

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL III
SINGLE AND DOUBLE LINKED LIST**



Disusun Oleh :

Anisah Syifa Mustika Riyanto
2311102080

Dosen

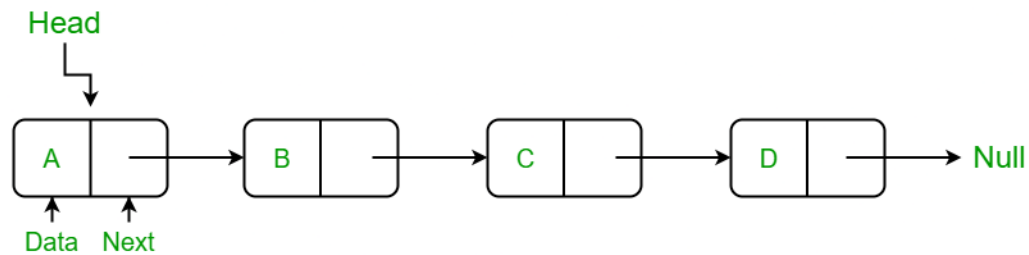
Wahyu Andi Saputra, S.Pd., M.Eng.

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

A. Dasar Teori

Single Linked List

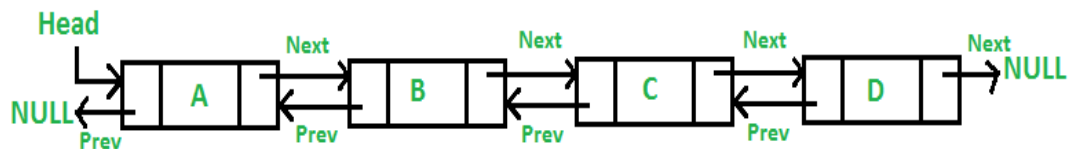
Single Linked List adalah salah satu bentuk struktur data yang berisi kumpulan node yang tersusun secara sekuensial, saling terhubung, dan dinamis. Linked list sering juga disebut Senarai Berantai. Linked list saling terhubung dengan bantuan pointer. Masing- masing data dalam linked list disebut dengan node yang terdiri dari beberapa field yaitu data dan pointer. Berikut adalah ilustrasi Single Linked List:



Pada gambar di atas, data terletak pada sebuah lokasi dalam sebuah memory, yang bertempat di node. Setiap node memiliki pointer (penunjuk) yang menunnjuk ke node berikutnya, sehingga terbentuk simpul impul node yang disebut Single Linked List. Dalam single linked list ponter hanya dapat bergerak ke satu arah saja maka dalam mencari data dan memproses data hanya dapat dilakukan dalam satu arah saja. Single liked list menggunakan dua variabel pointer yakni head dan tail. Head akan selalu menunjuk pada node pertama sedangkan tail akan selalu menunjuk pada node terakhir dalam list.

Double Linked List

Double Linked List adalah suatu linked list yang mempunyai 2 penunjuk yaitu penunjuk ke simpul sebelumnya dan ke simpul berikutnya. Perhatikan gambar di bawah ini :



Deklarasi secara umum double linked list :

Type

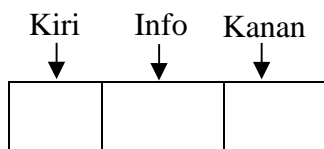
```
nama_pointer = ↑Simpul  
Simpul = Record  
    medan_data : tippedata  
    medan_sambungan_kiri, medan_sambungan_kanan :  
Namapointer EndRecord  
nama_var_pointer : nama_pointer
```

Contoh:

Type

```
Point =  
↑Data Data  
= Record  
    < info : char  
        prev, next : Point >  
Endrecord  
awal, akhir: Point
```

Jadi satu simpul di double linked list adalah sebagai berikut :



Dari gambar di atas, untuk setiap simpul terdiri dari 3 buah field yaitu medan sambungan kiri (prev), medan data (info), dan medan sambungan kanan (next).

B. Guided

Guided 1 Single Linked List Non- Circular

Source code:

```
// SINGLE LINKED LIST NON-CIRCULAR

#include <iostream>
using namespace std;

// Deklarasi Struct Node
struct Node
{
    int data;// menyimpan nilai yang diberikan kepada node saat node tersebut dibuat.
    Node *next;//pointer yang menunjuk ke node berikutnya
};

Node *head;
Node *tail;

// Inisialisasi Node
void init()
{
    head = NULL; // jadi head dan tail bernilai kosong
    tail = NULL;
}

// Pengecekan apakah list kosong
bool isEmpty()
{
    return head == NULL;
} // memeriksa apakah linked list kosong atau tidak. Jika head bernilai NULL, maka
dianggap linked list kosong dan fungsi akan mengembalikan true

// Tambah Node di depan
void insertDepan(int nilai)//int nilai adalah parameter yang memungkinkan Anda untuk
menentukan nilai apa yang akan dimasukkan ke dalam node baru saat menggunakan
fungsi insertDepan(). Berarti di sini nilai yang dimasukkan harus integer
{
    Node *baru = new Node;//membuat pointer baru yang menunjuk ke sebuah objek
Node baru. Objek tersebut, sesuai struct, akan memiliki 2 anggota yaitu data dan next.
New itu operator btw.
    baru->data = nilai;//nilai data dari node baru akan berisi nilai yang diberikan
parameter prosedur.
    baru->next = NULL;//pointer next dari node baru menjadi NULL. Karena node baru
akan dimasukkan di depan linked list, maka node baru tersebut akan menjadi elemen
pertama, sehingga pointer next-nya tidak menunjuk ke node lain.
    if (isEmpty())
    {
        head = tail = baru;
    } //kalo kosong maka akan menjadi satu- satunya node dalam linked list.
    else
    {
        baru->next = head;
        head = baru;
    }
}
```

```

    } //kalo engga maka akan ditambahkan di depan node yang telah ada, dan head di
    pindah ke node baru
}

// Tambah Node di belakang
void insertBelakang(int nilai)
{
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    } //node baru ditambahkan setelah node paling belakang, dan menjadi tail.
}

// Hitung jumlah Node di list
int hitungList() //parameter bersifat opsional. Jika fungsi itu membutuhkan input, maka
kita harus membuatkan parameter.
{
    Node *hitung = head; //terdapat node baru yang memiliki pointer hitung, node
    tersebut adalah head (berada di palig awal)
    int jumlah = 0; //jumlah awal adalah 0
    while (hitung != NULL)
    {
        jumlah++; //tiap iterasi akan bertambah 1
        hitung = hitung->next; //Pada setiap iterasi loop, pointer hitung akan diarahkan ke
        node berikutnya dalam linked list dengan mengakses pointer next
    }
    return jumlah;
}

// Tambah Node di posisi tengah
void insertTengah(int data, int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        Node *baru = new Node(); //()penggunaannya opsional, defaultnya aja gitu, cuma
        gaya penulisan, bukan fungsi.
        baru->data = data;
        Node *bantu = head; //Membuat pointer bantu yang menunjuk ke node pertama
        dari linked list. Atau anggap aja ini tu salinannya head, biasanya namanya cur kalo di

```

yfb.

```
int nomor = 1; // untuk melacak posisi saat ini dalam linked list.  
while (nomor < posisi - 1) // misal mau disisipkan di antar anode 2 dan 3 ya berarti  
posisi-1=2.
```

```
{  
    bantu = bantu->next; // kalo nomor masih memenuhi berarti bantu yang tadinya  
    sama dengan head, ya berarti menunjuk terus (bantu next) sampai tidak memenuhi.
```

```
    nomor++;  
} // nah kalo udah selesai bantu nya sekarang kan berada di antara node yang  
mau disisipkan, atau anggap aja sebelum posisi -1.
```

```
baru->next = bantu->next; // next pada baru disamakan dengan next pada bantu,  
jadi sekarang menunjuk node yang sama. Menunjuk ke node sebelah kanan yang  
akan disisipi.
```

```
bantu->next = baru; // next dari bantu sekarang adalah node baru. Jadi bantu  
yang tadinya ada 2 duplikat yaitu yang satunya node sebelah kiri sebelum disisipi,  
maka keduanya sekarang menunjuk ke node baru.
```

```
}  
} // https://youtu.be/ujXmAcDnSjc?si=oOpdeusrNhcZr2l8
```

// kenapa sih cur atau biasanya duplikat dari suatu node tertentu kalo ada yang dirubah
maka berubah dua duanya? ya karena memiliki pointer yang sama, alamat yang
sama.

// Hapus Node di depan

```
void hapusDepan()
```

```
{  
    if (!isEmpty())  
    {  
        Node *hapus = head; // duplikat head (bayangkan ada node duplikat di bawah node  
head).
```

```
        if (head->next != NULL)  
        {
```

```
            head = head->next;  
            delete hapus; // sama aja yang dihapus itu head yang sebelum dipindah ke  
node kedua karena hapus masih menunjuk ke head sebelumnya, dan head yang  
sekarang node kedua tidak dihapus.
```

```
        }  
        else
```

```
        {  
            head = tail = NULL;  
            delete hapus; // kalo cuma ada satu node maka di hapus langsung  
        }  
    }  
}
```

```
else  
    { // kalo listnya kosong makan akan ditampilkan pesan ini.  
        cout << "List kosong!" << endl;
```

```
    }  
} // https://youtu.be/VVemCxif9vg?si=nxRVbZOzIQLpn21v
```

// Hapus Node di belakang

```
void hapusBelakang()
```

```
{  
    if (!isEmpty())  
    {  
        if (head != tail)
```

```

    {
        Node *hapus = tail;
        Node *bantu = head;
        while (bantu->next != tail)
        {
            bantu = bantu->next;
        }
        tail = bantu;
        tail->next = NULL;
        delete hapus;
    }
    else
    {
        head = tail = NULL;
    }
}
else
{
    cout << "List kosong!" << endl;
}
} //sudah bisa dan pernah diilustrasikan sendiri.

// Hapus Node di posisi tengah
void hapusTengah(int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        Node *hapus;
        Node *bantu = head;
        for (int nomor = 1; nomor < posisi - 1; nomor++)
        {
            bantu = bantu->next;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;
        //jadi node bantu itu nextnya menuju ke tail atau sama saja menuju ke hapus
        next(node terakhir), jadi node tengah diantara bantu dan hapus next itu tidak ditunjuk,
        maka dari itu dihapus.
        delete hapus;
    }
}

// Ubah data Node di depan
void ubahDepan(int data)
{
    if (!isEmpty())
    {

```

```

        head->data = data;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah data Node di posisi tengah
//https://youtu.be/ujXmAcDnSjc?si=s_oGZOMgr2JRKbDI
void ubahTengah(int data, int posisi)
{
    if (!isEmpty())
    {
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "Posisi di luar jangkauan" << endl;
        }
        else if (posisi == 1)
        {
            cout << "Posisi bukan posisi tengah" << endl;
        }
        else
        {
            Node *bantu = head;
            for (int nomor = 1; nomor < posisi; nomor++) {
                //ini berarti bantu next nya nanti ada di node yang kita cari, jadi kalo misalnya
                cari 4 ya berarti nanti bantu next nya ada di node 4.
                bantu = bantu->next;
            }
            bantu->data = data;
            //isi datanya sama karena pointer-nya juga sama jadi nilainya sama.
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah data Node di belakang
void ubahBelakang(int data)
{
    if (!isEmpty())
    {
        tail->data = data;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Hapus semua Node di list
void clearList()

```



```

{
    Node *bantu = head;
    while (bantu != NULL)
    {
        Node *hapus = bantu;
        bantu = bantu->next;
        delete hapus;
        //jadi kalo bantunya udah pindah di node setelahnya, hapusnya itu masih di node
        yang tetap , atau node sebelum bantu next, jadi yang dihapus itu node sebelum bantu
        next.
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
    //kayaknya sih dihapus semua soalnya bantu = head = hapus, jadi kalo hapusnya di
    delete ya maka sama aja semua nilainya di delete karena kan pointernya sama?
}
// Tampilkan semua data Node di list
void tampil()
{
    if (!isEmpty())
    {
        Node *bantu = head;
        while (bantu != NULL)
        {
            cout << bantu->data << " ";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

int main()
{
    init();
    insertDepan(3);
    tampil();
    insertBelakang(5);
    tampil();
    insertDepan(2);
    tampil();
    insertDepan(1);
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, 2);
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1);

```

```

    tampil();
    ubahBelakang(8);
    tampil();
    ubahTengah(11, 2);
    tampil();
    return 0;
}

```

Screenshots Output

```

PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC> & 'c:\Users\hp151\.vscode\extensions\ms-vscode.cpptools-1.1
9.9-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-oqqzmmwfl.ive' '--stdout=Microsof
t-MIEngine-Out-ojbqgf5h.gzy' '--stderr=Microsoft-MIEngine-Error-pvqbs1z5.sre' '--pid=Microsoft-MIEngine-Pid-15qwq5xi.2cb'
'--dbgExe=C:\Program Files\CodeBlocks\MinGW\bin\gdb.exe' '--interpreter=mi'
3
3 5
2 3 5
1 2 3 5
2 3 5
2 3
2 7 3
2 3
1 3
1 8
1 11
PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC>

```

Deskripsi:

Program di atas dibuat menggunakan single linked list. Program memiliki fungsi untuk mengecek apakah list kosong, menambah node di depan, menambah node di tengah, menambah node di belakang, menghitung jumlah node dalam list, hapus node depan, hapus node tengah, hapus node belakang, ubah node depan, ubah node tengah, ubah node belakang, hapus semua node dalam list, dan tampilkan semua node dalam list.

Guided 2 Double Linked List

Source code:

```

//DOUBLE LINKED LIST

#include <iostream>
using namespace std;

//deklarasi
class Node
{
public:
    int data;
    Node *prev;
    Node *next;
};

class DoublyLinkedList
{
public:
    Node *head;
    Node *tail;
}

```

```
DoublyLinkedList()
```

```
{  
    head = nullptr;  
    tail = nullptr;  
}
```

```
void push(int data)
```

```
{  
    Node *newNode = new Node;  
    newNode->data = data;  
    newNode->prev = nullptr;  
    newNode->next = head;
```

```
    if (head != nullptr)
```

```
    {  
        head->prev = newNode;  
    }
```

```
    else
```

```
    {  
        tail = newNode;  
    }
```

```
    head = newNode;
```

//jadi sekarang hanya ada satu node/ tercipta 1 node karena headnya dipindah ke newNode sedangkan tail juga berada di newNode.

```
void pop()
```

```
{//https://youtu.be/HoubOPoC44s?si=_nFEdwb5D-oVWQhX
```

```
    if (head == nullptr)
```

```
    {  
        return;// Jika daftarnya kosong, tidak ada yang perlu dipop, jadi kembalikan.  
    }
```

```
    Node *temp = head;
```

```
    head = head->next;
```

```
    if (head != nullptr)
```

```
    {  
        head->prev = nullptr;  
    } //jadi head sekarang jadi node pertama, soalnya prev nya NULL.
```

```
    else
```

```
    {  
        tail = nullptr;  
    }
```

```
    delete temp;
```

```
}
```

```
bool update(int oldData, int newData)
```

```
{  
    Node *current = head;// current dan head emnunjuk ke simpul pertama.
```

```
    while (current != nullptr)
```

```
    {  
        if (current->data == oldData) // Nanti kan user disuruh untuk menginputkan old data, baru dibandingkan, kalo data cur nya sam adengan old data maka diganti ke
```

new data, lalu cur pindah ke cur next.

```
{
    current->data = newData;
    return true;
}
current = current->next;
}
return false;
}

void deleteAll()
{
    Node *current = head;
    while (current != nullptr)
    {
        Node *temp = current;
        current = current->next;
        delete temp;
    }
    head = nullptr;
    tail = nullptr;
}

void display()
{
    Node *current = head;
    while (current != nullptr)
    {
        cout << current->data << " ";
        current = current->next;
    }//ini diakhiri dengan current= current next, kalo current next nya menunjuk ke
    NULL ya berarti sudah tidak memenuhi syarat perulangan.
    cout << endl;
}
};
```

```
int main()
{
    DoublyLinkedList list;
    while (true)//jika diberi while maka setelah memilih 1 case program akan
    mengulang dan membiarkan user memilih case lainnya tanpa me run ulang program
    dan keluar dari program.
```

```
{
    cout << "1. Add data" << endl;
    cout << "2. Delete data" << endl;
    cout << "3. Update data" << endl;
    cout << "4. Clear data" << endl;
    cout << "5. Display data" << endl;
    cout << "6. Exit" << endl;
```

```
int choice;
cout << "Enter your choice: ";
cin >> choice;
```

```
switch (choice)
```

```

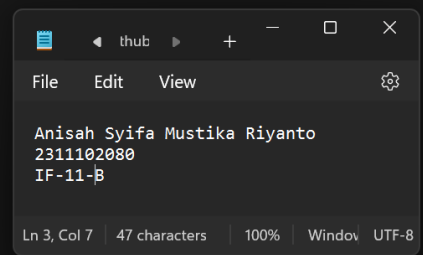
{
case 1:
{
    int data;
    cout << "Enter data to add: ";
    cin >> data;
    list.push(data);
    break;
}
case 2:
{
    list.pop();
    break;
}
case 3:
{
    int oldData, newData;
    cout << "Enter old data: ";
    cin >> oldData;
    cout << "Enter new data: ";
    cin >> newData;
    bool updated = list.update(oldData, newData);
    if (!updated)
    {
        cout << "Data not found" << endl;
    }
    break;
}
case 4:
{
    list.deleteAll();
    break;
}
case 5:
{
    list.display();
    break;
}
case 6:
{
    return 0;
}
default:
{
    cout << "Invalid choice" << endl;
    break;
}
}
return 0; //ketika pengguna memilih opsi "Exit" (case 6), program akan kembali dari
fungsi main() dengan pernyataan return 0;, yang menyebabkan program berakhir.
}

```

```

PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC> & 'c:\Users\hp151\.vscode\extensions\ms-vscode.cpptools-1.19.9-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-1u3sxlrx.vbe' '--stdout=Microsoft-MIEngine-Out-izwyqrot.5k0' '--stderr=Microsoft-MIEngine-Error-rnzgnodn.w52' '--pid=Microsoft-MIEngine-Pid-dh4zxopj.fgh' '--dbgExe=C:\Program Files\CodeBlocks\MinGW\bin\gdb.exe' '--interpreter=mi'
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 1
Enter data to add: 123
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 1
Enter data to add: 456
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 1
Enter data to add: 789
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5

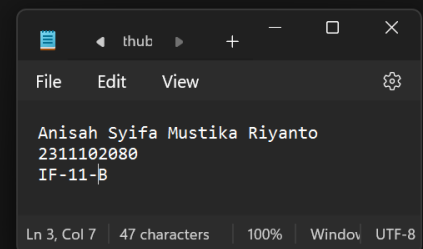
```



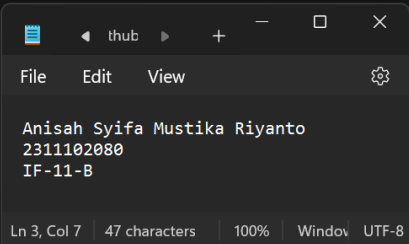
```

6. Exit
Enter your choice: 5
789 456 123
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 2
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
456 123
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 3
Enter old data: 456
Enter new data: 321
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
321 123

```



```
Enter your choice: 5
321 123
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 4
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 6
PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC>
```



Deskripsi:

Program menggunakan double linked list yang memiliki 6 menu, yaitu add data, delete data, update data, clear data, display data, dan exit. Perulangan untuk menampilkan menu akan terus berjalan ketika bernilai true, jadi ketika user sudah menginputkan pilihan menu dan mendapatkan output program, maka perulangan akan berjalan dan menampilkan pilihan menu kembali tanpa user perlu untuk me- run ulang program atau tanpa program berhenti terlebih dahulu. Program akan berhenti menampilkan pilihan menu hanya jika user memilih menu 6 yaitu exit dimana program akan kembali ke main membawa return 0; jadi program akan berhenti.

C. Unguided

Unguided 1

Buatlah program menu Single Linked List Non-Circular untuk menyimpan Nama dan usia mahasiswa, dengan menggunakan inputan dari user. Lakukan operasi berikut:

- a. Masukkan data sesuai urutan berikut. (Gunakan insert depan, belakang atau tengah). Data pertama yang dimasukkan adalah nama dan usia anda.

[Nama_anda]	[Usia_anda]
John	19
Jane	20
Michael	18
Yusuke	19
Akechi	20
Hoshino	18
Karin	18

- b. Hapus data Akechi
c. Tambahkan data berikut diantara John dan Jane : Futaba 18
d. Tambahkan data berikut diawal : Igor 20
e. Ubah data Michael menjadi : Reyn 18
f. Tampilkan seluruh data

Source code:

```
#include <iostream>
#include <iomanip>
using namespace std;

// Deklarasi Struct Node
struct Node
{
    string Nama;
    int Umur;
    Node *next;
};

Node *head;
Node *tail;

// Inisialisasi Node
void inisialisasi()
{
    head = NULL;
    tail = NULL;
}
```



```

// Cek Node
bool cekNode()
{
    if (head == NULL)
        return true;
    else
        return false;
}

// Add First
void addFirst(string name, int age)
{
    Node *baru = new Node;
    baru->Nama = name;
    baru->Umur = age;
    baru->next = NULL;

    if (cekNode() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}

// Add Last
void addLast(string name, int age)
{
    Node *baru = new Node;
    baru->Nama = name;
    baru->Umur = age;
    baru->next = NULL;

    if (cekNode() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}

// Hitung Jumlah Node dalam Linked List
int hitungNode()
{
    Node *hitung;
    hitung = head;

```

```

    int jumlah = 0;

    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }

    return jumlah;
}

// Add Middle
void addMiddle(string name, int age, int posisi)
{
    if (posisi < 1 || posisi > hitungNode())
    {
        cout << "Posisi di luar jangkauan." << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan di tengah." << endl;
    }
    else
    {
        Node *baru, *bantu;
        baru = new Node();
        baru->Nama = name;
        baru->Umur = age;

        // Tranversing (proses melintasi atau mengunjungi setiap elemen dalam struktur data)
        bantu = head;
        int nomor = 1;

        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }

        baru->next = bantu->next;
        bantu->next = baru;
    }
}

// Delete First
void deleteFirst()
{
    Node *hapus;

    if (cekNode() == false)
    {
        if (head->next != NULL)
        {
            hapus = head;

```

```

        head = head->next;
        delete hapus;
    }
    else
    {
        head = tail = NULL;
    }
}
else
{
    cout << "Kosong!" << endl;
}
}

// Delete Last
void deleteLast()
{
    Node *hapus;
    Node *bantu;

    if (cekNode() == false)
    {
        if (head != tail)
        {
            hapus = tail;
            bantu = head;

            while (bantu->next != tail)
            {
                bantu = bantu->next;
            }

            tail = bantu;
            tail->next = NULL;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
        }
    }
    else
    {
        cout << "Kosong!" << endl;
    }
}

// Delete Middle
void deleteMiddle(int posisi)
{
    Node *hapus, *bantu, *bantu2;

    if (posisi < 1 || posisi > hitungNode())
    {
        cout << "Posisi di luar jangkauan." << endl;
    }
}

```

```

    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah." << endl;
    }
    else
    {
        int nomor = 1;
        bantu = head;

        while (nomor <= posisi)
        {
            if (nomor == posisi - 1)
            {
                bantu2 = bantu;
            }

            if (nomor == posisi)
            {
                hapus = bantu;
            }

            bantu = bantu->next;
            nomor++;
        }

        bantu2->next = bantu;
        delete hapus;
    }
}

// Change Fisrt
void changeFirst(string name, int age)
{
    if (cekNode() == false)
    {
        head->Nama = name;
        head->Umur = age;
    }
    else
    {
        cout << "Tidak ada perubahan data." << endl;
    }
}

// Change Middle
void changeMiddle(string name, int age, int posisi)
{
    Node *bantu;

    if (cekNode() == false)
    {
        if (posisi < 1 || posisi > hitungNode())
        {
            cout << "Posisi di luar jangkauan." << endl;

```

```

    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah." << endl;
    }
    else
    {
        bantu = head;
        int nomor = 1;

        while (nomor < posisi)
        {
            bantu = bantu->next;
            nomor++;
        }

        bantu->Nama = name;
        bantu->Umur
    = age;
    }
}
else
{
    cout << "Tidak ada data." << endl;
}
}

```

```

// Change Last
void changeLast(string name, int age)
{
    if (cekNode() == false)
    {
        tail->Nama = name;
        tail->Umur
    = age;
    }
    else
    {
        cout << "Tidak ada data." << endl;
    }
}

```

```

// Delete Node
void remove()
{
    Node *bantu, *hapus;
    bantu = head;

    while (bantu != NULL)
    {
        hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
}

```

```

    head = tail = NULL;
    cout << "Menghapus semua node" << endl;
}

// Tampilkan Linked List
void print()
{
    Node *bantu;
    bantu = head;

    cout << left << setw(15) << "    Nama " << right << setw(4) << " Usia " << endl; //
    Supaya rapi
    cout << "    ----- " << " -----" << endl;

    if (cekNode() == false)
    {
        while (bantu != NULL)
        {
            cout << left << setw(15) << bantu->Nama << right << setw(4) << bantu->Umur
            << endl; // Supaya lurus di output
            bantu = bantu->next;
        }

        cout << endl;
    }
    else
    {
        cout << "Tidak ada data." << endl;
    }
}

int main()
{
    inisialisasi();
    //Jawaban A
    cout << "\n// Tambahkan nama Anda" << endl;
    addFirst("Karin", 18);
    addFirst("Hoshino", 18);
    addFirst("Akechi", 20);
    addFirst("Yusuke", 19);
    addFirst("Michael", 18);
    addFirst("Jane", 20);
    addFirst("John", 19);
    addFirst("Anisah Syifa", 18);
    print();

    // Jawaban B
    cout << "// Hapus data 'Akechi' " << endl;
    deleteMiddle(6);
    print();

    // Jawaban C
    cout << "// Tambah data 'Futaba (18)' di antara John & Jane" << endl;
    addMiddle("Futaba", 18, 3);
    print();
}

```

```

// Jawaban D
cout << "// Tambah data 'Igor (20)' di awal" << endl;
addFirst("Igor", 20);
print();

// Jawaban E dan F
cout << "// Ubah data 'Michael' menjadi 'Reyn (18)'" << endl;
cout << "// Tampilan Akhir" << endl;
changeMiddle("Reyn", 18, 6);
print();

return 0;
}

```

Screenshots Output

```

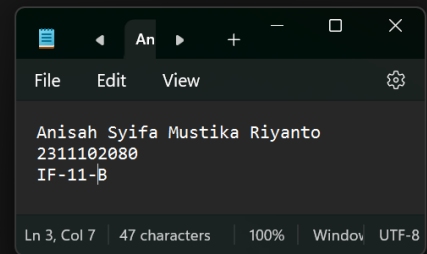
PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC\Modul 3> cd "d:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC\Modul 3"
; if ($?) { g++ Unguided1.cpp -o Unguided1 } ; if ($?) { .\Unguided1 }

// Tambahkan nama Anda
  Nama      Usia
  -----
Anisah Syifa 18
John          19
Jane          20
Michael       18
Yusuke        19
Akechi        20
Hoshino       18
Karin         18

// Hapus data 'Akechi'
  Nama      Usia
  -----
Anisah Syifa 18
John          19
Jane          20
Michael       18
Yusuke        19
Hoshino       18
Karin         18

// Tambah data 'Futaba (18)' di antara John & Jane
  Nama      Usia
  -----
Anisah Syifa 18
John          19
Futaba        18
Jane          20
Michael       18
Yusuke        19

```



```
Futaba      18
Jane       20
Michael    18
Yusuke     19
Hoshino    18
Karin      18

// Tambah data 'Igor (20)' di awal
Nama      Usia
-----
Igor      20
Anisah Syifa 18
John      19
Futaba    18
Jane      20
Michael   18
Yusuke    19
Hoshino   18
Karin     18

// Ubah data 'Michael' menjadi 'Reyn (18)'
// Tampilan Akhir
Nama      Usia
-----
Igor      20
Anisah Syifa 18
John      19
Futaba    18
Jane      20
Reyn      18
Yusuke    19
Hoshino   18
Karin     18

PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC\Modul 3>
```

File Edit View

Anisah Syifa Mustika Riyanto
2311102080
IF-11-B

Ln 3, Col 7 47 characters 100% Window UTF-8

Deskripsi:

Program di atas adalah implementasi dari Single Linked List dalam C++. Program di atas mencakup berbagai fungsi untuk menambah, menghapus, dan mengubah data dalam linked list, serta fungsi untuk menampilkan isi linked list. Pertama, sebuah struktur Node dideklarasikan untuk merepresentasikan elemen-elemen dalam linked list. Kemudian, fungsi-fungsi seperti addFirst, addLast, deleteFirst, deleteLast, dan lain-lain diimplementasikan untuk memodifikasi linked list sesuai kebutuhan. Di dalam main, contoh penggunaan fungsi-fungsi tersebut ditunjukkan dengan cara menambah, menghapus, dan mengubah data dalam linked list.

Unguided 2

Modifikasi Guided Double Linked List dilakukan dengan penambahan operasi untuk menambah data, menghapus, dan update di tengah / di urutan tertentu yang diminta. Selain itu, buatlah agar tampilannya menampilkan Nama produk dan harga.

Nama Produk	Harga
Originote	60.000
Somethinc	150.000
Skintific	100.000

Wardah	50.000
Hanasui	30.000

Case:

1. Tambahkan produk Azarine dengan harga 65000 diantara Somethinc dan Skintific
2. Hapus produk wardah
3. Update produk Hanasui menjadi Cleora dengan harga 55.000
4. Tampilkan menu seperti dibawah ini

Toko Skincare Purwokerto

1. ***Tambah Data***
2. ***Hapus Data***
3. ***Update Data***
4. ***Tambah Data Urutan Tertentu***
5. ***Hapus Data Urutan Tertentu***
6. ***Hapus Seluruh Data***
7. ***Tampilkan Data***
8. ***Exit***

Pada menu 7, tampilan akhirnya akan menjadi seperti dibawah ini :

Nama Produk	Harga
Originote	60.000
Somethinc	150.000
Azarine	65.000
Skintific	100.000
Cleora	55.000

Source code:

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

// Deklarasi Class Node
class Node
{
public:
    string namaProduk;
    int harga;
    Node *prev;
```

```

    Node *next;
};

// Deklarasi Class DoubleLinkedList
class DoubleLinkedList
{
public:
    Node *head;
    Node *tail;
    DoubleLinkedList()
    {
        head = nullptr;
        tail = nullptr;
    }

    // Menambahkan produk ke dalam linked list di urutan atas
    void addFirstProduct(string namaProduk, int harga)
    {
        Node *newNode = new Node;
        newNode->namaProduk = namaProduk;
        newNode->harga = harga;
        newNode->prev = nullptr;
        newNode->next = head;
        if (head != nullptr)
        {
            head->prev = newNode;
        }
        else
        {
            tail = newNode;
        }
        head = newNode;
    }

    // Menghapus produk teratas pada linked list
    void removeFirstProduct()
    {
        if (head == nullptr)
        {
            return;
        }
        Node *temp = head;
        head = head->next;
        if (head != nullptr)
        {
            head->prev = nullptr;
        }
        else
        {
            tail = nullptr;
        }
        delete temp;
    }

    // Mengubah data

```

```

bool changeProduct(string namaProdukLama, string namaProdukBaru, int
hargaBaru)
{
    Node *current = head;
    while (current != nullptr)
    {
        if (current->namaProduk == namaProdukLama)
        {
            current->namaProduk = namaProdukBaru;
            current->harga = hargaBaru;
            return true;
        }
        current = current->next;
    }
    return false; // Mengembalikan false jika data produk tidak ditemukan
}

// Menambahkan data produk pada posisi tertentu
void addFirstProductSpecificPosition(string namaProduk, int harga, int posisi)
{
    if (posisi < 1)
    {
        cout << "Posisi tidak valid." << endl;
        return;
    }
    Node *newNode = new Node;
    newNode->namaProduk = namaProduk;
    newNode->harga = harga;
    if (posisi == 1)
    { // Jika posisi adalah 1 maka tambahkan data produk di depan linked list
        newNode->next = head;
        newNode->prev = nullptr;
        if (head != nullptr)
        {
            head->prev = newNode;
        }
        else
        {
            tail = newNode;
        }
        head = newNode;
        return;
    }
    Node *current = head;
    for (int i = 1; i < posisi - 1 && current != nullptr; ++i)
    { // Looping sampai posisi sebelum posisi yang diinginkan
        current = current->next;
    }
    if (current == nullptr)
    {
        cout << "Posisi tidak valid" << endl;
        return;
    }
    newNode->next = current->next;
    newNode->prev = current;
}

```

```

    if (current->next != nullptr)
    {
        current->next->prev = newNode;
        // Pointer prev node setelah current menunjuk ke newNode jika node setelah
current tidak nullptr
    }
    else
    {
        tail = newNode;
    }
    current->next = newNode;
}

// Menghapus data pada posisi tertentu
void removeSpecificPosition(int posisi)
{
    if (posisi < 1 || head == nullptr)
    {
        cout << "Posisi tidak valid / list kosong. " << endl;
        return;
    }
    Node *current = head;
    if (posisi == 1)
    {
        head = head->next;
        if (head != nullptr)
        {
            head->prev = nullptr;
        }
        else
        {
            tail = nullptr;
        }
        delete current;
        return;
    }
    for (int i = 1; current != nullptr && i < posisi; ++i)
    { // Looping sampai posisi yang diinginkan
        current = current->next;
    }
    if (current == nullptr)
    {
        cout << "Posisi tidak valid" << endl;
        return;
    }
    if (current->next != nullptr)
    {
        current->next->prev = current->prev;
    }
    else
    {
        tail = current->prev;
    }
    current->prev->next = current->next;
    delete current;
}

```

```

}

// Menghapus semua data produk
void removeAll()
{
    Node *current = head;
    while (current != nullptr)
    {
        Node *temp = current;
        current = current->next;
        delete temp;
    }
    head = nullptr;
    tail = nullptr;
}

// Menampilkan data
void print()
{
    Node *current = head;
    cout << "\n=== List Produk dan Harga ===" << endl;
    cout << left << setw(20) << "Nama Produk"
        << "Harga" << endl;
    while (current != nullptr)
    {
        cout << left << setw(20) << current->namaProduk << current->harga << endl;
        current = current->next;
    }
    cout << endl;
}
};

int main()
{
    DoubleLinkedList list; // Deklarasi objek list dari class DoubleLinkedList

    list.addFirstProduct("Hanasui", 30000);
    list.addFirstProduct("Wardah", 50000);
    list.addFirstProduct("Skintific", 100000);
    list.addFirstProduct("Somethinc", 150000);
    list.addFirstProduct("Originote", 60000);

    list.print();

    while (true)
    {
        cout << "\nPilihan Menu: " << endl;
        cout << "1. Tambah Data" << endl;
        cout << "2. Hapus Data" << endl;
        cout << "3. Update Data" << endl;
        cout << "4. Tambah Data Urutan Tertentu" << endl;
        cout << "5. Hapus Data Urutan Tertentu" << endl;
        cout << "6. Hapus Seluruh Data" << endl;
        cout << "7. Tampilkan Data" << endl;
        cout << "8. Exit" << endl;
    }
}

```

```

int pilihan;
cout << "Pilih Menu: ";
cin >> pilihan;
switch (pilihan)
{
case 1:
{
    string namaProduk;
    int harga;
    cout << "Masukkan nama produk: ";
    cin >> namaProduk;
    cout << "Masukkan harga: ";
    cin >> harga;
    list.addFirstProduct(namaProduk, harga);
    cout << "Produk berhasil ditambahkan di posisi paling atas." << endl;
    list.print();
    break;
}
case 2:
{
    list.removeFirstProduct();
    cout << "Produk paling atas berhasil dihapus." << endl;
    list.print();
    break;
}
case 3:
{
    string namaProdukLama, namaProdukBaru;
    int hargaBaru;
    cout << "Input nama produk lama: ";
    cin >> namaProdukLama;
    cout << "Input nama produk baru: ";
    cin >> namaProdukBaru;
    cout << "Input harga baru: ";
    cin >> hargaBaru;
    bool updated = list.changeProduct(namaProdukLama, namaProdukBaru,
hargaBaru);
    if (!updated)
    {
        cout << "--Data produk tidak ditemukan--" << endl;
    }
    else
    {
        cout << "--Data berhasil diupdate--" << endl;
    }
    list.print();
    break;
}
case 4:
{
    string namaProduk;
    int harga, position;
    cout << "Input nama produk: ";
    cin >> namaProduk;
    cout << "Input harga: ";

```

```

        cin >> harga;
        cout << "Input posisi: ";
        cin >> position;
        list.addFirstProductSpecificPosition (namaProduk, harga, position);
        cout << "Produk berhasil ditambahkan pada posisi ke-" << position << endl;
        list.print();
        break;
    }
    case 5:
    {
        int position;
        cout << "Input posisi yang ingin dihapus: ";
        cin >> position;
        list.removeSpecificPosition(position);
        list.print();
        break;
    }
    case 6:
    {
        list.removeAll();
        break;
    }
    case 7:
    {
        list.print();
        break;
    }
    case 8:
    {
        return 0;
    }
    default:
    {
        cout << "Input Invalid" << endl;
        break;
    }
}
return 0;
}

```

Screenshots Output

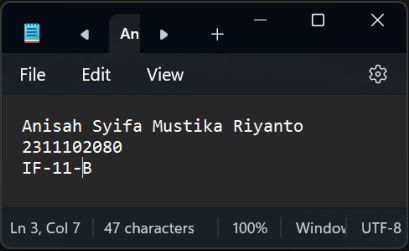
Case 1:

```
oft-MIEngine-Out-wspos0heg.mxl' '--stderr=Microsoft-MIEngine-Error-b2cc2iti.oyj' '--pid=Microsoft-MIEngine-Pid-11bb3j3w.w
w4' '--dbgExe=C:\Program Files\CodeBlocks\MinGW\bin\gdb.exe' '--interpreter=mi'

=== List Produk dan Harga ===
Nama Produk      Harga
Originote        60000
Somethinc         150000
Skintific         100000
Wardah            50000
Hanasui           30000

Pilihan Menu:
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan Data
8. Exit
Pilih Menu: 4
Input nama produk: Azarine
Input harga: 65000
Input posisi: 3
Produk berhasil ditambahkan pada posisi ke-3

=== List Produk dan Harga ===
Nama Produk      Harga
Originote        60000
Somethinc         150000
Azarine           65000
Skintific         100000
Wardah            50000
Hanasui           30000
```

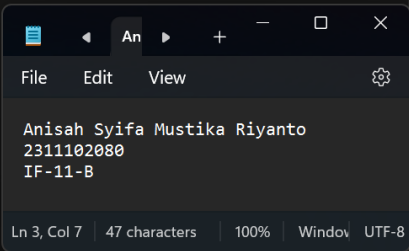


Case 2:

```
=== List Produk dan Harga ===
Nama Produk      Harga
Originote        60000
Somethinc         150000
Azarine           65000
Skintific         100000
Wardah            50000
Hanasui           30000

Pilihan Menu:
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan Data
8. Exit
Pilih Menu: 5
Input posisi yang ingin dihapus: 5

=== List Produk dan Harga ===
Nama Produk      Harga
Originote        60000
Somethinc         150000
Azarine           65000
Skintific         100000
Hanasui           30000
```

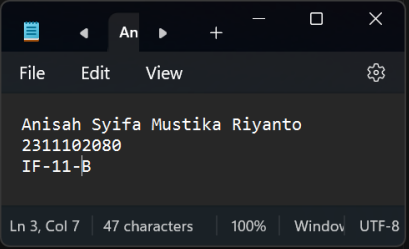


Case 3:

```
=== List Produk dan Harga ===
Nama Produk      Harga
Originote        60000
Somethinc         150000
Azarine          65000
Skintific         100000
Hanasui          30000

Pilihan Menu:
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan Data
8. Exit
Pilih Menu: 3
Input nama produk lama: Hanasui
Input nama produk baru: Cleora
Input harga baru: 55000
--Data berhasil diupdate--

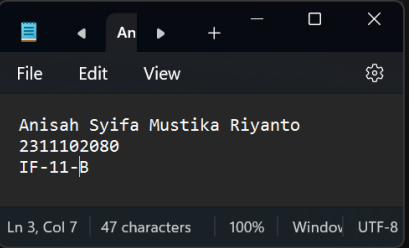
=== List Produk dan Harga ===
Nama Produk      Harga
Originote        60000
Somethinc         150000
Azarine          65000
Skintific         100000
Cleora           55000
```



Case 4:

```
Pilihan Menu:
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan Data
8. Exit
Pilih Menu: 7

=== List Produk dan Harga ===
Nama Produk      Harga
Originote        60000
Somethinc         150000
Azarine          65000
Skintific         100000
Cleora           55000
```



Deskripsi:

Program di atas merupakan implementasi double linked list dalam bahasa C++. Double linked list adalah struktur data linear yang terdiri dari simpul-simpul yang saling terhubung dengan dua pointer, yaitu pointer ke simpul sebelumnya (prev) dan pointer ke simpul berikutnya (next). Setiap simpul menyimpan informasi tentang nama produk dan harganya. Class DoubleLinkedList memiliki beberapa fungsi, antara lain untuk menambah produk ke dalam linked list di urutan atas (addFirstProduct), menghapus produk teratas pada linked list (removeFirstProduct), mengubah data produk (changeProduct), menambah data produk pada posisi tertentu (addFirstProductSpecificPosition), menghapus data produk pada posisi tertentu (removeSpecificPosition), serta menghapus semua data produk (removeAll). Di dalam fungsi main, program memungkinkan pengguna untuk memilih berbagai operasi melalui menu, seperti menambah data, menghapus data, mengubah data, menampilkan data, dan keluar dari program.

D. Kesimpulan

Linked list adalah sebuah struktur untuk menyimpan data yang bersifat dinamik dan terhubung satu sama lain. Beberapa operasi yang dapat diterapkan pada linked list seperti sisip(insert),hapus(delete), dan ubah(change). Terdapat dua jenis Linked list yakni Single Linked List dan Double Linked List, perbedaannya terdapat pada field yang ada pada node, jika pada Single Linked List field yang terdapat pada node adalah data dan pointer next, maka pada Double Linked List terdapat satu tambahan pointer yakni pointer prev. Terdapat Head yang menjadi node pertama dalam list, dan tail yang menjadi node terakhir dalam list. Pointer prev pada head menunjuk ke NULL dan pointer next pada tail juga menunjuk ke NULL. Null adalah suatu kondisi khusus dimana pointer itu belum di set dengan sebuah address tertentu, artinya pointer tidak menunjuk ke alamat manapun.

E. Referensi

Astuti, I. K. (2019). STRUKTUR DATA LINKED LIST.

Buana, I. K. S., Kom, S., Nata, G. N. M., Arnawa, I. B. K. S., Kom, S., & Kom, M. (2018). STRUKTUR DATA. Penerbit Andi.

Mukharil Bachtiar, A. (2012). Double Linked List (pelengkap).

Putra, A. k. (2019, April 25). Single linked list. Diakses 29 Maret 2024, dari <https://doi.org/10.31219/osf.io/u6qf7>

Siregar, A. A. N. (2019). Pengertian Linked Object.