

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 4
PROSEDUR**



Disusun Oleh:

Boutefhika Nuha Ziyadatul Khair/2311102316

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 merupakan sub program yang melakukan satu atau lebih proses tanpa mengembalikan nilai dari hasil dari proses yang dijalankan. Namun, pada prosedur diperbolehkan mencetak hasil luaran yang dihasilkan dari proses yang dijalankan.

A. Konsep Prosedur

Sub Program dengan jenis prosedur merupakan bagian dari kode program yang terpisah yang melakukan tugas spesifik dimana hasilnya dapat diketahui setelah semua proses didalamnya selesai dikerjakan. Prosedur perlu dipanggil dalam program utama agar hasil akhir dari sebuah prosedur dapat diketahui. Prosedur dapat dipanggil secara berulang kali di program utama. Penggunaan prosedur biasanya ditandai dengan penggunaan void.

B. Parameter Prosedur

Ketika sebuah prosedur dipanggil, maka pada dasarnya dapat melakukan proses pertukaran data antara program utama dan sub program. Proses pertukaran ini dapat dilakukan dengan penggunaan parameter. Pada prosedur terdapat parameter yaitu sebagai berikut:

1. Parameter aktual

Parameter aktual merupakan parameter yang disertakan pada saat prosedur dipanggil untuk dijalankan pada program utama, atau dapat sering jenis ini sebagai argument

2. Parameter formal

Parameter formal merupakan parameter yang dituliskan pada saat melakukan pendefinisian sebuah prosedur. Parameter ini ditulis pada bagian kode sub program untuk menerima nilai dari parameter aktual pada program utama.

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.

C. Pemanggilan Parameter pada Prosedur

Banyaknya parameter tidak dibatasi jumlahnya, namun dalam implementasi prosedur yang baik, jumlah parameter dalam sebuah prosedur sebaiknya paling banyak adalah tujuh buah. Jika parameter terlalu banyak, maka sebaiknya ada pemisahan lagi menjadi beberapa prosedur. Pemanggilan nama prosedur dapat dilakukan dengan menuliskan nama prosedur yang terdefiniskan. Pemanggilan terhadap prosedur dapat juga menambahkan nilai parameter aktual (argumen) untuk diproses pada prosedur yang menggunakan parameter formal. Urutan nilai parameter aktual (argumen) harus sama dengan dengan urutan pada parameter formal, karena tipe pada parameter aktual (argumen) dan parameter formal harus sama. Berikut ini merupakan contoh gambaran dari sebuah pseudocode untuk pemanggilan sebuah prosedur. Contoh:

```
Algoritma: Prosedur hitungKali {Penerapan algoritma dari sebuah perkalian
pada dua buah bilangan}

PROCEDURE hitungKali (input bil1, bil2: integer)
Deklarasi
hasil : integer
Deskripsi
hasil ← bil1*bil2
PRINT (hasil)

ALGORITMA UTAMA
Deklarasi
bilangan1, bilangan2: integer
Prosedure hitungJumlah (input bil1, bil2: integer)
Deskripsi
Input (bilangan1, bilangan2)
hitungKali (bilangan1, bilangan2)
```

Pada pseudocode diatas, variabel bilangan1 dan bilangan2 merupakan sebuah parameter aktual dari sebuah prosedur hitungJumlah(). Pada saat pemanggilan prosedur, dapat dilakukan dengan cara menuliskan nama prosedur yang sudah didefinisikan dan menambahkan nilai parameter aktual untuk dikirimkan ke parameter formal. Tipe data parameter formal yang dikirimkan harus memiliki tipe yang sama dengan parameter formal. Jika jumlah parameter aktual lebih dari satu, maka urutan pada parameter aktual yang dikirimkan harus sama dengan urutan

pada parameter formal yang menerimanya. Pada contoh diatas bil1 dan bil2 merupakan parameter formal untuk prosedur hitungKali().

II. GUIDED

1. Buatlah sebuah program menggunakan 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)
    var hasil int
    if a >= b {
        permutasi(a, b, &hasil)
    } else {
        permutasi(b, a, &hasil)
    }
    fmt.Println(hasil)
}

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

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

Screenshoot Program

```
PS C:\Users\ASUS\Documents\Semester3\Modul4>
go run "c:\Users\ASUS\Documents\Semester3\Modul4\Guided\guided1.go"
5 4
120
PS C:\Users\ASUS\Documents\Semester3\Modul4> go run "c:\Users\ASUS\Doc
2 6
30
PS C:\Users\ASUS\Documents\Semester3\Modul4> |
```

Deskripsi Program

Program ini berfungsi untuk menghitung permutasi dari dua angka bulat yang diinput, yaitu `a` dan `b`. Di dalam fungsi `main`, program mulai dengan membaca dua angka dari pengguna dan menyimpannya ke dalam variabel `a` dan `b`. Selanjutnya, program membandingkan kedua angka tersebut supaya yang lebih besar jadi `n` dan yang lebih kecil jadi `r`. Setelah itu, program memanggil fungsi `permutasi` untuk menghitung permutasi dengan rumus $P(n, r) = \frac{n!}{(n-r)!}$. Di dalam fungsi `permutasi`, program menggunakan fungsi `faktorial` untuk menghitung faktorial dari `n` dan `(n-r)`. Fungsi `faktorial` ini bekerja dengan cara mengalikan semua angka dari 1 hingga `n`. Setelah semua perhitungan selesai, hasil permutasi disimpan di variabel `hasil` dan ditampilkan ke layar.

2. 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"

// Prosedur untuk menghitung luas persegi
func hitungLuas(sisi int, luas *int) {
    *luas = sisi * sisi
}

// Prosedur untuk menghitung keliling persegi
func hitungKeliling(sisi int, keliling *int) {
    *keliling = 4 * sisi
}

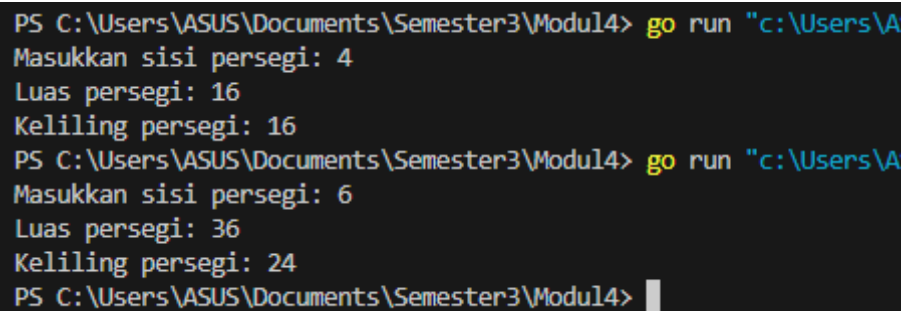
func main() {
```

```
var sisi int
fmt.Print("Masukkan sisi persegi: ")
fmt.Scan(&sisi)

var luas, keliling int
hitungLuas(sisi, &luas)
hitungKeliling(sisi, &keliling)

fmt.Println("Luas persegi:", luas)
fmt.Println("Keliling persegi:", keliling)
}
```

Screenshoot Program



```
PS C:\Users\ASUS\Documents\Semester3\Modul4> go run "c:\Users\A
Masukkan sisi persegi: 4
Luas persegi: 16
Keliling persegi: 16
PS C:\Users\ASUS\Documents\Semester3\Modul4> go run "c:\Users\A
Masukkan sisi persegi: 6
Luas persegi: 36
Keliling persegi: 24
PS C:\Users\ASUS\Documents\Semester3\Modul4> █
```

Deskripsi Program

Program diatas menghitung luas dan keliling dari sebuah persegi berdasarkan input sisi yang diberikan pengguna. Pertama, di dalam fungsi **`main`**, program meminta pengguna untuk memasukkan panjang sisi persegi dan menyimpannya dalam variabel **`sisi`**. Selanjutnya, program mendeklarasikan dua variabel, **`luas`** dan **`keliling`**, untuk menyimpan hasil perhitungan. Program kemudian memanggil prosedur **`hitungLuas`** dengan sisi yang diinput dan alamat dari variabel **`luas`**, yang akan menghitung luas dengan cara mengalikan sisi dengan dirinya sendiri. Setelah itu, prosedur **`hitungKeliling`** dipanggil untuk menghitung keliling dengan mengalikan sisi dengan 4, dan hasilnya disimpan dalam variabel **`keliling`**. Setelah semua perhitungan selesai, program mencetak hasilnya ke layar, menampilkan luas dan keliling persegi berdasarkan sisi yang telah dimasukkan.

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!

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

Sourcecode

```
package main

import (
    "fmt"
)

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

// Prosedur permutation menghitung P(n, r) = n! / (n - r)!
func permutation(n, r int, hasil *int) {
    var fn, fnr int
    factorial(n, &fn)           // Menghitung n!
    factorial(n-r, &fnr)        // Menghitung (n - r)!
    *hasil = fn / fnr           // Hasil permutasi
}

// Prosedur combination menghitung C(n, r) = n! / (r! * (n - r)!)
func combination(n, r int, hasil *int) {
    var fn, fr, fnr int
    factorial(n, &fn)           // Menghitung n!
```

```

        factorial(r, &fr)          // Menghitung r!
        factorial(n-r, &fnr)       // Menghitung (n - r)!
        *hasil = fn / (fr * fnr) // Hasil kombinasi
    }

    func main() {
        var a, b, c, d int
        var p1, c1, p2, c2 int

        fmt.Print("Masukkan nilai a, b, c, d: ")
        fmt.Scan(&a, &b, &c, &d)

        // Memvalidasi kondisi: a >= c dan b >= d
        if a < c || b < d {
            fmt.Println("Syarat tidak terpenuhi: a harus
            >= c dan b harus >= d.")
            return
        }

        // Menghitung permutasi dan kombinasi
        permutation(a, c, &p1)
        combination(a, c, &c1)
        permutation(b, d, &p2)
        combination(b, d, &c2)

        // Menampilkan hasil
        fmt.Printf("P(%d, %d) = %d\n", a, c, p1)
        fmt.Printf("C(%d, %d) = %d\n", a, c, c1)
        fmt.Printf("P(%d, %d) = %d\n", b, d, p2)
        fmt.Printf("C(%d, %d) = %d\n", b, d, c2)
    }

```

Screenshoot Output

```

PS C:\Users\ASUS\Documents\Semester3\Modul4> go run "c:\Users\A
SUS\Documents\Semester3\Modul4\Unguided\unguided1.go"
Masukkan nilai a, b, c, d: 5 10 3 10
P(5, 3) = 60
C(5, 3) = 10
P(10, 10) = 3628800
C(10, 10) = 1
PS C:\Users\ASUS\Documents\Semester3\Modul4> go run "c:\Users\A
Masukkan nilai a, b, c, d: 8 0 2 0
P(8, 2) = 56
C(8, 2) = 28
P(0, 0) = 1
C(0, 0) = 1

```


Deskripsi Program

Program diatas menghitung permutasi dan kombinasi dari dua pasangan angka yang diinput oleh pengguna. Pertama-tama, di dalam fungsi ``main``, program meminta pengguna untuk memasukkan empat angka: ``a``, ``b``, ``c``, dan ``d``. Setelah itu, program memeriksa apakah syarat ``a` >= `c`` dan ``b` >= `d`` terpenuhi. Jika tidak, program akan memberi tahu pengguna bahwa syarat tidak terpenuhi dan menghentikan eksekusi. Jika syarat terpenuhi, program melanjutkan untuk menghitung permutasi $P(n, r) = \frac{n!}{(n-r)!}$ dan kombinasi $C(n, r) = \frac{n!}{r!(n-r)!}$ menggunakan prosedur ``permutation`` dan ``combination``. Di dalam prosedur ini, program menghitung faktorial dari angka yang diperlukan dan kemudian menghitung hasilnya berdasarkan rumus yang sesuai. Setelah semua perhitungan selesai, program mencetak hasil permutasi dan kombinasi untuk kedua pasangan angka ke layar.

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 game 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.

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"

// Prosedur hitungSkor menghitung berapa banyak soal yang
diselesaikan dan berapa total waktu yang dibutuhkan
func hitungSkor(waktu [8]int, soal *int, skor *int) {
    *soal = 0
    *skor = 0
    for i := 0; i < 8; i++ {
        if waktu[i] <= 300 { // Waktu lebih dari 300
            dianggap gagal
            *soal++
            *skor += waktu[i]
        }
    }
}

func main() {
    var nama, pemenang string
    var waktu [8]int
    var soal, skor, soalMax, skorMin int

    soalMax = 0
    skorMin = 301 * 8 // Nilai skor maksimal yang mungkin
    (301 menit per soal)

    for {
        // Baca nama peserta
        fmt.Scan(&nama)
        if nama == "Selesai" {
            break
        }

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

        // Hitung jumlah soal yang diselesaikan dan total
        skor
        hitungSkor(waktu, &soal, &skor)

        // Tentukan pemenang berdasarkan soal yang
        diselesaikan dan skor
    }
}
```

```

        if soal > soalMax || (soal == soalMax && skor <
skorMin) {
            pemenang = nama
            soalMax = soal
            skorMin = skor
        }
    }

    // Cetak pemenang, jumlah soal yang diselesaikan,
    dan total skor
    fmt.Println(pemenang, soalMax, skorMin)
}

```

Screenshoot Program

```

PS C:\Users\ASUS\Documents\Semester3\Modul4>
go run "c:\Users\ASUS\Documents\Semester3\Modul4\Unguided\unguided2.go"
Astuti 20 50 301 301 61 71 75 10
Bertha 25 47 301 26 50 60 65 21
Selesai
Bertha 7 294
PS C:\Users\ASUS\Documents\Semester3\Modul4>

```

Deskripsi Program

Program diatas digunakan untuk menentukan pemenang dalam sebuah kompetisi pemrograman berdasarkan jumlah soal yang diselesaikan dan total waktu yang dibutuhkan peserta. Di dalam fungsi `'main'`, program mulai dengan mendeklarasikan beberapa variabel, termasuk nama peserta, waktu penyelesaian untuk 8 soal, serta variabel untuk menyimpan informasi pemenang. Program kemudian menggunakan loop untuk terus membaca nama peserta dan waktu penyelesaian soal hingga pengguna memasukkan kata **"Selesai"**. Untuk setiap peserta, program membaca waktu penyelesaian untuk 8 soal dan memanggil prosedur `'hitungSkor'`, yang menghitung berapa banyak soal yang diselesaikan dalam waktu 300 menit atau kurang, serta total waktu yang digunakan. Setelah itu, program membandingkan jumlah soal yang diselesaikan dan total skornya untuk menentukan pemenang, dengan kriteria soal terbanyak diselesaikan, dan jika sama, waktu terpendek. Ketika semua peserta sudah diproses, program mencetak nama pemenang, jumlah soal yang diselesaikan, dan total waktu yang digunakan.

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.

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.

Sourcecode

```
package main

import "fmt"

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

// Prosedur cetakDeret menerima n sebagai input
(parameter in)
func cetakDeret(n int) {
    for n > 1 {
        fmt.Print(n, " ")
        if n%2 == 0 {
            n = n / 2 // Jika genap
        } else {
            n = 3*n + 1 // Jika ganjil
        }
    }
    fmt.Print(n) // Mencetak 1 di akhir deret
}
```

Screenshoot Program

```
PS C:\Users\ASUS\Documents\Semester3\Modul4> go run "c:\Users\ASU
22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS C:\Users\ASUS\Documents\Semester3\Modul4> |
```

Deskripsi Program

Program ini dibuat untuk mencetak deret bilangan. Di dalam fungsi ``main``, program mulai dengan meminta pengguna untuk memasukkan sebuah angka bulat ``n``. Setelah itu, program memanggil prosedur ``cetakDeret``, yang menerima angka tersebut sebagai input. Di dalam prosedur ini, program menggunakan loop untuk mencetak nilai ``n`` selama ``n`` lebih besar dari 1. Jika ``n`` genap, program membagi ``n`` dengan 2, jika ``n`` ganjil, program menghitung nilai baru dengan rumus $3n+1$. Proses ini diulang hingga ``n`` bernilai 1, yang kemudian juga dicetak di akhir deret.

IV. KESIMPULAN

Prosedur dalam pemrograman adalah subprogram yang digunakan untuk melakukan tugas-tugas spesifik tanpa mengembalikan nilai langsung, sering kali hanya mencetak hasilnya. Prosedur membantu membagi program menjadi bagian-bagian yang lebih kecil dan modular, membuat kode lebih mudah dipahami dan dipelihara. Dalam prosedur, data dapat diteruskan melalui parameter, baik menggunakan **pass by value** (hanya menyalin nilai) atau **pass by reference** (mengakses langsung memori asli). Penggunaan prosedur ini meningkatkan efisiensi dalam menangani tugas-tugas yang berulang, seperti menghitung permutasi, kombinasi, atau menghitung faktorial, dengan memisahkan logika kompleks ke dalam bagian-bagian yang lebih sederhana dan dapat digunakan kembali.

V. REFERENSI

- [1] **Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi Republik Indonesia.** (2024). *Modul PB 10 - Prosedur*. Diakses dari <https://lmsspada.kemdikbud.go.id>.