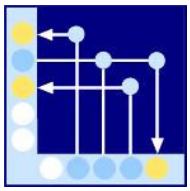
Hochschule Aalen
Fakultät Elektronik und Informatik
Studienbereich Informatik



Compilerbau-Praktikum

Lehrveranstaltung im Sommersemester 2021
Prof. Dr. habil. Christian Heinlein

3. Aufgabenblatt (26. April 2021)

Aufgabe 4: Parser des MOSTflexiPL-Compilers

Arbeiten Sie sich anhand der Dokumentation zumindest grob in die Struktur des MOSTflexiPL-Parsers ein!
Für diese Aufgabe ist insbesondere Abschnitt 3.1.5 relevant.

Die Arbeitsweise des Parsers kann durch Aktivierung der Ablaufverfolgung mittels `-D CH_TRACE` an einfachen Beispielen nachvollzogen werden.

Implementieren Sie dann durch Definition geeigneter Operatoren eine „Minisprache“, die momentan noch ungetypt ist und noch nicht ausgeführt werden kann, mit folgenden Bestandteilen:

- Ganzzahlige Konstanten
- Addition und Subtraktion (linksassoziativ)
- Multiplikation und Division (ebenfalls linksassoziativ) mit höherem Vorrang als Addition und Subtraktion
- Vorzeichenwechsel (Präfix-Minus) mit höherem Vorrang als Multiplikation und Division
- Potenz (rechtsassoziativ) mit noch höherem Vorrang, d. h. `- 2 ^ 3` bedeutet `- (2 ^ 3)` und nicht `(- 2) ^ 3`.
  Im rechten Operanden der Potenz können und sollen ungeklammerte Verwendungen des Vorzeichenwechsels aber möglich sein, z. B. `4 ^ - 5`.
- Fakultät (Postfix-Ausrufezeichen) mit nochmals höherem Vorrang,
  d. h. `2 ^ 3 !` bedeutet `2 ^ (3 !)` und nicht `(2 ^ 3) !`
- Klammern
- Ausgabe in der Form `print` *x* oder `print` *x* `width` *w*
  mit niedrigerem Vorrang als alle bisherigen Operatoren, d. h. `print 1 + 2` bedeutet `print (1 + 2)` und nicht `(print 1) + 2`. (`print` soll den Wert *x* nicht nur ausgeben sondern auch als Resultatwert zurückliefern.)
  Im rechten Operanden der vier Grundrechenarten sowie im Operanden des Vorzeichenwechsels können und sollen ungeklammerte Verwendungen von `print` aber möglich sein, z. B. `1 + print 2`.

  Was sollten dementsprechend die Ausdrücke `print 1 + 2 + 3`, `1 + print 2 + 3` und `1 + 2 + print 3` bedeuten, und wie kann man diese gewünschte Bedeutung mit Hilfe von Ausschlussangaben erreichen?
- Nacheinanderausführung (Infix-Semikolon) mit noch niedrigerem Vorrang.
  Zwischen `print` und `width` können und sollen ungeklammerte Verwendungen von Semikolon aber möglich sein. Zum Beispiel würde `print 1 + 2; 3 * 4 width 5; 6 + 7` den Wert 12 des Teilausdrucks `1 + 2; 3 * 4` mit Breite 5 ausgeben und anschließend die Werte 6 und 7 addieren.
- Nichtassoziative Vergleiche (<, <=, =, =/, >=, >) mit Vorrang zwischen Addition/Subtraktion und `print`.
  (Nichtassoziativ bedeutet, dass Ausdrücke wie z. B. `1 < 2 < 3` fehlerhaft sind.)

Verallgemeinern Sie die Vergleichsoperatoren dann dahingehend, dass beliebig „verkettete“ Vergleiche wie z. B. `1 < 2 < 3` oder auch `1 < 2 > 3 =/ 4` unterstützt werden!
Wie lauten sinnvolle Ausschlussangaben für diese Operatoren, die einerseits möglichst viele sinnvolle Ausdrücke erlauben und andererseits Mehrdeutigkeiten vermeiden?
Hinweis: Ein Parameter kann mehrere Ausschlussangaben besitzen, die für unterschiedliche Arten von Operanden gelten (Attribute `front`, `middle` und/oder `back`) und sich auf unterschiedliche Teilbereiche des Operanden beziehen (Attribute `left`, `top` und/oder `right`).

Schreiben Sie außerdem ein Hauptprogramm, das den Parser auf einer per Kommandozeilenargument übergebenen Quelldatei ausführt und die erkannten Ausdrücke ausgibt!