

Министерство цифрового развития, связи и массовых коммуникаций  
Ордена Трудового Красного Знамени федеральное государственное бюджетное  
образовательное учреждение высшего образования  
«Московский технический университет связи и информатики»

Кафедра Математической кибернетики и информационных технологий

Лабораторная работа № 1  
по дисциплине «Введение в информационные технологии»

Выполнил: студент группы БВТ1901

Лапин Виктор Андреевич

Проверила: ст. пр.

Мосева Марина Сергеевна

Москва

2021

## ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ.....	3
2. ВЫПОЛНЕНИЕ РАБОТЫ .....	3
3. ВЫВОД .....	16

## 1. ВВЕДЕНИЕ

Цель лабораторной работы состоит в изучении основ синтаксиса Scala и ознакомлении с функционалом REPL для выполнения задач.

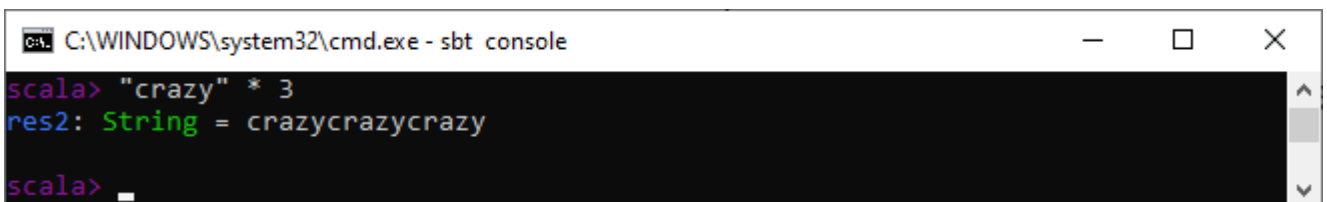
## 2. ВЫПОЛНЕНИЕ РАБОТЫ

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

Переменные `res` являются значениями `val` (неизменяемыми).

### 2. `"crazy" * 3` в REPL

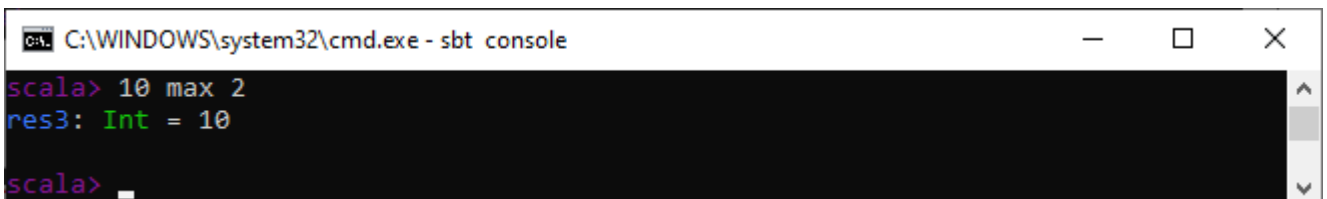
Возвращает новую строку, состоящую из трёх повторений заданной.



```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> "crazy" * 3
res2: String = crazycrazycrazy
scala> _
```

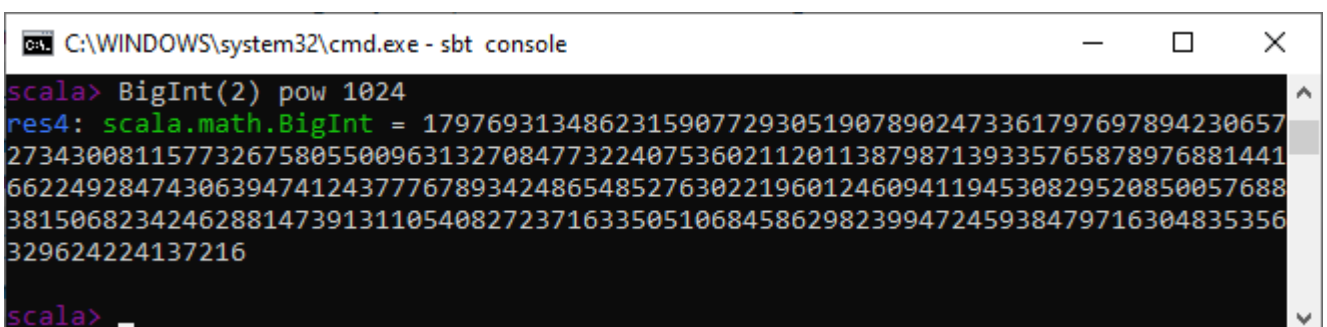
### 3. Что означает выражение `10 max 2`? В каком классе определен метод `max`?

Возвращает большее из двух чисел. Метод определён в классе `RichInt`.



```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> 10 max 2
res3: Int = 10
scala> _
```

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



```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> BigInt(2).pow(1024)
res4: scala.math.BigInt = 179769313486231590772930519078902473361797697894230657
27343008115773267580550096313270847732240753602112011387987139335765878976881441
66224928474306394741243777678934248654852763022196012460941194530829520850057688
38150682342462881473913110540827237163350510684586298239947245938479716304835356
329624224137216
scala> _
```

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

`BigInt.probablePrime` и `util.Random`.

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> import BigInt.probablePrime
import BigInt.probablePrime

scala> import util.Random
import util.Random

scala> probablePrime(100, Random)
res5: scala.math.BigInt = 1254315733424136056514597399857

scala> _
```

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.

Случайное число можно генерируется методом `probablePrime`. Метод `toString(radix: Int)` возвращает строковое представление `BigInt` по основанию `radix`.

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> probablePrime(100, Random).toString(36)
res1: String = 1vf1hysjff3nqqm90851

scala> _
```

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

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> "Hello World!"
res7: String = Hello World!

scala> res7(0)
res8: Char = H

scala> res7(res7.size - 1)
res9: Char = !

scala> _
```

## 8. Что делают строковые функции `take`, `drop`, `takeRight` и `dropRight`?

Какие преимущества и недостатки они имеют в сравнении с `substring`?

Метод `take(n: Int)` выбирает первые `n` символов, `drop(n: Int)` выбирает все символы кроме `n` первых, `takeRight(n: Int)` выбирает последние `n` символов, `dropRight(n: Int)` выбирает все символы кроме `n` последних. Преимущество этих методов заключается в том, что можно не указывать дополнительные аргументы, недостаток в том, что нельзя отбросить символы сразу слева и справа.

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> "Hello World!"
res8: String = Hello World!

scala> res8 take 5
res9: String = Hello

scala> res8 drop 6
res10: String = World!

scala> res8 takeRight 1
res11: String = !

scala> res8 dropRight 8
res12: String = Hell

scala> _
```

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

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> val sign = (n: Int) => if (n == 0) 0 else if (n > 0) 1 else -1
sign: Int => Int = $Lambda$4669/0x00000000801a9bcc8@bc634b5

scala> sign(0)
res9: Int = 0

scala> sign(3)
res10: Int = 1

scala> sign(-7)
res11: Int = -1

scala>
```

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

Блок имеет тип и значение последнего выражения.

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> {"string"}
res10: String = string

scala> {"string"(3)}
res11: Char = i

scala>
```

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

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> for(i <- 10 to 0 by -1) println(i)
10
9
8
7
6
5
4
3
2
1
0

scala> _
```

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

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> def countdown(n: Int) = for(i <- n to 0 by -1) println(i)
countdown: (n: Int)Unit

scala> countdown(7)
7
6
5
4
3
2
1
0

scala> _
```

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

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> var mul: Long = 1
mul: Long = 1

scala> for(c <- "Hello") mul *= c

scala> mul
res13: Long = 9415087488

scala> _
```

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

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> def product(s: String) = s.foldLeft(1: Long)((x, y) => x * y)
product: (s: String)Long

scala> product("Hello")
res14: Long = 9415087488

scala> _
```

16. Сделайте функцию из предыдущего упражнения рекурсивной.

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> def product(s: String, pos: Int = 0): Long =
  | if(pos >= s.size) 1 else s(pos) * product(s, pos + 1)
product: (s: String, pos: Int)Long

scala> product("Hello")
res16: Long = 9415087488

scala> _
```

17. 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.

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> def xn(x: Double, n: Int): Double = {
  |   if(n == 0) 1
  |   else if(n < 0) 1 / xn(x, -n)
  |   else if(n % 2 == 1) x * xn(x, n - 1)
  |   else {
  |     val y = xn(x, n / 2)
  |     y * y
  |   }
  | }
xn: (x: Double, n: Int)Double

scala> xn(2, 9)
res17: Double = 512.0

scala> xn(2, -9)
res18: Double = 0.001953125

scala> xn(0.25, -1)
res19: Double = 4.0

scala> xn(-7, 2)
res20: Double = 49.0

scala> xn(-7, 0)
res21: Double = 1.0

scala> _
```



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

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> val sum = (m: Int, n: Int) =>
  | (m to n).filter(x => x.toString.distinct.size == x.toString.size).sum
sum: (Int, Int) => Int = $Lambda$4750/0x0000000801ac8658@4ed51842

scala> sum(1, 5)
res18: Int = 15

scala> sum(1, 10)
res19: Int = 55

scala> sum(1, 11)
res20: Int = 55

scala> sum(440, 449)
res21: Int = 0

scala> sum(440, 450)
res22: Int = 450

scala> _
```

19. Список содержит целые числа, а также другие списки, такие же, как и первоначальный. Получить список, содержащий только целые числа из всех вложенных списков. Пример:  $f(\text{List}(\text{List}(1, 1), 2, \text{List}(3, \text{List}(5, 8)))) = \text{List}(1, 1, 2, 3, 5, 8)$ .

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> import scala.collection.mutable.ListBuffer
import scala.collection.mutable.ListBuffer

scala> def f(list: List[Any]): List[Int] = {
  | var newList = new ListBuffer[Int]()
  | list.foreach(x => {
  |   if(x.isInstanceOf[Int]) newList += x.asInstanceOf[Int]
  |   if(x.isInstanceOf[List[Any]]) newList ++= f(x.asInstanceOf[List[Any]])
  | })
  | newList.toList
  | }
f: (list: List[Any])List[Int]

scala> f(List(List(1, 1), 2, List(3, List(5, 8))))
res19: List[Int] = List(1, 1, 2, 3, 5, 8)

scala> _
```

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

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> val f = (n: Int) => (2 to n).filter(x => n % x == 0).filter(x => {
    | val y = (2 to (x - 1)); !y.exists(n => x % n == 0)
    | }).max.toString.foldLeft(0: Int)((x, y) => x + y - 48)
f: Int => Int = $Lambda$4768/0x00000000801a9f538@4fe6f451

scala> f(28)
res20: Int = 7

scala> f(1000)
res21: Int = 5

scala> f(1337)
res22: Int = 11

scala> f(65536)
res23: Int = 2

scala> f(65537)
res24: Int = 26

scala> _
```

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

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> val f = (list: List[Any], k: Int) => list.flatMap(x => List.fill(k)(x))
f: (List[Any], Int) => List[Any] = $Lambda$4759/0x00000000801aa9e88@4044cf67

scala> f(List(1, 2, 3), 3)
res21: List[Any] = List(1, 1, 1, 2, 2, 2, 3, 3, 3)

scala> f(List("first", "second", "third"), 2)
res22: List[Any] = List(first, first, second, second, third, third)

scala> _
```

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

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> val f = (n: Int) => (2 to n).filter(x => n % x == 0).filter(x => {
  |   val y = (2 to (x - 1)); !y.exists(n => x % n == 0)
  | }).max.toString.foldLeft(0: Int)((x, y) => x + y - 48)
f: Int => Int = $Lambda$4781/0x00000000801adb2b0@1b49913c

scala> f(28)
res22: Int = 7

scala> f(1000)
res23: Int = 5

scala> f(1337)
res24: Int = 11

scala> f(65536)
res25: Int = 2

scala> f(65537)
res26: Int = 26

scala> _
```

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

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> val f = (list: List[Any], k: Int) => list.flatMap(x => List.fill(k)(x))
f: (List[Any], Int) => List[Any] = $Lambda$4730/0x00000000801aa36b8@203fa910

scala> f(List(1, 2, 3), 3)
res23: List[Any] = List(1, 1, 1, 2, 2, 2, 3, 3, 3)

scala> f(List("first", "second", "third"), 2)
res24: List[Any] = List(first, first, second, second, third, third)

scala> _
```

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

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> val f = (m: Int, n: Int) =>
  | (m max n to m * n).filter(x => x % m == 0 && x % n == 0).min
f: (Int, Int) => Int = $Lambda$4748/0x0000000801acc8d8@118492dc

scala> f(3, 5)
res24: Int = 15

scala> f(24, 36)
res25: Int = 72

scala> f(10, 15)
res26: Int = 30

scala> _
```

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

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> import scala.collection.mutable.ListBuffer
import scala.collection.mutable.ListBuffer

scala> val f = (l: List[Any], k: Int) => {
  | var nl = new ListBuffer[Any]
  | (0 to l.size - 1).filter(i => (i + 1) % k != 0).foreach(i => nl += l(i))
  | nl.toList
  | }
f: (List[Any], Int) => List[Any] = $Lambda$4785/0x0000000801aa34b0@29276183

scala> f((1 to 10).toList, 3)
res25: List[Any] = List(1, 2, 4, 5, 7, 8, 10)

scala> _
```

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

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> val f = (n: Int, k: Int) =>
  | (n to (n - k + 1) by -1).foldLeft(1: Long)((x, y) => x * y)
f: (Int, Int) => Long = $Lambda$4795/0x0000000801aa6040@668e1a8a

scala> f(4, 2)
res26: Long = 12

scala> f(7, 4)
res27: Long = 840

scala> _
```

27. 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.

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> var f = (l: List[Any], k: Int) => {
  |   (0 to l.size - 1)
  |   .map(i => l((i + (1 - k / l.size) * l.size + k) % l.size))
  |   .toList
  | }
f: (List[Any], Int) => List[Any] = $Lambda$4741/0x00000000801aa3ed0@60d6bf2b
scala> f(List(1, 2, 3, 4, 5, 6), 1)
res27: List[Any] = List(2, 3, 4, 5, 6, 1)
scala> f(List(1, 2, 3, 4, 5, 6), 6)
res28: List[Any] = List(1, 2, 3, 4, 5, 6)
scala> f(List(1, 2, 3, 4, 5, 6), 7)
res29: List[Any] = List(2, 3, 4, 5, 6, 1)
scala> f(List(1, 2, 3, 4, 5, 6), 0)
res30: List[Any] = List(1, 2, 3, 4, 5, 6)
scala> f(List(1, 2, 3, 4, 5, 6), -1)
res31: List[Any] = List(6, 1, 2, 3, 4, 5)
scala> f(List(1, 2, 3, 4, 5, 6), -6)
res32: List[Any] = List(1, 2, 3, 4, 5, 6)
scala> f(List(1, 2, 3, 4, 5, 6), -7)
res33: List[Any] = List(6, 1, 2, 3, 4, 5)
scala> f(List(1, 2, 3, 4, 5, 6), 12)
res34: List[Any] = List(1, 2, 3, 4, 5, 6)
scala> f(List(1, 2, 3, 4, 5, 6), -12)
res35: List[Any] = List(1, 2, 3, 4, 5, 6)
scala> f(List(1, 2, 3, 4, 5, 6), 3)
res36: List[Any] = List(4, 5, 6, 1, 2, 3)
scala> f(List(1, 2, 3, 4, 5, 6), -3)
res37: List[Any] = List(4, 5, 6, 1, 2, 3)
scala> _
```

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

```
C:\WINDOWS\system32\cmd.exe - sbt console

scala> val f = (n: Int) =>
  | (2 to n).filter(x => (1 to x - 1).filter(y => x % y == 0).sum == x).max
f: Int => Int = $Lambda$4763/0x00000000801ae1c88@7125d133

scala> f(6)
res28: Int = 6

scala> f(7)
res29: Int = 6

scala> f(64)
res30: Int = 28

scala> f(512)
res31: Int = 496

scala> f(8192)
res32: Int = 8128

scala> _
```

29. Список содержит элементы одного, но любого типа. Получить два списка из элементов исходного, выбирая в первый элементы с четными индексами, а во второй с нечетными.

```
C:\WINDOWS\system32\cmd.exe - sbt console

scala> import scala.collection.mutable.ListBuffer
import scala.collection.mutable.ListBuffer

scala> val f = (l: List[Any]) => {
  | var l1 = new ListBuffer[Any]
  | var l2 = new ListBuffer[Any]
  | (0 to l.size - 1).foreach(i => if(i % 2 == 0) l1 += l(i) else l2 += l(i))
  | List(l1.toList, l2.toList)
  | }
f: List[Any] => List[List[Any]] = $Lambda$4770/0x00000000801ae7730@49cdaa31

scala> f(List(0, 1, 2, 3, 4, 5, 6, 7))
res29: List[List[Any]] = List(List(0, 2, 4, 6), List(1, 3, 5, 7))

scala> f(List("first", "second", "third", "fourth", "fifth"))
res30: List[List[Any]] = List(List(first, third, fifth), List(second, fourth))

scala> f(List("a", "b", "c", "d", "e", "f", "g", "h"))
res31: List[List[Any]] = List(List(a, c, e, g), List(b, d, f, h))

scala> _
```

30.  $f(n)$  - наибольшее из чисел от 1 до  $n$  включительно, обладающее свойством: сумма цифр  $n$  в некоторой степени  $> 1$  равна самому числу  $n$ .  
Пример:  $512 = 83$

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> val f = (n: Int) =>
  | (1 to n).filter(i => {
  |   var x = i.toString.foldLeft(0: Int)((x, y) => x + y - 48)
  |   x = x * x
  |   while(x < i && x > 1) x = x * x
  |   x == i
  | }).max
f: Int => Int = $Lambda$4757/0x00000000801aae718@f61e6d9
scala> f(512)
res30: Int = 81
scala> f(83)
res31: Int = 81
scala> f(80)
res32: Int = 1
scala> f(10000)
res33: Int = 2401
scala> f(1000000)
res34: Int = 614656
scala> _
```

31. 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.

```
C:\WINDOWS\system32\cmd.exe - sbt console
scala> import scala.collection.mutable.ListBuffer
import scala.collection.mutable.ListBuffer
scala> val f = (list: List[Tuple2[Int, String]]) => {
  |   var i = new ListBuffer[Int]
  |   var s = new ListBuffer[String]
  |   (0 to list.size - 1).foreach(x => { i += list(x)._1; s += list(x)._2 })
  |   List(i.toList, s.toList)
  | }
f: List[(Int, String)] => List[List[Any]] = $Lambda$4793/0x00000000801aae510@49e0fb12
scala> f(List((1, "one"), (2, "two"), (3, "three")))
res31: List[List[Any]] = List(List(1, 2, 3), List(one, two, three))
scala> _
```

### 3. ВЫВОД

В ходе данной лабораторной работы были изучены основы синтаксиса Scala и выполнено ознакомление с функционалом REPL для выполнения задач.