

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL VI  
REKURSIF**



**Disusun Oleh :**

**Aji Noto Sutrisno (2311102262)**

**IF 11 05**

**Dosen Pengampu :**

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

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## I. DASAR TEORI

*Recursive function* adalah sebuah *function* yang memanggil/mengeksekusi dirinya sendiri. Recursive function Bisa dikatakan salah satu yang Bisa kita gunakan Buat melakukan perulangan. Ketika menulis kode aplikasi, terkadang Terdapat kasus dimana akan lebih mudah Apabila dilakukan dengan recursive function. Misalnya penggunaan sederhana recursive function adalah ketika melakukan operasi factorial.

Dalam Go fungsi rekursif yang sederhana itu bentuknya seperti ini:

```
func doFactorial(value int) int {  
    if value == 1 {  
        return value  
    }  
    return value * doFactorial(value-1)  
}
```

Pada contoh di atas, Function tersebut menerima *value* dengan tipe integer dan mengembalikan data dengan tipe integer. Seperti yang sudah dijelaskan sebelumnya bahwa recursive function harus dapat berhenti, maka kita Membangun pengkondisian Buat memeriksa Apabila *value* bernilai 1 maka kita langsung mengembalikannya tanpa memanggil function lagi. Sebaliknya, Apabila *value* Tak sama dengan satu maka *value* akan dikalikan dengan nilai kembalian dari function *doFactorial* dengan mengisi parameter *value* dikurang 1. Pengurangan ini bertujuan agar parameter yang dikirimkan pada eksekusi function berikutnya adalah berisi nilai pada urutan sebelumnya. Sehingga, Apabila kita memanggil function *doFactorial* dengan parameter 5 maka parameter berikutnya akan menjadi 4, kemudian 3, kemudian 2, dan 1. Akhirnya recursive function Tak akan dipanggil lagi.

Berikut fungsi faktorial tanpa rekursif dengan cara menggunakan for loop.

```
func doFactorialWithForLoop(value int) int {
```

```
    result := 1
    for i := value; i > 0; i-- {
        result *= i
    }
    return result
}
```

Kedua program diatas mungkin tidak memiliki banyak perbedaan yang signifikan, apabila dilihat dari programnya. Tapi berikut ini adalah beberapa kelebihan dan kekurangan menggunakan *recursive function*.

Kelebihan:

- Kode mudah ditulis
- Kurangi pemanggilan fungsi yang Tak perlu
- Sangat Bermanfaat ketika menerapkan solusi yang sama
- Rekursi mengurangi panjang kode
- Sangat Bermanfaat dalam memecahkan masalah struktur data

Kekurangan:

- Recursive function umumnya lebih Gial daripada fungsi non-rekursif
- Terkadang memerlukan memori yang lebih Buat menyimpan hasil sementara dalam stack
- Terkadang sulit Buat menganalisis atau memahami kodenya
- Tak terlalu efisien dalam konsep kompleksitas waktu
- Kemungkinan out of memory Apabila kondisi recursive Tak diperiksa dengan Betul

## II. GUIDED

### 1. Soal Studi Case

#### Sourcecode

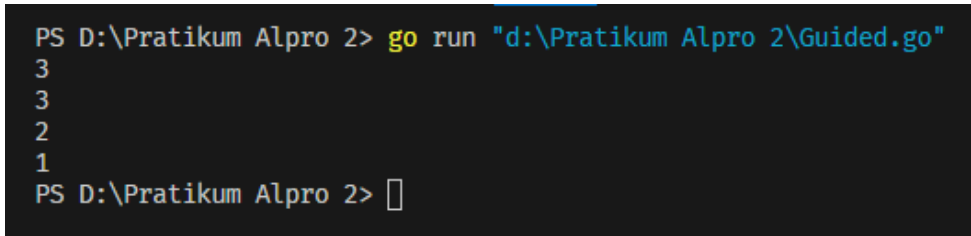
```
package main

import "fmt"

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

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

#### Output



```
PS D:\Pratikum Alpro 2> go run "d:\Pratikum Alpro 2\Guided.go"
3
3
2
1
PS D:\Pratikum Alpro 2> 
```

#### Deskripsi

Program diatas dibuat untuk menghitung penjumlahan dengan menggunakan fungsi rekursif, yang dimana terdapat `func baris(bilangan int)` memiliki kondisi base case dimana jika input (**bilangan**) itu sama dengan **1**, dan memiliki fungsi rekursif pada jika input(**bilangan**) tidak sama dengan **1**, maka melakukan perulangan secara terus menerus hingga memenuhi base case yaitu **1**, rekursif fungsinya terletak pada pemanggilan fungsi didalam sebuah fungsi `baris(bilangan - 1)`. Kenapa terdapat pengurangan dalam pemanggilannya karena jika tidak dikurangi maka akan infinite loop dan akan mengeprint inputan(**bilangan**) dari user. Contoh jika user menginputkan **10**, dan kondisi rekursifnya `baris(bilangan)`. Akan *infinite loop* mengeprint "**10**"

## 2. Soal Studi Case

### Sourcecode

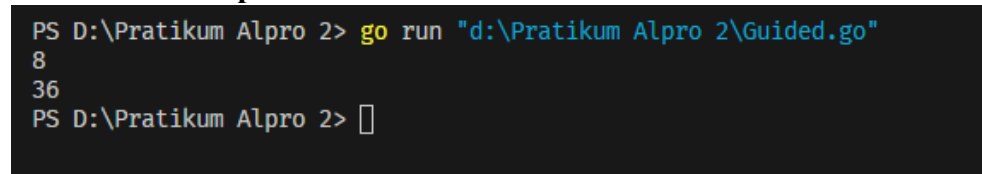
```
package main

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.Print(penjumlahan(n))
}
```

### Screenshoot Output



```
PS D:\Pratikum Alpro 2> go run "d:\Pratikum Alpro 2\Guided.go"
8
36
PS D:\Pratikum Alpro 2> █
```

### Deskripsi Program

Program diatas dibuat untuk mencari hasil penjumlahan dari inputan user yang dikurang secara terus menerus(rekursif). Program diatas memiliki fungsi `func penjumlahan(n int)` yang dimana memiliki kondisi base case berupa **n** sama dengan **1** maka program akan mereturn inputan user, cara kerja program diatas adalah, *user* akan menginputkan sebuah bilangan bulat, pada contoh output diatas, *user* menginputkan **"8"** dan hasilnya **"36"**. Fungsi rekursif diatas memiliki penyimpanan temporary yang dimana program akan menyimpan data sementara, pada kasus diatas, program akan menyimpan **n +**, kemudian melakukan perulangan rekursif dengan fungsi `penjumlahan(n-1)`, Jika dituliskan maka,  $8 + \text{penjumlahan}(7)$ , Program akan menyimpan data **n** yaitu **"8"** kemudian dijumlahkan dengan **n** yang baru yaitu **"7"**, kemudian fungsi akan mengembalikannya lagi `penjumlahan(6)` dan seterusnya hingga memenuhi kondisi base case yaitu **n == 1**. Maka akan terjadi penjumlahan seperti  $8+7+6+5+4+3+2+1 = 36$ .

### 3. Soal Studi Case

#### Sourcecode

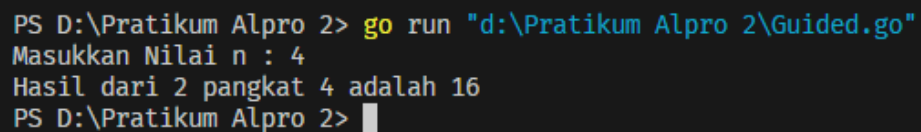
```
package main

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:\Pratikum Alpro 2> go run "d:\Pratikum Alpro 2\Guided.go"
Masukkan Nilai n : 4
Hasil dari 2 pangkat 4 adalah 16
PS D:\Pratikum Alpro 2> █
```

#### Deskripsi Program

Program diatas dibuat untuk mencari hasil perpangkatan dari 2 pangkat n, yang dimana program diatas melakukan perulangan sebanyak inputan n, Program diatas memiliki fungsi `func pangkat(n int)` dengan kondisi base case yaitu n sama dengan 0, dan program ini akan melakukan perulangan 2 sebanyak n-kali, cara kerja programnya adalah *user* menginputkan nilai **n**, kemudian **n** akan di lakukan perulangan rekursif `2 * pangkat(n-1)`. Program ini sama dengan program guided ke 2 yang dimana yang membedahkan adalah operasinya dan variabel kontans pada nilai *return*. Jadi jika  $n = 4$ , maka program akan melakukan  $2 * pangkat(4-1)$ . Program akan menyimpan sementara nilai  $2 *$ , kemudian melakukan perulangan hingga  $n == 0$ , maka akan didapatkan nilai *return* yaitu  $2 * 2 * 2 * 2 = 16$

#### 4. Soal Studi Case

##### Sourcecode

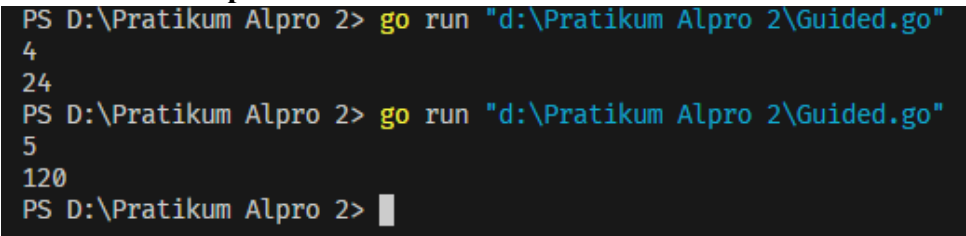
```
package main

import "fmt"

//n = 5
//5*4*3*2*1
func faktorial(n int) int {
    if n == 0 || n == 1 {
        return 1
    } else {
        return n * faktorial(n-1)
    }
}

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

##### Screenshoot Output



```
PS D:\Pratikum Alpro 2> go run "d:\Pratikum Alpro 2\Guided.go"
4
24
PS D:\Pratikum Alpro 2> go run "d:\Pratikum Alpro 2\Guided.go"
5
120
PS D:\Pratikum Alpro 2> █
```

##### Deskripsi Program

Program diatas dibuat untuk mencari sebuah faktorial dari **n**, Program diatas memiliki fungsi rekursif berupa `func faktorial(n int)` yang dimana memiliki kondisi base case yaitu jika `n == 0` atau `n == 1`, program ini melakukan perkalian **n**, dengan **(n-1)** sehingga akan menampilkan hasil faktorial dari **n**, Cara kerja program ini adalah *user* akan menginputkan nilai **n** yang berupa (4) kemudian akan melakukan perulangan rekursif `return 4 * faktorial(4-1)`, maka akan menjadi seperti ini  $4 * 3 * 2 * 1 = 24$ , dan begitulah cara kerja dari fungsi rekursif yang dimana jika `n == 1` maka akan berhenti atau bisa disebut juga sebagai base case.

### III. UNGUIDED

#### 1. Soal Studi Case

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

func main() {
    fmt.Println("n : ")
    for i := 0; i < 10; i++ {
        fmt.Print(i, " ")
    }

    fmt.Println("\nSn : ")
    for j := 0; j < 10; j++ {
        fmt.Print(fibonacci(j), " ")
    }
}
```

#### Screenshoot Output

```
PS D:\Pratikum Alpro 2\Laprak 5\Unguided 1> go run "d:\Pratikum Alpro 2\Laprak 5\Unguided 1\main.go"
n : 0 1 2 3 4 5 6 7 8 9
Sn : 0 1 1 2 3 5 8 13 21 34
PS D:\Pratikum Alpro 2\Laprak 5\Unguided 1> |
```



## Deskripsi Program

Program di atas dirancang untuk menghitung dan menampilkan deret Fibonacci hingga elemen ke-10. Deret Fibonacci merupakan rangkaian angka di mana setiap angka adalah hasil penjumlahan dari dua angka sebelumnya (yaitu  $S_n - 1 + S_n - 2$ ). Program ini menggunakan fungsi rekursif dengan base case ketika  $n = 0$ . Prosesnya mirip dengan metode guided, di mana fungsi tersebut memanggil dirinya sendiri dan melakukan penjumlahan seperti berikut:

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

## 2. Soal Studi Case

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

    fmt.Println()
}

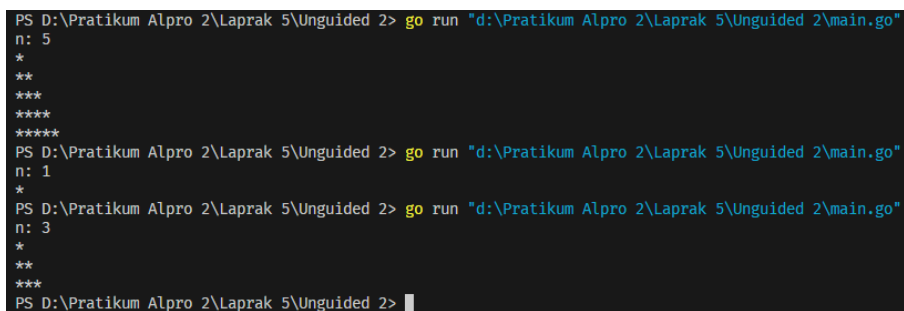
func main() {

    var n int

    fmt.Print("n: ")
    fmt.Scan(&n)

    bintang(n)
}
```

## Screenshoot Output



```
PS D:\Pratikum Alpro 2\Laprak 5\Unguided 2> go run "d:\Pratikum Alpro 2\Laprak 5\Unguided 2\main.go"
n: 5
*
**
***
****
*****
PS D:\Pratikum Alpro 2\Laprak 5\Unguided 2> go run "d:\Pratikum Alpro 2\Laprak 5\Unguided 2\main.go"
n: 1
*
PS D:\Pratikum Alpro 2\Laprak 5\Unguided 2> go run "d:\Pratikum Alpro 2\Laprak 5\Unguided 2\main.go"
n: 3
*
**
***
PS D:\Pratikum Alpro 2\Laprak 5\Unguided 2> █
```

## Deskripsi Program

Program ini dirancang untuk menghasilkan pola bintang yang menurun berbentuk seperti tangga atau piramida terbalik. Fungsi `bintang(n int)` memiliki base case pada `n = 0` dan memanfaatkan fungsi rekursif dengan pemanggilan kembali hingga mencapai base case. Proses kerjanya adalah sebagai berikut:

- Masukkan nilai (`n`)
- Panggil fungsi `bintang(n int)`

- Di dalam fungsi `bintang(n int)` terdapat pemanggilan `bintang(n int)`
- Nilai `n` akan terus berkurang hingga mencapai base case yaitu `n = 0`, setelah itu program akan berhenti.

### 3. Soal Studi Case

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

#### Sourcecode

```
package main

import "fmt"

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

func main() {
    var n int

    fmt.Print("n : ")
    fmt.Scan(&n)
    fmt.Print("Faktor dari ", n, ": ")
}
```

```
faktorbil(n, 1)
}
```

## Screenshoot Output

```
PS D:\Pratikum Alpro 2\Laprak 5\Unguided 3> go run "d:\Pratikum Alpro 2\Laprak 5\Unguided 3\main.go"
n : 5
Faktor dari 5: 1 5
PS D:\Pratikum Alpro 2\Laprak 5\Unguided 3> go run "d:\Pratikum Alpro 2\Laprak 5\Unguided 3\main.go"
n : 12
Faktor dari 12: 1 2 3 4 6 12
PS D:\Pratikum Alpro 2\Laprak 5\Unguided 3> █
```

## Deskripsi Program

Program ini dirancang untuk menghitung faktorisasi dari bilangan bulat  $n$ , dengan menghentikan proses ketika  $i > n$  (base case). Cara kerja fungsinya mirip dengan proses perkalian dua bilangan, yaitu dengan membagi nilai  $n$  dengan  $i$ . Setiap kali perulangan, nilai  $i$  ditambah 1 hingga  $i$  melebihi  $n$ . Selama kondisi pembagian terpenuhi, Jika inputan *user* berupa “12” hasilnya akan ditampilkan sebagai berikut:

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

Jika kondisi  $n \% i == 0$  tidak terpenuhi, maka program berhenti dan menampilkan semua hasil yang diperoleh.

## 4. Soal Studi Case

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 baris(n, i int) {
    if n == i {
        fmt.Print(1, " ")
        return
    }
    fmt.Print(n, " ")
    baris(n-1, i)
    fmt.Print(n, " ")
}

func main() {
    var n int
    fmt.Print("Input (n): ")
    fmt.Scan(&n)
    baris(n, 1)
}
```

## Screenshot Output

```
PS D:\Pratikum Alpro 2\Laprak 5\Unguided 4> go run "d:\Pratikum Alpro 2\Laprak 5\Unguided 4\main.go"
Input (n): 5
5 4 3 2 1 2 3 4 5
PS D:\Pratikum Alpro 2\Laprak 5\Unguided 4> go run "d:\Pratikum Alpro 2\Laprak 5\Unguided 4\main.go"
Input (n): 9
9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9
PS D:\Pratikum Alpro 2\Laprak 5\Unguided 4> █
```

## Deskripsi Program

Program ini dirancang untuk menghitung dan menampilkan urutan angka domino berdasarkan input pengguna. Program ini memiliki dua base case: ketika  $n == i$ , yang digunakan untuk menghentikan penurunan nilai `n`, dan kemudian melanjutkan ke base case kedua yang mengembalikan urutan angka dari 1 hingga N. Berikut adalah langkah kerja dari fungsi ini:

- Fungsi dipanggil pertama kali sebagai `baris(5, 1)`
- Memasuki rekursi untuk menurunkan nilai `n` (`baris(n-1, i)`)

- Mencetak 5, lanjut ke baris(4, 1)
- Mencetak 4, lanjut ke baris(3, 1)
- Mencetak 3, lanjut ke baris(2, 1)
- Mencetak 2, lanjut ke baris(1, 1)
- Ketika mencapai base case `n == i`, angka 1 dicetak, dan rekursi berhenti menurun, kemudian memulai pengembalian nilai.
- Kembali ke baris(2, 5) dan mencetak 2 lagi.
- Kembali ke baris(3, 5) dan mencetak 3 lagi.
- Kembali ke baris(4, 5) dan mencetak 4 lagi.
- Kembali ke baris(5, 5) dan mencetak 5 lagi. Output akhir untuk N = 5 adalah: `5 4 3 2 1 2 3 4 5`.

Dapat disimpulkan bahwa fungsi ini mencetak urutan angka menurun dari N ke 1, lalu meningkat kembali dari 1 ke N.

## 5. Soal Studi Case

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 bilGanjil(n, i int) {
    if i > n {
        return
    }
    if i%2 != 0 {
```

```

        fmt.Print(i, " ")
    }

    bilGanjil(n, i+1)
}

func main() {
    var n int

    fmt.Print("Input bilangan (n): ")
    fmt.Scan(&n)
    bilGanjil(n, 1)
}

```

## Screenshoot Output

```

PS D:\Pratikum Alpro 2\Laprak 5\Unguided 5> go run "d:\Pratikum Alpro 2\Laprak 5\Unguided 5\main.go"
Input bilangan (n): 5
1 3 5
PS D:\Pratikum Alpro 2\Laprak 5\Unguided 5> go run "d:\Pratikum Alpro 2\Laprak 5\Unguided 5\main.go"
Input bilangan (n): 20
1 3 5 7 9 11 13 15 17 19
PS D:\Pratikum Alpro 2\Laprak 5\Unguided 5>

```

## Deskripsi Program

Program ini dibuat untuk mencari dan mencetak bilangan ganjil berdasarkan input dari pengguna. Program ini menggunakan fungsi rekursif yang mengulangi prosesnya hingga mencapai kondisi tertentu, yaitu base case. Dalam program ini, base case terjadi ketika  $i > n$ , yang akan menghentikan rekursi. Berikut adalah tahapan kerja dari fungsi ini:

- Rekursif pada bilGanjil (n, i+1)
- Input dimulai dengan (5, 1), yang akan diproses sebagai berikut:
- bilGanjil(5, 1) mencetak angka karena 1 adalah bilangan ganjil.
- Fungsi terus berlanjut, mencetak bilangan ganjil berikutnya hingga mencapai kondisi base case di mana  $i > n$ , misalnya pada bilGanjil(5, 6), di mana fungsi tidak akan melanjutkan rekursif.

Program akan berhenti setelah semua bilangan ganjil dalam rentang yang diberikan dicetak.

## 6. Soal Studi Case

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

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

### Screenshoot Output

```
PS D:\Pratikum Alpro 2\Laprak 5\Unguided 6> go run "d:\Pratikum Alpro 2\Laprak 5\Unguided 6\main.go"
Masukkan Bilangan(x, y) : 2 2
Hasil 2^2 : 4
PS D:\Pratikum Alpro 2\Laprak 5\Unguided 6> go run "d:\Pratikum Alpro 2\Laprak 5\Unguided 6\main.go"
Masukkan Bilangan(x, y) : 5 3
Hasil 5^3 : 125
PS D:\Pratikum Alpro 2\Laprak 5\Unguided 6> █
```



## Deskripsi Program

Program ini dirancang untuk menghitung perpangkatan  $x^y$  menggunakan fungsi rekursif. Fungsi ini memiliki pengembalian nilai *return*, berdasarkan input  $x$  dan  $y$ . Base case fungsi ini adalah ketika  $y == 0$ , yang akan mengembalikan nilai  $x$  sebanyak 1 kali. Jika  $y \neq 0$ , fungsi akan memanggil dirinya sendiri dengan mengurangi  $y$  menjadi  $(y - 1)$  hingga mencapai base case ( $y == 0$ ), sambil mengalikan hasil dari setiap langkah rekursif dengan  $x$ . Ini menghasilkan penghitungan bertahap dari  $x$  yang dipangkatkan hingga  $x^y$ .

#### **IV. REFERENSI**

- [1] Digimensia, "Understanding Recursive Function in Golang," *Digimensia*, 2023. <https://digimensia.com/golang-recursive-function/>. Diakses pada 03 Nopember 2024