

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 4  
PROSEDUR**



Oleh:

NAMA: Lutfi Shidqi Mardian

NIM: 103112400077

KELAS: IF-12-01

**S1 TEKNIK INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO**

**2025**

## I. DASAR TEORI

Dalam pemrograman, penggunaan subprogram seperti fungsi dan prosedur sangat penting untuk meningkatkan modularitas dan keterbacaan kode. Pada bahasa pemrograman Go (Golang), baik fungsi maupun prosedur digunakan untuk memecah kode menjadi bagian yang lebih kecil dan lebih terorganisir.

### 1. Pengertian Function

adalah subprogram yang menerima parameter, melakukan operasi tertentu, dan mengembalikan hasil. Suatu subprogram dikategorikan sebagai fungsi jika memenuhi dua syarat utama:

- **Memiliki deklarasi tipe nilai yang dikembalikan.**
- **Menggunakan kata kunci return dalam badan fungsi untuk mengembalikan nilai.**

Sebagai contoh, berikut adalah deklarasi fungsi dalam Golang:

```
func hitungLuasPersegiPanjang(panjang, lebar  
int) int {  
    return panjang * lebar  
}
```

Fungsi `hitungLuasPersegiPanjang` menerima dua parameter, melakukan perhitungan, dan mengembalikan hasil.

**Prosedur** adalah subprogram yang melakukan suatu aksi tetapi tidak mengembalikan nilai. Suatu subprogram dikategorikan sebagai prosedur jika:

- **Tidak memiliki deklarasi tipe nilai yang dikembalikan.**
- **Tidak menggunakan kata kunci return.**

Contoh prosedur dalam Golang:

```
func cetakPesan(pesan string) {  
    fmt.Println(pesan)  
}
```

Prosedur ini hanya mencetak pesan tanpa mengembalikan nilai apa pun.

## 2. Deklarasi dan Pemanggilan Fungsi

Deklarasi fungsi dilakukan dengan menentukan nama, parameter, dan tipe pengembalian. Pemanggilan fungsi dilakukan dengan menyebutkan nama fungsi dan memberikan argumen sesuai dengan parameter yang telah dideklarasikan.

Contoh pemanggilan fungsi:

```
func main() {  
  
    luas := hitungLuasPersegiPanjang(5, 10)  
  
    fmt.Println("Luas Persegi Panjang:", luas)  
  
}
```

## 3. Deklarasi dan Pemanggilan Prosedur

Prosedur dapat dideklarasikan dan dipanggil dalam program utama untuk melakukan tugas tertentu tanpa mengembalikan nilai.

Contoh pemanggilan prosedur:

```
func main() {  
  
    cetakPesan("Selamat datang di program ini!")  
  
}
```

## 4. Parameter dalam Fungsi dan Prosedur

Parameter dalam fungsi dan prosedur dapat dibagi menjadi dua jenis berdasarkan cara pengirimannya:

- **Pass by Value:** Parameter formal mendapatkan salinan nilai dari parameter aktual.
- **Pass by Reference:** Parameter formal merujuk langsung ke alamat memori parameter aktual menggunakan pointer.

Contoh pass by reference dalam Golang:

```
func ubahNilai(a *int) {  
    *a = 10  
}  
  
func main() {  
    var angka int = 5  
    ubahNilai(&angka)  
    fmt.Println("Nilai setelah diubah:", angka) // Output:  
10  
}
```

## 5. Penggunaan Fungsi dan Prosedur dalam Program

Fungsi dan prosedur banyak digunakan untuk menghindari duplikasi kode, meningkatkan efisiensi, dan membuat program lebih mudah dikelola. Contoh implementasi dalam perhitungan permutasi:

```
func faktorial(n int) int {  
    hasil := 1  
    for i := 1; i <= n; i++ {  
        hasil *= i  
    }  
    return hasil  
}  
  
func permutasi(n, r int) int {  
    return faktorial(n) / faktorial(n-r)  
}  
  
func main() {  
    fmt.Println(permutasi(5, 3)) // Output: 60  
}
```

## II. GUIDED

### 1.

```
package main

import "fmt"

func hitungGaji(nama string, gajiPokok float64, jamLembur
int) {

    bonusLembur := float64(jamLembur) * 50000

    totalGaji := gajiPokok + bonusLembur

    fmt.Println("\n=== Slip Gaji ===")
    fmt.Println("Nama karyawan:", nama)
    fmt.Printf("Gaji pokok: Rp.%2f\n", gajiPokok)
    fmt.Printf("Bonus lembur: Rp.%2f (%d jam x
Rp.50,000)\n", bonusLembur, jamLembur)
    fmt.Printf("Total gaji: Rp.%2f\n", totalGaji)
}

func main() {

    var nama string
    var gajiPokok float64
    var jamLembur int

    fmt.Print("Masukkan nama karyawan:")
    fmt.Scan(&nama)
```

```

    fmt.Print("Masukkan gaji pokok:")

    fmt.Scan(&gajiPokok)

    fmt.Print("Masukkan jam lembur:")

    fmt.Scan(&jamLembur)

    hitungGaji(nama, gajiPokok, jamLembur)

}

```

#### Output Screenshot:



```

PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run kModul4\guided1.go
Masukkan nama karyawan:Lutfi
Masukkan gaji pokok:3000000
Masukkan jam lembur:3

=== Slip Gaji ===
Nama karyawan: Lutfi
Gaji pokok: Rp.3000000.000000
Bonus lembur: Rp.150000.000000 (3 jam x Rp.50,000)
Total gaji: Rp.3150000.000000
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>

```

#### Penjelasan:

Program ini menghitung dan mencetak total gaji karyawan berdasarkan gaji pokok dan jumlah jam lembur yang dimasukkan pengguna. Pengguna memberikan nama karyawan, gaji pokok, dan jumlah jam lembur, lalu program menghitung bonus lembur dengan tarif Rp.50.000 per jam dan menambahkan hasilnya ke gaji pokok untuk mendapatkan total gaji. Fungsi `hitungGaji(nama, gajiPokok, jamLembur)` digunakan untuk melakukan perhitungan dan mencetak slip gaji dengan format yang rapi. Semua nilai ditampilkan menggunakan `fmt.Printf` agar hasil lebih mudah dibaca.

## 2.

```
package main

import "fmt"

func hitungRataRata(nama string, n1, n2, n3 float64) {

    ratarata := (n1 + n2 + n3) / 3
    status := "Tidak Lulus"

    if ratarata >= 60 {
        status = "Lulus"
    }

    fmt.Println("\n=== Hasil Akademik ===")
    fmt.Println("Nama Mahasiswa :", nama)
    fmt.Printf("Nilai 1 : %.2f\n", n1)
    fmt.Printf("Nilai 2 : %.2f\n", n2)
    fmt.Printf("Nilai 3 : %.2f\n", n3)
    fmt.Printf("Rata-rata : %.2f\n", ratarata)
    fmt.Println("Status:", status)
}

func main() {
    var nama string
    var n1, n2, n3 float64
```

```

    fmt.Print("Masukkan Nama Mahasiswa: ")
    fmt.Scan(&nama)

    fmt.Print("Masukkan Nilai 1: ")
    fmt.Scan(&n1)

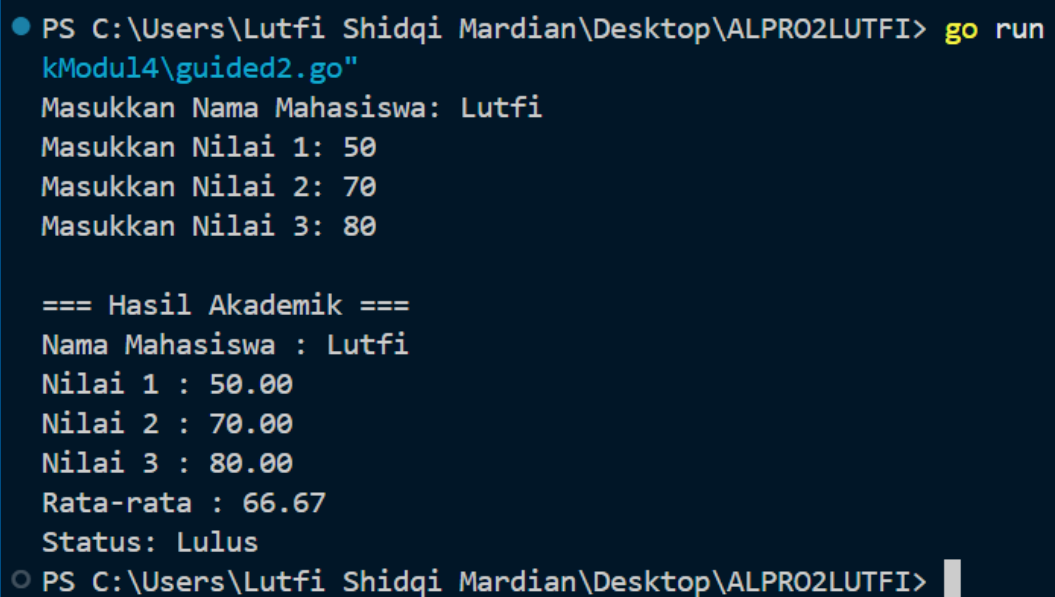
    fmt.Print("Masukkan Nilai 2: ")
    fmt.Scan(&n2)

    fmt.Print("Masukkan Nilai 3: ")
    fmt.Scan(&n3)

    hitungRataRata(nama, n1, n2, n3)
}

```

#### Output Screenshot:



```

● PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
  kModul4\guided2.go"
Masukkan Nama Mahasiswa: Lutfi
Masukkan Nilai 1: 50
Masukkan Nilai 2: 70
Masukkan Nilai 3: 80

=== Hasil Akademik ===
Nama Mahasiswa : Lutfi
Nilai 1 : 50.00
Nilai 2 : 70.00
Nilai 3 : 80.00
Rata-rata : 66.67
Status: Lulus
○ PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>

```



### Penjelasan:

Program ini menghitung dan mencetak hasil akademik mahasiswa berdasarkan tiga nilai yang dimasukkan pengguna. Pengguna memberikan nama mahasiswa serta tiga nilai ujian, lalu program menghitung rata-rata dari ketiga nilai tersebut. Jika rata-rata nilai mencapai 60 atau lebih, mahasiswa dinyatakan "Lulus", sedangkan jika kurang dari 60, mahasiswa dinyatakan "Tidak Lulus". Fungsi `hitungRataRata(nama, n1, n2, n3)` digunakan untuk melakukan perhitungan dan mencetak hasil akademik dengan format yang rapi. Semua nilai ditampilkan dengan dua angka di belakang desimal menggunakan `fmt.Printf` agar lebih mudah dibaca.

## III. UNGUIDED

### 1.

```
package main

import "fmt"

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

    if a >= c && b >= d {
        fmt.Printf("%d %d\n", permutasi(a, c), kombinasi(a,
c))
        fmt.Printf("%d %d\n", permutasi(b, d), kombinasi(b,
d))
    }
}

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)
}

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

```

#### Output Screenshot:

```

● PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run kModul4\unguided1.go"
5 10 3 10
60 10
3628800 1
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run kModul4\unguided1.go"
● 8 0 2 0
56 28
1 1
○ PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>

```

### Penjelasan:

Program ini menghitung dan mencetak nilai permutasi serta kombinasi dari dua pasang bilangan yang dimasukkan pengguna. Pengguna memberikan empat bilangan bulat  $a, b, c, d$ , lalu program memastikan bahwa  $a \geq c$  dan  $b \geq d$  sebelum melakukan perhitungan. Jika kondisi terpenuhi, program menghitung dan mencetak nilai permutasi dan kombinasi untuk pasangan  $(a, c)$  serta  $(b, d)$ . Fungsi faktorial( $n$ ) digunakan untuk menghitung faktorial suatu bilangan, sementara fungsi permutasi( $n, r$ ) dan kombinasi( $n, r$ ) digunakan untuk menghitung permutasi dan kombinasi berdasarkan rumus matematika yang berlaku. Jika  $r$  lebih besar dari  $n$ , hasil perhitungan dikembalikan sebagai 0.

## 2.

```
package main

import "fmt"

func hitungSkor(w1, w2, w3, w4, w5, w6, w7, w8 int, soalSelesai,
totalWaktu *int) {

    *soalSelesai, *totalWaktu = 0, 0

    if w1 < 301 {

        *soalSelesai++

        *totalWaktu += w1

    }

    if w2 < 301 {

        *soalSelesai++

        *totalWaktu += w2

    }

}
```

```
    if w3 < 301 {  
        *soalSelesai++  
        *totalWaktu += w3  
    }  
    if w4 < 301 {  
        *soalSelesai++  
        *totalWaktu += w4  
    }  
    if w5 < 301 {  
        *soalSelesai++  
        *totalWaktu += w5  
    }  
    if w6 < 301 {  
        *soalSelesai++  
        *totalWaktu += w6  
    }  
    if w7 < 301 {  
        *soalSelesai++  
        *totalWaktu += w7  
    }  
    if w8 < 301 {  
        *soalSelesai++  
        *totalWaktu += w8  
    }  
}  
  
func main() {
```

```

var nama, pemenang string

var w1, w2, w3, w4, w5, w6, w7, w8 int

var soalSelesai, totalWaktu, maxSoal, minWaktu int

maxSoal = -1

minWaktu = 99999

for {
    fmt.Scan(&nama)

    if nama == "Selesai" {
        break
    }

    fmt.Scan(&w1, &w2, &w3, &w4, &w5, &w6, &w7, &w8)

    hitungSkor(w1, w2, w3, w4, w5, w6, w7, w8, &soalSelesai,
&totalWaktu)

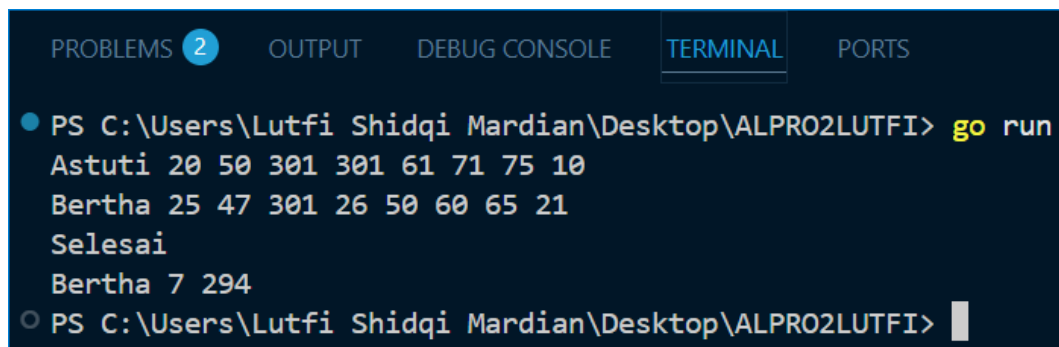
    if soalSelesai > maxSoal || (soalSelesai == maxSoal &&
totalWaktu < minWaktu) {
        maxSoal, minWaktu = soalSelesai, totalWaktu

        pemenang = nama
    }
}

fmt.Print(pemenang, maxSoal, minWaktu)
}

```

### Output Screenshot:

A screenshot of a Go IDE terminal window. The terminal has tabs for PROBLEMS (with a blue circle containing the number 2), OUTPUT, DEBUG CONSOLE, TERMINAL (which is active), and PORTS. The terminal shows the following output:

```
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run  
Astuti 20 50 301 301 61 71 75 10  
Bertha 25 47 301 26 50 60 65 21  
Selesai  
Bertha 7 294  
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>
```

### Penjelasan:

Program ini menentukan pemenang dalam sebuah kompetisi berdasarkan jumlah soal yang berhasil diselesaikan dan total waktu yang digunakan. Setiap peserta memasukkan namanya diikuti dengan waktu penyelesaian untuk 8 soal. Jika waktu penyelesaian sebuah soal kurang dari 301 detik, soal tersebut dianggap selesai, dan waktu tersebut ditambahkan ke total waktu peserta. Program membaca data peserta hingga ditemukan input "Selesai". Pemenang ditentukan berdasarkan peserta yang menyelesaikan soal terbanyak. Jika ada lebih dari satu peserta dengan jumlah soal yang sama, pemenang ditentukan berdasarkan total waktu terkecil. Hasil akhir mencetak nama pemenang, jumlah soal yang diselesaikan, dan total waktu yang digunakan.

### 3.

```
package main  
  
import "fmt"  
  
func skiena( n int) {  
    for n != 1 {  
        fmt.Print(n, " ")  
        if n % 2 == 0 {
```

```

        n /= 2
    } else {
        n = 3 * n + 1
    }
}

fmt.Print(n)
}

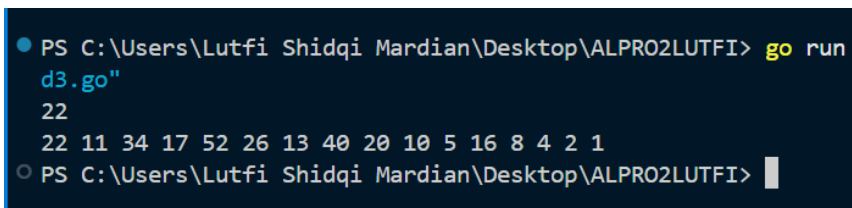
func main() {
    var n int

    fmt.Scan(&n)

    skiena(n)
}

```

#### Output Screenshot:



```

PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run d3.go
22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>

```

#### Penjelasan:

Program ini mencetak urutan bilangan berdasarkan aturan dari *Collatz conjecture* (kadang disebut *Skiena sequence*). Pengguna memasukkan sebuah bilangan bulat  $nn$ , lalu program mencetak deret angka berdasarkan aturan berikut: jika  $nn$  adalah bilangan genap, maka dibagi 2; jika  $nn$  adalah bilangan ganjil, maka dikalikan 3 dan ditambah 1. Proses ini terus berulang hingga  $nn$  mencapai nilai 1.

Fungsi skiena(n) digunakan untuk menjalankan perhitungan dan mencetak hasilnya, sementara fungsi main() bertanggung jawab membaca input dari pengguna.

#### **IV. KESIMPULAN**

Dari berbagai implementasi prosedur dalam program yang telah dibuat, dapat disimpulkan bahwa penggunaan prosedur dalam pemrograman Golang sangat membantu dalam mengurangi kompleksitas kode serta meningkatkan keterbacaan dan efisiensi program. Dengan memisahkan bagian-bagian program ke dalam prosedur, kita dapat menghindari duplikasi kode dan mempermudah pemeliharaan serta debugging. Selain itu, konsep parameter formal dan aktual, serta metode pengiriman parameter seperti pass by value dan pass by reference, memungkinkan komunikasi yang lebih fleksibel antara prosedur dan program utama. Dengan penerapan prosedur yang baik, program menjadi lebih modular, mudah diperbaiki, dan lebih terstruktur.

#### **V. REFERENSI**

1. Donovan, A. A., & Kernighan, B. W. (2015). *The Go Programming Language*. Addison-Wesley.
2. Official Golang Documentation. (n.d.). Functions in Go. Retrieved from [https://go.dev/doc/effective\\_go#functions](https://go.dev/doc/effective_go#functions)
3. W3Schools. (n.d.). Golang Functions. Retrieved from [https://www.w3schools.com/go/go\\_functions.asp](https://www.w3schools.com/go/go_functions.asp)
4. Stack Overflow. (n.d.). Idiomatic way of documenting a Golang program. Retrieved from <https://stackoverflow.com/questions/43638249>
5. GeeksforGeeks. (n.d.). Functions in Golang. Retrieved from <https://www.geeksforgeeks.org/functions-in-golang/>
6. Ardan Labs. (n.d.). Go Functions and Methods. Retrieved from <https://www.ardanlabs.com/blog/>