LAPORAN PRAKTIKUM

MODUL III
SINGLE AND DOUBLE LINKED LIST

Disusun oleh:
Anita Nurazizah Agussalim
NIM: 2311102017

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

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

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.

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

1. 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
    {
        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 << endl;
            cout << bantu->kata << endl;
            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, "tujuh");
    tampil();
    ubahBelakang(8, "delapan");
    tampil();
    ubahTengah(11, "sembilan", 2);
    tampil();
    return 0;
}
```

Screenshoot program

```cpp
#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)
{
```

```
50      // Buat Node baru
51      Node *baru = new Node;
52      baru->data = nilai;
53      baru->kata = kata;
54      baru->next = NULL;
55      if (isEmpty() == true)
56      {
57          head = tail = baru;
58          tail->next = NULL;
59      }
60      else
61      {
62          tail->next = baru;
63          tail = baru;
64      }
65  }
66  // Hitung Jumlah List
67  int hitungList()
68  {
69      Node *hitung;
70      hitung = head;
71      int jumlah = 0;
72      while (hitung != NULL)
73      {
74          jumlah++;
75          hitung = hitung->next;
76      }
77      return jumlah;
78  }
79  // Tambah Tengah
80  void insertTengah(int data, string kata, int posisi)
81  {
82      if (posisi < 1 || posisi > hitungList())
83      {
84          cout << "Posisi diluar jangkauan" << endl;
85      }
86      else if (posisi == 1)
87      {
88          cout << "Posisi bukan posisi tengah" << endl;
89      }
90      else
91      {
92          Node *baru, *bantu;
93          baru = new Node();
94          baru->data = data;
95          baru->kata = kata;
```

```
96          // tranversing
97          bantu = head;
98          int nomor = 1;
99          while (nomor < posisi - 1)
100         {
101             bantu = bantu->next;
102             nomor++;
103         }
104         baru->next = bantu->next;
105         bantu->next = baru;
106     }
107 }
108 // Hapus Depan
109 void hapusDepan()
110 {
111     Node *hapus;
112     if (isEmpty() == false)
113     {
114         if (head->next != NULL)
115         {
116             hapus = head;
117             head = head->next;
118             delete hapus;
119         }
120         else
121         {
122             head = tail = NULL;
123         }
124     }
125     else
126     {
127         cout << "List kosong!" << endl;
128     }
129 }
130 // Hapus Belakang
131 void hapusBelakang()
132 {
133     Node *hapus;
134     Node *bantu;
135     if (isEmpty() == false)
136     {
137         if (head != tail)
138         {
139             hapus = tail;
140             bantu = head;
141             while (bantu->next != tail)
```

```
142             {
143                 bantu = bantu->next;
144             }
145             tail = bantu;
146             tail->next = NULL;
147             delete hapus;
148         }
149         else
150         {
151             head = tail = NULL;
152         }
153     }
154     else
155     {
156         cout << "List kosong!" << endl;
157     }
158 }
159 // Hapus Tengah
160 void hapusTengah(int posisi)
161 {
162     Node *hapus, *bantu, *bantu2;
163     if (posisi < 1 || posisi > hitungList())
164     {
165         cout << "Posisi di luar jangkauan" << endl;
166     }
167     else if (posisi == 1)
168     {
169         cout << "Posisi bukan posisi tengah" << endl;
170     }
171     else
172     {
173         int nomor = 1;
174         bantu = head;
175         while (nomor <= posisi)
176         {
177             if (nomor == posisi - 1)
178             {
179                 bantu2 = bantu;
180             }
181             if (nomor == posisi)
182             {
183                 hapus = bantu;
184             }
185             bantu = bantu->next;
186             nomor++;
187         }
188         bantu2->next = bantu;
189         delete hapus;
190     }
```

```
}
// 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;
    }
}
```

```
237  // Ubah Belakang
238  void ubahBelakang(int data, string kata)
239  {
240      if (isEmpty() == false)
241      {
242          tail->data = data;
243          tail->kata = kata;
244      }
245      else
246      {
247          cout << "List masih kosong!" << endl;
248      }
249  }
250  // Hapus List
251  void clearList()
252  {
253      Node *bantu, *hapus;
254      bantu = head;
255      while (bantu != NULL)
256      {
257          hapus = bantu;
258          bantu = bantu->next;
259          delete hapus;
260      }
261      head = tail = NULL;
262      cout << "List berhasil terhapus!" << endl;
263  }
264  // Tampilkan List
265  void tampil()
266  {
267      Node *bantu;
268      bantu = head;
269      if (isEmpty() == false)
270      {
271          while (bantu != NULL)
272          {
273              cout << bantu->data << endl;
274              cout << bantu->kata << endl;
275              bantu = bantu->next;
276          }
277          cout << endl;
278      }
279      else
280      {
281          cout << "List masih kosong!" << endl;
282      }
283  }
```

```
284  int main()
285  {
286      init();
287      insertDepan(3, "satu");
288      tampil();
289      insertBelakang(5, "dua");
290      tampil();
291      insertDepan(2, "tiga");
292      tampil();
293      insertDepan(1, "empat");
294      tampil();
295      hapusDepan();
296      tampil();
297      hapusBelakang();
298      tampil();
299      insertTengah(7, "lima", 2);
300      tampil();
301      hapusTengah(2);
302      tampil();
303      ubahDepan(1, "tujuh");
304      tampil();
305      ubahBelakang(8, "delapan");
306      tampil();
307      ubahTengah(11, "sembilan", 2);
308      tampil();
309      return 0;
310  }
```

Screenshot Output

```
3
satu

3
satu
5
dua

2
tiga
3
satu
5
dua

1
empat
2
tiga
3
satu
5
dua

2
tiga
3
satu
5
dua

2
tiga
3
satu

2
tiga
7
lima
3
satu

2
tiga
3
satu

1
tujuh
3
satu

1
tujuh
8
delapan

1
tujuh
11
sembilan
```

Deskripsi program

Program di atas adalah implementasi dari struktur data linked. Linked list yang dibuat merupakan tipe non-circular, yang terdiri dari sejumlah node yang saling terhubung satu sama lain. Setiap node memiliki dua komponen, yaitu data bertipe integer dan kata bertipe string, serta pointer yang menunjukkan ke node selanjutnya dalam list. Program menyediakan fungsi-fungsi untuk melakukan operasi dasar pada linked list seperti menambahkan node di depan, di belakang, di tengah, menghapus node di depan, di belakang, di tengah, mengubah nilai node di depan, di belakang, di tengah, menghitung jumlah node dalam list, serta menampilkan isi dari linked list. Program tersebut kemudian diuji coba dengan beberapa operasi dasar pada linked list seperti penambahan, penghapusan, dan pengubahan nilai node.

2. 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;
                current->kata = newKata;
                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 << " ";
            cout << current->kata << endl;
            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;
            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;
        }
        }
    }
    return 0;
}
```

Screenshoot program

```cpp
#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;
    }

```

```
42      void pop()
43      {
44          if (head == nullptr)
45          {
46              return;
47          }
48          Node *temp = head;
49          head = head->next;
50          if (head != nullptr)
51          {
52              head->prev = nullptr;
53          }
54          else
55          {
56              tail = nullptr;
57          }
58          delete temp;
59      }
60
61      bool update(int oldData, int newData, string newKata)
62      {
63          Node *current = head;
64          while (current != nullptr)
65          {
66              if (current->data == oldData)
67              {
68                  current->data = newData;
69                  current->kata = newKata;
70                  return true;
71              }
72              current = current->next;
73          }
74          return false;
75      }
76
77      void deleteAll()
78      {
79          Node *current = head;
80          while (current != nullptr)
81          {
82              Node *temp = current;
83              current = current->next;
84              delete temp;
```

```
85          }
86          head = nullptr;
87          tail = nullptr;
88      }
89
90      void display()
91      {
92          Node *current = head;
93          while (current != nullptr)
94          {
95              cout << current->data << " ";
96              cout << current->kata << endl;
97              current = current->next;
98          }
99          cout << endl;
100     }
101 };
102
103 int main()
104 {
105     DoublyLinkedList list;
106     while (true)
107     {
108         cout << "1. Add data" << endl;
109         cout << "2. Delete data" << endl;
110         cout << "3. Update data" << endl;
111         cout << "4. Clear data" << endl;
112         cout << "5. Display data" << endl;
113         cout << "6. Exit" << endl;
114         int choice;
115         cout << "Enter your choice: ";
116         cin >> choice;
117         switch (choice)
118         {
119         case 1:
120         {
121             int data;
122             string kata;
123             cout << "Enter data to add: ";
124             cin >> data;
125             cout << "Enter kata to add: ";
126             cin >> kata;
127             list.push(data, kata);
128             break;
129         }
```

```
130         case 2:
131         {
132             list.pop();
133             break;
134         }
135         case 3:
136         {
137             int oldData, newData;
138             string newKata;
139             cout << "Enter old data: ";
140             cin >> oldData;
141             cout << "Enter new data: ";
142             cin >> newData;
143             cout << "Enter new kata: ";
144             cin >> newKata;
145             bool updated = list.update(oldData,
146                                        newData, newKata);
147             if (!updated)
148             {
149                 cout << "Data not found" << endl;
150             }
151             break;
152         }
153         case 4:
154         {
155             list.deleteAll();
156             break;
157         }
158         case 5:
159         {
160             list.display();
161             break;
162         }
163         case 6:
164         {
165             return 0;
166         }
167         default:
168         {
169             cout << "Invalid choice" << endl;
170             break;
171         }
172         }
173     }
174     return 0;
175 }
```

Screenshot Output

```
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: 13
Enter kata to add: tylerthecreator
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 3
Enter old data: 13
Enter new data: 8888
Enter new kata: asaprocky
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
8888 asaprocky

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

Deskripsi program

Program di atas adalah implementasi dari struktur data Doubly Linked List dalam bahasa C++. Doubly Linked List adalah jenis linked list di mana setiap node memiliki dua pointer, yaitu satu yang menunjuk ke node sebelumnya (prev) dan satu yang menunjuk ke node berikutnya (next). Program ini memanfaatkan kelas `Node` untuk merepresentasikan node dalam linked list dan kelas `DoublyLinkedList` untuk mengelola operasi-operasi pada linked list tersebut. Operasi yang dapat dilakukan meliputi menambahkan data di awal linked list, menghapus data dari awal linked list, mengubah data, menghapus semua data dalam linked list, dan menampilkan isi linked list. Program menyediakan antarmuka sederhana berupa pilihan menu yang memungkinkan pengguna untuk melakukan operasi tersebut.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

Source code

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

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

class LinkedList {
private:
    Mahasiswa* head;

public:
    LinkedList() {
        head = nullptr;
    }

    void tambahMahasiswa(string nama, int usia) {
        Mahasiswa* newMahasiswa = new Mahasiswa;
        newMahasiswa->nama = nama;
        newMahasiswa->usia = usia;
        newMahasiswa->next = head;
        head = newMahasiswa;
    }

    void hapusMahasiswa(string nama) {
        Mahasiswa* current = head;
```

```
        Mahasiswa* prev = nullptr;

        while (current != nullptr) {
            if (current->nama == nama) {
                if (prev == nullptr) {
                    head = current->next;
                } else {
                    prev->next = current->next;
                }
                delete current;
                cout << "Data mahasiswa dengan nama " << nama <<
" telah dihapus." << endl;
                return;
            }
            prev = current;
            current = current->next;
        }

        cout << "Data mahasiswa dengan nama " << nama << " tidak
ditemukan." << endl;
    }

    void  tambahMahasiswaDiUrutan(string  nama,  int  usia,  int
urutan) {
        Mahasiswa* newMahasiswa = new Mahasiswa;
        newMahasiswa->nama = nama;
        newMahasiswa->usia = usia;

        if (urutan == 1 || head == nullptr) {
            newMahasiswa->next = head;
            head = newMahasiswa;
        } else {
            Mahasiswa* current = head;
            int posisi = 1;
```

```
            while (posisi < urutan - 1 && current->next != nullptr) {
                current = current->next;
                posisi++;
            }

            newMahasiswa->next = current->next;
            current->next = newMahasiswa;
        }
    }

    void gantiDataMahasiswa(string nama, string namaBaru, int usiaBaru) {
        Mahasiswa* current = head;

        while (current != nullptr) {
            if (current->nama == nama) {
                current->nama = namaBaru;
                current->usia = usiaBaru;
                cout << "Data mahasiswa dengan nama " << nama << " telah diubah." << endl;
                return;
            }
            current = current->next;
        }

        cout << "Data mahasiswa dengan nama " << nama << " tidak ditemukan." << endl;
    }

    void tampilkanDataAwal() {
        Mahasiswa* current = head;
```

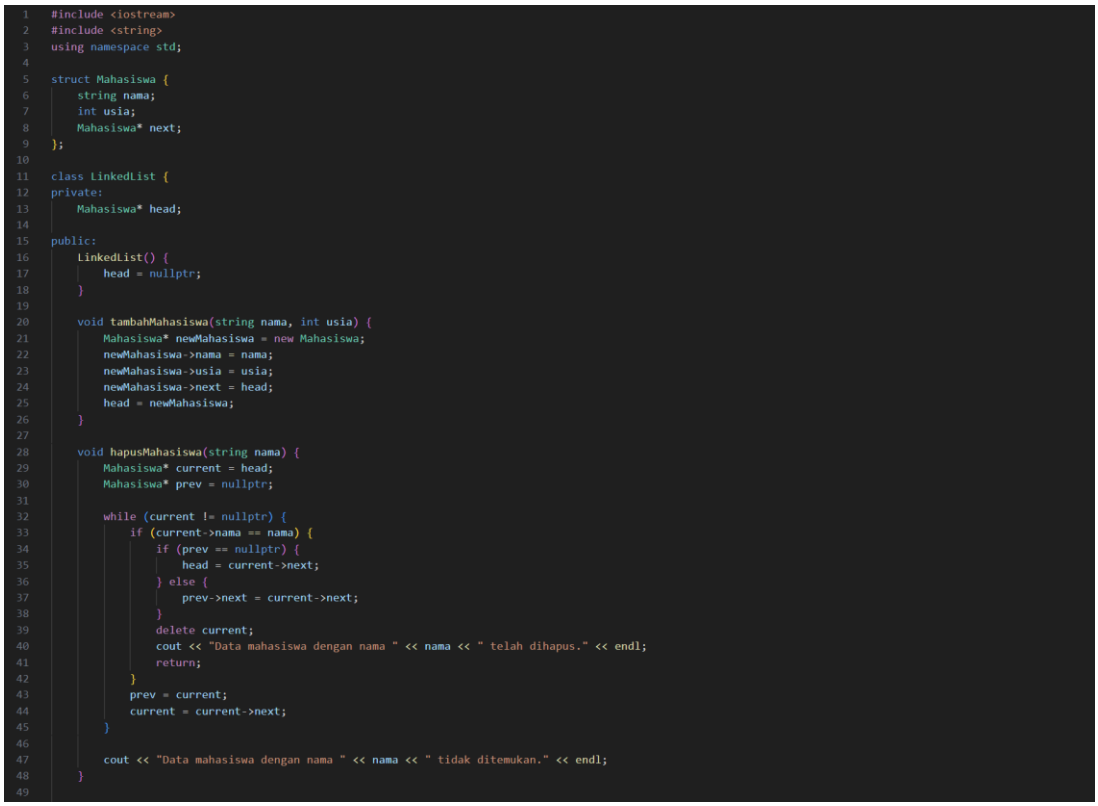
```
        if (current == nullptr) {
            cout << "Linked list kosong." << endl;
            return;
        }

        cout << "Data Awal Mahasiswa:" << endl;
        while (current != nullptr) {
            cout << current->nama << " " << current->usia <<
endl;
            current = current->next;
        }
        cout << endl;
    }

    void tampilkanMahasiswa() {
        Mahasiswa* current = head;

        if (current == nullptr) {
            cout << "Linked list kosong." << endl;
            return;
        }

        cout << "Data Mahasiswa:" << endl;
        cout << "-----------------" << endl;
        while (current != nullptr) {
            cout << current->nama << " " << current->usia <<
endl;
            current = current->next;
        }
        cout << endl;
    }
};

int main() {
```

```
    LinkedList daftarMahasiswa;

    daftarMahasiswa.tambahMahasiswa("Karin", 18);
    daftarMahasiswa.tambahMahasiswa("Hoshino", 18);
    daftarMahasiswa.tambahMahasiswa("Akechi", 20);
    daftarMahasiswa.tambahMahasiswa("Yusuke", 19);
    daftarMahasiswa.tambahMahasiswa("Michael", 18);
    daftarMahasiswa.tambahMahasiswa("Jane", 20);
    daftarMahasiswa.tambahMahasiswa("John", 19);

    daftarMahasiswa.tampilkanDataAwal();

    int pilihan;
    string nama, namaBaru;
    int usia, usiaBaru, urutan;

    do {
        cout << "\nMenu:\n";
        cout << "1. Tambah Data Mahasiswa\n";
        cout << "2. Hapus Data Mahasiswa\n";
        cout << "3. Tambah Data Mahasiswa di Urutan Tertentu\n";
        cout << "4. Ubah Data Mahasiswa\n";
        cout << "5. Tampilkan Semua Data Mahasiswa\n";
        cout << "6. Keluar\n";
        cout << "Pilih menu: ";
        cin >> pilihan;

        switch (pilihan) {
            case 1:
                cout << "Masukkan nama mahasiswa: ";
                cin >> nama;
                cout << "Masukkan usia mahasiswa: ";
                cin >> usia;
                daftarMahasiswa.tambahMahasiswa(nama, usia);
```

```
                break;
            case 2:
                cout << "Masukkan nama mahasiswa yang ingin
dihapus: ";
                cin >> nama;
                daftarMahasiswa.hapusMahasiswa(nama);
                break;
            case 3:
                cout << "Masukkan nama mahasiswa: ";
                cin >> nama;
                cout << "Masukkan usia mahasiswa: ";
                cin >> usia;
                cout << "Masukkan urutan: ";
                cin >> urutan;
                daftarMahasiswa.tambahMahasiswaDiUrutan(nama,
usia, urutan);
                break;
            case 4:
                cout << "Masukkan nama mahasiswa yang ingin
diubah: ";
                cin >> nama;
                cout << "Masukkan nama baru: ";
                cin >> namaBaru;
                cout << "Masukkan usia baru: ";
                cin >> usiaBaru;
                daftarMahasiswa.gantiDataMahasiswa(nama,
namaBaru, usiaBaru);
                break;
            case 5:
                daftarMahasiswa.tampilkanMahasiswa();
                break;
            case 6:
                cout << "Terima kasih!" << endl;
                break;
```

```
            default:
                cout << "Pilihan tidak valid. Silakan pilih
lagi." << endl;
        }
    } while (pilihan != 6);

    return 0;
}
```

Screenshoot program



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

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

class LinkedList {
private:
    Mahasiswa* head;

public:
    LinkedList() {
        head = nullptr;
    }

    void tambahMahasiswa(string nama, int usia) {
        Mahasiswa* newMahasiswa = new Mahasiswa;
        newMahasiswa->nama = nama;
        newMahasiswa->usia = usia;
        newMahasiswa->next = head;
        head = newMahasiswa;
    }

    void hapusMahasiswa(string nama) {
        Mahasiswa* current = head;
        Mahasiswa* prev = nullptr;

        while (current != nullptr) {
            if (current->nama == nama) {
                if (prev == nullptr) {
                    head = current->next;
                } else {
                    prev->next = current->next;
                }
                delete current;
                cout << "Data mahasiswa dengan nama " << nama << " telah dihapus." << endl;
                return;
            }
            prev = current;
            current = current->next;
        }

        cout << "Data mahasiswa dengan nama " << nama << " tidak ditemukan." << endl;
    }

```

```
    void tambahMahasiswaDiUrutan(string nama, int usia, int urutan) {
        Mahasiswa* newMahasiswa = new Mahasiswa;
        newMahasiswa->nama = nama;
        newMahasiswa->usia = usia;

        if (urutan == 1 || head == nullptr) {
            newMahasiswa->next = head;
            head = newMahasiswa;
        } else {
            Mahasiswa* current = head;
            int posisi = 1;

            while (posisi < urutan - 1 && current->next != nullptr) {
                current = current->next;
                posisi++;
            }

            newMahasiswa->next = current->next;
            current->next = newMahasiswa;
        }
    }

    void gantiDataMahasiswa(string nama, string namaBaru, int usiaBaru) {
        Mahasiswa* current = head;

        while (current != nullptr) {
            if (current->nama == nama) {
                current->nama = namaBaru;
                current->usia = usiaBaru;
                cout << "Data mahasiswa dengan nama " << nama << " telah diubah." << endl;
                return;
            }
            current = current->next;
        }

        cout << "Data mahasiswa dengan nama " << nama << " tidak ditemukan." << endl;
    }

    void tampilkanDataAwal() {
        Mahasiswa* current = head;

        if (current == nullptr) {
            cout << "Linked list kosong." << endl;
            return;
        }

        cout << "Data Awal Mahasiswa:" << endl;
        while (current != nullptr) {
            cout << current->nama << " " << current->usia << endl;
            current = current->next;
        }
        cout << endl;
    }
```

```
    void tampilkanMahasiswa() {
        Mahasiswa* current = head;

        if (current == nullptr) {
            cout << "Linked list kosong." << endl;
            return;
        }

        cout << "Data Mahasiswa:" << endl;
        cout << "----------------" << endl;
        while (current != nullptr) {
            cout << current->nama << " " << current->usia << endl;
            current = current->next;
        }
        cout << endl;
    }
};

int main() {
    LinkedList daftarMahasiswa;

    daftarMahasiswa.tambahMahasiswa("Karin", 18);
    daftarMahasiswa.tambahMahasiswa("Hoshino", 18);
    daftarMahasiswa.tambahMahasiswa("Akechi", 20);
    daftarMahasiswa.tambahMahasiswa("Yusuke", 19);
    daftarMahasiswa.tambahMahasiswa("Michael", 18);
    daftarMahasiswa.tambahMahasiswa("Jane", 20);
    daftarMahasiswa.tambahMahasiswa("John", 19);

    daftarMahasiswa.tampilkanDataAwal();

    int pilihan;
    string nama, namaBaru;
    int usia, usiaBaru, urutan;

    do {
        cout << "\nMenu:\n";
        cout << "1. Tambah Data Mahasiswa\n";
        cout << "2. Hapus Data Mahasiswa\n";
        cout << "3. Tambah Data Mahasiswa di Urutan Tertentu\n";
        cout << "4. Ubah Data Mahasiswa\n";
        cout << "5. Tampilkan Semua Data Mahasiswa\n";
        cout << "6. Keluar\n";
        cout << "Pilih menu: ";
        cin >> pilihan;
```

```
        switch (pilihan) {
            case 1:
                cout << "Masukkan nama mahasiswa: ";
                cin >> nama;
                cout << "Masukkan usia mahasiswa: ";
                cin >> usia;
                daftarMahasiswa.tambahMahasiswa(nama, usia);
                break;
            case 2:
                cout << "Masukkan nama mahasiswa yang ingin dihapus: ";
                cin >> nama;
                daftarMahasiswa.hapusMahasiswa(nama);
                break;
            case 3:
                cout << "Masukkan nama mahasiswa: ";
                cin >> nama;
                cout << "Masukkan usia mahasiswa: ";
                cin >> usia;
                cout << "Masukkan urutan: ";
                cin >> urutan;
                daftarMahasiswa.tambahMahasiswaDiUrutan(nama, usia, urutan);
                break;
            case 4:
                cout << "Masukkan nama mahasiswa yang ingin diubah: ";
                cin >> nama;
                cout << "Masukkan nama baru: ";
                cin >> namaBaru;
                cout << "Masukkan usia baru: ";
                cin >> usiaBaru;
                daftarMahasiswa.gantiDataMahasiswa(nama, namaBaru, usiaBaru);
                break;
            case 5:
                daftarMahasiswa.tampilkanMahasiswa();
                break;
            case 6:
                cout << "Terima kasih!" << endl;
                break;
            default:
                cout << "Pilihan tidak valid. Silakan pilih lagi." << endl;
        }
    } while (pilihan != 6);

    return 0;
}
```

Screenshot output

```
Data Awal Mahasiswa:
John 19
Jane 20
Michael 18
Yusuke 19
Akechi 20
Hoshino 18
Karin 18


Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Tambah Data Mahasiswa di Urutan Tertentu
4. Ubah Data Mahasiswa
5. Tampilkan Semua Data Mahasiswa
6. Keluar
Pilih menu: 1
Masukkan nama mahasiswa: Anita
Masukkan usia mahasiswa: 19

Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Tambah Data Mahasiswa di Urutan Tertentu
4. Ubah Data Mahasiswa
5. Tampilkan Semua Data Mahasiswa
6. Keluar
Pilih menu: 5
Data Mahasiswa:
-----------------
Anita 19
John 19
Jane 20
Michael 18
Yusuke 19
Akechi 20
Hoshino 18
Karin 18
```

```
Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Tambah Data Mahasiswa di Urutan Tertentu
4. Ubah Data Mahasiswa
5. Tampilkan Semua Data Mahasiswa
6. Keluar
Pilih menu: 2
Masukkan nama mahasiswa yang ingin dihapus: Akechi
Data mahasiswa dengan nama Akechi telah dihapus.

Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Tambah Data Mahasiswa di Urutan Tertentu
4. Ubah Data Mahasiswa
5. Tampilkan Semua Data Mahasiswa
6. Keluar
Pilih menu: 5
Data Mahasiswa:
-----------------
Anita 19
John 19
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18
```

```
Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Tambah Data Mahasiswa di Urutan Tertentu
4. Ubah Data Mahasiswa
5. Tampilkan Semua Data Mahasiswa
6. Keluar
Pilih menu: 3
Masukkan nama mahasiswa: Futaba
Masukkan usia mahasiswa: 18
Masukkan urutan: 3

Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Tambah Data Mahasiswa di Urutan Tertentu
4. Ubah Data Mahasiswa
5. Tampilkan Semua Data Mahasiswa
6. Keluar
Pilih menu: 5
Data Mahasiswa:
-----------------
Anita 19
John 19
Futaba 18
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18
```

```
Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Tambah Data Mahasiswa di Urutan Tertentu
4. Ubah Data Mahasiswa
5. Tampilkan Semua Data Mahasiswa
6. Keluar
Pilih menu: 1
Masukkan nama mahasiswa: Igor
Masukkan usia mahasiswa: 20

Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Tambah Data Mahasiswa di Urutan Tertentu
4. Ubah Data Mahasiswa
5. Tampilkan Semua Data Mahasiswa
6. Keluar
Pilih menu: 4
Masukkan nama mahasiswa yang ingin diubah: Michael
Masukkan nama baru: Reyn
Masukkan usia baru: 18
Data mahasiswa dengan nama Michael telah diubah.
```

```
Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Tambah Data Mahasiswa di Urutan Tertentu
4. Ubah Data Mahasiswa
5. Tampilkan Semua Data Mahasiswa
6. Keluar
Pilih menu: 5
Data Mahasiswa:
-----------------
Igor 20
Anita 19
John 19
Futaba 18
Jane 20
Reyn 18
Yusuke 19
Hoshino 18
Karin 18
```

Deskripsi program

Program di atas adalah implementasi dari menu Single Linked List Non-Circular untuk menyimpan data mahasiswa berupa nama dan usia. Program memungkinkan pengguna untuk menambahkan data mahasiswa baru, menghapus data mahasiswa berdasarkan nama, menambahkan data mahasiswa pada urutan tertentu, mengganti data mahasiswa, dan menampilkan semua data mahasiswa yang tersimpan. Data awal mahasiswa yang telah ditambahkan ke dalam linked list ditampilkan sebelum pengguna melakukan perubahan atau manipulasi data. Program berjalan dalam loop hingga pengguna memilih untuk keluar dari menu.

2. Unguided 2

Source code

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

struct Product {
    string nama_produk;
    int harga;
    Product* prev;
    Product* next;
};

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

public:
    DoubleLinkedList() {
        head = NULL;
        tail = NULL;
    }

    void addProduct(string nama_produk, int harga) {
        Product* newProduct = new Product;
        newProduct->nama_produk = nama_produk;
        newProduct->harga = harga;
        newProduct->prev = NULL;
        newProduct->next = NULL;

        if (head == NULL) {
            head = newProduct;
            tail = newProduct;
```

```
        } else {
            tail->next = newProduct;
            newProduct->prev = tail;
            tail = newProduct;
        }
    }

    void removeProduct(string nama_produk) {
        Product* current = head;
        while (current != NULL) {
            if (current->nama_produk == nama_produk) {
                if (current == head) {
                    head = head->next;
                    if (head != NULL) {
                        head->prev = NULL;
                    }
                } else if (current == tail) {
                    tail = tail->prev;
                    tail->next = NULL;
                } else {
                    current->prev->next = current->next;
                    current->next->prev = current->prev;
                }
                delete current;
                return;
            }
            current = current->next;
        }
        cout << "Produk tidak ditemukan." << endl;
    }

    void    updateProduct(string    old_nama_produk,    string
new_nama_produk, int new_harga) {
        Product* current = head;
```

```
        while (current != NULL) {
            if (current->nama_produk == old_nama_produk) {
                current->nama_produk = new_nama_produk;
                current->harga = new_harga;
                return;
            }
            current = current->next;
        }
        cout << "Produk tidak ditemukan." << endl;
    }

    void displayProducts() {
        cout << "Nama Produk\tHarga" << endl;
        cout << "======================" << endl;
        Product* current = head;
        while (current != NULL) {
            cout << setw(12) << left << current->nama_produk <<
"\t" << current->harga << endl;
            current = current->next;
        }
        cout << endl;
    }

    void   addProductBetween(string   prevProductName,   string
newProductName, int newPrice) {
        Product* current = head;
        while (current != NULL) {
            if (current->nama_produk == prevProductName) {
                Product* newProduct = new Product;
                newProduct->nama_produk = newProductName;
                newProduct->harga = newPrice;
                newProduct->prev = current;
                newProduct->next = current->next;
                if (current->next != NULL) {
```

```
                    current->next->prev = newProduct;
                }
                current->next = newProduct;
                return;
            }
            current = current->next;
        }
        cout << "Produk sebelumnya tidak ditemukan." << endl;
    }

    void removeAtPosition(int position) {
        if (head == NULL) {
            cout << "List kosong." << endl;
            return;
        }

        Product* current = head;
        int count = 1;

        while (current != NULL && count != position) {
            current = current->next;
            count++;
        }

        if (current == NULL) {
            cout << "Posisi tidak valid." << endl;
            return;
        }

        if (current == head) {
            head = head->next;
            if (head != NULL) {
                head->prev = NULL;
            }
```

```
            delete current;
        } else if (current == tail) {
            tail = tail->prev;
            tail->next = NULL;
            delete current;
        } else {
            current->prev->next = current->next;
            current->next->prev = current->prev;
            delete current;
        }
    }

    void removeAll() {
        Product* current = head;
        while (current != NULL) {
            Product* temp = current;
            current = current->next;
            delete temp;
        }
        head = NULL;
        tail = NULL;
    }
};

int main() {
    DoubleLinkedList productList;

    productList.addProduct("Originote", 60000);
    productList.addProduct("Somethinc", 150000);
    productList.addProduct("Skintific", 100000);
    productList.addProduct("Wardah", 50000);
    productList.addProduct("Hanasui", 30000);

    int choice;
```

```
    string prevProductName, newProductName;
    int newPrice;

    do {
        cout << "\nToko 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;
        cout << "Pilih menu: ";
        cin >> choice;

        switch(choice) {
            case 1:
                cout << "Masukkan Nama Produk: ";
                cin >> newProductName;
                cout << "Masukkan Harga: ";
                cin >> newPrice;
                productList.addProduct(newProductName,
newPrice);
                break;
            case 2:
                cout << "Masukkan Nama Produk yang Ingin Dihapus:
";
                cin >> newProductName;
                productList.removeProduct(newProductName);
                break;
            case 3:
                cout  <<  "Masukkan  Nama  Produk  yang  Ingin
Diupdate: ";
```

```
                cin >> prevProductName;
                cout << "Masukkan Nama Produk Baru: ";
                cin >> newProductName;
                cout << "Masukkan Harga Baru: ";
                cin >> newPrice;
                productList.updateProduct(prevProductName,
newProductName, newPrice);
                break;
            case 4:
                cout << "Masukkan Nama Produk Sebelumnya: ";
                cin >> prevProductName;
                cout << "Masukkan Nama Produk Baru: ";
                cin >> newProductName;
                cout << "Masukkan Harga Baru: ";
                cin >> newPrice;
                productList.addProductBetween(prevProductName,
newProductName, newPrice);
                break;
            case 5:
            {
                string productName;
                cout << "Masukkan Nama Produk yang Ingin Dihapus:
";
                cin >> productName;
                productList.removeProduct(productName);
                break;
            }
            case 6:
                productList.removeAll();
                cout << "Semua data telah dihapus." << endl;
                break;
            case 7:
                cout << "Data Produk:" << endl;
                productList.displayProducts();
```

```
                break;
            case 8:
                cout << "Terima kasih!" << endl;
                break;
            default:
                cout << "Pilihan tidak valid. Silakan pilih
lagi." << endl;
        }
    } while (choice != 8);

    return 0;
}
```

Screenshoot program

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

struct Product {
    string nama_produk;
    int harga;
    Product* prev;
    Product* next;
};

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

public:
    DoubleLinkedList() {
        head = NULL;
        tail = NULL;
    }

    void addProduct(string nama_produk, int harga) {
        Product* newProduct = new Product;
        newProduct->nama_produk = nama_produk;
        newProduct->harga = harga;
        newProduct->prev = NULL;
        newProduct->next = NULL;

        if (head == NULL) {
            head = newProduct;
            tail = newProduct;
        } else {
            tail->next = newProduct;
            newProduct->prev = tail;
            tail = newProduct;
        }
    }

```

```
40      void removeProduct(string nama_produk) {
41          Product* current = head;
42          while (current != NULL) {
43              if (current->nama_produk == nama_produk) {
44                  if (current == head) {
45                      head = head->next;
46                      if (head != NULL) {
47                          head->prev = NULL;
48                      }
49                  } else if (current == tail) {
50                      tail = tail->prev;
51                      tail->next = NULL;
52                  } else {
53                      current->prev->next = current->next;
54                      current->next->prev = current->prev;
55                  }
56                  delete current;
57                  return;
58              }
59              current = current->next;
60          }
61          cout << "Produk tidak ditemukan." << endl;
62      }
63
64      void updateProduct(string old_nama_produk, string new_nama_produk, int new_harga) {
65          Product* current = head;
66          while (current != NULL) {
67              if (current->nama_produk == old_nama_produk) {
68                  current->nama_produk = new_nama_produk;
69                  current->harga = new_harga;
70                  return;
71              }
72              current = current->next;
73          }
74          cout << "Produk tidak ditemukan." << endl;
75      }
76
```

```
77      void displayProducts() {
78          cout << "Nama Produk\tHarga" << endl;
79          cout << "======================" << endl;
80          Product* current = head;
81          while (current != NULL) {
82              cout << setw(12) << left << current->nama_produk << "\t" << current->harga << endl;
83              current = current->next;
84          }
85          cout << endl;
86      }
87
88      void addProductBetween(string prevProductName, string newProductName, int newPrice) {
89          Product* current = head;
90          while (current != NULL) {
91              if (current->nama_produk == prevProductName) {
92                  Product* newProduct = new Product;
93                  newProduct->nama_produk = newProductName;
94                  newProduct->harga = newPrice;
95                  newProduct->prev = current;
96                  newProduct->next = current->next;
97                  if (current->next != NULL) {
98                      current->next->prev = newProduct;
99                  }
100                 current->next = newProduct;
101                 return;
102             }
103             current = current->next;
104         }
105         cout << "Produk sebelumnya tidak ditemukan." << endl;
106     }
107
```

```
108     void removeAtPosition(int position) {
109         if (head == NULL) {
110             cout << "List kosong." << endl;
111             return;
112         }
113 
114         Product* current = head;
115         int count = 1;
116 
117         while (current != NULL && count != position) {
118             current = current->next;
119             count++;
120         }
121 
122         if (current == NULL) {
123             cout << "Posisi tidak valid." << endl;
124             return;
125         }
126 
127         if (current == head) {
128             head = head->next;
129             if (head != NULL) {
130                 head->prev = NULL;
131             }
132             delete current;
133         } else if (current == tail) {
134             tail = tail->prev;
135             tail->next = NULL;
136             delete current;
137         } else {
138             current->prev->next = current->next;
139             current->next->prev = current->prev;
140             delete current;
141         }
142     }
143 
144     void removeAll() {
145         Product* current = head;
146         while (current != NULL) {
147             Product* temp = current;
148             current = current->next;
149             delete temp;
150         }
151         head = NULL;
152         tail = NULL;
153     }
154 };
```

```cpp
int main() {
    DoubleLinkedList productList;

    productList.addProduct("Originote", 60000);
    productList.addProduct("Somethinc", 150000);
    productList.addProduct("Skintific", 100000);
    productList.addProduct("Wardah", 50000);
    productList.addProduct("Hanasui", 30000);

    int choice;
    string prevProductName, newProductName;
    int newPrice;

    do {
        cout << "\nToko 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;
        cout << "Pilih menu: ";
        cin >> choice;

        switch(choice) {
            case 1:
                cout << "Masukkan Nama Produk: ";
                cin >> newProductName;
                cout << "Masukkan Harga: ";
                cin >> newPrice;
                productList.addProduct(newProductName, newPrice);
                break;
            case 2:
                cout << "Masukkan Nama Produk yang Ingin Dihapus: ";
                cin >> newProductName;
                productList.removeProduct(newProductName);
                break;
            case 3:
                cout << "Masukkan Nama Produk yang Ingin Diupdate: ";
                cin >> prevProductName;
                cout << "Masukkan Nama Produk Baru: ";
                cin >> newProductName;
                cout << "Masukkan Harga Baru: ";
                cin >> newPrice;
                productList.updateProduct(prevProductName, newProductName, newPrice);
                break;
            case 4:
                cout << "Masukkan Nama Produk Sebelumnya: ";
                cin >> prevProductName;
                cout << "Masukkan Nama Produk Baru: ";
                cin >> newProductName;
                cout << "Masukkan Harga Baru: ";
                cin >> newPrice;
                productList.addProductBetween(prevProductName, newProductName, newPrice);
                break;
```

```
            case 5:
            {
                string productName;
                cout << "Masukkan Nama Produk yang Ingin Dihapus: ";
                cin >> productName;
                productList.removeProduct(productName);
                break;
            }
            case 6:
                productList.removeAll();
                cout << "Semua data telah dihapus." << endl;
                break;
            case 7:
                cout << "Data Produk:" << endl;
                productList.displayProducts();
                break;
            case 8:
                cout << "Terima kasih!" << endl;
                break;
            default:
                cout << "Pilihan tidak valid. Silakan pilih lagi." << endl;
        }
    } while (choice != 8);

    return 0;
}
```

Screenshot 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
Data Produk:
Nama Produk     Harga
======================
Originote       60000
Somethinc       150000
Skintific       100000
Wardah          50000
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: 4
Masukkan Nama Produk Sebelumnya: Somethinc
Masukkan Nama Produk Baru: Azarine
Masukkan Harga Baru: 65000
```

```
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: 5
Masukkan Nama Produk yang Ingin 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: 3
Masukkan Nama Produk yang Ingin Diupdate: Hanasui
Masukkan Nama Produk Baru: Cleora
Masukkan Harga Baru: 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
Data Produk:
Nama Produk     Harga
======================
Originote       60000
Somethinc       150000
Azarine         65000
Skintific       100000
Cleora          55000
```

Deskripsi program

Program ini merupakan implementasi dari sebuah sistem manajemen produk menggunakan struktur data Double Linked List. Program ini memungkinkan pengguna untuk melakukan beberapa operasi, termasuk

menambah produk baru, menghapus produk berdasarkan nama, memperbarui nama dan harga produk, menambahkan produk di antara dua produk tertentu, menghapus produk berdasarkan posisi, menghapus semua produk, dan menampilkan semua produk yang tersedia. Pengguna dapat memilih operasi yang diinginkan melalui menu yang ditampilkan secara iteratif hingga pengguna memilih untuk keluar dari program. Program ini memberikan kemudahan dalam manajemen produk bagi pengguna, baik untuk menambahkan, menghapus, atau memperbarui informasi produk yang tersedia.

BAB IV

KESIMPULAN

Single linked list adalah struktur data linier yang terdiri dari sejumlah node yang saling terhubung melalui referensi atau pointer. Setiap simpul dalam single linked list berisi data dan pointer yang menunjukkan ke simpul berikutnya. Perintah pada node dan single list berbeda. Pada node terdapat data dan next. Sedangkan pada single list terdapat length (panjang), head, add (Value), searchNodeAt (position), remove (position).

Double linked list adalah tipe khusus dari linked list di mana setiap simpul mengandung pointer ke simpul sebelumnya dan pointer ke simpul berikutnya. Ada dua jenis perintah, node dan double list. Pada node ada data, next, dan previous. Pada double list terdapat length (panjang), head, tail, add (Value), searchNodeAt (position), remove (position).

Double Linked List dan Single Linked List memiliki kegunaan yang berbeda tergantung pada kebutuhan aplikasi. DLL sering digunakan dalam navigasi web, manajemen cache, dan fitur undo/redo. SLL lebih sederhana dan efisien dalam hal penggunaan memori.

DAFTAR PUSTAKA

Asisten Praktikum. 2024. MODUL 3: SINGLE AND DOUBLE LINKED LIST.

Cho. S. Kim. 2022. Data Structures With JavaScript: Singly-Linked List and Doubly-Linked List. Envato Tuts+.