

**LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2**

**MODUL 4
PENGENALAN PROSEDUR**



Oleh:

PANDIA ARYA BRATA

2311102076

IF – 11 - 02

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

2024

I. DASAR TEORI

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	
2	procedure <nama (<params>)
3	kamus
4	{ <i>deklarasi variabel lokal dari procedure</i> }
5	algoritma
6	{ <i>badan algoritma procedure</i> }
7	...
8	endprocedure
	Notasi dalam bahasa Go
9	func <nama procedure> <(params)> {
10	/* <i>deklarasi variabel lokal dari procedure</i> */
11	...
12	/* <i>badan algoritma procedure</i> */
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 }
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 }
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 pass by value,</p> <p>sedangkan variabel hasil pada prosedur f2 adalah pass by reference.</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(&a,&b) f2(a,b,&c) output(c, f1(b,a)) endprogram </pre>	<p>x dan y pada fungsi f1 dan prosedur f2 adalah pass by value,</p> <p>sedangkan variabel hasil pada prosedur f2 adalah pass by reference.</p> <p>Karena variabel hasil adalah pointer to float64, maka untuk mengaksesnya menggunakan simbol bintang (*) pada variabelnya.</p> <p>Pada bahasa Go saat pemanggilan prosedur f2, maka parameter aktual untuk pass by reference harus diberi ampersand "&", contohnya &c</p>

II. GUIDED

Sourcecode :

```
package main

import "fmt"

func hallaw(name string) {
    fmt.Println("Hallo ", name)
}

func main() {
    name1 := "Ervan"
    name2 := "pandia"
    hallaw(name1)
    hallaw(name2)
}
```

Screenshot program :

```
Hallo Ervan
Hallo pandia
PS C:\Alpro_Praktikum\Pandia Arya Brata_2311102076_Modul4>
```

Deskripsi program :

Program prosedur sederhana yang difungsikan untuk mencetak sebuah “Hallo Nama” dan di main kita gunakan prosedur ini 2x dengan menginisiasikan nama1 adalah Ervan dan nama2 adalah pandia. Selanjutnya kita melakukan pemanggilan prosedur tersebut.

III. UNGUIDED

Sourcecode :

```
package main

import "fmt"

func cetakDeret(n int) {
    var hasil int = n
    fmt.Print(hasil)
    for hasil != 1 {
        if hasil%2 == 0 {
            hasil = hasil / 2
            fmt.Print(hasil)
        } else {
            hasil = 3*hasil + 1
            fmt.Print(hasil)
        }
    }
}

func main() {
    var n int
    fmt.Print("Masukan bilangan : ")
    fmt.Scan(&n)

    cetakDeret(n)
}
```

Screenshot program :

```
Masukan bilangan : 22
221134175226134020105168421
PS C:\Alpro_Praktikum\Pandia Arya Brata_2311102076_Modul4>
```

Deskripsi program :

Program ini menerima sebuah bilangan bulat positif sebagai input, kemudian mencetak deret bilangan yang dihasilkan dari penerapan algoritma Collatz Conjecture pada bilangan input tersebut hingga mencapai nilai 1.