While-Schleifen in C#

Neben den for-Schleifen, die vor allem zum Zählen verwendet werden, gibt es in C# noch weitere Schleifentypen, wie z. B. die while-Schleife.

Nehmen wir ein Beispiel aus der Küche. Teig rühren ist ein Vorgang, der wiederholt wird. Bei einer for-Schleife könnten wir im Kochrezept nur sagen: „Rühre den Teig 20 Mal."
Mittels einer while-Schleife sind jetzt Formulierungen möglich wie z. B.: „Rühre den Teig, bis er schaumig ist."

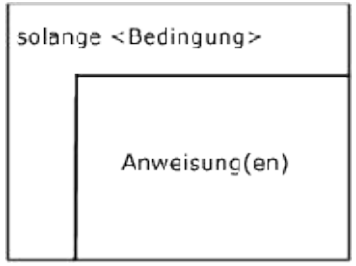
Bei der while-Schleife wird am Schleifenanfang eine Bedingung auf ihren Wahrheitsgehalt hin überprüft.

SOLANGE *Bedingung* FÜHRE AUS
Anweisung 1 ⎫
Anweisung 2 ⎬ Schleifenrumpf
.... ⎭
Anweisung n

Wenn die Ausführungsbedingung WAHR (true) ist, wird der Schleifenrumpf betreten und die Anweisung(en) wird (werden) ausgeführt. Ist die Ausführungsbedingung FALSCH (false), so wird der Schleifenrumpf nicht betreten und das Programm mit der dahinter folgenden Anweisung fortgesetzt. Es kann demnach sein, daß die while-Schleife kein einziges Mal durchlaufen wird. Man nennt daher die while-Schleife eine **abweisende Schleife**.

Wichtig: Man muss unbedingt darauf achten, dass die Schleife auch tatsächlich nach einer endlichen Anzahl von Durchläufen verlassen wird und keine Endlosschleife entsteht. Dies bedeutet, dass der Wahrheitswert der Ausführungsbedingung nach endlich vielen Durchläufen den Wert FALSCH annehmen muss. Erreicht wird das durch eine entsprechende Anweisung im Schleifenrumpf.

Struktogramm:



Formulierung in C#: `while (Bedingung) Anweisung;`

Ist die Anweisung aus mehreren Einzelanweisungen zusammengesetzt, so müssen. diese durch { und } zusammengefasst werden:

```
while (Bedingung)
{
    Anweisung1;
    Anweisung2;
}
```

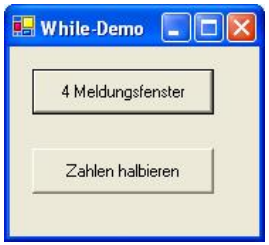

Beispiele:

```
void Button1Click(object sender, System.EventArgs e)
{
   int i=1;  // Initialisierung
   // Es werden 4 Meldungsfenster ausgegeben:
   while(i<=4){  // solange i<4 ist, führe Schleife aus
     MessageBox.Show("Hallo aus der Schleife!");

     // wichtig: i muss erhöht werden, sonst kein Schleifenende
     i++;
   }
}

void Button2Click(object sender, System.EventArgs e)
{
   int x=20;

   // Schleife
   while(x>1){
     MessageBox.Show(x.ToString());
      x=x/2;
   }
}
```



Wenn man den Button „Zahlen halbieren" klickt, werden die Zahlen 20,10,5,2 in einem Meldungsfenster aus.

Wenn in dem Beispiel die Zeile x=x/2 fehlen würde, so würde die Schleife endlos wiederholt. Wäre die Initialisierung x=1 anstelle von x=20, so würde die Schleife kein einziges Mal durchlaufen.

Aufgaben:

1. Schreibe ein Programm, dass die Zahlen 30, 27, 24, 21, 18, 15, 12, 9, 6, 3 in einer Listbox ausgibt. Orientiere dich dabei an der Ereignisbehandlung von Button2.
2. Die Summe aller ganzen Zahlen zwischen den ganzen Zahlen "AnfZahl" und "EndZahl" (beide sind einzugeben) einschließlich dieser beiden Werte soll mit Hilfe einer while-Schleife berechnet werden.
3. Programmiere die Aufgabe 5 vom letzten Aufgabenblatt (for-Schleifen Übungen) mit einer while-Schleife. Diesmal muss die Verdopplungszeit nicht vor der Ausgabe berechnet werden, sondern es kann während der while-Schleife geprüft werden, ob sich das Guthaben verdoppelt.
4. Berechne wie in Aufgabe 2 das Produkt aller ganzen Zahlen von einem einzugebenden Anfangswert bis zu einem Endwert.
5. Der Computer soll nach Eingabe einer natürlichen Zahl kleiner als 20 (Kontrolle durch das Programm!) die nächsten 15 natürlichen Zahlen zusammen mit ihrem Quadrat in einer Listbox ausgeben. Erstelle ein Struktogramm und ein C#-Programm!
6. Es soll geprüft werden, welche Zahlen x in einem vorgegebenen Bereich die Ungleichung $|x - 7| < 3$ erfüllen.