

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL III DAN IV
FUNGSI DAN PROSEDUR**



Disusun Oleh:

NAMA : Titanio Francy Naddiansa

NIM: 2311102289

KELAS: IF-11-07

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Fungsi adalah blok kode yang dirancang untuk menjalankan tugas tertentu dan dapat dipanggil berkali-kali dalam program. Fungsi biasanya menerima input berupa parameter dan dapat mengembalikan hasil dalam bentuk output. Fungsi membantu memecah tugas besar menjadi bagian-bagian yang lebih kecil, sehingga program menjadi lebih terstruktur dan mudah dipelihara. Salah satu keunggulan utama fungsi adalah kemampuannya untuk mengembalikan nilai, yang bisa digunakan oleh bagian lain dari program untuk melakukan perhitungan atau operasi lebih lanjut. Dengan demikian, fungsi dapat mengurangi duplikasi kode dan meningkatkan keterbacaan serta efisiensi program.

Sementara itu, prosedur adalah jenis blok kode serupa, tetapi tidak mengembalikan nilai. Prosedur sering digunakan untuk tugas-tugas tertentu yang tidak memerlukan pengembalian hasil, seperti menampilkan output di layar, memodifikasi variabel global, atau mengontrol aliran program. Meskipun prosedur tidak mengembalikan nilai, ia tetap penting dalam situasi di mana tugas perlu dilakukan tanpa memerlukan hasil perhitungan.

Baik fungsi maupun prosedur mendukung modularitas, yaitu pemisahan logika program ke dalam bagian-bagian kecil yang independen. Keduanya juga mendukung prinsip reusability, di mana kode yang sama dapat digunakan kembali dalam konteks yang berbeda. Dengan menggunakan fungsi dan prosedur, pengembangan program menjadi lebih efisien, mudah diuji, dan lebih terorganisir, terutama ketika program semakin kompleks.

II. GUIDED

1. Source Code

```
package main
```

```
import (
```

```
    "fmt"
```

```
)
```

```
func hitungVolumeTabung(jariJari, tinggi int) float64 {
```

```
    const pi = 3.14
```

```
    luasAlas := pi * float64(jariJari*jariJari)
```

```
    volume := luasAlas * float64(tinggi)
```

```
    return volume
```

```
}
```

```
func main() {
```

```
    var jariJari1, tinggi1, jariJari2 int
```

```
    var volume1, volume2 float64
```

```
    jariJari1 = 5
```

```
    tinggi1 = 10
```

```
    jariJari2 = 15
```

```
    volume1 = hitungVolumeTabung(jariJari1, tinggi1)
```

```
    volume2 = hitungVolumeTabung(jariJari1, tinggi1) + hitungVolumeTabung(jariJari2, tinggi1)
```

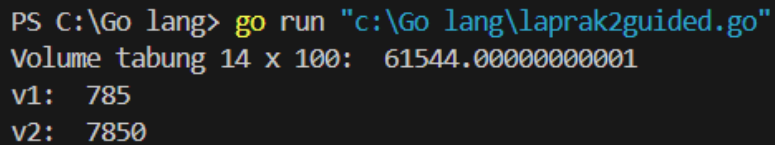
```
    fmt.Println("Volume tabung 14 x 100: ", hitungVolumeTabung(14, 100))
```

```
fmt.Println("Volume tabung pertama (v1):", volume1)

fmt.Println("Total volume dua tabung (v2):", volume2)

}
```

Screenshot Output



```
PS C:\Go lang> go run "c:\Go lang\laprak2guided.go"
Volume tabung 14 x 100: 61544.000000000001
v1: 785
v2: 7850
```

Deskripsi Program

Program ini dibuat untuk menghitung volume tabung berdasarkan jari-jari dan tinggi yang diberikan. Di dalam program, ada fungsi bernama `hitungVolumeTabung` yang menerima dua parameter: `jariJari` dan `tinggi`. Fungsi ini menggunakan rumus sederhana untuk menghitung volume tabung dengan cara mengalikan luas alas dengan tinggi.

Di bagian `main`, kita menginisialisasi dua variabel, yaitu `jariJari1` dan `tinggi1`, dengan nilai 5 dan 10. Kemudian, kita memanggil fungsi `hitungVolumeTabung` untuk menghitung volume tabung pertama dan menyimpan hasilnya di variabel `volume1`. Selain itu, kita juga menghitung volume tabung kedua dengan jari-jari 15 dan tinggi yang sama, lalu total volume dari kedua tabung ini disimpan dalam variabel `volume2`.

Setelah semua perhitungan selesai, program mencetak hasilnya. Ini termasuk volume tabung dengan jari-jari 14 dan tinggi 100, serta nilai dari `volume1` dan `volume2`. Dengan cara ini, program menunjukkan bagaimana kita bisa menghitung dan membandingkan volume tabung dengan ukuran yang berbeda.

2. Source Code

```
package main
```

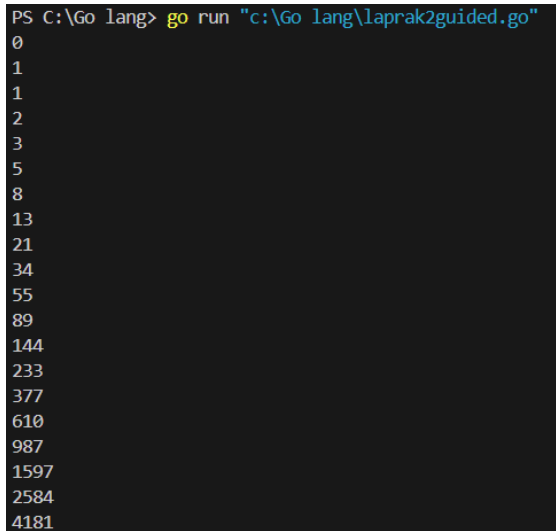
```
import (  
    "fmt"  
)
```

```
func cetakDeretFibonacci(n int) {  
    var a, b, c int  
  
    a = 0  
  
    b = 1  
  
    for i := 0; i < n; i++ {  
        if i == 0 {  
            fmt.Println(a)  
        } else if i == 1 {  
            fmt.Println(b)  
        } else {  
            c = a + b  
  
            fmt.Println(c)  
  
            a = b  
  
            b = c  
        }  
    }  
}
```

```
func main() {  
    var jumlahSuku int
```

```
    jumlahSuku = 20  
    cetakDeretFibonacci(jumlahSuku)  
}
```

Screenshot Output



```
PS C:\Go lang> go run "c:\Go lang\laprak2guided.go"  
0  
1  
1  
2  
3  
5  
8  
13  
21  
34  
55  
89  
144  
233  
377  
610  
987  
1597  
2584  
4181
```

Deskripsi Program

Program ini dibuat untuk mencetak deret Fibonacci sebanyak n suku yang kita tentukan. Jadi, deret Fibonacci itu dimulai dari angka 0 dan 1, dan setiap angka berikutnya adalah hasil penjumlahan dari dua angka sebelumnya. Di dalam program ini, kita punya fungsi yang namanya `cetakDeretFibonacci(n int)`, yang tugasnya untuk menghitung dan menampilkan setiap suku Fibonacci ke layar. Dengan begitu, kita bisa melihat deretnya dengan jelas.

III. UNGUIDED

Modul III

1. Source Code

```
package main

import (
    "fmt"
)

func faktorial(n int) int {
    if n == 0 {
        return 1
    }
    hasil := 1
    for i := 1; i <= n; i++ {
        hasil *= i
    }
    return hasil
}

func permutasi(n, r int) int {
    return faktorial(n) / faktorial(n-r)
}

func kombinasi(n, r int) int {
    return faktorial(n) / (faktorial(r) * faktorial(n-r))
}
```

```

func main() {

    var a, b, c, d int

    fmt.Print("Masukkan nilai a, b, c, d: ")

    fmt.Scan(&a, &b, &c, &d)


    if a >= c && b >= d {

        p1 := permutasi(a, c)

        k1 := kombinasi(a, c)

        p2 := permutasi(b, d)

        k2 := kombinasi(b, d)


        fmt.Printf("Permutasi(%d, %d) = %d, Kombinasi(%d, %d) = %d\n", a, c, p1, a, c, k1)

        fmt.Printf("Permutasi(%d, %d) = %d, Kombinasi(%d, %d) = %d\n", b, d, p2, b, d, k2)

    } else {

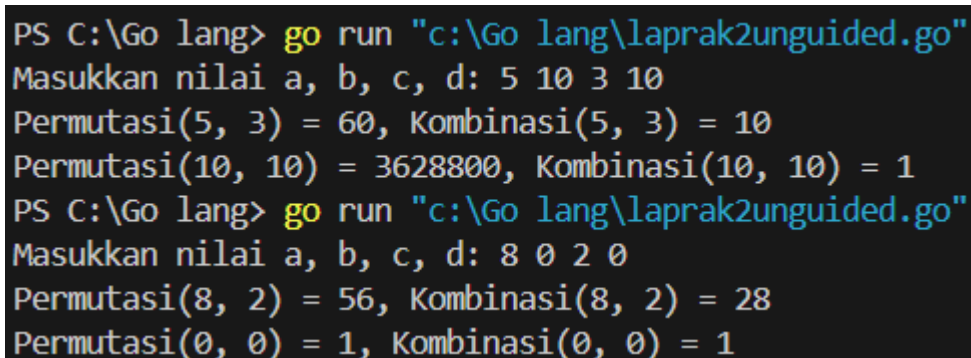
        fmt.Println("Input tidak valid, pastikan a >= c dan b >= d")

    }

}

```

Screenshoot Output



```

PS C:\Go lang> go run "c:\Go lang\laprak2unguided.go"
Masukkan nilai a, b, c, d: 5 10 3 10
Permutasi(5, 3) = 60, Kombinasi(5, 3) = 10
Permutasi(10, 10) = 3628800, Kombinasi(10, 10) = 1
PS C:\Go lang> go run "c:\Go lang\laprak2unguided.go"
Masukkan nilai a, b, c, d: 8 0 2 0
Permutasi(8, 2) = 56, Kombinasi(8, 2) = 28
Permutasi(0, 0) = 1, Kombinasi(0, 0) = 1

```

Deskripsi Program

Program ini dibuat untuk menghitung permutasi dan kombinasi dari dua pasangan nilai yang dimasukkan oleh user. Pertama, program akan meminta user untuk memasukkan empat nilai,

yaitu a, b, c, dan d. Setelah itu, program akan memeriksa apakah nilai a lebih besar atau sama dengan c, dan b lebih besar atau sama dengan d. Jika kondisi ini terpenuhi, program akan menghitung permutasi dan kombinasi untuk setiap pasangan nilai tersebut menggunakan fungsi faktorial, permutasi, dan kombinasi. Hasil perhitungan kemudian ditampilkan di layar. Jika input tidak valid, program akan memberikan pesan peringatan.

2. Source Code

```
package main

import (
    "fmt"
)

func f(x int) int {
    return x * x
}

func g(x int) int {
    return x - 2
}

func h(x int) int {
    return x + 1
}

func main() {
    var a, b, c int
    fmt.Println("Masukkan tiga bilangan bulat (a, b, c):")
    fmt.Scan(&a, &b, &c)
```

hasil1 := f(g(h(a)))

hasil2 := g(h(f(b)))

hasil3 := h(f(g(c)))

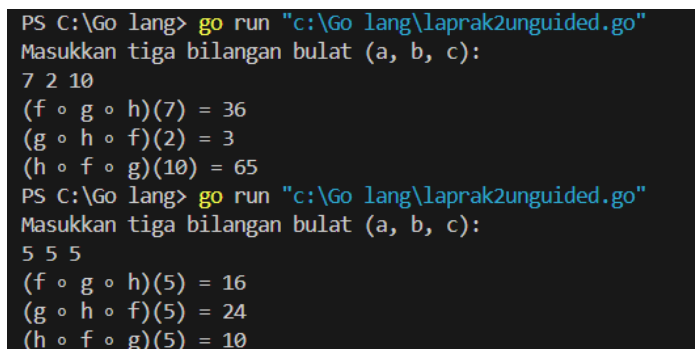
fmt.Printf("(f ◦ g ◦ h)(%d) = %d\n", a, hasil1)

fmt.Printf("(g ◦ h ◦ f)(%d) = %d\n", b, hasil2)

fmt.Printf("(h ◦ f ◦ g)(%d) = %d\n", c, hasil3)

}

Screenshot Output



```
PS C:\Go lang> go run "c:\Go lang\laprak2unguided.go"
Masukkan tiga bilangan bulat (a, b, c):
7 2 10
(f ◦ g ◦ h)(7) = 36
(g ◦ h ◦ f)(2) = 3
(h ◦ f ◦ g)(10) = 65
PS C:\Go lang> go run "c:\Go lang\laprak2unguided.go"
Masukkan tiga bilangan bulat (a, b, c):
5 5 5
(f ◦ g ◦ h)(5) = 16
(g ◦ h ◦ f)(5) = 24
(h ◦ f ◦ g)(5) = 10
```

Deskripsi Program

Program ini dirancang untuk melakukan komposisi fungsi menggunakan tiga fungsi sederhana: f, g, dan h. Fungsi f mengkuadratkan input, g mengurangi input dengan 2, dan h menambahkan 1 pada input. User diminta untuk memasukkan tiga bilangan bulat, yang kemudian diproses melalui komposisi fungsi. Hasil dari komposisi fungsi (f ◦ g ◦ h)(a), (g ◦ h ◦ f)(b), dan (h ◦ f ◦ g)(c) akan dihitung dan ditampilkan ke layar. Dengan program ini, kita dapat melihat bagaimana pengaruh urutan fungsi terhadap nilai input yang diberikan.

3. Source Code

```
package main
```

```
import (  
    "fmt"  
    "math"  
)
```

```
func hitungJarak(x1, y1, x2, y2 float64) float64 {  
    return math.Sqrt(math.Pow(x1-x2, 2) + math.Pow(y1-y2, 2))  
}
```

```
func titikDalamLingkaran(lx, ly, radius, tx, ty float64) bool {  
    return hitungJarak(lx, ly, tx, ty) <= radius  
}
```

```
func main() {  
    var cx1, cy1, r1, cx2, cy2, r2, x, y float64  
  
    fmt.Print("Masukkan pusat lingkaran 1 (cx1, cy1) dan jari-jari (r1): ")  
    fmt.Scan(&cx1, &cy1, &r1)  
  
    fmt.Print("Masukkan pusat lingkaran 2 (cx2, cy2) dan jari-jari (r2): ")  
    fmt.Scan(&cx2, &cy2, &r2)  
  
    fmt.Print("Masukkan koordinat titik (x, y): ")  
    fmt.Scan(&x, &y)
```

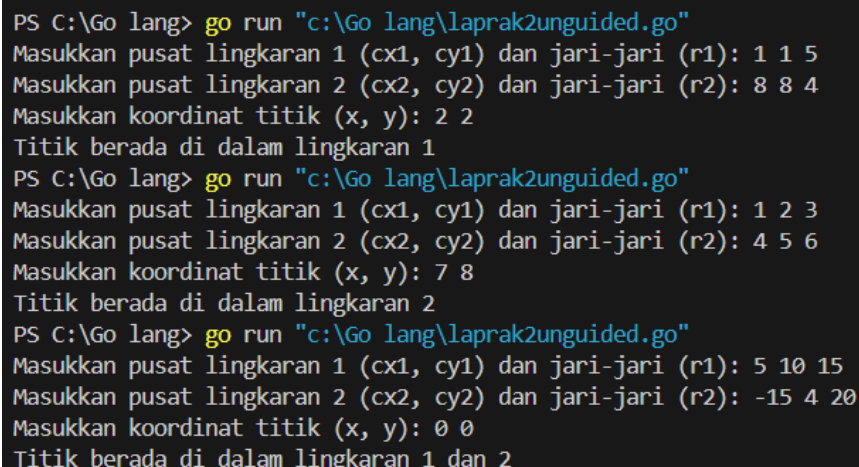
```

adaDiLingkaran1 := titikDalamLingkaran(cx1, cy1, r1, x, y)
adaDiLingkaran2 := titikDalamLingkaran(cx2, cy2, r2, x, y)

if adaDiLingkaran1 && adaDiLingkaran2 {
    fmt.Println("Titik berada di dalam lingkaran 1 dan 2")
} else if adaDiLingkaran1 {
    fmt.Println("Titik berada di dalam lingkaran 1")
} else if adaDiLingkaran2 {
    fmt.Println("Titik berada di dalam lingkaran 2")
} else {
    fmt.Println("Titik berada di luar lingkaran 1 dan 2")
}
}

```

Screenshoot Output



```

PS C:\Go lang> go run "c:\Go lang\laprak2unguided.go"
Masukkan pusat lingkaran 1 (cx1, cy1) dan jari-jari (r1): 1 1 5
Masukkan pusat lingkaran 2 (cx2, cy2) dan jari-jari (r2): 8 8 4
Masukkan koordinat titik (x, y): 2 2
Titik berada di dalam lingkaran 1
PS C:\Go lang> go run "c:\Go lang\laprak2unguided.go"
Masukkan pusat lingkaran 1 (cx1, cy1) dan jari-jari (r1): 1 2 3
Masukkan pusat lingkaran 2 (cx2, cy2) dan jari-jari (r2): 4 5 6
Masukkan koordinat titik (x, y): 7 8
Titik berada di dalam lingkaran 2
PS C:\Go lang> go run "c:\Go lang\laprak2unguided.go"
Masukkan pusat lingkaran 1 (cx1, cy1) dan jari-jari (r1): 5 10 15
Masukkan pusat lingkaran 2 (cx2, cy2) dan jari-jari (r2): -15 4 20
Masukkan koordinat titik (x, y): 0 0
Titik berada di dalam lingkaran 1 dan 2

```

Deskripsi Program

Program ini digunakan untuk menentukan apakah sebuah titik berada di dalam dua lingkaran yang berbeda. User diminta untuk memasukkan pusat dan jari-jari dari dua lingkaran, serta koordinat titik yang ingin dicek. Program ini menghitung jarak antara titik dan pusat lingkaran menggunakan rumus jarak. Kemudian, program akan memberi tahu apakah titik tersebut berada di dalam lingkaran 1, lingkaran 2, atau di luar kedua lingkaran tersebut.

Modul IV

1. Source Code

```
package main
```

```
import (  
    "fmt"  
)
```

```
func faktorial(n int) int {  
    if n == 0 {  
        return 1  
    }  
    hasil := 1  
    for i := 1; i <= n; i++ {  
        hasil *= i  
    }  
    return hasil  
}
```

```
func permutasi(n, r int) int {  
    return faktorial(n) / faktorial(n-r)  
}
```

```
func kombinasi(n, r int) int {  
    return faktorial(n) / (faktorial(r) * faktorial(n-r))  
}
```

```

func main() {

    var a, b, c, d int

    fmt.Println("Masukkan empat bilangan bulat positif a, b, c, dan d:")

    fmt.Scan(&a, &b, &c, &d)


    if a >= c && b >= d {

        p1 := permutasi(a, c)

        k1 := kombinasi(a, c)

        p2 := permutasi(b, d)

        k2 := kombinasi(b, d)


        fmt.Printf("Permutasi(%d, %d) = %d, Kombinasi(%d, %d) = %d\n", a, c, p1, a, c, k1)

        fmt.Printf("Permutasi(%d, %d) = %d, Kombinasi(%d, %d) = %d\n", b, d, p2, b, d, k2)

    } else {

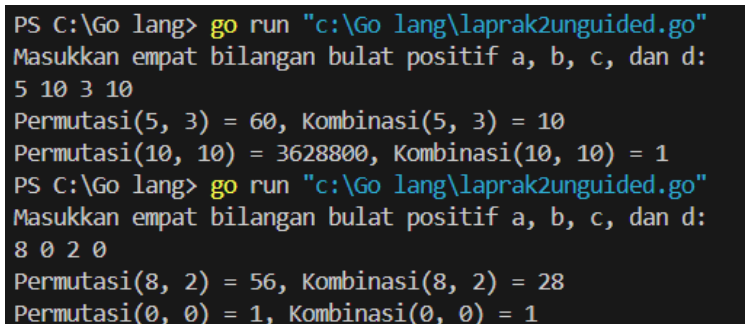
        fmt.Println("Input tidak valid, pastikan a >= c dan b >= d")

    }

}

```

Screenshoot Output



```

PS C:\Go lang> go run "c:\Go lang\laprak2unguided.go"
Masukkan empat bilangan bulat positif a, b, c, dan d:
5 10 3 10
Permutasi(5, 3) = 60, Kombinasi(5, 3) = 10
Permutasi(10, 10) = 3628800, Kombinasi(10, 10) = 1
PS C:\Go lang> go run "c:\Go lang\laprak2unguided.go"
Masukkan empat bilangan bulat positif a, b, c, dan d:
8 0 2 0
Permutasi(8, 2) = 56, Kombinasi(8, 2) = 28
Permutasi(0, 0) = 1, Kombinasi(0, 0) = 1

```

Deskripsi Program

Program ini dibuat untuk menghitung permutasi dan kombinasi dari dua pasangan bilangan bulat positif. User diminta untuk memasukkan empat bilangan bulat, yaitu a, b, c, dan d. Dimana a dan b adalah nilai dari n, sedangkan c dan d adalah nilai dari r untuk menghitung permutasi dan kombinasi. Program akan memastikan bahwa nilai a harus lebih besar atau sama dengan c, dan b harus lebih besar atau sama dengan d. Jika syarat tersebut terpenuhi, program akan menghitung dan menampilkan hasil permutasi dan kombinasi untuk kedua pasangan tersebut. Jika tidak, program akan memberi tahu bahwa input tidak valid.

2. Source Code

```
package main
```

```
import "fmt"
```

```
type Peserta struct {
```

```
    Nama    string
```

```
    Waktu []int
```

```
    Skor    int
```

```
    Selesai int
```

```
}
```

```
func hitungSkor(peserta *Peserta) {
```

```
    peserta.Skor = 0
```

```
    peserta.Selesai = 0
```

```
    for _, waktu := range peserta.Waktu {
```

```
        if waktu == 301 {
```

```
            continue
```

```
        } else {
```

```
            peserta.Skor += waktu
```

```

        peserta.Selesai++
    }
}
}

```

```

func main() {
    peserta := []Peserta{
        {Nama: "Astuti", Waktu: []int{20, 50, 301, 301, 61, 71, 75, 10}},
        {Nama: "Bertha", Waktu: []int{25, 47, 301, 26, 50, 60, 65, 21}},
    }
}

```

```

for i := range peserta {
    hitungSkor(&peserta[i])
}

```

```

var pemenang Peserta
for i := range peserta {
    if peserta[i].Selesai > pemenang.Selesai {
        pemenang = peserta[i]
    } else if peserta[i].Selesai == pemenang.Selesai {
        if pemenang.Skor == 0 || peserta[i].Skor < pemenang.Skor {
            pemenang = peserta[i]
        }
    }
}
}

```

```

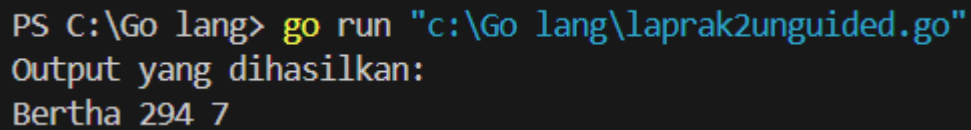
fmt.Printf("Output yang dihasilkan:\n")

```



```
    fmt.Printf("%s %d %d\n", pemenang>Nama, pemenang>.Skor, pemenang>.Selesai)  
}
```

Screenshot Output



```
PS C:\Go lang> go run "c:\Go lang\laprak2unguided.go"  
Output yang dihasilkan:  
Bertha 294 7
```

Deskripsi Program

Program ini dirancang untuk menghitung skor dan menentukan pemenang dari sejumlah peserta berdasarkan waktu yang mereka habiskan. Setiap peserta memiliki nama dan daftar waktu yang mencerminkan durasi yang dihabiskan untuk menyelesaikan soal. Jika waktu yang dihabiskan adalah 301 detik, maka tidak akan ada penambahan skor. Program ini menghitung total skor dan jumlah soal yang selesai untuk setiap peserta. Pemenang ditentukan berdasarkan jumlah soal yang selesai, dan jika ada peserta dengan jumlah soal yang sama, skor terendah yang akan menang. Hasil akhir menampilkan nama pemenang beserta total skornya dan jumlah soal yang berhasil diselesaikan.

3. Source Code

```
package main
```

```
import (  
    "fmt"  
)
```

```
func cetakDeret(n int) {  
    for n != 1 {  
        fmt.Printf("%d ", n)  
        if n%2 == 0 {  
            n = n / 2  
        } else {
```

```

        n = 3*n + 1
    }
}

fmt.Println(n)
}

func main() {
    var n int

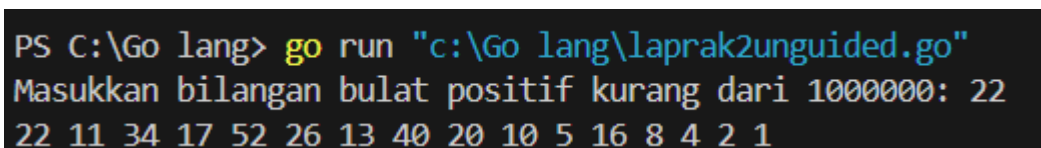
    fmt.Print("Masukkan bilangan bulat positif kurang dari 1000000: ")

    fmt.Scan(&n)

    if n > 0 && n < 1000000 {
        cetakDeret(n)
    } else {
        fmt.Println("Bilangan harus positif dan kurang dari 1000000")
    }
}

```

Screenshot Output



```

PS C:\Go lang> go run "c:\Go lang\laprak2unguided.go"
Masukkan bilangan bulat positif kurang dari 1000000: 22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

```

Deskripsi Program

Program ini mengimplementasikan algoritma Collatz, yang bertujuan untuk menghasilkan deret berdasarkan bilangan bulat positif yang dimasukkan oleh user. User diminta untuk memasukkan bilangan bulat positif yang kurang dari 1.000.000. Jika bilangan tersebut valid, program akan mencetak deret angka hingga mencapai angka 1. Dalam deret ini, jika angka saat ini genap, maka angka tersebut dibagi dua; jika ganjil, maka angka tersebut dikalikan tiga dan ditambah satu. Proses ini berlanjut sampai angka mencapai satu.