

4. Übung des Programmierpraktikums

Abgabetermin: 6. April 2009, 23:59 Uhr

Die Übungen sind grundsätzlich allein zu machen. Gruppenarbeit ist nicht erlaubt. Abzugeben sind jeweils die sinnvoll dokumentierten Programmfiles (Files für Bsp. 5: `bsp_5.cpp`, `bsp_5_fkt.cpp`, `bsp_5_fkt.hpp`) indem Sie diese in Ihr Verzeichnis im globalen Abgabeordner kopieren.

10. Berechnen Sie die Taylorreihe für $\ln x$, $x \in]0, 2]$ bis zum n -ten Glied

$$\begin{aligned} \ln x &\approx \sum_{k=1}^n (-1)^{k+1} \frac{(x-1)^k}{k} & (1) \\ &= \frac{x-1}{1} - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \dots + (-1)^{n+1} \frac{(x-1)^n}{n} \end{aligned}$$

für die einzugebenden Größen x und n . Geben Sie den berechneten Wert aus (1 Pkt.) und vergleichen Sie diesen mit dem exakten Wert.

Add: Versuchen Sie, ohne die Funktion `pow` auszukommen.

Testdaten (x, n) : (1.9, 16), (0.01,1000), (0.01,10)

11. Berechnen Sie die Taylorreihenapproximation aus Gleichung (1) bis Sie eine relative Genauigkeit von ε bzgl. des exakten Wertes erreicht haben.

Geben Sie den approximierten Wert, die Differenz zum exakten Wert und den Abbruchindex n (letztes n , für welches noch ein Term addiert wurde) aus.

Die Befehle `break`, `continue`, `goto` sind hierbei (und generell im Rahmen der LV) **nicht erlaubt**.

Hinweis: Eine Formel für die relative Genauigkeit ist $\frac{|v_{\text{exakt}} - v_{\text{approx}}|}{|v_{\text{exakt}}|}$ falls (1 Pkt.)

$v_{\text{exakt}} \neq 0$.

Testdaten (x, ε) : (1.9, 1e-5), (0.01, $3.2 \cdot 10^{-12}$)

12. Schreiben Sie ein Programm, welches solange einen C++-`string`¹ einliest bis der eingegebene String mit dem selbstgewählten Passwortstring übereinstimmt.

Verdreifachen Sie bei jedem Fehlversuche eine Strafzahl (welche z.B., die Zeit regeln könnte, bis ein neuer Eingabeversuch erfolgen darf). (1 Pkt.)

Lassen Sie max. 5 Eingabeversuche zu.

Hinweis: Beginnen Sie mit der ersten Teilaufgabe und erweitern Sie Ihre vorhandene Lösung um die nächste(n) Teilaufgabe(n).

Die Befehle `break`, `continue`, `goto` sind hierbei (und generell im Rahmen der LV) **nicht erlaubt**.

selbstgewählte Eingabedaten.

¹<http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/Script/html/node23.html>

13. **doppelter Zählzyklus:** Erzeugen Sie eine Tabelle aus den natürlichen Zahlen $k \in \mathcal{I}$, deren Quadraten, dritten Potenzen bis zur p -ten Potenz ($p \geq 3$) und geben Sie diese Tabelle wie folgt aus: (2 Pkt.)

Zahl	Quadrat	Kubik	...	Zahl ^p
4	16	64	...	berechne 4 ^p
⋮	⋮	⋮	...	⋮
k	k^2	k^3	...	k^p
⋮	⋮	⋮	...	⋮
n	n^2	n^3	...	n^p
$\sum_{k \in \mathcal{I}} k$	$\sum_{k \in \mathcal{I}} k^2$	$\sum_{k \in \mathcal{I}} k^3$		

Dabei sollen nur die Summen der Zahlen, Quadrate und Kubikzahlen berechnet werden.

Die Zahl $k \in \mathcal{I}$ soll alle durch vier teilbaren natürlichen Zahlen mit $k \leq n$ durchlaufen.

Eingabedaten (n, p) : (33, 6), (431, 3)

Hinweis 1: Testen Sie zuerst mit allen natürlichen Zahlen $k = 1, \dots, 10$.

Hinweis 2: Zur tabellarischen Ausgabe, insbesondere rechtsbündige Ausgabe mit Platzhalter siehe §10 des Skriptes², `#include <iomanip>` nicht vergessen.

Hinweis 3: Was passiert bei $n = 432$ und $p = 3$?

generelle Hinweise:

Schauen Sie sich die folgenden C++-Funktionalitäten an und nutzen Sie diese gegebenenfalls: `for`, `while`, `do`, `fabs`³, `string`, `cout.width`, `setw`

Die Befehle `break`, `continue`, `goto` sind **nicht erlaubt**.

Abgabe der Lösungen:

Die Abgabe der Lösungen (*.cpp-Files, *.hpp, ..., **keine** exe-Files) erfolgt ausschließlich durch das Kopieren dieser Files in Ihr (beim erstmal anzulegendes) Verzeichnis `Nachname_Vorname` im globalen Abgabeordner `y: \\pers.ad.uni-graz.at\fs\ou\621\stud_haase\Nachname_Vorname`.

Hinweise hierzu sind unter der LV-Homepage⁴ zu finden.

Die Filenamen **müssen** dem Schema `bsp_nummer`, gefolgt von der Fileextension, entsprechen. Z.B. ist in Beispiel 1 das File `bsp_1.cpp` abzugeben. Andere Filebezeichner zählen als nicht abgegebene Files.

²<http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/Script/html/node93.html>

³<http://www.cplusplus.com/reference/clibrary/cmath/fabs.html>

⁴<http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/SS08/index.html>