FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Semestrální práce z KIV/PC

Vizualizace grafu matematické funkce

Autor:
Trong Quoc Huy Dinh
qhdt@students.zcu.cz

Vyučující:
Ing. Kamil Ekštein, Ph.D.
kekstein@kiv.zcu.cz

Ing. František Pártl
fpartl@ntis.zcu.cz

Obsah

1 Zadání

KIV/PC — ZS 2024/2025 VARIANTA 2 (KEx)

ZADÁNÍ SEMESTRÁLNÍ PRÁCE

VIZUALIZACE GRAFU MATEMATICKÉ FUNKCE

Zadání

Naprogramujte v ANSI C přenositelnou[1] **konzolovou aplikaci**, která jako vstup načte z parametru na příkazové řádce matematickou funkci ve tvaru $y = f(x); x, y \in \mathbb{R}$, provede její analýzu a vytvoří soubor ve formátu PostScript s grafem této funkce na zvoleném definičním oboru.

Program se bude spouštět příkazem `graph.exe` ⟨`func`⟩ ⟨`out-file`⟩ `[`⟨`limits`⟩`]`. Symbol ⟨`func`⟩ 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 ⟨`out-file`⟩ 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.**

Hotovou práci odevzdejte v jediném archivu typu ZIP prostřednictvím automatického odevzdávacího a validačního systému. Postupujte podle instrukcí uvedených na webu předmětu. Archiv nechť obsahuje všechny zdrojové soubory potřebné k přeložení programu, **makefile** pro Windows i Linux (pro překlad v Linuxu připravte soubor pojmenovaný `makefile` a pro Windows `makefile.win`) a dokumentaci ve formátu PDF vytvořenou v typografickém systému TEX, resp. LATEX. **Bude-li některá z částí chybět, validační systém Vaši práci odmítne.**

Podrobné informace k odevzdání práce najdete na webové stránce předmětu Programování v jazyce C na adrese URL `https://www.kiv.zcu.cz/studies/predmety/pc/index.php#work`.

Specifikace vstupu programu

Vstupem programu jsou pouze parametry na příkazové řádce: Prvním (a nejdůležitějším) parametrem je matematická funkce, kterou má program zpracovat. S ohledem na to, že může její zápis obsahovat mezery, napište program tak, aby akceptoval jak zápis funkce obklopený uvozovkami, tak bez nich, tj. aby bylo možné program spouštět jak zadáním příkazu `graph.exe x+x^2`, tak příkazem `graph.exe "x + x^2"`. Také použití mezer v zápisu funkce musí být libovolné: Uvědomte si, že funkce se bude analyzátoru předávat jako parametr na příkazové řádce, tj. pokud zápis parametru není uzavřen v uvozovkách, nesmí obsahovat mezery, jinak by byl příkazovým

[1] Je třeba, aby bylo možné Váš program přeložit a spustit na PC s operačním prostředím Win32/64 (tj. operační systémy Microsoft Windows NT/2000/XP/Vista/7/8/10/11) a s běžnými distribucemi Linuxu (např. Ubuntu, Debian, Red Hat, atp.). Server, na který budete Vaši práci odevzdávat a který ji otestuje, má nainstalovaný operační systém Debian GNU/Linux 11 (bullseye) s jádrem verze 5.10.0-28-amd64 a s překladačem gcc 10.2.1-6.

procesorem (ve Windows program `cmd`) vyhodnocen jako několik parametrů. Naopak, pokud je nutné do zápisu funkce mezery vložit (z estetických důvodů nebo kvůli přehlednosti), pak musí být zápis funkce uzavřen v uvozovkách, `"..."`. Umožněte proto oba dva způsoby zápisu, tedy „hustý“ zápis bez mezer i zápis s mezerami uzavřený do uvozovek. Je také možné, že uživatel vašeho programu z nepořádnosti někde v zápisu mezery použije a někde zase ne, třeba takto: `"sin (x ^2)*cos(x / 2.0)"` – tato varianta zápis, ač ošklivá, je také akceptovatelná!

Způsob zápisu funkcí

V zápisu matematické funkce, jejíž graf bude program generovat, se mohou vyskytnout tyto symboly: (i) konstanty, (ii) proměnná, (iii) aritmetické operátory, (iv) funkce, (v) závorky. Jiné symboly zápis obsahovat nesmí – pokud se v zápisu objeví, měl by program reagovat srozumitelným chybovým hlášením.

Konstanty (i) mohou být ve všech akceptovatelných formátech zápisu celého či reálného čísla v jazyce C, tj. např. i `.5E-02` či `1E0`. S ohledem na to, že má program řešit jen 2D grafy, jediná možná nezávislá proměnná (ii), která se může v zápisu funkce vyskytovat je `x`. V zápisu se nedeklaruje, předpokládáme její implicitní deklaraci (prostě existuje a je připravená k použití uživatelem v zápisu funkce).

Tabulka uvádí přehled všech operátorů (iii), které je možné v zápisu matematické funkce použít:

Operace	Zápis operátoru	Příklad
unární mínus	`-`	`-x`
sčítání	`+`	`x + 2`
odčítání	`-`	`x - 2`
násobení	`*`	`2 * x`
dělení	`/`	`x / 2`
umocnění	`^`	`x^-2`

Jiné operátory nejsou povolené. Priorita operátorů je dána matematickými pravidly, případně upravena závorkami `(` a `)` – jiné druhy závorek se v zápisu funkce nesmí vyskytovat. Platný zápis je tedy např.: `(x * sin(x^(2^x))) / 2`.

Specifikace rozsahu zobrazení

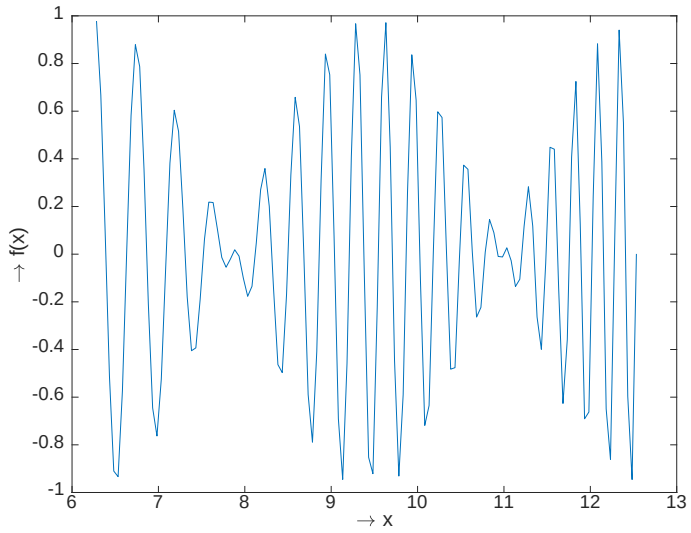
Třetí – **nepovinný** – parametr `[`⟨`limits`⟩`]` na příkazové řádce slouží k předání informací o rozsazích zobrazení grafu analyzované funkce, tedy o horní a dolní mezi definičního oboru a oboru hodnot funkce. Má tvar ⟨*xdol*⟩:⟨*xhor*⟩:⟨*ydol*⟩:⟨*yhor*⟩. Chceme-li tedy např. zobrazit funkci $y = \sin(x^2)\cdot\cos(x)$ na intervalu $\langle 2\pi; 4\pi\rangle$, přičemž obor hodnot funkce bude roztažen přes celý graf, spustíme program příkazem `graph.exe "sin(x^2)*cos(x)" sin2cos.ps 6.28:12.56:-1:1`. Obrázek 1 ukazuje, jak by měl vypadat výsledek.

Není-li rozsah uveden (protože se jedná o nepovinný parametr), použijte implicitní nastavení, které nechť je pro definiční obor i obor hodnot $\langle -10; 10\rangle$.

Matematické funkce v zápisu

Váš program by měl akceptovat v zápisu zobrazované funkce některé běžně používané matematické funkce. Funkce uvedené v následujícím výčtu program **musí** akceptovat, jinak nebude uznán za řádně dokončený: absolutní hodnota `abs`; funkce e^x `exp`, přirozený logaritmus `ln`, dekadický logaritmus `log`; goniometrické funkce `sin`, `cos`, `tan`; cyklometrické funkce `asin`, `acos`, `atan`; hyperbolometrické funkce `sinh`, `cosh`, `tanh`. Další funkce můžete samozřejmě implementovat z vlastní iniciativy.

2



Obrázek 1: Graf funkce $y = \sin(x^2) \cdot \cos(x)$ na intervalu $\langle 2\pi; 4\pi \rangle$.

Specifikace výstupu programu

Výstup programu bude směrován na konzoli pouze v případě chyby v zadání parametrů. Pokud budou všechny parametry akceptovatelné a funkce k zobrazení syntakticky správně zapsaná, nemusí program vypisovat na konzoli nic. Pokud program neobdrží požadovaný počet parametrů na příkazovém řádku či pokud je funkce špatně zapsána, měl by vypsat srozumitelné chybové hlášení v angličtině a **skončit nenulovým návratovým kódem**. Návratové kódy jsou určeny tabulkou 1.

Tabulka 1: Návratové kódy programu v případě chyby.

Popis chybového stavu	**Návratová hodnota funkce `int main()`**
Bez chyby/korektní ukončení činnosti programu.	`0`
Nebyly předány správné parametry na příkazové řádce při spuštění programu.	`1`
Řetězec předaný programu jako první parametr není akceptovatelným zápisem matematické funkce (neobsahuje právě jednu proměnnou `x`, obsahuje nepovolené operátory, symboly, atp.).	`2`
Řetězec předaný programu jako druhý parametr nemůže být v hostitelském operačním systému názvem souboru nebo takový soubor nelze vytvořit či do něj ukládat informace.	`3`
Uvedený nepovinný třetí parametr nelze dekódovat jako rozsahy zobrazení, protože např. obsahuje nepovolené znaky nebo uvedené hodnoty nedávají smysl, apod.	`4`

Ostatním chybovým stavům můžete přiřadit návratové kódy dle svého uvážení.

Výsledkem činnosti programu by měl být soubor ve formátu PostScript, který bude obsahovat vykreslený graf zadané funkce. PostScript je vlastně zásobníkově orientovaný programovací jazyk podobný jazyku FORTH. Ukázka níže demonstruje základní techniky:

```
%!PS-Adobe-2.0
%%Creator: Kamil
%%Title: untitled
%%CreationDate: Tue Oct 08 10:00:00 2019
%%PageOrder: Ascend
%%EndComments

3 setlinewidth
newpath
0 0 moveto
100 100 lineto
closepath
stroke

.5 setlinewidth
newpath
0 100 moveto
100 0 lineto
closepath
stroke

/Helvetica-Bold findfont 50 scalefont setfont
0.7 setgray
10 30 moveto
(T) show
0.5 setgray
36 30 moveto
(E) show
0.3 setgray
64 30 moveto
(X) show

showpage
```

Soubory ve formátu PostScript můžete zobrazovat volně dostupným programem GSview, který je grafickým front-endem softwarového RIPu[2] GhostScript – ke stažení z URL `http://www.ghostgum.com.au/software/gsview.htm`.

Úplný popis jazyka PostScript najdete v příručce vydané společností Adobe, která je k dispozici ke stažení na adrese URL `https://www.adobe.com/jp/print/postscript/pdfs/PLRM.pdf`

Užitečné techniky a odkazy

Uvedené techniky je možné (ale nikoliv nezbytně nutné) využít při řešení úlohy. Protože se jedná o postupy víceméně standardní, lze k nim nalézt velké množství dokumentace:

1. syntaktická analýza rekurzivním sestupem,
2. algoritmus Shunting-yard,
3. zásobník a fronta (a jejich použití pro převod infixového výrazu do polské notace),
4. binární strom.

Řešení úlohy je zcela ve vaší kompetenci – zvolte takové algoritmy a techniky, které podle vás nejlépe povedou k cíli. Můžete se volně inspirovat např. webovou aplikací **desmos**, kterou je možno vyzkoušet na adrese URL `https://www.desmos.com/calculator`.

[2]Raster Image Processor – software nebo hardware, který převádí jazyk PostScript na bitmapu vhodnou k tisku nebo zobrazení na monitoru.

2 Analýza úlohy

Hlavní problémy a výzvy spojené s úlohou zahrnují:

- Zpracování uživatelského vstupu a validace matematického výrazu.
- Priorita operátorů a správné vyhodnocení výrazů (shunting-yard algoritmus).
- Bezpečné generování PostScript výstupu a ošetření hraničních hodnot.

2.1 Validace uživatelského vstupu

Jelikož se jedná o konzolovou aplikaci, která reaguje na vstup od uživatele, tak je důležité aby vstup obsahoval všechny nutné parametry a byl smysluplný. Pokud tyto požadavky vstup nesplňuje, program vyhodí chybu a ukončí se.

2.1.1 Matematické funkce a operátory

Program pracuje s matematickými funkcemi, tudíž je nutné, aby uměl pracovat s operátory, závorkami a různými funkcemi.

- Podporované operátory: $+, -, \times, \div, \wedge$
- Podporované funkce:

sin	cos	tan	exp
log	ln	$asin$	$acos$
$atan$	sinh	cosh	tanh
abs			

2.2 Priorita operátorů a správné vyhodnocení výrazu

Pro správné vyhodnocení matematické funkce je klíčové zohlednit prioritu operátorů a správné závorkování. Efektivním řešením je rozdělit matematický výraz na menší části a postupně je vyhodnocovat. Tento proces probíhá od nejvnitřnějších závorek směrem k vnějším částem výrazu, což zajišťuje správnou posloupnost výpočtů. K řešení této problematiky je použit Shunting-yard algoritmus, který efektivně převádí infixový zápis výrazu na postfixový (RPN – Reverse Polish Notation) a umožňuje jeho vyhodnocení.

2.3 Generování PostScript výstupu

Výstupem programu je graf zadané matematické funkce, který je uložen ve formátu PostScript. Graf může být vykreslen v implicitním intervalu (výchozím nastavení) nebo v uživatelsky definovaném intervalu. Při generování grafu jsou body mimo definiční obor funkce a nedefinované body automaticky vynechány. Tím je zajištěna správnost a přehlednost výsledného výstupu.

3 Popis implementace

Program je rozdělen do několika hlavních modulů, které spolupracují za účelem načítání matematického výrazu, jeho validace a vytvoření grafu.

3.1 Povinné a nepovinné argumenty programu

Program přijímá argumenty příkazové řádky, které určují vstupní data a chování výstupu.

Povinné argumenty

1. **První argument** – obsahuje matematický výraz ve tvaru $y = f(x)$.
   Příklad: `sin(2*x)`.
2. **Druhý argument** – určuje název výstupního PostScript souboru.
   Příklad: `output.ps`.

Nepovinné argumenty

1. **Třetí argument** – specifikuje definiční obor a obor hodnot hodnot zadané matematické funkce.
   Výchozí definiční obor a obor hodnot je interval $< -10, 10 >$.

3.2 Validace argumentů příkazové řádky: `main.c`

Validace a zpracování argumentů příkazové řádky je nezbytnou součástí programu, která zajišťuje správné předání vstupních parametrů pro další zpracování. Hlavním cílem je:

- Zkontrolovat správný počet a formát zadaných argumentů.
- Validovat zadanou matematickou funkci.
- Ošetřit chybové stavy, jako je nesprávný formát vstupu nebo chybný název výstupního souboru.
- Umožnit uživatelské nastavení rozsahu **definičního oboru** a **oboru hodnot**, případně použít výchozí hodnoty.

Funkce modulu `main.c`

- `parse_args()` – hlavní funkce pro kontrolu a zpracování argumentů:
  - Kontroluje počet argumentů.
  - Čistí vstupní matematický výraz od nadbytečných mezer.
  - Validuje, že matematický výraz obsahuje pouze povolené znaky a funkce.
  - Kontroluje, že matematická funkce obsahuje pouze **jednu** neznámou.
  - Kontroluje dostupnost výstupního souboru a jeho validitu.
  - Zpracovává volitelný čtvrtý argument, který specifikuje rozsah hodnot definičního oboru a oboru hodnot.
- `main()` – hlavní řídicí funkce programu, která:
  - Volá funkci `parse_args()` pro validaci vstupních argumentů.
  - Volá funkci `generate_postscript`, která vytváří graf zadané matematické funkce v zadaném rozsahu a ukládá ho do souboru ve formátu PostScriptu.
  - Ošetřuje paměťové úniky a správně uvolňuje alokovanou paměť.

Ukázka kódu: `main.c`

Listing 1: Validace argumentů v `main.c`

```
1
2  int parse_args(int argc, char *argv[], char **func, char **outfile,
       double *x_min, double *x_max, double *y_min, double *y_max, int *
       calc_x_range, int *calc_y_range) {
3      if (argc < 3) {
4          fprintf(stderr, "Usage: %s <function> <output file> [x_min:x_max
               :y_min:y_max]\n", argv[0]);
5          return 1;
6      }
7      *func = (char*)malloc(MAX_EXPR_LENGTH * sizeof(char));
8      if (*func == NULL) {
9          fprintf(stderr, "Memory allocation failed\n");
10         return 1;
11     }
12     remove_whitespace(*func, argv[1]);
13     if (!validate_expression(*func)) {
14         free(*func);
15         return 2;
16     }
17     *outfile = argv[2];
18     FILE *test_file = fopen(*outfile, "w");
19     if (test_file == NULL) {
20         fprintf(stderr, "Error: Cannot create file '%s'.\n", *outfile);
21         free(*func);
22         return 3;
23     }
24     fclose(test_file);
25     *x_min = -10.0; *x_max = 10.0; *y_min = -10.0; *y_max = 10.0;
26     if (argc >= 4) {
27         if (sscanf(argv[3], "%lf:%lf:%lf:%lf", x_min, x_max, y_min,
               y_max) != 4) {
28             fprintf(stderr, "Invalid range format\n");
29             free(*func);
30             return 4;
31         }
32     }
33     return 0;
34 }
```

Shrnutí

Modul `main.c` tvoří základní řídicí část programu. Zajišťuje správné načtení vstupních parametrů, jejich validaci a předání do dalších modulů, které generují výsledný graf funkce ve formátu PostScript.

3.3 Validace a parsování výrazu: `parser.c` / `parser.h`

Modul **parser.c** zajišťuje několik klíčových funkcionalit pro validaci a zpracování matematického výrazu:

- **Kontrola závorek** – funkce `check_parentheses()` ověřuje, že každý otevírací znak závorky má odpovídající uzavírací znak. Funkce obsahuje konstantu, která se inkre-

mentuje, když se objeví otevírací znak závorky a dekrementuje se, pokud se objeví uzavírací znak závorky. Pokud je hodnota této konstanty **0**, pak byly všechny závorky uzavřeny.

- **Validace výrazu** – funkce `validate_expression()` kontroluje, že výraz obsahuje pouze povolené znaky, standardní funkce a jedinou proměnnou x.
- **Shunting-yard algoritmus** – převádí infixový výraz do postfixové notace (RPN), která je vhodná pro následné vyhodnocení.

Výsledkem zpracování je validní a optimalizovaný výraz, který může být předán dalším modulům k vyhodnocení a generování grafu.

Funkce `validate_expression()` – Kontrola správnosti výrazu

Jednou z klíčových funkcí v modulu `parser.c` je `validate_expression()`, která zajišťuje kompletní validaci matematického výrazu. Hlavní úlohou této funkce je:

- Ověřit správnost syntaxe výrazu.
- Zajistit přítomnost jediné proměnné x.
- Rozpoznat a validovat všechny povolené funkce jako `sin`, `cos`, `log`, `abs`, atd.
- Detekovat chyby, jako jsou nevyvážené závorky nebo nepovolené znaky.

Ukázka kódu: `parser.c`

Listing 2: Funkce `validate_expression()` v `parser.c`

```
int validate_expression(const char *expr) {
    int variable_found = FALSE;
    int paren_count = 0;

    while (*expr) {
        if (is_valid_character(*expr)) {
            if (*expr == 'x') {
                variable_found = TRUE;
            }
        } else if (isalpha(*expr)) {
            if (!handle_function(&expr, &paren_count, &variable_found))
               {
                return FALSE;
            }
        } else if (*expr == '(' || *expr == ')') {
            if (!handle_parentheses(*expr, &paren_count)) {
                return FALSE;
            }
        } else {
            report_invalid_character(*expr);
            return FALSE;
        }
        expr++;
    }

    if (paren_count != 0) {
```

```
26          report_unmatched_parenthesis();
27          return FALSE;
28      }
29
30      if (!variable_found) {
31          fprintf(stderr, "Error: No variable 'x' found in the expression
                .\n");
32          return FALSE;
33      }
34
35      return TRUE;
36  }
```

Popis funkcionality

- Prochází celý výraz znak po znaku.
- Rozpoznává povolené znaky, operátory a matematické funkce.
- Detekuje syntaktické chyby jako:
  - **Nevyvážené závorky** – pokud počet otevíracích a zavíracích závorek nesouhlasí.
  - **Neplatné znaky** – pokud výraz obsahuje nepovolené znaky.
  - **Více proměnných** – pokud se vyskytují ve výrazu jiné proměnné než x.

Pokud funkce najde chybu, vrátí hodnotu `FALSE` a vypíše odpovídající chybové hlášení.

Shrnutí

Funkce `validate_expression()` je základem pro syntaktickou analýzu matematického výrazu. Ověřuje správnost zápisu, čímž zajišťuje, že do dalších fází programu postoupí pouze korektní výraz, připravený k následnému vyhodnocení.

3.4 Generování PostScript výstupu: `post_script.c` / `post_script.h`

Modul **post_script.c** se stará o generování výstupního grafu ve formátu PostScript. Výsledkem je soubor, který lze zobrazit v kompatibilních prohlížečích nebo vytisknout. Funkce pracuje s validovaným matematickým výrazem a vykresluje graf v kartézské soustavě souřadnic.

Hlavní kroky implementace

1. **Vykreslení souřadnicové soustavy a mřížky.**
2. **Získání hodnot y pro daný interval x** – Body jsou iterativně spočítány a převedeny na souřadnice PostScript.
3. **Vykreslení spojité čáry grafu.**
4. **Omezení nedefinovaných bodů a extrémních hodnot** – Hodnoty mimo definiční obor funkce jsou automaticky ignorovány.
5. **Uložení výsledku do PostScript souboru.**

Klíčová funkce: `generate_postscript()`

Funkce `generate_postscript()` je zodpovědná za vytvoření grafu. Příklady hlavních činností zahrnují:

- Dynamické nastavení měřítka souřadnic podle rozsahu x a y.
- Iterativní vykreslování bodů grafu.
- Přidání os a popisků na obě souřadnice.

Listing 3: Funkce `generate_postscript()` v `post_script.c`

```
1  void generate_postscript(const char *outfile, const char *func, double
       x_min, double x_max, double y_min, double y_max, int calc_x_range,
       int calc_y_range) {
2      FILE *ps_file = initialize_postscript(outfile);
3      Node* expression_tree = parse_expression(&func);
4      double step = 0.001;
5
6      /* Calculate ranges if necessary */
7      calculate_ranges(expression_tree, &x_min, &x_max, &y_min, &y_max,
           step, calc_x_range, calc_y_range);
8
9      /* Draw grid, axes, and the graph */
10     draw_grid(ps_file);
11     plot_graph(ps_file, expression_tree, x_min, x_max, y_min, y_max,
           step);
12     draw_axes_and_labels(ps_file, x_min, x_max, y_min, y_max);
13
14     /* Cleanup */
15     free_tree(expression_tree);
16     fclose(ps_file);
17 }
```

Popis funkcionality

- **Dynamické přizpůsobení rozsahu grafu** – Pokud uživatel nezadá rozsah, funkce automaticky nastaví výchozí interval $[-10, 10]$.
- **Kontrola hodnot** y – Hodnoty mimo definiční obor nebo asymptoty jsou ignorovány.
- **Převod na PostScript souřadnice** – Souřadnice x a y jsou převedeny na formát vhodný pro PostScript.
- **Vykreslování čáry grafu** – Body jsou spojeny spojitou čarou, která reprezentuje matematickou funkci.

Výsledek

Výsledkem je soubor ve formátu PostScript, který obsahuje vykreslený graf zadané matematické funkce. Tento soubor lze otevřít v programech podporujících PostScript, jako jsou Ghostview nebo Adobe Reader.

Klíčovou funkcí je `generate_postscript()`, která zapisuje do souboru příkazy PostScript.

3.5 Pomocné funkce: `utils.c` / `utils.h`

Modul **utils.c** obsahuje soubor pomocných funkcí, které zajišťují základní operace nezbytné pro správné fungování programu. Tyto funkce se zaměřují na úpravu vstupního výrazu, detekci chyb a správu nedefinovaných hodnot (například dělení nulou).

Hlavní funkce modulu

- `remove_whitespace()` – Odstraní všechny mezery ze vstupního řetězce, čímž připraví výraz k validaci a parsování.
- `create_nan()` – Vytvoří hodnotu NaN (Not a Number) pro reprezentaci nedefinovaných výsledků.
- `is_nan()` – Kontroluje, zda je daná hodnota typu NaN.

Funkce `remove_whitespace()`

Funkce `remove_whitespace()` je zodpovědná za odstranění všech mezer z výrazu, což je nezbytné pro zpracování matematického výrazu. Výraz je zpracováván znak po znaku a pouze ne-blank znaky jsou kopírovány do nového řetězce.

Listing 4: Funkce `remove_whitespace()` v `utils.c`

```c
#include <ctype.h>
#include <string.h>

/* Function to remove whitespace from a string */
void remove_whitespace(char* dest, const char* src) {
    while (*src != '\0') {
        if (!isspace((unsigned char)*src)) {
            *dest++ = *src;
        }
        src++;
    }
    *dest = '\0';
}
```

Funkce `create_nan()`

Tato funkce generuje hodnotu `NaN` (Not a Number), která reprezentuje nedefinované výsledky matematických operací, jako je dělení nulou nebo logaritmus záporného čísla. Implementace využívá bitové operace a kopírování binární reprezentace.

Listing 5: Funkce `create_nan()` v `utils.c`

```c
#include <string.h>

double create_nan() {
    unsigned long nan_bits = 0x7FF8000000000000;
    double nan_value;
    memcpy(&nan_value, &nan_bits, sizeof(nan_value));
    return nan_value;
}
```

Funkce `is_nan()`

Funkce `is_nan()` kontroluje, zda je daná hodnota typu `NaN`. Implementace ověřuje specifické bity v binární reprezentaci čísla, které odpovídají standardu IEEE 754 pro NaN hodnoty.

Listing 6: Funkce `is_nan()` v `utils.c`

```
#include <string.h>

int is_nan(double x) {
    unsigned long bits;
    memcpy(&bits, &x, sizeof(bits));
    return (bits & 0x7FF0000000000000) == 0x7FF0000000000000 &&
           (bits & 0x000FFFFFFFFFFFFF) != 0;
}
```

Popis funkcionality

- **Odstranění mezer** – Funkce `remove_whitespace()` je nezbytná pro normalizaci vstupního výrazu, což usnadňuje jeho následnou validaci a parsování.
- **Správa nedefinovaných hodnot** – Hodnoty typu `NaN` jsou využity pro reprezentaci chybných nebo nedefinovaných výsledků. Funkce `create_nan()` a `is_nan()` zajišťují jejich vytvoření a detekci.

Shrnutí

Modul **utils.c** poskytuje jednoduché, ale nezbytné pomocné funkce pro:

- Úpravu vstupního výrazu před jeho validací.
- Bezpečnou detekci nedefinovaných hodnot v průběhu matematických výpočtů.
- Zajištění robustního chodu aplikace v případech, kdy matematické operace vedou k chybám.

3.6 Přenositelnost mezi platformami

Pro přenositelnost mezi Windows a Linux byly vytvořeny dva Makefile soubory:

- `Makefile` – pro Linux.
- `Makefile.win` – pro Windows.

4 Uživatelská příručka

4.1 Překlad a spuštění

Překlad programu:

```
make
./graph "sin(2*x+3)" output.ps
```

5 Závěr

Cílem této semestrální práce bylo vytvoření konzolové aplikace v jazyce ANSI C, která načte matematický výraz, provede jeho validaci a analýzu a následně vygeneruje graf této funkce ve formátu PostScript. Výsledná aplikace splňuje všechny stanovené požadavky, přičemž implementace byla rozdělena do modulárních částí pro lepší přehlednost a udržitelnost kódu.

Hlavní dosažené cíle

- **Validace a analýza matematických výrazů**: Program podporuje širokou škálu matematických funkcí (`sin, cos, tan, log, exp, abs`, atd.) a operátorů ($+, -, \times, \div, \wedge$). Byla implementována validace výrazů, která ověřuje správnost syntaxe, kontroluje přítomnost jediné proměnné x, zajišťuje správné závorkování a odmítá neplatné výrazy.
- **Generování grafu ve formátu PostScript**: Grafické znázornění matematické funkce je vytvořeno pomocí PostScript formátu, který je široce podporovaný a umožňuje přesné vykreslení. Součástí grafu je:
  - Kartézská soustava souřadnic s popisky os.
  - Přizpůsobení měřítka dle hodnot zadaného intervalu.
  - Ignorování nedefinovaných bodů (např. asymptoty).

  Výsledný soubor lze otevřít pomocí standardních nástrojů pro zobrazení PostScript souborů (např. Ghostview, Adobe Reader).

Výhody implementace

- **Modularita**: Program je rozdělen do několika modulů (`parser.c`, `post_script.c`, `utils.c`), což zajišťuje lepší čitelnost a udržovatelnost kódu.
- **Rozšiřitelnost**: V případě potřeby lze snadno přidat další matematické funkce nebo upravit generování výstupu.
- **Robustnost**: Program zvládá různé vstupní formáty, identifikuje chyby ve výrazu a správně ošetřuje extrémní situace (např. dělení nulou, asymptoty).

Možnosti rozšíření

Ačkoli program splňuje zadané požadavky, existují možnosti jeho dalšího vylepšení:

- Přidání podpory pro vykreslování více funkcí současně.
- Rozšíření grafických výstupů o další formáty (např. PNG, SVG).
- Implementace interaktivního uživatelského rozhraní pro zadávání funkcí a intervalů.
- Optimalizace výpočtů pro rychlejší generování grafu při vysokém rozlišení.

Shrnutí

Tato práce demonstruje schopnost implementovat komplexní aplikaci v jazyce C, která kombinuje analýzu matematických výrazů, práci s grafickým výstupem a zajištění kompatibility na různých platformách. Výsledný program je funkční, modulární a připravený pro další rozšíření. Generovaný PostScript graf poskytuje uživatelům přesný a čitelný výstup, který splňuje akademické i praktické požadavky.