

**LAPORAN**  
**PRAKTIKUM**  
**ALGORITMA DAN PEMROGRAMAN 2**  
**MODUL 5**  
**REKURSIF**



Oleh:

NAMA: MOHAMMAD REYHAN ARETHA FATIN

NIM: 103112400078

KELAS: IF-12-01

**S1 TEKNIK INFORMATIKA**  
**TELKOM UNIVERSITY PURWOKERTO**  
**2025**

## **I. DASAR TEORI**

### **1. Pengertian Rekursif**

Rekursif dalam pemrograman adalah suatu teknik di mana sebuah fungsi atau prosedur memanggil dirinya sendiri untuk menyelesaikan suatu masalah. Teknik ini sering digunakan untuk menyelesaikan masalah yang dapat dibagi menjadi sub-masalah yang lebih kecil dengan bentuk yang serupa. Dalam rekursif, terdapat dua komponen utama yang harus diperhatikan: base-case dan recursive-case.

1. Base-case: Merupakan kondisi yang menghentikan pemanggilan rekursif. Tanpa base-case, pemanggilan fungsi atau prosedur akan berlangsung tanpa henti (infinite recursion). Base-case adalah hal yang harus dipahami terlebih dahulu sebelum menulis algoritma rekursif. Sebagai contoh, dalam kasus menghitung faktorial, base-case bisa berupa kondisi ketika  $n = 0$  atau  $n = 1$ .
2. Recursive-case: Merupakan kondisi pemanggilan fungsi atau prosedur itu sendiri. Pada recursive-case, fungsi memanggil dirinya lagi dengan parameter yang lebih kecil atau lebih sederhana, mendekati kondisi base-case. Sebagai contoh, dalam kasus faktorial,  $n$  dikurangi 1 setiap kali prosedur dipanggil.

Pada algoritma rekursif, pemanggilan fungsi berlanjut hingga mencapai base-case, kemudian proses akan "berbalik" atau mundur (backward), mengembalikan nilai dari pemanggilan-pemanggilan sebelumnya. Setiap langkah mundur akan mengembalikan hasil akhir ke pemanggilan pertama yang memulai proses tersebut.

Beberapa contoh masalah yang dapat diselesaikan dengan rekursif termasuk:

1. Menghitung faktorial ( $n!$ )
2. Menghitung pangkat dua ( $2^n$ )
3. Menghasilkan deret Fibonacci
4. Menampilkan pola tertentu, seperti pola bintang
5. Menampilkan faktor dari sebuah bilangan

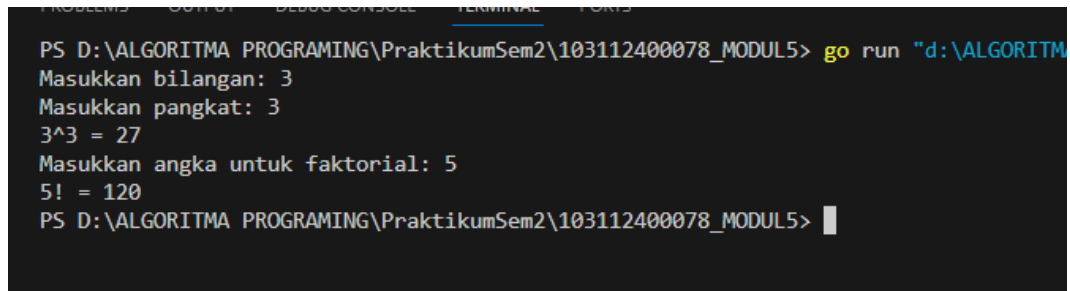
Secara umum, rekursif adalah alternatif untuk menggantikan penggunaan struktur perulangan (looping) dalam pemrograman.

## GUIDED 1

### SOURCE CODE:

```
guided3 > -go guided3.go > ...
1  package main
2
3  import "fmt"
4
5  // Fungsi iteratif untuk menghitung pangkat (base^exp)
6  func pangkatIteratif(base, exp int) int {
7      hasil := 1
8      for i := 0; i < exp; i++ {
9          hasil *= base
10     }
11     return hasil
12 }
13
14 // Fungsi iteratif untuk menghitung faktorial (n!)
15 func faktorialIteratif(n int) int {
16     hasil := 1
17     for i := 2; i <= n; i++ {
18         hasil *= i
19     }
20     return hasil
21 }
22
23 func main() {
24     var base, exp, n int
25
26     // Input pangkat
27     fmt.Print("Masukkan bilangan: ")
28     fmt.Scanln(&base)
29     fmt.Print("Masukkan pangkat: ")
30     fmt.Scanln(&exp)
31
32     fmt.Printf("%d^%d = %d\n", base, exp, pangkatIteratif(base, exp))
33
34     // Input faktorial
35     fmt.Print("Masukkan angka untuk faktorial: ")
36     fmt.Scanln(&n)
37     fmt.Printf("%d! = %d\n", n, faktorialIteratif(n))
38 }
39
```

## OUTPUT:

A screenshot of a terminal window with a dark background. At the top, there are tabs labeled 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'TOOLS'. The 'TERMINAL' tab is active. The terminal shows the following text: 'PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078\_MODUL5> go run "d:\ALGORITMA"', 'Masukkan bilangan: 3', 'Masukkan pangkat: 3', '3^3 = 27', 'Masukkan angka untuk faktorial: 5', '5! = 120', and 'PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078\_MODUL5>'.

```
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL5> go run "d:\ALGORITMA
Masukkan bilangan: 3
Masukkan pangkat: 3
3^3 = 27
Masukkan angka untuk faktorial: 5
5! = 120
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL5> |
```

## DEKSRIPSI:

Program ini berfungsi untuk menghitung dua hal secara terpisah pangkat dan faktorial. Program pertama-tama meminta pengguna untuk memasukkan angka dasar dan eksponen, kemudian menghitung hasil pangkat menggunakan fungsi iteratif pangkatIteratif. Fungsi ini mengalikan angka dasar sebanyak eksponen yang dimasukkan. Program juga meminta pengguna untuk memasukkan angka untuk dihitung faktorialnya, menggunakan fungsi iteratif faktorialIteratif, yang mengalikan angka dari 1 hingga angka yang dimasukkan untuk menghasilkan hasil faktorial. Kedua hasil perhitungan tersebut kemudian ditampilkan ke layar.

## GUIDED 2

### SOURCE CODE:

```
guided4 > go guided4.go > ...
1 package main
2
3 import "fmt"
4
5 func pangkatRekursif(base, exp int) int {
6     if exp == 0 {
7         return 1
8     }
9     return base * pangkatRekursif(base, exp-1)
10 }
11
12 func faktorialRekursif(n int) int {
13     if n == 0 || n == 1 {
14         return 1
15     }
16     return n * faktorialRekursif(n-1)
17 }
18
19 func main() {
20     var base, exp, n int
21
22     // Input pangkat
23     fmt.Print("Masukkan bilangan: ")
24     fmt.Scanln(&base)
25     fmt.Print("Masukkan pangkat: ")
26     fmt.Scanln(&exp)
27     fmt.Printf("%d^%d = %d\n", base, exp, pangkatRekursif(base, exp))
28
29     // Input faktorial
30     fmt.Print("Masukkan angka untuk faktorial: ")
31     fmt.Scanln(&n)
32     fmt.Printf("%d! = %d\n", n, faktorialRekursif(n))
33 }
```

### OUTPUT:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL5> go run "d:\ALGORITMA PROGRAMING\Praktikum5
Masukkan bilangan: 3
Masukkan pangkat: 3
3^3 = 27
Masukkan angka untuk faktorial: 5
5! = 120
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL5> |
```

**DEKSRIPSI:**

Program ini berfungsi untuk menghitung pangkat dan faktorial menggunakan metode rekursif. Pada bagian pertama, pengguna diminta untuk memasukkan angka dasar dan eksponen, kemudian fungsi pangkatRekursif akan menghitung hasil pangkat dengan memanggil dirinya sendiri hingga mencapai eksponen 0. Bagian kedua dari program meminta pengguna untuk memasukkan angka untuk dihitung faktorialnya, dan fungsi faktorialRekursif akan menghitung faktorial dengan memanggil dirinya sendiri hingga mencapai angka 1. Hasil perhitungan pangkat dan faktorial kemudian ditampilkan di layar.

## II. UNGUIDED

### UNGUIDED 1

#### SOURCE CODE

```
unguided1 > ~\go 1.go > ...
1  package main
2  // Mohammad Reyhan Aretha Fatin
3  // 103112400078
4  import "fmt"
5
6  func deretFibonacci(x int) int {
7      if x == 0 {
8          return 0
9      } else if x == 1 {
10         return 1
11     } else {
12         return deretFibonacci(x-1)+deretFibonacci(x-2)
13     }
14 }
15
16 func main() {
17     var a int
18     fmt.Scan(&a)
19     for i := 0; i <= a; i++ {
20         fmt.Print(deretFibonacci(i), " ")
21     }
22 }
```

#### OUTPUT

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL5> go run "d:\ALC
10
0 1 1 2 3 5 8 13 21 34 55
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL5> |
```

## **DEKSRIPSI**

Program ini adalah implementasi fungsi untuk menghitung deret Fibonacci dalam bahasa pemrograman Go. Fungsi `deretFibonacci(x int)` akan menghasilkan angka Fibonacci berdasarkan nilai `x` yang dimasukkan. Jika nilai `x` adalah 0, fungsi ini mengembalikan 0; jika `x` adalah 1, maka fungsi ini mengembalikan 1; dan untuk nilai `x` lainnya, fungsi ini memanggil dirinya sendiri secara rekursif untuk menghitung angka Fibonacci. Pada bagian `main()`, program meminta input dari pengguna untuk menentukan berapa banyak angka Fibonacci yang ingin ditampilkan, kemudian mencetak angka-angka Fibonacci tersebut dalam urutan yang benar dengan pemisah spasi.



## UNGUIDED 2

### SOURCE CODE

```
unguided2 > -go 2.go > ...
1  package main
2  // Mohammad Reyhan Aretha Fatin
3  // 103112400078
4  import "fmt"
5
6  func cetakSegitiga(n int) {
7      for i := 1; i <= n; i++ {
8          for j := 1; j <= i; j++ {
9              fmt.Print("*")
10             }
11             fmt.Println()
12         }
13     }
14
15     func main() {
16         var a int
17         fmt.Scan(&a)
18         cetakSegitiga(a)
19     }
```

### OUTPUT

```
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL5> go run "d:\ALGORITMA PROGR
5
*
**
***
****
*****

PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL5> go run "d:\ALGORITMA PROGR
1
*

PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL5> 
```

## **DEKSRIPSI**

Program ini adalah implementasi fungsi untuk mencetak segitiga bintang (\*) dalam bahasa pemrograman Go. Fungsi cetakSegitiga(n int) menerima input berupa angka n, yang menentukan jumlah baris dalam segitiga. Program menggunakan dua loop bersarang: loop pertama untuk mencetak baris-baris segitiga, dan loop kedua untuk mencetak bintang pada setiap baris. Setiap baris segitiga akan memiliki jumlah bintang yang sesuai dengan nomor barisnya. Setelah mencetak semua bintang dalam satu baris, program melanjutkan ke baris berikutnya dengan perintah `fmt.Println()`. Pada bagian `main()`, program meminta input dari pengguna untuk menentukan jumlah baris segitiga yang ingin dicetak, kemudian memanggil fungsi `cetakSegitiga(a)` untuk menampilkan segitiga sesuai input yang diberikan. Contoh output menunjukkan bagaimana segitiga dengan lima baris dicetak.

## UNGUIDED 2

### SOURCE CODE

```
unguided3 > 3.go > ...
1 package main
2 // Mohammad Reyhan Aretha Fatin
3 // 103112400078
4 import "fmt"
5
6 func faktorBilangan(x int) {
7     for i := 1; i <= x; i++ {
8         if x%i == 0 {
9             fmt.Print(i, " ")
10        }
11    }
12 }
13
14 func main() {
15     var a int
16     fmt.Scan(&a)
17     faktorBilangan(a)
18 }
```

### OUTPUT

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL5> go run "d:\ALGORITMA PROGRAMING\
5
1 5
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL5> go run "d:\ALGORITMA PROGRAMING\
12
1 2 3 4 6 12
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL5> |
```

### DEKSRIPSI

Program ini digunakan untuk mencari dan menampilkan faktor-faktor dari sebuah bilangan yang dimasukkan oleh pengguna. Fungsi faktorBilangan(x int) menerima input berupa angka x, dan menggunakan loop untuk memeriksa setiap angka dari 1 hingga x apakah merupakan faktor dari x tersebut. Jika angka tersebut membagi x tanpa sisa (dengan menggunakan operator modulus %), angka itu akan dicetak sebagai faktor. Pada fungsi main(), program meminta pengguna untuk memasukkan sebuah angka, kemudian memanggil fungsi faktorBilangan(a) untuk menampilkan semua faktor dari angka yang dimasukkan. Output dari program ini, seperti yang terlihat pada terminal, menunjukkan daftar faktor dari bilangan 5 dan 12 yang dimasukkan.

### III. KESIMPULAN

Kesimpulan dari laporan praktikum ini adalah bahwa rekursif merupakan salah satu teknik yang sangat berguna dalam pemrograman untuk menyelesaikan masalah yang dapat dibagi menjadi sub-masalah yang lebih kecil dengan bentuk yang serupa. Teknik rekursif bekerja dengan cara memanggil fungsi atau prosedur itu sendiri secara berulang sampai mencapai kondisi yang disebut base-case, yaitu kondisi yang menghentikan pemanggilan fungsi tersebut agar tidak terjadi *infinite recursion* yang dapat menyebabkan program berhenti atau crash.

Pada praktikum ini, teknik rekursif digunakan untuk menghitung dua hal, yaitu pangkat dan faktorial, yang keduanya dapat diselesaikan dengan menggunakan metode rekursi. Pada bagian pertama, program menghitung pangkat dengan meminta input berupa angka dasar dan eksponen, kemudian menghitung hasil pangkat secara rekursif dengan memanggil fungsi yang sama hingga eksponen mencapai 0. Begitu pula dengan perhitungan faktorial, program meminta input berupa angka yang kemudian dihitung secara rekursif dengan memanggil fungsi yang sama hingga mencapai angka 1, yang menjadi base-case.

Penggunaan rekursif dalam kedua kasus ini menunjukkan bagaimana rekursi dapat menggantikan penggunaan struktur perulangan (looping) dalam pemrograman. Meskipun rekursi sering kali lebih elegan dan lebih mudah dipahami, penting untuk diperhatikan bahwa teknik ini memerlukan perhatian khusus pada penentuan base-case yang tepat agar rekursi tidak berlanjut tanpa henti. Jika base-case tidak ditentukan dengan baik, pemanggilan fungsi akan terus berlangsung tanpa akhir, menyebabkan program mengalami infinite recursion.

Selain itu, rekursif juga memungkinkan pemecahan masalah yang kompleks menjadi langkah-langkah yang lebih sederhana, yang membantu programmer untuk lebih fokus pada logika dasar dari masalah yang sedang dihadapi. Secara keseluruhan, rekursif merupakan alternatif yang sangat berguna untuk menyelesaikan masalah yang memiliki pola atau struktur yang berulang, menjadikannya alat yang sangat penting dalam pemrograman.

Dengan menggunakan rekursif dalam praktikum ini, kita dapat melihat bagaimana cara kerja pemrograman yang lebih efisien dan terstruktur, yang pada akhirnya meningkatkan pemahaman tentang algoritma dan pemrograman secara keseluruhan.

## **REFERENSI**

MODUL 5. REKURSIF ALGORITMA PEMOGRAMAN 2