ZUSAMMENFASSUNG Schleifen

WAS SIND SCHLEIFEN?

- Mit Schleifen können wir bestimmten Code wiederholt ausführen
- Sie eignen sich zum Durchlaufen von Datenstrukturen (z. B. Arrays)
- Sie ermöglichen das Ausführen von Codeblocks bis zur Erfüllung einer Bedingung
- Es gibt verschiedene Arten von Schleifen
  (While, Do-While, For, Foreach)

DIE WHILE-SCHLEIFE

Die While-Schleife ist **kopfgesteuert** und wird so lange ausgeführt, bis die Bedingung nicht mehr erfüllt wird. Auf der folgenden Abbildung ein Beispiel:

```
int zahl = 0;

while(zahl < 5) //Solange "zahl" kleiner als 5 ist
{
    zahl += 1; //Erhöhe "zahl" um 1
    Console.WriteLine(zahl); //und gebe "zahl" aus
}

Console.ReadKey();
```

Abbildung 1, While-Schleife

Abbildung 2, Output

DIE DO-WHILE-SCHLEIFE

Die Do-While-Schleife ist **fußgesteuert** und wird unabhängig von der Bedingung **mindestens einmal** ausgeführt, da die Bedingung erst nach dem Codeblock geprüft wird. **Beispiel:**

```
int zahl = 0;

do //Betrete den Codeblock der Schleife
{
    zahl += 1; //Erhöhe "zahl" um 1
    Console.WriteLine(zahl); //Und gebe "zahl" aus
}
while (zahl < 5); //Wenn die Bedingung erfüllt wird, wiederhole den Block

Console.ReadKey();
```

Abbildung 3, Do-While-Schleife

Abbildung 4, Output

DIE WHILE-SCHLEIFE IN DER PRAXIS

Auf der folgenden Abbildung sehen wir die While-Schleife nun einmal in der Praxis. In unserem Beispiel lassen wir den Benutzer zuerst eine Zahl eingeben und direkt im Anschluss eine zweite. Nun sollen sowohl beide dieser Zahlen als auch die Zahlen, die zwischen diesen liegen, ausgegeben werden.
Beispiel: Der Benutzer gibt zuerst die Zahl 3 an und in der nächsten Zeile die Zahl 7. Ausgegeben werden sollen also die Zahlen 3, 4, 5, 6 und 7. Umsetzen können wir das nun mit der While-Schleife:

```
{
    Console.Write("Gebe Integer 1 an: ");
    int zahl1 = Convert.ToInt32(Console.ReadLine());

    Console.Write("Gebe Integer 2 an: ");
    int zahl2 = Convert.ToInt32(Console.ReadLine());

    while (zahl1 <= zahl2)
    {
        Console.WriteLine(zahl1);
        zahl1 += 1;
    }

    Console.ReadKey();
}
```

Abbildung 5, die While-Schleife in einem Praxisbeispiel

Ausgegeben wird nun folgendes:

```
Gebe Integer 1 an: 3
Gebe Integer 2 an: 7
3
4
5
6
7
```

Abbildung 6, Output

DIE DO-WHILE-SCHLEIFE IN DER PRAXIS

Im nächsten Beispiel (Abbildung 7) werden wir eine Passwortabfrage programmieren. Zuerst legen wir ein Passwort fest. Wird dieses nun vom Benutzer eingegeben, lassen wir den Text ausgeben „Du hast das Passwort erraten!“. Gibt er hingegen ein falsches Passwort ein, wird er erneut aufgefordert ein Passwort einzugeben. Dieser Vorgang geschieht so lange, bis er das richtige Passwort eingegeben hat.

```
{
    string password = "HelloWorld";
    string input = "";

    do
    {
        Console.Write("Gebe das Passwort ein: ");
        input = Console.ReadLine();
    } while (password != input);

    Console.WriteLine("Du hast das Passwort erraten!");
    Console.ReadKey();
}
```

Abbildung 7, Do-While-Schleife in einem Praxisbeispiel

Die Ausgabe kann abhängig davon, was der Benutzer eingibt, etwa so aussehen:

```
Gebe das Passwort ein: 123
Gebe das Passwort ein: eeeee
Gebe das Passwort ein: iii
Gebe das Passwort ein: HelloWorld
Du hast das Passwort erraten!
```

Abbildung 8, Output

DIE FOR-SCHLEIFE

- Die For-Schleife ist eine Zählerschleife
- Sie besitzt eine Zählervariable, die man im Codeblock nutzen kann
- Sie eignet sich gut zum Durchlaufen von Arrays und anderen Datenstrukturen

Beispiel:

```
for (int i = 0; i < 10; i++)
{
    Console.WriteLine(i);
}
```

Abbildung 9, Beispiel einer For-Schleife

Erläuterung:

int i = 0;	Initialisiere Zählervariable
i < 10;	Bedingung zum Weitermachen
i++	Erhöhe i um 1

Ablauf:

1. Die Zählervariable i wird mit dem Wert 0 initialisiert
2. Der Codeblock wird einmal ausgeführt: 0 wird also als erste Zahl ausgegeben
3. Nun wird mit i++ zur Zählervariable 1 addiert
4. Danach wird die Bedingung überprüft (ist i kleiner als 10?)
5. Da die Bedingung noch erfüllt wird, (weil i jetzt 1 ist) wird der Codeblock der Schleife nochmals ausgeführt – Die 1 wird ausgegeben
6. Im nächsten Schritt wird i wieder um 1 erhöht und direkt danach wird wieder die Bedingung überprüft
7. Sobald die Bedingung nicht mehr erfüllt wird, d. h. sobald die Zählervariable 10 oder größer als 10 ist, wird der Codeblock der Schleife **nicht** mehr ausgeführt

Ausgegeben wird:

0
1
2
3
4
5
6
7
8
9

DIE FOR-SCHLEIFE IN DER PRAXIS

Im ersten Beispiel werden wir ein Array mit einer For-Schleife durchlaufen lassen. Dazu erstellen wir ein Array mit mehreren Namen. Die Namen werden wir danach einzeln mithilfe der For-Schleife ausgeben lassen:

```
{
    string[] names = new string[]
    {
      "Sebastian",
      "Sabine",
      "Sabrina",
      "Alina",
      "Horst",
      "Hendrik",
      "Klaus"
    };

    for (int i = 0; i < names.Length; i++)
    {
        Console.WriteLine(names[i]);
    }

    Console.ReadKey();
}
```

Abbildung 10, Array mit For-Schleife durchlaufen lassen

Ausgabe:

```
Sebastian
Sabine
Sabrina
Alina
Horst
Hendrik
Klaus
```

Abbildung 11, Output

Info:

Die Eigenschaft **Length** beinhaltet die Anzahl der Elemente eines Arrays.

Im nächsten Beispiel werden wir ein 2D-Array mit einer For-Schleife durchlaufen lassen. Wir legen für unser Beispiel drei Produkte und deren Kategorien fest. Ausgegeben werden sollen dann Produkt und Kategorie jeweils zusammen in einer Zeile. Dafür benötigen wir **eine For-Schleife**, die durch die **Zeilen** läuft und **eine weitere**, die durch die **Spalten** iteriert. Wir beginnen mit der äußeren For-Schleife, die die einzelnen Zeilen des Arrays durchlaufen wird und schreiben anschließend eine innere For-Schleife, die durch die Spalten laufen wird.
Dimension 0 = Horizontale (Spalten), Dimension 1 = Vertikale (Zeilen)

```
{
    string[,] products = new string[,]
    {
        {
            "Apfel",
            "Smartphone",
            "Schokoladentafel"
        },
        {
            "Lebensmittel",
            "Elektronik",
            "Lebensmittel"
        }
    };

    for(int zeile = 0; zeile < products.GetLength(1); zeile++)
    {
        for (int spalte = 0; spalte < products.GetLength(0); spalte++)
        {
            Console.Write(products[spalte, zeile]);
            Console.Write(" ");
        }

        Console.WriteLine();
    }

    Console.ReadKey();

}
```

Abbildung 12, zwei For-Schleifen iterieren durch einen 2D-Array

```
Apfel Lebensmittel
Smartphone Elektronik
Schokoladentafel Lebensmittel
```

Abbildung 13, Output

DIE FOREACH-SCHLEIFE

- Mit Foreach-Schleifen können wir ganz einfach die einzelnen Werte von Datenstrukturen durchlaufen
- Die Variable mit dem aktuellen Wert kann man **nicht** manipulieren – dafür müsste man direkt eine For-Schleife verwenden

```
int[] numbers = new int[] { 1, 4, 5, 6 };

foreach(int number in numbers)
{
    Console.WriteLine(number);
}

Console.ReadKey();
```

Abbildung 14, die Foreach-Schleife

Ausgabe:

```
1
4
5
6
```

Abbildung 15, Output

DIE FOREACH-SCHLEIFE IN DER PRAXIS

Für unser erstes Beispiel erstellen wir ein String Array, das wir mithilfe der Foreach-Schleife durchlaufen lassen. Hier sollen alle Namen des Arrays ausgegeben werden:

```
string[] names = new string[]
{
    "Alina",
    "Horst",
    "Enrico",
    "Sandra",
    "Hendrik"
};

foreach(string name in names)
{
    Console.WriteLine(name);
}

Console.ReadKey();
```

Abbildung 16, die Foreach-Schleife in einem Praxisbeispiel

Ausgabe:

```
Alina
Horst
Enrico
Sandra
Hendrik
```

Abbildung 17, Output

Wichtig:

Der Wert der Variable „name“ kann bei der Foreach-Schleife **nicht überschrieben** werden, da „name“ bereits als Foreach-Iterationsvariable verwendet wird.

DIE SCHLÜSSELWÖRTER BREAK UND CONTINUE

Break:

Mit break kann man eine Schleife an der Stelle abbrechen, an der sie sich gerade befindet. Anders gesagt: Man springt aus der Schleife heraus.

Beispiel: Laut Bedingung wird eine While-Schleife bis zur Zahl 10 ausgeführt. Wenn wir sie nun aber schon bei der Zahl 5 beenden möchten, verwenden wir dafür das Schlüsselwort **break**.

```
{
    int zahl = 1;

    while(zahl <= 10)
    {
        Console.WriteLine(zahl);

        if (zahl == 5)
        {
            break;
        }

        zahl++;
    }

    Console.ReadKey();
}
```

Abbildung 18, break zum Beenden der While-Schleife

Ausgabe:

```
1
2
3
4
5
```

Abbildung 19, Output

Continue:

Das Schlüsselwort **Continue** beendet nicht die gesamte Schleife, sondern unterbricht lediglich deren aktuellen Durchlauf.

Beispiel: Wir möchten, dass nur die **Zahl 5 nicht ausgegeben** wird.

```
{
    int zahl = 0;

    while(zahl < 10)
    {
        zahl++;

        if (zahl == 5)
        {
            continue;
        }

        Console.WriteLine(zahl);
    }

    Console.ReadKey();
}
```

Abbildung 20, continue zur Unterbrechung des Schleifendurchlaufs

```
1
2
3
4
6
7
8
9
10
```

Abbildung 21, Output

Info:

Wenn wir nur eine **einzige Anweisung** in ein if-Statement schreiben möchten, benötigen wir **keine** geschweifte Klammer jeweils vor und nach der Anweisung.
Beispiel:

```
if (zahl == 5)
{
    continue;
}
```

oder…

```
if (zahl == 5)
    continue;
```