PRAKTIKUM 9

DOUBLY LINKED LIST

A. Tujuan

Setelah praktikum ini, praktikan diharapkan dapat:

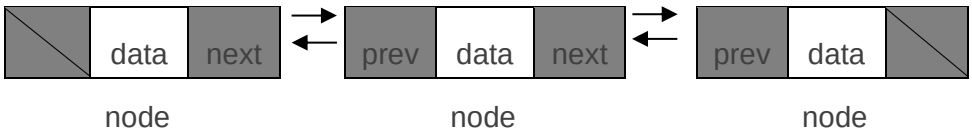
1. Memahami tipe data dasar
2. Memahami tipe data bentukan
3. Stuktur Program menggunakan bahasa C++

B. Peralatan

1. PC Desktop
2. Windows 7
3. Notepad++ dan MinGW atau Dev C++

C. Teori

- List berpointer ganda atau metode *doubly linked list* hadir untuk mengatasi kelemahan *single linked list* yang hanya dapat bergerak satu arah saja, maju/mundur, atau kanan/kiri saja.
- Terdiri dari 3 bagian, yaitu untuk menyimpan nilai dan dua reference yang menunjuk ke node. Node dirangkai dengan 2 link yang dapat mempermudah pengaksesan *successor node* (*next node*) dan *predecessor node (previous node*) dari sembarang node.
- Setiap node memiliki 2 variabel pointer yaitu yang menunjuk node kirinya (*previous*) dan yang menunjuk ke node kanannya (*next*). Ini berguna untuk melakukan penelusuran dengan arah *forward* dan *backward*.



Gambar 9. 1 Ilustrasi doubly linked list

Keberadaan 2 pointer penunjuk (*next* dan *prev*) menjadikan *Doubly Linked List* menjadi lebih fleksibel dibandingkan *Singly Linked List,* namun membutuhkan memori tambahan dengan adanya pointer tambahan tersebut. *Doubly Linked List* mempunyai reference ***front*** untuk menandai awal node dan reference ***back***
untuk menandai akhir list

Pembacaan pada Doubly Linked List
Doubly Linked List dapat dibaca melalui dua arah, yaitu:

- Pembacaan maju (*forward scan*) yaitu membaca *doubly linked list* dimulai dari reference front dan berakhir pada reference back.

- Pembacaan mundur (*backward scan*) yaitu membaca *doubly linked list* dimulai dari reference back dan berakhir pada reference front.

D. PRAKTIKUM

```
/*  ================================
  == PROGRAM DOUBLY LINKED LIST==
  ================================ */

#include<iostream>
#include<conio.h>
#include<stdlib.h>

using namespace std;

typedef struct node *simpul;
struct node
{
  char Isi;
  simpul kanan;
  simpul kiri;
};
//======================
//==Prototype Function==
//======================
void Sisip_Depan(simpul &DL, char elemen);
void Sisip_Belakang(simpul &DL, char elemen);
void Sisip_Tengah1(simpul &DL, char elemen1, char elemen2);
void Sisip_Tengah2(simpul &DL, char elemen1, char elemen2);
void Hapus_Depan(simpul &DL);
void Hapus_Belakang(simpul &DL);
void Hapus_Tengah(simpul &DL, char elemen);
void Cetak(simpul DL);

//==================
//==FUNCTION Main==
//==================
main()
{
  char huruf, huruf2;
  simpul DL = NULL;  //Pastikan bahwa DL kosong
  int i,n;
  cout<<"\t == OPERASI PADA DOUBLY LINKED LIST ==\n\n";
```

Percobaan 1 - Penyisipan Simpul di Depan

```
40    //===============
41    //==Sisip Depan==
42    //===============
43    cout<<"====          Percobaan 1       ====<<endl;
44    cout<<"==== Penyisipan Simpul Di Depan ====<<endl<<endl;
45    cout<<"Masukkan jml data : ";cin>>n;
46    cout<<endl;
47    for(i=1; i<=n; i++)
48    {
49      cout<<"Masukkan Huruf :"; cin>>huruf;
50      Sisip_Depan(DL, huruf);
51    }
52    Cetak(DL);
```

Percobaan 2 - Penyisipan Simpul di Belakang

```
54    //==================
55    //==Sisip Belakang==
56    //==================
57    cout<<"\n\n====           Percobaan 2         ====<<endl;
58      cout<<"==== Penyisipan Simpul Di Belakang ====<<endl<<endl;
59
60    cout<<"Masukkan jml data : ";cin>>n;
61    cout<<endl;
62
63    for(i=1; i<=n; i++)
64    {
65      cout<<"Masukkan Huruf :"; cin>>huruf;
66      Sisip_Belakang(DL, huruf);
67    }
68    Cetak(DL);
```

Percobaan 3.a - Penyisipan Simpul di Tengah – Setelah Simpul Tertentu

```
70    //=======================================
71    //==Sisip Simpul Setelah Simpul Tertentu==
72    //=======================================
73    cout<<"\n\n====                        Percobaan 3                ====<<endl;
74      cout<<"==== Penyisipan Simpul Di Tengah Sebelum Node Tertentu ====<<endl<<endl;
75    cout<<"Masukkan Huruf yg disisipkan  :";
76    cin>>huruf;
77    cout<<"Disisip Setelah Huruf : " ; cin>>huruf2;
78    cout<<huruf<<" Disisip Setelah "<<huruf2<<endl;
79    Sisip_Tengah1(DL, huruf, huruf2);
80    Cetak(DL);
```

Percobaan 3.b - Penyisipan Simpul di Tengah – Sebelum Simpul Tertentu

```
82      //=======================================
83      //==Sisip Simpul Sebelum Simpul Tertentu==
84      //=======================================
85       cout<<"\n\n====                      Percobaan 4                    ====";<<endl;
86          cout<<"==== Penyisipan Simpul Di Tengah Setelah Node Tertentu ====";<<endl<<endl;
87      cout<<"Masukkan Huruf  :";
88      cin>>huruf;
89      cout<<"Disisip Sebelum Huruf :";
90      cin>>huruf2;
91      cout<<huruf<<" Disisip Sebelum "<<huruf2<<endl;
92      Sisip_Tengah2(DL, huruf, huruf2);
93      Cetak(DL);
```

Percobaan 4 - Penghapusan Simpul di Depan

```
95      //======================
96      //==Hapus Simpul Depan==
97      //======================
98      cout<<"\n\n====   Percobaan 5      ====";<<endl;
99          cout<<"==== Hapus Simpul Depan ====";<<endl<<endl;
100     cout<<"\nSetelah Hapus Simpul Depan \n";
101     Hapus_Depan(DL);
102     Cetak(DL);
```

Percobaan 5 - Penghapusan Simpul di Belakang

```
104     //==========================
105     //==Hapus Simpul Belakang==
106     //==========================
107     cout<<"\n\n====   Percobaan 6      ====";<<endl;
108         cout<<"==== Hapus Simpul Belakang ====";<<endl<<endl;
109     cout<<"\nSetelah Hapus Simpul Belakang ";<<endl;
110     Hapus_Belakang(DL);
111     Cetak(DL);
```

Percobaan 6 - Penghapusan Simpul di Tengah

```
113     //=======================
114     //==Hapus Simpul Tengah==
115     //=======================
116     cout<<"\n\n====   Percobaan 7      ====";<<endl;
117     cout<<"==== Hapus Simpul Tengah ====";<<endl<<endl;
118     cout<<"\n\nMasukkan Huruf Tengah Yang Akan Dihapus :";
119     cin>>huruf;
120     Hapus_Tengah(DL,huruf);
121     Cetak(DL);
122     getch();
123 }
```

Jangan lupa untuk melengkapi fungsi-fungsi yang dipanggil pada kode program diatas, berikut fungsi-fungsinya :

```
//***********************************
//**FUNCTION SISIP SIMPUL DI DEPAN**
//***********************************
void Sisip_Depan(simpul &DL, char elemen)
{
  simpul baru;
  baru = (simpul) malloc(sizeof(simpul));
  baru->Isi = elemen;
  baru->kanan = NULL;
  baru->kiri = NULL;
  if(DL == NULL)
    DL=baru;
  else
  {
    baru->kanan = DL;
    DL->kiri = baru;
    DL=baru;
  }
}
```

```
//***********************************
//**FUNCTION SISIP SIMPUL DI BELAKANG**
//***********************************
void Sisip_Belakang(simpul &DL, char elemen)
{
  simpul bantu, baru;
  baru = (simpul) malloc(sizeof(simpul));
  baru->Isi = elemen;
  baru->kanan = NULL;
  baru->kiri = NULL;
  if(DL == NULL)
    DL=baru;
  else
  {
    bantu=DL;
    while(bantu->kanan != NULL)
      bantu=bantu->kanan;
    bantu->kanan=baru;
    baru->kiri = bantu;
  }
}
```

```
//*****************************************************
//**FUNCTION SISIP SIMPUL SETELAH SIMPUL TERTENTU**
//*****************************************************
void Sisip_Tengah1(simpul &DL, char elemen1, char elemen2)
{
  simpul bantu, baru;
  baru = (simpul) malloc(sizeof(simpul));
  baru->Isi = elemen1;
  baru->kanan = NULL;
  baru->kiri = NULL;
  if(DL == NULL)
    cout<<"List Kosong ................"<<endl;
  else
    {
      bantu = DL;
      while(bantu->Isi != elemen2) bantu=bantu->kanan;
      baru->kanan = bantu->kanan;
      baru->kiri = baru;
      baru->kiri = bantu;
      bantu->kanan->kiri = baru;
      bantu->kanan = baru;
    }
}


//*****************************************************
//**FUNCTION SISIP SIMPUL SEBELUM SIMPUL TERTENTU**
//*****************************************************
void Sisip_Tengah2(simpul &DL, char elemen1, char elemen2)
{
  simpul bantu, baru;
  baru = (simpul) malloc(sizeof(simpul));
  baru->Isi = elemen1;
  baru->kanan = NULL;
  baru->kiri = NULL;
  if(DL == NULL)
    cout<<"List Kosong ............."<<endl;
  else
  {
    bantu = DL;
    while(bantu->kanan->Isi != elemen2) bantu=bantu->kanan;
    baru->kanan = bantu->kanan;
    baru->kiri = bantu;
    bantu->kanan->kiri = baru;
    bantu->kanan = baru;
  }
}
```

```
//************************************
//**FUNCTION MENCETAK ISI LINKED LIST**
//************************************
void Cetak(simpul DL)
{
  simpul bantu;
  if(DL==NULL)
    cout<<"Linked List Kosong ........."<<endl;
  else
  {
    bantu=DL;
    cout<<"Isi Linked List : ";
    while (bantu->kanan != NULL)
    {
      cout<<bantu->Isi<<" <--> ";
      bantu=bantu->kanan;
    }
    cout<<bantu->Isi;
  }
}


//*******************************
//**FUNCTION HAPUS SIMPUL DEPAN**
//*******************************
void Hapus_Depan(simpul &DL)
{
  simpul Hapus;
  if(DL==NULL)
    cout<<"Linked List Kosong ............";
  else
  {
    Hapus = DL;
    DL = DL->kanan;
    DL->kiri = NULL;
    Hapus->kanan = NULL;
    free(Hapus);
  }
}


//*********************************
//**FUNCTION HAPUS SIMPUL BELAKANG**
//*********************************
void Hapus_Belakang(simpul &DL)
{
  simpul bantu, hapus;
  if(DL==NULL)
    cout<<"Linked List Kosong ..............";
  else
  {
    bantu = DL;
    while(bantu->kanan->kanan != NULL)
    bantu=bantu->kanan;
    hapus = bantu->kanan;
    bantu->kanan = NULL;
    hapus->kiri = NULL;
    free(hapus);
  }
}
```

```
//**********************************
//**FUNCTION HAPUS SIMPUL DI TENGAH**
//**********************************
void Hapus_Tengah(simpul &DL, char elemen)
{
  simpul bantu, hapus;
  if(DL==NULL)
    cout<<"Linked List Kosong ................";
  else
  {
    bantu = DL;
    while(bantu->kanan->Isi != elemen)
      bantu=bantu->kanan;
    hapus = bantu->kanan;
    bantu->kanan->kanan->kiri=bantu;
    bantu->kanan = bantu->kanan->kanan;
    hapus->kanan = NULL;
    hapus->kiri = NULL;
    free(hapus);
  }
}
```

E. LATIHAN

Note: kerjakan semua percobaan diatas baru anda mengerjakan tugas dibawah ini:

1. Lakukan revisi pada source code doubly linked list non circular diatas supaya dapat memilih menu di awal running program.
2. Tambahkan fungsi untuk pencarian pada source code doubly linked list.
3. Tambahkan fungsi pengurutan pada source code doubly linked list.