

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL IV
PROSEDUR**



Disusun Oleh :

Wafiq Nur Azizah / 2311102270

SIIF-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 dalam pemrograman adalah sekumpulan instruksi yang dapat dipanggil dan dijalankan untuk melakukan tugas tertentu. Dalam konteks bahasa pemrograman Go, prosedur sering kali disebut sebagai fungsi (function). Prosedur digunakan untuk mengorganisir kode, memfasilitasi pemrograman yang lebih terstruktur, dan meningkatkan keterbacaan serta pemeliharaan kode. Dengan memisahkan logika program menjadi bagian-bagian kecil yang terpisah, programmer dapat lebih mudah memahami dan mengelola kode yang ditulis.

1. Perbedaan Prosedur dan Fungsi

Meskipun istilah prosedur dan fungsi sering digunakan secara bergantian dalam banyak bahasa pemrograman, ada perbedaan mendasar di antara keduanya. Dalam banyak konteks, fungsi merujuk pada blok kode yang mengembalikan nilai, sedangkan prosedur tidak mengembalikan nilai. Dalam Golang, semua prosedur juga merupakan fungsi, tetapi tidak semua fungsi diharuskan mengembalikan nilai. Fungsi yang tidak mengembalikan nilai dapat dianggap sebagai prosedur, dan sering kali digunakan untuk melakukan aksi, seperti mencetak output atau mengubah nilai variabel.

2. Deklarasi Prosedur

Deklarasi prosedur dalam Golang dimulai dengan kata kunci `func`, diikuti dengan nama prosedur, parameter (jika ada), dan tipe data yang dikembalikan (jika ada). Penamaan prosedur sebaiknya deskriptif dan mencerminkan fungsinya. Misalnya, `func HitungLuas()` dapat digunakan untuk prosedur yang menghitung luas suatu bentuk. Penting untuk diingat bahwa dalam Golang, penamaan fungsi dan prosedur yang dimulai dengan huruf kapital berarti fungsi tersebut dapat diakses dari paket lain, sedangkan yang dimulai dengan huruf kecil hanya dapat diakses dalam paket yang sama.

3. Parameter dan Argumen

Prosedur dapat memiliki parameter, yang merupakan variabel yang ditentukan dalam deklarasi prosedur dan digunakan untuk menerima nilai yang diberikan saat prosedur dipanggil. Dalam Golang, parameter dapat memiliki tipe data yang berbeda, dan beberapa parameter dapat didefinisikan dalam satu prosedur. Ketika prosedur dipanggil, nilai yang diberikan untuk parameter tersebut dikenal sebagai argumen. Penting untuk

memastikan bahwa tipe data argumen sesuai dengan tipe parameter yang didefinisikan dalam prosedur.

4. Mengembalikan Nilai

Prosedur dalam Golang dapat mengembalikan nilai atau tidak. Jika prosedur dirancang untuk mengembalikan nilai, tipe data pengembalian harus ditentukan dalam deklarasi prosedur. Sebuah prosedur yang tidak mengembalikan nilai dapat diakses hanya untuk menjalankan aksi tertentu. Sebagai contoh, prosedur `func TampilkanPesan()` mungkin tidak mengembalikan nilai, sedangkan prosedur `func HitungJumlah(a int, b int) int` mengembalikan jumlah dari dua angka sebagai integer.

5. Scope dan Lifetime Variabel

Variabel yang dideklarasikan di dalam prosedur memiliki lingkup (scope) yang terbatas pada prosedur itu sendiri. Artinya, variabel tersebut tidak dapat diakses di luar prosedur. Prosedur juga dapat menggunakan variabel global yang dideklarasikan di luar prosedur, tetapi sebaiknya ini dilakukan dengan hati-hati untuk menghindari konflik dan meningkatkan keterbacaan kode. Lifetime dari variabel lokal di dalam prosedur berakhir ketika prosedur selesai dieksekusi.

6. Prosedur Rekursif

Prosedur rekursif adalah prosedur yang memanggil dirinya sendiri dalam rangka untuk menyelesaikan tugas tertentu. Rekursi sangat berguna dalam memecahkan masalah yang dapat dibagi menjadi sub-masalah yang lebih kecil. Namun, penting untuk memastikan bahwa ada kondisi dasar (base case) untuk menghentikan rekursi; jika tidak, prosedur dapat menyebabkan stack overflow. Dalam Golang, prosedur rekursif digunakan dalam berbagai algoritma seperti pencarian dan pengurutan.

II. GUIDED

1. Soal Studi Case

Menghitung faktorial dan permutasi menggunakan prosedur.

Sourcecode

```
package main

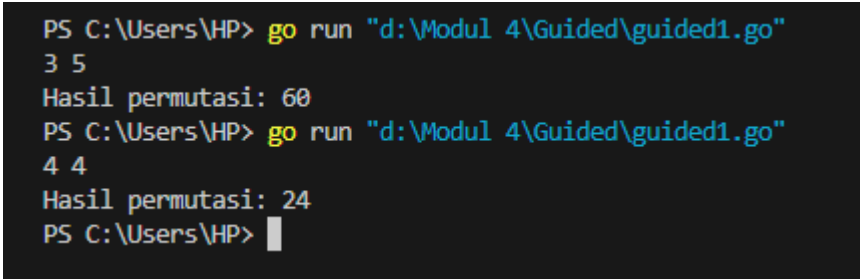
import "fmt"

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) {
    *hasil = 1
    for i := 1; i <= n; i++ { // Perbaikan di sini: `i
    <= n`
        *hasil *= i
    }
}

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

Screenshoot Output



```
PS C:\Users\HP> go run "d:\Modul 4\Guided\guided1.go"
3 5
Hasil permutasi: 60
PS C:\Users\HP> go run "d:\Modul 4\Guided\guided1.go"
4 4
Hasil permutasi: 24
PS C:\Users\HP> █
```

Deskripsi Program

Program Go di atas menghitung permutasi ($P(n, r)$) menggunakan prosedur untuk memfaktorkan dua bilangan yang diberikan oleh pengguna. Program diawali dengan meminta input dua bilangan bulat 'a' dan 'b', lalu mengevaluasi mana yang lebih besar untuk menyesuaikan parameter yang benar bagi fungsi 'permutasi()'. Fungsi ini bertugas menghitung permutasi dengan memanggil prosedur 'faktorial()' dua kali: pertama untuk menghitung ($n!$) dan kedua untuk menghitung $(n - r)!$. Fungsi 'faktorial()' bekerja dengan pendekatan berbasis prosedur menggunakan pointer untuk mengembalikan hasil faktorial dari suatu bilangan. Nilai permutasi kemudian dihitung dengan membagi hasil dari kedua faktorial tersebut dan ditampilkan ke layar. Prosedur 'faktorial()' dan 'permutasi()' memanfaatkan pembagian tugas agar kode lebih modular dan terorganisir.

2. Soal Studi Case

Menghitung luas dan keliling persegi dengan prosedur.

Sourcecode

```
package main

import "fmt"

func main() {
    var s int
    fmt.Print("Input Sisi: ")
    fmt.Scan(&s)

    luasPersegi(s)
    kelilingPersegi(s)
}

func luasPersegi(s int) {
    hasil := s * s
    fmt.Println("Hasil Luas: ", hasil)
}

func kelilingPersegi(s int) {
    hasil := 4 * s
    fmt.Println("Hasil Keliling: ", hasil)
}
```

Screenshoot Output

```
PS C:\Users\HP> go run "d:\Modul 4\Guided\guided2.go"
Input Sisi: 15
Hasil Luas: 225
Hasil Keliling: 60
PS C:\Users\HP> 
```

Deskripsi Program

Program Go di atas menghitung luas dan keliling dari sebuah persegi menggunakan prosedur. Pertama, program meminta pengguna untuk memasukkan panjang sisi persegi melalui input. Setelah nilai sisi diperoleh, program memanggil dua prosedur terpisah: `luasPersegi` untuk menghitung dan menampilkan luas persegi, serta `kelilingPersegi` untuk menghitung dan menampilkan kelilingnya. Kedua prosedur ini menerima parameter berupa panjang sisi (variabel s) dan menghitung hasil menggunakan rumus dasar untuk luas ($s \times s$) dan keliling ($4 \times s$), kemudian mencetak hasilnya. Program ini menekankan penggunaan prosedur yang terpisah untuk setiap perhitungan, sehingga meningkatkan keterbacaan dan modularitas kode.

III. UNGUIDED

1. Soal Studi Case

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 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!

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

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 >= 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 >= 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 && b >= d {
        var permAC, combAC, permBD, combBD int

        permutation(a, c, &permAC)
        combination(a, c, &combAC)
    }
}

```



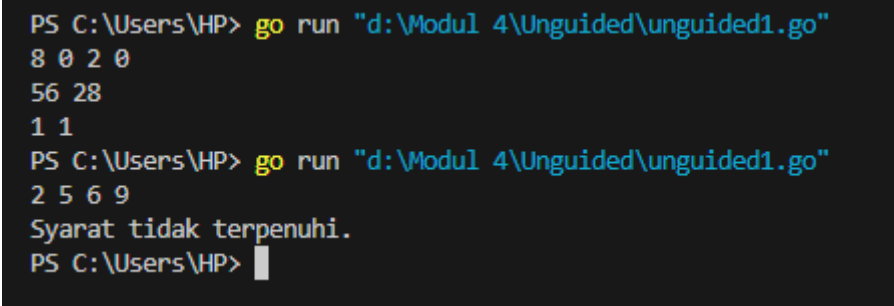
```

        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 C:\Users\HP> go run "d:\Modul 4\Unguided\unguided1.go"
8 0 2 0
56 28
1 1
PS C:\Users\HP> go run "d:\Modul 4\Unguided\unguided1.go"
2 5 6 9
Syarat tidak terpenuhi.
PS C:\Users\HP>

```

Deskripsi Program

Program Go di atas menghitung permutasi dan kombinasi dari dua pasangan angka menggunakan pendekatan prosedural. Pertama, program mendefinisikan fungsi `factorial` untuk menghitung faktorial dari suatu angka yang disimpan dalam parameter pointer. Selanjutnya, terdapat dua fungsi, `permutation` dan `combination`, yang masing-masing menghitung permutasi dan kombinasi berdasarkan nilai yang diberikan, juga menggunakan pointer untuk menyimpan hasilnya. Di dalam fungsi `main`, program meminta input dari pengguna untuk empat angka: a, b, c, dan d. Jika syarat $a \geq c$ dan $b \geq d$ terpenuhi, program akan memanggil fungsi `permutation` dan `combination` untuk menghitung dan menyimpan hasil permutasi dan kombinasi dari kedua pasangan angka, kemudian mencetak hasilnya. Jika syarat tidak terpenuhi, program akan memberikan pesan bahwa syarat tidak terpenuhi.

2. Soal Studi Case

Kompetisi pemrograman tingkat nasional berlangsung ketat. Setiap peserta diberikan 8 soal yang harus dapat diselesaikan dalam waktu 5 jam. 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, 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 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 jam1 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) (int, int) {
    totalSoal := 0
    totalSkor := 0

    for _, w := range waktu {
        if w < 301 {
            totalSoal++
            totalSkor += w
        }
    }
    return totalSoal, totalSkor
}

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

    pemenang := ""
    maxSoal := -1
    minSkor := 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])

        totalSoal, totalSkor := hitungSkor(waktu[:])

        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 C:\Users\HP> go run "d:\Modul 4\Unguided\unguided2.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 C:\Users\HP> |
```

Deskripsi Program

Program Go di atas dirancang untuk menghitung pemenang dalam suatu kompetisi berdasarkan waktu yang dihabiskan oleh peserta untuk menjawab soal. Pertama, program meminta input jumlah peserta dan kemudian mengumpulkan nama serta waktu untuk setiap peserta yang terdiri dari delapan soal. Fungsi `hitungSkor` digunakan untuk menghitung total soal yang dijawab dan total waktu yang dihabiskan oleh masing-masing peserta. Peserta yang menjawab lebih banyak soal dalam waktu kurang dari atau sama dengan 300 detik akan mendapatkan poin, dan di antara peserta yang menjawab jumlah soal sama, pemenang ditentukan berdasarkan skor terendah (total waktu). Setelah semua peserta dimasukkan, program mencetak nama pemenang beserta jumlah soal yang dijawab dan total waktu yang dihabiskan. Dengan menggunakan prosedur, program ini terstruktur dengan baik, memisahkan logika perhitungan skor dari proses input dan output, sehingga meningkatkan keterbacaan dan pemeliharaan kode.

3. Soal Studi Case

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.

prosedure cetakDeret(in n : integer)

Masukan berupa satu bilangan integer positif yang lebih kecil 1000000.

Keluaran terdiri dari satu baris saja. Setiap suku dari deret tersebut dicetak dalam baris yang 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 {
        fmt.Print(n, " ")
        if n == 1 {
            break
        }
        if n%2 == 0 {
            n = n / 2
        } else {
            n = 3*n + 1
        }
    }
}

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 C:\Users\HP> go run "d:\Modul 4\Unguided\unguided3.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 C:\Users\HP> █
```

Deskripsi Program

Program Go di atas bertujuan untuk mencetak deret bilangan berdasarkan aturan tertentu hingga mencapai angka satu. Dalam fungsi utama (main), pengguna diminta untuk memasukkan bilangan bulat positif yang lebih kecil dari 1.000.000. Jika input valid, program memanggil prosedur `cetakDeret`, yang menerima input tersebut dan menjalankan loop yang mencetak nilai n diikuti dengan spasi. Dalam setiap iterasi, program memeriksa apakah n adalah genap atau ganjil; jika genap, n dibagi dua, sedangkan jika ganjil, n diubah menjadi $3*n + 1$. Proses ini berulang hingga n mencapai satu, dimana loop akan berhenti. Jika input tidak memenuhi kriteria yang ditetapkan, program menampilkan pesan kesalahan yang menjelaskan bahwa input harus positif dan kurang dari 1.000.000.

DAFTAR PUSTAKA

- Dinas Kesehatan Provinsi Jawa Timur. (n.d.). Dasar pemrograman Golang. Diakses pada 20 Oktober 2024, dari <https://dinkes.jatimprov.go.id/userfile/dokumen/Dasar%20Pemrograman%20Golang.pdf>
- Go Dev. (n.d.). *Getting started*. Retrieved October 20, 2024, from <https://go.dev/doc/tutorial/getting-started>
- TutorialsPoint. (n.d.). Fungsi dalam Go. Diakses 20 Oktober 2024, dari https://www.tutorialspoint.com/go/go_functions.htm