# Rectangular vs. Jagged Arrays

- bei einem Rectangular Array liegen alle Elemente direkt hintereinander in einem zusammenhängenden Speicherbereich.
- bei einem Jagged Array sind die Elemente selbst lediglich Referenzen auf andere Array-Objekte. Die Daten liegen also "verstreut" im Speicher.
    - diese Form steht in den meisten Programmiersprachen zur Verfügung

```
// Rectangular Array mit den Dimensionen 3 und 4.
// Dieses Array legt seine 3 * 4 = 12 int Werte in einem
zusammenhängenden Speicherbereich direkt hintereinander ab.
int[,] table = new int[3, 4];
table[1, 2] = 10; // Legt Wert 10 in zweite Zelle (1,2) ab.
table.Rank // => 2 (Rank = Anzahl Dimensionen)
table.Length // => 3 * 4 = 12 Elemente
table.GetLength(0) // => 3 (Größe der ersten Dimension)
table.GetLength(1) // => 4 (Größe der zweiten Dimension)

// Nummeriert die Zellen der Tabelle aufsteigend (0, 1, 2, ...,
Length - 1)
for (int row = 0; row < table.GetLength(0); row++)
{
        for (int column = 0; column < table.GetLength(1); column++)
        {
                table[row, column] = row * table.GetLength(1) +
column;
        }
}

// Ein Jagged Array ist ein eindimensionales Array, dessen Elemente
selbst Arrays sind.
// Achtung: Die erste [] bezieht sich auf das äußere und die zweite
[] auf das innere Array.
int[][] table = new int[3][];
table[0] = new int[4]; // erstes Element ist Array von 4 ints
table[1] = new int[2]; // zweites Element ist Array von 2 ints
table[2] = new int[3]; // drittes Element ist Array von 3 ints
table.Length // => 3
table.Rank // => 1
table[0].Length // => 4
table[1].Length // => 2
```

```
table[2].Length // => 3
table[1][0] = 10; // Weist dem ersten int-Element des zweiten Array
Elements den Wert 10 zu.

// Nummeriert alle ints innerhalb des Jagged Arrays aufsteigend.
int index = 0;
for (int row = 0; row < table.Length; row++)
{
        for (int column = 0; column < table[row].Length; column++)
        {
                table[row][column] = index++;
        }
}
```

# Nullability in Kombination mit Arrays

```
// Fragezeichen nach der Elementtyp-Deklaration bezieht sich immer
auf die
// Nullability der Variable selbst.
// int[] numbers = null;    // Warnung: numbers darf eigentlich nicht
null sein
// int[]? numbers = null;  // OK: numbers darf null sein

int?[]? nullNumbers = new int?[3]; // OK: nullNumbers darf selbst
null sein, aber auch ihre Elemente.
nullNumbers[0] = 1;
nullNumbers[1] = null;

// Ein Array, dessen Elemente (int-Arrays) null sein dürfen.
int[]?[] arrayOfIntArrays = { null, new int[] { 1, 2, 3 }, null };
arrayOfIntArrays[0] = null; // OK
```