

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

КУРСОВАЯ РАБОТА

По дисциплине «Функциональное программирование»

Выполнил
студент гр. 3530904/80002

Карпенко А.В.

Руководитель

Лукашин А.А.

20 декабря 2019 г.

Оглавление

Описание задачи	3
Ход работы	4
Результаты работы	Ошибка! Закладка не определена.
Вывод	6

Описание задачи

Функциональное программирование — раздел дискретной математики и парадигма программирования, в которой процесс вычисления трактуется как вычисление значений функций в математическом понимании последних (в отличие от функций как подпрограмм в процедурном программировании).

Противопоставляется парадигме императивного программирования, которая описывает процесс вычислений как последовательное изменение состояний (в значении, подобном таковому в теории автоматов). При необходимости, в функциональном программировании вся совокупность последовательных состояний вычислительного процесса представляется явным образом, например, как список.

Функциональное программирование предполагает обходиться вычислением результатов функций от исходных данных и результатов других функций, и не предполагает явного хранения состояния программы. Соответственно, не предполагает оно и изменяемость этого состояния (в отличие от императивного, где одной из базовых концепций является переменная, хранящая своё значение и позволяющая менять его по мере выполнения алгоритма).

Реализовать калькулятор на функциональном языке.

Ход работы

Идея реализации калькулятора состоит в том, чтобы преобразовать инфиксную форму записи выражения в постфиксную форму, и дальше подсчитать полученное выражение с учетом приоритетов. Язык реализации – Scala.

```
// Задаём два значения  
calculator("1.9", "+", "7.2")
```

Полуаем результат

SCALA ⚙	9.1
---------	-----

Код программы

```
1 import scala.collection.immutable.StringOps._
2 import scala.util.{Try, Success, Failure}
3
4 def calculator(left: String, op: String, right: String): Unit = {
5
6     def parse(value: String) = Try(value.toDouble)
7
8     (parse(left), parse(right)) match {
9         case (Success(leftDouble), Success(rightDouble)) => {
10             op match {
11                 case "/" => println(leftDouble / rightDouble)
12                 case "*" => println(leftDouble * rightDouble)
13                 case "+" => println(leftDouble + rightDouble)
14                 case "-" => println(leftDouble - rightDouble)
15                 case invalid: String => println(s"Invalid operator $invalid.")
16             }
17         }
18         case (Failure(e), _) => println(s"Could not parse $left.")
19         case (_, Failure(e)) => println(s"Could not parse $right.")
20         case (Failure(e1), Failure(e2)) => println(s"Could not parse $left and $right.")
21     }
22 }
23
24
```

SCALA ⚙️

Вывод

В ходе выполнения расчетно-графической работы были улучшены навыки владения функциональным языком Scala.