Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Отчёт

по лабораторной работе №3

на тему:

**ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ И ЛЯМБДА-ВЫРАЖЕНИЯ**

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

(подпись)

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.И. Козел

(подпись) гр. 214302

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.

**ЗАДАНИЕ**

Вариант 12. Динамическое создание и компиляция функций: Реализуйте программу, которая принимает текстовое описание функции от пользователя, компилирует его в лямбда-выражение и выполняет вычисления на основе введенных данных.

**РЕШЕНИЕ**

fun main() {

print("Введите математическое выражение (формат: 1 + 1): ")

val input = readLine()

if (input.isNullOrEmpty()) {

println("Пустой ввод!")

return

}

val mathTask = input.split(" ")

if (mathTask.size != 3) {

println("Неверный формат!")

return

}

val op1 = mathTask[0].toDoubleOrNull()

val operator = mathTask[1]

val op2 = mathTask[2].toDoubleOrNull()

if (op1 == null || op2 == null) {

println("Ошибка с числами!")

return

}

val operation: (Double, Double) -> Double = when (operator) {

"+" -> { a, b -> a + b }

"-" -> { a, b -> a - b }

"\*" -> { a, b -> a \* b }

"/" -> { a, b -> if (b != 0.0) a / b else throw ArithmeticException("Ошибка деления на ноль!") }

else -> {

println("Ошибка с оператором!")

return

}

}

val result = operation(op1, op2)

println("Результат: $result")

}

Результат работы программы представлен на рисунке 1.

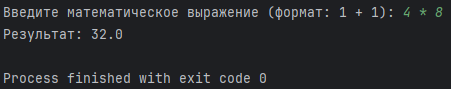


Рисунок 1 – Результат выполнения программы

**ВЫВОД**

В ходе лабораторной работы были изучены основы функционального программирования в *Kotlin*, лямбда-выражения, анонимные функции и замыкания, использованы эти концепции для написания более гибкого и читаемого кода.