

## MODUL 4. PROSEDUR

### 4.1 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**, ...

### 4.2 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 $\leftarrow$ 0
6	f3 $\leftarrow$ 1
7	for i $\leftarrow$ 1 to n do
8	output(f3)
9	f1 $\leftarrow$ f2
10	f2 $\leftarrow$ f3
11	f3 $\leftarrow$ 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	}

Catatan: Kata kunci **in** pada contoh di atas akan dijelaskan pada materi parameter di modul 5 ini.

### 4.3 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 melalui perantara subprogram yang lain.

Pemanggilan suatu prosedur 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 parameter  $n$ . Contoh:

	Notasi Algoritma
1	program contohprosedur
2	kamus
3	x : integer
4	algoritma
5	x $\leftarrow$ 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).

#### 4.4 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.

```

1 package main
2 import "fmt"
3
4 func main(){
5     var bilangan int
6     var pesan string
7     fmt.Scan(&bilangan, &pesan)
8     cetakPesan(pesan, bilangan)
9 }
10
11 func cetakPesan(M string, flag int){
12     var jenis string = ""
13     if flag == 0 {
14         jenis = "error"
15     }else if flag == 1 {
16         jenis = "warning"
17     }else if flag == 2 {
18         jenis = "informasi"
19     }
20     fmt.Println(M, jenis)
21 }

```

```

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

```

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

## 4.5 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.

```
1 func volumeTabung(jari_jari,tinggi int) float64 {  
2     var luasAlas,volume float64  
3  
4     luasAlas = 3.14 * float64(jari_jari * jari_jari)  
5     volume = luasAlas * tinggi  
6     return volume  
7 }  
8  
9 func main() {  
10     var r,t int  
11     var v1,v2 float64  
12     r = 5; t = 10  
13     v1 = volumeTabung(r,t)  
14     v2 = volumeTabung(r,t) + volumeTabung(15,t)  
15     fmt.Println(volumeTabung(14,100))  
16 }
```

### 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 Aktual

Sedangkan parameter aktual adalah argumen yang digunakan pada bagian parameter saat pemanggilan suatu subprogram. Banyaknya argumen dan tipe data yang terdapat pada parameter 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 parameter 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	
<pre>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</pre>	<p>x dan y pada fungsi f1 dan prosedur f2 adalah <b>pass by value</b>,</p> <p>sedangkan variabel <b>hasil</b> pada prosedur f2 adalah <b>pass by reference</b>.</p> <p>Untuk pemanggilan dengan notasi pseudocode masih sama dengan materi yang sudah dipelajari sebelumnya</p>
Notasi dalam bahasa Go	
<pre>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(&amp;a,&amp;b)     f2(a,b,&amp;c)     output(c, f1(b,a)) endprogram</pre>	<p>x dan y pada fungsi f1 dan prosedur f2 adalah <b>pass by value</b>,</p> <p>sedangkan variabel <b>hasil</b> pada prosedur f2 adalah <b>pass by reference</b>.</p> <p>Karena variabel <b>hasil</b> adalah <b>pointer to float64</b>, maka untuk mengaksesnya menggunakan <b>simbol bintang (*)</b> pada variabelnya.</p> <p>Pada bahasa Go saat pemanggilan prosedur f2, maka <b>parameter aktual</b> untuk <b>pass by reference</b> harus diberi <b>ampersand "&amp;"</b>, contohnya <b>&amp;c</b></p>

#### 4.6 Soal Latihan Modul 4

- 1) Minggu ini, mahasiswa Fakultas Informatika mendapatkan tugas dari mata kuliah matematika

diskrit untuk mempelajari kombinasi dan permutasi. Jonas salah seorang mahasiswa, iseng untuk mengimplementasikannya ke dalam suatu program. Oleh karena itu bersediakah kalian membantu Jonas? (tidak tentunya ya :p)

**Masukan** terdiri dari empat buah bilangan asli  $a$ ,  $b$ ,  $c$ , dan  $d$  yang dipisahkan oleh spasi, dengan syarat  $a \geq c$  dan  $b \geq d$ .

**Keluaran** terdiri dari dua baris. Baris pertama adalah hasil permutasi dan kombinasi  $a$  terhadap  $c$ , sedangkan baris kedua adalah hasil permutasi dan kombinasi  $b$  terhadap  $d$ .

**Catatan:** permutasi (P) dan kombinasi (C) dari  $n$  terhadap  $r$  ( $n \geq r$ ) dapat dihitung dengan menggunakan persamaan berikut!

$$P(n, r) = \frac{n!}{(n-r)!}, \text{ sedangkan } C(n, r) = \frac{n!}{r!(n-r)!}$$

**Contoh**

No	Masukan	Keluaran	Penjelasan
1	5 10 3 10	60 10 3628800 1	$P(5, 3) = 5!/2! = 120/2 = 60$ $C(5, 3) = 5!/(3! \times 2!) = 120/12 = 10$ $P(10, 10) = 10!/0! = 3628800/1 = 3628800$ $C(10, 10) = 10!/(10! \times 0!) = 10!/10! = 1$
2	8 0 2 0	56 28 1 1	

Selesaikan program tersebut dengan memanfaatkan prosedur yang diberikan berikut ini!

```
procedure factorial(in n: integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n
 F.S. hasil berisi nilai faktorial dari n}

procedure permutation(in n,r : integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n dan r, dan n >= r
 F.S. hasil berisi nilai dari n permutasi r}

procedure combination(in n,r : integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n dan r, dan n >= r
 F.S. hasil berisi nilai dari n kombinasi r}
```

- 2) Kompetisi pemrograman tingkat nasional berlangsung ketat. Setiap peserta diberikan 8 soal yang harus dapat diselesaikan dalam waktu 5 jam saja. Peserta yang berhasil menyelesaikan soal paling banyak dalam waktu paling singkat adalah pemenangnya.

Buat program **gema** yang mencari pemenang dari daftar peserta yang diberikan. Program harus dibuat modular, yaitu dengan membuat prosedur `hitungSkor` yang mengembalikan total soal dan total skor yang dikerjakan oleh seorang peserta, melalui parameter formal. Pembacaan nama peserta dilakukan di program utama, sedangkan waktu pengerjaan dibaca di dalam prosedur.

**prosedure** `hitungSkor(in/out soal, skor : integer)`

Setiap baris **masukan** dimulai dengan satu string nama peserta tersebut diikuti dengan adalah 8 integer yang menyatakan berapa lama (dalam menit) peserta tersebut menyelesaikan soal. Jika tidak berhasil atau tidak mengirimkan jawaban maka otomatis dianggap menyelesaikan dalam waktu 5 jam 1 menit (301 menit).

Satu baris **keluaran** berisi nama pemenang, jumlah soal yang diselesaikan, dan nilai yang diperoleh. Nilai adalah total waktu yang dibutuhkan untuk menyelesaikan soal yang berhasil diselesaikan.

No	Masukan	Keluaran
1	Astuti 20 50 301 301 61 71 75 10 Bertha 25 47 301 26 50 60 65 21 Selesai	Bertha 7 294

**Keterangan:**

Astuti menyelesaikan 6 soal dalam waktu 287 menit, sedangkan Bertha 7 soal dalam waktu 294 menit. Karena Bertha menyelesaikan lebih banyak, maka Bertha menang. Jika keduanya menyelesaikan sama banyak, maka pemenang adalah yang menyelesaikan dengan total waktu paling kecil.

- 3) 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  $\frac{1}{2}n$ , 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.

**procedure** 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