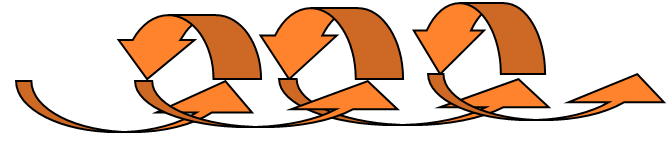
**Modul 319**

***Block 02***

***- Kontrollstrukturen -***





**Dokumentenversion:**

|  |  |
| --- | --- |
| V1.1 | Erstellung durch Roland Bucher |
| V1.2 | Integration der Übungen aus dem Stützkurs |

**Voraussetzung:**

Dieser Block setzt den Block 1 des Moduls 319 voraus.

**Aufbau dieses Blocks:**

Der Block stellt im ersten Kapitel in Form einer Zusammenfassung die wichtigsten Elemente von Verzweigungen (Auswahl) dar.

Im zweiten Kapitel werden die wichtigsten Elemente von Schleifen (Iterationen oder Wiederholungen) zusammengefasst. Das letzte Kapitel des Dokumentes enthält Übungen im Umfang von über 6 Lektionen zu diesen Themen.

**Ziele dieses Blocks:**

* Die Lernenden können c#- und Java-Programmcode mit einseitigen, zweiseitigen und mehrstufigen Verzweigungen korrekt interpretieren und Fehler in existierenden Programmen erkennen
* Die Lernenden können gegebene Aufgabenstellungen mit Einbezug von einseitigen, zweiseitigen und mehrstufigen Verzweigungen in c#- oder Java-Programmcode realisieren.
* Die Lernenden können c#- und Java-Programmcode mit kopf- und fussgesteuerten Schleifen (inkl. Zählschleife) korrekt interpretieren und Fehler in existierenden Programmen erkennen.
* Die Lernenden können gegebene Aufgabenstellungen mit Einbezug von kopf- und fussgesteuerten Schleifen (inkl. Zählschleife) in c#- oder Java-Programmcode realisieren.

**Zeitaufwand:**

Für die Abarbeitung dieses Blockes sind im Unterricht 8 Lektionen vorgesehen. Der Rest ist als Hausaufgabe zu lösen.

**Inhaltsverzeichnis:**

[1 Übersicht über Kontrollstrukturen 2](#_Toc85436817)

[2 Verzweigungen 2](#_Toc85436818)

[2.1 Einseitige Auswahl ( if ) 2](#_Toc85436819)

[2.2 Zweiseitige Auswahl ( if-else ) 2](#_Toc85436820)

[2.3 Auswahlen verschachteln 3](#_Toc85436821)

[2.4 Mehrstufige Auswahl (Fallauswahl) 4](#_Toc85436822)

[3 Schleifen (Wiederholungen) 5](#_Toc85436823)

[3.1 Kopfgesteuerte Schleife (while) 6](#_Toc85436824)

[3.2 Fussgesteuerte Schleife (do-while) 7](#_Toc85436825)

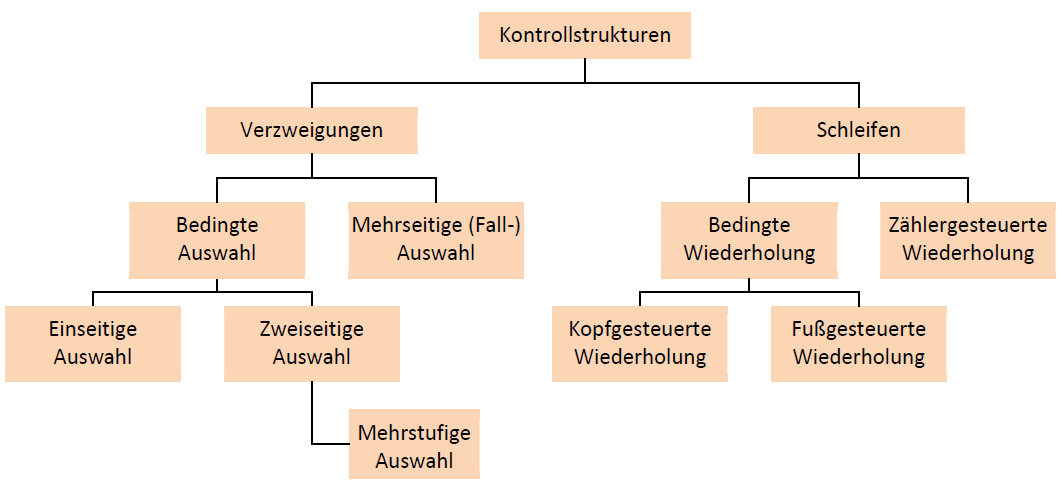
[3.3 Zählergesteuerte Wiederholung (for) 8](#_Toc85436826)

[3.4 break und continue Anweisungen in Schleifen 9](#_Toc85436827)

[4 Übungen (110’) 10](#_Toc85436828)

# Übersicht über Kontrollstrukturen

Kontrollstrukturen bestehen aus zwei Kategorien: Verweigungen (z.B. if-then-else) und Schleifen (z.B. while). Bei Verzwigungen geht es prinzipiell darum, ob eine Anweisung oder ein Block von Anweisungen ausgeführt werden soll oder nicht. Bei Schleifen geht es darum, dass eine Anweisung oder ein Block von Anweisungen unter Umständen mehrmalls ausgeführt werden sollen.

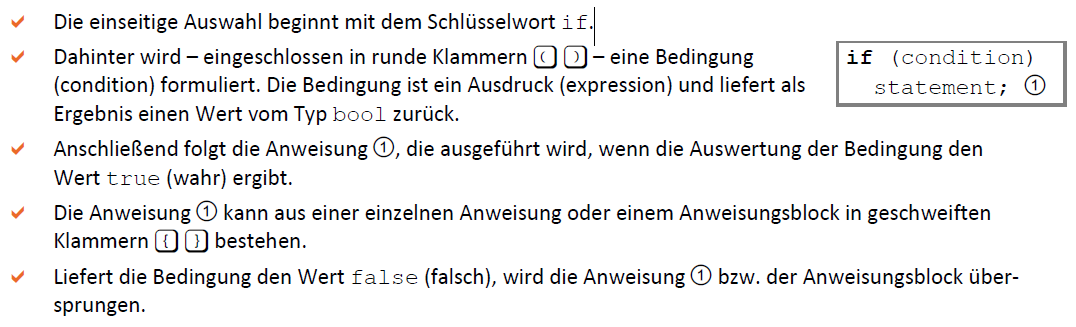


# Verzweigungen

*(Ein Auszug aus ihrem Herdt-Buch C# - Grundlagen der Programmierung mit VisualStudio 2019, Kap. 7.2 bis 7.5)*

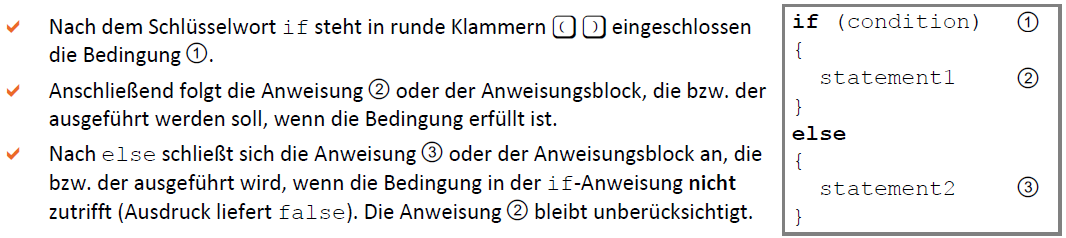
## Einseitige Auswahl ( if )

Bei vielen Problemstellungen ist die Verarbeitung von Anweisungen von Bedingungenabhängig. Nur wenn die Bedingung erfüllt ist, wird die betreffende Anweisung (bzw. der Anweisungsblock) ausgeführt. Anderenfalls wird die Anweisung übersprungen.



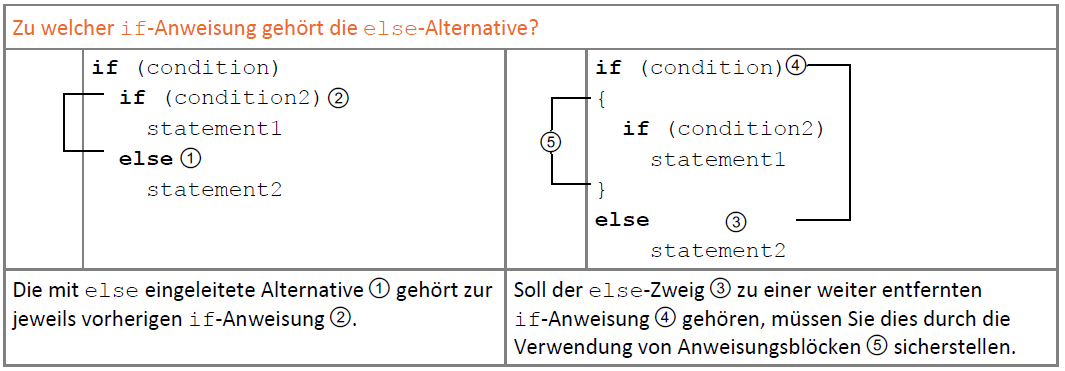
## Zweiseitige Auswahl ( if-else )

Im Gegensatz zur einseitigen Auswahl hat die zweiseitige Auswahl noch am Schluss einen else -Teil. Während die einseitigen Auswahl nur entschieden kann, ob eine Anweisung oder ein Block von Anweisungen ausgeführt werden soll oder nicht, kann die zweiseitige Auswahl entscheiden, welcher der beiden Teile (if oder else) ausgeführt werden soll.



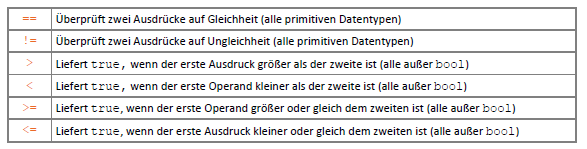
## Auswahlen verschachteln

Eine mehrstufige Auswahl erreichen Sie in C#, indem Sie mehrere if- bzw. if-else-Anweisungen schachteln. Dazu schreiben Sie innerhalb eines Anweisungsblocks einer if- bzw. if-else-Anweisung eine weitere if- bzw. if-else-Anweisung.



## Vergleichs- und Verknüfungsmöglichkeiten bei der Formulierung von Bedingungen

Eine Bedingung (Condition) muss gesamthaft immer auf true oder falsch auswertbar sein. Dazu existieren verschiedene Verghleichsmöglichkieten:



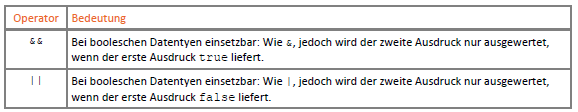
Nicht selten gibt es mehrere Bedigungen, die unterinander mit UND oder ODER verknüpft sind.   
Zum Beispiel: Die Variable x muss grösser als 10 UND kleiner als 20 sein:

**if (x>10 && x<20)**

**{**

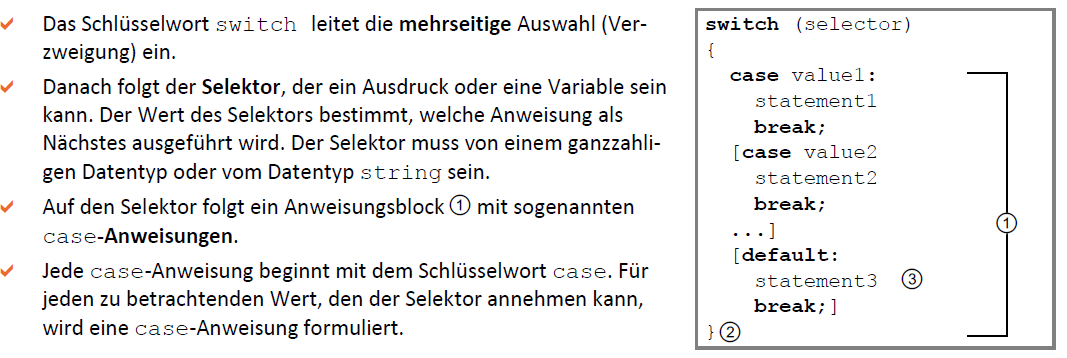
**…**

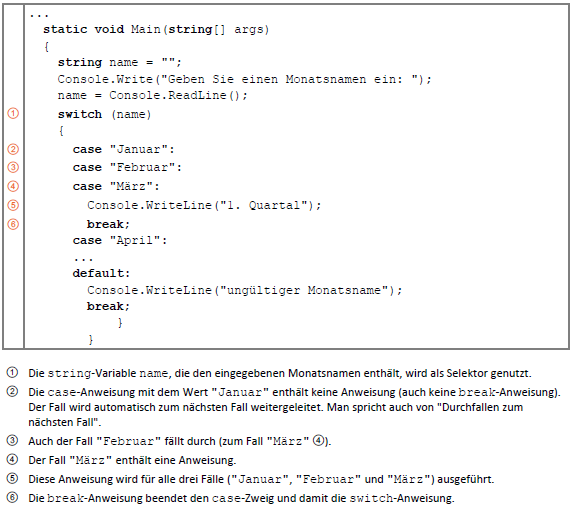
**}**



## Mehrstufige Auswahl (Fallauswahl)

Bei if-Konstruktionen lassen sich immer nur zweialternative Programmteile ausführen, da eine Bedingung als logischer Ausdruck nur die Werte true bzw. false ergeben kann. Bei einer mehrseitigenAuswahltesten Sie den Wert einer Variablen oder eines komplexen Ausdrucks. Diese Variable bzw. dieser Ausdruck wird als Selektorbezeichnet. In Abhängigkeit vom Wert des Selektors können fallweise (case) verschiedene Anweisungsblöcke ausgeführt werden. Die mehrseitige Auswahl wird daher auch Selektion genannt.



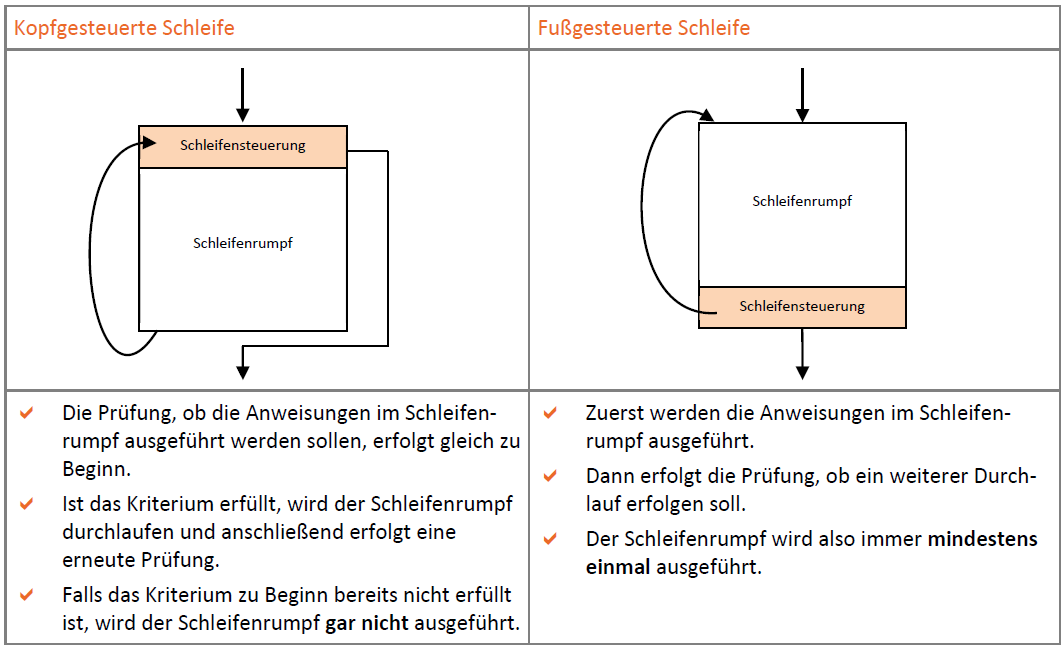


# Schleifen (Wiederholungen)

*(Ein Auszug aus dem Herdt-Buch C# - Grundlagen der Programmierung mit VisualStudio 2019, Kap. 7.6 bis 7.10)*

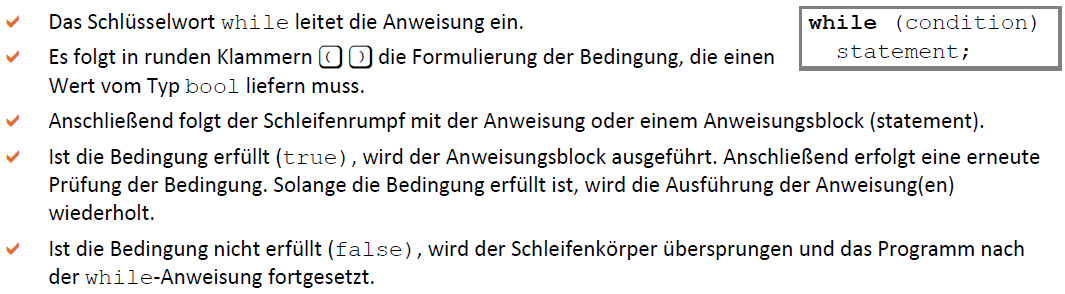
Eine Schleife besteht aus der Schleifensteuerungund dem Schleifenrumpf. Der Schleifenrumpf umfasst den Programmteil, der wiederholt werden soll. Die Schleifensteuerung legt fest, wie oft die Anweisungen im Schlei-fenrumpf wiederholt werden sollen bzw. nach welchen Kriterien entschieden wird, ob eine Wiederholung erfolgen soll.

Es werden zwei Steuerungsarten unterschieden:

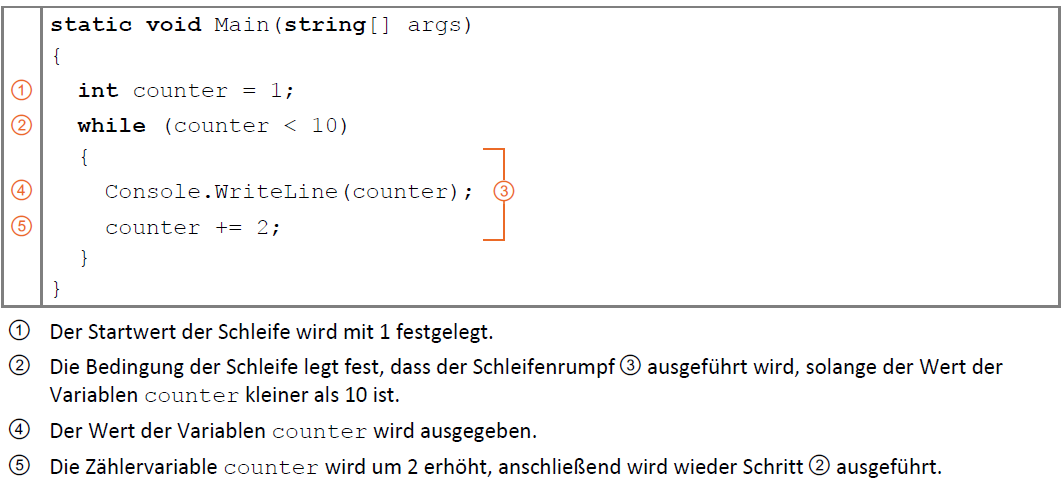


## Kopfgesteuerte Schleife (while)

Bei der while-Anweisung handelt es sich um eine kopfgesteuerteSchleifenstruktur. Die Ausführung der Anweisungen im Schleifenrumpf ist von der Gültigkeit einer Bedingung abhängig, die gleich zu Beginnder Anweisung überprüft wird. Solange(while) die Bedingung erfüllt ist, werden die folgenden Anweisungen (Schleifenrumpf) ausgeführt. Anschließend folgt der nächste Durchlauf der Schleife und die Bedingung wird erneut geprüft. Ist die Bedingung nichterfüllt, wird das Programm hinter der while-Anweisung fortgesetzt.

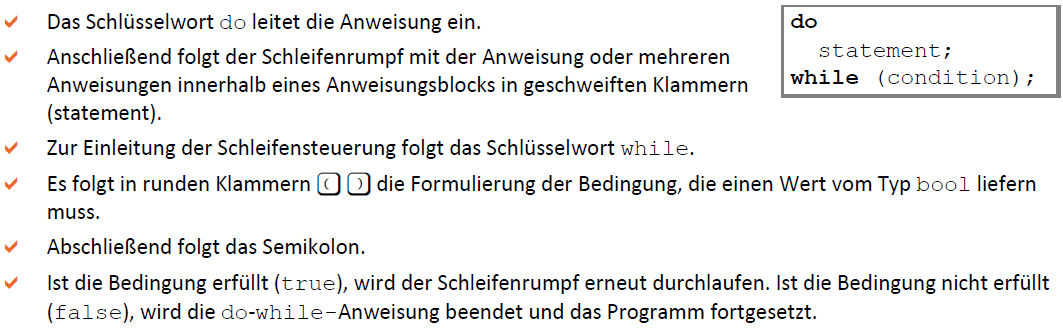


Beispiel: Das folgende Programm gibt die Zahlen 1, 3, 5, 7, 9 aus. Dazu wird beim Startwert 1 begonnen und in jedem Schleifendurchlauf der Wert 2 addiert, solange der Wert der Variablen counter kleiner als 10 ist.

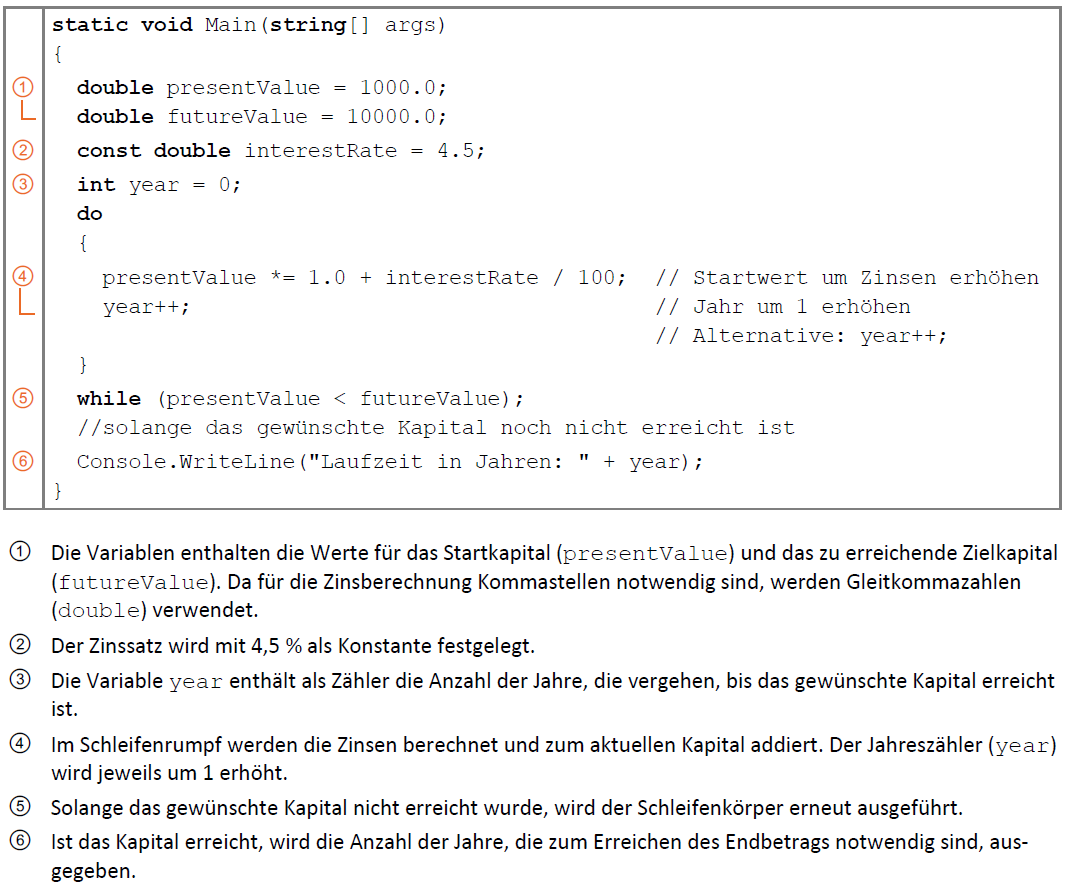


## Fussgesteuerte Schleife (do-while)

Die do-while-Anweisung ist eine fussgesteuerteSchleifenstruktur. Die Bedingung wird erst nachdem Durchlauf des Schleifenrumpfes ausgewertet. Dadurch wird der Schleifenkörper mindestenseinmalausgeführt. Wie bei der kopfgesteuerten while-Anweisung wird der Schleifenrumpf ausgeführt, solange die Bedingung erfüllt ist.

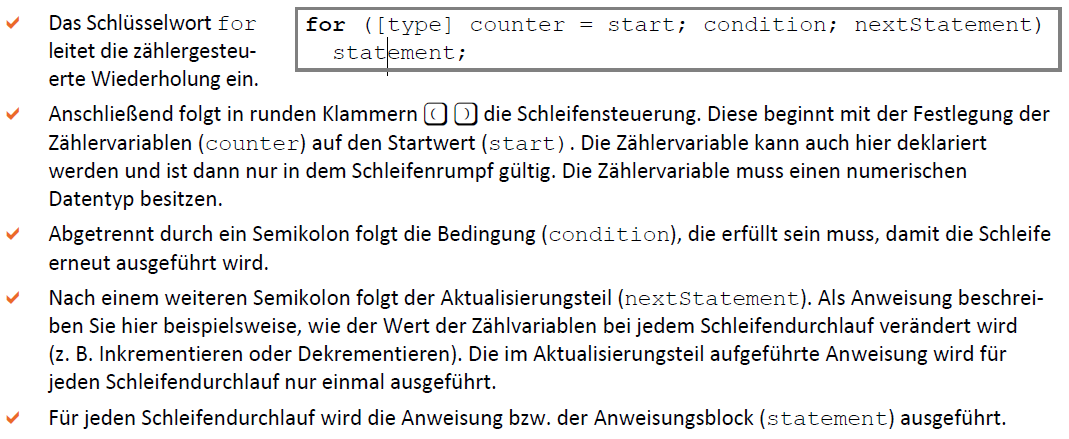


Beispiel: Das folgende Programm berechnet die Anzahl der Jahre, die benötigt werden, um – beginnend mit einem Startkapital von 1’000 EUR – nur über die Zinsen (4,5 %) einen Endbetrag von 10’000 EUR zu erreichen.

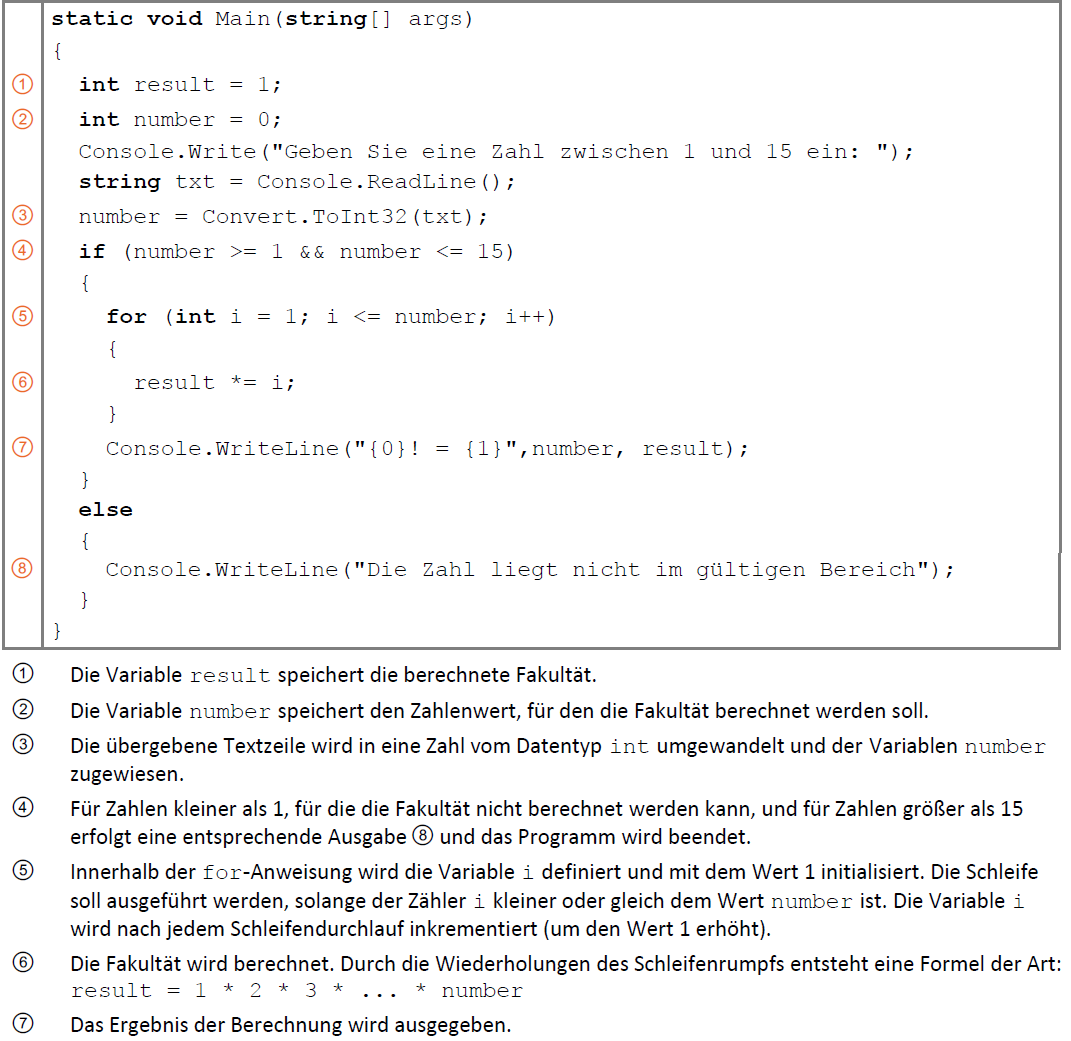


## Zählergesteuerte Wiederholung (for)

Wenn bereits zu Begin der Schleife genau bekannt ist, wie oft der Schleifenrumpf (eine Anweisung oder ein Anweisungsblock) wiederholt werden soll, ist die Verwendung der for-Schleife die kompakteste Form.

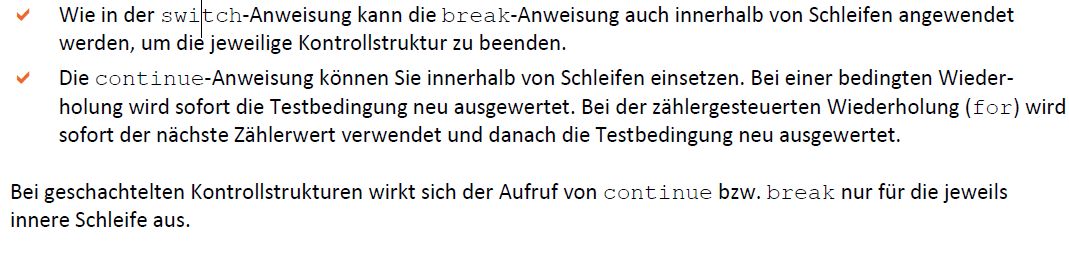


Beispiel: Das folgende Programm berechnet die Fakultät einer natürlichen Zahl im Bereich zwischen 1 und 15. Zum Beispiel ist die Fakultät der Zahl 4 gleich 24 (Berechnung: 4! = 1 \* 2 \* 3 \* 4).

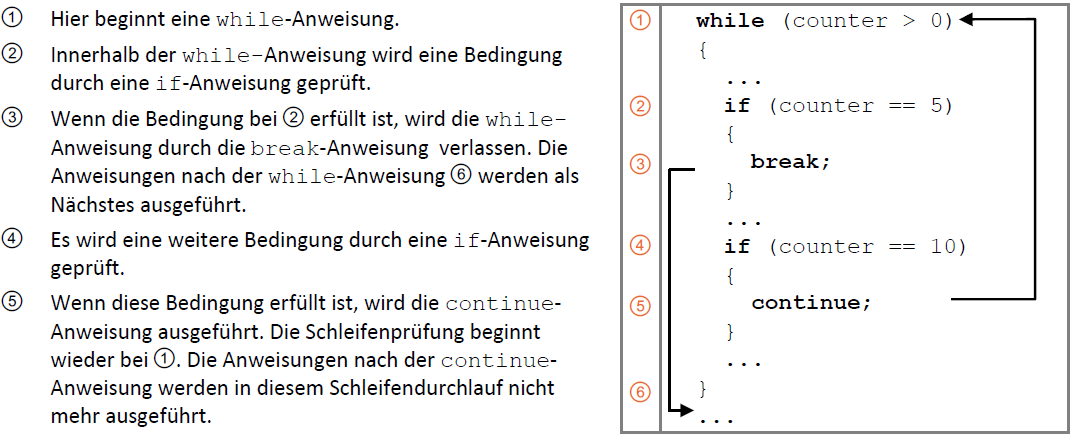


## break und continue Anweisungen in Schleifen

Zusätzlich zur Schleifensteuerung haben Sie mit einer break- bzw. einer continue-Anweisung die Möglichkeit, den Ablauf des Programms zu beeinflussen.



Beispiel:



# Übungen (260’ / 6 Lektionen)

**Aufgabe 01 (Programmcode mit einer Verweigungen interpretieren können)**

**Lernziel**: Die Lernenden können c#- und Java-Programmcode mit einseitigen, zweiseitigen und mehrstufigen Verzweigungen korrekt interpretieren und Fehler in existierenden Programmen erkennen

**Zeit**: 10’

**Aufgabe**: Was gibt das Programm aus, wenn folgende Werte eingegeben würden?

a) Zahl 1 = 6, Zahl 2 = 3

b) Zahl 1 = 7, Zahl 2 = 7

|  |
| --- |
| static void Main(string[] args)  {  Console.Write("Zahl 1:");  int zahl1 = Convert.ToInt32(Console.ReadLine());  Console.Write("Zahl 2:");  int zahl2 = Convert.ToInt32(Console.ReadLine());  if (zahl2 < zahl1 && zahl1 > 5)  {  int temp = zahl1;  zahl1 = zahl2;  zahl2 = temp;  }  else  {  if (zahl1 == zahl2)  zahl1 = 5;  zahl2 = 8;  }  Console.WriteLine("Ausgabe 1 = " + Convert.ToString(zahl1));  Console.WriteLine("Ausgabe 2 = " + Convert.ToString(zahl2));  Console.ReadLine();  } |

|  |  |
| --- | --- |
| Ihre Lösungen: | |
| Aufgabe a | Aufgabe b |
| Zahl 1: 6  Zahl 2: 3  Ausgabe 1: 3…………………………  Ausgabe 2: 6…………………………  …………………………………………  …………………………………………  ………………………………………… | Zahl 1: 7  Zahl 2: 7  Ausgabe 1: 5…………………………  Ausgabe 2: 8…………………………  …………………………………………  ………………………………………….  …………………………………………. |

**Aufgabe 02 (Programmcode mit mehreren Verweigungen interpretieren können)**

**Lernziel**: Die Lernenden können c#- und Java-Programmcode mit einseitigen, zweiseitigen und mehrstufigen Verzweigungen korrekt interpretieren und Fehler in existierenden Programmen erkennen

**Zeit**: 10’

**Aufgabe**: Was gibt das Programm aus, wenn folgende Noten eingegeben würden?

a) Note 1 = 4.2, Note 2 = 3.5

b) Note 1 = 5.2, Note 2 = 5.8

c) Note 1 = 3.8, Note 2 = 3.3

|  |
| --- |
| static void Main(string[] args)  {  int anzahl = 0;  double note1 = 0.0, note2 = 0.0;  Console.Write("Note 1:");  note1 = Convert.ToDouble(Console.ReadLine());  anzahl+=1;  Console.Write("Note 2:");  note2 = Convert.ToDouble(Console.ReadLine());  anzahl++;  double schnitt = (note1 + note2) / anzahl;  if (schnitt >= 4)  {  Console.WriteLine("\*\*\*\*\*");  int durchschnitt = (int)(schnitt \* 2);  durchschnitt = (durchschnitt + 1) / 2;  if (durchschnitt == 4)  Console.WriteLine("Typ 2");  else  {  if (durchschnitt == 5)  Console.WriteLine("Typ 3");  else  Console.WriteLine("Typ 4");  }  }  else  {  Console.WriteLine("-----");  if (note1 >= 4 || note2 >= 4)  Console.WriteLine("Typ 1");  else  Console.WriteLine("Typ 0");  }  Console.ReadLine();  } |

|  |  |  |
| --- | --- | --- |
| Ihre Lösungen: | | |
| Aufgabe a | Aufgabe b | Aufgabe c |
| Note 1: 4.2  Note 2: 3.5  ………………….…………  ………………………….…  ………………………….… | Note 1: 5.2  Note 2: 5.8  ………………………….…  ………………………….…  ………………………….… | Note 1: 3.8  Note 2: 3.3  ………………………….…  ………………………….…  ………………………….… |

**Aufgabe 03 (Fehler im Programmcode erkennen und korrigieren können)**

**Lernziel**: Die Lernenden können c#- und Java-Programmcode mit einseitigen, zweiseitigen und mehrstufigen Verzweigungen korrekt interpretieren und Fehler in existierenden Programmen erkennen

**Zeit**: 10’

**Aufgabe**: Das folgende Programm sollte drei Werte (Wert 1, Wert 2 und Wert 3) vom Benutzer entgegennehmen und den grössten der drei Werte wieder ausgeben. Leider haben sich zwei Fehler eingeschlichen. Erklären sie am Schluss bei welcher Zeile (Zeilen­nummer) es warum einen Kompilerfehler gibt und versuchen sie ihn zu korrigieren.

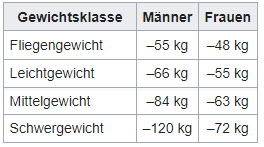
|  |
| --- |
| 01 static void Main(string[] args)  02 {  03 int anzahl = 1;  04 double wert1 = 0, wert2 = 0, wert3=0, max= 0;  05 Console.Write("Wert " + Convert.ToString(anzahl++) );  06 wert1 = Convert.ToDouble(Console.ReadLine());  07 Console.Write("Wert " + Convert.ToString(anzahl++));  08 wert2 = Convert.ToDouble(Console.ReadLine());  09 Console.Write("Wert " + Convert.ToString(anzahl++));  10 wert3 = Convert.ToDouble(Console.ReadLine());  11 if (wert1 > wert2 && > wert3)  12 max = wert1;  13 else  14 {  15 if (wert2 > wert3)  16 max = wert2;  17 else (wert3 > wert2)  18 max = wert3;  19 }  20 Console.WriteLine("Der grösste Wert ist:"+ Convert.ToString(max));  21 Console.ReadLine();  22 } |
| **Kompilerfehler 1 bei Zeile[n] Nummer:** ………………….…  **Grund:** ………………….…………………………………………………………………………  ………………….…………………………………………………………………………  **Korrektur:** ………………….…………………………………………………………………………  ………………….………………………………………………………………………… |
| **Kompilerfehler 2 bei Zeile[n] Nummer:** ………………….…  **Grund:** ………………….…………………………………………………………………………  ………………….…………………………………………………………………………  **Korrektur:** ………………….…………………………………………………………………………  ………………….………………………………………………………………………… |

**Aufgabe 04 (Gegebene Aufgabenstellung in Programmcode umsetzen können)**

**Lernziel**: Die Lernenden können c#- und Java-Programmcode mit einseitigen, zweiseitigen und mehrstufigen Verzweigungen korrekt interpretieren und Fehler in existierenden Programmen erkennen

**Zeit**: 25’

**Aufgabe**: Beim Ringen wird in verschiedenen Gewichtsklassen gerungen.



Schreiben Sie ein Programm, das den Benutzer zur Eingabe der beiden Informationen Geschlecht und Gewicht auffordert.

Im Verarbeitungteil ermittelt ihre App mit Hilfe von Verweigungen in welcher Kategorie gerungen werden kann. Die ermittelte Information wird danach dem Benutzer ausgegeben.

Gibt der Benutzer zum Beispiel folgende Werte ein:

Geschlecht [m/w] : w

Gewicht [in Kg] : 75

Ausgabe = Gewichstklasse: Mittelgewicht

Gemacht

**Aufgabe 05 (Jahreszeiten mit Fallauswahl)**

**Thema:** Eingabe, Verarbeitung (mit Fallauswahl), Ausgabe

**Lernziele**: Die Lernenden können gegebene Aufgabenstellungen mit Einbezug von mehrstufigen Verzweigungen in c#- oder Java-Programmcode realisieren.

**Zeit**: 20’

**Aufgabe**: Schreiben Sie ein Programm, das den Benutzer zur Eingabe eines Monats auffordert (Zahl von 1 bis 12).

Im Verarbeitungteil ermittelt ihre App mit Hilfe einer mehrstufigen Verweigung (switch -case) ob es sich um einen Frühlings, Sommer- , Herbst- oder Wintermonat handelt. Die ermittelte Information wird danach dem Benutzer ausgegeben.

Gibt der Benutzer zum Beispiel folgende Werte ein:

Monat : 4

Erfolgt die Ausgabe:

Der Monat 4 ist ein Frühlingsmonat

**Ihre Lösung (Kopieren sie am Schluss ihre Lösung hier hinein):**

|  |
| --- |
| static void Main(string[] args)  {  Console.WriteLine("Jahreszeiten");  Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Console.ReadLine(); nicht machen  } |

**Aufgabe 06 (Gegebene Aufgabenstellung in Programmcode umsetzen können)**

**Lernziel**: Die Lernenden können c#- und Java-Programmcode mit einseitigen, zweiseitigen und mehrstufigen Verzweigungen korrekt interpretieren und Fehler in existierenden Programmen erkennen

**Zeit**: 25’

**Aufgabe**: Untenstehend exisitert eine Anleitung in Pseudocode/Textform, anhand der man zu jedem beliebigen Datum den zugehörigen Wochentag (z.B. Montag) ermitteln kann. Schreiben sie ein Programm, das vom Benutzer die drei ganzzahlige Werte t (Tag), m (Monat) und j (Jahr) entgegennimmt und eine Aussage in Textform macht, um welchen Wochentag es sich handelt.

**Testdaten**: 25.10.2021 🡪 Montag , 24.12.1980 🡪 Mittwoch, 07.04.2037 🡪 Dienstag

Weitere Testdaten mit dem Windowskalender überprüfen

|  |
| --- |
| **Anleitung in Pseudocode / Textform:**   1. Erstellen Sie drei Variablen ( t, m, j ) vom Typ int und lesen Sie die Werte ein 2. Berechnen Sie den Wochentag h nach folgendem Algorithmus:    * Falls m <= 2 ist, erhöhe m um 10 und erniedrige j um 1, andernfalls erniedrige m um 2.    * Berechne die ganzzahligen Werte c = j/100 und j = j Modulo 100 (Modulo 🡪 %)    * Berechne den ganzzahligen Wert   h = (((26\*m-2)/10)+t+j+j/4+c/4-2\*c) Modulo 7   * + Falls h kleiner 0 ist, erhöhe h um 7   + Anschliessend hat h einen Wert zwischen 0 und 6, wobei die Werte 0,1,…,6 den Tagen Sonntag, Montag, ….., Samstag entsprechen.  1. Geben Sie das Ergebnis in der Form “Der 24.12.2001 ist ein Montag“ aus. |

**Aufgabe 07 (Gegebene Aufgabenstellung in Programmcode umsetzen können)**

**Lernziel**: Die Lernenden können c#- und Java-Programmcode mit einseitigen, zweiseitigen und mehrstufigen Verzweigungen korrekt interpretieren und Fehler in existierenden Programmen erkennen

**Zeit**: 60’

**Aufgabe**: Untenstehend exisitert eine Anleitung in Pseudocode/Textform, anhand der man zu jedem beliebigen Jahr das Datum des Ostersonntags ermitteln kann. Ostern fällt immer auf den Sonntag nach dem ersten Vollmond im Frühling.Es hat also mit den Bewegungen von Erde, Sonne und Mond zu tun. Klingt komplex, ist es auch. Zum Glück hat der Mathematiker Carl Friedrich Gauss (1777 - 1855) dieses Problem bereits mathematisch für uns gelöst. Nach­folgend wird ihnen textuell - gespickt mit Mathematik - erklärt, wie man ex­emplarisch für das Jahr 2019 berechnen kann, wann der Ostersonntag ist. Schreiben sie ein Programm, das vom Benutzer eine Jahreszahl entgegennimmt und für das gewählte Jahr das Osterdatum ausgibt.

**Testdaten**: Testen Sie die Korrektheit ihrer App mittels den folgenden Testwerten (fett= Osterparadoxon-Jahr):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1974:** 14.April | 2006: 16.April | 2011: 24. April | 2016: 27. März | 2021: 04. April |
| 1899: 02.April | 2007: 08.April | 2012: 08. April | 2017: 16. April | 2022: 17. April |
| 1900: 15.April | 2008: 23.März | 2013: 31. März | 2018: 01. April | 2023: 09. April |
| 1901: 07.April | 2009: 12.April | 2014: 20. April | **2019:** 21. April | 2024: 31. März |
| **2000**: 23.April | 2010: 04. April | 2015: 05. April | 2020: 12. April | **2038:** 25. April |

**Anleitung in Pseudocode / Textform:**

|  |  |  |
| --- | --- | --- |
| Ab dem Jahr | m | n |
| 1583 | 22 | 2 |
| 1700 | 23 | 3 |
| 1800 | 23 | 4 |
| 1900 | 24 | 5 |
| 2100 | 24 | 6 |
| 2200 | 25 | 0 |

*I) Die Zwischenwerte m und n bestimmen:*

*Zuerst müssen wir die Zwischenwerte m und n bestimmen, die je nach Jahreszahl variieren können. Z.B. ist im Jahr 2019 m=24 und n= 5. Die nebenstehende Tabelle zeigt die Sollwerte von m und n je nach Jahreszahl. Kann man m und n auch berechnen? Klar:*

*II) Die Zwischenwerte m und n berechnen:*

|  |  |  |
| --- | --- | --- |
| *Step* | *Allgemein* | *Für das Jahr 2019* |
| *01* | *m = (8\*( Jahr/100 ) + 13 ) / 25 - 2* | *m = (8\*( 2019/100 ) + 13 ) / 25 - 2*  ***m=*** *(8\* 20 + 13) / 25 - 2 =* ***4*** |
| *02* | *s = Jahr/100 - Jahr/400 - 2* | ***s =*** *2019/100 - 2019/400 - 2 =* ***13*** |
| *03* | *m = (15 + s – m ) % 30* | ***m*** *= (15 + s – m ) % 30 = (15 + 13 – 4 ) % 30 =* ***24*** |
| *04* | *n = (6 + s) % 7* | ***n*** *= (6 + 13) % 7* ***= 5*** |

*III) Die Zwischenwerte d und e berechnen:*

|  |  |  |
| --- | --- | --- |
| *Step* | *Allgemein* | *Für das Jahr 2019* |
| *05* | a = Jahr % 19  b = Jahr % 4  c = Jahr % 7 | a = 2019 % 19 = 5  b = 2019 % 4 = 3  c = 2019 % 7 = 3 |
| *06* | d = ( 19 \* a + m ) % 30 | **d** **=** (19 \* a + m ) % 30 = (19 \* 5 + 24 ) % 30 = **29** |
| *07* | Falls d gleich 29 ist setze d auf 28  andernfalls:  falls d=28 und a>=11 ist, setze d=27 | **d = 28** |
| *08* | e = ( 2 \* b + 4 \* c + 6 \* d + n ) % 7 | e = ( 2 \* b + 4 \* c + 6 \* d + n ) % 7  ***e =*** ( 2 \* 3 + 4 \* 3 + 6 \* 28 + 5 ) % 7 ***= 2*** |

*IV) Nun können der Tag und der Monat bestimmt werden:*

|  |  |  |
| --- | --- | --- |
| *09* | Ostertag = am ( 22 + d + e )ten März | Ostertag = am ( 22 + 28 + 2 )ten März  Ostertag = am 52. März |
| *10* | Falls der Ostertag > 31 ist, setze den Ostertag = Ostertag%31 und Ostermonat auf April | Ostertag = Ostertag%31  **Ostertag =** 52%31 = **21**  **Ostermonat = 4** |

* Weitere Infos: <https://de.wikipedia.org/wiki/Osterparadoxon>

**Aufgabe 08 (Gegebene Aufgabenstellung in Programmcode umsetzen können)**

**Lernziele**:

* Die Lernenden können c#- und Java-Programmcode mit einseitigen, zweiseitigen und mehrstufigen Verzweigungen korrekt interpretieren und Fehler in existierenden Programmen erkennen.
* Die Lernenden können gegebene Aufgabenstellungen mit Einbezug von kopf- und fussgesteuerten Schleifen (inkl. Zählschleife) in c#- oder Java-Programmcode realisieren

**Zeit**: 30’

**Aufgabe**: Mit Sicherheit kennen sie noch die Teibarkeitsregeln aus dem Primarschulunterricht:

* Eine Zahl ist durch 3 teilbar, wenn die Quersumme durch 3 teilbar ist.
* Eine Zahl ist durch 9 teilbar, wenn die Quersumme durch 9 teilbar ist.
* etc.

Schreiben sie ein Programm, bei dem sie eine ganze Zahl eingeben können.   
Für die eingegebene Zahl wird dann die Quersumme bereichnet und eine Aussage zur Teibarkeit durch 3 und 9 gemacht.

Am Ende wird gefragt, ob eine weitere Berechnung erwünscht ist. Falls Ja, wird der Bildschirminhalt gelöscht (Konsole.Clear() enspricht «cls») und das Programm erneut ausgeführt.

Beispiel: Zahl: 5989785

Quersumme = 51  
Teilbar durch 3: Ja

Teilbar durch 9: Nein

Möchten sie eine weitere Berechnung [ j / n ]: n

Gemacht

**Aufgabe 09 (Gegebene Aufgabenstellung in Programmcode umsetzen können)**

**Lernziele**: Die Lernenden können c#- und Java-Programmcode mit einseitigen, zweiseitigen und mehrstufigen Verzweigungen korrekt interpretieren und Fehler in existierenden Programmen erkennen.

Die Lernenden können gegebene Aufgabenstellungen mit Einbezug von kopf- und fussgesteuerten Schleifen (inkl. Zählschleife) in c#- oder Java-Programmcode realisieren

**Zeit**: 30’

**Aufgabe**: Schreiben Sie ein Programm, das dem Benutzer 6» zufällige» Zahlen im Bereich von 1 bis 42 und eine >Glückszahl im Bereich von 1 bis 6 ausgibt. Überprüfen sie, dass der Lotto-Tipp keine doppelten Zahlen enthält (also 12 soll z.B. nicht 2x vorkommen. Die Ausgabe muss noch nicht sorrtiert ausgegeben werden.

**Beispiele:** Ausgabe = 41, 29, 22, 36, 24, 5 Glueckszahl = 3

Ausgabe = 26, 12, 16, 25, 30, 18 Glueckszahl = 5

Ausgabe = 13, 28, 12, 23, 42, 19 Glueckszahl = 1

gemacht

**Aufgabe 10 (Gegebene Aufgabenstellung in Programmcode umsetzen können)**

**Lernziele**: Die Lernenden können c#- und Java-Programmcode mit einseitigen, zweiseitigen und mehrstufigen Verzweigungen korrekt interpretieren und Fehler in existierenden Programmen erkennen.

Die Lernenden können gegebene Aufgabenstellungen mit Einbezug von kopf- und fussgesteuerten Schleifen (inkl. Zählschleife) in c#- oder Java-Programmcode realisieren

**Zeit**: 60’

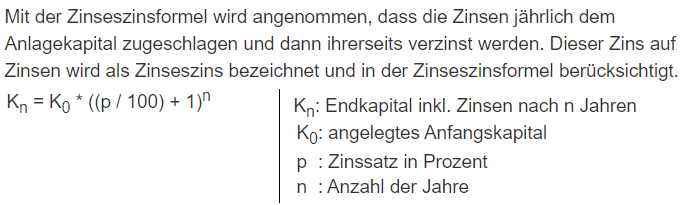
**Aufgabe**: Ihr Freund Bonifacius Pfiffikus erzählt ihnen von einer Theorie die er gehört habe. Die Theorie des Josefsrappen:   
*„Geld, das Zinseszinsen trägt, wächst anfangs langsam; da aber durch den Zinseszinseffekt die Rate fortwährend wächst, wird sie nach einiger Zeit unvorstellbar schnell wachsen. Ein Rappen, ausgeliehen bei der Geburt von Jesus im Jahre 0 zu einem Zinssatz von 5 %, würde heute zu einer grösseren Summe herangewachsen sein, als 70 Milliarden Erden aus purem Gold entsprechen.*

Sie können dies kaum glauben und möchten die Aussage mit einem zu erstellenden Programm überprüfen. Fragen Sie den Benutzer nach der Eingabe eines Start- und Zieljahres, nach einem Betrag und einem Zinssatz. Als Antwort liefert ihre App den Betrag in CHF und die Entprechung in Anzahl Erden aus purem Gold.

**Recherchen:**

* Berechnen Sie das Volumen der Erde anhand des Erdradius. Obwohl die Erde nicht exakt eine Kugel ist, approximieren wir sie als eine Kugel. Die Formel um ein Kugelvolumen zu berechnen wie auch den Erdradius finden sie im Web.
* Recherchieren Sie, welches Volumen ein Kilogramm Gold hat und was dies kostet

**Tipp:** Die Zinseszinsen können sie auf zwei Arten berechnen. Mit Verwendung der Zinseszinsformel oder mit Verwendung einer Schleife die jedes Jahr die Jahreszinsen dazurechnet.

****

**Hintgergrundinfos: (Auszug aus Wikipedia):**

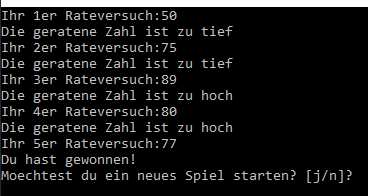
Das Gedankenexperiment vom Josephspfennig geht zurück auf den britischen Moralphilosophen, Geistlichen und Ökonom Richard Price (1723-1791) und illustriert in der Zinsrechnung das im englischen Sprachraum als miracle of compound interest bekannte Wachstum eines über einen langen Zeitraum angelegten Vermögens durch Zinseszinsen.

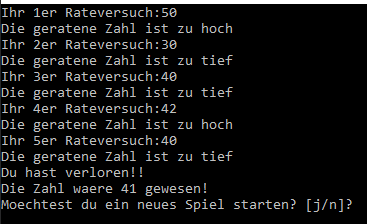
**Aufgabe 11 (Ratespiel)**

**Lernziele**: Die Lernenden können gegebene Aufgabenstellungen mit Einbezug von Kontrollstrukturen (Schleifen und Verzweigungen) in c#- oder Java-Programmcode realisieren

**Zeit**: 60’

**Aufgabe**: Ihre Aufgabe ist ein Ratespiel zu programmieren. Der Computer generiert eine zufällige ganze Zahl zwischen 1 und 100. Sie haben fünf Rateversuche um die Zahl zu erraten. Nach jedem ihrer Versuche erhalten sie vom Computer ein Feedback, ob sie richtig geraten haben oder nicht. Das Programm gibt ihnen sogar einen Tipp, ob ihr Rateversuch zu hoch oder zu tief lag.   
Wenn es ihnen innert fünf Versuchen nicht gelingt, wird ihnen mitgeteilt, dass sie verloren haben. Am Ende eines Spiels werden sie gefragt, ob sie nocheinmal Raten möchten oder nicht.

****

****

gemacht

**Aufgabe 12 (Zahlensequenzen ausgeben)**

**Lernziele**: Die Lernenden können gegebene Aufgabenstellungen mit Einbezug von kopf- und fussgesteuerten Schleifen (inkl. Zählschleife) in c#- oder Java-Programmcode realisieren

**Zeit**: 30’

**Aufgabe**: Schreiben Sie ein Programm, das vom Benutzer eine Start- und eine Endzahl endgegennimmt und speichert. Ihre App gibt danach alle Zahlen von der Start- bis zur Endzahl, getrennt durch ein Komma und ein Leerschlag so aus, dass immer 10 Zahlen pro Zeile erscheinen.

Beim Zeilenende um am Schluss sollen aber die Kommas nicht erscheinen.

Verwenden sie dazu den Schleifentyp while(…) { …}

Gibt der Benutzer zum Beispiel folgende Werte ein:

Startzahl : 62

Endzahl : 87

Gibt das Programm als Resultat folgendes aus:

62, 63, 64, 65, 66, 67, 68, 69, 70, 71

72, 73, 74, 75, 76 ,77, 78, 79, 80, 81

82, 83, 84, 85, 86, 87

**Ihre Lösung (Kopieren sie am Schluss ihre Lösung hier hinein):**

|  |
| --- |
| import java.util.Scanner; public class Main {  public static void main(String[] args) {  int startZahl = 0;  int endZahl = 0;   System.*out*.println("Startzahl: ");  Scanner scan1 = new Scanner(System.*in*);  startZahl = scan1.nextInt();   System.*out*.println("Endzahl: ");  Scanner scan2 = new Scanner(System.*in*);  endZahl = scan2.nextInt();   int i = startZahl;  while (i <= endZahl)  {  System.*out*.print(i);  if ((i-startZahl+1) % 10 == 0)  System.*out*.println();  else  if (i!=endZahl)  System.*out*.print(", ");  i++;  }   } } |

gemacht

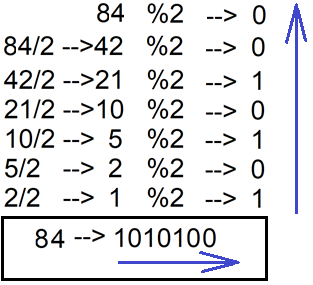
**Aufgabe 13 (Eine Dezimalzahl ins Binärsystem umrechnen)**

**Lernziele**: Die Lernenden können gegebene Aufgabenstellungen mit Einbezug von kopf- und fussgesteuerten Schleifen (inkl. Zählschleife) in c#- oder Java-Programmcode realisieren

**Zeit**: 45’

**Aufgabe**: Schreiben Sie ein Programm, das eine Dezimalzahl zwischen 0 und 99’999 entgegennimmt und diese Zahl im Binärsystem ausgibt.

Gibt man zum Beispiel die Dezimalzahl 328 ein, gibt die App als Resultat 101001000 aus.

**Tipp**: Mann kann dieses Problem auf versch. Arten lösen. In meiner Lösung bin ich so vorgegangen, dass ich die Dezimalzahl (hier z.B. 84) immer zuerst Modulo 2 rechne (ergibt die Binärziffer) und danach mit der ganzzahligen Division durch 2 die Dezimalzahl mindestens halbiere und solange weiterfahre bis die Dezimalzahl 0 ist.

**Ihre Lösung (Kopieren sie am Schluss ihre Lösung hier hinein):**

|  |
| --- |
| public class Main {  public static void main(String[] args) {   String quit = "";   do {   System.*out*.println("Tippe eine Zahl, die du in Binär umrechnen möchtest");   Scanner Zahl = new Scanner(System.*in*);  int n = Zahl.nextInt();  String bin = "";   while (n > 0) {  int lsb = n & 1;   if (lsb != 0)  bin = "1" + bin;  else  bin = "0" + bin;   n >>= 1;  }   System.*out*.println("Ihre eingegebene Zahl in Binär: ");  System.*out*.println(bin);  System.*out*.println("Q to quit");   Scanner scanner2 = new Scanner(System.*in*);   quit = scanner2.nextLine();   if (quit.toLowerCase().equals("q")) {  System.*exit*(0);  }  } while (!quit.equals("q"));  } } |

gemacht

**Aufgabe 14 (ggT und kgV zweier Zahlen bestimmen)**

**Thema:** Eingabe, Verarbeitung (mit Verzweigungen und Schleifen), Ausgabe

**Lernziel**: Die Lernenden können gegebene Aufgabenstellungen mit Einbezug von kopf- und fussgesteuerten Schleifen (inkl. Zählschleife) in c#- oder Java-Programmcode realisieren

**Zeit**: 45’

**Aufgabe**: Bei der Bruchrechnung haben sie den Einsatz des ggT (grösster gemeinsamer Teiler) und des kgV (kleinste gemeinsame Vielfache) gelernt. Schreiben sie eine App, die den Benutzer auffordert zwei ganze Zahlen einzugeben. Berechnen sie danach für eingegebenen Zahlen den ggT und das kgV und geben sie diese Informationen dem Benutzer aus.

Der **ggT** ist die grösste Zahl, durch die sowohl A als auch B ohne Rest teilbar sind. Das **kgV** ist die kleinste Zahl, die sowohl ein Vielfaches der ersten Zahl als auch ein Vielfaches der zweiten Zahl ist.

Für die Berechnung des ggT sollen sie dabei den Algorithmus nach Euklid umsetzen. Dieser funktioniert wie folgt:

Man teilt die grössere durch die kleinere Zahl. Geht die Division auf, ist der Divisor der ggT.

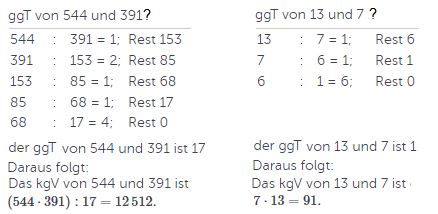
Geht die Division nicht auf, bleibt ein Rest. Dieser Rest ist der neue Divisor. Der alte Divisor wird

zum Dividenden. Nun setzt man das Verfahren fort. Nach endlich vielen Schritten erhält man den

ggT. Da das kleinste gemeinsame Vielfache (kgV) zweier Zahlen der Quotient aus ihrem Produkt

und ihrem ggT ist, kann man auf der Basis des ggT sehr einfach das kgV berechnen.

**Beispiele**:



**Hinweise**: Kürzt man einen Bruch mit dem grössten gemeinsamen Teiler von Zähler und Nenner, entsteht ein Bruch, der nicht weiter kürzbar ist.

Wenn man zwei Brüche mit unterschiedlichem Nenner addieren möchte, muss man die Brüche zuerst gleichnamig machen. Das kleinste gemeinsame Vielfache ist der ideale Nenner dafür.

**Lösung:**

|  |
| --- |
| static void Main(string[] args) {  int zahl1 = 0, zahl2 = 0, ggT = 0, kgV=0;  do {  Console.Write("Zahl 1 eingeben:");  } while (int.TryParse(Console.ReadLine(), out zahl1) == false);  zahl1 = Math.Abs(zahl1); *//Vorzeichen weglassen (Absolutwert)*  do {  Console.Write("Zahl 2 eingeben:");  } while (int.TryParse(Console.ReadLine(), out zahl2) == false);  zahl2 = Math.Abs(zahl2); *//Vorzeichen weglassen (Absolutwert)*  int z1 = zahl1, z2 = zahl2, rest = 0;  while (true) {  rest = z1 % z2;  if (rest == 0)  break;  z1 = z2;  z2 = rest;  }  ggT = z2;  kgV = (zahl1 \* zahl2) / ggT;  Console.WriteLine("Der ggT der beiden Zahlen {0} und {1} ist: {2}", zahl1, zahl2, ggT);  Console.WriteLine("Das kgV der beiden Zahlen {0} und {1} ist: {2}", zahl1, zahl2, kgV);  } |

**Aufgabe 15 (Würfelspiel)**

**Thema:** Eingabe, Verarbeitung (mit Verzweigungen und Schleifen), Ausgabe

**Lernziele**:

* Die Lernenden können gegebene Aufgabenstellungen mit Einbezug von kopf- und fussgesteuerten Schleifen (inkl. Zählschleife) in c#- oder Java-Programmcode realisieren

**Zeit**: 180’

**Aufgabe**: Programmieren sie ein Würfelspiel. Die spielende Person muss zuerst den Namen und die Zielpunktzahl eingeben. Ist die Zielpunktzahl z.B. 100, gewinnt die Person, welche zuerst 100 erreicht hat.

Das Programm entscheidet zufällig, welche/r Spieler/-in beginnen darf.

Spieler\*in 1 beginnt zu würfeln. Dies wird mit zufallszahlen von 1 bis und mit 6 simuliert.

Nach jedem Würfeln muss die würfelnde Person entscheiden, ob sie weiterfürfeln um noch mehr Punkte holen möchte oder ob die in dieser Würfelsequenz erspielten Punkte gesichert werden sollen. Achtung!! Wenn man eine 1 würfelt, verliert man alle in der aktuellen Würfelsequenz erspielten Punkte. Nur die bereits gesicherten Punkte gehen nicht verloren, da sie in Sicherheit seind. Wenn man sich für das Sichern der Punkte entscheidet kommt der/die Gegner\*in an die Reihe.

Erstellen sie das Spiel so, dass sie gegen den Computer spielen können.

**Beispiel**: Jakobus spielt gegen Kunigunde. Sie spielen auf 30. Kunigunde darf beginnen. Sie würfelt eine 3, dann eine 2, eine 4, eine 5 und entscheidet sich danach zur Sicherung der Punkte. Ihr werden 14 Punkte gutgeschrieben. Nun ist Jakobus an der Reihe. Er würfelt eine 6, eine 4 und danach eine 1. Er verliert damit die Punkte der letzten Würfelsequenz (in diesem Fall alle Punkte). Sein Kontostand ist 0 Punkte. Nun ist Kunigunde an der Reihe. Sie würfelt eine 5, eine 3 und danach eine 1. Sie verliert damit die Punkte der letzten Würfelsequenz. Ihr Kontostand bleibt bei 14 und Jakobus ist ander Reihe, etc….. Wer zuerst über den Wert 30 kommt hat gewonnen!

**Ihre Lösung (kopieren Sie am Schluss hier Ihre Lösung hinein):**

|  |
| --- |
|  |