

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL 4

REKURSIF



Oleh:

NAMA: ICHYA ULUMIDDIIN

NIM: 103112400076

KELAS: 12IF-01-A

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2025

I. DASAR TEORI

Rekursi adalah konsep dalam pemrograman di mana sebuah fungsi memanggil dirinya sendiri untuk menyelesaikan masalah yang lebih kompleks dengan memecahnya menjadi sub-masalah yang lebih sederhana. Dalam bahasa Go, rekursi didukung sepenuhnya, memungkinkan fungsi untuk memanggil dirinya sendiri selama terdapat kondisi penghentian yang jelas untuk mencegah loop tak terbatas.

Struktur Fungsi Rekursif:

- **Kasus Dasar (Base Case):** Kondisi yang menentukan kapan fungsi harus berhenti memanggil dirinya sendiri. Tanpa kasus dasar, rekursi akan berjalan tanpa henti dan dapat menyebabkan kesalahan seperti stack overflow.
- **Kasus Rekursif:** Bagian di mana fungsi memanggil dirinya sendiri dengan parameter yang dimodifikasi, mendekati kasus dasar.

Keuntungan Menggunakan Rekursi:

- **Kesederhanaan Kode:** Rekursi dapat membuat kode lebih mudah dibaca dan ditulis, terutama untuk masalah yang secara alami bersifat rekursif, seperti traversal struktur data pohon.
- **Pemecahan Masalah Kompleks:** Beberapa masalah lebih mudah dipecahkan dengan rekursi karena dapat dipecah menjadi sub-masalah yang identik dengan masalah asli.

II. GUIDED

Source Code Guided 1

```
//ICHYA ULUMIDDIIN
package main

import "fmt"

// Fungsi iteratif untuk menghitung pangkat (base^exp)
func pangkatIteratif(base, exp int) int {
    hasil := 1

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

// Fungsi iteratif untuk menghitung faktorial (n!)
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.Scanln(&base)
    fmt.Print("Masukkan pangkat: ")
    fmt.Scanln(&exp)

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

    // Input Faktorial
    fmt.Print("Masukkan angka untuk faktorial: ")
    fmt.Scanln(&n)

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

Output

```
PS C:\Users\ICHYA ULUMIDDIN\OneDrive\Documents\Guided1-rekursif\Guided1-rekursif.go"
Masukkan bilangan: 6
Masukkan pangkat: 3
6^3 = 216
Masukkan angka untuk faktorial: 8
8! = 40320
PS C:\Users\ICHYA ULUMIDDIN\OneDrive\Documents\Guided1-rekursif\Guided1-rekursif.go"
```

Penjelasan

Program Golang ini menghitung pangkat dan faktorial suatu bilangan menggunakan metode iteratif untuk efisiensi memori. Fungsi `pangkatIteratif(base, exp)` menghitung hasil pangkat dengan perulangan `for-loop`, sedangkan `faktorialIteratif(n)` menghitung faktorial dengan perkalian berulang dari 1 hingga n .

Dalam fungsi `main()`, pengguna memasukkan bilangan dan pangkatnya untuk perhitungan pangkat, serta bilangan n untuk faktorial. Program kemudian memanggil fungsi yang sesuai dan mencetak hasilnya.

Misalnya, jika pengguna memasukkan 3^4 , program mencetak 81, dan untuk $6!$, program mencetak 720. Dengan pendekatan iteratif, program lebih efisien dibandingkan rekursi, menjadikannya cocok untuk aplikasi matematika dan ilmiah.

Source Code Guided 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 bilnangan: ")
    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))
}
```

Output

```
PS C:\Users\ICHYA ULUMIDDIN\OneDrive\D
ed2-rekursif\Guided2-rekursif.go"
Masukkan bilnangan: 8
Masukkan pangkat: 3
8^3 = 512
Masukkan angka untuk faktorial: 8
8! = 40320
```

Penjelasan

Program Golang ini menghitung pangkat dan faktorial menggunakan metode rekursif. Fungsi `pangkatRekursif(base, exp)` menghitung pangkat dengan memanggil dirinya sendiri hingga `exp` mencapai 0, sedangkan `faktorialRekursif(n)` menghitung faktorial dengan memanggil dirinya sendiri hingga `n` bernilai 1.

Dalam fungsi `main()`, pengguna memasukkan bilangan dan pangkatnya serta bilangan untuk faktorial. Program kemudian memanggil fungsi rekursif yang sesuai dan mencetak hasilnya.

Source Code Guided 3

```
package main

import (
    "fmt"
    "math"
)

func luasPermukaanTabung(r, t float64) float64 {
    return 2 * math.Pi * r * (r + t)
}

func volumeTabung(r, t float64) float64 {
    return math.Pi * math.Pow(r, 2) * t
}

func main() {
    var r, t float64
    fmt.Print("Masukkan jari-jari tabung: ")
    _, errR := fmt.Scan(&r)
    fmt.Print("Masukkan tinggi tabung: ")
    _, errT := fmt.Scan(&t)

    // Memeriksa apakah input valid
    if errR != nil || errT != nil {
        fmt.Println("Input tidak valid! Harap masukkan angka yang benar.")
        return
    }
    if r <= 0 || t <= 0 {
        fmt.Println("Jari-jari dan tinggi tabung harus lebih dari nol.")
        return
    }
    luas := luasPermukaanTabung(r, t)
    volume := volumeTabung(r, t)
    fmt.Println("=====")
    fmt.Printf("Luas Permukaan Tabung: %.2f satuan2\n", luas)
    fmt.Printf("Volume Tabung: %.2f satuan3\n", volume)
    fmt.Println("=====")
}
```

Output

```
PS C:\Users\ICHYA ULUMIDDIN\OneDrive\Documents\
ed3-rekursif\Guided3-rekursif.go"
Masukkan jari-jari tabung: 8
Masukkan tinggi tabung: 7
=====
Luas Permukaan Tabung: 753.98 satuan2
Volume Tabung: 1407.43 satuan3
=====
```

Penjelasan

Program Golang ini bertujuan untuk menghitung luas permukaan dan volume tabung berdasarkan input jari-jari (r) dan tinggi (t) yang dimasukkan oleh pengguna. Program ini menggunakan dua fungsi utama, yaitu `luasPermukaanTabung(r, t)` untuk menghitung luas permukaan tabung dengan rumus $2 \times \pi \times r \times (r + t)$, dan `volumeTabung(r, t)` untuk menghitung volume tabung dengan rumus $\pi \times r^2 \times t$.

Dalam fungsi `main()`, program meminta input dari pengguna berupa jari-jari dan tinggi tabung. Sebelum melanjutkan perhitungan, program melakukan validasi input, yaitu memastikan bahwa data yang dimasukkan adalah angka yang valid dan lebih besar dari nol. Jika input tidak valid atau bernilai nol atau negatif, program akan menampilkan pesan error dan berhenti.

III. UNGUIDED

Source Code Unguided 1

```
//ICHYA ULUMIDDIIN
package main

import (
    "fmt"
)

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

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

    if n < 0 {
        fmt.Println("Masukkan angka positif!")
        return
    }

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

Output

```
PS C:\Users\ICHYA ULUMIDDIN\OneDrive\Documents\Un-  
guided1-Rekursif\Unguided1-Rekursif.go"  
Masukkan jumlah suku Fibonacci: 11  
Deret Fibonacci:  
0 1 1 2 3 5 8 13 21 34 55
```

Penjelasan

Program Golang ini menghasilkan deret Fibonacci menggunakan metode rekursif. Fungsi fibonacci(n) menghitung bilangan Fibonacci ke-n dengan kondisi dasar ($F(0) = 0$, $F(1) = 1$) dan memanggil dirinya sendiri untuk nilai n-1 dan n-2.

Di fungsi main(), pengguna memasukkan jumlah suku Fibonacci yang diinginkan. Jika input negatif, program menampilkan error. Jika valid, deret Fibonacci dicetak menggunakan perulangan for.

Misalnya, jika input n = 10, program mencetak 0 1 1 2 3 5 8 13 21 34.

Source Code Unguided 2

```
//ICHYA ULUMIDDIIN
package main

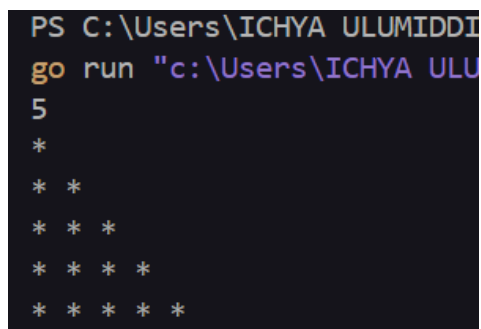
import (
    "fmt"
)

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

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

    if n < 1 {
        fmt.Println("Masukkan angka positif!")
        return
    }
    printStars(n)
}
```

Output



```
PS C:\Users\ICHYA ULUMIDDIIN> go run "c:\Users\ICHYA ULUMIDDIIN\source_code_unguided_2.go"
5
*
* *
* * *
* * * *
* * * * *
```

Penjelasan

Program Golang ini bertujuan untuk menampilkan pola bintang bertingkat menggunakan metode rekursif. Fungsi `printStars(n)` bertanggung jawab untuk mencetak pola segitiga bintang dengan jumlah baris sebanyak `n`. Dalam fungsi `printStars(n)`, terdapat basis kasus yaitu jika `n == 0`, fungsi berhenti agar rekursi tidak berjalan tanpa henti. Sebelum mencetak bintang, fungsi memanggil dirinya sendiri (`printStars(n-1)`), sehingga pola tercetak dari baris terkecil ke terbesar. Setelah rekursi selesai, program mencetak `n` bintang dalam satu baris dan melanjutkan ke baris berikutnya. Pada fungsi `main()`, program meminta pengguna memasukkan bilangan positif `n` untuk menentukan jumlah baris pola bintang. Jika input tidak valid (`n < 1`), program menampilkan pesan error. Jika valid, program memanggil `printStars(n)` untuk mencetak pola.

Source Code Unguided 3

```
//ICHYA ULUMIDDIIN
package main

import (
    "fmt"
)

func cetakFaktor(bilangan, pembagi int) {
    if pembagi > bilangan {
        return
    }

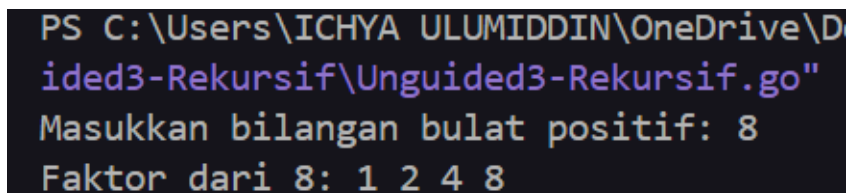
    if bilangan%pembagi == 0 {
        fmt.Printf("%d ", pembagi)
    }
    cetakFaktor(bilangan, pembagi+1)
}

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

    if bilangan < 1 {
        return
    }

    fmt.Printf("Faktor dari %d: ", bilangan)
    cetakFaktor(bilangan, 1)
    fmt.Println()
}
```

Output



```
PS C:\Users\ICHYA ULUMIDDIIN\OneDrive\Documents\Un-
guided3-Rekursif\Unguided3-Rekursif.go"
Masukkan bilangan bulat positif: 8
Faktor dari 8: 1 2 4 8
```

Penjelasan

Program Golang ini bertujuan untuk menampilkan semua faktor dari suatu bilangan bulat positif menggunakan rekursi. Faktor suatu bilangan adalah angka yang dapat membagi bilangan tersebut tanpa sisa. Program ini menggunakan fungsi rekursif `cetakFaktor(bilangan, pembagi)`, yang akan memeriksa dan mencetak faktor-faktor bilangan dari 1 hingga bilangan itu sendiri.

Fungsi `cetakFaktor(bilangan, pembagi)` memiliki basis kasus, yaitu jika pembagi lebih besar dari bilangan, maka rekursi berhenti. Jika pembagi dapat membagi bilangan tanpa sisa ($\text{bilangan} \% \text{pembagi} == 0$), maka angka tersebut dicetak sebagai faktor. Selanjutnya, fungsi memanggil dirinya sendiri dengan nilai `pembagi+1`, sehingga rekursi berjalan hingga mencapai batas bilangan.

Di dalam fungsi `main()`, program meminta pengguna untuk memasukkan bilangan bulat positif. Jika input kurang dari 1, program langsung berhenti tanpa melakukan perhitungan. Jika valid, program memanggil `cetakFaktor(bilangan, 1)` untuk mencetak faktor-faktor bilangan tersebut.

IV. KESIMPULAN

Rekursi adalah teknik pemrograman di mana sebuah fungsi memanggil dirinya sendiri untuk menyelesaikan masalah yang lebih kecil hingga mencapai basis kasus. Dalam bahasa Golang, rekursi sangat berguna untuk menyelesaikan masalah yang memiliki sifat berulang, seperti fibonacci, faktorial, pencarian faktor bilangan, dan pencetakan pola bintang.

Dari berbagai implementasi rekursi di atas, dapat disimpulkan bahwa rekursi membuat kode lebih sederhana dan mudah dipahami, terutama untuk masalah yang dapat dipecah menjadi sub-masalah yang lebih kecil. Namun, rekursi juga memiliki kekurangan, terutama dalam hal penggunaan memori dan efisiensi, karena setiap pemanggilan rekursif akan menambahkan frame baru ke dalam call stack, yang bisa menyebabkan stack overflow jika tidak dikelola dengan baik.

.

REFERENSI

Alodyasari, A. (2024, October 31). *Pengertian Rekursi, Fungsi, dan Contohnya dalam Pemrograman*. Retrieved from lawencon:
<https://www.lawencon.com/rekursi-adalah/>

School, D. (2019, March 4). *Recursion dalam Bahasa Golang*. Retrieved from kursuswebprogramming: <https://kursuswebprogramming.com/recursion-dalam-bahasa-golang/>