcinki. Rozważmy przykład pokazany na listingu 11.6. Lewy rysunek obrazuje nasz problem. Mamy dwie krzywe Béziera, które przecinają się w dwóch punktach. Chcemy zaznaczyć graficznie punkty przecięcia i połączyć je odcinkiem. Oczekiwany efekt pokazano na prawym rysunku.

Zaczynamy od dołączenia biblioteki `intersections` w preambule oraz nazwania ścieżek – opcja `name path` (linie 4–5). Pierwszą ścieżkę nazwaliśmy `l1`, a drugą `l2`. Zaczniemy od narysowania odcinka, aby szare koła znalazły się nad nim. Gdy pierwszy raz sięgamy po wyznaczone punkty przecięcia, zostają one nazwane. Opcja polecenia `\draw` w linii 6 `name intersections={of=l1 and l2, name=i}` oznacza, że sięgamy po punkty przecięcia ścieżek `l1` i `l2` oraz że przedrostek nazwy dla tych punktów to `i`. Punkty te dostaną nazwy `i-1`, `i-2` itd. (u nas są tylko 2). Dalej już tylko korzystamy z tych nazw, aby narysować odcinek, a w kolejnych liniach szare koła o promieniu 2pt w tych punktach. Na koniec warto podkreślić, że w przypadku bardzo złożonych ścieżek, które zawierają bardzo małe segmenty, dokładność wyznaczenia punktów przecięcia może nie być zadowalająca.

Listing 11.6. Wyznaczanie współrzędnych punktów przecięcia dwóch ścieżek

```
1 \begin{tikzpicture}
2 \draw[thick,->] (0,0) -- (5.5,0);
3 \draw[thick,->] (0,0) -- (0,4);
4 \draw[name path=l1] (0,0) .. controls (1,5) and (2,3.5) .. (5,0.5);
5 \draw[name path=l2] (0.5,3.5) .. controls (1,1) and (3,0) .. (5,3);
```

```
\draw[name intersections={of=l1 and l2, name=i}] (i-1) -- (i-2);
\draw[fill=gray] (i-1) circle (2pt);
\draw[fill=gray] (i-2) circle (2pt);
\end{tikzpicture}
```



Jeżeli znamy liczbę punktów przecięcia, to zamiast opcji `name` możemy użyć `by`. Ta opcja przyjmuje jako wartość listę nazw dla punktów przecięcia, np. `by={A,B}`. Ponadto możemy zdefiniować ścieżkę, której nie rysujemy. Używamy do tego polecenia **\path**, np. `\path[name path=l3] (0,0) - (3,3);`. Takiej ścieżki możemy użyć do wyznaczania punktów przecięcia z inną ścieżką.

11.4 Węzły

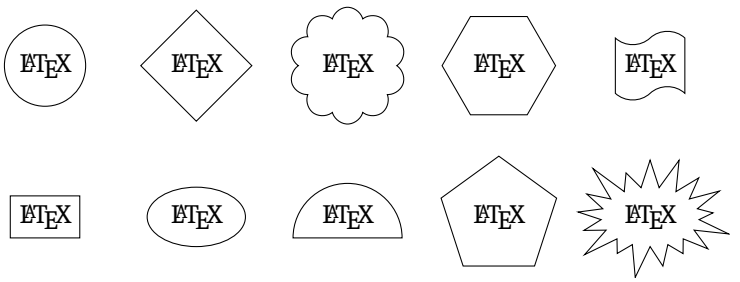
Jak już zostało to powiedziane przy opisie pierwszego przykładu, węzły są małymi fragmentami rysunku, które mają określone swoje położenie, kształt i mogą zawierać tekst, tabele, wzory matematyczne itd. Rozmiar węzła jest określany na podstawie jego zawartości. Domyślnym kształtem jest prostokąt, ale musimy dodać opcję `draw`, aby zostało narysowane obramowanie węzła. Domyślnie wyświetlana jest tylko jego zawartość. Jeżeli potrzebujemy bardziej złożonych kształtów węzłów niż prostokąt lub okrąg, to należy dołączyć w preambule bibliotekę `shapes`. Przykłady różnych kształtów pokazano na listingu 11.7.

```
\usetikzlibrary{shapes}
```

Listing 11.7. Przykładowe kształty węzłów – biblioteka `shapes`

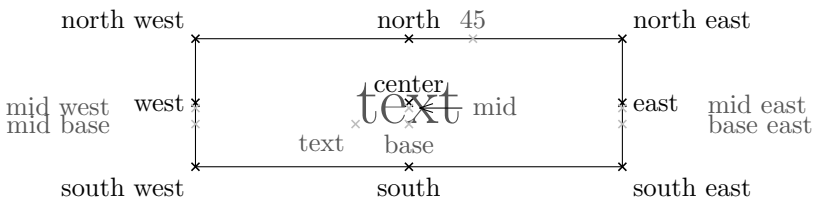
```
\begin{tikzpicture}
\draw (0,0) node [draw] {\LaTeX};
\draw (0,2) node [draw, circle] {\LaTeX};
\draw (2,0) node [draw, ellipse] {\LaTeX};
```

```
\draw (2,2) node [draw, diamond] {\LaTeX};
\draw (4,0) node [draw, semicircle] {\LaTeX};
\draw (4,2) node [draw, cloud] {\LaTeX};
\draw (6,0) node [draw, regular polygon] {\LaTeX};
\draw (6,2) node [draw, regular polygon, regular polygon sides=6]
  {\LaTeX};
\draw (8,0) node [draw, starburst] {\LaTeX};
\draw (8,2) node [draw, tape] {\LaTeX};
\end{tikzpicture}
```



Jeżeli polecenie w środowisku **tikzpicture** ma za zadanie wyłącznie utworzyć węzeł, to np. zamiast instrukcji w pierwszej linii pakiet `tikz` oferuje równoważne polecenie `\node[draw] at (0,0) {\LaTeX};`. Jeżeli dodatkowo chcemy temu węzłowi nadać nazwę, to przyjmie ono formę `\node[draw] (X) at (0,0) {\LaTeX};`.

W tabeli 11.5 przedstawiono podstawowe właściwości węzłów, których można użyć do ustalenia pożądanego wyglądu. Ponadto kluczowa jest pozycja umieszczenia węzła. Jeżeli nie podamy żadnej opcji, to wskazane współrzędne będą środkiem węzła. Jak pokazano już w pierwszym przykładzie (listing 11.1), można dodać opcję, precyzującą, gdzie względem tych współrzędnych ma być ulokowany węzeł. Możemy do tego użyć mechanizmu kotwic, tj. wstawiając węzeł, możemy wskazać, która kotwica ma się znaleźć w punkcie rysowania węzła (domyślnie jest to `center`). Kotwice używają nazw geograficznych. Ich rozmieszczenie dla węzła w kształcie prostokąta pokazano na rys. 11.4.



Rys. 11.4. Węzeł w kształcie prostokąta z zaznaczonymi kotwicami

Tabela 11.5. Podstawowe właściwości węzłów

Opcja	Opis
`shape=x`	kształt węzła (można pominąć `shape=`)
`draw`	rysowanie kształtu węzła
`inner sep=x`	ustawienia wewnętrznego marginesu między zawartością węzła a jego krawędzią
`inner xsep=x`	j.w., ale dotyczy kierunku osi x
`inner ysep=x`	j.w., ale dotyczy kierunku osi y
`outer sep=x`	ustawienia zewnętrznego marginesu między krawędzią węzła a lokalizacją kotwic (domyślnie 0,5 grubości linii)
`outer xsep=x`	j.w., ale dotyczy kierunku osi x
`outer ysep=x`	j.w., ale dotyczy kierunku osi y
`minimum height`	minimalna wysokość węzła
`minimum width`	minimalna szerokość węzła
`minimum size`	minimalny rozmiar (wysokość i szerokość)
`shape aspect=x`	stosunek szerokości do wysokości węzła
`color=x`	kolor węzła
`text`	kolor tekstu (jeśli ma być inny niż ustawienie opcji `color`)
`node font=x`	czcionka dla zawartości węzła i dodatkowych etykiet
`font=x`	czcionka dla zawartości węzła
`align=x`	sposób wyrównania tekstu w węźle: `left`, `right`, `justify`,...
`text width=x`	szerokość tekstu w węźle (możliwe łamanie wierszy)

Na rys. 11.4 widać, że poza kotwicami oznaczonymi kierunkami geograficznymi mamy kilka dodatkowych kotwic związanych z umieszczeniem tekstu w węźle (oznaczone szarym kolorem). Położenie `base` związane jest z położeniem linii bazowej tekstu. Na tej samej wysokości jest kotwica `text`, ale położona na lewo od tekstu. Położenie `mid` związane jest z wysokością w połowie małej litery „x”. Ostatnia kotwica oznaczona jako `45` jest położona pod kątem 45 stopni względem centrum. W dokumentacji [46] możemy sprawdzić rozmieszczenie kotwic dla każdego dostępnego kształtu. W przypadku niektórych nietypowych kształtów nie jest to takie oczywiste. Dla okręgu, który jest oprócz prostokąta najczęściej używany, rozkład jest analogiczny, ale kotwice znajdują się na okręgu.

Listing 11.8. Umieszczanie węzłów z użyciem opcji `anchor`

```
1 \begin{tikzpicture}
2 \node[draw, color=gray, anchor=center] at (0,0) {\LaTeX};
3 \node[draw, color=gray, anchor=south] at (2,0) {\LaTeX};
```

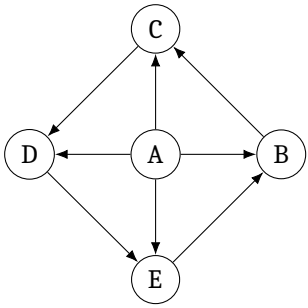
```
\node[draw, color=gray, anchor=north] at (4,0) {\LaTeX};
\node[draw, color=gray, anchor=south east] at (6,0) {\LaTeX};
\node[draw, color=gray, anchor=east] at (8,0) {\LaTeX};
\node[draw, color=gray, anchor=base] at (10,0) {\LaTeX};
\fill (0,0) circle (2pt);
\fill (2,0) circle (2pt);
\fill (4,0) circle (2pt);
\fill (6,0) circle (2pt);
\fill (8,0) circle (2pt);
\fill (10,0) circle (2pt);
\end{tikzpicture}
```

Na listingu 11.8 pokazano przykłady umieszczania węzłów z użyciem wskazanej kotwicy. Robimy to poprzez opcję `anchor`. Opcję `anchor=center` można pominąć, bo jest to wersja domyślna. Zamiast `anchor=south` można użyć `above`, zamiast `anchor=north` – `below`, zamiast `anchor=south east` – `above left` itd. Na rysunku dodatkowo zaznaczone zostały punkty, w których zakotwiczyliśmy węzły. Polecenie `\fill` nie rysuje krawędzi, ale zamalowuje wnętrze. Jest równoważne poleceniu `\draw` przy ustawionej opcji `draw=none`, co interpretujemy jako niewidoczny kolor krawędzi.

```
\usetikzlibrary{positioning}
```

Listing 11.9. Umieszczanie węzłów względem siebie – biblioteka `positioning`

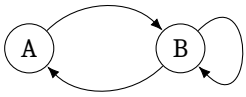
```
\begin{tikzpicture}
\node[draw,circle,] (A) at (0,0) {A};
\node[draw,circle,right=1cm of A] (B) {B};
\node[draw,circle,above=1cm of A] (C) {C};
\node[draw,circle,left=1cm of A] (D) {D};
\node[draw,circle,below=1cm of A] (E) {E};
\draw[-LaTeX] (A) -- (B);
\draw[-LaTeX] (A) -- (C);
\draw[-LaTeX] (A) -- (D);
\draw[-LaTeX] (A) -- (E);
\draw[-LaTeX] (B) -- (C);
\draw[-LaTeX] (C) -- (D);
\draw[-LaTeX] (D) -- (E);
\draw[-LaTeX] (E) -- (B);
\end{tikzpicture}
```



Jeżeli dołączymy bibliotekę `positioning`, to opcje `above`, `below`, `left` itd. dostępne są w wersji z przesunięciem. Na przykład opcja `above=1mm` oznacza nie tylko umieszczenie nad (kotwica `south`), ale dodatkowe przesunięcie o 1 mm w górę. W przypadku opcji `below=4pt` mamy dodatkowe przesunięcie w dół itd. Dodatkowo łatwo można ustawiać węzły względem siebie, co pokazano na listingu 11.9. Po zdefiniowaniu węzła `A` kolejny umieszczamy o 1 cm na prawo od `A` (opcja `right=1cm of A`). Warto zwrócić uwagę, że wtedy nie mamy już `at` i punktu położenia węzła. Podobnie rozmieszczane są wokół węzła `A` pozostałe węzły (linie 4–6). W liniach 7–14 mamy zdefiniowane krawędzie łączące węzły. W efekcie otrzymaliśmy graf skierowany. Widać, że pakiet automatycznie dobiera miejsca, gdzie mają zaczynać się i kończyć krawędzie. Gdyby była taka potrzeba, możemy wskazać właściwe kotwice.

Listing 11.10. Wykorzystanie kotwic do definiowania punktów zaczepienia krawędzi

```
1 \begin{tikzpicture}
2 \node[draw,circle] (A) at (0,0) {A};
3 \node[draw,circle] (B) at (2,0) {B};
4 \draw[-LaTeX] (A.north east) to[out=45,in=135] (B.north west);
5 \draw[-LaTeX] (B.south west) to[out=225,in=-45] (A.south east);
6 \draw[-LaTeX] (B.north east) .. controls (3,1) and (3,-1) ..
7   (B.south east);
8 \end{tikzpicture}
```

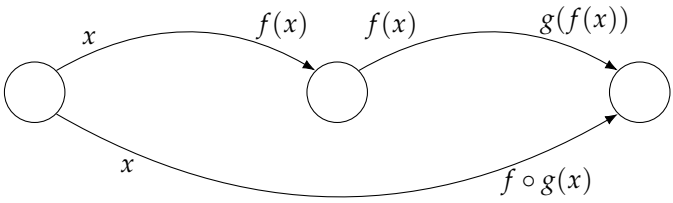


Przykład samodzielnego doboru kotwic, w których chcemy zaczepić krawędzie, pokazano na listingu 11.10. Do kotwic odwołujemy się, podając nazwę węzła, po której występuje kropka i nazwa kotwicy. W rozważanym przykładzie dwie pierwsze krawędzie (linie 4–5) narysowano z użyciem konstrukcji `to`. Parametrami są tutaj kąt, pod jakim krawędź wychodzi z węzła źródłowego, i kąt,

pod jakim dociera do węzła końcowego. Ostatnia krawędź (linie 6–7) została narysowana z użyciem krzywej Béziera.

Listing 11.11. Dołączanie węzłów (etykiet) wzdłuż krawędzi

```
1  \begin{tikzpicture}
2  \node[draw,circle,minimum size=8mm] (A) at (0,0) {};
3  \node[draw,circle,minimum size=8mm] (B) at (4,0) {};
4  \node[draw,circle,minimum size=8mm] (C) at (8,0) {};
5
6  \draw[-LaTeX] (A.north east) to[out=30,in=150]
7    node [very near start, above] {$x$}
8    node [very near end, above=2pt] {$f(x)$} (B.north west);
9  \draw[-LaTeX] (B.north east) to[out=30,in=150]
10   node [very near start, above=2pt] {$f(x)$}
11   node [very near end, above=4pt] {$g(f(x))$} (C.north west);
12 \draw[-LaTeX] (A.south east) to[out=330,in=210]
13   node [very near start, below] {$x$}
14   node [very near end, below=3pt] {$f\circ g(x)$} (C.south west);
15 \end{tikzpicture}
```



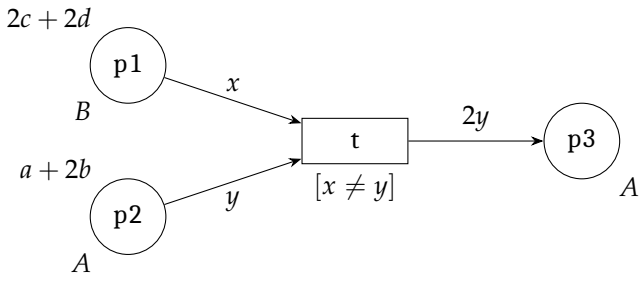
W wygodny sposób możemy dołączać dodatkowe węzły wzdłuż rysowanej ścieżki. Najczęściej pełnią one rolę etykiet. Na listingu 11.11 mamy prosty diagram ilustrujący ideę składania funkcji. Przy rysowaniu węzłów (linie 2–4) zastosowano opcję `minimum size=8mm`. Pozwoliło to uzyskać większe węzły, niż byłyby domyślnie (węzły w środku są puste, więc o ich rozmiarze decydowałaby opcja `inner sep`). Trzy krawędzie zostały narysowane z użyciem konstrukcji zaprezentowanej w poprzednim przykładzie, ale po `to[...]` mamy dodane dwa węzły. Przy pierwszej krawędzi (linie 6–8) dodajemy węzeł z treścią x bardzo blisko początku krawędzi oraz węzeł z etykietą $f(x)$ bardzo blisko końca. Oba węzły są umieszczone nad krawędzią, dodatkowo w drugim przypadku dodano małe przesunięcie, aby litera f nie dotykała krawędzi. Analogicznie zdefiniowano węzły/etykiety dla pozostałych krawędzi. Zamiast określania pozycji za pomocą predefiniowanych lokalizacji możemy użyć opcji `pos=x`, gdzie `x` wskazuje pozycję w drodze od początku do końca krawędzi. Predefiniowane pozycje przedstawiono w tabeli 11.6.

Tabela 11.6. Predefiniowane pozycje umieszczania węzłów na ścieżce

Opcja	Równoważne określenie
`at start`	`pos=0`
`very near start`	`pos=0.125`
`near start`	`pos=0.25`
`midway`	`pos=0.5`
`near end`	`pos=0.75`
`very near end`	`pos=0.875`
`at end`	`pos=1`

Listing 11.12. Dołączanie węzłów (etykiet) do węzłów

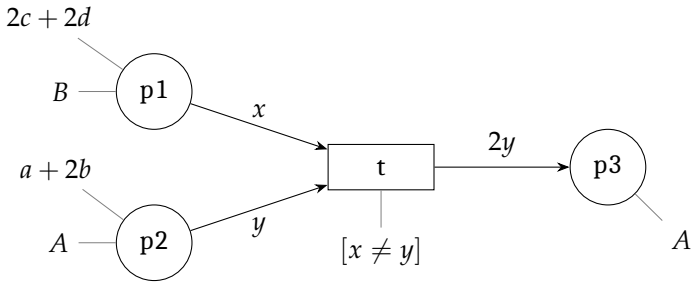
```
1  \begin{tikzpicture}
2  \node[draw,circle,minimum size=1cm,
3    label=above left:$a+2b$, label=below left:$A$] (p2) at (0,0) {p2};
4  \node[draw,circle,minimum size=1cm,
5    label=above left:$2c+2d$, label=below left:$B$] (p1) at (0,2) {p1};
6  \node[draw,circle,minimum size=1cm,
7    label=below right:$A$] (p3) at (6,1) {p3};
8  \node[draw,minimum width=1.4cm,minimum height=6mm,
9    label={below:$[x\ne y]$}] (t) at (3,1) {t};
10
11 \draw[-Stealth] (p1) -- node[above] {$x$} (t);
12 \draw[-Stealth] (p2) -- node[below] {$y$} (t);
13 \draw[-Stealth] (t) -- node[above] {$2y$} (p3);
14 \end{tikzpicture}
```



Możemy również dodawać węzły do innych węzłów. Również w tym przypadku pełnią one najczęściej rolę dodatkowych etykiet. Rozpatrzmy przykład z listingu 11.12. W liniach 2–3 rysujemy węzeł `p2` położony w punkcie `(0,0)`. Na liście opcji poza ustawieniami parametrów rysowania mamy definicje dwóch etykiet. Pierwsza z nich, `label=above left:$a+2b$`, definiuje etykietę

ulokowaną na lewo i w górę od naszego węzła, kolejna, `label=below left:$A$`, na prawo w dół. Jeżeli trzeba, możemy zdefiniować więcej etykiet.

Etykietę definiujemy według składni `label=położenie:tekst`. Jeżeli zachodzi konieczność ustawienia opcji rysowania węzła/etykiety, to stosujemy składnię z nawiasem klamrowym: `label={[opcje]położenie:tekst}`. Warto zwrócić uwagę, że nawiasów klamrowych użyto w linii 9, mimo że nie definiujemy tam żadnych dodatkowych opcji rysowania etykiety. W tym przypadku LaTeX miał problem z interpretacją nawiasów użytych wewnątrz wzoru matematycznego. Użycie klamer wyeliminowało błąd kompilacji. Jeżeli zamiast opcji `label` użyjemy `pin` (pozostała część składni taka jak poprzednio), to nasz dodatkowy węzeł/etykieta będzie połączony z oryginalnym węzłem za pomocą linii. Efekt taki pokazano na rysunku 11.5.



Rys. 11.5. Wykorzystanie opcji `pin` zamiast `label`

11.5 Rysowanie ze stylem

Jeżeli konstruujemy rysunek, dość często zdarza się, że dla wielu węzłów musimy ustawić te same opcje rysowania. Dodatkowo ich długa lista powoduje, że nasz kod jest coraz mniej czytelny. Pakiet `tikz` oferuje ciekawe mechanizmy, które pozwalają łatwo rysować w takim samym stylu podobne elementy rysunku. Wróćmy do rysunku z listingu 11.9. W tym przypadku wszystkie węzły używają opcji `draw` i `circle` oraz wszystkie krawędzie są rysowane z grotem `LaTeX`. Możemy ustawić te opcje na poziomie środowiska **`tikzpicture`** i uzyskać bardziej czytelny kod jak na listingu 11.13, który generuje dokładnie taki sam rysunek jak poprzednio.

Listing 11.13. Definiowanie wspólnych parametrów węzłów w opcjach środowiska

```
1 \begin{tikzpicture}[>=LaTeX, every node/.style={draw,circle}]
2 \node (A) at (0,0) {A};
3 \node[right=1cm of A] (B) {B};
```

```
\node[above=1cm of A] (C) {C};
\node[left=1cm of A]  (D) {D};
\node[below=1cm of A] (E) {E};
\draw[->] (A) edge (B);
\draw[->] (A) edge (C);
\draw[->] (A) edge (D);
\draw[->] (A) edge (E);
\draw[->] (B) edge (C);
\draw[->] (C) edge (D);
\draw[->] (D) edge (E);
\draw[->] (E) edge (B);
\end{tikzpicture}
```

W pierwszej linii ustawiamy domyślny kształt grotu i wspólne opcje dla wszystkich węzłów. Dzięki temu nie musimy ich później powtarzać. Co ważne, jeżeli dla danego węzła chcemy nadpisać domyślne ustawienia, to wystarczy podać odpowiednią opcję. Innymi słowy, jeżeli potrzebowalibyśmy prostokątnego węzła, to wystarczy dodać `rectangle` na liście jego opcji. W kodzie dokonano jeszcze jednej zmiany. Zamiast `--` wskazujących na rysowanie krawędzi między węzłami, użyto opcji `edge` (można jej użyć również zamiast `to`). Po tej zmianie można dodać w opcjach środowiska parametry rysowania krawędzi, np. `every edge/.style={draw,dashed}`. Po takiej zmianie wszystkie krawędzie będą rysowane linią przerywaną.

Przejdźmy teraz do przykładu z listingu 11.12. Gdybyśmy chcieli, aby wszystkie etykiety dodane za pomocą opcji `label` były rysowane pomniejszoną czcionką i w kolorze czerwonym, moglibyśmy dodać w opcjach środowiska **tikzpicture**: `every label/.style={font=\small,red}`.

Bardziej elastycznym rozwiązaniem jest definiowanie własnych stylów. Możemy mieć na przykład trzy style rysowania węzłów i w danym miejscu wskazujemy, którego stylu chcemy użyć. Dodatkowo style można definiować globalnie w preambule dokumentu. Podejście takie pozwala łatwo zmodyfikować, jeśli zajdzie taka potrzeba, wszystkie rysunki przygotowane z naszymi stylami.

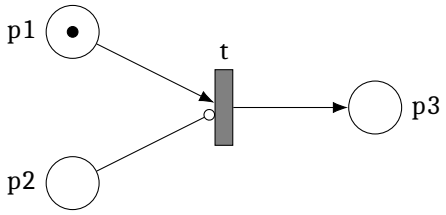
Styl to nic innego jak nazwany zbiór opcji, np. `vertex/.style={draw, circle, minimum size=8mm}`. Taka definicja może pojawić się w opcjach środowiska **tikzpicture** lub jako parametr polecenia `\tikzset{}` (lepsze rozwiązanie). Jeżeli polecenie **\tikzset** umieścimy poza środowiskiem **tikzpicture**, to styl taki będzie mógł być używany we wszystkich rysunkach ulokowanych w kodzie poniżej (zawsze możemy nadpisać jakąś opcję w pojedynczych przypadkach). Jeżeli chcemy mieć dostęp do stylu w całym dokumencie, należy umieścić polecenie **\tikzset** w preambule. Przykład zdefiniowanych własnych stylów pokazano na listingu 11.14, a ich wykorzystanie na listingu 11.15.

Listing 11.14. Definiowanie własnych stylów

```
1 \tikzset{place/.style={draw,circle,minimum size=7mm},
2 transition/.style={draw,rectangle,minimum height=10mm,fill=gray},
3 token/.style={inner sep=0.5mm,draw,circle,fill},
4 arc/.style={draw,-{LaTeX[scale=1.2]}},
5 inhibitor/.style={draw,-{Circle[open,scale=1.2]}}}
```

Listing 11.15. Wykorzystanie własnych stylów do konstrukcji grafu

```
1  \begin{tikzpicture}
2  \node[place, label=left:p1] (p1) at (0,2) {};
3  \node[place, label=left:p2] (p2) at (0,0) {};
4  \node[transition, label=above:t] (t) at (2,1) {};
5  \node[place, label=right:p3] (p3) at (4,1) {};
6  \node[token] at (p1) {};
7
8  \draw (p1) edge[arc] (t)
9        (p2) edge[inhibitor] (t)
10       (t)  edge[arc] (p3);
11 \end{tikzpicture}
```



Instrukcja `\tikzset` w listingu 11.14 zawiera definicje trzech stylów do rysowania węzłów i dwóch do rysowania krawędzi. Graf pokazany na listingu 11.15 jest tzw. siecią Petriego z łukami wzbraniającymi. Z naszej perspektywy ważne jest, że jest to graf z dwoma rodzajami węzłów (okręgi i prostokąty) i dwoma rodzajami krawędzi (różnią się grotem). Dodatkowo węzły w postaci okręgów mogą zawierać znaczniki rysowane jako czarne zamalowane kółka. W prezentowanym podejściu jest to trzeci typ węzłów. Ze stylu korzystamy, podając jego nazwę na liście opcji. W linii 2 tworzymy węzeł w stylu `place`, dodatkowo dodajemy etykietę z nazwą węzła umiejscowioną na lewo od okręgu. Kolejne trzy węzły tworzymy podobnie, przy czym w linii 4 jest to węzeł w stylu `transition`. Ostatni węzeł w stylu `token` rysowany jest w linii 6. Jest umieszczony nad węzłem `p1`. Wszystkie krawędzie tworzone są w jednym poleceniu `\draw` z użyciem konstrukcji `edge`. W tym przypadku nazwa stylu jest jedyną opcją rysowania każdej z krawędzi. Podsumowując ten przykład, na listingu 11.15 mamy wyłącznie

instrukcje konstruujące strukturę rysunku. Wszystkie kwestie związane z formatowaniem zostały przesunięte do definicji stylów.

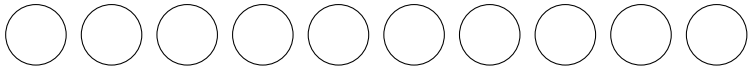
Style można parametryzować. Jeżeli potrzebowalibyśmy rysować węzły w stylu `place` w różnych kolorach, to definicja tego stylu mogłaby mieć postać `place/.style={draw, circle,minimum size=7mm,color=#1}`. `#1` oznacza tutaj pierwszy parametr stylu. Takiego stylu używamy później w postaci np. `place=red`.

11.6 Rysowanie w pętli

We wstępie powiedziałem, że przygotowanie rysunku w `tikz` to jak jego „zaprogramowanie". Nic dziwnego, że pakiet oferuje możliwość powtarzania w pętli wskazanych instrukcji. Rozpocznijmy od prostego przykładu zaczerpniętego z dokumentacji [46], a pokazanego na listingu 11.16. Po nazwie instrukcji pętli (`\foreach`) podajemy nazwę zmiennej sterującej pętlą, określamy listę wartości, które ta zmienna będzie przyjmować, i podajemy instrukcje wykonywane w ramach pętli. Jeżeli jest więcej instrukcji niż jedna, to umieszczamy je w nawiasach klamrowych. W naszym przykładzie zmienna sterująca to `\i` (nazwa zmiennej musi zaczynać się od ukośnika). Zmienna sterująca będzie przyjmować wartości całkowite od 1 do 10, więc instrukcja w środku w pętli rysująca okrąg o średnicy 0.4 cm będzie wykonana 10 razy. Zmieniać się będzie miejsce rysowania okręgu (środek), bo jego pierwsza współrzędna to aktualna wartość zmiennej `\i`. W efekcie otrzymujemy 10 okręgów.

Listing 11.16. Przykład użycia pętli

```
\begin{tikzpicture}
\foreach \i in {1,...,10} \draw (\i,0) circle (0.4cm);
\end{tikzpicture}
```



W podobny sposób pętla `\foreach` została użyta na listingu 11.4. W tamtym przypadku zmienna sterująca również była użyta do reprezentowania jednej ze współrzędnych, a w pętli dodawane były węzły.

Jeżeli chodzi o podanie zakresu wartości dla pętli `\foreach`, to mamy kilka możliwości. Po pierwsze można wyliczyć wszystkie wartości, np. `{2,4,7,12}`. Użycie wielokropka zakłada, że przed nim wystąpiły co najmniej 2 liczby, a po nim co najmniej jedna, tj. mamy sytuację $x, y, \ldots, z$. System wyznacza różnicę $d = y - x$. Jeżeli $d > 0$, to $x, y, \ldots, z$ zastępowane jest przez liczby $x, x+d, x+2d, \ldots$, które nie są większe od z. Jeżeli $d < 0$, to $x, y, \ldots, z$ zastępowane jest przez liczby

$x, x+d, x+2d, \ldots$, które nie są mniejsze od z. Jeżeli $d = 0$, to kompilacja skończy się błędem związanym z użyciem pamięci operacyjnej. Jeżeli wielokropek występuje bezpośrednio po pierwszej liczbie (tak było do tej pory w naszych przypadkach), tj. mamy $x, \ldots, z$, to system przyjmuje $d = 1$, jeśli $x > z$, i $d = -1$, jeśli $x < z$, i postępuje jak wcześniej.

Dodatkowo pakiet obsługuje kilka przypadków szczególnych. Jeśli zamiast liczb mamy litery, to d jest wyznaczane jako odstęp w alfabecie między znakami. Nie są wymagane dodatkowe nawiasy klamrowe, gdy wyliczamy punkty. Taka lista może mieć postać `{(2,1), (0,3), (4,2)}`. Ponadto funkcjonuje tzw. zastępowanie kontekstowe, np. `{p1,p2,p...,p6}` (zwracam uwagę na `p` przed wielokropkiem) wygeneruje listę `{p1,p2,p3,p4,p5,p6}`. W tabeli 11.7 pokazano przykłady list definiowanych z wielokropkiem i efekt ich użycia. Warto dodać, że w przypadku, gdy mamy do czynienia z liczbami rzeczywistymi, mogą pojawić się błędy w wyniku realizowanych operacji arytmetycznych, np. zamiast 2.6 dostaniemy 2.59999. Może to mieć znaczenie, gdy chcemy budować etykiety z użyciem generowanych liczb.

Tabela 11.7. Przykłady definiowania list z wielokropkiem

Definicja	Elementy listy
`{1,2,...,8}`	1, 2, 3, 4, 5, 6, 7, 8
`{1,3,4,...,8,10}`	1, 3, 4, 5, 6, 7, 8, 10
`{2,4,...,11}`	2, 4, 6, 8, 10
`{2,2.2,...,3.3}`	2, 2.2, 2.4, 2.6, 2.8, 3, 3.2
`{1,...,5,7,9,...,12}`	1,2,3,4,5,7,9,11
`{a,...,f}`	a, b, c, d, e, f
`{A,C,...,K}`	A, C, E, G, I, K
`{Z,T,...,A}`	Z, T, N, H, B

Pętle możemy w sobie zagnieżdżać. Przykład pokazano na listingu 11.17. Warto zwrócić uwagę na linię 3, gdzie wykonujemy prostą operację matematyczną na zmiennych sterujących. Można również jednocześnie iterować na wielu zmiennych sterujących w ramach pojedynczej pętli, ale wymaga to przygotowania listy wartości dla układu zmiennych z użyciem ukośnika (`/` – zwykły ukośnik, wyjątkowo!).

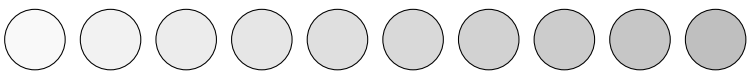
Listing 11.17. Przykład zagnieżdżenia pętli

```
1 \begin{tikzpicture}[vertex/.style={minimum size=4mm, fill=gray!30}]
2 \foreach \i in {1,...,8}
3   \foreach \j in {1,...,8} \node[vertex] at (\i*0.5,\j*0.5) {};
4 \end{tikzpicture}
```

Na listingu 11.18 mamy zmodyfikowany kod rysujący koła. Druga zmienna została użyta przy wyznaczeniu koloru do zamalowania wnętrza. Przy pierwszej iteracji `\i` przyjmuje wartość 1, a `\j` wartość 5, przy drugiej iteracji odpowiednio 2 i 10 itd. Jeżeli jest taka potrzeba, możemy stosować układ trzech zmiennych, np. `\i/\j/\k`, i przygotować dane na liście w postaci odpowiednich trójek, np. `10/10/red`.

Listing 11.18. Przykład użycia pętli z dwoma zmiennymi sterującymi

```
1 \begin{tikzpicture}
2 \foreach \i/\j in {1/5,2/10,3/15,4/20,5/25,6/30,7/35,8/40,9/45,10/50}
3   \draw[fill=gray!\j] (\i,0) circle (0.4cm);
4 \end{tikzpicture}
```



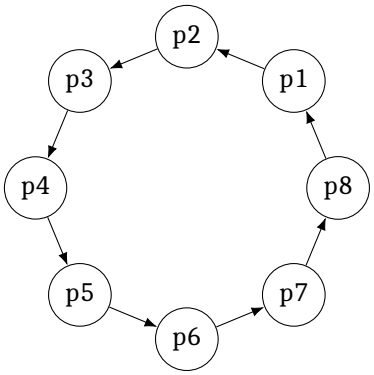
Polecenie `\foreach` może przyjąć kilka opcji, które mogą być pomocne przy rysowaniu. Dwie najciekawsze to `remember` i `count`. Pierwsza z nich pozwala na zapamiętanie bieżącej wartości zmiennej sterującej i użycie jej w kolejnej iteracji, druga na zliczanie iteracji pętli – dostarcza informację o numerze bieżącej iteracji.

Na listingu 11.19 pokazano kod rysujący graf z połączonych ze sobą węzłów rozmieszczonych w wierzchołkach ośmiokąta foremnego. Użyta tutaj metoda jest następująca: rysujemy pierwszy węzeł (`p1` – linia 3), a następnie w pętli dodajemy kolejny węzeł i krawędź od poprzedniego. Na koniec pozostaje nam dorysować ostatnią brakującą krawędź (linia 7). Realizacja zadania w pętli będzie możliwa tylko wtedy, gdy będziemy mieć dostęp do nazwy poprzedniego i bieżącego węzła. Oczywiście numer bieżącego węzła możemy mieć w zmiennej sterującej pętlą. Opcji `remember` użyjemy, aby pamiętać numer poprzedniego węzła. Kod w linii 4: `remember=\i as \j (initially 1)` oznacza, że do zapamiętywania poprzedniej wartości zmiennej `\i` użyjemy zmienną `\j`, a przy pierwszym przebiegu pętli (nie ma wtedy jeszcze poprzedniej wartości)

zmienna `\j` będzie miała wartość 1 (domyślnie byłoby 0). Przy pierwszym przebiegu pętli `\i` ma wartość 2, więc rysujemy węzeł `p2` (linia 5) i krawędź od `p1` do `p2` (linia 6). Przy drugim przebiegu pętli `\i` ma wartość 3, a `\j` – 2, więc rysujemy węzeł `p3` i krawędź od `p2` do `p3` itd. Warto jeszcze zwrócić uwagę na wyznaczanie współrzędnych, w których umieszczane są węzły. Używamy współrzędnych biegunowych (linie 3 i 5). To pozwala łatwo umieścić węzły w równych odstępach na okręgu.

Listing 11.19. Przykład użycia konstrukcji `remember` w pętli

```
1 \begin{tikzpicture}
2   [vertex/.style={draw,circle,minimum size=8mm},>=LaTeX]
3 \node[vertex] (p1) at (45:2) {p1};
4 \foreach \i [remember=\i as \j (initially  1)] in {2,...,8} {
5   \node[vertex] (p\i) at (\i*45:2) {p\i};
6   \draw[->] (p\j) -- (p\i); }
7 \draw[->] (p8) -- (p1);
8 \end{tikzpicture}
```



Ten sam efekt można uzyskać, korzystając z opcji `count`, co przedstawiono na listingu 11.20. Tym razem zadanie zostało podzielone na dwie pętle – w pierwszej rysujemy węzły, a w drugiej krawędzie. Podobnie jak poprzednio, ostatnia krawędź została dorysowana poza pętlą. W pierwszej pętli (linie 3–4) zmienna sterująca `\i` przyjmuje wartości 45, 90, 135 itd., czyli wartości interesujących nas kątów. Drugiej zmiennej używamy tutaj do liczenia, który mamy przebieg pętli, i numerowania węzłów od 1 do 8. Opcja `count=\j from 1` oznacza, że do liczenia przebiegów pętli użyjemy zmiennej `\j` i zaczynamy liczenie od 1 (wartość początkową możemy wybrać według potrzeb). Stąd przy pierwszym przebiegu pętli węzeł zostanie oznaczony jako `p1`, przy drugim `p2` itd. W drugiej pętli (linia 5) zmienna zliczająca przebiegi jest użyta do uzyskania oznaczenia poprzedniego węzła (`\i` zaczyna się od 2, a `\j` od 1, obie zmienne są zwiększane o 1 przy kolejnym przebiegu pętli).

Listing 11.20. Przykład użycia konstrukcji `count` w pętli

```
\begin{tikzpicture}
  [vertex/.style={draw,circle,minimum size=8mm},>=LaTeX]
\foreach \i [count=\j from 1] in {45,90,...,360}
  \node[vertex] (p\j) at (\i:2) {p\j};
\foreach \i [count=\j from 1] in {2,...,8} \draw[->] (p\j) -- (p\i);
\draw[->] (p8) -- (p1);
\end{tikzpicture}
```

Pakiet `tikz` pozwala na umieszczenie pętli `\foreach` wewnątrz instrukcji konstruującej ścieżkę. Podane na listingu 11.21 instrukcje będą miały dokładnie taki sam efekt. W przypadku wersji z linii 4 konieczny jest dodatkowy nawias klamrowy i można zrezygnować z ukośnika przy `\foreach`. Umieszczenie pętli w środku instrukcji nie jest tylko wybiegiem składniowym. Istotne zalety pokazano w kolejnym przykładzie.

Listing 11.21. Przesunięcie instrukcji pętli do wnętrza konstrukcji ścieżki

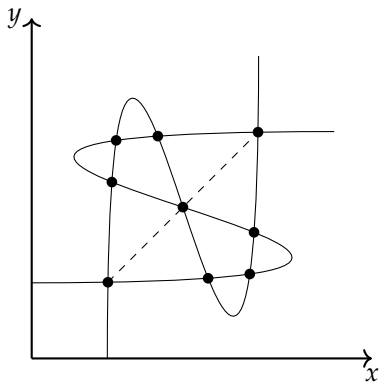
```
\begin{tikzpicture}
\foreach \i in {1,...,10} \draw (\i,0) circle (0.4cm);

\draw \foreach \i in {1,...,10} {(\i,0) circle (0.4cm)};
\end{tikzpicture}
```

Przesunięcie pętli `\foreach` do środka instrukcji otwiera nowe możliwości. Przeanalizujmy przykład z listingu 11.22 wzorowany na dokumentacji [46]. Dwie krzywe Béziera przecinają się aż w 9 punktach. Jeżeli nadamy im nazwy, to jak wiemy, możemy wyznaczyć wszystkie punkty przecięcia. W konstrukcji `name intersections=` mamy dodatkową opcję `total=\j`, która pozwala na zapamiętanie w zmiennej `\j` liczby znalezionych punktów przecięcia, co jest przydatne przy konstruowaniu uniwersalnych pętli, gdy nie wiemy z góry, ile takich punktów istnieje. Ponieważ pętla umieszczona jest wewnątrz instrukcji `\fill`, to ma dostęp do wartości tej zmiennej (`\j` nie jest dostępna poza instrukcją `\fill`). Pozwala to w wygodny sposób oznaczyć wszystkie punkty przecięcia. Przypominam, że poza instrukcją `\fill` nazwy punktów przecięcia pozostają dostępne, gdybyśmy chcieli z nich korzystać. Dla przykładu dorysowaliśmy linię łączącą pierwszy i ostatni z tych punktów (linia 10). W linii 10 mamy dostęp do nazw `i-1`, `i-2` itd., ale nie ma już dostępu do zmiennej `\j`.

Listing 11.22. Zastosowanie pętli w oznaczaniu punktów przecięcia krzywych

```
\begin{tikzpicture}
\clip (-0.5,-0.5) rectangle (4.7,4.7);
\draw[thick] [->] (0,0) -- (4.5,0) node [below] {$x$};
```

```
4 \draw[thick] [->] (0,0) -- (0,4.5) node [left] {$y$};
5 \draw[name path=l1] (0,1) .. controls (10,1) and (-6,3) .. (4,3);
6 \draw[name path=l2] (1,0) .. controls (1,10) and (3,-6) .. (3,4);
7
8 \fill [name intersections={of=l1 and l2, name=i, total=\j}]
9   \foreach \i in {1,...,\j} {(i-\i) circle (2pt)};
10 \draw[dashed] (i-1) -- (i-9);
11 \end{tikzpicture}
```

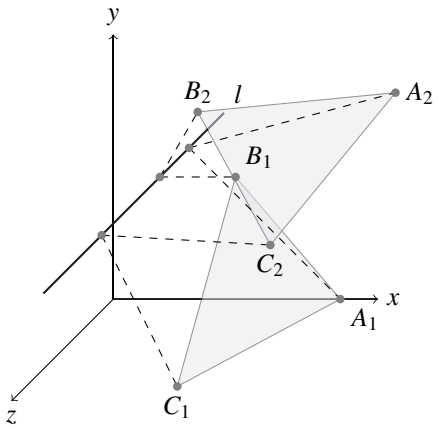


Na koniec warto jeszcze zwrócić uwagę na polecenie `\clip` (linia 1), które przycina rysunek do wskazanego prostokąta. Bez niego mielibyśmy wrażenie, że nasz rysunek ma zbędne duże marginesy, a to ze względu na współrzędne punktów kontrolnych, które choć niewidoczne, należą do rysunku. Rysunek możemy przyciąć do wnętrza dowolnej ścieżki. Ponadto instrukcję można wykorzystać, aby pokazywać oddzielnie różne fragmenty dużego rysunku. Jeżeli rysunek jako całość jest bardzo duży, to możemy go załączyć jako przeskalowany, aby tylko pokazać jego strukturę, a później na kolejnych rysunkach w czytelny sposób pokazywać jego fragmenty.

11.7 Rysowanie w trzech wymiarach

Pakiet `tikz` można wykorzystać do rysowania w kartezjańskiej przestrzeni trójwymiarowej. Oczywiście wynikiem jest tylko rzut na płaszczyznę, ale bez problemu możemy uzyskać wrażenie trójwymiarowości. Przeanalizujemy sobie, jak uzyskać wynik pokazany na rys. 11.6.

Rysunek ilustruje obrót wokół osi w przestrzeni trójwymiarowej. Obracamy trójkąt $A_1B_1C_1$. Wynikiem obrotu względem osi l jest trójkąt $A_2B_2C_2$. Dodatkowo na osi obrotu zaznaczono punkty będące rzutami wierzchołków trójkąta na oś. Ponieważ skupiamy się na przygotowaniu rysunku, a nie na



Rys. 11.6. Przykład rysowania w przestrzeni trójwymiarowej – obrót wokół osi

obliczeniach matematycznych, zakładamy, że znamy współrzędne wszystkich punktów (oczywiście dla pierwszego trójkąta zostały one przyjęte arbitralnie). W przypadku trójkąta $A_2B_2C_2$ konieczne jest wykonanie operacji na macierzach – tutaj obrót o 60° względem osi równoległej do osi z i przechodzącej przez punkt $(1,2,0)$.

Listing 11.23. Przykład rysowania w przestrzeni trójwymiarowej

```
\begin{tikzpicture}
\tikzset{mpoint/.style={circle, fill=gray, inner sep=0.4mm},
triangle/.style={fill=gray!20, opacity=0.5}}

\coordinate [label=below right:{$A_1$}] (A1) at (3,0,0);
\coordinate (B1) at (2,2,1);
\coordinate [label=below:{$C_1$}] (C1) at (2,0,3);
\coordinate [label=right:{$A_2$}] (A2) at (3.73,2.73,0);
\coordinate [label=above:{$B_2$}] (B2) at (1.5,2.86,1);
\coordinate (C2) at (3.23,1.87,3);
\coordinate (A) at (1,2,0);
\coordinate (B) at (1,2,1);
\coordinate (C) at (1,2,3);

\draw [->] (0,0,0) -- (3.5,0,0) node [right] {$x$};
\draw [->] (0,0,0) -- (0,3.5,0) node [above] {$y$};
\draw [->] (0,0,0) -- (0,0,3.5) node [below] {$z$};

\draw [thick] (1,2,5) -- (1,2,-1.2) node [above right] {$l$};
\filldraw [triangle] (A1) -- (B1) -- (C1) -- cycle;
```

```
21 \filldraw [triangle] (A2) -- (B2) -- (C2) -- cycle;
22 \draw [dashed] (A) -- (A1) (A) -- (A2) (B) -- (B1)
23   (B) -- (B2) (C) -- (C1) (C) -- (C2);
24
25 \foreach \i in {A,B,C,A1,B1,C1,A2,B2,C2} \node[mpoint] at (\i) {};
26
27 \node at (B1) [above right] {$B_1$};
28 \node at (C2) [below] {$C_2$};
29 \end{tikzpicture}
```

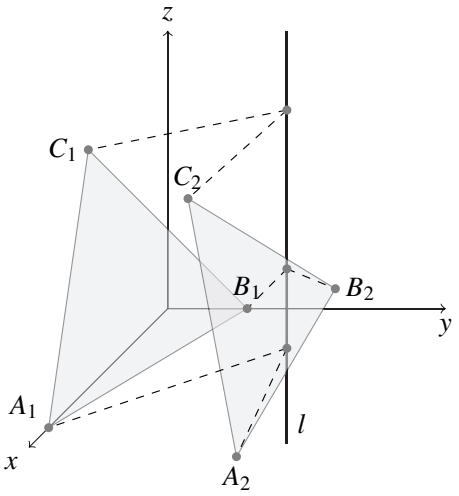
Kod źródłowy rysunku pokazano na listingu 11.23. Linie 5–13 to definicje współrzędnych dla naszych wszystkich ważnych punktów. Dla czterech punktów współrzędne są definiowane wraz z etykietami, co oznacza natychmiastowe dodanie tych etykiet do rysunku. Dwie ostatnie etykiety rysowane są dopiero na końcu (linie 27–28), aby nie zostały przesłonięte rysowanymi trójkątami. Polecenia w liniach 15–17 rysują trójwymiarowy układ współrzędnych, a polecenie w linii 19 oś obrotu. Oba trójkąty rysowane są w liniach 20–21. Korzystamy przy ich rysowaniu ze stylu `triangle` (linia 3), w którym m.in. ustawiliśmy 50-procentową przezroczystość. Polecenie `\draw` w liniach 22–23 kreśli pomocnicze linie między wierzchołkami trójkąta i ich rzutami na oś obrotu. Takie wyspecyfikowanie segmentów ścieżki pozwala na narysowanie kilku niepołączonych odcinków. Wreszcie w pętli (linia 25) dodajemy oznaczenia punktów charakterystycznych kreślone zgodnie ze stylem `mpoint` (linia 2).

Pakiet `tikz` umożliwia określenie, w jaki sposób chcemy zdefiniować osie x, y i z względem oryginalnego dwuwymiarowego układu współrzędnych. Przykład takiej specyfikacji podano na listingu 11.24 (konieczne jest użycie jednostek). Efekt tych zmian widać na rys. 11.7, przy czym w porównaniu z kodem z listingu 11.23 zmieniono położenie niektórych etykiet (i o 1 cm skrócono prostą l, bo wystawała znacznie powyżej osi z). Zaczynając od osi z, przypisaliśmy jej wektor $(0,1)$, więc będzie pionowo skierowana w górę, oś y z wektorem $(1,0)$ jest ułożona poziomo i skierowana w prawo. Dobór wektora dla osi x jest mniej oczywisty – tutaj przyjęto odchylenie 45° od prostej wyznaczonej przez oś z.

Listing 11.24. Przykład zmiany układu współrzędnych

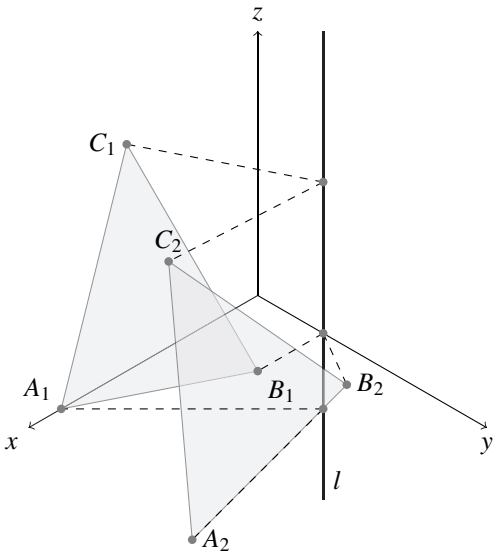
```
1 \begin{tikzpicture}
2 [x={(-0.5cm,-0.5cm)}, y={(1cm,0cm)}, z={(0cm,1cm)}]
3
4 % ... polecenie generujące rysunek
5 \end{tikzpicture}
```

Inne popularne ustawienie to rzut izometryczny (wszystkie kąty między osiami na rzucie wynoszą po 120°). Uzyskamy go, stosując następujące opcje:



Rys. 11.7. Obrót wokół osi – zmiana układu współrzędnych

`[x={(-0.866cm,-0.5cm)}, y={(0.866cm,-0.5cm)}, z={(0cm,1cm)}]`. Efekt widoczny jest na rys. 11.8. Tak jak poprzednio konieczna była korekta położenia niektórych etykiet. Układ współrzędnych możemy zmienić, również gdy rysujemy w dwóch wymiarach. Gdyby zaszła taka potrzeba, to ustawienie `[x=(0cm,1cm), y=(1cm,0cm)]` pozwala odwrócić osie (oś x będzie pionowa).



Rys. 11.8. Obrót wokół osi – rzut izometryczny

11.8 Osadzanie rysunków Ti*k*Z w dokumencie

Pakiet `tikz` oferuje dwie możliwości przygotowania rysunków. Poza środowiskiem **`tikzpicture`**, z którego korzystaliśmy na wszystkich listingach, dostępne jest polecenie **`\tikz`**, które jest alternatywą dla środowiska, gdy przygotowujemy proste rysunki. Polecenie przyjmuje jeden parametr w nawiasach klamrowych lub pojedyncze polecenie pakietu `tikz` (wszystko do pierwszego średnika). Polecenie pozwala na łatwe umieszczenie niewielkiego rysunku bezpośrednio w tekście (analogicznie jak dodajemy np. wzór matematyczny). Na przykład polecenie `\tikz\fill[gray] (0,0) -- (1,0) arc (0:30:1) -- cycle;` umieszcza ten wycinek koła.

Jeżeli chcemy umieścić rysunek przygotowany w `tikz` wewnątrz środowiska **`figure`** (aby dodać podpis, mieć numer i przede wszystkim wstawkę), to polecenie **`\tikz`** lub środowisko **`tikzpicture`** umieszczamy zamiast polecenia **`\includegraphics`**`{}`. Strukturę przykładowego kodu źródłowego pokazano na listingu 11.25.

Listing 11.25. Metoda osadzania środowiska **`tikzpicture`** w środowisku **`figure`**

```
\begin{figure}[!ht]   % odpowiednie opcje
\centering
\begin{tikzpicture}
  % polecenia tworzące rysunek
\end{tikzpicture}
\caption{...}
\label{fig:...}
\end{figure}
```

Istnieje również możliwość przygotowania rysunku w oddzielnym pliku i dołączenia go do dokumentu z użyciem polecenia **`\includegraphics`**`{}`. W tym celu przygotowujemy dokument, korzystając z klasy `standalone`. Klasa ta pozwala złożyć dokument zawierający tylko pojedynczy rysunek. Nie jest tutaj określony rozmiar papieru. Wielkość wynikowa zależy od zawartości i ustawionych marginesów (ustawiamy je za pomocą opcji `border` w klasie dokumentu, na przykład dodajemy opcję `[border=2pt]`). Strukturę takiego dokumentu pokazano na listingu 11.26. W wyniku dostajemy plik `pdf` z grafiką wektorową.

Listing 11.26. Przygotowanie rysunku z użyciem klasy `standalone`

```
\documentclass{standalone}
\usepackage[polish]{babel}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{newtx}
```

```
\usepackage{tikz}  % + inne potrzebne pakiety i biblioteki

\begin{document}
\begin{tikzpicture}
  % polecenia tworzące rysunek
\end{tikzpicture}
\end{document}
```

Jeżeli przygotowujesz artykuł naukowy czy książkę, która będzie później składana przez wydawnictwo, to lepszym rozwiązaniem jest przygotowanie rysunków w oddzielonych plikach zgodnie ze schematem z listingu 11.26 i dołączanie ich do dokumentu jako plików `pdf`. Mamy gwarancję, że będą wyglądać tak, jak je przygotowaliśmy, niezależnie od zabiegów, jakim poddany zostanie dokument w wydawnictwie.

Ponieważ istnieje możliwość dodania grafiki jako zawartości węzłów, to możemy budować rysunek, komponując go z innych. Podejście takie daje większe możliwości (swoboda umieszczania danej grafiki) niż np. rozmieszczanie rysunków w tabeli. Dodatkowo możemy dorysować nowe elementy, budując nowy rysunek, dla którego inne rysunki są tylko komponentami.

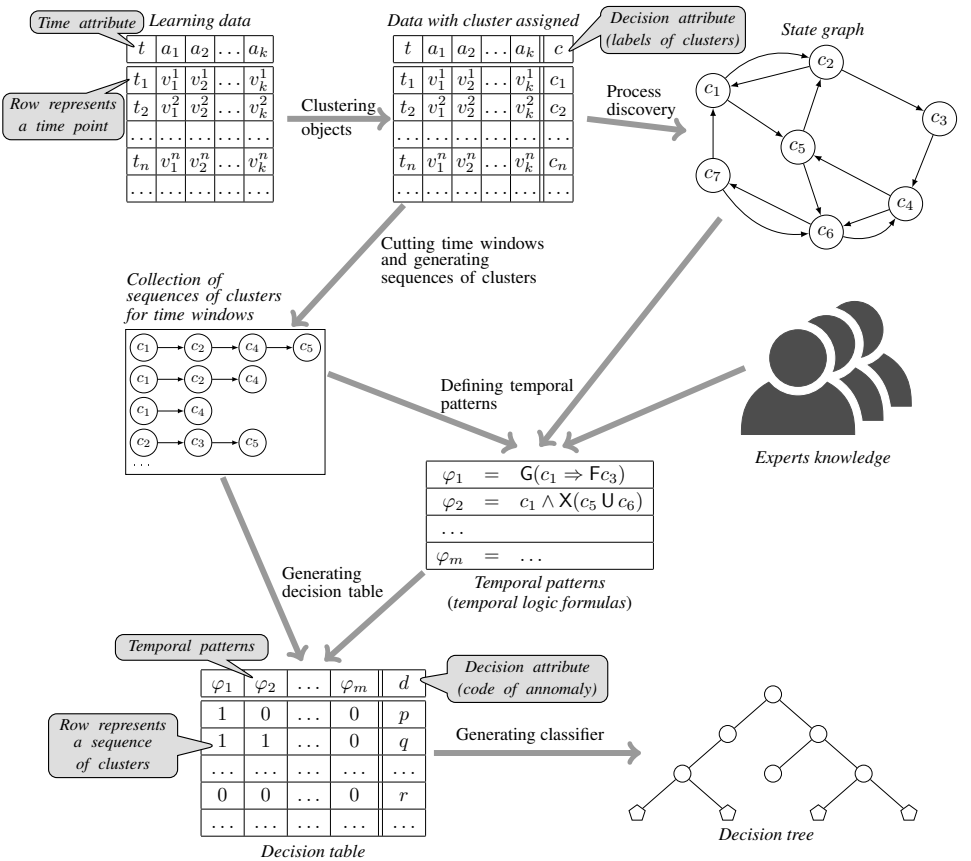
Na rys. 11.9 pokazano skonstruowaną w ten sposób grafikę pochodzącą z pracy [3]. Celem rysunku było graficzne przedstawienie idei metody postępowania opisywanej w artykule. Bazowa struktura tego rysunku składa się z ośmiu węzłów i połączeń między nimi. Przedstawiono ją na listingu 11.27 (bez etykiet tekstowych i dymków).

```
\usetikzlibrary{calc}
```

Listing 11.27. Szkielet kodu dla rys. 11.9

```
\tikzset{arc/.style={black!40!white,line width=3pt,->}}

\begin{tikzpicture}[scale=1,inner sep=0.4mm]
\node (e1) at (0,0) {...} % array
\node (e1b) at (5,0) {...} % array
\draw [arc] ($ (e1.east) +(0.2,0) $) -- (e1b);
\node (e2) at (11,-0.5) {\includegraphics[scale=1]{statesgraph}};
\draw [arc] ($ (e1b.east) +(0.2,0) $) -- (e2);
\node (e2b) at (0.4,-5) {\includegraphics[scale=0.8]{sequences}};
```

```
\node (e3) at (11,-4.4) {\includegraphics[scale=0.5]{people}};
\node (e4) at (6,-7) {...}; % array
\draw [arc] (e1b) -- (e2b);
\draw [arc] ($ (e2.south west) +(0.6,0.6) $) --
  ($ (e4.north) +(0,0.2) $);
\draw [arc] ($ (e3.west) +(0.4,0) $) -- ($ (e4.north) +(0.4,0.2) $);
\draw [arc] ($ (e2b.east) +(0.,0.5) $) -- ($ (e4.north) +(-0.4,0.2) $);
\node (e5) at (2,-11.2) {...}; % array
\draw [arc] ($ (e2b.south) +(0,0) $) -- ($ (e5.north) +(-0.2,0.1) $);
\draw [arc] ($ (e4.south west) +(0,0) $) --
  ($ (e5.north) +(0.2,0.1) $);
\node (e6) at (10,-11.2) {\includegraphics[scale=1]{classifier}};
\draw [arc] ($ (e5.east) +(0.1,0) $) -- ($ (e6.west) +(0.5,0) $);
\end{tikzpicture}
```



Rys. 11.9. Wieloetapowa konstrukcja rysunku w `tikz`

Kod rozpoczyna się od definicji stylu używanego do rysowania połączeń (linia 1). Następnie dodajemy dwa pierwsze węzły (linie 4–5). Ich zawartością są środowiska **array** (dwie tabele widoczne w pierwszym „wierszu"). Kolejne węzły ze środowiskiem **array** są dodawane w liniach 11 i 17 (tabele w centrum i na dole rysunku). Zawartością pozostałych czterech węzłów jest grafika ładowana z zewnętrznych plików (linie 7, 9, 10 i 21). Wszystkie te rysunki zostały wcześniej przygotowane z użyciem pakietu `tikz` jako dokumenty z klasą `standalone`. Rysunek `people` (linia 10) sam korzysta z zewnętrznej grafiki – powstał przez nałożenie na siebie trzy razy ikony reprezentującej człowieka. Węzły zostały połączone strzałkami, które nie dochodzą do krawędzi węzłów. Efekt taki uzyskano, odsuwając się o pewien wektor od wybranej kotwicy. Wszystkie takie obliczenia muszą być zapisane wewnątrz konstrukcji `($ $)` i wymagana jest biblioteka `calc`. Warto jeszcze dodać, że w kilku przypadkach zawartość węzłów wykorzystano w publikacji niezależnie. Ich kod został po prostu skopiowany do węzłów.

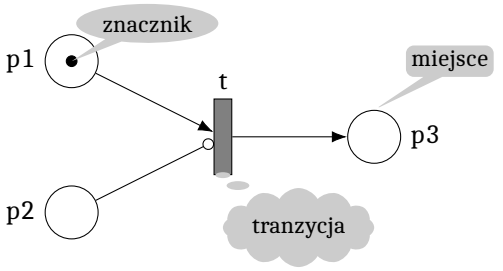
```
\usetikzlibrary{shapes.callouts}
```

Do przygotowania rys. 11.9 użyto także biblioteki `shapes.callouts`, która pozwala tworzyć etykiety w postaci „dymków" różnych kształtów. Sposób użycia tej biblioteki pokazano na listingu 11.28.

Listing 11.28. Przykład użycia biblioteki `shapes.callouts`

```
1  \begin{tikzpicture}
2  \node[place, label=left:p1] (p1) at (0,2) {};
3  \node[place, label=left:p2] (p2) at (0,0) {};
4  \node[transition, label=above:t] (t) at (2,1) {};
5  \node[place, label=right:p3] (p3) at (4,1) {};
6  \node[token] at (p1) {};
7
8  \draw (p1) edge[arc] (t)
9        (p2) edge[inhibitor] (t)
10       (t)  edge[arc] (p3);
11
12 \node[rectangle callout, fill=gray!40, rounded corners=3pt,
13   callout absolute pointer=(p3.north), inner sep=2pt] at (5,2)
14   {\small miejsce};
15
16 \node[ellipse callout, fill=gray!40, callout absolute pointer=(p1),
17   inner sep=2pt] at (1,2.5) {\small znacznik};
18
19 \node[cloud callout, cloud puffs=12, aspect=2.5, cloud puff arc=120,
```

```
20  fill=gray!40, callout absolute pointer=(t.south), inner sep=2pt]
21   at (3,-0.2) {\small tranzycja};
22 \end{tikzpicture}
```

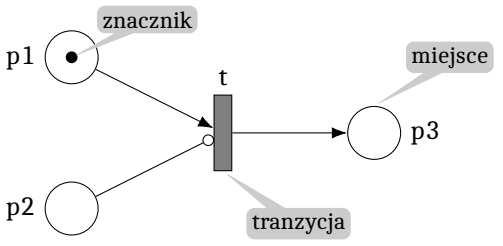


Trzy różne kształty dymków zostały dodane do wcześniej prezentowanego rysunku (zob. listing 11.15). Są to węzły opisane w liniach 12–21. W każdym przypadku pierwsza opcja wskazuje na kształt takiego węzła: `rectangle callout`, `ellipse callout` i `cloud callout`. Opcja `callout absolute pointer` określa miejsce, na które dany dymek wskazuje. Pozostałe opcje dotyczą wyłącznie doprecyzowania kształtu, np. zaokrąglone rogi w prostokącie lub liczba i kształt kłębów, z których złożona jest chmura.

Liczba parametrów rysowania dymku może nieco przerażać, ale należy pamiętać, że możemy je zapisać jako styl i wtedy kod upraszcza się do dwóch pozycji: nazwa stylu i parametr `callout absolute pointer`. Jeżeli sparametryzujemy styl, to unikniemy również tej drugiej opcji. Rozwiązanie takie pokazano na listingu 11.29. Założono, że wszystkie dymki będą tego samego kształtu, ale oczywiście można zdefiniować trzy różne style. Listing zawiera definicję stylu i przykłady jego użycia, pominięto część związaną z rysowaniem grafu.

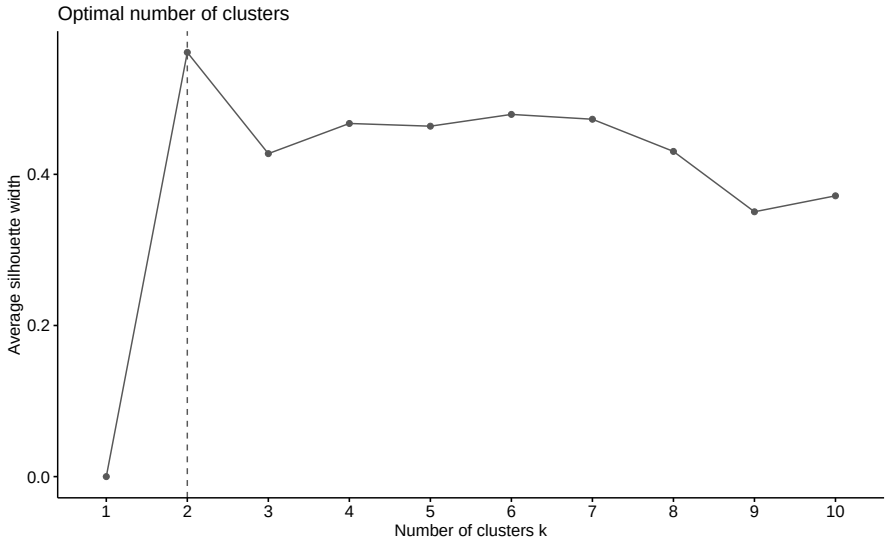
Listing 11.29. Sparametryzowany styl do rysowania dymków

```
1 \tikzset{notice/.style={rectangle callout, fill=gray!40,
2   rounded corners=3pt, callout absolute pointer=#1, inner sep=2pt}}
3
4 \node[notice=(p3.north)] at (5,2) {\small miejsce};
5 \node[notice=(p1)] at (1,2.5) {\small znacznik};
6 \node[notice=(t.south)] at (3,-0.2) {\small tranzycja};
```



11.9 Nakładanie wzorów na rysunek

Zdarza się, że naszym problemem nie jest przygotowanie rysunku „od zera", ale uzupełnienie rysunku, który już posiadamy, o jakieś elementy, np. etykiety z wzorami matematycznymi. Pakiet `\tikz` świetnie nadaje się do takiego zadania. Nawet jeżeli ktoś uznał, że przygotowanie rysunku z użyciem `tikz` jest dla niego zbyt skomplikowane, powinien docenić, jak łatwo i małym kosztem możemy uzyskać efekt przedstawiony w tym podrozdziale.



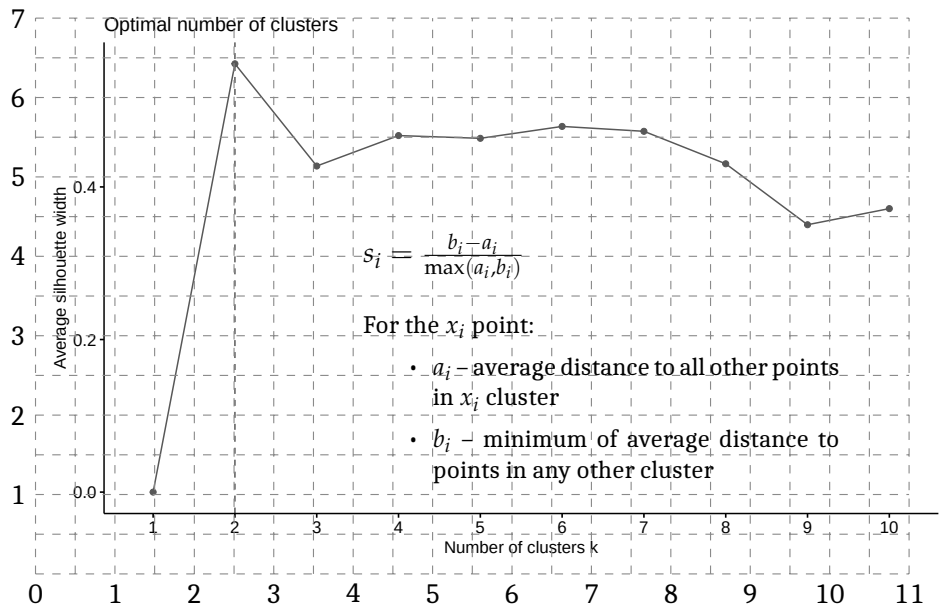
Rys. 11.10. Wykres wygenerowany w środowisku R

Rozważmy wykres przedstawiony na rys. 11.10. Został wygenerowany w języku R [27] i dotyczy wyboru optymalnej liczby klastrów dla algorytmu PAM (Partitioning Around Medoids). Powiedzmy, że chcemy wykorzystać ten wykres w prezentacji, ale chcielibyśmy dodać wzór definiujący współczynnik silhouette (na środku mamy dużo wolnej przestrzeni). Rozwiązanie przedstawiono na listingu 11.30, a efekt końcowy na rys. 11.11.

Listing 11.30. Dodawanie opisu w środowisku **`tikzpicture`**

```
\documentclass{standalone}
\usepackage[english]{babel}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{newtx}
\usepackage{tikz}

```

```
\begin{document}
\begin{tikzpicture}
\node[above right] at (0,0) {\includegraphics[scale=0.5]{pam.pdf}};
\draw[step=5mm,draw=gray,dashed] (0,0) grid (11,7);
\foreach \i in {0,...,11} \node[below] at (\i, 0) {\i};
\foreach \i in {1,...,7} \node[left] at (0, \i) {\i};

\node[right] at (4,4) {$s_i=\frac{b_i-a_i}{\max(a_i,b_i)}$};
\node[right] at (4,2.2) {\begin{minipage}{6cm}
\small
For the $x_i$ point:
\begin{itemize}
\item $a_i$ -- average distance to all other points in $x_i$ cluster
\item $b_i$ -- minimum of average distance to points in any other
  cluster
\end{itemize}
\end{minipage}};
\end{tikzpicture}
\end{document}
```



Rys. 11.11. Wykres z opisem dołączonym w środowisku `tikzpicture`

Po pierwsze zaczynamy od klasy `standalone`. Tym razem ze względu na rysunek bazowy dokument jest w języku angielskim. Wewnątrz środowiska

tikzpicture dodajemy węzeł z naszym rysunkiem z R (linia 10). Użyta opcja `above right` powoduje, że w punkcie (0,0) będzie lewy dolny róg rysunku. Oryginalny rysunek jest dość duży, więc jest skalowany (opcja `scale=0.5`). Następnie dodajemy tymczasowo naszą siatkę (linie 11–13). W tym przypadku wystarczy ona w wymiarze 11 na 7. Pozwala to bezbłędnie za pierwszym razem określić, gdzie chcemy umieścić dodatkowe etykiety (najlepiej z opcją `right`, wtedy etykieta będzie na prawo od wskazanego punktu). Pierwsza (linia 15) zawiera tylko jeden wzór matematyczny. W drugim węźle (linie 16–24) użyliśmy środowiska **minipage**, aby zamieścić wypunktowanie z wyjaśnieniem oznaczeń. Gdy efekt końcowy jest zadowalający, wystarczy zakomentować instrukcje kreślące siatkę i rysunek jest gotowy.

Poprawki nie muszą kończyć się na etykietach. Możemy dorysować coś nad rysunkiem. Sam rysunek tworzący tło może być równie dobrze zdjęciem, na którym chcemy coś zaznaczyć i dołączyć etykiety. Możliwości są nieograniczone, a jedyne, co było potrzebne w naszym przykładzie, to znajomość polecenia **\node** (mówimy o `tikz`, nie o znajomości LaTeX-a).

Podsumowując ten rozdział, warto dodać, że pakiet `tikz` kryje w sobie jeszcze dużo więcej możliwości. Jak wspomniałem we wstępie do tego rozdziału, dokumentacja dla pakietu `tikz` jest bardzo obszerna i warto do niej zaglądać, gdy chcemy rozwiązać konkretny problem. Do dalszej nauki warto sięgnąć po opracowania książkowe. W języku angielskim można znaleźć kilka pozycji poświęconych wyłącznie rysowaniu z użyciem pakietu `tikz`. Dobrym przykładem jest książka [32].

Rozdział 12

PGFPlots

Pakiet `tikz` w połączeniu z pakietem `pgfplots` pozwala na kreślenie wykresów funkcji. Jeżeli szukalibyśmy zalet wykorzystania tego pakietu na przykład względem narzędzi opartych na językach Python, R czy środowisku GNU Octave, to przede wszystkim dostajemy wysokiej jakości grafikę wektorową w pełni zgodną, jeśli chodzi o czcionki, z resztą dokumentu i to bez potrzeby szukania innych narzędzi poza LaTeX-em. O bogatych możliwościach pakietu `pgfplots` niech świadczy chociażby fakt, że jego dokumentacja [16] ma ponad 500 str. Pakiet umożliwia kreślenie w dwóch i trzech wymiarach, w tym krzywych i powierzchni zadanych parametrycznie z wykorzystaniem różnych układów współrzędnych. Ponadto mamy możliwość tworzenia wykresów punktowych, różnego typu wykresów słupkowych, histogramów itd. W tym rozdziale znajdziesz przegląd najciekawszych możliwości tego pakietu.

12.1 Dwuwymiarowy układ współrzędnych

Pakiet `pgfplots` bazuje na `tikz` i silniku PGF. Wszystkie wykresy tworzymy wewnątrz środowiska **`tikzpicture`**. Podstawowym środowiskiem do tworzenia wykresów jest środowisko **`axis`** (zagnieżdżone w **`tikzpicture`**). Tworzy ono układ współrzędnych, w którym mogą być dodawane wykresy. Domyślnie mamy do dyspozycji układ kartezjański liniowy. Zmieniając opcje środowiska, możemy dla każdej z osi niezależnie dodać skalę logarytmiczną. Możemy też korzystać ze środowiska **`polaraxis`**, które udostępnia biegunowy układ współrzędnych.

```
\usepackage{pgfplots}
\pgfplotsset{compat=1.18}
```

Wykresy wewnątrz środowiska tworzącego układ współrzędnych dodaje się za pomocą polecenia **\addplot**. Może ono być użyte wielokrotnie, jeśli chcemy umieścić kilka wykresów w jednym układzie współrzędnych. Polecenie funkcjonuje w kilku wersjach, zależnie od tego, jakie dane dostaje na wejściu do wygenerowania wykresu. Najciekawsze opcje to możliwość podania danych tabelarycznych (również czytanych z pliku) lub wyrażenia, które w sposób matematyczny określa zależności między zmiennymi. Zarówno środowisko (na razie skupimy się na **axis**), jak i polecenie **\addplot** mogą przyjąć szereg opcji. W przypadku opcji środowiska przyjmiemy zasadę, że każda opcja podawana jest w oddzielnej linii i na końcu linii stawiamy przecinek. Dopuszczalne jest pozostawienie przecinka po ostatniej opcji. Jest to o tyle wygodne rozwiązanie, że dodając kolejną opcję, mamy pewność, że nie zapomnimy o przecinku. Równie łatwo zakomentować dowolną linię z opcjami, nie martwiąc się o przecinki.

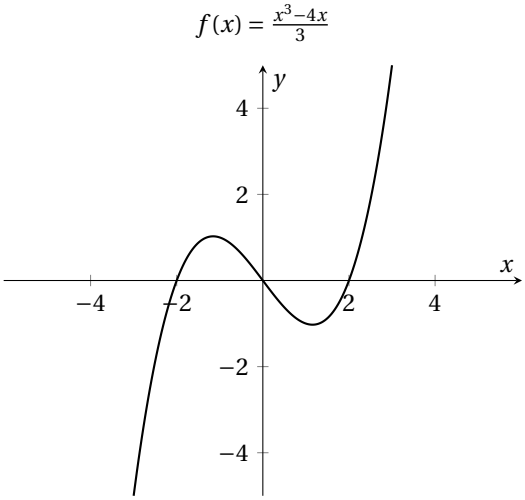
Podawanie opcji nie jest niezbędne. Pakiet wykorzystuje predefiniowaną listę stylów dla wykresów (określaną w dokumentacji jako `cycle list`) i jeżeli odpowiadają nam domyślne ustawienia, to lista opcji może być całkowicie pominięta. Lista jest zaprojektowana tak, aby przy dodawaniu kilku wykresów w ramach jednego układu współrzędnych były one kreślone z różnymi stylami. Jeżeli dodajemy wykres poleceniem **\addplot**`[] ...`, to predefiniowana lista stylów jest całkowicie ignorowana i używane są wyłącznie nasze opcje. Dostępna jest również wersja polecenia w postaci **\addplot+**`[] ...`. W tym przypadku nasze opcje zostaną dodane do domyślnych.

Użyty w opisie polecenia **\addplot** wielokropek ma wskazywać na różne wersje tej konstrukcji. Różnią się one zależnie od tego, co jest źródłem danych dla wykresu. Jedną z możliwości jest użycie wyrażenia matematycznego. Określamy dla niego w opcjach dziedzinę (zakres próbkowania) i liczbę próbek (powiedzmy n). Na tej podstawie wyliczanych jest n punktów należących do wykresu i przez nie prowadzona jest krzywa (lub łamana). Przykład tak generowanego wykresu przedstawiono na listingu 12.1.

Listing 12.1. Generowanie wykresu funkcji w układzie kartezjańskim

```
\begin{tikzpicture}
\begin{axis}[
  axis lines=center,
  axis equal,
  xtick={-4,-2,2,4},
  xlabel=$x$,
```

```
7   ylabel=$y$,
8   title={$f(x)=\frac{x^3-4x}{3}$},
9 ]
10 \addplot[domain=-3:3, samples=80, smooth, thick] {(x^3-4*x)/3};
11 \end{axis}
12 \end{tikzpicture}
```



W powyższym przykładzie kreślimy wykres funkcji $f(x) = \frac{x^3-4x}{3}$ (linia 10). Wykres powstaje przez wyznaczenie 80 punktów (o odciętych z dziedziny funkcji $[-3,3]$) i przeprowadzenie gładkiej krzywej przez te punkty (opcja `smooth`). Na wygląd uzyskanego rysunku wpływ mają również opcje środowiska **`axis`**. W tym przypadku wybraliśmy „szkolną" wersję układu współrzędnych z osiami przecinającymi się w punkcie $(0,0)$ (opcja `axis lines=center`) z równymi jednostkami na obu osiach (opcja `axis equal`) i wskazaliśmy miejsca umieszczenia znaczników na osi x (bez tego były jeszcze znaczniki w punktach -6 i 6, które nic nie wnosiły do wykresu). Ponadto zdefiniowana została etykieta dla osi x i y oraz tytuł wykresu. Jak widać, bez problemu możemy tutaj korzystać ze wzorów matematycznych LaTeX-a, a użyte czcionki są takie same jak ustawienia dla całego dokumentu.

Definiując funkcję, możemy zamiennie korzystać z `x` i `\x`. Z perspektywy pakietu mają to samo znaczenie. Jeżeli jest taka potrzeba, to można zmienić nazwę zmiennej, dodając opcję `variable`, np. `variable=t` i w wyrażeniu używać zmiennej `t` (opcja jest dostępna i dla środowiska, i dla polecenia **`\addplot`** – możemy mieć różne zmienne dla różnych wykresów). Możemy też korzystać z operatorów: `+`, `-`, `*`, `/`, `^` (potęgowanie), `!` (silnia), stałych `pi`, `e` i nazw popularnych funkcji. Przegląd wybranych funkcji dostępny jest w tabeli 12.1.

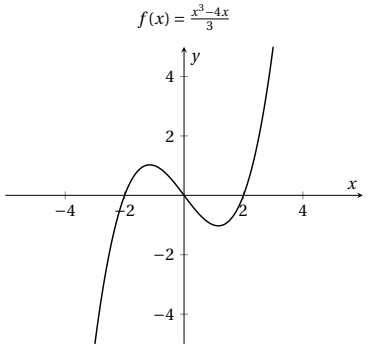
Tabela 12.1. Wybrane funkcje dostępne w pakiecie `pgfplots`

Opcja	Opis
`abs`	wartość bezwzględna
`round`	zaokrąglenie do najbliższej liczby całkowitej
`floor`, `ceil`	funkcje „podłoga” i „sufit”
`mod`	x modulo y (dwa argumenty)
`sqrt`	pierwiastek kwadratowy
`ln`, `log2`, `log10`	funkcje logarytmiczne
`exp`	eksponent e^x
`min`, `max`	minimum i maksimum (dowolna liczba argumentów)
`sin`, `cos`, `tan`, `cot`	funkcje trygonometryczne
`asin`, `acos`, `atan`	funkcje cyklometryczne
`deg`	konwersja radianów na stopnie
`rad`	konwersja stopni na radiany
`rand`	liczba losowa z przedziału $[-1,1]$
`rnd`	liczba losowa z przedziału $[0,1]$

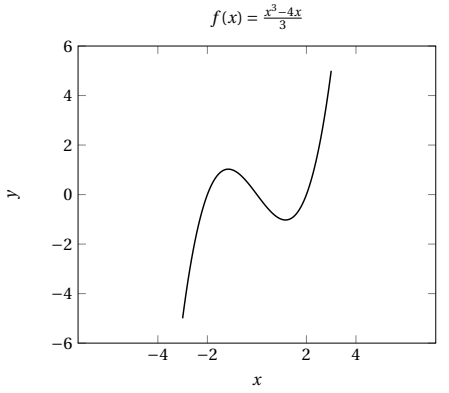
Korzystając z funkcji trygonometrycznych, należy pamiętać, że operują one na stopniach! Można użyć funkcji `deg` konwertującej radiany na stopnie.

W przedstawionym przykładzie zastosowano składnię `\addplot[] {wyrażenie matematyczne}`. Jest to skrócona wersja polecenia `\addplot[] expression {wyrażenie matematyczne}`. W pełnej wersji słowo kluczowe `expression` jawnie wskazuje, z jaką wersją polecenia **`\addplot`** mamy do czynienia. Jeżeli podajemy w nawiasach klamrowych pojedyncze wyrażenie `{f(x)}`, to w zasadzie opisujemy funkcję $f\colon x \to f(x)$. Równoważnie możemy podać przepis na funkcję jako `(x,f(x))`. Ta druga możliwość jest o tyle istotna, że pozwala określić funkcję parametrycznie `(x(t),y(t))` (zakładamy, że `t` jest nazwą zmiennej). To specjalne znaczenie nawiasów okrągłych powoduje, że powstanie problem z interpretacją, jeżeli w opisie `x(t)` lub `y(t)` użyjemy nawiasów. Rozwiązujemy go, opakowując dodatkowo wzór funkcji nawiasami klamrowymi, zatem za bezpieczną strukturę możemy przyjąć `({x(t)},{y(t)})`.

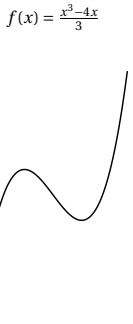
Układ współrzędnych jest domyślnie reprezentowany przez opcję `box` (rys. 12.1b). Możemy ponadto użyć opcji `center` (rys. 12.1a) lub `none` (rys. 12.1c). Wybór wyglądu całego układu realizowany jest przez przypisanie odpowiedniej wartości do opcji `axis lines`. Jeżeli chcemy niezależnie ustawiać każdą z osi, to mamy do dyspozycji opcje `axis x line` i `axis y line`. Dopuszczalne wartości dla tych opcji to: `box`, `bottom`, `center`, `top`, `left`, `right` i `none`. Trzy różne kombinacje pokazano na rys. 12.1d–12.1f.



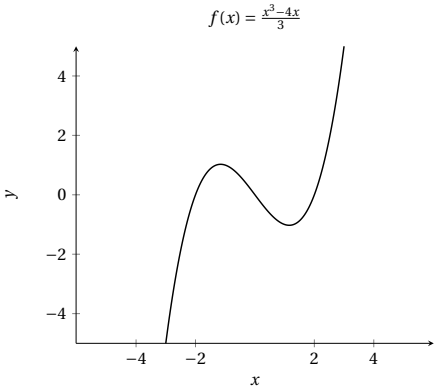
(a) x, y: `center`



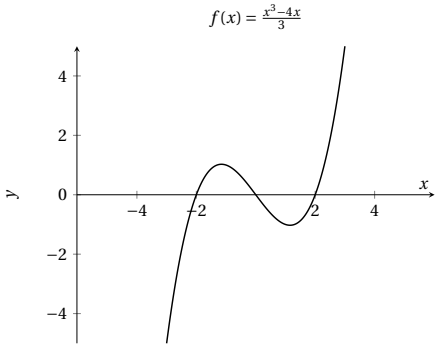
(b) x, y: `box`



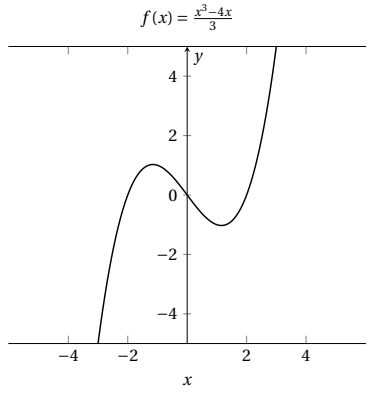
(c) x, y: `none`



(d) x: `bottom`, y: `left`



(e) x: `center`, y: `left`



(f) x: `box`, y: `center`

Rys. 12.1. Sposób wyświetlania układu współrzędnych zależnie od ustawień

Tabela 12.2. Wybrane opcje środowiska **`axis`**

Opcja	Opis
`title`	tytuł wykresu
`title style`	styl tytułu
`xlabel`, `ylabel`	etykiety osi x i y

`label style`
`xlabel style`
`ylabel style`
	styl etykiety: obie, oś x, oś y

`xmode`, `ymode`	typ osi: `normal` lub `log` (skala logarytmiczna)
`domain`	dziedzina (przedział próbkowania)
`xmin`, `xmax`	minimalna i maksymalna wyświetlana wartość x
`ymin`, `ymax`	minimalna i maksymalna wyświetlana wartość y

`axis lines`
`axis x line`
`axis y line`
	sposób wyświetlania osi: obie osie, oś x, oś y

`axis line style`
`x axis line style`
`y axis line style`
	styl osi: obie osie, oś x, oś y

`xminorticks`, `yminorticks` `xmajorticks`, `ymajorticks`	`true` lub `false` – włączenie i wyłączenie poszczególnych typów znaczników
`xtick` `ytick`	lista miejsc podstawowych znaczników: oś x, oś y
`xticklabels` `yticklabels`	lista etykiet dla znaczników ustawionych za pomocą: `xtick`, `ytick`

`ticklabel style`
`xticklabel style`
`yticklabel style`
	styl etykiet znaczników: obie osie, oś x, oś y

`xtick distance` `ytick distance`	odległość między podstawowymi znacznikami: oś x, oś y

`minor tick num`
`minor x tick num`
`minor y tick num`
	liczba znaczników drugiego rzędu między podstawowymi: obie osie, oś x, oś y

`minor xtick` `minor ytick`	lista miejsc dla znaczników drugiego rzędu: oś x, oś y
`tick style`	styl rysowania podstawowych znaczników
`minor tick style`	styl rysowania znaczników drugiego rzędu
`major tick length`	długość znaczników podstawowych
`minor tick length`	długość znaczników drugiego rzędu

`tick align`
`xtick align`
`ytick align`
	położenie znaczników względem osi: obie osie, oś x, oś y, dopuszczalne wartości: `inside`, `center`, `outside`

`grid`	ustawienia siatki, wartości: `minor`, `major`, `both`, `none`
`xminorgrids`, `yminorgrids` `xmajorgrids`, `ymajorgrids`	`true` lub `false` – włączenie i wyłączenie poszczególnych linii
`major grid style` `minor grid style`	styl siatki
`legend entries`	lista wpisów do legendy
`legend pos`	pozycja legendy: `south west`, `south east`, `north west`, `north east`, `outer north east`
`legend cell align`	wyrównanie pozycji legendy: `left`, `right`, `center`
`legend columns`	liczba kolumn w legendzie
`legend style`	styl legendy
`width`	szerokość wykresu
`height`	wysokość wykresu

W tabeli 12.2 zebrano większość opcji, których można użyć w środowisku **`axis`**, aby dopasować wygląd wykresu. Opcja `title`, jak już pokazano, pozwala zdefiniować tytuł dla wykresu. Wygląd tytułu możemy określić za pomocą opcji `title style`, np. `title style={font=\small,color=red}`. Kolejne opcje pozwalają zdefiniować etykiety dla poszczególnych osi i ich styl (analogicznie jak styl tytułu). Jeżeli chcemy na którejś z osi ustawić skalę logarytmiczną, to należy skorzystać z opcji `xmode` lub `ymode` i ustawić wartość `log` (można na obu osiach). Dziedzinę możemy zdefiniować zarówno jako opcję środowiska, jak i opcję polecenia **`\addplot`**. Jeżeli dla zadanej dziedziny funkcja osiąga bardzo małe lub duże wartości, to opcje `xmin`, `xmax` itd. pozwalają przyciąć zakres wartości dla x i/lub y, który ma być wyświetlany na wykresie. Opcje te funkcjonują niezależnie od dziedziny.

Różne kombinacje opcji definiujących układ osi pokazano na rys. 12.1. Opcja `axis line style` pozwala ustawić wygląd osi (możemy też zrobić to indywidualnie dla każdej osi), np. `axis line style={thick,color=blue,-}` (ostatnia opcja oznacza brak grotu). Jeżeli chcemy dodać znaczniki do osi, to dostępne są dwie metody. Możemy skorzystać z opcji `xtick distance` i `ytick distance`, aby znaczniki główne zostały wygenerowane automatycznie w równych odstępach. Możemy również samodzielnie dodać znaczniki, używając opcji `xtick` i `ytick`. W tym podejściu po prostu podajemy listę miejsc, gdzie te znaczniki mają się znaleźć. Można tutaj skorzystać z metod pokazanych w tabeli 11.7. Dla tak zdefiniowanych znaczników możemy sami zdefiniować etykiety, używając opcji

`xticklabels` i `yticklabels`. Drugorzędne znaczniki na osi są wyświetlane, tylko gdy odstępy między znacznikami głównego poziomu są równe. Jeśli główne znaczniki były generowane automatycznie, to za pomocą opcji `minor tick num` możemy określić, ile znaczników drugiego rzędu ma być wstawionych między każdą parę znaczników podstawowych. Jeżeli potrzebujemy sami ustawić pozycję dla takich znaczników, to do dyspozycji mamy opcje `minor xtick` i `minor ytick`. Z kolei opcje `tick align` pozwalają określić, czy znaczniki widoczne są po obu stronach osi, czy przesunięte tylko na wewnętrzną lub zewnętrzną stronę. Analogicznie jak dla znaczników, możemy ustalić, które linie siatki będą widoczne (dla głównych znaczników i/lub dla drugorzędnych).

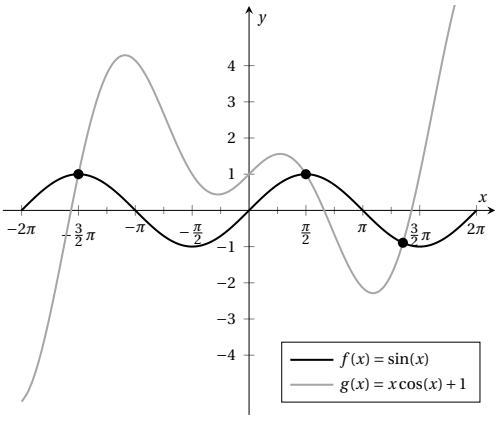
Przedostatnia grupa dotyczy ustawień legendy. Legenda jest zazwyczaj dodawana, gdy na jednym rysunku mamy więcej niż jeden wykres. Wpisy na listę (opcja `legend entries`) powinny być podawane w takiej kolejności, jak dodawane są wykresy. Domyślnie legenda składana jest w jednej kolumnie. Opcja `legend columns` pozwala zmienić te ustawienia. Jeżeli liczba kolumn będzie równa liczbie wpisów, to dostaniemy legendę w postaci jednego wiersza. Ostatnie dwie opcje pozwalają zmienić domyślną szerokość i wysokość wykresu.

W przykładzie z listingu 12.2 zademonstrowano kilka popularnych opcji, które pozwalają dostosować finalny rysunek do naszych potrzeb. Po pierwsze na tym wykresie wykreśliliśmy dwie funkcje (linie 25–26) i nadaliśmy nazwy tym ścieżkom. To pozwoliło wyznaczyć i oznaczyć punkty przecięcia krzywych (linie 27–28). W opcjach środowiska zdefiniowaliśmy etykiety dla osi z użyciem wzorów matematycznych (linie 5–6). Dziedzina dla obu funkcji została zdefiniowana z użyciem stałej `pi`, ale ograniczyliśmy zakres dla zmiennej y (linie 8–9), bo wykres drugiej funkcji na końcach dziedziny gwałtownie maleje/rośnie. Dodatkowo określenie minimalnej i maksymalnej wartości x poza dziedziną (linie 10–11) wydłużyło nieco oś x (grot osi x nie zbiega się teraz z wykresem pierwszej funkcji). Czcionka dla etykiet znaczników również została zmniejszona (linia 12) i wskazaliśmy, w których miejscach mają być umieszczone etykiety (linie 14 i 18). W końcu zdefiniowaliśmy eleganckie etykiety dla osi x z użyciem wzorów matematycznych (linie 15–16). Linie 19–22 to ustawienia dla legendy, kolejno: zawartość, położenie, sposób wyrównania wpisów (domyślnie są wycentrowane) i rozmiar czcionki. Ostatnia linia w opcjach środowiska to styl, którego używamy przy kreśleniu wykresów. Poza stylem możemy dostosować wygląd krzywej w opcjach polecenia **`\addplot`**, np. określić kolor (linia 26), ale również styl linii – dostępne są opcje typowe dla pakietu `tikz` (`dashed`, `dotted` itd.).

```
\usetikzlibrary{intersections}
```

Listing 12.2. Ustawianie opcji generowania wykresu funkcji

```
\begin{tikzpicture}
\begin{axis}[
  axis lines=center,
  axis equal,
  xlabel=$x$,
  ylabel=$y$,
  domain=-2*pi:2*pi,
  ymin=-5,
  ymax=5,
  xmin=-6.8,
  xmax=6.8,
  label style={font=\scriptsize},
  tick label style={font=\scriptsize},
  xtick={-2*pi, -(3/2)*pi,-pi, -(1/2)*pi, 1/2*pi, pi, 3/2*pi, 2*pi},
  xticklabels={$-2\pi$, $-\frac{3}{2}\pi$, $-\pi$, $-\frac{\pi}{2}$,
    $\frac{\pi}{2}$, $\pi$, $\frac{3}{2}\pi$, $2\pi$},
  minor xtick={-5*pi/4,-3*pi/4, -pi/4, pi/4, 3*pi/4, 5*pi/4},
  ytick={-4,-3,-2,-1,1,2,3,4},
  legend entries={$f(x)=\sin(x)$, $g(x)=x\cos(x)+1$},
  legend pos=south east,
  legend cell align=left,
  legend style={font=\scriptsize},
  fun/.style={thick, samples=80, smooth}
]
\addplot[name path=f1, fun] {sin(deg(x))};
\addplot[name path=f2, fun, gray!70] {x*cos(deg(x))+1};
\fill[name intersections={of=f1 and f2, name=p, total=\j}]
  \foreach \i in {1,...,\j} {(p-\i) circle (2pt)};
\end{axis}
\end{tikzpicture}
```



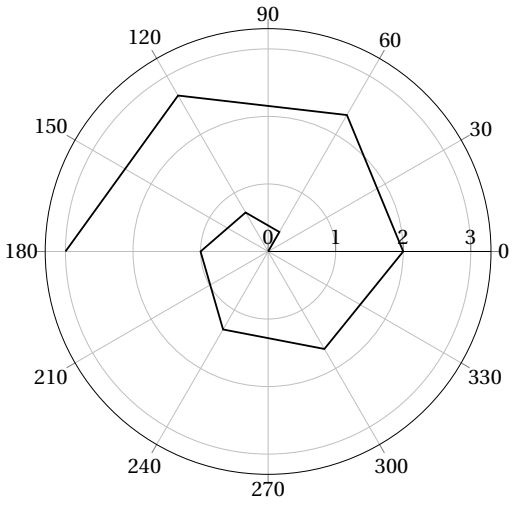
W opcjach środowiska **axis** użyliśmy określenia wielkości czcionek **\scriptsize**. Są one określone względem bazowej czcionki dokumentu. Jeżeli bezpośrednio w dokumencie użyjemy środowiska **tikzpicture**, to zmiana czcionki bazowej w opcjach polecenia **\documentclass** będzie miała wpływ na rozmiar czcionki na rysunku. Jeżeli rysunek przygotujemy w oddzielnym pliku, stosując klasę `standalone`, i osadzimy w dokumencie jako plik `pdf`, to ewentualna zmiana czcionki w dokumencie nie będzie miała żadnego wpływu na wygląd rysunku.

```
\usepgfplotslibrary{polar}
```

Biegunowy układ współrzędnych jest dostępny po dodaniu w preambule dokumentu biblioteki `polar`. Wykres taki tworzymy w środowisku **polaraxis**. Przyjmuje się, że współrzędna x zawsze oznacza kąt, a y promień. Domyślnie kąty podawane są w stopniach, ale dostępna jest opcja pozwalająca na podawanie kątów w radianach. Przykładowy wykres pokazano na listingu 12.3.

Listing 12.3. Kreślenie krzywych w układzie biegunowym

```
\begin{tikzpicture}
\begin{polaraxis}
\addplot[thick, samples=10, sharp plot, domain=0:3] (180*x, x);
\end{polaraxis}
\end{tikzpicture}
```

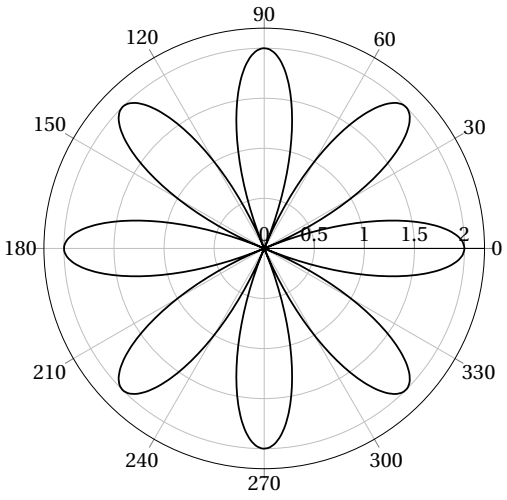


W podanym przykładzie krzywa jest zadana parametrycznie. Dziedziną jest przedział $[0,3]$ i mamy tylko 10 punktów. Oznacza to, że nasza dziedzina sprowadza się do zbioru $\{0, \frac{1}{3}, \frac{2}{3}, 1, 1\frac{1}{3}, 1\frac{2}{3}, 2, 2\frac{1}{3}, 2\frac{2}{3}, 3\}$. Dla tych wartości x wyznaczane są punkty postaci $(180x, x)$, a wykres jest łamaną przechodzącą przez kolejne punkty (decyduje o tym opcja `sharp plot` użyta zamiast `smooth`). Ten sam efekt uzyskalibyśmy, przyjmując jako dziedzinę zakres `0:540`, a funkcję definiując jako `{x/180}` (co jest równoważne zapisowi parametrycznemu `(x,x/180)`.

Na listingu 12.4 pokazano kod do wykreślenia róży polarnej z 8 płatkami. Liczba płatków zależy od wartości, przez którą mnożymy x w argumencie funkcji cosinus. Długość płatków zależy od liczby przed funkcją cosinus.

Listing 12.4. Kreślenie krzywych w układzie biegunowynm – róża polarna

```
\begin{tikzpicture}
\begin{polaraxis}
\addplot[domain=0:360, thick, samples=200, smooth] {2*cos(4*x)};
\end{polaraxis}
\end{tikzpicture}
```



W wykresach z biegunowym układem współrzędnych możemy dodawać etykiety osi, legendy, formatować etykiety itd. W szczególności użycie opcji `xmin` i `xmax` powoduje ograniczenie zakresu kątów i uzyskamy wykres częściowy.

12.2 Praca z danymi tabelarycznymi

Podstawowa forma polecenia kreślącego wykresy na postać `\addplot[] table[] {}`. Funkcja przyjmuje na wejściu dane tabelaryczne. Tabela może być podana

bezpośrednio w kodzie w nawiasach klamrowych, ale można też wczytać dane z zewnętrznego pliku (wtedy w klamrach podajemy nazwę pliku). Tabela może zawierać wiele kolumn, przy kreśleniu wykresu wskazujemy, która kolumna zawiera dane dla zmiennej x, a która dla y. Domyślnym formatem jest plik tekstowy, w którym separatorem komórek jest spacja lub tabulator, a separatorem wierszy znak nowej linii. Jeżeli korzystamy z innych separatorów, to można to wskazać w opcjach. Fragment przykładowego pliku pokazano na listingu 12.5.

Listing 12.5. Przykład pliku z danymi dla funkcji **\addplot**

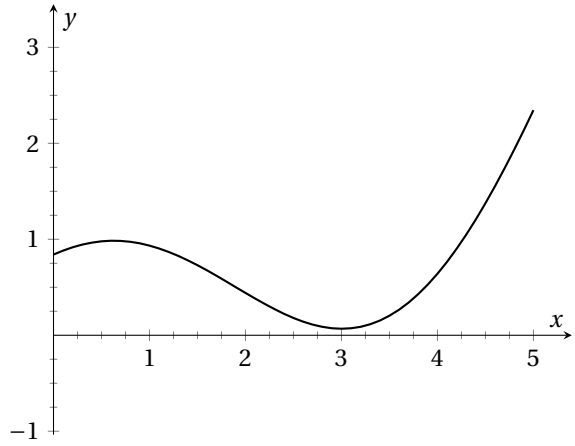
```
  x    y1     y2    y3
0.0 0.841 -1.000 -1.000
0.1 0.882 -0.985 -1.015
0.2 0.917 -0.940 -1.057
0.3 0.945 -0.865 -1.126
...
```

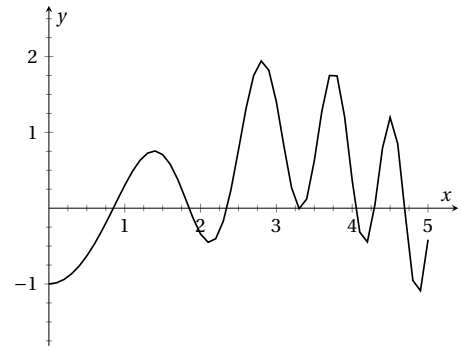
Przykład wykresu uzyskanego na podstawie takich danych pokazano na listingu 12.6. Wykres kreślimy na podstawie kolumn `x` i `y1`. Gdyby plik nie zawierał nazw kolumn, można by użyć ich numerów (licząc od 0), np. `table[x index=0, y index=1]`. Krzywa widoczna na rysunku wygląda bardzo elegancko, choć nie zastosowaliśmy żadnej opcji jej wygładzania. Wynika to wyłącznie z danych z kolumny `y1`. Domyślny wykres jest kreślony jako `sharp plot`, co oznacza, że kolejne punkty są po prostu łączone odcinkami. W przypadku, gdy wartości funkcji charakteryzują się nagłą zmiennością, wykres może miejscami przypominać łamaną. Widać to na rys. 12.2a, gdzie mamy wykres dla kolumny `y2`. Jeżeli chcemy uzyskać krzywą gładką, która interpoluje wskazane punkty, to należy dodać opcję `smooth` w poleceniu **\addplot**. Wykres uzyskany z tych samych danych, ale z tą opcją pokazano na rys. 12.2b. Jeszcze inną możliwość oferuje opcja `const plot`. W tym przypadku wykres będzie się składał wyłącznie z odcinków równoległych do osi układu współrzędnych.

Listing 12.6. Kreślenie wykresu funkcji na podstawie danych tabelarycznych

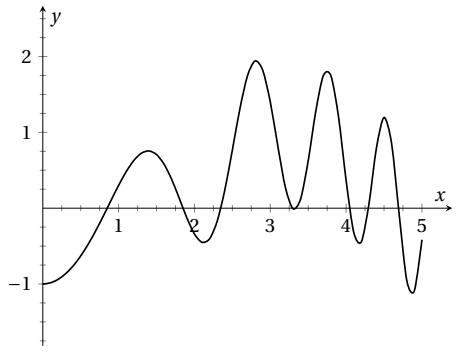
```
\begin{tikzpicture}
\begin{axis}[
  axis lines=center,
  axis equal,
  xtick distance=1,
  ytick distance=1,
  minor tick num=3,
  xlabel=$x$,
  ylabel=$y$,
  xmax=5.4,
```

```
]
\addplot[thick] table[x=x, y=y1] {dane.txt};
\end{axis}
\end{tikzpicture}
```





(a) Opcja `sharp plot`

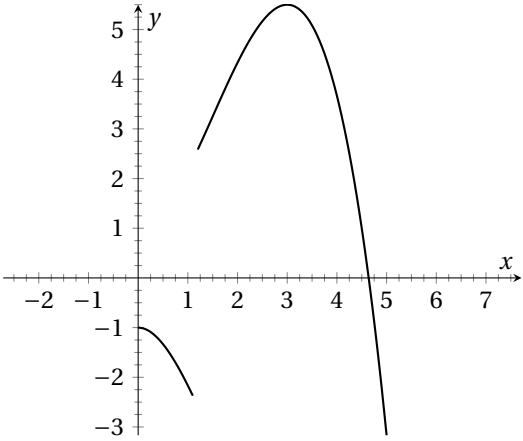


(b) Opcja `smooth`

Rys. 12.2. Różnica między opcjami `sharp plot` i `smooth`

Jeżeli chcemy wskazać nieciągłość funkcji, wystarczy zostawić w danych wejściowych pustą linię. W przypadku kreślenia w dwóch wymiarach domyślnie stosowana jest opcja `empty line=jump`, która oznacza przeskoczenie do kolejnego punktu, bez łączenia punktów, które oddziela pusta linia (wartość `none` spowoduje połączenie punktów). Efekt przeskoku pokazano na rys. 12.3.

Dane tabelaryczne mogą zawierać więcej informacji niż tylko współrzędne punktów do wykreślenia wykresu. Przy analizie danych często potrzebujemy wyróżnić punkty różnymi kolorami lub symbolami, jeżeli należą do różnych klas. W kolejnych przykładach wykorzystamy dane, których fragment pokazano



Rys. 12.3. Kreślenie funkcji z nieciągłością

na listingu 12.7. Kompletny plik zawiera informacje o 24 punktach, które są przypisane do 4 kategorii (od a do d). Nazwy kategorii nie muszą być ograniczone do pojedynczych znaków. Dodatkowe informacje o punktach (liczbowe lub tekstowe) będziemy określali terminem metadane.

Listing 12.7. Dane tabelaryczne z kategoriami punktów

```
x   y  z    k
1   1  2.0  a
3   1  2.4  a
2   2  2.8  a
2   3  2.0  a
4   2  2.5  a
8   3  3.1  b
8   5  3.0  b
9   4  3.9  b
% ...
```

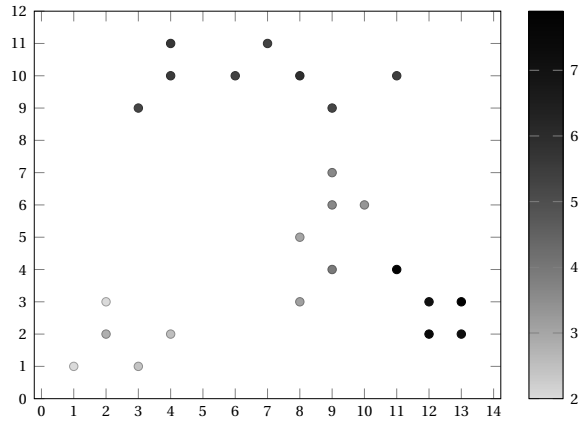
Jeżeli chcemy korzystać z kolorów na wykresie, powinniśmy wskazać mapę kolorów, która ma zostać użyta. Wybór jest realizowany poprzez opcję `colormap name` środowiska **`axis`**, np.: `colormap name=viridis`. Pakiet `pgfplots` zawiera całą gamę predefiniowanych map. Istnieje możliwość zdefiniowania własnej mapy, dodając opcję `colormap` do opcji środowiska **`axis`** (opcja poza definicją od razu aktywuje mapę). Mapa kolorów jest sekwencją kolorów z ciągłym przejściem między wskazanymi kolorami. Jeżeli podamy tylko dwa kolory, to minimalna wartość jest przypisywana pierwszemu, maksymalna drugiemu, a pozostałe wartości są liniowo skalowane pomiędzy minimum i maksimum. Przykładowa definicja może mieć postać `colormap={mygray}{color=(gray!20)`

color=(black)}. Pierwszy parametr zawiera nazwę naszej mapy, a drugi sekwencję kolorów. Ta mapa została przygotowana na potrzeby monochromatycznego wydruku. Jeżeli w drugim nawiasie klamrowym podamy trzy kolory, to będą użyte na początku, w środku i na końcu z ciągłym przejściem między nimi itd. Pakiet pozwala definiować również mapy z nierównomiernym rozkładem kolorów, tj. np. środkowy kolor nie jest w połowie, ale w 1/3 skali. Przykłady takich definicji można znaleźć w dokumentacji pakietu.

Na listingu 12.8 przedstawiono przykład wykresu punktowego. Użyto w nim własnej mapy kolorów (linia 7). Dodatkowo wyświetlony został pasek kolorów (linia 8), który pokazuje, jak wartości zostały zmapowane na kolory. Dostępna jest również opcja `colorbar horizontal`, jeżeli pasek chcemy umieścić poziomo. Opcja `only marks` (linia 9) odpowiada za wygenerowanie wykresu punktowego (wykres zawiera tylko oznaczenia punktów bez łączenia ich). W linii 10 nieznacznie zmniejszone zostały domyśle rozmiary znaczników. Jeżeli pominiemy kolejne dwie opcje (linie 11–12), to wszystkie punkty na wykresie zostaną wyświetlone w tym samym kolorze. Opcja `scatter` aktywuje wykres punktowy z indywidualnym doborem kolorów dla punktów, zaś opcja `point meta=explicit` wskazuje, że kolory będą przypisywane na podstawie jawnie wskazanych danych numerycznych. Bez tej opcji kolorowanie odbywałoby się na podstawie wartości zmiennej `y`. W poleceniu **`\addplot`** (linia 14) naszą tabelę danych wczytujemy z pliku `punkty.txt`. Poza określeniem kolumn dla `x` i `y` wskazana została kolumna metadanych (u nas `z`) i to wartości z tej kolumny decydują o kolorach poszczególnych punktów.

Listing 12.8. Wykres punktowy kreślony z zastosowaniem opcji `meta`

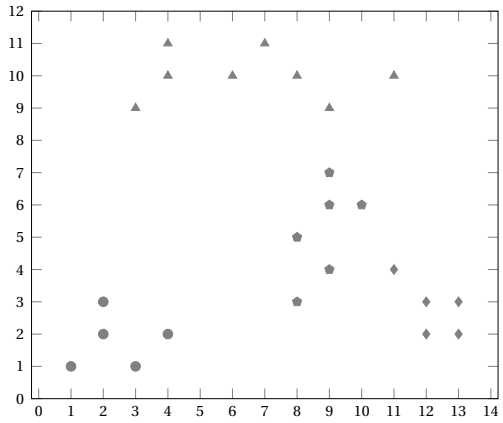
```
1  \begin{tikzpicture}
2  \begin{axis}[
3    axis equal,
4    xtick distance=1,
5    ytick distance=1,
6    tick label style={font=\scriptsize},
7    colormap={mygray}{color=(gray!20) color=(black)},
8    colorbar,
9    only marks,
10   mark size=1.75pt,
11   scatter,
12   point meta=explicit,
13 ]
14 \addplot table[x=x, y=y, meta=z]{punkty.txt};
15 \end{axis}
16 \end{tikzpicture}
```



Informacja meta dla punktów może mieć również charakter symboliczny, tak jak w kolumnie `k` (kategoria). W takim przypadku wartością dla opcji `point meta` jest `explicit symbolic`, a poszczególnym kategoriom możemy przypisać różne style w ramach opcji `scatter/classes`. Przykład taki pokazano na listingu 12.9. W tym przypadku style różnią się kształtem markera punktu, ale można zdefiniować również różne kolory. Po definicji ostatniego stylu (linia 14) nie można wstawić przecinka, bo wygeneruje to błąd kompilacji.

Listing 12.9. Wykres punktowy uwzględniający kategorie punktów

```
1  \begin{tikzpicture}
2  \begin{axis}[
3    axis equal,
4    xtick distance=1,
5    ytick distance=1,
6    tick label style={font=\scriptsize},
7    scatter,
8    only marks,
9    point meta=explicit symbolic,
10   scatter/classes={
11     a={mark=*, gray},
12     b={mark=pentagon*, gray},
13     c={mark=diamond*, gray},
14     d={mark=triangle*, gray}
15   },
16 ]
17 \addplot table[x=x, y=y, meta=k]{punkty.txt};
18 \end{axis}
19 \end{tikzpicture}
```



12.3 Wykresy słupkowe

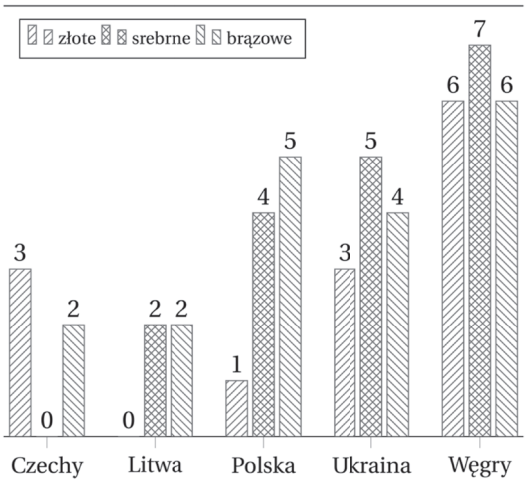
Proste wykresy słupkowe uzyskujemy, stosując opcje `xbar` (ułożenie poziome) i `ybar` (ułożenie pionowe). Jeżeli użyjemy funkcji **`\addplot`** wielokrotnie, to słupki automatycznie będą układane obok siebie. Przykład wykresu pionowego pokazano na listingu 12.10. W przykładzie wykorzystano możliwość zdefiniowania tabeli (linie 2–9), która następnie jest wielokrotnie używana. Zamiast podawać dane bezpośrednio w klamrach, można wskazać nazwę pliku, z którego mają zostać wczytane. Polecenie **`\pgfplotstableread`** wymaga dodania pakietu `pgfplotstable`. W tabeli mamy kolumnę z opisami (indeksami) wierszy i trzy kolumny numeryczne zawierające informacje o liczbie medali zdobytych podczas igrzysk olimpijskich w 2024 r. Tabela jest później dostępna pod nazwą **`\mydata`**.

```
\usepackage{pgfplots}
\usepackage{pgfplotstable}
\pgfplotsset{compat=1.18}
\usetikzlibrary{patterns.meta}
```

Listing 12.10. Pionowy wykres słupkowy

```
1 \begin{tikzpicture}
2 \pgfplotstableread{
3   kraj    złoto srebro brąz
4   Czechy  3     0      2
5   Litwa   0     2      2
6   Polska  1     4      5
7   Ukraina 3     5      4
```

```
  Węgry   6     7     6
}\mydata

\begin{axis}[
  ybar,
  axis x line=box,
  axis y line=none,
  xtick pos=bottom,
  ymin=0,
  yticklabel=\empty,
  xtick=data,
  xticklabels from table={\mydata}{kraj},
  xticklabel style={font=\small},
  nodes near coords,
  bar width=8pt,
  legend entries={złote, srebrne, brązowe},
  legend pos=north west,
  legend columns=3,
  legend style={font=\scriptsize},
]
\addplot[draw=gray, pattern={Lines[angle=45,distance=2pt]},
  pattern color=gray] table[x expr=\coordindex, y=złoto] {\mydata};
\addplot[draw=gray, pattern={Hatch[angle=45,distance=2pt]},
  pattern color=gray] table[x expr=\coordindex, y=srebro] {\mydata};
\addplot[draw=gray, pattern={Lines[angle=-45,distance=2pt]},
  pattern color=gray] table[x expr=\coordindex, y=brąz] {\mydata};
\end{axis}
\end{tikzpicture}
```



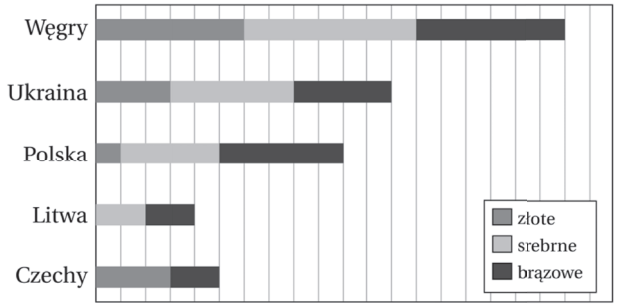
Opcja `ybar` (linia 12) wskazuje, że dodawane wykresy będą pionowymi wykresami słupkowymi. W liniach 13–15 ustawiamy wygląd układu współrzędnych i zostawiamy znaczniki osi x tylko na dole (opcja `xtick pos` ma sens tylko dla wersji osi typu `box`). Przesunięcie minimalnej wartości dla y na zero (linia 16) powoduje, że słupki stoją na osi x. Znaczniki na osi x będą ustawione po jednym dla każdego wiersza w danych (linia 18), a etykiety zostaną pobrane z kolumny `kraj` z tabeli **`\mydata`**. Opcja z linii 21 powoduje wypisanie wartości zmiennej y nad słupkami. Kolejna opcja (linia 22) ustawia szerokość słupków, a linie 23–26 zawierają ustawienia dla legendy. W liniach 28–33 mamy dodane trzy wykresy słupkowe. Opcje polecenia **`\addplot`** odpowiadają kolejno za kolor rysowania brzegu słupka, wzór wypełnienia i kolor wzoru wypełnienia. Wykresy tworzymy z użyciem konstrukcji **`\addplot`**`[] table[]{}`. W każdym przypadku sięgamy do zdefiniowanej wcześniej tabeli. Opcja `x expr=`**`\coordindex`** oznacza, że jako wartości zmiennej x brane będą indeksy (numery wierszy). Jako wartości dla y używane są kolejne kolumny numeryczne z tabeli **`\mydata`**.

Jeżeli jako typ wykresu przyjmiemy `xbar stacked` lub `ybar stacked`, to słupki nie będą rysowane obok, ale będą składane ze sobą w układzie poziomym lub pionowym. Przykład takiego wykresu pokazano na listingu 12.11. Do wygenerowania wykresu użyto tych samych danych co poprzednio.

Listing 12.11. Poziomy wykres słupkowy

```
1  \begin{tikzpicture}
2  \begin{axis}[
3    xbar stacked,
4    xtick distance=1,
5    xmajorgrids=true,
6    major tick length=0pt,
7    xmin=0,
8    xticklabel=\empty,
9    yticklabel style={font=\small},
10   bar width=8pt,
11   ytick=data,
12   yticklabels from table={\mydata}{kraj},
13   legend entries={złote, srebrne, brązowe},
14   legend pos=south east,
15   legend cell align=left,
16   legend style={font=\scriptsize},
17   width=8.4cm,
18   height=5.5cm,
19 ]
20 \addplot[draw=none, fill=black!40] table[x=złoto, y expr=\coordindex]
21   {\mydata};
```

```
\addplot[draw=none, fill=black!20] table[x=srebro, y expr=\coordindex]
  {\mydata};
\addplot[draw=none, fill=black!70] table[x=brąz, y expr=\coordindex]
  {\mydata};
\end{axis}
\end{tikzpicture}
```



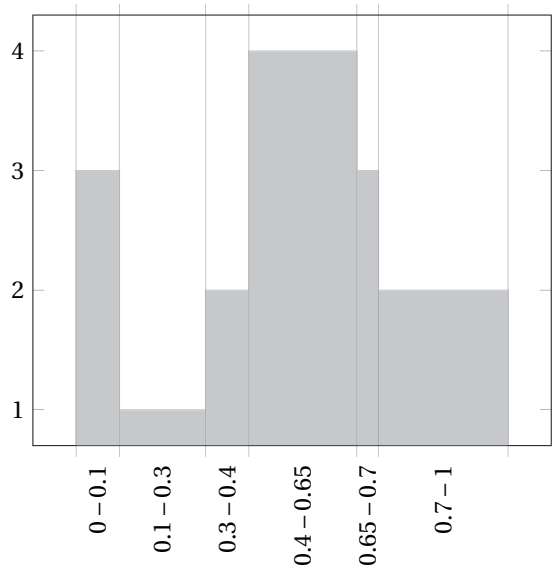
W przedstawionym przykładzie wybrano układ poziomy (linia 3). W porównaniu z poprzednim przykładem wiele użytych opcji zmieniło wyłącznie `x` na `y` lub odwrotnie. Pozostał domyślny układ współrzędnych. Włączone zostały pionowe linie siatki (linia 5) i w połączeniu z nimi ustawione na zero długości znaczników (linia 6), aby nie było różnic między siatką i znacznikami (alternatywnie można ustawić taki sam styl dla linii siatki i znaczników). Zmienione zostały również wymiary rysunku – mniejsza wysokość spowodowała, że słupki ułożone są bliżej siebie. Do wizualizacji słupków skorzystaliśmy tym razem z koloru wypełnienia i zmieniła się rola zmiennych `x` i `y`.

Kolejna wersja diagramów słupkowych kryje się pod opcjami `xbar interval` i `ybar interval`. W wykresach tych słupki przylegają do siebie. W przypadku opcji `xbar interval` kolejne dwie wartości zmiennej `x` są używane jako granice słupka, a różnica między nimi definiuje jego szerokość. W drugiej opcji mamy po prostu zmienioną rolę zmiennych. Przykład pokazano na listingu 12.12. Format etykiet w postaci przedziałów definiuje opcja `xticklabel interval boundaries` (linia 16). Dodatkowo obracamy etykiety o 90° (linia 17) i wskazujemy kotwicę, według której mają być one pozycjonowane. Brak opcji dotyczących zmiennych (linia 19) oznacza, że pierwsza kolumna jest traktowana jako wartości dla `x`, a druga dla `y`.

Listing 12.12. Wykres słupkowy typu `xbar interval`

```
\begin{tikzpicture}
\pgfplotstableread{
  x    y
  0.0  3
```

```
  0.1  1
  0.3  2
  0.4  4
  0.65 3
  0.7  2
  1    4
}\mydata

\begin{axis}[
  ybar interval,
  xtick=data,
  xticklabel interval boundaries,
  xticklabel style={rotate=90, anchor=east},
]
\addplot[gray, fill=gray, opacity=0.5] table {\mydata};
\end{axis}
\end{tikzpicture}
```



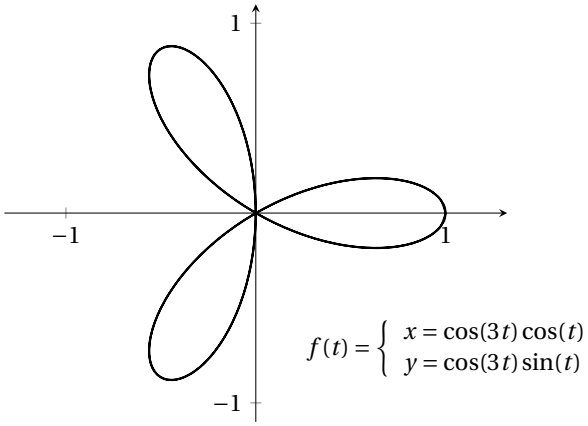
12.4 Rysowanie poza środowiskiem axis

Środowisko **axis** jest umieszczone wewnątrz środowiska **tikzpicture**. Poza dodawaniem wykresów daje to możliwość rysowania typowego dla pakietu tikz. Rysować możemy zarówno wewnątrz środowiska **axis**, jak i poza nim. Kluczowa dla korzystania z tej możliwości jest świadomość, że układ współrzędnych

używany wewnątrz środowiska **axis** nie pokrywa się z układem w środowisku **tikzpicture**. Jeżeli rysujemy wewnątrz środowiska **axis** i chcemy skorzystać ze współrzędnych zgodnych z osiami układu współrzędnych, to zapisujemy je jako `(axis cs:x,y)`. Pozwala to precyzyjnie odwoływać się na przykład do punktów wykresu. Użycie współrzędnych w postaci `(x,y)` oznacza współrzędne w środowisku **tikzpicture**.

Listing 12.13. Kreślenie wykresu funkcji parametrycznej

```
\begin{tikzpicture}
\begin{axis}[
  axis lines=center,
  axis equal,
  domain=0:2*pi,
  ymax=1.1,
  ymin=-1.1,
  xtick={-1,1},
  ytick={-1,1},
  xmax=1.2,
  xmin=-1.2
]
\addplot[thick, samples=80, smooth]
  ({cos(3*deg(x))*cos(deg(x))}, {cos(3*deg(x))*sin(deg(x))});
\end{axis}
\node[right] at (4,1) {
$f(t)=\left\{
\begin{array}{l}
x=\cos(3t)\cos(t)\\
y=\cos(3t)\sin(t)
\end{array}\right.$};
\end{tikzpicture}
```



Rozważmy przykład pokazany na listingu 12.13. Poza utworzeniem wykresu dodaliśmy opis funkcji, ale nie jako legendę, tylko jako węzeł w środowisku **tikzpicture**. Gdybyśmy chcieli to zrobić wewnątrz środowiska **axis**, to możemy użyć współrzędnych `(axis cs:0.22,-0.71)` – odpowiada to mniej więcej współrzędnym `(4,1)` użytym na listingu. Jest jednak ważna różnica między oboma rozwiązaniami. W rozwiązaniu z listingu 12.13 wzór wychodzi poza okno wykresu i nie jest to żaden problem. Jeżeli umieścimy go w środku środowiska **axis**, stosując współrzędne `(axis cs:0.22,-0.71)`, to zostanie brzydko przycięty do okna wykresu.

12.5 Zaznaczanie obszarów między krzywymi

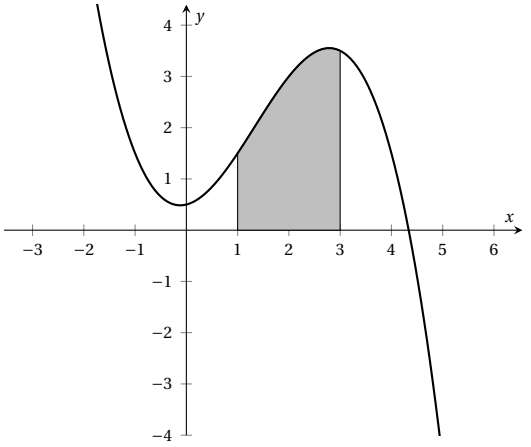
Jak widzieliśmy w przykładzie z listingu 12.2, możliwe jest nazwanie krzywych i wyznaczenie wszystkich ich punktów przecięcia. Biblioteka `fillbetween` pozwala na zaznaczenie obszaru pomiędzy nazwanymi krzywymi. Możemy zaznaczać cały obszar lub ograniczyć się do wybranego fragmentu. Jest to realizowane przez specjalną wersję polecenia **\addplot**, która potrzebuje na wejściu dwóch nazwanych ścieżek. Pierwszy przykład pokazano na listingu 12.14.

```
\usepgfplotslibrary{fillbetween}
\usetikzlibrary{patterns.meta}
```

Listing 12.14. Zaznaczanie obszaru pod krzywą

```
\begin{tikzpicture}
\begin{axis}[
  axis lines=center,
  axis equal,
  xlabel=$x$,
  ylabel=$y$,
  domain=-3:6,
  ymin=-4,
  ymax=4.4,
  xmin=-3.4,
  xmax=6.4,
  label style={font=\scriptsize},
  tick label style={font=\scriptsize},
  xtick distance=1,
  ytick distance=1,
  fun/.style={thick, samples=80, smooth}
]

```

```
19 \addplot[name path=f1, fun] {-(x^3)/4+x^2+x/4+0.5};
20 \path[name path=ox] (0,0) -- (6,0);
21 \addplot[draw=black, fill=gray!50] fill between
22   [of=f1 and ox, soft clip={domain=1:3}];
23 \end{axis}
24 \end{tikzpicture}
```

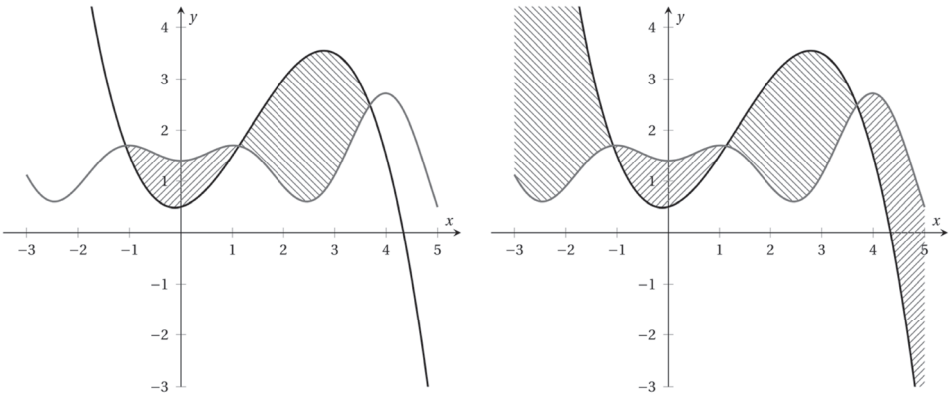


W linii 19 dodajemy wykres wielomianu trzeciego stopnia z przypisaną nazwą `f1`. Obszar jest zaznaczany pomiędzy dwoma ścieżkami. Ponieważ chcemy zaznaczyć obszar między naszym wielomianem a osią x, potrzebujemy ścieżki pokrywającej się z osią. Definiujemy ją za pomocą polecenia **`\path`** (linia 20). Użycie tego polecenia definiuje nam ścieżkę `ox`, ale niczego nie rysuje. W końcu w linii 21 dodajemy zamalowany obszar między naszymi dwoma ścieżkami, ale ograniczony do przedziału $[1,3]$ (opcja `soft clip`). Polecenia **`\addplot`** `fill between` możemy użyć wielokrotnie, zaznaczając różne interesujące nas fragmenty.

W rozważanym przykładzie przecięcia między ścieżką `f1` i `os` tworzą dwa segmenty – pierwszy do miejsca zerowego funkcji `f1`, a drugi po miejscu zerowym. Gdyby miejsc zerowych było więcej, to również mielibyśmy więcej segmentów między wykresem funkcji a osią x. Podobnie jest, gdy analizujemy obszary między dwiema krzywymi, jak w przykładzie z listingu 12.15. Tutaj pierwszy segment (o numerze 0) obejmuje obszar między krzywymi, do pierwszego punktu przecięcia, kolejny segment od pierwszego do drugiego punktu przecięcia itd. W tym przykładzie mamy cztery takie segmenty. Widać to dokładnie na prawym rysunku pod listingiem 12.15. Możemy indywidualnie zaznaczyć każdy z tych segmentów odpowiednio redefiniując style, tak jak pokazano to na tym listingu.

Listing 12.15. Zaznaczanie obszaru między krzywymi

```
1  \begin{tikzpicture}
2  \begin{axis}[
3    % ustawienia podobne jak poprzednio, zmienione: domain, ymin, xmax
4  ]
5  \addplot[name path=f1, fun] {-(x^3)/4+x^2+x/4+0.5};
6  \addplot[name path=f2, fun, gray] {sin(deg(x))*cos(deg(x))*(2*x)/3+1.4};
7  \addplot[fill=none] fill between [of=f1 and f2,
8    split,
9    every segment no 1/.style={pattern={Lines[angle=45,distance=2pt]},
10     pattern color=gray},
11   every segment no 2/.style={pattern={Lines[angle=-45,distance=2pt]},
12     pattern color=gray},
13 ];
14 \end{axis}
15 \end{tikzpicture}
```



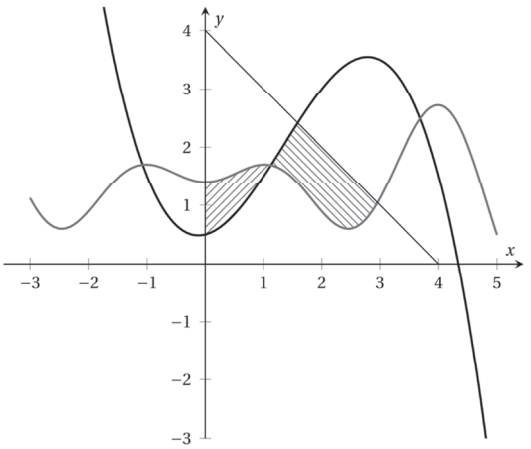
Przy zaznaczaniu obszaru między krzywymi `f1` i `f2` zastosowanie opcji `split` w połączeniu z różnymi stylami pozwala na indywidualne oznaczenie każdego z segmentów. Opcja `split` powoduje wyznaczenie wszystkich punktów przecięcia krzywych i rysowanie każdego uzyskanego segmentu indywidualnie. W naszym przykładzie ustaliliśmy domyślne wypełnienie segmentów na `none` (linia 7), a następnie zmieniliśmy ustawienia dla segmentu 1 i 2 (linie 9–12). Zamiast opcji `every segment no x` można użyć `every odd segment` i `every even segment`, co pozwala łatwo naprzemiennie pozaznaczać wszystkie segmenty (zobacz prawy rysunek).

Opcji `soft clip` możemy użyć nie tylko do ograniczenia się do wskazanej dziedziny, ale również do zdefiniowanego obszaru. Załóżmy, że interesują nas dwa ostatnio analizowane segmenty, ale dodatkowo ograniczone do

powierzchni trójkąta o wierzchołkach w punktach (0,0), (0,4) i (4,0). Rozwiązanie takie przedstawiono na listingu 12.16.

Listing 12.16. Zaznaczanie obszaru ograniczonego dodatkowo obszarem zamkniętym

```
1  \begin{tikzpicture}
2  \begin{axis}[
3    % ustawienia takie jak poprzednio
4  ]
5  \addplot[name path=f1, fun] {-(x^3)/4+x^2+x/4+0.5};
6  \addplot[name path=f2, fun, gray]
7    {sin(deg(x))*cos(deg(x))*(2*x)/3+1.4};
8  \draw[name path=tr] (0,0) -- (0,4) -- (4,0) -- cycle;
9  \addplot[fill=none] fill between [of=f1 and f2,
10   soft clip={tr},
11   split,
12   every segment no 0/.style={pattern={Lines[angle=45,distance=2pt]},
13     pattern color=gray},
14   every segment no 1/.style={pattern={Lines[angle=-45,distance=2pt]},
15     pattern color=gray},
16 ];
17 \end{axis}
18 \end{tikzpicture}
```



W linii 8 zdefiniowano i narysowano zamknięta ścieżkę określającą wspomniany trójkąt i użyto jej w opcji `soft clip` (linia 10). Efekt widoczny jest na rysunku pod listingiem. Warto zwrócić uwagę na numerację segmentów. Ponieważ rozważania ograniczamy tylko do trójkąta, to mamy tylko dwa segmenty o numerach 0 i 1.

12.6 Kreślenie wykresów w układzie trójwymiarowym

Za generowanie wykresu w trzech wymiarach odpowiada polecenie **\addplot3**. Polecenie to akceptuje analogiczne dane wejściowe jak **\addplot**, ale w każdym przypadku potrzebujemy określić wartości dla trzeciej zmiennej z. Polecenie realizuje kreślenie w postaci linii, jeżeli mamy odwzorowanie $\mathbb{R} \to \mathbb{R}^3$ i w postaci siatki dla $\mathbb{R}^2 \to \mathbb{R}^3$. To, z którym przypadkiem będziemy mieć do czynienia, zależy od tego, jak przygotowano dane wejściowe i jakie opcje przekażemy do środowiska **axis** lub polecenia **\addplot**.

Kompilacja kodu z grafiką trójwymiarową może zająć dłuższą chwilę, w szczególności, jeśli dokument zawiera kilka takich wykresów. Rysunki najlepiej przygotować w oddzielnych plikach i załączać gotową grafikę do dokumentu.

W przypadku polecenia **\addplot3** `expression {}` (słowo `expression` można pominąć) mamy wizualizację funkcji $f(x,y)$, gdzie $f\colon [x_1,x_2] \times [y_1,y_2] \to \mathbb{R}$. Przedziały $[x_1,x_2]$ i $[y_1,y_2]$ zadajemy za pomocą opcji `domain` i `domain y`, a gęstość próbkowania za pomocą opcji `samples` i `samples y`. Jeżeli pominiemy `domain y`, to przyjmowane jest $[x_1,x_2] = [y_1,y_2]$, a jeśli pominiemy `samples y`, to zakłada się takie samo próbkowanie dla y jak dla x. Przy wersji **\addplot3** `expression {}` domyślnie zakłada się, że w wyniku próbkowania powstaje macierz, więc mamy funkcję $\mathbb{R}^2 \to \mathbb{R}^3$. Jeżeli chcemy uzyskać odwzorowanie $\mathbb{R} \to \mathbb{R}^3$, to należy użyć opcji `samples y=1` lub określić `domain y` w formie przedziału o zerowej długości, np. `domain y=2:2`. W takim przypadku zakłada się, że funkcja nie zależy od y (zmienna nie może być użyta w wyrażeniu matematycznym), a uzyskany wykres jest krzywą w przestrzeni $\mathbb{R}^3$.

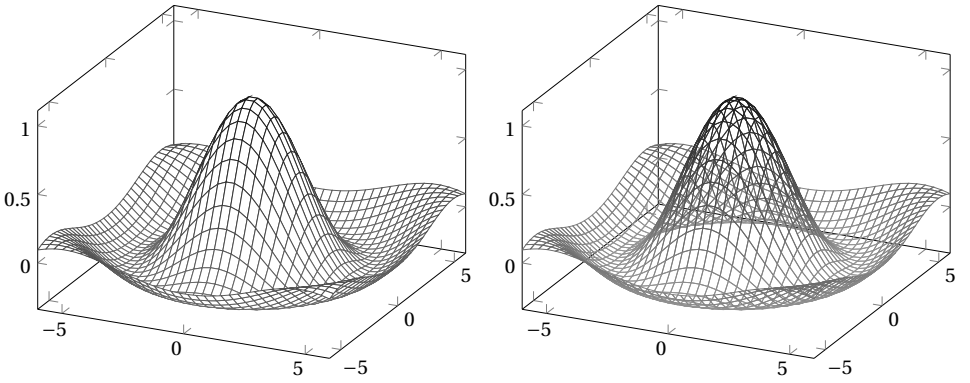
Na listingu 12.17 pokazano przykład zastosowania polecenia **\addplot3** `expression {}` do wykreślenia wykresu funkcji

$$f(x,y) = \frac{\sin\left(\sqrt{x^2+y^2}\right)}{\sqrt{x^2+y^2}}. \tag{12.1}$$

Powierzchnia jest reprezentowana za pomocą siatki, której gęstość zależy od parametrów próbkowania (parametr ten istotnie wpływa również na czas kompilacji). Jeżeli użyjemy opcji `surf`, to oka siatki zostaną zamalowane (w naszym przypadku na biało ze względu na opcję `fill` w linii 7). Rysunek po lewej stronie wygenerowano z widocznymi na listingu ustawieniami. Jeżeli użyjemy opcji `mesh` (w miejsce `surf`), to oka siatki nie są wypełniane, tak jak to pokazano na rysunku po prawej stronie.

Listing 12.17. Kreślenie wykresu powierzchni

```
\begin{tikzpicture}
\begin{axis}[
  domain=-6:6,
  samples=36,
  colormap={mygray}{color=(black!30) color=(black!80)},
]
\addplot3[surf, fill=white]
  {(sin(deg(sqrt(x^2+y^2))))/(sqrt(x^2+y^2))};
\end{axis}
\end{tikzpicture}
```

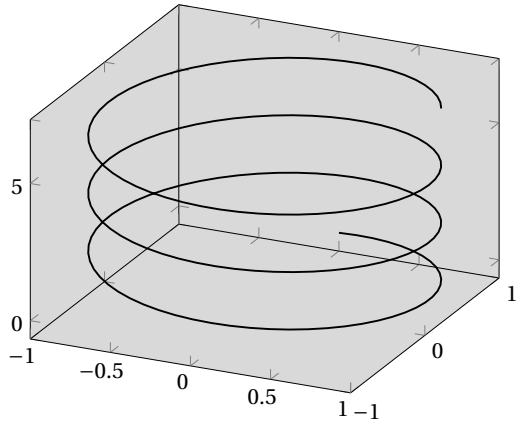


Podobnie jak w przypadku krzywych na płaszczyźnie, krzywe i powierzchnie w przestrzeni możemy zadawać parametrycznie. Wzór postaci `{f(x,y)}` możemy równoważnie zapisać jako `(x,y,f(x,y))`. Dla powierzchni parametrycznych każdy z tych trzech elementów będzie funkcją dwóch zmiennych – `({x(u,v)},{y(u,v)},{z(u,v)})`, zaś dla krzywej parametrycznej jednej zmiennej – `({x(u)},{y(u)},{z(u)})`, przy czym, o ile nie zmienimy nazwy, to domyślnymi zmiennymi są `x` i `y`. Możemy wprowadzić własne nazwy dla zmiennych za pomocą opcji `variable` i `variable y`. Dodatkowe nawiasy klamrowe są zbędne, jeśli w wyrażeniu nie używamy nawiasów okrągłych. Na listingu 12.18 pokazano przykład kreślenia krzywej parametrycznej w przestrzeni $\mathbb{R}^3$. Opcja `axis background/.style` pozwala na określenie stylu tła dla wykresu (można jej używać również dla wykresów dwuwymiarowych).

Listing 12.18. Krzywa parametryczna w przestrzeni trójwymiarowej – linia śrubowa

```
\begin{tikzpicture}
\begin{axis}[
  domain=0:20,
  samples=180,
```

```
  samples y=1,
  axis background/.style={fill=gray!30},
]
\addplot3[black, thick] ({sin(deg(x))}, {cos(deg(x))}, x/3);
\end{axis}
\end{tikzpicture}
```



W tabeli 12.3 zebrane zostały najważniejsze opcje związane z dopasowaniem wyglądu wykresów trójwymiarowych. Tabela ta nie zastępuje, ale raczej uzupełnia opcje z tabeli 12.2. Większość opcji z tabeli 12.2 może być stosowana na wykresach trójwymiarowych. Dodatkowo wiele opcji ma jeszcze wersje ze zmienną z, np. `zlabel`, `zlabel style` itd. Opcje z tabeli 12.3, które do tej pory nie były prezentowane, są omówione w kolejnych przykładach.

W przypadku polecenia `\addplot3 table {}` zawartość tabeli możemy podać bezpośrednio lub wczytać ją z pliku. Jeżeli chcemy, aby dane z tabeli zostały potraktowane jako macierz do wykreślenia powierzchni, to musimy wskazać, jak mają być interpretowane. Jeżeli po prostu podamy tabelę punktów, na przykład taką jak na listingu 12.5 lub 12.7, to powstanie wykres liniowy przez połączenie punktów w kolejności ich występowania w danych.

Jedną z metod zmiany interpretacji danych tabelarycznych jest podanie co najmniej jednej z opcji `mesh/rows` lub `mesh/cols`. Powodują one odpowiednio, że tabela zostanie zinterpretowana jako macierz z podaną liczbą wierszy lub podaną liczbą kolumn. Dodatkowo opcja `mesh/ordering` pozwala dookreślić właściwe interpretowanie danych. Domyślnie jest to wartość `rowwise` – czytamy kolejne wiersze macierzy. Możemy wskazać wartość `colwise`, jeżeli dane są ułożone kolumnami. Jeżeli podamy tylko jedną z liczb określających wymiary macierzy, to system sam wyznaczy drugą na podstawie liczby punktów.

Wróćmy jeszcze na moment do samej macierzy. Załóżmy, że mamy dane tabelaryczne w trzech kolumnach `x`, `y`, `z` i dokładnie tak je chcemy rozumieć

Tabela 12.3. Wybrane opcje środowiska **axis** dla wykresów trójwymiarowych

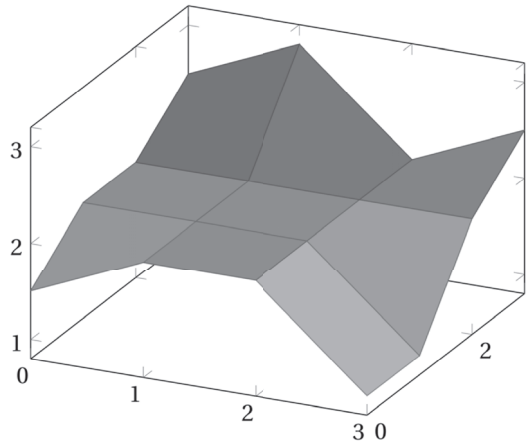
Opcja	Opis
`domain`	dziedzina (przedział próbkowania) dla zmiennej x
`domain y`	dziedzina (przedział próbkowania) dla zmiennej y
`samples`	gęstość próbkowania dla zmiennej x
`samples y`	gęstość próbkowania dla zmiennej y
`mesh/rows`	liczba wierszy macierzy będącej podstawą wyliczenia punktów powierzchni
`mesh/cols`	liczba kolumn macierzy będącej podstawą wyliczenia punktów powierzchni
`mesh/ordering`	określenie sposobu interpretacji danych tabelarycznych: `rowwise` lub `colwise`
`mesh`	wykres powierzchni w postaci siatki
`surf`	wykres powierzchni w postaci wypełnionych wielokątów
`mesh input`	określenie formatu danych wejściowych
`patch type`	typ wielokątów dla siatki danych wejściowych
`shader`	określenie sposobu zamalowania wielokątów siatki
`faceted color`	kolor rysowania siatki (przy opcji `surf`)
`axis background/.style`	styl rysowania tła
`view`	ustawienie kątów obrotu układu współrzędnych
`3d box`	ustawienie sposobu wyświetlania układu typu `box`

(pierwsza kolumna dane dla zmiennej x itd.). Te punkty wyznaczają węzły w naszej siatce. Gdybyśmy rysowali wykres punktowy, to nie ma znaczenia, jak one zostaną ułożone w macierzy, ale gdy kreślimy siatkę, to ich ułożenie w macierzy decyduje o tym, które węzły będą wspólnie tworzyć wierzchołki jednego segmentu. Nawet zmiana kolejności dwóch punktów może wpłynąć na wygląd powierzchni.

Listing 12.19. Powierzchnia wykreślona na podstawie danych tabelarycznych

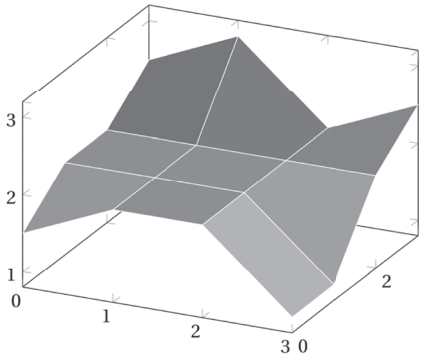
```
\begin{tikzpicture}
\begin{axis}[
  colormap={mygray}{color=(black!10) color=(black!70)},
  mesh/cols=4,
  mesh/rows=4,
]
\addplot3[surf] table {
  x  y  z
```

```
  0  0  1.5
  0  1  2
  0  2  2
  0  3  2.5
  1  0  2
  1  1  2
  1  2  2
  1  3  3
  2  0  2
  % ...
};
\end{axis}
\end{tikzpicture}
```

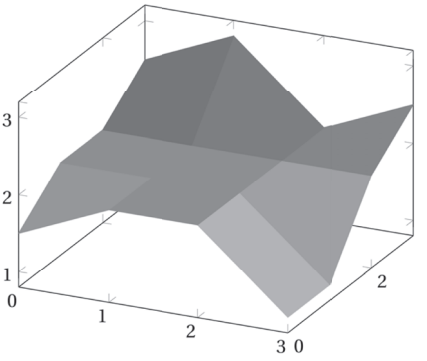


Na listingu 12.19 pokazano przykład powierzchni wykreślonej z danych tabelarycznych. Kompletne dane obejmują 16 punktów. Polecenie **\addplot3** domyślnie wykorzystuje wartości dla zmiennej *z* jako metadane do odpowiedniego kolorowania poszczególnych segmentów naszej powierzchni. Za realizację tego zadania odpowiada tzw. shader. Opcja o takiej właśnie nazwie pozwala na wybór jednej z kilku metod realizacji tego zadania. Wartość domyślna `faceted` oznacza zamalowanie wnętrza każdego segmentu jednolitym kolorem i wykreślenie siatki jednym kolorem, który można ustawić za pomocą opcji `faceted color`. Inne dostępne wartości przedstawiono na rys. 12.4.

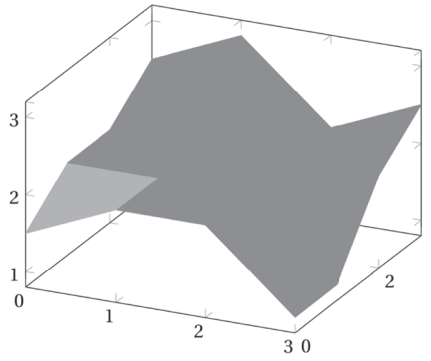
Wartość `flat mean` (rys. 12.4b) oznacza, że kolor segmentu wyznaczany jest jako średnia z wierzchołków. W przypadku `flat corner` (rys. 12.4c) to kolor pierwszego węzła decyduje o kolorze segmentu. Kolejne dwie wartości nie stosują jednolitego koloru dla segmentu, w przypadku `interp` (rys. 12.4d) mamy liniową interpolację dla dwóch trójkątów, które przybliżają segment, zaś przy `faceted interp` dodatkowo rysowane linie siatki (rys. 12.4e i 12.4f).



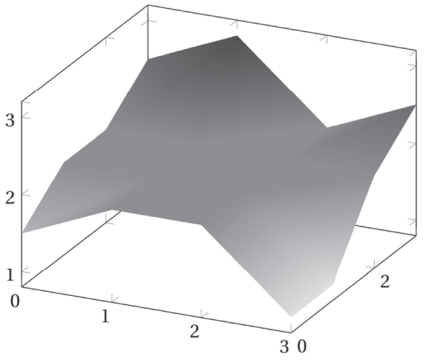
(a) `shader=feceted, faceted color=white`



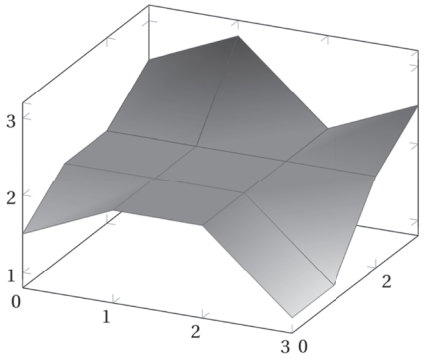
(b) `shader=flat mean`



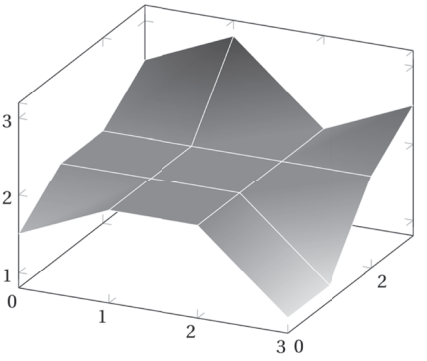
(c) `shader=flat corner`



(d) `shader=interp`



(e) `shader=faceted interp`



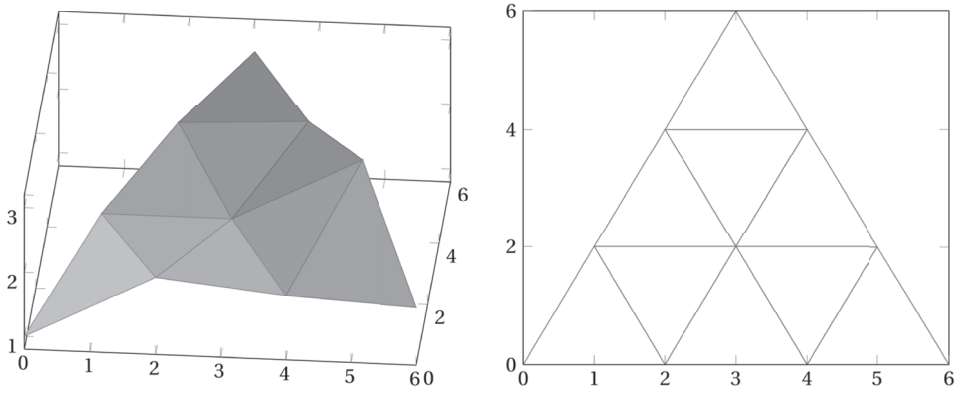
(f) `shader=faceted interp faceted color=white`

Rys. 12.4. Metody kolorowania powierzchni zależnie od ustawień opcji `shader`

W domyślnym układzie, gdy stosujemy macierzowy rozkład punktów, każdy segment jest odwzorowaniem pewnego czworokąta. Jeżeli skorzystamy z opcji `mesh input=paches`, to możemy sami określić, jaki wielokąt będzie podstawą do wygenerowania segmentu powierzchni. Określamy to za pomocą opcji `patch type`. Na listingu 12.20 pokazano przykład z siatką trójkątną.

Listing 12.20. Powierzchnia wykreślona z użyciem siatki trójkątnej

```
\begin{tikzpicture}
\begin{axis}[
  colormap={mygray}{color=(gray!10) color=(gray)},
  mesh input=patches,
  patch type=triangle,
  view={5}{50}
]
\addplot3[surf] table {
  x  y  z
  0  0  1
  2  0  2
  1  2  2

  2  0  2
  1  2  2
  3  2  2

  2  0  2
% ...
};
\end{axis}
\end{tikzpicture}
```



Rysunek po prawej stronie na listingu 12.20 pokazuje strukturę siatki, która została użyta do ułożenia punktów. Punkty dla dwóch pierwszych trójkątów są widoczne na listingu. Pełny zestaw danych zawiera 9 takich trójek. Niektóre punkty występują wielokrotnie, np. punkt znajdujący się w środku siatki $(3,2,2)$ należy aż do sześciu trójkątów i tyle razy występuje w danych wejściowych.

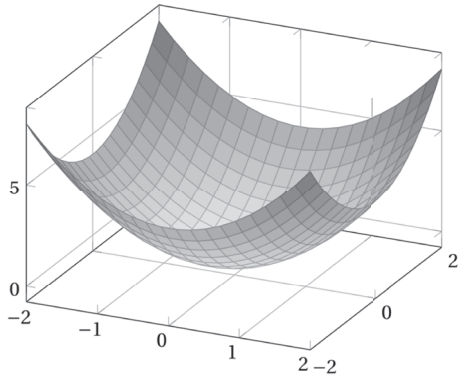
W prezentowanym przykładzie zmieniono domyślmy widok układu współrzędnych, aby lepiej widoczny był efekt użycia siatki trójkątnej. Jest to realizowane poprzez opcję `view` (linia 6). Widok bieżący definiowany jest jako złożenie dwóch obrotów, względem osi z i względem osi x (w takiej kolejności obracania). To właśnie kąty tych obrotów podajemy jako wartość opcji `view`. Punktem wyjścia do wyznaczenia bieżącego widoku jest położenie, w którym oś x jest skierowana w prawo, oś z do góry, a oś y prostopadle do płaszczyzny xz „od nas", tj. patrzymy wzdłuż osi y, im głębiej jest punkt, tym większa wartość współrzędnej y. Jeżeli zastosujemy ustawienie `view={90}{0}` (tylko obrót względem osi z), to oś y będzie skierowana w prawo, oś z do góry, a oś x prostopadle do płaszczyzny yz, ale w naszym kierunku. Domyślne ustawienie to wartości `{25}{30}`. Użyte ustawienie (linia 6) powoduje, że patrzymy bardziej z góry na naszą płaszczyznę.

Przy rysowaniu w trzech wymiarach możemy wybrać jedną z trzech wartości dla opcji `axis lines` (`box`, `center`, `none`), analogicznie jak dla wykresów dwuwymiarowych. Domyślnie jest używana wartość `box`, przy czym w trzech wymiarach nasze pudełko na wykres może być rysowane w kilku wersjach. Wybór jest realizowany poprzez opcję `3d box`. Domyślna wartość `background` powoduje, że nie są rysowane krawędzie z przodu. Wybranie wartości `complete` oznacza wykreślenie wszystkich krawędzi pudełka, a `complete*` dodatkowo wykreślenie linii siatki na przodzie pudełka (o ile linie są kreślone – opcja `grid`). Porównanie tych wersji pokazano na rys. 12.5.

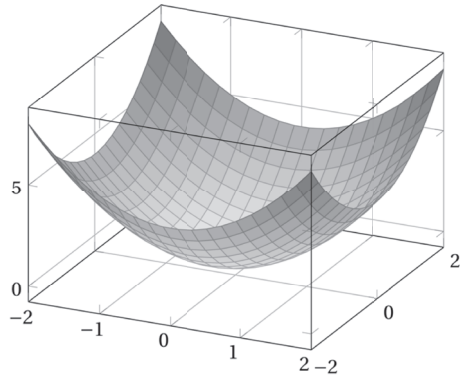
Przy wyborze opcji `axis lines=center` rysowane są trzy osie układu współrzędnych, które przecinają się w początku układu współrzędnych, o ile należy on do obszaru wykresu. Wróćmy do danych z listingu 12.19. Minimalna wartość dla współrzędnej z wynosi 1, więc przy domyślnych zakresach zmiennych początek układu nie znajdzie się w obszarze wykresu. Możemy to uzyskać, ustawiając opcję `zmin=0`. Efekt takich ustawień pokazano na listingu 12.21. Dodatkowo ustawienie częściowej przezroczystości wykresu powoduje, że w tle widzimy osie układu współrzędnych i punkt ich przecięcia.

Listing 12.21. Powierzchnia wykreślona z użyciem opcji `axis lines=center`

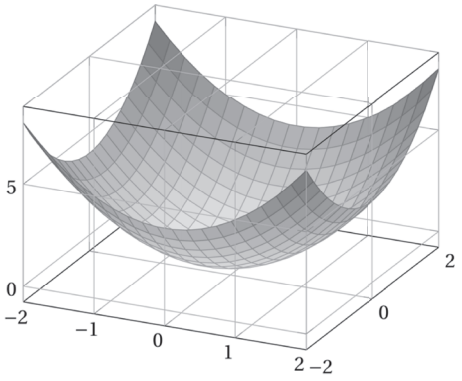
```
1 \begin{tikzpicture}
2 \begin{axis}[
3   axis lines=center,
```



(a) `3d box=background`



(b) `3d box=complete`

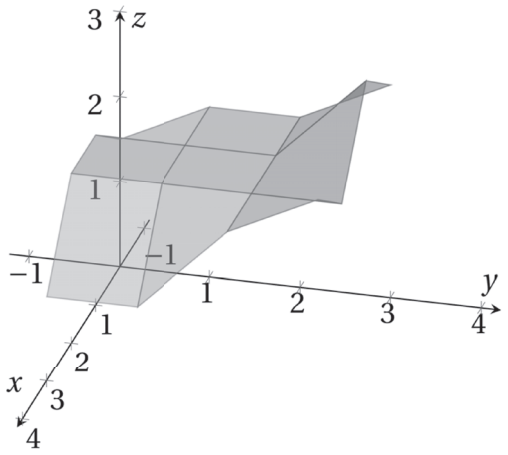


(c) `3d box=complete*`

Rys. 12.5. Wersje układu typu `box` zależnie od ustawień opcji `3d box`

```
4    axis equal,
5    xlabel=$x$,
6    ylabel=$y$,
7    zlabel=$z$,
8    xtick distance=1,
9    ytick distance=1,
10   ztick distance=1,
11   zmin=0,
12   colormap={mygray}{color=(black!10) color=(black!70)},
13   mesh/cols=4,
14   view={105}{25},
15 ]
16 \addplot3[surf, opacity=0.5] table {
17   ...
```

```
};
\end{axis}
\end{tikzpicture}
```



Rozdział 13

Przygotowanie prezentacji

Nieodłącznym elementem pracy naukowej jest prezentacja wyników swoich badań nie tylko w postaci artykułów, referatów konferencyjnych itd., ale również prezentacji. Nawet jeśli wybrałeś LaTeX-a tylko do złożenia swojej pracy dyplomowej, to pojawi się konieczność przygotowania prezentacji na jej obronę. Dlaczego warto przygotować prezentację w LaTeX-u, a nie w jakimś narzędziu z pakietów biurowych? Powodów jest kilka. Skupię się tutaj tylko na dwóch. Jeśli dotarłeś do tego miejsca, to już naprawdę dużo umiesz. Co więcej, jeśli napisałeś pracę dyplomową, artykuł naukowy itp., to masz tam gotowy kod LaTeX-a, który łatwo przenieść na slajdy. Pomyśl, że zamiast zwykłego kopiuj/wklej musiałbyś edytować w trybie WYSIWYG wszystkie wzory, tabelki itd.

13.1 Wprowadzenie do Beamera

W LaTeX-u dostępnych jest kilka klas do przygotowania prezentacji, ale zdecydowanym liderem w tym obszarze jest pakiet `beamer` [47], który zawiera klasę o tej samej nazwie. Podobnie jak z innymi klasami LaTeX-a, nasz dokument (prezentacja) będzie w formacie `pdf`, co stanowi istotną zaletę. Niezależnie od sprzętu użytego do jej wyświetlenia, nasza prezentacja powinna wyglądać perfekcyjnie (nie ma ryzyka, że coś się „rozjedzie"). Rozpoczęcie pracy w klasą `beamer` jest wyjątkowo proste. Szkielet do przygotowania naszej pierwszej prezentacji pokazano na listingu 13.1.

Listing 13.1. Szkielet prezentacji przygotowanej z użyciem klasy `beamer`

```
1 \documentclass{beamer}
2 \usepackage[utf8]{inputenc}
3 \usepackage[T1]{fontenc}
```

```
4  \usepackage[polish]{babel}
5  \usepackage{newtx}
6  \usetheme{Warsaw}
7
8  \title{Wprowadzenie do \LaTeX-a}
9  \author{Jan Techowski}
10 \institute[KIS AGH]{Katedra Informatyki Stosowanej\\AGH w Krakowie}
11 \date{07.02.2025}
12
13 \begin{document}
14 \begin{frame}
15   \titlepage
16 \end{frame}
17
18 \begin{frame}{Plan prezentacji}
19
20 \end{frame}
21 \end{document}
```

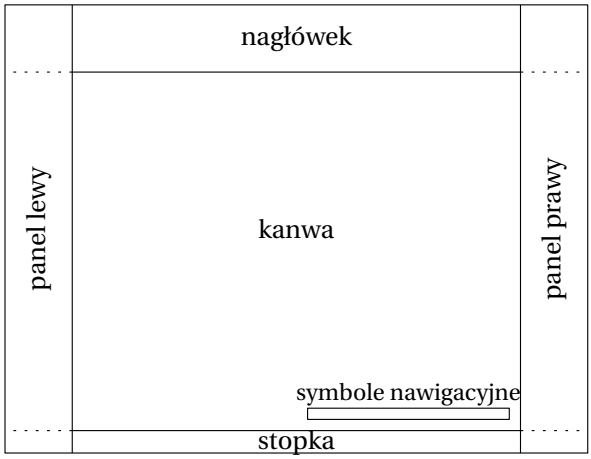
Zaczynamy od wybrania klasy `beamer` (linia 1). W liniach 2–5 dołączamy kilka podstawowych pakietów, analogicznie jak w pierwszym przykładzie z klasą `article` (zob. listing 1.1, str. 4). W linii 6 wybieramy motyw klasy `beamer`, który decyduje o wyglądzie naszych slajdów. Linie 8–11 zawierają podstawowe informacje o tytule, autorze, afiliacji i dacie prezentacji. Niektóre linie są identyczne jak w klasie `article` we wspomnianym przykładzie. W naszej prezentacji mamy dwa slajdy (linie 14–16 i 18–20). Każdy slajd tworzymy jako środowisko **`frame`**. Pierwszy slajd zawiera wyłącznie polecenie **`\titlepage`**, które generuje slajd tytułowy. Jest on tworzony na podstawie informacji dostarczonych w liniach 8–11. Drugi slajd ma tytuł „Plan prezentacji" i na razie jest pusty. Efekt kompilacji naszej prezentacji widzimy na rys. 13.1.

W pierwszym przykładzie skorzystaliśmy z motywu `Warsaw`. Beamer zawiera kilkadziesiąt gotowych motywów, z których możemy korzystać. Ich nazwy zostały wzięte od nazw miast, w których odbywały się konferencje powiązane z informatyką teoretyczną [47, str. 25], a ich przegląd można znaleźć w dokumentacji pakietu. Wystarczy zmienić nazwę motywu (spróbuj CambridgeUS), aby po ponownej kompilacji zobaczyć zupełnie odmieniony wygląd naszej prezentacji. Mnóstwo motywów dla Beamera można znaleźć w internecie. Wiele uczelni oferuje własne motywy z odpowiednią kolorystyką, logo itd. (jeżeli wybieramy się na obronę lub konferencję, warto skorzystać z motywu związanego z uczelnią). W wyszukiwarce wpisz frazę „beamer theme" i sprawdź wyniki wyszukiwania w formie graficznej. Jeżeli dalej nie znajdziesz nic dla siebie, to w podrozdziale 13.6 pokazano sposoby na szybkie przygotowanie własnego motywu.

Rys. 13.1. Prezentacja z listingu 13.1

13.2 Slajdy

Typowy slajd ma rozmiar 128 mm × 96 mm i powstaje jako wynik zastosowania środowiska **frame**. Jeżeli do środowiska dołączymy parametr umieszczony w nawiasach klamrowych, to zakłada się, że jest to tytuł slajdu. Jeżeli po nim podamy kolejny, to będzie on uznany za podtytuł, np. `\begin{frame}{To jest tytuł}{To jest podtytuł}`. Tytuł i podtytuł można również dodać za pomocą poleceń **\frametitle** i **\framesubtitle**, które umieszczamy wewnątrz środowiska **frame**. Środowisko może również przyjmować parametry opcjonalne, które zostaną omówione w dalszej części tego podrozdziału i w podrozdziale 13.3.



Rys. 13.2. Struktura slajdu

Na rys. 13.2 przedstawiono strukturę typowego slajdu w Beamerze. Panele prawy i lewy nie występują w każdym motywie, a jeśli już to tylko jeden z nich – motyw `Warsaw` w ogóle z nich nie korzysta. W nagłówku umieszczane są m.in. tytuł i podtytuł slajdu. Stopka bardzo często podzielona jest na części zawierające informacje o autorze, tytule, afiliacji lub dacie (zależnie od motywu). W przypadku motywu `Warsaw` stopka jest podzielona na dwie części – lewa zawiera nazwisko autora, a prawa tytuł prezentacji (zob. rys. 13.1). Jeżeli tytuł jest długi, to można zdefiniować jego krótką wersję do umieszczenia w stopce. Robimy to przez parametr opcjonalny analogicznie jak w przypadku polecenia **\institute** (zob. listing 13.1 linia 10). Analogicznie można zdefiniować skróconą wersję polecenia **\author** lub **\date**, np. `\author[J. Techowski et al.]{...}`, jeśli mamy kilku autorów. Symbole nawigacyjne pozwalają szybko poruszać się po prezentacji. Jeżeli nie chcemy z nich korzystać, można je wyłączyć za pomocą polecenia `\setbeamertemplate{navigation symbols}{}`.

Wewnątrz środowiska **frame** możemy stosować dowolne polecenia LaTeX-a, które poznaliśmy. Bez problemu skopiujemy tabelę, wzory matematyczne, definicje itd. Beamer stara się zmieścić na slajdzie całą zawartość środowiska **frame**. W tym celu odstępy pionowe między elementami slajdu mogą zostać zredukowane. Jeżeli zawartość jest za duża, to część nie będzie widoczna (istnieje możliwość włączenia automatycznego dzielenia na kolejne slajdy dużej zawartości, ale mamy większą kontrolę nad prezentacją, gdy dzielimy materiał na slajdy samodzielnie). Domyślnie zawartość slajdów jest wyśrodkowana w pionie. Jeżeli chcemy, aby zawartość była zawsze dosuwana do góry, to można dodać opcję `t` w poleceniu **\documentclass**. Niezależnie od ustawień globalnych możemy ustawiać dodatkowe opcje dla poszczególnych slajdów. W tabeli 13.1 przedstawiono najważniejsze opcje środowiska **frame**.

Tabela 13.1. Wybrane opcje środowiska **frame**

Opcje	Znaczenie
`b`, `c`, `t`, `s`	sposób wyrównania treści slajdu w pionie, dla opcji `s` konieczne jest dodanie elastycznych poleceń, np. **\vspace**
`fragile`	opcja informująca o obecności niestandardowego tekstu, używamy jej, jeśli na slajdzie korzystamy ze środowiska **verbatim** lub podobnego
`label=x`	zdefiniowanie własnej etykiety dla slajdu
`plain`	wyłączenie nagłówka, stopki i paneli bocznych – pusty slajd
`squeeze`	wyłączenie pionowych odstępów między wierszami

Użycie na slajdzie środowiska **verbatim**, **lstlisting** lub powiązanych z nimi poleceń wewnątrzakapitowych wygeneruje błąd kompilacji, jeśli zapomnimy o opcji `fragile`. Opcja `plain` przydaje się, jeżeli na slajd chcemy wrzucić większy rysunek i zależy nam na jak największej dostępnej przestrzeni.

Zawartość slajdu możemy organizować w wielu kolumnach za pomocą środowiska **columns**. Kolumny w środku środowiska tworzymy albo za pomocą środowiska **column**, albo polecenia **\column**. Jako parametry opcjonalne środowiska **columns** możemy użyć m.in. opcji związanych z wyrównaniem zawartości wewnątrz kolumn:

- `b` – wyrównanie w pionie według dolnych linii kolumn;
- `c` – wyśrodkowanie kolumn w pionie;
- `t` – wyrównanie w pionie według pierwszych linii kolumn (dokładniej wg linii bazowej);
- `T` – wyrównanie w pionie według góry pierwszej linii.

Opcja `t` dobrze sobie radzi, jeżeli kolumny zawierają tekst. Jeżeli zamieszczamy grafikę, to lepiej jest użyć `T` (przy opcji `t` zdarza się, że kolumna zawierająca wyłącznie rysunek jest źle wyrównana). Do opcji środowiska możemy też dodać `totalwidth=x`, co pozwala na zdefiniowanie szerokości środowiska **`columns`**. Jeżeli podamy wartość większą niż **`\textwidth`**, to zawartość wejdzie na marginesy. Podobnie się stanie, jeżeli nie podajemy tej opcji, a suma szerokości kolumn jest za duża. Przykład wykorzystania wielu kolumn pokazano na listingu 13.2.

Listing 13.2. Wykorzystanie kolumn przy składzie slajdu

```
\begin{frame}{Pierwsi polscy nobliści}

\begin{columns}[T]
\column{0.2\textwidth}
\begin{enumerate}
\item fizyka
\item literatura
\item chemia
\item literatura
\end{enumerate}
\column{0.5\textwidth}
\textbf{Fizyka}: badania nad zjawiskiem promieniotwórczości\\
\textbf{Chemia}: odkrycie polonu i radu, wydzielenie czystego radu
i badanie właściwości pierwiastków promieniotwórczych.
\column{0.15\textwidth}
\includegraphics[scale=0.05]{maria-sklodowska-curie}
\end{columns}
\vspace{5mm}

\alert{Maria Skłodowska-Curie jest jedyną kobietą, która otrzymała
dwie nagrody w dwóch różnych dyscyplinach.}
\vspace{3mm}

\begin{columns}[T, totalwidth=1\textwidth]
\column{0.2\textwidth}
\begin{block}{1903}
Maria Skłodowska-Curie
\end{block}
\column{0.2\textwidth}
\begin{block}{1905}
Henryk Sienkiewicz
\end{block}
\column{0.2\textwidth}
\begin{block}{1911}
```

```
35 Maria Skłodowska-Curie
36 \end{block}
37 \column{0.2\textwidth}
38 \begin{block}{1924}
39 Władysław Reymont
40 \end{block}
41 \end{columns}
42 \end{frame}
```

Pierwszy układ kolumn tworzy środowisko w liniach 3–17. W ramach tego środowiska za pomocą polecenia `\column{}` tworzymy trzy kolumny (linie 4, 11 i 15). Polecenie jako parametr przyjmuje szerokość tworzonej kolumny. Kolumna taka automatycznie zostaje zakończona, gdy pojawi się kolejne polecenie `\column` lub środowisko `column`, lub koniec środowiska `columns`. Zamiast polecenia można użyć środowiska. Ten sam efekt uzyskamy, jeśli np. w linii 4 umieścimy kod `\begin{column}{0.2\textwidth}`, a przed linią 11 dodamy `\end{column}`. Jako parametr opcjonalny możemy podać literę określającą wyrównanie, jeśli chcemy nadpisać ustawienia środowiska `columns`. W podanym przykładzie wszystkie kolumny są tworzone jako procent szerokości tekstu, ale możemy podać również konkretne liczby, np. 3 cm. Widać, że niezależnie od zawartości kolumny (wypunktowanie, tekst, rysunek) zostały one wyrównane do góry.

Pojedyncze środowisko **columns** nie musi zajmować całego slajdu. Jego wysokość jest wyznaczona przez wysokość najwyższej z kolumn. Poniżej środowiska możemy umieszczać dalszą zawartość slajdu (jeśli mamy miejsce). Tekst znajdujący się w liniach 20–21 jest już poza środowiskiem **columns** i jest składany na całej szerokości slajdu. W przypadku tego tekstu zastosowano polecenie Beamera **\alert{}**. Wyróżnia ono kolorem tekst będący parametrem. Domyślnie jest to kolor jaskrawo czerwony, ale zależy to od użytego motywu lub własnych ustawień, jeśli samodzielnie zmieniamy ustawienia motywu.

W liniach 24–41 mamy kod kolejnego środowiska **columns**. Tym razem utworzono 4 kolumny. Wszystkie kolumny mają przypisaną tę samą szerokość. Dodatkowo ograniczono szerokość środowiska (linia 24), co daje w efekcie dość gęste upakowanie kolumn.

W drugim środowisku **columns** zawartość każdej kolumny jest umieszczona w bloku – środowisko **block**. Jest to środowisko Beamera, które pozwala na umieszczenie swojej zawartości w ramce i nadanie jej tytułu (wygląd bloków zależy od motywu). Blok funkcjonuje podobnie jak ministrona, tj. może zawierać wiele akapitów tekstu, tabele, wzory matematyczne itd. Jedynym obowiązkowym parametrem jest tytuł bloku (linie 26, 30, 34 i 38). W przypadku motywu `Warsaw` dodawana jest granatowa belka, a tytuł bloku składany jest białym kolorem. Poniżej belki na szaroniebieskim tle mamy zawartość bloku. Jeżeli zostawimy pusty tytuł, belka nie zostanie wygenerowana, a cały blok będzie miał kolorystykę typową dla zawartości. Beamer oferuje jeszcze dwie wersje bloków – środowiska **alertblock** i **exampleblock**. Używa się ich dokładnie tak samo. Jedyna różnica między tymi środowiskami to ich kolory zależne od motywu. W przypadku motywu `Warsaw` kolorystyka środowiska **alertblock** bazuje na kolorze czerwonym, a **exampleblock** – zielonym. Jeżeli blok zostanie użyty poza środowiskiem **columns**, to będzie korzystał z całej szerokości slajdu. Możemy ograniczyć jego szerokość, umieszczając go w środowisku **minipage**.

13.3 Slajd jako seria scen

Gdy planujemy slajd, niekoniecznie chcemy od razu wyświetlać całą jego zawartość. Jeżeli zawiera on wypunktowanie, to możemy chcieć wyświetlić je punkt po punkcie, aby cały czas skupiać uwagę publiczności na ostatnim wyświetlonym (często równocześnie wyróżnionym) punkcie. Być może chcemy przedstawić diagram, który będzie odsłaniany partiami, albo chcemy w pewnym momencie podmienić część slajdu. Wszystkie te sytuacji prowadzą do przygotowania naszego slajdu w formie serii scen. Wybranie przycisku przejścia dalej (np. na prezenterze) będzie wówczas prowadzić do kolejnej sceny naszego slajdu,

a dopiero po wyświetleniu ostatniej z nich – do kolejnego slajdu. Sceny stosowane w Beamerze są odpowiednikami widoków, których używa edytor Ipe (zob. podrozdział 10.7).

Najprostszą metodą podziału wyświetlania slajdu na sceny jest użycie polecenia **\pause**. Załóżmy, że umieścilibyśmy to polecenie po 18 i 22 linii w kodzie na listingu 13.2. W takiej sytuacji w pierwszej scenie wyświetlony zostałby tylko górny układ kolumn (patrz listing 13.2), w drugiej scenie zostałby dodany tekst w środku slajdu, a w trzeciej scenie dolny układ kolumn.

Beamer numeruje sceny, zaczynając od 1. Niektóre polecenia i środowiska mogą jako dodatkowy parametr umieszczany między znakami < i > przyjąć specyfikację dotyczącą scen. Zawartością takiej specyfikacji może być lista wartości oddzielonych przecinkami, zakresy zapisywane w formie `x-y`, przy czym jeden z końców może zostać pominięty, i kombinacja tych wszystkich możliwości. Przykłady specyfikacji i ich znaczenie pokazano w tabeli 13.2.

Tabela 13.2. Przykłady specyfikacji wyświetlania scen

Specyfikacja	Znaczenie
`<2,4>`	dotyczy scen numer 2 i 4
`<2-4>`	dotyczy scen numer 2, 3 i 4
`<-4>`	dotyczy scen od początku do 4 włącznie
`<4->`	dotyczy scen od 4 do ostatniej
`<-2,4,6-8,10->`	dotyczy scen numer 1, 2, 4, 6, 7, 8, 10 i kolejnych

Jeżeli specyfikację dołączymy do poleceń: **\alert**, **\color**, **\emph**, **\textbf**, **\textit**, **\textmd**, **\textnormal**, **\textrm**, **\textsc**, **\textsf**, **\textsl**, **\texttt**, **\textup**, **\textcolor**, to ich działanie jest ignorowane w scenach niewymienionych w specyfikacji. Na przykład użycie kodu **\textcolor**`<2,4>{blue}{...}` oznacza, że tekst będący drugim parametrem będzie wyświetlany na niebiesko w scenach 2 i 4. W pozostałych scenach będzie wyświetlany tak, jakby polecenia **\textcolor** w ogóle nie było. Jeżeli polecenie przyjmuje więcej niż jeden parametr, jak np. **\textcolor**, to specyfikację podajemy od razu po nazwie polecenia.

Specyfikacja scen może być dołączona do niektórych środowisk, w tym do wszystkich bloków, o których była mowa, a także środowisk **theorem**, **definition**, **proof** itp. zdefiniowanych w pakiecie `beamer`. W tych przypadkach zawartość środowiska jest wyświetlana tylko w scenach wymienionych w specyfikacji. Wracając do naszego przykładu z kolumnami, załóżmy, że do bloków w drugim środowisku **columns** dodaliśmy specyfikacje jak na listingu 13.3. Wtedy blok 1903 jest wyświetlana cały czas, 1905 od drugiej sceny itd.

Listing 13.3. Wykorzystanie specyfikacji scen dla bloków

```
\column{0.2\textwidth}
\begin{block}<1->{1903}
Maria Skłodowska-Curie
\end{block}
\column{0.2\textwidth}
\begin{block}<2->{1905}
Henryk Sienkiewicz
\end{block}
\column{0.2\textwidth}
\begin{block}<3->{1911}
Maria Skłodowska-Curie
\end{block}
\column{0.2\textwidth}
\begin{block}<4->{1924}
Władysław Reymont
\end{block}
```

Ten sam efekt uzyskamy, stosując przy każdym bloku specyfikację `<+->`. Beamer używa licznika `beamerpauses`, który na początku każdego slajdu ma wartość 1. W specyfikacji `<+->` znak + jest zastępowany bieżącą wartością licznika, a sam licznik jest zwiększany o 1. W tym przypadku dostalibyśmy kolejno `<1->`, `<2->` itd. W kolejnym przykładzie zobaczymy większą korzyść z tego podejścia.

Dwie inne instrukcje współpracujące ze specyfikacją scen w podobny sposób jak wspomniane środowiska to **`\item`** i **`\includegraphics`**.

Listing 13.4. Wyświetlanie listy punkt po punkcie

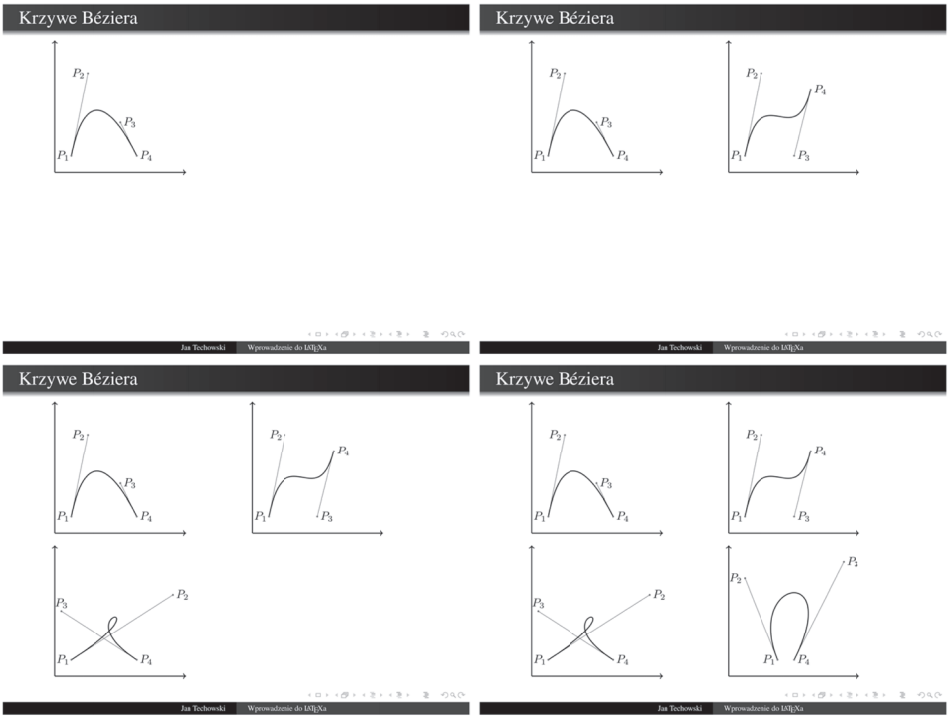
```
\begin{enumerate}
\item<+-> 1903 Maria Skłodowska-Curie -- fizyka
\item<+-> 1905 Henryk Sienkiewicz -- literatura
\item<+-> 1911 Maria Skłodowska-Curie -- chemia
\item<+-> 1924 Władysław Reymont -- literatura
\end{enumerate}
```

Na listingu 13.4 pokazano przykład organizacji środowiska **`enumerate`** do wyświetlania zawartości punkt po punkcie. Użycie specyfikacji `<+->` zamiast `<1->`, `<2->` itd. ma tę zaletę, że jeżeli w środku musielibyśmy dołożyć kolejne punkty, to nie musimy ręcznie poprawiać liczb. Wyświetlanie list punkt po punkcie jest na tyle popularne, że zamiast dodawać specyfikację do każdego polecenia **`\item`**, możemy dodać parametr `[<+->]` do środowiska **`enumerate`** (również **`itemize`** i **`description`**). Jeżeli specyfikację rozbudujemy do wersji `[<+-| alert@+>]`, to dodatkowo ostatnio wyświetlony punkt będzie wyróżniony (polecenie **`\alert`**).

Na listingu 13.5 pokazano wykorzystanie specyfikacji scen przy wyświetlaniu grafiki. Cztery rysunki zawierające przykłady krzywych Béziera rozmieszczono w środowisku **tabular**. Do każdego z poleceń **\includegraphics** dodano specyfikację w postaci <+->. W efekcie mamy cztery sceny, w których odsłaniane są kolejne rysunki.

Listing 13.5. Wyświetlanie grafiki scena po scenie

```
\begin{frame}[t]{Krzywe B\'eziera}

\begin{tabular}{p{5cm}p{5cm}}
\includegraphics<+->[scale=0.9]{bezier1.pdf}&
\includegraphics<+->[scale=0.9]{bezier2.pdf}\\
\includegraphics<+->[scale=0.9]{bezier3.pdf}&
\includegraphics<+->[scale=0.9]{bezier4.pdf}
\end{tabular}
\end{frame}
```



Beamer zawiera także kilka poleceń, które pozwalają na „opakowanie" części naszego slajdu i wyświetlanie jej tylko w określonych scenach. Dwa podstawowe to **\only**<>{} i **\uncover**<>{}. Różnica między nimi polega na tym, że w przypadku użycia polecenia **\only** niewyświetlana treść nie zajmuje miejsca, zaś

w przypadku **\uncover** – miejsce jest zajęte, nawet jeżeli treść jest niewidoczna. Własności te wpływają na ciekawe zastosowanie obu poleceń – **\uncover** używamy, gdy chcemy wyświetlać kolejne fragmenty, zaś **\only**, gdy chcemy podmieniać treść na slajdzie.

Pisząc o treści, niekoniecznie mam na myśli tekst. Klasa `beamer` i pakiet `tikz` dobrze ze sobą współpracują. Kolejny przykład, który omówię, jest jeszcze jednym argumentem za tym, aby poznać możliwości rysowania z użyciem pakietu `tikz`. Jeżeli w środowisku **frame** umieścimy środowisko **tikzpicture**, to wewnątrz tego środowiska możemy używać na przykład poleceń **\only** i **\uncover**. Pozwala nam to odsłaniać rysunek partiami. Ze względu na sposób konstruowania slajdów (kolejność dodawania komponentów) nie powinniśmy wewnątrz środowiska **tikzpicture** używać polecenia **\pause** (np. nie będziemy widzieć stopki slajdu, dopóki rysunek nie będzie kompletny).

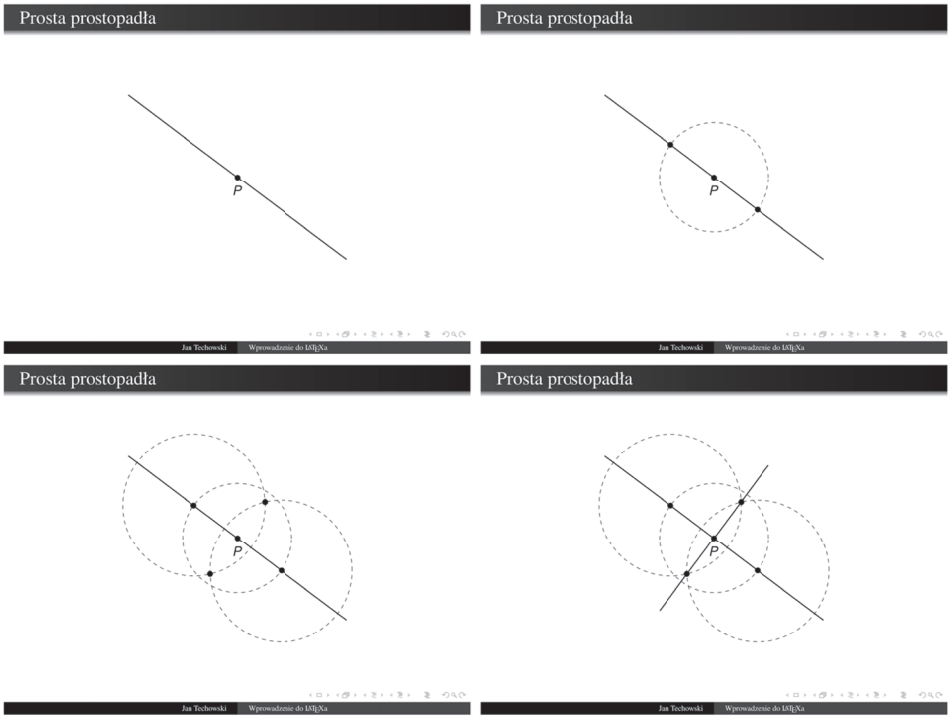
Rozpatrzmy przykład z listingu 13.6. W linii 3 użyta została opcja `scale`, dzięki czemu powiększyliśmy cały rysunek, aby lepiej pasował do wielkości slajdu. Rysowanie podzielone zostało na cztery etapy przez opakowanie instrukcji `tikz` poleceniem **\uncover** (linie 4, 9, 13, 18). Przy specyfikacji scen skorzystaliśmy z wersji `<+->`, co jest wygodne, gdyż kolejne partie do wyświetlenia są ułożone jedna po drugiej. W pierwszej scenie kreślimy prostą i zaznaczamy punkt, w których chcemy wystawić prostopadłą. W drugiej scenie kreślimy okrąg i wyznaczamy punkty przecięcia prostej z okręgiem (zob. podrozdział 11.6). W kolejnej scenie rysowane są dwa kolejne okręgi i wyznaczane ich punkty przecięcia. W ostatniej scenie kreślimy szukaną prostą.

```
\usepackage{tikz}
\usetikzlibrary{intersections,calc}
```

Listing 13.6. Konstrukcja prostej prostopadłej – wyświetlanie rysunku partiami

```
1  \begin{frame}{Prosta prostopadła}
2  \begin{center}
3  \begin{tikzpicture}[scale=1.5]
4  \uncover<+->{
5  \draw[name path=l, thick] (0,4) -- (4,1);
6  \coordinate (P) at (2,2.5);
7  \fill (P) circle (1.5pt);
8  \node at (P) [below=3pt] {$P$};}
9  \uncover<+->{
10 \draw[name path=o, dashed] (P) circle (1cm);
11 \fill [name intersections={of=l and o, name=Q}]
12   \foreach \i in {1,2} {(Q-\i) circle (1.5pt)};}
13 \uncover<+->{
```

```
14 \draw[name path=o1, dashed] (Q-1) circle (1.3cm);
15 \draw[name path=o2, dashed] (Q-2) circle (1.3cm);
16 \fill [name intersections={of=o1 and o2, name=R}]
17   \foreach \i in {1,2} {(R-\i) circle (1.5pt)};}
18 \uncover<+->{
19 \draw[thick] (P) -- ++($ (R-2) - (R-1) $);
20 \draw[thick] (P) -- ++($ (R-1) - (R-2) $);}
21 \end{tikzpicture}
22 \end{center}
23 \end{frame}
```

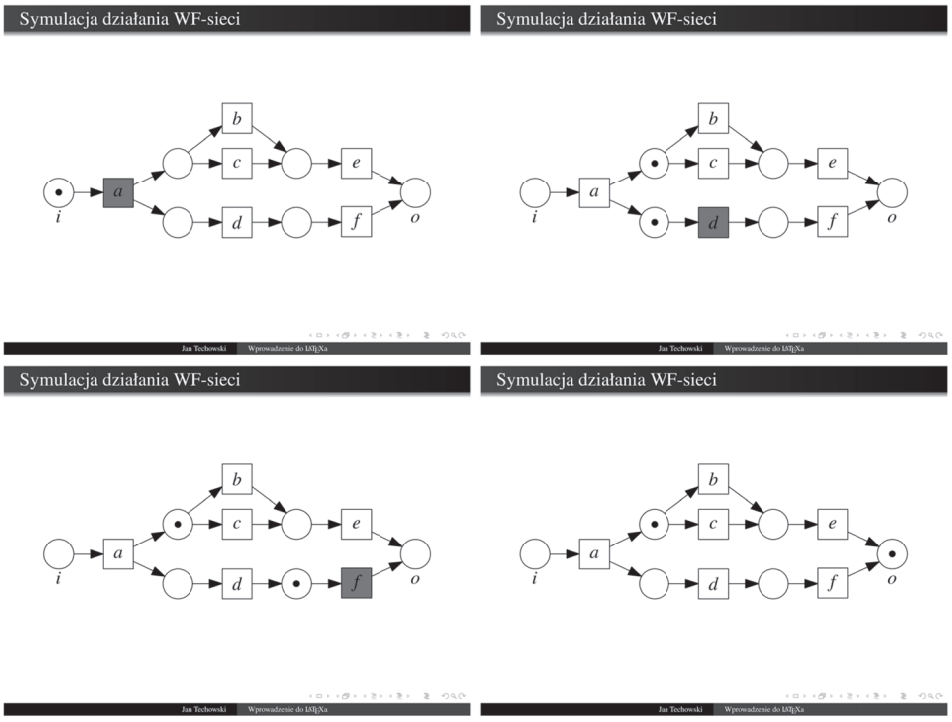


W liniach 19–20 skorzystano z możliwości wykonywania prostych obliczeń w pakiecie `tikz` (wymagana jest biblioteka `calc`). Wyrażenia do obliczenia umieszcza się wewnątrz `($ $)`. Tutaj służy ono do wyznaczenia wektora kierunkowego szukanej prostej (różnica dwóch punktów). Używamy go, aby prosta prostopadła była reprezentowana przez odcinek dłuższy niż ten łączący punkty przecięcia. Podsumowując ten przykład, rysunek przygotowany w `tikz` i umieszczony na przykład w publikacji możemy łatwo przenieść na slajd i wyświetlać go partiami.

Wewnątrz konstrukcji (`$ $`) można realizować obliczenia będące liniową kombinacją współrzędnych punktów, np. dla punktów P,Q,R możemy policzyć $\alpha P + \beta Q + \gamma R$, gdzie $\alpha,\beta,\gamma \in \mathbb{R}$. W szczególności możemy odjąć od siebie dwa punkty.

Listing 13.7. Złudzenie symulacji działania WF-sieci

```
\begin{frame}
\frametitle{Symulacja działania WF-sieci}

\begin{center}
\only<+>{\includegraphics{wfnet1.pdf}}%
\only<+>{\includegraphics{wfnet2.pdf}}%
\only<+>{\includegraphics{wfnet3.pdf}}%
\only<+>{\includegraphics{wfnet4.pdf}}%
\end{center}
\end{frame}
```



Kolejny przykład (listing 13.7) pokazuje zastosowanie polecenia **`\only`** do uzyskania wrażenia symulacji na slajdzie. W kolejnych scenach w tym samym miejscu wyświetlane są inne rysunki pokazujące różne stany modelu w postaci

WF-sieci (workflow net jest siecią Petriego o specyficznej strukturze, dla naszych potrzeb ważne jest, że zmieniają się stany – przesuwają się czarne tokeny oraz podświetlana jest kolejna aktywność). Specyfikacje w kolejnych liniach 5–8 zamienią się na <1>, <2>, <3> i <4> (możemy też sami wpisać liczby zamiast +). Ważny jest też znak % na końcu każdej linii 5–7. Jeśli o nim zapomnimy, to kolejny rysunek będzie przesunięty o jedną spację w prawo, co da nieestetyczne wrażenie drgania naszego rysunku.

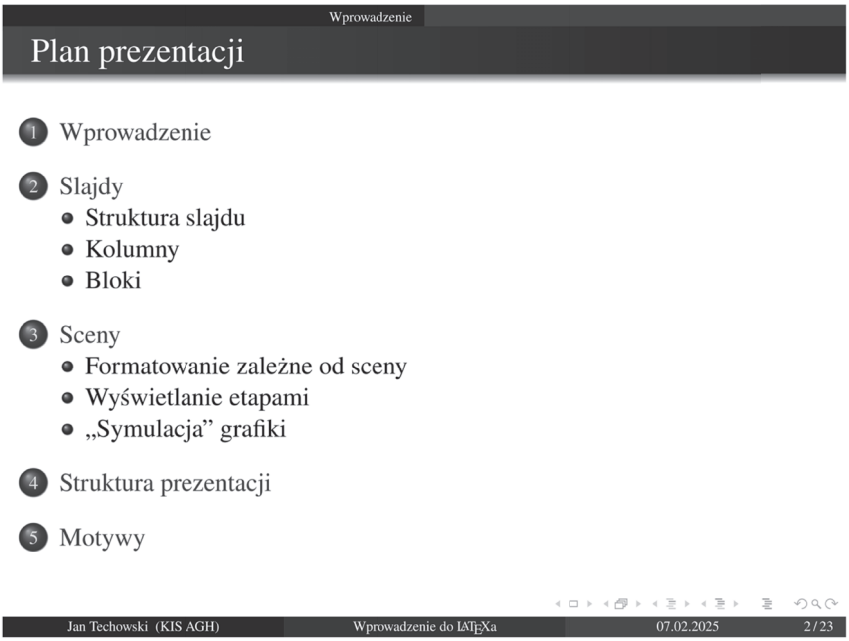
W prezentowanym przykładzie wszystkie rysunki były tej samej wielkości. Jeżeli będą się one różnić (np. w niektórych scenach będą dodatkowe etykiety, które zmieniają całkowity rozmiar rysunku), to będziemy mieć efekt drgań. Rozwiązaniem tego problemu może być użycie środowiska **overlayarea** [47] i dołączanie grafiki we wnętrzu tego środowiska. Środowisko przyjmuje dwa parametry, pierwszym jest szerokość, a drugim wysokość obszaru. Definiuje one prostokąt, który ma stałe rozmiary niezależnie od sceny.

13.4 Zarządzanie strukturą prezentacji

Przygotowując prezentację, dobrze jest pokazać na wstępie, jaki jest jej plan, a w trakcie prezentacji udostępniać w jakiś sposób informację, na jakim etapie jesteśmy. Beamer może automatycznie wygenerować spis treści dla prezentacji za pomocą polecenia **\tableofcontents**. Będzie to możliwe, jeżeli podzielimy prezentację na sekcje, podsekcje itd. Robimy to, wstawiając polecenia **\section**, **\subsection** itd. między środowiskami **frame**. W ten sposób poszczególne slajdy zostają przypisane do właściwych części prezentacji.

Na rys. 13.3 pokazano slajd z rys. 13.1, który powstał wyłącznie jako efekt umieszczenia w pustym środowisku **frame** polecenia **\tableofcontents**. Motyw `Warsaw` użyty w tej prezentacji został w tym przypadku uzupełniony motywem zewnętrznym `infolines` (zob. podrozdział 13.5). W efekcie mamy dodatkowy pasek na górze slajdu, na którym wyświetlany jest tytuł bieżącej sekcji i podsekcji (o ile są zdefiniowane, tutaj nie ma podsekcji przy tym slajdzie). Dodatkowo zmienił się wygląd stopki. W prawym dolnym rogu mamy informację o numerze bieżącego slajdu i liczbie wszystkich slajdów.

Domyślnie slajdy są wyświetlane w kolejności od pierwszego do ostatniego jeden po drugim. Zdarza się jednak, że podczas prezentacji chcemy na chwilę wrócić do wcześniejszego slajdu, a nawet do konkretnej jego sceny. Jednym z rozwiązań jest zastosowanie polecenia **\againframe**`<>{}`. Parametrem obowiązkowym polecenia jest etykieta slajdu, do którego chcemy się przenieść, a w specyfikacji możemy wskazać, o którą scenę tego slajdu chodzi. Etykiety tworzymy z użyciem opcji `label` środowiska **frame** (zob. tabela 13.1).



Rys. 13.3. Prezentacja z listingu 13.1

Wyobraźmy sobie taką sytuację. Część naszej prezentacji o polskich noblistach chcemy zorganizować następująco. Na pierwszym slajdzie mamy wypunktowanych noblistów. Chcemy ich wyświetlać punkt po punkcie z wyróżnieniem ostatnio wyświetlonego, ale po wyświetleniu danego punktu chcemy przejść do dodatkowego slajdu, na którym pokazano szczegóły dotyczące danej osoby. Po wyświetleniu tego slajdu chcemy wrócić do kolejnego punktu na pierwszym slajdzie. Szkielet takiego rozwiązania pokazano na listingu 13.8.

Listing 13.8. Powracanie do slajdów podczas prezentacji

```
\begin{frame}<1>[label=noblisci]{Pierwsi polscy nobliści}
\begin{enumerate}[<+- | alert@+>]
\item 1903 Maria Skłodowska-Curie -- fizyka
\item 1905 Henryk Sienkiewicz -- literatura
\item 1911 Maria Skłodowska-Curie -- chemia
\item 1924 Władysław Reymont -- literatura
\end{enumerate}
\end{frame}

\begin{frame}{Maria Skłodowska-Curie (fizyka)}
%... tu zawartość slajdu
\end{frame}
```

```
13 \againframe<2>{noblisci}
14
15 \begin{frame}{Henryk Sienkiewicz (fizyka)}
16 %... tu zawartość slajdu
17 \end{frame}
18 \againframe<3>{noblisci}
19
20 \begin{frame}{Maria Skłodowska-Curie (chemia)}
21 %... tu zawartość slajdu
22 \end{frame}
23 \againframe<4>{noblisci}
24
25 \begin{frame}{Władysław Reymont}
26 %... tu zawartość slajdu
27 \end{frame}
```

Pierwszy slajd w rozważanej części prezentacji został zdefiniowany z użyciem etykiety `noblisci`. Zawiera on używane wcześniej wypunktowanie. Opcja `<+- | alert@+>` oznacza, że wypunktowanie będzie wyświetlane punkt po punkcie z wyróżnieniem ostatnio wyświetlonego za pomocą polecenia **\alert**. Zatem nasz pierwszy slajd będzie miał 4 sceny. Specyfikacja `<1>` w linii 1 oznacza, że po wyświetleniu pierwszej sceny prezentacja przejdzie do kolejnego slajdu (linie 10–12). Po wyświetleniu slajdu nt. nobla z fizyki dla Marii Skłodowskiej-Curie wrócimy do drugiej sceny na slajdzie `noblisci` (linia 13). Tu będzie wyświetlona tylko ta jedna scena. Po tej scenie przejdziemy do pierwszego niewyświetlonego jeszcze slajdu (linie 15–17). Po nim wracamy do sceny trzeciej na slajdzie `noblisci` (linia 18) itd.

Gdyby specyfikacja przy pierwszym slajdzie była postaci `<1-2>`, to przejście do kolejnego slajdu nastąpiłoby po wyświetleniu dwóch scen (pomijamy teraz kwestie, czy to ma sens w tej prezentacji). Z drugiej strony, jeżeli przejdziemy dalej i nie będzie polecenia **\againframe** prowadzącego do jakiejś pominiętej sceny, to w ogóle nie zostanie ona wyświetlona.

13.5 Motywy

Wybrany motyw decyduje nie tylko o kwestiach kolorystycznych, ale też o kompozycji slajdów – na przykład czy wyświetlany jest panel boczny, czy nie. Ciekawą galerię porównującą motywy można znaleźć pod adresem `http://latex-beamer.net`. Kompletne motywy jak `Warsaw` lub `CambridgeUS` nazywane są motywami prezentacji. Dodatkowo w Beamerze wyróżnia się:

– motywy kolorów – zawierają specyfikację kolorów używanych na prezentacji, wśród nich są kompletne motywy kolorów, a także odnoszące się tylko do części zewnętrznej albo wewnętrznej slajdów;
– motywy czcionek – zawierają specyfikację dotyczącą czcionek i ich atrybutów;
– motywy wewnętrzne – zawierają specyfikację dotyczącą elementów umieszczanych wewnątrz środowiska **frame**, np. jak wygląda wyliczenie, wypunktowanie, bloki (np. narożniki zaokrąglone lub ostre) itd.
– motywy zewnętrzne – zawierają specyfikację dotyczącą wyglądu slajdu, np. czy jest panel boczny (i po której stronie), czy go nie ma, czy są wyświetlane symbole nawigujące itp.

Możemy wybrać dowolny typ motywu, korzystając z poleceń pakietu `beamer`: **\usecolortheme**{}, **\usefonttheme**{}, **\useinnertheme**{} i **\useoutertheme**{}. Oznacza to możliwość stworzenia własnej kombinacji motywów lub dołączenia do wybranego motywu prezentacji jakiegoś dodatkowego, aby motyw główny nieco zmienić. Najczęściej zmiana taka dotyczy kolorystyki, np. dodanie w naszej prezentacji pod poleceniem ładującym motyw `Warsaw` polecenia **\usecolortheme**{crane} spowoduje, że wprawdzie nie zmieni się struktura slajdów, ale będzie dominować kolorystyka związana z kolorem żółtym. Możemy w ogóle nie ładować motywu prezentacji, ale przygotować własną kombinację. Przykład pokazano na listingu 13.9. Użyty motyw `seahorse` dotyczy kolorystyki elementów zewnętrznych, `rose` wewnętrznych. Brak załadowania motywu dla czcionek oznacza użycie motywu domyślnego. Przegląd wszystkich motywów znajduje się w dokumentacji [47]. Wśród motywów dla czcionek warto zwrócić uwagę na motyw `structurebold`, który skutkuje użyciem pogrubienia dla treści umieszczanych w nagłówku, stopce i w bocznym panelu.

Listing 13.9. Kombinacja motywów określających wygląd prezentacji

```
1 \useoutertheme{infolines}
2 \useinnertheme{rectangles}
3 \usecolortheme{seahorse}
4 \usecolortheme{rose}
```

13.6 Przygotowanie własnego motywu

W podrozdziale tym nie będziemy mówili o budowie kompletnego motywu dla Beamera. Skupimy się przede wszystkim na kwestii tła slajdu i zobaczymy, jak szybko można odmienić prezentację, aby pasowała do naszych oczekiwań. Zacznijmy od wersji, w której tło jest po prostu gotowym plikiem graficznym. Może

być tak, że są to dwa pliki, jeden na slajd tytułowy, a drugi na pozostałe. W naszym przykładzie wykorzystamy jeden plik. Tło użyte do demonstracji zostało przygotowane w programie Gimp, ale bez problemu można pobrać darmowe atrakcyjne tła z internetu (w szczególności gdy robimy to na własny użytek). Obraz wyeksportowany do formatu `png` ma wymiary 1200 × 900 pikseli, przy czym bardziej istotne są proporcje niż dokładne wymiary. Oczywiście przy zbyt małym rozmiarze grafiki może się pojawić problem z jakością tła, gdy zostanie przeskalowane do wymaganych rozmiarów.

Na listingu 13.10 pokazano kluczowe dla naszej prezentacji ustawienia. Plik `tlo.png` został ustawiony jako tło dla wszystkich slajdów (linie 4–5). Jeżeli chcemy użyć oddzielną grafikę dla slajdu tytułowego, to ustawienia tła wraz z ramką tytułową najlepiej ulokować w dodatkowych nawiasach klamrowych. Ponadto został użyty motyw kolorów `seagull` i motyw czcionek `structurebold` (linie 1–2). Cały pokazany kod należy dodać w preambule. Sama zawartość prezentacji pozostaje bez zmian.

Listing 13.10. Ustawienia prezentacji z własnym tłem

```
1 \usecolortheme{seagull}
2 \usefonttheme{structurebold}
3
4 \setbeamertemplate{background}
5 {\includegraphics[width=\paperwidth]{tlo.png}}
```

W ostatnim przykładzie pokazanym na listingu 13.11 wszystkie ozdoby slajdu zostały narysowane z użyciem pakietu `tikz`. Na początku dołączamy pakiet `tikz` i `pgfornament` [37] (zawiera motyw gałązek, którego chcemy użyć). Korzystamy z tych samych motywów co poprzednio (linie 4–5), ale tym razem usuwamy symbole nawigacyjne (linia 6). Zamiast polecenia **`\includegraphics`** używamy tym razem środowiska **`tikzpicture`**. Opcje użyte w linii 9 są potrzebne ze względu na odwoływanie się do specjalnego węzła `current page`. W liniach 10–13 rysujemy dwie szare linie na górze i na dole strony (czyli slajdu). Do tego

dodajemy dwa razy ornament z pakietu `pgfornament`. Pierwszy węzeł (linia 15) jest ustawiony tak, że widzimy tylko fragment ornamentu w prawym górnym rogu. W drugim węźle (linie 16–17) ornament jest obrócony i widoczny w całości na lewym marginesie.

```
\usepackage{pgfornament}
\usepackage{tikz}
```

Listing 13.11. Konstrukcja tła prezentacji z użyciem pakietu `tikz`

```
1  \usepackage{pgfornament}
2  \usepackage{tikz}
3
4  \usecolortheme{seagull}
5  \usefonttheme{structurebold}
6  \setbeamertemplate{navigation symbols}{}
7
8  \setbeamertemplate{background canvas}{%
9  \begin{tikzpicture}[remember picture,overlay]
10 \draw[line width=12pt,gray] (current page.north west) --
11   (current page.north east);
12 \draw[line width=12pt,gray] (current page.south west) --
13   (current page.south east);
14
15 \node at (12,-0.2) {\pgfornament[scale=0.5, color=gray!30]{87}};
16 \node[rotate=90] at (0.4,-5) {\pgfornament[scale=0.5,
17   color=gray!30]{87}};
18 \end{tikzpicture}%
19 }
```

Rozdział 14

Uzupełnienie dla dużych projektów

W niniejszym rozdziale zebranych zostało kilka tematów, które stanowią uzupełnienie dla większych dokumentów, choć nie zawsze są używane. Będzie mowa między innymi o zarządzaniu nagłówkami i stopkami strony, generowaniu spisów rysunków, tabel, listingów i algorytmów, o przygotowaniu skorowidza, o definiowaniu własnych poleceń w celu przyspieszenia pracy nad większymi projektami, a zakończymy podrozdziałem na temat dodania hiperłączy do naszego pliku `pdf`.

14.1 Przypisy i notatki na marginesie

LaTeX udostępnia polecenie **`\footnote`**`[]{}` do tworzenia przypisów. Parametrem obowiązkowym polecenia jest treść przypisu. Przypis umieszczamy bezpośrednio po zdaniu lub słowie, do którego się odnosi (bez spacji przed poleceniem). Jeżeli przypis odnosi się do całego zdania, to w języku polskim umieszczamy go przed kropką kończącą zdanie. W języku angielskim przypisy umieszcza się po kropce lub przecinku. Przypisy są automatycznie numerowane, ale można wymusić konkretny numer, podając go jako parametr opcjonalny.

W miejscu użycia polecenia **`\footnote`** umieszczany jest w indeksie górnym numer porządkowy przypisu, a sam przypis umieszczany jest na dole strony. Zazwyczaj przypisy oddzielone są od reszty strony krótką poziomą linią. Przypis nie musi zawierać wyłącznie tekstu. Możemy w nim umieścić środowiska listowe, tabele, wzory matematyczne itp. W treści przypisu możemy zdefiniować etykietę za pomocą polecenia **`\label`**, co pozwala na odwoływanie się do danego przypisu. Dodatkowo dostępne są polecenia **`\footnotemark`**`[]` i **`\footnotetext`**`[]{}`. Pierwsze generuje wyłącznie numer przypisu, ale bez przypisu na dole strony, a drugie na odwrót – generuje przypis na dole strony,

ale nie umieszcza numeru w miejscu użycia polecenia. Parametr opcjonalny ma takie znaczenie jak dla polecenia `\footnote`. Jeżeli mamy dużo przypisów na stronie i w którymś przypadku wymusimy własny numer, to może się zdarzyć, że ten sam numer będzie użyty również przy automatycznym numerowaniu.

Na listingu 14.1 mamy przykład tekstu zapożyczonego z książki [23]. Pierwszy przypis rozpoczyna się w linii 1, jest notką na temat Charlesa Spearmana. Drugi przypis znajduje się w liniach 12–13 i jest krótką informacją o funkcji `rank`. Najciekawszy jest ostatni przypis. W kodzie źródłowym obejmuje on linie 16–29. W treści przypisu mamy m.in. trzy wzory złożone w trybie wystawowym. Przypis stanowi opis całej alternatywnej procedury, którą autor z jakiegoś powodu chciał wyłączyć z głównego tekstu. W książce [23] przypis ten nie znajduje się w całości na stronie, gdzie został wstawiony. Ze względu na długość jest kontynuowany na sąsiedniej stronie. Sytuacja taka często zdarza się przy bardzo długich przypisach.

Listing 14.1. Przykład składania przypisów

```
1  Współczynnik korelacji rang Spearmana\footnote{Charles Spearman
2  (1863-1945) -- psycholog angielski, pionier analizy czynnikowej,
3  zapoczątkował nowoczesne rozumienie inteligencji człowieka.} służy
4  do opisu siły korelacji dwóch cech w przypadku gdy:
5  \begin{itemize}[itemsep=0mm,topsep=0mm]
6  \item cechy są mierzalne, a badana zbiorowość jest nieliczna,
7  \item istnieją obserwacje odstające,
8  \item cechy mają charakter jakościowy, ale istnieje możliwość
9  ich uporządkowania.
10 \end{itemize}
11 Obliczenia rozpoczynamy od uporządkowania informacji. Wartościom
12 nadajemy rangi\footnote{Do rangowania w R przewidziana jest funkcja
13 \textit{rank}.}. Cesze $X$ nadajemy rangi $X_1,X_2,\dots,X_n$,
14 a cesze $Y$: $Y_1,Y_2,\dots,Y_n$. Za rangi przyjmuje się kolejne
15 liczby naturalne, zaczynając od 1. Jeśli mamy przypadek równych rang,
16 to nadajemy jako rangę średnią arytmetyczną\footnote{Jeśli liczba
17 remisów jest duża, należy zastosować poprawkę:
18 \begin{equation*}
19 r_S^* = \frac{S_x + S_y-\sum_{i=1}^{n}d_i^2}{2\sqrt{S_x S_y}},
20 \end{equation*}
21 gdzie
22 \begin{equation*}
23 S_x = \frac{n(n^2-1)- \sum_{i=1}^{g}(t_i^3 - t_i)}{12},
24 \end{equation*}
25 \begin{equation*}
26 S_y = \frac{n(n^2-1)- \sum_{i=1}^{h}(t_j^3 - t_j)}{12},
27 \end{equation*}
```

```
$g$, $h$ -- liczba remisowych sytuacji odpowiednio w próbie $X$ i $Y$,
$t_i$, $t_j$ -- liczebności tych remisowych grup, $d_i = X_i - Y_i$.}.
```

Współczynnik korelacji rang Spearmana[1] służy do opisu siły korelacji dwóch cech w przypadku gdy:

- cechy są mierzalne, a badana zbiorowość jest nieliczna,
- istnieją obserwacje odstające,
- cechy mają charakter jakościowy, ale istnieje możliwość ich uporządkowania.

Obliczenia rozpoczynamy od uporządkowania informacji. Wartościom nadajemy rangi[2]. Cesze X nadajemy rangi $X_1, X_2, \ldots, X_n$, a cesze Y: $Y_1, Y_2, \ldots, Y_n$. Za rangi przyjmuje się kolejne liczby naturalne, zaczynając od 1. Jeśli mamy przypadek równych rang, to nadajemy jako rangę średnią arytmetyczną[3].

[1] Charles Spearman (1863-1945) – psycholog angielski, pionier analizy czynnikowej, zapoczątkował nowoczesne rozumienie inteligencji człowieka.

[2] Do rangowania w R przewidziana jest funkcja *rank*.

[3] Jeśli liczba remisów jest duża, należy zastosować poprawkę:

$$r_S^* = \frac{S_x + S_y - \sum_{i=1}^{n} d_i^2}{2\sqrt{S_x S_y}},$$

gdzie

$$S_x = \frac{n(n^2-1) - \sum_{i=1}^{g}(t_i^3 - t_i)}{12},$$

$$S_y = \frac{n(n^2-1) - \sum_{i=1}^{h}(t_j^3 - t_j)}{12},$$

g, h – liczba remisowych sytuacji odpowiednio w próbie X i Y, t_i, t_j – liczebności tych remisowych grup, $d_i = X_i - Y_i$.

Przypisy umieszczane wewnątrz środowiska **minipage** są traktowane niezależnie i domyślnie oznaczane małymi literami. Są one umieszczane na dole ministrony.

Jeżeli chcemy umieścić notatkę na marginesie strony, możemy użyć polecenia **\marginpar**{}. Parametrem polecenia jest treść notatki. Zazwyczaj tego typu notatki używa się do wstawienia na margines słów kluczowych, aby wskazać czytelnikowi, o czym jest mowa w danym miejscu. Na przykład polecenie **\marginpar**{notatki\\ na marginesie} umieści na marginesie notatkę o treści swojego parametru. Jak widać, wewnątrz parametru możemy stosować łamanie linii, jeśli jest taka potrzeba. Przy druku jednostronnym notatki są umieszczane na prawym marginesie, przy druku dwustronnym na zewnętrznym. Jeżeli stosujemy druk dwukolumnowy, to LaTeX stara się wybrać bliższy margines. Jeżeli zawartość notki miałaby być zależna od wybranego marginesu (np. chcemy umieścić strzałkę, która musi wskazywać do wnętrz strony), to polecenie może

mieć postać `\marginpar[]{}`, gdzie parametr opcjonalny zawiera treść, jeśli notatka będzie umieszczona na lewym marginesie, a obowiązkowy treść notatki, jeśli będzie umieszczona na prawym.

14.2 Style strony

LaTeX zawiera predefiniowane style dla stron:

- `plain` – nagłówek strony (pagina górna) jest pusty, a stopka (pagina dolna) zawiera wyśrodkowany numer strony. Styl ten używany jest domyślnie;
- `headings` – nagłówek zawiera numer strony oraz tytuł, a stopka jest pusta;
- `empty` – nagłówek i stopka są puste.

Jeżeli chcemy ustawić wybrany styl używamy polecenia `\pagestyle{}`, którego parametrem jest nazwa stylu. Będzie on obowiązywał od miejsca użycia polecenia, dopóki go nie zmienimy. Jeżeli potrzebujemy zmienić styl dla pojedynczej strony, to używamy polecenia `\thispagestyle{}`. W powyższym zestawieniu pominięto styl `myheadings`. Jeżeli chcemy dostosować nagłówek i stopkę do własnych potrzeb, to najlepiej sięgnąć po pakiet `fancyhdr` [49]. Pakiet definiuje styl strony o nazwie `fancy`, który możemy łatwo modyfikować. Zacznijmy od przykładu dla wydruku jednostronnego.

```
\usepackage{fancyhdr}
```

Na listingu 14.2 pokazano kod umieszczony w preambule dokumentu klasy `article`, w którym pozostawiono domyślny jednostronny układ. Kod modyfikujący zawartość nagłówka i stopki można umieszczać wielokrotnie w dokumencie. Jeżeli zrobimy to tylko raz, jak w rozważanym przykładzie, to wszystkie strony będą używać tego samego stylu. Polecenie `\fancyhf{}` z pustym parametrem (linia 2) powoduje usunięcie wszelkich ustawień dla nagłówka i stopki („wyczyszczenie"). Polecenia `\fancyhead[]{}` i `\fancyfoot[]{}` służą do ustawiania odpowiednio zawartości nagłówka i stopki. Parametr opcjonalny wskazuje, o którą część nam chodzi: `L` – lewa strona, `C` – środek, `R` – prawa strona, a parametr obowiązkowy zawiera treść do umieszczenia w tym miejscu. W linii 3 wstawiamy po lewej stronie nagłówka symbol `\ding{97}`, który pełni rolę logo. W tym miejscu możemy też użyć polecenia `\includegraphics` i wstawić logo z pliku zewnętrznego. Po prawej stronie nagłówka umieszczamy tekst „Instytut LaTeX-a" (linia 4), a po prawej stronie w stopce słowo „Strona", po którym dodajemy numer bieżącej strony (linia 5). Ponieważ wyczyściliśmy wcześniej zawartość nagłówka i stopki, a także nic nie wstawiamy w środku w nagłówku

oraz po lewej i w środku w stopce, to części te pozostaną puste. Na koniec redefiniujemy polecenia **\headrulewidth** i **\footrulewidth**, które przechowują informacje o grubości linii oddzielającej nagłówek od treści strony i stopkę od treści strony. Jeżeli nie chcemy linii, to ustawiamy grubość na 0 pt. Efekt naszych ustawień pokazano poniżej kodu (oczywiście nie jest to pełna wysokość strony).

Listing 14.2. Definiowanie wyglądu nagłówka i stopki – wydruk jednostronny

```
\pagestyle{fancy}
\fancyhf{}
\fancyhead[L]{\huge\ding{97}}
\fancyhead[R]{\large Instytut \LaTeX-a}
\fancyfoot[R]{Strona \thepage}
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0.5pt}
```

Instytut LaTeX-a

Strona 2

W przypadku wydruku dwustronnego musimy rozróżniać strony parzyste i nieparzyste. W tym celu używane są odpowiednio litery `E` (even) i `O` (odd), a do ustawienia mamy aż 12 miejsc oznaczonych dwoma literami: `LE`, `CE`, `RE`, `LO`, `CO` i `RO`, po 6 dla nagłówka i stopki. Dodatkowo w przypadku większych opracowań często stosuje się tzw. żywą paginę, tzn. zawartość nagłówka i stopki nie jest stała (z wyjątkiem zmieniającego się numeru strony), ale umieszczamy tam informację o bieżącym rozdziale i podrozdziale (lub sekcji i podsekcji dla klasy `article`).

Informacje o bieżącym tytule na dwóch najwyższych poziomach (polecenie **\part** nie jest brane pod uwagę) LaTeX przechowuje w poleceniach **\leftmark** i **\rightmark**. Dla klas `book` i `report` **\leftmark** przechowuje nazwę rozdziału, a **\rightmark** – podrozdziału. Dla klasy `article` mamy odpowiednio sekcję i podsekcję. Warto wiedzieć, jak zachowują się te polecenia, jeżeli na przykład

w klasie `article` mamy na jednej stronie więcej niż jeden tytuł sekcji lub więcej niż jeden tytuł podsekcji. W takim przypadku **`\leftmark`** będzie zawierał ostatni tytuł sekcji na danej stronie, a **`\rightmark`** – pierwszy tytuł podsekcji. W przypadku klas `book` i `report` problem ten dotyczy tylko polecenia **`\rightmark`** (rozdziały zaczynają się od nowych stron). Będzie ono zawierało tytuł pierwszego podrozdziału na danej stronie. Zawartość poleceń **`\leftmark`** i **`\rightmark`** jest aktualizowana za każdym razem, gdy pojawia się numerowany tytuł rozdziału i podrozdziału (lub sekcji i podsekcji). Nie są aktualizowane w przypadku nienumerowanych poleceń, np. **`\chapter`**$*$. W takich przypadkach można zaktualizować wartości samodzielnie za pomocą poleceń **`\markright`**`{}` i **`\markboth`**`{}{}`. Pierwsze zmienia wartość zmiennej **`\rightmark`**, a drugie obu – pierwszy parametr to nowa wartość dla **`\leftmark`**, a drugi dla **`\rightmark`**.

Listing 14.3. Definiowanie wyglądu nagłówka i stopki – wydruk dwustronny

```
1  \renewcommand{\chaptermark}[1]{\markboth{\small\thechapter\ #1}{}}
2  \renewcommand{\sectionmark}[1]{\markright{\small\thesection\ #1}}
3
4  \pagestyle{fancy}
5  \fancyhf{}
6  \fancyhead[RO,LE]{\leftmark}
7  \fancyhead[LO,RE]{\rightmark}
8  \fancyfoot[RO,LE]{\thepage}
9  \renewcommand{\headrulewidth}{0.5pt}
10 \renewcommand{\footrulewidth}{0.5pt}
```


Na listingu 14.3 przedstawiono przykład ustawień nagłówka i stopki dla druku dwustronnego. Polecenia **`\fancyhead`** w liniach 6–7 ustawiają tytuł rozdziału po stronie zewnętrznej (prawa dla stron nieparzystych i lewa dla parzystych) oraz tytuł sekcji po stronie wewnętrznej. W stopce po stronie zewnętrznej

(linia 8) wstawiono numer strony. Domyślnie oba te tytuły będą złożone kapitalikami. Jeżeli chcemy to zmienić, to trzeba redefiniować polecenia **\chaptermark** i **\sectionmark**. W linii 1 redefiniujemy polecenie **\chaptermark**, które przyjmuje jeden parametr (`[1]`). Treścią będzie numer rozdziału + tytuł (`#1`) złożone pomniejszoną czcionką. Ten tekst jest przypisywany do zmiennej **\leftmark** przez polecenie **\markboth**. Analogicznie jest realizowana zmiana w linii 2.

14.3 Spisy

Poza spisem treści (zob. podrozdział 4.6) w większych dokumentach możemy spotkać spisy rysunków, tabel, listingów, algorytmów itp. Pakiety `titletoc` i `titlesec` pozwalają, podobnie jak w przypadku spisu treści, ustalić spójny wygląd tych części dokumentu. Jeżeli zastosowaliśmy polecenia **\titleformat** i **\titlespacing** (zob podrozdział 4.5) do ustawienia parametrów tytułów rozdziałów (przy założeniu, że używamy klasy `book` lub `report`), to ustawienia tytułów dla poszczególnych spisów powinny być spójne z ustawieniami tytułów innych rozdziałów. Pozostaje nam co najwyżej dopasowanie wyglądu wpisów na poszczególne listy. Z reguły tutaj też powinniśmy zadbać o spójny wygląd wszystkich spisów, których zdecydujemy się użyć. Różne formatowanie zastosowane w poszczególnych przykładach w tym podrozdziale ma za zadanie pokazać możliwości, a nie jest wskazówką, aby każdy spis inaczej formatować.

```
\usepackage{caption}
\usepackage{titlesec}
\usepackage{titletoc}
```

Poszczególne wspomniane wcześniej spisy generujemy poleceniami:

- **\listoffigures** – spis rysunków,
- **\listoftables** – spis tabel,
- **\lstlistoflistings** – spis listingów (pakiet `listings`),
- **\listofalgorithms** – spis algorytmów (pakiet `algorithm`).

Jeżeli chcemy dodać je wszystkie wraz z dodaniem ich do spisu treści, to np. na końcu dokumentu powinniśmy zamieścić kod z listingu 14.4.

Listing 14.4. Dodanie do dokumentu spisów rysunków, tabel, listingów i algorytmów

```
1 \addcontentsline{toc}{chapter}{Spis rysunków}
2 \listoffigures
3
4 \addcontentsline{toc}{chapter}{Spis tabel}
```

```
\listoftables

\addcontentsline{toc}{chapter}{Spis listingów}
\lstlistoflistings

\addcontentsline{toc}{chapter}{Spis algorytmów}
\listofalgorithms
```

O ile pakiet `babel` z opcją `polish` zadba o spolszczenie tytułów dla spisów treści i rysunków, to w przypadków pozostałych dwóch spisów konieczne jest wykonanie korekt w preambule. Dodatkowo, jeśli korzystamy z pakietu `caption`, to musimy zadbać, aby podpisy przy algorytmach były spójne z pozostałymi. Odpowiednie polecenia pokazano na listingu 14.5.

Listing 14.5. Korekty w preambule dla spisów listingów i algorytmów

```
1  \renewcommand{\lstlistlistingname}{Spis listingów}
2  \renewcommand{\listalgorithmname}{Spis algorytmów}
3  \floatname{algorithm}{Algorytm}
4
5  \captionsetup{font=small,labelfont=bf,labelsep=period,
6    singlelinecheck=off,format=hang}
7  \captionsetup[algorithm]{font=small,labelfont=bf,labelsep=period}
8
9  \contentsuse{lstlisting}{lol}
10 \contentsuse{algorithm}{loa}
```

Polecenia w liniach 1–2 redefiniują tytuły dla spisów listingów i algorytmów, a polecenie w linii 3 powoduje spolszczenie etykiety dla algorytmów. Jeżeli użyjemy polecenia modyfikującego sposób składania podpisów (linie 5–6), to nie zadziała ono dla algorytmów. Musimy to zrobić oddzielnie (linia 7), przy czym nie wszystkie opcje mają tutaj sens. Zmieniamy tylko rozmiar czcionki i ustawienia do składania etykiety. Ostatnie dwie linie z poleceniem `\contentsuse{}{}` powodują dodanie obsługi list listingów i algorytmów przez pakiet `titletoc`. Pakiet sam dodaje obsługę list rysunków i tabel. Pierwszym parametrem jest nazwa środowiska, a drugim nazwa listy.

Wszystkie używane tutaj listy mają swoje predefiniowane nazwy: `lof` (rysunki), `lot` (tabele), `lol` (listingi) i `loa` (algorytmy). Podczas kompilacji powstają pliki tymczasowe z takimi rozszerzeniami, które zawierają informacje o zebranych wpisach na poszczególne listy.

Niezależnie od używanej listy, jeśli podpis jest zbyt długi, to możemy zastosować parametr opcjonalny polecenia `\caption[]{}`, w którym dodajemy wersję do umieszczenia w spisie (lub nagłówku strony). Niezależnie od listy możemy użyć polecenia `\titlecontents` do ustawienia formatowania wpisu (zob. opis polecenia w podrozdziale 4.6, str. 82). Na listingu 14.6 pokazano przykład zmienionego spisu rysunków.

Listing 14.6. Ustawienia formatowania wpisów dla spisu rysunków

```
\titlecontents{figure}[2.0em]{}{\contentslabel{2.3em}}{}
  {\titlerule*[2pt]{\_}\contentspage}{}
```

Spis rysunków

1.1 Wektor zaczepiony 2
1.2 Wektory równoległe 3
1.3 Dodawanie wektorów 4
1.4 Układ współrzędnych w przestrzeni $\mathbb{R}^3$ 5
1.5 Kąt między wektorami 5

2.1 Iloczyn wektorowy 7
2.2 Iloczyn mieszany 8

Analogicznie możemy zdefiniować sposób składania wpisów dla tabel, listingów i algorytmów, zmieniając wartość pierwszego parametru. W pokazanym przykładzie bardzo gęste ułożenie znaku podkreślenia dało efekt linii ciągłej. Warto zwrócić uwagę na fakt, że polecenia `\listoffigures` i `\listoftables` wstawiają dodatkowy pionowy odstęp (10 pt) przed pierwszym wpisem z kolejnego rozdziału. Jeżeli zajrzymy do pliku `lof` lub `lot`, to łatwo zauważyć, że jest to skutek polecenia `\addvspace {10\p@ }` (`10\p@` to po protu `10pt`), którego nie znajdziemy w plikach `lol` i `loa`. Uzyskanie podobnego efektu dla pozostałych dwóch spisów wymaga więcej zabiegów – potrzebujemy np. pakietu `xpatch` [22], aby zmodyfikować wewnętrzne makra LaTeX-a. Na listingu 14.7 pokazano przykładowy kod modyfikujący spis listingów.

```
\usepackage{xpatch}
```

Listing 14.7. Ustawienia formatowania wpisów dla spisu listingów

```
1 \titlecontents{lstlisting}[2.0em]{}{\contentslabel{2.3em}}{}
2   {,\contentspage}{}
```

```
\xpretocmd{\@chapter}{\addtocontents{lol}{\protect\addvspace{10pt}}}
  {}{}
```

Spis listingów

1.1 Tworzenie ramki danych, 2
1.2 Analiza zawartości ramki danych, 3
1.3 Obliczenia statystyczne na ramce danych, 4
1.4 Filtrowanie danych, 5

2.1 Scalanie ramek danych, 7
2.2 Dodawanie i usuwanie kolumn w ramce danych, 8
2.3 Używanie funkcji *apply* i *map*, 9

W liniach 1–2 mamy ustawienie formatowania wpisów dotyczących listingów. W szóstym parametrze zrezygnowaliśmy z wypełnienia przestrzeni (mamy przecinek i numer strony). W efekcie tej zmiany numery nie są wyrównane do prawej strony. Za wstawienie wspomnianego odstępu w spisie rysunków i tabel odpowiada wewnętrzne makro LaTeX-a `\@chapter`. Chcemy je uzupełnić o dodawanie takiego samego odstępu na liście `lol`. Dodanie do definicji tego polecenia dodatkowego kodu można zrealizować za pomocą polecenia `\xpretocmd{}{}{}{}` z pakietu `xpatch`. Kluczowe są dla nas dwa pierwsze parametry – nazwa polecenia i kod, który dodajemy (pozostałe parametry zostawimy puste). Efekt tych zmian widać na wizualizacji pod listingiem 14.7.

Jeszcze więcej zmian trzeba wykonać w przypadku spisu algorytmów generowanego przez pakiet `algorithm`. Użycie polecenia `\titlecontents` z pierwszym parametrem `algorithm` nie powoduje zmian w wyglądzie listy algorytmów, która bazuje na pakiecie `float`. Jednym ze sposobów rozwiązania tego problemu jest zdefiniowanie polecenia `\listofalgorithms` jako kopii `\listoftables` i wykonanie w nim niezbędnych korekt, aby obsługiwało plik `loa`. Rozwiązanie takie przedstawiono na listingu 14.8.

```
\usepackage{tikz}
```

Listing 14.8. Ustawienia formatowania wpisów dla spisu algorytmów

```
\let\listofalgorithms\listoftables
\xpatchcmd{\listofalgorithms}{\listtablename}{\listalgorithmname}{}{}
\xpatchcmd{\listofalgorithms}{\listtablename}{\listalgorithmname}{}{}
\xpatchcmd{\listofalgorithms}{\listtablename}{\listalgorithmname}{}{}
```

```
5  \xpatchcmd{\listofalgorithms}{lot}{loa}{}{}
6
7  \xpretocmd{\@chapter}{\addtocontents{loa}{\protect\addvspace{10pt}}}
8    {}{}
9
10 \titlecontents{algorithm}[2.0em]{}{\contentslabel{2.3em}}{}
11   {\titlerule*[2.5mm]{\tikz\draw[dash dot] (0,0) node[anchor=base] {}
12   -- (0.25,0);} \contentspage}{}
```

Spis algorytmów

1.1 Hierarchiczny aglomeracyjny algorytm grupowania 2
1.2 Algorytm *k*-średnich 4
1.3 Algorytm *k*-centroidów 5
1.4 Algorytm DBSCAN 6

2.1 Algorytm konstrukcji drzewa decyzyjnego 7
2.2 Algorytm przycinania drzewa decyzyjnego 9
2.3 Algorytm AdaBoost 10

Rozpoczynamy od wspomnianej kopii polecenia **\listoftables** (linia 1). Robimy to za pomocą polecenia **\let**. Następnie korzystamy z polecenia **\xpatchcmd**{}{}{}{}{} z pakietu `xpatch`. W pierwszym parametrze podajemy nazwę polecenia, w drugim szukany ciąg znaków, a w trzecim czym on ma być zastąpiony (kolejne dwa parametry nie są tutaj ważne). Polecenie zastępuje pierwsze wystąpienie szukanego ciągu znaków. W naszym przypadku (linie 2–4) wyszukuje tekst **\listtablename** w definicji polecenia **\listofalgorithms** (nasza kopia) i zamienia na **\listalgorithmname**. Polecenie jest powtórzone trzy razy, bo szukany tekst występuje trzykrotnie w definicji modyfikowanego polecenia. Następnie polecenie **\xpatchcmd** podmienia tekst `lot` na `loa`. W liniach 7–8 mamy znaną już korektę związaną z odstępami przed pierwszym wpisem w danym rozdziale, a w liniach 10–12 polecenie **\titlecontents**, które po tych wszystkich zmianach działa bezproblemowo. Warto tutaj zwrócić uwagę na wypełnienie przestrzeni między tytułem a numerem strony. Użyte zostało polecenie **\tikz**, które kreśli linię złożoną z kresek i kropek.

14.4 Skorowidz

W przypadku książek lub większych raportów technicznych użytecznym narzędziem ułatwiającym wyszukanie informacji jest skorowidz, nazywany również indeksem. W LaTeX-u lista haseł skorowidza generowana jest przez zewnętrzny

program na podstawie haseł skorowidza zdefiniowanych w tekście. LaTeX oferuje kilka pakietów, które pozwalają na przygotowanie indeksu. Również sam indeks może być przetwarzany przez różne narzędzia zewnętrzne. Najpopularniejszą parę stanowią pakiet `makeidx` i program generujący skorowidz o nazwie `makeindex`. Niestety, jeżeli piszemy w języku polskim, to hasła skorowidza nie będą poprawnie sortowane dla słów zawierających znaki diakrytyczne. W takim przypadku lepszym wyborem jest użycie programu `xindy`, który obsługuje kodowanie `utf8` i przy podaniu odpowiednich opcji poprawnie sortuje hasła w języku polskim. W połączeniu z pakietem `imakeidx` [21], który daje możliwość zapisania w pliku źródłowym opcji przekazywanych do generatora skorowidza, para ta nadaje się do użycia w środowisku Overleaf.

```
\usepackage[xindy]{imakeidx}
\makeindex[intoc, options=-L polish -C utf8]
```

Poza dołączeniem pakietu `imakeidx` musimy w preambule (najlepiej na jej końcu) dodać polecenie **`\makeindex`**. W wersji pokazanej w ramce użyto po pierwsze opcji `xindy` dla pakietu, wskazującej program do generowania skorowidza, oraz podano dwie opcje dla polecenia **`\makeindex`**. Pierwsza (`intoc`) odpowiada za dodanie skorowidza do spisu treści, druga to opcje przekazywane do programu `xindy` – język polski i kodowanie `uft8`. Wygenerowany skorowidz dodawany jest w miejscu, w którym umieścimy polecenie **`\printindex`**.

Hasła skorowidza definiujemy z użyciem polecenia **`\index`**`{}`. Polecenie umieszczamy bezpośrednio po poprzedzającym je słowie, bez wstawiania dodatkowych spacji. Jeżeli wstawiamy kolejno więcej niż jedno polecenie **`\index`**, to nie wstawiamy między poleceniami żadnych znaków.

Jeżeli hasło skorowidza składa się z jednego słowa, to jest ono parametrem polecenia **`\index`**, np. **`\index`**`{średnia}`. Jeżeli hasło składa się z kilku słów, to dochodzą kwestie grupowania haseł. Można to robić na maksymalnie trzech poziomach. Jeżeli kolejne hasła w skorowidzu zdefiniujemy jako **`\index`**`{średnia!arytmetyczna}` i **`\index`**`{średnia!geometryczna}`, to wszystkie trzy hasła zostaną zgrupowane. Głównym hasłem jest tutaj `średnia`. Słowo to nie jest powtarzane przy kolejnych dwóch hasłach, za to słowa `arytmetyczna` i `geometryczna` są umieszczane z wcięciem. Separatorem między poziomami jest wykrzyknik. Można go użyć maksymalnie dwa razy, np. **`\index`**`{średnia!arytmetyczna!ważona}`. Jeżeli chcemy ograniczyć grupowanie do dwóch poziomów, to w ostatnim haśle zamiast drugiego wykrzyknika po prostu zostawiamy spację.

Może się zdarzyć, że chcemy wprowadzić hasło, które wymaga użycia poleceń LaTeX-a do jego wygenerowania lub sformatowania, na przykład hasłem

ma być π. W takim przypadku hasło podajemy w dwóch wersjach, które oddzielamy znakiem @. Pierwsza wersja jest używana do sortowania, a druga do umieszczenia w skorowidzu. W tym przypadku polecenie będzie miało formę `\index{pi@$\pi$}`. Podobny zabieg zastosujemy, gdy hasło zawiera obcy znak diakrytyczny, np. `\index{Cerna@\v{C}erna}` (co zostanie wyświetlone jako Černa).

Specjalne znaki LaTeX-a (zob. tabela 2.1, str. 16) indeksujemy, stosując znak ucieczki, np. `\index{\%}`, zaś znaki mające specjalne znaczenie dla indeksowania, tj. `!`, `"`, `@` i `|`, poprzedzając je znakiem `"`, np. `\index{"!}`.

W skorowidzu hasło jest wyświetlane wraz z numerami stron, na których je zaindeksowano. Jeżeli opis jakiegoś hasła obejmuje kilka stron, to możemy dodać je do skorowidza z zakresem stron, zamiast z ich indywidualnymi numerami. Początek zakresu oznaczamy, dodając do hasła `|(`, a koniec – dodając `|)`, np. `\index{wektor|(}` i `\index{wektor|)}`.

Może się zdarzyć, że do tego samego pojęcia odnoszą się dwa hasła, np. dodaliśmy do indeksu hasło wektor jednostkowy, a potem pojawia się hasło wersor, które oznacza to samo. Możemy je dodać z odniesieniem do pierwszego znaczenia, tj. `\index{wersor|see{wektor!jednostkowy}}`. W takim przypadku obok hasła wersor nie zobaczymy numeru strony, ale przekierowanie do hasła wektor jednostkowy.

LaTeX zbiera informacje o wpisach do skorowidza i odpowiednich numerach strony do pliku tymczasowego o rozszerzeniu `idx` (nazwa pliku jest taka jak nazwa pliku głównego projektu). Plik ten jest przetwarzany przez program `xindy` i generowany jest plik o takiej samej nazwie, ale rozszerzeniu `ind`. Plik ten zawiera środowisko **theindex**, którego zawartością jest poskładany skorowidz. Przy kolejnej kompilacji złożony skorowidz jest umieszczany w dokumencie. Może się zdarzyć, że wymagana jest kolejna kompilacja, aby skład był kompletny po umieszczeniu wszystkich referencji. To prowadzi to następującej sekwencji wywoływanych programów:

1) `pdflatex main.tex`
2) `texindy -L polish -C utf8 main.idx`
3) `pdflatex main.tex`
4) `pdflatex main.tex`

Listing 14.9. Fragment tekstu z dodanymi hasłami skorowidza

```
1 Dla zbioru liczby $x_1,x_2,\dots,x_n$ i wag $w_1,w_2,\dots,w_n$
2 \emph{średnia arytmetyczna ważona}\index{średnia!arytmetyczna!ważona}
3 wyliczana jest jako
4 \begin{equation}
```

```
\label{eq:srednia:arytmetyczna:wazona}
\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}.
\end{equation}
Jeśli wszystkie wagi są równe, to jest ona równa średniej
arytmetycznej\index{średnia!arytmetyczna}.
```

Na listingu 14.9 pokazano fragment tekstu z dodanymi hasłami skorowidza. Przykład skorowidza wygenerowanego przy domyślnym stylu używanym przez `xindy` pokazano na rys. 14.1. W przypadku hasła wektor widoczny jest zakres stron, a dla hasła wersor odwołanie do hasła wektor jednostkowy. Program bez problemu poradził sobie z sortowaniem haseł w języku polskim. Dla porównania `makeindex` umieszcza hasła na literę ś na końcu skorowidza.

Skorowidz

P
π, 4
początek
  układu współrzędnych, 2
  wektora, 2

Ś
średnia
  arytmetyczna, 6
    ważona, 7
  geometryczna, 6
  harmoniczna, 7

U
układ współrzędnych, 2

W
wektor, 2–4
  jednostkowy, 4
  zaczepiony, 4
  zerowy, 3
wektory
  prostopadłe, 3
  równe, 3
  równoległe, 3
wersor, *zob.* wektor, jednostkowy

Rys. 14.1. Domyślny wygląd skorowidza generowany z użyciem `xindy`

Jeżeli chcemy zmienić styl domyślny, to należy rozszerzyć wywołanie programu `texindy` o opcję `-M`, po której podajemy nazwę pliku ze stylem. Jeżeli chcemy przygotować własny styl, to wystarczy umieścić w nim opis tych elementów, które chcemy zmienić. Za punkt wyjścia można przyjąć plik `makeidx.xdy`, który znajdziemy po zainstalowaniu programu `xindy`. Na listingu 14.10 pokazano przykład pliku `myxindy.xdy`, który wykorzystano, aby uzyskać skorowidz z rys. 14.2. Linie 1–4 to redefinicja znacznika, który odpowiada za wygenerowanie początku i końca środowiska **`theindex`**. Dodane zostały dwa polecenia, które zmieniają definicje **`\subitem`** i **`\subsubitem`** – usunięte

wcięcia (w wersji domyślnej po poleceniu **\par** znajduje się **\hspace**, definiujące wielkość wcięcia). Linie 6–8 definiują początek punktu na każdym z trzech poziomów. W tym przypadku została dodana pauza na poziomie drugim (głębokość 1) i dwie pauzy na poziomie trzecim (głębokość 2). Poziom pierwszy pozostał bez zmian. W linii 10 zmieniony został znacznik rozpoczynający listę lokalizacji (numerów stron) – zamiast domyślnego przecinka używane będzie polecenie **\quad**. Linie 12–15 zawierają redefinicję wyglądu nagłówka grupy haseł zaczynających się na daną literę. W tym przypadku litera została obudowana poleceniem **\tcbox** z pakietu `tcolorbox`. Pełną dokumentację znaczników, których można używać, definiując style dla programu `xindy`, można znaleźć pod adresem `http://www.xindy.org/doc/manual.html`.

W przypadku używania TeXstudio (zob. podrozdział 15.2) domyślne ustawienia środowiska ignorują opcję `options` polecenia `makeindex`. Globalne zmienianie ustawień przez dodanie opcji `-shell-escape` ze względów bezpieczeństwa nie jest zalecane. Można to zrobić poprzez magiczny komentarz dla danego dokumentu. W tym celu dodajemy na początku dokumentu linię:
`% !TeX TXS-program:compile = txs:///pdflatex/[-shell-escape]`.
Przy pierwszej kompilacji po dodaniu tego komentarza środowisko zapyta o wyrażenie zgody. Wybieramy odpowiedź `Tak`. System zapamięta odpowiedź w postaci dodatkowego komentarza `% !TeX document-id = {...}`.

Listing 14.10. Plik ze stylem dla programu `xindy`

```
1  (markup-index :open  "~n\begin{theindex}~n \renewcommand{\subitem}
2    {\par}~n\renewcommand{\subsubitem}{\par}~n"
3                 :close "~n\end{theindex}~n"
4                 :tree)
5
6  (markup-indexentry :open "~n \item "           :depth 0)
7  (markup-indexentry :open "~n \subitem --- "      :depth 1)
8  (markup-indexentry :open "~n \subsubitem --- --- " :depth 2)
9
10 (markup-locclass-list :open "\quad{}")
11
12 (markup-letter-group :open-head "~n \tcbox[colframe=gray,
13   colback=white,left=5mm,right=5mm,boxsep=0mm,top=0mm,bottom=0mm,
14   height=6mm,valign upper=center]{\bfseries "
15   :close-head "}\nopagebreak" :capitalize)
```

Skorowidz

P

π 4
początek
— układu współrzędnych 2
— wektora 2

Ś

średnia
— arytmetyczna 6
— — ważona 7
— geometryczna 6
— harmoniczna 7

U

układ współrzędnych 2

W

wektor 2–4
— jednostkowy 4
— zaczepiony 4
— zerowy 3
wektory
— prostopadłe 3
— równe 3
— równoległe 3
wersor *zob.* wektor, jednostkowy

Rys. 14.2. Wygląd skorowidza wygenerowany z użyciem zmienionego stylu

Jeżeli chcemy skorzystać z własnego stylu, to plik z rozszerzeniem `xdy` należy umieścić w katalogu, w którym mamy główny plik projektu (w tym przypadku jest to plik `myxindy.xdy`). Polecenie **`\makeindex`** powinno wówczas przyjąć postać **`\makeindex`**`[intoc, options=-L polish -C utf8 -M myxindy]`. W przypadku środowiska Overleaf nie trzeba robić nic więcej. Przy kolejnej kompilacji skorowidz zostanie wygenerowany zgodnie z podanym stylem.

Podobnie jak w przypadku bibliografii, gdyby zaszła potrzeba drobnych ingerencji w finalną wersję skorowidza, to możemy to zrobić na etapie, gdy wiemy, że nic w układzie dokumentu już się nie zmieni. Wtedy można zakomentować polecenia **`\makeindex`** i **`\printindex`**, a w miejscu tego drugiego zamieścić zawartość pliku `ind` (środowisko **`theindex`**), np. za pomocą polecenia **`\input`**, i dokonać ręcznie wymaganych poprawek.

14.5 Definiowanie poleceń przez użytkownika

Polecenie **`\newcommand`**`{}[]{}` pozwala na definiowanie własnych poleceń użytkownika. Obowiązkowe parametry to nazwa nowego polecenia oraz jego definicja. Parametr opcjonalny używany jest, jeśli nasze polecenie ma przyjmować

obowiązkowe parametry. Wskazuje on wówczas liczbę parametrów (od 1 do 9). Brak parametru opcjonalnego oznacza, że definiujemy polecenie bezparametrowe. W najprostszym przypadku bezparametrowe polecenia zastępują nam sekwencje często wpisywanych poleceń. Załóżmy, że bardzo często wpisujemy polecenie `\mathbb{R}` na oznaczenie zbioru liczb rzeczywistych. Możemy zdefiniować krótsze polecenie, które da dokładnie ten sam efekt `\newcommand{\R}{\ensuremath{\mathbb{R}}}`. Nowe polecenie ma nazwę `\R`. Użyte w definicji polecenie `\ensuremath` gwarantuje, że parametr będzie przetwarzany w trybie matematycznym niezależnie od miejsca użycia. Teraz niezależnie, czy użyjemy naszego polecenia w trybie tekstowym `\R` ($\mathbb{R}$), czy w trybie matematycznym `$x \in \R$` ($x \in \mathbb{R}$), wszystko działa zgodnie z oczekiwaniami.

Definicje nowych poleceń umieszczamy z reguły w preambule, ale nie jest to niezbędne. W definicji polecenia możemy używać zarówno standardowych instrukcji LaTeX-a, jak też zdefiniowanych wcześniej przez nas, z wyjątkiem poleceń `\newcommand` i `\newenvironment`. W nazwach poleceń nie możemy używać znaków diakrytycznych. Nie można też doprowadzić do rekurencyjnej zależności poleceń (np. `\x` korzysta w definicji z `\y`, `\y` korzysta w definicji z `\z`, a `\z` korzysta z `\x`).

Jeżeli chcemy zdefiniować polecenia sparametryzowane, to w definicji polecenia wartości kolejnych parametrów (jeśli występują) reprezentowane są przez `#1`, `#2` itd. Załóżmy, że składamy test wielokrotnego wyboru i odpowiedzi są podawane w formie tabelki, np. do pytania o klucz w bazie danych mamy takie cztery odpowiedzi:

	a)	zbiór wszystkich atrybutów
	b)	minimalny nadklucz
	c)	jeden wybrany atrybut
	d)	minimalny identyfikujący zbiór atrybutów

Takich tabelek pojawi się w naszym teście tyle, ile jest pytań, np. 50. Czym będą się różnić? Tylko zawartością ostatniej kolumny. Wyobraźmy sobie naszą tabelkę, w której cztery odpowiedzi zostały zastąpione parametrami `#1`, `#2`, `#3` i `#4`. Wystarczy ją teraz włożyć w klamry, dodać nazwę i mamy nowe polecenie. Definicję tego polecenia pokazano na listingu 14.11. W liniach 11–15 mamy przykład wywołania polecenia `\odpowiedzi`, które generuje powyższą tabelkę. Dla wygody poszczególne parametry umieszczono w kolejnych liniach, ale nie jest to konieczne.

Listing 14.11. Przykład definicji polecenia z 4 parametrami

```
1 \newcommand{\odpowiedzi}[4]{
2 \begin{tabular}{|l|r|p{10cm}|}
```

```
\hline
\hspace{2mm} & a) & #1 \\\hline
& b) & #2 \\\hline
& c) & #3 \\\hline
& d) & #4 \\\hline
\end{tabular}
}

\odpowiedzi
{zbiór wszystkich atrybutów}
{minimalny nadklucz}
{jeden wybrany atrybut}
{minimalny identyfikujący zbiór atrybutów}
```

Istnieje możliwość zdefiniowania polecenia z jednym parametrem opcjonalnym. Na początek załóżmy, że chcemy zdefiniować polecenie `\note`, którym będziemy mogli zostawiać dla siebie rzucające się w oczy notatki typu FIXME, TODO itd. Słowo kluczowe wrzucimy sobie na jaskrawym tle, a komunikat złożymy pogrubioną czcionką. Definicję takiego polecenia pokazano na listingu 14.12.

Listing 14.12. Przykład definicji polecenia sparametryzowanego

```
\newcommand{\note}[3]
  {\colorbox{#1}{\color{white}\bfseries\textsf{#2}:} \textbf{#3}}
```

Nasze polecenie przyjmuje trzy parametry: kolor tła dla polecenia `\colorbox`, słowo kluczowe umieszczane w ramce i treść notki składaną już poza ramką. Jeżeli użyjemy go w wersji `\note{red}{TODO}{dodać przykład}`, to dostaniemy TODO: **dodać przykład**.

Załóżmy, że w większości przypadków chcemy używać ramki, która będzie miała czerwone tło. Pierwszy parametr polecenia może być opcjonalny. Definicję polecenia zapiszemy wtedy, jak pokazano na listingu 14.13. Jedyna zmiana to dodanie domyślnej wartości pierwszego parametru po nawiasie kwadratowym z liczbą parametrów.

Listing 14.13. Definicja polecenia z parametrem opcjonalnym

```
\newcommand{\note}[3][red]
  {\colorbox{#1}{\color{white}\bfseries\textsf{#2}:} \textbf{#3}}
```

Nasze nowe polecenie można zastosować z dwoma parametrami, np. `\note{TODO}{dodać przykład}` wygeneruje dokładnie taką samą notkę jak poprzednio. Możemy też wykorzystać parametr opcjonalny, np. `\note[black]{DONE}{poprawione}` wygeneruje DONE: **poprawione**.

Jeżeli próbujemy zdefiniować polecenie, które już istnieje, to użycie **\newcommand** wygeneruje błąd. W takim przypadku, jeżeli koniecznie chcemy zmienić definicję, należy użyć polecenia **\renewcommand**`{}[]{}`. Ma ono składnię taką samą jak polecenie **\newcommand**. Jeżeli nie ma bezwzględnej konieczności, powinniśmy unikać redefiniowania nie swoich poleceń.

14.6 Hiperłącza i metadane

Format `pdf` pozwala na umieszczanie w dokumentach zakładek i hiperłączy. Jeżeli chcemy dołączyć do naszego dokumentu taką możliwość, to możemy to zrobić za pomocą pakietu `hyperref`. O ile w dokumentacji jakiegoś pakietu nie wskazano inaczej, to z reguły pakiet `hyperref` dodajemy jako ostatni w preambule (wyjątkiem jest np. pakiet `algorithm`, który powinien być dodawany po `hyperref`). Pakietu nie dołączamy, jeśli robi to sama klasa dokumentu (np. `beamer`). Samo dołączenie pakietu skutkuje następującymi zmianami w pliku `pdf` (po kolejnej kompilacji): wszystkie odniesienia w dokumencie stają się łączami, tj. stają się nimi oznaczenia będące wynikiem kompilacji poleceń **\ref**, **\pageref**, **\eqref**, **\cite**, **\footnote**, **\url** itp. Ponadto łączami są wpisy w spisie treści, spisie rysunków itd. Kliknięcie dowolnego łącza przenosi nas do właściwego miejsca w dokumencie, a w przypadku łączy zewnętrznych do wskazanego pliku lub adresu.

Domyślnie wszystkie łącza są reprezentowane za pomocą kolorowych ramek wokół tekstu łącza. Dodatkowo różne grupy łączy reprezentowane są różnymi kolorami. Grupy obejmują: łącza wewnętrzne w dokumencie (kolor czerwony), łącza do bibliografii (kolor zielony), łącza do plików (kolor magenta) i łącza typu URL (kolor cyjan). Wszystkie ramki widoczne są tylko w wersji elektronicznej i nie są kreślone przy drukowaniu dokumentu. Na listingu 14.14 pokazano fragment kodu źródłowego i efekt jego kompilacji przy domyślnych ustawieniach pakietu `hyperref`. Na listingu użyto między innymi polecenia **\href**`{}{}`, które pozwala definiować dodatkowe odnośniki. Pierwszym parametrem jest adres docelowy, a drugim wyświetlany tekst.

\usepackage`{hyperref}`

Listing 14.14. Łącza wstawiane do pliku `pdf`

```
Środowisko pracy znajduje się pod adresem
\url{https://www.overleaf.com}. Po pierwszym zalogowaniu się do
systemu zobaczymy ekran startowy pokazany na rys.~\ref{fig:overleaf2}.
```

```
Poza materiałami na temat \LaTeX-a dostępnymi na stronie
\href{https://www.overleaf.com}{Overleaf} można sięgnąć po pozycje
książkowe~\cite{datta:latex:2017} lub \cite{kottwitz:latex:2021}.
```

Środowisko pracy znajduje się pod adresem `https://www.overleaf.com`. Po pierwszym zalogowaniu się do systemu zobaczymy ekran startowy pokazany na rys. 1. Poza materiałami na temat LaTeX-a dostępnymi na stronie Overleaf można sięgnąć po pozycje książkowe [1] lub [2].

Domyślne ustawienia mogą zostać zmienione za pomocą polecenia `\hypersetup{}`, które przyjmuje listę ustawień typu `klucz=wartość`. Wybrane opcje przedstawiono w tabeli 14.1.

Tabela 14.1. Wybrane opcje pakietu `hyperref`

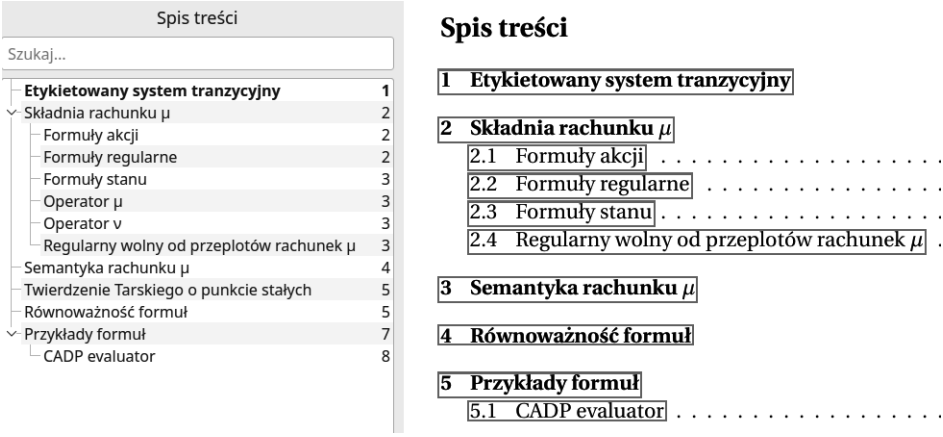
Opcja	Opis
`hidelinks`	ukrycie linków (usunięcie kolorów i ramek)
`colorlinks`	reprezentowanie linków za pomocą kolorów (zamiast ramek)
`linkcolor`	kolor wewnętrznych łączy
`citecolor`	kolor łączy do bibliografii
`filecolor`	kolor łączy do plików
`urlcolor`	kolor łączy typu URL
`hyperindex`	zamiana numerów stron w skorowidzu na łącza
`pagebackref`	dodaje listę łączy do stron z cytowaniami pozycji z bibliografii
`pdftitle`	tytuł dokumentu (metadane)
`pdfauthor`	autor dokumentu (metadane)
`pdfkeywords`	słowa kluczowe (metadane)

Użycie opcji `hidelinks` powoduje usunięcie wizualnej warstwy łączy, ale same linki nadal działają. Po ustawieniu kursora myszy nad łączem zmienia się kształt kursora i możemy kliknąć łącze. Z kolei użycie opcji `colorlinks` powoduje zastąpienie ramek kolorami linków (np. przy domyślnych kolorach cały spis treści wyświetlany jest na czerwono). Kolory można zmienić za pomocą opcji podanych w tabeli. Kolorowe łącza są widoczne na wydruku dokumentu. Opcje, które przyjmują wartości logiczne, możemy zapisywać jako np. `hyperindex=true` lub równoważnie jako `hyperindex` (w takim przypadku `True` jest domyślną wartością). Opcje o wartościach logicznych to również: `hidelinks`, `colorlinks` i `pagebackref`. Użycie opcji `pagebackref` powoduje dodanie po każdej pozycji bibliograficznej listy numerów stron, na których jest cytowana. Numery te są łączami.

Większości opcji można używać również jako opcji pakietu. Jeżeli opcja `pagebackref` nie działa z daną klasą po umieszczeniu jej w parametrze polecenia **`\hypersetup`**, to warto sprawdzić jej działanie po przeniesieniu do opcji pakietu, tj. **`\usepackage`**`[pagebackref]{hyperref}`.

Poza obsługą hiperłączy pakiet `hyperref` pozwala dodać do pliku `pdf` metadane, jak na przykład imię i nazwisko autora, tytuł dokumentu lub słowa kluczowe. Informacje takie mogą być szczególnie użyteczne, jeżeli chcemy umieścić nasz dokument w internecie. Metadane są używane między innymi przez wyszukiwarki internetowe. Są one również dostępne w przeglądarkach plików `pdf`, jeśli wyświetlimy właściwości dokumentu. Metadane dodajemy jako kolejne opcje w poleceniu **`\hypersetup`**. Najczęściej używane opcje zamieszczono w tabeli 14.1.

W efekcie użycia pakietu `hyperref` automatycznie definiowane są zakładki na podstawie wpisów tworzących spis treści. Możemy uzupełnić tę listę o zakładki definiowane za pomocą polecenia **`\pdfbookmark`**`[]{}{}`. Pierwszy parametr obowiązkowy zawiera treść zakładki, a drugi jej unikatową nazwę (analogicznie do etykiet). Opcjonalny parametr wskazuje poziom (w sensie struktury dokumentu – zob. podrozdział 4.6), na którym tworzona jest zakładka. Domyślnie jest to poziom 0, czyli w przypadku klas `book` lub `report` zakładki są dodawane na poziomie rozdziałów. Zakładki możemy wykorzystać, aby wskazać istotne elementy dokumentu (tabele, algorytmy, wykresy itp.).



Rys. 14.3. Widok zakładek w przeglądarce plików `pdf`

Treść zakładki jest traktowana jako czysto tekstowa, nie możemy w niej użyć na przykład wzorów matematycznych. Problem ten dotyczy również zakładek

tworzonych automatycznie. Jeżeli np. tytuł sekcji zawiera znaki specjalne, to zostaną one pominięte przy tworzeniu treści odpowiedniej zakładki. Częściowym rozwiązaniem tego problemu jest użycie polecenia `\texorpdfstring{}{}`. Jego parametrami są tekst używany przez LaTeX-a i jego alternatywna wersja dla zakładek w pliku `pdf`. Polecenia można użyć w tytułach rozdziałów, sekcji itd., aby wskazać, co ma zostać wstawione do zakładki w miejsce pomijanych fragmentów. Przy domyślnych ustawieniach w zakładkach można używać znaków unikodu, co wykorzystano w przykładach z rys 14.3, aby wstawić do zakładek greckie litery. Na przykład tytuł drugiej sekcji został złożony jako: `\section{Składnia rachunku \texorpdfstring{$\mu$}{μ}}`, gdzie jako drugi parametr wklejono znak unikodowy. Podobnie rozwiązano problem z dodatkowymi zakładkami, przy czym tutaj znaki unikodu możemy wstawiać bezpośrednio w treści zakładki, np. `\pdfbookmark[2]{Operator μ}{operator:mu}`.

Rozdział 15

Środowisko pracy

Jak wspomniałem we wstępie do tej książki, dokument LaTeX-a możemy przygotować w dowolnym edytorze tekstu. Jeżeli chcemy pracować wygodnie, to minimum wsparcia, jakie powinno udostępniać środowisko pracy, stanowi podświetlanie składni i wygodna współpraca z narzędziami takimi jak `pdflatex`, `bibtex` itp. W tym rozdziale nieco bliżej przyjrzymy się środowisku Overleaf przedstawionemu wstępnie w podrozdziale 1.2 oraz omówię jedno z popularniejszych środowisk instalowanych lokalnie – TeXstudio.

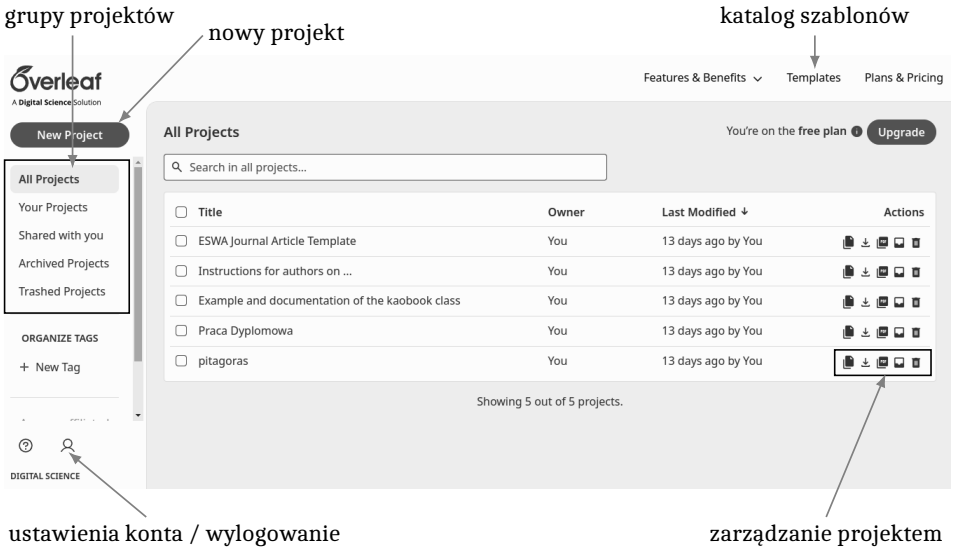
15.1 Overleaf

Krótkie wprowadzenie do systemu Overleaf przedstawiono w podrozdziale 1.2. Podstawową zaletą środowiska jest brak konieczności instalowania oprogramowania na własnym komputerze (o ile mamy już którąś z popularnych przeglądarek internetowych). Już darmowa wersja systemu oferuje rozbudowany edytor plików źródłowych, dostęp do dystrybucji TeX Live (zob. podrozdział 15.2), wbudowaną przeglądarkę plików `pdf`, nieograniczoną liczbę projektów i możliwość współpracy nad projektem z drugim użytkownikiem. W wersji płatnej m.in. mamy możliwość współpracy z nieograniczoną liczbą współpracowników, dostęp do historii dokumentu, współpracę z Dropboxem i GitHubem[1]. Niestety darmowa wersja ma również swoje ograniczenia. Jednym z nich jest limit czasu kompilacji projektu. Przy większych projektach będzie on niewystarczający i nie

[1] Niektóre uczelnie oferują swoim pracownikom i studentom dostęp do pełnej wersji Overleafa. Są to np. Akademia Górniczo-Hutnicza w Krakowie, Politechnika Gdańska i Politechnika Wrocławska.

uda nam się skompilować dokumentu. Problem taki może pojawić się już przy książce mającej kilkadziesiąt stron, jeśli kompilacja jest wymagająca.

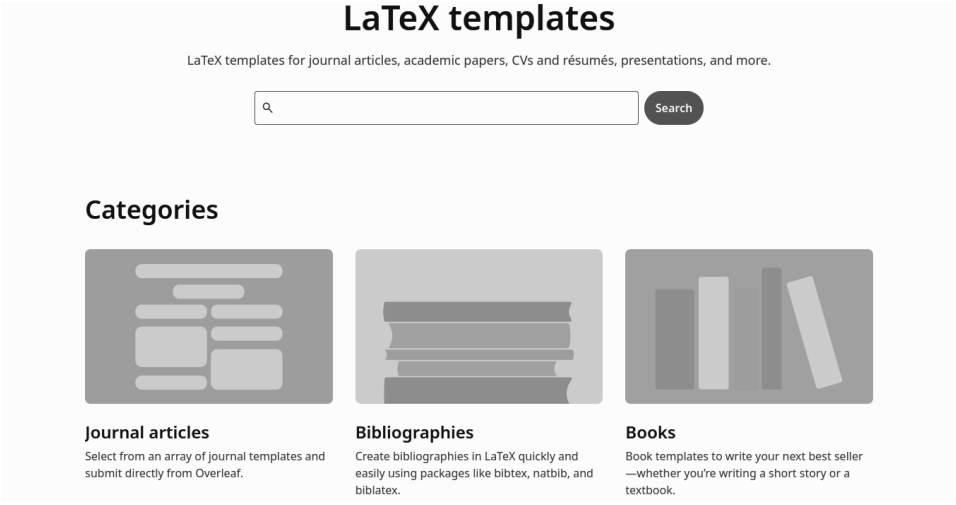
Sposób tworzenia pustego dokumentu pokazałem już w podrozdziale 1.2. Tym razem zaczniemy od bardziej zaawansowanych projektów. Rozpoczynamy od poziomu zarządzania projektami (zob. rys. 15.1) – jeśli masz otwarty projekt, to wystarczy kliknąć ikonę z domkiem, aby do niego przejść. Po wybraniu przycisku `Templates` przejdziemy do katalogu szablonów dokumentów (zob. rys. 15.2). Jeżeli wiemy, czego konkretnie szukamy, możemy skorzystać z przycisku `Search`, np. wpisując nazwę czasopisma „Expert Systems with Applications", bez problemu znajdziemy szablon artykułu. Klikając obrazek przykładowej strony, możemy zobaczyć szczegóły szablonu, w tym podejrzeć źródła lub plik `pdf`, ale przede wszystkim możemy wybrać przycisk `Open as Template` i po chwili mamy na swoim koncie nowy projekt utworzony z tego szablonu. Nowy projekt to nie tylko plik `tex`, na liście plików możemy znaleźć m.in. plik z klasą `elsarticle.cls`, style dla BibTEX-a, przykładowy plik z bibliografią, katalog `figures` na rysunki itd.



Rys. 15.1. Overleaf – poziom zarządzania projektami

Przy przeglądaniu szablonów możemy również skorzystać z podziału na kategorie – po wybraniu kategorii `Thesis` znajdziemy prawie 300 stron z szablonami prac dyplomowych lub doktorskich (również z polskich uczelni). Na dole strony mamy etykiety powiązane z szablonami. Po kliknięciu przycisku `Show more` możemy odszukać i wybrać etykietę `Polish`. Widzimy kilka szablonów prac

dyplomowych. Jeżeli jest wśród nich ten, którego szukamy, to postępujemy podobnie jak przed chwilą z czasopismem ESA.



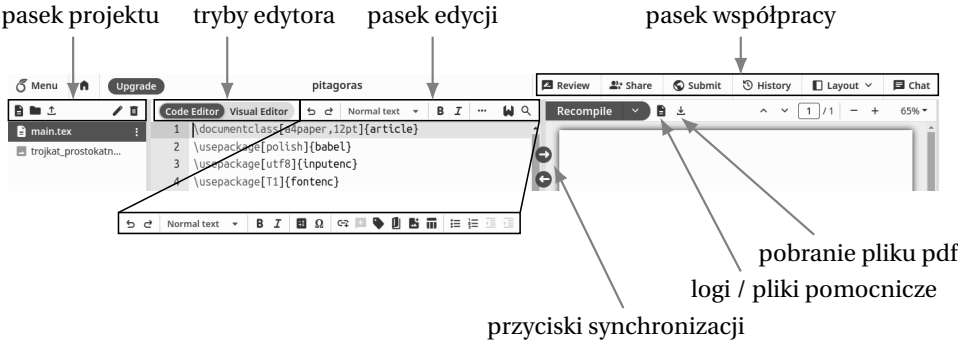
Rys. 15.2. Overleaf – katalog szablonów

Innym miejscem na poszukiwanie szablonów dokumentów LaTeX-a jest strona `https://latextemplates.com`. Tutaj również katalog pogrupowany jest według kategorii. Jeżeli znajdziemy interesujący nas szablon, to możemy go dodać do naszego konta Overleaf jako nowy projekt, korzystając z przycisku „Open in Overleaf”.

W wielu przypadkach może się okazać, że naszego szablonu musimy szukać na stronie czasopisma. W ramach przykładu pobierzemy szablon ze strony `https://www.amcs.uz.zgora.pl/`. Znajdziemy go po wybraniu w menu opcji `guide`. Pobrany szablon ma formę archiwum `zip`. Na poziomie zarządzania projektami (zob. rys. 15.1) wybieramy `New project`, a następnie z listy rozwijalnej `Upload project`. Do okienka, które się wyświetli, przeciągamy pobrany plik `zip` i po chwili nowy projekt czeka na wypełnienie naszą treścią. W ten sposób możemy też przenieść dowolny nasz projekt do środowiska Overleaf. Jeżeli nie chcemy dodawać plików pojedynczo, wystarczy spakować je do formatu `zip`.

Zakładając, że mamy już otwarty interesujący nas projekt, przyjrzymy się bliżej pracy z edytorem Overleaf. Po pierwsze po dodaniu nowego projektu powinniśmy wybrać język dla programu sprawdzającego ortografię. Wyświetlamy w tym celu główne menu (zob. rys. 1.5, str. 13) i ustawiamy opcję `Spell check`. W tym miejscu warto też ustawić parametry pracy edytora: rozmiar czcionki, czcionkę dla kodu źródłowego czy wreszcie motyw edytora, zwłaszcza jeżeli lubimy pracować z ciemnym tłem. Dodatkowo korzystając z przycisku `Layout`

(pasek współpracy, zob. rys. 15.3), możemy wybrać układ środowiska. Zamiast domyślnego podziału okna na pół możemy na przykład podgląd `pdf` otworzyć w oddzielnej zakładce (przydatne, jeżeli pracujemy na małym ekranie).



Rys. 15.3. Overleaf – paski narzędziowe

Wracając do pracy z edytorem, mamy do dyspozycji dwa tryby (zob. rys. 15.3). Po pierwsze jest to typowy edytor kodu źródłowego (`Code Editor`), po drugie mamy dostęp do edytora określanego jako `Visual Editor`. Nie jest to dosłownie tryb WYSIWYG (What You See Is What You Get), ale raczej znany z edytora LyX tryb WYSIWYM (What You See Is What You Mean). Tryb ten może być dużo bardziej przyjazny dla osób, które nieco przeraża praca z czystym kodem źródłowym. Między oboma trybami można się w dowolnym momencie przełączać. Praca w trybie wizualnym nie oznacza tutaj całkowitego wyeliminowania poleceń LaTeX-a, ale raczej automatyczną wizualizację większości poleceń po ich wprowadzeniu, np. jeżeli wybierzemy wstawienie nowej sekcji, to zobaczymy polecenie `\section{}`, ale po edycji tytułu będzie wyświetlany tylko sformatowany tekst. Podobnie, gdy chcemy wstawić wzór matematyczny, to musimy go edytować w klasyczny sposób, ale po zakończeniu edycji zobaczymy skompilowaną wersję.

Pasek edycji (zob. rys. 15.3) funkcjonuje w obu trybach. Znajdziemy na nim kilkanaście przycisków związanych z często powtarzanymi akcjami. Są to np. wstawienie polecenia dodającego pogrubienie tekstu lub kursywę, przejście do trybu matematycznego, wstawianie szkieletów list, wstawianie cytowań lub odwołań do etykiet, uruchamianie kreatora tabel, uruchamianie kreatora wstawienia grafiki itp. Nie wszystkie narzędzia pracują w trybie darmowym, niedostępne jest na przykład wyszukiwanie symboli czy też zaawansowane narzędzia do wyszukiwania bibliografii. W wersji pełnej przycisk wstawiający symbole udostępnia paletę nieco ponad 100 najpopularniejszych symboli (w tym litery greckie). Wstawienie polecenia sprowadza się do kliknięcia ikony.

Kreator wstawiania rysunków umożliwia przy okazji dodanie nowego rysunku do projektu. Możemy go wczytać z własnego komputera, z innego projektu, a nawet pobrać z internetu, wskazując adres strony internetowej. Kreator wstawiania tabel pozwala wyłącznie wskazać rozmiar docelowy tabeli i generuje pusty jej szkielet.

Tabela 15.1. Overleaf – ważniejsze skróty klawiaturowe

Skrót	Działanie
`Ctrl + Enter` lub `Ctrl + S`	kompiluj projekt
`Ctrl + Z`	cofnij
`Ctrl + Y`	ponów
`Ctrl + A`	zaznacz wszystko
`Ctrl + C`	kopiuj
`Ctrl + X`	wytnij
`Ctrl + V`	wklej
`Ctrl + F`	szukaj i zastąp
`Ctrl + /`	zakomentuj/odkomentuj
`Ctrl + D`	usuń
`Ctrl + Shift + D`	powiel
`Alt + ↑`	przesuń w górę
`Alt + ↓`	przesuń w dół
`Ctrl + I`	kursywa
`Ctrl + B`	pogrubienie tekstu
`Ctrl + Spacja`	otwarcie autouzupełniania

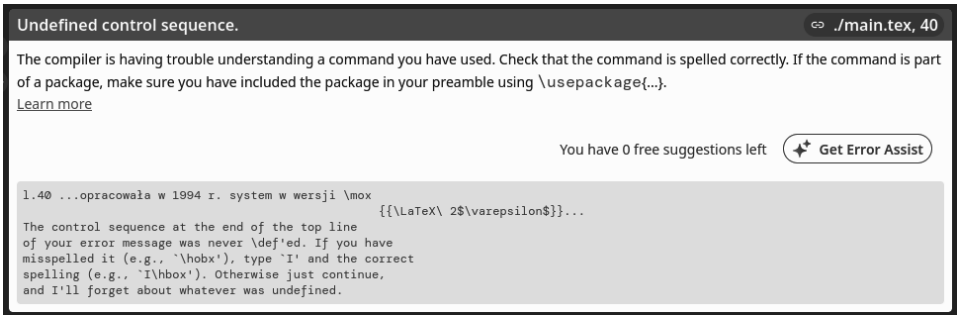
Pasek edycji wspierany jest przez skróty klawiaturowe. Najważniejsze z nich pokazano w tabeli 15.1. Teoretycznie można wyświetlić zestawienie skrótów z poziomu menu głównego (`Show Hotkeys`), ale nie wszystkie tam pokazane działają (widać konflikt z przeglądarką), a po drugie nie wszystkie działające są tam udokumentowane. W większości polecenia działają po zaznaczeniu co najmniej fragmentu tekstu, ale `Ctrl + /`, `Ctrl + D`, `Ctrl + Shift + D`, `Alt + ↑`, `Alt + ↓` przy braku zaznaczenia działają na bieżącej linii.

Warto zwrócić uwagę na autouzupełnianie. Wywołane podczas wprowadzania tekstu udostępnia listę poleceń LaTeX-a. Jeżeli jesteśmy wewnątrz nawiasów klamrowych polecenia `\ref` (`\pageref`, `\eqref`), to wyświetli listę etykiet, a dla polecenia `\cite` listę etykiet pozycji bibliograficznych. Jeżeli sami chcemy wywołać autouzupełnianie, należy skorzystać z kombinacji `Ctrl+Spacja`.

Panel po lewej stronie podzielony jest na dwie części (zob. rys. 1.3 str. 11). W górnej części wyświetlone są pliki wchodzące w skład projektu. Pasek projektu (zob. rys. 15.3) pozwala łatwo tworzyć nowe pliki, dodawać, usuwać,

zmieniać ich nazwy lub organizować je w podkatalogi. Jeżeli chcemy przełączyć się na inny plik, wystarczy go kliknąć. W dolnej części, zatytułowanej `File outline`, wyświetlana jest drzewiasta struktura bieżącego pliku. Jest ona aktualizowana po dodaniu poleceń typu **`\chapter`**, **`\section`**, **`\subsection`** itd. Kliknięcie dowolnego elementu drzewka przeniesie nas we właściwe miejsce w dokumencie. Dodatkowo, jeżeli pracujemy w trybie podzielonego ekranu, to w górnej części linii oddzielającej edytor od podglądu `pdf` dostępne są dwa przyciski w formie strzałek. Służą one do synchronizacji obu widoków (zob. rys. 15.3). Jeżeli zaznaczymy fragment tekstu w pliku źródłowym i klikniemy strzałkę w prawo, to na podglądzie `pdf` zobaczymy ten sam fragment. Synchronizacja w drugą stronę działa jeszcze prościej – wystarczy dwukrotnie kliknąć jakieś słowo w pliku `pdf`, a w edytorze kursor ustawi się na początku odpowiedniego akapitu.

Jeżeli w wyniku kompilacji pojawiają się błędy, to jest to sygnalizowane informacją o liczbie błędów wyświetlaną na czerwonym tle na przycisku `Logs and output files` (logi / pliki pomocnicze – zob. rys. 15.3). Kliknięcie przycisku wyświetli komunikaty z ostatniej kompilacji. Przykład komunikatu o błędzie pokazano na rys. 15.4. W prawym górnym rogu znajduje się wskazanie, w którym pliku i w której linii znaleziono błąd. Jest to informacja orientacyjna. Kliknięcie tego przycisku przeniesie nas we właściwe miejsce w kodzie. Pokazany błąd związany jest z literówką w poleceniu LaTeX-a. Inny typowy powód nierozpoznania polecenia (jeśli jest zapisane poprawnie), to brak dołączonego odpowiedniego pakietu. Po poprawieniu danego błędu warto ponowić kompilację. Czasami błędy narastają kaskadowo i jedna poprawka może dużo zmienić.



Rys. 15.4. Komunikat o błędzie kompilacji

Jedną z zalet środowiska Overleaf jest możliwość współpracy w zespole bez potrzeby korzystania na przykład z oprogramowania do wersjonowania. W wersji darmowej jest to ograniczone do dwuosobowych zespołów. W wersji płatnej nie ma tego limitu. Jeżeli chcemy udostępnić komuś nasz projekt, należy

skorzystać z przycisku Share (pasek współpracy, zob. rys. 15.3) i podać adres e-mail, którego nasz współpracownik używa w systemie Overleaf.

Podczas wspólnej pracy nad projektem Overleaf pokazuje ikony zalogowanych aktualnie osób, które mają otwarty ten sam projekt. Z poziomu systemu można korzystać z czatu (przycisk Chat – pasek współpracy). Współpracownicy mogą komentować tekst – jeżeli tylko zaznaczymy fragment tekstu, pojawi się dymek z poleceniem Add comment. Zalogowani użytkownicy od razu widzą komentarze dodane przez inne osoby. Jeżeli włączymy przycisk Review, to zobaczymy dodatkowy panel z wszystkimi komentarzami. Pozwala to w wygodny sposób na nie odpowiadać. Jeżeli mamy dostęp do pełnej wersji środowiska, to możemy też śledzić historię zmian (przycisk History). System pokaże, kto, kiedy i jakie zmiany wprowadził w projekcie. Możemy też przywrócić wybraną wcześniejszą wersję dokumentu.

Jeżeli nasz dokument jest gotowy lub po prostu chcemy pobrać na lokalny dysk aktualną wersję, to mamy kilka możliwości, zależnie od tego, czego szukamy. Jeżeli chcemy pobrać wyłącznie plik pdf, to możemy skorzystać z przycisku pobrania pliku pdf (zob. rys. 15.3). Ten sam efekt uzyskamy, wybierając przycisk PDF z menu głównego. Przycisk Source w menu głównym służy do pobrania plików źródłowych projektu spakowanych do archiwum w formacie zip. Zarówno plik pdf, jak i archiwum z plikami źródłowymi możemy pobrać również z poziomu zarządzania projektami. Jeżeli chcemy pobrać pliki będące wynikiem kompilacji, to wybieramy przycisk Logs and output files (logi / pliki pomocnicze – zob. rys. 15.3) i listę dostępnych plików wyświetlamy przyciskiem Other logs and files, który znajduje się w prawym dolnym rogu. Wybranie dowolnego z plików rozpoczyna proces pobierania. W tym miejscu dostępny jest również przycisk Clear cached files, którym możemy usunąć te pliki.

Na pasku współpracy znajduje się jeszcze jeden przycisk – Submit. Po jego wybraniu mamy możliwość wysłać nasz artykuł do znajdujących się na liście czasopism lub repozytoriów online. Możemy również dodać nasz projekt jako nowy szablon do galerii Overleafa.

15.2 TeXstudio

Jeżeli nie do końca odpowiada nam praca z Overleaf (wymaganie ciągłego dostępu do internetu, dokumenty przechowywane w chmurze, potrzebujemy bardziej zaawansowanego środowiska, chcemy kompilować większe projekty, a nie mamy dostępu do pełnej wersji itd.) i mamy możliwość instalacji oprogramowania na komputerze, na którym chcemy pracować, to warto takie oprogramowanie zainstalować. W tym rozdziale proponowany jest zestaw TeXstudio + TeX

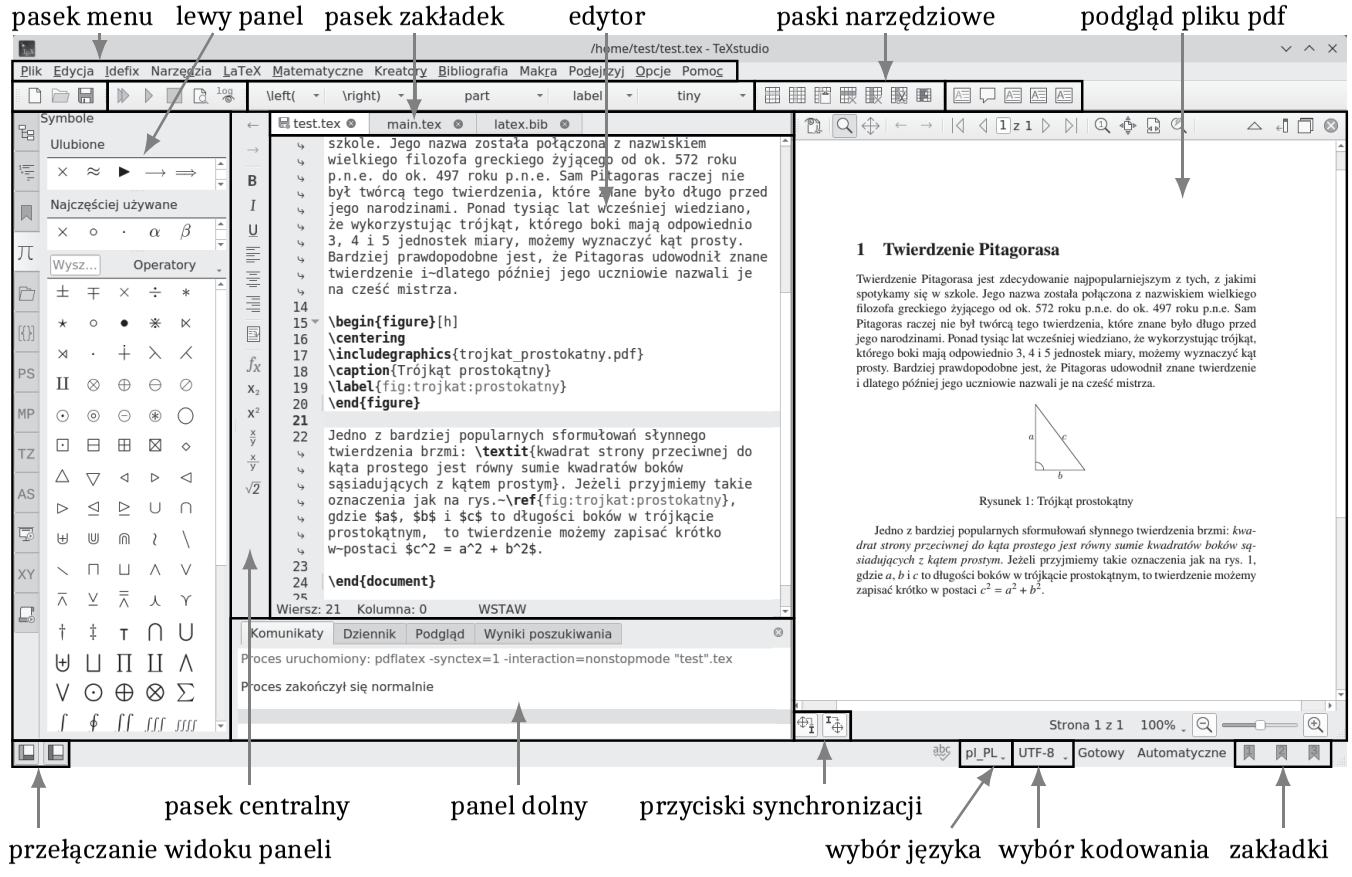
Live, ale równie dobrze można wybrać inną dystrybucję LaTeX-a. Oba narzędzia są całkowicie darmowe, intensywnie rozwijane, bez problemu można uzyskać wsparcie od społeczności użytkowników w przypadku jakichkolwiek kłopotów.

TeXstudio `https://texstudio.org` jest zaawansowanym środowiskiem do pracy z LaTeX-em. Jest to oprogramowanie typu *open source*. Środowisko bez problemu integruje się z narzędziami LaTeX-a, udostępnia edytor z podświetleniem składni, wbudowaną przeglądarkę plików `pdf`, rozbudowane wsparcie do wstawiania poleceń LaTeX-a, umożliwia tworzenie makr użytkownika itd. TeXstudio dostępne jest dla różnych systemów uniksowych (w tym dystrybucje Linuksa), macOS i Windows. Wersję instalacyjną można pobrać z podanej strony internetowej. Na stronie można też znaleźć informacje na temat instalacji. W przypadku popularnych dystrybucji Linuksa (Ubuntu, Mint, Debian, Red Hat, Fedora, SUSE) TeXstudio dostępne jest w repozytoriach systemowych, ale jeśli chcemy korzystać z najnowszej stabilnej wersji, to z podanej strony można pobrać aplikację w formacie `AppImage`.

Samo środowisko TexStudio udostępnia wygodne narzędzie do edycji plików `tex` i `bib` oraz podglądu plików `pdf`. Jeżeli chcemy mieć możliwość kompilowania naszych dokumentów, to musimy zainstalować również jakąś dystrybucję LaTeX-a. Istnieje kilka wiodących dystrybucji. Ze swojej strony polecam TeX Live `https://tug.org/texlive/`. Dystrybucja ta jest dostępna dla różnych systemów uniksowych, włączając w to Linuksa, dla macOS i Windows. Z podanej strony internetowej można pobrać wersję instalacyjną, dostępna jest również szczegółowa instrukcja instalacji dla poszczególnych systemów operacyjnych. Podobnie jak TeXstudio, w przypadku popularnych dystrybucji Linuksa TeX Live dostępny jest w repozytoriach systemowych.

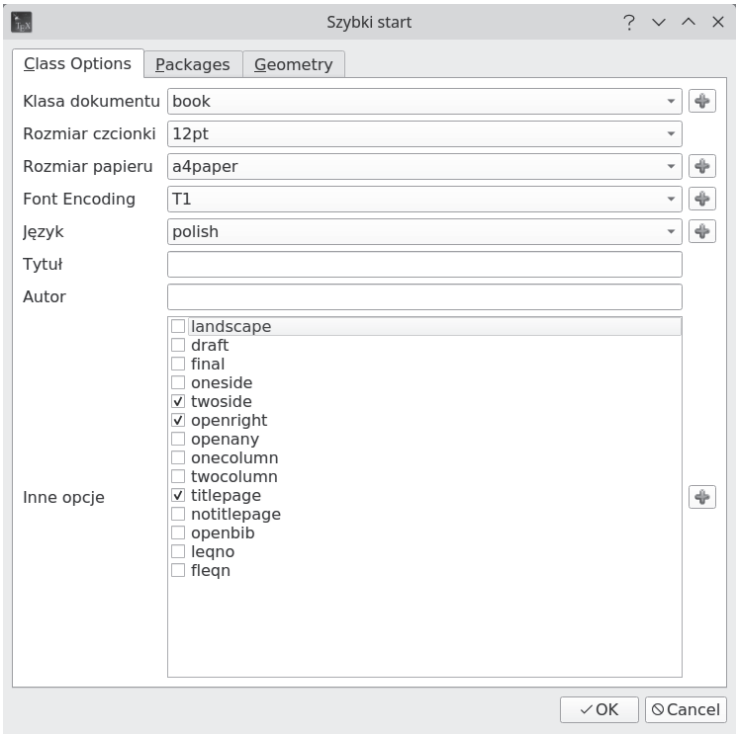
Widok środowiska TeXstudio z otwartym przykładem z rozdziału 1 przedstawiono na rys. 15.5. Okno programu podzielone jest na dwie główne części, podobnie jak w przypadku Overleafa, po lewej stronie mamy edytor kodu źródłowego, a po prawej podgląd pliku `pdf`. Istnieje możliwość przełączenia środowiska w tryb dwóch oddzielnych okien, jednego z edytorem, drugiego z podglądem pliku `pdf`. Niezależnie od trybu oba widoki świetnie się synchronizują – wystarczy przytrzymać wciśnięty klawisz `Ctrl` i kliknąć lewym przyciskiem myszy w jednym widoku, a drugi widok dopasuje wyświetlanie. Ponadto przyciski synchronizacji (zob. rys. 15.5) umożliwiają włączenie automatycznej synchronizacji obu widoków. Włączenie obu przycisków powoduje automatyczną synchronizację w obu kierunkach.

Nowy dokument możemy utworzyć na kilka sposobów. Po pierwsze możemy otwarty już dokument zapisać pod nową nazwą i modyfikować jego zawartość. Po drugie możemy utworzyć całkowicie pusty dokument i samodzielnie edytować go od początku. TeXstudio pozwala również utworzyć nowy dokument



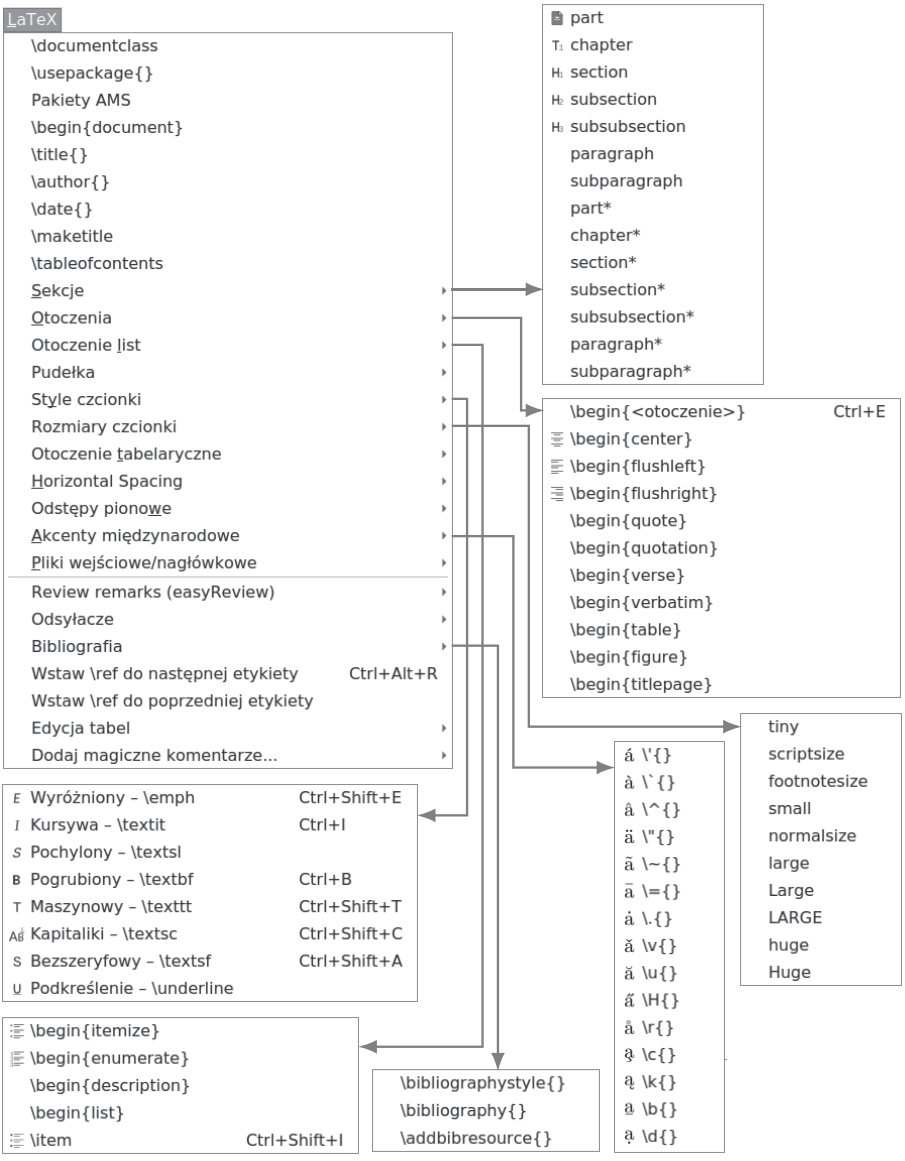
Rys. 15.5. Środowisko TeXstudio z otwartym plikiem źródłowym i podglądem pdf

na podstawie kilkunastu szablonów dostępnych po wybraniu polecenia z menu `Plik→Nowy z szablonu...` (można dodawać własne szablony na podstawie istniejących dokumentów). Najciekawszym rozwiązaniem jest kreator `Kreatory→Szybki start...` pokazany na rys. 15.6. W pierwszej zakładce mamy możliwość m.in. wyboru klasy dokumentu i powiązanych z nią opcji, druga zakładka pozwala odhaczyć pakiety, które chcemy dodać do preambuły (lista pakietów do wyboru może być edytowana przez użytkownika), a trzecia zakładka pozwala ustawić opcje pakietu `geometry`, definiujące szerokości marginesów oraz szerokość i wysokość tekstu. Na podstawie ustawień generowany jest dokument z wstępnie gotową preambułą i pustym środowiskiem **`document`**.



Rys. 15.6. Kreator `Szybki start...`

W edytorze możemy mieć otwartych jednocześnie kilka dokumentów. Pasek zakładek (zob. rys. 15.5) pozwala na szybkie przełączanie się między nimi. Edytor sam rozpoznaje kodowanie pliku, a informacja o kodowaniu jest wyświetlana na przycisku wyboru kodowania (zob. rys. 15.5). Kliknięcie przycisku wyświetla listę dostępnych kodowań i umożliwia zmianę kodowania dla bieżącego dokumentu. Po sąsiedzku znajduje się przycisk wyboru języka. Ustawienie opcji innej niż `none` włącza sprawdzanie pisowni w wybranym języku.

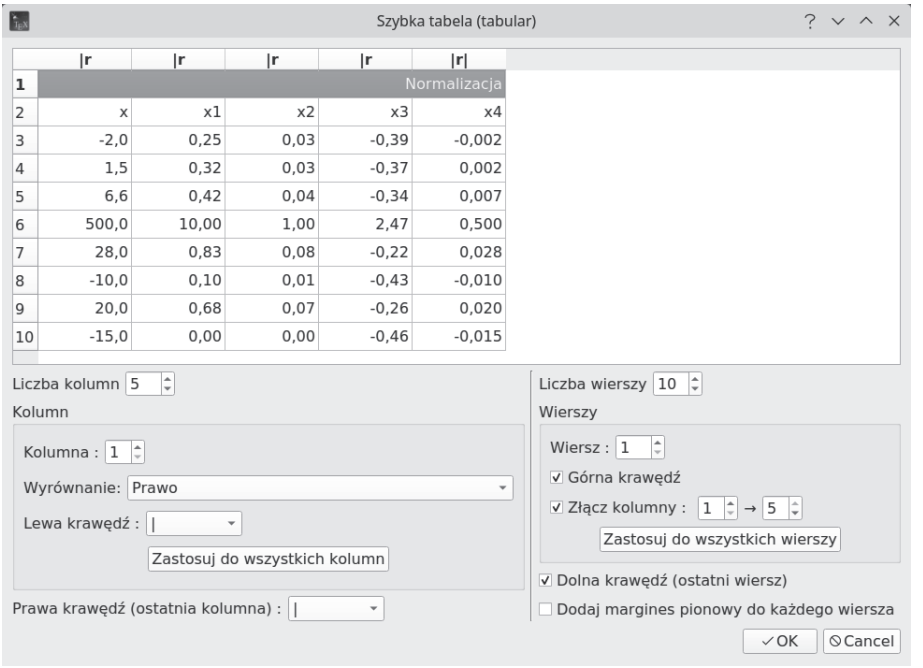


Rys. 15.7. Menu LaTeX z rozwiniętymi przykładowymi podmenu

TeXstudio stara się maksymalnie ułatwiać edycję plików LaTeX-a. W menu LaTeX i Matematyczne zgromadzono ponad 200 poleceń LaTeX-a, które wstawiamy do kodu źródłowego kliknięciem myszki. Na rys. 15.7 pokazano menu LaTeX z rozwiniętymi wybranymi podmenu. Paski narzędziowe (na rys. 15.7 widzimy centralny i poziome) pozwalają na szybki dostęp do kilkudziesięciu poleceń

LaTeX-a. Są one w pełni konfigurowalne. Użytkownik może dowolnie skomponować ich zawartość, umieszczając na nich dowolne polecenia dostępne przez menu główne. Użytkownik może również sam ustawić sobie klawisze skrótu do najczęściej używanych poleceń. Uzupełnieniem dla menu i pasków narzędziowych jest czwarta od góry zakładka w lewym panelu (zob. rys. 15.5). Zawiera on ponad tysiąc symboli matematycznych i znaków specjalnych. Są one pogrupowane w kilkanaście palet, co ułatwia wyszukiwanie. Dodatkowo system zapamiętuje najczęściej używane oraz pozwala zorganizować paletę ulubionych symboli. Ponadto na szóstej zakładce zebrano zestaw ograniczników.

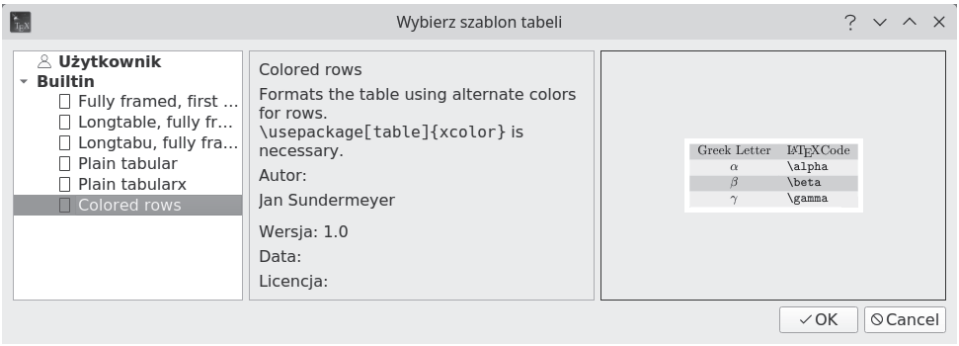
Menu `Kreatory` poza przedstawionym już kreatorem `Szybki start...` oferuje m.in. przygotowanie kompletnej tabeli wraz z zawartością komórek w środowiskach **`tabular`** i **`array`** oraz wygodne dodawanie grafiki do dokumentu. Kreatory te są zdecydowanie bardziej zaawansowane niż te dostępne w środowisku Overleaf.



Rys. 15.8. Kreator `Szybka tabela (tabular)...`

Widok kreatora `Szybka tabela (tabular)...` przedstawiono na rys. 15.8. Pracę rozpoczynamy od ustawienia liczby kolumn i wierszy. Dane można edytować bezpośrednio w komórkach w kreatorze lub skopiować np. z arkusza kalkulacyjnego. Kreator pozwala na ustawienie standardowych typów kolumn,

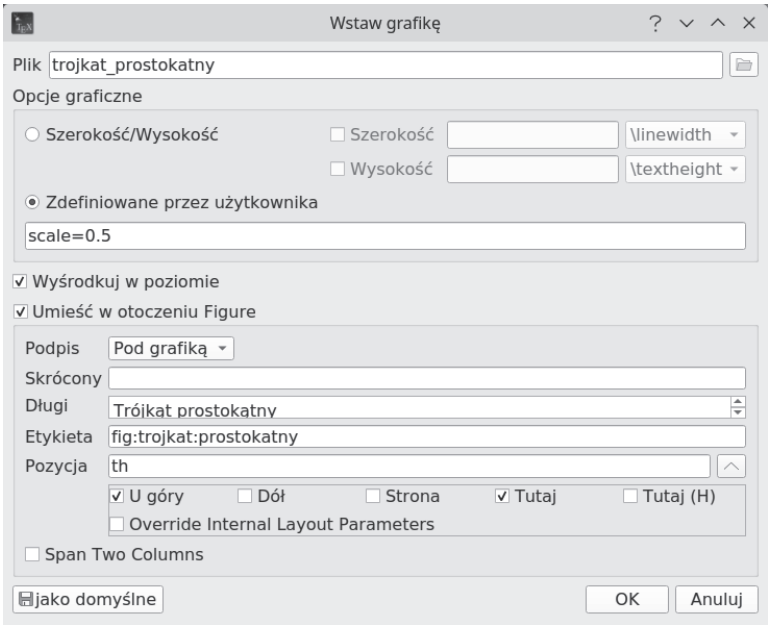
dodanie pionowych i poziomych linii, a także na scalanie komórek w poziomie. W pokazanym przykładzie do poprawy w kodzie źródłowym został tylko sposób wyrównania w scalonej komórce (na wyśrodkowanie). Cała reszta została wygenerowana przez kreator. Ewentualna modyfikacja struktury tabeli po jej wygenerowaniu musi być prowadzona już w kodzie źródłowym. W menu `LaTeX→Edycja tabel` (a także na domyślnym pasku narzędziowym – piąty od prawej na rys. 15.5) dostępne są polecenia pozwalające: dodać lub usunąć wiersz i kolumnę, wycinać i wklejać kolumny, dodawać i usuwać linie poziome oraz przeformatować kod źródłowy tabeli, tak aby wyrównać kolumny. Jeżeli tabela nie ma skomplikowanej struktury (scalane komórki), to operacje te działają bez zarzutu. Dodatkowo można uruchomić polecenie do formatowania tabeli według wzorca, które pokazano na rys. 15.9 (ustawiamy kursor gdzieś w środku kodu tabeli i wybieramy polecenie `LaTeX→Edycja tabel→Przekształć tabelę zgodnie z wzorcem`). Istnieje możliwość wybrania jednego z sześciu predefiniowanych wzorców, a także definiowania własnych.



Rys. 15.9. Formatowanie tabeli według wzorca

Na rys. 15.10 pokazano kreator umożliwiający wstawianie grafiki. Można go uruchomić z menu `Kreatory`, a także przez przeciągnięcie (lub skopiowanie) pliku z grafiką z menedżera plików do obszaru edytora. Kreator wygeneruje kompletne środowisko **`figure`** z poleceniem **`\includegraphics`**, podpisem, definicją etykiety itd. Kreator można uruchomić dla już istniejącego środowiska **`figure`** i dokonać w nim odpowiednich poprawek (ustawiamy kursor gdzieś w środku kodu środowiska i wybieramy polecenie uruchamiające kreator). Wprowadzone opcje wstawiania grafiki można zapisać jako domyślne (przycisk w lewym dolnym rogu okna kreatora).

Podczas edycji tekstu funkcjonuje system autouzupełniania. Jeżeli wprowadzimy ukośnik wsteczny i pierwszą literę polecenia, system wyświetli listę możliwości. Możemy z niej wybrać polecenie lub jeśli lista jest długa, wpisać



Rys. 15.10. Kreator `Wstaw grafikę...`

kolejną literę, aby wyeliminować część opcji. Jeżeli wstawiamy polecenie przyjmujące parametry, to nawiasy są wypełnione krótkimi wskazówkami o ich przeznaczeniu. Jeżeli polecenie przyjmuje opcje w układzie `klucz=wartość`, to system autouzupełniania uruchamia się także po wpisaniu pierwszej litery z nazwy pierwszego klucza i po wpisaniu przecinka po poprzedniej wartości. System autouzupełniania można wywołać również przy wprowadzaniu zwykłego tekstu. Po wpisaniu fragmentu słowa (wystarczy jedna litera) kombinacja `Ctrl+Spacja` wywoła okno, w którym znajdą się wszystkie słowa z bieżącego dokumentu, które mają ten sam początek.

Edycja tekstu jest wspierana przez szereg poleceń z menu `Edycja`. Mamy tam dostępne m.in. polecenia do konwersji wielkości liter, usuwania wierszy, powielania wierszy, przesuwania w górę i w dół, sortowania wierszy, zaawansowane mechanizmy zaznaczania (np. zaznaczenie całej zawartości nawiasu, w którym znajduje się kursor) itp. Ponadto TeXstudio oferuje możliwość pracy z wieloma kursorami jednocześnie. Jeżeli chcemy ustawić kilka kursorów w dowolnych miejscach, to po ustawieniu pierwszego (zwykłe kliknięcie lewym przyciskiem myszy) kolejne dodajemy, klikając lewym przyciskiem z jednocześnie wciśniętymi klawiszami `Ctrl+Alt`. Jeżeli chcemy ustawić kolumnę kursorów, to ustawiamy pierwszy (tak jak poprzednio) i ostatni z wciśniętymi klawiszami `Ctrl+Shift`. Kolumnę kursorów można również ustawić klawiszami ↑ i ↓ przy

wciśniętych klawiszach `Ctrl+Alt`. Naciśnięcie klawisza `Esc` lub kliknięcie gdziekolwiek w tekście powoduje powrót do trybu jednego kursora.

Listing 15.1. Kod źródłowy tabeli przed zmianami

```
\begin{tabular}{|r|r|r|r|r|}
\hline\multicolumn{5}{|c|}{Normalizacja} \\\hline
x & x1 & x2 & x3 & x4 \\\hline
-2,0 & 0,25 & 0,03 & -0,39 & -0,002 \\
1,5 & 0,32 & 0,03 & -0,37 & 0,002 \\
6,6 & 0,42 & 0,04 & -0,34 & 0,007 \\
500,0 & 10,00 & 1,00 & 2,47 & 0,500 \\
28,0 & 0,83 & 0,08 & -0,22 & 0,028 \\
-10,0 & 0,10 & 0,01 & -0,43 & -0,010 \\
20,0 & 0,68 & 0,07 & -0,26 & 0,020 \\
-15,0 & 0,00 & 0,00 & -0,46 & -0,015 \\\hline
\end{tabular}
```

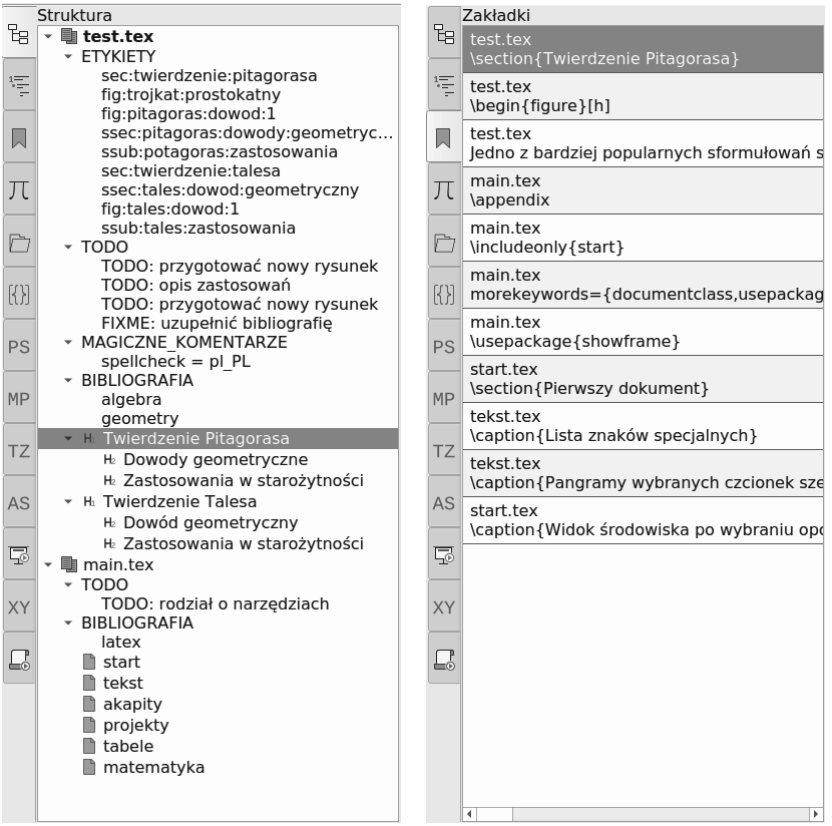
Załóżmy, że chcemy poprawić kod w tabeli z listingu 15.1 w następujący sposób: dodać po dwa zera na końcu liczb w pierwszej kolumnie i po jednym zerze w kolumnach 2–4. Zaczynamy od ustawienia kursora gdzieś wewnątrz środowiska **`tabular`** i wybieramy polecenie `LaTeX→Edycja tabel→Wyrównaj kolumny`. Efekt tego polecenia widzimy na listingu 15.2.

Listing 15.2. Kod źródłowy tabeli po wyrównaniu kolumn

```
\begin{tabular}{|r|r|r|r|r|}
  \hline
  \multicolumn{5}{|c|}{Normalizacja}    \\
  \hline
      x &    x1 &   x2 &    x3 &     x4 \\
  \hline
    -2,0 &  0,25 & 0,03 & -0,39 & -0,002 \\
     1,5 &  0,32 & 0,03 & -0,37 &  0,002 \\
     6,6 &  0,42 & 0,04 & -0,34 &  0,007 \\
  500,0 & 10,00 & 1,00 &  2,47 &  0,500 \\
    28,0 &  0,83 & 0,08 & -0,22 &  0,028 \\
  -10,0 &  0,10 & 0,01 & -0,43 & -0,010 \\
    20,0 &  0,68 & 0,07 & -0,26 &  0,020 \\
  -15,0 &  0,00 & 0,00 & -0,46 & -0,015 \\
  \hline
\end{tabular}
```

Teraz zadanie możemy łatwo wykonać, używając kolumny kursorów. Klikamy na początku linii 7, a następnie z wciśniętymi klawiszami `Ctrl+Shift` na początku linii 14. W tym momencie powinniśmy widzieć 8 mrugających

kursorów. Naciskamy 7 razy klawisz →, wpisujemy 00, 8 razy klawisz →, wpisujemy 0 itd. To nie magia, tryb wielu kursorów dostępny jest w dobrych edytorach kodu źródłowego. W naszym przypadku wyrównanie kolumn bardzo ułatwiło zadanie, ale warto pamiętać, że wszystkie ustawione kursory możemy przesuwać kombinacjami `Ctrl+→` i `Ctrl+←` (słowo w prawo, słowo w lewo) lub klawiszami `Home` i `End`, np. ustawienie kolumny kursorów jak opisano przed chwilą i wciśnięcie `End` spowoduje, że wszystkie kursory będą na końcach swoich linii niezależnie od ich długości.



Rys. 15.11. Widok zakładek `Struktura` i `Zakładki`

TeXstudio zawiera kilka mechanizmów, które ułatwiają poruszanie się po projekcie, zwłaszcza gdy składa się z wielu plików. Zakładka `Struktura` (pierwsza od góry) w lewym panelu gromadzi mnóstwo informacji o otwartych dokumentach: listę zdefiniowanych etykiet, listę komentarzy TODO, listę magicznych komentarzy, listę podpiętych plików z bibliografią, drzewo struktury dokumentu i listę dołączonych plików. Przykładowy widok zakładki `Struktura`

pokazano na rys. 15.11 po lewej stronie. Kliknięcie dowolnego z tych elementów oznacza albo przeniesienie we właściwe miejsce w otwartym dokumencie, albo otwarcie dokumentu, jeżeli jest to nazwa pliku. W przypadku etykiet widocznych w panelu w ich menu podręcznym (po kliknięciu etykiety prawym przyciskiem myszy) mamy m.in. możliwość wstawienia etykiety jako parametru polecenia **\ref** lub **\pageref**. Stosowne polecenie jest wstawiane w aktualnym miejscu kursora. Ponadto menu zawiera polecenie `Znajdź wystąpienia`, które pozwala na znalezienie wszystkich odwołań do danej etykiety. Wyniki wyszukiwania są wyświetlane w dolnym panelu, w zakładce o takiej właśnie nazwie. Przy okazji mamy możliwość zastąpienia szukanej etykiety nową wartością (zmiana nazwy etykiety).

Komentarze TODO to szczególny typ komentarzy, które są zdefiniowane w ustawieniach TeXstudio poprzez wyrażenie regularne. Są one rozpoznawane przez środowisko, wyróżniane innym kolorem i gromadzone w panelu `Struktura`. Wyrażenie regularne ustawiamy w: `Opcje→Konfiguruj TeXstudio...→Zaawansowany edytor`. Na przykład zdefiniowanie wyrażenia regularnego jako `%(TODO|FIXME|NOTE):` oznacza, że każdy komentarz zaczynający się od `%TODO:`, `%FIXME:` lub `%NOTE:` będzie traktowany jako komentarz TODO.

Komentarz magiczny jest specjalnym komentarzem LaTeX-a wstawianym na początku dokumentu. Rozpoznawalne komentarze wstawiamy za pomocą poleceń z menu `LaTeX→Dodaj magiczne komentarze...`. Przykładem takiego komentarza jest informacja o języku dokumentu, np. `% !TeX spellcheck = pl_PL`. Na podstawie takiego komentarza środowisko przy otwarciu pliku automatycznie ustawia właściwy język do sprawdzania pisowni.

Drzewo struktury dokumentu zawiera informacje o tytułach rozdziałów, sekcji, podsekcji itd. Jest ono aktualizowane, jeżeli zmieniamy cokolwiek, co wpływa na strukturę dokumentu. W przypadku prezentacji (`beamer`) drzewo zawiera tytuły wszystkich slajdów. W menu podręcznym elementów drzewa struktury mamy dostępne m.in. operacje kopiuj, wytnij, wklej przed, wklej po. Pozwalają one kopiować/wycinać i wklejać całe rozdziały, sekcje, podsekcje itd.

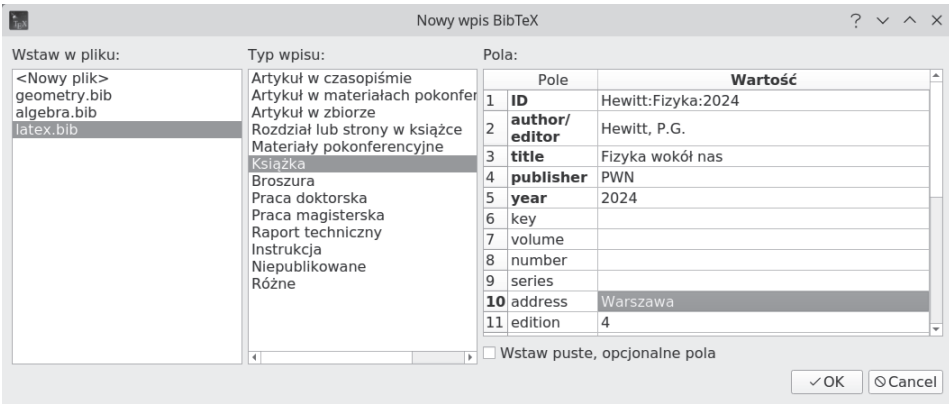
Niezależnie od wszystkich informacji gromadzonych w zakładce `Struktura` użytkownik ma do dyspozycji system zakładek. W każdym dokumencie możemy dodać 10 numerowanych i dowolną liczbę nienumerowanych zakładek. Zakładki dodajemy, klikając puste miejsce na lewo od numeru linii (w tym miejscu wyświetlany jest znacznik zdefiniowanej zakładki), a usuwamy, klikając taki znacznik. System najpierw dodaje zakładki numerowane, a gdy ich braknie – nienumerowane. Możemy je dodawać również z menu podręcznego (klikamy prawym przyciskiem myszy w obszarze przeznaczonym na zakładki), a wtedy sami wybieramy, którą zakładkę chcemy dodać. Warto pamiętać, że ponowne dodanie tej samej numerowanej zakładki spowoduje usunięcie jej

w poprzednim miejscu. Zakładki dla wszystkich otwartych dokumentów oraz dla dokumentów podpiętych do otwartych są wyświetlane w lewym panelu w zakładce `Zakładki` (zob. rys. 15.11 po prawej stronie). Kliknięcie zakładki powoduje przeniesienie nas do odpowiedniego miejsca we właściwym dokumencie. Szybki dostęp do trzech pierwszych zakładek (numery 1–3) w bieżącym dokumencie mamy poprzez przyciski zakładek w prawym dolnym rogu okna edytora (zob. rys. 15.5). Ponadto zakładki 1–9 mają domyślnie przypisane skróty klawiaturowe `Ctrl+`*n*, gdzie *n* oznacza numer zakładki. Więcej poleceń związanych z używaniem zakładek znajduje się w menu `Edycja→Idź do` i `Edycja→Idź do zakładki`.

Kompilacja dokumentu realizowana jest poprzez polecenia w menu `Narzędzia` – dostępne są różne kompilatory, podglądy, narzędzia do generowania skorowidza, glosariusza, usuwanie plików tymczasowych itp. Wybrane z nich są dostępne na pasku narzędziowym (drugi od lewej na rys. 15.5). Jeżeli otwarty jest podgląd pliku `pdf`, to jest on automatycznie aktualizowany po kompilacji dokumentu. Log z kompilacji dostępny jest w dolnym panelu w zakładce `Dziennik`. W zależności od włączonych przełączników (`Plik dziennika`, `Problemy` itd.) widzimy komplet informacji lub tylko wybrane. Kliknięcie dowolnego wiersza na liście problemów przenosi nas do miejsca, gdzie kompilator ten problem zauważył (jeżeli jest taka potrzeba TeXstudio automatycznie otwiera dodatkowe pliki).

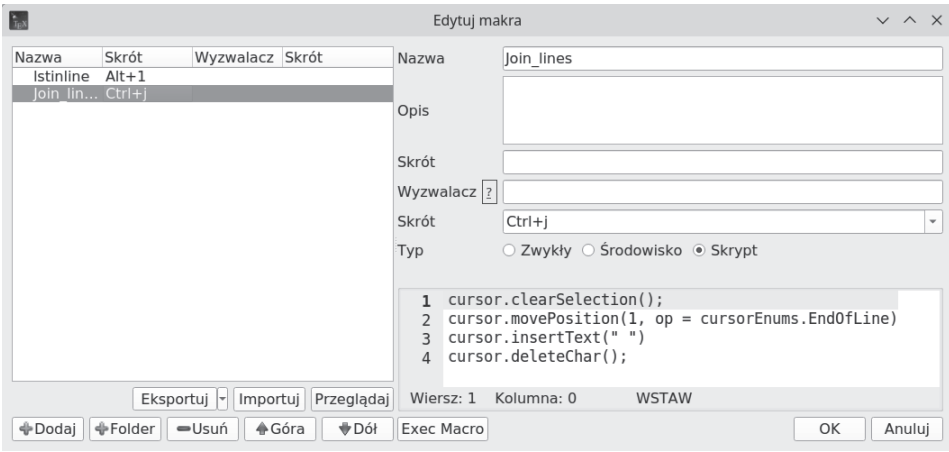
Jeżeli podgląd nie jest włączony, to uruchamiamy go poleceniem `Narzędzia→Podejrzyj` (lub właściwym przyciskiem na pasku narzędziowym). Wśród ciekawszych poleceń okna podglądu warto zwrócić uwagę na przycisk `Powiększenie`, którego wybranie zamienia kursor myszy w lupę, którą możemy powiększyć fragment pliku `pdf`. Ponadto podmenu `Podgląd→Siatka` pozwala zorganizować podgląd pliku `pdf` w taki sposób, że jednocześnie widzimy kilka sąsiednich stron. Rozwiązanie przydaje się, gdy chcemy dopracować rozkład wstawek w dokumencie.

TeXstudio wspiera również edycję blików `bib`. Menu `Bibliografia` zawiera m.in. polecenia wstawiające wszystkie typy rekordów bibliograficznych. Zawartość menu zależy od tego, czy do obsługi bibliografii wybierzemy BibTEX-a, czy BibLATEX-a (podmenu `Bibliografia→Typ`). Ponadto poleceniem `Bibliografia→Wstaw element bibliografii...` możemy uruchomić kreatora, który pomoże przygotować właściwy rekord. Widok okna kreatora pokazano na rys. 15.12. Możemy wybrać, do którego pliku `bib` z używanych w otwartych dokumentach chcemy dodać nowy rekord, wybrać typ rekordu i wypełnić jego zawartość. Po zatwierdzeniu gotowy rekord zostaje wstawiony do wybranego pliku `bib`.



Rys. 15.12. Widok okna `Wstaw element bibliografii...`

Na koniec warto wspomnieć, że TeXstudio umożliwia użytkownikowi definiowanie własnych makr (menu `Makra`). Okno edycji makr pokazano na rys. 15.13. W najprostszym przypadku możemy przypisać skrót do własnego polecenia (zob. podrozdział 14.5) lub do polecenia, które nie występuje w menu. Na przykład przypisanie skrótu do polecenia **`\lstinline`** sprowadza się do: dodania nowego makra w oknie z rys. 15.13, podania nazwy (tutaj `lstinline`), dodania skrótu, wybrania typu `Zwykły` i dodania kodu **`\lstinline`**`!%< code %>!`. Po zamknięciu okna przyciskiem `OK` możemy zaznaczyć dowolny fragment tekstu, a po naciśnięciu skrótu `Alt+1` wstawione zostanie polecenie **`\lstinline`** i zaznaczony tekst zostanie umieszczony pomiędzy znakami wykrzyknika.



Rys. 15.13. Widok okna edycji makr

Bardziej zaawansowane makra można tworzyć z użyciem języka Java Script. Listę dostępnych poleceń, których można użyć w skryptach, znajdziemy pod adresem `https://texstudio-org.github.io/advanced.html`. Przykład pokazany na rysunku odpowiada za połączenie bieżącej linii z kolejną. Pierwsze polecenie usuwa ewentualne zaznaczenie, drugie przesuwa kursor na koniec linii, następnie dodajemy spację i usuwamy jeden znak (`\n`).

Rozdział 16

Użycie AI do pracy nad dokumentem

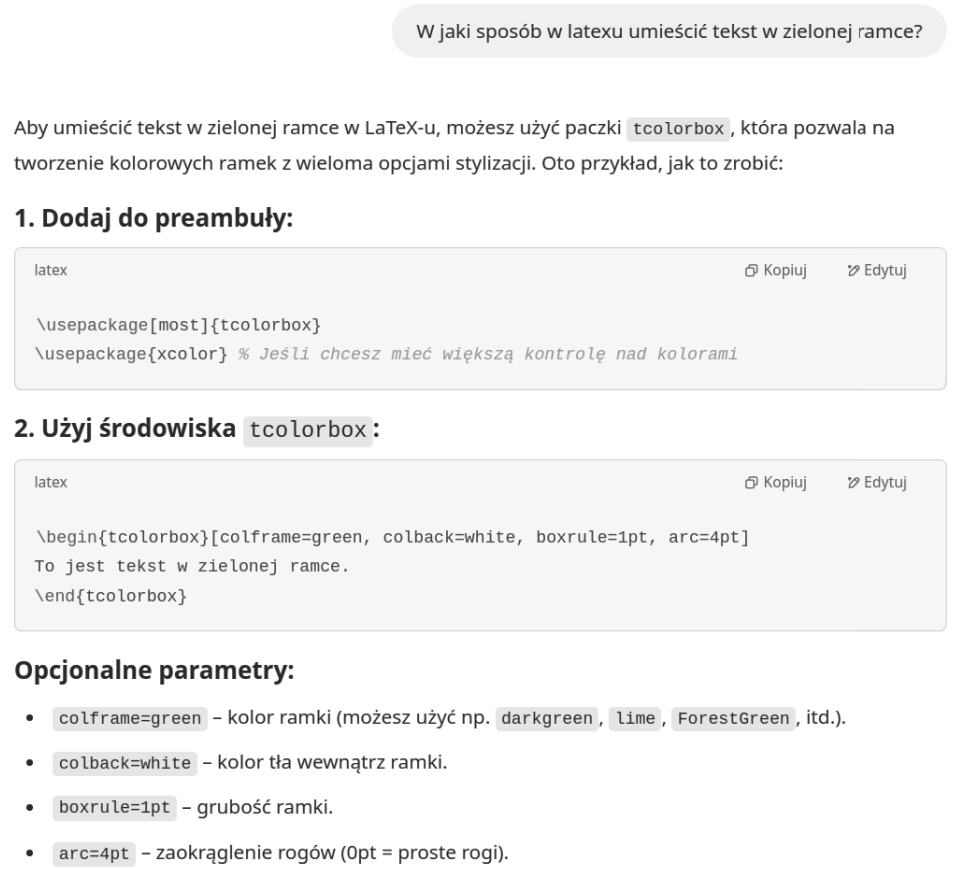
W czasie gdy piszę tekst tej książki, niezwykle popularne stały się tzw. duże modele językowe (LLM – Large Language Model), czyli algorytmy sztucznej inteligencji stosujące techniki głębokiego uczenia i olbrzymie zbiory danych uczących do analizowania i generowania treści tekstowych. Algorytmy te mogą również tłumaczyć tekst z jednego języka na drugi, generować podsumowanie dla wskazanej partii tekstu, klasyfikować tekst, analizować wydźwięk emocjonalny tekstu itd. Wśród najczęściej spotykanych zastosowań są czatboty, czyli programy, z którymi użytkownik może prowadzić rozmowę podobnie jak z drugą osobą. Jednym z najbardziej popularnych czatbotów jest ChatGPT firmy OpenAI. Czatbot jest w stanie przetwarzać i analizować kontekst rozmowy, a co za tym idzie – generować odpowiedzi adekwatne do tematu. Czatboty szybko znalazły zastosowanie w narzędziach do programowania. Czatbot może nie tylko udzielać wskazówek, jak coś zrobić, ale generować fragmenty kodu źródłowego. W tym rozdziale przyjrzymy się możliwości wykorzystania czatbotów do przygotowania dokumentu w LaTeX-u.

16.1 Praca z ChatGPT

Firma OpenAI oferuje zarówno bezpłatny, jak i płatny dostęp do ChatGPT. W tym rozdziale skupimy się na dostępie darmowym, bo możliwości, które oferuje, są naprawdę duże i wystarczające jako wsparcie pracy z LaTeX-em. Najprostsza wersja chata jest dostępna poprzez stronę `https://chatgpt.com`. Na starcie możemy wypróbować ChatGPT bez zakładania konta, po prostu wpisujemy pytanie i po chwili otrzymujemy odpowiedź. Jak łatwo zauważyć po języku interfejsu, z czatem możemy rozmawiać w języku polskim. Po kilku pytaniach zostaniemy poproszeni o założenie konta i jest to zdecydowanie dobre rozwiązanie.

Nie musimy wybierać płatnych subskrypcji, a za to mamy m.in. dostęp do historii rozmów i zdecydowanie bardziej wiarygodne odpowiedzi.

Jeżeli już zdążyłeś uciąć sobie pogawędkę z czatem, to czas zapytać go o LaTeX-a. Po pierwsze możemy go potraktować jako szybki dostęp do pomocy. Na rys. 16.1 pokazano przykład pytania i wygenerowanej odpowiedzi, a właściwie jej części. W odpowiedzi znalazła się bowiem jeszcze jedna ramka, która zawierała przykład pełnego dokumentu. Zawartość każdej ramki możemy łatwo skopiować, korzystając z przycisku w jej prawym górnym rogu.

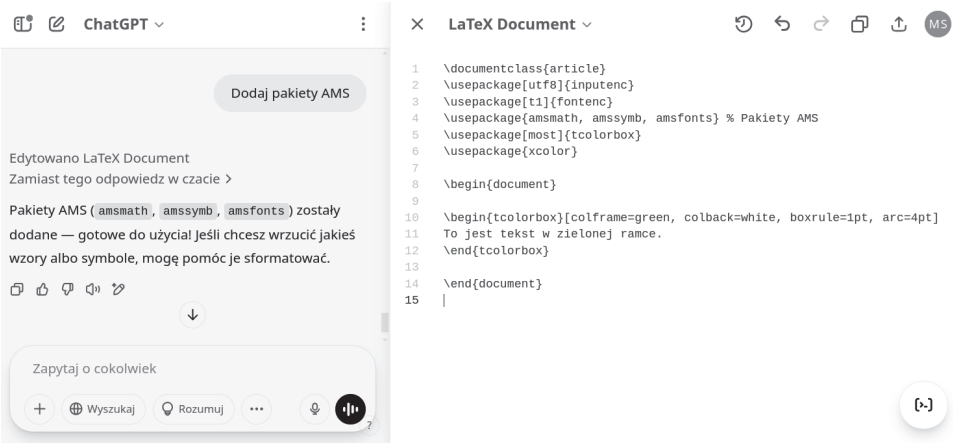


Rys. 16.1. Praca z CharGPT

Rozmowę możemy łatwo kontynuować. ChatGPT pracuje kontekstowo i wie, o czym toczy się rozmowa. Na kolejne polecenie/pytanie: *A jak zmienić kolor w ramce na ciemnozielony?* dostajemy dwie propozycje zmiany koloru ramki: jedną z użyciem predefiniowanego koloru, a drugą z definicją własnego.

Możemy jednak doprecyzować pytanie: *Ale chodzi o kolor tekstu* i zobaczymy odpowiedź w stylu: *Aaa, jasne! Myślałeś o kolorze tekstu w ramce, a nie samej ramki – dzięki za doprecyzowanie. Dodaj opcję* `coltext=ForestGreen` *albo* `coltext=darkgreen`, *np.:...*

Na potrzeby pracy z kodem źródłowym firma OpenAI wprowadziła nowy interfejs graficzny określany jako ChatGPT Canvas. W pokazanych na rys. 16.1 dwóch ramkach z kodem źródłowym widoczny jest przycisk `Edytuj`. W momencie, gdy klikniemy ten przycisk dla którejś ramki, staje się ona bieżącym dokumentem, nad którym interaktywnie pracujemy z czatem.



Rys. 16.2. Praca z ChatGPT Canvas

Przykład takiego trybu pracy przedstawiono na rys. 16.2. Ekran pracy podzielony jest na dwa obszary – okno czatu i okno z edytorem kodu. W tym przypadku otwarto niepokazaną na rys. 16.1 ramkę z przykładem pełnego dokumentu. Kolejne polecenie *Dodaj pakiety AMS* ChatGPT odniósł do otwartego kodu i w odpowiedzi na polecenie uzupełnił kod o linię 4. W dowolnym momencie możemy zamknąć edytor (przycisk ×), zapisać lub odrzucić zmiany i otworzyć inny kawałek kodu.

Taki tryb pracy pozwala na interaktywne poprawianie fragmentów kodu bez potrzeby wielokrotnego przenoszenia ich między oknem czata a naszym edytorem. Przy pracy z Canvas możemy zaznaczyć kawałek kodu i kolejne polecenie dla ChatGPT wydać w odniesieniu do tego fragmentu (polecenie możemy wpisać w edytorze inline, który zostanie wyświetlony po kliknięciu podpowiedzi w dymku, która wyświetla się przy zaznaczeniu), możemy wyświetlić wprowadzone zmiany albo wrócić do wcześniejszej wersji kodu.

Nasze zadanie dla ChatGPT może być zdecydowanie bardziej ambitne niż tylko wygenerowanie fragmentu kodu. Polecenie *Wygeneruj szkielet dokumentu*

w LaTeX-u do przygotowania artykułu dla IEEE Access. Uwzględnij w preambule pakiety AMS i pakiet tikz. Dodaj trzy przykładowe sekcje i do każdej dwie podsekcje. Umieść przykładową tabelę, rysunek i wzór matematyczny eksponowany numerowany. Artykuł będzie w języku angielskim – wygeneruje kompletny szkielet dokumentu zgodnie z naszymi wytycznymi. Ostatnie zdanie jest istotne, jeżeli nie chcemy, aby przykładowe zdania w środku były w języku polskim.

Samo wygenerowanie szkieletu dokumentu nie pozwala na jego kompilację. Ze strony wydawcy musimy pobrać archiwum z dodatkowymi plikami, które należy umieścić w katalogu projektu (lokalnie lub w Overleaf). Potrzebujemy m.in. plików z klasą dokumentu i stylem bibliografii.

ChatGPT Canvas możemy użyć nie tylko do pracy z kodem wygenerowanym przez ChatGPT, ale możemy skopiować własny kawałek kodu, który chcemy modyfikować we współpracy z czatem. Kod taki możemy wkleić bezpośrednio w czacie, albo dodać do bieżącego dokumentu Canvas. Niektóre błędy są wyłapywane bez stawiania konkretnych poleceń i od razu mamy sugestie poprawek, na przykład wklejenie kodu tabeli z brakującym w jakimś wierszu znakiem `&` spowoduje, że czat od razu zaproponuje nam poprawioną wersję kodu.

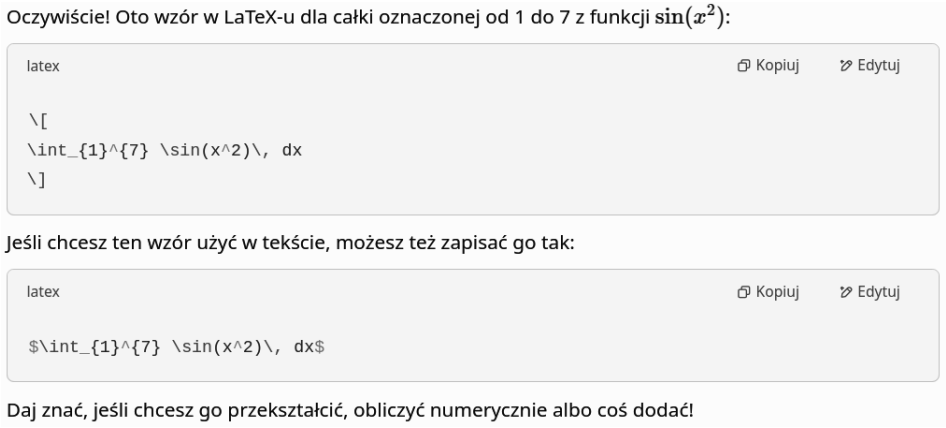
Na zakończenie tego podrozdziału warto wspomnieć o zagrożeniach związanych z pracą z ChatGPT. Jeżeli kopiujemy do czata fragmenty naszych dokumentów, nad którymi chcemy pracować, musimy zadbać, aby nie było tam wrażliwych danych. Niezależnie od ustawień prywatności, które oferuje Open AI, nie powinniśmy przenosić tam danych, które uważamy za poufne.

16.2 Wzory matematyczne

W kolejnych podrozdziałach przyjrzymy się kilku wybranym aspektom pracy z LaTeX-em i jak ChatGPT może nam w tym pomóc. Nie jest możliwe podanie wszystkich potencjalnych zastosowań, ale pokazane przykłady powinny wystarczająco zobrazować te możliwości. Zaczniemy od tego, co stanowi serce LaTeX-a, czyli wzorów matematycznych. Pierwsze polecenie możemy zapisać jako: *Wygeneruj wzór w LaTeX-u: całka od 1 do 7 z sinus z x do kwadratu.*

Wygenerowaną odpowiedź widzimy na rys. 16.3. Została ona dostarczona w dwóch wersjach eksponowanej (ograniczniki `\[ \]` są jednym ze sposobów LaTeX-a na umieszczanie nienumerowanych eksponowanych wzorów matematycznych) i wewnątrzakapitowej. Dodatkowo informacja przed pierwszym wzorem wskazuje, jak zinterpretowane zostało nasze polecenie. Język polski nie jest w tym miejscu precyzyjny. Nasze polecenie można było zinterpretować co

najmniej na trzy sposoby. Jeżeli w kolejnym poleceniu dodamy *sinus z x ma być podniesiony do kwadratu*, to dostaniemy poprawiony wzór $\int_1^7 \sin^2(x)\,dx$. Na kolejne pytanie: *A gdyby cała całka miała być podniesiona do kwadratu?* dostaniemy odpowiedź *Świetne pytanie! Jeśli chcesz, żeby wynik całkowania (czyli wartość całki oznaczonej) był podniesiony do kwadratu, to musisz objąć całkę nawiasami i potem podnieść całość do potęgi 2. Oto jak to zapisać w LaTeX-u:* i dołączony wzór $\left(\int_1^7 \sin^2(x)\,dx\right)^2$.



Rys. 16.3. Wzór wygenerowany przez ChatGPT na podstawie opisu

Jeżeli wzór, o który nam chodzi, jest ogólnie znany, to nie musimy go opisywać. Wydając polecenie: *równanie elipsy o środku (0,0)*, dostajemy wzór:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{16.1}$$

ChatGPT umożliwia nie tylko tekstowe zadanie pytań, ale również przesyłanie grafiki. Na polecenie *Wygeneruj wzór w LaTeX-u taki jak na obrazku*, zostajemy poproszeni o przesłanie obrazka, po czym dostajemy odpowiedź jak na listingu 16.1. Przesłana grafika została pokazana pod listingiem .

Listing 16.1. Kod wygenerowany przez ChatGPT na podstawie grafiki

```
\[
\int \frac{x^{-1}}{\ln x} \, dx
\]
```

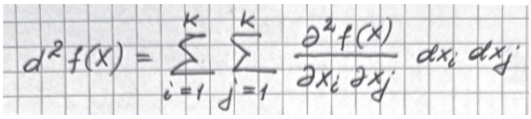
$$\int \frac{x^{-1}}{\ln x}\,dx$$

Zadanie możemy uczynić trudniejszym, dostarczając na przykład zdjęcie ręcznie zapisanego wzoru. Przykład pokazano na listingu 16.2. Poniżej kodu

przedstawiono obrazek przekazany do ChatGPT wraz z poleceniem *Wygeneruj mi ten wzór w LaTeX-u*. Poniżej zdjęcia widzimy efekt kompilacji uzyskanego kodu. Jako uzupełnienie pokazanego kodu podana została wiadomość *Ten wzór przedstawia różniczkę zupełną drugiego rzędu funkcji $f(x)$, wyrażoną za pomocą drugich pochodnych cząstkowych...*

Listing 16.2. Kod wygenerowany przez ChatGPT na podstawie pisma odręcznego

```
\[
d^2 f(x) = \sum_{i=1}^{k} \sum_{j=1}^{k} \frac{\partial^2 f(x)}
{\partial x_i\, \partial x_j} \, dx_i\, dx_j
\]
```



$$d^2 f(x) = \sum_{i=1}^{k} \sum_{j=1}^{k} \frac{\partial^2 f(x)}{\partial x_i \, \partial x_j} \, dx_i \, dx_j$$

Patrząc na uzyskane wyniki, można powiedzieć „Wow, super, to po co mi ta książka i po co uczyłem się składać matematykę w LaTeX-u?". Niestety nie wszystko idzie tak gładko jak te przykłady. Zdarza się, że generowany kod zawiera błędy składniowe, które musimy poprawić. Poza tym potrzebujemy wiedzy na temat tego, co jest możliwe, jeśli chcemy precyzyjnie wyrazić swoje oczekiwania. Dodatkowo nie wszystko da się generować tak „łatwo" jak wzory matematyczne, co pokażę w dalszej części tego rozdziału.

ChatGPT możemy użyć również do poprawy organizacji kodu w LaTeX-u. Załóżmy, że w naszym bieżącym dokumencie w Canvas mamy serię wzorów matematycznych pokazanych na listingu 16.3.

Listing 16.3. Seria wzorów matematycznych złożonych w środowisku **equation**

```
\begin{equation}
\label{eq:function1}
f(x) = x^2 -7
\end{equation}

\begin{equation}
\label{eq:function2}
f(x) = x^3 -\frac{1}{2}x^2 + 1
\end{equation}

\begin{equation}
\label{eq:function3}
```

```
f(x) = \sqrt{x-1} + \frac{1}{3}x
\end{equation}
```

Jeżeli zaznaczymy je i wpiszemy polecenie: *zamień na środowisko align*, to otrzymamy kod jak na listingu 16.4.

Listing 16.4. Kod wygenerowany przez ChatGPT do zastąpienia wzorów z listingu 16.3

```
\begin{align}
\label{eq:function1}
f(x) &= x^2 - 7 \\
\label{eq:function2}
f(x) &= x^3 - \frac{1}{2}x^2 + 1 \\
\label{eq:function3}
f(x) &= \sqrt{x - 1} + \frac{1}{3}x
\end{align}
```

Oczywiście działa to również w drugą stronę, możemy zaznaczyć środowisko **align** i zamienić je na serię indywidualnych wzorów.

Na zakończenie tego podrozdziału spróbujmy wygenerować dokument na podstawie zawartości rys. 16.4. Pokazany fragment pochodzi z książki [44]. Obrazek wystarczy dodać do czata i uzupełnić poleceniem: *wygeneruj mi ten dokument w LaTeX-u*. Efekt pracy ChatGPT widzimy na listingu 16.5 (zamieszczono tylko zawartość środowiska **document**). Po wydaniu kolejnych poleceń poprawiających kod: *dodaj pakiet fontenc z opcją T1 w preambule, zamień \[\] na środowisko equation i usuń polecenia \tag* oraz *zdefiniuj środowisko definition i użyj go zamiast \paragraf{Definicja 5.12.}* dostajemy naprawdę zadowalający fragment kodu.

Niech $\mathcal{N} = (P, T, A, W, M_0)$ będzie siecią uogólnioną i niech zbiory P i T będą zbiorami uporządkowanymi, $P = \{p_1, p_2, \ldots, p_n\}$ i $T = \{t_1, t_2, \ldots, t_m\}$.

Definicja 5.12. *Macierzą wejść* sieci $\mathcal{N}$ nazywamy macierz $N^+ = \left(\alpha_{ij}\right)_{n\times m}$, której współczynniki α_{ij} są zdefiniowane następująco:

$$\alpha_{ij} = \begin{cases} W(t_j, p_i), & \text{jeżeli } t_j \in In(p_i) \\ 0 & \text{w pozostałych przypadkach.} \end{cases} \tag{5.18}$$

Macierzą wyjść sieci $\mathcal{N}$ nazywamy macierz $N^- = \left(\alpha_{ij}\right)_{n\times m}$, której współczynniki α_{ij} są zdefiniowane następująco:

$$\alpha_{ij} = \begin{cases} W(p_i, t_j), & \text{jeżeli } t_j \in Out(p_i) \\ 0 & \text{w pozostałych przypadkach.} \end{cases} \tag{5.19}$$

■

Rys. 16.4. Fragment książki [44]

Listing 16.5. Kod wygenerowany przez ChatGPT na podstawie rys. 16.4

```
Niech $\mathcal{N} = (P, T, A, W, M_0)$ będzie siecią uogólnioną
i niech zbiory $P$ i $T$ będą zbiorami uporządkowanymi,
$P = \{p_1, p_2, \dots, p_n\}$ i $T = \{t_1, t_2, \dots, t_m\}$.

\paragraph{Definicja 5.12.}
\textit{Macierzą wejść} sieci $\mathcal{N}$ nazywamy macierz
$N^+ = (\alpha_{ij})_{n \times m}$, której współczynniki $\alpha_{ij}$
są zdefiniowane następująco:
\[
\alpha_{ij} =
\begin{cases}
W(t_j, p_i), & \text{jeżeli } t_j \in In(p_i) \\
0,           & \text{w pozostałych przypadkach}.
\end{cases}
\tag{5.18}
\]

\textit{Macierzą wyjść} sieci $\mathcal{N}$ nazywamy macierz
$N^- = (\alpha_{ij})_{n \times m}$, której współczynniki $\alpha_{ij}$
są zdefiniowane następująco:
\[
\alpha_{ij} =
\begin{cases}
W(p_i, t_j), & \text{jeżeli } t_j \in Out(p_i) \\
0,           & \text{w pozostałych przypadkach}.
\end{cases}
\tag{5.19}
\]
```

16.3 Tabele

Jako kolejny kłopotliwy problem przeanalizujemy możliwość użycia ChatGPT jako wsparcia przy składaniu tabel. Nasze pierwsze zadanie jest następujące: *Wygeneruj strukturę tabeli w LaTeX-u: 3 kolumny, 4 wiersze, w pierwszym wierszu dwie pierwsze komórki połączone w jedną.* Uzyskany kod (pomijając wyjaśnienia) pokazano na listingu 16.6. Patrząc na wynik, można stwierdzić, że struktura środowiska **`tabular`** jest jak najbardziej poprawna, ale kod jako całość nie kompiluje się. Problemem jest użycie polskich liter w przykładowym tekście w komórkach i brak odpowiednich pakietów (`inputenc` i `fontenc`). Z drugiej strony żaden z dodanych przez ChatGPT pakietów nie jest wymagany dla tego kodu. Na uwagę, że kod się nie kompiluje, dostajemy odpowiedź, że mamy rację,

i dokładnie ten sam kod. Dopiero zacytowanie błędu: *Dalej jest błąd: „Command \k unavailable in encoding OT1"* skutkuje uzyskaniem kodu z wymaganymi, ale również nadmiarowymi pakietami i stosownymi wyjaśnieniami odnośnie do przeznaczenia poszczególnych załączonych pakietów.

Listing 16.6. Kod wygenerowany przez ChatGPT (tabela 4×3)

```
\documentclass{article}
\usepackage{multirow}
\usepackage{multicol}
\usepackage{array}

\begin{document}
\begin{tabular}{|c|c|c|}
  \hline
  \multicolumn{2}{|c|}{Połączona komórka} & Kolumna 3 \\ \hline
  Kolumna 1 & Kolumna 2 & Kolumna 3 \\ \hline
  Kolumna 1 & Kolumna 2 & Kolumna 3 \\ \hline
  Kolumna 1 & Kolumna 2 & Kolumna 3 \\ \hline
\end{tabular}
\end{document}
```

Sam ChatGPT zachęca nas do dalszej konwersacji i sugeruje możliwość dodania nagłówków, wyrównania tekstu lub zmiany formatowania tabeli. Na kolejne polecenie: *Pierwsza kolumna ma być wyrównana do lewej strony, a ostatnia do prawej* podaje kod z odpowiednio zmienionym parametrem środowiska **tabular** (co ciekawe, tym razem bez pakietu `multicol`).

Możemy zwrócić się o pomoc z jakimś problemem, niekoniecznie o kawałek kodu. Na pytanie *Jak zwiększyć wysokość wierszy w tabeli?* dostajemy odpowiedź zawierającą aż trzy rozwiązania, w tym wykorzystanie redefinicji polecenia `arraystretch`. Możemy więc traktować ChatGPT jako szybki dostęp do pomocy kontekstowej.

Podobnie jak w przypadku grafiki, możemy wydać polecenie wygenerowania kodu dla tabeli przedstawionej na obrazku. Wystarczy dołączyć plik i wpisać np. *wygeneruj kod dla tabeli jak na obrazku* (z poprzednich poleceń ChatGPT wie, że ma to być kod w LaTeX-u). W naszym teście przesłany został obraz tabeli z listingu 5.14 (str. 101). Pomijając zbędne pakiety dołączone do preambuły, to samo środowisko **tabular** zostało wygenerowane nie najgorzej. Efekt widzimy na listingu 16.7.

Listing 16.7. Kod wygenerowany przez ChatGPT na podstawie obrazu tabeli

```
\begin{tabular}{|c|*{12}{c|}}
\hline
```

```
\multicolumn{13}{|c|}{2023} \\
\hline
 & \multicolumn{3}{c|}{kwartał 1} & \multicolumn{3}{c|}{kwartał 2} &
\multicolumn{3}{c|}{kwartał 3} & \multicolumn{3}{c|}{kwartał 4} \\
\hline
 & I & II & III & IV & V & VI & VII & VIII & IX & X & XI & XII \\
\hline
min & -9,8 & -14,1 & -11,6 & -7,4 & -2,5 & -1,5 & 5,6 & 5,9 & 3,7 &
-3,7 & -15,8 & -16,7 \\\hline
śr. & 2,9 & 1,5 & 4,5 & 7,7 & 12,8 & 17,5 & 19,3 & 19,6 & 17,7 &
10,9 & 4,2 & 2,6 \\\hline
max & 18,9 & 13,5 & 22,1 & 24,4 & 27,0 & 32,2 & 35,4 & 35,7 & 31,3 &
29,3 & 17,9 & 13,6 \\\hline
\end{tabular}
```

2023												
	kwartał 1			kwartał 2			kwartał 3			kwartał 4		
	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII
min	-9,8	-14,1	-11,6	-7,4	-2,5	-1,5	5,6	5,9	3,7	-3,7	-15,8	-16,7
śr.	2,9	1,5	4,5	7,7	12,8	17,5	19,3	19,6	17,7	10,9	4,2	2,6
max	18,9	13,5	22,1	24,4	27,0	32,2	35,4	35,7	31,3	29,3	17,9	13,6

Wynik nie jest idealny i potrzebujemy nieco wiedzy o składzie tabel, aby go poprawić, ale cały tekst został poprawnie odczytany i umieszczony w komórkach. Jeżeli potrzebujemy przenieść do LaTeX-a tabelę, którą mamy tylko w postaci wydruku, to ChatGPT wykonuje za nas większość pracy. Zresztą możemy kontynuować „rozmowę". Na uwagę: *Scalona komórka w pierwszym wierszu nie obejmuje pierwszej kolumny* dostaniemy kolejną nieco lepszą wersję tabeli.

```
 idmeczu |    gospodarze    |      goscie
---------+------------------+------------------
       1 | {28,23,13,12}    | {26,25,25,25}
       2 | {25,25,25}       | {12,17,20}
       3 | {25,25,25}       | {10,15,17}
```

Rys. 16.5. Wynik zapytania w terminalu SQL

Na rys. 16.5 pokazano wynik zapytania SQL uzyskany w programie `psql`. Dane wyświetlone w trybie pseudograficznym mają typową strukturę tabeli. Bez problemu możemy je szybko zamienić w tabelę w LaTeX-u. Po dodaniu rysunku do ChatGPT i wydaniu polecenia *Utwórz tabelę na podstawie danych z obrazu. Nie używaj czcionki monospace* dostaniemy kod pokazany na listingu 16.8. Drugie zdanie w poleceniu jest istotne, aby czat nie próbował naśladować dokładnie rysunku, bo zamiast tabeli dostaniemy kod w środowisku **`lstlisting`**.

Listing 16.8. Kod wygenerowany przez ChatGPT na podstawie rys. 16.5

```
\begin{tabular}{c|c|c}
\textbf{idmeczu} & \textbf{gospodarze} & \textbf{goscie} \\
\hline
1 & \{28, 23, 13, 12\} & \{26, 25, 25, 25\} \\
2 & \{25, 25, 25\}     & \{12, 17, 20\}     \\
3 & \{25, 25, 25\}     & \{10, 15, 17\}     \\
\end{tabular}
```

16.4 Wykresy

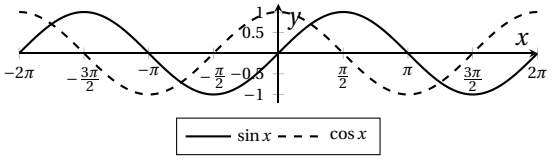
W rozdziale 12 przedstawiony został pakiet `pgfplots` pozwalający na kreślenie wykresów funkcji w dwóch i trzech wymiarach. Pakiet zawiera mnóstwo opcji związanych ze szczegółowym formatowaniem wykresu. ChatGPT może istotnie przyspieszyć uzyskanie potrzebnego kodu.

Zaczynamy od polecenia: *Wygeneruj wykresy funkcji sin x i cos x w przedziale od -2pi do 2pi. Oba wykresy umieść na jednym obrazku. Użyj klasy standalone i pakietu pgfplots. Ustaw osie jak, aby przecinały się w punkcie (0,0). Użyj tej samej skali na obu osiach. Dodaj etykiety osi X co pi/2*, które następnie zostało uzupełnione o doprecyzowanie: *Użyj rozmiaru scriptsize dla etykiet. Zamiast różnych kolorów linii użyj w jednym przypadku linii przerywanej. Nie używaj siatki.* Wynik takiego działania widzimy na listingu 16.9. Dość szybko możemy, już bez ChatGPT, poprawić wykres do zadowalającej wersji: ustawienie `ymax=1.7`, `xmin=-2*pi-0.2`, `xmax=2*pi+0.5`, usunięcie etykiet z osi y innych niż -1 i 1 oraz dodanie opcji `label style={font=\scriptsize}`.

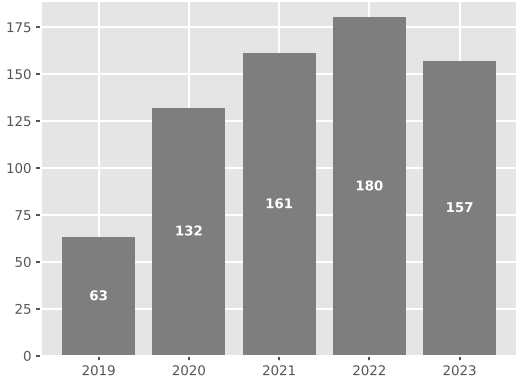
Listing 16.9. Wykres funkcji trygonometrycznych wygenerowany na podstawie opisu

```
\begin{tikzpicture}
\begin{axis}[
  axis lines=middle,
  xmin=-2*pi, xmax=2*pi,
  ymin=-1.2, ymax=1.2,
  xtick={-2*pi, -3*pi/2, -pi, -pi/2, 0, pi/2, pi, 3*pi/2, 2*pi},
  xticklabels={$-2\pi$, $-\frac{3\pi}{2}$, $-\pi$, $-\frac{\pi}{2}$,
    $0$, $\frac{\pi}{2}$, $\pi$, $\frac{3\pi}{2}$, $2\pi$},
  ytick={-1, -0.5, 0, 0.5, 1},
  yticklabels={$-1$, $-0.5$, $0$, $0.5$, $1$},
  tick label style={font=\scriptsize},
  axis equal image,
  enlargelimits=false,
  legend style={at={(0.5,-0.15)}, anchor=north, legend columns=-1,
```

```
    font=\scriptsize},
  xlabel={$x$},
  ylabel={$y$},
  grid=none
]
\addplot[black, thick, domain=-2*pi:2*pi, samples=200]{sin(deg(x))};
\addlegendentry{$\sin x$}

\addplot[black, thick, dashed, domain=-2*pi:2*pi, samples=200]
  {cos(deg(x))};
\addlegendentry{$\cos x$}
\end{axis}
\end{tikzpicture}
```



Spróbujmy odtworzyć prosty wykres słupkowy wygenerowany wcześniej w języku Python, który pokazano na rys. 16.6.



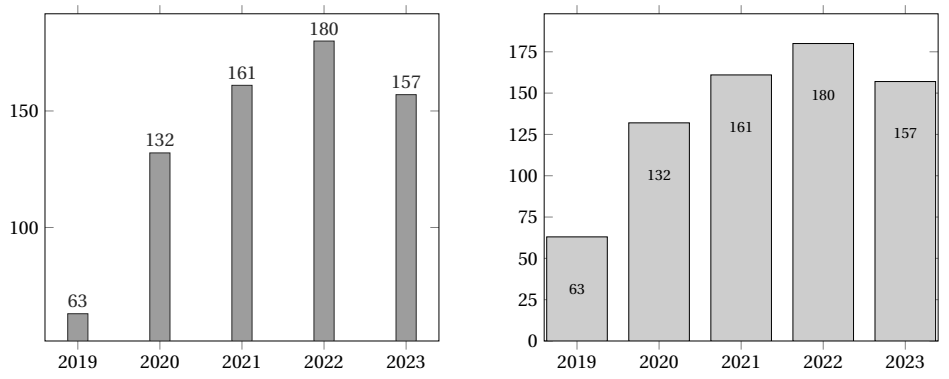
Rys. 16.6. Wykres słupkowy wygenerowany w języku Python, który chcemy odtworzyć

Polecenie zostało sformułowane jako: *Używając pakietu pgfplots, wygeneruj kod w LaTeX-u, generujący wykres słupkowy: słupki o wartościach 63, 132, 161, 180, 157 i etykietach 2019, 2020, 2021, 2022, 2023.* Wygenerowany kod (samo środowisko **tikzpicture**) pokazano na listingu 16.10, a afekt jego kompilacji na

lewym rysunku poniżej kodu. Kolejne dwa polecenia: *Ustaw szerokość słupków na 30 pt, przesuń etykiety z wartościami do środka słupków i ustaw znaczniki na osi Y co 25* oraz *Przesuń etykiety wartości niżej o 20 pt i ustaw zielony kolor słupków* prowadzą do wykresu pokazanego po prawej stronie pod kodem. Wynik jest bardzo przyzwoity i przy odrobinie pracy z naszej strony możemy poprawić go do formy zbliżonej do oryginału. Możemy również kontynuować dialog z ChatGPT, aby poprawić szczegóły.

Listing 16.10. Kod wygenerowany przez ChatGPT na podstawie opisu wykresu

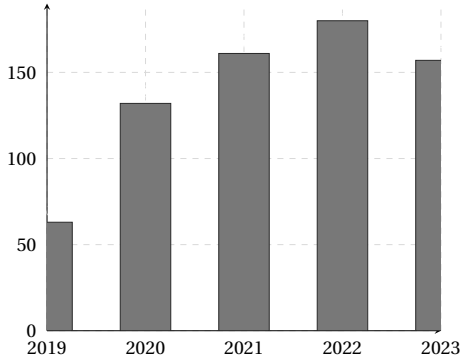
```
\begin{tikzpicture}
\begin{axis}[
    ybar,
    symbolic x coords={2019,2020,2021,2022,2023},
    xtick=data,
    nodes near coords
]
\addplot coordinates {
    (2019,63) (2020,132) (2021,161) (2022,180) (2023,157) };
\end{axis}
\end{tikzpicture}
```

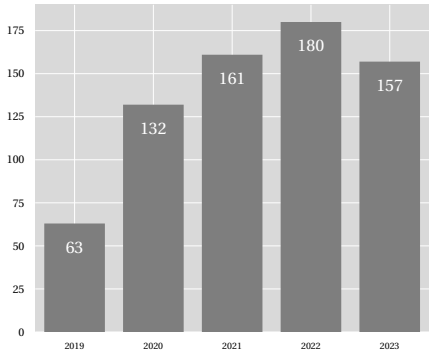


Do tego samego zadania możemy podejść inaczej, tj. dodać oryginalny wykres i wydać polecenie *Wygeneruj taki wykres w LaTeX-u, korzystając z pakietu pgfplots.* Efekt takiego polecenia pokazano na listingu 16.11. Jak widać na lewym rysunku pod kodem, wynik jest gorszy, niż moglibyśmy się spodziewać. ChatGPT pomylił nawet kolory dla tła i siatki (są zamienione). Potrzebujemy sporo poprawek, aby wynik był zbliżony do oryginału. Wszystkie zmiany przedstawiono na listingu w postaci komentarzy. Po ich wprowadzeniu uzyskamy wykres pokazany na prawym rysunku pod kodem.

Listing 16.11. Kod wygenerowany na podstawie istniejącego wykresu

```
\begin{tikzpicture}
\begin{axis}[
    ybar,                               % ytick distance=25,
    bar width=25pt, % zmiana na 30pt    % ticklabel style={font=\tiny},
    ymin=0,                             % axis line style={draw=none},
    ymax=190,                           % xmin=-0.5, xmax=4.5,
    enlarge x limits=0.2,  % usunięte, a dodane linie pokazane powyżej
    xtick=data,
    xticklabels={2019, 2020, 2021, 2022, 2023},
    axis lines=left,
    tick style={draw=none},
    grid=major,
    grid style={dashed, gray!30},           % zmiana na {white}
    nodes near coords,
    every node near coord/.append style={
        font=\bfseries\small,               % dodane: yshift=-2em,
        text=white, },
    axis background/.style={fill=white}     % zmiana na {fill=gray!30}
]
\addplot+[ybar, fill=green!70!black] coordinates {
 % zmiana na: \addplot+[ybar, fill=mygreen, draw=none] coordinates {
 % dodatkowo zdefiniowany kolor: \definecolor{mygreen}{RGB}{60,179,113}
    (0,63) (1,132) (2,161) (3,180) (4,157) };
\end{axis}
\end{tikzpicture}
```

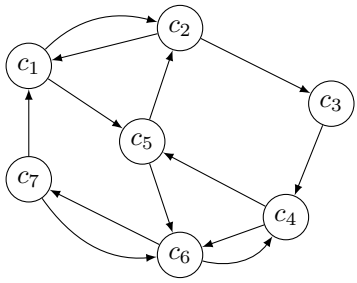




Podsumowując przedstawione przykłady, możemy stwierdzić, że ChatGPT może przyspieszyć przygotowanie wykresu, ale jeżeli chcemy bardzo dokładnie dopracować wykres zgodnie z naszymi oczekiwaniami, to musimy posiadać umiejętność korzystania z pakietu `pgfplots`. Trudno również ocenić, które podejście z dwóch pokazanych dla wykresu słupkowego jest lepsze. W obu przypadkach potrzebujemy jeszcze sporo poprawek, by uzyskać zadowalający efekt.

16.5 Grafika w Ti*k*Z

W porównaniu z generowaniem dowolnych rysunków z użyciem pakietu `tikz` generowanie i ewentualnie poprawianie wykresów wydaje się zadaniem łatwym. W tym rozdziale przyjrzymy się, na ile ChatGPT może przyspieszyć prace z pakietem `tikz`. Od razu trzeba powiedzieć, że zadania „otwarte" typu: *wygeneruj rysunek ilustrujący twierdzenie Pitagorasa, wygeneruj rysunek ilustrujący twierdzenie Talesa, wygeneruj rysunek pokazujący dwie przecinające się płaszczyzny w przestrzeni trójwymiarowej* kończą się całkowitą porażką. Dużo łatwiej byłoby samodzielnie programować takie rysunki od zera, niż poprawiać wytwory czata. Spróbujemy zatem odtworzyć graf przedstawiony na rys. 16.7, który został przygotowany bez żadnego wsparcia AI z użyciem pakietu `tikz`.



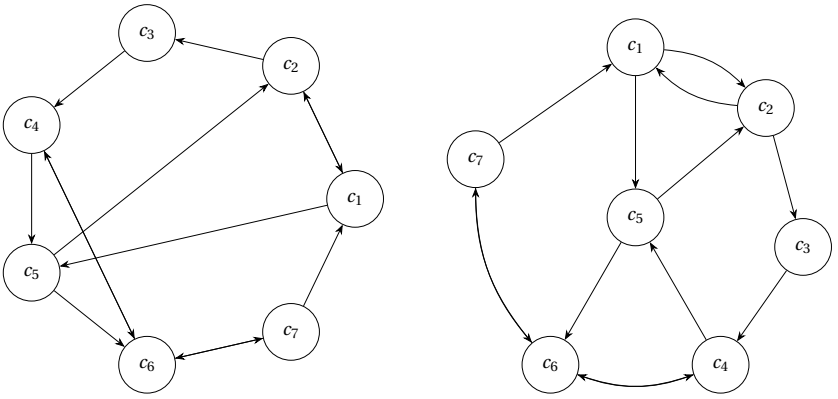
Rys. 16.7. Oryginalny graf, który chcemy odtworzyć

Możemy zacząć od polecenia, które dokładnie opisuje, z jakich węzłów i krawędzi składa się graf: *Wygeneruj rysunek, używając pakietu tikz, który reprezentuje graf: 7 węzłów w kształcie okręgu o etykietach c_1, c_2, c_3, c_4, c_5, c_6, c_7, krawędzie od c_1 do c_2 i c_5, od c_2 do c_1 i c_3, od c_3 do c_4, od c_4 do c_5 i c_6, od c_5 do c_2 i c_6, od c_6 do c_4 i c_7, od c_7 do c_1 i c_6*. Wygenerowany kod (niekompletny) pokazano na listingu 16.12. Widoczne na listingu komentarze zostały zamieszczone przez ChatGPT. Efekt kompilacji tego kodu widzimy na lewym rysunku. Dwa kolejne polecenia poprawiają sytuację: *Ustaw węzeł 5 w środku, a pozostałe na okręgu, Narysuj krawędzie między węzłami 1 i 2, 4 i 6 oraz 6 i 7 tak, aby się nie pokrywały*. Poprawiony graf pokazano na prawym rysunku. Sytuacja jest lepsza, ale nie wszystkie nasze polecenia są poprawnie realizowane przez ChatGPT. Tym niemniej możemy dość szybko poprawić kod ostatniego rysunku, tak aby zbliżyć się do oryginału.

Listing 16.12. Kod wygenerowany przez ChatGPT na podstawie opisu grafu

```
\begin{tikzpicture}[>=Stealth, node distance=3cm,
  every node/.style={circle, draw, minimum size=1cm}]

```

```
% Węzły na okręgu
\foreach \i/\name in {1/$c_1$, 2/$c_2$, 3/$c_3$, 4/$c_4$, 5/$c_5$,
  6/$c_6$, 7/$c_7$} { \node (\i) at ({360/7*(\i-1)}:3cm) {\name};
}
% Krawędzie
\draw[->] (1) -- (2);
\draw[->] (1) -- (5);
%...
\end{tikzpicture}
```

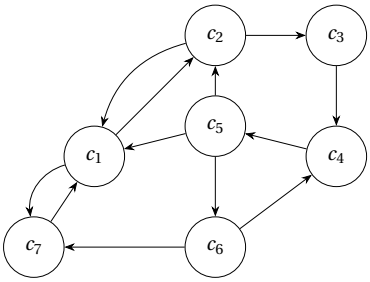


Gorszy wynik uzyskujemy, gdy próbujemy odtworzyć graf na podstawie obrazka. Na listingu 16.13 pokazano niekompletny kod wygenerowany na podstawie rys. 16.7. W tym przypadku widzimy, że kod przedstawia jedynie zasadę konstruowania takich grafów, nie próbuje odtworzyć oryginalnych połączeń. Jeżeli chcemy uzyskać podobne rozmieszczenie węzłów i poprawne połączenia między nimi, to wymaga to sporo dalszej pracy.

Listing 16.13. Kod wygenerowany przez ChatGPT na podstawie istniejącego grafu

```
\begin{tikzpicture}[
    node distance=2cm and 2cm,
    every node/.style={circle, draw, minimum size=1cm},
    ->, >=Stealth
]
% Wierzchołki
\node (c1) at (0,0) {$c_1$};
\node (c2) at (2,2) {$c_2$};
...
% Krawędzie
\draw (c1) -- (c2);
\draw (c2) -- (c3);
...
```

```
% Dodatkowe łuki
\draw (c2) to[bend right=30] (c1);
\draw (c1) to[bend right=40] (c7);
\end{tikzpicture}
```



16.6 Poprawianie tekstu

ChatGPT możemy wykorzystać również do korekty tekstu, na przykład zmiany stylu, skrócenia lub wydłużenia wypowiedzi itp. Punktem wyjścia do eksperymentów w tym podrozdziale jest pierwszy akapit z tekstu z rys. 1.1 (str. 3). Pierwsze polecenie dotyczyło zamiany tekstu na styl akademicki. Wygenerowana wersja prezentuje się następująco:

> Twierdzenie Pitagorasa stanowi jedno z fundamentalnych twierdzeń geometrii euklidesowej i zarazem jeden z najbardziej rozpoznawalnych rezultatów matematycznych, omawianych już na wczesnych etapach edukacji. Nazwa twierdzenia wiązana jest z postacią Pitagorasa z Samos (ok. 572–497 p.n.e.), greckiego filozofa i matematyka, chociaż istnieją uzasadnione wątpliwości co do jego autorstwa. Źródła historyczne oraz badania archeologiczne wskazują, że relacja między długościami boków trójkąta prostokątnego była znana w różnych cywilizacjach (m.in. babilońskiej i egipskiej) na długo przed czasami Pitagorasa. Przykładowo, już w drugim tysiącleciu p.n.e. stosowano tzw. trójkąty pitagorejskie, w tym najprostszy przypadek o bokach 3, 4 i 5 jednostek, jako narzędzie konstrukcyjne do wyznaczania kąta prostego. W związku z powyższym przypuszcza się, iż zasługą Pitagorasa nie było odkrycie samego twierdzenia, lecz jego formalny dowód, co w kontekście greckiej tradycji filozoficznej i matematycznej mogło stanowić wystarczający powód do trwałego związania twierdzenia z jego imieniem.

Jeżeli wydamy polecenia *skróć to do maksymalnie 800 znaków ze spacjami*, to uzyskamy tekst:

> Twierdzenie Pitagorasa, jedno z kluczowych twierdzeń geometrii euklidesowej, nosi nazwę Pitagorasa z Samos (ok. 572–497 p.n.e.), choć jego autorstwo jest kwestionowane. Zależność między bokami trójkąta prostokątnego znana była już w starożytnych cywilizacjach, m.in. w Babilonii, gdzie wykorzystywano trójkąty o bokach 3, 4 i 5 jednostek do wyznaczania kąta prostego. Pitagoras prawdopodobnie jako pierwszy przeprowadził formalny dowód tej zależności, co doprowadziło do nadania twierdzeniu jego imienia.

Z kolei polecenie *rozbuduj ten tekst do 1500 znaków* prowadzi do wersji:

> Twierdzenie Pitagorasa jest zdecydowanie jednym z najpopularniejszych i najczęściej przywoływanych twierdzeń matematycznych, z jakimi spotykamy się już od najmłodszych lat w szkole. Jego nazwa została połączona z nazwiskiem wielkiego filozofa i matematyka greckiego, Pitagorasa z Samos, żyjącego od około 572 roku p.n.e. do około 497 roku p.n.e. Choć twierdzenie nosi jego imię, historyczne źródła wskazują, że sam Pitagoras raczej nie był jego autorem. Wiedza o własnościach trójkątów prostokątnych istniała już w starożytnych cywilizacjach, m.in. w Babilonii czy Egipcie, na długo przed narodzinami greckiego filozofa. Ponad tysiąc lat wcześniej wykorzystywano zależność między bokami trójkąta o długościach 3, 4 i 5 jednostek, by wyznaczać kąty proste, co znalazło zastosowanie np. w geodezji czy budownictwie. Zatem twierdzenie znane było w praktyce, choć niekoniecznie w ujęciu formalnym. Bardziej prawdopodobne jest, że to właśnie Pitagoras jako pierwszy przeprowadził jego ścisły dowód matematyczny w ramach rozwijanej przez swoją szkołę teorii liczb i geometrii. Z tego powodu jego uczniowie i następcy przypisali twierdzeniu jego imię, chcąc w ten sposób uhonorować zasługi swojego mistrza dla rozwoju matematyki. Od tamtej pory twierdzenie to stało się fundamentem geometrii euklidesowej i punktem wyjścia do wielu dalszych rozważań matematycznych.

Analizując te trzy ostatnie wyniki, można śmiało powiedzieć, że ChatGPT dobrze radzi sobie z językiem polskim. Wszystkie trzy wersje prezentują się naprawdę nieźle. Możemy także wykorzystać ChatGPT do połączenia w jedno wskazanych zdań lub podzielenia bardzo długiego zdania na mniejsze. Możemy zamieścić większy fragment tekstu i wygenerować podsumowanie o wskazanej długości lub na podstawie treści artykułu wygenerować streszczenie i tytuł.

Poza poprawianiem samego tekstu możemy korygować skład naszego dokumentu w LaTeX-u. Już wcześniej omawialiśmy przykład zamiany serii wzorów

matematycznych na jedno środowisko **align**. Analogicznie możemy na przykład zaznaczyć kilka akapitów i wydać polecenie: *zamień na środowisko itemize* lub na odwrót – zamienić listę na akapit tekstu. To drugie zadanie nie sprowadza się wyłącznie do usunięcia poleceń LaTeX-a. ChatGPT stara się przekształcić listę w spójny tekst. Weźmy dla przykładu numerowanie z listingu 3.11 (str. 50). Po wydaniu polecenia: *zamień listę na akapit tekstu* otrzymamy:

> Maria Skłodowska-Curie otrzymała Nagrodę Nobla dwukrotnie: w 1903 roku w dziedzinie fizyki oraz w 1911 roku w dziedzinie chemii. W 1905 roku Nagrodą Nobla w dziedzinie literatury został uhonorowany Henryk Sienkiewicz, a w 1924 roku — Władysław Reymont, również w dziedzinie literatury.

Jeżeli wydamy polecenie *Wygeneruj kod wstawiający grafikę z pliku splajn.pdf z podpisem Splajn w trybie edycji, etykietą fig:splajn:edycja i przeskalowany do połowy swojego rozmiaru*, to dostaniemy kompletne środowisko **figure** do umieszczenia w kodzie – zobacz listing 16.14. Jak widać, w tym przypadku wsparcie AI jest perfekcyjne.

Listing 16.14. Kod wygenerowany przez ChatGPT do osadzenia grafiki

```
\begin{figure}[htbp]
  \centering
  \includegraphics[scale=0.5]{splajn.pdf}
  \caption{Splajn w trybie edycji}
  \label{fig:splajn:edycja}
\end{figure}
```

Jeżeli chcemy dodać skorowidz do naszego dokumentu i dodawanie haseł odłożyliśmy na koniec, to ChatGTP może znacznie przyspieszyć naszą pracę. Na listingu 16.15 pokazano hasła skorowidza dodane automatycznie do naszego tekstu o Pitagorasie. Nie należy traktować tego jako wersji finalnej, ale jako solidną pierwszą iterację.

Listing 16.15. Hasła skorowidza dodane przez ChatGPT

```
Twierdzenie Pitagorasa\index{Twierdzenie Pitagorasa} jest zdecydowanie
najpopularniejszym z tych, z jakimi spotykamy się w szkole. Jego nazwa
została połączona z nazwiskiem wielkiego filozofa greckiego
Pitagorasa\index{Pitagoras} żyjącego od ok. 572 roku p.n.e. do ok.
497 roku p.n.e. Sam Pitagoras raczej nie był twórcą tego twierdzenia,
które znane było długo przed jego narodzinami. Ponad tysiąc lat
wcześniej wiedziano, że wykorzystując trójkąt\index{trójkąt
prostokątny}, którego boki mają odpowiednio 3, 4 i 5 jednostek miary
(tzw. trójkąt 3-4-5\index{trójkąt 3-4-5}), możemy wyznaczyć kąt
```

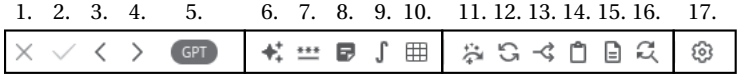
```
prosty\index{kąt prosty}. Bardziej prawdopodobne jest, że Pitagoras
udowodnił znane twierdzenie i dlatego później jego uczniowie nazwali
je na cześć mistrza.
```

Przedstawione przykłady użycia AI do usprawnienia przygotowania dokumentu w LaTeX-u to tylko demonstracja możliwości. Nie jest możliwe podanie kompletnego przeglądu, a wręcz można powiedzieć, że ograniczeniem jest wyłącznie nasza wyobraźnia. Stawiając czoło konkretnemu problemowi, możemy sięgnąć po AI i sprawdzić, na ile będzie pomocna. Na pewno niejeden raz zostaniemy pozytywnie zaskoczeni.

16.7 Wsparcie AI dla Overleaf

We wszystkich wcześniejszych podrozdziałach skupialiśmy się na możliwościach, jakie oferuje ChatGPT. Powód był całkowicie prozaiczny – możemy z tego narzędzia korzystać bez ponoszenia dodatkowych opłat. Jeżeli chcemy korzystać z narzędzi zintegrowanych ze środowiskami takimi jak Overleaf i TeXstudio, to są to opcje płatne. W tym podrozdziale krótko przedstawię rozszerzenie Writefull zintegrowane ze środowiskiem Overleaf.

Writefull funkcjonował do niedawna jako dodatek do przeglądarki, który współpracował ze środowiskiem Overleaf. Jego podstawowym przeznaczeniem było wsparcie dla osób piszących w języku angielskim. Funkcjonował jako dobre narzędzie sprawdzające ortografię, gramatykę i poprawiające styl tekstu. Aktualnie Writefull jest całkowicie zintegrowany z Overleaf. Oferuje m.in. narzędzie określane jako TeXGPT, będące wyspecjalizowanym wariantem ChatGPT zaprojektowanym, aby pomóc użytkownikom w technicznych aspektach LaTeX-a. TeXGPT pomaga tworzyć wzory matematyczne, tabele, formatować tekst, wstawiać grafikę, poprawiać organizację kodu, a także generuje odpowiedzi na pytania dotyczące LaTeX-a i pisania dokumentów naukowych bezpośrednio z poziomu Overleafa. W zasadzie nie oferuje ono niczego istotnie innego, niż moglibyśmy uzyskać bezpośrednio z poziomu ChatGPT, ale dostępne jest bezpośrednio w edytorze Overleaf, bez konieczności kopiowania tekstu poza edytor.



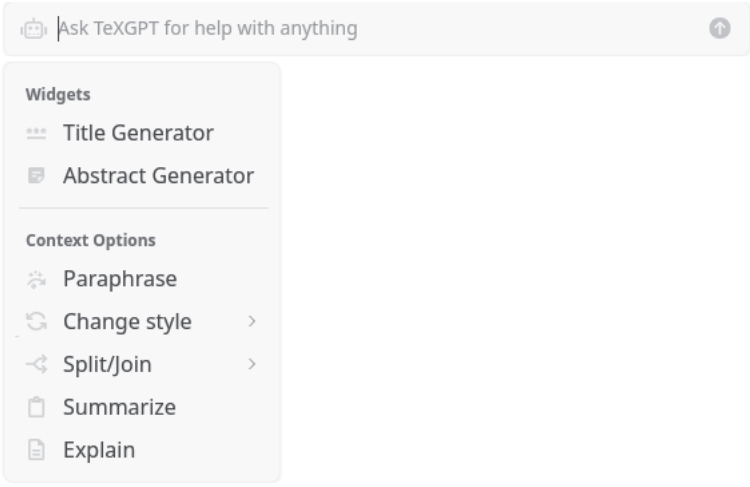
Rys. 16.8. Pasek narzędziowy Writefull

Writefull udostępnia dodatkowy pasek narzędziowy przedstawiony na rys. 16.8. Polecenie 5 pozwala na wybór modelu językowego, którego chcemy używać. Domyślna wartość `WRITEFULL` jest dobrym wyborem, jeśli piszemy tekst

w języku angielskim. Wartość `GPT` pozwala pracować z dowolnym językiem. Dodatkowo dostępna jest też druga opcja, którą możemy uzupełnić naszym poleceniem, np. wskazać, że chcemy poprawić tekst, zachowując znaczenie, ale zmniejszając liczbę znaków.

Pierwsze cztery polecenia paska narzędziowego dotyczą przeglądu sugestii korekt tekstu. Ich znaczenie to odpowiednio: odrzucenie i zaakceptowanie bieżącej sugestii oraz przejście do poprzedniej i następnej sugestii.

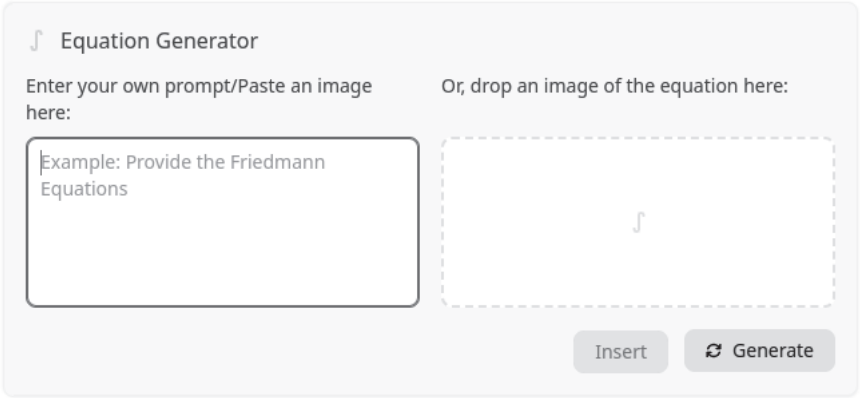
Przycisk 6 aktywuje TeXGPT. Ten sam efekt uzyskamy, ustawiając kursor w pustej linii i naciskając klawisz spacji. Efekt aktywacji widzimy na rys. 16.9. Możemy tutaj wpisywać polecenia podobnie, jak to robiliśmy z ChatGPT. Odpowiedzi są wyświetlane pod pytaniem.



Rys. 16.9. TeXGPT

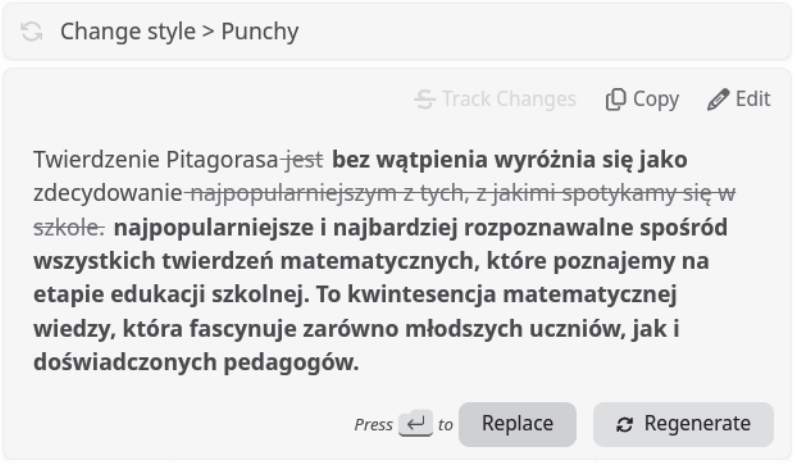
Polecenia 7 (`Title Generator`) i 8 (`Abstract Generator`) pozwalają sfinalizować pracę nad artykułem, generując streszczenie, a na jego podstawie tytuł. Oba polecenia pracują tylko z tekstem angielskim. Warto pamiętać, że wygenerowany tekst należy potraktować wyłącznie jako propozycję. Jako autorzy odpowiadamy za tekst przesłany do publikacji, a treść generowana przez modele językowe nie zawsze jest merytorycznie poprawna.

Polecenie 9 uruchamia generator wzorów matematycznych pokazany na rys. 16.10. Wzór możemy generować na podstawie naszego opisu lub na podstawie wstawionej grafiki. Naciskając klawisz `Enter` lub klikając przycisk `Insert`, możemy wstawić wygenerowany kod do naszego dokumentu. Jeżeli dodanie wzoru wymaga umieszczenia dodatkowego pakietu w preambule, to taka informacja jest przekazywana w postaci komentarza.



Rys. 16.10. Generator wzorów matematycznych

Polecenie 10 uruchamia generator tabel (okienko podobne do generatora wzorów matematycznych) – możemy podać opis tabeli lub wstawić obrazek. Wyniki ogólnie nie są złe, ale ChatGPT wypada nieco lepiej. Polecenie 11 (`Paraphrase`) pozwala przeformułować zdania lub akapity, aby uniknąć powtórzeń lub uzyskać bardziej akademicki ton. Wystarczy zaznaczyć fragment tekstu, który chcemy zmienić i kliknąć przycisk na pasku. Polecenie 12 (`Change style`) można potraktować jako specjalną wersję polecenia `Paraphrase`, która dotyczy tylko zmiany stylu wypowiedzi. Do wyboru mamy jedną z trzech możliwości: `Scientific` (naukowy), `Concise` (zwięzły) i `Punchy` (wyrazisty). Ta ostatnia opcja została użyta w przykładzie pokazanym na rys. 16.11.



Rys. 16.11. Polecenie `Change style`

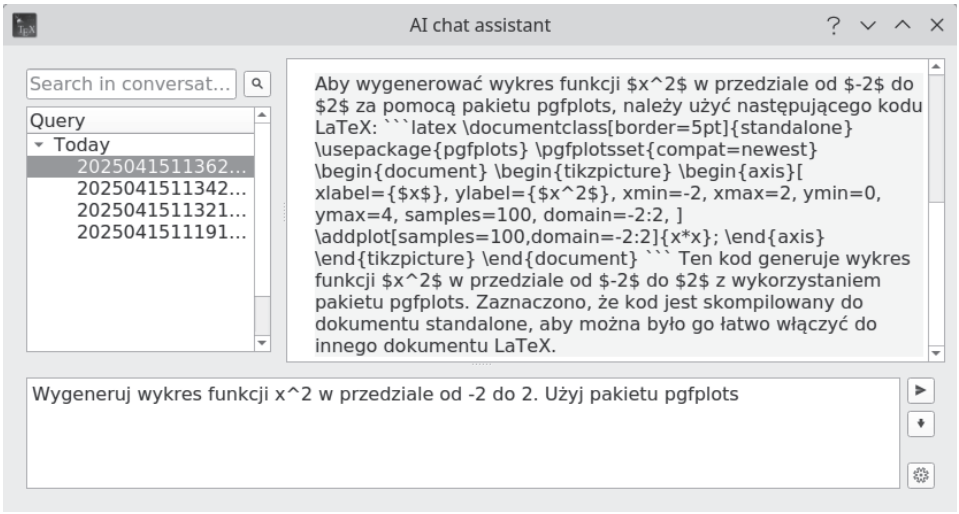
Kolejne polecenie (13 – `Split/join`) pozwala podzielić długie zdanie na dwie części lub połączyć dwa lub więcej zdań. Rodzaj operacji, którą chcemy wykonać, wybieramy po zaznaczeniu tekstu i kliknięciu przycisku. Polecenia `Summarize` (14) i `Explain` (15) stanowią wsparcie przy pracy z materiałami źródłowymi. Pierwsze z nich pozwala uzyskać krótki opis (podsumowanie) zaznaczonego fragmentu tekstu. Dzięki takiemu podejściu zamiast czytać dłuższy fragment, możemy na podstawie podsumowania uzyskać informację, czy jest to tekst, którego szukamy. Drugie polecenie oferuje szczegółowe objaśnienia wyróżnionego tekstu i wykorzystuje otaczającą treść w celu dostarczenia dodatkowych informacji. Oba polecenia generują odpowiedź w języku angielskim, niezależnie od języka źródłowego tekstu, który analizujemy.

Polecenie 16 (`Synonyms in Context` – 16) jest wyszukiwarką synonimów. Możemy zaznaczyć słowo lub frazę (do 5 słów) i po kliknięciu odpowiedniego przycisku otrzymamy zestaw pasujących alternatyw. Procenty umieszczone wraz z alternatywami wskazują semantyczne podobieństwo do oryginalnego wyboru – im wyższy procent, tym bardziej alternatywy przypominają oryginalne znaczenie. Dla przykładu: po zaznaczeniu słowa „filozofa" w naszym tekście o twierdzeniu Pitagorasa dostajemy propozycje: myśliciela 37%, mędrca 33% i uczonego 31%. Możemy kliknąć jedną z alternatyw, aby wstawić ją do tekstu.

16.8 Wsparcie AI dla TeXstudio

W najnowszych wersjach TeXstudio oferuje interfejs dla asystenta czatu AI, który obsługuje między innymi ChatGPT. Integracja z czatem wymaga rejestracji na stronie `https://platform.openai.com/api-keys` i nabycia płatnej subskrypcji, dzięki czemu uzyskamy klucz do API do modeli OpenAI. Koszt korzystania z tego rozwiązania zależy od liczby słów (tokenów) w pytaniu i odpowiedzi, szczegółowe informacje można znaleźć na wskazanej stronie. Dostęp do serwisu ustawiamy, korzystając z polecenia `Opcje→Konfiguruj TeXstudio→Sprawdzanie języka` (sekcja `AI char assistant`). Podajemy kolejno dostawcę AI, klucz do API, adres serwera i nazwę preferowanego modelu.

Jedyną bezpłatną opcją jest skorzystanie z modeli lokalnych, np. użycie któregoś z modeli ze strony `https://github.com/Mozilla-Ocho/llamafile`. Po pobraniu wystarczy uruchomić plik i można korzystać z serwera nasłuchującego na porcie 8080. Jeżeli w ustawieniach wybierzemy dla dostawcy wartość `Local Model`, to TeXstudio będzie z takim modelem współpracować. Przykład takiej sesji pokazano na rys. 16.12. Niestety bez bardzo wydajnego sprzętu współpraca z takim modelem jest mało wydajna. Dodatkowo jakość udzielanej odpowiedzi dla przykładów pokazanych w tym rozdziale jest niższa niż dla ChatGPT.



Rys. 16.12. TeXstudio AI chat assistant

Warto dodać, że samo rozszerzenie nie wprowadza niczego dodatkowego poza możliwością definiowania poleceń dla zaznaczonego tekstu i wstawiania wygenerowanego kodu bezpośrednio do edytora. Zresztą sama forma generowanych odpowiedzi jest mało czytelna i dopiero wstawienie kodu do edytora pozwala ocenić jego jakość.

Podsumowując ten rozdział, warto raz jeszcze podkreślić, że nie możemy ufać bezgranicznie odpowiedziom generowanym przez modele LLM. O ile w przypadku pytań o kod w LaTeX-u mamy szybką weryfikację odpowiedzi – kod się kompiluje i dokument wygląda tak, jak oczekiwaliśmy, albo nie, to jeśli pytamy o kwestie merytoryczne, musimy weryfikować odpowiedzi w oparciu o wiarygodne źródła. Modele AI mają tendencję do zmyślania, gdy nie ma danych. Bez trudu wygenerują życiorys poety, który nigdy nie istniał. Jeżeli więc chcemy wykorzystać rozwiązania AI do wygenerowania jakiegokolwiek tekstu, którego chcemy później użyć w naszych dokumentach, sami jesteśmy odpowiedzialni za zweryfikowanie, czy zawarte w nim informacje są prawdziwe.

Rozdział 17

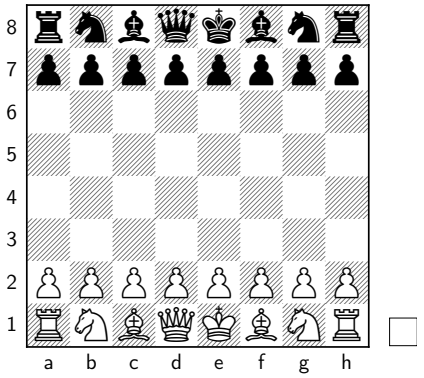
Podsumowanie

Nasza wspólna podróż kończy się w tym miejscu. Nie znaczy to, że o LaTeX-u powiedziałem już wszystko, ale wiesz już tyle, że dalej bez trudu poradzisz sobie sam. Starałem się, aby ta książka była w miarę kompletnym i przede wszystkim praktycznym wykładem. Jeżeli nie potrzebujesz bardzo specyficznych rozwiązań, to jestem przekonany, że przedstawiona tutaj wiedza będzie wystarczająca, aby napisać pracę dyplomową, doktorską, przygotować artykuł naukowy, prezentację na wystąpienie konferencyjne czy nawet wielotomową książkę. Pokazałem Ci też kilka ciekawych narzędzi, których używam od lat i które świetnie sprawdzają się przy opracowywaniu wyników naukowych lub przygotowywaniu materiałów dydaktycznych. Mam nadzieję, że spróbujesz ich użyć i również docenisz ich wyjątkowe zalety.

Używanie LaTeX-a to niekończąca się przygoda – dodatkowo z czasem uzależnia. Mam nadzieję, że wraz z nabywaniem większego doświadczenia będziesz dostrzegał coraz więcej zalet LaTeX-a i cieszyć Cię będzie każda nowa umiejętność z nim związana. Zawsze może pojawić się problem, którego rozwiązania będziesz szukał w internecie, a gdy do niego dotrzesz, to LaTeX zaskoczy Cię istnieniem gotowego rozwiązania, jego prostotą i pięknem. Tysiące pakietów czekają, aby je odkryć. Grasz w szachy? Chciałbyś generować w prosty sposób układy figur na szachownicy? LaTeX ma gotowe rozwiązania – pakiet `xskak`. Wystarczą dwa polecenia pokazane na listingu 17.1, aby wygenerować początkowy układu szachownicy.

Listing 17.1. Generowanie początkowego układu szachownicy z pakietem `xskak`

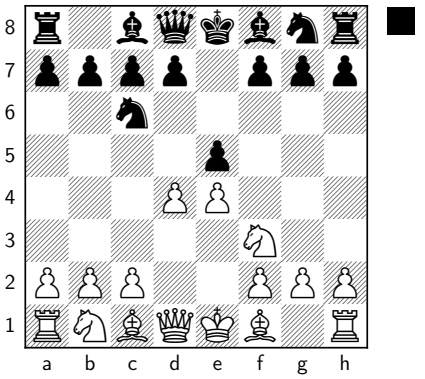
```
1 \newchessgame
2 \showboard
```



Jeżeli chcemy pokazać rozwój akcji w grze, wystarczy dodać sekwencję wykonanych ruchów. Na listingu 17.2 pokazano wygląd szachownicy po wykonaniu pięciu ruchów – partia szkocka. Zajrzyj do dokumentacji pakietu, aby poznać inne jego możliwości. Z pakietem `xskak` [18] bez problemu przygotujesz ciekawy materiał z analizą partii szachowej.

Listing 17.2. Partia szkocka

```
\newchessgame
\mainline{1.e4 e5 2.Nf3 Nc6 3.d4}
\showboard
```



1 e4 e5 2 ♘f3 ♘c6 3 d4

Jesteś fanem sudoku? Składanie tego typu łamigłówek w LaTeX-u jest wyjątkowo proste. Wystarczy sięgnąć po pakiet `sudoku` [1]. Przykład użycia pokazano na listingu 17.3.

Listing 17.3. Sudoku

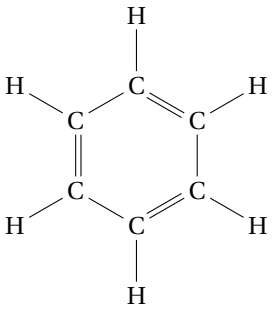
```
\begin{sudoku}
|2|5| | |3| |9| |1|.
| |1| | | |4| | | |.
|4| |7| | | |2| |8|.
| | |5|2| | | | | |.
| | | | |9|8|1| | |.
| |4| | | |3| | | |.
| | | |3|6| | |7|2|.
| |7| | | | | | |3|.
|9| |3| | | |6| |4|.
\end{sudoku}
```

2	5			3		9		1
	1				4			
4		7				2		8
		5	2					
				9	8	1		
	4				3			
			3	6			7	2
	7							3
9		3				6		4

Jesteś chemikiem i potrzebujesz wizualizacji cząsteczek? Sprawdź pakiet `chemfig` [48] – na przykład wystarczy jedna linia kodu, aby wygenerować wizualizację cząsteczki benzenu.

Listing 17.4. Cząsteczka benzenu

```
\chemfig{C*6((-H)-C(-H)=C(-H)-C(-H)=C(-H)-C(-H)=)}
```

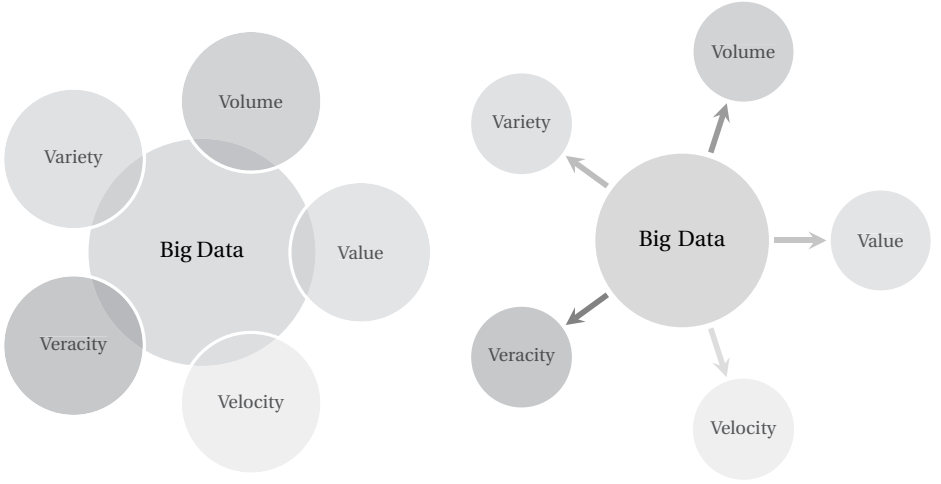


Potrzebujesz szybko wygenerować diagramy na prezentację? Sprawdź możliwości pakietu `smartdiagram` [17]. Bazuje on na pakiecie `tikz`, ale można go używać bez znajomości materiału z rozdziału 11. Jeżeli odpowiadają nam domyślne ustawienia, to w zasadzie generowanie diagramu sprowadza się do jednego polecenia. Pakiet `smartdiagram` bazuje na bardzo prostym założeniu. Użytkownik dostarcza listę elementów, a odpowiednie polecenie zamienia ją w diagram, którego kształt zależy od wybranej opcji.

Na listingu 17.5 pokazano dwa przykłady diagramów wygenerowanych przez polecenie `\smartdiagram` (jest to kompletny kod niezbędny do ich wygenerowania). Reprezentują on słynne litery „V” powiązane z terminem big data. Kod użyty do ich wygenerowania różni się tylko opcją wskazującą typ diagramu. W pakiecie znajdziemy aż 10 różnych typów diagramów.

Listing 17.5. Diagramy wygenerowane przez pakiet `smartdiagram`

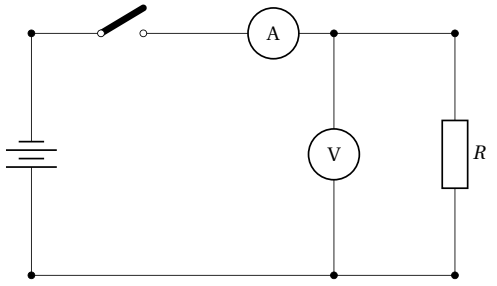
```
\smartdiagram[bubble diagram]
{Big Data, Volume, Variety, Veracity, Velocity, Value}

\smartdiagram[constellation diagram]
{Big Data, Volume, Variety, Veracity, Velocity, Value}
```



Jeżeli zajmujesz się elektroniką i potrzebujesz pakietu do składania schematów układów elektronicznych, sprawdź możliwości pakietu `circuitikz` [42]. Przykład prostego schematu przedstawiono na listingu 17.6. Pakiet zawiera również polecenia pozwalające składać układy bramek logicznych.

Listing 17.6. Schemat układu elektronicznego złożony z użyciem pakietu `circuitikz`

```
\begin{circuitikz}[european]
\draw (0,0) to [battery, *-*] (0,4) to [cute open switch] (3,4)
to [rmeter, t=A] (5,4) to [short, *-*] (7,4) to [R=$R$] (7,0)
to [short, *-*] (5,0) to [short] (0,0);
\draw (5,4) to [rmeter, t=V] (5,0);
\end{circuitikz}
```



Chciałbyś wykorzystać LaTeX-a przy zapisie nut? Zacznij od przeglądu dostępnych pakietów omówionego w [20]. A może chcesz przygotować słownik i potrzebujesz symboli IPA (International Phonetic Alphabet)? Znajdziesz je w pakiecie `tipa` [19]. Wszystkie pokazane w tym podsumowaniu przykłady to tylko wierzchołek góry lodowej. Niezależnie od tego, czego szukasz, masz dużą szansę znaleźć rozwiązania, a także wsparcie od ogromnej rzeszy użytkowników LaTeX-a. Dobrym punktem wyjścia do poszukiwań są strony `https://ctan.org/pkg/latex` oraz `https://tex.stackexchange.com`.

Jeżeli szukasz innych książek do LaTeX-a, to bez trudu znajdziesz w sieci pozycję [40]. Nie ma natomiast książek drukowanych o zasięgu ogólnopolskim wydanych w ostatnich kilkunastu latach. Ostatnią taką była prawdopodobnie książka [13]. Bogata jest natomiast literatura anglojęzyczna. Do ciekawych pozycji z ostatnich lat można zaliczyć m.in. [31], [12] i [30].

Dziękuję za wspólnie spędzony czas. Powodzenia!

Bibliografia

[1] P. Abraham. *The sudoku package*, 2006. https://www.ctan.org/pkg/sudoku.

[2] D. Arseneau. *The placeins package*, 2005. https://www.ctan.org/pkg/placeins.

[3] J.G. Bazan, M. Szpyrka, A. Szczur, Ł. Dydo, and H. Wojtowicz. Classifiers for behavioral patterns identification induced from huge temporal data. *Fundamenta Informaticae*, 143(1–2):19–34, 2016.

[4] C. Beccari. *The wrapfig2 package*, 2024. https://www.ctan.org/pkg/wrapfig2.

[5] J. Bezos. *The titlesec, titleps and titletoc Packages*, 2025. https://ctan.org/pkg/titlesec.

[6] D. Carlisle. *The dcolumn package*, 2023. https://ctan.org/pkg/dcolumn.

[7] D. Carlisle. *The colortbl package*, 2024. https://ctan.org/pkg/colortbl.

[8] D. Carlisle. *The longtable package*, 2024. https://www.ctan.org/pkg/longtable.

[9] D.P. Carlisle. *Packages in the 'graphics' bundle*, 2024. https://ctan.org/pkg/graphicx.

[10] S.D. Cochran. *The subfig package*, 2005. https://ctan.org/pkg/subfig.

[11] P.W. Daly. *Natural Sciences Citations and References*, 2010. https://ctan.org/pkg/natbib.

[12] D. Datta. *LaTeX in 24 hourse. A Practical Guide for Scientific Writing*. Springer, Cham, Switzerland, 2017.

[13] A. Diller. *LaTeX wiersz po wierszu*. Helion, Gliwice, 2001.

[14] R. Fairbairns, S. Rahtz, and L. Barroca. *A package for rotated objects in LaTeX*, 2024. https://www.ctan.org/pkg/rotating.

[15] S. Fear. *Publication quality tables in LaTeX*, 2020. https://ctan.org/pkg/booktabs.

[16] C. Feuersänger. *Manual for package PGFPlots*, 2021. https://ctan.org/pkg/pgfplots.

[17] C. Fiandrino. *The smartdiagram package*, 2016. https://ctan.org/pkg/smartdiagram.

[18] U. Fischer. *xskak: An extension to the package skak*, 2019. https://www.ctan.org/pkg/xskak.

[19] R. Fukui. *TIPA Manual*, 2004. https://www.ctan.org/pkg/tipa.

[20] G. Gonzato. *LaTeX for Musicians*, 2019. https://www.ctan.org/pkg/latex4musicians.

[21] E. Gregorio. *The package imakeidx*, 2016. https://ctan.org/pkg/imakeidx.

[22] E. Gregorio. *The xpatch package. Extending etoolbox patching commands*, 2020. https://ctan.org/pkg/xpatch.

[23] T. Górecki. *Podstawy statystyki z przykładami w R*. BTC, Legionowo, 2011.

[24] C. Heinz, B. Moses, and J. Hoffmann. *The listings package*, 2024. https://ctan.org/pkg/listings.

[25] P.G. Hewitt. *Fizyka wokół nas*. PWN, Warszawa, 2024.

[26] M. Høgholm and L. Madsen. *The mathtools package*, 2024. https://ctan.org/pkg/mathtools.

[27] A. Kassambara. *Practical Guide to Cluster Analysis in R*. STHDA, 2017.

[28] U. Kern. *Extending LaTeX's color facilities: the xcolor package*, 2024. https://ctan.org/pkg/xcolor.

[29] P. Kime, M. Wemheuer, and P. Lehman. *The biblatex package. Programmable Bibliographies and Citations*, 2024. https://www.ctan.org/pkg/biblatex.

[30] S. Kottwitz. *LaTeX Cookbook*. Packt Publishing, Birmingham, UK, 2015.

[31] S. Kottwitz. *LaTeX Beginner's Guide*. Packt Publishing, Birmingham, UK, 2021.

[32] S. Kottwitz. *LaTeX Graphics with TikZ*. Packt Publishing, Birmingham, UK, 2023.

[33] W. Krysicki and L. Włodarski. *Analiza matematyczna w zadaniach*. PWN, Warszawa, 1993.

[34] O. Lapko. *The makecell package*, 2009. https://ctan.org/pkg/makecell.

[35] A. Lingnau. *An Improved Environment for Floats*, 2001. https://www.ctan.org/pkg/float.

[36] C. Matt. *Algpseudocodex Package Documentation*, 2023. https://ctan.org/pkg/algpseudocodex.

[37] A. Matthes. *The Ornaments package*, 2024. https://www.ctan.org/pkg/pgfornament.

[38] A. Matthias. *The pdfpages package*, 2025. https://ctan.org/pkg/pdfpages.

[39] F. Mittelbach and D. Carlisle. *A new implementation of LaTeX's tabular and array environment*, 2024. https://ctan.org/pkg/array.

[40] T. Oetiker, H. Partl, I. Hyna, and E. Schlegl. *The Not So Short Introduction to LaTeX 2e*, 2022. (Polskie tłumaczenie wersji z 2006 r.: Przechlewski T., Kubiak R., Gołdasz J., Serwin M. *Nie za krótkie wprowadzenie do systemu LaTeX2e*, 2007).

[41] S. Pakin. *The Comprehensive LaTeX Symbol List*, 2024. https://ctan.org/pkg/comprehensive.

[42] M.A. Radaelli, S. Lindner, S. Erhardt, and R. Giannetti. *CircuiTikZ*, 2025. https://ctan.org/pkg/circuitikz.

[43] A. Sommerfeldt. *Customizing captions of floating environments*, 2023. https://ctan.org/pkg/caption.

[44] M. Szpyrka. *Sieci Petriego w modelowaniu i analizie systemów współbieżnych*. WNT, Warszawa, 2008.

[45] M. Szpyrka. *Modelowanie systemów współbieżnych w języku Alvis*. Wydawnictwa AGH, Kraków, 2013.

[46] T. Tantau. *TikZ & PGF Manual*, 2013. `https://www.ctan.org/pkg/pgf`.

[47] T.. Tantau, J. Wright, and V. Miletić. *The beamer class. User Guide for version 3.72 package*, 2025. `https://ctan.org/pkg/beamer`.

[48] Ch. Tellechea. *chemfig. A TeX package for drawing molecules*, 2023. `https://www.ctan.org/pkg/chemfig`.

[49] P. van Oostrum. *The fancyhdr and extramarks packages*, 2025. `https://ctan.org/pkg/fancyhdr`.

[50] P. van Oostrum, Ø. Bache, and J. Leichter. *The multirow, bigstrut and bigdelim packages*, 2024. `https://ctan.org/pkg/multirow`.

[51] O. Vheong. *Ipe*, 2024. `https://ipe.otfried.org/ipe-manual.pdf`.

[52] J. Wright. *siunitx – A comprehensive (SI) units package*, 2025. `https://ctan.org/pkg/siunitx`.

Skorowidz

A

akapit 39
apostrof wsteczny 32
autouzupełnianie 353, 361

B

biber 86, 194
biblioteka
— `arrows.meta` 240
— `calc` 263
— `fillbetween` 293
— `intersections` 237
— `polar` 280
— `positioning` 247
— `shapes` 243
— `shapes.callouts` 265

C

cudzysłów 32
— angielski 32
— francuski 32
— niemiecki 32
— polski 33
czcionka 19
— Arev 23, 25
— Bera Mono 23, 25–27
— bezszeryfowa 19, 25
— Bookman 24
— Computer Modern 21
— Computer Modern Roman 23
— Computer Modern Sans Serif 23
— Computer Modern Typewriter 23
— Courier 23, 25, 26
— domyślna 17
— Fourier 23, 25
— Helvetica 23–25, 27
— Inconsolata 23, 26
— Kepler Fonts 22
— Kerkis 23, 25
— Kerkis Sans Serif 23
— Kp-Fonts 22, 23
— Kp-Fonts Sans Serif 23
— Kp-Fonts Typewriter 23
— Kurier 23, 26
— Latin Modern 21
— Latin Modern Roman 23
— Latin Modern Sans Serif 23
— Latin Modern Typewriter 23
— LobsterTwo 27
— Miama Nueva 27
— nazwa kodowa 22
— New Century Schoolbook 23, 24

— Palatino 23, 24
— szeryfowa 19, 24
— TEX Gyre Adventor 23
— TEX Gyre Bonum 23, 24
— TEX Gyre Chorus 23, 27
— TEX Gyre Cursor 23
— TEX Gyre Heros 23, 25
— TEX Gyre Pagella 23, 24
— TEX Gyre Schola 23, 24
— TEX Gyre Termes 23, 24
— Times Roman 7, 23, 24, 27
— Times Roman Typewriter 23
— Utopia 23, 25

D

diagram strony 76
dywiz 33

E

etykieta 8, 71, 119, 132, 135, 158, 161, 184, 207, 273, 321, 327

H

hiperłącze 345

I

indeks 289, 337
— dolny 16, 120, 139
— górny 9, 16, 120, 139
interlinia 19, 44
ipelet 219

J

jednostka
— em 38
— ex 38
— in 38
— pt 38

K

klasa
— `article` 6, 63, 64, 76, 330
— `beamer` 307
— `book` 63, 68, 73
— dokumentu 2, 6, 63
— `letter` 63, 69
— `report` 63, 67
— `standalone` 262, 280
kodowanie
— cp1250 6
— latin2 6
— utf8 6, 172, 338
komentarz 5, 7, 17, 188
— magiczny 365
— TODO 365
kompilacja 5, 14
krój pisma 19
krzywa
— Béziera 203, 238
— sklejana 203
kształt czcionki 19
kursor Fifi 209

L

licznik 50, 163
— `beamerpauses` 316
— `enumi` 50
— `enumii` 50
— `enumiii` 50
— `enumiv` 50
— `section` 66
linia bazowa 54, 90
litera diakrytyzowana 35, 173

Ł

łącznik 33

M

makro 367
mapa kolorów 284
margines 56, 67, 258, 262, 358
— lewy 67, 76, 97
— prawy 67, 76, 97
— wewnętrzny 68, 76
— zewnętrzny 68, 76
metadane 201, 284, 301, 347
mieszanie kolorów 30
ministrona 31, 38, 54, 151, 208
model kolorów 29
motyw
— czcionek 324
— kolorów 324
— prezentacji 308, 323
— wewnętrzny 324
— zewnętrzny 324
myślnik 33

N

nazwa kodowa czcionki 22

O

obiekt pływający 8, 109
odsyłacz 8, 71
ogranicznik 126
— pusty 128

P

pakiet 6
— algorithm 180
— algpseudocodex 177
— amsfonts 118
— amsmath 118
— amssymb 118
— amsthm 161
— arev 23, 25
— arevtext 25
— array 89, 93
— babel 6, 33, 65, 118
— beamer 307
— beramono 23, 26, 27
— biblatex 193
— booktabs 91
— caption 158, 334
— chemfig 395
— color 28
— colortbl 92
— courier 23, 26
— dcolumn 98
— enumitem 46, 51
— fancyhdr 330
— float 110, 150
— fontenc 6
— fourier 23
— fouriernc 25
— geometry 78, 358
— graphicx 6, 145
— helvet 23, 24, 26
— hyperref 345
— imakeidx 338
— inconsolata 23, 26
— indentfirst 39
— inputenc 6
— kerkis 23, 25
— kmath 25
— kpfonts 22, 23, 27
— kurier 23, 26
— latex4musicians 397
— layouts 47, 76
— listings 167
— lmodern 22, 23
— LobsterTwo 27
— longtable 112
— makecell 102
— makeidx 338
— mathtools 118

— miama 26
— multirow 100
— natbib 193
— newcent 23, 25
— newlfm 70
— newpx 23
— newpxtext 23
— newtx 7, 23, 24, 27
— newtxtext 23, 24
— pdfpages 156
— pgfornament 325
— pgfplots 271
— pgfplotstable 287
— pifont 36
— placeins 149
— poemscl 43
— polski 7, 126
— ragged2e 42
— rotating 105
— setspace 44
— sfmath 26
— siunitx 142
— smartdiagram 395
— stackrel 123
— subfig 153
— sudoku 394
— tcolorbox 57, 95
— tgadventor 23
— tgbonum 23
— tgchorus 23, 27
— tgcursor 23
— tgheros 23
— tgpagella 23
— tgschola 23
— tgtermes 23
— tikz 231, 325
— tipa 397
— titlesec 79, 333
— titletoc 83, 333
— tocloft 83
— ulem 27
— url 34
— utopia 23
— verbatim 167
— wrapfig2 152
— xcolor 28, 90
— xpatch 335
— xskak 393
parametr
— obowiązkowy 5
— opcjonalny 5
pauza 33
pdflatex 9, 145
pętla \foreach 253
plik
— aux 86
— bbl 86, 190
— bcf 86
— bib 86, 185
— blg 86
— bst 86, 191
— cls 86
— główny 73
— idx 86, 339
— ilg 86
— ind 86, 339
— jpeg 146, 229
— loa 335
— lof 86, 335
— log 86
— lol 335
— lot 86, 335
— pdf 10, 109, 145, 199, 224, 262, 307, 345
— png 146, 229, 325
— pomocniczy 354
— sty 86
— tex 4, 74, 145
— tiff 146
— toc 86

— tymczasowy 14, 334
— wynikowy 5
— xdy 340
— xml 199
— zip 14
— źródłowy 4, 72, 109
podpodsekcja 66
podsekcja 66
polecenie 5
— \! 142
— \\ 32, 95
— * 32
— \, 142
— \: 142
— \; 142
— <{} 93
— >{} 93
— @{} 97
— \addbibresource 193
— \addcontentsline 83
— \addplot 272, 293
— \addplot3 297
— \address 69
— \addtocontents 83
— \addvspace 85
— \againframe 321
— \alert 314, 315
— \algorithmicensure 179
— \algorithmicrequire 179
— \algrenewcommand 180
— \Alph 51
— \alph 51
— \ang 143
— \appendix 75
— \arabic 51
— \arraybackslash 95
— \arraycolsep 131
— \arraystretch 110
— \author 65
— \autocite 197
— \backmatter 68, 75
— \backslash 16, 126
— \bar 135, 138
— \baselineskip 44
— \BeginBox 178
— \bfseries 20
— \bibitem 184
— \bibliography 189
— \bibliographystyle 189
— \bibname 191
— \Biggl 128
— \biggl 128
— \Biggr 128
— \biggr 128
— \Bigl 128
— \bigl 128
— \Bigr 128
— \bigr 128
— \bottomrule 91
— \Call 178
— \caption 9, 71, 109, 113, 149, 158
— \captionsetup 158
— \cc 69
— \Centering 42
— \centering 8, 42
— \chapter 67
— \chapter* 82
— \chaptermark 333
— \cite 184, 197, 345
— \citep 193
— \cleardoublepage 74
— \clearpage 74
— \cline 100
— \clip 258
— \closing 69
— \colon 124
— \color 29, 315
— \colorbox 30
— \column 311

— \columncolor 93
— \Comment 178
— \contentslabel 85
— \contentspage 85
— \contentsuse 334
— \coordinate 235
— \coordindex 289
— \dashuline 28
— \date 65, 69
— \ddddot 138
— \dddot 138
— \ddot 138
— \ddots 129
— \DeclareMathOperator 126
— \definecolor 29
— \diaghead 107
— \documentclass 6
— \dot 138
— \dotfill 37
— \dots 129
— \dotuline 28
— \doublespacing 44
— \Downarrow 127
— \downarrow 127
— \draw 232
— \Else 178
— \ElsIf 178
— \emph 27, 164, 315
— \EndBox 178
— \endfirsthead 115
— \endfoot 116
— \EndFor 178
— \EndFunction 178
— \endhead 116
— \EndIf 178
— \endlastfoot 116
— \EndLoop 178
— \EndProcedure 178
— \EndWhile 178
— \Ensure 177
— \ensuremath 343
— \eqref 119, 345
— \evensidemargin 76
— \familydefault 26
— \fancyfoot 330
— \fancyhead 330
— \fancyhf 330
— \fbox 36, 56
— \fboxsep 56
— \fcolorbox 30
— \figurename 160
— \filcenter 82
— \fill 37, 246
— \filleft 82
— \filright 81
— \fmmfamily 26
— \fontfamily 24
— \fontsize 18
— \footnote 327, 345
— \footnotemark 327
— \footnotesize 18
— \footnotetext 327
— \footrulewidth 331
— \For 178
— \ForAll 178
— \foreach 253
— \frac 120
— \framebox 36
— \framesubtitle 310
— \frametitle 310
— \frenchspacing 40
— \frontmatter 68, 74
— \Function 178
— \Gape 103
— \gape 103
— \geometry 78
— \graphicspath 72, 145
— \hat 138
— \headheight 78
— \headrulewidth 331

— \headsep 78
— \heavyrulewidth 91
— \hfill 38
— \hline 90
— \href 345
— \hspace 37
— \hspace∗ 37
— \Huge 18
— \huge 18
— \hypersetup 346
— \hyphenation 33
— \If 178
— \iiint 137
— \iint 137
— \imath 139
— \include 74
— \includegraphics 8, 88, 145, 316
— \includeonly 75
— \includepdf 156
— \index 338
— \input 75
— \institute 310
— \int 137
— \intertext 137
— \item 45, 316
— \itshape 16, 20
— \jmath 139
— \justifying 42
— \k 35
— kruche 83
— \L 35
— \l 35
— \label 8, 71, 109, 149
— \langle 125, 127
— \LARGE 18
— \Large 18
— \large 18
— \LaTeX 7
— \lceil 127
— \left 127
— \leftmark 331
— \let 95
— \lfloor 127
— \lgroup 127
— \lightrulewidth 91
— \limits 120
— \linebreak 32
— \listofalgorithms 333
— \listoffigures 333
— \listoftables 333
— \lmoustache 127
— \longindentation 69
— \Loop 178
— \lstdefinelanguage 175
— \lstinline 168
— \lstinputlisting 168
— \lstlistoflistings 333
— \lstloadlanguages 167
— \lstset 168
— \mainmatter 68, 74
— \makebox 36
— \makecell 102
— \makecell∗ 102
— \makeindex 338
— \maketitle 65
— \marginpar 329
— \markboth 332
— \markright 332
— \mathbb 122
— \mathbf 122
— \mathcal 122
— \mathclap 124
— \mathfrak 122
— \mathindent 117
— \mathit 122
— \mathllap 125
— \mathnormal 122
— \mathrlap 125
— \mathrm 122

— \mathsf 122
— \mathtt 122
— \mbox 36
— \mdseries 20
— \mid 126
— \midrule 91
— \multicolumn 99
— \multirow 100
— \newcolumntype 94
— \newcommand 342
— \newenvironment 343
— \newline 32, 95
— \newpage 32
— \newtheorem 161
— \newtheoremstyle 164
— \nocite 191
— \node 244
— \noindent 56
— \nolimits 120
— \nolinebreak 32
— \nonumber 135
— \nopagebreak 32
— \nopagecolor 30
— \normalfont 21
— \normalsize 18
— \notag 135
— \num 142
— \oddsidemargin 76
— \onehalfspacing 44
— \only 317
— \opening 69
— \Output 178
— \overbrace 139
— \overbracket 142
— \overleftarrow 138
— \overleftrightarrow 138
— \overline 138
— \overrightarrow 138
— \pagebreak 32
— \pagecolor 30
— \pagediagram 76
— \pageref 72, 345
— \pagestyle 330
— \pagevalues 76
— \par 38
— \parallel 126
— \parindent 40
— \parskip 40
— \part 66, 67
— \path 243
— \pause 315
— \pdfbookmark 347
— \pgfplotstableread 287
— \phantom 140
— \printbibliography 194
— \printindex 338
— \Procedure 178
— \protect 83
— \ps 69
— \qquad 128, 142
— \quad 128, 142
— \RaggedLeft 42
— \raggedleft 42
— \RaggedRight 42
— \raggedright 41
— \rangle 125, 127
— \rceil 127
— \ref 8, 17, 72, 119, 345
— \refname 191
— \renewcommand 21, 26, 345
— \Repeat 178
— \Require 177
— \Return 178
— \rfloor 127
— \rgroup 127
— \right 127
— \rightmark 331
— \rmdefault 21
— \rmfamily 19
— \rmoustache 127

— \Roman 51
— \roman 51
— \rotcell 105
— \rothead 105
— \rowcolor 92
— \rowcolors 92
— \scriptsize 18
— \scshape 20
— \section 7, 66, 67
— \section* 66
— \sectionmark 333
— \selectfont 18, 24
— \setbeamertemplate 310
— \setcounter 82
— \setlength 40, 96
— \setlist 47, 51, 53
— \setminus 126
— \setstretch 44
— \sfdefault 21, 26
— \sffamily 19
— \sideset 121
— \signature 69
— \sin 126
— \singlespacing 44
— \sisetup 143
— \slshape 20
— \small 18
— \sout 28
— \stackbin 123
— \stackrel 123
— \State 177
— \Statex 178
— \subfloat 154
— \subitem 340
— \subsection 66, 67
— \substack 124
— \subsubitem 340
— \subsubsection 66, 67
— \sum 120
— \tabcolsep 96
— \tablename 160
— \tableofcontents 74, 82, 321
— \tag 135
— \tag* 135
— \tbclower 57
— \tcbox 61
— \tcboxmath 62
— \tcbset 59
— \tcbsubtitle 60
— \texorpdfstring 348
— \text 122
— \textbackslash 16
— \textbf 20, 315
— \textcolor 28, 315
— \textheight 37
— \textit 9, 20, 315
— \textmd 20, 315
— \textnormal 315
— \textrm 19, 315
— \textsc 20, 315
— \textsf 19, 315
— \textsl 20, 315
— \texttt 19, 315
— \textup 20, 315
— \textwidth 37
— \thanks 65
— \thead 104
— \thead* 105
— \theenumi 52
— \theenumii 52
— \theoremstyle 162
— \thesection 66
— \thispagestyle 330
— \tikz 262
— \tikzset 251
— \tilde 138
— \tiny 18
— \title 65
— \titlecontents 84, 335
— \titleformat 79, 83

— \titlepage 308
— \titlerule 85
— \titlerule* 85
— \titlespacing 79, 83
— \topmargin 78
— \toprule 91
— \ttdefault 21
— \ttfamily 19
— \uline 28
— \uncover 317
— \underbrace 139
— \underbracket 142
— \underleftarrow 138
— \underleftrightarrow 138
— \underline 27, 138
— \underrightarrow 138
— \unit 143
— \Until 178
— \Uparrow 127
— \uparrow 127
— \Updownarrow 127
— \updownarrow 127
— \upshape 20
— \url 34, 345
— \usecolortheme 324
— \usefonttheme 324
— \useinnertheme 324
— \useoutertheme 324
— \usepackage 6
— \usetikzlibrary 237
— \uuline 28
— \uwave 28
— \vdots 129
— \vec 138
— \verb 166
— \verb* 167
— \Vert 126, 127
— \vert 126, 127
— \vfill 38
— \vspace 37
— \vspace* 37
— \While 178
— \widehat 138
— \widetilde 138
— \xout 28
— \xpatchcmd 337
— \xpretocmd 336
półpauza 33
preambuła 5, 64, 69, 73, 201
przypis 327
pusty komentarz 17

R

ramka 36
rodzina
— czcionki 19
— monospaced 19
— sans serif 19
— serif 19
rozdział 67
rozmiar
— bazowy czcionki 17
— czcionki 17
rysunek
— obracanie 147
— przycinanie 148
— skalowanie 146

S

scena 224, 321
sekcja 66
seria czcionki 19
skala logarytmiczna 277
skorowidz 337
sortowanie bibliografii 197
specyfikacja scen 315
spis treści 82
splajn 203
— kardynalny 203

— klotoidalny 203
— spiro 203
streszczenie 65
styl 250
— acm 192
— alpha 192
— alphabetic 194
— apalike 192
— authortitle 194
— authoryear 194
— bibliografii 189, 195
— cytowania 194
— czcionki 19
— definition 162
— empty 330
— fancy 330
— headings 330
— numeric 194
— plain 162, 192, 330
— remark 162
— strony 330
— verbose 194
szeryf 19

Ś

ścieżka 205, 231
środowisko 5
— abstract 65
— alertblock 314
— algorithmic 177
— align 132
— align* 132
— array 129
— axis 271
— block 314
— Center 42
— center 42
— column 311
— columns 311
— comment 167
— definition 315
— description 45, 49, 316
— description* 53
— document 7
— doublespace 45
— enumerate 45, 49, 316
— enumerate* 53
— equation 118
— equation* 119
— exampleblock 314
— figure 8, 149
— figure* 156
— FlushLeft 42
— flushleft 41
— FlushRight 42
— flushright 42
— frame 308
— gather 133
— gather* 133
— itemize 45, 316
— itemize* 53
— letter 69
— longtable 112
— lstlisting 168
— minipage 31, 54, 150, 208
— onehalfspace 45
— overlayarea 321
— polaraxis 271, 280
— proof 165, 315
— quotation 42
— quote 42
— sidewaysfigure 155
— sidewaystable 112
— singlespace 45
— spacing 45
— split 136
— subequations 134
— table 108
— table* 156

— `tabular` 87, 108
— `tcblisting` 173
— `tcolorbox` 57, 95
— `thebibliography` 183
— `theindex` 339, 342
— `theorem` 315
— `tikzpicture` 232, 271
— `verbatim` 166
— `verbatim*` 167
— `verse` 42
— `wrapfigure` 152
— `wraptable` 152
— `wraptext` 152

T

TeX Live 9, 14, 349, 356
tryb
— akapitowy 31
— eksponowany 117
— matematyczny 9, 16, 117
— przyciągania 209
— tekstowy LR 30
— wewnątrzakapitowy 117
— wierszowy 30, 36
twarda spacja 17

U

układ
— biegunowy 271
— kartezjański 271
— współrzędnych 211, 271, 274, 304
ukośnik wsteczny 5, 15

W

warstwa 222
wcięcie akapitowe 39
węzeł 231, 243
widok 224
współrzędne
— biegunowe 234
— kartezjańskie 233
wstawka 109, 149, 171, 180, 262
wydruk
— dwustronny 76
— jednostronny 76
wykres
— punktowy 285
— słupkowy 287
— trójwymiarowy 297

X

xindy 339

Z

zakładka 345, 347, 365
zmienna sterująca 253
znak
— ampersand 17
— daszek 16
— diakrytyczny 7, 34
— dolara 16
— grawis 32
— końca akapitu 7
— kratka 17
— ogonek 35
— podkreślenia 16
— procentu 17
— specjalny 15
— średnika 233
— tyldy 17, 34
— ucieczki 15

PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion