

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 6
REKURSIF**



Disusun Oleh:

Boutefhika Nuha Ziyadatul Khair/2311102316

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

Rekursi adalah konsep yang mendasari banyak algoritma dalam ilmu komputer. Secara sederhana, rekursi adalah kemampuan sebuah fungsi untuk memecah masalah menjadi sub-masalah yang lebih kecil. Fungsi tersebut kemudian memanggil dirinya sendiri untuk menyelesaikan sub-masalah tersebut. Prinsip ini mencerminkan cara yang seringkali memecahkan masalah dalam kehidupan sehari-hari, dengan membaginya menjadi bagian-bagian yang lebih kecil dan lebih mudah dikelola. Ini adalah ciri khas dari rekursi - memecah masalah menjadi versi yang lebih sederhana dari masalah itu sendiri.

Rekursi memiliki daya ungkit yang kuat dalam mengekspresikan ide-ide dan algoritma yang kompleks. Ia memberikan cara berpikir yang berbeda dalam menghadapi masalah, dengan melihatnya dari sudut pandang yang lebih abstrak. Anda mungkin bertanya-tanya mengapa kita perlu menggunakan rekursi ketika telah memiliki konsep perulangan (looping) untuk mengulangi tugas berulang kali. Jawabannya terletak pada jenis masalah yang dapat diselesaikan dengan rekursi. Rekursi sangat efektif ketika memiliki masalah yang memiliki struktur hierarki atau berulang yang dalam.

Salah satu contoh klasik yang sering disebutkan adalah perhitungan faktorial. Faktorial dari sebuah angka adalah hasil dari perkalian semua bilangan bulat positif mulai dari 1 hingga angka tersebut. Faktorial sering kali dinyatakan sebagai $n!$ dengan n adalah angka yang ingin dihitung faktorialnya. Misalnya, $5!$ (baca: lima faktorial) adalah $5 \times 4 \times 3 \times 2 \times 1 = 120$. Dalam hal ini, Anda dapat merumuskan perhitungan faktorial sebagai masalah rekursif. Faktorial dari n adalah n dikali faktorial dari $(n-1)$. Namun, juga harus memikirkan basis kasus, yaitu faktorial dari 0 adalah 1. Dalam pemrograman ini akan meningkatkan efisiensi, dan menghasilkan solusi yang lebih efektif. Berikut contoh program menghitung faktorial dari angka yang diberikan menggunakan fungsi rekursif:

```
package main

import "fmt"

func factorial(i int) int {
    if(i <= 1) {
        return 1
    }
    return i * factorial(i - 1)
}
```

```
func main() {  
    var i int = 15  
    fmt.Printf("Factorial of %d is %d", i, factorial(i))  
}
```

II. GUIDED

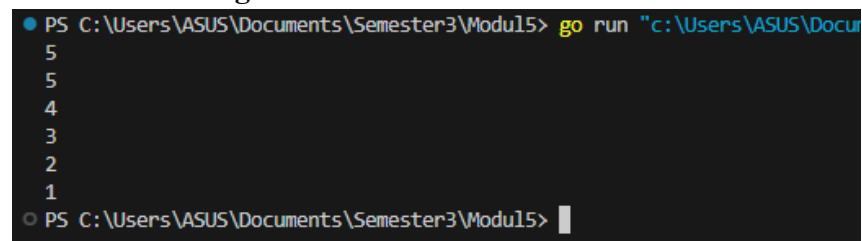
1. Membuat baris bilangan dari n hingga 1

Base-case: `bilangan == 1`

Sourcecode

```
package main  
  
import "fmt"  
  
func baris(bilangan int) {  
    if bilangan == 1 {  
        fmt.Println(1)  
    } else {  
        fmt.Println(bilangan)  
        baris(bilangan - 1)  
    }  
}  
  
func main() {  
    var n int  
    fmt.Scan(&n)  
    baris(n)  
}
```

Screenshoot Program



Deskripsi Program

Program diatas menggunakan fungsi rekursi untuk mencetak bilangan dari n hingga 1. Fungsi ``baris`` menerima satu parameter, ``bilangan``. Jika ``bilangan`` sama dengan 1, fungsi akan mencetak angka 1. Namun, jika ``bilangan`` lebih besar dari 1 maka fungsi akan mencetak nilai ``bilangan`` saat ini, lalu memanggil dirinya sendiri dengan parameter ``bilangan - 1``, sehingga mencetak angka secara berurutan dari n ke 1. Dalam fungsi ``main``, program mendeklarasikan variabel ``n`` dengan tipe data integer, lalu meminta input ``n`` dari pengguna dan memanggil ``baris(n)`` untuk memulai pencetakan.

2. Menghitung hasil penjumlahan 1 hingga n

Base-case: $n == 1$

Sourcecode

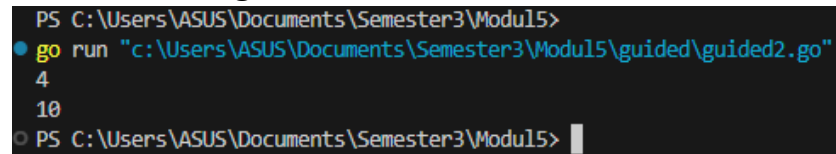
```
package main

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 Program



```
PS C:\Users\ASUS\Documents\Semester3\Modul5>
go run "c:\Users\ASUS\Documents\Semester3\Modul5\guided\guided2.go"
4
10
PS C:\Users\ASUS\Documents\Semester3\Modul5>
```

Deskripsi Program

Program diatas menggunakan fungsi rekursi untuk menghitung jumlah bilangan dari 1 hingga `n`. Fungsi **`penjumlahan`** menerima parameter `n` dan mengembalikan hasil penjumlahan itu. Jika `n` sama dengan 1, fungsi akan mengembalikan 1 sebagai kondisi dasar untuk menghentikan rekursi. Jika `n` lebih dari 1, fungsi memanggil dirinya sendiri dengan **`n-1`** dan menambahkan hasilnya dengan `n`, sehingga terus menurunkan nilai `n` sampai 1. Di dalam fungsi **`main`**, program mendeklarasikan variabel `n` dengan tipe data integer, lalu meminta pengguna memasukkan nilai `n`, dan memanggil **`penjumlahan(n)`** untuk menghitung dan mencetak hasilnya.

3. Mencari dua pangkat n atau 2^x

Base-case: $n == 0$

Sourcecode

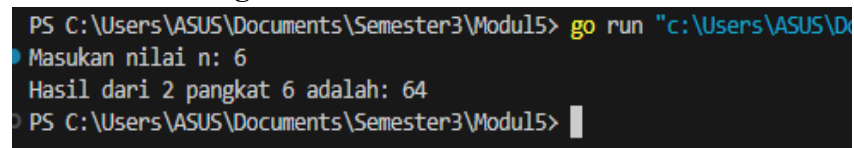
```
package main
```

```
import "fmt"

//fungsi rekursif untuk menghitung 2^n
func pangkat(n int) int {
    if n == 0 {
        return 1
    } else {
        return 2 * pangkat(n-1) // 2 * (3-1)
    }
}

func main() {
    var n int
    fmt.Print("Masukan nilai n: ")
    fmt.Scan(&n)
    fmt.Println("Hasil dari 2 pangkat", n, "adalah:",
    pangkat(n))
}
```

Screenshoot Program



```
PS C:\Users\ASUS\Documents\Semester3\Modul5> go run "c:\Users\ASUS\Documents\Semester3\Modul5\pangkat.go"
Masukan nilai n: 6
Hasil dari 2 pangkat 6 adalah: 64
PS C:\Users\ASUS\Documents\Semester3\Modul5>
```

Deskripsi Program

Program diatas menggunakan fungsi rekursi untuk menghitung nilai 2^n (2 pangkat n). Fungsi **pangkat** menerima parameter **n** dan mengembalikan hasil perpangkatan 2 sampai **n**. Jika **n** bernilai 0, fungsi akan mengembalikan 1 sebagai kondisi dasar, karena $2^0 = 1$. Jika **n** lebih besar dari 0 maka fungsi akan memanggil dirinya sendiri dengan **n-1**, lalu mengalikan hasilnya dengan 2. Di dalam fungsi **main**, program mendeklarasikan variabel **n** dengan tipe data integer, lalu meminta input **n** dari pengguna, dan mencetak hasil dari **2 pangkat n** dengan memanggil **pangkat(n)**.

4. Mencari nilai faktorial atau n!

Base-case: $n == 0$ atau $n == 1$

Sourcecode

```
package main

import "fmt"

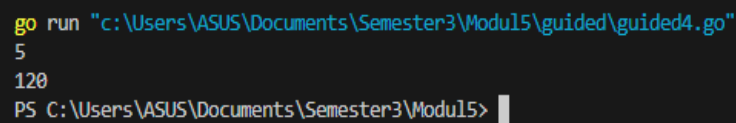
var n int

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))
}
```

Screenshoot Program



```
go run "c:\Users\ASUS\Documents\Semester3\Modul5\guided\guided4.go"
5
120
PS C:\Users\ASUS\Documents\Semester3\Modul5>
```

Deskripsi Program

Program diatas menggunakan fungsi rekursi untuk menghitung faktorial dari suatu bilangan **`n`**. Dimulai dengan program mendeklarasikan variabel **`n`** secara global dengan tipe data integer. Fungsi **`faktorial`** menerima parameter **`n`** dan mengembalikan hasil faktorial dari **`n`**. Jika **`n`** bernilai 0 atau 1, fungsi akan mengembalikan 1 sebagai kondisi dasar, karena faktorial dari 0 dan 1 adalah 1. Jika **`n`** lebih besar dari 1, fungsi akan memanggil dirinya sendiri dengan **`n-1`** dan mengalikan hasilnya dengan **`n`**. Dalam fungsi **`main`** program meminta pengguna memasukkan nilai **`n`**, dan memanggil **`faktorial(n)`** untuk menghitung dan mencetak hasil faktorialnya.

III. UNGUIDED

1. Deret fibonacci adalah 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 yang mengimplementasikan fungsi rekursif pada deret fibonacci tersebut.

Sourcecode

```
package main

import "fmt"

// Fungsi rekursif untuk menghitung nilai Fibonacci
pada suku ke-n
func fibonacci(n int) int {
    if n == 0 {
        return 0 // Suku ke-0
    } else if n == 1 {
        return 1 // Suku ke-1
    } else {
        return fibonacci(n-1) + fibonacci(n-2) //
Penjumlahan dua suku sebelumnya
    }
}

func main() {
    var n int
    fmt.Scan(&n)
    fmt.Println(fibonacci(n))
}
```

Screenshoot Output

```
PS C:\Users\ASUS\Documents\Semester3\Modul5>
go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided1.go"
0
0
PS C:\Users\ASUS\Documents\Semester3\Modul5> go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided1.go"
1
1
PS C:\Users\ASUS\Documents\Semester3\Modul5> go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided1.go"
2
1
PS C:\Users\ASUS\Documents\Semester3\Modul5> go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided1.go"
3
2
PS C:\Users\ASUS\Documents\Semester3\Modul5> go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided1.go"
4
3
PS C:\Users\ASUS\Documents\Semester3\Modul5> go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided1.go"
5
5
PS C:\Users\ASUS\Documents\Semester3\Modul5> go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided1.go"
6
8
PS C:\Users\ASUS\Documents\Semester3\Modul5> go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided1.go"
7
13
PS C:\Users\ASUS\Documents\Semester3\Modul5> go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided1.go"
8
21
PS C:\Users\ASUS\Documents\Semester3\Modul5> go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided1.go"
9
34
PS C:\Users\ASUS\Documents\Semester3\Modul5> go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided1.go"
10
55
PS C:\Users\ASUS\Documents\Semester3\Modul5>
```

Deskripsi Program

Program diatas menggunakan rekursi untuk menghitung nilai bilangan Fibonacci pada suku ke- n . Fungsi `fibonacci` menerima parameter `n` dan mengembalikan nilai suku Fibonacci ke- n . Jika n bernilai 0, fungsi mengembalikan 0 (karena suku ke-0 dalam deret Fibonacci adalah 0), dan jika n bernilai 1, fungsi mengembalikan 1 (suku ke-1 dalam deret Fibonacci adalah 1). Untuk nilai n yang lebih besar dari 1, fungsi akan memanggil dirinya sendiri dengan argumen $n-1$ dan $n-2$, lalu menjumlahkan kedua hasil tersebut untuk mendapatkan nilai suku ke- n . Dalam fungsi `main`, program mendeklarasikan variabel `n` dengan tipe data integer, lalu meminta input `n` dari pengguna, dan mencetak hasil perhitungan `fibonacci(n)`.

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

Sourcecode

```
package main

import "fmt"
```



```

var n int

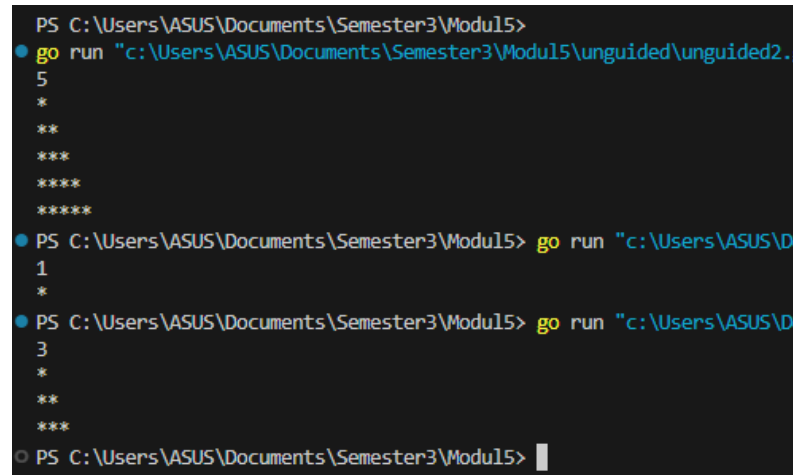
// Fungsi rekursif untuk mencetak bintang pada setiap
baris
func bintang(jumlah int) {
    if jumlah > 0 {
        bintang(jumlah - 1) // Panggil fungsi rekursif
        untuk jumlah - 1
        fmt.Print("*") // Cetak bintang
    } else {
        return // Kembali jika jumlah sudah mencapai 0
    }
}

// Fungsi rekursif untuk mencetak pola bintang hingga
baris ke-N
func pola(baris int) {
    if baris > n {
        return // Kembali jika sudah lebih dari N
    } else {
        bintang(baris) // Panggil fungsi untuk mencetak
        bintang
        fmt.Println()
        pola(baris + 1) // Rekursi untuk baris berikutnya
    }
}

func main() {
    fmt.Scan(&n)
    pola(1) // Memulai dari baris pertama
}

```

Screenshoot Program



```

PS C:\Users\ASUS\Documents\Semester3\Modul5>
● go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided2.
5
*
**
***
****
*****
● PS C:\Users\ASUS\Documents\Semester3\Modul5> go run "c:\Users\ASUS\D
1
*
● PS C:\Users\ASUS\Documents\Semester3\Modul5> go run "c:\Users\ASUS\D
3
*
**
***
○ PS C:\Users\ASUS\Documents\Semester3\Modul5>

```

Deskripsi Program

Program diatas menggunakan fungsi rekursi untuk mencetak pola segitiga bintang bertingkat hingga baris ke-n. Program mendeklarasikan variabel `n` secara global dengan tipe data integer. Fungsi `bintang` mencetak sejumlah bintang sesuai dengan parameter `jumlah` yang diberikan. Jika `jumlah` lebih dari 0, fungsi memanggil dirinya sendiri dengan `jumlah - 1` lalu mencetak satu bintang, sehingga bintang muncul berurutan dalam satu baris. Fungsi `pola` mencetak setiap baris pola hingga mencapai baris ke-n. Jika `baris` sudah lebih dari `n` fungsi akan berhenti, tetapi jika tidak, fungsi memanggil `bintang(baris)` untuk mencetak sejumlah bintang sesuai nomor baris, lalu melanjutkan ke baris berikutnya dengan `pola(baris + 1)`. Di dalam `main`, program meminta input `n` dari pengguna dan memulai pola dari baris pertama.

3. Buatlah program yang mengimplementasikan rekursif untuk menampilkan faktor 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 faktor dari N (terurut dari 1 hingga N ya).

Sourcecode

```
package main

import "fmt"

// Fungsi rekursif untuk menampilkan faktor bilangan
dari N
func faktor(bilangan int, n int) {
    if bilangan > n {
        return
    } else if n%bilangan == 0 {
        fmt.Print(bilangan, " ")
    }
    faktor(bilangan + 1, n) // Rekursi untuk bilangan
    berikutnya
}

func main() {
    var n int
    fmt.Scan(&n)
    faktor(1, n)
}
```

Screenshoot Program

```
PS C:\Users\ASUS\Documents\Semester3\Modul5>
● go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided3.go"
5
1 5
● PS C:\Users\ASUS\Documents\Semester3\Modul5> go run "c:\Users\ASUS\Docum
12
1 2 3 4 6 12
○ PS C:\Users\ASUS\Documents\Semester3\Modul5> |
```

Deskripsi Program

Program diatas menggunakan fungsi rekursi untuk menampilkan semua faktor dari bilangan n. Fungsi ``faktor`` menerima dua parameter yaitu ``bilangan`` yang dimulai dari 1 dan bertambah setiap kali fungsi dipanggil, dan ``n`` yang merupakan bilangan yang faktornya ingin ditentukan. Jika ``bilangan`` melebihi ``n``, fungsi berhenti. Jika ``bilangan`` merupakan faktor dari ``n`` yaitu ``n % bilangan == 0``, maka ``bilangan`` akan dicetak di layar. Setelah itu, fungsi ``faktor`` dipanggil kembali dengan nilai ``bilangan + 1`` untuk memeriksa faktor berikutnya. Di dalam ``main``, program mendeklarasikan variabel ``n`` dengan tipe data integer, lalu membaca input ``n`` dari pengguna dan memulai pencarian faktor dari ``1``.

4. 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.

Sourcecode

```
package main

import "fmt"

func barisanBilangan(n int) {
    if n > 0 { // Menampilkan n hingga 1
        fmt.Print(n, " ")
        barisanBilangan(n - 1)
    } else {
        return // Kembali jika x tidak lebih dari 0
    }
    // Menampilkan kembali dari 2 hingga n
    fmt.Print(n, " ")
}

func main() {
    var n int
```

```

    fmt.Scan(&n)
    barisanBilangan(n)
}

```

Screenshoot Program

```

PS C:\Users\ASUS\Documents\Semester3\Modul5>
● go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided4.go"
5
5 4 3 2 1 1 2 3 4 5
● PS C:\Users\ASUS\Documents\Semester3\Modul5> go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided4.go"
9
9 8 7 6 5 4 3 2 1 1 2 3 4 5 6 7 8 9
○ PS C:\Users\ASUS\Documents\Semester3\Modul5>

```

Deskripsi Program

Program diatas menggunakan fungsi rekursi untuk menampilkan barisan bilangan dari n hingga 1, kemudian kembali dari 1 hingga n. Fungsi **`barisanBilangan`** menerima parameter **`n`**. Jika **`n`** lebih dari 0, fungsi akan mencetak **`n`** terlebih dahulu, kemudian memanggil dirinya sendiri dengan **`n-1`**, sehingga menghasilkan urutan turun dari **`n`** hingga 1. Setelah mencapai 0, rekursi berhenti dan setiap kali fungsi kembali ke pemanggilan sebelumnya, ia mencetak nilai **`n`** lagi, yang menghasilkan urutan naik dari 1 hingga n. Di dalam fungsi **`main`**, program mendeklarasikan variabel **`n`** dengan tipe data integer, lalu meminta input **`n`** dari pengguna dan memulai pencetakan dengan memanggil **`barisanBilangan(n)`**.

5. 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.

Sourcecode

```

package main

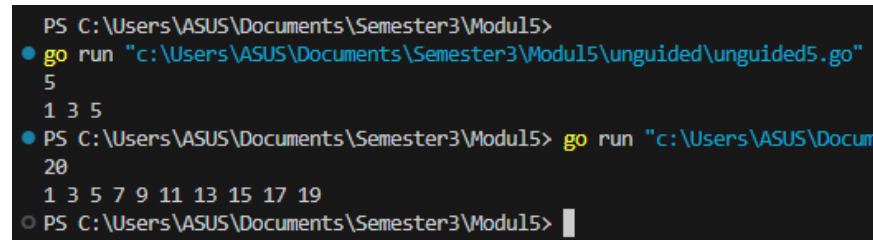
import "fmt"

// Fungsi rekursif untuk menampilkan bilangan ganjil
// dari 1 hingga n
func bilanganGanjil(n int) {
    if n <= 0 { // jika n tidak positif maka berhenti
        return
    }
    bilanganGanjil(n - 1) // // Rekursif dengan x - 1
    if n%2 != 0 {         // Cek apakah n ganjil
        fmt.Print(n, " ") // Cetak n jika ganjil
    }
}

```

```
func main() {
    var n int
    fmt.Scan(&n)
    bilanganGanjil(n) // Panggil fungsi untuk
menampilkan bilangan ganjil
}
```

Screenshoot Program



```
PS C:\Users\ASUS\Documents\Semester3\Modul5>
go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided5.go"
5
1 3 5
PS C:\Users\ASUS\Documents\Semester3\Modul5> go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided5.go" 20
1 3 5 7 9 11 13 15 17 19
PS C:\Users\ASUS\Documents\Semester3\Modul5> |
```

Deskripsi Program

Program diatas menggunakan fungsi rekursi untuk menampilkan bilangan ganjil dari 1 hingga n. Fungsi **'bilanganGanjil'** menerima parameter **'n'** dan memanggil dirinya sendiri dengan **'n-1'** hingga **'n'** mencapai 0, yang menjadi kondisi penghentian rekursi. Setelah mencapai 0, fungsi kembali ke pemanggilan sebelumnya dan memeriksa setiap nilai **'n'**, jika **'n'** adalah bilangan ganjil maka akan diperiksa dengan **'n % 2 != 0'**, dan nilai **'n'** dicetak. Dalam **'main'**, program mendeklarasikan variabel **'n'** dengan tipe data integer, lalu meminta input **'n'** dari pengguna dan memanggil **'bilanganGanjil(n)'** untuk mencetak semua bilangan ganjil dari 1 hingga n.

6. 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"**.

Sourcecode

```
package main

import "fmt"

func pangkat(a int, b int) int {
    if b == 0 { //jika b sama dengan 0 maka hasilnya 1
        return 1
    } else {
```

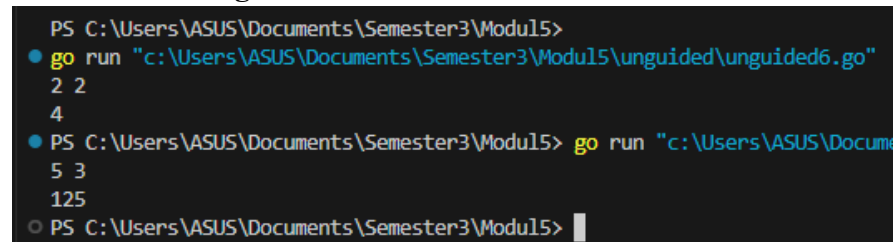
```

        return a * pangkat(a, b-1) // Rekursi
    a*pangkat(a, b-1)
    }
}

func main() {
    var x, y int
    fmt.Scan(&x, &y)
    fmt.Println(pangkat(x, y))
}

```

Screenshoot Program



```

PS C:\Users\ASUS\Documents\Semester3\Modul5>
● go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided6.go"
2 2
4
● PS C:\Users\ASUS\Documents\Semester3\Modul5> go run "c:\Users\ASUS\Documents\Semester3\Modul5\unguided\unguided6.go"
5 3
125
○ PS C:\Users\ASUS\Documents\Semester3\Modul5>

```

Deskripsi Program

Program diatas menggunakan fungsi rekursi untuk menghitung nilai pangkat a^b (a pangkat b). Fungsi **`pangkat`** menerima dua parameter yaitu **`a`** sebagai basis dan **`b`** sebagai eksponen. Jika **`b`** sama dengan 0, fungsi akan mengembalikan 1, sesuai dengan aturan matematika bahwa setiap bilangan pangkat 0 adalah 1. Jika **`b`** lebih dari 0, fungsi memanggil dirinya sendiri dengan parameter **`b-1`**, mengalikan hasilnya dengan **`a`**. Dalam fungsi **`main`**, program mendeklarasikan variabel **`x`** dan **`y`** dengan tipe data integer, lalu membaca dua input **`x`** dan **`y`** dari pengguna, yang mewakili nilai basis dan eksponen, kemudian mencetak hasil dari **`pangkat(x, y)`**.

IV. KESIMPULAN

Rekursi adalah konsep dasar dalam ilmu komputer yang memungkinkan suatu fungsi memecah masalah menjadi sub-masalah yang lebih kecil dengan memanggil dirinya sendiri. Prinsip ini umum digunakan dalam kehidupan sehari-hari untuk menyederhanakan masalah kompleks menjadi bagian yang lebih kecil dan lebih mudah dikelola. Walaupun looping dapat mengulangi tugas, rekursi efektif dalam menangani masalah yang memiliki struktur hierarkis atau pola berulang yang dalam. Sebagai contoh perhitungan faktorial, di mana faktorial n dapat direpresentasikan sebagai n dikali faktorial $(n-1)$ dengan kondisi dasar faktorial 0 sama dengan 1. Program rekursif pada Go dapat mengimplementasikan berbagai

algoritma, termasuk mencetak bilangan dari n ke 1, menghitung jumlah bilangan dari 1 hingga n , menghitung dua pangkat n , menghitung faktorial, serta menampilkan deret bilangan Fibonacci. Selain itu, rekursi juga dapat digunakan untuk pola seperti mencetak bintang bertingkat, menentukan faktor bilangan dari suatu bilangan, menampilkan barisan bilangan dari n hingga 1 dan kembali ke n , mencetak bilangan ganjil dari 1 hingga n , serta menghitung pangkat suatu bilangan tanpa fungsi matematika eksternal. Konsep rekursi memberikan kekuatan ekspresif dalam menyelesaikan masalah kompleks dengan pendekatan yang lebih abstrak, dan menekankan pentingnya memahami kondisi dasar agar rekursi dapat berakhir dengan tepat.

V. REFERENSI

- [1] Purbasari, W., Iqbal, T., Inayah, I., Munawir, M., Sutjiningtyas, S., Hikmawati, E., ... & Basri, H. (2024). ALGORITMA PEMROGRAMAN.
- [2] **Kursus Web Programming.** (n.d.). *Recursion dalam bahasa Golang*. Diakses pada 1 November 2024, dari <https://kursuswebprogramming.com/recursion-dalam-bahasa-golang/>