technische universität dortmund

fakultät für informatik

Jurij Kuzmic

Dortmund, den 24. Oktober 2019

Übung zur Vorlesung EidP (WS 2019/20)
Blatt 2

Block gelb

Es können 4 Punkte erreicht werden.

Abgabedatum: 31. Oktober 2019, 23:59 Uhr

Hinweise

- Bitte beachten Sie die aktuellen Hinweise unter

  `https://ls11-www.cs.tu-dortmund.de/teaching/ep1920uebung/`

- Die Aufgaben sind in Dreiergruppen zu bearbeiten. Der Lösungsweg und die Programmierung ist gemeinsam zu erarbeiten. Die Einzelabgabe ist ab Blatt 2 einschließlich **nicht** erlaubt.
- Die Gruppenmitglieder sollten gemeinsam an der gleichen Übungsgruppe teilnehmen. Die Lösung wird jeweils komplett bewertet und den Gruppenmitgliedern gleichermaßen angerechnet. Jedes Gruppenmitglied muss in der Lage sein, alle abgegebenen Lösungen vorzustellen.
- Diese müssen spätestens bis zum jeweiligen Abgabetermin (siehe jeweiliger Aufgabenzettel) abgegeben werden. In darauffolgenden Übungen sind teilweise einzelne abgegebene Lösungen oder auch eine Musterlösung zu präsentieren und zu besprechen.
- Die Abgabe geschieht zwingend über das Web-Interface unter

  `https://ess.cs.tu-dortmund.de/ASSESS/`

  Eine Abgabe per Email wird nicht akzeptiert. Dieses Programm fragt nach Informationen über die Gruppenteilnehmer (**Vor-, Nachnamen, Matrikelnummern**) und sammelt im aktuellen Verzeichnis die abzugebenden Dateien ein. Namen und Anzahl von abzugebenden C++-Quellcodedateien variieren und stehen in der jeweiligen Aufgabenstellung. Wählen sie dazu **Abgabe** und als Veranstaltung **EidP Übungen WS 2019/20** aus und laden Sie Ihre Ergebnisdateien hoch. Bis zum Abgabetermin kann eine Aufgabe beliebig oft abgegeben werden. Bewertet wird immer die **letzte** von Ihnen hochgeladene Version.
- Die abgegebenen Antworten/Programme werden automatisch auf Ähnlichkeit mit anderen Abgaben überprüft. Werden hierbei starke Übereinstimmungen festgestellt, wird die Aufgabe als nicht abgegeben bewertet. **Plagiieren kann zum Nichtbestehen der Übung führen.**

- Sobald eine Abgabe von den Betreuern korrigiert wurde, können die erzielte Punktzahl und die korrigierte Lösung ebenfalls unter dieser Adresse eingesehen werden.
- Es ist ratsam, die Programme vor der Abgabe zu kompilieren und auszuführen.
- Verwenden Sie für die Textaufgaben **reine Textdateien.** Das Abgabesystem erkennt **keine** Word- oder PDF-Dateien!

Aufgaben

Aufgabe 1: Grundlagen (1 Punkt)

Legen Sie für Ihre Antworten eine Text-Datei `Aufgabe_02_1.txt` an.

a) Wie ist die Definition eines Datenverbunds (`struct`) festgelegt? (0.2 Punkte)

b) Welche zusammengesetzten Datentypen haben Sie bisher kennengelernt? (0.2 Punkte)

c) Was ist der Unterschied zwischen einer `continue`- und einer `break`-Anweisung? (0.2 Punkte)

d) Es sei das Array `a` vom Typ `int` mit der Größe 101 gegeben. Mit welchem Index spricht man nun das letzte Element von `a` an? (0.2 Punkte)

e) Welche Eigenschaften hat ein (statisches) Array? (0.2 Punkte)

Aufgabe 2: Schleifen (1 Punkt)

Die Leibniz-Reihe ist wie folgt definiert:

$$L_\infty = \sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{2i-1} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} \cdots = \frac{\pi}{4}$$

Diese wird zur Annäherung an die Kreiszahl π verwendet, indem das Ergebnis mit 4 multipliziert wird. Nach 4 Iterationen ($\frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7}$) ist das Ergebnis der Summe 0,72381 beziehungsweise die Näherung an $\pi \approx 2{,}89524$.
Somit kann nach 1000 Iterationen der Leibniz-Reihe folgendes Ergebnis errechnet werden:

$$L_{1000} \approx 0,785148 \approx \frac{\pi}{4}$$

Legen Sie eine Datei `Aufgabe_02_2.cpp` an und ergänzen Sie darin das folgende C++-Programmfragment so, dass bei Ausführung des Programms das Ergebnis nach n Iterationen ausgegeben wird.

```
1  /*** Aufgabe_02_2.cpp ***/
2  #include <iostream>
3  using namespace std;
4
5  int main() {
6    unsigned int const n = 1000;
7    //************************************
8    // Hier beliebig viele Zeilen ergaenzen
9    //************************************
10   return 0;
```

```
}
/* Ausgabe:
*/
/*** Ende Aufgabe_02_2.cpp ***/
```

Kompilieren Sie das Programm und führen Sie es anschließend aus. Kopieren Sie die Ergebnisse als Block-Kommentar an das Ende der Datei `Aufgabe_02_2.cpp`.

Hinweis: Das entstehende Programm soll auch bei „beliebiger“ Veränderung des Wertes von `n` die korrekten Ausgaben berechnen.

Aufgabe 3: Schleifen (2 Punkte)

Legen Sie für ihre Lösungen die Dateien `Aufgabe_02_3a.cpp` und `Aufgabe_02_3b.cpp` an. Gegeben sei das folgende Programm:

```
/*** Aufgabe_02_3.cpp ***/
#include <iostream>
using namespace std;

int main() {
  int i = 777;
  while (i >= 0) {
    if (i % 2 == 0) {
      cout << i << endl;
    }
    i -= 17;
  }
  return 0;
}
/* Ausgabe:
760
726
692
658
624
590
556
522
488
454
420
386
352
318
284
250
216
182
```

```
148
114
80
46
12
*/
/*** Ende Aufgabe_02_3.cpp ***/
```

a) Ändern Sie das Programm derart, dass statt einer `while`-Schleife eine `do-while`-Schleife verwendet wird. Die Funktionalität des Programms soll auch für beliebige andere Werte von `i` gleich bleiben. Schreiben Sie Ihr komplettes Programm in die Datei `Aufgabe_02_3a.cpp`. Kompilieren Sie das Programm und führen Sie es anschließend aus. Kopieren Sie die Ergebnisse als Block-Kommentar an das Ende der Datei `Aufgabe_02_3a.cpp`. (1 Punkt)

b) Ändern Sie das Programm aus Teilaufgabe a) derart, dass statt einer `while`-Schleife eine `for`-Schleife mit vollständigem Schleifenkopf verwendet wird. Die Funktionalität des Programms soll auch für beliebige andere Werte von `i` gleich bleiben. Schreiben Sie Ihr komplettes Programm in die Datei `Aufgabe_02_3b.cpp`. Kompilieren Sie das Programm und führen Sie es anschließend aus. Kopieren Sie die Ergebnisse als Block-Kommentar an das Ende der Datei `Aufgabe_02_3b.cpp`. (1 Punkt)

Präsenzaufgabe 4: Reihenfolge (0 Punkte)

Ergänzen Sie das folgende C++-Programmfragment so, dass bei Ausführung die Zahlen des Arrays `feld` in umgekehrter Reihenfolge ausgegeben werden (die Ausgabe sollte also die Form `Die Zahlen des Arrays in umgekehrter Reihenfolge: 85 82 90 39 4 8 71 63 47 23 44 73 10 30 1 17 88 56`
haben).

```
/*** Aufgabe_02_4.cpp ***/
#include <iostream>
using namespace std;

int main() {
  unsigned int const n = 18;
  int feld[n] = {56, 88, 17, 1, 30, 10, 73, 44, 23,
          47, 63, 71, 8, 4, 39, 90, 82, 85};

  // ************************************
  // Hier beliebig viele Zeilen ergaenzen
  // ************************************

  cout << "Die Zahlen des Arrays in umgekehrter Reihenfolge:\n";
  for (int i = 0; i < n; ++i){
    cout << feld[i] << " ";
  }
  cout << endl;
  return 0;
}
/* Ausgabe:
Die Zahlen des Arrays in umgekehrter Reihenfolge:
```

```
23  56 88 17 1 30 10 73 44 23 47 63 71 8 4 39 90 82 85
24  */
25  /*** Ende Aufgabe_02_4.cpp ***/
```

Hinweis: In dem Array `feld` sollen nach Ablauf des Programms die Werte in umgekehrter Reihenfolge gespeichert sein. Sie dürfen nur Zeilen an der vorgesehenen Stelle einfügen und **nicht** die `for`-Schleife verändern!