LS 2.2 Situation Taschenrechner

Herr Schäring schreibt Ihnen:
Unser neuer Praktikant braucht Hilfe beim coden und ich muss dringend in eine Videokonferenz. Trotz EVA-Prinzip und ordentlichen Kommentaren bekommt er sein Programm nicht zum Laufen. Den Code und eine Beispielausgabe finden Sie im Anhang. Mit Ihren C#-Kenntnissen können Sie ihm bestimmt helfen!
Danke und VG Schäring

LS 2.2.1 Code-Review des Praktikanten-Taschenrechners

Codequelle: Teams/AWP10/C#/Schülervorlagen/PraktikantenCode.cs

```
6   class Program
7   {
8       static void Main(string[] args)
9       {
10          double zahl1 = 0.0;
11          double zahl2 = 0.0;
12          double ergebnis = 0.0;                          //Variablen für Zahlen
13          char rechenzeichen;                             //Variable für Rechenzeichen
14
15          Console.Write("Geben Sie die erste Zahl ein: ");    //Eingabeaufforderung ausgeben
16          zahl1 = Convert.ToDouble(Console.ReadLine());
17
18          Console.Write("Geben Sie das Rechenzeichen ein: "); //Eingabeaufforderung ausgeben
19          rechenzeichen = Convert.ToChar(Console.ReadLine());
20
21          Console.Write("Geben Sie die zweite Zahl ein: ");   //Eingabeaufforderung ausgeben
22          zahl2 = Convert.ToDouble(Console.ReadLine());       //Eingabe einlesen
23
24          if (rechenzeichen == +)                         //je nach Operator berechnen
25              ergebnis = zahl1 + zahl2;
26          if (rechenzeichen == -)
27              ergebnis = zahl1 - zahl2;
28          if (rechenzeichen == +)
29              ergebnis = zahl1 * zahl2;
30          if (rechenzeichen == /)
31              ergebnis = zahl1 / zahl2;
32
33          //Aufgabe noch mal komplett ausgeben
34          Console.WriteLine($"{zahl1} {rechenzeichen} {zahl2} = {ergebnis}");
35
36          Console.ReadKey();
37      }
38  }
```

Beispielausgabe:

```
Geben Sie die erste Zahl ein: 27
Geben Sie das Rechenzeichen ein: /
Geben Sie die zweite Zahl ein: 3
27 / 3 = 9
```

Handlungsaufträge:

1.) Warum kompiliert das Programm nicht?
2.) Wie könnte man alle Variablen eines Typs mit einer Code-Zeile definieren?
3.) Was passiert, wenn Sie eine Modulo-Berechnung eingeben? (bspw: **`2000 % 4`**)
4.) Erweitern Sie das Programm, sodass eine korrekte Modulo-Rechnung möglich ist.

LS 2.3 Erweiterung des Taschenrechners (Dauerbetrieb)

Der Praktikant ist verärgert, da er für jeden Berechnungstest das Programm immer wieder neu starten muss. Wie kann er sein Programm in **Dauerschleife** ausführen?

LS 2.3.1 Wiederholungen in C#

Bisher wurden Ihre Programme lediglich der Reihe nach abgearbeitet und u. U. verschiedenen Zweigen gefolgt. Mit Wiederholungen, sogenannten **Schleifen**, können Sie erreichen, dass Programmabschnitte mehrfach abgearbeitet werden können.

Ihnen stehen bei C# zwei Arten von Schleifen zur Verfügung:

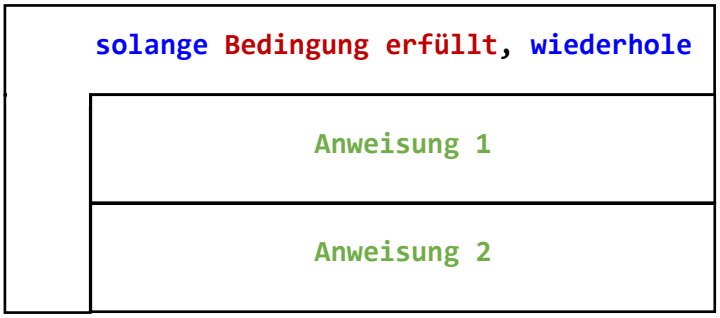
Kopfgesteuerte Schleife: while (bereits bekannt aus RobotC)

Kopfgesteuerte Wiederholungen haben Ihren Kontrollpunkt, also ihre **Ausführungsbedingung**, noch vor dem ersten Schleifendurchlauf, demnach vor dem Eintritt in den ***Schleifenrumpf***.

Syntax in C#

```
while (Bedingung)
{
      Anweisung 1;   } Schleifenrumpf
      Anweisung 2;   } bzw. Schleifenkörper
}
```

Struktogramm nach DIN 66261

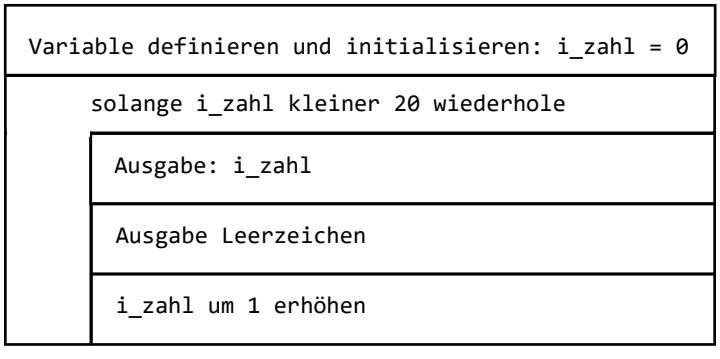


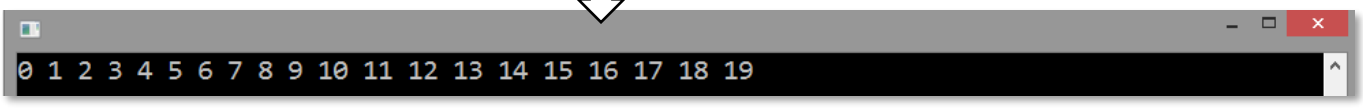
Vor Eintritt in die **Schleife** wird geprüft, ob die **Ausführungsbedingung** erfüllt ist - also der entsprechende Ausdruck einen Wert **true** hat. Ist dies der Fall, so werden die darauffolgenden **Anweisungen** ausgeführt. Anschließend wird die **Bedingung** erneut geprüft. Sobald der Bedingungsausdruck dem **Wert false** entspricht, wird das Programm mit der nächsten Anweisung **nach** der Schleife fortgesetzt.

Die kopfgesteuerte Schleife wird entweder einmal, mehrmals oder gar nicht ausgeführt.

Beispiel für kopfgesteuerte Schleifen

```
int i_zahl = 0;

while (i_zahl < 20)
{
      Console.Write(i_zahl);
      Console.Write(' ');
      i_zahl++;
}
```





Fußgesteuerte Schleife: do … while

Fußgesteuerte Wiederholungen haben Ihren Kontrollpunkt, also die **Ausführungsbedingung**, nach dem ersten Schleifendurchlauf, sprich im Anschluss zum ***Schleifenrumpf***.

Syntax in C++

```
do
{
        Anweisung 1;
        Anweisung 2;
}while (Bedingung);
```

Schleifenrumpf bzw. Schleifenkörper

Semikolon nicht vergessen

Struktogramm nach DIN 66261

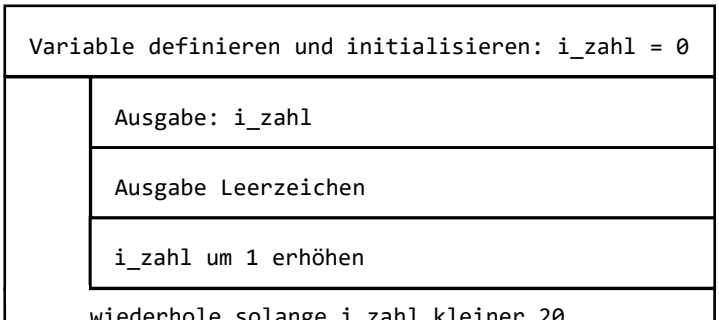
Fußgesteuerte Schleifen werden folgendermaßen abgearbeitet:

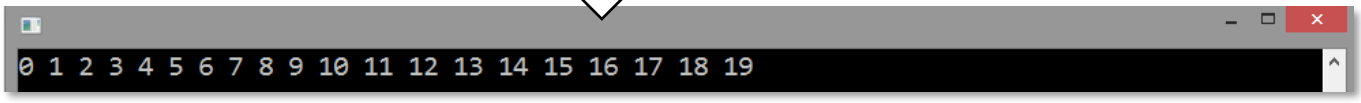
1. Zuerst werden die Anweisungen des ***Schleifenrumpfs*** ausgeführt.
2. Danach wird die **Bedingung** geprüft.
3. Ist der Wert des Bedingungsausdrucks **`true`**, wird die **Schleife** erneut durchlaufen.
   Sobald der Bedingungsausdruck dem **`false`** entspricht, wird das Programm mit der nächsten Anweisung **nach** der Schleife fortgesetzt.

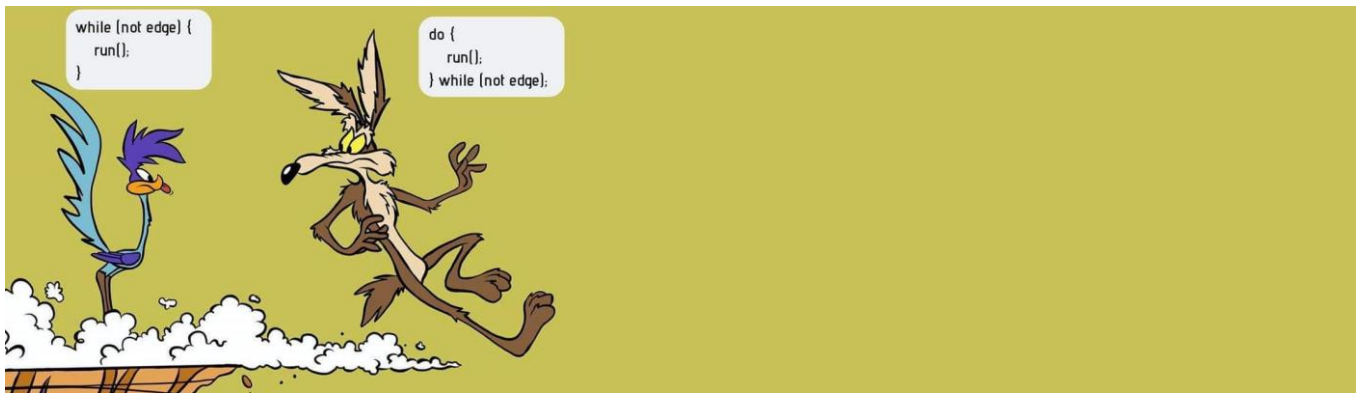
Die fußgesteuerte Schleife wird mindestens einmal oder mehrmals durchlaufen.

Beispiel für fußgesteuerte Schleifen

```
int i_zahl = 0;

do
{
        Console.Write(i_zahl);
        Console.Write(' ');
        i_zahl++;
}while (i_zahl < 20);
```







LS 2.3.3 Planung: Festlegen der Wiederholungsbedingung

Der Taschenrechner soll nur wiederholen, wenn das Zeichen j oder J nach einer Rechnung eingelesen wurde. Was benötigen Sie dazu?

```
Geben Sie die erste Zahl ein: 27
Geben Sie das Rechenzeichen ein: /
Geben Sie die zweite Zahl ein: 3
27 / 3 = 9
Noch einmal rechnen mit j bzw. J: _
```

LS 2.3.4 Implementierung des Dauerbetriebes

Ergänzen Sie die Funktionalität eines Dauerbetriebes für das Taschenrechnerprogramm. Achten Sie auf eine saubere Reinitialisierung der Variablen und darauf, dass sowohl das j als auch das J zu einer Wiederholung führen müssen. Legen Sie hierfür bitte ein neues Projekt an → *TaschenrechnerDauerbetrieb*

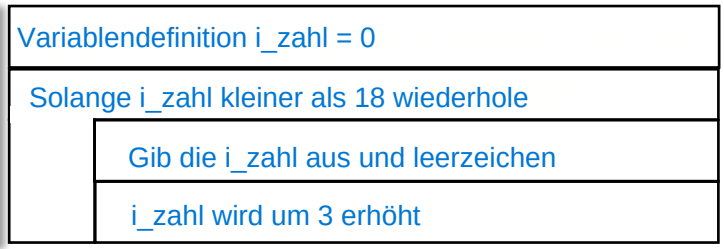
Testen Sie das Programm ausführlich.

LS 2.3.5 Weitere Übungsaufgaben zu Wiederholungen

Aufgabe 1

a) Interpretieren Sie den Quellcode und ergänzen Sie dazu das nebenstehende Struktogramm.
b) Geben Sie anschließend die entsprechende Ausgabe des Programms in der Konsole unten an.
c) Formen Sie die im Quellcode angegebene `while`-Schleife in eine `do-while`-Schleife um.

```
int i_zahl = 0;
while(i_zahl < 18)
{
      Console.Write(i_zahl + " ");
      i_zahl = i_zahl + 3;
}
```



do-while Version:

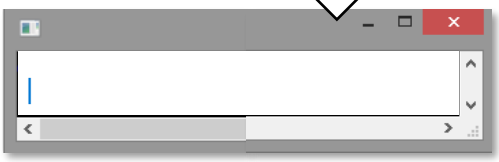
```
int i_zahl = 0;

do
{
Console.Write(i_zahl + " ");
i_zahl = i_zahl +3;
}

while (i_zahl < 18);
```

Aufgabe 2

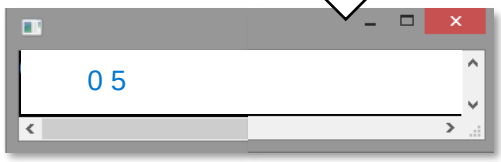
Ermitteln Sie die Konsolenausgaben beider Schleifenvarianten und begründen Sie die Unterschiede.

```
int i_zahl = 0;
while ((i_zahl > 0) && (i_zahl < 10))
{
      Console.Write(i_zahl + " ");
      i_zahl = i_zahl + 5;
}
```

```
int i_zahl = 0;
do
{
      Console.Write(i_zahl + " ");
      i_zahl = i_zahl + 5;
}while ((i_zahl > 0) && (i_zahl < 10));
```



Zahl muss zwischen 1 und 9 sein



Aufgabe 3

Erstellen Sie ein Programm, das den Wert einer eingelesenen Dezimalzahl im dualen System ausgibt. Das niederwertigste Bit soll zuerst ausgegeben werden, gefolgt von den höherwertigen Bits.

Bsp.: $110_{10} \rightarrow ?_2$

```
110 : 2 = 55  Rest 0
55  : 2 = 27  Rest 1
27  : 2 = 13  Rest 1
13  : 2 = 6   Rest 1
6   : 2 = 3   Rest 0
3   : 2 = 1   Rest 1
1   : 2 = 0   Rest 1
```

Ergebnis *der ganzzahligen Division durch* ***2***

Rest *der ganzzahligen Division durch* ***2***

Leserichtung

Ergebnis: $110_{10} \rightarrow 1101110_2$ 1101110

Hinweis:
Es genügt, die Dualzahl in umgekehrter Reihenfolge auszugeben.

Die Ausgabe des Beispiels lautet: