LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

MODUL IV
PROSEDUR

Oleh :

NAMA : FAISAL KHOIRUDDIN

NIM : 2311102046

Kelas : IF-11-02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

1. Definisi Procedure

Prosedur dapat dianggap sebagai potongan beberapa instruksi program menjadi suatu instruksi baru yang dibuat untuk mengurangi kerumitan dari kode program yang kompleks pada suatu program yang besar. Prosedur akan menghasilkan suatu akibat atau efek langsung pada program ketika dipanggil pada program utama. Suatu subprogram dikatakan prosedur apabila:

1. Tidak ada deklarasi tipe nilai yang dikembalikan, dan
2. Tidak terdapat kata kunci return dalam badan subprogram

Kedudukannya prosedur sama seperti instruksi dasar yang sudah ada sebelumnya (assignment) dan/atau instruksi yang berasal dari paket (fmt), seperti fmt.Scan dan fmt.Print. Karena itu selalu pilih nama prosedur yang berbentuk kata kerja atau sesuatu yang merepresentasikan proses sebagai nama dari prosedur. Contoh: cetak, hitungRerata, cariNilai, belok, mulai, …

2. Deklarasi Procedure

Berikut ini adalah cara penulisan deklarasi prosedur pada notasi Pseudocode dan GoLang

	Notasi Algoritma
1	`procedure <nama procedure> (<params>)`
2	`kamus`
3	`    {deklarasi variabel lokal dari procedure}`
4	`    ...`
5	`algoritma`
6	`    {badan algoritma procedure}`
7	`    ...`
8	`endprocedure`
	Notasi dalam bahasa Go
9	`func <nama procedure> <(params)> {`
10	`    /* deklarasi variabel lokal dari procedure */`
11	`    ...`
12	`    /* badan algoritma procedure */`
13	`    ...`
14	`}`

Penulisan deklarasi ini berada di luar blok yang dari program utama atau func main() pada suatu program Go, dan bisa ditulis sebelum atau setelah dari blok program utama tersebut.

Contoh deklarasi prosedur mencetak n nilai pertama dari deret Fibonacci

Notasi Algoritma

```
1  procedure cetakNFibo(in n : integer)
2  kamus
3      f1, f2, f3, i : integer
4  algoritma
5      f2 ← 0
6      f3 ← 1
7      for i ← 1 to n do
8          output(f3)
9          f1 ← f2
10         f2 ← f3
11         f3 ← f1 + f2
12     endfor
13 endprocedure
```

Notasi dalam bahasa Go

```
14 func cetakNFibo(n int) {
15     var f1, f2, f3 int
16     f2 = 0
17     f3 = 1
18     for i := 1; i <= n; i++ {
19         fmt.println(f3)
20         f1 = f2
21         f2 = f3
22         f3 = f1 + f2
23     }
24 }
```


3. Cara Pemanggilan Procedure

Seperti yang sudah dijelaskan sebelumnya, suatu **prosedur hanya akan dieksekusi apabila dipanggil** baik secara langsung atau tidak langsung oleh program utama. Tidak langsung di sini maksudnya adalah prosedur dipanggil oleh program utama melalu perantara subprogram yang lain.

Pemanggilan suatu prosedure cukup mudah, yaitu dengan hanya **menuliskan nama beserta parameter atau argumen yang diminta dari suatu prosedur**. Sebagai contoh prosedur cetakNFibo di atas dipanggil dengan menuliskan namanya, kemudian sebuah variabel atau nilai integer tertentu sebagai argumen untuk paramter

n. Contoh:

	Notasi Algoritma
1	program contohprosedur
2	kamus
3	x : integer
4	algoritma
5	x ← 5
6	cetakNFibo(x) *{cara pemanggilan #1}*
7	cetakNFibo(100) *{cara pemanggilan #2}*
8	endprogram
	Notasi dalam bahasa Go
9	func main() {
10	var x int
11	x = 5
12	cetakNFibo(x) {cara pemanggilan #1}
13	cetakNFibo(100) {cara pemanggilan #2}
14	}

Dari contoh di atas terlihat bahwa cara pemanggilan dengan notasi pseudocode dan GoLang adalah sama. Argumen yang digunakan untuk parameter n berupa integer (sesuai deklarasi) yang terdapat pada suatu variabel (cara pemanggilan #1) atau nilainya secara langsung (cara pemanggilan #2).

4. Contoh Program dengan Procedure

Berikut ini adalah contoh penulisan prosedur pada suatu program lengkap. Buatlah sebuah program beserta prosedur yang digunakan untuk menampilkan suatu pesan error, warning atau informasi berdasarkan masukan dari user.

Masukan terdiri dari sebuah bilangan bulat **flag** (0 s.d. 2) dan sebuah string pesan **M**.

Keluaran berupa string pesan **M** beserta jenis pesannya, yaitu error, warning atau informasi berdasarkan nilai flag 0, 1 dan 2 secara berturut-turut.

```go
package main
import "fmt"

func main(){
    var bilangan int
    var pesan string
    fmt.Scan(&bilangan, &pesan)
    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"
    }
    fmt.Println(M,jenis)
}
```

```
D:\DEV\DEMO>go build contoh.go
D:\DEV\DEMO>contoh.exe
1 hello_world
```

Penulisan argumen pada parameter cetakPesan(pesan,bilangan) harus sesuai urutan tipe data pada func cetakPesan(M string, flag int), yaitu string kemudian integer.

5. Parameter

Suatu subprogram yang dipanggil dapat berkomunikasi dengan pemanggilnya melalui argumen yang diberikan melalui parameter yang dideklarasikan pada subprogramnya. Berikut ini jenis atau pembagian dari parameter. Berdasarkan letak penulisannya pada program, maka parameter dapat dikelompokkan menjadi dua, yaitu parameter formal dan parameter aktual.

```go
func volumeTabung(jari_jari,tinggi int) float64 {
    var luasAlas,volume float64

    luasAlas = 3.14 * float64(jari_jari * jari_jari)
    volume = luasAlas * 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(volumeTabung(14,100))
}
```

1. **Parameter Formal**

   Parameter formal adalah parameter yang ditulis pada saat deklarasi suatu subprogram, parameter ini berfungsi sebagai petunjuk bahwa argumen apa saja yang diperlukan pada saat pemanggilan subprogram. Sebagai contoh parameter **jari_jari, tinggi** pada deklarasi **fungsi volumeTabung** adalah parameter formal (teks berwarna merah). Artinya ketika memanggil volumeTabung maka kita harus mempersiapkan dua integer (berapapun nilainya) sebagai jari_jari dan tinggi.

2. **Parameter Formal**

   Sedangkan parameter aktual adalah argumen yang digunakan pada bagian parameter saat pemanggilan suatu subprogram. Banyaknya argumen dan tipe data yang terdapat pada paramter aktual harus mengikuti parameter formal. Sebagai contoh argumen r, t, 15, 14 dan 100 pada contoh kode di atas (teks berwarna biru) adalah parameter aktual, yang menyatakan nilai yang kita berikan sebagai jari-jari dan tinggi.

Selain itu parameter juga dikelompokkan berdasarkan alokasi memorinya, yaitu pass by value dan pass by reference.

1. **Pass by Value**

   Nilai pada parameter aktual akan disalin ke variabel lokal (parameter formal) pada subprogram. Artinya parameter aktual dan formal dialokasikan di dalam memori komputer dengan alamat memori yang berbeda. Subprogram dapat menggunakan nilai pada parameter formal tersebut untuk proses apapun, tetapi tidak dapat mengembalikan informasinya ke pemanggil melalui parameter aktual karena pemanggil tidak dapat mengakses memori yang digunakan oleh subprogram. Pass by value bisa digunakan baik oleh fungsi ataupun

prosedur. Pada notasi pseudocode, secara semua parameter formal pada fungsi adalah pass by value, sedangkan pada prosedur diberi kata kunci in pada saat penulisan parameter formal. Sedangkan pada bahasa pemrograman Go sama seperti fungsi pada pseudocode, tidak terdapat kata kunci khusus untuk parameter formal fungsi dan prosedur.

2. **Pass by Reference (Pointer)**

   Ketika parameter didefinisikan sebagai pass by reference, maka pada saat pemanggilan parameter formal akan berperan sebagai pointer yang menyimpan alamat memori dari parameter aktual. Sehingga perubahan nilai yang terjadi pada paramter formal tersebut akan berdampak pada parameter aktual. Artinya nilai terakhirnya akan dapat diketahui oleh si pemanggil setelah subprogram tersebut selesai dieksekusi. Pass by reference sebaiknya digunakan hanya untuk prosedur. Penulisan parameter pass by reference pada prosedur baik pseudocode dan Go menggunakan kata kunci atau identifier khusus. Pada pseudocode menggunakan kata kunci in/out, sedangkan pada bahasa Go diberi identifier asterik (*) sebelum tipe data di parameter formal yang menjadi pass by reference.

Catatan:

- Parameter pada fungsi sebaiknya adalah pass by value, hal ini dikarenakan fungsi bisa mengembalikan (return) nilai ke pemanggil dan tidak memberikan efek langsung pada program, walaupun tidak menutup kemungkinan menggunakan pass by reference.

- Penggunaan pass by reference sebaiknya pada prosedur karena prosedur tidak bisa mengembalikan nilai ke pemanggil. Dengan memanfaatkan pass by reference maka prosedur seolah-olah bisa mengirimkan nilai kepada si pemanggil.

Untuk lebih jelas perhatikan contoh sebuah subprogram yang digunakan

untuk menghitung persamaan berikut ini:

$$f(x,y) = 2x - \frac{y}{2} + 3$$

Notasi Algoritma

```
function f1(x,y : integer) → real
kamus
    hasil : real
algoritma
    hasil ← 2*x - 0.5*y + 3
    return hasil
endfunction

procedure f2(in x,y : integer, in/out
hasil:real)
algoritma
    hasil ← 2*x - 0.5*y + 3
endprocedure

program Contoh
kamus
    a,b : integer
    c : real
algoritma
    input(a,b)
    f2(a,b,c)
    output(c, f1(b,a))
endprogram
```

x dan y pada fungsi f1 dan prosedur f2 adalah **pass by value**,

sedangkan variabel **hasil** pada prosedur f2 adalah **pass by reference**.

Untuk pemanggilan dengan notasi pseudocode masih sama dengan materi yang sudah dipelajari sebelumnya

Notasi dalam bahasa Go

```
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.Scan(&a,&b)
    f2(a,b,&c)
    output(c, f1(b,a))
endprogram
```

x dan y pada fungsi f1 dan prosedur f2 adalah **pass by value**,

sedangkan variabel **hasil** pada prosedur f2 adalah **pass by reference**.

Karena variabel **hasil adalah pointer to float64**, maka untuk mengaksesnya menggunakan **simbol bintang (*)** pada variabelnya.

Pada bahasa Go saat pemanggilan prosedur f2, maka **parameter aktual** untuk **pass by reference** harus diberi **ampersand "&"**, contohnya &c

II. GUIDED

Source Code

```
package main

import "fmt"

// Fungsi untuk menghitung faktorial
func factorial(n int) int {
     if n == 0 {
          return 1
     }
     result := 1
     for i := 1; i <= n; i++ {
          result *= i
     }
     return result
}

// Prosedur untuk menghitung dan
menampilkan permutasi
func permutasi(n, r int) {
     hasilPermutasi := factorial(n) /
factorial(n-r)
     fmt.Printf("Permutasi dari %dP%d
adalah: %d\n", n, r, hasilPermutasi)
}

func main() {
     // Memanggil prosedur untuk menghitung
dan menampilkan permutasi
```

```
    n, r := 5, 3
    permutasi(n, r)
}
```

Screenshots Output

Deskripsi:

Program tersebut merupakan program menghitung permutasi. Program tersebut menghitung permutasi dengan rumus hasilPermutasi := factorial(n) / factorial(n-r). Program tersebut menampilkan hasil permutasi 5P3 adalah 60.

- package main ➔ paket utama program golang
- import "fmt" ➔ mengimpor fmt
- func factorial(n int) int { ➔ fungsi untuk menghitung faktorial
- if n == 0 { ➔ percabangan if jika n sama dengan 0
- return 1 ➔ return 1
- result := 1 ➔ inisialisasi result sama dengan 1
- for i := 1; i <= n; i++ { ➔ perulangan for jika i sama dengan 1; i kurang dari n; i++(increment)
- result *= i ➔ mengalikan result dengan i setiap iterasi
- return result ➔ mengembalikan hasil faktorial
- func permutasi(n, r int) { ➔ prosedur untuk menghitung dan menampilkan permutasi

- hasilPermutasi := factorial(n) / factorial(n-r) ➔ menghitung permutasi
- fmt.Printf("Permutasi dari %dP%d adalah: %d\n", n, r, hasilPermutasi) ➔ menampilkan permutasi
- func main() { ➔ merupakan fungsi utama
- n, r := 5, 3 ➔ Memanggil prosedur untuk menghitung dan menampilkan permutasi
- permutasi(n, r) ➔ memanggil nilai dari n dan r

III. Unguided

1. Skiena dan Revilla dalam Programming Challenges mendefinisikan sebuah deret bilangan. Deret dimulai dengan sebuah bilangan bulat n. Jika bilangan n saat itu genap, maka suku berikutnya adalah ½n, tetapi jika ganjil maka suku berikutnya bernilai 3n+1. Rumus yang sama digunakan terus menerus untuk mencari suku berikutnya. Deret berakhir ketika suku terakhir bernilai 1. Sebagai contoh jika dimulai dengan n=22, maka deret bilangan yang diperoleh adalah:

   22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

   Untuk suku awal sampai dengan 1000000, diketahui deret selalu mencapai suku dengan nilai 1.

   Buat program skiena yang akan mencetak setiap suku dari deret yang dijelaskan di atas untuk nilai suku awal yang diberikan. Pencetakan deret harus dibuat dalam prosedur cetakDeret yang mempunyai 1 parameter formal, yaitu nilai dari suku awal.

   prosedure cetakDeret(**in** n : **integer**)

   Masukan berupa satu bilangan integer positif yang lebih kecil dari 1000000.

   Keluaran terdiri dari satu baris saja. Setiap suku dari deret tersebut dicetak dalam baris yang dan dipisahkan oleh sebuah spasi.

No	Masukan	Keluaran
1	22	22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Source Code

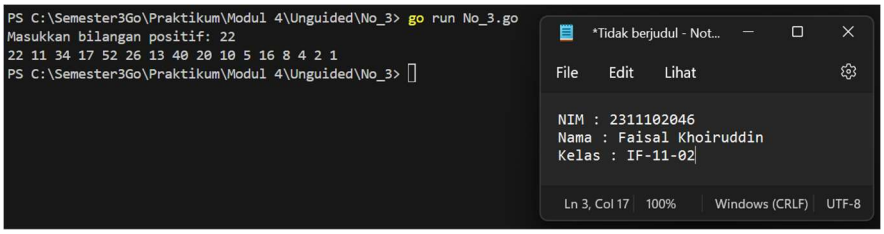
```
package main

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(1)
}

func main() {
    var n int
    fmt.Print("Masukkan bilangan positif:
     ")
    fmt.Scan(&n)
    if n < 1 || n >= 1000000 {
        fmt.Println("Bilangan harus lebih
     besar dari 0 dan kurang dari 1000000")
        return
    }
    cetakDeret(n)
}
```

Screenshots Output



Deskripsi:

Program tersebut merupakan program untuk mendefinisikan sebuah deret bilangan. Deret dimulai dengan sebuah bilangan bulat n. Jika bilangan n saat itu genap, maka suku berikutnya adalah ½n, tetapi jika ganjil maka suku berikutnya bernilai 3n+1. Program tersebut meminta prngguna untuk menginput bilangan integer posisitf. Output dari program tersebut yaitu deret dari bilangan yang telah diinputkan.

- package main ➔ paket utama program golang
- import "fmt" ➔ mengimpor fmt
- func cetakDeret(n int) { ➔ fungsi untuk mencetak deret bilangan dengan parameter n dengan tipe data integer
- for n != 1 { ➔ perulangan for dengan n tidak sama dengan 1
- fmt.Printf("%d ", n) ➔ menampilkan
- if n%2 == 0 { ➔ percabangan jika n % 2 sama dengan 0
- n = n / 2 ➔ n sama dengan n / 2
- } else { ➔ kalau tidak
- n = 3*n + 1 ➔ n sama dengan 3 kali n
- fmt.Println(1) ➔ menampilkan 1
- func main() { ➔ merupakan fungsi utama
- var n int ➔ deklarasi variabel n denga tipe data integer
- fmt.Print("Masukkan bilangan positif: ") ➔ menampilkan statement untuk Masukkan bilangan positif
- fmt.Scan(&n) ➔ menyimpan input ke dalam variabel n
- if n < 1 || n >= 1000000 { ➔ percabangan if jika n < 1 || n >=

1000000

- fmt.Println("Bilangan harus lebih besar dari 0 dan kurang dari 1000000") ➔ menampilkan statement Bilangan harus lebih besar dari 0 dan kurang dari 1000000
- return ➔ mengambalikan
- cetakDeret(n) ➔ cetak n