Informatik

Kontrollstrukturen

Kontrollstrukturen

Sprachelemente zur Formulierung eines Programmablaufs

- **Sequenzen**

  Handlungen als Anweisungsfolgen formulieren

- **Bedingte Anweisungen**

  Anweisungen abhängig von einer Bedingung ausführen

- **Bedingte Ausdrücke**

  Werte abhängig von einer Bedingung berechnen

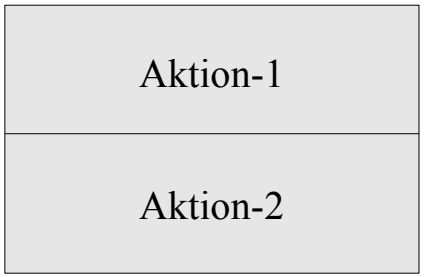
- **Schleifen**

  Anweisungen wiederholt ausführen

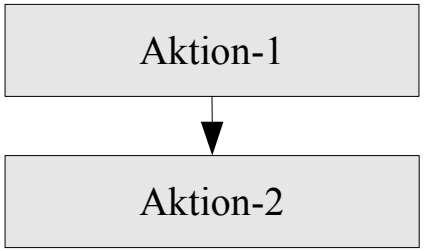
Sequenz

Handlungsfolge

- Im Programmtext durch Semikolon (und/oder) Untereinander-Schreibung getrennte Anweisungen
- werden von rechts nach links und von oben nach unten abgearbeitet



Struktogramm



Programmablaufplan

```
x = y+x;
y = h;
```

Programmbeispiel

Bedingte Anweisungen

Anweisungen abhängig von einer Bedingung ausführen

– **2-armige If-Anweisung**
  Eine von zwei Alternativen auswählen

– **1-armige If-Anweisung**
  Eine Aktion unter Umständen ausführen

– **Switch-Anweisung**
  Eine von mehreren Aktionen auswählen

```
if (x>y) {
  max = x;
} else {
  max = y;
}
```

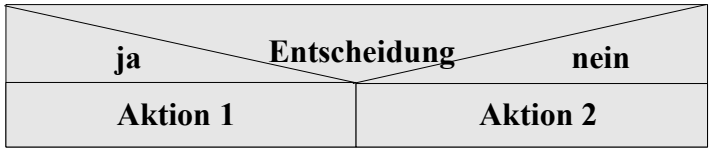
```
if (x>y) {
  max = x;
}
```

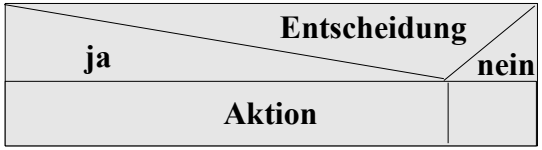
```
switch (x-2) {
case 1: y = 1; break;
case 2: y = 4; break;
case 3: y = 16; break;
default: y = 100;
}
```

If-Anweisungen

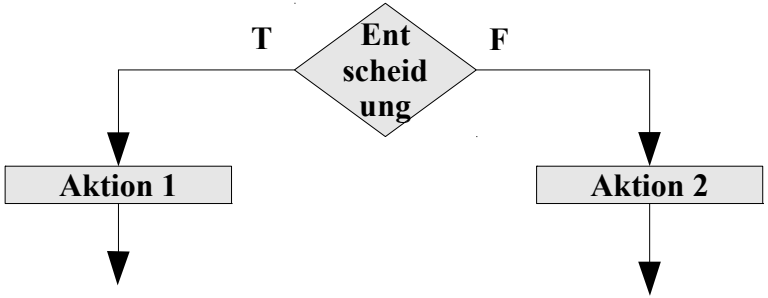
Formen:

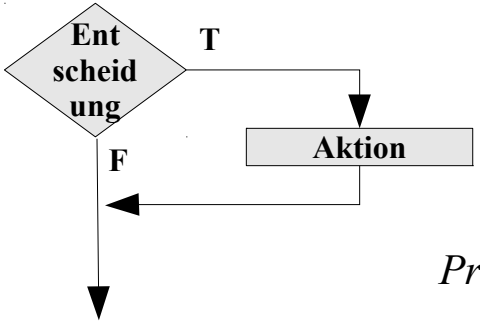
- **if (** *Bedingung* **)** *Anweisung*
- **if (** *Bedingung* **) {** $Anweisung_1$ $Anweisung_2$ … **}**
- **if (** *Bedingung* **)** $Anweisung_1$ **else** $Anweisung_2$
- **if (** *Bedingung* **) {** $Anweisung_{11}$ $Anweisung_{12}$ … **} else {** $Anweisung_{21}$ $Anweisung_{22}$ … **}**





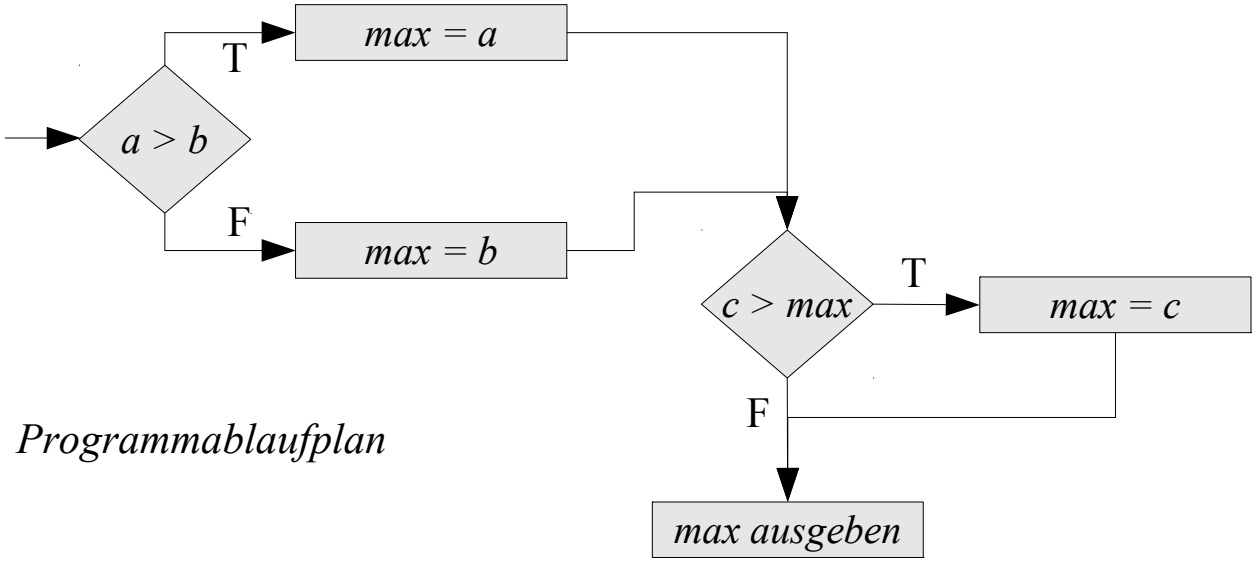
Struktogramme





Programmablaufpläne

If-Anweisungen / Beispiel



Programmablaufplan

Entwerfen Sie ein
entsprechendes Programm !

If-Anweisungen / Beispiel

```
package max;

import javax.swing.JOptionPane;

public class Maximum {

  public static void main(String[] args) {

    // Platz für groessten Wert
    double max;

    // Daten von Benutzer annehmen
    String aS = JOptionPane.showInputDialog("Bitte 1-te Zahl eingeben");
    String bS = JOptionPane.showInputDialog("Bitte 2-te Zahl eingeben");
    String cS = JOptionPane.showInputDialog("Bitte 3-te Zahl eingeben");

    double a = Double.parseDouble(aS);
    double b = Double.parseDouble(bS);
    double c = Double.parseDouble(cS);

    // Maximum von a und b bestimmen
    if ( a > b ) {
        max = a;
    } else
        max = b;

    // Maximum von a und b mit c vergleichen
   if ( c > max ){
        max = c;
    }
    // Ausgabe
    JOptionPane.showMessageDialog(null, "Maximum: " + max);
  }

}
```

Bedingte Anweisungen

Switch-Anweisung

nur Ausdrücke und Werte die mit **int** *kompatibel sind*

Form:

switch (*Ausdruck* **) {**

case *konstanter-Wert*$_1$ **:** *Anweisung*$_{12}$ *Anweisung*$_{13}$ … **break;**

case *konstanter-Wert*$_2$ **:** *Anweisung*$_{22}$ *Anweisung*$_{23}$ … **break;**

…

default : *Anweisung*$_{n1}$ *Anweisung*$_{n2}$ ….

optional

}

- Switch kann nur über **int** und eng verwandte Typen (char, short, byte) verzweigen
- der **default**-Zweig kann weggelassen werden
- **break** kann weggelassen werden, dann geht die Ausführung in den nächsten Zweig
- die *konstanten Werte* müssen zur Übersetzungszeit bekannt sein

Switch-Anweisung / Beispiel

```
String xS = JOptionPane.showInputDialog("Bitte 1-te Zahl eingeben");
String yS = JOptionPane.showInputDialog("Bitte 2-te Zahl eingeben");
String op = JOptionPane.showInputDialog("Operator");

int x = Integer.parseInt(xS);
int y = Integer.parseInt(yS);

switch ( op.charAt(0) ) {
case '+': System.out.println( x + y );
          break;
case '-': System.out.println( x - y );
          break;
case '*': System.out.println( x * y );
          break;
case '/': System.out.println( x / y );
          break;
default : System.out.println( "Ungültige Eingabe!" );
}
```

Bedingte Ausdrücke

Form:

Bedingung **?** *Ausdruck*$_1$ **:** *Ausdruck*$_2$

Verwendung

Alternative zu bedingten Anweisungen

Entscheidung in einem Ausdruck treffen (statt in einer Anweisung)

Beispiel:

```
max = (x>y) ? x : y;
```

∼

```
if (x>y) {
  max = x;
} else {
  max = y;
}
```

Bedingte Ausdrücke / Beispiele

```
max = a > b ? a : b;
max = c > max ? c : max;
```

~

```
if ( a > b ){
    max = a;
} else {
    max = b;
}
if ( c > max ){
    max = c;
}
```

Maximum von drei Werten mit bedingten Ausdrücken

Maximum von drei Werten mit bedingten Anweisungen

```
max = a>b ? a > c ? a
                  : c
          : b > c ? b
                  : c;
```

Maximum von drei Werten mit einem bedingten Ausdruck

Schleifen

Anweisungen wiederholt ausführen

– **while-Schleife**
Wiederholung solange eine Bedingung erfüllt ist

```
while (x<10) {
  x = x+1;
}
```

– **for-Schleife**
Eine Wertefolge durchlaufen

```
for (int i=0; i<10; i++) {
  x = x+i;
}
```

– **do-while Schleife**
while-Schleife mit mindestens einer Ausführung

```
do {
x = x+1;
} while (x<10);
```

While-Schleife

Form:

while (*Bedingung* **)** *Anweisung*

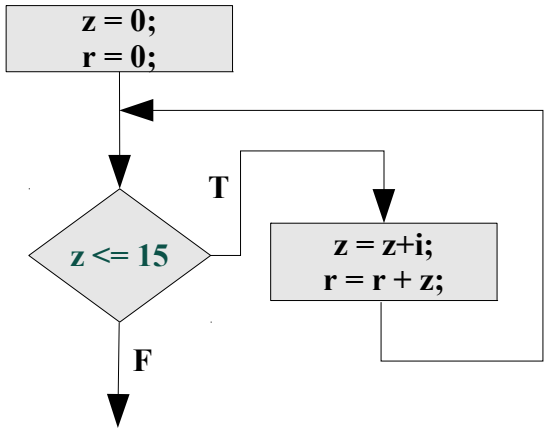
while (*Bedingung* **) {** $Anweisung_1$ $Anweisung_2$ … **}**

Verwendung

Anweisungen abhängig von einer Bedingung beliebig oft ausführen

Beispiel:

```
int z = 0;
int r = 0;
while (z < 15) {
    z = z+1;
    r = r+z;
}
```



While-Schleife / Beispiel

Die Summe der ersten n natürlichen Zahlen

```
int n = ……
int sum = 0;
int i   = 0;
while (i < n) {
    i = i + 1;
    sum = sum + i;
}
System.out.println("Summe 1 ... " + n + " = " + sum);
```

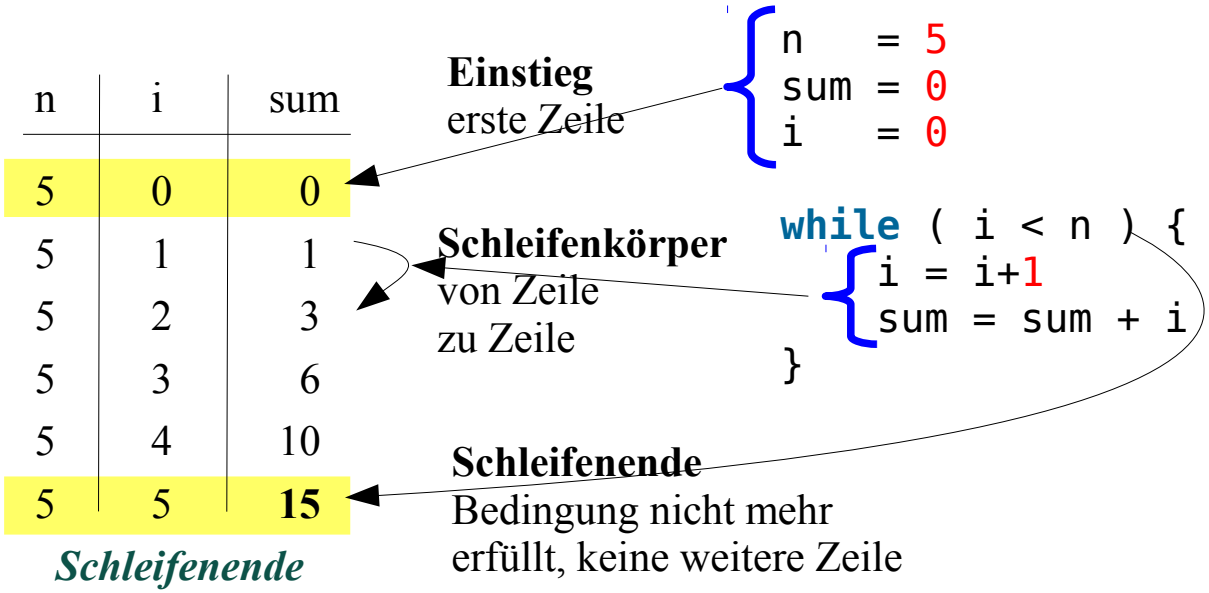
n	i	sum
5	0	0
5	1	1
5	2	3
	...	

Werteverlaufstabelle

Angenommen die Anweisungen in der Schleife werden vertauscht. Was passiert? Ist Berechnung noch korrekt ? Wenn nein: warum nicht ?

While-Schleife / Wertverlaufstabelle

Tabelle die die wechselnden Werte von Variablen darstellt.



While-Schleife / Schleifenkonstruktion

Konstruktion der Schleife aus einer Tabelle:

Welche Wertefolgen gibt es ?

=> In welcher Variablen wird welche Folge gespeichert

Wie werden die neuen aus den alten Werte in der Folge berechnet?

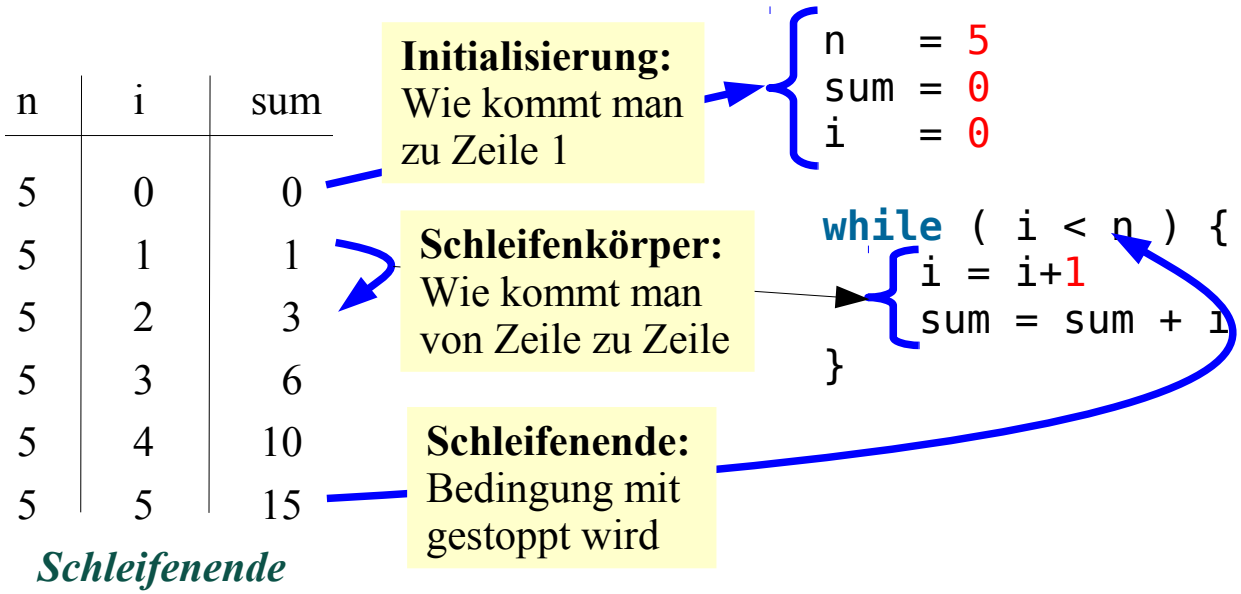
=> Schleifenkörper

Wann endet / wie lange läuft die Berechnung ?

=> Schleifenbedingung

While-Schleife / Schleifenkonstruktion

Konstruktion der Schleife aus einer Tabelle:



Do-While-Schleife

Form:

do { *Anweisung*$_1$ *Anweisung*$_2$ … } while (*Bedingung*)

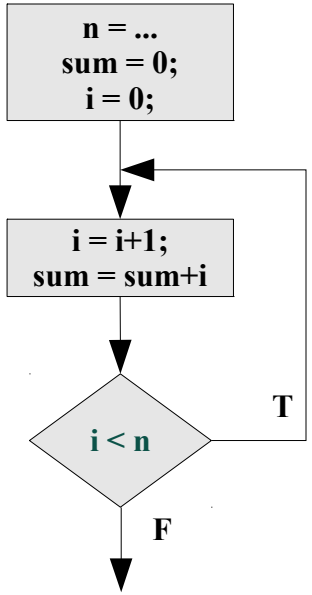
Verwendung

Schleifenkontrolle am Ende der Schleife, dadurch mindestens ein Durchlauf

„Fuß-gesteuerte“ Schleife

Beispiel:

```
int n = ……
int sum = 0;
int i   = 0;
do {
    i = i + 1;
    sum = sum + i;
} while (i < n);
```



For-Schleife

Form:

for (*Initizialisierungs-Anweisung***; *Bedingung*; *Inkrement-Anweisung*) *Anweisung***

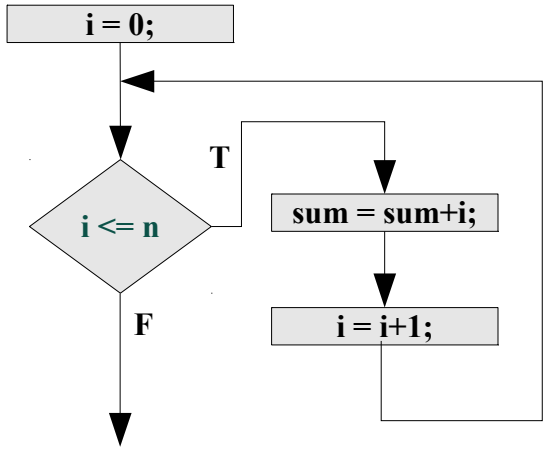
for (*Initizialisierungs-Anweisung***; *Bedingung*; *Inkrement-Anweisung*) {**
Anweisung$_1$ *Anweisung*$_2$ …
}

Verwendung

Schleife dient dem Durchlaufen einer Folge von Werten

Beispiel:

```
for ( int i = 0; i <= n; i=i+1 ) {
    sum = sum + i;
}
```



For-Schleife ist „verkleidete“ while-Schleife

Äquivalenz von while. und for-Schleife:

for (*Initizialisierungs-Anweisung***;** *Bedingung***;** *Inkrement-Anweisung* **) {**
    Anweisung$_1$ *Anweisung*$_2$ …
}

entspricht:

*Initizialisierungs-Anweisung***;**
while (*Bedingung* **) {**
    Anweisung$_1$ *Anweisung*$_2$ …
    Inkrement-Anweisung
}

i++ ~ i = i+1

```
for ( int i = 0; i <= n; i++ ) {
    sum = sum + i;
}
```

≈

```
int i = 0;
while ( i <= n ) {
    sum = sum + i;
    i = i + 1;
}
```

Kontroll-Anweisungen für Schleifen

- **`break`** - Anweisung

  beendet die Schleife

- **`continue`** -Anweisung

  bricht den aktuellen Schleifendurchlauf ab:
  weiter (continue !) mit dem nächsten Durchlauf

Kontroll-Anweisungen für Schleifen / Beispiel

```
int n = 0;
do {
  String nS = JOptionPane.showInputDialog("Bitte positive Zahl, Abbruch mit 0");
  n = Integer.parseInt(nS);;
  if (n < 0) continue;
  if (n == 0) break;
  int sum = 0;
  int i   = 0;
  while (i < n) {
    i = i + 1;
    sum = sum + i;
  }
  System.out.println("Summe 1 ... " + n + " = " + sum);
} while (true);
```

innere / geschachtelte Schleife