Министерство образования Республики Беларусь

Учреждение высшего образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ОТЧЕТ

по лабораторной работе №2

на тему

**Работа со списками и функциями**

|  |  |
| --- | --- |
| Студенты группы 150501  Проверила | Михалович Т.В.  Петров В.В.  Герман Ю. О. |

Минск 2023

**Цель:** изучить технику работы со списками и функциями в Scala.

**Краткие теоретические сведения:**

List - это коллекция данных, которая представляет собой упорядоченную последовательность элементов одного типа. Она является неизменяемой (immutable) структурой данных, что означает, что после создания список нельзя изменить, а вместо этого создаются новые списки на основе существующих.

Чтобы создать список в Scala, вы можете использовать ключевое слово List, за которым следуют элементы списка, заключенные в круглые скобки. Вот пример:

val myList = List(1, 2, 3, 4, 5)

Хэш-функция - это функция, которая принимает на вход произвольные данные (например, строку, файл, или байтовый массив) и преобразует их в фиксированный набор байтов, который обычно называется "хэш-значением" или "хэш-кодом". Основной целью хэш-функции является быстрое создание уникального идентификатора (хэша) для данных так, чтобы, даже если исходные данные незначительно изменятся, их хэш-коды будут сильно различаться. Хэш-функции широко используются в информационной безопасности, в хэш-таблицах для быстрого поиска и многих других областях.

**Выполнение работы:**

Перед выполнением основного задания лабораторной работы мы выполнили рекомендованные к изучению примеры.

В первом примере мы познакомились с функцией для работы со списками map в языке программирования Scala. Результатом исполнения программы является нижеприведенный скриншот:

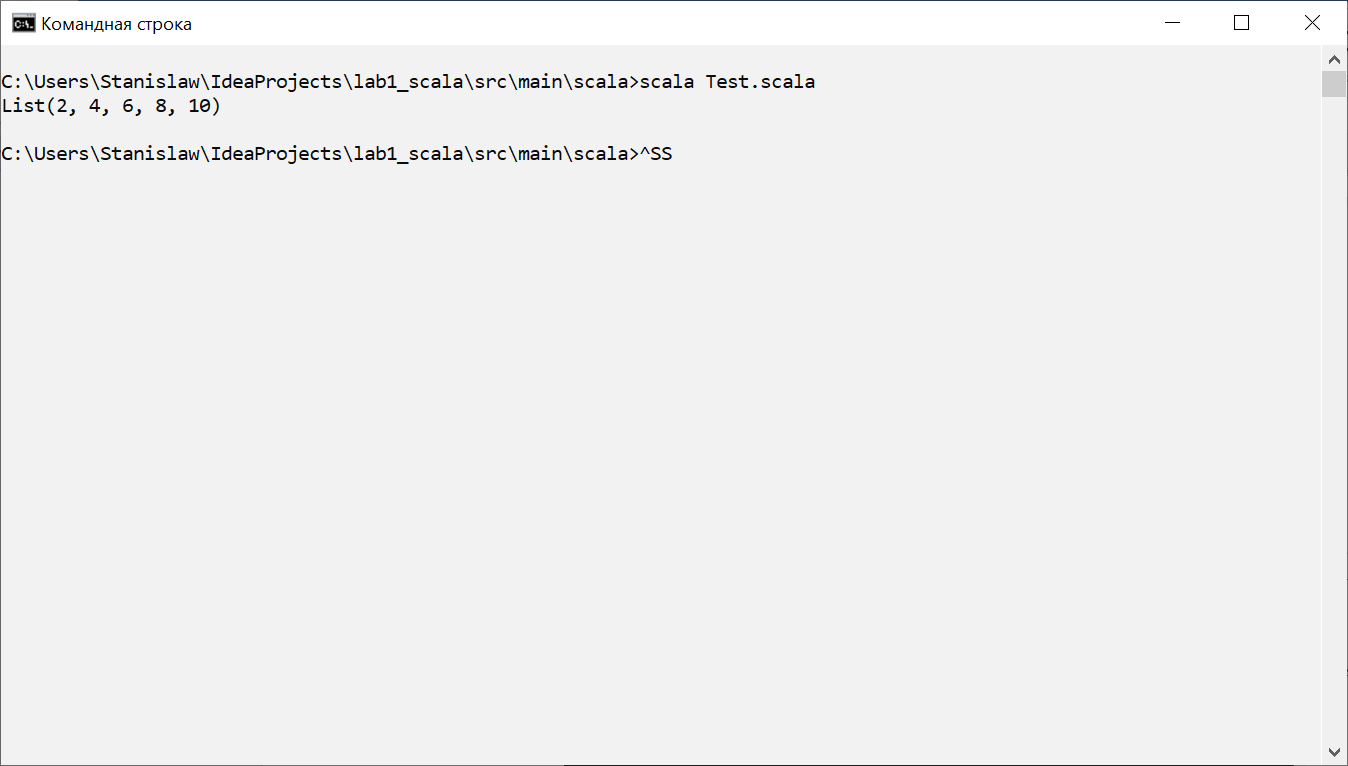


Рисунок 1 — Результат выполнения первого примера

Во втором примере с помощью языка Scala выводили четные элементы списка с помощью функции filter. Результатом исполнения программы является нижеприведенный скриншот:

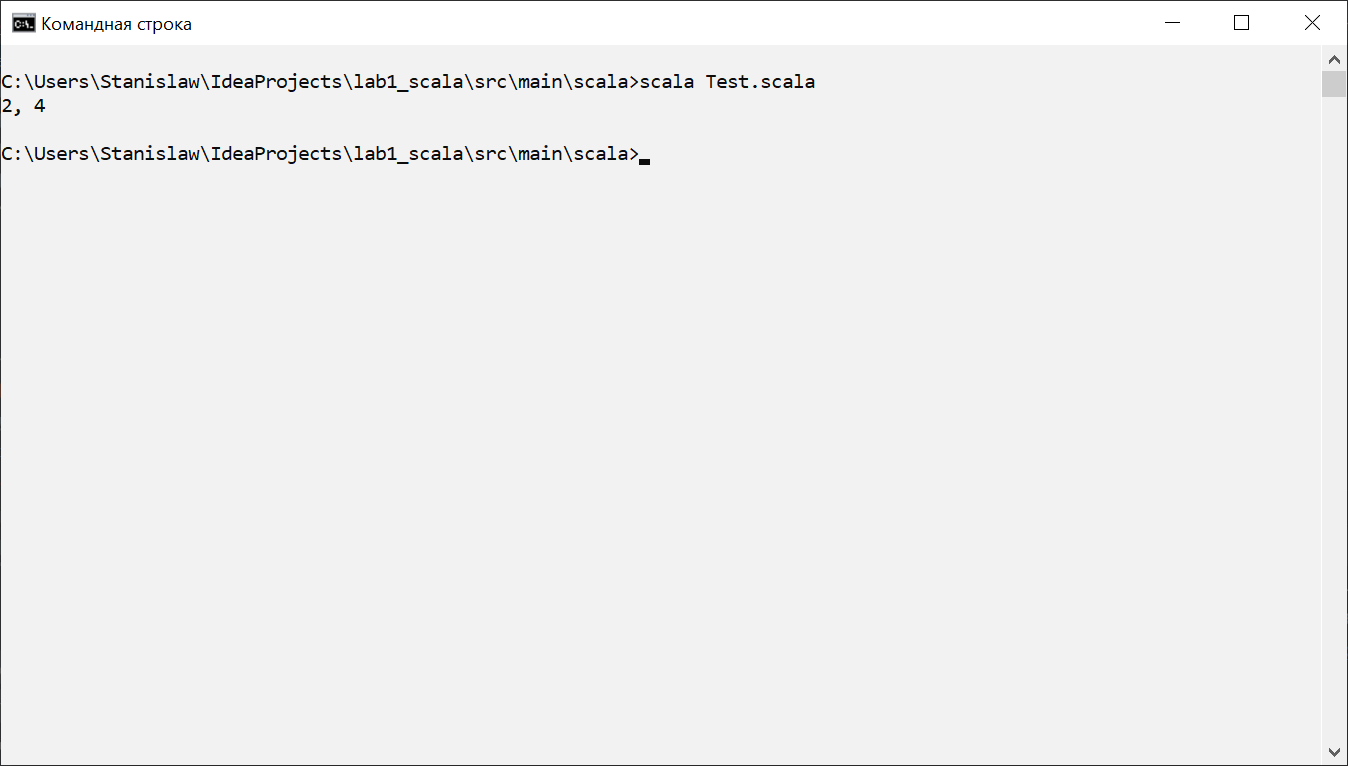


Рисунок 2 — Результат выполнения второго примера

В третьем примере с помощью языка Scala мы находили сумму элементов списка с помощью функции foldLeft. Результатом исполнения программы является нижеприведенный скриншот:

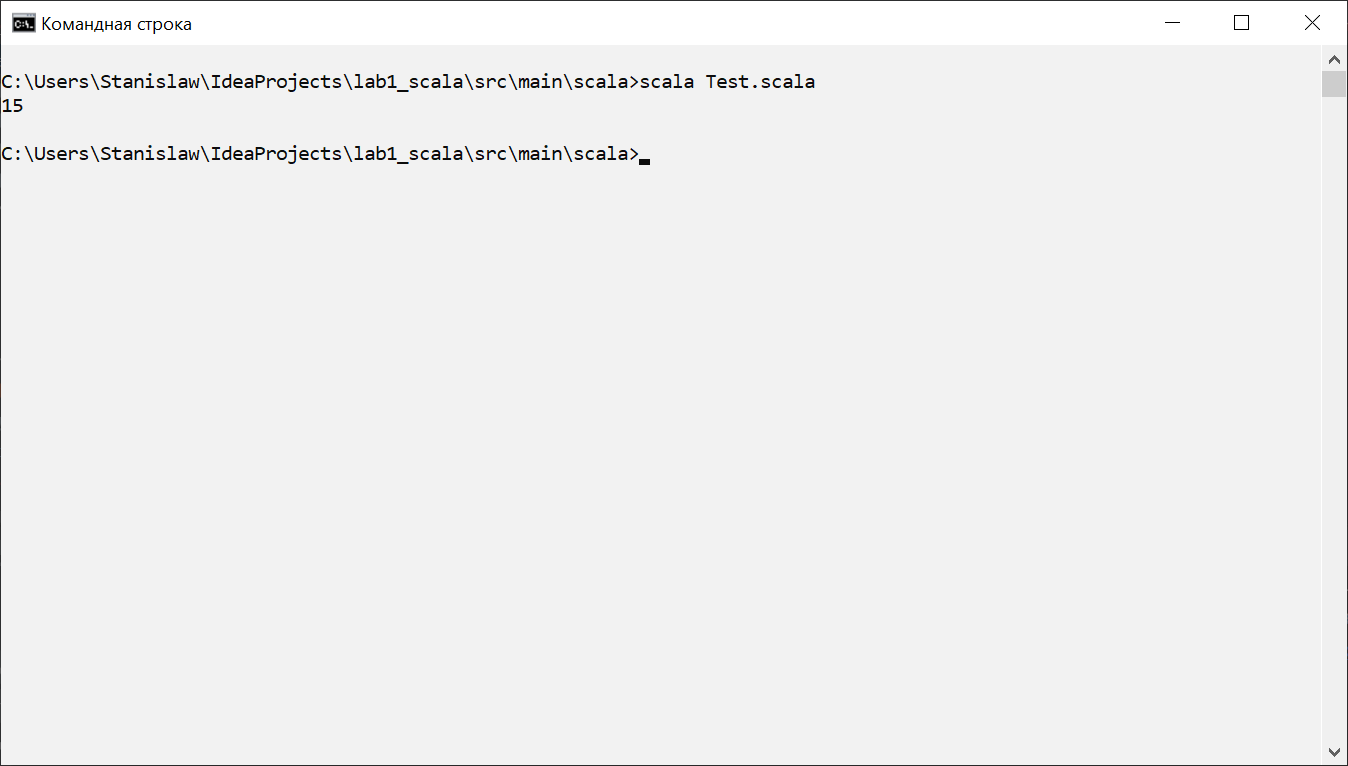


Рисунок 3 — Результат выполнения третьего примера

В четвертом примере с помощью языка Scala мы брали значение из пары в списке по ключу с помощью функции zip. Результатом исполнения программы является нижеприведенный скриншот:

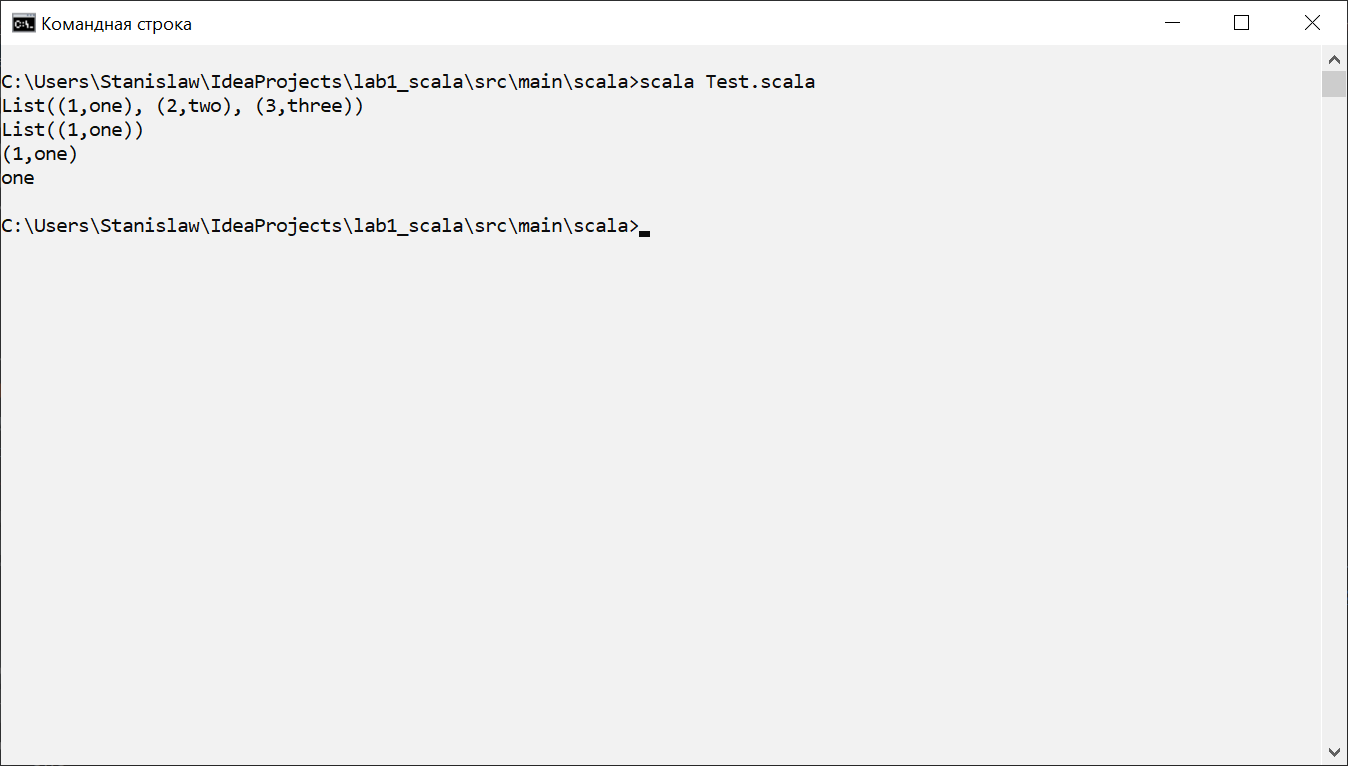


Рисунок 4 — Результат выполнения третьего примера

В пятом примере с помощью языка Scala мы находили сумму квадратов элементов списка с помощью функций head и tail. Результатом исполнения программы является нижеприведенный скриншот:

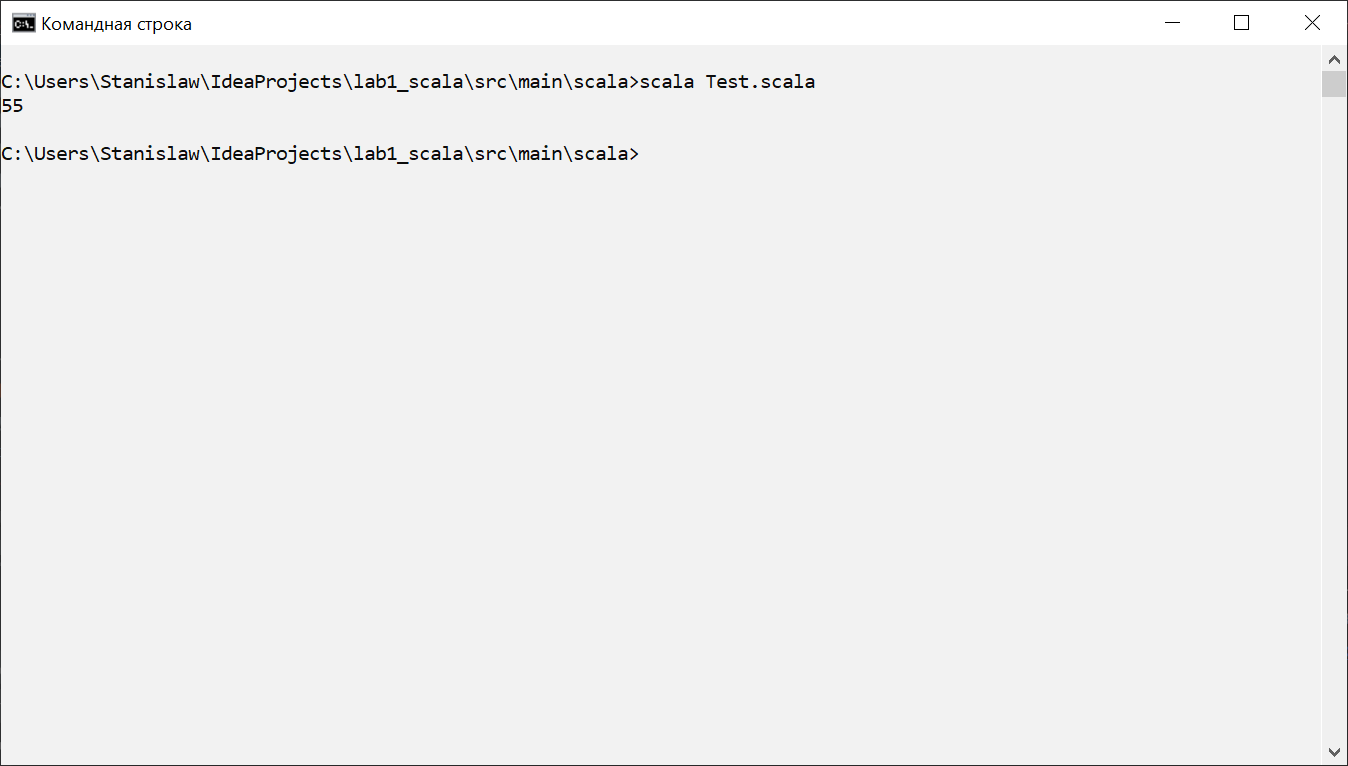


Рисунок 5 — Результат выполнения третьего примера

В ходе выполнения лабораторной работы был произведен анализ 5-ти заданий.

Задание 1: Написать функцию для подсчета суммы элементов списка, значение которых не превосходит 10. Список задать самостоятельно.

Результаты выполнения задания представлен скриншотам ниже:

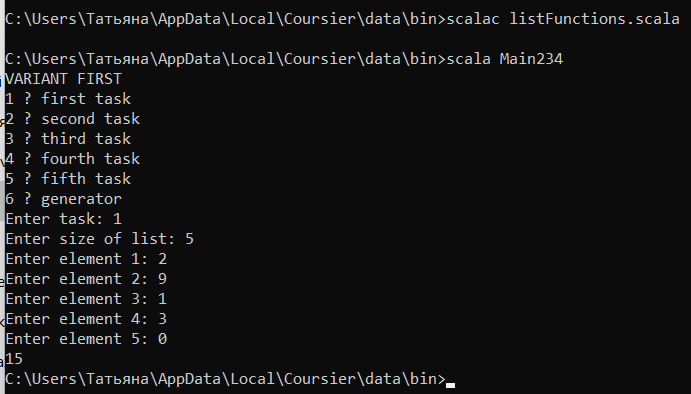


Рисунок 6 – Результат выполнения 1 задания

Задание 2: Написать функцию для подсчета суммы первых трех элементов списка из 10 элементов. Список задать самостоятельно.

Результаты выполнения задания представлен скриншотам ниже:

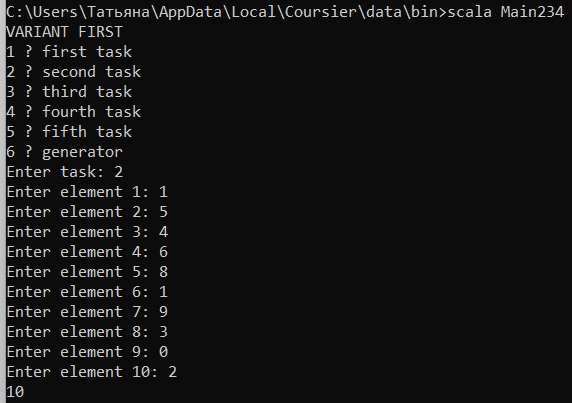


Рисунок 7 – Результат выполнения 2 задания

Задание 3: Написать функцию для отыскания (минимального) индекса максимального элемента списка. Список задать самостоятельно.

Результаты выполнения задания представлен скриншотам ниже:

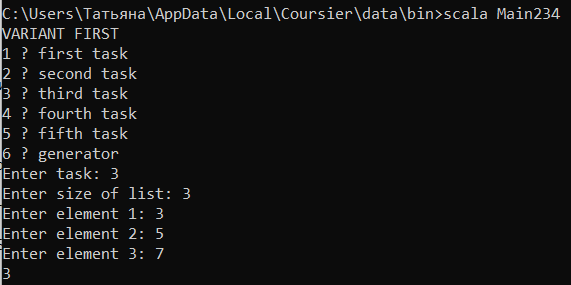


Рисунок 8 – Результат выполнения 3 задания

Задание 4: Написать функцию для проверки того, что список упорядочен по возрастанию. Список задать самостоятельно.

Результаты выполнения задания представлен скриншотам ниже:

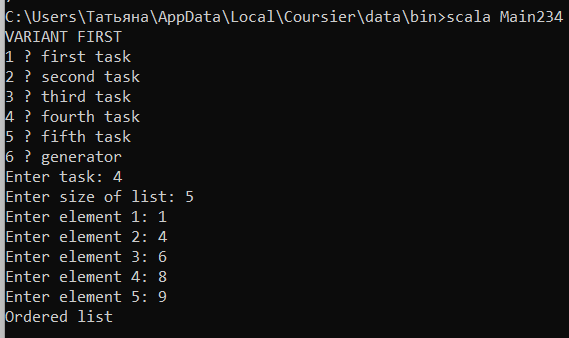


Рисунок 9 – Результат выполнения 4 задания

Задание 5: Написать функцию для проверки наличия одинаковых элементов в списке. Список задать самостоятельно. Функция возвращает значение Да или Нет

Результаты выполнения задания представлен скриншотам ниже:

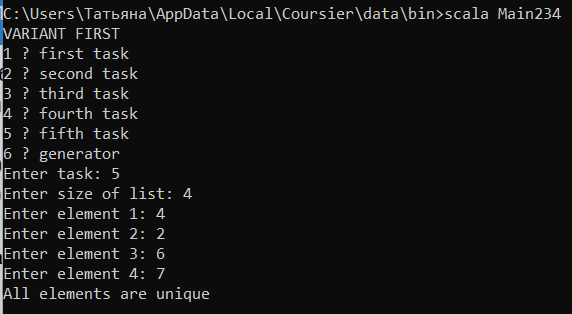


Рисунок 10 – Результат выполнения 5 задания

Листинг кода программы:

object Main234 {

def main(args: Array[String]): Unit = {

print("VARIANT FIRST\n" +

"1 — first task \n" +

"2 — second task\n" +

"3 — third task\n" +

"4 — fourth task\n" +

"5 — fifth task\n" +

"6 — generator\n" +

"Enter task: ")

val choice = scala.io.StdIn.readInt()

if(choice == 1){

print("Enter size of list: ")

val listSize = scala.io.StdIn.readInt()

val list = defineList(listSize)

print(sumOfListElements(list))

} else if(choice == 2){

val list = defineList(10)

print(SumOfFirstThreeElements(list, 0))

} else if(choice == 3){

print("Enter size of list: ")

val listSize = scala.io.StdIn.readInt()

val list = defineList(listSize)

print(searchIndexOfMaxElement(list, -1, 0, 0))

} else if(choice == 4){

print("Enter size of list: ")

val listSize = scala.io.StdIn.readInt()

val list = defineList(listSize)

if(list.nonEmpty) {

if (checkExchange(list.tail, list.head)) { print("Ordered list\n") }

else {print("Unordered list\n")}

} else{

print("Empty list\n")

}

} else if(choice == 5){

print("Enter size of list: ")

val listSize = scala.io.StdIn.readInt()

val list = defineList(listSize)

val element = list.head

val partOfList = list.tail

if(checkUniqueElements(list, partOfList, element)){

print("All elements are unique\n")

} else{

print("Not all elements are unique\n")

}

} else if(choice == 6){

generator()

}

}

// первая задача

def sumOfListElements(list: List[Int]): Int = {

if(list.nonEmpty) {

val element = list.head

var partOfList = list.tail

element + sumOfListElements(partOfList)

} else {

0

}

}

// вторая задача

def SumOfFirstThreeElements(list: List[Int], attempt: Int): Int = {

if(list.nonEmpty && attempt != 3) {

val element = list.head

var partOfList = list.tail

val newAttempt = attempt + 1

element + SumOfFirstThreeElements(partOfList, newAttempt)

} else {

0

}

}

def defineList(listSize: Int): List[Int] = {

var list = List.empty[Int]

for(i <- 1 to listSize){

print(s"Enter element $i: ")

val element = scala.io.StdIn.readInt()

list = list :+ element

}

list

}

// третья задача

def searchIndexOfMaxElement(list: List[Int], maxElement: Int, index: Int, itteration: Int): Int ={

if(list.nonEmpty) {

val element = list.head

val partOfList = list.tail

if(maxElement < element){

searchIndexOfMaxElement(partOfList, element, itteration + 1, itteration + 1)

} else{

searchIndexOfMaxElement(partOfList, element, index, itteration + 1)

}

} else {

index

}

}

// четвёртая задача

def checkExchange(list: List[Int], leftElement: Int): Boolean ={

if(list.nonEmpty) {

val rightElement = list.head

val partOfList = list.tail

if(leftElement < rightElement){

checkExchange(partOfList, rightElement)

} else{

false

}

} else {

true

}

}

// пятая задача

def checkUniqueElements(list: List[Int], partOfList: List[Int], currentElement: Int): Boolean ={

if(list.nonEmpty){

if(partOfList.nonEmpty){

if(currentElement == partOfList.head){

return false

}

checkUniqueElements(list, partOfList.tail, currentElement)

} else{

checkUniqueElements(list.tail, list.tail, list.head)

}

} else{

true

}

}

def generator(): Unit = {

print("Input count: ");

var count: Int = scala.io.StdIn.readInt();

var part: Double = 100.0 / count;

var list = List.empty[Double];

def generate(start: Double = 0.0): Unit = {

var end: Double = start + part;

if(end > 100.0) return;

var number: Double = Math.sqrt(start \* end);

if(start == 0.0){

number = Math.sqrt(end);

}

list = list :+ number;

println(s"${number} ( ${start} , ${end} )");

generate(start + part);

}

generate();

}

}

Вывод: В ходе выполнения данной лабораторной работы нами были проанализированы и изучены техники работы и использования функций и листов в Scala.