MODUL VII
FUNCTION, FUNCTION PARAMETER & FUNCTION RETURN VALUE

Pertemuan : 7
Waktu : 8 x 60 menit (Online)

1.1 Tujuan Modul VII

Setelah mahasiswa mempelajari materi ini, diharapkan dapat :

1. Memahami penggunaan dan pemanggilan function di golang.
2. Memahami penggunaan dan pemanggilan function parameter di golang.
3. Memahami penggunaan dan pemanggilan function return value di golang.

1.2 Landasan Teori

1.2.1 Function

Fungsi merupakan aspek penting dalam pemrograman. Definisi fungsi sendiri adalah sekumpulan blok kode yang dibungkus dengan nama tertentu. Penerapan fungsi yang tepat akan menjadikan kode lebih modular dan juga *dry* (kependekan dari *don't repeat yourself*), tak perlu menuliskan banyak kode yang kegunaannya berkali-kali, cukup sekali saja lalu panggil sesuai kebutuhan.

1.2.1.1 Penerapan Fungsi

Sebenarnya tanpa sadar, kita sudah menerapkan fungsi di bab-bab sebelum ini, yaitu pada fungsi `main`. Fungsi `main` merupakan fungsi yang paling utama pada program Go.

Cara membuat fungsi cukup mudah, yaitu dengan menuliskan keyword `func`, diikuti setelahnya nama fungsi, kurung yang berisikan parameter, dan kurung kurawal untuk membungkus blok kode.

Parameter sendiri adalah variabel yang disisipkan pada saat pemanggilan fungsi.

Silakan lihat dan praktekan kode tentang implementasi fungsi berikut.

```
package main

import "fmt"
import "strings"

func main() {
    var names = []string{"John", "Wick"}
    printMessage("halo", names)
}

func printMessage(message string, arr []string) {
    var nameString = strings.Join(arr, " ")
    fmt.Println(message, nameString)
}
```

Pada kode di atas, sebuah fungsi baru dibuat dengan nama `printMessage` memiliki 2 buah parameter yaitu string `message` dan slice string `arr`.

Fungsi tersebut dipanggil dalam `main`, dengan disisipkan 2 buah data sebagai parameter, data pertama adalah string `"hallo"` yang ditampung parameter `message`, dan parameter ke 2 adalah slice string `names` yang nilainya ditampung oleh parameter `arr`.

Di dalam `printMessage`, nilai `arr` yang merupakan slice string digabungkan menjadi sebuah string dengan pembatas adalah karakter **spasi**. Penggabungan slice dapat dilakukan dengan memanfaatkan fungsi `strings.Join()` (berada di dalam package `strings`).

1.2.1.2 Penggunaan Fungsi rand.Seed()

Fungsi ini diperlukan untuk memastikan bahwa angka random yang akan di-generate benar-benar acak. Kita bisa gunakan angka apa saja sebagai nilai parameter fungsi ini (umumnya diisi `time.Now().Unix()`).

```
rand.Seed(time.Now().Unix())
```

Fungsi `rand.Seed()` berada dalam package `math/rand`, yang harus di-import terlebih dahulu sebelum bisa dimanfaatkan.

Package `time` juga perlu di-import karena kita menggunakan fungsi `(time.Now().Unix())` disitu.

1.2.1.3 Import Banyak Package

Penulisan keyword `import` untuk banyak package bisa dilakukan dengan dua cara, dengan menuliskannya di tiap package, atau cukup sekali saja, bebas.

```
import "fmt"
import "math/rand"
import "time"

// atau

import (
    "fmt"
    "math/rand"
    "time"
)
```

1.2.1.4 Penggunaan Keyword return Untuk Menghentikan Proses Dalam Fungsi

Selain sebagai penanda nilai balik, keyword `return` juga bisa dimanfaatkan untuk menghentikan proses dalam blok fungsi dimana ia dipakai. Contohnya bisa dilihat pada kode berikut.

```
package main

import "fmt"

func main() {
    divideNumber(10, 2)
    divideNumber(4, 0)
    divideNumber(8, -4)
}

func divideNumber(m, n int) {
```

```
    if n == 0 {
        fmt.Printf("invalid divider. %d cannot divided by %d\n", m, n)
        return
    }

    var res = m / n
    fmt.Printf("%d / %d = %d\n", m, n, res)
}
```

Fungsi `divideNumber` didesain tidak memiliki nilai balik. Fungsi ini dibuat untuk membungkus proses pembagian 2 bilangan, lalu menampilkan hasilnya.

Didalamnya terdapat proses validasi nilai variabel pembagi, jika nilainya adalah 0, maka akan ditampilkan pesan bahwa pembagian tidak bisa dilakukan, lalu proses dihentikan pada saat itu juga (dengan memanfaatkan keyword `return`). Jika nilai pembagi valid, maka proses pembagian diteruskan.

1.2.2 Function Parameter

Khusus untuk fungsi yang tipe data parameternya sama, bisa ditulis dengan gaya yang unik. Tipe datanya dituliskan cukup sekali saja di akhir. Contohnya bisa dilihat pada kode berikut.

```
func nameOfFunc(paramA type, paramB type, paramC type) returnType
func nameOfFunc(paramA, paramB, paramC type) returnType

func randomWithRange(min int, max int) int
func randomWithRange(min, max int) int
```

1.2.3 Function Return Value

Sebuah fungsi bisa didesain tidak mengembalikan nilai balik (*void*), atau bisa mengembalikan suatu nilai. Fungsi yang memiliki nilai kembalian, harus ditentukan tipe data nilai baliknya pada saat deklarasi.

Program berikut merupakan contoh penerapan fungsi yang memiliki return value.

```
package main
```

```
import (
    "fmt"
    "math/rand"
    "time"
)

func main() {
    rand.Seed(time.Now().Unix())
    var randomValue int

    randomValue = randomWithRange(2, 10)
    fmt.Println("random number:", randomValue)
    randomValue = randomWithRange(2, 10)
    fmt.Println("random number:", randomValue)
    randomValue = randomWithRange(2, 10)
    fmt.Println("random number:", randomValue)
}

func randomWithRange(min, max int) int {
    var value = rand.Int() % (max - min + 1) + min
    return value
}
```

Fungsi `randomWithRange` bertugas untuk *generate* angka acak sesuai dengan range yang ditentukan, yang kemudian angka tersebut dijadikan nilai kembalian fungsi.

Cara menentukan tipe data nilai balik fungsi adalah dengan menuliskan tipe data yang diinginkan setelah kurung parameter. Bisa dilihat pada kode di atas, bahwa `int` merupakan tipe data nilai balik fungsi `randomWithRange`.

```
func randomWithRange(min, max int) int
```

Sedangkan cara untuk mengembalikan nilai itu sendiri adalah dengan menggunakan keyword `return` diikuti data yang ingin dikembalikan. Pada contoh di atas, `return value` artinya nilai variabel `value` dijadikan nilai kembalian fungsi.

Eksekusi keyword `return` akan menjadikan proses dalam blok fungsi berhenti pada saat itu juga. Semua statement setelah keyword tersebut tidak akan dieksekusi.

1.3 Praktikum

1.3.1 Latihan Praktikum

Buatkan program rekursif dari deret (tampilkan & jumlahkan) dan permasalahan berikut ini :

1. Sum = 1 + 6 + 36 + 216 + 1296 + ...
2. Sum = 5 + 50 + 500 + 5000 + 50000 +
3. Sum = 1 + 7 + 49 + 343 + 2401 + ...
4. Sum = 1 + 20 + 300 + 4000 + 50000 + ...
5. Sum = 5 + 50 + 500 + 5000 + 50000 + ...