

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL IV
PROSEDUR**



Disusun Oleh :

Naya Putwi Setiasih / 2311102155

S1 11 IF - 05

Dosen Pengampu :

Arif Amrulloh, S.Kom., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

1. Definisi Prosedur

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. Suatu subprogram dikatakan prosedur apabila :

- Tidak ada deklarasi tipe nilai yang dikembalikan, dan
- Tidak terdapat kata kunci return dalam badan subprogram

Kedudukan prosedur sama seperti instruksi dasar yang sudah ada sebelumnya atau instruksi yang berasal dari paket (fmt), seperti fmt.scan dan fmt.Print.

2. Deklarasi Prosedur

Penulisan deklarasi 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.

3. Cara Pemanggilan Prosedur

Suatu prosedur hanya akan dieksekusi apabila dipanggil baik secara langsung atau tidak langsung oleh program utama. Prosedur dipanggil oleh program utama melalui perantara subprogram yang lain. Pemanggilan cukup mudah, yaitu dengan hanya menuliskan nama beserta parameter atau argumen yang diminta dari suatu prosedur.

4. Parameter

Suatu subprogram yang dipanggil dapat berkomunikasi dengan pemanggilnya melalui argumen yang diberikan melalui parameter yang dideklarasikan pada subprogramnya.

- Parameter Formal

Parameter formal adalah parameter yang dituliskan pada saat deklarasi suatu subprogram, parameter ini berfungsi sebagai petunjuk bahwa argumen apa saja yang diperlukan pada saat pemanggilan subprogram.

- Parameter Aktual

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.

II. GUIDED

1.

Soal Studi Case

Contoh program dengan function

Sourcecode

```
package main

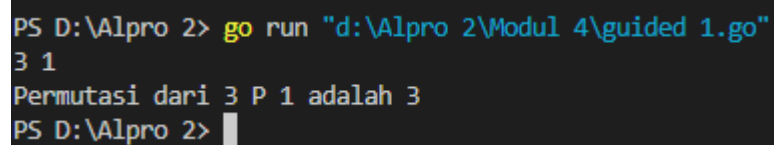
import "fmt"

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

// Prosedur faktorial yang mencetak hasilnya
func faktorial(n int) {
    var hasil int = 1
    for i := 1; i <= n; i++ {
        hasil = hasil * i
    }
    fmt.Println("Faktorial dari", n, "adalah", hasil)
}

// Prosedur permutasi yang mencetak hasilnya
func permutasi(n, r int) {
    // Faktorial dihitung langsung di dalam prosedur
    var hasilPermutasi int = 1
    for i := 0; i < r; i++ {
        hasilPermutasi *= n - i
    }
    fmt.Println("Permutasi dari", n, "P", r, "adalah",
        hasilPermutasi)
}
```

Screenshoot Output



```
PS D:\Alpro 2> go run "d:\Alpro 2\Modul 4\guided 1.go"
3 1
Permutasi dari 3 P 1 adalah 3
PS D:\Alpro 2> █
```

Deskripsi Program

Program ini ditulis dalam bahasa Go untuk menghitung dan mencetak nilai permutasi dari dua bilangan bulat yang diinputkan oleh pengguna. Prosedur utama (main) menerima dua input, a dan b, kemudian membandingkannya. Jika a lebih besar atau sama dengan b, fungsi permutasi dipanggil dengan argumen a dan b; jika tidak, dengan argumen b dan a. Fungsi permutasi menghitung nilai permutasi menggunakan rumus sederhana dengan mengalikan nilai dari n hingga n-r+1, dan hasilnya dicetak ke layar. Meskipun terdapat fungsi faktorial, fungsi ini tidak digunakan dalam program.

2. Menghitung Luas dan Keliling Persegi

Sourcecode

```
package main

import "fmt"

// Prosedur untuk menghitung dan mencetak luas persegi
func hitungLuas(sisi float64) {
    luas := sisi * sisi
    fmt.Printf("Luas persegi: %.2f\n", luas)
}

// Prosedur untuk menghitung dan mencetak keliling
// persegi
func hitungKeliling(sisi float64) {
    keliling := 4 * sisi
    fmt.Printf("Keliling persegi: %.2f\n", keliling)
}

func main() {
    var sisi float64

    fmt.Print("Masukkan panjang sisi persegi: ")
    fmt.Scan(&sisi)

    // Memanggil prosedur untuk menghitung luas dan
    // keliling
    hitungLuas(sisi)
    hitungKeliling(sisi)
}
```

Screenshoot Output

```
PS D:\Alpro 2> go run "d:\Alpro 2\Modul 4\guided 2.go"
Masukkan panjang sisi persegi: 7
Luas persegi: 49.00
Keliling persegi: 28.00
PS D:\Alpro 2> █
```

Deskripsi Program

Program ini ditulis dalam bahasa pemrograman Go dan bertujuan untuk menghitung serta mencetak luas dan keliling dari sebuah persegi berdasarkan panjang sisi yang diinputkan oleh pengguna. Prosedur utama (main) dimulai dengan mendeklarasikan variabel sisi bertipe float64 untuk menyimpan input dari pengguna. Setelah meminta pengguna memasukkan panjang sisi persegi, program menggunakan fungsi `fmt.Scan` untuk membaca nilai tersebut. Selanjutnya, program memanggil dua prosedur: `hitungLuas` untuk menghitung luas persegi dengan mengalikan sisi dengan dirinya sendiri, dan `hitungKeliling` untuk menghitung keliling persegi dengan mengalikan panjang sisi dengan 4. Hasil perhitungan luas dan keliling kemudian dicetak ke layar dengan format dua angka desimal. Program ini memberikan cara yang sederhana dan efektif untuk melakukan perhitungan geometri dasar pada persegi.

III. UNGUIDED

1. Minggu ini, mahasiswa Fakultas Informatika mendapatkan tugas dari matakuliah 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?

Masuka 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.

Sourcecode

```
package main

import "fmt"

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

// Prosedur untuk menghitung permutasi P(n, r)
func permutation(n, r int, hasil *int) {
    var nFact, nrFact int
    factorial(n, &nFact)           // menghitung n!
    factorial(n-r, &nrFact)       // menghitung (n-r)!
    *hasil = nFact / nrFact       // P(n, r) = n! / (n-r)!
}

// Prosedur untuk menghitung kombinasi C(n, r)
func combination(n, r int, hasil *int) {
    var nFact, rFact, nrFact int
    factorial(n, &nFact)           // menghitung n!
    factorial(r, &rFact)           // menghitung r!
    factorial(n-r, &nrFact)       // menghitung (n-r)!
    *hasil = nFact / (rFact * nrFact) // C(n, r) = n! / (r! * (n-r)!)
}

func main() {
    var a, b, c, d int
    fmt.Print("Masukkan 4 bilangan : ")
    fmt.Scan(&a, &b, &c, &d)

    // Memastikan bahwa a >= c dan b >= d
    if a < c || b < d {
```

```

        fmt.Println("Syarat a >= c dan b >= d tidak
terpenuhi.")
        return
    }

    // Hasil untuk permutasi dan kombinasi
    var hasilPermutasiC, hasilKombinasiC,
    hasilPermutasiD, hasilKombinasiD int

    // Hitung permutasi dan kombinasi untuk a terhadap c
    permutation(a, c, &hasilPermutasiC)
    combination(a, c, &hasilKombinasiC)

    // Hitung permutasi dan kombinasi untuk b terhadap d
    permutation(b, d, &hasilPermutasiD)
    combination(b, d, &hasilKombinasiD)

    // Cetak hasil
    fmt.Println(hasilPermutasiC, hasilKombinasiC)
    fmt.Println(hasilPermutasiD, hasilKombinasiD)
}

```

Screenshoot Output

```

PS D:\Alpro 2> go run "d:\Alpro 2\Modul 4\unguided 1.go"
Masukkan 4 bilangan : 6 10 3 10
120 20
3628800 1
PS D:\Alpro 2>

```

```

PS D:\Alpro 2> go run "d:\Alpro 2\Modul 4\tempCodeRunnerFile.go"
Masukkan 4 bilangan : 8 0 2 0
56 28
1 1
PS D:\Alpro 2>

```

Deskripsi Program

Program ditulis dalam bahasa Go untuk menghitung nilai permutasi dan kombinasi dari empat bilangan bulat yang diinputkan oleh pengguna. Setelah meminta input a, b, c, dan d program memeriksa syarat $a \geq c$ dan $b \geq d$. Jika syarat tidak terpenuhi, program akan menampilkan pesan kesalahan. Jika syarat terpenuhi, fungsi permutasi dan kombinasi digunakan untuk menghitung permutasi $P(n, r)$ dan kombinasi $C(n, r)$ dengan memanfaatkan fungsi faktorial untuk menghitung faktorial yang diperlukan. Hasil perhitungan untuk kedua pasangan bilangan kemudian dicetak ke layar. Program ini memberikan cara sederhana untuk melakukan perhitungan kombinatorial berdasarkan input pengguna.

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 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.

Sourcecode

```
package main

import (
    "fmt"
)

// Peserta merepresentasikan seorang peserta dengan nama
// dan waktu pengerjaan soal
type Peserta struct {
    nama string
    waktu []int
}

// hitungSkor menghitung total soal yang diselesaikan
// dan total waktu yang dibutuhkan
func hitungSkor(peserta Peserta) (int, int) {
    totalSoal, totalWaktu := 0, 0
    for _, waktu := range peserta.waktu {
        if waktu < 301 { // Jika soal diselesaikan dalam
            waktu kurang dari 5 jam 1 menit
            totalSoal++
            totalWaktu += waktu
        }
    }
    return totalSoal, totalWaktu
}
```



```

}

// cariPemenang mencari peserta dengan skor tertinggi
func cariPemenang(peserta []Peserta) Peserta {
    var pemenang Peserta
    skorTertinggi := 0
    waktuTerendah := 1000000 // Nilai besar untuk
    memastikan waktu pemenang dihitung dengan benar

    for _, p := range peserta {
        totalSoal, totalWaktu := hitungSkor(p)
        // Bandingkan total soal yang diselesaikan dan
        waktu penyelesaian
        if totalSoal > skorTertinggi || (totalSoal ==
        skorTertinggi && totalWaktu < waktuTerendah) {
            pemenang = p
            skorTertinggi = totalSoal
            waktuTerendah = totalWaktu
        }
    }

    return pemenang
}

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

    // Inisialisasi slice untuk menampung data peserta
    peserta := make([]Peserta, n)

    // Input data untuk setiap peserta
    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan nama peserta %d: ", i+1)
        fmt.Scan(&peserta[i].nama)

        // Inisialisasi slice untuk waktu pengerjaan
        soal
        peserta[i].waktu = make([]int, 8)
        fmt.Printf("Masukkan waktu pengerjaan untuk
        %s:\n", peserta[i].nama)

        // Input waktu soal dalam satu baris
        for j := 0; j < 8; j++ {
            fmt.Scan(&peserta[i].waktu[j])
        }
    }

    // Cari pemenang
    pemenang := cariPemenang(peserta)

```

```

// Hitung skor pemenang
totalSoal, totalWaktu := hitungSkor(pemenang)

// Tampilkan hasil
fmt.Printf("\nPemenang: %s\n", pemenang.nama)
fmt.Printf("Jumlah soal yang diselesaikan: %d\n",
totalSoal)
fmt.Printf("Total waktu: %d menit\n", totalWaktu)
}

```

Screenshoot Output

```

PS D:\Alpro 2> go run "d:\Alpro 2\Modul 4\tempCodeRunnerFile.go"
Masukkan jumlah peserta: 2
Masukkan nama peserta 1: astuti
Masukkan waktu pengerjaan untuk astuti:
20 50 301 301 61 71 75 10
Masukkan nama peserta 2: Bertha
Masukkan waktu pengerjaan untuk Bertha:
25 47 301 26 50 60 65 21

Pemenang: Bertha
Jumlah soal yang diselesaikan: 7
Total waktu: 294 menit

```

Deskripsi Program

Program ini ditulis dalam Bahasa Go untuk menentukan pemenang kompetisi berdasarkan waktu pengerjaan soal peserta. Setiap peserta diwakili oleh struktur peserta, yang menyimpan nama dan waktu pengerjaan untuk delapan soal. Setelah pengguna memasukkan jumlah peserta dan data mereka, program menghitung total soal yang diselesaikan dan total waktu menggunakan fungsi `hitungSkor`. Fungsi `cariPemenang` kemudian memilih pemenang berdasarkan jumlah soal yang diselesaikan dan waktu terendah. Akhirnya, program mencetak nama pemenang, jumlah soal yang diselesaikan, dan total waktu yang dibutuhkan. Program ini secara efisien menilai kinerja peserta dalam kompetisi berbasis waktu.

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 jika suku terakhir bernilai 1. Buatlah program Skiena yang akan mencetak setiap suku dari deret yang dijelaskan diatas untuk nilai suku awal yang

diberikan. Pencetakan deret harus dibuat dalam prosedur cetak deret yang mempunyai 1 parameter formal, yaitu nilai dari suku awal.

```
procedure 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

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) // Menampilkan 1 di akhir deret
}

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

Screenshoot Output

```
PS D:\Alpro 2> go run "d:\Alpro 2\Modul 4\unguided 3.go"
Masukkan nilai awal: 12
12 6 3 10 5 16 8 4 2 1
PS D:\Alpro 2> 
```

Deskripsi Program

Program ini ditulis dalam Bahasa Go untuk mencetak deret angka berdasarkan algoritma Collatz, yang dimulai dari nilai awal yang diinputkan oleh pengguna. Prosedur utama (main) meminta pengguna untuk memasukkan nilai integer, kemudian memanggil fungsi cetakDeret dengan nilai tersebut. Fungsi cetakDeret menjalankan perulangan hingga nilai n mencapai 1. Di dalam perulangan, program mencetak nilai n , lalu memeriksa apakah n genap atau ganjil. Jika genap, n dibagi dua, jika ganjil n diubah menjadi $3*n+1$. Proses ini berlanjut hingga mencapai angka 1, yang juga dicetak diakhir deret. Dengan program ini, menunjukkan bagaimana deret Collatz berkembang dari nilai awal hingga mencapai 1.

DAFTAR PUSTAKA

Asisten Praktikum, “Modul 4 Prosedur”, Learning Management System, 2024.