Лабораторная работа № 1 «Введение в функциональное программирование на языке Scala»

18 марта 2024 г.

Мадина Балтаева, ИУ9-62Б

Цель работы

Целью данной работы является изучение базовых объектно-ориентированных возможностей языка Scala.

Индивидуальный вариант

Целое число, представленное последовательностью нулей и единиц в фибоначчиевой системе счисления. Операции: сложение («+»); вычисление наибольшего числа, составленного из общих для двух чисел фибоначчиевых слагаемых («%»); перевод в BigInteger («toInteger»).

Реализация и тестирование

```
import java.math.BigInteger
import java.math.BigInteger.ONE

class NewFib(fib: List[Int]) {
  val list = fib

  def toIntHelper2(n: BigInteger): BigInteger = { n match {
      case BigInteger.ONE => n
      case BigInteger.ZERO => n
      case i =>
      toIntHelper2(n.subtract(ONE)).add(toIntHelper2(n.subtract(ONE).subtract(ONE)))
    }
  }

def toIntHelper1(f: List[Int], res: BigInteger): BigInteger = f match {
```

```
    case 0 :: tail => toIntHelper1(tail, res)
    case head :: tail =>
    toIntHelper1(tail, res.add(toIntHelper2(BigInteger.valueOf(f.size + 1))))
    case Nil => res
}

  def toInteger(): BigInteger = {
    toIntHelper1(this.list, BigInteger.ZERO)
  }

  def sumHelper2(f: List[Int], res: List[Int]): List[Int] = f match {
      case Nil => res.reverse
      case f1 :: Nil => sumHelper2(Nil, res ::: f)
      case f1 :: f2 :: Nil => sumHelper2(Nil, res ::: List(f1 + f2))
      case f1 :: f2 :: t => val sum = f1 + f2
        if (sum == 2) { sumHelper2(t.tail, res ::: List(0, 0, 1))}
        else sumHelper2(t, res ::: List(sum)) }


  def sumHelper1(f: List[Int]): List[Int] = f match {
    case 1 :: Nil => sumHelper2(List(0, 1), List())
    case 0 :: tail => sumHelper2(1 :: tail, List())
    case _ :: _ :: tail => sumHelper2(0 :: 1 :: tail, List())
  }

  def fib_recur(n: Int): Int = {
    n match {
      case i if i < 2 => i
      case i => fib_recur(n - 1) + fib_recur(n - 2)
    }
  }

  def maxHelper3(l1: List[Int], l2: List[Int], n: Int): Int = {
    if (l2.nonEmpty) {
      if (l1.contains(l2.head) && (l2.head > n))
        l2.head
      else maxHelper3(l1, l2.tail, l2.head)
    } else -1
  }

  def maxHelper2(f: List[Int], res: List[Int]): List[Int] = {
    if (f.nonEmpty) {
      if (f.head == 0) maxHelper2(f.tail, res)
      else maxHelper2(f.tail, res ::: List(fib_recur(f.size + 1)))
    } else res
  }
```

```
  def maxHelper1(f1: List[Int], f2: List[Int]): Int = {
    maxHelper3(maxHelper2(f1, List()), maxHelper2(f2, List()), -1)
  }

  def %(obj: NewFib): Int = {
    maxHelper1(this.list, obj.list)
  }
}

object Main {
  def main(args: Array[String]): Unit = {
    val fibNum54 = new NewFib(List(1, 0, 1, 0, 1, 0, 1, 0))
    val fibNum7 = new NewFib(List(1, 0, 1, 0))
    val fibNum6 = new NewFib(List(1, 0, 0, 1))
    val fibNum46 = new NewFib(List(1, 0, 0, 1, 0, 1, 0, 1))
    val fibNum13 = new NewFib(List(1, 0, 0, 1, 0, 1, 0, 1))

    println("toInteger:")
    println(fibNum54.toInteger)
    println(fibNum7.toInteger)
    println(fibNum13.toInteger)
    println(fibNum6.toInteger)

    println()

    val fibNum20 = new NewFib(List(1, 0, 1, 0, 1, 0))
    val fibNum15 = new NewFib(List(1, 0, 0, 0, 1, 0))
    println(fibNum20.toInteger + " % " + fibNum15.toInteger())
    println(fibNum20 % fibNum15)

    val N1 = new NewFib(List(1, 0, 1, 0))
    val N2 = new NewFib(List(1, 0, 0, 0))
    println(N1.toInteger + " % " + N2.toInteger())
    println(N1 % N2)

  }
}
```

Вывод

В результате выполнения лабораторной работы были изучены базовые принципы программирования на Scala , освоено создание классов в Scala, приобретен опыт использования контейнерных классов из стандартной библиотеки для эффективной работы с данными

Я научилась применять образцы для улучшения читаемости и компактности кода. В целом, выполнение лабораторной работы позволило углубить знания по языку Scala.