

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 4
PROSEDUR**



Disusun Oleh :

Loisa Vanica Saragih/2311102280

S1 IF11-05

Dosen Pengampu :

Arif Amrulloh, S.Kom., M.Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pengertian Prosedur

1. Menurut Mulyadi (2016:4) “Prosedur adalah suatu urutan kegiatan klerikal biasanya melibatkan beberapa orang dalam satu departemen atau lebih, yang dibuat untuk menjamin penanganan secara seragam transaksi perusahaan yang terjadi berulang-ulang”.
2. Menurut Rifka R.N (2017:75) “Prosedur adalah urutan kerja atau kegiatan yang terencana untuk menangani pekerjaan yang berulang dengan cara seragam atau terpadu”.
3. Menurut Asriel dkk (2016:25) “Prosedur adalah serangkaian dari tahapan-tahapan atau urutan dari langkah-langkah yang saling terkait dalam menyelesaikan suatu pekerjaan. Untuk mengendalikan pelaksanaan kerja agar efisiensi perusahaan tercapai dengan baik dibutuhkan sebuah petunjuk tentang prosedur kerja”.

Secara ringkas, dari keempat definisi prosedur di atas dapat disimpulkan bahwa prosedur adalah tata cara atau langkah-langkah yang dilakukan secara sistematis dan berurutan dalam mencapai suatu tujuan yang ditetapkan. Merupakan suatu rangkaian kegiatan yang berurutan dan sebagian besarnya meng Incorporating o berinteraksi dengan beberapa individu dalam satu departemen atau lebih, yang disiptakan untuk memastikan pengelolaan sama – sama pada tipe transaksi perusahaan yang terjadi secara berkesinambungan.

Prosedur banyak digunakan pada program yang terstruktur karena :

1. Ini merupakan bentuk dari tipe program Modular, yakni membagi suatu program yang kompleks menjadi beberapa program yang lebih skipe komponen lebih kecil dalam bentuk Proyek.
2. Namun demikian, untuk hal-hal yang dilakukan berulang kali, hanya perlu ditulis sekali saja dalam prosedur dan dapat dipanggil atau dipergunakan, jika diperlukan.
3. Membuat kode program lebih mudah dibaca.
4. Konon dapat digunakan untuk menyembunyikan detail program

Parameter dalam Prosedur

Ada dua jenis parameter:

Parameter Aktual: Tarif yang diterapkan pada saat prosedur dipanggil.

Parameter Formal: untuk menerima nilai dari parameter aktual di dalam prosedur Variabel Di definisikan

II. GUIDED

1. Menghitung Faktorial dan Permutasi dengan menggunakan Prosedur

Sourcecode

```
package main

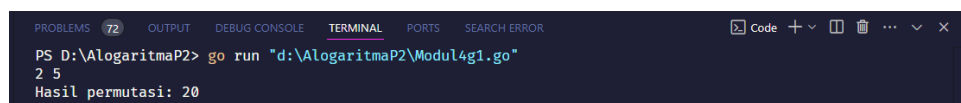
import "fmt"

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

func faktorial(n int, hasil *int) {
    //menggunakan pointer agar dapat mengembalikan nilai
    tanpa return
    *hasil = 1
    for i := 1; i <= n; i++ {
        *hasil *= i
    }
}

func permutasi(n, r int) {
    var hasiln, hasilnr int
    faktorial(n, &hasiln)
    faktorial(n-r, &hasilnr)
    fmt.Println("Hasil permutasi:", hasiln/hasilnr)
}
```

Screenshot Output



```
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul4g1.go"
2 5
Hasil permutasi: 20
PS D:\AlogaritmaP2>
```

Deskripsi Program

Program di atas ditulis dalam bahasa Go dan berguna untuk menghitung permutasi dari dua bilangan yang diinput oleh pengguna. Program ini mengimpor package bernama “fmt” untuk membaca nilai input serta menampilkan nilai output. Pendenginan pertama terdiri dari fungsi `main()` di mana dua bilangan, `a`, dan `b` ditetapkan nilai dari input pengguna. Contohnya, jika `a` lebih besar atau setara dengan `b`, maka

program akan mengunjuk fungsi `permutasi(a, b)`, dan jika tidak, maka program tersebut akan mengunjuk “`permutasi(b, a)`”. Prosedur yang sangat penting dalam pemrograman berikutnya ini adalah fungsi `faktorial()`, fungsi yang digunakan untuk menghitung faktorial dari bilangan tertentu, dan hasilnya disimpan dalam pointer agar tidak menggunakan `return` lagi. Di fungsi `permutasi(n, r)`, faktorial dari n dan (n-r) dihitung dan hasil permutasi diperoleh dengan cara membagi nilai faktorial n dengan faktorial (n-r) dengan rumus permutasi $P(n, r) = \frac{n!}{(n-r)!}$. Sebagai output dari permutasi di atas, hasil akhir permutasi tampil di layar konsol. Misalnya, jika pengguna memasukkan `5` dan `3`, program akan menghitung $\frac{5!}{(5-3)!} = \frac{120}{2} = 60$, dan mencetak hasilnya sebagai "Hasil permutasi: 60".

2. Menghitung luas dan keliling persegi menggunakan prosedur

Sourcecode

```

II      package main
IV
V      import "fmt"
VI
II      func main() {
II          var s int
IX          fmt.Print("Input Sisi: ")
XI          fmt.Scan(&s)
II
II          luasPersegi(s)
II          kelilingPersegi(s)
IV
XV      }
VI      func luasPersegi(s int) {
II          hasil := s * s
II          fmt.Println("Hasil Luas: ", hasil)
IX      }
XX
XI      func kelilingPersegi(s int) {
II          hasil := 4 * s
II          fmt.Print("Hasil keliling: ", hasil)
IV      }

```

Screenshoot Output

```
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul4g2.go"
Input Sisi: 12
Hasil Luas: 144
Hasil keliling: 48
PS D:\AlogaritmaP2>
```

Deskripsi Program

Program ini ditulis dalam bahasa Go dengan tujuan untuk menghitung luas dan keliling persegi dengan meminta input ukuran sisi dari pemakai. Mulai dari penggunaan library `fmt` untuk melakukan perintah input/output, program menentukan variabel integer `s` untuk menyimpan nilai sisi dari persegi. Setelah menampilkan pesan "Input Sisi:Program ini kemudian menggunakan `fmt.Scan(&s)` untuk membaca nilai dari pengguna dan memanggil dua fungsi yaitu `luasPersegi(s)` dan `kelilingPersegi(s)`. Jenis fungsi `luasPersegi` ini digunakan untuk mencari nilai luas dengan rumus. Hasil = $s \times s$ dan menampilkan hasilnya, sementara fungsi `kelilingPersegi` menghitung keliling dengan rumus $4 \times s$, jadi dengan menggunakan program di atas kita dapat menghitung hasil nya dengan mudah. Hasil = $4 \times s$ dan setelah itu mencetak hasilnya. Dengan konsep moduler ini, program menjadi lebih terstruktur, simplistic, readable dan lebih diperbarui. Sebagai contoh, bila nilai sisi yang diinput oleh pengguna adalah 5, maka program akan menghasilkan nilai luas = 25 dan keliling = 20.

II. 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? Masukan terdiri dari empat buah bilangan asli a, b, c, dan d yang dipisahkan oleh spasi, dengan syarat $a \leq c$ dan $b \leq d$. 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 ($n \geq r$) dapat dihitung dengan menggunakan persamaan berikut!

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

Contoh

No	Masukan	Keluaran	Penjelasan
1	5 10 3 10	60 10 3628800 1	$P(5, 3) = 5!/2! = 120/2 = 60$ $C(5, 3) = 5!/(3! \times 2!) = 120/12 = 10$ $P(10, 10) = 10!/0! = 3628800/1 = 3628800$ $C(10, 10) = 10!/(10! \times 0!) = 10!/10! = 1$
2	8 0 2 0	56 28 1 1	

Selesaikan program tersebut dengan memanfaatkan prosedur yang diberikan berikut ini!

```

procedure factorial(in n: integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n
 F.S. hasil berisi nilai faktorial dari n}

procedure permutation(in n, r : integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n dan r, dan  $n \geq r$ 
 F.S. hasil berisi nilai dari n permutasi r}

procedure combination(in n, r : integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n dan r, dan  $n \geq r$ 
 F.S. hasil berisi nilai dari n kombinasi r}

```

Sourcecode

```

package main

import "fmt"

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

```

```

    }
}

func permutation(n, r int, hasil *int) {
    var fn, fnr int
    factorial(n, &fn)
    factorial(n-r, &fnr)
    *hasil = fn / fnr
}

func combination(n, r int, hasil *int) {
    var fn, fr, fnr int
    factorial(n, &fn)
    factorial(r, &fr)
    factorial(n-r, &fnr)
    *hasil = fn / (fr * fnr)
}

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

    if a*c >= d && b >= d {
        var permAC, combAC, permBD, combBD int
        permutation(a, c, &permAC)
        combination(a, c, &combAC)
        permutation(b, d, &permBD)
        combination(b, d, &combBD)

        fmt.Println(permAC, combAC)
        fmt.Println(permBD, combBD)
    } else {
        fmt.Println("Syarat tidak terpenuhi.")
    }
}

```

Screenshoot Output

```

PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul4ug1.go"
8 0 2 0
56 28
1 1
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul4ug1.go"
2 5 6 9
Syarat tidak terpenuhi.
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul4ug2.go"

```

Deskripsi Program

Permutasi yaitu faktorial dari bilangan yang lebih besar dikalikan faktorial dari bilangan yang lebih kecil difokuskan pada dua set bilangan masuk. Kombinasi merupakan konten dari bilangan detail dari set bilangan masuk dikalikan konten dari bilangan rinci dari set bilangan kedua dikalikan faktorial dari bilangan detail dari bilangan satu set. Fungsi ini yaitu fungsi factorial menghitung faktorial bilangan n dengan cara mengalikan 1 sampai dengan n, fungsi permutation melakukan perhitungan untuk permutasi $P(n, r)$ dengan rumus $P(n,r) = n!/(n-r)!$, dan fungsi combination menghitung kombinasi $C(n,r)$. Dalam fungsi `main` program tersebut, input empat bilangan a, b, c, dan d dibaca oleh program dan kemudian diperiksa apakah syarat $a*c \geq d$ dan $b \geq d$ terpenuhi atau tidak. Jika ya, program menghitung permutasi dan kombinasi untuk nilai a dengan c dan b dengan d, lalu menekan tampilan hasilnya jika tidak, program menekan tampilan “Syarat tidak terpenuhi.”

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. prosedur 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.

No	Masukan	Keluaran
1	Astuti 20 50 301 301 61 71 75 10 Bertha 25 47 301 26 50 60 65 21 Selesai	Bertha 7 294

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"
func hitungSkor(waktu []int, soal *int, skor *int) {
    for _, w := range waktu {
        if w < 301 {
            *soal++
            *skor += w
        }
    }
}

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

    pemenang, maxSoal, minSkor := "", -1, 99999

    for i := 0; i < peserta; i++ {
        var nama string
        var waktu [8]int
        fmt.Print("Input nama dan waktu: ")
        fmt.Scan(&nama, &waktu[0], &waktu[1], &waktu[2],
            &waktu[3], &waktu[4], &waktu[5], &waktu[6], &waktu[7])

        var totalSoal, totalSkor int
        hitungSkor(waktu[:], &totalSoal, &totalSkor)

        if totalSoal > maxSoal || (totalSoal == maxSoal
            && totalSkor < minSkor) {
            pemenang, maxSoal, minSkor = nama,
                totalSoal, totalSkor
        }
    }

    fmt.Printf("Pemenang: %s\nJumlah soal: %d\nTotal waktu: %d\n", pemenang, maxSoal, minSkor)
}
```

Screenshoot Output

```
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul4ug2.go"
Input jumlah peserta: 2
Input nama dan waktu: Astuti 20 50 301 301 61 71 75 10
Input nama dan waktu: Bertha 25 47 301 26 50 60 65 21
Pemenang: Bertha
Jumlah soal: 7
Total waktu: 294
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul4ug3.go"
```

Deskripsi Program

Program ini mencari pemenang dari beberapa peserta yang menyelesaikan sejumlah soal sederhana dengan waktu yang dibawa untuk menyelesaikan masing-masing soal. HitungSkor mempunyai tugas untuk menghitung jumlah soal-soal yang diatasi oleh peserta dalam waktu di bawah 301 detik dan menambah total waktu yang tersedia. Di fungsi main program input jumlah peserta kemudian untuk setiap peserta nama dan waktu penyelesaian 8 soal disimpan. Program menentukan berapa banyak soal yang selesai dan berapa total waktu semua peserta kemudian memilih pemenang dengan skor soal yang banyak. Seandainya kedua peserta tersebut menj mismatch soal yang sama maka pemenang adalah yang mempunyai total waktu yang lebih kecil. Program kemudian menampilkan nama pemenang, banyaknya soal yang dijawab, dan banyaknya 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 Van , 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 sklena yang akan mencetak setiap suku dari deret yang dijelaskan di atas untuk nilai suku awal yang diberikan. Pencetakan deret harus dibuat dalam prosedur cetak Deret yang mempunyai 1 parameter formal, yaitu nilai dari suku awal. Prosedure cetakDeret(inn: 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

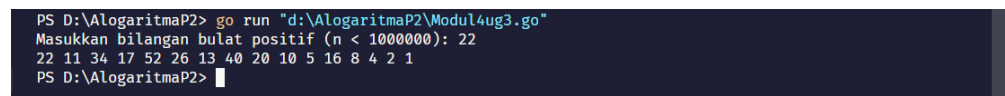
import "fmt"

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

func main() {
    var n int
    fmt.Print("Masukkan bilangan bulat positif (n < 1000000): ")
    fmt.Scan(&n)

    if n > 0 && n < 1000000 {
        cetakDeret(n)
    } else {
        fmt.Println("Input tidak valid. Pastikan n adalah bilangan positif dan kurang dari 1000000.")
    }
}
```

Screenshoot Output



```
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul4ug3.go"
Masukkan bilangan bulat positif (n < 1000000): 22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS D:\AlogaritmaP2>
```

Deskripsi Program

Program ini menggunakan bahasa go dimana untuk bilangan bulat positif yaitu n. Dalam fungsi main program mengambil input bilangan n dari pengguna, dan kemudian mengecek apakah nilai tersebut valid atau tidak, untuk disini nilai yang di pertimbangkan harus positif dan kurang dari 1000000. Jika valid, maka fungsi cetakDeret dipanggil untuk mengeksekusi deret Collatz tersebut. Sebagai fungsi – untuk nilai n tidak sama dengan 1, dalam hal ini angka ini akan dicetak lalu, dalam hal ini apabila n adalah genap maka nilai akan dibagi dua, dan selagi n adalah ganjil, maka nilai tersebut menjadi $3*n + 1$. Ini tetap berlangsung hingga n

menjadi 1, lalu akan dicetak juga. Jika input tidak valid, program menampilkan pesan kesalahan.

IV. Kesimpulan

Note di dalam algoritma pemrograman, prosedur adalah faktor penting dalam pembuatan kode program yang enak untuk dihimpun dalam struktur yang baik dan benar. Mengetahui bagaimana sebuah prosedur beroperasi memungkinkan programmer merancang aplikasi yang lebih modular, rapi dalam pemrograman, dan aplikasi dengan kesalahan yang rendah. Mengenal Prosedur dan Fungsi karena memecahkan suatu program yang kompleks dan besar menjadi beberapa program yang lebih sederhana atau lebih kecil. Untuk kerja yang dijalankan lebih dari sekali / kerap dilakukan beberapa kali. Memperbesar pengenalan kesalahan apabila ada kejadian kesalahan sebenarnya kita hanya perlu mencari fungsi atau prosedur terkait saja tanpa perlu seluruh program.

DAFTAR PUSTAKA

[https://repository.pnj.ac.id/id/eprint/16263/1/Identitas%20Skripsi%20&%20Pembukaan%20\(Bab%201\)%20dan%20Penutup%20\(Bab%205\).pdf](https://repository.pnj.ac.id/id/eprint/16263/1/Identitas%20Skripsi%20&%20Pembukaan%20(Bab%201)%20dan%20Penutup%20(Bab%205).pdf)

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