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. 214302 Усенко Ф.В.

Васильков Н.О.

Минск 2024

Задание: Замыкания с кэшированием результатов: Напишите программу, которая использует замыкания для выполнения сложных вычислений с кэшированием промежуточных результатов. Программа должна обеспечивать быструю повторную обработку данных с использованием сохраненных результатов.

Листинг кода:

fun main() {

// Создаем замыкание для вычисления факториала с кэшированием

fun createCachedFactorial(): (Long) -> Long {

val cache = mutableMapOf<Long, Long>()

return { n: Long ->

// Внутренняя функция для вычисления факториала

fun cachedFactorial(n: Long): Long {

if (n == 0L) return 1

if (cache.containsKey(n)) {

println("Возвращаем из кэша факториал($n)")

return cache[n]!!

}

println("Вычисляем факториал($n)")

val result = n \* cachedFactorial(n - 1)

cache[n] = result

return result

}

// Вызываем вычисление

cachedFactorial(n)

}

}

// Используем замыкание для кэшированного вычисления факториала

val cachedFactorial = createCachedFactorial()

while (true) {

try {

print("Введите число: ")

val number = readLine()

if (number.isNullOrBlank()) {

throw NullPointerException("Ввод не может быть пустым.")

}

if (number.toInt() < 0) {

throw NumberFormatException()

}

// Преобразуем введенное значение в Long и используем кэшированную функцию

println("Факториал $number: ${cachedFactorial(number.toLong())}")

} catch (e: NullPointerException) {

println(e.message)

} catch (e: NumberFormatException) {

println("Неправильный ввод. Введите корректное целое число.")

}

}

}

Контрольные вопросы:

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

Функция в Kotlin объявляется с использованием ключевого слова fun. Синтаксис выглядит так:

fun имяФункции(параметры): ТипВозвращаемогоЗначения {

// Тело функции

}

Пример обычной функции:

fun sum(a: Int, b: Int): Int {

return a + b

}

println(sum(2, 3)) // Вывод: 5

Однострочная функция – это функция, тело которой состоит из одной строки. В таком случае можно использовать сокращённый синтаксис с оператором =:

fun sum(a: Int, b: Int): Int = a + b

**Разница между обычной и однострочной функцией:**

* **Обычная функция** позволяет писать многострочный код с более сложной логикой.
* **Однострочная функция** используется для компактности, если логика функции помещается в одну строку.

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

Функция высшего порядка — это функция, которая:

* Принимает другую функцию в качестве параметра.
* Возвращает функцию в качестве результата.

**Пример использования:**

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

**return operation(a, b)**

**}**

**val sum = calculate(3, 5) { x, y -> x + y }**

**val multiply = calculate(3, 5) { x, y -> x \* y }**

**println("Sum: $sum") // Вывод: Sum: 8**

**println("Multiply: $multiply") // Вывод: Multiply: 15**

**3.В чем отличие анонимной функции от лямбда-выражений?**

В отличие от лямбда-выражений, анонимные функции могут содержать в себе несколько инструкций и поддерживают доступ к меткам возврата (return).

Отличия:

Синтаксис: Лямбда-выражения более краткие, анонимные функции могут содержать полный синтаксис с return и типом возвращаемого значения.

Типы: В анонимных функциях можно явным образом указывать возвращаемые типы, что иногда делает их более гибкими.

**чем лямбды и анонимные отличаются от функций высшего порядка?**

Лямбда и анонимные функции — это конкретные типы функций без имени, которые можно передавать как значения. Они являются строительными блоками для создания простых операций.

Функции высшего порядка — это функции, которые работают с другими функциями, принимая их как аргументы или возвращая их. Внутри этих функций можно использовать лямбды или анонимные функции.

Лямбды и анонимные функции — это способ определения функций,

а функции высшего порядка — это способ использования функций.

**что такое функциональные типы?**

Функциональные типы в Kotlin — это способ описания функций в виде значений, которые могут быть переданы, возвращены и хранимы в переменных.

Функциональные типы определяются как (A, B) -> C, где:

(A, B) — типы параметров, которые принимает функция.

C — тип возвращаемого значения.

Примеры:

() -> Unit — функция без параметров, не возвращающая значение.

(Int, Int) -> String — функция, принимающая два Int и возвращающая String.

**Именованные параметры** — это механизм, позволяющий указать названия параметров в объявлениях функциональных типов. Это делает код более читаемым и понятным, особенно когда функции принимают несколько параметров одного типа или когда важно понимать, что именно делает каждый параметр.

При помощи круглых скобок функциональные типы можно объединять: (Int) -> ((Int) -> Unit).