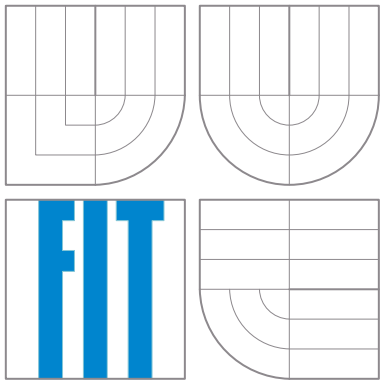


Dokumentace k projektu pro předměty IZP a IUS

Iterační výpočty

projekt č. 2

20. listopadu 2011

Autor: Jan Wrona, xwrona00@stud.fit.vutbr.cz
Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

Obsah

1 Úvod

Iterační výpočty jsou takové výpočty, které se opakují až do splnění určité podmínky. Jejich použití v problémech řešených v tomto projektu je klíčové především z důvodu, že přesný výpočet některých matematických funkcí může být velmi náročný. V první části projektu jsou tyto výpočty použity pro aproximaci daných funkcí, ve druhé části pro výpočet průběžné délky lomené čáry.

Tento dokument popisuje návrh a implementaci aplikace pro výpočet funkce arkus sinus, logaritmu o zadaném základu, pro výpočet délky lomené čáry a délky lomené čáry s chybou. Program funguje jako konzolová aplikace, která ze standardního vstupu čte libovolně dlouhou posloupnost číselných hodnot s plovoucí řadovou čárkou. Část zajišťující výpočet funkcí přečte jednu z těchto hodnot a na standardní výstup vypíše výsledek funkce pro zadanou hodnotu. Část obstarávající výpočet délky lomené čáry přečtě dvě z těchto hodnot jako souřadnice a vzhledem k předchozím souřadnicím vypíše průběžnou délku této čáry, nebo při zadané chybě měření se vypíší minimální a maximální délka čáry.

Dokumentace se skládá z několika částí. V kapitole 2 se věnuji analýze rekurentních problémů, Taylorových řad, také zde objasňuji počítané matematické funkce arkus sinus a logaritmus. Zaměřuji se i na problematiku výpočtu délky lomené čáry. Kapitola 3 se zabývá obecnými algoritmy pro aproximaci zadaných funkcí a pro výpočet délky lomené čáry, více potom pro lomenou čáru s chybou. Součástí této kapitoly je i série testů, které zkoumají funkci programu i při nesprávném použití. V kapitole 4 mimo jiné objasňuji mou konkrétní implementaci celého programu. V příloze A jsou uvedey metriky mého zdrojového kódu a programu.

2 Analýza problému a princip jeho řešení

Samotná iterace je, co se implementace týče, problém triviální. Aby ale iterační výpočet dával nějaký smysl a vedl k rozumným výsledkům v rozumném čase, musí být vhodně zvoleno tělo cyklu, tedy ta nejdůležitější část algoritmu, která vede k správnému výsledku. Nemenší problém je správné zvolení ukončovací podmínky této iterace. Důležité je samozřejmě také rozumnět všem funkcím, kterýma se budu v tomto projektu zabývat. Proto se v této kapitole budu věnovat především analýze těchto problémů.

2.1 Zadání problému

Cílem tohoto projektu je vytvoření programu v jazyce C, který pomocí základních matematických operací aproximuje funkce arkus sinus a logaritmus pro libovolně dlouhou vstupní posloupnost hodnot. Program musí umět zohlednit přesnost výpočtu zadávanou jako počet desetinných míst. Druhá část zajišťuje výpočet délky lomené čáry a lomené čáry s chybou. Chyba je zde zadávána jako absolutní.

2.2 Rekurentní problémy

Pro aproximaci některých matematických funkcí se často používá takový druh iterace, při které hodnota aktuálně počítaného kroku závisí až na několika hodnotách předchozích kroků. Tento speciální případ iterace se nazývá rekurze a tudíž problémy, které se toho týkají, nazýváme

problémy rekurentní a obecný rekurentní vztah [3] vypadá následovně:

$$Y_{n+1} = F(Y_n, Y_{n-1}, \ldots, Y_{n-k}) \tag{1}$$

kde Y jsou proměnné vyjadřující hodnotu n-tého kroku a F je funkce pro samotný výpočet následujícího kroku. I funkce pro arkus sinus a logaritmus, které jsou řešeny v tomto projektu je možno aproximovat tímto způsobem, proto se problémem budu dále zabývat.

Jeden z možných způsobů řešení je pomocí nekonečných řad. To je matematický výraz s obecným vztahem

$$\sum_{n=1}^{\infty} a_n = a_1 + a_2 + a_3 + \ldots \tag{2}$$

kde n je index kroku, a je posloupnost čísel v případě číselné posloupnosti, ale třeba také funkcí v případě posloupnosti funkční. Z těchto obecných vztahů lze vyjádřit, pro mé výpočty velmi důležitý, rekurentní vztah pro výpočet hodnoty následujícího členu řady záviesjícího na hodnotě členu předchozího:

$$t_n = F(t_{n-1}), i > 0 \tag{3}$$

Aby takový rekurentní vztah mohl začít počítat, je potřeba nejprve inicializovat první člen řady, ať už na nulu, nebo na jakoukoliv jinou hodnotu dle zvolené řady.

2.3 Taylorovy řady

Jednou z možností pro aproximaci určitých funkcí je využití Taylorových řad. Jedná se o zvláštní případ řady mocninné, což je funkční řada, která má ve svých členech mocninu. Definice Taylorovy řady zní následovně [1]. V případě existence všech derivací funkce f v bodě a lze Taylorovu řadu zapsat jako:

$$f(x) = \sum_{n=0}^{\infty} a_n a^x \tag{4}$$

Jelikož se jedná a řadu nekonečnou, není možné vyjádřit všechny členy řady, a proto se při aproximaci funkce využívá pouze určitý počet členů řady. Takto upravená Taylorova řada, kde využíváme konečný počet členů řady se nazývá Taylorův polynom (mnohočlen).

2.4 Funkce arkus sinus

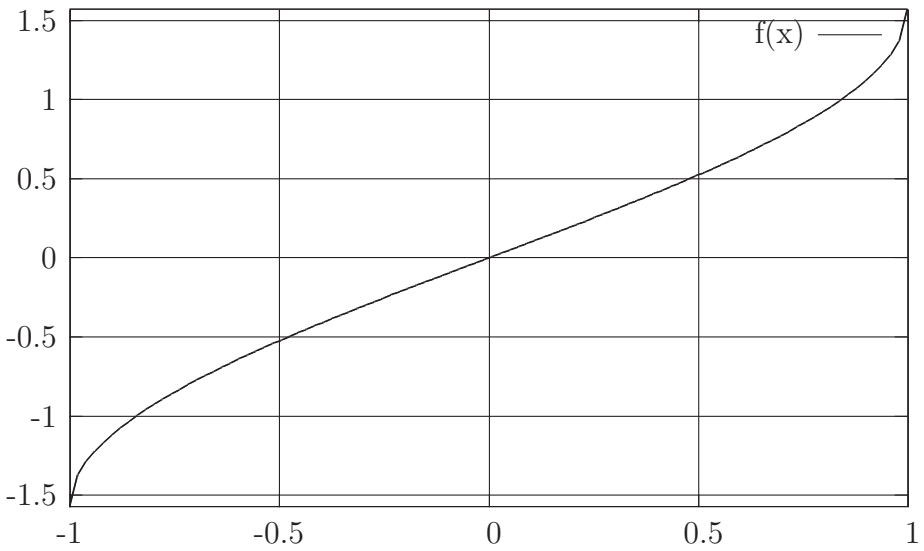
Arkus sinus (značená též jako arcsin, asin) je funkce cyklometrická, tzn. funkce inverzní k funkci goniomterické. V tomto případě tedy hovořím o funkci arkus sinus, která je inverzní k sinus a naopak. Definiční obor funkce je $\langle -1, 1 \rangle$, obor hodnot je $\langle -\frac{\pi}{2}, \frac{\pi}{2} \rangle$. Funkci arkus sinus, tak jako jiné cyklometrické funkce, lze vyjádřit pomocí jiných cyklometrických funkcí stejného argumentu [1]:

$$arcsin(x) = \frac{\pi}{2} - arccos(x) \tag{5}$$

pro definiční obor ($|x| \leq 1$), nebo také:

$$arcsin(x) = arctan\frac{x}{\sqrt{(1-x^2)}} \tag{6}$$

pro definiční obor ($|x| < 1$). Graf cyklometrických lze sestrojit z grafů goniometrických funkcí na intervalech, kde jsou monotónní. Funkce sinus je monotónní v intervalu $\langle -\frac{\pi}{2}, \frac{\pi}{2} \rangle$, graf funkce arkus sinus znázorňuje obrázek 1



Obrázek 1: Graf funkce arkus sinus

2.5 Funkce logaritmus

Logaritmus je matematická funkce inverzní k funkci exponenciální. Logaritmus zapisujeme jako

$$y = \log_a x \tag{7}$$

kde x je logaritmované číslo (někdy také numerus) a číslo a ($a \in R^+ - \{1\}$) označuje zálkad logaritmu (někdy také báze). Logaritmus o základu 10 nazýváme desítkový (značíme ho $\log x$, desítka se často nepíše), o základu e[1] nazýváme přirozený (ten je také možno značit jako $\ln x$). Je možno odvodit a zjistit spoustu různých vlastností a vztahů pro logaritmus, uvedu zde ale jen ty pro můj projekt podstatné, jako například tento vztah [2] pro vyjádření logaritmu o jakémkoliv základu pomocí dvou desítkových nebo přirozených logaritmů:

$$\log_a x = \frac{\log x}{\log a} = \frac{\ln x}{\ln a} \tag{8}$$

Pokud přepočítáme číslo jako mantisu m o exponentu e, lze desítkový logaritmus vyjádřit pomocí tohoto vztahu:

$$\log x = \log_m + e \tag{9}$$

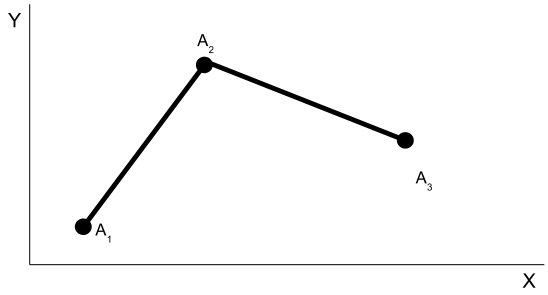
2.6 Lomená čára

Lomená čára je spojená série úseček. Pokud máme v rovině body $(A_1, A_2, \ldots, A_n)$, tak úsečky jejichž koncové body jsou počátečními body dalších úseček $(A_1A_2, A_2A_3, \ldots, A_{n-1}A_n)$ tvoří lomenou čáru. Pro představu uvádím příklad lomené čáry na obrázku 2.

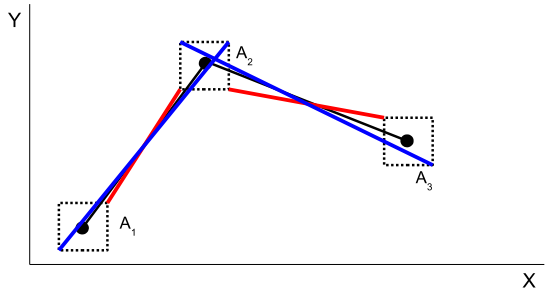
2.7 Lomená čára s chybou

Lomená čára s chybou je v podstatě totožná s klasickou lomenou čárou. Rozdíl je ten, že po zohlednění této chyby se už nebude jednat o body v rovině, ale o množinu bodů, a pokud by

[1] Eulerovo číslo, jeho hodnorta je 2,71828...



Obrázek 2: Lomená čára



Obrázek 3: Lomená čára s chybou. Černou barvou je vykreslena původní čára bez chyby, červenou čárou nejkratší možná délka a modrou je vykreslena nejdelší možná délka čáry.

chyba byla zadána totožně pro obě souřadnice, jednalo by se o množinu představující čtverec. Jak plyne z obrázku 3, po aplikaci této chyby vznikne nekonečný počet lomených čar, budu se zde ale zabývat pouze tou s nejmenší možnou a největší možnou délkou.

2.8 Rozsah hodnot

Rozsah vstupních hodnot je v případě obou funkcí jasný. U arkus sinus se jedná o definiční obor této funkce, který je v uveden v podkapitole 2.4. Také logaritmus je definován pouze pro hodnoty větší než nula a základ logaritmu mohou být pouze kladná reálná čísla vyjma jedničky. U vstupních hodnot pro lomené čáry platí omezení pouze rozsahem použitého datového typu.

3 Návrh řešení problému

3.1 Volba datových typů

U veškerých proměnných s plovoucí řadovou čárkou jsem zvoli datový typ `double` z důvodu většího rozsahu možných hodnot. Vstupní hodnoty je tedy možno zadávat v rozsahu přibližně $1.7e^{\pm 308}$. Stejný rozsah platí i pro hodnoty výsledné. U proměnných bez plovoucí řadové čárky jsem volil datový typ `int`. S takovýma proměnnýma se v tomto programu neprovádí žádné zásadní výpočty, jedná se tedy většinou o různé pomocné proměnné u kterých rozsah není podstatný.

3.2 Výpočet arkus sinus

Pro aproximaci funkce arkus sinus jsem zvolil jedno z možných vyjádření této funkce pomocí funkce jiné. Jedná se konkrétně o vztah 6. Jak je také uvedeno v podkapitole 2.4, tento vztah je možné využít pouze pro aproximaci hodnot v intervalu $(-1, 1)$, kdežto definiční obor funce arkus sinus je $\langle -1, 1 \rangle$. Proto je nutné zohlednit případy, kdy je požadován výpočet funkce s parametrem 1 nebo -1. Pro aproximaci arkus sinus je tedy potřeba aproximovat funkci arkus tanges a upravit její parametr. Proto budu nyní popisovat realizaci funce arkus tangens. Taylorův rozvoj pro výpočet arkus tangens [1]

$$\sum_{n=0}^{\infty}(-1)^n \frac{x^{2n-1}}{2n+1} \tag{10}$$

jsem upravil tak, aby každý další krok mohl být vyjádřen rekurentně pomocí předchozího kroku:

$$t_n = t_{n-1} \cdot (-\frac{x^2(n-1)}{n+1}) \tag{11}$$

Tímto se podstatně sníží náročnost výpočtu jednoho kroku. První krok je potřeba inicializovat, u tohoto Taylorova rozvoje je hodnota prvního kroku shodná s hodnotou zadanáho parametru. Postupný součet jednotlivých kroků se ukládá do proměnné. Tento rekurentní vztah se tedy opakuje v cyklu s ukončovací podmínkou, kterou rozebírám v samostatné podkapitole 3.4. Vzhledem k této Taylorově řadě jsem z důvodů zvýšení efektivity algoritmu upravil zadaný parametr dvěma způsoby. Když je záporný, tak se neguje, algoritmus počítá s číslem kladným a neguje se až výsledek. Další úprava invertuje parametr, pokud je větší než 1 a výpočítaný výsledek odečte od $\frac{\pi}{2}$.

3.3 Výpočet logaritmu

Pro výpočet přirozeného logaritmu jsem zvolil vztah 8, takže nadále budu popisovat funkci, která aproximuje přirozený logaritmus. V rámci heuristiky je nejprve potřeba patřičně upravit zadané logaritmované číslo. To se provede jeho rozdělením na exponent a mantisu tak, aby mantisa ležela vždy v intervalu $\langle 1, 10 \rangle$, přičemž se s těmito čísly nadále pracuje podle vztahu 9. Tento vztah je pro desítkový logaritmus, můj algoritmus ale počítá přirozený logaritmus, proto jsem si tento vztah patřičně upravil:

$$ln(x) = ln(m) + e \cdot ln(10) \tag{12}$$

Logaritmus má omezený definiční obor, je tedy potřeba ošetřit hodnoty logarotmových čísel ležících mimo definiční obor pomocí hodnot NAN a INFINITY. Taylorův rozvoj pro přirozený logaritmus

$$\sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} \cdot (x-1)^n \tag{13}$$

jsem upravil do podoby rekurentního vztahu:

$$t_n = t_{n-1} \cdot (\frac{(1-n)(x-1)}{n}) \tag{14}$$

Inicializace prvního kroku zde podle Taylorovy řady bude $x - 1$. Tento Taylorův rozvoj ale poskytuje přesnou aproximaci pouze v intervalu $(0, 2\rangle$, proto je pro výpočet logaritmu většího než 2 je řada přepočítaná pomocí Eulerovy transformace do podoby:

$$ln\frac{x}{x-1} = \sum_{n=1}^{\infty} \frac{1}{nx^n} \tag{15}$$

Aby ale počítal pro zadané logaritmové číslo x a ne pro $\frac{x}{x-1}$, je potřeba upravit x do podoby $\frac{x}{x-1}$, potom po dosazení:

$$\frac{\frac{\frac{x}{x-1}}{\frac{x}{x-1}-1}}{} = x \tag{16}$$

Vztah 15 vyjádřený rekurentně vypadá následovně:

$$t_n = t_{n-1} \cdot (\frac{n-1}{nx}) \tag{17}$$

kde inicializace prvního kroku bude $\frac{1}{x}$. Oba dva tyto rekurentní vztahy jsou umístěny v cyklech s ukončovací podmínkou, kterou rozebírám v samostatné podkapitole 3.4.

3.4 Ukončovací podmínka iteračních výpočtů

Požadovaná přesnost u arkus sinus a logaritmu je v tomto projektu zadávána relativně. S ohledem na to, že Taylorovy řady nejsou pro celý definiční obor stejně přesné, počet iterací nebude pro každou vstupní hodnotu stejný. Ukončovací podmínku tedy není možné zadat jako pevný počet iterací, ale musí být také zadána relativně, relativně vzhledem k výsledku.

Jedno z možných řešení tohoto problému je umístit do podmínky tento vztah:

$$|t_n| > |\sum t| * sigdig \tag{18}$$

Přesnost sigdig zadaná jako počet desetinných míst by se ale do tohoto vzahu nehodila, proto se musí přepočítat jako $10^{-sigdig}$.

3.5 Výpočet délky lomené čáry a lomené čáry s chybou

Jak je zmíněno v analýze, lomená čára je série úseček. Proto pro výpočet její délky je potřeba znát délky těchto úseček. Jednu úsečku v rovině si lze představit jako přeponu trojuhelníku. Výpočtem rozdílu zadaných souřadnic lze získat délku jeho odvěsen, a přeponu tedy vypočítáme pomocí Pythagorovy věty $c^2 = a^2 + b^2$. Součet jednotlivých přenom je roven délce lomené čáry.

U lomené čáry s chybou je situace komplikovanější. Nejprve je potřeba vytvořit intervaly možných hodnot souřadnic, to se provede odečtením/přičtením chyby od/ke každé souřadnici. Pro získání nejmenší a největší vzdálenosti těchto intervalů jsem použil vztah intervalové aritmetiky pro odečítání:

$$(a, b) - (c, d) = (a - d, b - c) \tag{19}$$

který jsem pro své potřeby upravil jako:

$$(a, b) - (c, d) = (d - a, c - b) \tag{20}$$

Tuto intervalovou aritmetiku provádím s intervaly vzniklými aplikací chyby na vstupní souřadnice. Výsledné dva intervaly určují největší a nejmenší vzdálenost těchto intervalů podle osy x a y. Následná nejkratší a nejdelší úsečka se z těchto hodnot vypočítá pomocí Pythagorovy věty. Mohou ale nastat určité situace, při kterých je výsledek jiný. Pokud jsou obě nejmenší délky (jak pro x tak pro y) menší než 0, budou se pomyslné čtverce překrývat a nejmenší chyba bude 0[2]. Pokud bude alespoň jedna z nejmenších délek menší než 0, překrývají se intervaly pouze podle jedné osy a nejmenší délka tudíž není vzálenost rohů těchto pomyslných čtverců, ale vzdálenost jejich bližších stran.

3.6 Specifikace testů

Z návrhu řešení vyplývá několik rizikových oblastí, které je potřeba otestovat, jako například chybný rozsah vstupních hodnot, chybně zadaná chyba či přesnost pomocí příkazové řádky, vstupní hodnota mimo definiční obor funkce nebo vstupní hodnota větší než je rozsah datového typu `double`.

Test 1: Chybně zadaná přesnost sigdig ⟶ Detekce chyby.

```
-3
3a
abc
```

Test 2: Chybně zadaná chyba ERR ⟶ Detekce chyby.

```
-4
4b
def
```

Test 3: Základ logaritmu neodpovídá definici ⟶ Výpis NAN

```
-2
-45
```

Test 4: Chybně zadaný základ logaritmu ⟶ Detekce chyby.

```
a
-3s
ghk
```

[2]Bez tohoto ošetření by mohla vzniknout lomená čára o záporné délce.

Test 5: Vstupní data přesahující rozsah datového typu double ⟶ Detekce chyby.

```
3e309
2e-309
```

Test 6: Chybějící parametr na příkazové řádce ⟶ Detekce chyby.

```
--logax 3 <input.txt
--arcsin <input.txt
--lble <input.txt
```

Test 7: Příliš mnoho parametrů na příkazové řádce ⟶ Detekce chyby.

```
--logax 1 2 3 <input.txt
--arcsin 1 2 <input.txt
--lbl 1 <input.txt
```

Test 8: Vstupní data mimo definiční obor funkce arkus sinus ⟶ Výpis NAN.

```
-2
2
a
```

Test 9: Správnost výpočtu logaritmu ⟶ Předpokládaná správná hodnota.

vstup při `--logax 5 5`	očekávaný výstup
`5`	1.0000000000e+00
`1`	0.0000000000e+00
`3.648`	8.0411877703e-01
`12345`	5.8536022399e+00

Test 10: Správnost výpočtu délky lomené čáry s chybou ⟶ Předpokládaná správná hodnota.

vstup při `--lble 5`	očekávaný výstup
`0 0`	0.0000000000e+00
	0.0000000000e+00
`10 20 30 40 50`	0.0000000000e+00
	0.0000000000e+00
	1.4142135624e+01
	4.2426406871e+01
	nan

4 Popis řešení

Při implementaci jsem vycházel ze závěrů popsaných v kapitolách 2 a 3.

4.1 Ovládání programu

Program funguje čistě jako konzolová aplikace, má pouze textové ovládání. Základní fungování programu se ovládá pomocí parametrů, tedy z příkazové řádky ještě před spuštěním programu. Tyto parametry jsou:

`--arcsin sigdig` - pro výpočet funkce arkus sinus se přesností na určitý počet desetiných míst zadaných parametrem sigdig

`--logax sigdig a` - pro výpočet obecného logaritmu se základem a s přesností na určitý počet desetinných míst zadaných parametrem sigdig

`--lbl` - pro výpočet délky lomené čáry

`--lble ERR` - pro výpočet nejmenší a největší délky lomené čáry s absolutní chybou zadanou parametrem ERR

`-h` - neprovádí žádné výpočty, pouze se vypíše nápověda. Při spuštění bez parametrů je funkčnost stejná jako s parametrem `-h`, tedy výpis nápovědy.

Po spuštění program očekává na standardním vstupu sérii číselných hodnot od sebe navzájem oddělených libovolným počtem bílých znaků. Program se ukončí při dosažení konce vstupního souboru, nebo při některé z chyb. Pokud na vstupu narazí na nenumerické znaky, vypíše na výstup hodnotu NAN, tento znak přeskočí a nadále pokračuje dalším znakem, tedy se neukončí. Při špatně zadaných parametrech (chybějící sigdig, nadbytečný parametr, ...) program vypíše chybové hlášení na stderr a svou funkci ukončí.

4.2 Vlastní implementace

Parametry příkazové řádky jsou zpracoány ve funkci `main`. Pokud jsou parametry v pořádku a je rozpoznán jeden z pěti hlavních parametrů (viz. podkapitola 4.1), jsou zkontrolovány případné další parametry (sigdig, základ logaritmu, ERR) a ošetřeno jejich chybné zadání. Hodnota sigdig je přepočítána funkcí `prepocet_sigdig`. Následně funkce v nekonečném cyklu načítá hodnoty ze vstupu, uloží hodnotu do proměnné, nebo v případě lomených čar uloží dvě ze vstupních hodnot do struktury `Tcoordinates`. Dále cyklus provádí volání požadované funkce a výpis hodnot. Požadovaná funkce je volána dle přepínače, a mohou to být funkce `myArcsin` pro arkus sinus, `myLog` pro logaritmus, `lbl` pro lomenou čáru nebo `lble` pro lomenou čáru s chybou. Tyto funkce jsou implementovány přesně tak, jak je detailně popsáno v kapitole 3.

Výpis chybových hlášení je realizováno voláním funkce `error`, které je předáván chybový kód. Tyto chybové kódy jsou definovány výčtovám typem `ecodes`. Podle odpovídajícího kódu je vypsána příslušná chybová hláška na standardní chybový výstup stderr. Dále funkce kontroluje hodnotu chybové proměnné `errno` z knihovny `errno.h` a podle její hodnoty také vypíše odpovídající chybovou hlášku.

5 Závěr

Program počítá funkce arkus sinus a logaritmus, dále počítá délku lomené čáry a minimální a maximální délku lomené čáry s chybou. Vstupní hodnoty mimo definiční obor jsem řešil podle

vzoru standardních matematických knihovních funkcí `asin` a `log`. Podle těchto funkcí jsem také prováděl kontrolu a s ohledem na zadanou přesnost program pracoval bezchybně.

Program byl úspěšně otestován v prostředí operačního systému Gnu/Linux 32 i 64 bitové architektury se všemi navrženými testovacími hodnotami. V obou případech jde program bez problému přeložit a bez problému také funguje. Věškeré požadavky na formát vstupních i výstupních dat jsou dodrženy. Také rychlost konvergence je přijatelná a při zadání sigdig menší než 100 program vždy končí v rozumném čase.

Reference

[1] BARTSCH, H. J.: *Matematické vzorce*. Praha: SNTL - Nakladatelství technické literatury, první vydání, 1983.

[2] MIKULČÁK, J.; KLIMEŠ, B.; ŠIROKÝ J.; ŠŮLA, V.; ZEMÁNEK, F.: *Matematické, fyzikální a chemické tabulky pro střední školy*. Praha: Státní pedagogické nakladatelství, 1989.

[3] *Studijní opora k předmětu IZP*. 2010.

A Metriky kódu

Počet souborů: 1 soubor

Počet řádků zdrojového textu: 584 řádků

Velikost statických dat: 696B

Velikost spustitelného souboru: 17566B (systém Linux, 64 bitová architektura, při překladu bez ladicích informací)

Počet funkcí: 9