technische universität dortmund

fakultät für informatik

Jurij Kuzmic
Horst Schirmeier
Jonas Ellert

Dortmund, den 3. Dezember 2020

Übung zur Vorlesung EidP (WS 2020/21)

Blatt 4

Block gelb

Es können 4 Punkte erreicht werden.

Abgabedatum: 10. Dezember 2020, 23:59 Uhr

Hinweise

- Bitte beachten Sie aktuelle Hinweise unter:

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

- Die Aufgaben sind **in Dreiergruppen** zu bearbeiten.
- Für die Abgabe sind die Dateien `Aufgabe_04_1.txt`, `Aufgabe_04_2a.cpp`, `Aufgabe_04_2b.cpp` und `Aufgabe_04_2c.cpp` zu erstellen.
- Die Verwendung von zusätzlichen **Bibliotheken** ist **nicht erlaubt**.
- Sie sollten während der Entwicklung Ihrer Programme diese unbedingt regelmäßig – und insbesondere noch einmal vor der Abgabe – **compilieren und ausführen**.

Aufgaben

Aufgabe 1: Grundlagen (1 Punkt)

Legen Sie für Ihre Antworten die Textdatei `Aufgabe_04_1.txt` an.

a) Geben Sie an, wie bei einer Funktion mit einem Parameter und Rückgabewert die Funktionsdeklaration und die Funktionsdefinition *allgemein* aussieht. (0.2 Punkte)

b) Worin liegt der Unterschied bei der Übergabe von Variablen als Wert, als Referenz und als Zeiger? (0.2 Punkte)

c) Welcher Zusammenhang existiert zwischen den *Argumenten eines Funktionsaufrufes* und den *Parametern in der Funktionsdeklaration* der aufgerufenen Funktion? (0.2 Punkte)

d) Worauf ist bei der Übergabe von Arrays an Funktionen zu achten? Wie kann die Länge des Arrays in der Funktion bestimmt werden? (0.2 Punkte)

e) Welche Bedeutung hat die Anweisung `return 89;` in einer Funktion mit Rückgabedatentyp `unsigned char`, und wie kann die Rückgabe weiterverwendet werden? (0.2 Punkte)

Aufgabe 2: Funktionen (3 Punkte)

a) Die n-te Dreieckszahl ist als Summe der positiven Ganzzahlen von 1 bis n definiert:

$$d_n = \sum_{i=1}^{n} i$$

Die Dreieckszahlenfolge sieht somit wie folgt aus: $1, 3, 6, 10, \ldots$
Schreiben Sie eine Funktion, die die n-te Dreieckszahl berechnet und zurückgibt. Diese Funktion soll in einer Schleife mit den Zahlen von 1 bis 25 aufgerufen werden, und die jeweils berechnete Dreieckszahl ausgegeben werden. Führen Sie Ihr Programm aus und fügen Sie die Ausgabe als Kommentar an das Ende der Datei `Aufgabe_04_2a.cpp` an.

```
1  /*** Aufgabe_04_2a.cpp ***/
2  #include <iostream>
3  using namespace std;
4
5  //***********************************************
6  // Ergaenzen: Definition der Funktion dreieckszahl
7  //***********************************************
8
9  int main() {
10   unsigned int const n = 25;
11
12   //*****************************************************************
13   // Ergaenzen: Schleife, die alle Dreieckszahlen von 1 bis n ausgibt
14   //*****************************************************************
15   return 0;
16 }
17 /* Ausgabe:
18 Ergaenzen!
19  */
```

(1 Punkt)

b) In dieser Aufgabe sind in einem Array `zahl` vom Typ `int` verschiedene Zahlen gespeichert. Die Zahlen in dem Array sollen mit Hilfe einer Funktion `halbiere` halbiert werden. Die Funktion und der Aufruf soll dabei so programmiert werden, dass sie auch bei beliebiger Veränderung des Arrays, insbesondere auch der Anzahl der Zahlen in dem Array, korrekt arbeitet. Hierbei soll nicht das ganze Array an die Funktion, sondern in einer Schleife die jeweils einzelnen Elemente des Arrays übergeben werden. Dabei können Sie entscheiden, welche der bereits kennengelernten Techniken der „Rückgabe“ (Referenz, Zeiger, Return) Sie verwenden. Fügen Sie die Ausgabe Ihres Programms als Kommentar an das Ende der Datei `Aufgabe_04_2b.cpp` an.

```
1  /*** Aufgabe_04_2b.cpp ***/
2  #include <iostream>
3  using namespace std;
4
5  //*****************************************
6  // Ergaenzen: Definition der Funktion halbiere
7  //*****************************************
```

```
8
9  int main() {
10   unsigned int const n = 6;
11   int zahl[n] = { 9, 34, -8, 42, 44221, 1 };
12   cout << "Zahlen vor der Halbierung" << endl;
13   for (unsigned int i = 0; i < n; ++i) {
14     cout << (i + 1) << ". Zahl: " << zahl[i] << endl;
15   }
16   //************************************************
17   // Ergaenzen: for-Schleife und Aufruf von halbiere
18   //************************************************
19   cout << "Zahlen nach Halbierung" << endl;
20   for (unsigned int i = 0; i < n; ++i) {
21     cout << (i + 1) << ". Zahl: " << zahl[i] << endl;
22   }
23   return 0;
24 }
25 /* Ausgabe:
26 Ergaenzen!
27 */
```

(1 Punkt)

c) In dieser Aufgabe soll *eine* Funktion `produkt_summe_und_differenz` geschrieben werden, die von zwei ganzen Zahlen vom Typ `int` das Produkt, die Summe und die Differenz berechnet. Nutzen Sie dabei *alle drei verschiedenen Arten*, das jeweilige Ergebnis der drei Berechnungen „zurückzugeben“. Verwenden Sie außerdem nur genau *einen* Funktionsaufruf.
Führen Sie Ihr Programm mit den vorgegebenen Variablen `zahl1` und `zahl2` aus. Das Programm soll auch mit einer beliebigen Veränderung der Variablenwerte korrekte Ergebnisse liefern. Fügen Sie die Ausgabe als Kommentar an das Ende der Datei `Aufgabe_04_2c.cpp` an.

```
1  /*** Aufgabe_04_2c.cpp ***/
2  #include <iostream>
3  using namespace std;
4  //************************************************************
5  // Ergaenzen: Definition der Funktion produkt_summe_und_differenz
6  //************************************************************
7  int main() {
8    int const zahl1 = 242, zahl2 = -9;
9    int produkt, summe, differenz;
10   //************************************************
11   // Ergaenzen: Aufruf von produkt_summe_und_differenz
12   //************************************************
13   cout << "Produkt: " << produkt << endl;
14   cout << "Summe: " << summe << endl;
15   cout << "Differenz: " << differenz << endl;
16   return 0;
17 }
18 /* Ausgabe:
19 Ergaenzen!
20 */
```

(1 Punkt)

Präsenzaufgabe 3: Parameter (0 Punkte)

Implementieren Sie die Funktionen `invers`, `grossbuch` und `umwandeln` wie folgt. Die Verwendung der `<cctype>`-Bibliothek ist nicht erlaubt.

- Die Funktion `invers` soll für Zeichenketten wie z. B. für `"Nashorn"`, die inverse Zeichenkette `"nrohsaN"` ermitteln. Der einzige Parameter der Funktion ist eine Zeichenkette vom Typ `char[]`. Die Funktion soll keinen Rückgabewert haben. Die Länge der Zeichenkette, die als Parameter übergeben wird, ist nicht bekannt.
- Die Funktion `grossbuch` erhält ein einzelnes Zeichen vom Typ `char` als Parameter, und liefert auch ein Zeichen vom Typ `char` als Ergebnis zurück. Falls das übergebene Zeichen ein Kleinbuchstabe ist, wird dieser in den entsprechenden Großbuchstaben umgewandelt und zurückgegeben. Ist das übergebene Zeichen bereits ein Großbuchstabe oder ein anderes Zeichen, so wird dieses einfach wieder zurückgegeben.
- Die Funktion `umwandeln` soll für komplette Zeichenketten wie z. B. `"naShorN"` die Zeichenkette `"NASHORN"` ermitteln, also sämtliche Kleinbuchstaben in Großbuchstaben umwandeln. Verwenden Sie dabei innerhalb der Funktion `umwandeln` die Funktion `grossbuch`. Diese Funktion soll keinen Rückgabewert haben.

Rufen Sie in der Funktion `main` die Funktionen so auf, dass das Array `word` zunächst den String `"tNaFelE"` enthält. Ergänzen Sie dann geeignete Funktionsaufrufe, damit das Array `word` den String `"ELEFANT"` enthält. In der `main`-Funktion sollen dafür keine Kontrollstrukturen verwendet werden.

```
1  /*** Aufgabe_04_3.cpp ***/
2  #include <iostream>
3  using namespace std;
4
5  //*********************************************************************
6  // Ergaenzen: Definition der Funktionen invers, grossbuch und umwandeln
7  //*********************************************************************
8
9  int main() {
10   char word[] = "EleFaNt";
11   //***************************************************
12   // Ergaenzen: geeignete Funktionsaufrufe und Ausgaben
13   //***************************************************
14   return 0;
15 }
16 /* Ausgabe:
17 */
```