

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL V**

**REKURSIF**



**Disusun Oleh :**

**Nama lengkap : Martin.C.Simbolon (2311102269)**

**Kelas: IF-11-G**

**Dosen Pengampu :**

**-----**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## I. DASAR TEORI

Rekursif adalah teknik pemrograman di mana suatu fungsi memanggil dirinya sendiri untuk menyelesaikan suatu masalah dengan cara memecahnya menjadi sub-masalah yang lebih kecil namun memiliki pola yang serupa. Dalam struktur rekursif, terdapat dua elemen penting, yaitu *base case* dan *recursive case*. *Base case* adalah kondisi dasar yang menghentikan proses rekursif ketika masalah sudah cukup sederhana untuk diselesaikan langsung.

Tanpa *base case*, fungsi akan terus berulang tanpa henti, menyebabkan *infinite loop* atau *stack overflow*. Penggunaan rekursif efektif untuk masalah yang memiliki pola berulang atau struktur berlapis, seperti perhitungan faktorial, deret Fibonacci, atau traversing struktur data seperti pohon biner. Rekursif memungkinkan kode menjadi lebih ringkas dan elegan dalam menyelesaikan masalah-masalah tersebut. Namun, rekursif juga memiliki kekurangan, terutama dalam efisiensi memori,

karena setiap pemanggilan fungsi menambah beban pada *stack memory*. Oleh karena itu, penggunaan rekursif perlu dipertimbangkan secara cermat, terutama pada masalah besar yang membutuhkan banyak iterasi.

## GUIDED

### 1. Soal Studi Case: cetak mundur

#### Sourcecode

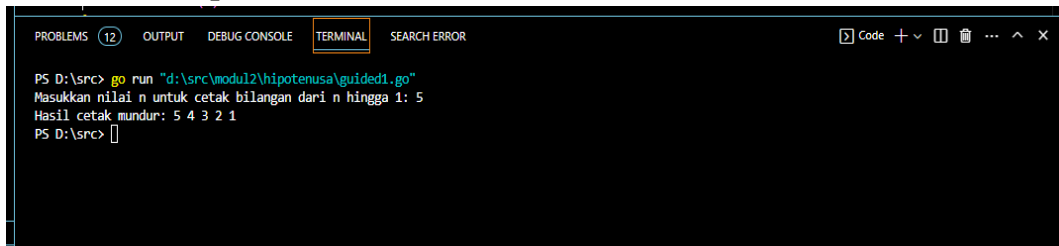
```
package main

import "fmt"

func cetakmundur(n int) {
    if n == 1 {
        fmt.Println(n)
        return
    }
    fmt.Print(n, " ")
    cetakmundur(n - 1)
}

func main() {
    var n int
    fmt.Print("Masukkan nilai n untuk cetak bilangan dari n hingga 1: ")
    fmt.Scan(&n)
    fmt.Print("Hasil cetak mundur: ")
    cetakmundur(n)
}
```

#### Screenshot output



```
PROBLEMS (12) OUTPUT DEBUG CONSOLE TERMINAL SEARCH ERROR
PS D:\src> go run "d:\src\modul2\hipotenusa\guided1.go"
Masukkan nilai n untuk cetak bilangan dari n hingga 1: 5
Hasil cetak mundur: 5 4 3 2 1
PS D:\src> 
```

#### Deskripsi Program

Kode ini menjalankan untuk program cetak mundur

## 2. Menghitung Rekursif

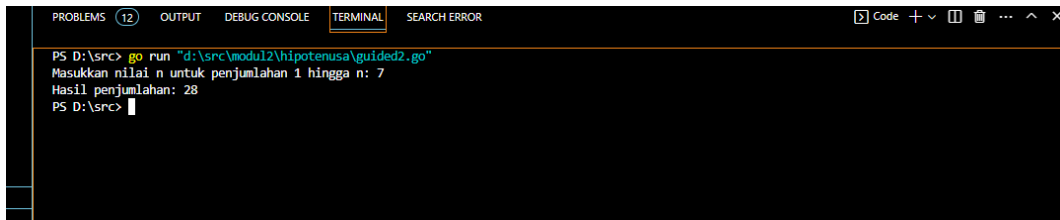
```
package main

import "fmt"

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

func main() {
    var n int
    fmt.Print("Masukkan nilai n untuk penjumlahan 1 hingga n: ")
    fmt.Scanln(&n)
    fmt.Println("Hasil penjumlahan:", jumlahrekursif(n))
}
```

Screenshoot output:



```
PS D:\src> go run "d:\src\modul2\hipotenusa\guided2.go"
Masukkan nilai n untuk penjumlahan 1 hingga n: 7
Hasil penjumlahan: 28
PS D:\src>
```

### Deskripsi program

**Program ini menghitung penjumlahan rekursif dari angka 1 hingga n menggunakan fungsi rekursif jumlahrekursif. Pengguna diminta memasukkan nilai n, kemudian program akan menghitung total penjumlahan mulai dari n hingga 1 secara bertahap dengan memanggil fungsi jumlahrekursif berulang kali. Fungsi ini memiliki kondisi dasar if n == 1, yang mengembalikan nilai 1 ketika n sudah mencapai 1, menghentikan rekursi. Hasil akhirnya ditampilkan sebagai penjumlahan semua angka dari 1 hingga n.**

### 3. Pangkat dua

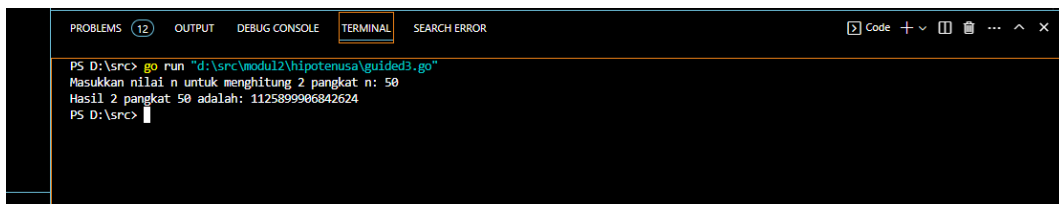
```
package main

import "fmt"

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

func main() {
    var n int
    fmt.Print("Masukkan nilai n untuk menghitung 2 pangkat n: ")
    fmt.Scanln(&n)
    fmt.Println("Hasil 2 pangkat", n, "adalah:", pangkatdua(n))
}
```

#### Screenshoot output



```
PS D:\src> go run "d:\src\modul2\hipotenusa\guided3.go"
Masukkan nilai n untuk menghitung 2 pangkat n: 50
Hasil 2 pangkat 50 adalah: 1125899906842624
PS D:\src>
```

#### Deskripsi program

Program ini menghitung hasil perpangkatan 2 dengan pangkat n menggunakan rekursi. Pengguna diminta memasukkan nilai n, dan fungsi pangkatdua akan menghitung  $2^n$  dengan memanggil dirinya sendiri secara berulang. Kondisi dasar if  $n == 0$  mengembalikan nilai 1, karena  $2^0$  adalah 1, yang menghentikan rekursi. Fungsi kemudian mengalikan hasil rekursi dengan 2 secara berulang hingga mencapai pangkat yang diinginkan, lalu hasilnya ditampilkan.

#### 4.Faktorial

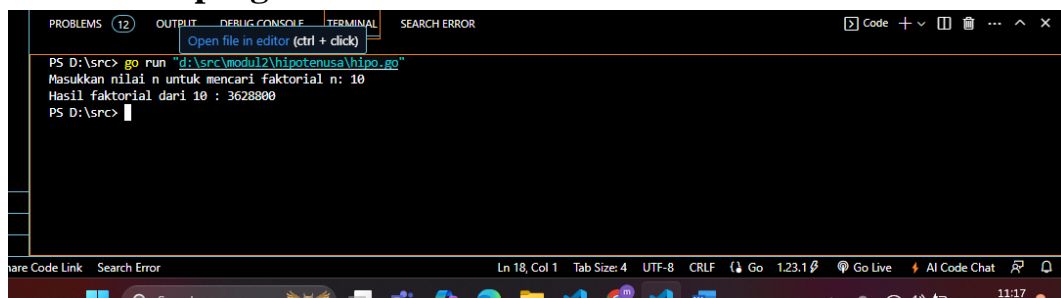
```
package main

import "fmt"

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

func main() {
    var n int
    fmt.Print("Masukkan nilai n untuk mencari faktorial n: ")
    fmt.Scanln(&n)
    fmt.Println("Hasil faktorial dari", n, ":", faktorial(n))
}
```

#### Screenshoot program



#### Deskripsi program

Program ini menghitung nilai faktorial dari angka n menggunakan rekursi. Pengguna memasukkan nilai n, dan fungsi faktorial akan menghitung nilai faktorial tersebut dengan memanggil dirinya sendiri secara berulang hingga mencapai kondisi dasar  $n == 0$  atau  $n == 1$ , yang mengembalikan nilai 1 untuk menghentikan rekursi. Setiap pemanggilan fungsi akan mengalikan nilai n dengan hasil faktorial dari  $n - 1$ ,

## UNGUIDED

### Soal Studi Case

fibonachi

### Sourcecode

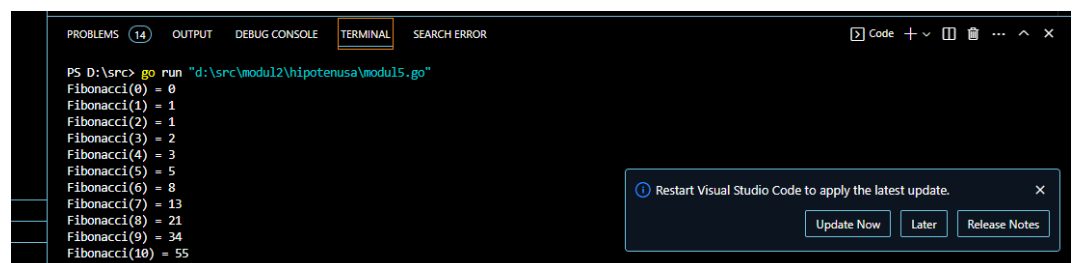
```
package main

import "fmt"

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

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

### Screenshoot Output

The screenshot shows the Visual Studio Code interface with the 'TERMINAL' tab active. The terminal displays the command 'PS D:\src> go run "d:\src\modul2\hipotenusa\modul5.go"' and its output, which lists Fibonacci numbers from 0 to 10. A notification box in the bottom right corner prompts the user to 'Restart Visual Studio Code to apply the latest update.' with buttons for 'Update Now', 'Later', and 'Release Notes'.

```
PS D:\src> go run "d:\src\modul2\hipotenusa\modul5.go"
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
```

### Deskripsi Program

Kode ini mendefinisikan fungsi **fibonacci(n)** yang menerima integer **n** sebagai input dan mengembalikan nilai Fibonacci ke-**n**. Fungsi ini menggunakan rekursi untuk menghitung nilai Fibonacci, dengan kasus dasar **n <= 1**, di mana fungsi langsung mengembalikan **n**. Jika **n** lebih

besar dari 1, fungsi memanggil dirinya sendiri dua kali dengan **n-1** dan **n-2** dan mengembalikan penjumlahan dari hasil rekursi. Fungsi **main()** kemudian mencetak nilai Fibonacci dari 0 hingga 10.

## 2.pola

```
package main

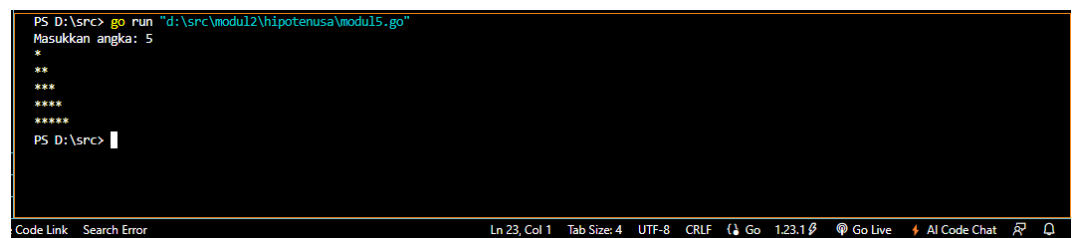
import "fmt"

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

func main() {
    var n int
    fmt.Print("Masukkan angka: ")
    fmt.Scanln(&n)

    printStars(n)
}
```

## Screenshot output



```
PS D:\src> go run "d:\src\modul2\hipotenusa\modul5.go"
Masukkan angka: 5
*
**
***
****
*****
PS D:\src>
```

## Deskripsi program

Program ini menggunakan fungsi rekursif **printStars** untuk menampilkan pola bintang. **printStars** menerima satu argumen **n** yang menentukan jumlah bintang



pada baris terakhir. Kasus dasarnya, jika **n** adalah 0, fungsi berhenti. Langkah rekursifnya memanggil dirinya sendiri dengan **n-1** sebagai argumen, mencetak bintang-bintang pada baris sebelumnya. Setelah panggilan rekursif, fungsi mencetak **n** bintang pada satu baris. Fungsi **main** meminta pengguna memasukkan angka **n** dan memanggil **printStars** dengan **n** sebagai argumen untuk memulai pencetakan bintang