Lembar Kerja Praktikum KOM120C Pemrograman
08: Functional Programming

PETUNJUK PRAKTIKUM

Scala adalah bahasa pemrograman high-level yang mendukung berbagai paradigma, termasuk Functional Programming. Untuk praktikum ini, Scala akan digunakan sebagai contoh untuk membahas konsep-konsep pada paradigma Functional Programming. Scala dapat dijalankan di berbagai IDE atau editor teks lainnya. Namun supaya mudah, bisa menggunakan interpreter online berikut ini: https://onecompiler.com/scala

Variabel

Terdapat dua jenis variabel, mutable (var) dan immutable (val). "var" nilainya bisa diubah, kurang lebih mirip dengan variabel pada bahasa pemrograman pada umumnya. Adapun "val" nilainya konstan, jadi mirip dengan variabel "const" pada Java maupun C++. Pada Functional Programming, yang lebih sering digunakan adalah **val**.

```
var x: Int = 10             // di C++ --> int x = 10;
x = 20                      // boleh

val y: Int = 30             // di C++ --> const int y = 30;
y = 40                      // compile error
```

Scala bisa secara otomatis mengetahui tipe data variabel berdasarkan apa yang kita tulis. Kode di bawah ini sama persis dengan kode di atas.

```
var x = 10                  // di C++ --> int x = 10;
x = 20                      // boleh

val y = 30                  // di C++ --> const int y = 30;
y = 40                      // compile error
```

I/O

Berikut adalah contoh kode untuk mengambil input dari stdin serta melakukan print output menggunakan Scala.

```
val a = scala.io.StdIn.readInt()
val b = scala.io.StdIn.readBoolean()
val c = scala.io.StdIn.readChar()
val d = scala.io.StdIn.readDouble()
val e = scala.io.StdIn.readLine()

println(a)
println(b)
println(c)
println(d)
```

```
println(e)
```

Input	Output
23 true A 3.14 Line of text	23 true A 3.14 Line of text

Structure Control: Conditionals

Sama seperti C++, scala juga menerapkan conditionals statement. Syntax yang bisa digunakan disini adalah “if”, “else if”, “else”. Berikut adalah beberapa contoh kode program yang menggunakan conditionals statement

Contoh 1

```
val x = 10

val result = if (x > 5) {
  "x is greater than 5"
} else if (x == 5) {
  "x is equal to 5"
} else {
  "x is less than 5"
}

println(result)
```

Contoh 2

```
val y = 20

if (x > 5) {
  if (y > 15) {
    println("Both x is greater than 5 and y is greater than 15")
  } else {
    println("x is greater than 5 but y is not greater than 15")
  }
} else {
  println("x is not greater than 5")
}
```

Structure Control: Looping

Sama seperti dalam C++, di Scala kalian juga bisa menerapkan looping statement. Beberapa metode looping yang umum digunakan adalah “for”, “while”, dan “do-while”

1. **For Statement**
   Perulangan for di Scala cukup fleksibel karena digunakan untuk mengulangi collection, range, atau list elemen apa pun. Berikut adalah beberapa contoh penggunaan for statement di Scala

Contoh 1

```
val numbers = List(1, 2, 3, 4, 5)

for (num <- numbers) {
  println(num)
}
```

Contoh 2

```
for (i <- 1 to 5) {
  println(i)
}

for (i <- 1 until 5) {
  println(i)
}
```

2. **While Statement**

   Sama seperti bahasa pemrograman lainnya, while statement akan melakukan pengulangan kondisi statemennya menghasilkan True.

Contoh 1

```
var i = 0
while (i < 5) {
  println(i)
  i += 1
}
```

3. **Do-while Statement**

   Mirip dengan perulangan while, bedanya adalah Do-while statement memeriksa kondisi setelah code dieksekusi, sehingga selalu mengeksekusi code setidaknya satu kali.

Contoh 1

```
var i = 0
do {
  println(i)
  i += 1
} while (i < 5)
```

Generator

Generator adalah cara untuk membuat list yang berisi elemen-elemen yang dihasilkan secara dinamis, biasanya dengan pola yang didefinisikan oleh sebuah aturan atau fungsi. Beberapa contoh generator yang umum digunakan adalah map dan set.

```
//Set Generator
val set1 = Set(1, 2, 3, 4, 5)

for (element <- set1) {
  println(element)
}

//Map Generator
val map = Map("a" -> 1, "b" -> 2, "c" -> 3)

for ((key, value) <- map) {
  println(s"Key: $key, Value: $value")
}
```

Di Scala, kalian bisa menggunakan fitur generator dengan menggunakan metode “yield” dalam perulangan for untuk membuat koleksi baru dengan elemen-elemen yang dihasilkan dalam setiap iterasi.

```
val angka = List(1, 2, 3, 4, 5)

val hasilKuadrat = for {
  angka <- angka
} yield angka * angka

println(hasilKuadrat) // Output: List(1, 4, 9, 16, 25)
```

Function

Berikut adalah contoh kode function di Scala. Di sini, fungsi mengambil input x dan y (keduanya integer) dan me-return tipe integer. Pada fungsi, ekspresi terakhir adalah yang menjadi return value-nya, atau di sini adalah “absX + absY”.

```
def sumAbs(x: Int, y: Int): Int = {
  val absX = if (x>0) x else -x
  val absY = if (y>0) y else -y
  absX + absY
}
println(sumAbs(-5, 3))    // 8
```

Jangan lupa untuk memperhatikan perbedaan antara fungsi murni dan tidak murni. Fungsi murni adalah fungsi yang hasilnya tidak dipengaruhi faktor luar (misalnya ada pengali yang di-declare di luar fungsi) dan tidak memiliki side effect (misalnya melakukan print ke stdout). Pada Functional Programming, fungsi yang digunakan hanya fungsi murni. Dengan menggunakan fungsi murni saja, kode menjadi lebih modular dan antar proses fungsi dapat dijalankan secara paralel.

Contoh kodingan di Scala

Ketika menulis program di Scala, tampilannya kurang lebih seperti berikut. Di sini, "object HelloWorld…" itu mirip seperti "public class…" di Java. Nama "HelloWorld" di sini bisa diganti, tapi biasanya dinamakan sesuai nama file. Berikutnya, ada fungsi main, dan kode yang ingin kita jalankan ditulis di fungsi main ini. Pada fungsi main, return value-nya adalah "Unit" atau kalau dalam bahasa pemrograman lain biasanya disebut void.

```
object HelloWorld {
  def main(args: Array[String]): Unit = {
    // code here
  }
}
```

Berikut contoh kode yang menerapkan fungsi di atas.

```
object HelloWorld {
  def main(args: Array[String]): Unit = {
    def sumAbs(x: Int, y: Int): Int = {
      val absX = if (x>0) x else -x
      val absY = if (y>0) y else -y
      absX + absY
    }
    println(sumAbs(-5, 3))    // 8
  }
}
```

Berikut contoh kode lain untuk dipelajari.

Contoh kode program mencari nilai minimum dari list

```
import scala.io.StdIn.readLine

// Membaca banyaknya anggota list dari input
val n = readLine().toInt

// Membaca anggota-anggota list dari input dan menyimpannya dalam list
var numbers = List[Int]()
for (_ <- 1 to n) {
  // Membaca anggota list satu per satu dari input
  val num = readLine().toInt
  numbers = numbers :+ num
}

// Mencari nilai minimum dari list
var minNumber = Int.MaxValue
for (num <- numbers) {
  if (num < minNumber) {
    minNumber = num
  }
}

// Mencetak nilai minimum ke output
println(minNumber)
```

TUGAS

1. Buat program untuk membaca input berupa N buah bilangan bulat, kemudian carilah nilai terkecil, terbesar dan jumlah dari semua bilangan yang ganjil.

Contoh Input
5 1 -1 -9 7 5
Contoh Output
-9 7 3

```
object Main {
  def main(args: Array[String]): Unit = {
    val n = scala.io.StdIn.readInt() // Membaca jumlah bilangan

    var smallest = Int.MaxValue
    var largest = Int.MinValue
    var sumOfOdd = 0


    for (i <- 1 to n) {
      val num = scala.io.StdIn.readInt() // Membaca bilangan dari pengguna

      // Memperbarui nilai terkecil dan terbesar
      if (num < smallest)
        smallest = num
      if (num > largest)
        largest = num

      // Memeriksa apakah bilangan ganjil dan menambahkan ke jumlah bilangan
ganjil
      if (num % 2 != 0)
        sumOfOdd += num
    }

    // Mencetak hasil
    println(s"$smallest")
    println(s"$largest")
    println(s"$sumOfOdd")
  }
```

Input:

```
5
1
-1
-9
7
5
```

Output:

```
Nilai terkecil:-9
Nilai terbesar:7
Banyak bilangan ganjil:3
```

2. Buat program untuk membaca dua buah array berukuran sama (N) masing-masing bisa positif/negatif. Kemudian hitunglah hasil perkalian antara nilai mutlak terbesar dari array pertama dengan nilai mutlak terbesar dari array kedua.

Contoh Input
5 1 -1 -9 7 5 20 7 -32 2 50
Contoh Output
450

```
object dragon {
  import scala.io.StdIn

  def main(args: Array[String]): Unit = {
    val N = StdIn.readInt()

    val array1 = readArray(N)

    val array2 = readArray(N)

    val maxAbsValue1 = array1.map(math.abs).max
    val maxAbsValue2 = array2.map(math.abs).max

    val result = maxAbsValue1 * maxAbsValue2
    println(s"$result")
  }

  def readArray(N: Int): Array[Int] = {
    val input = StdIn.readLine().split(" ").map(_.toInt)
    if (input.length != N) {
      println(s"$N")
      readArray(N)
    } else {
      input
    }
  }
```

Input:

```
5
1 -1 -9 7 5
20 7 -32 2 50
```

Output:

```
450
```

```
5
1 -1 -9 7 5
20 7 -32 2 50
450
```

3. Buatlah sebuah program yang menggunakan rekursi untuk menghitung nilai faktorial dari n, di mana n adalah bilangan bulat positif.

Contoh Input
5
Contoh Output
120

```
object Main {
  def main(args: Array[String]): Unit = {
    // Membaca input bilangan bulat positif
    val n = scala.io.StdIn.readInt()

    // Memanggil fungsi faktorial dan mencetak hasilnya
    println(factorial(n))
  }

  // Fungsi rekursif untuk menghitung faktorial
  def factorial(n: Int): Int = {
    if (n <= 1) 1 // Basis dari rekursi: faktorial dari 0 dan 1 adalah 1
    else n * factorial(n - 1) // Langkah rekursif: n! = n * (n-1)!
  }
}
```

Input/Output:

```
5
120
```