

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 4
PROSEDUR**



Disusun Oleh :

Zahra Tsuroyya Poetri / 2311102127

IF 11 - 05

Dosen Pengampu :

Arif Amrulloh

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

1. Definisi Procedure

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, dan
2. **Tidak terdapat** kata kunci **return** dalam badan subprogram.

Kedudukannya prosedur sama seperti instruksi dasar yang sudah ada sebelumnya (**assignment**) dan/atau instruksi yang berasal dari paket (**fmt**), seperti **fmt.Scan** dan **fmt.Print**. Karena itu selalu pilih nama prosedur yang berbentuk kata kerja atau sesuatu yang merepresentasikan proses sebagai nama dari prosedur. Contoh: **cetak**, **hitungRerata**, **cariNilai**, **belok**, **mulai**,...

2. Deklarasi Procedure

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	}

Penulisan deklarasi ini berada di luar blok yang dari program utama atau **func main()** pada suatu program Go, dan bisa ditulis sebelum atau setelah dari blok program utama tersebut.

Contoh deklarasi prosedur mencetak n nilai pertama dari deret Fibonacci

	Notasi Algoritma
1	procedure cetakNFibo(in n : integer)
2	kamus
3	f1, f2, f3, i : integer
4	algoritma
5	f2 ← 0
6	f3 ← 1
7	for i ← 1 to n do
8	output(f3)
9	f1 ← f2
10	f2 ← f3
11	f3 ← f1 + f2
12	endfor
13	endprocedure
	Notasi dalam bahasa Go
14	func cetakNFibo(n int) {
15	var f1, f2, f3 int
16	f2 = 0
17	f3 = 1
18	for i := 1; i <= n; i++ {
19	fmt.Println(f3)
20	f1 = f2
21	f2 = f3
22	f3 = f1 + f2
23	}
24	}

Catatan: Kata kunci **in** pada contoh di atas akan dijelaskan pada materi parameter di modul 5 ini.

3. Cara Pemanggilan Procedure

Seperti yang sudah dijelaskan sebelumnya, **suatu prosedur hanya akan dieksekusi apabila dipanggil** baik secara langsung atau tidak langsung oleh program utama. Tidak langsung di sini maksudnya adalah prosedur dipanggil oleh program utama melalui perantara subprogram yang lain.

Pemanggilan suatu prosedur cukup mudah, yaitu dengan hanya **menuliskan nama beserta parameter atau argumen yang diminta dari suatu prosedur**. Sebagai contoh prosedur cetakNFibo di atas dipanggil dengan menuliskan namanya, kemudian sebuah variabel atau nilai integer tertentu sebagai argumen untuk parameter n. Contoh:

	Notasi Algoritma	
1	program contohprosedur	
2	kamus	
3	x : integer	
4	algoritma	
5	x ← 5	
6	cetakNFibo(x)	{cara pemanggilan #1}
7	cetakNFibo(100)	{cara pemanggilan #2}

8	endprogram	
	Notasi dalam bahasa Go	
9	func main() {	
10	var x int	
11	x = 5	
12	cetakNFibo(x)	{cara pemanggilan #1}
13	cetakNFibo(100)	{cara pemanggilan #2}
14	}	

Dari contoh di atas terlihat bahwa cara pemanggilan dengan notasi pseudocode dan Gotang adalah sama. Argumen yang digunakan untuk parameter n berupa Integer (sesuai deklarasi) yang terdapat pada suatu variabel (cara pemanggilan #1) atau nilainya secara langsung (cara pemanggilan #2).

4. Contoh Program dengan Procedure

Berikut ini adalah contoh penulisan prosedur pada suatu program lengkap.

Buatlah sebuah program beserta prosedur yang digunakan untuk menampilkan suatu pesan eror, warning atau informasi berdasarkan masukan dari user.

Masukan terdiri dari sebuah bilangan bulat flag (0 s.d. 2) dan sebuah string pesan M.

Keluaran berupa string pesan M beserta jenis pesannya, yaitu error, warning atau informasi berdasarkan nilai flag 0, 1 dan 2 secara berturut-turut.

```
1 package main
2 import "fmt"
3
4 func main(){
5     var bilangan int
6     var pesan string
7     fmt.Scan(&bilangan, &pesan)
8     cetakPesan(pesan,bilangan)
9 }
10
11 func cetakPesan(M string, flag int){
12     var jenis string = ""
13     if flag == 0 {
14         jenis = "error"
15     }else if flag == 1 {
16         jenis = "warning"
17     }else if flag == 2 {
18         jenis = "informasi"
19     }
20     fmt.Println(M,jenis)
21 }
```

Penulisan argumen pada parameter cetak Pesan(pesan bilangan) **harus sesuai urutan tipe data** pada **func** cetak Pesan(M **string** flag **int**, yaitu string kemudian integer.

5. Parameter

Suatu subprogram yang dipanggil dapat berkomunikasi dengan pemanggilnya melalui argumen yang diberikan melalui parameter yang dideklarasikan pada subprogramnya. Berikut ini jenis atau pembagian dari parameter.

Berdasarkan letak penulisannya pada program, maka parameter dapat dikelompokkan menjadi dua, yaitu parameter formal dan parameter aktual.

```
1 func volumeTabung(jari_jari,tinggi int) float64 {
2     var luasAlas,volume float64
3
4     luasAlas = 3.14 * float64(jari_jari * jari_jari)
5     volume = luasAlas * tinggi
6     return volume
7 }
8
9 func main() {
10     var r,t int
11     var v1,v2 float64
12     r = 5; t = 10
13     v1 = volumeTabung(r,t)
14     v2 = volumeTabung(r,t) + volumeTabung(15,t)
15     fmt.Println(volumeTabung(14,100))
16 }
```

1. Parameter Formal

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.

Sebagai contoh parameter **jari_jari**, **tinggi** pada deklarasi **fungsi volumeTabung** adalah parameter formal (teks berwarna merah). Artinya ketika memanggil volumeTabung maka kita harus mempersiapkan dua integer (berapapun nilainya) sebagai jari_jari dan tinggi.

2. Parameter Aktual

Sedangkan parameter aktual adalah argumen yang digunakan pada bagian parameter saat pemanggilan suatu subprogram. Banyaknya argumen dan tipe data yang terdapat pada parameter aktual harus mengikuti parameter formal.

Sebagai contoh argumen **r**, **t**, **15**, **14** dan **100** pada contoh kode di atas (teks berwarna biru) adalah parameter aktual, yang menyatakan nilai yang kita berikan sebagai jari-jari dan tinggi.

Selain Itu parameter Juga dikelompokkan berdasarkan alokasi memorinya, yaitu pass by value dan pass by reference.

1. Pass by Value

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.

Pada notasi pseudocode, secara semua parameter formal pada fysi adalah pass by value, sedangkan pada prosedur diberi kata kunci in pada saat penulisan parameter formal. Sedangkan pada bahasa pemrograman Go sama seperti fungsi pada pseudocode, tidak terdapat kata kunci khusus untuk parameter formal fungsi dan prosedur.

2. Pass by Reference (Pointer)

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 parameter formal tersebut akan berdampak pada parameter aktual Artinya nilai terakhirnya akan dapat diketahui oleh st pemanggil setelah subprogram tersebut selesai dieksekusi. Pass by reference sebaliknya digunakan hanya untuk prosedur.

Penulisan parameter pass by reference pada prosedur baik pseudocode dan Go menggunakan kata kunci atau identifier khusus. Pada pseudocode menggunakan kata kunci in/out, sedangkan pada bahasa Go diberi identifier asterik (*) sebelum tipe data di parameter formal yang menjadi pass by reference.

Catatan:

Parameter pada fungsi sebaiknya adalah pass by value, hal ini dikarenakan fungsi bisa mengembalikan (return) nilai ke pemanggil dan tidak memberikan efek langsung pada program, walaupun tidak menutup kemungkinan menggunakan pass by reference.

Penggunaan pass by reference sebaiknya pada prosedur karena prosedur tidak bisa mengembalikan nilai ke pemanggil. Dengan memanfaatkan pass by reference maka prosedur seolah-olah bisa mengirimkan nilai kepada si pemanggil.

Untuk lebih jelas perhatikan contoh sebuah subprogram yang digunakan untuk menghitung persamaan berikut ini:

$$f(x,y) = 2x - \frac{y}{2} + 3$$

Notasi Algoritma	
<pre> function f1(x,y : integer) → real kamus hasil : real algoritma hasil ← 2*x - 0.5*y + 3 return hasil endfunction procedure f2(in x,y : integer, in/out hasil:real) algoritma hasil ← 2*x - 0.5*y + 3 endprocedure program Contoh kamus a,b : integer c : real algoritma input(a,b) f2(a,b,c) output(c, f1(b,a)) endprogram </pre>	<p>x dan y pada fungsi f1 dan prosedur f2 adalah pass by value,</p> <p>sedangkan variabel hasil pada prosedur f2 adalah pass by reference.</p> <p>Untuk pemanggilan dengan notasi pseudocode masih sama dengan materi yang sudah dipelajari sebelumnya</p>
Notasi dalam bahasa Go	
<pre> package main import "fmt" func f1(x,y int) float64 { var hasil float64 hasil = float64(2*x) - 0.5*float64(y) + 3.0 return hasil } func f2(x,y int, hasil *float64) { *hasil = float64(2*x) - 0.5*float64(y) + 3.0 } func main(){ var a,b int; var c float64 fmt.Scan(&a,&b) f2(a,b,&c) output(c, f1(b,a)) } endprogram </pre>	<p>x dan y pada fungsi f1 dan prosedur f2 adalah pass by value,</p> <p>sedangkan variabel hasil pada prosedur f2 adalah pass by reference.</p> <p>Karena variabel hasil adalah pointer to float64, maka untuk mengaksesnya menggunakan simbol bintang (*) pada variabelnya.</p> <p>Pada bahasa Go saat pemanggilan prosedur f2, maka parameter aktual untuk pass by reference harus diberi ampersand "&", contohnya &c</p>

II. GUIDED

Guided 1

Sourcecode

```
package main

import "fmt"

func main() {
    var a, b int // Mendeklarasikan variabel integer a
dan b
    fmt.Print("Masukkan dua angka: ") // Mencetak pesan
untuk pengguna dapat memasukkan nilai a dan b
    fmt.Scan(&a, &b) // Membaca input dari pengguna
untuk dua nilai integer a dan b
    var hasil int // Mendeklarasikan variabel 'hasil'
bertipe integer

    if a >= b { // Sebuah kondisi jika a lebih besar
sama dengan b, maka:
        // Jika a lebih besar atau sama dengan b,
panggil fungsi permutasi dengan parameter (a, b)
        permutasi(a, b, &hasil)
    } else {
        // Jika b lebih besar dari a, panggil fungsi
permutasi dengan parameter (b, a)
        permutasi(b, a, &hasil)
    }

    fmt.Println("Hasil permutasi: ", hasil) // Mencetak
pesan hasil
}
```

```

func faktorial(n int, hasil *int) {
    *hasil = 1
    var i int
    // Loop untuk menghitung faktorial dari n
    for i = 1; i <= n; i++ {
        *hasil *= i // Mengalikan hasil dengan nilai
        i pada setiap iterasi
    }
}

func permutasi(n, r int, hasil *int) {
    // Menghitung permutasi nPr dengan membagi
    faktorial n dengan faktorial (n-r)
    var N, NR int
    faktorial(n, &N)
    faktorial(n-r, &NR)
    *hasil = N / NR
}

```

Screenshot Output

```

PS C:\berkas aya\kuliah\sem 3\Praktikum ALPRO 2\Modul 4> go run
Masukkan dua angka: 2 5
Hasil permutasi: 20
PS C:\berkas aya\kuliah\sem 3\Praktikum ALPRO 2\Modul 4>

```

Deskripsi Program

Program tersebut merupakan program menghitung permutasi dari dua angka yang dimasukkan oleh pengguna lalu disimpan dalam variabel `a` dan `b`. Kemudian program memeriksa angka mana yang lebih besar dari inputan pengguna, dan menghitung permutasi dengan menggunakan angka yang lebih besar sebagai `n` dan angka yang lebih kecil sebagai `r`. Fungsi

permutasi digunakan untuk menghitung permutasi dengan rumus $P(n,r) = \frac{n!}{(n-r)!}$. Fungsi faktorial menghitung hasil faktorial dengan mengalikan angka dari 1 sampai n . Setelah dihitung, hasil permutasi didapatkan dari hasil faktorial n dan faktorial $n-r$. Setelah itu hasil akan ditampilkan. Program ini menggunakan perulangan if-else untuk menghitung dua angka inputan pengguna.

Guided 2

Sourcecode

```
package main

import "fmt"

// Fungsi untuk menghitung luas persegi
func luas(sisi int, luasPersegi *int) {
    *luasPersegi = sisi * sisi // Menghitung luas
}

// Fungsi untuk menghitung keliling persegi
func keliling(sisi int, kelPersegi *int) {
    *kelPersegi = 4 * sisi // Menghitung keliling
}

func main() {
    var sisi int
    var hasilLuas int
    var hasilKeliling int

    fmt.Print("Masukkan Sisi Persegi: ")
    fmt.Scan(&sisi) // Membaca input dari pengguna

    // Memanggil fungsi untuk menghitung luas dan
```

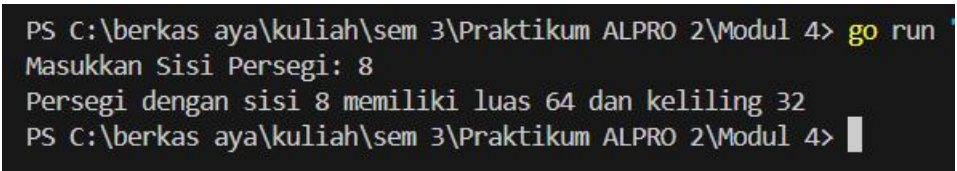
```

keliling
    luas(sisi, &hasilLuas)          // Mengisi hasilLuas
    keliling(sisi, &hasilKeliling) // Mengisi
hasilKeliling

    // Menampilkan hasil
    fmt.Println("Persegi dengan sisi", sisi, "memiliki
luas", hasilLuas, "dan keliling", hasilKeliling)
}

```

Screenshoot Output



```

PS C:\berkas aya\kuliah\sem 3\Praktikum ALPRO 2\Modul 4> go run
Masukkan Sisi Persegi: 8
Persegi dengan sisi 8 memiliki luas 64 dan keliling 32
PS C:\berkas aya\kuliah\sem 3\Praktikum ALPRO 2\Modul 4>

```

Deskripsi Program

Program di atas merupakan program untuk menghitung luas dan keliling persegi berdasarkan nilai sisi yang dimasukkan oleh pengguna lalu disimpan dalam variabel `sisi`. Kemudian, fungsi dipanggil dengan fungsi `luas` untuk menghitung luas persegi dan fungsi `keliling` untuk menghitung keliling persegi. Fungsi `luas` menerima dua parameter yaitu `sisi` dan pointer `luasPersegi` untuk menyimpan hasil luas dengan tipe data integer. Fungsi `keliling` juga menerima dua parameter, yaitu `sisi` dan pointer `kelPersegi` untuk menyimpan hasil keliling dengan tipe data integer. Fungsi `luas` menghitung luas dengan rumus $*luasPersegi = sisi * sisi$, sementara fungsi `keliling` menghitung keliling dengan rumus $*kelPersegi = 4 * sisi$. Setelah dihitung, program akan menampilkan hasil luas dan keliling persegi.

III. UNGUIDED

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 $b \geq$

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"

func main() {
```

```

        var a, b, c, d int // Mendeklarasikan variabel a,
b, c, d menggunakan integer
        fmt.Print("Masukkan nilai: ") // Mencetak untuk
inputan a, b, c, d
        fmt.Scan(&a, &b, &c, &d)

        var hasilPer, hasilKom int // Mendeklarasikan
variabel hasil permutasi dan hasil kombinasi menggunakan
integer

        // Menghitung permutasi dan kombinasi
        if a >= c && b>=d { // Jika a >= c dan b >= d,
maka hitung permutasi dan kombinasi

                // Permutasi untuk (a, c)
                permutation(a, c, &hasilPer)
                fmt.Println("Permutasi: ", hasilPer) // Cetak
hasil permutasi (a, c)

                // Kombinasi untuk (a, c)
                kombination(a, c, &hasilKom)
                fmt.Println("Kombinasi: ", hasilKom) // Cetak
hasil kombinasi (a, c)

                // Permutasi untuk (b, d)
                permutation(b, d, &hasilPer)
                fmt.Println("Permutasi: ", hasilPer) // Cetak
hasil permutasi (b, d)

                // Kombinasi untuk (b, d)
                kombination(b, d, &hasilKom)
                fmt.Println("Kombinasi: ", hasilKom) // Cetak
hasil kombinasi (b, d)

        } else { // Jika a >= c dan b >= d tidak terpenuhi,

```

```

makan akan mencetak pesan
        fmt.Println("Tidak memenuhi kondisi") //
Cetak pesan jika kondisi tidak terpenuhi
    }
}

// Prosedur untuk menghitung hasil faktorial
func factorial(n int, hasil *int){
    *hasil = 1 // Inisialisasi hasil sebagai 1 (karena
faktorial dimulai dari 1)
    for i := 1; i <= n; i++ {
        *hasil *= i
    }
}

// Prosedur untuk menghitung hasil permutasi dari n, n-r
func permutation(n, r int, hasil *int) {
    var N, NR int // Inisialisasi variabel untuk
menyimpan hasil faktorial n dan n-r
    factorial(n, &N) // Memanggil prosedur faktorial
untuk menghitung faktorial dari n
    factorial(n-r, &NR) // Memanggil prosedur faktorial
untuk menghitung faktorial dari n-r
    *hasil = N / NR // Menghitung permutasi
}

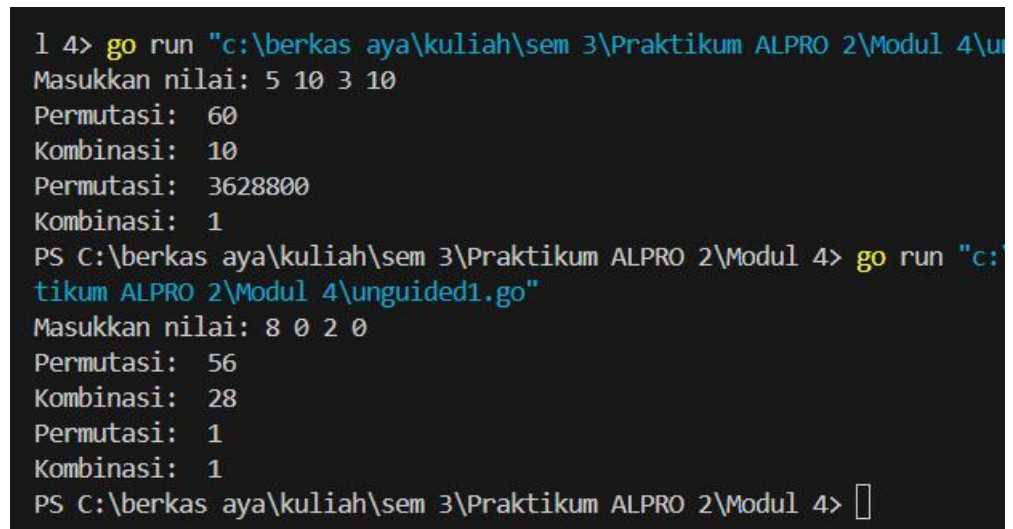
// Prosedur untuk menghitung hasil n, r, dan n-r
func kombination(n, r int, hasil *int) {
    var N, R, NR int // Inisialisasi variabel untuk
menyimpan hasil faktorial n, r, dan n-r
    factorial(n, &N) // Memanggil prosedur faktorial
untuk menghitung faktorial dari n
    factorial(r, &R) // Memanggil prosedur faktorial
untuk menghitung faktorial dari r

```



```
        factorial(n-r, &NR) // Memanggil prosedur faktorial
    untuk menghitung faktorial dari n-r
        *hasil = N / (R * NR) // Menghitung kombinasi
    }
```

Screenshoot Output



```
l 4> go run "c:\berkas aya\kuliah\sem 3\Praktikum ALPRO 2\Modul 4\un
Masukkan nilai: 5 10 3 10
Permutasi: 60
Kombinasi: 10
Permutasi: 3628800
Kombinasi: 1
PS C:\berkas aya\kuliah\sem 3\Praktikum ALPRO 2\Modul 4> go run "c:\u
tikum ALPRO 2\Modul 4\unguided1.go"
Masukkan nilai: 8 0 2 0
Permutasi: 56
Kombinasi: 28
Permutasi: 1
Kombinasi: 1
PS C:\berkas aya\kuliah\sem 3\Praktikum ALPRO 2\Modul 4> █
```

Deskripsi Program

Program di atas merupakan program untuk menghitung kombinasi dan permutasi dalam satu program yang sama. Program ini dideklarasikan dengan variabel **a**, **b**, **c**, **d** untuk pengguna dapat menginputkan angka yang diproses menggunakan `fmt.Print` dan `fmt.Scan`. Setelah pengguna memasukkan nilai program, program mengecek apakah $a \geq c$ dan $b \geq d$. Jika kondisi terpenuhi, program akan menghitung permutasi dan kombinasi menggunakan fungsi `permutasi(n, r)` dan `kombinasi(n, r)`. Fungsi `faktorial(n)` digunakan untuk menghitung faktorial dari angka yang dimasukkan. Hasil perhitungan permutasi dan kombinasi ditampilkan menggunakan `fmt.Println`. Jika kondisi tidak terpenuhi, program akan menampilkan pesan **“Tidak memenuhi kondisi”**.

Unguided 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 hitung Skor 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.

procedure 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, maka pemenang adalah yang menyelesaikan dengan total waktu paling kecil.

Sourcecode

```
package main

import (
    "fmt"
)

// Prosedur untuk menghitung jumlah soal yang
// diselesaikan dan total waktu (skor)
func hitungSkor(waktu []int, soal *int, skor *int) { //
// Mendefinisikan fungsi bernama hitungSkor yang menerima 3
// parameter
    *soal = 0 // Menginisialisasi nilai soal melalui
// pointer
    *skor = 0 // Menginisialisasi nilai skor melalui
// pointer
    for _, waktuSoal := range waktu { // Loop untuk
// setiap elemen dalam waktu. Elemen disimpan dalam variabel
// waktuSoal
        if waktuSoal <= 300 { // Memeriksa apakah
// waktusoal diselesaikan dalam waktu kurang dari atau sama
// dengan 300 menit
            *soal++
            *skor += waktuSoal
        }
    }
}
```

```
// Prosedur untuk mencari pemenang dari daftar peserta
func cariPemenang(namaPeserta []string, waktuTotal
[][]int, pemenang *string, hasilSoal *int, hasilSkor
*int) { // Mendefinisikan fungsi cariPemenang yang
menerima 5 parameter
    *pemenang = "" // inisialisasi variabel pemenang
dengan string
    *hasilSoal = -1 // Inisialisasi hasilSoal dengan -1
sebagai nilai awal
    *hasilSkor = 9999999 // Insialisasi hasilSkor dengan
nilai yang sabgat besar untuk memudahkan pencarian skor
yang lebih rendah nantinya

    for i, waktuSoal := range waktuTotal {
        var soal int
        var skor int

        // Memanggil prosedur hitungSkor untuk menghitung
soal dan skor peserta
        hitungSkor(waktuSoal, &soal, &skor)

        // Mengecek apakah peserta ini menyelesaikan
lebih banyak soal atau memiliki skor lebih kecil
        if soal > *hasilSoal || (soal == *hasilSoal &&
skor < *hasilSkor) {
            *pemenang = namaPeserta[i]
            *hasilSoal = soal
            *hasilSkor = skor
        }
    }
}

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

```

    fmt.Scan(&n) // Input jumlah peserta

    namaPeserta := make([]string, n)
    waktuTotal := make([][]int, n)

    for i := 0; i < n; i++ {
        // Input nama peserta dan waktu pengerjaan soal
        var nama string
        waktu := make([]int, 8)
        fmt.Printf("Masukkan: ")
        fmt.Scan(&nama) // Membaca nama peserta
        for j := 0; j < 8; j++ {
            fmt.Scan(&waktu[j]) // Membaca waktu
            pengerjaan soal
        }
        namaPeserta[i] = nama
        waktuTotal[i] = waktu
    }

    fmt.Println("Selesai")

    var pemenang string
    var hasilSoal int
    var hasilSkor int

    // Memanggil prosedur untuk mencari pemenang
    cariPemenang(namaPeserta, waktuTotal, &pemenang,
&hasilSoal, &hasilSkor)

    // Menampilkan hasil
    fmt.Printf("Pemenang adalah %s yang menyelesaikan %d
soal dengan waktu %d menit", pemenang, hasilSoal,
hasilSkor)
}

```

Screenshot Output

```
PS C:\berkas aya\kuliah\sem 3\Praktikum ALPRO 2\Modul 4> go run "c:\berkas
aktikum ALPRO 2\Modul 4\unguided2.go"
Masukkan jumlah peserta: 2
Masukkan: Astuti 20 50 301 301 61 71 75 10
Masukkan: Bertha 25 47 301 26 50 60 65 21
Selesai
Pemenang adalah Bertha yang menyelesaikan 7 soal dengan waktu 294 menit
PS C:\berkas aya\kuliah\sem 3\Praktikum ALPRO 2\Modul 4> █
```

Deskripsi Program

Program di atas merupakan program yang digunakan untuk menentukan pemenang dari kompetisi pemrograman tingkat nasional berdasarkan waktu yang digunakan dalam menyelesaikan soal. Pertama, program meminta inputan jumlah peserta, diikuti dengan nama peserta dan waktu yang dihabiskan setiap soal dari 8 soal. Data peserta dan waktu soal disimpan dalam array `namaPeserta[]` dan array dua dimensi `waktuTotal[][]`. Kemudian, fungsi `hitungSkor` digunakan untuk menghitung jumlah soal yang diselesaikan dalam waktu kurang dari sama dengan 300 menit dan menghitung total waktu yang digunakan. Fungsi `hitungSkor` digunakan untuk menghitung jumlah soal yang diselesaikan dan total waktu yang digunakan. Terdapat array waktu dan dua parameter, yaitu `soal` dan `skor` dengan tipe data integer pada fungsi ini. Setelah semua dihitung, fungsi `cariPemenang` membandingkan jumlah soal yang diselesaikan oleh setiap peserta. Jika dua peserta menyelesaikan jumlah soal yang sama, pemenang akan ditentukan berdasarkan waktu yang dihabiskan lebih sedikit. Fungsi `cariPemenang` menggunakan tiga parameter pointer, yaitu `*pemenang`, `*hasilSoal`, `*hasilSkor`. Setelah semua diproses, program akan mencetak menggunakan nama pemenang dengan jumlah soal yang diselesaikan dan total waktu yang dihabiskan menggunakan `fmt.Printf`.

Unguided 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 $1/2n$, 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. Pencetakkan deret harus dibuat dalam prosedur cetakDeret yang mempunyai 1 parameter formal, yaitu nilai dari suku awal.

prosedure cetak Deret(inn: integer)

Masukan berupa satu bilangan integer positif yang lebih kecil dari 1000000.

Keluaran terdiri dari satu baris saja. Setiap suku dari deret tersebut dicetali 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

Sourcecode

```
package main

import (
    "fmt"
)

// Prosedur untuk mencetak deret bilangan
func cetakDeret(n int) {
    fmt.Print(n) // Mencetak suku pertama (nilai awal)

    for n != 1{ // Melanjutkan mencetak hingga suku
        bernilai 1
        if n%2 == 0 { // Jika n adalah bilangan
            genap, suku berikutnya adalah n dibagi 2
            n = n / 2
        } else { // Jika n adalah bilangan ganjil,
            suku berikutnya adalah dikalikan 3 lalu tambahkan 1
            n = 3*n + 1
        }
        fmt.Print(" ", n) // Mencetak suku berikutnya
        sampai suku terakhir bernilai 1
    }
    fmt.Print()
}

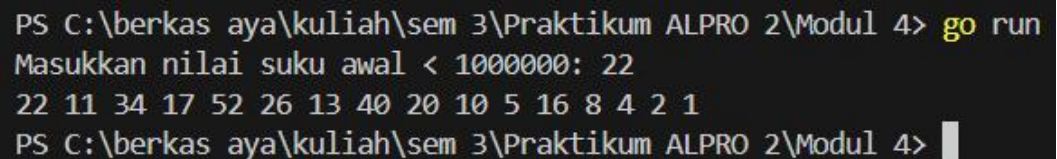
func main() {
    var n int
    fmt.Print("Masukkan nilai suku awal < 1000000: ")
    // Pengguna menginputkan nilai suku awal yang lebih kecil
    dari 1000000
    fmt.Scan(&n)

    if n >= 1 && n < 1000000 { // Menghitung atau
        memastikan nilai inputan valid
    }
```



```
        cetakDeret(n) // Memanggil untuk mencetak
deret menggunakan prosedur
    } else {
        fmt.Println("Masukkan harus berupa bilangan
bulat positif < 1000000.") // Tampilan pesan jika inputan
tidak valid
    }
}
```

Screenshoot Output



```
PS C:\berkas aya\kuliah\sem 3\Praktikum ALPRO 2\Modul 4> go run
Masukkan nilai suku awal < 1000000: 22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS C:\berkas aya\kuliah\sem 3\Praktikum ALPRO 2\Modul 4>
```

Deskripsi Program

Program di atas merupakan program yang dibuat untuk mencetak sebuah deret bilangan. Program dimulai dengan fungsi `cetakDeret` yang memiliki parameter `n` dengan tipe data integer. Parameter `n` tersebut merupakan nilai awal yang dimasukkan. Jika bilangan `n` yang diinputkan adalah genap maka, suku berikutnya diperoleh dengan cara membagi bilangan `n` dengan 2. Namun, jika bilangan `n` yang diinputkan adalah ganjil, maka suku berikutnya diperoleh dengan cara mengalikan bilangan `n` dengan 3 lalu ditambahkan 1. Rumus tersebut akan terus menerus mencetak bilangan sampai suku terakhir bernilai 1. Pengguna akan diminta untuk menginputkan sebuah bilangan bulat positif yang lebih kecil dari 1000000. Program kemudian akan mengecek apakah bilangan tersebut valid atau tidak. Jika nilai input valid, prosedur `cetakDeret` akan mencetak bilangan pertama hingga bernilai 1 barulah akan berakhir. Jika input tidak sesuai atau tidak memenuhi kondisi, maka program akan menampilkan pesan kesalahan yang dicetak oleh `fmt.Println`.

Daftar Pustaka

- [1] Susilowati, A. Zahara, A. (2024). *Modul 4 Praktikum Algoritma Pemrograman 2*. Program Studi Teknik Informatika, Telkom University Purwokerto.