

**LAPORAN PRAKTIKUM
ALGORITMA PROGRAM 3
MODUL 4
PENGENALAN FUNGSI**



Oleh:

ADITHANA DHARMA PUTRA

2311102207

IF – 11- 02

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

2024

I. DASAR TEORI

A. PENGERTIAN

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 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

B. DEKLARASI PROSEDUR

	Notasi Algoritma
1	procedure <nama procedure> (<pārams>)
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	}

C. 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 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).

D. 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 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 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 inout, sedangkan pada bahasa Go diberi identifier asterik (*) sebelum tipe data di parameter formal yang menjadi pass by reference.

II. GUIDED

1. Guided 1

Source Code

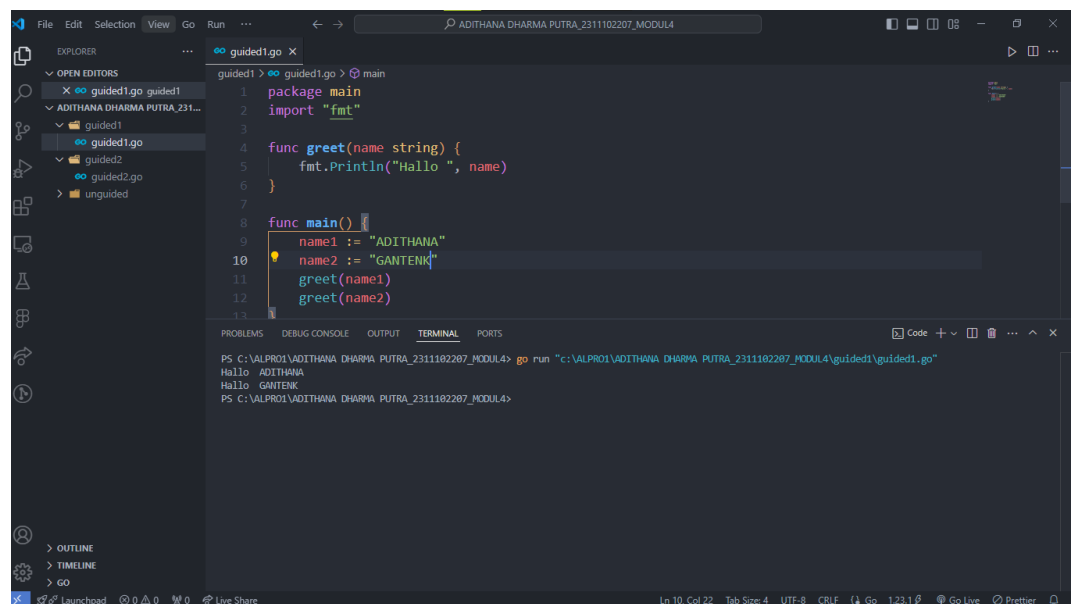
```
package main

import "fmt"

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

func main() {
    name1 := "ADITHANA"
    name2 := "GANTENK"
    greet(name1)
    greet(name2)
}
```

Screenshot



Deskripsi:

Program ini mendemonstrasikan implementasi penggunaan prosedur. Fungsi `greet` didefinisikan dengan satu parameter bertipe string bernama `name`. Di dalam fungsi ini, `fmt.Println` digunakan untuk mencetak pesan "Halo" diikuti dengan nilai dari parameter `name`. Fungsi ini memungkinkan program untuk menyapa siapa pun yang namanya diberikan sebagai argumen.

Di dalam `main`, dua variabel `name1` dan `name2` dideklarasikan dan diinisialisasi dengan nilai "ADITHANA" dan "GANTENK". Kemudian, fungsi `greet` dipanggil dua kali, masing-masing dengan `name1` dan `name2` sebagai argumen. Ini menghasilkan dua output di konsol: "Halo ADITHANA" dan "Halo GANTENK".

2. Guided 2**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)
}

```

Screenshot

```

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) {
    fmt.Println("Permutasi dari", n, "P", r, "adalah:", factorial(n) / factorial(n-r))
}

func main() {
    n, r := 5, 3
    permutasi(n, r)
}

```

PS C:\VALPRO1\ADITHANA DHARMA PUTRA_2311102207_MODUL4> go run "c:\valpro1\adithana dharm Putra_2311102207_MODUL4\guided2\guided2.go"

Permutasi dari 5P3 adalah: 60

PS C:\VALPRO1\ADITHANA DHARMA PUTRA_2311102207_MODUL4>

Deskripsi:

Fungsi factorial didefinisikan untuk menghitung faktorial dari sebuah bilangan bulat n. Jika n adalah 0, fungsi ini mengembalikan 1. Jika tidak, fungsi ini menggunakan loop untuk mengalikan semua bilangan dari 1 hingga n dan mengembalikan hasilnya. Selanjutnya, prosedur permutasi didefinisikan untuk menghitung dan menampilkan permutasi dari dua bilangan n dan r. Prosedur ini memanggil fungsi factorial untuk menghitung faktorial dari n dan n-r, kemudian membagi hasil faktorial n

dengan faktorial $n-r$ untuk mendapatkan hasil permutasi. Hasil ini kemudian dicetak ke konsol menggunakan `fmt.Printf`. di dalam `main`. di mana dua variabel `n` dan `r` diinisialisasi dengan nilai 5 dan 3. Prosedur permutasi kemudian dipanggil dengan `n` dan `r` sebagai argumen, menghasilkan output “Permutasi dari 5P3 adalah: 60” di konsol.

III. UNGUIDED

1. Unguided 1

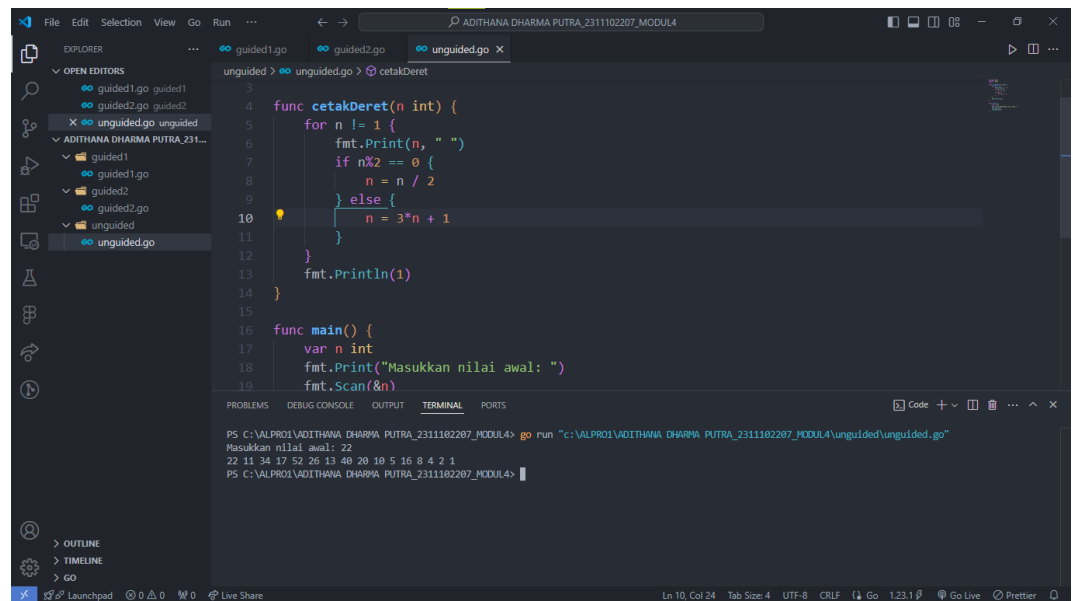
Source Code

```
package main
import "fmt"

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

func main() {
    var n int
    fmt.Print("Masukkan nilai awal: ")
    fmt.Scan(&n)
    cetakDeret(n)
}
```

Screenshot



```
3
4 func cetakDeret(n int) {
5     for n != 1 {
6         fmt.Print(n, " ")
7         if n%2 == 0 {
8             n = n / 2
9         } else {
10            n = 3*n + 1
11        }
12    }
13    fmt.Println()
14 }
15
16 func main() {
17     var n int
18     fmt.Print("Masukkan nilai awal: ")
19     fmt.Scan(&n)
```

PS C:\VALPRO1\ADITHANA DHARMA PUTRA_2311182287_MODULE4> go run "c:\valpro1\ADITHANA DHARMA PUTRA_2311182287_MODULE4\unguided\unguided.go"

Masukkan nilai awal: 22

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

PS C:\VALPRO1\ADITHANA DHARMA PUTRA_2311182287_MODULE4>

Deskripsi

Program dimulai dengan Fungsi cetakDeret didefinisikan untuk mencetak deret angka yang dimulai dari nilai n yang diberikan sebagai argumen. Di dalam fungsi ini, terdapat loop yang terus berjalan selama nilai n tidak sama dengan 1. Pada setiap iterasi, nilai n dicetak ke konsol diikuti dengan spasi. Jika n adalah bilangan genap, maka n dibagi dua. Jika n adalah bilangan ganjil, maka n diubah menjadi 3 kali n ditambah 1. Loop ini berlanjut hingga n menjadi 1, dan kemudian angka 1 dicetak ke konsol.

Pada main, sebuah variabel n dideklarasikan untuk menyimpan nilai awal yang dimasukkan oleh pengguna. Program meminta pengguna untuk memasukkan nilai awal melalui konsol menggunakan fmt.Scan. Setelah nilai n diperoleh, fungsi cetakDeret dipanggil dengan n sebagai argumen untuk mencetak deret angka sesuai dengan aturan yang telah dijelaskan.