

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 4  
PROSEDUR**



**Disusun Oleh :**

**Shiva Indah Kurnia**

**2311102035**

**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**

### **A. Definisi Prosedur**

Prosedur dapat di anggap 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:

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

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

### **B. Konsep Prosedur**

Subprogram dengan jenis prosedur merupakan bagian dari kode program yang terpisah yang melakukan tugas spesifik Dimana hasilnya dapat di ketahui setelah semua proses didalamnya selesai dikerjakan. Prosedur perlu dipanggil dalam program utama agar hasil akhir dari sebuah prosedur dapat diketahui. Prosedur dapat di panggil secara berulang kali di program utama. Penggunaan prosedur biasanya ditandai dengan penggunaan void.

### **C. Parametr Prosedur**

Ketika sebuah prosedur dipanggil, maka pada dasarnya dapat melakukan proses pertukaran data antara program utama dan sub program. Proses pertukaran ini dapat dilakukan dengan penggunaan parameter. Pada prosedur terdapat parameter yaitu sebagai berikut:

1. Parameter aktual Parameter aktual merupakan parameter yang disertakan pada saat prosedur dipanggil untuk dijalankan pada program utama, atau dapat sering jenis ini sebagai argument
2. Parameter formal Parameter formal merupakan parameter yang dituliskan pada saat melakukan pendefinisian sebuah prosedur. Parameter ini ditulis pada bagian kode sub program untuk menerima nilai dari parameter aktual pada program utama

Parameter juga dikelompokkan berdasarkan alokasi memorinya, yaitu pass by value dan pass by reference.

1. Pass by Value Nilai pada parameter aktual akan disalin ke variabel lokal (parameter formal) pada subprogram. Artinya parameter aktual dan formal dialokasikan di dalam memori komputer dengan alamat memori yang berbeda. Subprogram dapat menggunakan nilai pada parameter formal tersebut untuk proses apapun, tetapi tidak dapat mengembalikan informasinya ke pemanggil melalui parameter aktual karena pemanggil tidak dapat mengakses memori yang digunakan oleh subprogram. Pass by value bisa digunakan baik oleh fungsi ataupun prosedur.
2. Pass by Reference (Pointer) Ketika parameter didefinisikan sebagai pass by reference, maka pada saat pemanggilan parameter formal akan berperan sebagai pointer yang menyimpan alamat memori dari parameter aktual. Sehingga perubahan nilai yang terjadi pada parameter formal tersebut akan berdampak pada parameter aktual. Artinya nilai terakhirnya akan dapat diketahui oleh si pemanggil setelah subprogram tersebut selesai dieksekusi. Pass by reference sebaiknya digunakan hanya untuk prosedur

#### **D. Pemanggilan Parameter pada Prosedur**

Banyaknya parameter tidak dibatasi jumlahnya, namun dalam implementasi prosedur yang baik, jumlah parameter dalam sebuah prosedur sebaiknya paling banyak adalah tujuh buah. Jika parameter terlalu banyak, maka sebaiknya ada pemisahan lagi menjadi beberapa prosedur. Pemanggilan nama prosedur dapat dilakukan dengan menuliskan nama prosedur yang terdefiniskan. Pemanggilan terhadap prosedur dapat juga menambahkan nilai parameter aktual (argumen) untuk diproses pada prosedur yang menggunakan parameter formal. Urutan nilai parameter aktual (argumen) harus sama dengan urutan pada parameter formal, karena type pada parameter aktual (argumen) dan parameter formal harus sama. Berikut ini merupakan contoh gambaran dari sebuah pseudocode untuk pemanggilan sebuah prosedur

## II. GUIDED

### 1. Guided 1 Soal Studi Case

Buatlah sebuah program menggunakan prosedur yang digunakan untuk menghitung nilai faktorial dan permutasi. Masukan terdiri dari dua buah bilangan positif  $a$  dan  $b$ . Keluaran berupa sebuah bilangan bulat yang menyatakan nilai  $a$  permutasi  $b$  apabila  $a \geq b$  atau  $b$  permutasi  $a$  untuk kemungkinan yang lain.

#### Sourcecode

```
package main

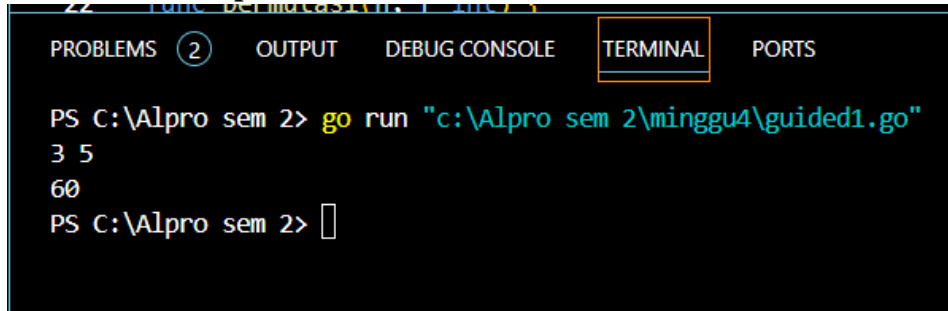
import "fmt"

func main() {
    var a, b int
    fmt.Scan(&a, &b) // Membaca input dari pengguna untuk
    dua nilai integer a dan b
    if a >= b {
        permutasi(a, b)
    } else {
        permutasi(b, a)
    }
}

func faktorial(n int) int {    var hasil int = 1
    // Loop untuk menghitung faktorial dari n
    for i := 1; i <= n; i++ {
        hasil *= i
    }
    return hasil
}

func permutasi(n, r int) {
    // Menghitung permutasi nPr dan langsung mencetak
    hasilnya
    hasil := faktorial(n) / faktorial(n-r)
    fmt.Println(hasil)
}
```

## Screenshoot Output



The screenshot shows a terminal window with a dark background. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and highlighted with an orange border), and 'PORTS'. Below the tabs, the terminal displays the following text: 'PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu4\guided1.go"', followed by the output '3 5' and '60'. The prompt 'PS C:\Alpro sem 2>' is shown again at the bottom with a cursor.

## Deskripsi Program

Script di atas adalah program dalam bahasa Go yang menghitung dan mencetak hasil permutasi dari dua bilangan integer yang dimasukkan oleh pengguna.

### 1. Fungsi main:

- Program membaca dua input bilangan integer, yaitu `a` dan `b`, dari pengguna.
- Jika `a` lebih besar atau sama dengan `b`, fungsi `permutasi(a, b)` dipanggil. Jika tidak, fungsi `permutasi(b, a)` yang dipanggil.

### 2. Fungsi factorial :

- Fungsi ini menghitung faktorial dari bilangan `n` menggunakan loop, mengalikan angka-angka dari 1 hingga `n`, lalu mengembalikan hasilnya.

### 3. Fungsi permutasi :

- Fungsi ini menghitung permutasi `nPr` menggunakan rumus faktorial, yaitu `faktorial(n) / faktorial(n-r)`, dan hasilnya langsung dicetak ke layar.

Secara keseluruhan, program ini membaca dua bilangan, menentukan permutasi mereka, dan menampilkan hasilnya menggunakan fungsi yang terpisah untuk perhitungan faktorial dan permutasi.

## 2. Guided 2 Soal Studi Case

Program beserta prosedur yang digunakan untuk menghitung luas dan keliling persegi. Masukan terdiri dari sisi persegi. Keluaran berupa hasil luas dan keliling persegi

### Sourcecode

```
package main

import "fmt"

// Prosedur untuk menghitung dan mencetak luas persegi
func hitungLuas(sisi float64) {
    luas := sisi * sisi
    fmt.Printf("Luas persegi: %.2f\n", luas)
}

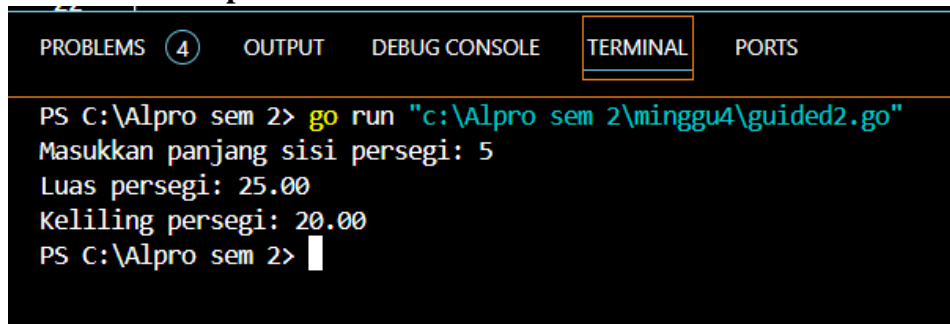
// Prosedur untuk menghitung dan mencetak keliling persegi
func hitungKeliling(sisi float64) {
    keliling := 4 * sisi
    fmt.Printf("Keliling persegi: %.2f\n", keliling)
}

func main() {
    var sisi float64

    fmt.Print("Masukkan panjang sisi persegi: ")
    fmt.Scan(&sisi)

    hitungLuas(sisi)
    hitungKeliling(sisi)
}
```

## Screenshoot Output



```
PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu4\guided2.go"
Masukkan panjang sisi persegi: 5
Luas persegi: 25.00
Keliling persegi: 20.00
PS C:\Alpro sem 2> 
```

### Deskripsi Program

Program di atas merupakan aplikasi sederhana dalam bahasa Go yang digunakan untuk menghitung dan mencetak luas serta keliling persegi berdasarkan panjang sisi yang dimasukkan oleh pengguna.

#### 1. Prosedur hitungLuas:

- Menghitung luas persegi menggunakan rumus  $sisi * sisi$ .
- Hasil luas dicetak dalam format dua angka desimal.

#### 2. Prosedur hitungKeliling:

- Menghitung keliling persegi menggunakan rumus  $4 * sisi$ .
- Hasil keliling juga dicetak dalam format dua angka desimal.

#### 3. Fungsi main:

- Program meminta pengguna untuk memasukkan panjang sisi persegi.
- Setelah itu, program memanggil prosedur hitungLuas dan hitungKeliling untuk menghitung serta menampilkan hasil luas dan keliling berdasarkan sisi yang dimasukkan.

Secara keseluruhan, program ini menghitung luas dan keliling persegi, lalu menampilkan hasilnya ke layar.

### III. UNGUIDED

#### 1. 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  $a \geq c$  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!

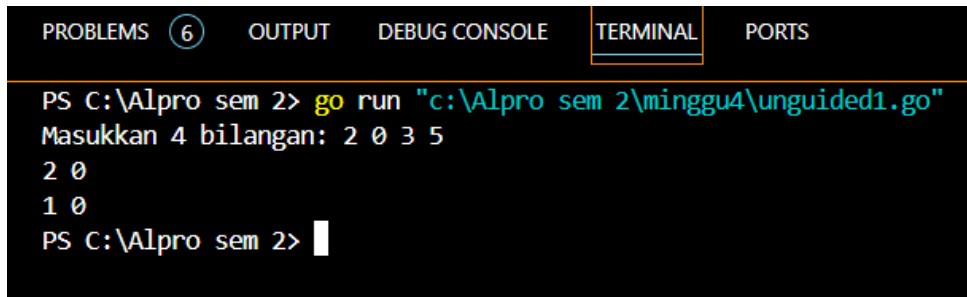
$$P(n, r) = \frac{n!}{(n-r)!}, \text{ sedangkan } C(n, r) = \frac{n!}{r!(n-r)!}$$



## Sourcecode

```
package main
import (    "fmt"
)
// Prosedur untuk menghitung faktorial
func factorial(n int, result *int) {
    *result = 1
    if n == 0 {
        *result = 1
        return
    }
    for i := 1; i <= n; i++ {
        *result *= i
    }
}
// Prosedur untuk menghitung permutasi
func permutation(n, r int, result *int) {
    var fn, fnr int
    factorial(n, &fn)
    factorial(n-r, &fnr)
    *result = fn / fnr
}
// Prosedur untuk menghitung kombinasi
func combination(n, r int, result *int) {
    var fn, fr, fnr int
    factorial(n, &fn)
    factorial(r, &fr)
    factorial(n-r, &fnr)
    *result = fn / (fr * fnr)
}
func main() {
    var a, b, c, d int
    fmt.Print("Masukkan 4 bilangan: ")
    fmt.Scan(&a, &b, &c, &d)
    var p1, c1, p2, c2 int
    // Baris pertama permutasi dan kombinasi a terhadap c
    permutation(a, c, &p1)
    combination(a, c, &c1)
    fmt.Printf("%d %d\n", p1, c1)
    // Baris kedua permutasi dan kombinasi b terhadap d
    permutation(b, d, &p2)
    combination(b, d, &c2)
    fmt.Printf("%d %d\n", p2, c2) }
```

## Screenshot Output



```
PROBLEMS (6) OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu4\unguided1.go"
Masukkan 4 bilangan: 2 0 3 5
2 0
1 0
PS C:\Alpro sem 2> 
```

## Deskripsi Program

Program di atas adalah aplikasi dalam bahasa Go yang menghitung permutasi dan kombinasi dari dua set bilangan yang dimasukkan oleh pengguna.

### 1. Prosedur factorial:

- Menghitung faktorial dari bilangan `n` menggunakan loop. Nilai faktorial disimpan dalam variabel yang ditunjuk oleh pointer result.

### 2. Prosedur permutation:

- Menghitung permutasi  $nPr$ , menggunakan rumus  $\text{factorial}(n) / \text{factorial}(n-r)$ , dan hasilnya disimpan dalam variabel result.

### 3. Prosedur combination:

- Menghitung kombinasi  $nCr$  dengan rumus  $\text{factorial}(n) / (\text{factorial}(r) * \text{factorial}(n-r))$ , kemudian hasilnya disimpan dalam variabel result.

### 4. Fungsi main:

- Program meminta pengguna memasukkan 4 bilangan: `a`, `b`, `c`, dan `d`.
- Program menghitung dan menampilkan permutasi dan kombinasi untuk pasangan `a` dengan `c`, serta `b` dengan `d`.
- Hasil perhitungan permutasi dan kombinasi dicetak dalam dua baris terpisah.

Secara keseluruhan, program ini membaca input dari pengguna, menghitung permutasi dan kombinasi, lalu menampilkan hasil perhitungannya ke layar.

## 2. Unguided 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 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. 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"
    "strings"
)

const maxTime = 301

// Prosedur untuk menghitung skor dan total waktu
func hitungSkor(soal [8]int, totalSoal *int, totalWaktu *int) {
    *totalSoal = 0
    *totalWaktu = 0
    for _, waktu := range soal {
        if waktu < maxTime {
            *totalSoal++
            *totalWaktu += waktu
        }
    }
}
```

```

}

func main() {
    var nama, pemenang string
    var soal [8]int
    var totalSoal, totalWaktu int
    maxSoal := -1
    minWaktu := maxTime * 8

    for {
        fmt.Print("Masukkan nama peserta (atau selesai jika ingin mengakhiri): ")
        fmt.Scan(&nama)
        if strings.ToLower(nama) == "selesai" {
            break
        }

        fmt.Println("Masukkan waktu pengerjaan: ")
        for i := 0; i < 8; i++ {
            fmt.Scan(&soal[i])
        }

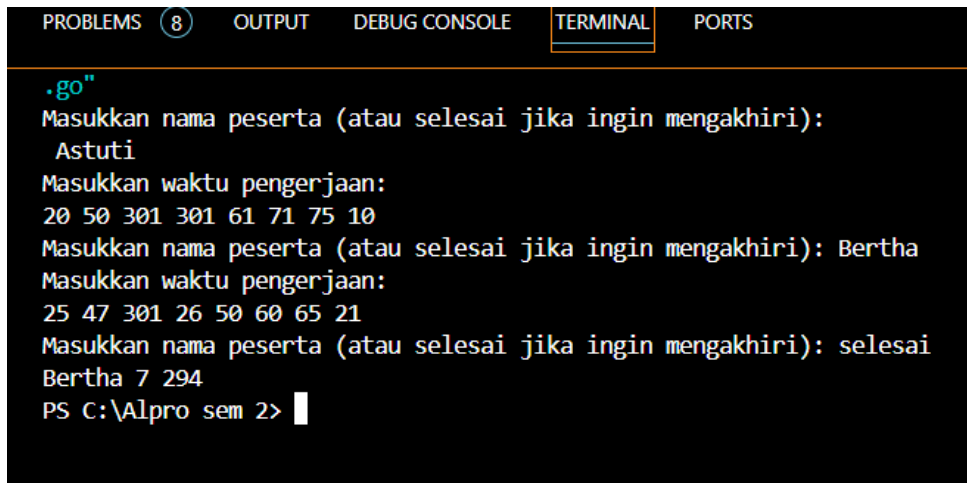
        // Hitung total soal yang diselesaikan dan total waktu yang dibutuhkan
        hitungSkor(soal, &totalSoal, &totalWaktu)

        if totalSoal > maxSoal || (totalSoal == maxSoal && totalWaktu < minWaktu) {
            maxSoal = totalSoal
            minWaktu = totalWaktu
            pemenang = nama
        }
    }

    fmt.Printf("%s %d %d\n", pemenang, maxSoal, minWaktu)
}

```

## Screenshot Output



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are tabs labeled 'PROBLEMS (8)', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and highlighted with a yellow border), and 'PORTS'. The terminal content shows the execution of a program. It starts with a prompt '.go' followed by several input prompts and user inputs. The prompts are 'Masukkan nama peserta (atau selesai jika ingin mengakhiri):', 'Masukkan waktu pengerjaan:', and 'Masukkan nama peserta (atau selesai jika ingin mengakhiri):'. The user inputs are 'Astuti', '20 50 301 301 61 71 75 10', 'Bertha', '25 47 301 26 50 60 65 21', and 'selesai'. The program then outputs 'Bertha 7 294'. The prompt 'PS C:\Alpro sem 2>' is visible at the bottom.

```
.go"
Masukkan nama peserta (atau selesai jika ingin mengakhiri):
Astuti
Masukkan waktu pengerjaan:
20 50 301 301 61 71 75 10
Masukkan nama peserta (atau selesai jika ingin mengakhiri): Bertha
Masukkan waktu pengerjaan:
25 47 301 26 50 60 65 21
Masukkan nama peserta (atau selesai jika ingin mengakhiri): selesai
Bertha 7 294
PS C:\Alpro sem 2>
```

### Deskripsi Program

Kode di atas digunakan untuk memilih pemenang dalam kompetisi pemrograman di mana peserta harus menyelesaikan setidaknya 8 soal dalam waktu 5 jam, atau 301 menit untuk setiap soal. Jumlah soal yang berhasil diselesaikan harus lebih dari 301 menit, dan program Skor menerima nama peserta dan waktu yang diperlukan untuk menyelesaikan delapan soal di bagian utama (main). Program tidak dapat menerima data jika nama yang dimasukkan adalah "selesai". Setiap peserta diberi waktu untuk menyelesaikan setiap soal, dan metode hitung skor menghitung jumlah soal yang diselesaikan dan jumlah waktu yang dihabiskan. Program akan memberikan informasi tentang pemenang sementara berdasarkan siapa yang menyelesaikan soal tambahan. Jika dua orang di antara mereka menyelesaikan jumlah

### 3. Unguided 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. 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 dan dipisahkan oleh sebuah spasi.

### Sourcecode

```
package main

import (
    "fmt"
    "os"
)

// Fungsi untuk menghitung langkah Collatz
func hitungCollatz(n int) ([]int, int) {
    deret := []int{n}
    langkah := 0

    for n != 1 {
        if n%2 == 0 {
            n = n / 2
        } else {
            n = 3*n + 1
        }
        deret = append(deret, n)
        langkah++
    }

    return deret, langkah
}

// Prosedur untuk mencetak deret
func cetakDeret(deret []int) {
    for i, nilai := range deret {
        if i == len(deret)-1 {
            fmt.Printf("%d", nilai)
        } else {
            fmt.Printf("%d ", nilai)
        }
    }
    fmt.Println()
}

// Fungsi untuk memvalidasi input
func validasiInput(n int) bool {
    return n > 0
}
```

```

}

func main() {
    var n int

    fmt.Print("Masukkan bilangan positif: ")
    _, err := fmt.Scan(&n)

    if err != nil || !validasiInput(n) {
        fmt.Println("Input tidak valid. Mohon masukkan
bilangan bulat positif.")
        os.Exit(1)
    }

    deret, langkah := hitungCollatz(n)

    fmt.Printf("\nDeret Collatz untuk %d:\n", n)
    cetakDeret(deret)
    fmt.Printf("Jumlah langkah: %d\n", langkah)
}

```

### Screenshot Output

```

PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu4\unguided3.go"
Masukkan bilangan positif: 22

Deret Collatz untuk 22:
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
Jumlah langkah: 15
PS C:\Alpro sem 2> 

```

### Deskripsi Program

Kode di atas adalah program yang memiliki kemampuan untuk mencetak deret bilangan berdasarkan aturan dari "Collatz Conjecture", atau masalah  $3n$  plus 1. Metode cetak adalah dengan memasukkan bilangan bulat  $n$  ke dalam daftar dan mencetaknya. Jika  $n$  adalah bilangan genap, nilai  $n$  dibagi dua dan jika  $n$  adalah bilangan ganjil, dihitung dengan rumus  $3n+1$ . Proses ini berulang hingga nilai  $n$  menjadi 1. Nilai ini juga akan dicetak sebagai elemen terakhir dari deret. Program meminta pengguna memasukkan bilangan awal  $n$  pada bagian utama. Setelah program menerima input, dia memanggil prosedur `cetakDeret` untuk mencetak deret angka hingga mencapai angka 1. Setiap angka dipisahkan oleh spasi.

#### **IV. KESIMPULAN**

Prosedur, juga dikenal sebagai subprogram dalam pemrograman, digunakan untuk melakukan tugas tertentu tanpa mengembalikan nilai langsung, seringkali hanya mencetak hasilnya. Teknik ini membuat program lebih mudah dipahami dan dipelihara karena membaginya menjadi bagian yang lebih kecil dan modular. Selama proses, data dapat dikirim melalui parameter. Salah satu cara untuk melakukan ini adalah dengan mengirimkan melalui nilai (hanya menyalin nilai) atau melalui referensi (mengakses langsung memori asli). Memecah logika kompleks menjadi bagian yang lebih kecil dan dapat digunakan kembali meningkatkan efisiensi dalam menyelesaikan tugas yang berulang, seperti menghitung permutasi, kombinasi, atau faktorial.

#### **V. REFERENSI**

- [1] Modul 4 Praktikum Algoritma 2
- [2] Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi Republik Indonesia. (2024). Modul PB 10 - Prosedur. Diakses dari <https://lmsspada.kemdikbud.go.id>.