Benutzerdefinierbare Kommentare, Zwischenräume und Literale in MOSTflexiPL

Christian Heinlein
Studiengang Informatik
Hochschule Aalen – Technik und Wirtschaft

Obwohl MOSTflexiPL (Modular, Statically Typed, Flexibly Extensible Programming Language, http://flexipl.info) inzwischen schon einige Jahre alt ist, bot die Sprache bisher noch keine direkte Möglichkeit, Programme zu kommentieren. (Man konnte sich mit Stringliteralen als Dummy-Ausdrücke behelfen.) Der Hauptgrund für diesen Mangel war die Tatsache, dass die Syntax von Kommentaren in einer Sprache wie MOSTflexiPL vom Benutzer individuell definierbar sein sollte und dass es für diesen Zweck bis jetzt noch kein geeignetes Sprachmittel gab.

In der neuesten Version des Compilers können Kommentare jetzt aber zum Beispiel wie folgt deklariert werden:

```
"/*" ... "*/";  /* Definition von Blockkommentaren wie in C. */
"//" ...;       // Definition von Zeilenkommentaren wie in C++.
print           // Ausgabe einer Summe:
  1 /* linker Operand */ + /* rechter Operand */ 2
```

Formal gibt es hierfür einen vordefinierten Infixoperator `•...•` sowie einen Postfixoperator `•...`, deren Name jeweils aus drei Punkten besteht und deren Operanden (hier durch fette Aufzählungspunkte symbolisiert) Zeichenfolgen in Anführungszeichen sein müssen.

Kommentare können prinzipiell nicht verschachtelt werden, d. h. sie enden beim ersten Auftreten der jeweiligen Endzeichenfolge. Aber durch die Verwendung unterschiedlicher Kommentararten lässt sich Verschachtelung faktisch doch leicht realisieren, zum Beispiel:

```
"(*" ... "*)";  (* Eine weitere Art von Blockkommentaren. *)
(* Auskommentierter Code
print 1 + 2 /* mit Kommentar */ *)
```

Da die Endzeichenfolge einer Kommentardeklaration (im Gegensatz zur Anfangszeichenfolge) auch leer sein darf, sind Zwischenraumzeichen wie z. B. Leer- und Tabulatorzeichen nun nichts anderes als Kommentare, deren Anfangszeichenfolge aus dem jeweiligen Zeichen besteht und deren Endzeichenfolge leer ist: Ein entsprechender „Kommentar“ beginnt somit immer mit dem jeweiligen Zeichen und endet beim ersten Auftreten der leeren Zeichenfolge, d. h. sofort danach. Zum Beispiel:

```
"_" ... "";     // Unterstriche können jetzt
print 1_+_2     // als Zwischenraum verwendet werden.
```

Literale mit benutzerdefinierter Syntax und Bedeutung können durch nullstellige Operatoren definiert werden, deren Name als regulärer Ausdruck interpretiert wird, zum Beispiel:

```
[s]
"[OL]+" : int
{ /* Interpretation der Zeichenfolge s als Dualzahl */ };
print LOLLO     // Ausgabe: 22
```

Das Muster `[OL]+` beschreibt nichtleere Folgen der Buchstaben `O` und `L`, die als Dualzahlen interpretiert werden sollen. Entsprechende Zeichenfolgen im Programm werden dann vom Compiler durch Anwendungen dieses Literal-Operators ersetzt, wobei der Parameter `s` (dessen Typ implizit `string` ist) die jeweilige Zeichenfolge (im Beispiel `LOLLO`) enthält, sodass die Implementierung des Operators diese prinzipiell beliebig interpretieren kann.