Łagodne wprowadzenie do TEX-a

Podręcznik

Michael Doob
Department of Mathematics
The University of Manitoba
Winnipeg, Manitoba, Canada R3T 2N2

MDOOB@UOFMCC.BITNET
MDOOB@CCU.UMANITOBA.CA

Tłumaczenie:
Stanisław Wawrykiewicz przy współpracy Barbary Kiełczyńskiej
Redakcja:
Bogusław Jackowski i Stanisław Wawrykiewicz

Wstęp

Na początek przykra wiadomość: TEX jest dużym i skomplikowanym programem, który rozrasta się do wyjątkowych rozmiarów jeśli ma prowadzić do powstania atrakcyjnego składu. Ta szczególna komplikacja prowadzi czasem do niespodzianek. Teraz dobra wiadomość: teksty tzw. gładkie można składać przy pomocy TEX-a w bardzo prosty sposób. Zatem możliwe jest rozpoczynanie od tekstów łatwiejszych i dochodzenie do publikacji bardziej wyrafinowanych pod względem typograficznym.

Naukę TEX-a, prowadzącą do tych bardziej skomplikowanych sytuacji, rozpoczynamy od samego początku. Nie zakładamy przy tym żadnej wcześniejszej znajomości TEX-a. Stopniowe zaznajomienie się z kolejnymi rozdziałami zaowocuje umiejętnością radzenia sobie z coraz różnorodniejszymi tekstami.

Oto kilka rad: w każdym rozdziale znajdują się ćwiczenia. Należy je wykonać! Jedynym sposobem nauczenia się TEX-a jest używanie go. Jeszcze lepiej jest eksperymentować samemu; zachęcamy cię, Czytelniku, do wypróbowywania różnych wariantów ćwiczeń. Zniszczenie TEX-a takim eksperymentowaniem jest niemożliwe. Zauważysz też zapewne odnośniki do podręcznika **TEXbook** na prawym marginesie. Jeżeli będziesz potrzebował więcej informacji na dany temat, możesz tam po nią sięgnąć.

W podręczniku pojawiło się kilka drobnych oszustw, które służą ukryciu pewnych komplikacji (traktuję je jak coś w rodzaju *licentia poetica*). Po dokładniejszym zaznajomieniu się z TEX-em będziesz umiał je odnaleźć.

TEX w wersji źródłowej jest dobrem wspólnym (tzw. programem *public domain*), dostępnym bez żadnych opłat. Powstał w Uniwersytecie Stanforda jako duży projekt autorstwa Donalda Knutha. Na rynku nastawionym na zysk kosztowałby zapewne wiele tysięcy dolarów. TEX Users Group (TUG), jako organizacja niedochodowa, rozprowadza kopie TEX-a, aktualne oprogramowanie oraz udziela informacji o nowych osiągnięciach zarówno w dziedzinie sprzętu jak i oprogramowania w czasopismach TUGboat i TEXniques. Przystąpienie do organizacji kosztuje niewiele[1]; proszę to rozważyć. Oto adres:

TEX Users Group
P.O. Box 9506
Providence, RI 02940
U.S.A.

[1] Niestety, jak na warunki zachodnie: roczna składka wynosi ok. 45$ — przyp. tłum.

Podręcznik ten nie zostałby napisany bez pomocy innych osób. Szczególnie cenne okazały się sugestie: Roberta Messera (Albion College), Anity Hoover (University of Delaware), Johna Lee (Northrop Corporation), Emily H. Moore (Grinnell College). *[...można by tu zamieścić bardziej odpowiedni zestaw nazwisk. Chciałbym, oczekując uwag, zamieścić tu i twoje nazwisko!]*

Ponadto parę osób przysłało mi fragmenty swoich opracowań. Niektóre z nich otrzymałem jako akty zemsty. W szczególności Elizabeth Barnhart (TV Guide), Stephan v. Bechtolsheim (Purdue University), Nelson H. F. Beebe (University of Utah) i Leslie Lamport (Western Digital Corporation), Marie McPartland-Conn i Luarie Mann (Stratus Computer), Robert Messer (Albion College), Noel Peterson (Library of Congress), Craig Platt (University of Manitoba), Alan Spragens (Stanford Linear Accelerator Center), Christina Thiele (Carleton University) i Daniel M. Zirin (California Institute of Technology) przygotowali dla mnie notatki, które okazały się najbardziej przydatne.

Nota tłumacza

Tłumaczenie niniejsze jest pierwszą — nie licząc krótkich artykułów w czasopismach — publikacją na temat TEX-a, ukazującą się w języku polskim. W TEX-u zawarte są wielowiekowe doświadczenia sztuki drukarskiej. Jednocześnie realizacja, zgodna z kanonami tej sztuki, wykorzystuje w pełni najnowsze osiągnięcia informatyki, wręcz tworzy pewną filozofię programowania.

Opis tak złożonego systemu wymaga nie tylko posługiwania się specjalistyczną terminologią z dziedzin dotychczas tak odrębnych, jak typografia i informatyka, ale i tworzenia wielu nowych pojęć. Według Autora oryginału prezentowana publikacja jest szkicem podręcznika do samodzielnej nauki TEX-a, próbą uprzystępnienia niezwykle rozbudowanego i wyrafinowanego sposobu składu tekstów. W ujęciu popularnym opis jest skazany na niedoskonałość. Stąd pewna swoboda w prezentowaniu nowych pojęć. Dokładne ich wyjaśnienie wymagałoby obszernych opisów, utrudniających korzystanie z podręcznika.

W czasie tłumaczenia natknięto się na szereg utrudnień wynikających z braku polskich odpowiedników dla wielu opisywanych tu i utrwalonych już w literaturze anglosaskiej pojęć. Przykładem niech będzie termin *font*. Oznacza on mniej więcej zestaw pisma jednego kroju i stopnia, w pewnym sensie komputerowy odpowiednik drukarskiej kaszty czcionek. Jak czytelnik zauważył, potrzebna jest tu znajomość znaczenia pojęć: krój i stopień pisma, kaszta czcionek. Zaraz, zaraz... Jakich czcionek? Wszak czcionka (ang. *type*) to element metalowy, dający odbitkę na papierze. Z kolei komputerowy *font* to cyfrowy zapis zestawu obrazów znaków. Ale „zestaw obrazów znaków danego kroju pisma" to ciut przydługa nazwa. W tłumaczeniu zastosowano zatem dla uproszczenia nazwy „czcionka" i „zestaw czcionek".

Kolejna sprawa to zamieszczone w podręczniku przykłady i ćwiczenia w języku polskim. Nadal brak jest standardu zapisu polskich znaków diakrytycznych i zapis taki zależy od sprzętu. Autorzy polskiej wersji TEX-a, systemu LEX, przyjęli uniwersalną konwencję zapisu dającą się zastosować na dowolnej instalacji i przy użyciu dowolnego edytora. Polega ona na reprezentacji polskich diakrytyków za pomocą dwóch znaków ASCII: '`/a`' dla 'ą', '`/A`' dla 'Ą' itd. Jest to kwestia wyłącznie implementacyjna, wszak pewne instalacje mogą wyświetlać '`ą`' na ekranie, zaś LEX — odpowiednio to przekodowywać. Aby uniknąć zamieszania związanego z taką czy inną reprezentacją polskich znaków diakrytycznych, a także poprawić czytelność przykładów, w podręczniku zastosowano zapis '`żółw`' zamiast '`/z/o/lw`'.

Czytelnik może być zaskoczony mnogością występujących w podręczniku wyrażeń angielskich. TEX jest w zasadzie językiem programowania komputerów i — tak jak PASCAL czy BASIC — posługuje się zwięzłymi i poręcznymi zwrotami oferowanymi w języku angielskim. Gdy siadamy do komputera, kontakt z tym językiem jest w mniemaniu tłumacza nieunikniony.

Część ćwiczeń pozostawiono w wersji angielskiej. Warto je przetwarzać przy użyciu oryginalnego TEX-a, gdyż zapewni to przenoszenie wyrazów zgodne z zasadami obowiązującymi w języku angielskim. Z kolei LEX zapewni prawidłową obsługę polskich znaków diakrytycznych, przenoszenie zgodne z zasadami języka polskiego, stosując się przy tym do konwencji obowiązujących w naszej typografii. Posługiwanie się dwiema wersjami TEX-a można zatem potraktować jako kolejne ćwiczenie.

S.W.

Spis treści

Rozdział 1
Zaczynamy!

1.1 Czym TEX jest, a czym nie jest

TEX jest programem służącym do elektronicznego składu typograficznego (*typeset output*), szczególnie przydatnym w przypadku tekstów zawierających symbole matematyczne. Uczenie się TEX-a jest trochę podobne do nauki języków obcych. Na początku odbiera się go jako natłok skomplikowanej terminologii, dotyczącej zarówno składu jak i programowania. Ale tak jak w języku obcym, jeśli zaczniemy od prostych wyrażeń i struktur, posiądziemy wiedzę pozwalającą radzić sobie w większości sytuacji, by przy odpowiedniej okazji nauczyć się trudniejszych zwrotów. Najbardziej kompletnym źródłem informacji o TEX-u jest bezsprzecznie **The TEXbook** Donalda E. Knutha[1]. Jest to encyklopedia i słownik dla użytkowników TEX-a i powinien być dostępny każdemu, kto korzysta z TEX-a regularnie. Podobnie jednak, jak prawie niemożliwym byłoby nauczenie się języka angielskiego, gdyby ograniczyć się do czytania podręczników gramatyki i słowników, tak trudno jest nauczyć się TEX-a czytając **The TEXbook**. Celem niniejszego opracowania jest przedstawienie pojęć niezbędnych do tworzenia przy użyciu TEX-a typowych dokumentów i publikacji. Po opanowaniu podstaw, bardziej ambitni użytkownicy dojdą przy pomocy **The TEXbook** do coraz bardziej różnorodnych i oryginalnych rezultatów. Pomocne w tym będą odnośniki umieszczone na prawym marginesie.

W rozdziale pierwszym zapoznamy się ogólnie z tym, co chcemy osiągnąć. Zobaczymy co TEX może, a czego nie może zrobić. W trakcie nauczymy się także kilku użytecznych terminów. Pod koniec będziemy w stanie utworzyć pewien materiał do składu (*typeset material*).

Przede wszystkim spójrzmy jakie kroki należy wykonać w celu otrzymania gotowego dokumentu bądź publikacji. Pierwszym krokiem jest napisanie pliku, który TEX odczyta. Nazywa się on zazwyczaj plikiem danych (*TEX file*) lub zbiorem wejściowym (*input file*) i może być utworzony przy pomocy prostego edytora tekstowego (jeśli używasz wyszukanych procesorów tekstu, musisz być pewien, że twój zbiór zostanie zapamiętany w trybie ASCII — bez specjalnych znaków kontrolnych). Program TEX przeczyta twój zbiór wejściowy i utworzy zbiór DVI (*DVI file*; DVI jest skrótem *DeVice Independent* co oznacza „niezależny od sprzętu"). Zbiór ten jest nie do odczytania, przynajmniej nie dla ludzi. Zbiór DVI jest następnie przetwarzany przez inny program, zwany programem obsługi urządzenia

[1] Addison-Wesley, Reading, Massachusetts, 1986, ISBN 0-201-13447-0.

wyjściowego, krócej sterownikiem (*device driver*), który tworzy czytelny — po przesłaniu na odpowiednie urządzenie — wynik. Po co zatem dodatkowy zbiór? Otóż ten sam zbiór DVI może być czytany przez różne sterowniki tworzące pliki wyjściowe na drukarkę igłową, drukarkę laserową lub fotonaświetlarkę bądź bezpośrednio wyświetlane na ekranie. Jeśli został utworzony zbiór DVI, który daje poprawny wynik na ekranie, możesz być pewien, że uzyskasz dokładnie taki sam wynik na drukarce laserowej, bez konieczności ponownego uruchamiania TEX-a. TEXbook: 23

Omówiony wyżej proces można przedstawić następująco:

Oznacza to, że nie uzyskamy zbioru wyjściowego w jego ostatecznej formie jeśli będziemy wprowadzać plik danych z terminala. Odrobina cierpliwości zostanie odpowiednio wynagrodzona, gdyż znaczna liczba symboli, niedostępnych większości programów przetwarzających teksty, staje się dostępna. Ponadto skład wykonany jest z większą precyzją, a zbiory wejściowe mogą być łatwo przesyłane między komputerami za pomocą poczty elektronicznej lub nośników magnetycznych.

Skoncentrujmy się na pierwszym kroku, czyli na tworzeniu pliku danych i uruchamianiu programu TEX. Są dwa sposoby uruchomienia programu TEX: w trybie wsadowym (*batch mode*) lub konwersacyjnym. W trybie wsadowym wprowadzasz program do komputera i otrzymasz rezultat, kiedy program zostanie wykonany. W trybie konwersacyjnym użytkownik może ingerować w trakcie przetwarzania — program może być zatrzymany w celu wprowadzenia nowych danych. Bezpośredni dostęp do TEX-a daje możliwość poprawiania błędów przez użytkownika; w trybie wsadowym błędy poprawia sam TEX — najlepiej jak potrafi. Implementacje na wszystkie komputery osobiste i wiele dużych systemów komputerowych pozwalają na konwersację. Jednak niektóre duże systemy komputerowe mogą pracować tylko w trybie wsadowym.

1.2 Od pliku TEX-owego do wydruku: jak wygląda sesja

[Nota od tłumacza: *Podrozdział ten jako jedyny mówi o sprawach związanych z konkretną instalacją i zawiera w oryginale opis pracy z TEX-em na Uniwersytecie Manitoba w Kanadzie (komputer Amdahl, system operacyjny MVS, edytor Mantes, oraz podobny, nie spotykany w Polsce sprzęt i oprogramowanie). Zgodnie z sugestiami autora został w niniejszym tłumaczeniu całkowicie zmieniony i opisuje przykład z naszego podwórka, a więc sesję na komputerze zgodnym z IBM PC i systemem DOS.*]

Zakładam, że umiesz posługiwać się którymś z edytorów tekstowych. Istotne jest aby zapis pliku na dysku nie zawierał kodów sterujących. W przypadku użycia edytora typu

Chi-Writer[2] bądź *TAG*[3] plik powinien zostać przefiltrowany przez odpowiedni program i sprowadzony do postaci czystego pliku ASCII. Zestaw *TAG* zawiera program o nazwie `tag2ascii.exe` wykonujący taką konwersję w oparciu o definiowaną przez użytkownika tablicę zamiany kodów. Brak tu niestety miejsca na omówienie kwestii polskich liter i zainteresowanych odsyłam do instrukcji obsługi polskiej wersji TEX-a, czyli systemu LEX.

Mam nadzieję, że wszystkie programy masz zapisane na twardym dysku i system jest, jak się to mówi, prawidłowo „skonfigurowany". W razie czego skontaktuj się natychmiast z TEX-magiem.

Ponieważ w czasie pracy wykonujemy wielokrotnie te same czynności, programy często uruchamiane jak: `tex.exe`, programy obsługi ekranu czy drukarki posiadają szereg parametrów, należy utworzyć tzw. pliki wsadowe o rozszerzeniu nazwy `.bat`

(1) Utwórz plik `t.bat` służący do wywołania programu składającego (np. *pcTEX* firmy Personal TeX Inc.):

```
@ echo off
c:\tex\TEX &c:\tex\texfmts\plain %1 /M=65534 /F=51199
```

(2) Utwórz plik `v.bat` uruchamiający program do podglądu składu na ekranie monitora (program *TEXview* firmy Polski TEX):

```
@echo off
c:\tex\uti\texview %1 Tt:\tfms\ Ft:\pixel300\dpi Q1000 R300 H29.7cm
    W21cm S3 %2 %3 %4 %5 %6 %7 %8 %9
```

Jak widać wywołanie pliku `texview.exe` wymaga podania wielu parametrów. Zgodnie z regułami DOSu muszą się one znajdować w jednym wierszu (tu i w następnym przykładzie okazało się to niemożliwe z powodu szerokości szpalty).

(3) Utwórz plik `p.bat` uruchamiający program tłumaczący wynik składu na język drukarki (*DVIjep* Nelsona F. Beebe dla drukarki Hewlett-Packard LaserJet series II):

```
@ echo off
c:\tex\uti\dvijep -b -d24 -eTEXFONTS=T:\PIXEL300\DPI -eDVIHELP=NONE %2 %3
  %4 %5 %6 %7 %8 %9 %1
```

Przechodzimy do składu tekstu zapisanego w pliku o nazwie np. `dokument.tex` — będzie to nasz plik wejściowy. Rozszerzenie `.tex` jest czysto umowne, służy jedynie do wy-

[2] firmy Horstman Software Design.
[3] firmy InfoService z Gdańska.

różnienia pliku zawierającego tekst i specjalne komendy sterujące jego składem. Przyjmijmy, że nasz tekst zawiera już takie komendy. Teraz go przetwarzamy, wpisując z klawiatury:

```
t dokument
```

W wyniku przetwarzania otrzymasz zwykle dwa pliki: `dokument.dvi` — zawierający zapis składu i `dokument.log` z zapisem komunikatów pracy TEX-a.

Efekt pracy TEX-a możesz obejrzeć na ekranie, wpisując z klawiatury polecenie:

```
v dokument
```

Wreszcie plik `dokument.dvi` można przetworzyć na zapis binarny „zrozumiały" dla danej drukarki:

```
p dokument
```

po czym przesłać wynik do drukarki za pomocą DOS-owego polecenia:

```
copy dokument.jep > prn /b
```

Plik `dokument.log` oglądamy za pomocą edytora bądź prostego programu przeglądającego, np. opcja `F3 view` popularnego *Norton Commander*. W pliku znajdziesz zapis komunikatów TEX-a: nazwy plików przetwarzanych (może ich być dowolnie wiele), ilość stron składu, miejsca wystąpienia ewentualnych błędów i szereg informacji przydatnych do analizy procesu składu.

Powyższy opis wyda się być może schematyczny, zaś praca z TEX-em skomplikowana. Chciałbym jedynie zaznaczyć, że to skomplikowanie wynika z szeregu ograniczeń systemu DOS, takich jak brak możliwości jednoczesnego podglądu wyniku składu w czasie pisania tekstu wejściowego, lub jednoczesnego składu i drukowania. Czynności te są rozdzielone zgodnie ze schematem: edycja pliku wejściowego → skład → podgląd → ewentualna ponowna edycja i korekta → przetwarzanie pliku `.dvi` dla konkretnego urządzenia drukującego → drukowanie. Poza tym podręcznik ten traktuje o języku TEX-a i sposobach w jakich chcesz opisać precyzyjnie postać publikacji. W tym kontekście nieistotne jest na jakim sprzęcie zostanie to zrealizowane. Opis składu w TEX-u jest identyczny tak na maszynie typu IBM PC, jak i superkomputerze CRAY.

1.3 Zróbmy to!

Zatem, z naszego punktu widzenia, całość postępowania sprowadza się do utworzenia zbioru wejściowego, który da prawidłowy, czytelny wynik. Jak wygląda zbiór wejściowy? Składa się z podstawowych znaków, czyli dużych i małych liter, liczb, znaków interpunkcyjnych, znaków akcentu (zwykłych znaków ASCII). Tekst, w większości, jest pisany normalnie.

Instrukcje specjalne zazwyczaj rozpoczynają się jednym z takich symboli, jak `#` lub `&` (dalej zostanie to opisane bardziej szczegółowo). A oto przykład zbioru wejściowego:

```
\TeX\ lubi płatać figle.
\bye
```

Po pierwsze, znaki w tym przykładzie wyglądają jak w tekście maszynowym. Używamy ich we wszystkich przykładach prezentujących pliki danych wprowadzanych do komputera. Po drugie, w-tył-ciach (znak '\', ang. *backslash*) pojawia się w tekście trzykrotnie. Wkrótce przekonamy się, że jest to jeden ze znaków specjalnych, o których wspomnieliśmy wcześniej i używany jest bardzo często podczas pisania plików danych. Proponujemy ci, Czytelniku, utworzenie pliku o nazwie `zdanie.tex` zawierającego ten przykład. Użyj programu TEX w celu utworzenia pliku DVI, a następnie sterownika w celu obejrzenia rezultatów. Jeśli wszystko pójdzie dobrze, otrzymasz na pojedynczej stronie następujące zdanie:

TEX lubi płatać figle.

Na dole strony pojawi się także jej numer. Jeśli to uzyskałeś — gratulujemy! Przejście do bardziej skomplikowanych przypadków jest tylko kwestią czasu. Teraz porównajmy to, co napisaliśmy z tym , co uzyskaliśmy. Podstawowe słowa zostały napisane normalnie i TEX przełożył je na takie same znaki. Ale wyraz „TEX", który nie może być wprowadzony standardowo, gdyż jego litery nie leżą w jednej linii, jest wprowadzany za pomocą symbolu rozpoczynającego się w-tył-ciachem, co nadaje temu słowu odpowiedni wygląd. Większość symboli, które nie są zwykłymi literami, cyframi, czy znakami interpunkcji jest wprowadzana za pomocą słowa rozpoczynającego się w-tył-ciachem. Jeśli przyjrzymy się temu uważniej, zauważymy także, że zmieniło się słowo „figle". Pierwsze dwie litery połączyły się ze sobą i zniknęła osobna kropka nad literą „i". Jest to standardowa praktyka składu: pewne zestawy liter łączy się ze sobą w formę zwaną *ligaturą*. Ma to swoje uzasadnienie estetyczne, wystarczy porównać dwie pierwsze litery w słowach „figle" i „figle". Zauważmy także, iż `\bye` na końcu pliku nie ma swojego odpowiednika w składzie. Jest to instrukcja programowa, informująca o końcu pliku wejściowego. W trakcie tych rozważań poznamy jeszcze wiele innych instrukcji programowych.

TEXbook: 4

Teraz spójrzmy na plik informacyjny (*log file*) powstający w trakcie uruchamiania TEX-a. Powinien on wyglądać z grubsza następująco:

```
This is TeX, Version 2.98 (no format preloaded)
** File zdanie.tex
(zdanie.tex [1]
Output written on ZDANIE.DVI (1 page, 512 bytes).
Transcript written on ZDANIE.LOG.
```

Jest to zbiór, który będzie zawierał wszelkie komunikaty i informacje o błędach. Informacja `(zdanie.tex)` w wierszu 3 mówi, że program TEX rozpoczął czytanie zbioru. Ukazanie się `[1]` wskazuje, że został zakończony skład strony 1. Jeśli na stronie 1 znajdą się jakiekolwiek błędy, zostaną od razu wymienione.

▷ Ćwiczenie 1.1 Dodaj do swojego pliku danych kolejny wiersz:

```
\TeX\ lubi płatać figle.
Owszem, to prawda!
\bye
```

Zobacz co uzyskałeś. Czy następne zdanie znalazło się w nowym wierszu?

▷ Ćwiczenie 1.2 A teraz dodaj na początku swojego pliku komendę:

```
\nopagenumbers
```

Zgadnij co się stanie, gdy teraz uruchomisz TEX-a?

▷ Ćwiczenie 1.3 Dodaj jeszcze trzy lub cztery zdania do swojego pliku. Użyj liter, cyfr, kropek, przecinków, znaków zapytania i wykrzykników, nie używaj jednak innych symboli.

▷ Ćwiczenie 1.4 Zostaw wolny wiersz i dodaj jeszcze parę zdań. W ten sposób otrzymasz nowy akapit.

Zapoznaliśmy się z główną zasadą dotyczącą przygotowywania plików danych. Rozmieszczenie tekstu w zbiorze wejściowym niekoniecznie musi odpowiadać jego rozmieszczeniu w zbiorze wyjściowym. Nie otrzyma się na przykład dodatkowych odstępów między słowami przez ich dodanie w pliku danych. Kilka następujących po sobie odstępów da taki sam wynik w składzie jak jeden odstęp. Jak można się domyślić, słowo na końcu wiersza zostanie oddzielone od pierwszego słowa w nowym wierszu. Jeśli pracujemy na bardzo dużych zbiorach, czasem wygodniej jest zaczynać każde zdanie od nowego wiersza. Odstępy na początku wiersza są zawsze ignorowane.

▷ Ćwiczenie 1.5 Dodaj do swojego pliku następujące zdanie w formie akapitu i sporządź jego skład:

```
Gratulacje! Odpowiedzi na 100% podchwytliwych pytań
egzaminacyjnych to spory sukces.
```

Znak `%` umożliwia pisanie komentarzy w plikach wejściowych. Następujące po nim znaki do końca wiersza są ignorowane. Zauważ, że zniknął nawet odstęp między ostatnim słowem przed znakiem % i pierwszym w wierszu następnym.

▷ Ćwiczenie 1.6 Dodaj następujące zdanie jako nowy akapit:

```
Adam jest mi winien 10$!
```

W pliku informacyjnym ukaże się informacja o błędzie (jeśli twoja implemantacja TEX-a wyświetla komunikat na ekranie i czeka na odpowiedź, należy nacisnąć klawisz RETURN lub ENTER). Przyjrzyj się temu plikowi i miejscu, w którym pojawiają się błędy, ale nie przejmuj się aktualnymi błędami. Powiemy o nich (także i o tym) później. Teraz zlikwiduj błąd stawiając w-tył-ciacha przed znakiem `$` i sporządź skład.

1.4 Wszystko w TEX-u jest pod kontrolą

Zauważyliśmy, że w-tył-ciach (znak '\') spełnia specjalną rolę. Każde słowo rozpoczynające się w-tył-ciachem jest interpretowane przez TEX-a w specjalny sposób. Słowo to nosi nazwę *ciągu sterującego* (*control sequence*). Istnieją dwa rodzaje takich ciągów: *słowa sterujące* (*control word*), kiedy po znaku '\' następuje litera (np. `\TeX`) i *symbole sterujące* (*control symbol*), gdy w-tył-ciach poprzedza znak nie będący literą (np. `\$`)[4]. Ponieważ odstęp nie jest literą, w-tył-ciach z odstępem jest traktowany jako symbol sterujący. Jeśli chcemy podkreślić istnienie odstępu w naszych przykładach, używamy w tej publikacji znaczka ␣, np. \␣. TEXbook: 7–8

Jeśli TEX czytając twój plik danych natrafi na w-tył-ciacha i następującą bezpośrednio po nim literę, wie, że rozpoczyna czytanie komendy. Komenda jest czytana, dopóki nie pojawi się znak nie będący literą. Zatem, jeśli twój zbiór zawiera

```
I like \TeX!
```

komenda `\TeX` jest ograniczona wykrzyknikiem. Problem pojawia się, jeśli chcesz uzyskać odstęp po komendzie. Jeżeli w swoim pliku wejściowym masz, na przykład, zdanie

```
I like \TeX and use it all the time.
```

[4] Słowa i symbole sterujące są dla TEX-a komendami; dalej będziemy używać tego krótszego terminu — przyp. tłum.

komenda `\TeX` jest ograniczona odstępem, który oczywiście nie jest literą. Wówczas nie otrzyma się odstępu między słowami „TEX" i „and"; dodawanie dodatkowych odstępów niczego nie zmieni, gdyż TEX nie widzi różnicy między pojedynczym odstępem i kilkoma odstępami. Używając natomiast symbolu sterującego \␣ jednocześnie ograniczamy komendę i wstawiamy odstęp. Jest pewne, że spotkasz się z tym nieraz podczas korzystania z TEX-a.

▷ Ćwiczenie 1.7 Sporządź plik, który utworzy następujący akapit: [5]

I like TEX! Once you get the hang of it, TEX is really easy to use. You just have to master the TEXnical aspects.

Większość komend nazwana jest tak, by nie było problemów z odczytaniem ich znaczenia. Na przykład, tworząc nowy akapit, zamiast wolnego wiersza możemy wprowadzić słowo kontrolne `\par` (od *paragraph* — akapit).

1.5 Czego TEX nie zrobi

TEX jest znakomity jeśli chodzi o skład, ale są rzeczy, z którymi słabo sobie radzi. Jedną z nich jest wykonywanie ilustracji. Można zostawić miejsce na ilustracje, natomiast sam język, jak dotąd, nie posiada żadnych procedur graficznych. Niektóre implementacje dopuszczają wprowadzanie instrukcji graficznych przez użycie komendy `\special`, ale należą one do wyjątków.

TEX układa czcionki w poziomie. Tekst ułożony wzdłuż krzywej lub w perspektywie (stopniowo rosnący lub malejący) nie sprawdza się zbyt dobrze.

Powiedzieliśmy o istnieniu cyklu „edytor, TEX, sterownik", koniecznego w przypadku każdego zbioru wyjściowego. Dotyczy to też sytuacji, gdy urządzeniem wyjściowym jest ekran monitora; niemożliwe jest wprowadzanie do pliku danych i natychmiastowe uzyskanie rezultatów na ekranie, jeśli pominiemy cały cykl. Nieliczne implementacje posiadają możliwość wyświetlania zarówno tekstu jak i efektu składu niemal jednocześnie, po dosyć szybkim przejściu cyklu; poprawa będzie następowała w miarę spadku cen sprzętu i wzrostu prędkości procesorów.

[5] W tłumaczeniu zachowano świadomie wersję angielską wielu ćwiczeń — przyp. tłum.

Rozdział 2
Wszelkie znaki, duże i małe

2.1 Niektóre znaki mają szczególne znaczenie

W poprzednim rozdziale zauważyliśmy, że większość tekstów jest wprowadzana do komputera podobnie jak na maszynie do pisania. Zauważyliśmy także, że w-tył-ciach może być użyty przynajmniej do dwóch celów. Może być użyty do składania znaków nie istniejących na klawiaturze, na przykład wpisujemy `\TeX` aby uzyskać TEX. Może być także używany w przypadku instrukcji specjalnych, na przykład `\bye` informującej o końcu pliku wejściowego. Generalnie, słowa rozpoczynające się w-tył-ciachem będą interpretowane przez TEX-a jako wymagające specjalnej uwagi. TEX zna kilkaset takich słów i możesz zdefiniować sobie jeszcze więcej własnych — zatem w-tył-ciach jest bardzo ważny. Spędzimy wiele czasu na nauce niektórych z tych słów, choć na szczęście w większości przypadków używamy niewielu z nich.

Jest dziesięć znaków, które — podobnie jak w-tył-ciach — używane są w TEX-u do specjalnych celów i niżej podamy ich kompletną listę. Co się będzie działo, jeśli zechcesz użyć w-tył-ciacha w którymś z twoich zdań? Nasuwa to następne pytania: TEXbook: 37–38

(1) Które ze znaków są „specjalne”?
(2) W jaki sposób składamy znak specjalny jeśli chcemy go wydrukować?

Oto tablica znaków specjalnych, ich znaczenia w TEX-u i sposób składania samego znaku jeśli zachodzi potrzeba jego umieszczenia w składanym tekście:

Znaki wymagające specjalnego wprowadzania

Znak	Znaczenie	Jak go uzyskać	Efekt
`\`	Symbole specjalne i instrukcje	`$\backslash$`	$\backslash$
`{`	Otwarcie grupy	`$\{$`	$\{$
`}`	Zamknięcie grupy	`$\}$`	$\}$
`%`	Komentarz	`\%`	%
`&`	Ustawianie tablic i znaków tabulacji	`\&`	&
`~`	Odstęp, na którym nie wolno łamać wiersza	`\~{}`	~
`$`	Początek lub koniec trybu matematycznego	`\$`	$
`^`	Górne indeksy matematyczne	`\^{}`	^
`_`	Dolne indeksy matematyczne	`\_{}`	_
`#`	Definiowanie symboli zastępujących tekst	`\#`	#

2.2 Skład z akcentami

Zacznijmy zatem korzystać z TEX-owych możliwości! Do tej pory używaliśmy TEX-a w celu poprawienia atrakcyjności składu, teraz zabierzemy się za rzeczy, które są trudne lub niemożliwe do uzyskania, jeśli korzystamy z maszyny do pisania. Przyjrzymy się teraz dokładniej akcentom. Jak uzyskać literę z akcentem, jeśli nie ma jej na klawiaturze? Podobnie jak w przypadku symbolu TEX, konieczne jest użycie w-tył-ciacha. Zacznijmy od słowa „première" — aby je uzyskać należy w pliku danych umieścić `premi\‘ere` (możesz się trochę naszukać „w-tył-prima" ‘‘ na klawiaturze, ale on tam na pewno jest). Generalnie, aby uzyskać akcent nad literą musimy tę literę poprzedzić odpowiednim ciągiem sterującym. Oto kilka przykładów:

Zbiór wejściowy	Zbiór wyjściowy
`\‘a la mode`	à la mode
`r\’esum\’e`	résumé
`soup\c␣con`	soupçon
`No\"el`	Noël
`na\"\i␣ve`	naïve

Powyższe przykłady obrazują kilka zasad. Większość akcentów można otrzymać używając symbolu sterującego z podobnym znakiem. Niektóre uzyskuje się stosując komendy zawierające pojedynczą literę. W tym przypadku konieczna jest uwaga, gdyż każda komenda wymaga kończącego separatora (najczęściej spacji). Jeżeli umieściłeś w swoim pliku słowo `soup\ccon`, TEX uzna `\ccon` za komendę i zaprotestuje komunikatem na ekranie i w pliku LOG. Istnieje także komenda `\i` dająca literę „i" bez kropki, co pozwala na umieszczenie nad nią dowolnego akcentu. Tak samo jest w przypadku komendy `\j`.

TEXbook: 52–53

Akcenty nie wymagające odstępu

Nazwa	Zbiór wejściowy	Zbiór wyjściowy
grave	`\`o`	ò
acute	`\´o`	ó
circumflex	`\^o`	ô
umlaut/dieresis/trémat	`\"o`	ö
tilde	`\~o`	õ
macron	`\=o`	-o
kropka	`\.o`	ȯ

Akcenty wymagające odstępu

Nazwa	Zbiór wejściowy	Zbiór wyjściowy
cedilla	`\c o`	o̧
underdot	`\d o`	ọ
underbar	`\b o`	o̲
háček	`\v o`	ǒ
breve	`\u o`	ŏ
tie-after	`\t {oo}`	o͡o
węgierski umlaut	`\H o`	ő

TEX pozwala także na korzystanie z liter, których nie ma w języku angielskim:

Symbole z innych języków

Przykład	Zbiór wejściowy	Zbiór wyjściowy
Ægean, æsthetics	`\AE, \ae`	Æ, æ
Œuvres, hors d'œuvre	`\OE \oe`	Œ, œ
Ångstrom	`\AA, \aa`	Å, å
Øre, København	`\O, \o`	Ø, ø
Łódź, łódka	`\L, \l`	Ł, ł
Nuß	`\ss`	ß
	`!``	¡
	`?``	¿
	`{\it\$}`	*£*

Wykonaj skład następujących zdań:

▷ Ćwiczenie 2.1 Does Æschylus understand Œdipus?

▷ Ćwiczenie 2.2 The smallest internal unit of TEX is about 53.63Å.

▷ Ćwiczenie 2.3 Plus ça change, plus ça la même chose.

▷ Ćwiczenie 2.4 Élèves, réfusez vos leçons! Jetez vos chaînes!

▷ Ćwiczenie 2.5 Zašto tako polako pijete čaj?

▷ Ćwiczenie 2.6 Mein Tee ist heiß.

▷ Ćwiczenie 2.7 Maar die ÿs is lekker.

▷ Ćwiczenie 2.8 Peut-être il préfère le café glacé.

▷ Ćwiczenie 2.9 Can you take a ferry from Öland to Åland?

2.3 Kropki, pauzy, cudzysłowy, . . .

Pisanie na maszynie jest zawsze formą kompromisu; niewielka liczba klawiszy w porównaniu z ilością symboli wymusza na piszącym stosowanie pewnych zmian w tekście. Pisząc tekst dla TEX-a mamy dużo większe możliwości. W tym rozdziale przyjrzymy się bliżej pewnym różnicom między pisaniem na maszynie a używaniem TEX-a.

W typografii wyróżnia się cztery rodzaje kresek poziomych. Są to: dywiz, półpauza, pauza i znak minus. Oto ich reprezentacja TEX-owa wraz z przykładami użycia: TEXbook: 3–5

Dywiz, półpauza, pauza, minus

Nazwa	Zbiór wejściowy	Zbiór wyjściowy	Przykład
dywiz	`-`	-	biało-czerwona
półpauza	`--`	–	w latach 1980–1981
pauza	`---`	—	nie jest najgorzej — co by nie gadać
znak minus	`$-$`	−	temperatura spadła do −30°C

▷ Ćwiczenie 2.1 Wszedłem na salę i — o zgrozo — zobaczyłem na scenie Niebiesko-Czarnych.

▷ Ćwiczenie 2.2 Lata 1980–1981 były powodem szczególnej troski władz PRL.

Następna różnica między maszyną do pisania i TEX-em pojawia się przy używaniu cudzysłowów. W języku angielskim maszyna do pisania otwiera i zamyka cudzysłów takim samym znakiem. W TEX-u cudzysłów tworzy się przy użyciu apostrofu lub znaku prim ’ oraz znaku w-tył-prim ‘. Cudzysłów otwierany jest przez ‘‘ i zamykany przez ’’. Podobnie TEXbook: 3

używane są pojedyncze znaki cudzysłowu: otwierający ‘ i zamykający ’. Jak widać nie ma potrzeby korzystania z typowego cudzysłowu " (zwykle w druku wygląda on jak cudzysłów zamykający, choć nie można być tego do końca pewnym)[1]

▷ Ćwiczenie 2.3 Frank wondered, “Is this a girl that can’t say ‘No!’?”

▷ Ćwiczenie 2.4 Piszę: „Pan Maciek powiedział «Każdy gram na wagę złotego»”.

▷ Ćwiczenie 2.5 Szóste: „Nie cudzysłów”

Jeśli chcemy korzystać z wielokropka wskazującego na kontynuację myśli lub urwaną informację, musimy pamiętać, że kropki wprowadzone w pliku danych jedna po drugiej znajdą się w składzie zbyt blisko siebie. Wielokropek z odpowiednimi odległościami otrzymuje się przy użyciu komendy `\dots`. TEXbook: 173

▷ Ćwiczenie 2.6 Pomyślał — ... i tak już będzie do końca, zapewne do ostatniego zapisanego słowa.

Następny problem z kropką związany jest z tym, że odstęp po kropce w skrótach powinien być mniejszy niż po końcu zdania. Są dwa sposoby korygowania takich odstępów: używając po kropce znaku `\␣` lub `~`. Drugie rozwiązanie daje odstęp, na którym nie wolno łamać wiersza, np. `Prof.~Knuth`. Jest to istotne w przypadku nazw własnych, jak Mr. Jones lub Vancouver, B. C. i wynika z przyjęcia pewnych konwencji estetycznych. Zauważ, że nie użyliśmy tutaj znaku ‘\’. TEXbook: 91–92

▷ Ćwiczenie 2.7 Have you seen Ms. Jones?

▷ Ćwiczenie 2.8 Prof. Smith and Dr. Gold flew from Halifax, N. S. to Montréal, P. Q. via Moncton, N. B.

[1] W języku polskim cudzysłowy wprowadza się przy pomocy dwóch przecinków `,,` oraz dwóch primów `''`. W cudzysłowach wewnętrznych, tzw. francuskich (« »), używa się dwóch znaków mniejszości `<<` przy otwieraniu i dwóch znaków większości `>>` przy zamykaniu cydzysłowu — przyp. tłum.

2.4 Różne kroje pisma

Różnica między pismem maszynowym i składem TEX-a jest najbardziej widoczna w rozmaitości krojów pisma i bogactwie dostępnych symboli. W TEX-u dostępnych jest od razu szesnaście różnych rodzajów pisma. Ich kompletna lista podana jest w Dodatku F podręcznika TEXbook. Większość z nich jest używana automatycznie; na przykład indeksy matematyczne są składane mniejszą czcionką, tzw. frakcją, bez ingerencji użytkownika.

TEXbook: 427–432

Zmiana zwykłej czcionki — antykwy (*roman type*) — na kursywę wymaga użycia komendy `\it`. Do antykwy wraca się przez użycie `\rm`. Na przykład zdanie: `Rozpocząłem antykwą, \it zmieniłem ją na kursywę\rm, po czym wróciłem do antykwy`, będzie wyglądało następująco: Rozpocząłem antykwą, *zmieniłem ją na kursywę*, po czym wróciłem do antykwy. Oto najczęściej używane kroje pisma:

Wzory czcionek

Nazwa	Symbol	Przykład
Antykwa (Roman)	`\rm`	This is roman type.
Półgrube (Boldface)	`\bf`	**This is boldface type.**
Kursywa (Italic)	`\it`	*This is italic type.*
Pochyłe (Slanted)	`\sl`	*This is slanted type.*
Maszynowe (Typewriter)	`\tt`	`This is typewriter type.`
Symbole matematyczne (Math symbol)[2]	`\cal`	$\mathcal{SOME}$ $\mathcal{SC}$.

Pismo pochyłe i kursywa na pierwszy rzut oka robią wrażenie bardzo podobnych. Różnicę między nimi najłatwiej zauważyć na przykładzie litery „a". Przy zmianie kursywy lub czcionki pochyłej na antykwę ostatnia litera pierwszego kroju „kładzie się" na pierwszą literę antykwy; robi to wrażenie ścisku i przydałoby się tu nieco więcej przestrzeni, którą można dodać używając tzw. *korekty kursywy* (*italic correction*). Służy do tego symbol `\/`. Zwróć uwagę na różnice w następującym zdaniu: *Jeśli* nie stosujemy korekty kursywy odstęp między literami jest zbyt mały, *jeśli* korektę stosujemy, odstęp jest lepszy.

W TEX-u możliwe jest definiowanie krojów i wielkości (ściślej: stopni) pisma, o ile są one oczywiście dostępne w twoim systemie komputerowym. Definiowanie wymaga znajomości *nazwy zewnętrznej* kroju w przechowywanej na dysku „bibliotece czcionek". Definicja ma postać: `\font\twojanazwa = nazwazbiblioteki` i polega na przypisaniu *nazwy zewnętrznej* nazwie wybranej przez użytkownika. Powiększenie tego samego kroju i stopnia uzyskuje się dzięki komendzie `\magstep`, którą poprzedza słowo specjalne `scaled`. Na przykład, w większości systemów, antykwa ma nazwę „`cmr10`[3]". Jeśli napiszesz definicję po-

[2] Przykład to trochę niebezpieczny, bo dobrze jest wiedzieć nieco więcej o wprowadzaniu symboli i wyrażeń matematycznych. Umieściliśmy je tu jednak dla porządku.

[3] w polskiej wersji TEX-a „p1r10" — przyp. tłum.

staci `\font\bigrm = cmr10 scaled \magstep 1`, możesz używać `\bigrm` dokładnie tak, jak `\it` lub `\rm`.Wstawienie `\bigrm` spowoduje wzrost antykwy o ok. 20%. Natomiast `\font\bigbigrm = p1r10 scaled \magstep 2` definiuje czcionkę o ok. 44% większą od normalnej. Dostępne są wielkości od `\magstep 0` do `\magstep 5`. W większości implementacji dostępny jest także `\magstephalf`; powoduje on powiększenie o ok. 9.5%.

TEXbook: 13–17

A oto przykłady różnych powiększeń czcionki

Oto próbka (magstep 0) — wielkość typowa.
Oto próbka (magstep 1).
Oto próbka (magstep 2).
Oto próbka (magstep 3).
Oto próbka (magstep 4).
Oto próbka (magstep 5).

TEX nie posiada żadnych ograniczeń definiowania dowolnych krojów pisma. Zależy to oczywiście od zasobności dostępnej „biblioteki czcionek”. Wiele systemów posiada krój bezszeryfowy (*sans serif font*) `cmss10`[4]. Podanie definicji `\font\sf = cmss10` pozwala na używanie komendy `\sf` tak, jak `\bf`. Po zapisie takiej definicji, dane wejściowe

```
\sf Oto przykład czcionki bezszeryfowej.
```

w druku będą wyglądać tak:

Oto przykład czcionki bezszeryfowej.

[4] W polskim TEX-u `p1ss10` — przyp. tłum.

Rozdział 3
Rzeczy nabierają kształtu

W tej części zobaczymy co zrobić, aby otrzymać różną postać i wielkość tekstu. Możemy oczywiście używać standardowych TEX-owych wielkości, tak jak robiliśmy to do tej pory, spróbujmy jednak podejść do naszej pracy bardziej twórczo. Projektując wielkość poszczególnych części strony tekstu możemy korzystać z kilku jednostek miar.

3.1 Jednostki, jednostki, jednostki

TEX mierzy wielkości używając wielu jednostek. Najpopularniejsze to: cal, centymetr, punkt typograficzny i pica (pajka) — o skrótach, odpowiednio, `in`, `cm`, `pt` i `pc`. Punkt jest definiowany następująco: 1 cal = 72.27 punktów, zaś pica: 1 pica = 12 punktów[1]. Punkt ma w przybliżeniu wielkość kropki. TEXbook: 57

1 cal:
1 centymetr:
20 punktów:
1 pica:

Zatem, punkty są używane do minimalnych poprawek, pica odpowiada w przybliżeniu odległości między dwoma wierszami typowego tekstu. TEX jest bardzo dokładny jeśli chodzi o miary; jego najmniejsza jednostka jest mniejsza od czterech milionowych cala. To, w jaki sposób będzie wyglądał efekt składu, zależy od rozdzielczości urządzenia wyjściowego.

Istnieją jeszcze dwie inne jednostki, które mogą być użyteczne. Wielkość ich zależy od aktualnej czcionki: `ex` ma w przybliżeniu wysokość litery „x”, `em` (firet) jest trochę mniejsze od szerokości litery „M”. TEXbook: 60

Postać składu jest determinowana użytymi komendami. Komend takich jest bardzo dużo i pozwalają na dokładną kontrolę procesu składu. W praktyce w większości przypadków korzystamy z niewielkiej ich ilości.

[1] TEX dysponuje również europejskimi miarami: punkt *Didôta* — `1dd = 1.07pt` oraz cycero — `1cc=12dd` — przyp. tłum.

3.2 Format strony

Na każdej stronie można wyodrębnić trzy podstawowe części. Powyżej głównego tekstu umieszcza się główkę strony: zawiera ona najczęściej tytuł, numer strony i może się różnić na stronach parzystych i nieparzystych. Niżej znajduje się główny tekst łącznie z przypisami, a najniżej stopka, która może zawierać numer strony.

W dotychczasowych przykładach główka była pusta. Stopka zawierała albo ułożony centralnie numer strony, lub jeśli używaliśmy `\nopagenumbers`, również była pusta. O główkach i stopkach powiemy sobie więcej nieco później. Teraz skoncentrujmy się na tekście głównym.

Nową stronę możemy rozpocząć używając komend `\vfill \eject`. Komenda `\eject` wymusza zakończenie aktualnej strony, podczas gdy `\vfill` powoduje wypełnienie całej pozostałej przestrzeni na dole strony materiałem niewidocznym — odpowiednikiem drukarskiego justunku (spróbuj wykonać eksperyment i opuść `\vfill`, aby się przekonać jak działa justowanie tekstu w pionie).

Aktualna szerokość tekstu na stronie określana jest komendą `\hsize`. Szerokość może być w każdej chwili zmieniona, na przykład do 4 cali, przez komendę `\hsize = 4 in`. Jak widać, dotyczy to również pojedynczego akapitu. Istnieje także możliwość podania aktualnej szerokości akapitu w stosunku do szerokości dotychczasowej przez użycie wyrażenia `\hsize = .75\hsize`.

Pionową analogią do `\hsize` jest `\vsize` określające aktualną wysokość głównego tekstu strony. Może być zmieniane analogicznie jak `\hsize`. Zatem `\vsize = 8 in` używamy, jeśli chcemy zmienić wysokość tekstu do ośmiu cali. Zwróć uwagę, że `\vsize` określa wysokość tekstu bez główek i stopek.

Tekst może być także przesuwany w ramach strony. Lewy górny róg tekstu jest umieszczany 1 cal od góry i 1 cal od lewego brzegu. Komend `\hoffset` i `\voffset` używa się do przesuwania tekstu odpowiednio w kierunku poziomym i pionowym. A więc `\hoffset = .75 in` i `\voffset = -.5 in` przesuną tekst dodatkowo o 0.75 cala w prawo i 0.5 cala w górę. TEXbook: 251

Komendy formatu strony

Nazwa	Komenda	Wartość domyślna (w calach)
szerokość	`\hsize`	6.5
wysokość	`\vsize`	8.9
przesunięcie poziome[2]	`\hoffset`	0
przesunięcie pionowe[2]	`\voffset`	0

[2] Tekst jest umieszczany domyślnie 1 cal od góry i 1 cal od lewego górnego brzegu kartki.

▷ Ćwiczenie 3.1 Wprowadź jeden akapit tekstu o długości kilku wierszy. Wykonaj kilka jego kopii i umieść komendę `\hsize = 5 in` na początku pierwszego i `\hsize = 10 cm` na początku drugiego akapitu. Spróbuj także kilku innych wartości `\hsize`.

▷ Ćwiczenie 3.2 Umieść `\hoffset = .5 in` i `\voffset = 1 in` na początku poprzedniego ćwiczenia.

▷ Ćwiczenie 3.3 A teraz umieść na początku pierwszego akapitu `\vsize = 2 in`

W poprzednim rozdziale dowiedzieliśmy się, że dzięki komendzie `\magstep` możemy powiększać wielkość czcionki. Możliwe jest także powiększenie od razu całego składanego tekstu. Jeżeli na początku pliku wejściowego pojawi się `\magnification = \magstep 1`, uzyskasz powiększenie o ok. 20%. Istnieje także możliwość używania innych, dostępnych w danym systemie wartości `\magstep`. **Podkreślenia wymaga fakt, że `\magnification` należy wprowadzić zanim TEX złoży pierwszą czcionkę.** Powiększanie czcionki pociąga za sobą problem jednostek; jeżeli tekst ma być powiększony o 20% i w zbiorze wejściowym pojawi się `\hsize = 5 in`, czy oznacza to, że strona będzie szerokości 5, czy 6 cali (powiększona o 20%)? Otóż, o ile nie zażyczysz sobie inaczej, powiększeniu ulega każdy wymiar, czyli w tym wypadku w składzie otrzymamy szerokość 6 cali. Jednoczesny wzrost wszystkich wielkości odbywa się zgodnie z powiększeniem określonym przez `\magnification`. W szczególnych sytuacjach może być to niepożądane: na przykład przestrzeń dokładnie 3 cali może być potrzebna do umieszczenia rysunku. W takim przypadku symbol jednostki poprzedzamy słowem `true` tak, że `\hsize = 5 true in` ustali rozmiar wiersza na 5 cali, niezależnie od wielkości powiększenia.

TEXbook: 59–60

▷ Ćwiczenie 3.4 Umieść `\magnification = \magstep 1` na początku swojego zbioru i przyjrzyj się różnicom w składzie.

3.3 Postać akapitu

Czytając twój plik wejściowy TEX czyta pojedynczy akapit i od razu go składa. Oznacza to dużą kontrolę nad jego postacią, ale także wymaga pewnej uwagi. Dowiedzieliśmy się już, że komenda `\hsize` może być użyta do zmiany szerokości pojedynczego akapitu. Przypuśćmy jednak, że na początku naszego zbioru pojawi się:

```
\hsize = 5 in
Four score and seven years ...
⋮
... from this earth.
\hsize = 6.5 in
```

Jaka będzie szerokość akapitu? Komenda `\hsize` została zmieniona w akapicie dwukrotnie — na początku i na końcu. Ponieważ akapit nie został zamknięty przed ponowną zmianą szerokości (przez umieszczenie wolnego wiersza lub komendy `\par`), skład będzie miał szerokość 6.5 cala. Jeśli przed `\hsize = 6.5 in` umieścimy wolny wiersz, szerokość zmieni się na 5 cali. Zatem, generalizując, wartości parametrów aktualne przed zamknięciem akapitu są tymi, które uwzględnia TEX.

Oto tablica niektórych parametrów akapitu:

Niektóre parametry akapitu

Funkcja	Komenda	Wartość domyślna
szerokość	`\hsize`	6.5 cali
wcięcie akapitowe	`\parindent`	20 punktów
odległość między wierszami	`\baselineskip`	12 punktów
odległość między akapitami	`\parskip`	0 punktów

Używając komendy `\noindent` na początku akapitu unikamy automatycznego umieszczania wcięcia akapitowego. Dotyczy ono tylko tego akapitu, w którym zostało wywołane. Komenda `\parindent = 0 pt` spowoduje usunięcie wcięć we wszystkich kolejnych akapitach.

Do umieszczania odstępu między akapitami służy komenda `\vskip`. Jeśli między akapitami wprowadzimy `\vskip 1 in`, odległość między nimi zwiększy się dodatkowo o 1 cal. **Zwróć uwagę, że w tym wypadku nie pojawi się żadna dodatkowa przestrzeń przy brzegach strony**. Komenda `\vskip 1 in` nie działa, jeśli pojawi się u góry strony! Ma to w wielu wypadkach swoje uzasadnienie. Ale co zrobić, jeśli wolna przestrzeń jest potrzebna — na przykład przed tytułem rozdziału? Możesz rozpocząć stronę od znaku `\␣`, co w efekcie spowoduje złożenie pustego akapitu wielkości jednego wiersza. Wolna przestrzeń będzie zatem wprowadzona dzięki stałemu działaniu komend `\baselineskip` i `\parskip`. Prostszą metodą jest zastąpienie komendy `\vskip` komendą `\vglue`: `\vglue 1 in` pozostawi u góry strony wolną przestrzeń wielkości jednego cala. W ogólnym wypadku należy się uciec do komend `\topinsert` i `\endinsert`. Tekst umieszczony miedzy nimi pojawi się u góry strony. Komenda `\vskip` zachowa w tym wypadku swoją ważność. Jest to wygodne kiedy planujemy umieszczenie tam rysunku określonej wielkości.

TEXbook: 352

TEXbook: 115

Istnieją także komendy do tworzenia określonych, małych odstępów pionowych: `\smallskip`, `\medskip` i `\bigskip`. Oto ich wielkości:

`\smallskip`: `\medskip`: `\bigskip`:

Użycie komend `\rightskip` i `\leftskip` daje możliwość elastyczniejszej kontroli szerokości akapitu. Na przykład, `\leftskip = 20 pt` powoduje przesunięcie lewego marginesu o dodatkowe 20 punktów. Wartości ujemne spowodują cofnięcie lewego marginesu. Komenda `\rightskip` zmienia w podobny sposób położenie prawego marginesu. Z kolei komenda `\narrower` jest równoznaczna jednoczesnemu użyciu komend `\leftskip` i `\rightskip` z wartościami równymi aktualnemu `\parindent`. Przydaje się to w przypadku długich cytatów, a ten akapit jest przykładem działania. TEXbook: 100

▷ Ćwiczenie 3.5 Złóż dwa akapity o następujących wymaganiach: lewy margines obu akapitów ma być wcięty o 1.5 cala, prawy margines — o 0,75 cala, odstęp między akapitami ma wynosić dokładnie 1 cal.

Dzięki komendom `\hangindent` i `\hangafter` w pojedynczym akapicie mogą występować wiersze różnych długości. Wielkość wcięcia zależy od wartości `\hangindent`. Jeśli jest ona dodatnia wcięcie pojawia się z lewej, jeśli ujemna — z prawej strony. Liczba wierszy, w których pojawia się wcięcie sterowana jest przez `\hangafer`; wartość dodatnia określa liczbę wierszy, po opuszczeniu których zacznie działać `\hangindent`. Zatem, wartości `\hangindent = 1.75 in` i `\hangafter = 6` (parametry dla tego akapitu) spowodują pozostawienie pierwszych sześciu wierszy w całości i wcięcie pozostałych o 1.75 cala z lewej strony. Odwrotnie — `\hangindent = -1.75 in` i `\hangafter = -6` spowodują wcięcie pierwszych sześciu wierszy o 1.75 cala z prawej strony. W każdym akapicie TEX nadaje powyższym komendom domyślne (*default*) wartości `\hangindent = 0 pt` i `\hangafter = 1`. Komendy te są wygodne w przypadku akapitów z „zawieszonym wcięciem" oraz w przypadku rozmieszczania tekstów wokół rysunków. `\hang` umieszczone na początku akapitu daje w efekcie pierwszy wiersz o pełnej długości (`\hangafter=1`) i pozostałą część akapitu wciętą o aktualną wartość parametru `\parindent`. TEXbook: 355 TEXbook: 102

Oto poprzedni akapit powtórzony z `\hangafter = -6` i `\hangindent = -1.75`.

Dzięki komendom `\hangindent` i `\hangafter` w pojedynczym akapicie mogą występować wiersze różnych długości. Wielkość wcięcia zależy od wartości `\hangindent`. Jeśli jest ona dodatnia wcięcie pojawia się z lewej, jeśli ujemna — z prawej strony. Liczba wierszy, w których pojawia się wcięcie sterowana jest przez `\hangafer`; wartość dodatnia określa liczbę wierszy, po opuszczeniu których zacznie działać `\hangindent`. Zatem, wartości `\hangindent = 1.75 in` i `\hangafter = 6` spowodują pozostawienie pierwszych sześciu wierszy w całości i wcięcie pozostałych o 1.75 cala z lewej strony. Odwrotnie — `\hangindent = -1.75 in`

i `\hangafter = -6` (parametry dla tego akapitu) spowodują wcięcie pierwszych sześciu wierszy o 1.75 cala z prawej strony. W każdym akapicie TEX nadaje powyższym komendom domyślne (*default*) wartości `\hangindent = 0 pt` i `\hangafter = 1`. Komendy te są wygodne w przypadku akapitów z „zawieszonym wcięciem" oraz w przypadku rozmieszczania tekstów wokół rysunków. `\hang` umieszczone na początku akapitu daje w efekcie pierwszy wiersz o pełnej długości (`\hangafter=1`) i pozostałą część akapitu wciętą o aktualną wartość parametru `\parindent`. TEXbook: 355 TEXbook: 102

Komenda `\parshape` daje możliwość składania akapitów o dużej różnorodności kształtu. TEXbook: 101

Inną użyteczną komendą jest `\item`. Sposób jej użycia jest następujący: `\item{...}`. Komenda powoduje powstanie akapitu wciętego z lewej o wartość określoną przez `\parindent` i pierwszy wiersz dodatkowo rozpoczynający się etykietą z tekstem, który umieszczony był w nawiasie klamrowym. Zwykle używa się tej komendy w połączeniu z `\parskip = 0 pt`, w celu uniknięcia zbyt dużych odstępów między wierszami. `\itemitem` ma podobne działanie jak `\item`, różnica polega na dwukrotnie większym wcięciu.Najlepiej ilustruje to przykład: TEXbook: 102

```
\parskip = 0pt \parindent = 30 pt
\noindent
Odpowiedz na następujące pytania:
\item(1) Po co jest pytanie 1?
\item(2) Po co jest pytanie 2?
\item(3) Po co jest pytanie 3?
\itemitem(3a) Po co jest pytanie 3a?
\itemitem(3b) Po co jest pytanie 3b?
```

co daje

Odpowiedz na następujące pytania:
(1) Po co jest pytanie 1?
(2) Po co jest pytanie 2?
(3) Po co jest pytanie 3?
(3a) Po co jest pytanie 3a?
(3b) Po co jest pytanie 3b?

3.4 Postać wiersza

W większości przypadków TEX dobrze sobie radzi z łamaniem wierszy w ramach akapitu. Niekiedy jednak zachodzi konieczność użycia dodatkowych instrukcji. Złamanie wiersza można wymusić komendą `\hfill \break`. Można także umieścić tekst w pojedyńczym wierszu, a służy do tego komenda `\line{...}`. Tekst umieszczony w nawiasach klamrowych zostanie złożony na szerokość szpalty, co niekiedy daje fatalne rezultaty. Komendy

`\leftline{...}`, `\rightline{...}` i `\centerline{...}` umieszczają tekst odpowiednio przy lewym marginesie, przy prawym marginesie lub w osi szpalty. Zatem

```
\leftline{Jestem po lewej.}
\centerline{Jestem w centrum.}
\rightline{A teraz jestem po prawej.}
\line{A teraz zdaje mi się, że jestem rozstrzelony.}
```

da taki oto skład:

Jestem po lewej.

Jestem w centrum.

A teraz jestem po prawej.

A teraz zdaje mi się, że jestem rozstrzelony.

Automatyczną kontrolę spacjowania zapewnia komenda `\hfil`. Powoduje ona umieszczenie całej, dostępnej w wierszu wolnej przestrzeni w miejscu, w którym się pojawia. Tekst `\line{A teraz zdaje mi się, że jestem \hfil rozstrzelony.}` zostanie złożony następująco:

A teraz zdaje mi się, że jestem rozstrzelony.

Jeżeli użyjemy kilku komend `\hfil`, wolna przestrzeń dostępna w wierszu zostanie podzielona równo na odpowiednią ilość części. Zatem `\line{lewica \hfil środek tekstu \hfil prawica.}` da w składzie:

lewica środek tekstu prawica.

▷ Ćwiczenie 3.6 Sporządź następujący skład:
lewa lewa pół lewa ćwierć centrum prawa ćwierć prawa pół prawa

▷ Ćwiczenie 3.7 Sporządź skład, w którym od „lewicy” do „centrum prawicowego” będzie dwa razy większa odległość niż od „centrum prawicowego” do „prawicy”:
lewica centrum prawicowe prawica

Istnieje także możliwość poziomego przesuwania tekstu przy pomocy komendy `\hskip`, która działa analogicznie do `\vskip`.

▷ Ćwiczenie 3.8 Jaki skład otrzymasz po wprowadzeniu następujących danych:
`\line{\hskip 1 in RAZ \hfil DWA \hfil TRZY}`

Wyrównywanie do prawego marginesu może zostać wyłączone po wprowadzeniu komendy `\raggedright` (jest to tzw. skład w chorągiewkę).

3.5 Przypisy

Przypisy wprowadza się w TEX-u według wzoru `\footnote{...}{...}`. W pierwszej parze nawiasów klamrowych umieszcza się odsyłacze. Oto niektóre z nich: `\dag` (†), `\ddag` (‡), `\S` (§), i `\P` (¶). Treść przypisu umieszcza się w drugiej parze nawiasów. Jeżeli jako odsyłaczy używamy liczb, sprawa nieco się komplikuje. Przypis* u dołu strony pojawia się po umieszczeniu komendy `\footnote{$^*$}{To jest przypis u dołu strony.}` tuż po słowie „Przypis". Struktura ta jest nieco skomplikowana; dlaczego — dowiemy się w trakcie nauki składania wyrażeń matematycznych. Teraz możemy na to spojrzeć z punktu widzenia „lepszy rydz ...". TEXbook: 117

▷ Ćwiczenie 3.6 Sporządź stronę tekstu z przypisem zajmującym kilka wierszy.

▷ Ćwiczenie 3.7 Sporządź stronę tekstu z dwoma przypisami.

3.6 Główki i stopki

Wiersze, w których można umieścić tytuł i numer strony pojawiają się nad i pod główną zawartością strony po wywołaniu komend `\headline={...}` i `\footline={...}`[3]. Obie działają tak samo jak komenda `\line{...}`. Przy definiowaniu główek i stopek przydaje się komenda `\pageno` określająca aktualny numer strony. Wywołanie `\headline={\hfil \tenrm Strona \the\pageno}` spowoduje pojawienie się w prawym górnym rogu numeru poprzedzonego słowem „Strona" (a teraz zerknij na prawy górny róg). Bezpieczniej jest podawać wprost nazwę czcionki, z której chcemy korzystać (`\tenrm` to „wewnętrzna", bezpośrednia nazwa 10-cio punktowej antykwy), ponieważ nie zawsze mamy gwarancję, że potrzebna czcionka jest aktualna gdy TEX składa główkę lub stopkę. Komenda `\the` powoduje wydrukowanie wartości przypisanej w danej chwili komendzie (ściślej parametrowi), która pojawia się bezpośrednio po niej. TEXbook: 252–253

Możesz także, Czytelniku, przypisać `\pageno` dowolną liczbę, od której zechcesz numerować kolejne strony. Numerację rzymską wprowadza się liczbami ujemnymi; `\pageno=-1` na początku zbioru spowoduje numerowanie stron cyframi rzymskimi. TEXbook: 252

Główki inne na stronie parzystej, a inne na nieparzystej, otrzymuje się dzięki konstrukcji

```
\headline={\ifodd \pageno {...}\else {...}\fi}
```

* To jest przypis u dołu strony.

[3] Wywołanie `\footline` przedefiniuje standardową, „gotową" stopkę z numerem strony umieszczonym w osi —przyp. tłum.

powodującej umieszczenie tekstu w pierwszej parze nawiasów klamrowych na stronach „prawych", a tekstu w drugiej parze — na stronach „lewych".

▷ Ćwiczenie 3.8 Zmień stopkę tak, aby numer strony umieszczony był na środku między półpauzami.

3.7 Nadmiary i niedomiary

Jednym z najmniej przyjemnych doświadczeń dla TEX-owego nowicjusza są nadmiary i niedomiary. Komunikaty na ten temat (*overfull* lub *underfull*) spotyka się do znudzenia w pliku informacyjnym, pojawiają się też na ekranie, jeśli pracujemy w trybie konwersacyjnym. Nadmiary są dodatkowo oznaczane na prawym marginesie składu sztabką — czarnym prostokącikiem wyglądającym tak: . Sztabka pojawia się i wtedy, gdy w naszym pliku wszystko wydaje się być w porządku. A więc po co ta sztabka i jak sobie z nią radzić?

Skład tekstu w TEX-u można porównać do układania pudełek. Mamy dwa rodzaje pudełek: poziome (hbox-y) i pionowe (vbox-y). W większości przypadków odnosi się to do poziomej organizacji tekstu w wiersze, oraz pionowej organizacji akapitów w strony. Przypomnij sobie, że TEX czyta cały akapit zanim zdecyduje jak go przełamać na poszczególne wiersze. Podejście takie jest bardziej praktyczne niż opracowywanie każdego wiersza osobno, ponieważ minimalna poprawka w jednym wierszu może prowadzić do katastrofalnych zmian w pozostałej części akapitu. Wprowadzanie dodatkowych odstępów wyrównujących do prawego marginesu ma miejsce po zebraniu słów w wiersze. Źle świadczy o wierszu zbyt duża odległość między słowami, czyli niedomiar pudełka poziomego (*underfull hbox*). Dokładniej, zły wiersz to taki, którego „lichość" (*badness*) zawiera się miedzy liczbami 0 (znakomicie) i 10.000 (tragicznie). Domyślna wartość parametru `\hbadness` wynosi 1000. Przekroczenie wartości tego parametru pociąga za sobą pojawienie się informacji o niedomiarze. Im większa będzie wartość parametru `\hbadness`, tym mniej będzie się pojawiało takich informacji. Jakiekolwiek informacje o niedomiarze można w prosty sposób zlikwidować wprowadzeniem komendy `\hbadness = 10000`[4].

W podobny sposób TEX składa niekiedy pojedynczy wiersz ciut dłuższy od `\hsize` zyskując w ten sposób bardziej równomierny wygląd. Decyduje o tym parametr `\tolerance`. Przekroczenie wartości tego parametru spowoduje, że TEX zwiększy długość wiersza, nawet jeśli ma to prowadzić do przekroczenia `\hsize`. Jeśli długość wiersza przekroczona zostanie tylko nieznacznie, nie pojawia się na ten temat żadna informacja. Dopuszczalne wydłużenie wiersza określa parametr `\hfuzz`. Jego wartość domyślna wynosi `\hfuzz = .1 pt`. W przypadku wydłużenia wiersza bardziej niż nieznacznie pojawia się problem — TEX umieszcza na marginesie sztabkę będącą poważnym ostrzeżeniem. W pliku LOG pojawia się zawiły

[4] Nie likwiduje to oczywiście samych niedomiarów — przy. tłum.

i rozbudowany komunikat: „`\overfull\hbox ...`" czyli przekroczenie pudełka poziomego wiersza. Istnieje możliwość uniknięcia ostrzeżeń i sztabek; `\tolerance=10000` likwiduje jakiekolwiek informacje na ten temat. Domyślna wartość parametru wynosi `\tolerance=200`. TEXbook: 29

Szerokość sztabki określana jest parametrem `\overfullrule`. Wprowadzenie do zbioru `\overfullrule = 0 pt` spowoduje zniknięcie sztabki, mimo, że nadmiary nadal pozostaną. Będzie je tylko trudniej zlokalizować.

Wiemy już dlaczego pojawiają się informacje o niedomiarach i nadmiarach oraz jak można wpływać na te informacje zmieniając wartości parametrów `\hbadness`, `\hfuzz` oraz `\tolerance`. Mała wartość `\hsize` znacznie utrudnia skład wiersza i powoduje pojawianie się częstszych informacji o niedomiarach i nadmiarach. Są to ostrzeżenia, które możesz zignorować jedynie na własną odpowiedzialność.

Wprowadzenie dodatkowych możliwości dzielenia wyrazów ułatwia niekiedy likwidację nadmiarów. TEX automatycznie wybiera najlepsze miejsce dzielenia, możliwe jest jednak wskazanie miejsc dodatkowych, które pozwolą na korzystniejsze złamanie wiersza. Na przykład, przy automatycznym łamaniu słowo „database"[5] nie zostałoby podzielone. Dopiero napisanie `data\-base` pozwala na wstawienie w tym wyrazie łącznika. Generalnie, użycie komendy `\hyphenation{data-base}` na początku pliku danych pozwoli na łamanie tego słowa po literze „a" ilekroć nastąpi taka potrzeba. W pliku LOG ukazują się informacje na temat możliwości łamania wyrazów w wierszu, w którym wystąpił niedomiar lub nadmiar. Bywa, że najlepszą metodą na niedomiary i nadmiary jest rozsądne zredagowanie tekstu. TEXbook: 28

Nasze rozważania dotyczyły poziomej organizacji tekstu. Poziome niedomiary i nadmiary informują na ile elegancko i poprawnie słowa zostały ułożone w wiersze. Analogicznie, pionowe niedomiary i nadmiary dotyczą formowania akapitów w strony. Na przykład duża tablica, która nie może być złamana w środku, spowoduje ukazanie się informacji o niedomiarze. `\vbadness` działa w przypadku pionowego rozmieszczania tekstu tak jak `\hbadness` w przypadku poziomego.

▷ Ćwiczenie 3.9 Wydrukuj kilka akapitów używając kilku (małych) wartości `\hsize` i zobacz jakie rodzaje nadmiaru to spowoduje. Powtórz ćwiczenie zmieniając wartości: `\hbadness`, `\hfuzz` i `\tolerance`.

[5] wg. angielskich wzorców dzielenia — przyp. tłum.

Rozdział 4
{Grupy, {grupy, {i nadal grupy}}}

Koncepcja zbierania tekstu w grupy pozwala na znaczne uproszczenie zapisu w plikach TEX-owych. Grupa rozpoczyna się znakiem '`{`' zaś kończy znakiem '`}`'. Komendy umieszczone w obrębie grupy przestają działać wraz z jej zamknięciem. Jeśli na przykład w tekście umieścimy `{\bf trzy półgrube słowa}`, to nawias klamrowy otwiera grupę, `\bf` zmienia krój pisma na półgruby, a następny nawias zamyka grupę. Po zamknięciu grupy używany jest ponownie krój pisma, który był aktualny przed jej rozpoczęciem. Jest to prostszy sposób wprowadzania krótkiego tekstu inną czcionką. Możliwe jest także umieszczanie jednej grupy w środku innej.

Kolejny przykład to chwilowe zmiany parametrów składu:

```
{
\hsize = 4 in
\parindent = 0 pt
\leftskip = 1 in
czyli utworzenie akapitu szerokości
⋮
(łatwo się tu pomylić).
\par
}
```

> czyli utworzenie akapitu o szerokości czterech cali i przesunięcie go o jeden cal w stosunku do położenia przed zdefiniowaniem grupy. Tak właśnie został utworzony ten akapit. Po zamknięciu grupy tekst rozmieszczany jest według poprzednich zasad. Zwróć uwagę, że zakończenie akapitu wymaga wywołania komendy `\par` lub pozostawienia wolnego wiersza; w innym przypadku TEX powróci do starych parametrów zanim akapit zostanie złożony (łatwo się tu pomylić).

Tekst w nawiasach klamrowych następujący po komendzie typu np. `\centerline` jest traktowany jako grupa. Zatem `\centerline{\bf Półgruby tytuł}` da umieszczony w osi szpalty tekst złożony czcionką półgrubą, zaś następujący po tym tekst rozpocznie się w nowym wierszu i złożony będzie czcionką aktualną uprzednio.

Przydatna jest również grupa tzw. pusta `{}`. Jej użycie pozwala na wprowadzanie do składu akcentów bez towarzyszących im zwykle liter: na przykład `\~{}` daje samą tyldę. Pusta grupa zabezpiecza także przed pochłanianiem przez TEX-a następujących po sobie spacji, jest więc separatorem komend. Stąd, jeśli piszę `\TeX{} jest wspaniały`, to otrzymam odpowiedni odstęp po słowie „TEX ...". TEXbook: 19–21

▷ Ćwiczenie 4.1 Zmień rozmiary akapitu na jednej stronie tekstu korzystając z zasady grupowania.

▷ Ćwiczenie 4.2 Matematycy anglosascy używają czasem słowa "iff" jako skrótu od "if and only if". W tym wypadku lepiej nie łączyć obu liter „f" w ligaturę. Jak to zrobić (istnieje kilka rozwiązań)?

Zapominanie o nawiasach przychodzi nam bardzo łatwo, a efekty pominiętego nawiasu mogą być opłakane — przykładowo tekst złożony kursywą od pewnego miejsca do samego końca. Dodatkowy nawias `{` w naszym pliku sygnalizowany jest w pliku LOG komunikatem: „`(\end occurred inside a group at level 1)`". Dodatkowy nawias `}` spowoduje komunikat: „`! Too many }'s`".

Rozdział 5
Nie straszna nam matematyka!

TEX pokazuje co jest wart dopiero przy składaniu wyrażeń matematycznych. Zasady i konwencje są złożone, ale możliwości TEX-a uwzględniają je i pozwalają na otrzymanie wysokiej jakości, atrakcyjnego składu. Jeśli planujesz składanie tekstów matematycznych rozdział ten da ci podstawy do tworzenia eleganckich wyników w prawie każdych warunkach. Oczywiście TEX-a można używać bez potrzeby składania skomplikowanych wzorów i równań. W tym wypadku informacje zawarte w dwóch następnych podrozdziałach całkowicie ci wystarczą.

5.1 Nowe symbole

Tekst matematyczny można wprowadzić do zwykłego tekstu na dwa sposoby: w wierszu (*in-line*), czyli jako część wiersza tekstu, lub wyeksponowany (*displayed*), czyli w osi światła wprowadzonego między wierszami. W obu przypadkach otrzymujemy inne rezultaty. Równanie $\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$ umieszczone w wierszu wygląda całkiem inaczej niż to samo równanie wyeksponowane:

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$

Ponieważ kroje pisma i stosowane odstępy są nieco inne w matematyce niż w zwykłym tekście, TEX musi być poinformowany, że zamierzamy wprowadzić wyrażenia matematyczne. Służy do tego symbol '`$`'. Dokładniej: tekst w wierszu jest traktowany jako matematyczny, jeśli ograniczymy go pojedynczymi znakami dolara: `$...$`; jeśli chcemy go wyeksponować używamy dwóch dolarów: `$$...$$`. Zatem `$x = y+1$` da w wierszu równanie $x = y + 1$, zaś `$$x = y+1$$` wyeksponowane

$$x = y + 1$$

Odstępami w obu przypadkach całkowicie „opiekuje" się TEX. Dodawanie spacji w pliku danych nie ma żadnego wpływu na skład. Co zatem robić, jeśli pojawi się konieczność wprowadzenia dodatkowego odstępu lub zwykłego tekstu? Można skorzystać z komendy `\hbox{...}`. Jest ona szczególnie użyteczna w przypadku wyrażeń wyeksponowanych. Zazwyczaj nie ma konieczności umieszczania w tekście matematycznym dodatkowych odstępów, podajemy jednak poniżej komendy, które do tego służą.

Dodatkowe odstępy w tekście matematycznym

Nazwa	Komenda	⊢Rozmiar⊣
Podwójny kwadrat	`\qquad`	
Kwadrat	`\quad`	
Spacja	`\␣`	
Szeroka spacja	`\;`	
Średnia spacja	`\>`	
Wąska spacja	`\,`	
Ujemna wąska spacja	`\!`	

Jeśli przyjrzysz się bliżej ujemnej wąskiej spacji, zauważysz, że — w przeciwieństwie do pozostałych przypadków — linie określające odstęp nachodzą na siebie. Dzieje się tak, ponieważ spacja ujemna działa w odwrotnym kierunku i podczas gdy inne komendy powodują wzrost odległości między znakami, spacja ujemna ją zmniejsza, prowadząc nawet do nachodzenia znaków na siebie.

▷ Ćwiczenie 5.1 Złóż wyrażenie: $C(n,r) = n!/r!\,(n-r)!$. Zwróć uwagę na odstęp przy mianowniku.

Między znakami dolara nie powinno być żadnych wolnych wierszy. TEX zakłada, że tekst matematyczny składany jest w obrębie jednego akapitu; pojawienie się nowego akapitu traktowane jest jako błąd. Okazuje się to pożyteczne, jako że często zapominamy o ograniczających znakach dolara (co i mnie zdarzyło się niejednokrotnie) i w ten sposób chroni nas przed złożeniem pozostałego tekstu jako wyrażenia matematycznego.

Większość tekstów matematycznych wprowadzana jest tak samo w przypadku składania w wierszu i eksponowania (oczywiście z różnicą w ilości zamykających znaków dolara). Wyjątki, takie jak ustawianie w osi kilku eksponowanych równań lub ich numerowanie na którymś z marginesów, omówione zostaną w ostatniej części tego rozdziału.

W składzie matematycznym pojawia się wiele nowych symboli. Większość znaków dostępnych z klawiatury może być użyta bezpośrednio: `+ - / * = ' | < > ( )`. W składzie prezentują się one następująco: $+ - / * = ' | < > ( )$.

▷ Ćwiczenie 5.2 Złóż równanie $a+b = c-d = xy = w/z$ jako tekst matematyczny w wierszu i wyeksponowany.

▷ Ćwiczenie 5.3 Złóż równanie $f(x,y) = x' + x(x+y)$ raz jako tekst matematyczny w ramach akapitu a raz jako wyeksponowany.

Pozostałe symbole, jak łatwo się domyślić, są wcześniej zdefiniowanymi komendami. W ten sposób dostępne są także wszystkie litery greckiego alfabetu: TEXbook: 434

Litery greckie

α	`\alpha`	β	`\beta`	γ	`\gamma`	δ	`\delta`
ϵ	`\epsilon`	ε	`\varepsilon`	ζ	`\zeta`	η	`\eta`
θ	`\theta`	ϑ	`\vartheta`	ι	`\iota`	κ	`\kappa`
λ	`\lambda`	μ	`\mu`	ν	`\nu`	ξ	`\xi`
o	`o`	π	`\pi`	ρ	`\rho`	ϱ	`\varrho`
σ	`\sigma`	ς	`\varsigma`	τ	`\tau`	υ	`\upsilon`
ϕ	`\phi`	φ	`\varphi`	χ	`\chi`	ψ	`\psi`
ω	`\omega`	Γ	`\Gamma`	Δ	`\Delta`	Θ	`\Theta`
Λ	`\Lambda`	Ξ	`\Xi`	Π	`\Pi`	Σ	`\Sigma`
Υ	`\Upsilon`	Φ	`\Phi`	Ψ	`\Psi`	Ω	`\Omega`

▷ Ćwiczenie 5.4 Złóż $\alpha\beta = \gamma + \delta$ jako tekst matematyczny w wierszu i jako wzór wyeksponowany.

▷ Ćwiczenie 5.5 Złóż $\Gamma(n) = (n-1)!$ jako tekst matematyczny w wierszu i jako wzór wyeksponowany.

Niekiedy używa się akcentów nad lub pod symbolami. Komendy stosowane do akcentowania w matematyce różnią się od komend służących do akcentowania w zwykłym tekście. Nie należy ich używać wymiennie. TEXbook: 135–136

Akcenty matematyczne

$\hat{o}$	`\hat o`	$\check{o}$	`\check o`	$\tilde{o}$	`\tilde o`
$\acute{o}$	`\acute o`	$\grave{o}$	`\grave o`	$\dot{o}$	`\dot o`
$\ddot{o}$	`\ddot o`	$\breve{o}$	`\breve o`	$\bar{o}$	`\bar o`
$\vec{o}$	`\vec o`	$\widehat{abc}$	`\widehat {abc}`	$\widetilde{abc}$	`\widetilde {abc}`

Operatory binarne służą do łączenia dwóch obiektów matematycznych w trzeci. Zwykłe dodawanie i mnożenie ($+$ i $\times$) są także operacjami binarnymi. Podczas składu przed i za operatorem TEX wstawia specjalne odstępy. A oto lista niektórych operatorów binarnych: TEXbook: 436

Operatory binarne

$\cdot$	`\cdot`	$\times$	`\times`	$*$	`\ast`	$\star$	`\star`
$\circ$	`\circ`	$\bullet$	`\bullet`	$\div$	`\div`	$\diamond$	`\diamond`
$\cap$	`\cap`	$\cup$	`\cup`	$\vee$	`\vee`	$\wedge$	`\wedge`
$\oplus$	`\oplus`	$\ominus$	`\ominus`	$\otimes$	`\otimes`	$\odot$	`\odot`

Własności obiektów matematycznych ujawniają się poprzez relacje między nimi. Wiemy już jak pokazać, że obiekty są równe, lub większe i mniejsze od innych (symbole te występują na większości klawiatur). Zaprzeczenie relacji wymaga poprzedzenia jej komendą `\not`. Oto niektóre relacje: TEXbook: 436

Relacje

$\leq$	`\leq`	$\not\leq$	`\not \leq`	$\geq$	`\geq`	$\not\geq$	`\not \geq`
$\equiv$	`\equiv`	$\not\equiv$	`\not \equiv`	$\sim$	`\sim`	$\not\sim$	`\not \sim`
$\simeq$	`\simeq`	$\not\simeq$	`\not \simeq`	$\approx$	`\approx`	$\not\approx$	`\not \approx`
$\subset$	`\subset`	$\subseteq$	`\subseteq`	$\supset$	`\supset`	$\supseteq$	`\supseteq`
$\in$	`\in`	$\ni$	`\ni`	$\parallel$	`\parallel`	$\perp$	`\perp`

▷ Ćwiczenie 5.6 Złóż: $\vec{x} \cdot \vec{y} = \langle \vec{x}, \vec{y} \rangle = 0$ wtedy i tylko wtedy, gdy $\vec{x} \perp \vec{y}$.

▷ Ćwiczenie 5.7 Złóż: $\vec{x} \cdot \vec{y} = \langle \vec{x}, \vec{y} \rangle \neq 0$ wtedy i tylko wtedy, gdy $\vec{x} \not\perp \vec{y}$.

A oto jeszcze inne dostępne symbole: TEXbook: 435

Różnorodne symbole

$\aleph$	`\aleph`	ℓ	`\ell`	$\Re$	`\Re`	$\Im$	`\Im`
∂	`\partial`	∞	`\infty`	$\|$	`\\|`	$\angle$	`\angle`
∇	`\nabla`	$\backslash$	`\backslash`	$\forall$	`\forall`	$\exists$	`\exists`
$\neg$	`\neg`	$\flat$	`\flat`	$\sharp$	`\sharp`	$\natural$	`\natural`

▷ Ćwiczenie 5.8 Złóż: $(\forall x \in \Re)(\exists y \in \Re)\ y > x$.

5.2 Ułamki

Istnieją dwie metody składu ułamków: w formie 1/2 bądź w formie $\frac{1}{2}$. Pierwsza jest zapisywana bez szczególnych komend, czyli `$1/2$`; druga korzysta z komendy `\over` i schematu: `{<licznik> \over <mianownik>}`. Wobec tego `$${a+b \over c+d}.$$` daje

$$\frac{a+b}{c+d}.$$

TEXbook: 139–140

▷ Ćwiczenie 5.9 Złóż następujące wzory: $\frac{a+b}{c}$ $\frac{a}{b+c}$ $\frac{1}{a+b+c} \neq \frac{1}{a}+\frac{1}{b}+\frac{1}{c}$.

▷ Ćwiczenie 5.10 Złóż: Dla jakich punktów zachodzi $\frac{\partial}{\partial x}f(x,y) = \frac{\partial}{\partial y}f(x,y) = 0$?

5.3 Indeksy górne i dolne

Skład indeksów górnych i dolnych jest w TEX-u szczególnie łatwy. Znaki będące indeksami poprzedzamy odpowiednio znakiem '_' (indeks dolny) bądź znakiem '^' (indeks górny). Wobec tego `$x^2$` i `$x_2$` dadzą w składzie odpowiednio x^2 i x_2. Umieszczenie kilku znaków w indeksie wymaga ujęcia ich w grupę. Zapisujemy zatem `$x^{21}$` by otrzymać x^{21} i `$x_{21}$` x_{21}. Zauważ, że indeksy są automatycznie składane mniejszą czcionką. Sytuacja nieco się komplikuje dla „indeksów indeksowanych". *Nie możesz* zapisać `$x_2_3$` ponieważ ma to dwie interpretacje: `$x_{2_3}$` lub `${x_2}_3$`, dające w rezultacie x_{2_3} oraz x_{2_3}. Pierwsza wersja odpowiada przyjętym konwencjom zapisu matematycznego. Stosując nawiasy klamrowe musisz ściśle określić rangę indeksu dolnego bądź górnego. Poziom indeksowania jest nieograniczony.

TEXbook: 128–130

Symbol z obu indeksami jednocześnie wymaga użycia obu znaków '_' i '^' w dowolnej kolejności. Zatem można zapisać `$x_2^1$` lub `$x^1_2$`, co za każdym razem da w składzie x^1_2.

▷ Ćwiczenie 5.11 Złóż: e^x e^{-x} $e^{i\pi}+1=0$ x_0 x_0^2 ${x_0}^2$ 2^{x^x}.

▷ Ćwiczenie 5.12 Złóż: $\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$.

W podobny sposób składamy sumy i całki. Zapis postaci `$\sum_{k=1}^n k^2$` da w efekcie $\sum_{k=1}^{n} k^2$. Z kolei `$\int_0^x f(t) dt$` da w składzie $\int_0^x f(t)dt$.

TEXbook: 144–145

Zbliżonego zapisu używamy dla wyrażeń z granicą: konstrukcja `$\lim_{x \to 0}x^x=1$` daje $\lim_{x\to 0} x^x = 1$.

▷ Ćwiczenie 5.13 Złóż następujące wyrażenie: $\lim_{x\to 0}(1+x)^{\frac{1}{x}} = e$.

▷ Ćwiczenie 5.14 Złóż: The cardinality of $(-\infty, \infty)$ is $\aleph_1$.

A oto rada pozwalająca na uzyskanie bardziej eleganckiego składu całek. Przyjrzyj się różnicy między $\int_0^x f(t)dt$ i $\int_0^x f(t)\,dt$. W drugim przypadku zauważysz mały odstęp za $f(t)$ i stwierdzisz, że zapis wzoru jest bardziej czytelny. Została tu użyta dodatkowa spacja, wprowadzona za pomocą `\,`.

▷ Ćwiczenie 5.15 Złóż następującą całkę: $\int_0^1 3x^2\,dx = 1$.

5.4 Pierwiastki

Aby złożyć pierwiastek kwadratowy wystarczy użyć prostej konstrukcji `\sqrt{...}`, np. `$\sqrt{x^2+y^2}$` da w wyniku $\sqrt{x^2+y^2}$. Zauważ, że TEX zapewnia odpowiednie rozmieszczenie symboli oraz wielkość znaku pierwiastka. Pierwiastek sześcienny bądź innego stopnia wymaga użycia `\root` i `\of`. Skład postaci $\sqrt[n]{1+x^n}$ otrzymasz po zapisie `$\root n \of {1+x^n}$`. TEXbook: 130–131

Możliwy jest również alternatywny zapis pierwiastków[1] za pomocą `\surd`; np. `$\surd 2$` złoży $\surd 2$.

▷ Ćwiczenie 5.16 Złóż: $\sqrt{2}$ $\sqrt{\frac{x+y}{x-y}}$ $\sqrt[3]{10}$ $e^{\sqrt{x}}$.

▷ Ćwiczenie 5.17 Złóż następujący wzór: $\|x\| = \sqrt{x \cdot x}$.

5.5 Linie nad i pod wyrażeniami

Umieszczenia linii nad bądź pod wyrażeniami matematycznymi wymaga użycia konstrukcji `\overline{...}` i `\underline{...}`. `$\overline{x+y}=\overline x + \overline y$` da w efekcie $\overline{x+y} = \overline{x} + \overline{y}$. Zauważ, że linie ponad literami są umieszczane na różnych wysokościach. Użycie `\overline{\strut x}` zwiększy wysokość linii ponad x. TEXbook: 130–131

Do podkreślania tekstów niematematycznych używa się `\underbar{...}`.

[1] spotykany w literaturze anglosaskiej — przyp. tłum.

▷ Ćwiczenie 5.18 Złóż: $\underline{x}$ $\overline{y}$ $\overline{\underline{x+y}}$.

5.6 Przeróżne nawiasy

Najczęściej używanymi w matematyce są nawiasy kwadratowe, klamrowe i okrągłe. Jak już wiesz, zapis `[ ] \{ \}( )` da w składzie [ ] { } ( ). Zróżnicowanie wielkości nawiasów poprawia czytelność wyrażeń matematycznych, jak w poniższym przykładzie:

$$\big(a \times (b+c)\big)\big((a \times b)+c\big).$$

Zwiększenie nawiasu wymaga poprzedzenia go jedną z komend: `\bigl`, `\Bigl`, `\biggl` i `\Biggl` — dla nawiasów „lewych" i podobnie: `\bigr`, `\Bigr`, `\biggr` oraz `\Biggr` — dla nawiasów „prawych". Zapis `$\Bigl[$` i `$\Bigr]$` da w efekcie $\Big[$ i $\Big]$. TEXbook: 145–147

Oto tablica porównawcza wielkości niektórych nawiasów.

Nawiasy o różnej wielkości

$\{$	`\{`	$\}$	`\}`	$($	`(`	$)$	`)`
$\bigl\{$	`\bigl\{`	$\bigr\}$	`\bigr\}`	$\bigl($	`\bigl(`	$\bigr)$	`\bigr)`
$\Bigl\{$	`\Bigl\{`	$\Bigr\}$	`\Bigr\}`	$\Bigl($	`\Bigl(`	$\Bigr)$	`\Bigr)`
$\biggl\{$	`\biggl\{`	$\biggr\}$	`\biggr\}`	$\biggl($	`\biggl(`	$\biggr)$	`\biggr)`
$\Biggl\{$	`\Biggl\{`	$\Biggr\}$	`\Biggr\}`	$\Biggl($	`\Biggl(`	$\Biggr)$	`\Biggr)`

Wybór odpowiedniej wielkości nawiasu możesz pozostawić TEX-owi poprzedzając odpowiednie nawiasy komendami `\left` i `\right`. Wobec tego `\left[...\right]` spowoduje dobranie wielkości nawiasów do wielkości zawartego między nimi materiału. **Uwaga:** każde użycie `\left` i nawiasu wymaga odpowiedniego nawiasu zamykającego poprzedzonego komendą `\right`. Przykładowo `$$\left|{a+b \over c+d}\right|$$` da TEXbook: 148

$$\left|\frac{a+b}{c+d}\right|$$

Separatory matematyczne

(	(	)	)	[	[
]	]	{	\{	}	\}
⌊	\lfloor	⌋	\rfloor	⌈	\lceil
⌉	\rceil	⟨	\langle	⟩	\rangle
/	/	\	\backslash	|	|
‖	\\|	↑	\uparrow	⇑	\Uparrow
↓	\downarrow	⇓	\Downarrow	↕	\updownarrow
⇕	\Updownarrow				

▷ Ćwiczenie 5.19 Złóż: $\lfloor \lceil x \rceil - 1 \rfloor < x$.

5.7 Te funkcje specjalne

Niektóre funkcje występuje szczególnie często w tekstach matematycznych. W równaniu takim jak $\sin^2 x + \cos^2 x = 1$ funkcje trygonometryczne sin i cos należy składać czcionką prostą, w odróżnieniu do innych tekstów matematycznych składanych tzw. *kursywą matematyczną*. Jest to przyjęta konwencja zapisu pozwalająca odróżnić funkcję sin od iloczynu trzech zmiennych $s\,i\,n$. Oto tablica funkcji specjalnych: TEXbook: 162

Funkcje matematyczne

\sin	\cos	\tan	\cot	\sec	\csc	\arcsin	\arccos
\arctan	\sinh	\cosh	\tanh	\coth	\lim	\sup	\inf
\limsup	\liminf	\log	\ln	\lg	\exp	\det	\deg
\dim	\hom	\ker	\max	\min	\arg	\gcd	\Pr

▷ Ćwiczenie 5.20 Złóż: $\sin(2\theta) = 2\sin\theta\cos\theta \quad \cos(2\theta) = 2\cos^2\theta - 1$.

▷ Ćwiczenie 5.21 Złóż:

$$\int \csc^2 x\,dx = -\cot x + C \qquad \lim_{\alpha\to 0}\frac{\sin\alpha}{\alpha} = 1 \qquad \lim_{\alpha\to\infty}\frac{\sin\alpha}{\alpha} = 0.$$

▷ Ćwiczenie 5.22 Złóż:

$$\tan(2\theta) = \frac{2\tan\theta}{1-\tan^2\theta}.$$

5.8 Popatrz, popatrz!

Istnieje szczególna komenda przydatna w składzie prawie każdego artykułu matematycznego i wymaga ona specjalnego omówienia. Jest to raczej *makrodefinicja* (komenda zastępująca użycie ciągu komend), krócej makro `\proclaim`. Znajduje zastosowanie przy składaniu wyróżnionych teorii, twierdzeń itp. Akapit poprzedzony słowem `\proclaim` jest rozdzielony na dwie części: pierwsza obejmuje tekst do pierwszej napotkanej kropki włącznie, druga — to pozostały tekst akapitu. Pomysł polega na potraktowaniu części pierwszej jako etykiety, np. „Twierdzenie 1." lub „Wniosek B." Pozostała część jest treścią twierdzenia bądź wniosku. Oto przykład: TEXbook: 202–203

```
\proclaim Twierdzenie 1. W kraju ślepców jednooki zostaje królem.
```

daje

Twierdzenie 1. *W kraju ślepców jednooki zostaje królem.*

Sformułowanie twierdzenia może oczywiście zawierać wyrażenia matematyczne.

▷ Ćwiczenie 5.23 Złóż:

Lemat 1. $\sum_{i<j}^{n} |X_i - X_j| = 0$ *wtedy i tylko wtedy, gdy* $X_1 = \cdots = X_n$.

5.9 Macierze

Macierze składa się stosując kombinację znaków ustawiania w linii poziomej (osiowania) '`&`' i komendy `\cr` oznaczającej koniec wiersza. Schemat ogólny to `$$\pmatrix{...}$$`. Pomiędzy klamrami umieszcza się wiersze macierzy, każdy zakończony `\cr`. Pozycje w wierszach są oddzielone separatorem '`&`'. Na przykład zapis TEXbook: 176–178

```
$$\pmatrix{
a & b & c & d \cr
b & a & c+d & c-d \cr
0 & 0 & a+b & a-b \cr
0 & 0 & ab & cd \cr
}$$
```

da w wyniku

$$\begin{pmatrix} a & b & c & d \\ b & a & c+d & c-d \\ 0 & 0 & a+b & a-b \\ 0 & 0 & ab & cd \end{pmatrix}$$

▷ Ćwiczenie 5.24 Złóż

$$I_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Oczywiście możliwe jest składanie macierzy ograniczonych innymi nawiasami. Użycie `\matrix` zamiast `\pmatrix` daje macierz bez nawiasów. Zatem żądany nawias musi być podany *explicité* i poprzedzony `\left` bądź `\right`. Oto jak można zmienić macierz z poprzedniego przykładu:

```
$$ \left |
\matrix{
a & b & c & d \cr
b & a & c+d & c-d \cr
0 & 0 & a+b & a-b \cr
0 & 0 & ab & cd \cr
}
\right | $$
```

co da w efekcie

$$\begin{vmatrix} a & b & c & d \\ b & a & c+d & c-d \\ 0 & 0 & a+b & a-b \\ 0 & 0 & ab & cd \end{vmatrix}$$

Otwierający bądź zamykający nawias można usunąć za pomocą `\left.` lub `\right.` (zwróć uwagę na konieczność użycia kropki).

▷ Ćwiczenie 5.24 Złóż

$$|x| = \begin{cases} x & x \geq 0 \\ -x & x \leq 0 \end{cases}$$

Ćwiczenie to można złożyć także przy pomocy makra `\cases`. TEXbook: 175

Macierze można umieścić wewnątrz akapitu, w formie nieeksponowanej, ale wyglądają raczej kiepsko, chyba że posiadają niewiele wierszy.

5.10 Ekspozycja równań

Przedstawione dotąd zasady odnosiły się w jednakowym stopniu do składu tekstu matematycznego w wierszu, jak i wyeksponowanego. W tym miejscu omówimy kilka sytuacji dotyczących wyłącznie ekspozycji równań.

Pierwszy przykład to osiowanie wielowierszowej ekspozycji. Stosujemy znane już znaki '`&`' oraz komendy `\cr` i `\eqalign`. Schemat konstrukcji to `$$\eqalign{...}$$`; równania ustawiane w wierszu kończy się komendą `\cr`, w każdym równaniu należy umieścić znak `&` w miejscu osi równań. Najczęściej oś ekspozycji wyznaczają znaki równości choć nie jest to obowiązujące. Przykładowo zapis: TEXbook: 190–192

```
$$\eqalign{
a+b &= c+d \cr
x &= w + y + z \cr
m + n + o + p &= q \cr
}$$
```

oznacza

$$\begin{aligned} a + b &= c + d \\ x &= w + y + z \\ m + n + o + p &= q \end{aligned}$$

Równania eksponowane mogą być numerowane na prawym bądź lewym marginesie strony. Umieszczony po komendzie `\eqno` tekst zostanie dosunięty do prawego marginesu. Zapis `$$ x+y=z \eqno (1)$$` da w składzie

$$x + y = z \tag{1}$$

Numerację po stronie lewej umożliwia komenda `\leqno`.

Wreszcie, przypuśćmy że zwykły tekst powinien znaleźć się w obrębie równania eksponowanego. Można to osiągnąć umieszczając go w pudełku poziomym (*hbox*). `$$X=Y \hbox{ if and only if }x=y.$$` da w składzie

$$X = Y \text{ if and only if } x = y.$$

▷ Ćwiczenie 5.25 Rozwiąż kilka problemów ze stron 180–181 „The TEXbook".

Rozdział 6
Pod sznurek . . .

Potrzeba umieszczenia tabeli w tekście zdarza się dosyć często. Na szczęście TEX potrafi to łatwo wykonać. Istnieją dwie odrębne metody rozmieszczania tekstu względem wybranych osi. Pierwsza przypomina nieco ustawianie tabulatorów maszyny do pisania. Każda wiersz składu traktowany jest indywidualnie, zgodnie z ustawioną tabulacją kolumn, ale o wiele bardziej elastycznie niż da się to zrobić na maszynie do pisania. Druga metoda to środowisko ustawiające tekst w wierszach i składające jednocześnie całą tabelę w oparciu o przygotowany wzorzec.

6.1 Najpierw tabulacja

Osiowanie tekstu wykorzystujące środowisko tabulacji wymaga określenia pozycji tabulacji za pomocą komendy `\settabs`. Każdy wiersz tekstu wykorzystujący te pozycje rozpoczynany jest symbolem sterującym `\+` zaś kończy się komendą `\cr` (liczba odstępów w wierszach pliku wejściowego jest nieistotna).

Najprostszy sposób użycia `\settabs` to umieszczenie tekstu w kolumnach o równej szerokości. Komendy `\settabs 4 \columns` wyznaczą cztery takie kolumny. Podział tekstu wiersza na poszczególne kolumny dokonany zostanie w oparciu o umieszczone w tekście znaki '`&`'. TEX napotykając '`&`' „przeskoczy" do następnej pozycji tabulacji, aby tam kontynuować skład znaków i wyrazów. Przykładowo: TEXbook: 231

```
\settabs 4 \columns
\+ British Columbia & Alberta & Saskatchewan & Manitoba \cr
\+ Ontario & Quebec & New Brunswick & Nova Scotia \cr
\+ & Prince Edward Island & Newfoundland \cr
```

złoży poniższą tabelkę

British Columbia	Alberta	Saskatchewan	Manitoba
Ontario	Quebec	New Brunswick	Nova Scotia
	Prince Edward Island	Newfoundland	

Zauważ możliwość przeskoczenia pozycji tabulacji i brak konieczności wykorzystania wszystkich pozycji w danym wierszu. Zamiana tej samej tabelki na pięciokolumnową wymaga jedynie modyfikacji: `\settabs 5 \columns`; trzy wiersze z ostatniego przykładu zostaną złożone następująco:

British Columbia Alberta Saskatchewan Manitoba
Ontario Quebec New Brunswick Nova Scotia
Prince Edward Island Newfoundland

W powyższym przykładzie kolumny są oczywiście węższe. W ostatnim wierszu nastąpiło także nałożenie dwóch haseł. Stało się tak ponieważ TEX przeskakuje do następnej pozycji tabulacji, nawet jeśli oznacza to cofnięcie w składanym wierszu.

Istnieje interesująca relacja między grupowaniem i tabulacją. Wartości `\settabs` odnoszą się tylko do grupy, w której zostały zdefiniowane. Możliwa jest więc chwilowa zmiana ustawienia tabulacji, ograniczona do grupy ujętej w nawiasy klamrowe. Ponadto każda sekcja tabeli jest traktowana jako grupa. Pozwala to na zmianę czcionki, np. `\bf`, bez potrzeby zamykania tekstu w klamry. Dodatkowe możliwości to umieszczanie haseł w osi kolumny, po lewej lub prawej jej stronie, a także wypełnianie sekcji w tabeli linią ciągłą bądź kropkowaną. Każda sekcja zawiera na końcu domyślny `\hfil`, tekst w sekcji zostanie wobec tego dosunięty do lewej. Dodanie `\hfil` przed tekstem spowoduje jego umieszczenie w osi kolumny, identycznie jak się to dzieje w przypadku komendy `\line`. Z kolei `\hfill` poprzedzające tekst dosunie go do prawej. `\hfill` działa podobnie jak `\hfil`, z tym że „rozpycha się" silniej w dostępnej wolnej przestrzeni — wieksza ilość '`l`' wskazuje na silniejsze działanie komendy.

```
\settabs 4 \columns
\+ \hfil British Columbia & \hfill Alberta \qquad & \bf Saskatchewan
                                  & Manitoba \cr
\+ \hfil Ontario & \hfill Quebec \qquad & \bf New Brunswick
                                  & Nova Scotia \cr
\+ \hfil --- & \hfill * \qquad & \bf Newfoundland
                                  & Prince Edward Island \cr
\+ \dotfill && \hrulefill & \cr
```

złoży tabelę z hasłami pierwszej kolumny umieszczonymi w jej osi, w drugiej kolumnie dosuniętymi do prawej wraz z „porcją" `\qquad`, w trzeciej złożonymi czcionką półgrubą. Komendy `\dotfill` i `\hrulefill` dają alternatywne zapisy w pozycjach naszej eleganckiej tabeli.

British Columbia	Alberta	**Saskatchewan**	Manitoba
Ontario	Quebec	**New Brunswick**	Nova Scotia
—	*	**Newfoundland**	Prince Edward Island

▷ Ćwiczenie 6.1 Zmień postać powyższej tabeli, tak aby tekst każdej sekcji był umieszczony w osi kolumny.

Pozycje tabulacji mogą być ustawione bardziej elastycznie niż tylko dzieląc szpaltę na kolumny o równej szerokości. Służy do tego określenie wzorcowego wiersza o postaci ogólnej `\settabs \+ ... & ... & ... \cr`. Odległości ustalone pomiędzy znakami '`&`' determinują ustawienie znaków tabulacji. Przykładowo `\settabs \+ \hskip 1 in & \hskip 2 in & \hskip 1.5 in & \cr` ustawi pierwszy tabulator w odległości jednego cala od lewego marginesu, następny dwa cale dalej i trzeci 1.5 cala. Możliwe jest użycie w tym samym celu tekstu, co ilustruje kolejny przykład wzorca: `\settabs \+ \quad Prowincja \quad & \quad Ludność \quad & \quad Powierzchnia \quad & \cr`. Szerokość każdej kolumny będzie dokładnie równa szerokości wyrazu nagłówka wraz z otaczającymi go odstępami wielkości kwadratu. Oto przykład ze wszystkimi szczegółami:

```
\settabs \+ \quad Rok \quad & \quad Cena \quad
& \quad Dywidenda & \cr
\+ \hfill Rok \quad & \quad Cena  \quad & \quad Dywidenda \cr
\+ \hfill 1971 \quad & \quad 41--54 \quad & \qquad \$2.60   \cr
\+ \hfill 2    \quad & \quad 41--54 \quad & \qquad \$2.70 \cr
\+ \hfill 3    \quad & \quad 46--55 \quad & \qquad \$2.87 \cr
\+ \hfill 4    \quad & \quad 40--53 \quad & \qquad \$3.24 \cr
\+ \hfill 5    \quad & \quad 45--52 \quad & \qquad \$3.40 \cr
```

co daje TEXbook: 247

Rok	Cena	Dywidenda
1971	41–54	$2.60
2	41–54	$2.70
3	46–55	$2.87
4	40–53	$3.24
5	45–52	$3.40

▷ Ćwiczenie 6.2 Przesuń powyższą tabelę bliżej osi strony.

▷ Ćwiczenie 6.3 Jednym ze sposobów umieszczania kilku wierszy w osi strony jest zapis: `$$\vbox{...}$$`. Skorzystaj z niego dla powyższej tabeli. Czy wiersz z `\settabs` musi być włączony do pudełka `\vbox`?

▷ Ćwiczenie 6.4 Udoskonal ostatni przykład umieszczając poniżej nagłówków poziomą linię. Komenda `\hrule` wykona taką linię gdy zostanie zapisana pomiędzy wierszami tabeli. Użyj komendy `\strut` po znakach `\+` wiersza zawierającego nagłówki (`\strut` zwiększa nieco odległość między sąsiednimi wierszami, wielkość ta może być zmieniona w stosunku do domyślnej). Zwróć uwagę na rezultaty. TEXbook: 82

▷ Ćwiczenie 6.5 Wykonaj następującą tabelę z umieszczoną w jednej osi kropką dziesiętną (można to interpretować w ten sposób, że pozycje całkowite są wyrównywane do prawej zaś centy do lewej względem kropki dziesiętnej):

Plums	$1.22
Coffee	1.78
Granola	1.98
Mushrooms	.63
Kiwi fruit	.39
Orange juice	1.09
Tuna	1.29
Zucchini	.64
Grapes	1.69
Smoked beef	.75
Broccoli	1.09
Total	$12.55

▷ Ćwiczenie 6.6 Znajdź metodę wykonania spisu treści z wykorzystaniem `\settabs` i z zapisem wyglądającym mniej więcej następująco:

```
Zaczynamy \dotfill & \hfill 1
Wszelkie znaki: duże i małe \dotfill & \hfill 9
```

6.2 Ustawianie tekstu w wierszach w oparciu o zawiłe wzorce

Tabulacja z zastosowaniem `\settabs` jest prosta w użyciu i raz określony wzorzec może być wielokrotnie wykorzystany w dalszym tekście. Posiada jednakże pewne wady. Po pierwsze rozmiar kolumn musi być określony zanim poznamy hasła tabeli. Po drugie nawet jeśli wiemy, że — jak w naszym przykładzie — trzecia kolumna będzie składana czcionką półgrubą, musimy to specyfikować w każdym wierszu. Oba te problemy likwiduje zastosowanie konstrukcji `\halign` o postaci ogólnej:

TEXbook: 235–238

```
\halign{ <wiersz wzorca> \cr
<wiersz pierwszy> \cr
<wiersz drugi> \cr
⋮
<ostatni wiersz tabeli> \cr
}
```

Do podziału wiersza wzorca i samej tabeli na sekcje służy znany nam już symbol '`&`'. W wierszu wzorca, w poszczególnych sekcjach, komendy działają w sposób podobny jak w `\line{...}`. Przykładowo komenda `\hfil` dosunie tekst w obrębie sekcji w lewo

lub w prawo, ewentualnie umieści go w osi. Krój pisma może być zmieniony dla danej sekcji za pomocą `\bf`, `\it` itp. We wzorcu można również umieścić tekst, znajdzie się on wtedy w każdym wierszu danej kolumny. Dodatkowo każda sekcja wzorca musi zawierać dokładnie jeden specjalny symbol '`#`'. W ramach sekcji tabeli będzie on zamieniony na tekst umieszczony zgodnie z położeniem '`#`' we wzorcu.

Rozważmy następujący przykład:

```
\halign{\hskip 2 in $#$ & \hfil \quad # \hfil & \qquad $#$
                                        & \hfil \quad # \hfil \cr
\alpha & alfa & \beta & beta \cr
\gamma & gamma & \delta & delta \cr
\epsilon & epsilon & \zeta & zeta \cr
}
```

Linia wzorca wskazuje, że pierwsza pozycja składanego tekstu umieszczona zostanie dwa cale od lewego marginesu i będzie składana jako tekst matematyczny. Druga pozycja będzie osiowana po odstępie wielkości kwadratu. Trzecia i czwarta będą składane podobnie. A oto wynik:

α	alfa	β	beta
γ	gamma	δ	delta
ϵ	epsilon	ζ	zeta

Pierwszy wiersz naszej tabelki jest formowany przez podstawienie `\alpha` w miejsce pierwszego znaku '`#`' wiersza wzorca, `alfa` — w miejsce drugiego '`#`', `\beta` — trzeciego i wreszcie `beta` — czwartego '`#`'. Cały wiersz jest zapamiętywany do czasu przeanalizowania wszystkich wierszy i automatycznym dostosowaniu szerokości kolumn do najszerszego tekstu, jaki wystąpił w tabeli, wraz z wymaganymi odstępami. Należy rozsądnie korzystać z tego mechanizmu, gdyż zbyt wielka tabela może wypełnić całą dostępną dla TEX-a pamięć. Nie należy zatem składać tabel przekraczających znacznie rozmiar strony.

Reasumując: wiersz wzorca ustala sposób umieszczania tekstu w sekcjach tabeli, zaś dalsze — wprowadzają w owe sekcje indywidualne hasła.

Niekiedy potrzebujemy rozgraniczenia sekcji tabeli za pomocą linii pionowych i poziomych. Umieszczenie linii poziomej wymaga użycia `\hrule`, identycznie jak to robiliśmy w konstrukcji `\settabs`. Nie zamierzamy oczywiście umieszczać linii zgodnie z wzorcem; zastosujemy tutaj zatem komendę `\noalign`. Linie poziome są wprowadzane komendami `\noalign{\hrule}`, pionowe — umieszczając `\vrule` w wierszu wzorca bądź w wierszach tabeli. To jeszcze nie wszystko. Przypuśćmy, że przerobimy naszą ostatnią tabelkę modyfikując wiersz wzorca tak, aby otrzymać linie pionowe i wstawiając ponadto komendy umieszczające linie poziome.

```
\halign{\hskip 2in\vrule\quad $#$\quad & \vrule \hfil\quad # \hfil
                                  & \quad \vrule \quad $#$ \quad \vrule
                                  & \hfil \quad # \quad \hfil \vrule \cr
\noalign{\hrule}
\alpha & alfa & \beta & beta \cr
\noalign{\hrule}
\gamma & gamma & \delta & delta \cr
\noalign{\hrule}
\epsilon & epsilon & \zeta & zeta \cr
\noalign{\hrule}
}
```

Nie daje to, niestety, spodziewanego wyniku:

	α	alfa	β	beta
	γ	gamma	δ	delta
	ϵ	epsilon	ζ	zeta

Widać tu sporo braków: największy to sterczące w lewo linie poziome, poza tym tekst wygląda nieatrakcyjnie, gdyż jest ścieśnięty w ramkach. Tak jak robiliśmy to w przykładach stosujących `\settabs`, linie mogą być umieszczone wyżej po umieszczeniu w wierszu wzorca komendy `\strut`. Kolejny problem może wystąpić przy składzie całej strony. TEX poprawia czasem jej wygląd regulując odrobinkę odległości między wierszami. Może to spowodować powstanie przerwy między liniami pionowymi. Eliminuje się to przez zastosowanie komendy `\offinterlineskip` w ramach `\halign`. Wreszcie wystawanie linii poziomych likwiduje skasowanie we wzorcu `\hskip 2 in`. Przesunięcie tabeli w prawo do tej samej pozycji zapewni komenda `\moveright`. Wobec tego poprawimy naszą tabelkę w następujący sposób:

TEXbook: 82

```
\moveright 2 in
\vbox{\offinterlineskip
\halign{\strut\vrule\quad $#$\quad & \vrule \hfil\quad # \hfil
& \quad \vrule \quad $#$ \quad \vrule & \hfil\quad # \quad \hfil \vrule \cr
\noalign{\hrule}
\alpha & alfa & \beta & beta \cr
\noalign{\hrule}
\gamma & gamma & \delta & delta \cr
\noalign{\hrule}
\epsilon & epsilon & \zeta & zeta \cr
\noalign{\hrule}
}}
```

co daje

α	alfa	β	beta
γ	gamma	δ	delta
ϵ	epsilon	ζ	zeta

Ogólnie, jeśli chcemy skonstruować umieszczoną w osi strony tabelę z ramkami, wkładamy ją do `\vbox` umieszczonego w `\centerline{}`. Sprytniejszy sposób, dający lepszy rezultat to umieszczenie `\vbox` pomiędzy podwójnymi znakami dolara, jak w eksponowanym składzie matematycznym. Oczywiście nie będziemy składać wyrażeń matematycznych, użyjemy jedynie narzędzia, wprowadzającego za jednym zamachem odpowiednie pionowe odstępy oraz osiowanie tabeli w szpalcie. Reasumując, należy kolejno:

(1) umieścić `\vbox` pomiędzy podwójnymi znakami dolara, (2) umieścić `\offinterlineskip` i `\halign` w `\vbox`, (3) w `\halign` zapisać wzorzec wraz z `\strut` na początku; każdą sekcję należy rozpocząć i zakończyć `\vrule`, (4) poniżej i powyżej każdego wiersza tabeli umieścić `\noalign{ \hrule}`. Schemat do naśladowania byłby następujący:

```
$$\vbox{\offinterlineskip
\halign{
\strut \vrule # & \vrule # & ...& \vrule # \vrule \cr
\noalign{\hrule}
<pierwsza kolumna> & <druga kolumna> & ...& <ostatnia kolumna> \cr
\noalign{\hrule}
...
\noalign{\hrule}
<pierwsza kolumna> & <druga kolumna> & ...& <ostatnia kolumna> \cr
\noalign{\hrule}
}}$$
```

Rozdział 7
Jak sobie pościelesz . . .

W rozdziale tym utworzymy nowe komendy. Definiowanie ich, zwane również tworzeniem makr, jest jednym z najsilniejszych narzędzi dostępnych w TEX-u. Pierwszym zastosowaniem które zanalizujemy będzie zaoszczędzenie pisania na klawiaturze przez zastąpienie długich sekwencji krótszymi.

7.1 Długie i krótkie

Nowe komendy definiujemy przy pomocy komendy `\def`. Najbardziej prosta forma to `\def\nowanazwa{...}`. Gdziekolwiek `\nowanazwa` ukaże się w twoim pliku wejściowym, zostanie zastąpiona przez to, co się znajduje w definicji pomiędzy nawiasami klamrowymi. Oczywiście `\nowanazwa` powinna spełniać wymogi zapisu sekwencji sterujących: musi być bądź słowem sterującym, zwanym przez nas komendą (zawiera wyłącznie litery), bądź symbolem sterującym (dokładnie jeden znak nie będący literą). Piszesz przykładowo publikację zawierającą wielokrotnie frazę „Stanford University". `\def\su{Stanford University}` zdefiniuje nową komendę `\su` do wykorzystania w każdej chwili. Zdanie `Prowadzę wykłady w \su.` ma sens a przy tym zaoszczędza pisaniny. Jeśli taka komenda już istnieje, nowa definicja zmieni jej znaczenie. Dotyczy to także komend zdefiniowanych w TEX-u, zachowaj zatem ostrożność przy wyborze nazwy. Każda definicja jest *lokalna* dla grupy, w której została sformułowana. Oto przykład:

```
\def\um{University of Manitoba}
Pierwsze wykłady prowadziłem w \um.
{
\def\um{Universit\'e de Montr\'eal}
Zajęcia kontynuowałem w \um.
}
W końcu wróciłem na \um.
```

co daje

Pierwsze wykłady prowadziłem w University of Manitoba. Zajęcia kontynuowałem w Université de Montréal. W końcu wróciłem na University of Manitoba.

Po zdefiniowaniu każda nowa sekwencja sterująca może być użyta w kolejnej, nowej definicji. Jest to jeden ze sposobów tworzenia prostych druków, np. listów. Zdefiniujmy najpierw prosty list.

```
\def\letter{
\par \noindent
Dear \name,
This is a little note to let you know that your name is \name.
\hskip 2 in Sincerely yours,
\vskip 2\baselineskip
\hskip 2 in The NameNoter
\smallskip \hrule
}
```

List ten używa komendy `\name`, która nie została jeszcze zdefiniowana. Gdy `\letter` zostanie użyte, bieżąca zawartość `\name` ukaże się w treści listu:

```
\def\name{Michael Bishop}
\letter
\def\name{Michelle L\'ev\^eque}
\letter
```

Powyższy zapis utworzy dwie kopie listu, każdą z prawidłowym adresatem i zakończoną poziomą linią.

Dear Michael Bishop,

This is a little note to let you know that your name is Michael Bishop.

Sincerely yours,

The NameNoter

Dear Michelle Lévêque,

This is a little note to let you know that your name is Michelle Lévêque.

Sincerely yours,

The NameNoter

W definicji `\def\name{...}` możemy umieścić w klamrach dowolny tekst; może on posiadać szereg akapitów a także zawierać komendy i symbole sterujące (w kontekście naszego listu będzie wyglądało to nieco dziwnie). W definicji `\letter` można też oczywiście zastosować konstrukcję `\vfill \eject` co pozwoli zmienić stronę po każdym liście.

▷ Ćwiczenie 7.1 Wykonaj blankiet listu stosujący komendy `\nazwisko`, `\adres`, `\miasto` i `\kod`.

▷ Ćwiczenie 7.2 Nienumerowaną listę tworzymy często przy pomocy `\item{$\bullet$}`. Zdefiniuj makro `\bitem`, które działa tak samo.

▷ Ćwiczenie 7.3 Przypuśćmy, że zamierzasz uformować szereg akapitów wykorzystując komendy: `\hangindent = 30 pt`, `\hangafter = 4` i `\filbreak`. Nie przejmuj się tym co one dokładnie robią, istotny w tej chwili jest fakt, że działają tylko w obrębie jednego akapitu. Zdefiniuj pojedynczą komendę `\setpar` umieszczaną przed każdym akapitem składanym zgodnie z naszą specyfikacją.

7.2 Korzystajmy z parametrów

Możemy korzystać z makr w sposób dużo ogólniejszy używając makr z parametrami. Przypomina to nieco wzorzec stosowany wraz z komendą `\halign`. Rozpatrzmy najpierw przypadek makra z jednym parametrem. Postać ogólna to: `\def\nowemakro#1{`*`treść`*`}`. Symbol `#1` może wielokrotnie wystąpić także wewnątrz nawiasów definicji `\nowemakro`. Materiał zawarty między tymi nawiasami traktowany jest jako wzorzec. Gdy `\nowemakro{`*`tekst zastępujący #1`*`}` pojawi się w tekście źródłowym, TEX wstawi zawartość ujętą w nawiasy w każdym miejscu wystąpienia `#1` treści makra. Odstępy użyte w treści definicji mają istotne znaczenie; należy uważać by niechcący nie wstawić spacji przed nawiasem otwierającym.

Jako przykład zmodyfikujmy blankiet listowy z poprzedniego podrozdziału.

```
\def\letter#1{
\par \noindent
Dear #1,
This is a little note to let you know that your name is #1.
\hskip 2 in Sincerely yours,
\vskip 2\baselineskip
\hskip 2 in The NameNoter
\smallskip \hrule
}
```

Możemy zatem użyć:

```
\letter{Michael Bishop}
\letter{Michelle L\'ev\^eque}
```

aby otrzymać

Dear Michael Bishop,

This is a little note to let you know that your name is Michael Bishop.

Sincerely yours,

The NameNoter

Dear Michelle Lévêque,

This is a little note to let you know that your name is Michelle Lévêque.

Sincerely yours,

The NameNoter

Zdefiniujmy teraz `\def\displaytext#1{$$\vbox{#1}$$}` jako nowe makro eksponujące tekst. Wystąpienie w pliku `\displaytext{...}` spowoduje umieszczenie materiału zawartego w klamrach w akapicie oddzielonym odpowiednim światłem i w osi szpalty. Akapit ten, poprzedzony `\hsize = 12 cm`, został złożony przy użyciu tak zdefiniowanego makra `\displaytext`.

Parametr makra nie powinien być dłuższy niż jeden akapit. Wykrycie w obrębie parametru zmiany akapitu spowoduje zasygnalizowanie komunikatu błędu. Brak takiego zabezpieczenia w przypadku przeoczenia nawiasu zamykającego spowodowałby włączenie pozostałej części pliku jako parametru.

▷ Ćwiczenie 7.4 Zdefiniuj makro `\twapremia`, tak aby wywołanie `\twapremia{89}` spowodowało skład zdania: Otrzymałeś 89% premii. Makro powinno oczywiście działać dla innych wartości procentów.

Zastosowanie kilku parametrów nie jest wcale trudniejsze. Schemat dla dwóch parametrów wygląda następująco: `\def\nowemakro#1#2{...}`. Definicja pomiędzy klamrami może zawierać `#1` i `#2` występujące wielokrotnie. Pojawienie się w tekście `\nowemakro{...}{...}` zamieni `#1` definicji makra na materiał zawarty w pierwszej parze nawiasów zaś `#2` — na materiał w drugiej parze. Oto przykład:

```
\def\talks#1#2{#1 talks to #2.}
\talks{John}{Jane}
\talks{Jane}{John}
\talks{John}{me}
```

```
\talks{She}{Jane}
```

John talks to Jane. Jane talks to John. John talks to me. She talks to Jane.

▷ Ćwiczenie 7.5 Podobnie jak w poprzednim ćwiczeniu zdefiniuj makro `\pensja`, tak aby wywołanie: `\pensja{890}{25}` złożyło następujące zdanie: Otrzyma Pan wynagrodzenie 890$, w tym 25% premii.

Ważne jest, jak to już było wspomniane, żeby nie umieścić spacji przed pierwszym nawiasem definicji. Gdyby się tak stało, TEX zinterpretuje definicję w sposób odmienny od opisanego wyżej. Większa od dwóch ilość parametrów wymaga analogicznej definicji. Definicja z trzema parametrami ma postać ogólną: `\def\nowemakro#1#2#3{...}`. Parametry `#1`, `#2` i `#3` mogą wystąpić pomiędzy nawiasami definicji. Gdy `\nowemakro{...}{...}{...}` pojawi się w pliku, materiał zawarty pomiędzy każdym zestawem nawiasów zostanie podstawiony w miejsce odpowiedniego symbolu definicji. Parametrów może być conajwyżej dziewięć.

7.3 Innymi słowy

Przydatna jest czasami możliwość nadania komendzie nazwy alternatywnej. Jeśli preferujesz inną pisownię możesz nazwać `\centerline` przykładowo `\centruj`. Używa się do tego komendy `\let`. Definicja `\let \centruj = \centerline` pozwala stosować zamiennie `\centruj` i `\centerline`. Można to zrobić także z nazwami matematycznymi, np. `\let \tensor = \otimes`. Oto możliwe zastosowanie: TEXbook: 206–207

```
$$ (A \tensor B) (C \tensor D) = AC \tensor BD. $$
```

co daje

$$(A \otimes B)(C \otimes D) = AC \otimes BD.$$

▷ Ćwiczenie 7.6 Zdefiniuj komendy `\ll`, `\cl` i `\rl` będące ekwiwalentami dla `\leftline`, `\centerline` oraz `\rightline`.

Komenda `\let` pozwala użytkownikowi nazywać komendy po swojemu. Ich indywidualny zestaw może być używany zamiast komend dostępnych w standardowym TEX-u.

Rozdział 8
Errare humanum est

Na niepoprawne dane TEX odpowiada komunikatem błędu (*error message*) ukazującym się na ekranie i zapisywanym w pliku LOG. Ponieważ TEX jest skomplikowanym programem, błąd może być wykryty na głębokim poziomie przetwarzania i pełen raport może być w związku z tym długi i zawikłany. Ponadto sam TEX posiada mechanizmy wychodzenia z błędu i zapisuje wykonane w takich wypadkach czynności. Z tych powodów czytanie komunikatów jest dla niewtajemniczonego niezbyt łatwe. Z praktycznego punktu widzenia użytkownika najważniejsza jest wiedza o tym co jest istotne w komunikatach, a co można zignorować. Spójrzmy zatem na typowe błędy i generowane przez nie komunikaty.

8.1 Zapomniane bye

Często popełnianym błędem jest brak komendy `\bye` na końcu pliku wejściowego. Jeśli używasz TEX-a w trybie konwersacyjnym, na ekranie ukaże się w takim wypadku gwiazdka
*
i nic więcej się nie dzieje, ponieważ TEX nie został poinformowany o zakończeniu pracy i w dalszym ciągu oczekuje danych (tym razem z klawiatury). Cokolwiek napiszesz, zostanie to dołączone do danych z plików wejściowych. Najprawdopodobniej użyjesz kończącej pracę sekwencji `\bye <CR>`[1].

8.2 Przekręcona lub nieznana komenda

Jest to dość powszechny błąd. TEX uruchomiony w trybie wsadowym generuje komunikat błędu i kontynuuje pracę ignorując niepoprawną komendę. W trybie konwersacyjnym możliwa jest naprawa błędu; oczywiście nie zmieni to zapisu w pliku wejściowym, który należy poprawić po zakończeniu pracy TEX-a. Przypuśćmy, że twój plik wygląda następująco:

```
\line{lewa strona \hfli prawa strona}
\bye
```

Komenda poprawna to oczywiście `\hfil`. Błędny zapis spowoduje wyświetlenie komunikatu:

[1] `<CR>` to klawisz, którym kończymy wprowadzany wiersz. Nazywany jest *carriage return, enter* lub po prostu *return*. Oznaczony jest niekiedy dużą strzałką skierowaną w lewo.

```
! Undefined  control sequence.
l.1 \line{ Lewa strona \hfli
                              prawa strona}
?
```

Pierwszy wiersz rozpoczyna się od '`!`' i zawiera informację rodzaju błędu (tu: „Niezdefiniowana komenda" — przyp. tłum.). Następny zawiera numer wiersza, w którym błąd wystąpił i poprawny fragment tekstu poprzedzającego błąd. Wiersz kolejny zawiera kontynuację tekstu poza miejscem błędu. Kończący komunikat znak zapytania oznacza oczekiwanie TEX-a na odpowiedź. Oto niektóre dozwolone odpowiedzi: TEXbook: 31

Odpowiedzi na komunikaty błędów TEX-a

Oczekiwana odpowiedź	Zapis dla TEX-a	Rezultat
Podpowiedź (*Help*)	`h` *<CR>*	Wyświetlenie powodu zatrzymania.
Wstawienie (*Insert*)	`i` *<CR>*	Wstawienie fragmentu tekstu.
Wyjście (*eXit*)	`x` *<CR>*	Koniec pracy; strony zakończone znajdą się w pliku DVI.
Przeglądanie (*Scroll*)	`s` *<CR>*	Kontynuacja z wyświetlaniem komunikatów, drobne błędy są ignorowane.
Przebieg (*Run*)	`r` *<CR>*	Kontynuacja z wyświetlaniem komunikatów bez względu na błędy.
Cichy przebieg (*Quiet*)	`q` *<CR>*	Kontynuacja bez wyświetlania komunikatów.
Podjęcie przetwarzania	*<CR>*	Kontynuacja do następnego błędu.

W naszym ostatnim przykładzie rozsądną odpowiedzią będzie `h` *<CR>* wyświetlające diagnozę błędu, następnie `i` *<CR>* pozwalające wprowadzić do przetwarzania dalszy tekst. W tym momencie TEX odpowie napisem `insert>`, będącym zachętą do wprowadzenia poprawnej komendy `\hfil`. A oto zapis konwersacji[2]:

```
? h
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., `\hobx'), type `I' and the correct
spelling (e.g., `I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.
? i
insert>\hfil
[1]
```

[2] W tłumaczeniu: `Komenda kończąca górny wiersz komunikatu błędu nie została zdefiniowana. Jeśli przekręciłeś pisownię (np. '\hobx') - popraw ją po naciśnięciu klawisza 'I'. W przeciwnym wypadku kontynuuj, a ja zapomnę o niezdefiniowanej komendzie` — przyp. tłum.

Końcowe [1] oznacza, że pierwsza (i jedyna) strona została pomyślnie złożona i przesłana do pliku DVI. Oryginalny plik wejściowy nadal, rzecz jasna, wymaga poprawienia.

8.3 Nieprawidłowa nazwa czcionki

Błąd ten, na pozór podobny do omawianych wyżej, generuje odmienny komunikat i na pierwszy rzut oka wprawia w zakłopotanie. Przypuśćmy, że w twoim pliku znajdzie się zapis:
`\font\sf = cmss01`

W tym przypadku zostały przestawione cyfry. Oto komunikat błędu wraz z podpowiedzią[3]:

```
! Font \sf=cmss01 not loadable: Metric (TFM) file not found.
<to be read again>
                   \par
\bye ->\par
            \vfill \supereject \end
l.3 \bye
? h
I wasn't able to read the size data for this font,
so I will ignore the font specification.
[Wizards can fix TFM files using TFtoPL/PLtoTF.]
You might try inserting a different font spec;
e.g., type `I\font<same font id>=<substitute font name>'.
```

Plik rozmiarowy kroju czcionek (TFM, *TEX font metric*) jest pomocniczym plikiem używanym przez TEX-a. W komunikacie istotna jest tylko informacja, że nasz system komputerowy nie posiada specyfikowanej czcionki.

8.4 Błędnie zaznaczana matematyka

Innym typowym błędem jest rozpoczęcie składu matematycznego ('$' lub '$$') i niezakończenie go analogicznym symbolem. Tekst występujący po wyrażeniach jest traktowany jako matematyka i, co gorsza, rozpoczęcie pojawiającej się dalej matematyki kolejnymi symbolami '$' lub '$$' spowoduje traktowanie jej jako zwykłego tekstu. Mówiąc krótko, należy spodziewać się mnóstwa komunikatów o błędach. TEX podejmie próbę naprawy wstawiając

[3] Kolejno: `! Czcionka \sf=cmss01 niedostępna: Nie znaleziono pliku TFM...` i dalej: `Nie dałem rady przeczytać danych wymiarowych dla tego kroju, ignoruję specyfikację. [Magowie potrafią majstrować przy plikach TFM przy pomocy TFtoPL/PLtoTF]. Spróbuj zmienić specyfikację czcionki, np. 'I\font<ta sama nazwa>=<nazwa zewnętrzna>'.` — przyp. tłum.

(w trakcie przetwarzania) brakujące '$' bądź '$$'. W najgorszym razie, problem zostanie rozwiązany przez koniec akapitu, ponieważ nowy akapit będzie automatycznie składany jako zwykły tekst.

Obejrzyj następujący prawidłowy zapis i efekt składu:

```
Ponieważ $f(x) > 0$, $a<b$ i $f(x)$ jest ciągła znajdujemy, że
$\int_a^b f(x)\,dx >0$.
```

Ponieważ $f(x) > 0$, $a < b$ i $f(x)$ jest ciągła znajdujemy, że $\int_a^b f(x)\,dx > 0$.

Jeśli opuścimy drugi znak dolara we fragmencie '`$f(x)$`', otrzymamy następujące komunikaty błędu i podpowiedzi[4]:

```
! Missing $ inserted.
<inserted text>
                $
<to be read again>
                   \intop
\int ->\intop
              \nolimits
l.2 $\int
          _a^b f(x)\,dx >0$.
? h
I've inserted a begin-math/end-math symbol since I think
you left one out. Proceed, with fingers crossed.
?
```

Wiersz rozpoczynający się od '`!`' informuje co zostało zrobione, zaś rozpoczynający się od '`l.2`' pokazuje w którym wierszu pliku wejściowego natknięto się na błąd. Tak jak w innych naszych przykładach, część wiersza poprawnie przeczytana znalazła się w komunikacie w jednej linijce, pozostała — w następnej. Całość wydaje się nieco zawiła, pośrednie komunikaty pokazują co się dzieje w TEX-owych czeluściach. Nowicjusz może je zignorować.

A oto efekt podjętej przez TEX-a próby wyjścia z błędu:

Ponieważ $f(x) > 0$, $a < b$ i $f(x) jest ciągła znajdujemy, że \int_a^b f(x)\,dx > 0$.

Zauważysz tutaj niepożądane ściśnięcie tekstu, złożonego na dodatek kursywą. Jest to typowe potraktowanie zwykłego tekstu jako matematyki. Gdy stwierdzisz na wydruku taki efekt, oznacza on pominięcie w pliku źródłowym symboli '$' lub '$$'.

[4] Kolejno: `!Wstawiono brakujący symbol $...` i `Wstawiłem symbol początku/końca matematyki bo myślę, że o nim zapomniałeś. Kontynuuj trzymając kciuki` — przyp. tłum.

8.5 Błędne wstawienie nawiasów klamrowych

Podczas tworzenia grup bardzo łatwo zapomnieć o zamykającym grupę nawiasie klamrowym lub pomylić ilość takich nawiasów. Rezultatem może być błąd względnie nieszkodliwy, a czasem wręcz katastrofalny. Przykładowo, napisałeś w twoim pliku `{\bf Wytłuszczony tytuł`, zapominając o nawiasie zamykającym. Rezultat będzie taki sam jak w przypadku pominięcia nawiasu otwierającego — tekst do końca pliku zostanie złożony czcionką półgrubą, o ile „po drodze" nie zmieniano kroju. Na zakończenie pracy TEX wygeneruje następujący komunikat[5]:

```
(\end occurred inside a group at level 1)
```

Ponowna pomyłka tego typu, czyli dwa nawiasy otwierające bez odpowiednich nawiasów zamykających, generuje komunikat:

```
(\end occurred inside a group at level 2)
```

Dopóki TEX nie osiągnie końca pliku nie ma możliwości stwierdzenia braku nawiasu zamykającego. Komunikat nie poda ci wobec tego miejsca wystąpienia błędu. W przypadku trudności z jego znalezieniem, możesz zawsze wstawić `\bye` w środku pliku. Uruchom TEX-a i jeśli komunikat się powtórzy przenieś `\bye` w inne miejsce w obrębie pierwszej części tekstu. W ten sposób możesz w końcu osaczyć błąd. Inna metodą jest oczywiście staranne przejrzenie efektu składu.

Znalezienie brakującego nawiasu otwierającego jest dużo łatwiejsze. Oto przykładowe dwa wiersze pliku źródłowego i odpowiednie komunikaty[6]:

```
\bf Tu jest początek}, a tu koniec.
\bye
```

```
! Too many }'s.
l.1 \bf  Tu jest początek}
                          , a tu koniec.
? h
You've closed more groups than you opened.
Such booboos are generally harmless, so keep going.
```

Całkiem możliwe, że brak lewego nawiasu nie będzie dotyczył wiersza, w którym TEX zlokalizował błąd.

Nawiasy nieprawidłowo umieszczone w definicji komendy mogą spowodować bardzo poważny błąd. Definicja taka zawiera niejednokrotnie szereg akapitów. Błąd zatem może

[5] W tłumaczeniu: `\end wystąpił wewnątrz grupy na poziomie 1.` `\end` jest komendą pierwotną TEX-a, tzw. *prymitywem*, który posłużył do zdefiniowania komendy `\bye` — przyp. tłum.

[6] W tłumaczeniu: `! Zbyt wiele '}'...` i dalej: `Zamknąłeś więcej grup niż otworzyłeś. Takie głupstwa są raczej nieszkodliwe, jedź więc dalej.` — przyp. tłum.

nie zostać wykryty na podstawie wystąpienia końca akapitu i coraz więcej tekstu może być traktowana jako element niezakończonej definicji. Możliwe jest nawet wyczerpanie dostępnej pamięci. Sytuacja taka określana jest jako „wymykająca się definicja" (*runaway definition*). TEXbook: 206
Przykładem niech będzie zapis dwóch wierszy z taką definicją:

```
\def\newword{the def
\bye
```

oraz komunikaty błędu i próby diagnozy[7]:

```
Runaway definition?
->the def
! Forbidden control sequence found while scanning definition of \newword.
<inserted text>
                }
<to be read again>
                   \bye
l.2 \bye
? h
I suspect you have forgotten a `}', causing me
to read past where you wanted me to stop.
I'll try to recover; but if the error is serious,
you'd better type `E' or `X' now and fix your file.
x
?
No pages of output.
```

Jest to, rzecz jasna, poważny błąd. Jeśli zdarzy się na początku pliku (jak w naszym przykładzie) nie zostanie złożony żaden materiał.

Nieco lepiej jest w przypadku odwołania do makra z parametrami: brak nawiasu kończącego parametr zostanie zasygnalizowany na końcu akapitu. Wobec tego zdefiniowanie `\def\newword#1{...}` i odwołanie '`\newword{...` ' pozbawione prawego nawiasu zepsuje najwyżej jeden akapit. TEXbook: 205

Reasumując: gdy wystąpi błąd spójrz na wiersz komunikatu rozpoczynający się wykrzyknikiem i prezentujący krótki opis błędu. Następnie zobacz, do którego wiersza TEX przetworzył twój plik. Jeśli błąd jest nadal niejasny, zażądaj dodatkowych informacji naciskając `h <CR>`.

[7] Kolejno: `Wymykająca się definicja?...! Niedozwolona komenda została znaleziona podczas czytania definicji \newword...` i dalej: `Podejrzewam, że zapomniałeś o '}' co zmusiło mnie do czytania poza miejscem, w którym chciałeś bym skończył. Spróbuję z tego wybrnąć, lecz jeśli błąd jest poważny naciśnij 'E' lub 'X' i popraw swój plik.` — przyp. tłum.

Rozdział 9
Na szerokie wody

Rozdział ten omawia kilka aspektów pozwalających bardziej elastycznie i efektywnie korzystać z TEX-a. Dokumenty przez nas tworzone są coraz większe i zastosowanie różnych technik może ułatwić ich składanie.

9.1 Duże pliki, małe pliki

Uruchomiony TEX potrafi zarówno czytać jak i pisać pliki. Pozwala to tworzyć małe i łatwiejsze w obróbce pliki wejściowe. Przykładem niech będzie niniejszy podręcznik, składający się z dziewięciu rozdziałów, wstępu i indeksu. Poza tym każda z części korzysta z szeregu tych samych makr; warto je zatem zapisać w pliku nazwanym, powiedzmy, `makra.tex`. Wstęp zapiszemy w pliku `intro.tex`, zaś każdy z rozdziałów — w osobnym pliku. Do czytania pliku używamy komendy `\input`. Zapis o postaci ogólnej `\input cosik`, umieszczony w pliku wejściowym, spowoduje czytanie pliku o nazwie `cosik.tex` i jego natychmiastowe przetwarzanie, zupełnie tak samo, jakby tekst zawarty w `cosik.tex` był częścią pliku wejściowego. Plik wczytywany również może zawierać komendy `\input`. W praktyce wygodnie jest utworzyć mały plik „czytający" nieduże kawałki, na przykład w ten sposób:

```
\input makra
\input intro
\input roz1
\input roz2
\input roz3
\input roz4
\input roz5
\input roz6
\input roz7
\input roz8
\input roz9
\input indeks
\bye
```

Podczas tworzenia i próbnego składania przetwarzamy jedynie wybrane pliki. Umieszczamy wtedy '`%`' na początku każdego wiersza zawierającego nazwę pliku, który chcemy pominąć.

Komenda `\input` pozwala na użycie przygotowanych wcześniej makr. Na przykład makra do składania listów można zapisać w pliku o nazwie `list.tex`. Makra mogą określać odpowiednie parametry, jak `\hsize`, `\vsize` i inne, a także umieszczać na drukach aktualną datę i czas. Raz napisane, przydają się wielokrotnie. Kolejne listy wystarczy rozpoczynać od komendy `\input list`, zachowując w ten sposób jednorodną postać korespondencji.

▷ Ćwiczenie 9.1 Utwórz plik TEX-owy, który wczytuje inny plik. Spróbuj dwukrotnego wczytania tego pliku, powtarzając komendę `\input`.

9.2 Większe pakiety makr

Szczególnie użyteczne jest projektowanie makr znajdujących zastosowanie w szerokiej gamie dokumentów. Większość uniwersytetów wymaga specyficznych i często skomplikowanych postaci publikacji. Zbiór — inaczej pakiet — makr realizujących takie wymagania niełatwo jest zaprojektować, pochłania to wiele czasu, a powstały plik bywa całkiem spory. Użycie komendy `\input` pozwala korzystać z pakietu tak samo, jak z twoich własnych makr. TEX posiada jeszcze lepsze udogodnienie dla większych pakietów makr.

Pakiety makr można odpowiednio przetworzyć po to, aby umożliwić TEX-owi szybkie czytanie. Wynikiem przetworzenia są tzw. *pliki formatowe* (*format files*), których postać jest tutaj nieistotna. Najważniejsze, że pozwalają one uruchamiać TEX-a z mnóstwem nowych, uprzednio zdefiniowanych komend. Standardowa wersja TEX-a wykorzystuje właśnie taki plik formatowy o nazwie `plain.fmt`.

Wiele ośrodków komputerowych korzysta z pakietu LATEX, pozwalającego automatycznie tworzyć indeksy, spisy treści i bibliografie. LATEX posiada również możliwości włączania do tekstu elementarnych figur geometrycznych, jak kółka, owale, linie i strzałki. Ponadto wykorzystuje tzw. pliki stylów (*style files*) określające specyficzne parametry składu. Dostępnych jest wiele takich plików i niektóre czasopisma akceptują artykuły nadesłane na nośniku magnetycznym, o ile napisano je z użyciem LATEX-a i określonego pliku stylu. Znając TEX-a nietrudno przesiąść się do LATEX-a. Autor pakietu Leslie Lamport napisał podręcznik: **LATEX: A document preparation system**[1].

Amerykańskie Towarzystwo Matematyczne (*American Mathematical Society*) używa w swoich czasopismach formatu o nazwie $\mathcal{AMS}$-TEX. Jest on dostarczany przez Towarzystwo wraz z podręcznikiem Michaela Spivaka: **The Joy of TEX**. Artykuły do czasopism AMS mogą być przesyłane w postaci zapisu magnetycznego plików tworzonych z wykorzystaniem $\mathcal{AMS}$-TEX-a.

[1] Addison-Wesley Reading, Massachusetts, 1986, ISBN 0-201-15790-X

Dostępne są także inne pakiety i niewątpliwie wiele jeszcze zostanie opracowanych. Sprzedawane są po cenach umiarkowanych i mogą być bardzo efektywne w niektórych przypadkach. Stowarzyszenie *TEX Users Group* upowszechnia nowości w swoich publikacjach.

9.3 Linie poziome i pionowe

Tworzenie poziomych i pionowych linii jest w TEX-u bardzo łatwe. Użycie w tekście komendy `\hrule` spowoduje zakończenie bieżącego akapitu i narysowanie poziomej linii o długości równej szerokości szpalty (`\hsize`). Dalszy tekst rozpocznie się od nowego akapitu. Możliwa jest specyfikacja długości linii, np. `\hrule width 5 cm`, a także użycie `\vskip` czy też `\bigskip` dla umieszczenia światła powyżej bądź poniżej linii. A oto przykład:

```
\parindent = 0 pt \parskip = 12 pt
Nieco tekstu powyżej linii.
\bigskip
\hrule width 3 in
A tutaj tekst poniżej linii.
```

co daje

Nieco tekstu powyżej linii.

A tutaj tekst poniżej linii.

W rzeczywistości linia nie tylko posiada *długość* (`width`) trzech cali, ale również *wysokość* (`height`) — wielkość domyślną 0.4 punkta powyżej linii pisma (*baseline*) do jakiej sięga linia bądź czcionka, oraz *głębokość* (`depth`)[2] — 0 punktów, rozmiar liczony poniżej linii pisma. Każdy z tych parametrów może być indywidualnie określony. Jeśli zatem zmienimy w ostatnim przykładzie `\hrule width 3 in height 2 pt depth 3 pt` otrzymamy:

Nieco tekstu powyżej linii.

A tutaj tekst poniżej linii.

Parametry `width`, `height` i `depth` mogą być podawane w dowolnej kolejności.

Linię pionową specyfikujemy analogicznie jak poziomą; w razie potrzeby możemy również określić `width`, `height` i `depth`.W odróżnieniu od linii poziomej `\vrule` nie

TEXbook: 221-222

[2] Można by tu zastosować przyjęty w opisie metalowych czcionek rozmiar zwany *odsadką*, niemniej jednak „głębokość" oddaje bardziej plastycznie sens pojęcia i jest zresztą dokładnym tłumaczeniem angielskiego *depth* — przyp. tłum.

rozpoczyna nowego akapitu i domyślnie posiada szerokość 0.4 punkta zaś wysokość taką samą jak wiersz, w którym jest wstawiana. Ilustruje to przykład:

```
Nieco tekstu przed linią pionową
\vrule\
i nieco poza nią.
```

co daje

Nieco tekstu przed linią pionową | i nieco poza nią.

▷ Ćwiczenie 9.2 Złóż trzy linie poziome 15 pt jedna nad drugą, długości 3 cali i 1 cal od lewego marginesu.

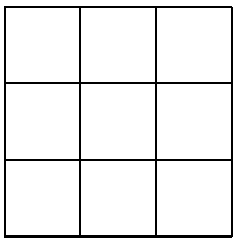
Chociaż na ogół traktujemy *hrule* i *vrule* jako linie poziome i pionowe, interpretacja taka nie jest to konieczna. Na przykład:

```
\noindent
Nazwisko: \vrule height .4pt depth 0 pt width 3 in
```

daje

Nazwisko: _____

▷ Ćwiczenie 9.3 Wykonaj szachownicę (każda kratka ma 1 cm kwadratowy):



9.4 Pudełka wewnątrz pudełek

Przy omawianiu postaci wiersza dowiedzieliśmy się, że pudełka (*vbox-y* i *hbox-y*) są obiektami mogącymi wykazywać nadmiary bądź niedomiary. Obecnie omówimy pudełka nieco bardziej szczegółowo. Mogą być one ustawiane zarówno w pionie jak i w poziomie, pozwalając na różnorodny układ tekstu na stronie.

Pudełko poziome formujemy za pomocą `\hbox{...}`. Materiał ujęty w klamrach zostaje umieszczony w pudełku, które staje się od tej chwili niepodzielną jednostką (w szczególności zawartość takiego pudełka nie może zostać przełamana na dwa wiersze). Istnieje możliwość określania rozmiaru pudełek; `\hbox to 5 cm{zawartość pudełka}` utworzy pudełko szerokości dokładnie pięciu centymetrów, zawierające skład: „zawartość pudełka". Łatwo w ten sposób o nadmiary lub niedomiary[3]. Niepodanie rozmiaru tworzy pudełko szerokości dokładnie wymaganej przez składany tekst.

TEXbook: 64–66

Analogicznie, pudełko pionowe formujemy za pomocą `\vbox{...}`. Ten rodzaj pudełek posiada interesującą cechę: jeśli zawiera wewnątrz inne pudełka, zostaną one umieszczone jedno nad drugim i będą składane jako jeden element. Podobnie *hbox* może zawierać pudełka składane tym razem w poziomym szeregu. Umieśćmy trzy pudełka poziome w jednym pudełku pionowym:

```
\vbox{
      \hbox{Zawartość pudełka 1}
      \hbox{Zawartość pudełka 2}
      \hbox{Zawartość pudełka 3}
      }
```

otrzymując

Zawartość pudełka 1
Zawartość pudełka 2
Zawartość pudełka 3

A teraz utwórzmy inne pudełko pionowe:

```
\vbox{
      \hbox{Zawartość pudełka 4}
      \hbox{Zawartość pudełka 5}
      }
```

[3] W zależności od wielkości użytej czcionki „zawartość pudełka" nie zmieści się w dostępnym wymiarze; przeciwnie, mając zbyt wiele miejsca — daje w składzie nieprzyjemne odstępy — przyp. tłum.

Jeśli oba nasze *vbox-y* umieścimy w pudełku poziomym, spowoduje to składanie ich w jednym wierszu. Innymi słowy

```
\hbox{
     \vbox{
           \hbox{Zawartość pudełka 1}
           \hbox{Zawartość pudełka 2}
           \hbox{Zawartość pudełka 3}
           }
     \vbox{
           \hbox{Zawartość pudełka 4}
           \hbox{Zawartość pudełka 5}
           }
     }
```

da w składzie

Zawartość pudełka 1
Zawartość pudełka 2 Zawartość pudełka 4
Zawartość pudełka 3 Zawartość pudełka 5

Zauważ, że podstawa obu pudełek pionowych znajduje się na jednej linii i są one oddzielone tylko małym odstępem. Dodatkowy odstęp, np. wielkości jednego centymetra, osiągniemy wstawiając pomiędzy *vbox-y* zapis `\hskip 1 cm`. Wyrównanie wierzchołków pudełek pionowych w linii poziomej możliwe jest po zastąpieniu `\vbox` przez `\vtop`. Wykonanie tych dwóch zmian da w efekcie:

Zawartość pudełka 1 Zawartość pudełka 4
Zawartość pudełka 2 Zawartość pudełka 5
Zawartość pudełka 3

Kombinacja pudełek (*vbox* i *hbox*) oraz linii (*vrule* i *hrule*) pozwala uzyskać tekst otoczony ramką. Jak to osiągnąć? Jednym ze sposobów jest umieszczenie w *hbox* materiału zawartego między pionowymi liniami (*vrule*). Całość poprzedzamy i kończymy liniami poziomymi (*hrule*) i wkładamy do pudełka pionowego. Przykładowo:

```
\vbox{
      \hrule
      \hbox{ \vrule Tekst w ramce \vrule}
      \hrule
     }
```

da w efekcie

Tekst w ramce

Tekst jest wprawdzie ściśnięty między liniami ramki, ale TEX wykonał dokładnie to co mu zlecono. Lepszy efekt osiągniemy przy pomocy komendy `\strut`, która nieco podwyższa i pogłębia pudełko *hbox*:

Tekst w ramce

▷ Ćwiczenie 9.4 Zastosuj powyższą metodę do tekstu umieszczonego w osi pudełka sięgającego lewego i prawego marginesu.

▷ Ćwiczenie 9.5 Złóż następujący kwadrat magiczny:

6	1	8
7	5	3
2	9	4

▷ Ćwiczenie 9.6 Napisz makro `\boxtext#1{...}`, które otoczy ramką zawarty w klamrach tekst. Przetestuj makro pisząc zdanie, w którym co drugi wyraz będzie w ramce. Nie bardzo jestem pewien po co to robić, bo rezultat jest nieco dziwny. Zauważ wyrównanie podstawy ramek z linią pisma.

Przesuwanie pudełek w prawo, w lewo, w dół i w górę jest bardzo proste. `\vbox` przesuniemy w prawo o jeden cal przy pomocy `\moveright 1 in \vbox{...}`. Przesunięcie w lewo wymaga użycia `\moveleft`. Analogicznie, `\hbox` przesuwamy w górę i w dół używając odpowiednio: `\raise` lub `\lower`.

▷ Ćwiczenie 9.7 Zmień makro `\boxtext` z poprzedniego ćwiczenia tak, aby całość tekstu znalazła się w jednej linii. Domyślna głębokość pudełka zawierającego `\strut` wynosi 3.5 punkta, a grubość `\hrule` — 0.4 punkta. W efekcie nasze zdanie powinno wyglądać następująco:
Nie bardzo jestem pewien po co to robić, bo rezultat jest nieco dziwny.

Na zakończenie wspomnimy o możliwości wypełniania pudełka linią poziomą bądź kropkowaną. Trik polega na użyciu w ramach *hbox* komendy `\hrulefill` albo `\dotfill`.

```
\hbox to 5 in{Zaczynamy\hrulefill 1}
\hbox to 5 in{Wszelkie znaki, duże i małe\hrulefill 9}
\hbox to 5 in{Rzeczy nabierają kształtu\hrulefill 16}
\hbox to 5 in{Nie straszna nam matematyka!\hrulefill 28}
```

daje w składzie

Zaczynamy 1
Wszelkie znaki, duże i małe 9
Rzeczy nabierają kształtu 16
Nie straszna nam matematyka! 28

Jeśli `\hrulefill` zastąpimy przez `\dotfill` otrzymamy:

Zaczynamy 1
Wszelkie znaki, duże i małe 9
Rzeczy nabierają kształtu 16
Nie straszna nam matematyka! 28

Indeks

Poniżej zamieszczono alfabetyczne zestawienie sekwencji sterujących, które występują w niniejszym podręczniku. Kompletny spis, a także bardziej dokładne omówienie znajdziesz w **The TEXbook**.

Symbole sterujące

Słowa sterujące

Złożono TEX-em 9 kwietnia 1990 r., o godz. 16:55