BERGISCHE UNIVERSITÄT
GESAMTHOCHSCHULE WUPPERTAL
GAUSS-STRASSE 20
42097 WUPPERTAL
(Korrespondenzanschrift)
42119 WUPPERTAL
(Lieferanschrift)
TELEX 8 592 262 bughw
TELEFAX (0202) 439-2901
TELEFON (0202) 439-1

Fachbereich 7
MATHEMATIK

Prof. Dr. Hans-Jürgen Buhl
Praktische Informatik / Numerik

e-mail: Juergen.Buhl@math.uni-wuppertal.de

Grundzüge der objektorientierten Programmierung

WS2001/2002 – Übungsblatt 12

Abgabetermin: 4. Februar 2002

Aufgabe 1. *Generische Klassen, 3 Punkte*
Die Dateien zu den in der Vorlesung eingeführten Matrix-Vektor-Operationen erhalten Sie über:

http://www.math.uni-wuppertal.de/~buhl/oop/MatrixVektor/

Verwenden Sie anstelle der bisherigen Klassen Template-Klassen, die sowohl für den Datentyp `double` als auch für den Datentyp `complex<double>` ausgeprägt werden können.

Testen Sie die Funktionsfähigkeit mit einem modifizierten Hauptprogramm.

Aufgabe 2. *Anschriftenliste, 5 Punkte*
Schreiben Sie ein Programm, das als Grundlage für die Verwaltung einer Anschriftenliste verwendet werden kann. Das Programm soll Anschriften von Freunden, Bekannten und Firmen einlesen, speichern und übersichtlich ausgeben. Dabei soll eine einzulesende Adresse die Form

(a) *Anrede Vorname Name Straße Hausnummer Postleitzahl Ort*

haben. Sie soll dann in der Form

(b)
Anrede
Vorname Name
Straße Hausnummer
Postleitzahl Ort

ausgegeben werden.
Schreiben Sie ein Programm, das diese Aufgabe mit Hilfe einer Klasse namens `anschrift` löst. Die Klasse soll dabei enthalten:

- **Als Datenelemente:**
  - Jeweils ein Datenelement vom Typ `char` und eines vom Typ `string`, um *Anrede* und *Postleitzahl* abzuspeichern. *Anrede* soll ein einzelnes Zeichen sein, dessen Wert bestimmt, was als Anrede (z. B. Frau, Herrn, ...) ausgegeben werden soll.

  - Fünf Komponenten vom Typ `string`, um *Vorname*, *Name*, *Straße*, *Hausnummer* und *Ort* abzuspeichern.

- **Als Friendfunktionen:**
  - Einen Eingabeoperator `>>`, der sämtliche Datenelemente eines Objektes der Klasse `anschrift`, die in der Form (a) gegeben sind, einliest.
  - Einen Ausgabeoperator `<<`, der die Datenelemente in der Form (b) ausgibt. Dabei soll, falls *Anrede* den Wert `h` oder `H` bzw. `f` oder `F` hat, `Herrn` bzw. `Frau` ausgegeben werden. Falls *Anrede* einen anderen Wert hat, soll nichts ausgegeben werden. Außerdem soll die Hausnummer nicht ausgegeben werden, falls sie nicht mit einer Zahl beginnt.

Verwenden Sie die Klasse in einem Hauptprogramm, das zunächst die Anzahl `n` der zu speichernden Anschriften einliest. Damit soll ein dynamisches Feld der Länge `n` aus Objekten der Klasse `anschrift` allokiert werden. Danach sollen sämtliche Datenelemente der `n` Objekte gelesen und zum Schluß wieder ausgegeben werden. Geben Sie den gesamten allokierten Speicherplatz vor Verlassen des Programms wieder frei!
Testen Sie Ihr Programm mit 4 Adressen Ihrer Wahl.

Beispiel:
Die Eingabe

```
f Ana Lysis Mathestraße 1a 40465 Satzwald
h Gerd Müller Siegerstraße 100 54740 Jubel
h F. Reund In-der-Kneipe 1-3 11111 Schluckbach
x Firma Profit Geldplatz x 99999 Kohlen
```

sollte die folgende Ausgabe erzeugen:

```
Frau              Herrn             Herrn
Ana Lysis         Gerd Müller       F. Reund            Firma Profit
Mathestraße 1a    Siegerstraße 100  In-der-Kneipe 1-3   Geldplatz
40465 Satzwald    54740 Jubel       11111 Schluckbach   99999 Kohlen
```

Sie können die Adressen auch untereinander ausgeben.

Aufgabe 3. *Syntaktische und semantische Fehler, 3 Punkte*
Das nachstehend angegebene C++-Programm soll einem Objekt `a` der zuvor definierten Klasse `matrix` mittels des Operators `+=` die Summe `a + a` zuweisen und das Ergebnis ausgeben. Leider wird es dies nicht tun, denn es enthält insgesamt 8 syntaktische bzw. semantische Fehler (keine logischen Fehler, keine Rechtschreibfehler).
Finden Sie die fehlerhaften Programmzeilen heraus. Die Zeilennummern am linken Rand des Programmtextes zählen dabei nicht zum Programm, sondern sollen lediglich Ihrer Orientierung dienen.
Notieren Sie jeweils die Zeilennummer, in der Sie einen Fehler entdeckt haben, und beschreiben Sie **kurz**, was in dieser Zeile falsch ist. Jeder entdeckte Fehler soll in allen nachfolgenden Zeilen als korrigiert gelten, so daß keine Folgefehler auftreten können.

```
1      #include <iostream>
2
3      class matrix
4      {
5         double **data;
6         int dim;
7       public:
8         matrix(int n); // Konstruktor
9         ~matrix();     // Destruktor
10        matrix& operator += (const matrix&);
11     }
12
13     matrix::matrix(int n)
14     {
15       dim=n;
16       data = new double[n];
17       for (int i=0;i<n;i++) data[i] = new double[n];
18     }
19
20     matrix& operator += (const matrix& b)
21     {
22       int i,j;
23       for (i=0;i<dim;i++)
24         for (j=0;j<dim;j++)
25           data[i][j]=data[i][j]+b[i][j];
26       return *this;
27     }
28

29     void matrix::~matrix()
30     {
31     for (int i=0;i<dim;i++)
32       delete[] data[i];
33     delete[] data;
34     }
35
36     int main()
37     {
38       matrix a;
39       a+=a;
40       cout << a;
41       return 0;
42     }
```

Aufgabe 4. *Vererbung, 3 Punkte*
Gegeben ist das folgende Programm:

```
class C {
   protected: int c;
};

class D {
```

```
   protected: int d;
};

class E: public D {
   protected: int e;
};

class A: public D {
   private: int a;
};

class B: public E, A, C {
   protected: int b;
};

class F: public A, C {
   protected: int f;
};

int main() {
   B obj;
   return 0;
}
```

Bearbeiten Sie schriftlich:

- Zeichnen Sie einen Ableitungsgraphen.
- Welche Datenmember besitzt das Objekt `obj` und welche sind mit `obj.` ansprechbar? Zeichnen Sie ein Diagramm, aus dem die enthaltenen Subobjekte hervorgehen.

Aufgabe 5. *Vererbung, Polymorphie, 5 Punkte*
Es soll in C++ eine Fahrzeugdatenbank für verschiedene Fahrzeugtypen erstellt werden. Ihr Programm soll dabei enthalten:

- Eine Klasse `Fahrzeug` mit dem `int`-Datenmember `AnzahlRaeder` und einer parameterlosen rein virtuellen Elementfunktion `info`.
- Eine von `Fahrzeug` abgeleitete Klasse `Kfz` mit dem zusätzlichen Datenmember `KW` (Kilowatt) und ebenfalls einer virtuellen Elementfunktion `info`.
- Von der Klasse `Kfz` sollen die Klassen `PKW` und `LKW` mit den zusätzlichen Datenmembern `Sitzplaetze` bzw. `Ladung` abgeleitet werden. Beide Klassen sollen eine parameterlose Elementfunktion `info` erhalten.
- Eine von der Klasse `Fahrzeug` abgeleitete Klasse `Zweirad` mit einem zusätzlichen Datenmember `Name` vom Typ `string` und einer Elementfunktion `info`.

- Ein Hauptprogramm, in dem zunächst eine Variable `Motorboot` vom Typ Zeiger auf `Kfz`, eine Variable `Sattelschlepper` von Typ Zeiger auf `LKW`, eine Variable `Cabrio` vom Typ Zeiger auf `PKW` und zwei Variablen `Tandem` und `Roller` vom Typ Zeiger auf `Zweirad` angelegt werden. Zu jeder dieser Zeigervariablen soll danach ein zum Typ passendes Objekt erzeugt und mit sinnvollen Daten (siehe nachfolgende Beispielausgabe) initialisiert werden.

  Legen Sie dann ein Feld `Datenbank` mit 5 Elementen vom Typ Zeiger auf `Fahrzeug` an. Weisen Sie die Adressen der angelegten Objekte den Zeigerelementen des Feldes `Datenbank` zu. Rufen Sie danach innerhalb einer `for`-Schleife die jeweilige Funktion `info()` auf.

Die verschiedenen Funktionen `info` sollen jeweils einen Text so ausgeben, daß vom Hauptprogramm die folgende Ausgabe möglich ist:

```
Ein Fahrzeug mit 0 Raedern und einem Motor mit 10 KW.
Ein Auto mit 4 Raedern und einem Motor mit 98 KW und 2 Sitzplaetzen.
Ein Laster mit 6 Raedern und einem Motor mit 234 KW und 28.5 Tonnen Ladegewicht.
Ein Zweirad mit Namen Tandem.
Ein Zweirad mit Namen Roller.
```

Jede Klasse soll mit einem geeigneten Konstruktor versehen sein. Bei den Konstruktoren sollen jeweils Initialisierungslisten verwendet werden.