

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 4
PROSEDUR**



Disusun Oleh :

ANDIKA NEVIANTORO / 2311102167

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

Definisi Prosedur

Prosedur adalah bagian dari suatu program yang disusun secara terpisah untuk melakukan suatu tugas khusus atau fungsi tertentu.

Pada dasarnya ada dua macam bentuk prosedur yaitu Subrutin (subprogram) dan Fungsi.

1. Subrutin (subprogram) adalah bagian dari program yang dibuat terpisah untuk melaksanakan sebagian dari tugas yang harus diselesaikan oleh suatu program. Pada umumnya yang dikenal dengan nama “Prosedur” sebenarnya adalah “Subprogram” .
2. Fungsi adalah bagian dari program yang dibuat terpisah untuk melaksanakan fungsi tertentu yang menghasilkan suatu nilai untuk dikembalikan ke program utama.

Suatu subprogram dikatakan prosedur apabila:

- Membentuk suatu instruksi baru
- Nama fungsi membentuk kata kerja
- Tidak memiliki tipe data
- Tidak terdapat “return”
- Tidak dapat menggunakan parameter secara langsung

Parameter

1. Parameter aktual

Merupakan variabel atau ekspresi dalam parameter yang digunakan untuk memanggil sebuah sub program

Misal: raise_salary(emp_nim.amount)

2. Parameter formal

Merupakan variabel yang ada didalam parameter list pada sebuah program

Misal: raise_salary emp_id INTEGER, amount REAL

Parameter juga dikelompokkan berdasarkan alokasi memorinya, yaitu pass by value dan pass by refrence.

1. Pass by value

Nilai ke subprogram kemudian lanjut kedalam variabel namun tidak dapat dikembalikan

-Syntax deklarasi:

```
Func kali_dua(x int) int{  
    }  
}
```

-Syntax pemanggilan

```
Kali.dua(x)
```

2. Pass by reference

Langsung dapat diproses dan langsung dikembalikan

-Syntax deklarasi

```
Func kali_dua(x *int){  
    }  
}
```

-Syntax pemanggilan

```
Kali_dua(&x)
```

II. GUIDED

1. Buatlah sebuah program beserta 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, hasil int
    fmt.Scan(&a, &b)

    if a >= b {
        permutasi(a, b, &hasil)
    } else {
        permutasi(b, a, &hasil)
    }

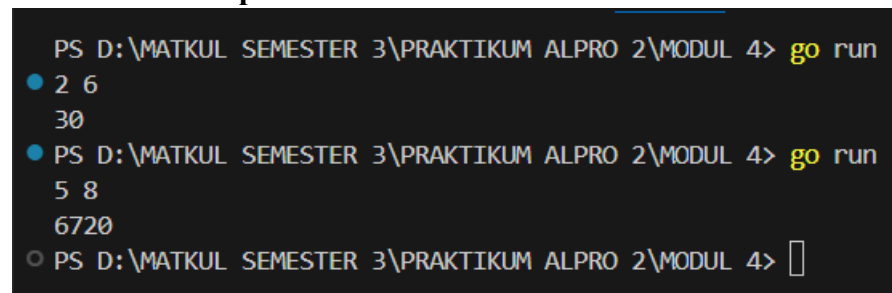
    fmt.Println(hasil)
}

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

func permutasi(n, r int, hasil *int) {
    var faktorialN, faktorialNR int
    faktorial(n, &faktorialN) // hitung faktorial(n)
    faktorial(n-r, &faktorialNR) // hitung faktorial(n-r)

    *hasil = faktorialN / faktorialNR
}
```

Screenshoot Output :



```
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4> go run
2 6
30
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4> go run
5 8
6720
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4> 
```

Deskripsi Program :

Program ini menghitung permutasi dua bilangan yang dimasukkan pengguna. Pertama, bilangan yang lebih besar dipilih sebagai n dan yang lebih kecil sebagai r . Fungsi `permutasi` kemudian menghitung permutasi $P(n, r) = \frac{n!}{(n-r)!}$ dengan memanggil fungsi *faktorial* untuk menghitung faktorial dari n dan $n-r$. Hasil akhirnya, yaitu jumlah cara mengatur r objek dari n objek, dicetak 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 luasPersegi(sisi float64, luas *float64) {
    *luas = sisi * sisi
}

// Prosedur untuk menghitung keliling persegi
func kelilingPersegi(sisi float64, keliling *float64) {
    *keliling = 4 * sisi
}

func main() {
    var sisi, luas, keliling float64

    // Meminta input dari user
    fmt.Print("Masukkan panjang sisi persegi: ")
    fmt.Scan(&sisi)

    // Menghitung luas dan keliling menggunakan prosedur
    luasPersegi(sisi, &luas)
    kelilingPersegi(sisi, &keliling)

    // Menampilkan hasil
    fmt.Printf("Luas persegi: %.2f\n", luas)
    fmt.Printf("Keliling persegi: %.2f\n", keliling)
}
```

Screenshoot Output :

```

● PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4> go ru
Masukkan panjang sisi persegi: 10
Luas persegi: 100.00
Keliling persegi: 40.00
○ PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4> █

```

Deskripsi Program :

Program ini menghitung luas dan keliling sebuah persegi berdasarkan input panjang sisi dari pengguna. Pertama, pengguna diminta memasukkan panjang sisi persegi, kemudian nilai tersebut diproses menggunakan dua prosedur: *luasPersegi* dan *kelilingPersegi*. Prosedur *luasPersegi* menghitung luas persegi dengan rumus ***luas = sisi²***, sementara *kelilingPersegi* menghitung keliling dengan rumus ***keliling = 4 x sisi***. Hasil perhitungan luas dan keliling disimpan melalui pointer dan kemudian ditampilkan ke layar dengan format dua angka desimal. Program ini memanfaatkan konsep parameter pointer untuk menyimpan hasil secara langsung dalam variabel yang dipakai di *main*.

2. UNGUIDED

1. **Masukan** terdiri dari empat buah bilangan asli a, b, c, dan d yang dipisahkan oleh spasi, dengan syarat $a < b$ dan $c \geq d$.

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"
)

// Fungsi untuk menghitung faktorial dari suatu angka
func faktorial(n int) int {

```

```

    if n == 0 {
        return 1
    }
    hasil := 1
    for i := 1; i <= n; i++ {
        hasil *= i
    }
    return hasil
}

// Fungsi untuk menghitung permutasi  $P(n, r) = n! / (n-r)!$ 
func permutasi(n, r int) int {
    return faktorial(n) / faktorial(n-r)
}

// Fungsi untuk menghitung kombinasi  $C(n, r) = n! / (r! * (n-r)!)$ 
func kombinasi(n, r int) int {
    return faktorial(n) / (faktorial(r) * faktorial(n-r))
}

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

    if a >= c && b >= d {
        // Baris pertama output: Permutasi dan Kombinasi
        // a terhadap c
        perm1 := permutasi(a, c)
        komb1 := kombinasi(a, c)
        fmt.Printf("%d %d\n", perm1, komb1)

        // Baris kedua output: Permutasi dan Kombinasi
        // b terhadap d
        perm2 := permutasi(b, d)
        komb2 := kombinasi(b, d)
        fmt.Printf("%d %d\n", perm2, komb2)
    } else {
        fmt.Println("Nilai input tidak memenuhi syarat yang diperlukan.")
    }
}

```

Screenshoot Output :

```
● PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4> go r
5 10 3 10
60 10
3628800 1
● PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4> go r
8 0 2 0
56 28
1 1
○ PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4> □
```

Deskripsi Program :

Program ini bertujuan menghitung permutasi dan kombinasi dari dua pasang bilangan input pengguna, yaitu (a, c) dan (b, d), dengan syarat $a \geq c$ dan $b \geq d$. Jika syarat terpenuhi, program menghitung permutasi $P(n, r)$ dan kombinasi $C(n, r)$ menggunakan fungsi faktorial, kemudian menampilkan hasilnya. Jika tidak, program memberi pesan bahwa input tidak memenuhi syarat.

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.

prosedure `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.

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"
)

// Prosedur untuk menghitung skor seorang peserta
func hitungSkor(nama string, waktu []int) (soal int, skor int) {
    for i := 0; i < 8; i++ {
        if waktu[i] <= 300 {
            soal++
            skor += waktu[i]
        } else {
            skor += 301 // Jika soal tidak dikerjakan
            // atau melebihi waktu 5 jam 1 menit
        }
    }
    return
}

func main() {
    var nama1, nama2 string
    var waktu1, waktu2 [8]int

    // Input data peserta pertama
    fmt.Scan(&nama1)
    for i := 0; i < 8; i++ {
        fmt.Scan(&waktu1[i])
    }

    // Input data peserta kedua
    fmt.Scan(&nama2)
    for i := 0; i < 8; i++ {
        fmt.Scan(&waktu2[i])
    }

    // Hitung soal dan skor peserta pertama
    soal1, skor1 := hitungSkor(nama1, waktu1[:])

    // Hitung soal dan skor peserta kedua
    soal2, skor2 := hitungSkor(nama2, waktu2[:])

    // Tentukan pemenang berdasarkan jumlah soal dan skor
    var pemenang string
```

```

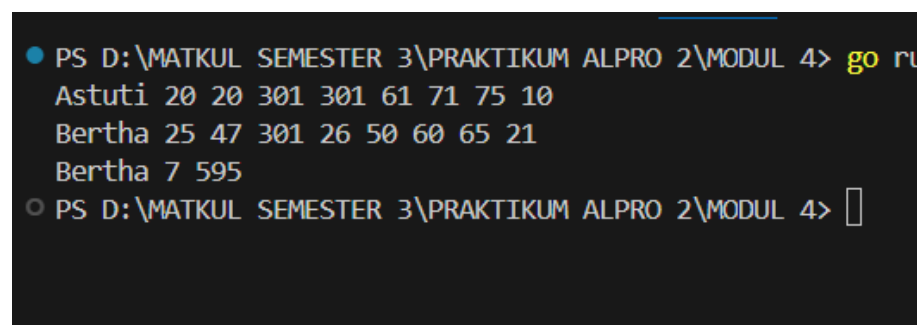
        var soalPemenang, skorPemenang int

        if soal1 > soal2 || (soal1 == soal2 && skor1 < skor2)
        {
            pemenang = nama1
            soalPemenang = soal1
            skorPemenang = skor1
        } else {
            pemenang = nama2
            soalPemenang = soal2
            skorPemenang = skor2
        }

        // Output pemenang, jumlah soal, dan total skor
        fmt.Printf("%s      %d      %d\n",
strings.TrimSpace(pemenang),          soalPemenang,
skorPemenang)
    }

```

Screenshoot Output :



```

● PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4> go run main.go
Astuti 20 20 301 301 61 71 75 10
Bertha 25 47 301 26 50 60 65 21
Bertha 7 595
○ PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4>

```

Deskripsi Program :

Program ini dirancang untuk menghitung skor dan menentukan pemenang dari dua peserta dalam suatu kompetisi berdasarkan waktu yang mereka habiskan untuk menyelesaikan delapan soal. Pertama, program meminta input nama dan waktu yang digunakan oleh masing-masing peserta untuk menyelesaikan setiap soal. Fungsi *hitungSkor* kemudian menghitung jumlah soal yang dikerjakan dalam waktu kurang dari atau sama dengan 300 detik, serta total skor, di mana waktu yang melebihi batas tersebut dihitung sebagai 301 detik. Setelah menghitung skor untuk kedua peserta, program membandingkan jumlah soal yang diselesaikan; jika jumlah soal sama, pemenang ditentukan berdasarkan skor yang lebih rendah (waktu lebih cepat). Akhirnya, program mencetak nama pemenang, jumlah soal yang dikerjakan, dan total skor. Tujuan dari program ini adalah untuk secara objektif menentukan peserta yang berprestasi terbaik berdasarkan kriteria waktu dan penyelesaian soal.

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 $n/2$, 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 dipisahkan oleh sebuah spasi.

Sourcecode :

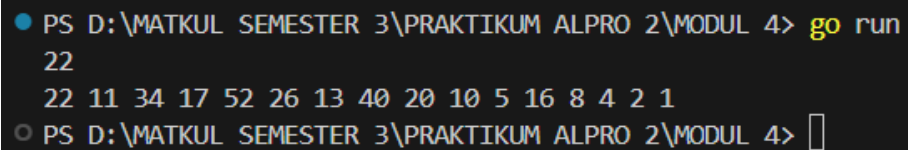
```
package main

import (
    "fmt"
)

// Prosedur untuk mencetak deret Skiena berdasarkan
aturan yang diberikan
func cetakDeret(n int) {
    for n != 1 {
        fmt.Printf("%d ", n)
        if n%2 == 0 {
            n = n / 2 // Jika n genap, bagi dua
        } else {
            n = 3*n + 1 // Jika n ganjil, kalikan 3 dan
            tambahkan 1
        }
    }
    fmt.Println(1) // Cetak 1 sebagai suku terakhir
}
```

```
func main() {
    var n int
    fmt.Scan(&n) // Baca input dari pengguna
    cetakDeret(n) // Panggil prosedur untuk mencetak
    deret
}
```

Screenshoot Output :



```
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4> go run
22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 4> █
```

Deskripsi Program :

Program ini menghitung dan mencetak deret Skiena, yang merupakan salah satu variasi dari masalah Collatz, berdasarkan aturan tertentu. Setelah pengguna memasukkan bilangan bulat positif n , fungsi *cetakDeret* akan dieksekusi. Dalam fungsi ini, program terus mencetak nilai n selama n tidak sama dengan 1. Jika n genap, nilainya dibagi dua, sedangkan jika n ganjil, nilai baru dihitung dengan rumus $n = 3n + 1$. Proses ini berlanjut hingga mencapai angka 1, yang juga dicetak sebagai suku terakhir dari deret. Tujuan dari program ini adalah untuk menunjukkan bagaimana deret Skiena berkembang dari nilai awal n hingga mencapai 1, sekaligus memberikan ilustrasi tentang perilaku angka dalam konteks rumus matematis yang sederhana namun menarik.

KESIMPULAN

Kesimpulan dari penjelasan di atas adalah bahwa prosedur dalam pemrograman merupakan bagian terpisah dari suatu program yang dirancang untuk melaksanakan tugas khusus. Prosedur dapat dibedakan menjadi dua jenis, yaitu subrutin (subprogram) dan fungsi. Subrutin berfungsi untuk melaksanakan sebagian tugas tanpa menghasilkan nilai kembali, sedangkan fungsi menghasilkan nilai yang dapat dikembalikan ke program utama.

Prosedur dikategorikan sebagai prosedur jika memenuhi beberapa kriteria, seperti membentuk instruksi baru dan tidak memiliki tipe data atau pernyataan "return." Parameter dalam prosedur juga dibagi menjadi parameter aktual, yang merupakan variabel atau ekspresi saat

pemanggilan, dan parameter formal, yang didefinisikan dalam daftar parameter prosedur.

Selain itu, parameter dapat dikelompokkan berdasarkan alokasi memorinya menjadi pass by value, di mana nilai dikirim tanpa dapat dikembalikan, dan pass by reference, di mana alamat memori variabel dapat diproses dan dikembalikan. Dengan pemahaman ini, programmer dapat merancang dan mengelola struktur program yang lebih efisien dan modular.

REFERENSI

[1] Studocu. Perbedaan function dengan procedure pada golang. Universitas Telkom:

<https://www.studocu.com/id/document/universitas-telkom/algorithm-and-programming/perbedaan-function-dengan-procedure-pada-golang/31410467>

[2] A, Wisnu. Prosedur dan Fungsi. Github Pages :

<https://wisnuagung.github.io/mkdocs-fix/menu/php/prosedurFungsi/>