Modul Kuliah
Struktur Data

Linked List

M. Bahrul Ulum, S.kom, M.Kom

Program Studi Teknik Informatika
Fakultas Ilmu Komputer
Universitas Esa Unggul
2018

LINKED LIST

Teori

Linked List adalah suatu cara untuk menyimpan data dengan struktur sehingga programmer dapat secara otomatis menciptakan suatu tempat baru untuk menyimpan data kapan saja diperlukan. Linked list dikenal juga dengan sebutan senarai berantai adalah stuktur data yang terdiri dari urutan record data dimana setiap record memiliki field yang menyimpan alamat/referensi dari record selanjutnya (dalam urutan). Elemen data yang dihubungkan dengan link pada linked list disebut Node. Biasanya dalam suatu linked list, terdapat istilah head dan tail .

1. Head adalah elemen yang berada pada posisi pertama dalam suatu linked list
2. Tail adalah elemen yang berada pada posisi terakhir dalam suatu linked list.

Jenis Linked List (yang akan dipelajari) adalah :

1. Single Linked List
2. Double Linked List
3. Circular Linked List
4. Multiple Linked List

Proses Dasar Linked List

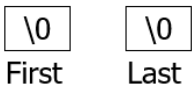
Ada 5 proses dasar dalam Linked List :

1. Proses Inisialisasi

- Proses awal → menyatakan Linked List belum ada.
- Algoritma :

```
First = Null;
Last = Null;
```

- Ilustrasi Proses :



2. Membuat simpul baru.

- Instruksi :

```
P = (simpul *) malloc(sizeof(simpul));
```

- Algoritma :

```
void Buat_Simpul(int x)
{
        P = (simpul *) malloc(sizeof(simpul));
        if (P != NULL)
        {       P -> Info = x; }
        else
                cout<<"Simpul gagal dibuat ";
}
```

3. Membuat simpul awal.

• Algoritma :

```
void Awal()
{
        First = P;
        Last = P;
        P -> Link = NULL;
}
```

Syarat :

1. Linked List belum ada.
2. Sudah ada simpul yang akan dijadikan simpul awal.

4. Menambahkan simpul baru ke dalam Linked List (INSERT)

Syarat :

1. Linked List sudah ada.
2. Sudah ada simpul yang akan ditambahkan ke Linked List.

a. Insert Kanan/Akhir

Algoritma :

```
void Ins_Akhir()
{
        Last -> Link = P;
        Last = P;
        P -> Link = NULL;
}
```

b. Insert Kiri/Awal

Algoritma :

```
void Ins_Awal()
{
        P -> Link = First;
        First = P;
}
```

c. Insert Tengah

Algoritma :

```
void Ins_Tengah()
{
        P -> Link = Q ->Link;
        Q -> Link = P;
}
```

5. Menghapus sebuah simpul dari Linked List (DELETE)

Syarat :

1. Linked List sudah ada.

a. Delete Kanan/Akhir

Algoritma :

```
void Del_Akhir()
{
        free(Last);
        Last = Q;
        Last -> Link = NULL;
}
```

b. Delete Kiri/Awal

Algoritma :

```
void Del_Awal()
{
        Q = First;
        First = Q -> Link;
        free(Q);
}
```

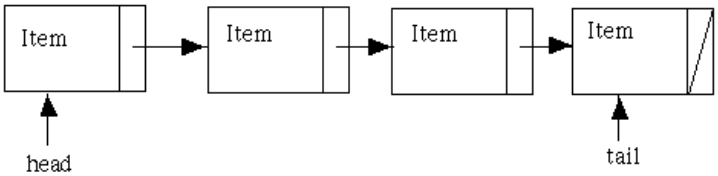
c. Delete Tengah

Algoritma :

```
void Del_Tengah()
{
        R = Q->Link ;
        Q->Link = R->Link;
        free(R);
}
```

Single Linked List

Single Linked List merupakan suatu linked list yang hanya memiliki satu variabel pointer saja. Dimana pointer tersebut menunjuk ke node selanjutnya. Biasanya field pada tail menunjuk ke NULL.



Gambar . Ilustrasi Single Linked List

Proses pembuatan Single Linked List Circular (SLLC)

Deklarasi Struct gerbong

```
typedef struct gerbong{
        int data;
        gerbong *next;
};
```

Buat juga variable pointer bertipe gerbong

```
gerbong *head;
gerbong *tail;
gerbong *baru;
gerbong *hapus;
gerbong *bantu;
```

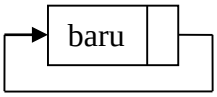
Pada main, inisialisasi head sebagai gerbong baru

```
head=new gerbong;
head=NULL;
```

Kita akan menggunakan isEmpty untuk mengecek

```
int isEmpty(){
        if (head==NULL)
                return 1;
        else
                return 0;
 }
```

Bagaimana jika penambahan data depan dan data belakang?



1. Tambah depan

```
void tambahdepan( int databaru ){
        baru=new gerbong;
        baru->data=databaru;
        baru->next=baru;

        if( isEmpty() ){
            head=tail=baru;
            head->next=head;
            tail->next=tail;
        }
        else{
            baru->next=head;
            head=baru;
            tail->next=head;
        }
        printf("data masuk\n");
}
```

Apakah ada bedanya dengan SLLNC?

2. Tambah belakang

```
void tambahbelakang( int databaru ){
        baru=new gerbong;
        baru->data=databaru;
        baru->next=baru;

        if( isEmpty() ){
            head=tail=baru;
            head->next=head;
            tail->next=tail;
        }
        else{
            tail->next=baru;
            tail=baru;
            tail->next=head;
        }
        printf("data masuk\n");
}
```

Apakah ada bedanya dengan SLLNC?

Latihan 1

Ubah fungsi tambah depan untuk data berikut: Nim, nama mahasiswa, dan ipk. Dengan Nim menambah secara otomatis. Sedangkan nama mahasiswa dan ipk diinputkan oleh user.

Bagaimana jika menampilan semua data?

```
void cetak(){
    if( !isEmpty() ){
        bantu=head;
        printf(" %i ", bantu->data);
        bantu=bantu->next;
        while( bantu!=head ){
            printf(" %i ", bantu->data);
            bantu=bantu->next;
        }
        printf("\n");
    }
    else{
        printf("Data kosong!");
    }
}
```

Untuk menampilkan semua data, kita hanya mengecek semua data sampai kembali ke data posisi awal (head). Jika telah kembali ke posisi head maka proses cetak dihentikan.

Bagaimana menghapus data depan dan data belakang?

1. Hapus depan

```
void hapusdepan(){
    int tampung;
    if( isEmpty() ){
        printf("data belum ada");
    }
    else{
        tampung=head->data;
        if(head->next!=NULL){
            hapus=head;
            head=head->next;
            tail->next=head;
            delete hapus;
        }
        else{
            head=tail=NULL;
        }
    printf("Data %i Terhapus\n", tampung);
    }
}
```

Hapus depan mengharuskan kita untuk menyambung ulang tail ke head, dengan cara ini linked list akan selalu berputar.

2. Hapus belakang

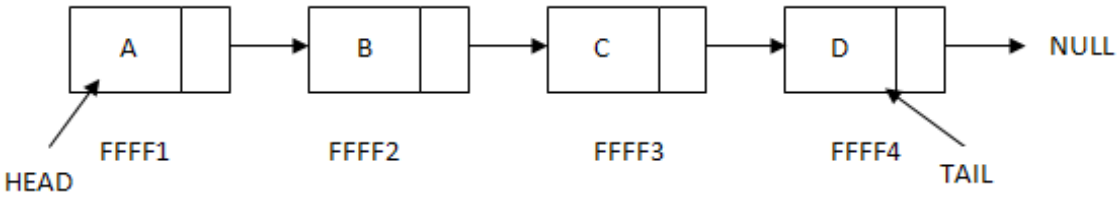
```
void hapusbelakang(){
        int tampung;
        if( isEmpty() ){
                printf("data belum ada");
        }
        else{
                tampung=tail->data;
                if(head==tail){
                        head=tail=NULL;
                }
                else{
                        bantu=head;
                        hapus=tail;
                        while(bantu->next!=tail){
                                bantu=bantu->next;
                        }
                tail=bantu;
                tail->next=head;
                delete hapus;
        }
        printf("Data %i Terhapus\n", tampung);
        }
}
```

Sama halnya dengan hapus depan, hapus belakang juga mengharuskan data sebelum tail(bantu) untuk menyambung kembali dengan head.

SINGLE LINKED LIST NON CIRCULAR

(SENARAI BERANTAI TUNGGAL TIDAK BERPUTAR)

Dilustrasikan sebagai kereta yang mempunyai gerbong-gerbong.



Pertama – tama kita membuat tipe data baru dari gerbong kereta tersebut.

```
//Gerbong berisi data dan pointer next.
typedef struct gerbong {
        int data;
        gerbong *next;
};
```

Kemudian kita membuat variable baru bertipe pointer.

```
gerbong *bantu;
gerbong *head; //Variable head selalu berada di awal rangkaian. Diibaratkan seperti lokomotif kereta api yang selalu berada didepan.
gerbong *baru; //Untuk menampung inputan data baru yang ingin dimasukkan.
gerbong *hapus; //Untuk menampung inputan data yang ingin dihapus.
gerbong *tail; //Variable head selalu berada di akhir rangkaian gerbong.
```

Selanjutnya, membuat sebuah NODE head yang diinisialisasi pada main.

```
head = new gerbong; //pembentukan node gerbong
head = NULL;
```

--- PEMBENTUKKAN NODE (GERBONG) ---

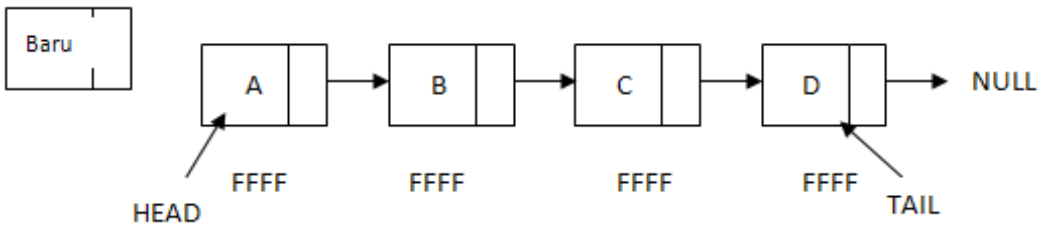
Digunakan keyword new yang berarti mempersiapkan sebuah node baru berserta alokasi memorinya, kemudian node tersebut diisi data dan pointer nextnya ditunjuk ke NULL. Pembentukan node tidak dapat dilakukan sekaligus namun harus satu persatu, hal ini berkaitan dengan bagaimana cara menyambungkan antar node tersebut. Contoh :

```
baru = new gerbong;
baru->data = databaru; //isi field data dengan databaru.
baru->next = NULL; //pointer milik baru diarahkan ke NULL
```

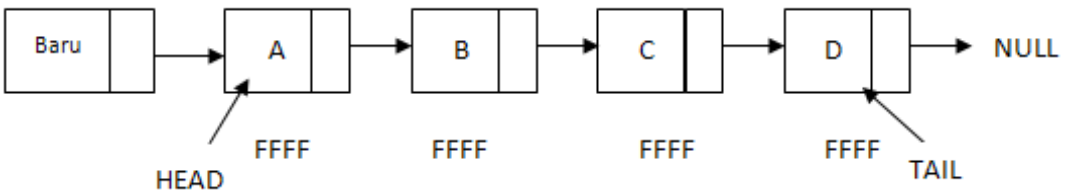
Tambah Depan

Penambahan node baru akan dikaitkan di gerbong paling depan. Tetapi jika data masih kosong, maka penambahan data dilakukan denagn cara menunjuk head pada gerbong tersebut.

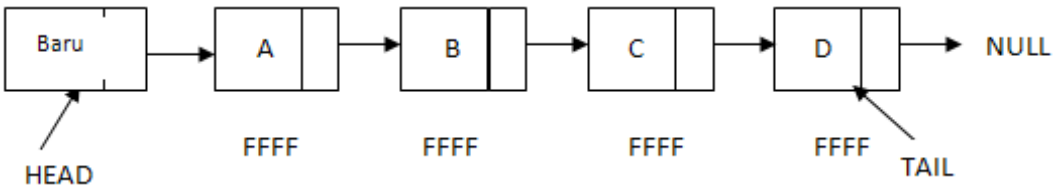
Ilustrasi Tambah Depan :



→ Data baru ditambahkan



→ Baru digandengkan dengan head



→ Head berpindah posisi menjadi di depan.

Fungsi tambah depan sebagai berikut (inputan dinamis dari user) :

```
void tambahdepan(int databaru) {
        baru = new gerbong;
        baru->data = databaru;
        baru->next = NULL; //pointer milik baru diarahkan ke NULL
        if(head == NULL) {
                head = baru;
                head->next = NULL;
                tail = baru;
                tail->next = NULL;
        }
        else {
                baru->next = head;
                head = baru;
        }
        printf("data masuk\n");
}
```

Tambah Belakang

Penambahan data dibelakang akan selalu dikaitkan dengan tail karena tail selalu berada di paling belakang gerbong. setelah dikaitkan dengan node, maka node tersebut menjadi tail yang baru.

Ilustrasi Tambah Belakang : ???

Fungsi tambah belakang sebagai berikut (inputan dinamis dari user) :

```
void tambahbelakang(int databaru) {
        baru = new gerbong;
        baru->data = databaru;
        baru->next = NULL;
        if(head == NULL) {
                head = baru;
                head->next = NULL;
                tail = baru;
                tail->next = NULL;
```

```
            }
            else {
                tail->next = baru;
                tail = baru;
            }
            printf("data masuk\n");
        }
```

Check this out …

Buatlah fungsi tambah tengah. Misalnya ada 15 gerbong, maka data baru akan ditambahkan setelah gerbong ke 7. Jika ada 10 gerbong, maka data baru akan ditambahkan setelah gerbong ke 5.

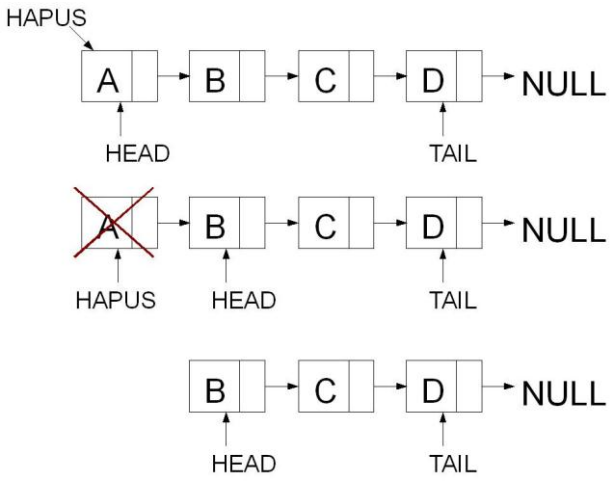
--- PENGHAPUSAN NODE ---

Pada senarai berkepala, penghapusan sebuah list dilakukan jika ada list lain yang bukan list "head" dalam barisan senarai tersebut. Dengan kata lain, list yang digunakan sebagai "head" tidak boleh dihapus, "head" harus dipindahkan terlebih dahulu. Keyword yang digunakan adalah delete.

Hapus Depan

Fungsi ini akan menghapus data terdepan yang ditunjuk oleh head. Penghapusan tidak boleh dilakukan jika keadaaanya ditunjuk oleh pointer. Oleh karena itu, dilakukan penunjukan terlebih dahulu dengan pointer hapus yang menujuk pada head. Kemudian dilakukan pergeseran node sehingga node setelah head menjadi head yang baru. Kemudian menghapus pointer hapus dengan perintah delete. Jika tail masih NULL maka data masih kosong.

Ilustrasi Hapus Depan :



Fungsi hapus depan :

```
void hapusdepan() {
        if(head == NULL) {
                printf("data belum ada");
        }
        else if(head->next!=NULL) {
                hapus = head;
                head = head->next;
                delete hapus;
                printf("Data Terhapus\n");
        }
        else {
                head = NULL;
                printf("Data Terhapus\n");
        }
}
```

Hapus Belakang

Karena adanya tail, maka penghapusan data dibelakang lebih mudah dilakukan. Penghapusan node tidak boleh dilakukan jika keadaan node sedang ditunjuk oleh pointer. Oleh karena itu dilakukan penujukan terlebih dahulu variable hapus pada tail. Kemudian dibutuhkan pointer bantu untuk membantu pergeseran dari head ke node berikutnya sampai sebelum tail, sehingga tail dapat ditunjukkan ke bantu tersebut, dan bantu tersebut akan menjadi tail yang baru. Setelah itu hapus pointer hapus dengan menggunakan perintah delete. Jika tail masih NULL maka berarti list masih kosong!

Ilustrasi Hapus Belakang : ???

Fungsi hapus belakang :

```
void hapusbelakang() {
         bantu = head;
        if(head==NULL) {
                printf("data belum ada");
        }
        else    if(head == tail) {
                head = NULL;
                printf("Data Terhapus\n");
                tail = NULL;
        }
        else    {
                hapus = tail;
```

```
                    while(bantu->next != tail) {
                        bantu = bantu->next;
                    }
                    tail = bantu;
                    tail->next = NULL;
                    delete hapus;
                    printf("Data Terhapus\n");
                }
        }
```

Check this out …

Buatlah fungsi hapus tengah.

--- PENCETAKKAN NODE ---

Program untuk mencetak :

```
        bantu=head;
        printf("\n");
        while(bantu != NULL) {
                printf(" %i ", bantu->data);
                bantu = bantu->next;
        }
        printf("\n");
```

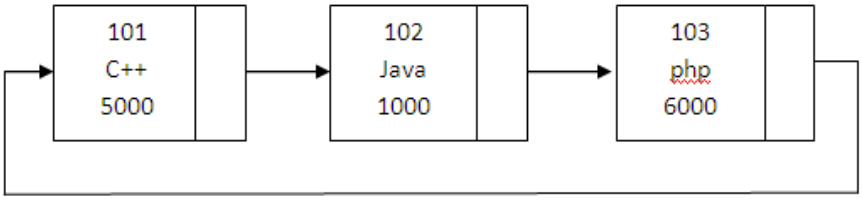
Check this out …

Bagaimana dengan fungsi search dan edit?

Latihan 2
Buat fungsi Hapus data. Data yang akan dihapus harus di inputkan oleh user.

EXERCISE

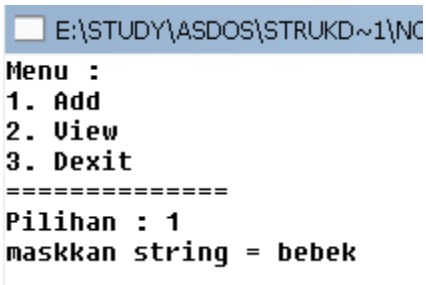
1. Buatlah fungsi tambah data yang dapat secara otomatis mengurutkan data yang diinputkan user. Data akan terurut secara ascending.
2. Buat sebuah Queue dengan menggunakan SLLC. Struck dari Queue ini berisi NoAntrian(int) dan Nama(String). Buat berupa menu yang berisi:
    1. Masuk data
    2. Liat data
    3. Keluar data
    4. Exit dari menu.
3. Buatlah SLLC untuk data buku seperti contoh dibawah ini:



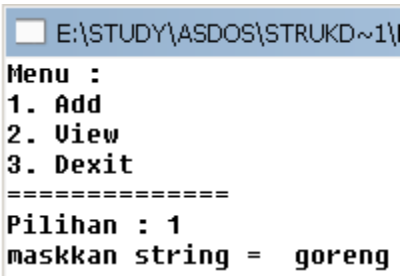
Buatlah menu untuk:

a. Menambah data buku (tambah belakang)
b. Cetak
c. Edit Harga(cari buku yang akan di edit berdasarkan nomer buku, lalu harganya saja yang diedit)
d. Exit

4. Buat converter dari array ke SLLC.
    a. Buat dalam bentuk menu (tambah data, lihat, dan exit)
    b. Tambah data merupakan sebuah string dan langsung di convert ke Link List
    c. Setiap Link List menampung sebuat karakter
    d. Setiap tambah data, tidak menghilangkan string yg diinputkan pertama. Contoh:



Inputan pertama:



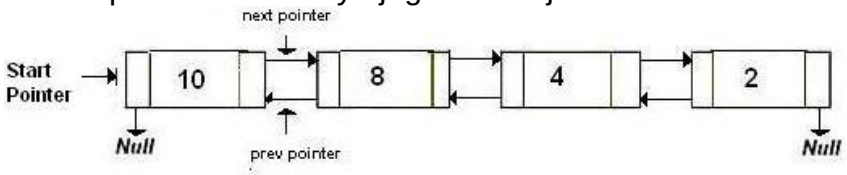
Input Kedua:

hasil:

```
E:\STUDY\ASDOS\STRUKD~1\NONAME00.EXE
Menu :
1. Add
2. View
3. Dexit
==============
Pilihan : 2
 b  e  b  e  k     g  o  r  e  n  g
```

Double Linked List

Double Linked List merupakan suatu linked list yang memiliki dua variabel pointer yaitu pointer yang menunjuk ke node selanjutnya dan pointer yang menunjuk ke node sebelumnya. Setiap head dan tailnya juga menunjuk ke NULL.



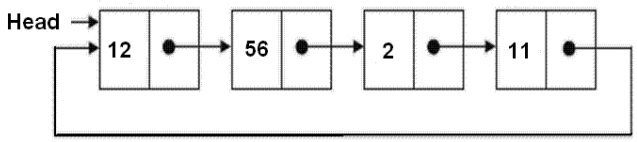
Gambar . Ilustrasi Double Linked List

Circular Linked List

Circular Linked List merupakan suatu linked list dimana tail (node terakhir) menunjuk ke head (node pertama). Jadi tidak ada pointer yang menunjuk NULL.

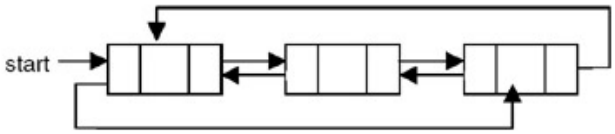
Ada 2 jenis Circular Linked List, yaitu :

1. Circular Single Linked List



Gambar . Ilustrasi Circular Single Linked List

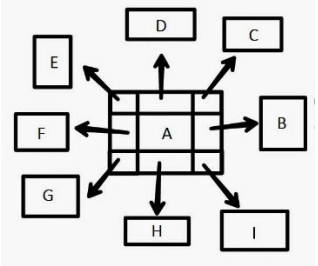
2. Circular Double Linked List



Gambar . Ilustrasi Circular Double Linked List

Multiple Linked List

Multiple Linked List merupakan suatu linked list yang memiliki lebih dar 2 buat variabel pointer.



Gambar . Ilustrasi Multiple Linked List

Pelaksanaan Praktikum

Program 1

```
#include <iostream>
#include <conio.h>
#include <cstdlib>
using namespace std;
struct data
{
        int angka;
        data *next;
}*baru, *kepala=NULL, *ekor=NULL, *tampil, *hapus, *tambah, *a, *b;

//fungsi untuk menginputkan data
void gerbong()
{
        int nilai;
        cout<<"\nMasukkan nilai : ";
        cin>>nilai;
        baru = new data;
        baru->angka = nilai;
        baru->next = NULL;
}
////////////////////////////////////

//fungsi untuk menambah depan
void tambah_depan()
{
        gerbong();
        if(kepala==NULL)
        {
         kepala = baru;
         ekor = baru;
        }
        else
```

```
            {
             baru->next = kepala;
             kepala = baru;
            }
}
//////////////////////////////////

//fungsi untuk menambah belakang
void tambah_belakang()
{
            gerbong();
            if(kepala==NULL)
            {
             kepala = baru;
             ekor = baru;
            }
            else
            {
             ekor->next = baru;
             ekor = baru;
            }
}
//////////////////////////////////

//fungsi untuk menampilkan data
void muncul()
{
            if(kepala==NULL)
            {
             cout<<"Data kosong";
            }
            else
            {
             tampil = kepala;
             while(tampil!=NULL)
             {
                    cout<<tampil->angka<<" ";
                    tampil = tampil->next;
             }
            }
}
//////////////////////////////////
```

```
//fungsi untuk menambah tengah
void tambah_tengah()
{
	int masuk;
	gerbong();
	cout<<"\nData yang ada : ";
	muncul();
	cout<<"\n\ningin dimasukkan setelah data : ";
	cin>>masuk;
	if(kepala==NULL)
	{
	 kepala = baru;
	 ekor = baru;
	}
	else
	{
	 tambah = kepala;
	 while(tambah->angka!=masuk)
	 {
		tambah = tambah->next;
	 }
	 baru->next = tambah->next;
	 tambah->next = baru;
	 if(baru->next==NULL)
	 {
		ekor=baru;
	 }
	}
}
//////////////////////////////////////////////

//fungsi untuk menghapus depan
void hapus_depan()
{
	hapus = kepala;
	if(kepala==NULL)
	{
	 cout<<"\nData kosong";
	}
	else
	{
	 kepala = kepala->next;
	 delete hapus;
```

```
        }
}
//////////////////////////////////////

//fungsi untuk menghapus belakang
void hapus_belakang()
{
        hapus = kepala;
        if(kepala==NULL)
        {
         cout<<"\nData kosong";
        }
        else if (kepala==ekor)
        {
         delete hapus;
         kepala = ekor = NULL;
        }
        else
        {
         while(hapus->next!=ekor)
         {
                hapus = hapus->next;
         }
         ekor = hapus;
         ekor->next = NULL;
         hapus = hapus->next;
         delete hapus;
        }
}
//////////////////////////////////////////

//fungsi untuk menghapus tengah
void hapus_tengah()
{
        int hps;
        cout<<"\nData yang ada : ";
        muncul();
        cout<<"\ningin menghapus nilai berapa : ";
        cin>>hps;
        hapus = kepala;
        if(kepala==NULL)
        {
         cout<<"\nData kosong";
```

```
            }
            else if (hapus->angka==hps)
            {
             hapus_depan();
            }
            else
            {
             while(hapus->angka!=hps)
             {
                    a = hapus;
                    hapus = hapus->next;
                    b = hapus->next;
             }
             a->next = b;
             delete hapus;
            }
}
//////////////////////////////////////////

int main()
{
            int pilih;
            do
            {
             cout<<"\n1. tambah depan";
             cout<<"\n2. tambah belakang";
             cout<<"\n3. tambah tengah";
             cout<<"\n4. hapus depan";
             cout<<"\n5. hapus belakang";
             cout<<"\n6. hapus tengah";
             cout<<"\n7. tampil";
             cout<<"\n8. keluar...";
             cout<<"\nMasukkan pilihan : ";
             cin>>pilih;
             if (pilih==1)
             {
                    tambah_depan();
             }
             else if (pilih==2)
             {
                    tambah_belakang();
             }
             else if (pilih==3)
```

```
            {
                    tambah_tengah();
                    getch();
            }
            else if (pilih==4)
            {
                    hapus_depan();
            }
            else if (pilih==5)
            {
                    hapus_belakang();
            }
            else if (pilih==6)
            {
                    hapus_tengah();
            }
            else if(pilih==7)
            {
                    muncul();
                    getch();
            }
            else
            {
                    exit(1);
            }
            system("cls");
           }
           while(1);
 }
```

Latihan

1. Jelaskan maksud program dalam pelaksanaan praktikum diatas (pilih salah satu) !
2. Buatlah sebuah program linked list (metode bebas) yang berisikan menu sebagai berikut :
   a. Insert depan
   b. Insert tengah
   c. Insert belakang
   d. Delete depan
   e. Delete tengah
   f. Delete belakang
   g. Tampilkan data
   h. Keluar