LAPORAN PRAKTIKUM
ALGORITMA PEMOGRAMAN II

MODUL III & IV

FUNGSI & PROSEDUR

Oleh:

NAMA: SHEILA DWI YULIANA

NIM: 2311102292

KELAS: IF-11-07

S1 TEKNIK INFORMATIKA

UNIVERSITY TELKOM PURWOKERTO

2024

I. DASAR TEORI

A. Fungsi

Fungsi adalah kumpulan instruksi yang memberikan atau menghasilkan suatu nilai dan biasanya memetakkan input ke nilai lain. Oleh karena itu, fungsi selalu menghasilkan atau mengembalikan nilai. Berikut adalah kondisi di mana subprogram dianggap sebagai fungsi:

1. Ada deklarasi tipe nilai yang dikembalikan, dan
2. Badan subprogram mengandung kata kunci return.

Jika suatu nilai biasanya diperlukan, maka fungsi digunakan. Misalnya, memberi nilai kepada suatu variabel, bagian dari ekspresi, bagian dari argumen subprogram, dll.

Karena itu, gunakan nama fungsi seperti kata benda dan kata sifat untuk menggambarkan nilai. Nama-nama fungsi seperti median, rerata, nilal terbesar, ketemu, selesal, dan sebagainya.

B. Prosedur

Salah satu cara untuk memahami prosedur adalah dengan memotong beberapa instruksi program menjadi instruksi baru. Ini dilakukan untuk mengurangi kompleksitas kode program yang ada pada program yang lebih besar. Ketika prosedur digunakan, prosedur mempengaruhi program utama secara langsung. Apabila:

1. Tidak ada deklarasi tipe nilai yang dikembalikan, dan
2. Badan subprogram tidak mengandung kata kunci return.

Nama prosedur harus berupa kata kerja atau nama proses, karena kedudukannya mirip dengan assignment atau instruksi dasar yang sudah ada sebelumnya. Untuk contoh, cetak, hitung, cari rata-rata, nilai, belok, mulai, dll.

II. GUIDED

1) Guided 1

Program:

```go
package main

import (
    "fmt"
)

func VolumeTabung(jari_jari, tinggi int) float64 {
    var LuasAlas, volume float64
    LuasAlas = 3.14 * float64(jari_jari*jari_jari)
    volume = LuasAlas * float64(tinggi)
    return volume
}

func main() {
    var r, t int
    var V1, V2 float64
    r = 5
    t = 10

    V1 = VolumeTabung(r, t)
    V2 = VolumeTabung(r, t) + VolumeTabung(15, t)

    fmt.Println("Volume Tabung 14 x 100:", VolumeTabung(14, 100))
    fmt.Println("V1:", V1)
    fmt.Println("V2:", V2)
}
```

Output:

```
Masukkan jari-jari tabung: 5
Masukkan tinggi dan massa jenis tabung kiri: 10 5
Masukkan tinggi dan massa jenis tabung kanan: 5 10
BALANCE
PS C:\Users\INTEL i3> go run "C:\Users\INTELI~1\AppD
Masukkan jari-jari tabung: 3
Masukkan tinggi dan massa jenis tabung kiri: 2 4
Masukkan tinggi dan massa jenis tabung kanan: 10 6
Selisih massa: -1470.27
```

Penjelasan:

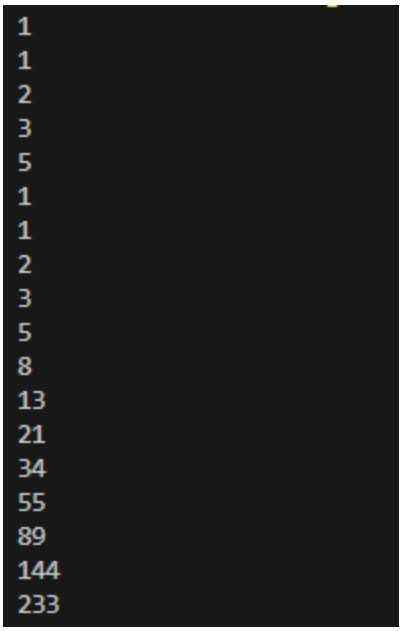
Program menghitung volume tabung dengan menggunakan rumusnya. Dalam fungsi main, program memiliki nilai jari-jari dan tinggi untuk beberapa tabung, kemudian memanggil fungsi volumetabung untuk menghitung volume.

2) Guided 2

Program:

```
package main

import (
    "fmt"
)

func cetakNFibo(n int){
    var f1, f2, f3 int
    f2 = 0;
    f3 = 1
    for i := 1; i<=n; i++ {
        fmt.Println(f3)
        f1 = f2
        f2 = f3
        f3 = f1 + f2
    }
}

func main(){
    var x int
    x = 5
    cetakNFibo(x)
    cetakNFibo(100)
}
```

Output:



Penjelasan:

Program ini untuk mencetak bilangan fibonachi sesuai masukkan. Main fungsi menerima masukkan berupa jumlah bilangan Fibonacci yang ingin dicetak, kemudian menggunakan perulangan for untuk mencetak setiap bilangan secara berurutan.

III. UNGUIDED

1). Unguided 1

Program:

```go
package main

import (
    "fmt"
)

// Fungsi untuk menghitung faktorial
func mencariFaktorial(n int) int {
    if n == 0 || n == 1 {
        return 1
    }
    return n * mencariFaktorial(n-1)
}

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

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

```go
func main() {
    // Memasukkan nilai a, b, c, dan d
    var a, b, c, d int
    fmt.Print("Masukkan 4 angka (a b c d): ")
    fmt.Scanf("%d %d %d %d", &a, &b, &c, &d)

    // Baris pertama: permutasi dan kombinasi a terhadap c
    P1 := permutation(a, c)
    C1 := combination(a, c)
    fmt.Println(P1, C1)

    // Baris kedua: permutasi dan kombinasi b terhadap d
    P2 := permutation(b, d)
    C2 := combination(b, d)
    fmt.Println(P2, C2)
}
```

Output:

```
Masukkan 4 angka (a b c d): 5 10 3 10
60 10
3628800 1
```

```
Masukkan 4 angka (a b c d): 8 0 2 0
56 28
1 1
```

Penjelasan:

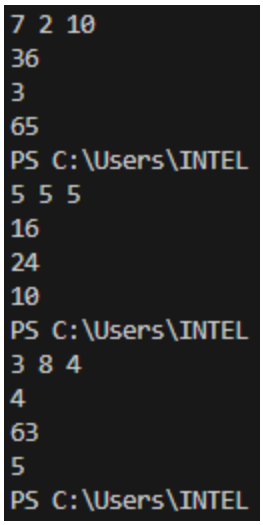
Menggunakan fungsi rekursif untuk menghitung factorial, dan kemudian menggunakan rumus permutasi dan kombinasi yang melibatkan factorial untuk menghitung kedua konsep tersebut. Pengguna memasukkan empat bilangan, program akan menghitung dan menampilkan hasil permutasi dan kombinasi dari dua pasang bilangan yang berbeda.

2). Unguided 2

Program:

```go
package main

import "fmt"

func fogoh(x int) int {
    return (x-1) * (x-1)
}

func gohof(x int) int {
    return x*x - 1
}

func hofog(x int) int {
    return (x-2)*(x-2) + 1
}

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

    fmt.Println(fogoh(a))
    fmt.Println(gohof(b))
    fmt.Println(hofog(c))
}
```

Output:



Penjelasan:
Program ini terdiri dari tiga fungsi yaitu fogoh, gohof, hofog. Fungsi fogoh untuk menghitung kuadrat selisih antara x dan 1, gohof menghitung selisih antara x dan 1, dan hofog menghitung selisih antara x dan 2, lalu menambahkan 1.

3). Unguided 3
Program:

```go
package main

import (
	"fmt"
	"math"
)

func jarak(x1, y1, x2, y2 float64) float64 {
	return math.Sqrt(math.Pow(x1-x2, 2) + math.Pow(y1-y2, 2))
}

func didalam(x, y, cx, cy, r float64) bool {
	return jarak(x, y, cx, cy) <= r
}

func main() {
	var cx1, cy1, r1, cx2, cy2, r2, x, y float64

	fmt.Scan(&cx1, &cy1, &r1)
	fmt.Scan(&cx2, &cy2, &r2)
	fmt.Scan(&x, &y)

	if didalam(x, y, cx1, cy1, r1) && didalam(x, y, cx2, cy2, r2) {
		fmt.Println("Titik di dalam lingkaran 1 dan 2")
	} else if didalam(x, y, cx1, cy1, r1) {
		fmt.Println("Titik di dalam lingkaran 1")
	} else if didalam(x, y, cx2, cy2, r2) {
		fmt.Println("Titik di dalam lingkaran 2")
	} else {
		fmt.Println("Titik di luar lingkaran 1 dan 2")
	}
}
```

Output:

```
1 1 5
8 8 4
2 2
Titik di dalam lingkaran 1
```

```
1 2 3
4 5 6
7 8
Titik di dalam lingkaran 2
```

```
5 10 15
-15 4 20
0 0
Titik di dalam lingkaran 1 dan 2
```

```
 1 1 5
 8 8 4
 15 20
 Titik di luar lingkaran 1 dan 2
```

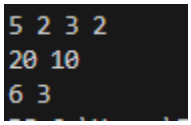
Penejelasan:
Program diminta untuk menentukan suatu titik pada dua lingkaran berbeda.
Program menghitung jarak antara dua titik, di dalam untuk mengetahui apakah suatu titik berada di dalam lingkaran. Program meminta pengguna untuk memasukkan titik koordinat pusaat dan jari-jari dari dua lingkaran, serta titik yang ingin diperiksa.

4). Unguided 4
Program:

```go
package main

import "fmt"

// Prosedur untuk menghitung faktorial
func factorial(n int, hasil *int) {
        *hasil = 1
        for i := 2; i <= n; i++ {
                *hasil *= i
        }
}

// Prosedur untuk menghitung permutasi
func permutation(n, r int, hasil *int) {
    var factN, factNR int
    factorial(n, &factN)
    factorial(n-r, &factNR)
    *hasil = factN / factNR
}
```

```go
// Prosedur untuk menghitung kombinasi
func combination(n, r int, hasil *int) {
    var factN, factR, factNR int
    factorial(n, &factN)
    factorial(r, &factR)
    factorial(n-r, &factNR)
    *hasil = factN / (factR * factNR)
}

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

        // Hitung permutasi dan kombinasi untuk pasangan pertama (a, c)
        permutation(a, c, &perm)
        combination(a, c, &comb)
        fmt.Println(perm, comb)

        // Hitung permutasi dan kombinasi untuk pasangan kedua (b, d)
        permutation(b, d, &perm)
        combination(b, d, &comb)
        fmt.Println(perm, comb)
}
```

Output:



Penjelasan:

Prosedur factorial menyimpan hasil dalam variabel

Prosedur permutasi, menghitung factorial dari n dan n-r, lalu menghitung permutasi.

Proseedur kombinasi, menghitung factorial dari n, r, dan n-r, kemudian menghitung kombinasi.

Fungsi utama untuk membaca input pengguna, memanggil proses permutasi dan kombinasi setiap pasangan bilangan dan mencetak hasil.

5). Unguided 5

Program :

```go
package main

import "fmt"

// Struktur peserta
type Peserta struct {
    nama string
    soal int
    skor int
}

// Fungsi untuk menghitung skor peserta
func hitungSkor(peserta *Peserta, waktu []int) {
    for _, t := range waktu {
        if t < 301 {
            peserta.soal++
            peserta.skor += t
        }
    }
}
```

```go
func main() {
    var peserta []Peserta
    var nama string
    var waktu [8]int

    // Membaca data peserta
    for {
        fmt.Scan(&nama)
        if nama == "Selesai" {
            break
        }
        for i := 0; i < 8; i++ {
            fmt.Scan(&waktu[i])
        }
        peserta = append(peserta, Peserta{nama, 0, 0})
        hitungSkor(&peserta[len(peserta)-1], waktu[:])
    }

    // Mencari pemenang
    pemenang := peserta[0]
    for _, p := range peserta[1:] {
        if p.soal > pemenang.soal || (p.soal == pemenang.soal && p.skor < pemenang.skor) {
            pemenang = p
        }
    }

    // Menampilkan hasil
    fmt.Printf("%s %d %d\n", pemenang.nama, pemenang.soal, pemenang.skor)
}
```

Output:

```
Astuti 20 50 301 301 62 72 75 10
Bertha 25 47 301 26 50 60 65 21
selesai
Bertha 7 249
```

Penjelasan:

Membaca data peserta dan waktu yang dibutuhkan untuk menyelesaikan soal. Kemudian program menghitung skor setiap peserta. Pemenang dipilih berdasarkan jumlah soal terbanyak; jika jumlah soal sama, pemenang adalah yang memiliki waktu tersisa terpanjang. Hasil akhir mencakup jumlah soal yang diselesaikan, jumlah waktu yang digunakan, dan nama pemenang. Struktur peserta digunakan dalam program untuk menyimpan data peserta, fungsi hitung skor digunakan untuk menghitung skor, dan perulangan digunakan untuk memproses data peserta dan menemukan pemenang.

6). Unguided 6

Program

```go
package main

import (
    "fmt"
)

func cetakDeret(n int) {
    for n != 1 {
        fmt.Printf("%d ", n)
        if n%2 == 0 {
            n = n / 2
        } else {
            n = 3*n + 1
        }
    }
    fmt.Println(n) // Mencetak suku terakhir, yaitu 1
}

func main() {
    var n int
    fmt.Print("Masukkan bilangan bulat positif kurang dari 1000000: ")
    fmt.Scan(&n)

    if n > 0 && n < 1000000 {
        cetakDeret(n)
    } else {
        fmt.Println("Bilangan harus positif dan kurang dari 1000000")
    }
}
```

Output:

```
Masukkan bilangan bulat positif kurang dari 1000000: 64
64 32 16 8 4 2 1
```

Penjelasan:

Deret bilangan yang dimulai dari bilangan bulat positif yang dimasukkan. jika suku sebelumnya genap, maka suku berikutnya adalah setengahnya, dan jika suku sebelumnya ganjil, maka suku berikutnya adalah tiga kali dirinya ditambah satu. Kode ini mengulangi prosedur hingga mencapai angka 1.

DAFTAR PUSTAKA

Modul 3 “FUNGSI”
Modul 4 “PROSEDUR”