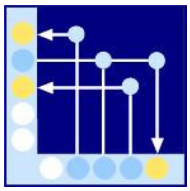
Fakultät Elektronik und Informatik
Studienbereich Informatik



Compilerbau-Praktikum

Lehrveranstaltung im Wintersemester 2022/2023
Prof. Dr. habil. Christian Heinlein

7. Aufgabenblatt (29. November 2022)

Aufgabe 7: Operatoren für Variablen und Ablaufsteuerung

Erweitern Sie die bis jetzt entwickelte Minisprache um folgenden Operatoren mit zugehörigen Auswertungsfunktionen:

a) Operator `uniq`, der bei jeder Anwendung einen neuen synthetischen Wert liefert.

b) Abfrage und Zuweisung von Variablen:

- Der rechtsassoziative Infixoperator `=!` mit Vorrang zwischen Konstantendeklaration und Disjunktion speichert den Wert seines zweiten Operanden in der Variablen, die durch den Wert seines ersten Operanden bezeichnet wird. Der Wert des ersten Operanden sollte hierfür normalerweise ein mittels `uniq` erzeugter synthetischer Wert sein. Der Wert des zweiten Operanden kann dann in einem zusätzlichen Attribut `varval` dieses synthetischen Werts gespeichert werden.
  Wenn der Wert des ersten Operanden kein synthetischer Wert (sondern ein natürlicher Wert oder nil) ist, ist die Zuweisung wirkungslos.
  Als Resultat wird in jedem Fall der Wert des zweiten Operanden geliefert.
- Der Präfixoperator `?` mit gleichem Vorrang wie Vorzeichenwechsel liefert den aktuellen Wert der Variablen, die durch den Wert seines Operanden bezeichnet wird.
  Wenn der Wert dieses Operanden kein synthetischer Wert ist oder wenn dieser Variablen noch kein Wert zugewiesen wurde, erhält man als Resultat den Wert nil.
- Zum Beispiel:

```
i := uniq;
i =! 1;
print ?i;          Ausgabe: 1
i =! ?i + 1;
print ?i           Ausgabe: 2
```

- Der rechtsassoziative Infixoperator `+!` mit gleichem Vorrang wie `=!` ist so definiert, dass `var +! val` gleichbedeutend mit `var =! ?var + val` ist, abgesehen davon, dass der Operand `var` nur einmal ausgewertet wird. Die Infixoperatoren `-!`, `*!`, `/!`, `^!`, `&!` und `|!` sind analog definiert.
- Der Postfixoperator `+!` mit gleichem Vorrang wie Fakultät ist so definiert, dass `var +!` gleichbedeutend mit `var +! 1` ist.
  Der Postfixoperator `-!` ist analog definiert.

c) Mehrfache Fallunterscheidung mit Signatur

```
if ... then ... { elseif ... then ... } [ else ... ] end
```

Als Ergebnis erhält man:

- entweder den Wert des ersten `then`-Teils, dessen zugehörige `if`- bzw. `elseif`-Bedingung erfüllt ist (d. h. einen echten Wert liefert)
- oder (wenn keine dieser Bedingungen erfüllt ist) den Wert des `else`-Teils
- oder (wenn es keinen `else`-Teil gibt) nil.

Die Auswertung wird abgebrochen, sobald dieses Ergebnis feststeht, d. h. weitere Operanden werden dann nicht mehr ausgewertet.

d) Sehr allgemeine Schleife mit Signatur

```
(while|until|do) ... { (while|until|do) ... } end
```

die bei jeder Iteration ihre Operanden nacheinander auswertet, bis die Auswertung eines `while`-Teils nil oder die Auswertung eines `until`-Teils einen echten Wert liefert. (Der Wert eines `do`-Teils ist irrelevant.)
Als Ergebnis wird die Anzahl der vollständig ausgeführten Iterationen geliefert.

Zum Beispiel (jede Zeile stellt eine einzige Anwendung des Schleifenoperators dar):

`while ... do ... end`	Kopfgesteuert mit Fortsetzungsbedingung
`do ... until ... end`	Fußgesteuert mit Abbruchbedingung
`do ... while ... do ... until ... do ... end`	Mehrere Forts.- und Abbr.-Bedingungen
`do ... end`	Endlosschleife

e) Erweitern Sie Ihr Testskript von Aufgabe 5c wie folgt, damit die Eingabedatei auch mehrzeilige Ausdrücke bzw. Programme enthalten kann:

- Wenn eine Zeile der Datei nur ein einziges Wort enthält, beschreibt dieses wie bisher das erwartete Ergebnis des Testfalls.
- Der zugehörige Ausdruck steht in dann in den nachfolgenden Zeilen bis zur nächsten Leerzeile (die nicht mehr zum Ausdruck gehört) bzw. bis zum Ende der Datei.

Zum Beispiel:

```
# Einzeiliger Testfall wie bisher.
3       1 + 2

# Mehrzeiliger Testfall.
3
x := uniq;
y := uniq;
x =! 1;
y =! 2;
?x + ?y

# Einzeiliger Testfall.
3       2 + 1
```