МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное автономное бюджетное образовательное учреждение высшего образования

Севастопольский государственный университет

кафедра Информационных систем

Институт информационных технологий и управления в технических системах

курс 1 группа ИC/м-11-о

Лисянский Александр Игоревич

09.04.02 Информационные системы (уровень магистра)

**ОТЧЁТ**

о лабораторном практикуме №2

«Исследование методов обработки коллекций и функций высших порядков в функциональном программировании»

Отметка о зачёте \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Руководитель практикума

Ст.преподаватель Строганов В.А.

(должность) (подпись) (инициалы, фамилия)

Севастополь

2016

**Цель работы**

Исследовать способы реализации и обработки кортежей, массивов и коллекций. Реализовать и исследовать функции высших порядков для обработки коллекций в языке Scala

**Задание на лабораторную работу**

3.1. Переписать функцию processList из пункта 2.6. с использованием хвостовой рекурсии.

3.2. Написать функцию типа (List[Int]) => List[String], которая преобразует число в строку «Элемент под номером \*\*индекс\_элемента\*\* равен \*\*значение\_элемента\*\*».

3.3. На основе своего варианта, выданного преподавателем, и используя данные из Приложения А написать функции:

Вариант 2 (АВТ)

3.3.1. Написать функцию, которая возвращает список, содержащий год рождения всех студентов факультета АВТ, проживающих в общежитии.

3.3.2. Написать функцию, которая возвращает список, содержащий имя курс и номер комнаты всех студентов факультета АВТ, проживающих в соседних комнатах в общежитии.

3.4. Самостоятельно изучить метод groupBy и для своего факультета (в зависимости от варианта)

**Текст программы**

package main.scala

/\*\*

\* Created by user on 28.09.2016.

\*/

object lab2 {

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

def processList(lst: List[Int], func: (Int) => Int): List[Int] = {

def processListRec(\_list: List[Int],\_func: (Int)=> Int, acc: List[Int]): List[Int]={

if (\_list == Nil) acc

else processListRec(\_list.tail,\_func,\_func(\_list.head)::acc)

}

processListRec(lst,func,Nil).reverse

}

def transformIntList(lst:List[Int]): List[String] =

{

(for (i <- 0 until lst.length) yield {s"Элемент под номером $i равен "+lst(i)}).toList

}

def getAVTYear(lst: List[Student]): List[Int]={

val studentsList = lst.filter(\_.\_4 == "AVT").filter(\_.\_7 == true)

for (i<-studentsList) yield (i.\_3)

}

def getHostelNeghbors(lst: List[Student], rm: List[Room]): List[\_]={//(String, Int, Int)] = {

def fromAVTWithRoom (s: Student): Boolean = s.\_4 == "AVT" && s.\_7 == true

def getStudentByID (id: Int) : Student = lst.filter(s => s.\_1 == id).head

val roomsNeghborWithAVT = rm.map(r => (r.\_1, r.\_2, r.\_3.filter(sID => fromAVTWithRoom(getStudentByID(sID)))))

var roomNeghbors = roomsNeghborWithAVT.zip(rm.tail).filter(r=>(r.\_1.\_1 -r.\_2.\_1).abs == 1)

def makeStudentsWithRoom(r: Room): List[(String, Int, Int)] = {

val roomStudents = r.\_3

roomStudents.map(sID=> getStudentByID(sID)).map(s => (s.\_2, s.\_6, r.\_1))

}

val result = roomNeghbors.flatMap(

{

case (r1: Room , r2: Room) => makeStudentsWithRoom(r1) ::: makeStudentsWithRoom(r2)

}

).distinct

result

}

type Student = (

Int, // ID

String, // Имя

Int, // Год рождения

String, // Факультет

Char, // Пол

Int, // Курс

Boolean // Проживает ли в общежитии

)

val students: List[Student] = List(

(0, "Алёна", 1995, "FIL", 'F', 1, true),

(1, "Гриша", 1994, "AVT", 'M', 2, true),

(2, "Настя", 1993, "MTS", 'F', 3, false),

(3, "Коля", 1997, "MTS", 'M', 1, false),

(4, "Миша", 1997, "AVT", 'M', 3, true),

(5, "Оля", 1992, "FIL", 'F', 3, false),

(6, "Маша", 1991, "AVT", 'F', 5, true),

(7, "Таня", 1993, "FIL", 'M', 4, true),

(8, "Женя", 1992, "FIL", 'F', 4, true),

(9, "Света", 1989, "AVT", 'F', 3, true),

(10, "Аня", 1996, "MTS", 'F', 4, false),

(11, "Лена", 1996, "AVT", 'F', 2, true),

(12, "Сергей", 1994, "FIL", 'M', 3, false),

(13, "Влад", 1993, "FIL", 'M', 5, false),

(14, "Гена", 1996, "MTS", 'M', 1, true),

(15, "Дима", 1995, "AVT", 'M', 5, false),

(16, "Катя", 1991, "FIL", 'F', 4, false),

(17, "Артём", 1994, "MTS", 'M', 3, true),

(18, "Диана", 1995, "FIL", 'M', 4, false)

)

type Room = (

Int, // Номер комнаты

Int, // Вместимость комнаты

List[Int] // ID студентов, проживающих в комнате

)

val rooms: List[Room] = List(

(37, 3, List(0, 7, 8)),

(42, 2, List(1, 4)),

(43, 3, List(6, 9, 11)),

(54, 2, List(14, 17))

)

val list = List.range(1, 6)

println("Source list: "+list)

println("Source list \*5:"+processList(list,\_ \* 5)+"\n")

println("Source list: "+list)

println(transformIntList(list)+"\n")

println("AVT year in hostel: "+getAVTYear(students))

println(getHostelNeghbors(students,rooms))

}

}

**Выводы**

В ходе выполнения лабораторной работы были получены теоретические знания по работе с кортежами и массивами данных в языке Scala, по заданию переписана функция processList с использованием хвостовой рекурсии, использованы полученные теоретические знания на практике для выполнения заданий, указанных в варианте.