FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Semestrální práce z předmětu

Programování v jazyce C

Vizualizace grafu matematické funkce

2. ledna 2025

Autor:

Jiří Joska
A23B0272P
jjoska@students.zcu.cz

Obsah

1 Zadání **2**

2 Analýza úlohy **2**
2.1 Specifikace problému . . . 2
2.1.1 Cíl projektu . . . 2
2.1.2 Požadavky na parser . . . 3
2.1.3 Požadavky na generátor grafů . . . 3
2.2 Teoretická analýza . . . 4
2.2.1 Parsování matematických výrazů . . . 4
2.2.2 Vyhodnocování výrazů . . . 5
2.2.3 Generování PostScript výstupu . . . 6

3 Popis implementace **7**
3.1 Architektura aplikace . . . 7
3.2 Klíčové datové struktury . . . 8
3.2.1 Struktura `Parser` . . . 8
3.2.2 Generický zásobník `stack` . . . 8
3.2.3 Parametry grafu `GraphParams` . . . 8
3.3 Implementace matematického parseru . . . 9
3.4 Generování PostScript výstupu . . . 9
3.4.1 Koordinační systém a škálování . . . 10
3.4.2 Systém mřížky a popisků . . . 10
3.4.3 Typografické aspekty . . . 10
3.4.4 Správa grafického kontextu . . . 10
3.4.5 Vykreslování funkční křivky . . . 11
3.4.6 Správa chyb a validace vstupů . . . 12

4 Uživatelská příručka **12**
4.1 Systémové požadavky . . . 12
4.2 Instalace a sestavení . . . 12
4.3 Použití programu . . . 12
4.3.1 Parametry programu . . . 13
4.3.2 Příklady použití . . . 13
4.4 Formát výstupu . . . 14
4.5 Chybové hlášky . . . 15

5 Závěr **16**

Seznam obrázků **17**

Seznam ukázek kódů **17**

1 Zadání

Naprogramujte v ANSI C přenositelnou **konzolovou aplikaci**, která jako vstup načte z parametru na příkazové řádce matematickou funkci ve tvaru $y = f(x)$, kde $x, y \in \mathbb{R}$, provede její analýzu a vytvoří soubor ve formátu PostScript s grafem této funkce na zvoleném definičním oboru konzolové aplikace.

Program se bude spouštět příkazem `graph.exe` $\langle func \rangle$ $\langle out - file \rangle$ [$\langle limits \rangle$]. Symbol $\langle func \rangle$ zastupuje zápis matematické funkce jedné reálné nezávisle proměnné (funkce ve více dimenzích program řešit nebude, nalezne-li během analýzy zápisu funkce více nezávisle proměnných než jednu, vypíše srozumitelné chybové hlášení a skončí). Závisle proměnná (zde y) je implicitní, a proto se levá strana rovnosti v zápisu nebude uvádět. Symbol $\langle out - file \rangle$ zastupuje jméno výstupního PostScriptového souboru. Takže Váš program může být během testování spuštěn například takto:

```
X:\>graph.exe "sin(2*x)^3" mygraph.ps
```

Výsledkem práce programu bude soubor ve formátu PostScript, který bude zobrazovat graf zadané matematické funkce – ve výše uvedeném případě $y = \sin(2x)^3$ – v kartézské souřadnicové soustavě $(O; x, y)$ s vyznačenými souřadnými osami a (aspoň) význačnými hodnotami definičního oboru a oboru hodnot funkce (viz Specifikace výstupu programu).

Pokud nebudou na příkazové řádce uvedeny alespoň dva argumenty, vypište chybové hlášení a stručný návod k použití programu v angličtině podle běžných zvyklostí (viz např. ukázková semestrální práce na webu předmětu Programování v jazyce C). **Vstupem programu jsou pouze argumenty na příkazové řádce – interakce s uživatelem pomocí klávesnice či myši v průběhu práce programu se neočekává.**

2 Analýza úlohy

2.1 Specifikace problému

2.1.1 Cíl projektu

Cílem tohoto projektu je vytvořit systém, který dokáže:

1. Zpracovat matematický výraz zadaný jako textový řetězec.
2. Vyhodnotit tento výraz pro různé hodnoty proměnné `x`.
3. Vytvořit vizualizaci funkce pomocí formátu PostScript.

2.1.2 Požadavky na parser

Implementovaný syntaktický analyzátor musí splňovat specifické požadavky, které vyplývají z povahy řešené úlohy. Obecně je požadováno zpracování matematických výrazů zadaných v textové podobě, přičemž parser musí podporovat základní matematické operace, zahrnující sčítání, odčítání, násobení, dělení, mocnění a unární mínus. Priorita operátorů musí být v souladu s matematickými pravidly, případně upravena pomocí kulatých závorek. Žádné jiné typy závorek nesmějí být povoleny.

Parser musí umožnit práci s vybranými matematickými funkcemi, mezi něž patří funkce pro výpočet absolutní hodnoty, exponenciální funkce, přirozený a dekadický logaritmus, standardní goniometrické funkce, jejich cyklometrické inverze a hyperbolické varianty. Zadání matematických výrazů nesmí obsahovat jiné symboly, než které odpovídají definovanému formátu. Pokud parser narazí na neplatné znaky, musí generovat srozumitelnou chybovou zprávu.

Matematické výrazy musí být schopny zpracovávat konstanty ve formátu celých a reálných čísel, včetně exponentové notace běžné v programovacím jazyce C. Vzhledem k tomu, že úloha předpokládá práci pouze s grafy v dvourozměrném prostoru, jedinou povolenou proměnnou je `x`, která je implicitně deklarována a připravena k použití uživatelem.

Při specifikaci rozsahu zobrazení grafu je třeba umožnit zadání definičního oboru a oboru hodnot prostřednictvím nepovinného parametru. Ten by měl být reprezentován čtyřmi hodnotami oddělenými dvojtečkou, které definují dolní a horní meze definičního oboru a oboru hodnot. Pokud tento parametr není zadán, parser musí pracovat s implicitním rozsahem, který je symetrický a přednastavený na interval od `-10` do `10` pro obě na sebe kolmé osy v kartézské soustavě.

2.1.3 Požadavky na generátor grafů

Prvním a nejdůležitějším požadavkem je schopnost vykreslení grafu matematické funkce v předem definovaném intervalu. Tento interval musí být přesně respektován a odpovídat jak definičnímu oboru, tak oboru hodnot, jež byly uživatelem specifikovány, nebo které odpovídají implicitnímu nastavení.

Dalším klíčovým požadavkem je generování výstupu ve formátu PostScript. Tento formát zajišťuje přenositelnost výsledků a jejich využití v různých grafických aplikacích. Generátor musí být schopen produkovat čistý, validní a efektivní PostScript kód, který lze zobrazit nebo vytisknout.

Graf musí zahrnovat podporu mřížky a os, které umožní uživateli snadnou orientaci v zobrazeném grafu. Mřížka by měla být vykreslena konzistentně v souladu se škálováním grafu a měla by být vizuálně přehledná. Osy grafu musí být správně označeny, včetně

popisků zobrazující rozsahy jednotlivých os.

Správné škálování grafu je dalším nezbytným požadavkem. Generátor musí zajistit, aby všechny prvky grafu, včetně křivek, mřížky a os, byly zobrazeny v odpovídající velikosti a proporcích. Škálování musí být automaticky přizpůsobeno specifikovanému intervalu tak, aby nedocházelo k deformaci zobrazení nebo ke ztrátě detailů.

Specifickým aspektem, který generátor musí zohlednit, je ošetření nedefinovaných bodů v grafu. Tyto body, například singularity nebo hodnoty mimo definiční obor, nesmí narušit konzistenci zobrazení. Generátor by měl tyto situace identifikovat a správně je zpracovat.

2.2 Teoretická analýza

2.2.1 Parsování matematických výrazů

V oblasti parsování matematických výrazů existuje několik přístupů, z nichž každý má své výhody a nevýhody. V následujícím textu se podíváme na tři hlavní metody parsování a provedeme jejich analýzu, přičemž zvolené řešení je rekurzivní sestupný parser.

Rekurzivní sestupný parser (zvolené řešení) Rekurzivní sestupný parser je jednoduchý a efektivní nástroj pro syntaktickou analýzu matematických výrazů. Tento přístup je velmi přímočarý, což znamená, že jeho implementace je relativně snadná. Parser je navržen tak, že každá funkce v parseru odpovídá jednomu pravidlu gramatiky. Díky tomu je snadno rozšiřitelný a udržovatelný. Je také přirozeným způsobem, jak zpracovávat operátorovou precedenci.

Výhody:

- Přímočará implementace, což usnadňuje tvorbu a pochopení kódu.
- Snadná údržba a rozšiřitelnost, což znamená, že parser lze relativně jednoduše modifikovat pro nové funkce nebo operátory.
- Přirozené zpracování operátorové precedence, které je nezbytné pro správné vyhodnocení matematických výrazů.

Nevýhody:

- Vyšší paměťová náročnost při hluboké rekurzi, což může být problém při zpracování složitějších výrazů.
- Potenciálně pomalejší než jiné přístupy, což může ovlivnit výkon při práci s rozsáhlými výrazy.

Shunting Yard algoritmus Shunting Yard algoritmus, navržený Edsgerem Dijkstrou, je efektivní metodou pro syntaktickou analýzu matematických výrazů zapsaných v infixové notaci. Tento přístup se vyznačuje nižší paměťovou náročností ve srovnání s rekurzivním sestupem, protože nevyžaduje použití rekurzivního zásobníku. V případě výrazů s více operátory nebo s operacemi s vyšší prioritou je tento algoritmus efektivnější.

Výhody:

- Efektivní zpracování výrazů, zejména u složitějších operátorů a funkcí.
- Nižší paměťová náročnost, což je výhodné při práci s velkými výrazy.

Nevýhody:

- Složitější implementace pro funkce, což může znamenat více kódu a složitější správu výjimek.
- Obtížnější udržitelnost a rozšiřitelnost, zejména pokud je potřeba přidat nové funkce nebo operátory.

Parser kombinátory Parser kombinátory jsou flexibilním přístupem k syntaktické analýze, kde lze jednotlivé dílčí parsovací funkce kombinovat do složitějších parserů. Tento přístup je velmi modulární a deklarativní, to znamená, že parser je snadno přizpůsobitelný a rozšiřitelný.

Výhody:

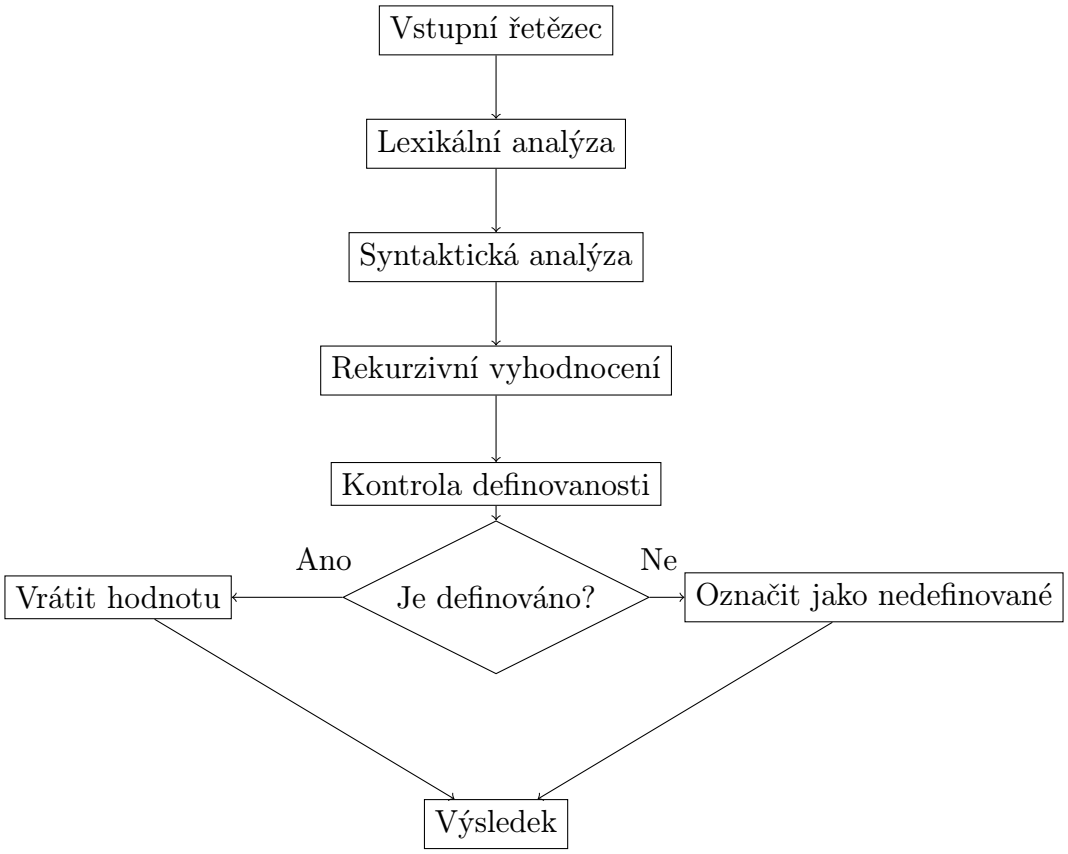
- Vysoká modularita, která umožňuje snadnou úpravu a přizpůsobení parseru.
- Deklarativní styl, který usnadňuje pochopení a práci s jednotlivými komponentami parseru.

Nevýhody:

- Složitější implementace v C, protože tento přístup není v tomto jazyce přirozeně podporován.
- Potenciálně horší výkon, zejména při práci s většími a složitějšími výrazy.

2.2.2 Vyhodnocování výrazů

Proces vyhodnocování zahrnuje několik fází, od lexikální analýzy, přes syntaktickou analýzu, až po samotné vyhodnocení. Diagram níže znázorňuje proces vyhodnocování výrazu.



Obrázek 1: Proces vyhodnocování matematického výrazu.

Tento diagram ukazuje kroky, které jsou prováděny při analýze a vyhodnocování matematického výrazu. Po lexikální analýze a syntaktické analýze následuje fáze rekurzivního vyhodnocení, kde je každá část výrazu vyhodnocena podle definovaných pravidel. Následně se ověřuje, zda jsou jednotlivé hodnoty definované. Pokud je hodnota definována, vrátí se její výsledek; pokud není, označí se jako nedefinovaná.

2.2.3 Generování PostScript výstupu

Generování PostScript výstupu je finální fází při tvorbě grafu. Postup zahrnuje několik základních kroků: definici souřadnicového systému, vykreslení mřížky a os, přidání popisků a legendy a nakonec zobrazení samotné křivky funkce, která je tvořena propojenými diskrétními body.

Struktura PostScript dokumentu PostScript dokument začíná hlavičkou, která obsahuje metadata, jako jsou informace o autorovi, název dokumentu a rozměry grafu. Následuje definice souřadnicového systému, jenž umožňuje přizpůsobení měřítka a po-

lohy grafu. Další částí je vykreslení mřížky a os, které pomáhají při orientaci v grafu. Poté se přidávají popisky a legenda, které zajišťují čitelnost a poskytují důležité informace pro interpretaci grafu. Následně je vykreslena funkční křivka podle vypočítaných bodů. Po vykreslování je vložen příkaz `showpage` pro zobrazení obsahu a na konci ukončující znak `%EOF`.

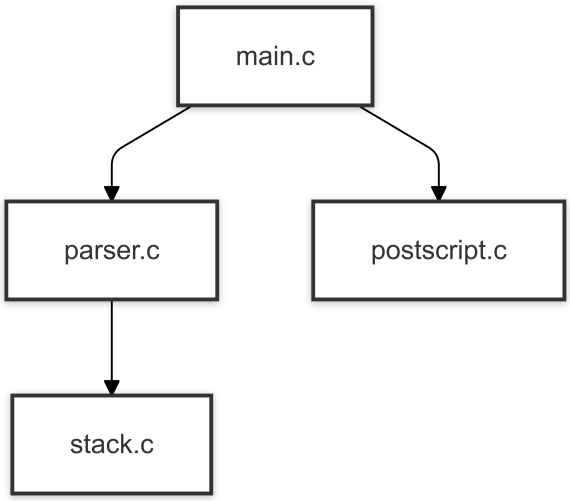
3 Popis implementace

3.1 Architektura aplikace

Systém pro parsování matematických výrazů a generování jejich grafické reprezentace využívá modulární architekturu rozdělenou do čtyř hlavních modulů:

- `main.c` – vstupní bod aplikace, zpracovává argumenty příkazové řádky a koordinuje ostatní moduly.
- `parser.c` – implementuje algoritmus rekurzivního sestupu pro analýzu matematických výrazů.
- `postscript.c` – generuje grafický výstup ve formátu PostScript.
- `stack.c` – poskytuje generickou implementaci zásobníku pro zpracování výrazů.

Následující schéma ilustruje závislosti mezi moduly:



Obrázek 2: Mudulární architektura projektu.

3.2 Klíčové datové struktury

3.2.1 Struktura `Parser`

Základní datová struktura enkapsulující informace pro zpracování matematického výrazu:

Kód 1: Struktura Parser

```
typedef struct {
    const char *expr; /* Aktualni pozice ve vyrazu */
    double x;         /* Hodnota promenne x */
} Parser;
```

3.2.2 Generický zásobník `stack`

Flexibilní datová struktura pro práci s různými typy:

Kód 2: Struktura Stack

```
typedef struct {
    void *items;       /* Pole polozek */
    size_t capacity;   /* Maximalni kapacita */
    size_t item_size;  /* Velikost polozky (v bytech) */
    size_t sp;         /* Ukazatel na vrchol zasobniku */
} Stack;
```

3.2.3 Parametry grafu `GraphParams`

Tato struktura obsahuje parametry nezbytné pro generování grafického výstupu:

Kód 3: Struktura GraphParams

```
typedef struct {
    double min_x, max_x, min_y, max_y; /* Rozsahy os */
    int width, height;                 /* Rozmery vystupu */
    int x_divisions, y_divisions;/* Pocet dilku na osach */
    Point *points;                     /* Vypoctene body */
    size_t num_points;                 /* Pocet bodu */
} GraphParams;
```

3.3 Implementace matematického parseru

Parser je implementován pomocí metody rekurzivního sestupu, která umožňuje efektivně zpracovávat matematické výrazy podle definované gramatiky. Tato gramatika je definována následující formální notací:

```
expression = term {("+"|"-") term}
term       = factor {("*"|"/") factor}
factor     = number | "x" | function "(" expression ")" |
             "(" expression ")" | "-" factor | factor "^" factor
```

Důležitým aspektem implementace je hierarchické členění operací podle jejich matematické priority. Funkce `parse_expression` zpracovává operace s nejnižší prioritou, tj. sčítání a odčítání, funkce `parse_term` zpracovává násobení a dělení a `parse_factor` řeší operace s ještě vyšší prioritou jako je uzávorkování, unární mínus a umocnění.

Pro zpracování matematických funkcí je ve funkci `parse_function` implementován mechanizmus, který nejprve extrahuje název funkce a poté rekurzivně vyhodnocuje její argumenty. V rámci funkce se nejprve do bufferu přečte název funkce, který je následně použit k určení konkrétní matematické operace. Po zpracování názvu funkce parser očekává otevírací závorku, argument funkce a uzavírací závorku. Argument je vyhodnocen rekurzivním voláním funkce `parse_expression`.

Implementace obsahuje kompletní sadu základních matematických funkcí: trigonometrické funkce (`sin`, `cos`, `tan`), jejich inverzní varianty (`asin`, `acos`, `atan`), hyperbolické funkce (`sinh`, `cosh`, logaritmické funkce (`ln`, `log`), exponenciální funkci (`exp`), `tanh`) a absolutní hodnotu (`abs`). Pro každou funkci jsou implementovány kontroly definičního oboru. Například pro inverzní trigonometrické funkce se kontroluje, zda argument leží v intervalu $[-1, 1]$, a pro logaritmické funkce se ověřuje, zda je argument kladný. V případě porušení těchto podmínek je nastaven příznak chyby `errno` na hodnotu `ERANGE` a funkce vrací hodnotu `HUGE_VAL`.

Zpracování číselných hodnot je realizováno funkcí `parse_number`, která implementuje komplexní algoritmus pro parsování čísel v různých formátech. Podporovány jsou celá čísla, desetinná čísla i vědecká notace. Implementace zahrnuje speciální ošetření případů, kdy číslo začíná desetinnou tečkou, toto vyžaduje manipulaci s řetězci pro zajištění korektního parsování.

3.4 Generování PostScript výstupu

Systém pro generování grafů ve formátu PostScript je navržen tak, aby poskytoval přesné a kvalitní vizualizace funkcí. Tento systém je postaven na principu využívající strukturu `GraphParams`, která zapouzdřuje všechny nezbytné parametry pro generování grafu.

3.4.1 Koordinační systém a škálování

Implementace zahrnuje systém pro konfiguraci a správu souřadnicového systému PostScriptu. Funkce `setup_coordinate_system` provádí přesný výpočet měřítkových faktorů pro obě osy. Zohledňuje přitom hraniční případy, například nulový rozsah os, a automaticky aplikuje odpovídající transformace. Tím je zajištěno optimální využití dostupného prostoru při zachování proporcionality dat.

Kód 4: Nastavení koordinačního systému

```
/* Vypocet rozsahu dat */
double x_range = params->max_x - params->min_x;
double y_range = params->max_y - params->min_y;
/* Vypocet meritka pro osu X a Y */
fprintf(ps_file, "/xScale %g def\n",
    (x_range != 0) ? params->width / x_range : 1.0);
fprintf(ps_file, "/yScale %g def\n",
    (y_range != 0) ? params->height / y_range : 1.0);
```

3.4.2 Systém mřížky a popisků

Generování mřížky a popisků využívá algoritmus pro formátování číselných hodnot na osách. Formátování se řídí následujícími pravidly:

- Hodnoty blízké nule ($|x| < 0.001$) jsou zobrazeny ve speciálním formátu pro lepší čitelnost.
- Pro velké hodnoty ($|x| \geq 1000$) je použit vědecký zápis.
- Ostatní hodnoty jsou prezentovány ve standardním desetinném formátu s konstantním počtem desetinných míst.

3.4.3 Typografické aspekty

Systém také klade důraz na typografickou kvalitu textových elementů. Pro textové prvky je použit font Helvetica, který zajišťuje dobrou čitelnost. Je zde implementován automatický výpočet pozic textových elementů pro jejich správné zarovnání. K popisku vertikální osy je použita rotace textu o 90 stupňů.

3.4.4 Správa grafického kontextu

Pro správu transformací a grafických atributů využívá systém zásobník grafických stavů PostScriptu. Implementace zajišťuje správnou inicializaci a ukončení grafického kontextu a přesné nastavení okrajů a transformací souřadnic.

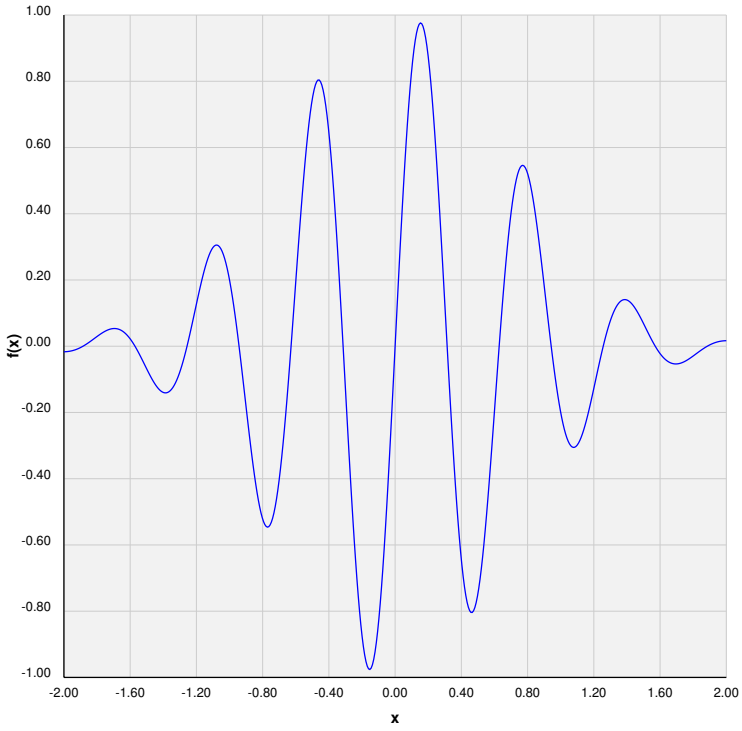
Všechny grafické operace jsou navrženy s ohledem na minimalizaci velikosti výsledného PostScript souboru a maximalizaci efektivity.

3.4.5 Vykreslování funkční křivky

Proces vykreslování funkční křivky využívá algoritmus, který se zaměřuje na vysokou přesnost a vizuální kvalitu výstupu. Implementace obsahuje systém pro detekci a zpracování nespojitostí ve funkcích, který pracuje na principu sledování validity bodů pomocí proměnné `first_valid_point`.

Při detekci bodu mimo definovaný rozsah dojde k přerušení aktuální cesty a při návratu do platného rozsahu začne cesta nová. Křivka je diskretizována na 512 bodů, které jsou rovnoměrně rozloženy na intervalu $[min_x, max_x]$, což ve většině případů zajišťuje dostatečně hladký vzhled při zachování rozumné výpočetní náročnosti.

Algoritmus provádí dvojí transformaci souřadnic – nejprve z indexu bodu na x-ovou souřadnici a následně z datových souřadnic na grafické. Přitom využívá PostScript příkazy (`moveto`, `lineto`) a minimalizuje jejich počet pomocí podmíněného vykreslování.



Obrázek 3: Graf funkce $f(x) = e^{-x^2} \sin(10x)$, kde $x \in [-2, 2]$.

3.4.6 Správa chyb a validace vstupů

Systém implementuje mechanismus zpracování chyb prostřednictvím předdefinovaných chybových kódů, které pokrývají tři hlavní kategorie problémů:

1. Neplatné vstupní parametry (`ERROR_INVALID_PARAMS`).
2. Selhání operací se soubory (`ERROR_FILE_OPERATION`).
3. Problémy s alokací paměti (`ERROR_MEMORY_ALLOCATION`).

Každá operace podléhá důkladné validaci vstupních parametrů. Tento přístup zajišťuje vysokou robustnost a spolehlivost generovaného výstupu.

4 Uživatelská příručka

4.1 Systémové požadavky

Pro úspěšné sestavení a spuštění programu potřebujete:

- ANSI C kompatibilní kompilátor (např. GCC)
- Make utilitu
- Standardní C knihovny (math.h, stdlib.h, stdio.h, string.h, stdbool.h, ctype.h, errno.h)

4.2 Instalace a sestavení

Linux/Unix systémy Pro sestavení programu na Unix-like systémech postupujte následovně:

```
$ make
```

Program bude vytvořen s názvem `program` v aktuálním adresáři.

Windows Pro sestavení programu na Windows:

```
make -f Makefile.win
```

Program bude vytvořen s názvem `program.exe` v aktuálním adresáři.

4.3 Použití programu

Program se spouští z příkazové řádky s následujícími parametry:

```
program <funkce> <vystupni_soubor> [xmin:xmax:ymin:ymax]
```

4.3.1 Parametry programu

- `funkce` – matematický výraz pro vykreslení (pokud je v uvozovkách, může obsahovat mezery). Dostupné funkce v matematickém zápisu jsou: absolutní hodnota abs; funkce exp, přirozený logaritmus ln, dekadický logaritmus log; goniometrické funkce sin, cos, tan; cyklometrické funkce arcsin, arccos, arctan; hyperbolické funkce sinh, cosh, tanh. A matematické operace: unární mínus, sčítání, odčítání, násobení ($*$), dělení ($/$), a umocnění ($\hat{}$).
- `vystupni_soubor` – název výstupního PostScript souboru,
- `xmin:xmax:ymin:ymax` – volitelný parametr určující rozsah vykreslování (výchozí hodnoty jsou $-10 : 10 : -10 : 10$).

4.3.2 Příklady použití

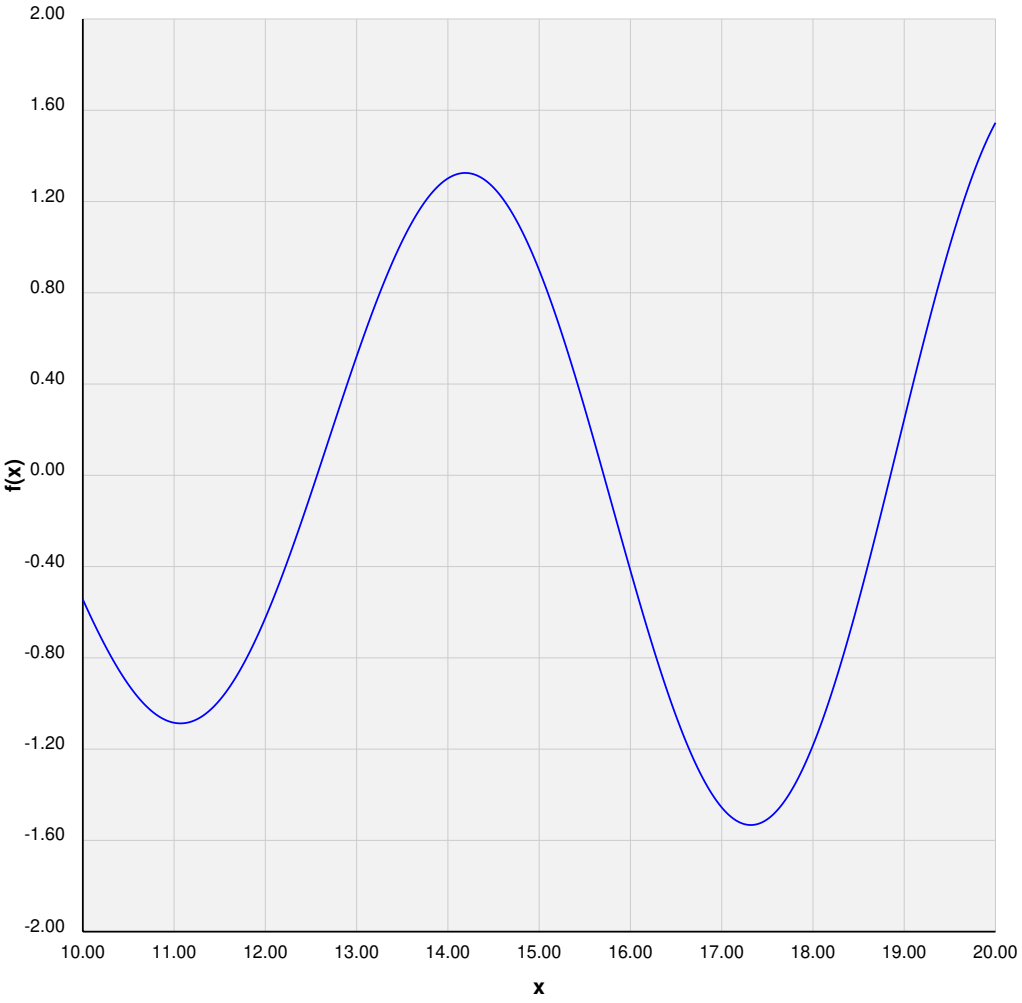
Kód 5: Příklady použití

```
# Zakladni pouziti
$ ./program "sin(x^2)" vystup1.ps

# Pouziti s vlastnim rozsahem
$ ./program "sin(x^2)" vystup2.ps -5:5:-1:1

# Funkce bez mezer, nevyzaduje uvozovky
$ ./program sin\(x\) vystup.ps

# Slozitejsi funkce s mezerami
$ ./program ./program " sin ( x ) * log ( x ) ^ 2 " vystup3
   .ps 10:20:-2:2
```



Obrázek 4: Zobrazení souboru `vystup3.ps`.

4.4 Formát výstupu

Program generuje výstup do kořenového adresáře ve formátu PostScript s následujícími vlastnostmi:

- Rozlišení křivky: 512 x 512 bodů
- Mřížka: 10 x 10 dílků
- Automatické škálování podle zadaného rozsahu

4.5 Chybové hlášky

V implementaci je zajištěno ošetření chybových stavů tak, aby uživatel získal srozumitelnou zpětnou vazbu. Každá z těchto hlášek je navržena tak, aby usnadnila diagnostiku problému a umožnila uživateli podniknout kroky k jeho vyřešení. Následující výčet obsahuje chybové hlášky použité v programu a jejich význam:

- **Error: Invalid mathematical expression.** Tato chyba se vyskytne, pokud je zadán neplatný matematický výraz, který nelze správně analyzovat. Uživatel by měl zkontrolovat syntaxi a platnost výrazu.
- **Error: Cannot create or write to output file '*filename*'.** Program se pokusil vytvořit nebo zapisovat do souboru, ale operace selhala. Možné příčiny zahrnují nedostatečná oprávnění, neexistující adresář nebo jiný problém se souborem.
- **Error: Memory allocation failed.** Došlo k selhání alokace paměti pro ukládání dat. Tato chyba může nastat při nedostatku systémových prostředků nebo při požadavku na nadměrné množství paměti.
- **Error: Invalid character in function: '*char*'.** Funkce obsahuje neplatný znak, který není povolen v matematických výrazech. Uživatel by měl odstranit všechny nepovolené znaky.
- **Error: Invalid xmin value.** Zadána byla neplatná hodnota minimální hranice osy x (`xmin`). Uživatel by měl zadat číselnou hodnotu.
- **Error: Invalid xmax value.** Zadána byla neplatná hodnota maximální hranice osy x (`xmax`). Hodnota musí být číselná a větší než `xmin`.
- **Error: Invalid ymin value.** Zadána byla neplatná hodnota minimální hranice osy y (`ymin`). Uživatel by měl zadat číselnou hodnotu.
- **Error: Invalid ymax value.** Zadána byla neplatná hodnota maximální hranice osy y (`ymax`). Hodnota musí být číselná a větší než `ymin`.
- **Error: Invalid range (max must be greater than min).** Rozsah os byl zadán nesprávně. Maximální hodnota musí být větší než minimální hodnota, jak pro osu x, tak pro osu y.
- **Error: Failed to generate PostScript graph. Code: *n*.** Generování PostScriptového grafu selhalo. Návratový kód chyby (n) poskytuje další informace o problému.
- **Warning: The function contains undefined values in the given range.** Varování upozorňuje, že ve zadaném rozsahu jsou body, kde matematický výraz není definován. Graf bude v těchto oblastech přerušen.

5 Závěr

V rámci této semestrální práce byl vytvořen program pro analýzu a vizualizaci matematických funkcí. Program úspěšně implementuje všechny požadované funkcionality ze zadání.

Implementovaný parser dokáže efektivně zpracovávat matematické výrazy s vysokou přesností. Během testování se ukázalo, že program zvládá analyzovat i komplexní matematické funkce v řádu milisekund. Generování výstupního PostScript souboru ze zadané funkce do terminálu trvá průměrně okolo 5 ms na běžném počítači.

Program byl důkladně otestován na různých operačních systémech (Linux, Windows) a vykazuje konzistentní chování napříč platformami. Důraz byl kladen na robustnost zpracování vstupů a přehledné hlášení případných chyb uživateli.

Během vývoje se vyskytlo několik zajímavých výzev. Především šlo o:

- Implementaci robustního parseru pro matematické výrazy v ANSI C.
- Generování souboru v PostScript formátu.
- Zajištění přenositelnosti kódu mezi různými platformami.

Pro další rozvoj projektu by bylo vhodné zvážit následující vylepšení:

- Rozšíření podpory matematických funkcí o komplexní analýzu.
- Optimalizace paměťové náročnosti při zpracování velmi dlouhých výrazů.
- Přidání podpory pro export do dalších grafických formátů.

Zadání bylo splněno ve všech bodech a výsledný program představuje robustní nástroj pro analýzu matematických funkcí. Zdrojový kód je dobře strukturovaný, důkladně komentovaný a připravený pro případná budoucí rozšíření. Program tak může sloužit jako základ pro další vývoj v oblasti matematické analýzy a vizualizace.

Seznam obrázků

1 Proces vyhodnocování matematického výrazu. 6
2 Mudulární architektura projektu. 7
3 Graf funkce $f(x) = e^{-x^2} \sin(10x)$, kde $x \in [-2, 2]$. 11
4 Zobrazení souboru `vystup3.ps`. 14

Seznam ukázek kódů

1 Struktura Parser 8
2 Struktura Stack 8
3 Struktura GraphParams 8
4 Nastavení koordinačního systému 10
5 Příklady použití 13