

**LAPORAN PRAKTIKUM
STRUKTUR DATA
Modul 6
“Double Linked List Bagian Pertama”**



**Disusun Oleh:
MUHAMMAD RALFI - 2211104054
SE-07-2**

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

**PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024**

1. Tujuan

- Memahami konsep modul linked list.
- Mahasiswa dapat mengaplikasikannya dengan benar.
- Mengaplikasikan konsep double linked list dengan menggunakan pointer dan dengan bahasa C.

2. Landasan Teori

Double Linked List

Adalah struktur data yang terdiri dari sekumpulan simpul, yang masing-masing berisi nilai dan dua penunjuk, satu menunjuk ke simpul sebelumnya dalam daftar dan satu menunjuk ke simpul berikutnya dalam daftar. Hal ini memungkinkan traversal daftar yang efisien di kedua arah, sehingga cocok untuk aplikasi yang memerlukan penyisipan dan penghapusan yang sering. Double linked list menggunakan dua buah successor utama yang terdapat pada list, yaitu first (successor yang menunjuk elemen pertama) dan last (susesor yang menunjuk elemen terakhir list).

- First : pointer pada list yang menunjuk pada elemen pertama list.
- Last : pointer pada list yang menunjuk pada elemen terakhir list.
- Next : pointer pada elemen sebagai successor yang menunjuk pada elemen didepannya.
- Prev : pointer pada elemen sebagai successor yang menunjuk pada elemen dibelakangnya.

Operasi-operasi dasar yang dapat dilakukan pada Single Linked List meliputi pembuatan list kosong (CreateList), penambahan node baru (Insert), penghapusan node (Delete), dan pencarian node (Search). Operasi CreateList digunakan untuk menginisialisasi list kosong dengan pointer first yang menunjuk ke NULL. Operasi Insert dapat dilakukan di awal list (InsertFirst), di akhir list (InsertLast), atau setelah node tertentu (InsertAfter). Operasi Delete juga dapat dilakukan di berbagai posisi seperti awal, akhir, atau setelah node tertentu

3. Guided

Source Code:

```
#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
    }

    // 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:

```
D:\STD_Muhammad_Ralfi_22
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: 12
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: 32
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
32 12
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: _
```

4. Unguided

1. Buatlah ADT Double Linked list sebagai berikut di dalam file “doublelist.h”:

```
Type infotype : kendaraan <
    nopol : string
    warna : string
    thnBuat : integer
>
Type address : pointer to ElmList
Type ElmList <
    info : infotype
    next : address
    prev : address
>
Type List <
    First : address
    Last : address
>
prosedur CreateList( in/out L : List )
fungsi alokasi( x : infotype ) : address
prosedur dealokasi( in/out P : address )
prosedur printInfo( in L : List )
prosedur insertLast( in/out L : List, in P : address )
```

Buatlah implementasi ADT Double Linked list pada file “doublelist.cpp” dan coba hasil implementasi ADT pada file “main.cpp”.

File doublelist.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(List L);
void insertLast(List &L, address P);

#endif
```

File doublelist.cpp

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

void CreateList(List &L) {
    L.first = NULL;
    L.last = NULL;
}

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

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

void printInfo(List L) {
    cout << "\nDATA LIST 1" << endl;
    address P = L.first;
    while (P != NULL) {
        cout << "no polisi : " << P->info.nopol << endl;
        cout << "warna      : " << P->info.warna << endl;
        cout << "tahun        : " << P->info.thnBuat << endl;
        if (P->next != NULL) {
            cout << endl;
        }
        P = P->next;
    }
}
```

```
void insertLast(List &L, address P) {
    if (L.first == NULL) {
        L.first = P;
        L.last = P;
    } else {
        P->prev = L.last;
        L.last->next = P;
        L.last = P;
    }
}
```

File Main.cpp

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

int main() {
    List L;
    CreateList(L);
    string nopol, warna;
    int tahun;
    bool isDuplicate;

    while (true) {
        cout << "masukkan nomor polisi: ";
        cin >> nopol;

        // Cek duplikasi
        isDuplicate = false;
        address P = L.first;
        while (P != NULL) {
            if (P->info.nopol == nopol) {
                isDuplicate = true;
                break;
            }
            P = P->next;
        }

        if (isDuplicate) {
            cout << "nomor polisi sudah terdaftar" << endl;
            continue;
        }

        cout << "masukkan warna kendaraan: ";
        cin >> warna;
        cout << "masukkan tahun kendaraan: ";
        cin >> tahun;

        kendaraan k;
        k.nopol = nopol;
        k.warna = warna;
        k.thnBuat = tahun;

        insertLast(L, alokasi(k));

        // Tampilkan list setelah setiap penambahan
        printInfo(L);
    }

    return 0;
}
```

Output:

```

D:\STD_Muhammad_Ralfi_2211104054\Q
masukkan nomor polisi: G001
masukkan warna kendaraan: Hitam
masukkan tahun kendaraan: 2010

DATA LIST 1
no polisi : G001
warna      : Hitam
tahun      : 2010
masukkan nomor polisi: G002
masukkan warna kendaraan: Putih
masukkan tahun kendaraan: 2011

DATA LIST 1
no polisi : G001
warna      : Hitam
tahun      : 2010

no polisi : G002
warna      : Putih
tahun      : 2011
masukkan nomor polisi: G003
masukkan warna kendaraan: Silver
masukkan tahun kendaraan: 2006

DATA LIST 1
no polisi : G001
warna      : Hitam
tahun      : 2010

no polisi : G002
warna      : Putih
tahun      : 2011

no polisi : G003
warna      : Silver
tahun      : 2006

```

- Carilah elemen dengan nomor polisi D001 dengan membuat fungsi baru. fungsi
`findElm(L : List, x : infotype) : address`

File main.cpp

```

// Menampilkan seluruh data list
cout << "\nDATA LIST 1" << endl;
printInfo(L);

// Memungkinkan pencarian berdasarkan nomor polisi
string searchNopol;
cout << "\nMasukkan nomor polisi yang ingin dicari: ";
cin >> searchNopol;
address found = findElm(L, searchNopol);
if (found != NULL) {
    cout << "Data ditemukan:\n";
    cout << "no polisi : " << found->info.nopol << endl;
    cout << "warna      : " << found->info.warna << endl;
    cout << "tahun      : " << found->info.thnBuat << endl;
} else {
    cout << "Data tidak ditemukan\n";
}

```



```
}
}
```

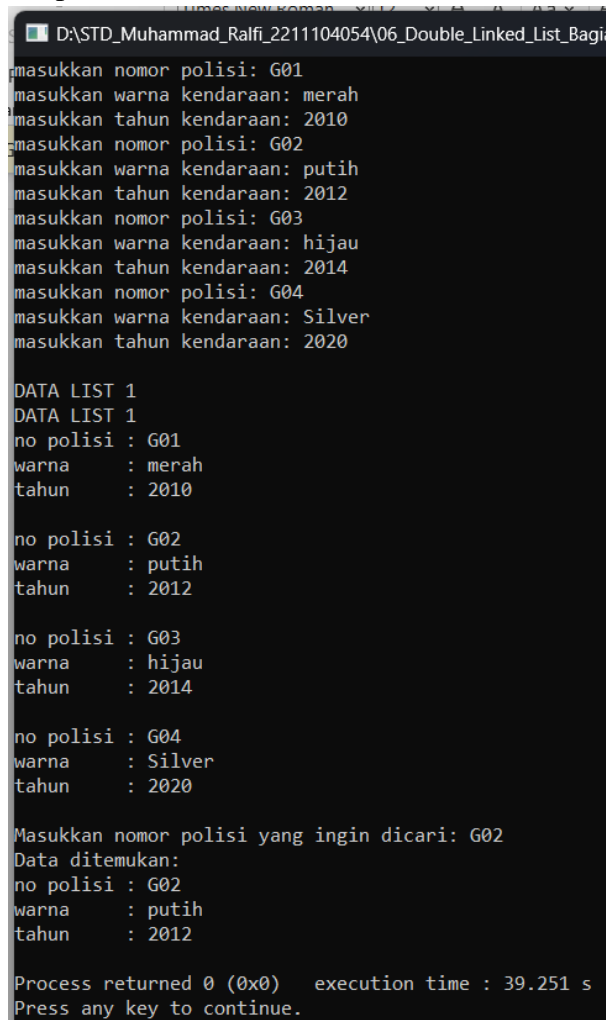
File Doublelist.cpp

```
address findElm(List L, string nopol) {
    address P = L.first;
    while (P != NULL) {
        if (P->info.nopol == nopol) {
            return P;
        }
        P = P->next;
    }
    return NULL;
}
```

File Doublelist.h

```
address findElm(List L, string nopol);
```

Output:



```
D:\STD_Muhammad_Ralfi_2211104054\06_Double_Linked_List_Bagja
masukkan nomor polisi: G01
masukkan warna kendaraan: merah
masukkan tahun kendaraan: 2010
masukkan nomor polisi: G02
masukkan warna kendaraan: putih
masukkan tahun kendaraan: 2012
masukkan nomor polisi: G03
masukkan warna kendaraan: hijau
masukkan tahun kendaraan: 2014
masukkan nomor polisi: G04
masukkan warna kendaraan: Silver
masukkan tahun kendaraan: 2020

DATA LIST 1
DATA LIST 1
no polisi : G01
warna      : merah
tahun      : 2010

no polisi : G02
warna      : putih
tahun      : 2012

no polisi : G03
warna      : hijau
tahun      : 2014

no polisi : G04
warna      : Silver
tahun      : 2020

Masukkan nomor polisi yang ingin dicari: G02
Data ditemukan:
no polisi : G02
warna      : putih
tahun      : 2012

Process returned 0 (0x0)   execution time : 39.251 s
Press any key to continue.
```

3. Hapus elemen dengan nomor polisi D003 dengan prosedur delete.
 - prosedur deleteFirst(in/out L : List, in/out P : address)
 - prosedur deleteLast(in/out L : List, in/out P : address)
 - prosedur deleteAfter(in Prec : address, in/out: P : address)

File main.cpp

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

void hapusKendaraan(List &L) {
    string nopol;
    cout << "Masukkan nomor polisi yang ingin dihapus: ";
    cin >> nopol;

    address P = findElm(L, nopol);
    if (P == NULL) {
        cout << "Data dengan nomor polisi " << nopol << " tidak
ditemukan.\n";
    } else {
        if (P == L.first) {
            deleteFirst(L, P);
            cout << "Data dengan nomor polisi " << nopol << "
berhasil dihapus dari posisi pertama.\n";
        } else if (P == L.last) {
            deleteLast(L, P);
            cout << "Data dengan nomor polisi " << nopol << "
berhasil dihapus dari posisi terakhir.\n";
        } else {
            deleteAfter(P->prev, P);
            cout << "Data dengan nomor polisi " << nopol << "
berhasil dihapus dari posisi tengah.\n";
        }
    }
}

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

    // Menambahkan data kendaraan
    for (int i = 0; i < 4; i++) {
        string nopol, warna;
        int tahun;

        cout << "masukkan nomor polisi: ";
        cin >> nopol;

        if (isDuplicate(L, nopol)) {
            cout << "nomor polisi sudah terdaftar" << endl;
            i--;
            continue;
        }

        cout << "masukkan warna kendaraan: ";
        cin >> warna;
        cout << "masukkan tahun kendaraan: ";
        cin >> tahun;
```

```
kendaraan k;
k.nopol = nopol;
k.warna = warna;
k.thnBuat = tahun;

insertLast(L, alokasi(k));
}

// Menampilkan seluruh data list
cout << "\nDATA LIST 1" << endl;
printInfo(L);

// Menghapus data kendaraan berdasarkan nomor polisi
hapusKendaraan(L);

// Menampilkan data list setelah penghapusan
cout << "\nDATA LIST SETELAH PENGHAPUSAN" << endl;
printInfo(L);

return 0;
}
```

File doublelist.cpp

```
void deleteFirst(List &L, address &P) {
    if (L.first != NULL) {
        P = L.first;
        if (L.first == L.last) { // Jika hanya satu elemen
            L.first = NULL;
            L.last = NULL;
        } else {
            L.first = L.first->next;
            L.first->prev = NULL;
        }
        P->next = NULL;
        dealokasi(P);
    }
}

void deleteLast(List &L, address &P) {
    if (L.last != NULL) {
        P = L.last;
        if (L.first == L.last) { // Jika hanya satu elemen
            L.first = NULL;
            L.last = NULL;
        } else {
            L.last = L.last->prev;
            L.last->next = NULL;
        }
        P->prev = NULL;
        dealokasi(P);
    }
}
```

```

}
void deleteAfter(address Prec, address &P) {
    if (Prec != NULL && Prec->next != NULL) {
        P = Prec->next;
        if (P->next != NULL) {
            Prec->next = P->next;
            P->next->prev = Prec;
        } else {
            Prec->next = NULL;
        }
        P->prev = NULL;
        P->next = NULL;
        dealokasi(P);
    }
}

```

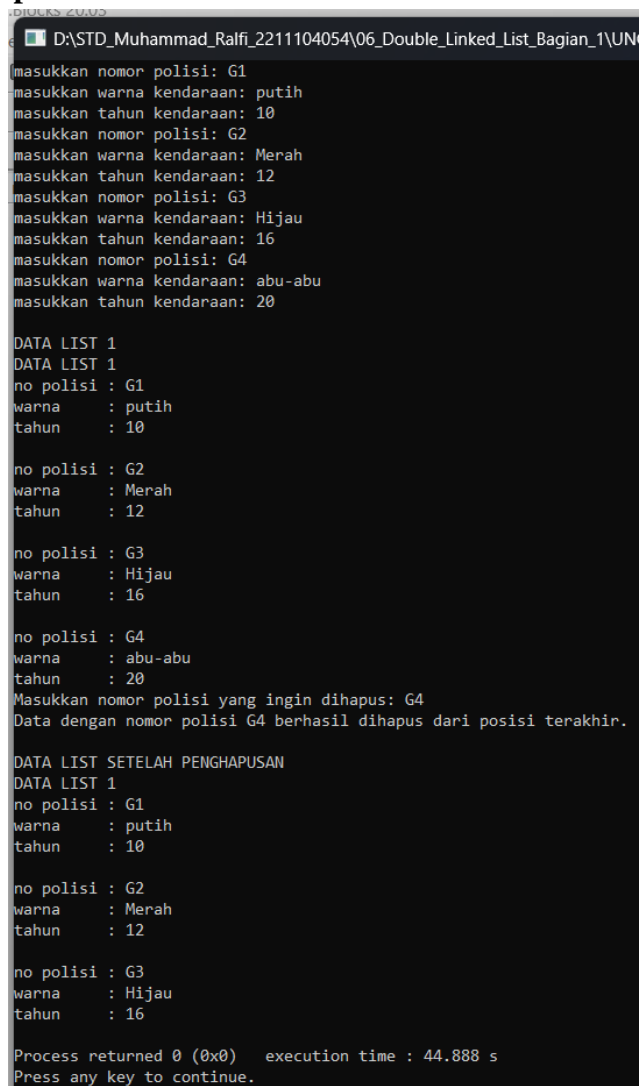
File doublelist.h

```

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

```

Output



```

D:\STD_Muhammad_Ralfi_2211104054\06_Double_Linked_List_Bagian_1\UN
masukkan nomor polisi: G1
masukkan warna kendaraan: putih
masukkan tahun kendaraan: 10
masukkan nomor polisi: G2
masukkan warna kendaraan: Merah
masukkan tahun kendaraan: 12
masukkan nomor polisi: G3
masukkan warna kendaraan: Hijau
masukkan tahun kendaraan: 16
masukkan nomor polisi: G4
masukkan warna kendaraan: abu-abu
masukkan tahun kendaraan: 20

DATA LIST 1
DATA LIST 1
no polisi : G1
warna      : putih
tahun      : 10

no polisi : G2
warna      : Merah
tahun      : 12

no polisi : G3
warna      : Hijau
tahun      : 16

no polisi : G4
warna      : abu-abu
tahun      : 20
Masukkan nomor polisi yang ingin dihapus: G4
Data dengan nomor polisi G4 berhasil dihapus dari posisi terakhir.

DATA LIST SETELAH PENGHAPUSAN
DATA LIST 1
no polisi : G1
warna      : putih
tahun      : 10

no polisi : G2
warna      : Merah
tahun      : 12

no polisi : G3
warna      : Hijau
tahun      : 16

Process returned 0 (0x0)   execution time : 44.888 s
Press any key to continue.

```

5. Kesimpulan

Double Linked List merupakan struktur data yang menawarkan fleksibilitas tinggi dalam pengaksesan dan manipulasi data. Tidak seperti Single Linked List, di mana pengaksesan elemen hanya dapat dilakukan secara satu arah, Double Linked List memungkinkan traversal baik ke arah depan maupun ke belakang melalui penggunaan dua pointer pada setiap elemennya—pointer `next` yang menunjuk ke elemen berikutnya dan pointer `prev` yang menunjuk ke elemen sebelumnya.

Struktur ini membuat operasi-operasi manipulasi data, seperti penyisipan (insertion), penghapusan (deletion), dan pencarian (searching), dapat dilakukan dengan lebih efisien. Hal ini karena kita tidak perlu selalu memulai dari elemen pertama atau terakhir untuk mengakses suatu elemen tertentu; akses dapat langsung dilakukan dari arah yang paling dekat. Dengan demikian, Double Linked List menjadi pilihan yang lebih optimal dibandingkan Single Linked List dalam skenario di mana diperlukan fleksibilitas tinggi dalam pengelolaan dan manipulasi data pada kedua arah.