Знакомство со Scala

Шестое занятие

Операторы

```
val a = if (flag) 1 else 2 // #1

if (flag) println(“hi”) else println(“bye”) // #2

while (flag) println(“in loop”) // #3

for (elem <- List(1, 2, 3) // #4
     i <- 1 to 10 // #5
     j <- 1 until 10 // #6
     if (i < j) // #7
     k = i * j) println(k)  // #8

for (elem <- List(1, 2, 3)) yield elem * 2 // #9
```

Pattern matching

```
val i = 1
val a = b match {
  case 1 => “a”
  case List(1, 2) => “b”
  case List(1, x) => “c, ” + x
  case List(_, x) => “d, ” + x
  case 1 :: 2 :: tail => “e, ” + tail
  case _ : String => “f”
  case x: Int if (x % 2 == 0) => “g”
  case x@List(_, y@List(1, z)) => “h”
  case `i` => “i”
}
val List(x, y) = List(1, 2)
```

Исключения

```
val s = try {
  Some(new ObjectInputStream(inputStream))
}
catch {
  case e: StreamCorruptedException => println(“bad”); None
  case e: IOException => println(“io”); None
  case e => e.printStackTrace(); throw e
}
finally {
  println(“finally”)
}
```

PartialFunction

```
trait PartialFunction [-A, +B] extends (A) => B

val div = new PartialFunction[Int, Int] {
  override def isDefinedAt(i: Int) = i != 0
  override def apply(i: Int) = 100 / i
}
div.lift(2) // Some(50)
div.lift(0) // None

val f: String => Unit = {case “b” => println(“b”)}
f(“b”) // ok
f(“a”) // exception: scala.MatchError
```

Решение прошлого задания

```
val s = "1111110100100011100100010110000"
val result = s.toList.foldLeft(
  (0, 1, Map[Int, List[Int]]()))
  ({
    case ((level, count, map), '0') =>
      (level - 1, count, map)

    case ((level, count, map), '1') =>
      (level + 1, count + 1, map.updated(
        level, count :: map.get(level).getOrElse(Nil)
      ))
   })

result._3.keys.toList.sorted.foreach(x =>
println(result._3(x).reverse.mkString(" ")))
```

Задание: упрощение выражений

```
trait Expr
case class Num(i: Int) extends Expr
case class Sum(e1: Expr, e2: Expr) extends Expr
case class Mult(e1: Expr, e2: Expr) extends Expr

def simplify(e: Expr): Expr = ???

simplify(Mult(Sum(Num(1), Num(2)), Sum(Num(3), Num(4))))
```