Technologie informacyjne
przed 6 wykładem

Andrzej Giniewicz

03.04.2024

Dziś zaczniemy cykl trzech wykładów o podstawach składu tekstu w systemie LaTeX. Zajmiemy się podstawowym formatowaniem tekstu. Za tydzień wzory matematyczne a za dwa tygodnie ilustracje, tabele i bibliografia. Informacje zawarte w tym bloku tyczą się głównie składu materiałów do druku, takich jak artykuły, raporty, książki i prace dyplomowe.

1 TeX i LaTeX

Historia systemu TeX (czytamy „tech" od $\tau\varepsilon\chi$, greckiego słowa oznaczającego sztukę i technologię — zapisywanego za pomocą liter tau, epsilon, chi) jest dość długa jak na narzędzie informatyczne rozwijane obecnie. Autorem TeXa jest Donald Knuth (czytamy „Knut"), matematyk i informatyk, autor cenionej serii *Sztuka programowania*. Pierwszy tom serii wydany w 1968 roku składany był metalowymi czcionkami firmy Monotype. Drugie wydanie, nad którym prace kończyły się w 1977 roku, miało być złożone techniką fotoskładu, jednakże jakość druku była niesatysfakcjonująca dla autora, który postanowił zainteresować się typografią i sam stworzyć system składu, który byłby zadowalający. Już w 1978 roku powstała pierwsza wersja systemu o nazwie TeX78. Druga wersja powstała w 1982 roku i nazywana była TeX82 — wprowadzono wtedy do systemu pełnoprawny język programowania. Trzecia wersja ukazała się w 1989 roku i nazwana była TeX90 — dzięki rozszerzeniu kodowania znaków do 8 bitów, możliwe na niej stało się korzystanie z systemu w innych językach niż angielski. Od czasu TeX90 różnice w samym sercu programu do składu są niewielkie. Odzwierciedla to zmiana systemu numeracji. TeX90 otrzymał numer 3,0, natomiast każda kolejna wersja jest coraz bliżej liczby π. Obecnie wersja TeXa to 3,141592653 wydana w 2021.

Pieczołowitość i precyzja, z jaką zaprojektowany został TeX, jest powodem, dla którego stał się jednym z wiodących systemów składu tekstu, w pewnych aspektach niedoścignionym. Po pierwsze, algorytm dzielenia wyrazów na akapity, nazywany algorytmem Knutha-Plassa, tworzy dużo ładniej wyglądający tekst justowany, niż algorytmy stosowane dziś w przeglądarkach lub edytorach tekstu z pakietów biurowych. Prosty algorytm z przeglądarki lub pakietu biurowego pracuje linijkami. Umieszcza tyle słów w linii, ile się zmieści, następne słowo dzieli zgodnie z regułami przenoszenia wyrazów, po czym całą pozostałą przestrzeń rozciąga. Następnie przechodzi do nowej linii. TeX nie patrzy lokalnie, na linie, ale na

całe akapity. Spośród wszystkich możliwych podziałów akapitu na linie, wybiera ten, który ma najmniejszą sumę kwadratów miejsca, które musi rozciągnąć, aby wyrównać prawy margines. Takie podejście powoduje, że zamiast lokalnie najlepszego rozwiązania, znajduje globalnie najlepsze rozwiązanie, dając w wyniku bardziej równomierne i przyjemniejsze dla oka połacie tekstu[1]. Do tego system TEX wspiera kerning (regulację odstępów pomiędzy literami takimi jak para „AV", która ma ujemny odstęp, aby „V" weszło nad „A"), ligatury, wiele grubości i rodzajów pisma, funkcje mikrotypograficzne i wiele więcej. Co prawda te same funkcjonalności są w innych profesjonalnych programach do składu tekstu, na przykład Adobe InDesign, Microsoft Publisher, QuarkXPress lub Scribus[2] (kolejność alfabetyczna), to jednak jest pewien aspekt, w którym programy te nie doścignęły TEXa — skład wzorów matematycznych. Ilość pracy, która została włożona w narzędzia w środowisku TEXa, jest ogromna. Z punktu widzenia matematycznego, o wiele łatwiej też napisać `\int` i uzyskać symbol całki, niż wyszukiwać właściwego symbolu klikając, co daje dużą przewagę TEXowi nad narzędziami graficznymi, w których wyrażenia matematyczne uzyskujemy za pomocą klikania w ogromne palety symboli. Warto też zapoznać się z tym, jakie zasady są wykorzystywane w kompilacji wzorów matematycznych[3,4]. Z powodu precyzji i tego, że przez długi czas był jedynym rozwiązaniem, TEX stał się de facto standardem składu wzorów matematycznych. LATEX jest obecnie najczęściej stosowanym standardem, jeśli chodzi o publikacje artykułów naukowych i książek naukowych posiadających wzory. Dominuje w publikacjach w dyscyplinach takich jak matematyka, statystyka, informatyka, fizyka, chemia i wiele innych, w których formuły matematyczne są codziennością. Dominuje on również niszę systemów składu do prac dyplomowych na tych kierunkach, gdzie używa się wzorów matematycznych.

Podstawowy system TEX składa się z około 300 komend, definiujących zarazem aspekty prezentacyjne, czyli na przykład szerokości odstępów, jak i elementy języka programowania, takie jak instrukcje warunkowe czy funkcje, wspierające wywołania rekurencyjne. Za pomocą tych komend bazowych niewiele dałoby się zrobić, więc powstały dodatkowe formaty — zbiory makr, funkcji i definicji ułatwiających pisanie tekstów. Najbardziej podstawowym formatem jest PlainTEX stworzony przez samego Knutha, który dodaje około 600 nowych definicji. Innymi formatami nadbudowującymi nad podstawowymi komendami TEXa są LATEX (czytamy „Latech") stworzony przez Lesliego Lamporta w 1984 roku oraz ConTEXt (czytamy „Kontecht") stworzony w 1991 roku przez firmę Pragma ADE. Niezależnie od tego, czy korzystamy z PlainTEXa, LATEXa czy ConTEXa, używamy pod spodem TEXa opracowanego przez profesora Knutha.

O samym TEXu i formatach możemy myśleć jak o idei, która może mieć różne implementacje. Do zaimplementowania TEXa począwszy od drugiej wersji Knuth używał

[1] Knuth, Donald E.; Plass, Michael F. (1981), „Breaking paragraphs into lines", Software: Practice and Experience, 11 (11): 1119–1184.

[2] Oprócz TEXa Scribus jest jedynym darmowym i popularnym rozwiązaniem do profesjonalnego składu tekstów, zainteresowanym polecam go do składu plakatów, ulotek lub książek fotograficznych — w ogólności materiałów, w których rozmieszczenie elementów graficznych gra istotną rolę.

[3] Bogusław Jackowski, „Appendix G illuminated" TUGboat. 27 (1), s. 83–90, 2006.

[4] Ulrik Vieth, „Understanding the æsthetic of math typesetting", Proceedings of the BachoTEX 2008 conference, `https://www.youtube.com/watch?v=EwCGOr1tj5Q`.

języka WEB, który opracował na tę właśnie okazję. WEB był pierwszym na świecie językiem wspierającym programowanie piśmienne (ang. literate programming), w którym mamy jeden kod źródłowy będący jednocześnie dokumentacją i kodem programu. Program ten jest obecnie nierozwijany, natomiast są inne silniki TEXa, które implementują język TEX i pozwalają na generowanie dokumentów w różnych formatach. Do najpopularniejszych należą PDFTEX, XeTEX oraz LuaTEX. Wszystkie trzy pozwalają na kompilacje dokumentów napisanych w języku TEX w którymś z jego formatów (PlainTEX, LATEX, ConTEXt) do pliku z rozszerzeniem PDF.

Jak widać, programów do TEXa jest dużo, a gdy pomyślimy o tysiącach bibliotek do różnych formatów, rozbudowujących ich funkcjonalność, łatwo się domyślić, że nie sposób instalować wszystkich elementów systemu składu TEX ręcznie. Aby poradzić sobie z instalacją w sposób prostszy, powstały dystrybucje TEXa. Tutaj do najpopularniejszych należą TEXLive (dostępny na wszystkie główne systemy operacyjne), MacTEX (dostępny na macOS), MikTEX i proTEX (dostępne na Windows). Istnieje również możliwość korzystania z TEXa przez przeglądarkę na różnych stronach, między innymi `https://papeeria.com/` lub `https://www.overleaf.com/`. Na zajęcia jednak nie zalecamy korzystania z wersji online, ponieważ zaletą instalacji TEXa na własnym komputerze jest to, że mamy dostęp do większej gamy pakietów, możemy pracować z bardziej rozbudowanymi narzędziami, współpracować z systemami kontroli wersji w ten sam sposób co dla innych plików i pracować bez połączenia z siecią.

Same silniki TEXa służą do konwertowania pliku z rozszerzeniem `.tex` do `.pdf`, zatem oprócz dystrybucji TEXa potrzebujemy też odpowiedniego edytora i innych narzędzi. Znów jest wiele opcji, w których możemy wybierać. Dobrą integrację z narzędziami do sprawdzania pisowni, co jest nieocenione przy pracy nad tekstem, ma edytor TEXStudio oraz dodatek LATEXWorkshop do Visual Studio Code.

Kwestia wyboru edytora, dystrybucji, silnika i formatu jest w dużej mierze kwestią gustu, jednak na potrzeby zajęć ustalimy środowisko tak, aby wszyscy korzystali z takiego samego zestawu narzędzi. Ułatwi to proces nauki. Na innych kursach zachęcam, aby popatrzeć na inne narzędzia i wybrać takie, które są najbardziej dostosowane do potrzeb. My będziemy korzystali z:

1. formatu LATEX,
2. silnika PDFTEX (często nazywany PDFLATEX, gdy łączymy go z formatem LATEX),
3. dystrybucji TEXLive,
4. edytora TEXStudio.

W edytorze TEXStudio pisać będziemy kod wykorzystujący format LATEXa w pliku `.tex`, po czym, gdy klikniemy przycisk odpowiedzialny za kompilację dokumentu, silnik PDFLATEX wygeneruje nam plik `.pdf`, który edytor TEXStudio wyświetli w podglądzie obok kodu źródłowego. Proces ten już nieco wcześniej nazwaliśmy kompilacją i tej nazwy będziemy się trzymać. Plik `.tex` jest kodem źródłowym, a plik `.pdf` dokumentem wynikowym. Niekiedy dość swobodnie będziemy mówić, że pracujemy w TEXu lub LATEXu, mając na myśli cały zestaw narzędzi wokół tych technologii.

2 Pierwszy dokument w LaTeXu

Dokument składa się z dwóch elementów — preambuły i ciała dokumentu. Preambuła zaczyna się od jego początku i trwa aż do wystąpienia `\begin{document}`. Ciało dokumentu zaczyna się zaraz za `\begin{document}` i trwa aż do `\end{document}`, co powinno jednocześnie kończyć plik. Najczęściej pierwszą linią preambuły jest załadowanie tak zwanej klasy dokumentu, zawierającej ogólne informacje o charakterze pisanego tekstu — inaczej składamy angielską książkę, inaczej polski artykuł — do tego drugiego użyjemy na przykład `\documentclass{mwart}`.

Już na tym etapie mamy pierwsze spotkanie z komendami LaTeXa. Na razie poznamy jedynie pewne przybliżenie tego, jak one działają. `\documentclass` jest typową komendą. Zaczyna się od symbolu \ (backslash) i trwa do końca słowa. Potem przekazujemy jej argument. Aby przekazać kilka liter jako argument, łączymy je w tak zwaną grupę. Do tworzenia grup służą nawiasy klamrowe. W `\documentclass{mwart}`, słowo `mwart` jest pierwszym argumentem komendy `\documentclass`. Komendy `\begin` oraz `\end` służą do określania tak zwanych środowisk. Środowisko służy do otaczania innych komend tak, aby nadać im pewne wspólne cechy. Na przykład poznamy środowisko do tworzenia wypunktowań, środowisko do umieszczania ilustracji, środowisko do wprowadzania wzorów, i wiele więcej przydatnych środowisk. Każde otwarte środowisko (za pomocą `\begin`), musi być zamknięte (za pomocą `\end`).

Znaczenie środowiska `document` jest wyjątkowe, ponieważ powinno występować tylko raz w całym dokumencie i otacza cały tekst, który trafia do rzeczywistego pliku PDF. Wszystko, co jest poza tym środowiskiem, nie trafi do pliku PDF, choć pozwala na przykład na skonfigurowanie odpowiednich fontów, rozmiarów strony i nadania innych cech całemu dokumentowi.

W przykładzie jako klasę dokumentu podana została `mwart`. Jest to polski odpowiednik klasy `article`, zgodny z polską tradycją typograficzną. Klasy te służą do składu krótkich tekstów, bez podziału na rozdziały. Najczęściej ich długość to kilka, maksymalnie kilkanaście stron. Drugą pod względem długości klasą są raporty: `mwrep` i `report` odpowiednio dla języka polskiego i angielskiego. Raporty są nieco dłuższe, mają osobną stronę tytułową i pozwalają na podział dokumentu na rozdziały. Zwykle raporty mają od kilkunastu do kilkudziesięciu stron. Raporty najczęściej składane są w ten sposób, aby były drukowane jednostronnie. Kolejnym rodzajem dokumentów są książki: `mwbk` i `book` odpowiednio dla języka polskiego i angielskiego. Książki zwykle mają od kilkudziesięciu do kilkuset stron i oprócz rozdziałów pozwalają na podział na części. Książki najczęściej składane są tak, aby były drukowane dwustronnie, zatem nowe rozdziały rozpoczynają zawsze po prawej stronie wydruku. Z punktu widzenia zastosowania na studiach, książka jest odpowiednią klasą dla pracy dyplomowej, raport dla sprawozdania, a artykuł dla opisu wystąpienia na konferencji.

Oprócz klasy dokumentów zwykle ładujemy dodatkowe pakiety. Służy do tego komenda `\usepackage`. Pakiety do zbiory makr dostarczające dodatkową funkcjonalność. I tak na przykład są standardowe pakiety, które ładujemy, gdy składamy tekst w języku polskim. Są pakiety, które ładujemy, gdy składamy wzory matematyczne. Są też takie, które ładujemy, gdy załączamy ilustracje lub korzystamy z różnych kolorów. Pakiety będziemy poznawać

kolejno, gdy będą potrzebne. Tutaj zaczniemy od pakietu do języka polskiego. Zdefiniujmy pierwszy dokument. Zapiszmy go w pliku tekstowym w kodowaniu UTF-8.

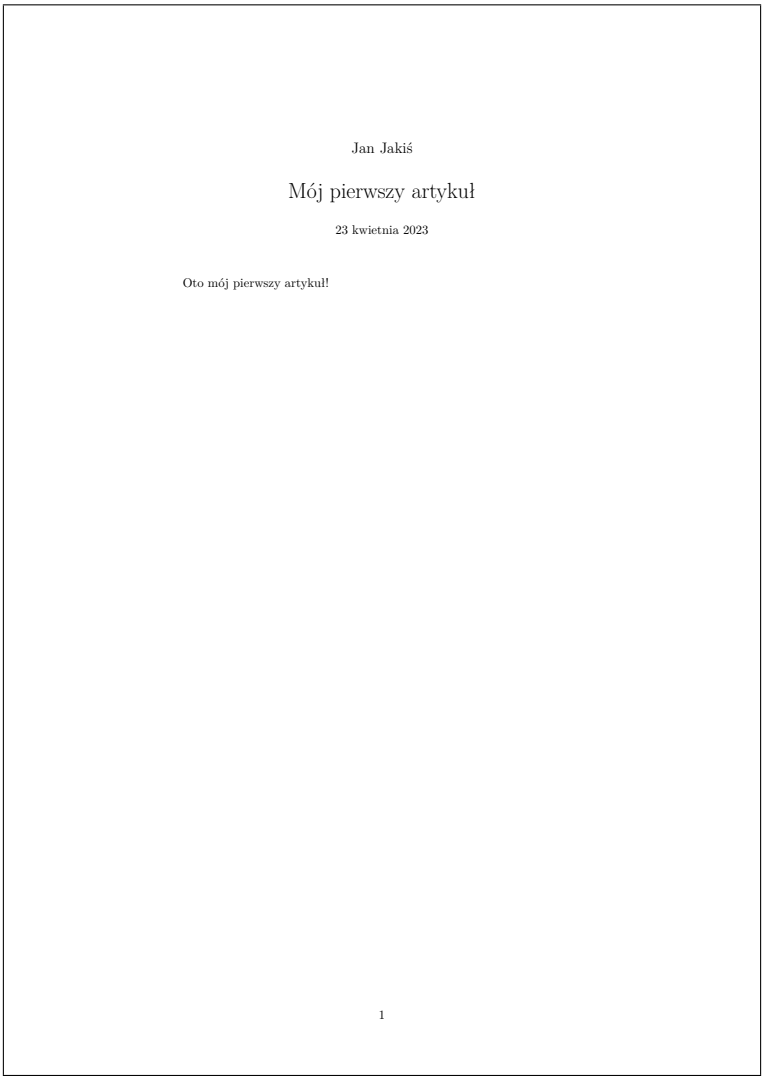
```
\documentclass{mwart}
\usepackage[utf8]{inputenc}
\usepackage{polski}
\usepackage{lmodern,microtype}

% dane o autorze
\author{Jan Jakiś}
\title{Mój pierwszy artykuł}
\date{\today}

\begin{document}
    \maketitle
    Oto mój pierwszy artykuł!
\end{document}
```

W dokumencie tym pojawiło się kilka nowości. Po pierwsze, niektóre komendy mają argumenty opcjonalne. Taką komendą jest `\usepackage`. Argumenty podajemy w nawiasach kwadratowych. Pakietu `inputenc` ładowany w drugiej linii, za pomocą argumentu opcjonalnego `utf8` podaje TEXowi, w jakim kodowaniu zapisany został plik. Wartość ta musi zgadzać się z rzeczywistym kodowaniem pliku, ponieważ bez tego nie będą działać znaki diakrytyczne. Kolejnym nowym elementem jest to, że komenda `usepackage` może ładować kilka pakietów na raz. W linii czwartej ładujemy pakiet `lmodern` definiujący odświeżony font o nazwie Latin Modern (w zamian za standardowy Computer Modern) oraz pakiet `microtype`, włączający automatyczną regulację cech mikrotypograficznych. Pakiet `polski` definiuje między innymi komendy do dywizu oraz pauzy, tłumaczenia standardowych nazw takich jak „rozdział” oraz polskie symbole matematyczne nierówności.

Po załadowaniu pakietów w kodzie źródłowym umieszczony jest komentarz. Komentarze w TEXu oznaczamy symbolem procent. Następnie występują trzy komendy `author`, `title` i `date` ustalające odpowiednio autora, tytuł i datę powstania dokumentu. Data sama w sobie jest wpisana za pomocą komendy `\today`, która nie przyjmuje argumentów i wypisuje datę aktualną na dzień kompilacji dokumentu. Komendy te tworzą tak zwane metadane.

Wewnątrz dokumentu umieszczamy tytuł, który zostanie złożony zgodnie z zasadami składu w języku polskim (komenda `\maketitle`) oraz fragment treści dokumentu. Gdy piszemy treść w LATEXu, wszystkie ciągi białych znaków traktowane są jako spacja, z wyjątkiem dwóch lub więcej wolnych linii występujących po sobie. Taki ciąg wolnych linii oznacza nowy akapit. Tutaj tekst składa się z jednego akapitu.

Dokument po skompilowaniu wygląda następująco.



Jeśli chcemy, aby były również ustawione w dokumencie PDF, możemy użyć dodatkowo pakietu

```
\usepackage[hidelinks,breaklinks,pdfusetitle,pdfdisplaydoctitle]{hyperref}
```

Ładujemy pakiet i przekazujemy mu cztery opcje:

1. `hidelinks` spowoduje, że linki w dokumencie nie będą otoczone prostokątami,
2. `breaklinks` spowoduje, że linki będą dzielone na linie,
3. `pdfusetitle` spowoduje, że tytuł i autor znajdą się we właściwościach pliku PDF, nie tylko w jego treści,
4. `pdfdisplaydoctitle` spowoduje, że tytuł trafi na pasek przeglądarki dokumentów zamiast nazwy pliku.

Dodatkowo komenda `hyperref` powoduje, że wszystkie referencje w dokumencie można klikać, aby się po nim poruszać, że do pliku PDF dołączony zostanie interaktywny spis treści oraz uzyskamy możliwość dodawania klikalnych linków do stron internetowych za pomocą komendy `\url`, na przykład `\url{https://www.google.com/}`.

3 Formatowanie tekstu

Ponieważ niektóre znaki są zarezerwowane, na przykład backslash, aby je wypisać, musimy posłużyć się odpowiednimi komendami. Do znaków $, &, %, #, _, {, }, \ i ~, użyjemy `\$`, `\&`, `\%`, `\#`, `\_`, `\{`, `\}`, `\textbackslash` i `\textasciitilde` odpowiednio. Nazwy TEX i LATEX zapisujemy jako `\TeX` i `\LaTeX`.

Spacje zapiszemy za pomocą symbolu spacji, `\ ` (backslash i spacja) lub ~. W pierwszym przypadku jest to zwykła spacja, w drugim taka, której nie da się zignorować a w trzecim niełamliwa spacja. Środkowy symbol jest przydatny, gdy piszemy komendy takie, jak `\TeX`. Spacja bezpośrednio po komendzie jest przez TEXa ignorowana, aby można było zapisać słowa takie jak `\TeX a` — bez spacji. Gdyby połączyć te słowa, stałoby się to komendą `\TeXa`, która nie jest zdefiniowana. Ponieważ spacje są ignorowane po komendach bez argumentów, powinniśmy w takim miejscu wstawić spację nieignorowalną, jeśli zależy nam na pojawieniu się spacji po komendzie. Na przykład

```
\LaTeX\ jest systemem składu
```

prawidłowo wstawi nieignorowaną spację po komendzie `\LaTeX`, podczas gdy

```
\LaTeX jest systemem składu
```

sklei dwa pierwsze słowa. Spacji niełamliwej (uzyskiwanej symbolem tyldy) używamy, gdy nie chcemy, aby w pewnym miejscu linia została podzielona. Pozwala nam to uniknąć błędu typograficznego zwanego sierotą, na przykład `było to w~2010~roku`. Zwróćmy uwagę, że również pomiędzy 2010 a „roku” wstawiono niełamliwą spację, aby nie oddzielać liczby od słowa rok.

Do zapisu wielokropka użyjemy `\ldots`. Polskie cudzysłowy otwierające i zamykające uzyskamy za pomocą symboli `,,` oraz `''` (dwa znaki na każdy), wewnętrzne za pomocą `<<` i `>>`. Na przykład

```
,,Witaj w <<moim>> świecie''.
```

da w wyniku „Witaj w «moim» świecie”. Słowa z dywizem zapisujemy z komendą `\dywiz` jak w połączeniu

```
niebiesko\dywiz złoty
```

natomiast symbole półpauzy i pauzy uzyskamy za pomocą komend \ppauza i \pauza. Cudzysłowy angielskie uzyskamy pisząc ‘ i ’ oraz ‘‘ i ’’ dla wewnętrznych. Dywiz zapiszemy za pomocą znaku -, półpauzę za pomocą -- oraz pauzę za pomocą ---. Stosowanie znaków typograficznych odpowiednich dla języka jest bardzo ważne.

Tekst wyróżniamy za pomocą komendy \emph, która składa go italikiem. Inny rodzaj wyróżnienia, często używany do skrótów i czasem do nazw własnych, to \textsc. Na przykład możemy napisać

```
\textsc{Windows} to \emph{system operacyjny}.
```

Uzyskując napis Windows to *system operacyjny*.

Istnieje więcej komend regulujących wygląd kroju pisma. Mamy na przykład \textup, \textit, \textsl, \textmd, \textbf, \textrm, \textsf, \texttt. Zachęcam do sprawdzenia i przetestowania różnych kombinacji tych słów kluczowych.

4 Podział dokumentu

Logiczny podział dokumentu uzyskujemy za pomocą odpowiednich komend.

Komenda	Podział
\part{tytuł}	część, tylko w książkach
\chapter{tytuł}	rozdział, tylko w książkach i raportach
\sectiton{tytuł}	podrozdział
\subsection{tytuł}	sekcja (podrozdział)
\subsubsection{tytuł}	podsekcja
\paragraph{tytuł}	paragraf (zwykle w dokumentach prawnych)
\subparagraph{tytuł}	podparagraf (zwykle w dokumentach prawnych)

Komendy te umieszczamy przed pierwszym akapitem danego ustępu. Zaraz po nich często występuje komenda \label. Komenda \label nadaje etykietę, na którą możemy się potem powołać. Z definicji etykieta dotyczy pierwszej rzeczy w dokumencie na lewo od jej miejsca wystąpienia, do której da się odwołać. Napiszemy na przykład

```
\section{Podział dokumentu}\label{sekcje}
```

Potem, gdy chcemy się odwołać do numeru podrozdziału, napiszemy \ref{sekcje} uzyskując 4. Możemy się też odwołać do strony, pisząc \pageref{sekcje}, na przykład

```
Patrz podrozdział \ref{sekcje} na stronie \pageref{sekcje}.
```

da w wyniku „Patrz podrozdział 4 na stronie 8.”

Inne przydatne komendy wstawiające pewien tekst do dokumentu to \maketitle, który mieliśmy już okazję zobaczyć (wstawia stronę tytuł lub tytułową zależnie od klasy dokumentu) i \tableofcontents, \listoffigures oraz \listoftables umieszczające kolejno spis treści, spis rysunków oraz spis tabel. W książkach mamy też \frontmatter, \mainmatter, \appendix i \backmatter, które przełączają tryb działania niektórych komend.

1. od `\frontmatter` do `\mainmatter` rozdziały nie są numerowane (np. „Wstęp”), a strony są numerowane cyframi rzymskimi od `i` w dal;
2. od `\mainmatter` do `\appendix` rozdziały są numerowane liczbami (np. „Rozdział 1”), a strony są numerowane cyframi arabskimi, ponownie od `1` w dal;
3. od `\appedix` do `\backmatter` rozdziały są numerowane literami (np. „Dodatek A”), a strony są numerowane cyframi arabskimi, kontynuując z wcześniejszego segmentu;
4. od `\backmatter` do końca dokumentu rozdziały nie są numerowane (np. „Bibliografia”), a strony są numerowane cyframi arabskimi, kontynuując z wcześniejszego segmentu.

Komendy te pozwalają na dostosowanie spisu treści i sposobu numeracji rozdziałów do naszych potrzeb.

5 Podstawowe środowiska

Podstawowe środowiska do formatowania tekstu to `quote` do składu cytatów krótkich, `quotation` do cytatów długich i `verse` do wierszy. Jak każde środowisko, którego przykład widzieliśmy przy środowisku `document`, musi zaczynać się za pomocą `\begin` i kończyć za pomocą `\end`. Aby złożyć jeden z cytatów prof. Knutha, napiszemy

```
\begin{quote}
 ,,Premature optimization is the root of all evil''.
\end{quote}
```

uzyskując:

> „Premature optimization is the root of all evil”.

Niekiedy zależy nam na wyśrodkowaniu tekstu, wtedy użyjemy środowiska `center`. Do składu tekstu w chorągiewkę, użyjemy środowisk `flushleft` lub `flushright`, odpowiednio wyrównującego tekst do lewego lub prawego marginesu. Środowisk tych powinniśmy używać sporadycznie, na przykład przy rozmieszczaniu ilustracji lub tabel.

Do tworzenia wypunktowań używamy trzech środowisk: `itemize` do wypunktowań, `enumerate` do wyliczeń oraz `description` do opisów. Spójrzmy na różnicę, pomiędzy trzeba wariantami tych środowisk.

```
\begin{itemize}
    \item Raz,
    \item dwa,
    \item trzy.
\end{itemize}
```

— Raz,

— dwa,

— trzy.

```
\begin{enumerate}
    \item Raz,
    \item dwa,
    \item trzy.
\end{enumerate}
```

1. Raz,
2. dwa,
3. trzy.

```
\begin{description}
    \item[czarny] black,
    \item[biały] white,
    \item[niebieski] blue.
\end{description}
```

czarny black,

biały white,

niebieski blue.