LAPORAN PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA
”SINGLE AND DOUBLE LINKED LIST”
MODUL 3

Disusun oleh :

Nama : Yasvin Syahgana

Nim : 2311102065

Dosen

Wahyu Andi Syahputra, S.pd.,M.Eng

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

MODUL 2
ARRAY

A. TUJUAN PRAKTIKUM

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

B. DASAR TEORI

1. Single Linked List

   Single Linked List adalah struktur data yang terdiri dari sekumpulan node di mana setiap node memiliki dua bagian: data dan referensi ke node berikutnya dalam urutan. Struktur ini memungkinkan penyisipan dan penghapusan elemen dengan efisien tanpa perlu memindahkan elemen lain.

   Properti

   - Head: Referensi ke node pertama dalam list.
   - Tail: Referensi ke node terakhir dalam list. Tail tidak wajib, tapi dapat mempermudah beberapa operasi.
   - Next: Setiap node memiliki referensi ke node berikutnya, kecuali node terakhir yang menunjuk ke `null`.

   Operasi Utama:

   - Penyisipan (Insertion): Menambahkan node baru.
   - Penghapusan (Deletion): Menghapus node yang ada.
   - Traversal: Mengunjungi setiap node untuk memproses atau mencari data.

2. Double Linked List

   Double Linked List mirip dengan Single Linked List, tetapi setiap node memiliki referensi tambahan ke node sebelumnya, yang memungkinkan navigasi dua arah.

Properti:

- Previous: Setiap node memiliki referensi ke node sebelumnya, memungkinkan iterasi mundur.

Operasi Utama:
Operasi utama sama dengan Single Linked List, tetapi dengan kemampuan navigasi ke belakang.
Keuntungan Double Linked List:
Iterasi Mundur: Kemampuan untuk bergerak mundur melalui list.
Penghapusan Efisien: Penghapusan node lebih efisien karena kita memiliki referensi ke node sebelumnya.

Kekurangan:
Memori Tambahan: Membutuhkan memori lebih karena adanya referensi tambahan.
Kompleksitas: Implementasi lebih kompleks dibandingkan dengan Single Linked List.
Contoh implementasi :

```plaintext
class Node {
   data
   next // Untuk Single Linked List
   prev // Hanya untuk Double Linked List
}

class LinkedList {
   head
   tail // Opsional, tergantung pada kebutuhan

   // Metode untuk operasi penyisipan, penghapusan, dll.
}
```

C. GUIDED

1. LATIHAN SINGLE LINKED LIST

```
// YASVIN SYAHGANA
// 2311102065

#include <iostream> using
namespace std;

//Deklarasi Struct Nodestruct Node{
        //komponen/memberint
        data;
        Node *next;
};
        Node *head;
        Node *tail;
//Inisialisasi Nodevoid init(){
        head = NULL;tail =
        NULL;
}
// Pengecekan bool
isEmpty(){
        if (head == NULL)return
        true;
        else
        return false;
}
//Tambah Depan
void insertDepan(int nilai){
        //Buat Node baru
        Node *baru = new Node;baru-
        >data = nilai; baru->next = NULL;
        if (isEmpty() == true){head = tail =
                baru;
                tail->next = NULL;
```

```
 }
 else{
    baru->next = head;head = baru;
    }
    }
    //Tambah Belakang
    void insertBelakang(int nilai){
    //Buat Node baru
    Node *baru = new Node;baru->data = nilai;
    baru->next = NULL;
if (isEmpty() == true){head = tail = baru;tail-
    >next = NULL;
    }
    else{
    tail->next = baru;tail = baru;
    }
    }
    //Hitung Jumlah Listint hitungList(){
    Node *hitung; hitung = head; int jumlah = 0;
while( hitung != NULL ){jumlah++;
    hitung = hitung->next;
    }
    return jumlah;
    }
    //Tambah 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" <<
  end
   l;
    }
 else{
    Node *baru, *bantu;
    baru = new Node();
    baru->data = data;
// tranversing bantu =
    head; int nomor =
    1;
    while( nomor < posisi - 1 ){bantu = bantu-
        >next; nomor++;
    }
    baru->next = bantu->next;bantu-
    >next = baru;
    }
    }
//Hapus Depan
d 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){
       if (isEmpty() == false){head->data = data;
  }
  else{
  cout << "List masih kosong!" << endl;
  }
  }
  //Ubah Tengah
  void ubahTengah(int data, 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;
    }
    }
    else{
    cout << "List masih kosong!" << endl;
    }
    }
    //Ubah Belakang
void ubahBelakang(int data){if (isEmpty() ==
    false){
    tail->data = data;
    }
    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 Listvoid tampil(){
  Node *bantu; bantu = head;
```

```
while (bantu != NULL)
{
cout << bantu->data
<< endl;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;
```

Output

```
 if ($?) { g++ guided1.cpp -o guided1 } ; if ($?) { .\guided1 }
3
35
235
1235
235
23
273
23
13
18
111
```

Deskripsi Program

Program ini adalah implementasi **linked list satu arah (single linked list) yang tidak bersifat sirkular**. Dalam program ini, kita dapat melakukan beberapa operasi seperti menambahkan node di depan atau belakang, menghitung jumlah node dalam linked list, serta menambahkan node di tengah. Selain itu, kita juga dapat menghapus node di depan atau belakang.

2. LATIHAN DOUBLE LINKED LIST

Source Code

```
// YASVIN SYAHGANA
// 2311102065
#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;
}
```

Output

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

Deskripsi Porgram

Program ini adalah implementasi **doubly linked list** (linked list dengan dua arah) dalam bahasa C++. Dalam program ini, kita dapat menambahkan dan menghapus data, serta mengganti nilai data. Semua operasi ini berlaku pada linked list yang terdiri dari node-node dengan tautan ke node sebelumnya dan berikutnya.

D. Unguided

1. *Soal mengenai Single Linked List*

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

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

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

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

2. *Soal mengenai Double Linked List*

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

Nama Produk	Harga
Originote	60.000
Somethinc	150.000

Skintific	100.000
Wardah	50.000
Hanasui	30.000

Case:

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

   Toko Skincare Purwokerto

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

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

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

UNGUIDED 1

```
// YASVIN SYAHGANA
// 2311102065
// UNGUIDED 1
#include <iostream>
#include <string>

using namespace std;

class Node {
public:
    string nama;
    int umur;
    Node* next;

    Node(string nama, int umur) : nama(nama),
umur(umur), next(nullptr) {}
};

class SingleLinkedList {
public:
    Node* head;

    SingleLinkedList() : head(nullptr) {}

    void tambahSiswaDiPosisi(const string& nama,
int umur, int posisi) {
        Node* newNode = new Node(nama, umur);
        if (posisi == 1) {
            newNode->next = head;
            head = newNode;
        } else {
            Node* current = head;
            Node* prev = nullptr;
            int currPosisi = 1;

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

            if (prev == nullptr) {
                head = newNode;
            } else {
                prev->next = newNode;
            }
            newNode->next = current;
```

```
}
    }

    bool hapusSiswa(const string& nama) {
        Node* current = head;
        Node* prev = nullptr;

        while (current != nullptr && current->nama
!= nama) {
            prev = current;
            current = current->next;
        }

        if (current == nullptr) {
            return false;
        }

        if (prev == nullptr) {
            head = current->next;
        } else {
            prev->next = current->next;
        }

        delete current;
        return true;
    }

    void tampilkanList() {
        Node* current = head;
        while (current != nullptr) {
            cout << "Nama: " << current->nama << ",
Umur: " << current->umur << endl;
            current = current->next;
        }
    }
};
void menu() {
    SingleLinkedList listSiswa;
    int pilihan, umur, posisi;
    string nama;

    while (true) {
        cout << "Menu:\n";
        cout << "1. Tambah Siswa\n";
        cout << "2. Hapus Siswa\n";
        cout << "3. Tampilkan List Siswa\n";
        cout << "4. Keluar\n";
        cout << "Masukkan pilihan Anda: ";
        cin >> pilihan;
```

```
        switch (pilihan) {
            case 1:
                cout << "Masukkan nama siswa: ";
                cin >> nama;
                cout << "Masukkan umur siswa: ";
                cin >> umur;
                cout << "Masukkan posisi untuk
menambahkan siswa: ";
                cin >> posisi;
                listSiswa.tambahSiswaDiPosisi(nama,
umur, posisi);
                break;
            case 2:
                cout << "Masukkan nama siswa yang
akan dihapus: ";
                cin >> nama;
                if (listSiswa.hapusSiswa(nama)) {
                    cout << nama << " berhasil
dihapus dari list.\n";
                } else {
                    cout << nama << " tidak
ditemukan dalam list.\n";
                }
                break;
            case 3:
                listSiswa.tampilkanList();
                break;
            case 4:
                cout << "Terima kasih telah
menggunakan program ini.\n";
                return;
            default:
                cout << "Pilihan tidak valid.
Silakan coba lagi.\n";
        }
    }
}

int main() {
    menu();
    return 0;
}
```

Output

```
Menu:
1. Tambah Siswa
2. Hapus Siswa
3. Tampilkan List Siswa
4. Keluar
Masukkan pilihan Anda: 3
Nama: john, Umur: 19
Nama: jane, Umur: 20
Nama: michael, Umur: 18
Nama: yusuke, Umur: 19
Nama: akechi, Umur: 20
Nama: hoshino, Umur: 18
Nama: karim, Umur: 18
Menu:
1. Tambah Siswa
2. Hapus Siswa
3. Tampilkan List Siswa
4. Keluar
Masukkan pilihan Anda: 2
Masukkan nama siswa yang akan dihapus: akechi
akechi berhasil dihapus dari list.
```

```
Masukkan pilihan Anda: 3
Nama: john, Umur: 19
Nama: futaba, Umur: 18
Nama: jane, Umur: 20
Nama: reyn, Umur: 18
Nama: yusuke, Umur: 19
Nama: hoshino, Umur: 18
Nama: karim, Umur: 18
Menu:
1. Tambah Siswa
2. Hapus Siswa
3. Tampilkan List Siswa
4. Keluar
Masukkan pilihan Anda:
```

Deskripsi Program

implementasi Single Linked List dalam bahasa C++ yang memungkinkan pengguna untuk melakukan operasi dasar seperti menambahkan siswa, menghapus siswa berdasarkan nama, dan menampilkan seluruh list siswa. Program ini dirancang untuk menyimpan informasi siswa yang terdiri dari nama dan umur.

Fungsi-fungsi utama dalam program tersebut:

1. tambahSiswa: Fungsi ini menambahkan siswa baru ke akhir list. Jika list kosong, siswa yang ditambahkan akan menjadi elemen pertama dalam list.

2. hapusSiswa: Fungsi ini menghapus siswa dari list berdasarkan nama. Jika siswa dengan nama yang diberikan ditemukan, ia akan dihapus dari list. tampilkanList: Fungsi ini menampilkan semua siswa yang ada dalam list beserta umur mereka. Program ini juga menyediakan menu interaktif yang memungkinkan pengguna untuk memilih operasi yang ingin dilakukan pada Single Linked List. Pengguna dapat menambahkan siswa baru, menghapus siswa, dan menampilkan seluruh list siswa melalui pilihan menu yang diberikan. Program ini memberikan kontrol yang lebih besar kepada pengguna untuk berinteraksi **dengan list dan melakukan operasi sesuai kebutuhan mereka.**

UNGUIDED 2

```
// YASVIN SYAHGANA
// 2311102065
#include <iostream>
#include <string>
#include <iomanip>

using namespace std;

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

    Node(string namaProduk, int harga) :
namaProduk(namaProduk), harga(harga), prev(nullptr),
next(nullptr) {}
};

class DoubleLinkedList {
private:
    Node* head;
    Node* tail;

public:
    DoubleLinkedList() : head(nullptr), tail(nullptr) {}

    void tambahData(string namaProduk, int harga) {
        Node* newNode = new Node(namaProduk, harga);
        if (!head) {
            head = tail = newNode;
        } else {
            tail->next = newNode;
            newNode->prev = tail;
            tail = newNode;
        }
    }

    void tambahDataDiUrutan(string namaProduk, int harga,
int urutan) {
        Node* newNode = new Node(namaProduk, harga);
        if (urutan == 1) {
            if (head) {
                head->prev = newNode;
            }
            newNode->next = head;
            head = newNode;
            if (!tail) {
                tail = newNode;
            }
        } else {
           Node* temp = head;
           int pos = 1;
```

```
while (temp && pos < urutan) {
                temp = temp->next;
                pos++;
            }
            if (temp) {
                newNode->next = temp;
                newNode->prev = temp->prev;
                if (temp->prev) {
                    temp->prev->next = newNode;
                }
                temp->prev = newNode;
            } else {
                tail->next = newNode;
                newNode->prev = tail;
                tail = newNode;
            }
        }
    }

    void hapusData(string namaProduk) {
        Node* temp = head;
        while (temp) {
            if (temp->namaProduk == namaProduk) {
                if (temp->prev) {
                    temp->prev->next = temp->next;
                } else {
                    head = temp->next;
                }
                if (temp->next) {
                    temp->next->prev = temp->prev;
                } else {
                    tail = temp->prev;
                }
                delete temp;
                return;
            }
            temp = temp->next;
        }
    }

    void updateData(string namaLama, string namaBaru, int
hargaBaru) {
        Node* temp = head;
        while (temp) {
            if (temp->namaProduk == namaLama) {
                temp->namaProduk = namaBaru;
                temp->harga = hargaBaru;
                return;
            }
            temp = temp->next;
        }
    }
```

```
void tampilkanData() {
        Node* temp = head;
        cout << "Toko Skincare Purwokerto\n";
        cout << left << setw(15) << "Nama Produk"
<< "Harga\n";
        while (temp) {
            cout << left << setw(15) << temp-
>namaProduk << temp->harga << endl;
            temp = temp->next;
        }
    }

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

    void clearList() {
        while (!isEmpty()) {
            hapusData(head->namaProduk);
        }
    }
};

void menu(DoubleLinkedList& dll) {
    int pilihan, harga, urutan;
    string namaProduk, produkSebelum, namaLama,
namaBaru;

    do {
        cout << "\nToko Skincare Purwokerto\n";
        cout << "1. Tambah Data\n";
        cout << "2. Hapus Data\n";
        cout << "3. Update Data\n";
        cout << "4. Tambah Data Urutan Tertentu\n";
        cout << "5. Hapus Data Urutan Tertentu\n";
        cout << "6. Hapus Seluruh Data\n";
        cout << "7. Tampilkan Data\n";
        cout << "8. Exit\n";
        cout << "Pilih menu: ";
        cin >> pilihan;

        switch (pilihan) {
            case 1:
                cout << "Masukkan nama produk: ";
                cin >> namaProduk;
                cout << "Masukkan harga produk: ";
                cin >> harga;
                dll.tambahData(namaProduk, harga);
                break;
```

```
case 2:
                cout << "Masukkan nama produk yang
akan dihapus: ";
                cin >> namaProduk;
                dll.hapusData(namaProduk);
                break;
            case 3:
                cout << "Masukkan nama lama produk:
";
                cin >> namaLama;
                cout << "Masukkan nama baru produk:
";
                cin >> namaBaru;
                cout << "Masukkan harga baru
produk: ";
                cin >> harga;
                dll.updateData(namaLama, namaBaru,
harga);
                break;
            case 4:
                cout << "Masukkan nama produk yang
akan ditambahkan: ";
                cin >> namaProduk;
                cout << "Masukkan harga produk: ";
                cin >> harga;
                cout << "Masukkan urutan untuk
menambahkan produk: ";
                cin >> urutan;
                dll.tambahDataDiUrutan(namaProduk,
harga, urutan);
                break;
            case 5:
                // Implementasi untuk menghapus
data berdasarkan urutan tertentu
                break;
            case 6:
                dll.clearList();
                cout << "Seluruh data telah
dihapus.\n";
                break;
            case 7:
                dll.tampilkanData();
                break;
            case 8:
                cout << "Terima kasih telah
menggunakan program ini.\n";
                break;
            default:
```

```
cout << "Pilihan tidak valid. Silakan coba
lagi.\n";
}
    } while (pilihan != 8);
}

int main() {
    DoubleLinkedList dll;
    menu(dll);
    return 0;
}
```

Output

```
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
Pilih menu: 7
Toko Skincare Purwokerto
Nama Produk    Harga
originote      60000
somethinc      150000
skintific      100000
wardah         50000
hanasui        30000
```

Case

```
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
Pilih menu: 2
Masukkan nama produk yang akan dihapus: wardah

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
Pilih menu: 7
Toko Skincare Purwokerto
Nama Produk    Harga
originote      60000
somethinc      150000
azarine        65000
skintific      100000
hanasui        30000
```

```
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
Pilih menu: 3
Masukkan nama lama produk: hanasui
Masukkan nama baru produk: cleora
Masukkan harga baru produk: 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
Pilih menu: 7
Toko Skincare Purwokerto
Nama Produk    Harga
originote      60000
somethinc      150000
azarine        65000
skintific      100000
cleora         55000
```

D. KESIMPULAN

Kesimpulan dari program yang telah dibahas adalah sebagai berikut:

- **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.

E. DAFTAR PUSTAKA

1. Pranowo. (2018). Penerapan Struktur Data Linked List untuk Pengelolaan Data Dinamis. Jurnal Teknologi Informasi dan Ilmu Komputer.
2. Sutanta, E. (2010). Struktur Data Untuk Pemrograman. Graha Ilmu.