technische universität dortmund

fakultät für informatik

Lutz Oettershagen
Jurij Kuzmic

Dortmund, den 06. Dezember 2018

Übung zur Vorlesung EidP (WS 2018/19)

Blatt 7

Block rot

Es können 4 Punkte erreicht werden.

Abgabedatum: 13. Dezember 2018, 23:59 Uhr

Hinweise

- Bitte beachten Sie die aktuellen Hinweise unter

  `https://ls11-www.cs.tu-dortmund.de/teaching/ep1819uebung/`

- Stellen Sie sicher, dass alle von Ihnen abgegebene Dateien reine Textdateien im UTF-8-Format sind.
- Für die Kompilierung muss ebenfalls sichergestellt werden, dass sich Ihre Programme mit den Parametern `-pedantic` und `-Werror` kompilieren lassen.
- Für die Programmieraufgaben kopieren Sie immer die Ergebnisse als Block-Kommentar an das Ende der Datei, welche das jeweilige Hauptprogramm enthält.

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 Delegierung 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) Schreiben Sie eine Klasse `Punkt`, die einen Punkt im mehrdimensionalen Raum mit Koordinaten vom Typ `double` repräsentieren soll. Halten Sie die Definition der Klasse in der Datei `Punkt.h` und Ihre Implementierung in der Datei `Punkt.cpp` fest. 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.
3. Folgende Operationen sollen 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 euklidischen Abstands $d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$ zwischen zwei Punkten p, q mit je n Koordinaten. Dies entspricht der Länge des Vektors $\overrightarrow{pq}$.
   - Berechnung des *Skalarprodukts* $\langle p, q\rangle$ zweier Punkte p, q mit je n Koordinaten durch $\langle p, q\rangle := \sum_{i=1}^{n} p_i \cdot q_i$.
   - Berechnung des Abstands eines Punktes mit n Koordinaten zum Ursprung (entspricht Ortsvektor).

Legen Sie eine Datei `Aufgabe_07_2a.cpp` an. 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. Geben Sie anschließend den Abstand der resultierenden Punkte aus. Kompilieren Sie Ihr Programm und führen Sie es anschließend aus.

Hinweis: In der C++-Bibliothek `cmath` befindet sich die Funktion `sqrt(double x)`, welche für die Variable x den Wert $\sqrt{x}$ berechnet. (1.5 Punkte)

b) Zwei Punkte a und b induzieren eine Gerade. Schreiben Sie nun eine Klasse `Gerade`, die eine Gerade im mehrdimensionalen Raum beschreiben soll, und eine Methode besitzt, die überprüft, ob ein gegebener Punkt c auf dieser Gerade liegt. Nutzen Sie hierfür den geometrischen Zusammenhang

$$\langle p, q\rangle = \|\vec{p}\|\|\vec{q}\| cos(\alpha),$$

wobei $\|\vec{p}\|$ die euklidischen Länge des Ortsvektors $\vec{p}$ von p bezeichnet und α den zwischen $\vec{p}$ und $\vec{q}$ eingeschlossenen Winkel. Überlegen Sie sich, wie Sie diesen Zusammenhang nutzen können, um den Test durchzuführen.

Geben Sie die Definition und Implementierung der Klasse `Gerade` in den Dateien `Gerade.h` und `Gerade.cpp` an. Erweitern Sie Ihr Programm in der Datei `Aufgabe_07_2b.cpp` aus Aufgabenteil a) um die Erzeugung einer Gerade mit den (Original-)Punkten a und b aus Aufgabenteil a). Geben Sie anschließend für die Punkte $c = (4.2, 5.6, 0.7)$ und $d = (3.4, 10.0, -10.6)$ an, ob sie kollinear mit den Punkten a und b sind (also auf der Gerade liegen). Kompilieren Sie Ihr Programm und führen Sie es anschließend aus.

Hinweise:

1. Falls Sie in der Teilaufgabe b) einen Wert a mit 0 vergleichen wollen, ist es sinnvoll, die Bedingung $a = 0$ durch $a < e$ zu ersetzen, wobei e eine sehr kleine positive Konstante, z. B. $e = 0.0000001$ ist. Überlegen Sie, warum dies sinnvoll ist.

2. Falls Sie in der Datei `Gerade.h` die Datei `Punkt.h` inkludieren, so müssen Sie in der Datei `Aufgabe_07_1b.cpp` nur `#include "Gerade.h"` einfügen. Die Datei `Punkt.h` wird dann bereits indirekt inkludiert, d. h. in der Datei `Aufgabe_07_1b.cpp` steht dann auch die Definition der Klasse Punkt zur Verfügung. Sollten Sie beim Kompilieren die Fehlermeldung erhalten, dass die Klasse Punkt mehrfach definiert wurde (*redefinition of 'class Punkt'*), so kann dies daran liegen, dass Ihre Datei sowohl die Direktive `#include "Punkt.h"` als auch die Direktive `#include "Gerade.h"` enthält. Entfernen Sie in diesem Fall `#include "Punkt.h"`. (In der Vorlesung werden Sie *Include guards* kennen lernen, um diesen Fehler zu in Zukunft anders zu verhindern.)

(1 Punkt)

Präsenzaufgabe 3: Konstruktoren und Destruktoren (0 Punkte)

Geben Sie die Ausgabe des folgenden Programms an.

```
1  /* foobarbaz.h */
2  // Header
3  class baz {
4  private:
5    char x;
6  public:
7    baz();
8    baz(char y);
9    ~baz();
10 };
11
12 class foo {
13 private:
14   char x;
15 public:
16   foo();
17   foo(char y);
18   foo(char x, char y);
19   ~foo();
20 };
21
22 class bar : foo, baz {
23 private:
24   char x;
25 public:
```

```
26    bar();
27    bar(char y);
28    ~bar();
29    void barbaz(char c);
30  };
```

```
1   /* foobarbaz.cpp */
2   #include <iostream>
3   #include "foobarbaz.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  }
```