Taschenrechner Version 4

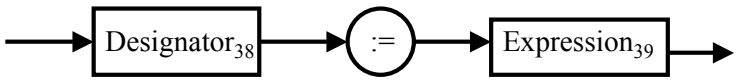
Wie wird eine **einfache Addition** ausgeführt ?

```
Module Taschenrechner;
(* Dies ist das Taschenrechnerprogramm Version 4 *)

FROM InOut IMPORT WriteInt,ReadInt,WriteString,WriteLn;
VAR    x,y : INTEGER;
       Erg : INTEGER;

BEGIN
   WriteString('Hallo, ich bin ein Taschenrechner.');
   WriteLn;
   WriteString('Eingabe x:'); ReadInt(x);
   WriteString('Eingabe y:'); ReadInt(y);
   Erg := x + y;
   WriteString('Ergebnis:'); WriteInt(Erg); WritelLn;
END Taschenrechner.
```


Wertzuweisung

Eine Zuweisung verändert den Wert einer Variablen

Assignment$_{46}$



- Reihenfolge der Ausführung
  1. Designator auswerten
  2. Ausdruck (Expression) auswerten
  3. Ergebnis eintragen
- Eine Variable kann **gleichzeitig** links und rechts des Zuweisungsoperators vorkommen

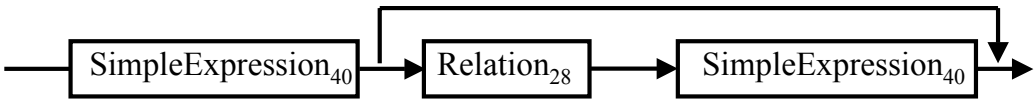
Beispiel für Zuweisungen:

```
x := 2 + y;
x := x*x; (* Berechnet das Quadrat von x *)
```

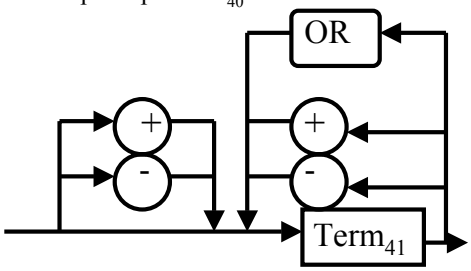
Ausdrücke

- Ein **Ausdruck** (Expression) ist eine Vorschrift, wie ein Wert aus anderen Werten berechnet werden soll
- Ein Ausdruck besteht aus **Operanden** und **Operatoren**
- Modula-2 schreibt vor, welche Arten von Ausdrücken es gibt und wie Ausdrücke gebildet werden können.

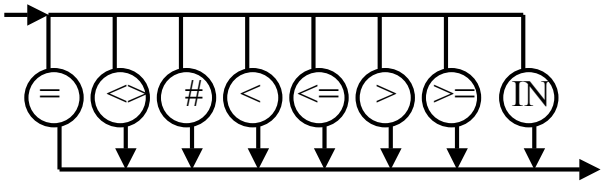
Expression$_{39}$



SimpleExpression$_{40}$



Relation$_{28}$



Operanden

Als **Operanden** sind zugelassen:

- **Konstante**: Zahlenkonstante, Zeichenkette, benannte Konstante
- **Variable** : Name der Variable, designierte Elemente von Feldern, Mengen, Verbunde (kommt später!)
- **Funktionsprozedur**: Berechungsvorschrift, die einen Wert zurückliefert (kommt später!)

Beispiel: Typen, Variablen, Ausdrücke

```
CONST one = 1;
VAR x,y : INTEGER;
    ...
    y := x + 1;
    y := y + one;
```

Operatoren

Arten von Operatoren

- Arithmetische Operatoren → arithmetisches Ergebnis
- Logische Operatoren → logisches Ergebnis
- Mengenoperatoren → Menge als Ergebnis
- Vergleich von „beliebigen“ Operanden → logisches Ergebnis

Operatoren

Arithmetische Operatoren

- Präzedenzrelation der arithmetischen Operatoren

```
  *   /
 DIV MOD
  +   -
```

Vergleichsoperatoren

Bei **gleicher** Präzedenzstufe von **links** nach **rechts**

Mit einer **Klammerung** kann die Reihenfolge verändert werden.

Beispiel:

```
6 * (11 MOD 3) + 1
6 *     2      + 1
      12       + 1
        13
```

Operatoren

Logische Operatoren

- Präzedenzrelation der logischen Operatoren

  NOT
  AND
  OR
  Vergleichsoperatoren

  Bei gleicher Präzedenzstufe von **links** nach **rechts!**

  Mit einer **Klammerung** kann die Reihenfolge verändert werden.

Beispiel: Sei `X:=5` und `Y:=7` vorgegeben.

```
(Y > X) AND NOT (X <= 6)
  TRUE  AND NOT     TRUE
  TRUE  AND       FALSE
       FALSE
```

Typgleichheit

- Die Ausführbarkeit von Operationen hängt von der **Verträglichkeit der Datentypen** der Operanden ab
- Zwei Variablen x und y haben den **gleichen Typ**, wenn

  Typ (x) ist gleich Typ (y)

  oder

  Typ (x) und Typ (y) in einer Typdeklaration als synonym deklariert wurden.
- Der Compiler kennt einige wenige **Konversionsregeln**. Zum Beispiel kann eine **Cardinalzahl** in eine **Integerzahl** konvertiert werden.

Typgleichheit

Beispiel:

```
VAR X,Y,ERG : CARDINAL;
    RES : REAL;
    SUM : INTEGER;
...
```

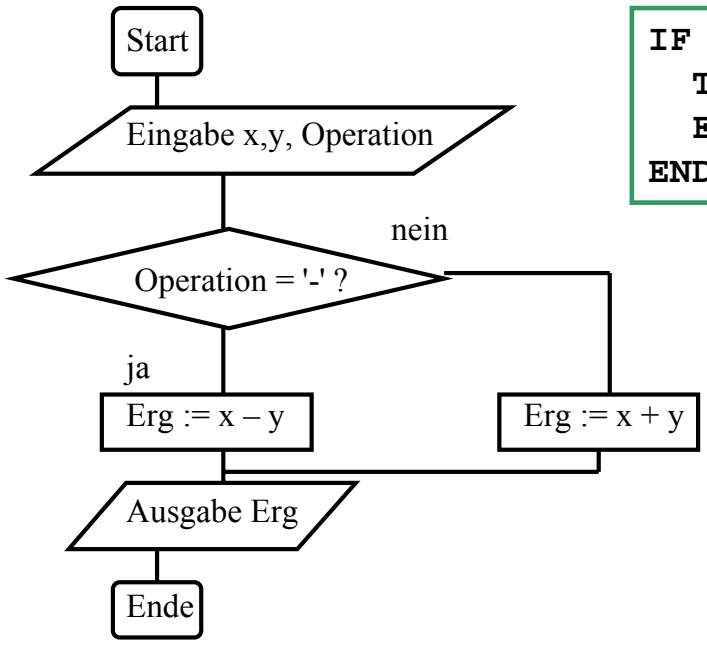
```
ERG := X + Y;
```
Möglich, weil `X`, `Y` und `ERG` vom selben Typ (`CARDINAL`) sind.

```
RES := X + Y;
```
Nicht möglich, weil `X` und `Y` vom Typ `CARDINAL` sind und `RES` vom Typ `REAL` ist.

```
SUM := X + Y;
```
Möglich, da `CARDINAL` in `INTEGER` konvertiert werden kann.

Taschenrechner Version 4

Erweitern des Taschenrechners um die **Subtraktion**.



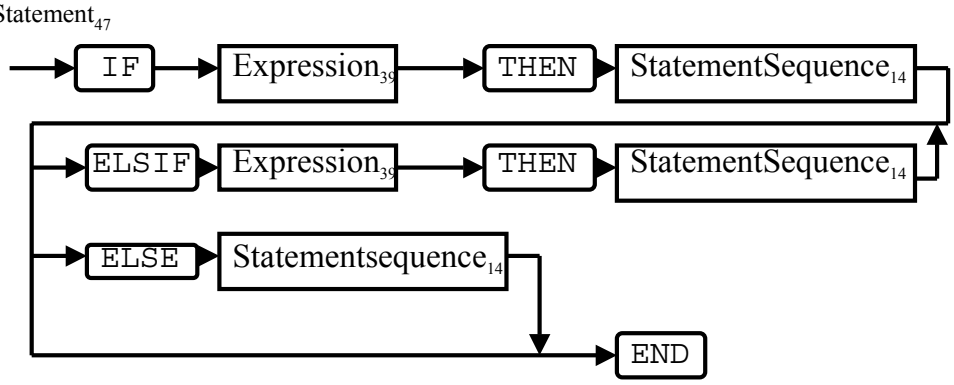

```
IF Op = '-'
  THEN Erg := x-y
  ELSE Erg := x+y
END;
```

Taschenrechner Version 4

```
Module Taschenrechner;
(* Dies ist das Taschenrechnerprogramm Version 4 *)
FROM    InOut IMPORT (*PROC*) WriteInt, ReadInt, WriteString,
                              WriteLn, Read;
VAR     x, y, Erg : INTEGER;
        Op : CHAR;      (* Angabe der Operation *)
BEGIN
   WriteString ('Hallo, ich bin ein Taschenrechner.');
   WriteLn;
   WriteString('Eingabe x:'); ReadInt (x);
   WriteString('Eingabe y:'); ReadInt (y);
   WriteString('Welche Operation (+,-) ?'); Read(Op);
   IF Op = '-'
        THEN Erg := x - y
        ELSE Erg := x + y;
   END;
   WriteString('Ergebnis:'); WriteInt(Erg); WriteLn;
END Taschenrechner.
```


Verzweigungen

Alternative Aktionen in Abhängigkeit einer **logischen Bedingung**

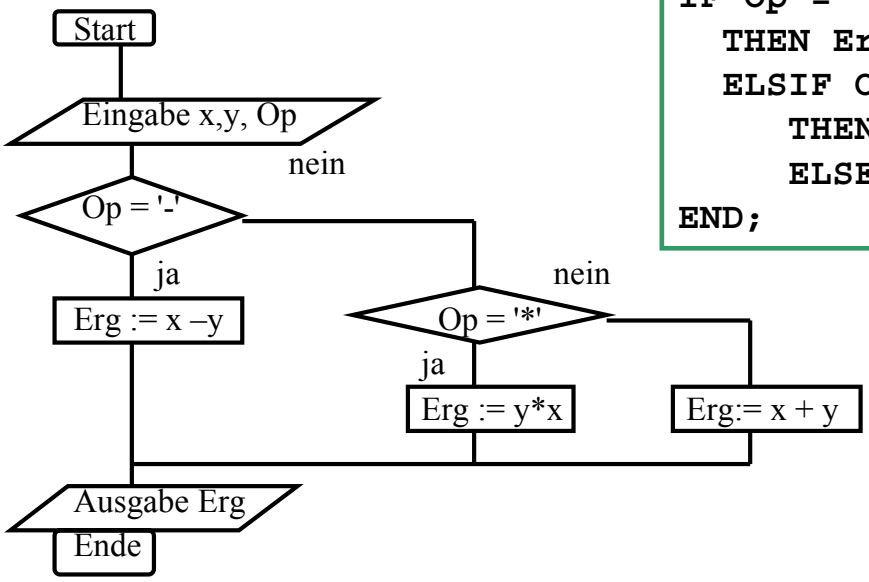


- Ausdrücke nach **`IF`** und **`ELSIF`** müssen immer vom Typ **`BOOLEAN`** sein
- Ergibt der Ausdruck **`TRUE`**, dann wird der **`THEN`**-Teil ausgeführt, sonst der **`ELSE`**-Teil

Taschenrechner Version 5.1

Erweitern des Taschenrechners um die Multiplikation.

Versuch 1: **`IF ... ELSIF ... ELSE ... END`** Konstrukt




```
IF Op = '-'
  THEN Erg := x-y
  ELSIF Op ='*'
     THEN Erg := x*y
     ELSE Erg := x+y
END;
```

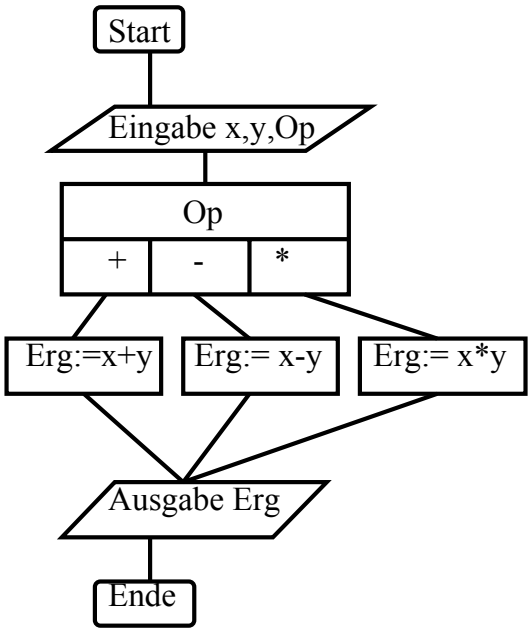

Taschenrechner Version 5.1

```
Module Taschenrechner;
(* Dies ist das Taschenrechnerprogramm Version 5.1 *)
FROM    InOut IMPORT (*PROC*) WriteInt, ReadInt, WriteString,
                              WriteLn, Read;
VAR     x, y, Erg : INTEGER;
        Op : CHAR; (* Angabe der Operation *)
BEGIN
   WriteString('Hallo, ich bin ein Taschenrechner.');
   WriteLn;
   WriteString('Eingabe x:'); ReadInt(x);
   WriteString('Eingabe y:'); ReadInt(y);
   WriteString('Welche Operation (+,-,*) ?'); Read(Op);
   IF Op = '-' THEN Erg := x - y
    ELSIF Op = '*' THEN Erg := x * y
     ELSE Erg := x + y;
   END;
   WriteString ('Ergebnis:'); WriteInt(Erg); WriteLn;
END Taschenrechner.
```

→ Diese Darstellung ist sehr unübersichtlich!

Taschenrechner Version 5.2

Versuch 2: CASE – Konstrukt




```
CASE Op OF
    '+' : Erg:= x+y
  | '-' : Erg:= x-y
  | '*' : Erg:= x*y

END;
```


Taschenrechner Version 5.2

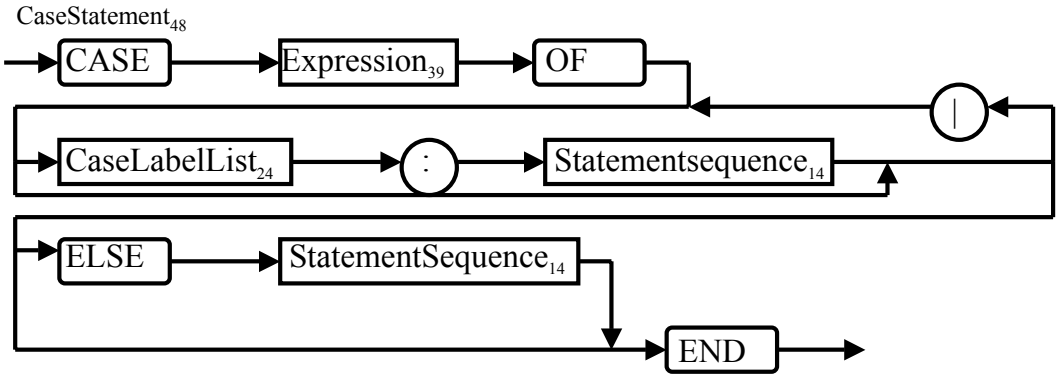
```
Module Taschenrechner;
(* Dies ist das Taschenrechnerprogramm Version 5.2 *)
FROM    InOut IMPORT (*PROC*) WriteInt, ReadInt, WriteString,
                              WriteLn, Read;
VAR     x, y, Erg : INTEGER;
        Op : CHAR; (* Angabe der Operation *)
BEGIN
   WriteString ('Hallo, ich bin ein Taschenrechner.');
   WriteLn;
   WriteString('Eingabe x:'); ReadInt(x);
   WriteString('Eingabe y:'); ReadInt(y);
   WriteString('Welche Operation (+,-,*) ?'); Read(Op);
   CASE Op OF
        '+' : Erg := x+y
   |    '-' : Erg := x-y
   |    '*' : Erg := x*y
   END;
   WriteString('Ergebnis:'); WriteInt(Erg); WriteLn;
END Taschenrechner.
```

Problem: Bei **anderen** Eingaben gibt es einen **Fehler** !!!

Verzweigungen

CASE-Anweisung

- **Alternative** Aktionen in Abhängigkeit des Wertes eines Ausdrucks



- Der Ausdruck darf vom Typ `INTEGER, CARDINAL, BOOLEAN, CHAR,` Enumeration oder Subrange sein.
- Der Konstantenausdruck muss mit dem Auswahl-Ausdruck **ausdruckskompatibel** sein.

Taschenrechner Version 5.2

```
Module Taschenrechner;
(* Dies ist das Taschenrechnerprogramm Version 5.2 *)
FROM    InOut IMPORT (*PROC*) WriteInt, ReadInt, WriteString,
                              WriteLn, Read;
VAR     x, y, Erg : INTEGER;
        Op : CHAR; (* Angabe der Operation *)
BEGIN
   WriteString ('Hallo, ich bin ein Taschenrechner.');
   WriteLn;
   WriteString('Eingabe x:'); ReadInt(x);
   WriteString('Eingabe y:'); ReadInt(y);
   WriteString('Welche Operation (+,-,*) ?'); Read(Op);
   CASE Op OF
        '+' : Erg := x+y
   |    '-' : Erg := x-y
   |    '*' : Erg := x*y
   ELSE WriteString('Die Eingabe war nicht korrekt')
   END;
   WriteString('Ergebnis:'); WriteInt(Erg); WriteLn;
END Taschenrechner.
```