Fachhochschule Ingolstadt

Studiengang Elektro- und Informationstechnik
Sommersemester 2001

Prüfungsfach:	Grundlagen Informatik
Prüfungsabschnitt:	2. Teilprüfung
Termin:	20.07.2001
Zeit:	11.00 Uhr
Prüfer:	Prof. Dr. B. Glavina, Prof. Dr. J. Schweiger
Prüfungsdauer:	90 Minuten
Hilfsmittel:	4 DIN-A4 Seiten Zusammenfassung Vorlesungsskript

Aufgabe 1: (ca. 25%)

Mit einer einfach verketteten Liste sollen in einem C-Programm Produktionsaufträge für Maschinen verwaltet werden. Jeder Auftrag besteht aus einer natürlichen Zahl als Auftragsnummer und einer statischen Zeichenkette mit bis zu 100 Zeichen für die Bezeichnung der Maschine.

a) Vereinbaren Sie einen Datentyp *Auftrag* mit den oben genannten Bestandteilen als Element einer einfach verketteten Liste.

b) Schreiben Sie eine Prozedur *Last,* die in einer Auftragsliste den Auftrag mit einer bestimmten Nummer für eine bestimmte Maschine ans Ende der Liste setzt. Die Auftragsliste wird als transienter Parameter übergeben. Die Nummer und die Bezeichnung der Maschine, die ans Ende gesetzt werden soll, werden als Eingabeparameter an die Prozedur übergeben. Ist kein entsprechender Auftrag vorhanden, bleibt die Liste unverändert. Kommentieren Sie kurz jede Anweisung in der Prozedur.

Aufgabe 2: (ca. 25 %)

In einem Programm in C++ sollen verschiedene Agenten modelliert werden. Jeder Agent ist durch ein eigenes Objekt repräsentiert. Jeder Agent ist durch die beiden, außerhalb des Objekts nicht verfügbaren Attribute *AgentName* und *RechnerName* eindeutig gekennzeichnet. *AgentName* und *RechnerName* sind dynamische Zeichenketten.

AgentName kann durch die Methode *setAgentName()*, *RechnerName* durch die Methode *setRechnerName()* von außerhalb des Objekts belegt werden. Diese Methoden besitzen keinen Rückgabewert. Die allgemein verfügbare Methode *getIdentifikation()* liefert *AgentName* und *RechnerName,* durch das Zeichen ‘@‘ getrennt, zurück.

Als spezieller Agent ist im Programm einmal ein Planungsagent enthalten. Dieser besitzt ein Attribut *Plan*, das aus einem quadratischen Feld mit 100 Wahrheitswerten besteht und das nach außen nicht sichtbar ist.

Bitte wenden !!!!

Durch die Methode *plane()* wird der Plan mit Werten gefüllt. Die Methode *plane()* besitzt keinen Rückgabewert und keinen Parameter. Sie ist außerhalb des Planungsagenten verfügbar. Im Konstruktor des Planungsagenten wird ein Wahrheitswert übergeben, mit dem das Feld vorbelegt wird.

Als weitere spezielle Agenten sind ein Profibusagent und ein Interbusagent im Programm enthalten. Diese Agenten besitzen beide ein ganzzahliges Attribut *Quelle*, auf das von außerhalb des Objekts nicht direkt zugegriffen werden kann.

Der Profibusagent besitzt zudem eine Methode *sende()* mit ganzen Zahlen für die Quelle und das Ziel als Parameter. Das Ziel wird nicht im Agenten gespeichert. Die Methode *sende()* besitzt einen booleschen Rückgabewert und ist außerhalb des Planungsagenten verfügbar.

Der Interbusagent besitzt statt der Methode *sende()* die Methode *broadcast().* Diese enthält als Parameter nur einen Wert für die Quelle und liefert keinen Rückgabewert. Sie ist außerhalb des Objekts sichtbar.

a) Vereinbaren Sie eine Klassenhierarchie für den obigen Sachverhalt in C++ . Die Klassenhierarchie soll, soweit möglich, durch Vererbung redundanzfrei sein.

b) Programmieren Sie den Rumpf der Methode *getIdentifikation()* in C++. Die Rümpfe der restlichen Methoden müssen nicht programmiert werden.

Aufgabe 3: (ca. 40 %)

In dieser Aufgabe soll eine einfache Anlagensteuerung in C++ erstellt werden. Ein Sensorobjekt erkennt das Teil, das von der Anlage zu bearbeiten ist, und gibt dessen Kennummer an ein Aktorobjekt weiter. Die Weitergabe erfolgt über ein Pufferobjekt, das genau eine Kennummer speichern kann. Sensor- und Aktorobjekt werden vom Hauptprogramm zyklisch aktiviert.

a) Implementieren Sie eine Klasse für das Pufferobjekt. Eine Kennummer ist eine positive, ganze Zahl. Die Kennummer, die im Puffer gespeichert ist, ist außerhalb des Pufferobjekts nicht sichtbar.

   Durch die Methode *put()* wird eine Kennummer im Pufferobjekt gespeichert. Ist bereits eine Kennummer im Puffer vorhanden, wirft die Methode eine geeignete Exception aus und läßt die bereits gespeicherte Kennummer unverändert.

   Durch die Methode *get()* wird die Kennummer ausgelesen und gelöscht. Ist keine Kennummer im Puffer vorhanden, wirft die Methode eine Exception aus.

   Der Wert der Kennummer im Pufferobjekt ist 0, falls keine Kennummer im Puffer gespeichert ist. Im Konstruktor wird die Kennummer entsprechend vorbelegt.

Bitte nächstes Blatt !!!!

b) Implementieren Sie eine Klasse für das Sensorobjekt. Diese erhält im Konstruktor einen Zeiger auf das Pufferobjekt als Parameter und speichert ihn, nach außen nicht sichtbar, in der Klasse.

Das Sensorobjekt enthält eine Methode *recognize()*, die durch den Aufruf der internen Methode *runSensor()* die Kennummer bestimmt. Die Methode *runSensor()* liefert bei Vorhandensein eines Teils eine gültige Kennummer und ansonsten 0 zurück. Die erkannte Kennummer wird in das Pufferobjekt eingetragen. Ist dies nicht möglich, wird eine geeignete Meldung ausgegeben.

Der Rumpf der Methode *runSensor*() muß nicht implementiert werden.

c) Implementieren Sie eine Klasse für das Aktorobjekt. Sie erhält das Pufferobjekt analog zum Sensorobjekt.

Das Aktorobjekt enthält eine Methode *act(),* die die Kennummer aus dem Pufferobjekt ausliest. Ist im Pufferobjekt keine Kennummer vorhanden, so wird eine Meldung ausgegeben. Ist im Pufferobjekt eine Kennummer vorhanden, ruft die Methode *act()* die interne Methode *runAktor()* mit der Kennummer als Parameter auf.

Der Aktor ist als ideal anzunehmen, wodurch *runAktor()* keinen Fehler zurückmeldet. Der Rumpf der Methode *runAktor()* muß nicht implementiert werden.

d) Implementieren Sie das Hauptprogramm der Anlagensteuerung. Erzeugen Sie Instanzen für einen Sensor und einen Aktor, sowie ein Pufferobjekt, das Sensor und Aktor verbindet. Rufen Sie in einer Endlosschleife die Methoden *recognize()* und *act()* der Objekte auf.

Aufgabe 4: (ca. 10%)

Nach dem Standard IEEE-754 werden Gleitkommazahlen einfacher Genauigkeit mit 8 Bit als Biased Exponent (Bias = 127) und 23 Bit als Fraction (=Mantisse ohne Hidden Bit) dargestellt. Gegeben seien die Zahlen A = 43,75 und B = 0,015625.

a) Wie lauten A und B im Dualsystem (Festkomma-Darstellung)?

b) Wie werden A und B gemäß dem oben genannten IEEE-Standard im Rechner gespeichert?