LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 5

REKURSIF

Oleh:

Muhammad Faris Rachmadi

103112400079

IF12-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

1. Pengertian Rekursi

Rekursi adalah teknik dalam pemrograman di mana suatu fungsi memanggil dirinya sendiri untuk menyelesaikan masalah. Proses ini sering digunakan untuk membagi masalah besar menjadi sub-masalah yang lebih kecil hingga masalah tersebut dapat diselesaikan. Rekursi terdiri dari dua bagian utama: base case (kasus dasar) dan recursive case (kasus rekursif).

a) **Base Case (Kasus Dasar):** Kondisi yang menghentikan pemanggilan fungsi rekursif. Ini mencegah pemanggilan yang tidak terhingga dan mengarah pada kesalahan seperti stack overflow.
b) **Recursive Case (Kasus Rekursif):** Bagian dari fungsi yang memanggil dirinya sendiri dengan parameter yang lebih kecil, mendekati kondisi dasar.

2. Karakteristik Rekursi dalam Go

a) **Penyelesaian Masalah Terstruktur:** Rekursi sering digunakan dalam masalah yang memiliki struktur terulang, seperti perhitungan faktorial, deret Fibonacci, dan pencarian dalam struktur data seperti pohon biner atau graf.
b) **Pemanfaatan Stack:** Setiap pemanggilan fungsi rekursif akan ditempatkan pada stack, yang akan dieksekusi setelah mencapai base case.
c) **Penyelesaian Masalah dengan Pembagian Sub-Masalah:** Rekursi cocok digunakan untuk masalah yang dapat dipecah menjadi sub-masalah yang lebih kecil, sehingga menyelesaikan bagian-bagian tersebut dapat mengarah pada penyelesaian keseluruhan masalah.

II. GUIDED

Guided 1

Code:

103112400079_Guided1 > Guided1.go > ...

```go
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\Faris\Documents\ALGORITMA PEMROGRAMAN 2\103112400079_
Masukkan bilangan: 6
Masukkan pangkat: 3
6^3 = 216
```

Deskripsi:

Kode di atas adalah program Go yang terdiri dari dua fungsi utama: pangkatIteratif untuk menghitung hasil perpangkatan (base^exp) secara iteratif, dan faktorialIteratif untuk menghitung faktorial dari suatu angka (n!) juga secara iteratif. Fungsi pangkatIteratif menggunakan loop untuk mengalikan bilangan dasar (base) sebanyak exp kali, sedangkan faktorialIteratif mengalikan angka dari 1 hingga n untuk menghitung faktorial. Di dalam fungsi main, program meminta input dari pengguna untuk bilangan dasar dan pangkat, serta angka untuk faktorial, kemudian menampilkan hasil perhitungan pangkat dan faktorial tersebut. Program ini memberikan contoh penggunaan perulangan (looping) untuk operasi matematika dasar dalam bahasa pemrograman Go.

Guided 2

Code:

```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
    // 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 C:\Users\Faris\Documents\ALGORITMA PEMROGRAMAN 2\103112400079_MODUL 5> go run
Masukkan bilangan: 6
Masukkan bilangan: 6
Masukkan pangkat: 2
6^2 = 36
Masukkan angka untuk faktorial: 6
6! = 720
```

Deskripsi:

Kode di atas adalah program Go yang menggunakan pendekatan rekursif untuk menghitung perpangkatan (base^exp) dan faktorial (n!). Fungsi pangkatRekursif menghitung pangkat dengan memanggil dirinya sendiri secara berulang, mengurangi nilai exp hingga mencapai 0, di mana pangkat apapun dari 0 akan menghasilkan 1. Fungsi faktorialRekursif bekerja serupa, dengan mengurangi nilai n hingga mencapai 1 atau 0, yang keduanya menghasilkan 1, dan kemudian mengalikan nilai tersebut secara rekursif. Di dalam fungsi main, program meminta input dari pengguna untuk bilangan dasar dan pangkat, serta angka untuk faktorial, kemudian menampilkan hasil perhitungan pangkat dan faktorial tersebut. Perbedaan utama antara kode ini dan kode sebelumnya adalah pada pendekatan yang digunakan untuk menghitung pangkat dan faktorial. Kode pertama menggunakan pendekatan iteratif, di mana perhitungan dilakukan dengan loop, sedangkan kode kedua menggunakan pendekatan rekursif, di mana fungsi memanggil dirinya sendiri untuk menghitung hasil.

III. UNGUIDED

Unguided 1

Code:

```go
103112400079_Unguided1 > Unguided1.go > ...
package main

// Muhammad Faris Rachmadi || 103112400079

import "fmt"

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

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

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

Output:

```
PS C:\Users\Faris\Documents\ALGORITMA PEMROGRAMAN 2\103112400079_MODUL 5> go run "c:\Users\Faris\
10
Deret Fibonacci hingga 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:

Kode di atas adalah program Go untuk menghasilkan deret Fibonacci hingga suku ke-n. Fungsi fibonacci menghitung nilai Fibonacci menggunakan pendekatan rekursif, di mana jika n kurang dari atau sama dengan 1, nilai Fibonacci adalah n itu sendiri, dan jika lebih besar dari 1, nilai Fibonacci dihitung dengan menjumlahkan dua nilai sebelumnya (fibonacci(n-1) dan fibonacci(n-2)). Fungsi main meminta input dari pengguna untuk menentukan jumlah suku yang ingin ditampilkan, kemudian mencetak deret Fibonacci dari suku ke-0 hingga suku ke-n.

Unguided 2

Code:

```go
103112400079_Unguided2 > Unguided2.go > printbintang
package main

// Muhammad Faris Rachmadi || 103112400079

import "fmt"

func printbintang(n int, baris int) {

    if baris > n {
        return
    }

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

func main() {
    var n int

    fmt.Print("Masukkan jumlah baris: ")
    fmt.Scan(&n)

    printbintang(n, 1)
}
```

Output:

```
PS C:\Users\Faris\Documents\ALGORITMA PEMROGRAMAN 2\103112400079_MODUL 5> go run "c:\User
Masukkan jumlah baris: 5
*
**
***
****
*****
```

Deskripsi:

Program Go di atas menghasilkan pola bintang segitiga yang jumlah barisnya ditentukan oleh input pengguna. Fungsi printbintang menggunakan pendekatan rekursif untuk mencetak bintang. Fungsi ini menerima dua parameter, n untuk jumlah baris dan baris untuk nomor baris saat ini. Jika baris lebih besar dari n, fungsi akan berhenti. Pada setiap pemanggilan rekursif, fungsi mencetak sejumlah bintang sesuai dengan nomor baris dan kemudian melanjutkan ke baris berikutnya dengan meningkatkan nilai baris. Fungsi main meminta input dari pengguna untuk menentukan jumlah baris, lalu memanggil fungsi printbintang untuk mencetak pola bintang.

Unguided 3

Code:

```go
103112400079_Unguided3 > Unguided3.go > ...
package main

//Muhammad Faris Rachmadi||103112400079
import "fmt"

func carifaktor(n int, i int) {

    if i > n {
        return
    }

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

    carifaktor(n, i+1)
}

func main() {
    var n int

    fmt.Print("Masukkan angka: ")
    fmt.Scan(&n)

    carifaktor(n, 1)
    fmt.Println()
}
```

Output:

```
PS C:\Users\Faris\Documents\ALGORITMA PEMROGRAMAN 2\103112400079_MODUL 5>
Masukkan angka: 12
1 2 3 4 6 12
```

Deskripsi:

Program Go di atas berfungsi untuk mencari dan mencetak faktor-faktor dari suatu angka yang dimasukkan oleh pengguna. Fungsi carifaktor menggunakan pendekatan rekursif untuk memeriksa setiap angka mulai dari 1 hingga angka n. Jika angka i dapat membagi n tanpa sisa (yaitu n % i == 0), maka i dianggap sebagai faktor dan dicetak. Fungsi ini akan terus memanggil dirinya sendiri dengan meningkatkan nilai i hingga i lebih besar dari n. Fungsi main meminta input angka dari pengguna, lalu memanggil fungsi carifaktor untuk mencari dan mencetak semua faktor dari angka tersebut.

IV. KESIMPULAN

Dari praktikum Modul 5 tentang rekursif, dapat disimpulkan bahwa rekursi adalah teknik dalam pemrograman yang memungkinkan suatu fungsi memanggil dirinya sendiri untuk menyelesaikan permasalahan dengan membagi masalah menjadi sub-masalah yang lebih kecil. Praktikum ini menunjukkan penerapan rekursi dalam berbagai masalah, seperti perhitungan faktorial, perpangkatan, deret Fibonacci, pencarian dalam struktur data, dan pola bintang. Meskipun rekursi menawarkan solusi yang lebih elegan dan intuitif untuk masalah yang bersifat terstruktur, penggunaannya perlu diperhatikan karena risiko stack overflow jika tidak ada base case yang jelas atau jika kedalaman rekursi terlalu dalam. Oleh karena itu, meskipun rekursi memiliki keunggulan dalam hal modularitas dan penyelesaian masalah yang lebih sistematis, optimasi seperti memoization atau penggantian dengan iterasi mungkin diperlukan untuk meningkatkan efisiensi dalam implementasinya

V. REFERENSI

MODUL 6 PRAKTIKUM ALGORITMA PEMROGRAMAN 2 – REKURSIF