LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL IV

PROSEDUR

Disusun Oleh :

ARIIQ RADHITYA PRADANA

2311102260

IF 11 06

Dosen Pengampu :

Abednego Dwi Septiadi

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO 2024

I. DASAR TEORI

Prosedur dalam pemrograman adalah konsep penting yang berfungsi untuk mengorganisir kode menjadi bagian-bagian yang lebih kecil dan terkelola. Sebuah prosedur terdiri dari sekumpulan instruksi yang dikelompokkan dengan tujuan tertentu, memungkinkan pengembang untuk memecah program kompleks menjadi bagian yang lebih sederhana. Struktur dasar dari sebuah prosedur mencakup identifier (nama prosedur), parameter (variabel untuk bertukar informasi), dan block (bagian utama yang berisi instruksi). Parameter dalam prosedur dapat berupa parameter formal, yang didefinisikan saat deklarasi, dan parameter aktif, yang dikirimkan saat pemanggilan prosedur. Pengiriman parameter bisa dilakukan dengan cara by value, di mana nilai asli tidak terpengaruh, atau by reference, di mana perubahan pada parameter formal mempengaruhi nilai asli. Manfaat penggunaan prosedur meliputi modularitas, reusabilitas, dan kemudahan pemeliharaan kode. Misalnya, dalam program untuk menghitung luas lingkaran, kita bisa mendefinisikan prosedur yang menerima jari-jari sebagai parameter dan menghitung luas berdasarkan rumus matematika. Dengan demikian, prosedur merupakan elemen fundamental dalam pemrograman yang membantu menciptakan kode yang lebih terstruktur dan mudah dipahami, terutama dalam pengembangan perangkat lunak berskala besar.

II. GUIDED

Guided 1

```
package main

import (
	"bufio"
	"fmt"
	"os"
)

func main() {
	var bilangan int
	var pesan string

	// Create a scanner to read input
	scanner := bufio.NewScanner(os.Stdin)

	// Read the integer
	fmt.Print("Masukkan bilangan (0, 1, 2): ")
	scanner.Scan() // Read the integer input
	fmt.Sscanf(scanner.Text(), "%d", &bilangan)

	// Read the string
	fmt.Print("Masukkan pesan: ")
	scanner.Scan() // Read the string input
	pesan = scanner.Text()

	// Call cetakPesan function
	cetakPesan(pesan, bilangan)
}

func cetakPesan(M string, flag int) {
	var jenis string
	if flag == 0 {
		jenis = "error"
	} else if flag == 1 {
		jenis = "warning"
	} else if flag == 2 {
		jenis = "informasi"
	} else {
		jenis = "unknown" // Handle invalid flag
values
	}
	// Print message and type
	fmt.Println(M, jenis)
}
```

Screenshoot Output

Deskripsi Program

(Program ini adalah aplikasi sederhana yang ditulis dalam bahasa pemrograman Go, bertujuan untuk mengkategorikan pesan berdasarkan nilai integer yang dimasukkan pengguna. Saat dijalankan, program akan meminta pengguna untuk memasukkan sebuah bilangan bulat (0, 1, atau 2) dan sebuah pesan dalam bentuk string. Berdasarkan nilai bilangan yang dimasukkan, program akan menentukan jenis pesan: "error" untuk nilai 0, "warning" untuk nilai 1, dan "informasi" untuk nilai 2. Jika pengguna memasukkan nilai di luar rentang tersebut, program akan mengembalikan kategori "unknown". Setelah menerima input, program akan mencetak pesan yang diinputkan bersamaan dengan jenisnya ke layar, memberikan umpan balik yang jelas kepada pengguna mengenai kategori pesan yang mereka masukkan.

Guided 2

```
package main

import "fmt"

func sendEmailNotification(email string) {
    fmt.Printf("Mengirim email ke %s: Pendaftaran
berhasil.\n", email)
}

func main() {
    emails := []string{"user@example.com",
"user2@example.com", "user3@example.com"}

    fmt.Println("Mengirim email ke pengguna yang baru
terdaftar: ")
    for _, email := range emails {
        sendEmailNotification(email)
    }
}
```

Screenshoot Output

```
go run /tmp/VfGx3z1rAL.go
Mengirim email ke pengguna yang baru terdaftar:
Mengirim email ke user@example.com: Pendaftaran berhasil.
Mengirim email ke user2@example.com: Pendaftaran berhasil.
Mengirim email ke user3@example.com: Pendaftaran berhasil.
```

Deskripsi Program

Program ini adalah contoh aplikasi sederhana yang ditulis dalam bahasa pemrograman Go, dirancang untuk mengirim notifikasi email kepada pengguna yang baru terdaftar. Dalam fungsi main, terdapat sebuah slice bernama emails yang berisi daftar alamat email pengguna yang telah mendaftar. Program ini mengawali dengan mencetak pesan yang memberi tahu bahwa email sedang dikirim kepada pengguna baru. Selanjutnya, menggunakan loop for, program mengiterasi setiap alamat email dalam slice tersebut dan memanggil fungsi sendEmailNotification, yang bertanggung jawab untuk menghasilkan output yang menunjukkan pengiriman email. Fungsi ini menerima parameter string berupa alamat email dan mencetak pesan yang menegaskan bahwa pendaftaran telah berhasil. Pendekatan ini tidak hanya mendemonstrasikan penggunaan fungsi dan iterasi dalam Go, tetapi juga mencerminkan cara efisien untuk menangani notifikasi dalam konteks aplikasi berbasis pengguna.umpan balik yang jelas kepada pengguna mengenai kategori pesan yang mereka masukkan.

Guided 3

```
package main

import "fmt"

func f1(x, y int) float64 {
     var hasil float64
     hasil = float64(2*x) - 0.5*float64(y) + 3.0
     return hasil
}

func f2(x, y int, hasil *float64) {
     *hasil = float64(2*x) - 0.5*float64(y) + 3.0
}

func main() {
     var a, b int
     var c float64

     fmt.Printf("Enter two integers: ")
     fmt.Scan(&a, &b)

     // Pass the address of c to the f2 function
     f2(a, b, &c)

     fmt.Println("Result from f2 (stored in c):", c)

     resultF1 := f1(b, a)
     fmt.Println("Result from f1:", resultF1)
}
```

Screenshoot Output

```
go run /tmp/U4iH8wMRgj.go
Enter two integers: 1 2
Result from f2 (stored in c): 4
Result from f1: 6.5
```

Deskripsi Program

Program ini adalah aplikasi sederhana yang ditulis dalam bahasa pemrograman Go, yang dirancang untuk melakukan perhitungan matematis berdasarkan dua bilangan bulat yang dimasukkan oleh pengguna. Program ini terdiri dari dua fungsi utama, yaitu `f1` dan `f2`. Fungsi `f1` menerima dua parameter bilangan bulat dan menghitung hasil dengan rumus tertentu, mengembalikannya dalam bentuk nilai float64. Sementara itu, fungsi `f2` juga melakukan perhitungan yang sama tetapi menggunakan pointer untuk mengubah nilai variabel yang diteruskan, sehingga hasil perhitungan dapat disimpan langsung ke dalam variabel yang ada di fungsi `main`. Dalam fungsi `main`, pengguna diminta untuk memasukkan dua bilangan bulat, yang kemudian digunakan sebagai argumen untuk kedua fungsi tersebut. Hasil dari fungsi `f2` disimpan dalam variabel `c`, yang ditampilkan di konsol, sedangkan hasil dari fungsi `f1` ditampilkan setelah memanggilnya dengan urutan parameter yang berbeda. Dengan demikian, program ini menunjukkan cara menggunakan fungsi dan pointer dalam Go untuk melakukan perhitungan dan mengelola hasil dengan efektif.

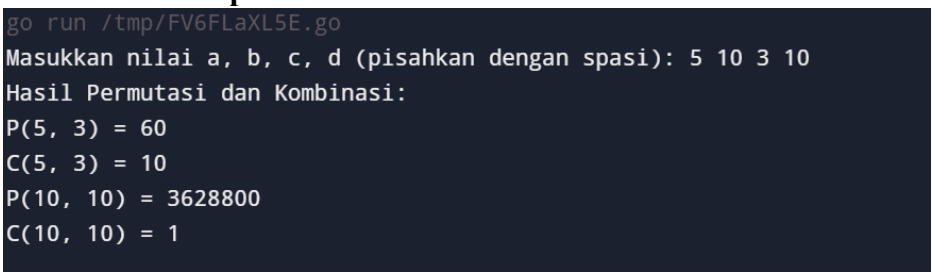
III. UNGUIDED

Unguided 1

```
package main

import (
    "fmt"
)

// Procedure untuk menghitung faktorial dari n
func mencariFaktorial(n int) int {
    if n == 0 {
        return 1
    }
    faktorial := 1
    for i := 1; i <= n; i++ {
        faktorial *= i
    }
    return faktorial
}

// Function untuk menghitung permutasi P(n, r)
func permutation(n int, r int) int {
    return mencariFaktorial(n) / mencariFaktorial(n-r)
}

// Function untuk menghitung kombinasi C(n, r)
func combination(n int, r int) int {
    return mencariFaktorial(n) / (mencariFaktorial(r) *
mencariFaktorial(n-r))
}

func main() {
    var a, b, c, d int
    fmt.Print("Masukkan nilai a, b, c, d (pisahkan
dengan spasi): ")
    fmt.Scan(&a, &b, &c, &d)

    // Validasi input
    if a < c || b < d {
        fmt.Println("Input tidak valid: a harus >= c dan
b harus >= d")
        return
    }

    // Menghitung permutasi dan kombinasi
    permA := permutation(a, c)
    combA := combination(a, c)
    permB := permutation(b, d)
    combB := combination(b, d)
```

```
    // Menampilkan hasil
    fmt.Printf("Hasil Permutasi dan Kombinasi:\n")
    fmt.Printf("P(%d, %d) = %d\n", a, c, permA)
    fmt.Printf("C(%d, %d) = %d\n", a, c, combA)
    fmt.Printf("P(%d, %d) = %d\n", b, d, permB)
    fmt.Printf("C(%d, %d) = %d\n", b, d, combB)
}
```

Screenshoot Output



Deskripsi Program

Program di atas ditulis dalam bahasa pemrograman Go (Golang) dan berfungsi untuk menghitung permutasi serta kombinasi dari dua pasangan angka yang dimasukkan oleh pengguna. Program ini dimulai dengan meminta input berupa empat nilai integer: `a`, `b`, `c`, dan `d`. Nilai-nilai ini digunakan untuk menghitung permutasi dan kombinasi.

Program pertama-tama memeriksa validitas input. Syaratnya adalah bahwa `a` harus lebih besar atau sama dengan `c`, dan `b` harus lebih besar atau sama dengan `d`. Jika syarat ini tidak terpenuhi, program akan menampilkan pesan kesalahan dan berhenti.

Setelah validasi, program menghitung permutasi dan kombinasi untuk dua pasangan angka: `P(a, c)` dan `P(b, d)` untuk permutasi, serta `C(a, c)` dan `C(b, d)` untuk kombinasi. Untuk melakukan perhitungan ini, program menggunakan dua fungsi utama: `permutation` untuk menghitung permutasi dan `combination` untuk menghitung kombinasi. Keduanya bergantung pada fungsi lain bernama `mencariFaktorial` yang menghitung faktorial dari suatu angka.

Hasil perhitungan permutasi dan kombinasi untuk kedua pasangan angka tersebut kemudian ditampilkan kepada pengguna dengan format yang jelas.

Unguide 2

```
package main

import (
    "bufio"
    "fmt"
    "math"
    "os"
    "strings"
)

// Prosedur hitungSkor menghitung jumlah soal yang
diselesaikan dan total waktu yang dibutuhkan
func hitungSkor(waktu []int, soal *int, skor *int) {
    *soal = 0 // jumlah soal yang diselesaikan
    *skor = 0 // total waktu pengerjaan soal yang
berhasil diselesaikan
    for _, w := range waktu {
        if w <= 300 {
            *soal++    // jika soal berhasil dikerjakan
(<= 300 menit)
            *skor += w // tambahkan waktu pengerjaan ke
skor
        }
    }
}

func main() {
    var pemenang string
    var maxSoal, minSkor int
    maxSoal = -1
    minSkor = math.MaxInt32

    reader := bufio.NewReader(os.Stdin) // Menggunakan
bufio untuk pembacaan input yang lebih fleksibel

    for {
        fmt.Println("Masukkan nama dan waktu pengerjaan
(ketik 'Selesai' untuk berhenti):")
        input, _ := reader.ReadString('\n') // Membaca
input baris penuh
        input = strings.TrimSpace(input)    // Menghapus
spasi atau karakter newline di akhir input

        // Periksa apakah inputnya "Selesai"
        if strings.ToLower(input) == "selesai" {
            break
        }

        // Cek apakah input kosong
        if input == "" {
```

```
            continue // lewati iterasi jika input kosong
        }

        // Pisahkan input menjadi bagian nama dan waktu
pengerjaan
        parts := strings.Fields(input)
        if len(parts) != 9 {
            fmt.Println("Input tidak valid, masukkan
nama dan 8 waktu pengerjaan soal.")
            continue
        }

        nama := parts[0] // Nama peserta
        var waktu [8]int

        // Parsing waktu pengerjaan
        for i := 0; i < 8; i++ {
            fmt.Sscanf(parts[i+1], "%d", &waktu[i])
        }

        // Hitung skor dan jumlah soal yang diselesaikan
        var soal, skor int
        hitungSkor(waktu[:], &soal, &skor)

        // Tentukan pemenang berdasarkan kriteria yang
disebutkan
        if soal > maxSoal || (soal == maxSoal && skor <
minSkor) {
            pemenang = nama
            maxSoal = soal
            minSkor = skor
        }
    }

    // Output pemenang
    if pemenang != "" {
        fmt.Printf("%s %d %d\n", pemenang, maxSoal,
minSkor)
    } else {
        fmt.Println("Tidak ada peserta.")
    }
}
```

Screenshoot Output

```
go run /tmp/jEJ1kHFS23.go
Masukkan nama dan waktu pengerjaan (ketik 'Selesai' untuk berhenti):
Astuti 20 50 301 301 61 71 75 10
Masukkan nama dan waktu pengerjaan (ketik 'Selesai' untuk berhenti):
Bertha 25 47 301 26 50 60 65 21
Masukkan nama dan waktu pengerjaan (ketik 'Selesai' untuk berhenti):
Selesai
Bertha 7 294
```

Deskripsi Program

Program di atas merupakan sebuah aplikasi berbasis CLI (Command Line Interface) yang berfungsi untuk menentukan pemenang dari sebuah kompetisi berdasarkan jumlah soal yang diselesaikan dan total waktu pengerjaan. Setiap peserta akan memasukkan nama dan waktu pengerjaan untuk 8 soal. Jika peserta menyelesaikan sebuah soal dalam waktu maksimal 300 menit, soal tersebut dianggap selesai dan waktu pengerjaannya akan dihitung sebagai skor.

Prosedur `hitungSkor` digunakan untuk menghitung jumlah soal yang diselesaikan dan mengakumulasi total waktu pengerjaan dari soal yang diselesaikan tersebut. Di dalam fungsi utama (`main`), input peserta diambil menggunakan `bufio.Reader` yang kemudian diproses dalam sebuah loop hingga pengguna memasukkan kata "Selesai" untuk mengakhiri input. Setiap peserta yang diinputkan akan dievaluasi berdasarkan dua kriteria: jumlah soal yang diselesaikan dan waktu pengerjaan. Pemenang adalah peserta yang menyelesaikan soal terbanyak, atau jika terdapat lebih dari satu peserta dengan jumlah soal yang sama, peserta dengan waktu pengerjaan total terkecil yang menjadi pemenang.

Program ini akan mengeluarkan nama pemenang beserta jumlah soal yang berhasil diselesaikan dan total waktu pengerjaan soal. Jika tidak ada peserta, program akan menampilkan pesan bahwa tidak ada peserta.

Unguide 3

```
package main

import "fmt"

// Prosedur untuk mencetak deret berdasarkan aturan yang
diberikan
func cetakDeret(n int) {
	// Cetak nilai awal
	fmt.Print(n)

	// Lanjutkan sampai n menjadi 1
	for n != 1 {
		if n%2 == 0 {
			n = n / 2
		} else {
			n = 3*n + 1
		}
		// Cetak setiap suku deret, dipisahkan dengan
spasi
		fmt.Print(" ", n)
	}
	fmt.Println()
}

func main() {
	var n int
	// Input bilangan bulat positif yang lebih kecil
dari 1000000
	fmt.Scan(&n)

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

Screenshoot Output

```
go run /tmp/Q11248aZWC.go
22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

Deskripsi Program

Program Go tersebut menerima input bilangan bulat positif `n` dari pengguna, yang harus lebih kecil dari 1.000.000, dan menghasilkan deret berdasarkan aturan Collatz (juga dikenal sebagai "3n + 1 conjecture"). Setelah menerima input, program memanggil prosedur `cetakDeret` yang mencetak nilai awal `n` dan melanjutkan mencetak suku-suku deret berikutnya. Jika `n` adalah bilangan genap, maka nilainya dibagi dua, sedangkan jika `n` adalah bilangan ganjil, nilainya dikalikan tiga dan ditambahkan satu. Proses ini berulang hingga nilai `n` mencapai 1, dan setiap suku deret dipisahkan oleh spasi. Akhirnya, program mencetak seluruh deret di satu baris, berhenti ketika nilai `n` sama dengan 1.