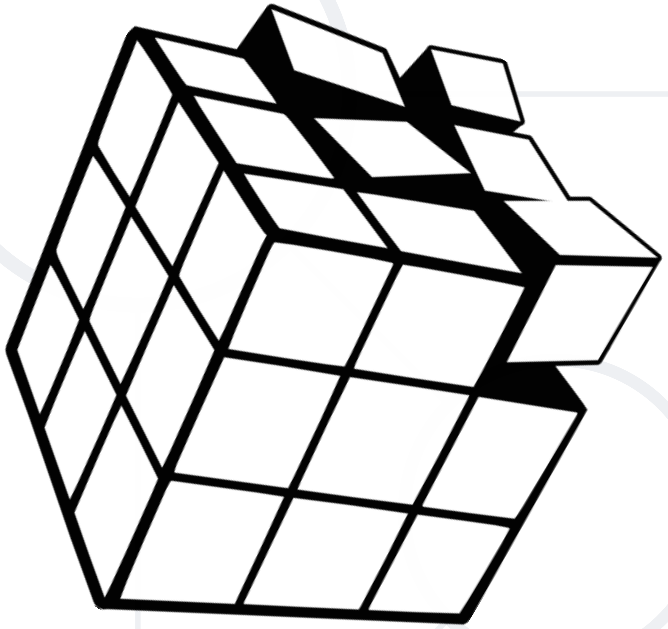
Multidimensional Arrays

Processing Matrices and Jagged Arrays



SoftUni Team

Technical Trainers

Software University

https://about.softuni.bg/

Table of Contents

Multidimensional Arrays

Definition and Usage

What is Multidimensional Array?

- Array is a systematic arrangement of **similar objects**
- **Multidimensional arrays** have more than one dimension
  - The most used multidimensional arrays are the **2-dimensional**

R	COLS				
O	[0][0]	[0][1]	[0][2]	[0][3]	[0][4]
W	[1][0]	[1][1]	[1][2]	[1][3]	[1][4]
S	[2][0]	[2][1]	[2][2]	[2][3]	[2][4]

Row Index

Col Index

Creating Multidimensional Arrays

- Creating a multidimensional array
  - Use the **new** keyword
  - Must specify the size of each dimension

```
int[,] intMatrix = new int[3, 4];
float[,] floatMatrix = new float[8, 2];
string[,,] stringCube = new string[5, 5, 5];
```

  - This syntax is specific only to C#

Initializing Multidimensional Arrays

- Initializing with values:

```
int[,] matrix = {
    {1, 2, 3, 4}, // row 0 values
    {5, 6, 7, 8}  // row 1 values
};
```

- Multidimensional arrays represent a **rows with values**
- The rows represent the first dimension and the columns - the second (**the one inside the first**)

Accessing Elements

- Accessing **N-dimensional array element**:

```
nDimensionalArray[index1, … , indexn]
```

- **Getting** element **value**:

```
int[,] array = {{1, 2}, {3, 4}}
int element11 = array[1, 1]; // element11 = 4
```

- **Setting** element value:

```
int[,] array = new int[3, 4];
for (int row = 0; row < array.GetLength(0); row++)
  for (int col = 0; col < array.GetLength(1); col++)
    array[row, col] = row + col;
```

Returns the length of the dimension

Printing Matrix – Example (1)

```
int[,] matrix =
               {  { 5, 2, 3, 1 },
                  { 1, 9, 2, 4 },
                  { 9, 8, 6, 11 }  };
for (int row = 0; row < matrix.GetLength(0); row++)
{
  for (int col = 0; col < matrix.GetLength(1); col++)
  {
    Console.Write("{0} ", matrix[row, col]);
  }

  Console.WriteLine();
}
```

Printing Matrix – Example (2)

- **Foreach** iterates through all elements in the matrix

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

foreach (int element in matrix)
{
  Console.WriteLine(element);
}
```

Problem: Sum Matrix Elements

- Read a matrix from the console
- Print the number of rows
- Print the number of columns
- Print the **sum of all numbers** in the matrix

```
3, 6
7, 1, 3, 3, 2, 1
1, 3, 9, 8, 5, 6
4, 6, 7, 9, 1, 0
```

$\rightarrow$

```
3
6
76
```

```
3, 4
1, 2, 3, 1
1, 2, 2, 4
2, 2, 2, 2
```

$\rightarrow$

```
3
4
24
```

Solution: Sum Matrix Elements (1)

```
int[] sizes = Console.ReadLine().Split(", ")
   .Select(int.Parse).ToArray();
int[,] matrix = new int[sizes[0], sizes[1]];
for (int row = 0; row < matrix.GetLength(0); row++)
{
  int[] colElements = Console.ReadLine()
       .Split(", ")
       .Select(int.Parse)
       .ToArray();
  for (int col = 0; col < matrix.GetLength(1); col++)
    matrix[row, col] = colElements[col];
}
```

Gets length of 0th dimension (rows)

Gets length of 1st dimension (cols)

Solution: Sum Matrix Elements (2)

```
int sum = 0;
for (int row = 0; row < matrix.GetLength(0); row++)
{
  for (int col = 0; col < matrix.GetLength(1); col++)
    sum += matrix[row, col];
}
Console.WriteLine(matrix.GetLength(0));
Console.WriteLine(matrix.GetLength(1));
Console.WriteLine(sum);
```

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/3174#11

Problem: Sum Matrix Columns

- Read matrix sizes
- Read a matrix from the console
- Print the **sum of all numbers** in matrix columns

```
3, 6
7 1 3 3 2 1
1 3 9 8 5 6
4 6 7 9 1 0
```

```
12
10
19
20
8
7
```

```
3, 3
1 2 3
4 5 6
7 8 9
```

```
12
15
18
```

Solution: Sum Matrix Columns (1)

```
var sizes = Console.ReadLine().Split(", ")
   .Select(int.Parse).ToArray();
int[,] matrix = new int[sizes[0], sizes[1]];
for (int r = 0; r < matrix.GetLength(0); r++)
{
  var col = Console.ReadLine().Split()
     .Select(int.Parse).ToArray();
  for (int c = 0; c < matrix.GetLength(1); c++)
  {
    matrix[r, c] = col[c];
  }
}
```

Solution: Sum Matrix Columns (2)

```
for (int c = 0; c < matrix.GetLength(1); c++)
{
  int sum = 0;
  for (int r = 0; r < matrix.GetLength(0); r++)
  {
    sum += matrix[r, c];
  }
  Console.WriteLine(sum);
}
```

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/3174#12

Problem: Square with Maximum Sum

- Find **2x2 square** with max sum in given matrix
  - Read matrix from the console
  - Find **biggest sum** of 2x2 submatrix
  - Print the result like a new matrix

```
int[,] matrix = {
  {7, 1, 3, 3, 2, 1},
  {1, 3, 9, 8, 5, 6},
  {4, 6, 7, 9, 1, 0}
};
```

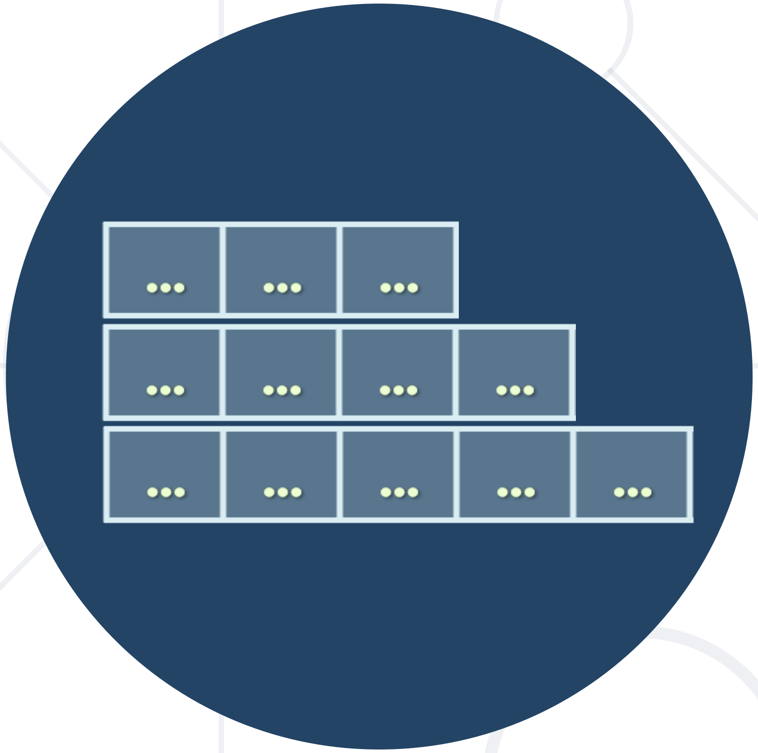
$\rightarrow$

```
9 8
7 9
33
```

Solution: Square with Maximum Sum

```
// TODO: Read the input from the console
for (int row = 0; row < matrix.GetLength(0) - 1; row++) {
  for (int col = 0; col < matrix.GetLength(1) - 1; col++) {
    var newSquareSum = matrix[row, col] +
                       matrix[row + 1, col] +
                       matrix[row, col + 1] +
                       matrix[row + 1, col + 1];
    // TODO: Check if the sum is bigger
  }
}
// TODO: Print the square with the max sum
```

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/3174#15



Jagged Arrays

Definition and Usage

What is Jagged Array

- **Jagged arrays** are multidimensional arrays
  - But each dimension has different size
  - A jagged array is an **array of arrays**
  - Each of the arrays has **different length**

```
int[][] jagged = new int[3][];
jagged[0] = new int[3];
jagged[1] = new int[2];
```

- **Accessing element**

Col Index

```
int element = jagged[0][0];
```

Row Index

Filling a Jagged Array

```
int[][] jagged = new int[5][];

for (int row = 0; row < jagged.Length; row++)
{
  string[] inputNumbers = Console.ReadLine().Split(' ');
  jagged[row] = new int[inputNumbers.Length];

  for (int col = 0; col < jagged[row].Lenght; col++)
  {
    jagged[row][col] = int.Parse(inputNumbers[col]);
  }
}
```

Printing a Jagged Array – Example

- **For loop**

Implement custom method

```
int[][] matrix = ReadMatrix();
for (int row = 0; row < matrix.Length; row++)
  for (int col = 0; col < matrix[row].Length; col++)
    Console.Write("{0} ", matrix[row][col]);
Console.WriteLine();
```

- **Foreach loop**

```
int[][] matrix = ReadMatrix();
foreach (int[] row in matrix)
{
  Console.WriteLine(string.Join(" ", row));
}
```

Problem: Jagged-Array Modification

- On the first line you will get count of rows: **n**
- Next **n** lines hold the elements for each row
- Until you receive "**END**", read commands
  - Add `{row} {col} {value}`
  - Subtract `{row} {col} {value}`
- If the coordinates are invalid print "`Invalid coordinates`"
- When you receive "**END**" you should print the jagged array

```
3
1 2 3
4 5 6
7 8 9
Add 0 0 5
Subtract 1 1 2
END
```

Solution: Jagged-Array Modification (1)

```
int rowSize = int.Parse(Console.ReadLine());
int[][] matrix = new int[rowSize][];

for (int r = 0; r < rowSize; r++)
{
  int[] col = Console.ReadLine()
                     .Split()
                     .Select(int.Parse)
                     .ToArray();
  matrix[r] = col;
}
// continues on the next slide
```

Solution: Jagged-Array Modification (2)

```
string line;
while ((line = Console.ReadLine()) != "END") {
  string[] tokens = line.Split();
  string command = tokens[0];
  int row = int.Parse(tokens[1]);
  int col = int.Parse(tokens[2]);
  int value = int.Parse(tokens[3]);
  if (row < 0 || row >= matrix.Length || … )
    { Console.WriteLine("Invalid coordinates"); }
  else
    { // TODO: Execute the command }
}
// TODO: Print the matrix
```

Check the col

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/3174#16

Problem: Pascal Triangle

- Write a program, which prints on the console a Pascal Triangle

```
6
```

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
```

```
4
```

```
1
1 1
1 2 1
1 3 3 1
```

```
2
```

```
1
1 1
```

Solution: Pascal Triangle (1)

```
int height = int.Parse(Console.ReadLine());
long[][] triangle = new long[height][];
int currentWidth = 1;
for (long row = 0; row < height; row++)
{
  triangle[row] = new long[currentWidth];
  long[] currentRow = triangle[row];
  currentRow[0] = 1;
  currentRow[currentRow.Length - 1] = 1;
  currentWidth++;
  // TODO: Fill elements for each row (next slide)
}
```

Solution: Pascal Triangle (2)

```
if (currentRow.Length > 2)
{
  for (int i = 1; i < currentRow.Length - 1; i++)
  {
    long[] previousRow = triangle[row - 1];
    long prevoiousRowSum = previousRow[i] + previousRow[i - 1];
    currentRow[i] = prevoiousRowSum;
  }
}
// TODO: Print triangle
foreach (long[] row in triangle)
  Console.WriteLine(string.Join(" ", row));
```

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/3174#17

Summary

- **Multidimensional arrays**
  - Have **more than one** dimension
  - Two-dimensional arrays are like tables with **rows** and **columns**
- **Jagged arrays**
  - Arrays of arrays
  - Each **element** is an array **itself**

Software University

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

License