

# Array

Wiederholungsfragen

# Nutzen von Array

- Wann verwendet man ein Array?
  - Wenn man viele Variablen eines Datentyps benötigt.
- Erkläre ein Array
  - Anstelle von einzelnen Variablen, nutzt man ein Feld (Array) mit einer beliebigen Anzahl von Variablen gleichen Typ (Datentyp).

# Initialisierung

- Deklaration eines Arrays
  - `Datentyp[] Bezeichner;`
  - `int[] array`
  - `char[] array`
  - `String[] array`
- Speicherreservierung eines Arrays
  - `Datentyp[] Bezeichner = new Datentyp[Anzahl];`
  - `int[] array = new int[10];`

# Aufgabe

- Erstelle ein Array mit 10 Elementen (Integer)
- Durchlaufe das Array und initialisiere mit 1-10
- Gib mit einer foreach-Schleife den Inhalt der Arrays in der Console aus

# Durchlaufen eines Arrays

- Deklarieren eines Arrays

```
int[] arr = new int[10];
```

- Initialisierung eines Arrays von 1 – 10

```
for(int i = 0; i < arr.Length; i++)  
    arr[i] = i+1;
```

- Ausgabe des Arrays

```
foreach(int item in arr)  
    Console.Write(item);
```

# Direktinitialisierung

- Welche Arten gibt es ein Array direkt zu initialisieren?

# Alles über Arrays:

Detaillierte Zusammenfassung über Arrays

Inklusive Arrays im Speicher

<http://www.introprogramming.info/tag/work-with-sequences-of-elements/>

# Funktionen

Wiederholungsfragen



# Nutzen von Funktionen

- Was ist eine Funktion?
  - Ein Bezeichner für mehrere Anweisungen
  - Ein Bezeichner der mehrere Anweisungen zusammenfasst
  - Dieser Bezeichner kann benutzt werden um diese Anweisungen auszuführen.
- Nutzen einer Funktion?
  - Übersichtlicher
  - Kann beliebig oft genutzt/aufgerufen werden

# Eigenschaften von Funktionen

- Variablen einer Funktion können innerhalb der Funktion genutzt werden, und sind nur innerhalb dieser Funktion gültig
  - Lebensdauer einer Variable -.- > Funktionsvariable
- Lebensdauer einer Variable
  - Von der Speicherreservierung bis zum Ende des Gültigkeitsbereichs
  - Gültigkeitsbereich einer Variable von { bis }
    - `public static void Funktion() { int var = 1; var++; cw(var); }`
  - Es gibt Schleifenvariablen, Funktionsvariablen und Klassenvariablen

# Parameter von Funktionen

- Wie können Variablenwerte zwischen Funktionen ausgetauscht werden?
  - Mit Hilfe von Parameter

# Nutzen von Parameter

- Aufgabe A
  - Funktion1 berechnet die Summe eines Arrays
  - Funktion2 gibt alle Werte des Arrays in der Console aus
  - Wie können beide Funktionen mit dem selben Array arbeiten?
- Aufgabe B
  - Funktion1 liest einen Wert vom Benutzer ein.
  - Diese Funktion wird 2 mal aufgerufen
  - Funktion2 addiert beide Werte und gibt die Summe in der Console aus.

# Die einfache und suboptimale Lösung

- Globale Variablen ----> iiiigitttt ;-)
  - Innerhalb einer Klasse eine (statische, public) Variable deklarieren
  - Gültigkeitsbereich dieser Variable ist auf die gesamte Klasse ausgedehnt und kann von allen Funktionen verändert werden.
- **Lösung:**
- Für eine bessere Struktur und schönere Lösung -> Arbeiten mit Parametern (Werte die einer Funktion übergeben werden)

# Syntax

- Syntax einer Funktion
  - Modifizierer `static` Rückgabetyp  
Bezeichner(Parameterliste){}
- Rückgabewert einer Funktion
  - Datentyp der nach Aufruf der Funktion zurück gegeben wird
- Parameter einer Funktion
  - Werte die an eine Funktion übergeben werden
  - In weiterer Folge: Eingangs- Übergangs- oder Ausgangsparameter

## Funktion ohne Parameter & ohne Rückgabe:

- Erstelle eine Funktion die eine Zeichenkette in der Konsole ausgibt.

```
public static void PrintName()
{
    Console.WriteLine("*****");
    Console.WriteLine("*** Hello Mike Rohsoft ***");
    Console.WriteLine("*****");
}
```

# Funktion mit Parameter & ohne Rückgabewert

- Erstelle eine Funktion die einen Namen als Parameter erhält und eine Zeichenkette in der Konsole ausgibt

```
//Funktion mit Parameter ohne Rückgabewert|
public static void PrintName(String name)
{
    Console.WriteLine("*****");
    Console.WriteLine("*** Hello {0} ***", name);
    Console.WriteLine("*****");
}
```



## Funktion mit 2 Parametern & ohne Rückgabewert

- Übergib Name und Ort als Parameter, gib beides in der Console aus.
- Setze die passende Anzahl an Sternen, und gib diese aus.

```
//Funktion mit 2 Parametern und ohne Rückgabewert
public static void PrintPerson(String name, String location)
{
    String s = String.Format("***  Hallo {0} aus {1} ***", name, location);
    String stars = new String('*', s.Length);
    Console.WriteLine(stars);
    Console.WriteLine(s);
    Console.WriteLine(stars);
}
```

# Funktion mit 2 Parametern & mit Rückgabewert

- Funktion mit 2 Int als Parameter und der Summe als Rückgabewert
- Funktion mit 2 Int als Parameter und dem Produkt als Rückgabewert

//Funktion mit 2 Int als Parameter und der Summe als Rückgabewert

```
public static int Sum(int a, int b)
{
    return a + b;
}
```

//Funktion mit 2 Int als Parameter und dem Produkt als Rückgabewert

```
public static int Mul(int a, int b)
{
    return a * b;
}
```

## Funktion ohne Parameter & ohne Rückgabe:

- Funktion ohne Parameter und Rückgabewert
- Erstelle eine Funktion die eine Zeichenkette in der Konsole ausgibt.

```
public static void PrintName()
{
    Console.WriteLine("*****");
    Console.WriteLine("*** Hello Mike Rohsoft ***");
    Console.WriteLine("*****");
}
```

# Array als Parameterliste


- Es können mehrere Werte in Form von Arrays an eine Funktion übergeben werden:

```
public int GetMaxValue(int[] arr) {  
    int maxValue = arr[0];  
    foreach (int element in arr)  
        if (element > maxValue)  
            maxValue = element;  
    return maxValue;  
}
```

# Beispiel: Summe vom Array

- Erstelle eine Funktion zum Initialisieren eines beliebig großen Arrays
- Erstelle eine Funktion für die Ausgabe eines beliebigen Arrays
- Erstelle Funktion für die Berechnung der Summe eines beliebigen Arrays

# Initialisiere Array

```
//Initialisieren des Array mit einer Funktion  
 (Array als Rückgabewert und Anzahl der Elemente als Parameter  
//Initialisiern mit Random Werten zwischen 0 und 100  
public static int[] InitializeArray(int amount)|  
{  
    int[] arr = new int[amount];  
  
    Random rand = new Random();  
    int min = 0;  
    int max = 101;  
  
    for (int i = 0; i < amount; i++)  
        arr[i] = rand.Next(min, max);  
    return arr;  
}
```

# Summe eines Arrays berechnen

```
//Funktion mit einem Int-Array als Parameter  
//und der Summe aller Werte als Rückgabewert  
public static int SumArray(int[] arr)  
{  
    int sum = 0;  
    foreach (int item in arr)  
        sum += item;  
    return sum;  
}
```

# Ausgabe eines Arrays

```
//Funktion für die Ausgabe eines Arrays
public static void PrintArray(int[] arr)
{
    foreach(int item in arr)
        Console.Write(item + ", ");
    Console.WriteLine();
}
```

```
static void Main(string[] args)
{
    //Array als Parameter
    int[] myArray = InitializeArray(5);
    PrintArray(myArray);
    int sum = SumArray(myArray);
    Console.WriteLine("Die Summe beträgt: {0} ", sum);
}
```



# Wochentag lang

- Erstelle ein Programm, das 4 Werte einliest:  
Wochentag Montag = 1, Sonntag = 7, Tag im Monat mit Wertenzwischen 1 und 31, Monat zwischen 1 und 12 für Jänner bis Dezember und die Jahreszahl.
- Löse die Wochentage mit if/else und die Monate mit switch case. Setze alles in eine Schleife, dass der Benutzer beliebig lange (bis zur Abbruchbedingung) das Programm ausführen kann.
- Diese Werte sollen nun als gesamtes langes Datumsformat in der Console ausgegeben werden.

# Wochentag lang

- **Eingabe im Programm: (4 Zahlenwerte)**
  - Wochentag: 1
  - Tag: 3
  - Monat: 11
  - Jahr 2014
- 
- **Ausgabe in der Konsole:**
  - Montag der 3. November 2014

# Gliedere in Funktionen

- Erstelle eine private Funktion für das Ermitteln des Wochentages
- Erstelle eine private Funktion für das Ermitteln des Monats
- Erstelle eine Funktion, die 4 Werte als Parameter erhält und eine Zeichenkette mit dem Datum Langformat retour bekommt.