LAPORAN PRAKTIKUM
Modul 6
“Double Linked List (Bagian Pertama)”

Disusun Oleh:

Muhammad Shafiq Rasuna - 2311104043

Kelas :

S1SE-07-02

Dosen :

Wahyu Andi Saputra, S.Pd, M.Eng

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

1. Tujuan

1. Memahami konsep modul linked list.
2. Mengaplikasikan konsep double linked list dengan menggunakan pointer dan dengan bahasa C.

2. Landasan Teori

2.1. Double Linked List

Double linked list adalah tipe linked list di mana setiap elemennya memiliki dua penunjuk atau successor, yaitu penunjuk ke elemen sebelumnya (prev) dan penunjuk ke elemen berikutnya (next). Dalam struktur ini, terdapat dua successor utama, yaitu first dan last. First adalah pointer pada list yang menunjuk ke elemen pertama, sedangkan last adalah pointer yang menunjuk ke elemen terakhir dalam list. Setiap elemen dalam double linked list memiliki pointer next, yang berfungsi sebagai penunjuk ke elemen setelahnya, serta pointer prev, yang menghubungkannya ke elemen sebelumnya.

Berikut adalah bagian – bagian yang ada pada double linked list:

1. Insert

   Operasi insert menambahkan node baru ke dalam daftar. Jenis-jenis penyisipan meliputi:

   A. Insert first: Menyisipkan node baru di awal double linked list. Operasi ini biasanya mengubah pointer head menjadi node baru dan mengatur pointer prev pada node lama untuk menunjuk ke node baru.
   B. Insert Last: Menambahkan node baru di akhir double linked list. Pointer next dari node terakhir sebelumnya diatur untuk menunjuk ke node baru, dan pointer prev dari node baru menunjuk ke node sebelumnya.
   C. Insert After: Menyisipkan node setelah node tertentu. Pointer node baru diatur untuk menunjuk ke node berikutnya dari node yang diberikan, dan pointer dari node setelahnya diatur untuk menunjuk kembali ke node baru.
   D. Insert Before: Menyisipkan node sebelum node tertentu, serupa dengan "Insert After" tetapi dalam posisi sebelumnya.

2. Delete

   Operasi delete menghapus node dari daftar. Jenis-jenis penghapusan meliputi:

   A. Delete First: Menghapus node pertama pada double linked list, dan mengatur ulang pointer head ke node berikutnya.
   B. Delete Last: Menghapus node terakhir, mengatur pointer dari node sebelumnya agar tidak menunjuk lagi ke node terakhir.
   C. Delete After: Menghapus node setelah node tertentu. Pointer node diatur ulang untuk melewati node yang dihapus.

D. Delete Before: Menghapus node sebelum node tertentu, dengan pengaturan ulang pointer dari node tertentu.

E. Update, View, dan Searching

Update, operasi update mengubah data yang tersimpan di suatu node. Proses update melibatkan pencarian node tertentu, lalu mengganti data pada node tersebut dengan data baru. View, menampilkan isi dari double linked list. Operasi ini bisa dilakukan baik dari node head (untuk menampilkan data dari awal ke akhir) atau dari node terakhir (untuk menampilkan data dari akhir ke awal). Seaching, operasi searching bertujuan menemukan node yang mengandung data tertentu. Double linked list dapat melakukan pencarian maju (dari head ke akhir) atau mundur (dari akhir ke head) sesuai kebutuhan.

3. Guided

3.1. Guided Modul 6

Program ini merupakan implementasi **Doubly Linked List (DLL)** dalam C++, yang mencakup operasi dasar seperti menambah, menghapus, memperbarui, menghapus semua data, dan menampilkan data. Kelas Node mendefinisikan elemen dalam list dengan atribut data, prev, dan next, sementara kelas DoublyLinkedList mengelola operasi pada list dengan pointer head dan tail. Antarmuka berbasis teks memungkinkan pengguna memilih operasi yang ingin dilakukan melalui menu interaktif.

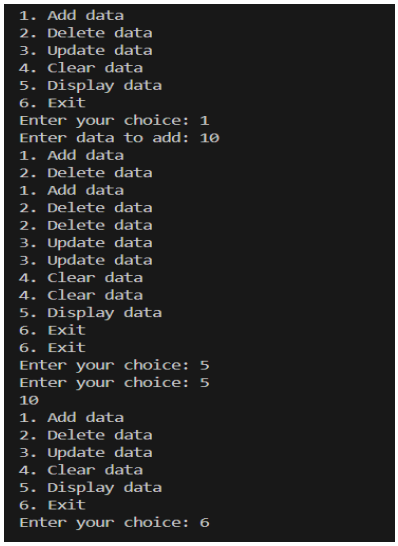
Keunggulan dari program ini adalah kesederhanaannya dan antarmuka interaktif yang memudahkan pengguna berinteraksi dengan list. Namun, ada beberapa area yang bisa diperbaiki, seperti menambahkan validasi input yang lebih baik, pengelolaan memori yang lebih hati-hati, dan penanganan kesalahan yang lebih jelas. Program juga bisa lebih fleksibel dengan menambahkan fitur untuk menghapus elemen dari posisi tengah atau akhir list. Secara keseluruhan, program ini sudah solid, tetapi beberapa peningkatan dapat membuatnya lebih efisien dan robust.

Kode Program :

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

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

class DoublyLinkedList {
public:
    Node* head;
    Node* tail;

    // Constructor untuk inisialisasi head dan tail
    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    // Fungsi untuk menambahkan elemen di depan list
    void insert(int data) {
        Node* newNode = new Node;
        newNode->data = data;
        newNode->prev = nullptr;
        newNode->next = head;

        if (head != nullptr) {
            head->prev = newNode;
        } else {
            tail = newNode; // Jika list kosong, tail juga mengarah ke node baru
        }
        head = newNode;
    }

    // Fungsi untuk menghapus elemen dari depan list
    void deleteNode() {
        if (head == nullptr) {
            return; // Jika list kosong
        }
        Node* temp = head;
        head = head->next;
        if (head != nullptr) {
            head->prev = nullptr;
        } else {
            tail = nullptr; // Jika hanya satu elemen di list
        }
        delete temp; // Hapus elemen
    }

```

```cpp
    // Fungsi untuk mengupdate data di list
    bool update(int oldData, int newData) {
        Node* current = head;
        while (current != nullptr) {
            if (current->data == oldData) {
                current->data = newData;
                return true; // Jika data ditemukan dan diupdate
            }
            current = current->next;
        }
        return false; // Jika data tidak ditemukan
    }

    // Fungsi untuk menghapus semua elemen di list
    void deleteAll() {
        Node* current = head;
        while (current != nullptr) {
            Node* temp = current;
            current = current->next;
            delete temp;
        }
        head = nullptr;
        tail = nullptr;
    }

    // Fungsi untuk menampilkan semua elemen di list
    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.insert(data);
                break;
            }
            case 2: {
                list.deleteNode();
                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 dari Kode Program :



4. Unguided

4.1. Kode Program:

```cpp
#include <iostream>
#include <vector>
#include <string>

using namespace std;

struct Kendaraan {
    string nomorPolisi;
    string warna;
    int tahun;
};

int main() {
    vector<Kendaraan> dataKendaraan;

    while (true) {
        Kendaraan kendaraanBaru;
        cout << "Masukkan nomor polisi: ";
        cin >> kendaraanBaru.nomorPolisi;

        // Cek duplikat
        bool adaDuplikat = false;
        for (const Kendaraan& kendaraan : dataKendaraan) {
            if (kendaraan.nomorPolisi == kendaraanBaru.nomorPolisi) {
                cout << "Nomor polisi sudah terdaftar." << endl;
                adaDuplikat = true;
                break;
            }
        }

        if (!adaDuplikat) {
            cout << "Masukkan warna kendaraan: ";
            cin >> kendaraanBaru.warna;
            cout << "Masukkan tahun kendaraan: ";
            cin >> kendaraanBaru.tahun;

            dataKendaraan.push_back(kendaraanBaru);
        }

        cout << endl;

        // Tampilkan opsi untuk melanjutkan atau tidak
        char pilihan;
        cout << "Ingin menambahkan data lagi? (y/n): ";
        cin >> pilihan;
        if (pilihan != 'y' && pilihan != 'Y') {
            break;
        }
    }

    // Tampilkan semua data kendaraan
    cout << "DATA LIST 1" << endl;
    for (const Kendaraan& kendaraan : dataKendaraan) {
        cout << "Nomor Polisi: " << ke vector<Kendaraan> main::dataKendaraan
        cout << "Warna       : " << kendaraan.warna << endl;
        cout << "Tahun       : " << kendaraan.tahun << endl;
    }

    return 0;
}
```

Output dari Kode Program:

```
Masukkan nomor polisi: D001
Masukkan warna kendaraan: hitam
Masukkan tahun kendaraan: 90

Ingin menambahkan data lagi? (y/n): y
Masukkan nomor polisi:  D003
Masukkan warna kendaraan: putih
Masukkan tahun kendaraan: 70

Ingin menambahkan data lagi? (y/n): y
Masukkan nomor polisi: D002
Masukkan warna kendaraan: kuning
Masukkan tahun kendaraan: 80

Ingin menambahkan data lagi? (y/n): y
Masukkan nomor polisi: R45UNA
Masukkan warna kendaraan: merah
Masukkan tahun kendaraan: 69

Ingin menambahkan data lagi? (y/n): n
DATA LIST 1
Nomor Polisi: D001
Warna       : hitam
Tahun       : 90
Nomor Polisi: D003
Warna       : putih
Tahun       : 70
Nomor Polisi: D002
Warna       : kuning
Tahun       : 80
Nomor Polisi: R45UNA
Warna       : merah
Tahun       : 69

Process returned 0 (0x0)   execution time : 232.876 s
Press any key to continue.
```

4.2. Kode Program:

```cpp
#include <iostream>
#include <vector>
#include <string>

using namespace std;

struct Kendaraan {
    string nomorPolisi;
    string warna;
    int tahun;
};
int main() {
    vector<Kendaraan> dataKendaraan;

    dataKendaraan.push_back({"D001", "hitam", 1998});
    dataKendaraan.push_back({"B1234", "merah", 2005});
    dataKendaraan.push_back({"D5678", "biru", 2010});

    string nomorPolisiCari;
    cout << "Masukkan Nomor Polisi yang dicari: ";
    cin >> nomorPolisiCari;

    bool ditemukan = false;
    for (const Kendaraan& kendaraan : dataKendaraan) {
        if (kendaraan.nomorPolisi == nomorPolisiCari) {
            cout << "Nomor Polisi: " << kendaraan.nomorPolisi << endl;
            cout << "Warna       : " << kendaraan.warna << endl;
            cout << "Tahun       : " << kendaraan.tahun << endl;
            ditemukan = true;
            break;
        }
    }

    if (!ditemukan) {
        cout << "Kendaraan tidak ditemukan." << endl;
    }

    return 0;
}
```

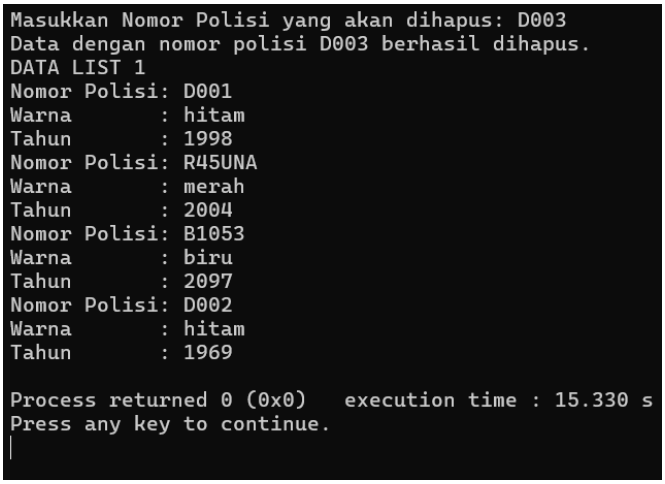
Ouput dari Kode Program:

```
Masukkan Nomor Polisi yang dicari: D001
Nomor Polisi: D001
Warna       : hitam
Tahun       : 1998

Process returned 0 (0x0)   execution time : 47.886 s
Press any key to continue.
```

4.3. Kode Program:

```cpp
#include <iostream>
#include <vector>
#include <string>

using namespace std;

struct Kendaraan {
    string nomorPolisi;
    string warna;
    int tahun;
};

int main() {
    vector<Kendaraan> dataKendaraan;

    // Contoh data kendaraan
    dataKendaraan.push_back({"D001", "hitam", 1998});
    dataKendaraan.push_back({"R45UNA", "merah", 2004});
    dataKendaraan.push_back({"B1053", "biru", 2097});
    dataKendaraan.push_back({"D003", "kuning", 1979});
    dataKendaraan.push_back({"D002", "hitam", 1969});

    string nomorPolisiHapus;
    cout << "Masukkan Nomor Polisi yang akan dihapus: ";
    cin >> nomorPolisiHapus;

    bool ditemukan = false;
    for (auto it = dataKendaraan.begin(); it != dataKendaraan.end(); ++it) {
        if (it->nomorPolisi == nomorPolisiHapus) {
            dataKendaraan.erase(it);
            ditemukan = true;
            break;
        }
    }

    if (ditemukan) {
        cout << "Data dengan nomor polisi " << nomorPolisiHapus << " berhasil dihapus." << endl;
    } else {
        cout << "Kendaraan tidak ditemukan." << endl;
    }

    // Menampilkan data yang tersisa
    cout << "DATA LIST 1" << endl;
    for (const Kendaraan& kendaraan : dataKendaraan) {
        cout << "Nomor Polisi: " << kendaraan.nomorPolisi << endl;
        cout << "Warna       : " << kendaraan.warna << endl;
        cout << "Tahun       : " << kendaraan.tahun << endl;
    }

    return 0;
}
```

Output dari Kode Program:



5. Kesimpulan

Dari laporan praktikum ini, dapat disimpulkan bahwa konsep Double Linked List (DLL) merupakan struktur data yang sangat berguna untuk pengelolaan elemen-elemen dalam daftar yang dapat diakses baik dari awal hingga akhir maupun sebaliknya. Dalam DLL, setiap elemen memiliki dua pointer, yaitu prev yang menunjuk ke elemen sebelumnya dan next yang menunjuk ke elemen berikutnya, yang memungkinkan traversal dalam dua arah. Berbagai operasi seperti penyisipan, penghapusan, pembaruan, pencarian, dan tampilan dapat dilakukan dengan fleksibilitas yang tinggi pada DLL. Dalam implementasinya menggunakan bahasa C, berbagai jenis penyisipan dan penghapusan data dapat dilakukan, seperti penyisipan di awal, akhir, atau setelah/before node tertentu, serta penghapusan node di posisi yang sama. Program praktikum yang dikembangkan menunjukkan penerapan DLL secara interaktif, meskipun terdapat ruang untuk perbaikan seperti validasi input yang lebih baik dan pengelolaan memori yang lebih hati-hati. Secara keseluruhan, praktikum ini membantu memperdalam,pemahaman

Telkom University Purwokerto