

Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Jakub Klímek

Lineárněalgebraický jukebox – vizualizace operací ve vektorových prostorech

Katedra aplikované matematiky

Vedoucí bakalářské práce: RNDr. Jiří Fiala, Ph.D.

Studijní program: Informatika, správa počítačových systémů

2007

Na tomto místě bych rád poděkoval svému vedoucímu ročníkového projektu a bakalářské práce RNDr. Jiřímu Fialovi, Ph.D, za vstřícný přístup a podnětné nápady. Dále bych chtěl poděkovat svým rodičům za to, že mi umožnili plně se soustředit na studium a přípravu této práce.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 17. 5. 2007

Jakub Klímek

Obsah

1	Úvod	5
2	Přehled srovnatelných produktů.....	6
2.1	Linear Algebra Toolkit	6
2.2	JavaScript Linear Algebra	7
2.3	Vector calculator	7
2.4	Podobné kalkulátory	8
3	Uživatelská dokumentace.....	9
3.1	Spuštění appletu.....	9
3.2	Společná nastavení appletů.....	10
3.3	Kalkulačka v tělesech	11
3.4	Vektorová kalkulačka	13
3.5	Seznam klávesových zkratk v kalkulačkách.....	15
3.6	Maticová kalkulačka.....	15
4	Programátorská dokumentace	17
4.1	Instalace a spuštění programu	17
4.2	Struktura programu.....	17
4.3	Balíček Types	18
4.4	Balíček Texts	26
4.5	Balíček Math	27
4.6	Balíček Progress	36
5	Obsah CD	37
	Literatura	38

Název práce: *Lineárněalgebraický jukebox – vizualizace operací ve vektorových prostorech*

Autor: *Jakub Klímek*

Katedra: *Katedra aplikované matematiky*

Vedoucí bakalářské práce: *RNDr. Jiří Fiala, Ph.D.*

e-mail vedoucího: *fiala@kam.mff.cuni.cz*

Abstrakt: *Cílem práce bylo vytvoření sady appletů v Javě, která by sloužila jako doplněk výuky Lineární algebry. Applety umožňují počítání nad tělesy racionálních čísel, reálných čísel, komplexních čísel a nad tělesy Z_p s operacemi sčítání, odčítání, násobení a dělení. Dále je možné počítat s vektory nad tělesy a s jejich operacemi sčítání a odčítání vektorů, násobení a dělení skalárem. Navíc lze počítat souřadnice vzhledem k bázi, zjišťovat lineárně nezávislou podmnožinu vektorů a v prostorech se standardním skalárním součinem také počítat ortogonální projekci na podprostor generovaný množinou vektorů. Předností appletů je jejich snadná ovladatelnost podobná klasické příruční kalkulačce.*

Klíčová slova: *lineární algebra, matice, vektor, těleso, applet*

Title: *Visualization in vector spaces*

Author: *Jakub Klímek*

Department: *Department of Applied Mathematics*

Supervisor: *RNDr. Jiří Fiala, Ph.D.*

Supervisor's e-mail address: *fiala@kam.mff.cuni.cz*

Abstract: *The goal was to create a set of Java applets, which would serve as an addition to the teaching of Linear algebra. These applets make possible to compute with rational numbers, real numbers, complex numbers and Z_p - integers modulo prime p using addition, subtraction, multiplication and division. It is also possible to compute with vectors of elements of those spaces using addition and subtraction of vectors and multiplication and division by scalar. In addition, the vector calculator can figure coordinates of a vector, determine a linearly independent subset of vectors and, when in standard scalar product space, figure an orthogonal projection onto a subspace generated by a set of vectors. The advantage of these applets is their ease of use simulating a classical handheld calculator.*

Keywords: *linear algebra, matrix, vector, space, applet*

Kapitola 1

Úvod

Při tvorbě Lineárněalgebraického jukeboxu jsme se zaměřili na to, aby vznikla pomůcka pro výuku lineární algebry, která se dá snadno použít a je snadno pochopitelná. Nešlo nám tedy o výkon, jakého dosahují profesionální aplikace jako Mathematica[®], Matlab[®] či Maple[™], nýbrž o názornost algoritmů, jednoduchost použití a o možnost provádět výpočty takového rozsahu, se kterými se setkávají studenti při studiu prvního ročníku. Applety pokrývají látku o tělesech, vektorech, vektorových prostorech a maticích. Aby je mohl používat a zkoušet skutečně každý, kdo o ně, třeba náhodou, zavadí, ale zároveň aby byly pohodlně použitelné i pro toho, kdo chce opravdu něco spočítat, jsou programované v Javě a tedy použitelné na všech platformách, na kterých běží Java Runtime Environment, který je dnes standardní výbavou každé pracovní stanice.

Počítat můžeme nad širokou škálou těles, kterými se při výuce zabývá, což jsou \mathbf{C} – komplexní čísla, \mathbf{R} – reálná čísla, \mathbf{Q} – racionální čísla (s automatickým krácením a převodem na společného jmenovatele) a tělesa \mathbf{Z}_p – počítání ve zbytkových třídách prvočísel [1]. Applety mají formu běžných kalkulaček, ať už jde o nejjednodušší počítání s prvky těles, o kalkulačku pro vektory, nebo o maticovou kalkulačku.

Forma běžných kalkulaček zajišťuje kromě vlastního počítání také zaznamenávání historie výpočtu, nabídku řady registrů pro mezivýsledky a ovládání stejné, na které jsme zvyklí z příručních kalkulaček, kde výsledek jednoho výpočtu můžeme okamžitě použít pro výpočet následující, nebo si ho uschovat pro pozdější použití. Applety ale navíc nabízí rozšířené funkce a algoritmy, u kterých zobrazují jejich průběh nebo je umožní krokovat, a jejich výsledky dále používat v jiných výpočtech.

Kapitola 2

Přehled srovnatelných produktů

Při hledání již existujících appletů a kalkulaček na Internetu jsem použil vyhledávání „Linear Algebra Calculator“ v Google. V tomto přehledu se zaměřím na to, co ve kterém produktu chybí a co je naopak šikovné a dobře použitelné.

2.1 Linear Algebra Toolkit

Adresa:	http://www.math.odu.edu/~bogacki/lat/
Autor:	Przemyslaw Bogacki, Old Dominion University
Typ:	Sada skriptů

Tato sada skriptů je přehledně rozdělena podle operací, které ukazují. Maticová část zahrnuje řádkové operace, Gaussovu eliminaci, Gauss-Jordanovu eliminaci, řešení soustavy lineárních rovnic, počítání inverzní matice pomocí řádkových operací a počítání determinantu pomocí řádkových operací. Všechny prvky musí být z tělesa racionálních čísel \mathbf{Q} .

Vektorová část obsahuje test lineární závislosti množiny vektorů, test, zda množina vektorů generuje vektorový prostor [1], nalezení báze vektorového prostoru generovaného množinou vektorů a nalezení báze nulového prostoru matice. Mezi podporované vektorové prostory patří $\mathbf{Q}^1 - \mathbf{Q}^6$, vektorový prostor polynomů nad \mathbf{Q} do stupně 5 a vektorový prostor matic 1×2 , 1×3 , 2×1 , 2×2 , 2×3 , 3×1 a 3×2 taktéž nad tělesem \mathbf{Q} .

Dále jsou k dispozici dva skripty z oblasti lineárního zobrazení. První hledá jádro lineárního zobrazení, druhý hledá obor hodnot (obraz) lineárního zobrazení. Jsou k dispozici stejné vektorové prostory jako v minulé části.

Používání skriptů je jednoduché, skládá se ze zadání rozměrů matice, zadání rozměru a počtu vektorů, výběru vektorových prostorů, vlastního zadání matice a vektorů a zobrazení výsledku včetně přehledného průběhu výpočtu s popisem postupu.

2.2 JavaScript Linear Algebra

Adresa:	http://mkaz.com/math/line_alg.html
Autor:	Marcus Kazmierczak
Typ:	Maticová kalkulačka

Toto je sada tří jednoduchých appletů, které jsou ale použitelné jen pro testovací výpočty.

Matrix Calculator je Javová aplikace, kterou je nutné stáhnout a zkompileovat. Nabízí práci se dvěma maticemi, jejich součet, součin zleva i zprava, inverzi, transpozici, determinant a adjugovanou matici. Výsledek se dá použít pouze ručním zkopírováním výsledné matice na vstup.

Linear Algebra Calculator je JavaScript umožňující práci se dvěma maticemi 3x3 a nabízející stejné funkce jako Matrix Calculator. Umí navíc přesouvat matice mezi sebou, takže se výsledek dá použít i pro jiný výpočet.

Linear System of 3 Equations je také JavaScript, který umožňuje řešit soustavu rovnic $Ax = b$, kde A je matice 3x3.

2.3 Vector Calculator

Adresa:	http://wims.unice.fr/wims/
Autor:	Gang Xiao
Typ:	Vektorový kalkulátor

Tento applet umožňuje zjistit lineární závislost zadaných vektorů, spočítat lineární kombinaci, ortogonální doplněk, zobrazit vektory v \mathbf{R}^2 a \mathbf{R}^3 , spočítat skalární součin a pro vektory v \mathbf{R}^3 i vektorový součin. Pracuje s reálnými a komplexními čísly, vstupem je 2 až 16 vektorů.

2.4 Podobné kalkulátory

Na následujících adresách jsou k dispozici kalkulátory v různých provedeních, které jsou funkčně podobné již zmíněným:

Vector Calculator	http://comp.uark.edu/~jgeabana/java/VectorCalc.html
Vector Calculator	http://wims.unice.fr/wims/
Vector Arithmetic Applet	http://www.pa.uky.edu/~phy211/VecArith/
Vector Applets Worksheet	http://www.slu.edu/classes/maymk/SketchpadApplets/AddVectorsWS.html
Linear Algebra Calculator	http://www.compute.uwlax.edu/lin_alg/
MCALCDOS	http://www.angelfire.com/space/matrixcalc/
Matrix Calculator	http://wims.unice.fr/wims/

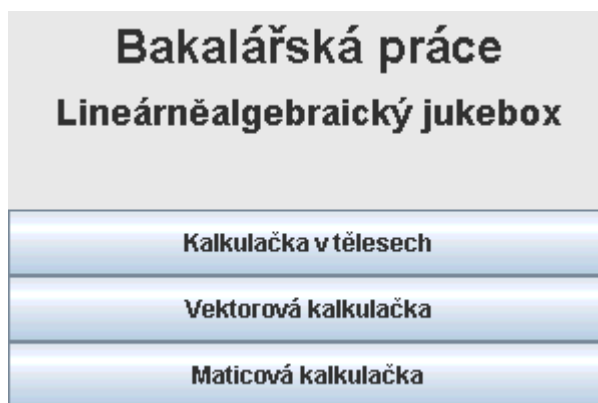
Kapitola 3

Uživatelská dokumentace

Všechny applety pro počítání mají společné nastavování vlastností a typů těles. K dispozici je těleso \mathbf{Q} (racionální čísla), tělesa \mathbf{Z}_p pro p prvočíslo z množiny $\{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31\}$ a tělesa \mathbf{R} (reálná čísla) a \mathbf{C} (komplexní čísla), pro která je možno nastavit, zda se mají zobrazovat v exponentovém tvaru a na kolik platných cifer se budou zobrazovat, a jaké číslo je již považováno za nulové.

3.1 Spuštění appletu

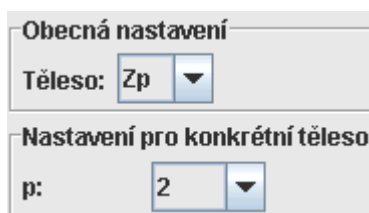
Sada appletů se nachází na přiloženém CD a na adrese <http://kubik.php5.cz/la.php> a k jejímu spuštění je třeba mít nainstalován Java Runtime Environment 6 [2].



Obrázek 1 - Úvodní menu

Zobrazí se menu, kde si uživatel může vybrat, kterou část programu spustí.

3.2 Společná nastavení appletů



Obecná nastavení


Těleso: Zp ▼

Nastavení pro konkrétní těleso

p: 2 ▼

Obrázek 2 - Nastavení Zp

Nastavení společná pro všechny applety jsou ta, která se týkají těles a zobrazení čísel. Pro těleso **Zp** nastavujeme prvočíslo p , v jehož zbytkových třídách budeme počítat.



Obecná nastavení

Těleso: C ▼

Nastavení pro konkrétní těleso

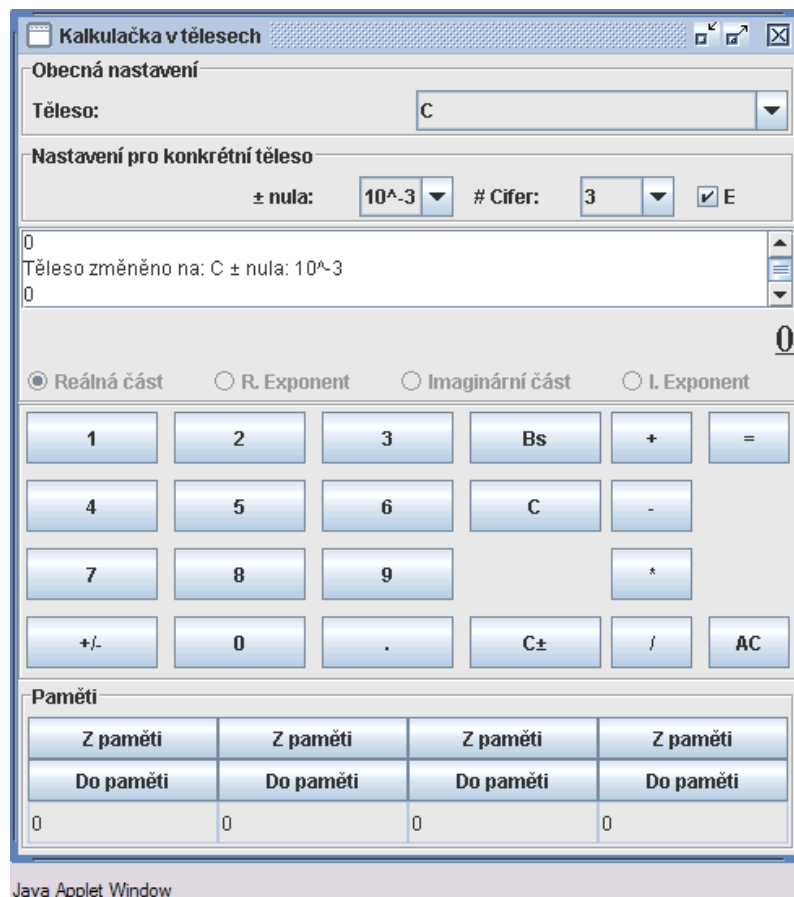
± nula: 10^{-3} ▼ # Cifer: 3 ▼ E

Obrázek 3 - Nastavení R a C

U těles **R** a **C** nastavujeme, jak malé číslo již bude považováno za nulové, na kolik platných cifer zobrazujeme a zda chceme zobrazovaná čísla zobrazovat v exponentovém tvaru s přihlédnutím k počtu platných cifer ($4e+4$) či ve tvaru, v jakém reálná čísla zobrazuje Java (40000.0).

3.3 Kalkulačka v tělesech

Tento applet se chová jako obyčejná příruční kalkulačka, ovšem s čísly z různých těles a s příslušným nastavením.



Obrázek 4 - Kalkulačka v tělesech

V kterékoliv chvíli lze aktuálně zadávané číslo uložit do jedné z pamětí pomocí příslušného tlačítka „Do paměti“, nebo pomocí klávesové zkratky $s + \langle \text{číslo paměti} \rangle$. Ve spodním okénku se zobrazuje aktuální obsah paměti. Číslo v paměti lze použít při výpočtu příslušným tlačítkem „Z paměti“, nebo klávesovou zkratkou $r + \langle \text{číslo paměti} \rangle$.

Při počítání v tělese \mathbf{Z}_p pro $p \in \{2, 3, 5, 7\}$ jsou přebytečné číslice schovány, pro $p \geq 11$ se při zadání čísla nepatřícího do tělesa použije zadané číslo modulo p .

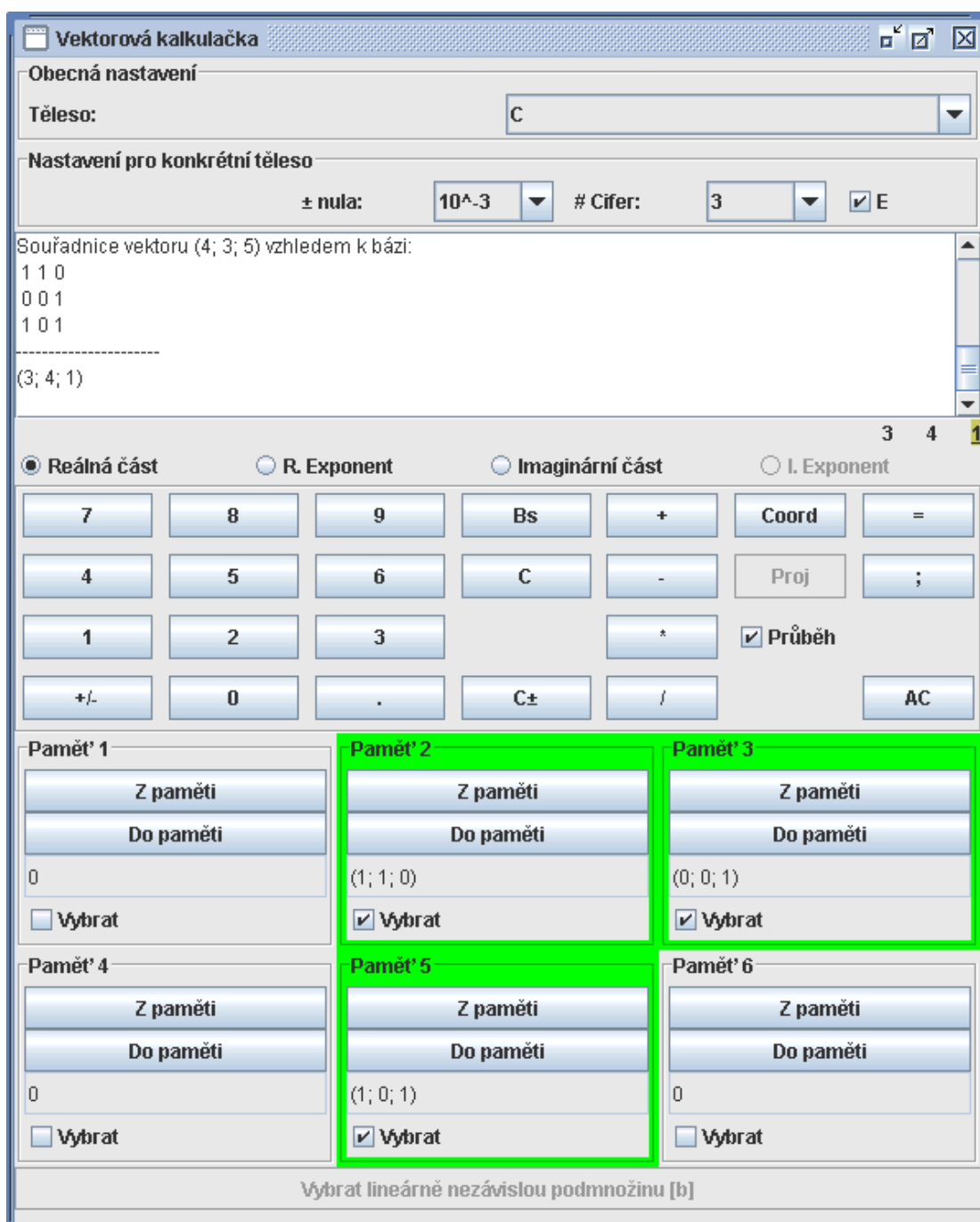
V průběhu zadávání čísla je možné se kdykoliv přepnout do jakékoliv jeho části pomocí tlačítek pod oblastí zadávání (pro tělesa \mathbf{Q} , \mathbf{R} a \mathbf{C}), nebo pomocí kláves e – přepínání mezi exponentovou částí a základem, i – přepínání mezi imaginární a

reálnou částí komplexního čísla, nebo „.“ – přepínání mezi čitatelem a jmenovatelem racionálního čísla. Právě zadávaná část je zvýrazněna podtržením.

Průběh výpočtu je zaznamenáván do historie v horní části okna. Při přepnutí mezi tělesy se aktuální výsledek převede do nového tělesa, pokud je to nějakým způsobem možné (například při přepnutí z **C** do **R** se zachová reálná složka čísla). Tuto kalkulačku lze ovládat jak klávesnicí, tak i myší.

3.4 Vektorová kalkulačka

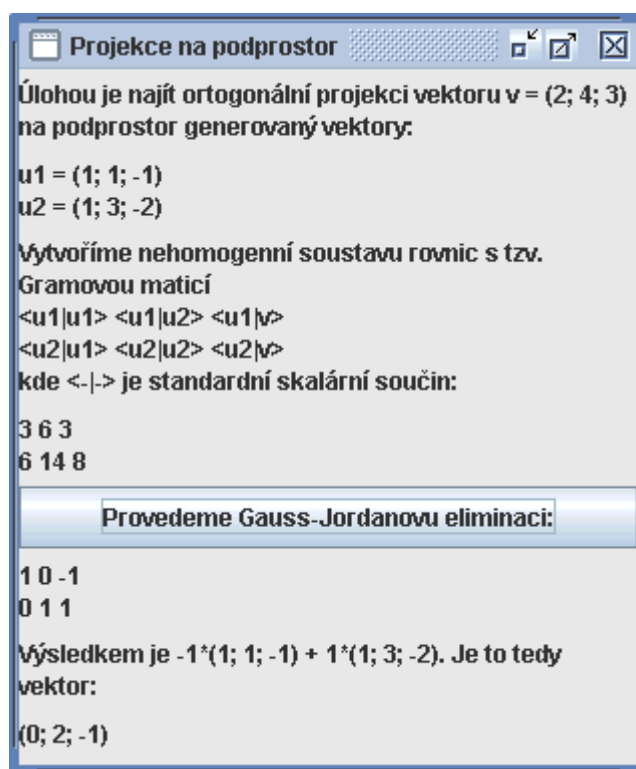
Tento applet je podobný Kalkulačce v tělesech s tím rozdílem, že funguje pro vektory a jejich operace (sčítání a odčítání vektorů, násobení a dělení skalárem). Jednotlivé složky vektoru se oddělují středníkem, dimenze je omezena na 7¹ z toho důvodu, že delší vektory by byly nepřehledné.



Obrázek 5 - Vektorová kalkulačka

¹ Tuto konstantu lze jednoduše změnit viz 4.3.7 Typ VectNum, konstanta MAXDIM

Pro test lineární závislosti, výpočet ortogonální projekce a výpočet souřadnic vzhledem k bázi stačí zaškrtnout, které paměťové pozice se budou používat jakožto množina vektorů nebo báze. Pokud vybraná množina vektorů je lineárně závislá, bude obarvena červeně a bude možné stisknout tlačítko „Vybrat lineárně nezávislou podmnožinu“. Pokud bude vybraná množina generovat podprostor nějakého vektorového prostoru, bude obarvena žlutě, a pokud se bude jednat o vektorový prostor se standardním skalárním součinem, bude možné tlačítkem „Proj“ spočítat ortogonální projekci vektoru na tento podprostor [1]. Pokud bude vybraná množina báze, bude obarvena zeleně a tlačítkem „Coord“ bude možné spočítat souřadnice vektoru vzhledem k této bázi.



Obrázek 6 - Ukázka průběhu algoritmu

Ke všem třem těmto algoritmům se vztahuje volba „Průběh“. Pokud je zaškrtnuta, pak se při výpočtu algoritmů zobrazí jejich podrobný průběh v samostatném okně. Tlačítko v okně průběhu spustí Maticovou kalkulačku v režimu krokování Gauss-Jordanovy eliminace příslušné matice.

Souřadnice vektoru vzhledem k bázi a ortogonální projekce na podprostor jsou zadány do kalkulačky a dají se použít pro další výpočty.

Pokud se uživatel pokusí vložit do kalkulačky vektor z paměti, který má jiný typ, než právě zvolené těleso, bude převeden dle možností na aktuální těleso. Pokud budou do množiny vektorů vybrány vektory různé délky, budou doplněny nulami na nejvyšší společnou dimenzi.

Pokud bude pro algoritmy hledání souřadnic vzhledem k bázi nebo hledání ortogonální projekce na podprostor zadán vektor větší dimenze, než jsou vektory v bázi, pro výpočet se použije vektor zkrácený na příslušnou délku.

3.5 Seznam klávesových zkratk v kalkulačkách

<i>Klávesová zkratka</i>	<i>Význam</i>
0 - 9	Zadávání čísel
p	+/-
o	Komplexní +/- (komplexně sdružené číslo)
. nebo ,	Desetinná část
i	Přesun do imaginární části
e	Přesun do exponentu
+ - * /	Aritmetické operace
Backspace	Smaže poslední symbol
C	Smaže číslo
A	Smazat vše
Enter	=
;	Další prvek
s + číslo	Uložit do paměti č.x
r + číslo	Použít paměť č.x
z	Spočítat souřadnice k bázi
x	Spočítat ortogonální projekci
b	Vybrat lineárně nezávislou podmnožinu

3.6 Maticová kalkulačka

V rámci ročníkového projektu vznikla také Maticová kalkulačka, jejímž autorem je Jakub Stárka. Matice se zadává stejně jako vektor, nový řádek se vkládá stiskem klávesy **n**. Mezi operace použitelné v režimu kalkulačky patří

Gaussova eliminace, Gauss-Jordanova eliminace, nalezení adjugované matice, Choleského rozklad matice, spočítání determinantu, transpozice a inverzní matice.

Druhá část kalkulačky umožňuje krokování jednotlivých algoritmů. K dispozici je Gauss-Jordanova eliminace, násobení dvou matic, hledání inverzní matice, hledání inverzní matice pomocí matice adjugované, spočítání kořenů matice pomocí Cramerova pravidla, spočítání determinantu Gauss-Jordanovou eliminací a spočítání determinantu podle definice. Pro krokování se dají použít matice uložené v pamětech, musí však splňovat podmínky, za kterých je možno algoritmy aplikovat. Například pro násobení matic je třeba, aby počet sloupců první matice byl roven počtu řádků druhé matice. Krokovat lze ručně nebo automaticky se zvoleným časovým intervalem mezi kroky.

Třetí část kalkulačky umožňuje volné úpravy. Uživatel si vybere matici z paměti a může na ni aplikovat tři elementární řádkové úpravy: násobení řádku číslem, prohození dvou řádků a přičtení k -násobku jednoho řádku k druhému.

Kapitola 4

Programátorská dokumentace

4.1 Instalace a spuštění programu

Applety jsou v archivu LA.jar a lze je umístit na webovou stránku dvojitým způsobem. Buď se budou spouštět přes společné menu, kdy si každý kalkulátor otevře své okno, pak se na stránku umístí applet LA.java, nebo se jednotlivé kalkulátory umístí přímo na plochu stránky. Pro tento případ je třeba vložit applety Calc2.java – kalkulačka v tělesech a VCalc.java – vektorová kalkulačka a nastavit jim odpovídající velikosti.

Pokud je vyžadována funkce Maticové kalkulačky „Uložit a nahrát práci“, je třeba pracovat s digitálně podepsanou verzí archivu LA.jar.

4.2 Struktura programu

Program je rozdělen to čtyř balíčků:

- Balíček *Types* obsahuje hierarchii tříd čísel a třídu pro vektory.
- Balíček *Texts* obsahuje třídy s texty, které se v programu vyskytují. V případě změny jazyka stačí přepsat texty v tomto balíčku a zkompileovat s nimi program.
- Balíček *Math* obsahuje hlavní třídy programu zajišťující uživatelské rozhraní a výpočetní algoritmy.
- Balíček *Progress* obsahuje třídy pro zobrazení průběhu algoritmů.

4.3 Balíček Types

Balíček Types obsahuje hierarchii tříd čísel a třídu pro vektory.

4.3.1 Typ Num

Tato abstraktní třída je předkem všech tříd reprezentujících typ čísla. Určuje interface tříd čísel a implementuje univerzální metody Parse() a Op() a metody getTypeFromIndex() a getIndexFromType().

Type t

Určuje typ třídy (NumR, NumC...).

public Num plus(Num n)

Číslo k sobě přičte n a vrátí výsledek jako nové číslo.

public Num mult(Num n)

Číslo se vynásobí n a vrátí výsledek jako nové číslo.

public String str(int x, boolean e)

Číslo se vypíše do Stringu (Pokud jde o R nebo C tak na x platných cifer, e udává, zda v exponentovém tvaru).

public boolean IsNull()

Vrátí, zda je číslo nulové.

public Num inv_plus()

Vrátí číslo, které je inverzem vzhledem ke sčítání.

public Num inv_mult()

Vrátí číslo, které je inverzem vzhledem k násobení.

public Num ToC()

Vrátí výsledek převodu čísla na komplexní s výchozí hranicí nulovosti.

public Num ToC(double z)

Vrátí výsledek převodu čísla na komplexní s hranicí nulovosti z .

public Num ToQ()

Vrátí výsledek převodu čísla na racionální.

public Num ToZp(int p)

Vrátí výsledek převodu čísla na Zp s prvočíslem p .

public Num ToR(double z)

Vrátí výsledek převodu čísla na reálné s hranicí nulovosti z .

public Num ToR()

Vrátí výsledek převodu čísla na reálné s výchozí hranicí nulovosti.

public boolean isSameType(Num n)

Vrací, zda je n stejného typu jako číslo, na němž je metoda vyvolána.

public static Num Parse(Type t, int p, double zero, String s)

Vrátí číslo typu t , pokud se parsování povedlo. Jinak vrátí číslo typu *Err*.

public static Num Op(Num n1, int OpIndex, Num n2)

Provede operaci OpIndex (0-"+" 1-"-" 2-"*" 3-"/") a vrátí výsledek $n = n1 \text{ Op } n2$.

public static Type getTypeFromIndex(int index)

Vrátí typ čísla podle hodnoty *index* - index typu v poli *textGlobal.telesa*.

public static int getIndexFromType(Type y)

Vrátí index typu čísla y v poli *textGlobal.telesa*.

4.3.2 Typ NumR

Třída NumR reprezentuje reálné číslo. Je potomkem třídy Num.

private double x

Interní uložení hodnoty reálného čísla.

private double zero

Hodnota nulovosti pro toto číslo.

public static String[] Presnost

Pole stringů, které se dá použít při tvorbě ComboBoxu pro volbu počtu platných cifer.

public NumR(double y, double z)

Konstruktor, y je hodnota reálného čísla, z je hodnota nulovosti.

public NumR(double y)

Konstruktor, y je hodnota reálného čísla, hladina nulovosti se použije výchozí.

public static String str(double d, int p)

Vrátí Stringovou reprezentaci čísla d na p platných cifer.

public double DoubleValue()

Vrátí double reprezentaci čísla.

private static double round(double val, int places)

Vrátí číslo val zaokrouhleno na places desetinných míst. Používá pro vlastní zaokrouhlení 2x metodu round2: s o 2 větší přesností a pak s požadovanou přesností. Řeší tím problém nepřesnosti reprezentace reálného čísla typem double.

public Num ToC()

Vrátí komplexní číslo s nulovou imaginární složkou.

public Num ToC(double z)

Vrátí komplexní číslo s nulovou imaginární složkou a hodnotou nulovosti z.

public Num ToQ()

Vrátí racionální číslo odpovídající zaokrouhlení na setiny.

public Num ToZp(int p)

Vrátí číslo ze Zp odpovídající zaokrouhlení na celé číslo a modulo p.

4.3.3 Typ NumC

Třída NumC reprezentuje komplexní číslo. Je potomkem třídy Num.

private double re, im

Interní uložení hodnoty reálné a imaginární složky.

private double zero

Hodnota nulovosti pro toto číslo.

public static String[] Presnost

Pole stringů, které se dá použít při tvorbě ComboBoxu pro volbu počtu platných cifer.

public NumC(double r, double i)

Konstruktor, r je hodnota reálné složky čísla, i je hodnota imaginární složky čísla, hodnota nulovosti se použije výchozí.

public NumC(double r, double i, double z)

Konstruktor, r je hodnota reálné složky čísla, i je hodnota imaginární složky čísla, z je hodnota nulovosti.

public boolean ReNull()

Vrátí, zda je reálná složka nulová.

public boolean ImNull()

Vrátí, zda je imaginární složka nulová.

public Num ToR()

Vrátí reálné číslo odpovídající reálné složce.

public Num ToR(double z)

Vrátí reálné číslo odpovídající reálné složce a hodnotou nulovosti z.

public Num ToQ()

Vrátí racionální číslo odpovídající zaokrouhlení reálné složky na setiny.

public Num ToZp(int p)

Vrátí číslo ze Zp odpovídající zaokrouhlení reálné složky na celé číslo a modulo p.

public Num CPLX_sdruzene()

Vrací komplexně sdružené číslo.

4.3.4 Typ NumZp

Třída NumZp reprezentuje číslo z tělesa Zp. Je potomkem třídy Num.

private int p

Prvočíslo, přes které se počítá modulo.

private int z

Interní uložení hodnoty celého čísla.

public static String[] Primes

Pole stringů, které se dá použít při tvorbě ComboBoxu pro volbu počtu prvočísla.

public NumZp(int prime, int x)

Konstruktor, x je hodnota celého čísla, prime je prvočíslo.

public Num ToC()

Vrátí komplexní číslo s nulovou imaginární složkou.

public Num ToC(double z)

Vrátí komplexní číslo s nulovou imaginární složkou a hodnotou nulovosti z.

public Num ToR()

Vrátí reálné číslo.

public Num ToR(double z)

Vrátí reálné číslo s hodnotou nulovosti z.

public Num ToQ()

Vrátí racionální číslo.

public Num ToZp(int p)

Vrátí číslo modulo p.

4.3.5 Typ NumQ

Třída NumQ reprezentuje racionální číslo. Je potomkem třídy Num.

private int num,den

Interní uložení hodnoty čitatele a jmenovatele

public NumQ(int x, int y)

Konstruktor, x je hodnota čitatele, y je hodnota jmenovatele

public Num ToC()

Vrátí komplexní číslo(num/den) s nulovou imaginární složkou

public Num ToC(double z)

Vrátí komplexní číslo(num/den) s nulovou imaginární složkou a hodnotou nulovosti z

public Num ToR()

Vrátí reálné číslo(num/den)

public Num ToR(double z)

Vrátí reálné číslo(num/den) s hodnotou nulovosti z

public Num ToZp(int p)

Vrátí zaokrouhlené číslo (num/den) modulo p

private int nsd(int x, int y)

Algoritmus pro největší společný dělitel

private int nsn(int x, int y)

Algoritmus pro nejmenší společný násobek

4.3.6 Typ NumErr

Typ NumErr reprezentuje chybu. Je potomkem třídy Num.

private String s

Interní uložení textu chyby

public NumErr()

Konstruktor, chyba bude označena jako nspecifikovaná

public NumErr(String x)

Konstruktor, chyba bude označena jako x

public Num plus(Num r)

Vrátí r, pokud je r chybového typu, jinak vrací sebe

public Num mult(Num r)

Vrátí r, pokud je r chybového typu, jinak vrací sebe

public String str(int x, boolean e)

Vrátí text této chyby

public boolean IsNull()

Vrátí true.

public Num inv_plus()

Vrátí sebe

public Num inv_mult()

Vrátí sebe

public Num ToC()

Vrátí sebe

public Num ToC(double z)

Vrátí sebe

public Num ToQ()

Vrátí sebe

public Num ToZp(int p)

Vrátí sebe

public Num ToR(double z)

Vrátí sebe

public Num ToR()

Vrátí sebe

public boolean isSameType(Num n)

Vrací, zda je n typu NumErr

4.3.7 Typ VectNum

Třída VectNum reprezentuje vektor typu Num. Není potomkem třídy Num.

private Num[] n

Interní uložení vektoru Num.

private int dim

Dimenze vektoru.

private Num.Type t

Typ prvků vektoru.

private int e_prime

Prvočíslo, pokud je typ Zp.

private double e_zero

Hodnota nulovosti, pokud je typ NumR nebo NumC.

public static final int MAXDIM = 7

Horní mez dimenze.

public static final int MINDIM = 1

Dolní mez dimenze.

public VectNum(int d, Num.Type t2, int p, double zero)

Konstruktor, vytvoří vektor dimenze d, typu t2, prvočíslo nastaví na p a hodnotu nulovosti na zero.

public VectNum()

Konstruktor, který vytváří prázdný vektor dimenze 0. Používá se pro hlášení chyb.

public int getDim()

Vrátí aktuální dimenzi vektoru.

public Num.Type getType()

Vrátí typ vektoru.

public Num getElem(int i)

Vrátí i-tý prvek vektoru.

public boolean setElem(int i, Num co)

Nastaví i-tý prvek vektoru, pokud se typ prvku neshoduje s typem vektoru, na jeho místo vloží prvek typu NumErr. Vrací, zda se přidání povedlo.

public static VectNum ParseVectNum(Num.Type t2, int p, double zero, String s)

Vrátí vektor typu t2, prvočíslo nastaví na p, hodnotu nulovosti na zero. s je vstupní řetězec.

public String str(int x, boolean e)

Vrací stringovou reprezentaci vektoru (e zda má být zobrazován exponentový tvar u NumR a NumC, x je počet platných cifer u NumR a NumC).

public void setDim(int d)

Nastaví dimenzi. Přebytečné prvky odstraní, prázdná místa zaplní nulami.

public VectNum plus(VectNum v)

Provede sčítání po složkách, při sčítání vektorů různé délky vrátí vektor větší dimenze.

public VectNum inv_plus()

Vrátí vektor opačný vzhledem ke sčítání.

public VectNum mult(Num n)

Každý prvek vektoru vynásobí n.

public VectNum ToX(...)

Zkonvertuje vektor do příslušného typu.

public static int getMaxDim(VectNum V1, VectNum V2)

Vrátí větší dimenzi z dimenzí V1 a V2.

public static Num stdSkalarSoucinR(VectNum V1, VectNum V2)

Vrací standardní skalární součin dvou reálných vektorů.

public static Num stdSkalarSoucinC(VectNum V1, VectNum V2)

Vrací standardní skalární součin dvou komplexních vektorů.

4.4 Balíček Texts

Balíček Texts obsahuje sady textů, které jsou použité v programu.

- Třída *Err* obsahuje texty používané jako chybová hlášení (například pro typ *NumErr*).
- Třída *textGlobal* obsahuje texty společné všem appletům.
- Třída *textHistory* obsahuje texty vypisované do okna Historie.
- Třída *textTester* obsahuje texty speciální pro applet *Tester*.

- Třída *textVCalc* obsahuje texty speciální pro applet *VCalc*.

4.5 Balíček Math

Balíček Math obsahuje všechny applety a Číselník, který je společný všem.

4.5.1 LA – úvodní stránka

Tento applet je základní „rozcestník“, obsahuje nadpis a tlačítka pro spuštění jednotlivých částí. Pro přidání nové části stačí přidat tlačítko, rozšířit *GridLayout* a přidat tlačítku jako *ActionListener this*. V metodě *ActionPerformed()* pak stačí přidat obsluhu nového tlačítka.

4.5.2 Ciselnik – Číselník, součást každého appletu

Třída Číselník rozšiřuje třídu *JPanel*, dá se tedy přidat do jakéhokoliv appletu. Slouží pro zadávání jednoho čísla (jednoho prvku Z_p , R , Q , C).

final int NUMBERS = 10

Počet tlačítek s čísly, typicky 10.

final int OTHERS = 4

Počet jiných tlačítek.

final int ROWS = 4

Počet řádek.

final int COLS = 5

Počet sloupců.

final int RBUTS = 4

Počet radiobuttonů pro části čísla.

final int MOC = 1000

Konstanta pro nekonečno.

final int CASTI = 4

Počet částí čísla.

JLabel lbl

Label, do kterého budu zadávat číslo.

JRadioButton[] rbCast

Radiobuttony pro výběr části.

Num.Type Current_Type

Typ aktuálního zobrazení/zadávaného čísla.

int Current_p = 0

p pro Zp.

String[] Casti

Moje reprezentace čísla (po částech).

GridLayout GL

Layout číselníku.

JButton[] btnNumbers, btnOthers

Tlačítka – čísla a ostatní.

public boolean clear

Stisk tlačítka s číslem znamená začátek nového čísla – toto nastavují applety například po provedené operaci.

boolean SkipRbEval = false

stavová proměnná používaná funkcí MyKeyPressed_C.

boolean Q_default_1 = false

zda je zadána defaultní jednička ve jmenovateli.

boolean IgnoreKey = false

Zda se blokuje vstup z klávesnice (např. pro práci s pamětí v kalkulačce).

public Ciselnik(JLabel label, JRadioButton[] rbs, Num.Type t, int p)

Konstruktor – *label* je JLabel, do kterého budu vypisovat číslo, *rbs* jsou Radiobuttony, které budu používat pro volbu editované části čísla, *t* je typ zadávaného čísla, *p* je prvočíslo pro Zp.

public void keyTyped(KeyEvent e)

Funkce ošetřující stisk klávesy (implementace *KeyListener*).

private void changeDisplay()

Obstarává změnu zobrazení číselníku (změna tělesa).

public void MyKeyPressed(int OpCode)

Tato funkce rozděluje obsluhu stisku klávesy funkcím *MyKeyPressed_X()* kde X je R, Zp, Q, C... zde se musí přidat případné další těleso.

public void ClearNow()

Způsobí vymazání právě zadávaného čísla (stisk C).

public void SetType(Num.Type t, int p)

Nastaví těleso (a prvočíslo) již existujícímu číselníku.

public void setRbsEnabled(boolean b)

Metoda nastavuje všem *RadioButton*ům pro volbu části čísla, zda jsou nebo nejsou povoleny (například po stisku = by být neměly).

private void setRbsC()

Nastavuje „povolenost“ *RadioButton*ům v případě komplexních čísel tak, aby to mělo smysl v závislosti na aktuálních částech.

private void SetVars()

Rozděluje se dle těles na *SetVars_X()*, slouží k nastavení stavových proměnných při zadání textu z vnější (*SetLabel*, *SetText*).

private String getSubstr_X(int cast, String s)

Metoda pro každý typ čísla zvlášť, vrací string s částí *cast* stringu *s*, ve kterém je číslo v textovém tvaru.

public String getText()

Tato metoda slouží appletům pro získání textu právě zadávaného čísla.

public void setText(String t)

Slouží appletům pro zadání textu číselníku.

private void UpdateLabel()

Vypíše do labelu číselníku aktuální číslo (vytváří ho vždy z částí).

private void Parse(String r)

String r rozdělí na části pro reprezentaci čísla v číselníku.

public void itemStateChanged(ItemEvent e)

Obsluhuje kliknutí na RadioButtony částí čísla, opět zde záleží na typu čísla a proto je zde nutné přidat případný nový typ.

public void setLabel(JLabel newlbl)

Nastaví číselníku nový Label.

private String deHTML(String s)

Odebere ze stringu s veškeré HTML tagy.

private int getSelectedRb()

Vrátí index aktuálně zaškrtnutého RadioButtону částí.

private void ClearCasti()

Nastaví všechny části čísla na „“ a první na „0“.

4.5.3 Postup pro implementaci číselníku v appletu

Pro implementaci číselníku do appletu je třeba do appletu přidat *Číselník* stejně jako *JPanel*, pro konstruktor je třeba mít *JLabel*, do kterého bude číselník zapisovat, a pole čtyř *JRadioButtonů*, které bude číselník používat pro volbu částí čísla. Tyto komponenty mohou být umístěny kdekoliv, na číselník to nemá vliv. Číselník musí dostávat informace o stisku klávesy, ať se *focus* nachází kdekoliv. Proto je nutné pro každou komponentu, která může mít focus (*isFocusable == true*) zaregistrovat číselník jako *KeyListener*. Podobně, pokud v appletu potřebujete poslouchat klávesy stisknuté na Číselníku, je nutné si zaregistrovat svůj *KeyListener* na každé z *Ciselnik.btnNumbers* (popřípadě *btnOthers*). Nakonec přehled obslužných metod. *ClearNow()* je pro smazání právě zadávaného čísla (třeba stisk AC), *setLabel()* pro nastavení jiného labelu (přechod na jinou složku vektoru/matice), proměnná *clear* pro vymazání před zadáním dalšího čísla (po stisku =), metoda *setRbsEnabled()* pro „zešedivění“ *Radiobuttonů* (třeba po stisku =) a nakonec metody *SetText()* a *GetText()* pro získání a nastavení textu v labelu Číselníku.

4.5.4 Postup pro přidání nového typu do číselníku

V případě přidání nového typu čísla (například uvažovaný typ QC – komplexní čísla s racionálními částmi) je třeba přidat kód do následujících metod:

- Konstruktor Císelniku, pokud by typ potřeboval svá (nová) tlačítka
- Zvýšit počet částí čísla, pokud je třeba
- MyKeyPressed() – zde je třeba do case přidat nový typ
- přidat metodu MyKeyPressed_X() kde X je nový typ
- SetVars() – přidat do case nový typ, pokud je to potřeba
- přidat metodu SetVars_X(), pokud ji nový typ potřebuje
- changeDisplay() – zde je třeba přidat svá nastavení
- přidat metodu GetSubstr_X() pokud je potřeba
- v metodě Parse přidat do case získání reprezentace v částech ze stringu

4.5.5 Tester – prostředí pro testovací počítání s typy čísel

Nejjednodušší applet pro vyzkoušení si počítání s čísly. Popsané procedury jsou v každém appletu se stejným významem.

private void createGUI()

Vytvoří grafické rozhraní.

public void actionPerformed (ActionEvent event)

Obsluhuje stisk tlačítka nebo výběr z ComboBoxu.

private void changeDisplay()

Mění vzhled v závislosti na zvoleném tělese.

private int getPrime()

Z indexu zvolené položky comboboxu dostane odpovídající prvočíslo.

private double getZero()

Z JTextField obsahujícího hodnotu nulovosti dostane její double reprezentaci.

private int getDecs()

Z JTextField obsahujícího počet platných cifer dostane jeho double reprezentaci.

4.5.6 Calc2 – kalkulačka v tělesech

Applet pro počítání s čísly z různých těles, lze ovládat jak klávesnicí, tak i myší, počítá se v něm stejně jako na kalkulačce. Podporuje paměti a historii výpočtu.

JFrame frmCalc2

Hlavní okno appletu.

JFrame frmPlus, frmMult, JPanel pnlPlus, pnlMult, GridLayout GLP, GLM

Objekty týkající se tabulek pro + a * v tělesech Z_p .

JLabel lblP, lblDigits, lblZero, JComboBox cmbTelesa, cmbPrimes, cmbDigits, cmbZero, JCheckBox chkE

Objekty pro nastavování vlastností těles.

JTextArea txtHistory, JScrollPane spnHistory

Objekty pro historii výpočtu.

JTextField[] txtsMem, JButton[] OpBut, MemToButs, MemFromButs, Num[] Mem

Objekty pro paměti – tlačítka, zobrazení a interní reprezentace.

final int MEMSLOTS = 4

Počet pamětí.

final int RBUTS = 4

Počet Radiobuttonů pro Číselník.

int Op = 10

Kód poslední operace.

int OpR = 10

Kód poslední operace před rovnítkem.

Num NR

Číslo, se kterým se bude opakovat operace OpR, pokud bude opakovaně stisknuto rovnítko.

Num Prev

Minulé číslo (na které se má provést operace Op s právě zadaným číslem).

boolean LastOp

Indikuje, zda byla naposledy stisknuta operace.

boolean rovnitko

True: Stisk čísla smaže předchozí výsledek, stisk operace ho použije dál.

int pamet

0 – nezadává se číslo paměti

1 – zadává se číslo paměti pro čtení

2 – zadává se číslo paměti pro uložení

public void keyTyped(KeyEvent e)

Obsluhuje stisk tlačítka klávesnice, obsahuje tabulku použitých kláves.

public void actionPerformed (ActionEvent event)

Obsluhuje stisk tlačítka nebo výběr z ComboBoxu včetně tlačítek Číselníku.

private void MyKeyPressed(int OpCode)

Hlavní procedura appletu – zpracovává stisk tlačítka a kliknutí myši.

private void History(String s)

Zapíše *s* do historie výpočtu.

4.5.7 VectCalc2 – prostředí pro testování počítání s vektory

Applet, který umožňuje základní výpočty s vektory – sčítání a odčítání vektorů, násobení a dělení vektoru skalárem. Struktura appletu je stejná jako v případě Testeru.

4.5.8 VCalc – Vektorová kalkulačka

Obdoba kalkulačky v tělesech, tentokrát pro vektory. Využívá systém zadávání z Maticové kalkulačky, umožňuje test lineární závislosti.

JFrame frmVCalc

Hlavní okno appletu.

JLabel lblP, lblDigits, lblZero, JComboBox cmbTelesa, cmbPrimes, cmbDigits, cmbZero, JCheckBox chkE

Objekty pro nastavování vlastností těles.

JTextArea txtHistory, JScrollPane spnHistory

Objekty pro historii výpočtu.

EnterMatrix zadani

Oblast pro zadávání vektorů (z Maticové kalkulačky).

JTextField[] txtsMem, JButton[] OpBut, MemToButs, MemFromButs, VectNum[] Mem, Jcheckbox[] chkLZ

Objekty pro paměti – tlačítka, zobrazení a interní reprezentace.

final int MEMSLOTS = 4

Počet pamětí.

final int RBUTS = 4

Počet Radiobuttonů pro Číselník.

int Op = 10

Kód poslední operace.

int OpR = 10

Kód poslední operace před rovnítkem.

Num NR

Číslo, se kterým se bude opakovat operace OpR, pokud bude opakovaně stisknuto rovnítko.

VectNum VR

Vektor, se kterým se bude opakovat operace OpR, pokud bude opakovaně stisknuto rovnítko.

VectNum PrevV

Minulý vektor (na který se má provést operace Op s právě zadaným číslem/vektorem).

boolean LastOp

Indikuje, zda byla naposledy stisknuta operace.

boolean rovnítko

True: Stisk čísla smaže předchozí výsledek, stisk operace ho použije dál.

int pamet

0 – nezadáva se číslo paměti

1 – zadává se číslo paměti pro čtení

2 – zadává se číslo paměti pro uložení

boolean ZadatSkalar

Určuje, zda se bude zadávat vektor či skalár.

private Matrix Vect2Matrix(VectNum v)

Převádí vektor na jednořádkovou matici.

private VectNum Matrix2Vect(Matrix m)

Převádí první řádek matice na vektor.

private void SetZadatSkalar(boolean b)

Nastaví proměnnou *ZadatSkalar* a příslušně upraví UI.

public static boolean LN(Matrix M)

Vrací *true*, pokud jsou vektory uložené v řádcích matice M lineárně nezávislé.

public static Matrix LN_subset(Matrix M)

Vrací matici, ve které jsou vektory uložené v řádcích lineárně nezávislé a jsou podmnožinou řádkových vektorů matice M.

4.6 Balíček Progress

Tento balíček obsahuje jednoduché třídy pro zobrazování průběhu algoritmů. Vždy stačí zavolat konstruktor třídy odpovídající algoritmu, jehož průběh chceme zobrazit, s příslušnými parametry.

Kapitola 5

Obsah CD

- Java Runtime Environment 6.0 update 1
- Zdrojové kódy (NetBeans 5.5 Project)
- Webové stránky s applety spouštěnými přes společné menu
- Webové stránky s applety spouštěnými přímo na stránce
- Elektronická podoba této bakalářské práce

Literatura

[1] Bečvář J.: *Lineární Algebra*, Matfyzpress, Praha, 2002.

[2] Klímek J.: *Lineárněalgebraický jukebox*, <http://kubik.php5.cz/la.php>.