

**LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK**

**MODUL 5
REKRUSIF**



Oleh:

M. AZKA HERMAWAN

2311102230

IF – 11- 02

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2024

I. DASAR TEORI

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 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	algorithm	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 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

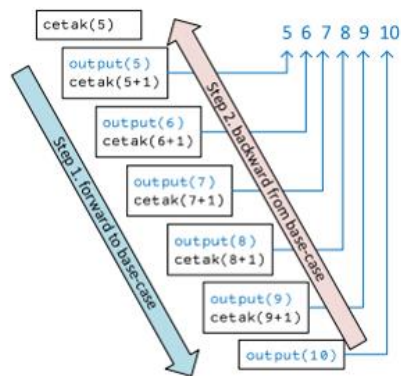
dengan **base-case**, artinya apabila kondisi base-case bernilai true maka proses rekursif akan berhenti. Sebagai contoh misalnya base case adalah ketika x bernilai 10 atau $x == 10$, maka tidak perlu dilakukan rekursif.

```
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 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 pemanggilan dirinya sendiri dilakukan. Kondisi **recursive-case** adalah komplement atau negasi dari **base-case**.
- Setiap algoritma rekursif selalu memiliki padanan dalam bentuk algoritma iteratif.

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.

II. GUIDED

1. Guided 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)
    }
}
```

Screenshot



```
7
6
5
4
3
2
1
```

Deskripsi

Program diatas merupakan program yang meminta user untuk menginputkan sebuah bilangan “n” kemudian program akan mencetak bilangan dari ‘n’ hingga ke-1 menggunakan fungsi rekursif.

Fungsi dari rekursi pada program diatas yaitu untuk menyederhanakan alur untuk pemanggilan fungsi secara berulang dengan mengurangi nilai bilangan sampai mencapai kondisi pemberhentian.

2. Guided 2

Source Code

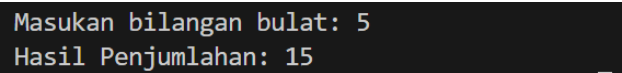
```
package main

import "fmt"

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

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

Screenshot



```
Masukan bilangan bulat: 5
Hasil Penjumlahan: 15
```

Deskripsi

Program diatas menggunakan rekursi untuk menghitung penjumlahan bilangan dari 1 hingga n.

III. UNGUIDED

1. Unguided 1

Source Code

```
package main

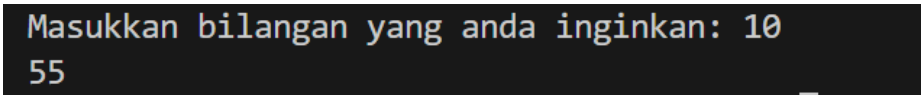
import "fmt"

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 yang anda inginkan: ")
    fmt.Scan(&n)
    fmt.Print(fibonacci(n))
}
```

Screenshot



```
Masukkan bilangan yang anda inginkan: 10
55
```

Deskripsi

Program ini merupakan program untuk meminta pengguna untuk memasukkan bilangan n, Setelah itu, program akan menghitung bilangan Fibonacci ke-n.

Program menggunakan 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.

2. Unguided 2

Source Code

```
package main

import "fmt"

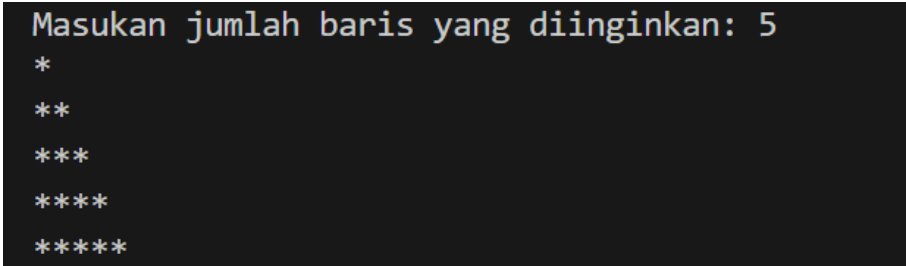
func main() {
    var N int
    fmt.Print("Masukan jumlah baris yang diinginkan: ")
    fmt.Scan(&N)
    pola(N, 1)
}

func cetakBintang(n, i int) {
    if i == 0 {
        return
    }

    fmt.Print("*")
    cetakBintang(n, i-1)
}

func pola(n, baris int) {
    if baris > n {
        return
    }
    cetakBintang(n, baris)
    fmt.Println()
    pola(n, baris+1)
}
```

Screenshot



```
Masukan jumlah baris yang diinginkan: 5
*
**
***
****
*****
```

Deskripsi

Program ini merupakan program untuk menampilkan baris dengan pola bintang sesuai dengan yang diinputkan oleh user.

Pada program ini, rekursi berfungsi untuk mencetak bintang dalam satu baris dan untuk mencetak setiap baris secara berurutan.

3. Unguided 3

Source Code

```
package main

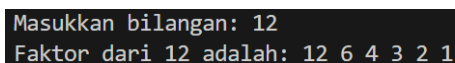
import "fmt"

func main() {
    var bilangan int

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

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

Screenshot

A screenshot of a terminal window showing the output of the Go program. The first line is "Masukkan bilangan: 12" and the second line is "Faktor dari 12 adalah: 12 6 4 3 2 1".

```
Masukkan bilangan: 12
Faktor dari 12 adalah: 12 6 4 3 2 1
```

Deskripsi

Program ini merupakan program untuk mencari faktor dari suatu bilangan, yaitu bilangan apa saja yang habis dibagi oleh n(input pengguna).

Program ini menggunakan fungsi rekursif untuk menghitung sebuah faktor untuk 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 memanggil dirinya sendiri dengan mengurangi nilai pembagi satu per satu hingga mencapai 1, pada saat itu fungsi berhenti.

4. Unguided 4

Source Code

```
package main

import "fmt"

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

func barisNaik(n, awal int) {
    if n > awal {
        return
    }
    fmt.Print(n, " ")
    barisNaik(n+1, awal)
}


func main() {
    var o int
    fmt.Print("Masukan bilangan bulat positif: ")
    fmt.Scan(&o)

    barisTurun(o)

    if o >= 1 {
        barisNaik(2, o)
    }

    fmt.Println()
}
```

Screenshot



Masukan bilangan bulat positif: 5
5 4 3 2 1 2 3 4 5

Deskripsi

Program berikut merupakan program yang meminta user untuk memasukkan bilangan bulat positif.

Pada fungsi “barisTurun” menggunakan rekursif yang akan menampilkan angka dari $n(\text{input})$ hingga ke-1.

Pada fungsi “barisNaik” menggunakan rekursif yang mencetak angka dari n hingga mencapai nilai awal.

5. Unguided 5

Source Code

```
package main

import "fmt"

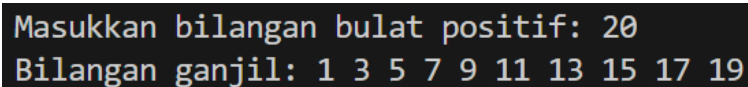
func main() {
    var o int
    fmt.Print("Masukkan bilangan bulat positif: ")
    fmt.Scan(&o)

    fmt.Print("Bilangan ganjil: ")
    bilanganGanjil(o)
    fmt.Print()
}

func bilanganGanjil(n int) {
    if n <= 0 {
        return
    }
    bilanganGanjil(n - 1)

    if n%2 != 0 {
        fmt.Print(n, " ")
    }
}
```

Screenshot

A screenshot of a terminal window showing the output of the Go program. The first line is the prompt "Masukkan bilangan bulat positif: 20" and the second line is the output "Bilangan ganjil: 1 3 5 7 9 11 13 15 17 19".

```
Masukkan bilangan bulat positif: 20
Bilangan ganjil: 1 3 5 7 9 11 13 15 17 19
```

Deskripsi

Program ini meminta user meng-inputkan sebuah bilangan, kemudian program akan meng-outputkan bilangan ganjil saja.

Program ini menggunakan fungsi rekursif yang digunakan untuk menghitung dan mencetak bilangan ganjil dari 1 hingga n, dengan memanggil dirinya sendiri untuk setiap bilangan yang lebih kecil hingga mencapai 0.

6. Unguided 6

Source Code

```
package main

import "fmt"

func pangkat(x, y int) int {
    if y == 0 {
        return 1
    }
    return x * pangkat(x, y-1)
}

func main() {
    var x, y int

    fmt.Print("Masukkan bilangan x: ")
    fmt.Scan(&x)

    fmt.Print("Masukkan bilangan y: ")
    fmt.Scan(&y)

    hasil := pangkat(x, y)
    fmt.Printf("%d dipangkatkan %d adalah %d/n", x, y,
    hasil)
}
```

Screenshot

```
Masukkan bilangan x: 2
Masukkan bilangan y: 2
2 dipangkatkan 2 adalah 4/n
```

Deskripsi

Program ini merupakan program untuk menghitung 2 bilangan yang telah diinputkan oleh user dengan menggunakan metode rekursi.

Rekursi berfungsi untuk menghitung x dipangkatkan y dengan cara mengalikan x dengan hasil pemangkatan dari x dan $y-1$.