PERTEMUAN 10
LINKED LIST

A. TUJUAN PEMBELAJARAN

Mampu menjelaskan pengertian dan pembuatan *Linked List*, melakukan operasi penyisipan maupun penghapusan simpul pada *Linked List*, dan dapat mengimplementasikan *Linked List* pada bahasa pemrograman *C++*. Dimodul ini anda harus mampu :

Ketepatan dan penguasaan dalam penggunaan *Linked List* dalam membangun aplikasi pada bahasa pemrograman C++, Latihan Dan tugas.

B. URAIAN MATERI

1. Linked List

Linked List adalah struktur berupa rangkaian elemen saling berkait dimana tiap elemen dihubungkan ke elemen yang lain melalui pointer. Pointer adalah alamat elemen. Penggunaan pointer untuk mengacu elemen berakibat elemen-elemen bersebelahan secara logic walaupun tidak bersebelahan secara fisik di memori. Linked List merupakan kumpulan komponen yang saling berkaitan satu dengan yang lain melalui pointer. Masing-masing komponen sering disebut dengan simpul atau verteks.

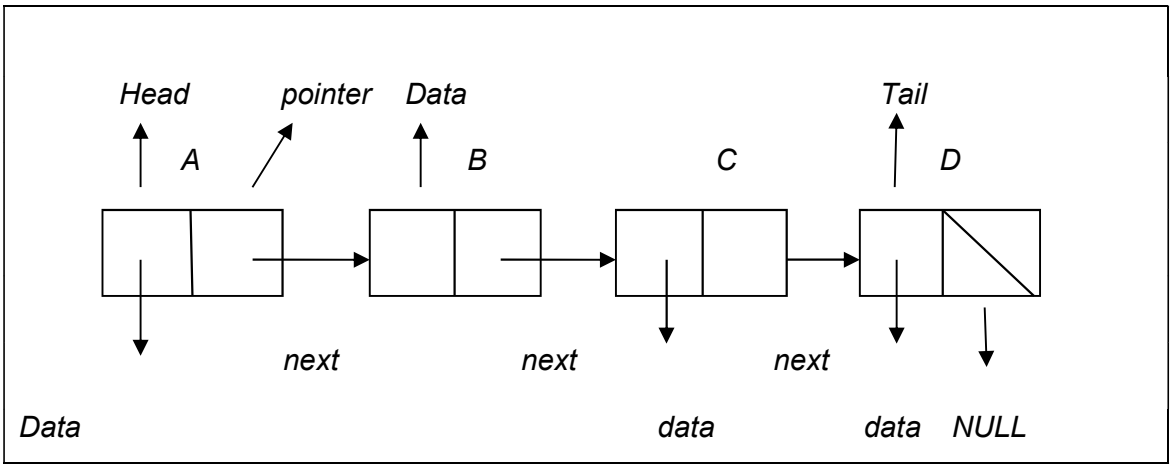
Linked List juga dapat diartikan sebagai structure atau komponen yang berangakai elemen saling terhubung ke elemen lainya melalui tipe variable pointer, komponen ini disebut simpul atau node, dan terbagi dua bagian. Bagian pertama adalah isi atau data yang disimpan oleh node. Bagian kedua disebut pointer berisi alamat node berikutnya dan sebelumnya.

Linked List ada 2 :

- *Single Linked List.*
- *Double Linked list.*

2. Single Linked List

Adalah *Linked List* contoh sederahana. Setiap node/simpul terbagi 2 : isi atau data dan pointer.

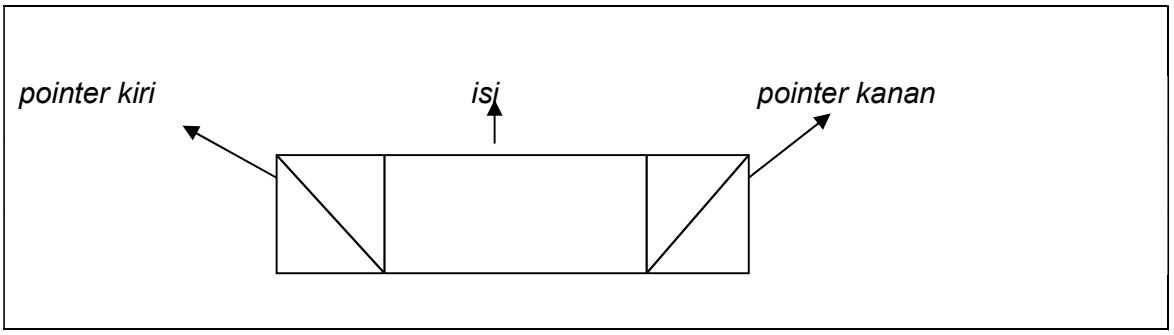


Gambar 10.10 Contoh *Single Linked List*

- 4 node : A, B, C, D
- 2 elemen :
  1. Integer.
  2. Pointer.
- Node A :
  1. Data isi.
  2. Pointer Isi alamat node 2.
- Node D :

Keterangan: Node atau simpul pertama/pertama dihubungkan pada node kedua/akhir melalui pointer sampai node akhir. Pointer akhir yang tidak terhubung ke node lainya disebut NULL.

Gambar 10.1, dapat diketahui setiap node dibagi 2 bagian, bagian isi/data dan bagian pointer.



Gambar 10.11 node double Linked List

Field : Data	field : Next
Tipe : Int/char	Tipe : Pointer
Isi : data	Isi : alamat record selanjutnya

Dengan demikian deklarasi *Single Linked List* pada *C++.*

```
// Simpul Linked List
typedef struct
struct simpul
{
        Typedata Isi ;
        Simpul->next ;
}
```

Operasi Single *Linked List* :

3. Menambah Node depan

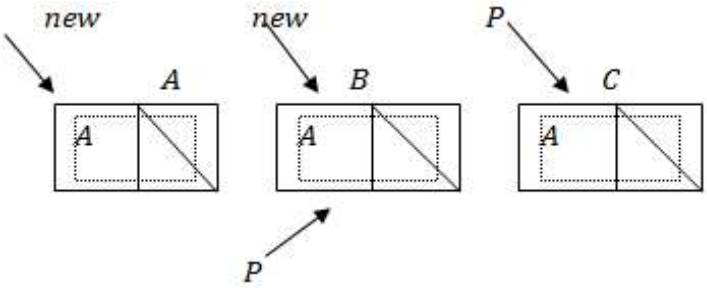
Menambah node baru dari posisi depan atau kiri Linked List.

Cara-caara sisip node baru pada posisi depan atau kiri linked list dengan langkah berikut:

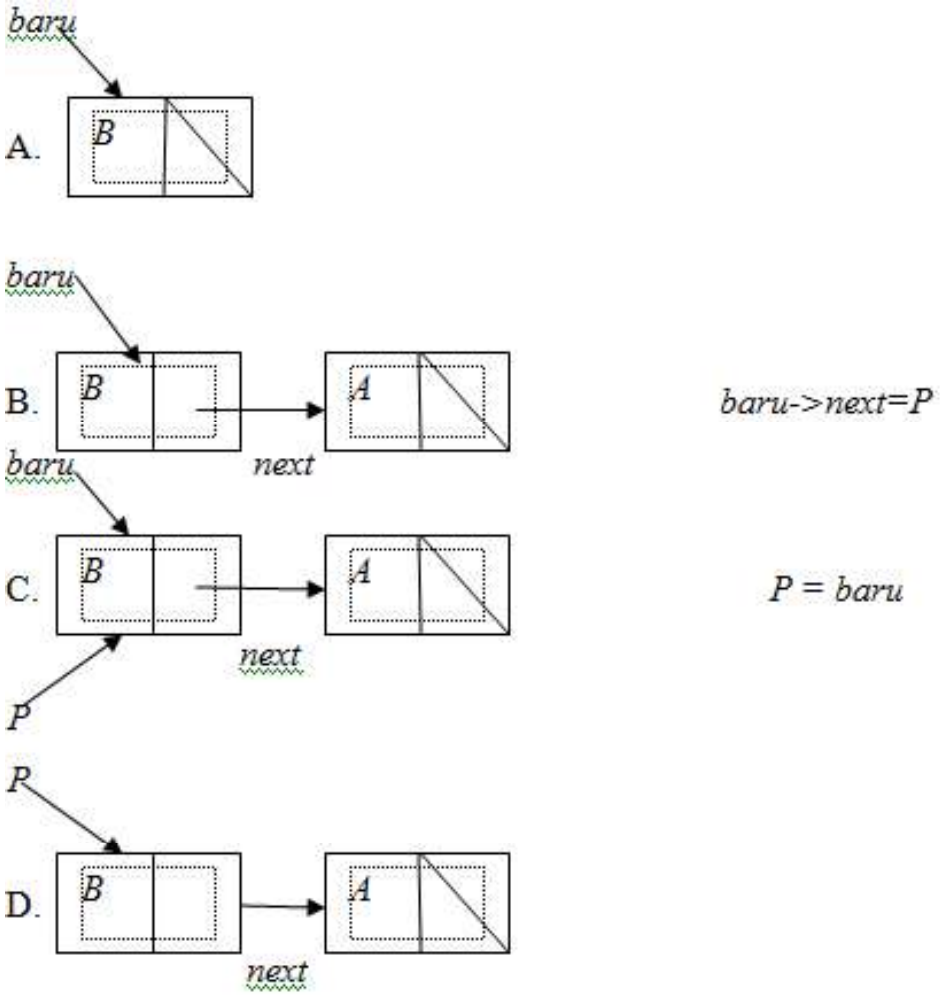
a. Membuat node baru

b. Apabila *Linked List* belum ada maka node baru menjadi *Linked List* menjadi *(P=Baru)*
c. Apabila *Linked List* ada, penyisipan dengan cara:
   - Pointer next node menunjuk *P (new->next=P)*
   - Pointer *P* pindah ke baru *(P=new)*

Contoh gamabaran penyisipan:



Gambar 10.12 penyisipan node *Linked List* belum ada.



Gambar 10.13 penyisipan node *Linked list* kiri.

Berikut langkah penyisipan node berada didepan atau kiri gambar 10.4.

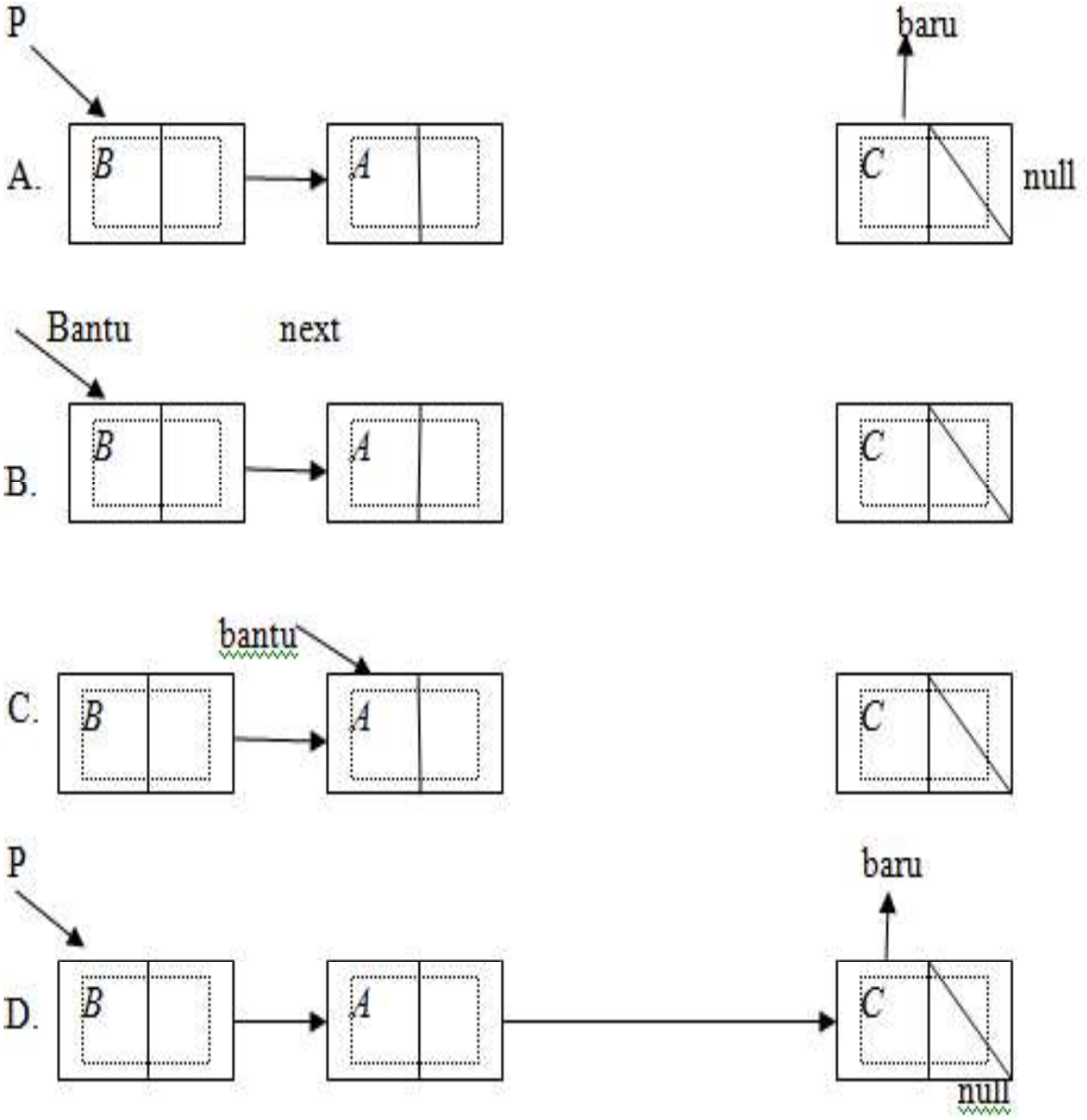
```
 //Simpul Linked List
void SIMPUL_KIRI ( simpul &P , char elemen
{
        SimpulBaru ;
        baru= (simpul)malloc(sizeof(simpul));
        baru->isi=elemen;
        baru->next=null;
        if(P==null)
                P=baru;
        Else
        {
        baru->next;
        p=baru:
}
Printf("Data Masuk\n");
}
```

4. Menambah Node Belakang

Menambahkan node baru pada posisi paling kanan atau belakang *linked list*, cara-cara pnyisipan node dapat dilakukan dengan langkah berikut ini.

a. Membuat node baru.
b. Node baru *Linked List (P=Baru)* apabila linked list belum tersedia.
c. Buat pointer yang dapat dipindahkan ke node belakang.
d. Disambungkan node baru *(bantu->next=baru)*.

Contoh gambar penyisipan:



Gambar 10.14 penyisipan node *Linked list* kanan atau dibelakang.

Berikut langkah penyisipan node berada dibelakang atau kanan gambar 10.5.

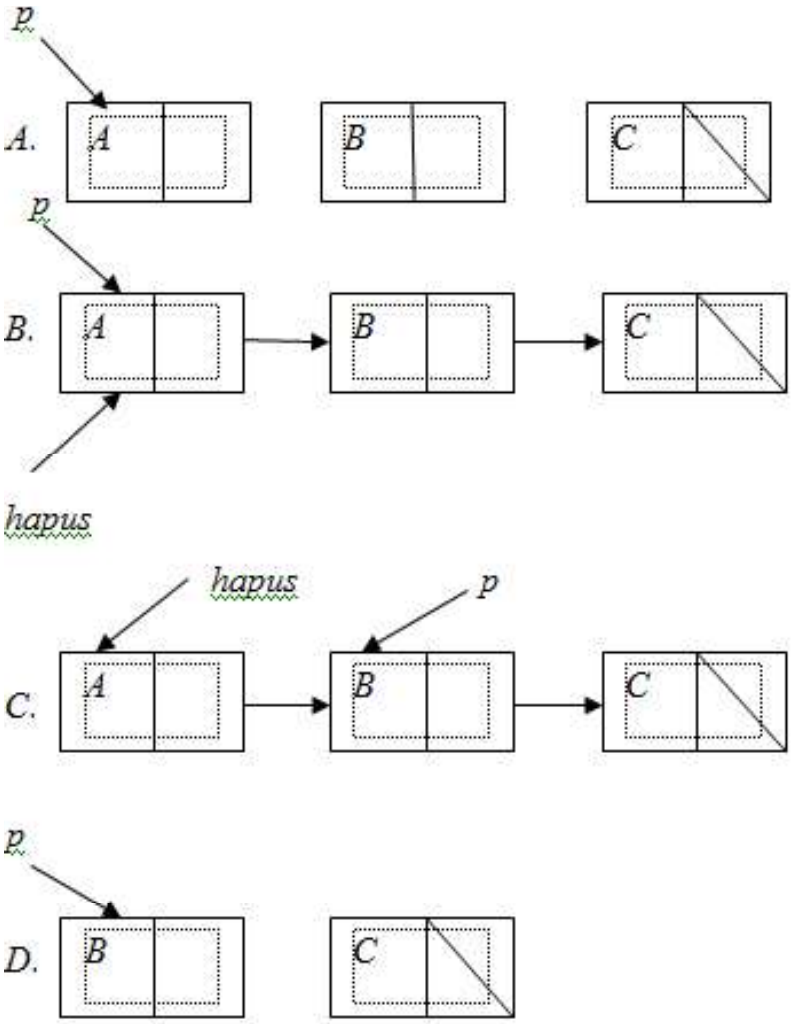
```
 //Simpul Linked List
void SIMPUL_KANAN ( simpul &P , char elemen
{
        SimpulBaru ;
        baru= (simpul)malloc(sizeof(simpul));
        baru->isi=elemen;
        baru->next=null;
        if(P==null)
                P=baru;
        else
        {
                bantu=P;
                while(bantu->next=null)
        {
                bantu=bantu->next;
                bantu->next=baru;
        }
                Printf("Data Masuk\n");
 }
```

5. Menghapus Node Depan

Menghapus node dengan dan dari Linked List, dan Linked List tidak kosong. Cara-cara hapus node bisa menggunakan dengan langkah berikut:

a. Buat pointer menunjuk node hapus
b. Node pertama di pointer hapus *(hapus=P)*
c. Pointer *P* ke node selanjutnya *(P=P->next)*
d. Putus node pertama *P (hapus->next=null)*

Contoh gambar penghapusan node kiri atau depan.



Gambar 10.15 penghapusan node *Linked list* kiri atau depan.

Berikut fungsi darl langkah penghapusan node berada didepan atau kiri gambar 10.6.

```
//Simpul Linked List
  Void HAPUS_KIRI(simpul &P)
  {
      simpul Hapus;
      if(P==null)
      count<<"Linked List Kosong">>;
      else
          {
              Hapus=P;
              P=P->next;
              Hapus->next=null;
              Free(hapus);
          }
      Printf("Data terhapus\n");
}
```

6. Menghapus Node Belakang

Menghapus node akhir atau dari kanan Linked List harus terisi, diketahui pointer P tidak boleh sampai digerakan,maka dari itu membuat pointer bantu yang bisa bergerak didalam Linked List, pointer ini mungkin tidak aka nada node putus atau hilang, dan juga membuat pointer hapus yang dapat berguna memilih node yang dihapus.

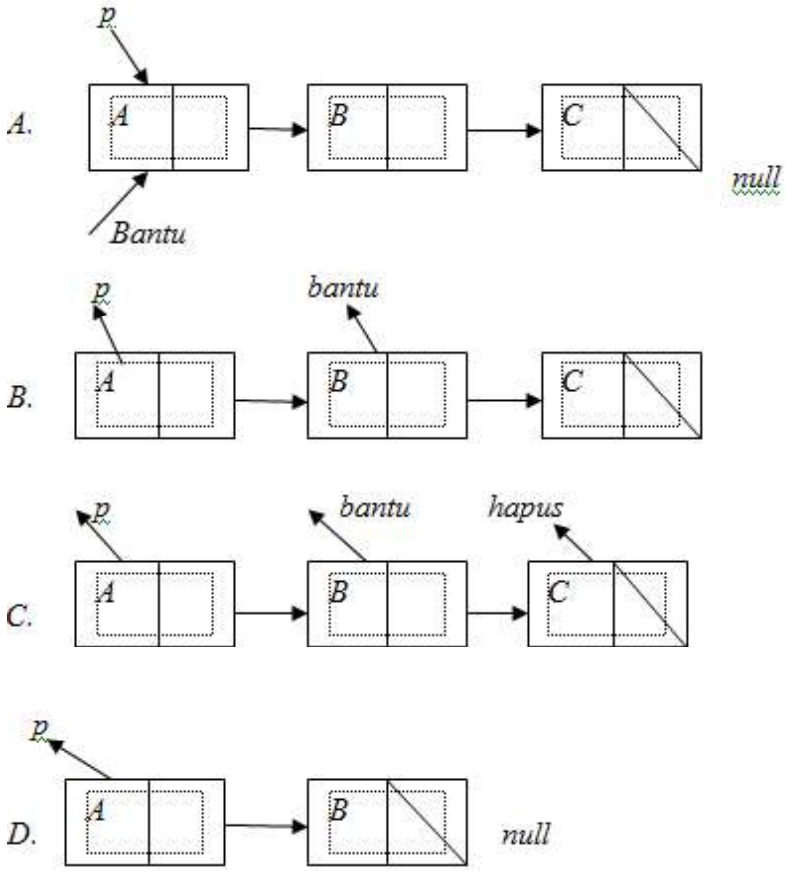
Berikut langkah-langkah:

a. Buat pointer bantu di node pertama.

b. Pindahkan pointer bantu sebelum node yang dihapus

c. Node akhir diarahkan pointer hapus

d. Putus node akhir dari *P(bantu->next=null)*



Gambar 10.16 penghapusan node *Linked list* kanan atau belakang.

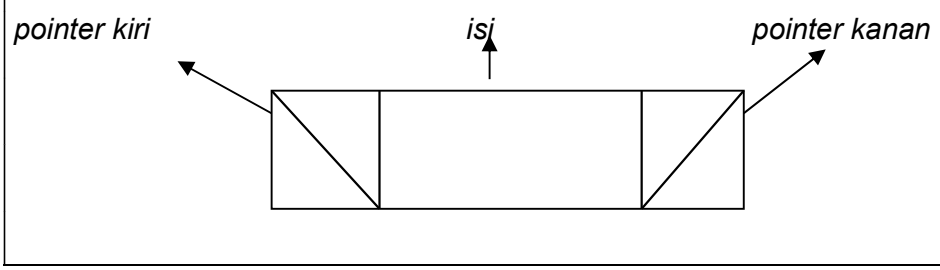
Berikut langkah penghapusan node berada dikanan atau belakang gambar 10.7.

```
//Simpul Linked List
Void HAPUS_KANAN(simpul &P)
{
        Simpul bantu,hapus;
        If(P==null)
        Count<<"Linked List Kosong">>;
  else
{
        Bantu=P;
        While(bantu->next->next!=null)
        Bantu=bantu->next;
        Hapus=bantu-next:
        Bantu->next=null;
        Free(hapus);
}
        Printf("Data terhapus\n");
}
```

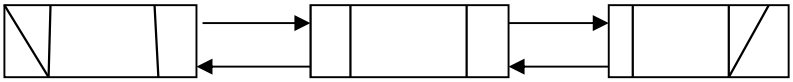
7. Double Linked List

Single Linked List mempunyai kelemahan karna hanya mempunyai satu pointer. Maka double Linked List ada untuk menutupi kelemahan single Linked List. Double Linked List terbagi menjadi 3 bagian, pointer kiri dan kanan lalu ada bagian isi di bagian tengah.

Contoh gambar 10.8 *Double Linked List.*



Gambar 10.17 node double Linked List



Gambar 10.18 node double Linked List 3 node

Dapat kita ketahui dari gambar tersebut, bahwa pointer kiri mengarah ke pointer selanjutnya dan dapat mengarah ke pointer sebelumnya, tetapi untuk pointer paling kiri di node awal dan akhir tidak mengarah ke node lain atau NULL.

Contoh gambaran deklarasi Duble linked List.

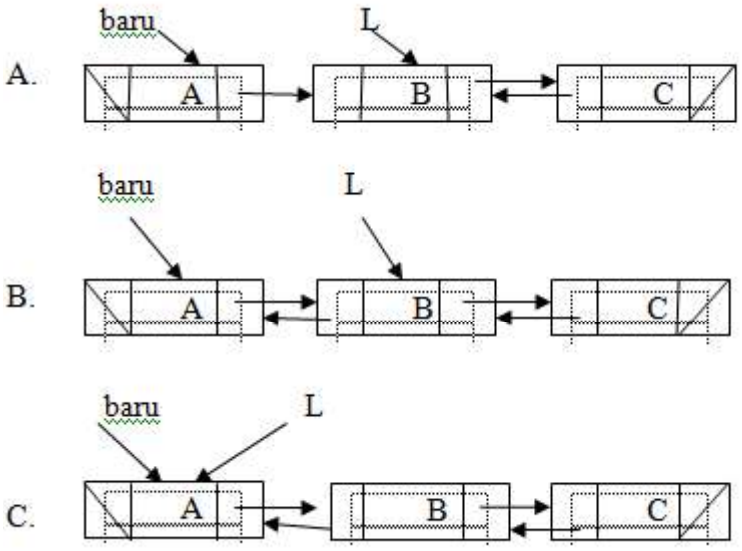
```
// Simpul Doubel Linked List
Typedef struct node;
Struct node;
        {
                Char isi;
                Node kanan;
                Node kiri;
        };
```

8. Penyisipan Node Depan

Sisip node depan bisa di kerjakan dengan cara-cara sebagai berikut:

a. Membuat node baru.
b. Linked List kosong (L=baru)
c. Apabila sudah ada Linked List,pointer kanan mengarah ke L(baru->kanan=L)
d. Dan pointer kiri L mengarah baru (L->kiri=baru).
e. Lalu pointer L digerakan mengarah node baru (L=Baru).

Contoh gambar double Linked List sisip depan:



Gambar 10.19 penyisipan node double Linked List depan.

Contoh gambaran deklarasi gambar 10.10 berikut.

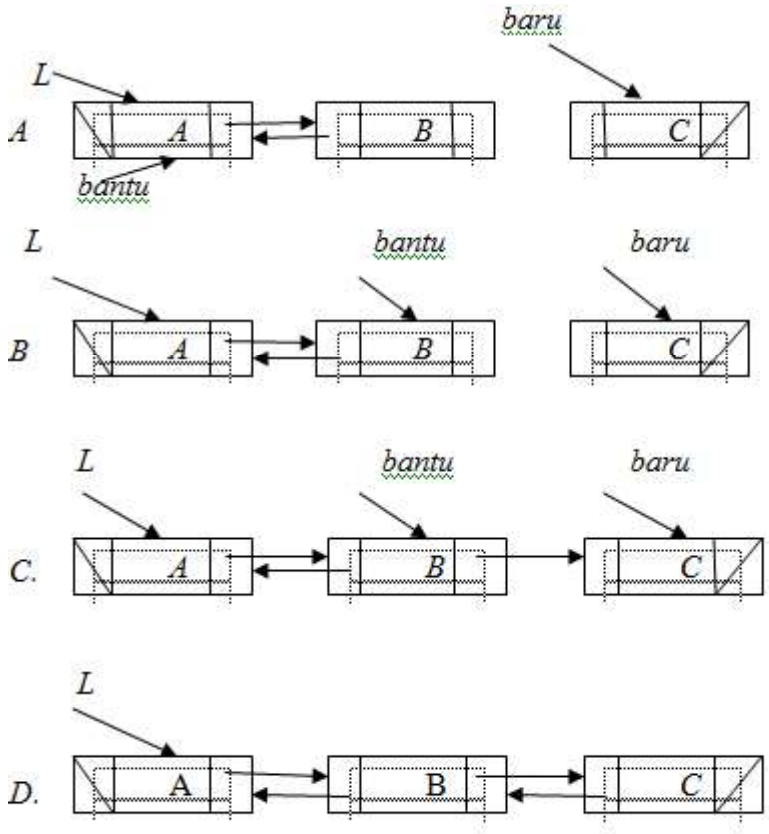
```
// Simpul Doubel Linked List
Void SISIP_KIRI(simpul&L char elemen)
{
    Simpul baru;
    Baru=(simpul)malloc(sizeof(simpul));
    Baru->isi=elemen;
    Baru->kanan=null;
    Baru->kiri=null;
    If(L==null)
    L=baru;
Else
{
    Baru->kanan=L;
    L->kiri=baru;
    L=baru;
}
}
```

9. Penyisipan Node belakang

Sisip node baru di belakang dengan atau tanpa pointer bantu.

a. Membuat node baru.
b. Apabila tidak ada double linked list (L=baru)
c. Apabila ada,ciptakan pointer bantu (bantu=L)
d. Arahakan pointer bantu sampai node akhir.
e. Lalu pointer kanan node bantu mengarah baru
f. Dan pointer kiri baru mengarah bantu.

Contoh gambaran *double Linked list* tidak kosong.



Gambar 10.20 penyisipan node *double Linked List* belakang.

Contoh gambaran deklarasi gambar 10.11 berikut.

```
// Simpul Doubel Linked List
Void SIMPUL_KANAN(simpul&L,char elemen)
Simpul baru,bantu;
Baru=(simpul)malloc(sizeof(simpul));
Baru->isi=elemen;
Baru->kanan=null;
Baru->kiri=null;
If(L==null)
L=baru;
```

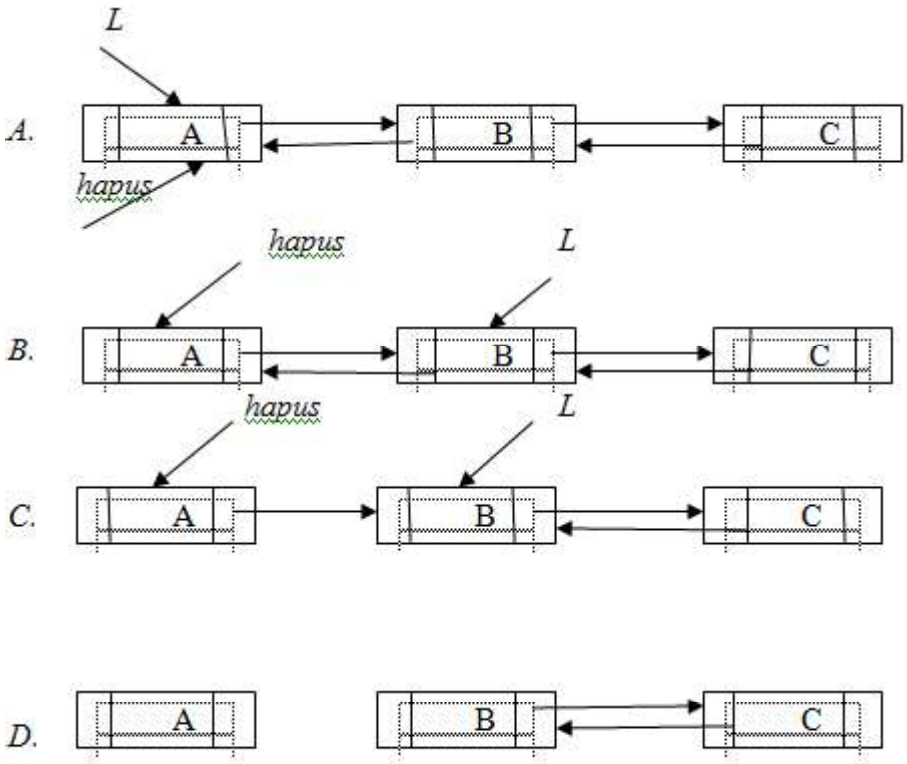
```
        Else
            {
                    Bantu=L;
                    While(bantu->kanan=null)
                    Bantu=bantu->kanan;
                    Bantu->kanan=baru;
                    Baru->kiri=bantu;
            }
        }
```

10. Hapus Node Depan Double Linked List

Adalah operasi yang dilakukan pada node depan.

a. *Node depan diarahkan oleh pointer hapus(hapus=L)*
b. *Lalu L diarahakan satu node kea rah kanan(L=L->kanan).*
c. *Selanjutnya putus pointer kiri L dari pointer hapus(L->kiri=null)*
d. *Dan putus pointer kanan L dari Linked list(hapus->kanan=null).*

Contoh gamabaran pengahapusan node awal.



Gambar 10.21 hapus penyisipan node *double Linked List* depan.

Contoh gambaran deklarasi gambar 10.12 berikut.

```
// Simpul Doubel Linked List
Void HAPUS_KIRI(simpul&L);
{
    Simpul hapus;
    If(L==null);
    Count<<"Linked list kosong":
Else
{
    Hapus=L;
```

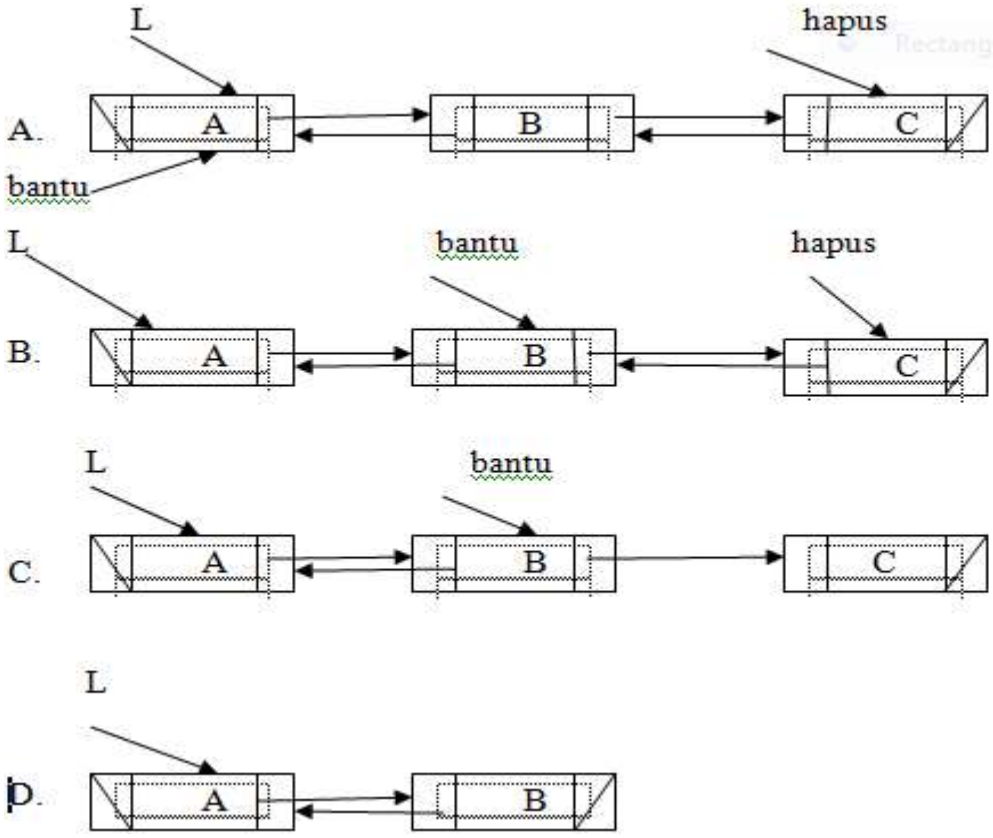
```
            L=L->kanan;
            L->kiri=null;
            Hapus->kanan=null:
            Free(hapus);
        }

    }
```

11. Hapus Node Belakang Double Linked List

Metode operasi yang dimana pengahapusan node beradi di paling akhir.contoh ada 3 node tentang informasi A,B,C dimana node akhir C akan di hapus. Carany mengunakan pointer bantu atau tidak.berikut cara caranya dengan pointer bantu:

a. Ciptakan pointer bantu mengarah node awal L(bantu=L)
b. Pindahkan pointer bantu ke sebelum node yang akan dihapus.
c. Node akhir yang dipilih pointer kanan bantu dan pointer hapus(hapus=bantu->kanan).
d. Putuskan pointer kiri dan kanan dari linked list.



Gambar 10.22 pengahapusan node belakang.

Contoh gambaran deklarasi gambar 10.13 berikut

```
// Simpul Doubel Linked List
                    Void HAPUS_KANAN(simpul&L)
                    {
                      Simpul bantu,hapus;
                      If(L==null);
                      Count<<"Linked list kosong";
                    Else
                            {
                                    Bantu=L;
                                    While(bantu->kanan->kanan=null)
                    Bantu=bantu->kanan;
```

```
Hapus=bantu->;
Bantu->kanan=null;
Bantu->kiri=null;
Free(hapus);
        }
}
```

C. SOAL LATIHAN/TUGAS PENDAHULUAN

Latihan 10

1. Apa perbedaan single Linked List dan Double Linked List?

LAPORAN AKHIR PERTEMUAN 10 (Dikumpulkan pada Pertemuan 11)

TUGAS AKHIR

1. Buat program lengkap dengan double Linked List !

LAPORAN AKHIR

1. Jelaskan program yang telah anda buat pada Tugas Akhir dengan detail dan jelas !

D. REFERENSI

1. C And Data Structure by practice by Ramesh Vasappanvara.
Single Linked List dan Double Linked List padaC++ by pintarkom
3.Struktur Data Array Stack Dan Queue | Aries Indra Gunawan