

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 5  
REKURSIF**



**Universitas  
Telkom**

Oleh:

**HENDWI SAPUTRA**

2311102218

IF-11-02

**S1 TEKNIK INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO**

**2024**

## I. DASAR TEORI

### 5.1 Pengantar Rekursif

Pada modul-modul sebelumnya sudah dijelaskan bahwa suatu subprogram baik fungsi atau prosedur bisa memanggil subprogram lainnya. Hal ini tidak menutup RemungRinan bahwa subprogram yang dipanggil adalah dirinya sendiri. Dalam pemrograman teknik 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
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 diperhatiRan subprogram cetak() di atas, terlihat pada baris ke-4 terdapat pemanggilan subprogram cetak() kembali. Misalnya apabila kita eksekusi 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(5)
5 }
6 func cetak(x int){
7     fmt.Println(x)
8     cetak(x+1)
9 }
```

Oleh karena itu bisanya 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.

```

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

```

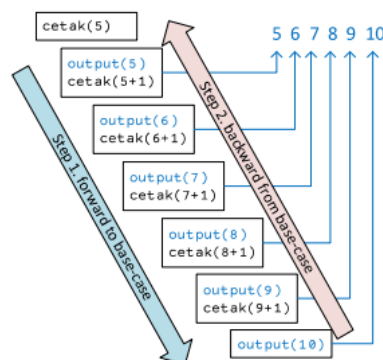
Apabila diperhatikan pada baris ke-3 di Program di atas, kita telah menambahkan **base-case** seperti penjelasan sebelumnya. Selanjutnya pada bagian aksi dari else di baris ke-6 dan ke-7 kita namakan **recursive-case** atau kasus pemanggilan dirinya sendiri tersebut terjadi. Kondisi dari **recursive-case**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

```

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

```

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. Ilustrasi 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(S).

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

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

**Catatan:**

- Teknik rekursif ini merupakan salah satu alternatif untuk mengganti struktur ontrol perulangan dengan memanfaatkan subprogram (bisa fungsi ataupun prosedure).
- Untuk menghentikan proses reursif digunakan percabangan (if-then).
- **Base-case** adalah kondisi proses rekursif berhenti. **Base-case** merupakan hal 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 Romplemen atau negasi dari **base-case**.
- Setiap algoritma rekursif selalu memiliki padanan dalam bentuk algoritma iteratif.

## 5.2 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.

## 5.3 Contoh Program dengan menggunakan Rekursif

- a. Membuat baris bilangan dari n hingga 1

Base-case: bilangan == 1

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

- 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  $2^n$

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  $n == 1$

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

## II. GUIDED

### GUIDED I

#### Source Code

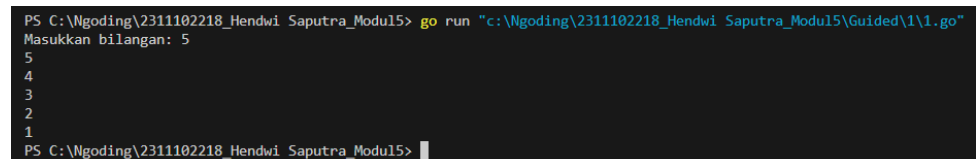
```
package main

import "fmt"

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

func baris(bilangan int) {
    if bilangan == 1 {
        fmt.Println(1)
    } else {
        fmt.Println(bilangan)
        baris(bilangan - 1)
    }
}
```

#### Screenshot



```
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul5\Guided\1\1.go"
Masukkan bilangan: 5
5
4
3
2
1
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5>
```

#### Deskripsi:

Program ini menggambarkan penggunaan rekursi sederhana yang mencetak deret bilangan dari nilai yang dimasukkan oleh pengguna hingga mencapai angka 1 secara menurun. Pengguna diminta untuk memasukkan sebuah bilangan bulat, kemudian program memanggil fungsi `baris` yang akan mencetak bilangan tersebut dan terus memanggil dirinya sendiri dengan mengurangi nilai bilangan satu per satu hingga mencapai angka 1. Fungsi `baris` menggunakan konsep rekursi untuk mencapai hal ini, di mana ia memanggil dirinya sendiri hingga mencapai kondisi dasar ketika bilangan sama dengan 1, yang kemudian dicetak.

## GUIDED II

### Source Code

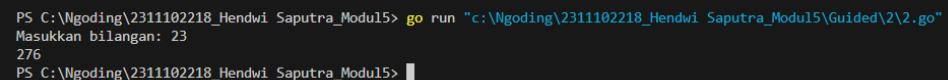
```
package main

import "fmt"

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

func penjumlahan(n int) int {
    if n == 1 {
        return 1
    } else {
        return n + penjumlahan(n-1)
    }
}
```

### Screenshot



```
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul5\Guided\2\2.go"
Masukkan bilangan: 23
276
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5> █
```

### Deskripsi:

Program ini menggambarkan penerapan rekursi untuk menghitung jumlah deret bilangan yang menghitung penjumlahan bilangan dari 1 hingga n menggunakan rekursi. Pengguna diminta untuk memasukkan sebuah bilangan bulat n. Program kemudian memanggil fungsi penjumlahan, yang merupakan fungsi rekursif. Fungsi ini menambahkan n dengan hasil dari pemanggilan dirinya sendiri dengan parameter n-1, hingga mencapai kondisi dasar ketika n sama dengan 1, yang kemudian dikembalikan sebagai hasil. Akhirnya, hasil penjumlahan tersebut dicetak ke layar.



### III. UNGUIDED

#### UNGUIDED I

##### Source Code

```
package main

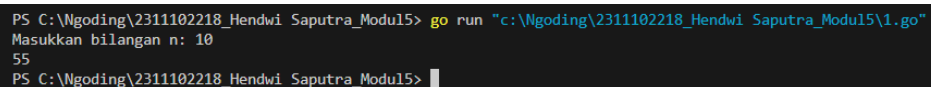
import "fmt"

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() {
    var n int

    fmt.Print("Masukkan bilangan n: ")
    fmt.Scan(&n)
    fmt.Print(fibonacci(n))
}
```

##### Screenshot



```
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul5\1.go"
Masukkan bilangan n: 10
55
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5> █
```

##### Deskripsi:

Program Ini adalah contoh sederhana dari penggunaan rekursi untuk menghitung deret Fibonacci yang menghitung nilai bilangan Fibonacci ke- $n$  menggunakan metode rekursi. Fungsi fibonacci menerima bilangan  $n$  sebagai parameter dan menggunakan kondisi dasar bahwa jika  $n$  kurang dari atau sama dengan 0, fungsi mengembalikan 0; jika  $n$  sama dengan 1, fungsi mengembalikan 1. Untuk nilai  $n$  yang lebih besar dari 1, fungsi memanggil dirinya sendiri dengan nilai  $n-1$  dan  $n-2$ , kemudian menjumlahkan hasilnya.

Dalam fungsi main, pengguna diminta untuk memasukkan bilangan n, dan hasil perhitungan nilai Fibonacci untuk n tersebut dicetak.

## UNGUIDED II

### Source Code

```
package main

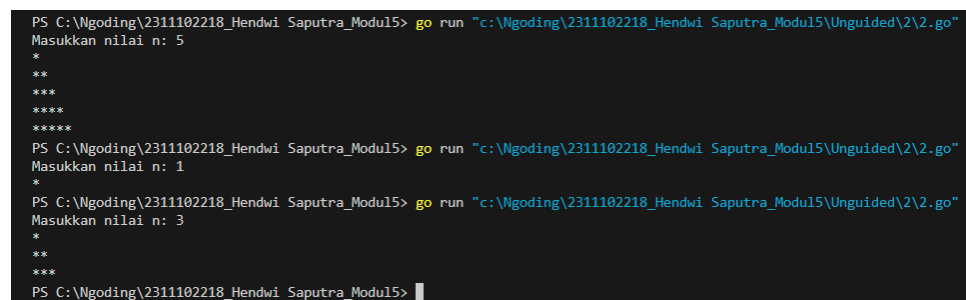
import "fmt"

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

func main() {
    var n int

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

### Screenshot



```
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul5\Unguided\2\2.go"
Masukkan nilai n: 5
*
**
***
****
*****
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul5\Unguided\2\2.go"
Masukkan nilai n: 1
*
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul5\Unguided\2\2.go"
Masukkan nilai n: 3
*
**
***
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5>
```

### Deskripsi:

Program ini adalah contoh rekursi yang mencetak pola bintang segitiga menurun. Fungsi cetakBintang menerima bilangan n sebagai parameter dan

mencetak baris dengan n bintang, menurun satu baris setiap kali fungsi dipanggil ulang dengan n-1. Kondisi dasar untuk fungsi ini adalah ketika n kurang dari atau sama dengan 0, di mana fungsi berhenti memanggil dirinya sendiri. Dalam fungsi main, pengguna diminta untuk memasukkan nilai n, dan fungsi cetakBintang dipanggil untuk mencetak pola bintang tersebut.

### UNGUIDED III

#### Source Code

```
package main

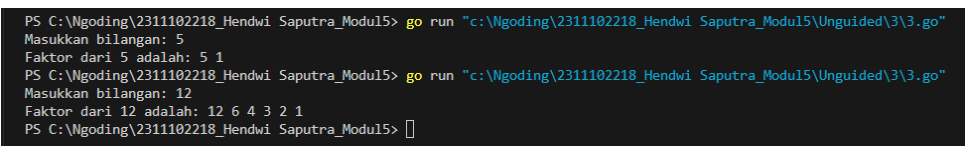
import "fmt"

func cariFaktor(bilangan, pembagi int) {
    if pembagi == 1 {
        fmt.Print(1)
        return
    }
    if pembagi == 1 {
        return
    }
    if bilangan%pembagi == 0 {
        fmt.Print(pembagi, " ")
    }
    cariFaktor(bilangan, pembagi-1)
}

func main() {
    var bilangan int

    fmt.Print("Masukkan bilangan: ")
    fmt.Scan(&bilangan)
    fmt.Print("Faktor dari ", bilangan, " adalah: ")
    cariFaktor(bilangan, bilangan)
    fmt.Println()
}
```

#### Screenshot



```
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul5\Unguided\3\3.go"
Masukkan bilangan: 5
Faktor dari 5 adalah: 5 1
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul5\Unguided\3\3.go"
Masukkan bilangan: 12
Faktor dari 12 adalah: 12 6 4 3 2 1
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5> █
```

## Deskripsi:

Program ini adalah mencari dan mencetak faktor-faktor dari sebuah bilangan yang dimasukkan oleh pengguna menggunakan metode rekursi. Program dimulai dengan meminta pengguna memasukkan sebuah bilangan. Fungsi cariFaktor kemudian memeriksa setiap bilangan dari nilai input hingga 1 untuk melihat apakah bilangan tersebut adalah faktor dari bilangan input. Jika ya, bilangan tersebut dicetak. Fungsi ini menggunakan rekursi untuk memanggil dirinya sendiri dengan mengurangi nilai pembagi satu per satu hingga mencapai 1, pada saat itu fungsi berhenti.

## UNGUIDED IV

### Source Code

```
package main

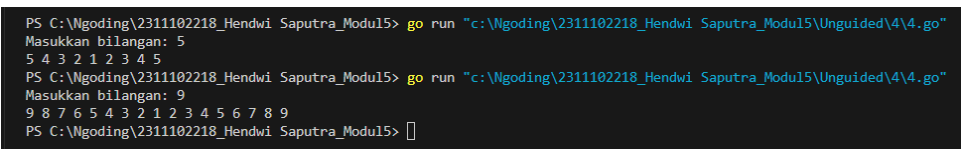
import "fmt"

func cetakBaris(n int) {
    if n > 1 {
        fmt.Print(n, " ")
        cetakBaris(n - 1)
        fmt.Print(n, " ")
    } else if n == 1 {
        fmt.Print(n, " ")
    }
}

func main() {
    var n int

    fmt.Print("Masukkan bilangan: ")
    fmt.Scan(&n)
    cetakBaris(n)
    fmt.Println()
}
```

### Screenshot



```
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul5\Unguided\4\4.go"
Masukkan bilangan: 5
5 4 3 2 1 2 3 4 5
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul5\Unguided\4\4.go"
Masukkan bilangan: 9
9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5> 
```

### Deskripsi:

Program ini menunjukkan cara menggunakan rekursi untuk mencetak bilangan dalam pola menurun dan naik dari n hingga 1 dan kembali naik ke n. Pengguna diminta untuk memasukkan sebuah bilangan n. Fungsi cetakBaris kemudian mencetak bilangan n, memanggil dirinya sendiri dengan nilai n-1, dan mencetak bilangan n lagi setelah rekursi selesai. Kondisi dasar fungsi adalah ketika n sama dengan 1, di mana fungsi hanya mencetak 1 dan berhenti memanggil dirinya sendiri.

### UNGUIDED V

#### Source Code

```
package main

import "fmt"

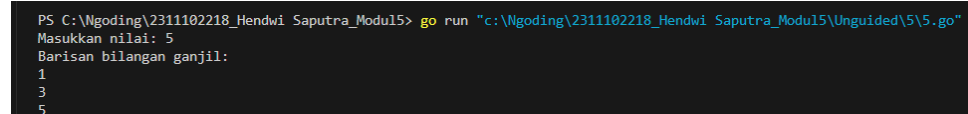
func cetak_BilanganGanjil(n int) {
    if n <= 0 {
        return
    }
    cetak_BilanganGanjil(n - 1)
    if n%2 != 0 {
        fmt.Println(n)
    }
}

func main() {
    var n int

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

    fmt.Println("Barisan bilangan ganjil: ")
    cetak_BilanganGanjil(n)
}
```

#### Screenshot



```
PS C:\Wgoding\2311102218_Hendwi Saputra_Modul5> go run "c:\Wgoding\2311102218_Hendwi Saputra_Modul5\Unguided\5\5.go"
Masukkan nilai: 5
Barisan bilangan ganjil:
1
3
5
```

```
PS C:\Ngoding\2311102218_Hendwi_Saputra_Modul5> go run "c:\Ngoding\2311102218_Hendwi_Saputra_Modul5\Unguided\5\5.go"
Masukkan nilai: 20
Barisan bilangan ganjil:
1
3
5
7
9
11
13
15
17
19
PS C:\Ngoding\2311102218_Hendwi_Saputra_Modul5> █
```

### Deskripsi:

Program ini menampilkan cara menggunakan rekursi untuk menghasilkan dan mencetak deret bilangan ganjil. Pengguna diminta untuk memasukkan nilai n. Fungsi cetak\_BilanganGanjil kemudian memanggil dirinya sendiri dengan nilai n-1 hingga mencapai 0. Setelah mencapai kondisi dasar, jika bilangan tersebut ganjil, nilai tersebut dicetak. Fungsi ini bekerja secara rekursif untuk memeriksa setiap bilangan dari 1 hingga n dan mencetak bilangan ganjil.

## UNGUIDED VI

### Source Code

```
package main

import "fmt"

func Perpangkatan(bil_dasar, pangkat int) int {
    if pangkat == 0 {
        return 1
    } else {
        return bil_dasar * Perpangkatan(bil_dasar,
pangkat-1)
    }
}

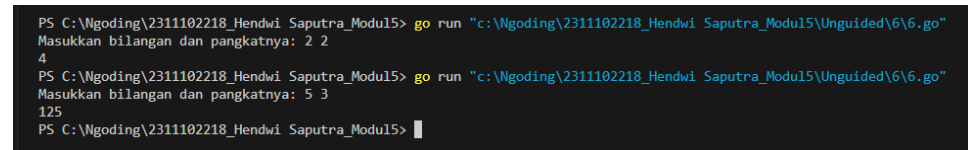
func main() {
    var bil_dasar, pangkat int

    fmt.Print("Masukkan bilangan dan pangkatnya: ")
    fmt.Scan(&bil_dasar, &pangkat)

    hasil := Perpangkatan(bil_dasar, pangkat)
    fmt.Print(hasil)

}
```

## Screenshot



```
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul5\Unguided\6\6.go"
Masukkan bilangan dan pangkatnya: 2 2
4
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul5\Unguided\6\6.go"
Masukkan bilangan dan pangkatnya: 5 3
125
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul5> █
```

## Deskripsi:

Program ini dapat menghitung hasil perpangkatan dari dua bilangan yang dimasukkan oleh pengguna menggunakan metode rekursi. Pengguna diminta untuk memasukkan bilangan dasar dan pangkatnya. Fungsi Perpangkatan menerima dua parameter tersebut dan mengembalikan hasil perkalian bilangan dasar dengan dirinya sendiri sebanyak pangkat kali, dengan menggunakan rekursi. Kondisi dasar fungsi ini adalah ketika pangkat sama dengan 0, yang akan mengembalikan nilai 1.