БГУИР

Кафедра ЭВМ

Отчет по лабораторной работе №1

Тема: «Работа со списками и функциями»

Вариант 2

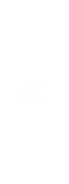
Проверила:

Герман Ю.О.

Выполнили:

Климович А.Н.

Тамашеня В.В.

Минск 2023



**1 ЦЕЛЬ**

Работа со списками и функциями

**2 КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ**

Map в Scala – это коллекция, которая представляет собой набор пар ключ-значение. Ключи должны быть уникальными, а значения могут повторяться. Ключи и значения могут быть любого типа, но они должны быть определены в момент создания Map. Map имеет множество методов для работы с ними, таких как добавление новых элементов, удаление элементов, изменение значений и т.д. Кроме того, Map поддерживает итерацию по элементам с помощью цикла for или метода foreach.

head и tail являются методами коллекций в Scala. Метод head возвращает первый элемент коллекции. Метод tail возвращает все элементы коллекции, кроме первого. Например, если у нас есть список List(1, 2, 3), вызов метода tail вернет список List(2, 3). Если коллекция пуста, то вызов метода head вызовет исключение NoSuchElementException, а вызов метода tail вернет пустую коллекцию.

**3 ХОД РАБОТЫ**

**3.1 Анализ примеров**

**3.1.1** Функция Map для работы со списками в Scala.

Далее представлен код примера 1:

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)

}

}

Результат выполнения данного примера представлен на рисунке 3.1:

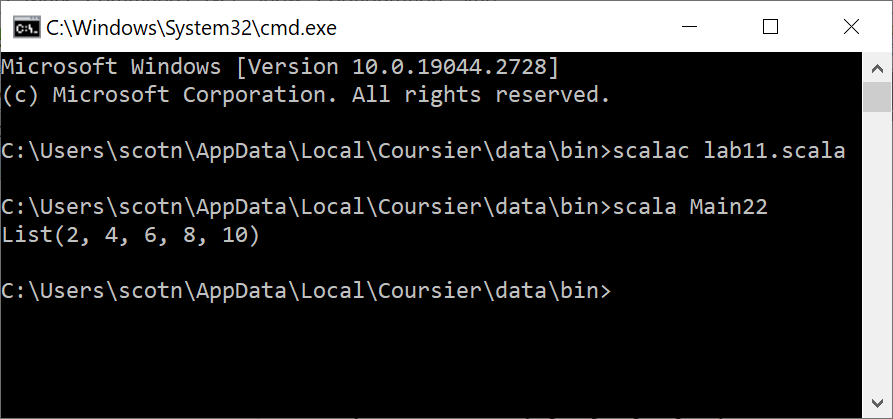


Рисунок 3.1 – Результат выполнения примера 1

**3.1.2** Поэлементный вывод списка.

Далее представлен код примера 2:

object Main22 {

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

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

println(myList.mkString(", "))

}

}

Результат выполнения данного примера представлен на рисунке 3.2:

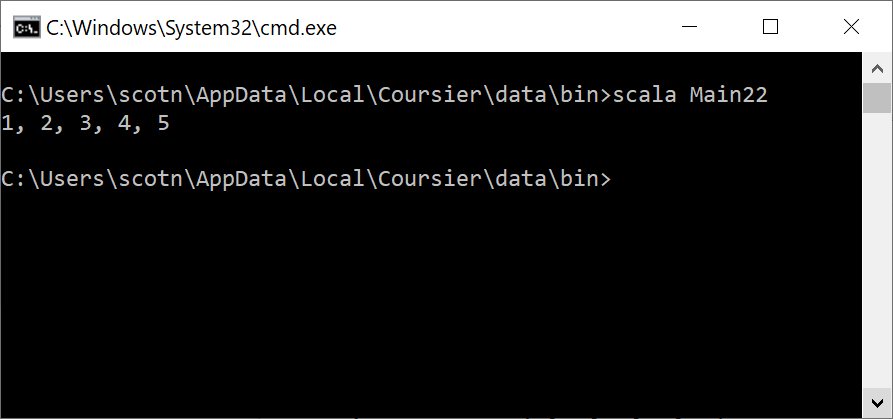


Рисунок 3.2 – Результат выполнения примера 2

**3.1.3** Функция filter, отбирающая элементы списка, удовлетворяющих заданному предикату.

Далее представлен код примера 3:

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)

}

}

Результат выполнения данного примера представлен на рисунке 3.3.

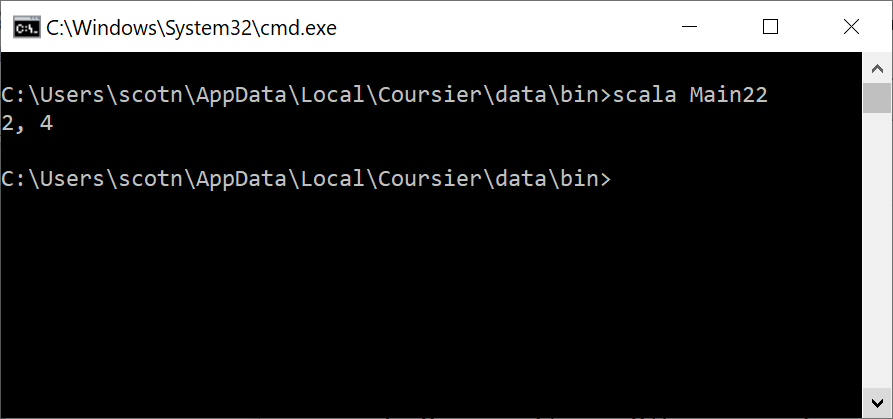


Рисунок 3.3 – Результат выполнения примера 3

**3.1.4** Функция zip – объединяет два списка на примере словаря (dictionary) – ключ-значение.

Далее представлен код примера 4:

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)

}

}

Результат выполнения данного примера представлен на рисунке 3.4.

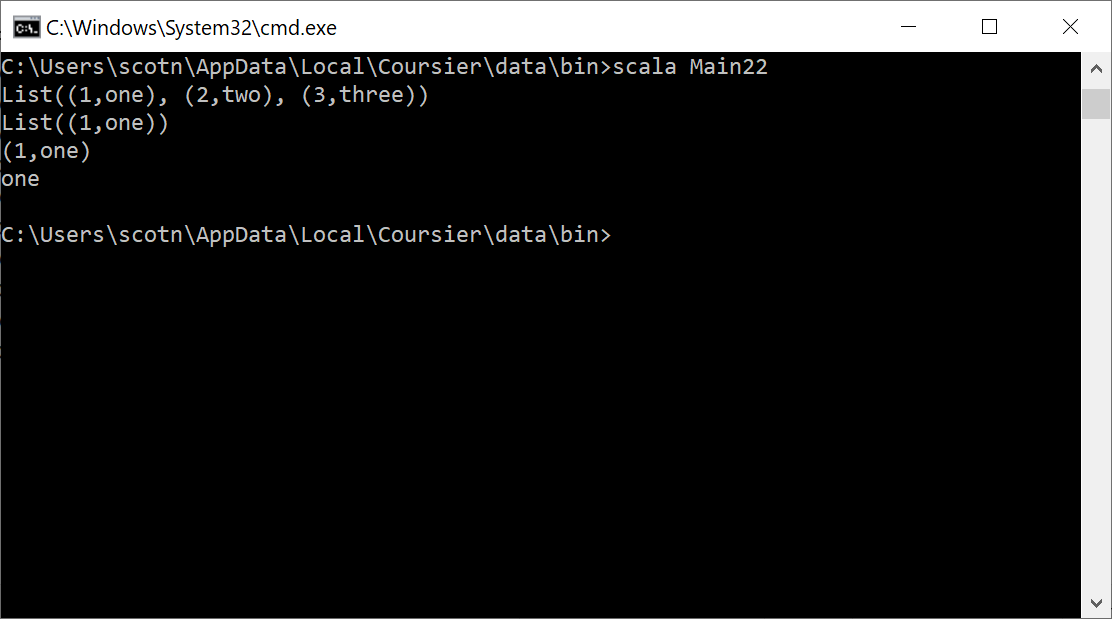


Рисунок 3.4 – Результат выполнения примера 4

**3.2 Задания варианта 2**

**3.2.1**Написать функцию для подсчета суммы отрицательных элементов списка. Список задать самостоятельно.

Далее представлен код программы задания 1:

object Main1 {

def negative\_sum(lst: List[Int]): Int = {

val negative\_numbers = lst.filter(x => x < 0)

if(negative\_numbers.isEmpty) 0

else negative\_numbers.sum

}

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

println("Enter elements of the list (integers), separated by spaces:")

val list = scala.io.StdIn.readLine().split(" ").map(\_.toInt).toList

println("Entered list: " + list)

if(list.isEmpty) {

println("The list is empty")

} else {

println("Sum of negative numbers: " + negative\_sum(list))

}

}

}

Результат выполнения задания 1 представлен на рисунке 3.5.

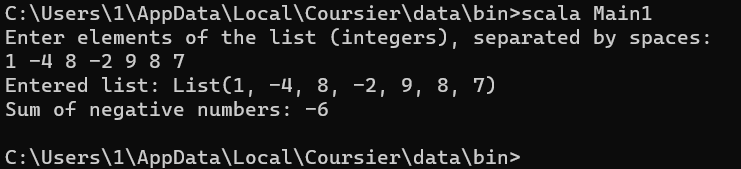


Рисунок 3.5 – Результат выполнения задания 1

**3.2.2**Написать функцию для подсчета суммы последних трех элементов списка из 10 элементов. Список задать самостоятельно.

Далее представлен код программы задания 2:

object Main2 {

def sum\_last\_3(lst: List[Int], index : Int): Int = {

if(lst.isEmpty)

return 0

if(index > 6)

lst.head + sum\_last\_3(lst.tail, index + 1)

else

sum\_last\_3(lst.tail, index + 1)

}

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

println("Enter 10 elements of the list (integers), separated by 'Enter':")

var list = List[Int]()

for(i <- 1 to 10)

list = list :+ scala.io.StdIn.readInt()

println("Entered list: " + list)

if(list.isEmpty) {

println("The list is empty")

} else {

println("Sum of last three numbers: " + sum\_last\_3(list, 0))

}

}

}

Результат выполнения задания 2 представлен на рисунке 3.6.

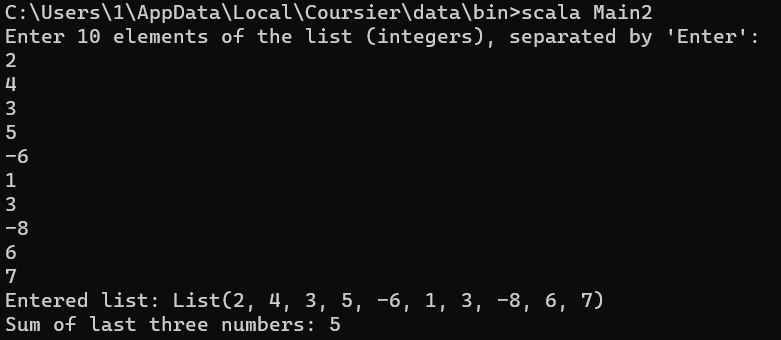


Рисунок 3.6 – Результат выполнения задания 2

**3.2.3**Написать функцию для отыскания индексов всех максимальных элементов списка. Список задать самостоятельно.

Далее представлен код программы задания 3:

var indexes = List[Int]()

object Main3 {

def find\_max(lst: List[Int], max : Int): Int = {

if(lst.isEmpty) return max

if(lst.head > max)

find\_max(lst.tail, lst.head)

else

find\_max(lst.tail, max)

}

def get\_indexes(lst: List[Int], index: Int, max: Int): List[Int] = {

if(lst.isEmpty) return indexes

if(max == lst.head)

indexes = indexes :+ index

get\_indexes(lst.tail, index + 1, max)

}

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

println("Enter elements of the list (integers), separated by spaces:")

val list = scala.io.StdIn.readLine().split(" ").map(\_.toInt).toList

println("Entered list: " + list)

if(list.isEmpty) {

println("The list is empty")

} else {

val max = find\_max(list, list.head)

println("Max number: " + max)

println("Indexes of max numbers: " + get\_indexes(list, 0, max).mkString(", "))

}

}

}

Результат выполнения задания 3 представлен на рисунке 3.7.

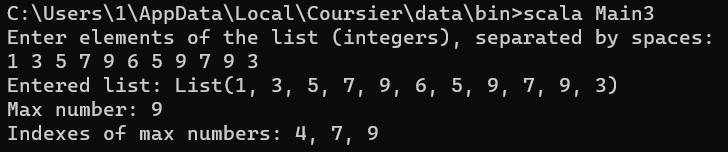


Рисунок 3.7 – Результат выполнения задания 3

**3.2.4**Написать функцию для проверки того, что список не упорядочен ни по возрастанию, ни по убыванию. Список задать самостоятельно.

Далее представлен код программы задания 4:

object Main4 {

def is\_sorted(lst: List[Int]): Boolean = {

is\_sorted\_in\_ascendeing\_order(lst, lst.head) &

is\_sorted\_in\_descending\_order(lst.reverse, lst.reverse.head)

}

def is\_sorted\_in\_ascendeing\_order(lst: List[Int], prev: Int): Boolean = {

if(lst.isEmpty)

true

else if(prev > lst.head)

false

else

is\_sorted\_in\_ascendeing\_order(lst.tail, lst.head)

}

def is\_sorted\_in\_descending\_order(lst: List[Int], prev: Int): Boolean = {

if(lst.isEmpty)

true

else if(prev < lst.head)

false

else

is\_sorted\_in\_descending\_order(lst.tail, lst.head)

}

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

println("Enter elements of the list (integers), separated by spaces:")

val list = scala.io.StdIn.readLine().split(" ").map(\_.toInt).toList

println("Entered list: " + list)

if(list.isEmpty) {

println("The list is empty")

} else {

if(is\_sorted(list))

println("List is sorted")

else

println("List is not sorted")

}

}

}

Результат выполнения задания 4 представлен на рисунке 3.8.

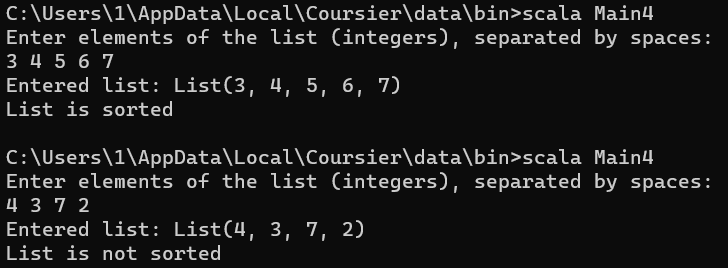


Рисунок 3.8 – Результат выполнения задания 4

**3.2.5**Написать функцию для проверки наличия трех одинаковых элементов в списке. Список задать самостоятельно. Функция возвращает значение такого элемента.

Далее представлен код программы задания 5:

var numbers = List[Int]()

object Main5 {

def find\_same\_numbers(lst: List[Int], repeation\_count: Int): List[Int] = {

if(lst.isEmpty)

return numbers

var count = count\_numbers(lst, lst.head)

if(count == repeation\_count)

numbers = numbers :+ lst.head

find\_same\_numbers(lst.tail, repeation\_count)

}

def count\_numbers(lst: List[Int], value: Int): Int = {

if(lst.isEmpty)

return 0

else if(value == lst.head)

1 + count\_numbers(lst.tail, value)

else

count\_numbers(lst.tail, value)

}

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

println("Enter elements of the list (integers), separated by spaces:")

val list = scala.io.StdIn.readLine().split(" ").map(\_.toInt).toList

println("Entered list: " + list)

if(list.isEmpty) {

println("The list is empty")

} else {

val lst\_of\_values = find\_same\_numbers(list, 3)

println("Values: " + lst\_of\_values.mkString(", "))

}

}

}

Результат выполнения задания 5 представлен на рисунке 3.8.

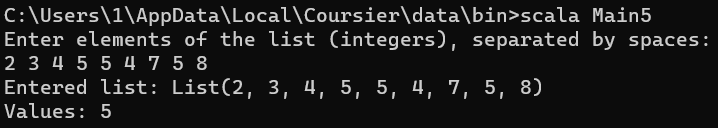


Рисунок 3.8 – Результат выполнения задания 5

**4 ВЫВОД**

Изучили работу со списками и функциями.