

8. Übung des Programmierpraktikums (MATH)

Abgabetermin: 30. Juni 2009, 23:59 Uhr

Die Übungen sind grundsätzlich allein zu machen. Gruppenarbeit ist nicht erlaubt. Abzugeben sind jeweils die sinnvoll dokumentierten Programmfiles (Files für Bsp. 5: `bsp_5.cpp`, `bsp_5_fkt.cpp`, `bsp_5_fkt.hpp`) indem Sie diese in Ihr Verzeichnis im globalen Abgabeordner kopieren.

29. Ersetzen Sie die statischen C-Felder in Aufg. 19 (6. Übung) durch die Vektor-klasse `vector<double>` welche eine dynamische Länge des Vektors während der Laufzeit erlaubt, siehe dazu §2.3.4¹ des Skriptes. Schreiben Sie Ihre Funktionen aus Aufg. 19 für obige Vektor-klasse um und speichern Sie diese separat in Header- und Quelltextfile (3 Files sind abzugeben!). Benutzen Sie dabei die Methode `size` der Vektor-klasse. (1 Pkt.)
Erschließen Sie sich die Funktionsweise der Methoden `push_back`² und `pop_back` der Vektor-klasse. Probieren Sie diese Methoden aus.

Eingabedaten (n): (4), (20), (31553)

30. Schreiben Sie eine Klasse `DATUM`, welche Tag, Monat und Jahr speichert. Implementieren Sie (Header- und Sourcefile in `bsp_30_fkt.hpp` und `bsp_30_fkt.cpp`) alles so, daß das folgende Programm³ funktioniert. (3 Pkt.)

```
1  #include <iostream>
2  #include "bsp_30_fkt.hpp"
3  using namespace std;
4  \
5  int main()
6  {
7      Datum aa( 4, 12, 1983);
8      Datum bb(aa);
9      Datum cc(31,  5, 1983);
10     Datum dd;
11     \
12     dd = cc;
13     cout << "dd : " << dd << endl;
14     if ( aa < dd ) // aa frueheres Datum als dd?
15         { cout << aa << " frueher als " << dd << endl; }
16     else
17         { cout << aa << " nicht frueher als " << dd << endl; }
18     \
19     if ( bb == aa )
20         { cout << "aa und bb sind gleich" << endl; }
21     \
22     return 0;
23 }
```

¹<http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/Script/html/node26.html>

²http://www.cplusplus.com/reference/stl/vector/push_back

³http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/SS09/uebung8/bsp_30.cpp

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.

31. Implementieren Sie eine Klasse für die Koeffizienten eines Polynoms vom Grad n . Diese Klasse sollte neben Standard-, Kopierkonstruktor und Destruktor einen Parameterkonstruktor mit einem Filenamens als Parameter enthalten. Letzterer liest die Daten von einem File ein, siehe (§ 9.2 des Skriptes⁴). Weiters sollten Zuweisungs- und Ausgabeoperator implementiert werden. Die Auswertung des Polynoms (4 Pkt.)

$$p_n(x) = \sum_{k=0}^n a_k \cdot x^k$$

in einem Punkt x ist als Methode der Klasse zu realisieren. Werten Sie das Polynom in einer zweiten Methode über das HORNER-Schema aus.

Speichern Sie Klassendeklaration und -implementierung in den Files *bsp_31_fkt.hpp* und *bsp_31_fkt.cpp*.

Testdaten (Filename, x): (data_31_a.txt⁵, 3.78), (data_31_b.txt⁶, -1.0087), (data_31_c.txt⁷, -0.945),

Die ASCII-(Text-)Files enthalten n und a_k , $k = 0, \dots, n$.

32. Erweitern Sie Ihre Klasse aus Aufg. 31 so, dass (2 Pkt.)

- die Klassenmember nur innerhalb der Methoden der Klasse sichtbar sind,
- die Member über (Inline-)Methoden gelesen werden können,
- alle Methoden, welche die Member nicht verändern, als solche gekennzeichnet werden.

Schreiben Sie ein kleines Testprogramm dafür.

33. Implementieren Sie eine Klasse Sparschwein, welche folgende Datenelemente besitzt:

- Zähler für vier Arten von Geldstücken (1-Cent, 10-Cent, 50-Cent, 1-Euro)
- Die maximale Anzahl von Geldstücken die ins Sparschwein passen.
- Ein Flag, welches anzeigt, ob das Sparschwein aufgebrochen wurde.

(3 Pkt.)

Die Klasse soll die folgenden Methoden besitzen:

- `init(...)` initialisiert ein leeres Sparschwein,

⁴<http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/Script/html/node91.html>

⁵http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/SS09/uebung8/data_31_a.txt

⁶http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/SS09/uebung8/data_31_b.txt

⁷http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/SS09/uebung8/data_31_c.txt

- `add1Cents(...)` wirft eine Anzahl 1-Cent-Münzen hinein,
- `add10Cents(...)`, `add50Cents(...)`, `add1Euro(...)` wie oben,
- `isEmpty(...)` überprüft, ob das Sparschwein leer ist,
- `isFull(...)` überprüft, ob das Sparschwein voll ist,
- `isBroken(...)` überprüft, ob das Sparschwein geknackt wurde,
- `breakInto(...)` bricht das Sparschwein auf, und liefert die gesparte Geldsumme zurück.

Benutzen Sie Header- und Sourcefile zur Implementierung der Klasse. Das Hauptprogramm (ein weiteres Sourcefile) soll Münzen solange in das Sparschwein einwerfen, bis dieses voll ist und es dann knacken. Zeigen Sie den angesammelten Betrag an. Werfen Sie danach noch 37 Cent in das Sparschwein und knacken Sie es erneut (Wieviel Geld ist drin?).

34. Nutzen Sie für diese Aufgabe die Klassen `string` und `vector` aus der STL. Schauen Sie sich dazu §2.3.1 und §2.3.4 des Vorlesungsskriptes (bzw. `string`⁸ und `vector`⁹) an. Zu `vector` können Sie sich auch ein Bsp. in §3.3.1 des Download-Buches von Breymann¹⁰ und zu `string` die Vorlesung von Schröder¹¹ anschauen. (4 Pkt.)

Deklarieren Sie eine Klasse zur Darstellung eines Handys, das durch einen Gerätenamen, den Namen des Besitzers, die Telefonnummer und die monatliche Grundgebühr charakterisiert ist. Diese Klasse soll außer den üblichen Konstruktoren (falls diese nötig sind) Zugriffsmethoden auf die Member enthalten und die gesamten Daten zum Handy ausgeben können.

Nutzen Sie diese Handy-Klasse, um sich ein Adressbuch Ihrer Freunde anzulegen (Sie sollten mehr als 5 haben). Wie wäre es mit einer Klasse Adressbuch? Lesen Sie die unsortierten Datensätze von einem File ein und nutzen Sie evtl. die Funktion `getline`¹² zum Einlesen ganzer Zeilen inklusive der Leerzeichen. Geben Sie Ihr sortiertes Adressbuch aus (Tip: in den oben gegebenen www-Referenzen nach einem Algorithmus `sort` suchen).

Speichern Sie Klassendeklaration und -implementierung in den Files *bsp_34_fkt.hpp* und *bsp_34_fkt.cpp*. Geben Sie auch Ihr(e) Datenfile(s) *bsp_34_data.txt* ab.

generelle Hinweise:

Die Befehle `break`, `continue`, `goto` sind **nicht erlaubt**.

⁸<http://www.cplusplus.com/reference/string/>

⁹<http://www.cplusplus.com/reference/stl/vector/>

¹⁰<http://www.informatik.hs-bremen.de/~brey/stlb.html>

¹¹http://www.cpp-tutor.de/cpp/1e06/1e06_02.htm

¹²http://www.cpp-tutor.de/cpp/1e06/1e06_02.htm

Abgabe der Lösungen:

Die Abgabe der Lösungen (*.cpp-Files, *.hpp, ..., **keine** exe-Files) erfolgt ausschließlich durch das Kopieren dieser Files in Ihr (beim erstmal anzulegendes) Verzeichnis `Nachname_Vorname` im globalen Abgabeordner `y: \\pers.ad.uni-graz.at\fs\ou\621\stud_haase\Nachname_Vorname` .

Hinweise hierzu sind unter der LV-Homepage¹³ zu finden.

Die Filenamen **müssen** dem Schema `bsp_nummer`, gefolgt von der Fileextension, entsprechen. Z.B. ist in Beispiel 1 das File `bsp_1.cpp` abzugeben. Andere Filebezeichner zählen als nicht abgegebene Files.

¹³<http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/SS09/index.html>