Fachhochschule Ingolstadt WS 2002/2003
Studiengang Informatik

Prüfung Grundlagen der Informatik 2

Aufgabensteller: Dr. B.Glavina, Dr. S.Hahndel, Dr. J.Schweiger, Dr. M.Rudolph
Prüfungsdauer: 90 Minuten
Hilfsmittel: eigenhändig geschriebene Notizen (max. vier Seiten DIN A4)

Aufgabe 1 (Listen in C; etwa 30%)

Im Rahmen eines Softwareprojektes sollen Sie eine Datenstruktur für Dosenpfand als verkettete Liste implementieren. Ein Listenelement soll dabei die folgenden Komponenten enthalten:

- Zeiger auf eine Zeichenkette für den Getränkehersteller
- Eine Zeichenkette der Länge 40 für die Bezeichnung des Produktes.
- Eintrag für den Getränkepreis als float für die Verpackungseinheit
- Eintrag für den Pfand als float für eine einzelne Dose in der Verpackungseinheit
- Anzahl der Dosen in dieser Verpackungseinheit

a) Geben Sie eine Typdefinition „pfandlist“ an, die die oben beschriebene Liste realisiert.

b) Schreiben Sie eine Funktion pfandsumme, die einen Zeiger auf das erste Listenelement übergeben bekommt und die Gesamtsumme des in der Liste enthaltenen Dosenpfands zurück gibt. Berücksichtigen Sie dabei auch die Anzahl der Dosen in einer Verpackungseinheit

c) Schreiben Sie eine C-Funktion delete_hersteller, die alle Einträge eines bestimmten Herstellers aus der übergebenen Liste löscht und als Rückgabewert die Anzahl der gefundenen Elemente zurückgibt, also durch folgenden Prototyp beschrieben wird:

```
unsigned int delete_hersteller(char *hersteller, pfandlist **liste);
```

d) Was müssen Sie machen, um eine doppelt verkettete Liste zu erhalten ? Skizzieren Sie kurz anhand eine doppelt verkettete Liste mit drei Elementen.

Aufgabe 2: (ca. 30%)

In dieser Aufgabe soll in C++ ein vereinfachtes Leitsystem zur Beschickung von 3 Montagelinien mit Autos programmiert werden. Autos können Kombis oder Limousinen sein. Jede Montagelinie ist aus 10 Montagezellen aufgebaut. Jedes Auto wird in eine der Montagelinien eingelastet, deren Montagezellen im Mittel die niedrigste Auslastung aufweisen.

a) Implementieren Sie dazu eine Klasse MontageZelle. Sie besitzt zwei Gleitkommazahlen für die aktuell gültige untere und obere Lastgrenze. Die Lastgrenzen sind außerhalb der Klasse nicht sichtbar und werden im Konstruktor der Klasse mit den Werten 0.5 und 1.5 initialisiert.

Die Klasse enthält weiterhin eine Methode berechneProzentLast, die die prozentuale Auslastung für ein Auto in der Montagezelle berechnet. Dazu erhält die Methode als Eingabeparameter die tatsächliche Last, die das Auto in der Zelle hervorruft, als Gleitkommazahl. Die Methode enthält einen booleschen Ausgabeparameter, der anzeigt, ob die tatsächliche Last innerhalb der unteren und der oberen Lastgrenze liegt. Falls ja, belegt die Methode einen weiteren Ausgabeparameter mit der prozentualen Auslastung gemäß folgender Formel:

$$\text{Prozentuale Last} = \frac{\text{Tatsächliche Last} - \text{Untere Lastgrenze}}{\text{Obere Lastgrenze} - \text{Untere Lastgrenze}} * 100$$

Schließlich enthält die Klasse eine Methode einlastenZelle, die einen tatsächlichen Lastwert für ein Auto als Parameter erhält und die Lastgrenzen der Zelle neu berechnet. Sie ist außerhalb der Klasse verfügbar und liefert kein Ergebnis. Der Rumpf der Methode einlastenZelle muß nicht implementiert werden.

b) Vereinbaren Sie eine weitgehend redundanzfreie Klassenhierarchie für die Autos. Jedes Auto besitzt eine Kennummer, die außerhalb der Klasse nicht sichtbar ist und im Konstruktor belegt wird. Jedes Auto enthält zudem ein Feld von tatsächlichen Lastwerten für jede der 10 Montagezellen einer Montagelinie. Das Feld ist außerhalb der Klasse verfügbar. Schließlich besitzt jedes Auto eine Methode setzeLast, die den tatsächlichen Lastwert des Autos für eine Zelle neu belegt. Dazu erhält die Methode den Index der Zelle im Feld und den neuen Lastwert als Parameter. Die Methode liefert einen booleschen Wert zurück, der anzeigt, ob die Belegung möglich war.

Kombis und Limousinen besitzen eigene Konstruktoren, in denen das Feld mit speziellen Lastwerten vorbesetzt wird. Der Konstruktor für die Limousinen belegt die Lastwerte aller Zellen mit 1 vor. Der Konstruktor für die Kombis belegt alle Zellen mit ungeradem Index mit 1 und alle anderen mit 1.2 vor.

c) Implementieren Sie eine Klasse MontageLinie. Sie speichert ihre 10 Montagezellen in einem Feld, das außerhalb der Klasse nicht verfügbar ist. Das Feld besteht aus Zeigern, die im Konstruktor belegt werden. Die Klasse besitzt ein allgemein zugängliches Attribut, das anzeigt, ob die Montagelinie gesperrt ist. Jede Montagelinie ist zunächst nicht gesperrt.

Die Klasse besitzt eine allgemein verfügbare Methode berechneMittlereLast, an die ein Auto als Parameter übergeben wird. Sie berechnet, falls die Lastwerte des Autos für alle Montagezellen zulässig sind, den Mittelwert der prozentualen Last aller Montagezellen der Linie. Über einen booleschen Resultatsparameter wird angezeigt, ob die Lastwerte des Autos für alle Zellen der Linie zulässig waren.

Schließlich enthält die Klasse eine Methode einlastenLinie. Auf sie kann außerhalb der Klasse zugegriffen werden. Sie erhält ein Auto als Parameter und liefert kein Ergebnis. Der Rumpf der Methode einlastenLinie muß nicht implementiert werden.

d) Implementieren Sie eine Klasse Leitsystem. Sie beinhaltet die 3 Montagelinien als Feld, das außerhalb der Klasse nicht sichtbar ist. Die Methode montiereAuto der Klasse ist außerhalb der Klasse sichtbar und erhält ein Auto als Parameter. Sie lastet dieses Auto in eine der nicht gesperrten Montagelinien ein, die die kleinste mittlere Auslastung besitzen. Über ein boolesches Resultat zeigt die Methode an, ob das Auto in eine Montagelinie eingelastet werden konnte.

e) Schreiben Sie ein Hauptprogramm, in dem ein Leitsystem, ein Kombi und eine Limousine erzeugt werden. Die Kennummern der Autos sind frei wählbar. Die beiden Autos werden nacheinander an das Leitsystem zur Montage übergeben. Falls ein Auto nicht montiert werden konnte, wird eine Meldung ausgegeben und die Montage abgebrochen.

Aufgabe 3 (Klassen und Polymorphie; etwa 30%)

Es sei das folgende Programm gegeben:

```
#include <iostream.h>
class Vehikel
{
        int Raeder;
        float Gewicht;
public:
        void Nachricht(void) {cout << "Nachricht vom Vehikel\n";}
 };

class Auto : public Vehikel
{
        int Passagieranzahl;
public:
        void Nachricht(void) {cout << "Nachricht vom Auto\n";}
};

class Laster : public Vehikel
{
        int Passagieranzahl;
        float Ladung;
public:
        int Passagiere(void) {return Passagieranzahl;}
};

class Boot : public Vehikel
{
        int Passagieranzahl;
public:
        int Passagiere(void) {return Passagieranzahl;}
        void Nachricht(void) {cout << "Nachricht vom Boot\n";}
};

int main()
{
Vehikel Hochrad;                        //***
Auto Sedan;                             //***
Laster Sattelschlepper;                 //***
Boot Segelboot;                         //***

        Hochrad.Nachricht();
        Sedan.Nachricht();
        Sattelschlepper.Nachricht();
        Segelboot.Nachricht();

        return 0;
}
// Ergebnis beim Ausführen
//
// Nachricht vom Vehikel
// Nachricht vom Auto
// Nachricht vom Vehikel
// Nachricht vom Boot
```

Im Programm wird die Basisklasse Vehikel definiert, von der die Klassen Auto, Laster und Boot abgeleitet werden.

a) Zeichnen Sie ein UML-Diagramm mit allen Attributen und Methoden.

b) Geben Sie die Ausgabe des Programms bei der Ausführung an.

c) In den mit „//***“ markierten Programmzeilen werden die Objekte Hochrad, Sedan, Sattelschlepper und Segelboot angelegt. Ersetzen Sie diese Deklarationen durch

   vehikel *ptrvehicle;

   Ändern Sie das gegebene Programm so ab, dass Sie mit Hilfe der Polymorphie und dynamischer Speicherallokierung eine identische Ausgabe erzeugen.

d) Vergleichen Sie die beiden Programme. Diskutieren ganz kurz die Vor- und Nachteile beider Lösungen.

Aufgabe 4 (Zahlendarstellung; etwa 10%)

Nach dem Standard IEEE-754 werden Gleitkommazahlen einfacher Genauigkeit mit Bias 127, 8 Bit als Biased Exponent und 23 Bit als Fraction dargestellt. Gegeben seien die Zahlen A = $222{,}65625_{10}$, B = $0{,}5625_{10}$

a) Wie lauten die Zahlen A und B im Dualsystem?
b) Wie werden A und B gemäß diesem Standard im Rechner gespeichert?

(Ende des Aufgabentextes)