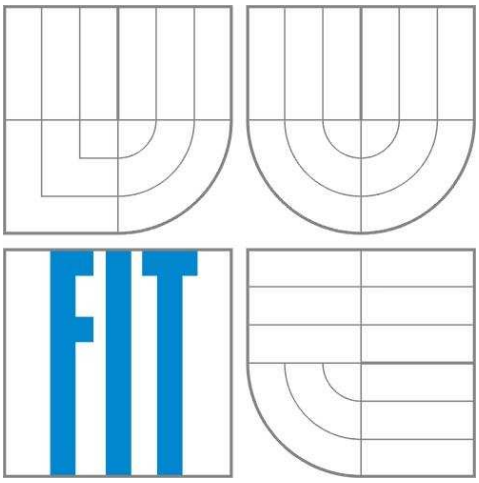

Dokumentácia k projektu pre predmety IZP a IUS

Iteračné výpočty

projekt č. 2

11. decembra 2010

Autor: Pavol Loffay, xloffa00@stu.fit.vutbr.cz

Fakulta Informačných Technologií

Vysoké Učení Technické v Brně

Obsah

1 Úvod 3

2 Analýza problému a princíp riešenia 3

2.1 Zadanie úlohy 3

2.2 Aproximácia funkcií 3

2.3 Newtonová metóda dotyčníc 4

2.4 Nekonečné rady 4

2.5 Statické funkcie 5

3 Návrh riešenia úlohy 5

3.1 Funkcia logaritmus 5

3.2 Funkcia hyperbolický tangens 6

3.3 Vážený aritmetický priemer 7

3.4 Vážený kvadratický priemer 8

3.5 Analýza vstupných hodnôt 8

3.6 Počet platných cifier 9

4 Špecifikácia testov 9

5 Popis riešenia 10

5.1 Ovládanie programu 10

5.2 Dátové typy 10

5.3 Vlastná implementácia 10

6 Záver 10

Použitá Literatúra 11

A. Metrika kódu 11

1 Úvod

Tento dokument popisuje návrh a implementáciu aplikácie pre výpočet logaritmu a hyperbolického tangensu. Taktiež je tu popísaná implementácia pre statické funkcie vážený aritmetický a kvadratický priemer. Ďalej rozoberá odvodenie rekurentných vzťahov pre uvedené matematické funkcie. Dokument taktiež popisuje optimalizáciu výpočtu daných funkcií.

Je rozdelený do niekoľkých tematických častí. V kapitole 2 sa venuje analýze daného problému s možnosťami jeho riešenia. Kapitola 3 sa zaoberá popisom vlastného návrhu riešenia vzhľadom na efektívnosť aplikácie. Kapitola 4 sa venuje testovaniu aplikácie s rôznymi vstupnými hodnotami, aby demonštrovala jej správnosť. V kapitole 5 je rozpísaná konkrétna implementácia riešenia a ovládanie programu. Kapitola 7 obsahuje záver s zhodnotením použiteľnosti vytvorenej aplikácie.

2 Analýza problému a princíp riešenia

2.1 Zadanie úlohy

Cieľom úlohy je implementácia iteračných algoritmov v programovacom jazyku C. Program ma spracovávať ľubovoľne dlhú postupnosť číselných hodnôt typu double. Ktoré budú v textovej podobe zapísané na štandardný vstup a budú oddelené ľubovoľne dlhou postupnosťou bielych znakov. Výstupom programu bude číselná postupnosť výsledkov.

Algoritmus ma pomocou základných matematických operácii (+, -, *, /) aproximovať hyperbolický tangens $y = \tanh x$ a obecný logaritmus $y = \log_a x,\ a, x \in \mathbb{R}^+ \backslash \{1\}$. Výsledná postupnosť hodnôt bude rovnako dlhá ako vstupná. Pre tieto funkcie sa bude zadávať na príkazovom riadku parameter sigdig, čo znamená, na aký počet cifier budú výstupné hodnoty presné.

Program má ďalej spracovávať statické funkcie vážený aritmetický a kvadratický priemer. V tomto prípade bude výsledná postupnosť tvorená ich priebežnými výsledkami.

Program sa ma ovládať parametrami príkazového riadku, ktorými užívateľ zvolí, aká funkcia sa bude aplikovať na vstupné hodnoty.

2.2 Aproximácia funkcií

Vyčísľovanie hodnôt zadaných matematických funkcií sa dá aproximovať súčtom nekonečných radov alebo Newtonovov metódou dotyčníc. Tieto matematické výrazy je úž triviálne previesť do podoby rekurentných funkcií. Samozrejme, že po prevode daného nekonečného súčtu sa dostaneme k iteračnému algoritmu. Hlavnými problémovými doménami sú rýchlosť konvergencie rady a odvodenie správneho definičného oboru v ktorom rada najlepšie konverguje. Taktiež je dôležité zistiť kedy sa má samotný algoritmus zastaviť. Minimálna požadovaná presnosť je zadávaná, ako počet platných cifier výsledku.

2.3 Newtonova metóda dotyčníc

Definícia metódy dotyčníc:

(1, str. 223-224) „Ak v rovnici $f(x) = 0$ je f funkcia, ktorá má na intervale $\langle a, b\rangle$ spojitú druhú deriváciu, $f(a)f(b) < 0$, derivácia f' a f'' sú na intervale $\langle a, b\rangle$ rôzne od nuly a $x_1 \in \langle a, b\rangle$ je taký bod, že $f(x_1)f''(x_1) > 0$, tak prvá aproximácia koreňa rovnice $f(x) = 0$ je bod $x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$.

Po výpočte aproximácie x_2 sa dajú korene znova použiť na nájdenie lepšej aproximácie koreňa. Pričom pri každom ďalšom kroku sa počet správnych číslic aproximácie koreňa prakticky zdvojnásobí.“

Uvediem príklad, ako by sa dala táto metóda implementovať pre výpočet prirodzeného logaritmu:

Chcel by sme vypočítať koreň rovnice $f(x) = e^x - a$, ak $f'(x) = e^x$

$$x_{n+1} = x_n - \frac{e^{x_n} - a}{e^{x_n}}$$

Z tohto riešenia plynie, že v cykle by sme museli pri výpočte každej aproximácie nového koreňa počítať e^{x_n} čo považujem za neefektívny spôsob.

Pri výpočte hyperbolického tangensu by som dostal nižšie uvedený vzťah, z ktorého sa však nedá odstrániť $argtgh(x)$.

$$x_{n+1} = x_n - \frac{argtgh(x) - a}{\frac{1}{1 - x^2}}$$

Preto považujem túto metódu za nevhodnú.

2.4 Nekonečné rady

Zo zadania projektu bolo jasné, že jedno z riešení bude spočívať v súčte nekonečnej rady. Otázka je či súčet rady bude plne odpovedať možnostiam riešenia: použitie matematických operácií (+, -, *, /).

Definícia nekonečného radu

(1, str. 962) „Ak je $\{a_k\}$ číselná postupnosť, tak nekonečný číselný rad (stručne číselný rad alebo rad) nazývame výraz $a_1 + a_2 + \cdots + a_m + \cdots = \sum_{k=1}^{\infty} a_k$, ktorý často stručne označujeme $\sum_1^{\infty} a_k$ alebo $\sum a_k$. Sčítanec a_m sa nazýva n-tý člen rady. Výraz $s_n = \sum_{k=1}^{n} a_k = a_1 + a_2 + \cdots + a_n$ sa nazýva n-tý čiastočný súčet rady $\sum_{k=1}^{\infty} a_k$.“

Ešte musím podotknúť, že súčet nekonečného radu môžeme aplikovať, ak je rada konvergentná.

(1, str. 962) „ Hovoríme, že rada $\sum_{k=1}^{\infty} a_k$ je konvergentná [konverguje], ak existuje limita $\lim_{n\to\infty} s_n$ postupnosti čiastočných súčtov $s_n = \sum_{k=1}^{n} a_k$ a má konečnú hodnotu s, ktorá sa nazýva súčet (konvergentného) radu.”

Takže výpočet logaritmu a hyperbolického tangensu budem aplikovať pomocou súčtu nekonečného radu. Program dostane ako parameter počet platných číslic výsledku, takže sa bude v praxi počítať čiastočný súčet rady $\sum_{k=1}^{\infty} a_k$, pričom dopredu nepoznám počet členov postupnosti. To bude závisieť od parametra sigdig.

2.5 Statické funkcie

Základné vzorce pre výpočet váženého aritmetického a kvadratického priemeru potrebujú pre výpočet všetky predošlé prvky postupnosti. Postupnosť však môže byť nekonečná. Z toho vyplýva, že je nutné upraviť tieto vzorce do podoby pre iteračný algoritmus. To znamená upraviť ich pomocou matematických úprav na rekurentné vzťahy. S takto upravenými výrazmi môžeme vyčísliť nový člen výslednej postupnosti zo starého s vstupom nových prvkov. Dôležité je taktiež uchovávať niektoré hodnoty, aby sme mohli použiť nami upravený vzorec.

3 Návrh riešenia problému

Z uvedenej analýzy vyplýva, že hlavnou úlohou bude vhodná úprava matematického výrazu pre použitie v iteračnom algoritme. Dôležitou súčasťou riešenia je taktiež následné zefektívnenie algoritmov, aby sa nevykonávali zbytočné výpočty. Program sa má ovládať pomocou parametrov príkazového riadku. Tak je nutné hneď na začiatku zistiť, či sú zadané parametre korektné a program prepnúť, aby vykonával požadovanú činnosť. Ak nie, tak program ukončiť s chybovým hlásením.

3.1 Funkcia Logaritmus

Predpis funkcie je v tvare:

$$y = \log_a x\,,\ a \in \mathbb{R}^+\{1\}, x \in \mathbb{R}^+ \tag{3.1}$$

Pre výpočet logaritmus sme použili nasledovný vzťah (1, str. 703):

$$\ln x = \sum_{i=1}^{\infty} \frac{(-1)^{i+1} * (x-1)^i}{i} = \frac{x-1}{1} - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \cdots \quad (0 < x \leq 2) \tag{3.2}$$

Vyššie uvedený vzorec platí pre prirodzený logaritmus, tak urobíme úpravu (1, str. 703):

$$y = \log_a x = \frac{\ln x}{\ln a} \tag{3.3}$$

Vytvorenie rekurentného vzťahu

Zo súčtu nekonečnej rady (vo vzorci 3.2) vytvorím rekurentný vzťah. Vidíme, že vo výraze sa opakuje $(x-1)^n$, tak zavediem substitúciu:

$$Y_i = (x-1) \tag{3.4}$$

Ďalší člen postupnosti vypočítam:

$$Y_{i+1} = \frac{(-1 * Y_i * (i-1))}{(i+1)} \tag{3.5}$$

Aby som výraz nemusel stále násobiť -1, tak zavediem ďalšiu substitúciu:

$$z = -1 * Y_i = 1 - x \tag{3.6}$$

Dostanem vzťah:

$$Y_{i+1} = z * \frac{i-1}{(i)} \tag{3.7}$$

Algoritmus bude mať nasledovnú podobu:

$$i = 1,\ Y_i = x - 1, \quad z = -1 * Y_i, \quad suma = Y_i$$

V cykle budem inkrementovať $i = i + 1$

$$Y_i = Y_{i-1} * z * \frac{i-1}{(i)}$$

$$suma = \ suma + Y_i$$

Optimalizácia

Vzhľadom k podmienke $(0 < x \leq 2)$ zo vzorca 3.2 je nutné upraviť hodnotu x. Lenže som zistil, že v intervale $(0.5; 1.5)$ sa dostanem k presnému výsledku veľmi malým počtom iterácií. Čiže program bude efektívnejší a presnejší.
Použijem nasledovné vzorce (1, str. 703):

$$\log_z a * b = \log_z a + \log_z b, \qquad \log_z \frac{a}{b} = \log_z a - \log_z b \tag{3.8}$$

Ak je číslo x väčšie ako 1.5 tak ho podelím e a dostanem $x_{nove} = \frac{x}{e}$ a k výsledku $suma$ pripočítam koľko krát sme číslo x delili e.

$$\ln x = \ln \frac{x}{e} + 1, \ pretože \ \ln e = 1 \tag{3.9}$$

Ak je číslo x menšie ako 0.5 tak postupujem analogicky. Číslo x násobím konštantou e a od $suma$ odpočítame počet koľko krát sme násobili.

Posledným problémom, ktorý môže nastať pri logaritme sú hodnoty x mimo definičný obor. Ak bude zadaná hodnota $x = 0$, tak výstupom bude –INFINITY. Ak $x < 0$, tak výstup bude NaN (Not a Nubmer).

3.2 Funkcia hyperbolický tangens

Pri návrhu algoritmu na výpočet funkčnej hodnoty hyperbolického tangensu som prišiel na dva riešenia.

Prvým z nich je vyčísliť výsledok vzťahom $\tanh x = \sinh x \,/\, \cosh x$. Avšak tento postup by bol efektívny v intervale $(-1, 1)$, kde nekonečné rady pre $\sinh x$ a $\cosh x$ sú konvergentné. Taktiež v tomto rozsahu by bol algoritmus veľmi efektívny a presný. Pretože by sa výsledná funkčná hodnota počítala veľmi malým počtom iterácii.

Druhým spôsobom je vyčísliť funkčnú hodnotu z vzťahu $\tanh x = \frac{e^{2x}-1}{e^{2x}+1}$. A výraz e^{2x} počítať pomocou nekonečného radu. Tento rad je taktiež veľmi efektívny a presný pre malé hodnoty. Avšak pri počítaní extrémne malých hodnôt, ako sú napríklad $e^{10^{-6}}, e^{10^{-10}}$ je nevhodný.

Výsledný algoritmus je postavený tak, že podľa aktuálnej hodnoty x sa vyberie, ktorým spôsobom sa vyčísli funkčná hodnota danej funkcie. Ak je x v intervale $(-1, 1)$,tak sa použije vzťah $\sinh x \,/\, \cosh x$. Ak nie, tak sa číslo x spracuje pomocou $\tanh x = \frac{e^{2x}-1}{e^{2x}+1}$. Takto zvolený algoritmus je veľmi efektívny a rýchly.

Ďalej popisujem výpočet podľa vzťahu (1, str. 380), ak je x v intervale $x \in (-1, 1)$:

$$y = \ \tanh x = \frac{\sinh x}{\cosh x}, \quad x \in \mathbb{R} \tag{4.0}$$

Hyperbolický sínus

Hyperbolický sínus budem počítať pomocou nekonečného radu (1, str. 704):

$$\sinh x = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} \dots , \ x \in \mathbb{R} \tag{4.1}$$

Vytvorím substitúciu x^2, pretože sa stále opakuje.

$$z = x * x \tag{4.2}$$

Aby som nemusel stále počítať nový faktoriál, tak použijem predošlý člen. Ďalší člen postupnosti sa bude počítať:

$$Y_{i+1} = Y_i * \frac{z}{i*(i+1)} \tag{4.3}$$

Algoritmus bude mať následnú podobu:

$$i = 0, \qquad Y_i = x, \qquad suma = Y_i$$

V cykle budeme inkrementovať

$$i = i + 2$$

$$Y_i = Y_{i-1} * \frac{z}{i * (i+1)}$$

$$suma = \ suma + Y_i$$

Hyperbolický kosínus

Hyperbolický kosínus sa výpočtom odlišuje od hyperbolického sínusu iba v malej miere. Budem ho počítať pomocou nasledovného nekonečného radu (1, str. 704):

$$\cosh x = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} \dots , \quad x \in \mathbb{R} \tag{4.4}$$

Vytvorím substitúciu x^2, pretože sa stále opakuje.

$$z = x * x \tag{4.5}$$

Aby som nemusel stále počítať nový faktoriál, tak použije predošlý člen. Ďalší člen postupnosti sa bude počítať:

$$Y_{i+1} = Y_i * \frac{z}{i*(i-1)} \tag{4.6}$$

Algoritmus bude mať následnú podobu:

$$i = 0, \qquad Y_i = 1, \qquad suma = Y_i$$

V cykle budeme inkrementovať

$$i = i + 2$$

$$Y_i = Y_{i-1} * \frac{z}{i * (i-1)}$$

$$suma = \ suma + Y_i$$

Nekonečný rad e^x

Funkčná hodnota $y = \tanh x$ sa bude počítať podľa vzťahu $\tanh x = \frac{e^{2x}-1}{e^{2x}+1}$ ak x nepatrí do intervalu $x \notin (-1, 1)$. Pritom aproximácia e^x sa bude aplikovať pomocou nekonečného radu. Aby sme výpočet ešte urýchlili tak číslo x rozdelíme na desatinnú a celú časť. Desatinnú časť vypočítame pomocou nekonečného radu a celú vyčíslime pomocou jednoduchého cyklu a konštanty e.

Nekonečný rad je daný vzorcom (1, str. 703):

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots , \ |x| < +\infty \tag{4.7}$$

Vytvorím rekurentný vzťah:

$$Y_i = Y_{i-1} * \frac{x}{i} \tag{4.8}$$

Algoritmus bude mať následnú podobu:

$$i = 1, \quad Y_i = 1, \quad suma = Y_i$$

V cykle budeme inkrementovať $i = i + 1$

$$Y_i = Y_{i-1} * \frac{x}{i}$$

$$sum = suma + Y_i$$

3.3 Vážený aritmetický priemer

Vzorec váženého aritmetického priemeru (1, str. 745):

$$y = \frac{x_1 * h_1 + x_2 * h_2 + x_{3*} h_3 + \cdots + x_n * h_n}{h_1 + h_2 + h_{3+\ldots} + h_n}, \ x \in \mathbb{R}, \ h \in \mathbb{N} \tag{4.9}$$

Vzhľadom nato, že sa majú výsledky priebežne vypisovať, je potrebné si uchovávať predošlý menovateľ zlomku, aby som mohol správne vypočítať novú hodnotu aritmetického priemeru. Hodnota x je aktuálne zadaná hodnota na vstupe a $vaha$ je váha prvku x. Pred začatím algoritmu je dôležité inicializovať premenné $priemer_{stary}$ a $vaha_{stara}$ na nulu.

$$priemer = \frac{priemer_{stary} * vaha_{stara} + x * vaha}{vaha_{stara} + vaha} \tag{5.0}$$

$$vaha_{stara} = vaha + vaha_{stara} \tag{5.1}$$

3.4 Vážený kvadratický priemer

Vzorec váženého kvadratického priemeru (1, str. 746):

$$y = \sqrt{\frac{x_1{}^2 * h_1 + x_2{}^2 * h_2 + x_3{}^2 * h_3 + \cdots + x_n{}^2 + h_n}{h_1 + h_2 + h_3 + \cdots + h_n}}, \quad x \in \mathbb{R}, h \in \mathbb{N} \tag{5.2}$$

Vzhľadom nato, že sa majú výsledky priebežne vypisovať, sa dostávam k tomu istému problému ako pri váženom aritmetickom priemere. Musím uchovávať predošlú hodnotu menovateľa zlomku, aby som mohol ďalej počítať. Taktiež je dôležité pred začatím algoritmu inicializovať premenné $priemer_{stary}$ a $vaha_{stara}$ na nulu.

$$priemer = \sqrt{\frac{priemer_{stary} * vaha_{stara} + x * vaha}{vaha_{stara} + vaha}} \tag{5.3}$$

$$vaha_{stara} = vaha + vaha_{stara} \tag{5.4}$$

3.5 Analýza vstupných hodnôt

Program ma spracovávať ľubovoľne dlhú postupnosť čísel. Preto musím ošetriť výnimočné stavy, kedy by sa na vstupe nenachádzalo číslo, ale znak. Na toto nám poslúži v jazyku C funkcia scanf. Jej pomocou dokážeme zistiť či sa podarilo korektne načítanie hodnôt. Prípadne ošetriť stav kedy by sa na vstupe nenachádzalo číslo.

3.6 Počet platných cifier

Počet Platných cifier je potrebné prepočítať na ε, pomocou ktorého budem ukončovať podmienku cyklu v tvare:

$$|Y_{i+1} - Y_i| \leq \varepsilon \tag{5.5}$$

Prepočet hodnoty sigdig na ε však záleží na konkrétnej implementácii problému. Pritom je nutné sa zamyslieť nad rozsahom sigdig. Zo zadania vyplýva, že to ma byť celé číslo. Rozsah som zvolil v intervale od $\langle 1;\ DBL\_DIG)$, pričom DBL_DIG je konštanta z hlavičkové súboru float.h, ktorá určuje koľko platných číslic ma dátový typ double.

4 Špecifikácia testov

Test 1: Zle zadanie parametrov na príkazovom riadku → Výpis chybového hlásenia a ukončenie programu.

-ha
--logax 3 2.14.4
--logax 2 1
--logax 2 -0,5
--tanh 3 aa
--tanh -9

Test 2: Krajné hodnoty a hodnoty mimo definičný obor funkcie logaritmus.

nan → nan
0 → -inf
inf → inf
-inf → nan
-9 →nan
reťazec → nan

Test 3: Krajné hodnoty a hodnoty mimo definičný obor funkcie hyperbolický tangens.

nan → nan
inf → 1
-inf → -1
reťazec → nan

Test 4: Správnosť výpočtu logaritmu s základom 5.644 na 10 platných číslic. → Správny výsledok.

0.00054 → -4.3476087507e+00
1.2165477 → 1.1326585134e-01
13 → 1.4821216237e+00
269.16645 → 3.2331864578e+00

Test 5: Správnosť výpočtu hyperbolického tangensu na 3 platné číslice. → Správny výsledok.

0 → 0
0.00054 → 5.3999994751e-04
-9 → -9.9999996954e-01
14 → 1.0000000000e+00

Test 6: Správnosť výpočtu váženého aritmetického priemeru. → Správna výsledná postupnosť výsledkov.

0 0 → nan
1 5 → 1.0000000000e+00
2 6 → 1.5454545455e+00
18 9 → 8.9500000000e+00

Test 7: Správnosť výpočtu váženého kvadratického priemeru. → Správna výsledná postupnosť výsledkov.

0 0 → nan
1 5 → 1.0000000000e+00
2 6 → 1.6236882818e+00
18 9 → 1.2134661100e+01

5 Popis riešenia

Implementácia algoritmov stojí na rekurentných vzorcoch z kapitoly 3 a uvedených optimalizáciách.

5.1 Ovládanie programu

Program funguje ako konzolová aplikácia, takže má čisto textové ovládanie. Riadi sa parametrami príkazového riadku. Parametrom –h sa spustí nápoveda. V ktorej je stručný popis programu a jeho ovládanie. Ostatné matematické funkcie sa spúšťajú prepínačmi v tvare:

--logax sigdig a spustí sa funkcia na výpočet logaritmu s základom a

--tanh sigdig spustí sa funkcia na výpočet hyperbolického tangensu

--wam pre vážený aritmetický priemer

--wqm pre vážený kvadratický priemer

sigdig znamená na koľko platných cifier má byť výpočet presný.

Ak sa program spustí s nesprávnymi parametrami je ukončený s chybovým hlásením.

5.2 Dátové typy

Na ukladanie hodnôt prevzatých z príkazového riadku som použil štruktúru TParams ktorá obsahuje informácie o stave programu a chybových hláseniach. Taktiež obsahuje parameter sigdig a parameter a , oba typu double. Pretože sigdig sa prepočíta na epsylon. Vo funkciách na počítanie priemerov som použil štruktúru TPriemer, ktorá uchováva predošlé výsledky, ktoré sú potrebné na vypočítanie aktuálnych.

5.3 Vlastná implementácia

Vo funkcii main sa ako prvá zavolá funkcia getParams, ktorá spracuje parametre príkazového riadku. Naplní štruktúru TParams. Podľa zadaných parametrov sa volajú funkcie, ktoré čítajú zo štandardného vstupu. Tam je vytvorený cyklus, ktorý s funkciou scanf číta prvky na vstupe, pokiaľ nenarazí na koniec súboru -EOF. Scanf analyzuje správny formát vstupu. Ak bol zadaný text, tak zmení výstup na NAN a posunie sa na ďalšiu hodnotu. Ďalej sa volajú príslušné matematické funkcie, ktoré zvolil užívateľ. Tie si hodnoty upravia do daných intervalov, ktoré som uviedol v analýze problému.

Hodnoty sa predávajú ukazateľom, aby sa zmenšila pamäťová náročnosť programu. Vypočítaná hodnota je okamžite posielaná na štandardní výstup funkciou printf.

6 Záver

Program sa dá úspešne použiť pre výpočet matematických funkcií $\sinh x$, $\cosh x$, $\tanh x$, e^x a $\log_a x$. Pri zadaní vyššej presnosti môžem dôjsť k veľmi presným výsledkom. Ako elegantné riešenie by som uviedol funkciu heurUprava, ktorá docieli, aby sa funkcia $\ln x$ počítala čo s najmenším počtom iterácii. Čím docielime najpresnejší výsledok a znížime výpočetnú náročnosť programu. Ďalej by som poukázal na funkciu tanhyperB, ktorá si vyberá akou metódou sa bude počítať funkčná hodnota hyperbolického tangensu. Celkovo som algoritmy koncipoval tak, aby boli čo najefektívnejšie, a výsledky boli čo najpresnejšie vzhľadom na veľmi malý počet iterácii.

U štatistických funkciách by som uviedol vzorové použitie štruktúry, ktorá uchováva dané predošlé výsledky, aby sme mohli vypočítať nový priemer bez použitia poľa.

Reálne použitie programu by som videl v integrácii jednotlivých modulov s funkciami do iného väčšieho projektu. Alebo by bolo možné program rozšíriť o iné matematické funkcie a následné použiť v skriptách, alebo v iných programoch.

Použitá literatúra

1. BARTSCH, Hans-Jochen. Matematické vzorce. Vydal: Praha Mladá fronta, tretie vydanie, 2000, 831 strán, ISBN 80-204-0607-7.

2. International Standard ISO/IEC 9899:1999, Programming languages – C, druhé vydanie 1.12.1991, 537 strán.

A. Metrika kódu

Počet súborov: jeden súbor.

Počet funkcií: 19 vrátane funkcie main.

Počet riadkov zdrojového textu: 635.

Veľkosť spustiteľného súboru: 22 614 B (systém Linux, 32 bitová architektúra).