

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL IV
PROCEDURE**



Disusun Oleh :

Aji Noto Sutrisno (2311102262)

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

Fungsi prosedural adalah unit dari blok kode yang memiliki nama, dapat dipanggil kembali, dan dapat menerima argumen serta mengembalikan nilai. Dalam paradigma prosedural, kode dibagi menjadi fungsi-fungsi kecil yang memiliki tanggung jawab tertentu untuk menyelesaikan tugas yang spesifik. Fungsi ini akan dipanggil atau dieksekusi kapan saja saat dibutuhkan, sehingga kode menjadi modular, mudah dibaca, dan dipelihara.

Fungsi dalam Golang ditulis dengan menggunakan kata kunci `func`, diikuti dengan nama fungsi, daftar parameter (jika ada), tipe data pengembalian (jika ada), dan badan fungsi. fungsi tanpa pengembalian nilai dalam Golang disebut sebagai `void function` atau fungsi tanpa `return value`. Fungsi jenis ini digunakan untuk mengeksekusi suatu tindakan atau melakukan serangkaian instruksi tanpa harus mengembalikan nilai apapun. Dalam bahasa pemrograman lain, ini disebut sebagai fungsi "void", namun dalam Golang, fungsi tersebut cukup dideklarasikan tanpa tipe data pengembalian.

Dalam Go if yang sederhana itu bentuknya seperti ini:

```
func cetakPesan() {  
    fmt.Println("Halo, ini adalah fungsi tanpa pengembalian  
    nilai.") }  

```

Pada contoh di atas, fungsi `cetakPesan` tidak memiliki tipe pengembalian dan hanya menjalankan perintah untuk mencetak pesan ke layar. Struktur Fungsi Tanpa Pengembalian Nilai Berikut adalah struktur umum dari fungsi tanpa pengembalian nilai:

```
func namaFungsi(parameter1 tipeData1, parameter2 tipeData2)  
{ // perintah yang dijalankan oleh fungsi  
}  

```

Keuntungan Fungsi Tanpa Pengembalian Nilai

- Sederhana dan Fokus pada Aksi: Fungsi yang tidak perlu mengembalikan nilai menjadi lebih sederhana dan mudah dimengerti karena fungsinya hanya untuk melakukan aksi tertentu.

- Pemrosesan IO atau Tugas Sistem: Fungsi tanpa pengembalian sering digunakan dalam penanganan input/output, di mana tugas utama adalah berinteraksi dengan sistem, misalnya menulis ke file, mengirim data ke server, atau mencetak ke layar.
- Meningkatkan Keterbacaan: Ketika kita tidak perlu menerima informasi kembali dari suatu fungsi, kita bisa langsung memahami bahwa fungsinya adalah untuk menjalankan suatu proses.

II. GUIDED

1. Soal Studi Case

Sourcecode

```
package main

import "fmt"

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

func faktorial(n int, hasil *int) {
    //menggunakan pointer agar dapat mengembalikan nilai
    tanpa return
    *hasil = 1
    for i := 1; i <= n; i++ {
        *hasil *= i
    }
}

func permutasi(n, r int) {
    var hasiln, hasilnr int
    faktorial(n, &hasiln)
    faktorial(n-r, &hasilnr)
    fmt.Println("Hasil permutasi:", hasiln/hasilnr)
}
```

Output

```
PS D:\Pratikum Alpro 2\Laprak 4> go run "d:\Pratikum Alpro 2\Laprak 4\Guided\guided1.go"
5 10
Hasil permutasi: 30240
PS D:\Pratikum Alpro 2\Laprak 4> █
```

Deskripsi

Program ini dibuat untuk menghitung permutasi (a,b) jika memenuhi kondisi $a \geq b$ dan permutasi (b,a) jika tidak memenuhi kondisi pertama, Program ini merupakan lanjutan dari Guided1 pertemuan sebelumnya yang dimana di program saat ini merubah fungsi yang memiliki nilai return diubah ke sebuah procedural yang dimana fungsi tersebut menggunakan pointer untuk mengembalikan nilai hasil perhitungan fungsi faktorial, program ini memiliki cara kerja. *User* menginputkan a, dan b kemudian program akan menjalankan faktorial lalu

kemudian program akan menjalankan prosedural ke dua yaitu untuk mencari permutasi.

2. Soal Studi Case

Sourcecode

```
package main

import "fmt"

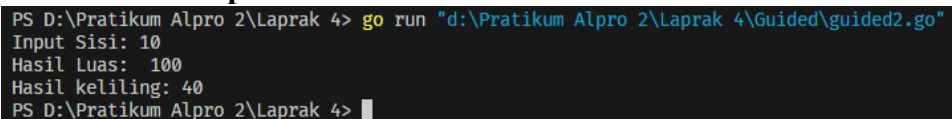
func main() {
    var s int
    fmt.Print("Input Sisi: ")
    fmt.Scan(&s)

    luasPersegi(s)
    kelilingPersegi(s)
}

func luasPersegi(s int) {
    hasil := s * s
    fmt.Println("Hasil Luas: ", hasil)
}

func kelilingPersegi(s int) {
    hasil := 4 * s
    fmt.Print("Hasil keliling: ", hasil)
}
```

Screenshoot Output



```
PS D:\Pratikum Alpro 2\Laprak 4> go run "d:\Pratikum Alpro 2\Laprak 4\Guided\guided2.go"
Input Sisi: 10
Hasil Luas: 100
Hasil keliling: 40
PS D:\Pratikum Alpro 2\Laprak 4>
```

Deskripsi Program

Program diatas dibuat untuk mencari hasil perhitungan luas dan keliling dari sebuah persegi. Memiliki beberapa prosedural yaitu `func luasPersegi(s int)` dan `func kelilingPersegi(s int)` Program ini memiliki cara kerja. *User* akan menginputkan (s) kemudian program akan menjalankan prosedural yang ada dan memberikan output sesuai hasil perhitungan

III. UNGUIDED

1. Soal Studi Case

Minggu Ini, mahasiswa Fakultas Informatika mendapatkan tugas dari mata kuliah matematika 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!

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

Sourcecode

```
package main

import "fmt"

var hasiln, hasilnr, hasilr int

func main() {
    var a, b, c, d int
    fmt.Print("Input nilai a, b, c, d : ")
    fmt.Scan(&a, &b, &c, &d)
    if a >= c && b >= d {
        permutasi(a, c)
        kombinasi(a, c)
        permutasi(b, d)
        kombinasi(b, d)
    } else {
        fmt.Print("Kondisi Tidak Terpenuhi")
    }
}

func faktorial(n int, hasil *int) {
    //menggunakan pointer agar dapat mengembalikan nilai
    tanpa return
    *hasil = 1
    for i := 1; i <= n; i++ {
        *hasil *= i
    }
}
```

```

    }
}

func permutasi(n, r int) {

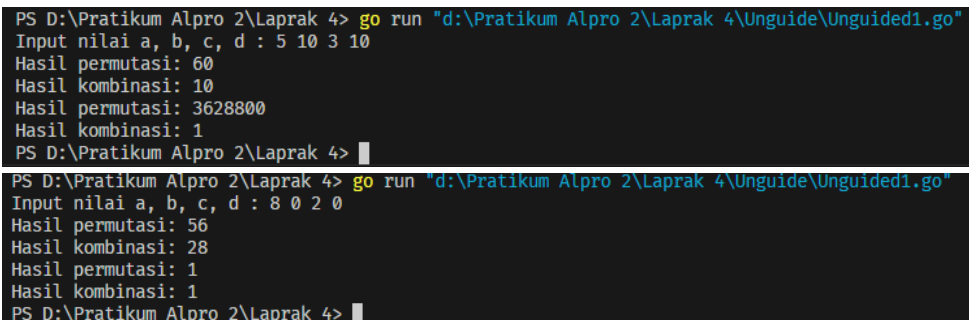
    faktorial(n, &hasiln)
    faktorial(n-r, &hasilnr)
    fmt.Println("Hasil permutasi:", hasiln/hasilnr)
}

func kombinasi(n, r int) {

    faktorial(n, &hasiln)
    faktorial(r, &hasilr)
    faktorial(n-r, &hasilnr)
    hasil := hasilr * hasilnr
    fmt.Println("Hasil kombinasi:", hasiln/hasil)
}
}

```

Screenshoot Output



```

PS D:\Pratikum Alpro 2\Laprak 4> go run "d:\Pratikum Alpro 2\Laprak 4\Unguide\Unguided1.go"
Input nilai a, b, c, d : 5 10 3 10
Hasil permutasi: 60
Hasil kombinasi: 10
Hasil permutasi: 3628800
Hasil kombinasi: 1
PS D:\Pratikum Alpro 2\Laprak 4>
PS D:\Pratikum Alpro 2\Laprak 4> go run "d:\Pratikum Alpro 2\Laprak 4\Unguide\Unguided1.go"
Input nilai a, b, c, d : 8 0 2 0
Hasil permutasi: 56
Hasil kombinasi: 28
Hasil permutasi: 1
Hasil kombinasi: 1
PS D:\Pratikum Alpro 2\Laprak 4>

```

Deskripsi Program

Program ini merupakan lanjutan dari guided 1, Program ini menghitung permutasi dan kombinasi dari dua pasang angka, yaitu (a, c) dan (b, d), berdasarkan input pengguna. Program akan memeriksa apakah nilai a lebih besar atau sama dengan c dan b lebih besar atau sama dengan d. Jika syarat ini terpenuhi, program akan menghitung dan menampilkan hasil permutasi dan kombinasi dari kedua pasangan tersebut. Jika tidak, program menampilkan pesan bahwa kondisi tidak terpenuhi.

Program diatas memiliki 3 prosedur yaitu

- `func faktorial(n int, hasil *int)` Digunakan untuk menghitung faktorial dari inputan user

- `func permutasi(n, r int)` Digunakan untuk menghitung permutasi dari dua inputan user
- `func kombinasi(n, r int)` Digunakan untuk menghitung kombinasi dari dua inputan user

Cara kerja program ini *user* akan menginputkan nilai 4 inputan (a, b , c, d) kemudian program akan menjalankan 2 prosedural diatas yaitu kombinasi dan permutasi, sedangkan prosedural faktorial digunakan pada prosedural permutasi dan kombinasi. Kemudian program akan menampilkan hasil dari keduanya.

2. Soal Studi Case

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

Keterangan:

Astuti menyelesaikan 6 soal dalam waktu 287 menit, sedangkan Bertha 7 soal dalam waktu 294 menit. Karena Bertha menyelesaikan lebih banyak, maka Bertha menang. Jika keduanya menyelesaikan sama banyak, aka pemenang adalah yang menyelesaikan dengan total waktu paling kecil.

Sourcecode

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strconv"
    "strings"
)

func main() {
    hitungNilai := make(map[string][]int)
    scanner := bufio.NewScanner(os.Stdin)

    // Mengambil input
    for {
        //membaca inputan dalam satu line
        scanner.Scan()
        input := scanner.Text()
        if strings.ToLower(input) == "selesai" {
            break
        }

        parts := strings.Fields(input)
        nama := parts[0]

        //:1 digunakan untuk menghitung panjang array
        dimulai dari index 1
        for _, part := range parts[1:] {
            skor, _ := strconv.Atoi(part)
            hitungNilai[nama] =
            append(hitungNilai[nama], skor)
        }
        hitungSkor(hitungNilai)
    }

    func hitungSkor(dataPeserta map[string][]int) {
```

```

var namaPemenang string
skorMaksimum := -1
skorMinimum := 0

for nama, skor := range dataPeserta {
    soalSelesai, totalWaktu := 0, 0

    // Hitung soal yang selesai dan total waktu
    for _, time := range skor {
        if time < 301 && time > 0 { // Cek waktu
            yang valid
            soalSelesai++
            totalWaktu += time
        }
    }

    // Tentukan pemenang berdasarkan soal maksimum
    dan total waktu
    if soalSelesai > skorMaksimum || (soalSelesai
    == skorMaksimum && totalWaktu < skorMinimum) {
        namaPemenang = nama
        skorMaksimum = soalSelesai
        skorMinimum = totalWaktu
    }
}

// Output hasil akhir
fmt.Printf("%s %d %d\n", namaPemenang, skorMaksimum,
skorMinimum)
}

```

Screenshoot Output

```

PS D:\Pratikum Alpro 2\Laprak 4> go run "d:\Pratikum Alpro 2\Laprak 4\Unguide\Unguided2.go"
Astuti 20 50 301 301 61 71 75 10
Bertha 25 47 301 26 50 60 65 21
Selesai
Bertha 7 294
PS D:\Pratikum Alpro 2\Laprak 4> 

```

Deskripsi Program

Program ini dibuat untuk menentukan juara dari sebuah lomba atau kompetisi dengan ketentuan 8 soal yang harus dapat diselesaikan dalam waktu 5 jam saja. Jika tidak berhasil atau tidak mengirimkan jawaban maka otomatis dianggap menyelesaikan dalam waktu 5 jam 1 menit (301 menit). Akan tetapi jika jumlah soal sama maka akan ditentukan dengan total waktu yang telah diselesaikan. Kemudian program akan menampilkan hasil dari perhitungan yang ada. Berikut alur program diatas

- `bufio.Scanner` untuk membaca input dari pengguna. Jadi *user* akan menginputkan satu baris, input akan dipisahkan dari spasi yang ada, dan input akan berakhir apabila jika *user* menginputkan “selesai”
- setelah menginputkan, input tersebut akan ditambahkan pada map yang tersedia pada program (`map[string][]int`), di mana nama peserta adalah kunci, dan daftar skornya adalah nilainya.
- Kemudian program akan menjalankan procedural perhitungan skor, yang dimana jika inputan diatas 301 atau sama dengan 301 maka soal tidak dianggap selesai. Total waktu diakumulasi dan dihitung totalnya sedangkan untuk Pemenang adalah peserta yang menyelesaikan soal terbanyak. Jika jumlah soal sama, peserta dengan total waktu terkecil menang.
- Output akan menampilkan nama peserta yang menang kemudian menampilkan total waktu yang diberikan dan menampilkan total soal yang diselesaikan

3. Soal Studi Case

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 $\frac{1}{2}n$, 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 cetak Deret 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.

Sourcecode

```
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

```

PS D:\Pratikum Alpro 2\Laprak 4> go run "d:\Pratikum Alpro 2\Laprak 4\Unguide\Unguided3.go"
Input (n): 22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS D:\Pratikum Alpro 2\Laprak 4>

```

Deskripsi Program

Program ini dibuat untuk menghitung sebuah deret dengan kondisi $n > 0 \ \&\& \ n < 1000000$ apabila memenuhi akan melakukan perulangan dengan kondisi jika $n \% 2$ atau bisa dibagi 2 maka $n = n/2$ dan jika tidak memenuhi maka $n = 3*n + 1$. Program ini dalam matematika biasa disebut baris Collatz. Proses ini dilakukan secara terus menerus hingga $n = 1$. Program ini memiliki cara kerja sebagai berikut, *user* menginputkan (n), kemudian n tersebut di proses dengan procedural func `printBaris(n int)` hingga $n=1$, dan setiap hasilnya akan menjadi output