technische universität dortmund

fakultät für informatik

Jurij Kuzmic
Horst Schirmeier
Jonas Ellert

Dortmund, den 19. November 2020

Übung zur Vorlesung EidP (WS 2020/21)

Blatt 2

Block gelb

Es können 4 Punkte erreicht werden.

Abgabedatum: 26. November 2020, 23:59 Uhr

Hinweise

- Bitte beachten Sie aktuelle Hinweise unter:

  `https://ls11-www.cs.tu-dortmund.de/teaching/ep2021uebung/`

- Wir empfehlen zum Bearbeiten der Programmieraufgaben die Entwicklungsumgebung *Visual Studio Code*. Für die **Installation auf Ihrem Rechner** stellen wir auf der Übungs-Webseite eine detaillierte Installationsanleitung bereit.
- Die Aufgaben sind in Dreiergruppen zu bearbeiten, die Lösung wird jeweils komplett bewertet und den Gruppenmitgliedern gleichermaßen angerechnet. Der Lösungsweg und die Programmierung sind gemeinsam zu erarbeiten. Die Einzelabgabe ist ab Blatt 2 einschließlich **nicht** erlaubt.
- Der Abgabetermin für jedes Aufgabenblatt ist jeweils oben vermerkt. Die Abgabe geschieht über das ASSESS-System: `https://ess.cs.tu-dortmund.de/ASSESS/`
  Dieses benötigt Informationen über die Gruppenteilnehmer (**Vor-, Nachnamen, Matrikelnummern**) und fordert Sie auf, die zur Aufgabe gehörenden C++-Quelltextdateien und ggf. weitere Dateien (siehe Aufgabenstellung) hochzuladen. Wählen Sie dazu **Abgabe** und als Veranstaltung **Einführung in die Programmierung (EidP) Übung** aus, und laden Sie Ihre Dateien hoch. Bis zum Abgabetermin kann eine Aufgabe beliebig oft abgegeben werden, bewertet wird immer die **letzte** von Ihnen hochgeladene Version.
- Die abgegebenen Dateien 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 im ASSESS-System eingesehen werden.
- Sie sollten während der Entwicklung Ihrer Programme diese unbedingt regelmäßig – und insbesondere noch einmal vor der Abgabe – **kompilieren und ausführen**. Ein solches iteratives Vorgehen ist grundlegend für jede Programmiertätigkeit!

- Verwenden Sie für die Textaufgaben **reine Textdateien.** Das Abgabesystem erkennt **keine** Word- oder PDF-Dateien!
- Die Verwendung von zusätzlichen **Bibliotheken** zur Lösung der Aufgaben ist **nicht erlaubt**!

Aufgaben

Aufgabe 1: Grundlagen (1 Punkt)

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

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

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

c) 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)

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

e) Was ist der Unterschied zwischen einer `continue`- und einer `break`-Anweisung? (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:

$$4 \cdot L_{1000} \approx 4 \cdot 0.785148 = 3.140592 \approx \pi$$

Legen Sie eine Datei `Aufgabe_02_2.cpp` an und ergänzen Sie darin das folgende C++-Programmfragment, sodass bei Ausführung des Programms die Näherung von π 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 = 1337;
7    double pi = 0.0;
8    //************************************
9    // Hier beliebig viele Zeilen ergaenzen
10   //************************************
11   cout << "Naeherung von Pi nach " << n << " Iterationen: " << pi << endl;
12   return 0;
```

```
}
/* Ausgabe:
Naeherung von Pi nach 1337 Iterationen: 0
*/
/*** 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 = 666;
  while (i >= 1) {
    if (i % 2 == 1) {
      cout << i << endl;
    }
    i -= 17;
  }
  return 0;
}
/* Ausgabe:
649
615
581
547
513
479
445
411
377
343
309
275
241
207
173
139
105
71
37
```

```
35  3
36  */
37  /*** 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).

```
1   /*** Aufgabe_02_4.cpp ***/
2   #include <iostream>
3   using namespace std;
4
5   int main() {
6     unsigned int const n = 18;
7     int feld[n] = {56, 88, 17, 1, 30, 10, 73, 44, 23,
8            47, 63, 71, 8, 4, 39, 90, 82, 85};
9     // ***********************************
10    // Hier beliebig viele Zeilen ergaenzen
11    // ***********************************
12    cout << "Die Zahlen des Arrays in umgekehrter Reihenfolge:\n";
13    for (int i = 0; i < n; ++i){
14      cout << feld[i] << " ";
15    }
16    cout << endl;
17    return 0;
18  }
19  /* Ausgabe:
20  Die Zahlen des Arrays in umgekehrter Reihenfolge:
21  56 88 17 1 30 10 73 44 23 47 63 71 8 4 39 90 82 85
22  */
23  /*** 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!