Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Дисциплина: Программирование мобильных информационных систем

Отчёт

К лабораторной работе № 3

на тему:

**ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ И ЛЯМБДА-ВЫРАЖЕНИЯ**

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

(подпись)

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Шубин

(подпись) 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 для самостоятельной работы (Вариант 8).**

**Функции с саморекурсией для работы с деревьями:** Напишите программу, которая использует саморекурсивные лямбда-выражения для обхода и обработки бинарного дерева (например, подсчет листьев, поиск максимального значения).

**Код программы.**

fun main() {  
  
 *print*("Введите желаемое количество узлов в дереве: ")  
 var size = *readlnOrNull*()?.*toIntOrNull*()  
 while (size == null) {  
 *println*("Ошибка ввода. Попробуйте ещё раз")  
 size = *readlnOrNull*()?.*toIntOrNull*()  
 continue  
 }  
  
 while (true) {  
  
 *print*("Введите элементы дерева через пробел: ")  
 val array1 = *readlnOrNull*()?.*split*(" ")?.*filter* { it.*isNotEmpty*() }?.*map* { it.*toInt*() }  
  
 if (array1 == null || array1.size != size) {  
 *println*("Пожалуйста, введите $size элементов, как вы и выбрали в начале.")  
 continue  
 }  
  
 val root = Node(null, null, array1[0])  
  
 val tree = BinaryTree(root)  
  
 for(el in array1.*drop*(1)){  
 tree.insert(el)  
 }  
 tree.printTree()  
 while(true) {  
 *userMenu*(tree)  
 *print*("Желаете продолжить?\n1.Да\n НЕТ - любая другая клавиша\n")  
 val input = *readlnOrNull*()?.*toIntOrNull*()  
 if(input != 1)break  
 }  
 break  
 }  
}  
  
fun userMenu(tree: BinaryTree?){  
 while(true) {  
 *println*(  
 "Выберите операцию, которую хотели бы провести с деревом:" +  
 "\n1. Посчитать количество листьев\n2. Найти максимальное значение" +  
 "\n3. Найти минимальное значение" +  
 "\n4. Проверка наличия элемента в дереве"  
 )  
  
 val input = *readlnOrNull*()?.*toIntOrNull*()  
  
 val condition = when (input) {  
 1 -> {  
 val leafCount = tree?.countLeafNodes()  
 *println*("Количество листьев в заданном дереве: $leafCount")  
 break  
 }  
 2 ->{  
 val max = tree?.maximum()  
 *println*("Максимальное значение элемента в дереве: ${max?.value}")  
 break  
 }  
 3 -> {  
 val min = tree?.minimum()  
 *println*("Минимальное значение элемента в дереве: ${min?.value}")  
 break  
 }  
 4 -> {  
 *print*("Введите целое число, которое желаете проверить на наличие в заданном дереве: ")  
 val element = *readlnOrNull*()?.*toIntOrNull*()  
 if(tree?.findNode(element!!) == true){  
 *println*("Данное число присутствует среди элементов дерева")  
 }  
 else *println*("Данного элемента нет в дереве")  
 break  
 }  
 else -> {  
 *println*("Ошибка ввода. Выберите существующий пункт меню")  
 continue  
 }  
 }  
 }  
}

**Контрольные вопросы.**

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

Функции в Kotlin объявляются с использованием ключевого слова fun, после которого следует имя функции, параметры и возвращаемый тип. Если функция не возвращает значение, можно не указывать тип возвращаемого значения (по умолчанию используется Unit).

Пример:

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

return a + b

}

Эта функция принимает два целых числа в качестве параметров и возвращает их сумму.

Если функция состоит из одного выражения, её можно объявить как однострочную, без использования return.

Пример:

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

Здесь функция multiply возвращает произведение двух чисел.

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

Функции высшего порядка принимают другие функции как параметры. Это позволяет делать код более гибким и расширяемым. Пример:

fun calculate(x: Int, y: Int, operation: (Int, Int) -> Int): Int {

return operation(x, y)

}

fun main() {

val result = calculate(4, 2) { a, b -> a + b }

println(result) // Выводит 6

}

1. Что такое анонимная функция, и как она отличается от лямбда-выражения?

Анонимные функции похожи на лямбда-выражения, но могут иметь более сложное тело, включая несколько операторов. Анонимные функции определяются с использованием ключевого слова fun без имени функции. Пример:

val multiply = fun(a: Int, b: Int): Int {

return a \* b

}

println(multiply(3, 4))

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

1. В чем преимущества использования функций высшего порядка в программировании?

Функции высшего порядка могут также возвращать другие функции, что позволяет создавать динамически изменяемые функции. Это позволяет делать код более гибким и расширяемым.

1. Что такое функция partition, и как она может быть полезна?

Функция partition в Kotlin позволяет разделить коллекцию на две части на основе заданного условия. Она возвращает пару списков: первый список содержит элементы, которые удовлетворяют условию, а второй — те, которые не удовлетворяют. [Функция partition полезна, когда нужно быстро и удобно разделить коллекцию на две части по какому-либо критерию](https://nasmorku-net.ru/kak-rabotaet-funkciya-partition-v-kotlin/).

1. Как работает функция filter, и как её использовать для фильтрации списка строк?

Функция filter в Kotlin используется для фильтрации коллекций на основе заданного условия (предиката). Она возвращает новую коллекцию, состоящую из тех элементов исходной коллекции, которые удовлетворяют этому условию.