

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 6
REKURSIF**



Disusun Oleh :

Kanasya Abdi Aziz / 2311102140

S1-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

A. Pengantar Rekursif

Suatu subprogram baik fungsi atau prosedur bisa memanggil subprogram lainnya. Hal ini tidak menutup kemungkinan bahwa subprogram yang di panggil adalah dirinya sendiri. Dalam pemograman teknik ini dikenal dengan istilah rekursif.

Rekursif secara sederhana dapat di artikan sebagai cara menyelesaikan suatu masalah dengan cara menyelesaikan sub – masalah yang identic dari masalah utama. Sebagai contohnya perhatikan prosedur case berikut

	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 di perhatikan 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.

Teknik rekursif ini merupakan salah satu alternatif untuk menggantikan struktur control perulangan dengan memanfaatkan subprogram (bisa fungsi ataupun prosedur). Untuk menghentikan proses rekursif digunakan percabangan (if – then). Base – case adalah kondisi proses rekursif berhenti. Basecase merupakan ha 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 – ase. Setiap algoritma rekursif selalu memiliki padanan dalam bentuk algoritma iteratif.

B. 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

Soal Studi Case

Membuat baris bilangan dari n hingga 1

Base – case: bilangan == 1

Sourcecode

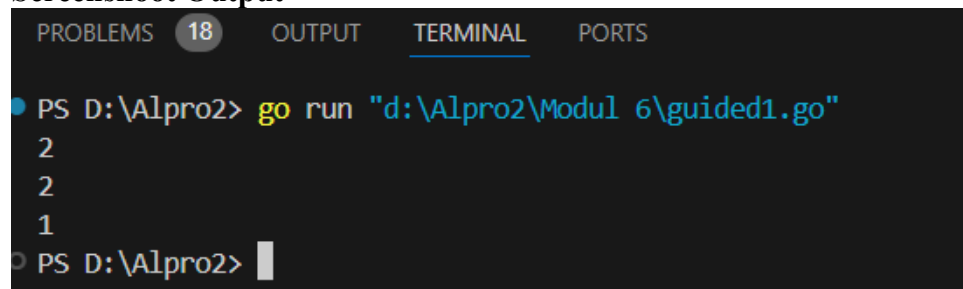
```
package main

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



```
PROBLEMS 18 OUTPUT TERMINAL PORTS
PS D:\Alpro2> go run "d:\Alpro2\Modul 6\guided1.go"
2
2
1
PS D:\Alpro2> 
```

Deskripsi Program :

Program ini adalah implementasi fungsi rekursif sederhana dalam bahasa Go yang mencetak bilangan dari N hingga 1 secara menurun. Program meminta input bilangan bulat positif dari user dan kemudian menampilkan deret bilangan dari input tersebut hingga 1 secara berurutan menurun.

Algoritma Program :

1. Mulai program
2. Deklarasi variabel n bertipe integer
3. Baca input dari user ke variabel n
4. Panggil fungsi rekursif baris(n)
5. Di dalam fungsi baris:
 - Jika bilangan == 1:
 - * Cetak 1
 - * Selesai rekursi
 - Jika tidak:
 - * Cetak bilangan
 - * Panggil baris(bilangan - 1)
6. Selesai program

Cara Kerja Program :

1. Program menerima input angka N dari user
2. Input tersebut dikirim ke fungsi rekursif baris(N)
3. Fungsi baris() bekerja dengan aturan:
Jika angka = 1: cetak 1 dan selesai (basis rekursi)
Jika angka > 1: cetak angka tersebut, lalu panggil baris(angka-1)

2. Guided 2

Soal Studi Case

Menghitung hasil penjumlahan 1 hingga n

Base – case: $n == 1$

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))
}

```

Screenshot Output

```

PS D:\Alpro2> go run "d:\Alpro2\Modul 6\guided2.go"
6
21
PS D:\Alpro2>

```

Deskripsi Program :

Program ini menghitung penjumlahan dari semua bilangan bulat positif dari 1 sampai **n**, dengan menggunakan rekursi.

Algoritma Program :

Program ini menggunakan fungsi penjumlahan yang menerima satu parameter integer **n** dan mengembalikan jumlah semua bilangan bulat positif dari 1 sampai **n**.

- **Basis Kasus:** Jika **n** sama dengan 1, fungsi akan mengembalikan 1.
- **Langkah Rekursi:** Jika **n** tidak sama dengan 1, fungsi akan mengembalikan hasil penjumlahan **n** dengan hasil pemanggilan penjumlahan dengan parameter **n-1**.

Cara Kerja Program :

1. **Meminta Input:** Program meminta pengguna untuk memasukkan nilai **n**.
2. **Memanggil Fungsi:** Program memanggil fungsi **penjumlahan** dengan nilai **n** yang telah dimasukkan.
3. **Rekursi:** Fungsi **penjumlahan** secara rekursif akan memanggil dirinya sendiri dengan nilai **n-1** sampai nilai **n** mencapai 1.
4. **Basis Kasus:** Ketika nilai **n** mencapai 1, fungsi **penjumlahan** akan mengembalikan nilai 1.

5. **Menghitung Total:** Fungsi **penjumlahan** secara bergantian akan menjumlahkan nilai **n** dengan nilai yang dikembalikan dari setiap pemanggilan fungsi rekursif sebelumnya.
6. **Menampilkan Hasil:** Program menampilkan hasil penjumlahan dari semua bilangan bulat positif dari 1 sampai **n**.

3. Guided 3

Soal Studi Case

Mencari dua pangkat n atau 2^n

Base – case: $n == 0$

Sourcecode

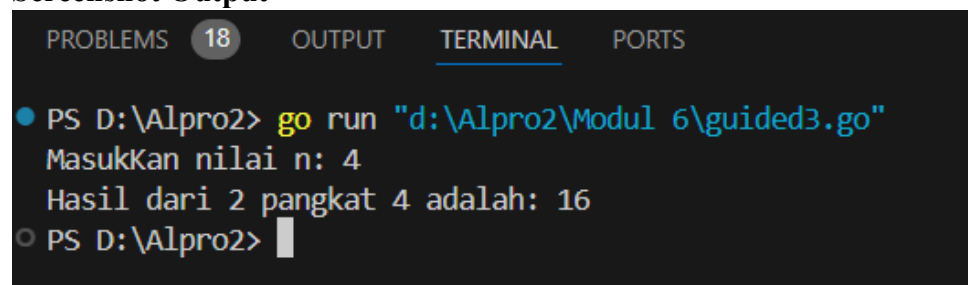
```
package main

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.Scan(&n)
    fmt.Println("Hasil dari 2 pangkat", n,
"adalah:", pangkat(n))
}
```

Screenshot Output



```
PROBLEMS 18 OUTPUT TERMINAL PORTS
PS D:\Alpro2> go run "d:\Alpro2\Modul 6\guided3.go"
Masukkan nilai n: 4
Hasil dari 2 pangkat 4 adalah: 16
PS D:\Alpro2>
```

Deskripsi Program :

Program ini menghitung nilai 2 pangkat n dengan menggunakan fungsi rekursif. Fungsi `pangkat(n)` akan memanggil dirinya sendiri dengan nilai n yang dikurangi 1 hingga n mencapai 0.

Algoritma Program :

1. **Input:** Program meminta pengguna untuk memasukkan nilai n.
2. **Fungsi Rekursif:**
 - Jika $n = 0$, kembalikan nilai 1 ($2^0 = 1$).
 - Jika $n > 0$, kalikan 2 dengan hasil rekursif **pangkat(n-1)**.
3. **Output:** Program menampilkan hasil dari 2^n .

Cara Kerja Program :

1. Ketika program dijalankan, fungsi main() meminta pengguna memasukkan nilai n.
2. Fungsi pangkat(n) dipanggil dengan nilai n yang dimasukkan.
3. Jika $n = 0$, fungsi pangkat(n) mengembalikan nilai 1 dan proses rekursi selesai.
4. Jika $n > 0$, fungsi pangkat(n) akan memanggil dirinya sendiri dengan nilai n-1.
5. Proses rekursi akan berulang hingga n mencapai 0.
6. Setelah n mencapai 0, fungsi pangkat(n) mengembalikan nilai 1 dan proses rekursi berakhir.
7. Hasil dari rekursi dikalikan dengan 2 pada setiap tahap rekursi, sehingga menghasilkan nilai 2^n .
8. Fungsi main() menampilkan hasil dari 2^n .

4. Guided 4

Soal Studi Case

Mencari nilai factorial atau n!

Base – case: $n == 0$ atau $n == 1$

Sourcecode

```
package main

import "fmt"

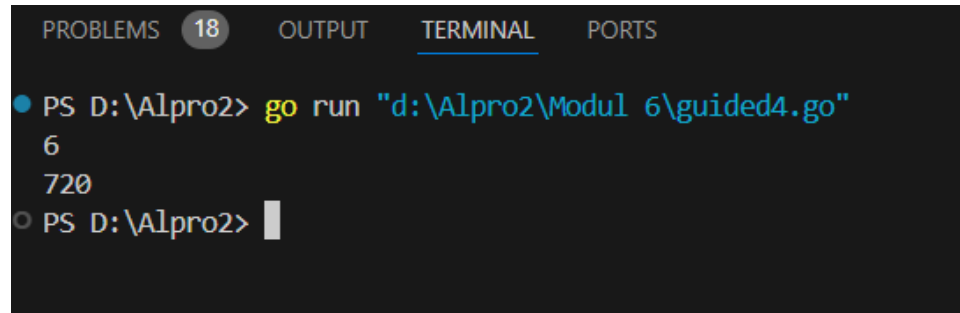
var n int

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

func main() {
    fmt.Scan(&n)
```

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

Screenshot Output



```
PROBLEMS 18 OUTPUT TERMINAL PORTS  
PS D:\Alpro2> go run "d:\Alpro2\Modul 6\guided4.go"  
6  
720  
PS D:\Alpro2>
```

Deskripsi Program :

Program ini menghitung faktorial dari sebuah bilangan bulat positif yang diinput oleh pengguna. Faktorial dari sebuah bilangan bulat positif adalah perkalian semua bilangan bulat positif dari 1 sampai bilangan tersebut.

Algoritma Program :

1. **Input:** Program meminta pengguna untuk memasukkan sebuah bilangan bulat positif.
2. **Fungsi Faktorial:** Program memanggil fungsi **faktorial(n)** yang menerima bilangan bulat positif **n** sebagai parameter.
3. **Kondisi Basis:** Jika **n** sama dengan 0 atau 1, fungsi mengembalikan nilai 1.
4. **Rekursi:** Jika **n** lebih besar dari 1, fungsi mengembalikan hasil perkalian antara **n** dengan hasil fungsi **faktorial(n-1)**.
5. **Output:** Program mencetak hasil dari fungsi **faktorial(n)**, yaitu faktorial dari bilangan yang diinput oleh pengguna.

Cara Kerja Program :

Ketika program dijalankan, pertama-tama program meminta input dari pengguna. Kemudian, program memanggil fungsi **faktorial(n)** dengan bilangan yang diinput oleh pengguna sebagai parameter.

III. UNGUIDED

1. Unguided 1

Soal Studi Case

Deret Fibonacci adlah 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 mengimplementasikan fungsi rekursif pada deret fibonacci tersebut.

n	0	1	2	3	4	5	6	7	8	9	10
S_n	0	1	1	2	3	5	8	13	21	34	55

Sourcecode

```
package main

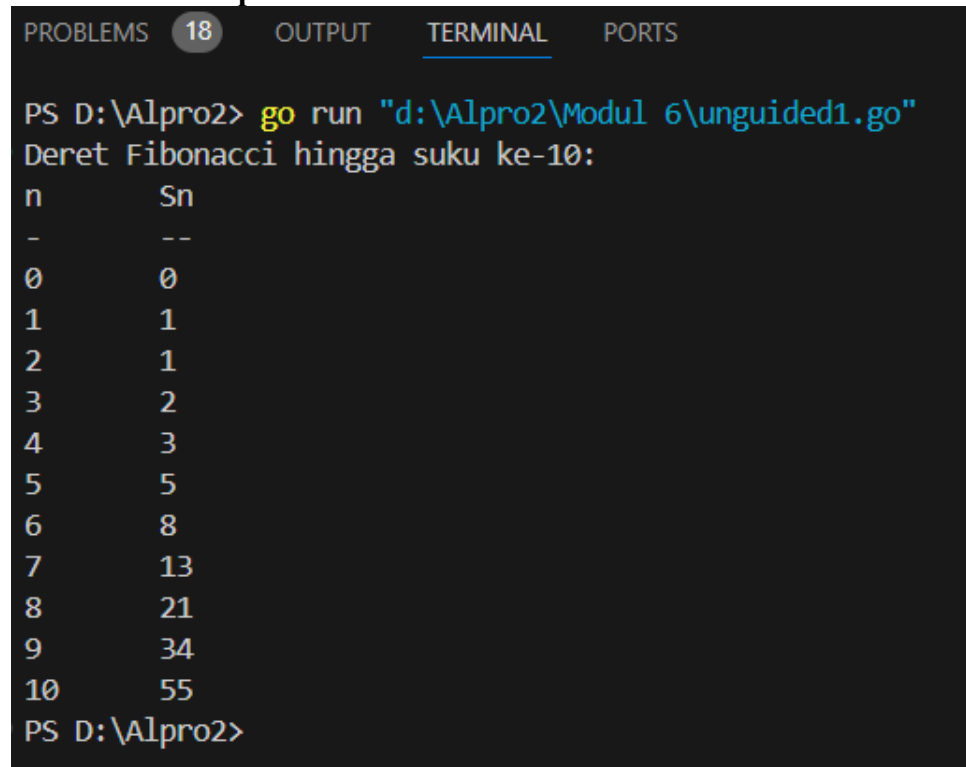
import "fmt"

func fibonacci(n int) int {
    // Basis rekursi: jika n <= 1, kembalikan n
    if n <= 1 {
        return n
    }
    // Rekursi:  $S_n = S_{n-1} + S_{n-2}$ 
    return fibonacci(n-1) + fibonacci(n-2)
}

func main() {
    // Menampilkan deret Fibonacci hingga suku ke-10
    fmt.Println("Deret Fibonacci hingga suku ke-10:")
    fmt.Println("n\tSn")
    fmt.Println("-\t--")

    // Loop untuk menghitung dan menampilkan setiap suku
    for i := 0; i <= 10; i++ {
        fmt.Printf("%d\t%d\n", i, fibonacci(i))
    }
}
```

Screenshoot Output



```
PROBLEMS 18 OUTPUT TERMINAL PORTS

PS D:\Alpro2> go run "d:\Alpro2\Modul 6\unguided1.go"
Deret Fibonacci hingga suku ke-10:
n      Sn
-      --
0      0
1      1
2      1
3      2
4      3
5      5
6      8
7      13
8      21
9      34
10     55
PS D:\Alpro2>
```

Deskripsi Program :

Program ini bertujuan untuk menghitung dan menampilkan deret Fibonacci hingga suku ke-10. Deret Fibonacci adalah barisan bilangan di mana setiap suku adalah hasil penjumlahan dari dua suku sebelumnya.

Algoritma Program :

Program ini menggunakan rekursi untuk menghitung deret Fibonacci. Algoritma rekursifnya adalah sebagai berikut:

1. Basis Rekursi:

- Jika $n \leq 1$, maka fungsi mengembalikan nilai n . Ini merupakan kasus dasar dari rekursi.

2. Rekursi:

- Jika $n > 1$, fungsi memanggil dirinya sendiri dua kali:
 - **fibonacci(n-1)**: Memanggil fungsi dengan n dikurangi 1.
 - **fibonacci(n-2)**: Memanggil fungsi dengan n dikurangi 2.
- Hasil dari kedua pemanggilan tersebut kemudian dijumlahkan dan dikembalikan sebagai hasil akhir.

Cara Kerja Program :

1. Program memulai dengan mendeklarasikan fungsi fibonacci(n). Fungsi ini menerima satu parameter n yang merupakan urutan suku Fibonacci yang ingin dihitung.
2. Fungsi fibonacci(n) memeriksa apakah n lebih kecil atau sama dengan 1. Jika ya, fungsi mengembalikan n. Ini adalah kasus dasar dari rekursi.
3. Jika n lebih besar dari 1, fungsi memanggil dirinya sendiri dua kali dengan n-1 dan n-2. Hasil dari kedua pemanggilan tersebut kemudian dijumlahkan dan dikembalikan.
4. Program utama (main()) mendeklarasikan sebuah loop for yang berulang 10 kali. Pada setiap iterasi loop, program memanggil fungsi fibonacci(i) untuk menghitung suku Fibonacci ke-i.
5. Nilai suku Fibonacci yang dihasilkan oleh fungsi fibonacci(i) kemudian ditampilkan di layar dengan format n\tSn.

2. Unguided 2

Soal Studi Case

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

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	* ** *** **** *****
2	1	*
3	3	* ** ***

Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi rekursif untuk mencetak bintang
func printStars(n int, current int) {
```

```

        // Basis rekursi: jika current > n, berhenti
        if current > n {
            return
        }

        // Cetak bintang sebanyak current
        for i := 0; i < current; i++ {
            fmt.Print("*")
        }
        fmt.Println()

        // Panggil rekursif untuk baris selanjutnya
        printStars(n, current+1)
    }

func main() {
    var jumlahTest int
    fmt.Print("Masukkan jumlah test case: ")
    fmt.Scan(&jumlahTest)

    // Array untuk menyimpan hasil input
    inputs := make([]int, jumlahTest)

    // Membaca input
    for i := 0; i < jumlahTest; i++ {
        fmt.Printf("Masukan #%d: ", i+1)
        fmt.Scan(&inputs[i])
    }

    // Memproses setiap test case
    fmt.Println("\nKeluaran:")
    for i := 0; i < jumlahTest; i++ {
        fmt.Printf("No\tMasukan\tKeluaran\n")
        fmt.Printf("%d\t%d\t\n", i+1, inputs[i])
        printStars(inputs[i], 1)
        fmt.Println()
    }
}

```

Screenshot Output

```
PROBLEMS 18 OUTPUT TERMINAL PORTS
PS D:\Alpro2> go run "d:\Alpro2\Modul 6\unguided2.go"
Masukkan jumlah test case: 3
Masukan #1: 5
Masukan #2: 1
Masukan #3: 3

Keluaran:
No      Masukan Keluaran
1       5
*
**
***
****
*****

No      Masukan Keluaran
2       1
*

No      Masukan Keluaran
3       3
*
**
***
```

Deskripsi Program :

Program ini dirancang untuk menerima input berupa sejumlah bilangan bulat positif dan menampilkan pola bintang berbentuk segitiga siku-siku dengan jumlah baris yang sesuai dengan setiap bilangan input. Program ini menggunakan fungsi rekursif **printStars** untuk mencetak pola bintang.

Algoritma Program :

1. Meminta Input: Program meminta pengguna untuk memasukkan jumlah test case dan bilangan bulat positif untuk setiap test case.
2. Pemrosesan: Program melakukan iterasi melalui setiap bilangan input dan memanggil fungsi **printStars** untuk mencetak pola bintang.
3. Fungsi **printStars**:

- Fungsi `printStars` menerima dua parameter: `n` (jumlah baris bintang) dan `current` (baris bintang yang sedang dicetak).
 - Fungsi `printStars` melakukan pengecekan kondisi dasar: Jika `current` lebih besar dari `n`, maka fungsi berhenti.
 - Fungsi `printStars` mencetak bintang sebanyak `current` kali untuk baris bintang yang sedang dicetak.
 - Fungsi `printStars` memanggil dirinya sendiri dengan parameter `n` dan `current + 1` untuk mencetak baris bintang selanjutnya.
4. Output: Program menampilkan hasil pola bintang untuk setiap input bilangan.

Cara Kerja Program :

1. Program memulai dengan meminta pengguna untuk memasukkan jumlah test case yang diinginkan.
2. Kemudian, program meminta pengguna memasukkan setiap bilangan bulat positif untuk setiap test case.
3. Untuk setiap bilangan input, program memanggil fungsi **`printStars`** dengan parameter **`n`** yang sama dengan bilangan input dan **`current`** yang diinisialisasi menjadi 1.
4. Fungsi **`printStars`** menjalankan proses rekursif untuk mencetak pola bintang.
 - Pertama, fungsi mengecek kondisi dasar: jika **`current`** lebih besar dari **`n`**, maka fungsi berhenti.
 - Jika kondisi dasar tidak terpenuhi, fungsi akan mencetak **`current`** bintang pada baris yang sedang dicetak.
 - Setelah mencetak bintang, fungsi memanggil dirinya sendiri dengan **`current + 1`** untuk mencetak baris bintang berikutnya.
5. Proses rekursif berlanjut hingga kondisi dasar terpenuhi, sehingga seluruh pola bintang dicetak untuk setiap bilangan input.
6. Program menampilkan output berupa pola bintang yang telah dicetak untuk setiap test case.

3. Unguided 3

Soal Studi Case

Buatlan program yang mengimplementasikan rekursif untuk menampilkan factor 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 factor dari N (terurut dari 1 hingga N ya).

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	1 5
2	12	1 2 3 4 6 12

Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi rekursif untuk mencari faktor
func findFactors(n int, current int) {
    // Basis rekursi: jika current > n, berhenti
    if current > n {
        return
    }

    // Jika current adalah faktor dari n
    if n%current == 0 {
        fmt.Printf("%d ", current)
    }

    // Panggil rekursif untuk bilangan
    selanjutnya
    findFactors(n, current+1)
}

func main() {
    var jumlahTest int
    fmt.Print("Masukkan jumlah test case: ")
    fmt.Scan(&jumlahTest)

    // Array untuk menyimpan input
```

```

inputs := make([]int, jumlahTest)

// Membaca input
for i := 0; i < jumlahTest; i++ {
    fmt.Printf("Masukan #%d: ", i+1)
    fmt.Scan(&inputs[i])
}

// Memproses dan menampilkan hasil
fmt.Println("\nNo\tMasukan\tKeluaran")
for i := 0; i < jumlahTest; i++ {
    fmt.Printf("%d\t%d\t", i+1, inputs[i])
    findFactors(inputs[i], 1)
    fmt.Println()
}
}

```

Screenshot Output

```

PS D:\Alpro2> go run "d:\Alpro2\Modul 6\unguided3.go"
Masukkan jumlah test case: 2
Masukan #1: 5
Masukan #2: 12

No      Masukan Keluaran
1       5      1 5
2       12     1 2 3 4 6 12
PS D:\Alpro2>

```

Deskripsi Program

Program ini dirancang untuk menemukan dan menampilkan faktor-faktor dari sebuah bilangan bulat positif. Program ini menggunakan pendekatan rekursif untuk melakukan proses pencarian faktor.

Algoritma Program :

1. **Input:** Program menerima jumlah test case (jumlah bilangan yang akan dicari faktornya) dan bilangan bulat positif untuk setiap test case.
2. **Fungsi Rekursif findFactors(n, current):**
 - **Basis rekursi:** Fungsi akan berhenti jika **current** lebih besar dari **n**.
 - **Periksa Faktor:** Jika **n** dapat dibagi habis dengan **current**, maka **current** merupakan faktor dari **n** dan akan dicetak.

- **Rekursi:** Fungsi memanggil dirinya sendiri dengan **current** ditambah 1, sehingga secara berurutan memeriksa setiap bilangan dari 1 hingga **n** sebagai calon faktor.
3. **Output:** Program menampilkan tabel yang menunjukkan nomor test case, bilangan input, dan daftar faktor-faktor dari bilangan tersebut.

Cara Kerja Program :

1. Program dimulai dengan meminta pengguna untuk memasukkan jumlah test case.
2. Program meminta pengguna untuk memasukkan bilangan bulat positif untuk setiap test case, dan menyimpannya dalam sebuah array inputs.
3. Program kemudian memulai perulangan untuk setiap test case.
4. Untuk setiap test case, program memanggil fungsi findFactors dengan bilangan input sebagai n dan current diinisialisasi dengan 1.
5. Fungsi findFactors akan secara rekursif memeriksa setiap bilangan dari 1 hingga n untuk menentukan apakah merupakan faktor dari n.
6. Jika suatu bilangan current merupakan faktor dari n, maka bilangan tersebut dicetak.
7. Program mengulangi proses ini untuk setiap test case, menampilkan hasil dalam bentuk tabel.

4. Unguided 4

Soal Studi Case

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.

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	5 4 3 2 1 2 3 4 5
2	9	9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9

Sourcecode

```
package main

import (
    "fmt"
)
```

```

// Fungsi rekursif untuk menurun
func printDescending(n int, current int) {
    if current < 1 {
        return
    }

    fmt.Printf("%d ", current)

    printDescending(n, current-1)
}

// Fungsi rekursif untuk menaik
func printAscending(n int, current int) {
    if current > n {
        return
    }

    fmt.Printf("%d ", current)

    printAscending(n, current+1)
}

// Fungsi untuk mencetak pola lengkap
func printPattern(n int) {
    printDescending(n, n)

    printAscending(n, 2)
}

func main() {
    var jumlahTest int
    fmt.Print("Masukkan jumlah test case: ")
    fmt.Scan(&jumlahTest)

    // Array untuk menyimpan input
    inputs := make([]int, jumlahTest)

    // Membaca input
    for i := 0; i < jumlahTest; i++ {
        fmt.Printf("Masukan #%d: ", i+1)
        fmt.Scan(&inputs[i])
    }

    // Memproses dan menampilkan hasil
    fmt.Println("\nNo\tMasukan\tKeluaran")
    for i := 0; i < jumlahTest; i++ {

```

```

        fmt.Printf("%d\t%d\t", i+1, inputs[i])
        printPattern(inputs[i])
        fmt.Println()
    }
}

```

Screenshot Output

```

PROBLEMS 18 OUTPUT TERMINAL PORTS
PS D:\Alpro2> go run "d:\Alpro2\Modul 6\unguided4.go"
Masukkan jumlah test case: 2
Masukan #1: 5
Masukan #2: 9

No      Masukan Keluaran
1       5      5 4 3 2 1 2 3 4 5
2       9      9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9
PS D:\Alpro2>

```

Deskripsi Program :

Program ini dirancang untuk menampilkan pola bilangan dengan pola menurun dan menaik. Program menerima input berupa jumlah test case dan nilai integer untuk setiap test case. Untuk setiap input, program akan mencetak pola bilangan menurun dari nilai integer tersebut ke 1, lalu dilanjutkan dengan pola menaik dari 2 hingga nilai integer tersebut.

Algoritma Program :

1. Meminta input jumlah test case dari pengguna.
2. Meminta input nilai integer untuk setiap test case.
3. Untuk setiap test case:
 - Memanggil fungsi `printPattern()` dengan nilai integer sebagai argumen.
 - Fungsi `printPattern()` memanggil fungsi `printDescending()` dan `printAscending()`.
 - Fungsi `printDescending()` mencetak bilangan dari nilai integer ke 1 secara menurun.
 - Fungsi `printAscending()` mencetak bilangan dari 2 hingga nilai integer secara menaik.
4. Menampilkan hasil pola bilangan untuk setiap test case.

Cara Kerja Program :

1. Program di run
2. Program dimulai dengan meminta input jumlah test case dari pengguna.
3. Program kemudian membuat array **inputs** dengan ukuran sesuai jumlah test case.
4. Program membaca input nilai integer untuk setiap test case dan menyimpannya dalam array **inputs**.
5. Program kemudian menampilkan tabel header dengan kolom "No", "Masukan", dan "Keluaran".
6. Program iterasi melalui array **inputs** dan untuk setiap nilai integer:
 - Program memanggil fungsi **printPattern()** dengan nilai integer sebagai argumen.
 - Fungsi **printPattern()** pertama-tama memanggil fungsi **printDescending()** dengan nilai integer dan nilai integer sebagai argumen.
 - Fungsi **printDescending()** menggunakan rekursi untuk mencetak bilangan dari nilai integer ke 1 secara menurun.
 - Fungsi **printDescending()** mencetak bilangan saat ini dan memanggil dirinya sendiri dengan nilai integer yang dikurangi 1. Proses ini berulang hingga nilai integer mencapai 1.
 - Setelah selesai mencetak bilangan secara menurun, fungsi **printPattern()** memanggil fungsi **printAscending()** dengan nilai integer dan 2 sebagai argumen.
 - Fungsi **printAscending()** menggunakan rekursi untuk mencetak bilangan dari 2 hingga nilai integer secara menaik.
 - Fungsi **printAscending()** mencetak bilangan saat ini dan memanggil dirinya sendiri dengan nilai integer yang ditambah 1. Proses ini berulang hingga nilai integer mencapai nilai integer awal.

5. Unguided 5

Soal Studi Case

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

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	1 3 5
2	20	1 3 5 7 9 11 13 15 17 19

Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi rekursif untuk mencetak bilangan ganjil
func printOddNumbers(n int, current int) {
    // Basis rekursi: berhenti jika current > n
    if current > n {
        return
    }

    // Jika current adalah bilangan ganjil, cetak
    if current%2 != 0 {
        fmt.Printf("%d ", current)
    }

    // Panggil rekursif dengan bilangan selanjutnya
    printOddNumbers(n, current+1)
}

func main() {
    var jumlahTest int
    fmt.Print("Masukkan jumlah test case: ")
    fmt.Scan(&jumlahTest)

    // Array untuk menyimpan input
    inputs := make([]int, jumlahTest)

    // Membaca input
    for i := 0; i < jumlahTest; i++ {
        fmt.Printf("Masukan #%d: ", i+1)
        fmt.Scan(&inputs[i])
    }

    // Memproses dan menampilkan hasil
```

```

        fmt.Println("\nNo\tMasukan\tKeluaran")
        for i := 0; i < jumlahTest; i++ {
            fmt.Printf("%d\t%d\t", i+1, inputs[i])
            printOddNumbers(inputs[i], 1) // Mulai
            dari 1
            fmt.Println()                // Pindah
            baris untuk test case berikutnya
        }
    }
}

```

Screenshot Output

```

PS D:\Alpro2> go run "d:\Alpro2\Modul 6\unguided5.go"
Masukkan jumlah test case: 2
Masukan #1: 5
Masukan #2: 20

No      Masukan  Keluaran
1       5        1 3 5
2       20      1 3 5 7 9 11 13 15 17 19
PS D:\Alpro2>

```

Deskripsi Program :

Program ini dirancang untuk menerima sejumlah input bilangan bulat positif dan mencetak semua bilangan ganjil dari 1 hingga setiap bilangan input. Program menggunakan rekursi untuk menyelesaikan tugas ini.

Algoritma Program :

- Input: Program menerima dua input:
 - jumlahTest: Jumlah bilangan bulat yang akan diproses.
 - inputs: Array bilangan bulat yang akan diproses.
- Iterasi: Program melakukan iterasi melalui setiap bilangan dalam array inputs.
- Panggilan Rekursif: Untuk setiap bilangan input, program memanggil fungsi printOddNumbers dengan dua parameter:
 - n: Bilangan input saat ini.
 - current: Nilai awal untuk perulangan rekursif, umumnya dimulai dari 1.
- Basis Rekursi: Fungsi printOddNumbers memiliki basis rekursi yaitu $current > n$. Artinya, rekursi berhenti ketika current lebih besar daripada n.

5. Pencetakan Bilangan Ganjil: Di dalam fungsi printOddNumbers, program memeriksa apakah current adalah bilangan ganjil ($\text{current} \% 2 \neq 0$). Jika ya, bilangan tersebut dicetak.
6. Rekursi: Fungsi printOddNumbers memanggil dirinya sendiri dengan current yang ditambah 1. Hal ini memastikan bahwa semua bilangan ganjil dalam rentang 1 hingga n akan diproses.
7. Output: Program mencetak nomor test case, bilangan input, dan semua bilangan ganjil dari 1 hingga bilangan input, dipisahkan oleh spasi.

Cara Kerja Program :

1. Run Program
2. Input: Program menerima input berupa jumlah test case dan array bilangan bulat yang akan diproses.
3. Iterasi Test Case: Program melakukan iterasi melalui setiap test case.
4. Panggilan Fungsi Rekursif: Fungsi printOddNumbers dipanggil untuk setiap test case.
5. Pencetakan Bilangan Ganjil: Fungsi printOddNumbers secara rekursif memeriksa setiap bilangan dari current hingga n. Jika bilangan ganjil ditemukan, maka bilangan tersebut dicetak.
6. Rekursi: Fungsi printOddNumbers memanggil dirinya sendiri dengan nilai current yang bertambah satu, sehingga iterasi rekursif terus berlanjut hingga mencapai n.
7. Output: Hasil keluaran

6. Unguided 6

Soal Studi Case

Buatlah program yang mengimplementasikan rekursif untuk mencari hasil pangkat 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".

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	2 2	4
2	5 3	125

Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi rekursif untuk menghitung pangkat
func power(base int, exponent int) int {
    if exponent == 0 {
        return 1
    }

    return base * power(base, exponent-1)
}

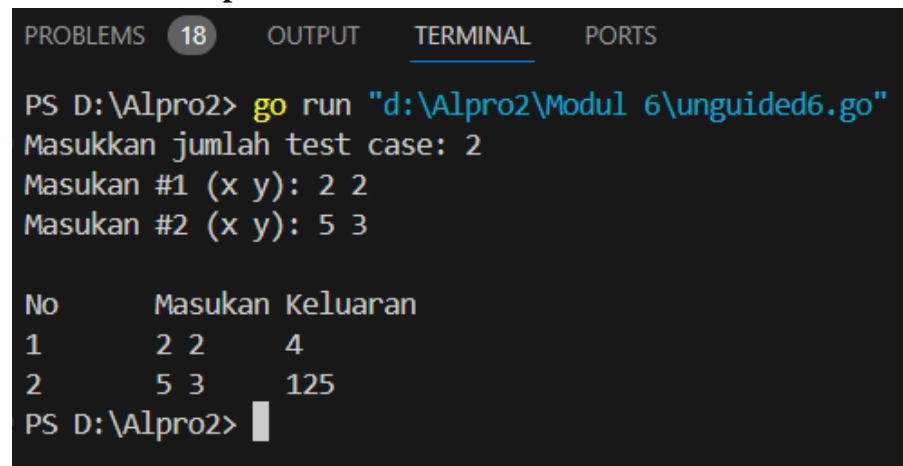
func main() {
    var jumlahTest int
    fmt.Print("Masukkan jumlah test case: ")
    fmt.Scan(&jumlahTest)

    // Array untuk menyimpan input
    baseInputs := make([]int, jumlahTest)
    exponentInputs := make([]int, jumlahTest)

    // Membaca input
    for i := 0; i < jumlahTest; i++ {
        fmt.Printf("Masukan #%d (x y): ", i+1)
        fmt.Scan(&baseInputs[i],
            &exponentInputs[i])
    }

    // Memproses dan menampilkan hasil
    fmt.Println("\nNo\tMasukan\tKeluaran")
    for i := 0; i < jumlahTest; i++ {
        result := power(baseInputs[i],
            exponentInputs[i])
        fmt.Printf("%d\t%d %d\t%d\n",
            i+1,
            baseInputs[i],
            exponentInputs[i],
            result)
    }
}
```


Screenshot Output



```
PROBLEMS 18 OUTPUT TERMINAL PORTS

PS D:\Alpro2> go run "d:\Alpro2\Modul 6\unguided6.go"
Masukkan jumlah test case: 2
Masukan #1 (x y): 2 2
Masukan #2 (x y): 5 3

No      Masukan  Keluaran
1       2 2      4
2       5 3     125
PS D:\Alpro2> 
```

Deskripsi Program :

Program ini menghitung pangkat dari suatu bilangan menggunakan rekursi. Program ini meminta pengguna untuk memasukkan jumlah test case, dan untuk setiap test case, meminta pengguna memasukkan nilai basis dan eksponen. Kemudian, program menghitung pangkat dari basis dan eksponen yang diberikan menggunakan fungsi rekursif `power()` dan menampilkan hasilnya.

Algoritma Program :

1. **Meminta Input:** Program meminta pengguna untuk memasukkan jumlah test case. Kemudian, program membaca nilai basis dan eksponen untuk setiap test case dari pengguna.
2. **Fungsi Rekursif `power()`:** Fungsi `power()` menerima dua parameter: basis dan eksponen. Fungsi ini berfungsi sebagai berikut:
 - **Kondisi Dasar:** Jika eksponen adalah 0, fungsi mengembalikan 1.
 - **Rekursi:** Jika eksponen tidak sama dengan 0, fungsi mengalikan basis dengan hasil rekursi fungsi `power()` dengan basis yang sama dan eksponen dikurangi 1.
3. **Memproses dan Menampilkan Hasil:** Program iterasi melalui setiap test case dan menghitung pangkat menggunakan fungsi `power()`. Kemudian, program menampilkan nomor test case, nilai basis, eksponen, dan hasilnya.

Cara Kerja Program :

Program pertama-tama meminta pengguna untuk memasukkan jumlah test case. Kemudian, program membaca nilai basis dan eksponen dari pengguna

untuk setiap test case dan menyimpannya dalam array. Selanjutnya, program mengulangi setiap test case dan memanggil fungsi `power()` untuk menghitung pangkat dari basis dan eksponen yang diberikan. Fungsi `power()` bekerja secara rekursif, mengalikan basis dengan hasil fungsi `power()` itu sendiri dengan eksponen yang lebih rendah sampai eksponen mencapai 0. Ketika eksponen mencapai 0, fungsi mengembalikan 1. Hasil dari fungsi `power()` kemudian ditampilkan bersama dengan nomor test case, nilai basis, dan eksponen.

IV. KESIMPULAN

Rekursi adalah teknik dalam pemrograman yang memungkinkan suatu subprogram memanggil dirinya sendiri. Hal ini memungkinkan untuk memecahkan masalah kompleks dengan memecahnya menjadi sub-masalah yang sama dengan masalah awal.

Setiap algoritma rekursif terdiri dari dua komponen utama:

1. **Base Case:** Kondisi yang menghentikan proses rekursif, biasanya merupakan kondisi dasar atau sederhana yang dapat dipecahkan secara langsung.
2. **Recursive Case:** Kondisi yang memanggil subprogram itu sendiri, memecah masalah menjadi sub-masalah yang lebih kecil.

Teknik rekursi bisa menjadi alternatif untuk struktur perulangan. Base Case berperan penting dalam rekursi untuk menghentikan proses dan mencegah program berjalan tak terbatas.

V. REFERENSI

[1] Modul 6 Praktikum Algoritma 2