

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 | _____     | Усенко Ф.В.    |
|          | (подпись) |                |
| Выполнил | _____     | Ахрамович С.П. |
|          | (подпись) | гр. 214301     |

Минск, 2024

**Цель:** 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 2. Функция для анализа больших данных.**

### **КОД ПРОГРАММЫ**

```
import kotlin.concurrent.thread

val calculateSum: (List<Int>) -> Int = { numbers -> numbers.sum() }
val calculateAverage: (List<Int>) -> Double = { numbers -> numbers.average() }
}
val calculateMedian: (List<Int>) -> Double = { numbers ->
    val sortedNumbers = numbers.sorted()
    val size = sortedNumbers.size
    if (size % 2 == 0) {
        (sortedNumbers[size / 2 - 1] + sortedNumbers[size / 2]) / 2.0
    } else {
        sortedNumbers[size / 2].toDouble()
    }
}

val analyzePart: (List<Int>) -> Triple<Int, Double, Double> = { numbers ->
    val sum = calculateSum(numbers)
    val avg = calculateAverage(numbers)
    val median = calculateMedian(numbers)
    Triple(sum, avg, median)
}

fun main() {
    val largeData = (1..10000000).toList()
    val chunkSize = 100000
    val chunks = largeData.chunked(chunkSize)

    val results = mutableListOf<Triple<Int, Double, Double>>()
    val threads = mutableListOf<Thread>()

    chunks.forEach { chunk ->
        val thread = thread {
            val result = analyzePart(chunk)
            synchronized(results) {
                results.add(result)
            }
        }
        threads.add(thread)
    }

    threads.forEach { it.join() }

    results.forEachIndexed { index, result ->
        println("Часть ${index + 1}: Сумма = ${result.first}, Среднее = ${result.second}, Медиана = ${result.third}")
    }
}
```

Результат работы программы представлен на рисунке 1.

```

"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.4\lib\idea_rt.jar=54880:C:\Program F
Часть 1: Сумма = 1078810960, Среднее = 4950000.5, Медиана = 4950000.5
Часть 2: Сумма = -1108507312, Среднее = 8450000.5, Медиана = 8450000.5
Часть 3: Сумма = -136941232, Среднее = 3950000.5, Медиана = 3950000.5
Часть 4: Сумма = 1596979536, Среднее = 2550000.5, Медиана = 2550000.5
Часть 5: Сумма = 1856063824, Среднее = 1350000.5, Медиана = 1350000.5
Часть 6: Сумма = -266483376, Среднее = 4550000.5, Медиана = 4550000.5
Часть 7: Сумма = -7399088, Среднее = 3350000.5, Медиана = 3350000.5
Часть 8: Сумма = -201712304, Среднее = 4250000.5, Медиана = 4250000.5
Часть 9: Сумма = -1417464496, Среднее = 3250000.5, Медиана = 3250000.5
Часть 10: Сумма = -1043736240, Среднее = 8150000.5, Медиана = 8150000.5
Часть 11: Сумма = 495871312, Среднее = 7650000.5, Медиана = 7650000.5
Часть 12: Сумма = 1532208464, Среднее = 2850000.5, Медиана = 2850000.5
Часть 13: Сумма = 301558096, Среднее = 8550000.5, Медиана = 8550000.5
Часть 14: Сумма = 1582081360, Среднее = 9250000.5, Медиана = 9250000.5
Часть 15: Сумма = -1223151280, Среднее = 2350000.5, Медиана = 2350000.5
Часть 16: Сумма = 316456272, Среднее = 1850000.5, Медиана = 1850000.5
Часть 17: Сумма = -1547006640, Среднее = 3850000.5, Медиана = 3850000.5
Часть 18: Сумма = 1014039888, Среднее = 5250000.5, Медиана = 5250000.5
Часть 19: Сумма = 1661750608, Среднее = 2250000.5, Медиана = 2250000.5
Часть 20: Сумма = -1482235568, Среднее = 3550000.5, Медиана = 3550000.5
Часть 21: Сумма = 122143056, Среднее = 2750000.5, Медиана = 2750000.5
Часть 22: Сумма = -1676548784, Среднее = 4450000.5, Медиана = 4450000.5

```

Рисунок 1 – Результат работы программы

## КОНТРОЛЬНЫЕ ВОПРОСЫ

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

В *Kotlin* функцию можно объявить с помощью ключевого слова *fun*. Разница между обычной и однострочной функцией заключается в том, что обычная функция может содержать несколько строк кода, и для ее тела обычно используются фигурные скобки. Однострочная функция использует выражение, которое возвращает результат, и может не требовать фигурных скобок, если тело функции состоит из одного выражения.

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

Функция высшего порядка – это функция, которая принимает другую функцию в качестве параметра или возвращает ее.

```

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

fun main() {
    val sum = operateOnNumbers(5, 3, { x, y -> x + y })
    println("Sum: $sum") // Вывод: Sum: 8

    val product = operateOnNumbers(5, 3, { x, y -> x * y })
    println("Product: $product") // Вывод: Product: 15
}

```

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

Функцию можно передать в качестве параметра, указав её тип. Тип функции определяется как (Тип\_параметра1, Тип\_параметра2) -> Тип\_возвращаемого\_значения.

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

fun main() {
    val result = calculate(10, 5, ::add) // Используем ссылку на функцию
    println("Result: $result") // Вывод: Result: 15
}

fun add(x: Int, y: Int): Int {
    return x + y
}
```

#### **4 Что такое анонимная функция, и как она отличается от лямбда-выражения?**

Анонимная функция — это функция без имени, объявленная с использованием ключевого слова *fun*. В отличие от лямбды, она может содержать сложные выражения и поддерживает метки *return*.

Пример анонимной функции:

```
val multiply = fun(a: Int, b: Int): Int {
    return a * b
}
println(multiply(3, 4)) // Вывод: 12
```

#### **5 Как создать лямбда-выражение с двумя параметрами? Приведите пример.**

Лямбда с двумя параметрами создается так:

```
val difference = { a: Int, b: Int -> a - b }
println(difference(10, 5)) // Вывод: 5
```

**Вывод:** Проведено ознакомление с основами функционального программирования в *Kotlin*, изучение лямбда-выражений, анонимных функций и замыканий. Научились использовать эти концепции для написания более гибкого и читаемого кода.