LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 5

REKURSIF

Oleh:

NAUFAL THORIQ MUZHAFFAR

2311102078

IF-11-02

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Rekursi adalah teknik pemrograman di mana sebuah fungsi memanggil dirinya sendiri. Pada dasarnya, rekursi melibatkan dua bagian utama: kasus dasar (base case) dan panggilan rekursif (recursive call). Kasus dasar menentukan kondisi di mana fungsi berhenti memanggil dirinya sendiri, sedangkan panggilan rekursif adalah bagian di mana fungsi memanggil dirinya sendiri dengan parameter yang berbeda.

Struktur Rekursi:

- Base Case: Kasus dasar adalah kondisi yang mengakhiri rekursi. Tanpa ini, fungsi rekursif akan terus memanggil dirinya sendiri tanpa henti dan mengakibatkan kesalahan "stack overflow".
- Recursive Call: Pada bagian ini, fungsi memanggil dirinya sendiri dengan parameter yang dimodifikasi. Ini akan mengarah pada pengurangan masalah hingga mencapai kasus dasar.

Berikut adalah contoh sederhana penggunaan rekursi dalam Golang untuk menghitung faktorial dari sebuah bilangan:

```
package main

import "fmt"

func faktorial(n int) int {
    if n == 0 {
        return 1 // Kasus dasar: faktorial 0 adalah 1
    }
    return n * faktorial(n-1) // Panggilan rekursif
}

func main() {
    fmt.Println(faktorial(5)) // Output: 120
}
```

Pada contoh di atas, fungsi faktorial memanggil dirinya sendiri sampai mencapai kasus dasar, di mana n == 0.

Kelebihan dan Kekurangan Rekursi:

Kelebihan:

- Mempermudah penulisan kode untuk masalah yang memiliki pola berulang, seperti algoritma pencarian, traversal struktur data (misalnya pohon dan graf), dan masalah matematika.
- Mengurangi kompleksitas kode untuk beberapa jenis masalah.

Kekurangan:

- Konsumsi memori lebih tinggi dibandingkan dengan iterasi, karena setiap pemanggilan fungsi memerlukan ruang di stack.
- Risiko kesalahan seperti "stack overflow" jika kasus dasar tidak didefinisikan dengan benar atau jika rekursi berjalan terlalu dalam.

Penerapan dalam Pemrograman

Rekursi sering digunakan pada algoritma tertentu seperti pencarian biner, algoritma pembagi dan taklukkan (divide and conquer), dan traversal pohon. Walaupun rekursi sangat berguna, penting untuk memastikan bahwa setiap panggilan rekursif mendekati kasus dasar untuk menghindari masalah seperti infinite loop atau stack overflow.

Rekursi adalah teknik yang kuat, tetapi perlu digunakan dengan bijaksana dalam pengembangan software di Golang. Saat menulis fungsi rekursif, penting untuk memastikan ada kondisi penghentian yang jelas untuk menghindari masalah dalam eksekusi program.

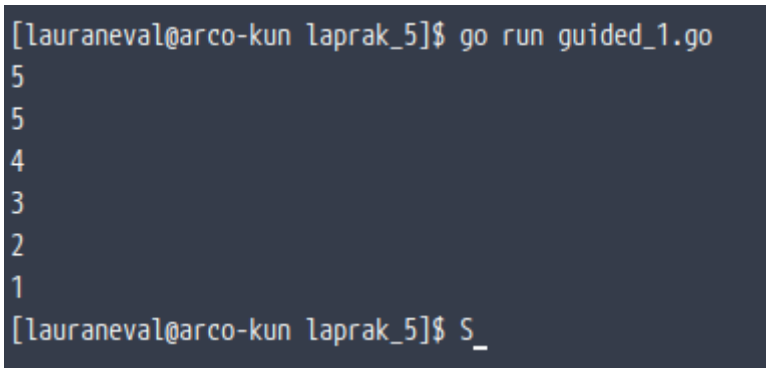
II. GUIDED

GUIDED 1

Source Code

```
package main
import "fmt"

func main(){
    var n int
    fmt.Scan(&n)     //Membaca Input Pengguna
    baris(n)         //Memanggil Fungsi Rekursif 'baris'
}

func baris(bilangan int){
    if bilangan == 1{              // Base case: Jika
bilangan sama dengan 1
        fmt.Println(1)           // Cetak angka 1
    } else {                      // Jika bilangan lebih
besar dari 1
        fmt.Println(bilangan)     // Cetak bilangan saat
ini
        baris(bilangan - 1)     // Panggil fungsi 'baris'
dengan bilngan dikurangi 1
    }
}
```

Screnshoot Output



Penjelasan Program

Program meminta pengguna untuk memasukkan sebuah angka integer n. Angka ini akan dibaca dan disimpan dalam variabel n. Fungsi baris adalah fungsi rekursif yang dipanggil dengan nilai n yang diinput oleh pengguna. Fungsi ini memiliki dua bagian:

- Kasus Dasar (Base Case): Jika nilai bilangan adalah 1, maka program mencetak 1 dan menghentikan rekursi.
- Rekursif Call: Jika bilangan lebih besar dari 1, program mencetak nilai bilangan saat ini, lalu memanggil kembali fungsi baris dengan bilangan - 1.

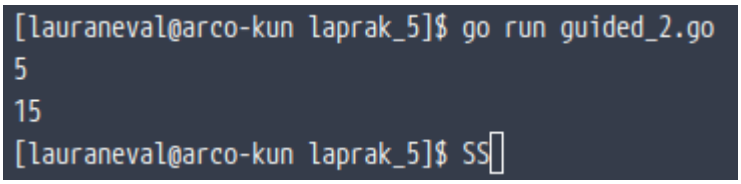
Program akan terus mencetak nilai bilangan, dimulai dari n hingga 1. Setelah mencapai 1, rekursi berhenti.

GUDED 2

Source Code

```
package main

import "fmt"


func main(){

    var n int

    fmt.Scan(&n)

    fmt.Println(penjumlahan(n))

}


func penjumlahan(n int) int{

    if n == 1{

        return 1

    } else {

        return n + penjumlahan(n - 1)

    }

}
```

Screenshoot Output



Penjelasan Program

Program meminta pengguna untuk memasukkan sebuah angka integer n. Angka tersebut kemudian dibaca dan disimpan dalam variabel n.

Fungsi penjumlahan bertujuan untuk menghitung jumlah bilangan dari n hingga 1 menggunakan rekursi.

- Base Case: Jika n sama dengan 1, fungsi akan mengembalikan nilai 1. Ini menjadi titik berhenti rekursi.
- Rekursif Call: Jika n lebih besar dari 1, fungsi akan mengembalikan nilai n ditambah hasil dari penjumlahan(n - 1). Artinya, ia akan terus menambahkan nilai hingga mencapai kasus dasar.

Program akan terus mencetak nilai bilangan, dimulai dari n hingga 1. Setelah mencapai 1, rekursi berhenti.

III. UNGUIDED

UNGUIDED 1

Deret fibonacci adalah sebuah deret dengan nilai suku ke-0 dan ke-1 adalah 0 dan 1, dan nila suku ke-n selanjutnya adalah hasil penjumlahan dua suku sebelumnya. Secara umum dapat diformulasikan $S_n = S_{n-1} + S_{n-2}$. Berikut ini adalah contoh nilai deret fibonacci hingga suku ke-10. Buatlah program yang mengimplementasikan fungsi rekursif pada deret fibonacc tersebut.

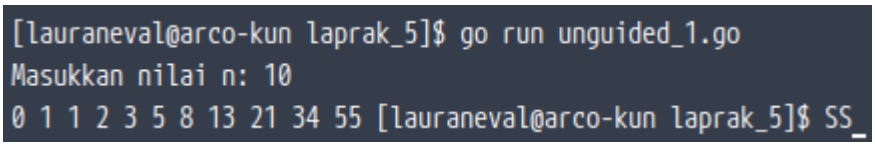
n	0	1	2	3	4	5	6	7	8	9	10
S_n	0	1	1	2	3	5	8	13	21	34	55

Source Code

```
package main

import "fmt"


func main() {

    var n int

    fmt.Print("Masukkan nilai n: ")

    fmt.Scan(&n)

    for i := 0; i <= n; i++ {

        fmt.Print(fibonacci(i), " ")

    }

}


func fibonacci(n int) int {

    if n == 0 {

        return 0

    } else if n == 1 {

        return 1
```

```
    } else {
        return fibonacci(n-1) + fibonacci(n-2)
    }
}
```

Screenshot Output



Deskripsi Program

Program meminta pengguna untuk memasukkan sebuah nilai integer n. Nilai ini menentukan hingga suku ke berapa dari deret Fibonacci yang ingin ditampilkan.

Fungsi Rekursif fibonacci: Fungsi ini digunakan untuk menghitung nilai deret Fibonacci berdasarkan formula: Sn=Sn−1+Sn−2S_n = S_{n-1} + S_{n-2}Sn =Sn−1+Sn−2

- Base Case**:** Jika n adalah 0, fungsi mengembalikan 0. Jika n adalah 1, fungsi mengembalikan 1.
- Rekursif Call**:** Jika n lebih besar dari 1, fungsi memanggil dirinya sendiri dengan fibonacci(n-1) + fibonacci(n-2) untuk menghitung nilai Fibonacci ke-n.

Setelah pengguna memasukkan nilai n, program akan mencetak deret Fibonacci dari 0 hingga n.

UNGUIDED 2

Buatlah sebuah program yang digunakan untuk menampilkan pola bintang berikut ini dengan menggunakan fungsi rekursif. N adalah masukan dari user.

Contoh masukan dan keluaran:

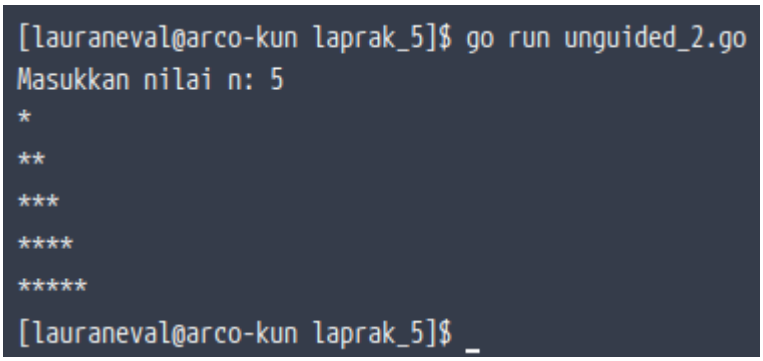
No	Masukan	Keluaran
1	5	* ** *** **** *****
2	1	*
3	3	* ** ***

Source Code

```
package main

import "fmt"


func main() {

    var n int

    fmt.Print("Masukkan nilai n: ")

    fmt.Scan(&n)

    polaBintang(n, 1)

}


func polaBintang(n, current int) {

    if current > n {

        return

    }
```

```
    cetakBintang(current)
    fmt.Println()
    polaBintang(n, current+1)
}

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

Sreenshoot Output



Penjelasan Program

Program meminta pengguna untuk memasukkan sebuah angka n, yang merupakan jumlah baris dari pola bintang yang akan ditampilkan.

Fungsi Rekursif polaBintang:

- Fungsi ini bertugas untuk mencetak bintang sesuai pola yang diinginkan. Pola ini dicetak per baris, dimulai dari 1 bintang hingga n bintang.
- Base Case: Jika nilai current (jumlah baris saat ini) lebih besar dari n, rekursi berhenti.

- Rekursif Call: Program akan mencetak sejumlah bintang pada setiap baris, kemudian memanggil dirinya sendiri dengan current ditambah 1, hingga mencapai n.

Fungsi Rekursif cetakBintang:

- Fungsi ini bertugas mencetak bintang pada satu baris sesuai dengan jumlah yang diinginkan.
- Base Case: Jika jumlah bintang yang akan dicetak (jumlah) adalah 0, rekursi berhenti.
- Rekursif Call: Jika jumlah masih lebih besar dari 0, maka program mencetak satu bintang dan memanggil kembali fungsi cetakBintang dengan jumlah dikurangi 1.

Setelah pengguna memasukkan angka n, program akan menampilkan pola bintang dari baris 1 hingga baris n. Setiap baris akan mencetak bintang sebanyak nomor baris tersebut.

UNGUIDED 3

Buatlah program yang mengimplementasikan rekursif untuk menampilkan faktor bilangan dari suatu N, atau bilangan yang apa saja yang habis membagi N.

Masukan terdiri dari sebuah bilangan bulat positif N.

Keluaran terdiri dari barisan bilangan yang menjadi faktor dari N (terurut dari 1 hingga N ya).

Contoh masukan dan keluaran:

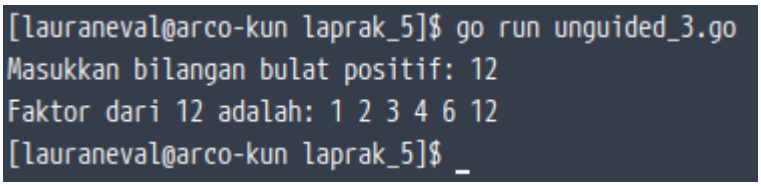
No	Masukan	Keluaran
1	5	1 5
2	12	1 2 3 4 6 12

Source Code

```
package main

import "fmt"


func cetakFaktor(n, i int) {
    if i > n {
        return
    }
    if n%i == 0 {
        fmt.Printf("%d ", i)
    }
    cetakFaktor(n, i+1)
}


func main() {
    var n int
    fmt.Print("Masukkan bilangan bulat positif: ")
    fmt.Scan(&n)
```

```
    fmt.Printf("Faktor dari %d adalah: ", n)
    cetakFaktor(n, 1)
    fmt.Println()
}
```

Screenshoot Output



Penjelasan Program

Program meminta pengguna untuk memasukkan sebuah bilangan bulat positif n. Nilai n ini akan digunakan untuk mencari faktor-faktor dari bilangan tersebut.

Fungsi Rekursif cetakFaktor:

- Fungsi cetakFaktor(n, i) digunakan untuk mencari dan mencetak faktor dari n.
- Fungsi ini memulai pencarian dari i = 1 dan terus bertambah hingga mencapai n.
- Base Case: Jika nilai i lebih besar dari n, rekursi berhenti.
- Pemeriksaan Faktor: Pada setiap langkah, program memeriksa apakah i adalah faktor dari n dengan mengecek apakah n % i == 0. Jika benar, nilai i dicetak sebagai faktor.
- Rekursif Call: Fungsi memanggil dirinya sendiri dengan i + 1, sehingga pencarian faktor dilanjutkan hingga semua faktor ditemukan.

Setelah pengguna memasukkan nilai n, program mencetak pesan yang menunjukkan bahwa faktor dari n akan dicari. Fungsi cetakFaktor akan dipanggil dengan i mulai dari 1, dan setiap kali i adalah faktor dari n, program akan mencetak i. Proses ini akan berulang sampai i mencapai n.

UNGUIDED 4

Buatlah program yang mengimplementasikan rekursif untuk menampilkan barisan bilangan tertentu.

Masukan terdiri dari sebuah bilangan bulat positif N.

Keluaran terdiri dari barisan bilangan dari N hingga 1 dan kembali ke N.

Contoh masukan dan keluaran:

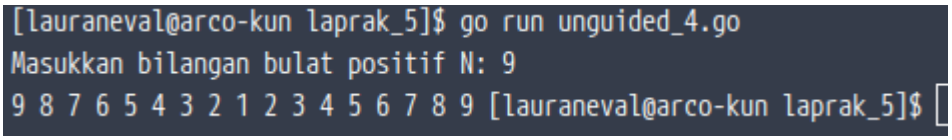
No	Masukan	Keluaran
1	5	5 4 3 2 1 2 3 4 5
2	9	9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9

Source Code

```
package main

import "fmt"


func cetakBarisan(n, current int) {

    if current == 1 {

        fmt.Print(current, " ")

        return

    }

    fmt.Print(current, " ")

    cetakBarisan(n, current-1)

    fmt.Print(current, " ")

}


func main() {

    var n int

    fmt.Print("Masukkan bilangan bulat positif N: ")

    fmt.Scan(&n)

    cetakBarisan(n, n)
```

```
}
```

Sreenchoot Output



Penjelasan Program

Program meminta pengguna untuk memasukkan sebuah bilangan bulat positif n. Nilai n ini akan digunakan untuk menentukan batas dari barisan angka yang akan dicetak.

Fungsi Rekursif cetakBarisan:

- Fungsi cetakBarisan(n, current) bertujuan untuk mencetak angka secara rekursif dengan pola yang unik.
- Base Case: Jika current sama dengan 1, program akan mencetak 1 dan berhenti (mengakhiri rekursi).
- Rekursif Call: Jika current lebih besar dari 1, program mencetak angka current, lalu memanggil fungsi cetakBarisan dengan current-1. Setelah rekursi selesai, angka current dicetak lagi.
- Pola ini menyebabkan angka dari n hingga 1 dicetak ke bawah, dan kemudian dari 1 hingga n kembali ke atas.

Setelah pengguna memasukkan nilai n, program memanggil fungsi cetakBarisan dengan current dimulai dari n. Fungsi akan mencetak angka dari n ke 1, lalu kembali mencetak angka dari 1 hingga n.

UNGUIDED 5

Buatlah program yang mengimplementasikan rekursif untuk menampilkan barisan bilangan ganjil.

Masukan terdiri dari sebuah bilangan bulat positif N.

Keluaran terdiri dari barisan bilangan ganjil dari 1 hingga N.

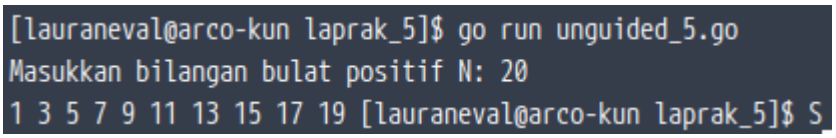
Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	1 3 5
2	20	1 3 5 7 9 11 13 15 17 19

Source Code

```
package main

import "fmt"


func cetakGanjil(n, current int) {

    if current > n {

        return

    }

    fmt.Print(current, " ")

    cetakGanjil(n, current+2)

}


func main() {

    var n int

    fmt.Print("Masukkan bilangan bulat positif N: ")

    fmt.Scan(&n)

    cetakGanjil(n, 1)

}
```

Screenshoot Output



Penjelasan Program

Program meminta pengguna untuk memasukkan sebuah bilangan bulat positif n. Nilai n ini menentukan batas akhir dari deret bilangan ganjil yang akan dicetak.

Fungsi Rekursif cetakGanjil:

- Fungsi cetakGanjil(n, current) bertujuan untuk mencetak bilangan ganjil secara rekursif.
-
- Base Case: Jika current lebih besar dari n, rekursi akan berhenti (mengakhiri proses pencetakan).
- Rekursif Call: Program mencetak nilai current (bilangan ganjil), lalu memanggil fungsi cetakGanjil lagi dengan nilai current + 2. Ini memastikan setiap angka yang dicetak adalah ganjil karena mulai dari 1 dan bertambah 2 setiap kali.

Setelah pengguna memasukkan nilai n, program memanggil fungsi cetakGanjil dengan current dimulai dari 1. Fungsi akan mencetak semua bilangan ganjil dari 1 hingga batas yang kurang atau sama dengan n.

UNGUIDED 6

Buatlah program yang mengimplementasikan rekursif untuk mencari hasil pangkat dari dua buah bilangan.

Masukan terdiri dari bilangan bulat x dan y.

Keluaran terdiri dari hasil x dipangkatkan y.

Catatan: diperbolehkan menggunakan asterik "*", tapi dilarang menggunakan import "math".

Contoh masukan dan keluaran:

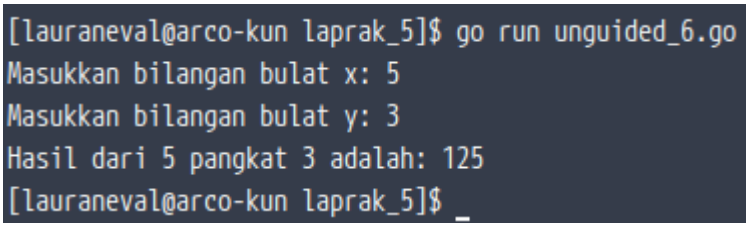
No	Masukan	Keluaran
1	2 2	4
2	5 3	125

Source Code

```
package main

import "fmt"


func power(x, y int) int {

    if y == 0 {

        return 1

    }

    return x * power(x, y-1)

}


func main() {

    var x, y int

    fmt.Print("Masukkan bilangan bulat x: ")

    fmt.Scan(&x)

    fmt.Print("Masukkan bilangan bulat y: ")

    fmt.Scan(&y)
```

```
    result := power(x, y)

    fmt.Printf("Hasil dari %d pangkat %d adalah: %d\n", x, y, result)

}
```

Screenshoot Output



Penjelasan Program

Program meminta pengguna untuk memasukkan dua bilangan bulat: x (basis) dan y (eksponen). Nilai x dan y ini akan digunakan untuk menghitung x pangkat y.

Fungsi Rekursif power:

- Fungsi power(x, y) bertujuan untuk menghitung hasil dari x pangkat y secara rekursif.
- Base Case: Jika y sama dengan 0, fungsi mengembalikan 1, karena setiap bilangan yang dipangkatkan 0 hasilnya adalah 1.
- Rekursif Call: Jika y lebih besar dari 0, fungsi mengalikan x dengan hasil power(x, y-1). Ini berarti fungsi terus memanggil dirinya sendiri dengan mengurangi y hingga mencapai 0.

Setelah pengguna memasukkan nilai x dan y, program memanggil fungsi power untuk menghitung x pangkat y. Hasilnya disimpan dalam variabel result, dan program akan mencetak hasil perpangkatan tersebut.