LAPORAN PRAKTIKUM

Modul 06

“DOUBLE LINKED LIST (BAGIAN SATU)”

Disusun Oleh:

Marvel Sanjaya Setiawan (2311104053)

SE-07-02

Dosen :

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

PURWOKERTO

2024

1. **Tujuan**
    - Memahami cara kerja struktur data linked list.
    - Membuat program C++ untuk linked list dua arah.
2. **Landasan Teori**

DLL (Bagian Pertama)

Double linked list adalah struktur data linier di mana setiap elemen (node) memiliki dua pointer: prev (menunjuk ke node sebelumnya) dan next (menunjuk ke node selanjutnya). Selain itu, ada dua pointer khusus: first (menunjuk ke node pertama) dan last (menunjuk ke node terakhir).

Operasi Dasar Double Linked List.

Menambahkan Node (Insert).

a. Insert First: Tambahkan node baru di awal list.

    1) Ubah next node baru menjadi first.

    2) Ubah prev node baru menjadi NULL.

    3) Ubah prev dari first menjadi node baru.

    4) Ubah first menjadi node baru.

b. Insert Last: Tambahkan node baru di akhir list.

    1) Ubah prev node baru menjadi last.

    2) Ubah next node baru menjadi NULL.

    3) Ubah next dari last menjadi node baru.

    4) Ubah last menjadi node baru.

c. Insert After: Tambahkan node baru setelah node tertentu.

    1) Ubah prev node baru menjadi node tertentu.

    2) Ubah next node baru menjadi next dari node tertentu.

    3) Ubah prev dari next node tertentu menjadi node baru.

    4) Ubah next dari node tertentu menjadi node baru.

d. Insert Before: Tambahkan node baru sebelum node tertentu.

    1) Ubah next node baru menjadi node tertentu.

    2) Ubah prev node baru menjadi prev dari node tertentu.

    3) Ubah next dari prev node tertentu menjadi node baru.

    4) Ubah prev dari node tertentu menjadi node baru.

Menghapus Node (Delete).

a. Delete First: Hapus node pertama.

    1) Ubah first menjadi next dari first.

    2) Ubah prev dari node baru (yang menjadi first) menjadi NULL.

b. Delete Last: Hapus node terakhir.

    1) Ubah last menjadi prev dari last.

    2) Ubah next dari node baru (yang menjadi last) menjadi NULL.

c. Delete Last: Hapus node terakhir.

    1) Ubah next dari node tertentu menjadi next dari node yang akan dihapus.

    2) Ubah prev dari next node yang akan dihapus menjadi node tertentu.

d. Delete Last: Hapus node terakhir.

1) Ubah prev dari node tertentu menjadi prev dari node yang akan dihapus.

2) Ubah next dari prev node yang akan dihapus menjadi node tertentu.

Operasi Lain

Update: Ubah data pada node tertentu.

View: Tampilkan semua data dalam list.

Searching: Cari node dengan data tertentu.

Telkom University Purwokerto

3. Guided

```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() {
    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
    }

    // 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;
}
```

Telkom University Purwokerto

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

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

Cara kerja program:

1. Program dimulai dengan membuat objek DoublyLinkedList.
2. Pengguna memilih operasi yang ingin dilakukan.
3. Berdasarkan pilihan, program memanggil fungsi yang sesuai dari kelas DoublyLinkedList.
4. Fungsi-fungsi tersebut melakukan operasi manipulasi data pada linked list, seperti menambahkan, menghapus, mengupdate, atau menampilkan data.
5. Program terus berulang hingga pengguna memilih untuk keluar.

4. Unguided

1. Membuat ADT Double Linked List.

```h
#ifndef DOUBLELIST_H
#define DOUBLELIST_H
#include <string>
using namespace std;

struct kendaraan {
    string nopol;
    string warna;
    int thnBuat;
};
typedef kendaraan infotype;
typedef struct ElmList *address;
struct ElmList {
    infotype info;
    address next;
    address prev;
};
struct List {
    address First;
    address Last;
};
void CreateList(List &L);
address alokasi(infotype x);
void dealokasi(address &P);
void printInfo(const List &L);
void insertLast(List &L, address P);

address findElm(const List &L, const infotype &x);

void deleteFirst(List &L, address &P);
void deleteLast(List &L, address &P);
void deleteAfter(address Prec, address &P);

#endif
```

File doublelist.h

Telkom University Purwokerto

```cpp
#include "doublelist.h"
#include <iostream>

void CreateList(List &L) {
    L.First = nullptr;
    L.Last = nullptr;
}

address alokasi(infotype x) {
    address P = new ElmList;
    P->info = x;
    P->next = nullptr;
    P->prev = nullptr;
    return P;
}

void dealokasi(address &P) {
    delete P;
    P = nullptr;
}

void printInfoReverse(const List &L) {
    address P = L.Last;
    int i = 1;
    cout << "DATA LIST 1" << endl << endl;
    while (P != nullptr) {
        cout << "no polisi : " << P->info.nopol << endl;
        cout << "warna     : " << P->info.warna << endl;
        cout << "tahun     : " << P->info.thnBuat << endl;
        P = P->prev;
    }
}

void insertLast(List &L, address P) {
    if (L.First == nullptr) {
        L.First = P;
        L.Last = P;
    } else {
        L.Last->next = P;
        P->prev = L.Last;
        L.Last = P;
    }
}

address findElm(const List &L, const infotype &x) {
    address P = L.First;
    while (P != nullptr) {
        if (P->info.nopol == x.nopol) {
            return P;
        }
        P = P->next;
    }
    return nullptr;
}

void deleteFirst(List &L, address &P) {
    if (L.First != nullptr) {
        P = L.First;
        L.First = L.First->next;
        if (L.First != nullptr) {
            L.First->prev = nullptr;
        } else {
            L.Last = nullptr;
        }
        P->next = nullptr;
        P->prev = nullptr;
    } else {
        cout << "List kosong, tidak ada elemen yang dihapus." << endl;
    }
}

void deleteLast(List &L, address &P) {
    if (L.Last != nullptr) {
        P = L.Last;
        L.Last = L.Last->prev;
        if (L.Last != nullptr) {
            L.Last->next = nullptr;
        } else {
            L.First = nullptr;
        }
        P->next = nullptr;
        P->prev = nullptr;
    } else {
        cout << "List kosong, tidak ada elemen yang dihapus." << endl;
    }
}

void deleteAfter(address Prec, address &P) {
    if (Prec != nullptr && Prec->next != nullptr) {
        P = Prec->next;
        Prec->next = P->next;
        if (P->next != nullptr) {
            P->next->prev = Prec;
        }
        P->next = nullptr;
        P->prev = nullptr;
    } else {
        cout << "Tidak ada elemen yang dapat dihapus setelah Prec." << endl;
    }
}

void deleteData(List &L, string nopol) {
    infotype searchData;
    searchData.nopol = nopol;

    address P = findElm(L, searchData);
    if (P != NULL) {
        if (P == L.First) {
            deleteFirst(L, P);
        } else if (P == L.Last) {
            deleteLast(L, P);
        } else {
            deleteAfter(P->prev, P);
        }
        cout << "Data dengan nomor polisi " << nopol << " berhasil dihapus." << endl;
        dealokasi(P);
    } else {
        cout << "Data tidak ditemukan!" << endl;
    }
}
```

file doublelist.cpp

```cpp
#include "doublelist.h"
#include "doublelist.cpp"
#include <iostream>

using namespace std;

int main() {
    List L;
    CreateList(L);
    infotype data;

    for (int i = 0; i < 4; i++) {
        cout << "Masukkan nomor polisi: ";
        cin >> data.nopol;

        cout << "Masukkan warna kendaraan: ";
        cin >> data.warna;

        cout << "Masukkan tahun kendaraan: ";
        cin >> data.thnBuat;

        address found = findElm(L, data);
        if (found != nullptr) {
            cout << "Nomor polisi sudah terdaftar" << endl << endl;
            continue;  // Skip this iteration and proceed to the next input
        }

        address P = alokasi(data);
        insertLast(L, P);
        cout << endl;
    }

    printInfoReverse(L);
    return 0;

}
```

file main.cpp versi 1

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

Masukkan nomor polisi: D003
Masukkan warna kendaraan: putih
Masukkan tahun kendaraan: 70

Masukkan nomor polisi: D001
Masukkan warna kendaraan: merah
Masukkan tahun kendaraan: 80
Nomor polisi sudah terdaftar

Masukkan nomor polisi: D004
Masukkan warna kendaraan: kuning
Masukkan tahun kendaraan: 90

DATA LIST 1

no polisi : D004
warna     : kuning
tahun     : 90
no polisi : D003
warna     : putih
tahun     : 70
no polisi : D001
warna     : hitam
tahun     : 90
```

Fungsi:

- CreateList: Membuat list kosong.
- alokasi: Mengalokasi memori untuk node baru.
- dealokasi: Membebaskan memori node.
- insertLast: Menambahkan node baru ke akhir list.
- findElm: Mencari node dengan nomor polisi tertentu.
- deleteFirst: Menghapus node pertama.
- deleteLast: Menghapus node terakhir.
- deleteAfter: Menghapus node setelah node tertentu.
- deleteData: Menghapus node dengan nomor polisi tertentu.
- printInfoReverse: Menampilkan data list secara terbalik..

Cara Kerja:

1) Inisialisasi List: List diinisialisasi dalam keadaan kosong.
2) Input Data: Pengguna diminta untuk memasukkan data kendaraan (nomor polisi, warna, tahun pembuatan).
3) Pencarian Duplikat: Sebelum memasukkan data, program memeriksa apakah nomor polisi sudah ada dalam list. Jika sudah, input diabaikan.
4) Alokasi Node: Memori dialokasikan untuk node baru.
5) Penambahan Node: Node baru ditambahkan ke akhir list.
6) Penampilkan Data: Data dalam list ditampilkan secara terbalik.

2. **Mencari Elemen.**

```cpp
#include "doublelist.h"
#include "doublelist.cpp"
#include <iostream>

using namespace std;

int main() {
    List L;
    CreateList(L);
    infotype data;

    infotype searchData;
    cout << "Masukkan nomor polisi yang ingin dicari: ";
    cin >> searchData.nopol;

    address foundElement = findElm(L, searchData);
    cout << endl;
    if (foundElement != nullptr) {
        cout << "Nomor Polisi  : " << foundElement->info.nopol << endl;
        cout << "Warna         : " << foundElement->info.warna << endl;
        cout << "Tahun         : " << foundElement->info.thnBuat << endl;
    } else {
        cout << "Kendaraan dengan nomor polisi " << searchData.nopol << " tidak ditemukan." << endl;
    }

    cout << endl;

    return 0;

}
```

file main.cpp versi 2

```
Masukkan nomor polisi yang ingin dicari: D001

Nomor Polisi  : D001
Warna         : hitam
Tahun         : 90
```

Cara Kerja:

1) Inisialisasi List: List diinisialisasi dalam keadaan kosong.
2) Input Data: Pengguna diminta untuk memasukkan data kendaraan (nomor polisi, warna, tahun pembuatan).
3) Pencarian Data: Pengguna dapat mencari data berdasarkan nomor polisi.
4) Penampilkan Data: Menampilkan Data dalam list yang tadi diinputkan untuk dicari.

3. **Menghapus Elemen.**

```
#include "doublelist.h"
#include "doublelist.cpp"
#include <iostream>

using namespace std;

int main() {
    List L;
    CreateList(L);
    infotype data;

    cout << "\nMasukkan Nomor Polisi yang akan dihapus: ";
    string nopolHapus;
    cin >> nopolHapus;

    deleteData(L, nopolHapus);

    cout << endl;

    printInfoReverse(L);

    return 0;

}
```

file main.cpp versi 3

```
Masukkan Nomor Polisi yang akan dihapus: D003
Data dengan nomor polisi D003 berhasil dihapus.

DATA LIST 1

no polisi : D004
warna     : kuning
tahun     : 90
no polisi : D001
warna     : hitam
tahun     : 90
```

Cara Kerja:

1) Inisialisasi List: List diinisialisasi dalam keadaan kosong.
2) Input Data: Pengguna diminta untuk memasukkan data kendaraan (nomor polisi, warna, tahun pembuatan).
3) Penghapusan Data: Pengguna dapat menghapus data berdasarkan nomor polisi.
4) Penampilkan Data: Data dalam list ditampilkan secara terbalik.

5. Kesimpulan

Praktikum ini berhasil mengimplementasikan struktur data double linked list dalam bahasa C++. Melalui serangkaian fungsi yang telah dibuat, seperti insert, delete, dan search, dapat disimpulkan bahwa double linked list memberikan fleksibilitas dalam manipulasi data karena setiap node memiliki pointer ke node sebelumnya dan selanjutnya.