МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное  
образовательное учреждение высшего образования  
«Самарский национальный исследовательский университет  
имени академика С.П. Королева»  
(Самарский университет)   
  
  
Факультет информатики  
Кафедра программных систем  
  
Дисциплина  
**Логическое программирование  
  
  
  
ОТЧЕТ**по лабораторной работе №6

Тема: Scala. Коллекции. Хвостовая рекурсия.

Студент: Лапин К.С.   
Группа: 6303-020302D   
  
Преподаватель: Лобанков А.А.  
  
Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Самара 2024

**Теория**

1. Коллекции.

Scala предоставляет широкий спектр коллекций для работы с данными, в том числе списки (List), массивы (Array), наборы (Set), карты (Map) и другие.

Списки (List) в Scala являются неизменяемыми и представляют упорядоченную коллекцию элементов одного типа. Новые элементы можно добавлять или удалять, создавая новые списки.

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

1. Case-классы.

Case-классы в Scala являются особыми классами, которые предназначены для представления данных.

Они автоматически предоставляют методы доступа к полям, методы toString, equals, hashCode, что делает их удобными для работы с данными.

Пример: case class Person(name: String, age: Int)

1. Работа со списками на Scala используя хвостовую рекурсию.

Хвостовая рекурсия в Scala позволяет оптимизировать рекурсивные вызовы функций, избегая переполнения стека.

При использовании хвостовой рекурсии последняя операция в функции является вызовом самой себя.

Пример:

def sumListTailRecursive(list: List[Int], acc: Int = 0): Int = list match {

case Nil => acc

case head :: tail => sumListTailRecursive(tail, acc + head)

}

1. Условные конструкции if-else.

Условные конструкции if-else в Scala используются для выполнения кода в зависимости от некоторого логического выражения.

Они могут быть вложенными и также могут использоваться в качестве выражений.

Пример: val result = if (x > 10) "Больше 10" else "Меньше или равно 10"

1. Конструкция match-case.

Конструкция match-case в Scala представляет собой альтернативу условной конструкции if-else и позволяет сопоставлять значение с различными шаблонами.

Она может использоваться для управления потоком выполнения программы.

Пример:

val day = 2

val dayType = day match {

case 1 => "Понедельник"

case 2 => "Вторник"

case \_ => "Другой день"

}

1. val и var - переменные:

В Scala val используется для создания неизменяемых (immutable) переменных, которые нельзя переопределять после инициализации.

var используется для создания изменяемых (mutable) переменных, значения которых можно изменять.

Пример: val x = 10 или var y = "Hello"

1. Класс object:

Ключевое слово object в Scala используется для определения объекта (singleton object), который может содержать методы и поля, аналогично статическим членам в Java.

Объекты могут использоваться для организации кода и предоставления глобальных функций или констант.

Пример: object Utils { def formatName(name: String): String = name.toUpperCase }

1. Функции def:

Ключевое слово def используется для определения функций в Scala.

Функции могут быть определены как методы у классов, функциональные значения или быть частью объектов.

Пример: def sum(a: Int, b: Int): Int = a + b

1. Вывод в консоль с помощью println():

Функция println() в Scala используется для вывода данных в консоль.

Она может выводить различные типы данных, такие как строки, числа, списки и другие.

Пример: val message = "Hello, World" println(message)

Эти конструкции помогают писать чистый, модульный и выразительный код на языке программирования Scala. Они также способствуют функциональному программированию и принципам иммутабельности данных.

**Задание**

**Цель и постановка задания**

В процессе выполнения заданий ознакомиться с case-классами и работой со списками на Scala, используя хвостовую рекурсию.

**Задание 1**

Придумайте задание и реализуйте программу на Scala, используя коллекции и хвостовую рекурсию. В качестве предметной области была выбрана предметная область из лабораторной работы номер 3.

**Программа**

Приведенный код содержит описание трех case class (классов) для представления данных о соревнованиях, результатах команд и о самих командах:

* Competition представляет информацию о соревнованиях, такую как ранг, вид спорта, год, страна и список команд участников;
* Result представляет результаты, которые команда достигла против оппонента, включая страну оппонента и результат;
* Team содержит информацию о команде, включая название, страну и список результатов против различных оппонентов.

Далее, в object main extends App, определены три различных комплекта соревнований в виде коллекции competitions. Каждое соревнование содержит информацию о названии, виде спорта, годе проведения, стране, а также список команд с их результатами против других команд.

Затем в коде определены две функции:

* findMaxTeamsYears(rank: String, competitions: List[Competition]): эта функция ищет года, в которые участвовало максимальное количество команд заданного ранга соревнований. Она рекурсивно итерируется по списку соревнований, находит максимальное количество команд, а затем возвращает список годов с максимальным количеством команд;
* findSportsByTeam(teamName: String, competitions: List[Competition]): данная функция находит виды спорта, в которых выступает заданная команда. Она также рекурсивно проходит по списку соревнований, ищет команду по названию, и возвращает список видов спорта, в которых команда участвовала;
* findOlympicTeams(sport: String, competitions: List[Competition]): Эта функция находит все команды, которые участвовали в Олимпийских играх по определенному виду спорта. Она итерирует по списку соревнований, и если соревнование относится к Олимпийским играм и имеет заданный вид спорта, добавляет имена команд в результат;
* findTeamsByYear(year: Int, competitions: List[Competition]): Данная функция находит все команды, участвовавшие в соревнованиях в заданном году. Она также итерирует по списку соревнований, добавляя имена команд, которые участвовали в указанном году;
* findWinningTeamsByCountry(country: String, competitions: List[Competition]): Эта функция ищет все команды из определенной страны, которые имели выигрыши. Она фильтрует команды по стране и затем по результатам, чтобы определить победы, после чего возвращает имена команд с победами.

Функции используют рекурсию для обхода списков Competitions, Teams, Results.

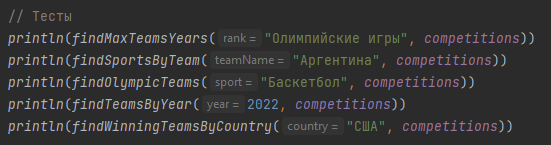
Наконец, в конце кода представлены вызовы этих функций с различными аргументами для тестирования функциональности:

* findMaxTeamsYears("Олимпийские игры", competitions): Поиск годов с максимальным количеством команд на Олимпийских играх;
* findSportsByTeam("Аргентина", competitions): Нахождение видов спорта, в которых выступала команда "Аргентина";
* findOlympicTeams("Баскетбол", competitions): Поиск всех команд, участвовавших в олимпийских соревнованиях по баскетболу;
* findTeamsByYear(2022, competitions): Поиск всех команд, участвовавших в соревнованиях в 2022 году;
* findWinningTeamsByCountry("США", competitions): Поиск всех команд из страны "США" с победами.

Таким образом, программа предоставляет функционал для анализа данных о соревнованиях, командах и их результатам, позволяя выполнять различные запросы и фильтрацию данных.

**Результат работы программы**

На рисунке 1 представлен код вызова функций для выполнения запросов с различными параметрами.

  
Рисунок 1 – Код выполнения запросов на поиск по заданиям

На рисунке 2 представлены результаты выполненных запросов выше,

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание  
Рисунок 2 – Результат выполнения запросов на поиск по заданиям

**Листинг программы**

// Case class для представления соревнований  
case class Competition(rank: String, sport: String, year: Int, country: String, teams: List[Team])

// Case class для представления соревнований  
case class Competition(rank: String, sport: String, year: Int, country: String, teams: List[Team])

// Case class для представления команд  
case class Team(name: String, country: String, results: List[Result])

import domain.{Competition, Result, Team}  
  
import scala.annotation.tailrec  
  
/\*  
 Предметная область – спортивные соревнования.  
 соревнование может быть описано структурой: ранг соревнований, вид спорта,  
 год проведения, страна проведения, список команд - участников. Команды -  
 участники могут быть описаны следующей структурой: название команды,  
 страна, результаты соревнований.  
 Результаты соревнований могут быть описаны списком структур: название команды – соперника,  
 страна, тип результата (выигрыш, проигрыш, ничья).  
\*/  
  
object main extends App {  
 // Коллекция соревнований  
 val *competitions* = List(  
 Competition("Олимпийские игры", "Баскетбол", 2016, "Бразилия", List(  
 Team("США", "США", List(  
 Result("Испания", "Испания", "Победа"),  
 Result("Аргентина", "Аргентина", "Победа"),  
 Result("Франция", "Франция", "Победа"),  
 )),  
 Team("Испания", "Испания", List(  
 Result("США", "США", "Поражение"),  
 Result("Аргентина", "Аргентина", "Победа"),  
 Result("Франция", "Франция", "Ничья"),  
 )),  
 Team("Аргентина", "Аргентина", List(  
 Result("США", "США", "Поражение"),  
 Result("Испания", "Испания", "Поражение"),  
 Result("Франция", "Франция", "Победа"),  
 )),  
 )),  
 Competition("Олимпийские игры", "Лыжный спорт", 2020, "Бразилия", List(  
 Team("США", "США", List(  
 Result("Испания", "Испания", "Победа"),  
 Result("Аргентина", "Аргентина", "Победа"),  
 Result("Франция", "Франция", "Победа"),  
 )),  
 Team("Испания", "Испания", List(  
 Result("США", "США", "Поражение"),  
 Result("Аргентина", "Аргентина", "Победа"),  
 Result("Франция", "Франция", "Ничья"),  
 )),  
 Team("Аргентина", "Аргентина", List(  
 Result("США", "США", "Поражение"),  
 Result("Испания", "Испания", "Поражение"),  
 Result("Франция", "Франция", "Победа"),  
 )),  
 )),  
 Competition("Чемпионат мира", "Футбол", 2022, "Катар", List(  
 Team("Аргентина", "Аргентина", List(  
 Result("Саудовская Аравия", "Саудовская Аравия", "Победа"),  
 Result("Мексика", "Мексика", "Ничья"),  
 Result("Польша", "Польша", "Победа"),  
 )),  
 Team("Бразилия", "Бразилия", List(  
 Result("Сербия", "Сербия", "Победа"),  
 Result("Швейцария", "Швейцария", "Победа"),  
 Result("Камерун", "Камерун", "Ничья"),  
 )),  
 )),  
 )  
  
 // Функция для поиска годов, в которые участвовало максимальное число команд в заданном ранге соревнований  
 def findMaxTeamsYears(rank: String, competitions: List[Competition]): List[Int] = {  
 // Функция для поиска максимального числа команд в заданном ранге соревнований  
 def findMaxTeams(competitions: List[Competition]): Int = competitions match {  
 case Nil => 0  
 case head :: tail =>  
 if (head.rank == rank) math.*max*(head.teams.size, findMaxTeams(tail))  
 else findMaxTeams(tail)  
 }  
  
 // Максимальное число команд  
 val maxTeams = findMaxTeams(competitions)  
  
 // Функция для поиска годов, когда участвовало максимальное число команд  
 @tailrec  
 def helper(maxTeams: Int, competitions: List[Competition], result: List[Int]): List[Int] = competitions match {  
 case Nil => result  
 case head :: tail =>  
 if (head.rank == rank && head.teams.size == maxTeams) helper(maxTeams, tail, head.year :: result)  
 else helper(maxTeams, tail, result)  
 }  
 helper(maxTeams, competitions, Nil).distinct  
 }  
  
 // Найти вид спорта, в котором выступает заданная команда;  
 def findSportsByTeam(teamName: String, competitions: List[Competition]): List[String] = {  
 @tailrec  
 def helper(competitions: List[Competition], acc: List[String]): List[String] = competitions match {  
 case Nil => acc  
 case head :: tail => head.teams.find(\_.name == teamName) match {  
 case Some(\_) => helper(tail, head.sport :: acc)  
 case None => helper(tail, acc)  
 }  
 }  
 helper(competitions, List.*empty*).distinct // Инициализируем накапливаемый результат пустым списком  
 }  
  
 // Функция для поиска всех команд, которые участвовали в Олимпийских играх по определенному виду спорта  
 def findOlympicTeams(sport: String, competitions: List[Competition]): List[String] = {  
 @tailrec  
 def helper(result: List[String], competitions: List[Competition]): List[String] = competitions match {  
 case Nil => result  
 case head :: tail =>  
 if (head.rank == "Олимпийские игры" && head.sport == sport) helper(result ++ head.teams.map(\_.name), tail)  
 else helper(result, tail)  
 }  
  
 helper(List.*empty*, competitions).distinct  
 }  
  
 // Функция для поиска всех команд, участвовавших в соревнованиях в заданном году  
 def findTeamsByYear(year: Int, competitions: List[Competition]): List[String] = {  
 @tailrec  
 def helper(result: List[String], competitions: List[Competition]): List[String] = competitions match {  
 case Nil => result  
 case head :: tail =>  
 if (head.year == year) helper(result ++ head.teams.map(\_.name), tail)  
 else helper(result, tail)  
 }  
  
 helper(List.*empty*, competitions).distinct  
 }  
  
 // Функция для поиска всех команд определенной страны, у которых были выигрыши  
 def findWinningTeamsByCountry(country: String, competitions: List[Competition]): List[String] = {  
  
 @tailrec  
 def helper(result: List[String], competitions: List[Competition]): List[String] = competitions match {  
 case Nil => result  
 case head :: tail =>  
 val winningTeams = head.teams.filter(\_.country == country).filter(\_.results.exists(\_.outcome == "Победа"))  
 helper(result ++ winningTeams.map(\_.name), tail) // Хвостовой вызов  
 }  
  
 helper(List.*empty*, competitions).distinct  
 }  
  
 // Тесты  
 *println*(*findMaxTeamsYears*("Олимпийские игры", *competitions*))  
 *println*(*findSportsByTeam*("Аргентина", *competitions*))  
 *println*(*findOlympicTeams*("Баскетбол", *competitions*))  
 *println*(*findTeamsByYear*(2022, *competitions*))  
 *println*(*findWinningTeamsByCountry*("США", *competitions*))  
}