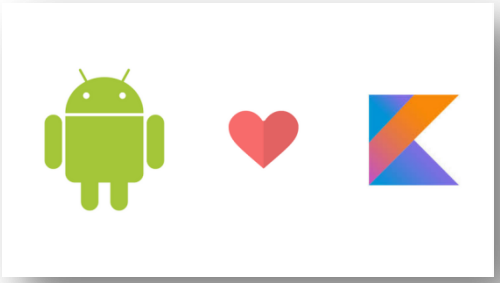
Pemrograman Kotlin Lanjutan



Praktikum 2

Tujuan :

Mahasiswa mengenal dan mengimplementasikan :

- Array (Tipe primitive, 2 Dimensi, Multi Dimensi)
- Collections (List, Set, Map)
- Ranges
- Percabangan (If, If-Else, If-Else-If, When)
- Perulangan (For, While, Do-While, Break, Continue, Return, Label)
- Null-Safety
- Penanganan Eksepsi (*Exception Handling*)
- Fungsi Math
- Fungsi dan Lambdas

Hasil Praktikum :

- Contoh program praktikum.

Array

Array adalah koleksi data yang sejenis. Untuk membuat sebuah array bisa menggunakan fungsi arrayOf() dan melewatkan nilai item kepada array tersebut, jadi arrayOf(1,2,3) berarti membuat array dengan item [1,2,3]. Alternative lain untuk membuat array dengan ukuran tertentu bisa dilakukan dengan fungsi arrayOfNulls().

Penggunaan arrayOf() untuk membuat array:

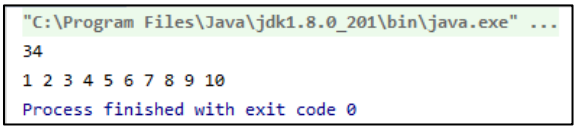
```
fun main(args: Array<String>) {
```

```
    //Array
    var arrA = arrayOf<Int>()
    arrA += 34
    println(arrA[0])

    var arrB = arrayOf<Int>()
    for (i in 1..10) {
        arrB += i
    }

    for (i in arrB) {
        print("$i ")
    }
}
```

Output:



Berikut adalah Library Array dan beberapa fungsi anggota seperti size, get(), set() yang ada di Kotlin:

```
class Array<T> private constructor() {
        val size: Int
        operator fun get(index: Int): T
        operator fun set(index: Int, value: T): Unit
        operator fun iterator(): Iterator<T>
        // ...
}
```

Perhatikan contoh pembuatan array berdasarkan template Array standard Kotlin diatas:

```
fun main(args: Array<String>) {
    // Creates an Array<String> with sizes 5 and values ["0", "1", "4", "9", "16"]
    val asc = Array(5) {
            i -> (i * i).toString()
    }
    asc.forEach { println(it) }
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
0
1
4
9
16

Process finished with exit code 0
```

Array dengan tipe primitive

Kotlin memiliki kelas khusus untuk membuat array dengan tipe primitive tanpa adanya fitur boxing seperti ByteArray , ShortArray , IntArray dan banyak lagi.

Berikut contoh penggunaan Array dengan tipe primitive:

```
fun main(args: Array<String>) {
    val x: IntArray = intArrayOf(1, 2, 3)
    x[0] = x[1] + x[2]
    x.forEach { print(it.toString() + ' ') }
    print('\n')

    // Array of int of size 5 with values [0, 0, 0, 0, 0]
    val byteArray = ByteArray(5)
    byteArray.forEach { print(it.toString() + ' ') }
    print('\n')
    // e.g. initialise the values in the array with a constant
    // Array of int of size 5 with values [42, 42, 42, 42, 42]
    val shortArray = ShortArray(5) { 42 }
    shortArray.forEach { print(it.toString() + ' ') }
    print('\n')
    // e.g. initialise the values in the array using a lambda
    // Array of int of size 5 with values [0, 1, 2, 3, 4] (values initialised to their index value)
    var intArray = IntArray(5) { it * 1 }
    intArray.forEach { print(it.toString() + ' ') }
    print('\n')
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
5 2 3
0 0 0 0 0
42 42 42 42 42
0 1 2 3 4

Process finished with exit code 0
```

Array Method

```
fun main(args: Array<String>) {
    //Array methods
    //sort()
```

```
    val simpsonsA = arrayOf("Homer", "Marge", "Bart", "Lisa", "Meggie")
    simpsonsA.sort()
    for (simpson in simpsonsA) {
        print(simpson + ' ')
    }
    print('\n')

    //sorted()
    val simpsonsB: Array<String> = arrayOf("Homer", "Marge", "Bart", "Lisa", "Meggie")
    val simpsons_sorted = simpsonsB.sortedArray()
    for (simpson in simpsons_sorted) {
        print(simpson + ' ')
    }
    print('\n')

    //reverse() and reversedArray()
    val simpsonsC: Array<String> = arrayOf("Homer", "Marge", "Bart", "Lisa", "Meggie")
    simpsonsC.sort()
    simpsonsC.reverse()
    for (simpson in simpsonsC) {
        print(simpson + ' ')
    }
    print('\n')

    //indexOf()
    val simpsonsD = arrayOf("Homer", "Marge", "Bart", "Lisa", "Meggie")
    val simpson = "Lisa"
    val position = simpsonsD.indexOf(simpson)
    if (position != -1) {
        println("Yeah, that's my number ${position + 1} Simpson!")
    } else {
        println("Hey that's not Simpson!")
    }

    val simpsonsE = arrayOf("Homer", "Marge", "Bart", "Lisa", "Meggie")
    println("simpsons size: " + simpsonsE.size)
    if(!simpsonsE.isEmpty())
        println("Array tidak kosong!")
    println("Min = "+ simpsonsE.min())
    println("Max = "+ simpsonsE.max())
    println("First = " + simpsonsE.first())
    println("Last = " + simpsonsE.last())
    println(simpsonsE.contains("Marge"))
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Bart Homer Lisa Marge Meggie
Bart Homer Lisa Marge Meggie
Meggie Marge Lisa Homer Bart
Yeah, that's my number 4 Simpson!
simpsons size: 5
Array tidak kosong!
Min = Bart
Max = Meggie
First = Homer
Last = Meggie
true

Process finished with exit code 0
```

Array 2 Dimensi

```
fun main(args: Array<String>){
    //initialize a 2D array
    var cinema = arrayOf<Array<Int>>()
    for (i in 0..4) {
        var array = arrayOf<Int>()
        for (j in 0..4) {
            array += 0
        }
        cinema += array
    }

    for (array in cinema) {
        for (value in array) {
            print("$value ")
        }
        println()
    }

    //Filling with data
    cinema[2][2] = 1
    for (i in 1..3) {
        cinema[3][i] = 1
    }
    for (i in 0..4) {
        cinema[4][i] = 1
    }

    println()
    for (array in cinema) {
        for (value in array) {
            print("$value ")
        }
        println()
```

```
    }
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0

0 0 0 0 0
0 0 0 0 0
0 0 1 0 0
0 1 1 1 0
1 1 1 1 1

Process finished with exit code 0
```

Array Multidimensi

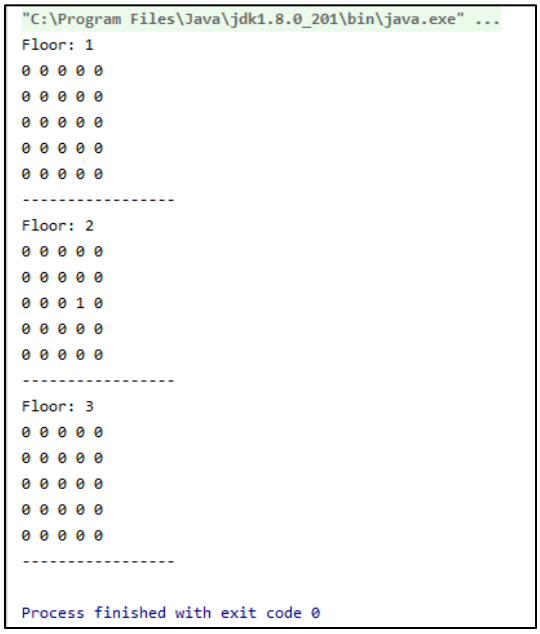
```
fun main(args: Array<String>) {
    //N-dimensional arrays
    var cinemas = arrayOf<Array<Array<Int>>>()
    for (i in 0..2) {
        var cinema = arrayOf<Array<Int>>()
        for (j in 0..4) {
            var array = arrayOf<Int>()
            for (k in 0..4) {
                array += 0
            }
            cinema += array
        }
        cinemas += cinema
    }

    cinemas[1][2][3] = 1 // the second-floor cinema, the third row, the fourth seat

    var floor = 1
    for (cinema in cinemas) {
        println("Floor: $floor")
        floor += 1
        for (array in cinema) {
            for (value in array) {
                print("$value ")
            }
            println()
        }
        println("-----------------")
    }
```

```
}
```

Output:



Collections

Koleksi biasanya berisi sejumlah objek (mungkin berjumlah 0) yang memiliki tipe yang sama. Objek dalam sebuah koleksi dipanggil dengan elemen atau item. Jenis koleksi di Kotlin terdiri dari:

1) **List** : adalah sebuah koleksi terurut yang dapat menyimpan elemen yang sama lebih dari satu kali. Contohnya kelompok kata dalam sebuah kalimat.
2) **Set** : adalah koleksi yang memiliki elemen yang unik. Contohnya set alpabhet.
3) **Map** (atau kamus) : adalah sebuah set pasangan kunci dan nilai (key-value). Kunci bersifat unik dan hanya memiliki satu nilai. Sedangkan nilai yang ada pada Map dapat bernilai ganda/sama.

Menurut sifatnya jenis koleksi terbagi menjadi 2 yaitu :

- A read-only interface yang menyediakan operasi untuk mengakses elemen.
- A mutable interface adalah perluasan dari read-only interface yang berkorespodensi dengan operasi : menambahkan, menghapus, dan mengubah elemen.

Koleksi di Kotlin dan Fungsi yang dibuat

Collection	**Read-Only**	**Mutable**
List	listOf	mutableListOf, arrayListOf

Set	setOf	mutableSetOf, hashSetOf, linkedSetOf, sortedSetOf
Map	mapOf	mutableMapOf, hashMapOf, linkedMapOf, sortedMapOf

Contoh Collections

```
fun printAll(strings: Collection<String>) {
    for(s in strings) print("$s ")
    println()
}

fun main(args: Array<String>) {
    val stringList = listOf("one", "two", "one")
    printAll(stringList)

    val stringSet = setOf("one", "two", "three")
    printAll(stringSet)
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
one two one
one two three

Process finished with exit code 0
```

List

listOf

```
fun main(args:Array<String>){
    val numbers = listOf("one", "two", "three", "four")
    println("Number of elements: ${numbers.size}")
    println("Third element: ${numbers.get(2)}")
    println("Fourth element: ${numbers[3]}")
    println("Index of element \"two\" ${numbers.indexOf("two")}")

    //List elements (including nulls) can duplicate
    val bob = Person("Bob", 31)
    val people = listOf<Person>(Person("Adam", 20), bob, bob)
    val people2 = listOf<Person>(Person("Adam", 20), Person("Bob", 31), bob)
    println(people == people2)
    bob.age = 32
    println(people == people2)
}

class Person(var name: String, var age: Int){}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Number of elements: 4
Third element: three
Fourth element: four
Index of element "two" 1
false
false

Process finished with exit code 0
```

mutableListOf

```
fun main(args: Array<String>) {
    //MutableList is a List with list-specific write operations
    val numbers = mutableListOf(1, 2, 3, 4)
    numbers.add(5)
    numbers.removeAt(1)
    numbers[0] = 0
    numbers.shuffle()
    println(numbers)

    val numbersStr = mutableListOf("one", "two", "three", "four")
    numbersStr.add("five")   // this is OK
    //numbersStr = mutableListOf("six", "seven")      // compilation error
    println(numbersStr)
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
[0, 4, 3, 5]
[one, two, three, four, five]

Process finished with exit code 0
```

arrayListOf

```
fun main(args: Array<String>) {
    //Array lists (mutable) bisa diubah
    /*
    Method:
        clear(), contains(), size, get() or [], add() or +=, remove() or -=,
        removeAt(), toTypedArray(), lastIndexOf(), removeAll(), reverse()
     */
    val arrayList = arrayListOf(1, 2, 3)
    print("ArrayList array : ")
    for(item in arrayList){
        print("$item ")
    }
    print("\n")

    arrayList += 9
```

```
    print("Tambahkan elemen 9 dalam arrayList : ")
    println(arrayList)

    arrayList.reverse()
    print("Reverse arrayList : ")
    println(arrayList)

    arrayList -= 1 // We can delete!
    print("Remove last element in arrayList : ")
    println(arrayList)

    arrayList += (1..5)
    print("Add range in arrayList : ")
    println(arrayList)
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
ArrayList array : 1 2 3
Tambahkan elemen 9 dalam arrayList : [1, 2, 3, 9]
Reverse arrayList : [9, 3, 2, 1]
Remove last element in arrayList : [9, 3, 2]
Add range in arrayList : [9, 3, 2, 1, 2, 3, 4, 5]

Process finished with exit code 0
```

Set

```
fun main() {
    //Set<T> stores unique elements; their order is generally undefined.
    val numbers = setOf(1, 2, 3, 4)
    println("Number of elements: ${numbers.size}")
    if (numbers.contains(1)) println("1 is in the set")

    val numbersBackwards = setOf(4, 3, 2, 1)
    println("The sets are equal: ${numbers == numbersBackwards}")

    //The default implementation of Set – LinkedHashSet – preserves the order of elements insertion
    println(numbers.first() == numbersBackwards.first())
    println(numbers.first() == numbersBackwards.last())

    val strings = hashSetOf("a", "b", "c", "c") //hash set value unik
    println("My Set Values are"+strings)
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Number of elements: 4
1 is in the set
The sets are equal: true
false
true
My Set Values are[a, b, c]

Process finished with exit code 0
```

Map

```
fun main(){
    //Map<K, V> is not an inheritor of the Collection interface;
    /*
    Map stores key-value pairs (or entries);
    keys are unique, but different keys can be paired with equal values.
     */
    val numbersMap = mapOf("key1" to 1, "key2" to 2, "key3" to 3, "key4" to 1)
    println("All keys: ${numbersMap.keys}")
    println("All values: ${numbersMap.values}")
    if ("key2" in numbersMap) println("Value by key \"key2\": ${numbersMap["key2"]}")
    if (1 in numbersMap.values) println("The value 1 is in the map")
    if (numbersMap.containsValue(1)) println("The value 1 is in the map") // same as previous

    //wo maps containing the equal pairs are equal regardless of the pair order.
    val anotherMap = mapOf("key2" to 2, "key1" to 1, "key4" to 1, "key3" to 3)
    println("The maps are equal: ${numbersMap == anotherMap}")

    //MutableMap is a Map with map write operations, for example,
    //you can add a new key-value pair or update the value associated with the given key
    val mutableMap = mutableMapOf("one" to 1, "two" to 2)
    mutableMap.put("three", 3)
    mutableMap["one"] = 11
    println(mutableMap)

    val readWriteMap = hashMapOf("foo" to 1, "bar" to 2)
    println(readWriteMap["foo"])  // prints "1"
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
All keys: [key1, key2, key3, key4]
All values: [1, 2, 3, 1]
Value by key "key2": 2
The value 1 is in the map
The value 1 is in the map
The maps are equal: true
{one=11, two=2, three=3}
1

Process finished with exit code 0
```

Ranges

Ranges adalah karakteristik unik dari Kotlin. Seperti python, ranges menyediakan sebuah operator yang dapat membantu kita dalam membuat sebuah deretan angka melalui suatu iterasi.

Contoh kode program:

```
fun main(args: Array<String>) {
    //Ranges
    val i:Int = 2
    for (j in 1..4)
        print(j) // prints "1234"
    println()

    if (i in 1..10) { // equivalent of 1 <= i && i <= 10
        println("we found your number --"+i)
    }
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
1234
we found your number --2

Process finished with exit code 0
```

Percabangan

Percabangan IF, IF-ELSE, IF-ELSE IF

Contoh kode program 1 :

```
fun main(args: Array<String>) {
    //Conditions
    if (15 > 5)
        println("True")
    println("The program continues here...")
```

```
    //If - Else
    val a:Int = 5
    val b:Int = 2
    var max: Int

    if (a > b) {
        max = a
    } else {
        max = b
    }
    println("Maximum of a or b is " + max)

    //If - else if
    var myVar: Int = 0 // the variable is initialized with a value of 0
    if (myVar == 0) { // if the value is 0, we change its value to 1
        myVar = 1
    } else if(myVar == 1){ // if the value is 1, we change its value to 0
        myVar = 0
    } else {
        myVar = -1
    }
    println("Nilai myVar adalah " + myVar)

    print("\nEnter a number : ")
    val intNumber = readLine()!!.toInt()
    if (intNumber > 5)
        println("The number you entered is greater than 5!")
    println("Thanks for the input!")
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
True
The program continues here...
Maximum of a or b is 5
Nilai myVar adalah 1

Enter a number : 12
The number you entered is greater than 5!
Thanks for the input!

Process finished with exit code 0
```

Contoh kode program 2 :

```
fun main(args: Array<String>) {
    println("Enter a number between 10-20:")
    val a = readLine()!!.toInt()
    if (a >= 10 && a <= 20) {
        println("The condition has been met.")
    } else {
        println("You did it wrong.")
    }

    println("\nEnter a number between 10-20 or 30-40:")
    val b = readLine()!!.toInt()
    if (((b >= 10) && (b <= 20)) || ((b >=30) && (b <= 40))) {
        println("The condition has been met.")
    } else {
        println("You did it wrong.")
    }
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Enter a number between 10-20:
15
The condition has been met.

Enter a number between 10-20 or 30-40:
25
You did it wrong.

Process finished with exit code 0
```

Kalkulator Sederhana Menggunakan Percabangan IF-ELSE IF

Kode program :

```
fun main(args: Array<String>) {
    println("Welcome to our calculator")
    println("Enter the first number:")
    val a = readLine()!!.toDouble()
    println("Enter the second number:");
    val b = readLine()!!.toDouble()
    println("Choose one of the following operations:")
    println("1 - addition")
    println("2 - subtraction")
    println("3 - multiplication")
    println("4 - division")
    val choice = readLine()!!.toInt()
    var result = 0.0
    if (choice == 1) {
        result = a + b
```

```
    } else if (choice == 2) {
        result = a - b
    } else if (choice == 3) {
        result = a * b
    } else if (choice == 4) {
        result = a / b
    }
    if ((choice > 0) && (choice < 5)) {
        println("result: $result")
    } else {
        println("Invalid choice")
    }
    println("Thank you for using our calculator.")
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Welcome to our calculator
Enter the first number:
12
Enter the second number:
13
Choose one of the following operations:
1 - addition
2 - subtraction
3 - multiplication
4 - division
1
result: 25.0
Thank you for using our calculator.

Process finished with exit code 0
```

When

When adalah operator pengganti switch-case di bahasa pemrograman C.

Contoh kode program :

```
fun main(args: Array<String>) {
    //Use of When
    var x:Int = 5
    when (x) {
        1 -> println("x == 1")
        2 -> println("x == 2")
        else -> { // Note the block
            println("x is neither 1 nor 2")
        }
    }

    x = 2
    when (x) {
```

```
        1,2 -> println("Value of X either 1,2")
        else -> { // Note the block
            println("x is neither 1 nor 2")
        }
    }
}
```

Output :

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
x is neither 1 nor 2
Value of X either 1,2

Process finished with exit code 0
```

Kalkulator Sederhana Menggunakan Percabangan When

Kode Program :

```
fun main(args: Array<String>) {
    println("Welcome to our calculator")
    println("Enter the first number:");
    val a = readLine()!!.toDouble()
    println("Enter the second number:")
    val b = readLine()!!.toDouble()
    println("Choose one of the following operations:")
    println("1 - addition")
    println("2 - subtraction")
    println("3 - multiplication")
    println("4 - division")
    val choice = readLine()!!.toInt()
    var result = 0.0
    when (choice) {
        1 -> result = a + b
        2 -> result = a - b
        3 -> result = a * b
        4 -> result = a / b
    }
    if ((choice > 0) && (choice < 5)) {
        println("Result: $result")
    } else {
        println("Invalid choice")
    }
    println("Thank you for using our calculator.")
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Welcome to our calculator
Enter the first number:
4
Enter the second number:
6
Choose one of the following operations:
1 - addition
2 - subtraction
3 - multiplication
4 - division
4
Result: 0.6666666666666666
Thank you for using our calculator.

Process finished with exit code 0
```

Perulangan

For Loop

Kotlin tidak memiliki tradisional *for-loop* seperti di Java 7 dan dibawahnya. Tradisional *for-loop* adalah seperti contoh berikut:

```
for (int i = 0; i < 10; i++) {
        statements
}
```

For-loop di Kotlin menggunakan iterator untuk mengakses elemen. Mirip dengan konsep *for-each* di JavaScript, C# atau Java 8.

Template :

```
for (variable in group){
        statements to access variable
}
```

Berikut adalah contoh program perulangan menggunakan *for-loop*.

```
fun main(args: Array<String>) {
    //The for loop
    for (i in 1..10) {
        print("$i ")
    }
    print('\n')

    print('\n')
    val arrInt = arrayOf(1, 2, 3, 4)
    for (i in arrInt) println("values of the array " +i)
```

```
    print('\n')
    val listInt = listOf(1, 22, 83, 4)
    for ((index, value) in listInt.withIndex()) {
        println("the element at $index is $value")
    }
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
1 2 3 4 5 6 7 8 9 10

values of the array 1
values of the array 2
values of the array 3
values of the array 4

the element at 0 is 1
the element at 1 is 22
the element at 2 is 83
the element at 3 is 4

Process finished with exit code 0
```

Penerapan For-Loop dalam Tabel Perkalian

Kode program:

```
fun main(args: Array<String>) {
    println("Simple multiplication table using loops:")
    for (i in 1..10) {
        print("$i ")
    }
    println()
    for (i in 1..10) {
        print("${i * 2} ")
    }
    println()
    for (i in 1..10) {
        print("${i * 3} ")
    }
    println()
    for (i in 1..10) {
        print("${i * 4} ")
    }
    println()
    for (i in 1..10) {
        print("${i * 5} ")
    }
    println()
    for (i in 1..10) {
```

```
        print("${i * 6} ")
    }
    println()
    for (i in 1..10) {
        print("${i * 7} ")
    }
    println()
    for (i in 1..10) {
        print("${i * 8} ")
    }
    println()
    for (i in 1..10) {
        print("${i * 9} ")
    }
    println()
    for (i in 1..10) {
        print("${i * 10} ")
    }

    println("\n\nHere's a simple multiplication table using nested loops:")
    for (i in 1..10) {
        for (j in 1..10) {
            print("${i * j} ")
        }
        println()
    }
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Simple multiplication table using loops:
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100

Here's a simple multiplication table using nested loops:
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100

Process finished with exit code 0
```

Penerapan For-Loop dalam Kalkulator Eksponensial

Kode program:

```
fun main(args: Array<String>) {
    println("Exponent calculator")
    println("===================")
    println("Enter the base: ")
    val a = readLine()!!.toInt()
    println("Enter the exponent: ")
    val n = readLine()!!.toInt()
    var result = a
    for (i in 1..n-1) {
        result = result * a
    }
    println("Result: $result")
    println("Thank you for using our exponent calculator")
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Exponent calculator
==================
Enter the base:
2
Enter the exponent:
3
Result: 8
Thank you for using our exponent calculator

Process finished with exit code 0
```

While dan Do-While Loop

While dan do-while loop di Kotlin sama penggunaannya dengan bahasa pemrograman Java.

```
fun main(args: Array<String>) {
    //While Loop
    var x:Int = 0
    println("Example of While Loop--")
    while(x <= 10) {
        print("$x ")
        x++
    }
    print('\n')

    //Do-While Loop
    var y:Int = 0
    do {
        y = y + 10
        println("I am inside Do block---" +y)
    } while(y <= 50)
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Example of While Loop--
0 1 2 3 4 5 6 7 8 9 10
I am inside Do block---10
I am inside Do block---20
I am inside Do block---30
I am inside Do block---40
I am inside Do block---50
I am inside Do block---60

Process finished with exit code 0
```

Penerapan While-Loop dalam Kalkulator Sederhana

Kode program:

```
fun main(args: Array<String>) {
```

```
    println("Welcome to our calculator")
    var goOn = "yes"
    while (goOn == "yes") {
        println("Enter the first number")
        val a = readLine()!!.toDouble()
        println("Enter the second number:")
        val b = readLine()!!.toDouble()
        println("Choose one of the following operations:")
        println("1 - addition")
        println("2 - subtraction")
        println("3 - multiplication")
        println("4 - division")
        val choice = readLine()!!.toInt()
        var result: Double = 0.0
        when (choice) {
            1 -> result = a + b
            2 -> result = a - b
            3 -> result = a * b
            4 -> result = a / b
        }
        if ((choice > 0) && (choice < 5)) {
            println("Result: $result")
        } else {
            println("Invalid choice")
        }
        println("Would you like to make another calculation? [yes/no]")
        goOn = readLine()!!
    }
    println("Thank you for using our calculator.")
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Welcome to our calculator
Enter the first number
12
Enter the second number:
13
Choose one of the following operations:
1 - addition
2 - subtraction
3 - multiplication
4 - division
3
Result: 156.0
Would you like to make another calculation? [yes/no]
no
Thank you for using our calculator.

Process finished with exit code 0
```

Break, Continue, Return dan Label

Kotlin mendukung tradisional *break* dan *continue*, sama seperti bahasa pemrograman Java. Kegunaan dari *break* dan *continue* adalah :

break : untuk menghentikan proses perulangan ke penutup terdekat dalam suatu blok program.

continue : proses untuk melanjutkan perulangan ke penutup terdekat dalam suatu blok program.

Berikut adalah contoh kode program penggunaan *break* dan *continue* pada sebuah perulangan :

```
fun main(args: Array<String>) {
    var x:Int
    println("Example of Break and Continue in For-Loop")
    for(x in 1..10) {
        if(x == 7) break
        if(x == 3) continue
        print("$x ")
    }
    print('\n')
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Example of Break and Continue in For-Loop
1 2 4 5 6

Process finished with exit code 0
```

Keyword *return* berguna untuk keluar dari fungsi dan mengembalikan suatu nilai kepada pemanggil fungsi.

Berikut kode program *return* pada perulangan *for-loop*:

```
fun main(args: Array<String>) {
    returnLoop()
}

fun returnLoop(){
    var x:Int
    println("Example of return in For-Loop")
    for(x in 1..10) {
        if(x < 5) {
            print("${x - 1} ")
        }else {
            return
        }
        print("$x ")
    }
    print("Tidak akan pernah dieksekusi")
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Example of return in For-Loop
0 1 1 2 2 3 3 4
Process finished with exit code 0
```

Apapun ekspresi di Kotlin dapat ditandai dengan label. Label memiliki bentuk sebuah identifier yang diikuti oleh tanda @ sebagai contoh : abc@, foo@, loop@.

Berikut kode program *break* dan *continue label* pada perulangan *for-loop*:

```
fun main(args: Array<String>) {
    //Continue & Break Label
    println("Example of Break and Continue Label")
    myLabel@ for(x in 1..10) { // appling the custom label
        if(x == 5) {
            println("I am inside if block with value"+x+"\n-- hence it will close the operation")
            break@myLabel //specifing the label
        } else {
            println("I am inside else block with value"+x)
            continue@myLabel
        }
        println("Not Print")
    }
}
```

Output:

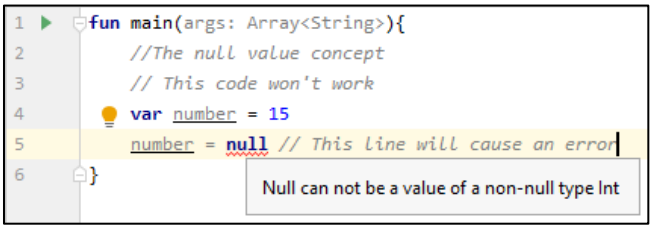
```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Example of Break and Continue Label
I am inside else block with value1
I am inside else block with value2
I am inside else block with value3
I am inside else block with value4
I am inside if block with value5
-- hence it will close the operation

Process finished with exit code 0
```

Null-Safety

Salah satu kesalahan yang banyak ditemukan pada bahasa pemrograman adalah pengaksesan terhadap anggota referensi null dan akan menghasilkan pesan kesalahan *null reference exception*, di Java sering disebut dengan Null Pointer Exception (NPE). Kotlin type system bertujuan untuk menghilangkan NPE dari kode program.

Contoh pemberian nilai null pada sebuah variabel:



Kotlin tidak mengizinkan pemberian nilai *null* secara tidak sah. Untuk dapat memberikan nilai *null* pada suatu variabel dapat dilakukan dengan cara menambahkan tanda ? diakhir tipe data. Berikut contoh pemberian nilai *null* yang diizinkan:

```
fun main(args: Array<String>){
    //Nullable types
    var maybeNumber: Int? = 15
    maybeNumber = null
    println("maybeNumber: $maybeNumber ${maybeNumber?.hashCode()}")
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
maybeNumber: null null

Process finished with exit code 0
```

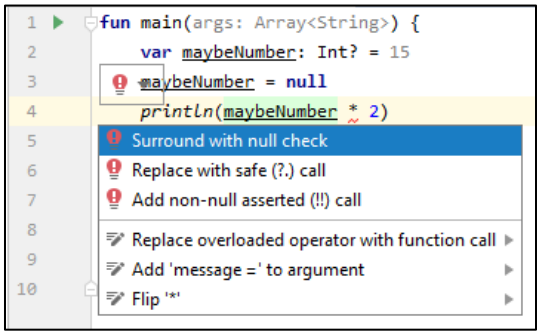
Operator !!

Perhatikan kesalahan kode program berikut:

```
fun main(args: Array<String>) {
    var maybeNumber: Int? = 15
    maybeNumber = null
    println(maybeNumber * 2)

    var s1 = "Hello"
    var s2: String? = "World"
    println(s1.length)
    println(s2.length)
}
```

Output:



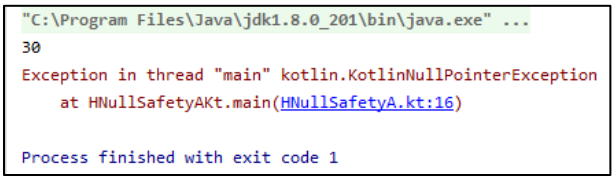
Kita tidak dapat bekerja dengan variabel maybeNumber dan s2 seperti biasanya karena kedua variabel tersebut mungkin saja memiliki nilai null. Untuk memanggil dan menggunakannya dengan aman, maka tambahkan operator !! yang fungsinya melewatkan pemeriksaan nilai null.

Contoh kode program :

```
fun main(args: Array<String>) {
    //Null safety
    //The !! operator
    var maybeNumber: Int? = 15
    println(maybeNumber!! * 2)

    maybeNumber = null
    println(maybeNumber!! * 2)
}
```

Output:



Pemeriksaan Kondisi Null

Contoh kode program:

```
fun main(args: Array<String>) {
    //Conditions
    var maybeNumber: Int? = 15
    if (maybeNumber != null)
        println(maybeNumber * 2)
    else
        println("The entered value isn't a number")
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
30

Process finished with exit code 0
```

Pemanggilan yang aman (Safe Calls)

Contoh kode program:

```
fun main(args: Array<String>) {
    //Safe calling
    //?.let
    var maybeNumber: Int? = 15
    maybeNumber?.let { println(it) }

    //The ?. chaining
    //student?.teacher?.supervisor?.name = "Seymour Skinner"
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
15

Process finished with exit code 0
```

Elvis Operator (?:)

Ketika kita memiliki sebuah null referensi ‘r’, kita mungkin akan berpikir jika ‘r’ tidak null maka kita gunakan, dengan kata lain kita akan menggunakan nilai non null ‘x’

val l: Int = if (b != null) b.length else -1

kode ini bisa disederhanakan dengan menggunakan operator elvis “?:” sehingga menjadi;

val l = b?.length ?: -1

Contoh kode program:

```
fun main(args: Array<String>) {
    //Elvis Operator ?:
    var maybeWelcome: String? = "Hello world"
```

```
    println(maybeWelcome?.length ?: 0)
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
11

Process finished with exit code 0
```

Penanganan Eksepsi (Exception Handling)

Penanganan eksepsi adalah bagian penting dari bahasa pemrograman. Teknik ini menghalangi aplikasi kita memberikan output yang salah saat runtime. Penggunaan eksepsi di Kotlin sama seperti di Java. Semua eksepsi adalah turunan dari kelas “Throwable”. Berikut adalah contoh kode program penanganan eksepsi:

```
fun main(args: Array<String>) {
    try {
        val myVar:Int = 10;
        val v:String = "Rekayasa Perangkat Lunak";
        v.toInt();
    } catch(e:Exception) {
        e.printStackTrace()
    } finally {
        println("Exception Handeling in Kotlin");
    }
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
java.lang.NumberFormatException: For input string: "Rekayasa Perangkat Lunak"
    at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
    at java.lang.Integer.parseInt(Integer.java:580)
    at java.lang.Integer.parseInt(Integer.java:615)
    at IExceptionKt.main(IException.kt:5)
Exception Handeling in Kotlin

Process finished with exit code 0
```

Fungsi Math

Untuk dapat menggunakan fungsi math terlebih dahulu import paket math berikut di baris atas kode program: *import kotlin.math.**

Contoh kode program fungsi math:

```
import kotlin.math.*

fun main(args: Array<String>) {
    //Constants
```

```
    println("Pi: ${PI}\ne: ${E}")
    //Available math functions
    //min(), max(), round(), ceil(), floor() and truncate()
    var varDouble = 2.72
    var roundResult: Int = round(varDouble).toInt()
    println("Pembulatan round $varDouble adalah " + roundResult)
    //abs() and sign()
    //sin(), cos(), tan(), acos(), asin(), atan()
    //pow() and sqrt()
    println("Nilai 2 pangkat 3 adalah " + 2.0.pow(3))
    //exp(), log(), log10(), log2()
    //8 = 8^(1/3)
    println("Nilai 8 pangkat 1/3 adalah " + 8.0.pow((1.0 / 3.0)))
    //Division
    val a = 5 / 2
    val b = 5.0 / 2
    val c = 5 / 2.0
    val d = 5.0 / 2.0
    // val e: Int = 5 / 2.0
    // val f: Double = 5 / 2
    print("$a\n$b\n$c\n$d")
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Pi: 3.141592653589793
e: 2.718281828459045
Pembulatan round 2.72 adalah 3
Nilai 2 pangkat 3 adalah 8.0
Nilai 8 pangkat 1/3 adalah 2.0
2
2.5
2.5
2.5
Process finished with exit code 0
```

Menghitung Luas dan Keliling Lingkaran

```
import kotlin.math.PI

fun main(args: Array<String>) {
    //Advanced exercise
    print("Enter the circle's radius (cm): ")
    val radius = readLine()!!.toDouble()
    var circumference = 2 * PI * radius
    var area = PI * radius * radius
    println("The circle's circumference based on the given radius is: " + circumference + " cm")
    println("Area of the circle is: " + area + " cm^2")
```

```
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Enter the circle's radius (cm): 7
The circle's circumference based on the given radius is: 43.982297150257104 cm
Area of the circle is: 153.93804002589985 cm^2

Process finished with exit code 0
```

Menghitung Akar Kuadrat

```
import kotlin.math.sqrt

fun main(args: Array<String>) {
    //calculate square root
    //sqrt() is an abbreviation of SQuare RooT,
    print("Enter some number and I'll calculate a square root: ")
    val a = readLine()!!.toDouble()
    if (a > 0) {
        println("The number you entered is greater than 0, so I can calculate it!")
        val root = sqrt(a);
        println("The square root of $a is $root")
    } else
        println("I can't calculate the square root of a negative number!")
    println("Thanks for the input")
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Enter some number and I'll calculate a square root: 25
The number you entered is greater than 0, so I can calculate it!
The square root of 25.0 is 5.0
Thanks for the input

Process finished with exit code 0
```

Fungsi dan Lambdas

Fungsi di Kotlin dideklarasikan menggunakan keyword *fun*. Fungsi dapat ditulis dalam 3 tempat yaitu : 1) dalam sebuah kelas, seperti method di Java ia sering disebut *member functions;* 2) diluar kelas, ia disebut *top-level function;* 3) dan didalam fungsi lainnya, ia disebut *local functions*. Aturan penulisan fungsi sama dengan Java yaitu 1) hindari menggunakan *reserved word* 2) tidak boleh dimulai dengan angka 3) tidak mengandung spesial karakter.

Bentuk dasar fungsi :

```
fun functionName([parameters]) [:type] {
```

```
        statements
}
```

Fungsi tanpa nilai kembali:

```
fun main(args: Array<String>) {
    displayMessage("Rekayasa Perangkat Lunak", 10);
}

fun displayMessage(msg: String, count: Int) {
    var counter = 1
    while(counter <= count ) {
        println("$msg $counter")
        counter++
    }
}
```

Output :

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Rekayasa Perangkat Lunak 1
Rekayasa Perangkat Lunak 2
Rekayasa Perangkat Lunak 3
Rekayasa Perangkat Lunak 4
Rekayasa Perangkat Lunak 5
Rekayasa Perangkat Lunak 6
Rekayasa Perangkat Lunak 7
Rekayasa Perangkat Lunak 8
Rekayasa Perangkat Lunak 9
Rekayasa Perangkat Lunak 10

Process finished with exit code 0
```

Fungsi dengan nilai kembali:

```
fun main(args: Array<String>) {
    val listNumber = listOf(1,2,3,4,5,6)
    println("Jumlah nilai pada elemen listNumber = ${getSum(listNumber)}")
}

fun getSum(values: List<Int>) : Int { // return type is Int
    var total = 0;
    for (i in values) total += i
    return total // return value
}
```

Output :

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Jumlah nilai pada elemen listNumber = 21

Process finished with exit code 0
```

Menggunakan Pair sebagai nilai kembali :

```
fun main(args: Array<String>) {
    var (x, y) = bigSmall(5,3)
    println("Nilai x = $x")
    println("Nilai x = $y")
}

fun bigSmall(a: Int, b:Int) : Pair<Int, Int> {
    if(a > b) return Pair(a,b)
    else {
        return Pair(b,a)
    }
}
```

Output :

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Nilai x = 5
Nilai x = 3

Process finished with exit code 0
```

Fungsi Ekspresi Tunggal (Single Expression Functions)

Terdapat cara kedua untuk menulis fungsi di Kotlin agar kode program menjadi lebih ringkas, jika terdapat kondisi yang bisa menghilangkan 1) return statement; 2) blok program; 3) return type secara bersamaan.

Contoh :

fun sumInt(a: Int, b: Int) = a + b

atau

fun sumInt (a: Int, b: Int): Int = a + b

Contoh kode program :

```
fun main(args: Array<String>) {
    println("Max value from max function = ${max(4,7)}")
    println("Max value from newMax function = ${newMax(3,5)}")
}

fun max(a: Int, b:Int): Int {
    val maxValue = if(a > b) a else b
```

```
    return maxValue
}

fun newMax(a: Int, b:Int) : Int = if(a > b) a else b
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Max value from max function = 7
Max value from newMax function = 5

Process finished with exit code 0
```

Argumen Default Fungsi

Parameter fungsi bisa memiliki nilai default yang membolehkan pemanggil fungsi untuk menghilangkan beberapa argumen ketika dipanggil.

Contoh :

```
fun main(args: Array<String>) {
    //panggil tanpa argumen
    connectToDb()
    println()
    //panggil dengan 2 argumen
    connectToDb("sqlserver", "depandi")
}

fun connectToDb(hostname: String = "localhost",
                username: String = "mysql",
                password:String = "secret") {
    println("hostname : $hostname")
    println("username : $username")
    println("password : $password")
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
hostname : localhost
username : mysql
password : secret

hostname : sqlserver
username : depandi
password : secret

Process finished with exit code 0
```

Ekspresi Lambdas

Bentuk dasar fungsi dengan ekspresi lambda adalah seperti berikut:

```
val sum: (Int, Int) -> Int = { x: Int, y: Int -> x + y }
```

Ekspresi lambda selalu dikelilingi oleh tanda blok '{}' deklarasi parameter berada dalam blok yang memiliki opsional type annotation pada bagian tubuh blok ditandai dengan tanda '->'. Jika tipe nilai kembalian dari fungsi bukan unit maka nilai kembalian diduga merupakan baris akhir dari program atau ekspresi dalam tubuh lambda tersebut.

Berikut adalah contoh program penggunaan fungsi lambda:

```
fun main(args: Array<String>) {
    println(MyFunction("Rekayasa Perangkat Lunak"))

    //Lambda Function
    val mylambda :(String)->Unit = {s:String->print(s)}
    val v:String = "Jurusan Teknik Informatika"
    mylambda(v)

    //Inline Function
    myFun(v,mylambda) //passing lambda as a parameter of another function
}

fun MyFunction(x: String): String {
    var c:String = "Hey!! Welcome To ---"
    return (c+x)
}

fun myFun(a:String, action:(String)->Unit) { //passing lambda
    print("\nHeyyy!!!")
    action(a)// call to lambda function
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Hey!! Welcome To ---Rekayasa Perangkat Lunak
Jurusan Teknik Informatika
Heyyy!!!Jurusan Teknik Informatika
Process finished with exit code 0
```

Tugas Praktikum

- Tuliskan pemahaman anda mengenai contoh koding program praktikum 1 dan screen shoot hasilnya.
- Buatlah laporan resmi praktikum 1.

Daftar Referensi

Ted Hagos, *Learn Android Studio 3 with Kotlin (Efficient Android App Development)*, Apress, Manila, 2018

Dmitry Jemerov and Svetlana Isakova, *Kotlin in Action*, Manning Publication Co., Shelter Island, 2017

https://www.tutorialspoint.com/kotlin

https://www.ict.social/kotlin/basics

https://kotlinlang.org/docs/reference/

** ☺ Selamat Mengerjakan ☺ ***