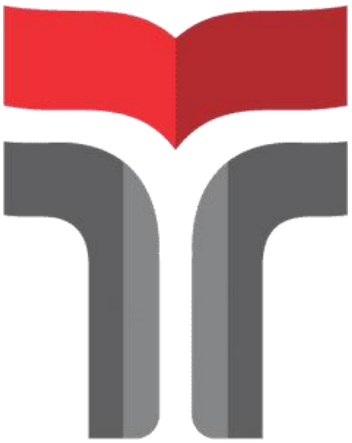
LAPORAN PRAKTIKUM STRUKTUR DATA DAN PEMROGRAGAMAN

MODUL 3

SINGLE AND DOUBLE LINKED LIST



Disusun oleh :

Rafa Aldhino Fatin

2311102023

IF-11-A

Dosen Pengampu :

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

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

PURWOKERTO

2023

BAB I

TUJUAN PRAKTIKUM

1. Mahasiswa dapat memahami konsep single and double linked list
2. Mahasiswa mampu menerapkan single dan double linked list ke dalam pemograman

BAB II DASAR TEORI

SINGLE LINKED LIST

Single linkedlist 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 vorteks.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.

DOUBLE LINKED LIST

Double linked list adalah struktur data linked list yang mirip dengan single linked list, namun dengan tambahan satu 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.

BAB III

GUIDED

GUIDED 1

SOURCE CODE

```
#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)
```

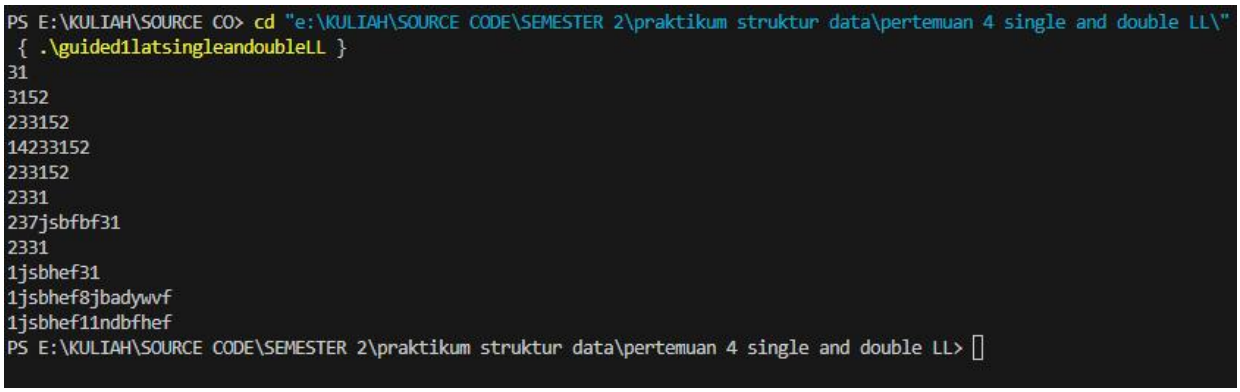
```
{
        bantu = bantu->next;
        nomor++;
    }
    baru->next = bantu->next;
    bantu->next = baru;
  }
}

// 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
  {
```

```
// 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)
    {
```

```
cout << "List masih kosong!" << endl;
    }
}

int main()
{
    init();
    insertDepan(3, "1");
    tampil();
    insertBelakang(5, "2");
    tampil();
    insertDepan(2, "3");
    tampil();
    insertDepan(1, "4");
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7,"jsbfbf" ,2);
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1, "jsbhef");
    tampil();
    ubahBelakang(8, "jbadywvf");
    tampil();
    ubahTengah(11,"ndbfhef", 2);
    tampil();
    return 0;
}
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Program ini mendemonstrasikan berbagai operasi pada single linked list,

seperti: Menambahkan node di depan dan belakang list.Menghitung jumlah node dalam list.

Menambahkan node di posisi tertentu dalam list. Menghapus node di depan, belakang, dan posisi tertentu dalam list. Mengubah data dan kata pada node di depan, belakang, dan posisi tertentu

dalam list.Menghapus semua node dalam list.

.

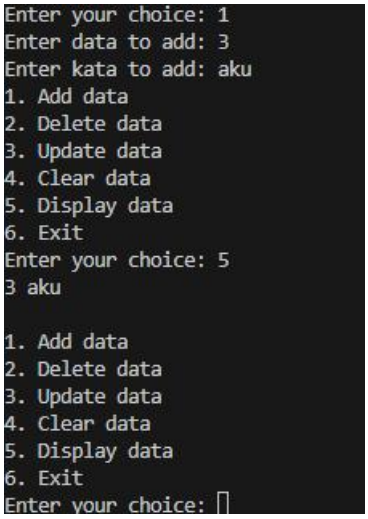
GUIDED 2

SOURCE CODE

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

class Node {
   public:
   int data;
   string kata;
   Node* prev;
   Node* next;
};

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

   void push(int data, string kata) {
      Node* newNode = new Node;
      newNode->data = data;
      newNode->kata = kata;
      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, string newKata) {
      Node* current = head;
      while (current != nullptr) {
         if (current->data == oldData) {
            current->data = newData;
```

```
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;
            string kata;
            cout << "Enter data to add: ";
            cin >> data;
            cout << "Enter kata to add: ";
            cin >> kata;
            list.push(data,kata);
            break;
         }
         case 2: {
            list.pop();
            break;
         }
         case 3: {
            int oldData, newData;
            string newKata;
            cout << "Enter old data: ";
            cin >> oldData;
            cout << "Enter new data: ";
            cin >> newData;
            cout << "Enter new kata: ";
            cin >> newKata;
            bool updated = list.update(oldData,
            newData, newKata);
            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;
         }
      }
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Program ini mendemonstrasikan struktur data doubly linked list dalam C++. Ini memungkinkan untuk menjelajah ke depan dan ke belakang melalui daftar. Berikut adalah deskripsi komponen utamanya:

1. Kelas Node:

Mewakili sebuah node dalam daftar Berisi atribut:

- data: Integer untuk menyimpan data.
- kata: String untuk menyimpan string.
- prev: Pointer ke node sebelumnya.
- next: Pointer ke node berikutnya.

2. Kelas DoublyLinkedList:

Mengelola struktur linked list.

- Berisi metode utama:
- -push(data, kata): Menambahkan node baru di depan daftar.
- -pop(): Menghapus node di depan daftar.
- -update(oldData, newData, newKata): Mengubah data dan kata pada node dengan nilai
- oldData.
- -deleteAll(): Menghapus semua node dalam daftar.
- -display(): Menampilkan data dan kata dari semua node dalam daftar.

3. Fungsi main():

- -Menciptakan objek DoublyLinkedList.
- -Menyajikan menu untuk pengguna agar dapat berinteraksi dengan daftar:
- 1.Menambahkan data.
- 2.Menghapus data.
- 3.Memperbarui data.
- 4.Menghapus semua data.
- 5.Menampilkan data.
- Keluar dari program.

.

BAB IV

UNGUIDED

UNGUIDED 1

SOURCE CODE

```
#include <iostream>

using namespace std;
class Node
{
public:
  string name;
  int age;
  Node *next;
};
class LinkedList
{
public:
  Node *head;
  LinkedList()
  {
    head = NULL;
  }
  void insertAtFront(string name, int age)
  {
    Node *newNode = new Node();
    newNode->name = name;
    newNode->age = age;
    newNode->next = head;
    head = newNode;
  }
void insertAtEnd(string name, int age)
  {
    Node *newNode = new Node();
    newNode->name = name;
    newNode->age = age;
    newNode->next = NULL;
    if (head == NULL)
    {
      head = newNode;
      return;
    }
    Node *temp = head;
    while (temp->next != NULL)
    {
      temp = temp->next;
    }
    temp->next = newNode;
  }
void insertAfter(string name, int age, string keyName)
  {
    Node *temp = head;
    while (temp != NULL)
    {
      if (temp->name == keyName)
      {
        Node *newNode = new Node();
        newNode->name = name;
        newNode->age = age;
        newNode->next = temp->next;
```

```
}
      temp = temp->next;
    }
    cout << keyName << " not found in the list." << endl;
  }
  void updateNode(string name, int age)
  {
    Node *temp = head;
    while (temp != NULL)
    {
      if (temp->name == name)
      {
        temp->age = age;
        return;
      }
      temp = temp->next;
    }
    cout << name << " not found in the list." << endl;
  }
  void deleteNode(string name)
  {
    Node *temp = head;
    Node *prev = NULL;
    while (temp != NULL)
    {
      if (temp->name == name)
      {
        if (prev == NULL)
        {
          head = temp->next;
        }
        else
        {
          prev->next = temp->next;
        }
        delete temp;
        return;
}
      prev = temp;
      temp = temp->next;
    }
    cout << name << " not found in the list." << endl;
  }
  void clearAll()
  {
    Node *temp = head;
    while (temp != NULL)
    {
      Node *next = temp->next;
      delete temp;
      temp = next;
    }
    head = NULL;
  }
 void display()
  {
    Node *temp = head;
    while (temp != NULL)
    {
      cout << "Name: " << temp->name << ", Age: " << temp->age << endl;
      temp = temp->next;
    }
  }
```

```
};
// Main function
int main()
{
  LinkedList list;
  int choice;
  string name, keyName;
  int age;
  do
  {
    cout << endl;
    cout << "MENU" << endl;
    cout << "1. Add data" << endl;
    cout << "2. Update data" << endl;
    cout << "3. Delete data" << endl;
    cout << "4. Clear data" << endl;
    cout << "5. Display data" << endl;
    cout << "6. Exit" << endl;
    cout << "Enter your choice: ";
    cin >> choice;
    switch (choice)
    {
    case 1:
      cout << "Enter name: ";
      cin >> name;
      cout << "Enter age: ";
      cin >> age;
      // Output
      // a. Masukkan data sesuai urutan
      list.insertAtFront(name, age);
      break;
case 2:
      cout << "Enter name to update: ";
      cin >> name;
      cout << "Enter new age: ";
      cin >> age;
      list.updateNode(name, age);
      break;
    case 3:
      cout << "Enter name to delete: ";
      cin >> name;
      list.deleteNode(name);
      break;
    case 4:
      list.clearAll();
      break;
    case 5:
      list.display();
      break;
    case 6:
      cout << "Exiting program..." << endl;
      break;
    default:
      cout << "Invalid choice." << endl;
    }
  } while (choice != 6);
  return 0;
}
```

SCREENSHOOT CODE

Output A

```
MENU
1. Add data
2. Update data
3. Delete data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
Name: karin, Age: 18
Name: hoshino, Age: 18
Name: akechi, Age: 18
Name: yusuke, Age: 19
Name: michael, Age: 18
Name: jane, Age: 20
Name: john, Age: 19
```

Output B

```
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
Name: karin, Age: 18
Name: hoshino, Age: 18
Name: yusuke, Age: 19
Name: michael, Age: 18
Name: jane, Age: 20
Name: john, Age: 19
```

Output C

```
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
Name: futaba, Age: 18
Name: karin, Age: 18
Name: hoshino, Age: 18
Name: yusuke, Age: 19
Name: michael, Age: 18
Name: jane, Age: 20
Name: john, Age: 19
```

Output D

```
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
Name: Igor, Age: 20
Name: futaba, Age: 18
Name: karin, Age: 18
Name: hoshino, Age: 18
Name: yusuke, Age: 19
Name: michael, Age: 18
Name: jane, Age: 20
Name: john, Age: 19
```

Output E

```
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
Name: Reyn, Age: 18
Name: Igor, Age: 20
Name: futaba, Age: 18
Name: karin, Age: 18
Name: hoshino, Age: 18
Name: yusuke, Age: 19
Name: jane, Age: 20
Name: john, Age: 19
```

Output F

```
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
Name: Reyn, Age: 18
Name: Igor, Age: 20
Name: futaba, Age: 18
Name: karin, Age: 18
Name: hoshino, Age: 18
Name: yusuke, Age: 19
Name: jane, Age: 20
Name: john, Age: 19
```

DESKRIPSI PROGRAM

Menciptakan objek LinkedList bernama list.Menampilkan menu secara berulang kepada pengguna hingga pengguna memilih opsi keluar (6).Menu tersebut berisi pilihan untuk:Menambah data (Pilihan 1).Memperbarui data (Pilihan 2).Menghapus data (Pilihan 3).Menghapus semua data (Pilihan 4).Menampilkan semua data (Pilihan 5).Keluar dari program (Pilihan 6).Berdasarkan pilihan pengguna, fungsi terkait dari kelas LinkedList akan Dipanggil.

UNGUIDED 2

SOURCE CODE

```
#include <iostream>
#include <string>
using namespace std;
struct Node
{
  string nama;
  int harga;
  Node *prev;
  Node *next;
};
class DoubleLinkedList
{
private:
  Node *head;
  Node *tail;
  int size;

public:
  DoubleLinkedList()
  {
    head = NULL;
    tail = NULL;
    size = 0;
  }
  void addData(string nama, int harga)
  {
    Node *node = new Node;
    node->nama = nama;
    node->harga = harga;
    node->prev = tail;
    node->next = NULL;
    if (head == NULL)
    {
      head = node;
      tail = node;
    }
    else
    {
      tail->next = node;
      tail = node;
    }
    size++;
  }
void addDataAt(int index, string nama, int harga)
  {
    if (index < 0 || index > size)
    {
      cout << "Index out of bounds" << endl;
      return;
    }
    Node *node = new Node;
    node->nama = nama;
    node->harga = harga;
    if (index == 0)
    {
      node->prev = NULL;
      node->next = head;
      head->prev = node;
      head = node;
```

```
else if (index == size)
    {
      node->prev = tail;
      node->next = NULL;
      tail->next = node;
      tail = node;
    }
    else
    {
      Node *current = head;
      for (int i = 0; i < index - 1; i++)
      {
        current = current->next;
      }
      node->prev = current;
      node->next = current->next;
      current->next->prev = node;
      current->next = node;
    }
    size++;
  }
void deleteDataAt(int index)
  {
    if (index < 0 || index >= size)
    {
      cout << "Index out of bounds" << endl;
      return;
    }

    if (index == 0)
    {
      Node *temp = head;
      head = head->next;
      head->prev = NULL;
      delete temp;
    }
    else if (index == size - 1)
    {
      Node *temp = tail;
      tail = tail->prev;
      tail->next = NULL;
      delete temp;
    }
    else
    {
Node *current = head;
      for (int i = 0; i < index; i++)
      {
        current = current->next;
      }
      current->prev->next = current->next;
      current->next->prev = current->prev;
      delete current;
    }
    size--;
  }
void clearData()
  {
    while (head != NULL)
    {
      Node *temp = head;
      head = head->next;
      delete temp;
```

```
tail = NULL;
    size = 0;
  }
  void displayData()
  {
    cout << "Nama Produk\tHarga" << endl;
    Node *current = head;
    while (current != NULL)
    {
      cout << current->nama << "\t\t" << current->harga << endl;
      current = current->next;
    }
  }
  void updateDataAt(int index, string nama, int harga)
  {
    if (index < 0 || index >= size)
    {
      cout << "Index out of bounds" << endl;
      return;
    }
    Node *current = head;
    for (int i = 0; i < index; i++)
    {
      current = current->next;
    }
    current->nama = nama;
    current->harga = harga;
  }
};
int main()
{
  DoubleLinkedList dll;
  int choice;
  string nama;
  int harga;
  int index;
  do
  {
    cout << "Menu:" << endl;
    cout << "1. Tambah Data" << endl;
    cout << "2. Hapus Data" << endl;
    cout << "3. Update Data" << endl;
    cout << "4. Tambah Data pada Urutan Tertentu" << endl;
    cout << "5. Hapus Data pada Urutan Tertentu" << endl;
    cout << "6. Hapus Semua Data" << endl;
    cout << "7. Tampilkan Data" << endl;
    cout << "8. Keluar" << endl;
    cout << "Pilih: ";
    cin >> choice;
    switch (choice)
{
    case 1:
      cout << "Nama Produk: ";
      cin >> nama;
      cout << "Harga: ";
      cin >> harga;
      dll.addData(nama, harga);
      break;
    case 2:
      cout << "Index: ";
      cin >> index;
      dll.deleteDataAt(index);
      break;
```

```
case 3:
      cout << "Index: ";
      cin >> index;
      cout << "Nama Produk: ";
      cin >> nama;
      cout << "Harga: ";
      cin >> harga;
      dll.updateDataAt(index, nama, harga);
      break;
    case 4:
      cout << "Index: ";
      cin >> index;
      cout << "Nama Produk: ";
      cin >> nama;
      cout << "Harga: ";
      cin >> harga;
      dll.addDataAt(index, nama, harga);
      break;
    case 5:
      cout << "Index: ";
      cin >> index;
      dll.deleteDataAt(index);
      break;
case 6:
      dll.clearData();
      break;
    case 7:
      dll.displayData();
      break;
    case 8:
      break;
    default:
      cout << "Pilihan tidak valid" << endl;
      break;
    }
    cout << endl;
  } while (choice != 8);
  return 0;
}
```

SCREENSHOOT PROGRAM

OUTPUT A

```
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 7
Nama Produk     Harga
originote               60000
somethinc               150000
azarin          650000
skintific               100000
wardah          50
hanasui         30000
```

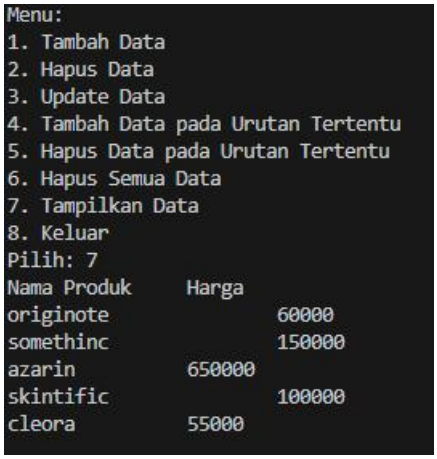
OUTPUT B

```
Menu:
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 7
Nama Produk     Harga
originote               60000
somethinc               150000
azarin          650000
skintific               100000
hanasui         30000
```

OUTPUT C

```
Menu:
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 7
Nama Produk     Harga
originote               60000
somethinc               150000
azarin          650000
skintific               100000
cleora          55000
```

OUTPUT D



DESKRIPSI / FUNGSI PROGRAM

Program tersebut menyediakan operasi dasar untuk mengelola linked list berganda, seperti tambah data, update data, dan tampil data. Setiap kali fungsi ini di eksekusi program akan menampilkan menu dan menunggu input dari user , ketika user sudah memasukkan input, makan progam akan mengeksekusi perintah yang sudah di inputkan oleh user.

BAB V

KESIMPULAN

Single linked list dan double linked list adalah dua struktur data fundamental yang menyimpan data secara linear. Perbedaan utama terletak pada jumlah pointer yang dimiliki setiap node. dimana Single linked list Setiap node memiliki satu pointer yang menunjuk ke node berikutnya, kemudian Double linked list Setiap node memiliki dua pointer, satu menunjuk ke node berikutnya dan satu menunjuk ke node sebelumnya.

DAFTAR PUSTAKA

Perbedaan antara single linkedlist dengan double linkedlist di c++

https://byjus.com/gate/difference-between-singly-linked-list-doubly-linked-list/