LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

MODUL IV

PROSEDUR

Oleh:

NAMA : AHMAD TITANA NANDA PRAMUDYA

NIM : 2311102042

KELAS : IF 11 02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. 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.

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`.

Parameter

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. 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

II. GUIDED

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\Eko Puji Susanto\go\2311102042_Ahmad Titana Nanda Pramudya _Modul4> go run guided.go
1 Hallo_sayang
Hallo_sayang warning
PS C:\Users\Eko Puji Susanto\go\2311102042_Ahmad Titana Nanda Pramudya _Modul4>
```

Penjelasan :

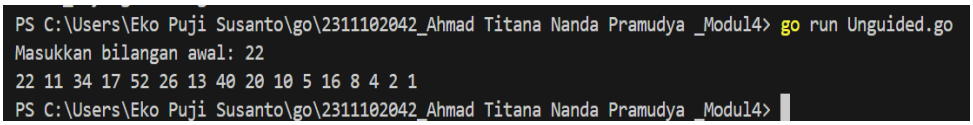
Program ini mengambil input dua buah nilai dari pengguna: sebuah bilangan integer dan sebuah pesan string. Berdasarkan nilai integer yang dimasukkan, program akan menambahkan jenis pesan (error, warning, atau informasi) dan mencetaknya bersama dengan pesan yang dimasukkan.

III. UNGUIDE

Source code :

```
package main

import "fmt"

func cetakDeret(n int) {

   for n != 1 {
      fmt.Printf("%d ", n)
      if n%2 == 0 {
         n = n / 2
      } else {
         n = 3*n + 1
      }
   }

   fmt.Println(n)
}

func main() {
   var n int
   fmt.Print("Masukkan bilangan awal: ")
   fmt.Scan(&n)


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

Output :



Penjelasan :
Program ini menghasilkan deret bilangan berdasarkan aturan yang diberikan oleh Skiena dan Revilla dalam Programming Challenges, yaitu:

- Jika bilangan saat ini genap, suku berikutnya adalah setengah dari bilangan tersebut (n / 2).
- Jika bilangan saat ini ganjil, suku berikutnya adalah 3n + 1.
- Deret berakhir ketika mencapai angka 1.

Program menerima input berupa bilangan bulat positif yang kurang dari 1.000.000, dan mencetak setiap suku deret hingga berakhir di angka 1, dengan setiap suku dipisahkan oleh spasi. Implementasi ini menggunakan struktur perulangan untuk mengolah setiap suku berdasarkan sifat genap atau ganjil dari bilangan tersebut.

Dengan aturan ini, deret selalu akan mencapai angka 1, seperti ditunjukkan oleh hasil untuk berbagai input awal.