LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2

MODUL 4

PROSEDUR

Oleh:

MUHAMMAD AGHA ZULFADHLI

2311102015

S1-IF11-02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

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.

2. Tidak terdapat kata kunci return dalam badan subprogram.

	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	`}`

Prosedur hanya akan dijalankan ketika dipanggil, baik secara langsung maupun tidak langsung oleh program utama. Pemanggilan tidak langsung berarti prosedur tersebut dipanggil melalui subprogram lain yang menjadi perantara. Cara memanggil prosedur cukup sederhana, yaitu dengan menuliskan nama prosedur beserta parameter atau argumen yang dibutuhkan.

Berdasarkan letak penulisannya pada program, maka parameter dapat dikelompokkan menjadi dua,yaitu parameter formal dan parameter aktual. 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. Sedangkan parameter aktual adalah argumen yang digunakan pada bagian parameter saat pemanggilan suatu subprogram. Banyaknya argumen dan tipe data yang terdapat pada paramter aktual harus mengikuti parameter formal.

Selain itu parameter juga dikelompokkan berdasarkan alokasi memorinya, yaitu pass by value dan pass by reference. Pass by Value mendaparkan 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. Sedangkan 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 paramter 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.

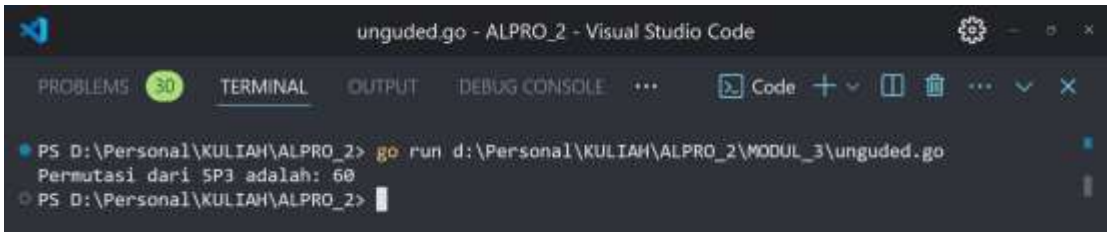
II. GUIDED

1. 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)
}
```

Screenshoot program



Deskripsi program

Program ini memiliki fungsi factorial untuk menghitung faktorial dari bilangan bulat, yang digunakan dalam prosedur permutasi. Prosedur ini menghitung

permutasi nPr menggunakan rumus n! / (n-r)! dan menampilkan hasilnya. Di bagian main, permutasi dari 5P3 dihitung dan hasilnya dicetak.

III. UNGUIDED

1. Unguided 3

Source code

```
package main

import "fmt"

func main() {
    var n int
    fmt.Scan(&n)
    if n > 0 && n < 1000000 {
        cetakDeret(n)
    }
}

func cetakDeret(n int) {
    fmt.Print(n, " ")
    if n%2 == 0 {
        cetakDeret(n / 2)
    } else if n != 1 {
        cetakDeret(3*n + 1)
    }
}
```

Screenshoot program

Deskripsi program

Program ini bertujuan untuk mencetak deret bilangan berdasarkan aturan tertentu menggunakan rekursi. Pertama, program meminta input bilangan `n`, lalu memanggil fungsi `cetakDeret` jika `n` berada di antara 1 dan 1.000.000. Fungsi `cetakDeret` mencetak nilai `n` dan kemudian memeriksa apakah `n` genap atau ganjil. Jika `n` genap, fungsi dipanggil kembali dengan nilai `n/2`. Jika `n` ganjil (selain 1), fungsi dipanggil lagi dengan nilai `3n + 1`. Deret akan berhenti ketika nilai `n` menjadi 1.