

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL V

REKURSIF



Disusun Oleh :

Daffa Aryaputra / 2311102272

IF-11-06

Dosen Pengampu :

Abednego Dwi Septiadi

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Rekursi adalah teknik dalam pemrograman di mana sebuah fungsi memanggil dirinya sendiri untuk menyelesaikan sebuah masalah. Ini berguna saat kita punya masalah besar yang bisa dipecah menjadi beberapa masalah kecil yang serupa. Dalam bahasa pemrograman Golang (Go), rekursi sering digunakan untuk memecahkan masalah yang berpola berulang, seperti menghitung faktorial (perkalian bertingkat), deret Fibonacci, atau pencarian dalam struktur data pohon.

Membuat fungsi rekursif mirip seperti membuat fungsi biasa, tetapi fungsi tersebut akan memanggil dirinya sendiri. Agar rekursi tidak berjalan terus-menerus dan menyebabkan error, kita perlu menetapkan syarat atau kondisi dasar yang akan menghentikan proses tersebut. Misalnya, saat menghitung faktorial dari angka tertentu, fungsi akan memanggil dirinya sendiri dengan nilai yang lebih kecil, hingga mencapai angka 1, lalu mengalikan semua hasilnya dari yang paling kecil hingga terbesar.

Meskipun rekursi bisa membuat kode terlihat lebih simpel, teknik ini kadang memerlukan lebih banyak memori dan waktu, karena setiap pemanggilan fungsi membutuhkan ruang di memori hingga kondisi akhirnya terpenuhi. Di Golang, yang umumnya berfokus pada efisiensi, kadang rekursi bisa diganti dengan cara lain seperti perulangan biasa (iterasi) agar lebih cepat. Namun, rekursi tetap menjadi pilihan yang bagus saat kita menghadapi masalah yang melibatkan pola berulang atau struktur yang bersarang.

II. GUIDED

1.

Sourcecode

```
package main

import "fmt"

//fungsi 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 n hingga 1 : ")
    fmt.Scanln(&n)
    fmt.Print("Hasil cetak mundur : ")
    cetakMundur(n)
}

```

Screenshoot Output

```

PS C:\Users\ACER> go run "e:\golang semester 3\modul5_guided1.go"
Masukkan nilai n untuk cetak bilangan n hingga 1 : 5
Hasil cetak mundur : 54321

```

Deskripsi Program

Program ini dibuat untuk mencetak bilangan dari nilai n yang diinputkan pengguna hingga angka 1 secara menurun. Program ini menggunakan rekursi dalam fungsi cetakMundur untuk mencetak angka secara berurutan dari n ke 1.

Algoritma program dimulai dengan meminta input dari pengguna berupa nilai integer n. Setelah menerima input, program akan memanggil fungsi cetakMundur dengan nilai n tersebut. Di dalam fungsi cetakMundur, terdapat kondisi if yang mengecek apakah nilai n sudah mencapai 1. Jika ya, maka angka 1 dicetak dan fungsi berhenti. Namun, jika n lebih dari 1, maka angka n akan dicetak, dan fungsi cetakMundur dipanggil kembali dengan nilai n yang dikurangi 1.

Proses pemanggilan rekursif ini akan terus berlangsung hingga nilai n mencapai 1, mencetak semua bilangan dari n hingga 1. Setelah seluruh bilangan dicetak, program akan mengakhiri eksekusi. Sebagai contoh, jika pengguna memasukkan nilai n = 5, maka output yang dihasilkan oleh program adalah "5 4 3 2 1".

2.

Sourcecode

```

package main

import "fmt"

func jumlahRekursif(n int) int {

```

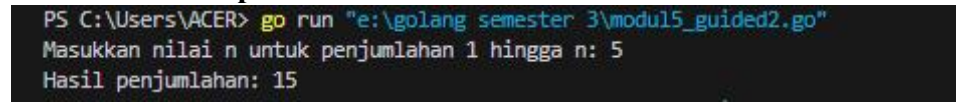
```
        if n == 1 {
            return 1
        }

        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



```
PS C:\Users\ACER> go run "e:\golang semester 3\modul5_guided2.go"
Masukkan nilai n untuk penjumlahan 1 hingga n: 5
Hasil penjumlahan: 15
```

Deskripsi Program

Program ini bertujuan untuk menghitung penjumlahan bilangan dari 1 hingga n, di mana n adalah nilai integer yang dimasukkan oleh pengguna. Program ini menggunakan rekursi melalui fungsi `jumlahRekursif` untuk menghitung total penjumlahan dari 1 sampai n.

Algoritma program ini dimulai dengan meminta input dari pengguna berupa nilai n. Setelah mendapatkan input, program akan memanggil fungsi `jumlahRekursif` dengan argumen n. Di dalam fungsi `jumlahRekursif`, terdapat kondisi yang mengecek apakah nilai n sama dengan 1. Jika iya, maka fungsi akan mengembalikan nilai 1, karena jumlah bilangan hingga 1 adalah 1. Namun, jika n lebih besar dari 1, fungsi akan menambahkan nilai n dengan hasil pemanggilan `jumlahRekursif` dengan argumen n - 1. Dengan cara ini, fungsi akan terus mengurangi nilai n hingga mencapai 1, lalu mulai menjumlahkan nilai-nilai yang telah disimpan pada setiap pemanggilan rekursif.

Setelah seluruh pemanggilan rekursif selesai dan hasil penjumlahan dari 1 hingga n diperoleh, nilai tersebut akan dikembalikan ke main dan dicetak sebagai output. Sebagai contoh, jika pengguna memasukkan nilai n = 5, maka output yang dihasilkan oleh program adalah "Hasil penjumlahan: 15", karena hasil penjumlahan 1 + 2 + 3 + 4 + 5 adalah 15.

3.

Sourcecode

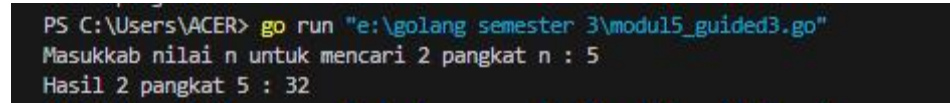
```
package main

import "fmt"

//fungsi untuk mencari 2 pangkat
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



```
PS C:\Users\ACER> go run "e:\golang semester 3\modul5_guided3.go"
Masukkan nilai n untuk mencari 2 pangkat n : 5
Hasil 2 pangkat 5 : 32
```

Deskripsi Program

Program ini dibuat untuk menghitung nilai 2 pangkat n, di mana n adalah bilangan integer yang dimasukkan oleh pengguna. Program ini menggunakan rekursi dalam fungsi pangkatDua untuk menghitung hasil perpangkatan 2 secara bertahap.

Algoritma program dimulai dengan meminta input dari pengguna berupa nilai n. Setelah mendapatkan nilai n, program akan memanggil fungsi pangkatDua dengan parameter n. Di dalam fungsi pangkatDua, terdapat kondisi yang mengecek apakah nilai n sama dengan 0. Jika n bernilai 0, fungsi akan mengembalikan 1, karena 2 pangkat 0 adalah 1. Jika n lebih besar dari 0, fungsi akan mengalikan 2 dengan hasil pemanggilan fungsi pangkatDua dengan nilai n - 1. Dengan cara ini, fungsi akan terus berkurang nilai n hingga mencapai 0, lalu mengalikan nilai 2 pada setiap langkah rekursif.

Setelah seluruh proses rekursif selesai dan hasil perpangkatan diperoleh, nilai tersebut dikembalikan ke fungsi main dan dicetak sebagai output. Sebagai contoh, jika pengguna memasukkan nilai $n = 3$, maka output yang dihasilkan adalah "Hasil 2 pangkat 3 : 8", karena hasil 2 pangkat 3 adalah 8.

4.

Sourcecode

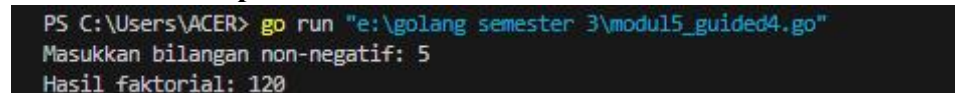
```
package main
import "fmt"

func main() {
    var n int
    fmt.Println("Masukkan bilangan non-negatif: ")
    fmt.Scan(&n)

    if n < 0 {
        fmt.Println("Faktorial tidak didefinisikan untuk bilangan negatif")
    } else {
        fmt.Println("Hasil faktorial:", faktorial(n))
    }
}

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

Screenshoot Output



```
PS C:\Users\ACER> go run "e:\golang semester 3\modul5_guided4.go"
Masukkan bilangan non-negatif: 5
Hasil faktorial: 120
```

Deskripsi Program

Program ini bertujuan untuk menghitung faktorial dari sebuah bilangan non-negatif yang dimasukkan oleh pengguna. Program menggunakan rekursi dalam fungsi faktorial untuk menghitung hasil faktorial dari bilangan tersebut.

Algoritma program ini dimulai dengan meminta pengguna memasukkan sebuah bilangan integer n . Setelah menerima input, program memeriksa apakah nilai n bernilai negatif. Jika n bernilai negatif, program akan menampilkan pesan bahwa faktorial tidak didefinisikan untuk bilangan negatif, dan program tidak akan melanjutkan perhitungan. Namun, jika n bernilai 0 atau lebih, program akan memanggil fungsi faktorial untuk menghitung hasil faktorial dari n .

Di dalam fungsi faktorial, terdapat kondisi yang mengecek apakah nilai n sama dengan 0 atau 1. Jika ya, fungsi akan mengembalikan nilai 1, karena faktorial dari 0 dan 1 adalah 1. Jika n lebih besar dari 1, fungsi akan mengembalikan hasil perkalian n dengan hasil faktorial($n-1$), sehingga nilai n akan terus berkurang hingga mencapai 1. Proses rekursif ini akan mengalikan setiap nilai dari n hingga 1, menghasilkan nilai faktorial yang diinginkan.

Setelah proses rekursif selesai, nilai faktorial yang diperoleh dikembalikan ke fungsi main dan dicetak sebagai output. Misalnya, jika pengguna memasukkan $n = 5$, output yang dihasilkan adalah "Hasil faktorial: 120", karena $5! = 5 * 4 * 3 * 2 * 1 = 120$.

III. UNGUIDED

1.

Sourcecode

```
package main

import "fmt"

func fibonacci(n int) int {
    if n == 0 {
        return 0
    } else if n == 1 {
        return 1
    }
}
```

```

        } else {

            return fibonacci(n-1) + fibonacci(n-2)

        }

    }

func main() {

    fmt.Println("Deret Fibonacci hingga suku ke-10:")

    for i := 0; i <= 10; i++ {

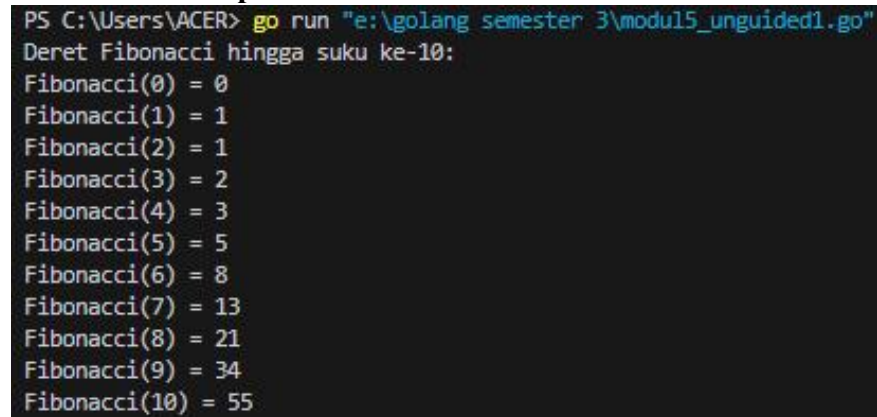
        fmt.Printf("Fibonacci(%d) = %d\n", i, fibonacci(i))

    }

}

```

Screenshoot Output



```

PS C:\Users\ACER> go run "e:\golang semester 3\modul5_unguided1.go"
Deret Fibonacci hingga suku ke-10:
Fibonacci(0) = 0
Fibonacci(1) = 1
Fibonacci(2) = 1
Fibonacci(3) = 2
Fibonacci(4) = 3
Fibonacci(5) = 5
Fibonacci(6) = 8
Fibonacci(7) = 13
Fibonacci(8) = 21
Fibonacci(9) = 34
Fibonacci(10) = 55

```

Deskripsi Program

Program ini dibuat untuk menghitung dan menampilkan deret Fibonacci hingga suku ke-10. Deret Fibonacci adalah suatu deret bilangan di mana setiap bilangan merupakan hasil penjumlahan dari dua bilangan sebelumnya, dengan dua bilangan pertama adalah 0 dan 1. Jadi, urutan Fibonacci dimulai dari 0, 1, 1, 2, 3, 5, 8, dan seterusnya. Program ini menggunakan rekursi dalam fungsi fibonacci untuk menghitung setiap suku dari deret tersebut.

Algoritma program ini dimulai dengan mendefinisikan fungsi fibonacci, yang menerima sebuah parameter n dan mengembalikan nilai suku ke-n dari deret Fibonacci. Fungsi ini memiliki kondisi dasar untuk menangani dua kasus awal, yaitu jika n adalah 0 atau 1. Jika n sama dengan 0, fungsi mengembalikan nilai 0; jika n sama dengan 1, fungsi mengembalikan nilai 1. Namun, jika n lebih besar dari 1, fungsi akan memanggil dirinya sendiri dengan argumen n-1 dan n-2, lalu menjumlahkan hasil dari kedua pemanggilan tersebut. Dengan cara ini, setiap suku dihitung sebagai penjumlahan dari dua suku sebelumnya.

Pada fungsi main, program menampilkan deret Fibonacci hingga suku ke-10 dengan cara menjalankan perulangan dari 0 hingga 10. Pada setiap iterasi, program akan memanggil fungsi fibonacci dengan nilai iterasi saat ini sebagai argumen, lalu mencetak hasilnya sebagai suku ke-i dari deret Fibonacci. Output yang dihasilkan menampilkan nilai setiap suku Fibonacci dari suku ke-0 hingga suku ke-10.

2.

Sourcecode

```
package main

import "fmt"

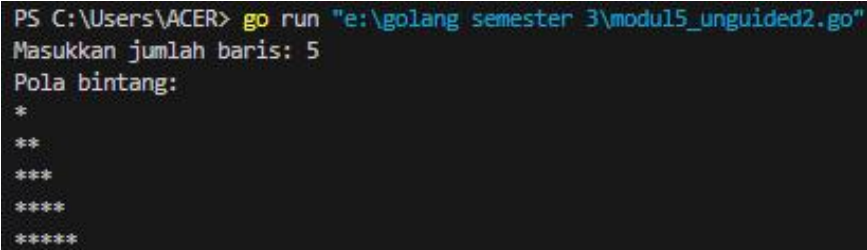
func printStars(n int) {
    if n > 0 {
        fmt.Print("*")
        printStars(n - 1)
    }
}

func printPattern(n int) {
    if n > 0 {
        printPattern(n - 1)
        printStars(n)
        fmt.Println()
    }
}

func main() {
    var n int
    fmt.Print("Masukkan jumlah baris: ")
    fmt.Scan(&n)
```

```
    fmt.Println("Pola bintang:")
    printPattern(n)
}
```

Screenshoot Output



```
PS C:\Users\ACER> go run "e:\golang semester 3\modul5_unguided2.go"
Masukkan jumlah baris: 5
Pola bintang:
*
**
***
****
*****
```

Deskripsi Program

Program ini dibuat untuk menampilkan pola bintang segitiga dengan menggunakan rekursi. Pengguna akan memasukkan jumlah baris, dan program akan mencetak pola bintang yang dimulai dari satu bintang pada baris pertama, lalu bertambah satu bintang di setiap baris berikutnya hingga baris terakhir sesuai jumlah baris yang diminta.

Algoritma program ini terdiri dari dua fungsi rekursif utama, yaitu `printStars` dan `printPattern`. Fungsi `printStars` bertanggung jawab untuk mencetak sejumlah bintang dalam satu baris. Fungsi ini menerima sebuah parameter `n`, yang menentukan jumlah bintang yang akan dicetak. Jika nilai `n` lebih dari 0, fungsi akan mencetak satu bintang ("`*`") dan memanggil dirinya sendiri dengan nilai `n-1`, sehingga bintang akan terus dicetak hingga `n` mencapai 0.

Fungsi kedua, `printPattern`, digunakan untuk mencetak pola bintang dalam beberapa baris. Fungsi ini menerima parameter `n`, yang menunjukkan jumlah baris yang harus dicetak. Jika `n` lebih dari 0, fungsi akan memanggil dirinya sendiri dengan `n-1` untuk mencetak baris sebelumnya terlebih dahulu, sehingga baris pertama dicetak terlebih dahulu. Setelah memanggil `printPattern` untuk baris sebelumnya, fungsi akan memanggil `printStars` dengan parameter `n` untuk mencetak sejumlah bintang yang sesuai pada baris saat ini, lalu mencetak baris baru.

Dalam fungsi main, program akan meminta pengguna untuk memasukkan jumlah baris `n`. Setelah input diterima, program akan memanggil `printPattern(n)` untuk mencetak pola bintang yang sesuai.

3.

Sourcecode

```
package main

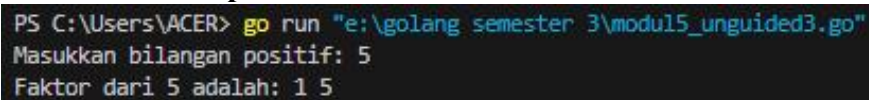
import "fmt"

func printFactors(n, divisor int) {
    if divisor > n {
        return
    }
    if n%divisor == 0 {
        fmt.Printf("%d ", divisor)
    }
    printFactors(n, divisor+1)
}

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

    fmt.Printf("Faktor dari %d adalah: ", n)
    printFactors(n, 1)
    fmt.Println()
}
```

Screenshoot Output



```
PS C:\Users\ACER> go run "e:\golang semester 3\modul5_unguided3.go"
Masukkan bilangan positif: 5
Faktor dari 5 adalah: 1 5
```

Deskripsi Program

Program ini dibuat untuk menampilkan faktor-faktor dari suatu bilangan positif yang dimasukkan oleh pengguna. Faktor dari sebuah bilangan n adalah bilangan-bilangan yang dapat membagi n tanpa menghasilkan sisa. Program ini menggunakan rekursi melalui fungsi `printFactors` untuk memeriksa dan mencetak setiap faktor dari bilangan tersebut.

Algoritma program dimulai dengan meminta input bilangan positif dari pengguna. Setelah menerima input, program akan memanggil fungsi `printFactors` dengan parameter n (bilangan yang dimasukkan pengguna)

dan divisor (nilai pembagi yang dimulai dari 1). Fungsi printFactors bertugas untuk memeriksa setiap bilangan mulai dari 1 hingga n untuk menentukan apakah bilangan tersebut merupakan faktor dari n.

Di dalam printFactors, terdapat dua kondisi utama. Kondisi pertama memeriksa apakah nilai divisor sudah lebih besar dari n. Jika ya, maka fungsi berhenti karena semua pembagi yang mungkin sudah diperiksa. Kondisi kedua memeriksa apakah nilai n habis dibagi dengan divisor (dengan mengecek $n \% divisor == 0$). Jika benar, berarti divisor adalah faktor dari n, dan fungsi akan mencetak nilai divisor. Terlepas dari apakah divisor adalah faktor atau bukan, fungsi akan memanggil dirinya sendiri dengan divisor+1 untuk melanjutkan pemeriksaan pada angka berikutnya, hingga mencapai batas n.

4.

Sourcecode

```
package main

import (
    "fmt"
)

func printSequence(n, current int) {
    if current < 1 {
        return
    }

    fmt.Print(current, " ")

    if current > 1 {
        printSequence(n, current-1)
    }

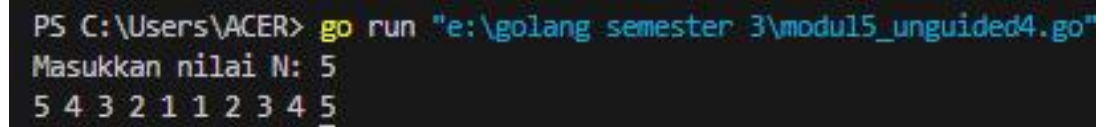
    fmt.Print(current, " ")
}

func main() {
    var n int

    fmt.Print("Masukkan nilai N: ")
    fmt.Scan(&n)
```

```
    printSequence(n, n)
    fmt.Println()
}
```

Screenshoot Output



```
PS C:\Users\ACER> go run "e:\golang semester 3\modul5_unguided4.go"
Masukkan nilai N: 5
5 4 3 2 1 1 2 3 4 5
```

Deskripsi Program

Program ini dibuat untuk mencetak urutan bilangan yang dimulai dari bilangan n (yang dimasukkan oleh pengguna) hingga 1, lalu kembali lagi dari 1 hingga n . Program ini menggunakan rekursi dalam fungsi `printSequence` untuk menghasilkan pola urutan tersebut.

Algoritma program dimulai dengan meminta input dari pengguna berupa bilangan positif n . Setelah menerima nilai n , program akan memanggil fungsi `printSequence` dengan parameter n sebagai nilai awal dan `current` sebagai nilai yang sama dengan n . Fungsi `printSequence` bertugas mencetak urutan angka dari n ke 1, lalu kembali dari 1 ke n .

Di dalam fungsi `printSequence`, terdapat dua kondisi utama. Kondisi pertama memeriksa apakah nilai `current` lebih kecil dari 1. Jika iya, fungsi berhenti, karena tidak ada angka yang perlu dicetak. Jika `current` bernilai lebih dari atau sama dengan 1, fungsi akan mencetak nilai `current` diikuti oleh spasi. Selanjutnya, fungsi akan memeriksa apakah `current` lebih besar dari 1. Jika iya, maka `printSequence` dipanggil kembali dengan nilai `current - 1`, sehingga mencetak urutan angka yang berkurang hingga mencapai 1.

Setelah mencapai 1, fungsi rekursif mulai kembali (`unwind`), dan setiap pemanggilan rekursif mencetak nilai `current` saat kembali dari rekursi. Hal ini menciptakan efek pencetakan urutan angka dari 1 kembali ke n , menghasilkan pola simetris.

5.

Sourcecode

```
package main

import (
```

```

        "fmt"
    )

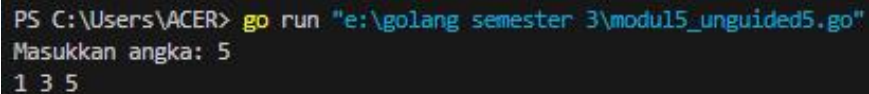
    func printOddSeries(n, current int) {
        if current <= n {
            fmt.Print(current, " ")
            printOddSeries(n, current+2)
        }
    }

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

        printOddSeries(n, 1)
        fmt.Println()
    }

```

Screenshoot Output



```

PS C:\Users\ACER> go run "e:\golang semester 3\modul5_unguided5.go"
Masukkan angka: 5
1 3 5

```

Deskripsi Program

Program ini dirancang untuk mencetak deret angka ganjil mulai dari 1 hingga angka n yang dimasukkan oleh pengguna. Program ini menggunakan rekursi dalam fungsi `printOddSeries` untuk menghasilkan dan menampilkan deret angka ganjil.

Algoritma program dimulai dengan meminta pengguna untuk memasukkan sebuah angka positif n . Setelah menerima input, program akan memanggil fungsi `printOddSeries` dengan dua argumen: n (angka batas atas) dan `current` (angka saat ini yang dimulai dari 1). Fungsi `printOddSeries` bertanggung jawab untuk mencetak angka ganjil hingga mencapai angka n .

Di dalam fungsi `printOddSeries`, terdapat kondisi yang memeriksa apakah nilai `current` kurang dari atau sama dengan n . Jika kondisi ini terpenuhi, program akan mencetak nilai `current`, diikuti oleh spasi. Setelah mencetak nilai tersebut, fungsi akan dipanggil kembali dengan argumen `current+2` untuk melanjutkan pencetakan angka ganjil berikutnya. Proses ini

berlanjut hingga nilai current lebih besar dari n, pada titik mana fungsi akan berhenti dan tidak mencetak angka lebih lanjut.

6.

Sourcecode

```
package main

import (
    "fmt"
)

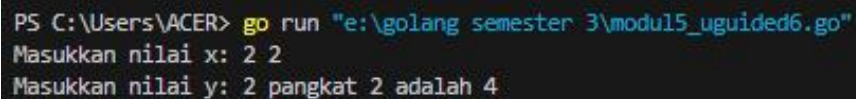
func power(x, y int) int {
    if y == 0 {
        return 1
    }

    return x * power(x, y-1)
}

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

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

Screenshoot Output



```
PS C:\Users\ACER> go run "e:\golang semester 3\modul5_uguided6.go"
Masukkan nilai x: 2 2
Masukkan nilai y: 2 pangkat 2 adalah 4
```

Deskripsi Program

Program ini dibuat untuk menghitung nilai dari x pangkat y (ditulis sebagai x^y), di mana x dan y adalah bilangan bulat yang dimasukkan oleh pengguna. Program ini menggunakan rekursi dalam fungsi power untuk menghitung hasil perpangkatan tersebut.

Algoritma program dimulai dengan meminta pengguna untuk memasukkan dua bilangan, yaitu x (basis) dan y (pangkat). Setelah kedua nilai tersebut diinput, program akan memanggil fungsi power dengan x dan y sebagai argumen untuk menghitung nilai pangkatnya.

Di dalam fungsi power, terdapat kondisi dasar yang memeriksa apakah nilai y sama dengan 0. Jika y bernilai 0, fungsi akan mengembalikan 1, sesuai dengan aturan matematika bahwa setiap bilangan pangkat 0 adalah 1. Jika y lebih besar dari 0, fungsi akan mengembalikan hasil kali antara x dan pemanggilan fungsi power dengan argumen x dan $y-1$. Proses ini akan terus berlangsung hingga nilai y mencapai 0, pada titik mana hasil akhir dari x^y akan dihitung.

Setelah proses rekursif selesai dan nilai pangkat diperoleh, hasilnya akan dikembalikan ke fungsi main, yang kemudian mencetak hasilnya dalam format yang mudah dibaca. Sebagai contoh, jika pengguna memasukkan nilai $x = 2$ dan $y = 2$, output yang dihasilkan akan berupa: 2 pangkat 2 adalah 4.