LAPORAN PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN 2

MODUL 5

“REKURSIF”

DISUSUN OLEH:

RAIHAN ADI ARBA

103112400071

S1 IF-12-01

DOSEN:

Dimas Fanny Hebrasianto Permadi, S.ST., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024/2025

DASAR TEORI

Rekursif dalam bahasa pemrograman Golang merupakan teknik pemrograman di mana sebuah fungsi memanggil dirinya sendiri untuk menyelesaikan suatu permasalahan. Rekursif diimplementasikan dengan mendefinisikan fungsi yang di dalam prosesnya akan memanggil kembali nama fungsi tersebut, dengan parameter yang berbeda hingga mencapai kondisi berhenti yang disebut dengan base case atau kasus dasar. Konsep rekursif sangat penting dalam pemrograman karena memungkinkan penyelesaian masalah kompleks dengan pendekatan "divide and conquer", yaitu membagi masalah menjadi sub-masalah yang lebih kecil dan identik, sehingga lebih mudah diselesaikan. Dengan pendekatan ini, programmer dapat menyelesaikan masalah yang sulit diimplementasikan dengan pendekatan iteratif biasa, serta membuat kode menjadi lebih ringkas dan elegan.

Dalam implementasinya, rekursif dalam Golang memiliki dua komponen utama: kasus dasar (base case) dan kasus rekursif (recursive case). Kasus dasar berfungsi sebagai kondisi pemberhentian yang mencegah fungsi memanggil dirinya sendiri secara tak terbatas, sementara kasus rekursif adalah bagian yang berisi pemanggilan fungsi itu sendiri. Penggunaan rekursif dalam Golang harus memperhatikan pengelolaan memori, karena setiap pemanggilan fungsi akan menambah entri baru pada stack memory. Untuk mengoptimalkan penggunaan rekursif, Golang mendukung konsep tail recursion, yaitu teknik di mana pemanggilan rekursif terjadi di akhir fungsi sebagai operasi terakhir, yang memungkinkan compiler untuk mengoptimalkan penggunaan memori.

Penggunaan rekursif memiliki berbagai keuntungan, seperti menghasilkan kode yang lebih bersih dan mudah dipahami untuk algoritma tertentu seperti tree traversal, pencarian mendalam (depth-first search), dan perhitungan faktorial. Rekursif juga sangat efektif dalam menyelesaikan masalah yang dapat dibagi menjadi sub-masalah identik, seperti algoritma divide and conquer. Dalam skala besar, pemahaman rekursif membantu programmer dalam menyelesaikan permasalahan yang kompleks dengan pendekatan yang lebih terstruktur dan elegatif. Namun, perlu diperhatikan bahwa penggunaan rekursif yang tidak tepat dapat mengakibatkan stack overflow jika terlalu banyak pemanggilan fungsi yang tertumpuk tanpa mencapai kasus dasar. Dengan memahami dan mengimplementasikan rekursif secara efektif, seorang programmer Golang dapat meningkatkan kualitas algoritmanya serta mengembangkan solusi yang lebih optimal untuk permasalahan yang membutuhkan pendekatan rekursif.

A. GUIDED

1. Source code :

```
103112400071_MODUL5 > 103112400071_guided1.go > faktorialIteratif
1   package main
2
3   import "fmt"
4
5   // Fungsi iteratif untuk menghitung pangkat (base^exp)
6   func pangkatIteratif(base, exp int) int {
7       hasil := 1
8
9       for i := 0; i < exp; i++ {
10          hasil *= base
11
12      }
13
14      return hasil
15  }
16
17  // Fungsi iteratif untuk menghitung faktorial (n!)
18  func faktorialIteratif(n int) int {
19      hasil := 1
20      for i := 2; i <= n; i++ {
21          hasil *= i
22      }
23      return hasil
24  }
25
26  func main() {
27      var base, exp, n int
28
29      // Input pangkat
30      fmt.Print("Masukkan bilangan: ")
31      fmt.Scanln(&base)
32      fmt.Print("Masukkan pangkat: ")
33      fmt.Scanln(&exp)
34
35      fmt.Printf("%d^%d = %d\n", base, exp, pangkatIteratif(base, exp))
36
37      // Input Faktorial
38      fmt.Print("Masukkan angka untuk faktorial: ")
39      fmt.Scanln(&n)
40
41      fmt.Printf("%d! = %d\n", n, faktorialIteratif(n))
42  }
43
```

Output :

```
PS D:\raihan\103112400071_MODUL5> go run 103112400071_guided1.go
Masukkan bilangan: 10
Masukkan pangkat: 3
10^3 = 1000
Masukkan angka untuk faktorial: 4
4! = 24
```

Deskripsi :

Program terdiri dari dua fungsi iteratif utama dan fungsi main untuk interaksi dengan pengguna. Fungsi pangkatIteratif bertugas menghitung nilai dari suatu bilangan yang dipangkatkan, dengan parameter base sebagai bilangan dasar dan exp sebagai eksponen. Fungsi ini menggunakan perulangan for untuk mengalikan nilai base

sebanyak exp kali, menghasilkan nilai pangkat yang diinginkan. Berbeda dengan pendekatan rekursif, metode ini memanfaatkan variabel hasil yang diinisialisasi dengan nilai 1, kemudian diperbarui pada setiap iterasi. Fungsi kedua, faktorialIteratif, digunakan untuk menghitung nilai faktorial dari suatu bilangan menggunakan perulangan. Fungsi ini juga menggunakan variabel hasil yang diinisialisasi dengan nilai 1, kemudian mengalikan hasil tersebut dengan semua bilangan dari 2 hingga n secara berurutan. Dalam fungsi main, program meminta pengguna untuk memasukkan nilai dasar dan eksponen untuk perhitungan pangkat, kemudian menampilkan hasilnya. Selanjutnya, program meminta pengguna untuk memasukkan sebuah angka untuk perhitungan faktorial dan menampilkan hasilnya. Program ini menggunakan pendekatan iteratif yang lebih efisien dalam penggunaan memori dibandingkan dengan pendekatan rekursif, terutama untuk nilai input yang besar.

2. Source code :

103112400071_MODUL5 > 103112400071_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:\raihan\103112400071_MODUL5> go run 103112400071_guided2.go
Masukkan bilangan: 10
Masukkan pangkat: 3
10^3 = 1000
Masukkan angka untuk faktorial: 4
4! = 24
PS D:\raihan\103112400071_MODUL5>
```

Deskripsi :

Program terdiri dari dua fungsi rekursif utama dan fungsi main. Fungsi pangkatRekursif bertugas menghitung nilai dari suatu bilangan yang dipangkatkan, dengan parameter base sebagai bilangan dasar dan exp sebagai eksponen. Fungsi ini memiliki kondisi dasar yaitu jika nilai eksponen adalah 0, maka akan mengembalikan nilai 1. Jika tidak, fungsi akan mengembalikan hasil perkalian base dengan hasil pemanggilan rekursif fungsi tersebut dengan eksponen yang dikurangi 1. Fungsi kedua, faktorialRekursif, digunakan untuk menghitung nilai faktorial dari suatu bilangan. Fungsi ini memiliki kondisi dasar yaitu jika nilai n adalah 0 atau 1, maka akan mengembalikan nilai 1. Jika tidak, fungsi akan mengembalikan hasil perkalian n dengan hasil pemanggilan rekursif fungsi tersebut dengan parameter n yang dikurangi 1. Dalam fungsi main, program meminta pengguna untuk memasukkan nilai dasar dan eksponen untuk perhitungan pangkat, kemudian menampilkan hasilnya. Selanjutnya, program meminta pengguna untuk memasukkan sebuah angka untuk perhitungan faktorial dan menampilkan hasilnya. Program ini menggunakan pendekatan rekursi yang berbeda dari pengulangan konvensional seperti for dan while, namun mencapai hasil yang sama dengan cara yang lebih elegan untuk perhitungan matematis semacam ini.

B. UNGUIDED

1. Latihan 1

Source Code:

103112400071_MODUL5 > 103112400071_unguided1.go > ...

```go
// Raihan Adi Arba
// 103112400071
package main

import "fmt"

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

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

    // Cetak deret Fibonacci
    for i := 0; i <= n; i++ {
        fmt.Print(deretFibonacci(i))
        if i < n {
            fmt.Print(" ")
        }
    }
    fmt.Println()
}

```

Output:

```
PS D:\raihan> go run 103112400071_MODUL5\103112400071_unguided1.go
10
0 1 1 2 3 5 8 13 21 34 55
PS D:\raihan> go run 103112400071_MODUL5\103112400071_unguided1.go
5
0 1 1 2 3 5
PS D:\raihan>
```

Penjelasan Program :

Program unguided1.go dimulai dengan mendefinisikan fungsi deretFibonacci(x int) yang akan menghitung nilai Fibonacci pada indeks x dengan pendekatan rekursif. Fungsi ini memiliki dua kasus dasar: jika x adalah 0, maka fungsi akan mengembalikan

nilai 0; jika x adalah 1, maka fungsi akan mengembalikan nilai 1. Untuk nilai x yang lebih besar dari 1, fungsi akan memanggil dirinya sendiri dengan rumus Fibonacci klasik yaitu F(n) = F(n-1) + F(n-2).
Di dalam fungsi main(), program menyediakan variabel n untuk menyimpan input dari pengguna, yang diambil menggunakan fmt.Scan(&n). Setelah menerima input, program menggunakan perulangan for untuk mencetak setiap nilai dalam deret Fibonacci dari indeks 0 hingga n. Setiap nilai dicetak dengan memanggil fungsi deretFibonacci(i) dan untuk memisahkan antar nilai, program menambahkan spasi kecuali untuk nilai terakhir. Di akhir, program mencetak baris baru dengan fmt.Println().

2. Latihan 2

Source Code:

103112400071_MODUL5 > 103112400071_unguided2.go > ...

```go
// Raihan Adi Arba
// 103112400071
package main

import "fmt"

func cetakBintang(n int) {
    if n <= 0 {
        return
    }
    fmt.Print("*")
    cetakBintang(n - 1)
}

func buatSegitiga(x int) {
    if x == 0 {
        return
    }
    buatSegitiga(x - 1)
    cetakBintang(x)
    fmt.Println()
}

func main() {
    var a int
    fmt.Scan(&a)
    buatSegitiga(a)
}
```

Output:

```
PS D:\raihan> go run 103112400071_MODUL5\103112400071_unguided2.go
5
*
**
***
****
*****
PS D:\raihan>
```

Deskripsi Program:

Program unguided2 mengimplementasikan pola segitiga bintang menggunakan rekursi. Cara kerjanya cukup sederhana namun efektif, dimana terdapat dua fungsi rekursif yang saling bekerja sama. Fungsi pertama, cetakBintang, bertugas mencetak karakter bintang secara horizontal sebanyak parameter yang diberikan, sedangkan fungsi kedua, buatSegitiga, mengatur pembentukan pola segitiga dengan memanggil fungsi cetakBintang untuk setiap barisnya. Program dimulai dengan meminta masukan angka dari pengguna yang menentukan tinggi segitiga, kemudian menghasilkan tampilan segitiga dengan jumlah baris sesuai input tersebut. Setiap baris memiliki jumlah bintang yang bertambah secara progresif, dimulai dari satu bintang hingga sejumlah nilai input. Keunikan dari program ini terletak pada penggunaan pendekatan rekursif untuk menghasilkan pola, bukan menggunakan perulangan biasa seperti for atau while yang umum digunakan

3. Latihan 3

Source Code:

103112400071_MODUL5 > 103112400071_unguided3.go > ...

```go
// Raihan Adi Arba
// 103112400071
package main

import "fmt"

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

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

```

Output:

```
PS D:\raihan> go run 103112400071_MODUL5\103112400071_unguided3.go
5
1 5
PS D:\raihan> go run 103112400071_MODUL5\103112400071_unguided3.go
12
1 2 3 4 6 12
PS D:\raihan>
```

Deskripsi Program:

Program unguided 3 bekerja dengan memanggil fungsi faktorBilangan yang secara rekursif memeriksa setiap bilangan dari 1 hingga nilai input untuk menentukan mana yang merupakan faktor dari bilangan tersebut. Fungsi ini menerima dua parameter: x sebagai bilangan yang akan dicari faktornya dan i sebagai iterator yang dimulai dari 1. Pada setiap pemanggilan fungsi, program memeriksa apakah nilai i adalah faktor dari x dengan menggunakan operasi modulo (x%i == 0). Jika benar, maka i akan dicetak sebagai salah satu faktor. Kemudian fungsi memanggil dirinya sendiri dengan menambahkan nilai i sebanyak 1 untuk memeriksa bilangan berikutnya. Proses rekursi akan berakhir ketika nilai i melebihi nilai x. Dalam fungsi main, program meminta pengguna memasukkan sebuah bilangan bulat N melalui fmt.Scan, kemudian memanggil fungsi faktorBilangan dengan parameter N dan 1 untuk mulai mencari dan menampilkan semua faktor dari bilangan N tersebut..

DAFTAR PUSTAKA

Prayogo, N. A. (2021). ***Dasar Pemrograman Go. Ebook***