technische universität dortmund

fakultät für informatik

Jurij Kuzmic
Horst Schirmeier
Jonas Ellert

Dortmund, den 07. Januar 2021

Übung zur Vorlesung EidP (WS 2020/21)

Blatt 7

Block rot

Es können 4 Punkte erreicht werden.

Abgabedatum: 14. Januar 2021, 23:59 Uhr

Hinweise

- Bitte beachten Sie aktuelle Hinweise unter:

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

- Verwenden Sie **keine zusätzlichen Bibliotheken** zur Lösung der Aufgaben.
- Sie sollten während der Entwicklung Ihrer Programme diese unbedingt regelmäßig – und insbesondere noch einmal vor der Abgabe – **compilieren und ausführen**.
- Auf der Übungswebseite wurde am 17.12. eine **Probeklausur** und wird am 11.01. ein **Zusatzblatt mit 6 Bonuspunkten** veröffentlicht.

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, und welchen Zusammenhang haben diese? (0.3 Punkte)

b) Welchen wesentlichen Unterschied haben `struct`- und `class`-Definitionen? (0.1 Punkte)

c) Welchen Zweck haben Konstruktoren und Destruktoren? (0.2 Punkte)

d) Erklären Sie das Prinzip des „Information Hiding“. Was hat die Aufteilung der Deklaration in `public`- und `private`-Attribute und -Methoden damit zu tun? (0.3 Punkte)

e) Erklären Sie den Begriff des Überladens von Methoden, und zeigen Sie ein einfaches C++-Beispiel. (0.3 Punkte)

f) Erklären Sie den Begriff der Delegierung von Konstruktoren. (0.1 Punkte)

g) Worauf muss man bei der Verwendung der Klassenschablonen bei Klassendefinition und Implementierung achten? (0.2 Punkte)

Aufgabe 2: Klassen (2.5 Punkte)

Für diese Aufgabe ist es notwendig, Visual Studio Code so umzukonfigurieren, dass es mehrere `.cpp`-Dateien übersetzen und zu einem Programm zusammenbinden (*linken*) kann. Dazu muss die Konfiguration der *Code Runner*-Extension (wie im Abschnitt *Einrichten und Verwenden von Visual Studio Code* der Installationsanleitung) angepasst werden:

- *File* → *Preferences* → *Settings*, dort den Punkt *Extensions* und den Unterpunkt *Run Code configuration* wählen. Bei *Executor Map* auf *Edit in settings.json* klicken.
- Hier in der "cpp"-Zeile die Variable `$filename` entfernen und vor dem zweiten `&&` die Wildcard `*.cpp` einfügen, so dass die Zeile hinterher so aussieht:
  ```
  "cpp": "cd $dir && g++ -Wall -Wextra -std=c++17 -pedantic
  -o $fileNameWithoutExt *.cpp && $dir$fileNameWithoutExt",
  ```

Beachten Sie, dass Sie nach dieser Änderung **jedes entwickelte Programm in einem eigenen Ordner ablegen müssen**, da immer *alle* `.cpp`-Dateien übersetzt und gelinkt werden. Sie können mit Rechtsklick auf den Workspace → *New Folder* einen neuen Ordner anlegen.

a) Legen Sie für diese Teilaufgabe die Datei `Vektor.h` an. Deklarieren Sie in dieser Datei die Klasse `Vektor`, die einen Vektor im mehrdimensionalen Raum mit Koordinaten vom Typ `double` repräsentieren soll. Halten Sie *nur die Deklaration* der Klasse in der Datei `Vektor.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 (sogenannte *Getter/Setter*).
3. Folgende Operationen sollen ebenfalls unterstützt werden:
   - Addition mit einem Vektor, der dieselbe Dimension n hat. Danach sind die neuen Koordinaten des Vektors das Ergebnis der Addition der jeweils korrespondierenden Koordinaten (*Vektorsumme*).
   - Multiplikation eines Vektors und einer Zahl vom Typ `double`, wobei die Zahl mit jeder Koordinate des Vektors multipliziert wird (*Skalarmultiplikation*).
   - Berechnung des *Skalarprodukts* $\langle p, q \rangle$ zweier Vektoren 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 `Vektor.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_2.cpp` an. Inkludieren Sie `Vektor.h`, also die Deklaration der zuvor erstellten Klasse `Vektor`. Schreiben Sie ein Programm, bei dem zwei Vektoren $a = (1.0, 2.0, 7.5)$ und $b = (3.0, 2.0, 0.5)$ erzeugt werden. Dann soll Vektor b zu Vektor a addiert werden und danach der Vektor b mit 1.75 multipliziert werden. Anschließend soll das Skalarprodukt der beiden Vektoren a und b ausgegeben werden. Kompilieren Sie Ihr Programm und führen Sie es anschließend aus. Kopieren Sie die Ausgabe als Block-Kommentar an das Ende der Datei `Aufgabe_07_2.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  /* Klasse: Alpha */
3  class Alpha {
4  private:
5    char x;
6  public:
7    Alpha();
8    Alpha(char y);
9    ~Alpha();
10 };
11
12 /* Klasse: Bravo */
13 class Bravo {
14 private:
15   char x;
16 public:
17   Bravo();
18   Bravo(char y);
19   Bravo(char x, char y);
20   ~Bravo();
21 };
22
23 /* Klasse: Charlie */
24 class Charlie : Bravo, Alpha {
25   char x;
26 public:
27   Charlie();
28   Charlie(char y);
29   ~Charlie();
30   void f(char c);
31 };
32 /*** Ende Aufgabe_07_3.h ***/
```

```
1  /*** Aufgabe_07_3.cpp ***/
2  #include <iostream>
3  #include "Aufgabe_07_3.h"
4  using namespace std;
5
6  /********** Klasse: Alpha **********/
7  Alpha::Alpha() : Alpha('A') {
8    cout << "Alpha\t" << x << endl;
9  }
10
11 Alpha::Alpha(char y) : x('a') {
12   char t = x;
13   x = y;
14   cout << "Alpha\t" << x << endl;
15   x = t;
```

```
16  }
17
18  Alpha::~Alpha() {
19    cout << "-Alpha\t" << x << endl;
20  }
21
22  /********** Klasse: Bravo **********/
23  Bravo::Bravo() : Bravo('B') {
24    x++;
25    cout << "Bravo\t" << x << endl;
26  }
27
28  Bravo::Bravo(char y) : x(y) {
29    cout << "Bravo\t" << x << endl;
30  }
31
32  Bravo::Bravo(char x, char y) : x(x) {
33    cout << "Bravo\t" << x << endl;
34    cout << "Bravo\t" << y << endl;
35  }
36
37  Bravo::~Bravo() {
38    cout << "-Bravo\t" << x << endl;
39  }
40
41  /********** Klasse: Charlie **********/
42  Charlie::Charlie() : Charlie('C') {
43    x = 'c';
44    cout << "Charlie\t" << x << endl;
45  }
46
47  Charlie::Charlie(char y) : x(y) {
48    cout << "Charlie\t" << x << endl;
49  }
50
51  Charlie::~Charlie() {
52    cout << "-Charlie\t" << x << endl;
53  }
54
55  void Charlie::f(char c) {
56    Alpha b(c);
57    Bravo f(c+1, c+2);
58  }
59  /********* main-Funktion *********/
60  int main() {
61    Charlie c;
62    c.f('X');
63    return 0;
64  }
65  /*** Ende Aufgabe_07_3.cpp ***/
```