

**LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
MODUL V
REKURSIF**



Oleh:

NAMA : AHMAD TITANA NANDA PRAMUDYA

NIM : 2311102042

KELAS : IF 11 02

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

I. DASAR TEORI

Pengantar Rekursif

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 (ln 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 x akan selalu P bertambah 1 (Increment by one) secara terus menerus tanpa henti.

```
package main

import "fmt"

func main(){
    cetak (5)
}

func cetak (x int){
    fmt.Println(x)
}

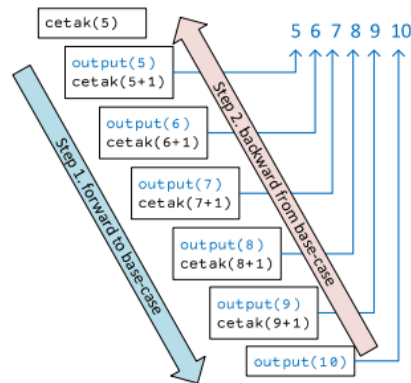
cetak (x+1)
```

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.

```
procedure cetak (in x:integer)
  algoritma
    if x ==10 then
      output(x)
    else
      output(x)
      cetak(x+1)
    endif
  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 $x \neq 10$.

```
package main
import "fmt"
func main(){
  cetak (5)
}
func cetak (x int){
  if x== 10 {
    fmt.Println(x)
  }else{
    fmt.Println(x)
    cetak (x+1)
  }
}
```



Gambar ini 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).

Catatan:

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

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

II. GUIDED

No 1.

Source code:

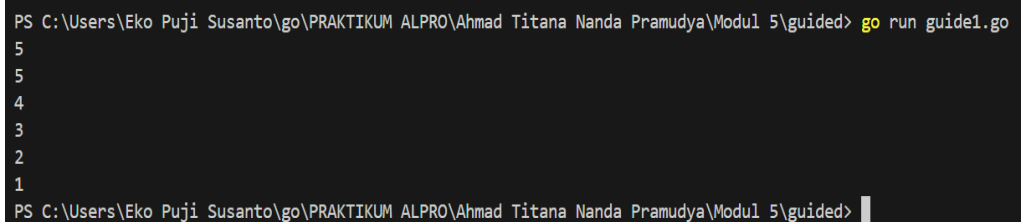
```
package main

import "fmt"

func main(){
    var n int
    fmt.Scan(&n)
    baris(n)
}

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

Output :



```
PS C:\Users\Eko Puji Susanto\go\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\guided> go run guide1.go
5
4
3
2
1
PS C:\Users\Eko Puji Susanto\go\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\guided> |
```

Penjelasan :

program ini menggunakan rekursi untuk mencetak angka dari nilai input n hingga 1 secara menurun

NO.2

Sourcode :

```
package main

import "fmt"

func main (){
    var n int
    fmt.Scan(&n)
    fmt.Println(perjumlahan(n))
}

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

Output :

```
PS C:\Users\Eko Puji Susanto\go\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\guided> go run guided2.go
10
55
PS C:\Users\Eko Puji Susanto\go\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\guided> |
```

Penjelasan :

program ini menghitung total penjumlahan dari 1 hingga n menggunakan pendekatan rekursif, dengan setiap langkah rekursi mengurangi nilai n hingga mencapai 1.

III. UNGUIDE

NO.1

Source code :

```
package main

import (
    "fmt"
)

func fibonacci(n int) int {
    if n == 0 {
        return 0
    } else if n == 1 {
        return 1
    }
    return fibonacci(n-1) + fibonacci(n-2)
}

func main() {

    fmt.Print("n : ")
    for i := 0; i <= 10; i++ {
        fmt.Printf("%d ", i)
    }
    fmt.Println() // Baris baru

    fmt.Print("Sn : ")
    for i := 0; i <= 10; i++ {
        fmt.Printf("%d ", fibonacci(i))
    }
    fmt.Println() // Baris baru setelah Sn
}
```


Output :

```
PS D:\titan\titan 2\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\Unguided> go run unguided1.go
n : 0 1 2 3 4 5 6 7 8 9 10
Sn : 0 1 1 2 3 5 8 13 21 34 55
PS D:\titan\titan 2\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\Unguided>
```

Penjelasan :

rogram ini menghitung dan menampilkan deret Fibonacci hingga suku ke-10 menggunakan fungsi rekursif. Ouput menunjukkan nilai indeks n dan hasil Fibonacci Sn secara Horizontal. Meskipun pendekatan rekursif ini mudah dipahami, ia kurang efisien untuk nilai n besar karena menghasilkan banyak pemanggilan fungsi berulang. Alternatif yang lebih efisien, seperti iterasi atau memoization, lebih cocok untuk perhitungan Fibonacci dalam skala besar.

NO.2

Source code :

```
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

    for {
        fmt.Print("Masukkan angka : ")
        fmt.Scan(&N)

        if N == 0 {
            fmt.Println("Program selesai.")
            break
        }
    }
}
```

```

    }
    printStars(N)
  }
}

```

Output :

```

PS D:\titan\titan 2\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\Unguided> go run unguided2.go
Masukkan angka : 5
*
**
***
****
*****
Masukkan angka : 1
*
Masukkan angka : 3
*
**
***

```

Penjelasan :

Program ini menunjukkan bagaimana menggunakan rekursi untuk menghasilkan pola bintang berdasarkan input pengguna. Dengan desain interaktif, pengguna dapat dengan mudah memasukkan angka baru untuk melihat hasilnya, dan program berhenti dengan elegan ketika pengguna memasukkan 0. Ini membuat program tidak hanya fungsional, tetapi juga user-friendly

NO.3

Source code :

```

package main

import (
    "fmt"
)

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

func main() {
    var N int

```

```

    fmt.Print("Masukkan angka N: ")
    fmt.Scan(&N)

    fmt.Print("Faktor dari", N, ": ")
    findFactors(N, 1)
    fmt.Println()
}

```

Output :

```

PS D:\titan\titan 2\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\Unguided> go run unguided3.go
Masukkan angka N: 5
Faktor dari 5: 1 5
PS D:\titan\titan 2\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\Unguided> go run unguided.go
CreateFile unguided.go: The system cannot find the file specified.
PS D:\titan\titan 2\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\Unguided> go run unguided3.go
Masukkan angka N: 12
Faktor dari 12: 1 2 3 4 6 12
PS D:\titan\titan 2\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\Unguided>

```

Penjelasan :

Program ini menggunakan fungsi rekursif `findFactors` untuk menghitung dan menampilkan semua faktor dari bilangan bulat positif `N`. Faktor-faktor ditentukan dengan memeriksa setiap angka dari 1 hingga `N` dan mencetak angka tersebut jika merupakan pembagi yang tepat dari `N`.

NO.4

Source code :

```

package main
import (
    "fmt"
)
func printSequence(n, current int) {
    if current > n {
        return
    }
    fmt.Print(current, " ")
    printSequence(n, current+1)
    if current < n {
        fmt.Print(current, " ")
    }
}

func main() {
    var N int
    fmt.Print("Masukkan angka: ")

```

```
    fmt.Scan(&N)
    printSequence(N, 1)
    fmt.Println()
}
```

Output :

```
PS D:\titan\titan 2\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\Unguided> go run unguided4.go
Masukkan angka: 5
1 2 3 4 5 4 3 2 1
PS D:\titan\titan 2\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\Unguided> go run unguided4.go
Masukkan angka: 9
1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1
PS D:\titan\titan 2\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\Unguided> █
```

Penjelasan :

Program ini menggunakan fungsi rekursif `printSequence` untuk mencetak urutan angka dari 1 hingga N, dan kemudian kembali ke 1. Dengan pendekatan rekursif, program memastikan setiap angka dicetak dengan urutan yang benar.

NO.5

Source code :

```
package main

import "fmt"
func tampilGanjil(n int, current int) {
    if current > n {
        return
    }
    fmt.Print(current, " ")
    tampilGanjil(n, current+2)
}

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

Output :

```
PS D:\titan\titan 2\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\Unguided> go run unguided5.go
Masukkan bilangan bulat positif N: 5
1 3 5
PS D:\titan\titan 2\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\Unguided> go run unguided5.go
Masukkan bilangan bulat positif N: 20
1 3 5 7 9 11 13 15 17 19
PS D:\titan\titan 2\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\Unguided> |
```

Penjelasan :

Program ini menggunakan pendekatan rekursif untuk mencetak deret bilangan ganjil dari 1 hingga N. Setiap kali fungsi rekursif dipanggil, ia menambah nilai bilangan ganjil sebesar 2, dimulai dari 1, dan berhenti ketika bilangan ganjil yang dicetak melebihi N. Pendekatan ini efektif untuk masalah yang membutuhkan proses yang berulang dengan struktur sederhana

NO.6

Source code :

```
package main

import "fmt"

func pangkat(x int, y int) int {
    if y == 0 {
        return 1
    }
    return x * pangkat(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)
    hasil := pangkat(x, y)
    fmt.Printf("Hasil dari %d pangkat %d adalah %d\n", x, y, hasil)
}
```

Output :

```
PS D:\titan\titan 2\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\Unguided> go run unguided6.go
Masukkan nilai x: 2
Masukkan nilai y: 2
Hasil dari 2 pangkat 2 adalah 4
PS D:\titan\titan 2\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\Unguided> go run unguided6.go
Masukkan nilai x: 5
Masukkan nilai y: 3
Hasil dari 5 pangkat 3 adalah 125
PS D:\titan\titan 2\PRAKTIKUM ALPRO\Ahmad Titana Nanda Pramudya\Modul 5\Unguided> █
```

Penjelasan :

Program ini menghitung hasil pangkat menggunakan fungsi rekursif. Fungsi pangkat bekerja dengan mengalikan angka dasar x berulang kali, sambil mengurangi eksponen y hingga mencapai nol. Ketika eksponen mencapai nol, nilai pangkat menjadi 1, yang merupakan syarat dasar dari pangkat.