Programmierkurs I (Inf., W-Info.), Übungsblatt 13

Claus/Weicker, Wintersemester 03/04 Abgabetermin: 05.02.2004, 23:59 Uhr

Bitte beachten Sie die Hinweise von Aufgabenblatt 8 sowie die Angaben in den jeweiligen Spezifikationsdateien $*$.ads.

Bitte denken Sie daran, dass für die Vergabe eines Scheins mindestens 40 Punkte aus den letzten 4 Aufgabenblättern benötigt werden! Das letzte Blatt wird Nummer 13 sein.

Aufgabe 1: Baumbart I (mittel) — 8 Punkte

In dieser Aufgabe sollen Sie eine Funktion schreiben, die für zwei Bäume feststellt, ob sie identisch sind. Dabei beschränken wir uns auf binäre Bäume, d.h. der Baum hat einen eindeutigen Wurzelknoten und jeder Knoten kann einen linken und einen rechten Unterbaum haben. Jeder Knoten erhält einen ganzzahligen Schlüssel. Die Identität von zwei Bäumen sei hier so definiert, dass die Schlüsselwerte der Wurzelknoten übereinstimmen und die Eigenschaft rekursiv für die rechten und linken Unterbäume ebenfalls gilt – insbesondere bedeutet dies auch, dass ein rechter Unterbaum am Knoten des einen Baums genau dann existiert, wenn dies auch am Knoten des anderen Baums gilt.
Die Schnittstelle zur Verarbeitung der Bäume ist in der Datei `baum.ads` enthalten. Hiervon sollten lediglich die Zugriffsfunktionen benutzt werden.
Geben Sie Ihre Implementation der Funktion

```
function Gleichheitstest (A, B: Baum) return Boolean;
```

in der Datei `gleichheit.adb` ab – die entsprechende Spezifikationsdatei kann ebenfalls von der Webseite geladen werden. Wie in obiger Definition können Sie in Ihrer Implementation die Funktion rekursiv nutzen.
Es gibt 5 Punkte für korrekte Funktionalität, 1.5 Punkt für die Beschreibung des Konzepts und 1.5 Punkt für die Einhaltung der Programmierrichtlinie.

Aufgabe 2: Baumbart II (mittel) — 8 Punkte

In dieser Aufgabe sollen die Schlüssel eines binären Baums (wie in obiger Aufgabe definiert) in Postorder ausgegeben werden, d.h. an jedem Knoten gilt, dass zunächst die Schlüssel des linken Unterbaums, dann die des rechten Unterbaums und zuletzt der Schlüssel des Knotens selbst ausgegeben werden.
Dies soll die Prozedur

```
procedure TraversierePostorder(T: in Baum; L: out Liste);
```

realisieren, wobei die Schlüssel in der Postorder-Reihenfolge in eine Instanz des Datentyps `Liste` eingefügt werden sollten. Der Datentyp `Liste` ist in der Datei `liste.ads` spezifiziert.
Geben Sie Ihre Implementation in der Datei `postorder.adb` ab – passend zur Spezifikation `postorder.ads` auf der Programmierkursseite.
Es gibt 5 Punkte für korrekte Funktionalität, 1.5 Punkte für die Beschreibung des Konzepts und 1.5 Punkte für die Einhaltung der Programmierrichtlinie.

Aufgabe 3: Baumbart III (leicht) — 4 (+ 3) Punkte

Um obige Programme sinnvoll testen zu können, sollten Sie die benötigten Datentypen Baum und Liste implementieren.

a) Beschreiben Sie ausführlich, welche Testfälle für Aufgabe 1 und 2 notwendig sind, um die Funktionalität zu gewährleisten. Hier erwarten wir die Eingaben, richtige Ausgaben und eine

Begründung, warum der Testfall relevant ist. Beschreiben Sie weiterhin, wie Sie diese Tests umgesetzt haben.

b) Zusatzaufgabe: Geben Sie Ihre vollständige Implementation von `baum.adb` ab. Beachten Sie dabei, dass Sie `baum.ads` nicht verändern, sondern den Datentyp `Baum_Element` in `baum.adb` vollständig deklarieren.

Hinweise

- Zur Abgabe wird das eClaus-System verwendet:
  `http://eclaus.informatik.uni-stuttgart.de`
- Die Abgabe zu jeder Teilaufgabe besteht aus einem kompilierbaren Ada-Quelldatei. In jeder Aufgabe wird ein Dateiname vorgegeben. Verwenden Sie diesen bitte für das Hauptprogramm. Auch sind in der Aufgabe Angaben zu Ein- und Ausgabetexten sowie zur Formatierung der Ausgabe enthalten. Bitte folgen Sie diesen Angaben so genau wie möglich.
- Beachten Sie beim Programmieren bitte die folgenden Hinweise („kleine Programmierrichtlinie“).
  - Halten Sie einzelne Bestandteile überschaubar, z.B. indem Sie nur eine Anweisung pro Zeile schreiben, sowie pro Zeile höchstens 80 Zeichen, höchstens 40 Zeilen pro Prozedur/Funktion, nicht mehr als 800 Zeilen pro Datei und höchstens 5 Parameter bei Prozeduren/Funktionen benutzen.
  - Bezeichner sollen selbsterklärend und maximal 15 Zeichen lang sein. Bezeichner enthalten nur Buchstaben, den Bindestrich oder den Unterstrich.
  - Durch Einrückung um 2 Zeichen soll die logische Gliederung eines Programms verdeutlicht werden. Beispielsweise wird der Anweisungsteil einer IF-Verzweigung eingerückt, während „`end if;`“ nicht mehr eingerückt wird. Auch auf der nächsten Zeile fortgesetzte Zeilen werden um 2 Zeichen eingerückt.
  - Zu Beginn des Programms muss in Kommentaren das Konzept und die Lösungsidee des Programms ausführlich erläutert werden.
  - Auch im Programmtext sind Kommentare einzufügen, um den Code zu erläutern.
- Beachten Sie, dass jede Abgabe individuell vom jeweiligen Studierenden erstellt werden muss. Werden von den Tutoren Plagiate erkannt, d.h. exakte oder leicht modifizierte Kopien, werden für die Aufgabe keine Punkte vergeben. Falls Sie die Lösung der Aufgaben vor der Abgabe in Gruppen besprechen, achten Sie darauf, dort nur das generelle Konzept abzuklären und die Programmierung jedem selbst zu überlassen.
- Bei Fragen wenden Sie sich bitte an Ihren Tutor oder an die Übungsleitung:
  `Karsten.Weicker@fmi.uni-stuttgart.de` oder Tel. 7816-337
- Weitere Veranstaltungshinweise einschließlich der Übungsblätter finden Sie unter:
  `http://www.informatik.uni-stuttgart.de/ifi/fk/lehre/ws03-04/ada95/`