

2. Übung des Programmierpraktikums

Abgabetermin: 17. April 2023, 23:59 Uhr

Die Übungen sind grundsätzlich **selbst zu lösen**. Zweiergruppen sind erlaubt, aber nur unter Angabe des Partners im gut sichtbaren Kommentar.

Abzugeben sind jeweils die sinnvoll dokumentierten Programmfiles (*.cpp, *.h) in einem separaten Ordner mit Namen `bsp_nummer`, d.h., der Ordner `bsp_2` gehört zu Aufgabe 2.

7. Berechnen Sie die Summe aller Quadrate von Kehrwerten der natürlichen Zahlen zwischen n_1 und n_2 ($n_2 \geq n_1$) für selbstgewählte n_1 und n_2 , also $\sum_{k=n_1}^{n_2} \frac{1}{k^2}$ über die folgenden (2 Pkt.)

4 Konstrukte:

- einer `for`-Schleife,
- einer `while`-Schleife,
- einer `do while`-Schleife,
- einer `for`-Schleife in umgekehrter Summationsreihenfolge, d.h. $\sum_{k=n_2}^{n_1} \frac{1}{k^2}$.

Lesen Sie dazu §4.4-4.6 des Skriptes¹.

Weshalb sind die Resultate von a) und d) für `float`-Daten und $n_2 \approx 10^6$ unterschiedlich? Welches Resultat ist genauer?

8. Fünf Personen haben versucht, die Summe der Zahlen 1 bis 100 zu berechnen [Bre17²,p.83]. Beurteilen Sie die folgenden Lösungsvorschläge (ohne diese zu programmieren!) bzgl. Korrektheit und weiterer möglicher Fallen. Begründen Sie Ihre Einschätzung! (1 Pkt.)

(a) _____

```
int n = 1, sum = 0;
while(n <= 100) {
    ++n;
    sum += n;
}
```

(b) _____

```
int n = 1, sum = 1;
while(n < 100) {
    n += 1;
    sum += n;
}
```

(c) _____

```
int n = 100;
int sum = n*(n+1)/2;
```

(d) _____

```
int n = 1;
while(n < 100) {
    int sum = 0;
    n = n + 1;
    sum = sum + n;
}
```

(e) _____

```
int n = 1, sum = 0;
while(n <= 0100) {
    sum += n;
    ++n;
}
```

Geben Sie ein File `main.cpp` (im Ordner `bsp_8`) ab, in welches Sie Ihre Einschätzungen per C++-Kommentar hineinschreiben.

¹http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Script/html/script_programmieren.pdf

²<http://www.cppbuch.de/>

11. Summation of specified numbers:

Write a function with input parameter n that adds all those positive integers less or equal n which are a multiples of 3 or of 5 (including or!). (2 Pkt.)

- The easiest approach uses a for-loop.
- Test your function in the main function with various parameters:
 - $n = 15$ results in 60.
 - $n = 1001$ results in 234 168.
 - $n = 1432987$ results in 479 139 074 204.
- Derive a formula for calculating the required sum without executing a loop. Implement it in a second function and test it.
- Compare the run time of your two functions by using the `chrono`⁷ functions for time measurement.
Run each function at least 1000 times to get some measurable timings.

Hints: `std::chrono::system_clock::now()`, `std::chrono::duration<double>`,
`std::chrono::duration_cast<...>(...`)

⁷https://en.cppreference.com/w/cpp/chrono/system_clock/now

12. Schreiben Sie eine **Funktion** welche die quadratische Gleichung

$$a \cdot x^2 + b \cdot x + c = 0$$

für **beliebige** $a, b, c \in \mathbb{R}$ mit $|a| + |b| \neq 0$ löst und geben Sie die entsprechende(n) (2 Pkt.) Lösung(en) in \mathbb{R} (d.h., wir betrachten keine komplexen Lösungen) an das aufrufende Programm zurück.

- Überlegen Sie zuerst, welche Fälle und damit welche Lösungsmengen es gibt.
 - Zeichnen Sie ein Struktogramm (mit abgeben oder zur Übung mitbringen).
 - Überlegen Sie welche Input- und Output-Parameter ihre Funktion haben muß.
Achtung: Anzahl der Lösungen ist flexibel.
 $|a| + |b| \neq 0 \implies$ Output-Parameter an die aufrufende Funktion nötig.
 - Jetzt (und nicht eher!) dürfen Sie die IDE starten.
 - Testen Sie den Code mit **allen** gegebenen **Testdaten**, geben Sie die Lösungen im Hauptprogramm (oder besser in einer weiteren Testfunktion) aus und überlegen Sie sich weitere Testdaten, sodaß alle Verzweigungen Ihres Codes getestet werden.
 - **Dokumentieren** Sie Ihre Funktion entsprechend des Vorlesungsbeispiels (Source⁸ (zweites Bsp.⁹), resultierendes html¹⁰) und der doxygen-Dokumentation¹¹.
Erzeugen Sie die html-Dokumentation mit doxygen¹² (Einstellung beim Expert-Build: EXTRACT_ALL) aus der IDE heraus.
- (*) Überprüfen Sie, ob Sie für $(1, -3 \cdot \text{fLarge}, 3 \cdot \text{fLarge})$ mit $\text{fLarge} = 2e+16$ die Lösungen $\{3, 2e+16\}$ erhalten.
Wenn nicht, dann den Satz von Vieta¹³ benutzen, um die Stellenauslöschung bei der Subtraktion großer Zahlen zu vermeiden [Gander2014, p.55].
Auch mit $(1, \text{fLarge}-3, -3 \cdot \text{fLarge})$ ausprobieren ($\implies \{-2e+16, 3\}$)

Eingabedaten (a,b,c):

(1.0, -4.0, 1.0), (2.0, -100, 1200), (0.0, 3.0, 4.5), (3.0, -6.0, 51.0), (0, 0, 4)

Literatur

[Gander2014] Gander, Walter and Gander, Martin and Kwok, Felix. Scientific Computing. Springer, TCSE 11, (2014).

Die Abgabe der Lösungen (*.cpp-Files, *.h, (Makefile*), ...) erfolgt an der KFU über Moodle, siehe dazu die Hinweise auf der LV-Homepage¹⁴.

Die Verzeichnisnamen mit den Files für die jeweilige Aufgabe **müssen** dem Schema `bsp_nummer` folgen, z.B. enthält das Verzeichnis (der Ordner) `bsp_1` alle Files für Beispiel 1. Andere Verzeichnisnamen zählen als nicht abgegeben.

Keine Leerzeichen, Sonderzeichen oder Umlaute in File- und Verzeichnisnamen benutzen (Portabilität).

⁸https://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS23/v_1b/main.cpp

⁹https://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS23/v_2a/main.cpp

¹⁰https://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS23/v_1b/html/main_8cpp.html

¹¹<https://www.doxygen.nl/manual/docblocks.html#cppblock>

¹²<http://www.doxygen.org/index.html>

¹³https://de.wikipedia.org/wiki/Satz_von_Vieta

¹⁴<http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Download/kfu.html>