LAPORAN PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN 2

MODUL 4

“PROSEDUR”

DISUSUN OLEH:

RAIHAN ADI ARBA

103112400071

S1 IF-12-01

DOSEN:

Dimas Fanny Hebrasianto Permadi, S.ST., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024/2025

DASAR TEORI

Prosedur dalam bahasa pemrograman Golang merupakan blok kode yang digunakan untuk mengeksekusi serangkaian instruksi tanpa harus mengembalikan nilai. Prosedur didefinisikan menggunakan kata kunci func, sama seperti fungsi, tetapi biasanya tidak memiliki pernyataan return, karena tugas utamanya adalah melakukan operasi tertentu, bukan menghasilkan output. Konsep prosedur sangat penting dalam pemrograman karena membantu menyusun kode secara lebih modular, sehingga setiap bagian kode dapat memiliki tanggung jawab yang spesifik dan lebih mudah dipahami. Dengan pendekatan ini, programmer dapat menghindari duplikasi kode serta meningkatkan keterbacaan program secara keseluruhan.

Dalam implementasinya, prosedur dalam Golang dapat menerima parameter sebagai input, baik dalam bentuk nilai biasa maupun pointer. Penggunaan pointer dalam parameter memungkinkan prosedur untuk langsung memodifikasi nilai dari variabel yang dikirimkan sebagai argumen, yang berguna dalam operasi yang memerlukan perubahan langsung terhadap data di luar prosedur. Selain itu, Golang juga mendukung passing variadic parameters, yang memungkinkan prosedur menerima sejumlah parameter yang tidak tetap. Fitur ini sangat membantu dalam kasus di mana jumlah input yang diterima bervariasi, seperti dalam fungsi untuk menjumlahkan sekumpulan angka atau mencetak sejumlah string.

Penggunaan prosedur memiliki berbagai keuntungan, seperti meningkatkan efisiensi kode, mempermudah debugging, serta mempercepat pengembangan perangkat lunak karena memungkinkan kode yang sudah dibuat dapat digunakan kembali di berbagai bagian program. Dalam skala besar, prosedur membantu dalam pengorganisasian proyek dengan membagi tugas menjadi bagian-bagian yang lebih kecil dan terstruktur, sehingga pemeliharaan kode menjadi lebih mudah. Dengan memahami dan mengimplementasikan prosedur secara efektif, seorang programmer dapat meningkatkan kualitas kodenya serta mengembangkan aplikasi yang lebih optimal dan mudah dikelola dalam jangka panjang.

A. GUIDED

1. Source code :

103112400071_MODUL4 > 103112400071_guide1.go > hitungGaji

```go
//Raihan Adi Arba
//103112400071

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.Scanln(&nama)

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

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

    hitungGaji(nama, gajiPokok, jamLembur)
}
```

Output :

```
PS D:\raihan\github\Laprak-Modul-4\103112400071_MOdul4> go run 103112400071_guide1.go
Masukkan Nama Karyawan: Raihan
Masukkan Gaji Pokok: 1000000
Masukkan Jumlah Jam Lembur: 20

=== SLip Gaji ===
Nama Karyawan:  Raihan
Gaji Pokok : Rp1000000.00
Bonus Lembur : Rp1000000.00 (20 jam x Rp 50,000)
Total Gaji : Rp2000000.00
PS D:\raihan\github\Laprak-Modul-4\103112400071_MOdul4>
```

Deskripsi :

Program ini meminta pengguna untuk memasukkan nama karyawan, gaji pokok, dan jumlah jam lembur. Setelah menerima input dari pengguna, program akan menghitung total gaji karyawan dengan menambahkan bonus lembur berdasarkan jumlah jam lembur yang dimasukkan. Bonus lembur dihitung dengan mengalikan jumlah jam lembur dengan tarif sebesar Rp50.000 per jam. Program ini menggunakan fungsi fmt.Print dan fmt.Scanln untuk menerima input dari pengguna. Setelah semua data

diperoleh, program akan memanggil fungsi hitungGaji yang bertugas menghitung total gaji dan menampilkan slip gaji dalam format yang rapi. Output yang ditampilkan mencakup nama karyawan, gaji pokok, bonus lembur, serta total gaji yang telah dihitung. Program ini tidak memiliki validasi input, sehingga pengguna diharapkan memasukkan data dengan format yang sesuai agar hasil perhitungan dapat ditampilkan dengan benar

2. Source code :

```go
//Raihan Adi Arba
//103112400071

package main

import "fmt"

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
    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)
    hitungRataRata(nama, nilai1, nilai2, nilai3)
}
```

Output :

```
PS D:\raihan\github\Laprak-Modul-4\103112400071_MOdul4> go run 103112400071_guide2.go
Masukkan Nama Mahasiswa : Raihan
Masukkan Nilai 1 : 10
Masukkan Nilai 2 : 11
Masukkan Nilai 3 : 19

=== Hasil Akademik ===
Nama Mahasiswa : Raihan
Nilai 1         : 10.00
Nilai 2         : 11.00
Nilai 3         : 19.00
Rata-Rata       : 13.33
Status  :  Tidak Lulus
PS D:\raihan\github\Laprak-Modul-4\103112400071_MOdul4>
```

Deskripsi :

Program ini meminta pengguna untuk memasukkan nama mahasiswa serta tiga nilai ujian. Setelah semua data diinput, program akan menghitung rata-rata dari ketiga nilai tersebut. Jika rata-rata nilai mahasiswa lebih dari atau sama dengan 60, maka mahasiswa dinyatakan "Lulus", sedangkan jika kurang dari 60, mahasiswa dinyatakan "Tidak Lulus". Program menggunakan fungsi fmt.Print dan fmt.Scanln untuk menerima input dari pengguna. Setelah itu, fungsi hitungRataRata akan dipanggil untuk menghitung rata-rata dan menentukan status kelulusan mahasiswa berdasarkan nilai yang diperoleh. Hasil akhirnya akan ditampilkan dalam format yang rapi, mencakup nama mahasiswa, ketiga nilai, rata-rata nilai, dan status kelulusan.

3. Source code:

```go
//Nama  : Raihan Adi Arba
//NIM   : 103112400071
//KELAS : IF-12-01

package main

import "fmt"

func main() {
    nilai := 80
    pctHadir := 0.75
    adaTubes := true

    var indeks string

    if nilai > 75 && adaTubes {
        indeks = "A"
    } else if nilai > 65 {
        indeks = "B"
    } else if nilai > 50 && pctHadir > 0.7 {
        indeks = "C"
    } else {
        indeks = "F"
    }
    fmt.Printf("nilai %d dengan kehadiran %.2f%% dan buat tubes=%t, mendapat indeks %s\n", nilai,
    pctHadir*100, adaTubes, indeks)
}
```

Output :

```
raihan@Raihans-MacBook-Pro guide % go run 103112400071_guide3.go
nilai 80 dengan kehadiran 75.00% dan buat tubes=true, mendapat indeks A
```

Deskripsi :

Program ini menentukan indeks nilai mahasiswa berdasarkan nilai ujian, persentase kehadiran, dan keberadaan tugas besar (tubes).

Program menggunakan beberapa kondisi `if-else` untuk menilai apakah mahasiswa mendafmt.Printfeks A, B, C, atau F berdasarkan aturan berikut:

- Jika nilai lebih dari 75 dan mahasiswa memiliki tugas besar, maka mendapat indeks A.
- Jika nilai lebih dari 65, maka mendapat indeks B.
- Jika nilai lebih dari 50 dan persentase kehadiran lebih dari 70%, maka mendapat indeks C.
- Jika tidak memenuhi syarat di atas, maka mendapat indeks F.

Pada akhir program, hasil evaluasi ditampilkan dalam format yang jelas menggunakan `fmt.Printf`.

B. UNGUIDED

1. Latihan 1

   Source Code:

103112400071_MODUL4 > 103112400071_unguided1.go > processValues

```go
// 103112400071
package main

import "fmt"

func factorial(n int) int {
    result := 1
    for i := 1; i <= n; i++ {
        result *= i
    }
    return result
}
func permutation(n, r int) int {
    return factorial(n) / factorial(n-r)
}
func combination(n, r int) int {
    return factorial(n) / (factorial(r) * factorial(n-r))
}
func processValues(a, b, c, d int) {
    if a >= c && b >= d {
        // Menghitung permutasi dan kombinasi
        perm1 := permutation(a, c)
        comb1 := combination(a, c)
        perm2 := permutation(b, d)
        comb2 := combination(b, d)
        //hasil
        fmt.Println(perm1, comb1)
        fmt.Println(perm2, comb2)
    } else {
        fmt.Println("input tidak sesuai")
    }
}
func main() {
    var a, b, c, d int
    fmt.Scan(&a, &b, &c, &d)
    processValues(a, b, c, d)
}
```

Output :

```
PS D:\raihan\github\Laprak-Modul-4\103112400071_MOdul4> go run 103112400071_unguided1.go
5
10
3
10
60 10
3628800 1
PS D:\raihan\github\Laprak-Modul-4\103112400071_MOdul4>
```

Penjelasan Program:

Program ini digunakan untuk menghitung permutasi dan kombinasi berdasarkan empat angka yang dimasukkan oleh pengguna. Pengguna diminta untuk memasukkan nilai a, b, c, dan d. Program kemudian memeriksa apakah a lebih besar atau sama

dengan c, serta b lebih besar atau sama dengan d. Jika kondisi ini terpenuhi, program akan menghitung permutasi dan kombinasi untuk pasangan (a, c) serta (b, d), kemudian menampilkan hasilnya. Jika tidak, program akan menampilkan pesan "input tidak sesuai". Perhitungan dilakukan menggunakan tiga fungsi utama, yaitu factorial(n) untuk menghitung faktorial, permutation(n, r) untuk menghitung permutasi, dan combination(n, r) untuk menghitung kombinasi. Program ini berguna untuk membantu perhitungan permutasi dan kombinasi dengan mudah, namun belum memiliki validasi untuk input negatif.

2. Latihan 2

   Source Code:

103112400071_MODUL4 > 103112400071_unguided2.go > ...

```go
// Raihan Adi Arba
// 103112400071
package main

import "fmt"

func main() {
    var (
        winnerNama                                          string
        winnerSkor                                          = 0
        winnerSoal                                          = 0
        nama                                                string
        waktu1, waktu2, waktu3, waktu4, waktu5, waktu6, waktu7, waktu8 int
        soal, skor                                          int
    )

    for {
        fmt.Scan(&nama)
        if nama == "Selesai" {
            break
        }

        fmt.Scan(&waktu1, &waktu2, &waktu3, &waktu4, &waktu5, &waktu6, &waktu7, &waktu8)

        hitungSkor(waktu1, waktu2, waktu3, waktu4, waktu5, waktu6, waktu7, waktu8, &soal, &skor)

        if soal > winnerSoal || (soal == winnerSoal && skor < winnerSkor) {
            winnerNama = nama
            winnerSkor = skor
            winnerSoal = soal
        }
    }

    fmt.Println(winnerNama, winnerSoal, winnerSkor)
}
```

```go
func hitungSkor(w1, w2, w3, w4, w5, w6, w7, w8 int, soal, skor *int) {
    *soal = 0
    *skor = 0

    if w1 <= 300 {
        *soal += 1
        *skor += w1
    }
    if w2 <= 300 {
        *soal += 1
        *skor += w2
    }
    if w3 <= 300 {
        *soal += 1
        *skor += w3
    }
    if w4 <= 300 {
        *soal += 1
        *skor += w4
    }
    if w5 <= 300 {
        *soal += 1
        *skor += w5
    }
    if w6 <= 300 {
        *soal += 1
        *skor += w6
    }
    if w7 <= 300 {
        *soal += 1
        *skor += w7
    }
    if w8 <= 300 {
        *soal += 1
        *skor += w8
    }
}

```

Output:

```
PS D:\raihan\github\Laprak-Modul-4\103112400071_MOdul4> go run 103112400071_unguided2.go
Astuti 20 50 301 301 61 71 75 10
Bertha 25 47 301 26 50 60 65 21
Selesai
Bertha 7 294
PS D:\raihan\github\Laprak-Modul-4\103112400071_MOdul4>
```

Deskripsi Program:

Program ini digunakan untuk menentukan pemenang dari sekumpulan peserta berdasarkan jumlah soal yang berhasil diselesaikan dalam waktu tertentu. Setiap peserta akan memasukkan namanya dan waktu penyelesaian untuk delapan soal. Jika waktu pengerjaan sebuah soal tidak melebihi 300 detik, maka soal tersebut dianggap berhasil diselesaikan, dan waktu tersebut akan dijumlahkan sebagai skor total peserta. Setelah semua peserta memasukkan datanya, program akan membandingkan hasil mereka. Pemenang adalah peserta yang menyelesaikan soal terbanyak. Jika ada lebih

dari satu peserta dengan jumlah soal yang sama, maka peserta dengan skor (total waktu) lebih kecil akan menjadi pemenang. Program terus menerima input hingga peserta memasukkan kata "Selesai", yang menandakan akhir dari proses input. Fungsi hitungSkor digunakan untuk menghitung jumlah soal yang berhasil diselesaikan dan total skor berdasarakan batas waktu yang ditentukan.

3. Latihan 3

Source Code:

103112400071_MODUL4 > 103112400071_unguided3.go > CetakDeret

```go
//RAIHAN ADI ARBA
//103112400071

package main

import "fmt"

func CetakDeret(n int, hasil *float64) {
    fmt.Print(n, " ")

    var ntemp int = n
    for ntemp != 1 {
        if ntemp%2 == 0 {
            *hasil = float64(ntemp) / 2
            ntemp = int(*hasil)
        } else {
            *hasil = 3*float64(ntemp) + 1
            ntemp = int(*hasil)
        }
        fmt.Print(ntemp, " ")
    }
}

func main() {
    var n int
    var deret float64
    fmt.Scan(&n)
    if n < 1000000 {
        CetakDeret(n, &deret)
    } else {
        fmt.Print("Eror")
    }
}

```

Output:

```
PS D:\raihan\github\Laprak-Modul-4\103112400071_MOdul4> go run 103112400071_unguided3.go
22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS D:\raihan\github\Laprak-Modul-4\103112400071_MOdul4>
```

Deskripsi Program:
Program ini mencetak deret angka berdasarkan aturan Collatz, di mana setiap angka dalam deret dihasilkan dari angka sebelumnya menggunakan aturan tertentu. Jika angka tersebut genap, maka akan dibagi dua; jika ganjil, maka akan dikalikan tiga dan ditambah satu. Proses ini berlanjut hingga angka mencapai nilai 1. Program meminta pengguna untuk memasukkan sebuah bilangan n. Jika nilai n kurang dari 1.000.000, program akan menghitung dan mencetak deret berdasarkan aturan tersebut menggunakan fungsi cetakDeret. Jika n lebih besar atau sama dengan 1.000.000, program akan menampilkan pesan "Eror" sebagai batasan input. Fungsi cetakDeret menerima input n dan sebuah variabel pointer hasil yang menyimpan nilai sementara hasil perhitungan. Perulangan dalam fungsi ini terus berjalan hingga angka mencapai 1, mencetak setiap angka dalam deret selama proses berlangsung.
.

DAFTAR PUSTAKA

Prayogo, N. A. (2021). ***Dasar Pemrograman Go. Ebook***