

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 6  
REKURSIF**



**Disusun Oleh :**

**ANDIKA NEVIANTORO / 2311102167**

**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

Rekursif adalah sebuah teknik di pemrograman, di mana sebuah fungsi memanggil dirinya sendiri. Mirip dengan loop (pengulangan), yang kedua tujuannya memanggil aksi yang sama berkali-kali.

Namun rekursi sering digunakan untuk menyelesaikan masalah yang lebih kompleks:

- yang tidak bisa (sulit) diselesaikan dengan loop biasa
- atau kode implementasinya akan sangat sulit dibaca jika menggunakan loop (iterasi)

Contoh sederhana fungsi rekursif yang memanggil dirinya sendiri

```
function show(image) {  
  show(image); // <- memanggil dirinya sendiri "show"  
}
```

Contoh di atas tidaklah lengkap, karena tidak ada kondisi yang menghentikan pemanggilan fungsi. **Jika kamu menjalankan kode di atas, terjadilah “infinite loop”.**

Infinite loop adalah kondisi ketika program terus menerus melakukan fungsi yang sama tanpa ada kasus yang menghentikannya, yang tentunya terjadi error, stack overflow, atau crash.

Rekursi dan loop (iterasi) adalah dua teknik yang sering digunakan untuk melakukan aksi yang sama berkali-kali.

**Jika kamu bisa menyelesaikan program dengan loop, maka gunakan loop saja. Karena loop lebih sederhana dan lebih efisien.**

Secara memori dan performa, **rekursi lebih mahal dibanding loop**. Karena setiap pemanggilan fungsi akan menambahkan data ke dalam stack, dibanding langsung mengeksekusinya.

Namun, ada beberapa kasus seperti yang disebutkan di atas, di mana rekursi bisa digunakan untuk menyelesaikan masalah yang lebih kompleks, yang tidak bisa atau sulit diselesaikan dengan loop biasa.

Atau terkadang jika harus menggunakan loop biasa, maka kodenya jadi sulit dibaca. Maka, rekursi bisa digunakan untuk membuat kode lebih mudah dibaca.

Selalu ada tradeoff dalam dunia programming.

## II. GUIDED

1. Membuat baris bilangan dari n hingga 1

### Sourcecode

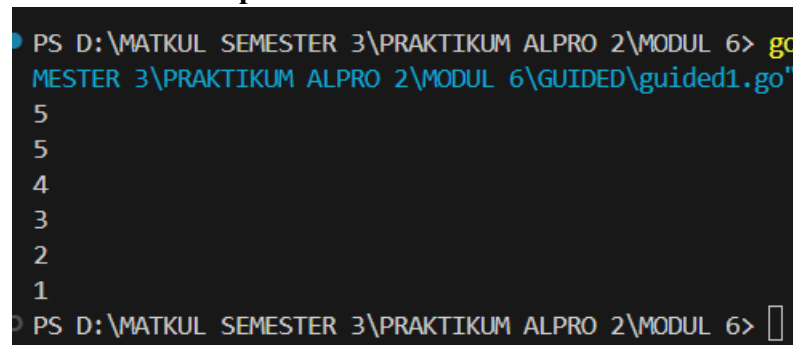
```
package main

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:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run MESTER 3\PRAKTIKUM ALPRO 2\MODUL 6\GUIDED\guided1.go
5
4
3
2
1
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> █
```

### Deskripsi Program :

Program di atas merupakan implementasi rekursif sederhana dalam bahasa Go yang menampilkan deret bilangan menurun dari `n` hingga 1. Program dimulai dengan membaca input bilangan bulat `n` menggunakan `fmt.Scan`. Fungsi utama `baris` dipanggil dengan parameter `n`, dan fungsi tersebut akan mencetak nilai `n`, kemudian memanggil dirinya sendiri dengan parameter yang dikurangi 1. Proses ini berlanjut hingga bilangan mencapai 1, yang menjadi kondisi dasar untuk menghentikan rekursi, di mana program mencetak angka `1` dan rekursi berhenti.

2. Menghitung hasil penjumlahan 1 hingga n

### Sourcecode

```

package main

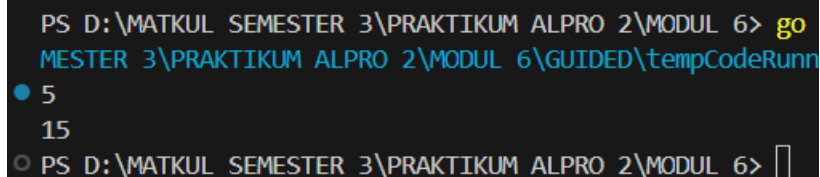
import "fmt"

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:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go
MESTER 3\PRAKTIKUM ALPRO 2\MODUL 6\GUIDED\tempCodeRunn
● 5
  15
○ PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> 

```

#### Deskripsi Program :

Program di atas merupakan implementasi rekursif dalam bahasa Go untuk menghitung jumlah total dari bilangan 1 hingga 'n'. Program pertama kali menerima input bilangan bulat 'n' melalui 'fmt.Scan'. Fungsi 'penjumlahan(n)' bekerja secara rekursif dengan menambahkan nilai 'n' ke hasil dari pemanggilan fungsi yang sama dengan parameter 'n-1'. Proses ini berlanjut hingga mencapai kondisi dasar di mana 'n' sama dengan 1, yang akan mengembalikan nilai 1 dan menghentikan rekursi. Hasil dari fungsi rekursif kemudian ditampilkan menggunakan 'fmt.Println'.

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

#### Sourcecode

```

package main

import "fmt"

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

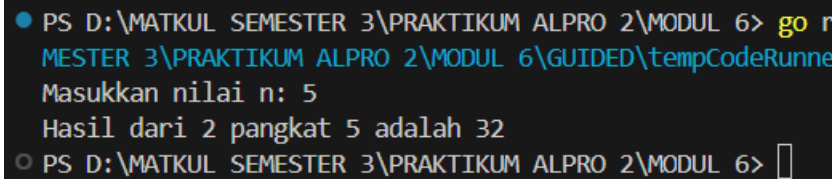
```

```

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

```

### Screenshot Output:



```

● PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run ...
MESTER 3\PRAKTIKUM ALPRO 2\MODUL 6\GUIDED\tempCodeRunne
Masukkan nilai n: 5
Hasil dari 2 pangkat 5 adalah 32
○ PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> █

```

### Deskripsi Program:

Program di atas merupakan implementasi rekursif dalam bahasa Go untuk menghitung hasil pangkat dua, yaitu  $2^n$ . Program menerima input bilangan bulat `n` melalui `fmt.Scanln`. Fungsi `pangkat(n)` bekerja secara rekursif dengan mengalikan 2 dengan hasil dari pemanggilan fungsi `pangkat(n-1)` hingga mencapai kondisi dasar, yaitu ketika `n == 0`, yang akan mengembalikan nilai 1 (karena  $2^0 = 1$ ). Setelah fungsi rekursif selesai, hasil pangkat tersebut dicetak menggunakan `fmt.Println` bersama dengan informasi hasil perhitungan  $2^n$ .

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

##### Sourcecode

```

package main

import "fmt"

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

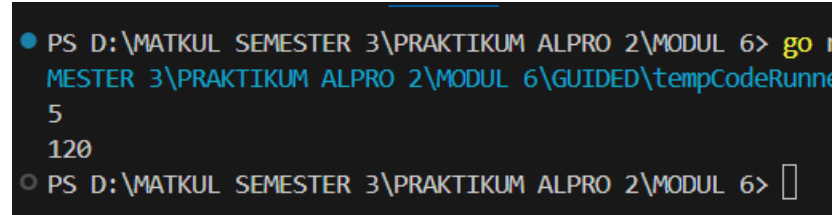
func main () {

    fmt. Scan(&n)

```

```
    fmt.Println(factorial(n))  
}
```

### Screenshot Output:



```
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run tempCodeRunner.go  
5  
120  
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6>
```

### Deskripsi Program:

Program di atas adalah implementasi rekursif dalam bahasa Go untuk menghitung faktorial dari sebuah bilangan bulat `n`. Variabel `n` dideklarasikan secara global, dan input nilainya diterima melalui `fmt.Scan`. Fungsi `factorial(n)` bekerja dengan memeriksa kondisi dasar, yaitu jika `n` bernilai 0 atau 1, fungsi mengembalikan 1 (karena  $0! = 1! = 1$ ). Jika tidak, fungsi mengalikan `n` dengan hasil dari pemanggilan rekursif `factorial(n-1)`, yang terus berlanjut hingga mencapai kondisi dasar. Hasil perhitungan faktorial kemudian dicetak menggunakan `fmt.Println`.

## 2. UNGUIDED

1. 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.

### Sourcecode

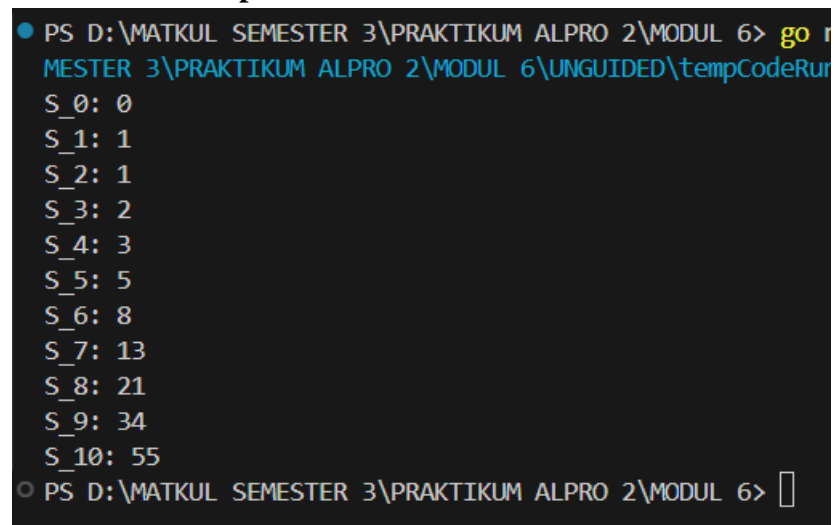
```
package main

import (
    "fmt"
)

// Fungsi rekursif untuk menghitung nilai Fibonacci
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() {
    // Menampilkan deret Fibonacci hingga suku ke-10
    for i := 0; i <= 10; i++ {
        fmt.Printf("S_%d: %d\n", i, fibonacci(i))
    }
}
```

### Screenshoot Output :



```
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run tempCodeRunner.go
S_0: 0
S_1: 1
S_2: 1
S_3: 2
S_4: 3
S_5: 5
S_6: 8
S_7: 13
S_8: 21
S_9: 34
S_10: 55
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> █
```

**Deskripsi Program :**

Program di atas adalah implementasi rekursif dalam bahasa Go untuk menghitung dan menampilkan deret Fibonacci hingga suku ke-10. Fungsi `fibonacci(n)` menghitung nilai Fibonacci dari bilangan `n` dengan dua kondisi dasar: jika `n == 0`, fungsi mengembalikan 0, dan jika `n == 1`, fungsi mengembalikan 1. Untuk nilai lainnya, fungsi mengembalikan jumlah dari dua nilai sebelumnya dalam deret Fibonacci, yaitu `fibonacci(n-1)` dan `fibonacci(n-2)`. Dalam fungsi `main`, program mencetak nilai Fibonacci dari suku ke-0 hingga ke-10 menggunakan loop `for`, dengan setiap suku ditampilkan dalam format yang rapi menggunakan `fmt.Printf`.

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

**Sourcecode**

```
package main

import (
    "fmt"
)

// Fungsi rekursif untuk mencetak baris bintang
func printStars(n int) {
    if n > 0 {
        printStars(n - 1)
        fmt.Print("*")
    }
}

// Fungsi rekursif untuk mencetak pola bintang
func printPattern(n int, current int) {
    if current <= n {
        printStars(current)
        fmt.Println()
        printPattern(n, current + 1)
    }
}

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

    printPattern(n, 1)
}
```

**Screenshoot Output :**



```
● PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run
MESTER 3\PRAKTIKUM ALPRO 2\MODUL 6\UNGUIDED\tempCodeRunn
Masukkan nilai N: 5
*
**
***
****
*****

● PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run
MESTER 3\PRAKTIKUM ALPRO 2\MODUL 6\UNGUIDED\tempCodeRunn
Masukkan nilai N: 1
*

● PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run
MESTER 3\PRAKTIKUM ALPRO 2\MODUL 6\UNGUIDED\tempCodeRunn
Masukkan nilai N: 3
*
**
***

○ PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> █
```

### Deskripsi Program :

Program di atas adalah program Go yang menggunakan fungsi rekursif untuk mencetak pola bintang berdasarkan input pengguna. Fungsi `printStars` mencetak sejumlah bintang sesuai dengan parameter `n`, menggunakan rekursi untuk menurunkan nilai `n` hingga mencapai 0 sebelum mencetak bintang. Fungsi `printPattern` mencetak pola bintang secara bertahap, mulai dari 1 bintang hingga `n` bintang. Di dalam fungsi `main`, pengguna diminta untuk memasukkan nilai `N`, dan program kemudian memanggil `printPattern` untuk menghasilkan pola bintang. Setiap baris akan memiliki satu bintang lebih banyak daripada baris sebelumnya hingga mencapai jumlah bintang yang ditentukan.

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).

### Sourcecode :

```
package main

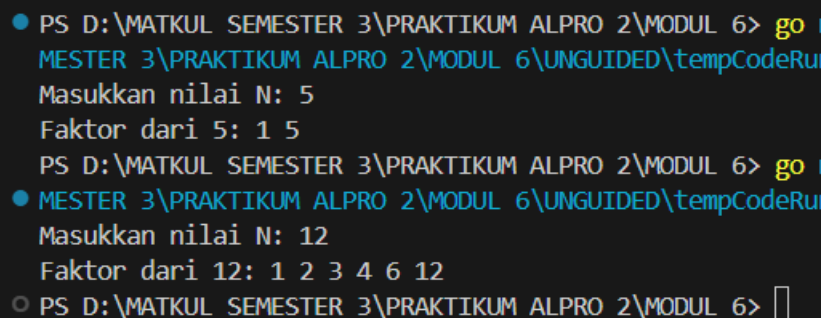
import (
    "fmt"
)
```

```
// Fungsi rekursif untuk mencari faktor bilangan
func printFactors(n int, current int) {
    if current > n {
        return
    }
    if n%current == 0 {
        fmt.Printf("%d ", current)
    }
    printFactors(n, current+1)
}

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

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

#### Screenshoot Output :



```
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run main.go
Masukkan nilai N: 5
Faktor dari 5: 1 5
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run main.go
Masukkan nilai N: 12
Faktor dari 12: 1 2 3 4 6 12
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> █
```

#### Deskripsi Program :

Program di atas adalah program Go yang menggunakan fungsi rekursif untuk mencetak faktor-faktor dari sebuah bilangan bulat yang dimasukkan oleh pengguna. Fungsi `printFactors` menerima dua parameter: `n`, yang merupakan bilangan yang akan dicari faktornya, dan `current`, yang digunakan untuk memeriksa angka dari 1 hingga `n`. Jika `current` adalah faktor dari `n` (yaitu, jika sisa pembagian `n` oleh `current` adalah 0), maka angka tersebut dicetak. Proses ini berlanjut hingga `current` lebih besar dari `n`. Di dalam fungsi `main`, pengguna diminta untuk memasukkan nilai `N`, dan setelah itu program akan memanggil `printFactors` untuk menampilkan semua faktor dari `N` di konsol.

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.

**Sourcecode**

```
package main

import "fmt"

// Fungsi rekursif untuk menampilkan angka dari N ke 1
func turun(n int) {
    if n == 1 {
        fmt.Printf("%d ", n)
    } else {
        fmt.Printf("%d ", n)
        turun(n - 1)
    }
}

// Fungsi rekursif untuk menampilkan angka dari 1 ke N
func naik(n, current int) {
    if current == n {
        fmt.Printf("%d ", current)
    } else {
        fmt.Printf("%d ", current)
        naik(n, current+1)
    }
}

// Fungsi utama untuk menampilkan barisan bilangan
func tampilkanBarisan(n int) {
    turun(n)
    naik(n, 2)
    fmt.Println() // baris baru setelah selesai
}

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

**Screenshot output:**

```

● PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go r
MESTER 3\PRAKTIKUM ALPRO 2\MODUL 6\UNGUIDED\tempCodeRun
Masukkan bilangan bulat positif: 5
5 4 3 2 1 2 3 4 5
● PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go r
MESTER 3\PRAKTIKUM ALPRO 2\MODUL 6\UNGUIDED\tempCodeRun
Masukkan bilangan bulat positif: 9
9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9
○ PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> █

```

### Deskripsi Program:

Program di atas adalah program Go yang menggunakan fungsi rekursif untuk menampilkan urutan angka dari `N` ke 1 dan kemudian dari 1 ke `N`. Fungsi `turun` mencetak angka mulai dari `n` hingga 1, dan menggunakan rekursi untuk menurunkan nilai `n` hingga mencapai 1. Fungsi `naik` mencetak angka mulai dari 1 hingga `n`, dengan parameter `current` yang bertambah hingga mencapai nilai `n`. Fungsi `tampilkanBarisan` menggabungkan kedua fungsi tersebut, memanggil `turun` terlebih dahulu diikuti oleh `naik`, dan menambahkan baris baru setelah semua angka dicetak. Di dalam fungsi `main`, pengguna diminta untuk memasukkan bilangan bulat positif, dan program akan memanggil `tampilkanBarisan` untuk menampilkan urutan angka sesuai dengan input tersebut.

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.

### Sourcecode

```

package main

import "fmt"

// Fungsi rekursif untuk menampilkan bilangan ganjil
func tampilkanGanjil(n, current int) {
    if current > n {
        return
    }
    fmt.Printf("%d ", current)
    tampilkanGanjil(n, current+2)
}

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

```

```

    fmt.Scanln(&n)
    tampilkanGanjil(n, 1)
    fmt.Println() // baris baru setelah selesai
}

```

### Screenshot Output:

```

● PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run
MESTER 3\PRAKTIKUM ALPRO 2\MODUL 6\UNGUIDED\tempCodeRunn
Masukkan bilangan bulat positif: 5
1 3 5
● PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run
MESTER 3\PRAKTIKUM ALPRO 2\MODUL 6\UNGUIDED\tempCodeRunn
Masukkan bilangan bulat positif: 20
1 3 5 7 9 11 13 15 17 19
○ PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> █

```

### Deskripsi Program:

Program di atas adalah program Go yang menggunakan fungsi rekursif untuk menampilkan bilangan ganjil dari 1 hingga `n`, di mana `n` adalah bilangan bulat positif yang dimasukkan oleh pengguna. Fungsi `tampilkanGanjil` menerima dua parameter: `n`, sebagai batas atas, dan `current`, yang dimulai dari 1. Fungsi ini mencetak nilai `current`, kemudian memanggil dirinya sendiri dengan menambahkan 2 pada `current`, sehingga hanya bilangan ganjil yang dicetak. Proses ini berlanjut hingga `current` melebihi `n`, pada titik mana fungsi akan berhenti. Di dalam fungsi `main`, pengguna diminta untuk memasukkan nilai, dan setelah itu program akan memanggil `tampilkanGanjil` untuk menampilkan semua bilangan ganjil hingga batas yang ditentukan, diakhiri dengan mencetak baris baru.

6. Buatlah program yang mengimplementasikan rekursif untuk mencari hasil pangkat dari dua gkat 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".

### Sourecode

```

package main

import "fmt"

// Fungsi rekursif untuk menghitung x^y
func pangkat(x, y int) int {
    // Basis: jika y == 0, maka x^0 = 1
}

```

```

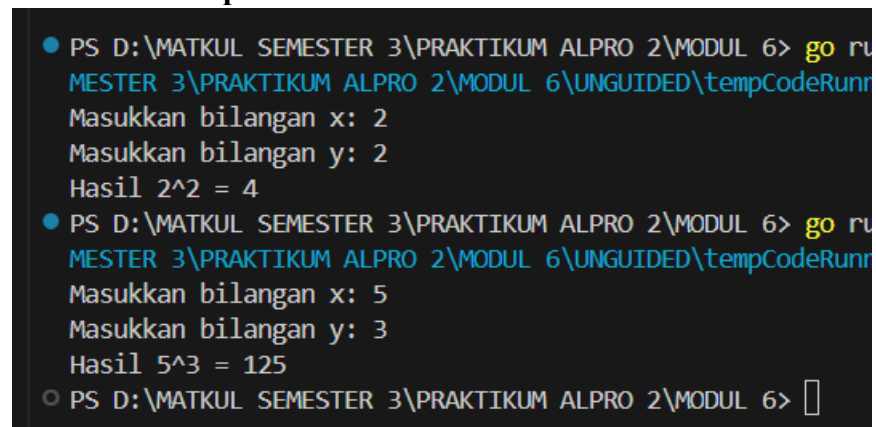
    if y == 0 {
        return 1
    }
    // Rekursi:  $x^y = x * x^{(y-1)}$ 
    return x * pangkat(x, y-1)
}

func main() {
    var x, y int
    fmt.Print("Masukkan bilangan x: ")
    fmt.Scanln(&x)
    fmt.Print("Masukkan bilangan y: ")
    fmt.Scanln(&y)

    hasil := pangkat(x, y)
    fmt.Printf("Hasil  $%d^{%d} = %d$ \n", x, y, hasil)
}

```

#### Screenshot Output:



```

● PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run main.go
MESTER 3\PRAKTIKUM ALPRO 2\MODUL 6\UNGUIDED\tempCodeRunner
Masukkan bilangan x: 2
Masukkan bilangan y: 2
Hasil 2^2 = 4
● PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6> go run main.go
MESTER 3\PRAKTIKUM ALPRO 2\MODUL 6\UNGUIDED\tempCodeRunner
Masukkan bilangan x: 5
Masukkan bilangan y: 3
Hasil 5^3 = 125
○ PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 6>

```

#### Deskripsi Program:

Program di atas adalah program Go yang menggunakan fungsi rekursif untuk menghitung hasil dari  $x^y$  ( $x$  pangkat  $y$ ), di mana  $x$  dan  $y$  adalah bilangan bulat yang dimasukkan oleh pengguna. Fungsi `pangkat` melakukan perhitungan dengan menggunakan dua kasus: pertama, jika  $(y)$  sama dengan 0, maka fungsi mengembalikan 1 sesuai dengan sifat matematis bahwa  $x^0 = 1$ . Jika tidak, fungsi akan menghitung  $x^y$  dengan cara mengalikan  $(x)$  dengan hasil dari pemanggilan fungsi `pangkat` dengan  $(y)$  dikurangi 1, sehingga fungsi akan terus memanggil dirinya sendiri hingga mencapai basis. Di dalam fungsi `main`, pengguna diminta untuk memasukkan nilai  $x$  dan  $y$ , dan hasil perhitungan  $x^y$  kemudian dicetak ke layar.

## **KESIMPULAN**

Kesimpulan dari penjelasan tentang rekursi adalah bahwa rekursi adalah teknik pemrograman di mana sebuah fungsi memanggil dirinya sendiri untuk menyelesaikan masalah, sering kali digunakan untuk kasus yang lebih kompleks yang sulit diselesaikan dengan loop biasa. Meskipun rekursi dapat meningkatkan keterbacaan kode, terutama untuk algoritma yang rumit, penggunaannya datang dengan biaya performa dan memori yang lebih tinggi, karena setiap pemanggilan fungsi menambah data ke dalam stack. Oleh karena itu, jika sebuah masalah dapat diselesaikan dengan loop, biasanya lebih efisien untuk menggunakan loop. Namun, dalam situasi di mana rekursi dapat menyederhanakan implementasi dan meningkatkan keterbacaan, rekursi bisa menjadi pilihan yang lebih baik. Pada akhirnya, pemilihan antara rekursi dan loop tergantung pada konteks masalah yang dihadapi dan tradeoff antara efisiensi dan keterbacaan kode.

## **REFERENSI**

[1] SkoDev, Apa itu Rekursif: pengertian dan informasi detail.

<https://sko.dev/wiki/rekursif>