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.

Санкт-Петербург
2019

# Оглавление

# 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

```scala
import scala.collection.immutable.StringOps._
import scala.util.{Try, Success, Failure}

def calculator(left: String, op: String, right: String): Unit = {

  def parse(value: String) = Try(value.toDouble)

  (parse(left), parse(right)) match {
    case (Success(leftDouble), Success(rightDouble)) => {
      op match {
        case "/" => println(leftDouble / rightDouble)
        case "*" => println(leftDouble * rightDouble)
        case "+" => println(leftDouble + rightDouble)
        case "-" => println(leftDouble - rightDouble)
        case invalid: String => println(s"Invalid operator $invalid.")
      }
    }
    case (Failure(e), _) => println(s"Could not parse $left.")
    case(_, Failure(e)) => println(s"Could not parse $right.")
    case(Failure(e1), Failure(e2)) => println(s"Could not parse $left and $r
  }

}
```

## Вывод

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