

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL IV

Procedure



Disusun Oleh :

AFRIZAL DWI NUGRAHA / 2311102136

S1 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 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.

4.2 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.

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

II. GUIDED

1. Contoh Program dengan Function

Sourcecode

```
package main

import "fmt"

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

func faktorian(n int) int {
    var hasil int = 1
    var i int
    for i = 1; i <= n; i++ {
        hasil = hasil * i
    }
}
```

```

        return hasil
    }

    func permutasi(n, r int, result *int) {
        *result = faktorial(n) / faktorial(n-r)
    }

```

Screenshoot Output

```

PS C:\Users\PC BRANDED\Documents\AfriZalVS\new\Semester03> go run
go"
4 3
24
PS C:\Users\PC BRANDED\Documents\AfriZalVS\new\Semester03> 

```

Deskripsi Program

Program ini bertujuan untuk menghitung permutasi antara dua bilangan bulat n dan r. Permutasi adalah proses mengatur r elemen secara terpisah dari n elemen total, dengan urutan yang diperhitungkan.

2. Menghitung Luas dan Keliling Persegi

Sourcecode

```

package main

import "fmt"

func luas(s int, hasil *int) {
    *hasil = s * s
}

func keliling(s int, hasil *int) {
    *hasil = s * 4
}

func main() {
    var s, luasHasil, kelilingHasil int
    fmt.Print("sisi: ")
    fmt.Scan(&s)
}

```

```
luas(s, &luasHasil)
keliling(s, &kelilingHasil)

fmt.Print(" Luas: ", luasHasil)
fmt.Print(" Keliling: ", kelilingHasil)
}
```

Screenshoot Output

```
PS C:\Users\PC BRANDED\Documents\AfrizalVS\new\Semester03> go run
go"
sisi: 6
Luas: 36 Keliling: 24
PS C:\Users\PC BRANDED\Documents\AfrizalVS\new\Semester03> |
```

Deskripsi Program

Program ini bertujuan untuk menghitung luas dan keliling persegi berdasarkan nilai sisi yang diberikan pengguna. Karena persegi adalah bangun datar dengan empat sisi panjang yang sama, rumus untuk menghitung luas dan kelilingnya cukup sederhana:

Luas persegi = $s \times s$, di mana panjang sisi s adalah s .

Keliling persegi = $4 \times s$, di mana panjang sisi s adalah s .

III. UNGUIDED

1. 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!

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

Selesaikan program tersebut dengan memanfaatkan prosedur yang diberikan berikut ini!

```
procedure factorial(in n: integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n
 F.S. hasil berisi nilai faktorial dari n}

procedure permutation(in n,r : integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n dan r, dan n >= r
 F.S. hasil berisi nilai dari n permutasi r}

procedure combination(in n,r : integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n dan r, dan n >= r
 F.S. hasil berisi nilai dari n kombinasi r}
```

Sourcecode

```
package main

import "fmt"

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

```

}

// Procedure untuk menghitung permutasi P(n, r)
func permutation(n, r int, hasil *int) {
    var fn, fnr int
    factorial(n, &fn)
    factorial(n-r, &fnr)
    *hasil = fn / fnr
}

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

func main() {
    var a, b, c, d int
    var permAC, combAC, permBD, combBD int

    fmt.Println("Masukkan nilai a, b, c, dan d (a >= c, b >= d):")
    fmt.Scan(&a, &b, &c, &d)

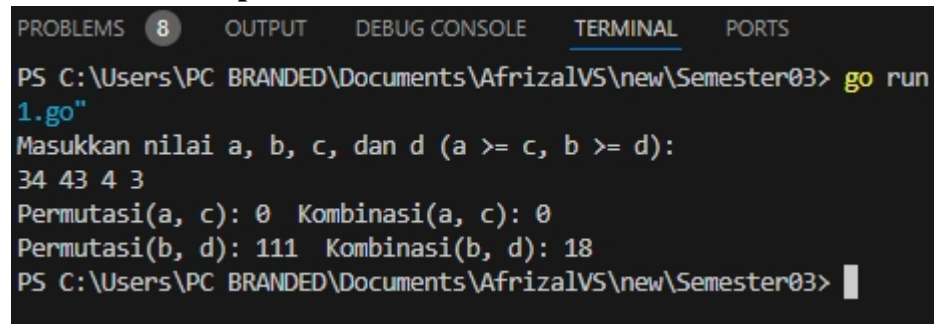
    // Hitung permutasi dan kombinasi untuk (a, c)
    permutation(a, c, &permAC)
    combination(a, c, &combAC)

    // Hitung permutasi dan kombinasi untuk (b, d)
    permutation(b, d, &permBD)
    combination(b, d, &combBD)

    fmt.Println("Permutasi (a, c):", permAC, " Kombinasi (a, c):",
combAC)
    fmt.Println("Permutasi (b, d):", permBD, " Kombinasi (b, d):",
combBD)
}

```

Screenshoot Output

A screenshot of a terminal window showing the execution of a Go program. The terminal has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The current tab is TERMINAL. The prompt is PS C:\Users\PC BRANDED\Documents\AfrizalVS\new\Semester03>. The user enters 'go run 1.go'. The program prompts 'Masukkan nilai a, b, c, dan d (a >= c, b >= d):'. The user enters '34 43 4 3'. The program outputs 'Permutasi(a, c): 0 Kombinasi(a, c): 0' and 'Permutasi(b, d): 111 Kombinasi(b, d): 18'. The prompt returns to PS C:\Users\PC BRANDED\Documents\AfrizalVS\new\Semester03>.

```
PS C:\Users\PC BRANDED\Documents\AfrizalVS\new\Semester03> go run 1.go
Masukkan nilai a, b, c, dan d (a >= c, b >= d):
34 43 4 3
Permutasi(a, c): 0 Kombinasi(a, c): 0
Permutasi(b, d): 111 Kombinasi(b, d): 18
PS C:\Users\PC BRANDED\Documents\AfrizalVS\new\Semester03>
```

Deskripsi Program

Program ini menggunakan prosedur untuk menghitung permutasi dan kombinasi dari dua pasang bilangan bulat, menerima empat bilangan bulat: a, b, c, dan d, dengan syarat bilangan bulat a lebih besar dari c dan b lebih besar dari d. Setelah menerima input, program akan menghitung: Perubahan dan kombinasi pasangan (a, c) dan pasangan (b, d).

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.

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"
    "strings"
)

const maxTime = 301

// Prosedur untuk menghitung skor dan total waktu
func hitungSkor(soal [8]int, totalSoal *int, totalWaktu *int) {
    *totalSoal = 0
    *totalWaktu = 0
    for _, waktu := range soal {
        if waktu < maxTime {
            *totalSoal++
            *totalWaktu += waktu
        }
    }
}

func main() {
    var nama, pemenang string
    var soal [8]int
    var totalSoal, totalWaktu int
```

```

maxSoal := -1
minWaktu := maxTime * 8

for {
    // masukan nama peserta dan soal-soal yang
    di kerjakan
    fmt.Print("Masukkan nama peserta: ")
    fmt.Scan(&nama)
    if strings.ToLower(nama) == "selesai" {
        break
    }

    fmt.Println("Masukkan waktu pengerjaan: ")
    for i := 0; i < 8; i++ {
        fmt.Scan(&soal[i])
    }

    // Hitung jumlah soal yang telah diselesaikan dan
    waktu yang dibutuhkan untuk menyelesaikannya
    hitungSkor(soal, &totalSoal, &totalWaktu)

    // Cari pemenang sementara.
    if totalSoal > maxSoal || (totalSoal == maxSoal &&
totalWaktu < minWaktu) {
        maxSoal = totalSoal
        minWaktu = totalWaktu
        pemenang = nama
    }
}

fmt.Printf("%s %d %d\n", pemenang, maxSoal, minWaktu)
}

```

Screenshoot Output

```

PROBLEMS 11 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PC BRANDED\Documents\AfriZalVS\new\Semester03> go run
2.go
Masukkan nama peserta: Astuti
Masukkan waktu pengerjaan:
20 50 301 301 61 71 75 10
Masukkan nama peserta: Bertha
Masukkan waktu pengerjaan:
25 47 301 26 50 60 65 21
Masukkan nama peserta: selesai
Bertha 7 294
PS C:\Users\PC BRANDED\Documents\AfriZalVS\new\Semester03>

```

Deskripsi Program

Program ini dimaksudkan untuk menentukan pemenang kompetisi pemrograman berdasarkan jumlah soal yang berhasil diselesaikan oleh peserta dan total waktu yang dibutuhkan untuk menyelesaikannya. Maksimal delapan soal harus diselesaikan oleh setiap peserta, dan hanya soal yang diselesaikan dalam waktu kurang dari 301 menit dianggap berhasil.

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 $\frac{1}{2}n$, 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 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 sama 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"

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

func main() {
    var n int
    fmt.Print("Masukkan nilai suku awal: ")
    fmt.Scanln(&n)
    cetakDeret(n)
}
```

Screenshoot Output

```
PS C:\Users\PC BRANDED\Documents\AfrizalVS\new\Semester03> go run 3.go
Masukkan nilai suku awal: 22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS C:\Users\PC BRANDED\Documents\AfrizalVS\new\Semester03> |
```

Deskripsi Program

Program ini adalah implementasi dari Collatz conjecture, yang merupakan algoritma yang mengevaluasi deret bilangan berdasarkan nilai awal yang dimasukkan. Pada bagian utama (main), program meminta pengguna memasukkan bilangan awal n. Setelah menerima input, program memanggil prosedur cetakDeret untuk mencetak deret angka yang dihasilkan hingga mencapai angka 1, dengan spasi sebagai pemisah digunakan untuk mencetak setiap angka dalam deret.

Daftar Pustaka :

1. Modul Praktikum Algoritma dan Pemrograman 2 (Prosedur)