

LAPORAN
PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL 5
REKURSIF



Oleh:

NAMA: SETYO NIGROHO

NIM: 103112400024

KELAS: IF-12-01

S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

I. DASAR TEORI

1. Pengertian Rekursif

Rekursif adalah teknik dalam pemrograman di mana sebuah fungsi atau prosedur memanggil dirinya sendiri untuk menyelesaikan masalah. Teknik ini sangat berguna untuk menyelesaikan masalah yang dapat dibagi menjadi sub-masalah yang lebih kecil dan serupa dengan masalah utama. Prinsip dasar rekursif terdiri dari dua komponen utama:

1. **Base-case (Kasus Dasar):** Ini adalah kondisi yang menghentikan proses rekursif. Tanpa base-case, rekursif akan terus berjalan tanpa henti, yang dapat menyebabkan *stack overflow*. Base-case mendefinisikan titik akhir dari rekursi, di mana fungsi tidak lagi memanggil dirinya sendiri.
2. **Recursive-case (Kasus Rekursif):** Ini adalah kondisi di mana fungsi memanggil dirinya sendiri untuk melanjutkan proses hingga mencapai base-case. Recursive-case adalah bagian yang menyelesaikan sub-masalah.

Secara sederhana, rekursif bekerja dengan membagi masalah utama menjadi sub-masalah yang lebih kecil dan menyelesaikannya secara berulang hingga kondisi base-case tercapai. Setelah base-case tercapai, proses rekursif berhenti dan hasil dihitung dari sub-masalah yang lebih kecil tersebut.

Contoh penerapan rekursif:

- **Faktorial (n!):** Fungsi rekursif digunakan untuk menghitung faktorial sebuah angka dengan memanggil dirinya sendiri hingga mencapai angka 1.
- **Baris Fibonacci:** Setiap angka dalam deret Fibonacci dihitung dengan menjumlahkan dua angka sebelumnya, yang merupakan masalah rekursif karena setiap angka bergantung pada dua angka sebelumnya.

Rekursif juga bisa diterapkan dalam berbagai algoritma lain, seperti pencarian, pengurutan, atau pemrosesan struktur data seperti pohon dan graf.

GUIDED 1

SOURCE CODE:

```
guided1.go
guided1 > cd guided1go > _
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:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400024_MODUL5> go run "d:\ALGORITMA PROGRAMING\PRAKTIKUM\
Masukkan bilangan: 3
Masukkan pangkat: 3
3^3 = 27
Masukkan angka untuk faktorial: 5
5! = 120
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400024_MODUL5> |
```

DEKSRIPSI:

Program ini menghitung dua hal terpisah: pangkat dan faktorial. Pertama, pengguna diminta memasukkan angka dasar dan eksponen, lalu program menghitung hasil pangkat dengan mengalikan angka dasar sebanyak eksponen kali menggunakan fungsi pangkatIteratif. Kedua, program meminta pengguna memasukkan angka untuk dihitung faktorialnya, dan hasil faktorial dihitung dengan mengalikan angka dari 1 hingga angka yang dimasukkan menggunakan fungsi faktorialIteratif.

GUIDED 2

SOURCE CODE:

```
guided4.go X
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\PRAKTIKUM\103112400024_MODUL5> go run "d:\ALGORITMA PROGRAMING\PRAKTIKUM\
Masukkan bilangan: 3
Masukkan pangkat: 3
3^3 = 27
Masukkan angka untuk faktorial: 5
5! = 120
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400024_MODUL5> 
```

DEKSRIPSI:

Program ini menghitung pangkat dan faktorial menggunakan metode rekursif. Pengguna diminta untuk memasukkan angka dasar dan eksponen, lalu fungsi pangkatRekursif menghitung hasil pangkat dengan memanggil dirinya sendiri hingga eksponen mencapai 0. Program juga meminta pengguna untuk memasukkan angka untuk dihitung faktorialnya, di mana fungsi faktorialRekursif menghitung faktorial dengan memanggil dirinya sendiri hingga mencapai angka 1..

II. UNGUIDED

UNGUIDED 1

SOURCE CODE & OUTPUT

```
unguided1.go X
unguided1 > unguided1.go > ...
1  package main
2  //SETYO NUGROHO
3  //103112400024
4  import "fmt"
5  func fibonacci(n int) int {
6      if n == 0 {
7          return 0
8      } else if n == 1 {
9          return 1
10     }
11     return fibonacci(n-1) + fibonacci(n-2)
12 }
13 func main() {
14     var n int
15     fmt.Print("Masukkan jumlah suku Fibonacci: ")
16     fmt.Scan(&n)
17
18     if n < 0 {
19         fmt.Println("Masukkan angka positif!")
20         return
21     }
22
23     fmt.Println("Deret Fibonacci:")
24     for i := 0; i < n; i++ {
25         fmt.Printf("%d ", fibonacci(i))
26     }
27     fmt.Println()
28 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

0 1 1 2 3 5 8 13 21 34

PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400024_MODUL5>

DEKSRIPI

Program ini mencetak deret Fibonacci berdasarkan jumlah suku yang diminta oleh pengguna. Pengguna diminta untuk memasukkan jumlah suku yang diinginkan, dan program akan menghitung setiap angka dalam deret Fibonacci menggunakan rekursi. Deret dimulai dari angka 0 dan 1, dan angka selanjutnya diperoleh dengan menjumlahkan dua angka sebelumnya. Program akan mencetak deret Fibonacci sampai jumlah yang diminta. Jika input yang diberikan adalah angka negatif, program akan memberikan pesan "Masukkan angka positif!" sebagai peringatan.

UNGUIDED 2

SOURCE CODE & OUTPUT

```
unguided2> -go unguided2.go > cetakBintang
1 package main
2
3 //SETYO MUGROHO
4 //103112400024
5 import "fmt"
6
7 func main() {
8     var n int
9     fmt.Scan(&n)
10    cetakBintang(n, 1)
11 }
12
13 func cetakBintang(n, i int) {
14     if i > n {
15         return
16     }
17     for j := 0; j < i; j++ {
18         fmt.Print("*")
19     }
20     fmt.Println()
21     cetakBintang(n, i+1)
22 }
23
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

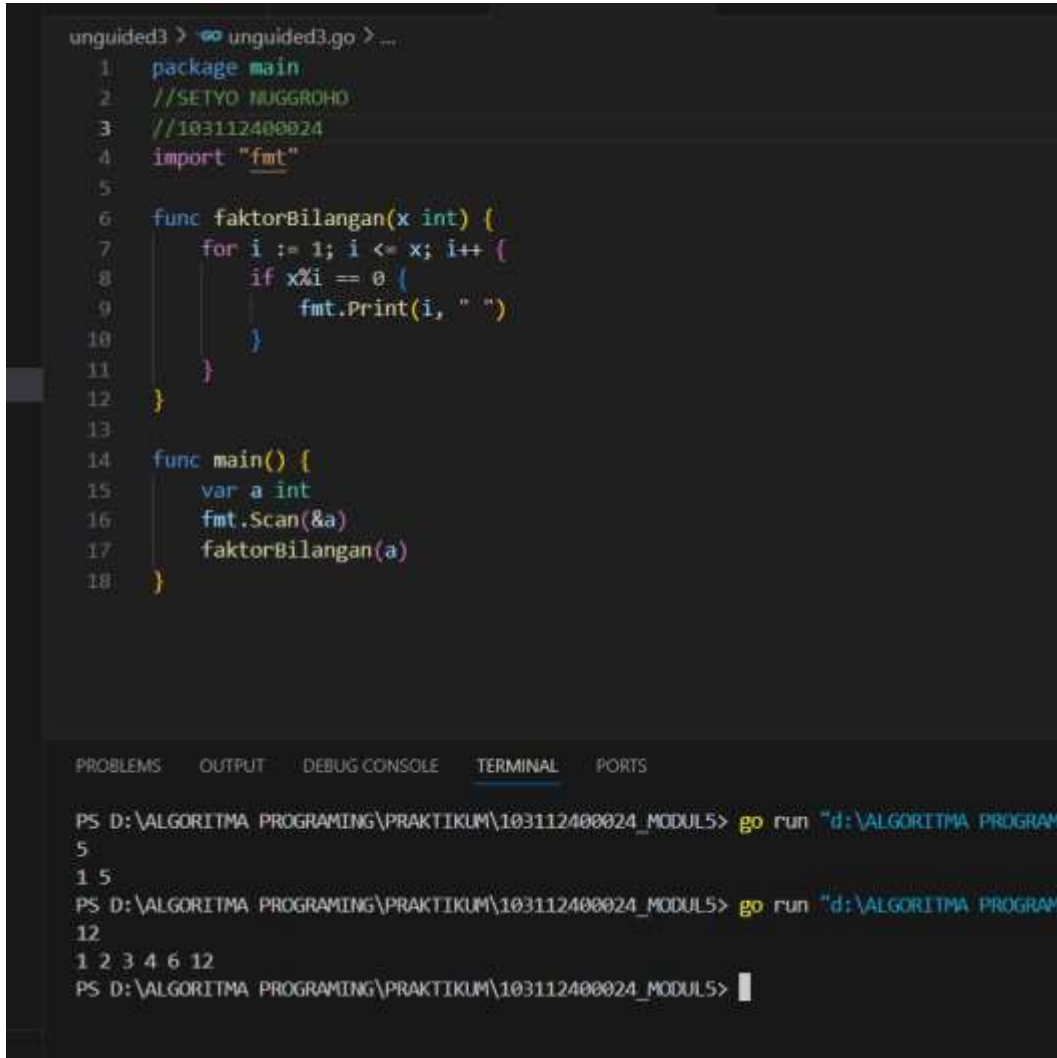
```
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400024_MODUL5> go run "d:\ALGORITMA PROGRAMING\PRAKTIKUM\1031124
5
*
**
***
****
*****
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400024_MODUL5>
```

DEKSRIPSI

Program ini akan mencetak pola bintang berbentuk segitiga, dengan jumlah baris yang sesuai dengan input dari pengguna. Pengguna diminta untuk memasukkan angka yang menunjukkan jumlah baris. Program akan menggunakan rekursi untuk mencetak pola, di mana setiap baris mencetak satu bintang lebih banyak daripada baris sebelumnya, dimulai dari satu bintang pada baris pertama. Pola ini terus berlanjut hingga mencapai jumlah baris yang diminta. Jika input lebih besar dari jumlah baris yang diinginkan, program akan berhenti.

UNGUIDED 3

SOURCE CODE & OUTPUT



```
unguided3 > unguided3.go > ...
1 package main
2 //SETYO MUGGROHO
3 //103112400024
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 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400024_MODUL5> go run "d:\ALGORITMA PROGRAM
5
1 5
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400024_MODUL5> go run "d:\ALGORITMA PROGRAM
12
1 2 3 4 6 12
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400024_MODUL5>
```

DEKSRIPSI

Program ini akan menerima input berupa angka dan kemudian mencetak faktor-faktor dari angka tersebut. Fungsi faktorBilangan digunakan untuk mencari dan mencetak semua angka yang dapat membagi angka input dengan sempurna. Program akan melakukan iterasi mulai dari angka 1 hingga angka yang dimasukkan, dan untuk setiap angka yang membagi angka input, angka tersebut akan dicetak sebagai faktor. Misalnya, jika inputnya adalah 12, maka program akan mencetak faktor-faktor 1, 2, 3, 4, 6, dan 12.

III. KESIMPULAN

Kesimpulan dari laporan praktikum ini menunjukkan bahwa teknik rekursif merupakan salah satu konsep penting dalam pemrograman yang memungkinkan pemecahan masalah dengan membagi masalah besar menjadi sub-masalah yang lebih kecil dan serupa. Dalam laporan ini, berbagai program yang menggunakan rekursif untuk menyelesaikan masalah seperti menghitung faktorial, menghitung pangkat, menghasilkan deret Fibonacci, mencetak pola bintang, dan mencari faktor bilangan telah dibahas secara rinci. Setiap program memanfaatkan prinsip dasar rekursif yang melibatkan dua bagian utama: base case untuk menghentikan rekursi dan recursive case untuk melanjutkan proses hingga mencapai base case tersebut.

Melalui penerapan teknik rekursif pada masalah-masalah yang diselesaikan dalam praktikum ini, dapat disimpulkan bahwa rekursif merupakan metode yang sangat efisien, terutama untuk masalah yang memiliki struktur berulang atau dapat dibagi menjadi bagian-bagian yang lebih kecil. Selain itu, rekursif memberikan cara yang elegan dan terstruktur untuk menyelesaikan masalah tanpa memerlukan banyak kode atau logika yang kompleks. Dengan memahami konsep dasar rekursif dan penerapannya, pemrogram dapat memecahkan berbagai masalah yang lebih kompleks dengan cara yang lebih efisien dan terorganisir. Secara keseluruhan, rekursif adalah teknik yang sangat berguna dalam pemrograman dan dapat meningkatkan kemampuan pemrogram dalam menyelesaikan masalah yang memiliki pola berulang dengan cara yang lebih optimal.

REFERENSI

MODUL 5 REKURSIF ALGORITMA PEMOGRAMAN 2