technische universität dortmund

fakultät für informatik

Jurij Kuzmic

Dortmund, den 31. Oktober 2019

Übung zur Vorlesung EidP (WS 2019/20)

Blatt 3

Block gelb

Es können 4 Punkte erreicht werden.

Abgabedatum: 7. November 2019, 23:59 Uhr

Hinweise

- Bitte beachten Sie die aktuellen Hinweise unter

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

- Für die Abgabe sind die Quellcode-Dateien `Aufgabe_03_1.txt` und `Aufgabe_03_2.cpp` zu erstellen.
- Stellen Sie sicher, dass alle von Ihnen abgegebene Dateien reine Textdateien im UTF-8-Format sind.
- Für die Kompilierung des Programms muss der C++14-Standard aktiviert sein. Dies kann im Referenzkompiler GCC 6.3 durch den Schalter `-std=c++14` sichergestellt werden. Es sollen zudem die Parameter `-pedantic` und `-Werror` genutzt werden. Der Befehl zum Kompilieren soll somit wie folgt aussehen:

  ```
  g++-6 -pedantic -Werror -std=c++14 Aufgabe_03_02.cpp -o Aufgabe_03_02
  ```

- 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_03_1.txt` an.

a) Es sei die Deklaration `double *pVar;` gegeben. Welche Adresse referenziert dieser Zeiger nach der Deklaration? (0.1 Punkte)

b) Erklären Sie den Unterschied zwischen einer `for`-Schleife und einer `while`-Schleife. (0.2 Punkte)

c) Was ist ein *Zeiger*? Was ist der Unterschied zwischen den Zeigeroperatoren `&` und `*`?
(0.2 Punkte)

d) Erklären Sie anhand eines Beispiels, wie sowohl in einer `for`-Schleife als auch in einer `while`-Schleife Endlosschleifen entstehen können. (0.2 Punkte)

e) Gegeben ist folgendes Programm. Erklären Sie was das Programm ausgibt und beschreiben Sie insbesondere die Deklaration der Variablen. Kommentieren Sie jede Zeile des Codes und erklären Sie was ausgegeben wird. Ändern Sie das Programm so, dass jede Variable in einer eigenen Zeile deklariert wird.

```
1  /*** Aufgabe_03_1.cpp ***/
2  #include <iostream>
3  using namespace std;
4
5  int main() {
6    int* x, y, z[1]={};
7    y = 1;
8    x = &y;
9    z[0] = ++*x;
10   cout << z[*x-2] << endl;
11   return 0;
12 }
13 /*** Ende Aufgabe_03_1.cpp ***/
```

(0.3 Punkte)

Aufgabe 2: Mehrfachauswahl (3 Punkte)

Ergänzen Sie in der Datei `Aufgabe_03_2.cpp` das unten angegebene Programmfragment so, dass für ein eingegebenes Datum der darauffolgende Tag berechnet wird. Dabei soll zuvor geprüft werden, ob das Datum gültig ist. Wird ein Datum vor dem 1.1.1600 oder nach dem 31.12.2600 eingegeben, soll `99.99.9999` ausgegeben werden. Dasselbe soll ausgegeben werden, wenn ein ungültiges Datum eingegeben wird. Verwenden Sie zur Bestimmung der Anzahl an Tagen des eingegebenen Monats eine Mehrfachauswahl (`switch/case`). Achten Sie dabei insbesondere auf Schaltjahre. Alle Jahre, die durch 400 teilbar sind, sind Schaltjahre. Alle anderen Jahre sind genau dann Schaltjahre, wenn sie durch 4, aber nicht durch 100 teilbar sind.
Kompilieren Sie das Programm und führen Sie es anschließend aus. Kopieren Sie die Ergebnisse für die Eingaben `0.11.2016`, `11.11.1111`, `30.11.2016`, `28.02.2016` und `28.02.2017` als Block-Kommentar an das Ende der Datei `Aufgabe_03_2.cpp`.

```
1  /*** Aufgabe_03_2.cpp ***/
2  #include <iostream>
3  using namespace std;
4
5  int main() {
6    unsigned int tag, monat, jahr;
7    cout << "Geben Sie Tag, Monat und Jahr ein" << endl;
8    cin >> tag;
9    cin >> monat;
10   cin >> jahr;
11   //************************************
12   // Hier beliebig viele Zeilen ergaenzen
```

```
13    //************************************
14    cout << "Naechster Tag ist "
15       << tag << "." << monat << "." << jahr << endl;
16    return 0;
17  }
18  /* Ausgabe:
19  */
20  /*** Ende Aufgabe_03_2.cpp ***/
```

Präsenzaufgabe 3: Zeiger (0 Punkte)

Gegeben sei das folgende C++-Programm:

```
1  /*** Aufgabe_03_3.cpp ***/
2  #include <iostream>
3  using namespace std;
4
5  int main() {
6    short a = 11;
7    short b[] = { 87, 131, 81 };
8    short *x1 = nullptr;
9    short *x2 = nullptr;
10   short **y1 = nullptr;
11   short **y2 = nullptr;
12
13   x1 = &a;
14   x2 = x1;
15   x1 = &b[1];
16   x2 = x1-1;
17   y1 = &x2;
18   *y1 = x1;
19   y2 = y1;
20   y1 = &x1;
21   x1 = &a;
22   *x2 = a**(*y2+1);
23   **y1 = 1;
24   *(x2-1) = **y2+(*(*y2+**x1))+3;
25
26   cout << "a: " << a << endl;
27   cout << "b: [" << b[0] << ",";
28   cout << b[1] << "," << b[2] << "]" << endl;
29
30   return 0;
31 }
32 /* Ausgabe:
33 a: 1
34 b: [975,891,81]
35 */
36 /*** Ende Aufgabe_03_3.cpp ***/
```

In der untenstehenden Tabelle sind die Werte einzelner Variablen bzw. Ausdrücke (*nach* der Ausführung der jeweils angegebenen Zeile) aufgeführt. Vervollständigen Sie die Tabelle.

Zeile	a	b	x1	*x1	x2	*x2	y1	*y1	**y1	y2	*y2	**y2	
10	11	[87, 131, 81]	nullptr	–	nullptr	–	nullptr	–	–	nullptr	–	–	
13	11	[87, 131, 81]	&a	11	nullptr	–	nullptr	–	–	nullptr	–	–	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

Hinweis: Die Einträge sollen möglichst einfach sein, z. B. steht in der Spalte `*x1` des Beispiels die Zahl 11 und nicht der Ausdruck `*&a`. Diese Aufgabe dient der Vorbereitung für die Klausur.