6. Übung für Programmierpraktikum

Abgabetermin: 29. Jänner 2004

Name:
Matrikelnummer: Punkte:

Die Übungen sind grundsätzlich allein zu machen. Gruppenarbeit ist nicht erlaubt. Abzugeben sind jeweils das sinnvoll dokumentierte Programmlisting mit Original-inputs und Original-outputs, falls angebracht für mehrere Testläufe mit unterschiedlichen Eingabedaten. Das Abgabeformat ist DIN A4. Heften Sie alle Unterlagen gemeinsam mit dem Übungsblatt zusammen !

Generelle Hinweise:
Werden Variablen aus der Parameterliste in einer Methode als Konstante behandelt, d.h., nicht verändert, dann ist dies auch in der Parameterliste kenntlich zu machen!
Globale Variablen sind nicht nötig und auch nicht erlaubt.

1. **(8 P)** Schreiben Sie eine Klasse DATUM, welche Tag, Monat und Jahr speichert. Implementieren Sie (Header- und Sourcefile) alles so, daß das folgende Programm[1] funktioniert.

```
1   #include <iostream.h>
2   #include "datum.hh"
3
4   int main()
5   {
6     Datum aa( 4, 12, 1983);
7     Datum bb(aa);
8     Datum cc(31,  5, 1983);
9     Datum dd;
10
11    dd = cc;
12    cout << "dd : " << dd << endl;
13    if ( aa < dd )        // aa frueheres Datum als dd?
14     { cout << aa << " frueher als " << dd << endl; }
15    else
16     { cout << aa << " nicht frueher als " << dd << endl; }
17
18    if ( bb == aa )
19     { cout << "aa und bb sind gleich" << endl; }
20
21    return 0;
22  }
```

   Hinweis: Implementieren Sie entweder zuerst den <<-Operator für Ihre Klasse DATUM oder kommentieren Sie die entsprechenden Zeilen im Programm anfangs aus. Ansonsten erscheinen lange kryptische Fehlermeldungen beim Kompilieren.

[1] http://www.numa.uni-linz.ac.at/Staff/haase/Lectures/Kurs-C/WS03/uebung6/u6A.cc

2. **(4 P)** Erweitern Sie die Klasse Datum um einen Konstruktor, der die Daten aus einem File einliest, dessen Name diesem Konstruktor als Parameter übergeben wird (§9.2 der Vorlesung). Das entsprechende Hauptprogramm ändert sich bzgl. Aufg. 1 derart, daß Zeilen 8–11 durch die Zeile
`Datum dd("input.6B_1.txt");`
ersetzt werden. Die Existenz des Files muß im Rahmen dieser Übung nicht überprüft werden.

   Eingabedaten (*tag*,*monat*,*jahr*) in den Files *input.6B_1.txt*, *input.6B_2.txt*.

   Zusatzpunkte (+**2**):
   Übergeben Sie den Filenamen als Kommandozeilenparameter (§7.6 der Vorlesung) an das Programm, sodaß Sie das Programm mit
   `./a.out input.6B_1.txt` bzw. `./a.out input.6B_2.txt`
   starten können.

3. **(12 P)** Implementieren Sie eine eigene Vektorklasse, welche den benötigten Speicher dynamisch anfordert, sodaß das folgende Programm[2] funktioniert. Die Zeilen 14–16 beinhalten eine Zusatzaufgabe (+**2**).

```
 1  #include <iostream.h>
 2  #include "vektor.hh"
 3  int main()
 4  {
 5    const double ALPHA = 3.5;
 6    const Vektor x("input.6C_x.txt"), y("input.6C_y.txt");
 7          Vektor z( x.length() ), u;
 8
 9    z.daxpy( ALPHA, x, y );
10
11    cout << "  <z,x> = " << InnerProd(z,x)
12         << "        ||z|| = " << z.Norm() << endl;
13
14    u = x * ALPHA + y;                              // Zusatzaufg.
15    cout << "  <u,x> = " << InnerProd(u,x)
16         << "        ||u|| = " << u.Norm() << endl;
17
18   return 0;
19  }
```

   Die DaXpY-Operation (**D**oppelt-genau: **a**lpha-mal-Vektor-**X**-**p**lus-Vektor-**Y**) in Zeile 9 ist identisch zu der Operation in Zeile 14 und bedeutet:

$$z_i := \alpha * x_i + y_i \qquad \forall i = 0, \ldots, n-1$$

   Die Methode NORM und die Funktion INNERPROD entsprechen den Definitionen in Aufgabe 1 der Übung 4.

   Eingabedaten (n,$x[0], \ldots, x[n-1]$) sind in den Files *input.6C_x.txt*, *input.6C_y.txt*.

[2] http://www.numa.uni-linz.ac.at/Staff/haase/Lectures/Kurs-C/WS03/uebung6/u6C.cc