G 6. Übung

- Besprechung 3. Aufgabe
- Backus Nauer Form (BNF)
- Syntaxanalyse mit rekursivem Abstieg

G-1 Backus-Nauer-Form

- Satz von Produktionsregeln
- Die Produktionsregeln haben die Form.

```
Symbol := Alternative1 | Alternative2 ...
```

- Das Symbol links von ":=" kann durch eine der Alternativen ersetzt werden
- Die Alternativen bestehen aus Symbolen oder Terminalsymbolen

G-2 Einfaches Beispiel

- Folgende Grammatik beschreibt die Sprache aller realen Zahlen

```
S := '-' FN | FN
FN := DL | DL '.' DL
DL := D | D DL
D := '0' | '1' | '2' | '3' | '5' | '6' | '7' | '8' | '9'
```

- Von S ausgehend kann z.B. 3.14 produziert werden 3..14 nicht

```
S
FN
DL . DL
3. DL
3. D DL
3.1 DL
3.1 D
3.14
```

G-3 Rekursiver Abstieg

- Zu jeder Produktionsregel eine Funktion
- Symbole führen zum Abstieg (divide and conquer)
- Terminalsymbole werden ausgewertet

G-4 Beispiel (Parser für reale Zahlen)

```
/**
 * S := '-' FN | FN
 */
int parseFloat(float* value, char *str) {

   if (str[0]=='-') {
      if (parseUnsignedFloat(value,str+1)) return -1;
      *value = -*value;
   } else {
      return parseUnsignedFloat(value,str);
   }

   return 0;
}
```

G-5 Beispiel

```
/**
* FN := DL | DL '.' DL
*/
int parseUnsignedFloat(float* value, char *str) {
    int ivalue = 0, dvalue = 0, i, dez = 1;

    if (parseInteger(&ivalue,str,strlen(str))) {
        char *dpnt = strchr(str,'.');
        if (dpnt == NULL ) return -1;
        if (parseInteger(&ivalue,str,dpnt - str)) return -1;
        dpnt++;
        if (parseInteger(&dvalue,dpnt,strlen(dpnt))) return -1;

        for (i=0;i<strlen(dpnt);i++) dez = dez * 10;
        *value = (float)ivalue + ((float)dvalue/dez);
        return 0;
    }
    *value = (float) ivalue;
    return 0;
}
```

G-6 Beispiel

```
/**
* DL := D | D DL
*/
int parseInteger(int* value, char *str, int length) {
    int digit,rest,i,dez;

    if (length==1) return parseDigit(value,str);

    if (parseDigit(&digit,str) ||
        parseInteger(&rest,str+1,length-1)) return -1;

    dez=1;
    for (i=0;i<length-1;i++) dez = dez * 10;
    *value = digit * dez + rest;

    return 0;
}
```

G-7 Beispiel

```
/**
 * D := '0' | '1' | '2' | '3' | '5' | '6' | '7' | '8' | '9'
 */

int parseDigit(int *value, char *str) {
    if (str[0]>='0' && str[0]<='9') {
        *value = str[0] - '0';
        return 0;
    }
    return -1;
}
```

G-8 Aufgabe (Präfix-Notation)

```
S := SN
SN := OP SP SN SP SN | DL
DL := D | D DL
SP := SC | SC SP
OP := '+' | '-' | '*' | '/'
SC := ' ' | '\t' | '\n' | '\r'
D := '0' | '1' | '2' | '3' | '5' | '6' | '7' | '8' | '9'
```

```
int main (int argc, char *argv[]) {
    int value;
    if (evalFormel(&value,argv[1])) {
        printf("syntax error\n");
    } else {
        printf(" %s = %d\n",value);
    }
}
```