BAB VI
RECURSIVE

- Rekursif adalah proses pemanggilan dirinya sendiri (fungsi atau prosedur).
- Fungsi maupun prosedur yang memanggil dirinya disebut fungsi atau prosedur rekursif.
- Fungsi antuk suatu bagian program yang mengembalikan (menghasilkan) hanya satu nilai. Sebuah function call adalah suatu ekspresi jadi ia memberikan satu nilai.
- Procedure adalah suatu bagian program yang melakukan aksi/fungsi khusus, biasanya berdasarkan sekumpulan parameter. Sebuah procedure call adalah suatu statemen, jadi ia melakukan aksi.
- Banyak obyek dalam matematika didefinisikan dengan menampilkan suatu proses untuk menghasilkan obyek-obyek tsb.
- Misalnya : n faktorial (n!)didefinisikan sebagai produk dari semua integer diantara n dan 1.
- Contoh lain adalah bilangan asli.

  1 adalah bilangan asli.
  Successor dari 1 adalah bilangan asli.
- Perbedaan rekursi dengan prosedur/fungsi adalah rekursi bisa memanggil kedirinya sendiri tetapi prosedur atau fungsi harus dipanggil lewat pemanggil prosedur/fungsi.
- Ciri masalah yang dapat diselesaikan secara rekursif adalah masalah tersebut dapat direduksi menjadi satu atau lebih masalah-masalah serupa yang lebih kecil.
- Secara umum suatu algoritma rekursif selalu mengandung 2 macam kasus :
  1. satu atau lebih kasus yang pemecahan masalahnya dilakukan dengan menyelesaikan masalah serupa yg lebih sederhana (menggunakan recursive call).
  2. satu atau lebih kasus pemecahan masalahnya dilakukan tanpa recursive call. Kasus ini disebut kasus dasar atau penyetop.
- Supaya tidak terjadi rekursif tak hingga, maka setiap langkah rekursif haruslah mengarah ke kasus penyetop.
- Sistem komputer mengikuti jalannya program yang rekursif biasanya dengan menggunakan suatu struktur data yang disebut stack.
- Ketika eksekusi program sampai pada suatu rekursif call, ia menghentikan sementara komputasi yg sedang dilaksanakannya sekarang untuk melakukan recursive call tsb, agar ia dapat kembali ke keadaan semula setelah recursive call itu selesai , ia harus menyimpan informasi yang cukup. Informasi yg diperlukan disebut activation frame.
- Activation frame disimpan pada bagian memori yg diatur dalam benruk stack.
- Rekursif yang berlapis-lapis dapat menghabiskan memori yang mengakibatkan stack overflow.
- Masalah yg mempunyai solusi rekursif juga mempunyai solusi iteratif(menggunakan loop).
- Versi iteratif seringkali lebih efisien daripada versi rekursif karena rekursif biasanya menggunakan memori yg lebih besar dan memerlukan waktu ekstra u/ penanganan stack of activation frame.

Contoh 1 menghitung faktorial :
n ! = 1 jika n = 0 atau n=1
n ! = n * (n-1) * (n-2) *…* 1, jika n > 0
Algoritma untuk definisi secara iteratif :

```
x ←  n
hasil ← 1
while x > 0 do
begin
      hasil  ←  hasil*x
      x←  x-1
endwhile
```

Definisi diatas dapat juga dituliskan sbb :
n ! = 1 jika n = 0 atau n=1
n ! = n * (n-1)! , jika n > 0
Algoritma untuk definisi secara rekursif:

```
Faktorial(n)
   If  (n = 0) or (n = 1) then faktorial  ←  1
     Else faktorial←  faktorial (n-1)*n
   endif
endfaktorial
```

Jika dituliskan dalam bentuk fungsi secara rekursif :

```
Uses Crt;
Function faktorial(N:integer):integer;
Begin
   If (N=0) or (N=1) then
      Faktorial := 1
   Else
      Faktorial := N * Faktorial (N-1);
End;
Var
  N:byte;
 {Program Utama}
Begin
   Clrscr;
   Write('Berapa faktorial : ');Readln(N);
   Writeln('Faktorial = ',faktorial(N);
   Readln;
End.
```

Contoh pencarian nilai 5! secara rekursif :
5 ! = 5*4!
Scr rekursif 4! Dihitung kembali sebesar 4*3! sehingga :
5 ! = 5*4*3!
Scr rekursif 3! Dihitung kembali sebesar 3*2! sehingga :
5 ! = 5*4*3*2!
Scr rekursif 2! Dihitung kembali sebesar 2*1! sehingga :
5 ! = 5*4*3*2*1 = 120

Jika dituliskan dalam bentuk prosedur secara rekursif :

```
Uses Crt;
Procedure faktorial(N:integer;var hasil : integer);
Begin
   If (N=0) or (N=1) then
      hasil := 1
   Else
      begin
            Hasil:=hasil*n;
            Faktorial (N-1,hasil);
      End;
End;
Var
  N,f:integer;
 {Program Utama}
```

```
Begin
   Clrscr;
   Write('Berapa faktorial : ');Readln(N);
   Faktorial(n,f);
   Writeln('Faktorial = ',f);
   Readln;
End.
```

Contoh 2 populer yang sering digunakan untuk menjelaskan proses rekursif adalah fungsi penjumlahan dengan mengggunakan sigma.

Perhatikan fungsi dibawah ini :

```
function Summation (num : integer) : integer;
begin
   if num = 1 then
      Summation := 1
   else
      Summation := Summation(num-1) + num;
end;
```

Asumsikan fungsi diatas dipanggil seperti dibawah ini :

```
a := Summation(3);
```

maka proses yang terjadi adalah :

- Summation (3) menjadi Summation (2) + 3
- Summation (2) menjadi Summation (1) + 2
- Pada saat num=1 proses berhenti dan summation bernilai sama dengan 1
- Summation(2) menjadi 1 + 2
- Summation(3) menjadi 3 + 3
- Dan akhirnya penjumlahan 3 bilangan menjadi 6