

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL VI  
REKURSIF**



**Disusun Oleh :**

**Wafiq Nur Azizah / 2311102270**

**S1IF-11-05**

**Dosen Pengampu :**

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

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## I. DASAR TEORI

Rekursi adalah konsep di mana sebuah fungsi memanggil dirinya sendiri, baik secara langsung maupun tidak langsung. Setiap panggilan yang dilakukan oleh fungsi rekursif adalah versi yang lebih sederhana dari masalah yang sama, sehingga akan mengarah pada titik penyelesaian. Dalam rekursi, penting untuk memiliki kasus dasar, yaitu kondisi yang menandai akhir dari eksekusi fungsi dan menghentikan panggilan berikutnya. Dalam bahasa pemrograman Go, fungsi dapat bersifat rekursif jika ia memanggil dirinya sendiri dan memiliki kondisi yang jelas untuk menghentikan proses tersebut.

Berikut ini ada contoh faktorial suatu bilangan menggunakan rekursif:

### Sourcecode

```
package main
import "fmt"

func factorial (num int) int {

    // condition to break recursion
    if num == 0 {
        return 1
    } else {
        // condition for recursion call
        return num * factorial (num-1)
    }
}

func main() {
    var number = 3

    // function call
    var result = factorial (number)

    fmt.Println("The factorial of 3 is", result)
}
```

### Keluaran:

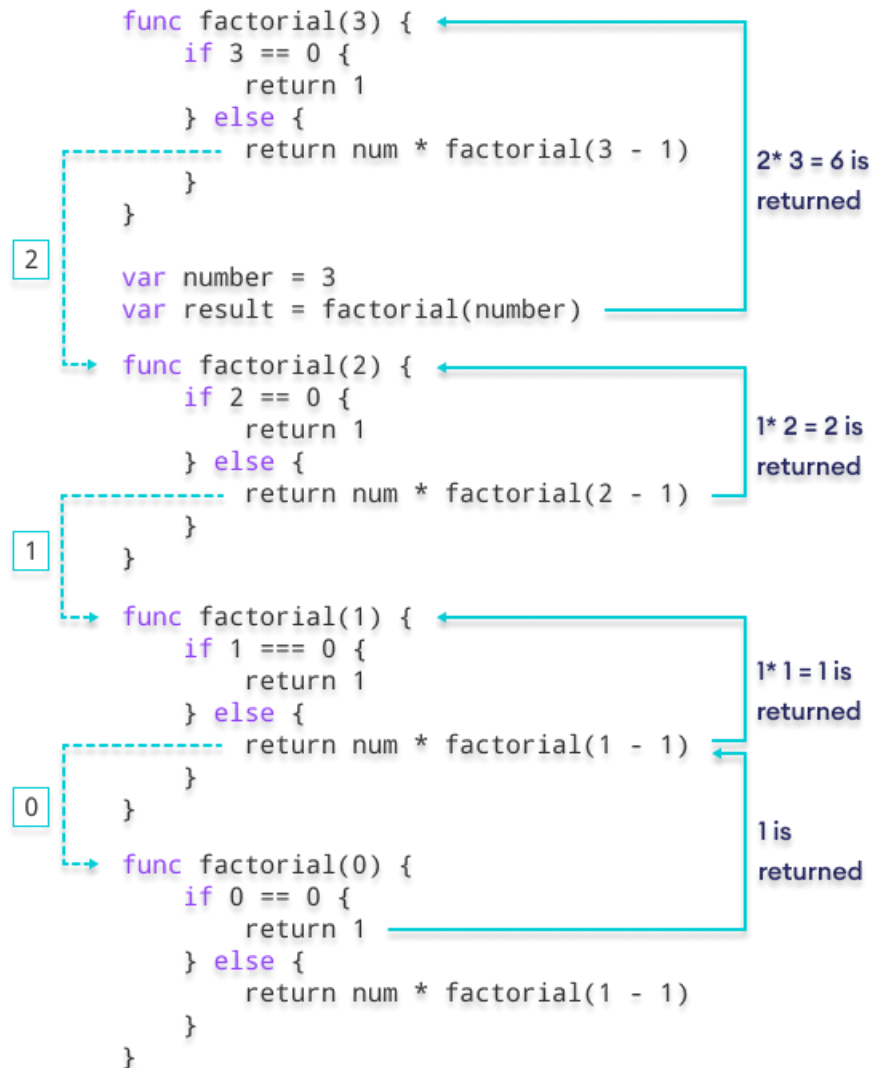
```
Faktorial dari 3 adalah 6
```

Dalam contoh di atas, telah dibuat fungsi rekursif bernama 'factorial()' yang memanggil dirinya sendiri jika nilai 'nomor' tidak sama dengan 0 .

```
return num * factorial(num - 1)
```

Pada setiap panggilan, program mengurangi nilai 'nomor' oleh 1 .

Berikut cara kerja dari program di atas, yaitu:



Terdapat berbagai jenis rekursi yang dapat dijelaskan sebagai berikut:

1. Rekursi Langsung

Ini adalah jenis rekursi dimana sebuah fungsi memanggil dirinya sendiri secara langsung tanpa melibatkan fungsi lain. Contohnya adalah fungsi yang menghitung faktorial dengan memanggil dirinya sendiri untuk menghitung faktorial dari nilai yang lebih kecil.

2. Rekursi Tidak Langsung

Dalam rekursi ini, satu fungsi memanggil fungsi lain dan fungsi yang dipanggil itu pada gilirannya memanggil fungsi awal. Oleh karena itu, fungsi yang memanggil dirinya sendiri secara tidak langsung melalui fungsi lain ini membutuhkan dukungan dari fungsi lain untuk mencapai hasil.

3. Rekursi Ekor

Pada jenis ini, panggilan ke fungsi yang sama dilakukan sebagai langkah terakhir atau final dari suatu fungsi. Ketika fungsi mengakhiri eksekusi dengan memanggil dirinya sendiri, maka fungsi tersebut disebut tail-recursive. Rekursi ini efisien karena tidak menyimpan status panggilan sebelumnya dalam tumpukan, sehingga dapat mengurangi penggunaan memori.

4. Rekursi Kepala

Di rekursi ini, pemanggilan rekursif menjadi langkah pertama dalam fungsi. Tidak ada pernyataan lain yang dijalankan sebelum panggilan tersebut, dan semua operasi yang diperlukan dilakukan pada saat fungsi mengembalikan hasilnya. Ini berarti bahwa fungsi harus menyelesaikan semua proses sebelum akhirnya kembali ke pemanggil.

5. Rekursi Tak Terbatas

Jenis ini adalah kebalikan dari fungsi rekursif yang pasti. Rekursi tak terbatas terjadi ketika fungsi terus memanggil dirinya sendiri tanpa pernah mencapai kondisi dasar yang menghentikannya. Akibatnya, jenis rekursi ini dapat menyebabkan sistem mengalami kegagalan atau menyebabkan kelebihan memori.

6. Rekursi Fungsi Anonim

Dalam bahasa pemrograman Go, terdapat konsep fungsi anonim, yaitu fungsi yang tidak memiliki nama. Rekursi juga dapat diterapkan menggunakan fungsi anonim di Go. Fungsi ini sering digunakan untuk menyelesaikan tugas-tugas tertentu di dalam program tanpa perlu mendefinisikan fungsi dengan nama eksplisit, sehingga memberikan fleksibilitas dan kemudahan dalam penulisan kode.

## II. GUIDED

### 1. Soal Studi Case

Membuat baris bilangan dari n hingga 1

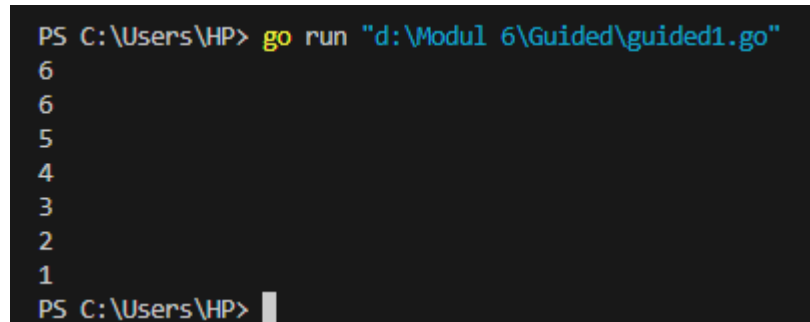
Base-case: bilangan == 1

#### 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 C:\Users\HP> go run "d:\Modul 6\Guided\guided1.go"
6
6
5
4
3
2
1
PS C:\Users\HP>
```

#### Deskripsi Program

Program Go di atas menggunakan rekursif untuk mencetak angka dari nilai input n hingga 1 dengan urutan menurun. Pada fungsi 'main', pengguna memasukkan nilai 'n', lalu fungsi 'baris' dipanggil untuk menampilkan urutan angka tersebut. Fungsi 'baris' memanggil dirinya sendiri secara berulang dengan mengurangi nilai bilangan satu per satu hingga mencapai 1. Ketika bilangan sama dengan 1, fungsi berhenti memanggil dirinya sendiri dan mencetak angka 1. Program ini menunjukkan cara pemanggilan fungsi secara berulang hingga suatu kondisi tertentu terpenuhi.

## 2. Soal Studi Case

Menghitung hasil penjumlahan 1 hingga n

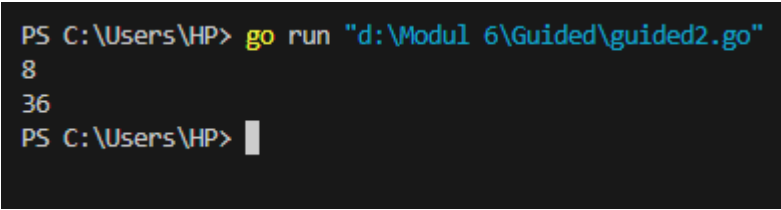
Base-case:  $n == 1$

### Sourcecode

```
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



```
PS C:\Users\HP> go run "d:\Modul 6\Guided\guided2.go"
8
36
PS C:\Users\HP> 
```

### Deskripsi Program

Program Go di atas menunjukkan cara kerja rekursi dalam fungsi `penjumlahan` yang digunakan untuk menghitung total angka dari 1 sampai `n`. Fungsi `penjumlahan` memeriksa apakah `n` bernilai 1. Jika iya, fungsi akan mengembalikan nilai 1 sebagai tanda berhenti. Jika tidak, fungsi akan memanggil dirinya sendiri dengan nilai `n-1` dan menambahkan `n` saat ini. Proses ini berulang hingga mencapai kondisi awal sehingga menghasilkan total penjumlahan dari 1 hingga `n`. Di dalam fungsi `main`, pengguna memasukkan angka `n` dan hasil penjumlahan tersebut ditampilkan menggunakan `fmt.Println`.

### 3. Soal Studi Case

Mencari dua pangkat  $n$  atau  $2^n$

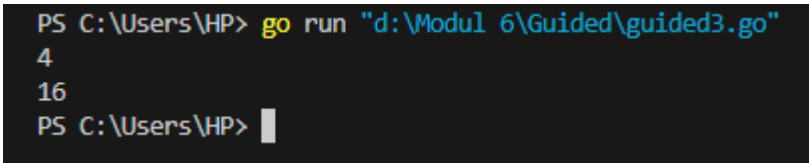
Base-case:  $n == 0$

#### Sourcecode

```
package main
import "fmt"
func main(){
    var n int
    fmt.Scan(&n)
    fmt.Println(pangkat(n))
}

func pangkat(n int) int {
    if n == 0 {
        return 1
    }else{
        return 2 * pangkat(n-1)
    }
}
```

#### Screenshoot Output



```
PS C:\Users\HP> go run "d:\Modul 6\Guided\guided3.go"
4
16
PS C:\Users\HP>
```

#### Deskripsi Program

Program Go di atas menggunakan metode rekursi untuk menghitung pangkat dua dari bilangan bulat yang dimasukkan. Fungsi `pangkat` menerima sebuah angka `n` dan memeriksa apakah `n` sama dengan 0. Jika benar, fungsi akan mengembalikan 1 karena 2 pangkat 0 adalah 1. Jika tidak, fungsi akan mengalikan 2 dengan hasil pemanggilan fungsi `pangkat` dengan `n` dikurangi 1. Dengan cara ini, fungsi menghitung pangkat dua secara bertahap hingga mencapai nilai dasar. Contoh ini menunjukkan bagaimana rekursi bisa digunakan untuk menyelesaikan perhitungan yang berulang dengan cara yang sederhana dan teratur.

#### 4. Soal Studi Case

Mencari nilai faktorial atau  $n!$

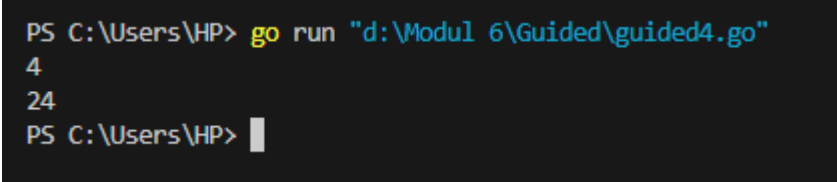
Base-case:  $n == 0$  atau  $n == 1$

##### Sourcecode

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

func faktorial(n int) int {
    if n == 0 || n == 1 {
        return 1
    }else{
        return n * faktorial(n-1)
    }
}
```

##### Screenshoot Output



```
PS C:\Users\HP> go run "d:\Modul 6\Guided\guided4.go"
4
24
PS C:\Users\HP> 
```

##### Deskripsi Program

Program Go di atas digunakan untuk menghitung faktorial dari bilangan bulat positif dengan cara rekursif. Fungsi `faktorial` akan mengembalikan 1 jika input `n` adalah 0 atau 1 karena faktorial dari kedua angka tersebut adalah 1. Jika `n` lebih besar dari 1, fungsi ini akan memanggil dirinya sendiri dengan mengurangi nilai `n` satu per satu hingga mencapai angka dasar. Jadi, fungsi ini mengalikan nilai `n` dengan hasil faktorial dari `n-1`.



### 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() {
    var n int

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

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

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

## Screenshoot Output

```
PS C:\Users\HP> go run "d:\Modul 6\Unguided\unguided1.go"
Masukkan nilai n: 10
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 C:\Users\HP> 
```

## Deskripsi Program

Program Go di atas menggunakan metode rekursif untuk menghitung deret Fibonacci, dimana setiap suku dihitung dengan menjumlahkan dua suku sebelumnya. Fungsi `fibonacci(n int)` memanggil dirinya sendiri untuk mencari nilai suku ke-n dengan kondisi dasar yang mengatasi kasus ketika 'n' adalah 0 atau 1. Dalam fungsi `main()`, pengguna diminta untuk memasukkan nilai 'n' dan program akan menampilkan deret Fibonacci dari suku 0 hingga suku ke-n.

## 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 cetakBintang(n int, current int) {
    if current > n {
        return
    }

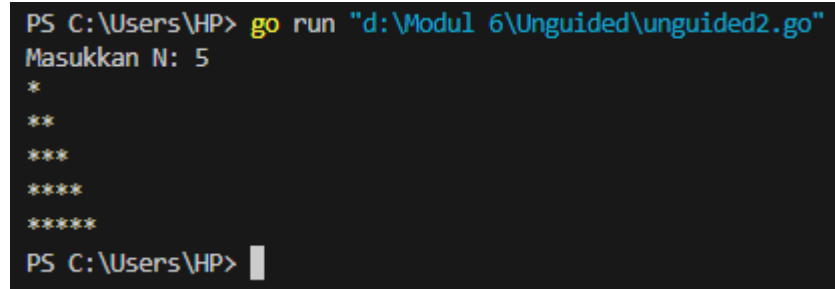
    for i := 0; i < current; i++ {
        fmt.Print("*")
    }
    fmt.Println()

    cetakBintang(n, current+1)
}

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

    cetakBintang(N, 1)
}
```

## Screenshoot Output



```
PS C:\Users\HP> go run "d:\Modul 6\Unguided\unguided2.go"
Masukkan N: 5
*
**
***
****
*****
PS C:\Users\HP> |
```

## Deskripsi Program

Program go di atas menggunakan teknik rekursif untuk mencetak pola bintang berbentuk segitiga. Fungsi `cetakBintang` memiliki dua parameter yaitu `n` yang menentukan jumlah baris bintang yang ingin dicetak dan `current` yang menunjukkan baris saat ini. Proses dimulai dari baris pertama (dengan `current` = 1) dan terus berlanjut hingga mencapai baris ke- $n$ . Setiap kali fungsi dipanggil, program mencetak bintang sebanyak nilai `current` dan kemudian memanggil dirinya sendiri dengan menambah nilai `current` satu. Ketika `current` melebihi `n`, fungsi akan berhenti. Dengan cara ini, program dapat menghasilkan pola bintang yang teratur tanpa menggunakan loop biasa.

### 3. Soal Studi Case

Buatlah program yang mengimplementasikan rekursif untuk menampilkan faktor bilangan dari suatu N atau bilangan yang apa saja 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 cariFaktor(n int, current int) {
    if current > n {
        return
    }
    if n%current == 0 {
        fmt.Print(current, " ")
    }
    cariFaktor(n, current+1)
}

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

    fmt.Print("Faktor dari ", N, ": ")
    cariFaktor(N, 1)
    fmt.Println()
}
```

## Screenshoot Output

```
PS C:\Users\HP> go run "d:\Modul 6\Unguided\unguided3.go"
Masukkan N: 12
Faktor dari 12: 1 2 3 4 6 12
PS C:\Users\HP> go run "d:\Modul 6\Unguided\unguided3.go"
Masukkan N: 5
Faktor dari 5: 1 5
PS C:\Users\HP> 
```

## Deskripsi Program

Program Go di atas menunjukkan bagaimana menggunakan metode rekursif untuk mencari dan mencetak faktor-faktor dari sebuah bilangan bulat positif. Fungsi `cariFaktor` memiliki dua parameter yaitu, `n` adalah bilangan yang ingin dicari faktornya dan `current` adalah angka yang sedang diperiksa untuk melihat apakah angka tersebut merupakan faktor dari `n`. Program mulai memeriksa dari angka 1 dan terus berlanjut hingga `current` lebih besar dari `n`. Jika `current` adalah faktor dari `n` (artinya sisa pembagian `n` dengan `current` adalah nol), maka angka itu akan dicetak. Dengan cara rekursif ini, fungsi memanggil dirinya sendiri dengan nilai `current` yang bertambah sehingga semua faktor dapat ditemukan dan ditampilkan outputnya.

#### 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 printSequence(n int) {
    if n < 1 {
        return
    }
    fmt.Print(n, " ")
    if n > 1 {
        printSequence(n - 1)
        fmt.Print(n, " ")
    }
}

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

    printSequence(N)
    fmt.Println()
}
```

## Screenshoot Output

```
PS C:\Users\HP> go run "d:\Modul 6\Unguided\unguided4.go"
Masukkan N: 5
5 4 3 2 1 2 3 4 5
PS C:\Users\HP> go run "d:\Modul 6\Unguided\unguided4.go"
Masukkan N: 9
9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9
PS C:\Users\HP> █
```

## Deskripsi Program

Program Go di atas menggunakan teknik rekursif melalui fungsi `printSequence` yang mencetak urutan angka mulai dari `N` hingga 1, lalu mencetak kembali angka dari 1 hingga `N`. Program ini dimulai dengan meminta pengguna untuk memasukkan nilai `N`. Jika `N` lebih besar dari 0, program akan mencetak angka tersebut, lalu memanggil fungsi itu sendiri dengan `N-1` untuk melanjutkan mencetak angka berikutnya. Setelah mencapai angka 1, program akan kembali mencetak angka-angka yang sebelumnya dilewati dalam urutan naik hingga akhirnya mencetak outputnya.



## 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 sebuah 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 printOddNumbers(n int, current int) {
    if current > n {
        return
    }
    fmt.Print(current, " ")
    printOddNumbers(n, current+2)
}

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

    if N > 0 {
        printOddNumbers(N, 1)
        fmt.Println()
    } else {
        fmt.Println("N harus bilangan bulat positif.")
    }
}
```

## Screenshoot Output

```
PS C:\Users\HP> go run "d:\Modul 6\Unguided\unguided5.go"
Masukkan N: 5
1 3 5
PS C:\Users\HP> go run "d:\Modul 6\Unguided\unguided5.go"
Masukkan N: 20
1 3 5 7 9 11 13 15 17 19
PS C:\Users\HP> 
```

## Deskripsi Program

Program Go di atas menggunakan metode rekursif untuk mencetak semua bilangan ganjil dari 1 hingga N dimana N diinput oleh pengguna. Fungsi `printOddNumbers` memiliki dua parameter, yaitu `n` sebagai batas maksimal dan `current` sebagai bilangan ganjil yang sedang dicetak. Jika nilai `current` melebihi N, fungsi akan berhenti. Jika tidak, fungsi akan mencetak nilai `current` diikuti dengan spasi, lalu memanggil dirinya sendiri dengan menambahkan 2 pada `current`. Metode ini membuat program dapat mencetak semua bilangan ganjil dengan cara yang efisien. Di bagian `main`, program juga memastikan bahwa N yang dimasukkan adalah bilangan bulat positif sebelum menjalankan fungsi rekursif tersebut.

## 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 angka(x int, y int) int {
    if y == 0 {
        return 1
    }
    if y < 0 {
        return 1 / angka(x, -y)
    }
    return x * angka(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 := angka(x, y)
    fmt.Printf("Hasil dari %d dipangkatkan %d adalah %d\n", x, y, result)
}
```

## Screenshoot Output

```
PS C:\Users\HP> go run "d:\Modul 6\Unguided\unguided6.go"
Masukkan bilangan bulat x: 2
Masukkan bilangan bulat y: 2
Hasil dari 2 dipangkatkan 2 adalah 4
PS C:\Users\HP> go run "d:\Modul 6\Unguided\unguided6.go"
Masukkan bilangan bulat x: 5
Masukkan bilangan bulat y: 3
Hasil dari 5 dipangkatkan 3 adalah 125
PS C:\Users\HP> █
```

## Deskripsi Program

Program Go di atas adalah contoh penggunaan fungsi rekursif untuk menghitung pangkat suatu bilangan bulat  $x$  yang dipangkatkan oleh  $y$ . Fungsi `angka` menerima dua parameter, yaitu  $x$  dan  $y$  dan menggunakan metode rekursif untuk melakukan perhitungan. Jika  $y$  sama dengan 0, fungsi ini akan mengembalikan nilai 1 karena setiap bilangan yang dipangkatkan dengan 0 adalah 1. Jika  $y$  bernilai negatif, fungsi akan memanggil dirinya sendiri dengan nilai  $y$  dan membalikkan hasilnya untuk mendapatkan nilai yang benar. Dalam kasus umum, fungsi akan mengalikan  $x$  dengan hasil dari pemanggilan rekursif `angka( $x$ ,  $y-1$ )`, secara bertahap mengurangi nilai  $y$  sampai mencapai 0.

Program ini juga memiliki fungsi `main` yang meminta input dari pengguna dan menampilkan hasil perhitungan dengan format yang jelas. Beberapa fungsi penting dalam program ini adalah `fmt.Print` dan `fmt.Scan` yang digunakan untuk berinteraksi dengan pengguna. Selain itu, ada fungsi `fmt.Printf` untuk menampilkan hasil dengan cara yang diinginkan.

#### IV. DAFTAR PUSTAKA

- GeeksforGeeks. (n.d.). Different types of recursion in Golang. Diakses 3 November 2024, dari <https://www.geeksforgeeks.org/different-types-of-recursion-in-golang/>
- Programiz. (n.d.). Recursion in Go (Golang). Diakses 3 November 2024, dari <https://www.programiz.com/golang/recursion>
- W3Schools. (n.d.). Fungsi rekursi dalam Go. Diakses 3 November 2024, dari [https://www.w3schools.com/go/go\\_function\\_recursion.php](https://www.w3schools.com/go/go_function_recursion.php)