

**LAPORAN PRAKTIKUM
ALGORITMA PEMOGRAMAN 2**

**MODUL 5
MATERI
REKURSIF**



Oleh:

FEBRIAN FALIH ALWAFI

2311102181

S1F-11-02

**S1 TEKNIK INFORMATIKA
UNIVERSITAS TELKOM PURWOKERTO**

2024

I. DASAR TEORI

- **Pengantar Rekursif**

Rekursif adalah sebuah metode pengulangan yang melibatkan penggunaan diri sendiri. Dalam konteks pemrograman, fungsi rekursif adalah suatu bentuk perulangan yang tidak melibatkan iterasi. Pada dasarnya, fungsi rekursif adalah fungsi biasa seperti definisi fungsi pada umumnya atau fungsi yang memanggil dirinya sendiri. Sebagai contoh perhatikan prosedur cetak berikut ini :

	Notasi Algoritma	Notasi dalam bahasa GO
1	procedure cetak(in x:integer)	func cetak(x int){
2	algoritma	fmt.Println(x)
3	output(x)	cetak(x+1)
4	cetak(x+1)	}
5	endprocedure	

Apabila diperhatikan subprogram cetak() di atas, terlihat pada baris ke-4 terdapat pemanggilan subprogram cetak() kembali. Misalnya apabila kita eksekusi perintah cetak(5) maka akan menampilkan angka 5 6 7 8 9...dst tanpa henti. Artinya setiap pemanggilan subprogram cetak() nilai x akan selalu bertambah 1 (Increment by one) secara terus menerus tanpa henti.

```
package main

import "fmt"

func main() {
    cetak(5)
}

func cetak(x int) {
    fmt.Println(x)
    cetak(x + 1)
}
```

Oleh karena itu bisanya ditambahkan struktur kontrol percabangan (if-then) untuk menghentikan proses rekursif ini. Kondisi ini disebut juga dengan base-case, artinya apabila kondisi base-case bernilai true maka proses rekursif akan berhenti. Sebagai contoh misalnya base case adalah ketika x bernilai 10 atau x < 10, maka tidak perlu dilakukan rekursif.

Catatan:

- Teknik rekursif ini merupakan salah satu alternatif untuk mengganti struktur kontrol perulangan dengan memanfaatkan subprogram (bisa fungsi ataupun prosedur).
- Untuk menghentikan proses rekursif digunakan percabangan (if then).
- **Base-case** adalah kondisi proses rekursif berhenti. Base-case merupakan hal terpenting dan pertama yang harus diketahui ketika akan membuat program rekursif. Mustahil membuat program rekursif tanpa mengetahui base-case terlebih dahulu.
- **Recursive-case** adalah kondisi dimana proses pemanggilan dirinya sendiri dilakukan. Kondisi recursive-case adalah komplemen atau negasi dari base-case.
- Setiap algoritma rekursif selalu memiliki padanan dalam bentuk algoritma iteratif.

Komponen Rekursif

Algoritma rekursif terdiri dari dua komponen utama:

- **Base-case (Basis)**, yaitu bagian untuk menghentikan proses rekursif dan menjadi komponen

terpenting di dalam sebuah rekursif

- **Recursive-case**, yaitu bagian pemanggilan subprogramnya

II. GUIDED

1. Guided 1

Source Code :

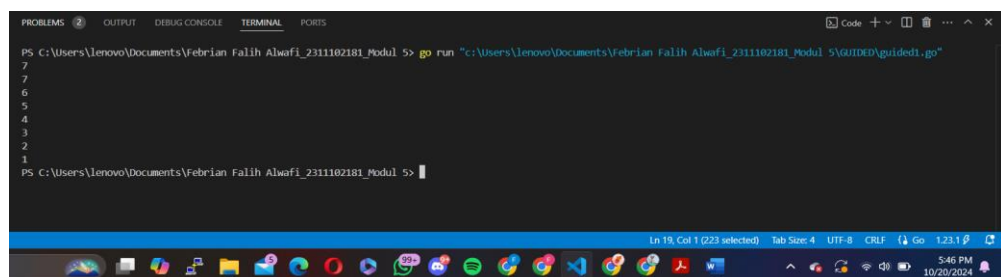
```
package main

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)
    }
}
```

Output :



```
PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 5> go run "C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 5\GUIDED\guided1.go"
7
6
5
4
3
2
1
PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 5>
```

Deskripsi :

Program ini meminta input berupa angka dari pengguna, kemudian mencetak angka tersebut secara menurun hingga mencapai angka 1. Saya mengambil contoh pada output diatas 7 dia akan mengeksekusi sampai 1. Penjelasan pada kode diatas, Fungsi baris menerima parameter berupa bilangan bulat, dan akan mencetak bilangan tersebut hingga mencapai angka 1. Jika bilangan yang diterima bernilai 1, fungsi akan berhenti dan

mencetak angka 1. Jika bilangan lebih dari 1, fungsi akan mencetak bilangan tersebut, lalu memanggil dirinya sendiri dengan parameter bilangan yang dikurangi 1. Proses ini akan terus berlangsung hingga kondisi dasar tercapai.

2. Guided 2

Source Code :

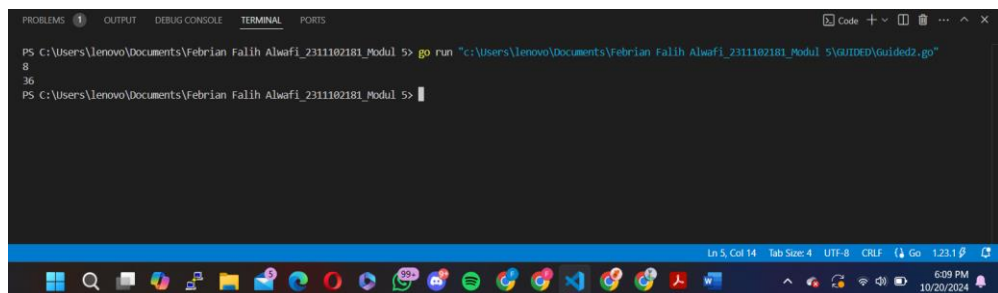
```
package main

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)
    }
}
```

Output :



```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS c:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 5> go run "c:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 5\GJIED\guided2.go"
8
36
PS c:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 5> |
```

Deskripsi :

Program diatas di gunakan untuk menghitung penjumlahan deret bilangan 1 hingga n. Saya ambil contoh pada output diatas yaitu 8, dia akan menambah terus hingga 1 seperti $8+7+6+5+4+3+2+1$ hasilnya 36.

Penjelasan kode diatas, Fungsi penjumlahan pada kode ini memiliki kondisi dasar yaitu ketika nilai n sama dengan 1, di mana fungsi langsung mengembalikan nilai 1. Jika nilai n lebih besar dari 1, fungsi akan memanggil dirinya sendiri dengan argumen $n-1$ dan menjumlahkan hasilnya dengan n . Proses ini berulang hingga nilai n mencapai 1, di mana rekursif berhenti dan hasil penjumlahan dari setiap panggilan dikembalikan secara bertahap.

III. UNGUIDED

1. Unguided 1

Source Code :

```
package main

import "fmt"

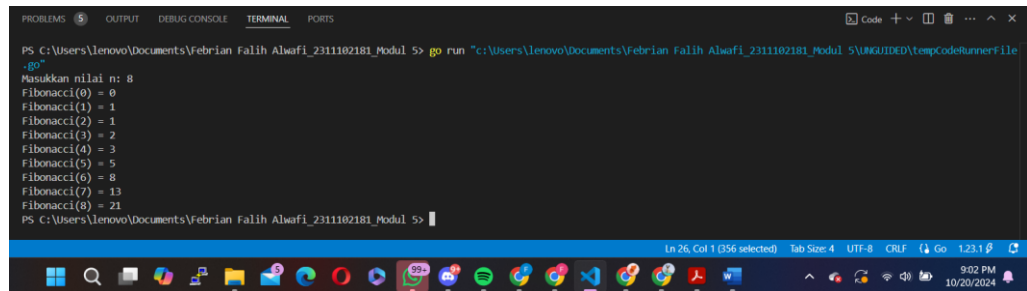
func fibonacci(n int) int {
    if n <= 1 {
        return n
    }

    a, b := 0, 1
    for i := 2; i <= n; i++ {
        a, b = b, a+b
    }
    return b
}

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

    for i := 0; i <= n; i++ {
        fmt.Printf("Fibonacci(%d) = %d\n", i, fibonacci(i))
    } //2311102181
}
```

Output :



```
PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 5> go run "c:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 5\UNGU/DED/tempCodeRunnerFile.go"
Masukkan nilai n: 8
Fibonacci(0) = 0
Fibonacci(1) = 1
Fibonacci(2) = 1
Fibonacci(3) = 2
Fibonacci(4) = 3
Fibonacci(5) = 5
Fibonacci(6) = 8
Fibonacci(7) = 13
Fibonacci(8) = 21
PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 5>
```

Deskripsi :

Program diatas adalah fungsi Fibonacci menggunakan pendekatan iteratif. Sebagai contoh output diatas memasukan 8 maka akan menghitung dan mencetak deret Fibonacci dari 0 hingga 8. Dalam kode diatas, fungsi fibonacci menerima parameter bilangan bulat n dan mengembalikan nilai Fibonacci yang sesuai. Jika n kurang dari atau sama dengan 1, fungsi ini langsung mengembalikan nilai n sebagai kondisi dasar. Dalam sebuah loop, fungsi menghitung nilai Fibonacci hingga mencapai n dengan menjumlahkan dua angka sebelumnya.

2. Unguided 2

Source Code :

```
package main

import "fmt"

func main() {
    var n int
    fmt.Print("Masukkan angka: ")
    fmt.Scanln(&n)

    printStars(n)
}

func printStars(n int) {
    if n == 0 {
        return
    }
}
```



```

    }

    printStars(n-1)

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

```

Output :

```

PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311182181_Modul 5> go run "c:\Users\lenovo\Documents\Febrian Falih Alwafi_2311182181_Modul 5\UNGUIDED\unguided2.go"
Masukkan angka: 8
*
**
***
****
*****
*****
*****
*****
PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311182181_Modul 5>

```

Deskripsi :

Program di atas digunakan untuk menampilkan pola bintang berdasarkan input angka yang diberikan oleh pengguna. Program dimulai dengan meminta pengguna untuk memasukkan sebuah angka, yang disimpan dalam variabel `n`. Fungsi `printStars` kemudian dipanggil dengan parameter `n`. Jika nilai `n` sama dengan 0, fungsi akan mengembalikan (return) tanpa mencetak apa pun, yang berfungsi sebagai kondisi dasar untuk menghentikan rekursi. Fungsi ini menggunakan pendekatan rekursif untuk mencetak bintang, di mana setelah semua panggilan rekursif selesai, fungsi mencetak bintang sebanyak `n` kali.

3. Unguided 3

Source Code :

```

package main

import "fmt"

```

```

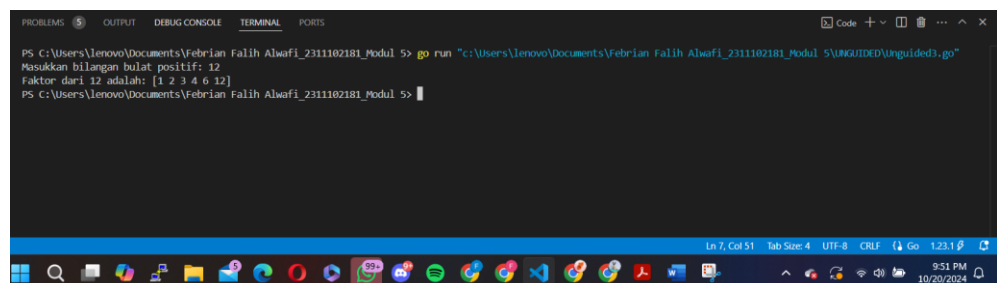
func main() {
    var n int
    fmt.Print("Masukkan bilangan bulat positif: ")
    fmt.Scanln(&n)

    fmt.Println("Faktor dari", n, "adalah:", faktor(n))
}

func faktor(n int) []int {
    var faktor []int
    for i := 1; i <= n; i++ {
        if n%i == 0 {
            faktor = append(faktor, i)
        }
    }
    return faktor
}

```

Output :



```

PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 5> go run "c:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 5\UNGUIDED\Unguided3.go"
Masukkan bilangan bulat positif: 12
Faktor dari 12 adalah: [1 2 3 4 6 12]
PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 5>

```

Deskripsi :

Program yang digunakan untuk menemukan dan menampilkan faktor dari bilangan bulat positif yang diberikan oleh pengguna. Sebagai contoh output diatas jika pengguna memasukkan angka 12, output program akan menampilkan "Faktor dari 12 adalah: [1 2 3 4 6 12]", yang menunjukkan semua angka yang dapat membagi 12 tanpa sisa. Program dimulai dengan meminta pengguna untuk memasukkan sebuah bilangan bulat positif, yang kemudian disimpan dalam variabel n. Setelah pengguna memberikan input, program akan memanggil fungsi faktor dengan parameter n untuk menghitung semua faktor dari bilangan tersebut.

4. Unguided 4

Source Code :

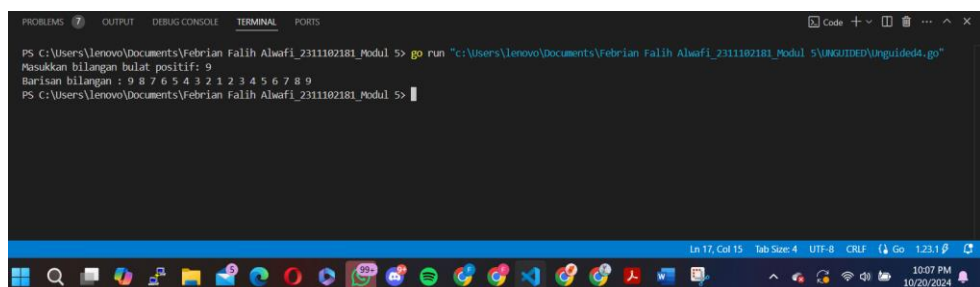
```
package main

import "fmt"

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

func printSequence(n int) {
    if n == 1 {
        fmt.Print(n, " ")
        return
    } //2311102181
    fmt.Print(n, " ")
    printSequence(n - 1)
    fmt.Print(n, " ")
}
```

Output :



```
PS C:\Users\lenovo\Documents\Febrian Falih Alhaifi_2311102181_Modul 5> go run "c:\Users\lenovo\Documents\Febrian Falih Alhaifi_2311102181_Modul 5\UNGUIDED4\unguided4.go"
Masukkan bilangan bulat positif: 9
Barisan bilangan : 9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9
PS C:\Users\lenovo\Documents\Febrian Falih Alhaifi_2311102181_Modul 5>
```

Deskripsi :

Program di atas menggunakan pendekatan rekursif untuk menampilkan barisan bilangan berdasarkan input bilangan bulat positif yang diberikan oleh pengguna. Sebagai contoh output diatas menampilkan, pengguna memasukkan angka 9, output program akan menghasilkan urutan "9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9". Pola ini menunjukkan bahwa fungsi pertama mencetak angka dalam urutan menurun hingga mencapai 1. Pertama Program dimulai dengan meminta pengguna untuk memasukkan sebuah bilangan bulat positif, yang disimpan dalam variabel n. Setelah mendapatkan input, program mencetak teks "Barisan bilangan : " sebagai pengantar sebelum memanggil fungsi printSequence dengan parameter n.

5. Unguided 5

Source Code :

```
package main

import "fmt"

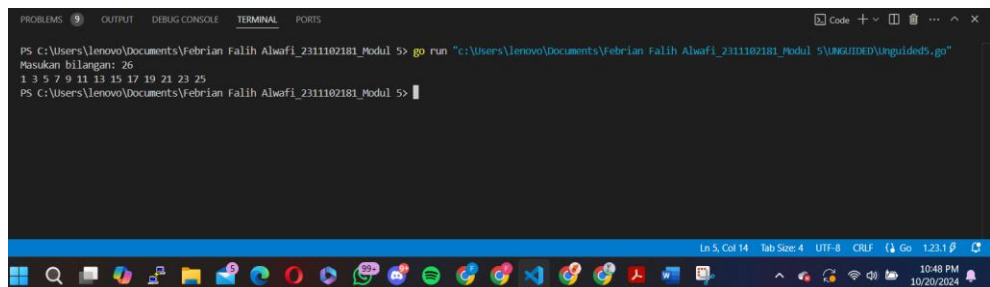
func main() {
    fmt.Print("Masukan bilangan: ")
    var n int
    fmt.Scan(&n)
    cetakGanjil(n)
}

func cetakGanjil(n int) {
    if n <= 0 {
        return
    } //2311102181

    cetakGanjil(n - 1)
```

```
    if n%2 != 0 {  
        fmt.Print(n, " ")  
    }  
}
```

Output :



```
PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 5> go run "c:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 5\UNGUIDED\unguided5.go"  
Masukan bilangan: 26  
1 3 5 7 9 11 13 15 17 19 21 23 25  
PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 5>
```

Deskripsi :

Program ini digunakan untuk menampilkan barisan bilangan ganjil dari 1 hingga bilangan yang dimasukkan oleh pengguna. Pertama, program meminta pengguna untuk memasukkan sebuah bilangan bulat positif, yang kemudian disimpan dalam variabel `n`. Setelah itu, fungsi `cetakGanjil(n)` dipanggil. Fungsi `cetakGanjil` bekerja secara rekursif. Pada setiap panggilan fungsi, nilai `n` akan berkurang satu hingga mencapai 0, yang merupakan kondisi dasar untuk menghentikan rekursi. Bilangan ganjil dari 1 hingga `n` ditampilkan dalam urutan menaik, karena pencetakan terjadi setelah pemanggilan rekursif selesai untuk setiap nilai `n`. Sebagai contoh pada output diatas, pengguna memasukkan angka 26, output yang dihasilkan adalah 1 3 5 7 9 11 13 15 17 19 21 23 25.

6. Unguided 6

Source Code :

```
package main

import "fmt"

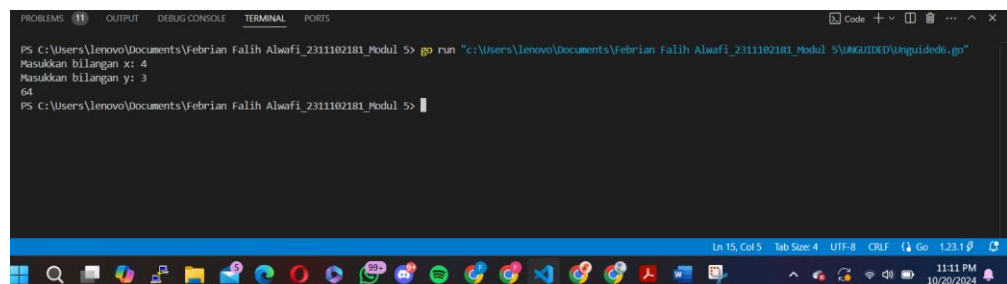
func pangkat(x, y int) int {
    if y == 0 {
        return 1
    }
    return x * pangkat(x, y-1)
}

func main() {
    var x, y int

    fmt.Print("Masukkan bilangan x: ")
    fmt.Scan(&x)
    fmt.Print("Masukkan bilangan y: ")
    fmt.Scan(&y)

    hasil := pangkat(x, y)
    fmt.Println(hasil)
}
```

Output :

A screenshot of a Windows terminal window with a dark background. The terminal shows the execution of a Go program. The prompt is 'PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 5>'. The user enters 'go run "c:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 5\ANGUIDED\unguidede.go"'. The program outputs 'Masukkan bilangan x: 4', then 'Masukkan bilangan y: 3', and finally '64'. The terminal window has tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The status bar at the bottom shows 'Ln 15, Col 5', 'Tab Size: 4', 'UTF-8', 'CRLF', 'Go', '1:23.1', and the date '10/20/2024'.

Deskripsi :

Program diatas bertujuan untuk menghitung hasil pangkat dari dua bilangan bulat, x dan y , menggunakan pendekatan rekursif. Dalam program ini, fungsi pangkat didefinisikan untuk menerima dua parameter: x , yang merupakan basis, dan y , yang merupakan eksponen. Fungsi ini menggunakan kondisi dasar bahwa jika y sama dengan 0, maka hasilnya adalah 1, sesuai dengan prinsip matematika bahwa setiap bilangan pangkat 0 adalah 1. Sebagai contoh output diatas, pengguna memasukan bilangan $x = 4$ dan $y = 3$, yang dimana x itu angka dan y itu pangkat, jadi 4 pangkat 3 adalah 64.