**Programmier-Grundkonzepte in C# und .NET (3)**

**Arrays**

Ein **Array** ist eine **Sammlung von Elementen des gleichen Typs**, die über Indizes zugreifbar sind.  
Es bekommt mit der Initialisierung eine feste Größe, kann aber über Array.Resize() geändert werden.

**Beispiel**:

int[] zahlen = new int[5]; // Ein Array mit 5 Plätze

Es bekommt mit der Initialisierung eine feste Größe und verfügt über einen 0-basierten Index (z.B.: zahlen[0] ist das erste Element)).

**Arrays deklarieren und initialisieren**

Leeres Array mit Standardwerten:

int[] zahlen = new int[5]; // Enthält automatisch {0, 0, 0, 0, 0}

Direkte Initialisierung:

int[] zahlen = { 1, 2, 3, 4, 5 };

Elemente setzen und auslesen:

zahlen[0] = 42; // Erstes Element setzen

int wert = zahlen[0]; // Erstes Element lesen

**Durchlaufen eines Arrays**

Mit for-Schleife:

for (int i = 0; i < zahlen.Length; i++)

{

Console.WriteLine(zahlen[i]});

}

**Übung**

Ein Programm, das den Benutzer Zahlen in ein Array eingeben lässt und diese mulztipliziert und anschließend das Ergebnis per Writeline ausgibt. Der Benutzer kann zahlen solange eingeben, bis er statt einer Zahl "A" (Abbruch) eigegeben hat.

P.S. "A" kann auch gleich am Anfang eigegeben werden.

**Mehrdimensionale Arrays**

**2D-Arrays (Matrizen)**

Ein zweidimensionales Array ist eine Tabelle mit Zeilen und Spalten:

int[,] matrix = new int[3, 3]; // 3x3 Matrix

**Gemeinsame Übung**

int[,] matrix =

{

{1, 2, 3},

{4, 5, 6},

{7, 8, 9}

};

Durchlaufen mit verschachtelten for-Schleifen:

for (int zeile = 0; zeile < matrix.GetLength(0); zeile++)

{

for (int spalte = 0; spalte < matrix.GetLength(1); spalte++)

{

Console.Write(matrix[zeile, spalte] + " ");

}

Console.WriteLine();

}

**Übung**

Erstellen ein Programm, das ein 5x5-Array mit Zufallszahlen (aus dem Rang: 1-100) füllt und die Summe der Zahlen berechnet und ausgibt.

**Hinweis**: So erstellen Sie eine Zufallszahl:   
Random rand = new Random(); // nur einmal initialisieren

int zahl = rand.Next(1, 10); // pro gewünschte Zahl aufrufen

**Gemeinsame Übung**

Eine 3x3-Matrix von Typ Char, die Buchstaben von Benutzer entgegennimmt.

Nach jeder Eingabe soll der aktuelle Zustand der Matrix in der Konsole visualisiert werden.

**Erweiterte Funktionen für Arrays**

**Array.Sort()** – Array sortieren

Sortiert ein Array aufsteigend:

int[] zahlen = { 5, 2, 8, 1, 3 };

Array.Sort(zahlen); // Ergebnis: { 1, 2, 3, 5, 8 }

Auch mit Strings verwendbar:

string[] namen = { "Anna", "Zoe", "Lukas" };

Array.Sort(namen); // Alphabetisch sortiert

**Array.Reverse()** – Array umdrehen

Kehrt die Reihenfolge um:

int[] zahlen = { 1, 2, 3, 4, 5 };

Array.Reverse(zahlen); // Ergebnis: { 5, 4, 3, 2, 1 }

**Array.IndexOf()** – Index eines Elements finden

Findet die Position eines Werts im Array:

int[] zahlen = { 10, 20, 30, 40 };

int index = Array.IndexOf(zahlen, 30); // index = 2

Falls das Element nicht existiert, gibt die Methode -1 zurück.

**Array.Resize()** – Größe eines Arrays ändern

Erweitert oder verkleinert ein Array:

int[] zahlen = { 1, 2, 3 };

Array.Resize(ref zahlen, 5); // Jetzt mit 5 Elementen (neue sind 0)

**Zufallszahlen**

In C# werden Zufallszahlen mit der Klasse **Random** erzeugt.

**Random-Objekt erstellen**

Random rand = new Random();

Dadurch wird eine **Instanz des Zufallszahlengenerators** erzeugt.

**Zufallszahlen generieren**

**Ganzzahlen (int) innerhalb eines Bereichs:**

int zahl = rand.Next(1, 101); // Zufallszahl zwischen 1 und 100 (exklusive 101)

**Ganzzahlen ohne Obergrenze:**

int zahl = rand.Next(); // Beliebige positive Ganzzahl

**Fließkommazahlen (double) zwischen 0.0 und 1.0:**

double zahl = rand.NextDouble(); // Wert zwischen 0.0 und 1.0

**Fließkommazahlen in einem anderen Bereich (z. B. 1.0 bis 10.0):**

double zahl = rand.NextDouble() \* 9.0 + 1.0; // Skaliert auf Bereich 1.0 - 10.0