

**Министерство цифрового развития, связи и массовых
коммуникаций Российской Федерации**

Ордена Трудового Красного Знамени

**Федеральное государственное бюджетное образовательное
учреждение высшего образования**

**Кафедра «Математической кибернетики и информационных
технологий»**

Отчет по лабораторной работе №1
по дисциплине **“Введение в информационные технологии”**
на тему:
“Введение в функциональное программирование”

Выполнил: студент БВТ1903

Нестеров Юрий Дмитриевич

Проверил:

Мосева Марина Сергеевна

Москва

2021

Цель: использовать функциональное программирование при написании кода на языке Scala и компилятора Scala REPL.

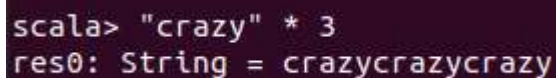
Ход работы

Используется терминал в Ubuntu LTS 20.04

1. Переменные `res` – это значения `val` или настоящие переменные `var`?

Ответ: `res` в Scala являются `val`

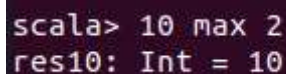
2. “crazy” * 3 в REPL



```
scala> "crazy" * 3
res0: String = crazycrazycrazy
```

Рисунок 1 – Выполнение команды в терминале

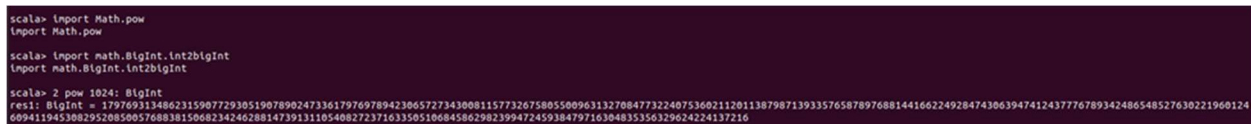
3. Что означают выражение `10 max 2`? В каком классе определён метод `max`?



```
scala> 10 max 2
res10: Int = 10
```

Рисунок 2 – Выполнение команды в терминале

4. Используя число типа `BigInt`, вычислите 2^{1024}



```
scala> import Math.pow
import Math.pow

scala> import math.BigInt.Int2BigInt
import math.BigInt.Int2BigInt

scala> 2 pow 1024: BigInt
res1: BigInt = 179769313486231590772930519078902473361797697894230657273430081157732675805500963132708477322407536021120113879871393357658789768814416622492847430639474124377767893424865485276302219601246094119453082952085005768838150682342462881473913110540827237163350510684586298239947245938479716304835356329624224137216
```

Рисунок 3 – Выполнение команд в терминале

5. Что нужно импортировать, чтобы найти случайное простое число вызовом метода `probablePrime(100, Random)` без использования каких-либо префиксов перед именами `probablePrime` и `Random`?

```
scala> import math.BigInt.probablePrime
import math.BigInt.probablePrime

scala> import util.Random
import util.Random

scala> probablePrime(100, Random)
res2: scala.math.BigInt = 961211626380467368414214375819
```

Рисунок 4 – Выполнение команд в терминале

6. Один из способов создать файл или каталог со случайным именем состоит в том, чтобы сгенерировать случайное число типа `BigInt` и преобразовать его в систему счисления по основанию 36, в результате получится строка, такая как "qsnvbevtomcj38o06kul". Отыщите в Scaladoc методы, которые можно было бы использовать для этого.

```
scala> probablePrime(100, Random) toString 36
res9: String = 3a08zqok0pqqylmqgrwhj
```

Рисунок 5 – Выполнение команды в терминале

7. Как получить первый символ строки в языке Scala? А последний символ?

```
scala> "nippyfox".head
res12: Char = n

scala> "nippyfox".last
res13: Char = x
```

Рисунок 6 – Выполнение команд в терминале

8. Что делают строковые функции `take`, `drop`, `takeRight` и `dropRight`? Какие недостатки и преимущества они имеют в сравнении с `substring`?

```
scala> "nippyfox" take 3
res22: String = nip

scala> "nippyfox" drop 3
res23: String = pyfox

scala> "nippyfox" takeRight 3
res24: String = fox

scala> "nippyfox" dropRight 3
res25: String = nippy
```

Рисунок 7 – Выполнение команд в терминале

9. Сигнум числа равен 1, если число положительное. -1 – если отрицательное, и 0 – если равно нулю. Напишите функцию, вычисляющую это значение.

```
scala> def findSignum(i:Int) = if (i>0) 1 else if (i<0) -1 else 0
findSignum: (i: Int)Int

scala> findSignum(14)
res0: Int = 1

scala> findSignum(-14)
res1: Int = -1

scala> findSignum(0)
res2: Int = 0
```

Рисунок 8 – Выполнение функции в терминале

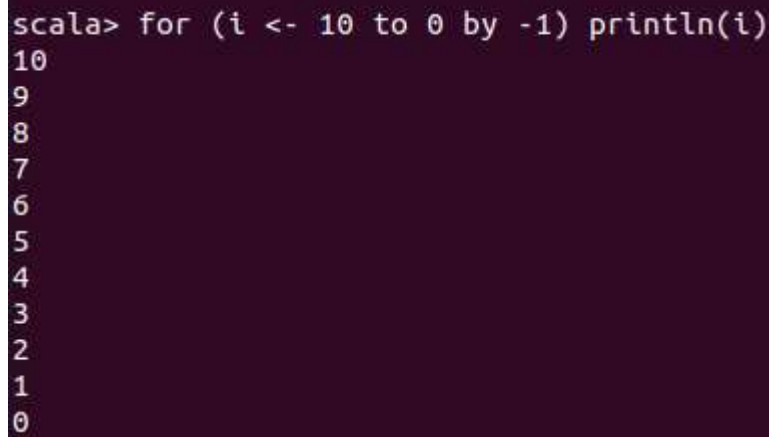
10. Какое значение возвращает блок {}? Каков его тип?

```
scala> {}

scala> val block = {}
block: Unit = ()
```

Рисунок 9 – Выполнение команд в терминале

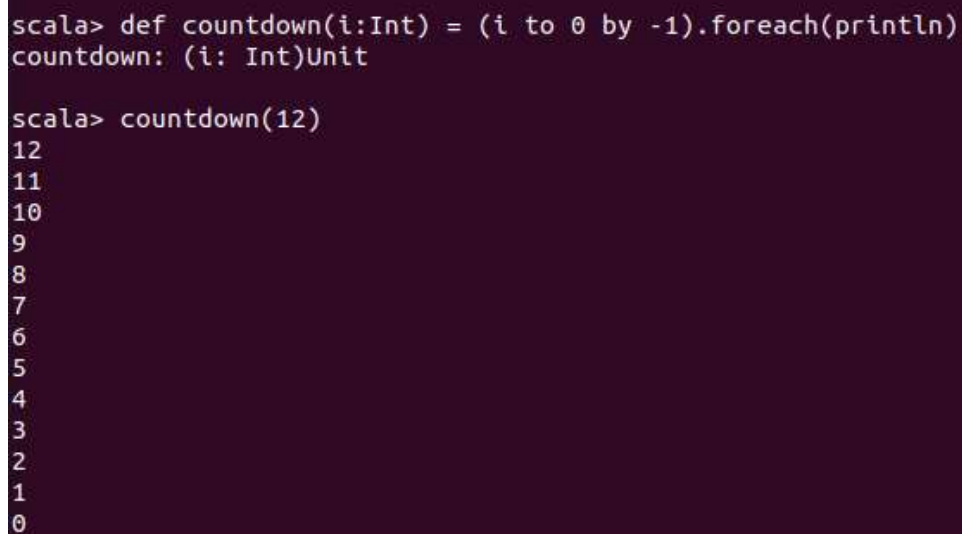
11. Напишите на языке Scala цикл, эквивалентный циклу на языке Java
`for (int i=10; i>=0; i--) System.out.println(i);`



```
scala> for (i <- 10 to 0 by -1) println(i)
10
9
8
7
6
5
4
3
2
1
0
```

Рисунок 10 – Выполнение команды в терминале

12. Напишите процедуру `countdown (n: Int)`, которая выводит числа от `n` до 0.

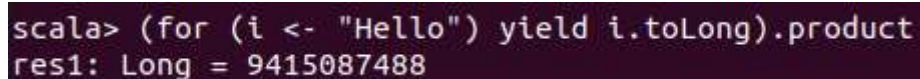


```
scala> def countdown(i:Int) = (i to 0 by -1).foreach(println)
countdown: (i: Int)Unit

scala> countdown(12)
12
11
10
9
8
7
6
5
4
3
2
1
0
```

Рисунок 11 – Выполнение функции в терминале

13. Напишите цикл `for` для вычисления кодовых пунктов Юникода всех букв в строке. Например, произведение символов в строке «Hello» равно 9415087488L.



```
scala> (for (i <- "Hello") yield i.toLong).product
res1: Long = 9415087488
```

Рисунок 12 – Выполнение команды в терминале

14. Решите предыдущее упражнение без применения цикла. Напишите функцию `product(s: String)`, вычисляющую произведение, как описано в предыдущих упражнениях.

```
scala> def product(s:String) = s.map(s => s.toLong).product
product: (s: String)Long

scala> product("Hello")
res2: Long = 9415087488
```

Рисунок 13 – Выполнение функции в терминале

15. Сделайте функцию из предыдущего упражнения рекурсивной.

```
scala> def product(s:String, res:Long = 1):Long = if (s.isEmpty) res else product(s.tail, s.head.toLong * res)
product: (s: String, res: Long)Long

scala> product("Hello")
res17: Long = 9415087488
```

Рисунок 14 – Выполнение функции в терминале

16. Напишите функцию, вычисляющую x^n , где n – целое число.

Используйте следующее рекурсивное определение:

- $x^n = y^2$, если n – четное и положительное число, где $y = x^{n/2}$
- $x^n = x * x^{n-1}$, если n – нечетное и положительное число.
- $x^0 = 1$.
- $x^n = 1/x^{-n}$, если n – отрицательное число.

Не используйте инструкцию `return`.

```
scala> import math.pow
import math.pow

scala> def findxn(x: Double, n: Int): Double = if (n > 0 && (n % 2 == 0)) pow(x, n/2) else if (n > 0 && (n % 2 != 0)) x * pow(x, n-1) else if (n == 0) 1 else 1 / pow(x, -n)
findxn: (x: Double, n: Int)Double

scala> findxn(4, 4)
res0: Double = 16.0

scala> findxn(4, 3)
res1: Double = 64.0

scala> findxn(4, 0)
res2: Double = 1.0

scala> findxn(4, -1)
res3: Double = 0.25
```

Рисунок 15 – Выполнение функции в терминале

17. $f(m,n)$ - сумма всех натуральных чисел от m до n включительно, в десятичной записи которых нет одинаковых цифр.

```
scala> def f(m: Int, n: Int): Int = (m to n).filter(x => (x.toString.size == x.toString.toSet.size)).sum
f: (m: Int, n: Int)Int

scala> f(1, 12)
res4: Int = 67

scala> f(1, 13)
res5: Int = 80

scala> f(1, 11)
res6: Int = 55

scala> f(1, 10)
res7: Int = 55
```

Рисунок 16 – Выполнение функции в терминале

18. Список содержит целые числа, а также другие списки такие же, как и первоначальный. Получить список, содержащий только целые числа из всех вложенных списков.

```
scala> def f(l: List[Any]): List[Any] = l match {
  | case Nil => Nil
  | case (x: List[Any]) :: tail => f(x) :: f(tail)
  | case x :: tail => x :: f(tail)
  | }
f: (l: List[Any])List[Any]

scala> f(List(List(1, 1), 2, List(3, List(5, 8))))
res4: List[Any] = List(1, 1, 2, 3, 5, 8)
```

Рисунок 17 – Выполнение функции в терминале

19. $f(n)$ - сумма цифр наибольшего простого делителя натурального числа n .

```
scala> def primeDev(n: Int) = (1 to n).filter(x => (n % x == 0 && ((2 until x) forall (x % _ != 0)))).max
primeDev: (n: Int)Int

scala> primeDev(14)
res6: Int = 7

scala> primeDev(15)
res7: Int = 5

scala> primeDev(16)
res8: Int = 2

scala> primeDev(17)
res9: Int = 17
```

Рисунок 18 – Выполнение функции в терминале

20. Список содержит элементы одного, но любого типа. Получить список, содержащий каждый имеющийся элемент старого списка k раз подряд. Число k задается при выполнении программы.

```
scala> def repeat(k: Int, l: List[Any]): List[Any] = l.flatMap(List.fill(k)(_))
repeat: (k: Int, l: List[Any])List[Any]

scala> repeat(4, List(1, 2, 3, 4))
res10: List[Any] = List(1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4)
```

Рисунок 19 – Выполнение функции в терминале

21. $f(m,n)$ - наименьшее общее кратное натуральных чисел m и n .

```
scala> def nok(n: Int, m: Int): Int = if (n % m == 0) n else nok(n + n, m)
nok: (n: Int, m: Int)Int

scala> nok(3, 4)
res0: Int = 12
```

Рисунок 20 – Выполнение функции в терминале

22. Список содержит элементы одного, но любого типа. Получить список, из элементов исходного, удаляя каждый k -й элемент. Число k задается при выполнении программы.

```
scala> def del(l: List[Any], k: Int): List[Any] = {
  | var lm: List[Any] = List();
  | var i = 0;
  | (0 to l.length - 1).foreach(i => if (((i + 1) % k) != 0) lm = lm :+ l(i));
  | lm
  | }
del: (l: List[Any], k: Int)List[Any]

scala> del(List(21, 22, 23, 24, 25, 26, 27), 3)
res1: List[Any] = List(21, 22, 24, 25, 27)
```

Рисунок 21 – Выполнение функции в терминале

23. $f(n,k)$ - число размещений из n по k . Факториал не использовать.

```
scala> def permutation(n: Int, k: Int) = ((1 to n).product) / ((1 to (n-k)).product)
permutation: (n: Int, k: Int)Int

scala> permutation(4, 3)
res4: Int = 24
```

Рисунок 22 – Выполнение функции в терминале

24. Список содержит элементы одного, но любого типа. Получить новый список, перемещая циклически каждый элемент на k позиций влево (при перемещении на одну позицию первый элемент становится последним, второй - первым и так далее). Число k задается при выполнении программы. Если k отрицательное, то перемещение происходит вправо.

```
scala> def swop(l: List[Int], i: Int) = if (i > 0) l.drop(i) ++ l.take(i) else l.takeRight((-i).abs) ++ l.dropRight((-i).abs)
swop: (l: List[Int], i: Int)List[Int]

scala> swop(List(1, 2, 3, 4, 5, 6, 7), 3)
res0: List[Int] = List(4, 5, 6, 7, 1, 2, 3)

scala> swop(List(1, 2, 3, 4, 5, 6, 7), -2)
res1: List[Int] = List(6, 7, 1, 2, 3, 4, 5)
```

Рисунок 23 – Выполнение функции в терминале

25. $f(n)$ - наибольшее совершенное число не превосходящее n . Совершенным называется натуральное число n равное сумме своих делителей, меньших n , например $6 = 1 + 2 + 3$ ($f(6) = 6$, $f(7) = 6$, ...).

```
scala> def q(x: Int) = (1 to x).filter(x => x == (1 to (x-1)).filter(y => x % y == 0).sum).max
q: (x: Int)Int

scala> q(6)
res17: Int = 6

scala> q(7)
res18: Int = 6
```

Рисунок 24 – Выполнение функции в терминале

26. Список содержит элементы одного, но любого типа. Получить два списка из элементов исходного, выбирая в первый элементы с чётными индексами, а во второй с нечётными.

```
scala> def oddEven(
  | l: List[Any],
  | l1: List[Any] = List(),
  | l2: List[Any] = List(),
  | isOdd: Boolean = false): List[Any] = {
  |   if (l.length > 0) { if (isOdd) oddEven(l.drop(1), l1, l2 := l(0), !isOdd) else oddEven(l.drop(1), l1 := l(0), l2, !isOdd) }
  |   else List(l1, l2)
  | }
oddEven: (l: List[Any], l1: List[Any], l2: List[Any], isOdd: Boolean)List[Any]

scala> oddEven(List(1, 2, 3, 4, 5, 6, 7))
res4: List[Any] = List(List(1, 3, 5, 7), List(2, 4, 6))
```

Рисунок 25 – Выполнение функции в терминале

27. $f(n)$ – наибольшее из чисел от 1 до n включительно, обладающее свойством: сумма цифр n в некоторой степени > 1 равна самому числу n . Пример: $512 = 8^3$.

```
scala> import math.pow
import math.pow

scala> import math.sqrt
import math.sqrt

scala> def sum(n: Int): Int = if (n == 0) 0 else (n % 10) + sum(n / 10)
sum: (n: Int)Int

scala> def searchin(n: Int) = (2 to (sqrt(n).toInt)).filter(x => n == pow(sum(n), x)).max
searchin: (n: Int)Int

scala> searchin(512)
res2: Int = 3
```

Рисунок 26 – Выполнение функции в терминале

28. Список в качестве элементов содержит кортежи типа: (n, s) , где n — целые числа, а s — строки. Получить два списка из элементов исходного, выбирая в первый числа, а во второй строки из кортежей.

```
scala> def cortage(l: List[List[Any]], l1: List[Any] = List(), l2: List[Any] = List()): List[List[Any]] = if (l.length > 0) cortage(l.drop(1), l1 := l(0)(0), l2 := l(0)(1)) else List(l1, l2)
cortage: (l: List[List[Any]], l1: List[Any], l2: List[Any])List[List[Any]]

scala> cortage(List(List(0, "nippyfox"), List(1, "Scala"), List(2, "functional")))
res7: List[List[Any]] = List(List(0, 1, 2), List(nippyfox, Scala, functional))
```

Рисунок 27 – Выполнение функции в терминале

Вывод

Используя язык Scala и терминал Ubuntu я реализовал задачи, используя принцип функционального программирования.