

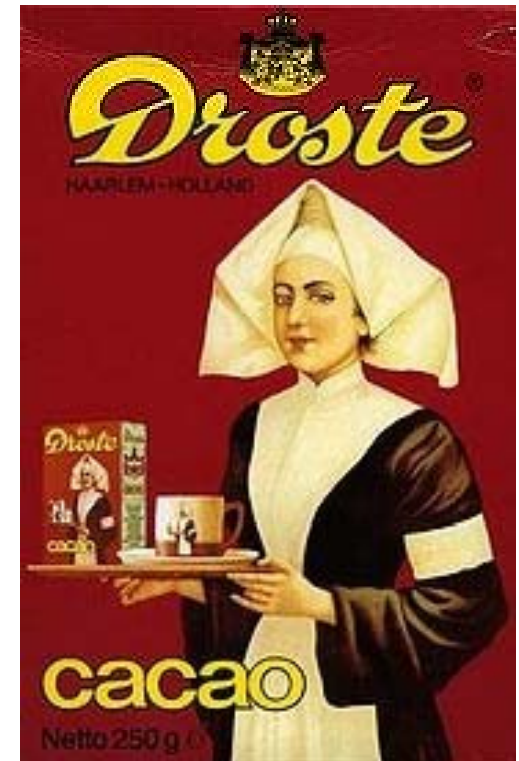


Studi List Rekursif dalam Konteks Prosedural

Tim Pengajar IF2030

Rekursifitas

- Suatu entitas disebut **rekursif** jika pada definisinya terkandung dirinya sendiri
- Program prosedural juga dapat bersifat rekursif
- Aspek-aspek pemrograman prosedural yang dapat bersifat rekursif: *prosedur, fungsi, struktur data*
 - Program utama tidak dapat bersifat rekursif karena tidak berparameter





Mengingat kembali: Analisis Rekurens

- Analisis rekurens: penalaran berdasarkan definisi rekursif
- Teks program 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)



Fungsi/Prosedur Rekursif

Studi Kasus: Faktorial

- Definisi – 1:
 $0! = 1$ { basis }
 $N! = N * (N-1)!$ { rekurens }
- Definisi – 2:
 $1! = 1$ { basis }
 $N! = N * (N-1)!$ { rekurens }
- Implementasi kasus faktorial ke dalam fungsi/prosedur rekursif dari sudut pandang:
 1. Terjemahan dari program fungsional rekursif
 2. Terjemahan sebuah loop (iteratif) menjadi rekursif (definisi faktorial ditransformasi menjadi perhitungan deret kali)



1. Terjemahan dari Program Fungsional Rekursif

- Mengingat kembali: Fungsi Factorial (Fungsional)

DEFINISI DAN SPESIFIKASI

factorial: $\text{integer} \geq 0 \rightarrow \text{integer} > 0$

{ *factorial(N) = N! sesuai dengan definisi rekursif factorial* }

REALISASI (VERSI-1)

{ *Realisasi dengan definisi factorial sebagai berikut jika factorial(N) adalah N!:*

$N = 0 : N! = 1$

$N \geq 1 : N! = N * (N-1)! \}$

factorial(N) : if $N = 0$ then { Basis 0 }

1

else { Rekurens : definisi faktorial }

$N * \text{factorial}(N-1)$



Fungsi Factorial Rekursif (Translasi ke Program Prosedural)

function factorial (N : integer) → integer

{ Mengirim N! sesuai dengan definisi faktorial definisi rekursif: }

{ 0! = 1; 1! = 1; N! = N * (N-1)! }

KAMUS LOKAL

ALGORITMA

if (N=0) then { Basis 0 }

→ 0

else {Rekurens}

→ N * factorial(N-1)

2. Terjemahan dari loop (iteratif) menjadi program rekursif



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

Fungsi Factorial (Iteratif) - 1

function factorial1 (N : integer) → integer

{ Mengirim N! sesuai dengan definisi faktorial:

1 * 1 * 2 * 3 * 4 * ... (N-1) * N }

{ Pada persoalan ini, definisi rekurens faktorial
ditransformasi menjadi perhitungan deret kali }

KAMUS LOKAL

Count : integer

F : integer

ALGORITMA

F ← 1 { inisialisasi }

Count ← 1 { first element }

while (Count <= N) do

 F ← F * i { Proses }

 Count ← Count + 1 { Next element }

{ Count > N, semua sudah dihitung }

→ F { Terminasi }

Fungsi Factorial (Iteratif) - 2



- Versi “kurang baik”, tapi sesuai definisi factorial

```
function factorial2 (N : integer) → integer  
{ Mengirim N! sesuai dengan definisi faktorial:  
  N*N-1*N-2*...*3*2*1 }  
{ Mengubah parameter input: "kurang baik" }
```

KAMUS LOKAL

F : integer

ALGORITMA

```
F ← 1 { inisialisasi }  
      { first element }  
while (N > 0) do  
    F ← F * N      { Proses }  
    N ← N - 1      { Next element }  
{ N = 0 semua sudah dihitung }  
→ F { Terminasi }
```

Prosedur faktorial iteratif dikelola secara rekursif - 1



```
procedure FactIter1 (input N, Count : integer,  
                     input Akumulator : integer,  
                     output F : integer)  
{ I.S. N > 0; Count : pencacah; Akumulator = Count! }  
{ F.S. F = N! jika Count = N }  
{ Proses : Mengirim N! sesuai dengan definisi faktorial: }  
{ versi iteratif dengan mekanisme eksekusi rekursif }
```

KAMUS LOKAL

ALGORITMA

```
if (N = Count) then  
    F ← Akumulator * N  
else { Recurrence }  
    factIter1(N, Count+1, Akumulator * Count, F)
```

**Bandingkan dengan
fungsi factorial1**



Prosedur faktorial iteratif dikelola secara rekursif - 1

```
procedure FactIter2 (input N : integer,  
                     input Akumulator : integer,  
                     output F : integer)  
{ I.S. N > 0, Akumulator = 0 }  
{ F.S. Menghasilkan F = N! }  
{ Proses : Mengirim N! sesuai dengan definisi faktorial }  
{ versi iteratif dengan mekanisme eksekusi rekursif }
```

KAMUS LOKAL

ALGORITMA

```
if (N = 0) then  
    F ← Akumulator  
else  
    factIter(N-1, Akumulator * N, F)
```

**Bandingkan dengan
fungsi factorial2**



Pemanggilan Prosedur Rekursif

```
procedure factorial (input N : integer, output F : integer)  
{ I.S. N > 0 }  
{ F.S. F = N! }  
{ Proses : menghasilkan N! dengan memanggil prosedur iterasi yang  
sesuai }
```

KAMUS LOKAL

ALGORITMA - 1

FactIter1(N,1,1,F)

ALGORITMA - 2

FactIter2(N,1,F)

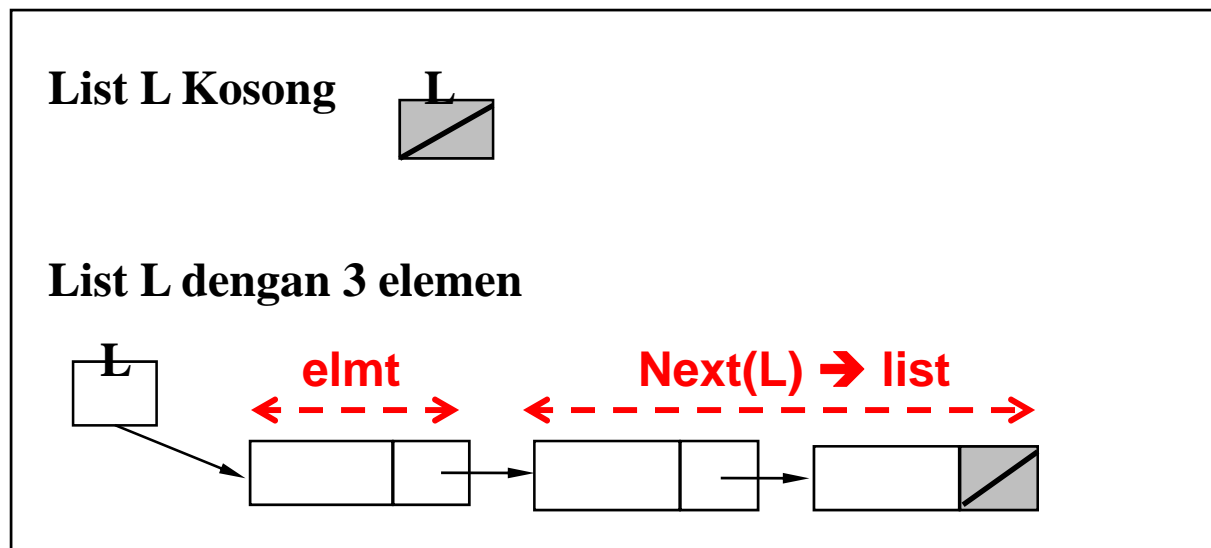
Hati-hati:
Kesalahan melakukan
inisialisasi parameter input
atau input/output dapat
berakibat fatal !



Pemrosesan List Linier secara Prosedural - Rekursif

List sebagai Struktur Data Rekursif

- Definisi rekursif list linier:
 - **Basis**: list kosong adalah list
 - **Rekurens**: list tidak kosong terdiri atas sebuah elemen dan sisanya adalah list



Struktur Data List untuk Pemrosesan secara Rekursif (Notasi Algoritmik)



KAMUS

```
{ List direpresentasi dg pointer }
    type infotype : ... { terdefinisi }
    type address  : ... { terdefinisi }
    type ElmtList : < info : InfoType,
                        next : address >
    type List : address

{ Deklarasi nama untuk variabel kerja }
    L : List
    P : address { address untuk traversal }
{ Maka penulisan First(L) menjadi L
  Next(P), Info(P) tergantung representasi fisik yang
  digunakan }
```


Struktur Data List untuk Pemrosesan secara Rekursif (Bahasa C, pointer)



```
#define Nil NULL

/* Selektor */
#define Info(P) (P)->info
#define Next(P) (P)->next

typedef int infotype;
typedef struct tElmtlist *address;
typedef struct tElmtlist {
    infotype info;
    address next;
} ElmtList;
/* Definisi list : */
/* List kosong : First(L) = Nil */

typedef address List;
```

Primitif Dasar: Pemeriksaan List Kosong

- Notasi Algoritmik (rep. berkait)

```
function IsEmpty (L : List) → boolean
{ Tes apakah sebuah list L kosong.
  Mengirimkan true jika list kosong, false jika tidak kosong }
KAMUS LOKAL
ALGORITMA
  → (L = Nil)
```

- Bahasa C (rep. berkait dgn. pointer)

```
boolean IsEmpty (List L)
/* Tes apakah sebuah list L kosong.
Mengirimkan true jika list kosong, false jika tidak kosong */
{
    /* Kamus Lokal */

    /* Algoritma */
    return (L == Nil);
}
```

Studi Kasus-1: PrintList

- Notasi Algoritmik

```
procedure Printlist (input L : List)
{ I.S. L terdefinisi }
{ F.S. Setiap elemen list diprint }
```

KAMUS LOKAL

ALGORITMA

```
if (IsEmpty(L)) then { Basis 0 }
    { tidak melakukan apa-apa }
else { Rekurens }
    output(Info(L))
    PrintList(Next(L))
```



Studi Kasus-2a: NbElmtList (1)

- Versi fungsi

```
function NBElmtlist (L : List) → integer  
{ Mengirimkan banyaknya elemen list L, Nol jika L kosong }
```

KAMUS LOKAL

ALGORITMA

```
if (IsEmpty(L)) then { Basis 0 }  
    → 0  
else { Rekurens }  
    → 1 + NBElmtList(Next(L))
```

Studi Kasus-2b: NbElmtList (2)

- Versi prosedur, dengan hasil diletakkan pada parameter output

```
procedure  NBElmtlist1 (input L : List, output N : integer)  
{ I.S. L terdefinisi }  
{ F.S. N berisi banyaknya elemen list }
```

KAMUS LOKAL

N1 : integer

ALGORITMA

```
if (IsEmpty(L)) then { Basis 0 }  
    N ← 0  
else { Rekurens }  
    NBElmtList1(Next(L),N1)  
    N ← 1 + N1
```

Studi Kasus-2c: NbElmtList (3) - 1

- Versi prosedur, dengan akumulator

```
procedure  NBElmtlistAcc (input L : List,  
                        input Acc : integer,  
                        output N : integer)  
  
{ I.S. L terdefinisi }  
{ F.S. N berisi banyaknya elemen list }
```

KAMUS LOKAL

ALGORITMA

```
if (IsEmpty(L)) then { Basis 0 }  
    N ← Acc  
else { Rekurens: Next element, Proses }  
    Acc ← Acc + 1  
    NBElmtListAcc(Next(L), Acc, N)
```



Studi Kasus-2c: NbElmtList (3) - 2

- Pemanggilan NbElmtlistAcc

```
procedure NBElmtlist (input L : List, output N : integer)  
{ I.S. L terdefinisi }  
{ F.S. N berisi banyaknya elemen list L }  
{ Proses : Memanfaatkan NBElmtlistAcc }
```

KAMUS LOKAL

ALGORITMA

NBElmtlistAcc(L, 0, N)



Studi Kasus-3: Search

```
function Search (L : List, X : infotype) → boolean  
{ Mengirim true jika X adalah anggota list, false jika tidak }
```

KAMUS LOKAL

ALGORITMA

```
  if (IsEmpty(L)) then { Basis 0 }  
    → false  
  else { Rekurens }  
    if (Info(L) = X) then  
      → true  
    else  
      → Search(Next(L), X)
```

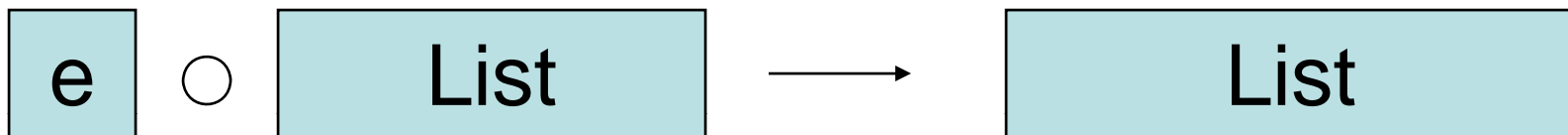



Pemrosesan List Linier secara Rekursif

(Mengacu pada Diktat
“Pemrograman Fungsional”)

Struktur Data List

- type List: [] atau [e o List]



- Primitif dasar:
 - Selektor: FirstElmt, Tail
 - Konstruktor: Konso, Kons•
 - Primitif-primitif lain: Copy, Concat, dll.

Selektor

<p><u>function</u> FirstElmt (L : List) → infotype { Mengirimkan elemen pertama sebuah list L yang tidak kosong }</p>
<p>KAMUS LOKAL</p>
<p>ALGORITMA → Info(L)</p>

<p><u>function</u> Tail (L : List) → List { Mengirimkan list L tanpa elemen pertamanya, mungkin yang dikirimkan adalah sebuah list kosong }</p>
<p>KAMUS LOKAL</p>
<p>ALGORITMA → Next(L)</p>

Konstruktor - Konso

function Konso (L : List, e : infotype) → List

{ Mengirimkan list L dengan tambahan e sebagai elemen pertamanya }
{ Jika alokasi gagal, mengirimkan L }

KAMUS LOKAL

P : address

ALGORITMA

P ← Alokasi(e)
if (P = Nil) then
 → L

else
 { Insert First }
 Next(P) ← L
 → P

Konstruktor – Kons•



function Kons• (L : List, e : infotype) → List

{ Mengirimkan list L dengan tambahan e sebagai elemen terakhir }
{ Jika alokasi gagal, mengirimkan L }

KAMUS LOKAL

P : address

ALGORITMA

if IsEmpty(L) then { insert ke list kosong }
 → Alokasi(e)
else if (Next(L) = Nil) then
 Next(L) ← Alokasi(e)
 else
 P ← Kons•(Tail(L), e)
 → L

{COBA EVALUASI, APA DAMPAKNYA TERHADAP L MASUKAN?}



Primitif Lain: Copy – 1

(versi fungsi)

```
function Copy (L : List) → List  
{ Mengirimkan salinan list L }  
{ Jika alokasi gagal, mengirimkan L }
```

KAMUS LOKAL

ALGORITMA

```
if (IsEmpty(L)) then { Basis 0 }  
    → Nil  
else { Rekurens }  
    → Konso(FirstElmt(L), Copy(Tail(L)))
```



Primitif Lain : Copy – 2

(versi procedure)

```
procedure MCopy (input Lin : List, output Lout : List)
{ I.S. Lin terdefinisi }
{ F.S. Lout berisi salinan dari Lin }
{ Proses : menyalin Lin ke Lout }
{ Jika alokasi gagal, Lout adalah ??? }
```

KAMUS LOKAL

LTemp : List

ALGORITMA

```
if (IsEmpty(Lin)) then { Basis - 0 }
    Lout ← Nil
else { Rekurens }
    MCopy(Tail(Lin), LTemp)
    Lout ← Konso(FirstElmt(Lin), LTemp)
```



Primitif Lain : Concat – 1

(versi fungsi)

function Concat (L1, L2 : List) → List

{ Mengirimkan salinan hasil konkatenasi list L1 dan L2 }

KAMUS LOKAL

ALGORITMA

if (IsEmpty(L1) then { Basis }

→ Copy(L2)

else { Rekurens }

→ Konso(FirstElmt(L1),Concat(Tail(L1),L2))



Primitif Lain : Concat – 2

(versi procedure)

```
procedure MConcat (input L1, L2 : List, output LHsl : List)
{ I.S. L1, L2 terdefinisi }
{ F.S. LHsl adalah hasil melakukan konkatenasi L1 dan L2 dengan
cara "disalin" }
{ Proses : Menghasilkan salinan hasil konkatenasi list L1 dan L2 }
```

KAMUS LOKAL

LTemp : List

ALGORITMA

```
if (IsEmpty(L1)) then { Basis - 0 }
    LHsl ← Copy(L2)
else { Rekurens }
    MConcat(Tail(L1), L2, LTemp)
    LHsl ← Konso(FirstElmt(L1), LTemp)
```



PR

- Ambillah ADT list of integer yang pernah dikerjakan sebagai tugas pra-praktikum pada pemrograman fungsional
- Translasikan menjadi program prosedural dalam bahasa C

Potongan header file (1)



```
#ifndef listrek_H
#define listrek_H

#include "boolean.h"
#include <stdio.h>

# define Nil NULL

typedef int infotype;
typedef struct tElmtlist *address;
typedef struct tElmtlist {
    infotype info;
    address next;
} ElmtList;

typedef address List;

/* Selektor */
#define Info(P) (P)->info
#define Next(P) (P)->next
```



Potongan header file (2)

```
/* Manajemen Memori */

address Alokasi (infotype X);
/* Mengirimkan address hasil alokasi sebuah elemen */
/* Jika alokasi berhasil, maka address tidak nil, dan misalnya */
/* menghasilkan P, maka info(P)=X, Next(P)=Nil */
/* Jika alokasi gagal, mengirimkan Nil */

void Dealokasi (address P);
/* I.S. P terdefinisi */
/* F.S. P dikembalikan ke sistem */
/* Melakukan dealokasi/pengembalian address P */
```

Potongan header file (3)



```
/* Pemeriksaan Kondisi List */
int IsEmpty(List L);
/* Mengirimkan 1 jika L kosong dan 0 jika L tidak kosong */
int IsOneElmt(List L);
/* Mengirimkan 1 jika L berisi 1 elemen dan 0 jika > 1 elemen atau kosong */

/* Primitif-Primitif Pemrosesan List */
infotype FirstElmt (List L);
/* Mengirimkan elemen pertama sebuah list L yang tidak kosong */
List Tail(List L);
/* Mengirimkan list L tanpa elemen pertamanya, mungkin menjadi list kosong */
List Konso(List L, infotype e);
/* Mengirimkan list L dengan tambahan e sebagai elemen pertamanya. e dialokasi
    terlebih dahulu. Jika alokasi gagal, mengirimkan L */

/* Fungsi dan Prosedur Lain */
...

#endif
```