

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL V

REKURSIF



Disusun Oleh :

Ariiq Radhitya Pradana / 2311102260

IF 11 06

Dosen Pengampu :

ABEDNEGO DWI SEPTIADI

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Rekursif dalam pemrograman, khususnya dalam bahasa Go (Golang), adalah teknik di mana sebuah fungsi memanggil dirinya sendiri untuk menyelesaikan suatu masalah. Konsep ini sangat berguna ketika masalah dapat dipecah menjadi sub-masalah yang lebih kecil dan lebih sederhana. Dalam implementasinya, fungsi rekursif harus memiliki kondisi dasar (base case) yang jelas untuk menghentikan pemanggilan berulang; jika tidak, pemanggilan yang tak terbatas dapat menyebabkan kesalahan `stack overflow`.

Kelebihan dari penggunaan rekursi adalah kode yang dihasilkan sering kali lebih sederhana dan lebih mudah dibaca dibandingkan dengan solusi iteratif, serta memberikan pendekatan elegan untuk masalah yang melibatkan struktur data seperti pohon dan grafik. Namun, ada juga beberapa kekurangan, seperti kinerja yang cenderung lebih lambat dan penggunaan memori yang lebih tinggi karena overhead pemanggilan fungsi. Selain itu, debugging dalam kode rekursif bisa menjadi tantangan tersendiri karena alur eksekusi yang kompleks.

Sebagai contoh, untuk menghitung faktorial dari sebuah angka menggunakan rekursi dalam Go, kita dapat mendefinisikan fungsi yang memanggil dirinya sendiri hingga mencapai kondisi dasar. Begitu juga dengan deret Fibonacci, di mana setiap angka merupakan penjumlahan dari dua angka sebelumnya. Meskipun rekursi memiliki kelebihan dalam hal kesederhanaan dan kejelasan, penting bagi programmer untuk memahami batasan dan potensi masalah yang mungkin muncul saat menggunakan teknik ini dalam pengembangan aplikasi.

II. GUIDED

Guided 1

```
package main

import "fmt"

// Fungsi untuk mencetak bilangan dari n hingga 1
func cetakMundur(n int) {
    if n == 1 {
        fmt.Println(n)
        return
    }
    fmt.Print(n, " ")
    cetakMundur(n - 1)
}

func main() {
    var n int
    fmt.Print(" Masukkan nilai n untuk cetak bilangan
dari n hingga 1: ")
    fmt.Scanln(&n)
    fmt.Print(" hasil cetak mundur: ")
    cetakMundur(n)
}
```

Screenshoot Output

```
go run /tmp/ba01a1D1qC.go
Masukkan nilai n untuk cetak bilangan dari n hingga 1: 2
hasil cetak mundur: 2 1
```

Deskripsi Program

Kode ini ditulis dalam bahasa Go (Golang) untuk mencetak bilangan secara mundur dari nilai yang dimasukkan oleh pengguna hingga 1. Pada fungsi ``main``, pengguna diminta untuk memasukkan sebuah bilangan bulat ``n``, yang kemudian disimpan dalam variabel ``n``. Setelah menerima input, program akan memanggil fungsi ``cetakMundur(n)`` yang bertugas mencetak bilangan dari ``n`` hingga 1.

Fungsi ``cetakMundur`` menggunakan rekursi untuk mencapai tujuan tersebut. Di dalam fungsi ini, terdapat kondisi dasar ``if n == 1``, yang berfungsi untuk menghentikan pemanggilan rekursif saat mencapai nilai 1, dengan mencetak ``1`` di baris baru. Jika ``n`` lebih besar dari 1, fungsi akan mencetak ``n`` diikuti dengan spasi, kemudian memanggil dirinya sendiri dengan nilai ``n - 1``, sehingga proses ini akan terus berulang hingga mencapai kondisi dasar ``n == 1``.

III. GUIDED

Guided 2

```
package main

import "fmt"

// fungsi untuk menghitung penjumlahan 1 hingga n
func jumlahRekursif(n int) int {
    if n == 1 {
        return 1
    }
    return n + jumlahRekursif(n-1)
}

func main() {
    var n int
    fmt.Print(" Masukkan nilai n untuk penjumlahan 1
hingga n: ")
    fmt.Scanln(&n)
    fmt.Println(" hasil penjumlahan: ",
jumlahRekursif(n))
}
```

Screenshoot Output

```
go run /tmp/lAebLKbSd1.go
Masukkan nilai n untuk penjumlahan 1 hingga n: 4
hasil penjumlahan: 10
```

Deskripsi Program

Kode ini ditulis dalam bahasa Go (Golang) dan berfungsi untuk menghitung penjumlahan dari angka 1 hingga `n` menggunakan rekursi. Dalam fungsi `main`, program meminta pengguna memasukkan sebuah bilangan bulat `n`, yang disimpan dalam variabel `n`. Setelah nilai `n` dimasukkan, program akan memanggil fungsi `jumlahRekursif(n)`, yang menghitung total penjumlahan dari 1 hingga `n`.

Fungsi `jumlahRekursif` adalah fungsi rekursif, yang artinya fungsi ini memanggil dirinya sendiri untuk menyelesaikan perhitungan. Di dalam fungsi ini, terdapat kondisi dasar `if n == 1` yang akan menghentikan rekursi ketika `n` mencapai nilai 1, dan mengembalikan nilai 1 sebagai hasil akhir. Jika `n` lebih besar dari 1, fungsi akan mengembalikan nilai `n` ditambah hasil dari `jumlahRekursif(n-1)`. Proses ini terus berulang dengan mengurangi nilai `n` satu per satu hingga mencapai 1, sehingga fungsi akan menjumlahkan seluruh angka dari 1 hingga `n`. Hasil akhir penjumlahan ini kemudian dicetak di layar oleh perintah `fmt.Println` dalam fungsi `main`.

IV. GUIDED

Guided 3

```
package main

import "fmt"

// Fungsi untuk mencari 2 pangkat n
func pangkatDua(n int) int {
    if n == 0 {
        return 1
    }
    return 2 * pangkatDua(n-1)
}

func main() {
    var n int
    fmt.Print(" Masukkan nilai n untuk mencari 2 pangkat
n: ")
    fmt.Scanln(&n)
    fmt.Println("Hasil 2 pangkat", n, ":",
pangkatDua(n))
}
```

Screenshoot Output

```
go run /tmp/Ztg5jQ8JQG.go
Masukkan nilai n untuk mencari 2 pangkat n: 3
Hasil 2 pangkat 3 : 8
|
```

Deskripsi Program

Kode ini, yang ditulis dalam bahasa Go (Golang), berfungsi untuk menghitung nilai dari 2 pangkat `n` menggunakan rekursi. Di dalam fungsi `main`, program meminta pengguna untuk memasukkan sebuah bilangan bulat `n`, yang kemudian digunakan untuk menghitung hasil dari 2 pangkat `n`. Setelah menerima input `n`, program memanggil fungsi `pangkatDua(n)` yang melakukan perhitungan pangkat.

Fungsi `pangkatDua` adalah fungsi rekursif yang menghitung 2 pangkat `n` dengan memanggil dirinya sendiri. Fungsi ini memiliki kondisi dasar `if n == 0` yang mengembalikan nilai 1, karena 2 pangkat 0 sama dengan 1. Jika nilai `n` lebih besar dari 0, fungsi akan mengalikan 2 dengan hasil `pangkatDua(n-1)`, sehingga nilai pangkat berkurang satu per satu dalam setiap panggilan rekursif hingga mencapai kondisi dasar. Proses ini akan menghasilkan nilai 2 pangkat `n`, yang kemudian dicetak ke layar oleh perintah `fmt.Println` di dalam fungsi `main`.

V. GUIDED

Guided 4

```
package main

import "fmt"

// fungsi untuk menghitung faktorial n!
func faktorial(n int) int {
    if n == 0 || n == 1 {
        return 1
    }
    return n * faktorial(n-1)
}

func main() {
    var n int
    fmt.Print(" Masukkan nilai n untuk mencari faktorial
n!: ")
    fmt.Scanln(&n)
    fmt.Println(" Hasil faktorial dari", n, ":",
faktorial(n))
}
```

Screenshoot Output

```
go run /tmp/ZF7XkGb0uT.go
Masukkan nilai n untuk mencari faktorial n!: 4
Hasil faktorial dari 4 : 24
|
```

Deskripsi Program

Kode ini ditulis dalam bahasa Go (Golang) dan berfungsi untuk menghitung faktorial dari sebuah bilangan bulat `n` menggunakan rekursi. Di dalam fungsi `main`, program meminta pengguna memasukkan nilai `n`, yang kemudian digunakan sebagai input untuk menghitung faktorial `n!`. Setelah menerima nilai `n`, program memanggil fungsi `faktorial(n)` untuk menghitung hasil faktorialnya.

Fungsi `faktorial` adalah fungsi rekursif yang menghitung nilai faktorial `n!`, yaitu hasil perkalian dari semua bilangan bulat positif dari `1` hingga `n`. Fungsi ini memiliki kondisi dasar `if n == 0 || n == 1`, yang mengembalikan nilai 1 karena faktorial dari 0 dan 1 sama dengan 1. Jika `n` lebih besar dari 1, fungsi akan mengembalikan nilai `n` dikalikan dengan `faktorial(n-1)`, sehingga nilai `n` akan terus dikalikan dengan nilai faktorial dari bilangan di bawahnya sampai mencapai kondisi dasar. Hasil akhir perhitungan faktorial ini kemudian dicetak ke layar oleh perintah `fmt.Println` di dalam fungsi `main`.

VI. UNGUIDED

Unguided 1

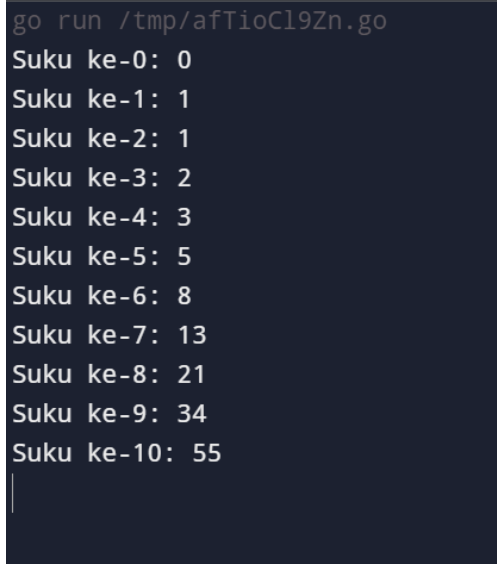
```
package main

import (
    "fmt"
)

// Fungsi rekursif untuk menghitung deret Fibonacci
func fibonacci(n int) int {
    if n <= 1 {
        return n
    }
    return fibonacci(n-1) + fibonacci(n-2)
}

func main() {
    // Menampilkan deret Fibonacci hingga suku ke-10
    for i := 0; i <= 10; i++ {
        fmt.Printf("Suku ke-%d: %d\n", i, fibonacci(i))
    }
}
```

Screenshot Output



```
go run /tmp/afTioCl9Zn.go
Suku ke-0: 0
Suku ke-1: 1
Suku ke-2: 1
Suku ke-3: 2
Suku ke-4: 3
Suku ke-5: 5
Suku ke-6: 8
Suku ke-7: 13
Suku ke-8: 21
Suku ke-9: 34
Suku ke-10: 55
```

Deskripsi Program

Kode ini ditulis dalam bahasa Go (Golang) dan berfungsi untuk menghitung serta menampilkan deret Fibonacci hingga suku ke-10 menggunakan fungsi rekursif. Di dalam fungsi ``main``, program menggunakan perulangan ``for`` untuk menghasilkan nilai deret Fibonacci dari suku pertama (ke-0) hingga suku ke-10. Pada setiap iterasi, fungsi ``fibonacci(i)`` dipanggil untuk menghitung nilai suku ke-``i`` dalam deret, kemudian hasilnya dicetak dengan menggunakan ``fmt.Printf``.

Fungsi ``fibonacci`` adalah fungsi rekursif yang menghitung nilai deret Fibonacci pada suku ke-``n``. Dalam fungsi ini, terdapat kondisi dasar ``if n <= 1`` yang mengembalikan nilai ``n`` itu sendiri, karena pada deret Fibonacci, suku pertama dan kedua masing-masing bernilai 0 dan 1. Jika ``n`` lebih besar dari 1, fungsi akan menghitung nilai suku ke-``n`` sebagai penjumlahan dari dua suku sebelumnya, yaitu ``fibonacci(n-1) + fibonacci(n-2)``. Dengan cara ini, fungsi akan terus memanggil dirinya sendiri hingga mencapai kondisi dasar, menghasilkan nilai Fibonacci yang sesuai. Hasil dari setiap suku dari suku ke-0 hingga ke-10 ditampilkan di layar sesuai format ``Suku ke-n: hasil``.

VII. UNGUIDED

Unguided 2

```
package main

import (
    "fmt"
)

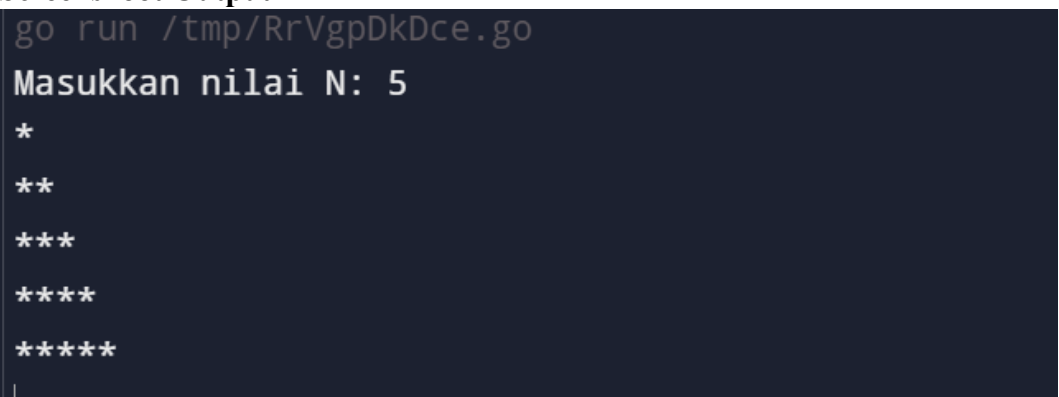
func printStars(n, i int) {
    if i <= n {
        // Cetak bintang sebanyak nilai i
        for j := 0; j < i; j++ {
            fmt.Print("*")
        }
        fmt.Println()

        // Panggil rekursif untuk baris berikutnya
        printStars(n, i+1)
    }
}

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

    printStars(n, 1)
}
```

Screenshoot Output



```
go run /tmp/RrVgpDkDce.go
Masukkan nilai N: 5
*
**
***
****
*****
```

Deskripsi Program

Kode ini ditulis dalam bahasa Go (Golang) dan berfungsi untuk mencetak pola segitiga terbalik dengan bintang menggunakan fungsi rekursif. Dalam fungsi `main`, program meminta pengguna untuk memasukkan sebuah bilangan bulat `n`, yang akan menentukan jumlah bintang pada baris pertama dan jumlah baris dalam pola tersebut. Setelah pengguna memasukkan nilai `n`, program memanggil fungsi `printStars(n)` untuk memulai proses pencetakan pola.

Fungsi `printStars` adalah fungsi rekursif yang mencetak pola bintang secara berkurang di setiap barisnya. Fungsi ini akan berhenti ketika `n` bernilai 0 atau kurang. Pada setiap pemanggilan, jika `n` lebih besar dari 0, perulangan `for` mencetak `n` bintang pada baris tersebut. Setelah mencetak bintang, fungsi ini memanggil dirinya sendiri dengan nilai `n-1`, sehingga jumlah bintang yang dicetak akan berkurang satu pada setiap baris berikutnya. Proses ini terus berlangsung hingga jumlah bintang pada baris mencapai 1. Hasilnya adalah pola segitiga terbalik dengan jumlah bintang pada baris pertama sebanyak `n`, yang berkurang satu bintang di setiap baris berikutnya.

VIII. UNGUIDED

Unguided 3

```
package main

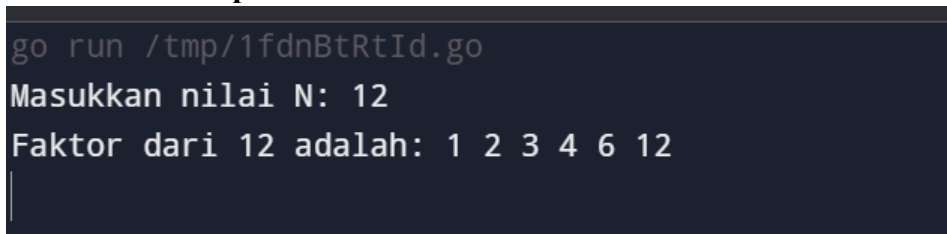
import (
    "fmt"
)

// Fungsi rekursif untuk menemukan dan mencetak faktor
// bilangan dari N
func printFactors(n, i int) {
    // Jika nilai i lebih besar dari N, berhenti
    if i > n {
        return
    }
    // Jika N habis dibagi i, maka i adalah faktor dari
    // N
    if n%i == 0 {
        fmt.Print(i, " ")
    }
    // Panggil fungsi printFactors dengan nilai i
    // berikutnya
    printFactors(n, i+1)
}

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

    fmt.Print("Faktor dari ", n, " adalah: ")
    printFactors(n, 1) // Mulai dari faktor 1
    fmt.Println()
}
```

Screenshoot Output



```
go run /tmp/1fdnBtRtId.go
Masukkan nilai N: 12
Faktor dari 12 adalah: 1 2 3 4 6 12
```

Deskripsi Program

Kode ini ditulis dalam bahasa Go (Golang) dan berfungsi untuk menemukan dan mencetak semua faktor dari bilangan bulat `n` menggunakan fungsi rekursif. Di dalam fungsi `main`, program meminta pengguna untuk memasukkan nilai `n`, yang akan digunakan sebagai bilangan yang faktornya dicari. Setelah menerima input `n`, program memanggil fungsi `printFactors(n, 1)` dengan nilai awal `i` sebesar 1 untuk memulai pencarian faktor dari bilangan `n`.

Fungsi `printFactors` adalah fungsi rekursif yang menemukan faktor dari bilangan `n`. Fungsi ini memiliki dua parameter: `n`, yang merupakan bilangan yang faktornya dicari, dan `i`, yang merupakan kandidat faktor yang diperiksa pada setiap pemanggilan. Fungsi ini berhenti ketika nilai `i` lebih besar dari `n`, karena tidak ada faktor yang lebih besar dari bilangan itu sendiri. Pada setiap pemanggilan, jika `n` habis dibagi `i` (yaitu `n % i == 0`), maka `i` dicetak sebagai faktor dari `n`. Kemudian, fungsi `printFactors` dipanggil kembali dengan nilai `i` yang ditingkatkan sebesar 1, sehingga proses ini akan memeriksa setiap angka dari 1 hingga `n` untuk menemukan semua faktor. Setelah seluruh faktor ditemukan, hasilnya ditampilkan dalam satu baris.

IX. UNGUIDED

Unguided 4

```
package main

import (
    "fmt"
)

// Fungsi rekursif untuk mencetak bilangan dari N hingga 1
func printDescending(n int) {
    if n == 0 {
        return
    }
    fmt.Print(n, " ")
    printDescending(n - 1)
}

// Fungsi rekursif untuk mencetak bilangan dari 1 hingga N
func printAscending(n, current int) {
    if current > n {
        return
    }
    fmt.Print(current, " ")
    printAscending(n, current+1)
}

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

    // Cetak bilangan dari N ke 1
    printDescending(n)
    // Cetak bilangan dari 2 ke N untuk menghindari
    pengulangan angka 1 dua kali
    printAscending(n, 2)
    fmt.Println()
}
```

Screenshoot Output

```
go run /tmp/1xJIB4L3pr.go
Masukkan nilai N: 5
5 4 3 2 1 2 3 4 5
```

Deskripsi Program

Kode ini ditulis dalam bahasa Go (Golang) dan berfungsi untuk mencetak bilangan secara berurutan dari `n` hingga 1, lalu dari 1 kembali ke `n` tanpa mengulangi angka 1 dua kali. Dalam fungsi `main`, program meminta pengguna memasukkan nilai `n`, yang akan menentukan batas bilangan yang dicetak dalam pola menurun dan menaik. Setelah menerima input `n`, program pertama-tama memanggil fungsi `printDescending(n)` untuk mencetak bilangan dari `n` hingga 1, lalu memanggil fungsi `printAscending(n, 2)` untuk mencetak bilangan dari 2 hingga `n`.

Fungsi `printDescending` adalah fungsi rekursif yang mencetak bilangan dari `n` hingga 1. Fungsi ini memiliki kondisi dasar `if n == 0` yang menghentikan rekursi ketika `n` mencapai 0. Pada setiap pemanggilan, jika `n` lebih besar dari 0, fungsi akan mencetak `n`, kemudian memanggil dirinya sendiri dengan nilai `n-1`, sehingga angka yang dicetak akan berkurang satu pada setiap langkah hingga mencapai 1.

Selanjutnya, fungsi `printAscending` adalah fungsi rekursif yang mencetak bilangan dari angka `current` hingga `n`. Kondisi dasar fungsi ini adalah `if current > n`, yang menghentikan rekursi ketika `current` lebih besar dari `n`. Pada setiap pemanggilan, fungsi akan mencetak nilai `current`, kemudian memanggil dirinya sendiri dengan nilai `current+1`, sehingga angka yang dicetak akan bertambah satu setiap kali hingga mencapai `n`. Memanggil `printAscending` dengan `current = 2` mencegah angka 1 dicetak dua kali, karena angka 1 sudah dicetak sebelumnya oleh fungsi `printDescending`.

X. UNGUIDED

Unguided 5

```
package main

import (
    "fmt"
)

// Fungsi rekursif untuk mencetak bilangan ganjil dari 1
// hingga N
func printOdd(n, current int) {
    // Jika nilai current lebih besar dari N, berhenti
    if current > n {
        return
    }
    // Cetak nilai current karena ganjil
    fmt.Print(current, " ")
    // Panggil rekursi dengan current ditambah 2 untuk
    // tetap ke bilangan ganjil
    printOdd(n, current+2)
}

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

    fmt.Print("Bilangan ganjil dari 1 hingga ", n, "
    adalah: ")
    printOdd(n, 1) // Mulai dari bilangan ganjil
    pertama, yaitu 1
    fmt.Println()
}
```

Screenshoot Output

```
go run /tmp/G3hRoLzTG9.go
Masukkan nilai N: 20
Bilangan ganjil dari 1 hingga 20 adalah: 1 3 5 7 9 11 13 15 17 19
```

Deskripsi Program

Kode ini ditulis dalam bahasa Go (Golang) dan berfungsi untuk mencetak bilangan ganjil dari 1 hingga `n` menggunakan fungsi rekursif. Dalam fungsi `main`, program meminta pengguna untuk memasukkan sebuah bilangan bulat `n`, yang menentukan batas atas dari bilangan ganjil yang akan dicetak. Setelah menerima input tersebut, program memanggil fungsi `printOdd(n, 1)` untuk memulai pencetakan dari bilangan ganjil pertama, yaitu 1.

Fungsi `printOdd` adalah fungsi rekursif yang mencetak bilangan ganjil hingga nilai `n`. Fungsi ini menerima dua parameter: `n`, yang merupakan batas atas, dan `current`, yang menunjukkan bilangan ganjil yang saat ini akan dicetak. Di dalam fungsi, terdapat kondisi dasar `if current > n` yang menghentikan proses rekursi ketika nilai `current` melebihi `n`. Pada setiap pemanggilan, fungsi mencetak nilai `current`, yang merupakan bilangan ganjil, kemudian memanggil dirinya sendiri dengan `current+2`, untuk tetap menghasilkan bilangan ganjil pada pemanggilan berikutnya. Proses ini berlanjut hingga semua bilangan ganjil dari 1 sampai `n` tercetak. Hasilnya adalah daftar bilangan ganjil yang ditampilkan di layar.

XI. UNGUIDED

Unguided 6

```
package main

import (
    "fmt"
)

// Fungsi rekursif untuk menghitung x pangkat y
func power(x, y int) int {
    // Base case: jika y == 0, hasilnya adalah 1
    if y == 0 {
        return 1
    }
    // Panggil rekursif untuk menghitung x^(y-1) dan
    kalikan dengan x
    return x * power(x, y-1)
}

func main() {
    var x, y int
    fmt.Print("Masukkan nilai x dan y (x^y): ")
    fmt.Scan(&x, &y)

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

Screenshoot Output

```
go run /tmp/XvR9VXcPaf.go
Masukkan nilai x dan y (x^y): 2 2
Hasil dari 2 pangkat 2 adalah: 4
```

Deskripsi Program

Kode ini ditulis dalam bahasa Go (Golang) dan berfungsi untuk menghitung nilai dari x pangkat y menggunakan metode rekursi. Di dalam fungsi `main`, program meminta pengguna untuk memasukkan dua bilangan bulat, x dan y , di mana x adalah basis dan y adalah eksponen. Setelah menerima input, program memanggil fungsi `power(x, y)` untuk melakukan perhitungan pangkat.

Fungsi `power` adalah fungsi rekursif yang menghitung nilai pangkat dengan dua parameter: x dan y . Fungsi ini memiliki kondisi dasar yang menetapkan bahwa jika y sama dengan 0, maka hasilnya adalah 1, karena setiap bilangan pangkat 0 sama dengan 1. Jika y lebih besar dari 0, fungsi ini akan memanggil dirinya sendiri dengan mengurangi nilai y sebesar 1, dan mengalikan hasil dari pemanggilan rekursif tersebut dengan x . Proses ini akan terus berlanjut hingga mencapai kondisi dasar. Setelah semua pemanggilan rekursif selesai, hasil akhir dari perhitungan pangkat ditampilkan di layar dengan format yang sesuai, memberikan pengguna informasi tentang hasil x^y .