MODUL 6. REKURSIF

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 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 **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 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.

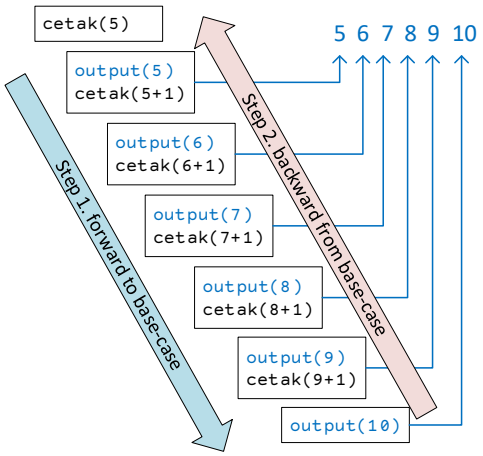
```
1  procedure cetak(in x:integer)
2  algoritma
3      if x == 10 then
4          output(x)
5      else
6          output(x)
7          cetak(x+1)
8      endif
9  endprocedure
```

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 != 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 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
5
```

Catatan:

- Teknik rekursif ini merupakan salah satu alternatif untuk mengganti struktur kontrol perulangan dengan memanfaatkan subprogram (bisa fungsi ataupun prosedure).
- 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 interatif.

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.

6.3 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

```
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)
    }
}
```

c. Mencari dua pangkat n atau 2^n

Base-case: n == 0

```
package main
import "fmt"
func main(){
    var n int
    fmt.Scan(&n)
    fmt.Println(pangkat(n))
}

func pangkat(n int) int {
    if n == 0 {
        return 1
    }else{
        return 2 * pangkat(n-1)
    }
}
```

d. Mencari nilai faktorial atau n!

Base-case: n == 0 atau n == 1

```
package main
import "fmt"
func main(){
    var n int
    fmt.Scan(&n)
    fmt.Println(faktorial(n))
}

func faktorial(n int) int {
    if n == 0 || n == 1 {
        return 1
    }else{
        return n * faktorial(n-1)
    }
}
```

6.4 Soal Latihan Modul 6

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 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

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	* ** ***

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

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

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

6) Buatlah program 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