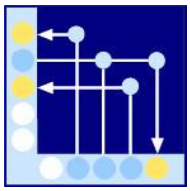

Programmieren in MOSTflexiPL

Vorlesung im Wintersemester 2025/2026
Prof. Dr. habil. Christian Heinlein

5. Aufgabenblatt (8. Dezember 2025)

Aufgabe 9: Lambda-Parameter

Teilaufgabe 9.a)

Ändern Sie die Operatoren für Konjunktion und Disjunktion aus Aufgabe 8.a dahingehend ab, dass der rechte Operand nur bei Bedarf ausgewertet wird, d. h. wenn sein Wert als Resultatwert geliefert wird.

Ändern Sie die verketteten Vergleichsoperatoren aus Aufgabe 7 und 8.b ebenfalls dahingehend ab, dass jeder Operand nur ausgewertet wird, wenn sein Wert zur Ermittlung des Resultatwerts benötigt wird, das heißt:

- Die ersten beiden Operanden werden immer ausgewertet.
- Jeder weitere Operand wird nur ausgewertet, wenn die Vergleiche aller vorhergehenden Operanden `true` geliefert haben.

Außerdem wird jeder Operand höchstens einmal ausgewertet, auch wenn sein Wert zweimal verwendet wird.

Teilaufgabe 9.b)

Definieren Sie einen Operator `if•then•{elseif•then•}[else•]end` für Fallunterscheidungen mit folgender Bedeutung:

- Die Bedingungen, d. h. die Operanden nach `if` und `elseif`, können jeweils beliebige (auch verschiedene) Typen besitzen.
- Alle Operanden nach `then` und ggf. `else` müssen jedoch den gleichen Typ besitzen, der ebenfalls beliebig sein kann und auch den Resultattyp der Fallunterscheidung darstellt.
- Zur Laufzeit werden der Reihe nach die Bedingungen ausgewertet, bis eine von ihnen nicht nil liefert, und dann der Wert des zugehörigen Operanden nach `then` geliefert.
- Wenn alle Bedingungen nil liefern, wird der Wert des Operanden nach `else` geliefert, falls vorhanden, andernfalls nil.
- Andere Operanden werden nicht ausgewertet.

Teilaufgabe 9.c)

Definieren Sie einen Operator `(while|until|do)•{(while|until|do)•}end` für Schleifen mit folgender Bedeutung:

- Alle Operanden können jeweils beliebige (auch verschiedene) Typen besitzen.
- Zur Laufzeit werden die Operanden immer wieder der Reihe nach ausgewertet, bis ein Operand nach `while` nil oder ein Operand nach `until` nicht nil liefert.
  Die Werte der Operanden nach `do` sind irrelevant.
- Als Resultatwert mit Typ `int` wird die Anzahl der vollständigen Schleifendurchläufe geliefert, d. h. die Anzahl der Durchläufe, bei denen jeweils alle Operanden ausgewertet wurden.

Zum Beispiel:

- Kopfgesteuerte Schleife mit Fortsetzungsbedingung:

  ```
  while ... do ... end
  ```

- Fußgesteuerte Schleife mit Abbruchbedingung:

  ```
  do ... until ... end
  ```

- Schleife mit mehreren Fortsetzungs- und Abbruchbedingungen an verschiedenen Stellen:

  ```
  do ... while ... do ... until ... do ... end
  ```

  Hier passiert in jedem Schleifendurchlauf folgendes:
  - Die ersten beiden Operanden werden ausgewertet.
  - Wenn der zweite nil liefert, wird die Schleife beendet.
  - Andernfalls werden die nächsten beiden Operanden ausgewertet.
  - Wenn der zweite von ihnen nicht nil liefert, wird die Schleife beendet.
  - Andernfalls wird der letzte Operand ausgewertet und der nächste Durchlauf begonnen.

- Endlosschleife:

  ```
  do ... end
  ```

Aufgabe 10: Fakultät und Potenz

a) Definieren Sie einen Postfix-Operator `•!` zur rekursiven Berechnung der Fakultät eines `int`-Werts. Wenn der Operand nicht größer oder gleich 0 (d. h. kleiner als 0 oder unnatürlich) ist, soll der Resultatwert nil sein.

b) Definieren Sie einen Infix-Operator `•^•` zur effizienten Berechnung von Potenzen, wie im Wikipedia-Artikel „Binäre Exponentiation“ beschrieben.
Beachten Sie sämtliche Sonderfälle wie z. B. negative Exponenten (für `k < 0` soll `x ^ k` äquivalent zu `1 : (x ^ -k)` sein) und unnatürliche Werte (dann soll der Resultatwert jeweils nil sein).
`x ^ 0` soll für alle natürlichen Werte von `x` einschließlich 0 gleich 1 sein.

c) Definieren Sie Vorrang und Assoziativität dieser Operator wie folgt:

- Der Potenzoperator ist rechtsassoziativ und bindet stärker als die Grundrechenarten einschließlich Vorzeichenwechsel. (Ein Vorzeichenwechsel als rechter Operand einer Potenz soll aber trotzdem möglich sein.)
- Der Fakultätsoperator bindet noch stärker als der Potenzoperator.