

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL IV
PROSEDUR**



Disusun Oleh :

Afra Lintang Maharani / 2311102258

S1 – 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, juga dikenal sebagai fungsi atau subrutin, adalah blok kode yang dirancang untuk melakukan tugas tertentu. Dalam bahasa pemrograman Go, prosedur sangat penting untuk strukturisasi kode, mempromosikan penggunaan kembali, dan meningkatkan keterbacaan serta pemeliharaan kode. Prosedur memungkinkan pengembang untuk mengemas logika yang kompleks ke dalam unit yang lebih kecil dan lebih mudah dikelola. Percabangan (Conditional Statements). Prosedur didefinisikan menggunakan kata kunci `func`, diikuti dengan nama fungsi, parameter yang diperlukan, tipe kembalian, dan badan fungsi yang berisi pernyataan yang akan dieksekusi.

- **Parameter dan Argumen,**
Prosedur dapat menerima satu atau lebih parameter yang memungkinkan pengiriman data dari pemanggil ke fungsi. Parameter ini dideklarasikan di dalam tanda kurung setelah nama fungsi. Saat memanggil prosedur, nilai yang dikirim ke parameter ini disebut sebagai argumen.
 - Parameter Biasa: Digunakan untuk mengirim nilai secara langsung.
 - Parameter Pointer: Digunakan untuk mengirim alamat memori dari variabel sehingga fungsi dapat memodifikasi nilai asli.
- **Keuntungan Menggunakan Prosedur**
 - Modularitas, Membagi program menjadi bagian-bagian yang lebih kecil dan modular.
 - Reusabilitas, Menggunakan kembali kode yang sama tanpa harus menulis ulang.
 - Keterbacaan, Meningkatkan keterbacaan dan pemeliharaan kode.
 - Abstraksi, Menyembunyikan detail implementasi dan hanya menampilkan antarmuka yang diperlukan.

II. GUIDED

1. Soal Studi Case

Menghitung Faktorial dan Permutasi menggunakan prosedur.

Sourcecode

```
package main

import "fmt"

func main() {
    var x, y int
    fmt.Print("Masukkan dua angka: ")
    fmt.Scanln(&x, &y)

    if x < y {
        x, y = y, x
    }

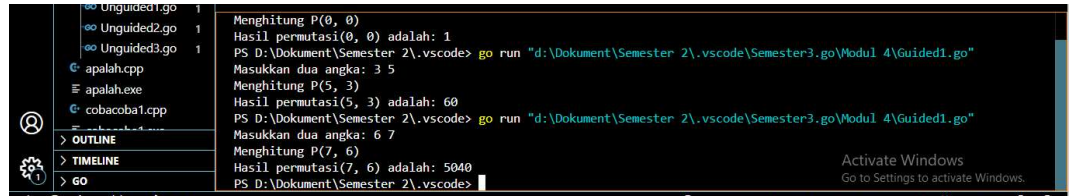
    hitungPermutasi(x, y)
}

func hitungPermutasi(n, r int) {
    fmt.Printf("Menghitung P(%d, %d)\n", n, r)
    permutasi(n, r)
}

func faktorial(n int, hasil *int) {
    *hasil = 1
    for j := 2; j <= n; j++ {
        *hasil *= j
    }
}

func permutasi(n, r int) {
    var factN, factNMinR int
    faktorial(n, &factN)
    faktorial(n-r, &factNMinR)
    fmt.Printf("Hasil permutasi(%d, %d) adalah: %d\n", n, r,
factN/factNMinR)
}
```

Screenshoot Output



Deskripsi Program

Program diatas adalah contoh program yang berguna untuk menghitung permutasi dari dua angka yang di inputkan oleh pengguna. Program ini akan dimulai dengan diminta nya pengguna untuk memasukkan dua inputan angka, jika angka pertama yang di inputkan oleh pengguna lebih kecil dari angka kedua yang di inputkan maka, nilai yang lebih kecil tersebut akan ditukar dengan nilai yang lebih besar. Sehingga, angka pertama akan selalu lebih besar daripada angka kedua. Setelah itu, fungsi hitungPermutasi akan dipanggil untuk mencetak informasi dan memanggil fungsi permutasi, kemudian fungsi permutasi akan menghitung kedua angka yang telah di inputkan oleh pengguna, setelahnya kedua angka tersebut akan di hitung sesuai rumus yang ada, terakhir program akan menampilkan output hasil hitungannya.

2. Soal Studi Case

Menghitung luas dan keliling persegi dengan prosedur.

Sourcecode

```
package main

import "fmt"

func main() {
    var sisi int
    fmt.Print("Masukkan panjang sisi persegi: ")
    fmt.Scanln(&sisi)

    tampilkanHasil(sisi)
}

func tampilkanHasil(s int) {
    fmt.Println("Menghitung luas dan keliling persegi dengan sisi:", s)
    fmt.Println("Luas Persegi:", hitungLuas(s))
    fmt.Println("Keliling Persegi:", hitungKeliling(s))
}
```

```

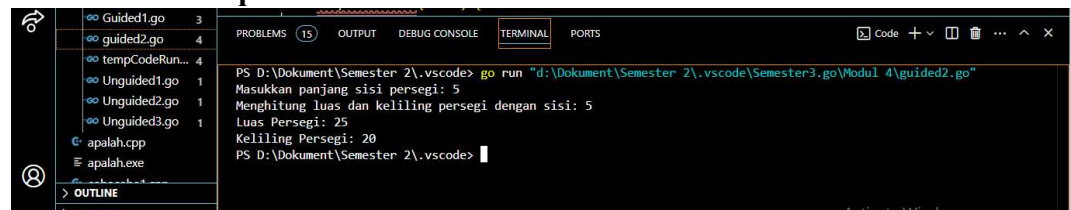
}

func hitungLuas(s int) int {
    return s * s
}

func hitungKeliling(s int) int {
    return 4*s
}

```

Screenshoot Output



Deskripsi Program

Program diatas adalah contoh program yang digunakan untuk menghitung luas dan keliling persegi berdasarkan angka yang telah di inputkan oleh pengguna. Pertama-tama, pengguna akan di minta untuk menginputkan sisi persegi menggunakan fmt.Scanln setelah itu, program akan menghitung luas dan keliling persegi berdasarkan dengan inputan dari pengguna. Terakhir, program akan menampilkan output berupa luas dan keliling pengguna.

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 membantu Jonas? (tidak tentunya ya :p) Masukan terdiri dari empat buah bilangan asli a, b, c, dan d yang dipisahkan oleh spasi, dengan syarat $a < b$ 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! $P(n,r) = n! / (n-r)!$, sedangkan $C(n,r) = n! / r!(n-r)!$

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

func Permutasi(n, r int, hasil *int) {
    var fn, fnr int
    Faktorial(n, &fn)
    Faktorial(n-r, &fnr)
    *hasil = fn / fnr
}
```

```

func Kombinasi(n, r int, hasil *int) {
    var fn, fr, fnr int
    Faktorial(n, &fn)
    Faktorial(r, &fr)
    Faktorial(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
        Permutasi(a, c, &permAC)
        Kombinasi(a, c, &combAC)
        Permutasi(b, d, &permBD)
        Kombinasi(b, d, &combBD)

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

```

Screenshoot Output



```

PS D:\Ghilbran Anjayy\Modul 4\Unguided 1>
Masukkan nilai a, b, c, d: 9 7 5 4
Hasil Permutasi dan Kombinasi untuk (a, c): 15120 126
Hasil Permutasi dan Kombinasi untuk (b, d): 840 35

```

Deskripsi Program

Program diatas adalah program yang untuk menghitung permutasi dan kombinasi berdasarkan empat angka yang dimasukkan oleh pengguna. Program ini terdiri dari tiga fungsi utama: Faktorial, Permutasi, dan Kombinasi. Fungsi Faktorial menghitung faktorial dari sebuah angka. Fungsi Permutasi menggunakan faktorial untuk menghitung permutasi dari dua angka, sementara fungsi Kombinasi menghitung kombinasi dari dua angka menggunakan faktorial dari kedua angka tersebut dan selisihnya. Di dalam fungsi main, program meminta pengguna untuk memasukkan empat angka. Setelah itu, program memeriksa apakah dua kondisi terpenuhi: $a*c \geq d$ dan $b \geq d$. Jika kedua kondisi ini terpenuhi, program menghitung dan mencetak permutasi serta kombinasi dari pasangan angka a dengan c, dan b dengan d. Jika kondisi tidak terpenuhi, program akan

mencetak pesan "Syarat tidak terpenuhi." untuk memberi tahu pengguna bahwa perhitungan tidak dapat dilakukan. Program ini menunjukkan cara dasar menggunakan pointer untuk menyimpan hasil perhitungan dan penggunaan pernyataan kondisional untuk memeriksa syarat sebelum melakukan perhitungan matematis.

2. Soal Studi Case

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 tau 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.

Activate Windows
Go to Settings to activate Windows

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:\Ghilbran Anjayy\Modul 4\Unguided 1> go run main.go
Input jumlah peserta: 2
Input nama dan waktu: Afra 20 50 301 301 61 71 75 10
Input nama dan waktu: Wafiq 25 47 301 26 50 60 65 21
Pemenang: Wafiq
Jumlah soal: 7
Total waktu: 294
PS D:\Ghilbran Anjayy\Modul 4\Unguided 1>

```

Deskripsi Program

Program diatas adalah contoh program untuk menentukan pemenang dari sebuah kompetisi berdasarkan waktu penyelesaian soal oleh beberapa peserta. Pertama, program meminta pengguna untuk memasukkan jumlah peserta. Kemudian, dalam loop untuk setiap peserta, program meminta pengguna untuk memasukkan nama dan waktu penyelesaian untuk delapan soal.

Fungsi HitungSKOR kemudian digunakan untuk menghitung jumlah soal yang diselesaikan dalam waktu kurang dari 301 detik dan total waktu yang digunakan untuk menyelesaikan soal-soal tersebut. Fungsi ini menerima slice waktu, dan dua pointer, soal dan skor, untuk menyimpan jumlah soal yang diselesaikan dan total waktu.

Setelah perhitungan, program membandingkan hasil peserta saat ini dengan hasil peserta sebelumnya. Jika peserta saat ini menyelesaikan lebih banyak soal, atau jika jumlah soal sama tetapi total waktu lebih sedikit, informasi peserta saat ini disimpan sebagai pemenang. Setelah loop selesai, program mencetak nama pemenang, jumlah soal yang diselesaikan, dan total waktu yang digunakan.

Secara keseluruhan, program ini menunjukkan penggunaan dasar dari loop, slice, dan pointer dalam Go untuk menyelesaikan permasalahan praktis dalam menentukan pemenang kompetisi berdasarkan kriteria yang ditentukan.

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 $2n$, 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 cetak Deret yang mempunyai 1 parameter formal, yaitu nilai dari suku awal. prosedur 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"

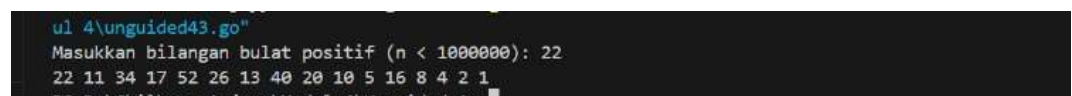
func MencetakDeret(n int) {
    for n != 1 {
        fmt.Print(n, " ")
        if n%2 == 0 {
            n = n / 2
        } else {
            n = 3*n + 1
        }
    }
    fmt.Print(n)
}

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

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

```

Screenshoot Output



```

ul 4\unguided43.go"
Masukkan bilangan bulat positif (n < 1000000): 22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

```

Deskripsi Program

Program diatas adalah contoh program yang digunakan untuk mencetak deret angka berdasarkan aturan tertentu, dimulai dari bilangan bulat positif yang dimasukkan oleh pengguna. Pengguna diminta untuk memasukkan bilangan bulat positif yang kurang dari 1.000.000. Jika input valid, fungsi MencetakDeret dipanggil.

Fungsi MencetakDeret mencetak angka dan memodifikasinya: jika angka genap, dibagi dua; jika ganjil, dihitung dengan $3*n + 1$. Proses ini berlanjut hingga angka menjadi 1, dan setiap langkah dicetak ke layar.

Jika input tidak valid, program menampilkan pesan kesalahan. Program ini menggunakan loop dan pernyataan kondisional untuk mengolah input pengguna secara iteratif.

III. DAFTAR PUSTAKA

* *TutorialsPoint*: <https://www.tutorialspoint.com/go/index.htm>