**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение**

**Высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра Математической кибернетики и информационных технологий

**Отчет по лабораторной работе**

По дисциплине «Функциональное программирование»

На тему:

«Основы scala»

Выполнил:

студент группы БВТ1801

Задоркин Максим Александрович

Москва 2020

**Цель работы:** изучение основ scala

**Задание:**

Решение задач из приложенных файлов

**Ход работы:**

Работа с функциями:

package exercise1

*/\*\* Напишите отдельные функции, решающие поставленную задачу.*

*\**

*\* Синтаксис:*

*\* // метод*

*\* def myFunction(param0: Int, param1: String): Double = // тело*

*\**

*\* // значение*

*\* val myFunction: (Int, String) => Double (param0, param1) => // тело*

*\*/*

object Functions {

*/\* a) Напишите функцию, которая рассчитывает площадь окружности*

*\* r^2 \* Math.PI*

*\*/*

def circleArea(val r: Double): Double = r \* r \* Math.PI

*// примените вашу функцию из пункта (a) здесь, не изменяя сигнатуру*

def testCircle(r: Double): Double = circleArea(r)

*/\* b) Напишите карированную функцию которая рассчитывает площадь прямоугольника a \* b.*

*\*/*

def rectangleArea(a: Double)(b: Double): Double = a \* b

*// примените вашу функцию из пукта (b) здесь, не изменяя сигнатуру*

def testRectangleCurried(a: Double, b: Double): Double = rectangleArea(a)(b)

*// c) Напишите не карированную функцию для расчета площади прямоугольника.*

def rectangleArea(a: Double, b: Double): Double = a \* b

*// примените вашу функцию из пункта (c) здесь, не изменяя сигнатуру*

def testRectangleUc(a: Double, b: Double): Double = rectangleArea(a, b)

}

Работа с HiOrder

package exercise1

*/\*\* Напишите ваши решения в виде функций. \*/*

object HigherOrder {

val plus: (Int, Int) => Int = \_ + \_

val multiply: (Int, Int) => Int = \_ \* \_

*/\* a) Напишите функцию, которая принимает `f: (Int, Int) => Int`, параменты `a` и `b`*

*\* и коэффициент умножения `n` и возвращает n \* f(a, b). Назовите `nTimes`.*

*\*/*

def nTimes(f: (Int, Int) => Int, a: Int, b: Int, n: Int): Int = n \* f(a, b)

*// примените вашу функцию (a) здесь, не изменяйте сигнатуру*

def testNTimes(f: (Int, Int) => Int, a: Int, b: Int, n: Int): Int = nTimes(f, a, b, n)

*/\* b) Напишите анонимную функцию, функцию без идентификатора ((a, b) => ???) для `nTimes` которая*

*\* выполняет следующее:*

*\* if (a > b) a else b*

*\*/*

def a = (a: Int, b: Int) => **if** (a > b) a **else** b

def testAnonymousNTimes(a: Int, b: Int, n: Int): Int = a(a, b)

}

Работа с Patterns

package exercise1

*/\*\* Напишите решение в виде функции.*

*\**

*\* Синтаксис:*

*\* val a: Int = ???*

*\**

*\* a match {*

*\* case 0 => true*

*\* case \_ => false*

*\* }*

*\*/*

object PatternMatching {

**sealed** trait Hand

case object Rock extends Hand

case object Paper extends Hand

case object Scissor extends Hand

**sealed** trait Result

case object Win extends Result

case object Lose extends Result

case object Draw extends Result

**sealed** trait Food

case object Meat extends Food

case object Vegetables extends Food

case object Plants extends Food

**sealed** trait Animal {

val name: String

val food: Food

}

case class Mammal(name: String, food: Food, weight: Int) extends Animal

case class Fish(name: String, food: Food) extends Animal

case class Bird(name: String, food: Food) extends Animal

*/\* a) Напишите функцию, которая ставит в соответствие числу строку слудеющим образом:*

*\* Если:*

*\* 1 => "it is one"*

*\* 2 => "it is two"*

*\* 3 => "it is three"*

*\* иначе => "what's that"*

*\*/*

def a(value: Int): String = {

value **match** {

**case** 1 => "it is one"

**case** 2 => "it is two"

**case** 3 => "it is three"

**case** \_ => "what's that"

}

}

*// примените вашу функцию из пункта (a) здесь, не изменяя сигнатуру*

def testIntToString(value: Int): String = a(value.toString)

*/\* b) Напишите функцию которая возвращает true если переменная `value` принимает значение:*

*\* "max" или "Max*

*\* "moritz" или "Moritz"*

*\*/*

def IsMaxAndMoritz(value: String): Boolean = {

value **match** {

**case** "max" => **true**

**case** "Max" => **true**

**case** "moritz" => **true**

**case** "Moritz" => **true**

**case** \_ => **false**

}

}

*// примените функции из пункта (b) здесь, не изменяя сигнатуру*

def testIsMaxAndMoritz(value: String): Boolean = IsMaxAndMoritz("asd")

*// c) Напишите функцию проверки является ли `value` четным*

def isEven(value: Int): Boolean = value % 2 == 0

*// примените функции из пункта (c) здесь, не изменяя сигнатуру*

def testIsEven(value: Int): Boolean = isEven(3)

*/\* d) Напишите функцию, моделирующую игру в Камень ножницы бумага*

*\* 1. камень побеждает ножницы*

*\* 2. ножницы побеждают бумагу*

*\* 3. бумага побеждает камень*

*\* Выиграет ли игрок `a`?*

*\*/*

def winsA(a: Hand, b: Hand): Result = {

a **match** {

**case** "rock" => b == "paper"

**case** "paper" => b == "scissors"

**case** "scissors" => b == "rock"

}

}

*// примените вашу функцию из пункта (d) здесь, не изменяя сигнатуру*

def testWinsA(a: Hand, b: Hand): Result = Draw

*// Примечание: используйте определение Animals*

*// e) Верните вес (weight: Int) объекта Mammal, иначе верните -1.*

def retWeight(animal: Animal): Int = **if** animal.weight >= 0 animal.weight **else** -1

*// примените функцию из пункта (e) здесь, не изменяйте сигнатуру*

def testExtractMammalWeight(animal: Animal): Int = retWeight(animal)

*// f) Измените поле еда объектов классов Fishes и Birds на Plants, класс Mammels оставьте неизмененным.*

def change(animal: Animal): Animal = {

Animal.fish.food = "plants"

Animal.bird.food = "plants"

Animal

}

*// примените функцию из пункта (f) здесь, не изменяйте сигнатуру*

def testUpdateFood(animal: Animal): Animal = change(animal)

}

Работа с classes

*// 1 часть задания*

class Animal(name: String, species: String, food: String) {

def eats(food: String): Boolean = Animal.animals(name)(1) == food

}

object Animal {

val animals = Map("cat" -> Array[String]("mammal", "meat"),

"parrot" -> Array[String]("bird", "vegetables"),

"goldfish" -> Array[String]("fish", "plants"))

}

*//2 часть задания (начиная с пункта d)*

trait Animal {

case object mammal extends Animal

case object bird extends Animal

case object fish extends Animal

}

object Animal {

def knownAnimal(name: String): Boolean = {

name **match** {

**case** "cat" => **true**

**case** "parrot" => **true**

**case** "goldfish" => **true**

**case** \_ => **false**

}

}

def apply(name: String): Option[Animal] = Some.apply(Animal.this)

}

class Animal(name: String, food: String) {

trait Food {

case class Meat

case class Vegetables

case class Plants

}

object Food {

def apply(food: String): Option[Food] = Some.apply(Food.this)

}

}

**Выводы:**

В процессе лабораторной работы были изучены основы языка scala.