

**LAPORAN PRAKTIKUM  
PEMROGRAMAN BERORIENTASI OBJEK  
MODUL IV  
“PROSEDUR”**



**Oleh:**

**MUHAMMAD RAGIEL PRASTYO**

**2311102183**

**S1IF-11-02**

**S1 TEKNIK INFORMATIKA  
TELKOM UNIVERSITY 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	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	}

### 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 dimlnta dart 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).

#### 4.4 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.

```
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 ini ditulis saat deklarasi subprogram dan berfungsi sebagai petunjuk bahwa argumen apa pun yang diperlukan saat memanggil subprogram diperlukan.

Pada deklarasi fungsi `volumeTabung`, parameter `Jari_Jari` adalah parameter formal, yang berarti bahwa ketika kita memanggil `volumeTabung`, kita harus mempersiapkan dua integer, apa pun nilainya, sebagai `jari_jari` dan `tinggi`.

2. **Parameter Aktual:** Argumen yang digunakan pada bagian parameter saat memanggil subprogram disebut parameter aktual. Parameter formal harus mengikuti banyak argumen dan jenis data yang ada pada parameter aktual. Dalam contoh kode di atas, argumen `r`, `t`, `15`, `14` dan `100` 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.

## II. GUIDED

### Source Code


```
//MUHAMMAD RAGIEL PRASTYO
//2311102183
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)
}
```

### Screenshot Output



```
PS C:\Users\USER\OneDrive\Desktop\Alpro 2> go run "c:\Users\USER\OneDrive\Desktop\Alpro 2\Muhammad Ragiel Prastyo_2311102183_Modul4\Guided\guided1.go"
Permutasi dari 5P3 adalah: 60
PS C:\Users\USER\OneDrive\Desktop\Alpro 2>
```

### Penjelasan:

Program diatas menggunakan prosedur permutasi, yang menggunakan fungsi factorial untuk menghitung faktorial dari dua bilangan. Fungsi factorial mengembalikan faktorial dari bilangan n dengan mengalikan bilangan dari 1 hingga n. Prosedur permutasi kemudian menggunakan nilai faktorial ini untuk menghitung permutasi dari n dan r dengan menggunakan rumus  $n! / (n - r)!$ , dan mencetak

hasilnya. Program digunakan untuk memanggil fungsi main dengan nilai  $n = 5$  dan  $r = 3$ , dan kemudian menampilkan hasil permutasi ke layar.



### 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  $\frac{1}{4}n$ , 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.

**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.

*Source Code*

```
//MUHAMMAD RAGIEL PRASTYO
//2311102183
package main
import (
    "fmt"
)

// Fungsi cetakDeret untuk mencetak deret berdasarkan aturan yang diberikan
func cetakDeret(n int) {
    var deret []int

    for n != 1 {
        deret = append(deret, n)
        if n%2 == 0 {
            n = n / 2
        }
    }
    deret = append(deret, 1)
    fmt.Println(deret)
```

```

    } else {
        n = 3*n + 1
    }
}

deret = append(deret, 1)

// Mencetak deret
for i, v := range deret {
    if i == len(deret)-1 {
        fmt.Print(v)
    } else {
        fmt.Print(v, " ")
    }
}
}

func main() {
    var n int


    fmt.Print("Masukkan bilangan bulat positif (kurang dari 1000000): ")
    fmt.Scan(&n)

    if n <= 0 || n >= 1000000 {
        fmt.Println("Masukan tidak valid. Masukkan bilangan positif kurang dari 1000000.")
        return
    }

    cetakDeret(n)
}

```

### *Screenshot Output*



```

PS C:\Users\USER\OneDrive\Desktop\Alpro 2> go run "c:\Users\USER\OneDrive\Desktop\Alpro 2\Muhammad Ragiel Prastyo_2311102183_Modul4\Unguided\unguided1.go"
Masukkan bilangan bulat positif (kurang dari 1000000): 22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS C:\Users\USER\OneDrive\Desktop\Alpro 2>

```

### **Penjelasan:**

Program diatas menghitung dan mencetak deret bilangan sesuai dengan aturan yang ditetapkan oleh Skiena dan Revilla. Prosedur cetakDeret mengambil satu parameter, yaitu bilangan bulat positif  $n$ , dan menggunakan loop untuk menghasilkan deret hingga mencapai angka 1. Jika  $n$  genap, program membagi  $n$  dengan 2; jika  $n$  ganjil, program menghitung  $3n+1$ . Setiap suku yang dihasilkan ditambahkan ke dalam slice deret, yang kemudian dicetak dengan spasi sebagai pemisah. Di dalam fungsi main, program meminta pengguna untuk memasukkan

nilai  $n$ , memvalidasi input agar sesuai dengan rentang yang ditentukan, dan kemudian memanggil prosedur cetakDeret untuk menampilkan hasilnya.