

**LAPORAN PRAKTIKUM  
PEMROGRAMAN BERORIENTASI OBJEK**

**MODUL III**

**FUNGSI**



Oleh:

**IKRAM IRIANSYAH**

2311102184

S1IF-11-02

**S1 TEKNIK INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

**2024**

## I. DASAR TEORI

### A. Dasar Fungsi dalam Bahasa Go

- Fungsi adalah blok kode yang digunakan untuk melakukan tugas tertentu secara modular dan reusable.
- Dalam Bahasa Go, fungsi merupakan first-class citizens, yang berarti mereka dapat disimpan dalam variabel, diteruskan sebagai argumen ke fungsi lain, dan bahkan dikembalikan dari fungsi.

### B. Deklarasi Fungsi

- Fungsi dideklarasikan menggunakan kata kunci **func**, diikuti oleh nama fungsi, parameter (jika ada), tipe kembalian (jika ada), dan blok kode yang akan dieksekusi.

```
func tambah(a int, b int) int {  
    return a + b  
}
```

- Fungsi **tambah** di atas menerima dua parameter **a** dan **b** bertipe **int**, kemudia mengembalikan hasil penjumlahan keduanya sebagai **int**.

### C. Multiple Return Values (Pengembalian Banyak Nilai)

- Go mendukung pengembalian lebih dari satu nilai dari suatu fungsi.
- Ini sering digunakan, terutama saat kita ingin mengembalikan hasil operasi bersama error.

```
func bagi(a, b int) (int, error) {  
    if b == 0 {  
        return 0, fmt.Errorf("tidak bisa membagi dengan nol")  
    }  
    return a / b, nil  
}
```

- Fungsi **bagi** mengembalikan dua nilai: hasil pembagian dan pesan error.
- Jika terjadi pembagian dengan nol, fungsi mengembalikan error.

#### D. Variadic Function (Fungsi dengan Argumen Tidak Terbatas)

- Variadic function adalah fungsi yang dapat menerima jumlah argumen yang tidak terbatas .
- Ini dicapai dengan menggunakan tiga titik (...) sebelum tipe parameter.

```
func jumlah(angka ...int) int {
    total := 0
    for _, n := range angka {
        total += n
    }
    return total
}
```

- Fungsi **jumlah** dapat menerima sejumlah argument integer dan menjumlahkannya.

#### E. Anonymous Function (Fungsi Anonim)

- Fungsi anonim adalah fungsi tanpa nama yang bisa didefinisikan secara langsung, sering kali digunakan untuk tugas sederhana atau sebagai callback.

```
func main() {
    func() {
        fmt.Println("Ini adalah fungsi anonim")
    }()
}
```

- Fungsi anonim ini langsung dieksekusi setelah didefinisikan.

## F. Fungsi sebagai Argumen dan Nilai Kembalian

- Fungsi dalam Go juga dapat diteruskan sebagai argumen ke fungsi lain atau dikembalikan sebagai nilai.

```
func apply(fn func(int) int, value int) int {  
    return fn(value)  
}  
  
func kaliDua(x int) int {  
    return x * 2  
}  
  
func main() {  
    result := apply(kaliDua, 5)  
    fmt.Println(result) // Output: 10  
}
```

- Fungsi **apply** menerima fungsi **kaliDua** sebagai argumen dan menerapkannya pada nilai **5**.

## G. Error Handling dengan Panic dan Recover

- Dalam Go, mekanisme panic dan recover dapat digunakan untuk menangani error yang lebih fatal.
- **Panic** digunakan untuk menghentikan program, sedangkan **recover** untuk memulihkan eksekusi program dari kondisi panic.

```
func safeDivision(a, b int) {  
    defer func() {  
        if r := recover(); r != nil {  
            fmt.Println("Recovered from:", r)  
        }  
    }  
}
```

```
}()
if b == 0 {
    panic("division by zero")
}
fmt.Println(a / b)
}

func main() {
    safeDivision(4, 2) // Output: 2
    safeDivision(4, 0) // Output: Recovered from:
division by zero
}
```

- Fungsi **safeDivision** menggunakan **panic** untuk menghentikan eksekusi jika pembagian dengan nol terjadi, dan **recover** untuk menangani kondisi tersebut.

## II. GUIDED

### Guided 1

```
package main

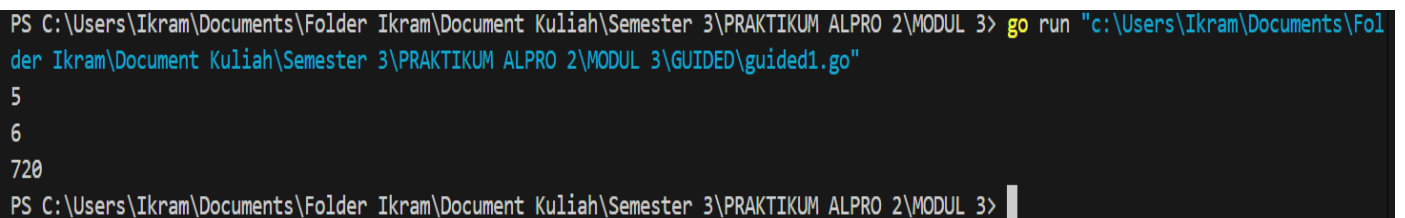
import "fmt"

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

### Screenshot output



```
PS C:\Users\Ikram\Documents\Folder Ikram\Document Kuliah\Semester 3\PRAKTIKUM ALPRO 2\MODUL 3> go run "c:\Users\Ikram\Documents\Folder Ikram\Document Kuliah\Semester 3\PRAKTIKUM ALPRO 2\MODUL 3\GUIDED\guided1.go"
5
6
720
PS C:\Users\Ikram\Documents\Folder Ikram\Document Kuliah\Semester 3\PRAKTIKUM ALPRO 2\MODUL 3>
```

## Deskripsi Program

Program ini menerima dua bilangan bulat dari pengguna, lalu memeriksa bilangan mana yang lebih besar. Program akan menghitung nilai permutasi dari bilangan yang lebih besar sebagai  $n$  dan bilangan yang lebih kecil sebagai  $r$ . Hasil dari permutasi dihitung menggunakan rumus  $P(n,r) = n!/(n-r)!$ , dimana  $n!$  adalah faktorial dari  $n$  dan  $(n-r)!$  adalah faktorial  $n-r$ .

## Guided 2

```
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.Println("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.")
    }
}

```

## Screenshot Output

```

PS C:\Users\Ikram\Documents\Folder Ikram\Document Kuliah\Semester 3\PRAKTIKUM ALPRO 2\MODUL 3> go run "c:\Users\Ikram\Documents\Fol
der Ikram\Document Kuliah\Semester 3\PRAKTIKUM ALPRO 2\MODUL 3\GUIDED\guided2.go"
Masukkan bilangan a, b, c, d (dengan spasi):
8 6 4 2
Permutasi(a, c) dan Kombinasi(a, c): 1680 70
Permutasi(b, d) dan Kombinasi(b, d): 30 15
PS C:\Users\Ikram\Documents\Folder Ikram\Document Kuliah\Semester 3\PRAKTIKUM ALPRO 2\MODUL 3>

```



## Deskripsi Program

Program ini menghitung permutasi dan kombinasi dari dua pasang bilangan. Permutasi digunakan untuk menghitung banyaknya cara mengatur objek dengan urutan tertentu, sedangkan kombinasi menghitung banyaknya cara memilih objek tanpa memperhatikan urutan. Pengguna diminta memasukkan empat bilangan: a, b, c, dan d. Program hanya melakukan perhitungan jika memenuhi syarat bahwa  $a \geq c$  dan  $b \geq d$ . Jika syarat terpenuhi, program akan menampilkan hasil perhitungan permutasi dan kombinasi dari pasangan (a, c) dan (b, d). Jika tidak, program akan menampilkan pesan bahwa syarat tidak terpenuhi. Program ini berguna untuk mempermudah perhitungan kombinatorial dan memastikan hasil yang akurat dengan validasi input yang baik.

### III. UNGUIDED

#### Unguided 1

```
package main

import (
    "fmt"
    "math"
)

// Fungsi jarak menghitung jarak antara dua titik (a,b) dan
// (c,d) dalam koordinat kartesian
// menggunakan rumus jarak Euclidean:  $\sqrt{(a-c)^2 + (b-d)^2}$ 
func jarak(a, b, c, d float64) float64 {
    return math.Sqrt(math.Pow(a-c, 2) + math.Pow(b-d, 2))
}

// Fungsi didalam memeriksa apakah suatu titik (x,y) berada
// di dalam lingkaran
// dengan pusat (cx,cy) dan radius r
func didalam(cx, cy, r, x, y float64) bool {
    return jarak(cx, cy, x, y) <= r
}

func main() {
    // Deklarasi variabel untuk koordinat dan radius dua
    // lingkaran, serta titik yang akan diperiksa
    var cx1, cy1, r1, cx2, cy2, r2, x, y float64

    // Meminta input dari pengguna untuk lingkaran 1
    fmt.Println("Masukkan koordinat titik pusat lingkaran 1
    (cx, cy) dan radius (r):")
    fmt.Scanln(&cx1, &cy1, &r1)

    // Meminta input dari pengguna untuk lingkaran 2
    fmt.Println("Masukkan koordinat titik pusat lingkaran 2
    (cx, cy) dan radius (r):")
    fmt.Scanln(&cx2, &cy2, &r2)
```

```
// Meminta input koordinat titik yang akan diperiksa
fmt.Println("Masukkan koordinat titik (x, y):")
fmt.Scanln(&x, &y)

// Memeriksa posisi titik terhadap kedua lingkaran dan
mencetak hasilnya
if didalam(cx1, cy1, r1, x, y) && didalam(cx2, cy2, r2, x,
y) {
    fmt.Println("Titik di dalam lingkaran 1 dan 2")
} else if didalam(cx1, cy1, r1, x, y) {
    fmt.Println("Titik di dalam lingkaran 1")
} else if didalam(cx2, cy2, r2, x, y) {
    fmt.Println("Titik di dalam lingkaran 2")
} else {
    fmt.Println("Titik di luar lingkaran ")
}
}
```

### Screenshot Output

```
PS C:\Users\Ikram\Documents\Folder Ikram\Document Kuliah\Semester 3\PRAKTIKUM ALPRO 2\MODUL 3> go run "c:\Users\Ikram\Documents\Fol
der Ikram\Document Kuliah\Semester 3\PRAKTIKUM ALPRO 2\MODUL 3\UNGUIDED\unguided1.go"
Masukkan koordinat titik pusat lingkaran 1 (cx, cy) dan radius (r):
1 2 3
Masukkan koordinat titik pusat lingkaran 2 (cx, cy) dan radius (r):
4 5 6
Masukkan koordinat titik (x, y):
7 8
Titik di dalam lingkaran 2
PS C:\Users\Ikram\Documents\Folder Ikram\Document Kuliah\Semester 3\PRAKTIKUM ALPRO 2\MODUL 3> |
```

### Deskripsi Program

Program ini adalah aplikasi Go yang menentukan posisi sebuah titik relatif terhadap dua buah lingkaran. Program meminta input koordinat pusat dan radius untuk dua lingkaran, serta koordinat sebuah titik, kemudian menentukan apakah titik tersebut berada di dalam salah satu, kedua, atau di luar kedua lingkaran.

