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

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 1 (филология):

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

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

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

# 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  
  
object Application {  
 // Функция возвращает новый список, в котором каждый элемент является  
 // суммой предыдущего элемента нового списка и текущего элемента входного списка  
 def cumsumList(lst: List[Int], sum: Int = 0): List[Int] = {  
 if (lst == Nil) Nil  
 else sum + lst.head :: cumsumList(lst.tail, sum + lst.head)  
 }  
  
 def cumsumListT(lst: List[Int]): List[Int] = {  
 def cumsumListAcc(\_lst: List[Int], sum: Int, acc: List[Int]): List[Int] = {  
 if (\_lst == Nil) acc  
 else cumsumListAcc(\_lst.tail, sum + \_lst.head, sum + \_lst.head :: acc)  
 }  
  
 cumsumListAcc(lst, 0, Nil).reverse  
 }  
  
 def main (args: Array[String]): Unit = {  
 scala.util.Random.setSeed(42)  
 var cnt = 0  
 var llst :List[Int] = List()  
 while(cnt<10000) {  
 llst = scala.util.Random.nextInt(10) :: llst  
 cnt += 1  
 }  
  
 //println(cumsumList(llst))  
 println(cumsumListT(llst))  
 }  
}