Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Дисциплина: Программирование мобильных информационных систем

ОТЧЁТ по лабораторной работе №3 на тему:

**«Функциональное программирование и лямбда-выражения»**

Выполнил: Проверил:

ст. гр. 214301 Усенко Ф.В.

Кузнецов Е.В.

Минск 2024

**Цель работы:** ознакомиться с основами функционального программирования в Kotlin, изучить лямбда-выражения, анонимные функции и замыкания. Научиться использовать эти концепции для написания более гибкого и читаемого кода.

**Вариант 13. Лямбда-выражения для работы с многомерными данными:** Напишите программу, которая использует лямбда-выражения для анализа и обработки многомерных данных, таких как матрицы или тензоры. Программа должна поддерживать сложные операции, такие как транспонирование, слияние и нормализация.

# КОД ПРОГРАММЫ

typealias Matrix = List<List<Double>>  
  
fun main() {  
 val transpose: (Matrix) -> Matrix = **{** matrix **->** val rows = matrix.size  
 val cols = matrix[0].size  
 *List*(cols) **{** col **->** *List*(rows) **{** row **->** matrix[row][col] **} }  
 }** val merge: (Matrix, Matrix) -> Matrix = **{** matrix1, matrix2 **->** if (matrix1.size != matrix2.size) throw IllegalArgumentException("Кол-во строк матрицы должно совпадать.")  
 matrix1.*zip*(matrix2) **{** row1, row2 **->** row1 + row2 **}  
 }** val normalize: (Matrix) -> Matrix = **{** matrix **->** val flat = matrix.*flatten*()  
 val max = flat.*max*()  
 val min = flat.*min*()  
 if (max == min) matrix  
 else matrix.*map* **{** row **->** row.*map* **{** (**it** - min) / (max - min) **} }  
 }** val matrix1 = *listOf*(  
 *listOf*(1.0, 2.0, 3.0),  
 *listOf*(4.0, 5.0, 6.0),  
 *listOf*(7.0, 8.0, 9.0)  
 )  
  
 val matrix2 = *listOf*(  
 *listOf*(10.0, 11.0),  
 *listOf*(12.0, 13.0),  
 *listOf*(14.0, 15.0)  
 )  
  
 while (true) {  
 *println*(  
 """  
 Выберите действие:  
 1. Транспонирование матрицы  
 2. Слияние двух матриц  
 3. Нормализация матрицы  
 4. Выход  
 """.*trimIndent*()  
 )  
  
 *print*("Введите номер действия: ")  
 when (*readLine*()?.*toIntOrNull*()) {  
 1 -> {  
 *println*("Оригинальная матрица:")  
 matrix1.*forEach* **{** *println*(**it**) **}** val transposed = transpose(matrix1)  
 *println*("\nТранспонированная матрица:")  
 transposed.*forEach* **{** *println*(**it**) **}** }  
  
 2 -> {  
 *println*("Матрица 1:")  
 matrix1.*forEach* **{** *println*(**it**) **}** *println*("\nМатрица 2:")  
 matrix2.*forEach* **{** *println*(**it**) **}** try {  
 val merged = merge(matrix1, matrix2)  
 *println*("\nСлитая матрица:")  
 merged.*forEach* **{** *println*(**it**) **}** } catch (e: IllegalArgumentException) {  
 *println*("Ошибка: ${e.message}")  
 }  
 }  
  
 3 -> {  
 *println*("Оригинальная матрица:")  
 matrix1.*forEach* **{** *println*(**it**) **}** val normalized = normalize(matrix1)  
 *println*("\nНормализованная матрица:")  
 normalized.*forEach* **{** *println*(**it**) **}** }  
  
 4 -> {  
 *println*("Выход из программы.")  
 System.exit(0)  
 }  
  
 else -> *println*("Некорректный ввод. Попробуйте снова.")  
 }  
 *println*()  
 }  
}

# ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как объявить функцию в Kotlin? В чем разница между обычной функцией и однострочной функцией?

fun greet(name: String) = "Hello, $name!"

Разница:

Однострочная функция сокращает синтаксис за счёт автоматического определения возвращаемого типа (если он может быть выведен).

В обычной функции требуется явно использовать return, если есть тело блока { }.

1. Что такое функция высшего порядка, и как её использовать? Приведите пример.

Функция высшего порядка принимает другую функцию в качестве аргумента или возвращает функцию как результат.

**Пример:**

fun operateOnNumbers(a: Int, b: Int, operation: (Int, Int) -> Int): Int {

return operation(a, b)

}

fun add(x: Int, y: Int) = x + y

fun main() {

val result = operateOnNumbers(5, 3, ::add)

println(result) // 8

}

1. Как передать функцию в качестве параметра другой функции? Приведите пример.

Можно передать функцию в качестве параметра, используя ссылку на неё (::) или лямбду.

**Пример:**

fun multiply(a: Int, b: Int) = a \* b

fun calculate(a: Int, b: Int, operation: (Int, Int) -> Int): Int {

return operation(a, b)

}

fun main() {

val result = calculate(4, 6, ::multiply) // Передаём функцию multiply

println(result) // 24

}

1. Чем отличается лямбда-выражение от обычной функции?

**Различия:**

Лямбда-выражение — это анонимная функция, которая не имеет имени.

Лямбда-выражения обычно используются для краткости при передаче функций как параметров.

Лямбда-выражение имеет более компактный синтаксис: параметры указываются перед ->.

1. Как создать лямбда-выражение с двумя параметрами? Приведите пример.

val calculate = { x: Int, y: Int -> x \* y }

fun main() {

val result = calculate(7, 8)

println(result) // 56

}

# ВЫВОД

В ходы выполнения лабораторной работы было проведено ознакомление с основами функционального программирования в Kotlin, изучены лямбда-выражения, анонимные функции и замыкания.