Kapitel 9

Schleifen

Schleifen

- Schleifen werden zur wiederholten Ausführung von Anweisungen verwendet.
- Es werden drei Arten von Schleifen unterschieden:
  - *for* -Schleife
  - *while* -Schleife
  - *do / while* -Schleife

Schleifen

- Die Anzahl der Schleifendurchläufe wird durch Bedingungen festgelegt.
- Bei der *for* und der *while* -Schleife stehen die Bedingungen am Beginn der Schleife.
- Bei der *do / while* -Schleife am Ende.

for -Schleife

- Ist die Anzahl der Schleifendurchläufe im Voraus bekannt, wird die *for* -Schleife verwendet.
- Die Elemente zur Kontrolle der Schleife werden im Schleifenkopf zusammengefasst.

```
for ([Initialisierung]; [Abbruchbedingung]; [Reinitialisierung])
        Anweisung;
```

for -Schleife

- Die **Initialisierung** erfolgt nur einmal und wird als Erstes vorgenommen.
- Die **Abbruchbedingung** wird zu Beginn jedes Schleifendurchlaufs bewertet.
- Ist der Wert der Abbruchbedingung *false*, wird die Schleife beendet.
- Ist der Wert *true*, wird die Anweisung (Schleifenkörper) ausgeführt und danach die **Reinitialisierung** durchgeführt.

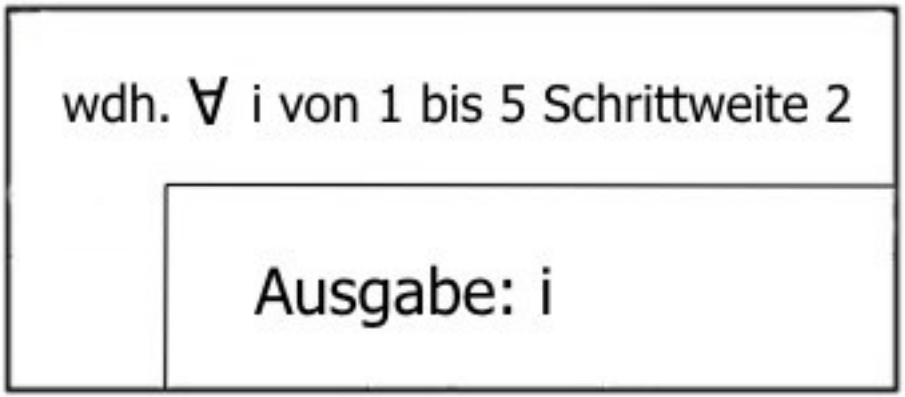
for -Schleife

```
int i;
for (i = 1; i <= 5; i+=2)
{
    cout << i << endl;
}
```

Ausgabe:

```
1
3
5
```

for -Schleife



for -Schleife

- Ist die Abbruchbedingung schon vor dem Eintritt in die Schleife *false*, wird der Schleifenkörper nicht durchlaufen.

```
int i, j = 1;
for (i = 1; j > 5; i++)
    cout << i << endl;
```

for -Schleife

- Soll mehr als eine Anweisung zum Schleifenkörper gehören, muss ein Block verwendet werden.

```
int i, j;
for (i = 1, j = 5; i <=  5; i += 2, j -= 2)
{
    cout << i << " ";
    cout << j << endl;
}
```

Ausgabe:

```
1  5
3  3
5  1
```

for -Schleife

- Die Elemente zur Schleifenkontrolle im Schleifenkopf können auch weggelassen werden.
- Bei folgendem Beispiel liegt eine Endlosschleife um eine leere Anweisung vor, da bei fehlender Abbruchbedingung *true* angenommen wird.

```
for ( ; ; ) ;
```

for -Schleife

- Bei folgendem Beispiel handelt es sich ebenfalls um eine Endlosschleife.
- Die Abbruchbedingung ist immer *true*.

```
int i;
for (i = 1; i < 5; i--)
     cout << i << endl;
```

for -Schleife

- Die Zählvariable kann auch innerhalb des Initialisierungsausdrucks definiert werden.

```
for (int i = 1; i <= 10; i++)
```

while -Schleife

- Die *while* –Schleife wird hauptsächlich verwendet, wenn die Abbruchbedingung erst innerhalb der Schleife festgelegt werden kann.
- Es wird vor Eintritt in die Schleife geprüft, ob die Schleife durchlaufen wird.

```
while (Ausdruck)
    Anweisung
```

- Der Ausdruck kann ganzzahlig oder vom Typ *bool* sein.
- Ist er ganzzahlig, entspricht 0 *false* und ungleich 0 *true*.

while -Schleife

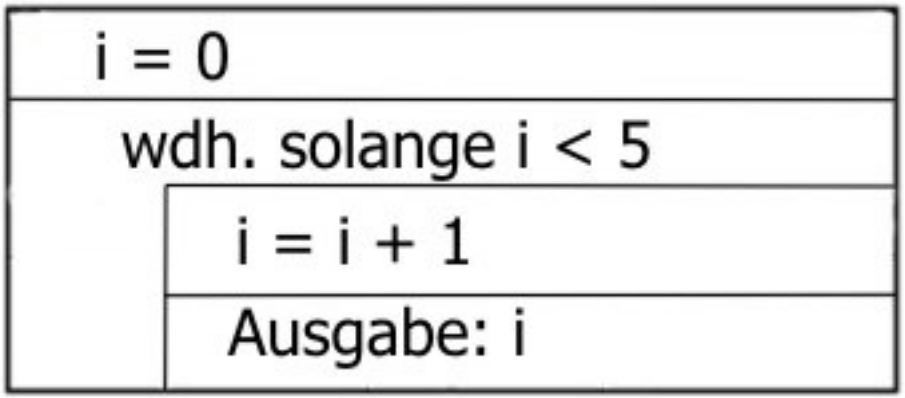
```
int i = 0;
while (i < 5)
    cout << ++i;
```

Ausgabe:

```
12345
```

while -Schleife

- Das zugehörige Struktogramm sieht wie folgt aus:



do/while -Schleife

- Das Abbruchkriterium der *do/while* –Schleife steht am Ende.
- Deshalb wird sie mindestens einmal durchlaufen.

```
do
    Anweisung;
while (Ausdruck);
```

- Die Schleife wird durchlaufen, solange der Ausdruck den Wert *true* hat.
- Im Gegensatz zur *while* –Schleife muss am Ende der *while* -Anweisung ein Semikolon stehen.

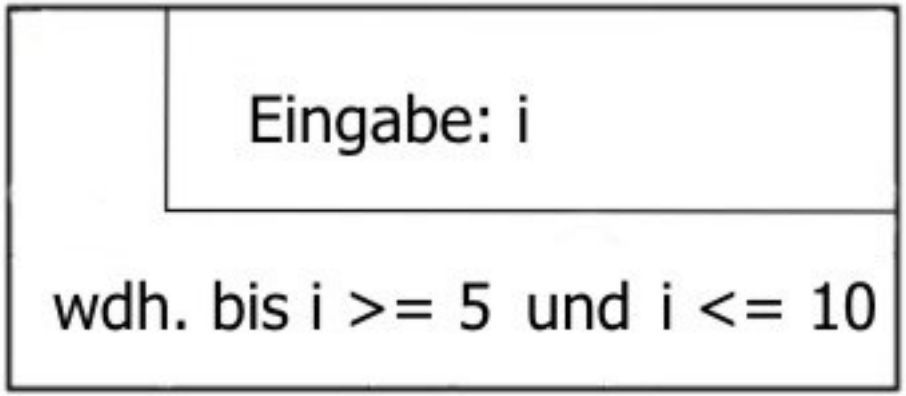
do/while -Schleife

- In folgendem Beispiel wird die Benutzereingabe wiederholt, bis der eingegebene Wert im Intervall [5, 10] liegt.

```
int i;
do
    cin >> i;
while (i < 5 || i > 10);
```

do/while -Schleife

- Das zugehörige Struktogramm sieht wie folgt aus:



Geschachtelte Schleifen

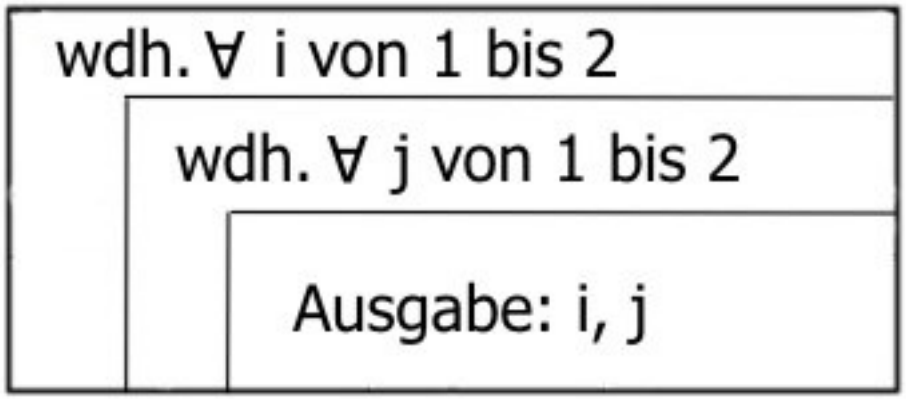
- Schleifen dürfen ineinander geschachtelt werden.

```
for (int i=1; i<=2; i++)
{
    for (int j=1; j<=2; j++)
    {
        cout << i << " " << j << endl;
    }
}
```

Ausgabe:
1 1
1 2
2 1
2 2

Geschachtelte Schleifen

- Das zugehörige Struktogramm sieht wie folgt aus:



break -Anweisung

- Die *break* –Anweisung wird verwendet, um eine Schleife unmittelbar zu verlassen.
- Bei verschachtelten Schleifen wird nur die aktuelle Schleife verlassen.

continue -Anweisung

- Die *continue* –Anweisung wird verwendet, um in einer Schleife unmittelbar die nächste Wiederholung zu beginnen.
- In *while* und *do/while* –Schleifen wird zur Schleifenbedingung gesprungen.
- In *for* –Schleifen wird zur Reinitialisierung gesprungen.

Funktion *exit*

- Durch Aufruf der *exit* –Funktion kann ein Programm sofort abgebrochen werden.
- Im Fehlerfall übergibt man einen Wert ungleich 0 und bei normaler Beendigung des Programms eine 0.

```
if (i == 3)
{
    cout << "Programm Ende" << endl;
    exit(0);
}
```

Umformungen

- Eine *do/while* –Schleife kann stets in eine *while* -Schleife umgeformt werden

do/while:

```
do
{
    Anweisung;
} while (Ausdruck);
```

while:

```
Anweisung;
while (Ausdruck) {
    Anweisung;
}
```

Umformungen

- Eine *for* –Schleife kann stets in eine *while* -Schleife umgeformt werden

```
for (Initialisierung; Abbruchbedingung; Reinitialisierung)
    Anweisung;
```

```
Initialisierung;
while (Abbruchbedingung) {
    Anweisung;
    Reinitialisierung;
}
```