LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 4

PROSEDUR

Oleh:

NAUFAL THORIQ MUZHAFFAR

2311102078

IF-11-02

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Prosedur adalah blok pernyataan yang dapat digunakan berulang kali dalam suatu program. Prosedur tidak akan dijalankan secara otomatis saat halaman dimuat. Prosedur akan dijalankan dengan memanggil prosedur tersebut.

Deklarasi Prosedur

Untuk membuat (sering disebut sebagai mendeklarasikan) suatu prosedur, lakukan hal berikut:

- Gunakan kata kunci **func**.
- Tentukan nama untuk prosedur tersebut, diikuti tanda kurung **()**.
- Terakhir, tambahkan kode yang menentukan apa yang harus dilakukan prosedur tersebut, di dalam kurung kurawal **{}**.

Sintaks

```
func NamaProsedur() {
  // code yang akan dieksekusi
}
```

Memanggil Prosedur

Prosedur tidak langsung dieksekusi. Prosedur tersebut "disimpan untuk penggunaan nanti", dan akan dieksekusi saat dipanggil.

Dalam contoh di bawah ini, kita membuat prosedur bernama "PesanSaya()". Tanda kurung kurawal pembuka ({) menunjukkan awal kode prosedur, dan tanda kurung kurawal penutup (}) menunjukkan akhir prosedur. Prosedur tersebut akan menampilkan "Ini code yang dieksekusi!". Untuk memanggil prosedur, cukup tulis namanya diikuti dua tanda kurung ():

Contoh

```
package main
import "fmt"

func PesanSaya() {
  fmt.Println("Ini code yang dieksekusi!")
}

func main() {
```

```
  PesanSaya() // call the function
}
```

Aturan Penamaan untuk Prosedur

- Nama prosedur harus dimulai dengan huruf
- Nama prosedur hanya boleh berisi karakter alfanumerik dan garis bawah (A-z, 0-9, dan _)
- Nama prosedur peka huruf besar-kecil
- Nama prosedur tidak boleh berisi spasi
- Jika nama prosedur terdiri dari beberapa kata, teknik yang diperkenalkan untuk penamaan variabel multikata dapat digunakan

Parameter dan Argumen

Informasi dapat diteruskan ke prosedur sebagai parameter. Parameter bertindak sebagai variabel di dalam prosedur. Parameter dan tipenya ditentukan setelah nama prosedur, di dalam tanda kurung. Anda dapat menambahkan parameter sebanyak yang diinginkan, cukup pisahkan dengan koma:

Sintaks

```
func NamaProsedur(param1 type, param2 type, param3 typ
e) {
  // code yang akan dieksekusi
}
```

Contoh Prosedur dengan Parameter

Contoh berikut memiliki prosedur dengan satu parameter (nama) bertipe string. Saat prosedur Keluarga() dipanggil, kami juga meneruskan nama (misalnya Liam), dan nama tersebut digunakan di dalam prosedur, yang menghasilkan beberapa nama depan yang berbeda, tetapi nama belakang yang sama:

Contoh

```
package main
import ("fmt")

func Keluarga(nama string) {
  fmt.Println("Hello", nama, "Refsnes")
```

```
}

func main() {
  Keluarga("Liam")
  Keluarga("Jenny")
  Keluarga("Anja")
}
```

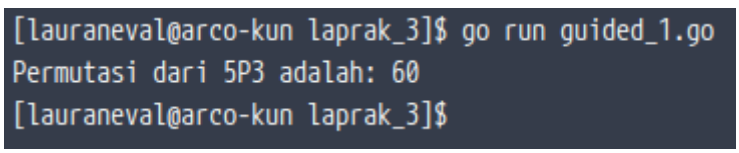
II. GUIDED

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)
}
```

Screnshoot Output



Penjelasan Program

Program di atas berfungsi untuk menghitung dan menampilkan hasil permutasi dari dua angka, n dan r.

Fungsi factorial: Menghitung faktorial dari suatu bilangan n. Jika n adalah 0, fungsi akan mengembalikan nilai 1. Jika lebih besar dari 0, fungsi ini menghitung hasil perkalian berturut-turut dari 1 hingga n.

Prosedur permutasi: Menghitung nilai permutasi dari nPr dengan rumus permutasi. Hasil perhitungannya ditampilkan ke layar menggunakan fmt.Printf.

Fungsi main: Memanggil prosedur permutasi dengan nilai n = 5 dan r = 3, sehingga program menampilkan hasil permutasi 5P3.

III. UNGUIDED

UNGUIDED 1

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 ½n, 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 cetakDeret 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.

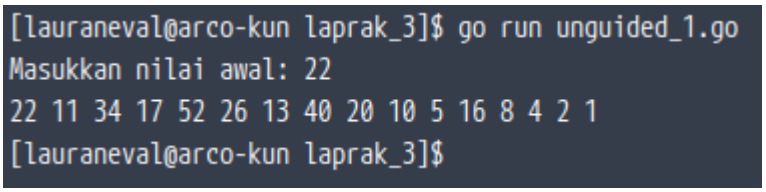
No	Masukan	Keluaran
1	22	22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Source Code

```
package main

import "fmt"


// Prosedur Deret

func cetakDeret(n int) {

     for n != 1 {

          fmt.Printf("%d ", n)
```

```
            if n%2 == 0 {
                  n = n / 2
            } else {
                  n = 3*n + 1
            }
      }
      fmt.Printf("%d\n", n) // Cetak angka 1 di
akhir
}


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


      // Panggil prosedur cetakDeret dengan nilai n
      cetakDeret(n)
}
```

Screenshot Output



Deskripsi Program

Program meminta input dari pengguna berupa angka awal untuk deret. Angka ini diteruskan ke prosedur cetakDeret untuk diproses dan dicetak deretnya.

Pada prosedur cetakDeret(n int): prosedur ini mencetak setiap angka dalam deret.

Program di bawah ini bertujuan untuk mencetak deret angka berdasarkan aturan berikut:

- Jika angka genap, maka angka berikutnya adalah hasil bagi angka tersebut dengan 2.
- Jika angka ganjil, maka angka berikutnya adalah hasil dari 3 kali angka tersebut ditambah 1.
- Proses ini berlanjut sampai angka mencapai 1.