

Arrays

Mehrere Werte gleichen Typs
zusammenfassen und speichern.

Übersicht Array - Teil 1

- Definition des Begriffs Array
- Deklaration & Initialisierung von Arrays
- Zugriff auf Array-Elemente
- Kontrollstruktur: Foreach-Schleife
- Arrays kopieren & Arrays sortieren
- Suchen in Arrays
 - Lineare und binäre Suche



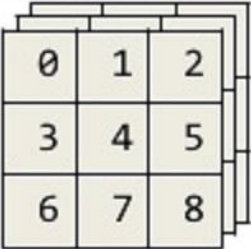
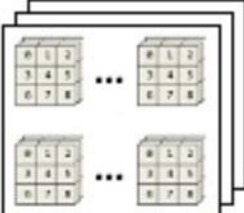
Übersicht Array - Teil 2

- Abbildung von Arrays im Speicher
- Zweidimensionale Arrays
- Dreidimensionale Arrays
- Jagged Arrays (ausgefanst)
- Wiederholungsfragen

Arrays oder Datenfelder

- definieren eine beliebig große Anzahl von Variablen gleichen Namens und gleichen Datentyps
- Unterschieden werden die einzelnen Elemente nur anhand einer Indizierung
- Arrays kommen dann zum Einsatz:
 - wenn eine Operationen auf alle oder einen Teil der Elemente ausgeführt werden soll

Was ist ein Array?

Dimensions	Example	Terminology
1		Vector
2		Matrix
3		3D Array (3 rd order Tensor)
N		ND Array

Arrays: Deklaration

- Arrays werden mit Hilfe der folgenden Syntax deklariert:

```
<type>[] identifier;
```

- Deklaration eines Integer Arrays

```
int[] elements;
```

- Der anfängliche Wert eines Arrays ist null.

Array: Deklaration und *Instantiierung*

- Ein Arrayobjekt wird mit Hilfe von **new** erstellt:

```
int[] elements;  
elements = new int[3];
```

```
int[] store = new int[50];  
string[] names = new string[50];
```

- Die Anzahl der Elemente eines Arrays ergibt sich aus der Angabe in den eckigen Klammern bei der Initialisierung mit **new**.
- Alle Elemente dieses Arrays sind danach mit dem Wert 0 vorinitialisiert.

Array: Deklaration und Initialisierung

- Deklaration mit Initialisierung wenn die Werte bereits bekannt sind nennt sich **literale Initialisierung**:

- Daten in geschweiften Klammern

```
int[] elements = new int[3]{23, 9, 7};
```

- Gleichwertig ist auch diese Initialisierung:

```
int[] elements = new int[]{23, 9, 7};
```

- Besonders kurz:

```
int[] elements = {23, 9, 7};
```

- **falsche literale Initialisierung**

```
int[] elements = new int[3]{23};
```


Zugriff auf Array-Elemente

- Elemente werden durchnummeriert
- erstes Element -> Index 0
- letztes Element -> Index Anzahl der Elemente - 1

- Array deklariert und instantiiert:

```
int[] elements = new int[3];
```

- enthält drei Elemente:

```
elements[0]
```

```
elements[1]
```

```
elements[2]
```

Zugriff auf Array-Elemente

- Werte an eine Stelle im Array speichern:
- erstem Element wird die Zahl 55 zugewiesen:

```
elements[0] = 55;
```

- Auswertung des Elementinhalts ebenfalls durch die Angabe des Index:

```
int value = elements[0];
```

Beispiel: Array mit großen Zahlen

- Erstelle ein Programm, das in ein Array vom Datentyp long die Zahlen 230, 4711 und 77 speichert.
- Anschließend werden diese 3 Werte in der Konsole ausgegeben.

Lösung: Array mit großen Zahlen

Deklaration und Instantiierung:

```
long[] lngVar = new long[4];
```

Wertzuweisung:

```
lngVar[0] = 230;  
lngVar[1] = 4711;  
lngVar[3] = 77;
```

Ausgabe mit Schleife?

Ausgabe:

```
for (int i = 0; i < lngVar.Length; i++)  
    Console.WriteLine("lngVar[{0}] = {1}", i, lngVar[i]);
```

Beispiel: Array mit Zeichenketten

- Erstelle ein Programm, das in ein Array vom Datentyp string die Werte: „C# „ und „macht Spaß!“ speichert.
- Anschließend werden diese 2 Werte in der Konsole in einer Zeile ausgegeben.

Beispiel: Array mit Zeichenketten

Deklaration und Instantiierung:

```
string[] strArr = new String[2];
```

Wertzuweisung:

```
strArr[0] = "C# ";  
strArr[1] = "macht Spaß!";
```

Ausgabe:

```
Console.Write(strArr[0]);  
Console.WriteLine(strArr[1]);
```

Foreach-Schleife

SYNTAX:

```
foreach(Datentyp Bezeichner in Array-Bezeichner) {  
    [...]  
}  
  
int[] elements = {2,4,6,8};  
foreach(int item in elements) {  
    Console.WriteLine(item);  
}
```

- Bei der Iteration wird item jedes Mal auf ein anderes Array-Element verweisen
 - statt über den Index jedes Element anzusprechen
 - wird das Array als eine Einheit angesehen
 - gebildet aus mehreren typgleichen Elementen

Beispiel: Array Durchschnittstemperatur

- Erstelle ein Programm, das 31 Werte von -12 bis +16 in ein Array speichert, das sind die Temperaturwerte im Januar.
- Anschließend durchlaufe das Array und berechne die Durchschnittstemperatur im Januar.

Durchschnittstemperatur:

```
// Temperaturen im Monat Januar
int[] temperaturen = {7, 9, 5, 4, 0, -1, -5,
                     -7, -12, -9, -5, 0, -1, -2,
                     4, 8, 15, 18, 15, 15, 14,
                     16, 12, 10, 5, 3, 6, 5,
                     8, 7, 7};

double dDurchschnitt = 0;

for (int i = 0; i < temperaturen.Length; ++i)
{
    dDurchschnitt += temperaturen[i];
}

dDurchschnitt /= temperaturen.Length;

Console.WriteLine("Durchschnittstemperatur im Januar: "
                  + "{0:F2} Grad", dDurchschnitt);
```

Durchschnitt mit Random

```
static void Main(string[] args) {  
    Random rand = new Random();  
    int amount = 31;  
    int[] temp = new int[amount];  
    int sum = 0;  
    double average;  
    for (int i = 0; i < amount; i++) {  
        temp[i] = rand.Next(-12, 17);  
    }  
    Console.WriteLine("Temperaturwerte:");  
    foreach (int item in temp) {  
        Console.Write(item + "\t ");  
        sum += item;  
    }  
    average = sum / amount;  
    Console.WriteLine("\nDurchschnittstemperatur: " + average);  
}
```

Beispiel: $x*x$

- Erstelle ein Array und fülle das Array mit einer for-Schleife. Das erste (Position 0) Element mit $1*1$, das zweite mit $2*2$, usw. Gib anschließend alle Werte im Array mit einer

```
f( int[] zahlen = new int[10];  
  
   for (int i = 0; i < zahlen.Length; ++i)  
   {  
       zahlen[i] = (i+1)*(i+1);  
   }  
  
   foreach (int elem in zahlen)  
   {  
       Console.WriteLine(" " + elem);  
   }
```

Name & Alter

- Lese vom Benutzer eine Zahl ein, lege 2 Arrays: eines für Namen, eines für Alter fest.
- Lese vom Benutzer die Anzahl von Personen mit deren Namen und Alter ein, die er angegeben hat.
- Berechne die Summe aller Altersangaben, den Durchschnitt. Gib den jüngsten und den Ältesten mit Name und Alter aus.

```
Wieviele Personen möchten Sie auflisten? 3
Name: Kurt
Alter: 8
Name: Max
Alter: 9
Name: Susi
Alter: 7
Name: Kurt Alter: 8
Name: Max Alter: 9
Name: Susi Alter: 7
Summe aller Alterswerte:      24
Durchschnittsalter:          8
Name & Alter der jüngsten Person: Susi ist 7 alt.
Name & Alter der ältesten Person: Max ist 9 alt.
```

Name & Alter

```
Console.Write("Wieviele Personen möchten Sie auflisten? ");
int amount = Int32.Parse(Console.ReadLine());
int[] ages = new int[amount];
string[] names = new string[amount];
int sum = 0;
double average;
int min=0;
int max=0;
for (int i = 0; i < amount; i++) {
    Console.Write("Name: ");
    names[i] = Console.ReadLine();
    Console.Write("Alter: ");
    ages[i] = Int32.Parse(Console.ReadLine());
}
for (int i = 0; i < amount; i++) {
    Console.WriteLine($"Name: {names[i]} Alter: {ages[i]}");
    sum += ages[i];
    if (ages[i] < ages[min])
        min = i;
    if (ages[i] > ages[max])
        max = i;
}
average = sum / amount;
Console.WriteLine($"Summe aller Alterswerte: \t{sum}");
Console.WriteLine($"Durchschnittsalter: \t{average}");
Console.WriteLine($"Name & Alter der jüngsten Person: {names[min]} ist {ages[min]} alt.");
Console.WriteLine($"Name & Alter der ältesten Person: {names[max]} ist {ages[max]} alt.");
```

```
Wieviele Personen möchten Sie auflisten? 3
Name: Kurt
Alter: 8
Name: Max
Alter: 9
Name: Susi
Alter: 7
Name: Kurt Alter: 8
Name: Max Alter: 9
Name: Susi Alter: 7
Summe aller Alterswerte:      24
Durchschnittsalter:          8
Name & Alter der jüngsten Person: Susi ist 7 alt.
Name & Alter der ältesten Person: Max ist 9 alt.
```

Array umdrehen

- Erstelle ein Array, speichere 10 zufällige Zahlen.
- Erstelle ein zweites Array, speichere dort die Zahlen des ersten Arrays in umgekehrter Reihenfolge.
- Gib beide Arrays in der Konsole aus.
- Ergänzung: Versuche nur 1 Array zu verwenden und trotzdem die Reihenfolge zu ändern.

Array Umdrehen

```
int[] array = { 3, 4, 5, 6, 7 };  
for (int i = 0; i < array.Length; i++) {  
    Console.Write(array[i] + ", ");  
}  
Console.WriteLine();  
  
int j = array.Length-1;  
for (int i = 0; i < array.Length/2; i++) {  
    int number = array[i];  
    array[i] = array[j];  
    array[j] = number;  
    j--;  
}  
  
for (int i = 0; i < array.Length; i++) {  
    Console.Write(array[i] + ", ");  
}
```

Arrayteile löschen

- Löschen im Sinne von aus dem Array entfernen ist nicht möglich - es können die Werte im Array überschrieben werden bzw auf einen Standardwert zurückgesetzt werden: Nullwert (0, false oder null)

```
for (int i = 3; i < 7; ++i)
    meinArray[i] = 0;
```

System.Array-Methode Clear ist ebenfalls verfügbar:

```
Array.Clear(meinArray, 3, 4);
```


Array kopieren - clonen

- Ein ganzes Array kopieren mit der Methode `Clone()`

```
int[] einArray = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
int[] anderesArray = (int[]) einArray.Clone();
```

- Arrays in andere Arrays einfügen mit `CopyTo()`

```
int[] arrayA = {1, 2, 3, 4, 5};  
int[] arrayB = {2, 2, 2, 2, 2, 2, 2, 2, 2, 2};  
arrayA.CopyTo(arrayB, 3);
```

Ausgabe:

'2 2 2 1 2 3 4 5 2 2

arrayB muss groß genug sein!!!

Beispiel mit Strings

- Erstelle ein Array mit 4 Namen, initialisiere direkt bei der Deklaration. Durchlaufe anschließend mit einer foreach-Schleife das Array für die Ausgabe der Namen in der Konsole

```
String[] namen = { "Mike Rohsoft", "Sue Permarkt",  
                  "Roman Tick", "Mario Nette" };  
  
foreach (String name in namen)  
    Console.WriteLine(name);
```

```
Mike Rohsoft  
Sue Permarkt  
Roman Tick  
Mario Nette
```

Array sortieren

- Arrays können mit der statischen Array-Methode `Array.Sort()` sortiert werden.
- enthaltene Elemente werden aufsteigend sortiert
- `Array.Sort(namen);`
- Die Reihenfolge umdrehen funktioniert mit der statischen Array-Methode `Array.Reverse()`.
- `Array.Reverse(namen);`

Arrays sortieren und umkehren

```
String[] namen = { "Mike Rohsoft", "Sue Permarkt",  
                  "Roman Tick", "Mario Nette" };  
  
Array.Sort(namen);  
//Array.Reverse(namen);  
foreach (String name in namen)  
    Console.WriteLine(name);
```

```
Mike Rohsoft  
Sue Permarkt  
Roman Tick  
Mario Nette
```

```
Mario Nette  
Mike Rohsoft  
Roman Tick  
Sue Permarkt
```

```
Sue Permarkt  
Roman Tick  
Mike Rohsoft  
Mario Nette
```

Sortiertes Array durchsuchen

- Voraussetzung für schnelles Suchen in einem Array:
 - Array muss sortiert sein
 - Binäre Suche kann genutzt werden
 - (wie im Telefonbuch - aufschlagen und im Alphabet weiter vorne - links weitersuchen, sonst rechts weitersuchen)
- System.Array stellt die Methode: `BinarySearch()` zur Verfügung.
 - Als Ergebnis erhält man den Index des Elements oder einen Wert kleiner 0, wenn der gesuchte Wert nicht gefunden wurde.

```
Array.BinarySearch(namen, „Mike Rohsoft“);
```

Binäre Suche - ein Beispiel

- Baue die Binäre Suche wie folgt in unser Programm ein:

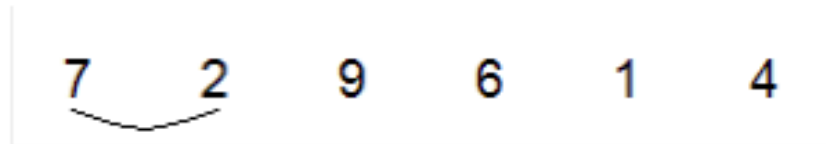
```
Mario Nette  
Mike Rohsoft  
Roman Tick  
Sue Permarkt  
Geben Sie den zu suchenden Namen ein:  
Mike Rohsoft  
Name gefunden an Position 1
```

Elemente im Array suchen...

```
String[] namen = { "Mike Rohsoft", "Sue Permarkt",  
                  "Roman Tick", "Mario Nette" };  
  
Array.Sort(namen);  
//Array.Reverse(namen);  
foreach (String name in namen)  
    Console.WriteLine(name);  
  
Console.WriteLine("Geben Sie den zu suchenden Namen ein: ");  
string sName = Console.ReadLine();  
  
int iPos = Array.BinarySearch(namen, sName);  
if (iPos >= 0)  
    Console.WriteLine("Name gefunden an Position " + iPos);  
else  
    Console.WriteLine("Name nicht gefunden ");
```

Eigene Sortialgorithmen schreiben: Bubblesort

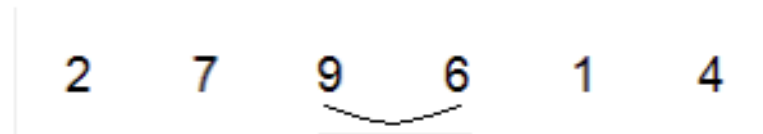
Im Prinzip gilt es folgendes Problem zu lösen. Im einfachsten Fall muss ein gegebenes Feld von z.B. Zahlen in aufsteigender Reihenfolge sortiert werden.



1. Nimm die ersten zwei Zahlen hier 7 und 2. Siehe Bild oben.
2. Vergleiche ob Zahl A größer ist als Zahl B.
3. Wenn ja, Vertausche die Zahlen. Nimm die Nächsten zwei Zahlen.
4. Wenn nein, nimm die Nächsten zwei Zahlen.

Bubblesort

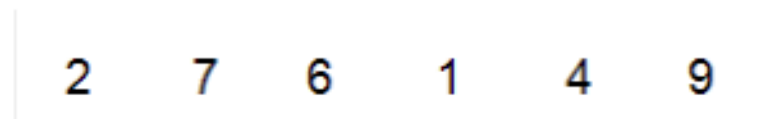
Hier sind mit $A=7$ und $B=9$, A nicht $> B$ also fahren wir mit Punkt 4 fort und vergleichen das nächste Zahlenpaar.



$A=9$ ist größer als $B=6$ also werden sie vertauscht und die nächsten zwei Zahlen werden verglichen.



So vergleichen jetzt bis zum Ende bis das Feld so aussieht.



Bubblesort

So vergleichen jetzt bis zum Ende bis das Feld so aussieht.



Wie man sieht haben wir die Zahlen Nach einer Iteration nicht endgültig sortiert. Also wiederholen wir diesen Vorgang solange bis keine Vertauschungen mehr nötig sind...



BubbleSort

```
class BubbleSort
{
    internal static int[] bubbleSortFunction(int[] list)
    {
        bool pairSorted;

        //solange nicht alle paare bei jedem Durchlauf
        //sortiert sind, Alg. wiederholen.
        //->BubbleSort verfahren
        do
        {
            pairSorted = true;

            for (int i = 0; i < list.Length - 1; i++)
            {
                if (list[i] > list[i + 1])
                {
                    //zahlen tauschen (nur ein Paar)
                    int temp = list[i];
                    list[i] = list[i + 1];
                    list[i + 1] = temp;

                    //nicht sortiert
                    pairSorted = false;
                }
            }
        } while (!pairSorted);

        //Zurückgeben der sortieren Liste
        return list;
    }
}
```

Arrays - Teil 2

Speicherreservierung eindimensionaler Arrays

Mehrdimensionale Arrays

Übersicht Array - Teil 2

- Abbildung von Arrays im Speicher
- Mehrdimensionale Arrays:
 - Zweidimensional
 - Dreidimensional
 - Mehrdimensional
 - Jagged Arrays vs. Blockarrays
(ausgefanst vs. rechteckig)
- Wiederholungsfragen

Eigenschaften von Arrays

- Ein Array kann **eindimensional**, **mehrdimensional** oder **verzweigt** sein.
- Größe des Arrays wird festgelegt,
 - wenn die Arrayinstanz erstellt wird
 - diese Größe kann während der Lebensdauer der Instanz nicht geändert werden
- Numerische Arrayelemente sind
 - standardmäßig auf 0 (Null) festgelegt,
 - Verweiselemente auf NULL.

Eigenschaften von Arrays

- verzweigte Arrays sind ein Array von Arrays,
 - deshalb sind seine Elemente Referenztypen und werden mit null initialisiert.
- Arrays sind nullbasiert:
 - Index eines Arrays mit n Elementen beginnt bei 0 und endet bei n-1.
- Arrayelemente können einen beliebigen Typ aufweisen

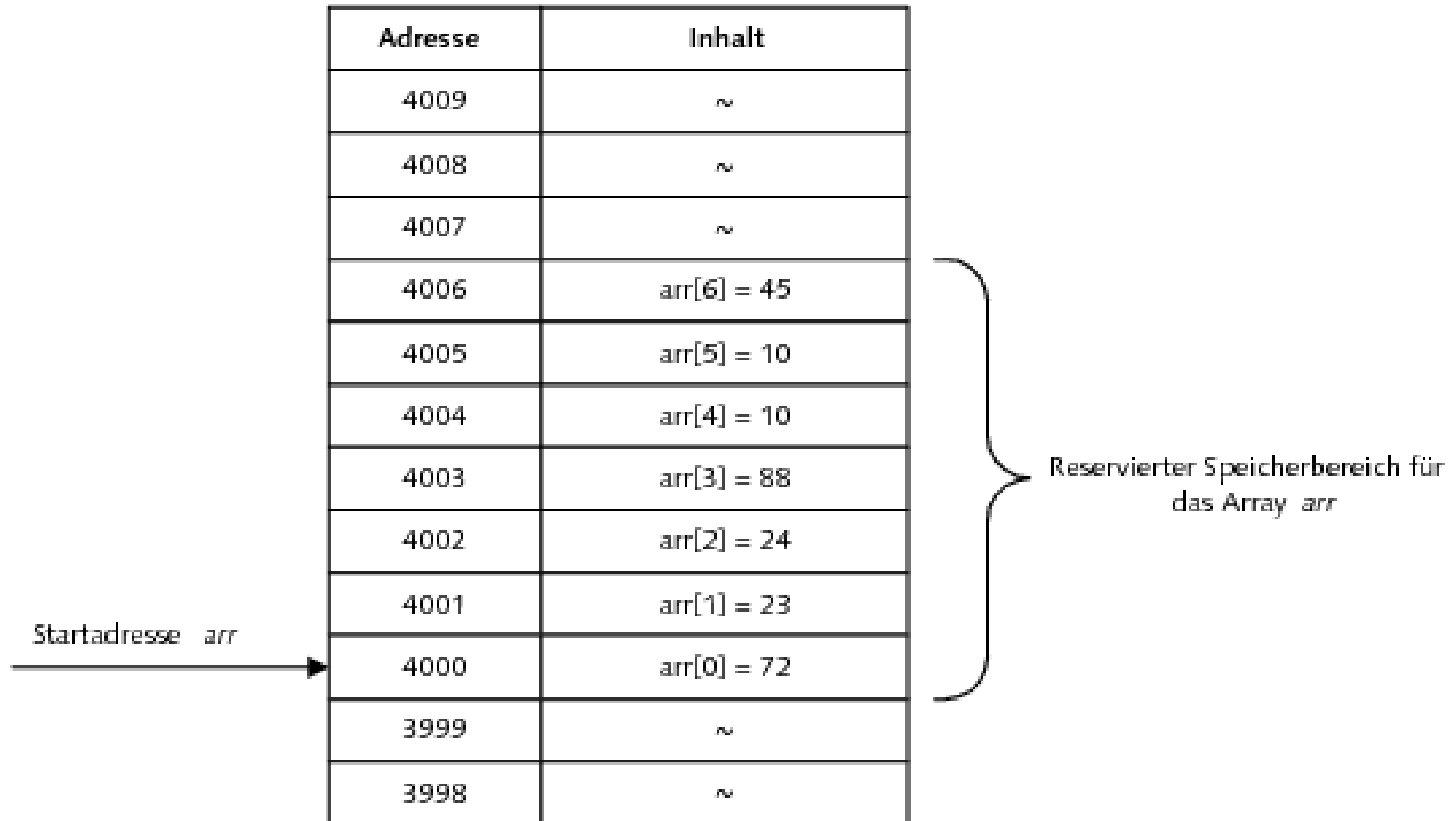
Speicherbereich für Arrays reservieren

- Array beschreibt eine Anzahl typgleicher **Elemente, die im Speicher aufeinanderfolgen**
- Angenommen, es sei das Array `arr` vom Typ `int` deklariert und den einzelnen Feldelementen werden die folgenden Werte zugewiesen:

```
int[] arr = new int[7];
```

```
arr[0] = 72; arr[1] = 23; arr[2] =  
24; arr[3] = 88; arr[4] = 10; arr[5]  
= 10; arr[6] = 45;
```


Abbildung im Speicher



Beispiel: Array anlegen, Werte ausgeben

```
int[] arr = new int[7];  
arr[0] = 72; arr[1] = 23; arr[2] = 24; arr[3] = 88;  
arr[4] = 10; arr[5] = 10; arr[6] = 45;  
// Zuweisung des Arrays arr an das neue Array newArr  
int[] newArr = arr;
```

```
// Konsolenausgabe  
Console.WriteLine("newArr[0] = {0}", newArr[0]);  
Console.WriteLine("newArr[1] = {0}", newArr[1]);  
Console.WriteLine("newArr[2] = {0}", newArr[2]);  
Console.WriteLine("newArr[3] = {0}", newArr[3]);  
Console.WriteLine("newArr[4] = {0}", newArr[4]);  
Console.WriteLine("newArr[5] = {0}", newArr[5]);  
Console.WriteLine("newArr[6] = {0}", newArr[6]);
```

Zuweisen eines Wertes

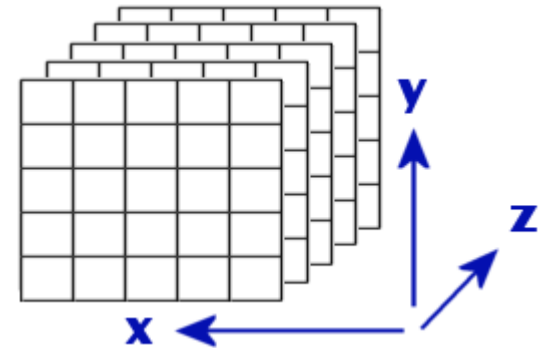
- Direkter Zugriff auf die Adresse:
 - Zuweisung eines Werts an ein Element des Arrays bzw.
 - Auswerten des Inhalts erfolgt direkt
 - auf die Adresse des entsprechenden Elements.
- Ein Array einer zweiten Array-Variablen zuweisen:
 - keine Schaffung einer Kopie (Clone)
 - sondern der Zugriff auf die Originaldaten des ursprünglichen Arrays unter einem anderen Bezeichner

Beispiel: Arrays anderen Arrays zuweisen

```
int[] arr = new int[7];  
arr[0] = 72; arr[1] = 23; arr[2] = 24; arr[3] = 88;  
arr[4] = 10; arr[5] = 10; arr[6] = 45;  
// Zuweisung des Arrays arr an das neue Array newArr  
int[] newArr = arr;  
newArr[3] = 7777;  
arr[5] = 7777;  
// Konsolenausgabe  
Console.WriteLine("newArr[0] = {0}", newArr[0]);  
Console.WriteLine("newArr[1] = {0}", newArr[1]);  
Console.WriteLine("newArr[2] = {0}", newArr[2]);  
Console.WriteLine("newArr[3] = {0}", newArr[3]);  
Console.WriteLine("newArr[4] = {0}", newArr[4]);  
Console.WriteLine("newArr[5] = {0}", newArr[5]);  
Console.WriteLine("newArr[6] = {0}", newArr[6]);
```

Wie lautet die Ausgabe?

```
newArr[0] = 72  
newArr[1] = 23  
newArr[2] = 24  
newArr[3] = 7777  
newArr[4] = 10  
newArr[5] = 7777  
newArr[6] = 45
```



Mehrdimensionale Arrays

Arrays können über mehr als eine Dimension verfügen

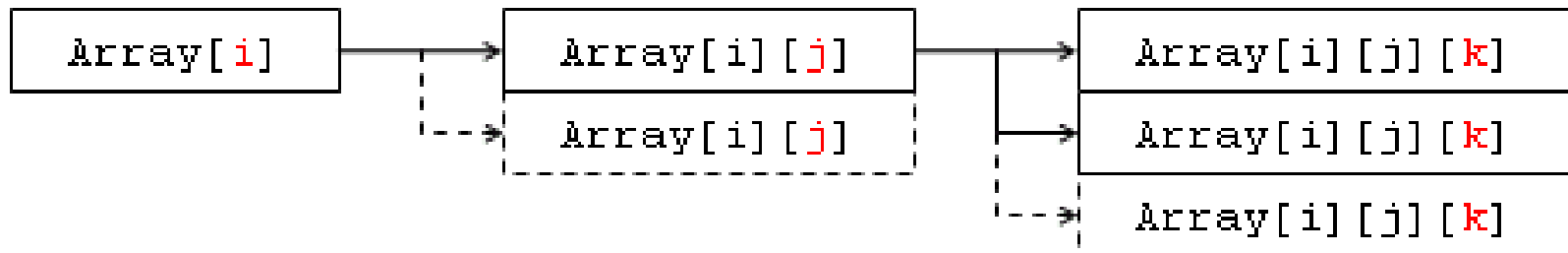
Mehrdimensionale Arrays

- Ein Array kann mehrdimensional sein:

1. Dimension

2. Dimension

3. Dimension



Arrays mit mehr Dimensionen

- Arrays können über mehr als eine Dimension verfügen.

- z. B. ein zweidimensionales Array

- mit vier Zeilen und zwei Spalten erstellt

```
int[, ] array = new int[4, 2];
```

- z. B. Array mit den drei Dimensionen 4, 2 und 3 erstellt:

- vier Zeilen, zwei Spalten, und 3 Ebenen (Höhe)

```
int[, , ] array1 = new int[4, 2, 3];
```

A 3x3 grid representing a 2D array. The rows are labeled 'Zeilen' (Rows) in blue text on the left, with a bracket indicating the three rows. The columns are labeled 'Spalten' (Columns) in green text at the bottom, with a bracket indicating the three columns. Each cell in the grid contains a coordinate pair (row, column) in green text. The coordinates are: (0,0), (0,1), (0,2) for the first row; (1,0), (1,1), (1,2) for the second row; and (2,0), (2,1), (2,2) for the third row.

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

Zweidimensionale Arrays

besteht aus Zeilen und Spalten

Zweidimensionales Array

- **Syntax:** Deklaration eines zweidimensionalen Arrays `Datentyp[,] Bezeichner;`

```
int[, ] array = new int[4, 3];
```

Array mit
4 Zeilen und 3 Spalten:

	Spalte 0	Spalte 1	Spalte 2
Reihe 0	zelle[0,0]	zelle[0,1]	zelle[0,2]
Reihe 1	zelle[1,0]	zelle[1,1]	zelle[1,2]
Reihe 2	zelle[2,0]	zelle[2,1]	zelle[2,2]
Reihe 3	zelle[3,0]	zelle[3,1]	zelle[3,2]

2	4	3
4	2	4
2	3	5
2	8	6

Deklaration & Initialisierung

- Array deklarieren und Instanzieren
- Array mit Werten befüllen

```
int[,] myarray = new int[3, 3]; // Ein Array mit 3 Zeilen und 3 Spalten
myarray[0, 0] = 1;              // 0.Zeile, 0.Spalte = 1
myarray[0, 1] = 2;              // 0.Zeile, 1.Spalte = 2
myarray[0, 2] = 3;              // 0.Zeile, 2.Spalte = 3
myarray[1, 0] = 4;              // 1.Zeile, 0.Spalte = 4
myarray[1, 1] = 5;              // 1.Zeile, 1.Spalte = 5
myarray[1, 2] = 6;              // 1.Zeile, 2.Spalte = 6
myarray[2, 0] = 7;              // 2.Zeile, 0.Spalte = 7
myarray[2, 1] = 8;              // 2.Zeile, 1.Spalte = 8
myarray[2, 2] = 9;              // 2.Zeile, 2.Spalte = 9
```

```
foreach (int i in myarray)      // Ausgabe aller Elemente des Arrays
{
    Console.WriteLine(i);
}
Console.WriteLine("\n");
```

Ausgabe mit Zeilenumbruch

- Spalteneinträge nebeneinander ausgeben
- Zeilenumbrüche nach dem Ende einer Spalte

```
int col = 2, row = 3;
```

```
int[,] myarray1 = new int[col, row]; // Ein Array mit 2 Zeilen und 3 Spalten
myarray1[0, 0] = 1;                  // 1.Zeile, 1.Spalte = 1
myarray1[0, 1] = 2;                  // 1.Zeile, 2.Spalte = 2
myarray1[0, 2] = 3;                  // 1.Zeile, 3.Spalte = 3
myarray1[1, 0] = 4;                  // 2.Zeile, 1.Spalte = 4
myarray1[1, 1] = 5;                  // 2.Zeile, 2.Spalte = 5
myarray1[1, 2] = 6;                  // 2.Zeile, 3.Spalte = 6
```



123
456

```
for (int i = 0; i < col; i++) {
    for (int j = 0; j < row; j++) {
        Console.Write(myarray1[i, j]);
    }
    Console.WriteLine("\n"); // Zeilenumbruch bei Beginn einer neuen Zeile
}
```

Direkte Initialisierung:

- Ein zweidimensionales Array
 - ist ein Feld, bei dem jedes Array-Element selbst wieder ein eigenes Feld gleichen Typs definiert,
 - wird jedes Element der Initialisierung eines eindimensionalen Arrays durch ein Paar geschweiffter Klammern ersetzt, in dem wiederum Werte des »Unterarrays« angegeben werden:

- Eindimensional:

`{Anzahl der Elemente der ersten Dimension}`

- Zweidimensional:

`{{Anzahl der Elemente der zweiten Dimension}, { }, ...}`

Direkte Initialisierung zweidimensionaler Arrays

- Literale Zuweisung an ein zweidimensionales Array:

```
int[, ] point = new int[, ] { {1, 2, 3}, {4, 5, 6} }
```

- kürzere Schreibweise mit:

```
int[, ] point = { {1, 2, 3}, {4, 5, 6} }
```

- Literale Zuweisung an ein dreidimensionales Array:

```
int[, , ] myArr = { { {1, 2, 3, 4}, {3, 4, 5, 6}, {6, 7, 8, 9} },  
                    { {3, 4, 6, 1}, {6, 19, 3, 4}, {4, 1, 8, 7} } };
```

- myArr entspricht einem Array myArr[2,3,4]
- in der dritten Dimension vier Elemente, in der zweiten drei und in der ersten zwei.

Zweidimensionale Arrays: Beispiel

1. Erstelle ein zweidimensionales Array mit 4 Zeilen und zwei Spalten, fülle die Elemente von 1 bis 8 aufsteigend an. Erstelle eine Lösung mit Deklaration und Initialisierung in einem Schritt.
2. Erstelle ein String-Array mit 3 Zeilen 2 Spalten, und schreibe die Zeichenketten von „eins“ bis „sechs“ mit direkter Initialisierung in das zweidimensionale Array.

```
// Two-dimensional array.  
int[,] array2D = new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };  
// The same array with dimensions specified.  
int[,] array2Da = new int[4, 2] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };  
// A similar array with string elements.
```

Zweidimensionalen Arrays: Beispiel

- Zugriff auf Arrayelemente:

```
arrayName[1,1] = 77;
```

```
// Two-dimensional array.
int[,] array2D = new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };
// The same array with dimensions specified.
int[,] array2Da = new int[4, 2] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };
// A similar array with string elements.
string[,] array2Db = new string[3, 2] { { "one", "two" }, { "three", "four" },
                                           { "five", "six" } };

// Accessing array elements.
System.Console.WriteLine(array2D[0, 0]);
System.Console.WriteLine(array2D[0, 1]);
System.Console.WriteLine(array2D[1, 0]);
System.Console.WriteLine(array2D[1, 1]);
System.Console.WriteLine(array2D[3, 0]);
System.Console.WriteLine(array2Db[1, 0]);
```

Wie lautet
die Ausgabe?

1
2
3
4
7
three

Ausgabe mit Schleifen

```
int rows = 3;
int col = 2;
int[,] arr = { { 1, 2 }, { 3, 4 }, { 5, 6 } };
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < col; j++) {
        Console.Write(arr[i,j]);
    }
    Console.WriteLine();
}
```

12
34
56

```
int rows = 3;
int col = 2;
string[,] arr = { { "one", "two" },
    { "three", "four" }, { "five", "six" } };
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < col; j++) {
        Console.Write(arr[i, j] + "\t ");
    }
    Console.WriteLine();
}
```

one	two
three	four
five	six

Zwei Dimensionales Array

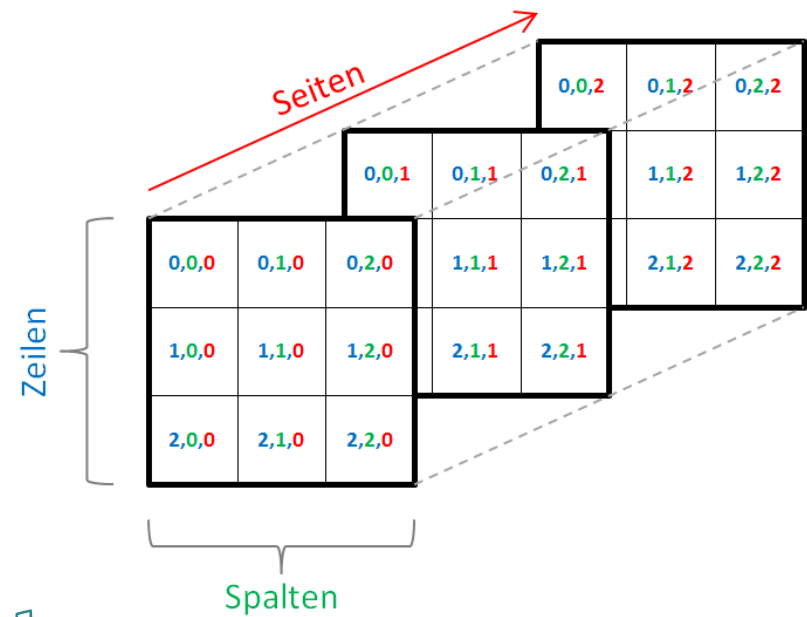
- Lese eine beliebige Zahl ein, addiere 1 falls nötig.
- Befülle in 2 Spalten und beliebig vielen Zeilen ein Array bis zur Zahl die Eingelesen wurde.
- Gib dieses in der Console aus.

```
Anzahl der Werte:
33
1      2
3      4
5      6
7      8
9      10
11     12
13     14
15     16
17     18
19     20
21     22
23     24
25     26
27     28
29     30
31     32
33     34
```

Einlesen & Ausgeben von Arrays

```
Console.WriteLine("Anzahl der Werte:");
int num = Convert.ToInt32(Console.ReadLine());
if (num % 2 != 0) num++;
int row = num / 2;
int col = 2;
int count = 1;
int[,] arr = new int[row, col];
for (int i = 0; i < row; i++) {
    for (int j = 0; j < col; j++) {
        arr[i, j] = count++;
    }
}
for (int i = 0; i < row; i++) {
    for (int j = 0; j < col; j++) {
        Console.Write(arr[i, j] + "\t");
    }
    Console.WriteLine();
}
Console.WriteLine();
```

```
Anzahl der Werte:
33
1      2
3      4
5      6
7      8
9      10
11     12
13     14
15     16
17     18
19     20
21     22
23     24
25     26
27     28
29     30
31     32
33     34
```



Dreidimensional

Zeilen & Spalten und Seiten oder Ebenen

Dreidimensionale Arrays: Beispiel

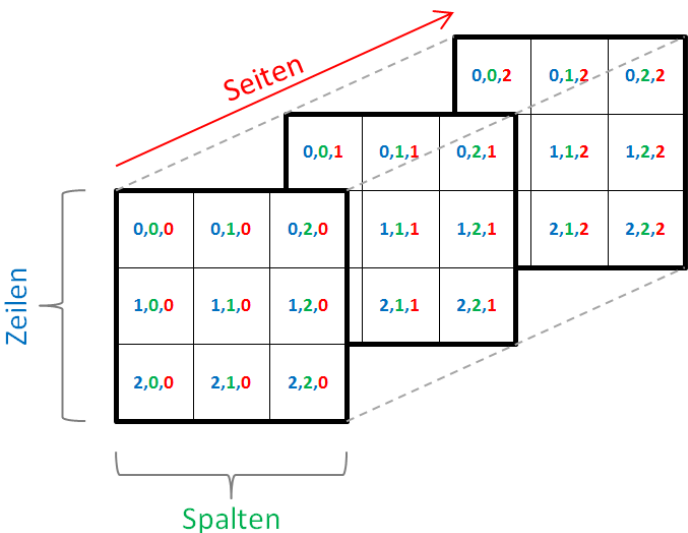
- Zugriff auf ein mehrdimensionales Array:
 - jede Dimension des entsprechenden Elements angeben:
`array3D[1,0,1];`

```
// Three-dimensional array.  
int[, ,] array3D = new int[, ,] { { { 1, 2, 3 }, { 4, 5, 6 } },  
                                   | { { 7, 8, 9 }, { 10, 11, 12 } } };  
  
// The same array with dimensions specified.  
int[, ,] array3Da = new int[2, 2, 3] { { { 1, 2, 3 }, { 4, 5, 6 } },  
                                         { { 7, 8, 9 }, { 10, 11, 12 } } };  
  
System.Console.WriteLine(array3Da[1, 0, 1]);  
System.Console.WriteLine(array3D[1, 1, 2]);
```

Wie lautet
die Ausgabe?

8
12

Direkt Initialisierung verstehen



000 = 1	010 = 4
100 = 7	110 = 10
###	
001 = 2	011 = 5
101 = 8	111 = 11
###	
002 = 3	012 = 6
102 = 9	112 = 12
###	

```
const int lines = 2;
const int columns = 2;
const int pages = 3;

int[, ,] arr = new int[lines, columns, pages] {
    { { 1, 2, 3 }, { 4, 5, 6 } }, { { 7, 8, 9 }, { 10, 11, 12 } }
    // 0 1 2      0 1 2      0 1 2      0 1 2
    // -----, -----
    //      0      1      0      1
    // -----
    //      0      1
};
```

Page 0	Page 0
Z0 S0 - 1	Z0 S1 - 4
Z1 S0 - 7	Z1 S1 - 10
Page 1	Page 1
Z0 S0 - 2	Z0 S1 - 5
Z1 S0 - 8	Z1 S1 - 11
Page 2	Page 2
Z0 S0 - 3	Z0 S1 - 6
Z1 S0 - 9	Z1 S1 - 12

Matrix mit 2x3x4

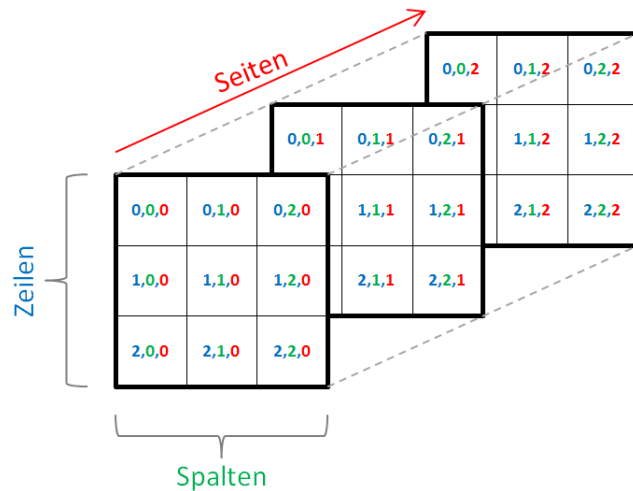
```
const int lines = 2;    //Zeilen
const int columns = 3;  //Spalten
const int pages = 4;    //Ebenen
int[, ,] arr = new int[lines, columns, pages]
{
    { {111,112,113,114},{121,122,123,124},{131,132,133,134} },
    ;
};
```

```
{ {211,212,213,214},{221,222,223,224},{231,232,233,234} }
```

Dreidimensionale Matrix mit 2x3x4

000 =	010 =	020 =
100 =	110 =	120 =
###		
001 =	011 =	021 =
101 =	111 =	121 =
###		
002 =	012 =	022 =
102 =	112 =	122 =
###		
003 =	013 =	023 =
103 =	113 =	123 =
###		

Matrix mit 2x3x1



```
const int lines = 2;    //Zeilen
const int columns = 3;  //Spalten
const int pages = 1;    //Ebenen
int[, ,] arr = new int[lines, columns, pages]
{
    { {111},{121},{131} }, { {211},{221},{231} }
};
```

```
Dreidimensionale Matrix mit 2x3x1
000 = 111    010 = 121    020 = 131
100 = 211    110 = 221    120 = 231
###
```

Ausgabe 2 & 3 Dimensionale Arrays

- Zwei Dimensionen:

```
for (int i = 0; i < lines; i++) {           //Zeile
    for (int j = 0; j < column; j++) {      //Spalte
        Console.Write(arr[i, j]);
    }
    Console.WriteLine();
}
```

- Drei Dimensionen:

```
for (int i = 0; i < pages; i++) {           // 3.) zum Schluß die Seiten
    for (int j = 0; j < lines; j++) {        // 2.) dann die Zeile
        for (int k = 0; k < columns; k++) {  // 1.) Zuerst die Spalten
            Console.Write($" {j}{k}{i} = {myarray3[j, k, i]} ");
        }
        Console.Write("\n");                // Beginn einer neuen Zeile
    }
    Console.Write("\n");                    // Beginn einer neuen Seite
}
```


Ausgabe mit Ebenen

```
const int lines = 2;
const int columns = 2;
const int pages = 3;

int[, ,] arr = new int[lines, columns, pages] {
    { { 1, 2, 3 }, { 4, 5, 6 } }, { { 7, 8, 9 }, { 10, 11, 12 } }
    // 0 1 2      0 1 2      0 1 2      0 1 2
    // -----, -----
    //      0          1          0          1
    //-----
    //          0          1
};
```

```
Console.WriteLine("Dreidimensional: ");
for (int page = 0; page < pages; page++) { //Ebene zuletzt

    for (int line = 0; line < lines; line++) { //Zeile danach
        for (int column = 0; column < columns; column++) { //Spalte zuerst
            Console.Write(arr[line, column, page] + " ");
        }
        Console.WriteLine();
    }
    Console.WriteLine("###");
}
```

```
Dreidimensional:
1 4
7 10
###
2 5
8 11
###
3 6
9 12
###
```

Ausgabe 3 Zeilen 5 Spalten 4 Ebenen

- Initialisiere folgendes Array und setze die Werte 1-8 bunt verstreut.
- Zeichne das Array und dessen Werte auf
- Gib das Array mit 3 verschachtelten Schleifen aus.

```
const int lines = 3;  
const int columns = 5;  
const int pages = 4;  
int[, ,] threeDimensional = new int[lines, columns, pages];
```

```
threeDimensional[0, 0, 0] = 1;  
threeDimensional[0, 1, 0] = 2;  
threeDimensional[0, 2, 0] = 3;  
threeDimensional[0, 3, 0] = 4;  
threeDimensional[0, 4, 0] = 5;  
threeDimensional[1, 1, 1] = 6;  
threeDimensional[2, 2, 2] = 7;  
threeDimensional[2, 2, 3] = 8;
```

Ausgabe 3x5x4 Matrix

- 3 Zeilen 5 Spalten 4 Ebenen

```
const int lines = 3;
const int columns = 5;
const int pages = 4;
int[, ,] threeDimensional = new int[lines, columns, pages];
```

```
threeDimensional[0, 0, 0] = 1;
threeDimensional[0, 1, 0] = 2;
threeDimensional[0, 2, 0] = 3;
threeDimensional[0, 3, 0] = 4;
threeDimensional[0, 4, 0] = 5;
threeDimensional[1, 1, 1] = 6;
threeDimensional[2, 2, 2] = 7;
threeDimensional[2, 2, 3] = 8;
```

```
// Ausgabe Für alle Ebenen,
for (int i = 0; i < pages; i++) {           //Ebenen zuletzt
    //jeweils Zeilenweise alle Spalten
    for (int j = 0; j < lines; j++) {       //Zeile danach
        for (int k = 0; k < columns; k++) { //Spalte zuerst
            Console.Write($" {j}{k}{i} = {threeDimensional[j, k, i]} ");
        }
        Console.WriteLine("");
    }
    Console.WriteLine("-");
}
```

000 = 1	010 = 2	020 = 3	030 = 4	040 = 5
100 = 0	110 = 0	120 = 0	130 = 0	140 = 0
200 = 0	210 = 0	220 = 0	230 = 0	240 = 0
001 = 0	011 = 0	021 = 0	031 = 0	041 = 0
101 = 0	111 = 6	121 = 0	131 = 0	141 = 0
201 = 0	211 = 0	221 = 0	231 = 0	241 = 0
002 = 0	012 = 0	022 = 0	032 = 0	042 = 0
102 = 0	112 = 0	122 = 0	132 = 0	142 = 0
202 = 0	212 = 0	222 = 7	232 = 0	242 = 0
003 = 0	013 = 0	023 = 0	033 = 0	043 = 0
103 = 0	113 = 0	123 = 0	133 = 0	143 = 0
203 = 0	213 = 0	223 = 8	233 = 0	243 = 0

Dreidimensionale Arrays: Länge

- Die Länge eines mehrdimensionalen Arrays berechnen:
 - Erstelle eine Funktion, die `arr.Length` überprüft. Es soll pro Ebene die Länge des eindimensionalen Array abgefragt werden und mit der Anzahl der Zeilen multipliziert werden.
 - Dies sollte dem Wert `arr.Length` entsprechen.
 - Übergib ein dreidimensionales Array als Parameter

Arrays - Arraylänge

Beispiele:

```
int[] a = new int[3];  
Console.WriteLine(a.Length);
```

→ 3

```
int[][] b = new int[3][];  
b[0] = new int[4];  
Console.WriteLine(b.Length);  
Console.WriteLine(b[0].Length);
```

→ 3
→ 4

```
int[,] c = new int[3,4];  
Console.WriteLine(c.Length);  
Console.WriteLine(c.GetLength(0));  
Console.WriteLine(c.GetLength(1));
```

→ 12
→ 3
→ 4

Drei Dimensionale Arrays

```
const int lines = 2;
const int columns = 2;
const int pages = 3;

int[, ,] arr = new int[lines, columns, pages] {
    { { 1, 2, 3 }, { 4, 5, 6 } }, { { 7, 8, 9 }, { 10, 11, 12 } }
    // 0 1 2      0 1 2      0 1 2      0 1 2
    // -----, -----
    //      0          1          0          1
    // -----
    //          0          1
};
```

```
// Ausgabe Für alle Ebenen,
for (int i = 0; i < arr.GetLength(2); i++) {           //Ebene (pages)
    //jeweils Zeilenweise alle Spalten
    for (int j = 0; j < arr.GetLength(0); j++) {        //Zeile (lines)
        for (int k = 0; k < arr.GetLength(1); k++) {    //Spalte (columns)
            Console.Write(" {0} ", arr[j, k, i]);
        }
        Console.WriteLine("");
    }
    Console.WriteLine("###");
}
```

```
000 = 1      010 = 4
100 = 7      110 = 10
###
001 = 2      011 = 5
101 = 8      111 = 11
###
002 = 3      012 = 6
102 = 9      112 = 12
###
```

Dreidimensionale Arrays:

- 2 Zeilen
- 2 Spalten
- 3 Ebenen

```
1 4
7 10
###
2 5
8 11
###
3 6
9 12
###
Länge des Arrays 12
Länge an der Stelle 0: 2
Länge an der Stelle 1: 2
Länge an der Stelle 2: 3
Total: 12
12 equals 12
```

```
const int lines = 2;
const int columns = 2;
const int pages = 3;

int[, ,] arr = new int[lines, columns, pages] {
    { { 1, 2, 3 }, { 4, 5, 6 } }, { { 7, 8, 9 }, { 10, 11, 12 } }
};

Ausgabe
int allLength = arr.Length;
int total = 1;
Console.WriteLine("Länge des Arrays {0}", allLength);

//Für jede Dimension (Rank)
for (int i = 0; i < arr.Rank; i++) {
    //Speichere die Anzahl der Elemente je Dimension mit GetLength
    int lenght = arr.GetLength(i);
    //berechne die Gesamtzahl von speicherbaren Elementen
    total *= lenght;
    Console.WriteLine("Länge an der Stelle {0}: {1}", i, lenght);
}
Console.WriteLine("Total: {0}", total);
Console.WriteLine("{0} equals {1}", allLength, total);
```

Erstelle 3x3x3 Array

Gib das
Array
in der
Konsole
aus

```
11 12 13
14 15 16
17 18 19
```

```
21 22 23
24 25 26
27 28 29
```

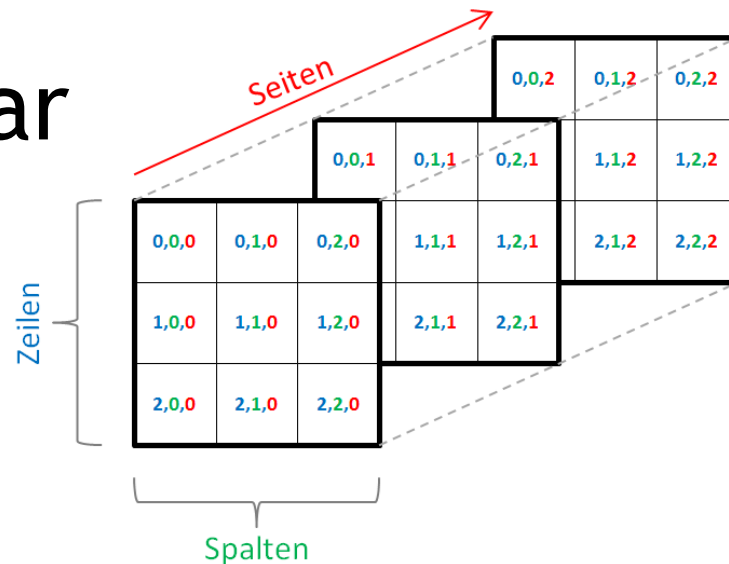
```
31 32 33
34 35 36
37 38 39
```

```
Console.WriteLine("3x3x3 Matrix");
const int lines = 3, columns = 3, pages = 3;
int[, ,] myarray3 = new int[lines, columns, pages];
#region Initialisierung
myarray3[0, 0, 0] = 11;           // 0.Zeile, 0.Spalte, 0.Seite
myarray3[0, 1, 0] = 12;           // 0.Zeile, 1.Spalte, 0.Seite
myarray3[0, 2, 0] = 13;           // 0.Zeile, 2.Spalte, 0.Seite
myarray3[1, 0, 0] = 14;           // 1.Zeile, 0.Spalte, 0.Seite
myarray3[1, 1, 0] = 15;           // 1.Zeile, 1.Spalte, 0.Seite
myarray3[1, 2, 0] = 16;           // 1.Zeile, 2.Spalte, 0.Seite
myarray3[2, 0, 0] = 17;           // 2.Zeile, 0.Spalte, 0.Seite
myarray3[2, 1, 0] = 18;           // 2.Zeile, 1.Spalte, 0.Seite
myarray3[2, 2, 0] = 19;           // 2.Zeile, 2.Spalte, 0.Seite

myarray3[0, 0, 1] = 21;           // 0.Zeile, 0.Spalte, 1.Seite
myarray3[0, 1, 1] = 22;           // 0.Zeile, 1.Spalte, 1.Seite
myarray3[0, 2, 1] = 23;           // 0.Zeile, 2.Spalte, 1.Seite
myarray3[1, 0, 1] = 24;           // 1.Zeile, 0.Spalte, 1.Seite
myarray3[1, 1, 1] = 25;           // 1.Zeile, 1.Spalte, 1.Seite
myarray3[1, 2, 1] = 26;           // 1.Zeile, 2.Spalte, 1.Seite
myarray3[2, 0, 1] = 27;           // 2.Zeile, 0.Spalte, 1.Seite
myarray3[2, 1, 1] = 28;           // 2.Zeile, 1.Spalte, 1.Seite
myarray3[2, 2, 1] = 29;           // 2.Zeile, 2.Spalte, 1.Seite

myarray3[0, 0, 2] = 31;           // 0.Zeile, 0.Spalte, 2.Seite
myarray3[0, 1, 2] = 32;           // 0.Zeile, 1.Spalte, 2.Seite
myarray3[0, 2, 2] = 33;           // 0.Zeile, 2.Spalte, 2.Seite
myarray3[1, 0, 2] = 34;           // 1.Zeile, 0.Spalte, 2.Seite
myarray3[1, 1, 2] = 35;           // 1.Zeile, 1.Spalte, 2.Seite
myarray3[1, 2, 2] = 36;           // 1.Zeile, 2.Spalte, 2.Seite
myarray3[2, 0, 2] = 37;           // 2.Zeile, 0.Spalte, 2.Seite
myarray3[2, 1, 2] = 38;           // 2.Zeile, 1.Spalte, 2.Seite
myarray3[2, 2, 2] = 39;           // 2.Zeile, 2.Spalte, 2.Seite
#endregion
```


Kopierbar



000 = 11	010 = 12	020 = 13
100 = 14	110 = 15	120 = 16
200 = 17	210 = 18	220 = 19
001 = 21	011 = 22	021 = 23
101 = 24	111 = 25	121 = 26
201 = 27	211 = 28	221 = 29
002 = 31	012 = 32	022 = 33
102 = 34	112 = 35	122 = 36
202 = 37	212 = 38	222 = 39

```

myarray3[0, 0, 0] = 11;
myarray3[0, 1, 0] = 12;
myarray3[0, 2, 0] = 13;
myarray3[1, 0, 0] = 14;
myarray3[1, 1, 0] = 15;
myarray3[1, 2, 0] = 16;
myarray3[2, 0, 0] = 17;
myarray3[2, 1, 0] = 18;
myarray3[2, 2, 0] = 19;

```

```

myarray3[0, 0, 1] = 21;
myarray3[0, 1, 1] = 22;
myarray3[0, 2, 1] = 23;
myarray3[1, 0, 1] = 24;
myarray3[1, 1, 1] = 25;
myarray3[1, 2, 1] = 26;
myarray3[2, 0, 1] = 27;
myarray3[2, 1, 1] = 28;
myarray3[2, 2, 1] = 29;

```

```

myarray3[0, 0, 2] = 31;
myarray3[0, 1, 2] = 32;
myarray3[0, 2, 2] = 33;
myarray3[1, 0, 2] = 34;
myarray3[1, 1, 2] = 35;
myarray3[1, 2, 2] = 36;
myarray3[2, 0, 2] = 37;
myarray3[2, 1, 2] = 38;
myarray3[2, 2, 2] = 39;

```

Ausgabe des Arrays

- Initialisiere ein 3x3x3 Array
- Gib dieses in der Konsole aus

3x3 Matrix		
11	12	13
14	15	16
17	18	19
21	22	23
24	25	26
27	28	29
31	32	33
34	35	36
37	38	39

```
for (int i = 0; i < pages; i++) {           // 3.) zum Schluß die Seiten
    for (int j = 0; j < lines; j++) {       // 2.) dann die Zeile
        for (int k = 0; k < columns; k++) { // 1.) Zuerst die Spalten
            Console.Write($" {j}{k}{i} = {myarray3[j, k, i]} ");
        }
        Console.WriteLine("\n");          // Beginn einer neuen Zeile
    }
    Console.WriteLine("\n");              // Beginn einer neuen Seite
}
```

000 = 11	010 = 12	020 = 13
100 = 14	110 = 15	120 = 16
200 = 17	210 = 18	220 = 19
001 = 21	011 = 22	021 = 23
101 = 24	111 = 25	121 = 26
201 = 27	211 = 28	221 = 29
002 = 31	012 = 32	022 = 33
102 = 34	112 = 35	122 = 36
202 = 37	212 = 38	222 = 39

Dreidimensionales Array

- Erstelle folgende 2 Arrays, gib diese in der Konsole mit korrekten Zeilenumbrüchen aus:

```
/* Das folgende Beispiel zeigt ein Array mit 3 Dimensionen:
 * Die 1. Dimension besteht aus 2 Elementen
 * Die 2. Dimension besteht aus 3 Elementen
 * Die 3. Dimension besteht aus 4 Elementen
 */
int[, ,] myarray2 = new int[2, 3, 4] // Ein Array mit 3 Dimensionen
{
    { {111,112,113,114},{121,122,123,124},{131,132,133,134} },
    { {211,212,213,214},{221,222,223,224},{231,232,233,234} }
};
```

```
/* Das folgende Beispiel zeigt ein Array mit 3 Dimensionen:
 * Die 1. Dimension besteht aus 2 Elementen
 * Die 2. Dimension besteht aus 3 Elementen
 * Die 3. Dimension besteht aus 1 Element
 */
int[, ,] myarray4 = new int[2, 3, 1] // Ein Array mit 3 Dimensionen
{
    { {111},{121},{131} },
    { {211},{221},{231} }
};
```

Ausgabe des Arrays

```

Console.WriteLine("Dreidimensionale Matrix mit 2x3x4");
const int lines = 2;      //Zeilen
const int columns = 3;    //Spalten
const int pages = 4;      //Ebenen
int[, ,] arr = new int[lines, columns, pages]
{
    { {111,112,113,114},{121,122,123,124},{131,132,133,134} },
    { {211,212,213,214},{221,222,223,224},{231,232,233,234} } };

//Für jede Ebene
for (int i = 0; i < arr.GetLength(2); i++) {           //Ebene
    //Zeilenweise die Spalten durchlaufen
    for (int j = 0; j < arr.GetLength(0); j++) {       //Zeile
        for (int k = 0; k < arr.GetLength(1); k++) {   //Spalte
            //Console.Write(arr[j,k, i] + "\t");
            Console.Write($" {j}{k}{i} = {arr[j, k, i]}");
        }
        Console.WriteLine("");
    }
    Console.WriteLine("### ");
}
Console.WriteLine();

```

```

111      121      131
211      221      231
###
112      122      132
212      222      232
###
113      123      133
213      223      233
###
114      124      134
214      224      234
###

```

```

Dreidimensionale Matrix mit 2x3x4
000 = 111   010 = 121   020 = 131
100 = 211   110 = 221   120 = 231
###
001 = 112   011 = 122   021 = 132
101 = 212   111 = 222   121 = 232
###
002 = 113   012 = 123   022 = 133
102 = 213   112 = 223   122 = 233
###
003 = 114   013 = 124   023 = 134
103 = 214   113 = 224   123 = 234
###

```

Ausgabe des Arrays

```

Console.WriteLine("Dreidimensionale Matrix mit 2x3x1");
const int lines = 2;      //Zeilen
const int columns = 3;    //Spalten
const int pages = 1;      //Ebenen
int[, ,] arr = new int[lines, columns, pages]
{
    { {111},{121},{131} }, { {211},{221},{231} }
    // 0 0 0 0 0 0
    //-----
    // 0 1 2 0 1 2
    //-----
    // 0 1
};
//Für jede Ebene
for (int i = 0; i < arr.GetLength(2); i++) { //E
    //Zeilenweise die Spalten durchlaufen
    for (int j = 0; j < arr.GetLength(0); j++) { //Z
        for (int k = 0; k < arr.GetLength(1); k++) { //S
            //Console.Write(arr[j, k, i] + "\t");
            Console.Write($" {j}{k}{i} = {arr[j, k, i]} ");
        }
        Console.WriteLine("");
    }
    Console.WriteLine("###");
}
Console.WriteLine();

```

```

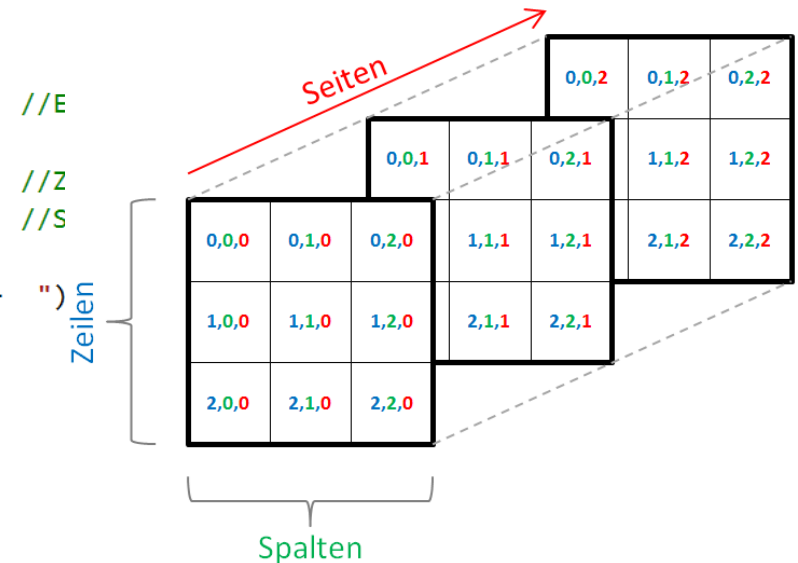
111 121 131
211 221 231
###

```

```

000 = 111 010 = 121 020 = 131
100 = 211 110 = 221 120 = 231
###

```



Übungsbeispiele

Als Hausübung

Beispiel: Eindimensionales Array

- Erstelle ein Eindimensionales Array mit 5 Integerwerten, gib diese in der Console aus.

```
// Single-dimensional array
// int[] numbers = new int[5];
// int[] numbers = new int[] { 1, 2, 3, 4, 5 };
int[] numbers = new int[5] { 1, 2, 3, 4, 5 };

Console.WriteLine("Int-Array eindimensional: ");
foreach(int number in numbers)
    Console.WriteLine( "Nummer lautet: {0}", number);
```

Beispiel: Zweidimensionales Array

- Erstelle ein Zweidimensionales Array mit 6 Vornamen, gib diese in der Console aus.
3 Elemente pro Zeile.

Beispiel Einnahmen 2 Dimensional

- Lese für 4 Wochen mit je 6 Arbeitstagen die Einnahmen des Tages ein.
- Berechne: Minimum, Summe & Durchschnitt
- Gib Minimum und Durchschnitt ebenfalls mit aus

```
Einnahmen 3 Woche, 2 Tag:
44,3
Einnahmen 3 Woche, 3 Tag:
33,3
Einnahmen 3 Woche, 4 Tag:
44,4
Einnahmen 3 Woche, 5 Tag:
33,3
Einnahmen 3 Woche, 6 Tag:
44,4
Einnahmen 4 Woche, 1 Tag:
33,3
Einnahmen 4 Woche, 2 Tag:
22,2
Einnahmen 4 Woche, 3 Tag:
33,3
Einnahmen 4 Woche, 4 Tag:
22,2
Einnahmen 4 Woche, 5 Tag:
33,3
Einnahmen 4 Woche, 6 Tag:
22,2
Summe der Einnahmen: 799,1
```

Lösung Earnings

```
//Einnahmen über 4 Wochen, à 6 Tage. Durchschnitt und Tag mit MIN?  
float sum = 0, average = 0, minimum = 0, day = 0, week = 0;  
float[,] earnings = new float[4, 6]; //4 Zeilen und 6 Spalten
```

```
Console.WriteLine("Summe der Einnahmen: {0}", sum);  
Console.WriteLine("Das Minimum ist: {0} am {1} Tag der {2} Woche", minimum, day, week);  
Console.WriteLine("Durchschnitt der Einnahmen: {0}", average);
```

Aufgabe: Randomfill

- Erstelle ein Programm:
 - Lese vom Benutzer ein, wie groß das Array werden soll
 - Fülle das Array mit Randomwerte an
 - Gib das Array in der Console aus
- *Löse die Aufgabe für ein eindimensionales Array zum Aufwärmen ;)*
- *Löse die Aufgabe für ein zweidimensionales Array*
- *Löse die Aufgabe mit einem dreidimensionalen Array*

- [illegible]

Änderung Kino

- Ändere die Ausgabe auf einen . für freie Sitzplätze und ein rotes x für reservierte Sitzplätze.

```
int lines = 10, columns = 15, cinemahall = 3;
int seatrow = 0, seatcol = 0, hallnumber = 0;

/* Erstmalige initialisierung des Arrays mit 0 */
char[,][,] mycinema = new char[lines, columns, cinemahall];

for (int i = 0; i < cinemahall; i++) {
    for (int j = 0; j < lines; j++) {
        for (int k = 0; k < columns; k++) {
            mycinema[j, k, i] = '.';
        }
    }
}
```

Reservieren Sie Ihren Sitzplatz!

=====

Geben Sie die Saalnummer ein: 1

Geben Sie die Sitzreihe ein: 1

Geben Sie den Sitzplatz ein: 1

x

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

Kino

```
do {
    /* Ausgabe des Arrays */
    for (int i = 0; i < cinemahall; i++) {                // 3.) zum Schluß der Saal
        for (int j = 0; j < lines; j++) {                // 2.) dann die Sitzreihen
            for (int k = 0; k < columns; k++) {           // 1.) Zuerst die Platznummer
                Console.Write(" " + mycinema[j, k, i]);
            }
            Console.WriteLine("\n");                    // Zeilenumbruch bei Beginn einer neuen Sitzreihe
        }
        Console.WriteLine("\n");                        // Zeilenumbruch bei Beginn eines neuen Saales
    }

    Console.WriteLine("Reservieren Sie Ihren Sitzplatz!");
    Console.WriteLine("=====");

    Console.Write("Geben Sie die Saalnummer ein: ");
    hallnumber = Convert.ToInt32(Console.ReadLine())-1;
    Console.Write("Geben Sie die Sitzreihe ein: ");
    seatrow = Convert.ToInt32(Console.ReadLine())-1;
    Console.Write("Geben Sie den Sitzplatz ein: ");
    seatcol = Convert.ToInt32(Console.ReadLine())-1;

    if ((seatrow >= lines) || (seatcol >= columns) || (hallnumber >= cinemahall)) // Prüfe auf fehlerhafte Eingabe
    {
        Console.WriteLine("Fehlerhafte Eingabe!");
        Console.ReadLine();
        Console.Clear();
    }

    else if (mycinema[seatrow, seatcol, hallnumber] == 1) // Sitzplatz bereits vergeben
    {
        Console.WriteLine("\nDieser Sitzplatz ist bereits vergeben! ");
        Console.ReadLine();
        Console.Clear();
    }
    else {
        mycinema[seatrow, seatcol, hallnumber] = 'x';    // Sitzplatz zuweisen
        Console.Clear();
    }
} while (true); // Endlosschleife !
```

Aufgabe: Index als Wert (int/string)

- Erstelle ein 3 dimensionales Array
- Befülle die Zellen mit den Indexpositionen.
- Achte bei der Ausgabe darauf, dass 3 Stellen ausgegeben werden, und führende Nullen angezeigt werden.
 - `Console.Write("{0:d3}\t ", arr[j, k, i]);`
 - `Console.Write("{0:000}\t ", arr[j, k, i]);`

```
Dreidimensionale Matrix mit 2x3x4
000=000 010=010 020=020
100=100 110=110 120=120
###
001=001 011=011 021=021
101=101 111=111 121=121
###
002=002 012=012 022=022
102=102 112=112 122=122
###
003=003 013=013 023=023
103=103 113=113 123=123
###
```

Array & Index

- Gefüllt mit Indexwerten
- Beliebige Größe
- Durch Parameter

```
public static void Dim3MatrixIndex(int lines, int columns, int pages) {
    Console.WriteLine($"Dreidimensionale Matrix mit {lines}x{columns}x{pages}");
    int[, ,] arr = new int[lines, columns, pages];
    for (int i = 0; i < arr.GetLength(2); i++) {           //Ebene
        //Zeilenweise die Spalten durchlaufen
        for (int j = 0; j < arr.GetLength(0); j++) {       //Zeile
            for (int k = 0; k < arr.GetLength(1); k++) {    //Spalte
                arr[j, k, i] = j * 100 + k * 10 + i;
            }
        }
    }

    //Ausgabe
    for (int i = 0; i < arr.GetLength(2); i++) {           //Ebene
        //Zeilenweise die Spalten durchlaufen
        for (int j = 0; j < arr.GetLength(0); j++) {       //Zeile
            for (int k = 0; k < arr.GetLength(1); k++) {    //Spalte
                Console.ForegroundColor = ConsoleColor.Green;
                Console.Write(j);
                Console.ForegroundColor = ConsoleColor.Red;
                Console.Write(k);
                Console.ForegroundColor = ConsoleColor.Blue;
                Console.Write(i);
                Console.ForegroundColor = ConsoleColor.White;
                Console.Write( "={0:000}\t", arr[j, k, i]);
            }
            Console.WriteLine();
        }
        Console.WriteLine("###");
    }
}
```


Index als String speichern

- Ändere das Beispiel
 - auf eine beliebig große Matrix
 - Speichere die Indexpositionen als Zeichenkette im Array
 - Trenne die Indexwerte mit Bindestrichen

```
Dreidimensionale Matrix mit 2x3x4
000=000 010=010 020=020
100=100 110=110 120=120
###
001=001 011=011 021=021
101=101 111=111 121=121
###
002=002 012=012 022=022
102=102 112=112 122=122
###
003=003 013=013 023=023
103=103 113=113 123=123
###
```

```
Dreidimensionale Matrix mit 2x3x4
000=0-0-0 010=0-1-0 020=0-2-0
100=1-0-0 110=1-1-0 120=1-2-0
###
001=0-0-1 011=0-1-1 021=0-2-1
101=1-0-1 111=1-1-1 121=1-2-1
###
002=0-0-2 012=0-1-2 022=0-2-2
102=1-0-2 112=1-1-2 122=1-2-2
###
003=0-0-3 013=0-1-3 023=0-2-3
103=1-0-3 113=1-1-3 123=1-2-3
###
```

Array

Index als String speichern

```
public static void Dim3MatrixIndexString(int lines, int columns, int pages) {
    Console.WriteLine($"Dreidimensionale Matrix mit {lines}x{columns}x{pages}");
    string[, ,] arr = new string[lines, columns, pages];

    for (int i = 0; i < arr.GetLength(2); i++) {           //Ebene
        //Zeilenweise die Spalten durchlaufen
        for (int j = 0; j < arr.GetLength(0); j++) {       //Zeile
            for (int k = 0; k < arr.GetLength(1); k++) {   //Spalte
                arr[j, k, i] = String.Format($"{j}-{k}-{i}");
            }
        }
    }

    //Ausgabe für jede Ebene
    for (int i = 0; i < arr.GetLength(2); i++) {           //Ebene
        //Zeilenweise die Spalten durchlaufen
        for (int j = 0; j < arr.GetLength(0); j++) {       //Zeile
            for (int k = 0; k < arr.GetLength(1); k++) {   //Spalte
                Console.ForegroundColor = ConsoleColor.Green;
                Console.Write(j);
                Console.ForegroundColor = ConsoleColor.Red;
                Console.Write(k);
                Console.ForegroundColor = ConsoleColor.Blue;
                Console.Write(i);
                Console.ForegroundColor = ConsoleColor.White;
                Console.Write("={0}\t ", arr[j, k, i]);
            }
            Console.WriteLine("");
        }
        Console.WriteLine("###");
    }
    Console.WriteLine();
}
```

J	A	G	G	E	D			
A	R	R	A	Y	S			
A	R	E						
C	O	N	F	U	S	I	N	G

Jagged Arrays

Mehrdimensionale Arrays (2)

Variante 2: Blockarrays (rechteckig):

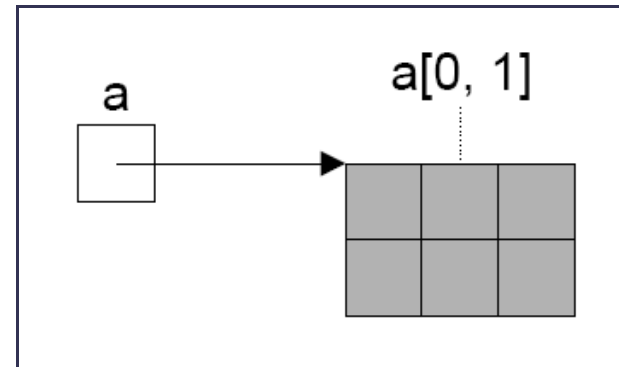
```
int[,] a = new int[2,3];
```

oder

```
int[,] b = {{1,2,3},{4,5,6}};
```

oder 3-dimensional

```
int[,,] c = new int[2,4,2];
```



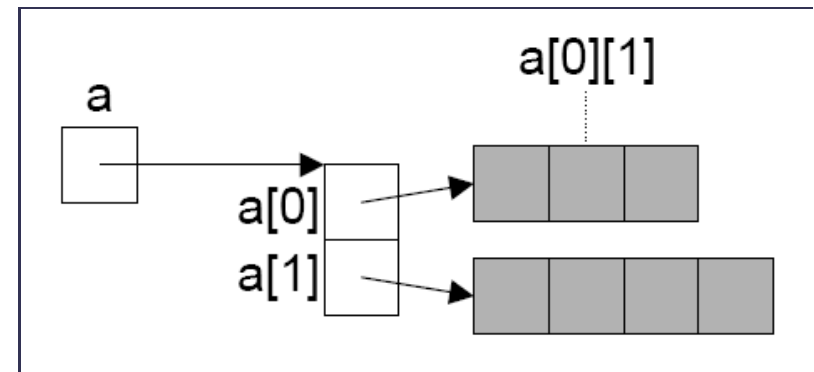
Zugriff auf ein einzelnes Element:

```
int x = a[0,1];
```

Jagged Arrays (1)

Variante 1: „ausgefranst“, jagged:

```
int[][] a = new int[2][];  
a[0] = new int[3];  
a[1] = new int[4];
```



Zugriff auf ein einzelnes Element:

```
int x = a[0][1];
```

J	A	G	G	E	D			
A	R	R	A	Y	S			
A	R	E						
C	O	N	F	U	S	I	N	G

Beispiel: Jagged- Array

- Erstelle ein Jagged-Array aus Byte-Werten mit 5 Zeilen und folgender Ausgabe:

- Length of row 0 is 3
- Length of row 1 is 4
- Length of row 2 is 5
- Length of row 3 is 6
- Length of row 4 is 7

```
int[][] a = new int[2][];  
a[0] = new int[3];  
a[1] = new int[4];
```

Lösung: Jagged- Array

```
// Array-of-arrays (jagged array)
byte[][] scores = new byte[5][];

// Create the jagged array
for (int i = 0; i < scores.Length; i++)
{
    scores[i] = new byte[i + 3];
}
```

```
// Print length of each row
for (int i = 0; i < scores.Length; i++)
{
    Console.WriteLine("Length of row {0} is {1}", i, scores[i].Length);
}
```

```
Length of row 0 is 3
Length of row 1 is 4
Length of row 2 is 5
Length of row 3 is 6
Length of row 4 is 7
0      1      2
10     11     12     13
20     21     22     23     24
30     31     32     33     34     35
40     41     42     43     44     45     46
```

- Speichere Werte in das Jagged Array
- Gib die Werte in der Console aus

Jagged Array

```
//Instantiieren des Arrays
byte[][] scores = new byte[5][];
for (int i = 0; i < scores.Length; i++) {
    scores[i] = new byte[i + 3];
}

//Ausgabe der Spalten pro Zeile
for (int i = 0; i < scores.Length; i++) {
    Console.WriteLine($"Length of row {i} is {scores[i].Length}");
}

//Werte Speichern
for (int i = 0; i < scores.Length; i++) {
    for (int j = 0; j < scores[i].Length; j++) {
        scores[i][j] = Convert.ToByte( i*10 + j);
    }
}

//Werte ausgeben
for (int i = 0; i < scores.Length; i++) {
    for (int j = 0; j < scores[i].Length; j++) {
        Console.Write("{0}\t", scores[i][j]);
    }
    Console.WriteLine();
}
```

```
Length of row 0 is 3
Length of row 1 is 4
Length of row 2 is 5
Length of row 3 is 6
Length of row 4 is 7
0      1      2
10     11     12     13
20     21     22     23     24
30     31     32     33     34     35
40     41     42     43     44     45     46
```


Beispiel Theater - Blocked

Stellen Sie sich die Sitzplätze im Parkett eines Theaters oder Kinos vor (siehe Abbildung 7.2). Dieses soll aus 200 Sitzplätzen bestehen, verteilt auf 10 Reihen zu je 20 Plätzen.

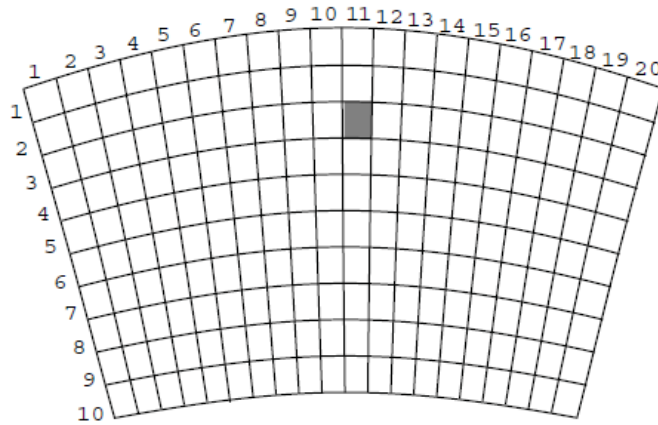


Abbildung 7.2:
Das Parkett als zweidimensionales Array

Ihre Aufgabe soll es nun sein, die Belegung dieser Sitzplätze mit einem Programm zu verwalten. Es bietet sich an, die Plätze in Form eines Arrays zu verwalten:

```
bool sitze = new bool[200];
```

Boolsches Array für die Sitzplätze

```
static void Main(string[] args)
{
    bool[,] sitze = new bool[10, 20];

    sitze[2,10] = true;

    for(int iReihe = 1; iReihe <= sitze.GetLength(0); ++iReihe)
    {
        for(int iPlatz = 1; iPlatz <= sitze.GetLength(1); ++iPlatz)
        {
            if (sitze[iReihe-1,iPlatz-1] == false)
                Console.Write("f ");
            else
                Console.Write("b ");
        }
        Console.WriteLine();
    }
}
```

Beispiel Theater - Jagged

Mehrdimensionale Arrays haben die besondere Eigenschaft, dass die Elemente einer Dimension stets die gleiche Länge haben. Ein zweidimensionales Array besteht beispielsweise aus einer bestimmten Zahl von Zeilen, die alle die gleiche Anzahl Spalten (zweite Dimension) aufweisen.

In der Praxis hat man es aber häufig mit Daten zu tun, die nicht ganz so regelmäßig angeordnet sind. Nehmen wir beispielsweise an, in dem Parkett aus dem vorangehenden Abschnitt gäbe es in der dritten und vierten Reihe nur jeweils 18 Plätze (siehe Abbildung 7.3).

Ein solches Parkett kann nicht durch ein zweidimensionales Array dargestellt werden, wohl aber als ein Array, dessen Elemente selbst auch wieder Arrays sind.

Beispiel Theater - Jagged

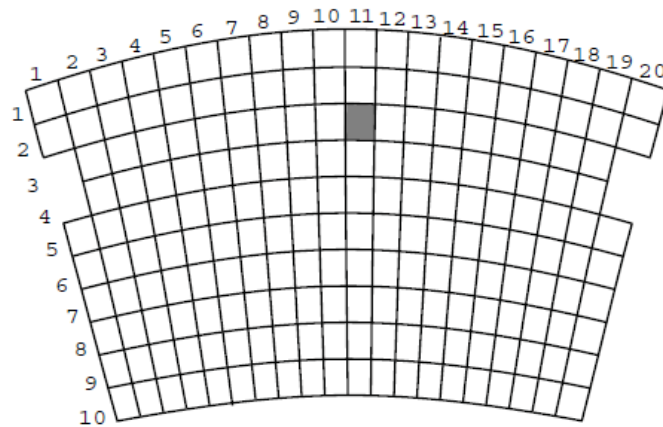


Abbildung 7.3:
Das Parkett als Array von Arrays

Sie haben jetzt ein Array-Objekt mit 10 Elementen, in denen Sie beliebig große Unterarrays mit `bool`-Elementen ablegen können. Der nächste Schritt besteht darin, diese Unterarrays, die die Sitzplätze pro Reihe repräsentieren, zu erzeugen:

```
for(int iReihe = 1; iReihe <= sitze.Length; ++iReihe)
{
    if (iReihe < 3 || iReihe > 4)
        sitze[iReihe-1] = new bool[20];
    else
        sitze[iReihe-1] = new bool[18];
}
```

Aufgabe Theater

- Erstelle ein Programm, welches für ein jagged-Array die Sitzplätze verwaltet.
- Der Benutzer erhält eine Übersicht der freien und besetzten Sitzplätze schön formatiert.
- Der Benutzer soll auswählen können welchen Sitzplatz er reservieren möchte, dann wird Zeile/Spalte eingelesen, wenn dieser noch frei ist wird er als besetzt markiert.
- Das Auswählen soll solange stattfinden bis keine Sitzplätze mehr frei sind.

Lösung mit Char statt Bool

```
int lines = 10, columns = 20;
int seatrow = 0, seatcol = 0;

char[][] theater = new char[lines][];
for (int i = 0; i < theater.Length; i++) {
    if (i < 3 || i > 4) {
        theater[i] = new char[columns];
    }
    else {
        theater[i] = new char[columns-2];
    }
}
// Sitze mit . initialisieren
for (int i = 0; i < theater.GetLength(0); i++) {
    for (int j = 0; j < theater[i].Length; j++) {
        theater[i][j] = '.';
    }
}
```

```
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
Reservieren Sie Ihren Sitzplatz!
=====
Geben Sie die Sitzreihe ein: _
```

```
do {
    /* Ausgabe des Arrays */
    for (int i = 0; i < theater.GetLength(0); i++) {
        if (i == 3 || i == 4)
            Console.WriteLine(" ");
        for (int j = 0; j < theater[i].GetLength(0); j++) {
            Console.Write(theater[i][j]);
        }
        Console.WriteLine("\n");
    }
    Console.WriteLine("\n");

    Console.WriteLine("Reservieren Sie Ihren Sitzplatz!");
    Console.WriteLine("=====");

    Console.Write("Geben Sie die Sitzreihe ein: ");
    seatrow = Convert.ToInt32(Console.ReadLine()) - 1;
    Console.Write("Geben Sie den Sitzplatz ein: ");
    seatcol = Convert.ToInt32(Console.ReadLine()) - 1;

    if ((seatrow >= lines) || (seatcol >= columns) ) // Prüfe auf fehlerhafte Eingabe
    {
        Console.WriteLine("Fehlerhafte Eingabe!");
        Console.ReadLine();
        Console.Clear();
    }

    else if (theater[seatrow][seatcol] == 'x') // Sitzplatz bereits vergeben
    {
        Console.WriteLine("\nDieser Sitzplatz ist bereits vergeben! ");
        Console.ReadLine();
        Console.Clear();
    }
    else {
        theater[seatrow][seatcol] = 'x'; // Sitzplatz zuweisen
        // Console.Clear();
    }
} while (true); // Endlosschleife !
```

Wiederholungsfragen

Arrays

Arrays - Begriffsbestimmungen

3 Schritte:

1. Deklaration

```
int[] my_array;  
// Array von Integerzahlen
```
2. *Instantiierung*

```
my_array = new int[3];  
// Erschafft Array mit 3 Elem.
```
3. Initialisierung

```
my_array[0] = 17;  
my_array[1] = 0;  
my_array[2] = -4*my_array[0];
```


Array mit 3 Integer

Deklarieren und Instantiieren:

```
int[] a = new int[3];
```

Deklaration mit Initialisierung:

```
int[] b = new int[] {3, 4, 5};
```

Deklaration mit *impliziter Erschaffung* und Initialisierung:

```
int[] c = {3, 4, 5};
```

Arrays - Operationen

Die Klasse **System.Array** enthält nützliche Operationen für den Umgang mit Arrays:

```
int[] a = {4, 5, 6};  
int[] b = new int[2];
```

```
//kopiert a[0..1] nach b  
Array.Copy(a, b, 2);
```

```
//Sortieren von Arrays  
Array.Sort(b);
```

```
...
```