

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 6  
REKURSIF**



**Disusun Oleh :**

**Loisa Vanica Saragih/2311102280**

**S1 IF11-05**

**Dosen Pengampu :**

**Arif Amrulloh, S.Kom., M.Kom.**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## I. DASAR TEORI

Rekursif adalah teknik pemrograman di mana suatu fungsi memanggil fungsi itu sendiri untuk menyelesaikan prototipe masalah yang lebih kecil dari kemungkinan volte masalah saat ini. Konsep ini didasarkan pada mathematical induction, secara lain hal ini terkait dengan tampilan dari bobol pewarisan dari solusi dengan memecahkan masalah yang lebih kecil dari hal itu. Penyelesaian rekursif melakukan 2 cara yaitu :

1. Base case
  - Kondisi yang menghentikan rekursi
  - Mencegah infinite recursion
  - Memberikan nilai return langsung
2. Recursive case
  - Pemanggilan fungsi terhadap dirinya sendiri
  - Parameter harus mendekati base case
  - Memecah masalah menjadi lebih kecil

### Kelebihan Rekursif

- Kode lebih mudah dibaca dan dipahami
- Mengurangi kompleksitas program
- Cocok untuk struktur data rekursif (tree, graph)
- Implementasi lebih alami untuk beberapa algoritma

### Keterbatasan Rekursif

- Membutuhkan lebih banyak memori
- Dapat lebih lambat dari iterasi
- Risk stack overflow untuk input besar
- Overhead dari pemanggilan fungsi berulang

## II. GUIDED

1. Membuat baris bilangan dari n hingga 1

Base-case: bilangan == 1

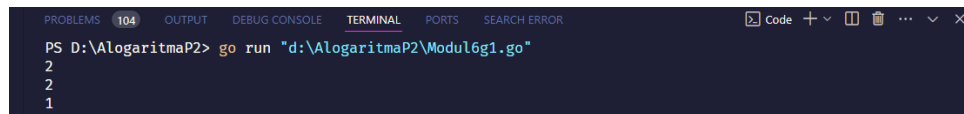
### Sourcecode

```
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)
    }
}
```

### Screenshoot Output



```
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul6g1.go"
5
4
3
2
1
```

### Deskripsi Program

Program Go ini diawali dengan didefinisikan package main dan mengimport package fmt user input/output, dalam fungsi main() variabel n dengan tipe data integer di deklarasikan dan diinputkan oleh user dengan `fmt.Scan(&n)` dan memanggil fungsi `baris(n)` rekursif. Fungsi `baris(bilangan int)` mempunyai base case di mana jika bilangan sama dengan 1 maka akan mencetak angka 1, tetapi jika bilangan lebih dari 1 maka akan mencetak nilai bilangan tersebut kemudian memanggil dirinya sendiri dengan parameter `(bilangan - 1)`. Sebagai contoh, jika user menginput `n = 5`, maka program akan mencetak angka dari 5 hingga 1 secara menurun dengan alur yaitu untuk `baris(5)` mencetak 5 lalu memanggil `baris(4)`, `baris(4)` mencetak 4 lalu memanggil `baris(3)`, `baris(3)` mencetak 3 lalu memanggil `baris(2)`, `baris(2)` mencetak 2 lalu memanggil `baris(1)`, dan pada akhirnya `baris(1)` mencetak 1.

2. Menghitung hasil penjumlahan 1 hingga n  
Base-case:  $n == 1$

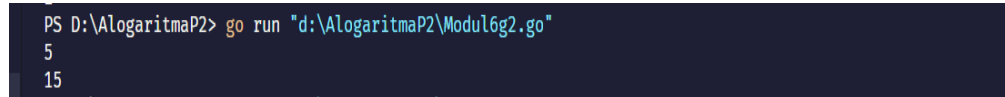
#### Sourcecode

```
package main
import "fmt"

func penjumlahan (n int) int{
    if n == 1{
        return 1
    } else {
        return n + penjumlahan(n-1)
    }
}

func main(){
    var n int
    fmt.Scan(&n)
    fmt.Println(penjumlahan(n))
}
```

#### Screenshoot Output



```
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul6g2.go"
5
15
```

#### Deskripsi Program

Program Go ini merupakan program rekursif untuk menghitung penjumlahan deret bilangan dari 1 sampai n. Program ini dimulai dengan deklarasi master dengan menggandeng package main dan memperkenalkan package fmt untuk input/output yang berisi fungsi penjumlahan(n int) yang menerima satu parameter berupa bilangan bulat dalam bentuk integer dan mengembalikan juga nilai integer. Base case dari fungsi penjumlahan ini adalah, apabila  $n=1$  maka akan mengembalikan nilai 1, selain  $n=1$  maka akan mengembalikan hasil n ditambah dengan pemanggilan penjumlahan sebanyak n-1. Dalam fungsi main(), program mendeklarasikan variabel n bertipe data integer yang diisi oleh user dengan `fmt.Scan(&n)` dan memanggil berdasarkan results fungsi penjumlahan(n). Sebagai contoh, jika user menginput  $n = 5$ , maka program akan menghitung  $5 + (4 + (3 + (2 + 1)))$  dengan alur rekursif  $\text{penjumlahan}(5) = 5 + \text{penjumlahan}(4)$ ,  $\text{penjumlahan}(4) = 4 + \text{penjumlahan}(3)$ ,  $\text{penjumlahan}(3) = 3 + \text{penjumlahan}(2)$ ,  $\text{penjumlahan}(2) = 2 + \text{penjumlahan}(1)$  yang diperoleh = 15 dari  $5+4+3+2+1$ .

### 3. Mencari dua pangkat $n$ atau $2^n$

Base-case:  $n == 0$

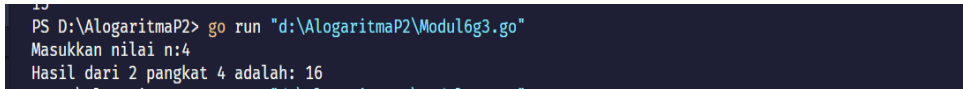
#### Sourcecode

```
package main
import "fmt"

//fungsi rekursif untuk menghitung  $2^n$ 
func pangkat(n int) int {
    if n == 0 {
        return 1
    } else {
        return 2 * pangkat(n-1)
    }
}

func main() {
    var n int
    fmt.Print("Masukkan nilai n:")
    fmt.Scan(&n)
    fmt.Println("Hasil dari 2 pangkat", n, "adalah:", pangkat(n))
}
```

#### Screenshoot Output



```
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul6g3.go"
Masukkan nilai n:4
Hasil dari 2 pangkat 4 adalah: 16
```

#### Deskripsi Program

Program ini Go ini adalah program rekursif yang digunakan untuk menentukan nilai dari  $2^n$ . Program diawali dengan penulisan statement dasar dengan meng-deklarasikan main package dan meng-import package `fmt` untuk input/outputnya. Adapun fungsi `pangkat(n int)` yang di input sebuah bilangan dan memerintahkan serta menerima output bilangan integer. Base case pada fungsi `pangkat` ini adalah bahwa jika  $n$  equals 0 function call kembacakn dengan nilai 1 (because  $2^0 = 1$ ) and if  $n > 0$  kembalikan nilai 2 times pangkat ( $n-1$ ). Dalam fungsi `main()`, program mendeklarasikan variabel `n` bertipe integer, menampilkan pesan "Masukkan nilai n: dan dengan menggunakan `fmt.Scan(&n)` terima input dari user lalu Menggunakan `pangkat(n)` dan menampilkan output berformat Hasil dari 2 pangkat  $n$  adalah: hasil. Sebagai contoh, jika user menginput  $n = 4$ , maka program akan menghitung  $2 * (2 * (2 * (2 * 1)))$

dengan alur rekursif: Pangkat(4) = 2 \* Pangkat (3), Pangkat (3) = 2 \* Pangkat (2), Pangkat (2) = 2 \* Pangkat (1), Pangkat (1) = 2 \* Pangkat (0), dan Pangkat (0) = 1, maka akan menghasilkan output sebesar 16 dengan hasil perhitungan  $2^4$ .

#### 4. Mencari nilai factorial atau n!

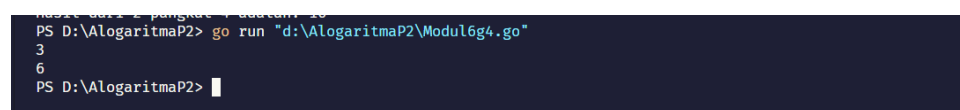
Base-case:  $n == 1$

##### Sourcecode

```
package main
import "fmt"

var n int
func faktorial (n int) int{
    if n == 0 || n == 1 {
        return 1
    }else{
        return n * faktorial(n-1)
    }
}
func main() {
    fmt. Scan(&n)
    fmt.Println(faktorial(n))
}
```

##### Screenshoot Output



```
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul6g4.go"
3
6
PS D:\AlogaritmaP2> 
```

##### Deskripsi Program

Program ini Go ini memang merupakan Program Rekursif yang digunakan untuk mengevaluasi faktorial bilangan n. Program di mulai dengan statement package main dan memanggil package fmt untuk input dan output dan juga meng declare variable global bernama n se tympe data integer. Program ini menzones sebuah fungsi faktorial atau factor dengan parameter bilangan bulan n dan menghasilkan nilai n int. Fungsi faktorial juga memiliki base case ini di mana jika n equal 0 or n equal 1, maka fungsi ini akan menghasilkan nilai equal 1 (sebab 0! equal 1 dan 1! equal 1), tetapi jika n greater than 1 maka fungsi ini akan menghasilkan nilai equal n multiplied by penggunaan recursive fungsi faktorial(n- 1). Di

dalam fungsi main(), program mengambil input dari user dengan menggunakan `fmt.Scan(&n)`, dan kemudian menclrkan fungsi faktorial(n) dan memrinta hasilnya. Sebagai contoh, jika user menginput  $n = 5$ , maka program akan menghitung  $5 * (4 * (3 * (2 * 1)))$  dengan alur rekursif:  $\text{faktorial}(5) = 5 * \text{faktorial}(4)$ ,  $\text{faktorial}(4) = 4 * \text{faktorial}(3)$ ,  $\text{faktorial}(3) = 3 * \text{faktorial}(2)$ ,  $\text{faktorial}(2) = 2 * \text{faktorial}(1)$ , dan  $\text{faktorial}(1) = 1$ , bahwa outputnya sedikh ini 120 yang merupakan  $5!$ .

## II. UNGUIDED

Deret fibonacci adalah sebuah deret dengan nilai suku ke-0 dan ke-1 adalah 0 dan 1, dan nilai 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 fibonacci 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

### Sourcecode

```
package main

import (
    "fmt"
)

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

func main() {
    var n int
    fmt.Print("Masukkan nilai n untuk batas deret Fibonacci: ")
    fmt.Scan(&n)
```

```

    fmt.Print("n\t| ")
    for i := 0; i <= n; i++ {
        fmt.Printf("%d\t", i)
    }
    fmt.Println()

    fmt.Print("Sn\t| ")
    for i := 0; i <= n; i++ {
        fmt.Printf("%d\t", fibonacci(i))
    }
    fmt.Println()
}

```

### Screenshoot Output

```

PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul6ug1.go"
Masukkan nilai n untuk batas deret Fibonacci: 0
n      | 0
Sn     | 0

```

### Deskripsi Program

Program ini merupakan tampilan dari deret Fibonacci yang ditulis dengan bahasa Go di sini fungsi yang digunakan adalah fungsi fibonacci(n int) dengan cara pada input n, apabila  $n \leq 1$  maka fungsi tersebut akan mengembalikan nilai n terpilih tersebut, sedangkan apabila  $n > 1$  maka fungsi tersebut akan menghitung nilai Fibonacci ke-n dengan memanggil fungsi fibonacci (n-1) + fibonacci(n-2). Sementara fungsi main() berperan mengambil input dari user berupa nilai n sebagai batas deret serta mencetak dua baris output yaitu baris pertama merupakan deret bilangan bulat mulai dari 0 sampai dengan n dan baris kedua sebagai deret Fibonacci. Sebagai contoh, jika user memasukkan  $n = 5$ , maka program akan menghasilkan deret: jika  $n = 0$  maka  $\text{fibonacci}(0) = 0$ , jika  $n = 1$  maka  $\text{fibonacci}(1) = 1$ , jika  $n = 2$  maka  $\text{fibonacci}(2) = \text{fibonacci}(1) + \text{fibonacci}(0) = 1 + 0 = 1$ , jika  $n = 3$  maka  $\text{fibonacci}(3) = \text{fibonacci}(2) + \text{fibonacci}(1) = 1 + 1 = 2$  Program ini menggunakan format tabel dengan tab (&sol; t) untuk memisahkan angka-angka supaya nampak lebih rapi and enak dilihat asalkan nilai-nilai dihitung dengan menggunakan recursion dengan cara setiap bilangan ini adalah jumlah dari dua bilangan sebelumnya dalam deret.



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	* ** ***

### Sourcecode

```
package main

import (
    "fmt"
)

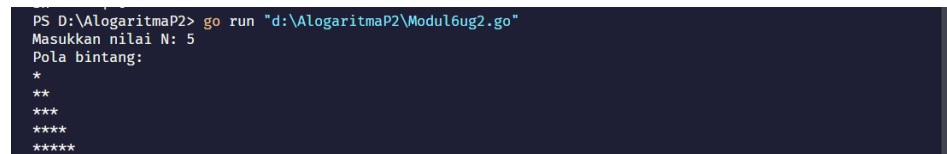
func printStars(n int) {
    if n == 0 {
        return
    }
    fmt.Print("*")
    printStars(n - 1)
}

func printPattern(n, current int) {
    if current > n {
        return
    }
    printStars(current)
    fmt.Println()
    printPattern(n, current+1)
}

func main() {
    var n int
    fmt.Print("Masukkan nilai N: ")
    fmt.Scan(&n)
```

```
    fmt.Println("Pola bintang:")
    printPattern(n, 1)
}
```

## Screenshoot Output



```
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul6ug2.go"
Masukkan nilai N: 5
Pola bintang:
*
**
***
****
*****
```

## Deskripsi Program

Program diatas adalah solusi untuk mengeprint pola bintang (\*) menggunakan fungsi rekursi dengan bahasa pemrograman Go. Fungsi `printStars()` untuk mencetak bintang secara horizontal dengan banyaknya bintang sesuai dengan nilai `n`, dalam mengimplementasikan rekursi dimana jika  $n = 0$  maka fungsi akan berhenti, jika tidak maka akan mencetak satu bintang dan memanggil fungsi lancar sendiri dengan nilai `n-1`. yang bertugas mengatur pola keseluruhan dengan parameter `n` sebagai batas maksimal dan `current` sebagai posisi baris saat ini, fungsi ini akan berhenti jika  $current > n$ , jika tidak maka akan memanggil fungsi `printStars()` untuk mencetak bintang sebanyak nilai `current`, mencetak baris baru, dan memanggil dirinya sendiri dengan `current+1`. Fungsi ketiga yaitu utama/diperintahkan oleh sistem yaitu `main()` meminta input secara langsung nilai `N` pada user interface dan memanggil fungsi `printPattern(n current=1)`. Sebagai contoh jika user memasukkan `N = 4`, maka program akan menghasilkan output:

```
'''
*
**
***
****
'''
```

di mana setiap baris mempunyai jumlah bintang sesuai dengan nomor barisnya (baris satu mempunyai satu bintang, baris dua mempunyai dua bintang dan seterusnya) yang dihasilkan oleh pemanggilan fungsi rekursif kedua ini.

**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 Nya).

Contoh masukan dan keluaran.

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

### Sourcecode

```
package main

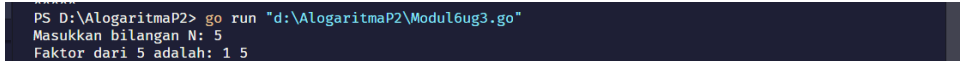
import (
    "fmt"
)

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

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

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

### Screenshoot Output



```
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul6ug3.go"
Masukkan bilangan N: 5
Faktor dari 5 adalah: 1 5
```

### Deskripsi Program

Program tersebut adalah solusi untuk mencari dan menampilkan faktor-faktor dari suatu bilangan N menggunakan rekursi yang dibuat dalam

bahasa pemrograman Go. Program terdiri dari 2 fungsi utama: Prosedur yang ada dalam fungsi `printFactors()` ini ialah menerima dua parameter yang pertama bernama `n` merupakan bilangan yang akan ditemukan faktornya dan yang kedua ialah `i` merupakan iterator/bilangan yang sedang diperiksa. Ini merupakan fungsi rekursif yang memiliki base case yaitu jika `i > n`, maka fungsi tersebut berhenti, Lalu mengecek apakah `n` habis dibagi dengan `i`, jika `i` divides `n`, maka `i` adalah faktor `n` dan akan dicetak, fungsi tersebut memanggil dirinya dengan input `i+1` untuk mengetes bilangan selanjutnya. Fungsi kedua ini adalah `main()`, fungsi ini harus meminta input bilangan `N` dari user dan memanggil fungsi `printFactors()` dengan parameter `n` dan `i = 1`. Sebagai contoh jika user memasukkan `N = 12`, maka program akan mengecek satu persatu bilangan dari 1 hingga 12, dimana bilangan yang habis membagi 12 akan dicetak sebagai faktor, sehingga outputnya adalah: “12 factors yang termasuk dalam 12 tersebut adalah: 1 2 3 4 6 12”. Program ini sama seperti program sebelumnya menggunakan cara rekursi untuk mengganti perulangan guna menemukan faktor-faktor bilangan.

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

#### Sourcecode

```
package main

import (
    "fmt"
)

func printDescending(n int) {
    if n < 1 {
        return
    }
    fmt.Print(n, " ")
    printDescending(n-1)
}
```

```

        printDescending(n - 1)
    }


    func printAscending(n, current int) {
        if current > n {
            return
        }
        fmt.Print(current, " ")
        printAscending(n, current+1)
    }

    func main() {
        var n int
        fmt.Print("Masukkan nilai N: ")
        fmt.Scan(&n)

        fmt.Printf("Hasil: ")
        printDescending(n)
        printAscending(n, 1)
        fmt.Println()
    }

```

## Screenshoot Output



```

PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul6ug4.go"
Masukkan nilai N: 5
Hasil: 5 4 3 2 1 1 2 3 4 5
PS D:\AlogaritmaP2>

```

## Deskripsi Program

Program tersebut adalah implementasi untuk mencetak deret bilangan dari yang besar ke kecil atau sebaliknya dengan bahasa pemrograman Go dengan menggunakan rekursi. Program memiliki 3 fungsi utama: Terlebih dahulu ada fungsi `printDescending(n)` yang merupakan fungsi dengan parameter `n` dan ini akan mencetak bilangan dari `n` hingga 1 dengan menggunakan rekursi dan base case akan berhenti jika `n < 1` sedang jika tidak maka akan mencetak nilai `n` dan memanggil dirinya dengan `n-1`. Kedua terdapat fungsi `printAscending()` yang memperkenalkan `n` sebagai batas maksimal dan `current` sebagai nilai saat ini, fungsi tersebut akan mencetak kedaan ‘menaik’ dari bilangan 1 hingga `n` dengan base case apabila `current > n` maka fungsi tersebut berhenti, jika tidak maka fungsi tersebut akan mencetak nilai `current` tersebut dan memanggil dirinya sendiri. Ketiga, fungsi `main()` yang mengambil input nilai `N` dari user dan

men(actions) kedua fungsi tersebut secara berurutan, yaitu fungsi `printDescending()` untuk menghasilkan deret bilangan menurun lalu `printAscending()` untuk menghasilkan deret bilangan menaik. Sebagai contoh jika user memasukkan  $N = 5$ , maka program akan menghasilkan output: Outputnya berurutan 5 4 3 2 1 1 2 3 kedua 4 5 1 di mana bagian pertama dua sama dengan dihasilkan oleh fungsi `printDescending()` dan bagian kedua dua dihasilkan oleh fungsi `printAscending`.

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

#### Sourcecode

```
package main

import (
    "fmt"
)

func printDescending(n int) {
    if n < 1 {
        return
    }
    fmt.Print(n, " ")
    printDescending(n - 1)
}

func printAscending(n, current int) {
    if current > n {
        return
    }
    fmt.Print(current, " ")
    printAscending(n, current+1)
}

func main() {
```

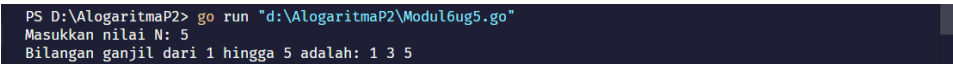
```

var n int
fmt.Print("Masukkan nilai N: ")
fmt.Scan(&n)

fmt.Printf("Hasil: ")
printDescending(n)
printAscending(n, 1)
fmt.Println()
}

```

## Screenshoot Output



```

PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul6ug5.go"
Masukkan nilai N: 5
Bilangan ganjil dari 1 hingga 5 adalah: 1 3 5

```

## Deskripsi Program

Program tersebut adalah solusi untuk mencontohkan bilangan ganjil dalam rentang 1 hingga N dengan menggunakan algoritma rekursi dalam bahasa Go. Program terdiri dari 2 fungsi utama: Yang pertama, method printOddNumbers yang menerima dua parameter integer yaitu, n sebagai batas posisi atau angka terbesar dan current sebagai bilangan yang dicek. Mekanisme limbubaru() ini merupakan fungsi yang memanggil dirinya sendiri dengan sintaks judul 'rekursi' dengan spektrak berikut ini. Visi eksekusi ini adalah dievaluasi apakah posisi 'current' lebih besar dari 'n' baru berhenti dan untuk posisi 'current' yang tepat maka akan diperiksa dengan modulo (%) apakah bukan nol, Kedua, dalam prosedur main() yang mana berfungsi untuk mengambil input nilai N dari user dan memanggil prosedur printOddNumbers() dengan parameter n dan variabel current sebesar 1. Sebagai contoh jika user memasukkan N = 10, maka program akan mengecek bilangan 1 hingga 10 satu persatu, dimana bilangan yang menghasilkan sisa 1 saat dibagi 2 (bilangan ganjil) akan dicetak, sehingga outputnya adalah: "Untuk bilangan ganjil antara 1 hingga 10 adalah: 1, 3, 5, 7, dan 9". Program yang dibuat ini menggunakan fungsi rekursif dalam penggantian perulangan untuk mencari dan mencetak bilangan ganjil.

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

### Sourcecode

```
package main

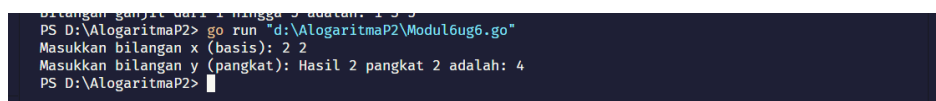
import (
    "fmt"
)

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 x (basis): ")
    fmt.Scan(&x)
    fmt.Print("Masukkan bilangan y (pangkat): ")
    fmt.Scan(&y)

    result := power(x, y)
    fmt.Printf("Hasil %d pangkat %d adalah: %d\n", x, y, result)
}
```

### Screenshoot Output



```
PS D:\AlogaritmaP2> go run "d:\AlogaritmaP2\Modul6ug6.go"
Masukkan bilangan x (basis): 2
Masukkan bilangan y (pangkat): 2
Hasil 2 pangkat 2 adalah: 4
PS D:\AlogaritmaP2>
```

### Deskripsi Program

Program tersebut adalah solusi untuk menghitung nilai hasil perpangkatan suatu bilangan dengan menggunakan fungsi rekursi di dalam bahasa Go. Program terdiri dari 2 fungsi utama: Terlebih dahulu, yaitu function `power()` dengan dua parameter yang mengambil sebuah dasar bernama `x`



dan sebuah eksponen bernama y. Fungsi ini bekerja secara rekursif dengan meng/ menentukan base case yaitu jika nilai  $y = 0$  maka fungsi akan mengembalikan nilai “1” ( karena apapun dikalikan satu maka hasilnya adalah 1) selainnya fungsi akan mengembalikan hasil dari pengali/none catcher dari  $x$  *power* ( $x, y-1$ ). *Kedua, dalam fungsi main() tersebut ada proses meminta input nilai x sebagai basis dan y sebagai pangkat dari user, kemudian mengesayikan fungsi power() dan menyimpan hasilnya dalam variabel result lalu men liking hasilnya. Sebagai contoh jika user memasukkan  $x = 2$  dan  $y = 3$ , maka proses perhitungannya adalah:*

$$\text{power}(2,3) = 2 \text{ power}(2,2) = 2 (2 \text{ power}(2,1)) = 2 (2 (2 \text{ power}(2,0))) = 2 (2 (2 \cdot 1)) = 2 (2 \cdot 2) = 2 * 4 = 8,$$

karena itu outputnya adalah: “Hasil 2 pangkat 3 adalah 8”. Program ini juga menggunakan teknik rekursi untuk menentukan perpangkatan tanpa melalui proses perulangan.

#### **IV. Kesimpulan**

Rekursif sebagai teknik pemrograman sangat efektif untuk memecahkan suatu masalah yang kompleks dalam cara yang sangat elegant. Namun, walaupun demikian, rekursif tetap menjadi solusi yang populer untuk banyak kasus pada pemrograman di mana terlibat struktur data piramida dan persoalan-persoalan yang relative bisa dipecah menjadi persoalan-persoalan serupa.

## **DAFTAR PUSTAKA**

- 1.Cormen, T. H., et al. (2009). Introduction to Algorithms, Third Edition. MIT Press.
- 2.Sedgewick, R., & Wayne, K. (2011). Algorithms, 4th Edition. Addison-Wesley.
- 3.Knuth, D. E. (1997). The Art of Computer Programming, Volume 1: Fundamental Algorithms. Addison-Wesley.