

# Übung zur Vorlesung EidP (WS 2019/20)

## Blatt 7

Block rot

Es können 4 Punkte erreicht werden.

Abgabedatum: 12. Dezember 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 jeweils genannten Dateien 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.cpp -o Aufgabe
```
- Die Verwendung von zusätzlichen **Bibliotheken** zur Lösung der Aufgaben ist **nicht erlaubt!**

### Aufgaben

#### Aufgabe 1: Grundlagen (1.5 Punkte)

Legen Sie für Ihre Antworten eine Textdatei `Aufgabe_07_1.txt` an.

- a) Was sind Klassen und Objekte? Beschreiben Sie den Zusammenhang zwischen diesen. (0.3 Punkte)
- b) Welches ist der wesentliche Unterschied zwischen einer `struct`- und `class`-Definition? (0.1 Punkte)
- c) Welchen Zweck haben Konstruktoren und Destruktoren? (0.2 Punkte)
- d) Erklären Sie das Prinzip des „information hiding“. (0.2 Punkte)

- e) Erklären Sie den Begriff des Überladens von Methoden. (0.2 Punkte)
- f) Erklären Sie den Zweck der Teilung der Deklaration in `public` und `private`. (0.2 Punkte)
- g) Erklären Sie den Begriff der Delegation von Konstruktoren. (0.1 Punkte)
- h) Worauf muss man bei der Verwendung der Klassenschablonen bei Klassendefinition und Implementierung achten? (0.2 Punkte)

## Aufgabe 2: Klassen (2.5 Punkte)

a) Legen Sie für diese Teilaufgabe die Datei `Aufgabe_07_2a.h` an. Deklarieren Sie in dieser Datei die Klasse `Punkt`, die einen Punkt im mehrdimensionalen Raum mit Koordinaten vom Typ `double` repräsentieren soll. Halten Sie nur die Deklaration der Klasse in der Datei `Aufgabe_07_2a.h` fest. Die Implementierung der notwendigen Methoden finden in der nächsten Teilaufgabe statt. Die Klasse soll folgende Eigenschaften haben:

1. Die Dimension  $n$  muss bei Erzeugung eines Objekts angegeben werden und kann anschließend nicht mehr geändert werden.
2. Die Koordinaten sollen sinnvoll initialisiert werden und einzeln abruf- und veränderbar sein (Getter/Setter).
3. Folgende Operationen sollen ebenfalls unterstützt werden:
  - Addition mit einem Punkt, der die gleiche Anzahl an Koordinaten hat. Danach sind die neuen Koordinaten des Punktes das Ergebnis der Addition. Dies entspricht der Summe zweier Vektoren.
  - Multiplikation eines Punktes und einer Zahl vom Typ `double`, wobei die Zahl mit jeder Koordinate des Punktes multipliziert wird (*Skalarmultiplikation*).
  - Berechnung des *Skalarprodukts*  $\langle p, q \rangle$  zweier Punkte  $p, q$  mit je  $n$  Koordinaten durch
$$\langle p, q \rangle := \sum_{i=1}^n p_i \cdot q_i.$$

(0.7 Punkte)

b) Legen Sie für diese Teilaufgabe die Datei `Aufgabe_07_2b.cpp` an. Implementieren Sie in dieser Datei die Methoden aus Teilaufgabe a). (1.3 Punkte)

c) Legen Sie für diese Teilaufgabe die Datei `Aufgabe_07_2c.cpp` an. Inkludieren Sie die zuvor erstellte Klasse `Punkt`. Schreiben Sie ein Programm, bei dem zwei Punkte  $a = (2.5, 3.8, -2.2)$  und  $b = (0.8, -1.2, 3.1)$  erzeugt werden. Dann soll Punkt  $b$  zu Punkt  $a$  addiert werden und danach der Punkt  $b$  mit 3 multipliziert werden. Anschließend soll das Skalarprodukt der beiden Punkte  $a$  und  $b$  ausgegeben werden. Kompilieren Sie Ihr Programm und führen Sie es anschließend aus. Kopieren Sie die Ergebnisse als Block-Kommentar an das Ende der Datei `Aufgabe_07_2c.cpp`. (0.5 Punkte)

## Präsenzaufgabe 3: Konstruktoren und Destruktoren (0 Punkte)

Geben Sie die Ausgabe des folgenden Programms an.

```

1  /*** Aufgabe_07_3.h ***/
2  class baz {
3  private:
4      char x;
5  public:
6      baz();
7      baz(char y);
8      ~baz();
9  };
10
11 class foo {
12 private:
13     char x;
14 public:
15     foo();
16     foo(char y);
17     foo(char x, char y);
18     ~foo();
19 };
20
21 class bar : foo, baz {
22 private:
23     char x;
24 public:
25     bar();
26     bar(char y);
27     ~bar();
28     void barbaz(char c);
29 };
30 /*** Ende Aufgabe_07_3.h ***/

```

```

1  /*** Aufgabe_07_3.cpp ***/
2  #include <iostream>
3  #include "Aufgabe_07_3.h"
4
5  using namespace std;
6
7  baz::baz() : baz('B') {
8      cout << "Baz\t" << x << endl;
9  }
10
11 baz::baz(char y) : x('Z') {
12     char t = x;
13     x = y;
14     cout << "Baz\t" << x << endl;
15     x = t;
16 }
17
18 baz::~~baz() {
19     cout << "-Baz\t" << x << endl;
20 }

```

```

21
22 foo::foo() : foo('F') {
23     x++;
24     cout << "Foo\t" << x << endl;
25 }
26
27 foo::foo(char y) : x(y) {
28     cout << "Foo\t" << x << endl;
29 }
30
31 foo::foo(char x, char y) : x(x) {
32     cout << "Foo\t" << x << endl;
33     cout << "Foo\t" << y << endl;
34 }
35
36 foo::~foo() {
37     cout << "-Foo\t" << x << endl;
38 }
39
40 bar::bar() : bar('B') {
41     x = 'A';
42     cout << "Bar\t" << x << endl;
43 }
44
45 bar::bar(char y) : x(y) {
46     cout << "Bar\t" << x << endl;
47 }
48
49 bar::~bar() {
50     cout << "-Bar\t" << x << endl;
51 }
52
53 void bar::barbaz(char c) {
54     baz b(c);
55     foo f(c+1, c+2);
56 }
57
58 int main() {
59     bar f;
60     f.barbaz('X');
61     return 0;
62 }
63 /* Ausgabe:
64 */
65 /*** Ende Aufgabe_07_3.cpp ***/

```