Kurs komputerowy T—LaTeX

Grzegorz Kowzan

master, 23 kwietnia 2017

Spis treści

I. Wprowadzenie

I.1. Czym jest LaTeX

W skrócie:

- LaTeX to język znaczników używany razem z programem do składu tekstu TeX do przygotowywania różnego rodzaju dokumentów, począwszy od artykułów naukowych i raportów, a skończywszy na książkach.
- LaTeX podobnie jak TeX jest oprogramowaniem o otwartym źródle. Podstawowa część LaTeX-a jest rozwijana przez grupę głównych programistów LaTeX-a, lecz wiele dodatkowych funkcji powstało dzięki możliwości modyfikacji działania tegoż przez każdego użytkownika.
- Dokumenty LaTeX-a składają się z jednego lub więcej plików źródłowych, które są zwykłymi plikami tekstowymi, zawierającymi komendy znacznikowe oraz właściwy tekst.
- Na podstawie dokumentów LaTeX-a program TeX generuje dokumenty PDF, HTML, XML, PostScript.

I.2. Języki znacznikowe

Obecnie najpowszechniejszym sposobem przygotowywania dokumentów przy użyciu komputera jest składanie ich przy pomocy edytorów WYSIWYG (ang. *what you see is what you get*). Mimo prostoty ich użycia przy składaniu prostych dokumentów można wymienić ich istotne wady:

- Niespodziewane zmiany w formacie plików lub oprogramowaniu używanym do ich wyświetlania może prowadzić do różnego wyglądu w zależności od zestawu czcionek zainstalowanych na danym komputerze, wersji oprogramowania albo wersji formatu pliku i dowolnych kombinacji tychże.
- Drobne zmiany w formatowaniu dokumentu mogą powodować drastyczne zmiany rozmieszczenia rysunków w dokumencie, o którym decyduje nie opisany nigdzie algorytm.
- Nieestetyczny wygląd wzorów matematycznych w dokumentach naukowych.
- Znacznie utrudnione oddzielenie logicznej struktury dokumentu od wyglądu, utrudniające zapewnienie jednolitego wyglądu całego dokumentu. Bardzo łatwo zmienić i dopasować wygląd pojedynczych elementów tekstu, nie zapewniając spójnego wyglądu całego dokumentu.

Alternatywnym podejściem są języki znacznikowe, w których surowy tekst zawiera komendy opisujące jak powinny wyglądać poszczególne elementy dokumentu. Przykładowo, aby otrzymać poniższy fragment tekstu

> Ten kawałek tekstu jest **pogrubiony**.

przy użyciu HTML-a, zapiszemy go następująco

```
Ten kawałek tekstu jest <b>pogrubiony</b>.
```

Natomiast w TeX-u zostałby on zapisany jako:

```
Ten kawałek tekstu jest {\bf pogrubiony}.
```

Powyżej został zaprezentowany przykład znaczników typograficznych, to znaczy takich które mówią jak ma być zmieniony wygląd fragmentu tekstu w stosunku do reszty dokumentu, tj. żeby zmienić czcionkę na pogrubioną. Alternatywą jest stosowanie znaczników logicznych, które wskazują na znaczenie danego fragmentu tekstu w dokumencie. W HTML-u jak i W LaTeX-u przykładem takich znaczników są znaczniki nagłówkowe:

```
<h1>To jest nagłówek</h1>
```

```
\section{To jest nagłówek}
```

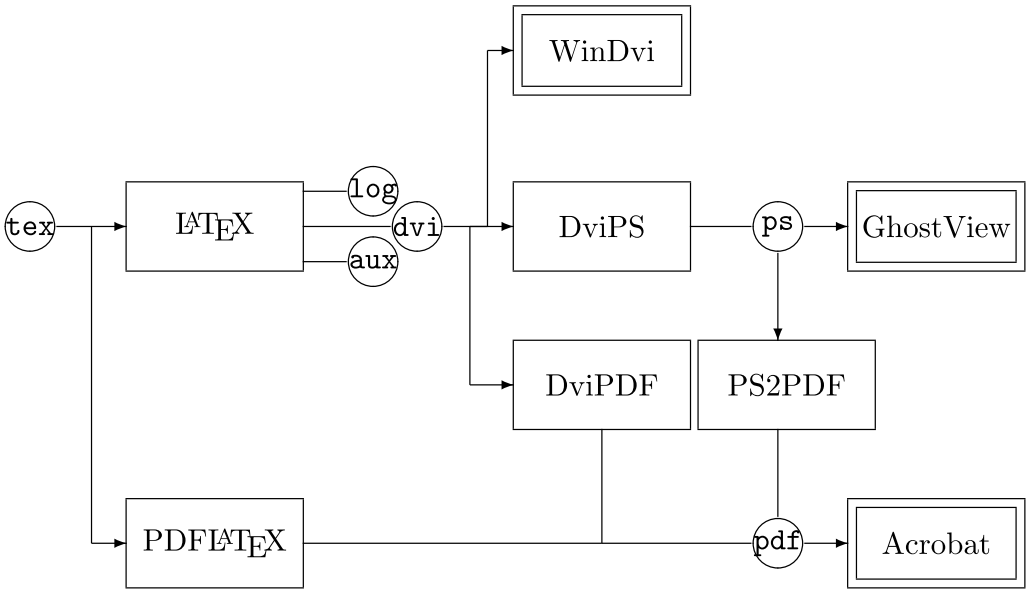
W HTML-u wygląd dokumentu sformatowanego znacznikami logicznymi może być zmieniony poprzez style CSS; z jednego dokumentu źródłowego możemy otrzymać różnorako sformatowane dokumenty końcowe nie zmieniając w żaden sposób dokumentu źródłowego. Podobnie w LaTeX-u, logicznie sformatowany plik źródłowy może pozostać nienaruszony za wyjątkiem wyboru klasy której używamy i odpowiednich komend w preambule dokumentu i mimo to być sformatowany na różne sposoby w zależności od potrzeb.

II. Podstawy składu tekstu w LaTeX-u

Każdy dokument LaTeX-owy składa się z *preambuły* i *ciała dokumentu*.

Preambuła zawiera komendy globalnie określające wygląd dokumentu, takie jak rozmiar strony, rozmiar obszaru na stronie zawierającego tekst, nagłówki stopki, rozmiar marginesów w zależności od parzystości strony. Minimalna preambuła zawiera komendę `\documentclass` określająca najogólniej jak dokument zostanie sformatowany. Domyślny format jest dostosowany do standardów amerykańskich, w związku z czym do każdego dokumentu będzie dodawać komendy do preambuły dopasowujące zachowanie LaTeX-a do polskich norm.

Ciało dokumentu zawiera się między komendami `\begin{document}` oraz `\end{document}`. Ciało może zawierać zwykły tekst, który będzie sformatowany zgodnie z wytycznymi wynikającymi z użytej klasy i komend w preambule, oraz komend o lokalnym działaniu, które np. powodują zmianę kroju czcionki, wstawienie odstępu, rysunku, tabeli, itd.



Rysunek 1: Schemat przetwarzania plików źródłowych LaTeX-a.

Ogólny szablon dokumentu wygląda następująco:

```
\documentclass[opcje]{klasa}
Komendy globalne
\begin{document}
Tekst dokumenty i komendy o lokalnym działaniu
\end{document}
```

Dokument o takiej budowie może być następnie przetworzony przez LaTeX-a i TeX-a do pliku końcowego. Rysunek 1 prezentuje schemat przetwarzania plików źródłowych. TeX odpowiada za część typograficzną procesu, tj. obsługę czcionek, umiejscowienie każdego ze znaków i elementów graficznych na stronie, formatowanie linijka po linijce tekstu, tak aby zajmować wskazaną część strony, itd. LaTeX natomiast tłumaczy zapisane źródła, które zawierają komendy opisujące logiczną strukturę tekstu, bez wchodzenia w szczegóły typograficzne, na komendy TeX-a. Odpowiada on również za tworzenie plików pośrednich umożliwiających automatyczną numerację odnośników do innych partii tekstu, spisów treści, rysunków, tabel, tworzenie bibliografii i skorowidzów.

II.1. Nazwy komend

Nazwy komend LaTeX-a dzielą się na trzy kategorie:

- pojedyncze znaki specjalne: `# $ & ~ _ ^ % { }`,
- ukośnik `\` i następujący zaraz po nim jeden z powyższych znaków specjalnych,
- ukośnik i następujący po nim ciąg liter zakończony znakiem, który nie jest literą; np. `\large` zwiększa rozmiar czcionki. Uwaga: wielkość liter ma znaczenie, `\large`, `\Large` i `\LARGE` to trzy różne komendy.

Komendy mogą przyjmować argumenty wymagane oraz argumenty opcjonalne. Argumenty wymagane otoczone są nawiasami klamrowymi, np. `\emph{tekst}` formatuje tekst między nawiasami klamrowymi czcionką wyróżnienia (domyślnie pochyłą). Komenda może przyjmować wiele argumentów, przykładowo `\rule[1mm]{10mm}{3mm}` wstawia pionową linię 1 mm nad linię bazową bieżącego wiersza, o szerokości 10 mm i wysokości 3 mm.

II.2. Środowiska

Środowiska są inicjowane komendą `\begin{srodowisko}` i zakańczane komendą `\end{srodowisko}`. Tak jak inne komendy, `begin` może przyjmować argumenty opcjonalne, ale `end` ich nigdy nie przyjmuje. Środowiska zmieniają to jak tekst ograniczony przez nie jest interpretowany przez LaTeX-a, mogą np. zmienić krój czcionki, zmienić wcięcie akapitu, umożliwić złożenie tabeli lub umieszczenie rysunku w tekście.

II.3. Deklaracje

Deklaracjami nazywamy komendy, które nie powodują wyświetlenia określonego elementu na stronie, ale zmieniają wartości lub znaczenie parametrów decydujących o formatowaniu tekstu. Zmiana powodowana przez deklarację ma efekt od miejsca w źródle, w którym została wprowadzona do końca dokumentu lub innej deklaracji tego samego typu. Z dwoma istotnymi wyjątkami: znajduje się w środowisku, wtedy jej efekt kończy się ze środowiskiem; jest otoczona parą nawiasów klamrowych `{...}`, wtedy jej efekt kończy się z nawiasem zamykającym.

Przykładowo, aby pogrubić fragment tekstu możemy użyć deklaracji

```
{\bfseries To jest fragment pogrubiony}
```

To jest fragment pogrubiony

albo

```
\setlength{\parindent}{0.5cm}
```

które zmienia wcięcie każdego nowego akapitu na pół centymetra.

II.4. Długości

Długości w LaTeX-u składają się z liczby dziesiętnej i następującej jednostki (por. przykład w poprzedniej części). Błędem jest podanie samej liczby, nawet jeśli jest to zero. Dopuszczalne jednostki to:

- `cm` centymetr,
- `mm` milimetr,
- `in` cal (1 cal = 2,54 cm),

- `pt` punkt (1 cal = 72,27 pt),
- `pc` pika (1 pc = 12 pt),
- `dd` didot (1157 dd = 1238 pt),
- `cc` cycero (1 cc = 12 dd),
- `em` wielkość zależna od aktualnej czcionki, równa szerokości wielkiej litery M,
- `ex` wielkość zależna od aktualnej czcionki, równa wysokości małej litery x.

Istnieją również wielkości elastyczne, których faktyczną wartość LaTeX określa w zależności od sytuacji. Określa się je następująco:

```
wartość nominalna plusrozciągnięcie minusskurczenie
```

przykładowo:

```
\setlength{\parskip}{1em plus5pt minus0.3ex}
```

II.5. Spacje

W przeciwieństwie do domyślnych ustawień najpopularniejszych edytorów WYSIWYG, LaTeX składa tekst tak aby każda linijka miała taką samą szerokość (tekst jest wyjustowany). To oznacza że znakom spacji w kodzie źródłowym nie może być z góry przypisana określona szerokość, taka sama dla każdego znaku spacji, tylko muszą one być traktowane jako znaki specjalne o elastycznej szerokości i ich interpretacja podlega pewnym regułom:

- wiele spacji obok siebie jest traktowanych tak samo jak jedna spacja,
- spacje na początku linii są ignorowane,
- spacje kończące nazwę komendy są ignorowane,
- pojedynczy znak nowej linii jest traktowany jak spacja.

Można wymusić dodatkową spację, jednak prawie nigdy nie ma takiej potrzeby, wstawiając ukośnik przed spację: `\␣`. Można również wymusić spację i *zakazać łamania linii* w danym miejscu znakiem tyldy ~. (Wg reguł pisowni nie powinno się np. łamać linii między inicjałami imion.) Znak spacji jest wstawiany, gdy LaTeX spotyka znak nowej linii, ale można temu zapobiec kończąc linię znakiem `%` (znak komentarza):

```
Wlazł kotek%
na płotek
```

Wlazł kotekna płotek

Nowe akapity są kolei z kolei rozpoczynane, jeśli dwa fragmenty tekstu są oddzielone pustą linia, tj. dwoma znakami nowej linii bezpośrednio obok siebie. Podobnie jak ze spacjami wstawienie większego odstępu w źródle nie zmieni formatowania.

II.6. Cudzysłów

Zgodnie z polską pisownią znak rozpoczynający cudzysłów to dwie kreski na dole linii („), a znak kończący cudzysłów to dwie kreski na górze linii (”). W edytorach WYSIWYG aby wstawić otwierający cudzysłów wpisujemy znak podwójnego cudzysłowu (") i potem po ponownym wstawieniu tego samego znaku, jeśli autokorekta zauważy zamykający znak to automatycznie poprawi znak otwierający na poprawny. W LaTeX-u aby otrzymać poprawnie wyglądający znak

otwarcia cudzysłowu używamy dwóch przecinków (`,,`) i aby otrzymać znak zamknięcia—dwóch pojedynczych cudzysłowów (`''`).

II.7. Odstępy

Odstępy poziome można wstawiać komendami

```
\hspace{odstęp}
\hspace*{odstęp}
```

gdzie jedna i druga komenda wstawia odstęp poziomy określony zgodnie z II.4. Jeśli zostanie użyta wersja bez gwiazdki, to odstęp który miałby się pojawić na początku linii jest ignorowany. Spacje otaczające komendę `\hspace` nie są ignorowane. Komenda `\hfill`, skrót od `\hspace{\fill}`, wstawia nieskończenie elastyczny odstęp poziomy. Przykładowo:

```
Lewy\hfill{}Środek\hfill{}Prawy
```

Lewy Środek Prawy

Analogicznie do `\hfill` działają komendy `\dotfill` oraz `\hrulefill`:

```
1. Wstęp \dotfill 1\\
2. Metodologia \hrulefill 6
```

1. Wstęp 1
2. Metodologia _____ 6

II.8. Łamanie linii

Wymusić łamanie linii możemy komendą `\\`. W ogólności przyjmuje ona argument opcjonalny

```
\\[odstep]
\\*[odstep]
```

która oznacza o ile do dołu ma być przesunięta następna linia, przy czym jeśli następna linia znajdowałaby się na nowej stronie, to w przypadku wariantu bez gwiazdki ten odstęp jest ignorowany. W przypadku wariantu z gwiazdką, jeśli nowa linia miałaby się znaleźć na następnej stronie, to bieżąca linia też jest przenoszona na następną stronę i wskazany odstęp jest uwzględniany. Komendy łamania linii nie działają (nie dodadzą odstępu), jeśli są umieszczone między akapitami.

Łamanie linii podwójnym ukośnikiem sprawia, że tekst w tej linii nie będzie wyjustowany do prawej strony. Można również mniej lub bardziej silnie zasugerować LaTeX-owi żeby złamał linie i odpowiednio zwiększył odstępy między wyrazami żeby wypełnić linijkę komendą:

```
\linebreak[num]
```

gdzie *num* jest liczbą między 0 a 4. Przykładowo:

```
Ta linia jest trochę za krótka, ale\linebreak chcę ją złamać.\\
Ta linia jest trochę za krótka, ale\\ chcę ją złamać.
```

Ta linia jest trochę za krótka, ale
chcę ją złamać.
Ta linia jest trochę za krótka, ale
chcę ją złamać.

Brak argumentu opcjonalnego `\linebreak` ma takie same działanie jak argument 4, czyli bezwzględnie wymusza łamanie linii. Argumenty między 0 a 3 jedynie zniechęcają LaTeX-a do złamania linii.

II.9. Odstępy pionowe

Odstępy pionowe można wstawiać komendami

```
\vspace{odstęp}
\vspace*{odstęp}
```

Wersja z gwiazdką doda wskazany odstęp nawet jeśli znajduje się na początku strony. Jeśli te komendy są użyte w obrębie akapitu, to wskazany odstęp zostanie dodany dopiero po nowej linii; nie wymusza on łamania linii w miejscu wystąpienia tak jak `\\`. Podobnie jak z odstępami poziomymi, mamy również do dyspozycji komendę `\vfill`.

II.10. Łamanie stron

Odpowiednikiem `\linebreak` dla łamania stron jest komenda `\pagebreak[num]`. Ta komenda, podobnie jak `\vspace`, nie wymusza łamania linii. Interpretacja argumentów jest analogiczna do `\linebreak`. Podobnie do `\linebreak`, jeśli złamanie strony powoduje, że część strony pozostanie pusta, to LaTeX zwiększy interlinie w obrębie akapitów i odstępy pionowe między akapitami, aby tekst zajmował całą stronę.

Z kolei odpowiednikiem `\\` lub `\newline` jest komenda `\newpage`.

Jest jeszcze kilka istotnych komend związanych z łamaniem stron. Jeśli dokument zawiera tabele, rysunki i podobne elementy, to LaTeX stara się je umieścić jak najbliżej miejsca wystąpienia odpowiedniej komendy w kodzie. Czasami jednak nie jest to możliwe, wtedy dany element jest przesuwany na kolejną stronę.[1] Jeśli chcemy, możemy wymusić umieszczenie elementów graficznych komendą `\clearpage`. Powoduje ona natychmiastowe złamanie strony tak jak `\newpage` i w wolne miejsce umieszcza wszystkie zaległe tabele i rysunki.

W dokumentach dwukolumnowych komendy `\newpage`, `\pagebreak` kończą bieżącą kolumnę zamiast całej strony. Natomiast `\clearpage` i `\cleardoublepage` kończą całą stronę, wstawiając pustą kolumnę jeśli trzeba.

Jeśli używany klasy z opcją `twoside`, to mamy do dyspozycji też komendę `\cleardoublepage`, której działanie jest analogiczne do `\clearpage`, lecz dodatkowo dba oto aby tekst po tej komendzie zaczynał się na stronie nieparzystej.

Czasami mimo starań LaTeX-a i naszych łamanie strony wygląda niezbyt estetycznie. Możemy wtedy spróbować pomóc algorytmowi zwiększając odrobinę wysokość bieżącej strony komendami

[1] Tym od czego dokładnie zależy czy taki element zostanie umieszczony zajmiemy się później.

```
\enlargethispage{rozmiar}
\enlargethispage*{rozmiar}
```

gdzie wersja z gwiazdką zmniejsza jeszcze odpowiednio interlinię.

III. Struktura dokumentu

III.1. Klasy dokumentów

Każdy dokument LaTeX-a musi mieć zdefiniowaną klasę, która decyduje o formatowaniu całego dokumentu, rozkładzie elementów na stronie i o tym na jakie jednostki logiczne możemy podzielić dokumenty. Książki mają domyślnie zdefiniowane nagłówki i rozróżniają strony parzyste i nieparzyste. Domyślnie nowy rozdział zaczyna się na stronie nieparzystej (prawej). Tabela 2 zawiera opis opcji, którymi możemy zmodyfikować domyślne zachowanie klas. Każda z klas oferuje domyślnie mechanizm prezentacji informacji tytułowych na początku dokumentu. W preambule dokumentu standardowo używamy komend:

```
\title{tytuł dokumentu}
\author{autor dokumentu}
\date{data}
```

a następnie na samym początku ciała dokumentu umieszczamy komendę `\maketitle`. Alternatywnie możemy pominąć powyższe komendy i samodzielnie sformatować stronę tytułową. Do tego używamy środowiska `titlepage` umieszczonego tuż po `\begin{document}`.

Innym częstym elementem różnego rodzaju artykułów i raportów jest *streszczenie* (abstract). Do wprowadzenia streszczenia używamy środowiska `abstract`, które umieszczamy po stronie tytułowej, ale przed pierwszą komendą sekcjonującą.

Wygląd stopek i nagłówków Domyślnie w LaTeX-u zdefiniowane są *style strony*, decydujące o zawartości i wyglądzie stopek i nagłówków:

`plain` stopka zawiera wyśrodkowany numer strony,
`empty` stopka i nagłówek są puste,
`headers` nagłówek zawiera numer strony i tytuł aktualnego rozdziału lub sekcji.

Style strony można zmienić deklaracją

```
\pagestyle{styl}
\thispagestyle{styl}
```

gdzie druga deklaracja zmienia tylko styl bieżącej strony.

III.2. Ładowanie pakietów

Pakiety ładujemy umieszczając następujące polecenie w preambule:

Tablica 1: Klasy dokumentów domyślnie dostępne w LaTeX-u.

Klasa	Opis
`article`	Krótkie dokumenty (do ok. 20 stron); artykuły naukowe, krótkie raporty, podania, dokumentacja programów.
`report`	Dłuższe raporty, krótkie książki.
`book`	Książki.
`letter`	Listy, posiada ułatwienia w adresowaniu tego samego dokumentu do wielu odbiorców

Tablica 2: Opcje klas domyślnie dostępnych w LaTeX-u.

Opcja	Opis
	wymiar strony:
letterpaper	11 in. × 8.5 in. (domyślny rozmiar)
a4paper	297 mm × 210 mm
a5paper	210 mm × 148 mm
b5paper	250 mm × 176 mm
legalpaper	14 in. × 8.5 in.
executivepaper	10.5 in. × 7.25 in.
10pt, 11pt, 12pt	wielkość czcionki podstawowej dokumentu (10pt jest domyślnie)
final, draft	wersja końcowa lub robocza
onecolumn, twocolumn	dokument jedno- lub dwukolumnowy
oneside, twoside	dokument jedno- lub dwustronny
openany, openright	rozpoczynanie nowego rozdziału na dowolnej lub prawej stronie
notitlepage, titlepage	umieszczaj lub nie informacje tytułowe na osobnej stronie
landscape	druk poziomy
fleqn	wyjustowanie wzorów matematycznych do lewej strony
leqno	numeracja wzorów matematycznych po lewej stronie

```
\usepackage[opcja1,opcja2,...]{pakiet1,pakiet2,...}
```

gdzie każda z opcji jest przekazywana każdemu pakietowi.

III.3. Sekcjonownanie

W ogólności dokument możemy podzielić na części (`\part`), rozdziały (`\chapter`, tylko klasy `book` i `report`), sekcje (`\section`), podsekcje (`\subsection`), podpodsekcje (`\subsubsection`), akapity (`\paragraph`) i podakapity (`\subparagraph`):

```
\komenda_podziału[krótka nazwa]{nazwa}
\komenda_podziału*[krótka nazwa]{nazwa}
```

gdzie wersja z gwiazdką nie dodaje danej pozycji do spisu treści a argument opcjonalny, jeśli jest podany, jest używany w spisie treści zamiast *nazwy*.

Poszczególnym częściom dokumentu, tabelom, rysunkom i wzorom możemy przypisać identyfikatory, aby potem móc się do nich odwoływać w innych częściach dokumentu. Przykładowo, jeśli użyjemy konstrukcji:

```
\section{Wstęp}
\label{sec:wstep}
Treść dokumentu
W rozdziale~\ref{sec:wstep} pisaliśmy o...
```

to LaTeX w miejsce `\ref{sec:wstep}` wstawi numer sekcji *Wstęp*.

Każdej komendzie sekcjonującej jest przypisany pewien *poziom* i *licznik*. Licznik danej komendy sekcjonowania jest inkrementowany za każdym razem gdy użyjemy danej komendy, np. użycie `\section` inkrementuje licznik `section`. Do każdej komendy sekcjonowania przypisany jest licznik

o tej samej nazwie. Możemy też zmienić jego wartość ręcznie, np:

```
\setcounter{section}{2}
```

Poziomy `section`, `subsection`,... wynoszą 1, 2,... dla każdej klasy. Dla `book` i `report` komendzie `\part` odpowiada poziom -1, a `\chapter` odpowiada 0; w klasie `article` komendzie `\part` odpowiada 0. Zmieniając wartość licznika `secnumdepth` zmieniamy maksymalny poziom numerowania komend sekcjonowania. Konsekwencje tego są takie, że powyżej tego poziomu nagłówki nie będą zawierać numerów i nie będą dodawane do spisu treści.

Wygląd nagłówków Najwygodniejszy sposób na wprowadzenie prostych zmian w wyglądzie nagłówków oferuje pakiet `titlesec`. Gdy już zostanie załadowany w preambule, możemy użyć komendy

```
\titleformat*{\komenda_podziału}{komendy_formatowania}
```

przykładowo

```
\titleformat*{\section}{\LARGE\scshape\bfseries}
```

sprawi, że nagłówki sekcji będą sformatowane pogrubionymi kapitalikami o rozmiarze `LARGE`.

III.4. Spis treści

Spis treści generujemy komendą `\tableofcontents`. Możemy go ręcznie modyfikować komendami

```
\addcontentsline{toc}{licznik}{tekst}
\addtocontents{toc}{tekst}
```

gdzie *licznik* jest licznik wybranej komendy sekcjonowania. Pierwsza komenda pozwala umieścić w odpowiednim miejscu hierarchii sekcjonowania element spisu treści. Z kolei druga komenda jest raczej używana do zmiany formatowania spisu treści (np. złamania strony, dodania odstępu), ponieważ umieszcza dowolny podany kod LaTeX-a w kodzie generujący spis treści.

Mamy jeszcze domyślnie zdefiniowanie: spis rysunków (`\listoffigures`, `lof` zamiast `toc`) oraz spis tabel (`\listoftables`, `lot` zamiast `toc`).

III.5. Numerowanie stron

Styl numerowania stron zmieniamy komendą

```
\pagenumbering{styl}
```

gdzie styl to jedno z:

arabic liczby arabskie,
roman liczby rzymskie (małe),
Roman liczby rzymskie (wielkie),
alph litery alfabetu (małe),
Alph litery alfabetu (wielkie).

Możemy ponadto ręcznie zmienić wartość aktualnego numeru strony zmieniając wartość odpowiedniego licznika:

```
\setcounter{page}{numer strony}
```

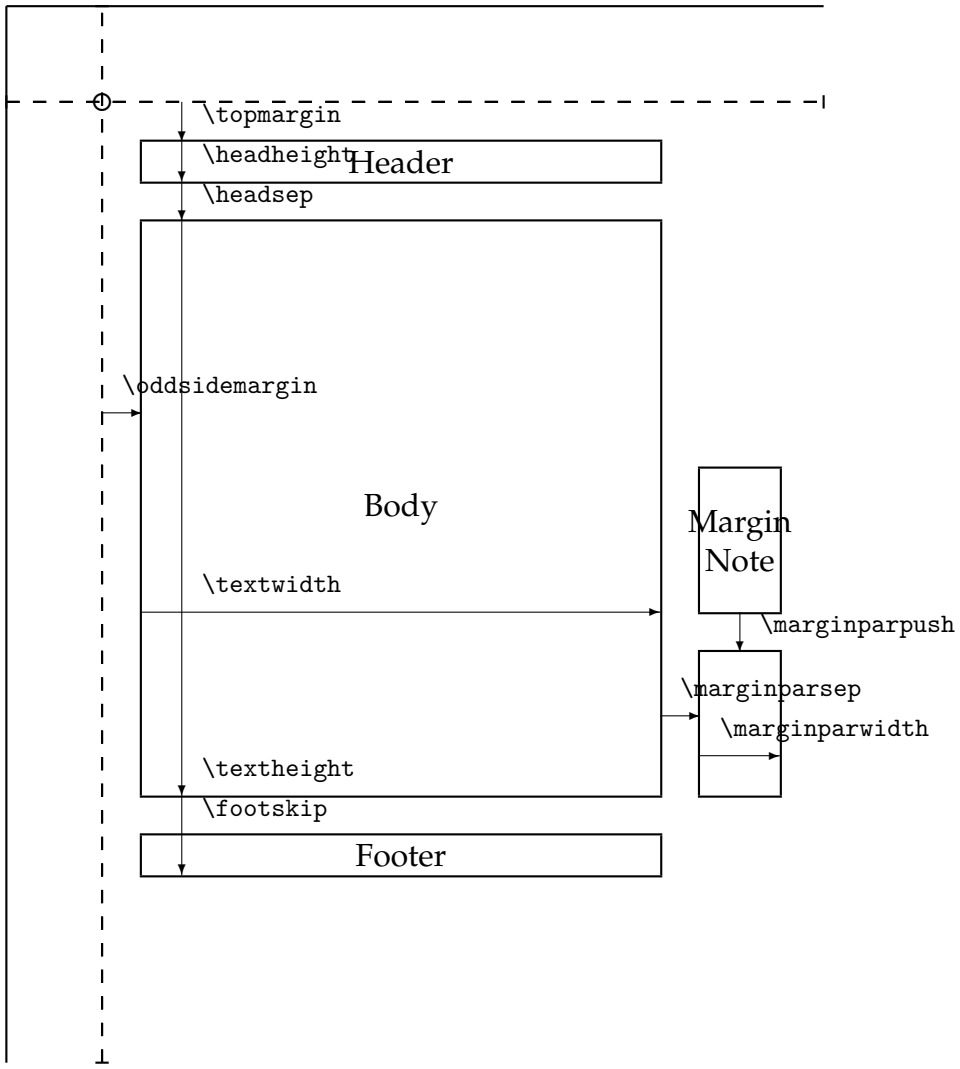
III.6. Modyfikowanie formatowania dokumentu

Ogólny wygląd dokumentu możemy zmienić modyfikując poniższe wielkości:

`\columnsep` odstęp między kolumnami,
`\columnseprule` grubość linii pionowej między kolumnami,
`\parskip` odstęp między akapitami,
`\parindent` wcięcie akapitów,
`\baselinestretch` interlinia, zmieniamy komendą: `\renewcommand{\baselinestretch}{mnożnik}`

W ogólności każdą stronę LaTeX dzieli na części zgodnie ze schematem z Rysunku 2.

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.



Rysunek 2: Schemat strony i wartości określające formatowanie strony w LaTeX-u.

Widzimy, że jest zarezerwowane miejsce na nagłówek, stopkę i notatki na marginesie. Można też zauważyć, że układ współrzędnych strony w LaTeX-u nie zaczyna się od lewego górnego rogu strony (tak by było logicznie), lecz od punktu przesuniętego o 1 cal w dół i w prawo względem rogu strony. Wielkości zdefiniowane na diagramie możemy zmieniać ręcznie zestawem komend `\setlength`, jednak jest to dość żmudna procedura wymagająca uzgadniania wartości zależnych od siebie rozmiarów i podatna na błędy. Rozwiązaniem jest użycie pakiety `geometry`[1], który znacznie ułatwia zmianę formatu strony.

[1] https://www.ctan.org/pkg/geometry

IV. Wygląd tekstu

IV.1. Czcionki

Dokumenty LaTeX-a mają przypisane trzy domyślne kroje czcionki: szeryfową, bezszeryfowany oraz `stałej szerokości znaku`. Domyślnym krojem dokumentu jest krój szeryfowy, który jest tradycyjnie uważany za mniej męczący przy czytaniu dłuższych tekstów. Między krojami możemy przełączać się deklaracjami (ograniczonymi nawiasami klamrowymi) lub komendami. Oprócz krojów możemy również zmieniać *wagę* (pogrubienie) oraz *kształt* czcionki. Podsumowanie dostępnych komend i deklaracji do zmiany wyglądu czcionki znajduje się pod adresem: http://www.fizyka.umk.pl/~gkowzan/latex/zadania/kroje.pdf. Trzecią opcją, potencjalnie zwiększającą czytelność kodu źródłowego, jest użycie środowiska o takiej samej nazwie jak odpowiednia deklaracja:

```
\begin{scshape}
To jest tekst napisany kapitalikami.
\end{scshape}
```

TO JEST TEKST NAPISANY KAPITALIKAMI.

Możemy też zmieniać rozmiar czcionek odpowiednimi deklaracjami, których podsumowanie znajduje się pod adresem: http://www.fizyka.umk.pl/~gkowzan/latex/zadania/rozmiary.pdf.

IV.2. Zmiana czcionki dokumentu

Domyślną rodziną czcionek w LaTeX-u jest rodzina czcionek *Computer Modern*. Jeśli chcemy zmienić rodzinę czcionek dokumentu, to najlepszym sposobem jest odwiedzenie *Katalogu czcionek LaTeX-a*[1], wybranie odpowiedniej i sprawdzenie kodu źródłowego przedstawionych przykładów.

W ogólności, możemy określić wszystkie możliwe parametry czcionki, która ma być wykorzystana do złożenia fragmentu tekstu, tj. kodowanie, rodzinę, serię (wagę i szerokość), kształt oraz rozmiar serią następujących deklaracji:

```
\fontencoding{kodowanie}\fontfamily{rodzina}\fontseries{waga_szerokość}
\fontshape{kształt}\fontsize{rozmiar}{wysokość}\selectfont
```

Nie należy mylić kodowania czcionki z kodowaniem pliku źródłowego, w praktyce zawsze będziemy używać czcionek z kodowaniem `T1`, co ma odzwierciedlenie w preambule dokumentów, które już składaliśmy w postaci komendy `\usepackage[T1]{fontenc}`. Należy również pamiętać o wywołaniu komendy `\selectfont`, w innym przypadku deklaracje nie odniosą skutku. Przykładowo, jeśli chcemy złożyć kawałek tekstu czcionką *Paratype Sans Narrow*, nie zmieniając przy tym żadnych innych parametrów czcionki, to napiszemy:

```
{\fontfamily{PTSansNarrow-TLF}\selectfont%
To jest fragment tekstu złożony rodziną Paratype Sans Narrow}.
```

To jest fragment tekstu złożony rodziną Paratype Sans Narrow.

Możemy też użyć skrótowej formy pozwalającej określić wszystkie atrybuty czcionki naraz:

[1] http://www.tug.dk/FontCatalogue/

Tablica 3: Lista najpopularniejszych czcionek LaTeX-owych i ich symboli (argumentów `\fontfamily`)

Symbol	Nazwa rodziny
	Szeryfowe
`cmr`	Computer Modern (domyślna)
`lmr`	Latin Modern
`pbk`	Bookman
`bch`	Charter
`pnc`	New Century Schoolbook
`ppl`	Palatino
`ptm`	Times
	Bezszeryfowe
`cmss`	Computer Modern Sans Serif (domyślna)
`lmss`	Latin Modern Sans Serif
`pag`	Avant Garde
`phv`	Helvetica
	O stałej szerokości znaków
`cmtt`	Computer Modern Typewriter (domyślna)
`lmtt`	Latin Modern
`pcr`	Courier

Tablica 4: Serie i kształy czcionek, możliwe argumenty `\fontseries` oraz `\fontshape`.

(a) Serie czcionek.

Symbol	Znaczenie
`m`	Średnia (domyślna)
`b`	Pogrubiony
`bx`	Pogrubiony szeroki
`sb`	Częściowo pogrubiony
`c`	Ściśnięty

(b) Kształty czcionek.

Symbol	Znaczenie
`n`	Normalny (domyślny)
`it`	Kursywa
`sl`	Pochyły
`sc`	Kapitaliki

```
{\usefont{T1}{yv1d}{\seriesdefault}{\shapedefault}\selectfont%
To jest fragment tekstu złożony rodziną Venturis}.
```

To jest fragment tekstu złożony rodziną Venturis.

Można się domyśleć, że są zdefiniowane domyślne: kodowanie, rodzina, seria oraz kształt, odpowiednio: `\encodingdefault`, `\familydefault`, `\seriesdefault`, `\shapedefault`. Było już wspomniane, że w każdym dokumencie są zdefiniowane domyślne rodziny czcionek, szeryfowa kryje się pod komendą `\rmdefault`, bezszeryfowa—`\sfdefault`, o stałej szerokości znaku—`\ttdefault`. Naturalnie możemy je zmienić, przykładowo:

```
\renewcommand*\rmdefault{ppl}
\renewcommand*\sfdefault{phv}
\renewcommand*\ttdefault{lmt}
\renewcommand*\familydefault{\sfdefault}
```

Ustawia czcionkę szeryfową na *Palatino*, bezszeryfową na *Helvetica* i o stałej szerokości znaków na *Latin Modern*, a następnie ustawia domyślną czcionkę dokumentu na bezszeryfową. W praktyce z czcionką albo zestawem czcionek zazwyczaj stowarzyszony jest pakiet, który po załadowaniu zmienia wartości powyższych komend i oferuje dodatkowe opcje skonfigurowania danej czcionek. Takie pakiety zazwyczaj nie zmieniają wartości `\familydefault`.

Każda czcionka może zdefiniować i przyjmować dowolne wartości *serii* oraz *kształtu*, w praktyce ich zbiór jest dość ograniczony i Tabele 4a oraz 4b przedstawiają najczęściej dostępne wartości i ich znaczenie.

IV.3. Standardowe środowiska

Justowanie Tekst w LaTeX-u jest wyjustowany do lewej i do prawej strony. Wybrane partie tekstu możemy wyrównać do lewej, do prawej albo wyśrodkować odpowiednimi deklaracjami lub środowiskami:

Działanie	Deklaracja	Środowisko
wyrównanie do lewej	`\raggedleft`	`\begin{flushleft}`
wyrównanie do prawej	`\raggedright`	`\begin{flushright}`
wyśrodkowanie	`\centering`	`\begin{center}`

Możemy też zastosować obustronne wcięcie, zazwyczaj stosowane do przytaczania cytatów, środowiskami `quote` oraz `quotation`. Gdzie jedyna różnica między nimi jest taka, że w `quotation` nowe akapity są wcięte i nie ma nimi odstępu, a w `quote` wcięć nie ma, ale są odstępy.

Listy, wyliczenia, słowniki Standardowo w LaTeX-u są dostępne trzy środowiska generujące listy: `itemize`, `enumerate` oraz `description`. `itemize` tworzy nienumerowane listy, domyślnie wcięte w stosunku do reszty tekstu i z zwiększonymi odstępami między elementami listy:

```
\begin{itemize}
\item Pierwszy element listy.
\item Drugi element listy

  Ten element składa się z dwóch akapitów.
\item Ostatni element.
```

```
\end{itemize}
```

- Pierwszy element listy.
- Drugi element listy
  Ten element składa się z dwóch akapitów.
- Ostatni element.

`enumerate` z kolei tworzy listę numerowaną:

```
\begin{enumerate}
\item Pierwszy element listy.
\item Drugi element listy

  Ten element składa się z dwóch akapitów.
\item Ostatni element.
\end{enumerate}
```

1. Pierwszy element listy.
2. Drugi element listy
   Ten element składa się z dwóch akapitów.
3. Ostatni element.

Natomiast `description` pozwala sformatować słownik:

```
\begin{description}
\item[Pojęcie 1] Definicja 1.
\item[Pojęcie 2] Definicja 2.
\item[Pojęcie 3] Definicja 3.
\end{description}
```

Pojęcie 1 Definicja 1.
Pojęcie 2 Definicja 2.
Pojęcie 3 Definicja 3.

Zarówno `itemize` jak i `enumerate` można zagnieżdżać:

- To jest pierwszy poziom, to już widzieliśmy.
  - To jest drugi poziom. Żeby był rozróżnialny mamy inny symbol po lewej stronie i większe wcięcie.
    - To jest trzeci poziom.
      - I czwarty, ostatni. Dla wyższych poziomów nie ma już zdefiniowanych innych symboli po lewej stronie.

Teraz `enumerate`:

1. To jest pierwszy poziom, to już widzieliśmy.
   (a) To jest drugi poziom. Jak w `itemize` wcięcie jest większe i numeracja jest inna.
      i. To jest trzeci poziom.

A. I czwarty, ostatni. Dla wyższych poziomów nie ma już zdefiniowanych innych styli numeracji.

Zmianę wyglądu symboli `itemize` dla czterech poziomów zagnieżdżenia zmieniamy redefiniując komendy `\labelitemi`, `\labelitemii`, `\labelitemiii`, `\labelitemiv`. Przykładowo:

```
\renewcommand\labelitemi{$\diamond$}
\begin{itemize}
\item Kopnięty kwadrat po lewej.
\end{itemize}
```

◇ Kopnięty kwadrat po lewej.

Przy redefiniowaniu tych komend pomocna może być najbardziej kompletna lista symboli dostępnych w LaTeX-u z instrukcją ich uzyskania w dokumencie[2].

W przypadku `enumerate` redefiniujemy komendy `\labelenumi`, `\labelenumii`, `\labelenumiii`, `\labelenumiv`. Z tym że z każdym poziomem jest jeszcze powiązany *licznik*: `enumi`, `enumii`, `enumiii`, `enumiv`, odpowiednio. W części III.5 wymienione były style numerowania stron, do każdego z tych styli jest przypisana komenda o takiej samej nazwie, pozwalająca wyświetlić wartość licznika. Przykładowo:

```
\renewcommand\labelenumi{\textsf{\textbf{(\Alph{enumi})}}}
\begin{enumerate}
\item To jest niezbyt użyteczny przykład list numerowanej z\dots
\item \dots dwoma elementami.
\end{enumerate}
```

(A) To jest niezbyt użyteczny przykład list numerowanej z…
(B) …dwoma elementami.

Definiowanie własnych list W ogólności możemy definiować własne listy. Możemy określić szerokość pola numeracji, wcięcia, odstępy między akapitami w obrębie elementu, odstępy między elementami, itd. Służy do tego środowisko `list`

```
\begin{list}{def_etykiety}{deklaracje}
\item ...
...
\end{list}
```

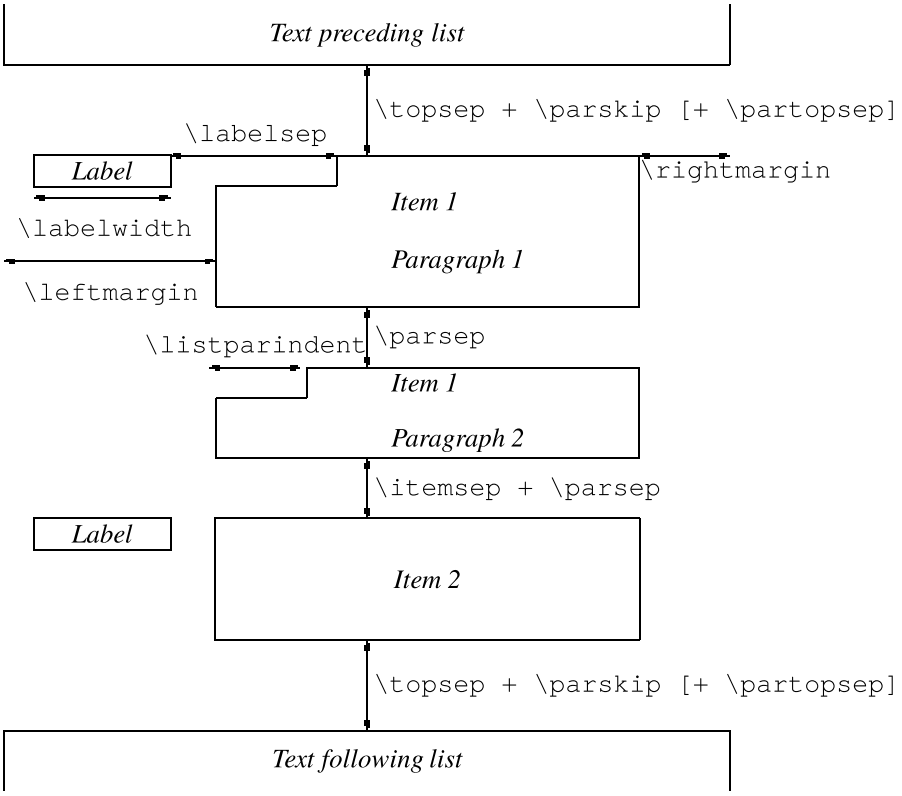
gdzie *`def_etykiety`* określa formatowanie etykiety, czyli symbolu albo numeracji, a *`deklaracje`* określają odstępy, wcięcia, itd. dla tej listy.

W najprostszym przypadku, czyli własnej wersji nienumerowanej listy w stylu `itemize`, argument *`def_etykiety`* to po prostu tekst albo komenda wyświetlająca interesujący nas symbol. W miejsce `\item` jest po prostu wstawiane to, co podamy jako argument *`def_etykiety`*.

Jeśli chcemy stworzyć listę numerowaną, to najpierw musimy zdefiniować nowy licznik. Robimy to komendą `\newcounter{`*`nazwa`*`}`, gdzie *`nazwa`* oczywiście nie może kolidować z wbudowanymi lub wcześniej zdefiniowanymi licznikami. Poza tym żeby móc używać tego licznika w

[2] ftp://ftp.gust.org.pl/TeX/info/symbols/comprehensive/symbols-a4.pdf

definiowanej liście musimy wywołać komendę `\usecounter{licznik}`, najlepiej włączając ją do argumentu *deklaracje*.



Rysunek 3: Parametry określające formatowanie list.

Argument *deklaracje* składa się w dużej części z wywołań komendy `\setlength` zmieniających parametry określające formatowanie listy. Poniżej znajduje się opis tych parametrów, a Rysunek 3 ilustruje ich znaczenie. Godny uwagi jest też bardziej obszerny artykuł *Customizing LaTeX lists*[3] twórców LaTeX-a.

`\partopsep` odstęp pionowy, który jest dodawany przed listą i po liście do odstępu `\topsep + \parskip`, jeśli akapit poprzedzający lub następujący po liście, odpowiednio, jest oddzielony od listy pustą linijką; tj. dodatkowy odstęp, gdy lista tworzy własny akapit;
`\parsep` odstęp między akapitami w obrębie elementu listy;
`\itemsep` odstęp między elementami listy;
`\leftmargin` odstęp między lewą krawędzią środowiska a tekstem elementów listy; dla listy pierwszego poziomu jest to odstęp od lewego marginesu dokumentu;
`\rightmargin` odstęp między lewą krawędzią środowiska a tekstem elementów listy;
`\listparindent` wcięcie pierwszej linii akapitu elementu listy względem `\leftmargin`; nie stosuje się do pierwszego akapity elementu listy;
`\labelwidth` szerokość pola etykiety, domyślnie etykieta jest wyrównana do prawej krawędzi;
`\labelsep` odstęp między polem etykiety a tekstem listy;
`\itemindent` wcięcie pierwszej linii elementu listy względem `\leftmargin`;

Poniżej przykład nowej listy stworzonej środowiskiem `list`:

```
\newcounter{myfig}
\begin{list}{\bfseries\upshape Rysunek \arabic{myfig}:}
```

[3] https://www.ntg.nl/maps/11/33.pdf

```
  {\usecounter{myfig}
\setlength{\labelwidth}{2cm}\setlength{\leftmargin}{2.6cm}
\setlength{\labelsep}{0.5cm}\setlength{\rightmargin}{1cm}
\setlength{\parsep}{0.5ex plus0.2ex minus0.1ex}
\setlength{\itemsep}{0ex plus0.2ex} \slshape}
\item Schemat przetwarzania plików źródłowych \LaTeX-a.
\item Parametry określające formatowanie list.
\item Jakiś inny rysunek.
\end{list}
```

Rysunek 1: *Schemat przetwarzania plików źródłowych LaTeX-a.*
Rysunek 2: *Parametry określające formatowanie list.*
Rysunek 3: *Jakiś inny rysunek.*

Jeśli zamierzamy więcej niż raz użyć zdefiniowanej przez nas listy, to nie ma sensu za każdym razem wpisywać długiej listy deklaracji i poleceń określających wygląd etykiety. Lepiej zdefiniować własne środowisko:

```
\newenvironment{figlist}{%
\begin{list}
  {\bfseries\upshape Rysunek \arabic{myfig}:}
  {\usecounter{myfig} ... \slshape}}
{\end{list}}
```

które potem może być używane jak każde inne środowisko

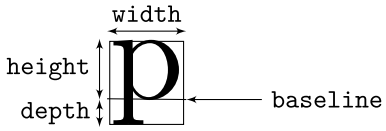
```
\begin{figlist}
\item ...
...
\end{figlist}
```

V. Pudełka

Pudełka w TEX-u to kawałki tekstu traktowane jako nierozerwalne całości. Najprostszym przykładem pudełka TEX-u jest pojedynczy znak. Takie pudełko może być przemieszczane w lewo, w prawo, w górę i w dół, lecz LATEX nie będzie w żaden sposób zmieniał jego zawartości i względnego rozmieszczenia elementów w środku. Z pudełek oczywiście można składać większe pudełka i to jest dokładnie to co robi LATEX przy formatowaniu tekstu: pojedyncze znaki są wkładane do pudełek, a potem te pudełka są umieszczane w pudełkach reprezentujących linie tekstu. W obrębie pudełek linii umieszczane są elastyczne odstępy między pudełkami reprezentującymi słowa. Pudełka linii są składane pionowo w pudełka akapitów, a między nimi są umieszczane odstępy elastyczne. Ostatecznie, pudełka akapitów są umieszczane w pudełku reprezentującym treść strony. To pudełko jest umieszczane razem z pudełkami nagłówka i stopki do pudełka reprezentującego całą stronę.

LATEX rozróżnia trzy rodzaje pudełek: pudełka LR, pudełka akapitów oraz pudełka linii. Pudełka LR (left-right) porządkują zawartość od lewej do prawej w obrębie pojedynczej linijki. Pudełka akapitów składają się z pionowo uporządkowanych linijek.[1] Podczas gdy pudełko linii składa jest wypełnionym prostokątem, który zazwyczaj służy do rysowania pionowych i poziomych linii.

V.1. Pudełka LR



Rysunek 4: Wymiarowanie znaków i pudełek LR w LATEX-u.

Określając wielkość pudełek znaków LATEX nie „mierzy" sam wielkości znaków, lecz pobiera tę informację z pliku czcionki. Tj. dostępne czcionki zawierają informację o wielkości pudełek odpowiadających każdemu ze znaków. Co ciekawe, nie zawsze zdefiniowane pudełko jest wystarczająco duże, aby pomieścić znak. W ogólności takie pudełko jest określane przez trzy wielkości, jak pokazano na Rysunku 4, tj. *linię bazową*, *wysokość* oraz *głębokość*. Składając pudełka znaków w pudełku LR, LATEX umieszcza je tak żeby przylegały do siebie[2] i żeby linia bazowa każdego ze znaków była na tej samej wysokości.

Do ręcznego tworzenia pudełek LR mamy do dyspozycji komendy

```
\mbox{tekst}
\makebox[szerokość][położenie]{tekst}
\fbox{tekst}
\framebox[szerokość][położenie]{tekst}
```

gdzie krótsze komendy tworzą pudełko o szerokości równej dokładnie szerokości zawartego tekstu, a `\fbox` ponadto rysuje obramowanie pudełka. Dłuższe komendy pozwalają określić jaka jest szerokość pudełka oraz jak rozmieścić tekst w pudełko. Bez parametru *położenie* tekst jest wyśrodkowany, z parametrem `l`—wyrównany do lewej, `r`—wyrównany do prawej, `s`—rozciągnięty na całe pudełko. Z tym że aby tekst był faktycznie rozciągnięty, musimy w pudełko umieścić jakiś odstęp elastyczny (np. `\hfill`), aby było co rozciągać. Parametr *szerokość* może być określony

[1] Pewną analogią-u są elementy `span` oraz `div` w HTML-u albo w ogólności elementy o stylu `display: inline` oraz `display: block`, odpowiednio.

[2] Od tego jest kilka wyjątków, patrz: *kerning* albo *ligatury*.

tak jak każda inna długość w LaTeX-u, ale może być też określony jako krotność wielkości lokalnych dla danego pudełka:

`\width` to naturalna szerokość pudełka,
`\height` to naturalna wysokość pudełka,
`\depth` to naturalna głębokość pudełka,
`\totalheight` to `\height` + `\depth`.

Dodatkowo, o grubości obramowania decyduje wartość[3] `\fboxrule` (domyślnie 0.4pt), a o odstępie między pudełkiem a obramowaniem—`\fboxsep` (domyślnie 3pt).

PRZYKŁADY

```
To jest przykład pudełka \fbox{z obramowaniem}.\\
Tutaj mamy pudełko \makebox[2\width]{bez obramowania} o szerokości dwa razy
większej od naturalnej.\\
Tutaj mamy pudełko \makebox[2\width]{bez obramowania} o szerokości dwa razy
większej od \mbox{naturalnej}.\\
Mimo że pudełka \texttt{mbox} są niewidoczne, to można je wykorzystać do
przeszkodzenia w dzieleniu słów.\\
Tutaj z kolei jest pudełko o \makebox[0pt]{zerowej} szerokości.\\
Tutaj mamy \makebox[2\width][s]{pudełko\dotfill elastyczne}, a tutaj mamy
pudełko wyrównane \makebox[2\width][r]{do prawej}.
```

To jest przykład pudełka z obramowaniem.
Tutaj mamy pudełko bez obramowania o szerokości dwa razy większej od naturalnej.
Tutaj mamy pudełko bez obramowania o szerokości dwa razy większej od naturalnej.
Mimo że pudełka `mbox` są niewidoczne, to można je wykorzystać do przeszkodzenia w dzieleniu słów.
Tutaj z kolei jest pudełko zerowej szerokości.
Tutaj mamy pudełko ..... elastyczne, a tutaj mamy pudełko wyrównane do prawej.

Pudełka może również zapisywać i używać ich później. Do stworzenia nowego pudełka używamy komendy

```
\newsavebox{\nazwa}
```

Żeby zapisać coś do pudełka mamy parę komend analogicznych do `\mbox` i `\makebox`:

```
\sbox{\nazwa}{tekst}
\savebox{\nazwa}[szerokość][położenie]{tekst}
```

oraz dodatkowo środowisko

```
\begin{lrbox}{nazwa}
tekst
\end{lrbox}
```

[3] Zmieniana poleceniem `\setlength`.

Żeby wstawić zawartość pudełka używamy komendy

```
\usebox{\nazwa}
```

V.2. Podnoszenie pudełek

Możemy tworzyć też pudełka LR, które są przesunięte w pionie względem linii bazowej

```
\raisebox{lift}[wysokość][głębokość]{tekst}
```

Najlepiej wyjaśnić to na przykładach:

```
To jest linijka nad przesuniętym tekstem. To jest linijka nad przesuniętym tekstem.\\
1. Przesuwamy \raisebox{1ex}{ten} kawałek o 1ex wyżej a \raisebox{-1ex}{ten} o
1ex niżej.\\
To jest linijka pod przesuniętym tekstem. To jest linijka pod przesuniętym tekstem.
```

To jest linijka nad przesuniętym tekstem. To jest linijka nad przesuniętym tekstem.
1. Przesuwamy $^{\text{ten}}$ kawałek o 1ex wyżej a $_{\text{ten}}$ o 1ex niżej.
To jest linijka pod przesuniętym tekstem. To jest linijka pod przesuniętym tekstem.

```
To jest linijka nad przesuniętym tekstem. To jest linijka nad przesuniętym tekstem.\\
2. Przesuwamy \raisebox{1ex}[\height-1ex][\depth]{ten} kawałek o 1ex wyżej a
\raisebox{-1ex}[\height][\depth-1ex]{ten} o 1ex niżej.\\
To jest linijka pod przesuniętym tekstem. To jest linijka pod przesuniętym tekstem.
```

To jest linijka nad przesuniętym tekstem. To jest linijka nad przesuniętym tekstem.
2. Przesuwamy $^{\text{ten}}$ kawałek o 1ex wyżej a $_{\text{ten}}$ o 1ex niżej.
To jest linijka pod przesuniętym tekstem. To jest linijka pod przesuniętym tekstem.

W pierwszym przykładzie nie podaliśmy argumentów opcjonalnych, więc wysokość i głębokość pudełka została wyliczona automatycznie. LaTeX sprawdził wysokości i szerokości wszystkich pudełek w obrębie linijki i na podstawie największej wysokości i głębokości określił wysokość i głębokość *całej* linijki, tak aby znaki z sąsiadujących linijek się nie pokrywały. W drugim przypadku, od wyliczonych wartości wysokości i głębokości odjęliśmy `1ex`, czyli tyle ile wynosiło przesunięcie, żeby wysokość linijki była taka jakby przesunięcia w ogóle nie było.

V.3. Pudełka akapitowe

Pudełka akapitowe składamy komendą

```
\parbox[położenie]{szerokość}{tekst}
```

oraz środowiskiem

```
\begin{minipage}[położenie]{szerokość}
tekst
\end{minipage}
```

Z tym że środowisko `minipage` jest bardziej ogólne, bo oprócz zwykłego tekstu może zawierać również inne środowiska, np. listy. Parametr *`położenie`* określa wyrównanie *pionowe* pudełka względem linii bazowej, na której znajduje się pudełko. Domyślnie zawartość jest wyśrodkowana, z parametrem `b` dolna krawędź pudełka jest na wysokości linii bazowej, a z parametrem `t`—górna.

PRZYKŁADY

```
\parbox{3.5cm}{To jest pudełko o szerokości 3.5 cm, wyśrodkowane względem linii
  bazowej} \hfill BIEŻĄCA LINIA \hfill \parbox{7.5cm}{To jest trochę szersze
  pudełko.  Wąskie akapity trudno jest sformatować tak aby wyglądały
  estetycznie, więc tworzenie takich wąskich pudełek zazwyczaj nie jest dobrym
  pomysłem.}
```

To jest pudełko o szerokości 3.5 cm, wyśrodkowane względem linii bazowej BIEŻĄCA LINIA To jest trochę szersze pudełko. Wąskie akapity trudno jest sformatować tak aby wyglądały estetycznie, więc tworzenie takich wąskich pudełek zazwyczaj nie jest dobrym pomysłem.

```
\parbox[b]{3.5cm}{To jest pudełko o szerokości 3.5 cm, wyśrodkowane względem linii
  bazowej} \hfill BIEŻĄCA LINIA \hfill \parbox[t]{7.5cm}{To jest trochę szersze
  pudełko.  Wąskie akapity trudno jest sformatować tak aby wyglądały
  estetycznie, więc tworzenie takich wąskich pudełek zazwyczaj nie jest dobrym
  pomysłem.}
```

To jest pudełko o szerokości 3.5 cm, wyśrodkowane względem linii bazowej BIEŻĄCA LINIA To jest trochę szersze pudełko. Wąskie akapity trudno jest sformatować tak aby wyglądały estetycznie, więc tworzenie takich wąskich pudełek zazwyczaj nie jest dobrym pomysłem.

Jeśli chcemy złożyć To jest pierwsze pudełko, nie ma w nim nic ciekawego To jest drugie pudełko, też nie ma w nim nic ciekawego akapit taki jak ten, tj. z dwoma pudełkami wystającymi do góry, ale wyrównanymi dolnymi krawędziami do dołu, to wydawałoby się logiczne, że poniższy kod zadziała:

```
Jeśli chcemy złożyć
\begin{minipage}[b]{7cm}
  \parbox[t]{3.5cm}{To jest pierwsze pudełko, nie ma w nim nic ciekawego}%
  \hfill%
  \parbox[t]{3cm}{To jest drugie pudełko, też nie ma w nim nic ciekawego}
\end{minipage}
akapit taki jak ten\dots
```

Jeśli chcemy złożyć To jest pierwsze pudełko, nie ma w nim nic ciekawego To jest drugie pudełko, też nie ma w nim nic ciekawego
akapit taki jak ten...

Jak widać efekt jest inny od zamierzonego. Wynika to z tego, że używając poleceń `\parbox` sprawiamy, że ich cała zawartość jest traktowana jak jedno pudełko, czyli jeden znak. To oznacza, że zawartość środowiska `minipage` wg LaTeX-a składa się z dwóch znaków obok siebie na jednej linii. Działanie parametru opcjonalnego `b`, to jest wyrównanie krawędzi dolnej do linii bazowej, tak naprawdę mówi żeby najniższa linijka tekstu w `minipage` była wyrównana do linii bazowej, a ponieważ tekst składa się efektywnie z dwóch znaków (więc jednej linijki), to wg LaTeX-a wszystko jest w porządku. Żeby pozyskać zamierzony efekt trzeba dodać sztuczną linijkę do środowiska `minipage`:

```
Jeśli chcemy złożyć
\begin{minipage}[b]{7cm}
  \parbox[t]{3.5cm}{To jest pierwsze pudełko, nie ma w nim nic ciekawego}%
  \hfill%
  \parbox[t]{3cm}{To jest drugie pudełko, też nie ma w nim nic ciekawego} \\ \mbox{}
\end{minipage}
akapit taki jak ten...
```

gdzie `\mbox{}` jest kluczowe, ponieważ nowa linia nie może być pusta.

Polecenie `\parbox` oraz środowisko `minipage` przyjmują jeszcze dwa dodatkowe argumenty opcjonalne:

```
\parbox[położenie][wysokość][wew_położenie]{szerokość}{tekst}
```

To znaczy, możemy z góry określić wysokość pudełka i możemy określić położenie zawartości wewnątrz pudełka. Argument `wew_położenie` przyjmuje wartości: `t`—wyrównanie do góry, `b`—wyrównanie do dołu, `c`—wycentrowanie, `s`—wypełnienie pudełka (potrzebne elastyczne odstępy wewnątrz).

PRZYKŁADY

```
\begin{minipage}[t][3cm][t]{4cm}
To jest \texttt{minipage} o wysokości 3~cm z zawartością u góry.
\end{minipage}
\hrulefill
\parbox[t][3cm][c]{4cm}{Te same wymiary co \texttt{minipage}.  Wyrównanie
  zawartości do środka.}
\hrulefill
\begin{minipage}[t][3cm][b]{4cm}
To jest \texttt{minipage} o wysokości 3~cm z zawartością u dołu.
\end{minipage}
```

To jest minipage o wysokości 3 cm z zawartością u góry.

Te same wymiary co minipage. Wyrównanie zawartości do środka.

To jest minipage o wysokości 3 cm z zawartością u dołu.

V.4. Pudełka prostokątne

Pudełka które są wypełnionymi prostokątami wykonujemy komendą

```
\rule[lift]{szerokość}{wysokość}
```

gdzie znaczenie argumentu *lift* jest takie same jak w komendzie \raisebox. Przy pracy z pudełkami przydatne są też komendy zdefiniowane w pakiecie calc

```
\widthof{tekst}
\heightof{tekst}
\depthof{tekst}
\totalheightof{tekst}
```

które zwracają odpowiednie wymiary podanego tekstu.

PRZYKŁADY

```
To jest linijka z \rule{2mm}{2mm} kwadracikiem.\\
To jest linijka z\setlength{\dimen0}{\widthof{przekreśleniem}}
\makebox[0pt]{\hspace{\dimen0}
\rule[0.2\baselineskip]{\widthof{przekreśleniem}}{0.5pt}
}przekreśleniem.
```

To jest linijka z ■ kwadracikiem.
To jest linijka z ~~przekreśleniem~~.

V.5. Zagnieżdżanie pudełek

Nic nie stoi na przeszkodzie żeby zagnieżdżać okna różnego typu. Możemy np. zagnieździć parbox w fbox żeby otrzymać obramowanie albo parbox w raisebox żeby przesunąć pudełko. Pudełka akapitowe mogą być też zapisywane poleceniem savebox tak samo jak pudełka LR.

VI. Tabele

Do definiowania tabel wykorzystuje się środowiska

```
\begin{tabular}[położenie]{kolumny}     rzędy     \end{tabular}
\begin{array}[położenie]{kolumny}       rzędy     \end{array}
```

gdzie `tabular` jest używane w normalnym tekście, a `array` w trybie matematycznym. Znaczenie argumentu *położenie* jest takie same jak dla środowiska `minipage`, ponieważ każda tabela jest zawarta w środowisku `minipage`.

Parametr *kolumny* określa formatowanie kolumn. Każdej kolumnie tabeli musi odpowiadać specyfikacja jej formatowania, dodatkowo można umieszczać symbole reprezentujące obramowanie i dodatkowe odstępy między kolumnowe. Dostępne symbole określające formatowanie kolumn to:

`l` zawartość kolumny jest wyrównana do lewej strony,
`r` zawartość kolumny jest wyrównana do prawej strony,
`c` zawartość kolumny jest wyśrodkowana,
`p{`*szer*`}` zawartość kolumny jest umieszczona w pudełku `\parbox[t]{`*szer*`}{...}`,
`*{`*num*`}{`*kol*`}` specyfikator *col* jest powtórzony *num* razy, np. specyfikacja `*{5}{|c}|` daje `|c|c|c|c|c|`.

Poza tymi symbolami formatowania kolumn można jeszcze używać następujących symboli:

`|` rysuje pionową linię,
`||` rysuje podwójną pionową linię,
`@{`*tekst*`}` powoduje wstawienie *tekst* w każdym rzędzie, między kolumnami lub na skrajach tabeli. Użycie tej konstrukcji *usuwa* automatyczny odstęp między kolumnami. Żeby go wprowadzić ponownie trzeba umieścić komendę `\hspace{ }` w *tekst*.

Rzędy zawierają natomiast właściwą zawartość tabeli. Każdy rząd jest zakańczany komendą `\\`; w obrębie każdego rzędu kolumny oddzielane są symbolem `&`. Każda komórka jest interpretowana tak, jakby była otoczona nawiasami klamrowymi, więc umieszczane deklaracje (np. zmiany czcionki) mają działanie lokalne. W obrębie tabeli możemy używać zestawu komend specjalnych:

`\hline` umieszczane przed pierwszym rzędem albo natychmiast po komendzie `\\`; wstawia linię poziomą na całą szerokość tabeli.
`\cline {m-n}` wstawia linię poziomą od lewej krawędzi kolumny `m` do prawej krawędzi kolumny `n`.
`\multicolumn {num}{col}{tekst}` scala `num` komórek i ustawia formatowanie scalonej komórki na `col`.
`\vline` wstawia pionową linię o wysokości danej komórki.

Na wygląd tabel ma również wpływ poniższy zestaw długości:

`\tabcolsep` długość równa połowie odstępu między kolumnami w środowisku `tabular`,
`\arraycolsep` długość równa połowie odstępu między kolumnami w środowisku `array`,
`\arrayrulewidth` grubość pionowym i poziomych linii w tabelach,
`\doublerulesep` odstęp między podwójnymi liniami w tabelach,

których wartość można standardowo zmieniać komendą `\setlength`. Ponadto istnieje komenda `\arraystretch`, która jest mnożnikiem (analogicznie do `\baselinestretch`) określającym odstęp między rzędami tabeli. Jej domyślna wartość to `1.0` i może być zmieniona komendą `\renewcommand\arraystretch{`*mnożnik*`}`.

Istnieje również środowisko `tabular*`, w którym można z góry ustalić szerokość tabeli. Można by się spodziewać że dopasowanie szerokości odbywa się przez odpowiednie zwiększanie szerokości kolumn, lecz zamiast tego zwiększany jest jedynie odstęp między wybranymi kolumnami, tak aby osiągnąć wymaganą

szerokość całej tabeli. To sprawia, że te środowisko nie jest zbyt użyteczne i nie jest tutaj omówione.

VI.1. Pakiety rozszerzające

PAKIET `array` Pakiet `array` wprowadza nowe definicje formatowania kolumn:

`m{szer}` zawartość kolumny jest umieszczona w pudełku `\parbox{szer}{...}`,
`b{szer}` zawartość kolumny jest umieszczona w pudełku `\parbox[b]{szer}{...}`.

oraz symbole

`>{tekst}` wstawia *tekst* przed następną kolumną; przykładowo specyfikacja `>{\bfseries}rl` definiuje dwie kolumny, z czego w pierwszej tekst będzie pogrubiony i wyrównany do prawej, a w drugiej będzie złożony zwykłą czcionką i wyrównany do lewej.
`<{tekst}` wstawia *tekst* po poprzedniej kolumnie.
`!{tekst}` wstawia *tekst* między kolumnami, nie usuwając odstępu między kolumnami.

Ponadto możemy zdefiniować własne symbole formatowania kolumn komendą:

```
\newcolumntype{symbol}{deklaracje}
```

Przykładowo

```
\newcolumntype{C}{>{\itshape}c}
```

definiuje wyśrodkowaną kolumnę złożoną kursywą.

PAKIET `dcolumn` Automatycznie ładuje pakiet `array` i definiuje nowy symbol formatowania kolumn służący do formatowania liczb:

```
D{wej}{wyj}{num}
```

gdzie *wej* określa znak wejściowy reprezentujący separator dziesiętny, *wej* określa znak, który ma być wstawiony na jego miejsce w tekście, a *num* określa maksymalną ilość cyfr po separatorze dziesiętnym. Jeśli *num* jest ujemny to kolumna jest wycentrowana względem separatora dziesiętnego, a jeśli nie to jest wyrównana do prawej. Tzn. jeśli chcemy wstawić do tabeli szereg liczb rzeczywistych i mamy je zapisane z częścią dziesiętną oddzieloną kropkami, a—zgodnie z polską normą—chcemy jako separatora dziesiętnego użyć przecinka i wyrównać liczby względem przecinka, to użyjemy deklaracji

```
D{.}{,}{-1}
```

! *Kolumny typu `D` są składane w trybie matematycznym, o którym będzie mowa później. Nie rodzi to problemów jeśli w danej kolumnie znajdują się tylko liczby, ale może powodować trudności przy składaniu nagłówków kolumn. Żeby z trybu matematycznego wrócić do trybu zwykłego należy użyć komendy `\text{zawartość}`.*

PAKIET `tabularx` Pakiet ten definiuje nowe środowisko `tabularx` pozwalające na wstawienie tabeli o z góry ustalonej szerokości i wskazanie których kolumn szerokość ma być elastycznie dopasowywania, tak aby otrzymać wskazaną szerokość. Przykładowo

```
\begin{tabularx}{10cm}{rXlX}
rzędy
\end{tabularx}
```

wstawi tabelę o szerokości 10 cm z drugą i czwartą kolumną o elastycznej szerokości.

VI.2. Przykłady tabel

W praktyce tworzenie większości tabel w LaTeX-u nie jest zbyt skomplikowane, aczkolwiek przydają się narzędzia, które automatycznie przetwarzają dane tabelaryczne na odpowiedni kod LaTeX-owy. Poniżej znajduje przykład jednej z prostszych tabel

```
\begin{center}
\begin{tabular}{rlrrr}
  Pozycja & Gracz & Punkty & Wygrane & Przegrane \\[0.5ex]
  1 & [Ence] Serral & 3350 & 221 & 84\\
  2 & IIIIIIIII & 1390 & 43 & 8\\
  3 & IlIlIlIlIlIl & 3326 & 160 & 56\\
  4 & [M3GA] Fietsendief & 1191 & 143 & 68\\
  5 & [lARMAl] ShoWTimE & 3297 & 228 & 78
\end{tabular}
\end{center}
```

Pozycja	Gracz	Punkty	Wygrane	Przegrane
1	[Ence] Serral	3350	221	84
2	IIIIIIIII	1390	43	8
3	IlIlIlIlIlIl	3326	160	56
4	[M3GA] Fietsendief	1191	143	68
5	[lARMAl] ShoWTimE	3297	228	78

gdzie zwiększyliśmy odstęp między pierwszym a drugim wierszem podając argument opcjonalny komendy łamania linii. Możemy zmodyfikować powyższą tabelę i dodać linie pionowe do niej:

```
\begin{center}
\begin{tabular}{r|l||r|rr}
  Pozycja & Gracz & Punkty & Wygrane & Przegrane \\[0.5ex]
  1 & [Ence] Serral & 3350 & 221 & 84\\
  2 & IIIIIIIII & 1390 & 43 & 8\\
  3 & IlIlIlIlIlIl & 3326 & 160 & 56\\
  4 & [M3GA] Fietsendief & 1191 & 143 & 68\\
  5 & [lARMAl] ShoWTimE & 3297 & 228 & 78
\end{tabular}
\end{center}
```

Pozycja	Gracz	Punkty	Wygrane	Przegrane
1	[Ence] Serral	3350	221	84
2	IIIIIIIII	1390	43	8
3	IlIlIlIlIlIl	3326	160	56
4	[M3GA] Fietsendief	1191	143	68
5	[lARMAl] ShoWTimE	3297	228	78

i dodać linie poziome

```
\begin{center}
\begin{tabular}{|r|l||r|rr|}
  \hline
  Pozycja & Gracz & Punkty & Wygrane & Przegrane \\\hline\hline
  1 & [Ence] Serral & 3350 & 221 & 84\\\hline
  2 & IIIIIIIII & 1390 & 43 & 8\\\hline
  3 & IlIlIlIlIlIl & 3326 & 160 & 56\\\hline
  4 & [M3GA] Fietsendief & 1191 & 143 & 68\\\hline
  5 & [lARMAl] ShoWTimE & 3297 & 228 & 78\\\hline
\end{tabular}
\end{center}
```

Pozycja	Gracz	Punkty	Wygrane	Przegrane
1	[Ence] Serral	3350	221	84
2	IIIIIIIII	1390	43	8
3	IlIlIlIlIlIl	3326	160	56
4	[M3GA] Fietsendief	1191	143	68
5	[lARMAl] ShoWTimE	3297	228	78

```
\renewcommand\arraystretch{1.2}
\begin{center}
\begin{tabular}{|c|r|l||c|c|r@{:}l|}\hline
  \multicolumn{7}{|c|}{\rule[-3mm]{0mm}{9mm}\bfseries
 Liga arcymistrzowska --- 2017 Sezon 1 --- Europa}\\ \hline
      &   & \itshape Gracz & \itshape Rasa & \itshape Punkty &
 \itshape W & \itshape P \\\hline\hline
  --  & 1 & [Ence] Serral & Zerg & 3350 & 221 & 84\\\hline
  ! & 2 & IIIIIIIII & Protoss & 1390 & 43 & 8\\\hline
  --  & 3 & IlIlIlIlIlIl & Protoss & 3326 & 160 & 56\\\hline
  ! & 4 & [M3GA] Fietsendief & Protoss & 1191 & 143 & 68\\\hline
  --  & 5 & [lARMAl] ShoWTimE & Protoss & 3297 & 228 & 78\\\hline
  $\uparrow$  & 6 & [IESGI] Nerchio & Zerg & 3070 & 206 & 99 \\\hline
\end{tabular}
\end{center}
```

Liga arcymistrzowska — 2017 Sezon 1 — Europa					
		Gracz	*Rasa*	*Punkty*	*W:P*
–	1	[Ence] Serral	Zerg	3350	221:84
!	2	IIIIIIIII	Protoss	1390	43:8
–	3	IlIlIlIlIlIl	Protoss	3326	160:56
!	4	[M3GA] Fietsendief	Protoss	1191	143:68
–	5	[lARMAl] ShoWTimE	Protoss	3297	228:78
↑	6	[IESGI] Nerchio	Zerg	3070	206:99

VII. Definiowanie komend i środowisk

VII.1. Liczniki

Poznaliśmy już niektóre liczniki, czyli *de facto* zmienne liczbowe, które są używane do numerowania elementów dokumentu, i które są inkrementowane przy wywoływaniu odpowiednich komend i których wartości są odczytywane i formatowanie przy pomocy innych komend. Przykładowo, licznik `section` służy do numerowania sekcji artykułów i jest zwiększany o jeden za każdym razem, gdy użyjemy komendy `\section`. Możemy się odwołać do jego wartości (ale nie wyświetlić) komendą `\value{section}`. Możemy wyświetlić jego wartość przy pomocy komend `\styl{section}`, gdzie *styl* jest jedną z wartości z listy w dziale III.5. Przykładowo, `\arabic{section}` wyświetli numer aktualnej sekcji cyfrą arabską. Każdemu licznikowi jest przypisana również komenda `\the`*licznik*, którą możemy wyświetlić wartość licznika. Właśnie definicja tych komend decyduje jak będzie wyglądać numeracja w nagłówkach rozdziałów, sekcji, itd. W preambule bieżącego dokumentu znajdują się poniższe redefinicje komend określające wygląd numeracji nagłówków:

```
\renewcommand\thesection{\Roman{section}}
\renewcommand\thesubsection{\thesection.\arabic{subsection}}
\renewcommand\thesubsubsection{\thesubsection.\Alph{subsubsection}}
```

W LaTeX-u domyślnie zdefiniowane są liczniki: sekcjonowania — `part`, `chapter`, `section`, `subsection`, `subsubsection`, `paragraph`, `subparagraph`, list numerowanych — `enumi`, `enumii`, `enumiii`, `enumiv`, i pozostałe — `page`, `equation`, `figure`, `table`, `footnote`, `mplfootnote`.

Nowy licznik definiujemy komendą

```
\newcounter{licznik}[nadrzędny]
```

gdzie *licznik* to nazwa nowego licznika. Jeśli *nadrzędny* jest podany, to musi być nazwą już istniejącego licznika i mówi o tym żeby zresetować *licznik* za każdym razem, gdy *nadrzędny* jest inkrementowany komendą `\stepcounter` lub `\refstepcounter`. Łatwo się domyśleć, że dla `section` licznikiem nadrzędnym jest `chapter`, dla `subsection` jest to `section`, itd.

Liczniki możemy modyfikować komendami:

`\setcounter{licznik}{num}` Ustawia `licznik` na wskazaną wartość `num`.
`\addtocounter{licznik}{num}` Dodaje do `licznika` wskazaną wartość `num`.
`\stepcounter{licznik}` Inkrementuje `licznik` i resetuje wszystkie liczniki podrzędne.

VII.2. Długości

Od długościach była już mowa w dziale II.4. W dziale o pudełkach prostokątnych (V.4) były opisane komendy zwracające szerokość, wysokość i głębokość podanego fragmentu tekstu. Mamy też do dyspozycji komendy, które przypisują wskazanej komendzie długości wartość odpowiadającą długości podanego tekstu. Są to:

- `\settodepth{\dlugosc}{tekst}`,
- `\settowidth{\dlugosc}{tekst}`,
- `\settoheight{\dlugosc}{tekst}`.

Nowe długości definiujemy z kolei komendą `\newlength{\dlugosc}`, domyślna długość wynosi `0pt`.

VII.3. Definiowanie komend

Komendy definiujemy jedną z trzech poniższych komend:

```
\newcommand{\nazwa}[narg][opc]{definicja}
\renewcommand{\nazwa}[narg][opc]{definicja}
\providecommand{\nazwa}[narg][opc]{definicja}
```

gdzie *nazwa* to nazwa nowej komendy, argument opcjonalny *narg* to ilość argumentów przyjmowanych przez komendę, a *opc* to wartość domyślna pierwszego argumentu. To znaczy, jeśli podamy *opc*, to możemy wywoływać zdefiniowaną komendę bez podawania pierwszego argumentu i wtedy w jego miejsce w ciele funkcji będzie wstawiona wartość *opc*. `\newcommand` pozwala zdefiniować nową komendę, jeśli podana nazwa nowej komendy jest już zajęta, to zostanie zwrócony błąd i dokument się nie skompiluje. Podobnie `\renewcommand` — pozwala zredefiniować istniejącą komendę i zwróci błąd jeśli komenda o takiej nazwie nie istnieje. Z kolei `\providecommand` zdefiniuje nową komendę, jeśli komenda o podanej nazwie nie istnieje i nic nie zrobi, jeśli komenda już istnieje.

Komendy bez argumentów są przydatne, jeśli często w dokumencie chcemy wprowadzić ten sam tekst i chcemy się ustrzec przed pomyłkami i nieczytelnym kodem. Koronnym przykładem jest komenda `\LaTeX` (LaTeX), która w specyficzny sposób formatuje ciąg znaków „LaTeX" i wymagałaby inaczej za każdym razem wprowadzania komend zmieniających krój czcionki oraz użycia `\raisebox` żeby odpowiednio podnieść i obniżyć pudełka niektórych liter. Przykładowo, wykrzyknik na marginesie w dziale VI (akapit *Pakiet* `dcolumn`) został złożony przy pomocy dwóch komend:

```
\newcommand{\exclamation}{{\normalfont\Large\sffamily !}} % wykrzyknik
\newcommand{\important}[1]{%
{\setlength{\fboxsep}{2mm}%
\noindent\makebox[0pt]{\hspace{-15mm}%
\raisebox{#1}[0pt][0pt]{\fbox{\exclamation}}}}} % pudełko na marginesie
```

Jeśli chcielibyśmy wyświetlić w tekście oznaczenie na wektor n-elementowy, to moglibyśmy użyć kodu `$x_1,\ldots,x_n$` ($x_1, \ldots, x_n$) i zdefiniować komendę

```
\newcommand{\xvec}{$x_1,\ldots,x_n$}
\xvec
```

$x_1, \ldots, x_n$

jeśli wprowadzalibyśmy to oznaczenie częściej. Przy pomocy argumentów możemy łatwo uogólnić ten przypadek

```
\newcommand{\xvec}[1]{$#1_1,\ldots,#1_n$}
\xvec{y}
```

$y_1, \ldots, y_n$

Wtedy zamiast wektora x-ów możemy na przykład wywołać `\xvec{y}` i dostać wektor y-ów. Jak można zauważyć, do wartości argumentów odwołujemy się w ciele funkcji konstrukcją `#num`. Możemy jeszcze bardziej uogólnić komendę, pozwalając na inne wartości indeksu początkowego i końcowego:

```
\newcommand{\xvec}[3]{$#1_{#2},\ldots,#1_{#3}$}
\xvec{y}{k}{m}
```

$y_k, \ldots, y_m$

Ponieważ prawdopodobnie najczęściej będziemy używać wektorów x-ów, wykorzystajmy argument opcjonalny:

```
\newcommand{\xvec}[3][x]{$#1_{#2},\ldots,#1_{#3}$}
\xvec{k}{m}\\
\xvec[z]{l}{n}
```

$x_k, \ldots, x_m$
$z_l, \ldots, z_n$

VII.4. Definiowanie środowisk

Środowiska definiuje się analogicznie do komend:

```
\newenvironment{nazwa}[narg][opc]{początek}{koniec}
\renewenvironment{nazwa}[narg][opc]{początek}{koniec}
```

gdzie znaczenie większości argument jest takie same jak przy definiowaniu komend. *początek* jest wstawiane w miejsce `\begin{nazwa}`, a *koniec* w miejsce `\end{nazwa}`. Przykładowo:

```
\newenvironment{mycomment}{\begin{quote}\small\itshape}{\end{quote}}
\begin{mycomment}
To jest środowisko z wcięciami po lewej i po prawej stronie, służące do
wprowadzania komentarzy do głównego tekstu.
\end{mycomment}
```

To jest środowisko z wcięciami po lewej i po prawej stronie, służące do wprowadzania komentarzy do głównego tekstu.

Wprowadźmy teraz nagłówek komentarza i numerację komentarzy:

```
\newcounter{com}
\newenvironment{mycomment}{%
\noindent\slshape Komentarz:%
\begin{quote}\small\itshape}
{\stepcounter{com}\hfill(\arabic{com})\end{quote}}
\begin{mycomment}
To jest środowisko z wcięciami po lewej i po prawej stronie, służące do
wprowadzania komentarzy do głównego tekstu.
\end{mycomment}
```

Komentarz:

To jest środowisko z wcięciami po lewej i po prawej stronie, służące do wprowadzania komentarzy do głównego tekstu. *(1)*

i autora komentarza z domyślnym autorem:

```
\newenvironment{mycomment}[1][G.~Kowzan]{%
\noindent\slshape Komentarz: #1%
\begin{quote}\small\itshape}
{\stepcounter{com}\hfill(\arabic{com})\end{quote}}
\begin{mycomment}[T.~Student]
To jest środowisko z wcięciami po lewej i po prawej stronie, służące do
wprowadzania komentarzy do głównego tekstu.
\end{mycomment}
```

Komentarz: T. Student

> *To jest środowisko z wcięciami po lewej i po prawej stronie, służące do wprowadzania komentarzy do głównego tekstu.* *(2)*

VIII. Wstawki

VIII.1. Elementy graficzne

Komendy potrzebne do umieszczania elementów graficznych są zawarte w pakiecie `graphicx`. Jeśli używamy programu `pdfLaTeX`[1], to możliwe jest umieszczanie grafiki w formatach: `pdf`, `jpeg`, `jbig2`, `png` oraz `eps`[2]. Pliki graficzne wstawia się do dokumentu komendą:

```
\includegraphics[klucz=wartość,...]{nazwa_pliku}
```

gdzie *nazwa_pliku* to ścieżka do pliku graficznego bez rozszerzenia. Dla wygody możemy jeszcze w preambule wywołać komendę

```
\graphicspath{{katalog1}{katalog2}...}
```

po wywołaniu tej komendy `\includegraphics` będzie szukać pliku o podanej nazwie w katalogach podanych jako argumenty `\graphicspath` i nie będzie konieczne podawanie pełnych ścieżek do plików.

Podając odpowiednie argumenty opcjonalne (pary `klucz=wartość`) komendzie `\includegraphics`, możemy manipulować na różne sposoby elementem graficznym przed umieszczeniem go w dokumencie. Dostępne opcje to

`scale=`*`liczba`* Przeskalowuje wstawiany obrazek o podany czynnik.

`width=`*`szerokość`*

`height=`*`wysokość`* Zmienia szerokość (wysokość) grafiki. Jeśli tylko jeden z wymiarów jest podany, to drugi jest automatycznie skalowany, tak aby stosunek wymiarów został zachowany.

`keepaspectratio=`*`(true|false)`* Jeśli `true` i zarówno wysokość i szerokość grafiki jest zmieniana obydwoma poprzednimi opcjami, to `graphicx` zmniejsza wartość jednego z podanych wymiarów tak, aby oryginalny stosunek wymiarów był zachowany.

`angle=`*`liczba`* Obraca rysunek o `liczba` stopni w kierunku przeciwnym do wskazówek zegara.

`origin=`*`punkt`* Określa punkt względem, którego rysunek ma być obrócony. Dopuszczalne wartości: `c` — środek, `t` — góra, `b` — spód, `B` — linia bazowa, `l` — lewa krawędź, `r` — prawa krawędź, oraz kombinacje. Domyślna wartość to `bl`, czyli lewy dolny róg.

`draft=`*`(true|false)`* Jeśli `true`, to na miejsce obrazka wstawiany jest prostokąt z obramowaniem o takich samych wymiarach co obrazek. *Może być użyte jako opcja pakietu, wtedy stosuje się do każdego obrazka.*

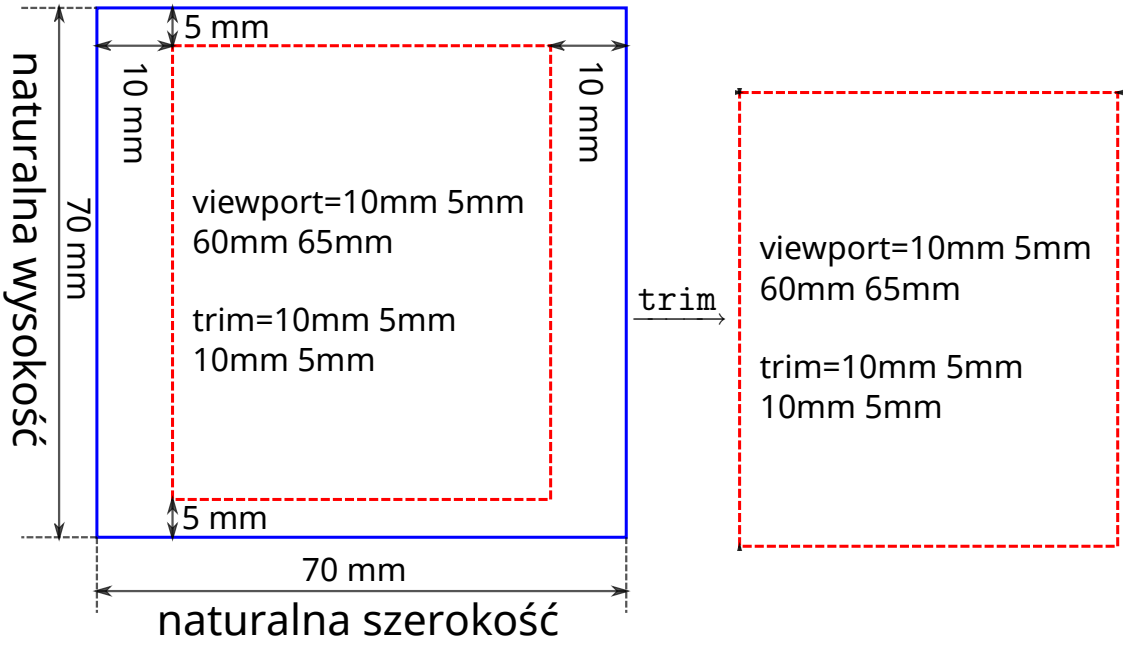
`viewport=`*`llx lly urx ury`* Określa fragment obrazka, który ma być wstawiony do dokumentu poprzez podanie współrzędnych lewego dolnego rogu (`llx lly`) oraz prawego górnego rogu (`urx ury`) prostokąta kadrowania. Patrz Rys. 5.

`trim=`*`dllx dlly durx dury`* Przycina wstawiany obrazek obcinając `dlly` od dołu, `dllx` od lewej, `durx` od prawej i `dury` od góry. Patrz Rys. 5.

`clip=`*`(true|false)`* Określa czy obszar obrazka poza obszarem ograniczonym przez opcję `viewport` albo `trim` ma być wyświetlany. Jeśli `false` to zostanie wyświetlony cały, ale będzie potraktowany tak jakby zajmował tylko obszar wskazany przez wyżej wymienione opcje.

[1] Na zajęciach używamy tego programu.

[2] Dochodzi do automatycznej konwersji z formatu `eps` do `pdf`. W katalogu z plikami `eps` zostaną zapisane pliki z nazwą *`nazwa`*`-eps-converted-to.pdf`.



Rysunek 5: Ilustracja działania opcji `viewport` oraz `trim` komendy `\includegraphics` z pakietu `graphicx`. Obydwie komendy służą do przycięcia (lub zwiększenia marginesów) wstawianego obrazka. Rysunek pokazuje jak z elementu graficznego (niebieski prostokąt) o rozmiarach 70 mm × 70 mm wyciąć fragment (czerwony prostokąt) o rozmiarach 50 mm × 60 mm.

VIII.2. Wstawki

Elementy dokumentu takie jak rysunki czy tabele możemy umieszczać bezpośrednio w tekście odpowiednimi komendami — komenda `\includegraphics` i środowisko `tabular` — wtedy znajdą się one w wygenerowanym PDF-ie dokładnie w tym miejscu, w którym były umieszczone w kodzie źródłowym. Jeśli okaże się, że to miejsce przypada na koniec strony (lub kolumny) i brakuje miejsca na dany element, to zostanie rozpoczęta nowa strona (lub kolumna) i tam znajdzie się ten element, a część poprzedniej strony zostanie nie wykorzystana. W ogólności, decydując samodzielne o rozmieszczeniu tego typu elementów bardzo łatwo uzyskać nieestetyczny efekt końcowy i narazić się na konieczność ponownego rozmieszczana elementów przy zmianie marginesów, rozmiaru czcionki czy jakichkolwiek innych aspektów formatowania dokumentu. Ponadto, zazwyczaj istnieje konieczność umieszczania podpisów do elementów, numerowania ich i odwoływania się do nich w tekście, czego poprawne wykonywanie własnymi siłami wymaga dużo pracy. Rozwiązaniem tego problemu są *wstawki*, czyli elementy pływające (ang. *float*), tj. takie które nie są umieszczane dokładnie w tym miejscu, w których są wprowadzone, lecz w pobliże i zgodnie z wytycznymi LaTeX-a.

Do wprowadzania wstawek używamy środowisk:

```
\begin{figure}[gdzie]   rysunek   \end{figure}
\begin{figure*}[gdzie]  rysunek   \end{figure*}
\begin{table}[gdzie]    rysunek   \end{table}
\begin{table*}[gdzie]   rysunek   \end{table*}
```

gdzie wersje z gwiazdkami są używane w dokumentach dwukolumnowych, aby umieścić tabelę lub rysunek zajmujące obydwie kolumny. Standardowo do umieszczenia rysunku w dokumencie używamy fragmentu kodu:

```
\begin{figure}[ht]
\centering
\includegraphics{ścieżka}
\caption{Opis rysunku...\label{fig:nazwa}}
\end{figure}
...
Na Rysunku~\ref{fig:nazwa} (strona~\pageref{fig:nazwa}) widoczna jest zależność...
```

Komenda `\caption` umieszcza pod rysunkiem jego opis poprzedzony numerem rysunku. Komenda `\label` powoduje zapamiętanie numeru aktualnego rysunku i numeru strony, na którym się znajdzie, dzięki czemu można się potem odwołać do nich komendami `\ref` oraz `\pageref`, odpowiednio.

Komenda `\caption` przyjmuje jeszcze argument opcjonalny, który jest krótszym podpisem umieszczanym w spisie rysunków zamiast podpisu głównego.

O położeniu rysunku decyduje wartość argumentu *`gdzie`* połączone z algorytmem umieszczania wstawek parametryzowanym przez wartości szeregu komend i liczników (podsekcja VIII.3), które zostaną omówione później. Argument *`gdzie`* może przyjmować wartości będące sekwencjami poniższych liter[3]:

`h` *tutaj*: rysunek jest umieszczony w miejscu wystąpienia w kodzie; *po załadowaniu pakietu `here` dostępna jest opcja `H`, która wymusza umieszczenie rysunku dokładnie w miejscu jego wystąpienia w kodzie.*

`t` *u góry*: rysunek jest umieszczony na początku bieżącej strony, o ile jest na niej wystarczająca ilość na tekst przed rysunkiem i na rysunek; jeśli nie, to rysunek jest umieszczany na początku następnej strony.

`b` *na dole*: rysunek jest umieszczony na końcu bieżącej strony, o ile jest na niej wystarczająca ilość na tekst przed rysunkiem i na rysunek; jeśli nie, to rysunek jest umieszczany na końcu następnej strony.

`p` *strona*: rysunek jest umieszczony na osobnej stronie, na której nie ma normalnego tekstu, potencjalnie z innymi rysunkami.

`!` używane razem z innymi literami, zawiesza ograniczenia opisane w podsekcji VIII.3, oprócz `\topfraction` i `\bottomfraction`.

Rysunki umieszczane są tak szybko jak to możliwe, zgodnie z poniższymi zasadami:

- wstawki są umieszczane nie wcześniej niż na stronie, na której są wstawione;
- rysunki i tabele są wstawanie zgodnie z kolejnością występowania w kodzie; tzn. dla każdego typu wstawek (rysunków, tabel i potencjalnie innych) jest zdefiniowana oddzielna kolejka FIFO;
- dopuszczalne położenia na stronie są określone przez argument opcjonalny *`gdzie`*,
- bez argumentu `!` stosowane są dodatkowe ograniczenia opisane w podsekcji VIII.3,
- w kombinacji argumentów `ht`, argument `h` ma priorytet.

Ponadto wywołanie komendy `\clearpage` lub `\cleardoublepage` wymusza umieszczenie wszystkich wstawek w kolejce na następnej i kolejnych stronach.

Ponieważ używane są kolejki FIFO, może się zdarzyć tak, że obrazek zbyt duży aby był umieszczony na stronie razem z tekstem „zepchnie" wszystkie inne wstawki, które były umieszczone po nim na osobną stronę dla wstawek na sam koniec dokumentu. Możemy wtedy tymczasowo zmienić wartości parametrów z następnej sekcji (zaczynają one działać dopiero od następnej strony) albo wykorzystać pakiet `here`.

[3] Kolejność nie ma znaczenia i nie określa która ewentualność jest preferowana!

VIII.3. Parametry rozmieszczania wstawek

Algorytm umieszczania wstawek na stronie jest parametryzowany przez wartości poniższych liczników[4]:

`topnumber` maksymalna ilość wstawek na górze strony,
`bottomnumber` maksymalna ilość wstawek na dole strony,
`totalnumber` całkowita maksymalna ilość wstawek na stronie,
`dbltopnumber` to samo co `topnumber`, tylko dla wstawek dwukolumnowych z gwiazdką.

Jeśli chodzi o umieszczanie wstawek to istotne są jeszcze wartości poniższych komend[5]:

`\topfraction` maksymalny ułamek wysokości strony (`\textheight`) do wykorzystania na górne wstawki,
`\bottomfraction` j.w. tylko, że dla dolnych wstawek,
`\textfraction` minimalny ułamek strony, który musi być wykorzystany na tekst; ma wyższy priorytet niż powyższe parametry,
`\floatpagefraction` minimalny ułamek wysokości strony przeznaczonej tylko na wstawki, który musi być wykorzystany przed przejściem do nowej strony,
`\dbltopfraction`
`\dblfloatpagefraction` to samo co `\topfraction` i `\floatpagefraction`, odpowiednio, tylko że dla wstawek z gwiazdkami.

Ponadto mamy jeszcze długości[6] określające odstępy między wstawkami:

`\floatsep` odstęp pionowy między następującymi bezpośrednio po sobie wstawkami.
`\textfloatsep` odstęp pionowy między wstawką a tekstem.
`\intextsep` odstęp pionowy między wstawką umieszczoną parametrem `h` a tekstem.
`\dblfloatsep`
`\dbltextfloatsep` j.w. tylko że dla wstawek z gwiazdką.
`\abovecaptionskip`
`\belowcaptionskip` odstęp pionowy nad i pod podpisem rysunku lub tabeli.

Dla klasy `article`[7] ich domyślne wartości są zdefiniowane w pliku `article.cls`, którego fragment jest przytoczony poniżej:

```
\setcounter{topnumber}{2}
\renewcommand\topfraction{.7}
\setcounter{bottomnumber}{1}
\renewcommand\bottomfraction{.3}
\setcounter{totalnumber}{3}
\renewcommand\textfraction{.2}
\renewcommand\floatpagefraction{.5}
\setcounter{dbltopnumber}{2}
\renewcommand\dbltopfraction{.7}
\renewcommand\dblfloatpagefraction{.5}
\setlength\abovecaptionskip{10pt}
\setlength\belowcaptionskip{0pt}
```

[4] Zmieniane komendą `\setcounter`.
[5] Zmieniane komendą `\renewcommand`.
[6] Zmieniane komendą `\setlength`.
[7] Tak samo dla klasy `book` oraz `report`.

VIII.4. Odnośniki

Komendy `\label`, `\ref` oraz `\pageref` mają znacznie szersze zastosowanie niż numerowanie wstawek. Gdy wywoływania jest komenda `\label{`*`znacznik`*`}`, sprawdzany jest kontekst jej wywołania, tzn. czy jesteśmy aktualnie w środowisku `figure`, `equation`, jaka była ostatnia wywołana komenda sekcjonowania i wartość odpowiedniego licznika razem z numerem bieżącej strony jest wiązana ze *`znacznikiem`*. Należy pamiętać, że zapisywana jest wartość „najgłębszego" licznika, tj. dla fragmentu kody:

```
\section{Wstawki}
\label{sec:wstawki}
\subsection{Odnośniki}
\label{ssec:odnosniki}
```

do znacznika `sec:wstawki` będzie przypisany numer sekcji, a do znacznika `ssec:odnośniki` — numer podsekcji. Zwyczajowo przyjęło się, że właściwa nazwa znacznika jest poprzedzana skrótem od typu znacznika, tzn. dla sekcji używamy nazwy `sec:`*`nazwa`*, dla rysunków — `fig:`*`nazwa`*, dla równań — `eq:`*`nazwa`*, itd. Nic nie stoi na przeszkodzie żeby komendę `\label` umieszczać w argumencie odpowiedniej komendy powiązanej z licznikiem:

```
\section{Wstawki\label{sec:wstawki}}
\caption{Wykres zależności...\label{fig:I-T-dep}}
```