LAPORAN

PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 5

REKURSIF

Oleh:

NAMA: NUFAIL ALAUDDIN TSAQIF

NIM: 103112400084

KELAS: IF-12-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

1. Pengertian Teori Rekursif

Rekursif adalah suatu teknik pemrograman di mana suatu fungsi atau prosedur memanggil dirinya sendiri untuk menyelesaikan masalah. Teknik ini digunakan untuk memecahkan masalah yang dapat dibagi menjadi sub-masalah yang lebih kecil dan identik dengan masalah utama. Rekursif sangat berguna untuk menyelesaikan masalah yang memiliki pola yang dapat dipecah menjadi bagian-bagian lebih kecil, seperti dalam perhitungan deret atau pencarian nilai.

2. Komponen dalam Rekursif:

1. Base Case (Kasus Dasar): Ini adalah kondisi yang menghentikan pemanggilan rekursif. Jika tidak ada base case, maka rekursif akan berjalan tanpa henti.
2. Recursive Case (Kasus Rekursif): Ini adalah bagian di mana fungsi atau prosedur memanggil dirinya sendiri untuk menyelesaikan sub-masalah yang lebih kecil.

Sebagai contoh, ketika kita menggunakan rekursif untuk menghitung faktorial dari suatu angka n!n!, base case-nya adalah ketika n=1n = 1 atau n=0n = 0, yang mana akan mengembalikan nilai 1, sedangkan recursive case akan memanggil faktorial untuk nilai n−1n-1 hingga mencapai base case.

3. Contoh Program Rekursif

Contoh 1: Menampilkan bilangan dari n hingga 1. Program ini akan memanggil dirinya sendiri untuk menampilkan angka dari n ke 1, lalu berhenti setelah mencapai base case (n == 1).

Contoh 2: Penjumlahan dari 1 hingga n. Dengan menggunakan rekursif, kita bisa menghitung hasil penjumlahan dengan memanggil fungsi penjumlahan untuk nilai yang lebih kecil hingga mencapai nilai 1.

4. Proses Rekursif

1. Forward Process (Proses Maju): Pemanggilan rekursif dilakukan untuk mengurangi masalah menjadi bagian yang lebih kecil.
2. Backward Process (Proses Mundur): Setelah mencapai base case, fungsi mulai mengembalikan hasilnya satu per satu.

GUIDED 1
SOURCE CODE:

guided3 > guided3.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:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL5> go run "d:\ALGORITMA PROGRAM
Masukkan bilangan: 3
Masukkan pangkat: 3
3^3 = 27
Masukkan angka untuk faktorial: 5
5! = 120
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL5>
```

DEKSRIPSI:

Program ini menghitung dua hal terpisah: pangkat dan faktorial. Pertama, program meminta pengguna untuk memasukkan angka dasar dan eksponen, kemudian menghitung hasil pangkat dengan mengalikan angka dasar sebanyak eksponen kali menggunakan fungsi pangkatIteratif. Kedua, program meminta pengguna memasukkan angka untuk dihitung faktorialnya, dan hasil faktorial dihitung dengan mengalikan angka dari 1 hingga angka yang dimasukkan menggunakan fungsi faktorialIteratif. Hasil perhitungan kedua operasi tersebut kemudian ditampilkan di layar.

GUIDED 2
SOURCE CODE:

guided4 > guided4.go > ...

```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:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL5> go run "d:\ALGORITMA PRO
Masukkan bilangan: 3
Masukkan pangkat: 3
3^3 = 27
Masukkan angka untuk faktorial: 5
5! = 120
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL5>
```

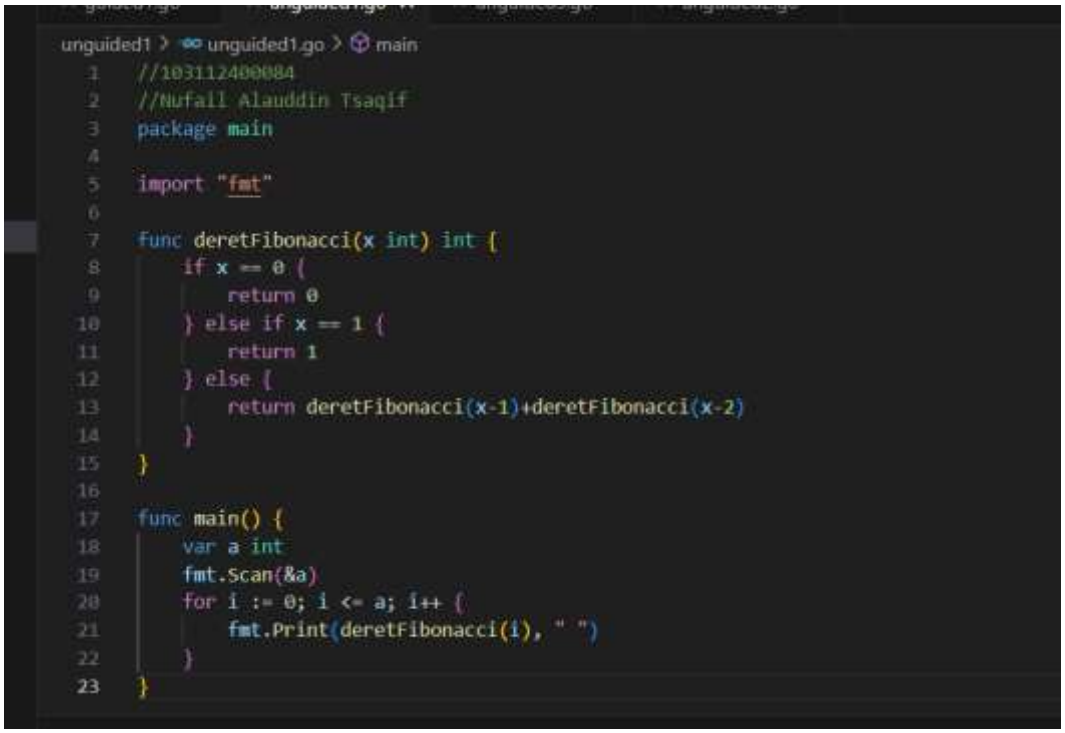
DEKSRIPSI:

Program ini menghitung pangkat dan faktorial menggunakan metode rekursif. Pengguna diminta untuk memasukkan angka dasar dan eksponen, lalu fungsi pangkatRekursif menghitung hasil pangkat dengan memanggil dirinya sendiri hingga eksponen mencapai 0. Program juga meminta pengguna memasukkan angka untuk dihitung faktorialnya, di mana fungsi faktorialRekursif menghitung faktorial dengan memanggil dirinya sendiri hingga mencapai angka 1. Hasil perhitungan pangkat dan faktorial kemudian ditampilkan di layar.

II. UNGUIDED

UNGUIDED 1

SOURCE CODE



```go
//103112400084
//Nufail Alauddin Tsaqif
package main

import "fmt"

func deretFibonacci(x int) int {
    if x == 0 {
        return 0
    } else if x == 1 {
        return 1
    } else {
        return deretFibonacci(x-1)+deretFibonacci(x-2)
    }
}

func main() {
    var a int
    fmt.Scan(&a)
    for i := 0; i <= a; i++ {
        fmt.Print(deretFibonacci(i), " ")
    }
}
```

OUTPUT

```
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL5> go run "d:\ALGORITMA PROGRAMING\PRAK
10
0 1 1 2 3 5 8 13 21 34 55
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL5>
```

DEKSRIPSI

Program ini berfungsi untuk mencetak deret Fibonacci hingga jumlah yang dimasukkan oleh pengguna. Fungsi deretFibonacci(x int) adalah fungsi rekursif yang menghitung nilai Fibonacci berdasarkan rumus: jika x sama dengan 0, maka hasilnya 0; jika x sama dengan 1, hasilnya 1; dan untuk nilai lainnya, hasilnya adalah jumlah dari dua angka sebelumnya dalam deret Fibonacci. Di dalam fungsi main(), program meminta pengguna untuk memasukkan angka a, yang menunjukkan jumlah elemen Fibonacci yang ingin ditampilkan. Program kemudian mencetak deret Fibonacci mulai dari angka 0 hingga a, memanggil fungsi deretFibonacci(i) untuk setiap nilai i dari 0 sampai a. Sebagai contoh, jika input adalah 10, outputnya adalah 0 1 1 2 3 5 8 13 21 34 55.

UNGUIDED 2

SOURCE CODE

unguided3 > unguided3.go > Segitiga

```go
//103112400084
//Nufail Alauddin Tsaqif
package main

import "fmt"

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

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

OUTPUT

```
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL5> go run "d:\ALGORITMA PROGRAMIN
5
*
**
***
****
*****
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL5>
```

DEKSRIPSI

Program ini digunakan untuk mencetak pola segitiga bintang berdasarkan input yang diberikan oleh pengguna. Fungsi Segitiga(n, i int) berfungsi untuk mencetak bintang sebanyak i pada setiap baris, dimulai dari 1 bintang pada baris pertama dan bertambah 1 bintang pada setiap baris berikutnya hingga mencapai jumlah yang diminta. Di dalam fungsi main(), program meminta pengguna untuk memasukkan angka n, yang menentukan jumlah baris segitiga yang akan dicetak. Kemudian, program memanggil fungsi Segitiga(n, 1)

UNGUIDED 3

SOURCE CODE

unguided2 > unguided2.go > main

```go
//103112400084
//Nufail Alauddin Tsaqif

package main

import "fmt"

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

func main() {
    var a int
    fmt.Scan(&a)
    faktorBilangan(a, 1)
}
```

OUTPUT

```
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL5> go run "d:\ALGORITM
5
1 5
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL5> go run "d:\ALGORITM
12
1 2 3 4 6 12
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL5>
```

DEKSRIPSI

Program ini berfungsi untuk mencetak semua faktor bilangan dari angka yang dimasukkan oleh pengguna. Fungsi faktorBilangan(x, i int) akan mencetak angka yang dapat membagi habis angka x mulai dari angka 1 hingga angka yang dimasukkan. Program ini menggunakan rekursi, dimana fungsi faktorBilangan akan dipanggil kembali dengan parameter i+1 sampai i lebih besar dari x. Di dalam fungsi main(), pengguna diminta untuk memasukkan sebuah angka, dan fungsi faktorBilangan akan mencetak faktor-faktor dari angka tersebut. Sebagai contoh, jika input adalah 12, maka output yang dihasilkan adalah faktor-faktor dari 12: 1 2 3 4 6 12**.**

III. KESIMPULAN

Kesimpulan dari laporan praktikum ini adalah bahwa rekursif merupakan salah satu teknik pemrograman yang efisien dan efektif dalam menyelesaikan masalah yang memiliki struktur berulang atau dapat dibagi menjadi sub-masalah yang lebih kecil. Dalam praktikum ini, berbagai contoh penerapan rekursi telah dijelaskan dengan rinci, seperti dalam program untuk mencetak deret Fibonacci, membuat pola segitiga dengan bintang, dan mencari faktor suatu bilangan. Teknik rekursif ini bekerja dengan cara memanggil fungsi itu sendiri, yang akan terus berulang hingga mencapai kondisi dasar (base case), di mana proses rekursi akan berhenti. Dalam program Fibonacci, rekursi digunakan untuk menghitung angka-angka dalam deret dengan cara menambahkan dua angka sebelumnya. Pada program segitiga bintang, rekursi digunakan untuk mencetak baris bintang secara berulang berdasarkan nilai yang diberikan. Sementara itu, pada program faktor bilangan, rekursi membantu dalam mencari semua faktor dari sebuah bilangan dengan memeriksa pembagi-pembagi yang mungkin. Semua contoh ini menunjukkan bahwa rekursi tidak hanya mempermudah penulisan kode tetapi juga dapat menghasilkan solusi yang lebih sederhana dan lebih elegan dalam menyelesaikan masalah tertentu. Penggunaan teknik ini dalam pemrograman sangat berguna terutama dalam permasalahan yang melibatkan proses berulang atau pemecahan masalah secara bertahap.

REFERENSI

MODUL 5 REKURSIF ALGORITMA PEMOGRAMAN 2