

Anweisungen

Unter einer **Anweisung** versteht man einen Befehl oder eine Befehlsfolge im Programm zur Steuerung von Prozessen.

Einfache Anweisung

- Syntax: `ausdruck ;`
- jede Anweisung muß durch ein Semikolon abgeschlossen werden
- Bsp.:

```
a = a + 1; ←  
a += 1; ← Identische Anweisungen  
cout << a << endl;
```

Leere Anweisung

- Syntax: `;`
- nützlich, wenn syntaktisch Anweisung erforderlich, aber programmlogisch nicht
- Bsp.:

```
for(;;)  
    a = a + 1; ← Variable a unendlich um  
                1 erhöhen  
                (siehe 4.Übung)
```

Block-/Verbund-Anweisung

- Syntax:

```
{  
    ausdrück_1;  
    ausdrück_2;  
    ...  
    ausdrück_n;  
}
```
- dort, wo syntaktisch eine Anweisung verlangt wird, können mehrere Anweisungen durch Blockbildung zu einer Anweisung zusammengefaßt werden
- Variablendeklarationen innerhalb von Blöcken sind nur in selbigen gültig und verlieren an Gültigkeit, sobald die Blockanweisung beendet wird

• Bsp.:

```

#include <iostream>
using namespace std;

main()
{
    int i = 10;
    cout << i << endl;
    cout << "Speicheradresse = " << &i << endl;
    {
        int i = 5;
        cout << i << endl;
        cout << "Speicheradresse = " << &i << endl;
    }
    cout << i << endl;
    cout << "Speicheradresse = " << &i << endl;
}

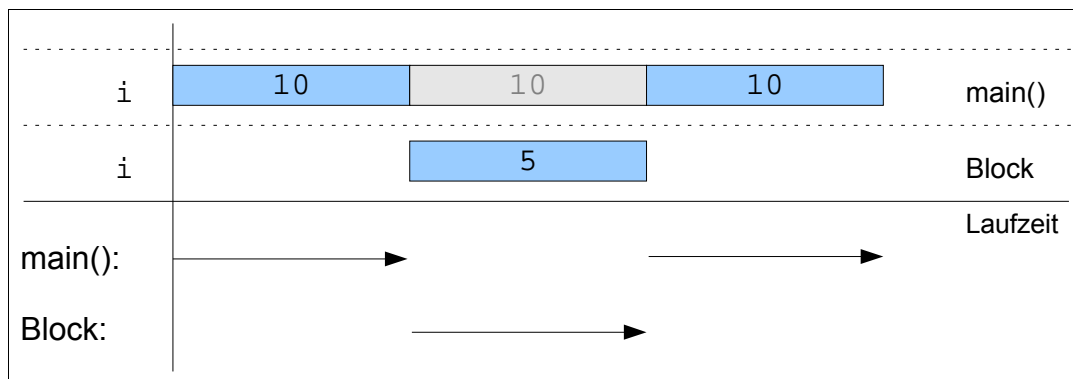
```

Dieser Block enthält "lokale Variable" i mit Wert 10.

Adress-Operator

Dieser Block enthält "lokale Variable" i mit Wert 5.

=> Speicheradressen unterschiedlich, d.h. unterschiedliche Variablen, jedoch selbe symbolische Bezeichnungen



Kontrollstrukturen

Kontrollstrukturen werden verwendet, um den Ablauf eines Computerprogramms zu steuern. Eine Kontrollstruktur gehört entweder zur Gruppe der **Verzweigungen** oder **Schleifen**. Ihre Ausführung wird durch sogenannte **Laufzeitbedingungen** beeinflusst, welche logische Ausdrücke darstellen.

- Verzweigungen:
 - if-else – Verzweigung
 - else if – Verzweigung
 - switch – Verzweigung
- Schleifen:
 - while – Schleife
 - do-while – Schleife
 - for – Schleife

Verzweigungen

Verzweigungen veranlassen den Programmfluss verschiedene Wege einzuschlagen. Der gewählte Fluß ist dabei von einer Bedingung abhängig. Die Art der Bedingungsauswertung klassifiziert Verzweigungen in **einfach bedingte Verzweigungen** und **mehrfach bedingte Verzweigungen**.

if-else – Anweisung

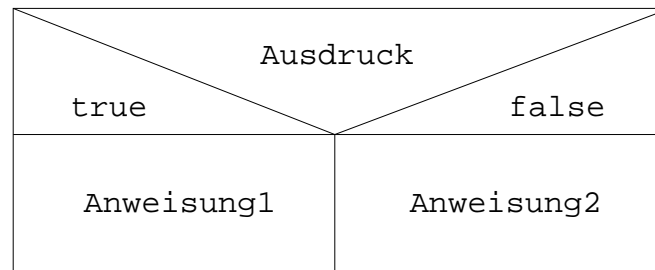
- Syntax:

```
if (Ausdruck)
    Anweisung1
[ else
    Anweisung2
]
```

wenn Ausdruck wahr,
dann führe Anweisung1 aus...

andernfalls, führe Anweisung2
aus (optional)

- if-Teil wird ausgeführt, wenn Ausdruck wahr ist, also `true` und somit ungleich 0
- der else-Zweig ist optional
- Struktogramm:



- Bsp.:

```
int a = 5, b = 4;
if (a < b)
    cout << "a ist kleiner als b" << endl;
else
    cout << "a ist größer oder gleich b" << endl;
```

- da das if-else – Konstrukt selbst eine Anweisung ist, ist Verschachtelung möglich:

```
if (n > 0)
    if (n%2 == 1)
        cout << "Positive und ungerade Zahl";
    else
        cout << "Positive und gerade Zahl";
```

% ... Modulo-Operator (gibt Rest der
Division zweier Zahlen zurück)

- da der else-Zweig optional ist, muß Regel definiert sein, wozu der aktuelle else-Teil gehört:

Ein `else`-Zweig gehört stets zum letzten `if`, dem noch kein `else` zugeordnet ist.

- soll eine andere Zugehörigkeit eines `else`-Zweiges erreicht werden, so müssen Block-Anweisungen verwendet werden:

```

if ( n > 0 )
{
    cout << "Positive ";
    if ( n%2 == 1 )
        cout << " und ungerade ";
    cout << "Zahl" << endl;
}
else
    cout << "Die Zahl ist negativ oder null\n";

```

bei mehreren Anweisungen muß Block gebildet werden, da syntaktisch nur eine Anweisung im if-Zweig erlaubt ist

- es ist möglich, im `if`-Ausdruck eine Variable zu definieren und zu initialisieren, ihre Gültigkeit beschränkt sich dann auf den `if`-Zweig:

```

#include <cmath>
using namespace std;
...

double x;
cin >> x;
if (double y = sin(x))
{
    cout << "Der Sinus von x ist größer oder kleiner 0\n";
    cout << "Sein Wert beträgt: " << y << endl;
}

```

enthält Sinus Funktion:
double sin(double arg);

else if – Anweisung

- durch Bildung einer `else if` – Kette kann die Auswahl mehrerer Alternativen programmiert werden

Syntax:

```

if (Ausdruck_1)
    Anweisung_1
else if (Ausdruck_2)
    Anweisung_2
...
else if (Ausdruck_n)
    Anweisung_n
[ else Anweisung ]

```

wenn Ausdruck_1 wahr, dann führe Anweisung_1 aus

wenn Ausdruck_2 wahr (und Ausdruck_1 nicht), dann führe Anweisung_2 aus

wenn alle n-1 Ausdrücke falsch und Ausdruck_n wahr, so führe Ausdruck_n aus

trifft kein Ausdruck zu, führe Anweisung aus (optional)

- Hinweis: ist ein Ausdruck wahr, so wird zugehörige Anweisung ausgeführt und die `else if` – Kette beendet

Bsp.:

```
#include <iostream>
using namespace std;

main()
{
    int i = -10;
    if ( i < 0 )
        cout << "Kleiner 0" << endl;
    else if ( i > 0 )
        cout << "Größer 0" << endl;
    else
        cout << "Gleich 0" << endl;
}
```

- jede Auswahl kann mit else if – Kette implementiert werden, für den Spezialfall des Tests eines Ausdruckwertes auf Auswahl von Werten existiert Vereinfachung...

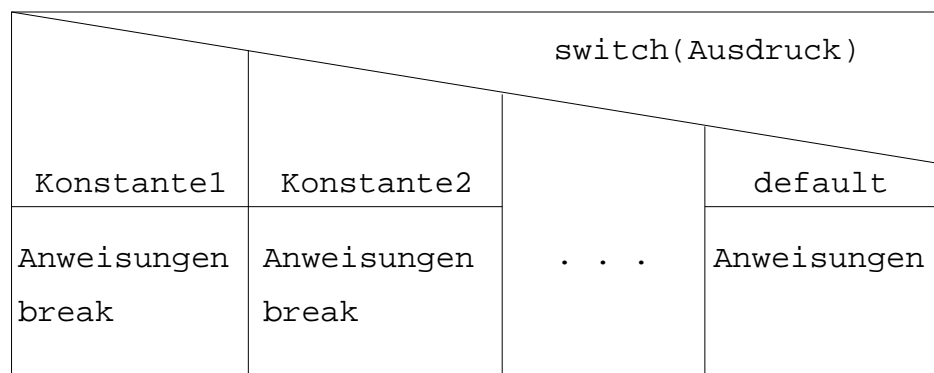
switch – Anweisung

- Syntax:

```
switch (Ausdruck)
{
    case Konstante1:    [Anweisungen]
                      [break;]
    case Konstante2:    [Anweisungen]
                      [break;]
    ...
    [default:          Anweisungen]
}
```

- Ausdruck muß ein ganzzahliger Typ sein, sein Wert wird dann mit Konstanten verglichen
- zu ganzzahligen Konstanten gehören auch Zeichenkonstanten und boolesche Konstanten

- Struktogramm:



- stimmt Ausdruck mit einer Konstante überein, so wird Anweisungsfolge ausgeführt
-> Hinweis: hier werden syntaktisch keine/eine/mehrere Anweisung(en) benötigt
- mit break Anweisung kann switch unmittelbar verlassen werden; wird sie nicht angegeben, so werden nachfolgende case-Fälle auch geprüft
- trifft keine Konstante zu, so werden die default-Anweisungen ausgeführt, sofern ein default-Fall definiert wurde

• Bsp.:

```

#include <iostream>
using namespace std;

main()
{
    float i, j;    // Operanden
    char op;      // Operator

    cout << "Gib Ausdruck ein: ";
    cin >> i >> op >> j;

    switch (op)
    {
        case '+':
            cout << "Ergebnis: " << i+j << endl;
            break;
        case '-':
            cout << "Ergebnis: " << i-j << endl;
            break;
        case '/':
            ← triff ein Fall zu, so führe alles aus bis zum break;
        case ':':
            if (j != 0)
                cout << "Ergebnis: " << i/j << endl;
            else
                cout << "Fehler: Division durch 0!\n";
            break; ←
        case '*':
            cout << "Ergebnis: " << i*j << endl;
            break;
        default:
            cout << "Operator unbekannt" << endl;
    }
}

```