БГУИР

Кафедра ЭВМ

Отчёт по лабораторной работе №2

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

Выполнили:

студенты группы 150504

Горбачевский К.В.

Горбачевский М.В.

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**

Остановимся на использовании функций для работы со списками

Примеры функций для работы со списками в Scala:

1. 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 = {

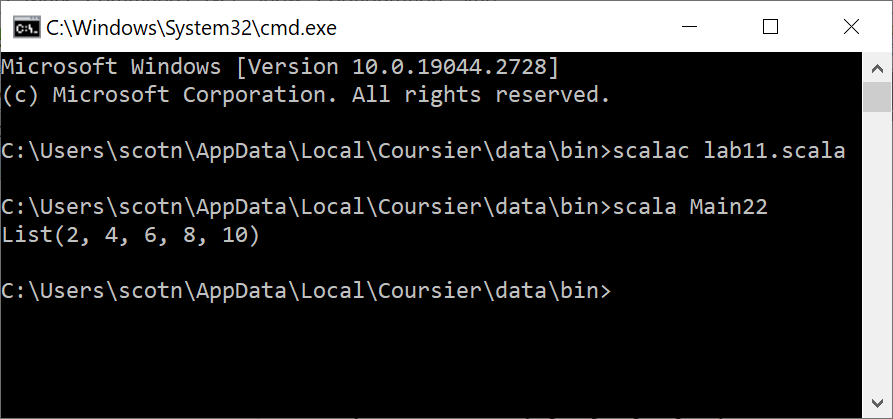
val myList = List(1, 2, 3, 4, 5)

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 = {

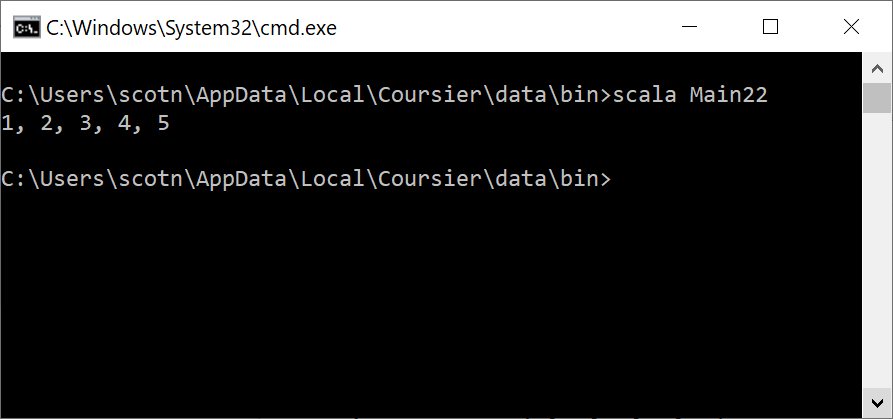
val myList = List(1, 2, 3, 4, 5)

val doubledList = myList.map(double)

println(myList.mkString(", "))

}

}



1. filter: Эта функция отбирает элементы списка, удовлетворяющие заданному предикату.

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

val evenNumbers = numbers.filter(x => x % 2 == 0) // List(2, 4)

Следующий пример показывает, как вывести четные элементы списка

object Main22 {

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

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

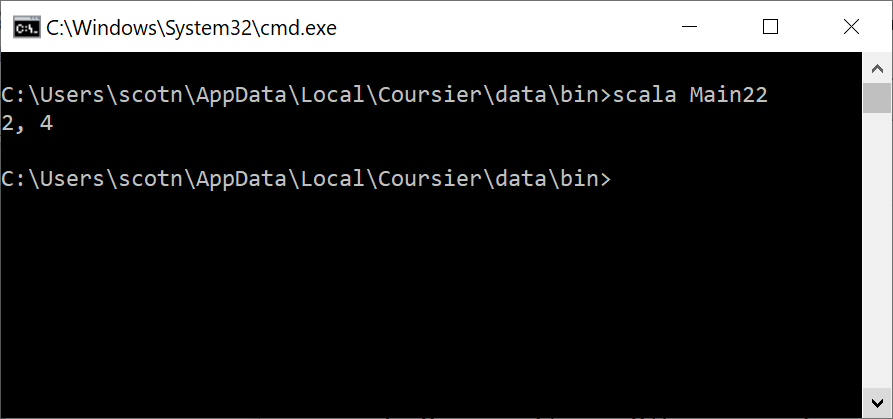
val myList = List(1, 2, 3, 4, 5)

val filteredList = myList.filter(isEven)

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

}

}



1. 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)

1. 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

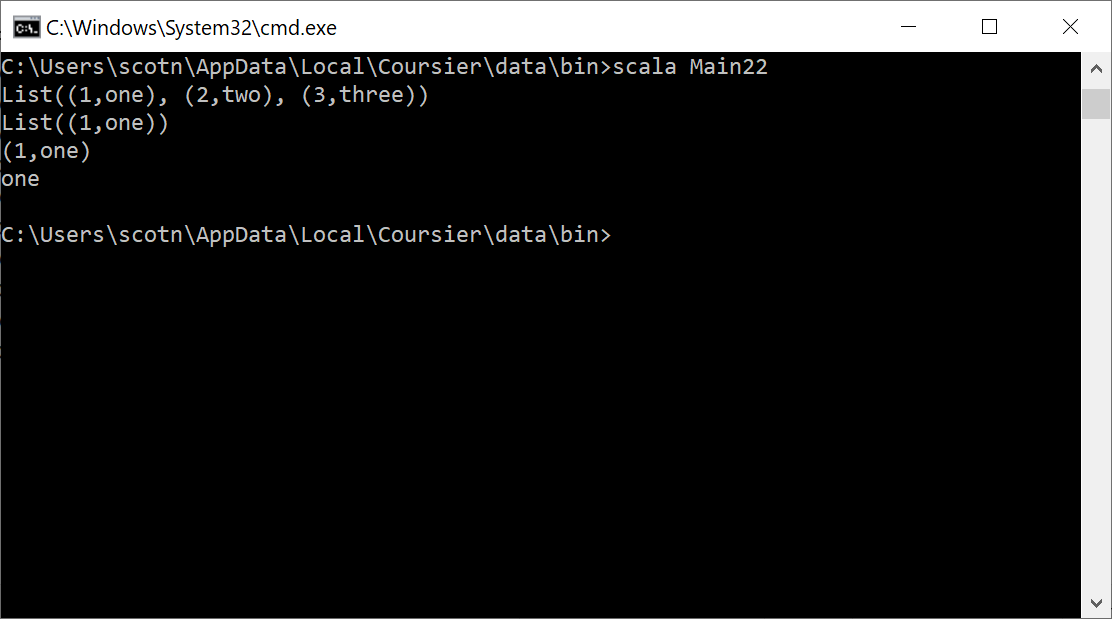
println(tup)

val secondItem = tup.\_2

println(secondItem)

}

}



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

1. 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)

}

}

Отыскание минимального элемента в списке

object Main22 {

def minList(lst: List[Int], minval: Int): Int = {

if (lst.isEmpty) minval

else if (lst.head<minval) minList(lst.tail,lst.head)

else minList(lst.tail,minval)

}

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

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

val minw = minList(myList,myList.head)

println(minw)

}

}

Или для ввода списка с клавиатуры:

import scala.io.StdIn

object Main224 {

def minList(lst: List[Int], minval: Int): Int = {

if (lst.isEmpty) minval

else if (lst.head<minval) minList(lst.tail,lst.head)

else minList(lst.tail,minval)

}

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

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

val myList = StdIn.readLine().split(" ").map(\_.toInt).toList

println("Entered list: " + myList)

if (myList.isEmpty) {

println("The list is empty")

} else

val minw = minList(myList,myList.head)

println(minw)

}

}

Отыскание элемента на позиции *i* при нумерации позиций с нуля, список содержит положительные целые числа. Если индекс выходит за пределы списка, возвращается -1:

object Main22 {

def posList(lst: List[Int], pos: Int): Int = {

if (lst.isEmpty) -1

else if (pos==0) lst.head

else {val newpos=pos-1

posList(lst.tail,newpos)}

}

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

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

val pos =3

val minw = posList(myList,pos)

println(minw)

}

}

1. reverse: Эта функция возвращает список в обратном порядке.

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

val reversed = numbers.reverse // List(5, 4, 3, 2, 1)

1. **ЗАДАНИЕ**

**Вариант 3.**

1. Написать функцию для подсчета суммы элементов списка, значение которых лежит в диапазоне [0,5]. Список задать самостоятельно.
2. Написать функцию для подсчета суммы элементов списка с номерами, содержащимися в другом списке. Список задать самостоятельно.
3. Написать функцию для отыскания индекса минимального элемента списка. Список задать самостоятельно.
4. Написать функцию для проверки того, что элементы списка не превосходят заданной величины. Список задать самостоятельно.
5. Написать функцию для подсчета числа элементов списка, которые не превосходят заданной величины. Список задать самостоятельно.

**Листинг программы:**

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

println("1 - Sum of array range [0, 5]")

println("2 - Sum of elements by indexes")

println("3 - Index of min element")

println("4 - Are all elements less than")

println("5 - Count of elements that less than")

println("0 - exit")

print("Input list: ")

var stringArrayOfInts = scala.io.StdIn.readLine().split(" ").map(item => item.toInt).toList

val list = List(stringArrayOfInts: \_\*)

var action = 1;

while(action != 0) {

print("Select action: ")

action = scala.io.StdIn.readInt()

action match

case 1 => {

val flteredList = list.filter(item => item <= 5 && item >= 0)

println("Answer: " + sum(flteredList))

}

case 2 => {

print("Input indexes: ")

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

var listForSum = indexes.map(i => list.apply(i))

println("Answer: " + sum(listForSum))

}

case 3 => {

println("Answer: " + indexOfMin(list, 0, 0))

}

case 4 => {

print("Input num: ")

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

println("Answer: " + areAllLessThan(list, num))

}

case 5 => {

print("Input num: ")

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

println("Answer: " + countOfElementsThatLessThan(list, num, 0))

}

case 0 => {

action = 0

println("Exit...")

}

}

}