ФГАОУ ВО «Севастопольский государственный университет»

Институт информационных технологий   
и управления в технических системах

ОТЧЁТ  
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

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

по дисциплине «Парадигмы современных языков программирования»

Выполнила:  
студентка группы ИС/м-21-1-з  
Ускова Екатерина Дмитриевна

Севастополь

2022

# ЦЕЛЬ РАБОТЫ

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

# Ход работы

Переписана функция processList с использованием хвостовой рекурсии.

def processListT(lst: List[Int], func: (Int) => Int): List[Int] = {  
 def processListAcc(\_lst: List[Int], \_func: (Int) => Int, acc: List[Int]): List[Int] = {  
 if (\_lst == Nil) acc  
 else processListAcc(\_lst.tail, \_func, func(\_lst.head) :: acc)  
 }  
 processListAcc(lst, func, Nil).reverse  
}

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

def sayList(lst: List[Int]): List[String] = {  
 lst.zipWithIndex.map(el => s"Элемент под номером ${el.\_2} равен ${el.\_1}")  
}

Написана функция, которая возвращает список, содержащий имя и курс всех студентов факультета филологии, старше 93 года.

def getFil(lst: List[Student]): List[(String, Int)] = {  
 lst.filter(student => (student.\_4 == "FIL")&&(student.\_3<1993)).map(student => (student.\_2, student.\_6))  
}

Написана функция, которая возвращает список, содержащий имя, ID и номер комнаты студентов факультета филологии, проживающих в одной комнате.

def getFilNeighbors(lst: List[Student], rms: List[Room]): List[(String, Int, Int)] = {  
 val dormFils = lst.filter(student => (student.\_4 == "FIL") && student.\_7)  
 val dormFilsID = dormFils.map(student => student.\_1)  
 var roomNum: Int = 0  
 for (room <- rms)  
 if (room.\_3 == dormFilsID) {  
 roomNum = room.\_1  
 }  
 rms.filter(\_.\_1 == roomNum).flatMap(\_.\_3.flatMap(id => students.filter(\_.\_1 == id).map(student => (student.\_2, student.\_1, roomNum))))  
 }

Самостоятельно изучен метод groupBy.

# Вывод

Выполняя эту работу, я исследовала способы реализации и обработки кортежей, массивов и коллекций. Реализовала и исследовала функции высших порядков для обработки коллекций в языке Scala.

# Приложение А

Листинг программного кода

package main.scala  
  
import scala.annotation.tailrec  
  
object Application {  
 def processList(lst: List[Int], func: (Int) => Int): List[Int] = {  
 if (lst == Nil) Nil  
 else func(lst.head) :: processList(lst.tail, func)  
 }  
  
 def processListT(lst: List[Int], func: (Int) => Int): List[Int] = {  
 def processListAcc(\_lst: List[Int], \_func: (Int) => Int, acc: List[Int]): List[Int] = {  
 if (\_lst == Nil) acc  
 else processListAcc(\_lst.tail, \_func, func(\_lst.head) :: acc)  
 }  
 processListAcc(lst, func, Nil).reverse  
 }  
  
 def sayList(lst: List[Int]): List[String] = {  
 lst.zipWithIndex.map(el => s"Элемент под номером ${el.\_2} равен ${el.\_1}")  
 }  
  
  
  
 def main (args: Array[String]): Unit = {  
 val lst = List(1, 3, 6, 9)  
// println(processList(lst, x => 2 \* x))  
// println(processListT(lst, x => 2 \* x))  
// println(sayList(lst))  
  
 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))  
 )  
  
 def getFil(lst: List[Student]): List[(String, Int)] = {  
 lst.filter(student => (student.\_4 == "FIL")&&(student.\_3 < 1993)).map(student => (student.\_2, student.\_6))  
 }  
  
// println(getFil(students))  
  
 def getFilNeighbors(lst: List[Student], rms: List[Room]): List[(String, Int, Int)] = {  
 val dormFils = lst.filter(student => (student.\_4 == "FIL") && student.\_7)  
 val dormFilsID = dormFils.map(student => student.\_1)  
 var roomNum: Int = 0  
 for (room <- rms)  
 if (room.\_3 == dormFilsID) {  
 roomNum = room.\_1  
 }  
 rms.filter(\_.\_1 == roomNum).flatMap(\_.\_3.flatMap(id => students.filter(\_.\_1 == id).map(student => (student.\_2, student.\_1, roomNum))))  
  
// val dormFilsRoomNum = rms.map(room => room.\_3.map(id => students.filter(\_.\_1 == id).map(st => (st.\_1, room.\_1))))  
//  
// var  
// for (room <- rms; st <- students)  
// if (room.\_3.contains(st.\_1))  
 }  
  
 println(getFilNeighbors(students, rooms))  
 }  
}