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 №1

на тему

РАБОТА СО СПИСКАМИ И ФУНКЦИЯМИ

|  |  |
| --- | --- |
| Студенты группы 150503:  Проверила: | Ходосевич М.А.  Семков А. Д.  Герман Ю. О. |

Минск 2023

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

1. **Краткие теоретические сведения:**

Примеры функций для работы со списками в Scala из методички:

В первом примере выполнения перебор всех элементов с помощью Map и умножение каждого элемента на 2. Кода данной программы приведен ниже, а выполнение данного кода на рисунке 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)

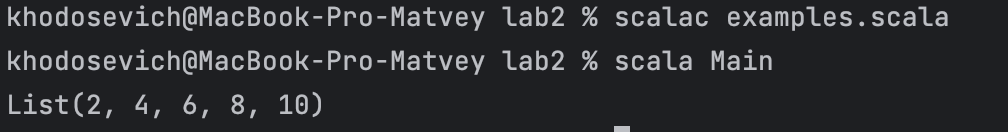


рисунок 1 - пример выполнения примера

Также продемонстрирован по элементный вывод List.

Пример выполнения данного кода приведен на рисунке 2.

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(", "))

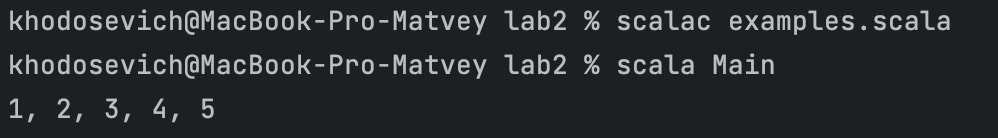


рисунок 2 - пример выполнения примера

Далее у нас рассматривается метод filter - эта функция отбирает элементы списка, удовлетворяющие заданному предикату.

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(", "))

Пример выполнения данного кода приведен на рисунке 3.

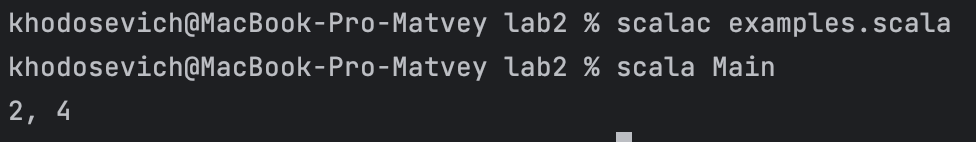


рисунок 3 - пример выполнения примера

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)

Пример выполнения данного кода приведен на рисунке 4.

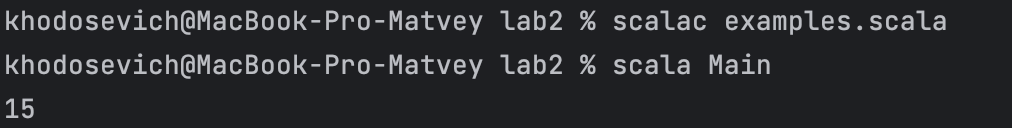


рисунок 4 - пример выполнения примера

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)

println(zipped)

Пример выполнения данного кода приведен на рисунке 5.

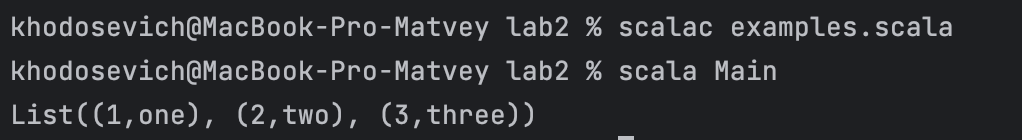


рисунок 5 - пример выполнения примера

Head and tail - эти функции возвращают голову и хвост списка соответственно.

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

val first = numbers.head

val rest = numbers.tail

Пример выполнения данного кода приведен на рисунке 6.

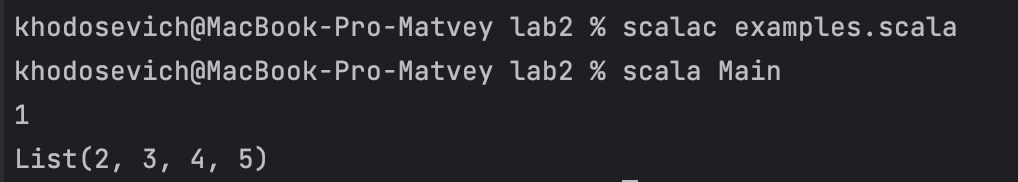


рисунок 6 - пример выполнения примера

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)

Пример выполнения данного кода приведен на рисунке 7.

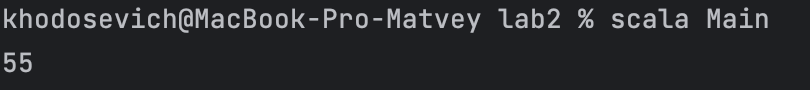


рисунок 7 - пример выполнения примера

Reverse – эта функция возвращает список в обратном порядке.

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

val reversed = numbers.reverse

Пример выполнения данного кода приведен на рисунке 8.

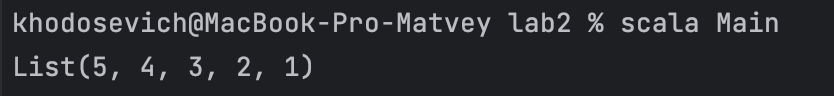


рисунок 8 - пример выполнения примера

**2. Ход работы:**

**Вариант 4.**

1. Написать функцию для подсчета суммы элементов списка, значение которых по модулю не превосходит 5. Список задать самостоятельно.

Код первой программы и результаты выполнения:

object Main1 {

def filterList(list: List[Double]) = {

if (list.isEmpty) {

throw new IllegalArgumentException("Список пуст")

}

var list1 = list.filter(x => Math.abs(x) <= 5)

val result = sumList(list1)

result

}

def sumList(list: List[Double]): Double = {

var res = 0.0;

if (list.isEmpty) 0.0

else res = list.head + sumList(list.tail)

res

}

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

var list = List[Double](1.0, 2.0, 3, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0);

println(s"Cписок: ${list.toList}")

println(s"Cумма элементов списка по модулю меньше 5: ${filterList(list)}")

}

}

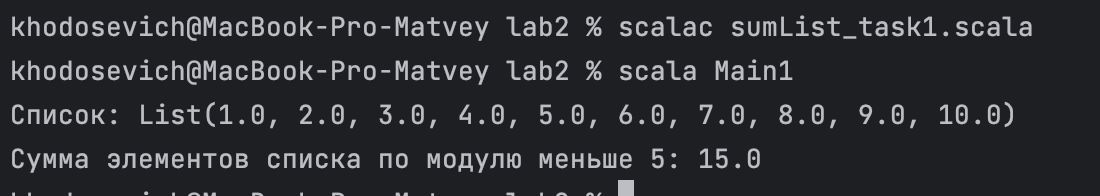


рисунок 9 - пример выполнения первой программы

1. Написать функцию для подсчета суммы каждого второго элемента списка из 10 элементов. Список задать самостоятельно.

Код второй программы и результаты выполнения:

object Main2 {

def summaEvenNumber(list: List[Double], index: Int = 0, result: Double = 0): Double = {

if (list.isEmpty) {

throw new IllegalArgumentException("Список пуст")

}

if (index >= list.length) result

else if (index % 2 == 0) summaEvenNumber(list, index + 1, result + list(index))

else summaEvenNumber(list, index + 1, result)

}

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

var list = List[Double](1.0, 2.0, 3, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0);

println(s"Cписок: ${list.toList}")

println(s"Cумма каждого второго элемента: ${summaEvenNumber(list)}")

}

}

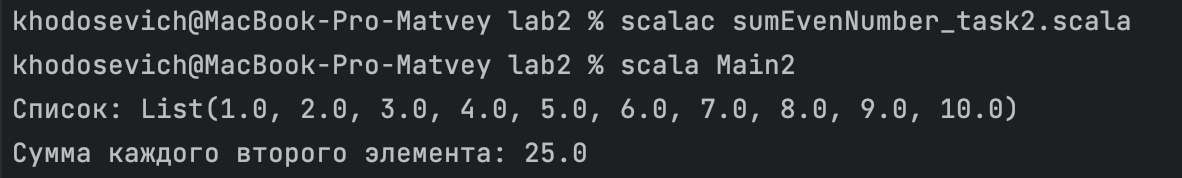


рисунок 10 - пример выполнения второй программы

1. Написать функцию для отыскания индекса элемента списка, наименее отклоняющегося от среднего значения по списку. Список задать самостоятельно.

Код третьей программы и результаты выполнения:

object Main3 {

def findClosestToAverageIndexRec(list: List[Double], currentIndex: Int = 0, closestIndex: Int = 0, minDiff: Double = Double.MaxValue, average: Double = 0): Int = {

if (currentIndex >= list.length) closestIndex

else {

val currentElement = list(currentIndex)

val diff = Math.abs(currentElement - average)

if (diff < minDiff) {

findClosestToAverageIndexRec(list, currentIndex + 1, currentIndex, diff, average)

} else if (diff == minDiff && currentElement < list(closestIndex)) {

findClosestToAverageIndexRec(list, currentIndex + 1, currentIndex, diff, average)

} else {

findClosestToAverageIndexRec(list, currentIndex + 1, closestIndex, minDiff, average)

}

}

}

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

val list = List(1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0)

var sum : Double = 0;

for(element <- 0 to list.length){

sum = sum + element

}

val avg = sum / list.length

println(s"Индекс наименее отклоняющегося от среднего: ${findClosestToAverageIndexRec(list , average = avg)}")

}

}

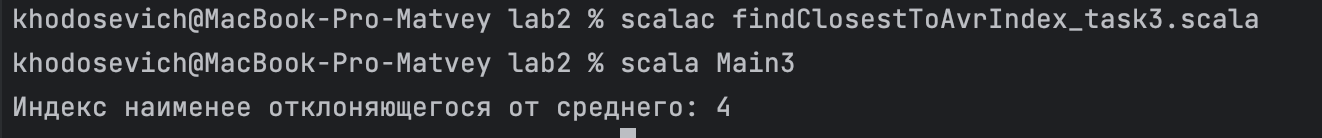


рисунок 11 - пример выполнения третьей программы

1. Написать функцию для проверки того, что список содержит квадрат одного из своих элементов. Список задать самостоятельно.

Код четвертой программы и результаты выполнения:

object Main4 {

def containsSquareRec(list: List[Double]): Boolean = {

if (list.isEmpty) {

false

}

else {

val currentElement = list.head

if (Math.sqrt(currentElement) % 1 == 0) {

true

}

else {

containsSquareRec(list.tail)

}

}

}

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

var list = List[Double](1.0, 2.0, 3, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0);

println(s"Cписок: ${list.toList}")

println(s"Cписок содержит квадрат одного из своих элементов: ${containsSquareRec(list)} ")

}

}

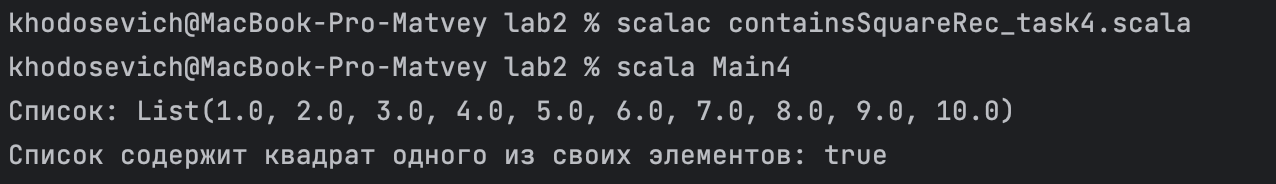


рисунок 12 - пример выполнения четвертой программы

1. Написать функцию для проверки наличия трех разных элементов в списке. Список задать самостоятельно. Функция возвращает значение Да или Нет

Код пятой программы и результаты выполнения:

object Main5 {

def hasThreeDifferentElementsRec(list: List[Double], uniqueElements: Set[Double] = Set.empty, count: Int = 0): Boolean = {

if (count >= 3) {

true

} else if (list.isEmpty) {

false

} else {

val currentElement = list.head

val updatedSet = uniqueElements + currentElement

val updatedCount = if (uniqueElements.contains(currentElement)) count else count + 1

hasThreeDifferentElementsRec(list.tail, updatedSet, updatedCount)

}

}

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

var list = List[Double](1.0, 2.0, 3, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0);

println(s"Cписок: ${list.toList}")

println(s"Проверки на наличие трех разных элементов в списке: ${hasThreeDifferentElementsRec(list)}")

}

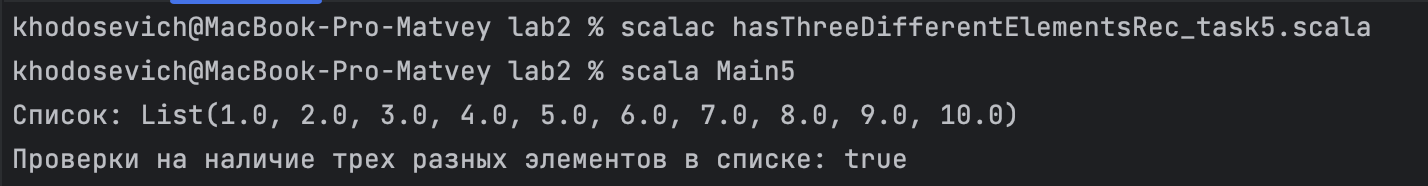


рисунок 13 - пример выполнения четвертой программы

1. **Вывод:**

В ходе выполнения данной лабораторной работы нами были проанализированы и изучены техники работы со списками и функциями в Scala.