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.

Карлюк П.А.

Минск 2024

Вариант 8.Напишите программу, которая использует саморекурсивные лямбда-выражения для обхода и обработки бинарного дерева (например, подсчет листьев, поиск максимального значения).

Листинг кода:

data class TreeNode(  
 val value: Int,  
 val left: TreeNode? = null,  
 val right: TreeNode? = null  
)  
  
fun main() {  
 val tree = TreeNode(  
 10,  
 left = TreeNode(5, left = TreeNode(3, left = TreeNode(2), right = TreeNode(4)), right = TreeNode(7)),  
 right = TreeNode(20, left = TreeNode(15), right = TreeNode(30))  
 )  
  
 val countLeaves: (TreeNode?) -> Int = run {  
 var countLeavesRef: (TreeNode?) -> Int = { 0 }  
 countLeavesRef = { node ->  
 if (node == null) {  
 0  
 } else if (node.left == null && node.right == null) {  
 1  
 } else {  
 countLeavesRef(node.left) + countLeavesRef(node.right)  
 }  
 }  
 countLeavesRef  
 }  
  
 val findMax: (TreeNode?) -> Int = run {  
 var findMaxRef: (TreeNode?) -> Int = { 0 }  
 findMaxRef = { node ->  
 if (node == null) {  
 Int.MIN\_VALUE  
 } else {  
 val leftMax = findMaxRef(node.left)  
 val rightMax = findMaxRef(node.right)  
 maxOf(node.value, leftMax, rightMax)  
 }  
 }  
 findMaxRef  
 }  
  
 val getHeight: (TreeNode?) -> Int = run {  
 var heightLambda: (TreeNode?) -> Int = { 0 }  
 heightLambda = { node ->  
 if (node == null) 0 else 1 + maxOf(heightLambda(node.left), heightLambda(node.right))  
 }  
 heightLambda  
 }  
  
 val printTreeLevelOrder: (TreeNode?) -> Unit = { root ->  
 val height = getHeight(root)  
 val maxWidth = 2.0.pow(height).toInt() - 1  
 val queue = LinkedList<TreeNode?>().apply { add(root) }  
  
 for (level in 0..<height) {  
 val levelSize = queue.size  
 val spaceBetweenNodes = maxWidth / (2.0.pow(level).toInt() )  
  
 val spaceBetweenPairs = " ".repeat(spaceBetweenNodes + 1)  
 val spaceInsideNode = " ".repeat(spaceBetweenNodes / 2)  
  
 val levelOutput = StringBuilder(" ".repeat(spaceBetweenNodes / 2))  
  
 repeat(levelSize) {  
 val node = queue.poll()  
  
 if (node != null) {  
 levelOutput.append(spaceInsideNode)  
 levelOutput.append(node.value)  
 levelOutput.append(spaceInsideNode)  
 queue.add(node.left)  
 queue.add(node.right)  
 } else {  
 levelOutput.append(spaceInsideNode)  
 levelOutput.append(" ")  
 levelOutput.append(spaceInsideNode)  
 queue.add(null)  
 queue.add(null)  
 }  
  
 levelOutput.append(spaceBetweenPairs)  
 }  
  
 println(levelOutput)  
 }  
 }  
  
 println("Количество листьев: ${countLeaves(tree)}")  
 println("Максимальное значение: ${findMax(tree)}")  
 println("Деревце: ")  
 printTreeLevelOrder(tree)  
  
}

Контрольные вопросы:

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

В *Kotlin* функции можно объявлять с помощью ключевого слова *fun*. Есть два основных типа функций: обычные и однострочные.

Обычная функция:

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

return a + b

}

Однострочная функция:

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

Основное отличие однострочной функции от обычной заключается в отсутствии *return*.

1. 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 функцию. Это позволяет создавать гибкие и переиспользуемые компоненты.

Пример:

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

return operation(a, b)

}

fun main() {

val sum = calculate({ x, y -> x + y }, 5, 3) // 8

val product = calculate({ x, y -> x \* y }, 5, 3) // 15

println("Sum: $sum, Product: $product")

}

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

Ниже представлен пример, где функция *calculate* принимает другую функцию в качестве параметра.

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

return operation(a, b)

}

fun main() {

val result = operateOnNumbers(10, 5, { x, y -> x - y }) // 5

println("Result: $result")

}