

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL V  
REKURSIF**



**Disusun Oleh :**

**Marsep Trianto Pakondo / 2311102251**

**IF-11-06**

**Dosen Pengampu :**

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

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## **I. DASAR TEORI**

Rekursif berarti bahwa suatu proses bisa memanggil dirinya sendiri. Rekursif adalah kemampuan suatu rutin untuk memanggil dirinya sendiri. Dalam Rekursif sebenarnya terkandung pengertian prosedur dan fungsi. Perbedaannya adalah bahwa rekursif bisa memanggil ke dirinya sendiri, tetapi prosedur dan fungsi harus dipanggil lewat pemanggil prosedur dan fungsi. Rekursif merupakan teknik pemrograman yang penting dan beberapa bahasa pemrograman mendukung keberadaan proses rekursif ini. Dalam prosedur dan fungsi, pemanggilan ke dirinya sendiri bisa berarti proses berulang yang tidak bisa diketahui kapan akan berakhir.

- **Prosedur Dan Fungsi Rekursif**

Prosedur dan fungsi merupakan sub program yang sangat bermanfaat dalam pemrograman, terutama untuk program atau proyek yang besar. Manfaat penggunaan sub program antara lain adalah :

1. meningkatkan readability, yaitu mempermudah pembacaan program
2. meningkatkan modularity, yaitu memecah sesuatu yang besar menjadi modulmodul atau bagian-bagian yang lebih kecil sesuai dengan fungsinya, sehingga mempermudah pengecekan, testing dan lokalisasi kesalahan.
3. meningkatkan reusability, yaitu suatu sub program dapat dipakai berulang kali
4. dengan hanya memanggil sub program tersebut tanpa menuliskan perintahperintah yang semestinya diulang-ulang.

## II. GUIDED

### Soal Studi Case

XXXXXXXXXXXXXXXXXXXX

### Sourcecode

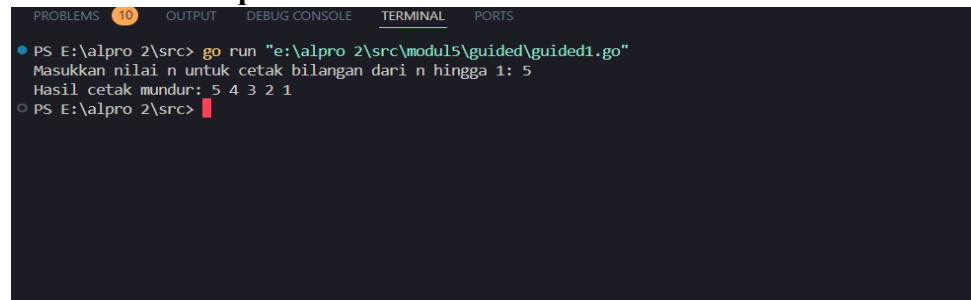
```
package main

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.Scanln(&n)
    fmt.Print("Hasil cetak mundur: ")
    cetakMundur(n)
}
```

### Screenshoot Output



```
PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS E:\alpro 2\src> go run "e:\alpro 2\src\modul5\guided\guided1.go"
Masukkan nilai n untuk cetak bilangan dari n hingga 1: 5
Hasil cetak mundur: 5 4 3 2 1
PS E:\alpro 2\src>
```

### Deskripsi Program

Program diatas digunakan untuk mencetak nilai mundur. Dalam fungsi main pengguna diminta memasukkan nilai n. Setelah itu, fungsi cetakMundur akan dipanggil dengan argumen n. Fungsi ini bekerja secara

*rekursif dikarenakan fungsi ini memanggil dirinya sendiri didalam fungsi itu sendiri. Setiap kali fungsi dipanggil, nilai n akan dicetak, kemudian fungsi memanggil dirinya sendiri dengan nilai n-1. Proses ini berulang hingga n mencapai 1.*

### Sourcecode

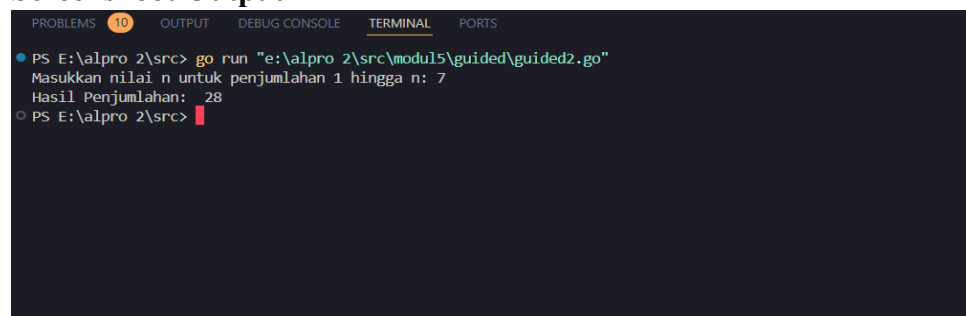
```
package main

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: ")
    n: ")
    fmt.Scanln(&n)
    fmt.Println("Hasil Penjumlahan: ", jumlahRekursif(n))
}
```

### Screenshoot Output



```
PS E:\alpro 2\src> go run "e:\alpro 2\src\modul5\guided\guided2.go"
Masukkan nilai n untuk penjumlahan 1 hingga n: 7
Hasil Penjumlahan: 28
PS E:\alpro 2\src>
```

### Deskripsi Program

*Program diatas digunakan untuk menghitung jumlah bilangan dari 1 hingga n. Program ini dimulai dengan meminta pengguna memasukkan nilai n. Kemudian, fungsi jumlahRekursif akan dipanggil dengan argumen n. Fungsi ini bekerja dengan cara membagi masalah menjadi masalah yang lebih kecil. Jika n sama dengan 1, maka fungsi langsung*

*mengembalikan 1. Jika tidak, fungsi akan mengembalikan hasil penjumlahan dari n dengan hasil pemanggilan fungsi jumlahRekursif dengan argumen n-1. Proses ini berulang hingga n mencapai 1.*

### Sourcecode

```
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 mencari 2 pangkat n: ")
    fmt.Scanln(&n)
    fmt.Println("Hasil 2 Pangkat ", n, " : ", pangkatDua(n))
}
```

### Screenshoot Output

### Deskripsi Program

*Program diatas digunakan untuk menghitung nilai 2 pangkat n. Program ini dimulai dengan meminta pengguna memasukkan nilai n. Fungsi pangkatDua kemudian dipanggil dengan argumen n. Fungsi ini bekerja dengan cara membagi masalah menjadi masalah yang lebih kecil. Jika n sama dengan 0, maka fungsi langsung mengembalikan 1 (karena 2 pangkat 0 = 1). Jika tidak, fungsi akan mengembalikan hasil perkalian 2 dengan hasil pemanggilan fungsi pangkatDua dengan argumen n-1. Proses ini berulang hingga n mencapai 0.*

## Sourcecode

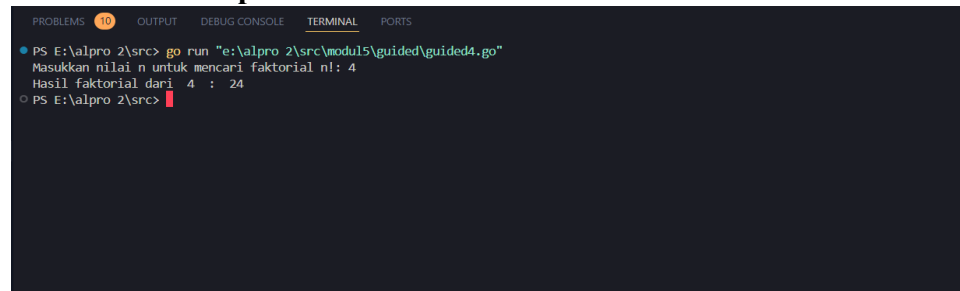
```
package main

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 Output

A screenshot of a Go IDE's terminal window. The terminal shows the command to run a Go program: `PS E:\alpro 2\src> go run "e:\alpro 2\src\modul5\guided\guided4.go"`. The output of the program is displayed below the command: `Masukkan nilai n untuk mencari faktorial n!: 4` followed by `Hasil faktorial dari 4 : 24`. The terminal interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS.

## Deskripsi Program

*Program diatas digunakan untuk menghitung faktorial dari suatu bilangan bulat  $n$ . Program ini dimulai dengan meminta pengguna memasukkan nilai  $n$ . Fungsi faktorial kemudian dipanggil dengan argumen  $n$ . Fungsi ini bekerja dengan cara membagi masalah menjadi masalah yang lebih kecil. Jika  $n$  sama dengan 0 atau 1, maka faktorialnya adalah 1. Jika tidak, faktorial dari  $n$  adalah hasil perkalian  $n$  dengan faktorial dari  $n-1$ . Proses ini berulang hingga  $n$  mencapai 0 atau 1.*

### III. UNGUIDED

#### Soal Studi Case

XXXXXXXXXXXXXXXXXXXX

#### Sourcecode

```
package main

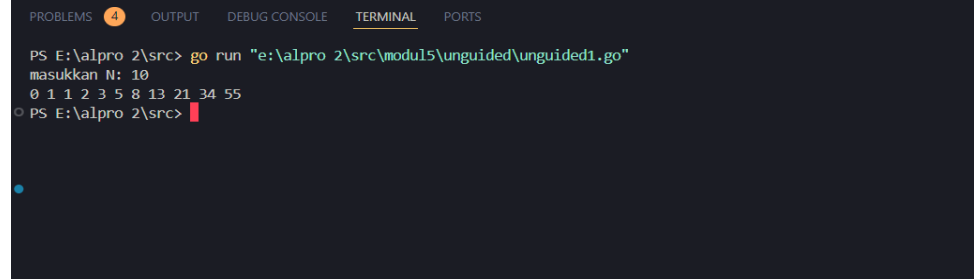
import "fmt"

// Fungsi rekursif untuk menghitung Fibonacci
func fibonacci(n int) int {
    if n <= 1 {
        return n
    }
    return fibonacci(n-1) + fibonacci(n-2)
}

func printFibo(n int) {
    for i := 0; i <= n; i++ {
        fmt.Print(fibonacci(i), " ")
    }
}

func main() {
    var n int
    fmt.Print("masukkan N: ")
    fmt.Scan(&n)
    printFibo(n)
}
```

#### Screenshoot Output



#### Deskripsi Program

Program diatas digunakan untuk menghitung dan mencetak deret Fibonacci hingga suku ke-n. Fungsi fibonacci merupakan inti dari

*perhitungan, di mana nilai Fibonacci suatu bilangan diperoleh dari penjumlahan nilai Fibonacci dua bilangan sebelumnya. Fungsi ini secara rekursif memanggil dirinya sendiri hingga mencapai kondisi dasar. Fungsi printFibo digunakan untuk mencetak deret Fibonacci hingga suku ke-n dengan memanfaatkan fungsi fibonacci. Fungsi main bertindak sebagai titik awal program, meminta input dari pengguna, dan memanggil fungsi printFibo.*

### Sourcecode

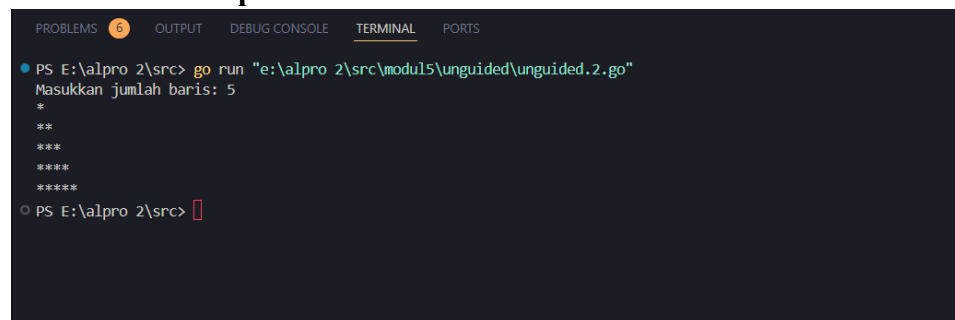
```
package main

import "fmt"

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

func main() {
    var N int
    fmt.Print("Masukkan jumlah baris: ")
    fmt.Scan(&N)
    printStars(N)
}
```

### Screenshot Output



```
PS E:\alpro 2\src> go run "e:\alpro 2\src\modul5\unguided\unguided.2.go"
Masukkan jumlah baris: 5
*
**
***
****
*****
PS E:\alpro 2\src>
```



## Deskripsi Program

*Program diatas digunakan untuk mencetak pola segitiga siku-siku yang terdiri dari tanda bintang. Program ini bekerja secara rekursif dengan memanfaatkan fungsi printStars. Fungsi ini akan terus memanggil dirinya sendiri dengan nilai yang lebih kecil hingga mencapai kondisi dasar (jumlah baris sama dengan 0). Pada setiap pemanggilan, fungsi akan mencetak sejumlah tanda bintang sesuai dengan jumlah baris yang diminta, lalu berpindah ke baris baru.*

## Sourcecode

```
package main

import "fmt"

func faktor(n, x int) {
    if x == n {
        fmt.Print(x)
    } else {
        if n % x == 0 {
            fmt.Print(x, " ")
            faktor(n, x+1)
        } else {
            faktor(n, x+1)
        }
    }
}

func main() {
    var n int

    fmt.Print("Masukkan bilangan N : ")
    fmt.Scan(&n)

    faktor(n, 1)
}
```

## Screenshoot Output

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS E:\alpro 2\src> go run "e:\alpro 2\src\modul5\unguided\unguided3.go"
Masukkan bilangan N : 5
1 5
PS E:\alpro 2\src> go run "e:\alpro 2\src\modul5\unguided\unguided3.go"
Masukkan bilangan N : 12
1 2 3 4 6 12
PS E:\alpro 2\src>
```

## Deskripsi Program

Program diatas digunakan untuk mencari dan mencetak semua faktor dari sebuah bilangan bulat. Fungsi faktor menerima dua input, yaitu bilangan yang akan dicari faktornya ( $n$ ) dan bilangan pembagi yang akan dicoba ( $x$ ). Fungsi ini bekerja secara rekursif dengan mencoba membagi  $n$  dengan bilangan dari 1 hingga  $n$  secara berurutan. Jika  $n$  habis dibagi  $x$ , maka  $x$  adalah faktor dari  $n$  dan akan dicetak. Proses ini berulang dengan nilai  $x$  yang terus bertambah hingga mencapai nilai  $n$ . Jika  $n$  tidak habis dibagi  $x$ , maka fungsi akan langsung mencoba dengan nilai  $x$  berikutnya.

## Sourcecode

```
package main

import "fmt"

func barisBilanganTurun(n, x int) {
    if x == 0 {
        return
    } else {
        if x >= 1 {
            fmt.Print(x, " ")
            barisBilanganTurun(n, x-1)
        }
    }
}

func barisBilanganNaik(n, x int) {
    if x > n {
        return
    } else {
        if x <= n {
```

```

        fmt.Print(x, " ")
        barisBilanganNaik(n, x+1)
    }
}

func main() {
    var n int

    fmt.Print("Masukkan bilangan N : ")
    fmt.Scan(&n)

    x := n
    barisBilanganTurun(n, x)
    barisBilanganNaik(n, 2)
}

```

### Screenshoot Output

```

PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL PORTS
● PS E:\alpro 2\src> go run "e:\alpro 2\src\modul5\unguided\unguided4.go"
Masukkan bilangan N : 5
5 4 3 2 1 2 3 4 5
● PS E:\alpro 2\src> go run "e:\alpro 2\src\modul5\unguided\unguided4.go"
Masukkan bilangan N : 9
9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9
○ PS E:\alpro 2\src>

```

### Deskripsi Program

*Program diatas memiliki dua fungsi utama: barisBilanganTurun dan barisBilanganNaik. Fungsi barisBilanganTurun mencetak bilangan dari nilai maksimum (yang ditentukan pengguna) hingga 1 secara menurun, sementara fungsi barisBilanganNaik mencetak bilangan dari 2 hingga nilai maksimum secara menaik.*

## Sourcecode

```
package main

import "fmt"

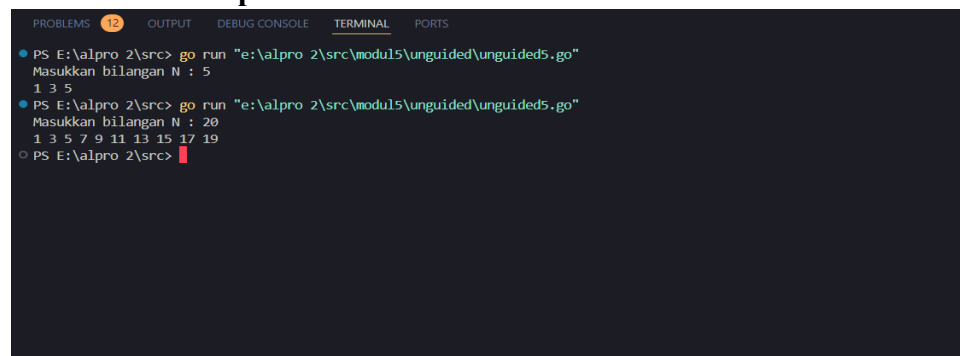
func bilanganGanjil(n, x int) {
    if x == n {
        if x % 2 != 0 {
            fmt.Print(x)
        }
    } else {
        if x % 2 != 0 {
            fmt.Print(x, " ")
            bilanganGanjil(n, x+1)
        } else {
            bilanganGanjil(n, x+1)
        }
    }
}

func main() {
    var n int

    fmt.Print("Masukkan bilangan N : ")
    fmt.Scan(&n)

    bilanganGanjil(n, 1)
}
```

## Screenshoot Output



The screenshot shows a Go IDE interface with a terminal window. The terminal displays the execution of a Go program. The first run shows the input '5' and the output '1 3 5'. The second run shows the input '20' and the output '1 3 5 7 9 11 13 15 17 19'. The terminal window has tabs for PROBLEMS (12), OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The terminal text is as follows:

```
PS E:\alpro 2\src> go run "e:\alpro 2\src\modul5\unguided\unguided5.go"
Masukkan bilangan N : 5
1 3 5
PS E:\alpro 2\src> go run "e:\alpro 2\src\modul5\unguided\unguided5.go"
Masukkan bilangan N : 20
1 3 5 7 9 11 13 15 17 19
PS E:\alpro 2\src>
```

## Deskripsi Program

*Program diatas digunakan untuk mencetak semua bilangan ganjil dari 1 hingga nilai n. Fungsi bilanganGanjil secara berulang memeriksa setiap bilangan mulai dari 1 hingga n. Jika bilangan tersebut ganjil, maka akan langsung dicetak. Kemudian, fungsi akan memanggil dirinya sendiri untuk memeriksa bilangan berikutnya. Proses ini berulang terus hingga semua bilangan dari 1 sampai n telah diperiksa.*

## Sourcecode

```
package main
import "fmt"

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

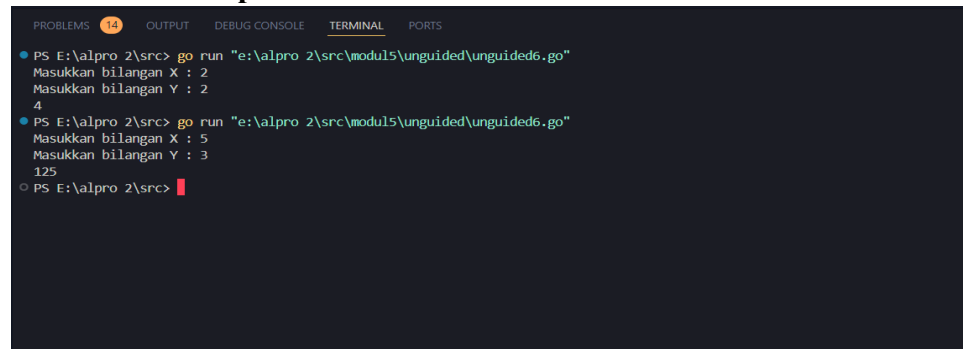
    return x * pangkat(x, y-1)
}

func main() {
    var x, y int

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

    fmt.Print(pangkat(x, y))
}
```

## Screenshoot Output



```
PROBLEMS 14 OUTPUT DEBUG CONSOLE TERMINAL PORTS
● PS E:\alpro 2\src> go run "e:\alpro 2\src\modul5\unguided\unguided6.go"
Masukkan bilangan X : 2
Masukkan bilangan Y : 2
4
● PS E:\alpro 2\src> go run "e:\alpro 2\src\modul5\unguided\unguided6.go"
Masukkan bilangan X : 5
Masukkan bilangan Y : 3
125
○ PS E:\alpro 2\src>
```

## Deskripsi Program

*Program diatas digunakan untuk menghitung nilai pangkat dari suatu bilangan. Fungsi pangkat menerima dua input, yaitu bilangan pokok (x) dan pangkat (y). Fungsi ini bekerja dengan cara berikut: jika pangkat (y) adalah 0, maka hasil pangkatnya adalah 1 (karena bilangan apa pun pangkat 0 adalah 1). Jika tidak, fungsi akan mengalikan bilangan pokok (x) dengan hasil pemanggilan fungsi pangkat itu sendiri dengan pangkat yang dikurangi 1 (y-1). Proses ini akan terus berulang hingga pangkat mencapai 0. Didalam fungsi main, pengguna diminta untuk memasukkan bilangan pokok (x) dan pangkat (y), kemudian memanggil fungsi pangkat untuk menghitung dan mencetak hasilnya.*