LAPORAN PRAKTIKUM

MODUL 3
Single dan Double Linked List

Disusun Oleh:
Riyon Aryono **: 2211102241**

Dosen

Muhammad Afrizal Amrustian

PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM
PURWOKERTO

2023

BAB I
TUJUAN PRAKTIKUM

A. Tujuan Praktikum

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

BAB II
DASAR TEORI

1. 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. Untuk menghubungkan satu node dengan node yang lainnya Linked List menggunakan pointer sebagai penunjuk node selanjutnya. Node sendiri merupakan sebuah struct yang terdiri dari beberapa field, minimal ada 2 buah field yaitu field untuk isi dari struct datanya sendiri, dan 1 field arbitrary bertipe pointer sebagai penunjuk node selanjutnya.

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. Untuk menghubungkan satu node dengan node yang lainnya Linked List menggunakan pointer sebagai penunjuk node selanjutnya. Node sendiri merupakan sebuah struct yang terdiri dari beberapa field, minimal ada 2 buah field yaitu field untuk isi dari struct datanya sendiri, dan 1 field arbitrary bertipe pointer sebagai penunjuk node selanjutnya.

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.

2. Double Linked List

Dalam pembahasan artikel sebelumnya telah diperkenalkan Single Linked List, yakni linked list dengan sebuah pointer penghubung. Dalam artikel ini, dibahas pula varian linked list dengan 2 pointer penunjuk, yakni Doubly linked list yang memiliki pointer penunjuk 2 arah, yakni ke arah node sebelum (previous/prev) dan node sesudah (next).

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
LATIHAN & TUGAS

1. Guided

- Demo Single Linked List

Source Code

```
#include <iostream>

using namespace std;

// Program Single Linked List Non-Circular
// Deklrasi Struct Node

struct Node{
    int data;
    Node *next;
};

    Node *head;
    Node *tail;

void init(){
    head = NULL;
    tail = NULL;
}

// pengecekan
bool isEmpty(){
    if(head == NULL){
        return true;
    }else{
        return false;
    }
}

// Tambah Depan
void insertDepan(int nilai){
    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;
    }
}

void insertBelakang(int nilai){
    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;
    }
}

int hitungList(){
    Node *hitung;
    hitung = head;
    int jumlah = 0;

    while(hitung != NULL){
        jumlah++;
        hitung = hitung->next;
    }

    return jumlah;
}


void insertTengah(int data, int posisi){
    if(posisi < 1 ||  posisi > hitungList()){
        cout << "Posisi diluar jangkauan" << endl;
    }else if(posisi == 1){
        cout << "Posisi bukan posisis tengah" << endl;
    }else{
        Node *baru, *bantu;
        baru = new Node();
        baru->data = data;

        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;
    }
}

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;
    }
```

```
}

void hapusTengah(int posisi){
   Node *hapus, *bantu, *bantu2;
   if(posisi < 1 || posisi > hitungList()){
      cout << "Posisi diluar jangkauan" <<endl;
   }else if(posisi == 1){
      cout << "Posisi bukan posisis tengah";
   }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;
   }
}

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

void ubahBelakang(int data){
   if(isEmpty() == false){
      tail->data = data;
   }else{
      cout << "List masih kosong!" << endl;
   }
}

void ubahTengah(int data, int posisi){
   Node *bantu;
```

```
    if(isEmpty() == false){
        if(posisi < 1 || posisi > hitungList()){
            cout << "Posisi diluar jangkuan" << endl;
        }else if(posisi == 1){
            cout << "Posisi bukan ditengah" <<endl;
        }else{
            bantu = head;
            int nomor = 1;
            while(nomor < posisi){
                bantu = bantu->next;
                nomor++;
            }
            bantu->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 dihapus!"<< endl;
}


void tampil(){
    Node *bantu;
    bantu = head;

    if(isEmpty() == false){
        while(bantu != NULL){
            cout << bantu->data << " "<< ends;
            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();
}
```

Output

Penjelesan

Pada Program diatas kita mengimplementasi program single linked list non-circular. Kita juga mengimplementasi beberapa hal seperti menambahkan data didepan tengah dan belakang lalu juga ada ubah depan tengah dan belakang dan yang terakhir kita juga mengimplementasi kan hapus depan tengah dan belakang.

- Demo Double Linked List

  Source Code

```
#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

```
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: 6
```

Penjelesan

Kode program diatas adalah contoh Impelementasi double Linked list menggunakan Class. Prinsip kerja nya sama seperti single Linked list yang membedakan double linked list dengan single linked list adalah double linked list memiliki 2 pointer yaitu next dan prev

2. Unguided

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

  Source Code

```
#include <iostream>
#include <string>
#include <iomanip>

using namespace std;

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

Node *head, *tail;

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

bool isEmpty(){
   if(head == NULL){
      return true;
   }else{
      return false;
   }
}

// tambah data :igor 20 -> add first
void insertFirst(string nama, int umur){
   Node *baru = new Node;
   baru->nama = nama;
   baru->umur = umur;
   baru->next = NULL;

   // cek isEmpty?
   if(isEmpty() == true){
      head = tail = baru;
      tail->next  = NULL;
   }else{
      baru->next = head;
      head       = baru;
   }
}

// Tambah data yang lain nya -> Insert back
```

```
void insertBack(string nama, int umur){
    Node *baru = new Node;
    baru->nama = nama;
    baru->umur = umur;
    baru->next = NULL;

    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }else{
        tail->next = baru;
        tail = baru;
    }

}

int countList(){
    Node *count;
    count = head;
    int total = 0;

    while(count != NULL){
        total++;
        count = count->next;
    }

    return total;
}

// Add Data antara carol dan ann : Futuba 18 -> insert middle
void insertMiddle(string nama, int umur, int posisi){
    if(posisi < 1 || posisi > countList()){
        cout << "Posisi diluar jangkauan" <<endl;
    }else if(posisi == 1){
        cout << "Posisi bukan posisi tengah" <<endl;
    }else{
        Node *baru, *bantu;
        baru = new Node();
        baru->nama = nama;
        baru->umur = umur;

        bantu = head;
        int number = 1;

        while(number < posisi - 1){
```

```
            bantu = bantu->next;
            number++;
        }
        baru->next  = bantu->next;
        bantu->next = baru;
    }
}

// ubah carol jadi : reyn 18 -> change middle
void changeMiddle(string nama, int umur, int posisi){
    Node *bantu;

    if(isEmpty() == false){
        if(posisi < 1 || posisi > countList()){
            cout << "Posisi diluar jangkauan" << endl;
        }else if(posisi == 1){
            cout << "Posisi bukan posisi tengah" <<endl;
        }else{
            bantu = head;
            int num = 1;

            while (num < posisi)
            {
                bantu = bantu->next;
                num++;
            }
            bantu->nama = nama;
            bantu->umur = umur;
        }
    }else{
        cout << "List Kamu masih kosong nih, coba tambahin data dulu ya"
   <<endl;
    }
}

// Hapus Data Akechi -> delete middle
void deleteMiddle(int posisi){
    Node *hapus, *bantu, *bantu2;

    if(posisi < 1 || posisi > countList()){
        cout << "Posisi di luar jangakuan" <<endl;
    }else if(posisi == 1){
        cout << "Posisi bukan posisi tengah" <<endl;
    }else{
        int num = 1;
        bantu   = head;
```

```
        while(num <= posisi){
            if(num == posisi-1){
                bantu2 = bantu;
            }

            if(num == posisi){
                hapus  = bantu;
            }

            bantu = bantu->next;
            num++;
        }
        bantu2->next = bantu;
        delete hapus;
    }
}


// tampil seluruh data
void showList(){
    Node *bantu;
    bantu = head;

    if(isEmpty() == false){
        cout << left << setw(7) << "[Nama_anda]" << right << setw(7) <<
   "[Usia_anda]"<<endl;
        while(bantu != NULL){
            cout << left  << setw(7) << bantu->nama
                 << right << setw(7) << bantu->umur
                 << endl;
            bantu = bantu->next;
        }
        cout <<endl;
    }else{
        cout << "List kamu masih kosong nih, coba inputin data dulu deh"
   <<endl;
    }
}

int main(){
    init();
    insertFirst("Budi", 19);
    insertBack("Carol",20);
    insertBack("Ann",18);
    insertBack("Yusuke",19);
```

```
    insertBack("Akechi",20);
    insertBack("Hoshino",18);
    insertBack("Karin",18);
    showList();
    deleteMiddle(5);
    insertMiddle("Futaba",18, 3);
    insertFirst("Igor",20);
    changeMiddle("Reyn", 18, 3);
    showList();
}
```

Output

```
[Nama_anda][Usia_anda]
Budi        19
Carol       20
Ann         18
Yusuke      19
Akechi      20
Hoshino     18
Karin       18

[Nama_anda][Usia_anda]
Igor        20
Budi        19
Reyn        18
Futaba      18
Ann         18
Yusuke      19
Hoshino     18
Karin       18
```

Penjelesan

Kode Program diatas menggunakan Single List. Operasi yang pertama menghapus data akechi menggunakan teknik hapus tengah cara kerjanya cukup mudah yang pertama kita menentukan posisi data yang akan dihapus lalu nanti nya linked list akan menghapus data tersebut. Operasi yang kedua menambahkan data futaba sama seperti sebelum nya menggunakan tekin insert tengah dan cara kerja nya juga sama dengan hapus tengah. Operasi yang ketiga menambahkan data Igor diawal menggunakan teknik tambah awal. Lalu operasi terkhir mengubah data Carol menjadi Reyn menggunakan teknik ubah tengah sama seperti teknik hapus tengah dan tambah tengah.

- Modifikasi Guided Double Linked List ditambahkan fungsi menambahkan data, menghapus, dan update di tengah / di urutan tertentu yang diminta. Selain itu, buatlah agar tampilannya menampilkan Nama produk dan harga.

  Source Code

```
#include <iostream>
#include <iomanip>

using namespace std;
class Node
{
public:
    string name;
    int price;

    Node *next;
    Node *prev;

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


    void push(string name, int price)
    {
        Node* temp = new Node();
        temp->name = name;
        temp->price = price;
        temp->next = nullptr;
        if (head == nullptr) {
            temp->prev = nullptr;
            head = temp;
            tail = temp;
            return;
```

```
    }
    temp->prev = tail;
    tail->next = temp;
    tail = temp;
}

int countList(){
    Node *count;
    count = head;
    int total = 0;

    while(count != nullptr){
        total++;
        count = count->next;
    }

    return total;
}

// Tambahkan produk Azarine dengan harga 65000 diantara Somethinc dan Skintific -> insert middle

void insertByPos(string name, int price, int pos){
    if(pos < 0 || pos > countList()){
        cout << "Cek Ulang Posisi" <<endl;
        return;
    }

   Node* newNode = new Node();
   newNode->name = name;
   newNode->price = price;

    if (!head) {
        head = tail = newNode;
    } else if (pos == 0) {
        newNode->next = head;
        head->prev = newNode;
        head = newNode;
    } else if (pos == countList()) {
        newNode->prev = tail;
        tail->next = newNode;
        tail = newNode;
    } else {
        Node* current = head;
        for (int i = 1; i < pos - 1; i++) {
            current = current->next;
```

```
        }
        newNode->prev = current;
        newNode->next = current->next;
        current->next->prev = newNode;
        current->next = newNode;
    }
}

// Update produk Hanasui menjadi Cleora dengan harga 55.000
bool update(string name, int price, int posisi){
    Node* current = head;
    int counter = 1;
    while(current != nullptr) {
        if(counter == posisi) {
            current->name = name;
            current->price = price;
            break;
        }
        current = current->next;
        counter++;
    }
}

 void pop(){
    if (head == nullptr){
        return;
    }

    Node *temp = head;
    head = head->next;

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

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

```
        delete temp;
    }
    head = nullptr;
    tail = nullptr;
}

// Hapus produk wardah

void deleteByPos(int pos){
    if(head == nullptr){
        return;
    }

    if(pos == 1){
        Node* temp = head;
        head = head->next;

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

        delete temp;
        return;
    }

    Node* curr = head;
    int count = 1;

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

    if (curr == nullptr){
        return;
    }

    if(curr == tail){
        tail = tail->prev;
        tail->next = nullptr;
        delete curr;
        return;
    }
```

```
        curr->prev->next = curr->next;
        curr->next->prev = curr->prev;
        delete curr;
    }

    // Tampilkan Seluruh Data

    void display(){
        Node *current = head;

        cout << left << setw(15) << "Nama Produk" << right << setw(18) <<
   "Harga" << endl;
        while (current != nullptr){
            cout << left << setw(15) << current->name
                << right << setw(18) << current->price
                << endl;
            current = current->next;
        }
        cout << endl;
    }
};

int main(){
    DoublyLinkedList list;
    while (true){
        cout << "Toko Skincare Purwokerto" << endl;
        cout << "1. Tambah data" << endl;
        cout << "2. Hapus data" << endl;
        cout << "3. Update data" << endl;
        cout << "4. Tambah Data Urutan Tertentu" << endl;
        cout << "5. Hapus Data Urutan Tertentu" << endl;
        cout << "6. Hapus Seluruh Data" << endl;
        cout << "7. Tampilkan Data" << endl;
        cout << "8. Exit" << endl;

        int choice;
        cout << "Enter your choice: ";
        cin >> choice;

        cout << endl;
        switch (choice) {
        case 1:
        {
            string name;
            int price;
            cout << "Masukkan Nama Produk: ";
```

```
            cin >> name;

            cout << "Masukkan Harga: ";
            cin >> price;
            list.push(name, price);
            break;
        }
        case 2:
        {
            list.pop();
            break;
        }
        case 3:
        {
            string  newName;
            int newPrice,posisi;
            cout << "Masukan nama produk baru: ";
            cin  >> newName;
            cout << "Masukan harga produk baru: ";
            cin  >> newPrice;
            cout << "Masukan Posisi: ";
            cin  >> posisi;
            list.update(newName, newPrice, posisi);
            break;
        }
        case 4:
        {
            int pos,price;
            string name;
            cout << "Masukan Nama Produk: ";
            cin  >> name;
            cout << "Masukan harga Produk: ";
            cin  >> price;
            cout << "Masukan urutan posisi: ";
            cin  >> pos;
            list.insertByPos(name,price, pos);
            break;
        }
        case 5:
        {
            int pos;
            cout << "Masukan posisi yang ingin dihapus: ";
            cin  >> pos;
            list.deleteByPos(pos);
            break;
        }
```

```
        case 6:
        {
            list.deleteAll();
           break;
        }
        case 7:
        {
           list.display();
           break;
        }
        case 8:
        {
           return 0;
        }
        default:
        {
           cout << "Invalid choice" << endl;
           return 0;
        }
        }
    }
    return 0;
}
```

Output

Case 1:

```
Masukan Nama Produk: Azarine
Masukan harga Produk: 65000
Masukan urutan posisi: 3
```

```
Nama Produk              Harga
Originote                60000
Somethinc               150000
Azarine                  65000
Skintific                60000
Wardah                   50000
Hanasui                  30000
```

Case 2:

```
Masukan posisi yang ingin dihapus: 5
```

```
Nama Produk              Harga
Originote                60000
Somethinc               150000
Azarine                  65000
Skintific                60000
Hanasui                  30000
```

Case 3:

```
Masukan nama produk baru: Cleora
Masukan harga produk baru: 55000
Masukan Posisi: 5
```

```
Nama Produk                Harga
Originote                   60000
Somethinc                  150000
Azarine                     65000
Skintific                   60000
Cleora                      55000
```

Case 4:

```
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
Enter your choice: 7

Nama Produk                Harga
Originote                   60000
Somethinc                  150000
Azarine                     65000
Skintific                   60000
Cleora                      55000
```

Penjelesan

Kode di atas merupakan implementasi dari double linked list. Selanjutnya, kita mendefinisikan kelas DoubleLinkedList untuk merepresentasikan DoubleLinkedList itu sendiri. Kelas DoubleLinkedList ini memiliki dua atribut yaitu head (penunjuk ke node pertama) dan tail (penunjuk ke node terakhir). Kemudian kita juga mendefinisikan konstruktor di kelas DoubleLinkedList yang digunakan untuk menginisialisasi nilai atribut tersebut. lalu kita buat fungsi tambah, tambah tengah, update tengah, delete , hapus tengah, hapus seluruh data.

BAB IV
KESIMPULAN

1. Single Linked List merupakan struktur data yang terdiri dari kumpulan node yang saling terhubung secara linear dengan menggunakan pointer next. Setiap node pada Single Linked List terdiri dari sebuah data dan sebuah pointer next yang menunjuk ke node selanjutnya. Keuntungan dari Single Linked List adalah mudah untuk ditambahkan atau dihapus pada posisi awal dan akhir, namun sulit untuk mengakses atau menghapus elemen di tengah.
2. Double Linked List merupakan struktur data yang terdiri dari kumpulan node yang saling terhubung secara linear dengan menggunakan pointer prev dan pointer next. Setiap node pada Double Linked List terdiri dari sebuah data, sebuah pointer prev yang menunjuk ke node sebelumnya, dan sebuah pointer next yang menunjuk ke node selanjutnya. Keuntungan dari Double Linked List adalah dapat dengan mudah mengakses, menambahkan, atau menghapus elemen pada posisi awal, akhir, atau di tengah. Namun, Double Linked List memerlukan alokasi memori lebih banyak karena setiap node harus menyimpan dua pointer, dan kompleksitas algoritma untuk mengakses atau menghapus elemen di tengah lebih tinggi dibandingkan dengan Single Linked List.