LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 4

PROSEDUR

Oleh:

DWI OKTA SURYANINGRUM

103112400066

12-IF-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

Pada bahasa pemrograman Go, konsep prosedur dan fungsi diimplementasikan melalui **fungsi** (function). Go tidak menggunakan istilah "prosedur" secara eksplisit, namun secara konsep, prosedur dapat dianggap sebagai sebuah fungsi yang tidak mengembalikan nilai (void function). Fungsi di Go didefinisikan dengan kata kunci func, diikuti dengan nama fungsi, parameter yang diterima oleh fungsi, tipe parameter tersebut, dan tipe data yang dikembalikan oleh fungsi (jika ada). Fungsi yang tidak mengembalikan nilai bisa dianggap sebagai prosedur dalam konteks pemrograman. Sebuah fungsi dapat menerima beberapa parameter, dan Go mendukung pengembalian lebih dari satu nilai, yang menjadi fitur khas bahasa ini. Fungsi juga bisa menggunakan parameter variadic, yang memungkinkan kita untuk mengirimkan sejumlah parameter yang tidak tetap, yaitu sejumlah argumen yang bervariasi.

Secara garis besar, fungsi di Go memiliki dua jenis:

1. **Fungsi yang mengembalikan nilai**: Fungsi ini mengembalikan satu atau lebih nilai, dan nilai-nilai ini dapat digunakan lebih lanjut di dalam program. Pengembalian nilai dilakukan menggunakan pernyataan return.
2. **Fungsi yang tidak mengembalikan nilai**: Fungsi ini tidak mengembalikan apa pun, tetapi bisa melakukan beberapa tugas atau aksi seperti mencetak output atau memodifikasi variabel lain. Fungsi jenis ini pada dasarnya adalah prosedur dalam paradigma pemrograman lain.

Fungsi di Go membuat program menjadi lebih modular, mudah dibaca, dan lebih terstruktur. Program dapat dipecah menjadi bagian-bagian kecil yang dapat dipanggil secara terpisah, memudahkan pemeliharaan kode dan pengembangan lanjutan. Fungsi dapat dipanggil berulang kali di berbagai bagian program tanpa harus menulis ulang logika yang sama.

II. GUIDED

Source Code + Screenshot hasil program beserta penjelasan

a. Guided 1

```go
// DWI OKTA SURYANINGRUM
// package main

import "fmt"

// Fungsi hitungGaji menghitung total gaji karyawan berdasarkan gaji pokok dan jam lembur.
// Parameter:
// - nama: Nama karyawan.
// - gajipokok: Gaji pokok karyawan.
// - jamLembur: Jumlah jam lembur yang dilakukan karyawan.
func hitungGaji(nama string, gajipokok float64, jamLembur int) {
    // Hitung bonus lembur: Rp50.000 per jam
    bonusLembur := float64(jamLembur) * 50000

    // Hitung total gaji: gaji pokok + bonus lembur
    totalGaji := gajipokok + bonusLembur

    // Cetak slip gaji
    fmt.Println("\n=== Slip Gaji ===")
    fmt.Println("")
    fmt.Println("Nama Karyawan :", nama)
    fmt.Printf("Gaji Pokok : Rp %2.f\n", gajipokok)
    fmt.Printf("Bonus Lembur : Rp %2.f(%d jam x Rp50.000)\n", bonusLembur, jamLembur)
    fmt.Printf("Total Gaji : Rp %2.f\n", totalGaji)
}

func main() {
    // Variabel untuk menyimpan input dari pengguna
    var nama string
    var gajipokok float64
    var jamLembur int

    // Minta pengguna memasukkan nama karyawan
    fmt.Print("Masukkan Nama Karyawan : ")
    fmt.Scan(&nama)

    // Minta pengguna memasukkan gaji pokok
    fmt.Print("Masukkan Gaji Pokok : ")
    fmt.Scan(&gajipokok)

    // Minta pengguna memasukkan jumlah jam lembur
    fmt.Print("Masukkan Jumlah Jam Lembur : ")
    fmt.Scan(&jamLembur)

    // Panggil fungsi hitungGaji dengan data yang dimasukkan pengguna
    hitungGaji(nama, gajipokok, jamLembur)
}
```

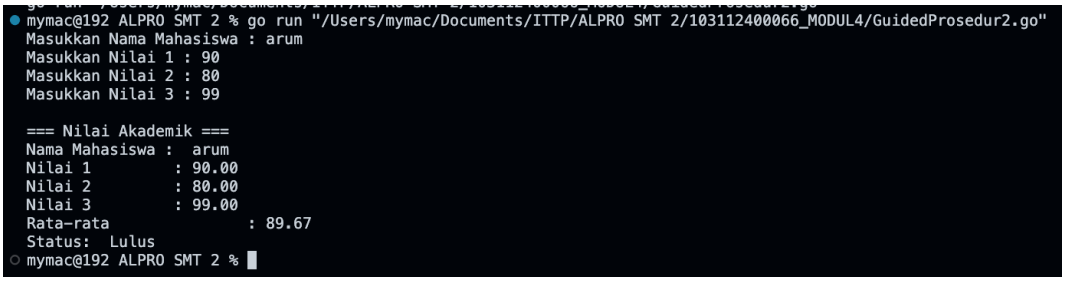
Output :

```
mymac@192 ALPRO SMT 2 % go run "/Users/mymac/Documents/ITTP/ALPRO SMT 2/103112400066_MODUL4/GuidedProsedur1.go"
Masukkan Nama Karyawan : Andi
Masukkan Gaji Pokok : 12000000
Masukkan Jumlah Jam Lembur : 2

=== Slip Gaji ===

Nama Karyawan : Andi
Gaji Pokok : Rp 12000000
Bonus Lembur : Rp 100000(2 jam x Rp50.000)
Total Gaji : Rp 12100000
```

b. Guided 2

```go
//DWI OKTA SURYANINGRUM
package main

import "fmt"

// Fungsi hitungRataRata menghitung rata-rata nilai mahasiswa dan menentukan status kelulusan.
// Parameter:
// - nama: Nama mahasiswa.
// - nilai1, nilai2, nilai3: Nilai-nilai mahasiswa.
func hitungRataRata(nama string, nilai1, nilai2, nilai3 float64) {
    // Hitung rata-rata nilai
    rataRata := (nilai1 + nilai2 + nilai3) / 3

    // Tentukan status kelulusan
    status := "Tidak Lulus" // Default status adalah "Tidak Lulus"
    if rataRata >= 60 {     // Jika rata-rata >= 60, status diubah menjadi "Lulus"
        status = "Lulus"
    }

    // Cetak hasil nilai akademik
    fmt.Println("\n=== Nilai 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() {
    // Variabel untuk menyimpan input dari pengguna
    var nama string
    var nilai1, nilai2, nilai3 float64

    // Minta pengguna memasukkan nama mahasiswa
    fmt.Print("Masukkan Nama Mahasiswa : ")
    fmt.Scanln(&nama)

    // Minta pengguna memasukkan nilai 1
    fmt.Print("Masukkan Nilai 1 : ")
    fmt.Scanln(&nilai1)

    // Minta pengguna memasukkan nilai 2
    fmt.Print("Masukkan Nilai 2 : ")
    fmt.Scanln(&nilai2)

    // Minta pengguna memasukkan nilai 3
    fmt.Print("Masukkan Nilai 3 : ")
    fmt.Scanln(&nilai3)

    // Panggil fungsi hitungRataRata dengan data yang dimasukkan pengguna
    hitungRataRata(nama, nilai1, nilai2, nilai3)
}
```

Output :


mymac@192 ALPRO SMT 2 % go run "/Users/mymac/Documents/ITTP/ALPRO SMT 2/103112400066_MODUL4/GuidedProsedur2.go"
Masukkan Nama Mahasiswa : arum
Masukkan Nilai 1 : 90
Masukkan Nilai 2 : 80
Masukkan Nilai 3 : 99
=== Nilai Akademik ===
Nama Mahasiswa : arum
Nilai 1 : 90.00
Nilai 2 : 80.00
Nilai 3 : 99.00
Rata-rata : 89.67
Status: Lulus
mymac@192 ALPRO SMT 2 %

III. UNGUIDED

a. Unguided 1

```go
//DWI OKTA SURYANINGRUM
package main

import "fmt"

// Fungsi untuk menghitung faktorial dari sebuah angka.
// Faktorial adalah hasil perkalian semua angka dari 1 sampai angka tersebut.
func factorial(n int, hasil *int) {
    *hasil = 1 // Mulai dengan hasil = 1
    for i := 1; i <= n; i++ {
        *hasil *= i // Kalikan hasil dengan angka i (1, 2, 3, ..., n)
    }
}

// Fungsi untuk menghitung permutasi.
// Permutasi adalah jumlah cara mengatur r elemen dari n elemen yang tersedia, dengan memperhatikan urutan.
// Rumus permutasi: P(n, r) = n! / (n-r)!
func permutation(n, r int, hasil *int) {
    var fn, fnr int // Variabel untuk menyimpan faktorial n dan faktorial (n-r)
    factorial(n, &fn) // Hitung faktorial n
    factorial(n-r, &fnr) // Hitung faktorial (n-r)
    *hasil = fn / fnr // Hitung permutasi menggunakan rumus
}

// Fungsi untuk menghitung kombinasi.
// Kombinasi adalah jumlah cara memilih r elemen dari n elemen yang tersedia, tanpa memperhatikan urutan.
// Rumus kombinasi: C(n, r) = n! / (r! * (n-r)!)
func combination(n, r int, hasil *int) {
    var fn, fr, fnr int // Variabel untuk menyimpan faktorial n, r, dan (n-r)
    factorial(n, &fn) // Hitung faktorial n
    factorial(r, &fr) // Hitung faktorial r
    factorial(n-r, &fnr) // Hitung faktorial (n-r)
    *hasil = fn / (fr * fnr) // Hitung kombinasi menggunakan rumus
}

func main() {
    // Variabel untuk menyimpan input dari pengguna
    var a, b, c, d int

    // Variabel untuk menyimpan hasil permutasi dan kombinasi
    var perm1, comb1, perm2, comb2 int

    // Minta pengguna memasukkan 4 angka
    fmt.Scan(&a, &b, &c, &d)

    // Cek apakah input valid (a >= c dan b >= d)
    // Ini penting karena permutasi dan kombinasi hanya bisa dihitung jika n >= r
    if a >= c && b >= d {
        // Hitung permutasi dan kombinasi untuk a dan c
        permutation(a, c, &perm1) // Hitung P(a, c)
        combination(a, c, &comb1) // Hitung C(a, c)

        // Hitung permutasi dan kombinasi untuk b dan d
        permutation(b, d, &perm2) // Hitung P(b, d)
        combination(b, d, &comb2) // Hitung C(b, d)

        // Tampilkan hasil permutasi dan kombinasi untuk a dan c
        fmt.Println(perm1, comb1)

        // Tampilkan hasil permutasi dan kombinasi untuk b dan d
        fmt.Println(perm2, comb2)
    } else {
        // Jika input tidak valid, tampilkan pesan kesalahan
        fmt.Println("input tidak sesuai")
    }
}
```

Output :

```
mymac@192 ALPRO SMT 2 % go run "/Users/mymac/Documents/ITTP/ALPRO SMT 2/103112400066_MODUL4/Unguided1.go"
5 10 3 10
60 10
3628800 1
mymac@192 ALPRO SMT 2 % go run "/Users/mymac/Documents/ITTP/ALPRO SMT 2/103112400066_MODUL4/Unguided1.go"
8 0 2 0
56 28
1 1
mymac@192 ALPRO SMT 2 %
```

b. Unguided 2

```go
// DWI OKTA SURYANINGRUM
package main

import (
    "fmt"
)

// Fungsi hitungSkor digunakan untuk menghitung berapa banyak soal yang berhasil diselesaikan dan total waktu yang dibutuhkan oleh seorang peserta.
// Parameter:
// - waktuPengerjaan: Daftar waktu yang dibutuhkan untuk menyelesaikan setiap soal.
// - totalSoal: Variabel untuk menyimpan jumlah soal yang berhasil diselesaikan.
// - totalSkor: Variabel untuk menyimpan total waktu yang dibutuhkan.
func hitungSkor(waktuPengerjaan []int, totalSoal *int, totalSkor *int) {
    // Mulai dengan mengatur totalSoal dan totalSkor ke 0
    *totalSoal = 0
    *totalSkor = 0

    // Loop melalui setiap waktu pengerjaan
    for _, waktu := range waktuPengerjaan {
        // Jika waktu pengerjaan kurang dari atau sama dengan 300 menit (5 jam), artinya soal tersebut berhasil diselesaikan.
        if waktu <= 300 {
            *totalSoal++ // Tambahkan 1 ke totalSoal
            *totalSkor += waktu // Tambahkan waktu ke totalSkor
        }
    }
}

func main() {
    // Variabel untuk menyimpan nama pemenang, jumlah soal terbanyak, dan total waktu terkecil
    var namaPemenang string
    var maxSoal, minSkor int

    // Loop utama untuk meminta input dari user
    for {
        var nama string
        // Meminta user memasukkan nama peserta
        fmt.Print("Masukkan nama peserta (atau ketik 'Selesai' untuk mengakhiri): ")
        fmt.Scan(&nama)

        // Jika user mengetik "Selesai", maka loop akan dihentikan
        if nama == "Selesai" {
            break
        }

        // Array untuk menyimpan waktu pengerjaan 8 soal
        var waktuPengerjaan [8]int
        // Meminta user untuk memasukkan waktu pengerjaan untuk 8 soal
        fmt.Print("Masukkan waktu pengerjaan untuk 8 soal (dalam menit): ")
        for i := 0; i < 8; i++ {
            fmt.Scan(&waktuPengerjaan[i]) // Membaca waktu pengerjaan untuk setiap soal
        }

        // Variabel untuk menyimpan total soal dan total skor peserta
        var totalSoal, totalSkor int
        // Panggil fungsi hitungSkor untuk menghitung total soal dan total skor
        hitungSkor(waktuPengerjaan[:], &totalSoal, &totalSkor)

        // Bandingkan skor peserta dengan skor tertinggi saat ini
        // Jika peserta ini menyelesaikan lebih banyak soal, atau
        // jika jumlah soalnya sama tetapi waktu pengerjaannya lebih cepat,
        // maka peserta ini menjadi pemenang sementara.
        if totalSoal > maxSoal || (totalSoal == maxSoal && totalSkor < minSkor) {
            namaPemenang = nama // Simpan nama pemenang
            maxSoal = totalSoal // Simpan jumlah soal terbanyak
            minSkor = totalSkor // Simpan total waktu terkecil
        }
    }

    // Setelah loop selesai, tampilkan hasil pemenang
    if namaPemenang != "" {
        fmt.Printf("Pemenang: %s, Jumlah soal: %d, Nilai: %d\n", namaPemenang, maxSoal, minSkor)
    } else {
        // Jika tidak ada peserta yang masuk, tampilkan pesan ini
        fmt.Println("Tidak ada peserta yang masuk.")
    }
}
```

Output :

```
mymac@192 ALPRO SMT 2 % go run "/Users/mymac/Documents/ITTP/ALPRO SMT 2/103112400066_MODUL4/Unguided2.go"
Masukkan nama peserta (atau ketik 'Selesai' untuk mengakhiri): astuti
Masukkan waktu pengerjaan untuk 8 soal (dalam menit): 20 50 301 301 61 71 75 10
Masukkan nama peserta (atau ketik 'Selesai' untuk mengakhiri): Bertha
Masukkan waktu pengerjaan untuk 8 soal (dalam menit): 25 47 301 26 50 60 65 21
Masukkan nama peserta (atau ketik 'Selesai' untuk mengakhiri): Selesai
Pemenang: Bertha, Jumlah soal: 7, Nilai: 294
mymac@192 ALPRO SMT 2 %
```

c. Unguided 3

```go
// DWI OKTA SURYANINGRUM
package main

import "fmt"

// Prosedur cetakDeret mencetak deret bilangan sesuai aturan Skiena dan Revilla.
// Parameter:
// - n: Bilangan awal untuk memulai deret.
func cetakDeret(n int) {
    // Loop akan terus berjalan selama n tidak sama dengan 1
    for n != 1 {
        fmt.Print(n, " ") // Cetak nilai n saat ini diikuti spasi
        if n%2 == 0 {     // Jika n genap
            n = n / 2 // Hitung suku berikutnya: n = n / 2
        } else { // Jika n ganjil
            n = 3*n + 1 // Hitung suku berikutnya: n = 3n + 1
        }
    }
    fmt.Print(n) // Cetak nilai terakhir (1)
}

func main() {
    var n int
    // Minta pengguna memasukkan bilangan awal
    fmt.Scan(&n)

    // Panggil prosedur cetakDeret untuk mencetak deret
    cetakDeret(n)
}
```

Output :

```
mymac@192 ALPRO SMT 2 % go run "/Users/mymac/Documents/ITTP/ALPRO SMT 2/103112400066_MODUL4/Unguided3.go"
22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
mymac@192 ALPRO SMT 2 %
```

IV. KESIMPULAN

Di dalam bahasa pemrograman Go, tidak ada istilah "procedure" seperti dalam beberapa bahasa lain, tetapi konsep tersebut diwakili oleh **fungsi (function)**. Sebuah fungsi di Go dapat berfungsi seperti prosedur jika tidak mengembalikan nilai. Fungsi di Go didefinisikan dengan kata kunci func, diikuti dengan nama fungsi, daftar parameter yang diterima, tipe parameter tersebut, dan tipe data yang dikembalikan (jika ada). Fungsi dapat mengembalikan satu atau lebih nilai, dan jika tidak mengembalikan nilai, fungsi tersebut bertindak seperti prosedur dalam paradigma pemrograman lain. Go juga mendukung fitur **parameter variadic**, yang memungkinkan sebuah fungsi untuk menerima jumlah parameter yang tidak tetap.

Fungsi di Go memungkinkan pemrogram untuk menulis kode yang modular dan terstruktur dengan lebih baik. Dengan menggunakan fungsi, program dapat dipisah menjadi bagian-bagian yang lebih kecil dan dapat dipanggil kembali sesuai kebutuhan tanpa menulis ulang kode yang sama. Fungsi yang mengembalikan nilai dapat digunakan untuk perhitungan dan operasi lainnya, sementara fungsi tanpa nilai kembalian dapat digunakan untuk mengeksekusi aksi tertentu (seperti mencetak output atau memodifikasi data).

V. REFERENSI

“Effective Go.” 2025. *Go*. Accessed March 19. https://go.dev/doc/effective_go#