LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 5

REKURSIF

Oleh:

HISYAM NURDIATMOKO

103112400049

IF-12-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

Rekursif dalam Pemrograman

Rekursif adalah suatu teknik dalam pemrograman di mana sebuah fungsi atau prosedur memanggil dirinya sendiri untuk menyelesaikan sub-masalah yang lebih kecil dari masalah utama. Teknik ini sangat berguna untuk memecahkan masalah yang memiliki struktur berulang atau dapat dibagi menjadi bagian-bagian yang lebih sederhana.

1. Pengertian Rekursif

Rekursif dapat didefinisikan sebagai cara menyelesaikan suatu masalah dengan cara membagi masalah tersebut menjadi sub-masalah yang identik. Proses ini terus berlanjut hingga kondisi dasar (base-case) tercapai, yang akan menghentikan pemanggilan rekursif lebih lanjut.

2. Komponen Rekursif

Terdapat dua komponen utama dalam algoritma rekursif:

- Base-case (Kondisi dasar): Merupakan kondisi yang menghentikan rekursi. Tanpa base-case, rekursi akan berjalan tanpa henti, menyebabkan stack overflow.
- Recursive-case (Kondisi rekursif): Merupakan bagian dari algoritma yang memanggil dirinya sendiri untuk menyelesaikan sub-masalah yang lebih kecil dari masalah utama. Kondisi ini adalah negasi dari base-case dan akan terus dipanggil hingga kondisi base-case tercapai.

3. Algoritma Rekursif

Algoritma rekursif terdiri dari dua langkah utama:

- Memecah masalah menjadi sub-masalah yang lebih kecil.
- Mengembalikan hasil dari pemecahan sub-masalah ke solusi utama setelah rekursi selesai.

Setiap algoritma rekursif dapat diterjemahkan dalam bentuk algoritma iteratif, namun terkadang rekursi memberikan solusi yang lebih elegan, terutama untuk masalah yang berbentuk hierarkis atau berulang.

4. Pentingnya Base-case

Base-case adalah komponen yang sangat penting dalam rekursi. Tanpa adanya base-case yang jelas, algoritma rekursif akan berjalan tanpa henti dan menghasilkan kesalahan sistem. Misalnya, dalam perhitungan

faktorial, base-case adalah saat nilai n mencapai 0 atau 1, yang menyatakan bahwa faktorial dari 0 atau 1 adalah 1.

5. Proses Forward dan Backward
Pada saat eksekusi rekursif, terjadi dua fase utama:

- Forward (Pemanggilan maju): Fungsi akan terus memanggil dirinya sendiri, menyelesaikan sub-masalah yang lebih kecil.
- Backward (Pemanggilan mundur): Setelah kondisi base-case tercapai, proses rekursif akan mulai mundur, dan hasil dari sub-masalah akan digabungkan untuk membentuk solusi akhir.

6. Contoh Pemakaian Rekursif
Beberapa masalah yang dapat diselesaikan menggunakan rekursi antara lain:

- Faktorial: Menghitung faktorial dari suatu angka n.
- Fibonacci: Menghitung deret Fibonacci.
- Pencarian dan pengurutan: Seperti algoritma Quick Sort dan Merge Sort yang berbasis rekursif.

II. GUIDED

Source code Guided 1:

guid1 > 103112400049_Guided1.go > main

```go
package main

import "fmt"

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

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

// fungsi interatif 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 faktorial
    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 D:\1_Matkul\Alpro2\week4\103112400049_MODUL5> go run "d:\1_Matkul\Alpro2\
Masukkan bilangan: 2
Masukkan pangkat: 5
2^5 = 32
Masukkan angka untuk faktorial: 10
10! = 3628800
PS D:\1_Matkul\Alpro2\week4\103112400049_MODUL5>
```

Deskripsi :
Program Go Guided 1 ini menghitung pangkat dan faktorial menggunakan metode iterasi. Fungsi pangkatIteratif(base, exp int) int menghitung pangkat dengan mengalikan bilangan base sebanyak exp kali, sementara fungsi faktorialIteratif(n int) int menghitung faktorial dengan mengalikan setiap bilangan mulai dari 1 hingga n. Di dalam fungsi main, program guided 1 meminta pengguna untuk memasukkan nilai untuk pangkat dan faktorial, kemudian menghitung hasilnya menggunakan kedua fungsi tersebut dan menampilkan hasilnya.

Source code Guided 2 :

guid2 > 103112400049_Guided2.go > ...

```go
package main

import "fmt"

// fungsi rekursif untuk menghitung pangkat (base^exp)
func pangkatRekursif(base, exp int) int {
    if exp == 0 {
        return 1
    }
    return base * pangkatRekursif(base, exp-1)
}

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

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

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

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

Output :

```
PS D:\1_Matkul\Alpro2\week4\103112400049_MODUL5> go run "d:\1_Matkul\Alpro2\w
Masukkan bilangan: 2
Masukkan pangkat: 5
2^5 = 32
Masukkan angka untuk faktorial: 10
10! = 3628800
PS D:\1_Matkul\Alpro2\week4\103112400049_MODUL5>
```

Deskripsi :
Program Go Guided 2 ini menghitung pangkat dan faktorial menggunakan metode rekursif. Fungsi pangkatRekursif(base, exp int) int menghitung pangkat dengan memanggil dirinya sendiri hingga mencapai kondisi dasar (exp == 0), yang mengembalikan 1. Fungsi faktorialRekursif(n int) int menghitung faktorial dengan memanggil dirinya sendiri, mengalikan angka n dengan faktorial dari n-1, dan berhenti ketika n mencapai 0 atau 1. Di dalam fungsi main, program guided 2 meminta pengguna untuk memasukkan nilai untuk menghitung pangkat dan faktorial, kemudian menghitung hasilnya menggunakan kedua fungsi rekursif tersebut dan menampilkan hasil perhitungannya.

III. UNGUIDED

Source code Unguided 1 :

unguided1 > 103112400049_Unguided1.go > hitungFibonacci

```go
package main

//HISYAM NURDIATMOKO 103112400049
import "fmt"

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

func cetakFibonacci(n int) {
    for i := 0; i <= n; i++ {
        fmt.Print(hitungFibonacci(i), " ")
    }
    fmt.Println()
}

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

Output:

```
PS D:\1_Matkul\Alpro2\week4\103112400049_MODUL5> go run "d:\1_Matkul"
10
0 1 1 2 3 5 8 13 21 34 55
PS D:\1_Matkul\Alpro2\week4\103112400049_MODUL5>
```

Deskripsi:

Program Go Unguided 1 ini menghitung dan mencetak deret Fibonacci hingga angka ke-n. Fungsi hitungFibonacci(x int) menggunakan iterasi untuk menghitung angka Fibonacci ke-x. Fungsi cetakFibonacci(n int) mencetak deret Fibonacci dari 0 hingga n. Di fungsi main, program Unguided 1 meminta input n dan mencetak deret Fibonacci sesuai nilai n yang diinputkan.

Source code Unguided 2 :

unguided2 > 103112400049_Unguided2.go > bintang

```go
package main

//HISYAM NURDIATMOKO
import "fmt"

func main() {
    var n int
    fmt.Scan(&n)
    bintang(n, 1)
}

func bintang(n, i int) {
    if i > n {
        return
    }
    for j := 0; j < i; j++ {
        fmt.Print("*")
    }
    fmt.Println()
    bintang(n, i+1)
}
```

Output:

```
PS D:\1_Matkul\Alpro2\week4\103112400049_MODUL5> go run "d:\1_M
5
*
**
***
****
*****
```

Deskripsi:

Program Go unguided 2 ini mencetak pola bintang segitiga naik dengan rekursi. Fungsi bintang(n, i int) mencetak bintang sebanyak i pada setiap baris, dimulai dari 1 bintang hingga mencapai n bintang di baris terakhir. Fungsi ini dipanggil secara rekursif dengan meningkatkan nilai i hingga melebihi n, yang menghentikan rekursi. Di dalam fungsi main, program unguided 2 meminta input nilai n dari pengguna dan kemudian memanggil fungsi bintang untuk mencetak pola bintang.

Source code Unguided 3 :

unguided3 > 103112400049_Unguided3.go > cariFaktor

```go
package main

//HISYAM NURDIATMOKO 103112400049
import "fmt"

func main() {
    var n int
    fmt.Scan(&n)
    cariFaktor(n, 1)
}

func cariFaktor(n, i int) {
    if i > n {
        return
    }
    if n%i == 0 {
        fmt.Print(i, " ")
    }
    cariFaktor(n, i+1)
}
```

Output:

```
PS D:\1_Matkul\Alpro2\week4\103112400049_MODUL5> go run "d:\1_Matkul\
5
1 5
PS D:\1_Matkul\Alpro2\week4\103112400049_MODUL5> go run "d:\1_Matkul\
12
1 2 3 4 6 12
PS D:\1_Matkul\Alpro2\week4\103112400049_MODUL5>
```

Deskripsi:

Program Go Unguided 3 ini mencari dan mencetak faktor-faktor dari angka n menggunakan rekursi. Fungsi cariFaktor(n, i int) memeriksa setiap angka i dari 1 hingga n untuk melihat apakah i merupakan faktor dari n (dengan memeriksa apakah n % i == 0). Jika iya, maka angka i dicetak. Fungsi ini dipanggil secara rekursif dengan meningkatkan nilai i hingga lebih besar dari n, yang menghentikan rekursi. Di dalam fungsi main, program unguided 3 meminta input nilai n dari pengguna dan kemudian memanggil fungsi cariFaktor untuk mencetak faktor-faktor dari angka n.

IV. KESIMPULAN

Pada praktikum Modul 5 ini, kita mempelajari konsep rekursi dalam pemrograman menggunakan bahasa Go. Rekursi adalah teknik di mana sebuah fungsi memanggil dirinya sendiri untuk menyelesaikan sub-masalah, yang berulang hingga mencapai kondisi dasar. Praktikum ini mencakup penerapan rekursi dalam beberapa masalah, seperti menghitung pangkat, faktorial, deret Fibonacci, mencetak pola bintang, dan mencari faktor dari suatu angka. Melalui implementasi rekursif dan iteratif, kita dapat melihat perbedaan cara penyelesaian masalah dengan pendekatan rekursif yang lebih elegan dan sering kali lebih ringkas. Dengan memahami cara kerja rekursi, kita dapat menyelesaikan masalah yang memiliki struktur berulang dengan lebih efisien.

V. REFERENSI

Modul 5 Rekursif. Praktikum Algoritma Pemrograman 2