LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL IV

PROSEDUR

Disusun Oleh :

Raihan Ramadhan/2311102040

IF-11-05

Dosen Pengampu :

Arif Amrulloh, S.Kom., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Prosedur dapat diartikan sebagai sekumpulan instruksi program yang dipecah menjadi instruksi baru dengan tujuan menyederhanakan kode program yang rumit, terutama dalam program besar. Ketika dipanggil dalam program utama, prosedur akan memberikan dampak atau efek langsung pada program. Sebuah subprogram disebut sebagai prosedur jika memenuhi dua kondisi berikut:

1. Tidak mendeklarasikan tipe nilai yang dikembalikan, dan

2. Tidak menggunakan kata kunci return di dalam tubuh subprogram.

Berikut ini adalah cara penulisan deklarasi prosedur pada notasi Pseudocode dan GoLang.

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 }
```

Deklarasi ini dituliskan di luar blok program utama atau fungsi main() dalam program Go, dan dapat ditempatkan sebelum atau setelah blok program utama tersebut.

Cara Memanggil Prosedur

Seperti yang telah dijelaskan sebelumnya, prosedur hanya akan dijalankan jika dipanggil, baik secara langsung maupun tidak langsung oleh program utama. Pemanggilan tidak langsung berarti prosedur tersebut dipanggil melalui subprogram lain yang menjadi perantara.

Pemanggilan prosedur itu cukup mudah; cukup dengan menuliskan nama prosedur disertai parameter atau argumen yang dibutuhkan. Sebagai contoh, prosedur cetakNFibo dapat dipanggil dengan menyebutkan namanya, diikuti dengan sebuah variabel atau nilai integer tertentu sebagai argumen untuk parameter n.

Parameter

Sebuah subprogram yang dipanggil dapat berinteraksi dengan pemanggilnya melalui argumen yang diberikan melalui parameter yang dideklarasikan dalam subprogram tersebut. Berikut adalah jenis atau pembagian parameter. Berdasarkan posisinya dalam program, parameter dapat dibagi menjadi dua kategori: parameter formal dan parameter aktual.

1. Parameter Formal

Parameter formal adalah parameter yang dituliskan saat mendeklarasikan sebuah subprogram. Parameter ini berfungsi sebagai panduan mengenai argumen apa saja yang diperlukan saat subprogram tersebut dipanggil. Sebagai contoh, parameter jari_jari dan tinggi dalam deklarasi fungsi volumeTabung merupakan parameter formal (teks berwarna merah). Ini berarti, ketika memanggil fungsi volumeTabung, kita harus menyiapkan dua nilai integer (dengan nilai berapapun) sebagai jari_jari dan tinggi.

2. Parameter Aktual

Sementara itu, parameter aktual adalah argumen yang digunakan dalam bagian parameter saat memanggil suatu subprogram. Jumlah dan tipe data dari argumen yang terdapat pada parameter aktual harus sesuai dengan parameter formal.Sebagai contoh, argumen r, t, 15, 14, dan 100 dalam contoh kode di atas (teks berwarna biru) adalah parameter aktual, yang menunjukkan nilai yang kita berikan sebagai jari-jari dan tinggi.Selain itu, parameter juga dapat dikelompokkan berdasarkan alokasi memorinya, yaitu pass by value dan pass by reference.

1. Pass by Value

Pada metode ini, nilai dari parameter aktual akan disalin ke variabel lokal (parameter formal) di dalam subprogram. Ini berarti bahwa parameter aktual dan formal dialokasikan di lokasi memori yang berbeda. Subprogram dapat menggunakan nilai dari parameter formal untuk berbagai proses, tetapi tidak dapat mengembalikan informasi ke pemanggil melalui parameter aktual, karena pemanggil tidak dapat mengakses memori yang digunakan oleh subprogram. Pass by value dapat digunakan baik oleh fungsi maupun prosedur. Dalam notasi pseudocode, semua parameter formal dalam fungsi adalah pass by value, sedangkan dalam prosedur, parameter formal diberi kata kunci "in" saat penulisannya. Dalam bahasa pemrograman Go, seperti halnya pada fungsi dalam pseudocode, tidak ada kata kunci khusus untuk parameter formal fungsi dan prosedur.

2. Pass by Reference (Pointer)

Ketika parameter didefinisikan sebagai pass by reference, saat pemanggilan, parameter formal akan berfungsi sebagai pointer yang menyimpan alamat memori dari parameter aktual. Dengan demikian, perubahan nilai yang terjadi pada parameter formal juga akan mempengaruhi parameter aktual. Ini berarti nilai terakhir dari parameter tersebut dapat diketahui oleh pemanggil setelah subprogram selesai dieksekusi. Pass by reference sebaiknya digunakan hanya untuk prosedur.

II. Guided

1. Guided 1

Source Code

```go
package main

import "fmt"

func main() {
    var a, b int
    fmt.Scan(&a, &b)
    if a >= b {
        permutasi(a, b)
    } else {
        permutasi(b, a)
    }
}

// Prosedur faktorial
func faktorial(n int, result *int) {
    *result = 1
    for i := 1; i <= n; i++ {
        *result *= i
    }
}

// Prosedur permutasi
func permutasi(n, r int) {
    var faktN, faktNR int
    faktorial(n, &faktN)      // Hitung faktorial(n)
    faktorial(n-r, &faktNR)   // Hitung faktorial(n-r)
    result := faktN / faktNR  // Hasil perhitungan permutasi
    fmt.Println(result)       // Cetak hasil
}
```

Screenshot Output

```
PS D:\Kuliah\Matkul\SEMESTER 3\ALPRO PRAKTEK\PRAKTIKUM\MODUL IV>
_1.go"
2 5
20
PS D:\Kuliah\Matkul\SEMESTER 3\ALPRO PRAKTEK\PRAKTIKUM\MODUL IV>
```

Deskripsi Program

Program ini menghitung permutasi dari dua angka yang dimasukkan oleh pengguna. Pertama, program meminta pengguna untuk memasukkan dua bilangan bulat. Kemudian, program membandingkan kedua angka tersebut dan memastikan angka yang lebih besar dipakai sebagai nilai n untuk menghitung permutasi. Program menggunakan dua prosedur: satu untuk menghitung faktorial dari suatu angka dan satu lagi untuk menghitung permutasi. Hasil perhitungan permutasi akan ditampilkan di layar.

2. **Guided 2**

Source Code

```go
package main

import "fmt"

var sisi int

func main() {
    fmt.Print("Masukan Panjang Sisi: ")
    fmt.Scan(&sisi)

    var luas, keliling int
    LuasPersegi(sisi, &luas)
    KelilingPersegi(sisi, &keliling)

    fmt.Printf("Luas Persegi adalah: %d\n", luas)
    fmt.Printf("Keliling Persegi adalah: %d\n", keliling)
}

// Prosedur LuasPersegi tanpa return
func LuasPersegi(sisi int, result *int) {
    *result = sisi * sisi
}

// Prosedur KelilingPersegi tanpa return
func KelilingPersegi(sisi int, result *int) {
    *result = 4 * sisi
}
```

Screenshoot Output

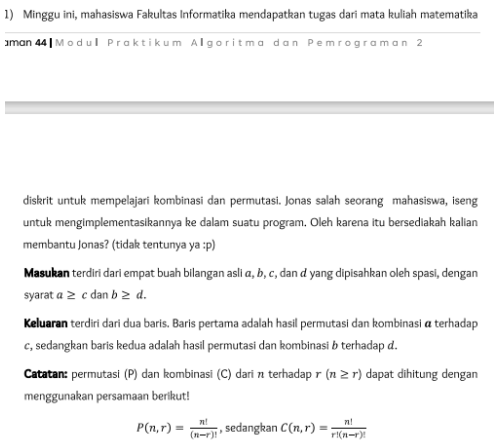
```
PS D:\Kuliah\Matkul\SEMESTER 3\ALPRO PRAKTEK\PRAKTIKUM\MODUL IV>
_2.go"
Masukan Panjang Sisi: 5
Luas Persegi adalah: 25
Keliling Persegi adalah: 20
PS D:\Kuliah\Matkul\SEMESTER 3\ALPRO PRAKTEK\PRAKTIKUM\MODUL IV>
```

Deskripsi Program

Program ini digunakan untuk menghitung luas dan keliling sebuah persegi. Pertama, program meminta pengguna untuk memasukkan panjang sisi persegi. Setelah itu, program menghitung luas dengan mengalikan panjang sisi dengan dirinya sendiri. Kemudian, program juga menghitung keliling dengan mengalikan panjang sisi dengan 4. Hasil dari luas dan keliling persegi akan ditampilkan di layar.

III. Unguided

1. Unguided 1



1) Minggu ini, mahasiswa Fakultas Informatika mendapatkan tugas dari mata kuliah matematika

ɔman 44 | Modul Praktikum Algoritma dan Pemrograman 2

diskrit untuk mempelajari kombinasi dan permutasi. Jonas salah seorang mahasiswa, iseng untuk mengimplementasikannya ke dalam suatu program. Oleh karena itu bersediakah kalian membantu Jonas? (tidak tentunya ya :p)

Masukan terdiri dari empat buah bilangan asli a, b, c, dan d yang dipisahkan oleh spasi, dengan syarat $a \geq c$ dan $b \geq d$.

Keluaran terdiri dari dua baris. Baris pertama adalah hasil permutasi dan kombinasi a terhadap c, sedangkan baris kedua adalah hasil permutasi dan kombinasi b terhadap d.

Catatan: permutasi (P) dan kombinasi (C) dari n terhadap r ($n \geq r$) dapat dihitung dengan menggunakan persamaan berikut!

$$P(n,r) = \frac{n!}{(n-r)!}, \text{ sedangkan } C(n,r) = \frac{n!}{r!(n-r)!}$$

Source Code

```go
package main

import (
    "fmt"
)

// Fungsi untuk menghitung faktorial
func factorial(n int, hasil *int) {
    *hasil = 1
    for i := 1; i <= n; i++ {
        *hasil *= i
    }
}
```

```go
// Fungsi untuk menghitung permutasi P(n, r)
func permutasi(n int, r int, hasil *int) {
    var fn, fn_r int
    factorial(n, &fn)
    factorial(n-r, &fn_r)
    *hasil = fn / fn_r
}

// Fungsi untuk menghitung kombinasi C(n, r)
func kombinasi(n int, r int, hasil *int) {
    var fn, fr, fn_r int
    factorial(n, &fn)
    factorial(r, &fr)
    factorial(n-r, &fn_r)
    *hasil = fn / (fr * fn_r)
}

func main() {
    var a, b, c, d int

    // Meminta input dari pengguna dalam satu baris
    fmt.Print("Masukkan nilai a, b, c, dan d : ")
    fmt.Scanf("%d %d %d %d", &a, &b, &c, &d)

    // Cek syarat a ≥ c dan b ≥ d
    if a >= c && b >= d {
        var hasilPermutasi1, hasilKombinasi1, hasilPermutasi2,
hasilKombinasi2 int

        // Hitung permutasi dan kombinasi untuk a, c
        permutasi(a, c, &hasilPermutasi1)
        kombinasi(a, c, &hasilKombinasi1)

        // Hitung permutasi dan kombinasi untuk b, d
        permutasi(b, d, &hasilPermutasi2)
        kombinasi(b, d, &hasilKombinasi2)

        // Tampilkan hasil
        fmt.Println("Hasil:")
        fmt.Printf("P(%d,%d) = %d\n", a, c, hasilPermutasi1)
        fmt.Printf("C(%d,%d) = %d\n", a, c, hasilKombinasi1)
        fmt.Printf("P(%d,%d) = %d\n", b, d, hasilPermutasi2)
        fmt.Printf("C(%d,%d) = %d\n", b, d, hasilKombinasi2)
    } else {
```

```
        fmt.Println("Syarat a ≥ c dan b ≥ d tidak terpenuhi")
    }
}
```

Screenshot Output

```
PS D:\Kuliah\Matkul\SEMESTER 3\ALPRO PRAKTEK\PRAKTIKUM\MODUL IV>
ed_1.go"
Masukkan nilai a, b, c, dan d : 5 10 3 10
Hasil:
P(5,3) = 60
C(5,3) = 10
P(10,10) = 3628800
C(10,10) = 1
PS D:\Kuliah\Matkul\SEMESTER 3\ALPRO PRAKTEK\PRAKTIKUM\MODUL IV>
```

Deskripsi Program

Program ini menghitung permutasi dan kombinasi berdasarkan empat angka yang dimasukkan oleh pengguna. Pertama, program meminta pengguna untuk memasukkan nilai a, b, c, dan d dalam satu baris. Program kemudian memeriksa apakah syarat a ≥ c dan b ≥ d terpenuhi. Jika syarat tersebut terpenuhi, program akan menghitung permutasi dan kombinasi untuk pasangan (a, c) dan (b, d). Hasil perhitungan permutasi dan kombinasi ditampilkan di layar, sedangkan jika syarat tidak terpenuhi, program akan memberi tahu pengguna bahwa syarat tersebut tidak terpenuhi.

2. Unguided 2

Kompetisi pemrograman tingkat nasional berlangsung ketat. Setiap peserta diberikan 8 soal yang harus dapat diselesaikan dalam waktu 5 jam saja. Peserta yang berhasil menyelesaikan soal paling banyak dalam waktu paling singkat adalah pemenangnya.

Buat program **gema** yang mencari pemenang dari daftar peserta yang diberikan. Program harus dibuat modular, yaitu dengan membuat prosedur hitungSkor yang mengembalikan total soal dan total skor yang dikerjakan oleh seorang peserta, melalui parameter formal. Pembacaan nama peserta dilakukan di program utama, sedangkan waktu pengerjaan dibaca di dalam prosedur.

```
prosedure hitungSkor(in/out soal, skor : integer)
```

Setiap baris **masukan** dimulai dengan satu string nama peserta tersebut diikuti dengan adalah 8 integer yang menyatakan berapa lama (dalam menit) peserta tersebut menyelesaikan soal. Jika tidak berhasil atau tidak mengirimkan jawaban maka otomatis dianggap menyelesaikan dalam waktu 5 jam 1 menit (301 menit).

Satu baris **keluaran** berisi nama pemenang, jumlah soal yang diselesaikan, dan nilai yang diperoleh. Nilai adalah total waktu yang dibutuhkan untuk menyelesaikan soal yang berhasil diselesaikan.

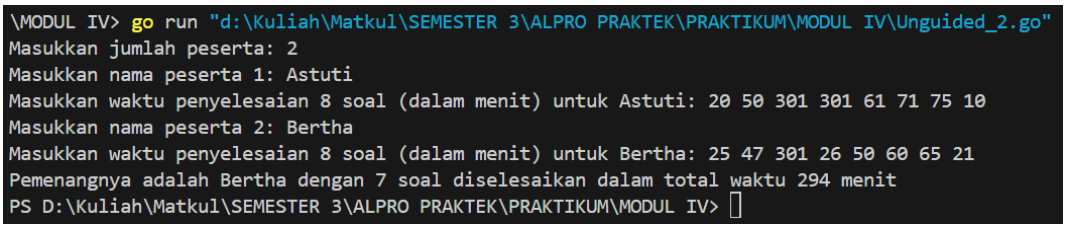
No	Masukan	Keluaran
1	Astuti 20 50 301 301 61 71 75 10 Bertha 25 47 301 26 50 60 65 21 Selesai	Bertha 7 294

Source Code

```go
package main

import (
    "fmt"
)

// Fungsi hitungSkor menghitung jumlah soal yang diselesaikan dan
menambahkan total waktu ke parameter skor dan waktu
func hitungSkor(soal *[8]int, skor *int, totalWaktu *int) {
    *skor = 0
    *totalWaktu = 0

    for i := 0; i < 8; i++ {
        if soal[i] < 301 {
            *skor++
            *totalWaktu += soal[i] // Tambahkan waktu soal yang
valid
        }
    }
}

func main() {
    var jumlahPeserta int
    fmt.Print("Masukkan jumlah peserta: ")
```

```
    fmt.Scan(&jumlahPeserta)

    namaPeserta := make([]string, jumlahPeserta)
    skorPeserta := make([]int, jumlahPeserta)
    totalWaktuPeserta := make([]int, jumlahPeserta)

    for i := 0; i < jumlahPeserta; i++ {
        fmt.Printf("Masukkan nama peserta %d: ", i+1)
        fmt.Scan(&namaPeserta[i])

        var soal [8]int
        fmt.Printf("Masukkan waktu penyelesaian 8 soal (dalam menit)
untuk %s: ", namaPeserta[i])
        for j := 0; j < 8; j++ {
            fmt.Scan(&soal[j])
        }

        // Panggil fungsi hitungSkor tanpa return, gunakan parameter
input-output
        hitungSkor(&soal, &skorPeserta[i], &totalWaktuPeserta[i])
    }

    // Menentukan pemenang
    pemenang := 0
    for i := 1; i < jumlahPeserta; i++ {
        if skorPeserta[i] > skorPeserta[pemenang] || (skorPeserta[i]
== skorPeserta[pemenang] && totalWaktuPeserta[i] <
totalWaktuPeserta[pemenang]) {
            pemenang = i
        }
    }

    // Menampilkan hasil pemenang
    fmt.Printf("Pemenangnya adalah %s dengan %d soal diselesaikan
dalam total waktu %d menit\n",
        namaPeserta[pemenang], skorPeserta[pemenang],
totalWaktuPeserta[pemenang])
}
```

Screenshot Output



Deskripsi Program

Program ini mencari pemenang kompetisi pemrograman berdasarkan jumlah soal yang diselesaikan dan total waktu yang dibutuhkan. Setiap peserta menyelesaikan 8 soal dengan waktu maksimal 301 menit per soal, dan hanya soal yang selesai dalam waktu kurang dari 301 menit yang dihitung sebagai soal yang terjawab. Fungsi hitungSkor menghitung jumlah soal yang diselesaikan dan menambahkan total waktu secara langsung tanpa mengembalikan nilai. Di program utama, data peserta seperti nama, waktu pengerjaan soal, dan skor dihitung dan disimpan. Pemenang adalah peserta yang menyelesaikan soal terbanyak, atau jika ada kesamaan jumlah soal, yang membutuhkan waktu paling singkat.

3. Unguided 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 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 **skiena** 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

```go
package main

import "fmt"

// Fungsi untuk mencetak 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("1\n") // mencetak 1 sebagai elemen terakhir
}
```

```go
func main() {
    // Contoh masukan
    var n int
    fmt.Print("Masukkan nilai awal: ")
    fmt.Scan(&n)
    if n > 0 && n < 1000000 {
        cetakDeret(n)
    } else {
        fmt.Println("Nilai harus antara 1 dan kurang dari
1.000.000")
    }
}
```

Screenshot Output

```
PS D:\Kuliah\Matkul\SEMESTER 3\ALPRO PRAKTEK\PRAKTIKUM\MODUL IV>
ded_3.go"
Masukkan nilai awal: 22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS D:\Kuliah\Matkul\SEMESTER 3\ALPRO PRAKTEK\PRAKTIKUM\MODUL IV>
```

Deskripsi Program

Program ini mencetak deret berdasarkan aturan yang dikenal sebagai "konjectur Collatz." Pengguna diminta untuk memasukkan nilai awal n, yang harus berada di antara 1 dan kurang dari 1.000.000. Jika nilai yang dimasukkan valid, program akan mencetak deret hingga mencapai angka 1. Deret tersebut dibentuk dengan membagi n dengan 2 jika n genap, atau mengalikan n dengan 3 dan menambah 1 jika n ganjil. Proses ini berlanjut sampai angka 1 dicetak sebagai elemen terakhir deret.