LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

MODUL IV

PROSEDUR

Oleh:

Alden Audy Akbar

2311102309

IF-11-07

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

4.1 Definisi Procedure

Prosedur dapat dianggap sebagai potongan beberapa instruksi program menjadi suatu instruksi baru yang dibuat untuk mengurangi kerumitan dari kode program yang kompleks pada suatu program yang besar. Prosedur akan menghasilkan suatu akibat atau efek langsung pada program ketika dipanggil pada program utama. Suatu subprogram dikatakan prosedur apabila:

1. Tidak ada deklarasi tipe nilai yang dikembalikan, dan
2. Tidak terdapat kata kunci return dalam badan subprogram.

Kedudukannya prosedur sama seperti instruksi dasar yang sudah ada sebelumnya (assignment) danlatau instruksi yang berasal dari paket (fmt), seperti fmt.Scan dan fmt.Print. Karena itu selalu pilih nama prosedur yang berbentuk kata kerJa atau sesuatu yang merepresentasikan proses sebagai nama dari prosedur. Contoh: cetak, hitungRerata, cariNilai, belok, mulai, ...

4.2 Deklarast Procedure

Berikut ini adalah cara penulisan deklarasi prosedur pada notasi Pseudocode dan GoLang.

	Notasi Algoritma
2 3 4 5 6 7 8	procedure <nama procedure> («params>) kamus {deklarasi variabel lokal dari procedure} algoritma {badan algoritma procedure} endprocedure
	Notasi dalam bahasa Go
9 10 11 12 13 14	func <nama procedure> < (params)> / * deklarasi variabel lokal dari procedure */ / * badan algoritma procedure */

Penulisan deklarasi ini berada di luar blok yang dari program utama atau func main() pada suatu program Go, dan bisa ditulis sebelum atau setelah dari blok program utama tersebut.

Contoh deklarasi prosedur mencetak n nilai pertama dari deret Fibonacci.

	Notasi Algoritma
1 2 3 4 5 6 7 8 9 10 11 12 13	procedure cetakNFibo(in n integer) kamus f1 , f2, f3, i intege algoritma f2 ê f3 I for i ← 1 to n do output (f3) f2 f3 f3 f1 + f2 endfor endprocedure
	Notasi dalam bahasa Go
14 15 16 17 18 19 21 22 23 24	func cetakNFibo(n int) { var f1 , f2, f3 int f2 f3 for i fmt . print1n(f3) f1 = f2 f2f3 f3 - f1 + f2

4.3 Cara Pemanggilan Procedure

Seperti yang sudah dijelaskan sebelumnya, suatu prosedur hanya akan dieksekusi apabila dipanggil baik secara langsung atau tidak langsung oleh program utama. Tidak langsung di sini maksudnya adalah prosedur dipanggil oleh program utama melalu perantara subprogram yang Iain.

Pemanggilan suatu prosedure cukup mudah, yaitu dengan hanya menuliskan nama beserta parameter atau argumen yang dimlnta dart suatu prosedur. Sebagai contoh prosedur cetakNFibo di atas dipanggil dengan menuliskan namanya, kemudian sebuah variabel atau nilai integer tertentu sebagai argumen untuk paramter n. Contoh:

	Notasi Algoritma
2 3 4 5 6 7	program contohprosedur kamus x integer algoritma cetakNFibo(x) {cara pemanggilan } cetakNFibo(1êê) {cara pemanggilan #2}
8	endprogram
	Notasi dalam bahasa Go
9 10 11 12 13 14	func main() { var x int 5 cetakNFibo(x) cetakNFibo(100) {cara pemanggilan #1 } {cara pemanggilan #2}

Dari contoh di atas terlihat bahwa cara pemanggilan dengan notasi pseudocode dan GoLang adalah sama. Argumen yang digunakan untuk parameter n berupa integer (sesuai deklarasi) yang terdapat pada suatu variabel (cara pemanggilan #1) atau nilainya secara langsung (cara pemanggilan #2).

II. GUIDED

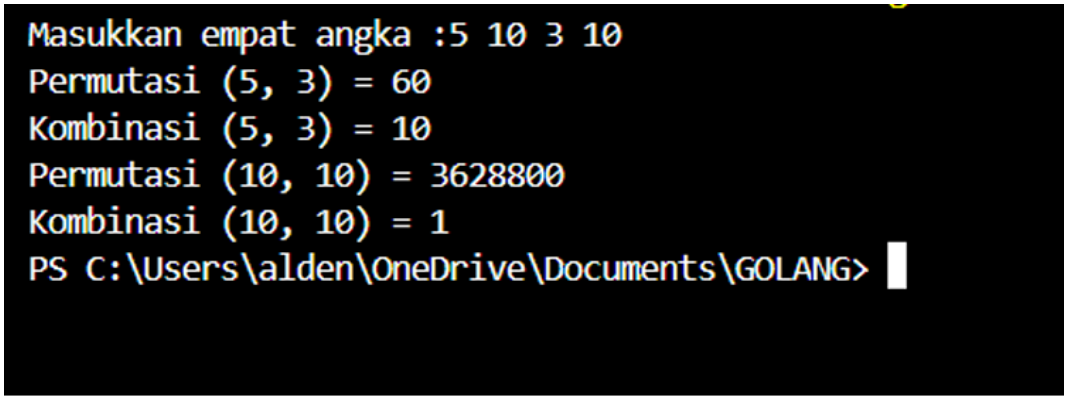
Source Code + Screenshot hasil program beserta penjelasan

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)

```
package main


//2311102309
import (
    "fmt"
    "math/big"
)


func factorial(n int64) *big.Int {
    result := big.NewInt(1)
    for i := int64(2); i <= n; i++ {
        result.Mul(result, big.NewInt(i))
    }
    return result
}
func perm(n, r int64) *big.Int {
    if r > n {
        return big.NewInt(0)
    }
    return new(big.Int).Div(factorial(n), factorial(n-r))
}
```

```
func comb(n, r int64) *big.Int {
    if r > n {
        return big.NewInt(0)
    }

    return new(big.Int).Div(factorial(n), new(big.Int).Mul(factorial(r),
        factorial(n-r)))
}

func main() {
    var a, b, c, d int64
    fmt.Print("Masukkan empat angka :")
    fmt.Scan(&a, &b, &c, &d)
    P_ac := perm(a, c)
    C_ac := comb(a, c)
    P_bd := perm(b, d)
    C_bd := comb(b, d)
    fmt.Printf("Permutasi (%d, %d) = %d\n", a, c, P_ac)
    fmt.Printf("Kombinasi (%d, %d) = %d\n", a, c, C_ac)
    fmt.Printf("Permutasi (%d, %d) = %d\n", b, d, P_bd)

    fmt.Printf("Kombinasi (%d, %d) = %d\n", b, d, C_bd)
}
```



Penjelasan: Program meminta pengguna untuk memasukkan empat bilangan bulat.Empat bilangan ini akan digunakan sebagai input untuk perhitungan permutasi dan kombinasi.Program memiliki fungsi khusus untuk menghitung faktorial dari suatu bilangan. Faktorial dari sebuah bilangan adalah hasil perkalian bilangan itu dengan semua bilangan bulat positif yang lebih kecil darinya.Fungsi ini sangat penting karena rumus permutasi dan kombinasi melibatkan faktorial.Menggunakan rumus permutasi, program menghitung banyaknya cara menyusun objek-objek dalam suatu urutan tertentu. Rumus permutasi: P(n, r) = n! / (n-r)! P(n, r) adalah jumlah permutasi adalah jumlah total objek r adalah jumlah objek yang dipilih Hasil perhitungan permutasi kemudian disimpan.Menggunakan rumus kombinasi, program menghitung banyaknya cara memilih objek-objek tanpa memperhatikan urutannya. Rumus kombinasi: C(n, r) = n! / (r! * (n-r)!) C(n, r) adalah jumlah kombinasi n dan r memiliki arti yang sama seperti pada permutasi Hasil perhitungan kombinasi kemudian disimpan. Setelah semua perhitungan selesai, program akan menampilkan hasil perhitungan permutasi dan kombinasi untuk setiap pasangan bilangan yang telah dimasukkan oleh pengguna.

```
Program Permutasi_Kombinasi_Dengan_Validasi

kamus
  a, b, c, d: integer

algoritma

  Tulis("Masukkan empat angka (dipisahkan spasi): ")
  Baca(a, b, c, d)


  jika r > n untuk semua (r, n) dari pasangan [(a, c), (b, d)] lakukan
    Tulis("Input tidak valid. r tidak boleh lebih besar dari n.")
    hentikan program
```

```
  akhir jika


  permutasiAC := Permutasi(a, c)
  kombinasiAC := Kombinasi(a, c)



  permutasiBD := Permutasi(b, d)
  kombinasiBD := Kombinasi(b, d)


  Tulis("Permutasi dan Kombinasi:")
  Tulis("  Permutasi (", a, ",", c, ") = ", permutasiAC)
  Tulis("  Kombinasi (", a, ",", c, ") = ", kombinasiAC)
  Tulis("") // Baris kosong untuk pemisah
  Tulis("  Permutasi (", b, ",", d, ") = ", permutasiBD)
  Tulis("  Kombinasi (", b, ",", d, ") = ", kombinasiBD)

akhirprogram

fungsi Permutasi(n, r: integer): big.Int

  jika r > n maka
    kembalikan big.NewInt(0)
  akhir jika
  kembalikan Faktorial(n) / Faktorial(n - r)

akhirfungsi

fungsi Kombinasi(n, r: integer): big.Int

  jika r > n maka
    kembalikan big.NewInt(0)
  akhir jika
  return Faktorial(n) / (Faktorial(r) * Faktorial(n - r))

akhirfungsi

fungsi Faktorial(n: integer): big.Int

  hasil := big.NewInt(1)
  untuk i := 2 hingga n lakukan
    hasil := hasil * big.NewInt(i)
  akhir untuk
  kembalikan hasil
```

```
akhirfungsi
```

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.

```
package main

//2311102309
import "fmt"

func hitungSkor(soal []int, skor *map[string]int) {
    soalDiselesaikan := 0
    totalWaktu := 0
    for _, waktu := range soal {
        if waktu <= 300 {
            soalDiselesaikan++
            totalWaktu += waktu
        }
    }
    (*skor)["soalDiselesaikan"] = soalDiselesaikan
    (*skor)["totalWaktu"] = totalWaktu
}
func tentukanPemenang(skorAstuti, skorBertha map[string]int) {
    var pemenang string
    var soalDiselesaikan, totalWaktu int
    if skorBertha["soalDiselesaikan"] > skorAstuti["soalDiselesaikan"] {
        pemenang = "Bertha"
        soalDiselesaikan = skorBertha["soalDiselesaikan"]
        totalWaktu = skorBertha["totalWaktu"]
    } else if skorAstuti["soalDiselesaikan"] > skorBertha["soalDiselesaikan"] {
```

```
        pemenang = "Astuti"
        soalDiselesaikan = skorAstuti["soalDiselesaikan"]
        totalWaktu = skorAstuti["totalWaktu"]
    } else {

        if skorBertha["totalWaktu"] < skorAstuti["totalWaktu"] {
            pemenang = "Bertha"
            soalDiselesaikan = skorBertha["soalDiselesaikan"]
            totalWaktu = skorBertha["totalWaktu"]
        } else {
            pemenang = "Astuti"
            soalDiselesaikan = skorAstuti["soalDiselesaikan"]
            totalWaktu = skorAstuti["totalWaktu"]
        }
    }
    fmt.Printf("Pemenang: %-10sSoal: %-8dWaktu: %-6d\n", pemenang,
soalDiselesaikan, totalWaktu)
}

func main() {
    soalAstuti := []int{20, 50, 301, 301, 61, 71, 75, 10}
    soalBertha := []int{25, 47, 301, 26, 50, 60, 65, 21}
    skorAstuti := make(map[string]int)
    skorBertha := make(map[string]int)

    hitungSkor(soalAstuti, &skorAstuti)
    hitungSkor(soalBertha, &skorBertha)
    tentukanPemenang(skorAstuti, skorBertha)
}
```

```
Pemenang: Bertha    Soal: 7       Waktu: 294
PS C:\Users\alden\OneDrive\Documents\GOLANG>
```

Penjelasan: hitungSkor:Menghitung jumlah soal yang diselesaikan dan total waktu yang digunakan oleh seorang peserta. Menerima slice soal yang berisi waktu penyelesaian setiap soal. Mengiterasi setiap elemen dalam slice soal.

Jika waktu penyelesaian kurang dari atau sama dengan 300 detik, soal dianggap selesai dan dihitung ke dalam total soal yang diselesaikan dan total waktu. Hasil perhitungan disimpan dalam map skor.

tentukanPemenang: Membandingkan skor kedua peserta dan menentukan pemenang. Membandingkan jumlah soal yang diselesaikan oleh kedua peserta. Jika jumlah soal berbeda, peserta dengan jumlah soal lebih banyak menang. Jika jumlah soal sama, peserta dengan total waktu lebih sedikit menang. Hasil akhir dicetak ke layar.

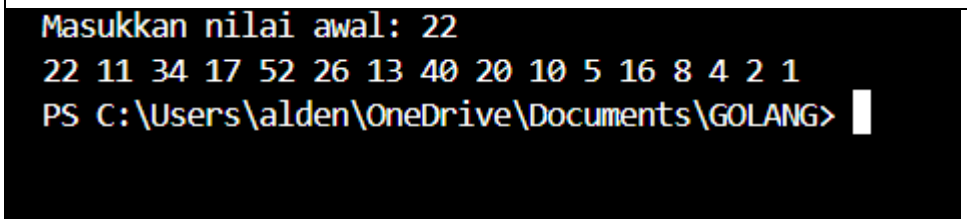
Struktur Data: Slice Digunakan untuk menyimpan waktu penyelesaian soal setiap peserta. Map Digunakan untuk menyimpan hasil perhitungan skor (jumlah soal diselesaikan dan total waktu).

Inisialisasi data waktu penyelesaian soal untuk kedua peserta. Hitung skor masing-masing peserta menggunakan fungsi hitungSkor. Tentukan pemenang menggunakan fungsi tentukanPemenang. Cetak hasil pemenang ke layar.

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 $^{1}/4$n, tetapijika ganjil maka suku berikutnya bernilai 3n+1. Rumus yang sama digunakan terus menerus untuk mencari suku berikutnya. Deret berakhir ketika suku terakhir bernilai 1.

```
package main

//2311102309
import "fmt"

func collatzSequence(n int) []int {
    var sequence []int
    for n != 1 {
        sequence = append(sequence, n)
        if n%2 == 0 {
            n /= 2
        } else {
            n = 3*n + 1
        }
    }
    sequence = append(sequence, 1)
    return sequence
}
func cetakDeret(deret []int) {
    for _, nilai := range deret {
        fmt.Print(nilai, " ")
```

```
    }
    fmt.Println()
}
func main() {
    var n int
    fmt.Print("Masukkan nilai awal: ")
    fmt.Scan(&n)
    result := collatzSequence(n)
    cetakDeret(result)
}
```



Penjelasan: Program ini menghitung dan mencetak deret bilangan yang mengikuti aturan Collatz. Aturan Collatz menyatakan bahwa jika suatu bilangan n adalah:

Genap: Bilangan berikutnya adalah n/2.

Ganjil: Bilangan berikutnya adalah 3n+1.

Deret ini dihentikan ketika mencapai nilai 1.

Fungsi collatzSequence: n adalah bilangan awal deret. Slice of int yang berisi deret bilangan yang dihasilkan. Inisialisasi slice kosong sequence untuk menyimpan deret.Selama n tidak sama dengan 1, terus menambahkan n ke slice dan menghitung suku berikutnya berdasarkan aturan Collatz. Tambahkan 1 ke akhir slice.

Kembalikan slice yang berisi deret bilangan. Fungsi cetakDeret:deret adalah slice yang berisi deret bilangan.Mencetak setiap elemen dalam slice deret dengan spasi pemisah. Fungsi main Meminta pengguna untuk memasukkan nilai awal n. Memanggil fungsi collatzSequence untuk menghitung deret Memanggil fungsi cetakDeret untuk mencetak deret yang dihasilkan.