**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение высшего образования  
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ имени Н.Э.БАУМАНА  
(национальный исследовательский университет)»**

Факультет: Информатика и системы управления

Кафедра: Теоретическая информатика и компьютерные технологии

**Лабораторная работа № 1**

“Введение в функциональное программирование на языке Scala”

”по дисциплине «Объектно-функциональное программирование»

Вариант 16

Работу выполнил

студент группы ИУ9-62Б

Сербин Денис

Москва, 2022

# **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, ко- торый можно установить через менеджер пакетов ОС Linux или вручную, скачав с сайта по ссылке: https://downloads.lightbend.com/scala/2.13.4/scala-2.13.4.tgz.

Инструкции по установке можно почитать здесь: https://www.scala-lang.org/download/ install.html.

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 вво- дить 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, кото- рый 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. При составлении функции за- прещается использовать возможности Scala, выходящие за рамки функциональной парадиг- мы. Кроме того, запрещается применять функции стандартной библиотеки языка Scala.

**4. Индивидуальный вариант**

# Функция partition: (List[Int], Int) => (List[Int], List[Int]), разделяющая список целых чисел на два списка: в первый список помещаются числа, которые меньше указанного числа, а во второй числа, которые не меньше.

# **5. Результат выполнения**

**val** *partition* : (List[Int], Int) => (List[Int], List[Int]) = {  
 **def** union: ((List[Int], List[Int]), (List[Int], List[Int])) => (List[Int], List[Int]) = {  
 **case** ((l1, l2), (l3, l4)) => (l1 ::: l3, l2 ::: l4)  
 }  
 {  
 **case** (*Nil*, n) => (*Nil*, *Nil*)  
 **case** (x *::* xs, n) **if** x < n => union((*List*(x), *Nil*), *partition*(xs, n))  
 **case** (x *::* xs, n) **if** x >= n => union((*Nil*, *List*(x)), *partition*(xs, n))  
 }  
}

Рисунок 1. Код программы

# 

Рисунок 2. Проверка работоспособности программы

# **6. Вывод**

# В рамках данной лабораторной работы была реализована функция partition: (List[Int], Int) => (List[Int], List[Int]), разделяющая список целых чисел на два списка: в первый список помещаются числа, которые меньше указанного числа, а во второй числа, которые не меньше.