

# Лабораторная работа №1

## «Введение в функциональное программирование на языке Scala»

Скоробогатов С.Ю.

27 февраля 2016 г.

### 1 Цель работы

Целью данной работы является ознакомление с программированием на языке Scala на основе чистых функций.

### 2 Исходные данные

Для выполнения лабораторной работы потребуется REPL-интерпретатор языка Scala, который можно установить через менеджер пакетов ОС Linux или вручную, скачав с сайта по ссылке: <http://www.scala-lang.org/download/>.

Для запуска REPL-интерпретатора нужно запустить команду

```
scala
```

Для выхода из REPL-интерпретатора нужно ввести команду «:quit» или использовать комбинацию клавиш Ctrl-D.

В командной строке интерпретатора сразу же после приглашения «scala>» можно вводить операторы языка Scala. При этом интерпретатор будет сразу же вычислять и выводить значения, связанные с переменными, или сообщать об ошибках в коде.

Например, после ввода оператора

```
scala> val x = 10*(2+3)
```

интерпретатор напечатает

```
x: Int = 50
```

Операторы, занимающие несколько строк, удобно вводить в специальном режиме, который инициируется командой «:paste». При этом для завершения ввода оператора нужно нажать Ctrl-D. Например,

```
scala> :paste
// Entering paste mode (ctrl-D to finish)
val sum: (Int => Boolean) => (List[Int] => Int) =
  p => {
    case Nil                => 0
    case x :: xs if (p(x)) => x + sum(p)(xs)
    case x :: xs            => sum(p)(xs)
  }
// Exiting paste mode, now interpreting.
sum: (Int => Boolean) => (List[Int] => Int) = <function1>
```

### 3 Задание

Выполнение лабораторной работы заключается в составлении и отладке с помощью REPL-интерпретатора одной из функций, приведённых в таблицах 1 и 2. При составлении функции запрещается использовать возможности Scala, выходящие за рамки функциональной парадигмы. Кроме того, запрещается применять функции стандартной библиотеки языка Scala.

Таблица 1: Варианты функций

1	Функция <code>fib: (Int, Int =&gt; Boolean) =&gt; List[Int]</code> , порождающая последовательность чисел Фибоначчи, не превышающих заданного целого числа и удовлетворяющих некоторому предикату.
2	Функция <code>powers: (List[Int], Int =&gt; Boolean) =&gt; List[Int]</code> , удаляющая из списка те числа, которые не являются заданными предикатом степенями числа 2.
3	Закаренная функция <code>power: Int =&gt; (Int =&gt; Int)</code> , выполняющая быстрое возведение числа в указанную степень (параметр функции – степень).
4	Функция <code>reverseP: (List[Int], Int =&gt; Boolean) =&gt; List[Int]</code> , выполняющая переворачивание списка целых чисел и удаление из него элементов, не удовлетворяющих предикату.
5	Функция <code>translate: (List[Int], Int =&gt; Int) =&gt; List[Int]</code> , порождающая список целых чисел, полученный из исходного списка путём применения к каждому его элементу функции перевода, принимающей <code>Int</code> и возвращающей <code>Int</code> .
6	Функция <code>filterIndexes: (List[Int], Int =&gt; Boolean) =&gt; List[Int]</code> , удаляющая из списка целых чисел те числа, номера которых в списке не удовлетворяют заданному предикату.
7	Функция <code>flatten: (List[List[Int]], Int =&gt; Boolean) =&gt; List[Int]</code> , выполняющая конкатенацию списков целых чисел, находящихся в списке списков целых чисел и имеющих длину, удовлетворяющую предикату.
8	Функция <code>split: (List[Int], Int =&gt; Boolean) =&gt; List[List[Int]]</code> , выполняющая разбиение последовательности целых чисел на подпоследовательности, разделённые числами, удовлетворяющими предикату.
9	Функция <code>trim: (List[Int], Int =&gt; Boolean) =&gt; List[Int]</code> , выполняющая удаление из списка подряд идущих нулей, количество которых удовлетворяет предикату.
10	Функция <code>filterBySum: (List[Int], Int =&gt; Boolean) =&gt; List[Int]</code> , выполняющая удаление из списка элементов, для которых сумма предыдущих элементов не удовлетворяет предикату.
11	Функция <code>kadane: List[Int] =&gt; (Int, Int)</code> , выполняющая поиск границ подпоследовательности с максимальной суммой (алгоритм Кадана).
12	Функция <code>partition: (List[Int], Int =&gt; Boolean) =&gt; (List[Int], List[Int])</code> , разделяющая элементы исходного списка на два списка в зависимости от того, удовлетворяют ли они предикату.
13	Функция <code>peaks: List[Int] =&gt; List[Int]</code> , формирующая список индексов пиков последовательности (пик – такой элемент, что соседние элементы его не превышают).
14	Закаренная функция <code>digits: Int =&gt; (Int =&gt; List[Int])</code> , выполняющая перевод числа в заданную систему счисления (параметр функции – основание системы счисления).

Таблица 2: Варианты функций

15	<p>Функция</p> <p><code>zipP: (List[Int], List[Int], (Int, Int) =&gt; Boolean) =&gt; List[(Int, Int)]</code>, превращающая два списка целых чисел в список пар целых чисел, в котором первый элемент пары принадлежит первому списку, а второй – второму, и оставляющая в списке только те пары, которые удовлетворяют предикату.</p>
16	<p>Закаренная функция <code>slices: Int =&gt; (List[Int] =&gt; List[List[Int]])</code>, выполняющая разбиение списка целых чисел на фрагменты указанной в качестве параметра функции длины.</p>
17	<p>Закаренная функция <code>frames: Int =&gt; (List[Int] =&gt; List[List[Int]])</code>, формирующая список, состоящий из всех подсписков списка целых чисел указанной в качестве параметра функции длины. Подсписком будем считать список, который можно получить удалением произвольного количества элементов от начала и от конца списка.</p>
18	<p>Закаренная функция <code>comb: Int =&gt; (List[Int] =&gt; List[List[Int]])</code>, формирующая список всех сочетаний элементов списка целых чисел. Размер сочетания передаётся через параметр функции.</p>
19	<p>Функция <code>coprimes: List[Int] =&gt; List[(Int, Int)]</code>, выполняющая поиск в списке целых чисел пар взаимно простых чисел. Функция должна возвращать список найденных пар, причём в каждой паре первое число должно быть меньше второго.</p>
20	
21	
22	
23	
24	
25	
26	
27	
28	