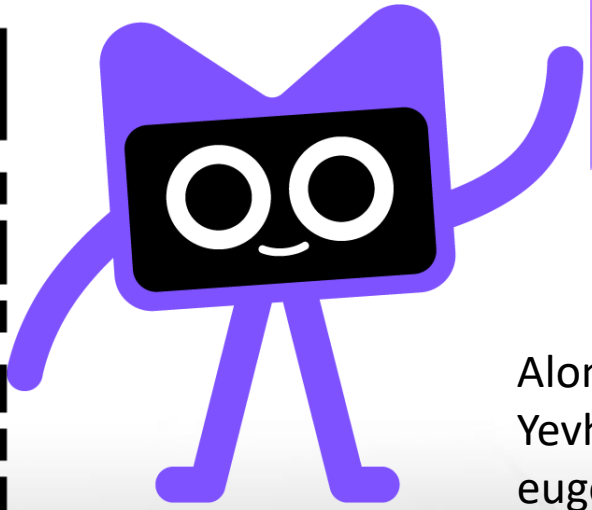
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Algorithms & Programming

Kotlin language

(p.8 – Two-dimensional arrays)

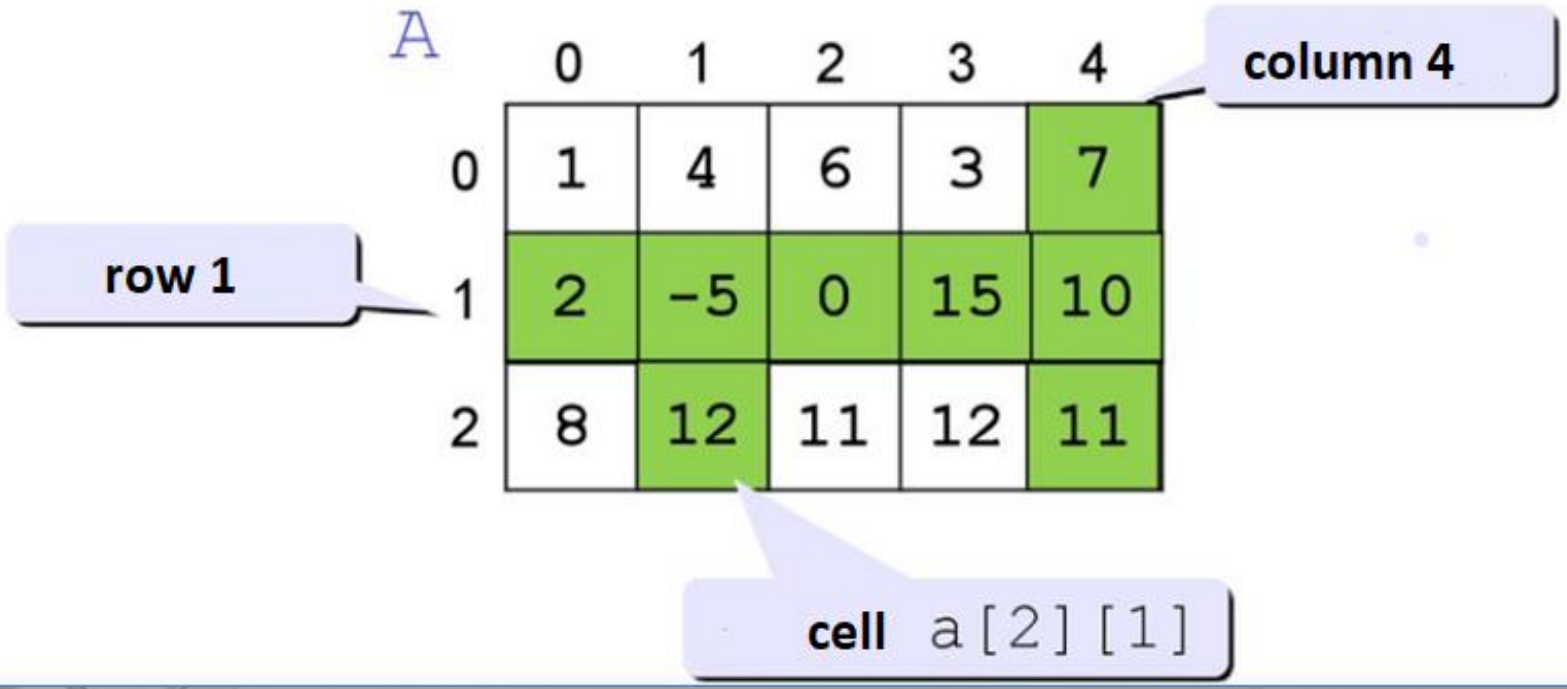


Alona Pavlenko,
Yevhen Berkunskyi, NUoS
eugeny.berkunsky@gmail.com
http://www.berkut.mk.ua

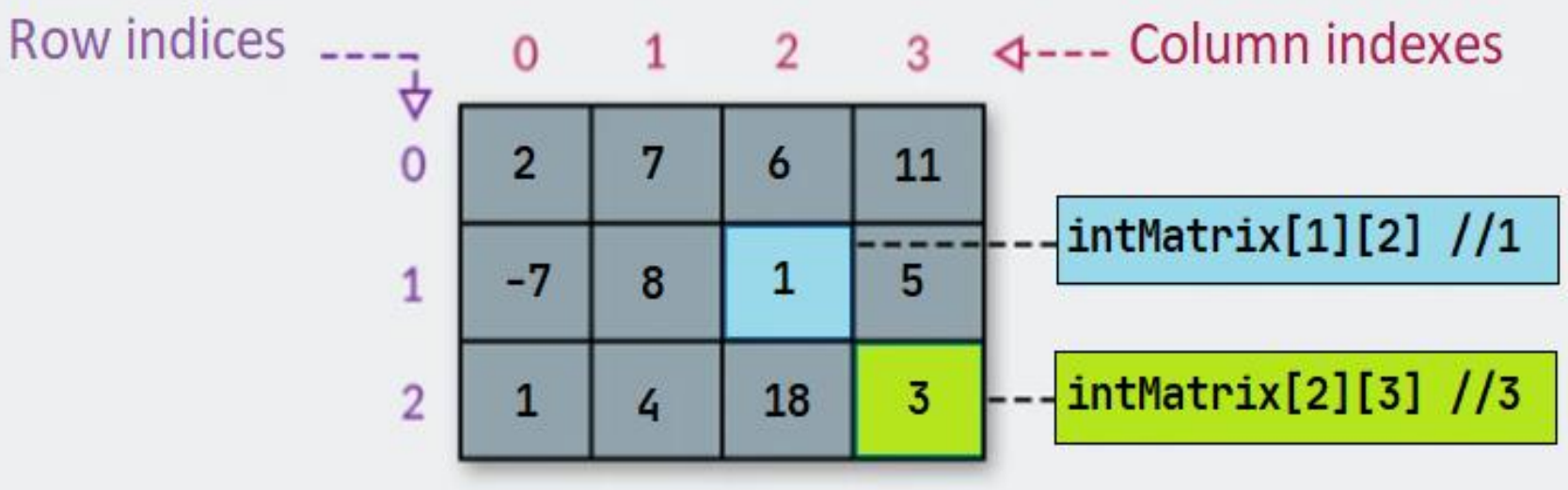
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Matrix

- A matrix is a rectangular table of elements of the same type.
- A matrix is an array where each element has two indices (row number and column number)



Two-dimensional arrays



In total there are 3 rows and 4 columns in the array, that is, 3 x 4 = **12 elements**

```
val intMatrix = Array(3) { IntArray(4) }
```

Declaring a two-dimensional array

```
val intMatrix = Array(3) { IntArray(4) }  // filled with 0
```

- We declared a two-dimensional array consisting of 3 rows, 4 elements in each row
- It is sometimes said that such a two-dimensional array consists of 3 rows and 4 columns

Initializing a two-dimensional array

- For a one-dimensional array, we could write like this:

```
val arr1 = IntArray(10) { 1 }
val arr2 = IntArray(10) { it+1 }
val arr3 = IntArray(10) { readln().toInt() }
```

But what about a two-dimensional array? What is the difficulty?

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Initializing a two-dimensional array

- For a one-dimensional array, we could write like this:

```
val arr1 = IntArray(10) { 1 }
val arr2 = IntArray(10) { it+1 }
val arr3 = IntArray(10) { readln().toInt() }
```

But what about a two-dimensional array? What is the difficulty?

```
val a = Array(3) { IntArray(4) { 1 } } // filled with 1
```

Initializing a two-dimensional array

- Fill an array with «multiplication table»

```
val a = Array(10) {
        row -> IntArray(10) {
                column -> (row + 1) * (column + 1)
        }
}
```

<u>Note:</u> It may be written in one line.

Note: Creating a two-dimensional array of 10 rows. Each row represents a one-dimensional array of 10 integers. Depending on the position in the array, the values of the elements are determined according to the rule.

```
(row + 1) * (column + 1)
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Initializing a two-dimensional array

- In Kotlin, you can create a two-dimensional array using the **Array** class or the **arrayOf** function.
- Here's an example:

```
// Using Array class
val matrix = Array(3) { IntArray(3) }

// Using arrayOf function
val matrix = arrayOf(
    intArrayOf(1, 2, 3),
    intArrayOf(4, 5, 6),
    intArrayOf(7, 8, 9)
)
```

Initializing a two-dimensional array

```
// Using Array class
val matrix = Array(3) { IntArray(3) }

// Using arrayOf function
val matrix = arrayOf(
    intArrayOf(1, 2, 3),
    intArrayOf(4, 5, 6),
    intArrayOf(7, 8, 9)
)

val matrix = Array(3) { row ->
       IntArray(3) { col -> row * 3 + col + 1 }
}
```

In examples, we create a 3x3 matrix. The **matrix** variable is initialized with a two-dimensional array of integers.
Each row is represented by an **IntArray**, and the outer array represents the rows.

Initializing a two-dimensional array

To access or modify elements in the matrix, you can use the row and column indices:

```
// Accessing element at row 1, column 2 (value = 6)
val value = matrix[1][2]

// Modifying element at row 0, column 1
matrix[0][1] = 10
```

You can iterate over the elements of a two-dimensional array using nested loops:

```
for (row in matrix) {
    for (element in row) {
        println(element)
    }
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Input from the keyboard

1. Classic way

```
val (rows, columns) = readln().split(" ").map{ it.toInt() }
val intMatrix = Array(rows) { IntArray(columns) }
for (i in intMatrix.indices) {
    for (j in intMatrix[i].indices) {
        intMatrix[i][j] = readln().toInt()
    }
}
```

First, we read the dimensions,
then all the elements of the array, one per line

Input from the keyboard

2. Using java.util.Scanner

```
val input = Scanner(System.`in`)
val rows = input.nextInt()
val columns = input.nextInt()
val intMatrix = Array(rows) { IntArray(columns) }
for (i in intMatrix.indices) {
    for (j in intMatrix[i].indices) {
        intMatrix[i][j] = input.nextInt()
    }
}
```

First, we read the dimensions,
then all the elements of the array

Input from the keyboard

3. Row by row

```
val intMatrix = Array(3) {
    readln().split(" ").map{ it.toInt() }.toIntArray()
}
```

First, we read the dimensions,
then all the elements of the array, one row per line

Input from the keyboard

4. Using java.util.Scanner (all elements at once)

```
val input = Scanner(System.`in`)
val rows = input.nextInt()
val columns = input.nextInt()
val intMatrix = Array(rows) {
        IntArray(columns) { input.nextInt() }
}
```

Input from the keyboard

5. Row by row (different number of elements)

```
val intMatrix = Array(3) {
    readln().split(" ").map{ it.toInt() }.toIntArray()
}
```

What happens if you enter a different number of elements in the lines?

Output on screen

1. Classic way

```
for (i in intMatrix.indices) {
    for (j in intMatrix[i].indices) {
        print("%3d".format(intMatrix[i][j]))
    }
    println()
}
```

Output on screen

2. Enhanced way

```
for (row in intMatrix) {
    for (x in row) {
        print("%3d".format(x))
    }
    println()
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Output on screen

3. «Fashion» way

```
intMatrix.forEach {
    it.forEach { print("%3d".format(it)) }
    println()
}
```

2D arrays problems

- Sum of all elements

```
fun sum(a: Array<IntArray>) : Int {
    var result = 0
    for (row in a) {
        result += row.sum()
    }
    return result
}
```

2D arrays problems

- Max of elements

```
fun maxElement(a: Array<IntArray>) : Int {
    var result = a[0][0]
    for (row in a) {
        result = max(result, row.max())
    }
    return result
}
```

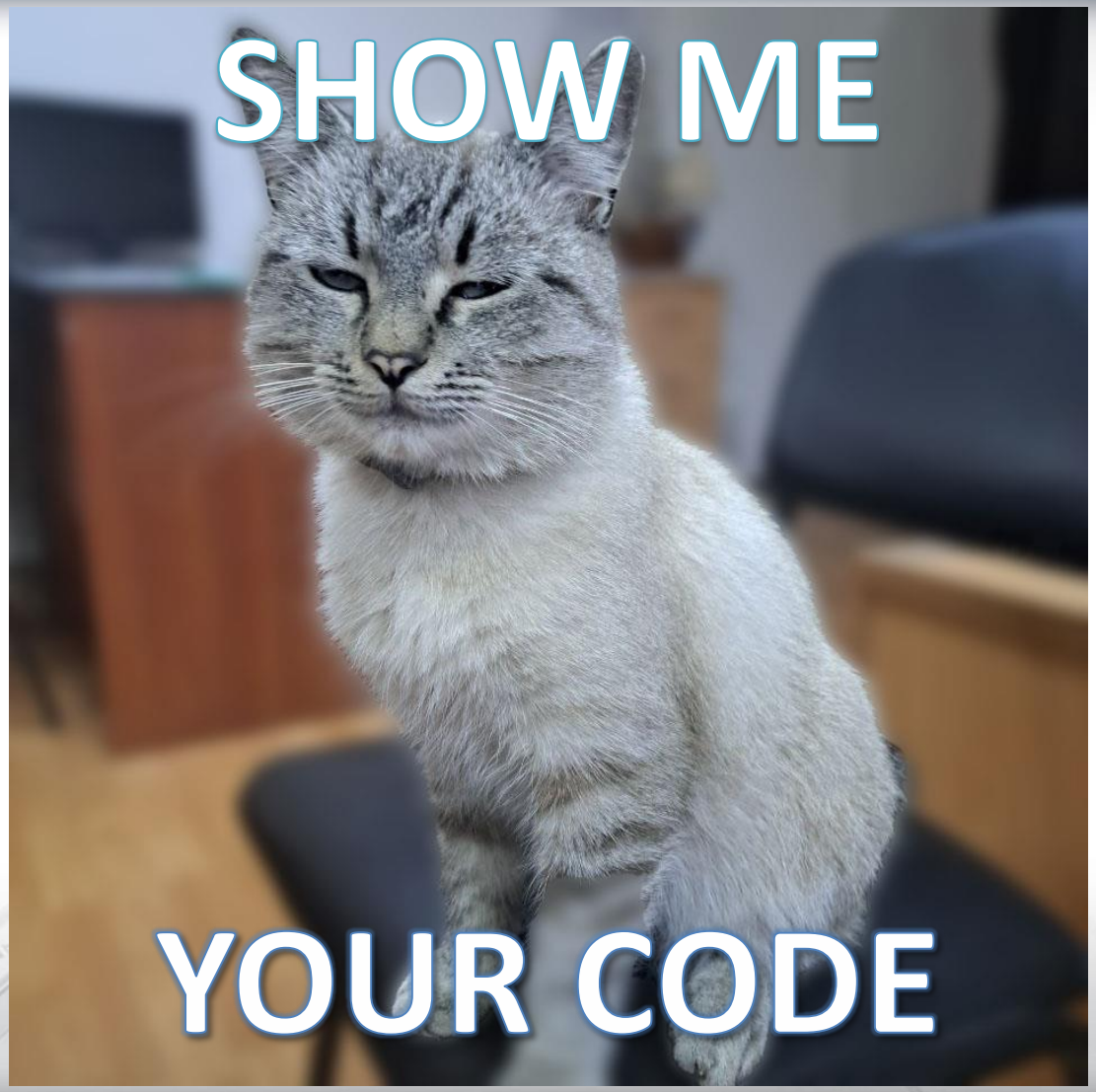
2D arrays problems

- Indices of max element

```
fun indicesOfMax(a: Array<IntArray>): Pair<Int, Int> {
    var result = Pair(0,0)
    var max = a[0][0]
    for (i in a.indices) {
        for (j in a[i].indices) {
            if (a[i][j] > max) {
                max = a[i][j]
                result = Pair(i,j)
            }
        }
    }
    return result
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Let’s code!



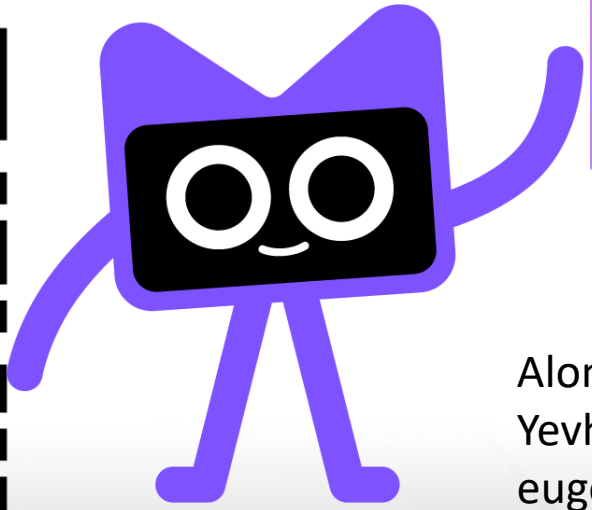
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Algorithms & Programming

Kotlin language

(p.8 – Two-dimensional arrays)



Alona Pavlenko,
Yevhen Berkunskyi, NUoS
eugeny.berkunsky@gmail.com
http://www.berkut.mk.ua