Dokumentace k projektu č. 2 do IZP

Iterační výpočty

24. listopadu 2004

Autor: Kamil Dudka, xdudka00@stud.fit.vutbr.cz
Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

Obsah

1. Úvod

Tato dokumentace popisuje návrh a implementaci programu na výpočet matematických funkcí (odmocnina, sinus, e^x a přirozený logaritmus). V kapitole 2 se nachází stručná charakteristika řešeného problému, jeho analýza a z ní vyplývající princip řešení. V této kapitole je přiblížena problematika použití konvergenčních řad pro výpočet matematických funkcí.

Kapitola 3 se zabývá návrhem vlastního řešení, zejména problematikou efektivnosti a konečnosti algoritmů pro výpočet těchto matematických funkcí. Z ohledem na omezení těchto algoritmů byly zvoleny testovací hodnoty aplikace (kapitola 4). Implementaci (kapitola 5) přímo vyplývá z provedené analýzy a návrhu řešení. V závěru (kapitola 6) je zhodnoceno celkové řešení programu.

2. Analýza

2.1. Zadání problému

Bylo požadováno vytvoření programu v jazyce C, který bude počítat matematické funkce **odmocnina, sinus, e^x a přirozený logaritmus** se zadanou přesností. Dále bylo požadováno, aby program načetl z parametrů přepínač, determinující funkci, a přesnost **epsilon**. Přesnost epsilon je maximální přípustný rozdíl vypočtené hodnoty od skutečného součtu konvergenční řady. Vlastní vstupní hodnoty má program načítat ze standartního vstupu a vypočtené hodnoty má program vypisovat na standartní výstup.

2.2 Konvergentní řada

Cílem úlohy bylo odvození iteračních výpočtů z rekurentních vztahů. Rekurentní vztahy pro odmocninu, sinus a e^x byly uvedeny v zadání úlohy. Pro přirozený logaritmus jsem použil vztah z [2]. Nekonečné řady pro výpočet hodnoty matematických funkcí nemusí být konvergentní pro všechna reálná čísla (ani pro celý definiční obor funkce). Pokud je vstupní hodnota mimo **konvergenční interval**, přestává být řada konvergentní a může dojít k zacyklení algoritmu. Proto je potřeba zajistit (např. vhodnou úpravou), aby vstupní hodnoty spadaly do konvergenčního intervalu. Zejména je nutné ošetřit vstupní hodnoty, zdali nejsou mimo definiční obor funkce.

2.3. Věta o konvergenčním intervalu [2]

Ke každé mocninné řadě $\sum_{k=0}^{\infty} a_k x^k$ existuje takové číslo (tzv. poloměr konvergence) $r \geq 0$ (připouštíme i $r = +\infty$), že pro všechna čísla x pro něž $|x| < r$ (tzv. **interval konvergence**) je mocninná řada absolutně konvergentní a pro všechna čísla x pro něž $|x| > r$, je divergentní. V bodech x, pro něž $|x| = r$, nelze obecně rozhodnout, zda mocninná řada je konvergentní nebo divergentní. Je-li $|r| = 0$, pak mocninná řada je konvergentní jen v bodě $x = 0$. Jestliže $r = +\infty$, pak je konvergentní na množině $\mathbb{R}$.

3. Návrh řešení problému

3.1. Definiční obory jednotlivých funkcí

Obor vstupních hodnot je omezen jednak skutečným definičním oborem dané funkce a jednak rozsahem hodnot použitého datového typu. Přímo v zadání byl určen datový typ **double** s plovoucí řádovou tečkou. Tento datový typ je omezen rozsahem i přesností a s tím je nutno v implementaci počítat. Double umí mimo jiné reprezentovat neurčité hodnoty jako je $-\infty, \infty$ a NAN (Not A Number). Součástí výpočtu každé matematické funkce je ošetření vstupní hodnoty, zdali spadá do zvoleného rozsahu přípustných vstupních hodnot.

3.2. Řady pro výpočet jednotlivých funkcí

Některé vztahy (vztah pro odmocninu ze zadání úlohy) jsou dány rekurentně. S ohledem na omezení rekurentních algoritmů a jejich složitost byl zvolen algoritmus využívající **iteraci** (cyklus).

- Odmocnina (rekurentní vztah pro výpočet řady)

  $Y_0 = X \quad , \quad Y_{i+1} = \frac{1}{2} \cdot \left(\frac{X}{Y_i} + Y_i \right)$, kde X je vstupní hodnota

- Sinus (konvergentní řada)

  $$\sin(x) = \frac{x^1}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots \quad , \quad |x| < +\infty$$

- e^x (konvergentní řada)

  $$e^x = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \quad , \quad |x| < +\infty$$

- přirozený logaritmus (konvergentní řada)[2]

  $$\ln(x) = \frac{x-1}{1} + \frac{x-2}{2} + \frac{x-3}{3} + \cdots \quad , \quad 0 < x \leq 2$$

3.3. Efektivita konvergentní řady

Z věty o konvergenčním intervalu (kapitola 2.3.) vyplývá, že řada je konvergentní pro všechna x z konvergenčního intervalu. To však neznamená, že je řada pro všechna x stejně efektivní. Například funkce sinus má **interval nejlepší konvergence** $\langle 0, \frac{\Pi}{4} \rangle$ (s ohledem na rozsah úlohy byl zvolen interval $\langle 0, \frac{\Pi}{2} \rangle$, který je dostačující). Funkce sinus je **periodická**, dá se tady vstupní hodnota snadno upravit na rozsah periody $\langle 0, 2\Pi \rangle$. Dále je možné využít průběhu funkce (funkce sinus je lichá, 1. a 2. kvadrant je osově souměrný podle přímky $x = \frac{\pi}{2}$) a upravit tak vstupní hodnotu na hodnotu z intervalu $\langle 0, \frac{\Pi}{2} \rangle$. Obdobným způsobem jsou upraveny vstupní hodnoty ostatních funkcí.

Vzhledem k tomu že typ double má jen omezenou přesnost, může být tento převod velmi nepřesný, zejména pro vysokou absolutní hodnotu vstupních hodnot (projevuje se u funkce sinus).

4. Specifikace testů

Testoval jsem zejména ošetření parametrů a rozsahu vstupních hodnot. Dále jsem se zaměřil na čísla z okraje definičních oborů a tzv. „nečísla“ neboli NAN. Testy byly provedeny na systému Linux.

- Výpis nápovědy
  `$proj2 -h` (vypíše se nápověda)

- Ošetření 1. parametru
  `$proj2 -cos 0.001` (výpis chybové hlášky na `stderr`)

- Ošetření 2. parametru
  `$proj2 -sqrt ahoj` (výpis chybové hlášky na `stderr`)
  `$proj2 -sin 0` (výpis chybové hlášky na `stderr`)

- Testování funkce **odmocnina**
```
$proj2 -sqrt 1e-12 <<< „-inf 0 2 4 1e30 inf“
nan
0.0000000000e+00
1.4142135624e+00
2.0000000000e+00
1.0000000000e+15
inf
```
  (test proběhl bez problémů)

- Testování funkce **sinus**
```
$proj2 -sin 1e-30 <<<  „-4 0 3.5 10“
7.5680249531e-01
0.0000000000e+00
-3.5078322769e-01
-5.4402111089e-01
```
  (test proběhl bez problémů)

- Testování funkce $\mathbf{e^x}$
```
$proj2 -ex .007 <<< "-inf .75 1e10"
0.0000000000e+00
2.1147460938e+00
inf
```
  (test proběhl bez problémů)

- Testování funkce **ln(x)**
```
$proj2 -ln 1e-12 <<< "-.1 0 1 10 7.456e3 1e50"
nan
-inf
0.0000000000e+00
2.3025850930e+00
8.9167743564e+00
1.1512925465e+02
```
  (test proběhl bez problémů)

5. Popis řešení

5.1. Ovládání programu

Program je řešen jako konzolová aplikace, má tedy pouze textové ovládání. Je volán se dvěma parametry. Prvním parametrem uživatel volí matematickou funkci, jejíž hodnotu bude program počítat. Pro výpis nápovědy je použit parametr `-h` (nápověda se také vypíše pokud je program spuštěn s méně než dvěma parametry). Další možnosti jsou:

`-sqrt`	odmocnina
`-sin`	sinus
`-ex`	e^x
`-ln`	přirozený logaritmus

Druhý parametr je přesnost **epsilon** (viz. Kapitola 2.1). Zadává se buď jako desetinné číslo, nebo v semilogaritmickém tvaru. Přesnost musí být vždy větší než nula, jinak je parametr požadován za neplatný.

Vlastní hodnoty pro výpočet jsou načítány ze standartního vstupu a může jich být nekonečně mnoho (není omezeno vlastním programem). Každá hodnota je předána zvolené funkci a výsledek je zapsán na standratní výstup.

Během samotného výpočtu nejsou vypisovány žádné chybové hlášky. Pokud je vstupní hodnota mimo definiční obor funkce vypíše se `nan`. Pokud je výsledek v nekonečnu, resp. v minus nekonečnu vypíše se `inf`, resp. `-inf`. Pokud je výsledek příliš velký a není možné ho zapsat do proměnné typu double, vrátí funkce nekonečno.

5.2. Vlastní implementace

Aby byl vlastní program přehlednější, vytvořil jsem si hlavičkový soubor `proj2.h`, který zajišťuje výpis nápovědy a chybových hlášek. Dále jsem vytvořil čtyři podprogramy – pro každou matematickou funkci jeden.

Hlavní program zajišťuje rozpoznání prvního parametru, podle kterého nastaví ukazatel `vypocti` na požadovanou funkci. Dále načte přesnost epsilon z druhého parametru a zkontroluje její rozsah. Následuje cyklus `while`, který postupně načítá data ze standartního vstupu, provádí výpočet a vypisuje výsledek v přesně stanoveném tvaru na standartní výstup. Pokud jsou na vstupu neplatná data (data, která nenačte funkce `scanf` s konverzí `%le`), cyklus se ukončí dříve než dojde na konec souboru (znak `EOF`).

Na začátku každé funkce je kontrola vstupní hodnoty (viz. kapitola 3.1). Vstupní hodnota je vždy nejprve upravena na **interval nejrychlejší konvergence** (viz. kapitola 3.4). Výjimku tvoří funkce `odmocnina`, ve které je spuštěna iterace přímo se zadaným číslem. V této části jsou využívány knihovní funkce z `math.h`, jejichž použití bylo omezeno na oblast mimo vlastní iterační výpočet. Jedná se o funkce `sqrt()`, `modf()`, `fmod()` a `exp()`. Výsledný součet řady je upraven tak, aby odpovídal vstupní hodnotě před úpravou na interval nejrychlejší konvergence (podrobně popsáno v následující kapitole).

5.3. Přirozený logaritmus

Výpočet přirozeného logaritmu je realizován ve funkci `lnx`. Této funkci jsou předávány dva parametry datového typu double. První parametr je argument přirozeného logaritmu a druhý je přesnost epsilon. Funkce vrací vypočtenou hodnotu přirozeného logaritmu, rovněž typu double.

Ještě před výpočtem součtu řady, upravuji vstupní hodnotu na interval $\langle \frac{\sqrt{e}}{e}, \sqrt{e} \rangle$. Tím získám získám „celý počet polovin“ výsledku logaritmu. K tomuto účelu jsou použity dva cykly typu `while` – jeden pro vstupní hodnoty $x < \frac{\sqrt{e}}{e}$ a druhý pro hodnoty $x > \sqrt{e}$. Složitost těchto cyklů roste pro $x \to \infty$ a $x \to 0$, ale i pro mezní hodnoty skončí cyklus v rozumném čase.

Konvergenční interval pro použitou řadu je $0 < x \leq 2$, ale pro hodnotu z intervalu $\langle \frac{\sqrt{e}}{e}, \sqrt{e} \rangle$ získám výslednou hodnotu mnohem rychleji při stejné přesnosti.

Vlastní algoritmus pro výpočet řady byl volen s ohledem na jeho efektivnost. Aktuální člen řady je vždy vypočten na základě předchozího členu. Absolutní hodnota aktuálního členu je porovnána s epsilon – pokud je dosažena přesnost, cyklus se ukončí. Aktuální člen je přičten částečnému součtu řady a provede se další iterace. Po skončení cyklu přičte funkce k součtu řady, ještě „celý počet polovin“ (viz. výše).

Uvnitř funkce není nijak kontrolována hodnota epsilon předaná parametrem `eps`. Proto je důležité, volat funkci vždy s $eps > 0$. Pokud by tato podmínka nebyla splněna, stal by se cyklus nekonečný. Rozsah hodnot přípustný pro epsilon je kontrolován v hlavním programu.

6. Závěr

Algoritmy byly voleny co nejefektivněji s ohledem na rozsah úlohy. Přesto je praktická využitelnost programu mizivá. Knihovní funkce `sqrt()`, `sin()`, `exp()` a `log()` totiž využívají koprocesor a jsou tak mnohem rychlejší a přesnější.

Program byl otestován se všemi navrženými testovacími hodnotami a jeho výsledky byly srovnány s knihovními funkcemi z `math.h`. Výsledky testů jsou patrné z kapitoly 4. Program přesně dodržuje požadavky na formát vstupních a výstupních dat, takže může být bezproblémově používán spolu s dalšími programy ve skriptech nebo jiných programech.

7. Metriky kódu

Počet souborů:	2 soubory
Počet řádků zdrojového textu:	282 + 43 řádků
Velikost statických dat:	4604 bytů
Velikost spustitelného souboru:	15091 bytů (systém Linux)

8. Použitá literatura

[1] Vztahy uvedené v zdání úlohy

[2] **Bartsch: Matematické vzorce**