2. Übung des Programmierpraktikums

Abgabetermin: 4. April 2022, 23:59 Uhr

Die Übungen sind grundsätzlich selbst zu lösen, Zweiergruppen sind erlaubt.
Abzugeben sind jeweils die sinnvoll dokumentierten Programmfiles (`*.cpp`, `*.h`) in einem separaten Ordner dessen Name `bsp_`*`nummer`* zu sein hat, d.h., der Ordner `bsp_2` gehört zu Aufgabe 2.

---

7. Berechnen Sie die Summe aller Quadrate von Kehrwerten der natürlichen Zahlen zwischen $n1$ und $n2$ ($n2 >= n1$) für selbstgewählte $n1$ und $n2$, also $\sum_{k=n_1}^{n_2} \frac{1}{k^2}$ über die folgenden 4 Konstrukte: (2 Pkt.)

   a) einer `for`-Schleife,
   b) einer `while`-Schleife,
   c) einer `do while`-Schleife,
   d) 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 Skriptes[1].

   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 [Bre17[2],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.

---

[1] `http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Script/html/script_programmieren.pdf`
[2] `http://www.cppbuch.de/`

9. Programmieren Sie eine Funktion (oder mehrere)

```
bool is_palindrom(const string &s);
```

welche an das aufrufende Hauptprogramm zurückgibt, ob der String `s` ein Palindrom[3] ist, d.h., vorwärts und rückwärts gelesen das gleiche ergibt. Testen Sie Ihre Funktion im Hauptprogramm. (1 Pkt.)
Hierbei sind (wie immer) keine `break`, `continue`, `goto` erlaubt.

- Wir nehmen an, daß nur Kleinbuchstaben verwendet werden (nicht darauf testen!).
- Finden Sie eine Lösung, die möglichst wenige Tests auf Gleichheit einzelner Buchstaben erfordert.
- Für Fortgeschrittene: Nutzen sie den STL-Algorithmus `equal`[4] für den Vergleich.

Eingabedaten (mystring): radar, abca, rentner, otto, abrakadabra

10. Wir schreiben Funktionen welche argtanh(x) (Umkehrfunktion von Tangens hyperbolicus), $|x| < 1$ mittels einer Reihe approximieren:

$$\text{argtanh}x \approx t_n := \texttt{reihe}(x,n) := x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \cdots + \frac{x^{2n-1}}{2n-1}$$

Entwerfen Sie **zuerst** ein Struktogramm der jeweiligen Funktion **bevor** Sie zu programmieren beginnen.

- Schreiben Sie eine Funktion, welche `reihe`(x,n) für ein fixes n berechnet, d.h., x, n sind Input der Funktion und der resultierende Wert stellt den Output dar. Die math. Voraussetzung an x ist in der Funktion zu überprüfen.
- Testen Sie Ihre Funktion vom Hauptprogramm heraus.
- Schreiben Sie **unter Nutzung obiger Funktion** eine zweite Funktion, welche für ein übergebenes x und ε, mit $n = 2$ beginnend, die Approximationen $t_n := \texttt{reihe}(x,n)$ berechnet und n solange erhöht, bis $|\texttt{reihe}(x,n-1) - \texttt{reihe}(x,n)| < \varepsilon$ erfüllt ist. Speichern Sie die berechneten t_n in einem dynamischen[5] Vektor `vector<double>`[6] (siehe auch § 2.3.4 des Skriptes). Die Funktion soll diesen dynamischen Vektor an das aufrufende Programm zurückliefern. (2 Pkt.)
- Beide Funktionen sind für die Verarbeitung mit doxygen zu **dokumentieren**!
- Geben Sie im Hauptprogramm x ein und rufen Sie die zweite Funktion mit $\varepsilon = 10^{-6}$ auf. Geben Sie anschließend den Abbruchindex n und die letzten beiden Werte des Vektors t im Hauptprogramm aus.

[3] `https://de.wikipedia.org/wiki/Palindrom`
[4] `http://www.cplusplus.com/reference/algorithm/equal/?kw=equal`
[5] `http://www.cplusplus.com/reference/stl/vector/push_back/`
[6] `http://www.cplusplus.com/reference/vector/vector/at/`

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<...>(...)`

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

12. Schreiben Sie eine oderer mehrere **Funktion**(en), welche alle reellen Lösungen $x \in \mathbb{R}$ der Ungleichung

$$ax^3 + bx + c \leq 0 \tag{1}$$

für **beliebige** $a, b, c \in \mathbb{R}$, für welche $\left(\frac{c}{2a}\right)^2 + \left(\frac{b}{3a}\right)^3 \geq 0$ gilt, bestimmt und geben Sie die entsprechende Lösungsmenge als `string` aus (Tip: Funktion `to_string()` nutzen). (2 Pkt.)

- Betrachten wir zuerst, wie man die Nullstellen einer kubische Gleichung der Form $ax^3 + bx + c = 0$ unter der Annahme $\left(\frac{c}{2a}\right)^2 + \left(\frac{b}{3a}\right)^3 \geq 0$ bestimmen kann (Visuelle Hilfe mit Matlab-Skript[8]):
  (a) Für $a \neq 0$ dividiert man die Gleichung durch a und erhält $x^3 + px + q = 0$.
  (b) Nun wird die Diskriminante $D = \left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3$ berechnet.
    – Wenn $D > 0$, dann hat die Gleichung eine einzige reelle Lösung, siehe [Bartsch2014, §3.3.1.3] oder [Zeidler2013, §2.1.6.2]

$$x = \sqrt[3]{-\frac{q}{2} + \sqrt{D}} + \sqrt[3]{-\frac{q}{2} - \sqrt{D}}.$$

    – Wenn $D = 0$ und $q \neq 0$, gibt es zusätzlich noch eine zweite (doppelte) Lösung

$$x' = -\frac{x}{2}.$$

    – Der Fall $D = 0$ und $q = 0$ ist trivial.
    – Warum gilt bei unserer Aufgabe stets $D \geq 0$?
  (c) Was passiert für $a = 0$?
- Angenommen, Sie können die Gleichung $ax^3 + bx + c = 0$, $\left(\frac{c}{2a}\right)^2 + \left(\frac{b}{3a}\right)^3 \geq 0$, lösen: Welche Fälle und damit welche Lösungsmengen gibt es für die Ungleichung (1)? Hinweis: Wenn Sie nicht wissen, wie Sie hier vorgehen sollen, dann zeichnen Sie ein paar Fälle graphisch. Die Lösungsmengen sind dann leicht ersichtlich.
- Zeichnen Sie ein Struktogramm (manuell; als pdf, jpg ö.ä mit abgeben).
- Überlegen Sie welche Input- und Output-Parameter ihre Funktion(en) haben muß.
- Jetzt (und nicht eher!) dürfen Sie die IDE starten.
- Testen Sie den Code mit **allen** gegebenen **Testdaten**, geben Sie die Lösungen im Hauptprogramm aus und überlegen Sie sich weitere Testdaten, sodaß alle Verzweigungen Ihres Codes getestet werden.
- **Dokumentieren** Sie Ihre Funktion entsprechend des Vorlesungsbeispiels (Source[9] (zweites Bsp.[10]), resultierendes html[11]) und der doxygen-Dokumentation[12].
  Erzeugen Sie die html-Dokumentation mit doxygen[13] (Einstellung beim Expert-Build: `EXTRACT_ALL`) aus der IDE heraus.

Hinweise: `if`, `cin`, `cout`, `to_string`, `string`, `vector`, `pow`, `sqrt`, `cbrt`[14], `auto`

Eingabedaten (a,b,c): $(2, -2, 12)$, $(-1, 1, -6)$, $(1, -12, 16)$, $(1, -12, -16)$, $(0, 2, 3)$, $(0, 0, 0)$

[8] `http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS22/bsp_12_cubic.m`
[9] `http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS22/v_1b/main.cpp`
[10] `http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS22/v_2a/main.cpp`
[11] `http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS22/v_1b/html/main_8cpp.html`
[12] `http://www.doxygen.nl/manual/docblocks.html`
[13] `http://www.doxygen.nl/index.html`
[14] `https://en.cppreference.com/w/cpp/numeric/math/cbrt`

Literatur

[Bartsch2014] Bartsch, Hans-Jochen, and Michael Sachs. Taschenbuch Mathematischer Formeln für Ingenieure und Naturwissenschaftler. Hanser Verlag, 23., überarb. Aufl.. ed. (2014).

[Zeidler2013] Zeidler, Eberhard. Springer-Taschenbuch der Mathematik : Begründet von I.N. Bronstein und K.A. Semendjaew. Weitergeführt von G. Grosche, V. Ziegler und D. Ziegler. Herausgegeben von E. Zeidler (2013); UB Graz: **E-Book**

Die Abgabe der Lösungen (`*.cpp`-Files, `*.h`, (`Makefile*`,) ...,) erfolgt an der KFU über Moodle, siehe dazu die Hinweise auf der LV-Homepage[15].

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).

[15] `http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Download/kfu.html`