BAB 1

PENDAHULUAN

1.1. Latar Belakang

Pemograman dalam struktur data ada beberapa macam. Salah satunya adalah pemograman C++. Dalam pemograman ini biasanya menggunakan variable Array, Struktur dan Linked List. Makalah ini membahas tentang salah satu variabel tersebut.

Single Linked List dikembangkan pada tahun 1955-1956 oleh Allen Newell, Cliff Shaw dan Herbert Simon di RAND Corporation sebagai struktur data utama untuk bahasa *Information Processing Language* (IPL). IPL dibuat untuk mengembangkan program *artificial intelligence* (kecerdasan buatan), seperti pembuatan *Chess Solver* .

Victor Yngve di Massachusetts Institute of Technology (MIT) juga menggunakan linked list pada *natural language processing* dan *machine transitions* pada bahasa pemrograman COMMIT.

1. Linked List adalah salah satu bentuk struktur data, berisi kumpulan data (node) yang tersusun secara sekuensial, saling sambung-menyambung, dinamis dan terbatas.
2. Linked List sering disebut juga Senarai Berantai.
3. Linked List saling terhubung dengan bantuan variabel pointer.
4. Masing-masing data dalam *Linked List* disebut dengan *node* (simpul) yang menempati alokasi memori secara dinamis dan biasanya berupa *struct* yang terdiri dari beberapa *field*.

1.2. Tujuan

- Mahasiswa memahami tentang Single Linked List
- Mahasiswa dapat menerapkan Linked List
- Mahasiswa dapat mengoperasikan Linked List

BAB 2

PEMBAHASAN

2.1 Landasan Teori

Pengertian:

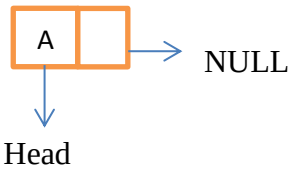
1. *Single* : artinya *field* pointer-nya hanya satu buah saja dan satu arah serta pada akhir *node* pointernya menunjuk *NULL*.
2. *Linked List* : artinya *node-node* tersebut saling terhubung satu sama lain.
3. Setiap *node* pada *linked list* mempunyai *field* yang berisi pointer ke *node* berikutnya, dan juga memiliki *field* yang berisi data.
4. *Node* terakhir akan menunjuk ke *NULL* yang akan digunakan sebagai kondisi berhenti pada saat pembacaan isi *linked list*.

Single linked list adalah linked list dengan simpul berisi satu *link* / Pointer yang mengacu kesimpul berikutnya. Operasi-operasi pada single linked list:

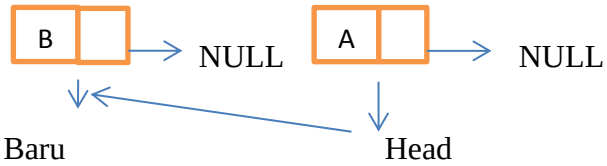
- **Penyisipan**

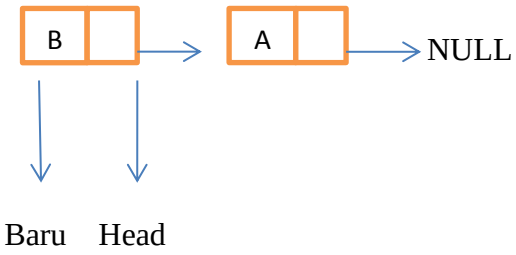
Penyisipan didepan

Penyisipan didepan pada single linked list adalah dengan cara menyisipkan data pada elemen awal list, sehingga pointer awal menunjuk list baru.



Datang data baru yaitu B:



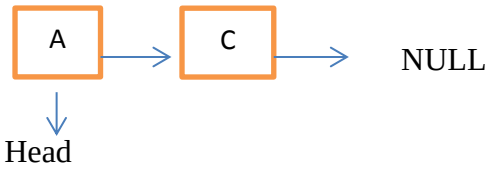


Perintah yang digunakan di c++ adalah

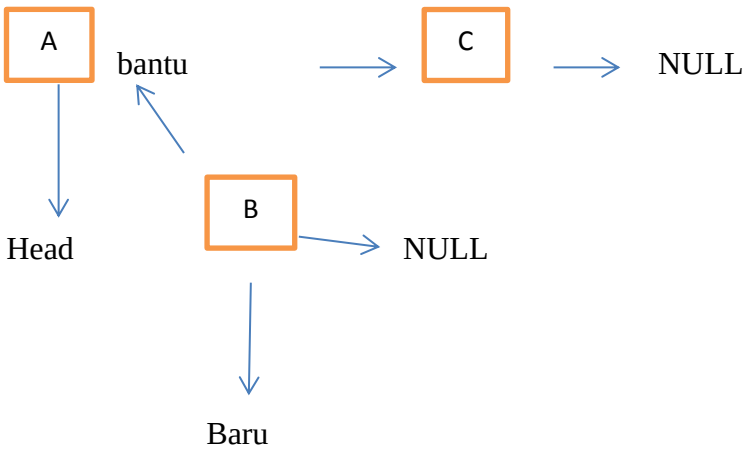
```
void insertDepan(int databaru)
{
TNode *baru;
baru = new TNode;
baru->data = databaru;
baru->next = NULL;
if(isEmpty()==1)
{
head=baru;
head->next = NULL;
}
else
{
baru->next = head;
head = baru;
}
printf(”Data masuk\n”);
}
```

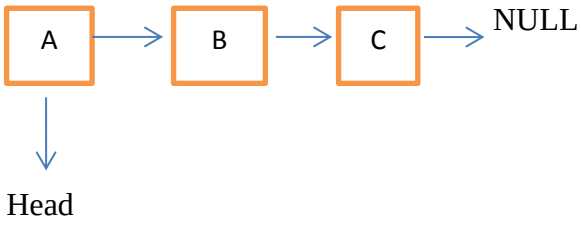
Penyisipan ditengah

Penyisipan ditengah pada single linked list adalah dengan cara menyisipkan data baru setelah elemen yang ditunjuk oleh variabel bantu pada list, kemudian pointer variable baru menunjuk pointer variable bantu.



Sisipkan data baru B setelah data A



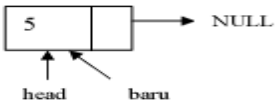


Penyisipan dibelakang

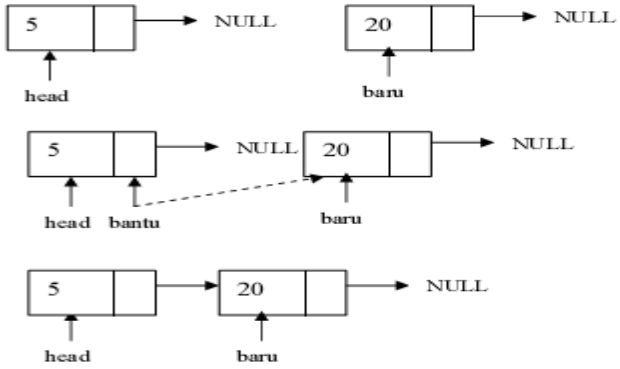
Penyisipan dibelakang pada single linked list adalah dengan cara menyisipkan data pada elemen akhir list, sehingga pointer akhir menunjuk list baru.

1. List masih kosong (head=NULL)

2. Masuk data baru, misalnya 5



3. Datang data baru, misalnya 20 (penambahan di belakang)



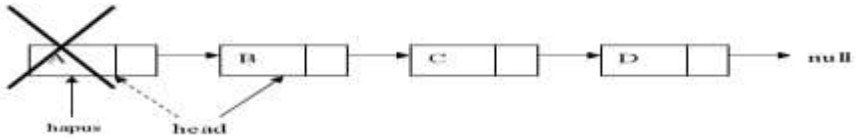
Perintah yang digunakan dalam c++ adalah :

```
void insertBelakang (int databaru)
{
TNode *baru,*bantu;
baru = new TNode;
baru->data = databaru;
baru->next = NULL;
if(isEmpty()==1) {
head=baru;
head->next = NULL;
}
else {
bantu=head;
while(bantu->next!=NULL){
bantu=bantu->next;
}
bantu->next = baru;
}
printf("Data masuk\n“);
}
```

- **Penghapusan**

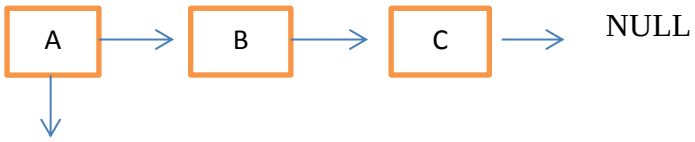
Penghapusan di depan

Untuk menghapus sebuah simpul diperlukan satu buah tambahan variabel pointer yaitu variabel bantu yang berguna untuk menunjukkan simpul manakah yang akan dihapus.



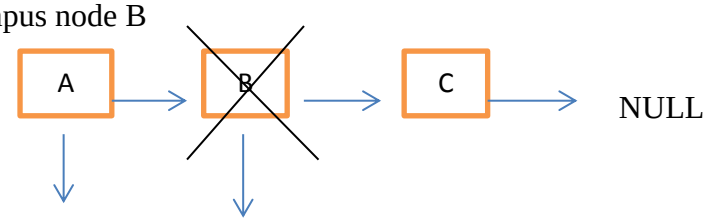
Penghapusan di tengah

Dengan cara menghapus data setelah elemen yang ditunjuk oleh variabel bantu pada list, kemudian pointer variable hapus menunjuk pointer variable bantu.

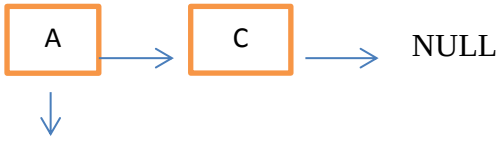


Head

Hapus node B



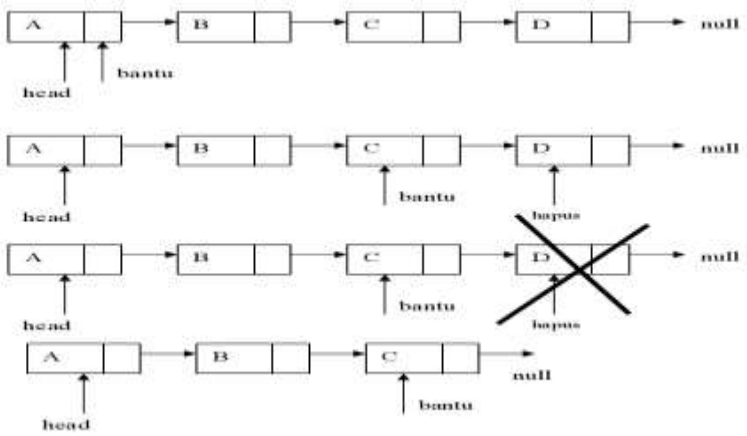
Head Hapus



Head

Penghapusan di belakang

Penghapusan dibelakang menggunakan pointer bantu dan hapus.pointer hapus digunakan untuk menunjuk node yang akan dihapus. Pointer bantu digunakan untuk menunjuk node sebelum node yang akan dihapus, yang kemudian akan dijadikan node terakhir.



- **Fungsi untuk menghapus semua node**

Untuk menghapus semua node dalam linked list di C++ dapat menggunakan perintah sebagai berikut:

```
void clear()
{
TNode *bantu,*hapus;
bantu = head;
while(bantu!=NULL)
{
hapus = bantu;
bantu = bantu->next;
delete hapus;
}
head = NULL;
}
```

- **Menampilkan Linked List**

Jika head sama dengan null, maka linked list tersebut adalah kosong. Jika tidak NULL, maka node bantu akan berpindah ke node selanjutnya dan membaca isi datanya dengan menggunakan field next sehingga dapat saling berkait.
Pemanggilan dalam c++ adalah sebagai berikut:

```
void tampil(){
TNode *bantu;
bantu = head;
if(isEmpty()==0){
while(bantu!=NULL){
cout<<bantu->data<<" ";
bantu=bantu->next;
}
printf("\n");
} else printf("Masih kosong\n");
}
```

2.2. Contoh Kasus dan Program

Contoh pemrograman dibawah ini adalah pemrograman pada bank.
Dengan input seperti dibawah ini:

```
#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <conio.h>

struct nasabah
{
 char norek[12];
```

```
 char nama[25];
 char alamat[50];
 double saldo;
};

struct simpul
{
 char norek[12];
 char nama[25];
 char alamat[50];
 double saldo;
 struct simpul *berikut;
};

struct simpul *awal=NULL, *akhir=NULL;
struct nasabah bank;

void tambah_list_dibelakang(struct nasabah info);
void isi_list();
void sisip_list(struct simpul *first,struct nasabah info,char posisi[20]);
void sisip_isi_list();
void tampil_list();
void hapus_simpul(char info);
void hapus_data();
void hapus_list();
void menu();

void main()
{
 clrscr();
 menu();
 getch();
```

```
}
void menu()
{
 int pil;
 clrscr();
 cout<<"Pilih Transaksi  : "<<endl;
 cout<<"          1. Isi List"<<endl;
 cout<<"          2. Sisip List"<<endl;
 cout<<"          3. Tampil List"<<endl;
 cout<<"          4. Hapus Salah Satu"<<endl;
 cout<<"          5. Hapus Semua List"<<endl;
 cout<<"          6. Exit"<<endl<<endl;
 cout<<"Tentukan Pilihan : ";
 cin>>pil;
 switch (pil)
 {
  case 1:
      isi_list();
      break;
  case 2:
      sisip_isi_list();
      break;
  case 3:
      tampil_list();
      break;
  case 4:
      hapus_data();
      break;
  case 5:
      hapus_list();
      break;
  case 6:
```

```
      clrscr();
      gotoxy(21,13);cout<<"Terima Kasih Telah Menggunakan Program Ini";
      break;
 }
}

void isi_list()
{
 char jawab;

 do
 {
  clrscr();
  cout<<"No Rekening : ";
  cin>>bank.norek;
  cout<<"Nama       : ";
  cin>>bank.nama;
  cout<<"Alamat     : ";
  cin>>bank.alamat;
  cout<<"Saldo Awal  : ";
  cin>>bank.saldo;
  tambah_list_dibelakang(bank);
  cout<<"\n\nTambah Data [Y/T] : ";
  cin>>jawab;
 }
 while (toupper(jawab)!='T');
 menu();
}

void tambah_list_dibelakang(struct nasabah info)
{
 struct simpul *baru;
```

```
 baru = (struct simpul *) malloc (sizeof(struct simpul));
 strcpy (baru -> norek,info.norek);
 strcpy (baru -> nama,info.nama);
 strcpy (baru -> alamat,info.alamat);
 baru -> saldo = info.saldo;

 if (awal == NULL)
 {
  awal = baru;
 }
  else
  {
   akhir -> berikut = baru;
  }
 akhir = baru;
 akhir -> berikut = NULL;
}

void tampil_list()
{
 clrscr();
 struct simpul *baca;
 int i;

 baca = awal;
 i = 1;

 while (baca != NULL)
 {
  cout<<"Data Ke : "<<i<<endl;
  cout<<"No Rekening  : "<<baca -> norek<<endl;
```

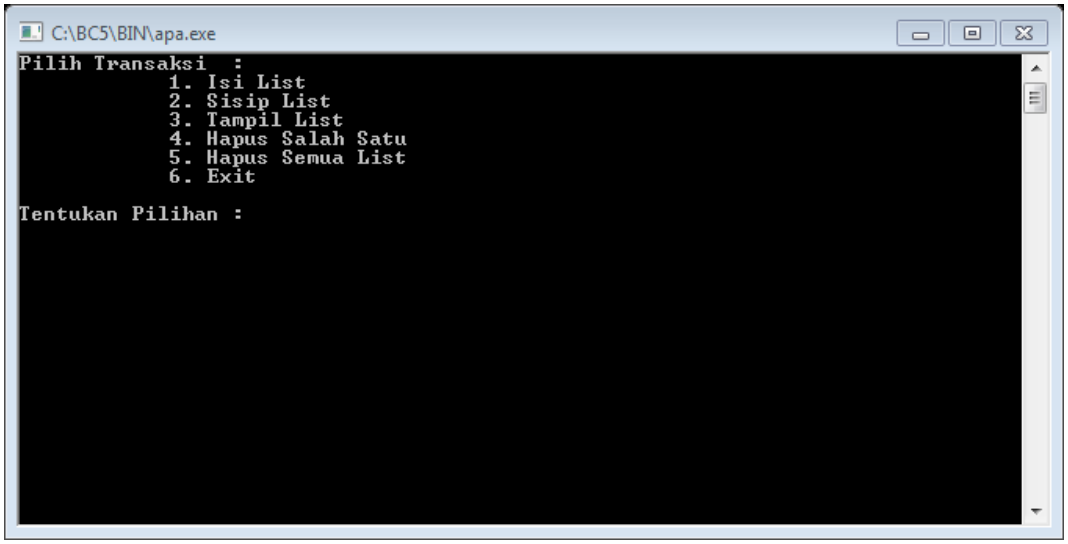
```
 cout<<"Nama         : "<<baca -> nama<<endl;
 cout<<"Alamat       : "<<baca -> alamat<<endl;
 cout<<"Saldo Awal   : "<<baca -> saldo<<endl;
 cout<<endl;
 i++;
 baca = baca -> berikut;
 }
 getch();
 menu();
}

void hapus_list()
{
 struct simpul *hapus;
 hapus = awal;
 while (hapus != NULL)
 {
  awal = hapus -> berikut;
  free(hapus);
  hapus = awal;
 }
 menu();
}


void sisip_list(struct simpul *first, struct nasabah info, char posisi[20])
{
 struct simpul *bantu,*baru;

 baru = new simpul;
 strcpy (baru -> norek,info.norek);
 strcpy (baru -> nama,info.nama);
```

```
 strcpy (baru -> alamat,info.alamat);
 baru-> saldo = info.saldo;
 bantu = first;

 do
 {
  if (strcmp(posisi,bantu->norek)!=0)
  {
   bantu = bantu->berikut;
  }
 }
 while (bantu !=NULL && strcmp(posisi,bantu->norek)!=0);

 baru->berikut = bantu -> berikut;
 bantu -> berikut = baru;
}

void sisip_isi_list()
{
 char cari[20];
 struct nasabah ganti;
  clrscr();
  cout<<"Input Data Baru"<<endl;
  cout<<"No Rekening : ";
  cin>>ganti.norek;
  cout<<"Nama        : ";
  cin>>ganti.nama;
  cout<<"Alamat      : ";
  cin>>ganti.alamat;
  cout<<"Saldo Awal  : ";
  cin>>ganti.saldo;
  cout<<endl;
```

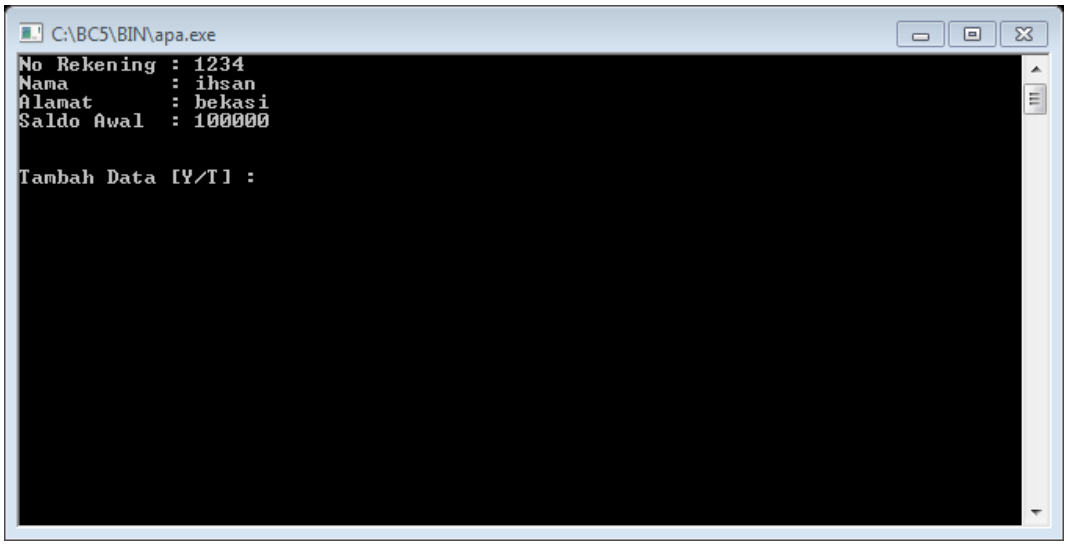
```
 cout<<"Data Akan Disisipkan Setelah No Rekening : ";
 gets(cari);
 sisip_list(awal,ganti,cari);
 menu();
}

void hapus_simpul(char info[20])
{
 struct simpul *bantu,*hapus;

 if (awal==NULL) { cout<<"\n Data List Kosong ";}
 else
 {
  if (strcmp(awal->norek,info)==0)
  {
   hapus=awal;
   awal=hapus->berikut;
   free(hapus);
  }
   else
   {
    bantu=awal;
    while ((strcmp(info,bantu->berikut->norek)!=0) && (bantu-
>berikut!=NULL))
    {
     bantu=bantu->berikut;
    }
    hapus=bantu->berikut;
    if (hapus!=NULL)
    {
     if (hapus!=akhir) { bantu->berikut=hapus->berikut;}
     else
```

```
    {
     akhir=bantu;
     akhir->berikut=NULL;
    }
    free(hapus);
   }
  }
 }
}


void hapus_data()
{
 char cari[20];
 char jawab;

 clrscr();

 cout<<"\nAda data yang akan dihapus Y/T :";cin>>jawab;
 if (toupper(jawab)=='Y')
 {
  cout<<"\nNo.Rekening yang akan dihapus :";
  gets(cari);
  hapus_simpul(cari);
  menu();
 }
}
```

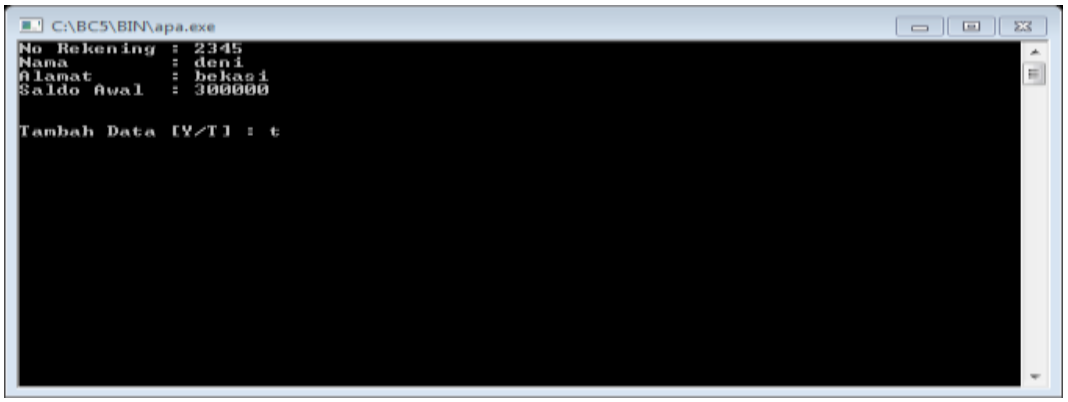
Sehingga output yang dikeluarkan adalah sebagai berikut :



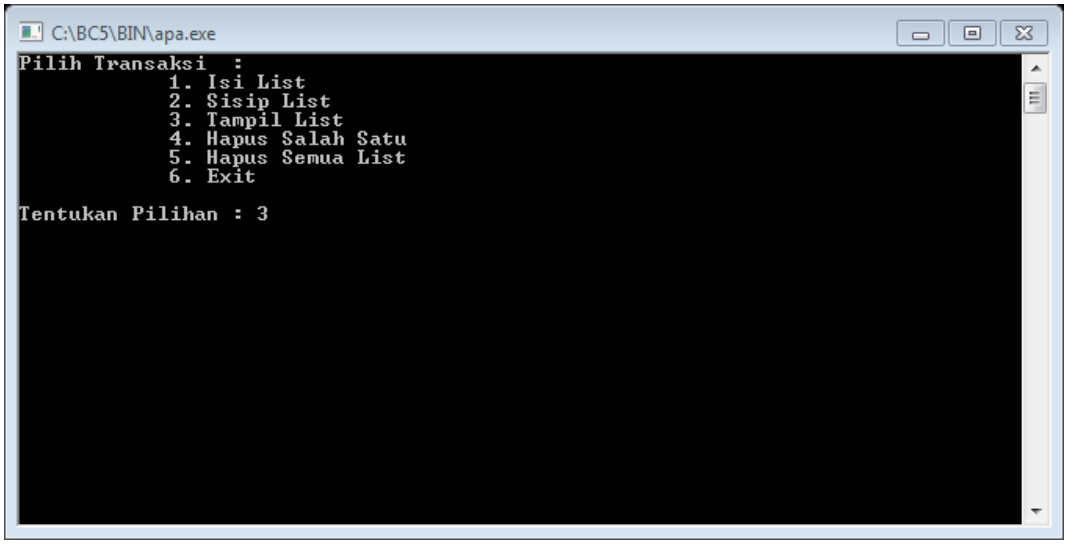
Tampilan program



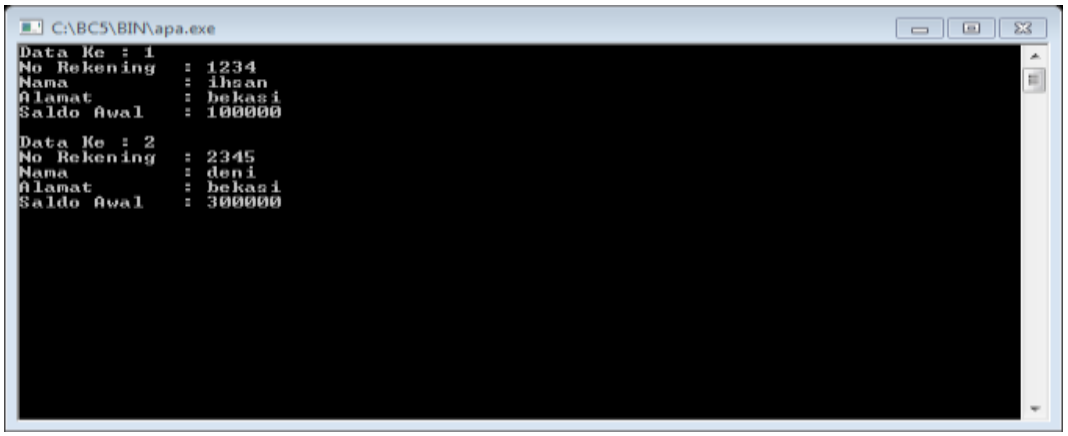
Input data pertama



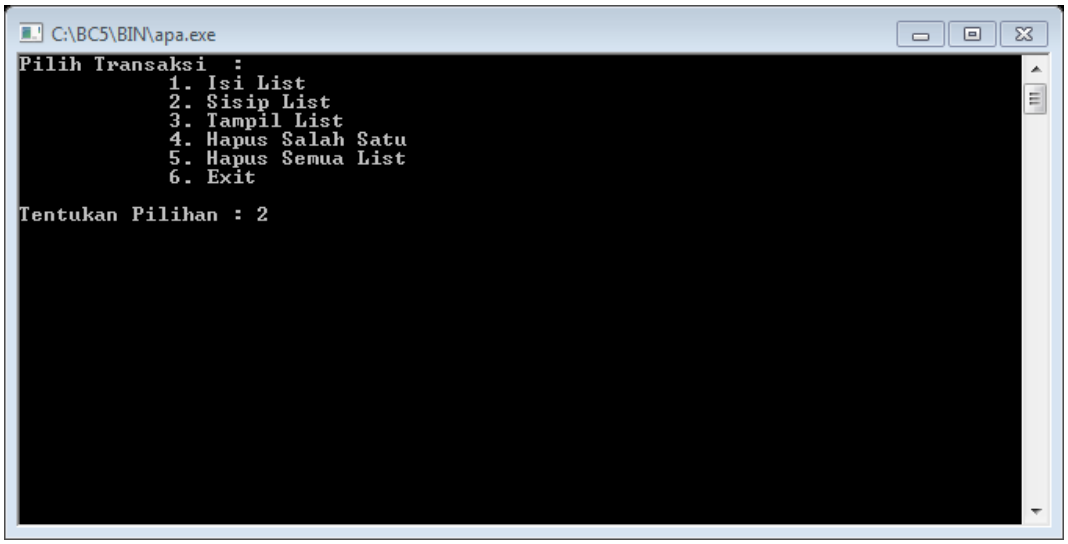
Input data kedua



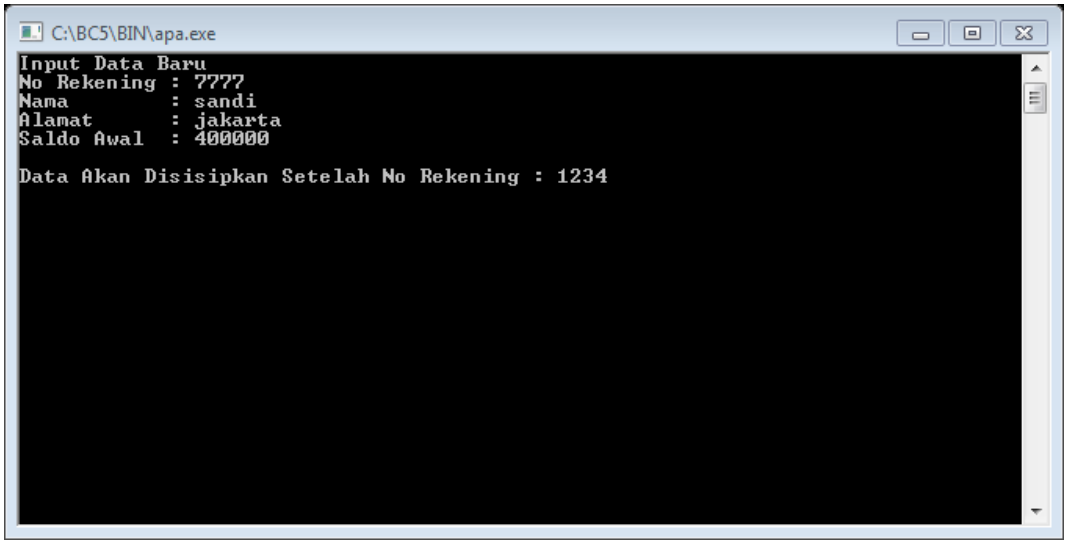
Setelah diinput, pilih nomer 3 untuk melihat data yang sudah tersimpan.



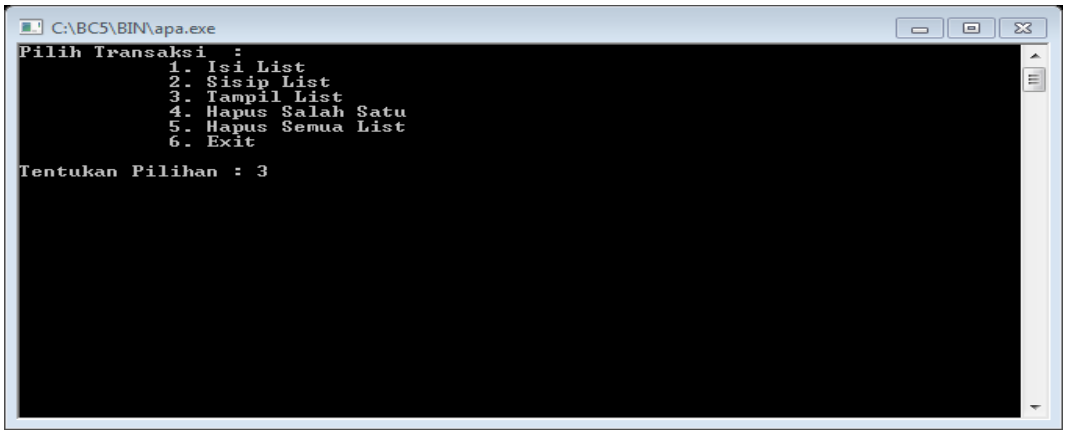
Tampilan saat data yang sudah tersimpan ditampilkan



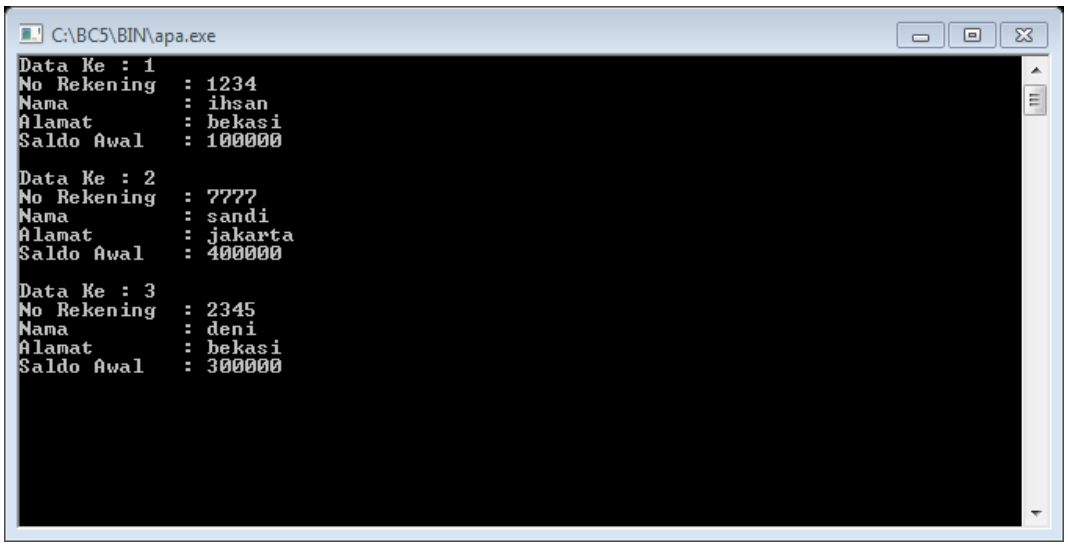
Pilih nomer 2 untuk menambahkan data baru



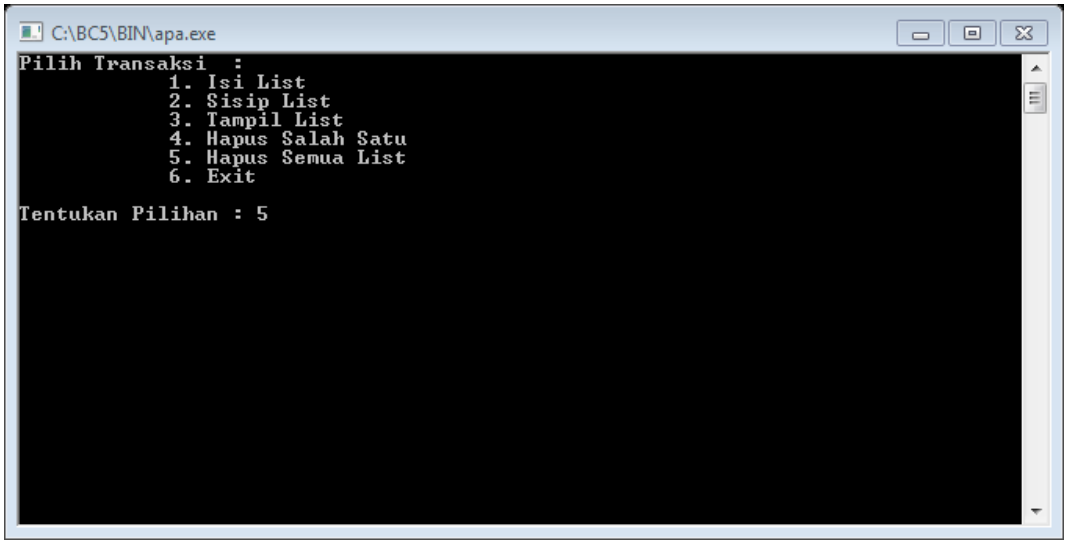
Tampilan untuk menambah data baru



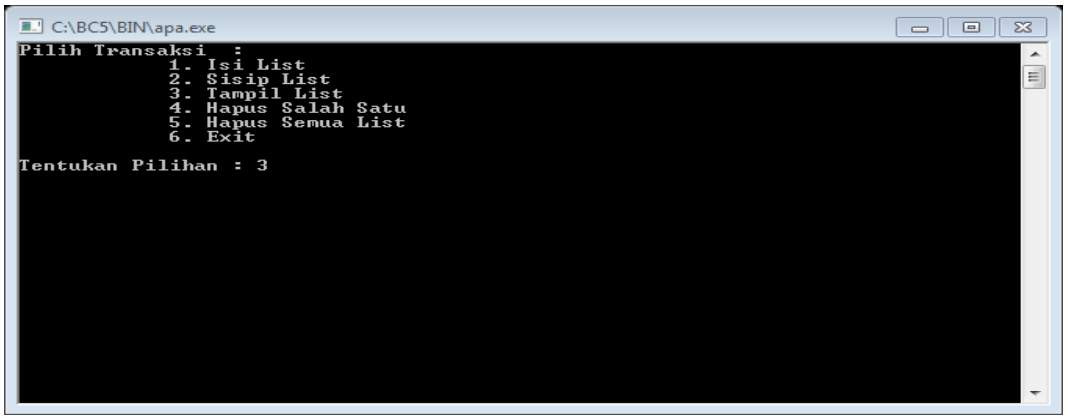
Setelah menambahkan data, pilih nomer 3 kembali untuk melihat data yang telah tersimpan



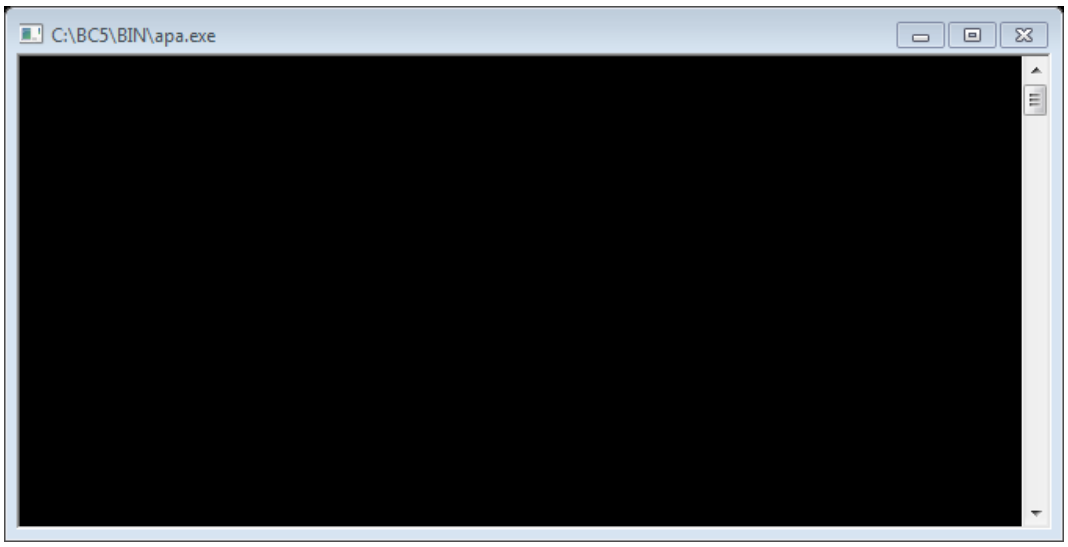
Tampilan data yang sudah tersimpan



Pilih nomer 5 untuk menghapus semua data



Setelah memilih menu 5, keluar dan kembali memilih menu 3 unutk melihat data



Data kosong setelah dihapus

BAB 3
PENUTUP

3.1. Kesimpulan

Linked List adalah salah satu bentuk struktur data, berisi kumpulan data (node) yang tersusun secara sekuensial, saling sambung-menyambung, dinamis dan terbatas.

Didalam linked list ada tiga pembahasan tentang penyisipan, yaitu penyisipan didepan, penyisipan ditengah dan penyisipan dibelakang .

Dan ada tiga pembahasan tentang penghapusan, yaitu penghapusan didepan, penghapusan ditengah dan penghapusan dibelakang .

Menampilkan Linked List , jika head sama dengan null, maka linked list tersebut adalah kosong. Jika tidak NULL, maka node bantu akan berpindah ke node selanjutnya dan membaca isi datanya dengan menggunakan field next sehingga dapat saling berkait.