

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 5
REKURSIF**



Oleh:

MUHAMMAD RUSDIYANTO

2311102053

S1IF-11-02

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

I. DASAR TEORI

Fungsi atau prosedur adalah blok kode yang dirancang untuk melakukan tugas tertentu dan dapat dipanggil dari bagian lain dalam program. Salah satu fitur penting dari fungsi adalah kemampuannya untuk memanggil fungsi lain atau bahkan dirinya sendiri. Ketika sebuah fungsi memanggil dirinya sendiri, konsep ini dikenal sebagai `rekursi`. Rekursi merupakan salah satu teknik dasar dalam pemrograman yang memungkinkan penyelesaian masalah dengan memecahnya menjadi bagian-bagian yang lebih kecil, sering kali dengan struktur yang sama seperti masalah asli. Teknik ini memudahkan pemrograman dalam kasus di mana masalah dapat dipecah menjadi sub-masalah yang serupa tetapi lebih sederhana.

Pada dasarnya, fungsi rekursif akan terus memanggil dirinya sendiri sampai mencapai kondisi tertentu yang dikenal sebagai `base case`. `Base case` adalah kondisi penghentian dalam rekursi yang mencegah fungsi untuk terus memanggil dirinya sendiri tanpa batas, sehingga menghindari terjadinya `infinite loop`. Tanpa adanya `base case`, fungsi rekursif akan terus berjalan tanpa akhir, yang dapat menyebabkan error `stack overflow` atau error memori lainnya. Oleh karena itu, penting bagi setiap fungsi rekursif untuk memiliki kondisi penghentian yang jelas agar eksekusi program dapat selesai dengan benar dan memberikan hasil yang diinginkan.

Rekursi sering kali dibandingkan dengan loop atau perulangan, seperti `for` atau `while`. Keduanya bertujuan untuk melakukan proses berulang hingga kondisi tertentu terpenuhi. Namun, perbedaan utama antara rekursi dan loop adalah cara mereka melakukan pengulangan. Dalam loop, proses berulang dilakukan dalam blok kode yang terus menerus berjalan berdasarkan kondisi, sedangkan dalam rekursi, pengulangan terjadi melalui pemanggilan fungsi yang menyelesaikan tugas dengan memecahnya menjadi masalah yang lebih kecil. Rekursi cenderung lebih elegan dan alami untuk beberapa jenis masalah, seperti pemrosesan struktur data berbentuk pohon atau grafik, namun dalam beberapa kasus, loop lebih efisien karena rekursi dapat menyebabkan overhead yang lebih besar pada penggunaan memori.

Titik rekursi atau tempat di mana fungsi memanggil dirinya sendiri sangat mempengaruhi hasil akhir dari fungsi tersebut. Jika titik rekursi diletakkan sebelum base case, hasil yang diinginkan akan dihasilkan setelah semua panggilan rekursif selesai, seperti pada pola `post-order` dalam

traversal pohon. Sebaliknya, jika titik rekursi berada setelah base case atau proses lainnya, hasil yang diperoleh dapat segera terlihat sebelum rekursi diproses lebih lanjut, seperti pada pola `pre-order`. Peletakan titik rekursi yang tepat tergantung pada bagaimana kita ingin memecahkan masalah dan bagaimana sub-masalah saling terkait satu sama lain.

II. GUIDED

Guided 1 | Source Code

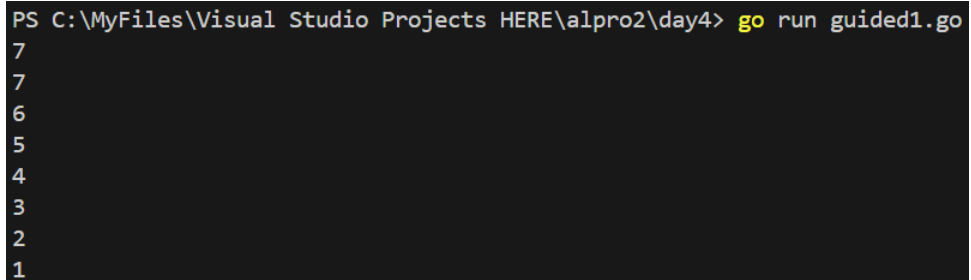
```
package main

import "fmt"

func main() {
    var n int
    fmt.Scan(&n) // Membaca input pengguna
    baris(n)     // Memanggil fungsi rekursif `baris`
}

func baris(bilangan int) {
    if bilangan == 1 { // Base case: Jika bilangan sama dengan
        fmt.Println(1) // Cetak angka 1
    } else { // Jika bilangan lebih besar dari 1
        fmt.Println(bilangan) // Cetak bilangan saat ini
        baris(bilangan - 1) // Panggil fungsi `baris` dengan bilangan
    }
}
```

Guided 1 | Output



```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day4> go run guided1.go
7
7
6
5
4
3
2
1
```

Guided 1 | Penjelasan

Program di atas merupakan sebuah program yang menampilkan baris bilangan dari angka yang diinputkan sampai dengan 1. Proses dalam program dimulai dengan mengambil input dari pengguna. Lalu, program akan lanjut dengan memanggil fungsi `baris`. Di dalam fungsi `baris`, fungsi akan mengecek apakah bilangan sama dengan 1. Jika iya, maka program akan mencetak 1 dan program akan berhenti. Jika tidak, maka program akan mencetak bilangan, lalu fungsi `baris` akan memanggil dirinya sendiri dengan parameter bilangan - 1. Nilai parameter dibuat seperti itu supaya angka yang akan ditampilkan selanjutnya semakin kecil dan menuju angka 1. Hal ini juga dilakukan supaya rekursi tidak terjadi secara tak hingga.

Guided 2 | Source Code

```
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)
    }
}
```

Guided 2 | Output

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day4> go run guided2.go
5
15
```

Guided 2 | Penjelasan

Program di atas adalah program untuk menjumlahkan bilangan dari bilangan yang diinputkan hingga 1. Misalkan, kita memberikan input 4, maka program akan menjumlahkan angka dari $4 + 3 + 2 + 1$. Proses dalam program ini dimulai dari input suatu bilangan. Setelah bilangan diinputkan, maka program akan memanggil fungsi `penjumlahan`. Di dalam fungsi `penjumlahan` nilai dari `n` (nilai dari input) akan dicek, apakah sama dengan 1. Jika sama dengan 1, maka program akan mengembalikan nilai 1. Jika tidak, maka program akan mengembalikan nilai `n` ditambah nilai yang dikembalikan dari pemanggilan dirinya sendiri dimana `n` dikurangi 1. Pemanggilan tersebut adalah titik rekursi dalam fungsi ini. Jika rekursi telah selesai, maka nilai terakhir akan dikembalikan dan ditampilkan oleh program.

III. UNGUIDED

Unguided 1 | Source Code

```
package main

import "fmt"

func fibonacci(n int) int {
    if n == 0 {
        return 0
    } else if n == 1 {
        return 1
    } else {
        var result int = fibonacci(n-1) + fibonacci(n-2)
        return result
    }
}

func nthFibonacci(num int) {
    if num == 0 {
        fmt.Printf("%v ", 0)
    } else if num == 1 {
        nthFibonacci(0)
        fmt.Printf("%v ", 1)
    } else {
        nthFibonacci(num - 1)
        fmt.Printf("%v ", fibonacci(num))
    }
}

func main() {
    var num int
    fmt.Scanln(&num)
    nthFibonacci(num)
}
```

Unguided 1 | Output

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day4> go run unguided1.go
10
0 1 1 2 3 5 8 13 21 34 55
```

Unguided 1 | Penjelasan

Program di atas adalah program untuk mencetak deret Fibonacci hingga bilangan ke-n, di mana n adalah angka yang dimasukkan oleh pengguna. Deret Fibonacci dimulai dengan 0 dan 1, dan setiap bilangan

berikutnya merupakan penjumlahan dari dua bilangan sebelumnya. Misalkan pengguna memasukkan angka 5, program akan mencetak deret Fibonacci hingga elemen ke-5, yaitu 0 1 1 2 3 5.

Proses dimulai dengan mengambil input num dari pengguna melalui fungsi main, kemudian input tersebut diteruskan ke fungsi `nthFibonacci` untuk mencetak deret Fibonacci hingga bilangan ke-n. Pada fungsi ini, rekursi digunakan untuk menghitung setiap bilangan Fibonacci secara berurutan, dengan memanggil fungsi `fibonacci` untuk menghitung elemen ke-n.

Fungsi `fibonacci` bekerja secara rekursif untuk menghitung nilai Fibonacci berdasarkan dua nilai sebelumnya (yaitu fibonacci(n-1) dan fibonacci(n-2)). Rekursi akan berhenti ketika nilai n sama dengan 0 atau 1, yang merupakan kasus dasar untuk deret Fibonacci. Setelah rekursi selesai, nilai Fibonacci dihitung dan dicetak melalui fungsi `nthFibonacci`, mulai dari elemen pertama hingga elemen ke-n.

Unguided 2 | Source Code

```
package main

import (
    "fmt"
)

func printStarLine(numOfStars int) string {
    if numOfStars <= 1 {
        return "*"
    } else {
        return "*" + printStarLine(numOfStars-1)
    }
}

func printStars(numOfStars int) {
    if numOfStars <= 1 {
        fmt.Println("*")
    } else {
        printStars(numOfStars - 1)
        fmt.Println(printStarLine(numOfStars))
    }
}

func main() {
    var num int
```

```
    fmt.Scanln(&num)
    printStars(num)
}
```

Unguided 2 | Output

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day4> go run unguided2.go
5
*
**
***
****
*****
```

Unguided 2 | Penjelasan

Program di atas adalah program untuk mencetak pola bintang secara bertahap menggunakan rekursi. Pengguna diminta memasukkan angka, yang akan menentukan jumlah baris bintang yang dicetak. Misalkan pengguna memasukkan angka 4, maka program akan mencetak pola bintang dengan 1 bintang di baris pertama, 2 bintang di baris kedua, dan seterusnya hingga 4 bintang di baris keempat.

Proses dimulai dengan fungsi `main` yang mengambil input dari pengguna dan memanggil fungsi `printStars`. Fungsi `printStars` merupakan fungsi rekursif yang bertugas mencetak setiap baris bintang. Pada setiap pemanggilan, fungsi ini mencetak pola bintang yang sesuai dengan jumlah bintang pada baris tersebut yang dihasilkan oleh fungsi `printStarLine`. Fungsi `printStarLine` juga menggunakan rekursi untuk membentuk barisan bintang pada setiap baris, di mana fungsi tersebut menambahkan satu bintang setiap kali dipanggil sampai mencapai jumlah bintang yang diminta.

Unguided 3 | Source Code

```
package main

import "fmt"

func factorial(num int, i int) {
    if i >= num {
        fmt.Printf("%v", num)
    } else {
        if num%i == 0 {
            fmt.Printf("%v ", i)
        }
    }
}
```



```

    }
    factorial(num, i+1)
}
}

func main() {
    var num int
    fmt.Scanln(&num)
    factorial(num, 1)
}

```

Unguided 3 | Output

```

PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day4> go run unguided3.go
12
1 2 3 4 6 12

```

Unguided 3 | Penjelasan

Program di atas merupakan program untuk menemukan dan mencetak faktor-faktor dari sebuah bilangan yang dimasukkan oleh pengguna. Misalkan pengguna memasukkan angka 12, maka program akan mencetak faktor-faktor dari 12, yaitu 1, 2, 3, 4, dan 6, diikuti oleh angka 12 itu sendiri.

Proses dimulai dengan mengambil input dari pengguna melalui fungsi `main`. Setelah bilangan diinputkan, fungsi `factorial` dipanggil dengan dua parameter, yaitu angka yang diinputkan dan nilai awal 1 sebagai indeks pengecekan faktor. Fungsi `factorial` bekerja secara rekursif untuk mengecek apakah bilangan tersebut habis dibagi oleh nilai *i* (nilai yang bertambah di setiap langkah rekursi). Jika variabel `num` habis dibagi *i*, maka nilai *i* akan dicetak sebagai faktor. Fungsi ini akan terus memanggil dirinya sendiri hingga nilai *i* mencapai atau melebihi nilai `num`.

Unguided 4 | Source Code

```

package main

import "fmt"

func barisBilangan(num int) {
    if num < 1 {
        fmt.Println("Bilangan harus positif")
    }
}

```

```

    } else if num == 1 {
        fmt.Print("1 ")
    } else {
        fmt.Printf("%v ", num)
        barisBilangan(num - 1)
        fmt.Printf("%v ", num)
    }
}

func main() {
    var num int
    fmt.Scanln(&num)
    barisBilangan(num)
}

```

Unguided 4 | Output

```

PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day4> go run unguided4.go
9
9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9

```

Unguided 4 | Penjelasan

Program di atas adalah program untuk mencetak pola baris bilangan yang menurun hingga angka 1, kemudian kembali naik ke angka semula secara rekursif. Misalkan pengguna memasukkan angka 4, maka output yang dihasilkan adalah: 4 3 2 1 2 3 4. Proses dimulai dengan mengambil input dari pengguna melalui fungsi `main`, lalu memanggil fungsi `barisBilangan`. Fungsi tersebut mengecek apakah bilangan yang dimasukkan lebih kecil dari 1. Jika iya, maka akan ditampilkan pesan bahwa bilangan harus positif. Jika tidak, fungsi akan mencetak angka saat ini (num), memanggil dirinya sendiri dengan nilai num - 1, lalu mencetak kembali angka yang sama setelah rekursi selesai.

Unguided 5 | Source Code

```

package main

import "fmt"

func barisGanjil(num int) {
    if num <= 1 {
        fmt.Print("1 ")
    } else if num%2 != 0 {
        barisGanjil(num - 1)
        fmt.Printf("%v ", num)
    } else {

```

```

        barisGanjil(num - 1)
    }
}

func main() {
    var num int
    fmt.Scanln(&num)
    barisGanjil(num)
}

```

Unguided 5 | Output

```

PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day4> go run unguided5.go
20
1 3 5 7 9 11 13 15 17 19

```

Unguided 5 | Penjelasan

Program di atas adalah program untuk mencetak bilangan ganjil dari 1 hingga angka yang dimasukkan oleh pengguna secara rekursif. Proses dimulai dengan mengambil input dari pengguna melalui fungsi main. Input tersebut kemudian diteruskan ke fungsi `barisGanjil`. Fungsi ini memeriksa apakah nilai `num` (input yang dimasukkan) lebih kecil atau sama dengan 1. Jika ya, program akan mencetak 1 sebagai bilangan ganjil terkecil. Jika tidak, program mengecek apakah num merupakan bilangan ganjil dengan menggunakan operator modulus ($\text{num} \% 2 \neq 0$). Jika `num` adalah ganjil, maka program memanggil fungsi secara rekursif dengan $\text{num} - 1$ terlebih dahulu, lalu mencetak bilangan ganjil tersebut setelah rekursi selesai. Jika `num` adalah bilangan genap, program hanya memanggil rekursi tanpa mencetak angka tersebut.

Unguided 6 | Source Code

```

package main

import "fmt"

func power(num int, power_of int) int {
    if power_of <= 1 {
        return num * power_of
    } else {
        return num * power(num, power_of-1)
    }
}

```

```
    }  
}  
  
func main() {  
    var num, power_of int  
    fmt.Scanln(&num, &power_of)  
    fmt.Println(power(num, power_of))  
}
```

Unguided 6 | Output

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day4> go run unguided6.go  
5 3  
125
```

Unguided 6 | Penjelasan

Program di atas adalah program untuk menghitung pangkat dari sebuah bilangan secara rekursif. Proses dimulai dengan mengambil input dari pengguna, yaitu dua bilangan: bilangan dasar (num) dan pangkatnya (power_of). Input ini kemudian diteruskan ke fungsi `power`, yang menghitung hasil perpangkatan. Pada fungsi `power`, jika nilai power_of kurang dari atau sama dengan 1, fungsi mengembalikan hasil perkalian antara `num` dan `power_of`. Jika `power_of` lebih besar dari 1, fungsi akan mengembalikan hasil dari perkalian num dengan hasil panggilan rekursif dari power(num, power_of - 1). Dengan ini, program akan menghitung pangkat dari bilangan dasar dengan mengalikan num berulang kali hingga nilai power_of mencapai 1. Hasil akhir dari perhitungan pangkat ini kemudian ditampilkan di layar.