Senarai berantai

linked list

Pertemuan keenam

Struktur data

st3telkom.ac.id

by : tenia wahyuningrum
& Sisilia Thya Safitri

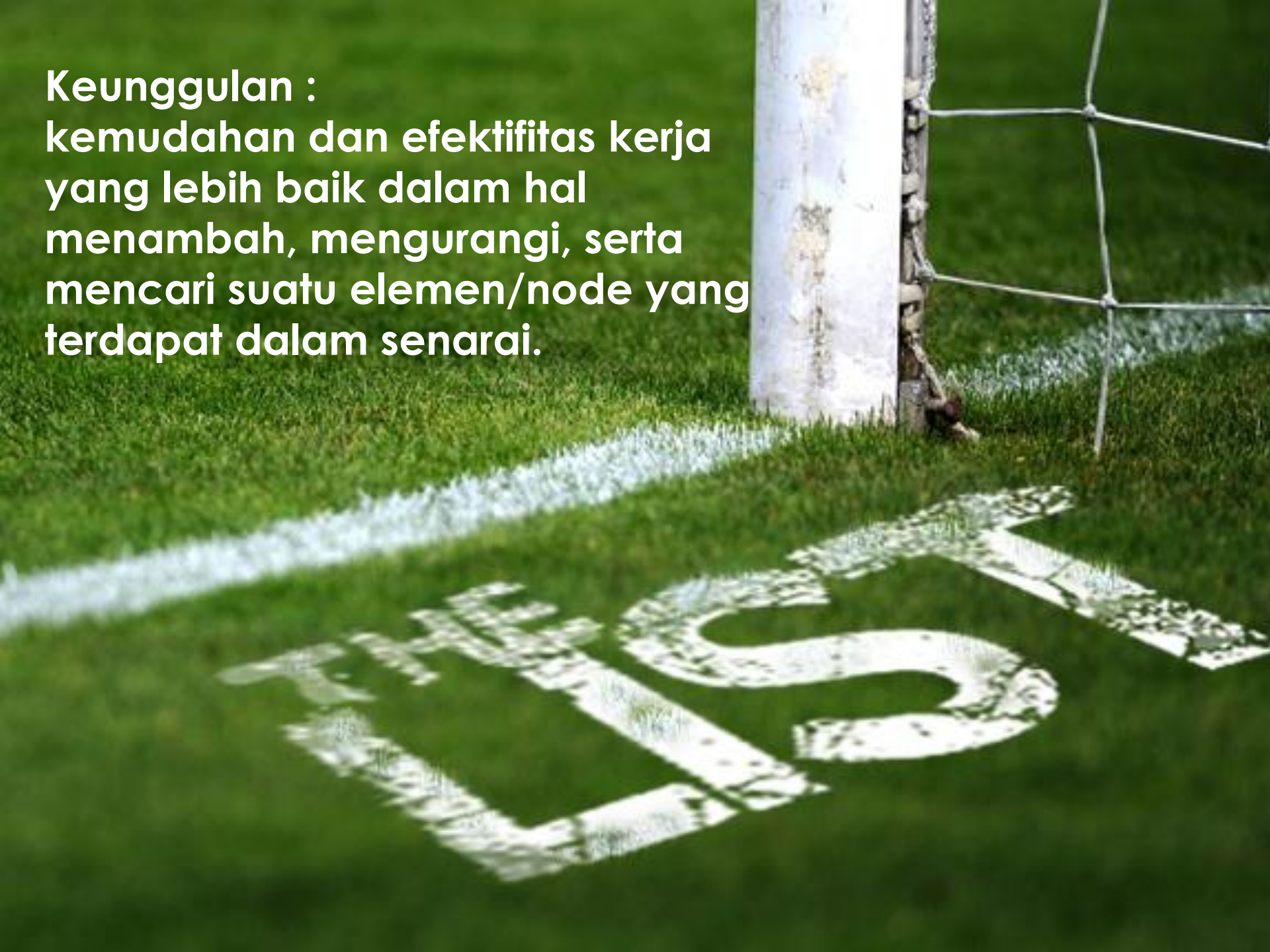
Senarai berantai

Dalam pemakaian sehari-hari istilah senarai berantai (list) adalah

kumpulan linier

sejumlah data.


Ada yang
dihapus
jika barang
telah dibeli
atau
ditambah
kan barang
baru yang akan
dibeli.
Daftar
belanja



Keunggulan :
kemudahan dan efektifitas kerja yang lebih baik dalam hal menambah, mengurangi, serta mencari suatu elemen/node yang terdapat dalam senarai.

"Elemen-elemen yang terdapat pada sebuah senarai berantai tersimpan dalam blok memori terpisah"

"Penambahan, pengurangan, ataupun penggantian node dapat dilakukan dengan

mengubah

elemen rujukan atas tiap-tiap node yang terkait"

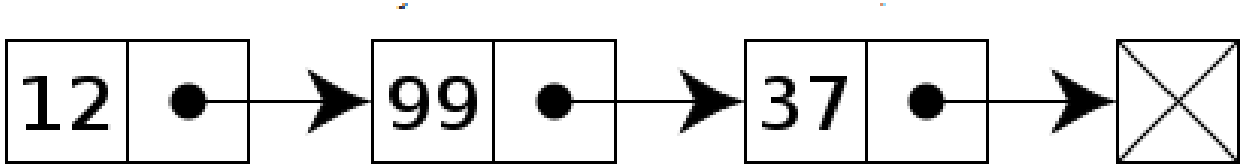
"Kerugiannya, sebuah senarai berantai tidak memungkinkan pengaksesan elemen secara acak"

Jenis-jenis

senarai berantai

Senarai Tunggal

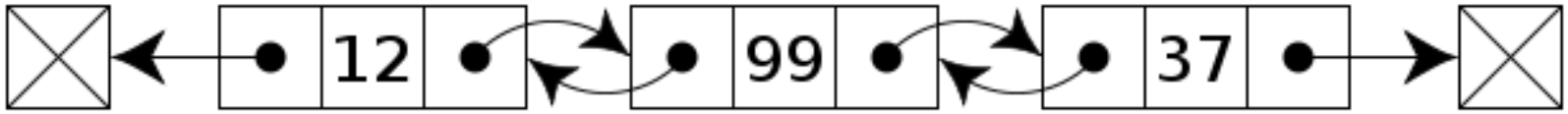
Bila struktur data sebuah node hanya memiliki satu tautan atas node berikutnya dalam sebuah senarai, maka senarai tersebut dinamakan sebagai senarai tunggal.



Senarai tunggal dengan tiap-tiap node yang terdiri atas dua elemen, data integer, dan elemen rujukan ke node berikutnya

Senarai Ganda

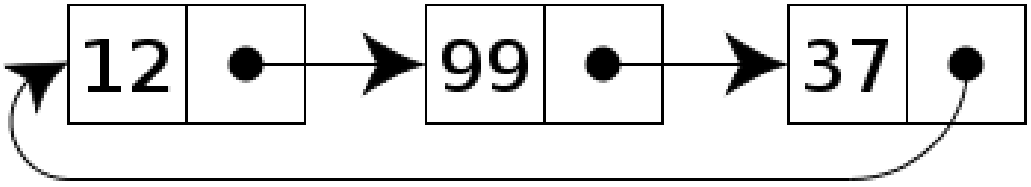
Berbeda halnya dengan senarai tunggal, pada senarai ganda, struktur data atas tiap-tiap node memiliki rujukan pada node sebelum dan berikutnya. Sebagian algoritma membutuhkan taut ganda, contohnya sorting dan reverse traversing.



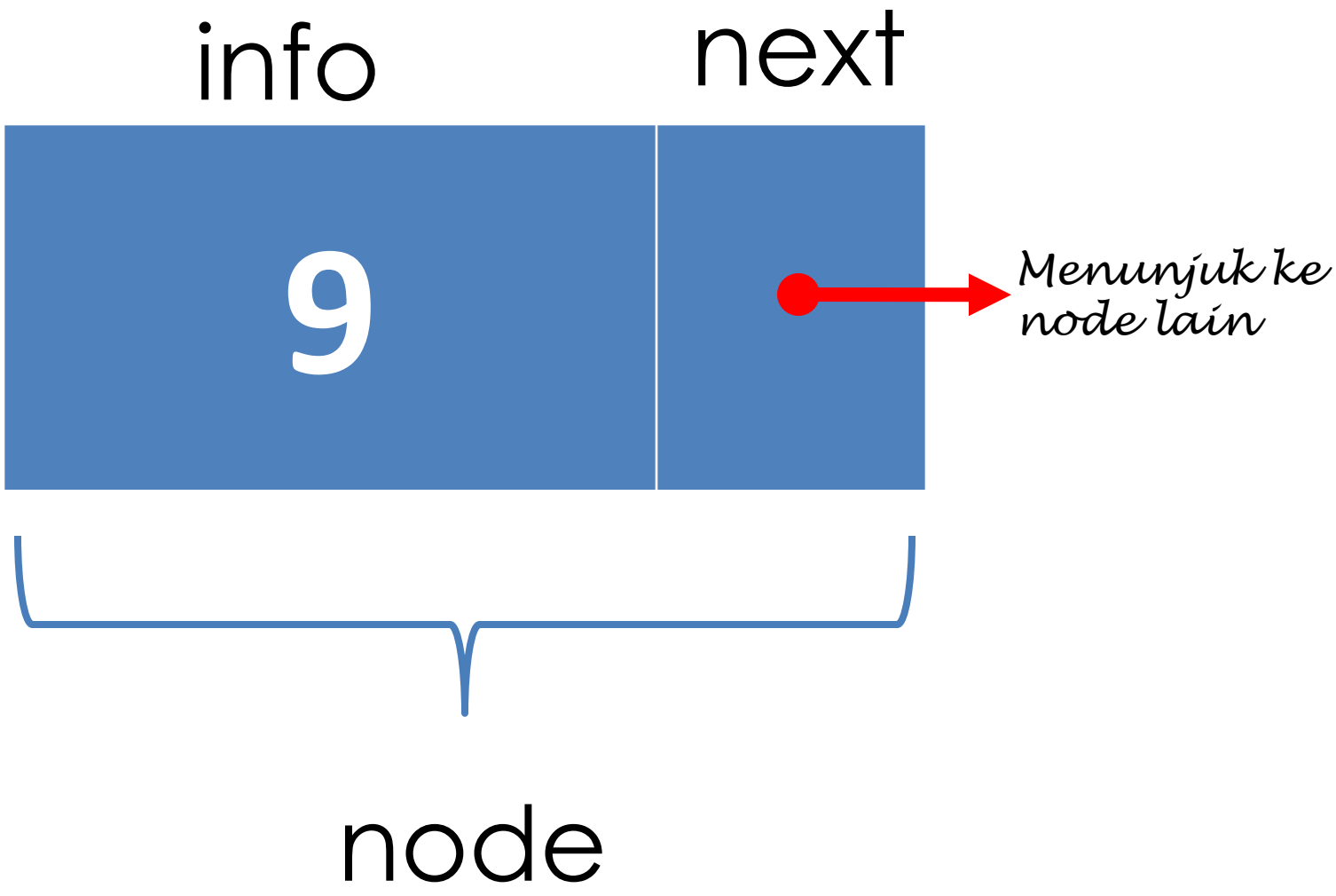
Senarai ganda dengan tiap-tiap node yang terdiri atas tiga elemen, data integer, dan dua elemen rujukan ke node sebelum serta berikutnya

- Pada dua jenis senarai sebelumnya, node terakhir dalam senarai tersebut merujuk pada **null** yang artinya akhir dari sebuah senarai, begitu pula **null** sebagai rujukan node sebelumnya pada node pertama bila senarai yang dimaksudkan adalah senarai ganda.

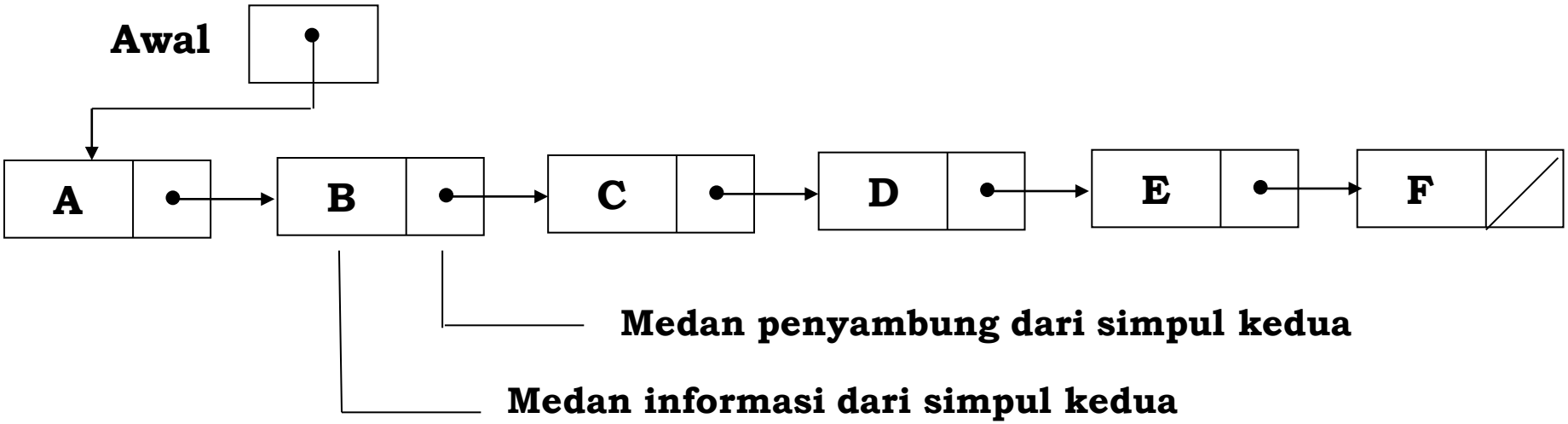
- Pada senarai sirkular, informasi rujukan pada node terakhir akan merujuk pada node pertama, dan rujukan pada node pertama akan merujuk pada node terakhir bila yang digunakan sebagai dasar implementasi adalah senarai ganda.



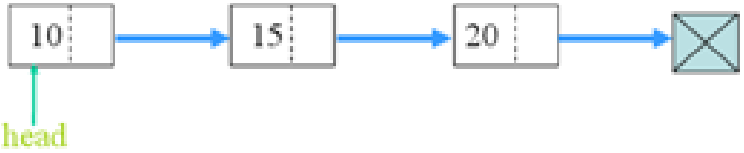
Senarai sirkular dengan menggunakan model implementasi senarai tungal. Node terakhir menyimpan rujukan pada node pertama


info
next
9
Menunjuk ke node lain
node

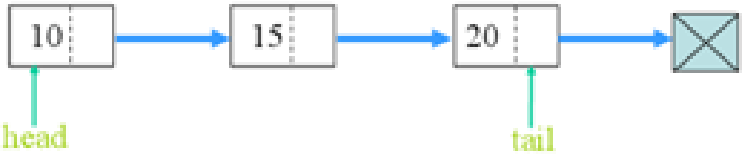
Contoh senarai berantai dengan 6 simpul :

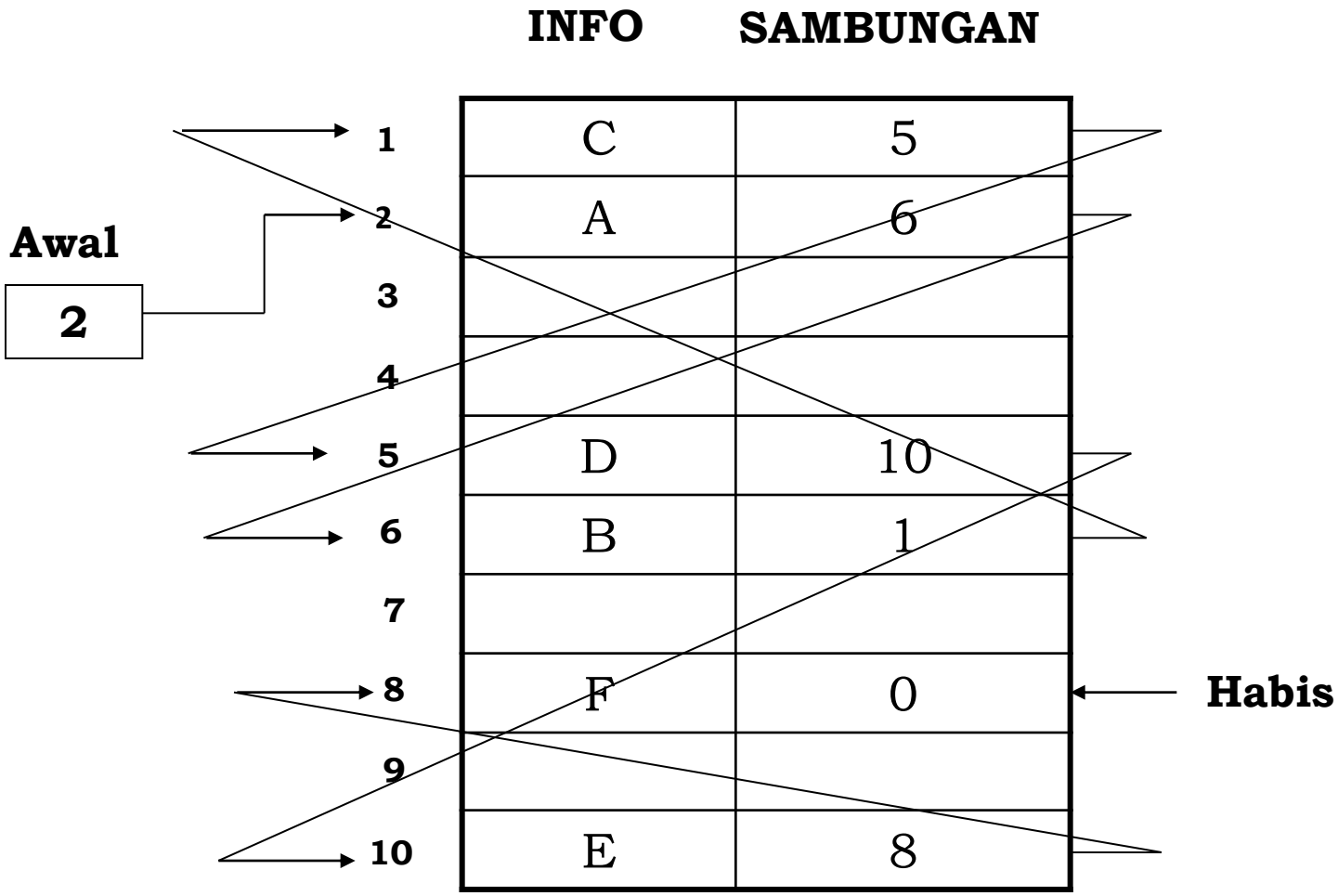


Single linked list dengan HEAD



Single linked list dengan HEAD dan TAIL




INFO
SAMBUNGAN
Awal
2
1
2
3
4
5
6
7
8
9
10
C
5
A
6
D
10
B
1
F
0
E
8
Habis

`Awal`	=	2,	maka	`Info[ 2]`	=	“A”
`Sambungan[ 2]`	=	6,	maka	`Info[ 6]`	=	“B”
`Sambungan[ 6]`	=	1,	maka	`Info[ 1]`	=	“C”
`Sambungan[ 1]`	=	5,	maka	`Info[ 5]`	=	“D”
`Sambungan[ 5]`	=	10,	maka	`Info[10]`	=	“E”
`Sambungan[10]`	=	8,	maka	`Info[ 8]`	=	“F”
`Sambungan[ 8]`	=	0,	akhir senarai berantai			

sehingga kita dapat memperoleh rangkaian “ABCDEF”.


ARE YOU
READY?

Senarai sederhana

```
#include <iostream>

using namespace std;

struct node{

int data;
node *next;
};


int main()
{

    node *n;
    node *t;
    node *h;

    n= new node;
    n->data=1;
    t=n;
    h=n;

    n= new node;
    n->data=2;
    t->next=n;
    t=t->next;

    n=new node;
    n->data=3;
    t->next=n;
    n->next=NULL;

    t=h;

    while( t!= NULL ){
    cout << "Data : "
    << t->data <<
    endl;
    t = t->next;
    }

     return 0;
}
```

Latihan soal

```
No. Baris
1  #include <iostream>
2
3  using namespace
   std;
4
5  struct node{
6
7  int data;
8  node *next;
9  };
10
11
12 int main()
13 {
14
15      node *n;
16      node *t;
17      node *h;

18      node *bantu;
19
20
21      n= new node;
22      n->data=2;

23 t=n;

24 h=n;
25
```

```
No. Baris
26      n= new node;
27      n->data=5;

28      t->next=n;
29      t=t->next;
30
31      n= new node;
32      n->data=8;
33      n->next=NULL;
34      t->next=n;
35      bantu=n;
36
37      n=new node;
38      n->data=7;
39      t->next=n;
40      n->next=bantu;
41
42

43 t=h;
44
45       while( t!= NULL ){
46           cout << "Data : " << t->data <<
             endl;
47           t = t->next;

48           }

49      return 0;
50 }
```

Tuliskan Outputnya!

Representasi Simpul (Node)

```
typedef struct node *list;
  struct node {
  int datalist;
  struct node *next;
};
```

Membuat tipe data baru (tipe data bentukan)

dilanjutkan dengan deklarasi dari pointer ke struktur di atas sebagai berikut:

```
struct node *head;
```

atau

```
list head
```

Mengalokasikan node

```
int allocate_node(int data, list *new)
{
   new = (list) malloc (sizeof(node));
   if(new==NULL)
   return 0;

   new->datalist = data;
   new->next=NULL;
   return 1;
}
```

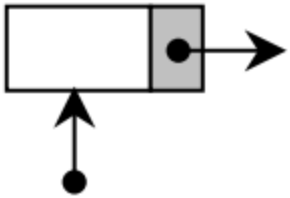
/* Perintah “malloc” merupakah perintah “memory allocation”. Perintah ini berfungsi mengalokasikan suatu alamat pada memory pada sebuah variabel*/

Untuk inisialisasi list setelah alokasi untuk node pertama maka ditambahkan statement sebagai berikut:

```
head = new
```

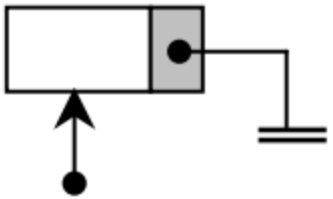
Ilustrasi fungsi allocate_node

```
new->datalist = data;
```



new

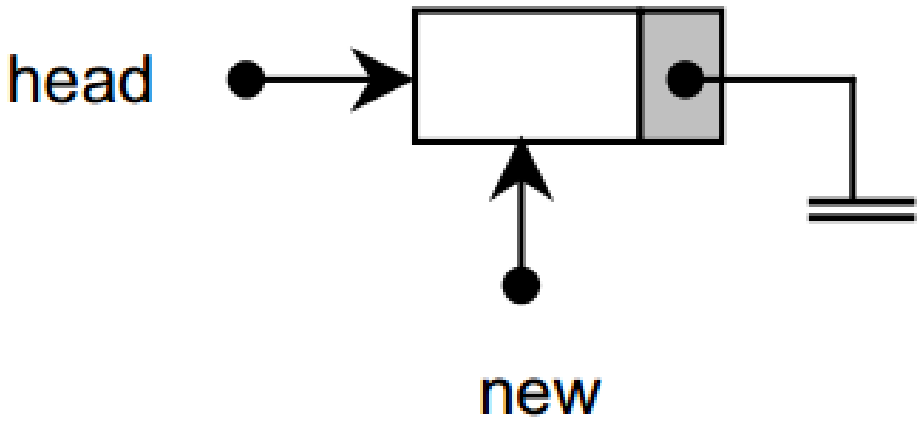
```
new->next=NULL;
```



new

Untuk inisialisasi list setelah alokasi untuk node pertama ilustrasinya adalah sebagai berikut

```
head = new;
```



operasi

senarai berantai

insert

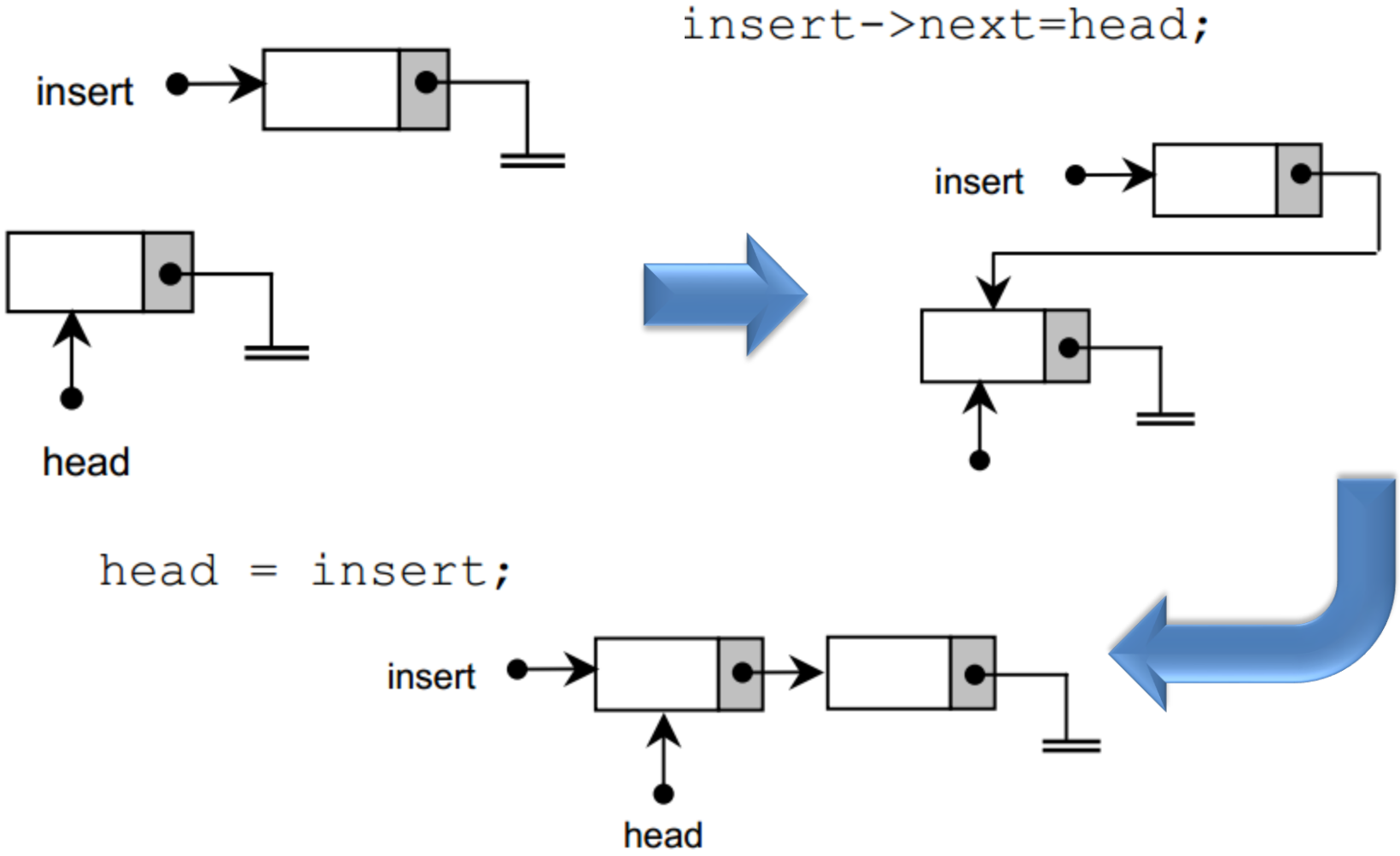
Fungsi insert pada linked list meliputi :

- insert sebagai node awal (head) dari linked list
- insert setelah node tertentu
- insert sebelum node tertentu
- insert sebagai node akhir (tail) dari linked list

Insert sebagai node awal (head) dari linked list

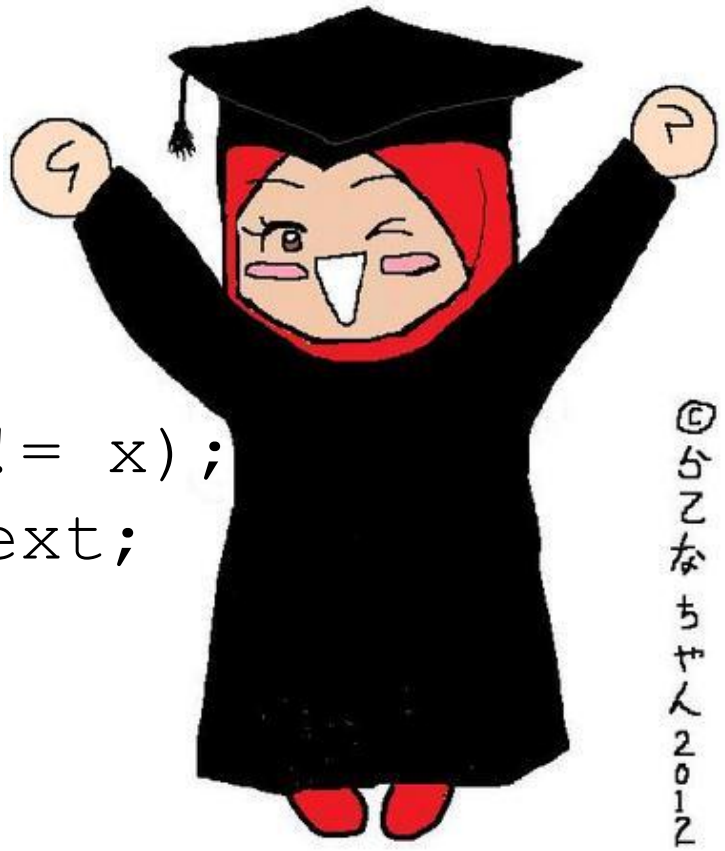
```
void insertashead(list insert)
{
insert->next=head;
 head = insert;
}
```

ilustrasi dari fungsi diatas adalah sebagai berikut:

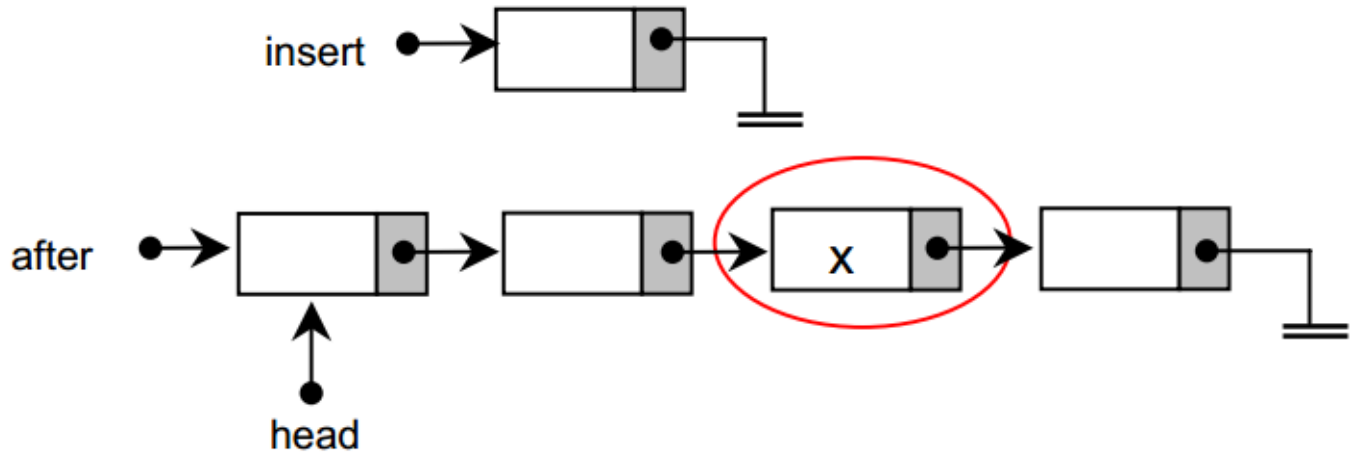


Insert setelah node tertentu

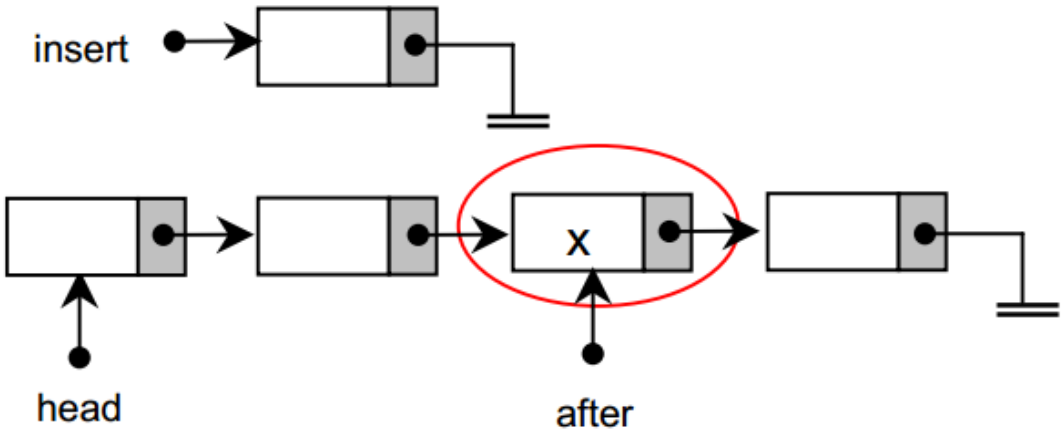
```
void insertafternode(int x, list
  insert)
{
list after;
 after = head;
do
after = after->next;
 while (after->datalist != x);
 insert->next = after->next;
 after->next = insert;
}
```



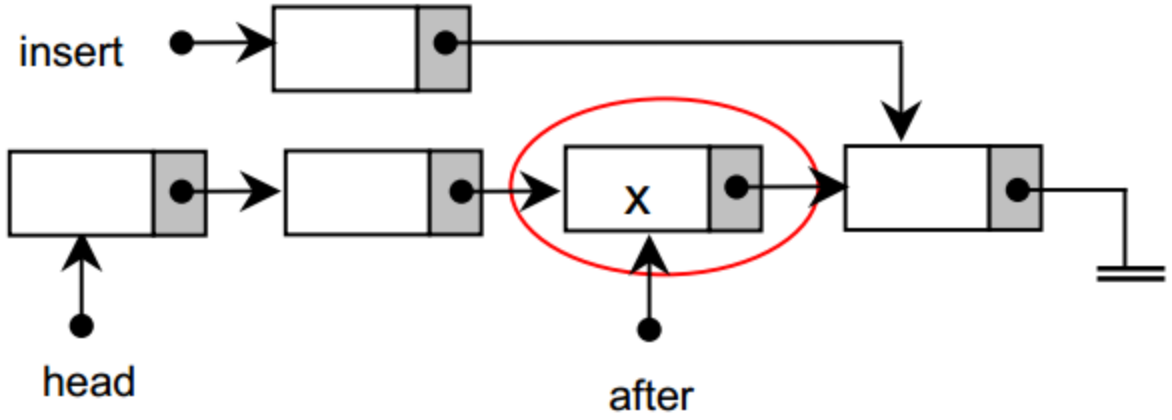
```
after = head;
```



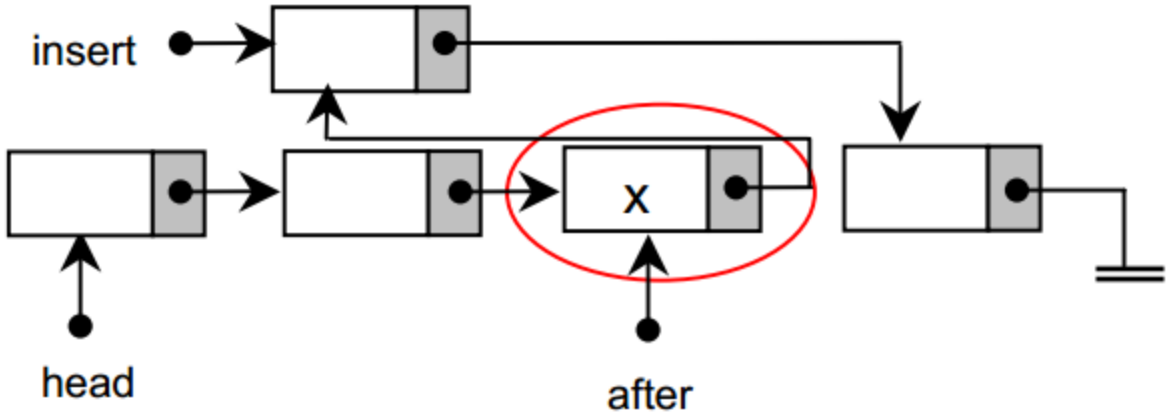

```
do
        after = after->next;
while (after->datalist != x);
```



```
insert->next = after->next;
```




```
after->next = insert;
```



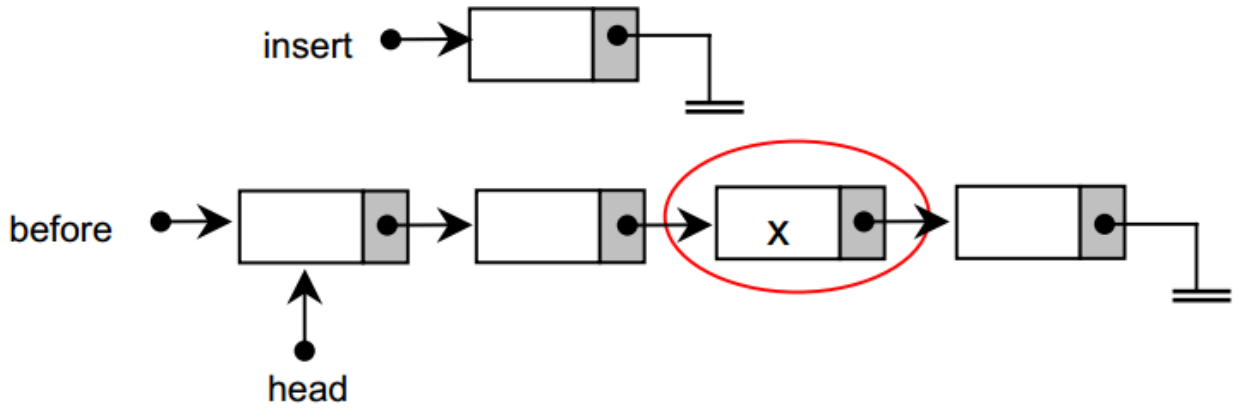
Langkah-langkah untuk proses di atas adalah sebagai berikut:

1. Pointer next dari elemen baru menunjuk elemen setelah elemen tertentu
2. Pointer next elemen sebelumnya menunjuk ke elemen baru

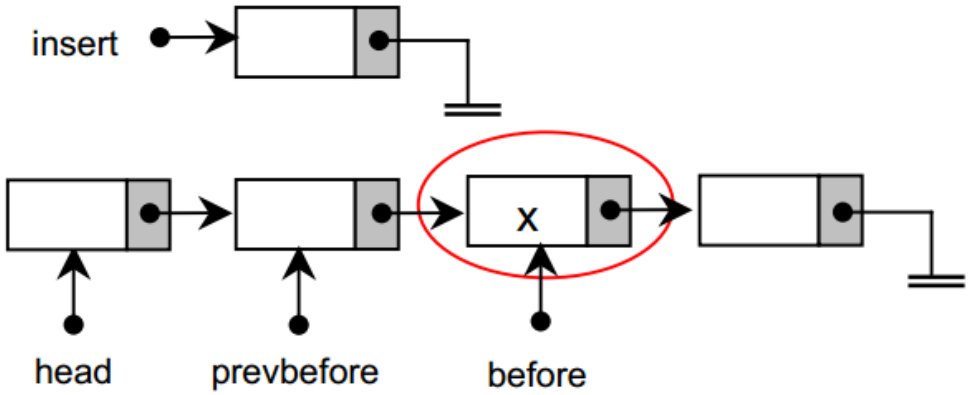
Insert sebelum node tertentu

```
void insertbeforenode(int x, list insert)
{
     list before, prevbefore;
     if (head->datalist = x)
     insertashead(insert)
else
{
 before = head;
    do
       prevbefore = before;
       before = before->next;
    while (before->datalist != x);
     insert->next = before;
     prevbefore->next = insert;
} }
```

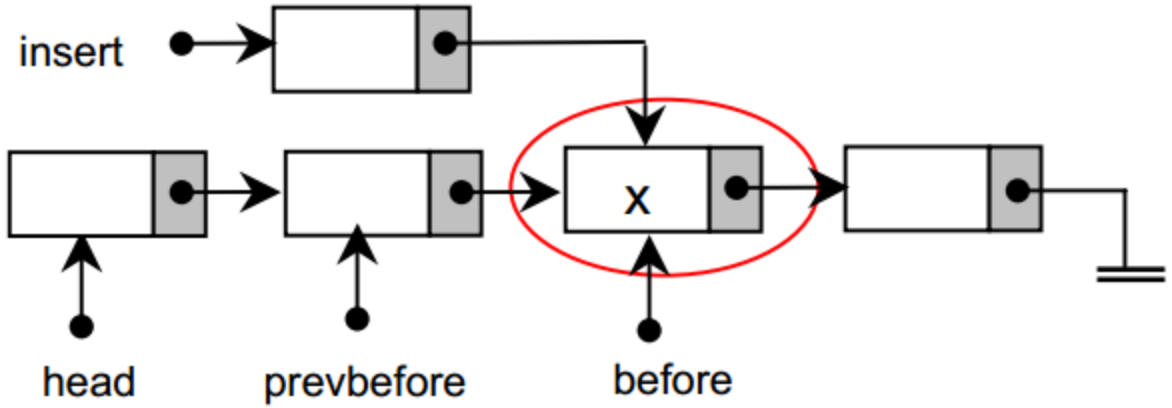
```
before = head;
```



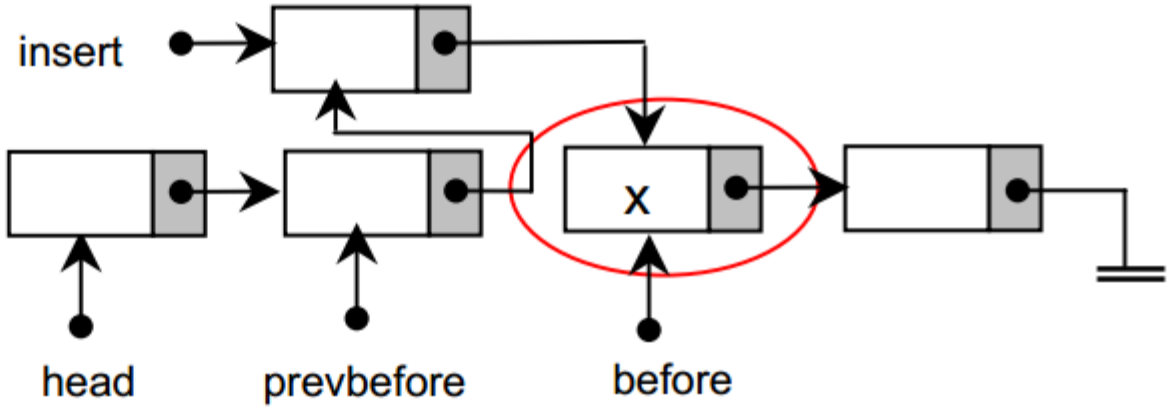

```
do
        prevbefore = before;
        before = before->next;
while (before->datalist != x);
```



```
insert->next = before;
```




```
prevbefore->next = insert;
```



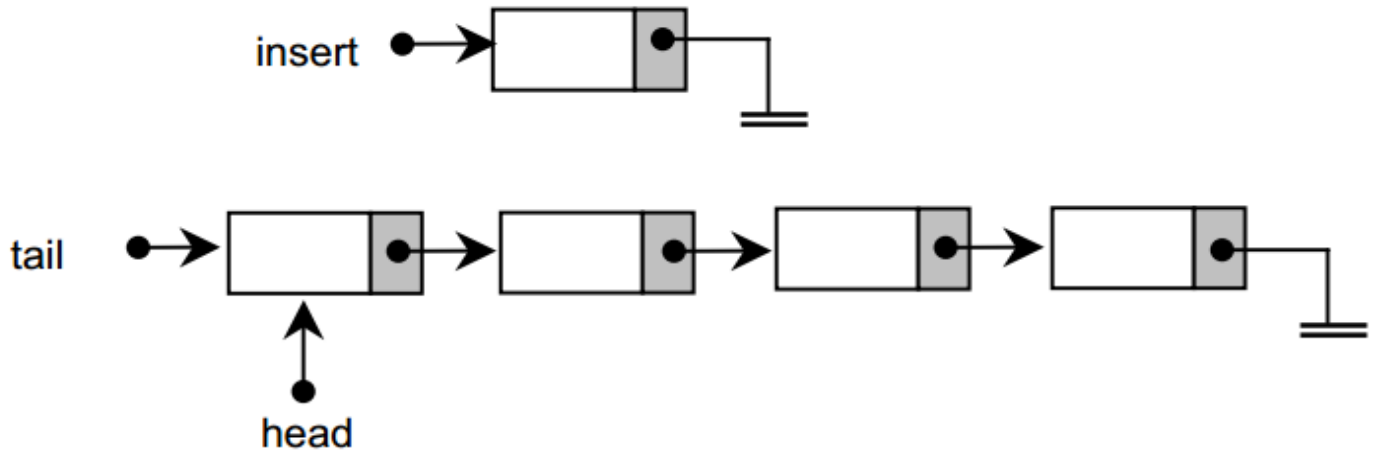
Langkah-langkah untuk proses di atas adalah sebagai berikut:

1. Telusuri list sampai elemen tertentu, catat juga elemen sebelumnya
2. Lakukan penyisipan sebelum elemen tertentu tersebut

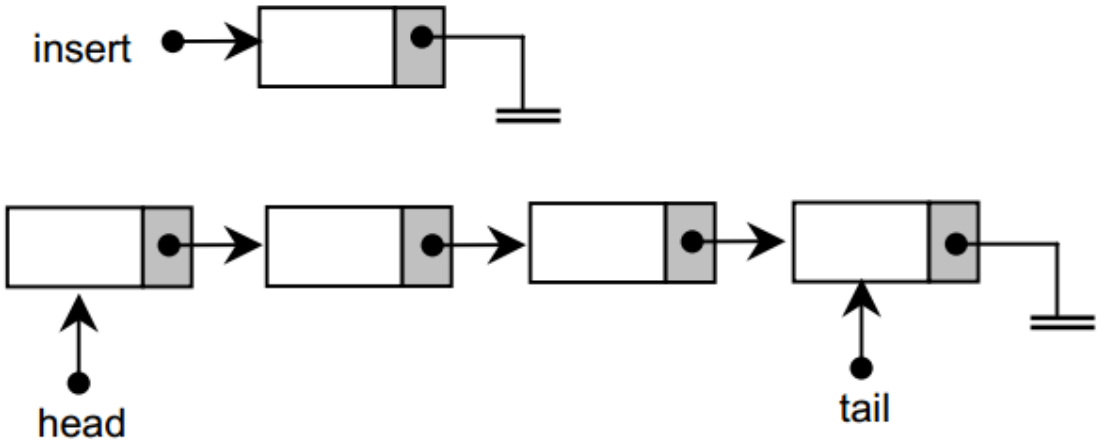
Insert di akhir (tail) dari linked list

```
void insertastail(list insert)
{
list tail;
     tail = head;
do
  tail = tail->next;
  while (tail->next != NULL);
  tail->next = insert;
  tail = tail->next;
}
```

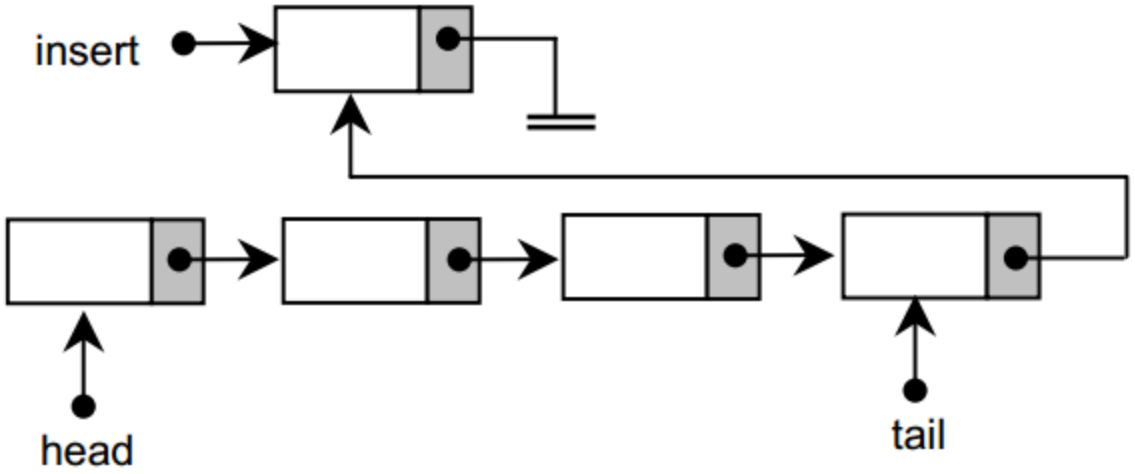
```
tail = head;
```



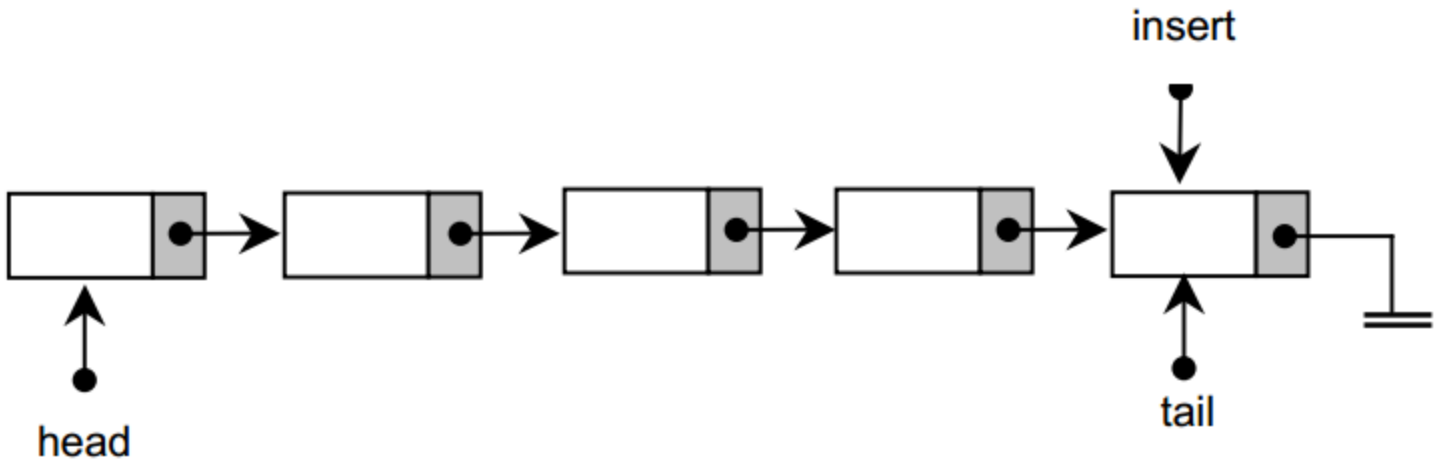

```
do
      tail = tail->next;
while (tail->next != NULL);
```



```
tail->next = insert;
```




```
tail = tail->next;
```



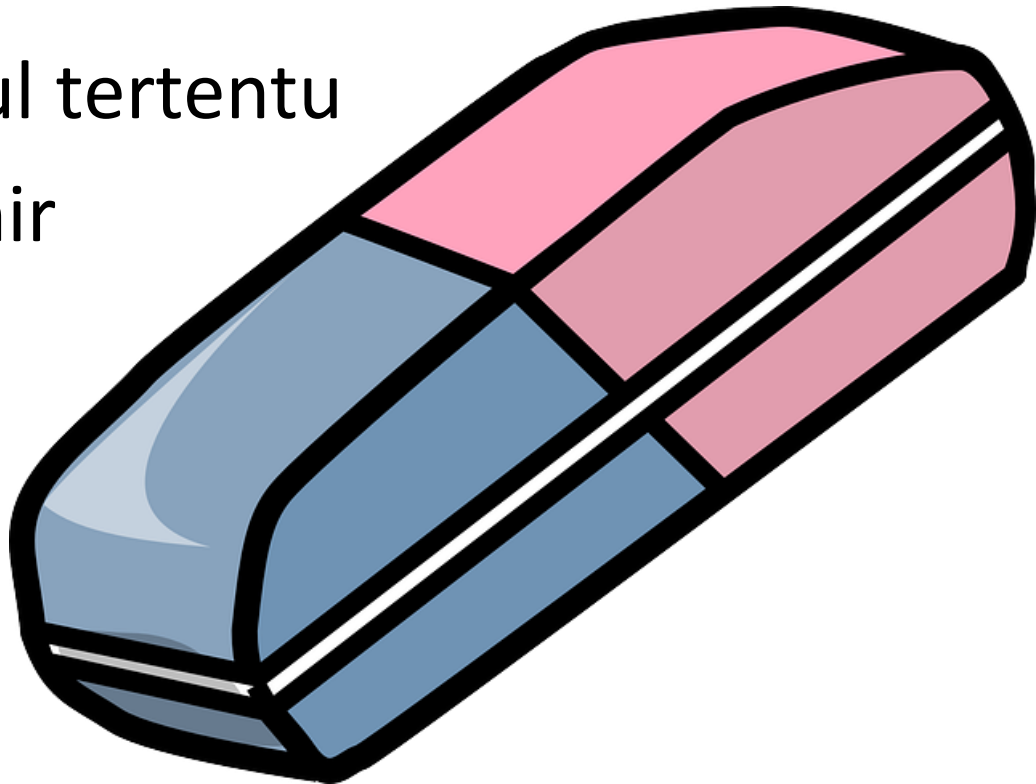
Langkah-langkah untuk proses di atas adalah sebagai berikut:

1. Telusuri list sampai elemen terakhir (tail->next=NULL)
2. Lakukan penyisipan setelah elemen terakhir

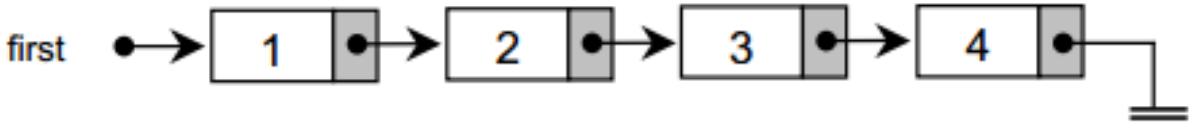
Delete

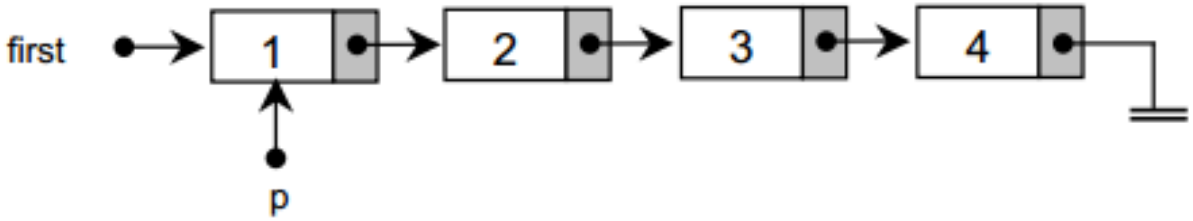
Fungsi delete pada linked list meliputi :

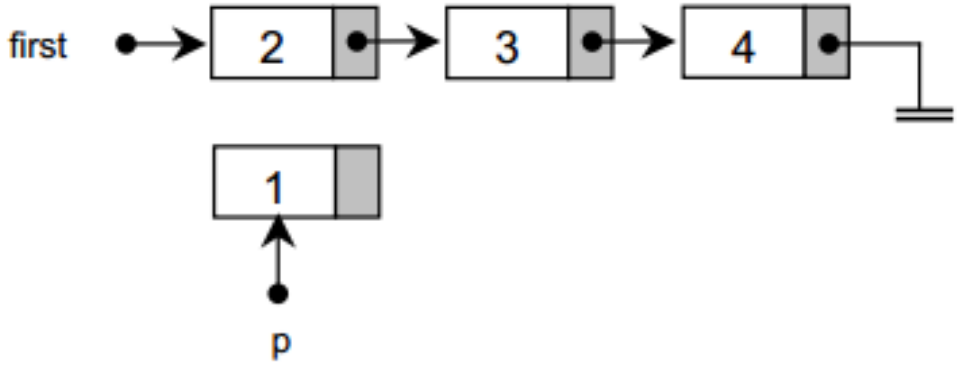
- delete sebagai simpul pertama(head) dari linked list
- delete setelah simpul tertentu
- delete simpul terakhir

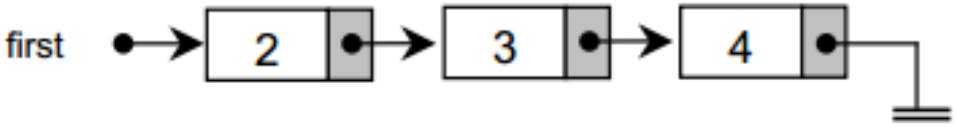


delete sebagai simpul pertama (head) dari linked list


first
1
2
3
4


first
1
2
3
4
p

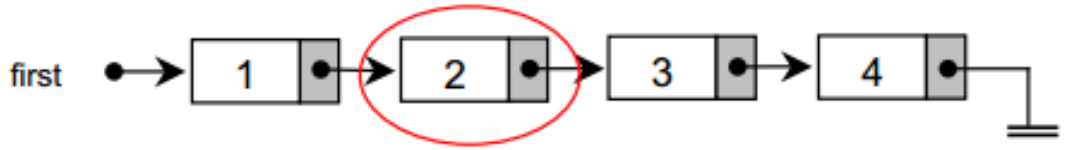

first
2
3
4
1
p

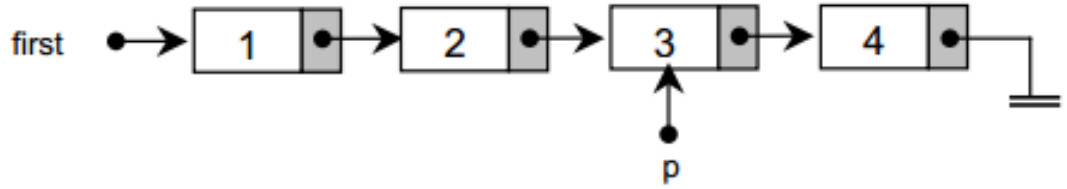

first
2
3
4

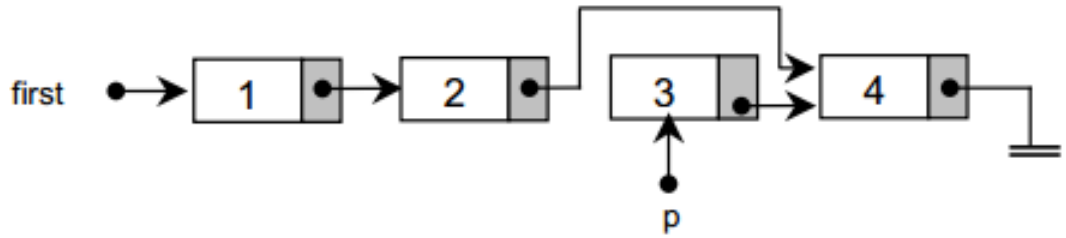
Langkah-langkah untuk proses di atas adalah sebagai berikut:

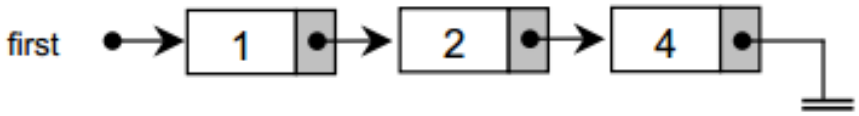
1. Pointer first diarahkan pada data ke-2
2. Pointer p diarahkan pada data ke-1
3. Bebaskan pointer p (secara otomatis data ke-1 terhapus)

delete setelah simpul tertentu


first
1
2
3
4


first
1
2
3
4
p

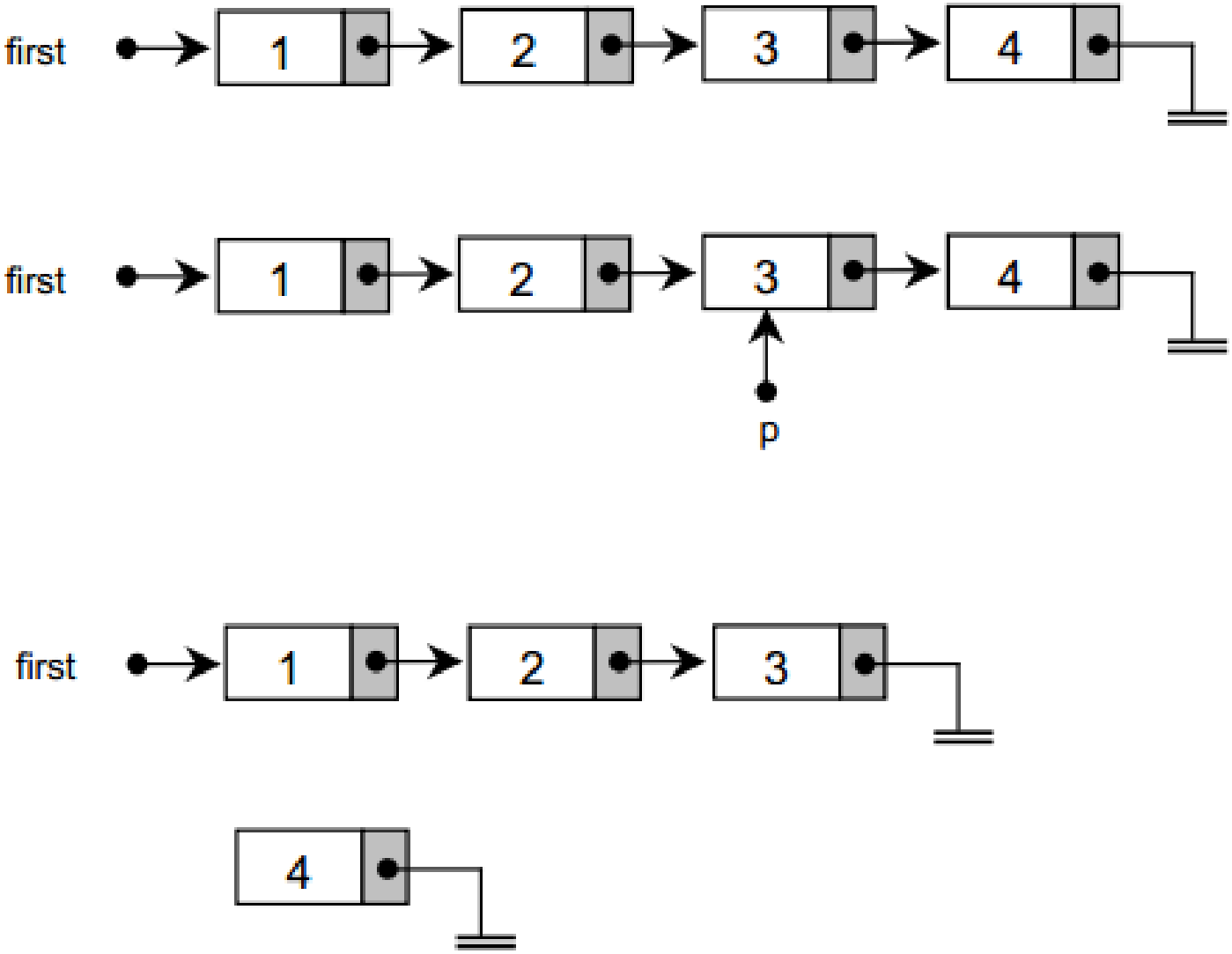

first
1
2
3
4
p


first
1
2
4

Langkah-langkah untuk proses di atas adalah sebagai berikut:

1. Arahkan pointer first s/d data yang ditunjuk
2. Pointer p diarahkan pada first->next
3. Arahkan pointer first->next pada p->next
4. Bebaskan pointer p (secara otomatis data setelah simpul tertentu terhapus)

delete simpul terakhir


first
1
2
3
4
first
1
2
3
4
p
first
1
2
3
4

Langkah-langkah untuk proses di atas adalah sebagai berikut:

1. Telusuri simpul s/d first->next = NULL
2. Arahkan pointer p ke first
3. Bebaskan pointer p->next (Simpul Terakhir)
4. Arahkan p->next ke NULL

Latihan soal

```
No. Baris
 1  #include <iostream>
 2
 3  using namespace std;
 4
 5  struct node{
 6
 7  int data;
 8  node *next;
 9  };
10
11
12  int main()
13  {
14
15      node *n;
16      node *t;
17      node *h;
18      node *bantu;
19      node *hapus;
20
21
22      n= new node;
23      n->data=2;
24      t=n;
25      h=n;
26
27      n= new node;
28      n->data=5;
```

```
No. Baris
29      t->next=n;
30      t=t->next;
31
32      n= new node;
33      n->data=8;
34      n->next=NULL;
35      t->next=n;
36      bantu=n;
37
38
39      n=new node;
40      n->data=7;
41      t->next=n;
42      n->next=bantu;
43
44      hapus = h;
45      h = hapus->next;
46      delete hapus;
47
48      t=h;
49
50       while( t!= NULL ){
51          cout << "Data : " << t->data << endl;
52          t = t->next;
53          }
54
55      return 0;
56  }
```

Tuliskan Outputnya!

Selesai