Dokumentace k projektu pro předměty IZP a IUS

Iterační výpočty

projekt č. 2

6. prosince 2013

Autor: Jan Pawlus xpawlu00@stud.fit.vutbr.cz
Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

Obsah

1 Úvod

2. projekt do předmětu IZP se jmenuje iterační výpočty. Slovo iterace se dá zaměnit se slovy jako cyklus nebo opakování - principem projektu je tedy napsat zadané funkce tak, aby byly v cyklech správně použité matematické vzorce.

Úkolem je vytvoření programu na výpočet úhlů trojúhelníku. Další bod zadání je vytvoření vlastních funkcí odmocniny (pomocí Newtonovy metody) a arcus sinus (pomocí Taylorovy řady). Program je navržen tak, aby spolu funkce souvisely - na výpočet úhlů trojúhelníku se dá (má) využít jak odmocnina, tak arcus sinus. Funkce z knihovny `math.h` jsou zakázané - student si je tedy musí vytvořit sám. V zadání jsou přímo určené vzorce, které se mají použít. Jakékoliv výpočty jsou povolené pouze skrz aritmetické operace $+$, $-$, $\cdot$ a $/$.

Výpočet úhlů trojúhelníku se může zdát jako jednoduchý úkol, za určitých okolností ale mohou nastat problémy. Tato dokumentace popisuje návrh a implementaci programu, který je schopen úhly a další zadané věci spočítat. Také jsou zde okomentovány zmíněné problémy a jejich řešení.

2 Analýza a princip řešení

2.1 Arcus sinus

Aby měly goniometrické funkce smysl, musí být definovány také jejich inverzní fuknce - tzv. cyklometrické funkce arcus (arcus sinus, arcus cosinus atd). Arcus sinus je funkce lichá a rostoucí v celém definičním oboru. Výpočet této funkce lze provést pomocí Taylorovy řady. Výpočet Taylorovy řady proběhne v cyklu, podmínka provedení cyklu bude záležet na tzv. přesnosti výpočtu, viz níže. Pro náš projekt se počítá arcus sinus jen v definičním oboru $< -1, 1 >$

Podobné jako u Taylorovy řady to bude i u odmocniny - zde se v cyklu provádí Newtonova metoda. Přesnost zde bude využita stejně, jako u arcus sinus. V množině reálných čísel nelze odmocnit číslo záporné, vstup pro odmocninu musí být tedy větší nebo rovno 0.

2.2 Přesnost výpočtu

V zadání je napsáno, že přesnost výpočtů má být na 11 platných číslic, vytvořil jsem si tedy konstantu `PRESNOST` o hodnotě $1 \cdot 10^{-11}$. Tuto konstantu jsem použil jako podmínku pro provedení cyklů - pokud je přírůstek posledních dvou cyklů větší než `PRESNOST`, výpočet probíhá dále. Konstantu jsem také využil kvůli nepřesnosti datového typu `double`, viz 2.3.

2.3 Trojúhelníková nerovnost a nepřesnost double

Standardní trojúhelníková nerovnost říká, že součet jakýchkoliv dvou stran musí být větší než strana třetí. Teoreticky lze takto odhalit, zda uživatel zadal pouze přímku, prakticky to ale neplatí vždy. Problémem je nepřesnost `double` - tento datový typ je totiž přesný "pouze"na 15 - 16 desetinných míst, nelze tedy spoléhat pouze na trojúhelníkovou nerovnost. Pokud by byl rozdíl až například po 16. desetinném místě, je podmínka nerovnosti irelevantní. Proto ji lehce upravím - součet jakýchkoliv dvou stran musí být větší než strana třetí + konstanta `PRESNOST`. Pokud by byl tedy rozdíl až na vzdálenějším desetinném místě, přičtení této konstanty ke třetí straně by tento problém eliminovalo, neboť by se rozdíl vytvořil již na 11. desetinném místě.

2.4 Problém krajních hodnot

Potenciální problém, který je zmíněn i v zadání projektu, vzniká při výpočtu arcus sinusu pro krajní hodnoty blížícím se 1 a -1 včetně. *Konvergence Taylorovy řady pro arcsin je u mezních hodnot ($|x| \approx 1$) velmi pomalá. To může mít za následek, že výsledek i přes zadanou přesnost nebude odpovídat správné hodnotě.*

Logicky by tedy nebylo na škodu vyhnout se výpočtu v krajních hodnotách. Jedno z řešení by mohlo být rozdělení této krajní hodnoty, tedy najít podobný vzorec, jako např. $\sin(x + y) = \sin x \cdot \cos y + \cos x \cdot \sin y$.

3 Návrh řešení

3.1 Taylorova řada a Newtonova metoda

Pro výpočet odmocniny slouží Newtonova metoda, tedy

$$\sqrt{x} = y_n, \qquad y_0 = 1, \qquad y_{i+1} = \frac{1}{2} \cdot (\frac{x}{y_i} + y_i)$$

Taylorova řada pro arcus sinus vypadá takto:

$$\arcsin x = x + \frac{1}{2} \cdot \frac{x^3}{3} + \frac{1 \cdot 3}{2 \cdot 4} \cdot \frac{x^5}{5} + \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6} \cdot \frac{x^7}{7} + ...$$

Pokud vytvořím funkci `mocnina`, u které mi stačí násobení, splním tím podmínku pro použití pouze matematických operací uvedených výše.

3.2 Výpočet úhlů trojúhelníku

Jako vstup zadá uživatel 6 bodů trojúhelníku - $a_x, a_y, b_x, b_y, c_x, c_y$. K výpočtu úhlů je potřeba znát délky stran. Například stranu AB lze zjistit ze vzorečku:

$$\sqrt{(b_x - a_x)^2 + (b_y - a_y)^2}$$

.

Poté, co jsou vypočteny všechny strany, je nutné ověřit, zda je zadaný objekt vůbec trojúhelníkem. Nejlepším řešením je pravděpodobně ověření trojúhelníkové nerovnosti - součet jakýchkoliv dvou stran musí být větší, než strana třetí + `PRESNOST` (viz 2.3). Následně je nutno využít Kosinovu větu, která např. pro úhel α vypadá takto:

$$a^2 = b^2 + c^2 - 2bc\cos\alpha$$

Z tohoto vzorce se vyjádří $\cos\alpha$ - to poté dosadím zde a dostanu výsledné úhly.

$$\arccos\alpha = \frac{\pi}{2} - \arcsin(\cos\alpha)$$

3.3 Optimalizace Taylorovy metody

Jak jsem zmínil výše, problém při výpočtu arcus sinus nastává, pokud uživatel zadá krajní hodnotu blížící se 1 nebo -1. Proto jsem se rozhodl najít řešení, jak se krajním hodnotám vyhnout.

$$\arcsin(p+q) = \pi - P - Q, \quad kde \quad P = \sqrt{\frac{1}{2} \cdot (1 + p^2 - q^2 + \sqrt{(-4p^2) + (-1 - p^2 + q^2)^2})}$$

Vzorec pro Q psát nemusím, jelikož si krajní hodnotu rozdělím napůl, dostanu tedy:

$$\arcsin(p+p) = \pi - P - P, \quad kde \quad P = \sqrt{\frac{1}{2} \cdot (1 + \sqrt{(-4p^2) + 1})}$$

3.4 Vstupní hodnoty

Předem je nutné zmínit, jak vůbec zpracovávám vstupní údaje - ze zadání vím, že argument pro výpočet odmocniny je `--sqrt`, pro arcus sinus `--asin` a pro úhly trojúhelníku `--triangle`. Rozeznání těchto argumentů lze provést jednoduše pomocí funkce `strcmp`, která porovnává řetězce znaků. Následující argument - hodnoty, které chce uživatel vypočítat - má být číslo, v argumentu je to ale ve formě `string`. Proto použiju funkci `strtod`, která má dva parametry - zdroj, který chci převést, a ukazatel na pole, do kterého se uloží případná chyba při převodu. Lze tedy jednoduše ověřit, pokud se převod nezdařil - funkcí `strlen` změřím délku chybového pole, pokud je délka nenulová, převod neproběhl správně. (např. argument `--sqrt 2p`). U odmocniny tedy ověřím, zda je v argumentu opravdu číslo a také, zda je toto číslo větší nebo rovno 0.

Při argumentu `--asin` musím opět ověřit, zda jsou zadané opravdu čísla, a také, zda je číslo z intervalu $< -1, 1 >$. U trojúhelníku je třeba ověřit zase správnost čísla, a poté, zda je objekt vůbec trojúhelník pomocí trojúhelníkové nerovnosti. Pro všechny funkce musí být také zadaný správný počet argumentů (např. pro trojúhelník musí být 8 argumentů - cesta k souboru (automaticky), `--triangle` a 6 bodů.)

3.5 Výstup programu

Výstup pro odmocninu a arcus sinus je 1 číslo ve tvaru `%.10e` - je tedy v exponenciálním tvaru a za desetinnou čárkou (tečkou) následuje 10 desetinných míst. Pokud program počítá úhly trojúhelníku, výstup bude stejný, jen se vypíšou 3 čísla (úhly v radiánech).

Pokud uživatel zadal argumenty syntakticky špatně nebo neodpovídá počet argumentů, program vypíše hlášku o špatně zadaných argumentech a odkáže uživatele na funkci `--help`. Proběhne-li výpočet špatně (uživatel zadá hodnoty mimo definiční obory funkcí apod.), vypíše se `nan`.

3.6 Použité funkce

Výpočetní funkce jsou typu `double`. Pro výpočet arcus sinusu se má funkce jmenovat `my_asin(double x)`, pro odmocninu `my_sqrt(double x)` a pro úhly trojúhelníku není definován ze zadání název - můj název je `triangle(AX AY BX BY CX CY)`. Dále jsem si vytvořil pomocné funkce pro absolutní hodnotu `abshod(double x)`, pro výpočet mocniny `mocnina(double x)` a o výpis nápovědy se stará funkce typu `void` nazvaná `printhelp()`.

4 Specifikace testů

Z analýzy vstupních dat vyplývá, že uživatel má možnost zadat hodně špatných argumentů - od špatného čísla přes nesplnění definičních oborů funkcí až po objekt, co není trojúhelník.

Test 1: Chybně zadané argumenty pro výběr funkce ⟶ Detekce chyby
`--sqprt 1`
`--arcsin 0.667`
`--triangle0 0 5 7 6 9`

Test 2: Chybně zadaná čísla nebo počet argumentů ⟶ Detekce chyby
`--sqrt p2`
`--sqrt 2 5`
`--asin 0.66q7`
`--triangle 0 0 5p 7 6 9`
`--triangle 0 0 5 7 6`

Test 3: Čísla jsou zadána mimo povolený rozsah ⟶ Detekce chyby
`--sqrt -1`
`--asin 1.11111`

Test 4: Správný výpočet ⟶ Výpis výsledku
`--sqrt 1` → `1.0000000000e+00`
`--asin .5` → `5.2359877560e-01`
`--triangle 0 0 1 0 0 2` → `1.5707963268e+00, 1.1071487178e+00, 4.6364760900e-01`
`--help` → Vypíše nápovědu
`--sqrt nan` → `nan`
`--sqrt -inf` → `-inf`

5 Popis řešení

5.1 Ovládání programu

Tento projekt je konzolová aplikace, ovládá se tedy pouze textově. Za běhu programu toho uživatel již moc nezmění - ovlivnit může jen vstupní argumenty, kde si vybírá funkce, kterou chce provést, a zadává hodnoty. `--sqrt X` reprezentuje výpočet odmocniny, `--asin X` výpočet arcus sinus, kde X je číslo, a `--triangle AX AY BX BY CX CY` výpočet úhlů trojúhelníku zadaného 6 body. Argument `--help` vypíše nápovědu (více o vstupních hodnotách v kapitole 3.4).

5.2 Použité datové typy

Zadání říká, že má student v tomto projektu pro výpočty používat datový typ `double`. Tento datový typ zabírá na 32bitových a 64bitových systémech 8 bytů (64 bitů) a jeho přesnost řeším v kapitole 2.3. Funkce `main()` je samozřejmě datového typu `int` a `printhelp()` je typu `void`, jelikož nemá žádnou návratovou hodnotu.

5.3 Vlastní implementace

Detekce argumentů se zpracovává přímo ve funkci `main` - provádí se několik podmínek `if`, které porovnávají funkcí `strcmp` argumenty na pozici 1 (volba výpočetní funkce). Až program zjistí, že uživatel zadal správný argument výpočetní funkce (např. `--sqrt`), následují podmínky, které řeší definiční obory jednotlivých funkcí a které také ošetřují vstupní hodnoty typu `nan`, `-inf` a další. Pokud všechny podmínky jednotlivé funkce platí, program funkci zavolá a proběhne výpočet, resp. výpis nápovědy.

Pokud program zavolá `my_sqrt`, provedou se uvnitř pomocné funkce `mocnina` a `abshod`. Uvnitř `my_asin` se provede `my_sqrt`, díky tomu i obě pomocné funkce a `triangle` kombinuje všechny funkce dohromady.

6 Závěr

Pokud jsou splněny vstupní podmínky (správně zadané argumenty, definiční obory vstupních hodnot), program správně spočítá odmocninu pomocí Newtonovy metody, arcus sinus pomocí Taylorovy řady a úhly trojúhelníku.

Jako zajímavé body mého projektu bych uvedl optimalizaci výpočtu arcus sinus pro krajní hodnoty nebo problém s přesností datového typu `double` u ověřování trojúhelníkové nerovnosti. Oba tyto problémy byly zanalyzovány a úspěšně eliminovány.

Program byl testován pro vstupní hodnoty navržené v zadání, ale také pro další vhodně zvolené. Funkčnost je stejná na operačních systémech Windows i Linux. Výstupní hodnoty odpovídají požadavkům a fungování je správné na 64bitových a 32bitových systémech. U 16bitových systémů by nastal problém s rozsahem datového typu `double`, ten by zde měl totiž menší přesnost. Řešení by byl výpis méně než 10 desetinných míst a zmenšení konstanty `PRESNOST`.

Literatura

[1] SLOVÁK, Jan, Martin PANÁK a Michal BULANT. *Matematika drsně a svižně*. 1. vyd. Brno: Masarykova univerzita, 2013. 773 s. ISBN 978-80-210-6307-5. doi:10.5817/CZ.MUNI.O210-6308-2013.

7 Metriky kódu

Počet řádků: 228
Počet funkcí: 7
Velikost kódu programu: 5 429 B
Velikost statických dat: 720 B