Allgemeine Informatik II SoSe 2001

Prof. Dr. H. Neumann • Dr. K. Murmann • S. Geschwentner • Dr. F. Schwenker

5. Aufgabenblatt (bis zum 21.06.2001)

11. Aufgabe:

In der Vorlesung wurde ein Algorithmus vorgestellt, der mit Hilfe eines Stapels, einen in Infix-Notation vorliegenden arithmetischen Ausdruck in seine Postfix-Notation überführt (Skript Teil III). Wir betrachten im folgenden arithmetische Ausdrücke mit den Operatoren $\{*, +\}$ und der Punktrechnung-vor-Strichrechnung-Prioritätsregel.

1. Transformieren Sie den Ausdruck

   $$((a + b) * c) * ((x + y) * (e + f))$$

   nach diesem Verfahren in seine Postfix-Form.

   Für jeden Transformationsschritt geben Sie bitte die aktuell angewandte Regel, die Zeichenkette die noch an der Eingabe steht, die Zeichenkette die bereits an der Ausgabe steht, sowie den Inhalt des Stapels aus.

2. Implementieren Sie diesen Algorithmus in Modula-2. Sie können dabei vereinfachend davon ausgehen, dass in der Eingabezeichenkette keine Leerzeichen vorkommen und, dass die Namen der Operanden aus einem einzelnen Buchstaben bestehen.

   Das Programm soll zunächst den vollständigen Ausdruck (in Infix-Notation) als Zeichenkette einlesen und diese dann zeichenweise verarbeiten. Die Postfix-Form des Ausdrucks soll ebenfalls in einer Zeichenkette gespeichert werden. Die Größe der Eingabe- und Ausgabezeichenkette (vom Typ `ARRAY OF CHAR`) soll durch eine Konstantendefinition festgelegt werden. Realisieren Sie den Stapel durch eine verkettete Liste.

   Für jeden Transformationsschritt soll das Programm die o.g. Ausgaben machen (angewandte Regel, Eingabe-, Ausgabestring und Stapelinhalt).

   Vergleichen Sie die Ausgabe des Programms mit den Ergebnissen Ihrer Simulation aus Teil 1.

12. Aufgabe:

Schreiben Sie ein Modula-2-Programm zum Aufbau eines Analysebaumes aus der Postfix-Notation eines arithmetischen Ausdrucks (Skript Teil III). Verwenden Sie die folgenden Datenstrukturen für den Analysebaum:

```
TYPE
   TREE = POINTER TO NODE;
   NODE = RECORD
             info        : CHAR;
             left, right : TREE;
          END;
```

Implementieren Sie die folgenden Prozeduren und Funktionsprozeduren:

1. `PROCEDURE BuildTree() : TREE;`
   Funktionalität: Die Postfix-Form eines arithmetischen Ausdrucks soll zeichenweise eingelesen werden. Sie können davon ausgehen, dass innerhalb des Ausdrucks keine Leerzeichen vorkommen, die Namen der Operanden aus genau einem einzelnen Buchstaben bestehen und, als Operatoren nur + und $*$ vorkommen.

   Als `RETURN`-Wert soll ein Zeiger auf die Wurzel des Baumes zurückliefert werden.

2. `PROCEDURE PrintNodes(root : TREE);`
   Funktionalität: Analysebaum mit der Wurzel `root` wird nach der Postorder-Strategie traversiert, hierbei werden die `info`-Felder aller Knoten ausgedruckt.

3. `PROCEDURE PrintLeaves(root : TREE);`
   Funktionalität: Analysebaum mit der Wurzel `root` wird nach der Postorder-Strategie traversiert, hierbei werden die `info`-Felder aller Blätter ausgedruckt.

4. `PROCEDURE NumberOfNodes(root : TREE) : CARDINAL;`
   Funktionalität: Die Zahl der Knoten des Analysebaumes mit der Wurzel `root` wird als `RETURN`-Wert zurückliefert.

5. `PROCEDURE NumberOfLeaves(root : TREE) : CARDINAL;`
   Funktionalität: Die Zahl der Blätter des Analysebaumes mit der Wurzel `root` wird als `RETURN`-Wert zurückliefert.

Testen Sie die von Ihnen implementierten Prozeduren und Funktionsprozeduren mit den Postfix-Formen der folgenden arithmetischen Ausdrücke
$a + b$ $x + y * z$ $(x + y) * (e + f)$ $((a + b) * c) * ((x + y) * (e + f))$