Übungen zu
Grundlagen der Informatik für Studierende des Maschinenbaus
im Sommersemester 2001
(Dipl.-Inf. Jörg Barner)

Aufgabe 10: Rekursiver Abstieg
Schreiben Sie in `C` einen Parser nach dem Prinzip des rekursiven Abstiegs, der prüft, ob seine Eingabe korrekt gemäß folgender Grammatik ist.

```
statements  ::= <statement> <statements> | ϵ
statement   ::= <assign_stmt> | <block>
assign_stmt ::= <id> = <expression>;
block       ::= { <statements> }
id          ::= a | b | ...| z
integer     ::= 0 | 1 | ...| 9
```

Das Startsymbol ist `<statements>`. Leerzeichen und Zeilenwechsel sind zu ignorieren.

Lösung:
Die Grammatik definiert eine stark vereinfachte Programmiersprache, die etwas an `C` angelehnt ist. Sie erlaubt als Anweisungen nur einfachste Zuweisungen, als Variablennamen sind nur einzelne Buchstaben erlaubt. Möglich sind allerdings geschachtelte Gültigkeitsblöcke. (Programmlisting: siehe Extrablatt)

Aufgabe 11: Blockstruktur

a) Was versteht man unter *Blockstruktur*?

b) Erklären Sie die Begriffe *Gültigkeitsbereich*, *Zugriffsbereich* und *Existenzbereich*.

c) Geben Sie für das folgende Programm die Gültigkeits- und Zugriffsbereiche der verwendeten Variablen an:

```
BEGIN
     VAR a,c: INTEGER;
     PROCEDURE p( b: INTEGER )
          VAR c: INTEGER;
          c := b * 2;
          a := c + 1;
     END p;
     a := 2; c := 3;
    BEGIN
          VAR a,d: INTEGER;
```

```
11              d := c + 1;
12              BEGIN
13                  VAR c: INTEGER;
14                  p( d );
15              END
16              a := 5;
17          END
18  END
```

Lösung:

a) Unter einer *Blockstuktur* versteht man die Einteilung des Programms in relativ geschlossene Abschnitte (Blöcke), die Deklarationen, Anweisungen und wieder Blöcke enthalten (Blockschachtelung).

b) Der Gültigkeitsbereich einer Variablen beginnt mit ihrer Deklaration und endet mit dem Block, in dem sie deklariert wurde. Eine Untermenge davon ist der Zugriffsbereich: Auf ein Objekt kann nicht zugegriffen werden, wenn es von einem anderen Objekt gleichen Namens überdeckt wird, das in einem inneren Block deklariert wurde.

   Während Gültigkeits- und Zugriffsbereich durch die Struktur des Quelltextes definiert sind, bezieht sich der Existenzbereich dagegen auf die Lebensdauer des Objekts zur Laufzeit des Programms. Ein Objekt existiert z.B. während eines Prozeduraufrufs weiter, man kann aber nicht unbedingt darauf zugreifen (wenn sich die Prozedur außerhalb des Zugriffsbereichs befindet).

c)

Variable	Gültigkeitsber.	Zugriffsber.
a_1	3–18	3–9, 18
c_1	3–18	3, 8–12, 16–18
b_2	4–7	4–7
c_2	5–7	5–7
a_3	11–17	11–17
d_3	11–17	11–17
c_4	14–15	14–15

Aufgabe 12: Prozeduren

a) Nennen Sie einige Vorteile von Prozeduren gegenüber einfachen Blöcken.

b) Welche Parameterübergabemechanismen kennen Sie?

c) Warum ist es sinnvoll, unterschiedliche Übergabemechanismen zu verwenden? Geben Sie jeweils ein Beispiel an.

Lösung:

a) Prozeduren können beliebig oft und auch mit unterschiedlichen Parametern aufgerufen werden.

b) **call by value**: Berechnung des tatsächlichen Parameters und Übergabe dieses Wertes. Zuweisungen an Parameter innerhalb der Prozedur wirken sich nicht nach außen aus.

   call by reference: Übergabe eines Speicherplatzes, an dem sich der aktuelle Parameterwert befindet. Zuweisungen an den formalen Parameter wirken sich also auch auf den aktuellen Parameter aus.

   call by name: Jedes Auftreten des formalen Parameters wird textuell durch den tatsächlichen Parameter ersetzt.

c) *call by value*-Parameter können in der aufgerufenen Prozedur wie lokale Variablen verwendet werden. Andererseits kann sich der Aufrufer sicher sein, daß der aktuelle Parameter durch den Aufruf nicht verändert werden kann.

   call by reference wird meis verwendet, wenn in den Parametern auch Ergebnisse zurückgegeben werden sollen (Ein- und Ausgabeparameter), z.B. wenn eine Funktion mehr als einen Rückgabewert besitzt. Eine andere Anwendung ist zur Übergabe großer Objekte (z.B. Felder), denn bei *call by value* müßte das gesamte Objekt kopiert werden.

   call by name ist in Ausnahmefällen sinnvoll, wenn mehrfache Auswertung des aktuellen Parameters erwünscht ist. Ein Beispiel für diesen Mechanismus sind die Präprozessor-Makros von `C`, wenn sie Argumente besitzen.