LAPORAN PRAKTIKUM
ALGORITMA DAN PEMOGRAMAN 2

MODUL III

FUNGSI

Oleh:

YASVIN SYAHGANA

2311102065

S1 IF 11 02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

Fungsi adalah sekumpulan blok kode yang dibungkus dengan nama tertentu. Penerapan fungsi yang tepat akan menjadikan kode lebih modular dan juga dry (singkatan dari don't repeat yourself) yang artinya kita tidak perlu menuliskan banyak kode untuk kegunaan yang sama berulang kali. Cukup deklarasikan sekali saja blok kode sebagai suatu fungsi, lalu panggil sesuai kebutuhan.

A. Penerapan Fungsi

Fungsi *main()* sendiri merupakan fungsi utama pada program Go, yang akan dieksekusi ketika program dijalankan. Selain fungsi *main()*, kita juga bisa membuat fungsi lainnya. Dan caranya cukup mudah, yaitu dengan menuliskan keyword *func* kemudian diikuti nama fungsi, lalu kurung () (yang bisa diisi parameter), dan diakhiri dengan kurung kurawal untuk membungkus blok kode.

Parameter merupakan variabel yang menempel di fungsi yang nilainya ditentukan saat pemanggilan fungsi tersebut. Parameter sifatnya opsional, suatu fungsi bisa tidak memiliki parameter, atau bisa saja memeliki satu atau banyak parameter (tergantung kebutuhan).
Dalam Go fungsi yang sederhana itu bentuknya seperti ini`:

```
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)

}
```

Output :

```
novalagung:belajar-golang $ go run bab17.go
halo John Wick
novalagung:belajar-golang $
```

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(),* dalam pemanggilannya disisipkan dua buah argument parameter.

1. Argument parameter pertama adalah string "halo" yang ditampung parameter message
2. Argument 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)

B. Fungsi Dengan Return Value / Nilai Balik

Sealin parameter, fungsi bisa memiliki attribute return value atau nilai balik. Fungsi yang memiliki return value, saat deklarasinya harus ditentukan terlebih dahulu tipe data dari nilai baliknya. Fungsi yang tidak mengembalikan nilai apapun (contohnya seperti fungsi *main()* dan *printMessage()*) biasa disebut dengan *void function*. Berikut adalah contoh fungsi dengan pengembalian nilai.

```
package main

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

```
    "time"
)

var randomizer =
rand.New(rand.NewSource(time.Now().Unix()))

func main() {
    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 = randomizer.Int()%(max-min+1) + min
    return value
}
```

Output kode ini akan mencetak output berikut:

```
[novalagung:belajar-golang $ go run bab17.go                                ]
random number: 9
random number: 6
random number: 2
novalagung:belajar-golang $
```

Fungsi *randomWithRange()* didesain untuk generate angka acak sesuai dengan *range* yang ditentukan lewat parameter, yang kemudian angka tersebut dijadikan nilai balik 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()*.

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

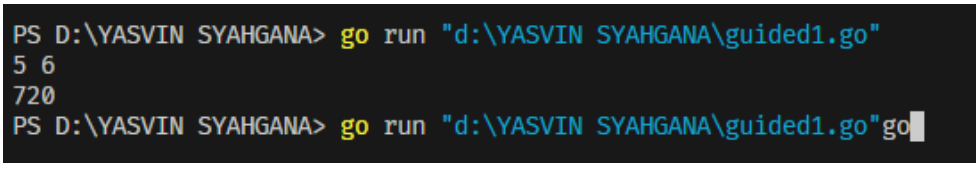
Eksekusi keyword *return* akan menjadikan proses dalam blok fungsi berhenti pada saat itu juga. Semua statement setelah keyword tersebut tidak akan dieksekusi.

II. GUIDED

Guided 1 | Source Code + Screenshot hasil program beserta penjelasan

```
package main

import "fmt"

func main() {
	var a, b int
	fmt.Scan(&a, &b)
	if a >= b {
		fmt.Print(permutasi(a, b))
	} else {
		fmt.Print(permutasi(b, a))
	}

}
func faktorial(n int) int {
	var hasil int = 1
	var i int
	for i = 1; i <= n; i++ {
		hasil = hasil * i
	}
	return hasil
}

func permutasi(n, r int) int {
	return faktorial(n) / faktorial(n-r)
}
```

Output



Deskripsi

Program ini digunakan untuk menghitung inputan dari user lalu menghitung nilai permutasi. Program ini memiliki beberapa fungsi yaitu `faktorial(n int)` Fungsi ini menghitung faktorial dari angka n, yaitu n! Yang kedua `permutasi(n,r int)` Fungsi ini menghitung nilai permutasi menggunakan rumus P(n,r)= n!/(n−r)! dengan memanggil fungsi faktorial. Dan yang ketiga Fungsi utama memeriksa apakah a >= b dan menghitung permutasi berdasarkan perbandingan tersebut. Memiliki Algoritma :

- Input a, b
- Jika a >= b, hitung permutasi P(a, b), jika tidak maka P(b, a)
- Kemudian tampilkan output dari P

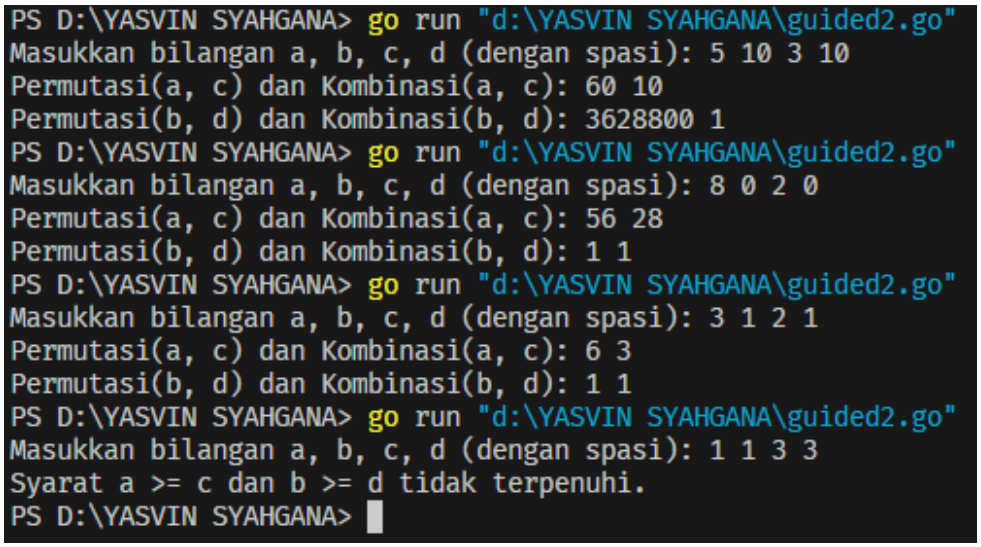
Guided 2 | Source Code + Screenshot hasil program beserta penjelasan

Sourcecode

```
package main

import "fmt"

// Fungsi buat menghitung faktorial
func factorial(n int) int {
	if n == 0 || n == 1 {
		return 1
	}

	result := 1
	for i := 2; i <= n; i++ {
		result *= i
	}
	return result
}

// Fungsi buat menghitung permutasi
func permutation(n, r int) int {
	return factorial(n) / factorial(n-r)
}

// Fungsi buat menghitung kombinasi
func combination(n, r int) int {
```

```
        return factorial(n) / (factorial(r) * factorial(n-
    r))
    }

    func main() {
        // Input 4 bilangan
        var a, b, c, d int
        fmt.Print("Masukkan bilangan a, b, c, d (dengan
    spasi): ")
        fmt.Scanf("%d %d %d %d", &a, &b, &c, &d)
        // Cek syarat a >= c dan b >= d
        if a >= c && b >= d {
            // Menghitung permutasi dan kombinasi a dan c
            permutasiAC := permutation(a, c)
            kombinasiAC := combination(a, c)
            // Menghitung permutasi dan kombinasi b dan d
            permutasiBD := permutation(b, d)
            kombinasiBD := combination(b, d)
            // Output hasil
            fmt.Println("Permutasi(a, c) dan Kombinasi(a,
    c):",
                permutasiAC, kombinasiAC)
            fmt.Println("Permutasi(b, d) dan Kombinasi(b,
    d):",
                permutasiBD, kombinasiBD)
        } else {
            fmt.Println("Syarat a >= c dan b >= d tidak
    terpenuhi.")
        }
}
```

Screenshoot Output



Deskripsi Program

Program ini merupakan implementasi lebih lanjut dari guided 1 yang dimana program ini digunakan untuk menghitung permutasi dan kombinasi. Program ini memiliki fungsi yang sama dengan guided1 tetapi memiki fungsi lain juga berikut fungsi tambahan dari guided1 `Kombinasi(n, r int)` fungsi ini menghitung nilai kombinasi dengan rumus C(n,r) = n! /r!*(n-r)!. Algoritmanya adalah :

- Input (a, b, c, d) type data int
- Cek kondisi apakah `a >= c && b >= d`
- Jika true maka, lakukan permutasi dan kombinasi dari P(a,c), C(a,c), P(b,d), C(b,d) dan lakukan print output hasil dari permutasi dan kombinasi
- Jika false maka output akan ”`Kondisi 'a >= c && b >= d' tidak terpenuhi "`

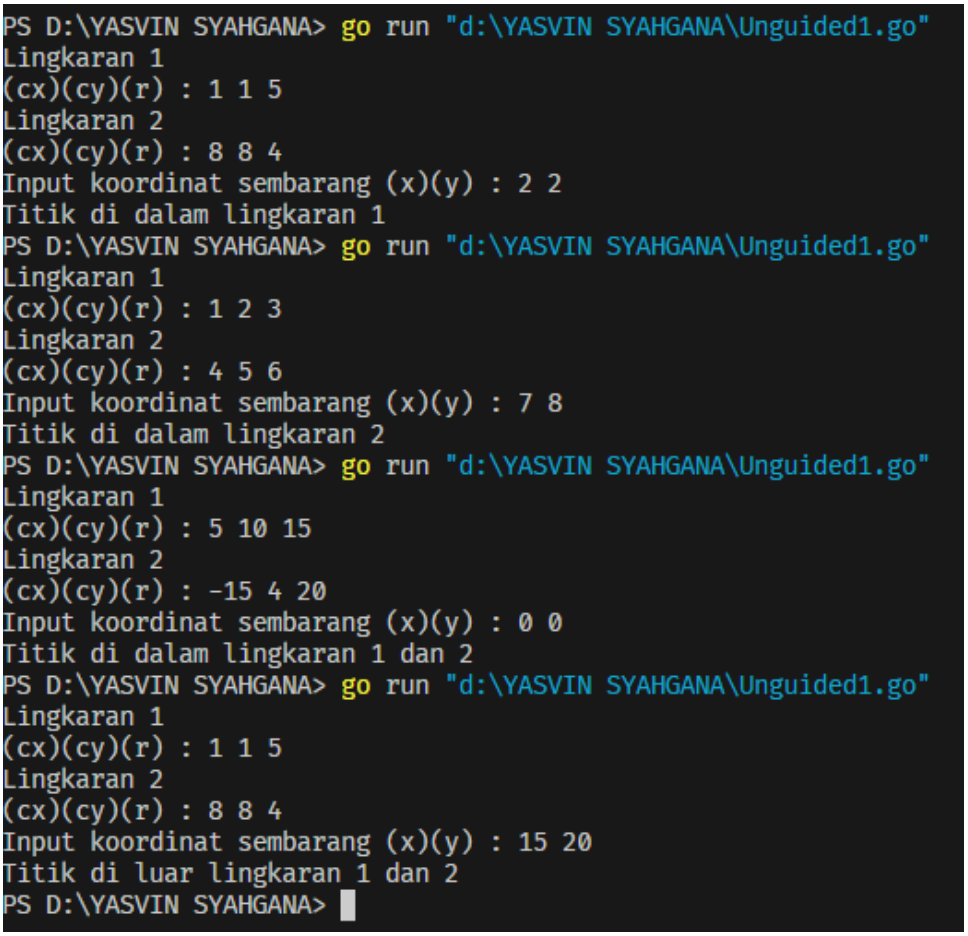
III. UNGUIDED

Source Code + Screenshot hasil program beserta penjelasan

```
package main

import (
	"fmt"
	"math"
)

func main() {
	var cx, cxx, cy, cyy, r, rr, x, y float64

	fmt.Println("Lingkaran 1")
	fmt.Print("(cx)(cy)(r) : ")
	fmt.Scanln(&cx, &cy, &r)

	fmt.Println("Lingkaran 2")
	fmt.Print("(cx)(cy)(r) : ")
	fmt.Scanln(&cxx, &cyy, &rr)

	fmt.Print("Input koordinat sembarang (x)(y) : ")
	fmt.Scanln(&x, &y)

	//mari kita cek posisi Lingkaran1 dang Lingkaran2
	posisi1 := inercircle(cx, cy, x, y, r)
	posisi2 := inercircle(cxx, cyy, x, y, rr)

	if posisi1 && posisi2 {
		fmt.Println("Titik di dalam lingkaran 1 dan
2")
	} else if posisi1 {
		fmt.Println("Titik di dalam lingkaran 1")
	} else if posisi2 {
		fmt.Println("Titik di dalam lingkaran 2")
	} else {
		fmt.Println("Titik di luar lingkaran 1 dan 2")
	}

}

func jarak(a, b, c, d float64) float64 {
	//pengecekan Jarak
	kiri := (a - c) * (a - c)
	kanan := (b - d) * (b - d)
	return math.Sqrt(kiri + kanan)
	//math.Sqrt digunakan untuk pemanggilan akar
}

func didalam(cx, cy, x, y, r float64) bool {
	//pengecekan apakah x, y berada dalam lingkaran
```

```
        return jarak(cx, cy, x, y) <= r
}
```

Screenshoot Output



Deskripsi Program

Program ini dibuat untuk menenentukan apakah titik koordinat berada pada lingkaran 1 atau lingkaran 2, atau di tengah keduanya. Program ini memiliki beberapa fungsi yaitu `func jarak(a, b, c, d float64)` untuk menghitung jarak antara titik sembarang dan pusat lingkaran, dan fungsi `dan func didalam(cx, cy, x, y, r float64)` untuk memeriksa apakah titik tersebut berada di dalam lingkaran dengan membandingkan jarak ini dengan jari-jari lingkaran.

Algoritma dari program ini adalah :

- //lingkarang pertama
- Input (cx, cy, r)
- Input (x,y)
- //lingkarang kedua
- Input (cxx, cyy, rr)
- Input (x,y)
- `posisi1 := inercircle(cx, cy, x, y, r)`
- `posisi2 := inercircle(cxx, cyy, x, y, rr)`
- posisi1, dan posisi2 digunakan untuk mengecek apakah kedua lingkaran *true* atau *false* dan menghitung jarak antara `(cx, cy, x, y)`
- Kemudian program akan mengklasifikasi apakah
  - `if posisi1 && posisi2 maka "Titik di dalam lingkaran 1 dan 2"`
  - `if posisi1 maka "Titik di dalam lingkaran 1"`
  - `if posisi2 maka "Titik di dalam lingkaran 2"`
  - `jika tidak ketiganya maka "Titik di luar lingkaran 1 dan 2"`