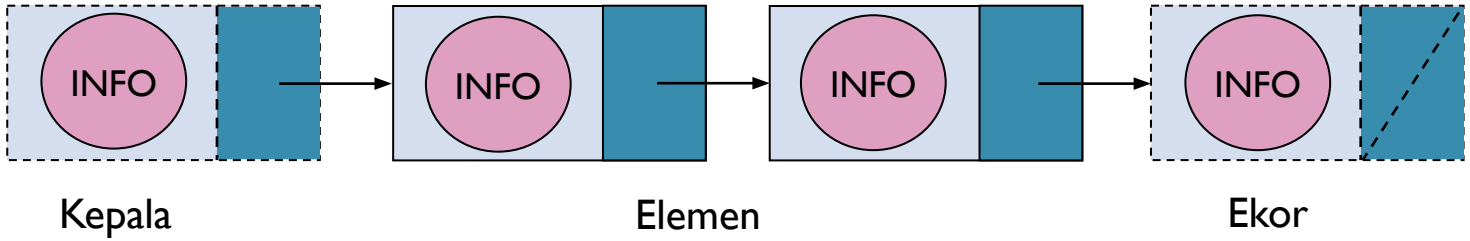
LINKED LIST BERKEPALA DAN BEREKOR

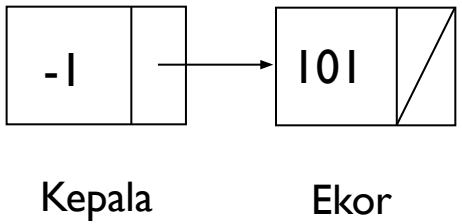
LINKED LIST KEPALA DAN BEREKOR



Kepala dan Ekor hanyalah dummy, ada untuk memudahkan penambahan dan penghapusan data

- Info Kepala diberi nilai yang pasti lebih kecil dari semua kemungkinan nilai yang ada
- Info Ekor diberi nilai yang pasti lebih besar dari semua kemungkinan nilai yang ada

LINKED LIST KOSONG

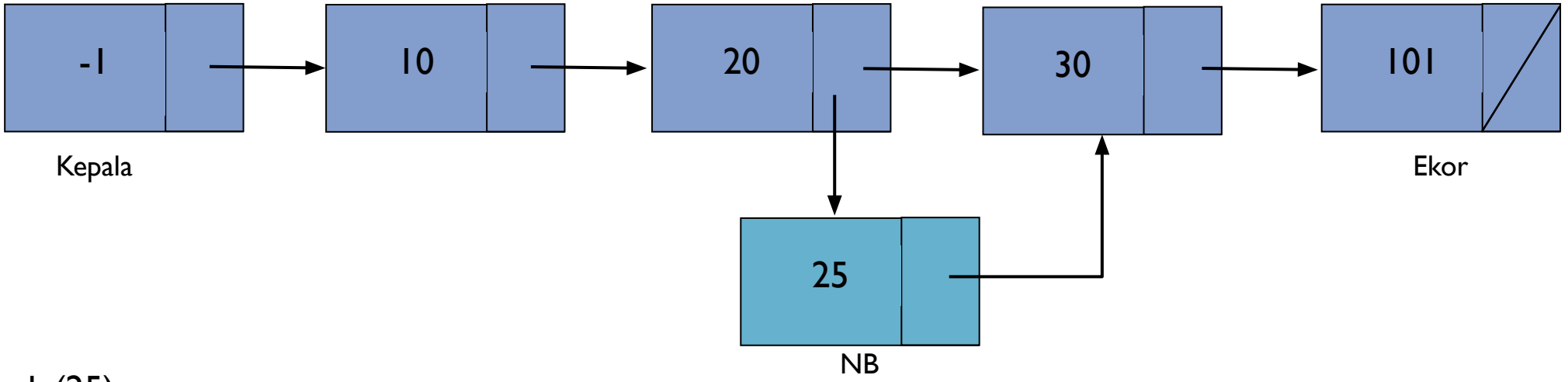


- **Linked List Kosong**
  - Jika linked list hanya terdiri dari kepala dan ekor saja, karena yang dihitung sebagai node adalah list yang berada diantara kepala dan ekor.

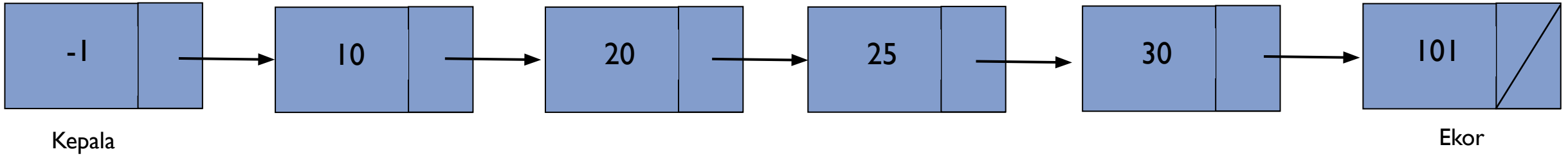
- **Keuntungan**:

  Penyisipan dan penghapusan selalu terjadi ditengah. !

SISIP NODE



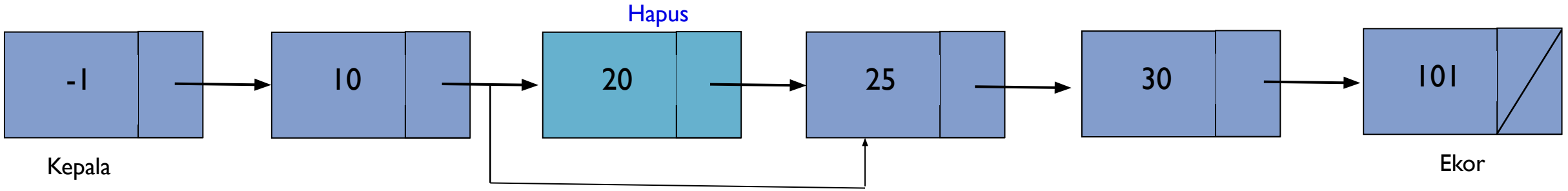
Sisipnode(25)



CONTOH CODE

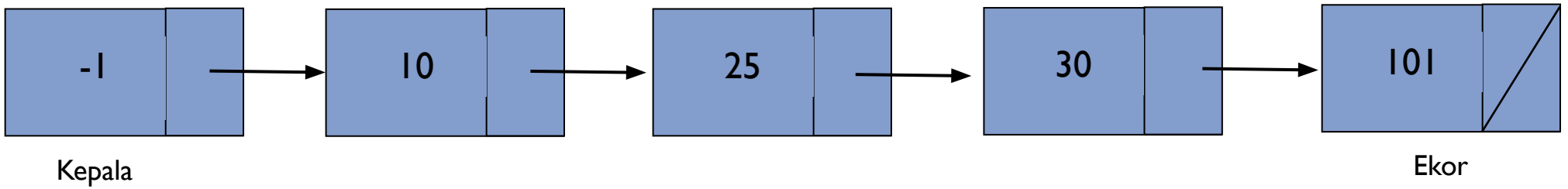
```
void sisipnode(typeinfo IB)
{
typeptr NB, bantu;
NB=(typenode *) malloc(sizeof(typenode));
NB->info=IB;
bantu=kepala;
while (IB > bantu->next->info)
bantu=bantu->next;
NB->next=bantu->next;
bantu->next=NB;
}
```

HAPUS NODE



hapusnode(20)

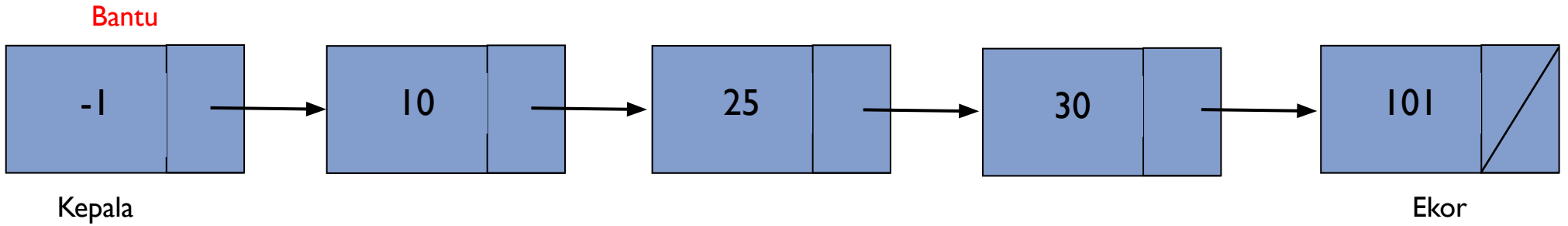
Hasil :



CODE

```
void hapusnode(typeinfo IH)
{ typeptr hapus, bantu;
   if (listkosong())
   cout << "List masih kosong";
   else
   { bantu=kepala;
   while (bantu->next!=ekor && IH!=bantu->next->info)
   bantu=bantu->next;
   if (IH==bantu->next->info)
   { hapus=bantu->next;
   bantu->next=hapus->next;
   free(hapus); }
   else
   cout << "Node tidak ditemukan!\n";}
}
```

BACA NODE (MAJU)



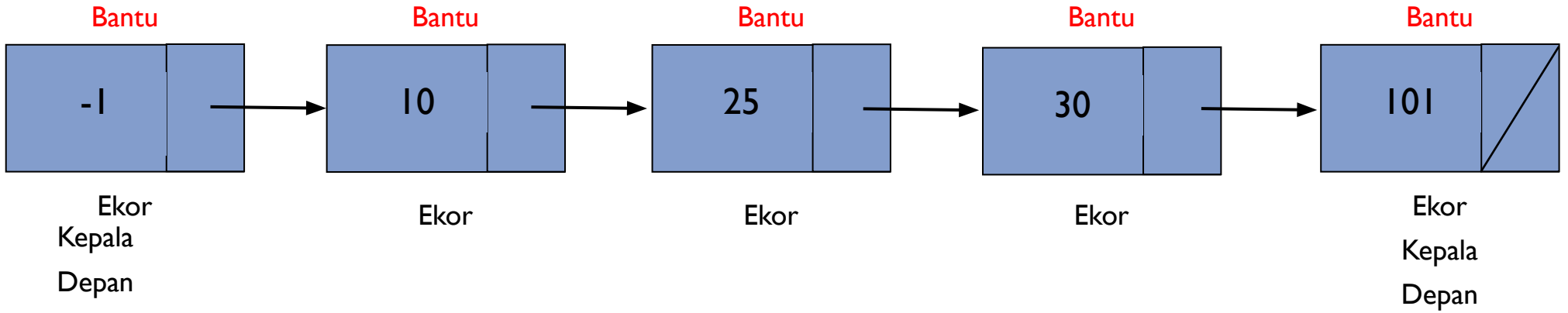
Hasil :

10
25
30

CODE

```
void bacamaju()
{
typeptr bantu;
bantu=kepala->;next;
while (bantu!=ekor)
{
cout << " " << bantu->info;
bantu=bantu->next;
}
}
```

BACA NODE (REVERSED)



Hasil :

30
25
10

CODE

```
void bacamundur()
{
typeptr depan,bantu;
depan=kepala;
kepala=ekor;
do { bantu=depan;
while (bantu->next!=ekor)
bantu=bantu->next;
ekor->next=bantu;
ekor=bantu;
} while (ekor!=depan);
ekor->next=NULL;
bantu=kepala->next;
while (bantu!=ekor)
{ cout << " " << bantu->info;
bantu=bantu->next; }
}
```