

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 4**



**DISUSUN OLEH:  
RIZKINA AZIZAH  
103112400082  
S1 IF-12-01**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO  
2025**

## **DASAR TEORI**

### **Pemrograman Prosedural dalam Go**

Pemrograman prosedural merupakan bagian dari pemrograman imperatif di mana program tersebut diorganisasikan ke dalam prosedur, yang juga dikenal sebagai fungsi dalam Go. Setiap prosedur berisi serangkaian langkah yang harus dijalankan

### **Fitur Utama Pemrograman Imperatif/Prosedural di Go:**

1. Alur Kendali yang Jelas: Pemrograman imperatif dalam Go memungkinkan Anda menggunakan pernyataan alur kendali seperti if, for, dan switch untuk menentukan alur eksekusi.
2. Modularitas: Pemrograman prosedural mendorong penggunaan fungsi untuk memecah program menjadi bagian-bagian yang lebih kecil dan mudah dikelola. Hal ini mendorong penggunaan kembali dan pemeliharaan kode.
3. Manipulasi Variabel: Pemrograman imperatif dalam Go melibatkan manipulasi variabel secara langsung untuk menyimpan dan memodifikasi data.
4. Pemrograman Terstruktur: Go mengikuti prinsip pemrograman terstruktur, menekankan kode yang jelas dan terorganisir menggunakan blok, loop, dan prosedur.

### **Pemrograman Bersamaan**

Pemrograman serentak adalah paradigma di mana beberapa tugas dieksekusi secara serentak, yang memungkinkan penggunaan sumber daya yang lebih efisien dan peningkatan responsivitas dalam sistem perangkat lunak. Pemrograman ini melibatkan pemecahan masalah menjadi tugas-tugas yang lebih kecil dan independen yang dapat dieksekusi secara serentak. Gois dirancang dengan dukungan bawaan untuk pemrograman serentak, sehingga sangat cocok untuk mengembangkan sistem yang dapat diskalakan dan serentak.

### **Tinjauan Umum Pemrograman Serentak**

#### **1. Konkurensi vs. Paralelisme:**

- **Concurrency** : Ini tentang mengelola beberapa tugas yang sedang berlangsung, tetapi tidak harus dijalankan pada saat yang bersamaan. Concurrency lebih berkaitan dengan struktur program.
- **Paralelisme** : Melibatkan pelaksanaan beberapa tugas secara bersamaan. Paralelisme adalah cara untuk mencapai konkurensi dengan menjalankan tugas secara bersamaan pada beberapa prosesor.

## 2. Goroutine:

- Dalam Go, tugas bersamaan sering kali diimplementasikan menggunakan goroutine, yang merupakan untaian eksekusi ringan yang dijadwalkan secara independen.
- Goroutine dibuat menggunakan kata kunci `go`, yang memudahkan peluncuran tugas bersamaan.

## 3. Saluran:

- Goroutine berkomunikasi satu sama lain menggunakan saluran, yang merupakan cara untuk melewatkan data di antara tugas yang bersamaan.
- Saluran menyediakan sinkronisasi dan komunikasi antara goroutine.

## 4. Pola Konkurensi:

- Go mendukung berbagai pola konkurensi, seperti *fan-out*, *fan-in*, *worker pool*, dan *pernyataan select* untuk menangani beberapa saluran.

## GUIDED

### 1. Latihan1

Source Code:

```
package main

import "fmt"

func hitungGaji(nama string, gajiPokok float64, jamLembur int) {
    bonusLembur := float64(jamLembur) * 500
    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%f (%d jam x Rp50,000)\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.Scanln(&nama)

    fmt.Print("Masukkan Gaji Pokok: ")
    fmt.Scanln(&gajiPokok)

    fmt.Print("Masukkan Jumlah Jam Lembur: ")
    fmt.Scan(&jamLembur)

    hitungGaji(nama, gajiPokok, jamLembur)
```

```
}
```

#### Deskripsi Program:

- Program ini digunakan untuk menghitung gaji karyawan dengan tambahan bonus lembur berdasarkan jumlah jam kerja tambahan.
- *func hitungGaji*
  - Menerima tiga parameter: nama karyawan, gaji pokok, dan jumlah jam lembur.
  - Menghitung bonus lembur dengan rumus:  $\text{jamLembur} * 500$ .
  - Menghitung total gaji dengan menjumlahkan gaji pokok dan bonus lembur.
  - Mencetak slip gaji dengan informasi lengkap.
- *func main*
  - Input nama, gaji pokok, dan jumlah jam lembur.
  - Memanggil *fungsi hitungGaji*

#### 2. Latihan2

##### Source Code:

```
package main
import "fmt"
//Prosedur untuk menghitung rata-rata dan menentukan kelulusan
func hitungRataRata(nama string, nilai1, nilai2, nilai3 float64) {
    rataRata:=(nilai1+nilai2+nilai3) / 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", nilai1)
    fmt.Printf("Nilai 2 : %.2f\n", nilai2)
    fmt.Printf("Nilai 3 : %.2f\n", nilai3)
    fmt.Printf("Rata-rata : %.2f\n", rataRata)
    fmt.Println("Status :", status)
```

```

}

func main() {
    var nama string
    var nilai1, nilai2, nilai3 float64

    //Input dari pengguna
    fmt.Print("Masukkan Nama Mahasiswa: ")
    fmt.Scanln(&nama)

    fmt.Print("Masukkan Nilai 1: ")
    fmt.Scanln(&nilai1)

    fmt.Print("Masukkan Nilai 2: ")
    fmt.Scanln(&nilai2)

    fmt.Print("Masukkan Nilai 3: ")
    fmt.Scanln(&nilai3)

    // Memanggil prosedur dengan data dari pengguna
    hitungRataRata(nama, nilai1, nilai2, nilai3)
}

```

#### Deskripsi Program:

- Program ini digunakan digunakan untuk menentukan apakah mahasiswa lulus berdasarkan tiga nilai ujian.
- *Prosedur hitungRataRata* berfungsi untuk:
  - Menerima nama mahasiswa dan tiga nilai ujian sebagai parameter.
  - Menghitung rata-rata nilai dengan rumus:  $\text{rata-rata} = \frac{\text{nilai1} + \text{nilai2} + \text{nilai3}}{3}$
  - Menentukan status kelulusan:
    - Jika  $\text{rata-rata} \geq 60$ , maka mahasiswa **Lulus**.
    - Jika  $\text{rata-rata} < 60$ , maka mahasiswa **Tidak Lulus**.

- Menampilkan output (nama, nilai, rata-rata, dan status).
- *func main* berfungsi untuk:
  - Input mahasiswa dan tiga nilai ujian
  - Memanggil *prosedur hitungRataRata*

## UNGUIDED

### 1. Latihan 1

#### Source Code:

```
//RizkinaAzizah
package main

import "fmt"

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

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

```

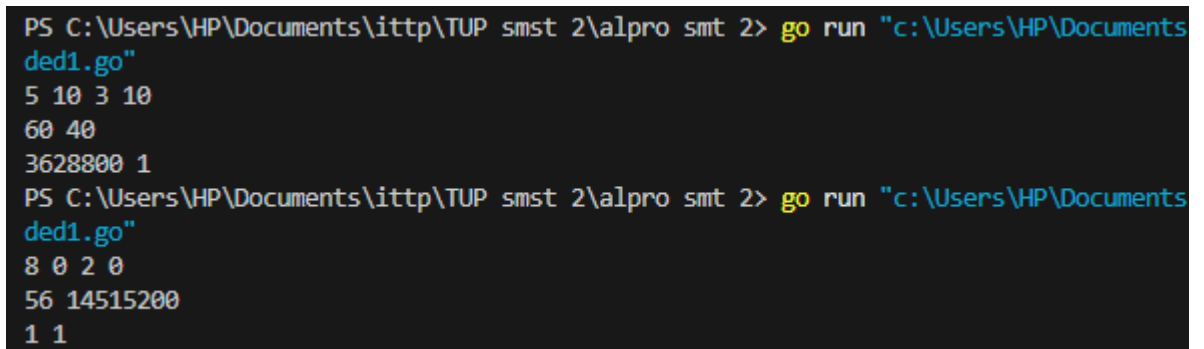
    return hasil
}

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:



```

PS C:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2> go run "c:\Users\HP\Documents\ded1.go"
5 10 3 10
60 40
3628800 1
PS C:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2> go run "c:\Users\HP\Documents\ded1.go"
8 0 2 0
56 14515200
1 1

```

Deskripsi Program:

- Program ini digunakan untuk menentukan kombinasi dan permutasi
- *Fungsi Utama* digunakan untuk:
  - Memasukkan input
  - Menentukan nilai yang lebih besar
  - Memanggil fungsi
  - Menampilkan output •



- *faktorial (n int) int* fungsi ini digunakan untuk menghitung faktorial dari bilangan n menggunakan paradigma perulangan
- *permutasi(n, r int)int* fungsi ini digunakan untuk menghitung permutasi menggunakan rumus  $\text{faktorial}(n)/\text{faktorial}(n-r)$
- *func kombinasi(n,r int)int* fungsi ini digunakan untuk menghitung permutasi menggunakan rumus  $\text{faktorial}(n)/\text{faktorial}(r)*\text{faktorial}(n-r)$

## 2. Latihan 2

### Source Code:

```
//RizkinaAzizah
package main

import (
    "fmt"
)

func hitungSkor(waktu []int) (int, int) {
    totalSoal, totalWaktu := 0, 0
    for _, w := range waktu {
        if w < 301 {
            totalSoal++
            totalWaktu += t
        }
    }
    return totalSoal, totalWaktu
}

func main() {
    var pemenang string
    maxSoal, minWaktu := 0, 99999

    for {
        var nama string
        waktu := make([]int, 8)
```

```

_, err := fmt.Scan(&nama, &waktu[0], &waktu[1], &waktu[2], &waktu[3], &waktu[4],
&waktu[5], &waktu[6], &waktu[7])
if err != nil || nama == "Selesai" {
    break
}

soal, skor := hitungSkor(waktu)
if soal > maxSoal || (soal == maxSoal && skor < minWaktu) {
    pemenang, maxSoal, minWaktu = nama, soal, skor
}
}

fmt.Println(pemenang, maxSoal, minWaktu)
}

```

Output :

```

PS C:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2> go run
"c:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2\1031124000
82_MODUL4\Unguided\103112400082_Unguided2.go"
Astuti 20 50 301 301 61 71 75 10
Bertha 25 47 301 26 50 60 65 21
Selesai
Bertha 7 294

```

Deskripsi Program:

- *func hitungSkor berfungsi untuk:*
  - input waktu pengerjaan 8 soal
  - Mengembalikan jumlah soal yang selesai dan total waktu yang digunakan.
- *Func main berfungsi untuk:*
  - Menggunakan paradigma perulangan untuk membaca input peserta (nama dan waktu pengerjaan).
  - Memanggil *func hitungSkor*.
  - Menentukan pemenang berdasarkan jumlah soal terbanyak atau waktu tercepat.
  - Mencetak hasil: nama pemenang, jumlah soal yang diselesaikan, dan total waktu.

### 3. Latihan 3

#### Source Code:

```
package main

import (
    "fmt"
)

func cetakDeret(n int) {
    for n != 1 {
        fmt.Print(n, " ")
        if n%2 == 0 {
            n /= 2
        } else {
            n = 3*n + 1
        }
    }
    fmt.Println(1)
}

func main() {
    var n int
    fmt.Scan(&n)
    if n > 0 && n < 1000000 {
        cetakDeret(n)
    } else {
        fmt.Println("Masukan harus bilangan positif kurang dari 1000000")
    }
}
```

Output:

```
"c:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2\103112400  
82_MODUL4\Unguided\tempCodeRunnerFile.go"  
22  
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

#### Deskripsi Program:

- *Prosedur cetakDeret* berfungsi untuk:
  - Mencetak deret :
    - Jika **bilangan genap**, maka dibagi **2**.
    - Jika **bilangan ganjil**, maka dikalikan **3** dan ditambah **1**.
  - Proses ini berlanjut **hingga bilangan mencapai 1**.
  - Setiap bilangan dalam deret dicetak dan dipisahkan oleh spasi.
- *Func main* berfungsi untuk:
  - Input satu bilangan positif.
  - Mengecek apakah bilangan tersebut lebih dari 0 dan kurang dari 1.000.000.
  - Jika valid, memanggil *prosedur cetakDeret* untuk mencetak deret.

## DAFTAR PUSTAKA

Zakaria Saif (2023), Panduan Paradigma Pemrograman di Golang (Go)  
(<https://medium.com/@zakariasaiif/guide-to-programming-paradigms-in-golang-go-eff42b678a40> ,2023)