Inhaltsverzeichnis

1. Compilerbau: EBNF und SYNTAX-Graphen

1.1. Ziele

☑ EBNF und Syntaxgraphen zur

- ☐ **Beschreibung der Grammatik/Syntax einer Sprache** einzusetzen ,
- ☐ **Einfaches Scanner- u. Parser-Programm**

1.2. RDP-Aufgabe: EBNF und arithmetische Ausdrücke

Die **Korrektheit von arithmetischen Ausdrücken** bestimmen.

Erstellen Sie eine EBNF-Grammatik und ein Programm, das von der Standardeingabe einen arithmetischen Ausdruck einliest und die Korrektheit der Eingabe bestimmt.

```
Beispiele:
(3+4)       3*4+5       3           4(5+7)                6*(7+5)+8
korrekt     korrekt     korrekt     nicht korrekt         korrekt
```

1. **Besprechen Sie in diesem Zusammenhang die Elemente der EBNF-Grammatik.**
2. **Geben Sie eine EBNF-Grammatik eines arithmetischen Ausdruckes an.**
3. **Skizzieren Sie die Implementierung eines entsprechenden Parser-Programmes.**

1.3. BNF, EBNF

1.3.1. BNF

Die BNF http://de.wikipedia.org/wiki/Backus-Naur-Form

ist eine **Beschreibung** für (Programmier)sprachen, **um** deren **Syntax/Grammatik zu definieren.**

Dabei verwendet man sogenannte **Produktionsregeln**.

1.3.1.1 Der Begriff: Produktionsregeln

☑ Die **Produktionsregeln** bestehen aus folg. Elementen:

- ☐ **Definition**
  die Zeichenfolge **::=** legt Produktionsregeln fest
- ☐ **Alternative**
  das Zeichen **|** wird zur Definition von Alternativen verwendet
- ☐ **Nichtterminalsymbole**
  werden mit spitzen Klammern **<…>** umschlossen
- ☐ **Terminalsymbole**
  werden mit **“...“** umschlossen

☑ Beispiel: Produktionsregel für Ziffern außer Null

```
<Ziffer außer Null> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

Anmerkung: Man findet häufig folg. Varianten:

- '=' statt '::='
- Die spitzen Klammern und die Anführungszeichen fehlen oft.
- Ein Punkt kennzeichnet das Ende einer Produktionsregel.

1.3.1.2 Der Begriff: Sequenz

Eine **Sequenz** Ist eine **Abfolge** von **Terminalsymbolen** und **Nichtterminalsymbolen**

☑ Beispiele: Sequenz

```
<Ziffer>              ::= 0 | <Ziffer außer Null>

<Zweistellige Zahl>   ::= <Ziffer außer Null> <Ziffer>

<Zehn bis Neunzehn>   ::= 1 <Ziffer>

<Zweiundvierzig>      ::= 42
```

1.3.1.3 Der Begriff: Wiederholungen

Wiederholungen müssen in BNF über **Rekursionen** definiert werden:

Beispiel:

```
<Ziffernfolge>          ::= <Ziffer> | <Ziffer> <Ziffernfolge>
<Ziffer>                ::= 0 | <Ziffer außer Null>
<Ziffer außer Null>     ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

Lies: *Eine Ziffernfolge ist eine Ziffer oder eine Ziffer gefolgt von einer Ziffernfolge*.

Frage: Eine Ziffernfolge passt also zu den

- Symbolfolgen 0, 1, 2, 10, 9870, 8970635 usw., jedoch auch zu 00, 000, ….

Frage: Eine positive Zahl darf nicht mit 0 beginnen.

- Antwort: Dies leistet die folgende Regel:

```
<Positive Zahl> ::= <Ziffer außer Null> | <Ziffer außer Null> <Ziffernfolge>
```

1.3.1.4 Zusammenfassung: BNF

☐ Definition	**::=**
☐ Alternative	**|**
☐ Sequenz	**<Ziffer außer Null> <Ziffernfolge>**
☐ Terminale Symbole	**"0"**
☐ Nichtterminale Symbole	**<dot-string>**

1.3.1.5 Beispiele: email und C-Syntax

☑ Im (Request for Comments) RFC821 und **RFC822** werden u.a. der Aufbau einer Email-Adresse definiert.

☑ BNF: Sprachmittel
In **RFC 821 auf S. 29ff.** Am Beispiel einer konkreten E-Mail-Adresse sehen wir folgende Sprachmittel der BNF:

☑ **Definition:**	**::=**
☑ **Nichtterminal-Symbole:**	**<mailbox>**
☑ **Terminal-Symbole:**	**"@"**
☑ **Alternative:**	**|**
☑ **Sequenz:**	**<dot-string> "@" <dot-string>**

Auch die Rekursion spielt eine bedeutende Rolle.

Der Einfachheit halber reduzieren wir die offizielle Definition auf die folgende stark vereinfachte Form.

```
<mailbox>          ::= <dot-string> "@" <dot-string>
<dot-string>       ::= <string> | <string> "." <dot-string>
<string>           ::= <char> | <char> <string>
<char>             ::= any one oft he 128 ASCII characters, but not any <special>
                       or the space character (ASCII code 32)
<special>          ::= "<"    |  ">"   | "("   |  ")"  |  "@" …...
```

Hier ein Ausdruck, der sich auf die obige BNF-Syntax bezieht.
Max.Mustermann@schule.at

☑ Hier ein Beispiel zu BNF und die Sprache C
http://www.cs.man.ac.uk/~pjj/bnf/c_syntax.bnf

1.3.2. EBNF

In der **erweiterten BNF (EBNF)** gibt es einige zusätzliche Möglichkeiten:

- **Optionale** Teile stehen in eckigen Klammern: z.B.: **[<*Elsepart*>]**
- **Wiederholungen** Geschweifte Klammern umschliessen beliebige (auch Null):
  <var> **{, <var>}**
- **Prioritäten** werden durch runde Klammern gesetzt:
  (<*A*>|<*B*>) <*C*>

Aus der **EBNF** lässt sich häufig **direkt** ein Parser erstellen! (s. u.)

1.3.2.1 Übung: Worte einer EBNF-Grammatik bestimmen

☑ Zählen Sie alle Worte der folgenden Sprache auf. Das Startsymbol ist S.

```
S ::= A B.
A ::= "a" | "b".
B ::= "c" | "d".
```

Lösung:
L = {ac, ad, bc, bd} stimmt: o ja o nein

☑ Zählen Sie alle Worte der folgenden Sprache auf. Das Startsymbol ist A.

```
A ::= "a" [B] | C
B ::= "b" "c" ("b" | "c")
C ::= "d" (["e"] | "f")
```

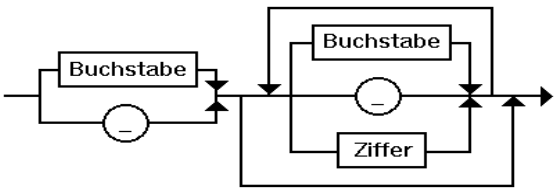
Lösung:
L = {a, abcb, abcc, d, de, df} stimmt: o ja o nein

1.4. SYNTAX-Graphen/Diagramme

Zur **Veranschaulichung** der Syntax formaler Sprachen eignen sich am besten Syntaxgraphen/Syntaxdiagramme (siehe z.B. Wirth (1986)).

1.4.1. Übung: Syntaxgraph: Bezeichner

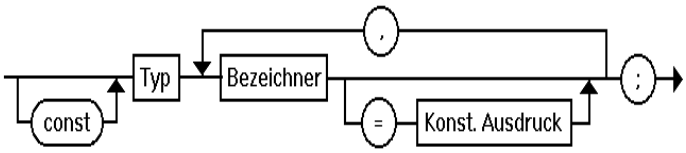
Gegeben sei folgender Syntax-Graph:



☑ **Frage**: Welche Sätze entsprechen dem obigen Syntax-Graphen?

- ☑ _ stimmt: o ja o nein
- ☑ _1B stimmt: o ja o nein
- ☑ 1_3 stimmt: o ja o nein
- ☑ 1_B stimmt: o ja o nein
- ☑ 1B stimmt: o ja o nein

1.4.2. Übung: Syntaxgraph: Definition

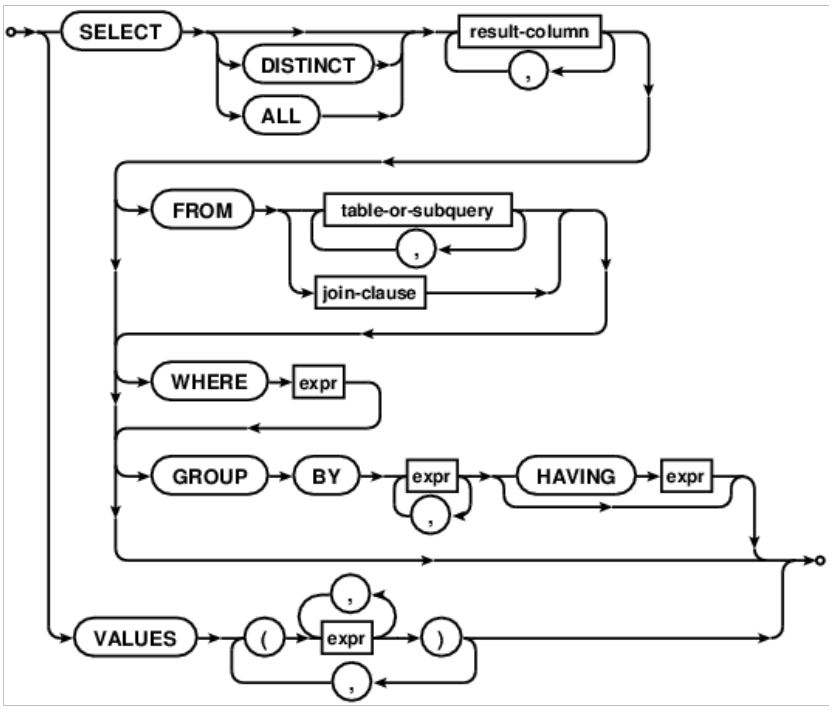


☑ **Frage**: Welche Sätze entsprechen dem obigen Syntax-Graphen?

☑ int const name;	stimmt: o ja	o nein
☑ double _=123, fertig=33	stimmt: o ja	o nein
☑ float bar;	stimmt: o ja	o nein
☑ float betrag=3.14;	stimmt: o ja	o nein

1.4.3. Übung: Syntaxgraph: SELECT

http://www.sqlite.org/syntaxdiagrams.html



☑ **Frage**: Welche Anweisungen entsprechen der obigen Grammatik

☐ select * from tabelle
stimmt: o ja o nein

☐ select spalte1 as nr, id from tabelle where id >100 sort by nr
stimmt: o ja o nein

☐ select * from tabelle1 union select * from tabelle2
stimmt: o ja o nein

☐ select spalte1 as nr, spalte2 from tabelle as group by tabelle.spalte1, spalte2
stimmt: o ja o nein