

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL V**

**REKURSIF**



**Disusun Oleh :**

**Egi Umar Ferdhika / 2311102277**

**11-IF-06**

**Dosen Pengampu :**

**Abednego Dwi Septiadi, S.Kom., M.Kom**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## **I. DASAR TEORI**

Rekursif (recursive) dalam pemrograman Go adalah teknik di mana sebuah fungsi memanggil dirinya sendiri untuk menyelesaikan suatu masalah. Melalui rekursi, masalah besar dapat dipecah menjadi bagian-bagian yang lebih kecil hingga mencapai kondisi dasar atau "base case" yang menghentikan pemanggilan diri. Rekursi memungkinkan penulisan kode yang lebih sederhana untuk masalah-masalah yang bersifat hierarkis atau berulang, seperti menghitung faktorial, deret Fibonacci, atau menelusuri struktur data seperti pohon dan grafik. Pada setiap pemanggilan, Go akan menyimpan setiap langkah pemanggilan di memori (stack), sehingga rekursi yang dalam tanpa kondisi dasar yang tepat dapat menyebabkan kesalahan stack overflow. Namun, pada masalah tertentu, rekursi sangat bermanfaat karena membuat kode lebih intuitif dan mudah dibaca, terutama untuk algoritma yang mengikuti pola pemecahan masalah bertahap, seperti pembagian dan penaklukan (divide and conquer).

Meskipun Go tidak memiliki optimasi rekursi tail (seperti pada beberapa bahasa lain), dengan memperhatikan kedalaman pemanggilan dan kondisi dasar yang jelas, rekursi tetap menjadi alat yang kuat dalam Go untuk menyelesaikan masalah-masalah tertentu secara efisien dan elegan.

## II. GUIDED

1. Berisi source code dan output dari kegiatan praktikum yang telah dilaksanakan.

Source Code diberi penjelasan maka akan menjadi nilai ++

### GUIDED 1

#### Sourcecode

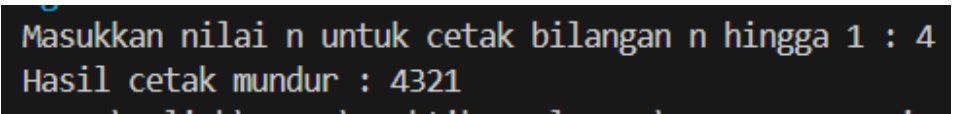
```
package main

import "fmt"

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

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

#### Screenshoot Output



```
Masukkan nilai n untuk cetak bilangan n hingga 1 : 4
Hasil cetak mundur : 4321
```

#### Deskripsi Program

Program di atas mencetak bilangan dari n hingga 1 secara menurun menggunakan fungsi rekursif bernama cetak. Fungsi ini akan mencetak nilai n, kemudian memanggil dirinya sendiri dengan n-1 hingga mencapai 1, dimana proses akan berhenti. Pada fungsi main, pengguna diminta memasukkan nilai n, dan setelah itu, program akan menampilkan urutan mundur dari n menggunakan fungsi cetak.

### GUIDED 2

#### Sourcecode

```

package main

import "fmt"

func jmlRekursif(n int) int {
    if n == 1 {
        return 1
    }

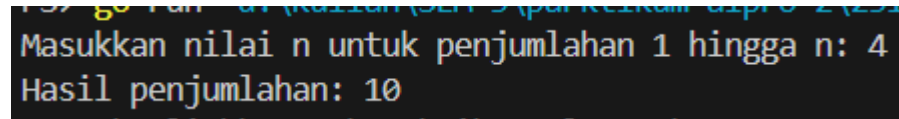
    return n + jmlRekursif(n-1)
}

func main() {
    var n int

    fmt.Print("Masukkan nilai n untuk penjumlahan 1
    hingga n: ")
    fmt.Scanln(&n)
    fmt.Println("Hasil penjumlahan:",
    jmlRekursif(n))
}

```

### Screenshoot Output



```

Masukkan nilai n untuk penjumlahan 1 hingga n: 4
Hasil penjumlahan: 10

```

### Deskripsi Program

Program di atas adalah program yang menghitung penjumlahan dari 1 hingga n menggunakan fungsi rekursif jmlRekursif. Fungsi ini menerima parameter n dan bekerja dengan menambahkan n ke hasil pemanggilan dirinya sendiri dengan n-1, hingga mencapai kondisi dasar, yaitu  $n = 1$ , yang akan mengembalikan nilai 1 dan mengakhiri proses rekursi. Pada fungsi main, pengguna diminta memasukkan nilai n. Setelah menerima input, program akan memanggil jmlRekursif dengan nilai n dan mencetak hasil penjumlahan dari 1 hingga n.

## GUIDED 3

### Sourcecode

```

package main

import "fmt"

```

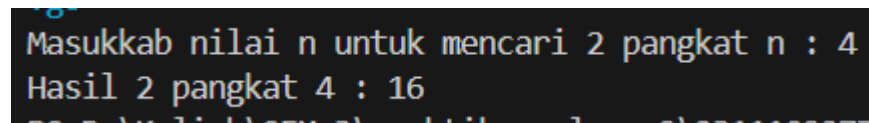
```

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

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

```

### Screenshoot Output



```

Masukkan nilai n untuk mencari 2 pangkat n : 4
Hasil 2 pangkat 4 : 16

```

### Deskripsi

### Program

Program di atas adalah program yang menghitung nilai  $2^n$  menggunakan fungsi rekursif pangkat. Fungsi pangkat menerima parameter  $n$  dan akan mengalikan 2 dengan hasil panggilan dirinya sendiri dengan  $n-1$ . Kondisi dasar dalam fungsi ini adalah ketika  $n == 0$ , yang akan mengembalikan nilai 1, karena  $2^0=1$ . Di dalam fungsi main, program meminta pengguna memasukkan nilai  $n$ . Setelah menerima input, program akan memanggil fungsi pangkat dengan nilai  $n$  dan menampilkan hasil perhitungan  $2^n$ . Misalnya, jika pengguna memasukkan 3, maka program akan mencetak 8, karena  $2^3 = 8$ .

## GUIDED 4

### Sourcecode

```

package main

import "fmt"

func main() {
    var n int
    fmt.Print("Masukkan bilangan non-negatif: ")
    fmt.Scan(&n)

    if n < 0 {

```

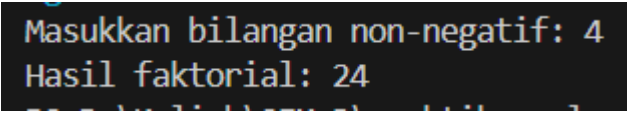
```

        fmt.Println("Faktorial tidak didefinisikan untuk
bilangan negatif")
    } else {
        fmt.Println("Hasil faktorial:", faktorial(n))
    }
}

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

```

### Screenshoot Output



```

Masukkan bilangan non-negatif: 4
Hasil faktorial: 24

```

### Deskripsi Program

Program di atas adalah program yang menghitung faktorial dari suatu bilangan non-negatif  $n$  menggunakan fungsi rekursif faktorial. Di dalam fungsi main, program meminta pengguna memasukkan sebuah bilangan. Jika bilangan yang dimasukkan negatif, program akan menampilkan pesan bahwa faktorial tidak didefinisikan untuk bilangan negatif. Jika bilangan tersebut non-negatif, program akan memanggil fungsi faktorial dan menampilkan hasilnya. Fungsi faktorial bekerja dengan memeriksa kondisi dasar, yaitu ketika  $n$  bernilai 0 atau 1, yang akan mengembalikan nilai 1 (karena  $0!$  dan  $1!$  sama dengan 1). Jika  $n$  lebih dari 1, fungsi ini mengalikan  $n$  dengan hasil pemanggilan  $\text{faktorial}(n-1)$ , yang secara rekursif akan menghitung faktorial dari  $n$ .

## III. UNGUIDED

1. Berisi source code dan output dari kegiatan praktikum yang telah dilaksanakan.

Source Code diberi penjelasan maka akan menjadi nilai ++

### UNGUIDED 1

#### Sourcecode

```
package main
```

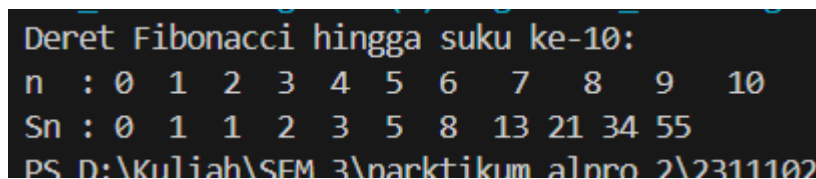
```

import "fmt"
// NAMA : Egi Umar Ferdhika
// NIM : 2311102277
func fibonacci(n int) int {
    if n <= 1 {
        return n
    }
    return fibonacci(n-1) + fibonacci(n-2)
}

func main() {
    fmt.Println("Deret Fibonacci hingga suku ke-10:")
    fmt.Println("n  :")
    0 1 2 3 4 5 6 7 8 9 10")
    fmt.Print("Sn : ")
    for i := 0; i <= 10; i++ {
        fmt.Printf("%-3d", fibonacci(i))
    }
    fmt.Println()
}

```

### Screenshoot Output



```

Deret Fibonacci hingga suku ke-10:
n  : 0 1 2 3 4 5 6 7 8 9 10
Sn : 0 1 1 2 3 5 8 13 21 34 55
PS D:\Kuliah\SEM 3\praktikum_alpro 2\2311102

```

### Deskripsi Program

Program di atas adalah program yang menampilkan deret Fibonacci hingga suku ke-10. Program ini menggunakan fungsi rekursif fibonacci untuk menghitung nilai dari setiap suku dalam deret Fibonacci. Fungsi fibonacci menerima parameter *n* dan bekerja dengan memeriksa kondisi dasar: jika *n* bernilai 0 atau 1, fungsi akan mengembalikan nilai *n* tersebut langsung (karena 0 dan 1 adalah suku pertama dan kedua dalam deret Fibonacci). Jika *n* lebih dari 1, fungsi ini mengembalikan hasil penjumlahan dari fibonacci(*n*-1) + fibonacci(*n*-2), yang secara rekursif menghitung suku ke-*n* berdasarkan dua suku sebelumnya. Di dalam fungsi main, program mencetak judul dan label untuk memudahkan pembacaan hasil. Kemudian, menggunakan perulangan for, program memanggil fungsi fibonacci dari suku ke-0 hingga ke-10, dan mencetak hasilnya dalam satu baris.

## UNGUIDED 2

### Sourcecode

```
package main

import "fmt"
// NAMA : Egi Umar Ferdhika
// NIM : 2311102277
func cetakBintang(n int, current int) {
    if current > n {
        return
    }

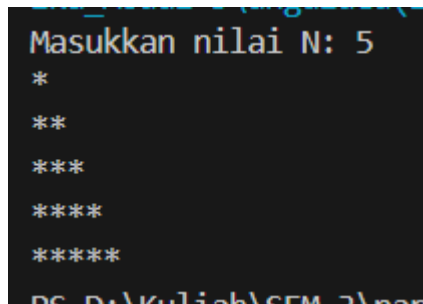
    for i := 0; i < current; i++ {
        fmt.Print("*")
    }
    fmt.Println()

    cetakBintang(n, current+1)
}

func main() {
    var n int
    fmt.Print("Masukkan nilai N: ")
    fmt.Scan(&n)

    cetakBintang(n, 1)
}
```

### Screenshoot Output



### Deskripsi Program

Kode di atas adalah program Go yang mencetak pola segitiga bintang berdasarkan nilai  $n$  yang diberikan pengguna. Fungsi `cetakBintang` menerima dua parameter:  $n$ , yaitu jumlah baris yang diinginkan, dan  $current$ , yaitu baris yang sedang diproses. Pada setiap pemanggilan fungsi, jika  $current$  melebihi  $n$ , fungsi akan berhenti (dengan perintah `return`). Jika tidak, program akan mencetak sejumlah bintang yang sesuai dengan nilai  $current$ , lalu membuat baris baru. Setelah itu, fungsi memanggil dirinya sendiri dengan  $current + 1$ , sehingga baris berikutnya akan



mencetak satu bintang lebih banyak. Pada fungsi main, program meminta pengguna memasukkan nilai n sebagai jumlah baris yang akan dicetak. Kemudian, program memanggil cetakBintang dengan parameter n dan current yang dimulai dari 1 untuk mencetak pola bintang dari satu baris hingga n baris.

### UNGUIDED 3

#### Sourcecode

```
package main

import "fmt"
// NAMA : Egi Umar Ferdhika
// NIM : 2311102277
func cetakFaktor(n int, current int) {
    if current > n {
        return
    }

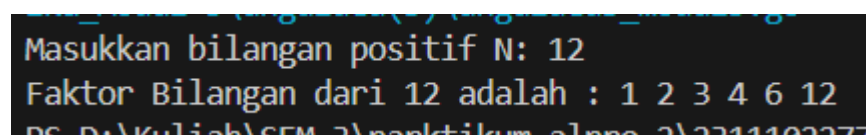
    if n%current == 0 {
        fmt.Print(current)
        if current != n {
            fmt.Print(" ")
        }
    }
    cetakFaktor(n, current+1)
}

func main() {
    var n int
    fmt.Print("Masukkan bilangan positif N: ")
    fmt.Scan(&n)

    if n <= 0 {
        fmt.Println("Mohon masukkan bilangan positif!")
        return
    }

    fmt.Print("Faktor Bilangan dari ", n, " adalah : ")
    cetakFaktor(n, 1)
    fmt.Println()
}
```

#### Screenshoot Output



```
Masukkan bilangan positif N: 12
Faktor Bilangan dari 12 adalah : 1 2 3 4 6 12
D:\Kuliah\SEM 2\penelitian_alpro_2\2311102277
```

## Deskripsi Program

Program di atas adalah program yang menampilkan semua faktor dari sebuah bilangan positif  $n$  yang dimasukkan oleh pengguna. Fungsi `cetakFaktor` menerima dua parameter:  $n$ , yaitu bilangan yang ingin dicari faktornya, dan `current`, yaitu angka yang sedang diperiksa apakah merupakan faktor dari  $n$ . Pada setiap pemanggilan fungsi, jika `current` lebih besar dari  $n$ , fungsi berhenti (dengan `return`). Jika tidak, fungsi memeriksa apakah `current` adalah faktor dari  $n$  dengan kondisi `% current == 0`. Jika kondisi ini benar, `current` dicetak sebagai faktor dari  $n$ . Fungsi juga mencetak spasi setelah setiap faktor, kecuali faktor terakhir. Di dalam fungsi `main`, program meminta pengguna untuk memasukkan bilangan positif  $n$ . Jika  $n$  bernilai negatif atau nol, program akan menampilkan pesan kesalahan dan berhenti. Jika  $n$  adalah bilangan positif, program memanggil `cetakFaktor` dengan nilai awal `current = 1` untuk mulai mencari dan mencetak faktor dari  $n$ .

## UNGUIDED 4

### Sourcecode

```
package main

import "fmt"
// NAMA : Egi Umar Ferdhika
// NIM : 2311102277
func cetakUrutan(n int, descending bool) {
    if n < 1 {
        return
    }

    fmt.Printf("%d ", n)

    if descending && n == 1 {
        cetakUrutan(2, false)
        return
    }

    if descending {
        cetakUrutan(n-1, true)
    } else if n < max {
        cetakUrutan(n+1, false)
    }
}

var max int

func main() {
    var n int
    fmt.Print("Masukkan bilangan positif N: ")
}
```

```

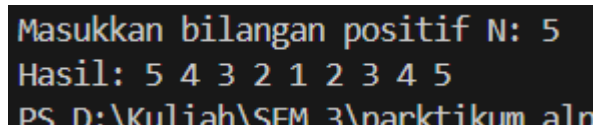
        fmt.Scan(&n)

        if n <= 0 {
            fmt.Println("error")
            return
        }

        max = n
        fmt.Print("Hasil: ")
        cetakUrutan(n, true)
        fmt.Println()
    }

```

### Screenshoot Output



```

Masukkan bilangan positif N: 5
Hasil: 5 4 3 2 1 2 3 4 5
PS D:\Kuliah\SEM 3\praktikum aln

```

### Deskripsi Program

Program di atas mencetak pola urutan bilangan dari n hingga 1, lalu kembali dari 1 hingga n, membentuk pola simetris seperti gunung. Fungsi cetakUrutan menerima dua parameter: n, yaitu bilangan yang sedang dicetak, dan descending, boolean yang menunjukkan apakah urutan saat ini menurun (jika true) atau menaik (jika false). Pada awalnya, fungsi mencetak n, lalu jika descending bernilai true, program memanggil cetakUrutan dengan n-1 untuk terus menurun hingga mencapai 1. Saat mencapai 1, fungsi beralih ke mode menaik dengan memanggil cetakUrutan(2, false). Dalam mode menaik, fungsi memanggil cetakUrutan(n+1, false) hingga mencapai nilai awal n. Pada fungsi main, pengguna memasukkan bilangan positif n. Jika bilangan negatif atau nol dimasukkan, program menampilkan pesan kesalahan. Namun, jika valid, program menetapkan n sebagai max dan memanggil cetakUrutan untuk mencetak pola yang diinginkan.

## UNGUIDED 5

### Sourcecode

```

package main

import "fmt"
// NAMA : Egi Umar Ferdhika
// NIM : 2311102277

```

```

func cetakGanjil(current, n int) {
    if current > n {
        return
    }

    if current%2 != 0 {
        fmt.Printf("%d ", current)
    }

    cetakGanjil(current+1, n)
}

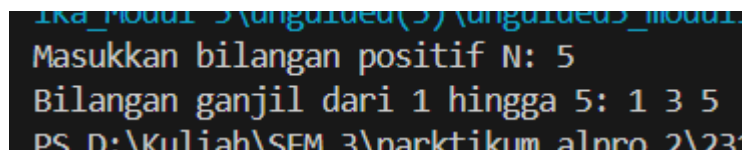
func main() {
    var n int
    fmt.Print("Masukkan bilangan positif N: ")
    fmt.Scan(&n)

    if n <= 0 {
        fmt.Println("error")
        return
    }

    fmt.Print("Bilangan ganjil dari 1 hingga ", n, ": ")
    cetakGanjil(1, n)
    fmt.Println()
}

```

### Screenshoot Output



```

ika_Modul_5\unguided(5)\unguided5_modul1
Masukkan bilangan positif N: 5
Bilangan ganjil dari 1 hingga 5: 1 3 5
PS D:\Kuliah\SEM 3\praktikum_alpro_2\23

```

### Deskripsi Program

Program di atas mencetak bilangan ganjil dari 1 hingga n, di mana n adalah bilangan positif yang dimasukkan oleh pengguna. Fungsi cetakGanjil digunakan untuk mencetak bilangan ganjil tersebut dengan cara rekursif. Fungsi ini menerima dua parameter: current, yang menunjukkan bilangan saat ini yang sedang diperiksa, dan n, yaitu batas akhir. Setiap kali fungsi dipanggil, program memeriksa apakah current lebih besar dari n; jika ya, fungsi berhenti (return). Jika tidak, program memeriksa apakah current adalah bilangan ganjil (dengan kondisi  $\text{current} \% 2 \neq 0$ ); jika benar, bilangan tersebut dicetak. Setelah itu, fungsi memanggil dirinya sendiri dengan  $\text{current}+1$  untuk memeriksa bilangan berikutnya. Di dalam fungsi main, pengguna diminta memasukkan nilai n. Jika n bernilai negatif atau nol,

program akan menampilkan pesan kesalahan. Jika valid, program akan mencetak bilangan ganjil dari 1 hingga n menggunakan cetakGanjil yang dimulai dari 1.

## UNGUIDED 6

### Sourcecode

```
package main

import "fmt"
// NAMA : Egi Umar Ferdhika
// NIM : 2311102277
func pangkat(x, y int) int {
    if y == 0 {
        return 1
    }
    if y == 1 {
        return x
    }

    return x * pangkat(x, y-1)
}

func main() {
    var x, y int

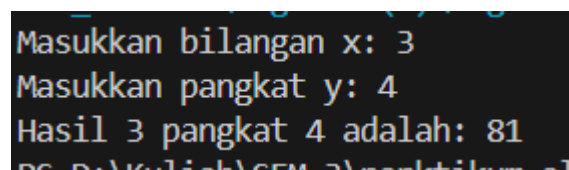
    fmt.Print("Masukkan bilangan x: ")
    fmt.Scan(&x)

    fmt.Print("Masukkan pangkat y: ")
    fmt.Scan(&y)

    if y < 0 {
        fmt.Println("Mohon masukkan pangkat non-
negatif!")
        return
    }

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

### Screenshoot Output



Masukkan bilangan x: 3  
Masukkan pangkat y: 4  
Hasil 3 pangkat 4 adalah: 81

### **Deskripsi Program**

Program di atas adalah program yang menghitung nilai  $x$  pangkat  $y$  menggunakan fungsi rekursif pangkat. Fungsi pangkat menerima dua parameter:  $x$ , yaitu bilangan yang akan dipangkatkan, dan  $y$ , yaitu pangkat yang diterapkan pada  $x$ . Dalam fungsi ini, jika  $y$  bernilai 0, hasilnya adalah 1, sesuai dengan aturan bahwa bilangan apa pun yang dipangkatkan 0 bernilai 1. Jika  $y$  bernilai 1, fungsi mengembalikan  $x$  karena bilangan berpangkat 1 tetap bernilai bilangan itu sendiri. Jika  $y$  lebih dari 1, fungsi mengembalikan hasil perkalian  $x * \text{pangkat}(x, y-1)$ , yang memanggil dirinya sendiri dengan nilai  $y$  berkurang 1 hingga mencapai salah satu kondisi dasar. Di dalam fungsi main, program meminta pengguna memasukkan bilangan  $x$  dan pangkat  $y$ . Jika pengguna memasukkan pangkat negatif, program menampilkan pesan kesalahan. Jika pangkat valid (yaitu, non-negatif), program akan memanggil fungsi pangkat dan menampilkan hasilnya.