

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL 5

REKURSIF



Oleh:

Raja Muhammad Lufhti

103112400027

12 IF 01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

Rekursif adalah teknik pemrograman di mana suatu fungsi memanggil dirinya sendiri untuk menyelesaikan masalah dengan cara membagi menjadi sub-masalah yang lebih kecil hingga mencapai kondisi dasar (*base-case*).

Komponen Rekursif

1. Base-case Kondisi yang menghentikan rekursi agar tidak berjalan terus-menerus.
2. Recursive-case Bagian di mana fungsi memanggil dirinya sendiri dengan parameter yang mendekati *base-case*.

Cara Kerja Rekursif

-) Forward Fungsi terus memanggil dirinya sendiri hingga mencapai *base-case*.
-) Backward Setelah mencapai *base-case*, fungsi mulai kembali ke pemanggil sebelumnya.

Keunggulan Rekursif

-) Mempermudah pemecahan masalah kompleks.
-) Kode lebih ringkas dan mudah dibaca untuk masalah yang bersifat rekursif.

Kelemahan Rekursif

-) Konsumsi memori lebih besar dibanding iterasi.
-) Jika tidak memiliki *base-case* yang benar, dapat menyebabkan *infinite loop* atau *stack overflow*.

Contoh Penggunaan Rekursif

-) Mencari faktor bilangan
-) Menghitung faktorial
-) Menghitung pangkat
-) Algoritma pencarian dan sorting (misalnya QuickSort, MergeSort)

Kesimpulan:

Rekursif adalah metode penyelesaian masalah dengan memanggil fungsi itu sendiri hingga mencapai kondisi akhir tertentu. Teknik ini berguna untuk menyelesaikan masalah yang dapat dipecah menjadi bagian yang lebih kecil, tetapi perlu digunakan dengan hati-hati untuk menghindari konsumsi memori berlebih.

II. GUIDED

Contoh 1

```
package main
import "fmt"
func pangkatIteratif(base, exp int) int {
    hasil := 1

    for i := 0; i < exp; i++ {
        hasil *= base
    }

    return hasil
}
func faktorialIteratif(n int) int {
    hasil := 1
    for i := 2; i <= n; i++ {
        hasil *= i
    }
    return hasil
}
func main() {
    var base, exp, n int
    fmt.Print("Masukkan bilangan: ")
    fmt.Scanln(&base)
    fmt.Print("Masukkan pangkat: ")
    fmt.Scanln(&exp)

    fmt.Printf("%d^%d = %d\n", base, exp, pangkatIteratif(base, exp))
    fmt.Print("Masukkan angka untuk faktorial: ")
    fmt.Scanln(&n)
    fmt.Printf("%d! = %d\n", n, faktorialIteratif(n))
}
```

Screenshots Output

```
PS D:\ALPRO 2\103112400027_MODUL 5> go run "d:\ALPRO 2\103112400027_MODUL 5\guidedrek1\1r.go"
Masukkan bilangan: 3
Masukkan pangkat: 3
3^3 = 27
Masukkan angka untuk faktorial: 5
5! = 120
PS D:\ALPRO 2\103112400027_MODUL 5> █
```

// Foto hasil dari menjalankan code

Deskripsi: Program ini menghitung perpangkatan dan faktorial secara rekursif. Fungsi pangkatRekursif menghitung perpangkatan dengan mengalikan base hingga eksponen nol, sementara faktorialRekursif menghitung faktorial dengan mengalikan n hingga satu. Pada main, pengguna memasukkan bilangan, pangkat, dan angka untuk faktorial, lalu hasilnya ditampilkan.

Contoh 2

```
package main

import "fmt"
func pangkatRekursif(base, exp int) int {
    if exp == 0 {
        return 1
    }
    return base * pangkatRekursif(base, exp-1)
}
func faktorialRekursif(n int) int {
    if n == 0 || n == 1 {
        return 1
    }
    return n * faktorialRekursif(n-1)
}

func main() {
    var base, exp, n int

    fmt.Print("masukkan bilangan : ")
    fmt.Scanln(&base)
    fmt.Print("masukkan pangkat : ")
    fmt.Scanln(&exp)
```

```
fmt.Printf("%d pangkat %d = %d\n", base, exp, pangkatRekursif(base, exp))

fmt.Print("masukkan angka untuk faktorial : ")
fmt.Scanln(&n)

fmt.Printf("%d! = %d\n", n, faktorialRekursif(n))
}
```

Screenshots Output

```
PS D:\ALPRO 2\103112400027_MODUL 5> go run "d:\ALPRO 2\103112400027_MODUL 5\guidedrek2\2r.go"
masukkan bilangan : 3
masukkan pangkat : 3
3 pangkat 3 = 27
masukkan angka untuk faktorial : 5
5! = 120
PS D:\ALPRO 2\103112400027_MODUL 5> █
```

// Foto hasil dari menjalankan code

Deskripsi: Program ini menghitung perpangkatan dan faktorial menggunakan metode iteratif. Fungsi pangkatIteratif menghitung hasil perpangkatan dengan mengalikan base sebanyak exp kali dalam perulangan for, sedangkan fungsi faktorialIteratif menghitung faktorial dengan mengalikan angka dari 2 hingga n. Dalam fungsi main, pengguna memasukkan bilangan dan pangkatnya, serta angka untuk faktorial, kemudian hasil perhitungan ditampilkan.

III. UNGUIDED

Soal 1

```
//103112400027_RAJA MUHAMMAD LUFHTI
package main

import "fmt"

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

func main() {
    var q int
    fmt.Print("Masukkan angka: ")
    fmt.Scan(&q)
    fmt.Print("Fibonacci: ")
    for i := 0; i <= q; i++ {
        fmt.Print(fibonacci(i), " ")
    }
    fmt.Println()
}
```

Screenshots Output



```
PS D:\ALPRO 2\103112400027_MODUL 5> go run "d:\ALPRO 2\103112400027_MODUL 5\un1\u1.go"
Masukkan angka: 10
Fibonacci: 0 1 1 2 3 5 8 13 21 34 55
PS D:\ALPRO 2\103112400027_MODUL 5> 
```

// Foto hasil dari menjalankan code

Deskripsi: Program ini menghitung dan menampilkan deret Fibonacci menggunakan rekursi. Fungsi fibonacci(ok) mengembalikan nilai Fibonacci ke-ok dengan menjumlahkan dua nilai sebelumnya hingga mencapai 0 atau 1 sebagai *base-case*. Dalam fungsi main(), pengguna memasukkan sebuah angka q, lalu program mencetak deret Fibonacci sebanyak q angka dengan memanggil fungsi fibonacci(i) dalam perulangan.

Soal 2

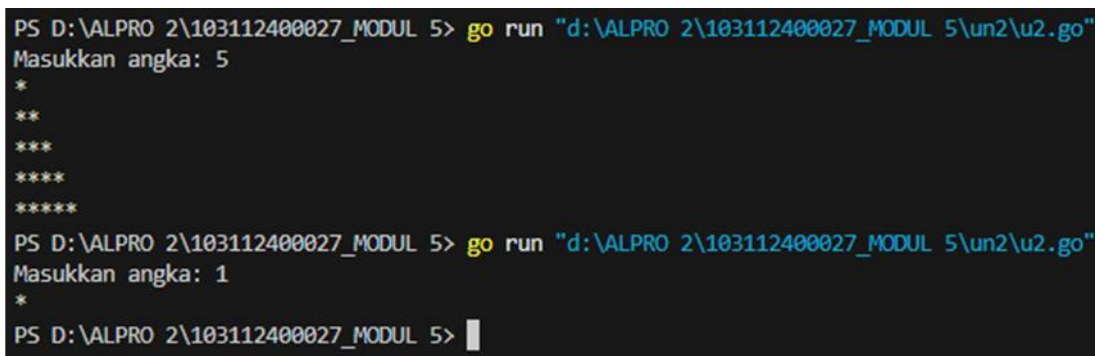
```
//103112400027_RAJA MUHAMMAD
LUFHTI
package main

import "fmt"

func cetakBintang(n int) {
    for i := 1; i <= n; i++ {
        for j := 1; j <= i; j++ {
            fmt.Print("*")
        }
        fmt.Println()
    }
}

func main() {
    var n int
    fmt.Print("Masukkan angka: ")
    fmt.Scan(&n)
    cetakBintang(n)
}
```

Screenshots Output



```
PS D:\ALPRO 2\103112400027_MODUL 5> go run "d:\ALPRO 2\103112400027_MODUL 5\un2\u2.go"
Masukkan angka: 5
*
**
***
****
*****
PS D:\ALPRO 2\103112400027_MODUL 5> go run "d:\ALPRO 2\103112400027_MODUL 5\un2\u2.go"
Masukkan angka: 1
*
PS D:\ALPRO 2\103112400027_MODUL 5> █
```

// Foto hasil dari menjalankan code

Deskripsi: Program ini mencetak pola segitiga bintang menggunakan perulangan *nested*. Fungsi cetakBintang(n) memiliki dua perulangan bersarang di mana perulangan luar mengontrol jumlah baris, dan perulangan dalam mencetak bintang sesuai jumlah barisnya. Dalam fungsi main(), pengguna memasukkan angka n,

lalu program memanggil cetakBintang(n) untuk mencetak pola bintang sebanyak n baris.

Soal 3

```
//103112400027_RAJA MUHAMMAD LUFHTI
package main

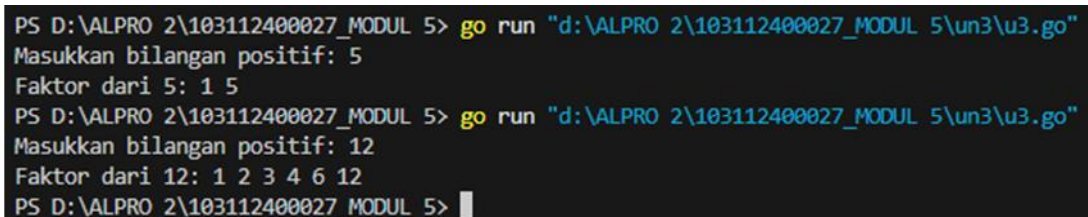
import (
    "fmt"
)

func cariFaktorial(angka, pembagi int) {
    if pembagi > angka {
        return
    }
    if angka%pembagi == 0 {
        fmt.Print(pembagi, " ")
    }
    cariFaktorial(angka, pembagi+1)
}

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

    fmt.Print("Faktor dari ", n, ": ")
    cariFaktorial(n, 1)
    fmt.Println()
}
```

Screenshots Output



```
PS D:\ALPRO 2\103112400027_MODUL 5> go run "d:\ALPRO 2\103112400027_MODUL 5\un3\u3.go"
Masukkan bilangan positif: 5
Faktor dari 5: 1 5
PS D:\ALPRO 2\103112400027_MODUL 5> go run "d:\ALPRO 2\103112400027_MODUL 5\un3\u3.go"
Masukkan bilangan positif: 12
Faktor dari 12: 1 2 3 4 6 12
PS D:\ALPRO 2\103112400027_MODUL 5> █
```

// Foto hasil dari menjalankan code

Deskripsi : Program ini mencari dan menampilkan faktor dari suatu bilangan menggunakan rekursi. Fungsi cariFaktorial(angka, pembagi) memeriksa apakah pembagi dapat membagi angka tanpa sisa, lalu mencetaknya jika memenuhi syarat, dan memanggil dirinya sendiri dengan pembagi+1 hingga melebihi angka. Dalam fungsi main(), pengguna memasukkan bilangan positif n, lalu program memanggil cariFaktorial(n, 1) untuk mencetak semua faktor dari n.

IV. KESIMPULAN

Materi ini membahas konsep rekursif dan iteratif dalam pemrograman, khususnya dalam bahasa Go. Rekursif adalah teknik pemanggilan fungsi yang memanggil dirinya sendiri hingga mencapai *base-case*, sementara iteratif menggunakan perulangan untuk menyelesaikan masalah. Beberapa penerapan rekursif dan iteratif yang dibahas meliputi perhitungan faktorial, perpangkatan, deret Fibonacci, pencarian faktor bilangan, serta pencetakan pola bintang. Rekursif memberikan solusi yang lebih sederhana dan elegan untuk masalah yang bersifat berulang, tetapi dapat mengonsumsi lebih banyak memori dibanding iterasi. Oleh karena itu, pemilihan antara rekursif dan iteratif harus disesuaikan dengan kebutuhan dan efisiensi program.

V. REFERENSI

MODUL 5 REKURSIF ALGORITMA PEMROGRAMAN