

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 4
PROSEDUR**



Oleh:

HENDWI SAPUTRA

2311102218

IF-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 | } |

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 ← 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).

4.5 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 }

```

informatics lab

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.

3. 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.

4. 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.

II. GUIDED

GUIDED I

Source Code

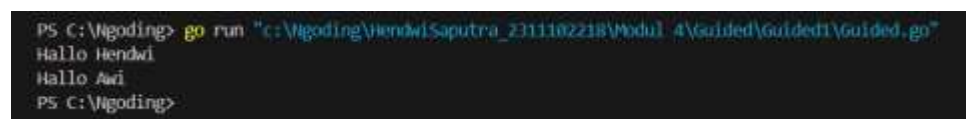
```
package main

import "fmt"

func sapa(nama string) {
    fmt.Println("Hallo", nama)
}

func main() {
    nama1 := "Hendwi"
    nama2 := "Awi"
    sapa(nama1)
    sapa(nama2)
}
```

Screenshot



```
PS C:\Ngoding> go run "c:\Ngoding\HendwiSaputra_2311102218\Modul 4\Guided\Guided1\Guided.go"
Hallo Hendwi
Hallo Awi
PS C:\Ngoding>
```

Deskripsi:

Program ini adalah program yang mencetak sapaan untuk dua nama yang berbeda. Fungsi sapa menerima satu parameter berupa string nama, dan mencetak sapaan "Hallo" diikuti oleh nama tersebut. Dalam fungsi main, dua variabel nama1 dan nama2 masing-masing diisi dengan string "Hendwi" dan "Awi". Fungsi sapa kemudian dipanggil dua kali, masing-masing dengan parameter nama1 dan nama2, sehingga mencetak sapaan untuk kedua nama tersebut.

GUIDED II

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
    menampilkanpermutasi
    n, r := 5, 3
    permutasi(n, r)
}
```

Screenshot



```
PS C:\Ngoding> go run "c:\Ngoding\kendiSaputra_2311182218\Modul 4\Guided\Guided2\Guided2.go"
Permutasi dari 5P3 adalah: 60
PS C:\Ngoding>
```

Deskripsi:

Program ini dimulai dengan mendefinisikan fungsi factorial yang menghitung faktorial dari suatu bilangan n. Fungsi ini mengembalikan nilai 1 jika n adalah 0, dan mengalikan semua bilangan dari 1 hingga n jika tidak. Kemudian, terdapat prosedur permutasi yang menerima dua parameter n

dan r . Prosedur ini menghitung permutasi dengan membagi faktorial n dengan faktorial dari $n-r$, dan mencetak hasilnya dalam format yang sesuai.

Dalam fungsi main, program memanggil prosedur permutasi dengan parameter $n = 5$ dan $r = 3$, lalu mencetak hasil permutasi tersebut. Singkatnya, program ini menghitung dan menampilkan permutasi $5P3$, yang dalam hal ini adalah 60.

III. UNGUIDED

UNGUIDED I

Source Code

```
package main

import "fmt"

func printDeret(n int) {
    for n != 1 {
        fmt.Print(n, " ")

        if n%2 == 0 {
            n /= 2
        } else {
            n = 3*n + 1
        }
    }
    fmt.Println(1)
}

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

    fmt.Print("Keluaran: ")
    printDeret(bilAwal)
}
```

Screenshot



```
PS C:\Vgoding> go run "c:\Vgoding\wendalSaputra_2311182218\Modul 4\Unguided\unguided1\unguided1.go"
Masukan: 22
Keluaran: 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS C:\Vgoding>
```

Deskripsi:

Program dimulai dengan meminta pengguna untuk memasukkan sebuah bilangan awal. Fungsi printDeret kemudian mencetak deret tersebut dengan aturan: jika bilangan tersebut genap, dibagi dua; jika ganjil, dikalikan tiga dan ditambah satu. Proses ini berlanjut hingga bilangan tersebut menjadi satu. Hasilnya adalah program ini menampilkan deret bilangan yang dihasilkan sampai mencapai angka satu.