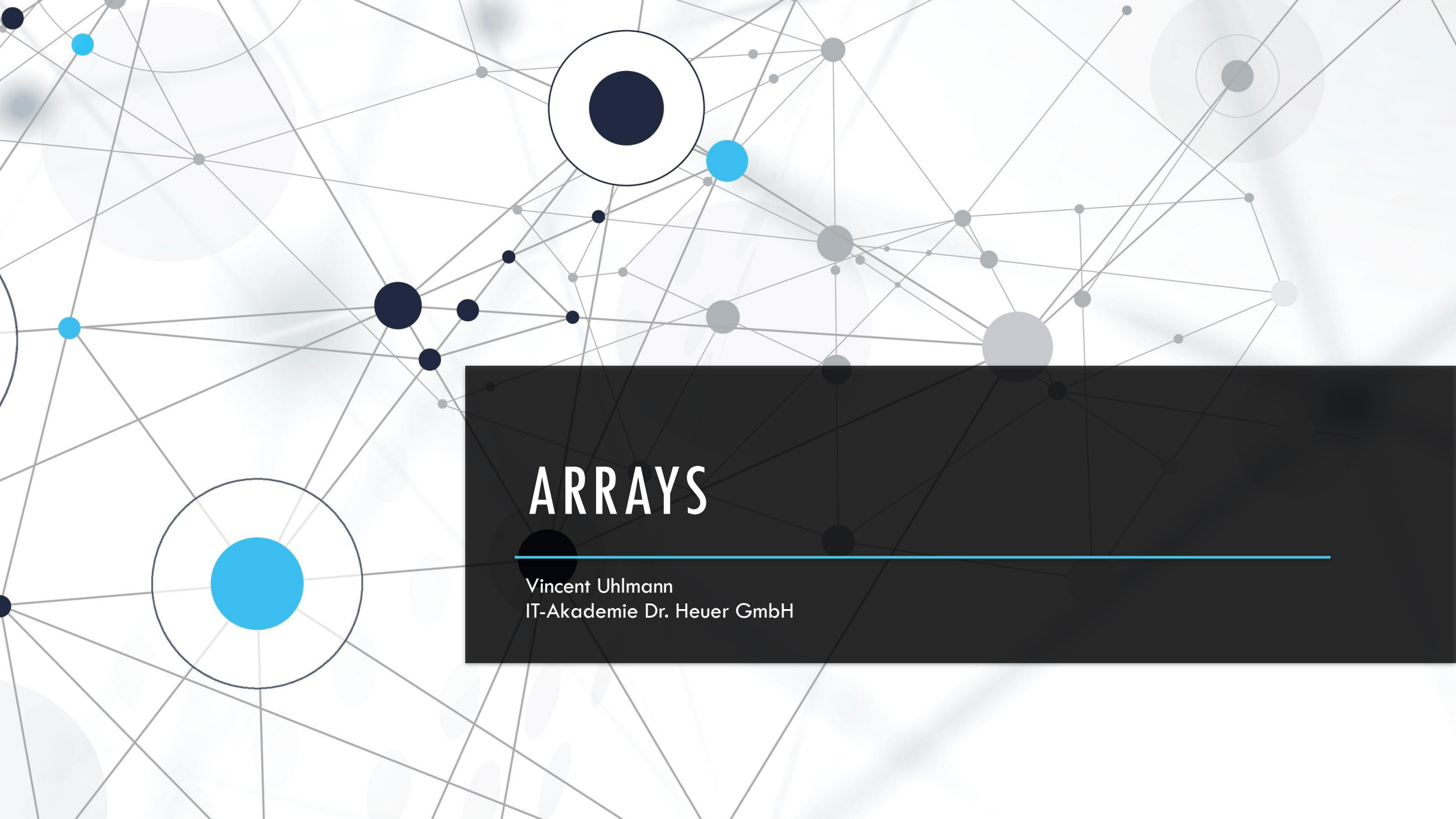

ARRAYS

Vincent Uhlmann
IT-Akademie Dr. Heuer GmbH

EINFÜHRUNG

- Arrays sind spezielle Datentypen, mit denen mehrere Werte desselben Typs in einer einzigen Variablen gespeichert werden können
- Arrays werden verwendet, um eine Sammlung von Elementen desselben Typs zu halten und darauf zuzugreifen
- Effizienter Zugriff auf Elemente über den Index, kompakte Speicherung von Daten

SYNTAX

- Deklaration: Datentyp[] arrayName;
- Statische Initialisierung: Datentyp[] arrayName = {Wert1, Wert2, Wert3, ...};
- Dynamische Initialisierung: Datentyp[] arrayName = new Datentyp[Größe];

```
int[] zahlen = {1, 2, 3, 4, 5};
string[] namen = new string[3];
```

ZUGRIFF

- Für den Zugriff auf Elemente wird der Index verwendet
- Der Index beginnt bei 0 und geht bis zum Wert der Länge -1
- Um die Länge eines Arrays zu bestimmen, verwenden wir den Namen des Arrays, gefolgt von .Length

```
int[] zahlen = {1, 2, 3, 4, 5};
int length = zahlen.Length;
int secondValue = zahlen[1];
zahlen[1] = 4;
```

CONTAINS

- Verwendung der Methode Contains(Wert), um zu prüfen, ob ein Wert im Array vorhanden ist

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

if(zahlen.Contains(4))
{
    Console.WriteLine("4 ist enthalten");
}
```

MULTIDIMENSIONALE ARRAYS

- Erweitern die Konzepte von Arrays, um Daten in mehreren Dimensionen zu organisieren
- Können als Tabellen oder Matrizen betrachtet werden, die in Zeilen und Spalten (oder mehr) organisiert sind
- Werden häufig für komplexe Datenstrukturen verwendet, die mehrere Ebenen von Informationen erfordern

ZUGRIFF

- Verwendung der Methode GetLength(int dimension) für die Länge in einer bestimmten Dimension
- .Length gibt uns jetzt die Anzahl aller Elemente im Array

```
int[,] zahlen ={ { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
int length = zahlen.Length;
int lengthDim0 = zahlen.GetLength(0);
int lengthDim1 = zahlen.GetLength(1);
```

SYNTAX 2D

- Deklaration: Datentyp[,] arrayName = new Datentyp[Zeilen, Spalten];
- Statische Initialisierung: Datentyp[,] arrayName = {{Wert1, Wert2 …}, {Wert1, Wert2 …}};
- Dynamische Initialisierung: Datentyp[,] arrayName = new Datentyp[2, 4];

```
int[,] zahlen ={ { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
string[,] namen = new string[3, 3];
```

SYNTAX 3D

- Deklaration: Datentyp[,,] arrayName = new Datentyp[Dimensionen, Zeilen, Spalten];
- Statische Initialisierung: Datentyp[,] arrayName = {{{Wert1, Wert2…}, {Wert 1, Wert2…}}, {{Wert1, Wert2…}, {Wert1, Wert2…}}};
- Dynamische Initialisierung: Datentyp[,,] arrayName = new Datentyp[3, 3, 3];

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

JAGGED ARRAYS

- Ein Jagged Array ist ein Array, dessen Elemente Arrays sind
- Wird auch „Array aus Arrays“ genannt
- Deklaration: Datentyp[][] arrayName = new Datentyp[Anzahl Arrays][];
- Statische Initialisierung: Datentyp[][] arrayName = { new Datentyp[Größe] { Wert1, Wert2…}, new Datentyp[Größe] { Wert1, Wert2, Wert3…}};
- Dynamische Initialisierung: Datentyp[][] arrayName = new Datentyp[2][];

```
int[][] zahlen = new int[2][];
zahlen[0] = new int[5];
zahlen[1] = new int[4];
```

```
string[][] namen = {
    new string[2] { "", "" },
    new string[3] { "", "", "" }
};
```