

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL IV
PROSEDUR**



Oleh:

FEROS PEDROSA VALENTINO

103112400055

IF-12-01

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2025

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 bisa dikatakan sebagai prosedur jika memenuhi syarat berikut:

1. Tidak ada deklarasi nilai tipe 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.

Cara pemanggilan prosedur yaitu dengan menuliskan nama beserta parameter atau argumen yang diminta dari suatu prosedur. Karena suatu prosedur hanya akan dieksekusi apabila dipanggil baik secara langsung atau tidak langsung oleh program utama. Tidak langsung maksudnya itu prosedur dipanggil oleh program utama melalui perantara subprogram lain.

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. 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.

2. 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.

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.

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 si pemanggil setelah subprogram tersebut selesai dieksekusi. Pass by reference sebaiknya digunakan hanya untuk prosedur.

II. GUIDED

1. Guided 1

Source code:

```
//Feros Pedrosa

package main

import "fmt"

// prosedur untuk menghitung total gaji karyawan
func hitungGaji(nama string, gajiPokok float64, jamLembur
int) {
    bonusLembur := float64(jamLembur) * 50000
    totalGaji := gajiPokok + bonusLembur

    fmt.Println("\n=== Slip Gaji ===")
    fmt.Println("Nama Karyawan :", nama)
    fmt.Printf("Gaji Pokok   : Rp%.2f\n", gajiPokok)
    fmt.Printf("Bonus Lembur  : Rp%.2f (%d jam x
Rp50,000)\n", bonusLembur, jamLembur)
    fmt.Printf("Total Gaji    : Rp%.2f\n", totalGaji)
}

func main() {
    var nama string
    var gajiPokok float64
    var jamLembur int

    //input pengguna
    fmt.Print("Masukkan Nama Karyawan : ")
    fmt.Scanln(&nama)

    fmt.Print("Masukkan Gaji Pokok: ")
    fmt.Scanln(&gajiPokok)

    fmt.Print("Masukkan Jumlah Jam Lembur: ")
    fmt.Scanln(&jamLembur)

    //mengambil prosedur dengan data dari pengguna
    hitungGaji(nama, gajiPokok, jamLembur)
}
```

Output:

```
PS C:\ALPRO\semester2_alpro2\alpro2_week4> go run "c:\ALPRO\semester2_alpro2\alpro2_week4\coso1\guided1.go"
Masukkan Nama Karyawan : rosi
Masukkan Gaji Pokok: 100000
Masukkan Jumlah Jam Lembur: 2

=== Slip Gaji ===
Nama Karyawan : rosi
Gaji Pokok    : Rp100000.00
Bonus Lembur  : Rp100000.00 (2 jam x Rp50,000)
Total Gaji    : Rp200000.00
```

Deskripsi program:

Program diatas ditulis dalam bahasa Go dan berfungsi untuk menghitung total gaji karyawan berdasarkan gaji pokok dan jumlah jam lembur. Prosedur fungsi hitungGaji menerima tiga parameter yaitu nama, gajiPokok, jamLembur. Prosedur ini berfungsi untuk menghitung totalGaji dengan rumus mengalikan jumlah jam lembur dengan Rp50.000 per jam, lalu menjumlahkannya dengan gaji pokok untuk mendapatkan total gaji. Kemudian hasilnya akan ditampilkan dalam slip gaji yang berisi nama karyawan, gaji pokok, bonus lembur, total gaji. Lalu di fungsi main pertama deklarasikan variabel nama sebagai tipe data string, variabel gajiPokok sebagai tipe data float64, dan variabel jamLembur sebagai tipe data integer. Selanjutnya program meminta pengguna memasukkan nama karyawan, gaji pokok, jumlah jam lembur. Setelah itu program akan memanggil prosedur hitungGaji untuk menampilkan slip gaji berdasarkan data inputan.

2. Guided 2

Source code:

```
//Feros Pedrosa

package main

import "fmt"

//prosedur untuk menghitung rata-rata dan menentukan
kelulusan
func hitungRataRata(nama string, nilai1, nilai2, nilai3
float64) {
    rataRata := (nilai1 + nilai2 + nilai3) / 3
    status := "Tidak Lulus"
    if rataRata >= 60 {
        status = "Lulus"
    }

    fmt.Println("\n=== Hasil Akademik ===")
    fmt.Println("Nama Mahasiswa :", nama)
    fmt.Printf("Nilai 1      : %.2f\n", nilai1)
    fmt.Printf("Nilai 2      : %.2f\n", nilai2)
    fmt.Printf("Nilai 3      : %.2f\n", nilai3)
    fmt.Printf("Rata-rata    : %.2f\n", rataRata)
    fmt.Println("Status      :", status)
}

func main() {
    var nama string
    var nilai1, nilai2, nilai3 float64

    //input dari pengguna
    fmt.Print("Masukkkkan Nama Mahasiswa: ")
    fmt.Scanln(&nama)

    fmt.Print("Masukkan Nilai 1: ")
    fmt.Scanln(&nilai1)

    fmt.Print("Masukkan Nilai 2: ")
    fmt.Scanln(&nilai2)

    fmt.Print("Masukkan Nilai 3: ")
    fmt.Scanln(&nilai3)

    hitungRataRata(nama, nilai1, nilai2, nilai3)
}
```

Output:

```
PS C:\ALPRO\semester2_alpro2\alpro2_week4> go run "c:\ALPRO\semester2_alpro2\alpro2_week4\coso2\guided2.go"
Masukkan Nama Mahasiswa: ros
Masukkan Nilai 1: 100
Masukkan Nilai 2: 90
Masukkan Nilai 3: 95

=== Hasil Akademik ===
Nama Mahasiswa : ros
Nilai 1       : 100.00
Nilai 2       : 90.00
Nilai 3       : 95.00
Rata-rata     : 95.00
Status        : Lulus
```

Deskripsi program:

Program diatas ditulis dalam bahasa Go dan berfungsi untuk menghitung rata-rata nilai mahasiswa dan menentukan status kelulusan berdasarkan nilai yang diperoleh. Prosedur bernama `hitungRataRata` menerima empat parameter yaitu nama, nilai1, nilai2, nilai3. Pada prosedur ini nilai rata-rata dihitung dengan menjumlahkan ketiganya lalu membagi dengan tiga. Selanjutnya program akan menentukan status kelulusan dengan aturan jika nilai rata-rata diatas atau sama dengan 60 maka statusnya lulus. Kemudian hasilnya ditampilkan dalam hasil akademik yang berisi nama mahasiswa, nilai1, nilai2, nilai3, rata-rata, status. Pada fungsi main deklarasikan variabel nama sebagai tipe data string dan variabel nilai1, nilai2, nilai3 sebagai tipe data float64. Lalu program meminta untuk menginputkan nama mahasiswa, nilai1, nilai2, nilai3. Setelah data dimasukkan program akan memanggil prosedur `hitungRataRata` untuk melakukan perhitungan dan menampilkan hasilnya.

III. UNGUIDED

1. Unguided 1

Source code:

```
//Feros Pedrosa

package main

import "fmt"

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

func permutasi(n, r int, hasil *int) {
    var test1, test2 int
    if r > n {
        *hasil = 0
        return
    }
    faktorial(n, &test1)
    faktorial(n-r, &test2)
    *hasil = test1 / test2
}

func kombinasi(n, r int, hasil *int) {
    var test1, test2, test3 int
    if r > n {
        *hasil = 0
        return
    }
    faktorial(n, &test1)
    faktorial(n-r, &test2)
    faktorial(r, &test3)
    *hasil = test1 / (test2 * test3)
}

func main() {
    var a, b, c, d int
    var permu, kombi int
    fmt.Scan(&a, &b, &c, &d)

    if a >= c && b >= d {
```



```

    permutasi(a, c, &permu)
    kombinasi(a, c, &kombi)
    fmt.Println(permu, kombi)

    permutasi(b, d, &permu)
    kombinasi(b, d, &kombi)
    fmt.Println(permu, kombi)
} else {
    fmt.Println("input tidak valid")
}
}

```

Output:

```

PS C:\ALPRO\semester2_alpro2\alpro2_week4> go run "c:\ALPRO\semester2_alpro2\alpro2_week4\latsol1-MODUL4\unguided1.go"
5 10 3 10
60 10
3628800 1
PS C:\ALPRO\semester2_alpro2\alpro2_week4> go run "c:\ALPRO\semester2_alpro2\alpro2_week4\latsol1-MODUL4\unguided1.go"
8 0 2 0
56 28
1 1

```

Deskripsi program:

Program diatas ditulis dalam bahasa Go dan berfungsi untuk menghitung nilai permutasi dan kombinasi. Fungsi faktorial berguna untuk menghitung faktorial bilangan n dengan perulangan. Fungsi permutasi berguna untuk menghitung permutasi dengan rumus $n! / (n-r)!$ dengan memanggil fungsi faktorial untuk menghitung nilai n faktorial dan n-r faktorial. Jika $r > n$ maka hasil permutasi diatur menjadi 0. Fungsi kombinasi berguna untuk menghitung kombinasi dengan rumus $n! / ((n-r)! * r!)$ dan fungsi ini juga membutuhkan fungsi faktorial. Jika $r > n$ maka hasil kombinasi diatur menjadi 0. Di fungsi main deklarasikan variabel a, b, c, d sebagai tipe data integer dan deklarasikan variabel permu, kombi sebagai tipe data integer lalu program meminta pengguna untuk memasukkan 4 bilangan. Kemudian program akan mengecek jika $a \geq c$ dan $b \geq d$ program akan menghitung permutasi dan kombinasi untuk (a, c) dan (b, d), lalu mencetak hasilnya. Jika tidak maka program akan mencetak "input tidak valid".

2. Unguided 2

Source code:

```
//Feros Pedrosa

package main

import "fmt"

const maxTime = 301 // Waktu maks jika soal tidak selesai

func hitungSkor(waktu [8]int, soal *int, skor *int) {
    *soal = 0
    *skor = 0
    for _, waktuSoal := range waktu {
        if waktuSoal < maxTime {
            *soal++
            *skor += waktuSoal
        }
    }
}

func main() {
    var pemenang string
    var soalPemenang, skorPemenang int
    soalPemenang = 0
    skorPemenang = maxTime * 8

    for {
        var nama string
        var waktu [8]int
        fmt.Scan(&nama)

        if nama == "Selesai" {
            break
        }

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

        var soal, skor int
        hitungSkor(waktu, &soal, &skor)

        if soal > soalPemenang || (soal == soalPemenang &&
            skor < skorPemenang) {
            pemenang = nama
        }
    }
}
```

```

        soalPemenang = soal
        skorPemenang = skor
    }
}

fmt.Printf("Pemenang: %s, Soal yang diselesaikan: %d,
Total waktu: %d menit\n", pemenang, soalPemenang,
skorPemenang)
}

```

Output:

```

PS C:\ALPRO\semester2_alpro2\alpro2_week4> go run "c:\ALPRO\semester2_alpro2\alpro2_week4\latso12-MODUL4\unguided2.go"
Astuti 20 50 301 301 61 71 75 10
Bertha 25 47 301 26 50 60 65 21
Selesai
Pemenang: Bertha, Soal yang diselesaikan: 7, Total waktu: 294 menit

```

Deskripsi program:

Program diatas ditulis dalam bahasa Go dan berfungsi untuk menentukan pemenang berdasarkan jumlah soal dan total waktu. Jika suatu soal memiliki waktu lebih dari 301 menit, maka dianggap tidak selesai dan tidak dihitung dalam penilaian. Fungsi hitungSkor untuk menghitung jumlah soal yang berhasil diselesaikan dan total waktu yang digunakan. Jika waktu pengerjaan suatu soal kurang dari 301 menit, maka soal tersebut dihitung sebagai soal yang diselesaikan dan waktu pengerjaannya dijumlahkan ke skor total. Di fungsi main deklarasi variabel pemenang sebagai tipe data string dan variabel soalPemenang, skorPemenang sebagai tipe data integer lalu program terus membaca input peserta dan memanggil fungsi hitungSkor untuk menentukan jumlah soal yang diselesaikan dan total waktu yang dihabiskan. Jika pengguna memasukkan kata "Selesai" maka program akan berhenti dan menampilkan pemenang. Pemenang ditentukan berdasarkan jumlah soal yang paling banyak diselesaikan. Jika ada lebih dari satu peserta dengan jumlah soal yang sama, maka peserta dengan total waktu lebih sedikit yang menjadi pemenang. Terakhir program mencetak nama pemenang beserta jumlah soal yang diselesaikan beserta total waktunya.

3. Unguided 3

Source code:

```
//Feros Pedrosa

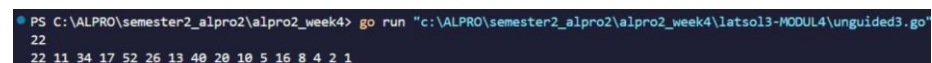
package main

import "fmt"

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

func main() {
    var n int
    fmt.Scan(&n)
    cetakDeret(n)
}
```

Output:



```
PS C:\ALPRO\semester2_alpro2\alpro2_week4> go run "c:\ALPRO\semester2_alpro2\alpro2_week4\latso13-MODUL4\unguided3.go"
22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

Deskripsi program:

Program diatas ditulis dalam bahasa Go dan berfungsi untuk menghitung deret suatu bilangan. Fungsi cetakDeret(n int) digunakan untuk mencetak deret dari bilangan n dengan menggunakan perulangan dan dengan rumus jika n adalah bilangan genap maka n dibagi 2, jika n bilangan ganjil maka n dikali 3 dan ditambah 1. Selama n tidak sama dengan 1 maka program terus mencetak nilai n. Setelah mencapai nilai 1 angka tersebut akan dicetak dan akhirnya program selesai. Didalam fungsi main pertama deklarasikan variabel n sebagai tipe data integer lalu program meminta untuk memasukkan sebuah angka dan kemudian program memanggil fungsi cetakDeret untuk menampilkan hasilnya.

IV. KESIMPULAN

Prosedur dalam pemrograman merupakan cara yang efektif untuk menyederhanakan kode dengan membagi menjadi bagian-bagian yang lebih kecil. Prosedur memungkinkan pemrogram untuk mengorganisir kode dengan lebih baik serta menghindari duplikasi dengan memanfaatkan parameter formal dan aktual dalam proses komunikasi antar bagian program. Prosedur tidak hanya meningkatkan keterbacaan dan efisiensi kode, tetapi juga membantu dalam pemanfaatan memori secara lebih optimal dengan teknik pass by value dan pass by reference.

REFERENSI

Modul 4 – Praktikum Alpro 2