

Analisis Rekurens dalam Konteks Prosedural

IF2110 – Algoritma dan Struktur Data
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Tujuan

Mahasiswa memahami bahwa selain metode iteratif, mengulang suatu aksi dalam program prosedural dapat dilakukan secara rekursif

Mahasiswa memahami definisi rekursif dalam konteks pemrograman prosedural

Mahasiswa dapat mengkonstruksi algoritma sederhana secara rekursif dan mengimplementasikan dalam bahasa pemrograman terpilih

Mekanisme Pengulangan

Sampai sejauh ini untuk aksi yang dilakukan secara berulang-ulang dalam suatu algoritma digunakan pengulangan (repeat N-times, repeat-until, while-do, traversal, iterate-stop).

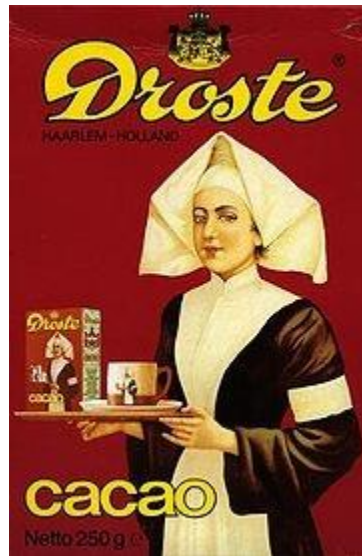
Cara menyelesaikan persoalan dengan pengulangan sering disebut cara iteratif.

Selain cara iteratif, mengulang suatu aksi dapat dilakukan secara rekursif.

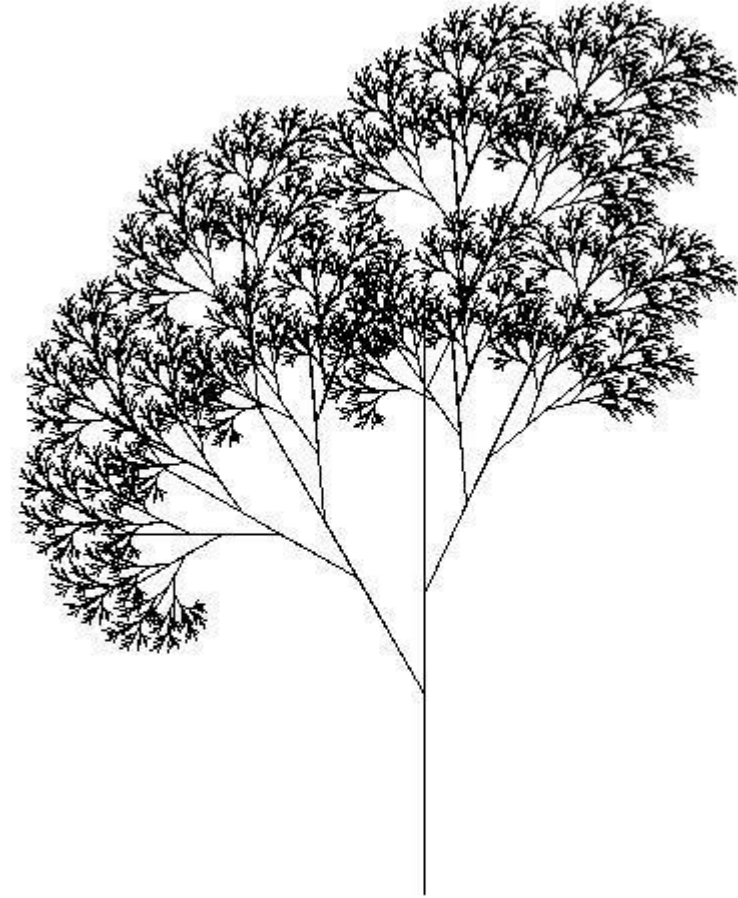
Rekursifitas

Definisi:

Suatu entitas disebut rekursif jika pada definisinya terkandung terminologi dirinya sendiri



Contoh gambar rekursif



Aplikasi Rekursifitas dalam bidang informatika

Algoritma-algoritma tingkat lanjut, misalnya:

- Algoritma *sorting* lanjut (*quick sort*, *merge sort*, dll)
- Algoritma *backtracking*

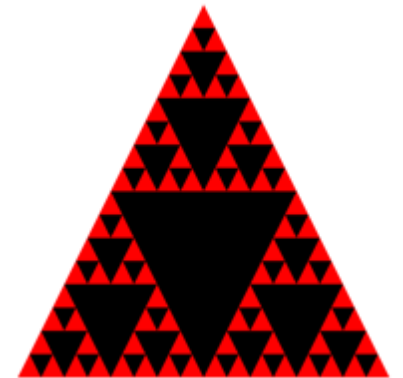
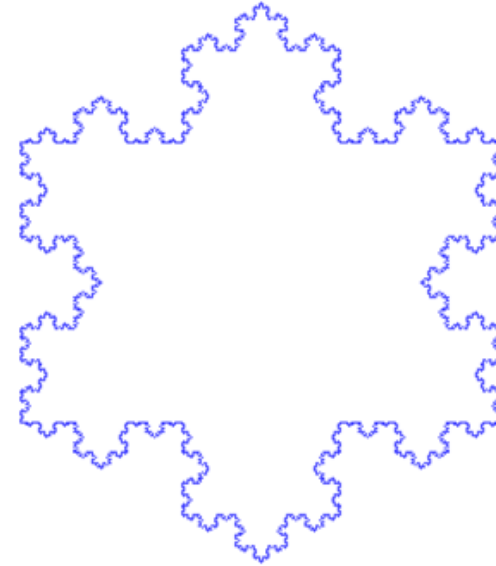
Pengelolaan struktur data kompleks (misalnya list, graf, pohon, dll.)

Aplikasi Rekursifitas dalam bidang informatika

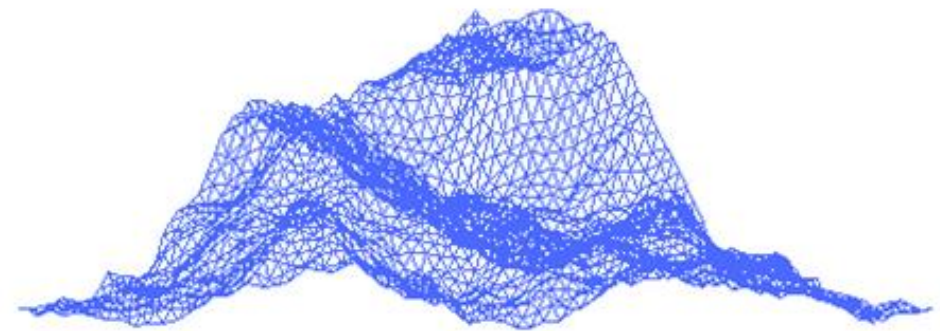
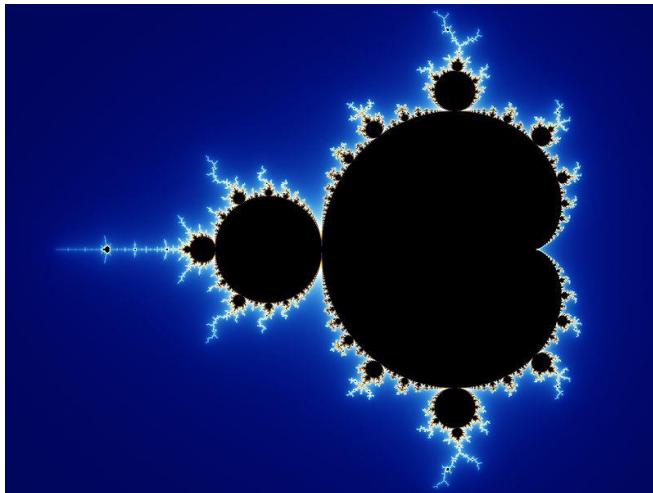
Fraktal:

bentuk-bentuk geometris yang terdiri atas bagian-bagian kecil, tiap bagian adalah kopi (dalam ukuran yang lebih kecil) dari bentuk keseluruhan

Biasanya diimplementasikan dari fungsi matematis yang bersifat rekursif

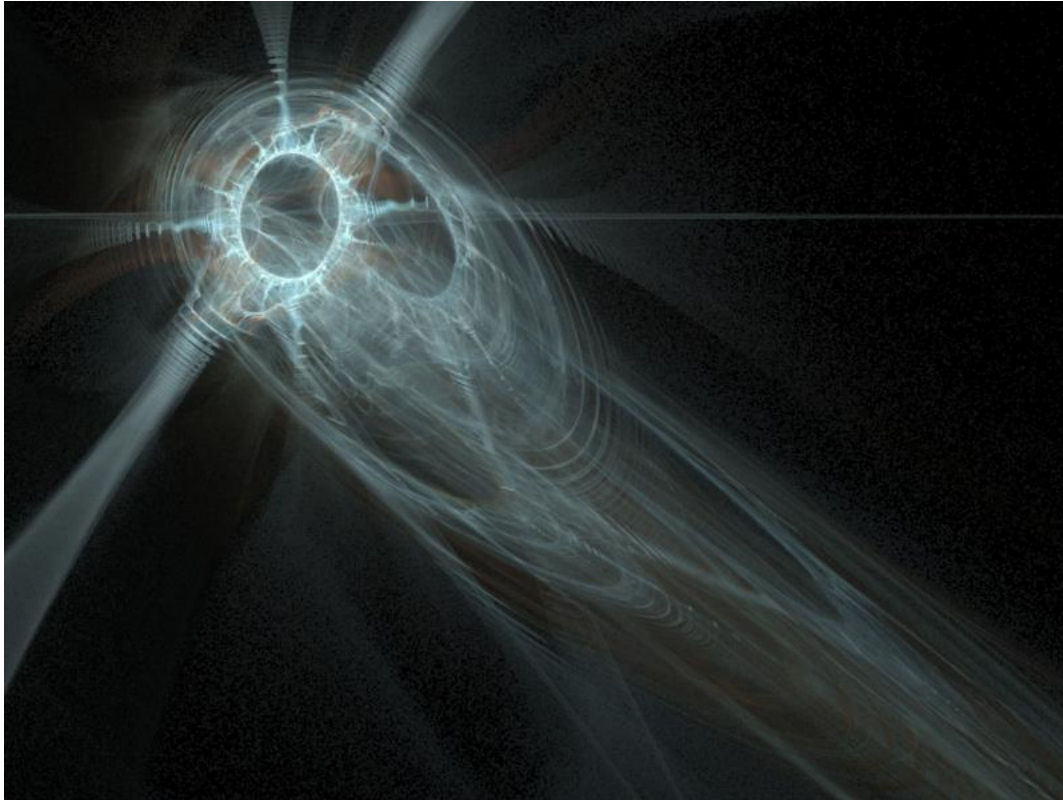


Contoh-contoh gambar fraktal



Contoh-contoh gambar fraktal

Gambar berikut dibuat dengan menggunakan Pascal 😊



Rekursif dalam Pemrograman Prosedural

Aspek pemrograman prosedural yang bisa bersifat rekursif:

- Fungsi
- Prosedur
- Struktur Data

Yang dibahas pada materi ini hanyalah **fungsi dan prosedur** yang bersifat rekursif.

Analisis Rekurens

Analisis rekurens: penalaran berdasarkan definisi rekursif.

Teks program yang bersifat rekursif terdiri dari dua bagian:

- **Basis** (Basis-0 atau Basis-1): menyebabkan prosedur/fungsi berhenti
- **Rekurens**: mengandung call terhadap prosedur/fungsi tersebut, dengan parameter bernilai mengecil (menuju basis)

Studi Kasus

Faktorial:

- Diberikan sebuah bilangan bulat ≥ 0 , yaitu N
- Harus dihitung, $N!$
- $N! = N \times (N-1) \times (N-2) \times (N-3) \times \dots \times 3 \times 2 \times 1 \times 1$

Penyelesaian secara iteratif

function factorialIter(n: integer) → integer

{ Prekondisi: $n \geq 0$ }

{ *factorialIter(n)* menghasilkan 1 jika $n = 0$;

 menghasilkan $n! = n * (n-1)!$ untuk $n > 1$ dengan loop }

{ Faktorial adl. perhitungan deret: $1 \times 1 \times 2 \times 3 \times 4 \times \dots \times n$

 sehingga hasil disimpan dalam result, dan elemen pengulangan adalah i }

KAMUS LOKAL

i: integer

result: integer

ALGORITMA

result ← 1 { inisialisasi }

i ← 1 { first element }

while (i ≤ n) do

 result ← result * i { Proses }

 i ← i + 1 { Next element }

{ i > n: sudah dihitung $1*1*2*3*\dots*n$ }

→ result { Terminasi }

Penyelesaian secara rekursif

Basis:

Jika $N = 0$, maka: $N! = 1$

Rekurens:

Jika $N > 0$, maka:

$$N! = N * (N-1) * (N-2) * (N-3) * \dots * 3 * 2 * 1 * 1$$



$$N! = N * (N-1)!$$

Penyelesaian secara rekursif (fungsi)

function factorial(n: integer) → integer

{ Mengirim $n!$ sesuai dengan definisi rekursif faktorial: $0! = 1$; $n! = n * (n-1)!$ }

KAMUS LOKAL

ALGORITMA

if (n = 0) then { Basis }
 → 1

else { rekurens }
 → n * factorial(n-1)

Penyelesaian secara rekursif (prosedur alt-1)

procedure factorialP1 (input n: integer, output result: integer)

{ I.S. $n \geq 0$ }

{ F.S. factorialP1(n,result) memberikan result=1 jika $n = 0$;

memberikan $result = n! = n*(n-1)!$ untuk $n > 0$ dengan variabel lokal untuk menyimpan hasil sementara }

KAMUS LOKAL

temp: integer

ALGORITMA

if ($n = 0$) then { basis }
 result \leftarrow 1

else { rekurens }

 factorialP1($n - 1$, temp)

 result \leftarrow $n * temp$

temp menampung hasil
perhitungan (n-1)!

Penyelesaian secara rekursif (prosedur alt-2)

```
procedure factorialP2 (input n: integer,  
                        input/output tmpResult: integer)  
  
{ I.S.  $n \geq 0$  }  
{ F.S. factorialP2(n) menghasilkan 1 jika  $n = 0$ ;  
      menghasilkan  $n! = n * (n-1)!$  untuk  $n > 0$  }  
{ dengan hasil sementara selalu disimpan pada parameter I/O }  
{ Nilai hasil harus diinisialisasi dengan 1 saat pemanggilan }
```

KAMUS LOKAL

ALGORITMA

```
if (n = 0) then { basis }  
  { do nothing: kenapa? }  
else { rekurens }  
  factorialP2(n-1, tmpResult)  
  tmpResult  $\leftarrow$  n * tmpResult
```

tmpResult adalah parameter I/O yang menampung hasil perhitungan $(n-1)!$
Di program utama harus diinisialisasi dengan 1

Program ContohRekursif

{ Program ini merupakan contoh implementasi faktorial dalam bentuk prosedur dan fungsi }

KAMUS

```
function factorial (n: integer) → integer
{ Prekondisi:  $n \geq 0$  }
{ factorial(n) menghasilkan 1 jika  $n = 0$ ;
  menghasilkan  $n! = n * (n-1)!$  untuk  $n > 1$  }
procedure factorialP1 (input n: integer, output result: integer)
{ I.S.  $n \geq 0$  }
{ F.S. factorialP1(n) menghasilkan 1 jika  $n = 0$ ;
  menghasilkan  $n! = n * (n-1)!$  untuk  $n > 0$  }
procedure factorialP2 (input n: integer,
                      input/output tmpResult: integer)
{ I.S.  $n \geq 0$  }
{ F.S. factorialP2(n) menghasilkan 1 jika  $n = 0$ ;
  menghasilkan  $n! = n * (n-1)!$  untuk  $n > 0$ , dengan paramater I.O }
{ variabel global program }
result, tmpResult: integer
```

ALGORITMA

```
output (factorial(5));
factorialP1(5, result); output (result)
tmpResult ← 1 { harus diinisialisasi dengan invarian sebab parameter I/O }
factorialP2(5, tmpResult); output (tmpResult)
```

Pemrosesan List secara Rekursif

Studi Kasus Pemrosesan List secara Rekursif

List dapat diproses secara rekursif dengan memperhatikan rentang indeks elemen yang diproses

```
constant CAPACITY: integer = 100  
constant IDX_UNDEF: integer = -999
```

```
type ElType: integer { jenis elemen list }  
type List: < contents: array [0..capacity-1] of ElType,  
               { memori tempat penyimpan elemen (container) }  
               nEff: integer  $\geq$  0 {banyaknya elemen efektif} >
```

{ Contoh penggunaan:

Deklarasi variabel: *list: List*

Pengaksesan elemen type: *list.contents[i]*
 list.nEff

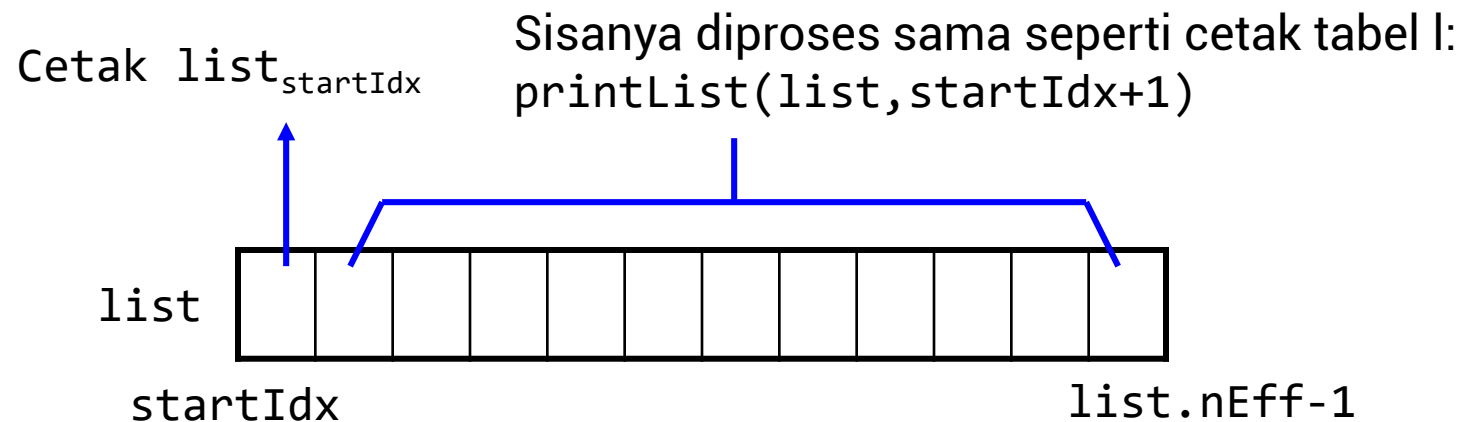
List kosong adalah List dengan list.nEff = 0 }

Print Isi List (secara Rekursif) - 1

Basis-0: $\text{startIdx} \geq \text{list.nEff}$ → array kosong

Untuk array kosong, tidak ada yang bisa dicetak → do nothing

Rekurens: $\text{startIdx} < \text{list.nEff}$



Print Isi List (secara Rekursif) - 2

```
procedure printList (input l: List, input startIdx: integer)  
{ Prosedur ini mencetak nilai elemen List l utk indeks startIdx s.d. nEff-1 }  
{ I.S. l terdefinisi dan startIdx  $\geq 0$  }  
{ F.S. Isi list l pada indeks startIdx s.d. l.nEff-1 tercetak ke layar }
```

KAMUS LOKAL

ALGORITMA

```
if (startIdx  $\geq$  l.nEff) then           { basis kosong = stop condition }  
    { do nothing }  
else                                   { startIdx < l.nEff, rekurens }  
    output (l.contents[startIdx]) { current element }  
    printList(l, startIdx+1)         { startIdx dapat di-passing sebagai ekspresi karena  
                                     merupakan parameter input }
```

Contoh Pemanggilan:

```
printList(l,0)
```


**Call Rekursif sebagai
“mekanisme mengulang”**

Call Rekursif sebagai “mekanisme mengulang”

Dalam konteks prosedural kita memiliki “loop” sebagai mekanisme untuk mengulang.

(Lihat kembali contoh fungsi faktorial iteratif.)

Kita dapat mensimulasi mekanisme pengulangan secara rekursif:

- Parameter hasil dan variabel temporer pada mekanisme pengulangan dipindahkan menjadi parameter prosedur.
- Hati-hati dalam meletakkan nilai inisialisasi untuk parameter input dan input/output

Faktorial iteratif dikelola secara rekursif alt-1

```
procedure factorialIterP1 (input n: integer, input/output i, accumulator: integer,  
                        output result: integer)  
{ I.S.:  $n \geq 0$  }  
{      Nilai i harus diinisialisasi 1; accumulator harus diinisialisasi dengan 1;  
      accumulator = i!; result sembarang }  
{ F.S.: result = 1 jika  $n = 0$  atau  $n = 1$ ;  
      result =  $n! = n*(n-1)!$  untuk  $n > 1$  dengan loop yang mekanismenya dilakukan dengan call  
      rekursif }  
{ Versi ini merupakan prosedur, dan tidak mungkin sebagai fungsi, karena adanya parameter  
  input/output }
```

KAMUS LOKAL

ALGORITMA

```
if (i > n) then { basis, stop condition }  
    result ← accumulator  
else { rekurens }  
    accumulator ← i * accumulator  
    i ← i + 1  
    factorialIterP1 (n, i, accumulator, result)
```

Bandingkan dengan
fungsi **factorialIter**

Faktorial iteratif dikelola secara rekursif alt-2

```
procedure factorialIterP2 (input n: integer, input/output accm: integer,  
                           output result: integer)  
{ I.S.:  $n \geq 0$  }  
{      Nilai accm harus diinisialisasi 1. Kesalahan inisialisasi menimbulkan kesalahan  
      komputasi. result sembarang. }  
{ F.S.: result = 1 jika  $n = 0$ ;  
      result =  $n! = n * (n-1)!$  untuk  $n > 1$  dengan loop yang mekanismenya dilakukan dengan  
      call rekursif }  
{       $n * (n-1) * (n-2) * \dots \times 3 \times 2 \times 1 \times 1$  }  
{ Versi ini merupakan prosedur, dan tidak mungkin sebagai fungsi, karena adanya parameter  
  input/output }
```

KAMUS LOKAL

ALGORITMA

```
  if (n = 0) then { basis }  
    result  $\leftarrow$  accm  
  else { rekurens }  
    accm  $\leftarrow$  accm * n  
    factorialIterP2 (n-1, accm, result)
```

Pemanggilan Prosedur Rekursif

```
procedure factorialP (input n: integer, output result: integer)  
{ I.S.  $n \geq 0$  }  
{ F.S.  $result = n!$  }  
{ Proses: menghasilkan  $n!$  dengan memanggil prosedur rekursif factorialIterP1 atau  
          factorialIterP2 }
```

KAMUS LOKAL
 i, accm: integer

```
ALGORITMA-1    { menggunakan factorialIterP1 }  
  accm  $\leftarrow$  1    { inisialisasi akumulator dengan 1 }  
  i  $\leftarrow$  1        { inisialisasi counter i dengan 1 }  
  factorialIterP1(n,i,accm,result)
```

```
ALGORITMA-2    { menggunakan factorialIterP2 }  
  Acc  $\leftarrow$  1      { inisialisasi akumulator dengan 1 }  
  factorialIterP2(n,accm,result)
```

Hati-hati:
Kesalahan melakukan inisialisasi parameter
input atau input/output dapat berakibat fatal !