

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMOGRAMAN 2**

MODUL V

REKURSIF



Oleh:

YASVIN SYAHGANA

2311102065

S1 IF 11 02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

Pendekatan Rekursif merupakan pendekatan dengan fokus penggunaan method dengan instruksi berupa memanggil dirinya sendiri sehingga terciptalah perulangan dan akan berhenti pada kasus dasar tertentu. Kelebihan Fungsi Rekursif adalah program menjadi lebih singkat. Pada beberapa kasus, lebih mudah menggunakan fungsi rekursif, contohnya: pangkat, factorial, dan fibonacci, dan beberapa proses deret lainnya. Fungsi rekursif lebih efisien dan cepat dibandingkan proses secara iteratif. Kekurangan Fungsi Rekursif adalah memakan memori lebih besar, karena setiap bagian dari dirinya dipanggil, akan membutuhkan sejumlah ruang memori untuk penyimpanan. Rekursif sering kali tidak bisa berhenti sehingga memori akan terpakai habis dan program bisa hang. Ada dua hal yang menjadi *concern* kita jika ingin menyelesaikan masalah secara pendekatan rekursif, yaitu :

- **Base Case** : kasus paling sederhana dari suatu permasalahan.
- **Reccurence Relation** : Bagaimana hubungan rekursif dari permasalahan ini dengan permasalahan yang lebih kecil?

Berikut adalah komponen utama dari sebuah prosedur:

1. Buatlah sebuah algoritma untuk menyelesaikan permasalahan $n!$. Dengan definisi bahwa $n! = n * (n-1) * (n-2) * \dots * 3 * 2 * 1$, dengan $n \geq 0$ dan n suatu bilangan bulat.

```
package main
import "fmt"
func main() {
    var n int
    fmt.Print("Masukkan angka: ")
    fmt.Scan(&n)
    jawaban := faktorialDari(n)
```

```
        fmt.Printf("Hasil dari %d faktorial adalah %d\n", n,
jawaban)
    }

func faktorialDari(n int) int {
    // base case
    if n == 1 {
        return 1
    }
    return n * faktorialDari(n-1)
}
```

Strategi awal tentu dalam hal ini ialah memahami definisi dari faktorial itu sendiri, Anda bisa menyelesaikan suatu masalah secara rekursif jika dan hanya jika Anda paham definisi umum dari permasalahan itu sendiri.

Yang kedua, tentukan base case-nya. Kita tahu bahwa sebenarnya nilai faktorial akan tetap ketika sudah dikalikan dengan 1. Yang dimana kita tahu bahwa $1! = 1$.

Yang ketiga, tentukan Recurrence Relation-nya. Untuk kasus $n > 1$, dalam mencari nilai $n!$ kita harus cari dulu dong nilai $(n-1)$ nya. Lalu, tinggal kalikan saja $\rightarrow n * (n-1)$. Nah, sekarang kita sudah tahu apa relasi rekursifnya.

Dengan memakai pendekatan rekursif berarti Anda butuh alokasi memori lebih. Makin dalam pemanggilan fungsinya, maka makin besar memori, waktunya pun cenderung makin lambat dari iteratif.

II. GUIDED

Guided 1 | Source Code + Screenshot hasil program beserta penjelasan

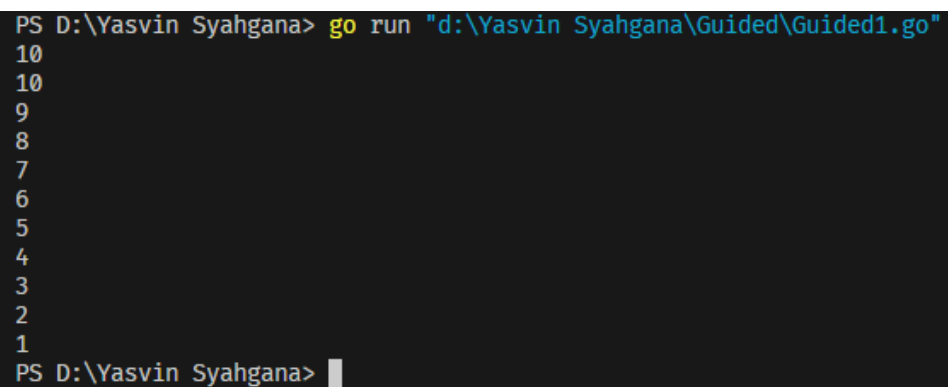
```
package main

import "fmt"

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

func baris(bilangan int) {
    if bilangan == 1 {
        fmt.Println(1)
    } else {
        fmt.Println(bilangan)
        baris(bilangan - 1)
    }
}
```

Output



```
PS D:\Yasvin Syahgana> go run "d:\Yasvin Syahgana\Guided\Guided1.go"
10
10
9
8
7
6
5
4
3
2
1
PS D:\Yasvin Syahgana>
```

Deskripsi

Program ini digunakan untuk menghitung baris dengan menggunakan fungsi rekursif yaitu `func baris(bilangan int)` yang dimana fungsi rekursif adalah mengembalikan nilai fungsi itu sendiri. Fungsi Rekursif pada fungsi ini bekerja dengan memecah masalah besar (mencetak n hingga 1) menjadi masalah yang lebih kecil (mencetak n-1 hingga 1) sampai mencapai kasus dasar di mana tidak ada lagi pemanggilan yang diperlukan. Program ini memiliki kondisi berhenti Ketika *bilangan* == 1, fungsi hanya mencetak angka 1 tanpa melakukan pemanggilan rekursif lagi. Ini menghindari rekursi tanpa batas dan memastikan fungsi berhenti.

Guided 2 | Source Code + Screenshot hasil program beserta penjelasan

```
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)
    }
}
```

Output

```
PS D:\Yasvin Syahgana> go run "d:\Yasvin Syahgana\Guided\Guided1.go"
5
15
PS D:\Yasvin Syahgana> █
```

Deskripsi

Program ini digunakan untuk menghitung sebuah penjumlahan rekursif yang dimana menghitung n secara berulang ulang hingga memenuhi kondisi base case yaitu $n = 1$, cara kerja output diatas adalah :

- $n = 5$, maka $n + \text{func penjumlahan } (4)$
- $n = 4$, maka $n + \text{func penjumlahan } (3)$
- $n = 3$, maka $n + \text{func penjumlahan } (2)$
- $n = 2$, maka $n + \text{func penjumlahan } (1)$
- $n = 1$, maka basecase (1)

Sehingga $n = 5 + 4 + 3 + 2 + 1 = 15$

III. UNGUIDED

Unguided 1 || Source Code + Screenshot hasil program beserta penjelasan

```
package main

import "fmt"

func fibonacci(n int) int {
    if n == 0 {
        return 0
    } else if n == 1 {
        return 1
    }
    return fibonacci(n-1) + fibonacci(n-2)
}

func main() {
    //var n int
    fmt.Print("n : ")
    for i := 0; i <= 10; i++ {
        fmt.Print(i, " ")
    }
    fmt.Print("\nSn: ")
    for j := 0; j <= 10; j++ {
        fmt.Print(fibonacci(j), " ")
    }
}
```

```
}
```

Screenshoot Output

```
PS D:\Yasvin Syahgana> go run "d:\Yasvin Syahgana\Unguided1\unguided1.go"
n : 0 1 2 3 4 5 6 7 8 9 10
Sn: 0 1 1 2 3 5 8 13 21 34 55
PS D:\Yasvin Syahgana>
```

Deskripsi Program

Program di atas dibuat untuk menghitung dan menampilkan deret Fibonacci hingga elemen ke-10. Deret Fibonacci adalah deret bilangan yang setiap angkanya merupakan penjumlahan dari dua angka sebelumnya (baris $S_{n-1} + S_{n-2}$), Program ini merupakan fungsi rekursif memiliki base case saat $n = 0$, Cara kerjanya memiliki kesamaan dengan guided yaitu, memanggil dirinya sendiri dan menjumlahkan juga seperti :

- $\text{Fibonacci}(n) = (S_{n-1}) + (S_{n-2})$
- $\text{fibonacci}(0) = 0$
- $\text{fibonacci}(1) = 1$
- $\text{fibonacci}(2) = 1 (1 + 0)$
- $\text{fibonacci}(3) = 2 (1 + 1)$
- $\text{fibonacci}(4) = 3 (2 + 1)$
- $\text{fibonacci}(5) = 5 (3 + 2)$
- dan seterusnya

Unguided 2 || Source Code + Screenshot hasil program beserta penjelasan

```
package main

import "fmt"

func polbint(n int) {
    if n == 0 {
        return
    }
}
```

```

    }
    polbint(n - 1)
    for i := 0; i < n; i++ {
        fmt.Print("*")
    }

    fmt.Println()
}

func main() {
    var n int
    fmt.Print("Input (n): ")
    fmt.Scan(&n)
    polbint(n)
}

```

Screenshoot Output

```

Input (n): 5
*
**
***
****
*****
PS D:\Yasvin Syahgana> go run "d:\Yasvin Syahgana\Unguided2\unguided2.go"
Input (n): 1
*
PS D:\Yasvin Syahgana> go run "d:\Yasvin Syahgana\Unguided2\unguided2.go"
Input (n): 3
*
**
***
PS D:\Yasvin Syahgana>

```

Deskripsi Program

Program di atas dibuat untuk membuat sebuah pola bintang yang menurun seperti sebuah tangga atau piramida terbalik, func polbint(n int) memiliki kondisi base case = 0, dan perulangan fungsi atau rekursif function, dengan kondisi (n-1) hingga basecase n=0. Program ini memiliki cara kerja seperti :

- Input (n)
- Kemudian polbint(n)
- Didalam fungsi polbint(n) terdapat pemanggilan func polbint(n-1)
- Sehingga n akan terus menerus habis dan hingga memenuhi kondisi base case yaitu n = 0. maka program akan berhenti

Unguided 3 || Source Code + Screenshot hasil program beserta penjelasan

```
package main

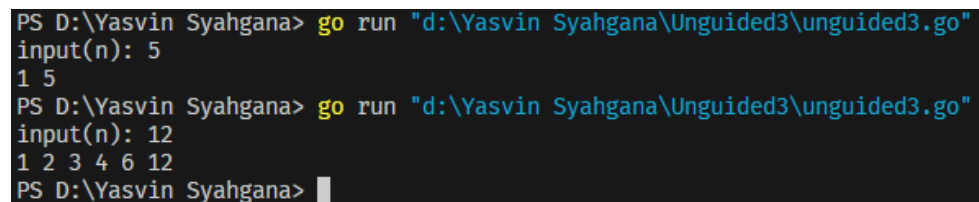
import "fmt"

func faktorisasi(n int, f int) {
    if n < f {
        return
    }
    if n%f == 0 {
        fmt.Print(f, " ")
    }
    faktorisasi(n, f+1)
}

func main() {
    f := 1
    var n int
    fmt.Print("input(n): ")
    fmt.Scanln(&n)

    faktorisasi(n, f)
}
```

Screenshoot Output



```
PS D:\Yasvin Syahgana> go run "d:\Yasvin Syahgana\Unguided3\unguided3.go"
input(n): 5
1 5
PS D:\Yasvin Syahgana> go run "d:\Yasvin Syahgana\Unguided3\unguided3.go"
input(n): 12
1 2 3 4 6 12
PS D:\Yasvin Syahgana> █
```

Deskripsi Program

Program ini dibuat untuk menghitung faktorisasi bilangan bulat **n**, dengan base case **n < f**, maka berhenti. Cara kerja fungsinya adalah seperti perkalian 2 bilangan dengan cara membagi nilai **n** dengan **f** dan setiap perulangan **f** akan ditambah 1, hingga nilai **f** melebihi nilai **n**, ketika masih memenuhi kondisi maka akan terlihat seperti berikut :

- 1 * 12
- 2 * 6
- 3 * 4

Jika tidak memenuhi kondisi `if n%f == 0`, maka program akan berhenti dan menampilkan semua output yang ada.

Unguided 4 || Source Code + Screenshot hasil program beserta penjelasan

```
package main

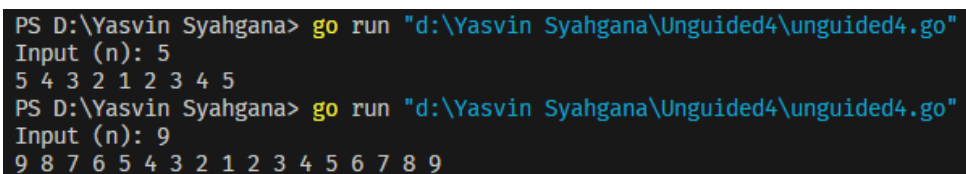
import "fmt"

func domino(n, f int) {
    if n == 1 {
        fmt.Print(1, " ")
        return
    }

    fmt.Print(n, " ")
    domino(n-1, f)
    fmt.Print(n, " ")
}

func main() {
    var n int
    fmt.Print("Input (n): ")
    fmt.Scan(&n)
    domino(n, n)
}
```

Screenshot Output



```
PS D:\Yasvin Syahgana> go run "d:\Yasvin Syahgana\Unguided4\unguided4.go"
Input (n): 5
5 4 3 2 1 2 3 4 5
PS D:\Yasvin Syahgana> go run "d:\Yasvin Syahgana\Unguided4\unguided4.go"
Input (n): 9
9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9
```

Deskripsi Program

Program ini dibuat untuk menghitung sebuah domino angka dari inputan pengguna, program ini memiliki 2 base case, yaitu `if n == 1` digunakan untuk pengurangan nilai **n**, memenuhi kondisi `n = 1`, ketika memenuhi kondisi tersebut maka berakhirnya pengurangan nilai **n**, program akan menjalankan kondisi base case kedua yaitu 1 ke N, berikut cara kerja fungsi diatas :

- Fungsi dipanggil pertama kali dengan domino(5, 5):
- Mencetak 5 → domino(4, 5)
- Memasuki rekursi untuk penuruna nilai **n**
- Mencetak 4 → domino(3, 5)
- Mencetak 3 → domino(2, 5)
- Mencetak 2 → domino(1, 5)
- Kondisi base case `n == 1`
- Saat mencapai domino(1, 5), angka 1 dicetak dan return maka rekursi pada func berhenti turun, dan akan memulai tahapan pengembalian nilai.
- Kembali ke domino(2, 5) dan mencetak 2 lagi.
- Kembali ke domino(3, 5) dan mencetak 3 lagi.
- Kembali ke domino(4, 5) dan mencetak 4 lagi.
- Kembali ke domino(5, 5) dan mencetak 5 lagi.
- Output akhir untuk `N = 5`:
- Hasil output menjadi: 5 4 3 2 1 2 3 4 5

Dapat disimpulkan maka fungsi tersebut memiliki pola pencetakan yang menurun dari `N` ke 1, kemudian naik kembali dari 1 ke `N`

Unguided 5 || Source Code + Screenshot hasil program beserta penjelasan

```
package main

import "fmt"

func ganjil(i, n int) {
    if i > n {
        return
    }
    if i%2 != 0 {
        fmt.Print(i, " ")
    }
    ganjil(i+1, n)
}

func main() {
```

```

    var n int
    fmt.Print("Input (n): ")
    fmt.Scan(&n)

    ganjil(0, n)
}

```

Screenshoot Output

```

Input (n): 5
1 3 5
PS D:\Yasvin Syahgana> go run "d:\Yasvin Syahgana\Unguided5\unguided5.go"
Input (n): 20
1 3 5 7 9 11 13 15 17 19
PS D:\Yasvin Syahgana>

```

Deskripsi Program

Program ini dibuat untuk mencari sebuah bilangan ganjil, dari inputan *user* akan tetapi pada fungsi kali ini menggunakan fungsi rekrusif yang dimana fungsi tersebut perulangan sebuah fungsi terus menerus hingga mencapai sebuah kondisi dari base case, base case pada program diatas adalah jika $i > n$, maka program akan berhenti merekrusif fungsi tersebut. Berikut beberapa tahapan dari fungsi tersebut :

- Input (0, 5) setelah diinputkan kedalam fungsi akan menjadi berikut
- `func ganjil(0, 5)` tidak mengeprint apa apa karena bukan bilangan ganjil kemudian
- `func ganjil(1, 5)` mengeprint bilangan ganjil, karena 1 merupakan bilangan ganjil
- program akan terus menerus seperti itu sampai memenuhi kondisi base case yang dimana $i > n$, contoh `func ganjil(6, 5)` kondisi tersebut tidak dapat melanjutkan rekursif function.

Unguided 6 || Source Code + Screenshot hasil program beserta penjelasan

```

package main

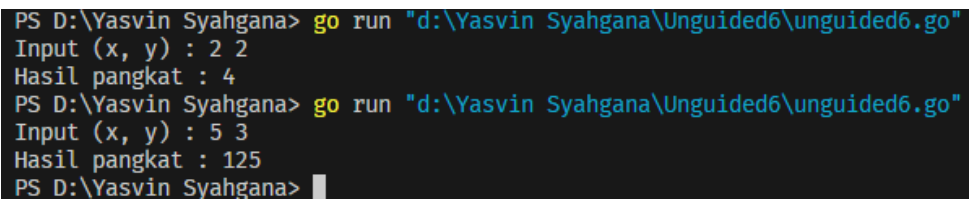
import "fmt"

func pangkat(x, y int) int {
    if y == 0 {
        return 1
    }
    return x * pangkat(x, y-1)
}

```

```
func main() {  
    var x, y int  
    fmt.Print("Input (x, y) : ")  
    fmt.Scan(&x, &y)  
    fmt.Print("Hasil pangkat : ", pangkat(x, y))  
}
```

Screenshoot Output



```
PS D:\Yasvin Syahgana> go run "d:\Yasvin Syahgana\Unguided6\unguided6.go"  
Input (x, y) : 2 2  
Hasil pangkat : 4  
PS D:\Yasvin Syahgana> go run "d:\Yasvin Syahgana\Unguided6\unguided6.go"  
Input (x, y) : 5 3  
Hasil pangkat : 125  
PS D:\Yasvin Syahgana>
```

Deskripsi Program

Program ini dibuat untuk mencari sebuah perpangkatan dari x^y , yang dimana menggunakan fungsi rekursif, fungsi diatas memiliki pengembalian nilai atau return, x (bilangan pokok) dan y (eksponen), fungsi tersebut memiliki base case yang dimana jika $y == 0$ maka return 1, maksudnya $x^0 = 1$, jika $y != 0$, maka fungsi memanggil dirinya sendiri dengan mengurangi eksponen ($y-1$) hingga mencapai kondisi dasar ($y == 0$), mengalikan hasil setiap langkah rekursif dengan x.

IV. DAFTAR PUSTAKA

A. Baihaqi, "Pemograman: Kupas Tuntas Rekursif," Medium, 21 Oktober 2023. <https://medium.com/@achmadbaihaqie9/pemograman-kupas-tuntas-rekursif-1cb97a195c65>