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

(подпись) гр. 214302

Минск, 2024

**Цель: Создание и обработка композиций функций с замыканиями.** 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.

**Код**

val double: (Int) -> Int = { it \* 2 }  
val increment: (Int) -> Int = { it + 1 }  
  
val upperCase: (String) -> String = { it.uppercase() }  
val addExclamation: (String) -> String = { "$it!" }  
  
val reverseList: (List<Int>) -> List<Int> = { it.reversed() }  
val addElementToList: (List<Int>) -> List<Int> = { it+2 }  
  
fun <T> compose(vararg funcs: (T) -> T): (T) -> T {  
 return { input: T ->  
 funcs.fold(input) { acc, func -> func(acc) }  
 }  
}  
  
fun selectFunctionForType(type: String): (Any) -> Any {  
 return when (type) {  
 "Int" -> {  
 println("Выберите функции для чисел: (1) Удвоить, (2) Инкрементировать")  
 val funcs = mutableListOf<(Int) -> Int>()  
 while (true) {  
 when (readln()) {  
 "1" -> funcs.add(double)  
 "2" -> funcs.add(increment)  
 "done" -> break  
 else -> println("Введите 1 или 2 для выбора функции, или 'done' для завершения.")  
 }  
 }  
 compose(\*funcs.toTypedArray()) as (Any) -> Any  
 }  
  
 "String" -> {  
 println("Выберите функции для строк: (1) Привести к верхнему регистру, (2) Добавить восклицательный знак")  
 val funcs = mutableListOf<(String) -> String>()  
 while (true) {  
 when (readln()) {  
 "1" -> funcs.add(upperCase)  
 "2" -> funcs.add(addExclamation)  
 "done" -> break  
 else -> println("Введите 1 или 2 для выбора функции, или 'done' для завершения.")  
 }  
 }  
 compose(\*funcs.toTypedArray()) as (Any) -> Any  
 }  
  
 "List" -> {  
 println("Выберите функции для списков: (1) Перевернуть список, (2) Добавить элемент 2")  
 val funcs = mutableListOf<(List<Int>) -> List<Int>>()  
 while (true) {  
 when (readln()) {  
 "1" -> funcs.add(reverseList)  
 "2" -> funcs.add(addElementToList)  
 "done" -> break  
 else -> println("Введите 1 или 2 для выбора функции, или 'done' для завершения.")  
 }  
 }  
 compose(\*funcs.toTypedArray()) as (Any) -> Any  
 }  
  
 else -> {  
 throw IllegalArgumentException("Неизвестный тип данных.")  
 }  
 }  
}  
  
fun main() {  
 val dataType = selectDataType()  
 val function = selectFunctionForType(dataType)  
  
 when (dataType) {  
 "Int" -> {  
 val input = getIntInput("Введите целое число:")  
 val result = function(input) as Int  
 println("Результат: $result")  
 }  
  
 "String" -> {  
 val input = getStringInput("Введите строку:")  
 val result = function(input) as String  
 println("Результат: $result")  
 }  
  
 "List" -> {  
 val input = getListInput("Введите элементы списка через пробел:")  
 val result = function(input) as List<Int>  
 println("Результат: $result")  
 }  
 }  
}  
  
fun selectDataType(): String {  
 while (true) {  
 try {  
 println("Выберите тип данных: (1) Int, (2) String, (3) List<Int>")  
 return when (readln().toInt()) {  
 1 -> "Int"  
 2 -> "String"  
 3 -> "List"  
 else -> throw IllegalArgumentException("Некорректный выбор.")  
 }  
 } catch (e: Exception) {  
 println("Ошибка ввода: ${e.message}. Попробуйте снова")  
 }  
 }  
}  
  
fun getIntInput(prompt: String): Int {  
 while (true) {  
 try {  
 println(prompt)  
 return readln().toInt()  
 } catch (e: NumberFormatException) {  
 println("Ошибка: введите корректное целое число")  
 }  
 }  
}  
  
fun getStringInput(prompt: String): String {  
 println(prompt)  
 return readln()  
}  
  
fun getListInput(prompt: String): List<Int> {  
 while (true) {  
 try {  
 println(prompt)  
 return readln().split(" ").map { it.toInt() }  
 } catch (e: NumberFormatException) {  
 println("Ошибка: убедитесь, что вы вводите целые числа, разделённые пробелами")  
 }  
 }  
}

**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, котрое 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

}

**Вывод**: Было изучено функциональное программирование. Была выявлена разница обычных функций от однострочной, а также познакомилась с лямбда-функцией.