

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL IV
PROSEDUR**



Disusun Oleh :

Liya Khoirunnisa / 2311102124

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

Fungsi adalah sekumpulan blok kode yang dibungkus dengan nama tertentu. Penerapan fungsi yang tepat akan menjadikan kode lebih modular dan juga dry (kependekan dari don't repeat yourself), karena tak perlu menuliskan banyak proses berkali-kali, cukup sekali saja dan tinggal panggil jika dibutuhkan. Cara membuat fungsi cukup mudah, yaitu dengan menuliskan keyword `func`, diikuti setelahnya nama fungsi, kurung yang berisikan parameter, dan kurung kurawal untuk membungkus blok kode. Parameter sendiri adalah variabel yang disisipkan pada saat pemanggilan fungsi. Sebuah fungsi bisa didesain tidak mengembalikan apa-apa (prosedur).

Dalam paradigma pemrograman fungsional, fungsi digunakan untuk membuat kode lebih terorganisir. Dengan membagi kode menjadi fungsi yang lebih kecil, setiap bagian dari program dapat dipahami dengan lebih mudah. Fungsi digunakan juga untuk menghindari pengulangan atau redundansi, mempermudah pengujian. Selain itu, fungsi juga mendukung konsep pure functions dan immutability. Dalam pemrograman fungsional, fungsi sering kali tidak memiliki efek samping dan tidak mengubah state program.

Procedure atau prosedur adalah subprogram dalam pemrograman yang terdiri dari serangkaian pernyataan atau langkahlangkah yang didefinisikan sebagai satu kesatuan untuk mengeksekusi tugas tertentu. Prosedur juga disebut modul, subrutin (sub-routine), function atau method, tergantung pada bahasa pemrograman yang digunakan. Setiap istilah ini memiliki makna dan peran yang sedikit berbeda, tetapi semuanya merujuk pada konsep yang sama, yaitu memecah program menjadi bagian-bagian yang lebih kecil dan terstruktur.

Prosedur dapat memiliki parameter ataupun tanpa parameter yang memungkinkan membuat sebuah prosedur yang dapat menerima satu atau lebih argumen sebagai input dan mengembalikan

Perbedaan fungsi dan prosedur terletak pada tugasnya. Fungsi hanya akan mengembalikan satu nilai yang artinya hanya mengerjakan satu tugas saja. Sedangkan prosedur dapat mengembalikan lebih dari satu nilai atau tidak mengembalikan nilai sama sekali yang artinya dapat mengerjakan lebih dari satu tugas.

Dengan pendekatan ini, pemrogram dapat mengembangkan program yang lebih mudah dipahami, dikelola, dan disusun. Prosedur dapat digunakan kembali dalam berbagai bagian program atau dalam program lain, memungkinkan efisiensi dalam pengembangan perangkat lunak. Sebuah prosedur memiliki nama yang digunakan untuk memanggilnya dan sering menerima parameter sebagai input.

II. GUIDED

1. Buatlah sebuah program beserta fungsi 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 permutasi a untuk kemungkinan yang lain

Source code

```
/* Liya Khoirunnisa - 2311102124*/

package main

import "fmt"

func main() {
    // Deklarasi variabel
    var a, b, hasil int

    // Meminta inputan dari pengguna
    fmt.Print("Masukkan dua angka: ")

    // Menyimpan inputan
    fmt.Scan(&a, &b)

    // Mengecek angka yang diinputkan pengguna
    if a >= b {
        // Jika hasil lebih besar atau sama dengan b
        permutasi(&a, &b, &hasil)

        // Cetak hasil
        fmt.Println("Hasil permutasi:", hasil)
    } else {
        // Jika a lebih kecil dari b
        permutasi(&b, &a, &hasil)

        // Cetak hasil
        fmt.Println("Hasil permutasi:", hasil)
    }
}

// Fungsi menghitung faktorial
func faktorial(n *int, hasil *int) {
    // Inisialisai
    *hasil = 1

    // Perulangan faktorial
    for i := 1; i <= *n; i++ {
        *hasil = *hasil * i
    }
}

// Fungsi menghitung permutasi
func permutasi(n, r, hasil *int) {
```

```

var faktorialN, faktorialNR int

// Menghitung faktorial dari n
faktorial(n, &faktorialN)

// Menghitung n-r
nr := *n - *r

// Menghitung faktorial dari n-r
faktorial(&nr, &faktorialNR)

// Hasil permutasi
*hasil = faktorialN / faktorialNR
}

```

Screenshoot Output

```

PROBLEMS 40 OUTPUT TERMINAL ... Code + v [] [X] ... ^
PS D:\Prak Alpro 2> go run "d:\Prak Alpro 2\laprakModul4\guided1.go"
Masukkan dua angka: 2 3
Hasil permutasi: 6
PS D:\Prak Alpro 2> go run "d:\Prak Alpro 2\laprakModul4\guided1.go"
Masukkan dua angka: 1 1
Hasil permutasi: 1
PS D:\Prak Alpro 2> go run "d:\Prak Alpro 2\laprakModul4\guided1.go"
Masukkan dua angka: 1 0
Hasil permutasi: 1
PS D:\Prak Alpro 2> go run "d:\Prak Alpro 2\laprakModul4\guided1.go"
Masukkan dua angka: 2 1
Hasil permutasi: 2
PS D:\Prak Alpro 2> go run "d:\Prak Alpro 2\laprakModul4\guided1.go"
Masukkan dua angka: 2 4
Hasil permutasi: 12
PS D:\Prak Alpro 2> 

```

Deskripsi Program

Program di atas dibuat untuk menghitung permutasi dari dua bilangan bulat. Pertama meminta pengguna untuk memasukkan dua angka dan membandingkan dua angka tersebut menggunakan percabangan if. Perhitungan diawali dengan menghitung faktorial dari suatu angka setelah itu menghitung permutasi dengan rumus yang ada. Dalam menghitung faktorial dan permutasi menggunakan prosedur. Setelah perhitungan selesai, program akan mencetak hasilnya ke layar.

***Note:** Pada terminal terdapat 40 problem dikarenakan package dan func main digunakan juga di file lain

2. Program untuk menghitung keliling dan luas dari persegi

Source code

```
/* Liya Khoirunnisa - 2311102124*/

package main

import (
    "fmt"
)

// Prosedur untuk menghitung dan menampilkan luas dan
// keliling persegi
func hitungPersegi(sisi float64) {
    // Menghitung luas persegi
    luas := sisi * sisi

    // Menghitung keliling persegi
    keliling := 4 * sisi

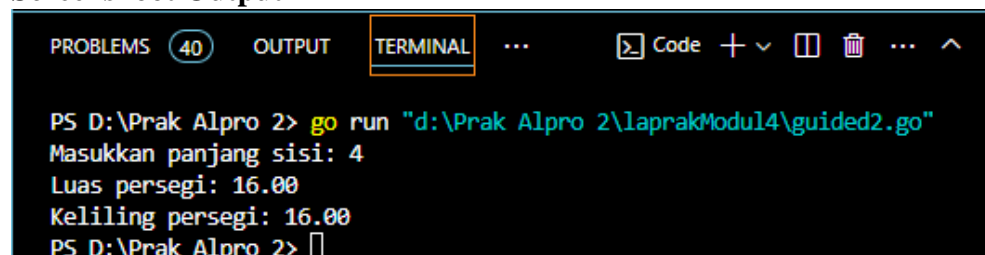
    // Hasil
    fmt.Printf("Luas persegi: %.2f\n", luas)
    fmt.Printf("Keliling persegi: %.2f\n", keliling)
}

func main() {
    // Deklarasi variabel
    var sisi float64

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

    // Menghitung dan menampilkan luas dan keliling persegi
    hitungPersegi(sisi)
}
```

Screenshot Output



```
PROBLEMS 40 OUTPUT TERMINAL ... Code + - [] [X] ... ^

PS D:\Prak Alpro 2> go run "d:\Prak Alpro 2\laprakModul4\guided2.go"
Masukkan panjang sisi: 4
Luas persegi: 16.00
Keliling persegi: 16.00
PS D:\Prak Alpro 2> 
```

Deskripsi Program

Program di atas dibuat untuk menghitung luas dan keliling sebuah persegi. Pengguna diminta untuk menginputkan panjang sisi persegi. Sisi yang telah diinputkan digunakan untuk menghitung luas dengan rumus $s \times s$ dan keliling dengan rumus $4 \times s$. Dalam menghitung luas dan keliling persegi menggunakan 1 prosedur. Hasil perhitungan tersebut ditampilkan dengan desimal yang memiliki 2 angka dibelakang koma.

***Note:** Pada terminal terdapat 40 problem dikarenakan package dan func main digunakan juga di file lain

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? 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!

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

Selesaikan program dengan memanfaatkan prosedur yang diberikat berikut ini

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}

Source code

```
/* Liya Khoirunnisa - 2311102124 */

package main

import "fmt"

func main() {
    // Deklarasi variabel
    var a, b, c, d, permutasiAC, kombinasiAC, permutasiBD,
    kombinasiBD int

    // Inputan dari pengguna
    fmt.Print("Masukkan angka a: ")
    fmt.Scan(&a)

    fmt.Print("Masukkan angka b: ")
    fmt.Scan(&b)

    fmt.Print("Masukkan angka c: ")
```

```

    fmt.Scan(&c)

    fmt.Print("Masukkan angka d: ")
    fmt.Scan(&d)

    // Mengecek apakah a >= c dan b >= d
    if a >= c && b >= d {
        // Menghitung permutasi dan kombinasi a terhadap c
        permutasi(&a, &c, &permutasiAC)
        kombinasi(&a, &c, &kombinasiAC)

        // Menghitung permutasi dan kombinasi b terhadap d
        permutasi(&b, &d, &permutasiBD)
        kombinasi(&b, &d, &kombinasiBD)

        // Cetak hasil untuk a terhadap c
        fmt.Println("Hasil permutasi a terhadap c:",
permutasiAC)
        fmt.Println("Hasil kombinasi a terhadap c:",
kombinasiAC)

        // Cetak hasil untuk b terhadap d
        fmt.Println("Hasil permutasi b terhadap d:",
permutasiBD)
        fmt.Println("Hasil kombinasi b terhadap d:",
kombinasiBD)
    } else {
        fmt.Println("Syarat a >= c dan b >= d harus
terpenuhi.")
    }
}

// Fungsi menghitung faktorial
func faktorial(n *int, hasil *int) {
    // Inisialisasi
    *hasil = 1

    // Perulangan faktorial
    for i := 1; i <= *n; i++ {
        *hasil = *hasil * i
    }
}

// Fungsi menghitung permutasi
func permutasi(n, r, hasil *int) {
    var faktorialN, faktorialNR int

    // Menghitung faktorial dari n
    faktorial(n, &faktorialN)

    // Menghitung n-r
    nr := *n - *r

    // Menghitung faktorial dari n-r
    faktorial(&nr, &faktorialNR)

```



```

    // Hasil permutasi
    *hasil = faktorialN / faktorialNR
}

// Fungsi menghitung kombinasi
func kombinasi(n, r, hasil *int) {
    var faktorialN, faktorialR, faktorialNR int

    // Menghitung faktorial dari n
    faktorial(n, &faktorialN)

    // Menghitung faktorial dari r
    faktorial(r, &faktorialR)

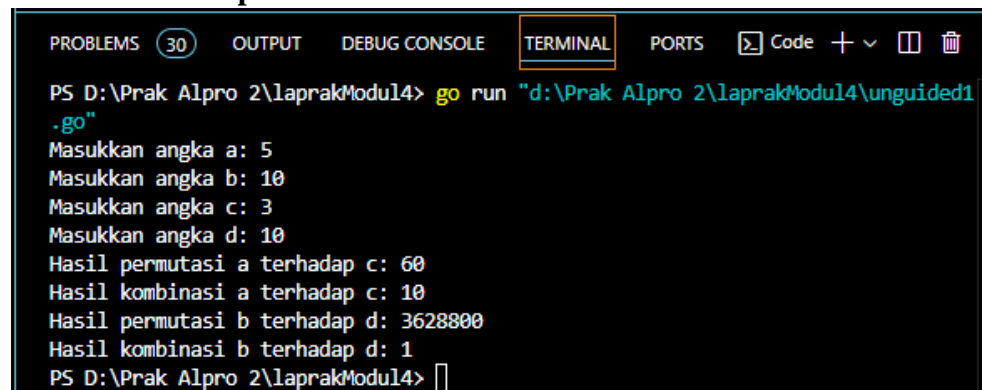
    // Menghitung n-r
    nr := *n - *r

    // Menghitung faktorial dari n-r
    faktorial(&nr, &faktorialNR)

    // Hasil kombinasi
    *hasil = faktorialN / (faktorialR * faktorialNR)
}

```

Screenshoot Output



```

PROBLEMS (30) OUTPUT DEBUG CONSOLE TERMINAL PORTS Code + v [] 
PS D:\Prak Alpro 2\laprakModul4> go run "d:\Prak Alpro 2\laprakModul4\unguided1
.go"
Masukkan angka a: 5
Masukkan angka b: 10
Masukkan angka c: 3
Masukkan angka d: 10
Hasil permutasi a terhadap c: 60
Hasil kombinasi a terhadap c: 10
Hasil permutasi b terhadap d: 3628800
Hasil kombinasi b terhadap d: 1
PS D:\Prak Alpro 2\laprakModul4> 

```

Deskripsi Program

Program di atas dibuat untuk menghitung permutasi dan kombinasi berdasarkan input pengguna. Pengguna diminta menginputkan angka dengan syarat $a \geq c$ dan $b \geq d$. Program akan mengecek apakah angka yang diinputkan sesuai dengan syarat yang ada menggunakan percabangan if. Jika angka tersebut sudah sesuai, maka program akan menghitung faktorial, permutasi dan kombinasi dengan rumus yang ada. Untuk menghitung faktorial, permutasi, dan kombinasi menggunakan prosedur. Setelah dihitung, hasil dicetak ke layar.

***Note:** Pada terminal terdapat 30 problem dikarenakan package dan func main digunakan juga di file lain

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.

procedure 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.

Source code

```
/* Liya Khoirunnisa - 2311102124*/

package main

import (
    "fmt"
    "strings"
)

// Waktu maksimum (5 jam 1 menit)
const maksWaktu = 301

func hitungSkor(jumlahSoal *int, totalSkor *int, dikerjakan
int) {
    var waktu int

    // Perulangan waktu soal
```

```

        for i := 0; i < dikerjakan; i++ {
            _, error := fmt.Scan(&waktu)
            if error != nil {
                fmt.Println("Kesalahan input:", error)
                return
            }

            // Hanya menghitung soal yang dikerjakan dalam waktu
            kurang dari batas maksimum
            if waktu < maksWaktu {
                *totalSkor += waktu
                *jumlahSoal++
            }
        }
    }

func main() {
    // Insialisai
    pemenang := ""
    soalTerbanyak := 0
    totalWaktu := maksWaktu

    // Deklarasi variabel
    var namaPeserta string

    for {
        // Meminta input pengguna
        fmt.Print("Masukkan nama peserta: ")
        fmt.Scan(&namaPeserta)

        // Menghentikan untuk menginputkan nama
        if strings.ToLower(namaPeserta) == "selesai" {
            break // Hentikan input jika pengguna mengetik
'selesai'
        }

        // Reset nilai jumlah soal dan total skor
        jumlahSoal := 0
        totalSkor := 0

        // Input soal yang dikerjakan
        fmt.Print("Berapa banyak soal yang dikerjakan dari 8
soal: ")
        var dikerjakan int
        fmt.Scan(&dikerjakan)

        // Cek jumlah soal
        if dikerjakan < 1 || dikerjakan > 8 {
            fmt.Println("Jumlah soal harus antara 1 dan 8.")
            continue
        }

        // Input waktu pengerjaan
        fmt.Print("Masukkan waktu untuk ", dikerjakan, "
soal: ")
        hitungSkor(&jumlahSoal, &totalSkor, dikerjakan) //

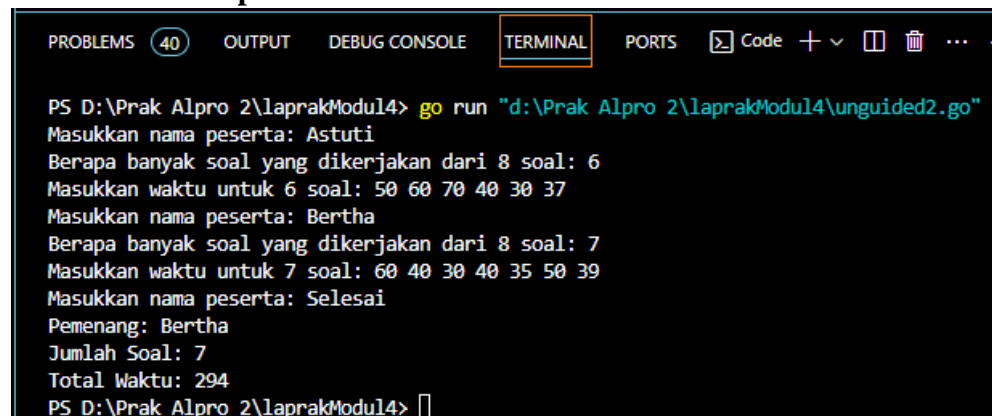
```

Mengoper jumlah soal yang dikerjakan

```
// Menentukan pemenang
if jumlahSoal > soalTerbanyak || (jumlahSoal ==
soalTerbanyak && totalSkor < totalWaktu) {
    pemenang = namaPeserta
    soalTerbanyak = jumlahSoal
    totalWaktu = totalSkor
}

// Menampilkan hasil akhir
if pemenang != "" {
    fmt.Printf("Pemenang: %s\n", pemenang)
    fmt.Printf("Jumlah Soal: %d\n", soalTerbanyak)
    fmt.Printf("Total Waktu: %d\n", totalWaktu)
} else {
    fmt.Println("Tidak ada peserta yang dimasukkan.")
}
}
```

Screenshot Output



```
PS D:\Prak Alpro 2\laprakModul4> go run "d:\Prak Alpro 2\laprakModul4\unguided2.go"
Masukkan nama peserta: Astuti
Berapa banyak soal yang dikerjakan dari 8 soal: 6
Masukkan waktu untuk 6 soal: 50 60 70 40 30 37
Masukkan nama peserta: Bertha
Berapa banyak soal yang dikerjakan dari 8 soal: 7
Masukkan waktu untuk 7 soal: 60 40 30 40 35 50 39
Masukkan nama peserta: Selesai
Pemenang: Bertha
Jumlah Soal: 7
Total Waktu: 294
PS D:\Prak Alpro 2\laprakModul4> 
```

Deskripsi Program

Program di atas dibuat untuk menentukan pemenang dari sebuah kompetisi mengerjakan soal. Setiap peserta diminta untuk memasukkan nama, jumlah soal yang dikerjakan, dan waktu yang dihabiskan pada setiap soal. Program akan menghitung total skor berdasarkan sedikitnya waktu pengerjaan soal dan banyak soal yang dikerjakan. Perhitungan skor dihitung dalam prosedur hitungskor. Jika ada dua peserta yang mengerjakan jumlah soal yang sama, pemenang akan ditentukan oleh total waktu pengerjaan yang lebih cepat. Setelah input berhenti dengan mengetik "selesai," program akan menampilkan nama pemenang, jumlah soal yang dikerjakan, dan total waktu pengerjaan.

***Note:** Pada terminal terdapat 40 problem dikarenakan package dan func main digunakan juga di file lain

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 cetak Deret 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.

Source code

```
package main

import "fmt"

// Prosedur untuk mencetak deret
func deretBilangan(n int) {
    fmt.Print("Deret bilangan: ")
    for n != 1 {
        // Cetak suku deret saat ini
        fmt.Printf("%d ", n)
        if n%2 == 0 {
            n = n / 2 // Jika suku genap
        } else {
            n = 3*n + 1 // Jika suku ganjil
        }
    }

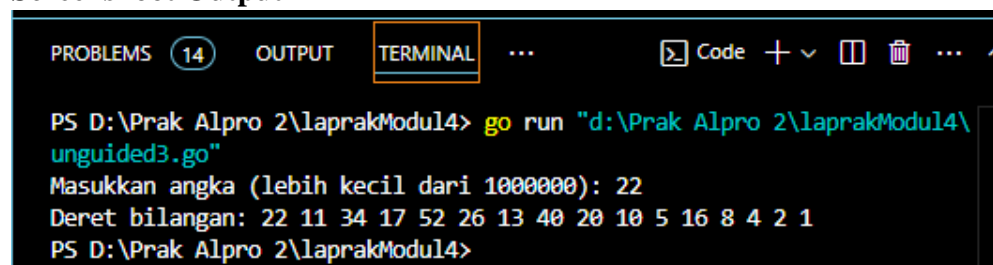
    // Cetak suku terakhir yang selalu bernilai 1
    fmt.Printf("1\n")
}

func main() {
    var n int
```

```
    fmt.Print("Masukkan angka (lebih kecil dari 1000000): ")
    fmt.Scan(&n)

    if n > 0 && n < 1000000 {
        deretBilangan(n) // Panggil prosedur deretBilangan
        dengan parameter n
    } else {
        fmt.Println("Input harus bilangan positif yang
kurang dari 1000000.")
    }
}
```

Screenshoot Output



```
PS D:\Prak Alpro 2\laprakModul4> go run "d:\Prak Alpro 2\laprakModul4\
unguided3.go"
Masukkan angka (lebih kecil dari 1000000): 22
Deret bilangan: 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS D:\Prak Alpro 2\laprakModul4>
```

Deskripsi Program

Program di atas dibuat untuk mencetak deret bilangan. Pengguna diminta untuk menginputkan bilangan bulat positif yang lebih kecil dari 1000000. Bilangan akan dicek jika genap, maka dibagi 2 sedangkan jika ganjil, maka dikalikan tiga dan ditambah satu. Dalam menghitung deret bilangan menggunakan prosedur deretBilangan. Setelah dihitung, hasil dicetak ke layar.

***Note:** Pada terminal terdapat 14 problem dikarenakan package dan func main digunakan juga di file lain

DAFTAR PUSTAKA

- [1] Purbasari, W., Iqbal, T., Inayah, I., Munawir, M., Sutjiningtyas, S., Hikmawati, E., ... & Basri, H. (2024). ALGORITMA PEMROGRAMAN.