

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL 5

REKURSI



Oleh:

NAMA: Lutfi Shidqi Mardian

NIM: 103112400077

KELAS: IF-12-01

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2025

I. DASAR TEORI

Dalam pemrograman Golang, rekursi adalah metode di mana suatu fungsi atau prosedur dapat memanggil dirinya sendiri untuk menyelesaikan suatu permasalahan. Rekursi sering digunakan dalam berbagai algoritma pemrograman, terutama pada masalah yang dapat dipecah menjadi sub-masalah yang lebih kecil dengan pola yang sama.

1. Pengertian Rekursi

Rekursi adalah suatu teknik pemrograman yang memungkinkan suatu fungsi memanggil dirinya sendiri dalam proses eksekusinya. Rekursi digunakan untuk menyelesaikan masalah yang memiliki pola pengulangan, seperti pencarian dalam struktur data berbasis pohon, perhitungan faktorial, dan deret Fibonacci

2. Deklarasi dan Pemanggilan Fungsi

Dalam implementasi rekursi, terdapat dua bagian utama yang harus diperhatikan:

- **Base-Case (Kondisi Berhenti):** Suatu kondisi yang menentukan kapan rekursi harus berhenti agar tidak terjadi infinite loop.
- **Recursive-Case:** Kondisi di mana fungsi akan terus memanggil dirinya sendiri hingga mencapai base-case.

Berikut adalah contoh implementasi fungsi rekursi dalam bahasa Go

```
func cetakBilangan(n int) {  
    if n == 0 {  
        return  
    }  
    fmt.Println(n)  
    cetakBilangan(n - 1)  
}
```

Pada contoh di atas, fungsi cetakBilangan(n) akan terus memanggil dirinya sendiri hingga mencapai kondisi `n == 0`, yang merupakan base-case.

3. Contoh Implementasi Rekursi dalam Golang

a. Faktorial dengan Rekursi

Faktorial dari suatu bilangan n dapat dihitung menggunakan rekursi dengan rumus:

$$n! = n \times (n-1)! \quad n! = n \times (n-1)!$$

Berikut adalah implementasinya dalam Golang:

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

Pada contoh ini, base-case terjadi saat $n == 0$ atau $n == 1$, di mana fungsi akan mengembalikan nilai 1.

b. Deret Fibonacci dengan Rekursi

Deret Fibonacci didefinisikan sebagai berikut:

$$F(n) = \begin{cases} 0, & \text{jika } n = 0 \\ 1, & \text{jika } n = 1 \\ F(n-1) + F(n-2), & \text{jika } n > 1 \end{cases}$$

Berikut adalah implementasi Fibonacci secara rekursif dalam Golang:

```
func fibonacci(n int) int {  
    if n == 0 {  
        return 0  
    } else if n == 1 {  
        return 1  
    }  
  
    return fibonacci(n-1) + fibonacci(n-2)  
}
```

Dalam contoh ini, fungsi fibonacci(n) akan terus dipanggil hingga mencapai base-case saat $n = 0$ atau $n = 1$.

4. Keuntungan dan Kekurangan Rekursi

Keuntungan

- Kode lebih sederhana dan lebih mudah dibaca untuk beberapa masalah.
- Memudahkan penyelesaian masalah yang secara alami memiliki sifat rekursif (misalnya, pencarian dalam struktur pohon).
- Mengurangi kompleksitas kode jika dibandingkan dengan iterasi dalam beberapa kasus.

Kekurangan

- Dapat menyebabkan stack overflow jika tidak memiliki base-case yang jelas.
- Memerlukan lebih banyak memori karena menyimpan banyak panggilan fungsi dalam stack.
- Biasanya lebih lambat dibandingkan dengan metode iteratif karena banyaknya pemanggilan fungsi.

II. GUIDED

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
    //Input pangkat
    fmt.Print("Masukkan bilangan: ")
    fmt.Scan(&base)
```

```

    fmt.Print("Masukkan pangkat: ")

    fmt.Scan(&exp)

    fmt.Printf("%d^%d = %d\n", base, exp,
pangkatIteratif(base, exp))

    //input faktorial

    fmt.Print("Masukkan angka untuk faktorial: ")

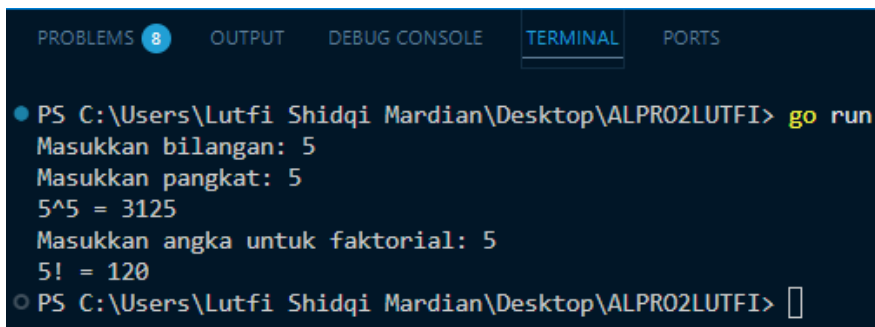
    fmt.Scan(&n)

    fmt.Printf("%d! = %d\n", n, faktorialIteratif(n))

}

```

Output Screenshot:



```

PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS
● PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
Masukkan bilangan: 5
Masukkan pangkat: 5
5^5 = 3125
Masukkan angka untuk faktorial: 5
5! = 120
○ PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> 

```

Penjelasan:

Program ini merupakan implementasi dalam bahasa Go yang menghitung perpangkatan dan faktorial secara iteratif. Fungsi pangkatIteratif menerima dua bilangan, yaitu basis dan eksponen, lalu menghitung hasil perpangkatan menggunakan perulangan. Sementara itu, fungsi faktorialIteratif menghitung faktorial dari suatu bilangan dengan mengalikan semua bilangan dari 1 hingga n. Program meminta input dari pengguna untuk kedua operasi ini dan menampilkan hasilnya di layar. Perhitungan dan mencetak slip gaji dengan format yang rapi. Semua nilai ditampilkan menggunakan fmt.Printf agar hasil lebih mudah dibaca.

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
    //Input pangkat
    fmt.Print("Masukkan bilangan: ")
    fmt.Scan(&base)

    fmt.Print("Masukkan pangkat: ")
    fmt.Scan(&exp)
```

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

        //input faktorial

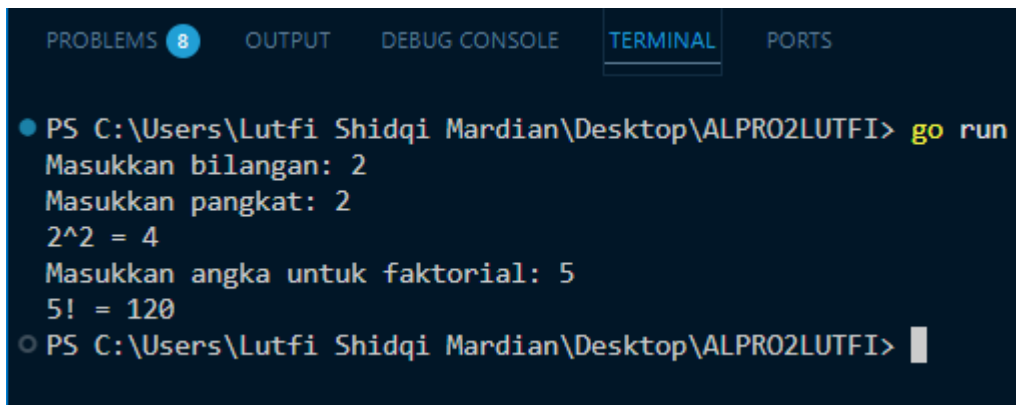
        fmt.Print("Masukkan angka untuk faktorial: ")

        fmt.Scan(&n)

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

    }
```

Output Screenshot

A screenshot of a terminal window with a dark blue background. At the top, there are tabs labeled 'PROBLEMS 8', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is active and highlighted), and 'PORTS'. The terminal shows the execution of a Go program. The prompt is 'PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>'. The user enters 'go run', and the program outputs 'Masukkan bilangan: 2', 'Masukkan pangkat: 2', and '2^2 = 4'. Then, the user enters '5' for the factorial prompt, and the program outputs 'Masukkan angka untuk faktorial: 5' and '5! = 120'. The prompt returns to 'PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>'.

Penjelasan:

Program ini merupakan implementasi dalam bahasa Go yang menghitung perpangkatan dan faktorial menggunakan metode rekursif. Fungsi `pangkatRekursif` menghitung hasil perpangkatan dengan memanggil dirinya sendiri hingga eksponen mencapai nol, sedangkan fungsi `faktorialRekursif` menghitung faktorial suatu bilangan dengan cara yang sama hingga mencapai satu atau nol sebagai kasus dasar. Program meminta input dari pengguna untuk kedua operasi ini dan menampilkan hasilnya di layar.

III. UNGUIDED

1.

```
package main\  
  
import "fmt"  
  
func main() {  
    var n int  
    fmt.Scan(&n)  
  
    for i := 0; i <= n; i++ {  
        fmt.Print(fibonacci(i), " ")  
    }  
}  
  
func fibonacci(n int) int{  
    if n == 0 {  
        return 0  
    } else if n == 1 {  
        return 1  
    }  
    return fibonacci(n - 1) + fibonacci(n - 2)  
}
```

Output Screenshot:



```
PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run  
10  
0 1 1 2 3 5 8 13 21 34 55  
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>
```

Penjelasan:

Program ini merupakan implementasi dalam bahasa Go untuk mencetak deret Fibonacci hingga suku ke-n menggunakan metode rekursif. Fungsi fibonacci secara rekursif menghitung nilai Fibonacci dengan menjumlahkan dua nilai sebelumnya hingga mencapai kasus dasar, yaitu ketika n bernilai 0 atau 1. Program meminta input berupa bilangan n dari pengguna, lalu mencetak deret Fibonacci dari suku pertama hingga ke-n dalam satu baris.

2.

```
package main

import "fmt"

func asteriks(n, i int) {
    if i > n {
        return
    }

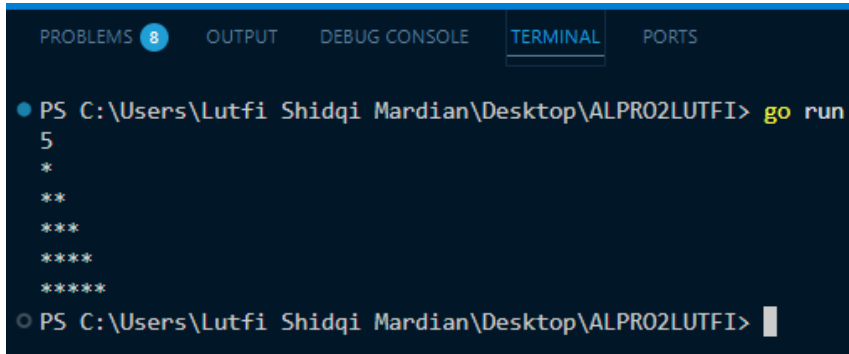
    for j := 0; j < i; j++ {
        fmt.Print("*")
    }
    fmt.Println()

    asteriks(n, i + 1)
}

func main() {
    var n int
    fmt.Scan(&n)

    asteriks(n, 1)
}
```

Output Screenshot:



```
PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
5
*
**
***
****
*****
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>
```

Penjelasan:

Program ini merupakan implementasi dalam bahasa Go untuk mencetak pola segitiga bintang (*) menggunakan rekursi. Fungsi asteriks menerima dua parameter, yaitu n sebagai jumlah baris yang diinginkan dan i sebagai baris saat ini. Fungsi ini mencetak i bintang pada setiap baris, lalu memanggil dirinya sendiri dengan $i + 1$ hingga mencapai n . Program meminta input n dari pengguna dan memulai pencetakan pola bintang dari satu hingga n baris secara rekursif.

3.

```
package main

import "fmt"

func faktor(n, i int) {
    if i > n {
        return
    }

    if n % i == 0 {
        fmt.Print(i, " ")
    }

    faktor(n, i + 1)
}

func main() {
    var n int
    fmt.Scan(&n)

    faktor(n, 1)
}
```

Output Screenshot:



```
PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
5
1 5
● PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
12
1 2 3 4 6 12
○ PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> 
```

Penjelasan:

Program ini merupakan implementasi dalam bahasa Go untuk mencari dan mencetak semua faktor dari suatu bilangan menggunakan rekursi. Fungsi faktor menerima dua parameter, yaitu n sebagai bilangan yang akan dicari faktornya dan i sebagai angka yang digunakan untuk membagi n . Jika i habis membagi n , maka i dicetak sebagai faktor. Fungsi ini kemudian memanggil dirinya sendiri dengan $i + 1$ hingga i melebihi n . Program meminta input n dari pengguna dan mencetak semua faktor dari n dalam satu baris.

IV. KESIMPULAN

Dari berbagai implementasi rekursi dalam pemrograman Golang, dapat disimpulkan bahwa rekursi merupakan teknik yang berguna untuk menyelesaikan masalah dengan cara memecahnya menjadi sub-masalah yang lebih kecil dan identik. Dengan adanya base-case, proses rekursif dapat dikendalikan agar tidak berjalan tanpa henti, sedangkan recursive-case memungkinkan pemanggilan fungsi secara berulang. Rekursi dapat menggantikan perulangan dalam beberapa kasus, terutama pada permasalahan yang secara alami memiliki sifat rekursif seperti faktorial, deret Fibonacci, dan pencarian pangkat. Meskipun rekursi memudahkan pemahaman konsep tertentu, penggunaannya harus diperhatikan agar tidak menyebabkan stack overflow akibat pemanggilan fungsi yang berlebihan.

V. REFERENSI

1. Modul 5 - Praktikum Algoritma dan Pemrograman 2, Telkom University Purwokerto.
2. Donovan, A. A., & Kernighan, B. W. (2015). *The Go Programming Language*. Addison-Wesley.
3. Official Golang Documentation. (n.d.). *Recursion in Go*. Retrieved from https://go.dev/doc/effective_go#recursion
4. W3Schools. (n.d.). *Golang Recursion*. Retrieved from https://www.w3schools.com/go/go_recursion.asp
5. GeeksforGeeks. (n.d.). *Recursion in Golang*. Retrieved from <https://www.geeksforgeeks.org/recursion-in-golang/>