

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 4
PROSEDUR**



Disusun Oleh :

Afif Rijal Azzami / 2311102235

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 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.

C. Cara Pemanggilan Procedure

Dengan memanggil parameter pada func main, parameter merupakan variabel yang menempel di fungsi yang nilainya ditentukan saat pemanggilan fungsi tersebut. Parameter sifatnya opsional, suatu fungsi bisa tidak memiliki parameter, atau bisa saja memiliki satu atau banyak parameter (tergantung kebutuhan).

D. Parameter

Suatu subprogram yang dipanggil dapat berkomunikasi dengan pemanggilannya melalui argumen yang diberikan melalui parameter yang dideklarasikan pada subprogramnya. Berdasarkan letak penulisannya pada program, maka parameter dapat dikelompokkan menjadi dua, yaitu :

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 :

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.

Selain itu, parameter juga dikelompokkan berdasarkan lokasi memorinya, yaitu:

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

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.

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.

II. GUIDED 1

Sourcecode

```
package main

import "fmt"

func hitungFaktorial(n int) {
    var hasil int = 1
    for i := 1; i <= n; i++ {
        hasil *= i
    }
    fmt.Println("Hasil faktorial dari", n, "adalah", hasil)
}

func hitungPermutasi(n, r int) {
    faktorialN := 1
    for i := 1; i <= n; i++ {
        faktorialN *= i
    }

    faktorialNR := 1
    for i := 1; i <= n-r; i++ {
        faktorialNR *= i
    }
}
```

```

hasilPermutasi := faktorialN / faktorialNR

    fmt.Println("Hasil permutasi =", hasilPermutasi)
}

func main() {

    var a, b int

    fmt.Print("Masukkan bilangan a = ")

    fmt.Scan(&a)

    fmt.Print("Masukkan bilangan b = ")

    fmt.Scan(&b)

    if a >= b {

        hitungPermutasi(a, b)

    } else {

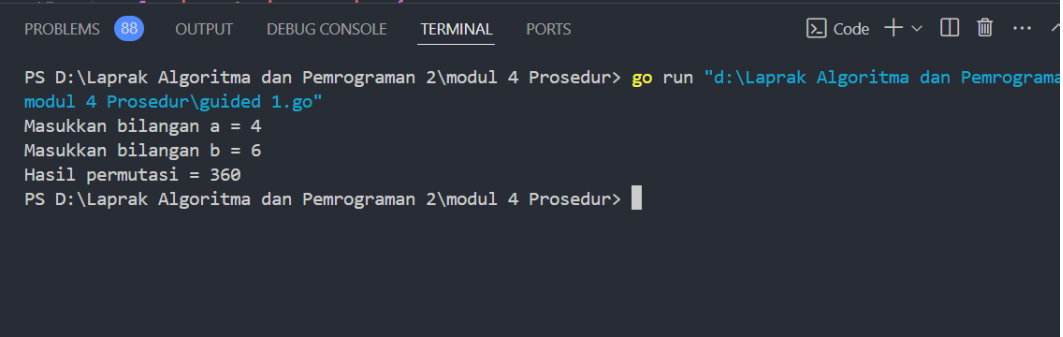
        hitungPermutasi(b, a)

    }

}

```

Screenshot Output



```

PROBLEMS 88 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur> go run "d:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur\guided 1.go"
Masukkan bilangan a = 4
Masukkan bilangan b = 6
Hasil permutasi = 360
PS D:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur>

```

Deskripsi Program

Sintaks tersebut adalah program untuk mencari hasil dari permutasi menggunakan prosedur, untuk menghitungnya pertama kita membuat prosedur yang diberi nama `hitungFaktorial`, prosedur tersebut untuk mencari hasil dari faktorial dengan perulang dan mengalikan setiap bilangan yang diulang, kemudian kita membuat prosedur yang diberi nama `hitungPermutasi`, fungsi tersebut berisikan rumus dari permutasi yaitu faktorial dari nilai n dibagi dengan faktorial dari hasil pengurangan n dan r ($n-r$). Pada func main kita menggunakan pengondisian `if`, karna untuk mencari permutasi nilai $n \geq r$, jika $b < a$ maka yang jadi nilai n itu b dan nilai r itu b .

III. GUIDED 2

Sourcecode

```
package main

import (
    "fmt"
)

// Prosedur untuk menghitung dan mencetak luas persegi
func hitungLuasPersegi(sisi int) {
    luas := sisi * sisi

    fmt.Printf("Luas persegi: %d\n", luas)
}

// Prosedur untuk menghitung dan mencetak keliling persegi
func hitungKelilingPersegi(sisi int) {
    keliling := 4 * sisi

    fmt.Printf("Keliling persegi: %d\n", keliling)
}
```

```

func main() {

    // Meminta input panjang sisi persegi

    var panjangSisi int

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

    fmt.Scan(&panjangSisi)


    // Menghitung dan menampilkan luas dan keliling persegi
    menggunakan prosedur

    hitungLuasPersegi(panjangSisi)

    hitungKelilingPersegi(panjangSisi)

}

```

Screenshot Output



```

PROBLEMS 88 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur> go run "d:\Laprak Algoritma dan Pemrogra
modul 4 Prosedur\guided 2.go"
Masukkan panjang sisi persegi: 8
Luas persegi: 64
Keliling persegi: 32
PS D:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur>

```

Deskripsi Program

Sintaks tersebut adalah program untuk menghitung luas dan panjang pada persegi menggunakan prosedur. Pertama kita membuat prosedur yang dinamai `hitungLuasPersegi` dan memasukan rumus yaitu sisi dikali sisi, dan prosedur `hitungKelilingPersegi` yang di dalamnya terdapat rumus yaitu sisi dikali 4. kemudian pada `func main` kita panggil prosedur yg sudah dibuat dan nantinya ketika user meng-input sisi persegi, hasil dari operasi luas dan keliling persegi akan ditampilkan.

IV. UNGUIDED 1

```
package main

import "fmt"

// Prosedur untuk menghitung faktorial dari n
func faktorial(n int, hasil *int) {

    *hasil = 1

    for i := 1; i <= n; i++ {

        *hasil *= i

    }

}

// Prosedur untuk menghitung permutasi  $P(n, r)$ 
func permutasi(n, r int, hasil *int) {

    var faktN, faktNR int

    faktorial(n, &faktN)

    faktorial(n-r, &faktNR)

    *hasil = faktN / faktNR

}

// Prosedur untuk menghitung kombinasi  $C(n, r)$ 
func kombinasi(n, r int, hasil *int) {

    var faktN, faktR, faktNR int

    faktorial(n, &faktN)

    faktorial(r, &faktR)

    faktorial(n-r, &faktNR)

    *hasil = faktN / (faktR * faktNR)

}
```



```
func main() {  
  
    var a, b, c, d int  
  
    fmt.Print("Masukkan nilai a dan c (dengan a >= c dan b >=  
d): ")  
  
    fmt.Scan(&a, &c)  
  
    fmt.Print("Masukkan nilai b dan d (dengan a >= c dan b >=  
d): ")  
  
    fmt.Scan(&b, &d)  
  
  
    // Validasi syarat a >= c dan b >= d  
  
    if a >= c && b >= d {  
  
        // Variabel untuk menampung hasil permutasi dan kombinasi  
  
        var hasilPermutasi1, hasilKombinasi1 int  
  
        var hasilPermutasi2, hasilKombinasi2 int  
  
  
        // Baris pertama: Hasil permutasi dan kombinasi a  
terhadap c  
  
        permutasi(a, c, &hasilPermutasi1)  
  
        kombinasi(a, c, &hasilKombinasi1)  
  
        fmt.Printf("Permutasi P(%d, %d) = %d\n", a, c,  
hasilPermutasi1)  
  
        fmt.Printf("Kombinasi C(%d, %d) = %d\n", a, c,  
hasilKombinasi1)
```

```
// Baris kedua: Hasil permutasi dan kombinasi b terhadap d

    permutasi(b, d, &hasilPermutasi2)

    kombinasi(b, d, &hasilKombinasi2)

    fmt.Printf("Permutasi P(%d, %d) = %d\n", b, d,
hasilPermutasi2)

    fmt.Printf("Kombinasi C(%d, %d) = %d\n", b, d,
hasilKombinasi2)

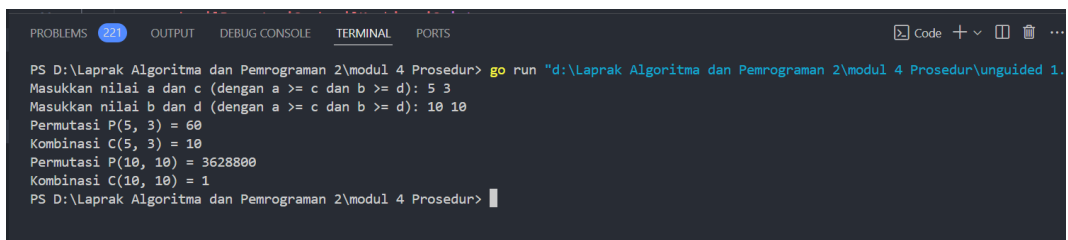
    } else {

        fmt.Println("Pastikan bahwa a >= c dan b >= d.")

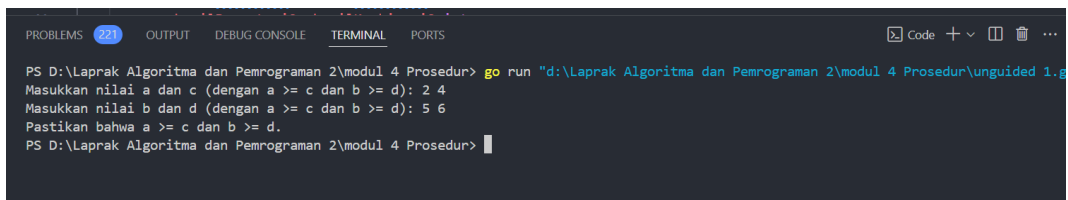
    }

}
```

Screenshoot Output



```
PS D:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur> go run "d:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur\unguided 1.g
Masukkan nilai a dan c (dengan a >= c dan b >= d): 5 3
Masukkan nilai b dan d (dengan a >= c dan b >= d): 10 10
Permutasi P(5, 3) = 60
Kombinasi C(5, 3) = 10
Permutasi P(10, 10) = 3628800
Kombinasi C(10, 10) = 1
PS D:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur>
```



```
PS D:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur> go run "d:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur\unguided 1.g
Masukkan nilai a dan c (dengan a >= c dan b >= d): 2 4
Masukkan nilai b dan d (dengan a >= c dan b >= d): 5 6
Pastikan bahwa a >= c dan b >= d.
PS D:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur>
```

Deskripsi Program

Sintaks tersebut adalah program untuk menghitung permutasi dan kombinasi menggunakan kombinasi dengan syarat $a \geq c$ dan $b \geq d$. Pertama kita buat prosedur yang dinamai faktorial untuk menghitung hasil faktorial dari bilangan

yang di-inputkan dengan mengalikan setiap bilangan pada perulangan, kemudian kita membuat prosedur yang dinamai permutasi untuk menghitung permutasi dengan rumus faktorial dari n dibagi dengan hasil faktorial dari n dikurangi r, kemudian kita buat prosedur yang dinamai kombinasi untuk menghitung kombinasi dengan rumus faktorial dari n dibagi dengan hasil faktorial r dikalikan dengan faktorial dari $n - r$. Pada int main kita memasukan if $a \geq c$ dan $b \geq d$ agar syarat dari rumus terpenuhi, jika user menginputkan tidak sesuai syarat maka program akan menampilkan pastikan bahwa $a \geq c$ dan $b \geq d$.

I. UNGUIDED 2

Sourcecode

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strconv"
)

var (
    totalSoalTerbanyak, totalWaktuTercepat int
    namaPemenang                          string
)

func hitungSkor(soal []int, nama string) {
    totalSoal, totalWaktu := 0, 0
    for _, waktu := range soal {
        if waktu != 301 {
```

```

totalSoal++

        totalWaktu += waktu

    }

}

    if totalSoal > totalSoalTerbanyak || (totalSoal ==
totalSoalTerbanyak && totalWaktu < totalWaktuTercepat) {

        totalSoalTerbanyak, totalWaktuTercepat, namaPemenang =
totalSoal, totalWaktu, nama

    }

}

func bacaInput(scanner *bufio.Scanner) {

    fmt.Print("Masukkan nama peserta: ")

    scanner.Scan() // Baca nama peserta

    nama := scanner.Text()

    waktu := make([]int, 8)

    fmt.Println("Masukkan waktu pengerjaan untuk 8 soal (dalam
menit, max 301 menit):")

    for i := 0; i < 8; i++ {

        fmt.Printf("Soal %d: ", i+1)

        scanner.Scan()

        waktu[i], _ = strconv.Atoi(scanner.Text())

    }

    hitungSkor(waktu, nama)

}

```

```

func main() {

    scanner := bufio.NewScanner(os.Stdin)

    var jumlahPeserta int

    fmt.Print("Masukkan jumlah peserta: ")

    fmt.Scan(&jumlahPeserta)

    // Buang newline yang tersisa setelah fmt.Scan

    scanner.Scan()

    for i := 0; i < jumlahPeserta; i++ {

        fmt.Printf("\nPeserta ke-%d\n", i+1)

        bacaInput(scanner)

    }

    fmt.Printf("\nPemenang: %s\nJumlah soal diselesaikan:
%d\nTotal waktu: %d menit\n", namaPemenang, totalSoalTerbanyak,
totalWaktuTercepat)

}

```

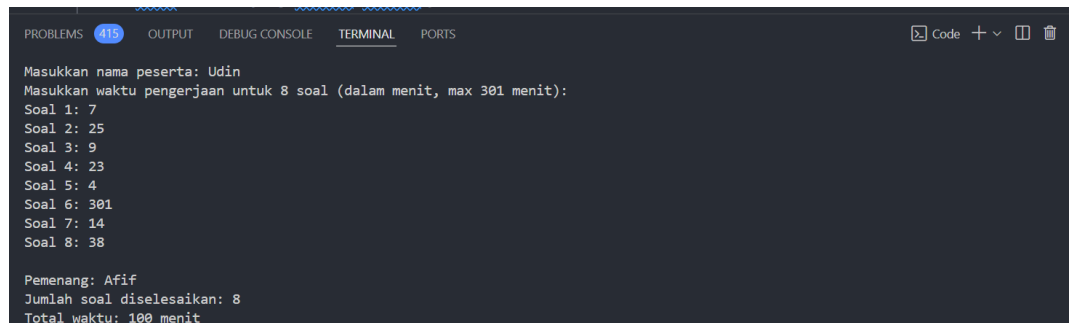
Screenshoot Output

```

PROBLEMS 415 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur> go run "d:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur\unguided"
Masukkan jumlah peserta: 2

Peserta ke-1
Masukkan nama peserta: Afif
Masukkan waktu pengerjaan untuk 8 soal (dalam menit, max 301 menit):
Soal 1: 15
Soal 2: 6
Soal 3: 10
Soal 4: 21
Soal 5: 4
Soal 6: 32
Soal 7: 9
Soal 8: 3

```

A screenshot of a terminal window with a dark background. The terminal shows the output of a program. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is active), and 'PORTS'. The output text is as follows:

```
Masukkan nama peserta: Udin
Masukkan waktu pengerjaan untuk 8 soal (dalam menit, max 301 menit):
Soal 1: 7
Soal 2: 25
Soal 3: 9
Soal 4: 23
Soal 5: 4
Soal 6: 301
Soal 7: 14
Soal 8: 38

Pemenang: Afif
Jumlah soal diselesaikan: 8
Total waktu: 100 menit
```

Deskripsi Program

Sintaks tersebut adalah program untuk menentukan pemenang pada sebuah kompetisi berdasarkan jumlah soal yang diselesaikan oleh peserta dan waktu yang dihabiskan. Peserta yang menyelesaikan soal terbanyak dan tercepat akan dinyatakan sebagai pemenang. Pertama kita mendeklarasikan variabel global dimana variabel tersebut untuk menyimpan total soal terbanyak, total waktu tercepat, dan nama pemenang, kemudian kita membuat prosedur `hitungSkor` untuk menghitung jumlah soal yang diselesaikan dan total waktu pengerjaan. Jika waktu pengerjaan soal adalah 301 (menandakan soal tidak selesai), soal tersebut diabaikan. Setelah jumlah soal dan total waktu dihitung, program membandingkan nilai ini dengan pemenang saat ini (`totalSoalTerbanyak` dan `totalWaktuTercepat`), Jika jumlah soal lebih banyak, atau jika jumlah soal sama tetapi total waktu lebih sedikit, peserta tersebut menjadi pemenang baru. Kemudian kita membuat prosedur `bacaInput` untuk mengumpulkan data peserta (nama dan waktu pengerjaan soal) dan kemudian mengirimkan data tersebut untuk dihitung skornya.

II. UNGUIDED 3

```
package main

import (
    "fmt"
)

// Fungsi untuk mencetak deret bilangan sesuai dengan aturan
func cetakDeret(n int) {
    // Cetak suku awal
    fmt.Print(n)

    // Ulangi sampai nilai n mencapai 1
    for n != 1 {
        // Jika n genap, bagi dengan 2
        if n%2 == 0 {
            n /= 2
        } else { // Jika n ganjil, kalikan dengan 3 dan tambahkan
1          n = 3*n + 1
        }

        // Cetak nilai suku berikutnya, dipisahkan dengan spasi
        fmt.Print(" ", n)
    }

    fmt.Println() // Baris baru setelah selesai mencetak deret
}
```

```

func main() {

    var n int

    // Input nilai n

    fmt.Print("Masukkan nilai n: ")

    fmt.Scan(&n)

    // Cetak deret berdasarkan n

    cetakDeret(n)

}

```

Screenshot Output

```

PS D:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur> go run "d:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur\unguidec
Masukkan nilai n: 22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS D:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur>

```

```

PS D:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur> go run "d:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur\unguidec
Masukkan nilai n: 23
23 70 35 106 53 160 80 40 20 10 5 16 8 4 2 1
PS D:\Laprak Algoritma dan Pemrograman 2\modul 4 Prosedur>

```

Deskripsi Program

Sintaks tersebut adalah program untuk mencetak deret bilangan berdasarkan aturan jika bilangan saat ini adalah genap, suku berikutnya adalah $n / 2$, Jika bilangan saat ini adalah ganjil, suku berikutnya adalah $3n + 1$, Deret akan berhenti ketika mencapai nilai 1. Pertama kita membuat prosedur cetak deret dengan n sebagai parameter formal, prosedur tersebut untuk mencetak deret bilangan berdasarkan aturan yang diberikan. Cara kerjanya program mencetak nilai awal n . Selanjutnya, program memasuki loop yang akan terus berjalan hingga nilai n menjadi 1, Pada setiap perulangan Jika n adalah bilangan genap maka nilai n dibagi 2, dan jika n adalah bilangan ganjil, nilai n diperbarui menjadi $3n+1$.