Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Ondrej Krč-Jediný

Maticová kalkulačka

Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce:
Mgr. Lukáš Chrpa

Studijní program: Informatika, programování

2008

Ďakujem svojmu vedúcemu za spoluprácu a rady poskytnuté v priebehu práce, otcovi za trpezlivosť a čas strávený pri debatách ohľadom práce a ďalším členom rodiny a kamarátom, ktorí ma nepriamo podporili v jej dokončení.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím s požičiavaním práce a jej zverejňovaním.

Ondrej Krč-Jediný

V Prahe dňa 30. 5. 2008

Obsah

Názov práce: Maticová kalkulačka

Autor: Ondrej Krč-Jediný

Katedra (ústav): Katedra teoretické informatiky a matematické logiky

Vedúci bakalárskej práce: Mgr. Lukáš Chrpa

e-mail vedúceho: chrpa@kti.mff.cuni.cz

Abstrakt:

Cieľom práce je vytvorenie prostredia, ktoré umožní užívateľovi pracovať s vybranými maticovými algoritmami za účelom porozumenia ich princípom. Pomocou jednoduchého jazyka umožní zadávať požiadavky na výpočty a zobrazovať výsledky. Ďalej obsahuje možnosť zobrazovať vysvetlenie a priebeh výpočtu jednotlivých algoritmov, a tak poskytnúť informácie o ich spôsobe fungovania. V projekte sú implementované niektoré základné a niektoré pokročilé algoritmy na prácu s maticami.

Kľúčové slová: algoritmus, matica, výuka

Title: Matrix Calculator

Author: Ondrej Krč-Jediný

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Mgr. Lukáš Chrpa

Supervisor's e-mail address: chrpa@kti.mff.cuni.cz

Abstract:

The aim of this project is creating an environment which would allow the user to work with chosen matrix algorithms, with the emphasis on understanding their principles. Using a simple programming language, the user will be able to perform computations and display their results. Moreover, the environment is able to show a characterisation, and a description of the course of computation of each algorithm, thus providing information about the means of its operation. The project implements several basic and several advanced algorithms for handling matrices.

Keywords: algorithm, matrix, education, teaching

1 Úvod

1.1 Úvod do problematiky

Obsahom tejto bakalárskej práce je realizácia vybraných maticových algoritmov výukovou formou, čo znamená nielen samotný výpočet algoritmu, ale aj zobrazenie popisu jeho princípu a medzivýsledkov užívateľovi. Okrem samotného výpočtu si teda kladie za cieľ poskytnúť užívateľovi študijnú pomôcku slúžiacu k pochopeniu samotných algoritmov.

Na trhu sa vyskytujú viaceré nástroje realizujúce maticové algoritmy, ako napríklad Mathematica, MATLAB a Maple. Sú to rozsiahle nástroje poskytujúce širokú škálu matematických funkcií, z čoho väčšinou vyplýva netriviálny spôsob práce s nimi. Často je nutné osvojiť si zložitú syntax a mať isté skúsenosti s programovaním. Maticová kalkulačka, tým, že sa špecializuje len na maticové algoritmy, ponúka jednoduchý a intuitívny spôsob na zadávanie matíc a výpočet algoritmov. S programom bude tým pádom môcť bez problémov pracovať aj užívateľ, ktorý nie je zbehlý v programovaní.

Všetky uvedené nástroje ako aj väčšina ďalších sa špecializujú len na jednotlivé výpočty, ale neposkytujú možnosť zistiť, ako sa k výsledku dospelo. Samotný výpočet je akási čierna skrinka do ktorej užívateľ nemôže nahliadnuť a tým pádom nemá možnosť hlbšie preniknúť do problematiky. Maticová kalkulačka na druhej strane ku každému realizovanému algoritmu poskytuje výukovú formu, a teda popis priebehu jeho výpočtu. Jeho myšlienkou je poskytnutie študijnej pomôcky na pochopenie vybraných odvetví práce s maticami.

1.2 Štruktúra práce

Kapitola číslo 1 obsahuje stručný úvod do problematiky a objasňuje moje rozhodnutie venovať sa práve tejto téme.

V kapitole číslo 2 sa nachádza popis algoritmov, ktoré program poskytuje a ich podrobný rozbor, teda samotný spôsob fungovania a zložitosť.

Kapitola číslo 3 poskytuje návod na používanie pre bežného užívateľa. Popisuje prerekvizity potrebné na jeho spustenie a samotnú prácu so všetkými jeho súčasťami.

Kapitola číslo 4 popisuje spôsob používania programu pre programátora. Poskytuje informácie o implementačnej platforme, použitých technológiách, dôležitých dátových štruktúrach a návod na preklad a zostavenie programu.

Kapitola číslo 5 zhŕňa prácu ako celok a popisuje jeho celkové výhody a nevýhody, ako aj prípadné možnosti ďalšieho vývoja.

2 Rozbor

Táto kapitola sa zaoberá rozborom maticových algoritmov, ktoré sú implementované v programe. Skúma ich vlastnosti, vzťahy a zložitosť.

2.1 Vymedzenie základných pojmov

Matica je v matematike definovaná ako obdĺžniková tabuľka čísel alebo iných matematických objektov – *prvkov matice* (táto práca sa zaoberá len maticami obsahujúcimi čísla). Pozostáva z *m* riadkov a *n* stĺpcov, vtedy hovoríme o matici typu $m \times n$. Matice sa využívajú napríklad k výpočtu sústav lineárnych rovníc, pre vyjadrenie prechodu vektoru od jednej báze k druhej, alebo k vyjadreniu obecnej rotácie vektorov. *Prvky* matice označujeme indexmi udávajúcimi riadok a stĺpec, v ktorom sa prvok nachádza. Prvok matice v i-tom riadku a j-tom stĺpci sa značí a_{ij} .

Maticu typu $m \times n$ môžme zapísať ako:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} .$$

Podmatica matice A je matica vzniknutá vynechaním niektorých riadkov a niektorých stĺpcov z A.

Unitná matica je matica obsahujúca 1 na hlavnej diagonále (prvky mimo diagonály môžu byť ľubovolné).

Jednotková matica je matica typu $n \times n$ obsahujúca 1 na hlavnej diagonále a 0 všade inde.

$$I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} .$$

Štvorcová matica je matica typu $n \times n$.

Boolovská matica je matica obsahujúca len čísla 0 a 1. Boolovské matice sa vyskytujú v mnohých problémoch diskrétnej matematiky.

Permutačná matica je matica obsahujúca len 0 a 1 taká, že v každom riadku a v každom stĺpci matice sa nachádza číslo 1 práve raz.

Horná (dolná) trojuholníková matica Matica A typu $m \times n$ je dolná (horná) trojuholníková, ak $A_{ij} = 0$ pre $1 \leqslant j < i \leqslant m (1 \leqslant i < j \leqslant n)$. .

Elementárnymi riadkovými úpravami v matici nazývame tieto operácie:

1. vynásobenie i-tého riadku matice nenulovým číslom
2. pripočítanie j-tého riadku k riadku i

Pivot matice je prvý prvok v riadku matice rôzny od 0.

Hodnosť matice je rovná počtu pivotov v ľubovolnej matici v trojuholníkovom tvare, ktorá bola získaná z matice A elementárnymi riadkovými úpravami.

Regulárna matica je štvorcová matica $n \times n$, ktorej hodnosť je rovná n .

Permutácia je vzájomne jednoznačné zobrazenie (bijekcia) $X \to X$. S_n označujeme všetky permutácie množiny $\{1,2,...,n\}$.

Množina inverzií permutácie p $I(p)=\{(i,j): i<j \; a \; p(i)>p(j)\}$.

Znamienko permutácie p $sgn(p)=(-1)^{|I(p)|}$, kde $I(p)$ je množina inverzií permutácie p a $|I(p)|$ je počet prvkov tejto množiny.

2.2 Operácie a algoritmy obsiahnuté v programe

Táto časť popisuje algoritmy implementované v programe aj s poznámkou o ich zložitosti. V prípade maticových algoritmov sa pri počítaní zložitosti berie do úvahy počet násobení ktoré boli vykonané v priebehu algoritmu.

2.2.1 Sčítanie a odčítanie matíc.

Súčet dvoch matíc je definovaný pre matice rovnakého typu. Súčtom matíc A a B typu $m \times n$ je matica typu $m \times n$ vytvorená sčítaním odpovedajúcich prvkov:

$$\begin{bmatrix} a_{11} a_{21} ... \\ a_{12} a_{22} ... \\ ... \; ... \; ... \end{bmatrix} \pm \begin{bmatrix} b_{11} b_{21} ... \\ b_{12} b_{22} ... \\ ... \; ... \; ... \end{bmatrix} = \begin{bmatrix} a_{11} \pm b_{11} a_{21} \pm b_{21} ... \\ a_{12} \pm b_{12} a_{22} \pm b_{22} ... \\ ... \qquad ... \qquad ... \end{bmatrix}$$

Vzhľadom k tomu, že sčítanie matíc neobsahuje žiadne násobenie, tento projekt sa nezaoberá ďalšími algoritmami na jeho výpočet.

2.2.2 Násobenie matíc

2.2.2.1 Z definície

Súčin dvoch matíc je definovaný pre matice, pre ktoré platí, že počet stĺpcov v prvej matici sa rovná počtu riadkov v druhej matici. Teda súčin matice A typu $m \times n$ a matice B typu $n \times p$ je matica C, ktorej prvky vyzerajú nasledovne:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} * b_{kj}$$

Asymptotická časová zložitosť pre vynásobenie dvoch matíc typu $n \times n$ z definície je $O(n^3)$. Zložitosť pre maticu $m \times n$ a maticu $n \times p$ je $O(mnp)$.

2.2.2.2 Strassenov algoritmus.

Strassenov algoritmus na násobenie matíc funguje nasledovne:

Majme dve matice A a B typu $n \times n$, kde *n* je mocnina 2. Rozdeľme A aj B na 4 podmatice typu $(n/2) \times (n/2)$ a vyjadrime súčin podľa nasledujúcej schémy:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

kde A_{11} je ľavá horná, A_{12} pravá horná, A_{21} ľavá dolná a A_{22} pravá dolná štvrtina matice A (podobne pre matice B a C). Časti výslednej matice matice C môžeme vypočítať takto:

$$\begin{aligned} C_{11} &= A_{11} B_{11} + A_{12} B_{21} \\ C_{12} &= A_{11} B_{12} + A_{12} B_{22} \\ C_{21} &= A_{21} B_{11} + A_{22} B_{21} \\ C_{22} &= A_{21} B_{12} + A_{22} B_{22} \end{aligned}$$

Tento výpočet používa 8 násobení na výpočet matíc C_{ij} . Existuje však spôsob na ich výpočet, ktorý využíva len 7 násobení, ktorý vyzerá takto (Strassenov algoritmus):

$$\begin{aligned} M_1 &= (A_{12} - A_{22})(B_{21} + B_{22}), \\ M_2 &= (A_{11} + A_{22})(B_{11} + B_{22}), \\ M_3 &= (A_{11} - A_{21})(B_{11} + B_{12}), \\ M_4 &= (A_{11} + A_{12}) B_{22} \\ M_5 &= A_{11}(B_{12} - B_{22}), \\ M_6 &= A_{22}(B_{21} - B_{11}), \\ M_7 &= (A_{21} + A_{22}) B_{11} \\ C_{11} &= M_1 + M_2 - M_4 + M_6 \\ C_{21} &= M_4 + M_5 \\ C_{21} &= M_6 + M_7 \\ C_{22} &= M_2 - M_3 + M_5 - M_7 \quad . \end{aligned}$$

Na čiastkové maticové súčiny v týchto rovniciach je možné rekurzívne aplikovať Strassenov algoritmus, až pokiaľ pracujeme s maticami 2×2 , na ktoré použijeme rovnaký postup s tým, že operácie v rovniciach už potom budú obyčajné číselné operácie.

V prípade, že násobené matice nemajú rozmery rovné mocnine 2, budeme postupovať tak, že pokiaľ pracujeme s maticami nepárneho rozmeru, rozšírime ich o jeden nulový riadok a jeden nulový stĺpec (aby boli párneho rozmeru). Párne matice je vždy možné rozdeliť na 4 rovnaké podmatice. Keď ale rozdeľujeme matice, ktorých rozmer nie je mocninou 2, tak musí nastať situácia, že po rozdelení pracujeme s maticami nepárneho rozmeru (napríklad rozdelenie

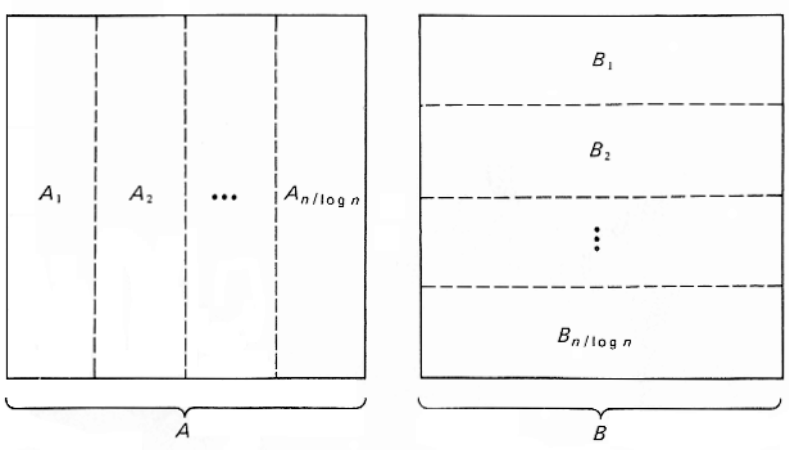
matice typu 6×6 dáva 4 matice typu 3×3). Tieto matice opäť rozšírime na párny rozmer.

Tento algoritmus je efektívnejší ako klasické násobenie matíc z definície, vďaka spomínanému ušetrenému násobeniu pri výpočte súčinu. Jeho asymptotická časová zložitosť pre maticu typu $n\times n$ je $O(n^{2.81})$ [1]. V praxi sa však jeho efektivita prejavuje až pri maticiach veľkých rozmerov. Algoritmus má veľký teoretický význam, pretože ako prvý dokázal, že Gaussova eliminácia nie je optimálna [7]. Na jeho princípe je založený aj do dnes najrýchlejší známy algoritmus na násobenie matíc [2].

2.2.2.3 Algoritmus štyroch Rusov

Algoritmus štyroch Rusov je efektívny spôsob na násobenie boolovských matíc.

Majme 2 boolovské matice A a B typu $n\times n$. Rozdeľme maticu A na matice typu $n\times(\log n)$ a maticu B na matice typu $(\log n)\times n$ podľa nasledujúcej schémy:



Obrázok 2.1: Schéma rozdelenia matíc pri algoritme štyroch Rusov.

Potom platí, že $AB=\sum_{i=1}^{\lceil n/\log(n)\rceil} A_i B_i$.

Algoritmus využíva nasledujúcu skutočnosť: tým, že každý riadok matice A_i má maximálne $\lceil\log n\rceil$ prvkov, existuje najviac $2^{\log(n)}=n$ rôznych súčtov riadkov medzi všetkými A_i . To isté platí pre matice B_i . Tým, že matice sú boolovské, každý riadok vo výslednej matici C_i bude súčtom niektorých riadkov B_i podľa toho, na ktorých miestach obsahuje matica A_i jednotky. Algoritmus si dopredu spočíta všetky možné súčty riadkov matice B_i a uloží do poľa, a potom vždy namiesto počítania súčinu $a_j B_i$ sa na základe a_j zaindexuje do tohto poľa pre výsledok.

Priebeh algoritmu v pseudokóde:

Majme 2 boolovské matice A a B typu $n\times n$.

Definujeme m ako $\lceil\log n\rceil$ (dolná celá časť).

Podľa obrázka 2.1 rozdelíme A na matice $A_1, A_2, \ldots, A_{\lceil n/m\rceil}$, kde $A_i, 1\leqslant i<\lceil n/m\rceil$ (horná celá časť) je matica pozostávajúca z stĺpcov $m(i-1)+1$ až mi matice A, a $A_{\lceil n/m\rceil}$ (horná

celá časť) pozostáva zo zvyšných stĺpcov, prípadne s ďalšími nulovými stĺpcami, keď potrebujeme m stĺpcov. To isté spravíme s maticou B. NUM (v) je číslo reprezentujúce hodnotu obráteného vektoru 0 a 1 (napr. $NUM(0,1,0,1)=9$). Algoritmus vypočíta maticu $C=AB$.

```
begin
    for  i ← 1  until  ⌈n/m⌉  (horná celá časť)
        begin
            ROWSUM[0]  ←  [0, 0, ⋯, 0]   (n)
            for  j = 1  until  2^m − 1  do
                begin
                    nájdime k také, že  2^k ⩽ j < 2^(k+1)
                    ROWSUM[j]  ←  ROWSUM  [j − 2^k]  + b^(i)_(k+1)
                end
            C_i  = matica ktorej j-tý riadok pre  1 ⩽ j ⩽ n  sa rovná
            ROWSUM[NUM ( a_j )], kde  a_j  je j-tý riadok  A_i
        end
    C = Σ_{i=1}^{⌈n/m⌉} C_i   (horná celá časť)
end
```

Asymptotická časová zložitosť tohto algoritmu pre maticu typu $n \times n$ je $0(n^3/\log n)$ [1]. Ako vidíme, algoritmus nepoužíva násobenia, takže tento vzorec sa vzťahuje k počtu sčítaní. V praxi je výrazne rýchlejší ako Strassenov algoritmus, je však určený len pre boolovské matice.

Pseudokód a obrázok v tejto časti je prevzatý z [1].

2.2.3 Gaussova eliminácia

Gaussova eliminácia je algoritmus na úpravu matice do horného trojuholníkového tvaru. Používa sa pri riešení sústav lineárnych rovníc, hľadaní inverzných matíc alebo na zistenie hodnosti matice. Na dosiahnutie cieľa používa elementárne riadkové úpravy. Prebieha v nasledujúcich krokoch (prepis tohto algoritmu je prevzatý z [3]):

Vstup: matica A typu $m \times n$, pre každý riadok (ozn. i):

1. prehľadávame stĺpce matice A zľava a nájdeme prvý stĺpec, ktorý obsahuje nenulový prvok v i-tom riadku alebo v riadku pod ním; ak takýto riadok neexituje, pokračujeme posledným krokom,
2. ak je prvý taký stĺpec B_{xj} , označíme si miesto (i, j) ako miesto pre i-tý pivot,
3. ak je prvok b_{ij} na mieste (i, j) rovný 0, zameníme i-tý riadok s ľubovolným riadkom prod i-tým riadkom, v ktorom je prvok v j-tom stĺpci nenulový; v opačnom prípade pokračujeme hneď nasledujúcim krokom,
4. ak je na mieste (i, j) nenulové číslo, pričítame ku všetkým riadkom pod i-tým riadkom vhodné násobky i-tého riadku tak, aby sme vynulovali všetky prvky v stĺpci B_{xj} , ktoré sa nachádzajú pod pivotom na mieste (i, j) ,
5. ak sú všetky prvky v i-tom riadku matice B a vo všetkých riadkoch pod i-tým riadkom nulové, algoritmus končí.

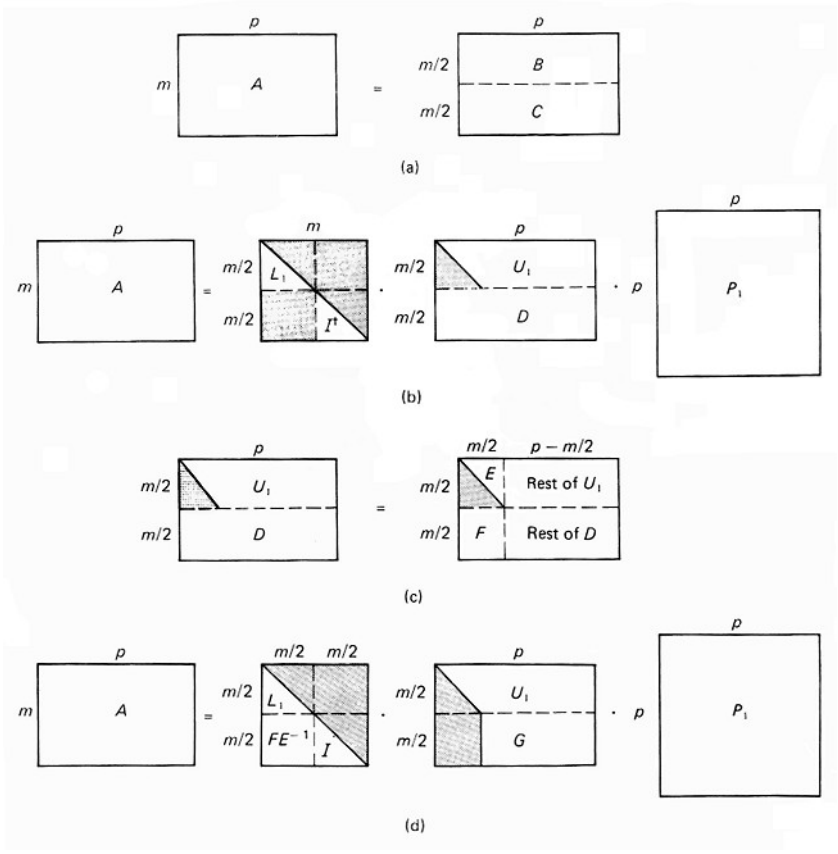
Asymptotická časová zložitosť Gaussovej eliminácie pre maticu typu $n \times n$ je $O(n^3)$.

2.2.4 LUP dekompozícia

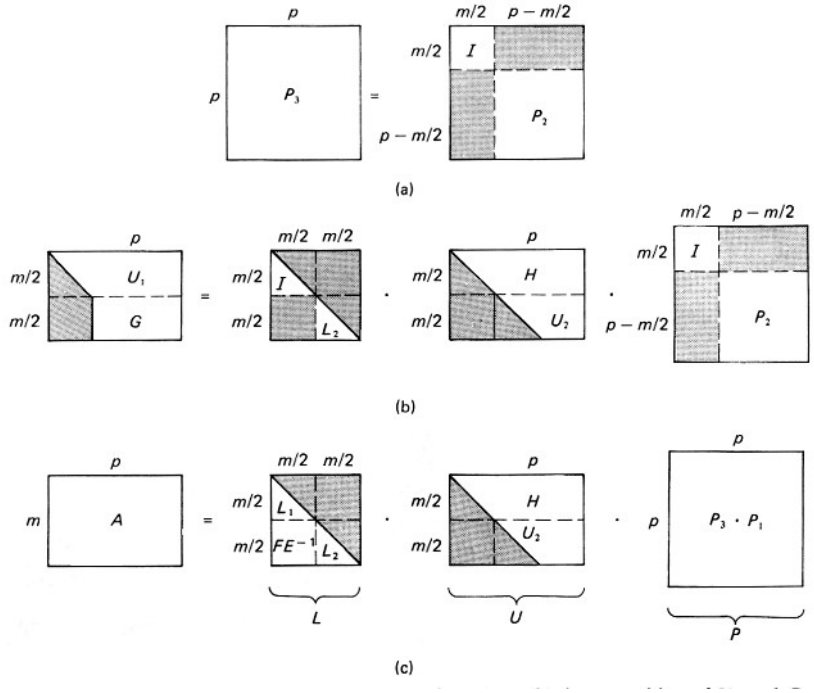
LUP dekompozíciou regulárnej matice A nazývame trojicu matíc L, U, P, pre ktoré platí $A = LUP$, kde L je dolná trojuholníková unitná matica, U horná trojuholníková matica, a P permutačná matica. Využíva sa napríklad na efektívne počítanie podobných lineárnych rovníc. Maticová kalkulačka implementuje takýto algoritmus na jej určenie:

Majme regulárnu maticu A typu $n \times n$, kde n je mocnina 2 (v prípade, že jej rozmer nespĺňa túto podmienku, rozšírime maticu na maticu rozmeru rovnému najbližšej vyššej mocnine 2). Na maticu zavoláme rekurzívnu procedúru FACTOR (A, n, n), ktorej priebeh je znázornený na obrázkoch 2.2 a 2.3. V pseudokóde vyzerá nasledovne:

```
comment A je matica typu m×p
procedure FACTOR (A, m, p)
if (m = 1) then
   begin
      L = [1] (L je jednotková matica 1×1 )
      c = stĺpec matice A obsahujúci nenulový prvok (ak
      existuje)
      P = permutačná matica p×p s vymenenými stĺpcami 1 a c;
      U = AP
      return (L, U, P)
    end
   else
      rozdeľ A na matice B a C veľkosti (m/2)×p  (obrázok 2.2
      (a))
      FACTOR (B, (m/2) , p), dostaneme L1,U1,P1
       D=C P1^-1
      comment V tomto momente môžeme maticu A zapísať ako súčin troch matíc
      z obrázku 2.2 (b)
      E = prvých (m/2) stĺpcov U1  (obrázok 2.2 (c))
      F = prvých (m/2) stĺpcov D (obrázok 2.2 (c))
       G=D−FE^-1 U1
      comment Prvých (m/2) stĺpcov matice G je nulových. V tomto momente
      môžeme maticu A zapísať ako súčin matíc z obrázku o 2.2 (d)
       G' = pravých p−(m/2)  stĺpcov G
      FACTOR ( G' , (m/2) , p−(m/2) ), dostaneme L2,U2,P2
       P3 permutačná matica p×p s I m/2 vľavo hore a P2
      vľavo dole
      H = U1 P3^-1
      comment V tomto bode môžeme vyjadriť maticu zloženú z U1 a G ako
      ukazuje obrázok 2.3 (a). Vloženie pravej časti z obrázku 2.3 (b) do 2.2
      (d) vyjadruje maticu A ako súčin piatich matíc. Prvé dve sú dolné
      trojuholníkové unitná matice, tretia je horná trojuholníková a posledné
      dve sú permutačné matice. Vynásobením prvých dvoch a posledných dvoch
      dostaneme požadovanú dekompozíciu matice A.
      L = matica m×m pozostávajúca z L1,0 m/2,F E^-1 a L2
      (2.3 (c))
      U = matica m×p s H v hornej polovici a O m/2 a U2 v
      spodnej polovici (2.3 (c))
       P=P3 P1
      return (L, U, P)
   end
```



Obrázok 2.2: Priebeh LUP dekompozície 1



Obrázok 2.3: Priebeh LUP dekompozície 2

Algoritmus vypočíta A = LUP pre akúkoľvek regulárnu maticu A. Asymptotická časová zložitosť pre maticu typu $n \times n$ je rovnaká ako asymptotická zložitosť použitého algoritmu na násobenie v rámci algoritmu. V tomto projekte je použité násobenie z definície, a teda zložitosť je $O(n^3)$ [1].

Pseudokód a obrázky v tejto časti sú prevzaté z [1].

2.2.5 Determinant

2.2.5.1 Z definície

Determinant štvorcovej matice A veľkosti $n \times n$ je definovaný nasledovne:

$$det(A) = \sum_{p \in S_n} sgn(p) \prod_{i=1}^{n} a_{i,p(i)} \quad ,$$

kde S_n je množina všetkých permutácií množiny $\{1,2,...,n\}$ a sgn (p) je znamienko permutácie p. Výpočet determinantu z definície je však veľmi neefektívny so zložitosťou $O(n!)$ a je použiteľný len pre matice malých rozmerov.

2.2.5.2 Z Gaussovej eliminácie

Prvým krokom pri výpočte determinantu je aplikovanie Gaussovej eliminácie na skúmanú maticu, a teda jej prevedenie do horného trojuholníkového tvaru. Pre maticu v trojuholníkovom tvare platí, že jej determinant (až na znamienko) sa rovná súčinu prvkov na diagonále. Znamienko sa zmení vždy, keď v je priebehu Gaussovej eliminácie aplikovaná úprava výmena riadkov. Asymptotická časová zložitosť pre maticu typu $n \times n$ je $O(n^3)$.

2.2.5.3 Rozvoj podľa riadku

Pre výpočet determinantu matice rozvojom podľa riadku použijeme vzorec:

$$det(A) = \sum_{j=1}^{n} a_{i,j} (-1)^{i+j} M_{i,j} \quad ,$$

kde $M_{i,j}$ je determinant matice, ktorá vznikne z A odstránením i-tého riadku a j-tého stĺpca. Asymptotická časová zložitosť pre maticu typu $n \times n$ je $O(n!)$. Táto metóda je výhodná, keď riadok, podľa ktorého rozvoj robíme obsahuje veľa núl, pretože zo súčtu vypadnú všetky členy v ktorých sa násobí týmto nulovým prvkom.

2.2.5.4 Z LUP dekompozície

Platí:

$$det(A) = det(L)\, det(U)\, det(P) \quad ,$$

kde A = LUP.

Determinant matice L = 1, pretože je to trojuholníková unitná matica a teda má na diagonále samé jednotky. Z predchádzajúcej časti vieme, že determinant trojuholníkovej matice získame vynásobením prvkov na diagonále. Tým, že U je trojuholníková matica, taktiež stačí na výpočet jej determinantu vynásobiť prvky na diagonále. Matica P je permutačná, čo znamená, že $det(P) = \pm 1$. Znamienko závisí od toho, či je daná permutácia párna alebo nepárna, a teda od znamienka tejto permutácie. Asymptotická časová zložitosť pre maticu typu $n \times n$ je

opäť rovnaká ako asymptotická zložitosť použitého algoritmu na násobenie v rámci LUP dekompozície, a teda v rámci tohto projektu $O(n^3)$.

2.3 Inverzná matica

Inverzná matica k matici A je matica (ozn. A^{-1}), ktorá po vynásobení s maticou A dáva jednotkovú maticu:

$$A\,A^{-1} = I_n \quad .$$

Inverznú maticu je možné zostrojiť pre regulárne matice.

2.3.1 „Klasická“ metóda

Pod **„klasickou“ metódou** na výpočet inverznej matice rozumieme metódu založenú na Gaussovej eliminácii. Platí, že inverznú maticu majú len regulárne matice.

Majme maticu A typu $n \times n$, ku ktorej chceme zostrojiť inverznú maticu. Vedľa seba si napíšeme maticu A a jednotkovú maticu veľkosti n. Na maticu A aplikujeme elementárne riadkové úpravy tak, aby viedli k jej úprave na jednotkovú maticu, pričom rovnaké úpravy ako v matici A budeme robiť aj na jednotkovej matici:

$$\begin{matrix}(A)(I_n)\\ \vdots \\ (I_n)(A^{-1})\end{matrix}$$

Matica vzniknutá z pôvodnej jednotkovej matice je hľadaná A^{-1} .
Asymptotická zložitosť algoritmu pre maticu typu $n \times n$ je rovná zložitosti Gaussovej eliminácie, a teda $O(n^3)$.

2.3.2 Pomocou násobenia matíc

Ďalším spôsobom, ako vypočítať inverznú maticu je metóda založená na násobení matíc a funguje pre všetky regulárne dolné alebo horné trojuholníkové matice. Pre ostatné matice táto metóda nemusí viesť k výsledku.

Majme regulárnu hornú alebo dolnú trojuholníkovú maticu A. Maticu rozdelíme na 4 menšie matice podľa schémy

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

Maticu A_{11} zvolíme tak, aby bola regulárna, a teda budeme postupne voliť podmatice typu 1×1 , 2×2 , … , $n \times n$ začínajúce v ľavom hornom rohu matice A až dovtedy, kým neobjavíme regulárnu maticu. Matice A_{12} a A_{21} zvolíme tak, aby boli zarovnané s maticou A_{11} . Matica A_{22} bude obsahovať zvyšné prvky matice A. Potom môžeme

inverznú maticu vypočítať ako:

$$A^{-1} = \begin{bmatrix} A_{11}^{-1} + A_{11}^{-1} A_{12} \nabla^{-1} A_{21} A_{11}^{-1} & -A_{11}^{-1} \nabla^{-1} \\ -\nabla^{-1} A_{21} A_{11}^{-1} & \nabla^{-1} \end{bmatrix} ,$$

kde $\nabla = A_{22} - A_{21} A_{11}^{-1} A_{12}$.

Na čiastkové inverzie môžeme rekurzívne aplikovať rovnaký postup, až kým docielime výsledku. Pre ukončenie rekurzie budeme čiaskové inverzie matíc počítať klasickou metódou.

Asymptotická časová zložitosť tohto algoritmu pre maticu typu $n \times n$ je rovnaká ako asymptotická zložitosť použitého algoritmu na násobenie matíc počas výpočtu. V tomto projekte je použité násobenie z definície, takže zložitosť je $O(n^3)$.

2.3.3 Z LUP dekompozície

Pre regulárnu maticu A platí:

$$A^{-1} = P^{-1} U^{-1} L^{-1} ,$$

kde $A = LUP$ je LUP dekompozícia matice A. Asymptotická časová zložitosť pre maticu typu $n \times n$ je opäť rovnaká ako asymptotická zložitosť použitého algoritmu na násobenie v rámci LUP dekompozície, a teda v rámci tohto projektu $O(n^3)$.

3 Užívateľská dokumentácia

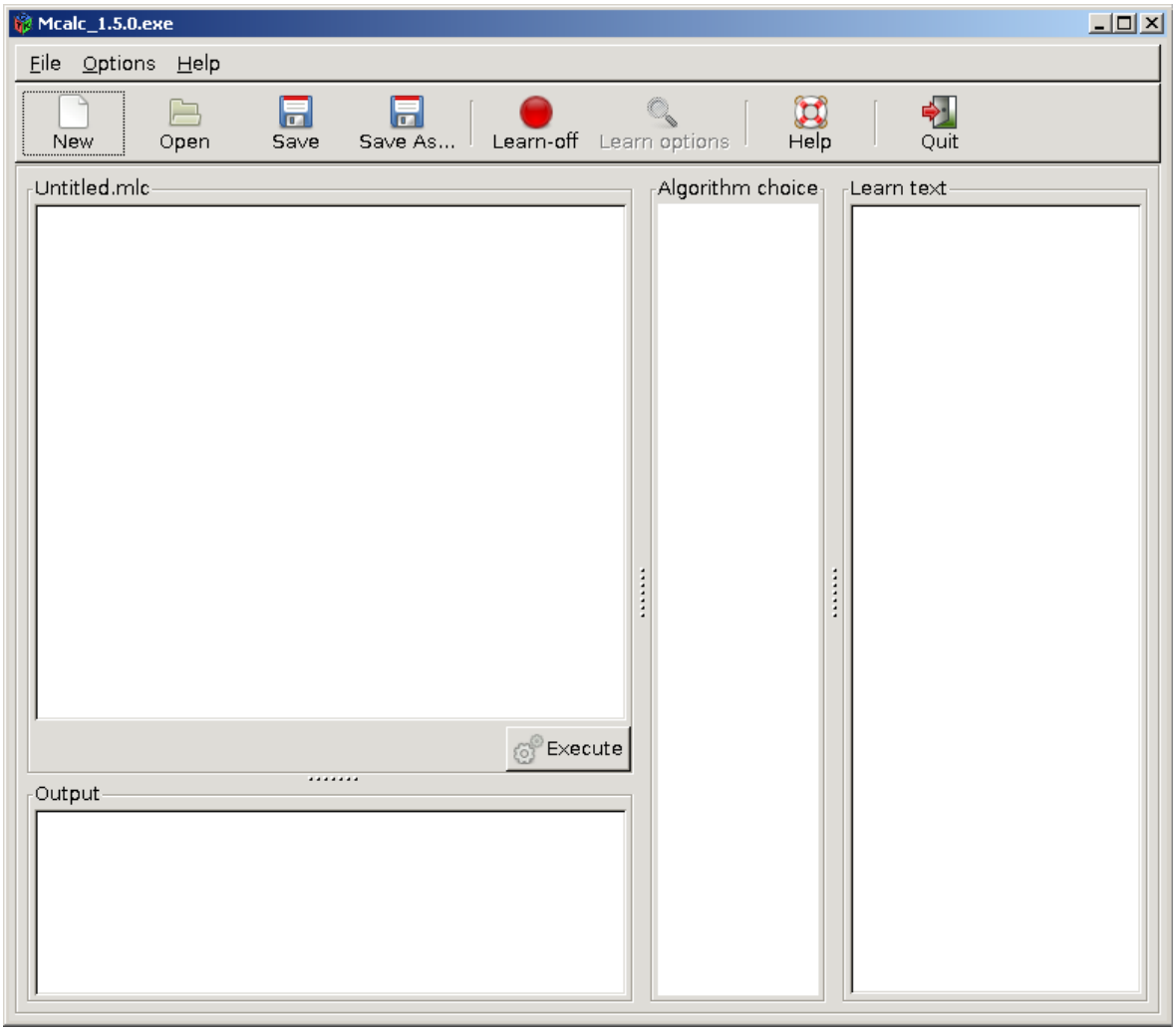
V tejto kapitole sa nachádza podrobný popis práce s programom pre užívateľa. Venuje sa vysvetleniu princípu fungovania grafického užívateľského rozhrania a poskytuje návod na to, ako používať jazyk M.

3.1 Inštalácia a spustenie

Program je vyvinutý pre platformu Windows a nemá žiadne špeciálne systémové požiadavky. Všetky knižnice potrebné na jeho spustenie vrátane knižníc pre beh užívateľského rozhrania sú obsiahnuté v balíčku projektu. Samotný balíček je adresár s názvom Mcalc_1.5.0 a nachádza sa na priloženom CD. Priamo v tomto adresári sa nachádza súbor Mcalc_1.5.0.exe, ktorým sa spúšťa program.

3.2 Ovládanie

Po spustení programu sa zobrazí užívateľské rozhranie, ktoré vyzerá nasledovne:



Obrázok 3.1: GUI programu

V hornej časti hlavného okna sa nachádza menu a ovládací panel. Pomocou menu „File“ a prvých štyroch položiek ovládacieho panelu je možné obvyklým spôsobom otvárať, ukladať a vytvárať nové vstupné súbory. Program zvykne pracovať so súbormi s príponou .mlc, ale prípustné sú akékoľvek textové súbory. Položky menu „Options“ a ďalšie dve tlačítka na ovládacom paneli slúžia na manipuláciu s výukou a práca s nimi je vysvetlené v časti 3.3.1. Položka menu „Help/Contents“ a tlačítko „Help“ na ovládacom paneli zobrazujú návod na používanie programu. Pod ovládacím panelom sa nachádzajú textové okná, ktoré majú nasledovné funkcie (v zátvorke je vždy uvedený popisok k tomuto oknu nachádzajúci sa nad každým z týchto okien):

Vstupné textové okno (na začiatku „Untitled.mlc“) je najväčšie z okien, umiestnené vľavo hore. Sem sa píšu požiadavky na výpočty.

Výstupné textové okno („Output“) sa nachádza vľavo dole priamo pod vstupným textovým oknom. Jeho obsahom sú chybové hlásenia, generované programom a prípadný ďalší výstup špecifikovaný užívateľom.

Okno pre výber algoritmu („Algorithm choice“) je umiestnené napravo od vstupného textového okna. Pri zapnutej výuke sa v ňom nachádza zoznam algoritmov špecifikovaných na vstupe. Kliknutím na konkrétny algoritmus sa vo výukovom textovom okne zobrazí text k tomuto algoritmu.

Výukové textové okno („Learn text“) sa nachádza napravo od okna pre výber algoritmu. Zobrazuje text výuky k jednotlivým algoritmom.

3.3 Práca s programom

Zadávanie požadovaných výpočtov prebieha prostredníctvom vstupného textového okna a realizuje sa pomocou jazyka M, ktorého detailný popis sa nachádza v časti 3.3.2. Výpočet sa spúšťa tlačítkom „Execute“ umiestnenom pod pravým dolným rohom vstupného okna. V prípade, že bol vstup zadaný správne, vo výstupnom textovom okne sa zobrazí hlásenie „Everything is OK.“, prípadne ďalší výstup špecifikovaný užívateľom opäť pomocou jazyka M. Ak sa vo vstupe vyskytli chyby, vo výstupnom textovom okne sa zobrazí informácia o chybe (prípadne viacerých) obsahujúca riadok, na ktorej sa chyba vyskytla a jej popis. Po opravení chýb je možné výpočet opäť spustiť. Nakoľko sa výpočet spúšťa načítaním vstupu zo súboru, ten je treba pred spustením uložiť. Ak sa tak nestane, bude užívateľ o uloženie požiadaný vyskakovacím oknom.

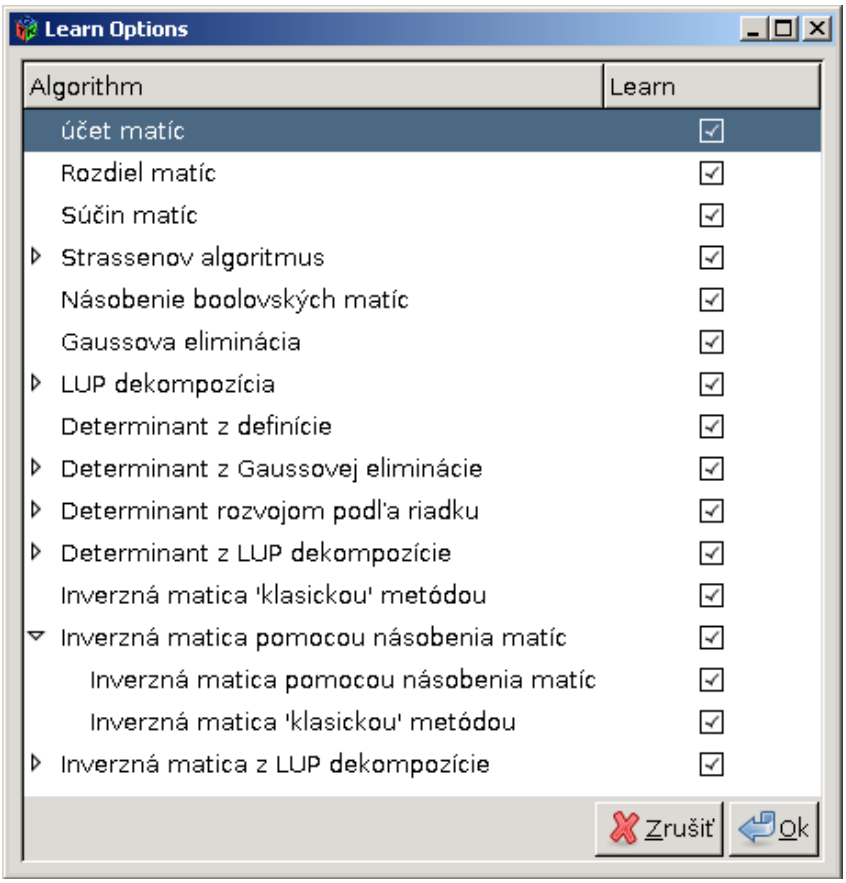
Program je urobený tak, aby mohol jeho výpočetnú časť používať aj anglicky hovoriaci užívateľ. Preto sú názvy súčastí užívateľského rozhrania, ako aj chybové hlásenia v anglickom jazyku. Samotná výuka už prebieha v jazyku slovenskom. Preložením textových súborov s výukou je možné realizovať aj výuku v inom jazyku.

3.3.1 Výuka

Výuka sa spúšťa zvolením položky menu „Options/Learn“, prípadne položkou s názvom „Learn-off“ v ovládacom paneli programu. Od tohto momentu sa po každom spustení

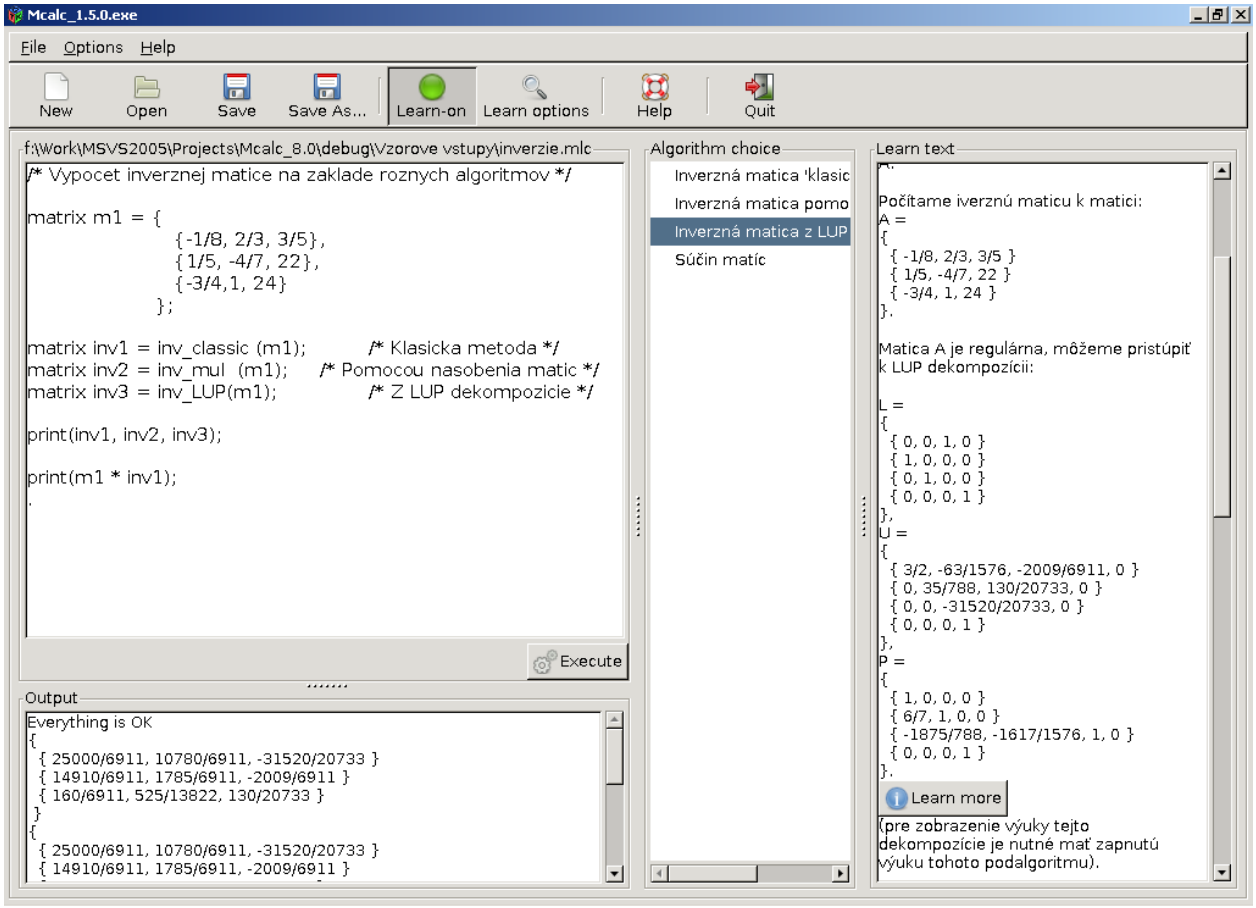
výpočtu zobrazí v okne pre výber algoritmu zoznam algoritmov, ktoré boli zadané na výpočet. Sú zobrazené v poradí, v akom sa vykonávali. Po kliknutí na ľubovolný z nich sa zobrazí text výuky k tomuto algoritmu vo výukovom textovom okne.

Na zvolenie podrobnosti výuky slúži položka menu „Options/Learn options“, prípadne položka s rovnakým menom v ovládacom paneli. Zobrazuje zoznam všetkých algoritmov vrátane podalgoritmov, ktoré obsahujú (Obrázok 3.2). Pod pojmom podalgoritmus rozumieme volanie iného základného algoritmu na získanie medzivýsledku v rámci aktuálneho algoritmu. Zaškrtnutím (odškrtnutím) jednotlivých algoritmov je možné určiť, výuka ktorých algoritmov sa má v priebehu výpočtu generovať a po jeho skončení zobraziť v okne na výber algoritmu. Zaškrtnutím podalgoritmu zaručíme, že vo výuke nadradeného algoritmu sa zobrazí tlačítko na zobrazenie výuky tohto podalgoritmu (pri východzom nastavení sa jeho výuka negeneruje).



Obrázok 3.2: Okno na výber podrobností výuky.

Na príklade na obrázku 3.3 je typická situácia za behu programu pri zapnutej výuke a po spustení výpočtu. V okne na výber algoritmu sa zobrazili zadefinované algoritmy, v tomto prípade výpočet inverznej matice tromi rôznymi spôsobmi a násobenie matíc pre kontrolu výsledkov inverzných funkcií. Zvolený je algoritmus na výpočet inverznej matice z LUP dekompozície, takže jeho výuka je zobrazená vo výukovom textovom okne. Tento algoritmus má naviac zvolenú výuku podalgoritmu „LUP dekompozícia“, preto je vo výukovom textovom okne možnosť kliknúť na tlačítko „Learn more“ pre zobrazenie výuky tohto podalgoritmu. Následne sa môžeme vrátiť k výuke pôvodného algoritmu stlačením tlačítka „Go back“, ktoré sa nachádza na na konci textu výuky. Vo výstupnom textovom okne sa okrem hlásenia „Everything is OK“ oznamujúceho, že vstup bol zadaný správne objavil aj výpis matíc zadaných pomocou funkcie „print“.



Obrázok 3.3: Program po spustení výpočtu so zapnutou výukou.

3.4 Vlastnosti jazyka M

3.4.1 Kľúčové slová, procedúry, funkcie

Pomocou jazyka M je možné programu Mcalc zadať jednotlivé výpočty. Zadávanie funguje ako typický programovací jazyk typu C. Príkazy sa oddeľujú bodkočiarkami (středníkmi), celý program sa končí bodkou. Jazyk M rozlišuje veľké a malé písmená a obsahuje nasledujúce *kľúčové slová* (slová so špeciálnym významom v rámci jazyka):

int – deklarácia premennej typu celé číslo (viď príklad nižšie)

fraction – deklarácia premennej typu zlomok

matrix – deklarácia premennej typu matica,

Procedúry:

print – výpis (ľubovolného počtu) matíc, zlomkov, celých čísel do výstupného textového okna

LUP – výpočet LUP dekompozície matice. Procedúra má 1 argument - maticu, ktorej dekompozíciu počítame. Výsledok sa uloží do matíc s názvami L, U, P, ktoré je možné ďalej v programe využívať ako premenné. Pozor, definovanie vlastných matíc s týmito názvami vedie k chybe (duplicitná definícia).

Procedúry nemajú žiadnu návratovú hodnotu a preto je ich možné volať len samostatne, narozdiel od funkcií, ktoré je možné volať aj v rámci výrazov. Argumenty procedúry (funkcie) sa zadávajú do zátvorky za identifikátor procedúry (funkcie).

Funkcie:

strassen – súčin štvorcových matíc rovnakých rozmerov Strassenovým algoritmom. Funkcia má 2 argumenty – matice, ktoré násobíme. Vráti súčin, ak sú splnené vyššie uvedené podmienky, inak vráti prvý argument a zobrazí varovanie do výstupného textového okna

bool_mul – súčin štvorcových boolovských (obsahujúcich len číslice 0 a 1) matíc rovnakých rozmerov algoritmom štyroch Rusov. Funkcia má 2 argumenty – matice, ktoré násobíme. Vráti súčin, ak sú splnené vyššie uvedené podmienky, inak vráti prvý argument. Pre iné ako boolovské matice vracia nezmyselné výsledky, nakoľko sa na prvky týchto matíc aplikuje logický súčet.

gauss – Gaussova eliminácia matice. Funkcia má 1 argument a to maticu, ktorí chceme eliminovať. Vráti eliminovanú maticu.

det_def – výpočet determinantu matice z definície. Funkcia má 1 argument - maticu, ktorej determinant počítame. Vráti determinant zadanej matice, ak je matica štvorcová, inak vráti 0 a zobrazí varovanie do výstupného textového okna. Za pomoci tohto varovania je možné rozlíšiť, či determinant vyšiel 0 (žiadne varovanie), alebo sa jedná o chybu (varovanie).

det_gauss – determinant matice vypočítaný z Gaussovej eliminácie. Ostatné vlastnosti ako pri 'det_def'.

det_exp – determinant matice vypočítaný rozvojom podla riadku. Ostatné vlastnosti ako pri 'det_def'.

det_LUP – determinant matice vypočítaný z LUP dekompozície. Ostatné vlastnosti ako pri 'det_def'.

inv_classic – výpočet inverznej matice klasickou metódou. Funkcia má 1 argument - maticu, ktorej inverznú maticu počítame. Vráti inverznú maticu, ak bola zadaná matica regulárna, inak vráti zadanú maticu a zobrazí varovanie do výstupného textového okna.

inv_mul – výpočet inverznej matice pomocou násobenia matíc, ostatné vlastnosti ako pri 'inv_classic'

inv_LUP – výpočet inverznej matice z LUP dekompozície. Funkcia má 1 argument - maticu, ktorej inverznú maticu počítame. Vráti inverznú maticu, ak bola zadaná matica regulárna, inak vráti zadanú maticu a zobrazí varovanie do výstupného textového okna.

Príkazy:

Príkazy jazyka M možno rozdeliť na 3 druhy:

- deklarácie a definície premenných v tvare:
  - **názov_typu** identifikátor **;** alebo
  - **názov_typu** identifikátor = výraz **;**
- volanie procedúr a funkcií v tvare:
  - **názov_procedúry** (argumenty) **;**
- príkazy v tvare:
  - **identifikátor** = telo_výrazu **;**

Definícia premennej typu matrix sa uzatvára do množinových zátvoriek, v ktorých sú v ďalších sadách množinových zátvoriek definované jednotlivé riadky matice. Riadok matice sa definuje ako postupnosť celých čísel alebo zlomkov oddelených čiarkami. Premenná typu fraction sa definuje ako čitateľ/menovateľ, kde čitateľ aj menovateľ musia byť v rozsahu -2147483648 až 2147483647. Nakoniec premenná typu int sa definuje ako číslo v uvedenom rozsahu. Telo výrazu je súbor vhodne usporiadaných identifikátorov, operátorov, konštánt a volaní funkcií (3.4.4).

Operátory:

'+' - sčítanie
'-' - odčítanie
'*' - násobenie
'/' - delenie
'=' - priradenie hodnoty (premennej, výsledku výrazu) do premennej.

V prípade maticového násobenia operátorom * je použitý výpočet z definície (2.2.2.1). Kdekoľvek medzi príkazy je možné písať komentáre, ktoré sa začínajú dvojicou znakov /* a končia dvojicou */.

3.4.2 Chybové hlásenia

V prípade chybne zadaného vstupu v jazyku M sa vo výstupnom textovom okne objaví jedno alebo viac chybových hlásení. Tieto hlásenia popisujú charakter chyby, ktorá sa vyskytla na vstupe a sú samovysvetľujúce.

3.4.3 Príklad použitia

Príklad správneho vstupu programu v jazyku M:

```
/* Vypocet determinantu matice na zaklade roznych algoritmov */

matrix m1 = {
            {3, 1, 2},
            {2, 3, 1},
            {1, 2, 3}
           };              /* definicia matice */
```

```
fraction det1 = det_def     (m1);  /* Determinant z definicie */
fraction det2 = det_gauss   (m1);  /* Z gaussovej eliminacie  */
fraction det3 = det_rozvoj  (m1);  /* Rozvojom podla riadku   */
fraction det4 = det_LUP     (m1);  /* Z LUP dekompozicie      */

print (det1, det2, det3, det4); /* vypis zadanych matic */

fraction z5; /* Deklaracia zlomku, tentokrat bez definicie */

z5 = 2 * det_def ({{1,1},{1,1/4}})
       + 22/4331;                  /* Vyraz */

print (z5);
.                          /* koniec programu */
```

Ďalšie príklady vstupu sa nachádzajú na priloženom CD.

4 Programátorská dokumentácia

Táto kapitola sa venuje rozboru implementácie programu, popisuje jej základné vlastnosti a odôvodňuje hlavné rozhodnutia, ktoré boli v priebehu jeho návrhu urobené.

4.1 Platforma, programovací jazyk, vývojové nástroje

Program bol vyvinutý pod operačným systémom Microsoft Windows XP. Je napísaný v jazyku C++. Ako vývojové prostredie bolo použité Microsoft Visual Studio 2005 Professional Edition, verzia 8.0.50727.42. Tvorba grafického užívateľského rozhrania prebieha prostredníctvom technológie GTK, verzia 2.2.19. Pre vývoj jazyka M bol použitý nástroj na vygenerovanie lexikálneho analyzátoru Flex, verzia 2.5.4 a nástroj na vygenerovanie parseru Bison, verzia 2.1. Na prehľadné vytvorenie programátorskej dokumentácie zo zdrojových súborov sa využíva generátor dokumentácie Doxygen, verzia 1.5.5.

Nástroj na tvorbu grafického užívateľského rozhrania GTK bol zvolený vďaka faktu že podporuje operačný systém Windows rovnako ako aj Linuxové systémy, a tak necháva otvorené dvere pre vytvorenie verzie programu pre tieto systémy. Domovská stránka projektu GTK sa nachádza na [4]. Prenositeľné sú aj ďalšie nutné súčasti, Flex a Bison.

4.2 Štruktúra programu

Samotná implementácia programu pozostáva zo štyroch hlavných častí, z ktorých prvou je práca s maticami. Zdrojové súbory pre túto časť začínajú predponou Mcalc_Matrix. Druhou hlavnou súčasťou je spracovanie jazyka M, ktorého zdrojové súbory majú predponu Mcalc_M. V tretej časti sa nachádza je tvorba grafického užívateľského rozhrania, ktorej zdrojové súbory majú predponu Mcalc_GUI. Nakoniec štvrtou hlavnou súčasťou sú nástroje na generovanie výuky, ktoré sú implementované v súboroch začínajúcich predponou Mcalc_Learn. Jednotlivé časti vyhadzujú rôzne typy výnimiek, ktoré sú zachytávané na vhodných miestach, pričom sa vygeneruje príslušná chybová hláška. Všetky deklarácie a definície v rámci programu sú uzavrené do priestoru mien Mcalc. Nasledujúci text sa podrobne venuje jednotlivým popísaným súčastiam.

4.2.1 Malc_Matrix

Hlavnou súčasťou zdrojových súborov začínajúcich predponou Mcalc_Matrix je trieda Matrix, ktorá slúži na reprezentáciu matice. Hodnoty matíc sú uložené v dvojrozmernom poli prvkov triedy Fraction, ktoré je dynamicky vytvorené na základe zadaných rozmerov matice. Základné operácie s maticami ako súčet, súčin a pod. sú definované pomocou preťaženia štandardných operátorov na typ Matrix. Ďalšie metódy aplikujúce rôzne algoritmy na matice sú verejnými metódami triedy Matrix s tým, že funkcie, ktoré súvisia s triedou Matrix ale nehodia sa do nej ako metódy, sú deklarované ako friend funkcie. Sem patria napríklad operátory +,=, … a napr. metóda na spojenie 4 menších matíc do jednej. Keby boli tieto funkcie metódami triedy Matrix, nepracovali by bezprostredne s dátovými položkami matice, na ktorú boli zavolané, a preto sa ako metódy nehodia.

Dôležitým rozhodnutím v návrhu triedy Matrix bolo vytvorenie dvoch funkcií pre algoritmy, ktoré môžu generovať výuku s tým, že jedna z týchto funkcií realizuje daný algoritmus bez generovania výuky, druhá výuku generuje. Tá časť týchto funkcií, ktorá realizuje výpočet je v oboch funkciách rovnaká. Tento prístup bol zvolený kvôli prílišnej neprehľadnosti funkcií s generovaním výuky, kde je kód výpočtu poprekladaný kódom na generovanie výuky, čo výrazne znižuje celkovú čitateľnosť kódu funkcie. Ďalším dôvodom bolo ponechanie možnosti definovať štandardné operácie ako sčítanie a násobenie matíc preťažovaním operátorov, čo by kvôli potrebnému parametru výukových funkcií, kde sa ukladá samotná výuka, nebolo možné. Výhodou je taktiež rýchlejší beh funkcií pri vypnutej výuke, ako keby bola spoločná metóda pre výukový aj nevýukový mód. Nevýhodou tohto prístupu je určitá redundancia kódu a nutnosť upravovať obe verzie funkcie pri prípadnej zmene výpočetnej časti. Samotné generovanie výuky prebieha počas výpočtu algoritmu a je realizované prostredníctvom súčastí popísaných v 4.2.4.

Ďalšou podstatnou črtou triedy Matrix je prístup k chybovým stavom. Vzhľadom k tomu, že matice a operácie na nich zadáva užívateľ, môže dojsť k nekorektným volaniam funkcií, ako napríklad zavolanie sčítania matíc na dve matice rôzneho typu. Táto situácia, rovnako ako aj všetky ďalšie podobného charakteru, je vyriešená tak, že sa výpočet nezrealizuje a vráti sa prvý sčítanec, pričom sa vo výstupnom textovom okne zobrazí upozornenie, že došlo k nekorektnému volaniu. Jedná sa o istý spôsob zotavenia sa z chýb. Zdrojový kód môže totiž obsahovať veľké množstvo nezávislých výpočtov, pričom tie korektne zadané môžu byť úspešne vykonané, aj keď im predchádzajú nejaké nekorektné výpočty.

Trieda Fraction nesie vlastnosti zlomkov a jej prostredníctvom sú uložené prvky v matici. Dátové položky čitateľ a menovateľ sú typu long long. Tento typ prvkov matice bol zvolený pretože je to spôsob ako dosahovať presných výsledkov (narozdiel od prvkov typu double). Nevýhodou tohto prístupu je možné pretečenie dátových položiek, čo má za dôsledok zlyhanie výpočtu, nakoľko je v tomto prípade vyhodená výnimka. Táto výnimka je zachytenNa druhej strane, trieda zvláda väčšinu základných výpočtov a na výukové účely by mala stačiť. Operácie medzi zlomkami sú taktiež implementované pomocou preťaženia základných operátorov.

4.2.2 Mcalc_M.

Zdrojové súbory s predponou Mcalc_M poskytujú nástroj na spracovanie jazyka M. Tento jazyk slúži užívateľovi programu na zadávanie požiadavkov na výpočty. Takýto spôsob zadávania bol zvolený vďaka svojej vysokej robustnosti v poskytovaní možností na prácu s maticami a manipuláciu s výsledkami výpočtov.

Súbor Mcalc_M.lex obsahuje zdrojový kód určený pre program Flex na vygenerovanie analyzátora lexikálnych elementov jazyka M. Pod lexikálnymi elementami jazyka rozumieme postupnosti znakov, ktoré je možné zadať v zdrojovom kóde napísanom v danom jazyku. Patrí sem napríklad celé číslo, identifikátor, množinová zátvorka, ale nepatrí sem hranatá zátvorka alebo 8a, čo nie je ani číslo ani identifikátor, pretože tie musia začínať písmenom. Jednotlivým lexikálnym elementom je možné priradiť konkrétnu hodnotu. Výstupom analyzátora je kód v jazyku C obsahujúci funkciu yylex (), ktorej opakovaným volaním sa postupne získavajú druhy rozpoznaných lexikálnych elementov zdrojového kódu a ich prípadné priradené hodnoty. Táto

funkcia nie je v programe priamo volaná, ale využíva ju parser vytvorený programom Bison. Podrobnejší popis princípov práce s programom Flex sa nachádza na [5].

V súbore Mcalc_M.y sa nachádza kód pre program Bison. Tento kód definuje pravidlá gramatiky, ktorá určuje jazyk M. Na základe týchto pravidiel je určené poradie lexikálnych elementov, ktoré je správne v rámci jazyka M. Definujú napríklad, že postupnosť „int a = 1;“ je správna, ale „int matrix=fraction“ už správna nie je. Taktiež definujú akcie, ktoré budú vykonané pri rozbore (parsovaní) zdrojového kódu. Výstupom programu Bison je kód obsahujúci funkciu yyparse (), ktorej zavolanie vykoná definovaný rozbor vstupného zdrojového kódu vrátane akcií. Kompletný prepis gramatiky jazyka M sa nachádza v prílohe B.

Pomocou týchto akcií, ktoré sa píšu ako C++ kód do zátvoriek za pravidlá gramatiky sú realizované všetky výpočty programu. Pre tento účel sú vytvorené funkcie v súbore Mcalc_M_sem.cpp (deklarácie v Mcalc_M_sem.h) , ktoré počas rozboru zdrojového kódu pracujú s premennými typu Expression (popis nižšie) a vykonávajú v kóde definované operácie a algoritmy na nich. Podrobnejší popis práce s programom Bison sa nachádza na [6].

Jazyk M definuje 3 základné dátové typy a to typ matrix (matica), fraction (zlomok), int (celé číslo). V rámci parseru sú reprezentované pomocou triedy Expression, ktorá môže byť reprezentantom ktoréhokoľvek z nich. Sú na nej definované základné operácie +,-,*,-, pričom pri aplikovaní operátoru na dve inštancie triedy správneho typu vzhľadom k tomuto operátoru (napríklad matica + matica alebo matica * celé číslo) vráti požadovaný výsledok, a naopak pri aplikovaní operátoru na nesprávne typy (matica + celé číslo) vyhodí odpovedajúcu výnimku. Takáto trieda je potrebná na definovanie operácií medzi rôznymi typmi a prípadné ošetrenie nesprávneho užívateľského vstupu. Pomocou tejto triedy sú reprezentované všetky vyššie uvedené dátové typy v rámci parseru.

Poznámky:

- Zdrojové súbory vygenerované programami Flex a Bison obsahujú drobné prehrešky voči norme C++ a pri kompilácii spôsobujú niektoré varovania, ktoré však nie sú problémom a je možné ich nechať bez povšimnutia.
- Pamäť alokovaná funkciou yylex nie je uvoľnená hneď po skončení parsovania, ale až na konci programu (pri opakovanom parsovaní sa používa rovnaká pamäť).

4.2.3 Mcalc_GUI

V súboroch s predponou Mcalc_GUI sa nachádza trieda GUI, ktorá implementuje tvorbu grafického užívateľského rozhrania a prácu s ním. Niektoré súčasti rozhrania sú oddelené do samostatných tried, ktorých inštancie sú dátovými položkami triedy GUI. Je implementovaná za pomoci nástroja GTK.

4.2.4 Mcalc_Learn

Text výuky pre jednotlivé algoritmy je uložený v súboroch s príponou „.lrn“ v adresári „resource“, vo forme textu so špeciálnymi značkami (tagmi), ktoré určujú ďalšie vlastnosti textu. Popis týchto značkami sa nachádza vo vygenerovanej dokumenácii na priloženom CD. Text výuky je v týchto súboroch rozdelený na jednotlivé časti oddelené znakom '@'. Jednotlivé

texty výuky sú uložené v textovej forme v kódovaní UTF-8, ktoré umožňuje písať tieto texty s diakritikou. V priebehu výpočtu algoritmu (teda jeho výukovej verzie) sa jednotlivé časti textu výuky ukladajú do inštancie triedy „Alg_learn_text“ spolu s hodnotami medzivýsledkov výpočtu a prípadne aj textov výuky jeho podalgoritmov, ak je ich výuka zapnutá. Táto inštancia sa neskôr odovzdá triede „Mcalc_GUI_text_buffer_creator“, ktorá interpretuje tagy a pretvorí text do podoby potrebnej pre zobrazenie v GUI. Všetky tieto inštancie sú uložené v globálnej premennej „learn_text“.

4.3 Preklad a zostavenie programu

Na úspešné preloženie zdrojových kódov programu je potrebný kompilátor založený na ISO/ANSI C++ 1998 štandarde. Pre spracovanie súborov Mcalc_M.lex a Mcalc_M.y sú potrebné nástroje Flex (popis inštalácie na [5]) a Bison (popis inštalácie na [6]). Zostavenie projektu je však možné aj bez týchto programov, nakoľko zdrojové súbory vygenerované týmito programami sú priložené v projekte (Mcalc_M_l.cpp pre Flex, Mcalc_M_g.cpp a Mcalc_M_g.hpp pre Bison). Ďalej je potrebný nástroj GTK vrátane ďalších balíčkov, ktoré sú potrebné pre jeho úspešný beh. Popis inštalácie GTK vrátane súčastí sa nachádza na [4]. Verzie programov, ktoré boli použité na zostavenie sa nachádzajú v úvode tejto kapitoly. Nie je však vylúčené, že zostavenie projektu prebehne úspešne aj pri ich starších verziách.

4.4 Ďalšia dokumentácia

Podstatnou súčasťou programátorskej dokumentácie sú html súbory vygenerované Doxygenom na základe komentárov v zdrojovom kóde. Obsahujú popis jednotlivých tried, metód, funkcií, premenných, dátových štruktúr, výčtových typov a konštánt a nachádzajú sa na priloženom CD.

5 Záver

5.1 Zhrnutie

Projekt Maticová kalkulačka poskytuje prostredie pre prácu s vybranými maticovými algoritmami s dôrazom na pochopenie ich princípov prostredníctvom podrobnej výuky. Implementuje 14 maticových algoritmov vrátane výukovej formy ku každému z nich, pričom umožňuje prácu s nimi vo výukovom aj nevýukovom režime. Vo výukovom režime je možné pri každom algoritme sledovať priebeh výpočtu vrátane zobrazenia medzivýsledkov vo zvolenej podrobnosti výuky. Jednotlivé algoritmy sú implementované s väčším ohľadom na pedagogickú názornosť ako na efektívnosť. Projekt sa ďalej snaží poskytovať prívetivé grafické užívateľské rozhranie pre jednoduchú prácu s ním. Ako spôsob na komunikáciu s užívateľom používa svoj vlastný jazyk, ktorým je možné účinne zadávať rôznorodé výpočty. Program je funkčný na platforme Windows, ale bez väčšieho úsilia je možné upraviť ho pre systém Linux. Taktiež necháva priestor pre rozšírenie o ďalšie algoritmy vrátane výukovej časti.

Sada implementovaných algoritmov nebola zvolená najvhodnejšie a zaslúžila by si rozšíriť napríklad o klasický algoritmus na LUP dekompozíciu. Iným námetom na rozšírenie je implementácia matíc s prvkami iného dátového typu, ako je v súčasnosti trieda Fraction (zlomok). Dátové položky zlomkov môžu pri zložitých výpočtoch pretiecť, a teda niektoré výpočty nie je možné vykonať. Matice s prvkami typu double, prípadne triedou implementujúcou ľubovolne dlhé čísla by umožnili úspešné ukončenie aj zložitejších výpočtov.

Literatúra

[1] Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman: *The Design and Analysis of Computer Algorithms,* Addison-Wesley, Reading, 1974.

[2] D. Coppersmith, S. Winograd: *Matrix multiplication via arithmetic progressions*. In Proc. Nineteenth ACM Symp. Theory of Computing, pages 1-6, 1987.

[3] Doc. RNDr. Jiří Tůma, DrSc.: *Gaussova eliminace*,

http://www.karlin.mff.cuni.cz/~tuma/2003/NNLinalg2.pdf

[4] The GTK+ Project,

http://www.gtk.org

[5] Flex (The Fast Lexical Analyzer),

http://www.gnu.org/software/flex/

[6] Bison – GNU parser generator

http://www.gnu.org/software/bison/

[7] V. Strassen: *Gaussian Elimination is Not Optimal*, Numerische Mathematik 13, 1969

[8] Wikipedia, the free encyclopedia

http://www.wikipedia.org

[9] Gene H. Golub, Charles F. Van Loan: *Matrix Computations 3rd edition*, The Johns Hopkins University Press, 1996

[10] Carl D. Meyer, *Matrix Analysis and Applied Linear Algebra*, Society for Industrial and Applied Mathematics, 2000

Príloha A – Obsah priloženého CD

V adresári **/bin** sa nachádza spustiteľná verzia programu, a teda .exe súbor a ďalšie súbory potrebné pre spustenie programu. Zdrojové kódy sa nachádzajú v adresári **/source**. Doxygenom vygenerovaná dokumentácia sa nachádza v adresári **/doc**. Obsahom adresára **/input** sú vzorové vstupné súbory pre program. V adresári **/sln** je projekt pre MS Visual Studio 2005, v ktorom bol program vyvíjaný. V adresári **/text** sa nachádza elektronická podoba tohoto textu vo formáte pdf. Nakoniec adresár **/install** obsahuje inštalačné súbory pre súčasti nutné na zostavenie projektu (Flex, Bison, GTK).

Príloha B – Gramatika jazyka M

Obsahom tejto prílohy je definícia gramatiky, ktorá určuje jazyk M. Sú zapísané vo forme, v akej ich prijíma program Bison. V prvej časti sú direktívou „token“ definované lexikálne tokeny, ktoré rozpoznal generátor syntaktického analyzátora (vyrobený programom Flex). Tieto tokeny slúžia ako terminály gramatiky. V ďalšej časti sú definované jednotlivé neterminály gramatiky. Počiatočný neterminál je „mcalc“.

```
/* literals */
%token   MCALC_TYPE_IDENTIFIER     /* type identifier */
%token   MCALC_IDENTIFIER          /* identifier */
%token   MCALC_UINT                /* unsigned integer */

/* delimiters */
%token MCALC_SEMICOLON             /* ; */
%token MCALC_DOT                   /* . */
%token MCALC_COMMA                 /* , */
%token MCALC_EQ                    /* == */
%token MCALC_COLON                 /* : */
%token MCALC_LPAR                  /*  ( */
%token MCALC_RPAR                  /* ) */
%token MCALC_LSBRA                 /* [ */
%token MCALC_RSBRA                 /* ] */
%token MCALC_LBRACKET              /* { */
%token MCALC_RBRACKET              /* } */
%token MCALC_ASSIGN                /* = */
%token MCALC_SOLIDUS               /* / */

/* grouped operators and keywords */

%token MCALC_OPER_SIGNADD          /* +, - */
%token MCALC_OPER_MUL              /* *, / */

/* ---------------------------- */

mcalc:           commands MCALC_DOT;

commands:                 command
        |        commands MCALC_SEMICOLON command;

command:         /* empty */
```

```
        |       MCALC_TYPE_IDENTIFIER MCALC_IDENTIFIER
        |       MCALC_TYPE_IDENTIFIER MCALC_IDENTIFIER
                        MCALC_ASSIGN expression
        |       MCALC_IDENTIFIER MCALC_ASSIGN expression
        |       MCALC_IDENTIFIER MCALC_LPAR real_parameters
                        MCALC_RPAR;

real_parameters: expression
        |       real_parameters MCALC_COMMA expression;

expression:     terms
        |       MCALC_OPER_SIGNADD terms;

terms:          term
        |       terms MCALC_OPER_SIGNADD;

term:           faktor
        |       term MCALC_OPER_MUL faktor;

faktor:         MCALC_IDENTIFIER
        |       uconst ;
        |       MCALC_IDENTIFIER MCALC_LPAR real_parameters
                        MCALC_RPAR
        |       MCALC_LPAR expression MCALC_RPAR
        |       matrix;

matrix:         MCALC_LBRACKET matrix_rows MCALC_RBRACKET  ;

matrix_rows:    matrix_row
        |       matrix_rows MCALC_COMMA matrix_row;

matrix_row:     MCALC_LBRACKET m_list MCALC_RBRACKET;

m_list:         const
        |       m_list MCALC_COMMA const;

uconst:         MCALC_UINT
        |       fraction;

const:          uconst
        |       MCALC_OPER_SIGNADD MCALC_UINT
        |       MCALC_OPER_SIGNADD fraction;

fraction:       MCALC_UINT MCALC_SOLIDUS MCALC_UINT;
```