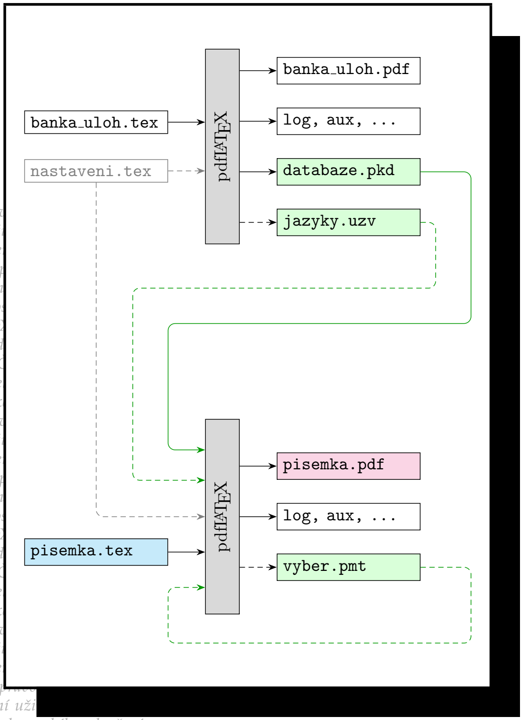
CSTUG



ZPRAVODAJ

Československého sdružení uživatelů TEXu

1–2

2020

ISSN 1211-6661
Tištěná verze

ISSN 1213-8185
Online verze

Ročník 30

OBSAH

Zpravodaj Československého sdružení uživatelů TEXu je vydáván v tištěné podobě a distribuován zdarma členům sdružení. Vydaná čísla Zpravodaje v elektronické podobě (PDF) jsou bezodkladně veřejně vystavena na webové adrese `https://www.cstug.cz/`.

Své příspěvky do Zpravodaje můžete zasílat v elektronické podobě, nejlépe jako jeden archivní soubor (`.zip, .arj, .tar.gz`), na e-mailovou adresu `bulletin@cstug.cz`. Nezapomeňte přiložit všechny soubory, které dokument načítá (s výjimkou standardních součástí TEX Live), zejména v případě, kdy Vás nelze kontaktovat e-mailem.

ISSN 1211-6661 (tištěná verze)
ISSN 1213-8185 (online verze)

Úvodník

Petr Sojka

Milé čtenářky a čtenáři,

jsem rád, že dostáváte a otevíráte včas první letošní dvoučíslo Zpravodaje, plné zajímavých informací ze světa kvalitní typografie sázecím systémem TEX. Dovolte mi několik poznámek k obsahu čísla, a zejména pak k loňské návštěvě Grand Wizarda, tedy autora TEXu Donalda Ervina Knutha, v Brně.

Toto číslo začíná článkem o novém balíku na generování zadání písemek. Přesto, že podobná úloha již byla v několika balících řešena, článek ilustrativně a názorně ukazuje výhody, které takové cvičení pro autora a řešení jeho potřeb přináší, kromě seznámení se s makroprogramováním. Taková řešení umožňují řešení přesně na míru a dobrou udržitelnost v čase. Po podobném řešení dříve nebo později sáhne většina pedagogů, a poptávka po nich v dnešní koronavirové době a potřebě distančního zkoušení roste. Na TEXu jsou postavena pro jeho algoritmizovatelnost mnohá řešení: informační systém MU například generování písemek podporuje již víc než dekádu a je rutinně využíván tisíci zaměstnanci Masarykovy univerzity.

Další článek Vítka Novotného posunuje zájemce o minimalistické značkování na trůn, za který je autorem a širokou komunitou uživatelů považován jazyk Markdown. V článku se dozvíme jak TEX a Lua spolu spolupracují na sazbě Markdown dokumentů ať již z příkazové řádky nebo při sazbě TEXových dokumentů s částmi v Markdown.

Jak načíst a zpracovat dat vytvořená v tabulkovém editoru ukazuje Honza Šustek, jako příklad makroprogramování na úrovni PlainTEXu.

V říjnu 2019 na pozvání Fakulty informatiky Masarykovy univerzity (FI MU) navštívil Brno sám autor TEXu, Grand Wizard, Donald Ervin Knuth (DEK). Měl jsem tu čest ho po dobu jeho desetidenního pobytu v Brně provázet a koordinovat přípravy provedení české premiéry jeho varhanního opusu Fantasia Apocalyptica, kvůli které primárně přijel.[1]

DEK měl na FI MU dvě přednášky, které vám formou transkriptu přináší článek Tomáše Szaniszla. Ač jsem Knutha provázel již v roce 1996 při udílení čestného doktorátu od MU a loni ho navštívil při TUG 2019 u jeho varhan doma v Kalifornii, z každé z obou brněnských přednášek jsem si odnesl několik nových, jen tak bokem zmíněných, mouder, které si připomínám a reflektuji ještě mnoho let poté. Pozitivní afinita při aktivním hledání svých chyb a radost

[1] Záznam české premiéry varhanního oratoria Fantasia Apocalyptica si můžete poslechnout na adrese `https://youtu.be/wk7dEKMPP68`.

doi: 10.5300/2020-1-2/1

Obrázek 1: Donald Ervin Knuth hraje na své domácí varhany

při jejich nalezení, začínání dne těmi nepříjemnými, ale nutnými úkoly, hledání maximální diverzity při výběru svých činností[2], získání maximální koncentrace a výkonu minimalismem přepínání činností a prací v dávkách, minimalizací komunikace (emailů), a při tom všem obrovské soustředění a laskavost na okolí, se kterým komunikuje kontaktně. Doufám, že si svou inspiraci hodnou následování v transkriptech najdete také! Obrázek o atmosféře přednášek si můžete udělat z přiložené fotoreportáže; fotky z 8. a 9. 10. jsou dílem Martiny Morávkové, ostatní jsem fotil sám.

Číslo uzavírají přeložené ukázky ze seriálu *Glisterings* Petera Wilsona, tentokráte věnované variantám způsobů opakování textu číslovaných prostředí.

Summary: Introductory Word

Editorial introduces the content of the Zpravodaj issue, and the author makes his reminiscence comments from personal communication with about Don's trips The recent visit of the Grand Wizard in Brno is also reported with insight.

Go forth and participate in CSTUG to make the bright future of TEX & Friends a reality! *You can!*

Masarykova univerzita, Fakulta informatiky, Botanická 68a, 602 00 Brno
sojka@fi.muni.cz

[2]Místo několikáté návštěvy varhan DEK raději volil návštěvu Muzea romské kultury.

Obrázek 2: U DEK v Palo Alto při TUG 2020

Obrázek 3: DEK ve své hlavní pracovně

Obrázek 4: Dračí křivka (s chybičkou) ve vstupní hale DEK

Obrázek 5: Motto u vstupních dveří do domu DEK

Obrázek 6: Na exkurzi v Muzeu romské kultury v Brně

Obrázek 7: Knuthova přednáška 8. 10. 2019 na FI MU: P $\neq$ NP?

Obrázek 8: Podepisování po přednášce 8. 10. 2019 na FI MU

Obrázek 9: Focení s návštěvníky přednášky 8. 10. 2019 na FI MU

Obrázek 10: Focení s návštěvníky přednášky a akademiky FI MU 8. 10. 2019

Obrázek 11: Focení s návštěvníky přednášky 8. 10. 2019: prorektor Michal Bulant

Obrázek 12: Zleva: Honza Šustek, Jiří Rybička, Donald Ervin Knuth, Petr Sojka a Tomáš Hála, Brno, FI MU 8. 10. 2019

Obrázek 13: Knuthova přednáška 9. 10. 2019 na FI MU: ukázky z Fantasie

Obrázek 14: Knuthova přednáška 9. 10. 2019 na FI MU uváděná děkanem Jiřím Zlatuškou

Obrázek 15: S dalším držitelem Turingovy ceny Dana Scottem po jejich přednáškách 9. 10. 2019 v Brně

Obrázek 16: Příprava varhanního představení: DEK a Jan Rotrekl

Obrázek 17: 11. 10. 2019 Domluva DEK s Janem Rotreklem u varhan u Jezuitů

Obrázek 18: 11. 10. 2019 Soustředění před koncertem

Obrázek 19: 11. 10. 2019 Závěr Fantasie Apocalyptiky: AMEN!

Vícejazyčné pseudonáhodné generování písemných testů z databází

Marian Genčev

Cílem tohoto článku je popis třídy `ngt.cls`, kterou autor vytvořil s cílem zjednodušení přípravy písemných testů pro pedagogy disponující běžnou uživatelskou obeznámeností s LaTeXem. Popisované zjednodušení přípravy spočívá především v pseudonáhodném generování testů z předem připravené databáze úloh. K dalším přednostem vytvořené soustavy lze řadit snadnost ovládání pro koncového uživatele a možnost vytvoření verze s výsledky, nebo bez nich. Zápis zadání úloh je uzpůsoben tak, aby bylo možno pracovat s libovolným počtem jazykových verzí v jediném zdrojovém souboru s jednoduchým přepínáním mezi nimi.

1. Úvod

Běžnou součástí pedagogické činnosti na většině typů škol je zjišťování míry pochopení probraného učiva pomocí didaktických testů. Vhodné testy má pedagog zpravidla připraveny již napřed a při menším počtu studentů a s jistou dávkou opatrnosti si může s několika verzemi vystačit.

V případě většího počtu studentů se však již pedagogové mohou setkat při sestavování testů s obtížemi plynoucími z potřeby značného počtu jejich verzí. Charakteristickým problémem jsou třeba situace u vyučovaných celofakultních kursů, jež nejsou u studentů příliš populární (např. matematika nebo fyzika v prvním semestru). V takových případech je procento opakujících studentů poměrně vysoké, čemuž je nutno přizpůsobit strategii vytváření písemných testů za účelem zamezení konfrontace opakujícího studenta s téměř stejným, nebo dokonce stejným zadáním.

Vzhledem k tomu, že studenti rozšiřují písemné testy prostřednictvím sociálních sítí nebo jinak elektronicky, je nutno vytvářet testy buď zcela nové, nebo přeskupit úlohy mezi již existujícími testy. V souvislosti s tímto fenoménem lze v určitém smyslu označit za zajímavý projekt nazvaný `Math4U` provozovaný VŠB-TUO, který studentům nabízí webové rozhraní pro generování testů k procvičování látky, resp. pedagogům možnost náhodně vygenerovat písemný test z vybraných otázek.[1] Všeobecně známou výhodou takového přístupu je, že vyučující nemusí disponovat enormním počtem statických verzí písemných testů, ani nemusí provádět ruční přeskupení úloh. Dle propozicí uvedených na webu projektu obsahuje

[1] `http://math4u.vsb.cz/`

 doi: 10.5300/2020-1-2/12

databáze celkem asi 4 000 úloh z různých oblastí matematiky (středoškolské i vysokoškolské).

Jistou alternativou k `Math4U` je přístup prezentovaný v tomto článku. V něm se zaměříme na popis relativně jednoduchého systému maker v pdfLaTeXu, který je vhodný k automatickému generování písemných testů. Ve srovnání s `Math4U` může být počáteční nevýhodou vytvoření dostatečně velké databáze v jazykových verzích, s nimiž budeme pracovat. Na druhou stranu máme možnost zcela kontrolovat obsah databáze, přirozeně rozšiřovat databázi o další úlohy nebo nové jazykové verze, resp. možnost si kdykoliv výstup nebo formát upravit podle vlastních představ.

Princip, který používáme při náhodném generování testů, spočívá na vytvoření databáze typových úloh s t dílčími databázemi $d_1, \ldots, d_t$, přitom z každé z nich bude náhodně[2] vybrána úloha. Z takto vybraných úloh sestavíme písemný test, který bude pro studenta při dostatečně početné databázi úloh vždy poměrně nový.

2. Existující přístupy a motivace

V současné době existuje v distribuci TeXu několik balíčků, které podporují náhodné generování písemných testů z databází.[3] Mezi nimi vynikají např.

`AcroTeX`, `esami`, `examdesign`, `probsoln`.

Uvedené balíky využívají pro generování pseudonáhodných čísel makra D. Arsenaua oficiálně k dispozici na `https://ctan.org/pkg/random`, o nichž je známo, že generují nové pseudonáhodné číslo při opětovné kompilaci až při změně hodnoty minutového čítače `\time`. Protože se chceme takovému omezení vyhnout, je v třídě `ngt.cls` pro generování pseudonáhodných čísel využito základní funkcionality pdfTeXu,[4] primitivu `\pdfuniformdeviate`, viz [1, s. 40]. Pro vygenerování pseudonáhodného celého čísla z množiny $\{0, 1, \ldots, n-1\}$ tak stačí psát

```
1 \pdfuniformdeviate n
```

přičemž toto číslo není generováno pouze jednou za minutu, ale každou mikrosekundu.

Kromě zmíněného nedostatečně častého generování pseudonáhodného čísla ve výše uvedených balících je pro precizní výstup podle požadavků uživatele nutno nastavit řadu parametrů, což vyžaduje prostudování někdy i velmi obsáhlých manuálů (např. balík `AcroTeX`). Vzhledem ke skutečnosti, že studium rozsáhlejších manuálů nebo akceptování ústupků oproti naší představě může být málo

[2] Slovem „náhodně“ zde i v dalším textu myslíme vždy „pseudonáhodně“.

[3] Podrobněji viz třeba `https://ctan.org/topic/exam`.

[4] Nabízené řešení popsané v dalším je proto vhodné pro uživatele kompilující pdfTeXem s verzí 1.30.0 nebo vyšší.

uspokojující, rozhodl se autor sestavit vlastní makra, která slouží popisovanému účelu a jejich případná další modifikace se jeví jako snadná. Motivací pro jejich vytvoření bylo zformování takového prostředí pro koncového uživatele, aby zápis úloh a řešení, resp. následné generování písemných testů z nich, bylo velmi jednoduché. K přednostem lze řadit tyto vlastnosti:

- snadnost zápisu úloh a doprovodných informací do databází,
- kontrola nad obsahem databáze i formou výstupu,
- generování nové náhodné písemky nezávisle na hodnotě čítače `\time`,
- možnost přidání libovolného počtu jazykových verzí,
- intuitivní přepínání mezi jednotlivými verzemi výstupu,
- manuál umožňující rychlé pochopení ovládání i pro začátečníky.

Autor věří, že popisovaná třída může být pro běžné uživatele LaTeXu užitečným nástrojem při generování testů, zatímco zkušeným TeXpertům poslouží třeba jako východisko pro její další úpravy, které jsou jistě možné v mnoha ohledech. Pro oba dva jmenované účely je zájemcům k dispozici zdrojový kód `ngt.cls` na webové stránce `http://robimematiku.cz/bebechy/`.

Zbytek článku lze rozdělit do dvou logických oddílů. První má formu uživatelského manuálu a popisuje, jakým způsobem je možno třídu `ngt.cls` použít. Druhý oddíl je věnován implementaci a popisujeme v něm zvolené technické řešení.

3. Uživatelský manuál třídy `ngt.cls`

Tato část je věnována uživatelskému seznámení s třídou `ngt.cls`, která disponuje dvěma volbami – `databaze`, určenou k sazbě úloh v dílčích databázích, a `pisemka`, určenou k sazbě náhodně generovaného testu z úloh uvedených v připraveném souboru databází. Jejich aktivace je obvyklá, tj. stačí psát jednu z možností

```
2 \documentclass[databaze]{ngt}
```

nebo

```
3 \documentclass[pisemka]{ngt}
```

Pro správné fungování třídy `ngt.cls` je vyžadována instalace třídy `article` a balíků `lmodern`, `inputenc`, `fontenc`, `geometry` a `babel`, které jsou třídou načítány (detaily viz sekci 4 popisující implementaci). Pro sazbu databáze i písemky předpokládáme, že zdroj bude zapisován v kódování `utf8`.

Než se v dalším blíže vyjádříme k jednotlivým volbám, připojíme ještě krátkou technickou poznámku. Při popisu příkazů uvádíme vždy název příkazu a jeho syntaxi, přitom na konci řádku bude uvedeno `[P]` nebo `[T]` značící, že daný příkaz lze použít jen v preambuli, resp. jen v textové části, tj. za `\begin{document}`. Jsou-li uvedeny obě volby, je možno příkaz použít v obou částech dokumentu. Pokud je v syntaxi za názvem příkazu použit místo černé barvy odstín šedé,

označuje tato skutečnost variabilitu syntaxe, která je v popisu příkazu vysvětlena. Např. zápis

```
\prikaz #1/#2                                                    [P]
```

uvozuje uživatelský popis příkazu `\prikaz`, který lze použít pouze v preambuli a má jeden povinný parametr. Ten je možno v případě potřeby doplnit nepovinným druhým argumentem za lomítkem.

3.1. Volba `databaze`

3.1.1. Vytvoření databáze – základní přehled

Při aktivní volbě `databaze` jsou uživateli k dispozici prostředky uvedené v Tabulce 1.

`\databaze` `\subdatabaze`	zahájení sazby příslušné databáze úloh nebo její části
`\uloha` `\reseni`	zápis zadání úlohy, resp. jejího řešení
`\jazyky`	načtení jazyků, v nichž budeme text zapisovat
`\jazyk`	specifikace jazyku, v němž bude text vysázen
`\vyres`	ovládání zobrazení výsledků

Tabulka 1: Příkazy dostupné při volbě `databaze`

Pro základní představu uvádíme příklad struktury souboru `banka_uloh.tex`, v němž `<uloha_m_n>` značí zadání a eventuální řešení úlohy, která budou později zapsána pomocí příkazu `\uloha`, eventuálně `\reseni`, viz následující odstavce.

banka_uloh.tex

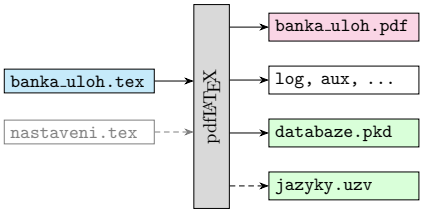
```
\documentclass[databaze]{ngt}
  \jazyky <deklarace pouzivanych jazyku>
  \jazyk  <specifikace jazyku>
  \vyres  <0/1>

\begin{document}
\databaze <Nazev_databaze_1>/<bodovy_zisk_1>b
  <uloha_1_1>
  <uloha_1_2>
  ...
  <uloha_1_n>

```

```
\databaze <Nazev_databaze_2>/<bodovy_zisk_2>b
  ...
\databaze <Nazev_databaze_3>/<bodovy_zisk_3>b
  ...
\end{document}
```

Podstatným výstupem kompilace souboru databází pomocí

```
pdflatex banka_uloh.tex
```

je soubor `banka_uloh.pdf` obsahující zapsané úlohy a řešení ve zvoleném jazyce a v podobě, kterou budou mít na písemce generované později. Uživatel tak má náhled do souhrnu úloh a lepší kontrolu nad sázeným obsahem. Při kompilaci vznikají ještě další pracovní soubory s příponami `pkd` a `uzv` (viz Obrázek 1), které jsou třídou `ngt` načítány při volbě `pisemka` (viz Obrázek 3). Soubory s příponami `pkd` a `uzv` mají vždy stejný název nezávisle na názvu kompilovaného souboru (zde `banka_uloh.tex`). Pro běžného uživatele a pro popis příkazů v této části textu nejsou důležité, a proto se o nich přesněji zmíníme až v sekci o implementaci.



Obrázek 1: Výstupy kompilace souboru `banka_uloh.tex`

O významu nepovinného souboru s uživatelským nastavením `nastaveni.tex` je podrobněji pojednáno v sekci 3.1.3.

3.1.2. Popis příkazů pro sazbu databáze

`\jazyky` `<jazyk_1>`/`<jzk_1>`,...,`<jazyk_s>`/`<jzk_s>` [P]
`\jazyk` `<jzk_i>` [P]

Vzhledem k tomu, že příkazy `\jazyky` a `\jazyk` jsou úzce svázány, vyložíme jejich užití současně v následujících odstavcích.

Pomocí příkazu `\jazyky` deklarujeme v preambuli zdrojového souboru názvy jazyků, v nichž budeme text databáze sázet (`<jazyk_1>`, ..., `<jazyk_s>`) a zástupné zkratky jazyků (`<jzk_1>`, ..., `<jzk_s>`), pomocí nichž budeme v případě vícejazyčné volby konkrétní jazyk v preambuli souboru `banka_uloh.tex` volat příkazem `\jazyk`.

Název jazyku a s ním svázaná zkratka musí být v deklaraci `\jazyky` odděleny lomítkem `/`. Koncovým separátorem příkazu `\jazyky` je konec řádku a v případě užití více jazyků je nutno dílčí jazykové deklarace oddělit čárkou. Je důležité poznamenat, že název jazyku `<jazyk_i>` musí být ve shodě s názvy jazyků balíku `babel`, který třída `ngt` načítá, a že mezera je v syntaxi příkazu `\jazyky` nežádoucím znakem. Další popis obou příkazů rozčleníme podle počtu načítaných jazyků.

Sazba textu databáze v jediném jazyce

V tomto případě lze deklaraci uvádět pouze ve tvaru

```
24 \jazyky <jazyk_1>
```

Tím je implicitně zajištěno načtení balíku `babel` s vybraným jazykem `<jazyk_1>`. Uvedení specifikace `\jazyk` se v tomto triviálním případě neočekává, protože jsme v jazykové deklaraci žádnou zkratku jazyku neuvedli. Chceme-li např. sázet databázi úloh ve slovenštině, píšeme

```
25 \jazyky slovak
```

Situace může být ovšem mírně odlišná u uživatelů sázejících text databáze prozatím v jediném jazyce, avšak v budoucnu zamýšlejících doplnit ještě alespoň jednu další jazykovou verzi. V tomto případě je vhodné již v přípravné jednojazyčné fázi zapisovat texty úloh do makra odpovídajícího příslušné jazykové verzi. Je proto lepší deklarovat

```
26 \jazyky <jazyk_1>/<jzk_1>
```

a text úlohy zapisovat užitím

```
27 \<jzk_1>{<text v jazyce jazyk_1>}
```

kde řetězec `<jzk_1>` je uveden v deklaraci `\jazyky` za lomítkem. Pokud budeme chtít psát např. text databáze v budoucnu vícejazyčně a máme aktuálně k dispozici prozatím třeba slovenskou verzi zadání, deklarujeme v preambuli

```
28 \jazyky slovak/sk
```

a text úlohy sázíme pomocí

```
29 \sk{Slovenská verzia...}
```

Případná specifikace jazyku sazby pomocí `\jazyk sk` je sice možná, ale nadbytečná, protože třída `ngt` ji načítá automaticky.

Konečně poslední situací jednojazyčné sazby textu databáze je forma privilegia poskytnutá uživateli preferujícímu češtinu. Pokud v preambuli souboru `banka_uloh.tex` neuvedeme deklaraci `\jazyky`, je tento stav ekvivalentní s deklarací `\jazyky czech`. Sázíme-li ale v jediném jazyce, kterým není čeština, explicitně zapíšeme do preambule `\jazyky <nejaky_jazyk>` (např. řádek 25), případně i jeho zkratku za lomítko (např. řádek 28). Tím uživatel zajistí načtení vzorů pro dělení slov odpovídajících deklarovanému jazyku.

Sazba textu databáze ve více jazycích

Zde je popis snadnější. Mohou nastat celkem dvě běžné situace. První z nich je uvedení obou příkazů `\jazyky` a `\jazyk`. Uveďme příklad, kdy bude uživatel sázet text databáze ve třech jazycích, němčině, češtině a angličtině. Jazyková deklarace bude mít formát

```
30 \jazyky german/de,czech/cz,english/en
```

čímž je zajištěno načtení `\usepackage[german,czech,english]{babel}` a uživatelské zpřístupnění příkazů `\de`, `\cz` a `\en`, pomocí nichž budeme zapisovat jednotlivé jazykové verze textu databáze, např.

```
31 \cz{Toto je česká verze.}%
32 \en{This is the english version.}%
33 \de{...und endlich auch die deutsche.}
```

Jazyková verze zobrazená ve výstupním souboru je určena specifikací `\jazyk`. Např. pro zobrazení anglické verze píšeme do preambule

```
34 \jazyky german/de,czech/cz,english/en
35 \jazyk  en
```

přitom se automaticky na začátek dokumentu zapíše `\selectlanguage{english}` pro načtení odpovídajících vzorů dělení slov. Platí, že třída `ngt` při specifikaci `\jazyk <jzk_i>` zapíše na začátek dokumentu `\selectlanguage{<jazyk_i>}` a není třeba se o tuto věc dále starat.

Je nutno upozornit, že zkratku jazyku `<jzk_i>` musíme volit s jistou opatrností tak, aby zpřístupněné makro `\<jzk_i>` nekolidovalo s již definovanými makry. Nebylo by např. vhodné zavádět zkratku `ge` pro německý jazyk, protože příkaz `\ge` je běžně užíván v matematické sazbě pro vysázení relace $\geq$.

Druhá situace, která může nastat, je případ, kdy sice uvedeme deklaraci `\jazyky`, ale již nikoliv specifikaci `\jazyk`. V takovém případě bude ve výstupním souboru zobrazena jazyková verze odpovídající prvnímu jazyku v pořadí uvedeném za `\jazyky`. Ve výše uvedeném příkladu by to znamenalo zobrazení varianty v německém jazyce.

`\databaze #1/#2b` [T]

Obecně předpokládáme, že písemka bude sestávat z t různých typů úloh. Pro pozdější rozlišení při generování náhodného testu proto každému typu přiřadíme samostatnou dílčí databázi d_i, $i = 1, \ldots, t$, jejíž sazbu zahájíme pomocí `\databaze`. Do argumentů příkazu jsou načítány tyto informace: název databáze (`#1`), číselná hodnota symbolizující bodové ohodnocení úloh dané databáze v písemném testu (`#2`). Koncovým separátorem argumentu `#2` je písmeno `b`, které současně symbolizuje, že se jedná o bodové ohodnocení. Třída `ngt.cls` detekuje konec dané databáze buď v místě následujícího nejbližšího příkazu `\databaze` nebo při dosažení konce souboru `banka_uloh.tex`. Příkladem užití může být zápis

```
37 \databaze Množiny/8b
```

uvozující databázi úloh „Množiny“ s bodovým ohodnocením 8 bodů. V případě vícejazyčné varianty lze psát např.

```
38  \databaze\cz{Množiny}\en{Sets}\de{Mengen}/8b
```

přitom na pořadí jazykových verzí nezáleží.

`\subdatabaze #1` [T]
Někdy je vhodné pro lepší orientaci rozdělit dílčí databázi úloh týkající se jediného vyučovacího celku na menší úseky. V takovém případě lze použít příkaz `\subdatabaze`, který do svého argumentu načítá její název. Koncovým separátorem argumentu je konec řádku. Bodové ohodnocení úloh libovolné subdatabáze je určeno v nejbližší předchozí části `\databaze` a není třeba jej specifikovat. Příkladem užití v jednojazyčné (české) verzi je zápis

```
40  \databaze Soustavy lineárních rovnic/5b
41    \subdatabaze Gaussova eliminace
42      <ulohy a reseni>
43    \subdatabaze Toky v sítích
44      <ulohy a reseni>
45    \subdatabaze Cramerovo pravidlo
46      <ulohy a reseni>
```

`\uloha #1***` [T]
`\reseni` [T]
Pro sazbu úloh v databázích je určen příkaz `\uloha`, jehož jediným argumentem je zadání úlohy a případné řešení. Konec každé úlohy musí být označen třemi hvězdičkami, které slouží jako koncový separátor argumentu. Při běžném užití se jeví tato možnost jak postačující, tj. nepředpokládáme užití tří po sobě jdoucích hvězdiček za sebou v textu úlohy, tak především snadná a přehledná.[5]

Příkladem sazby textu úlohy pro vícejazyčnou databázi při jazykové deklaraci na řádku 30 je třeba

```
49  \uloha
50    \cz{Text úlohy v českém jazyce.}%
51    \de{Aufgabentext in deutscher Sprache.}%
52    \en{Text of the problem in English.}%
53  ***
```

Je nutno si uvědomit význam uvedených znaků `%` na konci zápisu každé jazykové verze. Bez jejich uvedení by byly do textu úlohy zavlečeny nežádoucí mezery.

Jestliže z nějakého důvodu nechceme některou z úloh použít při pozdějším náhodném generování testu a ani ji zobrazit v souhrnném přehledu při kompilaci

[5] Makra byla vytvářena v adventním období. Čtenář se může zamyslet, zda tato skutečnost nemohla mít vliv na volbu koncového separátoru.

souboru `banka_uloh.tex`, je možno namísto smazání celé úlohy z databáze tuto pouze označit hvězdičkou bezprostředně za příkazem a psát

```
\uloha*
  Tady je zadání (ignorované) úlohy.
***
```

Snadné vyřazení vybraných úloh při pozdějším generování testu může být výhodné třeba po zveřejnění vzorového testu obsahujícího takto označené úlohy.

V případě, že k úloze chceme připojit řešení nebo návod k řešení, který můžeme využít při opravování testů, zapíšeme do těla úlohy separátor `\reseni`. Příkladem může být třeba

```
\uloha
  Tady je zadání úlohy.
\reseni
  A sem zapíšeme výsledek, návod nebo i celý postup řešení.
***
```

`\vyres #1` [P]

Přepínač `\vyres` slouží k ovládání zobrazení výsledků. Přípustné hodnoty pro `#1` jsou pouze hodnoty `0` a `1`.

Při nastavení na `\vyres 0`, resp. `\vyres 1`, vypneme, resp. zobrazíme, řešení úloh databáze ve výstupu kompilace `banka_uloh.tex`. Pokud přepínač v preambuli databáze neuvedeme, bude implicitně nastaveno `\vyres 1`. Pokud je zobrazení řešení povoleno, je vždy uvozeno k tomu určenou ikonou . Jinou ikonu nebo návěští je možno nastavit makrem `\ReseniIkona`, viz sekci 4.1.2.

6.4 Řešte danou diferenciální rovnici s počátečními podmínkami:

$$y'' + 9y = 27x^2 - 18, \qquad y(0) = 2, \quad y'(0) = 0.$$

$y(x) = \frac{14}{3} \cdot \cos(3x) + 3x^2 - \frac{8}{3}$.

Obrázek 2: Zobrazené řešení úlohy v databázi

3.1.3. Načítání dalších uživatelských nastavení

`nastaveni.tex`

Pro generování písemky pomocí třídy `ngt` je nutná úspěšná kompilace souboru `banka_uloh.tex`, při niž jsou zapsána podstatná data do souboru `databaze.pkd`. V případě, že je nutno načíst v preambuli další balíky, definice nebo nastavení, lze to provést explicitně v preambuli souboru `banka_uloh.tex`.

Vzhledem k tomu, že popisované nastavení bude potřeba i při generování písemky, doporučujeme spíše vytvořit v aktuálním adresáři soubor s názvem `nastaveni.tex` obsahující tato nastavení. Pokud bude takový soubor v adresáři existovat (s identickým názvem i příponou), třída `ngt` jej automaticky načte při obou volbách (viz Obrázek 3), aniž by bylo nutno to provést ručně. Pokud takový soubor v aktuálním adresáři nebude zjištěn, žádné doplňující nastavení se nenačte. Oproti přímému zápisu nastavení do preambule souboru `banka_uloh.tex` se takto zásadně zvýší její přehlednost, což je důležité. Další výhodou je, že nastavení pro dva soubory (databáze a písemka) můžeme kontrolovat na jediném místě. Praktickou možností využtí je ale i zápis definic maker obsahujících opakující se textové části úloh, např. v rámci jednotlivých databází.

3.2. Volba `pisemka`

3.2.1. Vygenerování písemného testu – základní přehled

Pro vygenerování testu nejprve vytvoříme samostatný soubor, např. `pisemka.tex`, který musí být ve stejném adresáři jako i dříve vytvořený soubor `banka_uloh.tex` a který načítá třídu `ngt` s volbou `pisemka`. Pro ovládání a generování výstupu jsou k dispozici příkazy uvedené v Tabulce 2.

`\generujTEST`	makro pro generování písemného testu
`\jazyk`	ovládání jazyku výstupu
`\vyres`	ovládání zobrazení výsledků
`\novy`	ovládání náhodnosti generování při další kompilaci
`\body`	ovládání zobrazení bodového zisku u každé úlohy
`\hlavicka`	makro pro sazbu uživatelem definovaného záhlaví testu
`\datum` `\Datum`	makro pro formátování, resp. tisk datumu

Tabulka 2: Příkazy dostupné při volbě `pisemka`

Základní struktura souboru `pisemka.tex` může vypadat takto:

pisemka.tex

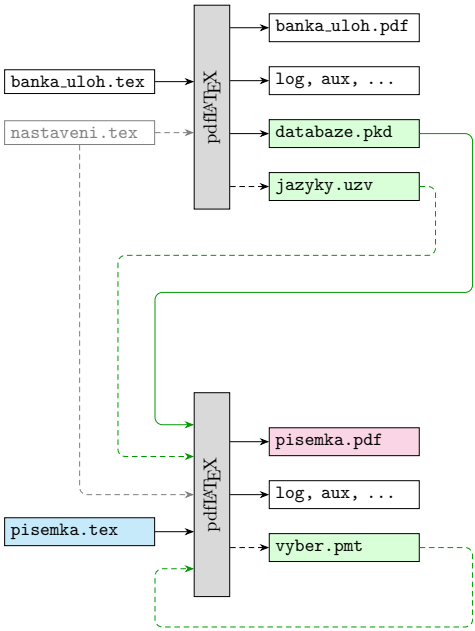
```
\documentclass[pisemka]{ngt}
  \jazyk <jzk_i>
  \vyres <0/1>
  \novy  <0/1>
  \body  <0/1>
  \datum <specifikace>
```

```
70    \hlavicka
71      <(vicejazycny) obsah sazeny v hlavicce>
72    ***
73
74  \begin{document}
75    \generujTEST
76  \end{document}
```

Pro úspěšnou kompilaci pomocí

```
77  pdflatex pisemka.tex
```

se předpokládá dříve provedená kompilace databáze `banka_uloh.tex`.



Obrázek 3: Kompilace souboru `pisemka.tex`

Jsou-li všechny požadované soubory k dispozici (viz kompilační schéma na Obrázku 3)[6], dostáváme spuštěním příkazu na řádku 77 soubory `pisemka.pdf` a `vyber.pmt`. Zatímco soubor PDF obsahuje vygenerovaný test z dříve vytvořených databází, soubor s příponou `pmt` obsahuje informace o posledním náhodném

[6]Čárkovaná propojení symbolizují načítání nebo zápis souborů pouze při jejich existenci, resp. při odpovídajícím uživatelském nastavení.

výběru úloh z jednotlivých databází. Tento soubor je možno chápat jako paměťový, který při vypnutém náhodném generování v `pisemka.tex` umožňuje načíst údaje o výběru úloh z minulé kompilace (podrobněji viz popis ovladače `\novy`).

3.2.2. Popis příkazů pro generování testu

`\generujTEST` [T]
Příkaz zajišťující generování testu z dříve vytvořených databází. Ve většině případů se jedná o jediný údaj zapsaný mezi `\begin{document}` a `\end{document}`.

`\jazyk <jzk_i>` [P]
Nastavením specifikace `\jazyk <jzk_i>` určujeme, v jakém jazyce bude vygenerován test, přitom zkratka `<jzk_i>` musí být jedna z platných zkratek jazyků uvedených v souboru `banka_uloh.tex` v deklaraci `\jazyky`. Pokud není uvedena žádná specifikace, bude test vygenerován v jazyce, který je uveden v deklaraci `\jazyky` na prvním místě. Zapsání deklarace `\jazyky` do souboru `pisemka.tex` je provedeno automaticky ve shodě s deklarací v soboru `banka_uloh.tex`. To je znázorněno na Obrázku 3 v podobě načtení souboru `jazyky.uzv`. Uvádění této deklarace v souboru `pisemka.tex` je tudíž nadbytečné a neočekává se.

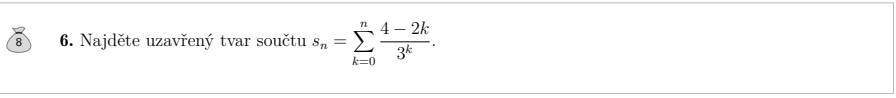
`\novy #1` [P]
Přepínač umožňující povolení, nebo naopak zablokování generování nové náhodné písemky při následující kompilaci. Povolené hodnoty za `#1` jsou buď `0`, nebo `1`.

Zamknutí náhodného výběru pomocí `\novy 0` může být užitečné tehdy, když chceme vygenerovat písemku se stejným výběrem úloh i při opakované kompilaci (třeba nejprve v českém a potom v anglickém jazyce, resp. nejprve písemku pro studenty bez řešení a potom tutéž písemku s řešeními nebo návody pro pedagoga). Není-li přepínač v preambuli uveden, je načteno implicitně `\novy 1`.

Je nutno poznamenat, že při zcela první kompilaci souboru pro generování písemky není rozumné mít nastaveno `\novy 0`, kdy nebyl paměťový soubor s příponou `pmt` ještě vytvořen. Nastane-li přece jen tento stav, je vyhodnocen jako chyba.

`\body #1` [P]
Přepínač pro ovládání zobrazení maximálního bodového zisku u každého návěští úlohy na levém okraji písemky (připomeňme, že zmíněné bodové zisky byly zadány u zahájení každé databáze v souboru `banka_uloh.tex`). Povolené hodnoty za `#1` jsou buď `0` nebo `1`.

Při `\body 0` nejsou bodová ohodnocení zobrazena, při `\body 1` naopak ano. Pokud není přepínač uveden, je implicitně nastaveno `\body 1`. Protože předpokládáme sazbu v libovolném jazyce podle deklarace `\jazyky`, bylo nutno zvolit místo jednotek bodového ohodnocení (např. body, b, points, pts) grafický prvek, který by evokoval patřičný dojem zisku, zde v podobě měšce. Pokud není čtenáři



Obrázek 4: Zobrazený bodový zisk u zadání úlohy písemného testu

popisovaná grafická úprava sympatická, může nastavit `\body 0` nebo předefinovat makro `\BodyIkona`, viz odstavec 4.3.4.

`\vyres` [P]
Funkce tohoto ovladače je shodná s funkcí stejně nazvaného ovladače u volby `databaze`.

`\hlavicka #1***` [P]
Ačkoliv název makra napovídá, že tento příkaz bude zajišťovat sazbu hlavičky písemky, začneme popisem situace, kdy jej v preambuli neuvedeme. V takovém případě bude vysázena implicitně nastavená hlavička obsahující místo pro jméno a osobní číslo studenta a v pravé části hlavičky bude vytištěno také datum.

Obrázek 5: Implicitně nastavená hlavička testu

Navíc při současně uvedeném `\vyres 1` (verze se zobrazenými výsledky pro pedagoga) bude implicitně nastavená hlavička zredukována na datum konání písemného testu a dolní dvojici linek. V takovém případě jsou totiž ikony pro jméno a osobní číslo studenta zbytné.

Pokud do preambule souboru `pisemka.tex` uvedeme makro `\hlavicka`, může pracovat dvěma způsoby, čemuž odpovídá i dvojí syntaxe příkazu. Píšeme-li

```
84 \hlavicka 0
```

nebude zobrazena žádná hlavička. Pokud píšeme

```
85 \hlavicka <nejaky_kod>***
```

kde `<nejaky_kod>` nezačíná znakem `0`, vytiskne se hlavička písemného testu podle zapsaného kódu `<nejaky_kod>`, přitom koncovým separátorem argumentu je již známá trojice hvězdiček.[7] Při této syntaxi je možno makro chápat jako kontejner, do nějž lze zapsat příkazy, které budou vykonány hned po `\begin{document}`

[7] Je malá pravděpodobnost, že hlavička písemky bude začínat tokenem `0`. Pokud taková výjimečná situace nastane, lze psát `\hlavicka\relax 0...***`.

a jazykových nastaveních dokumentu. Příkladem snadné hlavičky v jednojazyčné české verzi může být

```
\hlavicka
  Jméno:\hfill\Datum[8]
  \par\medskip
  \hrule\bigskip
***
```

Po načtení příslušných balíků v preambuli `pisemka.tex` lze však použít i daleko širších grafických možností (např. `pstricks`, `tcolorbox` nebo `tikz`).

`\datum #1` [P]
`\Datum` [P],[T]
Příkaz `\datum` slouží pro nastavení datumu konání písemky, který bude vytištěn pomocí příkazu `\Datum` (to je voláno i při sazbě implicitně nastavené hlavičky). Koncovým separátorem argumentu `#1` příkazu `\datum` je konec řádku.

Chování makra je určeno prvním znakem v načteném argumentu `#1`. Pokud je prvním znakem `+` nebo `-`, které je následováno celým číslem `<z>`, bude aktuální datum v době kompilace změněno o $+$`<z>` nebo $-$`<z>` dní. Pokud `<z>` není celé číslo, bude vygenerována chybová hláška. Nezačíná-li `#1` ani jedním ze znaků `+` nebo `-`, bude při použití `\Datum` (např. v uživatelem definované hlavičce) vytištěn obsah argumentu `#1`. Pokud v preambuli neuvedeme příkaz `\datum` s nějakou specifikací, vytiskne `\Datum` aktuální datum v době kompilace souboru. Příklady užití jsou třeba

```
\datum +2
\datum -1
\datum 10. ledna 2021
```

kde jednotlivé případy postupně nastaví datum na pozítří, včerejší datum, resp. fixní datum (à la DEK). Výhodou notace `\datum +/-<z>` je, že při změně hodnoty přepínače `\jazyk` bude datum vytištěný pomocí `\Datum` přeložen do příslušného jazyku.

4. Implementace

Tato část je věnována technické realizaci třídy `ngt`, jejíž báze je tvořena základní třídou v LaTeXu, `article`.

[8] Makro `\Datum` je popisováno v následujícím odstavci.

4.1. Preambule

4.1.1. Zavádění voleb třídy

Nejdříve standardním způsobem zavádíme obě volby třídy `ngt`, tj. `databaze` a `pisemka`. Přitom požadujeme, aby byla třída načtena vždy s právě jednou volbou. Proto zavádíme čítač `\PocetVoleb`, do nějž je uložen počet načtených voleb třídy `ngt`.

ngt.cls

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{ngt}[2020/02/29]
\LoadClass[12pt]{article}

\newcount\PocetVoleb
\newif\ifdatabaze

\DeclareOption{databaze}{%
  \databazetrue
  \advance\PocetVoleb by 1}

\DeclareOption{pisemka}{%
  \databazefalse
  \advance\PocetVoleb by 1}

\ProcessOptions
```

V případě, že není načtena žádná volba nebo načítáme obě volby současně, bude tento stav vyhodnocen jako chyba.

```
\ifnum\PocetVoleb=1
\else
  \ClassError{ngt}{%
    Prilis mnoho nactenych voleb nebo zadna volba}{}%
\fi
```

4.1.2. Společná nastavení

Třída `ngt` implicitně načítá balíky `lmodern`, `inputenc` a `fontenc`, přitom předpokládáme, že zdrojový soubor bude zapisován v kódování `utf8`. Pro nastavení délkových registrů geometrie stránky načítáme balík `geometry` s nastavením okrajů na 2 cm.

```
\RequirePackage{lmodern}
\RequirePackage[utf8]{inputenc}
\RequirePackage[T1]{fontenc}
\RequirePackage[margin=2cm]{geometry}
```

```
\def\bfdefault{b}
```

Dále zavedeme čítač `\PocetDatabazi`, s jehož pomocí evidujeme celkový počet uvedených databází. Při sazbě eventuálního řešení v databázi nebo písemce nastavujeme vertikální odsazení řešení od předchozího zadání jako `\bigskip`. Pro vyšší přehlednost kódu uvádíme také zkrácené verze primitivů `\expandafter` a `\noexpand`.

```
\newcount\PocetDatabazi
\parindent=0pt
\let\OdsadReseni\bigskip
\let\ea\expandafter
\let\nx\noexpand
```

Pro sazbu případného řešení nyní nadefinujeme jeho návěští, které bude obsahovat jistý grafický element (v případě vícejazyčného dokumentu je tato možnost vhodnější) a příslušné odsazení. Abychom se vyhnuli načítání externích obrázků nebo balíků, které generují grafiku, využíváme možností pdfTEXu disponujícího příkazem `\pdfliteral`, viz Olšák [2]. Analýzu následujícího kódu v definici příkazu `\ReseniIkona` přenecháváme za cvičení.

```
\def\ReseniIkona{%
  \pdfliteral{%
   q .7 g
   .9 0 0 .9 0 0 cm
   0 0 4 4 re f
   1 0 0 1 5 0 cm
   0 0 4 4 re f
   1 0 0 1 0 5 cm
   0 0 4 4 re f
   .7071 .7071 -.7071 .7071 -5 1 cm
   1 0 0 rg
   0 0 4 4 re f Q}}
```

Za účelem velmi častého vykreslování ikony řešení v PDF především při volbě `databaze` (velký počet úloh, a tedy i řešení) je výhodnější vykreslit ikonu pouze jednou a později se na ni odkazovat pomocí `\pdfrefxform`.

```
\setbox 0\hbox to 1.8em{%
  \hskip3pt\vbox to 1em{\vss\ReseniIkona}\hss}
\pdfxform 0
\edef\PDFreseni{\the\pdflastxform}

\def\NavestiReseni{%
  \leavevmode
  \pdfrefxform\PDFreseni}
```

4.1.3. Pomocná makra

V dalším budeme několikrát potřebovat zapisovat a číst jistá data. Pro zjednodušení definujeme makra `\otevri`, `\zapis` a `\zavri`, která zápis kódu příslušejícího těmto úkonům zkrátí.

```
\def\otevri{%
  \immediate\openout }
\def\zapis{%
  \immediate\write }
\def\zavri{%
  \immediate\closeout }
```

Pro pohodlnější čtení argumentů obsahujících lomítko (např. jazyková deklarace `\jazyk`), bude užitečné umět uložit si část před a za lomítkem do odpovídajících maker.

```
\def\PredLomitkem#1/#2,{#1}
\def\ZaLomitkem#1/#2,{#2}
```

Několikrát bude potřeba v zadaném seznamu tokenů spočítat počet výskytů konkrétního tokenu (postačí nám vnější tokeny, tedy neuvedené uvnitř skupiny). Definujeme proto makro `\PocetVyskytu`, které má dva povinné parametry. Prvním parametrem je token, jehož počet výskytů chceme spočítat. Druhým parametrem je čítač, v němž tento počet bude uložen. Za touto specifikací následuje předložený seznam tokenů (zpravidla text), v němž počet výskytů zjišťujeme. Je-li načten token `\UkonciPocitadlo`, je činnost makra `\PocetVyskytu` ukončena.

```
\def\UkonciPocitadlo{}
\def\UkonciPocitadloToken{\UkonciPocitadlo}
\newcount\pocet

\def\PocetVyskytu[#1,#2]{%
  \def\TestovanyZnak{#1}%
  \pocet=0
  \newcount#2%
  \let\pocitadlo#2%
  \let\next\nacti\next}

\long\def\nacti#1{%
  \def\znak{#1}%
  \ifx\znak\UkonciPocitadloToken
    \def\next{\pocitadlo=\pocet}%
  \else
    \ifx\znak\TestovanyZnak
      \advance\pocet by 1
    \fi
```

```
79    \fi
80    \next}
```

4.1.4. Jazyková nastavení

V této části se budeme věnovat popisu maker, která umožňují jazyková nastavení, tj. maker `\jazyky`, `\jazyk` a pomocných maker. V souvislosti s těmito makry zavádíme čítač `\PocetJazyku`, v němž bude uložen celkový počet jazyků uvedený v deklaraci `\jazyky`. Čítač `\PoradiJazyku` má pouze pomocný charakter.

```
81  \newcount\PocetJazyku
82  \newcount\PoradiJazyku
```

Jazyková nastavení dokumentu budou určena tím, zda uživatel uvede jazykovou deklaraci `\jazyky`, resp. specifikaci `\jazyk`, a co do nich zapíše. Pro detekci uvedení těchto nastavení zavádíme logické proměnné `\ifjazyky` a `\ifjazyk`, které nastavujeme na `false`.

```
83  \newif\ifjazyky
84  \newif\ifjazyk
85    \jazykyfalse
86    \jazykfalse
```

Aby nebylo nutno opakovaně zapisovat shodnou deklaraci `\jazyky` u databáze i písemky, zapíšeme toto uživatelské nastavení uvedené v preambuli databáze do pomocného souboru `jazyky.uzv`.

```
87  \newwrite\nastaveniJazyky
```

Nyní definujeme makro `\jazyky`, které nastaví `\jazykytrue` a v případě, že aktuální volba třídy je `databaze`, je uvedené nastavení zapsáno do pomocného souboru. Ten později při volbě `pisemka` načteme, viz řádek 312.

```
88  \def\jazyky#1 {%
89    \jazykytrue
90    \ifdatabaze
91      \otevri\nastaveniJazyky jazyky.uzv
92      \zapis\nastaveniJazyky{\nx\jazyky #1}%
93      \zavri\nastaveniJazyky
94    \fi
```

Dále spočítáme, jaký počet lomítek uživatel zapsal za `\jazyky`. Pokud je počet lomítek nulový, ukládáme uvedený argument jako `\JedinyJazyk` a nastavujeme počet jazyků na 1.

```
95    \PocetVyskytu[/,\PocetLomitek]#1\UkonciPocitadlo
96    \ifnum\PocetLomitek=0
97      \def\JedinyJazyk{#1}%
98      \PocetJazyku=1
```

Pokud je v argumentu uvedeno alespoň jedno lomítko, bude s argumentem `#1` doplněným o tokeny `,/,` vykonán příkaz `\xjazyky` (viz řádek 107). Pomocí příkazu `\xseznam` (viz řádek 118) se vytvoří seznam zadaných jazyků z částí argumentu `#1` před lomítky a uloží se do `\SeznamJazyku`. Takto vytvořený seznam bude později použit při načtení jazyků v balíku `babel`.

```
99    \else
100     \xjazyky#1,/,
101     \def\SeznamJazyku{\xseznam#1,/,}
```

Pokud je počet uvedených jazyků v deklaraci `\jazyky` roven 1 (s právě jedním uvedeným lomítkem), ukládáme uvedenou jazykovou zkratku za lomítkem jako `\JedinyJzk` a nastavujeme jazykovou specializaci pomocí makra `\jazyk` automaticky na uvedený jazyk.

```
102     \ifnum\PocetJazyku=1
103       \def\JedinyJzk{\ZaLomitkem #1, }
104       \ea\jazyk\JedinyJzk
105     \fi
106   \fi}
```

Mechanismus makra `\xjazyky` zmíněného dříve spočívá v postupném načítání položek oddělených čárkou a uvedených za tímto makrem. V každém cyklu makra se do argumentu načte text až po nejbližší čárku a dále s ním pracuje. Pokud je načteným argumentem smluvně stanovená značka (zde lomítko `/`), je činnost makra ukončena uložením počtu uvedených jazyků do `\PocetJazyku`.

```
107 \def\xjazyky#1,{%
108   \ifx/#1%
109     \PocetJazyku=\the\PoradiJazyku
```

Není-li argumentem lomítko, považujeme jej za jednu z jazykových deklarací, navýšíme hodnotu čítače `\PoradiJazyku` o 1 a vykonáme s aktuálním argumentem příkaz `\PridejJazyk` (viz řádek 124). Speciálně při prvním cyklu makra (`\PoradiJazyku=1`) ukládáme jazykovou zkratku prvního jazyku jako `\PrvniJzk`.

```
110   \else
111     \PridejJazyk#1/%
112     \advance\PoradiJazyku by 1
113     \ifnum\PoradiJazyku=1
114       \def\PrvniJzk{\ZaLomitkem #1, }
115     \fi
116     \ea\xjazyky
117   \fi}
```

Dříve použité makro `\xseznam` pracuje se seznamem jazykových deklarací uvedených za `\jazyky` a načítá do svého argumentu položku až do nejbližší čárky. Z této dílčí deklarace odstraní lomítko a část za ním a připojí čárku. Funkce

makra je ukončena, pokud je načteným argumentem smluvní lomítko, což se stane až na konci seznamu, kde je za tímto účelem přidáno `,/,` (viz řádek 101).

```
\def\xseznam#1,{%
  \ifx/#1%
  \else
    \PredLomitkem #1,,%
    \ea\xseznam
  \fi}
```

Makro `\PridejJazyk` uvedené na řádku 111 má dvě funkce. První funkcí je přiřazení názvu jazyku jeho zkratce (tj. `#2` $\mapsto$ `#1`). Vzhledem k tomu, že uživatel bude zadávat jazykovou verzi pomocí zkratky jazyku, je tento přístup výhodný. Druhou funkcí je definice makra jazykové verze, jehož název je identický s uváděnou zkratkou jazyku. Ve fázi uvedení deklarace `\jazyky` pracují všechna tato makra naprázdno, tj. nic se nevytiskne. Ve finálním dokumentu budeme totiž požadovat zobrazení právě jedné jazykové verze, která bude určena až specifikací `\jazyk`. Makro pro příslušnou jazykovou verzi pak postačí předefinovat.

```
\def\PridejJazyk#1/#2/{%
  \ea\def\csname jazyk:#2\endcsname{#1}%
  \ea\def\csname #2\endcsname##1{}}
```

Následuje definice makra `\jazyk`, které umožní uživateli specifikovat aktuální zobrazený jazyk. Uvedením této specifikace nastavujeme `\jazyktrue` a také definujeme `\AktivniJzk` jako zkratku jazyku načtenou do argumentu. Toto uložení využíváme na řádku 169 při načítání odpovídajících vzorů dělení slov.

```
\def\jazyk#1 {%
  \jazyktrue
  \def\AktivniJzk{#1}%
```

Pokud odpovídá uvedená zkratka některému jazyku uvedenému v `\jazyky`, předefinujeme příslušné makro pro zápis jazykové verze tak, aby se vysázel jeho argument. Tím je zaručeno, že se zobrazí pouze aktuálně zvolená jazyková verze. Ostatní makra jazykových verzí totiž stále pracují naprázdno.

```
  \ifcsname jazyk:#1\endcsname
    \ea\def\csname #1\endcsname##1{##1}%
```

Pokud uživatelem uvedený jazykový identifikátor nekoresponduje se žádným jazykem uvedeným v `\jazyky`, třída ohlásí při kompilaci chybu.

```
  \else
    \ifjazyky
      \ClassError{ngt}{Neznama jazykova specifikace ‘#1’}{}%
    \fi
  \fi}
```

4.1.5. Předefinování \begin{document}

V následujícím zapíšeme před, resp. za \begin{document}, potřebné nastavení. Za tímto účelem předefinujeme původní příkaz \document definovaný v latex.ltx. Ekvivalent původního \document si pracovně označíme pro pozdější použití jako \origdocument.

```
\let\origdocument\document
\def\document{%
  \endgroup
```

Pokud uživatel použil deklaraci \jazyky s více než jedním jazykem, ale neuvedl specifikaci \jazyk, třída ngt vygeneruje informativní upozornění o automatickém načtení prvního jazyku v deklaraci. Příslušné načtení jazykových vzorů provádíme na řádku 174, tj. za \origdocument.

```
  \ifjazyky
    \ifjazyk
    \else
      \ifnum\PocetJazyku>1
        \ea\jazyk\PrvniJzk
        \ClassWarning{ngt}{%
          Nastaveno \string\jazyk\space `\PrvniJzk'}%
      \fi
    \fi
  \fi
```

Dále načítáme balík babel v závislosti na tom, v jakém formátu byla zadána jazyková deklarace v \jazyky.

```
  \ifjazyky
    \ifnum\PocetLomitek=0
      \usepackage[\JedinyJazyk]{babel}%
    \else
      \usepackage[\SeznamJazyku]{babel}%
    \fi
```

V případě, že uvádíme specifikaci \jazyk bez uvedení deklarace \jazyky, je ohlášena chyba kompilace.

```
  \else
    \ifjazyk
      \ClassError{ngt}{Doplnte deklaraci \string\jazyky}{}%
```

Pokud neuvedeme ani specifikaci \jazyk, je načtena čeština.

```
    \else
      \usepackage[czech]{babel}
    \fi
  \fi
```

V části za `\begin{document}` potom načítáme odpovídající vzory pro dělení slov. Pro celkový počet jazyků roven 1 nemá význam uvádět `\selectlanguage`.

```
\begingroup\origdocument
\ifjazyky
  \ifjazyk
    \ifnum\PocetJazyku=1
    \else
      \ea\ea\ea\selectlanguage
      \ea\ea\ea{\csname jazyk:\AktivniJzk\endcsname}
    \fi
  \else
    \ifnum\PocetJazyku>1
      \ea\ea\ea\selectlanguage
      \ea\ea\ea{\csname jazyk:\PrvniJzk\endcsname}
    \fi
  \fi
\fi}
```

4.1.6. Interní `0/1` přepínače

Kromě jazykového nastavení bude potřeba při volbě `databaze` a především `pisemka` nastavit další parametry sazby. Ty pro přehlednost omezíme na formát, kdy za klíčovou sekvenci uvádíme hodnoty `0` (ne) nebo `1` (ano). Definujeme proto zástupné sekvence `\etiam` a `\nihil`[9] pomocí

```
\def\etiam{1}
\def\nihil{0}
```

Makro `\OIprepinac` definuje popsaný druh ovladače. Do svých dvou povinných argumentů načítá postupně název přepínače a iniciální předvolenou hodnotu.

```
\def\OIprepinac#1/#2 {%
  \def\init{#2}%
```

Po uložení iniciální hodnoty do makra `\init` zavádíme novou logickou proměnnou obsahující název přepínače. V závislosti na hodnotě uložené v `\init` nastavujeme tuto hodnotu na `false`, resp. `true`. Pokud hodnota v `\init` není `0` ani `1`, bude ohášena chyba.[10]

```
  \ea\newif\csname if#1\endcsname
  \ifx\init\etiam
    \csname #1true\endcsname
```

[9] Bohužel nelze použít `\ano` a `\ne`, protože druhá by kolidovala s příkazem pro sazbu $\neq$. Volíme tedy neutrální latinské termíny.

[10] Tuto chybu nemůže uživatel bez editace souboru `ngt.cls` způsobit. Jedná se spíše o pojistku pro autora balíku v případě zavlečení překlepu.

```
    \else
      \ifx\init\nihil
        \csname #1false\endcsname
      \else
        \ClassError{ngt}{%
          Neocekavana inicialni hodnota ‘#2’ v ‘#1’}{}%
      \fi
    \fi
```

Současně připravíme koncovému uživateli makro, jehož název bude identický s názvem přepínače a bude načítat do argumentu jediný token (`0` nebo `1`), který si uložíme do `\OIhodnota`. Ve shodě s touto hodnotou nastavujeme hodnotu příslušné logické proměnné, kterou může uživatel ovlivnit zobrazení výstupu v PDF. V případě, že uložená hodnota není `0` ani `1`, bude ohlášena chyba.

```
    \ea\def\csname #1\endcsname##1 {%
      \def\OIhodnota{##1}%
      \ifx\OIhodnota\etiam
        \csname #1true\endcsname
      \else
        \ifx\OIhodnota\nihil
          \csname #1false\endcsname
        \else
          \ClassError{ngt}{Neocekavana hodnota (#1)}{}%
        \fi
      \fi}}
```

V případě, že uživatel neuvede žádné nastavení, bude implicitně načteno

```
\OIprepinac vyres/1
\OIprepinac novy/1
\OIprepinac body/1
```

Tímto nastavením jsme současně definovali přepínače `\vyres`, `\novy` a `\body`.

4.1.7. Jiná pomocná makra a nastvení

Pro zvýšení přehlednosti kódu volíme v některých případech definice maker, jejichž koncovým separátorem je konec řádku. Tato sympatická praxe má své zastoupení třeba v systému OPmac P. Olšáka popsané v [2, s. 46]. Jeho makro `\eoldef` jsme pro naše účely převzali v nezměněné podobě.

```
\def\eoldef#1{%
  \def#1{\begingroup \catcode‘\^^M=12 \eoldefA#1}%
   \ea\def\csname\string#1:M\endcsname}
{\catcode‘\^^M=12 \gdef\eoldefA#1#2^^M{%
  \endgroup\csname\string#1:M\endcsname{#2}}}
```

Často je vhodné vědět, jaký je první token, který následuje za řídící sekvencí. Pro účely tohoto specifického rozdělení načítaného argumentu nějakého makra definujeme `\RozdelArgument`, které rozloží původní argument a uloží si první token, resp. zbylé tokeny, do odpovídajících maker.

```
\long\def\RozdelArgument#1#2***{%
  \def\PrvniToken{#1}%
  \long\def\ZbyleTokeny{#2}}
```

Posledním přípravným krokem bude případné načtení souboru `nastaveni.tex` s uživatelským nastavením (v případě, že existuje). Za tímto účelem přebíráme ještě jedno makro P. Olšáka (viz [3, s. 288]), `\softinput`.

```
\newread\testin

\def\softinput #1 {%
  \let\next\relax
  \openin\testin=#1
  \ifeof\testin
  \else
    \closein\testin
    \def\next{\input #1 }%
  \fi
  \next}

\softinput nastaveni.tex
```

Pro zajištění vertikálního odsazení mezi jednotlivými úlohami kromě prvních úloh v aktuální databázi nebo subdatabázi (tj. oddílu), definujeme logickou proměnnou `\ifoddil`.

```
\newif\ifoddil
```

Tu nastavíme na `true` na začátku sazby každé databáze a subdatabáze (viz řádky 251 a 261).

4.2. Nastavení pro volbu `databaze`

```
\ifdatabaze
```

4.2.1. Všeobecná nastavení

V této části jsou nastaveny základní parametry sazby a některé pomocné prostředky. Především zavádíme vertikální odsazení úlohy jako `\bigskip` a také související čítače pro číslování úloh a dílčích databází (sekcí).

```
  \def\OdsadUlohu{\vskip1.5em}
  \newcount\PoradiUlohy
```

```
\newcount\sekce
```

Pro zápis zadání (a pokud existují, i řešení) úloh deklarujeme identifikátor souboru pro zápis `\zadaniSoubor`. V souvislosti s tímto zápisem deklarujeme registry `\zadaniTokeny`, resp. `\reseniTokeny`, do nichž budeme ukládat jednotlivé definice příkazů pro zadání, resp. řešení úloh. Soubor, do něhož bude zápis proveden, nazveme `databaze.pkd`, kde `pkd` značí překódovaný. Cílem je totiž jednoznačně přiřadit každé položce databáze (úloze, případně řešení) uspořádanou trojici $(\texttt{X}, m, n)$, tj. zakódovat ji. Zde `X` označuje typ položky (`U` – úloha, `R` – řešení), číslo m pořadí databáze a n pořadí úlohy v rámci m-té databáze. Každou úlohu, nebo případně její řešení, si tak po zápisu těchto definic budeme moci zavolat pouze zadáním typu objektu `X` a čísel m, n.

```
\newwrite\zadaniSoubor
  \newtoks\zadaniTokeny
  \newtoks\reseniTokeny
  \otevri\zadaniSoubor databaze.pkd
```

Aby nebyl zápis definic těžkopádný, připravíme si makro `\ZapisDefinici` se třemi parametry, identifikátorem souboru pro zápis, názvem makra a ukládaným obsahem.

```
\def\ZapisDefinici#1#2#3{%
  \zapis#1{\nx\ea\def\nx\csname#2\endcsname{#3}}}
```

Pro sazbu návěští úlohy nadefinujeme malý čtvereček, za nímž následuje pořadí databáze a pořadí úlohy v rámci databáze.

```
\def\ctverecek{%
  \pdfliteral{%
   q .7 g 0 0 6 6 re f Q}}

\setbox 0\hbox to 1em{\vbox to 1em{\vss\ctverecek}\hss}
\pdfxform 0
\edef\PDFctverecek{\the\pdflastxform}
\def\NavestiUlohy{%
  \leavevmode
  \llap{\pdfrefxform\PDFctverecek\hskip5pt}{%
    \bfseries\the\sekce.\the\PoradiUlohy}~~}

```

Následuje definice makra `\databaze`, které má dva povinné parametry oddělené lomítkem, koncovým separátorem je písmeno `b`. Pokud je hodnota čítače `\sekce` větší než nula, zapíšeme do souboru `databaze.pkd` počet úloh předchozí sekce jako obsah makra `\csname PocetUloh|<poradi_sekce>|\endcsname`. Počet úloh konkrétní dílčí databáze (tj. pořadí její poslední úlohy) tak provádíme hned na začátku zavolání nejbližšího následujícího `\databaze`, kdy hodnota čítače `\sekce` ještě není změněna.

```
\def\databaze#1/#2b{%
  \ifnum\sekce>0
    \ZapisDefinici\zadaniSoubor{%
      PocetUloh|\the\sekce|}{\the\PoradiUlohy}%
  \fi
```

Na začátku databáze vynulujeme čítač `\PoradiUlohy`, nastavíme `\oddiltrue` a sázíme `\section` známé z LaTeXu. Současně pokládáme hodnotu čítače `\sekce` rovnu hodnotě čítače `\thesection` číslující v LaTeXu sekce.

```
  \PoradiUlohy=0
  \section{#1}
  \oddiltrue
  \sekce=\arabic{section}%
```

Do souboru `databaze.pkd` si ještě uložíme počet bodů příslušející úlohám dané dílčí databáze (makro `\csname Body|<poradi_sekce>|\endcsname`).

```
  \ZapisDefinici\zadaniSoubor{Body|\the\sekce|}{#2}}
```

Pokud bude potřeba vytvořit jemnější strukturu dílčí databáze, definujeme pro tento účel makro `\subdatabaze`, které je klasickým `\subsection` z LaTeXu. Pokud je bezprostředně před zapsaným `\subdatabaze` uvedeno `\databaze`, je nastaveno ještě před sazbou subdatabáze `\oddiltrue`. V tomto případě nadpis subdatabáze od nadpisu databáze neodsazujeme. Při `\oddilfalse` naopak odsadíme nadpis subdatabáze (od předchozí úlohy).

```
\eoldef\subdatabaze#1{%
  \ifoddil\else\OdsadUlohu\fi
```

Po vysázení nadpisu subdatabáze nastavíme `\oddiltrue`.

```
  \subsection{#1}
  \oddiltrue}
```

4.2.2. Definice maker pro sazbu úloh a řešení

Při zápisu úlohy s řešením budeme zadání od řešení oddělovat separátorem `\reseni`. Při pozdějším zobrazení verze s výsledky nebo bez nich (ovladač `\vyres`) bude vhodné uložit si zvlášť část před, resp. za separátorem, jako `\xzadani`, resp. `\xreseni`.

```
\long\def\RozdelZadaniReseni#1\reseni#2***{%
  \long\def\xzadani{#1}%
  \long\def\xreseni{#2}}
```

Pro účely ignorování úlohy při sazbě databáze a při generování testu definujeme příkaz `\hvezda` jako token `*`.

```
\def\hvezda{*}
```

Nyní následuje definice důležitého makra `\uloha`. Nejdříve si rozdělíme argument (tj. zadání úlohy s případným řešením) na první token a zbylé tokeny a také v něm spočítáme počet výskytů tokenu `\reseni`, který ukládáme do čítače `\PocetReseni`.

```
\long\def\uloha#1***{%
  \RozdelArgument#1***
  \PocetVyskytu[\reseni,\PocetReseni]#1\UkonciPocitadlo
```

Pokud je první token `*`, činnost makra je ukončena a nic se nevysází. V opačném případě navýšíme čítač `\PoradiUlohy` o 1.

```
  \ifx\PrvniToken\hvezda
  \else
    \advance\PoradiUlohy by 1%
```

Současně vertikálně odsadíme každou úlohu, která nenásleduje bezprostředně po vysázení nadpisu databáze nebo subdatabáze. Pokud úloha následuje bezprostředně po uvedeném typu nadpisu, nastavujeme `\oddilfalse`, aby bylo zajištěno odsazení dalších úloh dané databáze nebo subdatabáze.

```
    \ifoddil
      \oddilfalse
    \else
      \OdsadUlohu
    \fi
```

Dále se činnost makra větví podle toho, kolikrát napsal uživatel do argumentu `#1` separátor `\reseni`. Pokud není uveden ani jeden separátor `\reseni`, uložíme celý argument do registru `\zadaniTokeny`, vysázíme návěští úlohy se zadáním, uložíme text zadání do makra `\csname U|<m>|<n>|\endcsname` a zapíšeme tuto definici do souboru `databaze.pkd`.

```
    \ifcase\PocetReseni
      \zadaniTokeny={#1}
      \NavestiUlohy #1\par
      \ZapisDefinici\zadaniSoubor{%
        U|\the\sekce|\the\PoradiUlohy|}{\the\zadaniTokeny}
```

Pokud je uveden právě jeden separátor `\reseni`, rozdělíme načtený argument na část `\xzadani`, resp. `\xreseni` a uložíme je do registrů `\zadaniTokeny`, resp. `\reseniTokeny`. Opět provádíme analogický kódovací zápis definic do souboru `databaze.pkd`.

```
    \or
      \RozdelZadaniReseni#1***
      \ea\zadaniTokeny\ea=\ea{\xzadani}
      \ea\reseniTokeny\ea=\ea{\xreseni}
      \ZapisDefinici\zadaniSoubor{%
```

```
290             U|\the\sekce|\the\PoradiUlohy|}{\the\zadaniTokeny}
291           \ZapisDefinici\zadaniSoubor{%
292             R|\the\sekce|\the\PoradiUlohy|}{\the\reseniTokeny}
```

Dále již sázíme návěští úlohy, a pokud uživatel nastavil ovladač `\vyres` na hodnotu `1` (tj. `\resenitrue`), sázíme také její řešení s příslušným odsazením a návěštím.

```
293           \NavestiUlohy\xzadani\par
294           \ifvyres
295             \OdsadReseni
296             {\footnotesize\NavestiReseni\xreseni}\par
297           \fi
```

Pokud uživatel zadal více než jeden separátor `\reseni`, bude tento stav vyhodnocen jako chyba.

```
298         \else
299           \ClassError{ngt}{%
300             V prostredi \string\uloha\space muzete pouzit
301             \string\reseni\space nejvyse jednou}{}%
302         \fi
303       \fi}
```

Zcela závěrem ještě zapíšeme počet úloh poslední uvedené databáze, uložíme počet databází a ukončíme zápis do souboru `databaze.pkd`.

```
304     \AtEndDocument{%
305       \ZapisDefinici\zadaniSoubor{%
306         PocetUloh|\the\sekce|}{\the\PoradiUlohy}
307       \zapis\zadaniSoubor{\PocetDatabazi=\the\sekce}
308       \zavri\zadaniSoubor}
```

4.3. Nastavení pro volbu `pisemka`

```
309   \else
```

4.3.1. Všeobecná nastavení

Kromě úpravy horního okraje pro hlavičku načítáme překódovanou databázi úloh a jazykové nastavení uvedené v řídicím souboru, což je v naší ukázce soubor `banka_uloh.tex` (pokud bylo nějaké uvedeno). Definujeme také odpovídající vertikální odsazení úlohy na písemce.

```
310     \advance\topmargin by -20pt
311     \input databaze.pkd
312     \softinput jazyky.uzv
```

```
313
314     \def\OdsadUlohu{%
315       \bigskip\bigskip}
```

4.3.2. Nastavení sazby datumu

Pro usnadnění zadávání datumu bez nutnosti jeho ručního stanovení definujeme několik pomocných maker. Nejdříve definujeme makro `\DnuVMesici`, které expanduje na počet dnů v daném měsíci.[11]

```
316     \def\DnuVMesici{%
317       \ifcase\month
318       \or 31
319       \or\ifnum\numexpr\year/4*4-\year=0 29 \else 28 \fi
320       \or 31 \or 30 \or 31 \or 30
321       \or 31 \or 31 \or 30 \or 31
322       \or 30 \or 31 \fi}
```

Nyní definujeme makro `\PrictiDny`, které zajistí úpravu aktuálního datumu vpřed nebo vzad podle hodnoty v argumentu `#1`.

```
323     \def\PrictiDny#1{%
324       \advance\day#1\relax
325       \ifnum#1>0
326         \OpravDatumVpred
327       \else
328         \OpravDatumVzad
329       \fi}
```

Makra `\OpravDatumVpred` a `\OpravDatumVzad` pracují obě na shodné bázi jako cykly. Vzhledem ke skutečnosti, že makro `\PrictiDny` upravuje hodnotu čítače `\day`, je nutno v závislosti na přičítaném počtu dnů a na počtu dnů v jednotlivých měsících zjistit, jaký bude po úpravě měsíc (čítač `\month`) a rok (čítač `\year`).

```
330     \def\OpravDatumVpred{%
331       \ifnum\day>\DnuVMesici
332         \advance\day by -\DnuVMesici
333         \advance\month by 1
334         \ifnum\month>12
335           \month=1
336           \advance\year by 1
337         \fi
338       \ea\OpravDatumVpred
339       \fi}
```

[11] Uživatelé třídy `ngt` v roce 2100 musí počítat s malým překvapením.

```
340
341     \def\OpravDatumVzad{%
342       \ifnum\day<1
343         \advance\month by -1
344         \ifnum\month<1
345           \month=12
346           \advance\year by -1
347         \fi
348         \advance\day\DnuVMesici
349         \ea\OpravDatumVzad
350       \fi}
```

Pro sazbu datumu připravíme uživateli makro `\Datum`, které v tuto chvíli definujeme jako `\today`.

```
351     \def\Datum{\today}
```

Pokud bude chtít uživatel upravit datum (tj. `\Datum`) na jiné, provede potřebné nastavení pomocí makra `\datum`, které načítá argument do konce řádku. Další činnost makra `\datum` spočívá ve specifickém předefinování `\Datum` s větvením podle prvního tokenu v zadaném argumentu.

```
352     \eoldef\datum#1{%
353       \RozdelArgument#1***
354       \gdef\Datum{%
```

Pokud je prvním tokenem `+`, provede se úprava datumu přičtením dnů podle hodnoty v `\ZbyleTokeny`. Pokud je prvním tokenem `-`, provede se analogická úprava odečtením.

```
355         \if\PrvniToken+
356           {\PrictiDny{\ZbyleTokeny}\today}%
357         \else
358           \if\PrvniToken-
359             {\PrictiDny{-\ZbyleTokeny}\today}%
```

Pokud není první token ani `+` ani `-`, vysází se původní argument.

```
360           \else#1%
361           \fi
362         \fi}}
```

4.3.3. Hlavička písemky

Vzhledem k okolnostem plynoucím z předpokládané vícejazyčné sazby dokumentu je vhodné textové popisky v hlavičce redukovat na minimum. Namísto jména a příjmení budeme sázet ikonu osoby příkazem `\JmenoIkona`. Analýzu kódu v `\pdfliteral` přenecháváme opět za cvičení.

```
\def\JmenoIkona{%
  \pdfliteral{%
   q 1 G .5 g 2 w 0 0 m
    0  5  5 10 10 10 c
   15 10 20  5 20  0 c h B
   .3 G .4 w 30 0 m
   200 0 l s
   .7 G .7 g 10 10 m
    4 10    5 17.5 10 17.5 c 10 17.5 m
   15 17.5 16 10   10 10   c h f Q}}
```

Protože jméno a příjmení nejsou vždy jednoznačným identifikátorem studenta, je často nutné požadovat jeho osobní číslo nebo podobný údaj. Návěští pro tento údaj bude zastoupeno specifickou ikonou sázenou příkazem `\OsobniCisloIkona`.

```
\def\OsobniCisloIkona{%
  \pdfliteral{%
   q .5 G .5 g 2 w
   2.5 0 m  5   10 l s
   6   0 m  8.5 10 l s
    .7 3 m 10    3 l s
   1.5 7 m 10.7  7 l s Q}}
```

Sazba hlavičky pomocí `\Hlavicka` je ovlivněna nastavením ovladače pro sazbu výsledků (`\vyres`). Při `\vyres 1`, tj. verze zpravidla pro pedagoga, není nutno sázet identifikační ikony. Ostatní prováděné úkony sazby hlavičky jsou zřejmé.

```
\def\Hlavicka{%
  \ifvyres
  \else
    \hrule height 1.5pt
    \par\vskip30pt
    \leavevmode\JmenoIkona
    \par\medskip
    \leavevmode\OsobniCisloIkona\hskip10pt\OsobniCisloIkona
  \fi
  \hfill
  \Datum\medskip\par
  \hrule\par\vskip 2pt
  \hrule height 1.5pt\bigskip}%
```

Příkaz `\hlavicka` pro úpravu hlavičky dokumentu pracuje v závislosti na prvním tokenu. Pokud je jím token `0`, chová se `\hlavicka` jako `\relax`. V opačném případě se chová jako `\xhlavicka #1`, který pouze předefinuje `\Hlavicka`.

```
\def\hlavicka#1{%
  \ifx#10
```

```
395         \let\Hlavicka\relax
396       \else
397         \xhlavicka#1%
398       \fi}
399
400     \long\def\xhlavicka#1***{%
401       \def\Hlavicka{{#1}}%
```

4.3.4. Zobrazení bodových zisků

Bodové zisky uložené v souboru `databaze.pkd` jsou obecně rozdílné pro každou úlohu. Zavedeme proto pomocný čítač `\PoradiDatabaze`, pomocí nějž budeme indexovat bodové zisky v závislosti na pořadí aktuální databáze.

```
402     \newcount\PoradiDatabaze
```

Nejdříve vytvoříme box `\MujBox`, do nějž vložíme bodový zisk. Dále definujeme makro `\BodyBox`, které změří šířku boxu obsahujícího bodový zisk (registr `\sirka`) a horizontálně jej vycentruje na aktuální pozici sazby.

```
403     \newdimen\sirka
404
405     \def\BodyBox{%
406       \hbox to 0pt{%
407         \hss\scriptsize\sffamily
408         \csname Body|\the\PoradiDatabaze|\endcsname\hss}}
```

Protože předpokládáme možnost sazby testu v libovolném jazyce, vyhýbáme se opět textovým popiskům a využíváme vlastní ikony. Pro účely zobrazení bodů definujeme makro `\BodyIkona`. To vykreslí jednoduchý měšec, do kterého je umístěn bodový zisk. Nakonec bodový zisk s ikonou vytiskneme vlevo od aktuálního bodu sazby.

```
409     \def\BodyIkona{%
410       \llap{%
411         \pdfliteral{%
412          q .8 0 0 .8 0 0 cm
413         .85 g .5 G .7 w -10 0 m
414         -10 5 -3 12 0 12 c
415         3 12 10 5 10 0 c
416         10 0 10 -5 0 -5 c
417         -10 -5 -10 0 -10 0 c B
418         -2 11 m -5 15 l
419         0 13 0 18 5 15 c
420         2 11 l B
421         .2 G 1.5 w -2.5 11 m
```

```
422         0 11.5 0 11.5 2.5 11 c B Q}%
423         \BodyBox\hskip 2.5em}}
```

4.3.5. Návěští úlohy a některé pomocné prostředky

Návěští úlohy v písemném testu definujeme jako tučně vysázené pořadové číslo úlohy.

```
424     \def\NavestiUlohy{%
425       \leavevmode
426       \ifbody
427         \BodyIkona
428       \fi
429       {\bfseries\the\PoradiDatabaze. }}
```

Při generování testu bude potřeba vytisknout jednotlivé úlohy a v případě povolených řešení i tato řešení. Oba typy dat jsme již dříve zapsali do souboru `databaze.pkd` a uložili do maker typu `\csname X|<m>|<n>|\endcsname`, kde `X` značí buď `U` nebo `R`. Pro krátkost a přehlednost definujeme příkaz `\tiskni`, který zadáním specifikace `X|<m>|<n>|` vytiskne takový objekt.

```
430     \def\tiskni #1 {%
431       {\csname #1\endcsname}\par}
```

Pokud uživatel nastaví ovladač vypnutí náhodného generování na `\novy 0`, bude při příští kompilaci potřeba znát náhodnou volbu úloh z minulé kompilace. Proto definujeme příkaz `\ZapisPamet`, který takový zápis do paměti provádí. Objekt typu `X|<m>|<n>|` bude pracovně uložen v souboru s příponou `pmt` jako `XP|<m>|` (v případě `X` = `R` pouze tehdy, pokud je uveden v souboru `databaze.pkd`, viz `\ifcsname` na řádku 435). Pro příslušný zápis deklarujeme odpovídající identifikátor souboru `\pametSoubor`.

```
432     \newwrite\pametSoubor
433
434     \def\ZapisPamet #1|#2|#3|{%
435       \ifcsname #1|#2|#3|\endcsname
436         \zapis\pametSoubor{%
437           \nx\ea\def\nx\csname #1P|#2|\endcsname{%
438             \nx\csname #1|#2|#3|\endcsname}}
439       \fi}
```

Nyní zavádíme čítač `\NahodneCislo` a definujeme makro `\generujNC` generující pseudonáhodné číslo v daném rozsahu a uloží ho do čítače `\NahodneCislo`. Horním ohraničením rozsahu pro náš náhodný výběr je zřejmě počet úloh dané databáze.

```
440     \newcount\NahodneCislo
441
442     \def\generujNC{%
```

```
443       \NahodneCislo=\pdfuniformdeviate
444         \csname PocetUloh|\the\PoradiDatabaze|\endcsname
445       \advance\NahodneCislo by 1}
```

4.3.6. Generování testu

Pro náhodné vygenerování testu definujeme příkaz `\generujTEST`. Nejprve pokládáme `\PoradiDatabaze=1` a v případě, že uživatel nastavil `\vyres 0`, bude vypnuto číslování stránek dokumentu (předpokládáme, že zadání testu je na jediné straně, zatímco při zobrazení řešení může být těchto stran více). Následně sázíme hlavičku testu.

```
446     \def\generujTEST{%
447       \PoradiDatabaze=1
448       \ifvyres
449       \else
450         \thispagestyle{empty}
451       \fi
452       \Hlavicka
```

Pokud je nastaveno `\novy 1`, otevřeme paměťový soubor pro zápis aktuálního náhodného výběru. Pokud je nastaveno `\novy 0`, paměťový soubor naopak načítáme (předpokládáme, že byl vytvořen při předchozí kompilaci).

```
453       \ifnovy
454         \otevri\pametSoubor vyber.pmt
455       \else
456         \input vyber.pmt
457       \fi
```

Nyní spouštíme cyklus přes všechny databáze a provádíme náhodný výběr jednotlivých úloh.

```
458       \loop\ifnum\PocetDatabazi<\PoradiDatabaze
459       \else
460         \OdsadUlohu
```

Při `\novy 1` generujeme náhodné číslo uložené v čítači `\NahodneCislo`, a to v mezích daných počtem úloh aktuální databáze. Dále tiskneme návěští i vylosovanou úlohu a zapisujeme zadání úlohy a případně i řešení do paměťového souboru.

```
461         \ifnovy
462           \generujNC
463           \NavestiUlohy
464           \tiskni U|\the\PoradiDatabaze|\the\NahodneCislo|
465           \ZapisPamet U|\the\PoradiDatabaze|\the\NahodneCislo|
466           \ZapisPamet R|\the\PoradiDatabaze|\the\NahodneCislo|
```

Pokud je nastaveno `\vyres 1` a řešení vylosované úlohy existuje, vytiskneme toto řešení s ostatními náležitostmi.

```
          \ifvyres
            \ifcsname%
              R|\the\PoradiDatabaze|\the\NahodneCislo|\endcsname
              \OdsadReseni
              {\footnotesize\NavestiReseni
               \tiskni R|\the\PoradiDatabaze|\the\NahodneCislo|}
            \fi
          \fi
```

Pokud je nastaveno `\novy 0`, tiskneme úlohu dané dílčí databáze uloženou při předchozím náhodném výběru. Pokud je navíc uvedeno `\vyres 1`, sázíme při existujícím řešení i toto řešení.

```
        \else
          \NavestiUlohy
          \tiskni UP|\the\PoradiDatabaze|
          \ifvyres
            \OdsadReseni
            \ifcsname RP|\the\PoradiDatabaze|\endcsname
              {\footnotesize\NavestiReseni
               \tiskni RP|\the\PoradiDatabaze| }
            \fi
          \fi
        \fi
```

Nyní navyšujeme proměnnou `\PoradiDatabaze` a cyklus opakujeme až do splnění podmínky `\PoradiDatabaze=\the\PocetDatabazi`.

```
        \advance\PoradiDatabaze by 1
      \repeat
```

Pokud je nastaveno `\novy 1`, ukončíme ještě na konci zápis do paměťového souboru.

```
      \ifnovy
        \zavri\pametSoubor
      \fi}\fi
```

Odkazy

1. HÀN THÉ, Thành et al. The pdfTEX user manual [online]. 2018 [cit. 2020-03-24]. Dostupné z: `http://texdoc.net/texmf-dist/doc/pdftex/manual/pdftex-a.pdf`.
2. OLŠÁK, Petr. *TEX pro pragmatiky.* CSTUG, 2016.
3. OLŠÁK, Petr. *TEXbook naruby.* Konvoj, 2001.

Summary: Multilingual pseudorandomly generated tests from databases

The aim of this paper is the description of the class `ngt.cls` that was created to simplify the preparation of written tests for educators with common user knowledge of LaTeX. The described simplification consists mainly in pseudorandom generation of tests from a prepared database of problems. Further advantages of the created system include the ease of the control for the end user and the possibility of creating a version with or without results. Writing the problems in the database file is designed to work with any number of language versions in a single source file with easy switching between them.

Marian Genčev, VŠB – Technická univerzita Ostrava

Markdown 2.8.1: Směle k trůnu odlehčeného značkování v TEXu

Vít Novotný

Markdown je odlehčený značkovací jazyk, který je určený pro přípravu strukturně jednoduchých dokumentů. Existující systémy pro sazbu dokumentů v jazyce Markdown využívají často TEX, ale zachází s ním jako s černou skříňkou, čímž znemožňují použití standardních TEXových balíků a nástrojů. Oproti tomu balík Markdown umožňuje sazbu dokumentů v jazyce Markdown přímo z TEXu místo toho, aby se snažil nahradit TEX omezenějším systémem.

Od svého vydání v roce 2016 obdržel balík Markdown množství významných aktualizací, které přidaly nové funkce a zlepšily uživatelskou přístupnost balíku. V tomto článku si balík Markdown krátce představíme a podíváme se na novinky v jeho funkcionalitě a dokumentaci.

Klíčová slova: Markdown, Lua, plain TEX, LATEX, ConTEXt, Pandoc

Úvod

Za hlavní přednosti TEXu lze považovat jeho programovatelnost a typografické funkce a až v druhé řadě jeho syntax. S výjimkou matematiky jsou odlehčené značkovací jazyky jako Markdown [1] snáze (nau)čitelné a umožňují export do desítek výstupních formátů.

Existující nástroje pro sazbu dokumentů v jazyce Markdown, jakými jsou např. Pandoc a MultiMarkdown, mají několik významných nevýhod [2, 3]: zachází s TEXem jako s černou skříňkou, mají nekonzistentní podporu pro používání TEXových maker uvnitř markdownových dokumentů, znesnadňují údržbu existujících markdownových dokumentů a nejsou dostupné ve webových službách postavených na TEXu, jako je např. webový editor Overleaf. Balík Markdown řeší veškeré zmíněné nevýhody.

V předchozích článcích [2, 3] jsem představil balík Markdown [4] ve verzi 2.5.3, která trpěla několika vlastními problémy: Balík nefungoval při uvedení volby `-output-directory` na příkazové řádce TEXu a zaplňoval souborový systém konvertovanými markdownovými dokumenty, které musel uživatel ručně mazat. Dokumentace balíku byla vyčerpávající, ale pro běžného uživatele příliš technická.

V tomto článku představím verzi 2.8.1 balíku Markdown, která zmíněné problémy předchozích verzí řeší a zároveň přináší nové funkce, jako je porcování obsahu, značkování tabulek a konverze markdownových dokumentů z příkazové řádky.

 doi: 10.5300/2020-1-2/48

Nové funkce

Mezi verzemi 2.5.3 a 2.8.1 balíku Markdown byla jedna významná záplatová verze, 2.5.4, a tři významné malé verze, 2.6.0, 2.7.0 a 2.8.0. Verze 2.5.4 přidala podporu pro volbu `-output-directory` TEXu. Verze 2.6.0 představila rozhraní pro konverzi markdownových dokumentů z příkazové řádky a přidala podporu pro LATEXový balík `doc` [5]. Verze 2.7.0 umožnila porcování obsahu a představila nový uživatelský manuál. Verze 2.8.0 rozšířila porcování obsahu a umožnila značkovat tabulky pomocí syntaktického rozšíření jazyka Markdown.

V této sekci představím jednotlivé nové funkce balíku Markdown. Ačkoliv balík podporuje i formáty plain TEX a ConTEXt, veškeré ukázky jsou pro přístupnost ve formátu LATEX.

Změna pracovního adresáře

TEX je možné spustit s volbou `-output-directory`, která z pohledu TEXového dokumentu změní pracovní adresář. Toto umožňuje uživateli přesměrovat dočasné soubory TEXu do samostatného adresáře. Veškeré externí programy spuštěné pomocí mechanismu `\write18` jsou ale spuštěné v původním pracovním adresáři: Například externí interpret jazyka Lua spouštěný balíkem Markdown pracuje s dočasnými soubory TEXu, a pokud je nenalezne v pracovním adresáři, selže.

Pro vyřešení tohoto problému představila verze 2.5.4 balíku Markdown volbu `outputDir`, která informuje interpret jazyka Lua o tom, v jakém adresáři se nachází pomocné soubory TEXu. Pro vyzkoušení si vytvořte textový dokument `dokument.tex` s následujícím obsahem:

```
\documentclass{article}
\usepackage[outputDir=/dev/shm]{markdown}
\begin{document}
\begin{markdown}
Nadpis první úrovně
===================
Nadpis druhé úrovně
-------------------
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Vestibulum ac commodo. Toto je pouhý odstavec textu.
\end{markdown}
\end{document}
```

Nakonec spusťte následující příkaz pro vysázení markdownového dokumentu s jednou sekcí, jednou podsekcí a jedním odstavcem (vizte obrázek 1):

```
pdflatex -output-directory /dev/shm -shell-escape dokument.tex
```

Dočasné soubory budou přesměrovány do adresáře `/dev/shm`, což je RAM disk dostupný v aktuálním Linuxovém jádře.

Nadpis první úrovně

Nadpis druhé úrovně

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum ac commodo.
Toto je pouhý odstavec textu.

Obrázek 1: Markdown dokument s jednou sekcí, podsekcí a odstavcem

Markdown nám umožňuje část textu *zdůraznit*.
Pomocí dvou hvězdiček nebo podtržítek značíme **silný důraz**.

Obrázek 2: Markdown dokument s jedním odstavcem, který obsahuje zdůrazněný a silně zdůrazněný text

Rozhraní pro příkazovou řádku

Balík Markdown předává markdownové dokumenty externímu interpretu jazyka Lua. Intepret jazyka Lua převede markdownové dokumenty do TEXu a předá je zpět balíku Markdown, který je vysází, vizte obrázek 3. Výhodou tohoto postupu je automatizovanost. Zároveň však tento postup skýtá i množství nevýhod: Převedené markdownové dokumenty jsou ukládány na souborový systém, kde zabírají stále rostoucí množství místa. Pokud použitá varianta TEXu neobsahuje interpret jazyka Lua, vyžaduje balík Markdown přístup k příkazové řádce systému, což je bezpečnostní riziko. Postup je komplexní a z pohledu uživatele obtížně laditelný.

Řešením výše uvedených problémů je rozdělení postupu na dva kroky. V prvním kroku uživatel převede markdownové dokumenty do TEXu a v druhém kroku uživatel převedené markdownové dokumenty vysází, vizte obrázek 4. Před prvním krokem může uživatel odstranit všechny dříve převedené markdownové dokumenty. Před druhým krokem může uživatel libovolně upravovat převedené markdownové dokumenty. Během druhého kroku nepotřebuje balík Markdown přístup k příkazové řádce systému. Oddělení jednotlivých kroků postupu usnadňuje ladění.

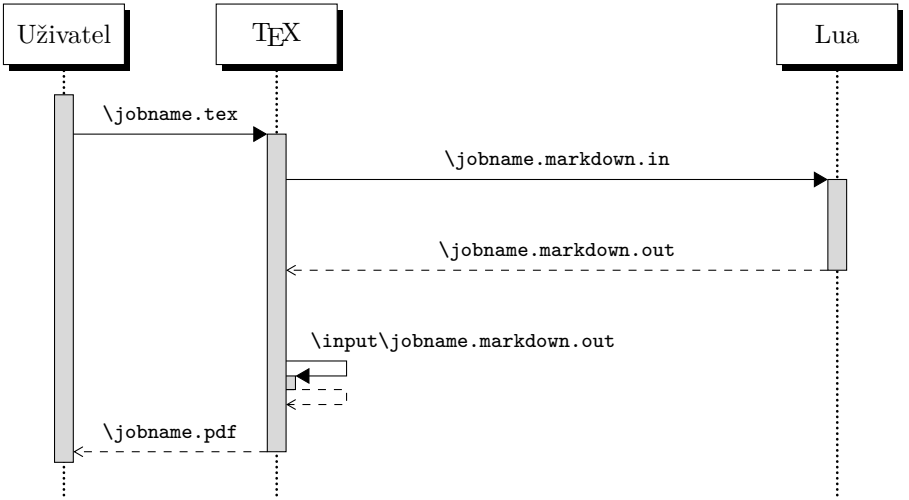
Verze 2.6.0 balíku Markdown proto zavedla rozhraní pro příkazovou řádku ve formě samostatného programu v jazyce Lua. Pro vyzkoušení si vytvořte textový dokument `priklad.md` s následujícím obsahem:

```
14 Markdown nám umožňuje část textu *zdůraznit*.␣␣
15 Pomocí dvou hvězdiček nebo podtržítek značíme **silný důraz**.
```

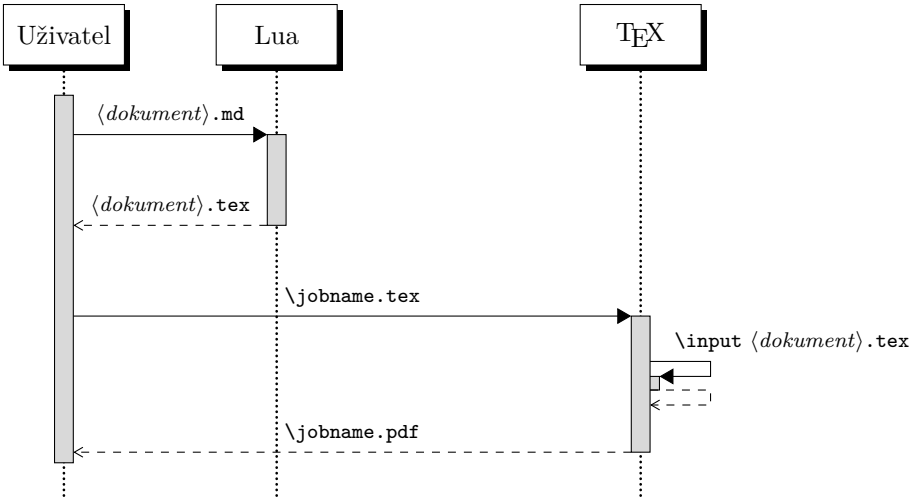
Dále spusťte následující příkaz pro zjištění umístění programu, který zpřístupňuje příkazovou řádku balíku Markdown:

```
16 kpsewhich markdown-cli.lua
```

Možným umístěním v distribuci TEX Live 2019 může být například `/usr/local/texlive/2019/texmf-dist/scripts/markdown/markdown-cli.lua`. Posléze



Obrázek 3: Sekvenční diagram sázení dokumentu v jazyce Markdown pomocí TEXového rozhraní balíku Markdown



Obrázek 4: Sekvenční diagram sázení dokumentu v jazyce Markdown pomocí rozhraní balíku Markdown pro příkazovou řádku

spusťte následující příkaz pro převod markdownového dokumentu `priklad.md` na TEXový dokument `priklad.tex`:

```
17 texlua /umístění/markdown-cli.lua -- priklad.md priklad.tex
```

Dále vytvořte textový dokument `dokument.tex` s následujícím obsahem:

```
18 \documentclass{article}
19 \usepackage{markdown}
20 \begin{document}
21 \input priklad
22 \end{document}
```

Nakonec spusťte následující příkaz pro vysázení markdownového dokumentu s jedním odstavcem, který obsahuje zdůrazněný a silně zdůrazněný text (vizte obrázek 2):

```
23 pdflatex dokument.tex
```

Dokumentace LATEXových balíků

LATEXový balík `doc` umožňuje dokumentaci jiných balíků vepsáním LATEXového dokumentu s dokumentací do komentářů ve zdrojovém textu balíku. Tento způsob vychází z Knuthova literárního programování [6] a je oblíbený mezi autory LATEXových balíků, protože udržuje dokumentaci poblíž dokumentovaného kódu.

Použití odlehčeného značkování v dokumentaci může zvýšit čitelnost zdrojového kódu a učinit ho tím snáze přístupným vývojářské komunitě. Za tímto účelem zavedla verze 2.6.0 balíku Markdown volbu `stripPercentSigns`, která informuje interpret jazyka Lua, aby ignoroval znak uvozující komentáře. Pro vyzkoušení si vytvořte textový dokument `dokument.dtx` s následujícím obsahem:

```
24 % \iffalse
25 \documentclass{ltxdoc}
26 \usepackage[T1]{fontenc}
27 \usepackage[stripPercentSigns, hashEnumerators]{markdown}
28 \begin{document}
29 \DocInput{dokument.dtx}
30 \end{document}
31 % \fi
32 % \begin{markdown}
33 % * Bylo smažno, lepě svihlí tlové
34 % * se batoumali v dálnici,
35 % + chrudošní byli borolové,
36 % - na mamné krsy žárnící.
37 %
38 % 1. „Ó synu, střez se Žvahlava,
39 % 2. má zuby, drápy přeostré;
40 % #. střez se i Ptáka Neklava,
41 % #. zuřmící Bodostre!"
42 % \end{markdown}
```

- Bylo smažno, lepě svihlí tlové
- se batoumali v dálnici,
- chrudošní byli borolové,
- na mamné krsy žárnící.

1. „Ó synu, střez se Žvahlava,
2. má zuby, drápy přeostré;
3. střez se i Ptáka Neklava,
4. zuřmící Bodostre!“

Obrázek 5: Markdown dokument s jedním odrážkovým seznamem a jedním číslovaným seznamem. Text je převzatý z Alenčina dobrodružství v říši divů a za zrcadlem v překladu Jaroslava Císaře.

Nakonec spusťte následující příkaz pro vysázení markdownového dokumentu s jedním odrážkovým seznamem a jedním číslovaným seznamem (vizte obrázek 5):

```
pdflatex -shell-escape dokument.dtx
```

Porcování obsahu

Při sazbě dokumentů je často užitečné, z důvodů jak z časových tak autorských, tyto dokumenty rozdělit nebo vhodně přeskládat jejich obsah: Přednáškové slajdy pro celý semestr lze rozdělit na samostatné dokumenty pro jednotlivé týdny. Z desítek minut strávených sazbou monografie se stanou vteřiny, pokud se omezíme na vybrané části. Neuspořádané poznámky můžeme při sazbě přeskládat do soudržného dokumentu.

Pro usnadnění formátování markdownových dokumentů existuje syntaktické rozšíření jazyka Markdown, které umožňuje přidělovat prvkům jazyka HTML atributy. Díky HTML atributům se můžeme jednoznačně odkazovat na jednotlivé sekce markdownových dokumentů. Verze 2.7.0 balíku Markdown podporuje zmíněné syntaktické rozšíření pomocí volby `headerAttributes`. Pomocí voleb `slice` a `shiftHeadings` můžeme následně vysázet pouze malou část markdownového dokumentu a měnit úrovně nadpisů. Pro vyzkoušení si vytvořte textový dokument `dokument.tex` s následujícím obsahem:

```
\documentclass{article}
\usepackage[headerAttributes]{markdown}
\begin{filecontents*}{recept.md}
# Palačinky
Moučník z mouky, mléka a vajec. Podávejte s marmeládou.
## Třetí krok  {#krok3}
Rychlým pohybem katapultujeme palačinku na strop. Hola hop!
### První krok {#krok1}
Přísady rozmixujeme.
## Druhý krok  {#krok2}
Výslednou směs nalijeme na rozpálenou pánev.
\end{filecontents*}
```

```
\begin{document}
\markdownInput[slice=^ ^krok3]{recept.md}
\markdownInput[slice=krok1, shiftHeadings=-1]{recept.md}
\markdownInput[slice=krok2]{recept.md}
\markdownInput[slice=krok3 ^krok1]{recept.md}
\end{document}
```

Nakonec spusťte následující příkaz pro vysázení markdownového dokumentu s jednou sekcí a třemi podsekcemi ve správném pořadí (vizte obrázek 6):

```
pdflatex -shell-escape dokument.tex
```

Značkování tabulek

Jedním z nejsložitějších prvků odlehčeného značkovacího jazyka bývají zpravidla tabulky. Jazyk Markdown tabulky nepodporuje, ale existuje množství syntaktických rozšíření, které podporu pro značkování tabulek přidávají.

Od verze 2.8.0 podporuje balík Markdown syntaktické rozšíření pro značkování tabulek pomocí volby `pipeTables` a syntaktické rozšíření pro značkování popisků tabulek pomocí volby `tableCaptions`. Pro vyzkoušení si vytvořte textový dokument `dokument.tex` s následujícím obsahem:

```
\documentclass{article}
\usepackage[pipeTables, tableCaptions]{markdown}
\begin{filecontents*}{tabulka.md}
|   Příjmení | Jméno   |     Funkce     | Přítomnost |
|-----------:|:--------|:--------------:|:-----------|
|      Němec | Karel   |    náčelník    | zde        |
|       Šofr | Vojtěch |    lékárník    | zde        |
| Fryštenský | Varel   |      potah     | zde        |
|    Poustka | Václav  | pomocný učitel | chybí      |
: Docházka bývalé polární výpravy
```

Palačinky

Moučník z mouky, mléka a vajec. Podávejte s marmeládou.

První krok

Přísady rozmixujeme.

Druhý krok

Výslednou směs nalijeme na rozpálenou pánev.

Třetí krok

Rychlým pohybem katapultujeme palačinku na strop. Hola hop!

Obrázek 6: Markdown dokument s jednou sekcí a třemi podsekcemi

Příjmení	Jméno	Funkce	Přítomnost
Němec	Karel	náčelník	zde
Šofr	Vojtěch	lékárník	zde
Fryštenský	Varel	potah	zde
Poustka	Václav	pomocný učitel	chybí

Tabulka 1: Docházka bývalé polární výpravy

Obrázek 7: Markdown dokument s jednou tabulkou a jejím popiskem

```
\end{filecontents*}
\begin{document}
\markdownInput{tabulka.md}
\end{document}
```

Nakonec spusťte následující příkaz pro vysázení markdownového dokumentu s tabulkou a jejím popiskem (vizte obrázek 7):

```
pdflatex -shell-escape dokument.tex
```

Uživatelský manuál

Tufte [7] tvrdí, že systémy pro přípravu prezentačních slajdů se nedílně pojí s kognitivním stylem, který brání efektivní komunikaci. Stejně tak literární programování v podobě, v jaké ho aplikují autoři TEXových balíků, často produkuje špatně čitelnou dokumentaci, která je příliš pevně svázána se strukturou dokumentovaného kódu.[1] Některé TEXové balíky obsahují samostatný uživatelský manuál, který je připravený nezávisle na kódu dokumentovaného balíku, což obvykle zvyšuje čitelnost, ale za cenu horší udržovatelnosti dokumentace.

Před verzí 2.5.6 obsahoval balík Markdown pouze technickou dokumentaci [8], která byla připravena literárním programováním a která byla určena pro vývojáře a pokročilé uživatele. Od verze 2.5.6 obsahoval balík Markdown také uživatelský manuál [9], který se zaměřoval na běžné uživatele. Stejně jako technická dokumentace byl ale i tento uživatelský manuál přímočaře vytvořen literárním programováním, díky čemuž byl špatně strukturovaný a obtížně čitelný. Od verze 2.7.0 je proto uživatelský manuál vytvořen ze tří textů, které popisují uživatelské rozhraní, volby balíku a prvky jazyka Markdown. Výsledkem je (doufám) čitelný uživatelský manuál, který je ale stále připraven pomocí literárního programování.

[1] Knuthovo literární programování [6] umožňuje rekurzivně definovat program po malých samostatně dokumentovaných částech. LATEXové balíky `doc` a `docstrip` umožňují pouze generovat soubor se zdrojovým textem programu v takovém sledu, v jakém je program dokumentován.

Odkazy

1. GRUBER, John. *Daring Fireball: Markdown* [online]. 2013 [cit. 2019-03-31]. Dostupné z: `https://daringfireball.net/projects/markdown`.
2. NOVOTNÝ, Vít. Sazba textu označkovaného v jazyce Markdown uvnitř TEXových dokumentů. *Zpravodaj Československého sdružení uživatelů TEXu* [online]. 2016, roč. 26, č. 1–4, s. 78–217 [cit. 2019-03-31]. ISSN 1213-8185. Dostupné z DOI: `10.5300/2016-1-4/78`.
3. NOVOTNÝ, Vít. Using Markdown Inside TEX Documents. *TUGboat* [online]. 2017, roč. 38, č. 2, s. 214–217 [cit. 2019-03-31]. ISSN 0896-3207. Dostupné z: `https://tug.org/TUGboat/tb38-2/tb119novotny.pdf`.
4. NOVOTNÝ, Vít. *A package for converting and rendering markdown documents inside TEX* [online]. 2019 [cit. 2019-11-26]. Dostupné z: `https://ctan.org/pkg/markdown`.
5. MITTELBACH, Frank. *Doc – Format LATEX documentation* [online]. 2018 [cit. 2019-03-31]. Dostupné z: `https://ctan.org/pkg/doc`.
6. KNUTH, Donald Ervin. Literate programming. *The Computer Journal*. 1984, roč. 27, č. 2, s. 97–111. Dostupné z DOI: `10.1093/comjnl/27.2.97`.
7. TUFTE, Edward Rolf. *The Cognitive Style of PowerPoint*. 2. vyd. Graphics Press Cheshire, CT, 2006. ISBN 0-961-39216-9.
8. NOVOTNÝ, Vít. *A Markdown Interpreter for TEX* [online]. 2019 [cit. 2019-11-26]. Dostupné z: `http://mirrors.ctan.org/macros/generic/markdown/markdown.pdf`. Verze 2.8.1.
9. NOVOTNÝ, Vít. *Markdown Package User Manual* [online]. 2019 [cit. 2019-11-26]. Dostupné z: `http://mirrors.ctan.org/macros/generic/markdown/markdown.html`. Verze 2.8.1.

Summary: Markdown 2.8.1: Boldly Unto the Throne of Lightweight Markup in TEX

Markdown is a lightweight markup language that makes it easy to write structurally simple documents. Existing tools for rendering markdown documents to PDF treat TEX as a black box. In contrast, the Markdown package provides support for styling and typesetting markdown documents in TEX, extending a TEXie's toolbox rather than forcing them to replace TEX with a more limited tool.

Since its release in 2016, the package has received several important updates improving the functionality and user experience. In this article, I will reintroduce the package, and describe its new functionality and documentation.

Keywords: Markdown, Lua, plain TEX, LATEX, CONTEXT, Pandoc

Vít Novotný, witiko@mail.muni.cz

Zpracování dat z tabulkového editoru TEXem

Jan Šustek

Abstrakt: V článku je ukázán způsob, jak je možné načíst a zpracovat v TEXu data vytvořená v tabulkovém editoru. Uvedená makra jsou podrobně popsána s cílem motivovat čtenáře nebát se v TEXu drobně programovat. Dále je v článku ukázáno třířádkové makro na vložení celé tabulky do TEXového dokumentu.

Klíčová slova: CSV, tabulka.

1 Úvod

Poslední dobou vyšlo několik článků, které se zabývají sazbou tabulek v TEXu. Marek Pomp [1] se věnuje sazbě tabulek vygenerovaných statistickým softwarem R s využitím balíčků LATEXu. Tomáš Hála [2] popisuje tvorbu tabulek v ConTEXtu.

Tento článek řeší problematiku na úrovni PlainTEXu[1] a dodržuje tři zásady OPmacu [3]:

- V jednoduchosti je síla.
- Makro není univerzální, ale je čitelné a srozumitelné.
- Uživatel si makro může snadno předefinovat k obrazu svému.

Uvedená makra jsou v článku podrobně popsána s cílem motivovat čtenáře nebát se v TEXu naprogramovat si jednoduchá makra, a tím si usnadnit práci.

Podobný problém řeší soubor scancsv.tex [4] od Petra Olšáka, makra jsou však obecnější a umožňují například používat jiný oddělovač položek, než je středník. Daní za toto je menší srozumitelnost a komplikovanější definice použitých maker.

2 Příklad

V sekci 3 si vytvoříme soubor nacticsv.tex a v něm nadefinujeme makro

```
1 \zpracujsoubor⟨soubor⟩⟨činnost⟩
```

Funkcí makra bude načíst ⟨*soubor*⟩ ve formátu CSV řádek po řádku a s každým řádkem provést ⟨*činnost*⟩. Jednotlivé údaje na řádku jsou uloženy do maker `\udaji`, `\udajii`, `\udajiii`, `\udajiv` atd. Počet údajů na daném řádku je uložen

[1] Makra jsou použitelná i v dalších formátech a nadstavbách TEXu.

DOI: 10.5300/2020-1-2/57

do číselného registru \pocetudaju. Číslo aktuálně zpracovávaného řádku CSV souboru je uloženo do číselného registru \cisloradkucsv.

Představme si, že máme v tabulkovém editoru tabulku se jmény účastníků nějakého závodu a jejich výsledky. Tuto tabulku uložíme ve formátu CSV do souboru 1.csv.

```
Jiří;Novák;Praha;48:29
Petr;Novotný;Ostrava;49:17
Jan;Svoboda;Brno;52:13
Martin;Procházka;Olomouc;53:07
Josef;Dvořák;Plzeň;53:43
Tomáš;Kučera;České Budějovice;56:59
Pavel;Černý;Liberec;57:23
Jaroslav;Veselý;Ústí nad Labem;1:02:31
```

Úkolem bude z této tabulky vysázet výsledkovou listinu v následujícím tvaru.

Jiří Novák (Praha) 48:29
Petr Novotný (Ostrava) 49:17
Jan Svoboda (Brno) 52:13
Martin Procházka (Olomouc) 53:07
Josef Dvořák (Plzeň) 53:43
Tomáš Kučera (České Budějovice) 56:59
Pavel Černý (Liberec) 57:23
Jaroslav Veselý (Ústí nad Labem) 1:02:31

K tomu si připravíme makro \vypisucastnika, které vysází jeden řádek výsledkové listiny. Použitím tohoto makra v argumentu makra \zpracujsoubor pak postupně vysázíme celou výsledkovou listinu.

```
\input nacticsv
\def\vypisucastnika{\hbox to\hsize{%
  \ifnum\cisloradkucsv<4 \bf \else \rm \fi
  \udaji\ \udajii\ (\udajiii)\dotfill\udajiv}}
\zpracujsoubor{1.csv}\vypisucastnika
```

3 Implementace

V této sekci je stručně popsán celý soubor nacticsv.tex.

Nejdříve nadeklarujeme dříve uvedené číselné registry a nadefinujeme si pomocné makro, které budeme využívat při testování podmínek uvnitř cyklů.

```
\newcount\pocetudaju
\newcount\cisloradkucsv
\def\nic{}
```

V makru `\radek` je vždy uložen obsah celého aktuálního řádku souboru. Makro `\rozdelradek` tento obsah rozdělí na údaje oddělené středníkem. K tomu využije makro `\rozdelradekA`, které v cyklu načte vždy jeden údaj ukončený středníkem a tento údaj uloží do příslušného makra `\udaj`n.

```
\def\rozdelradek{\pocetudaju0
  \expandafter\rozdelradekA\radek;\konecradku}
\def\rozdelradekA#1;#2\konecradku{\advance\pocetudaju1
  \expandafter\def
    \csname udaj\romannumeral\pocetudaju\endcsname{#1}%
  \def\tmp{#2}%
  \ifx\tmp\nic\let\next\relax
  \else\def\next{\rozdelradekA#2\konecradku}%
  \fi\next}
```

Makro `\zpracujsoubor` uloží svůj argument ⟨*činnost*⟩ pro pozdější použití do makra `\cinnost`. Dále lokálně nastaví `\endlinechar` na hodnotu -1, aby na konci načítaných řádků nevznikaly zavlečené mezery. Makrem `\zpracujsouborA` poté spustí cyklus přes jednotlivé řádky ⟨*souboru*⟩. V cyklu je pak každý řádek ⟨*souboru*⟩ zpracován uvnitř skupiny.[2]

Nesmíme zapomenout, že TEX na konec každého souboru načítaného pomocí `\read` automaticky přidává prázdný řádek.[3] Takový řádek samozřejmě neobsahuje žádný středník a není na něj možné aplikovat makro `\rozdelradek`. Tato situace je ošetřena testem na řádku 35.

```
\newread\vstup
\def\zpracujsoubor#1#2{{\def\cinnost{#2}%
  \openin\vstup#1
  \cisloradkucsv0
  \endlinechar-1
  \zpracujsouborA
  \closein\vstup}}
\def\zpracujsouborA{\read\vstup to\radek
  \ifx\radek\nic\else
    \advance\cisloradkucsv1
    {\rozdelradek
    \cinnost}%
  \fi
  \ifeof\vstup\else\expandafter\zpracujsouborA\fi}
```

[2] Pokud bychom skupinu nevytvořili, pak by v případě, že některý řádek obsahuje méně údajů, ve zbývajících makrech `\udaj`n nesprávně zůstaly uloženy údaje z předchozích řádků.

[3] Knuth tento jev bez zdůvodnění popisuje v [5] na straně 217.

4 Pojmenované sloupce

V této sekci se budeme zabývat situací, kdy v prvním řádku CSV souboru jsou uložena záhlaví sloupců. My tato záhlaví využijeme k pojmenování maker obsahujících údaje, namísto obecného pojmenování `\udaj`n. Ukážeme, jak je k tomuto účelu možné efektivně využít příkaz `\csname`, bez nutnosti zasahovat do souboru nacticsv.tex.

U našeho příkladu s výsledkovou listinou to znamená, že budeme mít následující data v tabulce uložené v souboru 2.csv.

```
jmeno;prijmeni;mesto;cas
Jiří;Novák;Praha;48:29
Petr;Novotný;Ostrava;49:17
Jan;Svoboda;Brno;52:13
Martin;Procházka;Olomouc;53:07
Josef;Dvořák;Plzeň;53:43
Tomáš;Kučera;České Budějovice;56:59
Pavel;Černý;Liberec;57:23
Jaroslav;Veselý;Ústí nad Labem;1:02:31
```

To nám umožní zpřehlednit makro `\vypisucastnika` a namísto řádku 13 psát řádek 52.

```
\def\vypisucastnika{\hbox to\hsize{%
  \ifnum\cisloradkucsv<4 \bf \else \rm \fi
  \jmeno\ \prijmeni\ (\mesto)\dotfill\cas}}
```

Zpracování prvního řádku CSV souboru odlišíme od zpracování zbývajících řádků. To provedeme pomocí makra `\jedenradek`, které je na začátku definováno jako makro `\prvniradek`, avšak uvnitř makra `\prvniradek` je předefinováno, aby se na dalších řádcích chovalo jako makro `\vypisucastnika`.

Uvnitř makra `\prvniradek` načteme záhlaví sloupců a nadefinujeme příslušná makra, ať se expandují na správná makra `\udaj`n. Zároveň opravíme hodnotu registru `\cisloradkucsv`, aby počítal pouze datové řádky souboru a ne řádek se záhlavím. S ohledem na poznámku pod čarou 2 musí být všechna nastavení v makru `\prvniradek` provedena globálně.

```
\input nacticsv
\newcount\tmpnum
\def\prvniradek{%
  \tmpnum0
  \loop \ifnum\tmpnum<\pocetudaju \advance\tmpnum1
    \expandafter\gdef
      \csname
```

```
60          \csname udaj\romannumeral\tmpnum\endcsname\expandafter
61        \endcsname\expandafter
62      {\csname udaj\romannumeral\tmpnum\endcsname}%
63    \repeat
64    \global\let\jedenradek\vypisucastnika
65    \global\cisloradkucsv0
66    }
67 \let\jedenradek\prvniradek
68 \zpracujsoubor{2.csv}\jedenradek
```

Hlavní trik je na řádcích 58–62, které stojí za to rozebrat důkladněji. Předpokládejme, že uvnitř makra `\prvniradek` v cyklu na řádku 57 máme `\tmpnum` = 1, tedy že zpracováváme první údaj prvního řádku. Díky makru `\rozdelradek` máme nadefinováno (viz řádek 22)

```
69 \def\udaji{jmeno}
```

Nyní se na řádku 58 expanduje příkaz `\expandafter`. V rámci této expanze se jedenkrát expanduje příkaz `\csname` na řádku 59. Tím dojde k úplné expanzi řádku 60:

1. Příkaz `\csname` na řádku 60 se postupně expanduje na

   ```
   70 \csname udaji\endcsname
   71 \udaji
   72 jmeno
   ```

2. Príkaz `\expandafter` na řádku 60 se expanduje tak, že se expanduje příkaz `\expandafter` na řádku 61 a ten se expanduje tak, že se jedenkrát expanduje příkaz `\csname` na řádku 62. Tímto na řádku 62 dostáváme

   ```
   73 {\udaji}%
   ```

Výsledkem expanze příkazu `\csname` z řádku 59 tak je, že z řádků 59–61 vznikl jediný token `\jmeno`. Expanzi příkazu `\expandafter` z řádku 58 tak máme hotovou a ve čtecí frontě máme tokeny

```
74 \gdef\jmeno{\udaji}%
```

Když poté, například na druhém řádku souboru, uživatel zavolá makro `\jmeno`, to se expanduje na makro `\udaji` a to se expanduje na výsledný text `Jiří`.

5 Přímé vysázení tabulky

Na závěr si ukážeme třířádkové řešení z [6], jak je možné v OPmacu přímo vysázet tabulku z CSV souboru.

Vytvoříme si makro `\csvtable`⟨*deklarace*⟩⟨*soubor*⟩, které z CSV souboru ⟨*soubor*⟩ vysází tabulku, jejíž sloupce budou zarovnány podle ⟨*deklarace*⟩. Zarovnání sloupců je definováno stejně jako v OPmacovém makru `\table`. V souboru jsou jednotlivé řádky tabulky odděleny koncem řádku souboru. Jednotlivé buňky jsou odděleny středníkem.

Nechť soubor 3.csv obsahuje text

```
a;bc;de
fgh;ijk;l
m;nop;qrsr
```

Potom budeme chtít, aby se po použití

```
\csvtable{|l|r|r|}{3.csv}
```

vysázela následující tabulka.

a	bc	de
fgh	ijk	l
m	nop	qrsr

Práci makra `\csvtable` vložíme do skupiny, v níž lokálně přiřadíme znaku středník kategorii 4 (bude se chovat jako oddělovač buněk v tabulce) a znak konce řádku nadefinujeme jako aktivní, aby se expandoval na `\crl`. Pak už jen použijeme makro `\table` a v něm načteme daný soubor. Dostáváme následující definici makra `\csvtable`.

```
\def\csvtable#1#2{{\catcode`\;=4 \adef{\^^M}{\crl}
  \table{#1}{\crl\input#2}}}
```

Nyní si naše makro vylepšíme tak, aby umělo zpracovávat buňky obsahující středník. Obsah takové buňky je uzavřen do uvozovek.

Nechť soubor 4.csv obsahuje text

```
a;bc;de
fgh;"ij;k";l
m;nop;"qr;sr"
```

Potom budeme chtít, aby se po použití

```
\csvtable{|l|r|r|}{4.csv}
```

vysázela následující tabulka.

a	bc	de
fgh	ij;k	l
m	nop	qr;sr

Pro tyto účely stačí makro `\csvtable` rozšířit o jeden řádek. Znak uvozovky nastavíme jako aktivní, ať otevře skupinu, uvnitř které

- bude mít znak středník kategorii 12 (vysází se) a
- znak uvozovky bude aktivní a zavře skupinu.

Celá definice makra `\csvtable` pak je tato.

```
\def\csvtable#1#2{{\catcode`\;=4 \adef{\^^M}{\crl}
  \adef{\"}{\begingroup\adef{\"}{\endgroup}\catcode`\;=12 }
  \table{#1}{\crl\input#2}}}
```

Seznam literatury

1. POMP, Marek. Tabulky v dobře dokumentovaných statistických výpočtech. *Zpravodaj ČSTUG*. 2018, roč. 25, č. 1–4, s. 22–37. ISSN 1211-6661. Dostupné z DOI: 10.5300/2018-1-4/22.
2. HÁLA, Tomáš. Tabulky v ConTEXtu: přístupy, možnosti, algoritmy. *Zpravodaj ČSTUG*. 2019, roč. 25, č. 1–4, s. 24–43. ISSN 1211-6661. Dostupné z DOI: 10.5300/2019-1-4/24.
3. OLŠÁK, Petr. *OPmac: rozšiřující makra plain TEXu* [online]. 2017-06-17 [cit. 2020-03-24]. Dostupné z: http://petr.olsak.net/opmac.html.
4. OLŠÁK, Petr. scancsv.tex [online]. 2005-02-26 [cit. 2020-03-24]. Dostupné z: http://petr.olsak.net/ftp/olsak/makra/scancsv.tex.
5. KNUTH, Donald Ervin. *Computers & Typesetting, Volume A: The TEXbook*. Addison-Wesley, 1986.
6. KOLOUCH, Ondřej; NOVOTNÝ, Lukáš; ŠUSTEK, Jan. *TEX for Master Level / TEX pro magisterské studium*. Ostravská univerzita, 2016.

Summary: Processing Spreadsheet Data in TEX

In the paper we show a way to read and process spreadsheet data in TEX. The macros are described in detail, motivating readers to create simple macros without doubts. We also show a three-line macro for inserting a whole table into a TEX document.

Key words: CSV, table.

Jan Šustek, jan.sustek@osu.cz
Ostravská univerzita, Přírodovědecká fakulta, Katedra matematiky
30. dubna 22, CZ-701 03 Ostrava, Czech Republic

Dva bloky otázok a odpovedí od Donalda Knutha na FI MU

Tomáš Szaniszlo

V októbri 2019 hostila Fakulta informatiky Masarykovej univerzity Donalda Knutha, ktorý pri tejto príležitosti viedol dva bloky otázok a odpovedí na témy informatiky a umenia. Po krátkom úvode k nim nájdete v tomto článku ich textové prepisy.

Klíčová slova: Donald Knuth, Umenie programovania, Záujmy bez hraníc, otázky a odpovede

Fakulta informatiky Masarykovej univerzity slávila v roku 2019 svoje 25. výročie. Pri tejto príležitosti sa konalo niekoľko oslavných akcií, z ktorých z hľadiska prestížnosti vynikli najviac tie v rámci *Týždňa s držiteľmi Turingovej ceny*, (Sojka, 2019a) kde boli hosťami Donald Knuth a Dana Stewart Scott. Počas neho bola realizovaná i česká premiéra Knuthovej skladby *Fantasia Apocalyptica*, (Knuth, 2020) o ktorej ste sa mohli dočítať v predchádzajúcom čísle Spravodaja CSTUG. (Lupták, 2019)

Donald Knuth nebol hosťom fakulty prvýkrát. Navštívil ju a prednáškami obohatil už v roku 1996, kedy tu získal čestný doktorát. (Zlatuška, 1996) O 25 rokov neskôr si pre nás pripravil dve prednášky, ktoré boli svojou formou otázok a odpovedí inšpirované formátom prednášok známeho fyzika Richarda Feynmana. Ich témy boli rámcovo vytýčené ich názvami, postupne *Umění programování (Computer Programming as Art)* a *Zájmy bez hranic (Boundless Interests)*.

Išlo vždy o moderovanú diskusiu, kde v prvom prípade bol moderátorom prof. RNDr. Jozef Gruska, DrSc., a v druhom prof. RNDr. Jiří Zlatuška, CSc. Otázky boli kladené v hojnom počte zastúpeným publikom a prioritizované pomocou platformy Slido.

V prvej prednáške sa môžete dozvedieť napríklad o tom, prečo považuje dávkové spracovanie za dôležitú pracovnú metódu, ako mu pomohlo odbremeniť sa od mailov, o zradnosti otázok „čo je vaše obľúbené X?“, čo prezradil o svojej spolupráci s NSA, ako občas zažívame renesanciu myšlienok v súvislosti s algoritmami pre magnetické pásky alebo o vzťahu zvuku fontány a detského plaču.

Druhá prednáška bola príjemne spestrená i krátkymi hudobnými ukážkami jeho vlastnej skladby Fantasia Apocalyptica prezentovanými na klávesách. Dozviete sa v nej napríklad niečo o hudobnej stránke Donalda Knutha, o témach použitých v skladbe, o tom, ako bola jej tvorba v istom zmysle problémom Constraint Logic Programming (CLP), o tom ako bojoval s vyhorením, o jeho spôsobe

 doi: 10.5300/2020-1-2/64

organizácie pracovných úloh, o programátoroch a dobrých programátoroch, o tom, čo považuje za najhorší program, ktorý videl, a tiež o jeho domácom organe.

Pár ďalších informácií o prednáškach vrátane ich videozáznamov nájdete na webstránkach týchto udalostí. (Sojka, 2019b; Sojka, 2019c) Prepisy sú dostupné aj na webe FI MU. (Szaniszlo, 2019a; Szaniszlo, 2019b) Boli vyrobené na základe tituliek automaticky vygenerovaných pomocou YouTube[1] a následne boli ručne korigované.

Questions and Answers Session 1: Computer Programming as Art

Gruska: Good afternoon. Dear informaticians, we have today very special colloquium. We have the legend of informatics, father of analysis of algorithm, father of art of programming, father of many other books, father of TEX and so on and so on... I better stop, because I would spend probably all my time here talking about his outcome. He came here to answer your questions. Sessions will be so interesting, how interesting will be the questions you ask. There are two ways to ask questions. Either by microphone or you can use mobile. Ok?

I mentioned that this legend of informatics and one argument supporting that is when in 1989 there was the IT World Computer Congress in US in San Francisco, professor Knuth was invited to give the first talk, the most important talk. So welcome, professor Knuth. Welcome here, and you can start the session, colloquium, by asking the questions from your audience. Welcome in Brno. *applause*

Knuth: Prof. Gruska, do you have a question?

Gruska: Yes! 42 years ago I wrote you letter.

Knuth: Oh!

Gruska: I was waiting for answer. After one month I decided to call you. Secretary took the phone and said: "Oh, he's three months behind answering letters. However, he is next to me." So I could talk to you. Has that changed with your custom to respond to letters?

Knuth: Can you hear me? Testing... Testing, testing. Zero one, zero one. *laughter*

So the secretary I had 42 years ago is Phyllis Winkler who served me for many years, but she died about 15 years ago and actually retired before that. However, I go into campus at Stanford four days a week, and I have lunch with the students every Thursday and with the faculty every Tuesday and things like that... However, I found that one of the great secrets of Computer Science is

[1] `https://support.google.com/youtube/answer/6373554`.

called batch processing. So when I answer mail, I don't do it by interruption, but by batch processing. So I gave up email, I guess, on January 1, 1990, and I've been a happy man ever since. But the questions come in, and they're filtered, and I answer a lot of them all at once. So it turns out that that's necessary for me in order to work efficiently. But Phyllis was a wonderful protector. So she kept it possible for me to work efficiently for me all these years.

Gruska: And there's also another question, well, they ask, one student: Can you talk to Knuth? He said: "Yes, Friday night."

Knuth: Friday night? Ok.

Before I go on to more questions I want to mention that the whole inspiration for this kind of a session came about when I was at Caltech during these 1960s and Richard Feynman, our famous physics professor, would end every one of his classes the last day of class: Anybody can ask any question they wanted to. And so I started using that in my own classes, and I kept that up until I retired. And so now I find this as a best way to customise the lecture. So I will try to give an answer to basically any question, and I'll try to keep it short, so we can move to a variety of question.

On the other hand, tomorrow I'm giving another talk at 12:30, and it's also called Questions and Answers. Tomorrow I'm gonna have a piano keyboard, so that I can answer questions if they have anything to do with music because I think when you walked in here, you've got an ad for a concert that's gonna be given on Friday. So today let's not have questions about music, but anything else is fine.

Then I also brought this with me. This is a prop for a paperback book that I published, I don't know, 2–3 years ago, which is the middle third of Volume 4B of The Art of Computer Programming. So far I've finished Volume 1, 2, 3 and 4A and I'm working on Volume 4B. I started writing it in the middle, and this came out. It's called satisfiability, and the big six on it here says Volume 6, Fascicle 6. Fascicles 0, 1, 2, 3 and 4 were part of Volume 4A.

Now the reason I'm saying this is that right now, maybe as we speak, Fascicle 5 is being printed and it will be available in bookstores a month from now, and you will love it. It's a wonderful book. Ask your parents to get it for you for Christmas. *laughter* And the main thing is that it's, I would say, maybe eighty percent of it is about puzzles or topics that are often considered not only worthwhile but fun. I sort of have been waiting all my life to present this material in showing how puzzles are very relevant to learning how to be a good computer programmer. And so that's Fascicle 5 gonna be coming out in a month. So that's end of advertisement.

Gruska: There is a question: What is your favorite unsolved problem in Computer Science?

Knuth: My favorite unsolved problem in Computer Science... Ok, well... I guess, personally, what is the worst kind of question to ask? And I'm sorry, but it's a question that starts by "what is your favorite X?". Because it's really hard to say what is my favorite X: almost always "what's my favorite algorithm?", "what's my favorite... relative? – Do I like my son better than my daughter?", and so on... So I hope not too many of the questions today are gonna be "what's my favourite X?", however, I don't wanna duck this one.

So unsolved problems in Computer Science. It depends on who and whether I think it's gonna be solved or not. So in order to be a really favorite problem, it would one where I expect that when I tell you what it is, then next week someone in the room will solve it. So often my favorite problem is one that I've just thought up. And last week I thought up a problem that I mentioned to one of Dan Král's students on Sunday night, and so I'm not gonna repeat that one now, so he can work on it and solve it for me.

Of course, the most famous unsolved problem in Computer Science in a sense that it's got a million-dollar price associated with it is a question "does P equal NP or not?" Someone is gonna ask me about that anyway, so I might as well spend one or two minutes on that. So the question is: The P is the set of all problems for which we can solve in polynomial time, and NP is a set of all problems for which we can verify a solution in polynomial time, more or less. And so the big question people say is "Does P equal NP or not?" Now, I wanna amplify that question. So let's try to be specific. It's known that the question "P equals NP?" is equivalent to saying "is there a polynomial-time algorithm to solve one particular problem?" Let me take satisfiability, for example because I held that up.

So there's a problem called 3-SAT which says: I have n clauses and each clause is of the form $(a \vee b \vee c) \wedge (d \vee e \vee f)$ and so on. Various clauses like these and all together [logical] and of these. And a, b, c, d, e and f are actually $x \vee \bar{y} \vee z$ or something like that... a, b and all of these things are Boolean variables that are either negated or not. And the question is: Can we satisfy all these clauses simultaneously? Is there a way to say that x is true, saying y is true and z is false, and all of the clauses are gonna be true? And that it's satisfiable if and only if there is a way to set these variables.

So when you look at this problem, you say: "Oh, sure, I can easily solve this problem." However, I don't wanna say if you've solved it tell me how you did it because a lot of people have told me that, but it was a waste of time. A lot of people think they've solved this problem, but they didn't.

The question "P equals NP?" says "is there an algorithm that solves 3-SAT to some constant steps?" I wanna ask another question, and that is: Could we know an algorithm that solves 3-SAT? Some people think that these are the same questions. If an algorithm exists, certainly, we would know what it is, right? However, that is a giant step to go from saying that there is an algorithm,

and there is an algorithm we actually can find. A lot of things are proved non-constructively, so it can go several ways. One is that we have "yes, there is an algorithm, yes, we know it". Or it might be "yes, there is an algorithm, but no, it actually is beyond our grasp". You can never write it down. It exists up there, but you need to be God in order to know, actually to use it. And then there is a case "well, there is no algorithm". Well, then this [the fourth cell in a schematic combinatorial table] had better be no. I'm not gonna have an N and Y in here.

Gruska: Would you be happy to have non-constructive solution?

Knuth: Well, that's what I believe is probably true. I'm not sure how happy I would be, but it could be that the total number of algorithms is huge, and if it doesn't exist, it's a completely different question. It's quite likely, but hardly anyone thinks about this puzzle. It's quite likely that the algorithm exists but only because there's only finitely many reasons why it doesn't exists. And that's not gonna tell us much. There are many cases of non-constructive things where we know, for example, that there's a winning strategy in a game of hex, but nobody has any way to actually win hex. It's just known that there is a winning strategy for the first ...

Gruska: So next question...

Knuth: Yeah... *laughter*

Gruska: Do you still write any code? If so, why and which programming language?

Knuth: In this case, I could even say what my favourite programming language is. *laughter* When I wrote Fascicle 5 I probably wrote about 600, I don't know, many hundreds of programs and every week I write on average at least five. Most of programs are rather short, of course, but some of them get to be fairly good size. The language I use all the time is called CWEB. It just really works for me. It comes installed with Linux, and so I've got many dozens of CWEB programs as examples on my website. If anybody wants more information about any of the ones that I didn't put on the website yet, I'm glad to put it online.

It's a combination of TEX and C. It's so much better than any other way to write programs, but I'd better not get started on it. To me, it's the greatest outcome of all of my work on TEX. The fact that I now have CWEB in order to solve lots of problems.

Gruska: Ok, a related question. Did you try to write program for quantum computer?

Knuth: Ahaaa! Quantum computing is something that I have absolutely no intuition for. If quantum computing turns out to be the best way to go and everybody switches over to quantum computing, in a way, that will be the best thing for me, because then I'll have more time to write my books. Because I'm never gonna write anything about quantum computing. I only promise to

write about things that were known in 1962 when I started a project [The Art of Computer Programming]...

Gruska: Another question is such that you may have a difficulty to answer. The question is: What were the research problems you worked on during your cooperation with NSA? *laughter*

Knuth: So I spent a year before going to Stanford working on code breaking, and I'm not allowed to tell my wife what I did during that year. *laughter*

Gruska: She's not here! *LAUGHTER*

Knuth: But this is being recorded, and maybe she would watch the recording.

Gruska: [Disappointedly] Oh...

My question is: What was your subject in your thesis? PhD. thesis. And did your advisor help you?

Knuth: I was lucky to work with Marshall Hall at Caltech, and so my thesis was about a general area that now would be called combinatorial designs. More specifically, finite projective plane. This is an arrangement of points and lines that are abstract. There is a parameter n, and I think it was... $n^2 + n + 1$ point, and every line contains $n + 1$ points and every point is on $n + 1$ lines and any two lines intersect at exactly one point. So this is like a projective geometry, but finite projective geometry, instead of the projective geometry of the sphere or something. So my thesis was to show that certain kinds of finite projective planes do exist and they hadn't been known before to exist. The whole area of designs is to find families of sets that have interesting properties that might be useful in application.

Now the interesting thing is that over the year since I graduated, I applied almost every branch of mathematics that I've ever heard of to computer programming except the theory of design. Because subsequently, we've found out that we can do better with random choices in almost all cases instead of trying to find these very rare patterns that sometimes exist. It was very good training, and my advisor helped me in the following way: I was working on another problem and one morning as I rode up in the elevator, got to my office and said hey, I betcha I can solve this particular problem I just heard about. And my advisor said: Ok, that's your thesis.

Gruska: Another question: Vim or Emacs? [The question wasn't answered at this moment. But see a later mention near the text "Tabs or spaces" in this transcript.]

Knuth: What do I think about Artificial Intelligence? Ok. Do you think it could be dangerous?

Certainly, I prefer real intelligence to Artificial Intelligence. I have always felt that the working ??? AI has been at the cutting edge of Computer Science. Of the last sixty years, the people working on the challenging problems of Artificial Intelligence have come up with many of the most important innovations in the field. I always regarded it as something that tells us how to stretch what we know and invent better algorithms rather than as something where I would actually use the algorithms afterwards and believe that.

In other words, suppose we develop a really good algorithm that decides whether or not a student ought to graduate. Should I let the computer decide which students graduate or should I try to understand their working and see if it has some value?

Right now, the question about the danger is extremely pressing and especially with respect to military applications. There is this really frightening movie, short video... I can't remember the name of it. Came out about three years ago. Where you could pretty much, with today's technology, you could program drones to kill anybody you wanted to. The scenario I should remember, some of you will maybe remember the title, but my friend Stu Russell at Berkeley was one of the people behind that film. Essentially it's already possible to do these horrible things, so we gotta find some way to keep that from happening. Stu has the best idea so far about how to prepare for such dangers, but still, I'm afraid his ideas are not satisfying me because they depend on assumption that human beings are rational. And the more I read about these days, the less and less I believe that human beings are rational.

It's very serious to see how to restrain the things that we don't understand and figure out how, because with the new techniques we are able to solve many problems, wonderful problems in all branches of science that we weren't able to solve before. The ones that are encouraging to me are the ones where there's no enemy involved. Like somebody trying to defeat us in the experiment, but the only enemy is that we're battling ignorance, we're trying to find some pattern in the way stars work or something like this, the way biology works, how to identify diseases of different kinds... And these machine learning techniques are wonderful. Even though we don't understand anything about how they work. But you use the same algorithm for something where there's adversarial interest involved, then everything gets bad.

Gruska: Another question was very nice, and I think you will like it: Could you tell us the story of TEX from the very beginning to implementation?

Knuth: Ah, yes. *laughter* Yes. In short, I found out that computer technology had changed so that the work that was once done by hand with hot metal was no longer being done. It was replaced by new technology which was based on photography. And the new technology looked awful. I saw the proofs for the new edition of Volume 2 of The Art of Computer Programming, and I was almost sick.

I didn't wanna have a book that looked like that. And then a couple of weeks later I learned that computers might be the answer, because somebody working in Southern California had found out that actually with digital methods, with pixel zeros and ones you could potentially make books that would look just as good as the real printed one.

So I got in an airplane and flew down to South California, talked to the people there and decided to change my sabbatical plans for the next year where I was gonna study combinatorial algorithms. Instead, I decided that oh, I can now solve the problem with the books if I only write the computer program that makes patterns of zeros and ones that say what should be on every page of the book. Well it took a little longer than a year, but that was the beginning of my work on typography.

Zlatuška: So the proofs for Volume 2. That means Volume 1 was already printed not using TEX? Does it exist?

Knuth: No. Volume 1, 2nd edition have already come out.

Zlatuška: The first edition you ??? about.

Knuth: Volume 1 would have come out looking bad at some point, but I was revising Volume 2, and so those proofs came in. What happened, is the technology went away from hot metal, basically monotype setting, and actually in Eastern Europe was the only place left for people who were still doing monotype during the 70s. So I have a translation of my book in Hungarian that as done by these hand methods still look good in the 70s, but the books that were printed... if you look at all the journals of mathematics that were printed in the 70s you see what I mean. So I was feeling bad about until I realised it was just a matter of programming. The machines were there that would make the book, but I need to get the pixels figured out. So I spent a long time in the library looking at everything that I could see about how to make good-looking books, and I brought the experts to Stanford, and we worked together on it for a while.

Gruska: How many months you worked on TEX?

Knuth: It's hard to say because I was also doing a few other things, but at one point I took a leave of absence from Stanford for a year because I found that working on software was harder than writing books. You can write books, you can write papers, but software involves much more of your brain, and I couldn't swap in and swap out so much without taking a year off of teaching and getting TEX done. Then I could go back and resume the other schedule. But in calendar time I finished this five-volume set of books called Computers in Typography. I finished that in 1984, I have started the project in 1977. So that's seven years. Then I came back to it in 1989. I came back to it because I didn't realise that people were gonna be using it for typesetting strange languages like Czech. *laughter* You know, you have accents on letters. Wow. So I went from 7-bit to 8-bits in 1989. Took a year.

Gruska: As far as I remember, when working on TEX every evening or every morning you wrote down what was good, what was bad. Did you make use of these comments?

Well, I got a paper out of it. I have a paper called "The Errors of TEX" which I think is a good idea for everybody – to keep track of what mistakes you make. Programming is too complicated, you can't get it right all. And I wanted to find out what were the kind of errors that I made and also learn something about patterns and that I could change my actions. I kept this log showing every century(???), every major non-trivial change that I made, some of the trivial ones too, to TEX and to METAFONT over the years. Then I was able to get a good understanding of the scale, how important are certain kind of errors. For example, people say: goto statements are bad. Look at my history with TEX – sure enough, I made some errors, because I used goto statements improperly. However, I had also used every other kind of statement improperly too. Every statement – assignment statements are bad, conditional statements are bad, everything from that aspect can be misused. But I did learn something about which kinds of things to avoid and the corrections were not only to fix errors but also to improve user interface and things like that.

Gruska: Next two questions are pretty ??? so please ???.

Knuth: All right so... What would you advise to your 25 years [old your]self?

So 25 year, that would be... 1963. That's the year I got my PhD. ... I decided that year to become a college professor. I actually been offered the year before when I was 24 to drop out of grad school. Essentially I was offered a salary of $100,000 a year plus an assistant. Now in 1962 that would be like $10,000,000. It's not anywhere near Bill Gates's salary, but anyway I knew that my role in life, my interest was going to be to work with students rather than to maximize the amount of income that I had. On the other hand, I'm not saying everybody should make that decision. Anyway, at 25, that's when you want to start becoming stable and making long-term decisions that you're gonna live with.

Knuth: Next question: Anonymous:

Gruska: How being a Christian affected you as computer scientist?

Knuth: It's hard for me to say what it would have been like if I had been born into a different family, but certainly, I guess, one of the ways, my Christian upbringing with respect to work on crypto and questions of computer security. I'm not very good at doing work on cryptography, because I'm not as sneaky as the people who are trying to defeat these things, but with respect to security all my life I had this idea... I'm sorry, not security, privacy... I always had the idea that everything that I do is known to God, so I don't have absolute privacy. And I didn't understand until later that there are people who think that nobody should know what their thoughts are. So it makes it harder for me to understand, but some people concern about privacy, although of course, I don't want my

thoughts to be used by the devil, by something that's going to exploit me, but I'm not very comfortable if there was something beneficent watching over what I'm doing.

All these things, I guess I should say, I'm very happy that there are parts of my life in which there's no mystery, and I can prove that something is right, but I wouldn't be happy if there was no mystery whatsoever, so I appreciate the fact that there are things that I'll never understand, so it teaches me some humility that I shouldn't expect to understand everything. So I don't claim to understand everything, and I'm very happy that God did not make it possible to prove or disprove the existence of God.

Gruska: Related to this question: Let us assume that you will live still 30 years.

Knuth: Do I? What now? ... Oh goodness. *laughter*

Gruska: By famous visionary Kurzweil at that time we should have laptop with information processing power better than all human brains. What would you do with such laptop?

Knuth: I would certainly try to finish The Art of Computer Programming. Let me rephrase your question. How do I want to continue, you know, what should I do the next week and the week after that? How do I want to continue to live? I have to be watching when am I gonna start going senile. At the moment I don't think I've reached that point yet, but it might come to the point where I should stop writing The Art of Computer Programming because I'm starting to write stupid stuff.

Gruska: In which area of Computer Science or mathematics do you see the most potential?

Knuth: In which area... This is one of these favourite questions again. The much harder question would be: "Which area does not have much potential?" because I think everywhere I look, I see potential. The problem is really have to go to sleep at night not using the knowledge we have. Everywhere I look, I see that it's not saturated yet.

Gruska: Additional question is pretty philosophic: What your opinion on Curry-Lambek correspondence? Do you think mathematics is constructed or exists independently?

Knuth: I guess I'm a Platonist in the sense that I'm discovering things that are there already. The stuff that's there is all consistent with my own attitude that these truths are there and I just am learning a few of them at a time. There might be an algorithm that solves 3-SAT, and that would be there. In fact, I'm not sure I even understand how I could be a non-Platonist.

Gruska: What was the most valuable ??? you learned during your career?

Knuth: Not to use email? *laughter* There we go.

Gruska: Is too late for 20 years old students to start learning real mathematics and programming?

Knuth: A 20-year old student? My goodness, no. I didn't now that much math when I was 20. It depends on what you've seen so far and the teachers you've had. But I consider life to be a binary search where you find out things that are relatively easy for you, because of the unique experiences that you've had, and you try some things, and some things work, and some things don't work. Then you keep learning more about yourself. I'm 81 years old now, I'm still not sure exactly where to go, but I keep trying different thing. I have given up on quantum computing, though. *laughter*. I tried to understand quantum computing, and I know people who do understand quantum computing, but I ??? it's not me.

Gruska: Do you believe in non-locality in physics?

Knuth: I understand a few things about multiverse and things like that. For example, there's no way to distinguish between whether or not this lecture I'm giving now is simultaneously forking into many different things and so each different incarnation of it will have a different series of questions, and I'll give different answers to different questions and so on. And all of this idea that there are these all the different universes all simultaneously existing is consistent with quantum mechanics.

Gruska: Biggest challenge of becoming a good programmer?

Knuth: What does it say? Spaces... Tabs or spaces? *hehehe laughter*

So I use Emacs for my hacking, and I always untabify ??? but I also read a lot of programs that other people have written, and I start out with changing all the tabs to spaces. Of course, I'm using CWEB, not Python.

Knuth: Next... Some of the exercises are known to be open research problems. Yes indeed. If the number is 46 or higher, it's something where, as far as I know, it hasn't been solved yet.

Knuth: Has anyone ever contacted you that they have solved one of them while reading the book? Yeah. People look at them ??? I do want to point out that a lot of people haven't been looking at those lately, so ... I'm sorry...

Gruska: Excuse me... We make break for 5 minutes because they need to change [the microphone].

Knuth: So we had a celebration at Berkeley, I guess a month ago, celebrating the life of Dick Karp and at that time there were a dozen speakers, and I decided I would say something about the Karp's work that the others weren't going to talk about. So I looked again at Volume 3 of The Art of Computer Programming, where Dick had told me about some things he never published that applied to sorting on tapes. Nowadays people don't use tapes for sorting, but when Volume 3 came out, this was one of the biggest topics in all of Computer Science. People were saying one third of all computer time was spent on sorting, and people had

these tapes, and these were must have to have in the book. But since we don't do sorting that way anymore, I should rip out all this, I don't know, sixty pages or something of Volume 3. And then people say "no, no, don't take it away, because we're just finding out there's a new memory kind of being invented which is rather similar to tape and so these old techniques are gonna be good!"

Anyway one of the things about magnetic tape is that you can read it forward and backwards, so you can write information on the tape, and then you can read it in the other direction, and that would be much faster than rewinding and reading back forward. And Dick Karp worked out a beautiful theory about patterns of using tapes for sorting that he showed me at lunch one day and I put it in my book, and he never published it. So I showed it to the people at Berkeley, and as I was looking through this I came across a research problem, you know the level 46, which it seems to me is ripe for solution now. 50 years have gone by, and people know a lot more math than then, so I suspect that there are dozen problems in there that are just waiting to be solved. So you can go through, you have to spend a little time paging through and checking out the numbers. And if it's 46 or more then think about it and say "hm, I wonder if I can solve this now", because people haven't been doing that systematically.

Zlatuška: How many girls you seduced because of Computer Science?

Knuth: How many girls have I had thanks to Computer Science?

Gruska: Forget it. Forget it.

Knuth: So, in fact, I had 28 grad students, but none of them were female, *laughter* but I did serve on many committees and many others and so on...

Gruska: Here is better question: What's the biggest motivation that kept you going through career?

Knuth: I guess it's the example of my parents which was always to somehow be a servant, to see how I could be of use to somebody else. My father's name was Erwin. He had an informal, I guess you'd call it a startup ???, he is a one-person operation, and he would do services for all churches and schools and a few other nonprofit things like this, and he called it ERW service. He thought it was clever he could write the word service but make ERW large, so would say ERW service. But anyway, that epitomized all the philosophy that I grew up with – to be a service. If I got to a point where I thought that I couldn't be of use to anybody anymore, I told my children not to keep me alive just to make me happy, but if I get to a point where I don't recognize them or anything like this... How do I express this... There's a novel, came out less than 20 years ago by P. D. James and it was about... the world... the people discover that from now on all women in the whole world would be infertile and so there will be no more children ever born again. So humanity was eventually going to die out. So what did people do? That was very devastating to me, to imagine what it would be like if I was

in a situation where I could not do something that would be of use to people, more than my immediate family, but to people in the future.

Well there's a short story by... Oh, goodness... Okay, who's the greatest Argentinan author?

Zlatuška: Borges.

Knuth: Borges, yes, of course, Borges. So he has this short story. I think it even takes place maybe in Czechoslovakia, I don't know. But anyway, it's a story of a person who is a playwright, but he's facing a firing squad, because of his political views. And just as the guns are aimed at him and so on, he prays to God, and he says: "God, I have to finish this play I'm working on. Please, make time stand still so that I can work out all the details and figure out exactly what should happen in this play." So God says okay and time stands still, and this playwright solves the problem, he figures out exactly what's the perfect play. And then he's shot. So nobody ever gets to see what happened in the play. It was just that the playwright himself was able to solve that particular problem. So that's the opposite of my own way to do it. It has to be something that I do that somebody can use. I hope that's making myself a little clearer what this idea of service is.

Gruska: There are quite a few people that try to put the idea of formal method into programming, software development... What do you think about that?

Knuth: When you're putting ideas into formal methods, it forces you to understand what the ideas are. You don't realize what you don't know until you try to formalize it some way. So it's a great educational experience. On the other hand, I guess I said a similar thing about Artificial Intelligence a while ago that I consider it as a very useful way, a source of good problems and continuing to learn. But then I was less interested in actually using the programs afterwards because I knew that they would only be approximately right.

I once had a language called SOL, Simulation Oriented Language, and the idea was that it would be it would be pretty easy to write models for discrete systems. And you could simulate the system. As I was working on this language, I looked at a lot of different applications, and in each case, I found out that the idea of formalizing the application and putting it into this language was a wonderful educational experience. I can build models of things to simulate, but I learned much more writing the model than I did actually running the model afterwards. So I almost thought I should take output statements out of the language. The people only use language for formalizing their model instead of for actually running it and believing the answers that they get afterwards.

Nobody seems to understand what I'm saying, but anyway, I believe the main advantage of formalism is educational rather than actually having a payoff afterwards.

Gruska: Next question: For how long can you program or read papers per day until you are exhausted? How do you relax? Do you relax sometimes? *laughter*

Knuth: Yes, but in fact, I'm relaxing right now. *laughter* At night, I have to take my mind off whatever I'm doing. So I read James Bond or something not really heavy literature. I'm reading right now a novel by Frederick Forsyth called The Odessa File which is story about the German SS in Latvia... what you'd call a thriller. Some kind of story like that and I go right to sleep. I read novels at the same speed as I read a math paper. *laughter* I don't know any other way to do it, so I don't get through that many books. However, when I do come across a book, that I think is especially nice, I put it on my website. So I think if you look on my website and under... I think, there's a Frequently Asked Questions and it says "so you're retired" or something like this. You click on that, and it'll tell you, I think, maybe three dozen books that I think were special to me.

Gruska: Did you work hard when you were student?

Knuth: I was a machine. I was a problem-solving machine. Somebody said: Okay, Don, do this, and I would do it. And I didn't start reading for pleasure probably until I was about thirty years old. I was given an assignment, and I was a good boy. So I did it.

At least that's the way I remember how it was. I don't know how it really was. I think I also was a... they might call me a wise guy. I mean, I was cracking jokes and not paying too much attention to what I was supposed to do.

Gruska: Did you watch westerns or detective stories?

Knuth: What about detective stories?

Gruska: Did you watch westerns movie or detective stories?

Knuth: I certainly watch a lot of movies, but, by the way, I might as well mention one thing. I guess it was a month ago when I saw again a movie. It's called Double Indemnity. It came out in 1944 or something like that. It's one of the earliest noir movies, and it stars Fred MacMurray and Edward G. Robinson and Barbara Stanwyck. That movie has special significance for me because when I was writing The Art of Computer Programming, that movie was playing almost every night on The Late Show. As I was typing The Art of Computer Programming, I would have the TV on, and I would see Double Indemnity over and over again. So The Art of Computer Programming was written largely to this movie Double Indemnity. The music to Double Indemnity was written by Miklós Rózsa, and it's haunting music that I hadn't remembered how haunting it was until about ten years ago. I saw Double Indemnity again at a theater and immediately when they showed the title, and I heard this music, and I said: "Oh my god, what powerful music is it." So I've included music from Double Indemnity in my piece that's gonna be played on Friday. Although I'm not sure if I'm gonna be sued for this. But one of the things that occurs, you can check it out by watching the movie.

Gruska: Didn't you have idea to write science fiction?

Knuth: I talked about writing it or? I enjoy different kinds of science fiction, of course, but I haven't had time to... I wrote this little book called Surreal Numbers. I kind of think of it as a little bit like opera in the sense that opera is good music to a little bit of a plot. My book Surreal Numbers is good mathematics to a little bit of a plot. The characters in this story work on a math problem together, and it makes a little plot, but mainly there's beautiful mathematics that they're discussing.

If I live long enough and finish The Art of Computer Programming, I'd like to write some science fiction. One book I'd like to write is where the story is told by ant colony. Not by an ant, but by ant colony. There are tens of thousands of ants, and somehow they cooperate with each other and so that they form a consciousness and so I think there ought to be a short novel that's told by an ant colony.

The other thing is a short story something like... you could call it The Fly. Now, Mark Twain in one of his books, I think it's called The Mysterious Stranger, there's a character in there representing the devil, and early on in the book the devil opens a window, and a fly goes out the window. And he said: "Because I let this fly out the window, there's gonna be war next year." He's predicting what they call the butterfly effect of chaos theory. All kind of things are codependent.

So maybe people have seen this movie Run Lola Run that came out of Germany a few years ago. It tells a story, three or four times the story starts out exactly the same way, but then there are three or four different, completely different endings, just because of little changes in time. So I think it would be fun to write stories that... Well, it's not one story, but it's two or three stories that all start out the same, but end completely differently. But on the other hand, did this movie, it's German name is Lola rennt. Run Lola Run in English. So the author of that movie already did it, what I was planning some time to do.

Gruska: So, please, ask questions. Yes, will be finishing. Maybe those who don't have mobile with, they should have the chance also to ask questions.

Knuth: Yes!

student: You have a trouble ??? distractions or maintaining focus when you're working on a problem during the day? And if you do, how do you sit down and focus and do some piece of work during the day? You have like a ???.

Knuth: If I understand, you're saying how can I focus on one thing instead of many others? I don't have all the distractions of modern day. I don't have the radio play, music blaring ???. I found that, for example, if my job for the day is to do something like proofreading, then it helps me to have some kind of music like Telemann or something playing in the background. It sharpens my mind. But if I have Bach playing the background, it's too much, and everything goes away. So part of it is having no distractions, no interruption.

When my children were babies, they would be crying. I had this problem: How am I gonna concentrate on writing a book when the baby is screaming? The answer was white noise. I had a waterfall, and I could turn the fountain on and crying plus white noise equal white noise. *laughter*

But in general to concentrate on one thing I collect a lot of material, all on the same subject. I'll read thirty papers about that subject all at once, instead of reading about many different things. So I spent a lot of my time filing things away to be read later. Batch processing is very important.

Sochor: My question is: Do you prefer screen and keyboard or paper and pencil?

Knuth: I love that question because it turns out that I learned when I was in college that I could write a letter home to my parents faster with pencil and paper than on a typewriter. The reason was, actually, because I'm a good typist, I type faster than I think. *laughter* I had actually gone to typing school, so I can type so fast that it's a synchronization problem. *laughter* I'm not ready for it. But pencil and paper for me is a perfect sync with my thought process. So I make the first draft of everything in pencil paper, but then I go to the computer, and I polish it, and I edit for style. And I'm typing on it ??? I do that.

Gruska: So, you don't have to think, you just looked on your fingers.

Knuth: That doesn't work for me.

student: So my question is: How has your stay in Czechia been so far?

Knuth: In fact, I wanted to say at the very beginning that it's a thrill for me to be invited here to celebrate your 25th anniversary. It couldn't have been better in every way, and I'm really enjoying this week. I was babbling over with enthusiasm for that, but I forgot about it at the beginning.

Of course, I've only had five/six days so far, but each one has been a joy.

Gruska: Ok, I think it's time to make break. The break will be quite long. Continuation will be tomorrow. Thank you, professor Knuth.

applause

Sojka: Come tomorrow and those who want to make a photo with Don and the faculty, the photography ...

Questions and Answers Session 2: Boundless Interests

organ music playing

Zlatuška: Ok, ladies and gentlemen, dear colleagues, I would like to welcome you at the second session of Questions and Answers with professor Donald Knuth. Today, again, there is a possibility to put questions over your phones or whatever connection this works (???). And today's topics are not limited to just Computer Science, but also to music.

We invited Don for 25th anniversary of Faculty of Informatics. Not only as the first honorary doctor of informatics of this university and this faculty, but also as personality who transcends boundaries, and especially we felt that it would be interesting to have him here also as a musician.

I believe he's got at home not only his work office devoted to The Art of Computer Programming and informatics, but also to music. He built his own organ in his house at Stanford, so this other Donald Knuth, I believe, will be for many people as interesting as Donald Knuth, the author of The Art of Computer Programming. If you look at that from other perspective, I feel that the idea of programming as an art—Don quoted that yesterday—is something which resonates with this second personality, but I hope that there will be also questions concerning this part.

Now. I thought Donald travelled with this, but he travelled so light, but it would be impossible so... The floor is yours with the questions. Any questions from the floor?

Knuth: Right, so, hello, everybody. I want to thank again for the wonderful hospitality this week, and before I forget, I also want to mention that there's another lecture today by Dana Scott. I think 4:30 in the afternoon. So, you know, if I give a bad lecture, you at least get one good one today. *Zlatuška's laughter*

The other thing I wanted... Before we start with new questions, I had a couple things to say. First of all, I don't know if you've ever given a lecture, but the night after it you wake up in the middle of night saying "Oh, I should have said that!" and so I thought of at least two things that I wish I had said yesterday.

So in the first place, I was trying to explain why the question about P versus NP is not as simple as people usually think when they talk about whether there exists an algorithm that runs in polynomial time, and they think it's the same as saying that we could know an algorithm that runs in polynomial time. But there's a big jump between that, and then someone asked me later on about the Artificial Intelligence. So it occurred to me during the middle of the night that one way to think about it is this:

I want to give an example where maybe there exists an algorithm to decide the 3-SAT problem in polynomial time. But we won't know what the algorithm is. So imagine that we have a problem of size n, and we build, let's say, $(n^{1,000,000})$-state, some kind of a neural network that has $n^{1,000,000}$ neurons. So this is a polynomial number of things. Just add a few more [zeroes to the exponent]. So now we trained this neural network on lots and lots of SAT problems. You know, we've got great advances now in machine learning theory. Now I feed it a new SAT problem, and how do I know that it's not gonna solve all of them? In fact, in order to prove that, we couldn't possibly train this neural network. In order to show that P is unequal NP, we would have to show that there's no way to train this neural network to do it.

And that's a situation that we're faced with. Right now, when we do train neural networks, we've got something that solves the problem, but we have no idea what the network is doing inside. And that might be a way to think about the difference between an algorithm existing and an algorithm that we know what it is. Ok. Do you have a comment on that?

Zlatuška: If I may.

Knuth: Yeah.

Zlatuška: Could you extend this problem to the problem of human mind? The human mind and limits of human mind?

Knuth: Okay.

Zlatuška: Because that *laughter* obviously... That's obviously similar problem.

Knuth: Yeah.

Zlatuška: So what's your idea about limits, the capacity of human mind? You know, the ideas like mysterianism, and the existence of problems which are inaccessible to human mind?

Knuth: In that case, we actually have rigorous proofs, because there are incompressibility theorems. So we know that there are things that cannot be made small, and so the human mind can't possibly understand something that has more states in it then there are, well, let's say, protons in the universe or something like this.

Zlatuška: So, something between man and the God.

Knuth: Yeah. But in one of my papers, I have a number that I called... I don't know, I forget... I shall call it Super K, which is also the name of a breakfast cereal in America, but anyway, I needed a special font in order to do it. Like I wanted to print it only in color, but if you look with a magnifying glass at the paper where I talk about Super K...

Anyway, this was a number that was, I forget, it was something like $10 \uparrow\uparrow\uparrow\uparrow 3$, which is defined in terms of the [Knuth] arrow notation. Anyway, I was once giving a talk at Livermore Lab where they were trying to impress me by how big their equipment is, so I said: "Well, here, let me show you some big numbers that are bigger than you ever thought of before." When you try to think—what is this number Super K—this simply means *writing on blackboard* that you take ten... let's see... quadruple arrow I don't even remember the thing... Anyway, $10 \uparrow\uparrow 10$ is equal to $10 \uparrow 10 \uparrow 10 \uparrow \ldots$ and then if you want to go to triple arrow, then you simply do this that many times. Pretty soon, it gets mind-boggling.

But still, this number is finite, and almost all numbers are bigger than this. *laughter* You get a little idea that even though computer scientists stick to studying finite numbers, we aren't limiting ourselves too much. There's still a lot of interesting stuff down in... I don't necessarily insist on immortality, I would settle for living this long. *laughter*

Zlatuška: And if I may misuse the moderator's role, I would like to ask about music a bit. Within Fantasia Apocalyptica, you have lots of quotations from... or sort of inspirations from other authors. And what seems to me really fantastic is the scope. You go from Gregorian chants to authors whose music is sort of dramatic, like Olivier Messiaen. And incidentally to see quotations from Olivier Messiaen and Bruce Springsteen at the same time, well, both of them are favorites of mine, but what seems to me interesting, just to combine those absolutely different styles which are usually considered incompatible. How did you solve this compatibility problem?

Knuth: So come on Friday, but what's your opinion of rap music? *Zlatuška laughs* Because there's a quotation from Eminem as well. *laughter* And of course Dvořák. There are a few notes that are actually original with me as well. I wanted to mention that now before I forget because it ties in with Brno. When I came 24 years ago, my first visit to this part of the world, as my plane was approaching the airport in Prague, I woke up that morning with a melody in my head. And I wrote it down. Probably in the airline magazine or something, but anyway, I wrote it down, and I kept that piece of paper, and I saved it. I sort of thought of it as a Prague theme. Because really, I woke up and here was this melody and I wanted to write it down before I forgot it.

So fast forward twenty more years, and I'm writing Fantasia Apocalyptica, and I came to a point of the piece where I'm trying to represent Chapter 21... I guess I gotta go back. The point of Apocalyptica refers to Apocalypse, the last book of the Bible. I'm trying something new in this piece, which I don't think had been done before. To make a fairly literal translation of an original Greek text and convert it into musical equivalent. So I find more than hundred fifty motifs for things that are repeatedly used in the Greek text. The same sequence of motifs occurs in my piece...

Zlatuška: You consider yourself Wagnerian?

Knuth: Yes, Wagner had motifs, but he never told anybody what they were. So the differences is...

Zlatuška: He wasn't university teacher, so...

Knuth: For example, the motif for war is staccato. The motif for angels is an arpeggio. *keyboard playing* The motif for God is a three-note melody. *keyboard playing* If you're referring to the first person of the Trinity, you emphasize the first note. *keyboard playing* If you're emphasizing the second person of the Trinity. *keyboard playing* And guess what the third person of the Trinity. *keyboard playing* And the motif for the devil, in the book of Revelation, there's also a [sic] Anti-Trinity. And so this is *keyboard playing* the devil. And there's the first person *keyboard playing* and the second. The Book of Revelation has about ten thousand words in Greek, almost exactly, and I didn't just go word by word because I want it to be good music.

I use this as the constraints to say what kind of good music is suggested by this pattern of motifs. And when I get to Chapter 21, the story talks about the New Jerusalem. Here's a Golden City coming out of the sky, and that's where I got to use my Prague theme. I'm not sure which... I'll just play it instead of trying to find the absolute best combination of voices. So it goes like this: *keyboard playing* So, da-da-da da-da-da da-da-da da. That's what I wrote down in the airplane. Then it goes on a few more bars, it uses some of the chord progressions of Dvořák's New World Symphony. And New World Symphony is like New Jerusalem. So this is a little bit of music here, that...

Zlatuška: So this is actually what we loose for not having international airport in Brno. Because if Don landed in Brno, that could have been Brno theme.

Knuth: So if I had never been invited here in the first place, who knows what would have happened. One other footnote to yesterday and that is: There was a question where I referred to Stuart Russell. Now I can give you the definite reference because I recommended that you watch this video. I think it's five minutes long, and it's easy to find. So Stuart Russell... *writing on blackboard* And there's been a lot of follow-up on it, but this came out not quite two years ago, and it's called Slaughterbots. It's sort of a mind-changing video that I recommend everybody to look at. It's very important, I think, that we should ban autonomous weapons, and there are thousands of computer scientists who have signed declarations and are actively trying to make sure that the dangers are minimized.

So that was end of footnotes to yesterday, now I'm ready for new question.

Zlatuška: So I guess... Have you ever burned out?

Knuth: There was a period about 1990, where I was feeling kind of low and I actually I got a little worried about it, because one of my uncle's had gone through a period where he was, I don't know, not getting along with anybody else in the family and so. So when he got old, and I said: "Oh-oh! Maybe I'm inheriting some mental problem." Anyway, so I went to see a psychiatrist, and he said: "Have you ever heard of a Type A personality?" He was a great doctor, he showed me his textbooks. Instead of telling me, sort of preaching to me, he said: "Here, look. Here's a book I was giving you. If you look in this chapter, you'll see something that describes you to a T (???)."

I learned from that how to reduce the stress, and so it was 1990. So how old am I? 50... 52 years old. And I realized why physical education was a required subject in college. I had never done much exercise, so I started swimming in 1990, and very quickly I was happy again.

But there was a time when... Well, I work sort of 24/7, in some sense, but it's fun. Mostly. I have to psych myself up in the morning, and once I get into something, then it's hard to stop again, so that's why I have to turn off like somebody mentioned yesterday. ... Well, how do I describe my daily schedule?

I have a very peculiar scheduling algorithm. You wake up in the morning, and you have to decide what you're gonna do next. The algorithm that I finally decided to work the best is that I always do what I hate the most. Of all the things that I have no good excuse for not doing, which one would I rather not do. And that's what I choose. So all the time I'm working on something I don't wanna do, in a sense.

On the other hand, at the end of the week, I've got stuff out the door. And I'd have to do those unpleasant things anyway, so as long as there's no good reason to procrastinate anymore, that's what I choose to do. Somehow that turns out to be a better scheduling algorithm than the other way, where, "what is the most fun all the time?"

I noticed that my mood, when I start feeling bad, is really if several weeks have gone by, and I've never had a time to do anything creative. There's something, there's some urge that says: "Prove a theorem or something!" So if there's too much busywork, I can stand that for a little while, but after three weeks I have to try to do something new. I can't explain why that is. But that seems to be true.

Zlatuška: I would just add to that Stuart Russell, for anybody interested: There is a new book, which is Human Compatible by him and that was published yesterday.

Knuth: Oh, yesterday! *smiling* ... Human...

Zlatuška: ... compatible. Staying... Artificial Intelligence and the problem of control. And that looks pretty much as this topic.

Knuth: Very good.

Zlatuška: Tabs or spaces?

Knuth: Tabs or spaces? *smiling* You're going back to that question again? *Knuth laughs*

Zlatuška: Oh! Okay, it was... ???

Knuth: No, I have a question: Why does [sic] people ask about tabs or spaces? To me, when I make somebody else's code in order to read it, [and] it comes in with tabs, it confuses TEX. The Tab prints as a letter gamma, so I have to go through it, get rid of all the Tabs, and change to space, and then I can print the file.

Zlatuška: That's a proper answer, after which nobody would ever use Tabs.

Knuth: *laughing* Ok.

Zlatuška: Ok. Biggest challenge of becoming a good programmer?

Knuth: Biggest challenge of becoming a good programmer? Well, being born a geek. I know, some people think that it's just a matter of motivation and trying harder and keep educating, read and write right books and so on. Well, that might be true, but my experience is differently.

My experience is that I know many many intelligent people who are, no matter how motivated they are, I don't think they'll ever be a good programmer, and conversely, I know that I'm never gonna be good as a programmer of a quantum computer. There's something about the way my brain works that gives me a great intuition for the classical computer, but not for quantum computing. And it's not a matter of good or bad or trying or harder or not being motivated. It's a quirk that I happen to [have] been born with one of these peculiarity [sic].

Zlatuška: What's your idea in this context about sort of mandatory courses of programming within elementary school? If it is true that not everybody can be a programmer.

Knuth: No, no! You left out the word "good". *laughter*

Zlatuška: Ah, ok. *laughter*

Knuth: I think people can become a programmer [sic], but dogs can walk on their hind legs.

Zlatuška: So you feel that there's not enough of bad code.

Knuth: What I'm saying is: If you somehow realize that you've been born a geek, then you owe it to the world to you use that talent because the world needs this talent. That's the way I look at it.

Zlatuška: Vim or Emacs... That was also yesterday. I am not quite sure...

Knuth: Vim or Emacs. Yeah, yeah, ok. I just did admit to using Emacs, but I learned how to use vi enough to know how to quit. *laughter* That was the hardest first lesson. I mean, I had to go Control-Q or something, I don't know.

Zlatuška: Do you think that we are living in a computer simulation?

Knuth: Do I? *smiling*

Zlatuška: Is your god a programmer?

Knuth: It's hard to tell. *laughter* So whether I think so or not doesn't matter. The whole question about how far Artificial Intelligence could go: It certainly could go to the point where it has decided that we should have this gathering today.

Zlatuška: There's one consequence of living in computer simulation because that means that the world would be only processes which are computable.

Knuth: And finite, yeah. So, for example, the game of life can simulate any such thing. You have any universal scheme, then it would represent the lecture I'm giving now. As one very special case, it would represent Stuart Russell's book, etc. It would represent all discordant ways to write the Fantasia Apocalyptica.

Zlatuška: What was the worst code you have ever seen?

Knuth: What was the worst code I have ever seen? Hey, I love this! *smiling*

Zlatuška: All you down to ??? level.

Knuth: No, no. I had this student in the 70s who was not born a geek *laughter*, but he came to Stanford from, I think, West Point. Anyway, he was trained as a soldier, and he would follow orders. He was not a Ph.D. student; he was master student. But he did a master's project which was to automate the system that we had in the Computer Science department for sharing technical reports with other universities. So we had to make subscriptions to others, and then if they send us their reports, we send them our reports gratis. We had this large database, sort of... data processing problem. So we needed somebody to do that, and I gave that to him as a master project. He wrote a system that was going to handle the technical parts. I saw the thing, I was busy at the time, and it looked like the right number of pages and so on. And it had a sample where he had taken a couple of... a small database with a few reports, and it gave the right report, so I gave him A on it, and he got his degree. Well, that was in June.

Then in July, the secretary called me: "So Don, we're having a little problem using this system." So I went, and that was one of my first trips to the Stanford AI Lab, where they had special computers there. I started looking at the code that he had written. I got to page three or page four, and it was an example of shellsort, a sorting routine, but it was implemented... It was the first time I had seen a program where I could change one character in the program and make it run hundred times faster. *laughter* And the thing was, the variable should have stepped by H instead of by one. So he wrote shellsort so that it was just a plain old insertion sort. And clearly, he didn't understand that. So I made a copy of that page: "Hey, I gotta show this to my students next ???!" Then I turned to the next page, and he did a binary search on that.

So every page I turned to, I saw a new kind of programming error that I had never seen before. *laughter* And finally I looked at the whole way he had organized the thing. It's really hard to explain, but the text editor that we used in that day—this was way before Emacs—it would start out with an index page, so text editor would always prepare automatically a table of contents at the beginning. He had assumed that text editor would always put all of the whole database into alphabetical order in just the way that you could read it, knowing that there would be line breaks or anything like this. So it was impossible to fix the program. You couldn't possibly write a program that was assuming that you were gonna use the text editor's database for table of contents to do the data processing. So that wins my vote.

Zlatuška: Well, given the fact that programming is actually about giving orders, and maybe he studied military academy. That means some privates should never be given the ability to issue orders.

Do you have any experience with psychedelic drugs, like Richard Feynman? Altered state of consciousness. LSD, psilocybin.

Knuth: No, I'm glad that I didn't, nor did my kids. As far as I know. Near-death experiences? No.

Zlatuška: Any problems you gave up on trying to solve?

Knuth: Any problems you gave up on trying to solve, yes. Many times, In fact, I certainly thought that I had proved that P was unequal NP rather earlier. And after I had solved it, then I wrote it up, and finally, everything disappeared, just about as I was writing the last line of the paper. I realized that it was hopeless.

I can remember the first time I actually solved a problem that I had given up on, which was a kind of a breakthrough. As a student, I had tried various things, and I sort of had a "give it five days and, if you haven't got any ideas, then let it go." But once, I let it go, and the next morning, I woke up, and I said: "Wait, what if I tried this." When you do work on a problem and fail, there's one trick you can try, and that is: Imagine somebody sent you email or letter or knocked on your door, and said "So ??? I just solved that problem." And then use: "Oh, I bet I know how he did it." Somehow, if you think the problem can be solved, sometimes it helps you solve it.

But the one that was most dramatic for me, I guess, years ago, when I started, about one third of Computer Science was a study of programming languages, and now my collected papers collected in eight volumes, and one of those volumes is all the papers that I wrote about programming languages. There's a journal called SIGACT News that has book reviews, and they have a page in there saying: "Here are books that we've received. Would you like to review them?" And this collection of my papers on programming languages was on that list for a couple years. Nobody wanted to review it. So nobody cares about this although this the hottest thing in Computer Science when I started.

I worked on a problem that was called parentheses languages. So I think everybody still knows what a context-free grammar is. A way of defining a language in terms of productions. But some context-free grammars have the property like nested parentheses. So if you look at the language, half of the characters are like left delimiters, and half of the characters are like right delimiters. And every string in the language has nested left and right delimiters, properly nested. So the question is: "Somebody gives you a grammar. Is the language that it generates a parenthesis language?"

The grammar won't show any matching between these, but can you somehow look at the grammar and figure out yes or no? Is it really gonna be possible to do this? That was an open problem, and I worked several weeks on it and finally gave up. But then, I think a week later, the key came to me how to solve it. And so, at that point, I was quite delighted to have the solution. I was able to use the insights I got from that when I was doing other work later on, but as far as I know, nobody has ever read that paper.

Zlatuška: Do you solve many programming problems or create music when you sleep?

Knuth: Do I... solve many programming problems, create music when I sleep? So, no. *laughter*

With respect to the Fantasia, it was kind of interesting that after I started working on it, I had the feeling that the music had already been written, and I just had to listen for it. I don't know, out of body experience... it's like I was channeling in a way, that it was there. It did feel that there was a muse helping somehow.

Now, with respect to programming problems, it goes the other way. If I'm trying to figure out how to solve a programming problem, I can't go to sleep. So I need somehow to either solve it or forget it. But when working on a difficult problem, I usually fill up sheets and sheets of scrap paper, and so I can't keep it all in my head, but I have to write down a whole bunch of stuff in order to get it into my brain. But when it comes to the point, when I can actually think about that problem when I'm swimming, then I know I'm about ready to solve. My brain has absorbed enough so that I've learned how to go from baby steps to giant steps in this territory, the problem domain. So there is this mysterious thing that takes place once I'm ready to do it.

So I always say to my grad students... They're working on the thesis, and I realized early on that was a good idea that they should keep. Every week when we would meet, they would write down in detail what they had been thinking about that week and what was on, what do we know, what don't we know. Then after the problem is solved, you can look at that, and I can use that to prove to them that they solved a hard problem. Because once you solve the hard problem, you think it was obvious, I didn't do anything. But once you see how many hurdles you actually got through... This was good.

Zlatuška: Also yesterday you mentioned that you are Platonist with respect to mathematics.

Knuth: Yes.

Zlatuška: Are you Platonist with respect to music? Music is up there and...

Knuth: The music of the spheres. There's a question. Why is it that some music I can hear ten times and next time I hear, it's just as if I never heard it at all.

Zlatuška: This happens to some students with mathematics. *laughter*

Knuth: With mathematics as well, yeah.

But, for example, in the old days, when you have CD, I mean long-play records, they would always have to add to the piece you wanted, they had to add something else to fill it out. So I have something by Brahms, and then the publisher added a piece by somebody named Bax. B-A-X. I've heard the piece by Bax as many times as I've heard the piece by Brahms because it's hard for me to get up and turn off the record player. However, I'll never recognize the piece

by Bax the next time I hear it. So there's something different about Brahms's music and Bax's music, and I haven't been able to reverse-engineer that at all.

My piece [Fantasia] has parts of it that involve more less random elements. And the question was, well, if you just have random music, does the human brain somehow learn to do it? Well, it didn't work with Bax's music, but there are parts of it where I based on Morse code. I had to make a decision how to spell some Greek words. It turns out that in Ancient Greek, they had a notation for absolute pitch, so you could spell a Greek word just by playing those notes. Let's see if I can find... There's a place in Chapter 2 where the name Jezebel comes out. So Jezebel in Greek is iota, epsilon, zeta, and so on. I think it comes in here... Yeah. *music playing* That's Jezebel. It's kind of ???. Now Jezebel was a prophetess, and the motif for prophet is contrary motion. And so after I play Jezebel, then I play it contrary motion. But anyway, that's random. Still, our brain gets it in, we find ourselves humming. Jezebel, even though there was no reason why we should be able to remember that melody.

So I'm not sure to what extent you can take random elements, and they actually become warm and somehow have a personality. On the other hand, if you have no randomness whatsoever—musicians found long ago—that if you go straight one two three four, one two three four, exactly right, it loses life. You have to go a little bit before the beat, a little bit after the beat in order for music to come to life. And I tried the same experiment with font design. So instead of drawing a letter precisely, I would wiggle some of the points a little bit, and then the alphabet seemed to have a personality.

Zlatuška: Software patents. Software patents, can you elaborate your stance regarding software patents. Patenting software. The second from bottom.

Knuth: Second from bottom? Stances regarding software patterns.

Zlatuška: Patents. Intellectual property.

Knuth: Patents! Ok, aha. Software patterns is another thing, okay software patents, right.

I think people deserve protection for their ideas, but not if just the ideas are trivial. So a great number of software patents were something that we would expect any student to do on an exam, but a lawyer—a patent lawyer not being a geek—wouldn't know how to distinguish those. And so there was a time when people went and tried pro bono make a patent on every trivial idea. So that people wouldn't be able to make us pay them every time they use this trivial idea.

When I wrote TEX, I didn't need to get permission for any of the ideas that I use, the trivial ideas that I used in TEX. But after intellectual property rights got more and more complicated, it might very well have been impossible to write TEX twenty years later.

So when it comes to a substantial piece of software, like the undoing mechanism in Photoshop or something like this, I think this really deserves patent protection for a limited amount of time, but not in perpetuity. That's my general feeling about patents in a nutshell.

Zlatuška: What's your favorite music band?

Knuth: My favorite music... band? *laughter* I remember The Beatles. But lot of the music after that sounds like noise to me.

Zlatuška: Some of the exercises in The Art of Computer Programming are known to be open research problems. Has anyone ever contacted you that they have solved one of them while reading the book?

Knuth: Yes, I think I mentioned that yesterday, but it's not that uncommon.

Zlatuška: And you don't pay actually...

Knuth: No, no, I only pay...

Zlatuška: It became closed problem, so it was an error...

Knuth: It's stated there that if you find a better answer, you don't get money, you get glory instead, and so instead I mention your name. But if you correct an error, I silently correct it as if I had known it.

Zlatuška: What would you like to redo?

Knuth: What would I like to redo if I could? I would base TEX on decimal instead of binary at the lowest level. TEX is based—at the lowest level—on a scaled point, of which there are 2^{16} scale point to weigh a point. So internally, everything is kept in binary but is communicated to the user in decimal. So the user doesn't get to see... This leads to strange results. You know, you say one third, and then you multiply it by three, and you get 0.999 instead of one. So that would have been better to do that.

Zlatuška: Do you still use TEX often?

Knuth: *giggles* Well, let's see... I haven't used it since last Friday because I've been in other town.

Here's a... Oh, I see. You're not raising your hand, you're raising the camera. Ok. But other people out here who...

Zlatuška: Anybody from the audience?

someone: Can you play something?

Zlatuška: Can you play something?

someone: Like a longer piece.

Zlatuška: Longer...

someone: Anything you like.

Knuth: Well, I only have this music here. So, choose a random chapter.

someone: Uh... Seven? *laughter*

Knuth: Seven. Ok, so seven is the chapter where the saints come marching in. And...

Zlatuška: 3:16?

Knuth: What?

Zlatuška: 3:16?

Knuth: So there are different motifs here. I'm trying to see which are the easiest to explain. Well, ??? mentioned that [it] has lots of different styles, and since this is about the Apocalypse, one of the styles had to be calypso. So in Chapter 7, we get calypso: *music playing* Now, that's trying to imitate an organ. Let's try to just do a piano. I don't know how to do this here. Voice... Voice "church [organ]"... Let's change other voices... How do we go down?

Szaniszlo: What kind of voice?

Knuth: Just take piano, for example. Grand piano, here we go. *music playing* Now this last thing is *music playing* Harry Belafonte, so "run Venezuela". *music playing and Knuth singing* "She ran with the tailor." *laughter* That's what I'm singing to myself when I was writing this piece. However, this is, of course, taking place as the saints come marching in. *music playing* So at this point there's a grand shout, comes along, and *chord plays* at the is called *music playing*. That's God. *music playing* Next is: *music playing* This is the motif for the lamb, one of the principal characters, and it's supposed to sound like baa baa. *music playing* So the scene we have here is that the saints go onto this hill, and then there are 24 elders, and the elders form a chromatic scale of 24 notes. *music playing* Besides the elders and other characters are Seraphim, and there are four creatures, and the theme for the four creatures is *music playing*. And you can do this: *music playing*.

Is that enough? *applause*

Zlatuška: By the way, is it difficult to actually play something for organ just on piano? If I understand...

Knuth: Yes, but it's also very difficult... It's also very difficult to play this on the organ. Jan [Rotrekl, the performer of the Fantasia Apocalyptica], the organist, has had to work very hard in order to do it. When I wrote this piece, I didn't hold back. I knew it was the only piece I was ever going to write, and so I didn't bother to simplify it. I wrote what I thought I wanted to hear, and so he has to play what I wrote.

Zlatuška: With (???) Don, when I asked him [Don] whether we can perform this, he mentioned that the organist who did the world premiere was ill and unable to play that, but there is a proof that it is playable. *laughter*

Knuth: *laughs* Yes. I can play it, but not at speed. Well, the organist who did work on it actually fell in love with—I'm glad to say—and he wrote me a couple weeks ago saying that you he's feeling withdrawal symptoms, he wants to play it again.

Zlatuška: Anybody else?

someone: I have a question. I have read about your WEB and CWEB projects. Do you think that the problem in Computer Science that you were trying to solve with this projects is already solved by maybe new programming languages or different approaches towards documentation? Or do you think that the projects are still relevant today? And maybe a follow-up: If you had ability to write CWEB or WEB again today, would you do anything differently than you did before?

Knuth: To me, it's one of the things I love the most about my life, is that I can write programs in CWEB. However, in order to do that, I'm also living with the fact that the implementation that I have had never been tuned up to a great programming environment. So, for example, I start a CWEB program, and I always type a few things. Like I always type a line that says: |@...| at-sign asterisk index period, and I put that at the bottom. It's a little bit of a nuisance, but in fact everything in life has some small nuisances, and so as long as something is working for 97 % of the time, I'm not gonna spend much time figuring out the 3 %, but a mature program, they start fixing up the 3 %.

So I use CWEB knowing that it was a quick and dirty implementation, but it still does almost everything that I want, exactly as I want. And I have an Emacs interface to CWEB. My wife will tell you I come out of my office several times a day saying: "It's so fun programming in CWEB!"

And the reason is that as I'm doing it, it seems to me that I'm combining the formal and the informal aspects of the program the way they ought to be. In order to understand any complicated technical subject, one of the main tricks of it, of a person who writes about Computer Science, mathematics, is to say everything twice: once formally and once informally. Maybe three times, but from different perspectives, but you don't only give a formula, but you say what the variables in the formula mean. You write a program, you not only declare something to be ??? and a variable. You say what has an invariant relation: This is the number of nonzero elements in the array or something, but you combine formal and informal.

And that's the way CWEB works. It breaks down a program into a small number of pieces that fit together in a small number of ways. But each module of the program is a combination of informal—which you write in natural language—and formal—which you write in whatever formal language you're using. In CWEB, it's C, but there are many different flavors of web.

So I really think there's nothing else anywhere near as good as an approach, but I know that there are many ways to refine it, and I'm not interested in actually pursuing the refinements because it's good enough for me. On the other hand, most of the world writes... you look on the solid programs on GitHub. You'll find that they're probably not using CWEB, but there's a certain style that's become... I don't know, I don't like C++ program because... it's so ambiguous. Each C++ compiler does different things with these programs, and so when people... If somebody sends me a program in C++, I don't know what subset of the language they're using. There's all kind of things going on automatically, and the programmer knows what she's doing, but their reader doesn't know which subset is there, so I don't care for that.

But let's suppose we look at a typical module that we get, that's written in C, and it's a style of programming that people have learned to read and maintain, and so it works. I can say: "Oh yeah, let me rewrite your program in CWEB, and you'll see that it actually not only is better, it's more reliable, and you'll realize that certain features were missing because of the discipline of literate programming." However, I don't believe I'll ever convert the world with this. It'd be like saying Esperanto is a much better language than English, so therefore let's abolish English. English works well enough so that some ideal language isn't going to displace it.

Zlatuška: There's question by Tim: Paul Erdős spoke of book in which God kept the most elegant mathematical proofs in the same sense of divinity that are key??? I would extend this also to the most elegant programs, but...

Knuth: I read a very strange paper recently where they asked people to compare algorithms to music. One group of subjects, they were supposed to compare algorithms to music, and another group of subjects was supposed to compare the algorithms to art. Where was this paper? It was one of the papers I read in the last three weeks. Anyway, these people presented keep sort, binary search, ... they took five algorithms that they thought were classic, and then they showed the subjects several pieces of music, and the subjects were supposed to say: "Okay, now match this algorithm to this piece of music." They found—maybe a hundred subjects—that there was a fairly good correlation, it wasn't random permutation of this matching between algorithms and music. But then they did another group, and they showed them five paintings, and they were supposed to again associate this with the painting, and that didn't work at all. But maybe they didn't choose the right paintings. I don't think all kinds of art are equivalent. Certainly, there's a quantitativeness to music that's similar to Computer Science.

Zlatuška: Suppose we create a technology to faithfully copy and simulate working human brain. Would you want to continue living as such a simulation? That's apparently the Kurzweil's idea.

Knuth: So there's a line in... That's a Gershwin's song Ain't Necessarily So. And it says something like Methuselah lived nine hundred years. Methuselah lived nine hundred years, but who calls that livin' if no gal won't give in to a man who was nine hundred years. So there's more to living than thinking.

Knuth: Your favorite fractal?

Knuth: My favorite fractal. Well, it has to be the dragon curve because that was the first one I knew about. The dragon curve—you can look it up—but basically, you take a long sheet of paper. Like thin sheet of paper, like we used to have adding machine tape or something. And you fold it in half, you crease it, you fold it again, and you fold it again, so each time it gets half as big as before. And you've got creases in the paper. Then you open it up, some of the creases go down, and some of them go up. It makes a pattern. An interesting pattern that also turns out to be equivalent to the Legendre symbol of minus one over anything (???). Anyway it's a pattern. You open up the paper, and you make all the bends go ninety degrees. So it'll go like this *drawing on blackboard*, and then go like this... something, left-right, left-right. You can round off the corners if you want, but this is the idea of dragon curve.

It turns out it never intersects itself, and if you take four of them, and start them out... I probably didn't drop correct dragon curve, but if I take four of them, it will cover the entire plane, with four of them. I had a lot of fun proving that theorem in the 80s. And I was really proud when that theorem was picked up in Russia in Quant magazine, which late sixties, of course, there was the iron curtain then, and they illustrated my theorem with four colors in this magazine written for high school students in Russia. It says, you know, I knew enough Russian to translate it, it said: This is a difficult theorem, it was proved by Donald Knuth.

Now we see Czech version of... my books.

Zlatuška: I think the topmost question was already tackled.

Knuth: Are there any other questions from the...

Zlatuška: From the floor?

someone: What kind of organ do you have at home? Is it a pipe organ? Mechanically controlled or a digital or electronic...

Knuth: Yes, it has 850 pipes or so, and the website explains it all. If you go to my home page and look under pipe organ. It was built by a firm called Abbott and Sieker—they're both dead now—but they used to make about four organs a year. And it has 17 ranks of pipes, and I have a major exercise in Fascicle 5—which is coming out next month to say—how many different sounds can you make on this organ that have exactly five pipes going, or exactly six pipes going, or so on. I found it to be quite a fascinating exercise. So if you want to know more about the organ, online has the specs, but also then you've got to buy the book, [to] look at this exercise about that organ that I have.

someone: So I assume that you also prefer pipe organ, mechanically controlled, instead of say, digital organs.

Knuth: I am sorry, I don't understand the question.

Zlatuška: You prefer classical organ to digital?

Knuth: Oh, I see. Yes, absolutely. I heard a pretty good digital organ in England in July, but it's quite rare. They make good digital recordings of organs, most of the organs that I've ever had a chance to play. Even though it's in a great building, and you had the reverberation and everything, it just doesn't match.

I come from a part of the United States where it was illegal to some—it's called America's Dairy State, America's Dairyland—and so it was illegal to sell margarine instead of butter in our state. If you wanted to get margarine, you had to go to Illinois *laughter* to buy it, it was a little cheaper. But I look at butter versus margarine, that sort of has the difference between pipe organ and electronic organ. But also in the early days, before we had good resolution in fonts, there was good printing–butter–and there was the kind that we could get in our lab in the early days–margarine.

Zlatuška: You originally wanted not to have your organ built by some American company, but you imported from some Nordic country. Do you regret that did not work? Or...

Knuth: I heard some really beautiful organs in Denmark, and I inquired about having a Danish organ, and this was in the 70s, there was only one Danish organ in America at that time, it was one in Boston. So I had talked to the builder, and then I found out that there was no way... I had a limited budget, and according to Danish law, the only way I could buy this organ was to agree that the price was indeterminate until after it was built because by Danish law whatever the Union workers wanted to get for it, had to be the amount that was done. So they couldn't get me a fixed price for it, and I might have gone broke, so I was unable to do that.

Zlatuška: So your love for organ was not that big that you would get broke because of it? *laughs*

Knuth: The organ that I finally bought cost $35,000. People were paying that for a house in those days. Now you get a house for $35,000, it's impossible.

Zlatuška: What will happen with your organ when the lease of your house expires?

Knuth: Who knows. But it has only existed in this room. People could unscrew it and reassemble it somewhere.

Knuth: Somebody else? Ah, over there.

someone: You mentioned once that in Super Bowl game the crowds, they ave flags with... showing 3:16 on it. I've always been puzzled what does that have to do with football game.

Knuth: There are people who flaunt their religion. And the most famous verse in the Bible by its number is John 3:16, which is said to be the gospel in a nutshell. It says the God loved the world in such a way that He gave His only child, and so on. So that's a very famous verse of the Bible. People think that by putting that number up that will give publicity, to make people look up this verse and they will suddenly realize that this should be their religion. At football games, it just happened that the cameras, the camera crews survey the crowd, and so this was a way to get advertising for whatever slogans you wanted to do. And a certain group of people started doing it.

Then there were jokes based on. I mean, there was in baseball game... what was his name... a guy from the Boston Red Sox, but anyway his batting average was 0.316, and his nick, his first name was John. So people hold up a chart saying "John! 0.316!" And this was to be a satire on this phenomenon. But it was something that sort of grew like... we have bumper stickers, slogans that people put on their cars and things like that.

But the fact is that this number, the verse got popular by its number, and I used that later when I wrote this book called 3:16 because I wanted to use a cross section of the Bible in order to understand the complexity of it and have some way of sampling. So I used this in order to get a good cross section of not the Bible itself so much, but all the secondary literature about the Bible. So there have been 100,000s of books written about the Bible, but I could go into a theological library, and most of those books have an index in the back, and they'll say which verses do I refer to. So I go through, and I find out, oh yeah, I only have to read a dozen pages of this book, and I can see what it says about Genesis 3:16 and what it says about Revelation 3:16.

By the way, Revelation 3:16 is one of the verses that's here [in Fantasia Apocalyptica], so I might as well go to Chapter 3. *flipping pages* The verse Revelation 3:16 says something like "because you are lukewarm, neither cold nor hot, I will spit you out of my mouth." The message is that God prefers atheist to people who don't care at all. So when I do verse 3:16, I had a little fun in the music. I used time signature 3/16, which is three sixteenth notes to a measure. And then it says: "I will spit you out of my mouth," so on the organ, we have here in the church on Friday, has a pipe called the spitzflute, so we use a spitzflute for this spitting out of the mouth. So it goes anyway *music playing* and then spitz! *music chord* Listen for that on Friday.

Zlatuška: I guess that we basically finished the time allocated for this. There will be the question: "How are you today?" but that will be postponed for next time. Maybe if Don wakes up in the night, maybe he starts with another sermon in the Church, but... Now, that was just a joke.

So thank you very much, I believe that... *applause* Thank you very much for coming, for having these sessions with us. An invitation to everybody for Friday, 7 p.m. in the Church of Jesuits, where the piece Fantasia Apocalypsa [sic] will be performed.

Citácie

KNUTH, Donald, 2020. *Fantasia Apocalyptica* [online]. Stanfordova univerzita [cit. 2020-06-23]. Dostupné z: www-cs-faculty.stanford.edu/~knuth/fant.html.

LUPTÁK, Dávid, 2019. Fantasia Apocalyptica: Česká premiéra. *Zpravodaj Československého sdružení uživatelů TEXu*. Roč. 29, č. 1–4, s. 11–18. ISSN 1213-8185. Dostupné z DOI: 10.5300/2019-1-4/11.

SOJKA, Petr, 2019a. *Donald Knuth a Dana Scott: Týden s držiteli Turingovy ceny v Brně* [online]. Ed. KUBÍČEK, Petr. Fakulta informatiky Masarykovej univerzity [cit. 2020-06-23]. Dostupné z: fi.muni.cz/events/2019-10-donald-knuth-dana-scott-turing-prize-laureates-brno.html.

SOJKA, Petr, 2019b. *Otázky a odpovědi s Donaldem Knuthem: Umění programování* [online]. Ed. KUBÍČEK, Petr. Fakulta informatiky Masarykovej univerzity [cit. 2020-06-23]. Dostupné z: fi.muni.cz/events/2019-10-08-donald-knuth-question-answer-session-computer-programming-as-an-art-brno.html.

SOJKA, Petr, 2019c. *Otázky a odpovědi s Donaldem Knuthem: Zájmy bez hranic* [online]. Ed. KUBÍČEK, Petr. Fakulta informatiky Masarykovej univerzity [cit. 2020-06-23]. Dostupné z: fi.muni.cz/events/2019-10-09-donald-knuth-question-answer-session-boundless-interests-brno.html.

SZANISZLO, Tomáš, 2019a. *Donald E. Knuth Q&A Session 1 Transcript* [online]. Fakulta informatiky Masarykovej univerzity [cit. 2020-06-23]. Dostupné z: fi.muni.cz/events/2019-10-08-donald-knuth-question-answer-session-computer-programming-as-an-art-transcript-brno.html.

SZANISZLO, Tomáš, 2019b. *Donald E. Knuth Q&A Session 2 Transcript* [online]. Fakulta informatiky Masarykovej univerzity [cit. 2020-06-23]. Dostupné z: fi.muni.cz/events/2019-10-09-donald-knuth-question-answer-session-boundless-interests-transcript-brno.html.

ZLATUŠKA, Jiří, 1996. *Donald E. Knuth doktorem honoris causa Masarykovy univerzity* [online]. Ústav výpočtovej techniky Masarykovej univerzity [cit. 2020-06-23]. Dostupné z: webserver.ics.muni.cz/zpravodaj/articles/59.html.

Summary: Two questions and answers sessions by Donald Knuth at FI MU

In October 2019 the Faculty of Informatics, Masaryk University hosted Donald Knuth as a guest who led two questions and answers sessions at this occasion, dedicated to the themes of Computer Science and art. Besides some background on these lectures you can also find their transcripts in this article.

Keywords: Donald Knuth, Computer Programming as Art, Boundless Interests, Q&A

Tomáš Szaniszlo, xszanisz@fi.muni.cz

Mělo by to fungovat IX – Opakování textu

Peter Wilson

Abstrakt

Článek ukazuje, jak je v LaTeXu možné opakovat text číslovaných prostředí, a to včetně opakování tohoto číslování. Dále ukazuje možnosti, jak zarovnat krátký text do obdélníku.

Klíčová slova: LaTeX, opakování, `\newtheorem`, zarovnání textu, `tabular`

Yet in his feverish mind
He still could find
The miraging domes of Samarkand
Glistering through the roiling sand.

autor neznámý

Cílem tohoto seriálu je ukázat čtenáři krátké kousky kódu, které mohou vyřešit některé z jeho problémů. Doufám, že situaci ještě více nezkomplikuji v důsledku mých chyb. Opravy, poznámky a návrhy na změny budou vždy vítány.

There was an old man named Michael Finnegan
Grew some whiskers on his chinnegan.
The wind came out and blew them innegan.
Poor old Michael Finnegan. Beginnagen.
There was ...

lidová píseň

1. Opakování textu

Existují situace, kdy autor potřebuje opakovaně použít nějaký text na dalších místech dokumentu, například v příloze.

Pokud se jedná o prostý text, pak je situace jednoduchá – definujeme makro obsahující daný text a toto makro použijeme na všech místech, kde se má text vyskytnout. Podobně se vyřeší situace, když máme nějaký častý text, který se má použít v mnoha dokumentech. Pokud je tento text krátký, definujeme makro obsahující tento text. Pokud je text delší, například na celou stránku, pak jej vložíme do nějakého souboru a definujeme makro, které tento soubor načte. Oba

Z anglického originálu *Glisterings* [1] přeložil Jan Šustek.

doi: 10.5300/2020-1-2/98

tyto přístupy jsem použil, když jsem tvořil soubory pro třídu [2] a balíček [3], které se využívají při psaní dokumentů ISO, ve kterých se často používaných podobných textů vyskytuje mnoho, a to jak krátkých, tak dlouhých.

Situace se zkomplikuje, když potřebujeme opakovat text, který obsahuje objekty, které jsou automaticky číslovány LaTeXem. To byl dotaz Davida Romana [4] na `texhax`, který mi přeposlala Barbara Beeton:

> *Snažím se vyřešit následující problém. Potřeboval bych mít v článku dvakrát stejný teorém, přičemž chci, aby měl i stejné číslo. Přitom nechci nastavovat číslo sekce a teorému ručně.*

Dále v textu budu používat následující definice.

```
\newtheorem{teorem}{Teorém}[subsection]
\newsavebox{\ulozenybox}
\newcounter{ulozenapodsekce}
\newcounter{ulozenyteorem}
\newcounter{soucasnapodsekce}
\newcounter{soucasnyteorem}
\newcommand*{\kun}{Příliš žluťoučký kůň úpěl ďábelské ódy.}
\newcommand*{\oslava}{Všichni dobří kamarádi přišli na oslavu.}
\newcommand*{\nudne}{Některé věci se stanou příliš nudnými,
  pokud se opakují příliš často.}
```

1.1. Vložení do boxu

Začneme jednoduchým teorémem.

Teorém 1.1.1 *Všichni dobří kamarádi přišli na oslavu.*

Další teorém je ten, který se bude opakovat. Jedna možnost je vysázet text do boxu a tento box použít všude, kde se má text vyskytnout. Pro vysázení teorému jsme použili následující kód.

```
\savebox{\ulozenybox}{%
  \begin{minipage}{\linewidth}
  \begin{teorem}\label{th1}
  \nudne
  \end{teorem}
  \end{minipage}}
\vspace{\topsep}
\noindent\usebox{\ulozenybox}
\vspace{\topsep}
```

Teorém 1.1.2 *Některé věci se stanou příliš nudnými, pokud se opakují příliš často.*

Zjistil jsem, že musím vložit vertikální mezeru velikosti `\topsep` nad a pod box, abych zachoval vertikální odstup teorému od okolního textu.

Uložený box můžeme použít znovu a tímto teorém zopakujeme.[1]

```
\vspace{\topsep}
\noindent\usebox{\ulozenybox}
\vspace{\topsep}
```

Teorém 1.1.2 *Některé věci se stanou příliš nudnými, pokud se opakují příliš často.*

Při použití této techniky se teorém musí vložit do prostředí minipage, aby se text správně zalomil do řádků. Nevýhodou prostředí minipage je, že není možné jej rozlomit na více stránek. Pokud je teorém krátký a neobsahuje pružné vertikální mezery, není to problém. Na druhou stranu, pokud je teorém dlouhý, může to způsobit problémy při stránkovém zlomu dokumentu.[2]

Následuje další teorém, který se bude opakovat. Nyní však pro pozdější použití uložíme hodnoty `subsection` a `teorem` platné před vysázením teorému.[3]

```
\setcounter{ulozenapodsekce}{\value{subsection}}
\setcounter{ulozenyteorem}{\value{teorem}}
\begin{teorem}\label{th2}
\oslava
\end{teorem}
```

Teorém 1.1.3 *Všichni dobří kamarádi přišli na oslavu.*

1.2. Uložení čísel

Všechny výskyty prostředí `teorem` použité v této podsekci budou automaticky číslovány s prefixem 1.2. Díky tomu, že jsme si hodnoty příslušných čítačů při prvním výskytu teorému 1.1.3 uložili, můžeme je nyní použít. Postup je následující.

1. Uložit současné hodnoty čítačů `subsection` a `teorem`.
2. Nastavit hodnoty těchto čítačů na hodnoty uložené před prvním vysázením teorému.
3. Vysázet teorém.
4. Vrátit hodnoty čítačů na hodnoty uložené v kroku 1.

[1] Při opakovaném vysázení boxu se také opakovaně do souboru `aux` zapíše značka `\label`, a proto LaTeX vypíše varování o opakovaném použití stejného odkazu. (pozn. překl.)

[2] Prostředí minipage interně ukládá svůj obsah do vboxu. V tomto vboxu jsou velikosti pružných vertikálních mezer (například kolem display matematiky) zafixovány a obecně budou jiné než stejné pružné vertikální mezery na příslušné stránce. Přitom triviální řešení používající přímo primitivy TeXu `\setbox` a `\unvcopy` uvedený nedostatek nemá a navíc se teorémy správně rozlomí na stránky. (pozn. překl.)

[3] Kdybychom teorém opakovali v jiné sekci, bylo by nutné uložit i hodnotu `section`. (pozn. překl.)

```
\setcounter{soucasnapodsekce}{\value{subsection}}
\setcounter{soucasnyteorem}{\value{teorem}}
\setcounter{subsection}{\value{ulozenapodsekce}}
\setcounter{teorem}{\value{ulozenyteorem}}
\begin{teorem}
\oslava
\end{teorem}
\setcounter{subsection}{\value{soucasnapodsekce}}
\setcounter{teorem}{\value{soucasnyteorem}}
```

Teorém 1.1.3 *Všichni dobří kamarádi přišli na oslavu.*

Mějme ještě další teorém.

```
\begin{teorem}
Toto je další teorém.
\end{teorem}
```

Teorém 1.2.1 *Toto je další teorém.*

Ještě jednou si zopakujeme teorém 1.1.2.

```
\vspace{\topsep}
\noindent\usebox{\ulozenybox}
\vspace{\topsep}
```

Teorém 1.1.2 *Některé věci se stanou příliš nudnými, pokud se opakují příliš často.*

Pokud je teorém příliš dlouhý, můžeme jej vložit do makra nebo uložit do souboru, který pak načteme.

```
\newcommand*{\teoremkun}{
  \begin{teorem}
  \kun
  \end{teorem}}
\teoremkun
```

Teorém 1.2.2 *Příliš žluťoučký kůň úpěl ďábelské ódy.*

> A work that aspires, however humbly, to the condition of art should carry its justification in every line.
>
> *The Nigger of the Narcissus*
> Joseph Conrad

2. Text zarovnaný do obdélníku

V předešlém článku [5] jsem ukazoval, jak vytvořit různé tvary odstavců. Neukázal jsem však ten, který poté potřeboval použít Brad Cooper. Ten se na `comp.text.tex` ptal:

> *Snažím se udělat něco, ... kde jsou dva řádky nadpisu zarovnány do bloku, a to bez dělení slov.*

Na dotaz přišlo několik odpovědí, které zde uvádím v abecedním pořadí autorů.

Donald Arseneau [6] navrhl řešení v prostředí `tabular` s použitím sloupce se zarovnáním do bloku. Nicméně takový typ sloupce předdefinovaný není, a proto navrhl sloupec zarovnat doprava a vložit vlevo záporně pružnou mezeru.[4]

```
\noindent
\begin{tabular}{@{}r@{}}
\hfilneg KRÁTKÝ ŘÁDEK \\
\hfilneg HODNĚ DLOUHÝ ŘÁDEK \\
\hfilneg Donald Arseneau
\end{tabular}
```

KRÁTKÝ ŘÁDEK
HODNĚ DLOUHÝ ŘÁDEK
Donald Arseneau

Také navrhl řešení s použitím makra `\newcolumntype` z balíčku `array`.

```
\newcolumntype{s}{@{}>{\hfilneg}r}
\noindent
\begin{tabular}{s}
KRÁTKÝ ŘÁDEK \\
HODNĚ DLOUHÝ ŘÁDEK \\
Donald Arseneau
\end{tabular}
```

KRÁTKÝ ŘÁDEK
HODNĚ DLOUHÝ ŘÁDEK
Donald Arseneau

Další řešení poslal Enrico Gregorio [7]. Definuje prostředí `stretchcenter`, které je podobné prostředí `center` a které se používá stejně.

```
\newenvironment{stretchcenter}
  {$$\let\\\cr\vbox\bgroup\ialign\bgroup
```

[4] Prostředí `tabular` k zarovnání doprava interně vloží zleva mezeru `0pt plus 1fil`. Příkaz `\hfilneg` vloží mezeru `0pt plus -1fil`. Tyto dvě mezery vedle sebe v součtu dají mezeru velikosti `0pt`, což znamená, že text ve sloupci bude roztažený až k levému okraji sloupce. (pozn. překl.)

```
    \unskip##\unskip\cr}
  {\crcr\egroup\egroup$$}
\begin{stretchcenter}
KRÁTKÝ ŘÁDEK \\
HODNĚ DLOUHÝ ŘÁDEK \\
Enrico Gregorio
\end{stretchcenter}
```

KRÁTKÝ ŘÁDEK
HODNĚ DLOUHÝ ŘÁDEK
Enrico Gregorio

Dan Luecking [8] poslal dvě řešení. V prvním řešení jednoduše změří nejdelší řádek a ostatní řádky vloží do boxů naměřené šířky.[5]

```
\newlength\maxsirka
\settowidth{\maxsirka}{HODNĚ DLOUHÝ ŘÁDEK}
\noindent\makebox[\maxsirka][s]{KRÁTKÝ ŘÁDEK}\par
\noindent\mbox{HODNĚ DLOUHÝ ŘÁDEK}\par
\noindent\makebox[\maxsirka][s]{Dan Luecking}\par
```

KRÁTKÝ ŘÁDEK
HODNĚ DLOUHÝ ŘÁDEK
Dan Luecking

Toto byla také moje první myšlenka, jak vyřešit původní problém Brada Coopera. Nicméně to vyžaduje ruční práci navíc.

Druhé řešení Dana Lueckinga elegantně využilo přímo primitivní příkaz TEXu `\halign`.

```
\halign{#\cr
KRÁTKÝ ŘÁDEK\cr
HODNĚ DLOUHÝ ŘÁDEK\cr
Dan Luecking\cr}
```

KRÁTKÝ ŘÁDEK
HODNĚ DLOUHÝ ŘÁDEK
Dan Luecking

Pokud chceme, aby řešení bylo odsazeno od okraje, můžeme použít horizontální mezeru přímo ve formátu sloupce tabulky.

[5] Při uvedeném řešení se pravděpodobně vypíše varování o podtečeném boxu. Toto varování je možné potlačit například nastavením `\hbadness=10000`, přičemž doporučuji takovéto potlačování dělat vždy lokálně. (pozn. překl.)

```
\halign{\qquad#\cr
KRÁTKÝ ŘÁDEK\cr
HODNĚ DLOUHÝ ŘÁDEK\cr
Dan Luecking\cr}
```

KRÁTKÝ ŘÁDEK
HODNĚ DLOUHÝ ŘÁDEK
Dan Luecking

Příkaz `\halign` a makro `\ialign`, které použili Dan a Enrico, jsou obvykle před LaTeXovými uživateli skryty. Jádro LaTeXu je však hojně používá při definování různých prostředí, například `tabular`.

Seznam literatury

1. WILSON, Peter. Glisterings. *TUGboat*. 2009, roč. 30, č. 2, s. 287–289.
2. WILSON, Peter. *LaTeX for ISO Standards* [online]. 2002-08-10 [cit. 2020-03-24]. Dostupné z: `http://mirrors.ctan.org/macros/latex/contrib/isostds/iso/isoman.pdf`.
3. WILSON, Peter. *LaTeX Package Files for ISO 10303: Source code* [online]. 2002-01-10 [cit. 2020-03-24]. Dostupné z: `http://mirrors.ctan.org/macros/latex/contrib/isostds/iso10303/stepe.pdf`.
4. ARSENEAU, Donald. *Setting counters to output of a* `\ref` *command* [online]. 2007-02-08 [cit. 2020-03-24]. Dostupné z: `https://tug.org/pipermail/texhax/2007-February/007840.html`.
5. WILSON, Peter. Glisterings. *TUGboat*. 2007, roč. 28, č. 2, s. 229–232.
6. ARSENEAU, Donald. *Re: Text filling the line* [online]. 2007-03-24 [cit. 2020-03-24]. Dostupné z: `news:comp.text.tex`.
7. GREGORIO, Enrico. *Re: Text filling the line* [online]. 2007-03-22 [cit. 2020-03-24]. Dostupné z: `news:comp.text.tex`.
8. LUECKING, Dan. *Re: Text filling the line* [online]. 2007-03-23 [cit. 2020-03-24]. Dostupné z: `news:comp.text.tex`.

Summary

This paper shows possibilities in LaTeX to repeat text of numbered environments, including repetition of the numbering. The paper also shows how to align a text into a rectangle.

Key words: LaTeX, repetition, `\newtheorem`, text alignment, `tabular`

Peter Wilson, herries.press@earthlink.net
18912 8th Ave. SW
Normandy Park, WA 98166 USA

Zpravodaj Československého sdružení uživatelů TEXu
ISSN 1211-6661 (tištěná verze), ISSN 1213-8185 (online verze)

Vydalo:	Československé sdružení uživatelů TEXu vlastním nákladem jako interní publikaci
Obálka:	Antonín Strejc
Ilustrace na obálce:	Marian Genčev
Počet výtisků:	280
Uzávěrka:	30. 6. 2020
Odpovědný redaktor:	Jan Šustek
Redakční rada:	Pavel Haluza, Lukáš Novotný, Vít Novotný, Michal Růžička a Jan Šustek (šéfredaktor)
Vědecká rada:	Ján Buša (předseda), Jiří Demel, Jaromír Kuben (zástupce předsedy), Jiří Rybička a Petr Sojka
Technická redakce:	Vít Novotný
Evidenční číslo MK:	E 7629
Adresa:	CSTUG, Nejedlého 373/1, 638 00 Brno
Email:	cstug@cstug.cz

Zřízené poštovní aliasy sdružení CSTUG:

`bulletin@cstug.cz`, `zpravodaj@cstug.cz`
korespondence ohledně Zpravodaje sdružení

`board@cstug.cz`
korespondence členům výboru

`cstug@cstug.cz`, `president@cstug.cz`
korespondence předsedovi sdružení

`gacstug@cstug.cz`
grantová agentura CSTUGu

`secretary@cstug.cz`, `orders@cstug.cz`
korespondence administrativní síle sdružení, objednávky CD a DVD

`cstug-members@cstug.cz`
korespondence členům sdružení

`cstug-faq@cstug.cz`
řešené otázky s odpověďmi navrhované k zařazení do dokumentu CSFAQ

`bookorders@cstug.cz`
objednávky tištěné TEXové literatury na dobírku

ftp server sdružení:
`ftp://ftp.cstug.cz`

www server sdružení:
`https://www.cstug.cz`

CONTENTS