

LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL V
REKURSIF



Disusun Oleh :

Reza Alvonzo / 2311102026

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

Pengertian Rekursif

Rekursif adalah teknik pemrograman di mana suatu fungsi memanggil dirinya sendiri untuk menyelesaikan suatu masalah yang lebih kecil dari masalah aslinya, hingga mencapai kondisi dasar (base case) yang akan menghentikan pemanggilan rekursif. Teknik ini berguna untuk memecah masalah besar atau kompleks menjadi sub-masalah yang lebih sederhana, sehingga lebih mudah diselesaikan.

Struktur Dasar Fungsi Rekursif

Pada dasarnya, sebuah fungsi rekursif dalam Golang (dan bahasa pemrograman lain) memiliki dua komponen utama:

1. **Base Case (Kondisi Dasar):** Ini adalah kondisi yang menghentikan pemanggilan fungsi secara rekursif. Base case sangat penting, karena tanpanya fungsi akan terus memanggil dirinya sendiri, menyebabkan **infinite recursion** (rekursi tanpa henti) yang bisa mengakibatkan **stack overflow**.
2. **Recursive Case (Kondisi Rekursif):** Kondisi di mana fungsi memanggil dirinya sendiri dengan parameter yang dimodifikasi, biasanya dalam bentuk masalah yang lebih kecil atau lebih sederhana, untuk mendekati base case.

Cara Kerja Rekursi

Ketika fungsi dipanggil secara rekursif, setiap panggilan fungsi akan disimpan di dalam stack memori hingga mencapai base case. Pada saat mencapai base case, fungsi mulai kembali (unwind) dari stack, dan hasil dari setiap pemanggilan fungsi akan dievaluasi dari yang paling dalam hingga ke yang paling luar.

Setiap kali fungsi memanggil dirinya sendiri, ia menyimpan konteks (state) saat ini di dalam stack memori, sehingga ketika fungsi kembali dari pemanggilan terakhir, ia dapat melanjutkan dari konteks yang disimpan sebelumnya. Proses ini memungkinkan rekursi menyelesaikan masalah dengan cara yang lebih alami untuk beberapa jenis masalah, seperti pencarian dalam struktur data pohon, perhitungan faktorial, atau penelusuran graf.

Contoh Rekursi Sederhana dalam Golang

Misalnya, kita ingin menghitung faktorial dari suatu bilangan n . Faktorial dari n (dilambangkan dengan $n!$) adalah hasil perkalian semua bilangan bulat positif hingga n , atau dapat didefinisikan secara rekursif sebagai berikut:

$$n! = n \times (n-1)! \quad \text{dengan kondisi dasar } 0! = 1$$

II. GUIDED

1. Guided 1

Sourcecode

```
package main

import "fmt"

// Fungsi untuk mencetak bilangan dari n hingga 1
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)
}
```

Screenshot Output

```
Masukkan nilai n untuk cetak bilangan dari n hingga 1: 25
Hasil cetak mundur: 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
```

Deskripsi Program

Program ini menggunakan rekursi untuk mencetak bilangan dari n hingga 1 dengan cara memanggil fungsi cetakMundur berulang kali dengan nilai n yang berkurang. Rekursi berhenti ketika n mencapai 1, dan hasilnya adalah serangkaian bilangan yang dicetak mundur dari n hingga 1.

2. Guided 2

Soal Studi Case

XXXXXXXXXXXXXXXXXXXX

Sourcecode

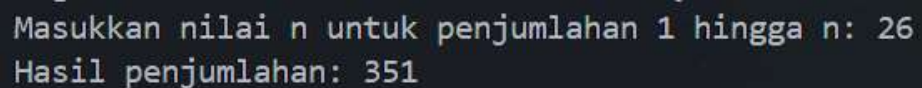
```
package main

import "fmt"

// Fungsi untuk menghitung penjumlahan 1 hingga n
func jumlahRekursi(n int) int {
    if n == 1 {
        return 1
    }
    return n + jumlahRekursi(n-1)
}

func main() {
    var n int
    fmt.Print("Masukkan nilai n untuk penjumlahan 1 hingga n: ")
    fmt.Scanln(&n)
    fmt.Println("Hasil penjumlahan:", jumlahRekursi(n))
}
```

Screenshoot Output



```
Masukkan nilai n untuk penjumlahan 1 hingga n: 26
Hasil penjumlahan: 351
```

Deskripsi Program

Program ini menggunakan rekursi untuk menghitung penjumlahan bilangan dari 1 hingga nnn. Setiap pemanggilan fungsi jumlahRekursi menjumlahkan n dengan hasil pemanggilan rekursif untuk n-1. Rekursi berakhir ketika n mencapai 1, dan hasil akhirnya adalah penjumlahan semua bilangan dari 1 hingga n.

3. Guided 3

Soal Studi Case

XXXXXXXXXXXXXXXXXXXX

Sourcecode

```
package main
```

```

import "fmt"

// Fungsi untuk mencari 2 pangkal n
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

```

Masukkan nilai n untuk mencari 2 pangkat n: 12
Hasil 2 pangkat 12 : 4096

```

Deskripsi Program

program ini meminta pengguna untuk memasukkan nilai n, lalu menggunakan rekursi untuk menghitung dan menampilkan hasil dari 2 pangkat n. Misalnya, jika pengguna memasukkan 3, program akan menghitung 2 pangkat 3, yang sama dengan 8.

4. Guided 4

Soal Studi Case

XXXXXXXXXXXXXXXXXXXX

Sourcecode

```

package main

import "fmt"

```

```
// Fungsi untuk menghitung faktorial n!
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

```
Masukkan nilai n untuk mencari faktorial n!: 26
Hasil faktorial dari 26 : -1569523520172457984
```

Deskripsi Program

program ini meminta pengguna untuk memasukkan nilai nnn, lalu menggunakan rekursi untuk menghitung dan menampilkan hasil dari faktorial nnn. Misalnya, jika pengguna memasukkan 5, program akan menghitung $5!5!5!$, yang sama dengan $5 \times 4 \times 3 \times 2 \times 1 = 120$ $\times 4 \times 3 \times 2 \times 1 = 120$ $\times 2 \times 1 = 120$ $\times 4 \times 3 \times 2 \times 1 = 120$.

III. UNGUIDED

Unguided 1

Deret fibonacci adalah sebuah deret dengan nilai suku ke-0 dan ke-1 adalah 0 dan 1, dan nilai suku ke-n selanjutnya adalah hasil penjumlahan dua suku sebelumnya. Secara umum dapat diformulasikan $S_2 = S_{n-1} + S_{n-2}$. Berikut ini adalah contoh nilai deret fibonacci hingga suku ke-10. Buatlah program yang mengimplementasikan fungsi rekursif pada deret fibonacci tersebut.

Sourcecode

```
package main

import (
    "fmt"
)

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

func main() {
    n := 10
    fmt.Printf("Deret Fibonacci hingga suku ke-%d:\n", n)
    for i := 0; i <= n; i++ {
        fmt.Printf("Suku ke-%d: %d\n", i, fibonacci(i))
    }
}

//Reza Alvonzo 2311102026
```


Screenshoot Output

```
Deret Fibonacci hingga suku ke-10:  
Suku ke-0: 0  
Suku ke-1: 1  
Suku ke-2: 1  
Suku ke-3: 2  
Suku ke-4: 3  
Suku ke-5: 5  
Suku ke-6: 8  
Suku ke-7: 13  
Suku ke-8: 21  
Suku ke-9: 34  
Suku ke-10: 55
```

Deskripsi Program

Program ini menghasilkan dan mencetak deret Fibonacci dari suku ke-0 hingga suku ke-10. Sebagai contoh, deret Fibonacci yang dihasilkan adalah 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, yang merupakan hasil dari penjumlahan dua suku sebelumnya.

Unguided 2

Buatlah sebuah program yang digunakan untuk menampilkan pola bintang berikut ini dengan menggunakan fungsi rekursif. N adalah masukan dari user.

Sourcecode

```
package main  
import "fmt"  
  
func cetakbintang(n int) {  
    if n <= 0 {  
        return  
    }  
    cetakbintang(n - 1)  
    for i := 0; i < n; i++ {  
        fmt.Print("* ")  
    }  
}
```

```

    }
    fmt.Println()
}

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

    fmt.Print ("")
    cetakbintang(n)
}

//Reza Alvonzo 2311102026

```

```
Masukkan bilangan n: 26
```

```
*  
* *  
* * *  
* * * *  
* * * * *  
* * * * * *  
* * * * * * *  
* * * * * * * *  
* * * * * * * * *  
* * * * * * * * * *  
* * * * * * * * * * *  
* * * * * * * * * * * *  
* * * * * * * * * * * * *  
* * * * * * * * * * * * * *  
* * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * * * * * * * * * * *
```

```
Masukkan bilangan n: 26
```

```
*  
* *  
* * *  
* * * *  
* * * * *  
* * * * * *  
* * * * * * *  
* * * * * * * *  
* * * * * * * *  
* * * * * * * * *  
* * * * * * * * * *  
* * * * * * * * * * *  
* * * * * * * * * * * *  
* * * * * * * * * * * * *  
* * * * * * * * * * * * * *  
* * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * * * * * * * *  
* * * * * * * * * * * * * * * * * * * * * * *
```

Deskripsi Program

Fungsi cetakbintang Fungsi ini menerima satu parameter n yang merupakan bilangan bulat. Jika n kurang dari atau sama dengan 0, fungsi akan berhenti (return). Fungsi memanggil dirinya sendiri dengan $n - 1$, sehingga menciptakan efek rekursif. Setelah panggilan rekursif, loop for digunakan untuk mencetak bintang sebanyak n kali, diikuti dengan `fmt.Println()` untuk pindah ke baris berikutnya.

Unguided 3

Buatlah program yang mengimplementasikan rekursif untuk menampilkan faktor bilangan dari suatu N , atau bilangan yang apa saja yang habis membagi N .

Masukan terdiri dari sebuah bilangan bulat positif N . Keluaran terdiri dari barisan bilangan yang menjadi faktor dari N (terurut dari 1 hingga N ya).

Sourcecode

```
package main
import (
    "fmt"
)

func findFactors(N, divisor int) {

    if divisor > N {
        return
    }

    if N%divisor == 0 {
        fmt.Printf("%d ", divisor)
    }

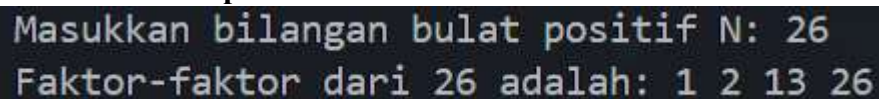
    findFactors(N, divisor+1)
}

func main() {
    var N int

    fmt.Print("Masukkan bilangan bulat positif N: ")
    fmt.Scan(&N)
```

```
    fmt.Printf("Faktor-faktor dari %d adalah: ", N)
    findFactors(N, 1)
    fmt.Println()
}
//Reza Alvonzo 2311102026
```

Screenshoot Output



```
Masukkan bilangan bulat positif N: 26
Faktor-faktor dari 26 adalah: 1 2 13 26
```

Deskripsi Program

Program ini meminta pengguna untuk memasukkan bilangan N dan menampilkan faktor-faktor dari N secara berurutan. Misalnya, jika pengguna memasukkan angka 12, output program adalah 1 2 3 4 6 12, yang merupakan semua faktor dari 12.

Unguided 4

Buatlah program yang menimplementasikan rekursif untuk menampilkan barisan bilangan tertentu

Masukan terdiri dari sebuah bilangan bulat positif N. Keluaran terdiri dari barisan bilangan dari N hingga 1 dan kembali ke N.

Sourcecode

```
package main
import (
    "fmt"
)

func printSeries(N, current int) {
    if current == 1 {
        fmt.Printf("%d ", current)
        return
    }

    fmt.Printf("%d ", current)
```

```

        printSeries(N, current-1)
        fmt.Printf("%d ", current)
    }

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

        fmt.Printf("Barisan bilangan dari %d hingga 1 dan
kembali ke %d:\n", N, N)
        printSeries(N, N)
        fmt.Println()
    }
//Reza Alvonzo 2311102026

```

Screenshoot Output

```

Masukkan bilangan bulat positif N: 26
Barisan bilangan dari 26 hingga 1 dan kembali ke 26:
26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26

```

Deskripsi Program

Program ini meminta pengguna memasukkan bilangan NNN dan mencetak deret bilangan dari NNN hingga 1, kemudian kembali naik ke NNN secara simetris. Misalnya, jika pengguna memasukkan angka 4, output program akan berupa 4 3 2 1 2 3 4, yang mencerminkan pola penurunan dan kenaikan yang simetris.

Unguided 5

Buatlah program yang menimplementasikan rekursif untuk menampilkan barisan bilangan ganjil.

Masukan terdiri dari sebuah bilangan bulat positif N. Keluaran terdiri dari barisan bilangan ganjil dari 1 hingga N.

Sourcecode

```

package main
import (

```

```

    "fmt"
)

func printOddNumbers(N, current int) {
    if current > N {
        return
    }
    if current%2 != 0 {
        fmt.Printf("%d ", current)
    }
    printOddNumbers(N, current+1)
}

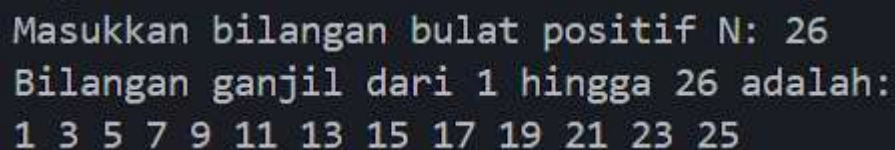
func main() {
    var N int
    fmt.Print("Masukkan bilangan bulat positif N: ")
    fmt.Scan(&N)

    fmt.Printf("Bilangan ganjil dari 1 hingga %d
adalah:\n", N)
    printOddNumbers(N, 1)
    fmt.Println()
}

//Reza Alvonzo 2311102026

```

Screenshoot Output



```

Masukkan bilangan bulat positif N: 26
Bilangan ganjil dari 1 hingga 26 adalah:
1 3 5 7 9 11 13 15 17 19 21 23 25

```

Deskripsi Program

Program ini meminta pengguna untuk memasukkan bilangan NNN dan kemudian mencetak semua bilangan ganjil dari 1 hingga NNN. Misalnya, jika pengguna memasukkan angka 10, output program akan berupa 1 3 5 7 9, yang merupakan semua bilangan ganjil dari 1 hingga 10.

Unguided 6

Buatlah program yang menimplementasikan rekursif untuk menampilkan barisan bilangan ganjil.

Masukan terdiri dari sebuah bilangan bulat positif N. Keluaran terdiri dari barisan bilangan ganjil dari 1 hingga N.

Sourcecode

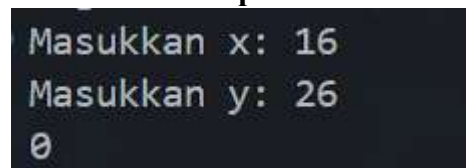
```
package main
import "fmt"

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

func main() {
    var x, y int
    fmt.Print("Masukkan x: ")
    fmt.Scanln(&x)
    fmt.Print("Masukkan y: ")
    fmt.Scanln(&y)

    fmt.Println(pangkat(x, y))
}
//Reza Alvonzo 2311102026
```

Screenshoot Output



```
Masukkan x: 16
Masukkan y: 26
0
```

Deskripsi Program

Program ini meminta pengguna untuk memasukkan bilangan x (basis) dan y (eksponen), kemudian menghitung dan menampilkan hasil dari x^y menggunakan rekursi. Misalnya, jika pengguna memasukkan x=2 dan y=3, output program akan berupa 8, yang merupakan hasil dari $2^3 = 2 \times 2 \times 2 = 8$