

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL VI  
REKURSIF**



**Disusun Oleh :**

**Nia Novela Ariandini / 2311102057**

**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 Rekursif**

Rekursif adalah teknik pemrograman di mana suatu fungsi memanggil dirinya sendiri selama proses eksekusinya. Teknik ini berguna dalam situasi di mana masalah dapat dipecah menjadi sub-masalah yang serupa, seperti perhitungan faktorial, deret Fibonacci, dan pencarian dalam struktur data seperti pohon atau graf. Dalam Go, rekursi diimplementasikan dengan membuat fungsi memanggil dirinya sendiri sampai mencapai kondisi dasar, yang berfungsi sebagai titik akhir untuk menghentikan pemanggilan berulang.

### **B. Karakteristik Rekursif**

1. Memiliki kasus dasar (base case)
2. Memiliki kasus rekursif (recursive case)
3. Setiap pemanggilan mengarah ke base case
4. Bersifat stack-based execution

### **C. Komponen Dasar Rekursif**

- a) Base Case : Menentukan kapan fungsi harus berhenti memanggil dirinya sendiri. Tanpa kondisi dasar, fungsi akan berlanjut tanpa batas dan menghasilkan kesalahan stack overflow. Kondisi penghentian rekursif, mengembalikan nilai tanpa pemanggilan rekursif dan mencegah infinite recursion.
- b) Pemanggilan Rekursif : Memanggil fungsi itu sendiri dengan parameter yang dimodifikasi, sehingga mendekati kondisi dasar. Pemanggilan fungsi terhadap dirinya sendiri, memecah masalah menjadi bagian lebih kecil dan harus mengarah ke base case.

### **D. Jenis-jenis Rekursif**

- 1) Regular Recursion : Fungsi memanggil dirinya sendiri hingga mencapai kondisi dasar. Biasanya, pemanggilan ini tidak langsung menyelesaikan perhitungan, tetapi menyusun solusi hingga akhirnya semua panggilan selesai.
- 2) Tail Recursion : Pemanggilan rekursif adalah operasi terakhir yang dilakukan dalam fungsi. Tail recursion lebih efisien karena beberapa kompiler yang mampu mengoptimalkan eksekusi ini sehingga menghindari pertumbuhan stack yang berlebihan.

## **E. Kelebihan dan Kekurangan Rekursif**

### **1) Kelebihan :**

- Kode lebih bersih dan elegant
- Ideal untuk struktur data hierarkis
- Cocok untuk algoritma divide-and-conquer
- Mudah diimplementasi untuk masalah rekursif

### **2) Kekurangan :**

- Overhead memori tinggi
- Risiko stack overflow
- Performa bisa lebih lambat dari iterasi
- Konsumsi stack space

## II. GUIDED

### 1. GUIDED 1

**Studi Case :** Membuat baris bilangan dari n hingga 1

Base – case : bilangan == 1

#### Sourcecode

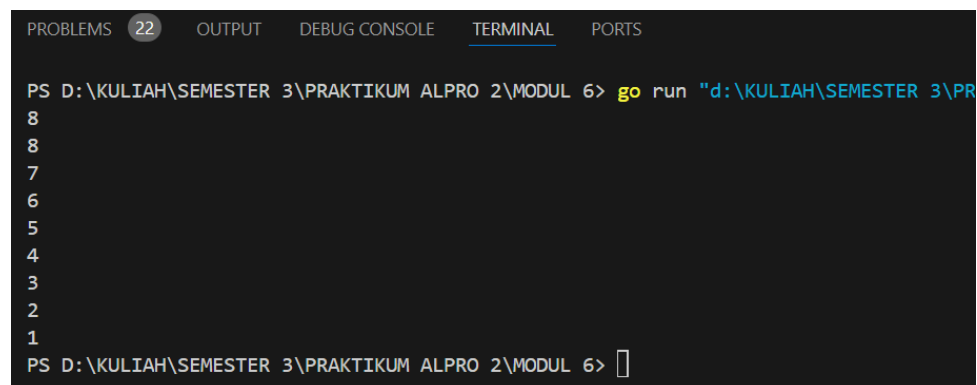
```
package main

import "fmt"

func baris(bilangan int) {
    if bilangan == 1 {
        fmt.Println(1)
    } else {
        fmt.Println(bilangan)
        baris(bilangan - 1)
    }
}

func main() {
    var n int
    fmt.Scan(&n)
    baris(n)
}
```

#### Screenshoot Output



```
PROBLEMS 22 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run "d:\KULIAH\SEMESTER 3\PR
8
8
7
6
5
4
3
2
1
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> █
```

#### Deskripsi Program

Program ini digunakan untuk deretan angka dari input pengguna hingga angka 1, dengan setiap angka dicetak di baris yang berbeda. Fungsi baris digunakan setelah pengguna memasukkan angka integer n. Fungsi baris menentukan apakah nilai bilangan yang diterima sama dengan 1. Jika nilainya benar, maka fungsi mencetak angka 1 dan berhenti, karena kondisi

ini adalah kondisi dasar (base case) dari rekursi. Namun, jika nilainya lebih besar dari 1, maka fungsi mencetak nilai tersebut dan memanggil dirinya sendiri dengan parameter bilangan - 1, yang mengulangi proses rekursif dengan nilai yang semakin berkurang hingga nilainya mencapai 1.

**Algoritma dan Cara Program Berfungsi :** Algoritma program ini dimulai dengan menerima input integer pengguna, yang disimpan dalam variabel `n`. Kemudian, program memanggil fungsi `baris(n)`. Kemudian, fungsi `baris` memeriksa apakah nilai bilangan sama dengan 1. Jika ya, maka fungsi mencetak angka 1 dan berhenti. Jika tidak, fungsi mencetak nilai yang ada, lalu memanggil dirinya sendiri dengan angka 1, dan proses ini berulang hingga kondisi dasar tercapai, yaitu ketika nilai bilangan sama dengan 1. Misalnya, jika pengguna memasukkan angka 8, program akan mencetak angka 8 dalam urutan menurun, masing-masing di baris baru.

## 2. GUIDED 2

**Studi Case :** Menghitung hasil penjumlahan 1 hingga `n`

Base – case : `n == 1`

### Sourcecode

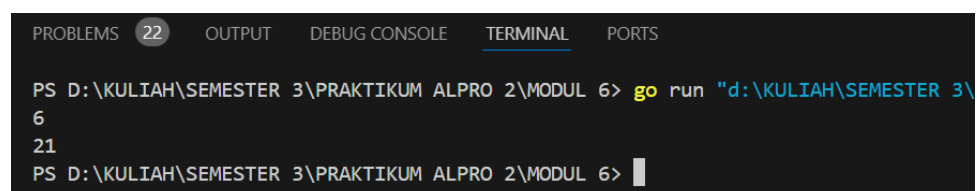
```
package main

import "fmt"

func penjumlahan(n int) int {
    if n == 1 {
        return 1
    } else {
        return n + penjumlahan(n-1)
    }
}

func main() {
    var n int
    fmt.Scan(&n)
    fmt.Print(penjumlahan(n))
}
```

### Screenshoot Output



```
PROBLEMS 22 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run "d:\KULIAH\SEMESTER 3\
6
21
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> |
```

## Deskripsi Program

Program ini untuk menghitung jumlah total dari angka  $n$  hingga 1. Pertama, pengguna memasukkan angka integer  $n$ , yang disimpan dalam variabel  $n$ . Kemudian, program memanggil fungsi penjumlahan dengan parameter  $n$ . Fungsi penjumlahan mengecek apakah nilai  $n$  sama dengan 1. Jika nilainya sama dengan 1, fungsi akan mengembalikan nilai 1 sebagai kondisi dasar (base case), yang mencegah rekursi tanpa batas. Jika nilainya tidak sama, fungsi akan menjumlahkan nilai  $n$  dengan hasil pemanggilan fungsi penjumlahan( $n-1$ ).

Algoritma dan Cara Program Berfungsi : Algoritma program ini dimulai dengan menerima angka dari pengguna dan menyimpannya dalam variabel  $n$ . Kemudian, fungsi penjumlahan dipanggil dengan parameter  $n$  untuk mengetahui apakah  $n$  bernilai 1. Jika itu benar, program akan menghitung nilai  $n$  ditambah hasil pemanggilan fungsi penjumlahan( $n-1$ ) jika tidak. Proses ini berlanjut secara rekursif, di mana  $n$  dikurangi 1 setiap kali hingga kondisi dasar adalah 1. Misalnya, jika pengguna memasukkan angka 6, fungsi akan menghitung  $6 + 5 + 4 + 3 + 2 + 1$ , menghasilkan output 21 yang dicetak dengan `fmt.Print(total(n))`.

### 3. GUIDED 3

**Studi Case :** Mencari dua pangkat  $n$  atau  $2n$

Base – case:  $n == 0$

#### Sourcecode

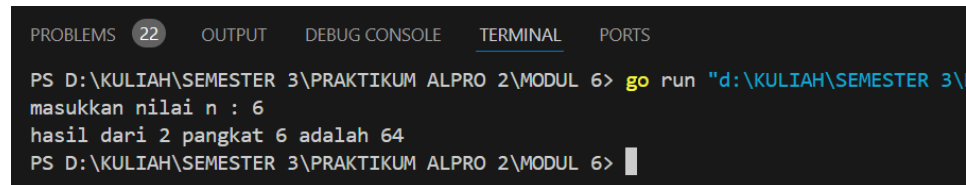
```
package main

import "fmt"

func pangkat(n int) int {
    if n == 0 {
        return 1
    } else {
        return 2 * pangkat(n-1)
    }
}

func main() {
    var n int
    fmt.Print("masukkan nilai n : ")
    fmt.Scan(&n)
    fmt.Println("hasil dari 2 pangkat", n, "adalah",
    pangkat(n))
}
```

## Screenshoot Output



```
PROBLEMS 22 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run "d:\KULIAH\SEMESTER 3\
masukkan nilai n : 6
hasil dari 2 pangkat 6 adalah 64
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> |
```

## Deskripsi Program

Program ini untuk menghitung nilai 2 pangkat n. Program meminta input angka n dari pengguna dan kemudian menggunakan fungsi pangkat untuk menghitung hasil 2 pangkat n. Fungsi pangkat menentukan apakah nilai n sama dengan 0. Jika n sama dengan 0, fungsi mengembalikan nilai 1, karena 2 pangkat 0 adalah 1—kondisi dasar yang menghentikan rekursi—dan jika n lebih besar dari 0, fungsi akan mengalikan 2 dengan hasil pemanggilan fungsi pangkat dengan parameter n-1.

Algoritma dan Cara Program Berfungsi : Algoritma program ini dimulai dengan meminta pengguna memasukkan angka integer n ke dalam variabel n. Setelah itu, program memanggil fungsi pangkat dengan argumen n. Jika n adalah 0, fungsi pangkat mengembalikan 1, tetapi jika tidak, fungsi mengalikan 2 dengan hasil dari pangkat(n-1). Nilai n dikurangi secara berulang hingga nilainya menjadi 0, dan kemudian hasil perkalian dikembalikan ke fungsi sebelumnya sampai fungsi pertama yang disebut. Misalnya, program akan menghitung  $2 \times 2 \times 2 \times 2 \times 2 \times 2$ , menghasilkan 64, dan mencetaknya sebagai "hasil dari 2 pangkat 6 adalah 64".

## 4. GUIDED 4

**Studi Case :** Mencari nilai factorial atau n!

Base – case:  $n == 0$  atau  $n == 1$

## Sourcecode

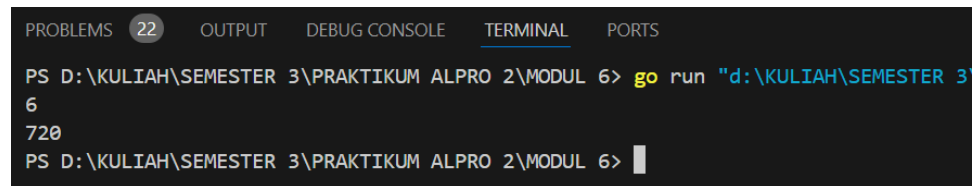
```
package main

import "fmt"

func faktorial(n int) int {
    if n == 0 || n == 1 {
        return 1
    } else {
        return n * faktorial(n-1)
    }
}
```

```
func main() {  
    var n int  
    fmt.Scan(&n)  
    fmt.Println(faktorial(n))  
}
```

## Screenshoot Output



```
PROBLEMS 22 OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run "d:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6\main.go"  
6  
720  
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> |
```

## Deskripsi Program

Program ini untuk menghitung faktorial dari angka  $n$  yang dimasukkan oleh pengguna. Program meminta pengguna untuk memasukkan nilai integer  $n$ , dan kemudian menggunakan fungsi faktorial untuk menghitung faktorial dari  $n$ . Fungsi ini memiliki dua kondisi. Jika  $n$  sama dengan 0 atau 1, fungsi akan mengembalikan nilai 1, karena faktorial dari 0 dan 1 adalah 1. Jika  $n$  lebih dari 1, fungsi akan mengalikan nilai  $n$  dengan hasil pemanggilan fungsi faktorial dengan parameter  $n-1$ , yang menghentikan proses rekursif dengan menghitung faktorial dari angka sebelumnya.

Algoritma dan Cara Program Berfungsi : Algoritma program ini dimulai dengan mengambil integer pengguna dan menyimpannya dalam variabel  $n$ . Kemudian, program memanggil fungsi faktorial( $n$ ). Fungsi faktorial kemudian menghitung  $n * \text{faktorial}(n-1)$  dan mengembalikan 1 sebagai kondisi dasar jika nilai  $n$  sama dengan 0 atau 1. Jika tidak, fungsi menghitung  $n * \text{faktorial}(n-1)$ , yang kemudian memanggil fungsi secara berulang dengan nilai  $n$  yang berkurang satu per satu hingga mencapai kondisi dasar. Misalnya, program akan menghitung  $6*5*4*3*2*1$ , yang menghasilkan 720. Nilai akhir ini kemudian dicetak oleh fungsi main.

## III. UNGUIDED

### 1. UNGUIDED 1

**Studi Case :** Deret fibonacci adalah sebuah deret dengan nilai suku ke-0 dan ke-1 adalah 0 dan 1, dan nilai suku ke- $n$  selanjutnya adalah hasil penjumlahan dua suku sebelumnya. Secara umum dapat diformulasikan  $S_n = S_{n-1} + S_{n-2}$ . Berikut ini adalah contoh nilai deret



fibonacci hingga suku ke-10. Buatlah program yang mengimplementasikan fungsi rekursif pada deret fibonacci tersebut.

$n$	0	1	2	3	4	5	6	7	8	9	10
$S_n$	0	1	1	2	3	5	8	13	21	34	55

### Sourcecode

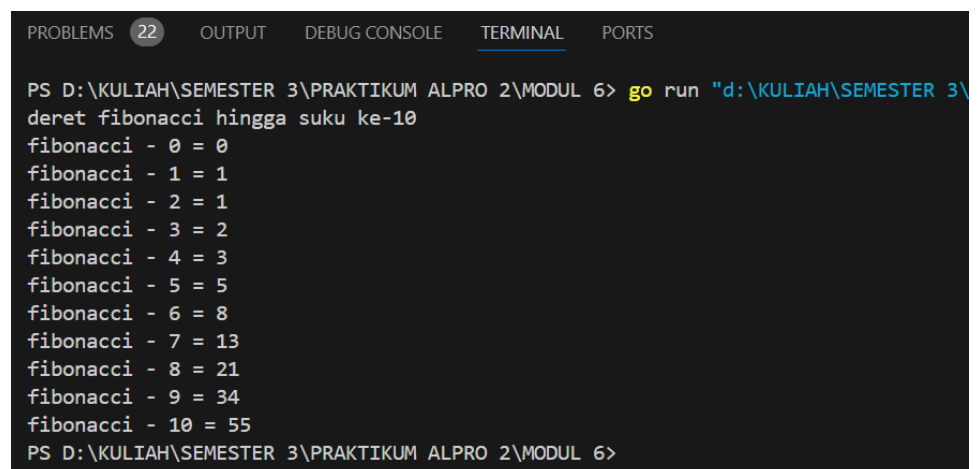
```
package main

import (
    "fmt"
)

func fibonacci(n int) int {
    if n == 0 {
        return 0
    } else if n == 1 {
        return 1
    }
    return fibonacci(n-1) + fibonacci(n-2)
}

func main() {
    fmt.Printf("deret fibonacci hingga suku ke-10\n")
    for i := 0; i <= 10; i++ {
        fmt.Printf("fibonacci - %d = %d\n", i,
            fibonacci(i))
    }
}
```

### Screenshoot Output



The screenshot shows a terminal window with the following content:

```
PROBLEMS 22 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run "d:\KULIAH\SEMESTER 3\
deret fibonacci hingga suku ke-10
fibonacci - 0 = 0
fibonacci - 1 = 1
fibonacci - 2 = 1
fibonacci - 3 = 2
fibonacci - 4 = 3
fibonacci - 5 = 5
fibonacci - 6 = 8
fibonacci - 7 = 13
fibonacci - 8 = 21
fibonacci - 9 = 34
fibonacci - 10 = 55
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6>
```

## Deskripsi Program

Program ini dapat menghasilkan deret Fibonacci hingga suku ke-10. Pola Fibonacci terdiri dari serangkaian angka yang setiap satunya adalah hasil penjumlahan dari dua angka sebelumnya, yang dimulai dengan nol dan satu. Fungsi Fibonacci menggunakan rekursi untuk mendefinisikan pola ini. Fungsi Fibonacci memanggil dirinya sendiri dengan parameter  $n-1$  dan  $n-2$  dan mengembalikan hasilnya untuk memberikan nilai dari suku ke- $n$  dalam deret Fibonacci. Ketika fungsi menerima parameter  $n$  yang bernilai 0, ia mengembalikan 0; jika  $n$  bernilai 1, ia mengembalikan 1.

Algoritma dan Cara Program Berfungsi : Algoritma program ini dimulai dengan mencetak judul "deret Fibonacci hingga suku ke-10" di layar. Kemudian, untuk menghitung setiap suku dalam deret Fibonacci, program memanggil fungsi `fibonacci(i)` setiap kali untuk menghitung nilai suku ke- $i$  dalam deret, dan kemudian mencetak hasilnya dalam format "fibonacci -  $i$  = nilai". Misalnya, fungsi `fibonacci(0)` mengembalikan 0 pada iterasi pertama (ketika  $i = 0$ ). Begitu pula, ketika  $i = 1$ , fungsi mengembalikan 1, dan ketika  $i = 2$ , fungsi memanggil `fibonacci(1) + fibonacci(0)`, yang menghasilkan 1. Proses ini berlanjut hingga suku ke-10, yang menghasilkan deret Fibonacci dari 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, dan 55.

## 2. UNGUIDED 2

**Studi Case :** Buatlah sebuah program yang digunakan untuk menampilkan pola bintang berikut ini dengan menggunakan fungsi rekursif.  $N$  adalah masukan dari user.

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	* ** *** **** *****
2	1	*
3	3	* ** ***

## Sourcecode

```
package main

import (
    "fmt"
```

```

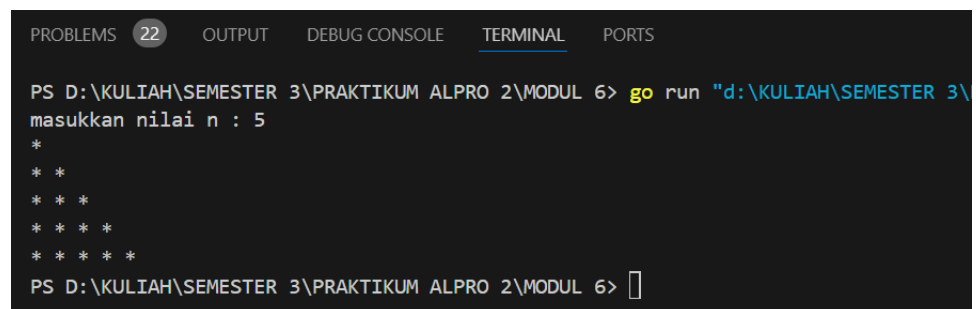
)

func tampilkanbintang(n int) {
    if n == 0 {
        return
    }
    tampilkanbintang(n - 1)
    for i := 0; i < n; i++ {
        fmt.Print("* ")
    }
    fmt.Println()
}

func main() {
    var n int
    fmt.Print("masukkan nilai n : ")
    fmt.Scan(&n)
    tampilkanbintang(n)
}

```

## Screenshoot Output



```

PROBLEMS 22 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run "d:\KULIAH\SEMESTER 3\
masukkan nilai n : 5
*
* *
* * *
* * * *
* * * * *
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> 

```

## Deskripsi Program

Fungsi rekursif yang mengatur tampilan pola ini adalah fungsi tampilkan bintang. Untuk memulai, fungsi menentukan apakah nilai  $n$  sama dengan 0. Jika hasilnya benar, fungsi langsung kembali tanpa melakukan apa pun. Ini adalah kondisi dasar untuk menghentikan rekursi. Untuk mencetak baris sebelumnya, fungsi memanggil dirinya sendiri dengan nilai  $n-1$  terlebih dahulu jika  $n$  lebih dari 0. Selanjutnya, fungsi mencetak jumlah bintang  $n$  pada baris saat ini dengan perulangan for dan menggunakan fmt untuk memindahkan ke baris berikutnya.

Algoritma dan Cara Program Berfungsi : Algoritma program ini dimulai dengan meminta pengguna input untuk menentukan berapa banyak baris bintang yang akan ditampilkan. Fungsi tampilkanbintang( $n$ ) dipanggil setelah nilai  $n$  dimasukkan. Fungsi ini secara rekursif menurunkan nilai  $n$  dan memanggil dirinya sendiri hingga mencapai kondisi dasar, yaitu ketika

n sama dengan 0. Setelah mencapai kondisi dasar, setiap panggilan rekursif yang tertunda mulai mencetak bintang sesuai jumlah baris yang terakhir. Misalnya, jika pengguna memasukkan nilai 3, fungsi akan memanggil `tampilkanbintang(2)`, `tampilkanbintang(1)`, dan `tampilkanbintang(0)`. Pada titik ini, pencetakan dimulai.

### 3. UNGUIDED 3

**Studi Case :** Buatlah program yang mengimplementasikan rekursif untuk menampilkan faktor bilangan dari suatu N, atau bilangan yang apa saja yang habis membagi N. Masukan terdiri dari sebuah bilangan bulat positif N. Keluaran terdiri dari barisan bilangan yang menjadi faktor dari N (terurut dari 1 hingga N ya).

Contoh masukan dan keluaran :

No	Masukan	Keluaran
1	5	1 5
2	12	1 2 3 4 6 12

#### Sourcecode

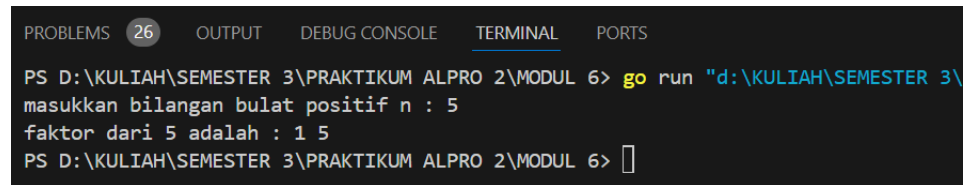
```
package main

import (
    "fmt"
)

func faktorbilangan(n, i int) {
    if i > n {
        return
    }
    if n%i == 0 {
        fmt.Print(i, " ")
    }
    faktorbilangan(n, i+1)
}

func main() {
    var n int
    fmt.Print("masukkan bilangan bulat positif n : ")
    fmt.Scan(&n)
    fmt.Print("faktor dari ", n, " adalah : ")
    faktorbilangan(n, 1)
    fmt.Println()
}
```

## Screenshoot Output



```
PROBLEMS 26 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run "d:\KULIAH\SEMESTER 3\
masukkan bilangan bulat positif n : 5
faktor dari 5 adalah : 1 5
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> 
```

## Deskripsi Program

Fungsi rekursif ini untuk mencari dan mencetak semua faktor dari bilangan bulat positif  $n$ . Pengguna diminta untuk memasukkan nilai  $n$ , dan kemudian program menggunakan fungsi faktorbilangan untuk menampilkan faktor-faktor dari  $n$ . Fungsi faktorbilangan memiliki dua parameter:  $n$ , bilangan yang ingin dicari oleh faktor, dan  $i$ , angka yang akan diuji untuk mengetahui apakah menjadi faktor dari  $n$ . Jika nilai  $i$  melebihi  $n$ , fungsi ini berhenti dan tidak memanggil dirinya lagi; ini adalah kondisi dasar rekursi. Fungsi melakukan pemeriksaan apakah  $n$  telah dibagi dengan  $i$  dengan kondisi  $n \% i == 0$ . Jika hasilnya benar, maka  $i$  dianggap sebagai faktor  $n$ . Setelah itu, fungsi faktorbilangan dipanggil kembali dengan nilai  $i + 1$ .

Algoritma dan Cara Program Berfungsi : Algoritma program dimulai dengan meminta input pengguna, yang kemudian disimpan dalam variabel  $n$ . Selanjutnya, program mencetak kalimat pengantar "faktor dari  $n$  adalah" sebelum menghubungi fungsi faktorbilangan( $n$ , 1). Fungsi ini memulai pemeriksaan dengan nilai  $i = 1$  dan terus menaikkan nilai  $i$  secara bertahap hingga mencapai  $n$ . Pada setiap iterasi, fungsi menentukan apakah  $i$  adalah faktor dari  $n$ , dan jika ya, nilai  $i$  dicetak. Jika tidak, proses dilanjutkan dengan nilai  $i$  berikutnya. Misalnya, jika pengguna memasukkan angka 5, program akan mencetak faktor 1 dan 5.

## 4. UNGUIDED 4

**Studi Case :** Buatlah program yang mengimplementasikan rekursif untuk menampilkan barisan bilangan tertentu. Masukan terdiri dari sebuah bilangan bulat positif  $N$ . Keluaran terdiri dari barisan bilangan dari  $N$  hingga 1 dan kembali ke  $N$ .

Contoh masukan dan keluaran :

No	Masukan	Keluaran
1	5	5 4 3 2 1 2 3 4 5
2	9	9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9

## Sourcecode

```
package main

import (
    "fmt"
)

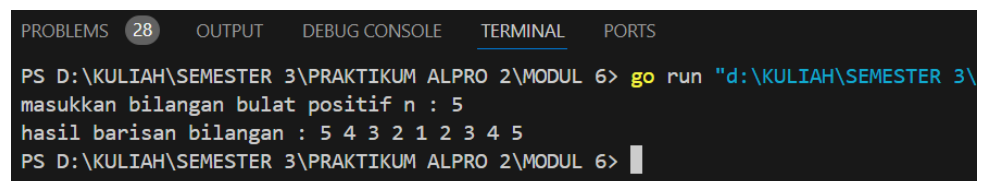
func turun(n int) {
    if n == 1 {
        fmt.Printf("%d ", n)
    } else {
        fmt.Printf("%d ", n)
        turun(n - 1)
    }
}

func naik(n, i int) {
    if i == n {
        fmt.Printf("%d ", i)
    } else {
        fmt.Printf("%d ", i)
        naik(n, i+1)
    }
}

func tampilkanbarisan(n int) {
    turun(n)
    naik(n, 2)
    fmt.Println()
}

func main() {
    var n int
    fmt.Print("masukkan bilangan bulat positif n : ")
    fmt.Scanln(&n)
    fmt.Print("hasil barisan bilangan : ")
    tampilkanbarisan(n)
}
```

## Screenshoot Output



The screenshot shows a terminal window with the following content:

```
PROBLEMS 28 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run "d:\KULIAH\SEMESTER 3\
masukkan bilangan bulat positif n : 5
hasil barisan bilangan : 5 4 3 2 1 2 3 4 5
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> |
```

## Deskripsi Program

Program di atas menunjukkan penggunaan fungsi rekursif untuk mencetak dua pola angka : pola menurun dari nilai n hingga 1, dan pola menaik dari nilai n hingga 2. Pengguna memasukkan nilai integer positif n, dan program menggunakan dua fungsi rekursif, turun dan naik, untuk menampilkan pola ini. Fungsi turun mencetak angka mulai dari n hingga 1 dengan menurunkan nilai n secara rekursif, sementara fungsi naik mencetak angka dari

Algoritma dan Cara Program Berfungsi : Algoritma aplikasi ini dimulai dengan meminta input pengguna dan menyimpannya dalam variabel n. Untuk mencetak barisan angka, Anda dapat menggunakan fungsi tampilanbarisan. Pertama, fungsi turun menentukan apakah n sama dengan 1, yang merupakan kondisi dasar untuk menghentikan rekursi. Fungsi akan memanggil dirinya sendiri dengan n-1 jika n lebih dari 1. Setelah selesai, fungsi naik disebut sebagai awal dengan i = 2 dan n sebagai batas. Fungsi naik mencetak nilai i dan memanggil dirinya sendiri dengan i+1 hingga mencapai nilai n. Misalnya, jika pengguna memasukkan n = 5, program akan mencetak "5 4 3 2 1 2 3 4 5", dengan angka pertama menciptakan pola turun dari 5 ke 1 dan kemudian berbalik menjadi pola naik dari 2 ke 5.

## 5. UNGUIDED 5

**Studi Case :** Buatlah program yang mengimplementasikan rekursif untuk menampilkan barisan bilangan ganjil. Masukan terdiri dari sebuah bilangan bulat positif N. Keluaran terdiri dari barisan bilangan ganjil dari 1 hingga N.

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	1 3 5
2	20	1 3 5 7 9 11 13 15 17 19

## Sourcecode

```
package main

import (
    "fmt"
)

func tampilkanganjil(n int, i int) {
    if i > n {
        return
    }
```

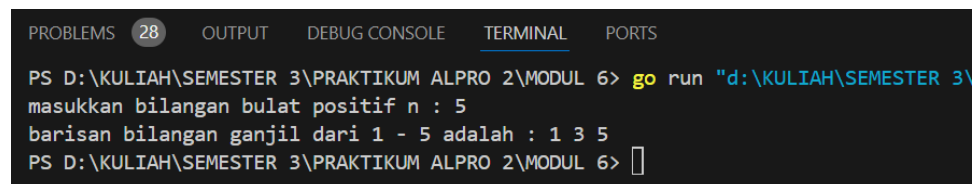
```

        if i%2 != 0 {
            fmt.Print(i, " ")
        }
        tampilkanganjil(n, i+1)
    }

    func main() {
        var n int
        fmt.Print("masukkan bilangan bulat positif n : ")
        fmt.Scan(&n)
        fmt.Print("barisan bilangan ganjil dari 1 - ", n, "
adalah : ")
        tampilkanganjil(n, 1)
        fmt.Println()
    }

```

## Screenshoot Output



```

PROBLEMS 28 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run "d:\KULIAH\SEMESTER 3\
masukkan bilangan bulat positif n : 5
barisan bilangan ganjil dari 1 - 5 adalah : 1 3 5
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6>

```

## Deskripsi Program

Fungsi rekursif untuk menampilkan deret bilangan ganjil dari 1 hingga n, di mana n adalah angka yang dimasukkan pengguna. Untuk mencetak angka ganjil secara rekursif, gunakan fungsi tampilkanganjil. Dua parameter diberikan kepada fungsi ini: n, batas atas angka yang akan diperiksa, dan i, angka saat ini yang diperiksa apakah ganjil atau tidak. Kondisi dasar untuk menghentikan rekursi adalah jika i lebih besar dari n. Jika itu benar, fungsi berhenti dan tidak melakukan apa pun lagi. Fungsi mengecek apakah i ganjil dengan kondisi  $i \% 2 \neq 0$ . Jika i dicetak, fungsi tampilkanganjil dipanggil kembali dengan parameter  $i + 1$  untuk memeriksa angka berikutnya. Ini dilakukan jika i tidak lebih besar dari n.

Algoritma dan Cara Program Berfungsi : Di awal algoritma program, pengguna diminta untuk memasukkan nilai integer n, yang merupakan batas atas deret bilangan ganjil yang akan ditampilkan. Selanjutnya, program memanggil fungsi tampilkanganjil(n, 1) untuk memulai pencarian angka ganjil dari 1 dan mencetak teks pengantar "barisan bilangan ganjil dari 1 hingga n adalah". Fungsi akan mencetak setiap angka ganjil dari 1 hingga n sebelum berhenti. Misalnya, jika pengguna memasukkan angka 5, program akan mencetak "1 3 5" sebagai hasil akhir. Hal ini disebabkan oleh fakta bahwa angka-angka ini adalah bilangan ganjil yang berada di antara 1 – 5.



## 6. UNGUIDED 6

**Studi Case :** Buatlah program yang mengimplementasikan rekursif untuk mencari hasil pangkat dari dua buah bilangan. Masukan terdiri dari bilangan bulat x dan y. Keluaran terdiri dari hasil x dipangkatkan y. Catatan : diperbolehkan menggunakan asterisk "\*", tapi dilarang menggunakan import "math".

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	2 2	4
2	5 3	125

### Sourcecode

```
package main

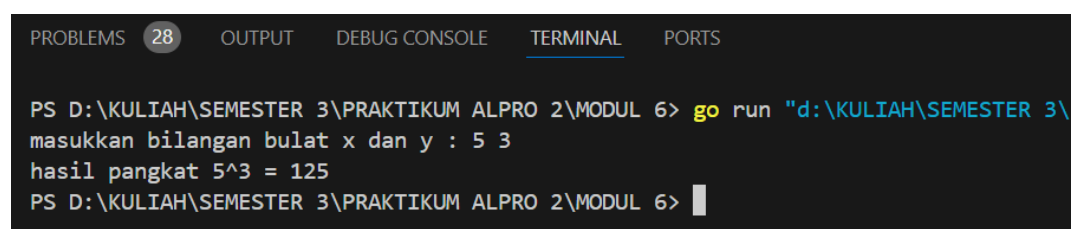
import (
    "fmt"
)

func pangkat(x int, y int) int {
    if y == 0 {
        return 1
    }
    return x * pangkat(x, y-1)
}

func main() {
    var x, y int
    fmt.Print("masukkan bilangan bulat x dan y : ")
    fmt.Scan(&x, &y)

    hasil := pangkat(x, y)
    fmt.Printf("hasil pangkat %d^%d = %d\n", x, y, hasil)
}
```

### Screenshoot Output



```
PROBLEMS 28 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run "d:\KULIAH\SEMESTER 3\
masukkan bilangan bulat x dan y : 5 3
hasil pangkat 5^3 = 125
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> |
```

## Deskripsi Program

Dalam program di atas, fungsi rekursif digunakan untuk hasil pangkat dari bilangan  $x$  dengan pangkat  $y$ . Program meminta pengguna memasukkan dua bilangan bulat,  $x$  sebagai basis, dan  $y$  sebagai eksponen. Kemudian, program memanggil fungsi pangkat untuk menghitung nilai  $x$  yang berada di pangkat  $y$ . Fungsi pangkat memiliki kondisi dasar yang menentukan apakah nilai  $y$  sama dengan 0. Jika nilai  $y$  sama dengan 0, fungsi akan mengembalikan 1 karena setiap bilangan yang dipangkatkan 0 adalah 1. Jika nilai  $y$  tidak sama dengan 0, fungsi akan mengembalikan hasil perkalian antara  $x$  dan hasil dari pangkat( $x$ ,  $y-1$ ), yang mengalikan  $x$  hingga  $y$  mencapai 0 secara rekursif.

Algoritma dan Cara Program Berfungsi : Algoritma program ini dimulai dengan mengumpulkan nilai  $x$  dan  $y$ , yang merupakan basis dan eksponen, dari pengguna. Fungsi pangkat disebutkan dengan  $x$  dan  $y$  sebagai argumen setelah nilai dimasukkan. Setiap kali nilai  $y$  dikurangi satu per satu, fungsi ini akan terus memanggil dirinya sendiri. Setelah nilai  $y$  mencapai 0, kondisi dasar terpenuhi dan proses rekursif berhenti. Fungsi utama menerima nilai dari perkalian akhir dan mencetak hasilnya. Misalnya, program akan menghitung  $5*5*5$  jika pengguna memasukkan  $x = 5$  dan  $y = 3$ , menghasilkan 125 dan akan mencetak hasil pangkat  $5^3$  adalah 125.

## IV. KESIMPULAN

Hasil dari belajar rekursif adalah bahwa membagi masalah yang kompleks menjadi bagian-bagian yang lebih kecil adalah teknik yang efektif untuk menyelesaikan masalah tersebut. Menghitung faktorial, menemukan deret Fibonacci, atau mencetak pola angka adalah beberapa tugas yang dapat kita selesaikan dalam game Go dengan menggunakan fungsi yang memanggil dirinya sendiri. Contoh yang telah dipelajari menunjukkan bahwa rekursi membuat kode lebih sederhana dan lebih mudah dibaca, serta membantu menyelesaikan masalah terstruktur secara alami. Namun, penting untuk memperhatikan bahwa penggunaan rekursi dapat memengaruhi kinerja program, terutama jika kedalaman rekursi terlalu besar, yang dapat menyebabkan penggunaan memori yang tinggi dan kesalahan stack overflow.

## V. REFERENSI

- [1] Modul VI Praktikum Algoritma dan Pemrograman 2
- [2] <https://golangdocs.com/recursion-in-golang>