Fachhochschule Bingen

Programmieren

BNF / Syntaxdiagramme

Prof. Dr. Maximilian Mengel,
Professur Programmiermethodik,
Grundlagen der Informatik und Multimedia
Gebäude 1, Raum 212
Tel.: 06721-409 152
E-Mail: mengel@fh-bingen.de

Programmiersprachen: Die Syntax

- Eine notwendige (wenn auch leider nicht hinreichende) Voraussetzung zum Schreiben eines korrekten Programms ist das Beachten der Syntax
  - Das Programm muß so geschrieben sein, daß es den Syntax-Regeln genügt
  - Das Programm kann durch einen entsprechenden Compiler übersetzt werden
- Die Syntax einer Programmiersprache beschreibt den formalen Aufbau aller gültigen Programme dieser Programmiersprache

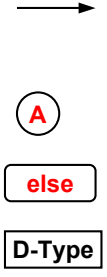
Beschreibung der Syntax

- Um die Syntax einer Programmiersprache zu Beschreiben existieren mehrere Methoden:
  - Syntaxdiagramme
    - Graphische Diagramme zur Definition der Syntax
    - Graphische Darstellung erleichtert Interpretation
  - Backus-Naur-Form (BNF) bzw. Erweiterte-Backus-Naur-Form (EBNF)
    - Ableitungsregeln zur Definition der Syntax
    - Grundlage für Parser (erster „Einsatz“ der BNF zur Spezifikation von ALGOL-60)

Syntaxdiagramme

- Graphische Ableitungsregeln, durch die alle korrekten Programme einer Programmiersprache beschrieben sind
- Die Ableitungsregeln bauen aufeinander auf
- Die kleinsten Einheiten sind:
  - Einzelne Zeichen
  - Schlüsselworte
  - Syntaxelemente

Grundelemente von Syntaxdiagrammen



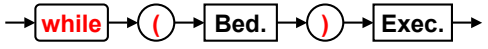
- **Pfeile** geben an wie ein Diagramm durchlaufen wird. Verzweigungen zeigen alternative Wege
- **Einzelne Zeichen**, die genau so im Programm stehen müssen
- **Schlüsselworte**, die genau so im Programm stehen müssen
- **Bezeichner** sind Syntaxelemente, für die es wiederum ein eigenes Syntaxdiagramm zur Beschreibung gibt

Formale Grundelemente einer Syntax

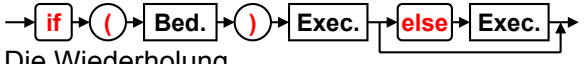
- **Token bzw. Terminale**
  - Die kleinsten nicht weiter teilbaren Elemente
  - **Schlüsselworte**
  - **Einzelne Zeichen**
- **Nonterminale**
  - Komplexe Gebilde, die noch anderweitig aufgeschlüsselt werden müssen
  - **Bezeichner**
- Die Programmiersprache selbst besteht nur aus **Tokens** und **Terminalen**

Typische Syntaxdiagramme

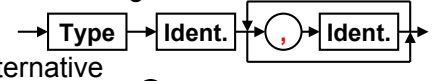
- Die Sequenz



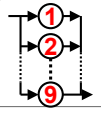
- Optionale Elemente



- Die Wiederholung



- Die Alternative



Die Backus-Naur-Form (BNF)

- Metasprachliche Beschreibung einer Syntax
  - Ableitungsregeln zur Beschreibung der Syntax
  - Das **Alphabet** der zu beschreibenden Sprache sollte die in der Metasprache benutzten Symbole möglichst nicht enthalten
  - Bezeichner werden definiert durch
    - Bezeichner
    - Einzelne Zeichen
    - Schlüsselworte

Elemente der BNF

- < > — Nonterminale werden in diesen Klammer eingeschlossen
- ::= — Links von diesem Definitionszeichen steht das zu definierende Nonterminale; rechts davon die Ableitungsvorschrift
- | — Die vor und nach diesem Zeichen stehenden Sprachelemente bilden sich ausschließende Alternativen.
- Ø — Das Leere Zeichen.

Beispiel BNF

- IF-ELSE Statement:

  <IFStatement> ::= IF (<Expression>) <Statement> <Elsepart>
  <Elsepart> ::= ELSE <Statement> | Ø
- Variablenname:

  <Identifier> ::= <Buchstabe> | _ |
  <Identifier> _ |
  <Identifier><Buchstabe> I
  <Identifier><Ziffer>
- Übung:
  - Definieren Sie mittels BNF eine Telefonnummer, die aus einer optionalen Vorwahl und einer Rufnummer besteht (Vorwahl und Rufnummer jeweils mindestens drei Ziffern)

EBNF: Erweiterte BNF

- In der Erweitereten BNF existieren zusätzliche Möglichkeiten, die eine Definition vereinfachen
  - [ ] — In eckige Klammern eingeschlossene Sprachelemente sind optional.
  - { } — In geschweifte Klammern eingeschlossene Elemente können wiederholt vorkommen.
  - * — Mit * versehene geschweifte Klammern können beliebig oft vorkommen (auch keinmal).
  - + — Mit + versehene geschweifte Klammern müssen mindestens einmal vorkommen.

Typische Beispiele EBNF

- Die Sequenz

  <while-loop> ::= while (<Bedingung>) <Exec>
- Optionale Sprachelemente

  <if-else> ::= if (< Bedingung >) < Exec> [else < Exec>]
- Wiederholung

  <Declaration> ::= <Type> <Identifier> {, <Identifier>}*
- Alternative

  <Non-Zero-Digit> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Beispiel

- Telefonnummern:

<Non-Zero-Digit>::= 1|2|3|4|5|6|7|8|9
<Digit> ::= 0|<Non-Zero-Digit>
<Local-Call> ::= <Non-Zero-Digit><Digit>{< Digit>}$^{+}$
<City> ::= <Non-Zero-Digit>{< Digit>}$^{+}$
<Country> ::= <Non-Zero-Digit>|
<Non-Zero-Digit><Non-Zero-Digit>|
<Non-Zero-Digit>< Non-Zero-Digit><Non-Zero-Digit>
<Phone-Number>::= <Local-Call>|
0<City><Local-Call>|
00<Country><City><LocalCall>

- Übung:
  - Erstellen Sie die EBNF-Definition für Zahlen-Ausdrücke mit +, -, *, /, (,). Zahlen wie 003 sind verboten. +3 oder -7 sind OK.