Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Отчёт

по лабораторной работе №2

на тему:

**ЦИКЛЫ, ДИАПАЗОНЫ И МАССИВЫ**

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ф.В. Усенко

(подпись)

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Косович П.В.

(подпись) гр. 214302

Минск, 2024

**Цель.** Закрепить понимание работы циклов, диапазонов и массивов в Kotlin. Научиться эффективно использовать эти конструкции для решения различных задач.

**Задание.** **Генерация и анализ случайных матриц**: Создайте программу, которая генерирует несколько случайных матриц заданного размера, а затем выполняет над ними арифметические операции (сложение, умножение) и анализирует результат (например, на симметричность, наличие нулевых строк и столбцов).

**Код**

import kotlin.random.Random  
  
fun generateMatrix(rows: Int, cols: Int): Array<IntArray> {  
 return Array(rows) **{** IntArray(cols) **{** Random.nextInt(0, 20) **} }**}  
  
fun addMatrices(matrix1: Array<IntArray>, matrix2: Array<IntArray>): Array<IntArray> {  
 val rows = matrix1.size  
 val cols = matrix1[0].size  
 val result = Array(rows) **{** IntArray(cols) **}** for (i in 0 *until* rows) {  
 for (j in 0 *until* cols) {  
 result[i][j] = matrix1[i][j] + matrix2[i][j]  
 }  
 }  
 return result  
}  
  
fun addAllMatrices(matrices: Array<Array<IntArray>>): Array<IntArray> {  
 val rows = matrices[0].size  
 val cols = matrices[0][0].size  
 var result = Array(rows) **{** IntArray(cols) **}** for (i in matrices.*indices*) {  
 result = *addMatrices*(result, matrices[i])  
 }  
 return result  
}  
  
fun multiplyMatrices(matrix1: Array<IntArray>, matrix2: Array<IntArray>): Array<IntArray>? {  
 val rows1 = matrix1.size  
 val cols1 = matrix1[0].size  
 val rows2 = matrix2.size  
 val cols2 = matrix2[0].size  
  
 if (cols1 != rows2) {  
 *println*("Невозможно умножить матрицы: несоответствие размеров.")  
 return null  
 }  
  
 val result = Array(rows1) **{** IntArray(cols2) **}** for (i in 0 *until* rows1) {  
 for (j in 0 *until* cols2) {  
 for (k in 0 *until* cols1) {  
 result[i][j] += matrix1[i][k] \* matrix2[k][j]  
 }  
 }  
 }  
 return result  
}  
  
fun multiplyAllMatrices(matrices: Array<Array<IntArray>>): Array<IntArray>? {  
 var result = matrices[0]  
  
 for (i in 1 *until* matrices.size) {  
 result = *multiplyMatrices*(result, matrices[i]) ?: return null  
 }  
  
 return result  
}  
  
fun isSymmetric(matrix: Array<IntArray>): Boolean {  
 if (matrix.size != matrix[0].size) return false  
  
 for (i in matrix.*indices*) {  
 for (j in i + 1 *until* matrix.size) {  
 if (matrix[i][j] != matrix[j][i]) {  
 return false  
 }  
 }  
 }  
 return true  
}  
  
fun hasZeroRow(matrix: Array<IntArray>): Boolean {  
 return matrix.*any* **{** row **->** row.*all* **{ it** == 0 **} }**}  
  
fun hasZeroColumn(matrix: Array<IntArray>): Boolean {  
 for (j in matrix[0].*indices*) {  
 if (matrix.*all* **{ it**[j] == 0 **}**) {  
 return true  
 }  
 }  
 return false  
}  
  
fun printMatrix(matrix: Array<IntArray>) {  
 for (row in matrix) {  
 *println*(row.*joinToString*(" "))  
 }  
 *println*()  
}  
  
fun readPositiveInt(prompt: String): Int {  
 while (true) {  
 *println*(prompt)  
 val input = *readlnOrNull*()  
 try {  
 val number = input?.*toInt*()  
 if (number != null && number > 0) {  
 return number  
 } else {  
 *println*("Пожалуйста, введите положительное целое число.")  
 }  
 } catch (e: NumberFormatException) {  
 *println*("Ошибка ввода. Пожалуйста, введите корректное число.")  
 }  
 }  
}  
  
fun main() {  
 val numberOfMatrixes = *readPositiveInt*("Введите количество матриц: ")  
 val rows = *readPositiveInt*("Введите количество строк: ")  
 val cols = *readPositiveInt*("Введите количество столбцов: ")  
 val matrixes = Array(numberOfMatrixes) **{** Array(rows) **{** IntArray(cols) **} }** for (i in 0..numberOfMatrixes - 1) {  
 matrixes[i] = *generateMatrix*(rows, cols)  
 val num = i + 1  
 *println*("Матрица $num")  
 *printMatrix*(matrixes[i])  
 }  
 *println*("Результат сложения матриц: ")  
 *printMatrix*(*addAllMatrices*(matrixes))  
 val resultMatrix = *multiplyAllMatrices*(matrixes)  
 if (resultMatrix != null) {  
 *println*("\nРезультат умножения всех матриц:")  
 *printMatrix*(resultMatrix)  
 } else {  
 *println*("\nУмножение матриц невозможно.")  
 }  
 for (i in 0..numberOfMatrixes - 1) {  
 val num = i + 1  
 *println*("Матрица $num симметрична: ${*isSymmetric*(matrixes[i])}")  
 *println*("Матрица $num содержит нулевую строку: ${*hasZeroRow*(matrixes[i])}")  
 *println*("Матрица $num содержит нулевой столбец: ${*hasZeroColumn*(matrixes[i])}")  
 }  
}

**Контрольные вопросы**

1. Чем отличается цикл while от цикла for?

While выполняется пока истинно условие, а for заданное количество раз

1. Какую роль выполняет цикл do...while, и в чем его отличие от while?

do...while выполняется в целом как и while, то бишь пока истинно условие, за исключением того, что один раз он точно выполнится, даже если условие ложно.

11. Как обратиться к элементу массива по индексу? Что будет, если индекс выйдет за пределы массива?

Пусть array – массив.

array[i] – обращение по индексу

Если индекс выйдет за пределы массива, будет ошибка.

16. Как удалить элемент из массива, если известно его значение?

Для удаления элемента из массива можно использовать функцию removeItem().

removeItem(array, value), где array – массив, а value – значение удаляемого элемента.

**Вывод**: Были изучены циклы и диапазоны. Написана программа, которая выделяет подмассивы из массива.