LAPORAN PRAKTIKUM
ALGORITMA DAN PEMOGRAMAN 2

MODUL IV

PROCEDURE

Oleh:

YASVIN SYAHGANA

2311102065

S1 IF 11 02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

Procedure atau function adalah blok kode yang memiliki nama dan dapat dipanggil untuk menjalankan serangkaian instruksi. Prosedur didefinisikan menggunakan kata kunci func, dan dapat menerima parameter serta mengembalikan nilai. Prosedur digunakan untuk mengelompokkan kode yang dapat digunakan kembali, sehingga memudahkan pengelolaan dan pembacaan kode.

Berikut adalah komponen utama dari sebuah prosedur:

A. Kata kunci func, menandakan awal dari sebuah fungsi. Setiap fungsi dideklarasikan menggunakan kata kunci func. Kata kunci ini menandakan bahwa blok kode yang diikuti oleh func adalah sebuah fungsi. Fungsi Void adalah entitas mandiri yang dapat dipanggil untuk menjalankan tugas tertentu. Fungsi void bisa menerima input berupa parameter, dan setelah menjalankan tugasnya.

B. Nama fungsi adalah identitas unik yang digunakan untuk memanggil atau mengeksekusi fungsi di tempat lain dalam program. Nama ini harus mengikuti aturan penamaan variabel, yaitu diawali dengan huruf atau underscore (_), diikuti oleh kombinasi huruf, angka, atau underscore, dan harus deskriptif sehingga jelas tugas fungsi tersebut.

   Nama fungsi penting karena memberikan petunjuk tentang apa yang dilakukan oleh fungsi tersebut. Fungsi yang baik harus memiliki nama yang jelas dan deskriptif, mencerminkan tindakan atau operasi yang dilakukan oleh fungsi tersebut. Nama yang deskriptif membantu programmer lain (atau diri sendiri di masa mendatang) untuk memahami tujuan fungsi hanya dengan melihat namanya, tanpa perlu memeriksa implementasi internalnya.

C. Parameter (opsional): Daftar parameter yang diterima oleh fungsi, bersama dengan tipe datanya.
    - Fungsi dapat menerima nol atau lebih parameter.
    - Setiap parameter harus ditentukan dengan tipe datanya.
    - Jika fungsi mengembalikan nilai, tipe nilai kembalian ditulis setelah daftar parameter.
    - Fungsi bisa mengembalikan lebih dari satu nilai.

    Contoh Fungsi tanpa Nilai Kembali (Void Function) Fungsi bisa tidak mengembalikan apa pun, mirip dengan konsep void dalam bahasa lain. Dalam kasus ini, tipe kembalian bisa diabaikan

```
func cetakPesan(pesan string){
   fmt.Println(pesan)
}
```

D. Tipe pengembalian (opsional), adalah tipe data dari nilai atau nilai-nilai yang dikembalikan oleh fungsi. Penulisan tipe pengembalian ini bersifat opsional, jika fungsi tidak mengembalikan nilai apa pun, maka kita bisa mengabaikan deklarasi tipe pengembalian. Namun, jika fungsi mengembalikan nilai, tipe pengembalian harus ditulis setelah daftar parameter. Fungsi di Go dapat mengembalikan:
    - Satu nilai seperti fungsi bisa mengembalikan satu nilai dengan tipe tertentu (misalnya int, float64, string, dll.).
    - Beberapa nilai contohnya mendukung fitur multi-value return, di mana fungsi bisa mengembalikan lebih dari satu nilai sekaligus. Ini sangat berguna untuk mengembalikan hasil operasi sekaligus status operasi (misalnya hasil dan error).

II. GUIDED

Guided 1 | Source Code + Screenshot hasil program beserta penjelasan

```
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)
}
```

Output

```
PS D:\YASVIN SYAHGANAA> go run "d:\YASVIN SYAHGANAA\guided1.go"
Permutasi dari 5P3 adalah: 60
PS D:\YASVIN SYAHGANAA>
```

Deskripsi

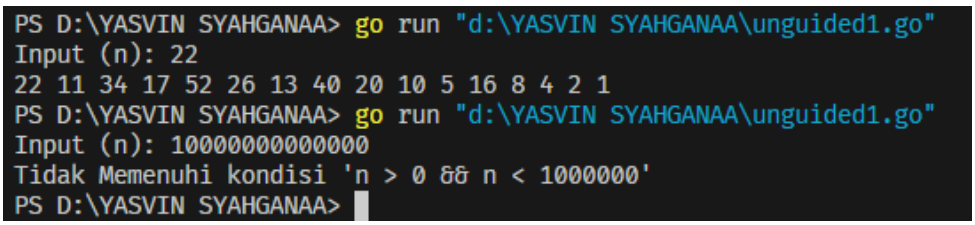
Program ini digunakan untuk menghitung permutasi dengan cara menggunakan prosedur, yang dimana dengan inputan konstan, program ini memiliki alur yang dimana **n** akan dihitung ke prosedur faktorial yang dimana **n** akan diklasifikasi apakah n = 0, atau tidak. Jika tidak akan melakukan perulangan **i** sebanyak **n** kemudian **i** akan dikalikan terus menerus dan di masukkan kedalam variable **result.** Kemudian hasil dari faktorisasi **n** akan dibagi dengan hasil faktorisasi (**n-r)** Kemudian output akan tanpil pada fungsi main.

III. UNGUIDED

Source Code + Screenshot hasil program beserta penjelasan

```
package main

import "fmt"

func printBaris(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("Input (n): ")
	fmt.Scan(&n)

	if n > 0 && n < 1000000 {
		printBaris(n)
	} else {
		fmt.Println("Tidak Memenuhi kondisi 'n > 0 &&
n < 1000000'")
	}
}
```

Screenshoot Output



Deskripsi Program

Program di atas merupakan salah satu masalah matematika klasik. Program ini digunakan untuk menampilkan deret angka yang dihasilkan dari aturan Collatz yaitu jika angka tersebut genap, maka dibagi dengan 2, dan jika

ganjil, maka dikalikan dengan 3 lalu ditambahkan 1. Proses ini berulang hingga angka tersebut menjadi 1. Cara kerja program diatas adalah dengan cara *user* akan menginputkan sebuah bilangan bulat positif <= 1000000. Apabaila inputas *user* memenuhi kondisi tersebut maka program akan menjalankan sebuah fungsi yang dimana Jika genap, nilai **n** dibagi 2. Jika ganjil, nilai **n** dikalikan 3 dan ditambahkan 1. Hingga nilai **n** menjadi 1.