

**LAPORAN PRAKTIKUM  
ALGORITMA PEMROGRAMAN 2  
MODUL III & IV  
FUNGSI DAN PROSEDUR**



Oleh:

NAMA: Didik Weka Pratama

NIM: 2311102285

KELAS: S1 IF-11-07

**S1 TEKNIK INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

## **I. DASAR TEORI**

Fungsi adalah blok kode yang dapat dipanggil berulang kali dan dirancang untuk melakukan tugas tertentu. Fungsi memiliki input, yang disebut parameter, dan dapat mengembalikan output. Ini memungkinkan tugas besar dibagi menjadi tugas-tugas kecil yang lebih mudah dikelola, yang memudahkan pengembangan program. Setiap fungsi juga memiliki kemampuan untuk mengembalikan nilai, yang memungkinkan program untuk mengambil hasil perhitungan dan menggunakannya di bagian lain program. Ini mengurangi duplikasi kode, meningkatkan keterbacaan dan pemeliharaan program, dan memungkinkan program untuk menggunakan hasil perhitungan di bagian lain program.

Namun, ada satu perbedaan utama antara kedua prosedur dan fungsinya: prosedur tidak mengembalikan nilai. Prosedur digunakan dalam beberapa bahasa pemrograman untuk menyelesaikan tugas tertentu tanpa mengembalikan hasil. Untuk melakukan tugas tertentu, seperti mencetak keluaran, mengubah nilai variabel global, atau mengatur kondisi program, prosedur berguna.

Penggunaan fungsi dan prosedur dalam pemrograman umumnya mendukung konsep modularitas (pemisahan logika program ke dalam modul-modul yang lebih kecil dan terpisah) dan reusability (penggunaan ulang kode). Dengan adanya fungsi dan prosedur, program menjadi lebih terstruktur, lebih mudah diuji, dan lebih mudah dikembangkan seiring bertambahnya kompleksitas.

## II. GUIDED

### 1. Source Code

```
package main
```

```
import (  
    "fmt"  
)
```

```
func volumeTabung(jari_jari, tinggi int) float64 {  
    var luasAlas, volume float64  
    luasAlas = 3.14 * float64(jari_jari*jari_jari)  
    volume = luasAlas * float64(tinggi)  
    return volume  
}
```

```
func main() {  
    var r, t int  
    var v1, v2 float64  
    r = 5  
    t = 10  
  
    v1 = volumeTabung(r, t)  
    v2 = volumeTabung(r, t) + volumeTabung(15, t)  
  
    fmt.Println("Volume tabung 14 x 100: ", volumeTabung(14, 100))
```

```
    fmt.Println("v1: ", v1)

    fmt.Println("v2: ", v2)

}
```

## Screenshot Output



```
PS C:\golang> go run "c:\golang\src\hello.go"
Volume tabung 14 x 100: 61544.000000000001
v1: 785
v2: 7850
PS C:\golang>
```

## Deskripsi Program

Program di atas adalah program Go sederhana yang menghitung volume tabung menggunakan fungsi `volumeTabung`. Fungsi ini menerima dua parameter, yaitu jari-jari (`jari_jari`) dan tinggi (`tinggi`), kemudian menghitung luas alas tabung dengan rumus luas lingkaran ( $\pi * r^2$ ) dan mengalikan hasilnya dengan tinggi untuk mendapatkan volume tabung. Dalam fungsi `main`, jari-jari dan tinggi tabung diinisialisasi, dan dua volume (`v1` dan `v2`) dihitung. `v1` mewakili volume dari satu tabung dengan jari-jari dan tinggi tertentu, sementara `v2` merupakan penjumlahan volume beberapa tabung. Program ini juga menampilkan volume tabung dengan jari-jari dan tinggi berbeda menggunakan fungsi tersebut.

## 2.Source Code

```
package main
```

```
import (

    "fmt"

)
```

```
func cetakNFibo(n int) {

    var f1, f2, f3 int
```

```
f2 = 0

f3 = 1

for i := 1; i <= n; i++ {

    fmt.Println(f3)

    f1 = f2

    f2 = f3

    f3 = f1 + f2

}
```

```
func main() {

    var x int

    x = 5

    cetakNFibo(x)

    cetakNFibo(100)

}
```

**SCREENSHOOT OUTPUT**

```
PS C:\golang> go run "c:\golang\src\hello.go"
1
1
2
3
5
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
10946
```

## Deskripsi program

Program di atas adalah program Go yang mencetak deret bilangan Fibonacci hingga nilai yang ditentukan. Fungsi cetakNFibo menerima parameter n yang menunjukkan berapa banyak bilangan Fibonacci yang ingin dicetak. Deret Fibonacci dimulai dengan dua angka pertama, yaitu 0 dan 1. Di dalam loop, nilai-nilai Fibonacci berikutnya dihitung dengan menjumlahkan dua angka sebelumnya. Program mencetak nilai-nilai tersebut satu per satu. Pada fungsi main, program memanggil cetakNFibo dua kali, pertama untuk mencetak 5 bilangan Fibonacci, kemudian untuk mencetak hingga 100 bilangan Fibonacci.

### **III. UNGUIDED**

#### **Modul III.**

##### **No 1. Source Code**

```
package main
```

```
import (  
    "fmt"  
)
```

```
func faktorial(n int) int {  
    if n == 0 {  
        return 1  
    }  
    hasil := 1  
    for i := 1; i <= n; i++ {  
        hasil *= i  
    }  
    return hasil  
}
```

```
func permutasi(total, pilih int) int {  
    return faktorial(total) / faktorial(total-pilih)  
}
```

```
func kombinasi(total, pilih int) int {
```

```

    return faktorial(total) / (faktorial(pilih) * faktorial(total-pilih))
}

func main() {
    var totalA, totalB, pilihA, pilihB int

    fmt.Print("Masukkan nilai totalA, totalB, pilihA, pilihB: ")

    fmt.Scan(&totalA, &totalB, &pilihA, &pilihB)

    if totalA >= pilihA && totalB >= pilihB {
        permutasiA := permutasi(totalA, pilihA)
        kombinasiA := kombinasi(totalA, pilihA)
        permutasiB := permutasi(totalB, pilihB)
        kombinasiB := kombinasi(totalB, pilihB)

        fmt.Printf("Permutasi(%d, %d) = %d, Kombinasi(%d, %d) = %d\n", totalA,
            pilihA, permutasiA, totalA, pilihA, kombinasiA)

        fmt.Printf("Permutasi(%d, %d) = %d, Kombinasi(%d, %d) = %d\n", totalB,
            pilihB, permutasiB, totalB, pilihB, kombinasiB)

    } else {
        fmt.Println("Input tidak valid, pastikan totalA >= pilihA dan totalB >=
            pilihB")
    }
}

```



## Screenshoot Output

```
PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan nilai totalA, totalB, pilihA, pilihB: 5 10 3 10
Permutasi(5, 3) = 60, Kombinasi(5, 3) = 10
Permutasi(10, 10) = 3628800, Kombinasi(10, 10) = 1
PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan nilai totalA, totalB, pilihA, pilihB: 8 0 2 0
Permutasi(8, 2) = 56, Kombinasi(8, 2) = 28
Permutasi(0, 0) = 1, Kombinasi(0, 0) = 1
PS C:\golang> 
```

## Deskripsi Program

Program di atas menghitung permutasi dan kombinasi dari dua pasang angka dengan menggunakan fungsi faktorial. Variabel a dan b mewakili dua angka yang akan dipasangkan dengan c dan d untuk menghitung permutasi dan kombinasi. Pertama, program menerima input nilai untuk a, b, c, dan d. Kemudian, program memeriksa apakah nilai a lebih besar atau sama dengan c dan apakah b lebih besar atau sama dengan d. Jika syarat tersebut terpenuhi, program menghitung dan menampilkan permutasi dan kombinasi untuk masing-masing pasangan angka. Jika tidak, program akan menampilkan pesan bahwa input tidak valid.

## No 2. Source Code

```
package main
```

```
import (
```

```
    "fmt"
```

```
)
```

```
func p(x int) int {
```

```
    return x * x
```

```
}
```

```
func q(x int) int {
```

```

    return x - 2
}

func r(x int) int {
    return x + 1
}

func main() {
    var x, y, z int
    fmt.Println("Masukkan tiga bilangan bulat (x, y, z):")
    fmt.Scan(&x, &y, &z)

    hasil1 := p(q(r(x)))
    hasil2 := q(r(p(y)))
    hasil3 := r(p(q(z)))

    fmt.Printf("(p ◦ q ◦ r)(%d) = %d\n", x, hasil1)
    fmt.Printf("(q ◦ r ◦ p)(%d) = %d\n", y, hasil2)
    fmt.Printf("(r ◦ p ◦ q)(%d) = %d\n", z, hasil3)
}

```

### **Screenshoot Output**

```

PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan tiga bilangan bulat (x, y, z):
7 2 10
(p ◦ q ◦ r)(7) = 36
(q ◦ r ◦ p)(2) = 3
(r ◦ p ◦ q)(10) = 65
PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan tiga bilangan bulat (x, y, z):
5 5 5
(p ◦ q ◦ r)(5) = 16
(q ◦ r ◦ p)(5) = 24
(r ◦ p ◦ q)(5) = 10

PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan tiga bilangan bulat (x, y, z):
3 8 4
(p ◦ q ◦ r)(3) = 4
(q ◦ r ◦ p)(8) = 63
(r ◦ p ◦ q)(4) = 5
PS C:\golang>

```

### Deskripsi Program

Program ini mengimplementasikan komposisi fungsi dengan tiga fungsi yang berbeda: p, q, dan r, yang masing-masing melakukan operasi matematika tertentu. Fungsi p menghitung kuadrat dari input, fungsi q mengurangi input dengan 2, dan fungsi r menambahkan 1 pada input. Dalam fungsi main, program meminta pengguna untuk memasukkan tiga bilangan bulat, yang kemudian digunakan sebagai argumen untuk menghitung hasil dari komposisi fungsi. Hasil dari komposisi  $(p \circ q \circ r)(x)$ ,  $(q \circ r \circ p)(y)$ , dan  $(r \circ p \circ q)(z)$  dicetak ke layar. Program ini secara efektif menunjukkan bagaimana fungsi dapat digabungkan untuk menghasilkan hasil yang kompleks dari operasi sederhana pada bilangan bulat.

### No 3. Source Code

```
package main
```

```
import (
```

```
    "fmt"
```

```
    "math"
```

```
)
```

```
// Fungsi untuk menghitung jarak antara dua titik
```

```
func jarak(a, b, c, d float64) float64 {  
    return math.Sqrt(math.Pow(a-c, 2) + math.Pow(b-d, 2))  
}
```

```
// Fungsi untuk menentukan apakah titik berada di dalam lingkaran
```

```
func diDalam(cx, cy, r, x, y float64) bool {  
    return jarak(cx, cy, x, y) <= r  
}
```

```
func main() {
```

```
    var cx1, cy1, r1, cx2, cy2, r2, x, y float64
```

```
    // Input untuk lingkaran 1
```

```
    fmt.Print("Masukkan cx1, cy1, r1: ")
```

```
    fmt.Scan(&cx1, &cy1, &r1)
```

```
    // Input untuk lingkaran 2
```

```
    fmt.Print("Masukkan cx2, cy2, r2: ")
```

```
    fmt.Scan(&cx2, &cy2, &r2)
```

```
    // Input untuk titik sembarang
```

```
    fmt.Print("Masukkan x, y: ")
```

```
    fmt.Scan(&x, &y)
```

```

// Cek posisi titik

inCircle1 := diDalam(cx1, cy1, r1, x, y)

inCircle2 := diDalam(cx2, cy2, r2, x, y)


// Tentukan output berdasarkan posisi titik

if inCircle1 && inCircle2 {

    fmt.Println("Titik di dalam lingkaran 1 dan 2")

} else if inCircle1 {

    fmt.Println("Titik di dalam lingkaran 1")

} else if inCircle2 {

    fmt.Println("Titik di dalam lingkaran 2")

} else {

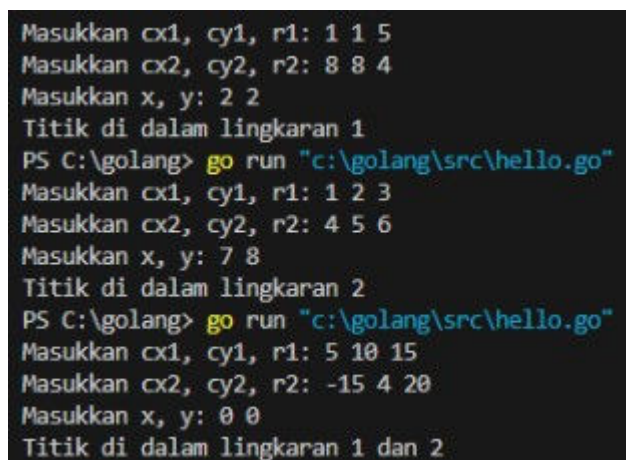
    fmt.Println("Titik di luar lingkaran 1 dan 2")

}

}

```

### Screenshoot Output



```

Masukkan cx1, cy1, r1: 1 1 5
Masukkan cx2, cy2, r2: 8 8 4
Masukkan x, y: 2 2
Titik di dalam lingkaran 1
PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan cx1, cy1, r1: 1 2 3
Masukkan cx2, cy2, r2: 4 5 6
Masukkan x, y: 7 8
Titik di dalam lingkaran 2
PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan cx1, cy1, r1: 5 10 15
Masukkan cx2, cy2, r2: -15 4 20
Masukkan x, y: 0 0
Titik di dalam lingkaran 1 dan 2

```

### Deskripsi Program

Program ini dirancang untuk menentukan posisi suatu titik relatif terhadap dua lingkaran yang berbeda. Pertama, pengguna diminta untuk memasukkan koordinat pusat dan jari-jari dari dua lingkaran. Selanjutnya, program meminta pengguna untuk memasukkan koordinat titik yang akan diperiksa. Dengan menggunakan fungsi jarak, program menghitung jarak antara titik dan pusat lingkaran untuk menentukan apakah titik tersebut berada di dalam lingkaran. Program kemudian memeriksa apakah titik tersebut berada di dalam satu atau kedua lingkaran, atau di luar keduanya. Hasil dari pemeriksaan ini akan dicetak ke layar, memberikan informasi yang jelas tentang posisi titik tersebut relatif terhadap lingkaran yang dimaksud.

## Modul IV

### No 1. Source Code

```
package main
```

```
import (
```

```
    "fmt"
```

```
)
```

```
// Prosedur untuk menghitung dan mencetak faktorial
```

```
func cetakFaktorial(n int) int {
```

```
    if n == 0 {
```

```
        return 1
```

```
    }
```

```
    hasil := 1
```

```
    for i := 1; i <= n; i++ {
```

```
        hasil *= i
```

```
    }
```

```
    return hasil
```

```
}
```

```
// Prosedur untuk menghitung dan mencetak permutasi
```

```
func cetakPermutasi(n, r int) {
```

```
    perm := cetakFaktorial(n) / cetakFaktorial(n-r)
```

```
    fmt.Printf("Permutasi(%d, %d) = %d\n", n, r, perm)
```

```
}
```

```
// Prosedur untuk menghitung dan mencetak kombinasi

func cetakKombinasi(n, r int) {

    komb := cetakFaktorial(n) / (cetakFaktorial(r) * cetakFaktorial(n-r))

    fmt.Printf("Kombinasi(%d, %d) = %d\n", n, r, komb)

}

func main() {

    var a, b, c, d int

    fmt.Println("Masukkan empat bilangan bulat positif a, b, c, dan d:")

    fmt.Scan(&a, &b, &c, &d)

    if a >= c && b >= d {

        cetakPermutasi(a, c)

        cetakKombinasi(a, c)

        cetakPermutasi(b, d)

        cetakKombinasi(b, d)

    } else {

        fmt.Println("Input tidak valid, pastikan a >= c dan b >= d")

    }

}
```

**Screenshot Output**



```

PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan empat bilangan bulat positif a, b, c, dan d:
5 10 3 10
Permutasi(5, 3) = 60
Kombinasi(5, 3) = 10
Permutasi(10, 10) = 3628800
Kombinasi(10, 10) = 1
PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan empat bilangan bulat positif a, b, c, dan d:
8 0 2 0
Permutasi(8, 2) = 56
Kombinasi(8, 2) = 28
Permutasi(0, 0) = 1
Kombinasi(0, 0) = 1
PS C:\golang>

```

### Deskripsi Program

Program ini bertujuan untuk menghitung dan menampilkan nilai faktorial, permutasi, dan kombinasi dari empat bilangan bulat positif yang dimasukkan oleh pengguna. Program dimulai dengan meminta pengguna untuk memasukkan dua pasangan bilangan, yaitu a, b, c, dan d. Setelah itu, program memeriksa apakah syarat  $a \geq c$  dan  $b \geq d$  terpenuhi. Jika ya, program akan menghitung dan mencetak nilai permutasi dan kombinasi untuk setiap pasangan bilangan menggunakan prosedur yang terpisah untuk masing-masing perhitungan. Hasil dari setiap perhitungan ditampilkan langsung di layar, memberikan gambaran jelas mengenai hubungan antara bilangan yang dimasukkan. Jika syarat tidak terpenuhi, program akan memberikan pesan kesalahan kepada pengguna. Program ini efektif dalam mengilustrasikan konsep dasar matematika kombinatorik melalui penggunaan prosedur dalam pemrograman.

### No 2. Source Code

```
package main
```

```
import "fmt"
```

```
// Didik Weka Pratama
```

```
// 2311102285
```

```
type Contestant struct {  
    Name string  
    Time []int  
    Score int  
    Completed int  
}
```

```
func calculateScore(contestant *Contestant) {  
    contestant.Score = 0  
    contestant.Completed = 0  
  
    for _, time := range contestant.Time {  
        if time == 301 {  
            // Score does not increase if time is 301  
            continue  
        } else {  
            contestant.Score += time  
            contestant.Completed++  
        }  
    }  
}
```

```
func main() {  
    contestants := []Contestant{  
        {Name: "Astuti", Time: []int{20, 50, 301, 301, 61, 71, 75, 10}},  
    },  
}
```

```

        {Name: "Bertha", Time: []int{25, 47, 301, 26, 50, 60, 65, 21}},
    }

    for i := range contestants {
        calculateScore(&contestants[i])
    }

    var winner Contestant

    for i := range contestants {
        if contestants[i].Completed > winner.Completed {
            winner = contestants[i]
        } else if contestants[i].Completed == winner.Completed {
            if winner.Score == 0 || contestants[i].Score < winner.Score
{
                winner = contestants[i]
            }
        }
    }

    fmt.Printf("Output:\n")

    fmt.Printf("%s %d %d\n", winner.Name, winner.Score,
winner.Completed)
}

```

### Screenshot Output

```
PS C:\golang> go run "c:\golang\src\hello.go"
Output:
Bertha 294 7
PS C:\golang>
```

### Deskripsi Program

Program ini dirancang untuk menghitung skor dan menentukan pemenang dalam suatu kompetisi berdasarkan waktu penyelesaian tugas yang diberikan kepada peserta. Setiap peserta diwakili oleh struktur data `Contestant`, yang menyimpan nama peserta, waktu penyelesaian untuk setiap tugas, total skor, dan jumlah tugas yang berhasil diselesaikan. Fungsi `calculateScore` digunakan untuk menghitung total waktu yang dihabiskan oleh peserta, mengabaikan waktu 301 detik yang dianggap tidak valid. Program ini kemudian membandingkan skor dan jumlah tugas yang diselesaikan oleh setiap peserta untuk menentukan pemenang, dengan mempertimbangkan jumlah tugas yang diselesaikan terlebih dahulu, dan jika ada kesamaan, mempertimbangkan skor total. Hasil akhir mencetak nama pemenang, skor total, dan jumlah tugas yang diselesaikan.

### No 3. Source Code

```
package main
```

```
import (  
    "fmt"  
)
```

```
func cetakDeret(n int) {  
    for n != 1 {  
        fmt.Printf("%d ", n)  
        if n%2 == 0 {  
            n = n / 2  
        } else {
```

```

        n = 3*n + 1
    }
}

fmt.Println(n) // Mencetak suku terakhir, yaitu 1
}

func main() {
    var n int

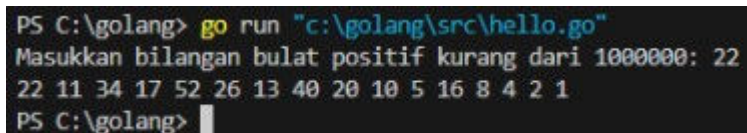
    fmt.Print("Masukkan bilangan bulat positif kurang dari 1000000: ")

    fmt.Scan(&n)

    if n > 0 && n < 1000000 {
        cetakDeret(n)
    } else {
        fmt.Println("Bilangan harus positif dan kurang dari 1000000")
    }
}

```

### Screenshoot Output



```

PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan bilangan bulat positif kurang dari 1000000: 22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS C:\golang>

```

### Deskripsi Program

Program ini berfungsi untuk mencetak deret angka berdasarkan aturan Collatz, yang dimulai dari bilangan bulat positif yang diinputkan oleh pengguna. Pengguna diminta untuk memasukkan bilangan bulat positif yang kurang dari 1.000.000. Setelah input diterima, program akan menjalankan fungsi cetakDeret, yang menerapkan aturan: jika angka genap, angka tersebut dibagi dua; jika angka ganjil,

angka tersebut dikalikan dengan tiga dan ditambah satu. Proses ini berlanjut hingga angka mencapai satu, dengan setiap angka yang dihasilkan dicetak ke layar. Program memastikan bahwa input yang diberikan memenuhi syarat yang ditentukan sebelum melakukan perhitungan dan pencetakan deret. Jika input tidak valid, program akan menampilkan pesan kesalahan.