Info2 Uebung vom 6.5.02

Gruppe F, Thomas Briner, http://n.ethz.ch/student/brinerth/info2

6. Mai 2002

1 Generische Module

ADT (Abstract Data Type):

Die Implementation ist für den Client nicht sichtbar. Bekannt ist nur die Schnittstelle (API Application Programming Interface)

Generisch:

Das Servermodul legt nicht schon fest, welche Inhalte in der Datenstruktur gespeichert werden. Die Funktionalität sind auf beliebigen Daten möglich.

1.1 Bsp Heap

Was könnte ein Benutzer mit einem Heap wollen?

- Adressen sortieren

- für Prozesse innerhalb eines Computers immer den "wichtigsten" ermitteln

1.1.1 Typendefinition

1.1.2 Prozedurvariablen

Das Servermodul kann bestimmte Prozeduren verlangen, die auf den ClientDaten bestimmte Operationen ausführen. Die Schnittstelle wird im Servermodul als Prozedurvariable definiert. Die vom Client implementierte Prozedur wird dann als Variableninhalt ans Servermodul mitgegeben (zB in der Prozedur init).

Wie sieht nun im Servermodul ein Vergleich zB für das SiftDown zwischen 2 Elementen aus?

1.1.3 Fehler im Servermodul

Das Servermodul darf keinesfalls Fehler zB ein pop auf einen leeren Stack als Bildschirmausgabe verarbeiten. Alle Prozeduren müssen also in irgendeiner Form zurückmelden, ob die Ausführung erfolgreich war. Mögliche solche Rückmeldungen:

- Boolsche Variable als Rückgabewert, eventuelle sonstige Rückgaben als VAR Parameter
- Globale Variablen, die angeben, ob bisher alles ok war (zB im In-Modul die Variable In.Done)
- Durch Rückgabe von Pointerobjekten. Der Client bemerkt dann einen allfälligen Fehler, indem er ein NIL-Objekt zurückerhält.

Prozeduren im Heap ServerModul:

1.1.4 Repetition Vererbung

```
TYPE Obj = RECORD;
     i:INTEGER;
     END;
     ObjErw = RECORD (Obj)
     r:REAL;
     END;
VAR  o:Obj; oe:ObjErw;
```

Was funktioniert, was nicht, was ist unnötig?

```
o:=oe;

oe:=o;

o:=oe; o.r:=3.5;

oe(Obj).i:=2;
```

1.1.5 Resultate aus Heap in Client auslesen

Im Server ist folgende Prozedur definiert:

```
PROCEDURE min*(VAR heap:Heap):HeapObj;
```

Wie können nun im ClientModul die vorher hineingespeicherten Adresskarteikarten herausgenommen und Name und Vorname ausgedruckt werden?

1.2 Regeln für die Arbeit mit generischen Modulen

1. Es soll nur sowenig wie möglich für den Client exportiert werden.
2. Die zu erweiternden Elemente und die Datenstruktur selber müssen für den Client sichtbar sein. Ebenso allfällige Prozedurvariablen.
3. Die Signatur für Prozedurvariablen muss im Clientmodul genau eingehalten werden.
4. In der konkreten Implementation einer solchen Prozedur muss mittels TypeGuard dem Compiler zugesichert werden, dass die Elemente die zusätzlichen Felder tatsächlich enthalten
5. Die Rückgaben aus dem Servermodul sind vom Grundtyp des Elements. Um auf die Felder des erweitereten Typs zugreifen zu können, müssen die Variablen auf den erweiterten Typ gecastet werden.

1.3 Uebungsaufgabe

Schreib die Typendefinition und eine Vergleichsprozedur für ein ServerModul, das 3 Elemente vergleicht und das kleinste Element zurückgibt. Dir steht eine Prozedurvariable
kleiner:PROCEDURE (a,b:Entry):BOOLEAN
zur Verfügung, die der Client so implementiert.

Zusatz: Gib eine Implementation der Prozedur kleinereFläche im ClientModul für die folgende TypErweiterung:

```
TYPE    Rechteck = POINTER TO RechteckDesc;
        RechteckDesc = RECORD (H.HeapObjDesc);
                länge,breite: INTEGER;
        END;
```

2 Binäre Suchbäume

Gegeben ist folgende Typdefinition:

```
TYPE    Node* = POINTER TO NodeDesc;
                NodeDesc*= RECORD;
                        key:INTEGER;
                        left,right:Node;
                END;
```

2.1 Schreib eine Prozedur, die die Knoten in PostOrderReihenfolge in eine Lineare Liste einhängt

2.2 Interne Pfadlänge

Die interne Pfadlänge, wie sie in Aufg 5.2 benutzt wird, ist die Summe aller Pfadlängen für jeden Knoten innerhalb des Baumes bis zur Wurzel über alle Knoten aufsummiert.

2.3 Knoten löschen

Je nach Fall verschiedene Vorgehen:

- keine Kinder:
- genau ein Kind:
- zwei Kinder:

3 AVL-Bäume

So benannt nach Adelson-Velskij und Landis.

Zusätzlich zu den Eigenschaften eines binären Suchbaums kommt noch die AVL Eigenschaft dazu:

So kann garantiert werden, dass der Baum nicht zu linearen Liste degeneriert.

3.1 Einfügen in den AVL Baum

Füge in den AVL Baum die Knoten mit Schlüssel 4, 12, 20, 1, 3, 6, 18, 19, 17 ein. Notier bei jedem Knoten immer den aktuellen Balancefaktor.