БГУИР

# Кафедра ЭВМ

### Отчет по лабораторной работе № 2

**Тема: «Работа со списками и функциями»**

Выполнил:

студенты группы 130501 Гнездилов А.М.

Проверила:

**к.т.н., доцент Герман Ю.О.**

#### Минск 2023

1. **ЦЕЛЬ РАБОТЫ**

Изучить технику работы со списками и функциями в Scala

1. **КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Map (карта,отображение) : эта функция применяет заданную функцию к каждому элементу списка и возвращает новый список с результатами.

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

val doubled = numbers.map(x => x \* 2) // List(2, 4, 6, 8, 10)

object Main22 {

def double(x: Int): Int = x \* 2

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

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

val doubledList = myList.map(double)

println(doubledList) // Output: List(2, 4, 6, 8, 10)

}

}

Поэлементный вывод списка

object Main22 {

def double(x: Int): Int = x \* 2

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

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

val doubledList = myList.map(double)

println(myList.mkString(", "))

}

}

Следующий пример показывает, как вывести четные элементы списка

object Main22 {

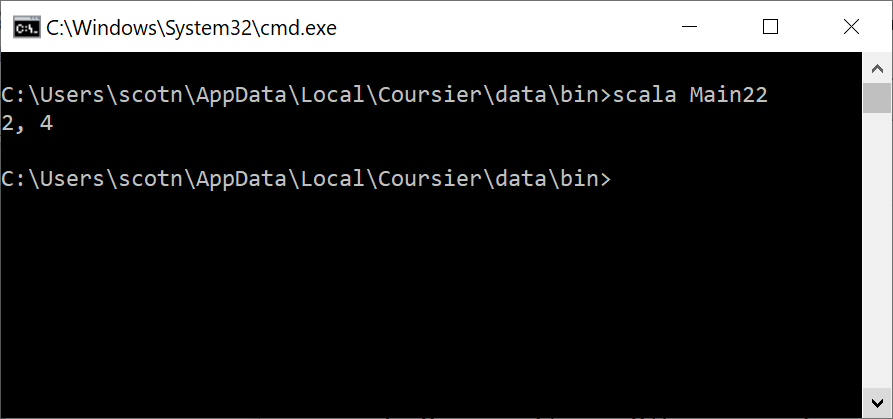
def isEven(x: Int): Boolean = x % 2 == 0

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

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

val filteredList = myList.filter(isEven)

println(filteredList.mkString(", ")) // Output: List(2, 4)}}



foldLeft: Эта функция последовательно применяется к элементам списка слева направо, накапливая результат. Сумму элементов списка можно найти таким образом

object Main22 {

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

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

val sum = myList.foldLeft(0)((ac\_c, x) => ac\_c + x)

println(sum) // Output: 15

}

}

Здесь переменная ac\_c играет роль аккумулятора. Первоначально ей присваивается значение 0:

myList.foldLeft(0)

zip: Эта функция объединяет два списка на примере словаря (dictionary) – ключ-значение.

val a = List(1, 2, 3)

val b = List("one", "two", "three")

val zipped = a.zip(b) // List((1, "one"), (2, "two"), (3, "three"))

Потом возникает вопрос, как взять значение из пары в списке по ключу. Последовательно покажем решение этой задачи

object Main22 {

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

val a = List(1, 2, 3)

val b = List("one", "two", "three")

val zipped = a.zip(b) // List((1, "one"), (2, "two"), (3, "three"))

println(zipped)

val filteredList = zipped.filter { case (a, \_) => a == 1 }

println(filteredList)

val tup=filteredList.head

println(tup)

val secondItem = tup.\_2

println(secondItem)

}

}

Обратимся к рекурсивным функциям. Со списками их показывать очень удобно.

head and tail: Эти функции возвращают голову и хвост списка соответственно.

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

val first = numbers.head // 1

val rest = numbers.tail // List(2, 3, 4, 5)

Сумма квадратов элементов списка:

object Main22 {

def sumList(lst: List[Int]): Int = {

if (lst.isEmpty) 0

else lst.head\*lst.head + sumList(lst.tail)

}

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

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

val sumw = sumList(myList)

println(sumw)

}

}

Этот же пример можно реализовать по-другому

object Main22 {

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

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

val sumOfSquares = myList.map(item => item \* item).sum

println(sumOfSquares)

}

}

Подсчет числа отрицательных элементов списка (почти предыдущий вариант)

object Main22 {

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

val myList = List(1, -2, 3, -7, -5)

val countNegative = myList.count(item => item < 0)

println(countNegative)

}

}

1. **ХОД РАБОТЫ**

**Вариант 2**

1. Написать функцию для подсчета суммы отрицательных элементов списка. Список задать самостоятельно.
2. Написать функцию для подсчета суммы последних трех элементов списка из 10 элементов. Список задать самостоятельно.
3. Написать функцию для отыскания индексов всех максимальных элементов списка. Список задать самостоятельно.
4. Написать функцию для проверки того, что список не упорядочен ни по возрастанию, ни по убыванию. Список задать самостоятельно.
5. Написать функцию для проверки наличия трех одинаковых элементов в списке. Список задать самостоятельно. Функция возвращает значение такого элемента.

**Листинг кода**

object Main extends App {

// Функция для подсчета суммы отрицательных элементов списка

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

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

if (list.isEmpty) 0

else if (list.head < 0) list.head + sum(list.tail)

else sum(list.tail)

}

sum(list)

}

// Функция для подсчета суммы последних трех элементов списка из 10 элементов

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

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

if (list.isEmpty) 0

else if (list.length <= 3) list.head + sumLastThree(list.tail)

else sumLastThree(list.tail)

}

sumLastThree(list)

}

// Функция для отыскания индексов всех максимальных элементов списка

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

val max = list.max

def findIndices(list: List[Int], currentIndex: Int = 0): List[Int] = {

def indices(list: List[Int], currentIndex: Int): List[Int] = {

if (list.isEmpty) Nil

else if (list.head == max) currentIndex :: indices(list.tail, currentIndex + 1)

else indices(list.tail, currentIndex + 1)

}

indices(list, currentIndex)

}

findIndices(list)

}

// Функция для проверки, что список не упорядочен ни по возрастанию, ни по убыванию

def isUnordered(list: List[Int]): Boolean = {

def isIncreasing(list: List[Int]): Boolean = {

def increasing(list: List[Int]): Boolean = {

if (list.isEmpty) true

else if (list.tail.isEmpty) true

else if (list.head <= list.tail.head) increasing(list.tail)

else false

}

increasing(list)

}

def isDecreasing(list: List[Int]): Boolean = {

def decreasing(list: List[Int]): Boolean = {

if (list.isEmpty) true

else if (list.tail.isEmpty) true

else if (list.head >= list.tail.head) decreasing(list.tail)

else false

}

decreasing(list)

}

!isIncreasing(list) && !isDecreasing(list)

}

// Функция для проверки наличия трех одинаковых элементов в списке и возврата значения такого элемента

def findThreeIdentical(list: List[Int]): Option[Int] = {

def findIdentical(list: List[Int], countMap: Map[Int, Int] = Map.empty): Option[Int] = {

if (list.isEmpty) {

val candidate = countMap.find { case (\_, count) => count >= 3 }

if (candidate.isDefined) Some(candidate.get.\_1)

else None

} else {

val updatedCountMap = countMap.updatedWith(list.head) {

case Some(count) => Some(count + 1)

case None => Some(1)

}

findIdentical(list.tail, updatedCountMap)

}

}

findIdentical(list)

}

// Функция для ввода списка с клавиатуры

def inputList(): List[Int] = {

println("Введите элементы списка через пробел:")

val input = scala.io.StdIn.readLine()

val elements = input.split(" ").map(\_.toIntOption)

if (elements.forall(\_.isDefined))

elements.map(\_.get).toList

else {

println("Ошибка ввода. Повторите ввод.")

inputList()

}

}

val myList = inputList()

if (myList.isEmpty) {

println("Список пуст.")

} else {

println("1. Сумма отрицательных элементов списка: " + sumOfNegatives(myList))

println("2. Сумма последних трех элементов списка: " + sumOfLastThree(myList))

println("3. Индексы всех максимальных элементов списка: " + findMaxIndices(myList))

println("4. Список не упорядочен ни по возрастанию, ни по убыванию: " + isUnordered(myList))

val threeIdentical = findThreeIdentical(myList)

if (threeIdentical.isDefined) {

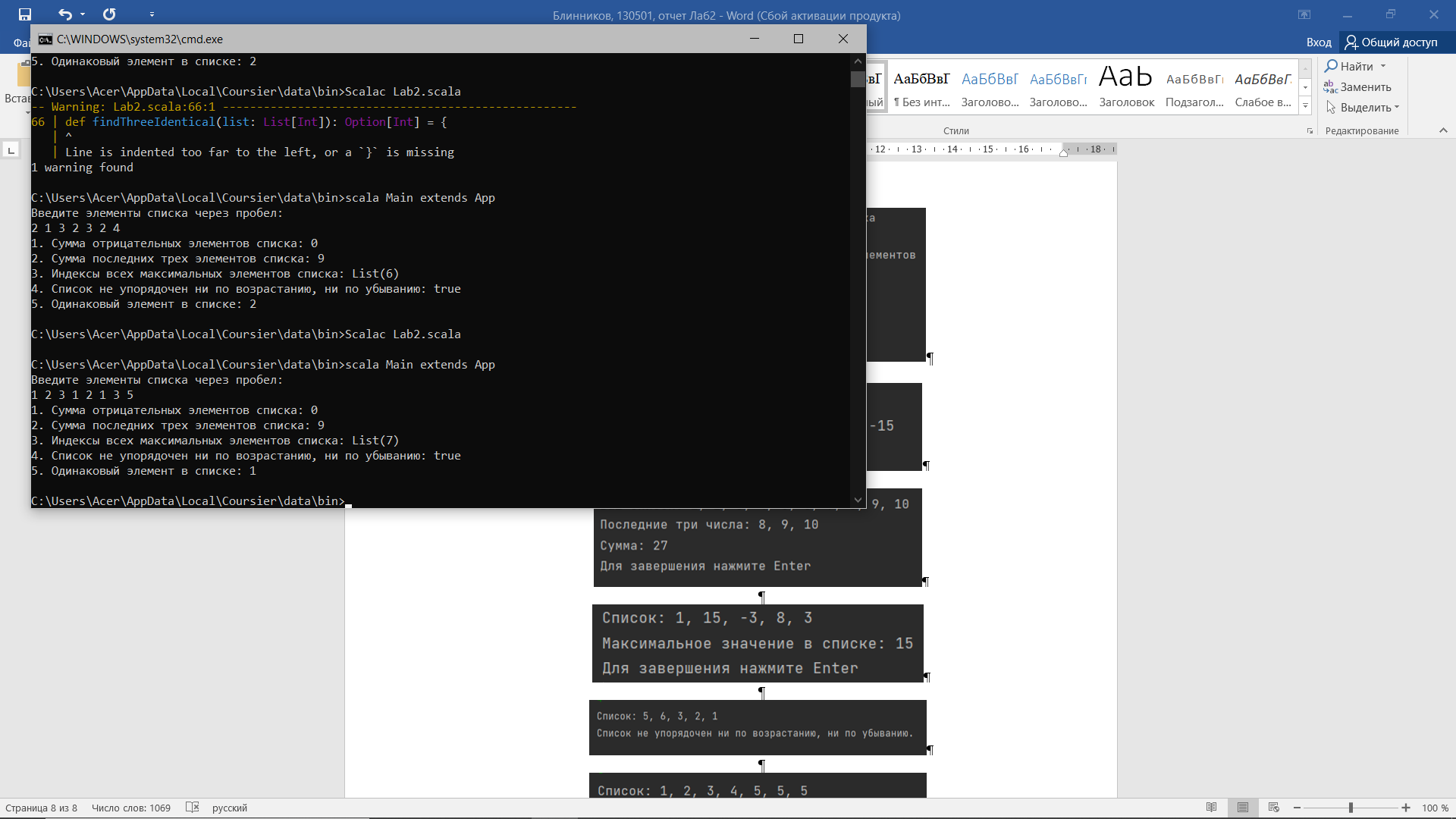
println(s"5. Одинаковый элемент в списке: ${threeIdentical.get}")

} else {

println("5. В списке нет трех одинаковых элементов.")

}

}



1. **ВЫВОД**

Изучили технику работы со списками и функциями в Scala. Научились объединять списки, использовать рекурсивные функции при работе со списками и брать значение из пары в списке по ключу.