Operasi Primitif List Linier

IF1210 – Algoritma dan Pemrograman 1
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Definisi ADT

```
KAMUS UMUM
  type ElType: integer
  type Address: pointer to Node
  type Node: < info: ElType,
               next: Address >
  type List: Address

{ Konstruktor }
procedure CreateList(output l: List)
{ I.S. Sembarang
  F.S. Terbentuk list l kosong: l diinisialisasi dengan NIL }
KAMUS LOKAL
  -
ALGORITMA
  l ← NIL
```

Operasi-operasi

- isEmpty
- indexOf
- length
- akses (getElmt, setElmt)
- insert-
  - -First
  - -At
  - -Last
- delete-
  - -First
  - -At
  - -Last
- concat

isEmpty

```
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)
```

indexOf

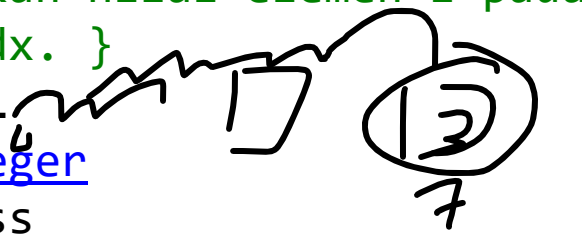
```
function indexOf(l: List, val: ElType) → integer
{ Prekondisi: l, x terdefinisi. Mengembalikan indeks elemen pertama l yang bernilai x
  (jika ada), atau mengembalikan IDX_UNDEF jika tidak ada. }
KAMUS LOKAL
  idx: integer;   p: Address;   found: boolean
ALGORITMA
  p ← l; found ← false; idx ← 0
  while p≠NIL and not found do
    if p↑.info=val then
      found ← true
    else
      idx ← idx+1
      p ← p↑.next
  if found then
    → idx
  else
    → IDX_UNDEF
```

length

```
function length(l: List) → integer
{ Prekondisi: l terdefinisi.
  Menghasilkan banyaknya elemen pada list l, 0 jika list kosong. }
KAMUS LOKAL
  ctr: integer {penghitung/counter}
  p: Address
ALGORITMA
  ctr ← 0
  p ← l
  while p≠NIL do
    ctr ← ctr+1
    p ← p↑.next
  { p=NIL }
  → ctr
```

akses (getElmt, setElmt)

```
function getElmt(l: List,
                 idx: integer) → ElType
{ Prekondisi: l terdefinisi,
    idx indeks yang valid dalam l,
    yaitu 0..length(l).
  Mengirimkan nilai elemen l pada
  indeks idx. }
KAMUS LOKAL
  ctr: integer
  p: Address
ALGORITMA
  ctr ← 0
  p ← l
  while ctr<idx do
    ctr ← ctr+1
    p ← p↑.next
  {ctr=idx}
  → p↑.info
```

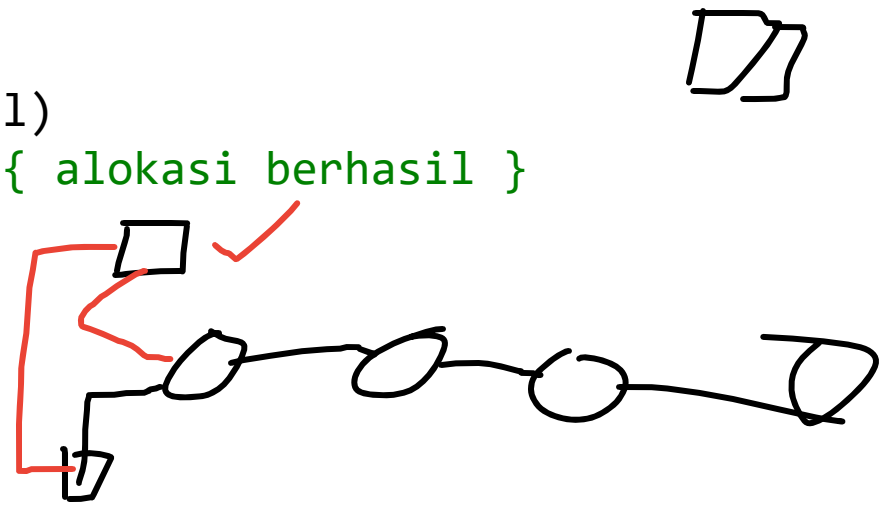


```
procedure setElmt(input/output l: List,
   input idx: integer, input val: ElType)
{ I.S. l terdefinisi, idx indeks yang
       valid dalam l, yaitu 0..length(l).
  F.S. elemen l pada indeks ke-idx
       diganti nilainya menjadi val. }

KAMUS LOKAL
  ctr: integer
  p: Address
ALGORITMA
  ctr ← 0
  p ← l
  while ctr<idx do
    ctr ← ctr+1
    p ← p↑.next
  {ctr=idx}
  p↑.info ← val
```

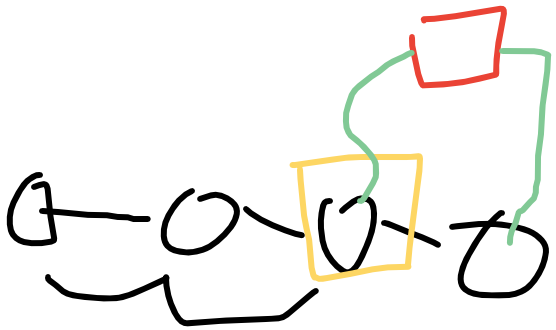
insertFirst

```
procedure insertFirst(input/output l: List, input val: ElType)
{ I.S. l terdefinisi, mungkin kosong.
  F.S. x menjadi elemen pertama l. }
KAMUS LOKAL
  p: Address
ALGORITMA
  p ← newNode(val)
  if p≠NIL then { alokasi berhasil }
    p↑.next ← l
    l ← p
```



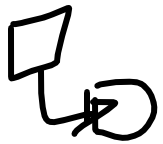
insertAt

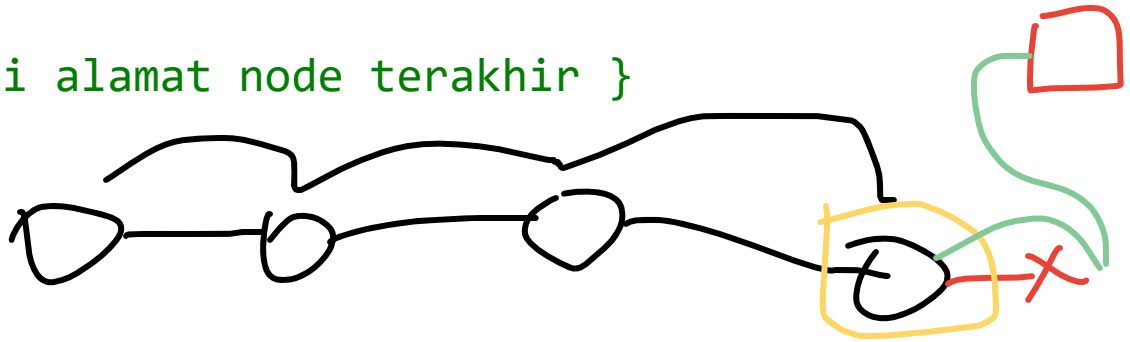
```
procedure insertAt(input/output l: List, input val: ElType, input idx: integer)
{ I.S. l terdefinisi, tidak kosong, i merupakan indeks yang valid di l.
  F.S. x disisipkan dalam l pada indeks ke-i (bukan menimpa elemen di i). }
KAMUS LOKAL
  ctr: integer
  p, loc: Address
ALGORITMA
  if idx=0 then
    insertFirst(l,val)
  else
    p ← newNode(val)
    if p≠NIL then { alokasi berhasil }
       ctr ← 0
       loc ← l
       while ctr<idx-1 do
          ctr ← ctr+1
          loc ← loc↑.next
       {ctr=idx-1}
       p↑.next ← loc↑.next
       loc↑.next ← p
```



insertLast

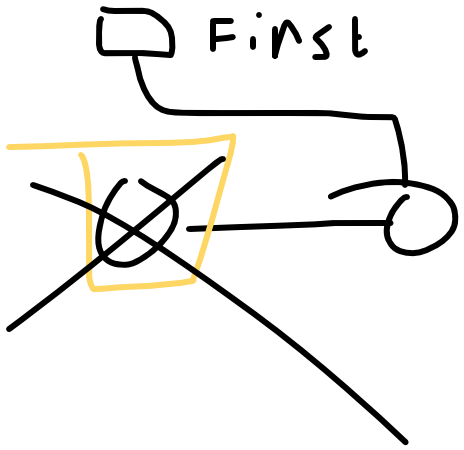
```
procedure insertLast(input/output l: List, input val: ElType)
{ I.S. l terdefinisi, mungkin kosong.
  F.S. x menjadi elemen terakhir l. }
KAMUS LOKAL
  p, last: Address
ALGORITMA
  if isEmpty(l) then
    insertFirst(l,val)
  else { List tidak kosong */ }
    p ← newNode(val)
    if p≠NIL then { alokasi berhasil }
      last ← l
      while (last↑.next≠NIL) do { cari alamat node terakhir }
        last ← last↑.next
      {last↑.next=NIL}
      last↑.next ← p
```





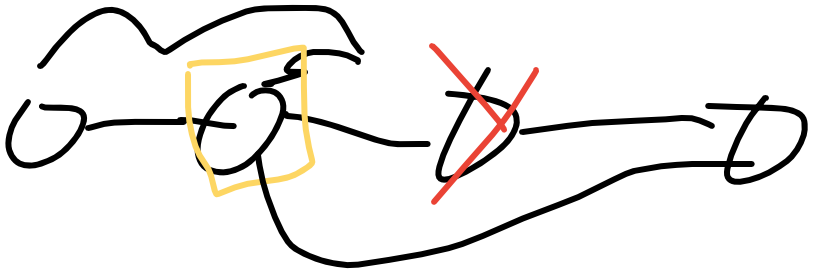
deleteFirst

```
procedure deleteFirst(input/output l: List, output val: ElType)
{ I.S. l terdefinisi, tidak kosong.
  F.S. e diset dengan elemen pertama l, elemen pertama l dihapus dari l. }
KAMUS LOKAL
  p: Address
ALGORITMA
  p ← l
  val ← p↑.info
  l ← p↑.next
  dealokasi(p)
```



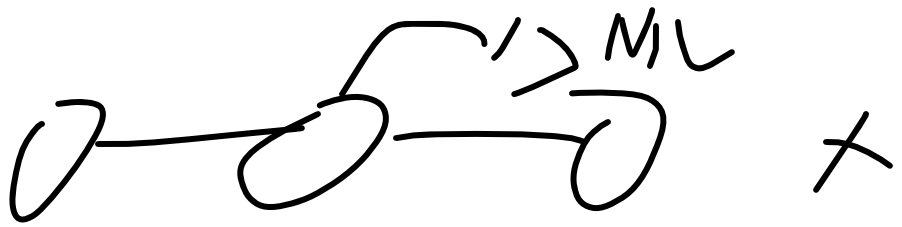
deleteAt

```
procedure deleteAt(input/output l: List, input idx: integer, output val: ElType)
{ I.S. l terdefinisi, tidak kosong, i merupakan indeks yang valid di l.
  F.S. e diset dengan elemen l pada indeks ke-idx.
       Elemen l pada indeks ke-idx dihapus dari l. }
KAMUS LOKAL
  ctr: integer
  p, loc: Address
ALGORITMA
  if idx=0 then
     deleteFirst(l,val)
  else
    ctr ← 0
    loc ← l
    while ctr<idx-1 do
      ctr ← ctr+1
      loc ← loc↑.next
    {ctr=idx-1}
    p ← loc↑.next
    val ← p↑.info
    loc↑.next ← p↑.next
    dealokasi(p)
```



deleteLast

```
procedure deleteLast(input/output l: List, output val: ElType)
{ I.S. l terdefinisi, tidak kosong.
  F.S. e diset dengan elemen terakhir l, elemen terakhir l dihapus dari l. }
KAMUS LOKAL
  p, loc: Address
ALGORITMA
  p ← l
  loc ← NIL
  while p↑.next≠NIL do
    loc ← p
    p ← p↑.next
 {p↑.next=NIL}
  if loc=NIL then
    l ← NIL
  else
    loc↑.next ← NIL
  val ← p↑.info
  dealokasi(p)
```



concat

```
function concat(l1: List, l2: List) → List
{ Prekondisi: l1 dan l2 terdefinisi, mungkin kosong.
  Mengembalikan hasil Konkatenasi ("Menyambung") dua buah list, l2 ditaruh di
  belakang l1 }
KAMUS LOKAL
  p: Address;  l3: List
ALGORITMA
  CreateList(l3)
  p ← l1
  while p≠NIL do
    insertLast(l3,p↑.info)
    p ← p↑.next
  {p=NIL}
  p ← l2
  while p≠NIL do
    insertLast(l3,p↑.info)
    p ← p↑.next
 {p=NIL}
  → l3
```