

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2  
MODUL 3  
MATERI**



Oleh:

**DAFFA TSAQIFNA FAUZTSANY**

**103112400032**

**S1 IF-12-01**

**S1 TEKNIK INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO  
2025**

## I. DASAR TEORI

### 1. Fungsi

Fungsi adalah bagian program yang menerima input dan mengembalikan nilai. Fungsinya membantu membuat program lebih terstruktur dan mudah digunakan kembali.

### 2. Ciri Fungsi:

- Ada nama fungsi
- Memiliki parameter (input)
- Mengembalikan nilai (menggunakan return)
- Ditulis dengan func

Contoh Penulisan:

```
func luasPersegi(sisi int) int {  
    return sisi * sisi  
}
```

Pemanggilan Fungsi:

```
hasil := luasPersegi(5)  
fmt.Println(hasil)
```

Contoh Lain:

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

## II. GUIDED

Source Code + Screenshot hasil program beserta penjelasan

### 1. GUIDED 1

Source Code:

```
package main

import "fmt"

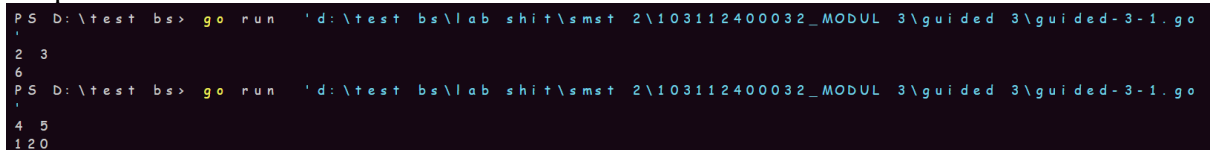
func main() {
    var a, b int
    fmt.Scan(&a, &b)

    if a >= b {
        fmt.Println(permutasi(a, b))
    } else {
        fmt.Println(permutasi(b, a))
    }
}

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

func permutasi(n, r int) int {
    if r > n {
        return 0
    }
    return faktorial(n) / faktorial(n-r)
}
```

Output:



```
PS D:\test bs> go run 'd:\test bs\lab shit\smst 2\103112400032_MODUL 3\guided 3\guided-3-1.go'
2 3
6
PS D:\test bs> go run 'd:\test bs\lab shit\smst 2\103112400032_MODUL 3\guided 3\guided-3-1.go'
4 5
120
```

Deskripsi Program:

Program ini digunakan untuk menghitung nilai permutasi dari dua bilangan bulat  $n$  dan  $r$  (dalam bentuk  $P(n, r) = n! / (n - r)!$ ). Program memastikan bahwa nilai  $n \geq r$ , lalu memanggil fungsi `permutasi()` untuk menghitung hasilnya. Jika  $r > n$ , maka hasil permutasi dianggap 0.

#### 1. Deklarasi Variabel dan Input:

```
var a, b int
fmt.Scan(&a, &b)
```

- a, b: Dua bilangan bulat yang dimasukkan oleh pengguna.
- Digunakan untuk menentukan nilai n dan r.

## 2. Logika Perbandingan:

```
if a >= b {
    fmt.Println(permutasi(a, b))
} else {
    fmt.Println(permutasi(b, a))
}
```

- Program secara otomatis menyesuaikan agar nilai yang lebih besar digunakan sebagai n, dan nilai yang lebih kecil sebagai r.
- Ini memastikan bahwa kondisi  $n \geq r$  selalu terpenuhi.

## 3. Fungsi faktorial:

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

- Menghitung nilai faktorial n! dengan perulangan.
- Misal: faktorial(5) akan menghasilkan  $5 \times 4 \times 3 \times 2 \times 1 = 120$ .

## 4. Fungsi permutasi:

```
func permutasi(n, r int) int {
    if r > n {
        return 0
    }
    return faktorial(n) / faktorial(n-r)
}
```

- Menghitung permutasi menggunakan rumus  $P(n, r) = n! / (n-r)!$ .
- Jika  $r > n$ , maka hasilnya dianggap tidak valid (return 0).

Contoh:

```
5 3 (input)
60 (output)
```

## 2. GUIDED 2

Source Code:

```
package main

import (
    "fmt"
)
```

```

func celsiusToFahrenheit(celsius float64) float64 {
    return (9.0/5.0)*celsius + 32
}

func main() {
    var N int
    fmt.Print("Masukkan jumlah data: ")
    _, err := fmt.Scan(&N)
    if err != nil || N <= 0 {
        fmt.Println("Input tidak valid, pastikan memasukkan angka positif.")
        return
    }

    temperatures := make([]float64, N)

    fmt.Println("Masukkan suhu dalam Celsius:")
    for i := 0; i < N; i++ {
        _, err := fmt.Scan(&temperatures[i])
        if err != nil {
            fmt.Println("Input tidak valid, pastikan memasukkan angka.")
            return
        }
    }

    fmt.Println("Suhu dalam Fahrenheit:")
    for _, temp := range temperatures {
        fmt.Printf("%.2f\n", celsiusToFahrenheit(temp))
    }
}

```

Output:

```

PS D:\test bs> go run 'd:\test bs\lab shit\sms1 2\103112400032_MODUL 3\guided 3\guided-3-2.go'
Masukkan jumlah data: 4
Masukkan suhu dalam Celsius:
18
20
22
24
Suhu dalam Fahrenheit:
64.40
68.00
71.60
75.20

```

### Deskripsi Program:

Program ini digunakan untuk mengonversi sejumlah suhu dari satuan Celsius ke Fahrenheit. Pengguna diminta untuk menentukan jumlah data suhu yang akan dikonversi, lalu memasukkan suhu-suhu tersebut satu per satu. Program menggunakan fungsi `celsiusToFahrenheit()` untuk menghitung hasil konversi, dan mencetak hasilnya dengan dua angka di belakang koma.

#### 1. Deklarasi Fungsi:

```

func celsiusToFahrenheit(celsius float64) float64 {
    return (9.0/5.0)*celsius + 32
}

```

```
}
```

- Fungsi ini menerima satu parameter bertipe float64 (suhu dalam Celsius).
- Rumus konversi:  $F = (9/5 \times C) + 32$ .
- Mengembalikan hasil konversi dalam satuan Fahrenheit.

Alur Program main

## 2. Input Jumlah Data:

```
var N int
fmt.Print("Masukkan jumlah data: ")
_, err := fmt.Scan(&N)
```

- N: Menyimpan jumlah data suhu yang akan dimasukkan pengguna.
- Validasi input:
  - Jika err terjadi atau  $N \leq 0$ , program akan menghentikan eksekusi dan mencetak pesan kesalahan.

## 3. Input Suhu:

```
temperatures := make([]float64, N)
```

- Slice temperatures digunakan untuk menyimpan data suhu sebanyak N.

```
for i := 0; i < N; i++ {
    _, err := fmt.Scan(&temperatures[i])
    if err != nil {
        fmt.Println("Input tidak valid, pastikan memasukkan angka.")
        return
    }
}
```

- Program membaca N suhu dalam satuan Celsius dari pengguna dan menyimpannya ke dalam slice.
- Jika terjadi kesalahan input (misal bukan angka), program akan berhenti.

## 4. Menampilkan Hasil Konversi:

```
for _, temp := range temperatures {
    fmt.Printf("%.2f\n", celsiusToFahrenheit(temp))
}
```

- Program mencetak hasil konversi untuk setiap suhu dalam Fahrenheit dengan format dua angka di belakang koma (%.2f).

Contoh:

Masukkan jumlah data: 4 (input)

Masukkan suhu dalam Celsius: (output)

18 (input)

20 (input)

22 (input)

24 (input)

Suhu dalam Fahrenheit: (output)

64.40 (output)

68.00 (output)

71.60 (output)

75.20 (output)

### 3. GUIDED 3

Source Code:

```
package main

import (
    "fmt"
    "math"
)

func luasPermukaanTabung(r, t float64) float64 {
    return 2 * math.Pi * r * (r + t)
}

func volumeTabung(r, t float64) float64 {
    return math.Pi * math.Pow(r, 2) * t
}

func main() {
    var r, t float64

    fmt.Print("Masukkan jari-jari tabung: ")
    _, errR := fmt.Scan(&r)
    fmt.Print("Masukkan tinggi tabung: ")
    _, errT := fmt.Scan(&t)
    if errR != nil || errT != nil {
        fmt.Println("Input tidak valid! Harap masukkan angka yang benar.")
        return
    }
    if r <= 0 || t <= 0 {
        fmt.Println("Jari-jari dan tinggi tabung harus lebih dari nol.")
        return
    }
    luas := luasPermukaanTabung(r, t)
    volume := volumeTabung(r, t)
    fmt.Println("=====")
    fmt.Printf("Luas Permukaan Tabung: %.2f satuan²\n", luas)
    fmt.Printf("Volume Tabung: %.2f satuan³\n", volume)
    fmt.Println("=====")
}
```

Output:

```
PS D:\test bs> go run 'd:\test bs\lab shift\smsr 2\103112400032_MODUL 3\guided 3\guided-3-3.go'
Masukkan jari-jari tabung: 4
Masukkan tinggi tabung: 10
=====
Luas Permukaan Tabung: 351.86 satuan²
Volume Tabung: 502.65 satuan³
=====
```

### Deskripsi Program:

Program ini digunakan untuk menghitung luas permukaan dan volume tabung berdasarkan nilai jari-jari (r) dan tinggi (t) yang diinput oleh pengguna. Program memvalidasi input agar tidak bernilai nol atau negatif dan menampilkan hasil dengan dua angka di belakang koma.

#### 1. Deklarasi Fungsi:

- Luas Permukaan Tabung:

```
func luasPermukaanTabung(r, t float64) float64 {
    return 2 * math.Pi * r * (r + t)
}
```

- Menggunakan rumus:  $2\pi r(r + t)$
- r: Jari-jari alas.
- t: Tinggi tabung.
- Volume Tabung:

```
func volumeTabung(r, t float64) float64 {
    return math.Pi * math.Pow(r, 2) * t
}
```

- Menggunakan rumus:  $\pi r^2 t$
- `math.Pow(r, 2)`: Menghitung kuadrat dari r.

#### 2. Alur Program main:

- Input:

```
var r, t float64
fmt.Print("Masukkan jari-jari tabung: ")
_, errR := fmt.Scan(&r)
fmt.Print("Masukkan tinggi tabung: ")
_, errT := fmt.Scan(&t)
```

- Input dua nilai bertipe float64: jari-jari (r) dan tinggi (t).
- Validasi input dilakukan dengan mengecek apakah terjadi error pada Scan atau nilai kurang dari/sama dengan 0.
- Validasi:

```
if errR != nil || errT != nil {
    fmt.Println("Input tidak valid! Harap masukkan
    angka yang benar.")
    return
}
if r <= 0 || t <= 0 {
    fmt.Println("Jari-jari dan tinggi tabung harus lebih
    dari nol.")
    return
}
```



- Jika ada kesalahan input atau nilai tidak logis ( $\leq 0$ ), program akan menghentikan proses dan memberikan pesan.
- Hitung dan Tampilkan Hasil:

```
luas := luasPermukaanTabung(r, t)
volume := volumeTabung(r, t)

fmt.Println("=====")
fmt.Printf("Luas Permukaan Tabung: %.2f satuan²\n", luas)
fmt.Printf("Volume Tabung: %.2f satuan³\n", volume)
fmt.Println("=====")
```

- Menampilkan hasil luas permukaan dan volume dengan format rapi

Contoh:

```
Masukkan jari-jari tabung: 4 (input)
Masukkan tinggi tabung: 10 (input)
===== (output)
Luas Permukaan Tabung: 351.86 satuan² (output)
Volume Tabung: 502.65 satuan³ (output)
===== (output)
```

### III. UNGUIDED

Source Code + Screenshot hasil program beserta penjelasan

#### 1. UNGUIDED 1

Source Code:

```
package main

import "fmt"

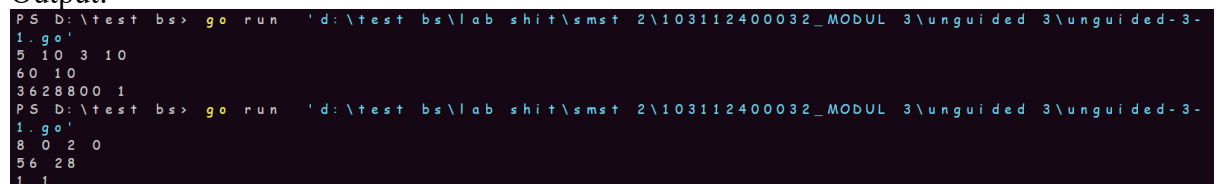
func faktorial(x int) int {
    var out int
    out = 1
    for i := 1; i <= x; i++ {
        out *= i
    }
    return out
}

func permutasi(x, y int) int {
    return faktorial(x) / faktorial(x-y)
}

func kombinasi(x, y int) int {
    return faktorial(x) / (faktorial(y) * faktorial(x-y))
}

func main() {
    var a, b, c, d int
    fmt.Scan(&a, &b, &c, &d)
    fmt.Println(permutasi(a, c), kombinasi(a, c))
    fmt.Print(permutasi(b, d), kombinasi(b, d))
}
```

Output:



```
P S D:\test bs> go run 'd:\test bs\lab shit\sms 2\103112400032_MODUL 3\unguided 3\unguided-3-1.go'
5 10 3 10
60 10
3628800 1
P S D:\test bs> go run 'd:\test bs\lab shit\sms 2\103112400032_MODUL 3\unguided 3\unguided-3-1.go'
8 0 2 0
56 28
1 1
```

Deskripsi Program:

Program ini digunakan untuk menghitung nilai permutasi (P) dan kombinasi (C) dari dua set pasangan bilangan. Program menerima empat input: dua bilangan untuk pasangan pertama (a dan c), dan dua bilangan untuk pasangan kedua (b dan d). Selanjutnya, program mencetak hasil permutasi dan kombinasi dari masing-masing pasangan menggunakan rumus matematika.

1. Deklarasi Fungsi:

- Fungsi Faktorial:

```
func faktorial(x int) int {
```

```

var out int
out = 1
for i := 1; i <= x; i++ {
    out *= i
}
return out
}

```

- Menghitung faktorial  $x!$  dengan menggunakan perulangan for.
- Misal:  $\text{faktorial}(4) = 4 \times 3 \times 2 \times 1 = 24$ .
- Fungsi Permutasi:

```

func permutasi(x, y int) int {
    return faktorial(x) / faktorial(x-y)
}

```

- Menggunakan rumus  $P(x, y) = x! / (x - y)!$
- Menghitung banyaknya urutan berbeda dari  $y$  elemen yang diambil dari  $x$ .
- Fungsi Kombinasi:

```

func kombinasi(x, y int) int {
    return faktorial(x) / (faktorial(y) * faktorial(x-y))
}

```

- Menggunakan rumus  $C(x, y) = x! / (y! * (x - y)!)$
2. Alur Program main:

- Input:

```

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

```

- Pengguna memasukkan empat bilangan:
  - $a, c \rightarrow$  pasangan pertama untuk permutasi dan kombinasi.
  - $b, d \rightarrow$  pasangan kedua.

```

fmt.Println(permutasi(a, c), kombinasi(a, c))
fmt.Print(permutasi(b, d), kombinasi(b, d))

```

- Menampilkan hasil permutasi dan kombinasi dari kedua pasangan.
- Output ditampilkan dalam satu baris untuk setiap pasangan.

Contoh:

```

5 10 3 10 (input)
60 10 (output)
3628800 1 (output)

```

## 2. UNGUIDED 2

Source Code:

```

package main

import "fmt"

```

```

func fx(x int) int {
    return x * x
}
func gx(x int) int {
    return x - 2
}
func hx(x int) int {
    return x + 1
}
func main() {
    var a, b, c int
    fmt.Scan(&a, &b, &c)
    fmt.Println(fx(gx(hx(a))))
    fmt.Println(gx(hx(fx(b))))
    fmt.Println(hx(fx(gx(c))))
}

```

Output:

```

PS D:\test bs> go run 'd:\test bs\lab shit\smst 2\103112400032_MODUL 3\unguided 3\unguided-3-2.go'
7 2 10
36
3
65
PS D:\test bs> go run 'd:\test bs\lab shit\smst 2\103112400032_MODUL 3\unguided 3\unguided-3-2.go'
5 5 5
16
24
10
PS D:\test bs> go run 'd:\test bs\lab shit\smst 2\103112400032_MODUL 3\unguided 3\unguided-3-2.go'
3 8 4
4
63
5
PS D:\test bs> go run 'd:\test bs\lab shit\smst 2\103112400032_MODUL 1&2\unguided 1\unguided-1&2-2.go'
nilai K = 10
nilai f(k) = 1.4054086752
PS D:\test bs> go run 'd:\test bs\lab shit\smst 2\103112400032_MODUL 1&2\unguided 1\unguided-1&2-2.go'
nilai K = 100
nilai f(k) = 1.4133299615
PS D:\test bs> go run 'd:\test bs\lab shit\smst 2\103112400032_MODUL 1&2\unguided 1\unguided-1&2-2.go'
nilai K = 1000
nilai f(k) = 1.4141251768

```

Deskripsi Program:

Program ini menunjukkan cara komposisi fungsi dalam pemrograman menggunakan tiga fungsi matematika sederhana:

- $fx(x) = x^2$
- $gx(x) = x - 2$
- $hx(x) = x + 1$

Pengguna diminta memasukkan tiga bilangan, kemudian program menghitung dan mencetak hasil dari tiga komposisi fungsi dengan urutan berbeda.

1. Deklarasi Fungsi:

- Fungsi fx:

```

func fx(x int) int {
    return x * x
}

```

```
}
```

- Mengembalikan kuadrat dari nilai x.

- Fungsi gx:

```
func gx(x int) int {  
    return x - 2  
}
```

- Mengembalikan hasil pengurangan x dengan 2.

- Fungsi hx:

```
func hx(x int) int {  
    return x + 1  
}
```

- Mengembalikan hasil penambahan x dengan 1.

## 2. Fungsi main:

- Input:

```
var a, b, c int  
fmt.Scan(&a, &b, &c)
```

- Program menerima tiga bilangan dari input: a, b, dan c.

- Komposisi Fungsi dan Output:

```
fmt.Println(fx(gx(hx(a))))
```

- Menghitung  $f(g(h(a)))$

- Langkah:

- $hx(a) = a + 1$
- $gx(\dots) = \text{hasil} - 2$
- $fx(\dots) = \text{hasil}^2$

```
fmt.Println(gx(hx(fx(b))))
```

- Menghitung  $g(h(f(b)))$

- Langkah:

- $fx(b) = b^2$
- $hx(\dots) = \text{hasil} + 1$
- $gx(\dots) = \text{hasil} - 2$

```
fmt.Println(hx(fx(gx(c))))
```

- Menghitung  $h(f(g(c)))$

- Langkah:

- $gx(c) = c - 2$
- $fx(...) = \text{hasil}^2$
- $hx(...) = \text{hasil} + 1$

Contoh:

5 5 5 (input)  
16 (output)  
24 (output)  
10 (output)

### 3. UNGUIDED 3

Source Code:

```
package main

import (
    "fmt"
    "math"
)

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

func main() {
    var cx1, cx2, cy1, cy2, r1, r2, x, y float64
    fmt.Scan(&cx1, &cy1, &r1)
    fmt.Scan(&cx2, &cy2, &r2)
    fmt.Scan(&x, &y)
    if jarak(cx1, cy1, x, y) <= r1 && jarak(cx2, cy2, x, y) <= r2 {
        fmt.Println("Titik di dalam lingkaran 1 dan 2")
    } else if jarak(cx1, cy1, x, y) <= r1 {
        fmt.Println("Titik di dalam lingkaran 1")
    } else if jarak(cx2, cy2, x, y) <= r2 {
        fmt.Println("Titik di dalam lingkaran 2")
    } else {
        fmt.Println("Titik di luar lingkaran 1 dan 2")
    }
}
```

Output:

```

PS D:\test bs> go run 'd:\test bs\lab shit\smst 2\103112400032_MODUL 3\unguided 3\unguided-3-3.go'
1 1 5
8 8 4
2 2
Titik di dalam lingkaran 1
PS D:\test bs> go run 'd:\test bs\lab shit\smst 2\103112400032_MODUL 3\unguided 3\unguided-3-3.go'
1 2 3
4 5 6
7 8
Titik di dalam lingkaran 2
PS D:\test bs> go run 'd:\test bs\lab shit\smst 2\103112400032_MODUL 3\unguided 3\unguided-3-3.go'
5 10 15
-15 4 20
0 0
Titik di dalam lingkaran 1 dan 2
PS D:\test bs> go run 'd:\test bs\lab shit\smst 2\103112400032_MODUL 3\unguided 3\unguided-3-3.go'
1 1 5
8 8 4
15 20
Titik di luar lingkaran 1 dan 2

```

### Deskripsi Program:

Program ini digunakan untuk menentukan apakah sebuah titik (x, y) berada di dalam, di luar, atau hanya dalam salah satu dari dua lingkaran. Program menerima input koordinat pusat dan jari-jari dari dua lingkaran, serta koordinat titik yang ingin diperiksa. Perhitungan jarak antara titik dan pusat lingkaran dilakukan menggunakan rumus Euclidean (jarak antar dua titik).

#### 1. Fungsi jarak:

```

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

```

- Menghitung jarak Euclidean antara dua titik (a, b) dan (c, d) menggunakan rumus:

$$\text{jarak} = \sqrt{(a - c)^2 + (b - d)^2}$$

#### 2. Deklarasi dan Input di main:

```

var cx1, cy1, cx2, cy2, r1, r2, x, y float64
fmt.Scan(&cx1, &cy1, &r1)
fmt.Scan(&cx2, &cy2, &r2)
fmt.Scan(&x, &y)

```

- cx1, cy1: Koordinat pusat lingkaran 1.
- r1: Jari-jari lingkaran 1.
- cx2, cy2: Koordinat pusat lingkaran 2.
- r2: Jari-jari lingkaran 2.
- x, y: Titik yang akan dicek.

#### 3. Logika Pengecekan Titik:

```

if jarak(cx1, cy1, x, y) <= r1 && jarak(cx2, cy2, x, y) <= r2 {
    fmt.Println("Titik di dalam lingkaran 1 dan 2")
} else if jarak(cx1, cy1, x, y) <= r1 {
    fmt.Println("Titik di dalam lingkaran 1")
} else if jarak(cx2, cy2, x, y) <= r2 {
    fmt.Println("Titik di dalam lingkaran 2")
} else {
    fmt.Println("Titik di luar lingkaran 1 dan 2")
}

```

- Menggunakan fungsi jarak() untuk membandingkan jarak titik ke pusat lingkaran dengan jari-jari:

- Jika lebih kecil dari atau sama dengan jari-jari  $\rightarrow$  titik berada di dalam lingkaran.
- Jika tidak  $\rightarrow$  titik di luar.

Contoh:

1 1 5 (input)

8 8 4 (input)

15 20 (input)

Titik di luar lingkaran 1 dan 2 (output)