

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL V
REKURSIF**



Disusun Oleh :

Deshan Rafif Alfarisi / 2311102326

S1-IF-11-06

Dosen Pengampu :

Abednego Dwi Septiadi S.Kom., M.Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

1.1 Pengantar Rekursif

Pada modul-modul sebelumnya sudah dijelaskan bahwa suatu subprogram baik fungsi atau prosedur bisa memanggil subprogram lainnya. Hal ini tidak menutup kemungkinan bahwa subprogram yang dipanggil adalah dirinya sendiri. Dalam pemrograman terapan ini dikenal dengan istilah rekursif.

Rekursif secara sederhana dapat diartikan sebagai cara menyelesaikan suatu masalah dengan cara menyelesaikan sub-masalah yang identik dari masalah utama.

Sebagai contoh perhatikan prosedur cetak berikut ini!

	Notasi Algoritma	Notasi dalam bahasa GO
2	procedure cetak(in x:integer)	func cetak(x int){
3	algoritma	fmt.Println(x) cetak(x+1)
4	output(x)	}
5	cetak(x+1)	
5	endprocedure	

Apabila diperhatikan subprogram **cetak()** di atas, terlihat pada baris ke-4 terdapat pemanggilan subprogram **cetak()** kembali. Misalnya apabila Rita memasukkan perintah **cetak(S)** 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**.

```
1 package main
2 import "fmt"
3 func main() {
4     cetak(S)
5 }
6 func cetak(x int) {
7     fmt.Println(x)
8     cetak(x+1)
9 }
```

D:\DEV\DEMO>go build contoh.go

D:\DEV\DEMO>contoh.exe

5
6
7
8
9
10
11
12
13


Oleh karena itu bisa ditambahkan struktur Kontrol percabangan (if-then) untuk menghentikan proses rekursif ini. Kondisi ini disebut juga dengan **base-case**, artinya apabila kondisi base-case bernilai true maka proses rekursif akan berhenti. Sebagai contoh misalnya base case adalah ketika x bernilai 10 atau $x == 10$, maka tidak perlu dilakukan rekursif.

```

2  procedure cetak(in x:integer)
3  algoritma
4      if x == 10 then
5          output(x)
6      else
7          output(x)
8          cetak(x+1)
9      endif
endprocedure

```

Apabila diperhatikan pada baris Re-3 di Program di atas, Rita telah menambahkan **base-case** seperti penjelasan sebelumnya. Selanjutnya pada bagian aksi dari else di baris Re-6 dan Re-7 Rita nama **recursive-case** atau kasus pemanggilan dirinya sendiri tersebut terjadi. Kondisi dari **recursive-case** ini adalah negasi dari kondisi **base-case** atau ketika nilai $x \neq 10$.



```

2  package main
3  import "fmt"
4  func main(){
5      cetak(5)
6  }
7  func cetak(x int){
8      if x == 10 {
9          fmt.Println(x)
10         }else{
11             fmt.Println(x)
12             cetak(x+1)
13         }
14     }
15 }

```

```

D:\DEV\DEMO>go build contoh.go
D:\DEV\DEMO>contoh.exe

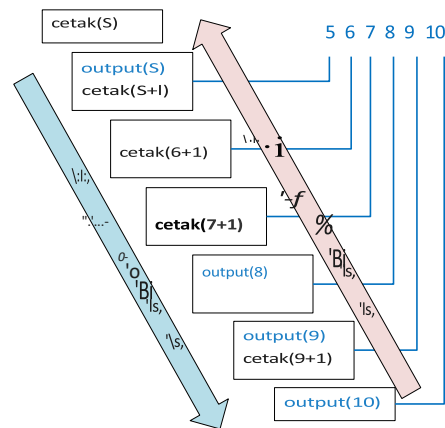
```

```

5
6
7
8
9
10

```

Apabila program di atas ini dijalankan maka akan tampil angka 5 6 7 8 9 10. Terlihat bahwa proses rekursif berhasil dihentikan ketika $x == 10$.



Gambar 1. 1/ustrasi proses forward dan backward pada saat rekursif.

Pada Gambar 2 memperlihatkan saat subprogram dipanggil secara rekursif, maka subprogram akan terus melakukan pemanggilan (**forward**) hingga berhenti pada saat kondisi **base-case** terpenuhi atau **true**. Setelah itu akan terjadi proses **backward** atau kembali ke subprogram yang sebelumnya. Artinya setelah semua instruksi **cetak(10)** selesai dieksekusi, maka program akan kembali ke **cetak(9)** yang memanggil **cetak(10)** tersebut. Begitu seterusnya hingga kembali ke **cetak(5)**.

Perhatikan modifikasi program di atas dengan menukar posisi baris 10 dan 11, maka akan terlihat bahwa program dijalankan maka akan menampilkan 10 9 8 7 6 5. Kenapa bisa demikian? Pahami proses **backward** pada Gambar 2

```

2 package main
3 import "fmt"
4 func main(){
5     cetak(5)
6 }
7 func cetak(x int){ if
8     x == 10 {
9         fmt.Println(x)
10    }else{
11        cetak(x+1)
12        fmt.Println(x)
13    }
14 }

```

D:\DEV\DEMO>go build contoh.go

D:\DEV\DEMO>contoh.exe

10
9
8
7
6
5

Catatan:

- TelmiR reRursif ini merupaRan salah satu alternatif untuR mengganti struRtur Rontrol perulangan dengan memanfaatkan subprogram (bisa fungsi ataupun prosedur).
- UntuR menghentiRan proses reRursif digunaRan percabangan (if-then).
- **Base-case** adalah Rondisi proses reRursif berhenti. **Base-case** merupaRan hal terpenting dan pertama yang harus diRetahui RetiRa aRan membuat program reRursif. **Mustahil** membuat program reRursif tanpa mengetahui **base-case** terlebih dahulu.
- **Recursive-case** adalah Rondisi dimana proses pemanggilan dirinya sendiri dilaRuRan. Kondisi **recursive-case** adalah Romplemen atau negasi dari **base-case**.
- Setiap algoritma reRursif selalu memilikiRi padanan dalam bentuR algoritma interatif.

1.1 Komponen Relwrsif

Algoritma reRursif terdiri dari dua Romponen utama:

- **Base-case (Basis)**, yaitu bagian untuR menghentiRan proses reRursif dan menjadi Romponen terpenting di dalam sebuah reRursif.
- **Recursive-case**, yaitu bagian pemanggilan subprogramnya.

1.2 Contoh Program dengan menggunaRan ReRursif

- a. Membuat baris bilangan dari n hingga 1
Base-case: bilangan == 1

```
2 package main
3 import "fmt"
4 func main(){
5     var n int
6     fmt.Scan(&n) baris(n)
7 }
8
9
10 func baris(bilangan int){ if
11     bilangan == 1 {
12         fmt.Println(1)
13     }else{
14         fmt.Println(bilangan) baris(bilangan -
15         1)
16     }
17 }
```

- b. Menghitung hasil
penjumlahan 1 hingga n
Base-case: n == 1

```

1  package main
2  import "fmt"
3  func main(){
4      var n int
5      fmt.Scan(&n)
6      fmt.Println(penjumlahan(n))
7  }
8
9  func penjumlahan(n int) int {
10     if n == 1 {
11         return 1
12     }else{
13         return n + penjumlahan(n-1)
14     }
15 }

```

c. Mencari dua pangkat n

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

```

1  package main
2  import "fmt"
3  func main(){
4      var n int
5      fmt.Scan(&n)
6      fmt.Println(pangkat(n))
7  }
8
9  func pangkat(n int) int {
10     if n == 0 {
11         return 1
12     }else{
13         return 2 * pangkat(n-1)
14     }
15 }

```

d. Mencari nilai faktorial atau $n!$

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

```

2  package main
3  import "fmt"
4  func main(){
5      var n int
6      fmt.Scan(&n)
7      fmt.Println(faktorial(n))
8  }
9
10 func faktorial(n int) int {
11     if n == 0 || n == 1 {
12         return 1
13     }else{
14         return n * faktorial(n-1)
15     }
16 }

```

II. GUIDED

1. Sourcecode

```
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

```
PS C:\Users\Lenovo\Documents\file kuliah\Laprak Alpro 2\modul 5> go run "c:\Users\Lenov
o\Documents\file kuliah\Laprak Alpro 2\modul 5\guided1.go"
Masukkan nilai n untuk cetak bilangan dari n hingga 1: 20
Hasil cetak mundur: 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
PS C:\Users\Lenovo\Documents\file kuliah\Laprak Alpro 2\modul 5> 
```

Deskripsi Program

Program ini mencetak bilangan dari nilai yang diberikan (n) hingga 1 secara mundur. Dengan rekursif, di mana fungsi cetakMundur memanggil dirinya sendiri dengan nilai argumen yang lebih kecil satu per satu. Proses ini berulang hingga nilai n mencapai 1. Setiap pemanggilan fungsi akan mencetak nilai n saat itu, lalu memanggil fungsi itu sendiri untuk mencetak nilai berikutnya yang lebih kecil. Dengan demikian, program ini menghasilkan output berupa urutan bilangan menurun dari n hingga 1.

2. Sourcecode

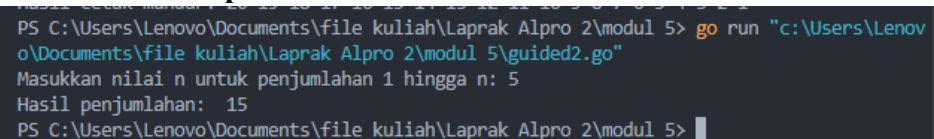
```
package main

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



```
PS C:\Users\Lenovo\Documents\file kuliah\Laprak Alpro 2\modul 5> go run "c:\Users\Lenovo\Documents\file kuliah\Laprak Alpro 2\modul 5\guided2.go"
Masukkan nilai n untuk penjumlahan 1 hingga n: 5
Hasil penjumlahan: 15
PS C:\Users\Lenovo\Documents\file kuliah\Laprak Alpro 2\modul 5>
```

Deskripsi Program

Program ini untuk menghitung jumlah bilangan bulat dari 1 hingga suatu nilai n yang diberikan oleh pengguna. Fungsi `jumlahRekursif` memanggil dirinya sendiri dengan nilai argumen yang lebih kecil satu per satu. Setiap pemanggilan fungsi akan menambahkan nilai n saat itu dengan hasil pemanggilan fungsi sebelumnya. Proses ini berulang hingga nilai n mencapai 1. Dengan demikian, program ini secara efektif menghitung jumlah seluruh bilangan dari 1 hingga n dengan cara yang rekursif.

3. Sourcecode

```
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

```
PS C:\Users\Lenovo\Documents\file kuliah\Laprak Alpro 2\modul 5> go run "c:\Users\Lenovo\Documents\file kuliah\Laprak Alpro 2\modul 5\guided3.go"
Masukkan nilai n untuk mencari 2 pangkat n: 4
Hasil 2 pangkat 4 : 16
PS C:\Users\Lenovo\Documents\file kuliah\Laprak Alpro 2\modul 5>
```

Deskripsi Program

Program ini menghitung nilai 2 pangkat n dengan menggunakan rekursif. Fungsi pangkatDua secara berulang memanggil dirinya sendiri dengan nilai n yang dikurangi satu setiap kali pemanggilan. Jika n sama dengan 0, fungsi akan mengembalikan nilai 1 (karena 2 pangkat 0 adalah 1). Hasil dari setiap pemanggilan fungsi akan dikalikan dengan 2, sehingga secara efektif menghitung nilai 2 pangkat n. Dengan kata lain, program ini memecahkan masalah menghitung pangkat dua dengan cara memecahnya menjadi masalah yang lebih sederhana (menghitung pangkat dua dari bilangan yang lebih kecil) dan kemudian menggabungkan hasilnya.

4. Sourcecode

```
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

```
PS C:\Users\Lenovo\Documents\file kuliah\Laprak Alpro 2\modul 5> go run "c:\Users\Lenovo\Documents\file kuliah\Laprak Alpro 2\modul 5\guided4.go"
Masukkan nilai n untuk mencari faktorial n!: 4
Hasil faktorial dari 4 : 24
PS C:\Users\Lenovo\Documents\file kuliah\Laprak Alpro 2\modul 5>
```

Deskripsi Program

Program ini menghitung faktorial dari suatu bilangan bulat yang diberikan oleh pengguna. Fungsi faktorial secara berulang memanggil dirinya sendiri dengan nilai n yang dikurangi satu setiap kali pemanggilan. Jika n sama dengan 0 atau 1, fungsi akan langsung mengembalikan nilai 1 (karena faktorial dari 0 dan 1 adalah 1). Hasil dari setiap pemanggilan fungsi akan dikalikan dengan nilai n saat itu, sehingga secara efektif menghitung faktorial dari n . Dengan kata lain, program ini memecahkan masalah menghitung faktorial dengan cara memecahnya menjadi masalah yang lebih sederhana (menghitung faktorial dari bilangan yang lebih kecil) dan kemudian menggabungkan hasilnya.

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

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"

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

func main() {
    // contoh penggunaan fungsi fibonacci
    for i := 0; i <= 10; i++ {
        fmt.Printf("Fibonacci ke-%d: %d\n", i, fibonacci(i))
    }
}
```

Screenshoot Output

```
Fibonacci ke-1: 1
Fibonacci ke-2: 1
Fibonacci ke-3: 2
Fibonacci ke-4: 3
Fibonacci ke-5: 5
Fibonacci ke-6: 8
Fibonacci ke-7: 13
Fibonacci ke-8: 21
Fibonacci ke-9: 34
Fibonacci ke-10: 55
PS C:\Users\Lenovo\Documents\file kuliah\Laparak Alpro 2\modul 5>
```

Deskripsi Program

Program ini untuk menghitung deret Fibonacci menggunakan konsep rekursif. Fungsi fibonacci secara rekursif memanggil dirinya sendiri untuk menghitung nilai Fibonacci berdasarkan rumus yang telah ditentukan, yaitu nilai Fibonacci suatu bilangan adalah hasil penjumlahan dari dua bilangan Fibonacci sebelumnya. Fungsi main kemudian menggunakan fungsi fibonacci untuk mencetak nilai-nilai Fibonacci dari 0 hingga 10. Rekursi memungkinkan fungsi untuk memecahkan masalah yang besar menjadi masalah-masalah yang lebih kecil yang serupa, sehingga solusi dapat ditemukan dengan cara yang elegan namun mungkin kurang efisien untuk nilai n yang sangat besar.

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"

func printStars(n int) {
    if n <= 0 {
        return
    }
    printStars(n - 1)
    for i := 0; i < n; i++ {
        fmt.Print("*")
    }
    fmt.Println()
}

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

Screenshoot Output

```
o\Documents\file kuliah\Laprak Alpro 2\modul 5\unguided2.go
Masukkan jumlah baris: 5
*
**
***
****
*****
PS C:\Users\Lenovo\Documents\file kuliah\Laprak Alpro 2\modul 5>
```

Deskripsi Program

Program ini mencetak pola bintang berbentuk segitiga siku-siku menggunakan konsep rekursif. Fungsi `printStars` secara rekursif memanggil dirinya sendiri untuk mencetak baris bintang sebanyak `n` kali. Pada setiap pemanggilan rekursif, jumlah bintang yang dicetak akan bertambah satu. Proses rekursif ini berulang hingga `n` mencapai 0. Fungsi `main` digunakan untuk meminta input jumlah baris dari pengguna dan kemudian memanggil fungsi `printStars` untuk mencetak pola bintang sesuai dengan input tersebut. Dengan kata lain, program ini memanfaatkan rekursi untuk memecahkan masalah mencetak pola bintang menjadi masalah yang lebih sederhana, yaitu mencetak baris bintang secara berulang.

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"

func findFactors(num, divisor int) {
    if divisor > num {
        return
    }
    if num%divisor == 0 {
        fmt.Print(divisor, " ")
    }
    findFactors(num, divisor+1)
}
```

```
func main() {
    var number int
    fmt.Print("Masukkan bilangan: ")
    fmt.Scan(&number)
    fmt.Printf("Faktor dari %d adalah: ", number)
    findFactors(number, 1)
    fmt.Println()
}
```

Screenshoot Output

```
o\Documents\file kuliah\Laprak Alpro 2\modul 5\unguided3.go"
Masukkan bilangan: 12
Faktor dari 12 adalah: 1 2 3 4 6 12
PS C:\Users\Lenovo\Documents\file kuliah\Laprak Alpro 2\modul 5> |
```

Deskripsi Program

Program ini untuk mencari dan mencetak semua faktor dari sebuah bilangan bulat yang diberikan oleh pengguna. Konsep rekursif digunakan dalam fungsi findFactors untuk menyelesaikan masalah ini. Fungsi ini secara berulang memanggil dirinya sendiri dengan nilai divisor yang terus bertambah satu. Pada setiap pemanggilan, fungsi memeriksa apakah divisor merupakan faktor dari num. Jika ya, maka divisor tersebut akan dicetak. Proses rekursif ini berlanjut hingga nilai divisor melebihi nilai num. Dengan demikian, fungsi findFactors secara sistematis memeriksa semua bilangan bulat dari 1 hingga num untuk menemukan faktor-faktornya. Fungsi main berfungsi sebagai titik masuk program, meminta input bilangan dari pengguna, dan kemudian memanggil fungsi findFactors untuk menghitung dan mencetak faktor-faktornya.

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 printSequence(n int) {
    if n == 0 {
        return
    }
    fmt.Print(n, " ")
    printSequence(n - 1)
    fmt.Print(n, " ")
}

func main() {
    var number int
    fmt.Print("Masukkan bilangan: ")
    fmt.Scan(&number)
    fmt.Printf("Barisan bilangan: ")
    printSequence(number)
    fmt.Println()
}

```

Screenshoot Output

```

o:\Documents\file kuliah\Laparak Alpro 2\modul 5\unguided4.go"
Masukkan bilangan: 5
Barisan bilangan: 5 4 3 2 1 1 2 3 4 5
PS C:\Users\Lenovo\Documents\file kuliah\Laparak Alpro 2\modul 5>

```

Deskripsi Program

Program ini untuk mencetak sebuah barisan bilangan yang unik, di mana setiap bilangan muncul dua kali: sekali di awal dan sekali di akhir. Konsep rekursif digunakan dalam fungsi `printSequence` untuk menghasilkan barisan ini. Fungsi ini secara rekursif memanggil dirinya sendiri dengan nilai `n` yang terus berkurang satu. Pada setiap pemanggilan, bilangan `n` dicetak, kemudian fungsi memanggil dirinya sendiri untuk mencetak bilangan-bilangan sebelumnya, dan setelah itu bilangan `n` dicetak lagi. Proses rekursif ini berulang hingga `n` mencapai 0. Fungsi `main` digunakan untuk meminta input bilangan dari pengguna dan kemudian memanggil fungsi `printSequence` untuk mencetak barisan bilangan sesuai dengan input tersebut.

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"

func printOddNumbers(n int) {
    if n < 1 {
        return
    }
    printOddNumbers(n - 2)
    fmt.Print(n, " ")
}

func main() {
    var number int
    fmt.Print("Masukkan bilangan: ")
    fmt.Scan(&number)
    fmt.Printf("Barisan bilangan ganjil hingga %d: ",
number)
    if number%2 == 0 {
        number-- // Jika bilangan genap, kurangi 1 agar
dimulai dari bilangan ganjil terbesar
    }
    printOddNumbers(number)
    fmt.Println()
}
```

Screenshoot Output

```
o\Documents\file kuliah\Laprak Alpro 2\modul 5\unguided5.go"
Masukkan bilangan: 20
Barisan bilangan ganjil hingga 20: 1 3 5 7 9 11 13 15 17 19
PS C:\Users\Lenovo\Documents\file kuliah\Laprak Alpro 2\modul 5> |
```


Deskripsi Program

Program ini mencetak barisan bilangan ganjil secara menurun dari suatu bilangan yang diberikan oleh pengguna. Konsep rekursif digunakan dalam fungsi `printOddNumbers` untuk menyelesaikan tugas ini. Fungsi ini secara berulang memanggil dirinya sendiri dengan nilai `n` yang terus dikurangi 2. Pada setiap pemanggilan, fungsi mencetak bilangan ganjil `n` dan kemudian memanggil dirinya sendiri untuk mencetak bilangan ganjil yang lebih kecil. Proses rekursif ini berlanjut hingga nilai `n` menjadi kurang dari 1. Fungsi `main` berfungsi sebagai titik masuk program, meminta input bilangan dari pengguna, dan kemudian memanggil fungsi `printOddNumbers` untuk mencetak barisan bilangan ganjil. Jika bilangan yang dimasukkan genap, program akan menguranginya sebesar 1 agar dimulai dari bilangan ganjil terbesar yang kurang dari atau sama dengan bilangan yang dimasukkan.

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 power(base, exponent int) int {
    if exponent == 0 {
        return 1
    } else if exponent < 0 {
        return 1 / power(base, -exponent)
    } else {
        return base * power(base, exponent-1)
    }
}

func main() {
    var base, exponent int
    fmt.Print("Masukkan bilangan pokok: ")
}
```

```
    fmt.Scan(&base)
    fmt.Print("Masukkan pangkat: ")
    fmt.Scan(&exponent)

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

Screenshoot Output

```
o\Documents\file kuliah\Laprak Alpro 2\modul 5\unguided6.go"
Masukkan bilangan pokok: 5
Masukkan pangkat: 2
5 pangkat 2 adalah 25
PS C:\Users\Lenovo\Documents\file kuliah\Laprak Alpro 2\modul 5> |
```

Deskripsi Program

Program ini menghitung pangkat dari sebuah bilangan menggunakan rekursi. Fungsi power menerima dua parameter: bilangan pokok base dan pangkat exponent. Jika exponent adalah 0, fungsi mengembalikan 1. Jika exponent negatif, fungsi membalik tanda exponent dan membagi 1 dengan hasil rekursif. Jika exponent positif, fungsi mengalikan base dengan hasil rekursif dari base pangkat exponent-1. Fungsi main meminta input bilangan pokok dan pangkat dari pengguna, memanggil fungsi power untuk menghitung hasilnya, dan mencetak hasilnya ke layar.