Lineare Abläufe von Programmen mit Entscheidungen: if -, else if- oder switch- Anweisungen

Schleifen / Iterationen / Loops: Beschreiben ein und dieselbe Technik und sind in der Informatik Synonyme. Sie dienen dazu Anweisungen wiederholt ausführen, wobei entweder eine bestimmte Anzahl vorgegeben oder der Ablauf dynamisch gesteuert wird.

Unterschied do while/while und for: Bei der for-Schleife ist die Anzahl Durchläufe der Schlaufe von Anfang an bekannt und bei der while Schlaufe ergibt es sich erst zur Laufzeit der Anwendung. Die while Schlaufe kann auch für den Zweck von der for-Schlaufe genutzt werden. Bei do/while, while und for gibt es Schlüsselwörter wie break und continue, die für den Unterbruch sorgen können.

If/else: Dient dem Programm um Entscheidungen zu treffen.

```
// MODUL 403: Programmabläufe prozedural implementieren
// M403-Codebeispiel-9:  elseif-Anweisung
// Ralph.Maurer@gmx.ch
// http://www.iet-gibb.ch
using System;
public class Entscheidung
{
  public static void Main()
  {
    string Name;
    Console.Write("Bitte geben Sie Ihren Namen ein: ");
    Name = Console.ReadLine();
    if(Name == "Bill Gates")
    {
      Console.WriteLine("Sie arbeiten wohl bei Microsoft!");
      Console.ReadLine();
    }
    else if(Name == "Steve Jobs")
    {
      Console.WriteLine("Danke für den Mac, den iPod, das iPhone, das iPad und die
Vi   es besser zu machen.");
      Console.ReadLine();
    }
    else if(Name == "Niklaus Wirth")
    {
      Console.WriteLine("Prozedural - können wir alle mal!");
      Console.ReadLine();
    }
    else {
      Console.WriteLine("Tut mir leid, aber Sie kenne ich nicht.");
      Console.ReadLine();
    }
  }
}
```

Syntax: if (Bedingung) ----> { //Anweisungsblock; }
| else { //Anweisungsblock; }
| else if (Bedingung) ----> { //Anweisungsblock; }

switch: Dient auch für Entscheidungen, bei mehr als 2 Möglichkeiten (**nur ganzzahlige Datentypen, Zeichnenketten**)

```
switch(variable)
{
   case „Hallo“:
      Console.WriteLine(„“);
      break; //Muss immer geschrieben werden! Sonst meldet fehler
   default: //Optional
      Console.WriteLine(„“);
      break;
}
```

Vergleichsoperatoren: = ><= is as == != && ||(or):

---

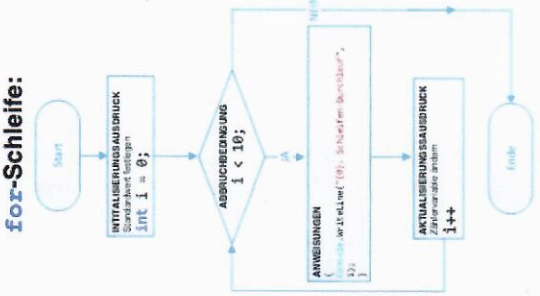
for-Schleife: Einfachste Iteration von C#, Zählschleife und führt Anweisungen eine bestimmte Anzahl mal durch.

- Die Anzahl Durchläufe der Schlaufe sind von Anfang an bekannt.
- Anweisungsblock ( ) hat kein Semikolon; am Ende.
- Geschweifte Klammern {} sind nicht zwingend notwendig, aber empfehlenswert.

```
using System;
public class Schleife
{
    ublic static void Main()
    {
      for (int i = 0; i < 10; i++)
        {
          Console.WriteLine("{0}. Schleifen-Durchlauf", i);
        }
        Console.ReadKey();
    }
}
```

```
file:///c/users/vm...
0. Schleifen-Durchlauf
1. Schleifen-Durchlauf
2. Schleifen-Durchlauf
3. Schleifen-Durchlauf
4. Schleifen-Durchlauf
5. Schleifen-Durchlauf
6. Schleifen-Durchlauf
7. Schleifen-Durchlauf
8. Schleifen-Durchlauf
9. Schleifen-Durchlauf
```

for-Schleife:



int i = 0: Zähler- /Zähler-variable mit Startwert 0 | **i < 10:** Abbruchbedingung | **i++:** Aktualisierungsausdruck (wird um 1 erhöht wie i= i+1)

Syntax: for (Initialisierungsausdruck; Abbruchbedingung; Aktualisierungsausdruck) ----> { //Anweisungsblock; }

Beispiel mit Array:

```
using System;
namespace squares
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] square;
            Console.Write("Geben Sie den höchsten Array-Index ein: ");
            square = new int[Convert.ToInt32(Console.ReadLine()) + 1];
            for (int i = 0; i <= square.Length -1; i++)
            {
                square[i] = i * i;
                Console.WriteLine("Quadrat von {0}: {1}", i, square[i]);
            }
            Console.ReadKey();
        }
    }
}
```

```
Geben Sie den höchsten Array-Index ein: 5
Quadrat von 0: 0
Quadrat von 1: 1
Quadrat von 2: 4
Quadrat von 3: 9
Quadrat von 4: 16
Quadrat von 5: 25
```

Beispiel mit Break:

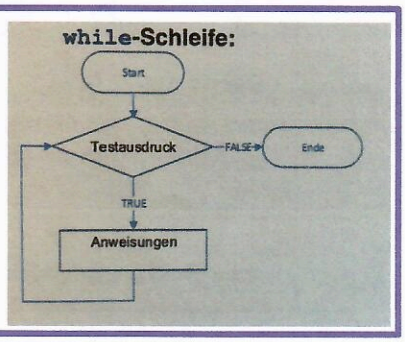
```
using System;
public class Quadratzahlen
{
    public static void Main()
    {
        int i;
        for (i = 1; i <= 100; i++)
        {
            Console.WriteLine("Das Quadrat von {0} ist {1}!", i, i * i);
            Console.Write("Möchten Sie die nächste Quadratzahl berechnen (j / n)? ");
            if (Console.ReadLine() == "n")
            { break; }
        }
        Console.WriteLine("So, fertig quadratiert!");
    }
}
```

while-Schleife: Diese Schlaufe wird so viel Mal ausgeführt, bis die Bedingung erfüllt ist. True oder False. Wenn beim ersten Durchlauf False erscheint, dann wird die Anweisung gar nicht ausgeführt. Wenn es True ergibt wird die Anweisung ausgeführt, bis die Bedingung False gibt.

```
using System;
public class Schleife
{
    static void Main(string[] args)
    {
        Console.Write("Geben Sie eine Zahl zwischen 1 und 10 ein");
        int anzahl = Convert.ToInt32(Console.ReadLine());
        int counter = 1;

        while (counter <= anzahl)
        {
            Console.WriteLine("{0}.Schleifendurchlauf", counter);
            counter++;
        }
        Console.ReadKey();
    }
}
```

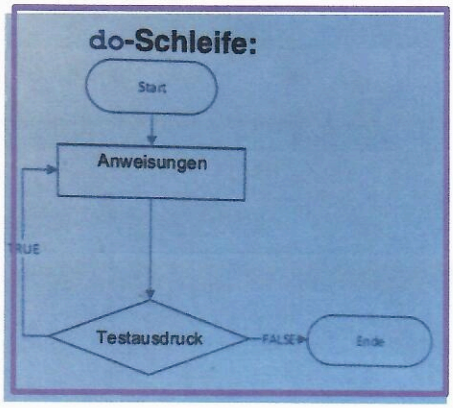
```
Geben Sie eine Zahl zwischen 1 und 10 ein5
1.Schleifendurchlauf
2.Schleifendurchlauf
3.Schleifendurchlauf
4.Schleifendurchlauf
5.Schleifendurchlauf
```



Syntax: while (Bedingung) -----> { //Anweisung; }

---

do while-Schleife: Die Anweisung wird im Gegensatz zur while-Schlaufe mindestens einmal ausgeführt.



Syntax: do {//Anweisung} while (Bedingung);

Beispiel mit while und do while:

```
namespace ab07_Rechentrainer
{
    class Program
    {
        static void Main(string[] args)
        {
            int reihe = 0;
            int resultat;
            int ausrechen;
            string frage = "j";
            Random Rnd = new Random();

            Console.WriteLine("Mit diesem Programm können Sie Kopfrechnen üben.");
            Console.WriteLine("Möglich sind Multiplikationen der Zahlen 2 - 12.");
            Console.WriteLine("Um das Programm zu Beenden, geben Sie die Zahlen 9999 ein."
            while (frage == "j")
            {
                while (reihe < 2 || reihe > 12)
                {
                    Console.Write("Geben Sie nun die Zahlenreihe ein, die Sie üben möchten
                    reihe = Convert.ToInt32(Console.ReadLine());
                }
                do
                {
                    int zufallszahl = Rnd.Next(2, 13);
                    Console.Write("{0} * {1}: ", zufallszahl, reihe);
                    resultat = Convert.ToInt32(Console.ReadLine());
                    ausrechen = zufallszahl * reihe;
                    while (resultat != ausrechen && resultat != 9999)
                    {
                        Console.WriteLine("Falsch");
                        Console.Write("{0} * {1}: ", zufallszahl, reihe);
                        resultat = Convert.ToInt32(Console.ReadLine());
                    }

                } while (resultat != 9999);
                Console.Write("Möchten Sie eine andere Zahlenreihe trainieren (j/n)? ");
                frage = Console.ReadLine();
            }
            Console.WriteLine("Übung fertig!");
            Console.ReadKey();
        }
```

Diverses Datentypen

Variablendeklaration: Datentyp Variablenname1, Variablenname2; **oder** Datentyp Variablenname = Datenwert;
2. Datentypen: Datenstrukturen von diverser Natur abspeichern
Wertetypen: Werte können direkt abgespeichert werden
Double(float, decimal) = Fliesskommazahlen
Integer(Int32/64)(long,sbyte,short,uint,ulong,uahort) = Ganze Zahlen
Boolean = Wahrheitswert True oder False
String = Text
Grösse von Wertetypen = sizeof(z.B. double)
3. Verweistypen/Referenztyp: Enthält Zeiger, der auf einen anderen Speicherbereich zeigt, der die Daten enthält
Array, String, Klassen, Delgaten (**Verweistyp** wird mit **new** deklariert sie werden langfristig gespeichert im heap, später wieder verwendbar) -> z.B. Rechteck r = new Rechteck();
Wertetypen in Verweistypen verwandeln: Boxing -> object name = variablezumboxen; **Unboxing ->** Wertetyp Neuevariable = (Wertetyp)name;

Array: Bei einem Array muss man zuerst bestimmen wie gross es sein soll

```
// Declare a single-dimensional array
int[] array1 = new int[5];
// Declare and set array element values
int[] array2 = new int[] { 1, 3, 5, 7, 9 };
// Alternative syntax
int[] array3 = { 1, 2, 3, 4, 5, 6 };
```

Beispiele:

Random Rnd = new Random(); // initialisiert die Zufallsklasse
----> int zufallszahl = Rnd.Next(0,2);

int[] square;
square = new int[Convert.ToInt32(Console.ReadLine()) + 1];

string[] Woche = {„mo“,“di“};
Console.WriteLine(Woche[0]);

Console.WriteLine(args.Length);