

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN
ALGORITMA**

**MODUL III
SINGLE & DOUBLE LINKED
LIST**



**Disusun Oleh :
Reza Alvonzo
2311102026**

Dosen Pengampu:
Muhammad Afrizal Amrustian, S. Kom., M. Kom

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2023**

BAB I

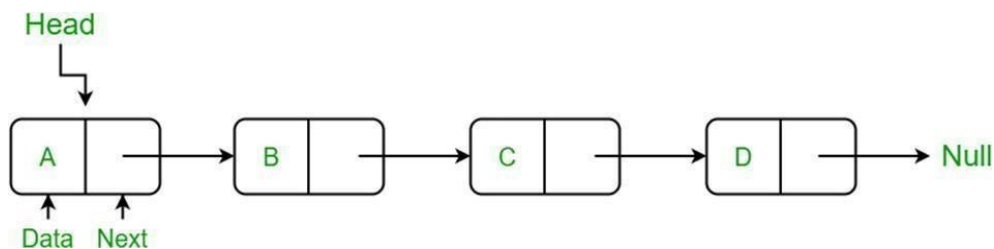
TUJUAN PRAKTIKUM

1. Mahasiswa memahami perbedaan konsep Single dan Double Linked List
2. Mahasiswa mampu menerapkan Single dan Double Linked List ke dalam pemrograman

BAB II DASAR TEORI

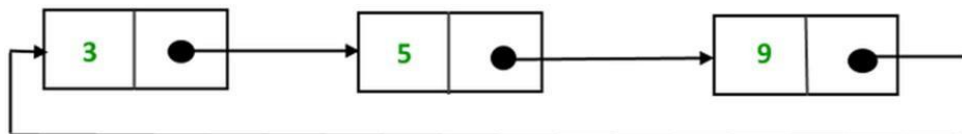
a.) Single Linked List

Linked List merupakan suatu bentuk struktur data yang berisi kumpulan data yang disebut sebagai node yang tersusun secara sekuensial, saling sambung menyambung, dinamis, dan terbatas. Setiap elemen dalam linked list dihubungkan ke elemen lain melalui pointer. Masing-masing komponen sering disebut dengan simpul atau node atau verteks. Pointer adalah alamat elemen. Setiap simpul pada dasarnya dibagi atas dua bagian pertama disebut bagian isi atau informasi atau data yang berisi nilai yang disimpan oleh simpul. Bagian kedua disebut bagian pointer yang berisi alamat dari node berikutnya atau sebelumnya. Dengan menggunakan struktur seperti ini, linked list dibentuk dengan cara menunjuk pointer next suatu elemen ke elemen yang mengikutinya. Pointer next pada elemen terakhir merupakan NULL, yang menunjukkan akhir dari suatu list. Elemen pada awal suatu list disebut head dan elemen terakhir dari suatu list disebut tail.



Dalam operasi Single Linked List, umumnya dilakukan operasi penambahan dan penghapusan simpul pada awal atau akhir daftar, serta pencarian dan pengambilan nilai pada simpul tertentu dalam daftar. Karena struktur data ini hanya memerlukan satu pointer untuk setiap simpul, maka Single Linked List umumnya lebih efisien dalam penggunaan memori dibandingkan dengan jenis Linked List lainnya, seperti Double Linked List dan Circular Linked List.

Single linked list yang kedua adalah circular linked list. Perbedaan circular linked list dan non circular linked adalah penunjuk next pada node terakhir pada circular linked list akan selalu merujuk ke node pertama.



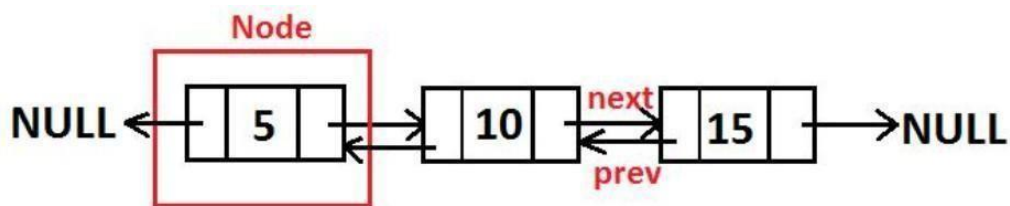
b.) Double Linked List

Double Linked List adalah struktur data Linked List yang mirip dengan Single Linked List, namun dengan tambahan satu pointer tambahan pada setiap simpul yaitu pointer prev yang menunjuk ke simpul sebelumnya.

Dengan adanya pointer prev, Double Linked List memungkinkan untuk melakukan operasi penghapusan dan penambahan pada simpul mana saja secara efisien. Setiap simpul pada Double Linked List memiliki tiga elemen penting, yaitu elemen data (biasanya berupa nilai), pointer next yang menunjuk ke simpul berikutnya, dan pointer prev yang menunjuk ke simpul sebelumnya.

Keuntungan dari Double Linked List adalah memungkinkan untuk melakukan operasi penghapusan dan penambahan pada simpul dimana saja dengan efisien, sehingga sangat berguna dalam implementasi beberapa algoritma yang membutuhkan operasi tersebut. Selain itu, Double Linked List juga memungkinkan kita untuk melakukan traversal pada list baik dari depan (head) maupun dari belakang (tail) dengan mudah. Namun, kekurangan dari Double Linked List adalah penggunaan memori yang lebih besar dibandingkan dengan Single Linked List, karena setiap simpul membutuhkan satu pointer tambahan. Selain itu, Double Linked List juga membutuhkan waktu eksekusi yang lebih lama dalam operasi penambahan dan penghapusan jika dibandingkan dengan Single Linked List

Representasi sebuah double 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.

BAB III

GUIDED

Guided 1 (Latihan Single Linked List)

```
#include <iostream>
using namespace std;
/// PROGRAM SINGLE LINKED LIST NON-CIRCULAR
// Deklarasi Struct Node
struct Node
{
    // komponen/member
    int data;
    string kata;
    Node *next;
};
Node *head;
Node *tail;
// Inisialisasi Node
void init()
{
    head = NULL;
    tail = NULL;
}
// Pengecekan
bool isEmpty()
{
    if (head == NULL)
        return true;
    else
        return false;
}
// Tambah Depan
void insertDepan(int nilai, string kata)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->kata = kata;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}
// Tambah Belakang
void insertBelakang(int nilai, string kata)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->kata = kata;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
```

```

        tail->next = NULL;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}
// Hitung Jumlah List
int hitungList()
{
    Node *hitung;
    hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}
// Tambah Tengah
void insertTengah(int data, string kata, 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, *bantu;
        baru = new Node();
        baru->data = data;
        baru->kata = kata;
        // tranversing
        bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}
// Hapus Depan
void hapusDepan()
{
    Node *hapus;
    if (isEmpty() == false)
    {
        if (head->next != NULL)
        {
            hapus = head;
            head = head->next;
            delete hapus;
        }
    }
}

```

```

        else
        {
            head = tail = NULL;
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}
// Hapus Belakang
void hapusBelakang()
{
    Node *hapus;
    Node *bantu;
    if (isEmpty() == 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 << "List kosong!" << endl;
    }
}
// Hapus Tengah
void hapusTengah(int posisi)
{
    Node *hapus, *bantu, *bantu2;
    if (posisi < 1 || posisi > hitungList())
    {
        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;
}

// Ubah Depan
void ubahDepan(int data, string kata)
{
    if (isEmpty() == false)
    {
        head->data = data;
        head->kata = kata;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah Tengah
void ubahTengah(int data, string kata, int posisi)
{
    Node *bantu;
    if (isEmpty() == false)
    {
        if (posisi < 1 || posisi > hitungList())
        {
            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->data = data;
            bantu->kata = kata;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah Belakang
void ubahBelakang(int data, string kata)
{
    if (isEmpty() == false)
    {
        tail->data = data;
        tail->kata = kata;
    }
}

```



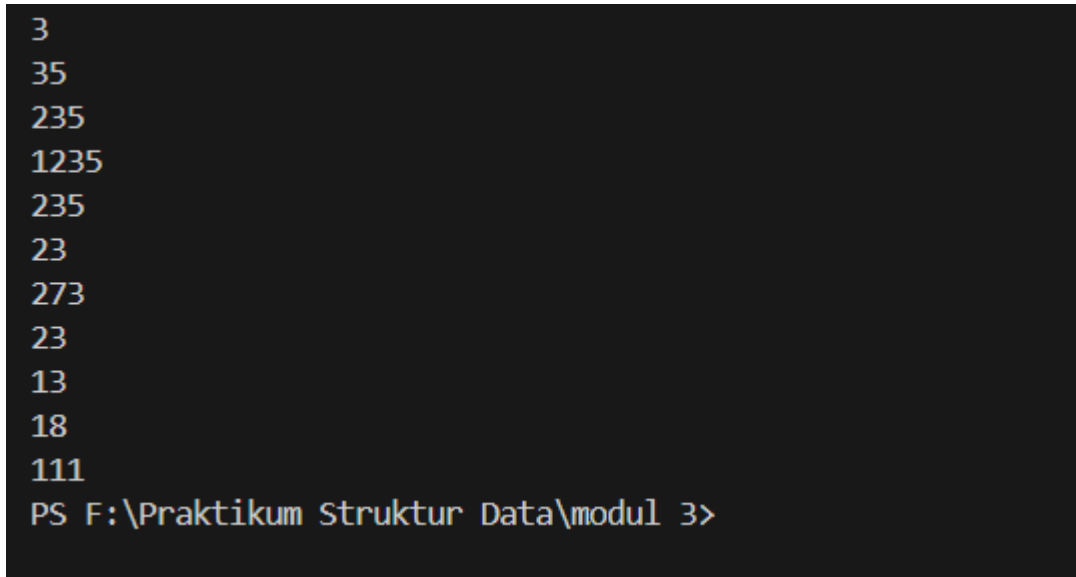
```

    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
// Hapus List
void clearList()
{
    Node *bantu, *hapus;
    bantu = head;
    while (bantu != NULL)
    {
        hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}
// Tampilkan List
void tampil()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << "\t";
            cout << bantu->kata << "\t";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
int main()
{
    init();
    insertDepan(3, "satu");
    tampil();
    insertBelakang(5, "dua");
    tampil();
    insertDepan(2, "tiga");
    tampil();
    insertDepan(1, "empat");
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, "lima", 2);
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1, "enam");
    tampil();
    ubahBelakang(8, "tujuh");
}

```

```
tampil();  
ubahTengah(11, "delapan", 2);  
tampil();  
  
return 0;  
}
```

Screenshots program



```
3  
35  
235  
1235  
235  
23  
273  
23  
13  
18  
111  
PS F:\Praktikum Struktur Data\modul 3>
```

Deskripsi program

Program ini merupakan implementasi dari linked list non-circular dalam bahasa C++. Linked list adalah struktur data yang terdiri dari simpul-simpul yang saling terhubung, di mana setiap simpul memiliki data dan referensi ke simpul berikutnya. Program ini memanfaatkan struktur data linked list untuk mengelola data secara dinamis.

Program dimulai dengan deklarasi struct Node yang merupakan simpul dalam linked list. Setiap simpul memiliki dua anggota, yaitu data yang merupakan

Guided 2 (Latihan Double Linked List)

```
#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: {

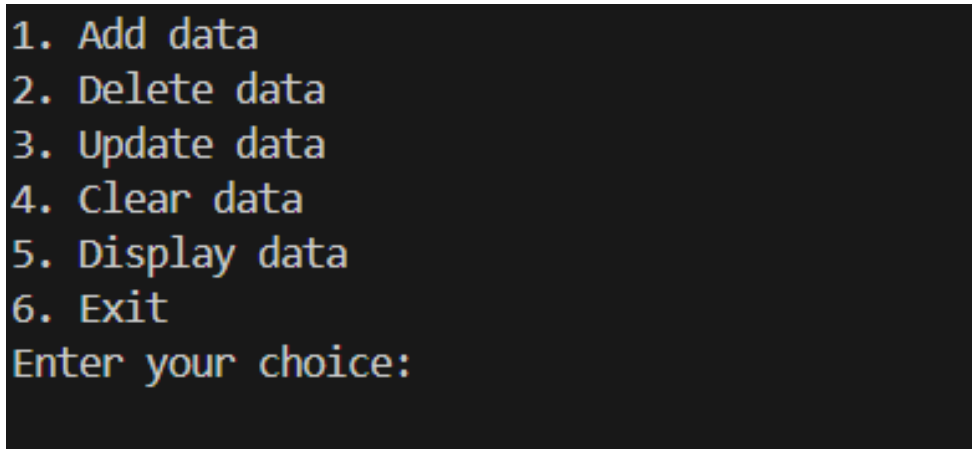
```

```

        return 0;
    }
    default: {
        cout << "Invalid choice" << endl;
        break;
    }
}
return 0;
}

```

Screenshots program



Deskripsi program

Program ini adalah implementasi dari Doubly Linked List dalam C++. Doubly Linked List adalah struktur data linear di mana setiap simpul memiliki dua pointer, satu menunjuk ke simpul sebelumnya (prev) dan satu menunjuk ke simpul berikutnya (next).

Kode program terdiri dari dua kelas utama: Node dan DoublyLinkedList. Kelas Node merepresentasikan simpul dalam Doubly Linked List dengan tiga anggota: data untuk menyimpan nilai, prev sebagai pointer ke simpul sebelumnya, dan next sebagai pointer ke simpul berikutnya. Kelas DoublyLinkedList

menyediakan operasi-operasi dasar seperti menambah, menghapus, mengupdate, dan menampilkan data dalam Doubly Linked List.

Fungsi *main()* memberikan pilihan kepada pengguna untuk melakukan operasi-operasi tersebut. Pengguna dapat memasukkan data baru, menghapus data, mengupdate data, menghapus semua data, menampilkan data, atau keluar dari program.

Program akan terus berjalan hingga pengguna memilih untuk keluar dengan memilih opsi "6. Exit". Ini memungkinkan pengguna untuk dengan mudah mengelola data dalam Double Linked List

LATIHAN KELAS - UNGUIDED

Unguided 1

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

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

class LinkedList {
private:
    Node* head;

public:
    LinkedList() {
        head = nullptr;
    }

    void insertAwal(string nama, int usia) {
        Node* newNode = new Node;
        newNode->nama = nama;
        newNode->usia = usia;
        newNode->next = head;
        head = newNode;
    }

    void insertAkhir(string nama, int usia) {
        Node* newNode = new Node;
        newNode->nama = nama;
        newNode->usia = usia;
        newNode->next = nullptr;

        if (head == nullptr) {
            head = newNode;
            return;
        }
    }
}
```

```

        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
    }

void insertSetelah(string nama, int usia, string namaSebelum) {
    Node* newNode = new Node;
    newNode->nama = nama;
    newNode->usia = usia;

    Node* temp = head;
    while (temp != nullptr && temp->nama != namaSebelum) {
        temp = temp->next;
    }

    if (temp == nullptr) {
        cout << "Node dengan nama " << namaSebelum << " tidak ditemukan." <<
endl;
        return;
    }

    newNode->next = temp->next;
    temp->next = newNode;
}

void hapus(string nama) {
    if (head == nullptr) {
        cout << "Linked list kosong." << endl;
        return;
    }

    if (head->nama == nama) {
        Node* temp = head;
        head = head->next;
        delete temp;
        return;
    }

```

```

    }

    Node* prev = head;
    Node* temp = head->next;
    while (temp != nullptr && temp->nama != nama) {
        prev = temp;
        temp = temp->next;
    }

    if (temp == nullptr) {
        cout << "Node dengan nama " << nama << " tidak ditemukan." << endl;
        return;
    }

    prev->next = temp->next;
    delete temp;
}

void ubah(string nama, string namaBaru, int usiaBaru) {
    Node* temp = head;
    while (temp != nullptr && temp->nama != nama) {
        temp = temp->next;
    }

    if (temp == nullptr) {
        cout << "Node dengan nama " << nama << " tidak ditemukan." << endl;
        return;
    }

    temp->nama = namaBaru;
    temp->usia = usiaBaru;
}

void tampilkan() {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->nama << " " << temp->usia << endl;
        temp = temp->next;
    }
}

```



```

    }
}
};

int main() {
    LinkedList myList;

    myList.insertAwal("Reza Alvonzo", 19);
    myList.insertAwal("John", 19);
    myList.insertAwal("Jane", 20);
    myList.insertAwal("Michael", 18);
    myList.insertAwal("Yusuke", 19);
    myList.insertAwal("Akechi", 20);
    myList.insertAwal("Hoshino", 18);
    myList.insertAwal("Karin", 18);

    cout << "Data setelah langkah (a):" << endl;
    myList.tampilkan();
    cout << endl;

    myList.hapus("Akechi");
    myList.insertSetelah("Futaba", 18, "John");
    myList.insertAwal("Igor", 20);
    myList.ubah("Michael", "Reyn", 18);
    cout << "Data setelah dilakukan semua operasi:" << endl;
    myList.tampilkan();

    return 0;
}

```

Screenshots program

```
Data setelah langkah (a):
Karin 18
Hoshino 18
Akechi 20
Yusuke 19
Michael 18
Jane 20
John 19
Reza Alvonzo 19

Data setelah dilakukan semua operasi:
Igor 20
Karin 18
Hoshino 18
Yusuke 19
Reyn 18
Jane 20
John 19
Futaba 18
Reza Alvonzo 19

Process returned 0 (0x0)   execution time : 0.057 s
Press any key to continue.
|
```

Deskripsi program

Program di atas merupakan implementasi dari linked list berganda (doublylinked list) yang menyimpan data berupa nama dan usia seseorang. Setiap node dalam linked list terdiri dari dua bagian data, yaitu nama dan usia, serta pointer yang menunjukkan ke node selanjutnya.

Unguided 2

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

class Node {
public:
    string namaProduk;
    int harga;
    Node* prev;
    Node* next;
};

class DoublyLinkedList {
public:
    Node* head;
    Node* tail;
    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    void push(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;
    }

    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(string oldNamaProduk, string newNamaProduk, int newHarga) {
        Node* current = head;
        while (current != nullptr) {
            if (current->namaProduk == oldNamaProduk) {
                current->namaProduk = newNamaProduk;
                current->harga = newHarga;
                return true;
            }
            current = current->next;
        }
        return false;
    }
};
```

```

}

void insertAfter(string namaProduk, int harga, string afterNamaProduk) {
    Node* newNode = new Node;
    newNode->namaProduk = namaProduk;
    newNode->harga = harga;
    Node* current = head;
    while (current != nullptr) {
        if (current->namaProduk == afterNamaProduk) {
            newNode->prev = current;
            newNode->next = current->next;
            if (current->next != nullptr) {
                current->next->prev = newNode;
            } else {
                tail = newNode;
            }
            current->next = newNode;
            return;
        }
        current = current->next;
    }
}

void deleteAfter(string afterNamaProduk) {
    Node* current = head;
    while (current != nullptr) {
        if (current->namaProduk == afterNamaProduk) {
            Node* temp = current->next;
            if (temp != nullptr) {
                current->next = temp->next;
                if (temp->next != nullptr) {
                    temp->next->prev = current;
                } else {
                    tail = current;
                }
            }
            delete temp;
        }
        return;
    }
    current = current->next;
}

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;
    cout << "Nama Produk\tHarga" << endl;
    while (current != nullptr) {
        cout << current->namaProduk << "\t\t" << current->harga << endl;
        current = current->next;
    }
    cout << endl;
}

```

```

    }
};

int main() {
    DoublyLinkedList list;
    while (true) {
        cout << "Toko Skincare Purwokerto" << 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 choice;
        cout << "Masukkan pilihan Anda: ";
        cin >> choice;
        switch (choice) {
            case 1: {
                string namaProduk;
                int harga;
                cout << "Masukkan nama produk: ";
                cin >> namaProduk;
                cout << "Masukkan harga: ";
                cin >> harga;
                list.push(namaProduk, harga);
                break;
            }
            case 2: {
                list.pop();
                break;
            }
            case 3: {
                string oldNamaProduk, newNamaProduk;
                int newHarga;
                cout << "Masukkan nama produk yang lama: ";
                cin >> oldNamaProduk;
                cout << "Masukkan nama produk yang baru: ";
                cin >> newNamaProduk;
                cout << "Masukkan harga yang baru: ";
                cin >> newHarga;
                bool updated = list.update(oldNamaProduk, newNamaProduk,
newHarga);

                if (!updated) {
                    cout << "Data tidak ditemukan" << endl;
                }
                break;
            }
            case 4: {
                string namaProduk, afterNamaProduk;
                int harga;
                cout << "Masukkan nama produk: ";
                cin >> namaProduk;
                cout << "Masukkan harga: ";
                cin >> harga;
                cout << "Masukkan nama produk setelahnya: ";
                cin >> afterNamaProduk;
                list.insertAfter(namaProduk, harga, afterNamaProduk);
                break;
            }
            case 5: {

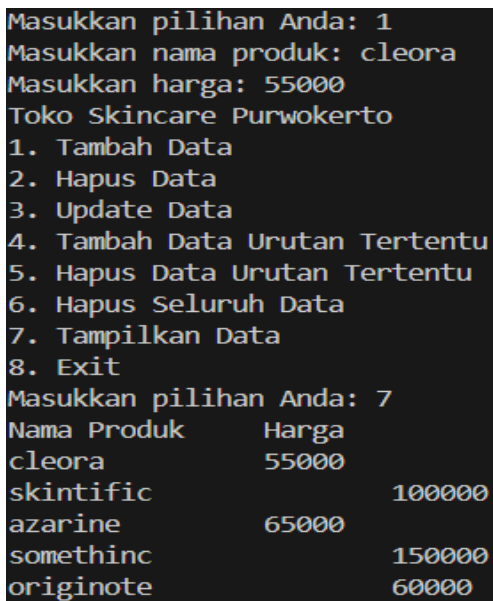
```

```

        string afterNamaProduk;
        cout << "Masukkan nama produk setelahnya: ";
        cin >> afterNamaProduk;
        list.deleteAfter(afterNamaProduk);
        break;
    }
    case 6: {
        list.deleteAll();
        break;
    }
    case 7: {
        list.display();
        break;
    }
    case 8: {
        return 0;
    }
    default: {
        cout << "Pilihan tidak valid" << endl;
        break;
    }
}
}
return 0;
}

```

Screenshots program



```

Masukkan pilihan Anda: 1
Masukkan nama produk: cleora
Masukkan harga: 55000
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
Masukkan pilihan Anda: 7
Nama Produk      Harga
cleora           55000
skintific        100000
azarine          65000
somethinc        150000
originote        60000

```

Deskripsi program

Program di atas merupakan implementasi dari sebuah aplikasi manajemen dataproduk toko skincare. Data produk disimpan dalam linked list berganda (doubly linked list), yang setiap nodenya terdiri dari dua bagian data, yaitu nama produk dan harganya.

BAB IV

KESIMPULAN

- Program 1: Program ini menggunakan struktur data Single Linked List untuk menyimpan dan mengelola informasi siswa. Fungsi-fungsi utamanya memungkinkan penambahan siswa baru, penghapusan siswa berdasarkan nama, dan penampilan daftar siswa lengkap. Program ini dirancang untuk keperluan administratif di lingkungan pendidikan, di mana data siswa perlu dikelola secara dinamis.
- Program 2: Program ini menggunakan struktur data Double Linked List untuk menyimpan dan mengelola informasi produk dalam konteks toko skincare. Program ini dilengkapi dengan menu interaktif yang memungkinkan pengguna untuk menambahkan produk baru, menghapus produk, memperbarui informasi produk, menambahkan produk ke urutan tertentu, dan menampilkan daftar produk. Program ini dirancang untuk memenuhi kebutuhan manajemen inventaris dan menyediakan antarmuka yang ramah pengguna untuk operasi yang berkaitan dengan produk.

Kedua program tersebut menunjukkan penerapan struktur data linked list dalam konteks nyata, dengan fokus pada kemudahan penggunaan dan fleksibilitas dalam manipulasi data.
