

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL VI
REKURSIF**



Disusun Oleh:
Bayu Kuncoro Adi / 2311102031
S1 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

a. Pengantar Rekursif

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 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 diperhatikan subprogram cetak(di atas, terlihat pada baris ke-4 terdapat pemanggilan subprogram cetak(kembali. Misalnya apabila kita eksekusi perintah cetak(5) maka akan menampilkan angka 5 6 7 8 9...dst tanpa henti. Artinya setiap pemanggilan subprogram cetak(nilai; + akan selalu bertambah (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 }
```

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

```
D:\DEV\DEMO>contoh.exe
```

```
5
6
7
8
9
10
11
12
13
...
```

Oleh karena itu bisanya ditambahkan struktur kontrol percabangan (ifthen) 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
```

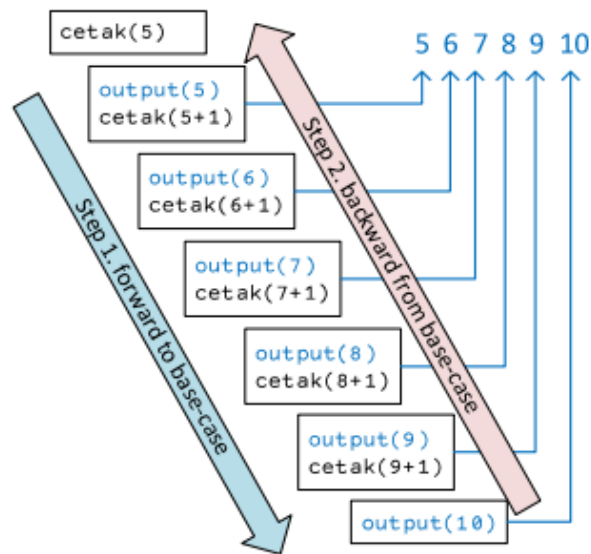
Apabila diperhatikan pada baris ke-3 di Program di atas, kita telah menambahkan base-case seperti penjelasan sebelumnya. Selanjutnya pada bagian aksi dari else di baris ke-6 dan ke-7 kita namakan recursive-case atau kasus pemanggilan dirinya sendiri tersebut terjadi. Kondisi dari recursive-case ini adalah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

```
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. 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(5).

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

```

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 }

```

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

D:\DEV\DEMO>contoh.exe

10
9
8
7
6
5

Catatan:

- Teknik rekursif ini merupakan salah satu alternatif untuk mengganti struktur kontrol perulangan dengan memanfaatkan subprogram (bisa fungsi ataupun prosedur).
- Untuk menghentikan proses rekursif 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 peranggilan dirinya sendiri dilakukan.
- Kondisi recursive-case adalah komplemen atau negasi dari base-case.
- Setiap algoritma rekursif selalu memiliki padanan dalam bentuk algoritma iteratif.

b. 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.

c. Contoh Program dengan menggunakan Rekursif

1. 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 }
```

2. Menhitung 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 }
```

3. 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 }
```

4. Mencari factorial atau $n!$

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

```
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

1. Guided 1

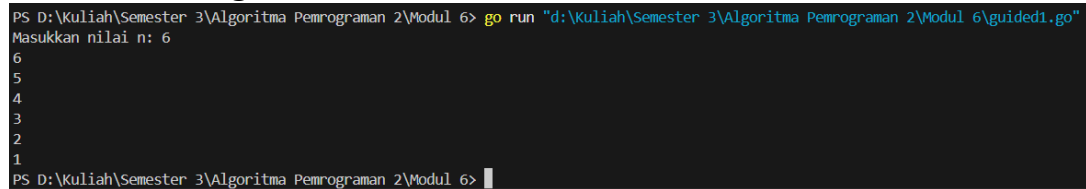
```
package main

import "fmt"

func main() {
    var n int
    fmt.Print("Masukkan nilai n: ")
    fmt.Scan(&n)
    if n > 0 {
        baris(n)
    } else {
        fmt.Println("Masukkan bilangan positif")
    }
}

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

Screenshoot Program



```
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6> go run "d:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6\guided1.go"
Masukkan nilai n: 6
6
5
4
3
2
1
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6> █
```

Deskripsi dan Cara Kerja Program

Program ini ditulis dalam bahasa Go dan bertujuan untuk menerima input bilangan bulat positif dari pengguna, kemudian mencetak urutan bilangan tersebut secara menurun hingga mencapai angka 1. Program dimulai di fungsi `main`, yang meminta pengguna memasukkan nilai `n`. Jika `n` adalah bilangan positif, maka fungsi `baris` akan dipanggil untuk mencetak urutan. Fungsi `baris` adalah fungsi rekursif yang memeriksa nilai dari `bilangan`. Jika `bilangan` sama dengan 1, program akan mencetak 1 dan mengakhiri rekursi. Jika `bilangan` lebih besar dari 1, program akan mencetak nilai saat ini dan memanggil dirinya sendiri dengan nilai `bilangan` dikurangi 1. Jika nilai `n` yang dimasukkan tidak positif, program akan menampilkan pesan untuk memasukkan bilangan positif.

2. Guided 2

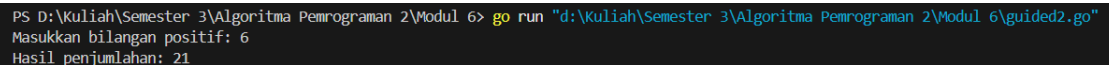
```
package main

import "fmt"

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

func main() {
    var n int
    fmt.Print("Masukkan bilangan positif: ")
    fmt.Scan(&n)
    if n > 0 {
        fmt.Println("Hasil penjumlahan:", penjumlahan(n))
    } else {
        fmt.Println("Masukkan bilangan positif")
    }
}
```

Screenshoot Program



```
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6> go run "d:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6\guided2.go"
Masukkan bilangan positif: 6
Hasil penjumlahan: 21
```

Deskripsi dan Cara Kerja Program

Program ini adalah program rekursif dalam bahasa Go yang menerima input bilangan bulat positif n dari pengguna dan menghitung jumlah total dari semua bilangan dari 1 hingga n . Di dalam fungsi `main`, program meminta pengguna untuk memasukkan bilangan n . Jika n adalah bilangan positif, maka program akan memanggil fungsi `penjumlahan` dengan parameter n . Fungsi `penjumlahan` adalah fungsi rekursif yang menambahkan n dengan hasil pemanggilan `penjumlahan(n-1)` sampai mencapai kondisi dasar, yaitu ketika n bernilai 1. Pada kondisi dasar ini, fungsi akan mengembalikan nilai 1 dan memulai proses penjumlahan dari seluruh pemanggilan rekursif. Hasil akhir dari penjumlahan akan ditampilkan di layar. Jika pengguna memasukkan bilangan yang tidak positif, program akan meminta pengguna untuk memasukkan bilangan positif.

3. Guided 3

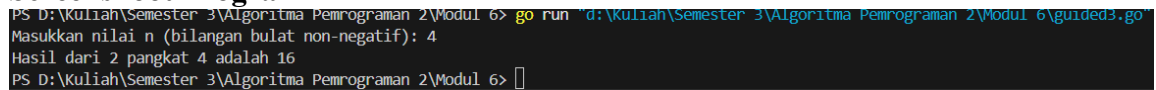
```
package main

import "fmt"

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

func main() {
    var n int
    fmt.Print("Masukkan nilai n (bilangan bulat non-negatif): ")
    fmt.Scan(&n)
    if n >= 0 {
        fmt.Println("Hasil dari 2 pangkat", n, "adalah",
pangkat(n))
    } else {
        fmt.Println("Masukkan bilangan bulat non-negatif")
    }
}
```

Screenshot Program



```
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6> go run "d:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6\guided3.go"
Masukkan nilai n (bilangan bulat non-negatif): 4
Hasil dari 2 pangkat 4 adalah 16
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6>
```

Deskripsi Program

Program ini bertujuan untuk menghitung nilai 2^n menggunakan fungsi rekursif dalam bahasa Go, di mana n adalah bilangan bulat non-negatif yang dimasukkan oleh pengguna. Program dimulai dengan meminta input n di fungsi main. Jika n adalah bilangan bulat non-negatif, program akan memanggil fungsi pangkat untuk menghitung hasil 2^n . Fungsi pangkat bekerja dengan prinsip rekursi, di mana kondisi dasarnya adalah ketika n sama dengan 0, maka fungsi akan mengembalikan nilai 1 (karena $2^0 = 1$). Jika n lebih besar dari 0, fungsi akan mengembalikan hasil dari 2 dikalikan dengan pemanggilan fungsi pangkat($n-1$). Hasil akhirnya adalah nilai 2^n yang dicetak di layar. Jika pengguna memasukkan nilai negatif, program akan menampilkan pesan untuk meminta bilangan non-negatif.

4. Guided 4

```
package main

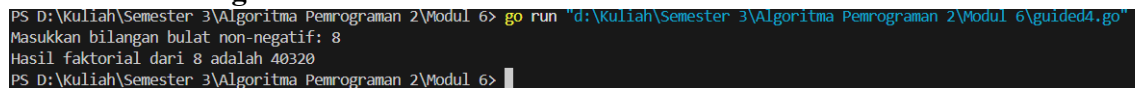
import "fmt"

// Fungsi rekursif untuk menghitung faktorial dari n
func faktorial(n int) int {
    if n == 0 || n == 1 {
        return 1
    } else {
        return n * faktorial(n-1)
    }
}

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

    if n >= 0 {
        fmt.Println("Hasil faktorial dari", n, "adalah",
faktorial(n))
    } else {
        fmt.Println("Masukkan bilangan bulat non-negatif")
    }
}
```

Screenshoot Program



```
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6> go run "d:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6\guided4.go"
Masukkan bilangan bulat non-negatif: 8
Hasil faktorial dari 8 adalah 40320
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6>
```

Deskripsi dan Cara Kerja Program

Program ini bertujuan untuk menghitung faktorial dari suatu bilangan bulat non-negatif n yang dimasukkan oleh pengguna, menggunakan fungsi rekursif dalam bahasa Go. Faktorial dari n didefinisikan sebagai hasil perkalian semua bilangan dari 1 hingga n , dengan kondisi dasar $0! = 10! = 10! = 1$ dan $1! = 11! = 11! = 1$. Pada fungsi main, program meminta pengguna memasukkan nilai n . Jika n adalah bilangan non-negatif, program akan memanggil fungsi faktorial. Fungsi faktorial ini bekerja secara rekursif, dengan kondisi dasar bahwa jika n adalah 0 atau 1, maka fungsi mengembalikan nilai 1. Jika n lebih dari 1, fungsi akan mengalikan n dengan hasil pemanggilan faktorial($n-1$). Hasil akhirnya adalah nilai faktorial dari n , yang kemudian dicetak di layar. Jika pengguna memasukkan nilai negatif, program akan memberikan pesan untuk memasukkan bilangan bulat non-negatif.

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 S 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

```
package main

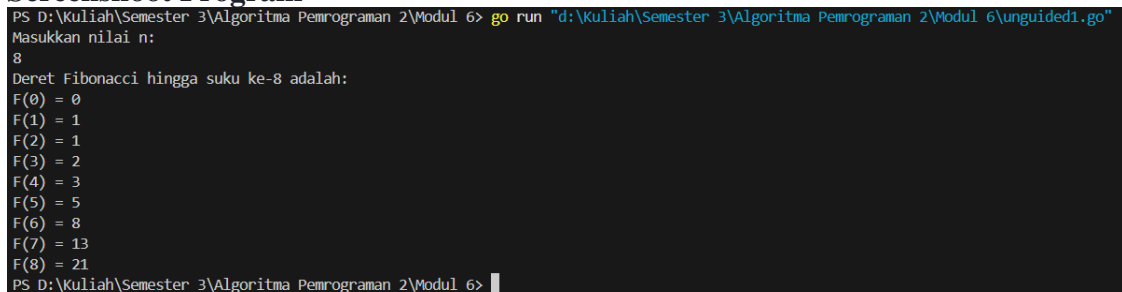
import (
    "fmt"
)

// 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() {
    var n int
    fmt.Println("Masukkan nilai n:")
    fmt.Scan(&n)

    fmt.Printf("Deret Fibonacci hingga suku ke-%d adalah:\n", n)
    for i := 0; i <= n; i++ {
        fmt.Printf("F(%d) = %d\n", i, fibonacci(i))
    }
}
```

Screenshoot Program



```
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6> go run "d:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6\unguided1.go"
Masukkan nilai n:
8
Deret Fibonacci hingga suku ke-8 adalah:
F(0) = 0
F(1) = 1
F(2) = 1
F(3) = 2
F(4) = 3
F(5) = 5
F(6) = 8
F(7) = 13
F(8) = 21
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6>
```

Deskripsi dan Cara Kerja Program

- Fungsi Rekursif fibonacci: Fungsi ini menghitung nilai deret Fibonacci untuk nilai nnn yang diberikan.
- Fungsi main: Fungsi utama yang akan dieksekusi saat program dijalankan. Fungsi ini membaca input dari pengguna, memanggil fungsi fibonacci, dan mencetak hasilnya.

Cara Kerja Program

- Baca Input: Program akan meminta pengguna untuk memasukkan nilai nnn.
 - Panggil Fungsi fibonacci: Fungsi rekursif ini akan dipanggil untuk menghitung nilai deret Fibonacci hingga suku ke-nnn.
 - Cetak Hasil: Hasil perhitungan akan dicetak ke layar.
2. Buatlah sebuah program yang digunakan untuk menampilkan pola Bintang berikut ini dengan menggunakan fungsi rekursif. N adalah masukan dari user.

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	* ** *** **** *****
2	1	*
3	3	* ** ***

```
package main

import (
    "fmt"
)

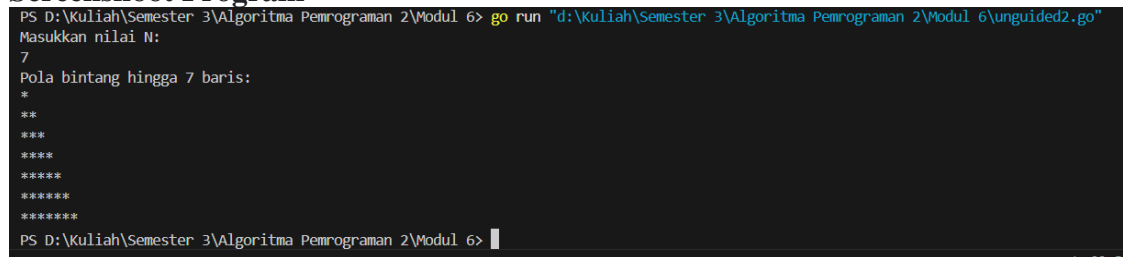
// Fungsi untuk mencetak sejumlah bintang
func printStars(n int) {
    if n > 0 {
        fmt.Print("*")
        printStars(n - 1)
    }
}
```

```
// Fungsi rekursif untuk mencetak pola bintang
func printPattern(n, current int) {
    if current <= n {
        printStars(current)
        fmt.Println()
        printPattern(n, current+1)
    }
}

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

    fmt.Printf("Pola bintang hingga %d baris:\n", n)
    printPattern(n, 1)
}
```

Screenshoot Program



```
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6> go run "d:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6\unguided2.go"
Masukkan nilai N:
7
Pola bintang hingga 7 baris:
*
**
***
****
*****
*****
*****
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6>
```

Deskripsi dan Cara Kerja Program

Program ini menggunakan dua fungsi utama: `printStars` dan `printPattern`. Fungsi `printStars` digunakan untuk mencetak sejumlah bintang sesuai dengan nilai yang diberikan. Fungsi `printPattern` adalah fungsi rekursif yang mencetak pola bintang mulai dari 1 bintang hingga NNN bintang secara bertahap. Fungsi ini memanggil dirinya sendiri dengan nilai `nnn` yang berkurang satu hingga mencapai 1. Fungsi utama `main` membaca input NNN dari pengguna dan memanggil `printPattern` untuk mencetak pola bintang.

Cara Kerja :

- **Baca Input:** Program meminta pengguna untuk memasukkan nilai NNN.
- **Cetak Pola Bintang:** Program memanggil fungsi rekursif `printPattern` yang mencetak bintang secara bertahap dari 1 hingga NNN.
- **Fungsi Rekursif:** Fungsi `printPattern` mencetak pola bintang dengan memanggil `printStars` untuk setiap tingkat dan memanggil dirinya sendiri dengan nilai yang berkurang satu hingga mencapai dasar rekursi.

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).

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	1 5
2	12	1 2 3 4 6 12

```
package main

import (
    "fmt"
)

// Fungsi rekursif untuk menemukan dan mencetak faktor-faktor dari N
func findFactors(N, current int) {
    if current > N {
        return
    }
    if N%current == 0 {
        fmt.Print(current, " ")
    }
    findFactors(N, current+1)
}

// Fungsi untuk memulai pencarian faktor dari 1 hingga N
func printFactors(N int) {
    findFactors(N, 1)
    fmt.Println()
}

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

    fmt.Printf("Faktor-faktor dari %d:\n", N)
    printFactors(N)
}
```

Screenshoot Program

```
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6> go run "d:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6\unguided1.go"
Masukkan nilai N:
8
Faktor-faktor dari 8:
1 2 4 8
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6>
```

Deskripsi dan Cara Kerja Program

Program ini menggunakan dua fungsi utama: `printFactors` dan `findFactors`. Fungsi `findFactors` adalah fungsi rekursif yang mencari dan mencetak faktor-faktor dari NNN. Fungsi `printFactors` memulai pencarian faktor dari angka 1 hingga NNN. Fungsi utama `main` membaca input NNN dari pengguna dan memanggil `printFactors` untuk memulai pencarian faktor-faktor.

Cara Kerja Program :

- **Baca Input:** Program meminta pengguna untuk memasukkan nilai NNN.
- **Cetak Faktor-faktor:** Program memanggil fungsi rekursif `findFactors` yang memeriksa setiap bilangan dari 1 hingga NNN apakah merupakan faktor dari NNN.
- **Fungsi Rekursif:** Fungsi `findFactors` memeriksa apakah nilai saat ini membagi NNN tanpa sisa. Jika ya, nilai tersebut dicetak sebagai faktor dan fungsi dipanggil kembali dengan nilai yang meningkat hingga NNN.

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.

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	5 4 3 2 1 2 3 4 5
2	9	9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9

```
package main

import (
    "fmt"
)

func displaySequence(n int) {
    if n == 1 {
        fmt.Print("1 ")
    } else {
        fmt.Printf("%d ", n)
        displaySequence(n - 1)
        fmt.Printf("%d ", n)
    }
}

func main() {
    var N int
    fmt.Print("Masukkan bilangan bulat positif: ")
    fmt.Scan(&N)
    displaySequence(N)
}
```

Screenshoot Program

```
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6> go run "d:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6\unguided4.go"
Masukkan bilangan bulat positif: 6
6 5 4 3 2 1 2 3 4 5 6
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6>
```

Deskripsi dan Cara Kerja Program

Program ini menampilkan deretan angka dari NNN hingga 1 dan kembali ke NNN menggunakan pendekatan rekursif. Fungsi `displaySequence` menerima satu parameter integer `n`. Jika `n` sama dengan 1, fungsi akan mencetak "1" dan selesai (basis rekursif). Jika `n` lebih besar dari 1, fungsi akan mencetak nilai `n`, kemudian memanggil dirinya sendiri dengan `n - 1`, dan setelah kembali dari pemanggilan rekursif, mencetak nilai `n` lagi. Pola ini membuat angka turun dari `N` ke 1, dan kembali naik ke `N`. Program utama (main) meminta input dari pengguna dan memanggil fungsi `displaySequence` untuk menampilkan hasil sesuai pola.

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`.

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	1 3 5
2	20	1 3 5 7 9 11 13 15 17 19

```
package main

import (
    "fmt"
)

func displayOddSequence(n, current int) {
    if current > n {
        return
    }
    fmt.Printf("%d ", current)
    displayOddSequence(n, current+2)
}

func main() {
    var N int
    fmt.Print("Masukkan bilangan bulat positif: ")
    fmt.Scan(&N)
    displayOddSequence(N, 1)
}
```

Screenshoot Program

```
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6> go run "d:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6\unguided5.go"
Masukkan bilangan bulat positif: 4
1 3
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6>
```


Deskripsi dan Cara Kerja Program

Program ini menampilkan deretan bilangan ganjil dari 1 hingga NNN menggunakan pendekatan rekursif. Fungsi `displayOddSequence` menerima dua parameter: `n` sebagai batas atas, dan `current` sebagai angka ganjil saat ini yang dimulai dari 1. Jika `current` melebihi `n`, fungsi berhenti (basis rekursif). Jika tidak, fungsi mencetak nilai `current`, kemudian memanggil dirinya sendiri dengan nilai `current` ditambah 2 untuk mendapatkan bilangan ganjil berikutnya. Program utama (main) meminta input dari pengguna dan memanggil `displayOddSequence` untuk menampilkan deretan bilangan ganjil dari 1 hingga NNN.

6. Buatlah program golang 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"`

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	2 2	4
2	5 3	125

```
package main

import "fmt"

func power(x, y int) int {
    // Basis case: jika pangkat 0, return 1
    if y == 0 {
        return 1
    }
    // Basis case: jika pangkat 1, return x
    if y == 1 {
        return x
    }
    // Basis case: jika pangkat negatif, return 0
    if y < 0 {
        return 0
    }

    // Recursive case: pangkat genap
    if y%2 == 0 {
        temp := power(x, y/2)
        return temp * temp
    }
    // Recursive case: pangkat ganjil
    return x * power(x, y-1)
}
```

```

}

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

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

```

Screenshoot Program

```

PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6> go run "d:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6\unguided6.go"
Masukkan bilangan x: 6
Masukkan pangkat y: 5
Hasil 6 pangkat 5 adalah: 7776
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 6>

```

Deskripsi dan Cara Kerja Program

Program ini mengimplementasikan fungsi pangkat secara rekursif dengan pendekatan "divide and conquer". Cara kerjanya adalah dengan membagi kasus menjadi beberapa kondisi: jika pangkat 0 return 1 (basis case), jika pangkat 1 return x (basis case), jika pangkat negatif return 0 (penanganan error), untuk pangkat genap program akan membagi pangkat menjadi dua bagian yang sama ($y/2$) lalu mengkuadratkan hasilnya, dan untuk pangkat ganjil program akan mengurangi pangkat sebesar 1 lalu mengalikan dengan x. Pendekatan ini membuat perhitungan menjadi lebih efisien dibandingkan dengan melakukan perkalian berulang sebanyak y kali. Program ini sesuai dengan contoh masukan dan keluaran yang diberikan. Misalnya untuk input $x=2$, $y=2$ akan menghasilkan output 4, dan untuk input $x=5$, $y=3$ akan menghasilkan output 125.

KESIMPULAN

Berdasarkan data laporan praktikum di atas, dapat disimpulkan bahwa rekursif merupakan teknik pemrograman di mana sebuah fungsi atau prosedur dapat memanggil dirinya sendiri untuk menyelesaikan suatu masalah dengan cara memecahnya menjadi sub-masalah yang identik. Konsep utama dalam rekursif terdiri dari dua komponen penting yaitu base-case (kondisi dasar untuk menghentikan rekursi) dan recursive-case (kondisi di mana fungsi memanggil dirinya sendiri). Base-case menjadi komponen terpenting karena tanpa kondisi penghentian yang tepat, rekursi akan berjalan tanpa henti.

Melalui praktikum yang dilakukan, telah diimplementasikan berbagai program rekursif dalam bahasa Go untuk menyelesaikan beragam permasalahan seperti menghitung deret Fibonacci, membuat pola bintang, mencari faktor bilangan, menampilkan barisan bilangan tertentu, dan menghitung perpangkatan. Program-program tersebut menunjukkan bahwa rekursif dapat menjadi alternatif yang efektif untuk menggantikan struktur perulangan iteratif, meskipun setiap algoritma rekursif selalu memiliki padanan dalam bentuk algoritma iteratif. Penggunaan rekursif sering kali membuat kode menjadi lebih elegant dan mudah dipahami, terutama untuk permasalahan yang secara natural memiliki struktur rekursif.

DAFTAR PUSTAKA

Modul 6 Praktikum Algoritma dan Pemrograman 2 Modul Rekursif