technische universität dortmund
fakultät für informatik

Amer Krivošija
Denis Kurz
Eugen Rusakov

Dortmund, den 15. Dezember 2016

Übungen zur Vorlesung EidP (WS 2016/17)

Blatt 8

Block rot

Es können 4 Punkte und 3 Bonuspunkte erreicht werden.

Abgabedatum: 22. Dezember 2016 23:59 Uhr

Hinweise

1) Bitte beachten Sie die aktuellen Hinweise unter

   `https://ls11-www.cs.tu-dortmund.de/teaching/ep1617uebung/`

2) In der optionalen Datei `Anmerkungen.txt` können Sie allgemeine Anmerkungen bezüglich Ihrer Lösungen notieren.
3) Stellen Sie sicher, dass alle von Ihnen abgegebene Dateien reine Textdateien im UTF-8-Format sind.
4) Dies ist das letzte Blatt im roten Block. Das bedeutet, dass Sie spätestens mit der Abgabe Ihrer Lösungen zum vorliegenden Blatt die zum Bestehen des roten Blocks nötigen 8 Punkte erreichen müssen. Wenn Sie die nötigen Punkte noch nicht erreicht haben, haben Sie mit Aufgabe 2 auf diesem Blatt die Möglichkeit, Bonuspunkte zu erhalten.

Aufgaben

Aufgabe 0: Grundlagen (1 Punkt)

Legen Sie für Ihre Antworten eine Text-Datei *Aufgabe_ 08_ 0.txt* an.

a) Erklären Sie das Prinzip des „information hiding“. (0,2 Punkte)

b) Erklären Sie den Begriff des Überladen der Methoden. (0,2 Punkte)

c) Erklären Sie den Zweck der Teilung der Deklaration in `public` und `private`. (0,2 Punkte)

d) Erklären Sie den Begriff der Delegierung von Konstruktoren. (0,2 Punkte)

e) Worauf muss man bei der Verwendung der Klassenschablonen bei Klassendefinition und Implementierung achten? (0,2 Punkte)

Speichern Sie Ihre Ergebnisse in der Ergebnisdatei *Aufgabe_ 08_ 0.txt*.

Aufgabe 1: Klassen (3 Punkte)

a) Schreiben Sie eine Klasse `Punkt`, die einen mehrdimensionalen Punkt 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. Da ein Punkt x einem Vektor $\overrightarrow{x}$ entspricht, der vom Ursprung zum x läuft, soll die Klasse folgende Eigenschaften haben:

i) Die Dimension d muss bei Erzeugung eines Objekts angegeben werden und kann anschließend nicht mehr geändert werden.

ii) Die Koordinaten sollen sinnvoll initialisiert werden und einzeln abruf- und veränderbar sein.

iii) Folgende Operationen sollen unterstützt werden:

- Addition mit einem Punkt, der die gleiche Anzahl an Dimensionen hat. Danach sind die neuen Koordinaten des Punktes das Ergebnis der Addition. Dies entspricht der Summe zweier Vektoren.
- Multiplikation von Punkt und Zahl vom Typ `double`, wobei die Zahl mit jeder Koordinate des Punktes multipliziert wird.
- Berechnung des euklidischen Abstands $d(p,q) = \sqrt{\sum_{i=1}^{d}(p_i - q_i)^2}$ zwischen zwei Punkten p, q mit Dimension d. Dies entspricht der Länge des Vektors $\overrightarrow{pq}$.
- Berechnung des Skalarprodukts $< p, q >$ zweier Punkte $p, q \in \mathbb{R}^d$ mit

$$< p, q >:= \sum_{i=1}^{d} (p_i \cdot q_i) \,.$$

  Dies entspricht dem Skalarprodukt zweier Vektoren.

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

Legen Sie eine Datei *Aufgabe_ 08_ 1.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 neu definierten Punkte A und B aus. Kompilieren Sie ihr Programm und führen Sie es anschließend aus. Kopieren Sie die Ergebnisse als Block-Kommentar an das Ende der Datei *Aufgabe_ 08_ 1a.cpp*. (2 Punkte)

b) Zwei Punkte A und B definieren eine gerade Linie. Schreiben Sie nun eine Klasse `Linie`, die eine gerade Linie im mehrdimensionalen Raum beschreiben soll, und eine Methode besitzt, die überprüft, ob ein gegebener Punkt C kollinear mit dieser Linie ist.

Notieren Sie die Definition und Implementierung der Klasse `Linie` in den Dateien *Linie.h* und *Linie.cpp*. Erweitern Sie ihr Programm in der Datei *Aufgabe_ 08_ 1.cpp* aus Aufgabenteil a) um die Erzeugung einer Linie mit (Original-)Punkte 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 Punkten A und B sind. Kompilieren Sie ihr Programm und führen Sie es anschließend aus. Kopieren Sie die Ergebnisse als weiteren Block-Kommentar an das Ende der Datei *Aufgabe_ 08_ 1b.cpp*. (1 Punkt)

Hinweise:

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

ii) Falls Sie in der Datei *Linie.h* die Datei *Punkt.h* inkludieren, so müssen Sie in der Datei *Aufgabe_08_1b.cpp* nur `#include "Linie.h"` einfügen. Die Datei *Punkt.h* wird dann bereits indirekt inkludiert, d. h. in der Datei *Aufgabe_08_1.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 "Linie.h"` enthält. Entfernen Sie in diesem Fall `#include "Punkt.h"`.

Bonusaufgabe 2: Zeiger (3 Punkte)

Gegeben sei das folgende C++-Programm:

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6          char a[] = { 'a', 'b', 'c', 'd', 'e', '\0' };
7          // Zeitpunkt 0
8
9          char *pa = &a[3];
10         *(pa-2) = *pa-3;
11         // Zeitpunkt 1
12
13         char *p = &a[4];
14         *(a+4) -= 5;
15         (*p)++;
16         // Zeitpunkt 2
17
18         char c = 'A', d = 's';
19         d -= *a;
20         a[0] = d+*(&c);
21         // Zeitpunkt 3
22
23         struct {
24                 int u, v;
25                 char* w, *y;
26         } s, *ps = &s;
27         (*ps).u = 11;
28         ps->v = s.u+5;
29         ps->w = a+2;
30         s.y = s.w+1;
31         *(*ps).w += s.u;
32         *(*ps).y += (&s)->v;
33         // Zeitpunkt 4
34
```

```
35          cout << a << endl;
36
37          return 0;
38  }
```

Wir interessieren uns für die Werte der Variablen an den mit `Zeitpunkt 1`, `Zeitpunkt 2` usw. gekennzeichneten Zeilen, sowie den Wert von `a`, der in der Zeile 35 ausgegeben wird. Vervollständigen Sie die Tabelle. Sie können die Werte von Variablen vom Typ `char` entweder als Zeichen (Variante 1) oder den Dezimalwert in der ASCII-Kodierung[1] (Variante 2) angeben. Geben Sie den Wert von `a` unter der Tabelle an.

Zeitpunkt	`a[0]`	`a[1]`	`a[2]`	`a[3]`	`a[4]`	`a[5]`	`c`	`d`	`s.u`	`s.v`	`s.w`	`s.y`
0 (Variante 1)	`'a'`	`'b'`	`'c'`	`'d'`	`'e'`	`'\0'`	-	-	-	-	-	-
0 (Variante 2)	97	98	99	100	101	0	-	-	-	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Speichern Sie Ihr Ergebnis in der Ergebnisdatei *Aufgabe_ 08_ 2.txt*. Eine vorformatierte Vorlage dieser Datei ist auf der Website der Übung zu finden.

Hinweis: Benutzen Sie für diese Aufgabe keinen Debugger! Dies ist eine typische Klausuraufgabe (in der Klausur wird natürlich keine ASCII-Kodierung benutzt), und dort wird Ihnen auch kein Debugger zur Verfügung stehen.

[1] Siehe z.B. `http://de.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange`