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 №4

«Исследование способов реализации алгоритмов сопоставления с образцом в языке Scala»

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

(дата)

Руководитель практикума

Ст.преподаватель Строганов В.А.

(должность) (подпись) (инициалы, фамилия)

Севастополь

2016

**Цель работы**

Исследовать особенности реализации алгоритмов сопоставления с образцом в языке Scala. Получить навыки использования case-классов и класса Option в функциональном программировании.

**1. Задание на лабораторную работу**

Все задания необходимо выполнять с помощью сопоставления с образцом (match) и не использовать условные выражения (if).

1.1.Написать функцию типа (List[(Int, Int)]) => List[Option[Double]], которая принимает на вход список из пар двух целочисленных значений и возвращает список результатов деления первого числа пары на второе в виде Option(Some[Double], если второй элемент пары не равен нулю, или None, если второй элемент пары равен нулю).

1.2. Написать функцию типа (List[Option[Double]]) => List[String], которая принимает на вход список Option’ов типа Double (результаты работы функции из п.1) и преобразует его в список строк по следующему правилу: значения Some преобразуются в строку «Результат деления = \_\_число\_из\_Some\_\_», а значения None преобразуются в строку «Деление на ноль невозможно».

1.3. Реализовать задание из п.1, но вместо пар использовать case-класс.

1.4. На основе case-классов реализовать структуру дерева вычислений. Написать функцию, которая это дерево преобразует в строку обратной польской нотации.

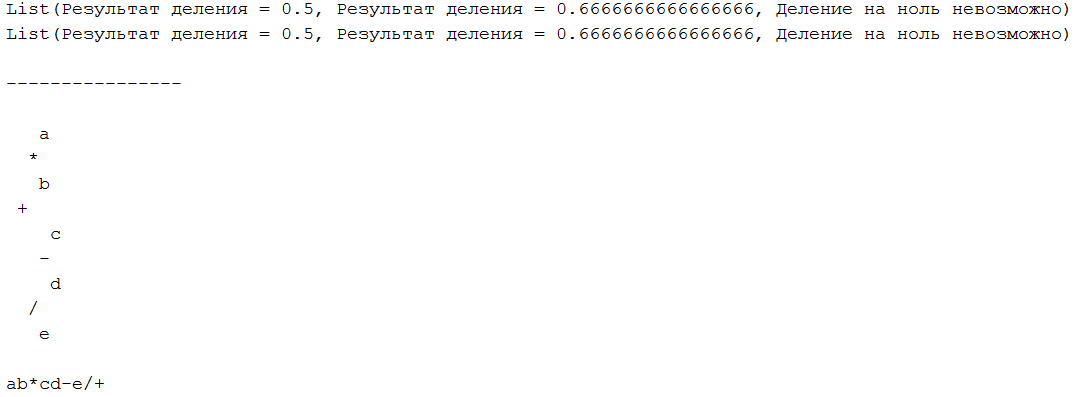
**2. Текст программы**

Основная программа

**package** main.scala  
  
*/\*\*  
 \* Created by user on 25.10.2016.  
 \*/***object** lab4 {  
 **abstract class** Pair  
 **case class** ZeroElement(dataFirst: Int, dataSecond: Int) **extends** Pair  
 **case class** Element(dataFirst: Int, dataSecond: Int) **extends** Pair  
  
 **abstract class** NodeElement  
 **case class** Operation(data: Int) **extends** NodeElement  
 **case class** Variable(data: Char) **extends** NodeElement  
  
 **abstract class** Tree  
 **case class** Leaf(data: NodeElement) **extends** Tree  
 **case class** Node(data: NodeElement, left: Tree, right: Tree) **extends** Tree  
  
 **def** printElement(el: NodeElement): Any = {  
 el **match** {  
 **case** *Operation*(o) => o **match** {  
 **case** 1 => **'+'  
 case** 2 => **'-'  
 case** 3 => **'\*'  
 case** 4 => **'/'** }  
 **case** *Variable*(v) => v  
 }  
 }  
  
 **def** printTree(tree: Tree, lvl: Int): Unit = {  
 tree **match** {  
 **case** *Node*(d, l, r) =>  
 *printTree*(l, lvl + 1)  
 *println*(" " \* lvl + *printElement*(d))  
 *printTree*(r, lvl + 1)

**case** *Leaf*(d) => *println*(**" "** \* lvl + *printElement*(d))  
 }  
 }  
  
 **def** printTreeElement(tree: Tree): Unit = {  
 tree **match** {  
 **case** *Node*(d, l, r) =>  
 *printTreeElement*(l)  
 *printTreeElement*(r)  
 *print*(*printElement*(d))  
 **case** *Leaf*(d) => *print*(*printElement*(d))  
 }  
 }  
  
  
 **def** listDivisionWithCase(lst: List[(Pair)]) : List[Option[Double]] = {  
 **if** (lst == *Nil*)  
 **return** *Nil* **val** res = lst.head **match** {  
 **case** *Element*(dF, dS) => *Some*(dF \* 1.0/ dS)  
 **case** *ZeroElement*(dF, dS)=> None  
 }  
 **val** result = res :: *listDivisionWithCase*(lst.tail)  
 result  
 }  
  
 **def** listDivision(lst: List[(Int,Int)]) : List[Option[Double]] = {  
 **if** (lst == *Nil*)  
 **return** *Nil* **val** res = lst.head.\_2 **match** {  
 **case** 0 => None  
 **case** \_ => *Some*(lst.head.\_1 \* 1.0/ lst.head.\_2)  
 }  
 **val** result = res :: *listDivision*(lst.tail)  
 result  
 }  
  
 **def** printDivision(lst:List[Option[Double]]) : List[String] = {  
 **if** (lst == *Nil*)  
 **return** *Nil* **val** res = lst.head **match** {  
 **case** *Some*(double) => **"Результат деления = "**+double  
 **case** None => **"Деление на ноль невозможно"** }  
 **val** result = res :: *printDivision*(lst.tail)  
 result  
 }  
  
 **def** main(args: Array[String]): Unit = {  
 **val** list = *List*((1,2),(2,3),(1,0))  
 *println*(*printDivision*(*listDivision*(list)))  
 **val** pairList: List[Pair] = *List*(**new** Element(1,2),**new** Element (2,3), **new** ZeroElement(1,0))  
 *println*(*printDivision*(*listDivisionWithCase*(pairList)))  
  
 *println*(**"\n----------------\n"**)  
  
 **val** tree = **new** Node(  
 **new** Operation(1),  
 **new** Node(  
 **new** Operation(3),  
 **new** Leaf(  
 **new** Variable(**'a'**)  
 ),  
 **new** Leaf(  
 **new** Variable(**'b'**)  
 )  
 ),  
 **new** Node(  
 **new** Operation(4),  
 **new** Node(  
 **new** Operation(2),  
 **new** Leaf(  
 **new** Variable(**'c'**)  
 ),  
 **new** Leaf(  
 **new** Variable(**'d'**)  
 )  
 ),  
 **new** Leaf(  
 **new** Variable(**'e'**)  
 )  
 )  
 )  
 *printTree*(tree,1)  
 *println*()  
 *printTreeElement*(tree)  
 }  
}

**3. Результаты работы программы**



Результат работы программы

**Выводы**

В ходе выполнения лабораторной работы были получены практические навыки по особенностям реализации алгоритмов сопоставления с образцом в языке Scala. получены навыки использования case-классов и класса Option в функциональном программировании, использованы полученные теоретические знания на практике для выполнения заданий, указанных в варианте.