LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 5

MATERI

Oleh:

NAMA:Muhammad Fahruli Ma’ruf

NIM:103112400057

KELAS: 12-IF-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

Rekursi adalah teknik penyelesaian masalah dengan cara memecah masalah utama menjadi sub-masalah yang serupa namun lebih kecil. Teknik ini sering menjadi alternatif dari struktur kontrol perulangan, di mana subprogram (baik fungsi maupun prosedur) akan memanggil dirinya sendiri.

Dalam setiap algoritma rekursif, terdapat dua komponen utama yang harus didefinisikan:

1. **Base Case (Basis)**: Ini adalah kondisi di mana proses rekursif akan berhenti. Base case sangat penting dan merupakan hal pertama yang harus ditentukan saat membangun program rekursif, karena tanpanya rekursi akan berjalan tanpa henti (infinite recursion).
2. **Recursive Case**: Ini adalah kondisi di mana subprogram melakukan pemanggilan dirinya sendiri. Kondisi recursive case biasanya merupakan kebalikan atau pelengkap dari base case, yang memastikan bahwa masalah terus dipecah hingga mencapai base case.

II. GUIDED

```go
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))
}
```

Penjelasan: Program ini meminta dua input dari pengguna untuk perhitungan pangkat (bilangan dasar dan pangkat), kemudian satu input lagi untuk perhitungan faktorial. Setelah menerima input, program akan menghitung dan menampilkan hasil perpangkatan serta hasil faktorial dari bilangan yang dimasukkan.

```
PS C:\Users\HP\OneDrive\modul5> go run "c:\Users\HP\OneDrive\modul5\contoh1\contoh1.go"
masukkan bilangan: 5
masukkan pangkat: 2
5^2 = 25
masukkan angka untuk faktorial: 5
5! = 120
```

```go
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^%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))
}

```

Penjelasan: Kode ini berisi dua fungsi utama yang menggunakan prinsip rekursi:

1. **pangkatRekursif(base, exp int)**: Fungsi ini menghitung nilai perpangkatan (base dipangkatkan exp) secara rekursif. Basis kasusnya adalah ketika exp bernilai 0, maka hasilnya 1.

2. **faktorialRekursif(n int)**: Fungsi ini menghitung nilai faktorial dari bilangan n secara rekursif. Basis kasusnya adalah ketika n bernilai 0 atau 1, maka hasilnya 1.

3. Fungsi main() akan meminta pengguna untuk memasukkan bilangan dasar dan pangkat untuk perhitungan perpangkatan, lalu meminta satu bilangan lagi untuk perhitungan faktorial. Setelah itu, program akan memanggil fungsi rekursif yang relevan dan menampilkan hasilnya di konsol.

```
PS C:\Users\HP\OneDrive\modul5> go run "c:\Users\HP\OneDrive\modul5\contoh2\2.go"
masukkan bilangan: 5
masukkan pangkat: 2
5^2 = 25
masukkan angka untuk faktorial: 10
10! = 3628800
```

UNGUIDED

soal1 > 1.go > ...

```go
package main

import "fmt"

func fibonacci(n int) int {
    if n <= 1 {
        return n
    }
    a, b := 0, 1
    for i := 2; i <= n; i++ {
        a, b = b, a+b
    }
    return b
}

func main() {
    var n int
    fmt.Print("Masukkan jumlah suku Fibonacci: ")
    fmt.Scan(&n)

    if n < 0 {
        fmt.Println("Error: Jumlah suku tidak boleh negatif!")
        return
    }

    fmt.Print("Deret Fibonacci: ")
    for i := 0; i < n; i++ {
        fmt.Printf("%d ", fibonacci(i))
    }
    fmt.Println()
}
```

Penjelasan: Program Go ini dirancang untuk menghasilkan dan menampilkan deret bilangan Fibonacci hingga jumlah suku tertentu yang ditentukan oleh pengguna.

Program ini terdiri dari dua bagian utama:

1. **Fungsi fibonacci(n int) int**:
    - Fungsi ini bertanggung jawab untuk menghitung suku Fibonacci ke-n.
    - Ia memiliki **basis kasus** yang menangani nilai n kurang dari atau sama dengan 1, di mana ia akan langsung mengembalikan n itu sendiri (misalnya, suku ke-0 adalah 0, suku ke-1 adalah 1).

- Untuk nilai n yang lebih besar dari 1, fungsi ini menggunakan pendekatan **iteratif** (perulangan). Dimulai dengan a=0 dan b=1 (dua suku pertama deret Fibonacci), ia kemudian menghitung suku-suku berikutnya dengan menjumlahkan dua suku sebelumnya di setiap iterasi, hingga mencapai suku ke-n.

2. **Fungsi main()**:
    - Ini adalah titik masuk utama program.
    - Pertama, ia akan meminta pengguna untuk memasukkan **jumlah suku deret Fibonacci** yang ingin dihasilkan.
    - Program kemudian melakukan **validasi input**; jika pengguna memasukkan angka negatif, ia akan menampilkan pesan kesalahan dan menghentikan eksekusi, karena jumlah suku tidak bisa negatif.

```
PS C:\Users\HP\OneDrive\modul5> go run "c:\Users\HP\OneDrive\modul5\soal1\1.go"
Masukkan jumlah suku Fibonacci: 10
Deret Fibonacci: 0 1 1 2 3 5 8 13 21 34
```

```go
package main

import "fmt"

func cetakBintang(n int) {
    if n <= 0 {
        fmt.Println("Error: Jumlah baris harus positif!")
        return
    }

    for i := 1; i <= n; i++ {
        for j := 0; j < i; j++ {
            fmt.Print("*")
        }
        fmt.Println()
    }
}

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

Penjelasan: Program ini akan meminta Anda memasukkan sebuah angka. Angka tersebut akan menentukan berapa baris bintang yang akan dicetak. Jika Anda memasukkan angka 5, misalnya, program akan mencetak pola segitiga bintang

```
PS C:\Users\HP\OneDrive\modul5> go run "c:\Users\HP\OneDrive\modul5\soal2\2.go"
Masukkan jumlah baris: 3
*
**
***
PS C:\Users\HP\OneDrive\modul5> go run "c:\Users\HP\OneDrive\modul5\soal2\2.go"
Masukkan jumlah baris: 4
*
**
***
****
```

```go
package main

import "fmt"

func cariFaktor(n int, faktor int) {
    if faktor > n {
        return
    }
    if n%faktor == 0 {
        fmt.Printf("%d ", faktor)
    }
    cariFaktor(n, faktor+1)
}

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

    fmt.Printf("Faktor dari %d adalah: ", n)
    cariFaktor(n, 1)
    fmt.Println()
}
```

Penjelasan: Kode Go ini adalah program sederhana yang dirancang untuk menemukan dan mencetak semua faktor dari sebuah bilangan bulat positif yang dimasukkan oleh pengguna. Program ini menggunakan pendekatan rekursif untuk mencari faktor-faktor tersebut. Ketika program ini dijalankan, ia akan meminta Anda untuk memasukkan sebuah bilangan (misalnya, 12). Setelah Anda memasukkannya, program akan mulai mencari faktor-faktor dari 12 secara berurutan, dimulai dari 1

```
PS C:\Users\HP\OneDrive\modul5> go run "c:\Users\HP\OneDrive\modul5\soal3\3.go"
N: 5
Faktor dari 5 adalah: 1 5
PS C:\Users\HP\OneDrive\modul5> go run "c:\Users\HP\OneDrive\modul5\soal3\3.go"
N: 12
Faktor dari 12 adalah: 1 2 3 4 6 12
```

III. KESIMPULAN

Konsep rekursi menawarkan metode alternatif dalam penyelesaian masalah pemrograman, khususnya di Bahasa Go, sebagai pengganti perulangan. Teknik ini terbukti efektif untuk berbagai persoalan komputasi, seperti perhitungan faktorial, perpangkatan bilangan, deret Fibonacci, pembentukan pola bintang, hingga pencarian faktor bilangan.

Berdasarkan implementasi yang telah dilakukan, dapat disimpulkan bahwa rekursi memiliki keunggulan dalam hal keringkasan kode dan peningkatan keterbacaan program. Namun, penting untuk selalu mendefinisikan *base case* dengan tepat guna mencegah rekursi tak terbatas yang dapat menyebabkan stack overflow.

Secara keseluruhan, praktikum ini telah memberikan pemahaman mendalam tentang rekursi dan penerapannya dalam pemrograman, sekaligus memperkuat kemampuan penggunaan Bahasa Go untuk menyelesaikan beragam permasalahan secara efisien.

IV. REFERENSI