

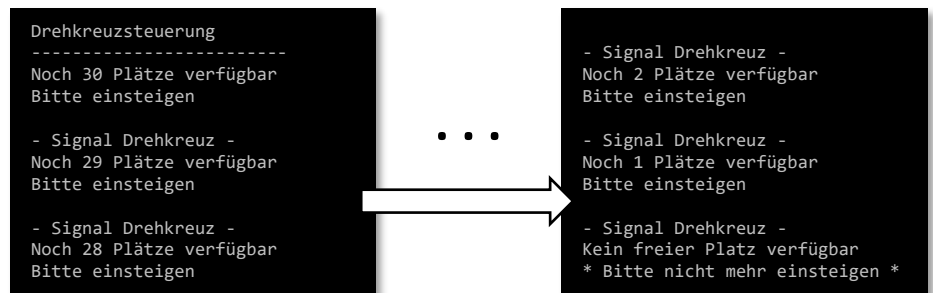
LS 2.5 Drehkreuzsteuerung

Herr Schäring (Abteilungsleiter Software) meldet sich mit folgendem Anliegen:



Liebe Kollegen,
wir wurden beauftragt eine sinnvolle Drehkreuzsteuerung zu programmieren. Hierbei soll das Drehkreuz den Zutritt nur für 30 Personen zulassen. Im Anhang dieser Mail finden Sie die im Lastenheft spezifizierte Ausgabe, welche unser Programm liefern soll. Viele Grüße!

Anforderungen des Kunden:



Schleifen führen Anweisungen so oft aus, solange die Bedingung dafür true ergibt. Ist vor Schleifeneintritt bereits die Anzahl der Durchläufe bekannt, ist es ratsam eine Zählergesteuerte Schleife zu nutzen.

Kopf- und Zählergesteuerte Schleife: for

Eine for -Schleife ist eine Kontrollstruktur, mit der Anweisungen für eine bestimmte Anzahl an Durchläufen wiederholt werden können. Zum Zählen dieser Durchläufe wird eine Variable benötigt. In der Regel benennt man diese Zählvariable lediglich mit i, j, k, usw.

Syntax in C#

```

for ( Initialisierung ; Ausführungsbedingung ; Reinitialisierung )
{
    Anweisungsblock;
}
    
```

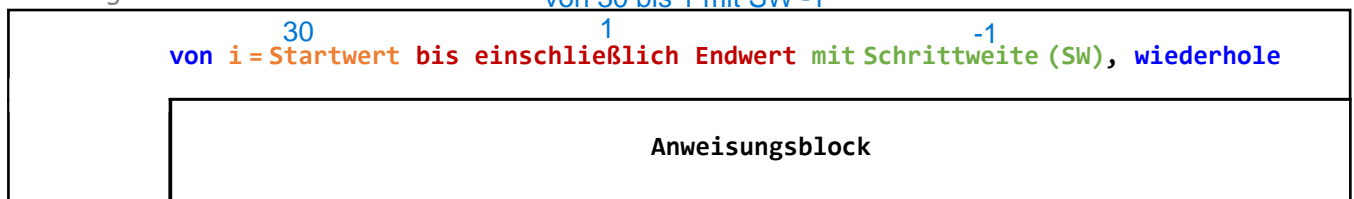
Vergleichsvariable erstellen
z.B. int i = 0

Nur bei true wird der Anweisungsblock ausgeführt
z.B. i < 30

Veränderung am Ende des Schleifendurchlaufs
z.B. i++

Struktogramm nach DIN 66261

von 30 bis 1 mit SW -1



Initialisierung

Hier wird der Startwert festgelegt. Wird diese Variable nur zum Zählen innerhalb der Schleife benötigt, wird sie im Schleifenkopf sowohl definiert als auch ein Wert zugewiesen (z. B.: `int i = 0`).

Der Startwert `i` sollte nach Möglichkeit immer im Schleifenkopf definiert werden. Somit ist sichergestellt, dass `i` nur innerhalb der Schleife zur Verfügung steht. Eine Definition vor der Schleife wird nur dann verwendet, wenn dieser Wert zu einem späteren Zeitpunkt erneut benötigt wird.

Ausführungsbedingung

Die Ausführungsbedingung entscheidet ob die Schleife ausgeführt wird. Liefert die Ausführungsbedingung den Wahrheitswert `true` werden die folgenden Anweisungen ausgeführt. Wird an dieser Stelle jedoch der Wahrheitswert `false` geliefert, wird der Anweisungsblock übersprungen.

Reinitialisierung

Bei der Reinitialisierung wird festgelegt um welchen Wert der Startwert pro Schleifendurchlauf verändert wird. Die gängigsten Anwendungen sind das Hochzählen um die Schrittweite 1 (`i = i + 1`) bzw. das entsprechende Herunterzählen (`i = i - 1`).

Beispiel für Kopf- und Zählergesteuerte Schleifen

```
for (int i = 0; i < 20; i++)  
{  
    Console.WriteLine(i);  
    Console.WriteLine(' ');  
}
```

von i = 0 bis 19 mit SW 1, wiederhole

Ausgabe: i
Ausgabe: Leerzeichen

0. Initialisierung → 1. Ausführungsbedingung ← 3. Reinitialisierung
2. Anweisungsblock

LS 2.5.1 Zeichnen Sie ein Struktogramm zur Drehkreuzsteuerung nach der oben angegebenen Ausgabe.

von i = 30 bis 1 mit SW -1, wiederhole

Ausgabe: noch i Plätze frei
Ausgabe: Bitte einsteigen
Eingabe: Taste, bzw. Signal des Drehkreuzes
Ausgabe: Kein freier Platz mehr
Ausgabe: Bitte nicht mehr einsteigen


LS 2.5.2 Schreiben Sie ein Programm in C# zur Drehkreuzsteuerung, welches Ihr oben stehendes Struktogramm entsprechend umsetzt.

LS 2.5.3 Wandeln Sie die im Programm verwendete for -Schleife in eine while -Schleife um.

LS 2.5.4 Weitere Übungsaufgaben zu kopf- und zählergesteuerte Schleifen

1. Schreiben Sie den Programmcode um alle positiven geraden Zahlen von 0 bis zu einem einzulesenden Endwert `i_Max` anzuzeigen (0, 2, 4, ... n). Nutzen Sie dazu eine for -Schleife.

```
int i_Max = 0;
```



Beispiel: 15

0
2
4
6
8
10
12
14

2. Schreiben Sie den Programmcode um alle positiven geraden Zahlen von 0 bis zu einem einzulesenden **ungeraden** Maximalwert auszugeben. Gleichzeitig soll in jeder Zeile eine Zahl (beginnend ab dem Maximalwert und absteigend) ausgegeben werden.

```
int i_Max = 0;
```

Beispiel: 15

0 15
2 14
4 13
6 12
8 11
10 10
12 9
14 8

3. Schreiben Sie den Programmcode zur Ausgabe der rechts stehenden mathematischen Reihe. Die Werte für `x` und `n` werden eingelesen, wobei für `x` Kommawerte zulässig sind und `n` ganzzahlig ist.

$$\begin{aligned} s(0) &= x^0 \\ s(1) &= x^0 + x^1 \\ \dots \\ s(n) &= x^0 + x^1 + x^2 + x^3 + \dots + x^n \end{aligned}$$

Beispiel:
n=5; x=15.1

1
16.1
244.11
3687.06
55675.6
840703

LS 2.5.5 Anwendungsbeispiele von for-Schleifen (Übung)



Welche Werte stehen nach Beendigung der for -Schleife auf dem Bildschirm?

```
for(int i = 0; i < 11; i = i + 2)
{
    Console.Write(i + " ");
}
```

0 2 4 6 8 10

```
for(int n = 2; n <= 300; n *= n)
{
    Console.Write(n + " ");
}
```

2 4 16 256

```
for(int i = 0, a = 10; i < 5; i++, a--)
{
    Console.Write(i + " - " + a + " ");
}
```

0-10 1-9 2-8 3-7 4-6

```
for(float zs = 100f; zs < 150; zs = zs * 1.1f)
{
    Console.Write(zs + " ");
}
```

100 101 121 133,1 146,41

```
for(int i = 12; i > 2; i -= 2)
{
    if(i % 3 == 0) Console.Write("Yeha ");
    else Console.Write("Nope ");
}
```

Yeha Nope Nope Yeha Nope

```
for(int i = 97; i < 't'; i = i + 4)
{
    Console.Write((char)i + " ");
}
```

a e i m q

```
for(int i = 23001; i > 1000; i /= 2)
{
    Console.Write(i + ": " + i%2 + " ");
}
```

23001: 1 11500: 0 5250: 0 2125: 1
1062: 1

Lösung:



```
23001: 1 11500: 0 5250: 0 2125: 1 1062: 1
a e i m q
Yeha Nope Nope Yeha Nope
100 101 121 133,1 146,41
0-10 1-9 2-8 3-7 4-6
2 4 6 8 10
```