LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

MODUL IV

PROSEDUR

Oleh:

NAMA : AJI TRI PRASETYO

NIM : 2311102064

KELAS : IF 11 02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

❖ Pengertian Prosedur

Prosedur merupakan kumpulan instruksi program yang digabungkan menjadi satu instruksi baru untuk menyederhanakan kode dalam program yang kompleks dan besar. Prosedur memberikan dampak langsung pada program saat dipanggil dari program utama. Sebuah subprogram dikategorikan sebagai prosedur jika:

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

Prosedur berfungsi sama seperti instruksi dasar (seperti penugasan) atau instruksi dari paket (misalnya `fmt.Scan` dan `fmt.Print`). Oleh karena itu, nama prosedur sebaiknya berupa kata kerja atau sesuatu yang menggambarkan suatu proses, seperti: cetak, hitungRerata, cariNilai, belok, mulai, dan sebagainya.

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 Memanggil Prosedur

Untuk memanggil sebuah prosedur, caranya cukup mudah, yaitu dengan menuliskan nama prosedur diikuti dengan parameter atau argumen yang dibutuhkan. Sebagai contoh, prosedur `cetakNFibo` dapat dipanggil dengan menyebutkan namanya dan memberikan sebuah variabel atau nilai integer tertentu sebagai argumen untuk parameter `n`.

	Notasi Algoritma

1
2
3
4
5
6
7
8
	program contohprosedur
	kamus
	  integer
	algoritma
	  x ← 5
	  cetakNFibo(x)
	  cetakNFibo(100)
	endprogram

	Notasi dalam bahasa Go

9
10
11
12
13
14
	func main() {
	  var x int
	  5
	  cetakNFibo(x)
	  cetakNFibo(100)
	}

Dari contoh di atas, dapat dilihat bahwa metode pemanggilan dengan notasi pseudocode dan GoLang memiliki kesamaan. Argumen yang digunakan untuk parameter \(n \) 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**

Parameter adalah cara bagi suatu subprogram untuk berinteraksi dengan pemanggilnya melalui argumen yang disampaikan melalui parameter yang telah dideklarasikan dalam subprogram. Parameter dapat dibedakan menjadi dua jenis berdasarkan posisinya dalam program: *parameter formal* dan *parameter aktual*.

1. **Parameter Formal**

Parameter formal adalah parameter yang dideklarasikan saat membuat subprogram. Parameter ini berfungsi sebagai indikator tentang argumen yang diperlukan saat subprogram dipanggil. Sebagai contoh, parameter `Jari_Jari` dan `Tinggi` dalam deklarasi fungsi `volumeTabung` adalah parameter formal (ditandai dengan warna merah). Ini berarti bahwa saat memanggil `volumeTabung`, kita harus menyediakan dua nilai integer untuk jari-jari dan tinggi.

2. **Parameter Aktual**

Parameter aktual adalah argumen yang digunakan dalam pemanggilan subprogram sesuai dengan parameter formal. Jumlah dan tipe data dari parameter aktual harus cocok dengan parameter formal. Sebagai contoh, argumen `r`, `t`, `15`, `14`, dan `100` dalam contoh kode (ditandai dengan warna biru) adalah parameter aktual yang mewakili nilai jari-jari dan tinggi.

Catatan:

Parameter dalam fungsi sebaiknya menggunakan **pass by value**, karena fungsi hanya mengembalikan nilai kepada pemanggil tanpa mengubah program secara langsung. Namun, **pass by reference** juga dapat digunakan, meskipun lebih disarankan untuk prosedur. Prosedur, yang tidak dapat mengembalikan nilai, dapat menggunakan ***pass by reference** untuk "mengirimkan" nilai kembali 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

```
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
```

x dan y pada fungsi f1 dan prosedur f2 adalah **pass by value**,

sedangkan variabel **hasil** pada prosedur f2 adalah **pass by reference**.

Untuk pemanggilan dengan notasi pseudocode masih sama dengan materi yang sudah dipelajari sebelumnya

Notasi dalam bahasa Go

```
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
}

                        float64
func main(){
    var a,b int; var c
    fmt.Scan(&a,&b)
    f2(a,b,&c)
    output(c, f1(b,a))
endprogram
```

x dan y pada fungsi f1 dan prosedur f2 adalah **pass by value**,

sedangkan variabel **hasil** pada prosedur f2 adalah **pass by reference**.

Karena variabel **hasil adalah pointer to float64**, maka untuk mengaksesnya menggunakan **simbol bintang** (*) pada variabelnya.

Pada bahasa Go saat pemanggilan prosedur f2, maka **parameter aktual** untuk **pass by reference** harus diberi **ampersand** "&", contohnya &c

II. GUIDED

Source Code + Screenshot hasil program beserta penjelasan

SOAL

NO 1.

Source code:

```
package main

import "fmt"

func main() {
   var bilangan int
   var pesan string

   fmt.Scan(&bilangan, &pesan)

   cetakPesan(pesan, bilangan)
}

func cetakPesan(M string, flag int) {
   var jenis string

   if flag == 0 {
      jenis = "error"
   } else if flag == 1 {
      jenis = "warning"
   } else if flag == 2 {
      jenis = "informasi"
   }

   fmt.Println(M, jenis)
}
```

Output :

```
PS C:\Users\MyBook Z Series> go run "C:\Users\MYBOOK~1\AppData\Local\Temp\tempCodeRunnerFile.go"
1 HELLO_WORLD
HELLO_WORLD warning
PS C:\Users\MyBook Z Series>
```

Penjelasan :

Program di atas meminta input berupa sebuah bilangan dan sebuah pesan dari pengguna. Berdasarkan nilai bilangan tersebut, program menentukan jenis pesan (error, warning, atau informasi) yang akan ditampilkan. Fungsi `cetakPesan` menerima dua parameter, yakni pesan (`M`) dan bilangan (`flag`). Berdasarkan nilai `flag`, jenis pesan diatur dengan kondisi if-else: `0` untuk error, `1` untuk warning, dan `2` untuk informasi. Setelah itu, pesan dan jenis pesan tersebut dicetak ke layar.

III. UNGUIDED

Source Code + Screenshot hasil program beserta penjelasan

SOAL 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. 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 sklena yang akan mencetak setiap suku dari deret yang dijelaskan di atas untuk nilai suku awal yang diberikan. Pencetakan deret harus dibuat dalam prosedur cetak Deret 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 :

```go
package main

import "fmt"

func cetakDeret(n int) {
    for n != 1 {
        fmt.Printf("%d ", n)
        n = hitungPanjangLangkah(n)
    }
    fmt.Println(n)
}

func hitungPanjangLangkah(n int) int {
    if n%2 == 0 {
        return n / 2
    }
    return 3*n + 1
}

func main() {
    var n int
    fmt.Print("Masukkan nilai suku awal (int positif kurang dari 1000000): ")
    fmt.Scan(&n)

    if n > 0 && n < 1000000 {
        cetakDeret(n)
    } else {
        fmt.Println("nilai input tidak valid,Input harus bilangan positif dan kurang dari 1000000.")
    }
}

```

Output :

```
PS C:\Users\MyBook Z Series> go run "C:\Users\MYBOOK~1\AppData\Local\Temp\tempCodeRunnerFile.go"
Masukkan nilai suku awal (int positif kurang dari 1000000): 22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS C:\Users\MyBook Z Series>
```

Penjelasan :

Program ini menerima input bilangan positif dari pengguna dan mencetak deret Collatz (atau deret 3n + 1), dimulai dari bilangan tersebut hingga mencapai 1. Fungsi `cetakDeret` mencetak setiap angka dalam deret selama nilai `n` tidak sama dengan 1. Fungsi `hitungPanjangLangkah` digunakan untuk menghitung langkah berikutnya: jika angka genap, dibagi 2, jika ganjil, dikalikan 3 dan ditambah 1. Input divalidasi untuk memastikan bilangan yang dimasukkan positif dan kurang dari 1.000.000. Jika valid, deret akan dicetak; jika tidak, pesan kesalahan ditampilkan.