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 07. Januar 2011

Übungen zur Vorlesung EINI (WS 2010)

Blatt 10

Block blau

Es können 4 Punkte erreicht werden.

Abgabedatum: 16. Januar 2011 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_10_0.txt*, *Aufgabe_10_1.cpp* und *Aufgabe_10_1.txt* zu erstellen.
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)

Die Komplexität eines Programmes steigt solange, bis die Fähigkeiten des Programmierer nicht mehr ausreichen, es zu warten.

Aufgaben

Aufgabe 0: Grundlagen (1 Punkt)

Legen Sie für Ihre Antworten eine Text-Datei *Aufgabe_10_0.txt* an.

a) Zählen sie mindestens drei Eigenschaften abstrakter Datentypen auf. (0,6 Punkte)

b) Was versteht man unter „Information Hiding“? (0,1 Punkt)

c) Mit welcher Auswahlstrategie arbeitet ein Keller? Nennen Sie ein Beispiel. (0,1 Punkt)

d) Wie ist die Höhe eines (binären) Baumes definiert? (0,1 Punkt)

e) Ist der leere Binärbaum ein Binärbaum? (0,1 Punkt)

Speichern Sie Ihre Ergebnisse in der Ergebnisdatei *Aufgabe_10_0.txt*.

Aufgabe 1: ADT Binärer Suchbaum (3 Punkte)

In der Vorlesung haben Sie binäre Suchbäume als effiziente Suchstruktur für Elemente kennengelernt.

a) Legen Sie eine cpp-Datei *Aufgabe_10_1.cpp* an. Implementieren Sie einen ADT `Binärer Suchbaum` - wie auf den Vorlesungsfolien Kapitel 9 (0,5 Punkte):

`class BinBaum`,

b) Implementieren, testen und kommentieren Sie eine Methode (1 Punkt):

`void einfuegen (T info)`,

die die Nutzinfo `info` an einer korrekten Stelle in den binären Suchbaum einfügt und eine Methode

`void drucken ()`,

die die Nutzinfos `info` sortiert auf dem Bildschirm ausgibt.

c) Implementieren, testen und kommentieren Sie eine Methode (1 Punkt):

`void BaumNachListe (Liste* m_Liste)`,

die einen binären Suchbaum in eine sortierte verkettete Liste umwandelt. Sie können dafür den in der Vorlesung vorgestellten ADT `Liste` verwenden. Erweitern Sie den ADT `Liste` um eine Methode `void drucken()`, die die Liste auf dem Bildschirm ausgibt.

d) Legen Sie eine Datei *Aufgabe_10_1.txt* an und beschreiben Sie, welche Probleme bei einer entsprechenden Umwandlung einer sortierten Liste in einen binären Suchbaum entstehen könnten? Wie könnten diese Probleme vermieden werden? Verdeutlichen Sie ihre Aussagen anhand von Beispielen (0,5 Punkte).

Hinweis: Verwenden Sie zum Testen Ihrer Implementierungen folgende `main`-Funktion:

```
1  int main()
2  {
3    BinBaum Baum;
4    Baum.einfuegen (100);//Aufgabe_10_1a
5    Baum.einfuegen (100);
6    Baum.einfuegen (23);
7    Baum.einfuegen (44);
8    Baum.einfuegen (1);
9    Baum.einfuegen (60);
10   Baum.einfuegen (-9);
11   Baum.einfuegen (8);
12   Baum.einfuegen (3);
13   Baum.drucken();
14
15   Liste *SortierteListe = new Liste;
16   Baum.BaumNachListe (SortierteListe);//Aufgabe_10_1b
17   SortierteListe -> drucken ();//Aufgabe_10_1c
18
19   return 0;
20 }
```

Kopieren Sie die Ergebnisse als Block-Kommentar an das Ende der cpp-Datei *Aufgabe_10_1.cpp*.

(Präsenz-)Aufgabe 2: Klassen(hierarchie) (0 Punkte)

Hinweis: Bei dieser Aufgabe handelt es sich um eine leicht modifizierte Prüfungsaufgabe aus dem Jahr 2008.
In dieser Aufgabe geht es um die Modellierung der neu an der Technischen Universität Dortmund eingeführten `Unicard`.
Hinweis: Lesen Sie sich diese Aufgabe erst ganz durch. Entwickeln Sie nur die Programmkonstrukte, die in der Aufgabenstellung gefordert werden. Sie müssen nicht jede Teilaufgabe für sich lösen, sondern können mehrere Teilaufgaben zusammenfassen. Die jeweilige Aufteilung in vier, bzw. sechs Teilaufgaben dient nur der Strukturierung dieser Aufgabe. Die folgenden Präprozessoranweisungen können Sie voraussetzen:

```
#include <iostream>
#include <string>
using namespace std;
```

I Das Modell der `Unicard` für Studierende soll die Attribute `TUDOID` vom Typ `unsigned int` sowie `Vorname`, `Nachname` beide vom Type `string` und `Semester` vom Typ `unsigned int`

haben.

(a) Entwerfen Sie eine Klasse `Unicard` und geben Sie hierzu die Klassendeklarationen mit `private`-Attributen in C++ an.

(b) Geben Sie für die Klasse `Unicard` einen Konstruktor an, mit dem allen Attributen Werte zugewiesen werden können.

(c) Geben Sie den C++-Code für eine Methode `Info` an, die in der Klasse `Unicard` formuliert ist und die Werte aller Attribute wohlformatiert auf dem Bildschirm ausgibt.

(d) Schreiben Sie ein Hauptprogramm, das ein Feld namens `TU` von drei Zeigern auf die Klasse `Unicard` anlegt. Erzeugen Sie je eine dynamische Instanz der Klasse `Unicard` und weisen Sie deren Adressen den Zeigern im Feld `TU` zu. Belegen Sie die Attribute mit Werten Ihrer Wahl. Anschließend laufen Sie mit einer `for`-Schleife über das Feld und nutzen die Methode `Info` zur Ausgabe. Was wird ausgegeben?

II Das Modell einer hierarchischen `Unicard` soll die Attribute `TUDOID` vom Typ `unsigned int` sowie `Name` vom Type `string` haben. Die Unicard für Bedienstete `Unicard_bed` soll zusätzlich das Attribut `Titel` vom Type `string` haben. Die Unicard für Studierende `Unicard_stud` soll zusätzlich die Attribute `Matrikelnummer` und `Semester` beide vom Typ `unsigned int` haben.

(a) Entwerfen Sie die Klassen `Unicard`, `Unicard_bed` und `Unicard_stud`, wobei Sie bei der Formulierung der Klassenhierarchie von der Vererbung Gebrauch machen sollen. Geben Sie hierzu die Klassendeklarationen mit `private`-Attributen in C++ an.

(b) Geben Sie für die Klasse `Unicard` einen Konstruktor an, mit dem den Attributen `TUDOID` und `Name` Werte zugewiesen werden.

(c) Geben Sie für die Klasse `Unicard_bed` einen Konstruktor an, mit dem auch dem Attribut `Titel` ein Wert zugewiesen wird. Geben Sie für die Klasse `Unicard_stud` einen Konstruktor an, mit dem auch den Attributen `Matrikelnummer` und `Semester` Werte zugewiesen werden.

(d) Geben Sie den C++-Code für eine virtuelle Methode `Info` an, die in der Klasse `Unicard` formuliert ist und die Werte der Attribute `TUDOID` und `Name` auf dem Bildschirm ausgibt.

(e) Die Methode `Info` soll nun in der Klasse `Unicard_bed` verfeinert werden. Erweitern Sie die Ausgabe um das zusätzliche Attribut der erbenden Unterklasse. Die Attribute `TUDOID` und `Name` sollen jedoch weiterhin durch die virtuelle Methode der Klasse `Unicard` ausgegeben werden.

(f) Schreiben Sie ein Hauptprogramm, das ein Feld namens `TU` von zwei Zeigern auf die Klasse `Unicard` anlegt. Erzeugen Sie je eine dynamische Instanz der Klassen `Unicard_stud` sowie `Unicard_bed` und weisen Sie deren Adressen den Zeigern im Feld `TU` zu. Belegen Sie die Attribute mit Werten Ihrer Wahl. Anschließend laufen Sie mit einer `for`-Schleife über das Feld und nutzen die Methode `Info` zur Ausgabe. Was wird ausgegeben?