LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA

MODUL III
SINGLE DAN DOUBLE
LINKED LIST

Disusun Oleh :
NAMA : Fadhel Yussie Ramadhan
NIM : 2311102322

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

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

A. Dasar Teori

Materi yang di bahas adalah tentang Single dan double Linked List

B. 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.Guided (berisi screenshot source code & output program disertai penjelasannya)

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.

Guided 1

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

// Deklarasi Struct Node
struct Node {
    int data;
    Node* next;
};

Node* head;
Node* tail;

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

// Pengecekan apakah list kosong
bool isEmpty() {
    return head == NULL;
}

// Tambah Node di depan
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;
    }
}

// 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;
    }
}
```

```
// Hitung jumlah Node di list
int hitungList() {
    Node* hitung = head;
    int jumlah = 0;
    while (hitung != NULL) {
        jumlah++;
        hitung = hitung->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();
        baru->data = data;
        Node* bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1) {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

// Hapus Node di depan
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;
    }
}

// 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;
    }
}

// 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;
        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
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;
    }
}

// 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;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}

// 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;
}
```

```
                cout << "Error! Tidak dapat melakukan pembagian
dengan nol.";             break;         default:
            cout << "Error! Operator tidak benar.";
    }
return 0;
}
```

Screenshots Output

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

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 C:\Users\USER\GitHub\Struktur-Data-2\Modul 3\Guided>
```

Deskripsi:

Program single linked list

E. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1

```cpp
#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

```
g++ guided2.cpp -o guided2 } ; if ($?) { .\guided2 }
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: 99
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 3
Enter old data: 99
Enter new data: 98
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
98
```

Deskripsi:

Program double linked list

F. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Unguided 1

```cpp
#include <iostream>
using namespace std;
//2311102322_fadhel Yussie Ramadhan
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# Linked List Menu #" << 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

```
john 19 , jane 20 , michael 18 , yusuke 19 , akechi 20 , hoshino 18 , karin 18 ,

 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 : 5

john 19 , jane 20 , michael 18 , yusuke 19 , hoshino 18 , karin 18 ,

 1. Insert Depan
 2. Insert Belakang
 3. Insert Tengah
 4. Hapus Depan
 5. Hapus Belakang
 6. Hapus Tengah
```

```
  Pilihan : 3
 Masukkan Posisi : 2
 Masukkan Nama : futaba
 Masukkan Usia : 18

 john 19 , futaba 18 , jane 20 , michael 18 , yusuke 19 , hoshino 18 , karin 18 ,

  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 : igor
 Masukkan Usia : 20
```

```
igor 20 , john 19 , futaba 18 , jane 20 , michael 18 , yusuke 19 , hoshino 18 , karin 18 ,

 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 : 5
Masukkan Nama : reyn
Masukkan Usia : 18

igor 20 , john 19 , futaba 18 , jane 20 , reyn 18 , yusuke 19 , hoshino 18 , karin 18 ,
```

```
 1. Insert Depan
 2. Insert Belakang
 3. Insert Tengah
 4. Hapus Depan
 5. Hapus Belakang
 6. Hapus Tengah
 7. Ubah Depan
 8. Ubah Belakang
 0. Keluar Program
 Pilihan : 10
igor 20 , john 19 , futaba 18 , jane 20 , reyn 18 , yusuke 19 , hoshino 18 , karin 18 ,
```

Deskripsi:

Single linked list non circular untuk menyimpan nama mahasiswa dan input dari user.

G. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Unguided 2

```
using #include <iostream>
#include <iomanip>
using namespace std;
//2311102322_Fadhel Yussie Ramadhan
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 pushCenter(string namaProduk, int harga, int posisi)
```

```
{
    if (posisi < 0)
    {
        cout << "Posisi harus bernilai non-negatif." << endl;
        return;
    }

    Node *newNode = new Node;
    newNode->namaProduk = namaProduk;
    newNode->harga = harga;

    if (posisi == 0 || head == nullptr)
    {
        newNode->prev = nullptr;
        newNode->next = head;

        if (head != nullptr)
        {
            head->prev = newNode;
        }
        else
        {
            tail = newNode;
        }
        head = newNode;
    }
    else
    {
        Node *temp = head;
        int count = 0;
        while (temp != nullptr && count < posisi)
        {
            temp = temp->next;
            count++;
        }

        if (temp == nullptr)
        {
            newNode->prev = tail;
            newNode->next = nullptr;
            tail->next = newNode;
            tail = newNode;
        }
        else
        {
            newNode->prev = temp->prev;
            newNode->next = temp;
            temp->prev->next = newNode;
            temp->prev = newNode;
```

```
        }
    }
}

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

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

    delete temp;
}

void popCenter(int posisi)
{
    if (head == nullptr)
    {
        cout << "List kosong. Tidak ada yang bisa dihapus." << endl;
        return;
    }

    if (posisi < 0)
    {
        cout << "Posisi harus bernilai non-negatif." << endl;
        return;
    }

    if (posisi == 0)
    {
        Node *temp = head;
        head = head->next;

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

```
            {
                tail = nullptr;
            }

            delete temp;
        }
        else
        {
            Node *temp = head;
            int count = 0;
            while (temp != nullptr && count < posisi)
            {
                temp = temp->next;
                count++;
            }

            if (temp == nullptr)
            {
                cout << "Posisi melebihi ukuran list. Tidak ada yang dihapus." << endl;
                return;
            }

            if (temp == tail)
            {
                tail = tail->prev;
                tail->next = nullptr;
                delete temp;
            }
            else
            {
                temp->prev->next = temp->next;
                temp->next->prev = temp->prev;
                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;
            }
```

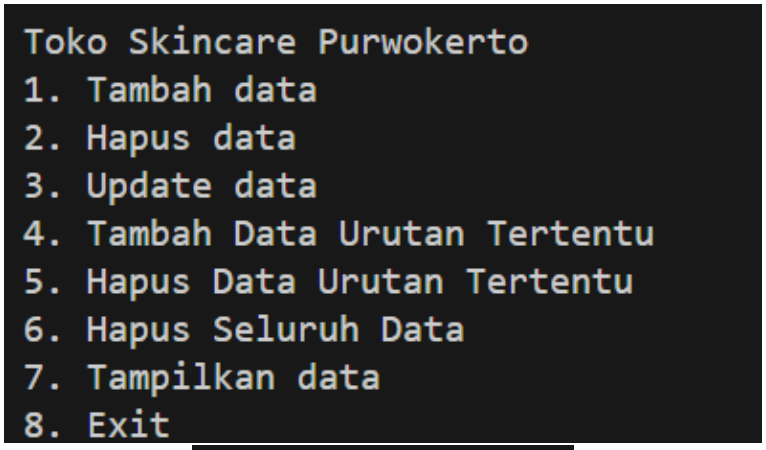
```
            current = current->next;
        }
        return false;
    }

    bool updateCenter(string newNamaProduk, int newHarga, int posisi)
    {
        if (head == nullptr)
        {
            cout << "List kosong. Tidak ada yang dapat diperbarui." << endl;
            return false;
        }

        if (posisi < 0)
        {
            cout << "Posisi harus bernilai non-negatif." << endl;
            return false;
        }

        Node *current = head;
        int count = 0;

        while (current != nullptr && count < posisi)
        {
            current = current->next;
            count++;
        }

        if (current == nullptr)
        {
            cout << "Posisi melebihi ukuran list. Tidak ada yang diperbarui." << endl;
            return false;
        }

        current->namaProduk = newNamaProduk;
        current->harga = newHarga;
        return true;
    }

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

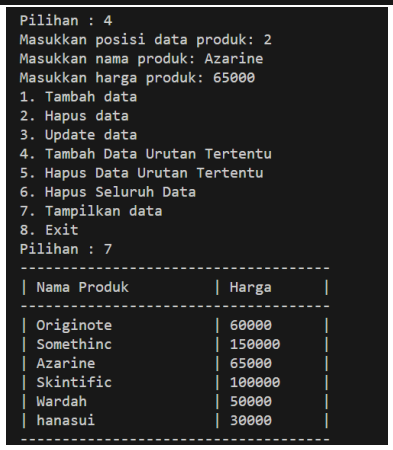
```
        tail = nullptr;
    }

    void display()
    {
        if (head == nullptr)
        {
            cout << "List kosong." << endl;
            return;
        }

        Node *current = head;

        cout << setw(37) << setfill('-') << "-" << setfill(' ') << endl;
        cout << "| " << setw(20) << left << "Nama Produk"
             << " | " << setw(10) << "Harga"
             << " |" << endl;
        cout << setw(37) << setfill('-') << "-" << setfill(' ') << endl;

        while (current != nullptr)
        {
            cout << "| " << setw(20) << left << current->namaProduk << " | " << setw(10
current->harga << " |" << endl;
            current = current->next;
        }
        cout << setw(37) << setfill('-') << "-" << setfill(' ') << endl;
    }
};

int main()
{
    DoublyLinkedList list;
    int choice;
    cout << endl
         << "Banyumas Shoes Market" << endl;
    do
    {
        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;

        cout << "Pilihan : ";
        cin >> choice;
```

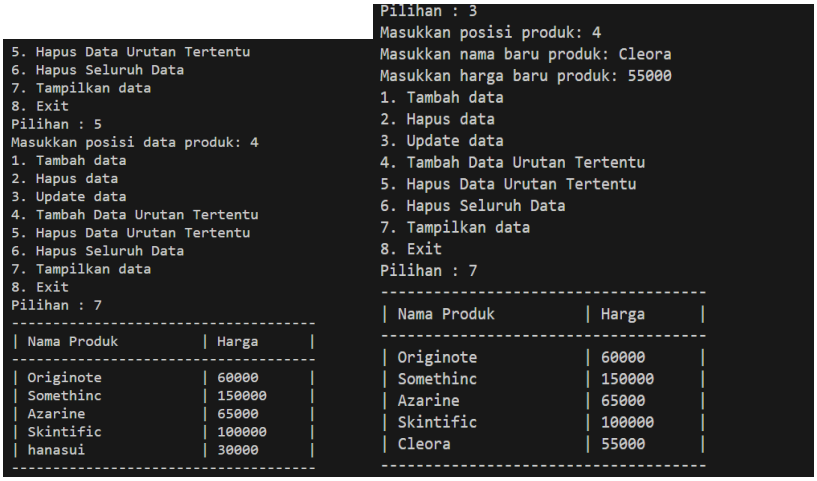
```
        switch (choice)
        {
        case 1:
        {
            string namaProduk;
            int harga;
            cout << "Masukkan nama produk: ";
            cin.ignore();
            getline(cin, namaProduk);
            cout << "Masukkan harga produk: ";
            cin >> harga;
            list.push(namaProduk, harga);
            break;
        }
        case 2:
        {
            list.pop();
            break;
        }
        case 3:
        {
            string newNamaProduk;
            int newHarga, posisi;
            cout << "Masukkan posisi produk: ";
            cin >> posisi;
            cout << "Masukkan nama baru produk: ";
            cin >> newNamaProduk;
            cout << "Masukkan harga baru produk: ";
            cin >> newHarga;
            bool updatedCenter = list.updateCenter(newNamaProduk, newHarga, posisi);
            if (!updatedCenter)
            {
                cout << "Data not found" << endl;
            }
            break;
        }
        case 4:
        {
            string namaProduk;
            int harga, posisi;
            cout << "Masukkan posisi data produk: ";
            cin >> posisi;
            cout << "Masukkan nama produk: ";
            cin.ignore();
            getline(cin, namaProduk);
            cout << "Masukkan harga produk: ";
            cin >> harga;
            list.pushCenter(namaProduk, harga, posisi);
            break;
```

```
        }
        case 5:
        {
            int posisi;
            cout << "Masukkan posisi data produk: ";
            cin >> posisi;
            list.popCenter(posisi);
            break;
        }
        case 6:
        {
            list.deleteAll();
            break;
        }
        case 7:
        {
            list.display();
            break;
        }
        case 8:
        {
            return 0;
        }
        default:
        {
            cout << "Invalid choice" << endl;
            break;
        }
        }
    } while (choice != 8);

    return 0;
}
```

Screenshots Output







Deskripsi:

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.
User dapat menginputkan data sesuai menu kemudian dapat mengubahnya sesuai menu juga.

H. Kesimpulan

Kesimpulannya adalah laprak 3 ini mempelajari tentang single dan double linked list dan bagaimana linked list bekerja dalam kode yang di buat. Fungsinya untuk menyimpan beberapa data, menghapus data, update data, tambah/hapus data di urutan tertentu, hapus seluruh data dan menampilkan seluruh data.

Referensi:

Modul 3 tipe data praktikum algoritma struktur data IF 11 B dari whatsapp group “Praktikum strukdat IF 11 B”