Лабораторная работа № 1. Введение в функциональное программирование на языке Scala

15 февраля 2023 г.

Яровикова Анастасия, ИУ9-61Б

Цель работы

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

Реализация

В варианте 7 необходимо реализовать функцию flatten, выполняющую конкатенацию списков целых чисел, находящихся в списке списков целых чисел и имеющих длину, удовлетворяющую предикату. Таким образом, функция должна иметь тип: `(List[List[Int]], Int => Bolean) => List[Int])`.

Особенности реализации

В моей реализации для поочередной обработки списков, вложенных в список списков, применена операция декомпозиции списка. А также в реализации использован прием, называемый “закарриванием”. Он позволяет переписать функцию `flatten`, имеющую два параметра (список списков и предикат), так, чтобы функция принимала сначала предикат и возвращала функцию, принимающую список списков. Стоит отметить, что данный прием помогает наглядно представить код функции: интуитивно понятно, что сначала задается предикат для длины вложенных списков, а затем в функцию передается список списков целых чисел для выполнения конкатенации. Впрочем данный аспект представления кода является довольно субъективным.

Код реализации:

```
val myflatten: (Int => Boolean) => (List[List[Int]] => List[Int]) =
    p => {
        case Nil => Nil
```

```
        case x :: xs if (p(x.length)) => x ::: myflatten(p)(xs)
        case x :: xs => myflatten(p)(xs)
    }

object Main {
    def main(args: Array[String]) = {
        val list = List(List(1, 2), List(3), List(-5, 6), List(1, -100))
        val sumLengthAppropriateLists = myflatten(_ == 2)
        val s = sumLengthAppropriateLists(list)
        println(s)
    }
}
```

Тестирование

Для тестирования необходимо в объекте `Main` в функции `main` задать значения переменной `list`, связанную со списоком списков целых чисел, а также предикат длины конкатенируемых вложенных списков, передающийся в качестве параметра функции `myflatten`.

```
// list = List(List(1, 2), List(3), List(-5, 6), List(1, -100))
// sumLengthAppropriateLists = myflatten(_ == 2)
List(1, 2, -5, 6, 1, -100)

// list = List(List(1, 2), List(3), List(-5, 6), List(1, -100))
// sumLengthAppropriateLists = myflatten(_ > 0)
List(1, 2, 3, -5, 6, 1, -100)

// list = List(List(1, 2), List(3), List(-5, 6), List(1, -100))
// sumLengthAppropriateLists = myflatten(_ > 2)
List()
```

Вывод

В ходе данной лабораторной работы было проведено ознакомление с языком программирования Scala, получен опыт программирования чистых функций на нем. Также были изучены особенности написания функций высших порядков, возвращающих значения типа функция. Одной из особенностей представления таких функций является прием “закарривания”, который был удачно применен в реализации поставленной задачи.