

**MAKALAH
LINKED LIST**



Nama	:	Lintang Mangku Langit	18615004
		Ali Arsa	18615008
		Angga	18615009
		Adit	18615013

TEKNOLOGI INFORMASI
POLITEKNIK NEGERI SAMARINDA
2018/2019

KATA PENGANTAR

Assalamu'alaikum Wr. Wb

Puji dan Syukur Penulis Panjatkan ke Hadirat Tuhan Yang Maha Esa karena berkat limpahan Rahmat dan Karunia-Nya sehingga penulis dapat menyusun makalah ini tepat pada waktunya. Makalah ini membahas Linked List.

Dalam penyusunan makalah ini, penulis banyak mendapat tantangan dan hambatan akan tetapi dengan bantuan dari berbagai pihak tantangan itu bisa teratasi. Olehnya itu, penulis mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah membantu dalam penyusunan makalah ini, semoga bantuannya mendapat balasan yang setimpal dari Tuhan Yang Maha Esa.

Penulis menyadari bahwa makalah ini masih jauh dari kesempurnaan baik dari bentuk penyusunan maupun materinya. Kritik konstruktif dari pembaca sangat penulis harapkan untuk penyempurnaan makalah selanjutnya.

Akhir kata semoga makalah ini dapat memberikan manfaat kepada kita sekalian.

Wassalamu'alaikum Wr. Wb

Samarinda, 16 Mei 2019

Penyusun

DAFTAR ISI

MAKALAH	i
KATA PENGANTAR	i
DAFTAR ISI	ii
BAB I PENDAHULUAN	1
1.1. Latar Belakang.....	1
1.2. Tujuan.....	1
BAB II ISI.....	2
2. 1. Pengertian.....	2
2.2 Jenis	3
2.2.1 Single Linked List.....	3
2.2.2 Double Linked List	4
2.2.3 Circular Linked List	6
BAB III SOURCE CODE dan OUTPUT PROGRAM	8
SOURCE CODE	8
OUTPUT	15
DAFTAR PUSTAKA	21

BAB I

PENDAHULUAN

1.1. Latar Belakang

Pada array, apabila programmer ingin menyimpan data, programmer diharuskan untuk mendefinisikan besar array terlebih dahulu, seringkali programmer mengalokasikan array yang sangat besar (misal 100). Hal ini tidak efektif karena seringkali yang dipakai tidak sebesar itu. Dan apabila programmer ingin menyimpan data lebih dari seratus data, maka hal itu tidak dapat dimungkinkan karena sifat array yang besarnya statik. Linked list adalah salah satu struktur data yang mampu menutupi kelemahan tersebut.

Linked list (list bertaut) adalah salah satu struktur data dasar yang sangat fundamental dalam bidang ilmu komputer. Dengan menggunakan linked list maka programmer dapat menyimpan datanya kapanpun dibutuhkan. Linked list mirip dengan array, kecuali pada linked list data yang ingin disimpan dapat dialokasikan secara dinamis pada saat pengoperasian program (run-time).

1.2. Tujuan

1. Mengetahui pengertian linked list.
2. Mengetahui macam-macam dari linked list.
3. Mengetahui contoh program sederhana yang menggunakan linked list.

BAB II

ISI

2. 1. Pengertian

Salah satu bentuk struktur data yang berisi kumpulan data yang tersusun secara sekuensial, saling bersambungan, dinamis, dan terbatas adalah senarai berkait (linked list). Suatu senarai berkait (linked list) adalah suatu simpul (node) yang dikaitkan dengan simpul yang lain dalam suatu urutan tertentu. Suatu simpul dapat berbentuk suatu struktur atau class. Simpul harus mempunyai satu atau lebih elemen struktur atau class yang berisi data.

Senarai berkait (linked list) adalah salah satu struktur data dasar yang sangat fundamental dalam bidang ilmu komputer. Dengan menggunakan linked list maka programmer dapat menyimpan datanya kapanpun dibutuhkan. Linked list mirip dengan array, kecuali pada linked list data yang ingin disimpan dapat dialokasikan secara dinamis pada saat pengoperasian program (run-time).

Didalam banyak aplikasi, ukuran dari data tidak diketahui saat compile, hal ini bisa merupakan suatu atribut yang baik juga. Setiap node akan berbentuk struct dan memiliki satu buah field yang bertipe struct yang sama, yang berfungsi sebagai pointer. Dalam menghubungkan setiap node, kita dapat menggunakan cara first-create-first-access maupun first-create-last-access.

Linked list saling terhubung dengan bantuan variabel pointer. 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.

2.2 Jenis

2.2.1 Single Linked List

Single linked list adalah sebuah linked list yang menggunakan variabel pointer saja untuk menyimpan banyak data dengan metode linked list, suatu daftar yang isinya saling berhubungan.

2.2.1.1 Representasi Single Linked List

- Representasi Simpul
Typedef struct simpul Node;
Struct simpul {
 Int data;
 Node *next;
};
- Deklarasi Global
Node *head = null;
Node *baru;

2.2.1.2 Operasi pada Single Linked List

- Operasi Sisip pada Single Linked List
 - a. Penyisipan didepan
Penyisipan didepan pada single linked list adalah dengan cara menyisipkan data pada elemen awal list, sehingga pointer awal menunjuk list baru.
 - b. 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 variabel baru menunjuk pointer variabel tertentu.

c. Penyisipan dibelakang

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

- Operasi Hapus
 - a. Menghapus simpul
 - b. Hapus simpul awal
 - c. Hapus simpul akhir
 - d. Hapus simpul tertentu

2.2.2 Double Linked List

Double linked list adalah suatu linked list yang mempunyai 2 penunjuk yaitu penunjuk ke simpul sebelumnya dan simpul berikutnya.

2.2.2.1 Representasi Double Linked List

- Representasi Simpul

```
Typedef struct simpul Dnode;
Struct simpul {
    Int data;
    Dnode *next;
    Dnode *prev;
};
```
- Deklarasi Global

```
Dnode *head = null;
Dnode *tail = null;
Dnode *baru;
```

2.2.2.2 Operasi pada Double Linked List

- Operasi Sisip

- a. Penyisipan didepan

Operasi ini berguna untuk menambahkan satu simpul baru diposisi pertama. Langkah pertama untuk menambahkan data adalah pembuatan simpul baru dan mengisinya dengan data pada field infonya. Pointer yang menunjuk ke simpul tersebut dipanggil dengan nama baru.

- b. Penyisipan ditengah

Operasi penyisipan data ditengah linked list adalah suatu operasi menambah data di posisi tertentu di dalam linked list. Karena double linked list memiliki dua pointer sambungan, maka penyisipan bisa dilakukan sebelum data tertentu atau sesudah data tertentu, berbeda dengan single linked list yang hanya memiliki satu pointer sambungan, yaitu sambungan next.

- c. Penyisipan dibelakang

Operasi ini berguna untuk menambahkan elemen baru diposisi akhir. Langkah pertama untuk penambahan data adalah pembuatan elemen baru dan pengisian nilai infonya. Pointer yang menunjuk ke data tersebut dipanggil dengan nama baru.

- Operasi Hapus

- a. Hapus simpul awal

Penghapusan data diawal adalah proses penghapusan simpul pertama (yang ditunjuk oleh variabel pointer awal), sehingga variabel pointer awal akan berpindah ke simpul berikutnya. Ada tiga kondisi yang perlu diperhatikan, yaitu kondisi linked list masih kosong, kondisi linked list hanya memiliki satu data, dan linked list yang memiliki data lebih dari satu elemen.

b. Hapus simpul akhir

Penghapusan data diakhir adalah proses penghapusan pada simpul akhir (yang ditunjuk pointer akhir) dengan cara mengcopy pointer akhir dengan pointer hapus dan memindahkan pointer akhir ke tetangga kirinya dan data pun dapat dihapus.

c. Hapus simpul tengah

Penghapusan data ditengah merupakan proses penghapusan data dengan cara user memasukkan data/posisi data yang akan dihapus dan ditunjuk oleh pointer hapus. Untuk menghapus data tersebut dibutuhkan sebuah pointer bantu yang menunjuk data sebelum data yang akan dihapus. Dengan tujuan sebagai penyambung kembali data sebelum dan sesudah data yang akan dihapus.

2.2.3 Circular Linked List

Circular Linked list terbagi dua, yaitu :

- Single Linked List Circular

Single linked list circular adalah single linked list yang pointer nextnya menunjuk pada dirinya sendiri. Jika single linked list tersebut terdiri dari beberapa node, maka pointer next pada node terakhir akan menunjuk ke node terdepannya.

Ilustrasi single linked list circular:

1. Setiap node pada linked list mempunyai field yang berisi pointer ke node berikutnya, dan juga memiliki field yang berisi data.
2. Pada akhir linked list node terakhir akan menunjuk ke node terdepannya sehingga linked list tersebut berputar.

- Double Linked List Circular

Double linked list circular adalah linked list yang menggunakan pointer, dimana setiap node memiliki 3 field, yaitu:

1. Satu field pointer yang menunjuk pointer berikutnya “next”.
2. Satu field pointer yang menunjuk pointer sebelumnya “prev”.
3. Satu field pointer yang berisi data untuk node tersebut.

Double linked list circular pointer next dan prev nya menunjuk ke dirinya sendiri secara circular. Ilustrasi double linked list circular:

1. Setiap node pada linked list mempunyai field yang berisi data dan pointer ke node berikutnya dan node sebelumnya.
2. Untuk pembentukan node baru, mulanya pointer next dan prev akan menunjuk ke nilai null.
3. Selanjutnya pointer prev akan menunjuk ke node sebelumnya, dan pointer next akan menunjuk ke node selanjutnya pada list.

BAB III

SOURCE CODE dan OUTPUT PROGRAM

SOURCE CODE

```
#include <iostream>

#include <conio.h>

#include <stdlib.h>

#define CLRSCR system("cls");

using namespace std;

struct node // deklarasi node sebagai struct yang isinya nama kelas dan npm
{

char nama[20];

char kls[6];

char npm[9];

node *nxt; // Pointer untuk node selanjutnya

};

node *mulai_ptr = NULL;

node *saat_ini;

int pilihan = 0;

void tambah_node_di_akhir()
```

```

{
node *temp, *temp2;

temp = new node;

cout << "Masukkan nama: ";cin >> temp->nama;

cout << "Masukkan Kelas : ";cin >> temp->cls;

cout << "Masukkan NPM : ";cin >> temp->npm;

temp->nxt = NULL;

if (mulai_ptr == NULL)

{

mulai_ptr = temp;

saat_ini = mulai_ptr;

}

else

{

temp2 = mulai_ptr;

while (temp2->nxt != NULL)

{

temp2 = temp2->nxt;

}

temp2->nxt = temp;

}

```

```

}

void tampilkan_list()

{
    node *temp;

    temp = mulai_ptr;

    cout << endl;

    if (temp == NULL)

        cout << "List kosong!" << endl;

    else

    {

        while (temp != NULL)

        {

            cout << "nama : " << temp->nama << " ";

            cout << "Kelas : " << temp->kls << " ";

            cout << "NPM : " << temp->npm;

            if (temp == saat_ini)

                cout << " <-- saat_ini node";

            cout << endl;

            temp = temp->nxt;

        }

        cout << "Akhir dari list!" << endl;
    }
}

```

```

    }

}

void delete_mulai_node()

{

    node *temp;

    temp = mulai_ptr;

    mulai_ptr = mulai_ptr->nxt;

    delete temp;

}

void delete_akhir_node()

{

    node *temp1, *temp2;

    if (mulai_ptr == NULL)

        cout << "List kosong!" << endl;

    else

    {

        temp1 = mulai_ptr;

        if (temp1->nxt == NULL)

        {

            delete temp1;

            mulai_ptr = NULL;

```

```

}

else

{

    while (temp1->nxt != NULL)

    {

        temp2 = temp1;

temp1 = temp1->nxt;

    }

    delete temp1;

temp2->nxt = NULL;

}

}

}

void move_saat_ini_on ()

{

    if (saat_ini->nxt == NULL)

        cout << "Kamu berada pada akhir list." << endl;

    else

        saat_ini = saat_ini->nxt;

}

void move_saat_ini_back ()

```

```

{

if (saat_ini == mulai_ptr)

cout << "Kamu berada pada awal list" << endl;

else

{

    node *sebelum;

sebelum = mulai_ptr;

while (sebelum->nxt != saat_ini)

{

    sebelum = sebelum->nxt;

}

saat_ini = sebelum;

}

}

int main()

{

mulai_ptr = NULL;

do

{

    system("CLS");

tampilkan_list();

```



```

cout << endl;

cout << "-----\n";

cout << "Masukkan pilihan Anda : " << endl;

cout << "-----\n";

cout << "0. Keluar dari program." << endl;

cout << "1. Tambah node di akhir list." << endl;

cout << "2. Hapus awal node dalam list." << endl;

cout << "3. Hapus akhir node dalam list." << endl;

cout << "4. Pindah pointer saat_ini ke depan satu node." << endl;

cout << "5. Pindah pointer saat_ini ke belakang satu node." << endl;

cout << endl << " >> ";

cin >> pilihan;

switch (pilihan)

{

case 1 : tambah_node_di_akhir(); break;

case 2 : delete_mulai_node(); break;

case 3 : delete_akhir_node(); break;

case 4 : move_saat_ini_on(); break;

case 5 : move_saat_ini_back();

}

}

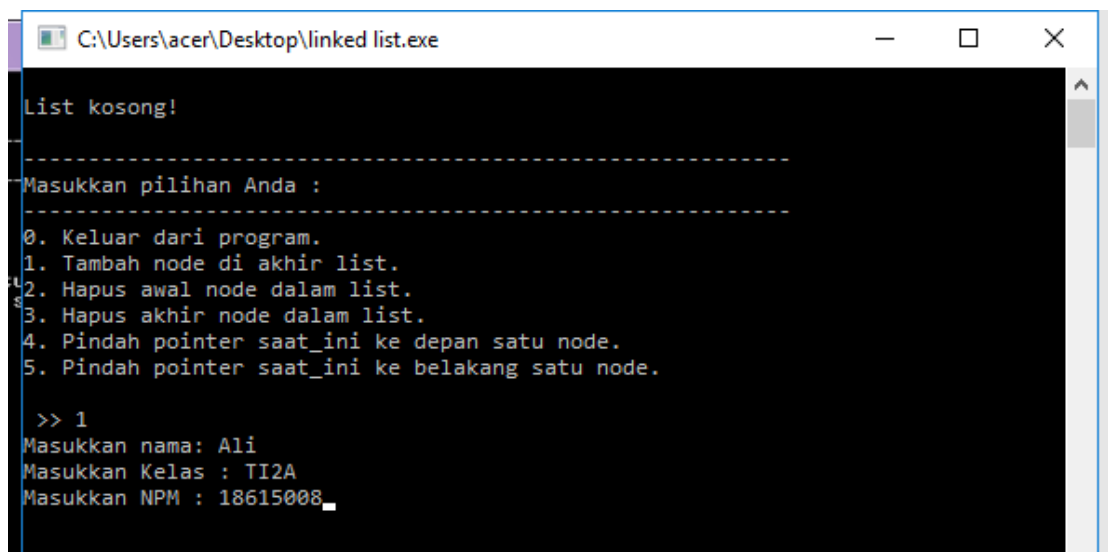
```

```
while (pilihan != 0);
```

```
}
```

OUTPUT

1. Berikut adalah output dari contoh program linked list , bila di input pilihan no. 1 maka akan tampil menu untuk memasukkan nama , kelas, dan npm.

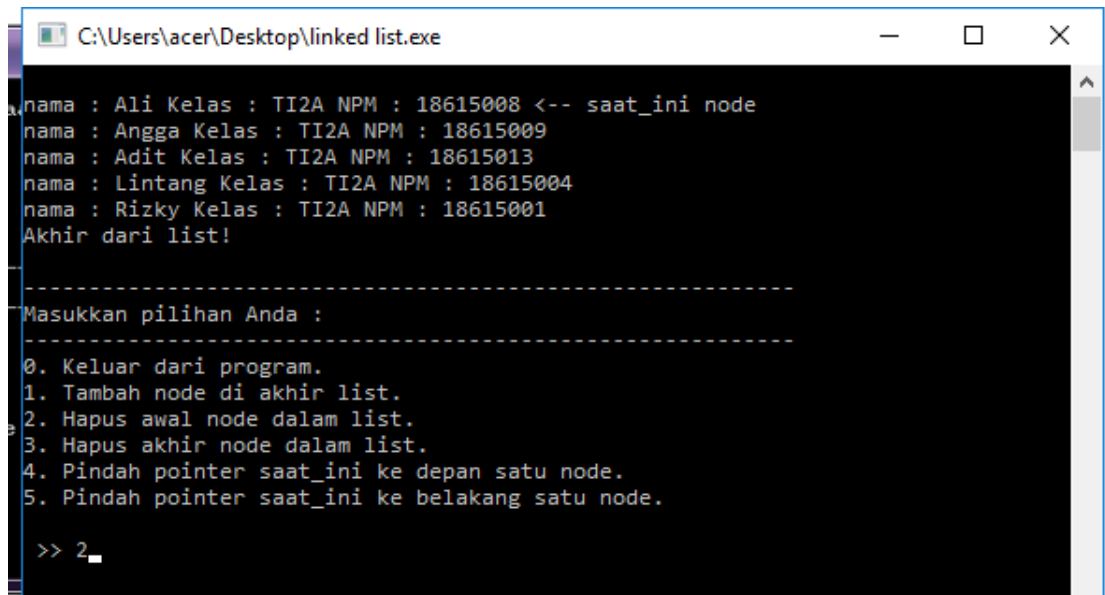


```
C:\Users\acer\Desktop\linked list.exe

List kosong!
-----
Masukkan pilihan Anda :
-----
0. Keluar dari program.
1. Tambah node di akhir list.
2. Hapus awal node dalam list.
3. Hapus akhir node dalam list.
4. Pindah pointer saat_ini ke depan satu node.
5. Pindah pointer saat_ini ke belakang satu node.

>> 1
Masukkan nama: Ali
Masukkan Kelas : TI2A
Masukkan NPM : 18615008_
```

2. Setelah selesai diinput , selanjutnya input pilihan no.2 yaitu pilihan untuk menghapus awal node dalam list .

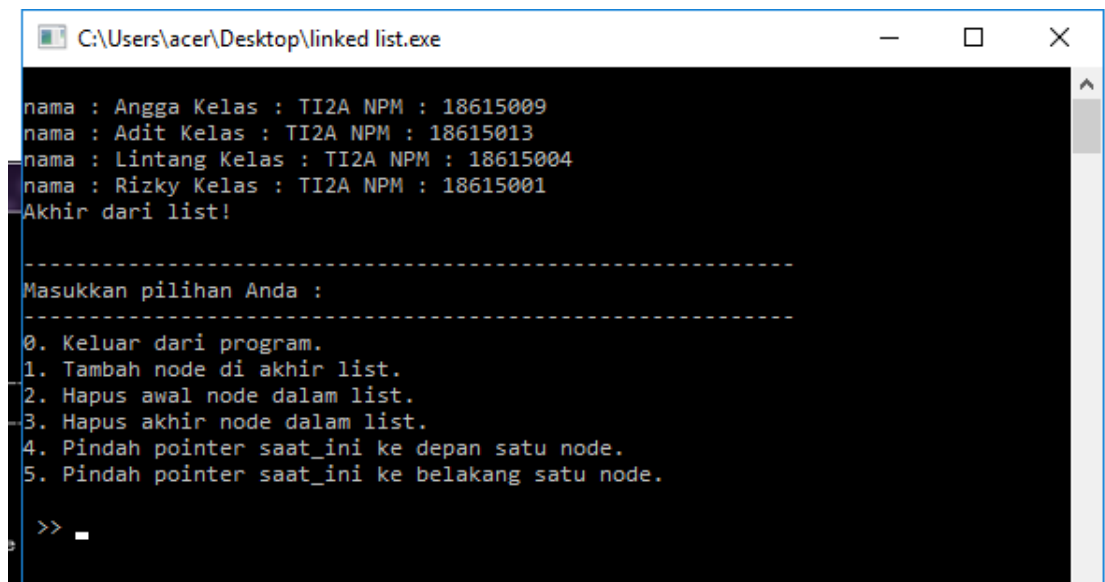


```
C:\Users\acer\Desktop\linked list.exe
nama : Ali Kelas : TI2A NPM : 18615008 <-- saat_ini node
nama : Angga Kelas : TI2A NPM : 18615009
nama : Adit Kelas : TI2A NPM : 18615013
nama : Lintang Kelas : TI2A NPM : 18615004
nama : Rizky Kelas : TI2A NPM : 18615001
Akhir dari list!

-----
Masukkan pilihan Anda :
-----
0. Keluar dari program.
1. Tambah node di akhir list.
2. Hapus awal node dalam list.
3. Hapus akhir node dalam list.
4. Pindah pointer saat_ini ke depan satu node.
5. Pindah pointer saat_ini ke belakang satu node.

>> 2.
```

3. Tampilan berikutnya seperti dibawah ini .

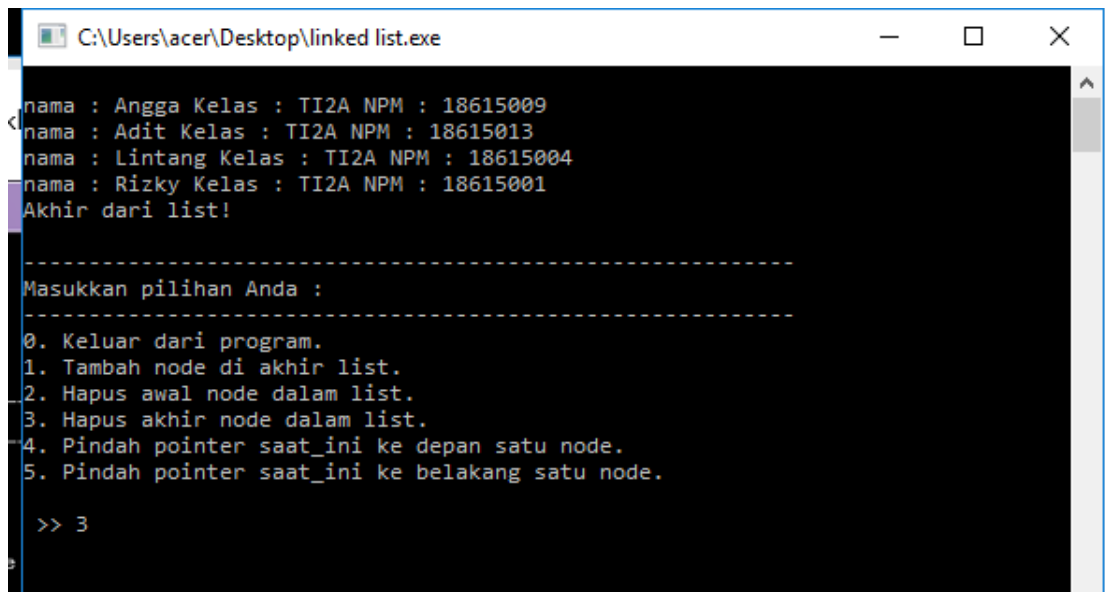


```
C:\Users\acer\Desktop\linked list.exe
nama : Angga Kelas : TI2A NPM : 18615009
nama : Adit Kelas : TI2A NPM : 18615013
nama : Lintang Kelas : TI2A NPM : 18615004
nama : Rizky Kelas : TI2A NPM : 18615001
Akhir dari list!

-----
Masukkan pilihan Anda :
-----
0. Keluar dari program.
1. Tambah node di akhir list.
2. Hapus awal node dalam list.
3. Hapus akhir node dalam list.
4. Pindah pointer saat_ini ke depan satu node.
5. Pindah pointer saat_ini ke belakang satu node.

>> 
```

4. Input no.3 yaitu menghapus node paling terakhir ,



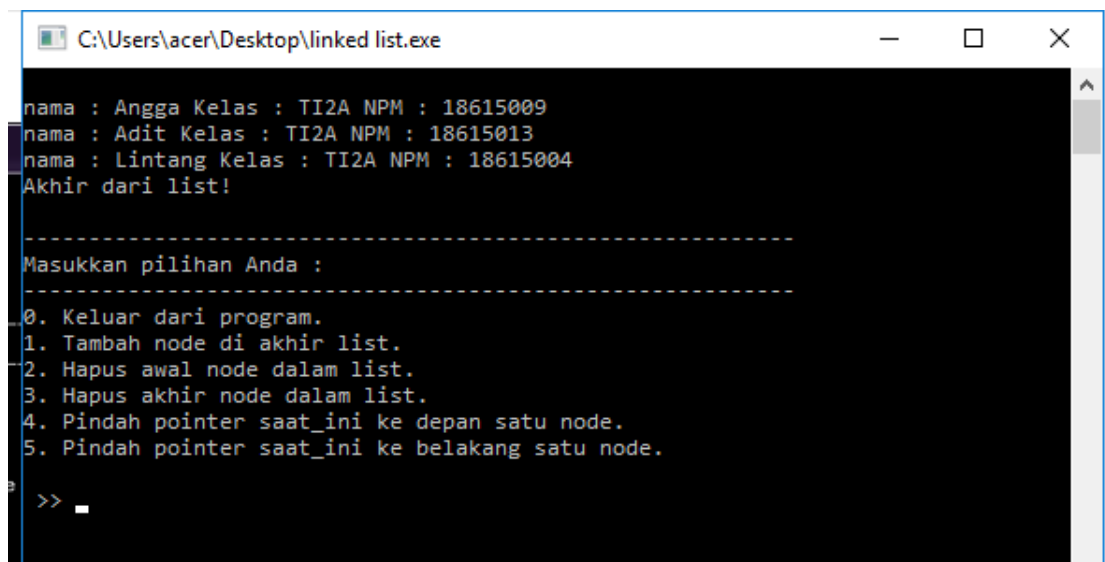
```
C:\Users\acer\Desktop\linked list.exe

nama : Angga Kelas : TI2A NPM : 18615009
nama : Adit Kelas : TI2A NPM : 18615013
nama : Lintang Kelas : TI2A NPM : 18615004
nama : Rizky Kelas : TI2A NPM : 18615001
Akhir dari list!

-----
Masukkan pilihan Anda :
-----
0. Keluar dari program.
1. Tambah node di akhir list.
2. Hapus awal node dalam list.
3. Hapus akhir node dalam list.
4. Pindah pointer saat_ini ke depan satu node.
5. Pindah pointer saat_ini ke belakang satu node.

>> 3
```

5. Tampilannya seperti berikut



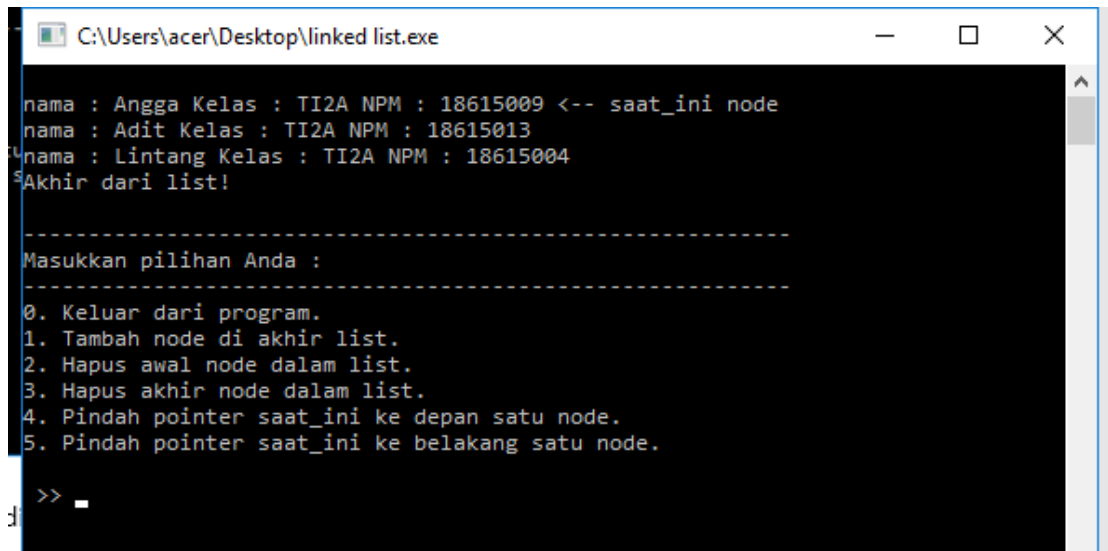
```
C:\Users\acer\Desktop\linked list.exe

nama : Angga Kelas : TI2A NPM : 18615009
nama : Adit Kelas : TI2A NPM : 18615013
nama : Lintang Kelas : TI2A NPM : 18615004
Akhir dari list!

-----
Masukkan pilihan Anda :
-----
0. Keluar dari program.
1. Tambah node di akhir list.
2. Hapus awal node dalam list.
3. Hapus akhir node dalam list.
4. Pindah pointer saat_ini ke depan satu node.
5. Pindah pointer saat_ini ke belakang satu node.

>> 
```

6. Perhatikan pointer yang berada di atas , saat di input no.4 , pointer akan berada didepan satu node.



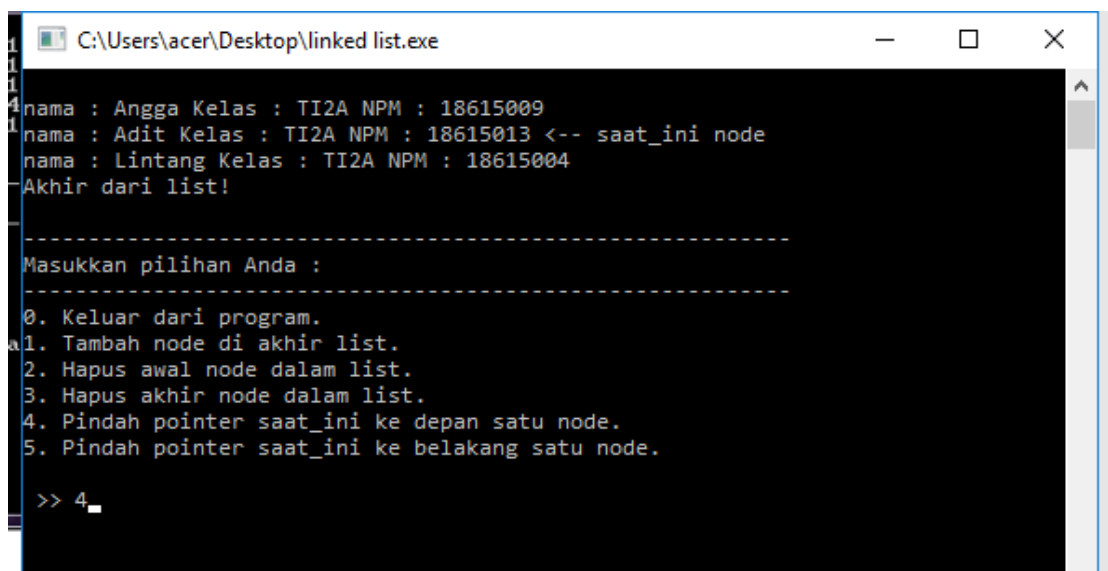
```
C:\Users\acer\Desktop\linked list.exe

nama : Angga Kelas : TI2A NPM : 18615009 <-- saat_ini node
nama : Adit Kelas : TI2A NPM : 18615013
nama : Lintang Kelas : TI2A NPM : 18615004
Akhir dari list!

-----
Masukkan pilihan Anda :
-----
0. Keluar dari program.
1. Tambah node di akhir list.
2. Hapus awal node dalam list.
3. Hapus akhir node dalam list.
4. Pindah pointer saat_ini ke depan satu node.
5. Pindah pointer saat_ini ke belakang satu node.

>> _
```

7. Seperti tampilan di bawah ini .



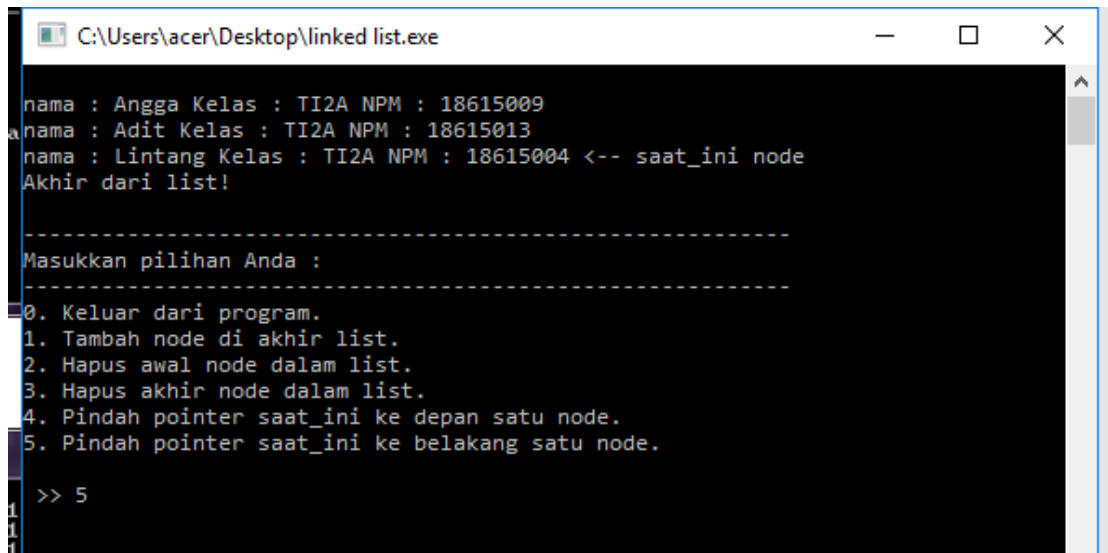
```
C:\Users\acer\Desktop\linked list.exe

nama : Angga Kelas : TI2A NPM : 18615009
nama : Adit Kelas : TI2A NPM : 18615013 <-- saat_ini node
nama : Lintang Kelas : TI2A NPM : 18615004
Akhir dari list!

-----
Masukkan pilihan Anda :
-----
0. Keluar dari program.
1. Tambah node di akhir list.
2. Hapus awal node dalam list.
3. Hapus akhir node dalam list.
4. Pindah pointer saat_ini ke depan satu node.
5. Pindah pointer saat_ini ke belakang satu node.

>> 4_
```

8. Perhatikan pointer , setelah di input no.5 , pointer akan berada di belakang sebanyak satu node .



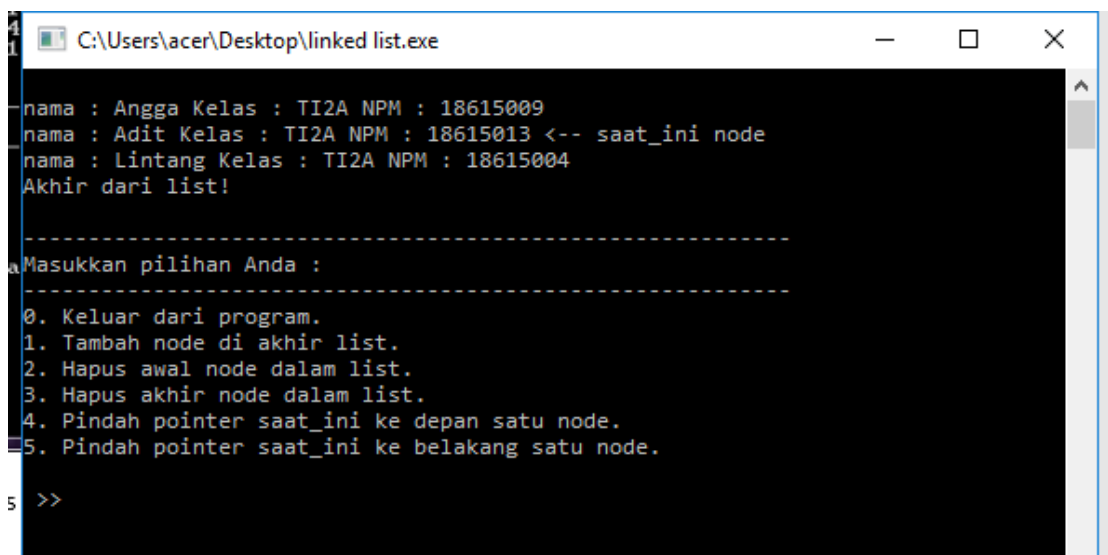
```
C:\Users\acer\Desktop\linked list.exe

nama : Angga Kelas : TI2A NPM : 18615009
nama : Adit Kelas : TI2A NPM : 18615013
nama : Lintang Kelas : TI2A NPM : 18615004 <-- saat_ini node
Akhir dari list!

-----
Masukkan pilihan Anda :
-----
0. Keluar dari program.
1. Tambah node di akhir list.
2. Hapus awal node dalam list.
3. Hapus akhir node dalam list.
4. Pindah pointer saat_ini ke depan satu node.
5. Pindah pointer saat_ini ke belakang satu node.

>> 5
```

9. Tampilan seperti dibawah ini



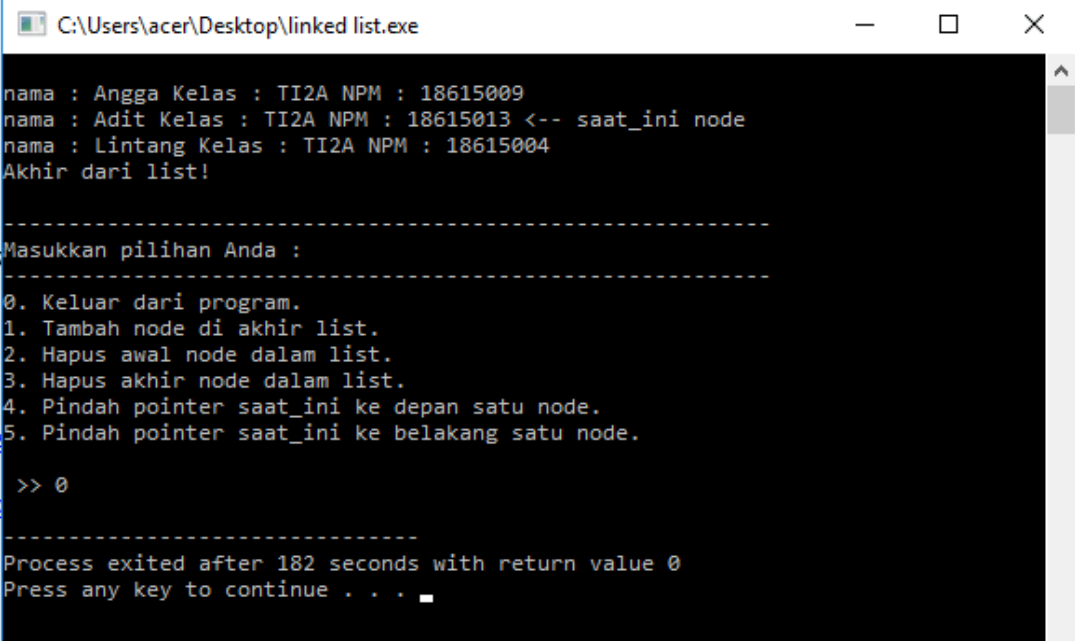
```
C:\Users\acer\Desktop\linked list.exe

nama : Angga Kelas : TI2A NPM : 18615009
nama : Adit Kelas : TI2A NPM : 18615013 <-- saat_ini node
nama : Lintang Kelas : TI2A NPM : 18615004
Akhir dari list!

-----
Masukkan pilihan Anda :
-----
0. Keluar dari program.
1. Tambah node di akhir list.
2. Hapus awal node dalam list.
3. Hapus akhir node dalam list.
4. Pindah pointer saat_ini ke depan satu node.
5. Pindah pointer saat_ini ke belakang satu node.

5 >>
```

10. Setelah selesai melakukan proses linked list , maka bisa menginput pilhan 0 untuk keluar dari program .



```
C:\Users\acer\Desktop\linked list.exe

nama : Angga Kelas : TI2A NPM : 18615009
nama : Adit Kelas : TI2A NPM : 18615013 <-- saat_ini node
nama : Lintang Kelas : TI2A NPM : 18615004
Akhir dari list!

-----
Masukkan pilihan Anda :
-----
0. Keluar dari program.
1. Tambah node di akhir list.
2. Hapus awal node dalam list.
3. Hapus akhir node dalam list.
4. Pindah pointer saat_ini ke depan satu node.
5. Pindah pointer saat_ini ke belakang satu node.

>> 0

-----
Process exited after 182 seconds with return value 0
Press any key to continue . . .
```

DAFTAR PUSTAKA

<http://timoen-devil.blogspot.com/>

<http://www.slideshare.net/yunanhiannasution/materi-linked-list-dan-bubble-sort>

<http://www.bocahit.com/2012/08/linked-list-c.html>

<http://faizallutvi.blogspot.com/2013/03/contoh-program-linked-list-c.html>

<http://brawlyvonfabre.blogspot.com/p/single-linked-list.html>

<http://brawlyvonfabre.blogspot.com/p/double-linked-list.html>

<http://lecturer.eepis->

its.edu/~arna/Modul_AS/6.%20Double%20Link%20List.pdf

<http://www.blog.renggagumilar.com/strukturdatadoublelinkedlist/>

<http://www.blog.renggagumilar.com/makalah-single-linked-list-bab-1-dan-2/>

<http://thesilent15.blogspot.com/2010/07/single-linked-list-circular.html>