Rýchlokurz LaTeXu

Peter Kostolányi

Verzia z 1. septembra 2014

Obsah

Úvod

Cieľom tohto krátkeho textu je v čo možno najkratšom čase oboznámiť čitateľa so základmi práce s LaTeXom – systémom na prípravu dokumentov bežiacim nad typografickým systémom TeX. Odhliadnuc od vysokej typografickej kvality výstupných dokumentov sa sila LaTeXu ukazuje najmä pri tvorbe dokumentov obsahujúcich väčšie množstvo matematických symbolov a vzorcov. Aj keď viaceré „bežné“ editory obsahujú nástroje umožňujúce vkladanie „matematiky“ do textu, ich použitie je často pomerne pracné a namáhavé a výsledok nie je vždy plne uspokojujúci. Znalosť aspoň základov LaTeXu tak môže napríklad pri písaní záverečnej práce ušetriť jej autorovi značné množstvo času, námahy, aj nervov. Navyše, LaTeX je v súčasnosti nepísaným štandardom pre vedecké publikácie (minimálne) v oblastiach matematiky, (teoretickej) informatiky a fyziky.

Východiskom pri tvorbe tohto textu boli predovšetkým potreby úvodného kurzu formálnych jazykov a automatov pre informatikov. Súčasťou cvičení k tomuto predmetu je aj vypracovanie väčšieho množstva domácich úloh, ktoré môžu slúžiť aj ako vhodné cvičenia na zvládnutie základov LaTeXu. Táto možnosť môže byť pre typického študenta zaujímavá hneď z dvoch dôvodov:

- Jediný spoľahlivý spôsob, ako sa LaTeX naučiť, je s ním pravidelne pracovať. Práve odovzdávanie riešení domácich úloh spísaných v LaTeXu môže byť vhodným spôsobom, ako získať skúsenosti, ktoré neskôr môžu prísť vhod napríklad pri písaní bakalárskej práce.
- Čítanie dokumentov vytvorených v LaTeXu je niekoľkonásobne jednoduchšie, než čítanie originálnych rukopisov. Študenti odovzdávajúci riešenia spísané v LaTeXu tak môžu prispieť k rýchlejšiemu vyhodnoteniu domácich úloh a predovšetkým majú jedinečnú šancu vykonať dobrý skutok.

Aj keď niektoré časti tohto textu sú cielené predovšetkým na špecifické potreby cvičení z formálnych jazykov a automatov, okruh potenciálnych čitateľov sa na študentov zapísaných na tento predmet neobmedzuje (taký bol aspoň zámer).

Predkladaný text rozhodne nemožno považovať za učebnicu LaTeXu v klasickom slova zmysle. Na rozdiel od bežných úvodných textov sa tu totiž hlavný dôraz nekladie na rôzne vymoženosti, ktoré systém LaTeX ponúka, ale predovšetkým na základnú filozofiu systému, na jeho inštaláciu a vysádzanie prvých jednoduchých dokumentov. Sú to totiž práve tieto prvé kroky, ktoré často od LaTeXu jeho potenciálnych používateľov odrádzajú. Záujemcov o preniknutie hlbšie možno odkázať na množstvo dostupných zdrojov, napríklad [10], [9], či [16].

Autor ešte považuje za potrebné zdôrazniť, že sa za žiadnych okolností nemôže považovať za odborníka na LaTeX alebo počítačovú typografiu – jeho znalosti sú skôr základné a čisto používateľské. Informácie uvedené v tomto texte tak treba brať len ako praktické rady na zvládnutie nutných základov, nie ako odborný výklad problematiky. Naopak, text určite obsahuje množstvo nepresností a nabádania na zlé návyky, za čo sa autor všetkým TeXnikom a TeXpertom vopred ospravedlňuje.

P.K.

Kapitola 1

Inštalácia a prvé kroky

Po prečítaní tejto úvodnej kapitoly by mal byť čitateľ schopný na svojom počítači vytvoriť s použitím LaTeXu prvý jednoduchý dokument („Hello World!“). Obsahovou náplňou kapitoly sú predovšetkým základné pojmy súvisiace so systémom LaTeX, prehľad vybraných TeXových distribúcií a náčrt základných princípov tvorby dokumentov v LaTeXu.

1.1 TeX a LaTeX (základné pojmy a stručné dejiny)

TeX (vyslovuje sa „*tech*“) je počítačový typografický systém, ktorý začal v roku 1977 vyvíjať americký informatik Donald Knuth. Hlavným Knuthovým impulzom pre tento počin bola predovšetkým zhoršujúca sa typografická kvalita postupne vydávaných častí jeho knihy *The Art of Computer Programming* v kombinácii s potenciálom, ktorý videl v počítačovej typografii. Knuthovou ambíciou bolo predovšetkým vytvoriť systém umožňujúci tvorbu dokumentov s vysokou typografickou kvalitou (a to aj pri sádzaní matematických vzorcov), ktorý navyše kedykoľvek v budúcnosti a na prakticky ľubovoľnej počítačovej architektúre vysádza z rovnakého vstupného súboru rovnaký výstupný dokument.

Posledná väčšia zmena sa v systéme udiala v roku 1989, keď bola zverejnená verzia TeX 3.0.[1] Nové verzie zverejnené po roku 1989 sa od verzie 3.0 líšia len v detailoch (väčšinou ide o opravy chýb, ktoré však v systéme takmer absentujú) a ich čísla konvergujú k hodnote π. Momentálne najnovšia verzia má číslo 3.1415926 [4]. Posledná a definitívna verzia má byť zverejnená po Knuthovej smrti a má mať číslo π.

Samotný TeX obsahujúci okolo 300 príkazov však nie je pre koncového používateľa úplne najpohodlnejším nástrojom. Preto sa k systému TeX pridávajú sady makier, ktoré majú prácu s ním uľahčiť. Donald Knuth sám prišiel s formátom plainTeX, ktorý pridáva ďalších okolo 600 príkazov. Jeho verzie tiež konvergujú k π a v súčasnosti je na verzii 3.141592653 [4].

Aj keď niektorí odborníci preferujú používanie plainTeXu a majú na to svoje dôvody [12], medzi „bežnými“ používateľmi je v súčasnosti najrozšírenejšou voľbou LaTeX (vyslovuje sa „*latech*“ alebo „*lejtech*“).[2] Hlavnou výhodou LaTeXu je, že odpadá veľká časť starostí s manuálnym formátovaním, keďže súčasťou LaTeXu je viacero preddefinovaných štýlov, ktoré robia veľkú časť tejto práce automaticky. Makrá, ktoré sú v LaTeXu k dispozícii, umožňujú používateľovi mimoriadne jednoduché udržiavanie logickej štruktúry dokumentu, ba ho k nemu priam nútia. Nezanedbateľnou výhodou LaTeXu je aj množstvo balíkov, s ktorými prichádza. Tie často robia veľmi jednoduchými aj inak pomerne zložité typografické úlohy (od vypisovania záznamov šachových partií vrátane obrázkov stavu šachovnice, cez vykresľovanie hudobných nôt, až po tvorbu prezentácií).

[1] To však netreba vnímať ako známku zastaranosti systému, ale naopak, ako známku jeho veľkej stability. Postupné zmrazovanie vývoja systému (ktorý má v súčasnosti iba podobu odhaľovania a opravovania minimálne sa vyskytujúcich chýb) je úmyselné, keďže snahou projektu je zaviesť štandard, ktorý bude kedykoľvek v budúcnosti fungovať rovnako. Možno konštatovať, že tento zámer je de facto naplnený už v súčasnosti.

[2] Slovo „*TEXovať*“ sa v bežnej reči používa väčšinou práve pre označenie práce s LaTeXom.

Nevýhodou LATEXu oproti plainTEXu však je, že jeho vývoj je ešte stále v plnom prúde. Momentálne najnovšia verzia LATEXu je LATEX 2_ε. Verzia LATEX 3, vyvíjaná dlhé roky, ešte stále nie je k dispozícii.

Spomeňme ešte, že viacero novších TEXových distribúcií už ako základný typografický systém (používaný napr. LATEXom) nepoužíva priamo Knuthov TEX, ale jeho rozšírenie pdfTEX, ktoré vzniklo na Masarykovej univerzite v Brne [3]. To sú však detaily, ktoré začínajúceho používateľa nemusia príliš zaujímať.

1.2 Distribúcie TEXu

Dostať sa k TEXu resp. LATEXu je pomerne jednoduché – ide totiž o slobodný a otvorený softvér a pod slobodnými licenciami je dostupná aj väčšina LATEXových balíkov a ďalších súčastí typických TEXových distribúcií. Túto filozofiu začal raziť Donald Knuth od samotných počiatkov TEXu, keď ešte pojem slobodného a otvoreného softvéru nebol známy. Zdrojový kód TEXu bol vždy voľne k dispozícii a Knuthova licencia umožňovala robiť s týmto kódom v zásade čokoľvek; s obmedzením, že pod názvom TEX je možné upravený program zverejniť iba v prípade, že prejde sadou Knuthom navrhnutých testov. Takýto prístup mal za následok, že v súčasnosti existuje funkčná a kvalitná verzia TEXu v zásade pre každý čo i len trochu rozšírený operačný systém [10].

V súčasnosti je k dispozícii mnoho TEXových distribúcií, typicky obsahujúcich TEX, LATEX, množstvo ďalších variantov týchto programov a ďalšieho softvéru nejakým spôsobom súvisiaceho s TEXom, veľké množstvo LATEXových balíkov (obsahujúcich rôzne makrá umožňujúce jednoducho robiť veci od výmyslu sveta) a nástroje na správu celej distribúcie. Rovnako ako ich základné komponenty, patrí aj väčšina TEXových distribúcií pod slobodný a otvorený softvér, takže sú prístupné zdarma.

Zakončime tento oddiel krátkym prehľadom často využívaných možností na získanie TEXovej distribúcie:

- Nejaká distribúcia TEXu býva často už priamou súčasťou inštalácií Linuxu.
- Pod MS Windows je pravdepodobne najznámejšou distribúciou už dlhé roky **MiKTEX**. Existuje aj vo verzii MiKTEX Portable, ktorá umožňuje používať TEX napríklad z DVD alebo USB kľúča. Najnovšiu verziu tejto distribúcie možno nájsť na projektovej stránke `http://miktex.org/`.
- Na MiKTEXu je založená aj ďalšia distribúcia TEXu pre Windows – **proTEXt**. Možno ju získať zo stránky `http://www.tug.org/protext/`.
- Multiplatformovou distribúciou a pravdepodobne najznámejšou voľbou pre Unixové systémy je **TEX Live**. Beží aj pod MS Windows a OS X. Podobne ako MiKTEX sa dá používať aj z DVD alebo USB kľúča. Je založená na kedysi populárnom projekte teTEX, určenom najmä pre Unixové systémy. Najnovšiu verziu tejto distribúcie možno získať zo stránky `http://www.tug.org/texlive/`.
- Pre OS X býva často odporúčanou distribúciou **MacTEX**, ktorý možno získať zo stránky `http://tug.org/mactex/`.

1.3 Inštalácia TEXovej distribúcie

Inštalácia väčšiny TEXových distribúcií je pomerne jednoduchá záležitosť a ničím podstatným sa nelíši od inštalácie ľubovoľného iného bežného softvéru.

Preto iba v stručnosti opíšeme inštaláciu distribúcie MiKTEX pod MS Windows. Je možné vybrať si medzi dvoma typmi inštalácie: pri prvom (*Basic Installer*) sa nainštalujú iba najčastejšie používané súčasti distribúcie a ostatné sa automaticky doinštalujú neskôr v prípade, že to bude potrebné. Druhý typ inštalácie (*Net Installer*) umožňuje nainštalovať kompletnú distribúciu

MiKTEX. Pri výbere tejto možnosti sa z projektovej stránky stiahne iba neveľký inštalátor. Pomocou neho je potom najprv potrebné z vybraného serveru stiahnuť kompletnú inštaláciu MiKTEXu na disk (tieto súbory potom možno použiť pri prípadných ďalších inštaláciách bez nutnosti ich opätovného sťahovania). Následne je potrebné z týchto súborov MiKTEX nainštalovať.

Celý tento proces je úplne štandardný a pravdepodobne ho nie je potrebné viac rozoberať. Podrobne je opísaný napr. na `http://docs.miktex.org/2.9/manual/installing.html`. Po jeho dokončení by malo byť všetko pripravené na použitie tak, ako je to uvedené v nasledujúcich oddieloch.

1.4 Základné princípy práce s LATEXom

Mnoho potenciálnych používateľov s LATEXom končí hneď vzápätí po inštalácii TEXovej distribúcie. Na rozdiel od väčšiny známych textových editorov typu MS Word totiž nie je okamžite jasné, akým spôsobom pri tvorbe prvého dokumentu postupovať. Základnú filozofiu práce s LATEXom by mal čitateľ pochopiť po prečítaní tohto oddielu.

Jeden rozdiel odlišuje LATEX od väčšiny široko používaných textových editorov viac, než ostatné. Kým väčšina editorov typu MS Word sú tzv. WYSIWYG editory,[3] v ktorých je tvorba dokumentov interaktívna, spôsob práce s LATEXom je diametrálne odlišný. Postup tvorby dokumentov je tu skôr podobný bežnému programovaniu, kde je najprv potrebné napísať zdrojový kód, ktorý neskôr kompilátor preloží do strojového kódu a vygeneruje spustiteľný súbor. Aj práca s LATEXom prebieha v podobných fázach: najprv je potrebné vytvoriť zdrojový súbor (s príponou `.tex`) a ten sa potom, zhruba povedané, za použitia LATEXu alebo niektorého z jeho variantov preloží do výsledného dokumentu (spravidla vo formáte PDF alebo PostScript). Na napísanie zdrojového súboru možno použiť ľubovoľný textový editor vytvárajúci obyčajné textové súbory.

Na prvý pohľad sa táto vlastnosť LATEXu môže zdať veľkou nevýhodou oproti WYSIWYG editorom, kde používateľ vidí výsledok okamžite po vykonaní ľubovoľnej zmeny a nemusí sa učiť žiaden špeciálny jazyk. Ukazuje sa však, že opak je pravdou. Pri zložitejších úlohách, ako sú napríklad sadzba matematických vzorcov, udržiavanie logickej štruktúry dokumentov, či vkladanie krížových odkazov je použitie „bežných“ WYSIWYG editorov veľmi nemotorné a ťažkopádne, kým vyjadrenie v zdrojovom kóde LATEXu je jednoduché a priamočiare. Analógiou tu môže byť napríklad tvorba webových stránok, kde sa použitie WYSIWYG nástrojov tiež môže zdať na prvý pohľad jednoduchšie, než písanie HTML kódu. Čitateľ so skúsenosťami s tvorbou zložitejších stránok však určite potvrdí, že použitie WYSIWYG nástrojov je pre takéto projekty omnoho ťažkopádnejšie a v mnohých prípadoch priam až nemožné.[4]

Preklad zo zdrojového kódu do výsledného dokumentu možno urobiť viacerými spôsobmi. Klasickou metódou je použitie programu LATEX, pomocou ktorého sa za použitia TEXu[5] vygeneruje zo zdrojového súboru určitý medzivýstup. Z neho sa potom pomocou ďalších programov (ktoré sú súčasťou bežných TEXových distribúcií) dá vygenerovať konečný výstup v podobe PDF súboru alebo súboru vo formáte PostScript. Novšou a rýchlejšou metódou (ktorú možno pre bežných používateľov odporúčať) je použitie programu pdfLATEX, ktorý zo zdrojového súboru priamo vygeneruje (s použitím pdfTEXu) výstup vo formáte PDF. Obidva prístupy sú detailnejšie opísané nižšie.

Všetky programy spomínané v predchádzajúcom odstavci sú konzolové aplikácie, ktoré ako parameter berú názov vstupného súboru. Dajú sa teda spúšťať rôznymi spôsobmi (čitateľ určite niektoré z nich ľahko vymyslí aj sám), no najbežnejšie a najpraktickejšie je pravdepodobne ich volanie pomocou príkazového riadku (*command line*). Väčšina inštalátorov pridá adresár s binárnymi súbormi danej distribúcie[6] do systémovej premennej `PATH`, čo znamená, že programy ako LATEX alebo pdfLATEX možno volať z príkazového riadku v ľubovoľnom pracovnom adresári. (Detaily sú samozrejme do istej miery závislé od konkrétnej platformy.)

[3] WYSIWYG = angl. *What You See is What You Get.*

[4] Bez manuálnych zásahov do HTML kódu vygenerovaného daným WYSIWYG nástrojom.

[5] Alebo v novších distribúciách už pdfTEXu.

[6] Napr. pri MiKTEXu to je adresár `⟨koreňový adresár MiKTEXu⟩\miktex\bin`.

1.4.1 Klasická metóda (LaTeX)

Cieľom tohto pododdielu je priblížiť čitateľovi vyššie spomínanú „klasickú“ metódu s použitím programu LaTeX. V nasledujúcom predpokladáme, že už máme k dispozícii hotový korektný zdrojový súbor (vytvorenie takéhoto súboru je opísané nižšie; pre účely tohto pododdielu môže čitateľ použiť napríklad niektorý z množstva ukážkových zdrojových súborov, ktoré bývajú súčasťou väčšiny TeXových distribúcií), nazvime ho napríklad `pokus.tex`. V nasledujúcich bodoch je zhrnutý postup, ako z tohto súboru vytvoriť výsledný dokument vo formáte PDF resp. PostScript.

1. Otvoríme príkazový riadok a nastavíme sa do adresára, v ktorom máme náš zdrojový súbor `pokus.tex` uložený.

2. Do príkazového riadku zadáme nasledujúci príkaz: `latex pokus`. Vidíme, že v našom adresári pribudli tri novovytvorené súbory: `pokus.dvi`, `pokus.aux` a `pokus.log`.

3a. Ak chceme vytvoriť dokument vo formáte PDF, do príkazového riadku zadáme nasledujúci príkaz: `dvipdfm pokus`. To nám zo súboru `pokus.dvi` vytvorí výsledný PDF súbor `pokus.pdf`.

3b. Ak chceme vytvoriť dokument vo formáte PostScript, do príkazového riadku zadáme takýto príkaz: `dvips pokus`. Výsledkom bude novovytvorený súbor `pokus.ps`.

Je dobré zvyknúť si aplikovať bod 2 dvakrát prípadne až trikrát po sebe (alebo si napísať jednoduchý skript, ktorý to urobí za nás). Dôvodom je skutočnosť, že LaTeX je jednoprechodový program, t.j. výstup generuje počas jedného sekvenčného prechodu zdrojovým súborom. Niektoré úlohy (ako napríklad tvorba obsahu alebo krížových odkazov) sa však zjavne pri takomto jednom prechode nedajú vykonať správne. LaTeX to rieši tak, že si potrebné informácie zapisuje do pomocných súborov, pričom tieto informácie využije pri jeho ďalšom volaní. Obsahy a krížové odkazy sú tak pri druhom volaní už vygenerované správne (za predpokladu, že sa medzi dvoma volaniami LaTeXu nezmenil zdrojový súbor).

V krátkosti sa ešte pristavme pri jednotlivých bodoch horeuvedeného postupu. Volanie `latex pokus` zavolá program LaTeX na vstupe `pokus.tex` – je teda dôležité, aby všetky zdrojové súbory mali príponu `.tex`. LaTeX pomocou typografického systému TeX (resp. pdfTeX) vytvorí súbor `pokus.dvi`. Ide o binárnu špecifikáciu výsledného dokumentu, ktorá je nezávislá od konkrétneho grafického formátu, zobrazovacieho zariadenia, či tlačiarne.[7] Súbor vo formáte DVI nie je určený na čítanie, ide len o medzivýstup určený na ďalšie spracovanie (konverzia do iného formátu, tlač, atď.). Napriek tomu väčšina TeXových distribúcií obsahuje aj program, ktorý dokáže DVI súbory zobrazovať.[8] DVI súbory v sebe nemajú zabudované fonty.

Súbor `pokus.aux` obsahuje pomocné informácie pre samotný LaTeX a súbor `pokus.log` obsahuje zachované všetky hlášky, ktoré LaTeX vypísal počas spracovania vstupu.

Program `dvipdfm` slúži na konverziu z DVI do PDF, program `dvips` na konverziu z DVI do PostScriptu.

1.4.2 Rýchla metóda (pdfLaTeX)

Ako už bolo avizované, v súčasnosti existuje aj rýchlejšia metóda, spočívajúca v použití programu pdfLaTeX, pomocou ktorej možno vygenerovať zo vstupného súboru priamo dokument vo formáte PDF. Túto metódu možno v zásade odporúčať: medzi LaTeXom a pdfLaTeXom síce existujú drobné rozdiely, ale nejde o nič skutočne zásadné. Používateľ si tak môže ušetriť jeden zbytočný krok, keďže v súčasnosti asi jediná vec, ktorú väčšina používateľov robí so súborom DVI, je jeho konverzia do PDF.

Metódu s použitím pdfLaTeXu možno zhrnúť v nasledujúcich dvoch bodoch (opäť predpokladáme existenciu korektného zdrojového súboru `pokus.tex`):

[7] Prípona `.dvi` je skratkou z *Device Independent File Format*.
[8] Napríklad súčasťou MiKTeXu je program Yap – *Yet Another Previewer*.

1. Otvoríme príkazový riadok a nastavíme sa do adresára, v ktorom máme náš zdrojový súbor `pokus.tex` uložený.

2. Do príkazového riadku zadáme nasledujúci príkaz: `pdflatex pokus`. V našom adresári pribudnú tri súbory: `pokus.aux` a `pokus.log` (ktoré majú rovnakú funkciu ako pri prvej metóde) a výsledný PDF súbor `pokus.pdf`.

Opäť je dobré zvyknúť si aplikovať bod 2 dvakrát alebo trikrát (z rovnakých dôvodov, ako boli uvedené pri predchádzajúcej metóde).

1.5 Prvý dokument („Hello World!“)

Po prečítaní predošlého oddielu by už mal čitateľ ovládať postup vygenerovania výsledného dokumentu z hotového zdrojového súboru. Teraz opíšeme postup, ako takýto jednoduchý zdrojový súbor vytvoriť. Nasledujúci kód je zdrojovým kódom dokumentu, v ktorom je vypísaný text „`Hello, World!`“. Mal by byť uložený v textovom súbore s príponou `.tex` a s kódovaním ASCII.[9] Čitateľ sa môže pokúsiť z tohto súboru vytvoriť výsledný dokument spôsobom uvedeným vyššie.

```
\documentclass{article}
\begin{document}
Hello, World!
\end{document}
```

Vysvetlime si teraz jednotlivé časti uvedeného kódu. Každý zdrojový súbor v LaTeXu má nasledujúcu štruktúru:

```
\documentclass{typ dokumentu}

% preambula

\begin{document}

% telo dokumentu

\end{document}
```

Príkaz `\documentclass` špecifikuje typ (triedu) dokumentu. Najčastejšie používané triedy dokumentov sú nasledujúce:

- `article` – krátke články,
- `report` – dlhšie dokumenty, pre jednostrannú tlač,
- `book` – knihy, pre obojstrannú tlač.

Začiatok a koniec samotného dokumentu sa v jeho zdrojovom kóde vyznačuje povinnými príkazmi `\begin{document}` a `\end{document}`, ktoré nesmú chýbať v žiadnom LaTeXovom zdrojovom kóde. Samotný obsah dokumentu sa píše medzi tieto dva príkazy a zodpovedajúca časť zdrojového kódu sa nazýva *telo dokumentu*. Časť zdrojového kódu medzi príkazom `\documentclass` a začiatkom samotného dokumentu sa nazýva *preambula*. Do preambuly sa zvyknú písať rôzne príkazy, napr. na načítanie balíkov rozširujúcich funkcionalitu LaTeXu, rozličné nastavenia, definície príkazov a podobne.

[9] Fungovať by ale malo napr. aj kódovanie CP1250 (stredoeúropske kódovanie pod MS Windows), ktoré je pri použití výhradne znakov z ASCII od ASCII nerozlíšiteľné.

1.6 Tvorba dokumentov v slovenčine

Vysvetlili sme si postup, ako v LaTeXu vytvoriť dokument obsahujúci nejaký jednoduchý text. Menší problém s vyššie uvedeným príkladom však je, že prípadné použitie rôznych slovenských znakov s diakritikou by nefungovalo. Na „spojazdnenie“ slovenčiny v LaTeXu je potrebné náš úvodný príklad o niečo rozšíriť a *zdrojový súbor uložiť v kódovaní UTF-8.*[10]

```
\documentclass{article}
\usepackage[slovak]{babel}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\begin{document}
Mäkčene a dĺžne sú v slovenčine dôležité.
\end{document}
```

Ako možno vidieť, rozdiel spočíva (okrem zmeny kódovania súboru so zdrojovým kódom) v pridaní troch príkazov `\usepackage` do preambuly. Príkaz `\usepackage` sa používa na načítanie balíkov rozširujúcich funkcionalitu LaTeXu – v tomto prípade teda načítavame balíky `babel` (s parametrom `slovak`), `fontenc` (s parametrom `T1`) a `inputenc` (s parametrom `utf8`).

Balík `babel` je de facto základným kameňom internacionalizácie LaTeXu. Pre rôzne jazyky obsahuje jednak makrá pre v nich sa vyskytujúce znaky, ďalej pravidlá na rozdeľovanie slov (ktoré v novších verziách `babel`u fungujú relatívne dobre aj pre slovenčinu), preklady rôznych automaticky vypisovaných úsekov textu (ako napr. „Obsah“, „Kapitola“, dátumy a pod.), ako aj ďalšie záležitosti potrebné na plnohodnotnú prácu v najrôznejších jazykoch.

Balík `fontenc` umožňuje určovať kódovanie symbolov použitých vo výslednom dokumente na reprezentáciu znakov. Východzia voľba `OT1` kóduje symboly pomocou 128 bitov. Pri takejto reprezentácií nie je výsledok pri vykresľovaní niektorých slovenských znakov uspokojujúci. Preto je v horeuvedenom príklade zvolené kódovanie `T1`, ktoré na reprezentáciu symbolov používa 256 bitov a výsledok je aj pri slovenských znakoch uspokojivý.

Nakoniec, balík `inputenc` umožňuje nastaviť kódovanie textového súboru so zdrojovým kódom, ktoré by malo zodpovedať jeho reálnemu kódovaniu. Nastavenie tohto kódovania na `utf8` vlastne iba LaTeX „upozorní“ na skutočnosť, že súbor so zdrojovým kódom má kódovanie UTF-8. Použitie kódovania UTF-8 má tú výhodu, že slovenské znaky možno do zdrojového kódu zadávať priamo z klávesnice – pri kódovaní ASCII by bolo potrebné na každý znak s diakritikou použiť nejaké špeciálne makro z balíku `babel`, čo by používateľovi podstatne skomplikovalo prácu. Ďalším kódovaním, ktoré možno použiť, je kódovanie CP1250, používané v systéme MS Windows ako východzie kódovanie stredoeurópskych jazykov. V takom prípade treba zadať balíku `inputenc` ako parameter `cp1250`. Odporúčané je však kódovanie UTF-8 – keď už nič iné, má to tú výhodu, že okrem bežných slovenských znakov možno priamo zadávať aj znaky o niečo exotickejšie.[11]

V starších textoch o LaTeXu sa možno dočítať aj o inej metóde sadzby českých resp. slovenských dokumentov. Namiesto použitia kombinácie LaTeX + `babel` sa pri tejto metóde používala akási česko-slovenská verzia LaTeXu, tzv. CSLaTeX. Ide však o metódu z doby, keď medzinárodná podpora v LaTeXu nebola na takej úrovni ako dnes a v súčasnosti je aj oficiálne jej tvorcami označená za zastaranú [11].[12]

1.7 Ďalší užitočný softvér

Pri práci s LaTeXom sa občas môže zísť softvér, ktorý väčšinou nebýva priamou súčasťou TeXových distribúcií. Nasledujúci zoznam obsahuje niekoľko najbežnejších príkladov takéhoto softvéru:

- Nástroje na prehliadanie a prácu s formátom PostScript, napr. GhostScript a GSView.

[10] Možnosť zmeny kódovania na UTF-8 by mal ponúkať každý čo i len trochu pokročilejší textový editor (bez ohľadu na operačný systém).

[11] Občas sa ako parameter balíku `inputenc` vyskytuje aj `utf8x`, čo je podpora o niečo širšej škály znakov z UTF-8. V súčasnosti sa však odporúča používať iba v prípade, že s niektorými znakmi nastanú pri obyčajnom `utf8` problémy.

[12] Použitie balíku `babel` však tiež nie je úplne bez problémov, keďže jeho nová česká a slovenská verzia obsahuje zopár chýb. Jedna zo známejších chýb sa napríklad prejaví pri používaní príkazu `\cline`.

- Nástroje na tvorbu vektorovej grafiky (keďže PDF je vektorový formát, použitie vektorovej grafiky je lepšou voľbou, než použitie rastrovej grafiky). Známym príkladom takéhoto softvéru je napr. Inkscape, trochu „vedeckejšie“ obrázky sa dajú tvoriť pomocou programu Ipe. Samotné TEXové distribúcie naviac zvyknú obsahovať napríklad Metapost alebo rôzne LATEXové balíky na tvorbu vektorovej grafiky priamo z LATEXového kódu (napríklad TikZ).
- Keďže LATEX nepodporuje všetky obrázkové formáty, zídu sa nástroje na ich konverziu. Napríklad softvér ImageMagick umožňuje vykonať takúto konverziu pomocou jediného príkazu zadaného do príkazového riadku (a zvládne toho omnoho viac).
- Textový editor. Je ich k dispozícii množstvo a konkrétna voľba je v zásade vecou osobného vkusu. Mnohé textové editory podporujú aj zvýrazňovanie LATEXovej syntaxe, iné sú dokonca určené výhradne pre písanie kódu v LATEXu a niektoré by sa dokonca dali nazvať kompletnými vývojovými prostrediami s možnosťou priamo vygenerovať výsledný dokument a podobne.

Kapitola 2

Základy práce s LaTeXom

Táto kapitola stručne vysvetľuje niektoré základné prvky LaTeXu, ktorých zvládnutie umožní čitateľovi vytváranie prvých jednoduchých avšak naozajstných dokumentov (napríklad riešení domácich úloh). Prezentovaný materiál si však rozhodne nekladie za cieľ byť úplný, ani postačujúci na všetky bežné účely – za žiadnych okolností nemôže nahradiť učebnice LaTeXu v bežnom slova zmysle.

Pre získanie solídnych praktických základov LaTeXu je vhodné (hoci aj namiesto čítania tejto kapitoly) siahnuť napríklad po úvodnom texte [10], ktorý je voľne prístupný na internete a taktiež aj súčasťou viacerých TeXových distribúcií. Existuje aj v množstve prekladov – pomerne čerstvý je český preklad [9], slovenský preklad [8] je už o niečo zastaranejší. Pokročilejšie znalosti možno získať napríklad z knihy o LaTeXu [16], ktorá je súčasťou projektu Wikibooks. Nadšenci by tiež mali byť oboznámení s Knuthovou učebnicou TeXu [5].

2.1 Bežný text

Ako sa bolo možné presvedčiť už v predchádzajúcej kapitole, na vysádzanie bežného textu je v zásade postačujúce napísať ho do tela dokumentu. Aj tu je však niekoľko dôležitých vecí, o ktorých treba vedieť.

Predovšetkým je dobré venovať určitú pozornosť tomu, ako LaTeX narába s tzv. bielymi znakmi (angl. *whitespaces*). Medzera aj tabulátor, ako aj ich ľubovoľná neprázdna postupnosť, majú vždy za následok vysádzanie práve jednej medzery. Vysádzanie viacerých medzier za sebou sa dá vynútiť pomocou príkazu \␣ (t.j. spätná lomka nasledovaná medzerou), nie je však odporúčané robiť to, pokiaľ to nie je nevyhnutné.

Podobne ako medzera a tabulátor, aj ďalšie biele znaky (ako napr. enter) spôsobia vysádzanie medzery. Výnimkou je ale prázdny riadok, ktorý slúži ako štandardný spôsob na začatie nového odstavca. Viacero prázdnych riadkov za sebou sa chová rovnako ako jeden prázdny riadok, t.j. začína nový odstavec.

Začatie nového riadku sa dá vynútiť pomocou príkazu \\ resp. `\newline`. Začiatok novej strany možno vynútiť pomocou príkazu `\newpage`. Vo všeobecnosti však platí, že s takýmito vynucovacími príkazmi treba zaobchádzať opatrne. Aj keď sa môžu zísť napríklad pri formátovaní titulnej strany, ich používanie v bežnom texte zvyčajne nie je najlepší nápad.

2.2 Príkazy a prostredia

Okrem bežného textu sú ďalšími dvoma základnými kameňmi LaTeXu predovšetkým príkazy a prostredia. Každý príkaz má v LaTeXu tvar

`\⟨názov príkazu⟩`,

kde ⟨názov príkazu⟩ je buď reťazec písmen (a-z a A-Z) alebo jeden znak rôzny od písmena. Pozor: LaTeX rozlišuje veľké a malé písmená!

Teda napríklad, \prikaz42 je (vymyslený) príkaz s názvom prikaz nasledovaný číslom 42, ktoré sa vypíše štandardným spôsobom, pretože sa už nepovažuje za súčasť názvu príkazu. Podobne, \\niečo je príkaz na začatie nového riadku, po ktorom sa bežným spôsobom vypíše slovo „niečo“ – keďže totiž prvý znak názvu príkazu nebolo písmeno, nasledujúce znaky sa už za súčasť názvu príkazu nepovažujú.

Jedinú menšiu výnimku z toho, čo bolo napísané, tvoria medzery a ostatné biele znaky. Medzery, ktoré nasledujú bezprostredne za názvom príkazu, totiž LaTeX nevypisuje. Uvažujme napríklad príkaz

\LaTeX,

ktorý vypíše „logo“ LaTeX. Potom:

- Výsledkom kódu \LaTeX blbne je „LaTeXblbne“, teda bez medzery.
- Riešením je správne použitie zložených zátvoriek { a }. Kód \LaTeX{} blbne, ako aj kód {\LaTeX} blbne už vyprodukuje zamýšľaný výsledný text „LaTeX blbne“.

Ďalším príkazom, na ktorom si je možné tieto záležitosti všimnúť, je napríklad príkaz

\ldots,

ktorý vypíše tri bodky... V prípade potreby vypísať tri bodky, vždy treba použiť tento (alebo obdobný) príkaz – výsledok priamočiareho kódu ... nie je typograficky pekný. Za príkazom \ldots *nie je* v bežnom texte potrebné vkladať medzeru (napríklad pomocou zložených zátvoriek).

Niektoré príkazy akceptujú aj jeden alebo viac parametrov, ktoré môžu byť povinné alebo nepovinné. Povinné parametre sa zadávajú priamo za názov príkazu do zložených zátvoriek, nepovinné do hranatých zátvoriek. Každý príkaz má určené svoje poradie parametrov.

Ako príklady príkazov s jedným parametrom uveďme niekoľko príkazov na formátovanie písma:

- Príkaz \textbf{časť textu tučne} vypíše **časť textu tučne**.
- Príkaz \emph{zvýrazní časť textu} *zvýrazní časť textu*.
- Príkaz \textsc{Časť Textu v Kapitálkach} vypíše Časť Textu v Kapitálkach.

Príkladom príkazu s jedným povinným a jedným nepovinným parametrom je napríklad nám už z predchádzajúcej kapitoly známy príkaz \usepackage[možnosti]{balík}.

Ďalším dôležitým stavebným kameňom LaTeXu sú tzv. prostredia. Ide o úseky LaTeXového kódu ohraničené príkazmi

\begin{⟨názov prostredia⟩}

a

\end{⟨názov prostredia⟩}.

Kód umiestnený medzi tieto dva príkazy potom LaTeX spracúva spôsobom, ktorý je špecifický pre dané prostredie. Typickými príkladmi prostredí sú prostredia na zarovnávanie textu (štandardne sa text zarovnáva do odstavcov):

- Prostredie center zarovná časť textu na stred.
- Prostredie flushleft zarovná časť textu vľavo.
- Prostredie flushright zarovná časť textu vpravo.

Na lepšie zžitie sa s vyššie uvedenými skutočnosťami môže čitateľ považovať za zmysluplné trochu sa pohrať s nasledujúcim príkladom zdrojového súboru.

```
\documentclass{article}
\usepackage[slovak]{babel}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\begin{document}
Toto je ukážkový dokument vytvorený v \LaTeX u.
Kebyže chcem dať za slovo \LaTeX{} medzeru, urobím to tak, ako som to urobil.
Všimnime si, že              veľa medzier vyústi len v jednu medzeru.
A tiež si všimnime, že entre na konci riadkov takisto vyústia len v jednu medzeru.

Nový odstavec začne až po prázdnom riadku. Napíšeme \textbf{kus textu tučne}.
A kus textu ešte trochu \emph{zvýrazníme}.
\begin{center}
Kus textu zarovnáme na stred.
\end{center}
\begin{flushleft}
Kus vľavo\ldots
\end{flushleft}
\begin{flushright}
\ldots{} a kus vpravo.
\end{flushright}
A teraz, keď už nie sme v žiadnom prostredí, môžeme pokojne pokračovať
v písaní textu zarovnaného do odstavcov. Všimnime si, že teraz ešte vlastne
pokračujeme v starom odstavci, začatom ešte pred experimentmi s prostrediami.
Dôvod: nebol žiaden prázdny riadok! Čo by sa stalo, keby sme ho niekam pridali?
Vyskúšajte sami!
\end{document}
```

2.3 Špeciálne znaky

V LaTeXu existuje niekoľko špeciálnych znakov, ktoré majú v jeho jazyku určitú funkciu (stretli sme sa už napríklad so spätnou lomkou alebo so zloženými zátvorkami). Takéto znaky sa teda logicky nevysádzajú do výsledného dokumentu, iba plnia svoju úlohu. Pre každý z nich však existuje aj príkaz, ktorý možno použiť v prípade, že je potrebné daný znak aj naozaj vysádzať. Špeciálne znaky a im zodpovedajúce príkazy sumarizuje nasledujúca tabuľka.

Znak	Príkaz
#	`\#`
$	`\$`
%	`\%`
^	`\^{}`
&	`\&`
_	`\_`
{	`\{`
}	`\}`
~	`\~{}`
\	`\textbackslash`

LaTeX je známy tým, že dokáže v pomerne vysokej typografickej kvalite vysádzať obrovské množstvo znakov (predovšetkým v matematickom móde, ktorým sa zaoberá oddiel 2.7). Niektoré z týchto znakov sú „zabudované do jadra“ LaTeXu alebo TeXu, no väčšina z nich je súčasťou rôznych balíkov.

Samozrejme nie je v možnostiach bežného človeka si všetky príkazy na vkladanie týchto znakov zapamätať. Preto je užitočné vedieť, kde sa dajú nájsť ich zoznamy.

Neoceniteľnou pomôckou je predovšetkým *The Comprehensive LaTeX Symbol List* [13]. Ide o (vo verzii z roku 2009) 164-stranový dokument voľne prístupný na internete, obsahujúci výhradne zoznamy symbolov a nim zodpovedajúcich LaTeXových príkazov.

Veľmi užitočnou pomôckou je tiež *The TEX Cookbook* [6]. Ide o (podstatne kratší) dokument, ktorý môže slúžiť ako „ťahák“ pre používateľov čistého TEXu (všetky príkazy však bez problémov možno použiť aj v LaTeXu, hoci ten má v niektorých prípadoch k dispozícii aj nejakú alternatívnu metódu).

2.4 Domáca úloha vytvorená v LaTeXu

Znalosti, ktoré mohol čitateľ získať v predchádzajúcich oddieloch, už stačia na vytvorenie jednoduchého vzoru riešenia domácej úlohy. Toto jednoduché praktické cvičenie je náplňou tohto oddielu. V nasledujúcom ukážkovom kóde sú použité aj dve novinky. Prvou je použitie balíku `a4wide`, ktorý optimalizuje výsledný text pre formát papiera A4.[1] Druhou novinkou je príkaz `\pagestyle{empty}`. Ten, zhruba povedané, ruší čísla stránok a podobné „ozdoby“ od miesta kde je použitý, až po koniec dokumentu – čísla stránok sú pri väčšinou krátkych domácich úlohách zbytočné. V prípade, že je treba potlačiť vypísanie takýchto „ozdôb“ iba na jednej konkrétnej strane, možno použiť obdobný príkaz `\thispagestyle{empty}`, ktorý slúži práve na tento účel.

```
\documentclass{article}
\usepackage[slovak]{babel}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage{a4wide}
\begin{document}
\pagestyle{empty}
\begin{center}
\textbf{Prvá domáca úloha z FoJe} \\
Jozef Mrkvička, 2INF
\end{center}

Hurá, moja prvá domáca úloha vysádzaná v {\LaTeX}u! Po formálnej stránke je to na 100\%!!!
Keďže som ale úplne dutý, toto je celé moje riešenie\ldots
\end{document}
```

2.5 Komentáre

Podobne ako vo väčšine programovacích a skriptovacích jazykov, aj v LaTeXu existuje možnosť vkladať do kódu komentáre.

Jednoduchý spôsob, ako zakomentovať časť textu, je pomocou špeciálneho znaku `%`. Po pridaní tohto znaku si LaTeX nebude všímať nič, čo je v danom riadku za ním. Ide teda o jednoriadkové komentáre.

Viacriadkové komentáre síce v LaTeXu priamo podporované nie sú, ale balík `verbatim` okrem iného obsahuje aj prostredie `comment`, ktoré slúži práve na tento účel.

Nasledujúci (neúplný) príklad kódu teda sumarizuje použitie komentárov v LaTeXu.

```
...
\usepackage{verbatim}
...
\begin{document}
...
... % zakomentovaný text
...
\begin{comment}
viacriadkový
komentár
\end{comment}
...
\end{document}
```

[1] Bez jeho použitia by okraje boli zbytočne veľké, aj keď fajnšmekri by možno oponovali optimalizáciou počtu znakov na riadok.

2.6 Logické členenie dokumentov

LaTeX umožňuje pomerne prirodzeným spôsobom udržiavať logickú štruktúru v ňom vytváraných dokumentov, vrátane číslovania kapitol, oddielov, pododdielov, matematických viet, obrázkov, tabuliek a ďalších logických jednotiek. Udržiavanie takejto logickej štruktúry dokumentu možno nie je nutné pri krátkych výtvoroch (ako sú napríklad domáce úlohy), ale štruktúrovanie dlhších odborných dokumentov (ako sú napríklad záverečné práce) je v zásade nutnosťou. Teraz si opíšeme len niekoľko základných možností, ktoré LaTeX na tomto poli ponúka.

V prvom rade je potrebné zmieniť sa o nadpisoch kapitol, oddielov, pododdielov a podobne. Keďže jedným z prvoradých cieľov LaTeXu je ľahká orientácia vo výslednom dokumente, LaTeX obsahuje niekoľko základných príkazov, ktoré takéto nadpisy vysádzajú určitým štandardným spôsobom. Tento štandardný spôsob síce možno zmeniť,[2] ale nie je to úplne odporúčané (aspoň ak je snahou vytvoriť naozaj prehľadný výstupný dokument).

Tri triedy dokumentov spomínané vyššie – t.j. `article`, `report` a `book` – sa okrem iného líšia aj úrovňou členenia. Najvyššou úrovňou členenia (ak nerátame časti) v triedach `book` a `report` je kapitola, v triede `article` je to oddiel.

Kapitoly, oddiely a pododdiely sú v LaTeXu dvoch typov: číslované a nečíslované. Na bežné členenie textu je odporúčané používať tie číslované (minimálne pri tvorbe odborných textov, kde väčšinou platí, že prehľadnosť je nadovšetko). Nečíslované kapitoly a oddiely by mali slúžiť skôr na špeciálne účely – nečíslovanou kapitolou (alebo v článku oddielom) môže byť napríklad úvod alebo záver, nečíslovaným oddielom môže byť napríklad aj záverečný oddiel na konci kapitoly obsahujúci rôzne poznámky, cvičenia alebo podobné záležitosti.

Základné príkazy (existujú aj ďalšie) na tvorbu číslovaných nadpisov sú nasledovné:

- `\chapter{⟨názov⟩}` – kapitola,
- `\section{⟨názov⟩}` – oddiel,
- `\subsection{⟨názov⟩}` – pododdiel,
- `\subsubsection{⟨názov⟩}` – „podpododdiel“,

kde `⟨názov⟩` treba nahradiť samotným názvom danej logickej jednotky textu. Pri východzom nastavení sa číslujе iba po úroveň pododdielu a „podpododdiely“ sa už nečíslujú ani vo svojej „číslovanej“ verzii. Toto nastavenie sa dá zmeniť, vysvetlenie by ale presahovalo rámec tohto textu.[3]

Na tvorbu nečíslovaných nadpisov sa používajú analogické „hviezdičkové“ príkazy:

- `\chapter*{⟨názov⟩}` – nečíslovaná kapitola,
- `\section*{⟨názov⟩}` – nečíslovaný oddiel,
- `\subsection*{⟨názov⟩}` – nečíslovaný pododdiel,
- `\subsubsection*{⟨názov⟩}` – (ani pri zmene hĺbky číslovania) nečíslovaný „podpododdiel“.

Najviac do očí bijúci rozdiel medzi číslovanou a nečíslovanou verziou nadpisu je asi pri kapitolách, kde v nečíslovanej verzii okrem samotného jej čísla „zmizne“ aj text „Kapitola“, ktorý pri číslovaných kapitolách LaTeX automaticky vypisuje.

LaTeX tiež ponúka jednoduchý spôsob ako vygenerovať obsah, pozostávajúci zo všetkých číslovaných kapitol, oddielov a pododdielov (v prípade, že je nezmenená hĺbka číslovania spomínaná vyššie). Obsah možno vygenerovať použitím príkazu

```
\tableofcontents
```

[2] Ako koniec koncov všetko v LaTeXu. Samotný TeX je turingovsky úplný, čo môže naznačovať, že jeho možnosti sú pomerne veľké. Neštandardné postupy akurát v LaTeXu vyžadujú vynaloženie väčšieho úsilia a väčšinou nie je veľa rozumných dôvodov na ich použitie.

[3] Rozhodne nie kvôli náročnosti, ale iba kvôli skutočnosti, že to nepatrí k nutným základom (a ani sa to veľmi často nepoužíva). Záujemcovia si jednoduchý recept určite ľahko nájdu.

na tom mieste zdrojového kódu, kde sa má vo výslednom dokumente obsah zjaviť. Pri triedach dokumentov `book` a `report` sa obsah vypíše ako samostatná (nečíslovaná) kapitola a pri triede `article` ako samostatný (nečíslovaný) oddiel.

Občas môže nastať situácia, že je do obsahu potrebné zahrnúť aj niektoré nečíslované logické jednotky textu (napríklad úvod alebo záver), ktoré automaticky do obsahu zahrnuté nie sú. LaTeX obsahuje špeciálny príkaz `\addcontentsline`, ktorý umožňuje do obsahu pridať úplne čokoľvek, a to s odkazom na ľubovoľné miesto v dokumente (presnejšie na to miesto, kde je uvedený príkaz použitý). Vložiť nečíslovanú jednotku do obsahu je teda možné jednoducho jej manuálnym vložením do obsahu s odkazom na miesto hneď za nadpisom tejto jednotky.

Príkaz `\addcontentsline` vyžaduje tri parametre – prvý udáva „typ obsahu“, do ktorého sa má daný záznam pridať (LaTeX totiž okrem klasického „hlavného“ obsahu podporuje aj zoznamy obrázkov, či tabuliek). Pre „hlavný“ obsah treba tento parameter vždy nastaviť na `toc`. Druhý parameter je úroveň logického členenia, v ktorej sa má nový záznam v obsahu objaviť (t.j. možné hodnoty sú napr. `chapter`, `section` a pod.). Tretí parameter je samotný text, ktorý sa má do obsahu vložiť. Praktický príklad použitia možno nájsť na konci tohto oddielu.

LaTeX umožňuje jednoducho vygenerovať aj jednoduchú titulnú stranu dokumentu (resp. nadpis článku). Aj keď sa tieto veci väčšinou riešia skôr manuálne, niekedy môže byť aj tento automatický spôsob užitočný – predovšetkým v prípadoch, keď vzhľad titulnej strany nie je až tak veľmi dôležitý.

Tento mechanizmus funguje tak, že sa najprv (najlepšie v preambule) použijú tieto dva príkazy:

- `\title{⟨názov dokumentu⟩}` – nastaví názov dokumentu na `⟨názov dokumentu⟩`,
- `\author{⟨autor⟩}` – nastaví autora dokumentu na `⟨autor⟩`.

V prípade, že sa má na titulnej strane zjaviť iný dátum, než aktuálny, možno použiť aj príkaz `\date`, ktorého parametrom je text, ktorý má byť vypísaný namiesto dátumu.

Samotná titulná strana (resp. nadpis článku) sa potom vygeneruje v tele dokumentu pomocou príkazu

`\maketitle`,

ktorý žiaden parameter nepotrebuje.

Nasledujúci príklad zdrojového kódu sumarizuje vyššie uvedené informácie.

```
\documentclass{report}
\usepackage[slovak]{babel}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\title{Geniálne štruktúrovaný dokument}
\author{Jozef Mrkvička}
\begin{document}
\maketitle
\tableofcontents
\chapter*{Úvod}
\addcontentsline{toc}{chapter}{Úvod}
\chapter{Prvá kapitola}
\section{Jej prvý oddiel}
\section{Jej druhý oddiel}
\subsection{A ten má aj pododdiel}
\subsection{A ďalší}
\section{Posledný oddiel v tejto kapitole}
\chapter{Druhá kapitola}
\chapter*{Záver}
\addcontentsline{toc}{chapter}{Záver}
\end{document}
```

2.7 Matematické vzorce

Nasledujúce partie tohto textu stručne vysvetľujú nutné základy, ktoré treba zvládnuť za účelom tvorby dokumentov obsahujúcich „matematiku“. Podobne ako to platí pre celú túto kapitolu, nebude výklad ani v tomto prípade úplný alebo postačujúci na všetky bežné účely. O niečo dôkladnejšie (avšak tiež nie úplné) spracovanie tejto problematiky možno nájsť napríklad v už spomínanom úvode [10] alebo v jeho českom preklade [9]. Podstatne rozsiahlejším zdrojom informácií môže byť napríklad internetová kniha [16], kde možno okrem iného nájsť aj ďalšie „vedecké“ aplikácie LaTeXu: napríklad sádzanie algoritmov a zdrojových kódov, vzorcov chemických zlúčenín, a podobne.

Aj keď základná funkcionalita sadzby matematických vzorcov je zabudovaná priamo do LaTeXu, pre väčšinu pokročilejších typografických úloh je potrebné použiť niektorý z balíkov zo sady $\mathcal{AMS}$-LaTeX, vyvinutej americkým združením matematikov $\mathcal{AMS}$ (American Mathematical Society). Najzákladnejšie balíky $\mathcal{AMS}$-LaTeXu sú pravdepodobne `amsmath`, `amsfonts` a `amssymb`, ktoré je dobré načítať v preambule každého dokumentu obsahujúceho matematiku. Vo zvyšku tohto oddielu budeme predpokladať, že tieto balíky sú používané a na prípadnú nutnosť ich použitia nebudeme explicitne upozorňovať.

2.7.1 Prostredia na sadzbu matematických vzorcov

Hrubo možno matematický text v LaTeXu rozdeliť do dvoch kategórií: na matematický text vypísaný priamo v odstavci, ako napr. $2x + 9 = 42$ a matematický text vypísaný v samostatnom riadku, ako napr.

$$2x + 9 = 42$$

alebo

$$\frac{\partial u}{\partial t} - \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) = 0.$$

Na vypísanie matematického textu priamo do odstavca stačí tento text vložiť medzi dva znaky `$`. Teda rovnicu $2x + 9 = 42$ dostaneme ako `$2x + 9 = 42$`.

Vypísanie matematického textu v samostatnom riadku možno dosiahnuť pomocou prostredia `displaymath`, do ktorého stačí daný matematický text obaliť. Napríklad,

$$2x + 9 = 42$$

dostaneme nasledovne.

```
...
\begin{displaymath}
2x + 9 = 42
\end{displaymath}
...
```

Hojne využívanou možnosťou je automatické číslovanie (významných) rovníc, na ktoré potom možno v neskoršom texte odkazovať (viď oddiel 2.11 o krížových odkazoch). Pre vloženie číslovanej rovnice stačí namiesto prostredia `displaymath` použiť prostredie `equation`. Existuje aj prostredie `equation*`, ktoré sa správa podobne ako `displaymath`.

2.7.2 Horné a dolné indexy

Dôležitú úlohu zohrávajú pri sádzaní matematických výrazov horné a dolné indexy (horné indexy sa využívajú aj ako exponenty). Vysádzanie horného a dolného indexu sa v LaTeXu dosiahne nasledovne (treba byť v matematickom móde):

```
⟨symbol⟩_{⟨dolný index⟩}^{⟨horný index⟩},
```

pričom ľubovoľný z týchto dvoch indexov možno samozrejme vynechať. Pri indexoch obsahujúcich iba jeden symbol možno vynechať aj zložené zátvorky okolo nich. Napríklad vysádzanie rovnice

$$a_1^3 + a_2^4 + \ldots + a_n^{n+2} = 42^{42}$$

možno docieliť pomocou nasledovného kódu.

```
...
\begin{displaymath}
a_1^3 + a_2^4 + \ldots + a_n^{n+2} = 42^{42}
\end{displaymath}
...
```

2.7.3 Zložitejšie matematické výrazy

V nasledujúcom si stručne vysvetlíme niektoré zložitejšie konštrukcie, ktoré sa môžu vyskytnúť pri písaní väčšiny matematických textov. Úvodom však treba poznamenať, že veľká časť procesu písania matematického textu je o rôznych symboloch (napr. grécke písmená, operátory, kvantifikátory, ...), fontoch a podobne. Tieto záležitosti rieši veľké množstvo jednoducho použiteľných príkazov. Tie najzákladnejšie, ktoré sú priamo súčasťou TEXu, možno nájsť v [6]. Ďalšie napríklad vo vyčerpávajúcom zozname [13] alebo v najrôznejších príručkách LATEXu. Niektoré z týchto príkazov sa budú v nasledujúcich príkladoch vyskytovať aj bez toho, aby sme na ne explicitne upozorňovali.[4]

Začnime zlomkami. Syntax vkladania zlomkov je v LATEXu nasledovná (samozrejme treba byť v matematickom móde – v budúcnosti už na to upozorňovať nebudeme):

`\frac{⟨čitateľ⟩}{⟨menovateľ⟩}`.

Nasleduje príklad použitia tohto príkazu a jeho výsledný efekt.

```
...
\begin{displaymath}
x = \frac{1 + \gamma^2}{4}
\end{displaymath}
...
```

$$x = \frac{1 + \gamma^2}{4}$$

Podobne ako zlomky fungujú aj binomické koeficienty – stačí nahradiť príkaz `\frac` príkazom `\binom`.

```
...
\begin{displaymath}
\binom{n}{k} = \frac{n!}{(n-k)!\, k!}
\end{displaymath}
...
```

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}$$

Odmocniny sa vkladajú pomocou príkazu `\sqrt`, ktorý má len jeden povinný parameter – výraz pod odmocninou. Príkazu možno pred povinný parameter zadať aj nepovinný parameter – stupeň odmocniny. Prácu s odmocninami ilustruje nasledujúci príklad.

```
...
\begin{displaymath}
\sqrt{x}\ \sqrt[4]{x} = \sqrt[4]{x^3}
\end{displaymath}
...
```

$$\sqrt{x}\,\sqrt[4]{x} = \sqrt[4]{x^3}$$

V zásade na jeden spôsob sa sádzajú integrály, sumy a súčiny – slúžia na to príkazy `\int`, `\sum` resp. `\prod` a prípadné horné a dolné hranice sa sádzajú pomocou horných a dolných indexov. Postupne teraz uvedieme príklady na neurčitý integrál, určitý integrál, sumu a súčin.

```
...
\begin{displaymath}
\int \sin x\,\mathrm{d}x = -\cos x + C,
 \qquad C \in \mathbb{R}
\end{displaymath}
...
```

$$\int \sin x \,\mathrm{d}x = -\cos x + C, \qquad C \in \mathbb{R}$$

```
...
\begin{displaymath}
\int_0^{2\pi} \sin x\,\mathrm{d}x = 0
\end{displaymath}
...
```

$$\int_0^{2\pi} \sin x \,\mathrm{d}x = 0$$

```
...
\begin{displaymath}
\sum_{k=1}^n k^2 = \frac{n(n+1)(2n+1)}{6}
\end{displaymath}
...
```

$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

```
...
\begin{displaymath}
\prod_{k=1}^n 2^k = 2^{\sum_{k=1}^n k} = 2^{\frac{n(n+1)}{2}}
\end{displaymath}
...
```

$$\prod_{k=1}^{n} 2^k = 2^{\sum_{k=1}^{n} k} = 2^{\frac{n(n+1)}{2}}$$

Pri integráloch ešte stojí za zmienku spôsob, akým sme sádzali symbol $\mathrm{d}x$. Jednak, symbol pre diferenciál je sádzaný vzpriameným fontom, čo zabezpečuje príkaz `\mathrm` a ďalej, pred ním je vysádzaná krátka medzera, čo zabezpečuje príkaz `\,`.

Uzavrime ešte tento pododdiel krátkou zmienkou o limitách. Na ich sádzanie slúži príkaz `\lim`, na vysádzanie šípky možno použiť príkaz `\to`, prípadne `\rightarrow`.

```
...
\begin{displaymath}
\lim_{x\to 0} \sin\frac{1}{x} =
 \ldots\mathrm{Ufff}
\end{displaymath}
...
```

$$\lim_{x\to 0} \sin\frac{1}{x} = \ldots \mathrm{Ufff}$$

2.7.4 Zátvorky

O zátvorkách si treba povedať predovšetkým dve veci. Po prvé, v LaTeXu existujú zátvorky viacerých typov. Najbežnejšie „guľaté“ a hranaté zátvorky možno jednoducho zadať z klávesnice, zložené zátvorky sa sádzajú pomocou príkazov `\{` a `\}` a pre ďalšie typy zátvoriek treba použiť niektorý z príkazov, ktoré možno nájsť v tabuľkách symbolov (napríklad v [13]).

Po ďalšie, menší problém so zátvorkami môže nastať v prípade, že sa pokúsime uzavrieť do zátvoriek nejaký „príliš vysoký výraz“. Zátvorky totiž stále ostanú na svojej základnej veľkosti, čo nevyzerá príliš dobre (a pri použití napríklad v záverečnej práci vzbudzuje intenzívny dojem práce na poslednú chvíľu).

Riešením je použitie príkazov `\left` a `\right`. Aj tieto príkazy musia byť „dobre uzátvorkované“, t.j. po použití `\left` na nejakú ľavú zátvorku je nutné na nejakú pravú zátvorku použiť aj `\right` (prípadne, ak je zamýšľaným efektom iba jedna zátvorka, možno použiť príkaz `\right.`,

[4] V nádeji, že sa tým čitateľovi aspoň trochu rozšíri jeho repertoár.

t.j. namiesto zátvorky použiť bodku – to znamená, že „sme si vedomí absencie uzatvárajúcej zátvorky a nevadí nám to“). Príkazy `\left` a `\right` presne riešia spomínaný problém – veľkosť oboch zátvoriek uspôsobia na správnu výšku podľa výrazu, ktorý obklopujú.[5] Nasleduje príklad nesprávneho a správneho použitia zátvoriek.

```
...
\begin{displaymath}
(\sum_{n=1}^{\infty}\frac{1}{n^2})
\end{displaymath}
...
```

$$(\sum_{n=1}^{\infty}\frac{1}{n^2})$$

```
...
\begin{displaymath}
\left(\sum_{n=1}^{\infty}\frac{1}{n^2}\right)
\end{displaymath}
...
```

$$\left(\sum_{n=1}^{\infty}\frac{1}{n^2}\right)$$

2.7.5 Systémy rovníc a výpočty na viac riadkov

Prostredia `align` a `align*` možno využiť na sadzbu systémov rovníc resp. výpočtov zaberajúcich viac riadkov. Od seba sa líšia jedine tým, že `align` na rozdiel od `align*` jednotlivé riadky čísluje ako rovnice (pre vybrané riadky sa ale dá číslovanie vypnúť).

Použitie týchto prostredí je podobné použitiu prostredí `equation` resp. `displaymath`, s dvoma rozdielmi. Po prvé, v každom zamýšľanom riadku treba vyznačiť symbolom `&` miesto, v ktorom sa má daná rovnica (časť výpočtu) zarovnať s ostatnými (väčšinou sa zarovnanie vkladá pred znamienko `=`). Po ďalšie, konce jednotlivých riadkov treba vyznačiť pomocou príkazu `\\`.

Použitie prostredia `align*` je demonštrované nasledujúcim príkladom.

```
...
\begin{align*}
e^x &= 1 + x + \frac{x^2}{2!} +
          \frac{x^3}{3!} + \ldots = \\
    &= \sum_{n=0}^{\infty} \frac{x^n}{n!}
\end{align*}
...
```

$$\begin{aligned} e^x &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \\ &= \sum_{n=0}^{\infty} \frac{x^n}{n!} \end{aligned}$$

Prostredie `align` funguje obdobne, s tým rozdielom, že čísluje jednotlivé riadky ako rovnice. Toto číslovanie sa však dá pre vybrané riadky vypnúť pomocou príkazu `\nonumber` tak, ako napríklad v nasledujúcom kuse kódu.

```
...
\begin{align}
e^{ix} &= \cos x + i \sin x \\
e^x &= 1 + x + \frac{x^2}{2!} +
          \frac{x^3}{3!} + \ldots = \nonumber\\
    &= \sum_{n=0}^{\infty} \frac{x^n}{n!}
\end{align}
...
```

$$\begin{aligned} e^{ix} &= \cos x + i \sin x && (1) \\ e^x &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \\ &= \sum_{n=0}^{\infty} \frac{x^n}{n!} && (2) \end{aligned}$$

[5] Poznamenajme však, že s použitím tejto dvojice príkazov je problém pri viacriadkových odvodeniach v prostredí `align*`, opísanom nižšie. V nich je možné používať tieto príkazy, iba pokiaľ je uzátvorkovaný výraz v rámci jedného riadku. Našťastie sa však veľkosti zátvoriek dajú určiť aj manuálne, pomocou príkazov na to určených – ich prehľad možno nájsť vo väčšine tabuliek symbolov.

2.7.6 Matice a svorky

V závere našej stručnej exkurzie do typografie matematických textov si ešte povedzme niečo o maticiach. Ukáže sa, že prostredie, ktoré budeme používať na sádzanie matíc, má aj iné využitia – napríklad pri sádzaní svoriek.

Týmto prostredím je prostredie `array`, ktoré sádza matematický text do políčok obdĺžnikovej mriežky.[6] Príkaz na začatie prostredia `array` však vyžaduje ešte jeden argument – reťazec nad abecedou $\{\mathtt{c}, \mathtt{l}, \mathtt{r}\}$, ktorý má nasledujúcu funkciu: jeho dĺžka udáva počet stĺpcov mriežky a jednotlivé znaky potom určujú zarovnanie daného stĺpca (`c` – na stred, `l` – vľavo, `r` – vpravo).

Použitie prostredia `array` demonštruje nasledujúci jednoduchý príklad, v ktorom sa vysádza mriežka typu 2×2.

```
...
\begin{displaymath}
\begin{array}{cc}
1 & 2 \\ 3 & 4
\end{array}
\end{displaymath}
...
```

$$\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array}$$

Z takejto mriežky možno získať maticu jednoducho jej „obalením“ do dostatočne veľkých zátvoriek.

```
...
\begin{displaymath}
\left(\begin{array}{cc}
1 & 2 \\ 3 & 4
\end{array}\right)
\end{displaymath}
...
```

$$\left(\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array}\right)$$

Pri sádzaní matíc sa môžu zísť príkazy na sádzanie rôzne orientovaných troch bodiek: menovite `\cdots` (horizontálne tri bodky vertikálne zarovnané na stred riadku), `\vdots` (vertikálne tri bodky) a `\ddots` (diagonálne tri bodky). Ich možné použitie pri sádzaní matíc ukazuje nasledujúci príklad. Záverečný príklad potom ukazuje, ako pomocou prostredia `array` vysádzať svorku.

```
...
\begin{displaymath}
A = \left(\begin{array}{cccc}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nn}
\end{array}\right)
\end{displaymath}
...
```

$$A = \left(\begin{array}{cccc} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{array}\right)$$

```
...
\begin{displaymath}
f(x) = \left\{\begin{array}{ll}
1 & \textrm{ ak } x \in \mathbb{Q} \\
0 & \textrm{ inak }
\end{array}\right.
\end{displaymath}
...
```

$$f(x) = \left\{\begin{array}{ll} 1 & \text{ak } x \in \mathbb{Q} \\ 0 & \text{inak} \end{array}\right.$$

[6] Na jeho použitie už ale treba pracovať v matematickom móde, t.j. `array` na rozdiel napríklad od prostredia `align` matematický mód automaticky nezapína.

2.8 Zoznamy

Prostredia `itemize` a `enumerate` v LaTeXu slúžia na sádzanie nečíslovaných resp. číslovaných zoznamov. Použitie týchto prostredí je jednoduché – v oboch možno použiť príkaz `\item`, ktorý vysádza jeden bod zoznamu, a ktorého argumentom je text vysádzaný v rámci daného bodu. Pomocou nepovinného parametra príkazu `\item` možno zmeniť „symbol odrážky“.

Prostredia na tvorbu zoznamov možno do seba ľubovoľne vnárať (t.j. v rámci argumentu príkazu `\item` možno tieto prostredia opätovne použiť). Tvorbu zoznamov v LaTeXu demonštrujeme na nasledujúcom príklade.

```
...
\begin{enumerate}
\item{Prvý bod.}
\item{Druhý bod.}
\item[3a.]{Bod 3a.}
\item[3b.]{Bod 3b.
\begin{itemize}
\item{Prvý nečíslovaný vnorený bod.}
\item{A druhý.}
\end{itemize}
}
\end{enumerate}
...
```

1. Prvý bod.
2. Druhý bod.

3a. Bod 3a.

3b. Bod 3b.

- Prvý nečíslovaný vnorený bod.
- A druhý.

2.9 Tabuľky

Tvorba tabuliek je v LaTeXu pomerne veľká veda. Tento oddiel preto obsahuje iba najzákladnejšie informácie o tvorbe tabuliek pomocou prostredia `tabular`.

Použitie prostredia `tabular` je v mnohom podobné použitiu prostredia `array`. Tu sa však nepracuje v matematickom, ale v textovom móde. Príkaz na začatie prostredia `tabular` má tvar

`\begin{tabular}[⟨pozícia⟩]{⟨zarovnania⟩}`.

Nepovinný parameter `⟨pozícia⟩` slúži na nastavenie vertikálneho zarovnania tabuľky vzhľadom na obklopujúci text v prípade, že tabuľka netvorí samostatný odstavec. Vo väčšine prípadov je však tento parameter zbytočný, a preto sa o ňom nebudeme bližšie zmieňovať. Častejšie môže vzniknúť potreba nastavenia horizontálneho zarovnania tabuľky – na to však možno použiť prostredia `center`, `flushleft`, resp. `flushright`, ktoré boli spomenuté na začiatku tejto kapitoly. Povinný parameter `⟨zarovnania⟩` má rovnakú funkciu, ako pri prostredí `array`.

Prostredie `tabular` funguje, podobne ako prostredie `array`, predovšetkým na základe špeciálneho znaku `&` a príkazu `\\`. Zmienku si však zaslúži spôsob, ako sádzať zvislé resp. vodorovné čiary tabuľky. Na sádzanie zvislých čiar možno použiť zmienený parameter `⟨zarovnania⟩` – ak do reťazca tvoreného znakmi `c`, `l` a `r` označujúcimi zarovnania vložíme symbol `|`, medzi zodpovedajúcimi stĺpcami sa vysádza zvislá čiara. Vodorovné čiary sa sádzajú pomocou príkazu `\hline` uvedeného na začiatku riadku, ktorý má byť v tabuľke pod ňou. Dvojitú zvislú čiaru dostaneme pomocou dvojitého použitia symbolu `|`, dvojitú vodorovnú čiaru pomocou dvojitého použitia príkazu `\hline`.

V LaTeXu existujú prostredia, ktoré umožňujú vysádzať aj zložitejšie tabuľky. Napríklad balík a rovnomenné prostredie `tabularx` obsahuje možnosť „natiahnutia“ vybraných stĺpcov na maximálnu možnú šírku a balík `longtable` umožňuje sádzať tabuľky na viac strán. Funkcionalitu oboch týchto balíkov spája (pomerne nový) balík `tabu`.

Nasleduje jednoduchý príklad použitia balíku `tabular`.

```
...
\begin{center}
\begin{tabular}{|l|l|}
\hline Prvý stĺpec & Druhý stĺpec \\
\hline \hline 50486 & 46586 \\
\hline 56645 & 64566 \\
\hline 75477 & 90111 \\
\hline
\end{tabular}
\end{center}
...
```

Prvý stĺpec	Druhý stĺpec
50486	46586
56645	64566
75477	90111

2.10 Vety, definície, lemy a podobné záležitosti

LaTeX tiež poskytuje spôsob, ako do textu zakomponovať automaticky číslované matematické vety, lemy, definície, poznámky, dôsledky a podobne. Keďže takýchto „jednotiek matematického textu na spôsob vety“ sa v literatúre zvykne vyskytovať neúrekom, nie sú ich jednotlivé druhy v LaTeXu predom definované, ale autor vytváraného dokumentu ich definuje v jeho preambule pomocou príkazov tvaru

`\newtheorem{⟨idn⟩}[⟨číslovať spoločne s⟩]{⟨vypisovaný text⟩}[⟨číslovať v rámci⟩].`

Povinnými parametrami sú `⟨idn⟩`, čo je označenie daného typu „vety“ používané interne v rámci zdrojového kódu dokumentu a `⟨vypisovaný text⟩`, čo je názov daného typu „vety“ tak, ako sa vypisuje priamo vo výslednom dokumente. Napríklad, identifikátor pre definície môže byť `dfn`, pričom vypisovaný text je `Definícia`.

Príkazu `\newtheorem` možno zadať aj dva nepovinné parametre. Prvým takýmto parametrom je parameter `⟨číslovať spoločne s⟩`. Jeho hodnotou je identifikátor nejakého skôr definovaného druhu „vety“. Novodefinovaný druh „vety“ potom nebude číslovaný zvlášť, ale bude s týmto prv definovaným druhom „vety“ zdieľať počítadlo. Druhým nepovinným parametrom je parameter `⟨číslovať v rámci⟩` – jeho hodnotou môže byť niektorá logická jednotka textu, napríklad `section`. Daný druh „vety“ sa potom bude číslovať v rámci tejto logickej jednotky a po začiatku ďalšej logickej jednotky daného typu sa počítadlo vynuluje. Číslo „vety“ má potom tvar

`⟨číslo logickej jednotky⟩.⟨číslo vety v rámci danej jednotky⟩.`

Napríklad, ak má parameter `⟨číslovať v rámci⟩` hodnotu `section` a oddiel je najvyššou úrovňou logického členenia, tretia veta druhého oddielu má číslo 2.3 a prvá veta tretieho oddielu má číslo 3.1. Ak má tento parameter hodnotu `section`, ale najvyššou úrovňou logického členenia je kapitola, potom napríklad piata veta šiesteho oddielu druhej kapitoly má číslo 2.6.5.

Ale pozor: tieto dva nepovinné parametre nemožno použiť súčasne! Dôvod je ten, že pri použití parametra `⟨číslovať spoločne s⟩` sa aj parameter `⟨číslovať v rámci⟩` prestaví na hodnotu, ktorú má nastavenú ten typ „vety“, na ktorý sme sa v parametri `⟨číslovať spoločne s⟩` odvolávali.

Príkazom `\newtheorem` sa vlastne zadefinuje prostredie, ktoré možno používať v tele dokumentu bežným spôsobom, t.j. tak ako v nasledujúcom kuse kódu.

```
...
\newtheorem{veta}{Veta}
...
\begin{document}
...
\begin{veta}
Toto je veľmi dôležitá matematická veta.
\end{veta}
...
\end{document}
```

Príkaz na začatie každého z týchto nami definovaných prostredí má aj jeden nepovinný parameter (ktorý sa píše za povinný parameter). Tento možno využiť v prípade, že daná „veta“ má aj nejaký názov, prípadne ak je potrebné uviesť jej autora a podobne.

Ďalšiu funkcionalitu súvisiacu s touto problematikou pridáva balík `amsthm`. Ten okrem iného obsahuje aj preddefinované prostredie `proof`, ktoré možno využiť na sádzanie dôkazov. Stačí ho použiť hneď vzápätí po ukončení prostredia „vety“.

2.11 Krížové odkazy

Jedným z hlavných dôvodov, prečo sa v LaTeXu dá číslovať takmer čokoľvek, je možnosť odkazovania na jednotlivé logické jednotky pomocou ich čísla. Treba si zapamätať jeden základný princíp: za žiadnych okolností do kódu nepísať „natvrdo“ priamo čísla jednotlivých objektov. Dokument sa totiž neskôr môže zmeniť a tieto čísla potom nemusia zodpovedať realite.

Správnou metódou je použitie tzv. krížových odkazov: po dokumente možno „rozvešať nálepky“ zodpovedajúce číslovanej logickej jednotke, v ktorej sa daná nálepka nachádza. Ak sa nálepka nachádza súčasne vo viacerých číslovaných jednotkách, zodpovedá tej z nich, ktorá je v kóde najhlbšie vnorená. Ak sa napríklad nálepka nachádza v prostredí `equation`, ktoré je súčasťou nejakého číslovaného oddielu, nálepka zodpovedá rovnici, nie oddielu. Nálepku zodpovedajúcu oddielu však možno pridať napríklad tak, že nálepku „zavesíme“ priamo pod príkaz `\section` (alebo kamkoľvek inam, kde nebude súčasťou nejakej hlbšie vnorenej číslovanej jednotky). Na nálepky potom možno kdekoľvek v texte odkazovať.[7]

Na „zavesenie nálepky“ slúži príkaz `\label`. Jeho jediným parametrom je identifikátor danej „nálepky“. Odvolávanie sa na existujúce „nálepky“ sa potom deje pomocou príkazu `\ref`, ktorého jediným parametrom je tiež identifikátor „nálepky“. Celý mechanizmus je ilustrovaný na nasledujúcom príklade.

```
...
\section{Nejaký oddiel}
\label{mudrosti}
Tu je nejaký veľmi múdry text.
...
\section{Nejaký iný oddiel}
Viď oddiel \ref{mudrosti}.
...
```

[7] A aj to je jedným z dôvodov, prečo sa odporúča zdrojový súbor preložiť dvakrát, prípadne až trikrát za sebou.

Kapitola 3

Ďalšie možnosti

Aj keď zvládnutie predchádzajúcich dvoch kapitol by malo začiatočníkovi umožniť vytvárať s použitím LaTeXu jednoduchšie dokumenty, je dobré mať na pamäti, že potenciál LaTeXu tým nie je ani zďaleka vyčerpaný. V tejto stručnej kapitole preto iba naznačíme niektoré jeho pokročilejšie možnosti s cieľom poskytnúť čitateľovi vhodné východiskové body pre ďalšie samoštúdium a experimentovanie.

Úvodný oddiel tejto kapitoly obsahuje zmienku o niektorých typografických zvláštnostiach, ktoré sa vyskytujú pri písaní textov v oblasti formálnych jazykov a automatov. V nasledujúcich partiách text v hrubých rysoch pojednáva o niektorých možnostiach na prácu s grafikou, na tvorbu zoznamov použitej literatúry a na tvorbu prezentácií. Nepôjde však o podrobný výklad – čitateľ bude zakaždým len usmernený na vhodné zdroje informácií. V závere tejto kapitoly uvádzame komentovaný zoznam niektorých zdrojov ďalších informácií.

3.1 Niektoré typografické špeciality z formálnych jazykov

Teória formálnych jazykov a automatov, podobne ako väčšina ďalších oblastí matematiky, je spojená s určitými notačnými špecifikami. V nasledujúcom ukážeme, ako sa s niektorými z nich v LaTeXu vyrovnať.

Jedným z najčastejších takýchto špecifík sú symboly pre triedy jazykov, kde sa často zvyknú používať „veľké písané písmená“ – napríklad bezkontextové jazyky sa zvyknú označovať $\mathscr{L}_{CF}$ a regulárne jazyky $\mathscr{R}$. Ukážeme, ako vysádzanie takýchto symbolov v LaTeXu dosiahnuť.

Takýto font v matematike nie je veľmi bežný a často sa preto používa jeho napodobnenie kaligrafickými fontmi (v matematickom móde príkaz `\mathcal`). Výsledok však nie je úplne najkrajší – napríklad `\mathcal{L}` vysádza $\mathcal{L}$, čo sa na zamýšľaný výsledok podobá len vzdialene.

Dokonalé riešenie ponúka balík `mathrsfs` a v ňom definovaný príkaz `\mathscr`. Skutočne, `\mathscr{L}` vysádza $\mathscr{L}$ a `\mathscr{R}` vysádza $\mathscr{R}$.

V teórii formálnych jazykov sa prázdne slovo často označuje gréckym písmenom epsilon. Tu len spomeňme, že symbol pre epsilon existuje v dvoch základných verziách – príkaz `\epsilon` vysádza ϵ a príkaz `\varepsilon` vysádza ε.

Ďalším špeciálnym symbolom, ktorý treba z času na čas vysádzať v textoch zaoberajúcich sa teóriou automatov, je symbol pre cent, používaný najmä ako zarážka na páskach Turingových strojov, lineárne ohraničených automatov a podobne.

V rôznych LaTeXových balíkoch sa vyskytujú rôzne verzie tohto symbolu, a to predovšetkým pre textový mód. Potreba vysádzať cent pri písaní dokumentov z oblasti teórie automatov však vznikne väčšinou v matematickom móde, ale aj tam možno použiť symboly pre textový mód – stačí ich obaliť do príkazu `\textrm`. V nasledujúcej tabuľke uvádzame prehľad rôznych verzií symbolu pre cent, zodpovedajúcich príkazov a balíkov, v ktorých ich možno nájsť.

Znak	Príkaz	Balík
¢	`\textcent`	`textcomp`
¢	`\textcentoldstyle`	`textcomp`
¢	`\cent`	`wasysym`

Niektoré balíky obsahujú aj symbol pre cent určený priamo pre matematický mód – tie však zmenia font používaný v celom dokumente. Aj keď sa to dá určitými trikmi obísť, výsledok nebýva úplne najlepší.

Asi najzložitejšou úlohou spadajúcou do tohto oddielu je kreslenie stavových diagramov automatov. Nebudeme tu podrobne rozoberať rozličné metódy, ale iba naznačíme niektoré možné prístupy. Prvým je nakresliť automat v nejakom grafickom editore (najlepšie v nejakom, ktorý dokáže produkovať vektorovú grafiku) a výsledok potom importovať do dokumentu pomocou príkazov na vkladanie obrázkov. Vhodnou voľbou môže byť napríklad editor Ipe, ktorý sa okrem iného vyznačuje aj podporou TeXu. Ak teda obrázok obsahuje text, možno ho vysádzať rovnakým fontom ako zvyšok dokumentu.

Druhou možnosťou je použiť program Metapost, ktorý je súčasťou väčšiny bežných TeXových distribúcií. Metapost je program na tvorbu vektorovej grafiky pomocou špeciálneho jazyka – ide teda o metódu určenú predovšetkým pre používateľov bez väčších umeleckých vlôh, zato však s dobrou orientáciou v zdrojovom kóde. Metapost je istým spôsobom zviazaný s LaTeXom, nie však s pdfLaTeXom. Pri práci s pdfLaTeXom je ale možné použiť Metapost a LaTeX na vygenerovanie súboru vo formáte Encapsulated Postscript, ten skonvertovať do PDF[1] a vložiť do dokumentu príkazom na importovanie grafiky. Viac o Metaposte sa možno dočítať napríklad v [2].

Vektorová grafika sa dá tvoriť dokonca aj priamo z LaTeXového zdrojového kódu, napríklad pomocou balíku TikZ. Viac informácií o tejto metóde možno nájsť napríklad v [16].

3.2 LaTeX a grafika

Metódy spomenuté vyššie ako vhodné na kreslenie stavových diagramov automatov možno použiť na tvorbu v zásade akejkoľvek vektorovej grafiky. Tento oddiel sa preto venuje výlučne už len importovaniu (vektorovej alebo rastrovej) grafiky do vytváraného dokumentu.

Na importovanie grafiky slúži príkaz `\includegraphics` z balíku `graphicx`. Balík `graphicx` sa načítava pomocou príkazu

```
\usepackage[⟨ovládač⟩]{graphicx},
```

kde `⟨ovládač⟩` je v LaTeXu názov programu na konverziu z formátu DVI do výstupného formátu – možné hodnoty sú `dvips` a `dvipdfm`. V pdfLaTeXu treba ako hodnotu tohto parametra uviesť `pdftex`.

Samotný príkaz `\includegraphics` potom ako povinný parameter berie názov vkladaného súboru a ako nepovinný parameter rôzne nastavenia. Podporované formáty vkladaného grafického súboru sú v pdfLaTeXu PDF, JPEG a PNG, v LaTeXu iba EPS (Encapsulated PostScript).

Ešte sa zmieňme o tom, že obrázky vkladané do LaTeXu sa zvyknú obaľovať do prostredia `figure`. To optimalizuje umiestnenie obrázku v dokumente, umožňuje číslovanie obrázkov a podobne. Podobné prostredie existuje aj pre tabuľky a má názov `table`.

Viac sa o práci s grafikou v LaTeXu možno dočítať napríklad v [14] alebo v [16].

3.3 Bibliografické odkazy

Dôležitou súčasťou väčšiny dokumentov sú zoznamy použitej literatúry a bibliografické odkazy. Teraz v stručnosti naznačíme metódu ich vytvárania v LaTeXu.

V LaTeXu existuje prostredie `thebibliography`, ktoré slúži na vypísanie zoznamu použitej literatúry. Jednotlivé záznamy sa pridávajú pomocou príkazu `\bibitem`, ktorého parametrom je

[1] Napríklad pomocou programu `epstopdf`, ktorý býva súčasťou TeXových distribúcií.

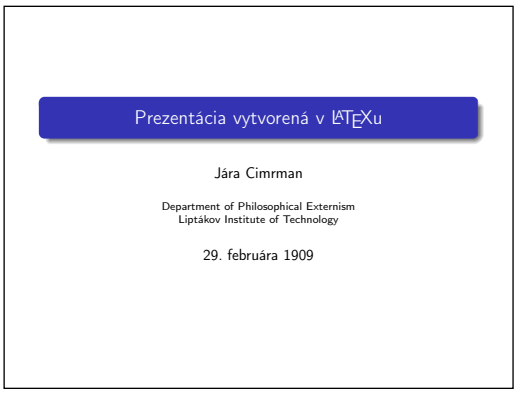
identifikátor daného záznamu. Za týmto príkazom nasleduje daný bibliografický záznam, na ktorý sa potom v texte možno odvolať pomocou príkazu `\cite`, ktorého parametrom je spomínaný identifikátor. Celkovo je to podobné práci s krížovými odkazmi.

Súčasťou väčšiny TEXových distribúcií býva aj program BibTEX, ktorý umožňuje zautomatizovať aj vypisovanie jednotlivých bibliografických záznamov, či utriedenie jednotlivých záznamov v zozname použitej literatúry. Viac sa o BibTEXu možno dozvedieť napríklad v [16].

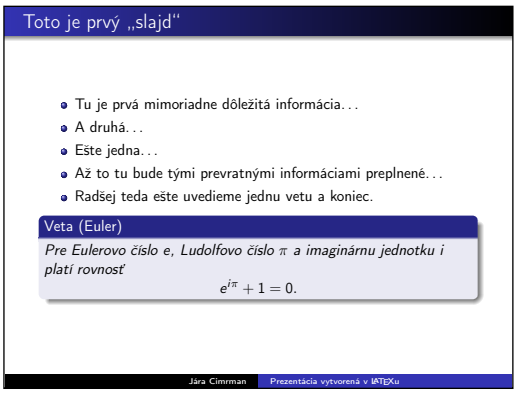
3.4 Tvorba prezentácií v LATEXu (Beamer)

Niektoré využitia LATEXu sa môžu na prvé počutie zdať pomerne prekvapivé až neuveriteľné. Takou sa môže javiť aj skutočnosť, že LATEX umožňuje tvorbu prezentácií, ktoré vyzerajú omnoho profesionálnejšie, než prezentácie vytvorené pomocou bežných WYSIWYG editorov. Tvorba prezentácií v LATEXu je navyše mimoriadne výhodná v prípadoch, keď je do „slajdov“ potrebné zahrnúť aj matematické vzorce. A nakoniec, tvorba prezentácií v LATEXu môže aj ušetriť čas – napríklad prezentáciu na obhajobu záverečnej práce možno z veľkej časti vytvoriť skopírovaním vhodných častí zdrojového kódu textu práce.

Na tvorbu prezentácií v LATEXu slúži trieda dokumentov Beamer, ktorá je súčasťou väčšiny moderných TEXových distribúcií. Beamer prichádza s viacerými preddefinovanými štýlmi a jeho použitie je veľmi jednoduché – po niekoľkých nastaveniach v preambule dokumentu už stačí v jeho tele každý „slajd“ obaliť do prostredia `frame`. Výsledok môže vyzerať napríklad tak, ako na obrázku 3.1.



(a) Úvodný „slajd“.



(b) Ďalší „slajd“.

Obr. 3.1: Príklady „slajdov“ vytvorených v LATEXu pomocou triedy dokumentov Beamer.

Viac sa o Beameri možno dozvedieť napríklad na stránkach [15], vyčerpávajúci zoznam štýlov možno nájsť na stránkach [1].

3.5 Zdroje ďalších informácií

Ako bol čitateľ už viackrát upozornený, tento text nemožno v žiadnom prípade považovať za vyčerpávajúci úvod do LATEXu a informácie v ňom uvedené pravdepodobne nepostačujú ani na všetky bežné účely. Preto na tomto mieste uvádzame prehľad zdrojov, v ktorých možno nájsť ďalšie informácie. Mnohé z nich už boli v texte spomínané.

Solídne praktické základy LATEXu možno získať z dnes už klasického úvodného textu, ktorý je voľne prístupný na internete, a to vo viacerých jazykových verziách:

- Anglická verzia má názov *The Not So Short Introduction to LATEX 2ε* [10].

- Český preklad *Ne úplně nejkratší úvod do formátu LaTeX 2ε* [9] je pomerne aktuálny.
- Slovenský preklad *Nie príliš stručný úvod do systému LaTeX 2ε* [8] z roku 2002 je už o niečo zastaranejší.

Výborným zdrojom (aj pomerne pokročilých) informácií je kniha, ktorá vzniká na projekte Wikibooks [16], a ktorú možno nájsť na adrese

- `http://en.wikibooks.org/wiki/LaTeX`.

Pre pokročilých používateľov LaTeXu je tiež vhodná kniha

- *The LaTeX Companion* [7],

ktorá už však nejaký ten rok má.

Užitočnými portálmi sú stránky TeX Users Group – TUG – a ich česko-slovenská obdoba CSTUG, ktoré možno nájsť na nasledujúcich adresách:

- `https://tug.org/`,
- `http://www.cstug.cz/`.

Nenahraditeľnými pomôckami používateľov LaTeXu sú tabuľky symbolov a rôzne podobné „ťaháky“. V texte sme spomínali tieto dve:

- *The Comprehensive LaTeX Symbol List* [13] je vyčerpávajúcim zoznamom symbolov z rôznych LaTeXových balíkov.
- *TeX Cookbook* [6] obsahuje príkazy podporované priamo TeXom.

Klasickou učebnicou „čistého“ TeXu je Knuthova kniha

- *The TeXbook* [5].

Niektorými konkrétnymi aspektmi práce s LaTeXom sa zaoberajú nasledujúce užitočné voľne prístupné materiály a webové lokality:

- [14] sumarizuje metódy použitia importovanej grafiky v LaTeXových dokumentoch,
- [2] obsahuje informácie o práci s Metapostom,
- [15] je rýchlym úvodom do tvorby prezentácií pomocou triedy dokumentov Beamer,
- [1] je obsiahlou galériou rôznych štýlov prezentácií podporovaných Beamerom.

Riešenia problémov možno hľadať na rôznych komunitných stránkach, či fórach. Tu spomenieme dve webové lokality:

- `http://www.latex-community.org/` – komunitná stránka a fórum venované LaTeXu,
- `http://tex.stackexchange.com/` – StackExchange stránky venované TeXu a LaTeXu.

Literatúra

[1] I. Blanes. Beamer theme gallery. http://deic.uab.es/~iblanes/beamer_gallery/. Dátum prístupu: 15.9.2013.

[2] J.D. Hobby. Metapost: a user's manual, version 1.503, 2013. https://www.tug.org/docs/metapost/mpman.pdf.

[3] J.M. Klymak. The pdfTEX FAQ, version 0.10, 1998. http://www.math.duke.edu/computing/Docs/pdfTeX-FAQ.pdf.

[4] D.E. Knuth. Knuth: Computers and Typesetting. http://www-cs-faculty.stanford.edu/~uno/abcde.html. Dátum prístupu: 5.9.2013.

[5] D.E. Knuth. *The TEXbook*. Addison-Wesley, second edition, 1984.

[6] MathPro Press, Inc. TEX Cookbook, 1989. http://www.math.upenn.edu/tex-stuff/cookbook.pdf.

[7] F. Mittelbach and M. Goossens. *The LATEX Companion*. Addison-Wesley, second edition, 2004.

[8] T. Oetiker, H. Partl, I. Hyna, and E. Schlegl. Nie príliš stručný úvod do systému LATEX 2_ε, 2002. http://mirrors.ctan.org/info/lshort/slovak/Slshorte.pdf.

[9] T. Oetiker, H. Partl, I. Hyna, and E. Schlegl. Ne úplně nejkratší úvod do formátu LATEX 2_ε, 2011. http://mirrors.ctan.org/info/lshort/czech/lshort-cs.pdf.

[10] T. Oetiker, H. Partl, I. Hyna, and E. Schlegl. The not so short introduction to LATEX 2_ε, 2011. http://mirrors.ctan.org/info/lshort/english/lshort.pdf.

[11] P. Olšák. CSTeX – česká a slovenská podpora TeXu. http://math.feld.cvut.cz/olsak/cstex.html. Dátum prístupu: 8.9.2013.

[12] P. Olšák. Proč nerad používám LATEX. *Zpravodaj Československého sdružení uživatelů TEXu*, 7(1–2):89–99, 1997. http://petr.olsak.net/ftp/olsak/bulletin/nolatex.pdf.

[13] S. Pakin. The comprehensive LATEX symbol list, 2009. http://www.tex.ac.uk/tex-archive/info/symbols/comprehensive/symbols-a4.pdf.

[14] K. Reckdahl. Using imported graphics in LATEX and pdfLATEX, version 3.0.1, 2006. ftp://ftp.tex.ac.uk/pub/tex/info/epslatex.pdf.

[15] R. Rostamian. A Beamer Quickstart. http://www.math.umbc.edu/~rouben/beamer/. Dátum prístupu: 15.9.2013.

[16] Wikibooks. LATEX – Wikibooks, open books for an open world. http://en.wikibooks.org/wiki/LaTeX. Dátum prístupu: 5.9.2013.