

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 6
REKURSIF**



Disusun Oleh :

Agnes Refilina Fiska / 2311102126

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

Rekursif adalah suatu teknik dalam pemrograman di mana sebuah subprogram (baik fungsi maupun prosedur) dapat memanggil dirinya sendiri. Teknik ini digunakan untuk menyelesaikan masalah dengan cara mengurai masalah utama menjadi sub-masalah yang lebih kecil dan identik.

	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	

Jika kita mengeksekusi perintah cetak(5), maka subprogram tersebut akan menampilkan angka 5, 6, 7, 8, 9, dan seterusnya, tanpa henti. Dalam hal ini, setiap pemanggilan subprogram cetak() akan membuat nilai x bertambah 1 (increment by one) secara terus-menerus.

Teknik rekursif ini merupakan alternatif untuk menggantikan struktur kontrol perulangan dengan memanfaatkan subprogram (baik fungsi maupun prosedur). Untuk menghentikan proses rekursif, digunakan percabangan (if-then).

Base Case adalah kondisi yang menghentikan proses rekursif. Memahami base case adalah langkah terpenting dalam merancang program rekursif, karena tidak mungkin membuat program semacam itu tanpa mengetahui base case terlebih dahulu. Sebaliknya,

Recursive Case adalah kondisi di mana pemanggilan dirinya sendiri dilakukan. Recursive case adalah negasi dari base case. Setiap algoritma rekursif selalu memiliki padanan di dalam bentuk algoritma iteratif.

B. Komponen Rekursif

Algoritma rekursif terdiri dari dua komponen utama:

1. **Base Case (Basis):**
 - Bagian dari algoritma yang menghentikan proses rekursif. Ini adalah komponen paling penting di dalam sebuah rekursi.
2. **Recursive Case:**
 - Bagian yang melakukan pemanggilan subprogramnya. Ini adalah kondisi di mana algoritma terus memanggil dirinya sendiri hingga mencapai base case.

Dengan pemahaman dasar ini, kita dapat merancang dan mengimplementasikan algoritma yang memanfaatkan teknik rekursif secara efektif.

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

A screenshot of a terminal window showing the execution of a Go program. The terminal has tabs for PROBLEMS (6), OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The command executed is 'go run -Refilina-Fiska\ALPRO_2\Modul6\guided1.go'. The output shows the numbers 2, 2, and 1, each on a new line, indicating the program is printing the sequence of numbers from n down to 1. The prompt 'PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2>' is visible at the bottom.

```
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> go run
-Refilina-Fiska\ALPRO_2\Modul6\guided1.go
2
2
1
PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> |
```

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)
    }
}

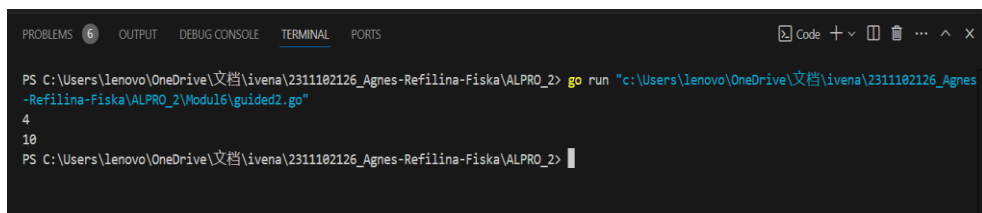
func main() {
    var n int

    fmt.Scan(&n)

    fmt.Println(penjumlahan(n))
}

```

Screenshoot Output



```

PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> go run "c:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2\Modul6\guided2.go"
4
PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2>

```

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 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> go run "c:\Users\lenovo\
-Refilina-Fiska\ALPRO_2\Modul6\guided3.go"
Masukan nilai n: 6
Hasil dari 2 pangkat 6 adalah: 64
PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> |
```

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 ($2 \text{ pangkat } 0 = 1$).
 - Jika $n > 0$, kalikan 2 dengan hasil rekursif **pangkat(n-1)**.
3. **Output:** Program menampilkan hasil dari 2 pangkat 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 pangkat n.
8. Fungsi main() menampilkan hasil dari 2 pangkat n.

4. Guided 4 Soal Studi Case

Mencari nilai factorial atau n! Base

– case: $n == 0$ atau $n == 1$

Sourcecode

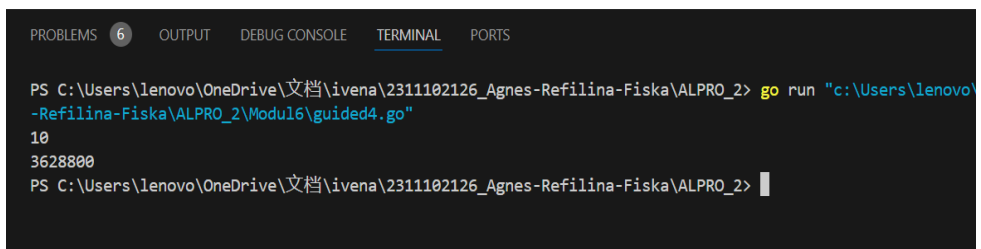
```
package main

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 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> go run "c:\Users\lenovo\
-Refilina-Fiska\ALPRO_2\Modul6\guided4.go"
10
3628800
PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> |
```

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 case
    if n <= 1 {
        return n
    }
    // Recursive case
    return fibonacci(n-1) + fibonacci(n-2)
}

func main() {
    // Print header
    fmt.Printf("n : ")
    for i := 0; i <= 10; i++ {
        fmt.Printf("%-4d", i)
    }
    fmt.Println()

    // Print Sn values
```

```

    fmt.Printf("Sn : ")
    for i := 0; i <= 10; i++ {
        fmt.Printf("%-4d", fibonacci(i))
    }
    fmt.Println()
}

```

Screenshoot Output

```

PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> go run "-Refilina-Fiska\ALPRO_2\Modul6\unguided1.go"
n : 0 1 2 3 4 5 6 7 8 9 10
Sn : 0 1 1 2 3 5 8 13 21 34 55
PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2>

```

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 mendefinisikan 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()) mendefinisikan 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 case
if n == 0 {
    return
}

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

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

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

    for i := 1; i <= testCases; i++ {
        var n int
        fmt.Printf("\nTest Case %d\n", i)
        fmt.Print("Masukkan N: ")
        fmt.Scan(&n)

        fmt.Printf("Keluaran:\n")
        printStars(n, 1)
    }
}
```

Screenshot Output

```
PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> go run
-Refilina-Fiska\ALPRO_2\Modul6\unguided2.go
Masukkan jumlah test cases: 3

Test Case 1
Masukkan N: 5
Keluaran:
*
**
***
****
*****

Test Case 2
Masukkan N: 1
Keluaran:
*

Test Case 3
Masukkan N: 3
Keluaran:
*
**
***

PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> 
```

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 :

Program ini menggunakan fungsi rekursif untuk mencetak pola bintang. Setiap test case memungkinkan pengguna untuk menentukan berapa banyak baris bintang yang akan dicetak, dengan setiap baris memiliki jumlah bintang yang bertambah satu dari baris sebelumnya. Pendekatan rekursi ini menyederhanakan proses pencetakan bintang dengan memanggil kembali fungsi hingga semua baris yang diperlukan telah dicetak.

Dengan menggunakan struktur ini, penjelasan cara kerja program menjadi lebih jelas dan terorganisir, memudahkan pemahaman bagi pembaca.

3. Unguided 3 Soal Studi Case

Buatlah 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 case
    if current > n {
```

```

        return

    }

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

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

func main() {
    var testCases int

    fmt.Print("Masukkan jumlah test cases: ")
    fmt.Scan(&testCases)

    for i := 1; i <= testCases; i++ {
        var n int

        fmt.Printf("\nTest Case %d\n", i)
        fmt.Print("Masukkan N: ")
        fmt.Scan(&n)

        fmt.Printf("Keluaran: ")
        findFactors(n, 1)
    }
}

```

```

        fmt.Println()
    }
}

```

Screenshot Output

```

PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> go run "-Refilina-Fiska\ALPRO_2\Modul6\unguided3.go"
Masukkan jumlah test cases: 2

Test Case 1
Masukkan N: 5
Keluaran: 1 5

Test Case 2
Masukkan N: 12
Keluaran: 1 2 3 4 6 12
PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> 

```

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 mencetak barisan turun
func printDescending(n int, current int) {
    // Basis case
    if current < 1 {
        return
    }

    // Cetak angka current
    if current == 1 {
```

```

        fmt.Printf("%d ", current)
    } else {
        fmt.Printf("%d ", current)
    }

    // Panggil rekursif untuk angka selanjutnya
    printDescending(n, current-1)
}

// Fungsi rekursif untuk mencetak barisan naik
func printAscending(n int, current int) {
    // Basis case
    if current > n {
        return
    }

    // Cetak angka current
    fmt.Printf("%d ", current)

    // Panggil rekursif untuk angka selanjutnya
    printAscending(n, current+1)
}

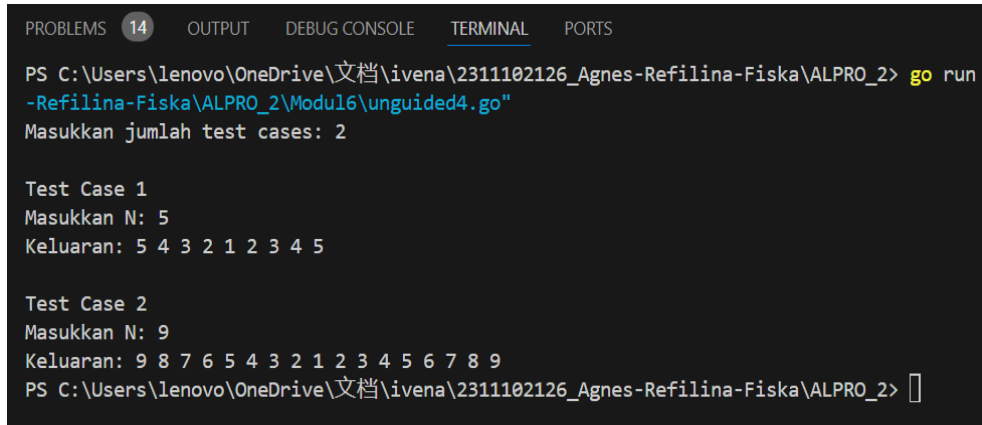
func main() {
    var testCases int
    fmt.Print("Masukkan jumlah test cases: ")
    fmt.Scan(&testCases)

    for i := 1; i <= testCases; i++ {
        var n int
        fmt.Printf("\nTest Case %d\n", i)
        fmt.Print("Masukkan N: ")
        fmt.Scan(&n)

        fmt.Printf("Keluaran: ")
        // Cetak barisan turun dari N ke 1
        printDescending(n, n)
        // Cetak barisan naik dari 2 ke N
        printAscending(n, 2)
        fmt.Println()
    }
}

```

Screenshot Output



```
PROBLEMS 14 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> go run
-Refilina-Fiska\ALPRO_2\Modul6\unguided4.go
Masukkan jumlah test cases: 2

Test Case 1
Masukkan N: 5
Keluaran: 5 4 3 2 1 2 3 4 5

Test Case 2
Masukkan N: 9
Keluaran: 9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9
PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> 
```

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 :

Program ini menggunakan dua fungsi rekursif untuk mencetak dua urutan bilangan. Fungsi `printDescending` mencetak bilangan dari N hingga 1, sedangkan fungsi `printAscending` mencetak bilangan dari 2 hingga N. Program ini memungkinkan pengguna untuk menjalankan beberapa test

cases, dan untuk setiap test case, pengguna dapat menentukan nilai N, yang akan digunakan untuk mencetak kedua urutan tersebut. Teknik rekursi digunakan di sini untuk menyederhanakan logika pencetakan dengan memanggil kembali fungsi untuk setiap angka yang akan dicetak.

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 case
    if current > n {
        return
    }
    // Jika current adalah bilangan ganjil
    if current%2 != 0 {
        if current == 1 {
            fmt.Printf("%d", current)
        } else {
            fmt.Printf(" %d", current)
        }
    }
}
```

```

    }

    // Panggil rekursif untuk angka selanjutnya
    printOddNumbers(n, current+1)
}

func main() {
    var testCases int

    fmt.Println("=====
=====")

    fmt.Println("    Program Deret Bilangan
Ganjil    ")

    fmt.Println("=====
=====")

    fmt.Print("\nMasukkan jumlah test cases: ")
    fmt.Scan(&testCases)

    for i := 1; i <= testCases; i++ {
        var n int

        fmt.Printf("\n-----
-----")

        fmt.Printf("\nTest Case %d:", i)
        fmt.Print("\nMasukkan N: ")
        fmt.Scan(&n)

        fmt.Printf("Keluaran: ")
        printOddNumbers(n, 1)
        fmt.Println()
    }
}

```

```

        fmt.Println("-----")
    ----")

    fmt.Println("\nProgram Selesai!")
}

```

Screenshot Output

```

PROBLEMS 16 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> go run
-Refilina-Fiska\ALPRO_2\Modul6\unguided5.go"
=====
    Program Deret Bilangan Ganjil
    =====

Masukkan jumlah test cases: 2

-----
Test Case 1:
Masukkan N: 5
Keluaran: 1 3 5

-----
Test Case 2:
Masukkan N: 20
Keluaran: 1 3 5 7 9 11 13 15 17 19
-----

Program Selesai!
PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> 

```

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.

4. 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 ($current \% 2 \neq 0$). 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 :

Program ini menggunakan fungsi rekursif untuk mencetak semua bilangan ganjil dari 1 hingga N yang ditentukan oleh pengguna. Pengguna dapat memasukkan beberapa test cases, dan untuk setiap test case, program akan meminta nilai N, lalu mencetak deret bilangan ganjil hingga N tersebut. Teknik rekursi memungkinkan program untuk menyelesaikan tugas ini dengan memanggil dirinya sendiri untuk setiap angka.

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, exp int) int {
    // Basis case
    if exp == 0 {
        return 1
    }

    // Recursive case
    return base * power(base, exp-1)
}

func main() {
    fmt.Println("=====
==")
    fmt.Println("    Program Menghitung
Pangkat    ")
    fmt.Println("=====
==")

    var testCases int
    fmt.Print("\nMasukkan jumlah test cases: ")
    fmt.Scan(&testCases)

    for i := 1; i <= testCases; i++ {
        var x, y int
        fmt.Printf("\n-----
-----")
        fmt.Printf("\nTest Case %d:", i)
        fmt.Print("\nMasukkan nilai x dan y
(dipisah spasi): ")
        fmt.Scan(&x, &y)

        result := power(x, y)
        fmt.Printf("Hasil %d pangkat %d: %d\n", x,
y, result)
    }

    fmt.Println("-----
-")
    fmt.Println("\nProgram Selesai!")
}

```


}

Screenshot Output

```
PROBLEMS 18 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> go run
-Refilina-Fiska\ALPRO_2\Modul6\unguided6.go"
=====
      Program Menghitung Pangkat
=====

Masukkan jumlah test cases: 2

-----
Test Case 1:
Masukkan nilai x dan y (dipisah spasi): 2 2
Hasil 2 pangkat 2: 4

-----
Test Case 2:
Masukkan nilai x dan y (dipisah spasi): 5 3
Hasil 5 pangkat 3: 125

-----

Program Selesai!
PS C:\Users\lenovo\OneDrive\文档\ivena\2311102126_Agnes-Refilina-Fiska\ALPRO_2> 
```

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 ini menggunakan fungsi rekursif untuk menghitung pangkat dari sebuah bilangan. Pengguna dapat memasukkan sejumlah test cases, dan untuk setiap test case, program akan meminta dua angka yang mewakili basis dan pangkat, lalu menghitung hasil pangkat tersebut dan menampilkannya. Teknik rekursi dalam fungsi power memungkinkan pemecahan masalah penghitungan pangkat menjadi lebih sederhana dengan memanggil kembali fungsi itu sendiri.

IV. KESIMPULAN

Rekursi merupakan suatu metode dalam pemrograman yang memungkinkan sebuah subprogram untuk memanggil dirinya sendiri. Metode ini efektif dalam menangani masalah yang kompleks dengan cara menguraikannya menjadi sub-masalah yang serupa dengan masalah aslinya.

Setiap algoritma yang menggunakan pendekatan rekursif terdiri dari dua elemen utama:

1. **Base Case:** Ini adalah kondisi yang mengakhiri proses rekursif, biasanya berupa kondisi dasar yang dapat diselesaikan dengan cara langsung.
2. **Recursive Case:** Ini adalah kondisi di mana subprogram memanggil dirinya kembali, sehingga masalah dapat dipecah menjadi sub-masalah yang lebih kecil untuk diselesaikan.

Metode rekursi dapat menjadi alternatif yang efektif bagi penggunaan struktur perulangan. Base Case memegang peranan penting dalam teknik rekursi, karena fungsinya untuk menghentikan proses dan mencegah terjadinya loop tak terbatas dalam eksekusi program.

V. REFERENSI

[1] Modul 6 Praktikum Algoritma 2