

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL III
SINGLE AND DOUBLE LINKED LIST**



Disusun Oleh :

NAMA : Tegar Serli Arunzika
NIM 2311102041

Dosen

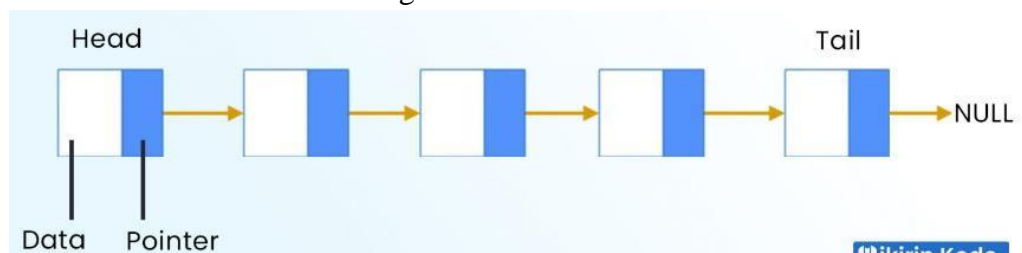
Wahyu Andi Saputra, S.Pd., M.Eng

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

A. Dasar Teori

a. Single Linked List

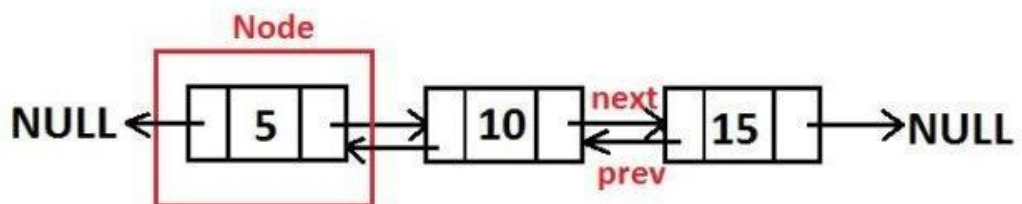
Single Linked List atau Singly Linked List merupakan linked list yang hanya memiliki satu variabel pointer untuk menunjuk ke node lainnya, variabel pointer ini biasanya dinamakan next. Pada dasarnya Single Linked List ini adalah linked list yang berbentuk umum seperti yang dijelaskan sebelumnya. Seperti yang dapat dilihat dari ilustrasi di atas, terdapat 5 node yang terhubung menjadi suatu single linked list. Node pertama biasa disebut head, pointer pada node ini menunjuk ke node selanjutnya dan begitu seterusnya hingga node terakhir yang pointernya menunjuk ke NULL. Hal itu menandakan bahwa node itu adalah node terakhir atau biasa disebut dengan node tail.



b. Double Linked List

Dalam pembahasan artikel sebelumnya telah diperkenalkan Single Linked List, yakni linked list dengan sebuah pointer penghubung. Dalam artikel ini, dibahas pula varian linked list dengan 2 pointer penunjuk, yakni Doubly linked list yang memiliki pointer penunjuk 2 arah, yakni ke arah node sebelum (previos/prev) dan node sesudah (next).

Representasi sebuah doubly linked list dapat dilihat pada gambar berikut ini:



Di dalam sebuah linked list, ada 2 pointer yang menjadi penunjuk utama,

yakni pointer HEAD yang menunjuk pada node pertama di dalam linked list itu sendiri dan pointer TAIL yang menunjuk pada node paling akhir di dalam linked list. Sebuah linked list dikatakan kosong apabila isi pointer head adalah NULL. Selain itu, nilai pointer prev dari HEAD selalu NULL, karena merupakan data pertama. Begitu pula dengan pointer next dari TAIL yang selalu bernilai NULL sebagai penanda data terakhir.

B. Guided

Guided 1

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

struct Node
{
    int data;
    Node *next;
};

Node *head;
Node *tail;

void init()
{
    head = NULL;
    tail = NULL;
}

bool isEmpty()
{
    return head == NULL;
}

void insertDepan(int nilai)
{
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}

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;
    }
}

int hitungList()
{
    Node *hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

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();
        baru->data = data;
        Node *bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1)

```

```

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

void hapusDepan()
{
    if (!isEmpty())
    {
        Node *hapus = head;
        if (head->next != NULL)
        {
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
            delete hapus;
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}

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;
}
}

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;
        delete hapus;
    }
}

void ubahDepan(int data)
{
    if (!isEmpty())
    {
        head->data = data;
    }
}

```

```

        else
        {
            cout << "List masih kosong!" << endl;
        }
    }

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++)
            {
                bantu = bantu->next;
            }
            bantu->data = data;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

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

```



```

}

void clearList()
{
    Node *bantu = head;
    while (bantu != NULL)
    {
        Node *hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}

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

The screenshot shows a Windows PowerShell terminal window and a Notepad text editor. The PowerShell window displays the output of a program, which is a sequence of numbers: 3, 5, 2, 3, 5, 1, 2, 3, 5, 2, 3, 2, 7, 3, 2, 3, 1, 3, 1, 8, 1, 11. The Notepad window shows the following text:

```

Nama : Tegar Serli arunzika
Nim : 2311102041
Kelas If-11_B

```

Deskripsi: Program di atas adalah sebuah program untuk mengelola sebuah Single linked list yang berisi bilangan bulat. Program ini dapat melakukan beberapa operasi dasar seperti menambahkan data di depan, belakang, atau di tengah linked list, menghapus data di depan, belakang, atau di tengah linked list, mengubah nilai data di depan, belakang, atau di tengah linked list, menampilkan isi linked list, dan menghapus semua data linked list. Data pada linked list ini terdiri dari sebuah struct Node yang berisi int data untuk menyimpan data bilangan bulat dan Node *next untuk menunjukkan ke data selanjutnya dalam linked list.

Dalam program ini terdapat beberapa fungsi yaitu:

- `init()`: Menginisialisasi head dan tail linked list menjadi NULL.

- isEmpty(): Mengembalikan true jika linked list kosong.
- insertDepan(int nilai): Menambahkan data baru di depan linked list.
- insertBelakang(int nilai): Menambahkan data baru di belakang linked list.
- hitungList(): Menghitung jumlah data dalam linked list.
- insertTengah(int data, int posisi): Menambahkan data baru di posisi tertentu dalam linked list.
- hapusDepan(): Menghapus data pertama dalam linked list.
- hapusBelakang(): Menghapus data terakhir dalam linked list.
- hapusTengah(int posisi): Menghapus data pada posisi tertentu dalam linked list.
- ubahDepan(int data): Mengubah data pertama dalam linked list.
- ubahTengah(int data, int posisi): Mengubah data pada posisi tertentu dalam linked list.
- ubahBelakang(int data): Mengubah data terakhir dalam linked list.
- clearList(): Menghapus semua data dalam linked list.
- tampil(): Menampilkan seluruh data dalam linked list.

semua fungsi diatas digunakan didalam fungsi main() untuk menjalankan berbagai operasi pada linked list seperti menambahkan, menghapus, mengubah, dan menampilkan data-data dalam linked list.

Guided 2

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

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;
    }
}
```

```

void pop()
{
    if (head == nullptr)
    {
        return;
    }
    Node *temp = head;
    head = head->next;

    if (head != nullptr)
    {
        head->prev = nullptr;
    }
    else
    {
        tail = nullptr;
    }

    delete temp;
}

bool update(int oldData, int newData)
{
    Node *current = head;

    while (current != nullptr)
    {
        if (current->data == oldData)
        {
            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;
    }
    cout << endl;
}

};

int main()
{
    DoublyLinkedList list;
    while (true)
    {
        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;
            }

```

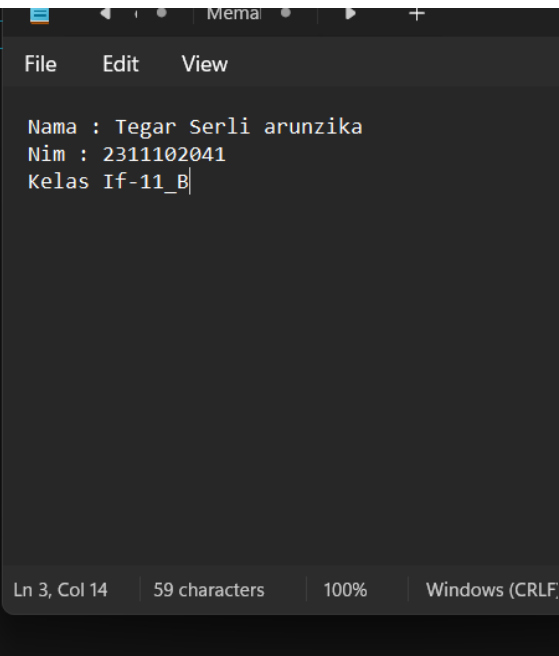
```

    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;
}

```

Screenshots Output

```
ngine-In-ienvutg2.ppt' '--stdout=Microsoft-  
0b' '--dbgExe=C:\TDM-GCC-64\bin\gdb.exe' '-  
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: 20  
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: 30  
1. Add data  
2. Delete data  
3. Update data  
4. Clear data  
5. Display data  
6. Exit
```

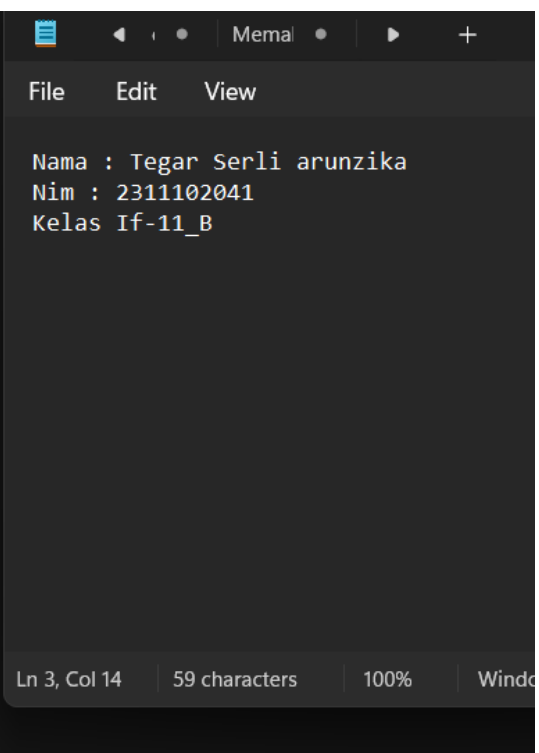


File Edit View

Nama : Tegar Serli arunzika
Nim : 2311102041
Kelas If-11_B

Ln 3, Col 14 | 59 characters | 100% | Windows (CRLF)

```
Enter your choice: 2  
1. Add data  
2. Delete data  
3. Update data  
4. Clear data  
5. Display data  
6. Exit  
Enter your choice: 3  
Enter old data: 20  
Enter new data: 30  
1. Add data  
2. Delete data  
3. Update data  
4. Clear data  
5. Display data  
6. Exit  
Enter your choice: 5  
30  
1. Add data  
2. Delete data  
3. Update data  
4. Clear data  
5. Display data  
6. Exit
```



File Edit View

Nama : Tegar Serli arunzika
Nim : 2311102041
Kelas If-11_B

Ln 3, Col 14 | 59 characters | 100% | Windo


```
Enter your choice: 2
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: 6
PS C:\Users\M S I> 
```

File Edit View

Nama : Tegar Serli arunzika
Nim : 2311102041
Kelas If-11_B

Deskripsi: Program di atas adalah sebuah program yang mengimplementasikan Doubly Linked List menggunakan class. Program ini dapat melakukan beberapa operasi dasar pada Doubly Linked List, seperti menambahkan data, menghapus data, mengubah data, menghapus semua data, dan menampilkan data.

Didalam program terdapat 2 class yaitu:

- Node: class ini merepresentasikan simpul atau node dalam Doubly Linked List. Setiap node memiliki tiga anggota data yaitu int data untuk menyimpan nilai data, Node *prev untuk menunjukkan ke simpul sebelumnya, dan Node *next untuk menunjukkan ke simpul berikutnya.
- DoublyLinkedList: class ini berisi Doubly Linked List. Class ini memiliki dua anggota data yaitu Node *head yang menunjukkan ke simpul pertama dalam linked list, dan Node *tail yang menunjukkan ke simpul terakhir dalam linked list. Class ini juga dapat melakukan operasi pada linked list, seperti menambahkan data (push), menghapus data (pop), mengubah data (update), menghapus semua data (deleteAll), dan menampilkan data (display).

Di dalam fungsi `main()`, program menampilkan menu untuk pengguna. Pengguna dapat memilih menu untuk menambahkan data baru, menghapus data, mengubah data, menghapus semua data, menampilkan data, atau keluar dari program. Program ini menggunakan switch-case untuk membuat menu dan menjalankan pilihan pengguna. Berikut adalah pilihan yang terdapat pada menu

- Add data: Menambahkan data baru ke depan Doubly Linked List.
- Delete data: Menghapus data dari depan Doubly Linked List.
- Update data: Mengubah data tertentu dalam Doubly Linked List.
- Clear data: Menghapus seluruh data dari Doubly Linked List.
- Display data: Menampilkan seluruh data dalam Doubly Linked List.
- Exit: Keluar dari program.

Program menggunakan perulangan `do while` yang tidak akan berhenti mengulang sampai mendapatkan inputan dari pengguna untuk keluar program.

C. Unguided/Tugas

Unguided 1

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

struct Node
{
    string nama;
    int usia;
    Node *next;
};

Node *head;
Node *tail;

void init()
{
    head = NULL;
    tail = NULL;
}

bool isEmpty()
{
    return head == NULL;
}

void insertDepan(string nama, int usia)
{
    Node *baru = new Node;
    baru->nama = nama;
    baru->usia = usia;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}
```

```

void insertBelakang(string nama, int usia)
{
    Node *baru = new Node;
    baru->nama = nama;
    baru->usia = usia;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}

int hitungList()
{
    Node *hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

void insertTengah(string nama, int usia, 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();
        baru->nama = nama;

```

```

        baru->usia = usia;
        Node *bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

void hapusDepan()
{
    if (!isEmpty())
    {
        Node *hapus = head;
        if (head->next != NULL)
        {
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
            delete hapus;
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}

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;
}
}

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;
        delete hapus;
    }
}

void ubahDepan(string nama, int usia)
{

```

```

        if (!isEmpty())
        {
            head->nama = nama;
            head->usia = usia;
        }
        else
        {
            cout << "List masih kosong!" << endl;
        }
    }

void ubahTengah(string nama, int usia, 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++)
            {
                bantu = bantu->next;
            }
            bantu->nama = nama;
            bantu->usia = usia;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

void ubahBelakang(string nama, int usia)
{
    if (!isEmpty())
    {

```

```

        tail->nama = nama;
        tail->usia = usia;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

void clearList()
{
    Node *bantu = head;
    while (bantu != NULL)
    {
        Node *hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}

void tampil()
{
    if (!isEmpty())
    {
        Node *bantu = head;
        while (bantu != NULL)
        {
            cout << bantu->nama << " ";
            cout << bantu->usia << " , ";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

int main()
{
    init();

```



```

int menu, usia, posisi;
string nama;
cout << "\n# Menu Linked List Mahasiswa #" << endl;
do
{
    cout << "\n 1. Insert Depan"
        << "\n 2. Insert Belakang"
        << "\n 3. Insert Tengah"
        << "\n 4. Hapus Depan"
        << "\n 5. Hapus Belakang"
        << "\n 6. Hapus Tengah"
        << "\n 7. Ubah Depan"
        << "\n 8. Ubah Belakang"
        << "\n 9. Ubah Tengah"
        << "\n 10. Tampilkan"
        << "\n 0. Keluar Program"
        << "\n Pilihan : ";

    cin >> menu;
    switch (menu)
    {
        case 1:
            cout << "Masukkan Nama : ";
            cin >> nama;
            cout << "Masukkan Usia : ";
            cin >> usia;
            insertDepan(nama, usia);
            cout << endl;
            tampil();
            break;
        case 2:
            cout << "Masukkan Nama : ";
            cin >> nama;
            cout << "Masukkan Usia : ";
            cin >> usia;
            insertBelakang(nama, usia);
            cout << endl;
            tampil();
            break;
        case 3:
            cout << "Masukkan Posisi : ";
            cin >> posisi;
            cout << "Masukkan Nama : ";
            cin >> nama;
            cout << "Masukkan Usia : ";

```

```
        cin >> usia;
        insertTengah(nama, usia, posisi);
        cout << endl;
        tampil();
        break;
    case 4:
        hapusDepan();
        cout << endl;
        tampil();
        break;
    case 5:
        hapusBelakang();
        cout << endl;
        tampil();
        break;
    case 6:
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        hapusTengah(posisi);
        cout << endl;
        tampil();
        break;
    case 7:
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan Usia : ";
        cin >> usia;
        ubahDepan(nama, usia);
        cout << endl;
        tampil();
        break;
    case 8:
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan Usia : ";
        cin >> usia;
        ubahBelakang(nama, usia);
        cout << endl;
        tampil();
        break;
    case 9:
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        cout << "Masukkan Nama : ";
```

```

        cin >> nama;
        cout << "Masukkan Usia : ";
        cin >> usia;
        ubahTengah(nama, usia, posisi);
        cout << endl;
        tampil();
        break;
    case 10:
        tampil();
        break;

    default:
        cout << "Pilihan Salah" << endl;
        break;
    }
} while (menu != 0);
return 0;
}

```

Screenshots Output

The screenshot shows the output of a C++ program. It displays a menu with 10 options: 1. Insert Depan, 2. Insert Belakang, 3. Insert Tengah, 4. Hapus Depan, 5. Hapus Belakang, 6. Hapus Tengah, 7. Ubah Depan, 8. Ubah Belakang, 9. Ubah Tengah, 10. Tampilkan, and 0. Keluar Program. The user has entered '3' as a choice. The program then prompts for 'Masukkan Posisi : 5', 'Masukkan Nama : Aji', and 'Masukkan Usia : 18'. The output shows the resulting array: 'Tegar 20 , rahmat 23 , budi 20 , Yusuf 19 , Aji 18 , samsul 19 , fahmi 23 ,'. The menu is shown again, and the user has entered '6' as a choice. The program then prompts for 'Masukkan Posisi : 4'. The output shows the resulting array: 'Tegar 20 , rahmat 23 , budi 20 , Aji 18 , samsul 19 , fahmi 23 ,'.

```

1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 3
Masukkan Posisi : 5
Masukkan Nama : Aji
Masukkan Usia : 18

Tegar 20 , rahmat 23 , budi 20 , Yusuf 19 , Aji 18 , samsul 19 , fahmi 23 ,

1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 6
Masukkan Posisi : 4

Tegar 20 , rahmat 23 , budi 20 , Aji 18 , samsul 19 , fahmi 23 ,

```

```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 1
Masukkan Nama : Pradit
Masukkan Usia : 23
Pradit 23 , Tegar 20 , rahmat 23 , budi 20 , Aji 18 , samsul 19 , fahmi 23 ,

1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 9
Masukkan Posisi : 6
Masukkan Nama : rudi
Masukkan Usia : 18
Pradit 23 , Tegar 20 , rahmat 23 , budi 20 , Aji 18 , rudi 18 , fahmi 23 ,
```

```
File Edit View
Nama : Tegar Serli arunzika
Nim : 2311102041
Kelas If-11_B
Ln 3, Col 14 59 characters 100% Windows (CRLF) UTF-8
```

```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 10
Pradit 23 , Tegar 20 , rahmat 23 , budi 20 , Aji 18 , rudi 18 , fahmi 23 ,

1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 0
Pilihan Salah
PS C:\Users\VM S I>
```

```
File Edit View
Nama : Tegar Serli arunzika
Nim : 2311102041
Kelas If-11_B
Ln 3, Col 14 59 characters 100% Windows (CRLF) UTF-8
```

Deskripsi: Program di atas adalah program untuk mengelola linked list yang berisi data mahasiswa. Setiap node dalam linked list memiliki dua atribut yaitu nama (string) dan usia (integer). Program ini dapat melakukan beberapa operasi dasar pada linked list seperti menambahkan data di depan, belakang, atau di tengah linked list, menghapus data dari depan, belakang, atau dari posisi tertentu, mengubah data di depan, belakang, atau di tengah linked list, dan menampilkan semua data dalam linked list. Berikut adalah fungsi fungsi yang ada didalam program diatas :

- `init()`: Inisialisasi pointer head dan tail menjadi NULL untuk menandakan bahwa linked list masih kosong.
- `isEmpty()`: Fungsi yang mengembalikan nilai boolean true jika linked list kosong.
- `insertDepan(string nama, int usia)`: Menambahkan data mahasiswa baru di depan linked list.
- `insertBelakang(string nama, int usia)`: Menambahkan data mahasiswa baru di belakang linked list.

- `hitungList()`: Menghitung jumlah data dalam linked list.
- `insertTengah(string nama, int usia, int posisi)`: Menambahkan data mahasiswa baru pada posisi tertentu dalam linked list.
- `hapusDepan()`: Menghapus data mahasiswa dari depan linked list.
- `hapusBelakang()`: Menghapus data mahasiswa dari belakang linked list.
- `hapusTengah(int posisi)`: Menghapus data mahasiswa dari posisi tertentu dalam linked list.
- `ubahDepan(string nama, int usia)`: Mengubah data mahasiswa di depan linked list.
- `ubahTengah(string nama, int usia, int posisi)`: Mengubah data mahasiswa pada posisi tertentu dalam linked list.
- `ubahBelakang(string nama, int usia)`: Mengubah data mahasiswa di belakang linked list.
- `clearList()`: Menghapus semua data dalam linked list.
- `tampil()`: Menampilkan semua data mahasiswa dalam linked list.

Di dalam fungsi `main()`, program menampilkan menu pilihan untuk pengguna. Pengguna dapat memilih pilihan untuk Menambahkan data baru di depan, belakang, atau di tengah linked list, menghapus data, mengubah data, atau menampilkan semua data dalam linked list. Program menggunakan perulangan `do while` yang akan terus berulang sampai mendapatkan inputan dari pengguna untuk keluar dari program (memilih pilihan 0).

Unguided 2

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

class Node {
public:

    string product_name;
    float price;
    Node* prev;
    Node* next;
};

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

    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    void push(string product_name, float price) {
        Node* newNode = new Node;
        newNode->product_name = product_name;
        newNode->price = price;
        newNode->prev = nullptr;
        newNode->next = head;
        if (head != nullptr) {
            head->prev = newNode;
        } else {
            tail = newNode;
        }
        head = newNode;
    }

    void pop() {
        if (head == nullptr) {
            return;
        }
        Node* temp = head;
        head = head->next;
        if (head != nullptr) {
            head->prev = nullptr;
        }
        delete temp;
    }
};
```

```

    } else {
        tail = nullptr;
    }
    delete temp;
}

bool update(string oldProduct, string newProduct, float
newPrice)
{
    Node* current = head;
    while (current != nullptr) {
        if (current->product_name == oldProduct) {
            current->product_name = newProduct;
            current->price = newPrice;
            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->product_name << " (Rp" << current-
>price
        << ")" << endl;
        current = current->next;
    }
}

```

```

void insert(string product_name, float price, int position) {
    if (position <= 0) {
        push(product_name, price);
        return;
    }
    Node* current = head;
    for (int i = 1; i < position && current != nullptr; i++) {
        current = current->next;
    }

    if (current == nullptr) {
        Node* newNode = new Node;
        newNode->product_name = product_name;
        newNode->price = price;
        newNode->prev = tail;
        newNode->next = nullptr;
        if (tail != nullptr) {
            tail->next = newNode;
        } else {
            head = newNode;
        }
        tail = newNode;
    } else {
        Node* newNode = new Node;
        newNode->product_name = product_name;
        newNode->price = price;
        newNode->prev = current->prev;
        newNode->next = current;
        if (current->prev != nullptr) {
            current->prev->next = newNode;
        } else {
            head = newNode;
        }
        current->prev = newNode;
    }
}

void remove(int position) {
    if (position <= 0) {
        pop();
        return;
    }
}

```



```

    }
    Node* current = head;
    for (int i = 1; i < position && current != nullptr; i++) {
        current = current->next;
    }
    if (current == nullptr) {
        return;
    }
    if (current == head) {
        head = current->next;
    } else {
        current->prev->next = current->next;
    }
    if (current == tail) {
        tail = current->prev;
    } else {
        current->next->prev = current->prev;
    }
    delete current;
}
};

int main()
{
    DoublyLinkedList list;
    int choice;
    string productName, newProductName;
    float price, newPrice;
    int position;
    do {
        cout << "1. tambah data\n";
        cout << "2. hapus data\n";
        cout << "3. Update data \n";
        cout << "4. tambah data urutan tertentu insert\n";
        cout << "5. hapus data urutan tertentu remove\n";
        cout << "6. hapus seluruh data\n";
        cout << "7. tampilkan data\n";
        cout << "8. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:

```

```

        cout << "masukan nama produk: ";
        cin >> productName;
        cout << "masukan harga: ";
        cin >> price;
        list.push(productName, price);
        break;
    case 2:
        list.pop();
        break;
    case 3:
        cout << "masukan nama produk yang lama: ";
        cin >> productName;
        cout << "masukan nama produk yang baru: ";
        cin >> newProductName;
        cout << "masukan harga baru: ";
        cin >> newPrice;
        if (list.update(productName, newProductName,
newPrice))
        {
            cout << "Produk berhasil ditambahkan\n";
        } else {
            cout << "produk tidak ada\n";
        }
        break;
    case 4:
        cout << "masukan nama produk: ";
        cin >> productName;
        cout << "masukan harga: ";
        cin >> price;
        cout << "masukan posisi: ";
        cin >> position;
        list.insert(productName, price, position);
        break;
    case 5:
        cout << "masukan urutan ke-: ";
        cin >> position;
        list.remove(position);
        break;
    case 6:
        list.deleteAll();
        cout << "semua produk berhasil di hapus\n";
        break;
    case 7:
        list.display();
        break;

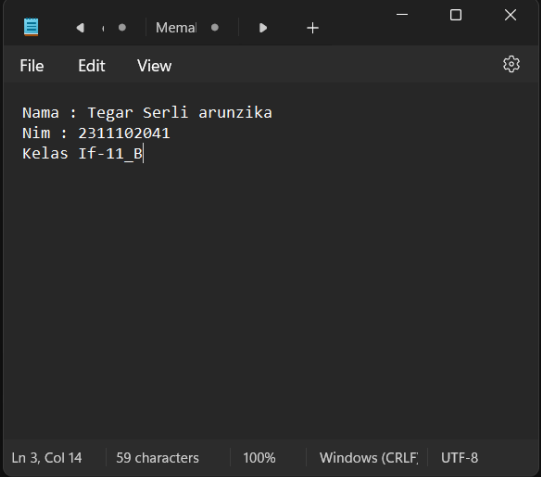
```

```
        list.display();
        break;
    case 8:
        cout << "keluar dari progam...\n";
        break;
    default:
        cout << "pilihan salah\n";
        break;
    }
} while (choice != 8);

return 0;
}
```

Screenshots Output

```
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 1
masukan nama produk: serum
masukan harga: 30000
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 1
masukan nama produk: toner
masukan harga: 12000
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 1
masukan nama produk: air-mawar
masukan harga: 8000
```

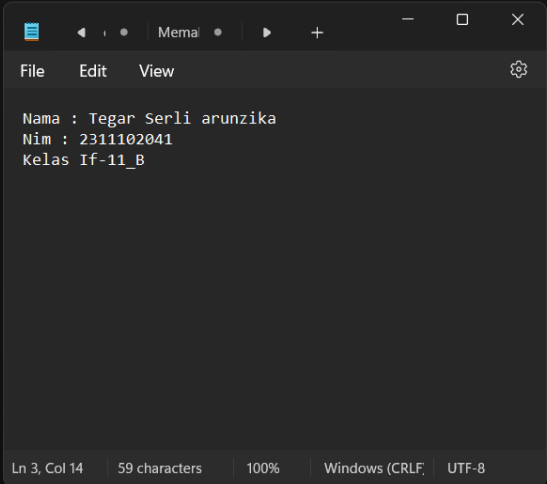


The screenshot shows a Notepad++ window with a menu bar (File, Edit, View) and a toolbar. The text content is as follows:

```
Nama : Tegar Serli arunzika
Nim : 2311102041
Kelas If-11_B
```

The status bar at the bottom indicates: Ln 3, Col 14 | 59 characters | 100% | Windows (CRLF) | UTF-8.

```
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 1
masukan nama produk: parfum
masukan harga: 15000
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 1
masukan nama produk: masker
masukan harga: 12000
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 7
masker (Rp12000)
parfum (Rp15000)
air-mawar (Rp8000)
toner (Rp12000)
serum (Rp30000)
```

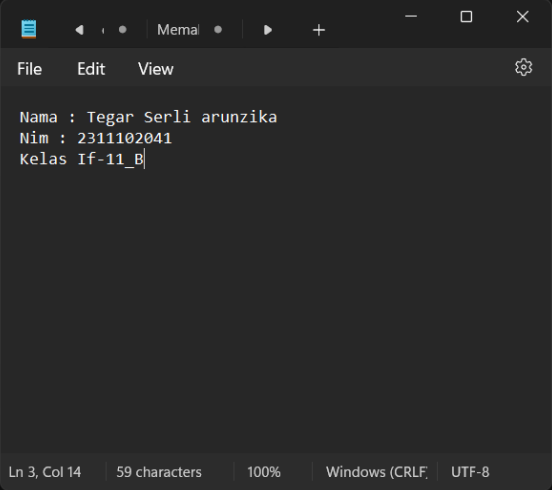


The screenshot shows a Notepad++ window with a menu bar (File, Edit, View) and a toolbar. The text content is as follows:

```
Nama : Tegar Serli arunzika
Nim : 2311102041
Kelas If-11_B
```

The status bar at the bottom indicates: Ln 3, Col 14 | 59 characters | 100% | Windows (CRLF) | UTF-8.

```
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 4
masukan nama produk: Maskara
masukan harga: 12000
masukan posisi: 3
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 7
masker (Rp12000)
parfum (Rp15000)
Maskara (Rp12000)
air-mawar (Rp8000)
toner (Rp12000)
serum (Rp30000)
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 
```



The screenshot shows a Notepad++ window with a menu bar (File, Edit, View) and a toolbar. The text content is as follows:

```
Nama : Tegar Serli arunzika
Nim : 2311102041
Kelas If-11_B
```

At the bottom of the window, the status bar displays: Ln 3, Col 14 | 59 characters | 100% | Windows (CRLF) | UTF-8.

Program ini adalah implementasi dari Doubly Linked List (DLL) yang digunakan untuk menyimpan data nama produk dan harga. Di dalam program ini terdapat class DoublyLinkedList yang memiliki head dan tail sebagai titik awal dan akhir dari DLL. Class ini juga memiliki beberapa method, yaitu push, pop, update, insert, remove, deleteAll, dan display.

Method push digunakan untuk menambah data baru di awal DLL, sedangkan pop digunakan untuk menghapus data pertama di DLL. Method update digunakan untuk mengubah data pada DLL dengan mencocokkan nama produk lama dan menginputkan nama produk baru beserta harga barunya. Method insert digunakan untuk menambah data baru pada posisi tertentu di DLL, sedangkan method remove digunakan untuk menghapus data pada posisi tertentu di DLL. Method deleteAll digunakan untuk menghapus semua data pada DLL. Selain itu, method display digunakan untuk menampilkan seluruh data yang ada pada DLL. Di dalam main program, terdapat looping do-while yang akan mengulangi menu utama hingga pengguna memilih untuk keluar. Pada setiap pilihan, akan dipanggil method yang sesuai dari class DoublyLinkedList.

D. Kesimpulan

Single Linked List adalah struktur data yang terdiri dari serangkaian node yang terhubung satu sama lain melalui pointer 'next', dimana node pertama disebut head dan node terakhir menunjuk ke NULL. Sedangkan Doubly Linked List adalah varian dari linked list yang memiliki dua pointer penunjuk, yaitu ke node sebelumnya ('previous/prev') dan ke node berikutnya ('next'). Doubly Linked List memungkinkan traversing maju dan mundur, memudahkan operasi seperti penghapusan atau penambahan node di antara dua node yang ada. Di kedua jenis linked list ini, HEAD menunjuk pada node pertama dan TAIL menunjuk pada node terakhir, dengan nilai NULL menandakan linked list kosong. Perbedaan utama antara keduanya adalah kemampuan Doubly Linked List untuk bergerak maju dan mundur, sementara Single Linked List hanya bergerak maju.

E. Referensi

Mikirin Kode. (n.d.). Single Linked List. Diakses pada 29 Maret 2024, dari <https://mikirinkode.com/single-linked-list/>

Sekolah Ilmu Komputer (SICS) Binus University. (2017, 15 Maret). Doubly Linked List. Diakses pada 29 Maret 2024, dari <https://socs.binus.ac.id/2017/03/15/doubly-linked-list/>