Department of Mathematics
Faculty of Mechanical Engineering
Slovak University of Technology in Bratislava

6th International Conference
APLIMAT 2008

LaTeX A NEOBVYKLÉ VÝPOČTY V PEVNEJ RÁDOVEJ ČIARKE

BLAŠKO, Rudolf, (SK)

Abstrakt. LaTeX používa mnoho užívateľov, mnohí ho používajú iba ako textový editor, ale mnohí pomocou neho dokážu interpretovať veľmi zaujimavé výsledky. Tento článok sa snaží poukázať na niektoré skutočnosti, ktoré LaTeX dokáže. Aj keď cesta týmto smerom nie je najelegantnejšia a ani jednoduchá, je rozhodne veľmi zaujimavá a poučná a dokazuje, že TeX je veľmi silný prostriedok v mnohých smeroch.

Kľúčové slová. LaTeX, programovanie, výpočty v pevnej rádovej čiarke, príprava skúšobných materiálov.

LaTeX AND UNUSUAL FIXED POINT COMPUTATIONS

Abstract. For many of LaTeX users it is sufficient to use this powerful system like advanced text processor, but there are also power users which can do many interesting things using LaTeX programming features. In this contribution we will show some of LaTeX capabilities, which enable us to prepare examination materials for students. The way of LaTeX programming is not the simplest one, nor it is very elegant, but can be interesting and instructive, showing the power of LaTeX language also in non-traditional areas of general computing, not only for text layout and processing.

Key words and phrases. LaTeX, programming, fixed point computations, preparation of examination materials.

Mathematics Subject Classification. Primary 68U15; Secondary 97D40.

1 Úvod

LaTeX je veľmi užitočný prostriedok na reprezentáciu našich výstupov. Ja ho používam už mnoho rokov na všetky svoje písomné prejavy. Človek samozrejme občas musí napísať dokument aj v inom prostredí, ale to sú v mojom prípade iba nepopulárne výnimky.

Keďže pôsobím ako pedagóg na vysokej škole, som nútený pripravovať písomné podklady pre študentov. A to nie sú len učebné texty, ale aj písomné skúšky. Dlhé roky som ich pripravoval tak, že som si na konkrétny termín pripravil písomku s konkrétnymi údajmi a jej riešenie som prepočítal. Ak som chcel použiť túto písomku v inom termíne s pozmenenými príkladmi, musel som to ručne prepísať v zdrojovom texte pre zadania a potom to prepočítať. Je tu samozrejme aj možnosť použiť predchádzajúce riešenie a prepísať iba konkrétne údaje. Ale to je podľa môjho názoru namáhavejšie ako vyriešiť celý príklad od začiatku.

Najlepšie by bolo mať príklad pripravený, vrátane zadania a riešenia, v zdrojovom kóde LaTeX-u a iba vhodným doplnením parametrov generovať vždy iný, relatívne nový príklad. To vyžaduje kompletné teoretické zvládnutie riešenia a tiež podporu od prostredia, ktorá by dovolila aspoň jednoduchú aritmetiku s reálnymi číslami. Nejaký čas som to riešil iba všeobecne s tým, že som pri riešení každej písomky nakoniec „ručne" doplnil parametre do všeobecného vzorca a výsledok som vyčíslil.

Pred niekoľkými rokmi som sa dopočul o možnosti jednoduchých výpočtov v prostredí LaTeX. Po prečítaní článku [1], *Výpočty a diagramy v LaTeXu*, ktorý bol uverejnený v Zpravodaji Československého sdružení uživatelů TeXu v r. 2004 som zmenil prístup. V spomínanom článku sú jednak opísané grafické príkazy použiteľné v prostredí `picture` a obsiahnuté v knižniciach `curves.sty`, `diagram.sty` a jednak príkazy pre aritmetiku v pevnej rádovej čiarke obsiahnuté v knižnici `fp.sty` (fixed point package [6]). Prostredie `picture` používam na kreslenie väčšiny obrázkov. Všetky obrázky v učebnici matematickej analýzy [2] sú urobené v tomto prostredí. Pri zložitejších grafoch som externým programom vygeneroval súradnice jednotlivých bodov a tie som exportoval do prostredia TeXu a graf vykreslil pomocou príkazu `\curves`.

Knižnica `fp.sty` ponúka široké možnosti pre vyhodnocovanie číselných výrazov a používam ju pri generovaní rôznych (pre študentov) a zároveň rovnakých (z pohľadu učiteľa) príkladov. Každý príklad je interpretovaný ako malý program. Podľa vstupných parametrov vypíše teoretické (všeobecné) riešenie, konkrétne riešenie pre konkrétne parametre alebo chybové hlásenie pre nekorektné parametre. Tento prístup objasním na dvoch konkrétnych príkladoch — na jednoduchej limite a na výpočte opakovanej Hornerovej schémy. Je samozrejmé, že v inom prostredí by to iný užívateľ vyriešil pre neho pohodlnejšie, ale určite nie s krajším výstupom. Na porovnanie je na konci článku uvedená ukážka generovania zdrojového textu pre LaTeX na výpočet opakovanej Hornerovej schémy v jazyku Python [5].

2 Knižnica fp.sty

Knižnica makier `fp.sty` je určená pre počítanie s reálnymi číslami a sú v nej implementované aj jednoduché reálne funkcie. Jej autorom je M. Mehlich [6] a nadväzuje na súbor `realcalc.tex` od F. Buchholza [4], ktorý vo všetkých smeroch podstatne rozširuje. Deklaruje sa v úvode dokumentu príkazom

```
\usepackage[<voľba>]{fp}
```

Parameter <voľba> môže mať tvar nomessages (do súboru *.log sa nevypisujú žiadne informácie), debug (do súboru *.log sa vypisujú všetky názvy a výsledky operácií), resp. ak tento parameter nie je zadaný, do súboru *.log sa vypisujú iba názvy vykonávaných operácií. Systém fp.sty počíta s reálnymi číslami v rozsahu $\pm 99\ldots 99.99\ldots 99$ (18 číslic pred a 18 číslic za desatinnou bodkou).

Všetky operácie sa realizujú so vstupnými parametrami <arg>, ktorými sú reťazce číslic v danom rozsahu v zložených zátvorkách, prípadne príkazy (commands) <cmd> obsahujúce tieto reťazce. Pomocou príkazu \FPset<cmd><arg> definujeme <cmd>, čo môže byť ľubovoľný reťazec, konštanta alebo iný príkaz.

Popis jednotlivých implementovaných operácií a funkcií nebudem uvádzať, čitateľ ich môže nájsť v [1, 6]. V systéme sú preddefinované konštanty \FPpi, \FPe (π a e) a operácie \FPadd (sčítanie), \FPsub (odčítanie), \FPmul (násobenie), \FPdiv (delenie), \FPabs (absolútna hodnota), \FPneg (zmena znamienka), \FPsgn (signum), \FPmin, \FPmax, \FPclip (vypustenie záverečných núl za desatinnou bodkou), \FPround (zaokrúhlenie na danú pozíciu), \FPtrunc (useknutie desatinných miest od danej pozície). Ďalej sú tu definované goniometrické funkcie \FPsin, \FPcos, \FPtan, \FPcot, exponenciálna funkcia \FPexp, logaritmus s prirodzeným základom \FPln, mocnina \FPpow a odmocnina \FProot. Číselný výraz $2\cdot 3+6$ by sme pomocou uvedených príkazov zapísali v tvare:

```
\FPmul\vA{2}{3} \FPadd\vB{\vA}{6} \FPclip\vys{\vB} \vys
```

Najprv sa vyjadrí $2\cdot 3$ a uloží do \vA, potom sa k tomuto výsledku pripočíta 6 a uloží do \vB. Lenže \vB $=$ 12.000000000000000000, preto musíme odrezať pomocou \FPclip nepotrebné nuly. Výsledok má potom tvar \vys $=$ 12.

Priebeh spracovania numerických výsledkov môžeme vetviť pomocou jednoduchých relačných operátorov \FPifzero (arg$=$0), \FPifneg (arg$<$0), \FPifpos (arg$>$0), \FPifint (arg je celé číslo), prípadne pomocou dvojparametrových operátorov \FPiflt (arg1$<$arg2), \FPifeq (arg1$=$arg2), \FPifgt (arg1$>$arg2).

Ich použitie si ukážeme na dvoch nasledujúcich makrách. Prvé makro \prI otestuje kladnosť daného čísla a druhé makro \prII zistí a napíše, či prvé číslo je alebo nie je väčšie ako druhé číslo. Deklarácia makier je nasledovná:

```
\def\prI[#1]{číslo $#1$ je \FPifpos{#1}kladné\else záporné\fi}

\def\prII#1#2{číslo $#1$ \FPiflt{#1}{#2}\else nie \fi %
                                     je menšie ako číslo $#2$}
```

Ak je podmienka splnená, vykoná sa časť priamo za príkazom, ak nie, vykoná sa časť za \else. Príkaz musí byť ukončený časťou \fi. Dôležitou je tiež skutočnosť, že tieto podmienené príkazy je možné použiť iba v priamych úsekoch programu, nie v cykloch. Napríklad pre \prI[2], \prI[0], \prI[-2] sa vypíše: číslo 2 je kladné, číslo 0 je kladné, číslo -2 je záporné. Príkaz \prII{2}{4} vypíše: číslo 2 je menšie ako číslo 4.

Vhodnou kombináciou s knižnicou ifthen.sty, obsahujúcou rozhodovacie príkazy a deklarovanou v úvode príkazom \usepackage{ifthen}, môžeme odstrániť aj nedostatok s výpisom nuly v príkaze \prI.

V knižnici `<fp.sty>` sú implementované makrá na určenie reálnych koreňov lineárnej (`\FPlsolve`), kvadratickej (`\FPqsolve`), kubickej (`\FPcsolve`) rovnice a aj rovnice 4. stupňa (`\FPqqsolve`).

Predchádzajúci spôsob zápisu pomocou `\FP`-funkcií je pomerne zložitý. V praxi sa používa oveľa jednoduchší zápis pomocou príkazu

```
\FPeval<cmd>{<výraz>}
```

ktorý vyhodnotí `<výraz>` v zložených zátvorkách a jeho výsledok uloží do príkazu `<cmd>`. Pre `<výraz>` platia nasledujúce pravidlá:

— `\FP`-funkcie sa zapisujú bez `\FP` (napr. `sin` namiesto `\FPsin`),

— namiesto `add`, `sub`, `mul`, `div`, `pow` sa môžu použiť `+`, `-`, `*`, `/`, `^`,

— príkazy a premenné sa píšu bez `\` (napr. `result` namiesto `\result`),

— výraz začínajúci mínusom musí byť v zložených zátvorkách (napr. `{-#1}`),

— záporné čísla v argumente musia byť v zátvorkách (napr. `\FPeval\vA{2*#1}` vyhlási chybu pre záporné `#1`, správne má byť `\FPeval\vA{2*(#1)}`).

3 Použitie fp.sty

Na úvod uvedieme makro, ktoré vypíše kvadratickú rovnicu a vypočíta jej reálne korene. Najprv definujeme niektoré pomocné makrá, ktoré využijeme aj neskôr.

```
\newcommand\ifJ[1]{%
   \ifthenelse{\equal{#1}{1}}{}{\ifthenelse{\equal{#1}{-1}}{-}{#1}}}

\newcommand\ifPMO[1]{%
             \ifthenelse{\equal{#1}{0}}{}{\FPifpos{#1}+\else\fi#1}}

\newcommand\ifPMJ[1]{%
               \FPifpos{#1}{\ifthenelse{\equal{#1}{1}}{+}{+#1}}%
                     \else{\ifthenelse{\equal{#1}{-1}}{-}{#1}}\fi}

\newcommand\ifPMOJ[2]{\ifthenelse{\equal{#1}{0}}{}{%
             \FPifpos{#1}{\ifthenelse{\equal{#1}{1}}{+}{+#1}}%
                   \else{\ifthenelse{\equal{#1}{-1}}{-}{#1}}\fi#2}}
```

Podobne definujeme príkazy `\ifO, \ifOJ, \ifPMJ`, dajú sa nájsť v úplnej verzii súboru s makrami [3]. Príkaz `\ifJ` vynechá znak 1, príkaz `\ifO` vynechá znak 0, príkaz `\ifJO` vynechá znak 1 a tiež znak 0 spolu s druhým parametrom. Tieto príkazy nevyhodnocujú výraz `#1`. Príkazy obsahujúce v názve `PM` vypíšu pred kladný výsledok symbol +. Príkazy obsahujúce v názve `J` vynechajú znak 1. Príkazy obsahujúce v názve `O` vynechajú znak 0, prípadne aj druhý parameter.

```
\newcommand\rb[2][4]{\FPeval{\rbV}{clip(round(#2,#1))}\rbV}

\newcommand\rbx[2][4]{\FPeval{\rbV}{clip(round(#2,#1))}}

\newcommand\rbPrirad[3][4]{%
                        \FPeval{\rbV}{clip(round(#3,#1))}\FPset#2\rbV}

\newcommand\rbSignum[2][4]{\FPeval{\rbV}{clip(round(#2,#1))}%
                                             \FPifneg\rbV-\else\fi}

\newcommand\rbAbs[2][4]{\FPeval{\rbV}{clip(abs(round(#2,#1)))}\rbV}
```

Príkaz `\rb` vyhodnotí výraz `#2`, zaokrúhli ho na `#1` číslic (implicitne sú nastavené 4 desatinné miesta) a potom odreže prebytočné nuly. Príkaz `\rbx` výsledok nevypíše, ale priradí ho príkazu `\rbV`. Príkaz `\rbPrirad` výsledok nepriradí príkazu `\rbV` ale príkazu s názvom `#2`. Príkaz `\rbSignum` vypíše znamienko výrazu `#2` (t. j. nič pre kladné a znamienko - pre záporné číslo) a príkaz `\rbAbs` vypíše absolútnu hodnotu výrazu `#2`.

Nasledujúce makro overí deliteľnosť celého čísla `#2` celým číslom `#1`. V prípade kladnej odozvy vykoná príkazy `#3` a v opačnom prípade príkazy `#4`. Implicitne je nastavené testovanie na deliteľnosť číslom 2.

```
\newcommand\rbDeli[4][2]{\rbPrirad{\rbX}{#1}%
    \ifthenelse{\equal{\rbX}{0}}{{\color{red} menovateľ=0}}{%
            \FPeval{\rbV}{trunc(abs(#2)/abs(#1),0)}%
            \FPeval{\rbW}{trunc((abs(#2)+abs(#1)-1)/abs(#1),0)}%
                            \ifthenelse{\rbV=\rbW}{#3}{#4}}}
```

Nasledujúce makro vyrieši kvadratickú rovnicu a vypíše jej reálne korene. Zadanie príkladu vypíše príkaz `\prikladI` a výsledok príkaz `\VprikladI`. Oba príkazy majú identické parametre, ktorými sú postupne koeficienty rovnice. V prípade `#1=0` sa vypíšu všeobecné tvary zadania a výsledku.

```
\newcommand\prikladI[3]{Nájdite reálne korene rovnice $\ifthenelse{%
  \equal{#1}{0}}{ax^2+bx+c}{\ifJ{#1}x^2\ifPMOJ{#2}{x}\ifPMO{#3}}=0$.}

\newcommand\VprikladI[3]{\ifthenelse{\equal{#1}{0}}{Rovnica %--- #1=0
   $ax^2+bx+c$ má pre $b^2-4ac\ge0$ reálne riešenia
   $x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$.}{%---------------- #1<>0
     \rbPrirad{\rbX}{(#2)*(#2)-4*(#1)*(#3)}%------------ diskriminant
     \rbPrirad{\rbY}{(-1)*(#2)/2/(#1)}%
     \rbPrirad{\rbA}{2*(#1)}\rbPrirad{\rbB}{(-1)*(#2)}%
Rovnica $\ifJ{#1}x^2\ifPMOJ{#2}{x}\ifPMO{#3}=0$
  \FPifneg\rbX nemá reálne riešenie.%------------------------ \rbX<0
    \else má \ifthenelse{\equal{\rbX}{0}}{jedno riešenie %---- \rbX=0
  $x=\rbDeli[\rbA]{\rbB}{}{\frac{\rbB}{\rbA}=}\rbY$.}{%------- \rbX>0
    dve riešenia $x_1=\frac{\ifO{\rbB}-\sqrt{\rbX}}{\rbA}=
                          \rb{((\rbB)-root(2,\rbX))/(\rbA)}$,
```

```
$x_2=\frac{\ifOJ{\rbB}{+}\sqrt{\rbX}}{\rbA}=
                    \rb{((\rbB)+root(2,\rbX))/(\rbA)}$.}\fi}}
```

Príkaz `\VprikladI` s parametrami `011`, `{1}{-4}{4}`, `{2}{0}{-9}`, `209`, `154` vypíše:

Rovnica ax^2+bx+c má pre $b^2-4ac \geqq 0$ reálne riešenia $x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$.

Rovnica $x^2-4x+4=0$ má jedno riešenie $x=2$.

Rovnica $2x^2-9=0$ má dve riešenia $x_1=\frac{-\sqrt{72}}{4}=-2.1213$, $x_2=\frac{\sqrt{72}}{4}=2.1213$.

Rovnica $2x^2+9=0$ nemá reálne riešenie.

Rovnica $x^2+5x+4=0$ má dve riešenia $x_1=\frac{-5-\sqrt{9}}{2}=-4$, $x_2=\frac{-5+\sqrt{9}}{2}=-1$.

Ďalšie makro vyrieši jednoduchú limitu. Pre zjednodušenie práce je definovaný príkaz `\newcommand\LIM[1]{\lim\limits_{n\to\infty}{#1}}`.

```
\newcommand\prikladJ[4]{\rbPrirad\OKtest{0}%-------- pomocny prepinac
\ifthenelse{\equal{#1}{0}}{\rbPrirad\OKtest{1}%----------------- #1=0
  \def\LZ{n^k+an+b}\def\PZ{n-c}\def\odm{\sqrt[k]}%----- pomocne makra
  \def\LZs{\odm{\LZ}}\def\LZz{\odm{n^{k-1}+a+\frac{b}{n}}}%
  \def\PZs{\odm{\PZ}}\def\PZz{\odm{1-\frac{c}{n}}}}{%---------- #1<>0
\def\ZLE{{\color{red}\bf zlé vstupy:} $k=#1$, $a=#2$, $b=#3$, $c=#4$
    $\Rightarrow$  OK: $k=2,3,\dots$, $|a|+|b|>0$}%------------------
\rbPrirad{\rbX}{abs(#2)+abs(#3)}%------- nasleduje test vstupnych dat
\ifthenelse{\equal{\rbX}{0} \or #1<2}{\ZLE}{\rbPrirad\OKtest{1}%
\def\odm{\ifthenelse{#1=2}{\sqrt}{\sqrt[#1]}}%--------- pomocne makra
\def\LZ{n^{#1}\rbPMOJ{#2}{n}\rbPMO{#3}}\def\PZ{n\rbPMO{(-1)*(#4)}}%
\def\LZs{\odm{\LZ}}\def\LZz{\odm{n^{\rbJ{(#1)-1}}\ifPMO{#2}{}%
                              \rbSignumPMO{#3}{\frac{\rbAbs{#3}}{n}}}}%
\def\PZs{\odm{\PZ}}%
\def\PZz{\odm{1\rbSignumPMO{(-1)*(#4)}{\frac{\rbAbs{#4}}{n}}}}}}%
\ifthenelse{\OKtest=0}{}{%----------------- nasleduje vlastny vypocet
$\LIM{\big[\LZs-\PZs\big]}=
     \LIM{\odm{n}\Big[\odm{\frac{\LZ}{n}}-\odm{\frac{\PZ}{n}}\Big]}=$

\hfill $=\LIM{\odm{n}\Big[\LZz-\PZz\Big]}=\infty(\infty-1)=\infty$.}}
```

Na ukážku uvádzame limity vypočítané pre parametre `{2}{1}{-5}{2}`, `4300`, `0000`:

$$\lim_{n\to\infty}\big[\sqrt{n^2+n-5}-\sqrt{n-2}\big]=\lim_{n\to\infty}\sqrt{n}\Big[\sqrt{\frac{n^2+n-5}{n}}-\sqrt{\frac{n-2}{n}}\Big]=$$
$$=\lim_{n\to\infty}\sqrt{n}\Big[\sqrt{n+1-\frac{5}{n}}-\sqrt{1-\frac{2}{n}}\Big]=\infty(\infty-1)=\infty.$$

$$\lim_{n\to\infty}\big[\sqrt[4]{n^4+3n}-\sqrt[4]{n}\big]=\lim_{n\to\infty}\sqrt[4]{n}\Big[\sqrt[4]{\frac{n^4+3n}{n}}-\sqrt[4]{\frac{n}{n}}\Big]=$$
$$=\lim_{n\to\infty}\sqrt[4]{n}\Big[\sqrt[4]{n^3+3}-\sqrt[4]{1}\Big]=\infty(\infty-1)=\infty.$$

$$\lim_{n\to\infty}\big[\sqrt[k]{n^k+an+b}-\sqrt[k]{n-c}\big]=\lim_{n\to\infty}\sqrt[k]{n}\Big[\sqrt[k]{\frac{n^k+an+b}{n}}-\sqrt[k]{\frac{n-c}{n}}\Big]=$$
$$=\lim_{n\to\infty}\sqrt[k]{n}\Big[\sqrt[k]{n^{k-1}+a+\frac{b}{n}}-\sqrt[k]{1-\frac{c}{n}}\Big]=\infty(\infty-1)=\infty.$$

Nasledujúce makro zrealizuje opakovanú Hornerovú schému pre ľubovoľný reálny polynóm supňa najviac 7 (pre potreby výuky to postačí). Makrá v LaTeXe môžu mať najviac 9 parametrov, preto parameter `#1` vyhradíme pre stred rozvoja a ostatných osem pre koeficienty daného polynómu. Všeobecné riešenie nemá praktický význam a nebudeme ho robiť. Z teórie vieme, že takýmto postupom prakticky dostaneme Taylorov polynóm so stredom v bode $c =$ `#1`. Tento problém rieši makro s názvom `\prikladK` (zadanie problému, viď [3]) a makro `\VprikladK` (vlastné riešenie). Implicitne je v makrách nastavené `#1=1`.

```
\newcommand\VprikladK[9][1]{%-------------------- vypocet premennych
\def\xxx{(x\!\rbSignumPM{(-1)*(#1)}\!\rbAbs{#1})}%
  \rbPrirad{\oOb}{(#2)*(#1)}\rbPrirad{\oab}{(#3)+(\oOb)}%
  \rbPrirad{\oOc}{(\oab)*(#1)}\rbPrirad{\oac}{(#4)+(\oOc)}%
  \rbPrirad{\oOd}{(\oac)*(#1)}\rbPrirad{\oad}{(#5)+(\oOd)}%
  \rbPrirad{\oOe}{(\oad)*(#1)}\rbPrirad{\oae}{(#6)+(\oOe)}%
  \rbPrirad{\oOf}{(\oae)*(#1)}\rbPrirad{\oaf}{(#7)+(\oOf)}%
  \rbPrirad{\oOg}{(\oaf)*(#1)}\rbPrirad{\oag}{(#8)+(\oOg)}%
  \rbPrirad{\oOh}{(\oag)*(#1)}\rbPrirad{\oah}{(#9)+(\oOh)}%

        ... nasledujú výpočty ostatných koeficientov ...

  \rbPrirad{\oFb}{(#2)*(#1)}\rbPrirad{\ogb}{(\ofb)+(\oFb)}%
  \def\polH{\ifOJ{#2}{x^7}}\def\TpolH{\ifOJ{#2}{\xxx^7}}%
\def\polG{\ifPMOJ{#3}{x^6}}\def\TpolG{\ifPMOJ{\ogb}{\xxx^6}}%
\def\polF{\ifPMOJ{#4}{x^5}}\def\TpolF{\ifPMOJ{\ofc}{\xxx^5}}%
\def\polE{\ifPMOJ{#5}{x^4}}\def\TpolE{\ifPMOJ{\oed}{\xxx^4}}%
\def\polD{\ifPMOJ{#6}{x^3}}\def\TpolD{\ifPMOJ{\ode}{\xxx^3}}%
\def\polC{\ifPMOJ{#7}{x^2}}\def\TpolC{\ifPMOJ{\ocf}{\xxx^2}}%
  \def\polB{\ifPMOJ{#8}{x}}\def\TpolB{\ifPMOJ{\obg}{\xxx}}%
      \def\polA{\ifPMO{#9}}\def\TpolA{\ifPMO{\oah}}%
\ifthenelse{\not \equal{#2}{0}}{\rbPrirad\stp{8}%-- stupen polynomu 7
}{\ifthenelse{\not \equal{#3}{0}}{\rbPrirad\stp{7}% stupen polynomu 6
  \def\polG{\ifOJ{#3}{x^6}}\def\TpolG{\ifOJ{\ogb}{\xxx^6}}%
}{\ifthenelse{\not \equal{#4}{0}}{\rbPrirad\stp{6}% stupen polynomu 5
  \def\polF{\ifOJ{#4}{x^5}}\def\TpolF{\ifOJ{\ofc}{\xxx^5}}%
}{\ifthenelse{\not \equal{#5}{0}}{\rbPrirad\stp{5}% stupen polynomu 4
  \def\polE{\ifOJ{#5}{x^4}}\def\TpolE{\ifOJ{\oed}{\xxx^4}}%
}{\ifthenelse{\not \equal{#6}{0}}{\rbPrirad\stp{4}% stupen polynomu 3
  \def\polD{\ifOJ{#6}{x^3}}\def\TpolD{\ifOJ{\ode}{\xxx^3}}%
}{\ifthenelse{\not \equal{#7}{0}}{\rbPrirad\stp{3}% stupen polynomu 2
  \def\polC{\ifOJ{#7}{x^2}}\def\TpolC{\ifOJ{\ocf}{\xxx^2}}%
}{\ifthenelse{\not \equal{#8}{0}}{\rbPrirad\stp{2}% stupen polynomu 1
    \def\polB{\ifOJ{#8}{x}}\def\TpolB{\ifOJ{\obg}{\xxx}}%
}{\rbPrirad\stp{1}\def\polA{#9}\def\TpolA{\oah}}}}}}}}%----- stupen 0
\def\ak[##1]##2{\ifthenelse{\stp<##1}{}{##2}}5
\def\FB[##1]{\fbox{$##1$}}%--------------------------------- riadok STP
 \rbPrirad\stpA{\stp-0}\rbPrirad\stpB{\stp-1}\rbPrirad\stpC{\stp-2}%
 \rbPrirad\stpD{\stp-3}\rbPrirad\stpE{\stp-4}\rbPrirad\stpF{\stp-5}%
```

```
  \rbPrirad\stpG{\stp-6}\rbPrirad\stpH{\stp-7}%
 Funkcia $f(x)=\polH\polG\polF\polE\polD\polC\polB\polA$ je polynóm
   stupňa $\stpB$ a jej Taylorov polynóm $T_{\stpB}(x)$
   so stredom v bode $#1$ môžeme určiť pomocou Hornerovej schémy:

\noindent  $\begin{array}{r|*{8}{c}} %----------------- Hornerova schema
\ak[8]{&#2} \ak[7]{&#3} \ak[6]{&#4} \ak[5]{&#5} \ak[4]{&#6}
                                            \ak[3]{&#7} \ak[2]{&#8} &#9 \\
#1 & \ak[8]{&\oOb} \ak[7]{&\oOc} \ak[6]{&\oOd} \ak[5]{&\oOe}
             \ak[4]{&\oOf} \ak[3]{&\oOg} \ak[2]{&\oOh} \\ \cline{1-\stpA}
\ak[8]{&#2} \ak[7]{&\oab} \ak[6]{&\oac} \ak[5]{&\oad}
                     \ak[4]{&\oae} \ak[3]{&\oaf} \ak[2]{&\oag} &\FB[\oah]
\ak[2]{\\ #1 & \ak[8]{&\oAb} \ak[7]{&\oAc} \ak[6]{&\oAd}
           \ak[5]{&\oAe} \ak[4]{&\oAf} \ak[3]{&\oAg} & \\ \cline{1-\stpB}
\ak[8]{&#2} \ak[7]{&\obb} \ak[6]{&\obc} \ak[5]{&\obd}
                                 \ak[4]{&\obe} \ak[3]{&\obf} &\FB[\obg] &}

             ... nasledujú ostatné riadky tabuľky ...


\ak[7]{\\ #1 & \ak[8]{&\oFb} &&&&&& \\ \cline{1-\stpG}
\ak[8]{&#2} &\FB[\ogb] &&&&&&}
\ak[8]{\\ #1 &&&&&&&& \\ \cline{1-\stpH}
&\FB[#2] &&&&&&&}        \end{array}$

 \smallskip

 \noindent $\Rightarrow$  %------------------- Vypis Taylorovho polynomu
   $T_{\stpB}(x)=\TpolH\TpolG\TpolF\TpolE\TpolD\TpolC\TpolB\TpolA$.}
```

Na ďalšej strane sú uvedené výstupy pre rozklad polynómu $x^7 - 2x^3 + x^2 - 1$ na Taylorov polynóm so stredom $c = 1$ (parametre `{1}{0}{0}{0}{-2}{1}{0}{1}`, obr. 1) a rozklad polynómu $2x^4 - 2.2x^3 + x^2 - 1$ na Taylorov polynóm so stredom $c = 1.1$ (parametre `[1.1]{0}{0}{0}{2}{-2.2}{1}{0}{1}`, obr. 2).

Makro by sa dalo zjednodušiť a Hornerova tabuľka by sa generovala vždy kompletne pre polynóm stupňa 7 (vrátane prebytočných núl na začiatku pre polynómy nižších stupňov). Rovnaký efekt dostaneme, ak do riadku s názvom `%------ riadok STP` napíšeme príkaz `\rbPrirad\stp{8}`.

Na záver uvádzam pre porovnanie (samozrejme oveľa jednoduchší program) na výpočet Hornerovej schémy implementovaný v Pythone, ktorého autorom je môj kolega p. Kaukič.

```
from numpy import asarray,empty_like

def complete_horner(pc,x):
    pc=asarray(pc)+0
    n=len(pc)
```

Funkcia $f(x) = x^7 - 2x^3 + x^2 + 1$ je polynóm stupňa 7 a jej Taylorov polynóm $T_7(x)$ so stredom v bode 1 môžeme určiť pomocou Hornerovej schémy:

$$
\begin{array}{c|cccccccc}
 & 1 & 0 & 0 & 0 & -2 & 1 & 0 & 1 \\
1 & & 1 & 1 & 1 & 1 & -1 & 0 & 0 \\
\cline{1-8}
 & 1 & 1 & 1 & 1 & -1 & 0 & 0 & \boxed{1} \\
1 & & 1 & 2 & 3 & 4 & 3 & 3 & \\
\cline{1-7}
 & 1 & 2 & 3 & 4 & 3 & 3 & \boxed{3} & \\
1 & & 1 & 3 & 6 & 10 & 13 & & \\
\cline{1-6}
 & 1 & 3 & 6 & 10 & 13 & \boxed{16} & & \\
1 & & 1 & 4 & 10 & 20 & & & \\
\cline{1-5}
 & 1 & 4 & 10 & 20 & \boxed{33} & & & \\
1 & & 1 & 5 & 15 & & & & \\
\cline{1-4}
 & 1 & 5 & 15 & \boxed{35} & & & & \\
1 & & 1 & 6 & & & & & \\
\cline{1-3}
 & 1 & 6 & \boxed{21} & & & & & \\
1 & & 1 & & & & & & \\
\cline{1-2}
 & 1 & \boxed{7} & & & & & & \\
1 & & & & & & & & \\
\cline{1-1}
 & \boxed{1} & & & & & & &
\end{array}
$$

$$\Rightarrow T_7(x) = (x-1)^7 + 7(x-1)^6 + 21(x-1)^5 + 35(x-1)^4 + 33(x-1)^3 + 16(x-1)^2 + 3(x-1) + 1.$$

Obrázok 1: Rozklad $x^7 - 2x^3 + x^2 - 1$ na Taylorov polynóm so stredom $c = 1$.

Funkcia $f(x) = 2x^4 - 2.2x^3 + x^2 + 1$ je polynóm stupňa 4 a jej Taylorov polynóm $T_4(x)$ so stredom v bode 1.1 môžeme určiť pomocou Hornerovej schémy:

$$
\begin{array}{c|ccccc}
 & 2 & -2.2 & 1 & 0 & 1 \\
1.1 & & 2.2 & 0 & 1.1 & 1.21 \\
\cline{1-5}
 & 2 & 0 & 1 & 1.1 & \boxed{2.21} \\
1.1 & & 2.2 & 2.42 & 3.762 & \\
\cline{1-4}
 & 2 & 2.2 & 3.42 & \boxed{4.862} & \\
1.1 & & 2.2 & 4.84 & & \\
\cline{1-3}
 & 2 & 4.4 & \boxed{8.26} & & \\
1.1 & & 2.2 & & & \\
\cline{1-2}
 & 2 & \boxed{6.6} & & & \\
1.1 & & & & & \\
\cline{1-1}
 & \boxed{2} & & & &
\end{array}
$$

$$\Rightarrow T_4(x) = 2(x-1.1)^4 + 6.6(x-1.1)^3 + 8.26(x-1.1)^2 + 4.862(x-1.1) + 2.21.$$

Obrázok 2: Rozklad $2x^4 - 2.2x^3 + x^2 - 1$ na Taylorov polynóm so stredom $c = 1.1$.

```
    Zc=[pc]
    for nd in range(n):
        pnew=empty_like(pc)
        pnew[0]=pc[0]
        for k~in range(1,len(pc)):
            pnew[k]=pnew[k-1]*x+pc[k]
        Zc+=[pnew]
        pc=pnew[:-1]
    return Zc

def tex_horner(pc,x):
    Zc=complete_horner(pc,x)
    kr=r'\\'+'\n'
    n=len(Zc)
    TT=r'\begin{tabular}{r|'+'r'*(n-1)+'}\n'
    TT+='& '+' & '.join([str(int(c)) for c in Zc[0]])+kr
    TT+=r'\hline\\[-3mm]'+'\n'
    for k~in range(1,n):
        TT+=' & '.join([str(int(x))]+[str(int(c)) for c in Zc[k][:-1]])
        TT+=r' & \framebox{%d}' %Zc[k][-1]
        TT+=' & '*(k-1)+kr
    TT+='\\end{tabular}\n'
    return TT
```

Literatúra

[1] Balda M.: *Výpočty a diagramy v LaTeXu*. Praha : Zpravodaj Československého sdružení uživatelů TeXu, č. 2, ročník 14, CSTUG, 2004.

[2] Blaško R.: *Matematická analýza I*, http://frcatel.fri.uniza.sk/~beerb/ma1/ma1.pdf, 2005.

[3] Blaško, R.: *Súbor blasko-makra.tex*, http://frcatel.fri.uniza.sk/~beerb/Aplimat08/RB-Aplimat08.zip, 2007.

[4] Buchholz F.: *Real arithmetic with big values and high precision.*, file [TeX_directory]\texmf\tex\generic\realcalc\realcalc.tex, 1993.

[5] Kaukič, M.: *Základy programovania v Pylabe*. Košice : FEI TU, 2006.

[6] Mehlkich M.: *Fixed point arithmetic for TeX.*, file [TeX_directory]\texmf\source\latex\fp\fp-package, 1996.