TUHH
Technische Universität Hamburg-Harburg

Institute of Embedded Systems

Hello World!

Eine Einführung in das Programmieren

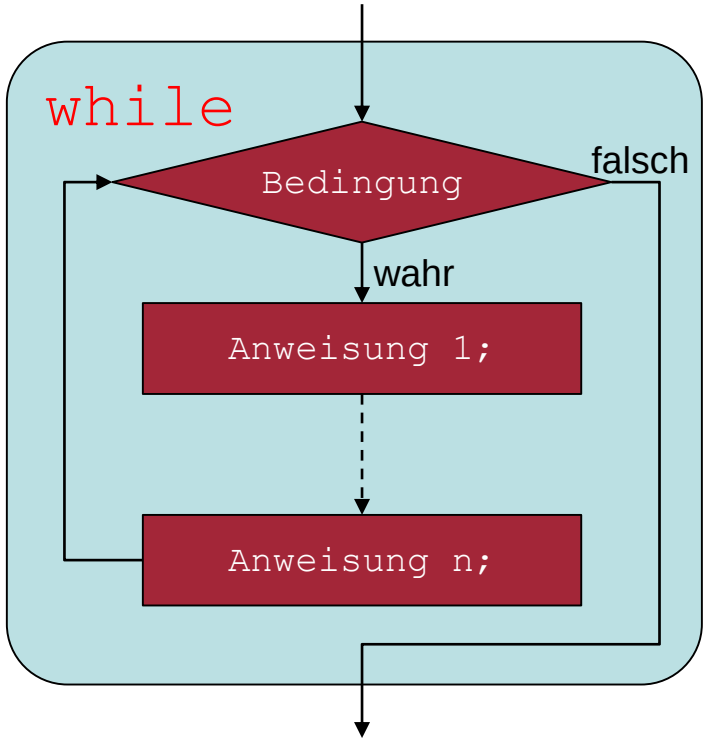
Schleifen

Görschwin Fey

Institute of Embedded Systems
Hamburg University of Technology

Die While-Schleife

```
while(Bedingung){
        Anweisung 1;
        …
        Anweisung n;
}
```



- Wird die `Bedingung` als wahr (true) ausgewertet, werden die Anweisungen ausgeführt
- Danach wird die Bedingung erneut überprüft
- Wenn die Bedingung falsch (false) wird, wird die Schleife verlassen

`while`-Schleifen können leicht zu endlosen Schleifen werden!!!

Endlosschleifen

Ein Programm, das in einer Endlosschleife „festhängt“, kann

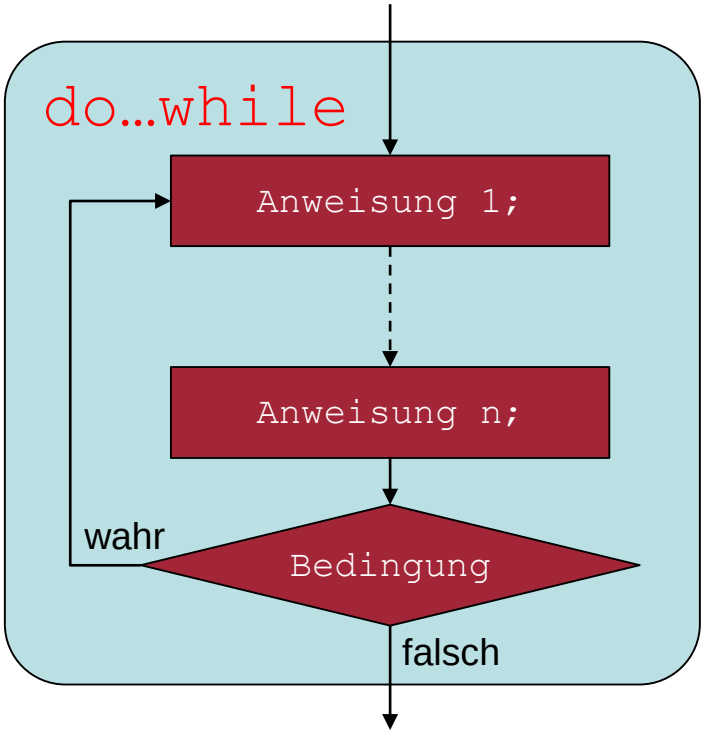
- unter Linux mit der Tastenkombination „STRG“ und „C“ oder durch Schließen des Terminalfensters,
- unter Windows durch Öffnen des Taskmanagers und Beenden des Tasks,
- in einer IDE durch einen „kill“- oder „stop“-Button beendet werden

Innerhalb einer `while`-Schleife muss an wenigstens einer Variablen, die in der Bedingung überprüft wird, etwas geändert werden, damit es überhaupt zu einem Abbruch der Schleife kommen kann!

```
while(a <= 213){
        a += 5;
}
```

Die Do-While-Schleife

```
do {
        Anweisung 1;
        …
        Anweisung n;
} while(Bedingung)
```

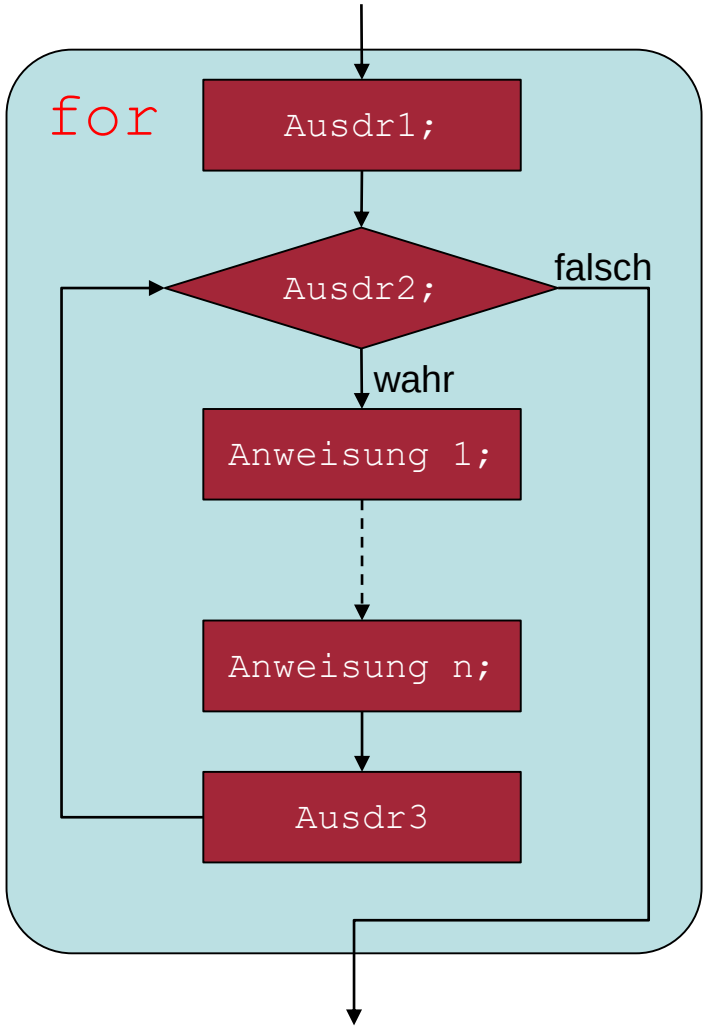


Auswertung der `Bedingung` nach der letzten Anweisung in der Sequenz

→ Die Anweisungen der Do-While-Schleife werden mindestens einmal ausgeführt!

Die For-Schleife

```
for(Ausdr1; Ausdr2; Ausdr3){
        Anweisung 1;
        …
        Anweisung n;
}
```

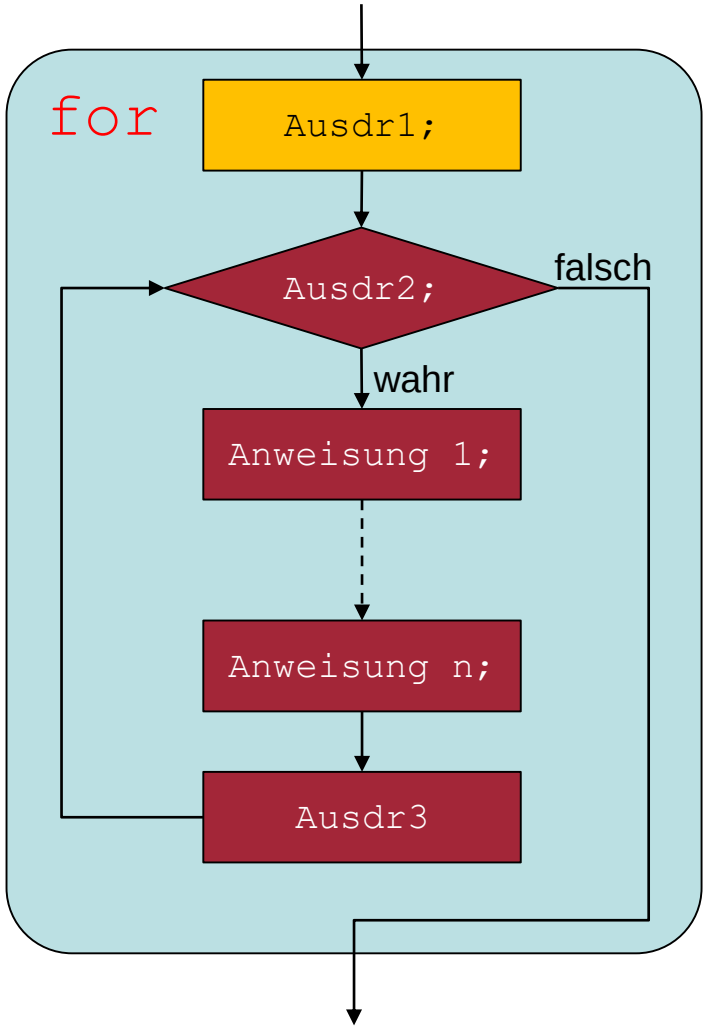


- Der erste Ausdruck (`Ausdr1`) wird einmalig zu Beginn des Ablaufs ausgeführt
- Der zweite Ausdruck (`Ausdr2`) wird vor jedem Anweisungsdurchlauf ausgewertet
- Der dritte Ausdruck (`Ausdr3`) wird am Ende des Anweisungsdurchlaufs ausgeführt

Initialisieren der For-Schleife

Beispiel:

$$\sum_{i=1}^{100} i = 1 + 2 + \cdots + 100 = ???$$



Der erste Ausdruck (`Ausdr1`) dient zur Initialisierung der For-Schleife.

Üblicherweise wird eine Zählvariable deklariert.

Beispiel: *int i = 1;*

Abbruchbedingung der For-Schleife

Beispiel:

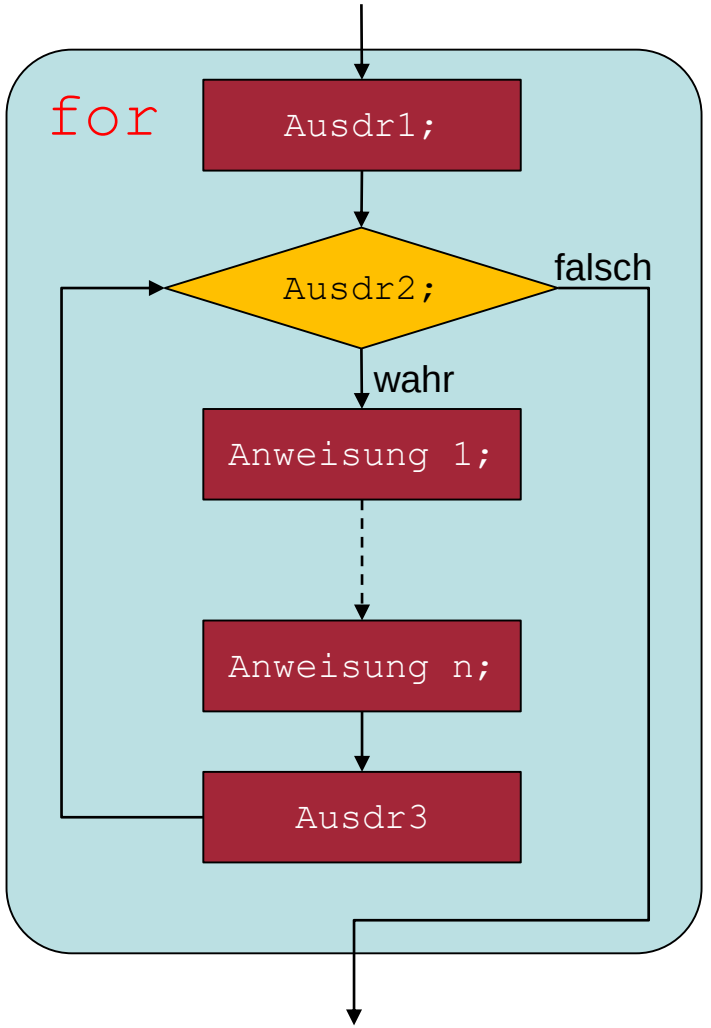
$$\sum_{i=1}^{100} i = 1 + 2 + \cdots + 100 = ???$$

Der zweite Ausdruck (`Ausdr2`) legt die Abbruchbedingung fest;

Der Ausdruck muss ein logisches Ergebnis liefern.

Ist der Ausdruck wahr, werden die Anweisungen der zugehörigen Sequenz ausgeführt.

Beispiel: *i <= 100;*



Reinitialisieren der For-Schleife

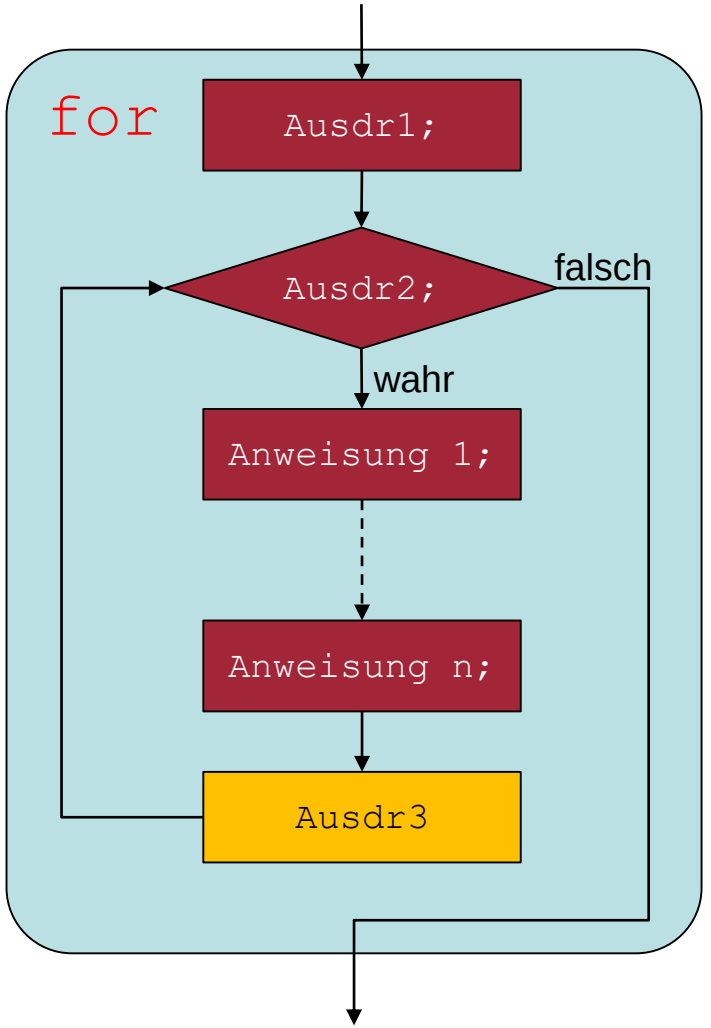
Beispiel:

$$\sum_{i=1}^{100} i = 1 + 2 + \cdots + 100 = ???$$

Der dritte Ausdruck (`Ausdr3`) reinitialisiert die Schleife.

Üblicherweise wird die Zählervariable für den nächsten Durchlauf angepasst.

Beispiel: *i++* *oder* *i+=1*



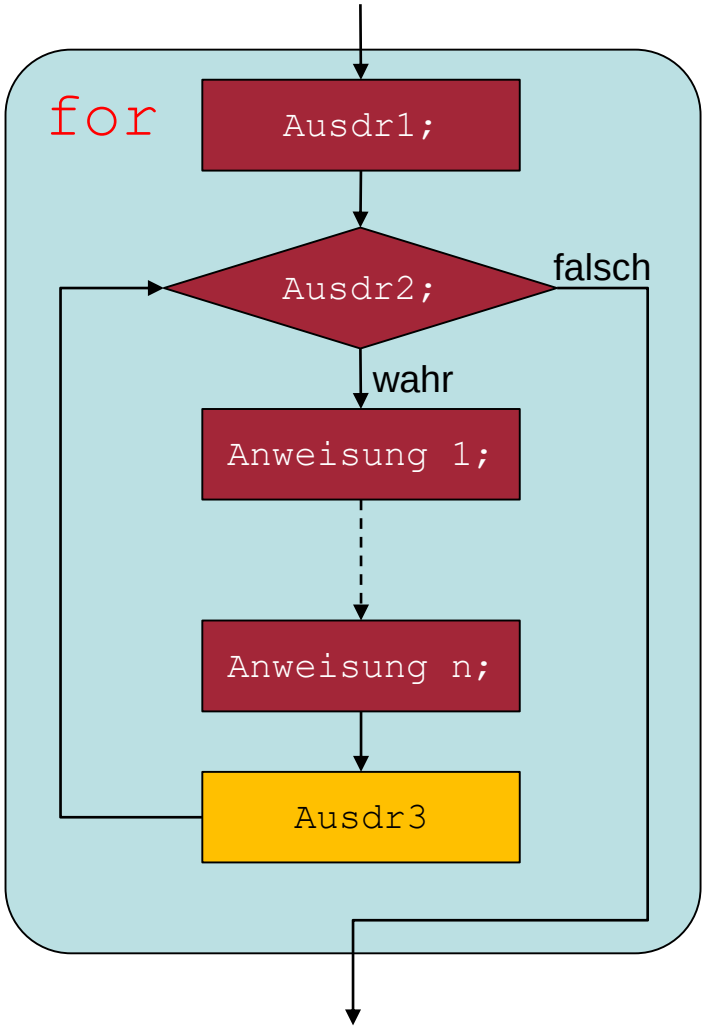
Reinitialisieren der For-Schleife

Beispiel:

$$\sum_{i=1}^{100} i = 1 + 2 + \cdots + 100 = ???$$

```
int summe = 0;

for(int i=1; i<=100; i++){
        summe += i;
}
```



Nach Durchlaufen dieser For-Schleife enthält die Variable `summe` das Ergebnis der Formel

Beispiel: Geschachtelte Schleife – Das kleine Einmaleins

```
for(int i=1; i<=10; i++){
      for(int j=1; j<=10; j++){
            cout << i * j << " ";
      }
      cout << endl;
}
```

Für jedes i = {1,2, … , 10} der äußeren `for`-Schleife durchläuft die innere `for`-Schleife ebenfalls alle Zahlen von 1 bis 10

Für jede „Kombination“ wird das Produkt von `i` und `j` mit einem nachgestellten Leerzeichen ausgegeben

Nach jeder Zahlenreihe wird eine neue Zeile begonnen

Potenzial für Fehler!

```
int size=5;

for (int i=0; i++; i<size) {
   cout <<"Eins: " <<i <<endl;
}

for (int i=0; i=size; i+=2) {
   cout <<"Zwei: " <<i <<endl;
}

for (int i=0; i!=size; i+=2) {
   cout <<"Drei: " <<i <<endl;
}
```

Manchmal hilft der Compiler

- Compilieren mit Option `-Wall`

```
g++ -Wall t.cpp
```

- Liefert

```
t.cpp: In function ‘int main()’:
t.cpp:29:25: warning: for increment expression has no
effect [-Wunused-value]
     for (int i=0; i++; i<size) {
                        ^
t.cpp:32:21: warning: suggest parentheses around
assignment used as truth value [-Wparentheses]
     for (int i=0; i=size; i+=2) {
```

Break und Continue

Ablauf jeder der gezeigten Schleifen kann beeinflusst werden:

- Die `break`-Anweisung bricht die Ausführung der Schleife sofort ab. Die Schleife wird verlassen.
- Die `continue`-Anweisung bricht den aktuellen Durchlauf sofort ab. Die Bedingung wird daraufhin überprüft und ggf. ein neuer Durchlauf begonnen.

Sonderfall: Eine einzelne Anweisung

- Für eine Kontrollstruktur, die nur eine Anweisung enthält, sind die geschweiften Klammern optional

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

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

Deshalb funktioniert auch das `else if`:

```
if(a == 3) {
        …
} else {
        if(b >= 4) {
                …
        }
}
```

```
if(a == 3) {
        …
} else if(b >= 4) {
        …
}
```

Sonderfall: Eine einzelne Anweisung

- Für eine Kontrollstruktur, die nur eine Anweisung enthält, sind die geschweiften Klammern optional

- Trotzdem hat jede Kontrollstruktur ihren eigenen Speicherbereich
- Variablen, die in einer Kontrollstruktur deklariert werden, stehen nach dem Verlassen nicht mehr zur Verfügung

```
if(a == 3) int b = 4;
cout << b;  // Fehler
```

`b` ist nicht deklariert

```
int b = 3;
if(a == 3) b = 4;
cout << b; // OK, b ist 4
```

Zusammenfassung

- `if(…) { … } [ else { … } ];`

- `while(…) { … };`
- `do { … } while(…);`
- `for(…; …; …) { … };`

Übung: Nochmal und nochmal und …

1. Zahlenreihen und ihre Summen:
    1. Berechnen Sie die Summe der Zahlen von 1 bis 2018.
    2. Wie lautet die Summe der Zahlen aus der Zahlenreihe 3,6,9,…,999?
2. Das „Kleine 1 mal 1“ im Terminal:
    1. Geben Sie das „1 mal 6“ in einer Zeile im Terminal aus.
    2. Fragen Sie den Benutzer für welche Zahlen er die Ausgabe möchte.
    3. Geben Sie das gesamte „1 mal 1“ im Terminal aus.
3. Eine Primzahl kann nur durch sich selbst oder durch 1 geteilt werden:
    1. Benutzen Sie den Rest der Division (Modulo %) um zu überprüfen, ob eine Zahl a durch eine andere Zahl b teilbar ist.
    2. Überprüfen Sie mit einer Schleife, ob die beliebige Zahl x eine Primzahl ist. Führen Sie die Prüfung in einer Schleife durch!
    3. Geben Sie die ersten 100 Primzahlen im Terminal aus.