technische universität dortmund

fakultät für informatik

**Martin Apel Informatik VI, GB5/R 431; Tel.: 4775**
**Nicola Beume Informatik XI, OH14/R 233; Tel.: 7704**
**Jürgen Mäter Informatik IV, GB5/R 402; Tel.: 2411**

Dortmund, den 14. Januar 2011

# Übungen zur Vorlesung EINI (WS 2010)

# Blatt 11

Block blau

**Es können 7 Punkte erreicht werden.**

Abgabedatum: 23. Januar 2010 23:59 Uhr

## Hinweise

1) Bitte beachten Sie die aktuellen Hinweise unter

   **http://www4.cs.uni-dortmund.de/home/maeter/UeEiniET10W/**

   Für die Abgabe sind die Dateien *Aufgabe_11_0.txt*, *Aufgabe_11_1.txt* und *Aufgabe_11_1.cpp* zu erstellen . Für die optionale Aufgabe legen Sie bitte die Dateien *Aufgabe_11_2a.cpp* und *Aufgabe_11_2b.cpp* an.
2) Es ist ratsam, die Programme vor der Abgabe zu kompilieren und auszuführen.
3) Verwenden Sie für die Textaufgaben reine Texteditoren. Das Abgabesystem erkennt **keine** Word- oder PDF-Dateien!!!
4) In der optionalen Datei *Anmerkungen.txt* können Sie allgemeine Anmerkungen bezüglich Ihrer Lösungen notieren.
5) Machen Sie sich im Vorhinein Gedanken, wie Sie die zu erstellenden Programme zu implementieren sind. Machen Sie sich also zuerst einen Programmentwurf.
6) Kommentieren Sie Ihre Quelltexte. Das gehört zur Aufgabenstellung und wird mitbewertet.
7) Verwenden Sie bei Ihrer Implementierung den `C++`-Standard (insbesondere bei der Deklaration von Feldern).

## Spruch des Tages

**(frei nach Murphy)**

# Ein Programm, das fehlerfrei läuft, ist längst überholt.

# Aufgaben

## Aufgabe 0: Grundlagen (1 Punkt)

Legen Sie für Ihre Antworten eine Text-Datei *Aufgabe_11_0.txt* an.

a) Erklären Sie folgende Begriffe aus der objektorientierten Programmierung: `Klasse`, `Objekt` und `Methode`. (0,3 Punkte)

b) Wie können Oberklassen auch genannt werden? (0,1 Punkt)

c) Wie werden `virtuelle Methoden` gekennzeichnet und wann werden sie gebunden? (0,2 Punkte)

d) Wie werden `rein virtuelle Methoden` definiert? (0,1 Punkt)

e) Wann nennt man eine Klasse `abstrakt`? (0,1 Punkt)

f) Wie werden `abstrakte Klassen` instanziiert? (0,1 Punkt)

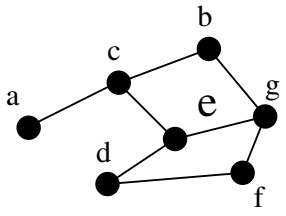
g) Wozu werden `abstrakte Klassen` eingesetzt? (0,1 Punkt)

Speichern Sie Ihre Ergebnisse in der Ergebnisdatei *Aufgabe_11_0.txt*.

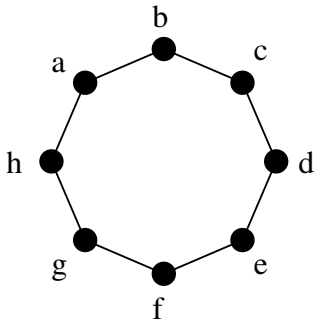
## Aufgabe 1: Graphen (3 Punkte)

In der Vorlesung haben Sie Graphen als abstrakte Datentypen kennen gelernt.

a) Legen Sie für Ihre Antworten eine Text-Datei *Aufgabe_11_1.txt* an. Geben Sie für die folgenden beiden Beispiele jeweils den zugehörigen Graphen $G = (V, E)$ an (Knotenmenge $V$ und Kantenmenge $E \subseteq V \times V$). Welchen Maxgrad $\triangle(G) = \max\{deg(v) : v \in V\}$, Mingrad $\delta(G) = \min\{deg(v) : v \in V\}$ und Durchmesser $diam(G) = \max\{dist(u, v) : (u, v) \in V \times V\}$ besitzen die beiden Graphen jeweils? Geben Sie weiterhin jeweils eine Repräsentation dieser Graphen mit Hilfe einer Adjazenzmatrix und mit Hilfe einer Adjazenzliste an. Welche Besonderheit weisen die Adjanzenzmatrizen auf? (1,5 Punkte)




(a)




(b)

b) Legen Sie eine cpp-Datei *Aufgabe_11_1.cpp* an, implementieren und kommentieren Sie den ADT Graph in C++ unter Verwendung von *Adjazenzmatrizen* zur Darstellung der Kantenmenge! Sie können sich an der in Vorlesung angegebenen Implementation orientieren, müssen jedoch die einzelnen Methoden (`addEdge`, `deleteEdge`, `hasEdge`, `noOfEdges`, `noOfNodes`, `printGraph` sowie den Konstruktor) entsprechend anpassen. Sie dürfen zur

Vereinfachung die Knotennamen `a, b, c, ...` auf die Zahlen `0, 1, 2, ...` abbilden. Der Konstruktor `Graph(uint NoOfNodes)` könnte demnach folgende Gestalt haben:

```
1  Graph::Graph(uint NoOfNodes) {
2    mNoOfNodes = NoOfNodes;
3    AdjacencyMatrix = new uint*[NoOfNodes];
4    for (uint i=0; i<NoOfNodes; i++)
5         AdjacencyMatrix[i] = new uint[NoOfNodes]
6
7    }
```

Testen Sie ihre Implementation, indem Sie in der `main`-Funktion die beiden oben angegebenen Beispiele erzeugen und ausgeben lassen. (0,6 Punkte)

c) Erweitern Sie ihre Implementation um zwei Methoden `int maxGrad()` und `int minGrad()`, welche jeweils den MaxGrad bzw. MinGrad des entsprechenden Graphen berechnen und zurückgeben. Testen Sie diese Methoden anhand der beiden obigen Beispiele durch Anpassung der `main`-Funktion! (0,4 Punkte)

d) Existiert in einem Graphen $G = (V, E)$ ein Weg von einem Knoten $s \in V$ zu einem Knoten $t \in V$, so ist $t$ von $s$ aus *erreichbar*. Existiert kein solcher Weg, so ist $t$ von $s$ aus *nicht erreichbar*. Erweitern Sie Ihre Implementation um eine Methode

`reachable(uint Node1, uint Node2)`,

welche `true` zurückgibt, wenn der Knoten `Node2` vom Knoten `Node1` aus erreichbar ist. Ansonsten soll die Methode `false` zurückgeben. Testen Sie diese Methode mit folgendem Beispiel: Löschen Sie die Kanten von `c` nach `e` und `b` nach `g`, testen Sie die Erreichbarkeit von `a` nach `b` und von `a` nach `f`. (0,5 Punkte)

Speichern Sie alle Ergebnisse als Block-Kommentar an das Ende der Datei *Aufgabe_11_1.cpp*.

## (Optionale) Aufgabe 2: Ausnahmebehandlung und Schablonen (3 Punkte)

a) Ausnahmebehandlung (1,5 Punkte)

**Hinweis**: Bei dieser Aufgabe handelt es sich um eine ehemalige leicht modifizierte Prüfungsaufgabe.

In dem unten angegebenen Programm wird in der Funktion `Formel` die Summe zweier Zahlen x und y durch ihre Quadratwurzel geteilt. Folglich darf die Summe weder Null noch negativ sein, da dann aus unterschiedlichen Gründen Fehler auftreten können. Falls ein Fehler auftritt, dann enthält die Variable `fehler` einen positiven Fehlercode, der nach jedem Aufruf abgeprüft werden muss.

```
1  #include <iostream>
2  #include <math.h>
3  using namespace std;
4  double Formel(double x, double y, int &fehler) {
5     double sum = x + y;
```

```
 6     fehler = 0;
 7     if (sum < 0) {
 8        fehler = 1;
 9        return 0;
10     }
11     if (sum == 0) {
12        fehler = 2;
13        return 0;
14     }
15     return sum / sqrt(sum);
16  }
17  int main() {
18     int fehler;
19     double x[4] = { 3., -2., -4., 0. };
20     double y[4] = { 1., 2., 3., 1. };
21     for (int i = 0; i < 4; i++) {
22        for (int j = 0; j < 4; j++){
23           double z = Formel(x[i], y[j], fehler);
24           switch (fehler) {
25           case 1: cerr << "negative Summe" << endl;
26              break;
27           case 2: cerr << "Division durch Null" << endl;
28              break;
29           default: cout << z << endl;
30           }
31        }
32     }
33     return 0;
34  }
```

Dieses umständliche Programm lässt sich in C++ mit Ausnahmen eleganter und auch weniger fehleranfällig formulieren. Das sind Ihre Aufgaben:

(a) Legen sie eine cpp-Datei *Aufgabe_11_2a.cpp* an. Schreiben Sie die Funktion und das Hauptprogramm um, indem Sie den Mechanismus der Ausnahmebehandlung aus C++ mit `try`, `throw` und `catch` verwenden. Die Ausgabe des Programms soll sich nicht ändern! Verwenden Sie für jede Fehlerart einen eigenen `catch`-Handler! (1 Punkt)

(b) Testen und kommentieren Sie Ihr Programm. Kopieren Sie die Ergebnisse als Kommentar an das Ende der cpp-Datei *Aufgabe_11_2a.cpp*. (0,5 Punkte)

b) Schablonen (1,5 Punkte)

**Hinweis**: Bei dieser Aufgabe handelt es sich um eine ehemalige Prüfungsaufgabe.
**Hinweis**: Der Stoff für diese Aufgabe wird erst im Laufe der nächsten Woche behandelt.

Betrachten Sie den folgenden Programmtext:

```
1  int PosSumInt(unsigned int size, int array[]) {
2      int posSum = 0;
3      for (unsigned int i=0; i<size; ++i)
4          if (array[i]>0) posSum+=array[i];
5      return posSum;
6  }
7
8  double PosSumDbl(unsigned int size, double array[]) {
9      double posSum = 0;
10     for (unsigned int i=0; i<size; ++i)
11         if (array[i]>0) posSum+=array[i];
12     return posSum;
13 }
14
15 int main() {
16     int ia [] = {1, -1, 7, -6, -3, 0, 2};
17     double da [] = { 1.3, -3.2, 0.1, -2.7};
18     cout << PosSumInt(7, ia) << ' ' << PosSumDbl(4,da) << endl;
19     return 0;
20 }
```

Offensichtlich existieren hier die zwei Funktionen `PosSumInt` und `PosSumDbl`, die zwar das gleiche Verhalten haben, jedoch auf verschiedenen Datentypen arbeiten.

(a) Legen sie eine cpp-Datei *Aufgabe_11_2b.cpp* an, kommentieren und implementieren Sie mittels der Schablonentechnik aus C++ eine Schablonen-Funktion `PosSum`, die die Summe aller positiven Elemente im übergebenen Feld zurück gibt und für jeden eingebauten numerischen Datentyp instantiiert werden kann. (0,5 Punkte)

(b) Geben Sie auch eine veränderte Form des Hauptprogramms an, welche die Schalonen-Funktion `PosSum` verwendet. (0,5 Punkte)

(c) Kopieren Sie die Ergebnisse als Kommentar an das Ende der cpp-Datei *Aufgabe_11_2b.cpp*. (0,5 Punkte)

# Das Eini-Team wünscht Ihnen viel Erfolg für die Klausur und Ihr weiteres Studium!