Dr. S. Lewandowski **Einführung in die Informatik I (autip)** WS 2007/08

Übungsblatt 13

Ausgabe: 30.01. Abgabeschluss: Mittw., 06.02., 9:45 Uhr, `eClaus.informatik.uni-stuttgart.de`

Abgabe erfolgt ausschließlich elektronisch über `eClaus.informatik.uni-stuttgart.de` – versuchen Sie nach Möglichkeit die Abgabe nicht in der letzten Minute zu machen!

Von jedem Aufgabenblatt werden maximal 20 Punkte auf den Schein angerechnet.

---

1. (1+1.5+1.5+2+4+2+3 Punkte, mittel) **Binärbäume / Suchbäume:** Implementieren Sie die Prozeduren und Funktionen aus der Vorlesung, Sie können hier als `Element_Typ` den Typ `Integer` verwenden.

   - `function empty return BinBaum` – erzeugt einen leeren (Such-)Baum
   - `procedure insert(baum : in out Liste; elem : Element_Typ)` – fügt in einen Suchbaum das Element `elem` gemäß den Eigenschaften eines Suchbaums ein
   - `function is_in(elem : Element_Typ; baum : BinBaum) return Boolean` – liefert `true` oder `false` abhängig davon, ob `elem` im `baum` vorkommt
   - `function number_of_elements(baum : BinBaum) return Natural` – zählt die Elemente in der Liste

   Beachten Sie auch die Hinweise zu diesen Prozeduren und Funktionen im Skript.

   (4 Punkte) Schreiben Sie ein Hauptprogramm, das zunächst eine Zufallszahl z aus dem Bereich 5..20 ermittelt und dann z Zahlen (aus einem ausreichend großen Integer-Bereich) zufällig erzeugt und diese in einen Suchbaum einfügt. Geben Sie den Suchbaum dann jeweils in Pre-, Post- und Inorder aus. Eine dieser drei Ausgaben hat eine auffällige Eigenschaft. (2 Punkte) Begründen Sie diese (fügen Sie die Begründung als Kommentar in Ihrem Programm mit ein).

   (3 Punkte) Schreiben Sie auch eine Prozedur, die bestimmt, wieviel Vergleiche man höchstens benötigt, um mit der Funktion `is_in` festzustellen, ob ein Element im Suchbaum vorhanden ist oder nicht. **Hinweis:** Sie müssen dazu nicht für alle möglichen Eingaben die Anzahl der Vergleiche bestimmen (zumal dies so auch nicht durchführbar wäre) – überlegen Sie sich eine rekursive Lösung.

   Achten Sie unbedingt auf ausreichend Kommentare und Beschreibung der Algorithmen in Ihrem Ada-95-Code.

3. (4 Punkte, mittel) **Eine kleine Graph-Eigenschaft:** Ein Kommillitone bietet Ihnen eine Wette an: Bei der nächsten Vorlesung soll beobachtet werden, wieviel Studierende sich gegenseitig begrüßen (das Ergebnis der Beobachtung ist dann z.B.: Student A begrüßte 23 Studenten, Student B begrüßte 18 Studenten, usw. – wenn Student A Student B begrüßt, dann begrüßt Student B auch stets Student A). Wenn die Anzahl der Studierenden, die eine ungerade Anzahl von Studierenden begrüßt haben, ungerade ist, sollen Sie 100 Euro von ihm erhalten, wenn die Anzahl jedoch gerade ist, erhält er nur 10 Euro.

   Gehen Sie diese Wette ein? Begründen Sie Ihre Antwort! (**Hinweis:** Modellieren Sie das obige Szenario als Graphen)

2. (5 Punkte, schwer) **Ein Binärbaumalgorithmus:** Betrachten Sie folgenden Algorithmus (`BinBaum` ist ein `access` Typ auf einen Binärbaumknoten `Knoten`):

```
procedure wastutdas (v : BinBaum) is
  r : BinBaum := new Knoten;
  p : BinBaum := r;
  c : BinBaum := v;
  h : BinBaum;
begin
  r.left:=v;
  while c /= r loop
    if c /= null then
      Put (c.inhalt);
      h := c.links;
      c.links := c.rechts;
      c.rechts := p;
      p := c;
      c := h;
    else
      h := c;
      c := p;
      p := h;
    end if;
  end loop;
end wastutdas;
```

Was tut diese Prozedur (wie wird der Unterbaum von `v` verändert) und welche Ausgabe liefert sie (vergleichen Sie die Ausgabe mit anderen Ihnen bereits bekannten Baumalgorithmen)?

Hinweis: Eine dringende Empfehlung: Die Aufgabe ist an sich recht leicht – Sie müssen aber sehr genau aufpassen, wohin welcher Zeiger jeweils gerade zeigt – man vertut sich beim ständigen Umsetzten der Verweise sehr leicht.
Sie sollten das Programm unbedingt von Hand durchführen, um mit Zeigern vertraut zu werden – nehmen Sie (zunächst) Beispiele mit höchstens 3 oder 4 Knoten. Wenn Sie dann eine Idee haben, was das Programm tut, können Sie es immer noch abtippen und Ihre Vermutung an größeren Beispielen verifizieren.

4. (2 Punkte, leicht) ... und noch eine leichte Programmieraufgabe zum Abschluss ... **Das ist der Rest ...** Es gibt Zahlentripel (a,b,c), so dass folgende Eigenschaft gilt: a*b mod c = 1 und b*c mod a = 1 und c*a mod b = 1.

   Schreiben Sie ein Ada 95 Programm, das alle Zahlentripel (a,b,c) mit $a \leq b \leq c$ und $a,b,c \leq 100$ auf obige Eigenschaft überprüft.

   Zusatzaufgabe (sehr schwer, +5 Punkte, nur für Mathefreaks und Zahlenknobler): Beweisen Sie, dass es keine weiteren Zahlentripel mit dieser Eigenschaft als die oben gefundenen gibt.

Fragen können im Forum `www.autip.de/forum/viewforum.php?f=218` diskutiert werden. Weitere Informationen zur Vorlesung und Übung unter `www.fmi.uni-stuttgart.de/fk/lehre/ws07-08/autip1/`