

**LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2**

**MODUL 5
REKURSIF**



Oleh:

PANDIA ARYA BRATA

2311102076

IF – 11 - 02

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

2024

I. DASAR TEORI

6.1 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 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 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 }
```

```
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 Control 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.

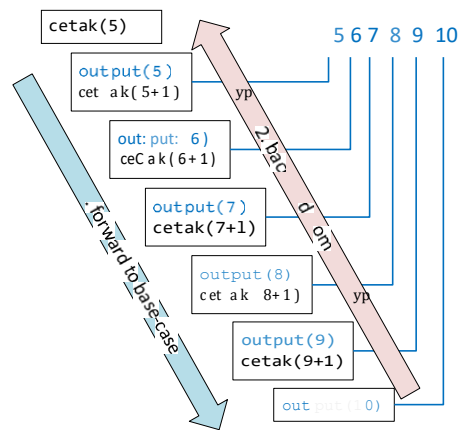
```
1 procedure cetak(in x:integer)
2   algoritma
3     if x ==10 then
4       output (x)
5     else
6       output (x)
7       cetak(x+1 )
8     endif
9 endprocedure
```

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 }
```

```
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. //ustrasi proses forward dan backward pada soot re#ursi/.

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 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 }

```

```

D:\DEV\DEMO>go build
contoh.go
D:\DEV\DEMO>contoh.exe
10
9
8
7
6

```

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 pedoman yang harus diketahui oleh programmer untuk 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.

6.2 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. Sourcecode :

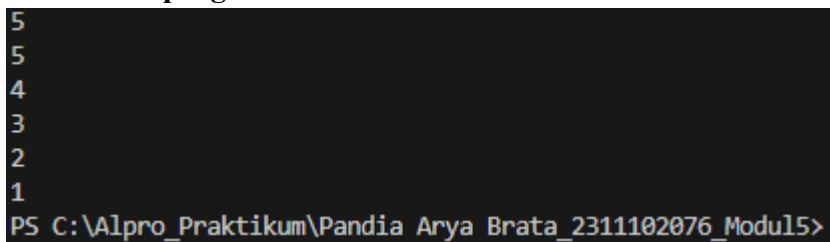
```
package main

import "fmt"

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

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

Screenshot program :



```
5
5
4
3
2
1
PS C:\Alpro_Praktikum\Pandia Arya Brata_2311102076_Modul5>
```

Deskripsi program :

Program tersebut meminta user menginputkan bilangan bulat pada variable n , dan program akan mengeksekusi nilai n dengan fungsi baris (n), Dimana fungsi ini memiliki kondisi jika n(bilangan) == 1 maka print 1 tetapi jika lain dr 1 maka program akan melakukan print dan memanggil Kembali fungsi baris (bilangan -1) Dimana ini merupakan REKURSIF.

2. Sourcecode :

```
package main

import "fmt"

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

func penjumlahan(n int) int {
```

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

Screenshot program :

Deskripsi program :

Program ini menghitung dan mencetak seluruh jumlah inputan dari n bilangan bulat, dan terdapat rekursif yang melakukan penambahan secara berulang dengan kondisi “n + penjumlahan (n-1)” dan akan berhenti jika nilai $n == 1$.

III. UNGUIDED

1. Sourcecode :

```
package main

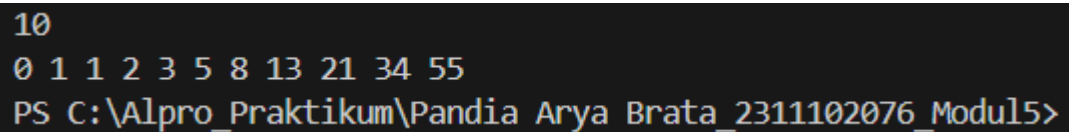
import "fmt"

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

    for i := 0; i <= n; i++ {
        fmt.Printf("%d ", fibonacci(i))
    }
    fmt.Println()
}

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

Screenshot program :



```
10
0 1 1 2 3 5 8 13 21 34 55
PS C:\Alpro_Praktikum\Pandia Arya Brata_2311102076_Modul5>
```

Deskripsi program :

Program rekursif yang memiliki fungsi Fibonacci yang mana program menampilkan dari suku ke-0, dalam fungsi Fibonacci memiliki kondisi Dimana jika $n \leq 1$ maka return n dan di outputkan oleh perulangan 0, i++ jadi $i:=1$, dan $i==n$, maka perulangan mengoutputkan 1. Dan kondisi lain $Fibonacci(n-1) + Fibonacci(n-2)$ semisal input adalah 3 maka $3-1 =$ suku ke-2 + $3-2 =$ suku ke-1, suku ke 1 = 1 dan suku ke 2 = 1 maka hasilnya adalah 2.

2. Sourcecode :

```
package main

import "fmt"

func main() {
    var tinggi int
    fmt.Scan(&tinggi)
    segitiga(tinggi, 1)
}

func segitiga(n, i int) {
    if i > n {
        return
    } else {
        for j := 0; j < i; j++ {
```



```

        fmt.Print("*")
    }
    fmt.Println()
    segitiga(n, i+1)
}
}

```

Screenshot program :

```

5
*
**
***
****
*****
PS C:\Alpro_Praktikum\Pandia Arya Brata_2311102076_Modul5>
1
*
PS C:\Alpro_Praktikum\Pandia Arya Brata_2311102076_Modul5>
3
*
**
***
PS C:\Alpro_Praktikum\Pandia Arya Brata_2311102076_Modul5>

```

Deskripsi program :

Program Bintang yang menggunakan rekursif, dengan user diminta mengisi nilai var tinggi bilangan bulat, dan terdapat prosedur segitiga yang berisi rekursif dan perulangan dengan perulangan untuk mencetak bintang,

3. Sourcecode :

```

package main

import "fmt"

func main() {
    var bilangan int
    fmt.Scan(&bilangan)

    fmt.Println("Faktor-faktor dari", bilangan, " adalah ")
    faktor(bilangan, 1)
}

func faktor(n, f int) {
    if f > n {
        return
    }
    if n%f == 0 {
        fmt.Println(f)
    }
    faktor(n, f+1)
}

```

Screenshot program :

```
PS C:\Alpro_Praktikum\Pandia Arya Brata_2311102076_Modul5>
5
Faktor-faktor dari 5 adalah
1
5
PS C:\Alpro_Praktikum\Pandia Arya Brata_2311102076_Modul5>
12
Faktor-faktor dari 12 adalah
1
2
3
4
6
12
```

Deskripsi program :

Program rekursif factorial yang akan menampilkan factor-faktor bilangan yang di inputkan oleh user, dengan kondisi faktor :=1 jika kondisi $f > \text{bilangan}/n$ maka program terhenti, tetapi selainnya jika n dibagi faktor == 0 atau tanpa sisa maka cetak faktor.

4. Sourcecode :

```
package main

import "fmt"

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

    fmt.Println(BilBalik(n))
}

func BilBalik(n int) string {
    if n == 1 {
        return "1"
    }
    return fmt.Sprintf("%d %s %d", n, BilBalik(n-1), n)
}
```

Screenshot program :

```
5
5 4 3 2 1 2 3 4 5
PS C:\Alpro_Praktikum\Pandia Arya Brata_2311102076_Modul5>
9
9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9
PS C:\Alpro_Praktikum\Pandia Arya Brata_2311102076_Modul5>
```

Deskripsi program :

Program rekursif ini memiliki fungsi menampilkan baris bilangan yang di input user, dengan fungsi yang memiliki rekursif dengan kondisi apabila $n == 1$ maka return 1,

kondisi lain yaitu cetak bilangan n , di panggil kembail “bilbalik(n-1)”, dan kembali cetak n.

5. Sourcecode :

```
package main

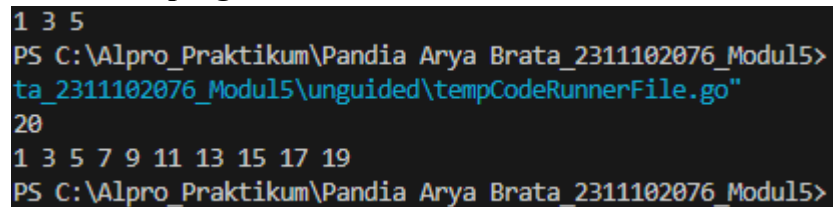
import "fmt"

func main() {
    var num int
    fmt.Scan(&num)

    bilganjil(num, 1)
}

func bilganjil(n, i int) {
    if i > n {
        fmt.Print(" ")
    } else {
        if i%2 == 1 {
            fmt.Printf("%d ", i)
            bilganjil(n, i+2)
        }
    }
}
```

Screenshot program :



```
1 3 5
PS C:\Alpro_Praktikum\Pandia Arya Brata_2311102076_Modul5>
ta_2311102076_Modul5\unguided\tempCodeRunnerFile.go"
20
1 3 5 7 9 11 13 15 17 19
PS C:\Alpro_Praktikum\Pandia Arya Brata_2311102076_Modul5>
```

Deskripsi program :

Program disini menggunakan rekursif untuk menentukan bilangan ganjil dari inputan user mulai dari terkecil hingga terbesar dengan kondisi rekursif $i > n$ maka program selesai, dan kondisi lain jika i yang sebelumnya sudah di inisiasikan $1 \% \text{modulus dari } 2 == 1$ maka, i akan di cetak dan prosedur memanggil dirinya kembali bilganjil ($n, i + 2$)

6. Sourcecode :

```
package main

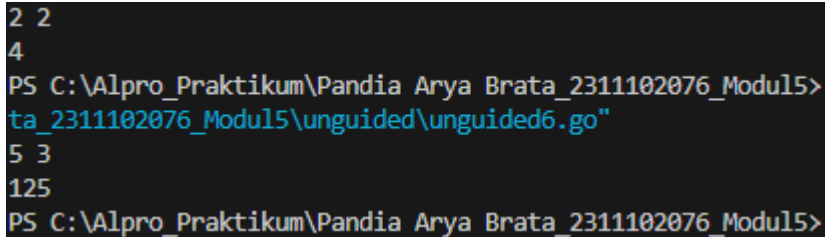
import "fmt"

func main() {
    var x, y int
    fmt.Scan(&x, &y)
```

```
    hasil := pangkat(x, y)
    fmt.Print(hasil)
}

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

Screenshot program :



```
2 2
4
PS C:\Alpro_Praktikum\Pandia Arya Brata_2311102076_Modul5>
ta_2311102076_Modul5\unguided\unguided6.go"
5 3
125
PS C:\Alpro_Praktikum\Pandia Arya Brata_2311102076_Modul5>
```

Deskripsi program :

Program rekursif ini memiliki fungsi mencari hasil pangkat dari dua buah bilangan yang di inputkan oleh user, dengan fungsi memiliki rekursif dengan kondisi apabila y atau pangkatnya adalah == 0 maka program selesai, kondisi lain yaitu bilangan x * x (memanggil kembali fungsi “pangkat(x, y-1)”)