

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 4
PROSEDUR**



Disusun Oleh :

PRIESTY AMEILIANA MAULIDAH / 2311102175

IF-11-05

Dosen Pengampu :

ARIF AMRULLOH, S.KOM.,M.KOM

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

1.) DASAR TEORI

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. Suatu subprogram dikatakan prosedur apabila:

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

2. Deklarasi procedure

3. cara pemanggilan procedure

Suatu prosedur hanya akan dieksekusi apabila dipanggil secara langsung atau tidak langsung oleh program utama. Menuliskan nama beserta parameter atau argumen yang diminta dari suatu prosedur

4. contoh program dengan procedure

5. parameter

Suatu subprogram yang dipanggil dapat berkomunikasi dengan pemanggilnya melalui argumen yang diberikan melalui parameter yang dideklarasikan pada subprogram. Maka parameter dikelompokkan menjadi dua:

1. Parameter formal

Parameter formal adalah parameter yang ditulis saat deklarasi subprogram, berfungsi sebagai petunjuk argumen yang diperlukan saat pemanggilan subprogram. Parameter jari-jari dan tinggi pada fungsi volumeTabung adalah parameter formal. Saat memanggil volumeTabung, kita perlu menyiapkan dua integer untuk jari-jari dan tinggi.

2. Parameter aktual

Parameter aktual adalah argumen yang digunakan saat memanggil subprogram. Argumen dan tipe data pada parameter aktual harus sesuai dengan parameter formal. Contohnya, argumen r, t, 15, 14, dan 100 dalam kode di atas adalah parameter aktual yang menunjukkan nilai jari-jari dan tinggi.

Selain parameter juga dikelompokkan berdasarkan alokasi memorinya, yaitu pass by value dan pass by reference:

1. pass by value

Pass by Value menyalin nilai parameter aktual ke variabel lokal di subprogram. Parameter aktual dan formal dialokasikan di memori dengan alamat berbeda. Subprogram dapat menggunakan nilai parameter formal, tetapi tidak bisa mengembalikannya pada pemanggil karena pemanggil tidak mengakses memori subprogram. Pass by value dapat digunakan oleh fungsi dan prosedur. Dalam pseudocode, semua parameter formal pada fungsi adalah pass by value, sedangkan pada prosedur ditandai dengan kata kunci "in". Pada bahasa pemrograman Go, tidak ada kata kunci khusus untuk parameter formal fungsi dan prosedur, mirip dengan pseudocode.

2. pass by reference

Saat parameter didefinisikan sebagai pass by reference, parameter formal berfungsi sebagai pointer yang menyimpan alamat memori dari parameter aktual saat pemanggilan. Perubahan nilai pada parameter formal akan mempengaruhi parameter aktual, sehingga nilai akhirnya dapat diketahui setelah subprogram selesai dieksekusi. Pass by reference sebaiknya hanya digunakan untuk prosedur, dengan penulisan parameter menggunakan kata kunci atau identifier khusus dalam pseudocode dan Go. Pada pseudocode, digunakan kata kunci in/out, sedangkan bahasa Go menggunakan asterik (*) sebelum tipe data di parameter formal untuk pass by reference.

2.) GUIDED

Soal Studi Case

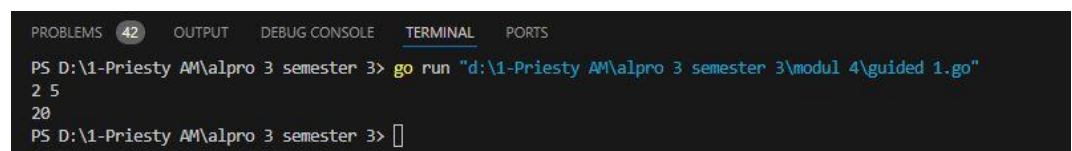
1

Sourcecode



```
modul 4 > go guided 1.go > ...
1  package main
2
3  import "fmt"
4
5  func main() {
6      var a, b int
7      fmt.Scan(&a, &b)
8      if a >= b {
9          permutasi(a, b)
10     } else {
11         permutasi(b, a)
12     }
13 }
14
15 func faktorial(n int) int {
16     var hasil int = 1
17     for i := 1; i <= n; i++ {
18         hasil = hasil * i
19     }
20     return hasil
21 }
22
23 func permutasi(n, r int) {
24     result := faktorial(n) / faktorial(n-r)
25     fmt.Println(result)
26 }
27
```

Screenshoot Output



```
PROBLEMS 42 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\1-Priesty AM\alpro 3 semester 3> go run "d:\1-Priesty AM\alpro 3 semester 3\modul 4\guided 1.go"
2 5
20
PS D:\1-Priesty AM\alpro 3 semester 3>
```

Deskripsi Program

Program meminta dua bilangan bulat a dan b dari pengguna. Jika $a \geq b$, program menghitung permutasi dengan a sebagai total objek dan b sebagai objek yang dipilih. Jika tidak, program menukar nilainya. Fungsi Faktorial menghitung faktorial dari n ($n! = n * (n-1) * \dots$). Silakan berikan teks yang

ingin Anda persingkat. Fungsi Permutasi: Fungsi permutasi(n, r) menghitung $P(n, r) = n! / (n-r)!$. Hasilnya dicetak sebagai output program yang menampilkan permutasi dari dua angka.

Input: $a = 2$ dan $b = 5$

Output: permutasi 2 objek yang dipilih 5 adalah 20 ($p(2,5) = 2! / 5! = 20$).

GUIDED

Soal Studi Case

2.

Sourcecode

```
modul 4 > guided 2.go modul 4 1 X  guided 2.go \ 6  guided 1.go modul 3 5

modul 4 > guided 2.go > main
1  package main
2
3  import (
4      "fmt"
5  )
6
7  func hitungPersegi(sisi float64) (float64, float64) {
8      luas := sisi * sisi
9      keliling := 4 * sisi
10     return luas, keliling
11 }
12
13 func main() {
14     var sisi float64
15     fmt.Print("masukkan panjang sisi persegi: ")
16     fmt.Scan(&sisi)
17
18     luas, keliling := hitungPersegi(sisi)
19
20     fmt.Printf("luas persegi: %.2f\n", luas)
21     fmt.Printf("keliling persegi: %.2f\n", keliling)
22 }
```

Screenshoot Output

```
PROBLEMS 42 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\1-Priesty AM\alpro 3 semester 3> go run "d:\1-Priesty AM\alpro 3 semester 3\modul 4\guided 2.go"
masukkan panjang sisi persegi: 5
luas persegi: 25.00
keliling persegi: 20.00
PS D:\1-Priesty AM\alpro 3 semester 3> go run "d:\1-Priesty AM\alpro 3 semester 3\modul 4\guided 2.go"
masukkan panjang sisi persegi: 4
luas persegi: 16.00
keliling persegi: 16.00
PS D:\1-Priesty AM\alpro 3 semester 3> 
```

Deskripsi Program

Program meminta pengguna memasukkan panjang sisi persegi. Fungsi `hitungPersegi` menerima parameter sisi (`float64`) dan mengembalikan dua nilai: Luas ($\text{sisi} \times \text{sisi}$) dan Keliling ($4 \times \text{sisi}$). Program kemudian mengambil input panjang sisi persegi. Fungsi `hitungPersegi` menghitung luas dan keliling dengan hasil dalam format desimal dua angka.

- jika pengguna memasukkan panjang sisi 5:

Input

Luas = $5 \times 5 = 25$

Keliling = $4 \times 5 = 20$

Output

Luas persegi: 25.00

Keliling persegi: 20.00

- Jika pengguna memasukkan panjang sisi 4:

Input

Luas = $4 \times 4 = 16$

Keliling = $4 \times 4 = 16$

Output

Luas persegi: 16.00

Keliling persegi: 16.00

3.) **UNGUIDED**

Modul 4

Soal Studi Case

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 bersedia kah 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!

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

Sourcecode

```
modul4 > go run "d:\1-Priesty AM\alpro 3 semester 3\modul 4\unguided 1.go"
1 package main
2
3 import (
4     "fmt"
5     "math/big"
6 )
7
8 func factorial(n int) *big.Int {
9     if n == 0 {
10         return big.NewInt(1)
11     }
12     result := big.NewInt(1)
13     for i := 1; i <= n; i++ {
14         result.Mul(result, big.NewInt(int64(i)))
15     }
16     return result
17 }
18
19 func permutation(n, r int) *big.Int {
20     return factorial(n).Div(factorial(n), factorial(n-r))
21 }
22
23 func combination(n, r int) *big.Int {
24     return factorial(n).Div(factorial(n), factorial(r).Mul(factorial(r), factorial(n-r)))
25 }
26
27 func main() {
28     var a, b, c, d int
29     fmt.Scan(&a, &b, &c, &d)
30
31     if a >= c && b >= d {
32         p1 := permutation(a, c)
33         c1 := combination(a, c)
34         p2 := permutation(b, d)
35         c2 := combination(b, d)
36
37         fmt.Printf("P(%d,%d) %s\n", a, c, p1.String())
38         fmt.Printf("C(%d,%d) %s\n", a, c, c1.String())
39         fmt.Printf("P(%d,%d) %s\n", b, d, p2.String())
40         fmt.Printf("C(%d,%d) %s\n", b, d, c2.String())
41     }
42 }
```

Screenshot Output

```
PROBLEMS 34 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\1-Priesty AM\alpro 3 semester 3> go run "d:\1-Priesty AM\alpro 3 semester 3\modul 4\unguided 1.go"
5 10 3 10
P(5,3) 60
C(5,3) 10
P(10,10) 3628800
C(10,10) 1
PS D:\1-Priesty AM\alpro 3 semester 3> go run "d:\1-Priesty AM\alpro 3 semester 3\modul 4\unguided 1.go"
8 0 2 0
P(8,2) 56
C(8,2) 28
P(0,0) 1
C(0,0) 1
PS D:\1-Priesty AM\alpro 3 semester 3>
```

Deskripsi Program

- Faktorial: Menghitung faktorial $n!$ menggunakan loop, disimpan dalam tipe `big.Int`.
- Permutasi: Menghitung permutasi

$P(n,r) = n! / (n-r)!$ menggunakan faktorial.

- Kombinasi: Menghitung kombinasi $C(n,r) = n! / r! \times (n-r)!$
- Logika Utama: Program menerima input, memeriksa apakah input valid ($a \geq c$ dan $b \geq d$), lalu menghitung permutasi dan kombinasi dari setiap pasangan.

UNGUIDED

Soal Studi Case

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 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.

Sourcecode

```
modul4 > unguided 2.go > ...
1 package main
2
3 import {
4     "fmt"
5     "math"
6 }
7
8 func hitungSkor(nama string, waktu []int) (int, int) {
9     totalSoal := 0
10    totalWaktu := 0
11
12    for _, w := range waktu {
13        if w <= 301 {
14            totalSoal++
15            totalWaktu += w
16        }
17    }
18
19    return totalSoal, totalWaktu
20 }
21
22 func main() {
23     var peserta string
24     var maxSoal, minWaktu int
25     var pemenang string
26
27     maxSoal = -1
28     minWaktu = math.MaxInt32
29
30     for {
31         _, err := fmt.Scan(&peserta)
32         if err != nil {
33             break
34         }
35
36         waktu := make([]int, 8)
37         for i := 0; i < 8; i++ {
38             fmt.Scan(&waktu[i])
39         }
40
41         totalSoal, totalWaktu := hitungSkor(peserta, waktu)
42
43         if totalSoal > maxSoal || (totalSoal == maxSoal && totalWaktu < minWaktu) {
44             maxSoal = totalSoal
45             minWaktu = totalWaktu
46             pemenang = peserta
47         }
48     }
49
50     fmt.Printf("%s %d %d\n", pemenang, maxSoal, minWaktu)
51 }
```

Screenshoot Output

Deskripsi Program

Fungsi `hitungSkor` menerima nama peserta dan array waktu sebagai input, lalu menghitung `totalSoal` yang dijawab dan `totalWaktu` untuk soal yang dijawab dalam waktu ≤ 301 detik. Logika Utama: Menggunakan loop untuk membaca nama peserta dan waktu untuk 8 soal, kemudian menghitung skor dengan fungsi `hitungSkor`. Memperbarui pemenang berdasarkan jumlah soal yang dijawab dan waktu total terendah jika jumlahnya sama. Input: Nama peserta diikuti 8 angka yang menunjukkan

waktu untuk setiap soal. Program terus menerima input hingga terjadi kesalahan. Output: Program mencetak nama pemenang, jumlah soal dijawab, dan total waktu.

UNGUIDED

Soal Studi Case

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 $1/2n$, 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 sklena 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 dari 1000000.

Keluaran terdiri dari satu baris saja. Setiap suku dari deret tersebut dicetak dalam baris yang dan dipisahkan oleh sebuah spasi.

Sourcecode

```

go modul 4 1  -go unguided 1.go modul 4 1  -go unguided
modul 4 > -go unguided 3.go > ...
1  package main
2
3  import (
4      "fmt"
5  )
6
7  func cetakDeret(n int) {
8      for n != 1 {
9          fmt.Print(n, " ")
10         if n%2 == 0 {
11             n = n / 2
12         } else {
13             n = 3*n + 1
14         }
15     }
16     fmt.Print(n)
17 }
18
19 func main() {
20     var n int
21     fmt.Scan(&n)
22     if n > 0 && n < 1000000 {
23         cetakDeret(n)
24     }
25 }

```

Screenshoot Output

```

PROBLEMS 46 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\1-Priesty AM\alpro 3 semester 3> go run "d:\1-Priesty AM\alpro 3 semester 3\modul 4\unguided 3.go"
22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS D:\1-Priesty AM\alpro 3 semester 3>

```

Deskripsi Program

Fungsi cetakDeret: Menerima bilangan bulat n dan mencetak nilai n hingga 1 menggunakan loop. Jika n genap, bagi 2; jika ganjil, hitung dengan rumus $n = 3n + 1$. Ulangi hingga n mencapai 1. Logika Utama: Program membaca bilangan bulat n dan memastikan n valid (1-999999). Memanggil fungsi cetakDeret untuk deret Collatz.

Output

Jika pengguna memasukkan $n = 22$, maka deret yang dicetak adalah :

22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1