LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 5
REKURSIF

Oleh:

DWI OKTA SURYANINGRUM

103112400066

12-IF-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

Apa itu fungsi rekursif? Fungsi rekursif adalah fungsi yang memanggil atau mengeksekusi dirinya sendiri. Fungsi ini sering digunakan untuk melakukan perulangan. Dalam pengembangan aplikasi, ada kalanya lebih mudah menggunakan fungsi rekursif untuk menyelesaikan suatu masalah. Sebagai contoh, operasi faktorial adalah salah satu contoh sederhana penggunaan fungsi rekursif.

Saat membuat fungsi rekursif, kita harus memastikan bahwa fungsi tersebut memiliki kondisi berhenti. Jika tidak, fungsi akan terus memanggil dirinya sendiri dan berisiko menyebabkan error stack overflow, yaitu melebihi batas stack. Oleh karena itu, fungsi rekursif biasanya akan memanggil dirinya hanya jika memenuhi kondisi tertentu.

Kelebihan:

- Kode lebih mudah ditulis.
- Mengurangi pemanggilan fungsi yang tidak perlu.
- Sangat bermanfaat untuk solusi masalah yang berulang.
- Membantu mengurangi panjang kode.
- Sangat berguna untuk menyelesaikan masalah yang berkaitan dengan struktur data.

Kekurangan:

- Fungsi rekursif cenderung lebih lambat dibandingkan fungsi non-rekursif.
- Terkadang membutuhkan lebih banyak memori untuk menyimpan hasil sementara dalam stack.
- Kadang-kadang kode bisa sulit untuk dipahami atau dianalisis.
- Tidak terlalu efisien dalam hal kompleksitas waktu.
- Dapat menyebabkan kehabisan memori jika kondisi berhenti tidak dikelola dengan benar.

II. GUIDED

Source Code + Screenshot hasil program beserta penjelasan

a. Guided 1

```go
// DWI OKTA SURYANINGRUM
package main

import "fmt"

// FUNGSI ITERATIF UNTUK MENGHITUNG PANGKAT (BASE^EXP)
// Fungsi ini digunakan untuk menghitung hasil dari base yang dipangkatkan dengan exp secara iteratif (dengan perulangan).
// Mulai dengan hasil = 1, kemudian kalikan hasil dengan base sebanyak exp kali.
func pangkatIteratif(base, exp int) int {
    // Inisialisasi hasil dengan 1
    hasil := 1
    // Perulangan sebanyak exp kali
    for i := 0; i < exp; i++ {
        // Kalikan hasil dengan base setiap kali perulangan
        hasil *= base
    }
    // Kembalikan hasil akhir setelah perulangan selesai
    return hasil
}

// FUNGSI ITERATIF UNTUK MENGHITUNG FAKTORIAL
// Fungsi ini menghitung faktorial dari angka n secara iteratif.
// Mulai dengan hasil = 1, kemudian kalikan hasil dengan angka dari 2 hingga n.
func faktorialIteratif(n int) int {
    // Inisialisasi hasil dengan 1
    hasil := 1
    // Perulangan dari 2 hingga n
    for i := 2; i <= n; i++ {
        // Kalikan hasil dengan i setiap kali perulangan
        hasil *= i
    }
    // Kembalikan hasil faktorial setelah perulangan selesai
    return hasil
}

func main() {
    // Mendeklarasikan variabel untuk menampung input base, exp, dan n
    var base, exp, n int

    // INPUT UNTUK MENGHITUNG PANGKAT
    // Meminta pengguna untuk memasukkan angka 'base' (bilangan dasar) dan 'exp' (pangkat)
    fmt.Print("Masukkan Bilangan : ")
    fmt.Scanln(&base)  // Membaca input untuk base
    fmt.Print("Masukkan Pangkat : ")
    fmt.Scanln(&exp)   // Membaca input untuk exp (pangkat)

    // Menampilkan hasil pangkat (base^exp) dengan memanggil fungsi 'pangkatIteratif'
    fmt.Printf("%d^%d = %d\n", base, exp, pangkatIteratif(base, exp))

    // INPUT UNTUK MENGHITUNG FAKTORIAL
    // Meminta pengguna untuk memasukkan angka 'n' untuk dihitung faktorialnya
    fmt.Print("Masukkan Angka Untuk Faktorial : ")
    fmt.Scanln(&n) // Membaca input untuk 'n'

    // Menampilkan hasil faktorial dari angka 'n' dengan memanggil fungsi 'faktorialIteratif'
    fmt.Printf("%d! = %d\n", n, faktorialIteratif(n))
}
```

Output :

```
mymac@192 ALPRO SMT 2 % go run "/Users/mymac/Documents/ITTP/ALPRO SMT 2/103112400066_MODUL5/GuidedIteratif1.go"
Masukkan Bilangan : 3
Masukkan Pangkat : 2
3^2 = 9
Masukkan Angka Untuk Faktorial : 9
9! = 362880
```

b. Guided 2

```go
//  DWI OKTA SURYANINGRUM
package main

import "fmt"

// FUNGSI REKURSIF UNTUK MENGHITUNG PANGKAT (BASE^EXP)
func pangkatRekursif(base, exp int) int {
    // Jika exp (pangkat) adalah 0, kembalikan 1, Karena setiap bilangan yang dipangkatkan 0 hasilnya adalah 1
    if exp == 0 {
        return 1
    }
    // Mengembalikan base dikali dengan pangkatRekursif(base, exp-1)
    // Ini akan terus mengurangi exp hingga mencapai 0
    return base * pangkatRekursif(base, exp-1)
}

// FUNGSI REKURSIF UNTUK MENGHITUNG FAKTORIAL
func faktorialRekursif(n int) int {
    // Jika n adalah 0 atau 1, kembalikan 1, Karena faktorial dari 0 dan 1 adalah 1
    if n == 0 || n == 1 {
        return 1
    }
    // mengembalikan n dikali dengan faktorialRekursif(n-1)
    // Ini akan terus mengurangi n hingga mencapai 0 atau 1
    return n * faktorialRekursif(n-1)
}

func main() {
    // Variabel untuk menyimpan input dari pengguna
    var base, exp, n int

    // INPUT PANGKAT
    // Minta pengguna memasukkan bilangan (base) dan pangkat (exp)
    fmt.Print("Masukkan Bilangan : ")
    fmt.Scanln(&base)
    fmt.Print("Masukkan Pangkat : ")
    fmt.Scanln(&exp)

    // Cetak hasil perhitungan pangkat menggunakan fungsi pangkatRekursif
    fmt.Printf("%d^%d = %d\n", base, exp, pangkatRekursif(base, exp))

    // INPUT FAKTORIAL
    // Minta pengguna memasukkan angka untuk menghitung faktorial
    fmt.Print("Masukkan Angka Untuk Faktorial : ")
    fmt.Scanln(&n)

    // Cetak hasil perhitungan faktorial menggunakan fungsi faktorialRekursif
    fmt.Printf("%d! = %d\n", n, faktorialRekursif(n))
}
```

Output :

```
mymac@192 ALPRO SMT 2 % go run "/Users/mymac/Documents/ITTP/ALPRO SMT 2/103112400066_MODUL5/GuidedRekursif2.go"
Masukkan Bilangan : 3
Masukkan Pangkat : 2
3^2 = 9
Masukkan Angka Untuk Faktorial : 9
9! = 362880
```

III. UNGUIDED

a. Unguided 1

```go
// DWI OKTA SURYANINGRUM
package main

import "fmt"

// Fungsi rekursif untuk menghitung nilai deret Fibonacci ke-n
func fibonacci(n int) int {
    // Jika n adalah 0 atau 1, kembalikan nilai n
    if n == 0 || n == 1 {
        return n
    }
    // Mengembalikan penjumlahan dua suku sebelumnya
    return fibonacci(n-1) + fibonacci(n-2)
}

func main() {
    var n int

    // Meminta pengguna memasukkan nilai n
    fmt.Print("Masukkan nilai n (suku ke-n): ")
    fmt.Scan(&n)

    // Cetak nilai deret Fibonacci dari suku ke-0 hingga suku ke-n
    fmt.Println("Deret Fibonacci hingga suku ke-", n, ":")
    for i := 0; i <= n; i++ {
        fmt.Println(fibonacci(i))
    }
}
```

Output :

```
mymac@192 ALPRO SMT 2 % go run "/Users/mymac/Documents/ITTP/ALPRO SMT 2/103112400066_MODUL5/Unguided1.go"
Masukkan nilai n (suku ke-n): 10
Deret Fibonacci hingga suku ke- 10 :
0
1
1
2
3
5
8
13
21
34
55
mymac@192 ALPRO SMT 2 %
```

b. Unguided 2

```go
//DWI OKTA SURYANINGRUM
package main

import "fmt"

// Fungsi rekursif untuk mencetak pola bintang
func Bintang(n, current int) {
    // Basis rekursif: jika current > n, hentikan rekursi
    if current > n {
        return
    }

    // Cetak bintang sebanyak current
    for i := 0; i < current; i++ {
        fmt.Print("*")
    }
    fmt.Println() // Pindah ke baris baru setelah mencetak bintang

    // Rekursif: panggil fungsi Bintang untuk baris berikutnya
    Bintang(n, current+1)
}

func main() {
    var n int

    // Minta pengguna memasukkan nilai n
    fmt.Print("Masukkan nilai N: ")
    fmt.Scan(&n)

    // Panggil fungsi rekursif untuk mencetak pola bintang
    Bintang(n, 1)
}
```

Output :

```
mymac@192 ALPRO SMT 2 % go run "/Users/mymac/Documents/ITTP/ALPRO SMT 2/103112400066_MODUL5/Unguided2.go"
Masukkan nilai N: 5
*
**
***
****
*****
mymac@192 ALPRO SMT 2 % go run "/Users/mymac/Documents/ITTP/ALPRO SMT 2/103112400066_MODUL5/Unguided2.go"
Masukkan nilai N: 1
*
mymac@192 ALPRO SMT 2 % go run "/Users/mymac/Documents/ITTP/ALPRO SMT 2/103112400066_MODUL5/Unguided2.go"
Masukkan nilai N: 3
*
**
***
```

c. Unguided 3

```go
// DWI OKTA SURYANINGRU,
package main

import "fmt"

// Fungsi rekursif untuk mencetak faktor bilangan dari N
func faktor(n, current int) {
    // Basis rekursif: jika current > n, hentikan rekursi
    if current > n {
        return
    }

    // Jika current adalah faktor dari n, cetak current
    if n%current == 0 {
        fmt.Print(current, " ")
    }

    // Rekursif: panggil fungsi faktor untuk nilai current+1
    faktor(n, current+1)
}

func main() {
    var n int

    // Minta pengguna memasukkan nilai n
    fmt.Scan(&n)

    // Panggil fungsi rekursif untuk mencetak faktor bilangan
    faktor(n, 1)
    fmt.Println() // Pindah ke baris baru setelah mencetak faktor
}
```

Output :

```
mymac@192 ALPRO SMT 2 % go run "/Users/mymac/Documents/ITTP/ALPRO SMT 2/103112400066_MODUL5/Unguided3.go"
5
1 5
mymac@192 ALPRO SMT 2 % go run "/Users/mymac/Documents/ITTP/ALPRO SMT 2/103112400066_MODUL5/Unguided3.go"
12
1 2 3 4 6 12
```

IV. KESIMPULAN

Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri untuk menyelesaikan masalah, sering digunakan dalam kasus yang memerlukan perulangan atau solusi berulang. Kelebihannya termasuk kemudahan penulisan kode, pengurangan panjang kode, dan penerapan solusi yang efisien pada struktur data. Namun, kekurangannya meliputi kinerja yang lebih lambat, penggunaan memori yang lebih besar, dan potensi kesulitan dalam memahami atau menganalisis kode. Agar efektif, fungsi rekursif perlu memiliki kondisi berhenti yang jelas untuk menghindari masalah seperti stack overflow.

V. REFERENSI

#27: Recursive Function - Belajar Golang Dari Dasar. (2022). Retrieved from https://blog.ruangdeveloper.com/golang-recursive-function/