

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL IV
PROSEDUR**



Disusun Oleh :

Nia Novela Ariandini / 2311102057

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

A. Definisi Procedur

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.

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,...

B. Deklarasi Procedur

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.

C. 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 dimintadari 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 golang adalah sama. Argumen yang digunakan untuk parameter n berupa integer (sesuai deklarasi) yang terdapat pada suatu variable (carapemanggilan#1) / nilainya secara langsung (carapemanggilan #2).

D. 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 error, 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 (Mstring flag int, yaitu string kemudian integer).

E. 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 }
```

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

- 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 volume tabung adalah parameter formal (teks berwarna merah). Artinya ketika memanggil volume Tabung 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 byreference. 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.

II. GUIDED

1. GUIDED 1

Studi Case : Buatlah sebuah program beserta prosedur yang digunakan untuk menghitung nilai faktorial dan permutasi. Masukan terdiri dari dua buah bilangan positif a dan b . Keluaran berupa sebuah bilangan bulat yang menyatakan nilai a permutasi b apabila $a \geq b$ atau b permutasi a untuk kemungkinan yang lain.

Sourcecode

```
package main

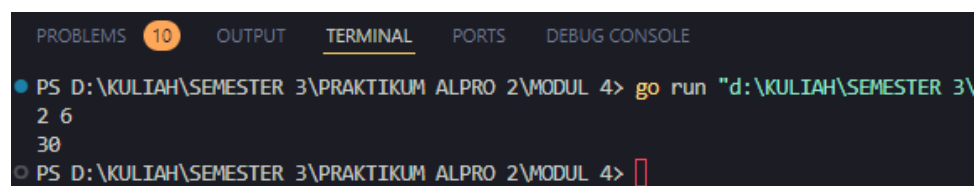
import "fmt"

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

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

func permutasi(n, r int, result *int) {
    var faktN, faktNR int
    faktorial(n, &faktN)
    faktorial(n-r, &faktNR)
    *result = faktN / faktNR
}
```

Screenshoot Output



```
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4> go run "d:\KULIAH\SEMESTER 3\
2 6
30
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4>
```

Deskripsi Program

Program ini digunakan untuk menghitung permutasi $P(n, r)$, yaitu jumlah cara menyusun r elemen dari n elemen tanpa pengulangan. Ini mengambil dua bilangan bulat a dan b , membandingkan nilai keduanya, dan kemudian menghitung permutasi berdasarkan nilai yang lebih besar. Program ini menggunakan dua fungsi utama : faktorial untuk menghitung faktorial suatu bilangan dan permutasi untuk menghitung nilai permutasi, yang dihitung berdasarkan rumus.

Algoritma dan Cara Program Berfungsi : Program dimulai dengan membaca dua nilai dari input. Kemudian, program membandingkan nilai a dan b untuk menentukan pasangan n dan r yang digunakan dalam perhitungan permutasi. Setelah menentukan n dan r , program memanggil fungsi permutasi, yang pertama-tama memanggil fungsi faktorial untuk menghitung faktorial $n!$ dan $(n-r)!$. Setelah nilai faktorial tersebut dihitung, hasil permutasi dihitung dengan membagi $n!$ dengan $(n-r)!$, dan hasilnya dicetak.

2. GUIDED 2

Studi Case : Buatlah program beserta prosedur yang digunakan untuk menghitung luas dan keliling persegi. Masukan terdiri dari sisi persegi. Keluaran berupa hasil luas dan keliling persegi.

Sourcecode

```
package main

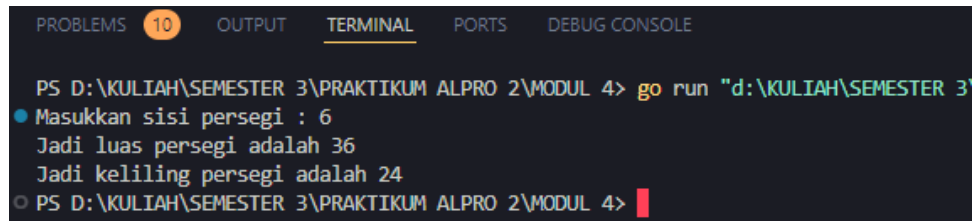
import "fmt"

func luas_persegi(s int) {
    luas := s * s
    fmt.Println("Jadi luas persegi adalah", luas)
}

func keliling_persegi(s int) {
    keliling := 4 * s
    fmt.Println("Jadi keliling persegi adalah", keliling)
}

func main() {
    var s int
    fmt.Print("Masukkan sisi persegi : ")
    fmt.Scan(&s)
    luas_persegi(s)
    keliling_persegi(s)
}
```


Screenshoot Output



```
PROBLEMS 10 OUTPUT TERMINAL PORTS DEBUG CONSOLE
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4> go run "d:\KULIAH\SEMESTER 3\
Masukkan sisi persegi : 6
Jadi luas persegi adalah 36
Jadi keliling persegi adalah 24
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4>
```

Deskripsi Program

Program ini menghitung luas persegi dan keliling persegi berdasarkan panjang sisi yang dimasukkan oleh pengguna. Fungsi `luas_persegi` dan `keliling_persegi` digunakan untuk menghitung kedua luas dan keliling persegi, dan setelah pengguna memasukkan nilai sisi, program akan menghitung dan menampilkan hasil dari kedua perhitungan tersebut.

Algoritma dan Cara Program Berfungsi : Program dimulai dengan meminta input dari pengguna untuk memasukkan panjang sisi persegi. Setelah nilai sisi diinputkan, program memanggil fungsi `luas_persegi` untuk menghitung luas persegi dengan mengalikan panjang sisi dengan dirinya sendiri, lalu mencetak hasilnya. Kemudian, program memanggil fungsi `keliling_persegi` untuk menghitung keliling persegi dengan mengalikan panjang sisi dengan 4, dan mencetak hasilnya.

III. UNGUIDED

1. UNGUIDED 1

Studi Case : Minggu ini, mahasiswa Fakultas Informatika mendapatkan tugas dari mata kuliah matematika diskrit untuk mempelajari kombinasi dan permutasi. Jonas adalah salah seorang mahasiswa, iseng untuk mengimplementasikannya kedalam suatu program. Oleh karena itu bersediakah kalian membantu Jonas? Masukan terdiri dari empat buah bilangan asli a, b, c , dan d yang dipisahkan oleh spasi, dengan syarat $a \geq c$ dan $b \geq c$. 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"
)

//untuk menghitung faktorial dan menyimpannya dalam
pointer hasil
func hitungFaktorial(n int, hasil *int) {
    *hasil = 1
    for i := 1; i <= n; i++ {
        *hasil *= i
    }
}

//untuk menghitung dan mencetak permutasi
func cetakPermutasi(n, r int) {
    if n < r {
        fmt.Println("Permutasi tidak dapat dihitung
karena n < r.")
    } else {
        var faktN, faktNR int
        hitungFaktorial(n, &faktN)
        hitungFaktorial(n-r, &faktNR)
        perm := faktN / faktNR
        fmt.Printf("Permutasi (%dP%d) = %d\n", n,
r, perm)
    }
}

//untuk menghitung dan mencetak kombinasi
func cetakKombinasi(n, r int) {
    if n < r {
        fmt.Println("Kombinasi tidak dapat dihitung
karena n < r.")
    } else {
        var faktN, faktR, faktNR int
        hitungFaktorial(n, &faktN)
        hitungFaktorial(r, &faktR)
        hitungFaktorial(n-r, &faktNR)
        komb := faktN / (faktR * faktNR)
        fmt.Printf("Kombinasi (%dC%d) = %d\n", n,
r, komb)
    }
}

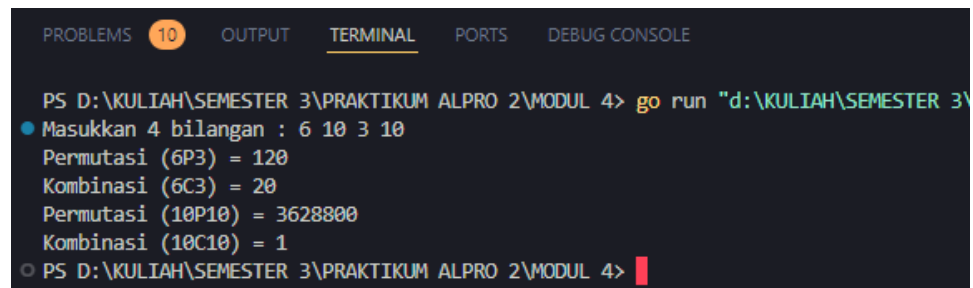
func main() {
    //menginputkan empat bilangan : a, b, c, d
    var a, b, c, d int
    fmt.Print("Masukkan 4 bilangan : ")
}
```

```

fmt.Scan(&a, &b, &c, &d)
//baris pertama : permutasi dan kombinasi a terhadap c
cetakPermutasi(a, c)
cetakKombinasi(a, c)
//baris kedua : permutasi dan kombinasi b terhadap d
cetakPermutasi(b, d)
cetakKombinasi(b, d)
}

```

Screenshoot Output



```

PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4> go run "d:\KULIAH\SEMESTER 3\
Masukkan 4 bilangan : 6 10 3 10
Permutasi (6P3) = 120
Kombinasi (6C3) = 20
Permutasi (10P10) = 3628800
Kombinasi (10C10) = 1
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4>

```

Deskripsi Program

Program ini dapat menghitung dan mencetak permutasi serta kombinasi dari dua pasangan bilangan yang dimasukkan oleh pengguna. Ini memiliki dua fungsi utama, yaitu cetak permutasi dan cetak kombinasi. Program ini juga menghitung permutasi $P(n, r)$ dan kombinasi $C(n, r)$. Jika nilai n lebih kecil dari r , program akan menampilkan pesan kesalahan karena permutasi dan kombinasi.

Algoritma dan Cara Program Berfungsi : Program dimulai dengan meminta pengguna memasukkan empat bilangan. Setelah nilai bilangan dimasukkan, program memanggil fungsi cetakPermutasi dan cetakKombinasi untuk menghitung permutasi dan kombinasi dari pasangan a dengan c, dan kemudian dari pasangan b dengan d. Jika nilai n lebih kecil dari r , program akan menampilkan pesan bahwa permutasi atau kombinasi tidak dapat dihitung. Hasil akhir perhitungan dicetak ke layar untuk setiap operasi yang valid.

2. UNGUIDED 2

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 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. Prosedur 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"
    "strings"
)

const maxTime = 301 //waktu max kalo soal tidak selesai

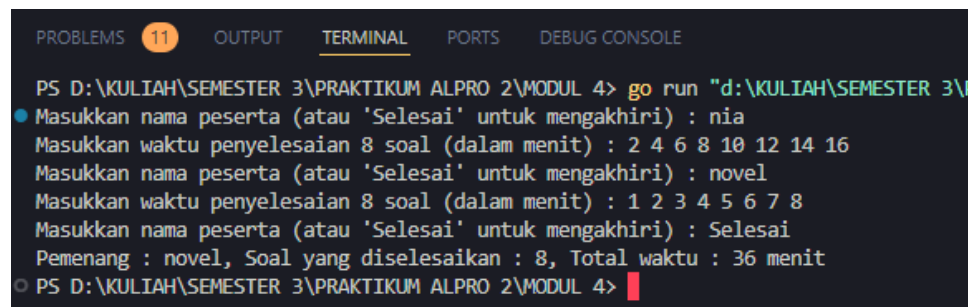
//untuk menghitung skor (jumlah soal yang diselesaikan dan total waktu)
func hitungSkor(waktu [8]int, soal *int, skor *int) {
    *soal = 0
    *skor = 0
    for _, waktuSoal := range waktu {
        if waktuSoal < maxTime { //kalo soal diselesaikan
            *soal++
            *skor += waktuSoal
        }
    }
}
```

```

func main() {
    var pemenang string
    var soalPemenang, skorPemenang int
    soalPemenang = 0
    skorPemenang = maxTime * 8 //nilai awal skor pemenang
    maksimal
    for {
        var nama string
        var waktu [8]int
        //membaca input nama peserta
        fmt.Print("Masukkan nama peserta (atau
'Selesai' untuk mengakhiri) : ")
        fmt.Scan(&nama)
        nama = strings.TrimSpace(nama)
        //kalo input adalah 'Selesai', hentikan loop
        if nama == "Selesai" {
            break
        }
        //membaca waktu penyelesaian 8 soal
        fmt.Print("Masukkan waktu penyelesaian 8 soal
(dalam menit) : ")
        for i := 0; i < 8; i++ {
            fmt.Scan(&waktu[i])
        }
        //hitung soal yang diselesaikan dan total waktu
        var soal, skor int
        hitungSkor(waktu, &soal, &skor)
        //tentukan pemenang berdasarkan jumlah soal dan
total waktu
        if soal > soalPemenang || (soal == soalPemenang
&& skor < skorPemenang) {
            pemenang = nama
            soalPemenang = soal
            skorPemenang = skor
        }
    }
    //cetak pemenang
    fmt.Printf("Pemenang : %s, Soal yang diselesaikan :
%d, Total waktu : %d menit\n", pemenang, soalPemenang,
skorPemenang)
}

```

Screenshoot Output



```
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4> go run "d:\KULIAH\SEMESTER 3\
Masukkan nama peserta (atau 'Selesai' untuk mengakhiri) : nia
Masukkan waktu penyelesaian 8 soal (dalam menit) : 2 4 6 8 10 12 14 16
Masukkan nama peserta (atau 'Selesai' untuk mengakhiri) : novel
Masukkan waktu penyelesaian 8 soal (dalam menit) : 1 2 3 4 5 6 7 8
Masukkan nama peserta (atau 'Selesai' untuk mengakhiri) : Selesai
Pemenang : novel, Soal yang diselesaikan : 8, Total waktu : 36 menit
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4>
```

Deskripsi Program

Pemenang kompetisi pemrograman ditentukan oleh program ini berdasarkan jumlah soal yang diselesaikan dan waktu total yang dihabiskan. Setiap peserta mengerjakan delapan soal, dan jika mereka memiliki waktu kurang dari 301 menit untuk menyelesaikannya, soal tersebut dianggap selesai. Sampai pengguna mengklik "Selesai", program akan terus menerima input peserta. Setelah itu, pemenang yang menyelesaikan soal terbanyak dengan waktu penyelesaian tercepat akan diumumkan.

Algoritma dan Cara Program Berfungsi : Program akan meminta input nama peserta dan waktu penyelesaian untuk delapan soal. Fungsi hitungSkor kemudian menghitung berapa soal yang selesai dan berapa total waktu yang digunakan. Setelah setiap peserta, program membandingkan hasil tersebut dengan pemenang sementara berdasarkan jumlah soal dan waktu. Proses ini terus berlangsung hingga pengguna memasukkan "Selesai", dan program akan mencetak nama pemenang, jumlah soal yang diselesaikan, dan total waktu yang dibutuhkan.

3. UNGUIDED 3

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. Pencetakan deret harus dibuat dalam prosedur cetakDeret yang

mempunyai 1 parameter formal, yaitu nilai dari suku awal. prosedur 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

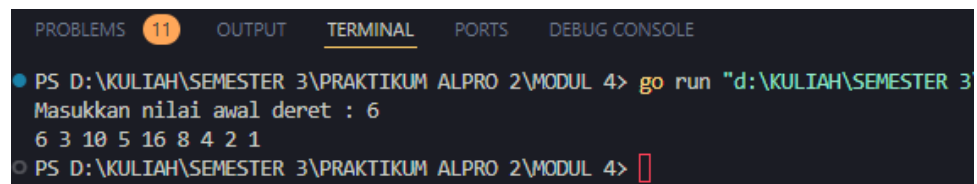
```
package main

import "fmt"

//untuk mencetak deret bilangan
func cetakDeret(n int) {
    for n != 1 {
        fmt.Printf("%d ", n) //cetak nilai n
        if n%2 == 0 {          //kalo n genap
            n = n / 2
        } else { //kalo n ganjil
            n = 3*n + 1
        }
    }
    fmt.Println(1) //cetak nilai akhir 1
}

func main() {
    var n int
    fmt.Print("Masukkan nilai awal deret : ")
    fmt.Scan(&n) //ambil input dari pengguna
    cetakDeret(n) //panggil prosedur cetakDeret
}
```

Screenshoot Output



```
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4> go run "d:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4"
Masukkan nilai awal deret : 6
6 3 10 5 16 8 4 2 1
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4>
```

Deskripsi Program

Program ini mencetak deret bilangan berdasarkan aturan Collatz (atau disebut juga *Collatz Conjecture*). Deret dimulai dari sebuah bilangan bulat positif yang diinput oleh pengguna. Jika bilangan tersebut genap, maka dibagi dua; jika ganjil, maka dikalikan tiga dan ditambah satu. Proses ini berlanjut hingga mencapai angka 1, dan deret tersebut ditampilkan ke layar.

Algoritma dan Cara Program Berfungsi : Program meminta pengguna memasukkan sebuah bilangan sebagai nilai awal. Fungsi cetakDeret kemudian menjalankan logika untuk menentukan apakah bilangan tersebut genap atau ganjil, dan memperbarui nilai n berdasarkan aturan Collatz. Nilai n akan terus diproses dan dicetak hingga menjadi 1, di mana proses berhenti dan angka 1 dicetak sebagai nilai akhir deret.

IV. KESIMPULAN

Prosedur dalam pemrograman merupakan bagian terpisah dari program yang dirancang untuk menjalankan tugas tertentu. Ada dua jenis prosedur, yaitu subrutin dan fungsi. Subrutin digunakan untuk melaksanakan tugas tanpa mengembalikan nilai, sedangkan fungsi menghasilkan nilai yang dikembalikan ke program utama. Suatu prosedur disebut prosedur jika memenuhi beberapa syarat, seperti membentuk instruksi baru dan tidak memiliki tipe data atau pernyataan "return." Parameter dalam prosedur dibagi menjadi dua: parameter aktual, yaitu variabel atau ekspresi saat pemanggilan, dan parameter formal, yang didefinisikan dalam daftar parameter prosedur. Selain itu, parameter dapat dikelompokkan berdasarkan alokasi memorinya menjadi pass by value, di mana nilai dikirim tanpa bisa diubah, dan pass by reference, di mana alamat memori variabel diproses dan dapat diubah. Pemahaman ini membantu programmer merancang program yang lebih modular dan efisien.

V. REFERENSI

- [1] Modul IV Praktikum Algoritma dan Pemrograman 2
- [2] <https://appmaster.io/id/blog/dasar-dasar-bahasa-pemrograman-go>
- [3] <https://www.petanikode.com/go-untuk-pemula/>