

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 6
REKURSIF**



Disusun Oleh :

M. Faleno Albar Firjatulloh / 2311102297

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

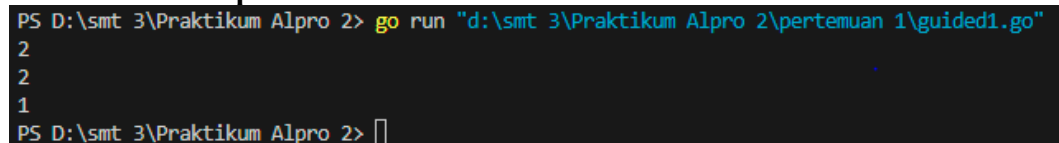
```
package main

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



```
PS D:\smt 3\Praktikum Alpro 2> go run "d:\smt 3\Praktikum Alpro 2\pertemuan 1\guided1.go"
2
2
1
PS D:\smt 3\Praktikum Alpro 2> 
```

Deskripsi Program :

Program ini dibuat menggunakan bahasa pemrograman Go yang menunjukkan penggunaan fungsi rekursif. Tujuan utamanya adalah menampilkan urutan angka secara mundur, dimulai dari angka yang dimasukkan pengguna sampai ke angka 1.

Cara kerjanya:

1. Pertama, program akan meminta pengguna memasukkan sebuah angka positif
2. Setelah angka dimasukkan, program akan menampilkan deret angka yang dimulai dari input pengguna

3. Angka-angka tersebut akan ditampilkan secara berurutan menurun hingga mencapai angka 1

Misalnya, jika pengguna memasukkan angka 5, maka program akan menampilkan: 5, 4, 3, 2, 1. Program ini memanfaatkan konsep rekursi, yaitu fungsi yang memanggil dirinya sendiri, untuk menghasilkan urutan angka menurun tersebut.

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: $n == 1$

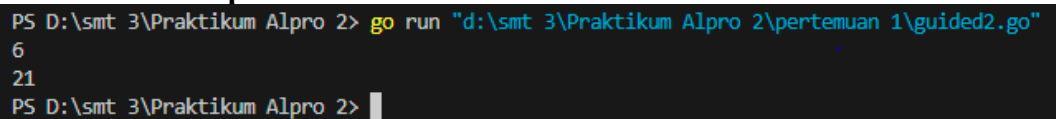
Sourcecode

```
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 D:\smt 3\Praktikum Alpro 2> go run "d:\smt 3\Praktikum Alpro 2\pertemuan 1\guided2.go"
6
21
PS D:\smt 3\Praktikum Alpro 2> 
```

Deskripsi Program :

Program ini dibuat menggunakan bahasa pemrograman Go yang menunjukkan penggunaan fungsi rekursif. Tujuan utamanya adalah menampilkan urutan angka secara mundur, dimulai dari angka yang dimasukkan pengguna sampai ke angka 1.

Cara kerjanya:

1. Pertama, program akan meminta pengguna memasukkan sebuah angka positif
2. Setelah angka dimasukkan, program akan menampilkan deret angka yang dimulai dari input pengguna
3. Angka-angka tersebut akan ditampilkan secara berurutan menurun hingga mencapai angka 1

Misalnya, jika pengguna memasukkan angka 5, maka program akan menampilkan: 5, 4, 3, 2, 1.

Program ini memanfaatkan konsep rekursi, yaitu fungsi yang memanggil dirinya sendiri, untuk menghasilkan urutan angka menurun tersebut.

3. Guided 3

Soal Studi Case

Mencari dua pangkat n atau 2^n

Base – case: $n == 0$

Sourcecode

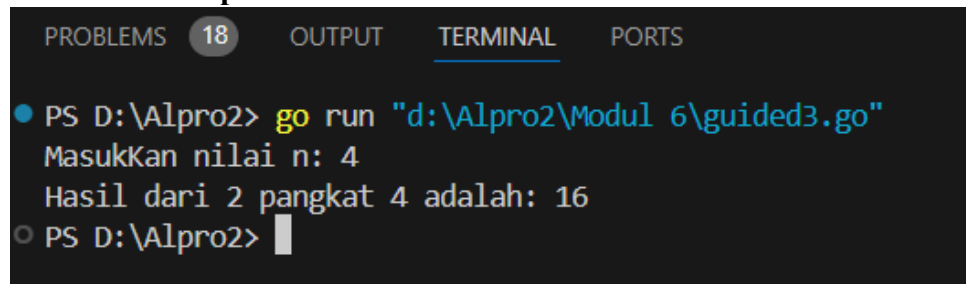
```
package main

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 berfungsi untuk menghitung hasil 2 pangkat n menggunakan metode rekursif.

Cara kerjanya sederhana:

1. User diminta memasukkan nilai n

2. Program menggunakan fungsi rekursif pangkat(n) dengan aturan:
 - o Jika $n=0$, hasilnya 1
 - o Jika $n>0$, hasilnya $2 \times \text{pangkat}(n-1)$
3. Fungsi akan terus memanggil dirinya dengan nilai n yang berkurang satu per satu sampai mencapai 0
4. Hasil akhir dihitung dengan mengalikan angka 2 pada setiap tahap rekursi

Contoh: Jika $n=3$, maka proses rekursinya:

- $\text{pangkat}(3) = 2 \times \text{pangkat}(2)$
- $\text{pangkat}(2) = 2 \times \text{pangkat}(1)$
- $\text{pangkat}(1) = 2 \times \text{pangkat}(0)$
- $\text{pangkat}(0) = 1$

Sehingga hasilnya: $2 \times 2 \times 2 \times 1 = 8$

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

```
PS D:\smt 3\Praktikum Alpro 2> go run "d:\smt 3\Praktikum Alpro 2\pertemuan 1\guided4.go"
6
720
PS D:\smt 3\Praktikum Alpro 2> █
```

Deskripsi Program :

Program ini adalah implementasi perhitungan faktorial menggunakan fungsi rekursif dalam bahasa Go.

Komponen utama program:

1. Fungsi `faktorial(n)`:
 - Jika $n=0$ atau $n=1$, mengembalikan nilai 1
 - Jika $n>1$, menghitung $n \times \text{faktorial}(n-1)$
2. Fungsi `main()`:
 - Menerima input angka dari user
 - Memanggil fungsi faktorial dan menampilkan hasilnya

Contoh cara kerja untuk input $n=4$:

- $\text{faktorial}(4) = 4 \times \text{faktorial}(3)$
- $\text{faktorial}(3) = 3 \times \text{faktorial}(2)$
- $\text{faktorial}(2) = 2 \times \text{faktorial}(1)$
- $\text{faktorial}(1) = 1$

Sehingga: $4 \times 3 \times 2 \times 1 = 24$

Jadi program ini akan menghitung $n!$ (n faktorial) dari angka yang dimasukkan user.

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

```
PS D:\smt 3\Praktikum Alpro 2> go run "d:\smt 3\Praktikum Alpro 2\pertemuan 1\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:\smt 3\Praktikum Alpro 2>
```

Deskripsi Program :

Program ini menghasilkan deret Fibonacci hingga suku ke-10, di mana setiap angka merupakan jumlah dari dua angka sebelumnya.

Metode:

1. Fungsi rekursif `fibonacci(n)` digunakan dengan dua kondisi:
 - Dasar: Jika $(n \leq 1)$, kembalikan nilai (n) .
 - Rekursif: Untuk $(n > 1)$, hitung `fibonacci(n-1) + fibonacci(n-2)`.

Proses:

1. Fungsi `fibonacci(n)` menerima posisi suku yang dicari.
2. Jika $(n > 1)$, fungsi dipanggil dua kali dengan argumen $(n-1)$ dan $(n-2)$, lalu hasilnya dijumlahkan.
3. Fungsi `main()` menggunakan loop untuk menghitung dan menampilkan 10 suku pertama.

Contoh:

- Suku ke-0 = 0
- Suku ke-1 = 1
- Suku ke-2 = 1 (0+1)
- Suku ke-3 = 2 (1+1)
- Dan seterusnya.

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

```

PS D:\smt 3\Praktikum Alpro 2> go run "d:\smt 3\Praktikum Alpro 2\pertemuan 1\unguided2.go"
guided2.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
*
**
***

PS D:\smt 3\Praktikum Alpro 2>

```

Deskripsi Program :

Program ini menerima sejumlah bilangan bulat positif sebagai input dan mencetak pola bintang berbentuk segitiga siku-siku sesuai jumlah baris setiap bilangan input. Pola dicetak menggunakan fungsi rekursif `printStars`.

Algoritma Program:

1. Input: Program meminta jumlah test case dan bilangan positif untuk setiap test case.
2. Pemrosesan: Untuk setiap bilangan input, fungsi `printStars` dipanggil untuk mencetak pola bintang.
3. Fungsi `printStars`:
 - Menerima parameter `n` (jumlah baris) dan `current` (baris saat ini).
 - Jika `current > n`, fungsi berhenti.
 - Jika tidak, fungsi mencetak `current` bintang, lalu memanggil dirinya sendiri dengan `current + 1`.
4. Output: Program menampilkan pola bintang untuk setiap input bilangan.

Cara Kerja:

1. Program meminta jumlah test case dan bilangan bulat untuk setiap test case.
2. Untuk tiap bilangan input, `printStars` dipanggil dengan `n` sebagai jumlah baris dan `current` mulai dari 1.
3. `printStars` mencetak pola bintang secara rekursif hingga kondisi dasar tercapai.
4. Program menampilkan pola bintang sesuai setiap test case.

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 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:\smt 3\Praktikum Alpro 2> go run "d:\smt 3\Praktikum Alpro 2\pertemuan 1\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:\smt 3\Praktikum Alpro 2> |
```

Deskripsi Program

Program ini mencari dan menampilkan faktor-faktor dari bilangan bulat positif secara rekursif.

Algoritma Program:

1. **Input:** Program menerima jumlah test case dan bilangan positif untuk tiap test case.
2. **Fungsi findFactors(n, current):**
 - **Basis Rekursi:** Fungsi berhenti jika current lebih besar dari n.
 - **Periksa Faktor:** Jika n habis dibagi current, maka current adalah faktor dan dicetak.
 - **Rekursi:** Fungsi memanggil dirinya sendiri dengan current + 1, memeriksa semua angka dari 1 hingga n.
3. **Output:** Program menampilkan tabel yang memuat nomor test case, bilangan input, dan daftar faktor.

Cara Kerja:

1. Program meminta jumlah test case dan bilangan untuk setiap case.
2. Untuk setiap test case, program memanggil findFactors dengan current mulai dari 1.
3. findFactors memeriksa dan mencetak faktor hingga mencapai batas n.
4. Hasil ditampilkan dalam bentuk tabel untuk setiap test case.

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)

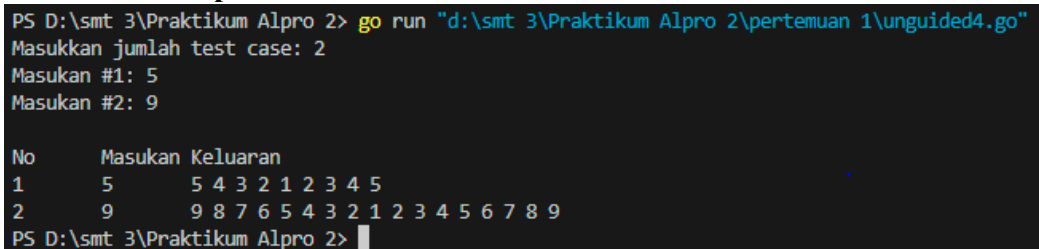
        // Array untuk menyimpan input
        inputs := make([]int, jumlahTest)

        // 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



```

PS D:\smt 3\Praktikum Alpro 2> go run "d:\smt 3\Praktikum Alpro 2\pertemuan 1\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:\smt 3\Praktikum Alpro 2>

```

Deskripsi Program :

Program ini menampilkan pola bilangan yang menurun dari nilai integer hingga 1, lalu naik kembali ke nilai awal.

Algoritma Program:

1. **Input:** Program meminta jumlah test case dan nilai integer untuk setiap test case.
2. **Pemrosesan untuk setiap test case:**

- Panggil fungsi `printPattern()` dengan nilai integer.
 - **`printPattern()`:**
 - Memanggil `printDescending()` untuk mencetak bilangan menurun dari nilai awal hingga 1.
 - Memanggil `printAscending()` untuk mencetak bilangan naik dari 2 hingga nilai awal.
3. **Output:** Program menampilkan pola bilangan untuk tiap test case.

Cara Kerja:

1. Program menerima jumlah test case dan menyimpan setiap nilai integer dalam array.
2. Untuk setiap nilai integer:
 - **Pola Menurun:** `printDescending()` mencetak bilangan secara rekursif dari nilai awal ke 1.
 - **Pola Naik:** `printAscending()` mencetak bilangan naik dari 2 hingga nilai awal.
3. Hasil ditampilkan dalam tabel dengan nomor, masukan, dan pola keluaran.

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:\smt 3\Praktikum Alpro 2> go run "d:\smt 3\Praktikum Alpro 2\pertemuan 1\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:\smt 3\Praktikum Alpro 2> 
```

Deskripsi Program :

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

Algoritma Program:

1. **Input:** Program menerima jumlah test case (jumlahTest) dan array bilangan bulat (inputs).
2. **Iterasi:** Program menjalankan fungsi untuk setiap bilangan dalam inputs.
3. **Rekursi:**
 - o Untuk setiap bilangan, panggil printOddNumbers dengan parameter:
 - n (nilai batas).
 - current, dimulai dari 1.
 - o **Basis Rekursi:** Rekursi berhenti saat $current > n$.
 - o **Cek Ganjil:** Jika current ganjil, cetak current.
 - o **Lanjutkan Rekursi:** Panggil kembali printOddNumbers dengan $current + 1$ untuk memproses bilangan berikutnya.
4. **Output:** Program menampilkan nomor test case, bilangan input, dan daftar bilangan ganjil dari 1 hingga bilangan input.

Cara Kerja:

1. Program menerima input jumlah test case dan array bilangan bulat.
2. Untuk setiap bilangan dalam array:
 - o printOddNumbers memeriksa bilangan dari 1 hingga n secara rekursif.
 - o Jika ganjil, bilangan dicetak.
3. Output berupa daftar bilangan ganjil untuk setiap test case

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

```

PS D:\smt 3\Praktikum Alpro 2> go run "d:\smt 3\Praktikum Alpro 2\pertemuan 1\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:\smt 3\Praktikum Alpro 2>

```

Deskripsi Program :

Program ini menghitung pangkat bilangan menggunakan rekursi. Program meminta jumlah test case, lalu untuk setiap test case, menerima nilai basis dan eksponen, menghitung hasil pangkat dengan fungsi rekursif power(), dan menampilkan hasilnya.

Algoritma Program:

1. **Input:** Program meminta jumlah test case, lalu menerima nilai basis dan eksponen untuk setiap test case.
2. **Fungsi Rekursif power():**
 - **Kondisi Dasar:** Jika eksponen = 0, kembalikan 1.
 - **Rekursi:** Jika eksponen > 0, kembalikan basis * power(basis, eksponen - 1).
3. **Output:** Untuk setiap test case, program menghitung dan menampilkan hasil pangkat.

Cara Kerja:

1. Program menerima jumlah test case dan membaca nilai basis serta eksponen.
2. Untuk setiap test case, power() menghitung pangkat secara rekursif.
3. Hasil perhitungan ditampilkan bersama nomor test case, 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