

**LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK**

**MODUL IV
PROSEDUR**



Oleh:

Destia Ananda Putra

2311102176

IF-11-02

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

DASAR TEORI

Definisi Prosedur

Prosedur adalah sekelompok instruksi program yang dikemas menjadi sebuah instruksi baru untuk menyederhanakan kode pada program yang kompleks dan besar. Prosedur akan memberikan efek langsung pada program saat dipanggil dari program utama. Suatu subprogram disebut prosedur jika:

1. Tidak mendeklarasikan tipe nilai yang akan dikembalikan, dan
2. Tidak menggunakan kata kunci ``return`` dalam tubuh subprogram.

Prosedur memiliki fungsi yang sama dengan instruksi dasar (seperti assignment) atau instruksi dari paket (seperti ``fmt.Scan`` dan ``fmt.Print``). Oleh karena itu, nama prosedur sebaiknya berupa kata kerja atau sesuatu yang menunjukkan proses, misalnya: cetak, hitungRerata, cariNilai, belok, mulai,dll.

Deklarasi prosedur

Berikut ini notasi pada pseudocode dan notasi pada golang

	Notasi algoritma
1	procedure procedure> (<params>)
2	kamus
3	{deklarasi
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	}

Cara Pemanggilan Prosedur

Memanggil sebuah prosedur sangat sederhana, yaitu hanya dengan menuliskan nama prosedur beserta parameter atau argumen yang diperlukan. Sebagai contoh, prosedur ``cetakNFibo`` di atas dapat dipanggil dengan menyebutkan namanya, kemudian memberikan sebuah variabel atau nilai integer tertentu sebagai argumen untuk parameter ``n``.

	Notasi Algoritma
1	program contohprosedur
2	kamus
3	integer
4	algoritma
5	$x \leftarrow 5$
6	cetakNFibo(x)
7	cetakNFibo(100)
8	endprogram
	Notasi dalam bahasa Go
9	func main() {
10	var x int
11	5
12	cetakNFibo(x)
13	cetakNFibo(100)
14	}

Dari contoh di atas, dapat dilihat bahwa metode pemanggilan dengan notasi pseudocode dan GoLang memiliki kesamaan. Argumen yang digunakan untuk parameter $\backslash (n) \backslash$ bertipe integer (sesuai dengan deklarasi) dapat berupa variabel (seperti pada cara pemanggilan #1) atau dapat juga diisi dengan nilainya secara langsung (seperti pada cara pemanggilan #2).

Parameter

adalah cara bagi suatu subprogram untuk berkomunikasi dengan pemanggilnya melalui argumen yang disampaikan melalui parameter yang telah dideklarasikan di dalam subprogram. Parameter dapat dikelompokkan menjadi dua jenis berdasarkan posisinya dalam program, yaitu **parameter formal** dan **parameter aktual**.

1. Parameter Formal

Parameter formal adalah parameter yang dituliskan saat mendeklarasikan suatu subprogram. Parameter ini berfungsi sebagai petunjuk mengenai argumen yang dibutuhkan saat pemanggilan subprogram. Contohnya, parameter Jari_Jari dan Tinggi dalam deklarasi fungsi volumeTabung adalah parameter formal (ditandai dengan teks berwarna merah). Ini berarti bahwa saat memanggil volumeTabung, kita harus menyiapkan dua integer (dengan nilai apapun) sebagai jari-jari dan tinggi.

2. Parameter Aktual

Parameter aktual, di sisi lain, adalah argumen yang digunakan dalam parameter saat pemanggilan subprogram. Jumlah dan tipe data pada parameter aktual harus sesuai dengan parameter formal. Sebagai contoh, argumen r, t, 15, 14, dan 100 dalam contoh kode di atas (ditandai dengan teks berwarna biru) adalah parameter aktual yang menyatakan nilai yang diberikan sebagai jari-jari dan tinggi.

3. Parameter Aktual

Parameter aktual, di sisi lain, adalah argumen yang digunakan dalam parameter saat pemanggilan subprogram. Jumlah dan tipe data pada parameter aktual harus sesuai dengan parameter formal. Sebagai contoh, argumen *r*, *t*, 15, 14, dan 100 dalam contoh kode di atas (ditandai dengan teks berwarna biru) adalah parameter aktual yang menyatakan nilai yang diberikan sebagai jari-jari dan tinggi.

Catatan:

Parameter dalam fungsi sebaiknya menggunakan **pass by value*, karena fungsi dapat mengembalikan nilai ke pemanggil tanpa memberikan efek langsung pada program, meskipun tidak menutup kemungkinan untuk menggunakan ***pass by reference*. Penggunaan ***pass by reference* lebih disarankan pada prosedur, karena prosedur tidak dapat mengembalikan nilai kepada pemanggil. Dengan menggunakan **pass by reference*, prosedur seolah-olah dapat mengirimkan nilai kepada 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 : integer c : real algoritma input(a,b) f2(a,b,c) output(c, f1(b,a)) endprogram</pre>	<p><i>x</i> dan <i>y</i> pada fungsi <i>f1</i> dan prosedur <i>f2</i> adalah pass by value,</p> <p>sedangkan variabel <i>hasil</i> pada prosedur <i>f2</i> adalah pass by reference.</p>
Notasi dalam bahasa Go	
<pre>package main import "fmt"</pre>	<p><i>x</i> dan <i>y</i> pada fungsi <i>f1</i> dan prosedur <i>f2</i> adalah pass by value,</p> <p>Pada bahasa Go saat pemanggilan prosedur <i>f2</i>, maka parameter aktual untuk pass by reference harus diberi ampersand "&", contohnya &c</p>

II. GUIDED

1. Source Code

```
package main

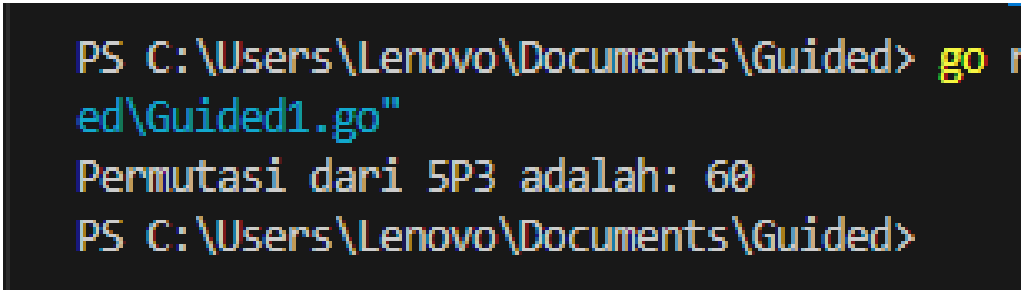
import "fmt"

// 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)
}
```

Output



```
PS C:\Users\Lenovo\Documents\Guided> go run ed\Guided1.go
Permutasi dari 5P3 adalah: 60
PS C:\Users\Lenovo\Documents\Guided>
```

Penjelasan

Program tersebut menghitung permutasi $P(n,r)$, yang menunjukkan jumlah cara untuk menyusun r elemen dari n elemen di mana urutan penting. Permutasi dihitung menggunakan rumus yaitu cara untuk menyusun r elemen dari n elemen dengan memperhatikan urutan. Proses ini melibatkan perhitungan faktorial $n!$ dan $(n-r)!$. Hasil perhitungan tersebut kemudian dicetak dalam bentuk pernyataan

III. Unguided

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 $1/2n$, 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.

```
package main

import "fmt"

func cetakDeret(n int, deret *[]int) {
    *deret = append(*deret, n)
    fmt.Print(n, " ")
    if n%2 == 0 {
        cetakDeret(n / 2, deret)
    } else if n != 1 {
        cetakDeret(3*n + 1, deret)
    } else {
```

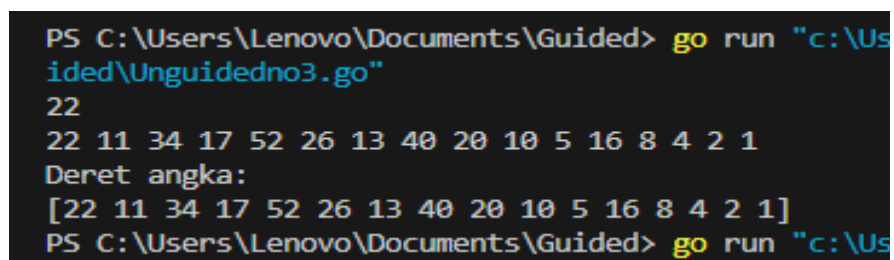
```

        fmt.Println()
    }
}

func main() {
    var n int
    fmt.Scan(&n)
    if n > 1000000 {
        fmt.Println("Bilangan harus <= 1000000")
    } else {
        var deret []int
        cetakDeret(n, &deret)
        fmt.Println("Deret angka:")
        fmt.Println(deret)
    }
}

```

Output



```

PS C:\Users\Lenovo\Documents\Guided> go run "c:\Users\Lenovo\Documents\Guided\Unguidedno3.go"
22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
Deret angka:
[22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1]
PS C:\Users\Lenovo\Documents\Guided> go run "c:\Users\Lenovo\Documents\Guided\Unguidedno3.go"

```

Penjelasan ;

Output program menunjukkan urutan angka yang dihasilkan dari algoritma Collatz berdasarkan input n. Program mencetak setiap langkah perhitungan. Menerima parameter n (bilangan bulat) dan deret pointer ke slice yang

menyimpan hasil dan Menambahkan n ke dalam slice deret dan mencetaknya. Memeriksa apakah n lebih besar dari 1.000.000. Jika ya, menampilkan pesan kesalahan. Jika n valid, fungsi cetakDeret dipanggil untuk menghitung dan mencetak deret angka berdasarkan nilai n. Setelah proses selesai, program mencetak daftar lengkap angka yang disimpan dalam slice deret.

IV. UNGUIDED

Source Code + Screenshot hasil program beserta penjelasan