

LAPORAN PRAKTIKUM
MODUL 6
“DOUBLE LINKED LIST (1)”



Disusun Oleh:
Tiurma Grace Angelina (2311104042)
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 konsep modul linked list.
- Mengaplikasikan konsep double linked list dengan menggunakan pointer dan dengan Bahasa C.

2. Landasan Teori

- Double Linked List

Double Linked List, atau daftar berantai ganda, adalah struktur data yang setiap elemennya memiliki dua penunjuk, yaitu penunjuk ke elemen sebelumnya (prev) dan penunjuk ke elemen berikutnya (next). Keberadaan kedua penunjuk ini memungkinkan navigasi pada list dilakukan ke dua arah, baik maju maupun mundur. Double linked list memiliki dua penunjuk utama, yaitu First, yang menunjuk ke elemen pertama dalam list, dan Last, yang menunjuk ke elemen terakhir. Struktur ini lebih fleksibel dibanding single linked list, yang hanya memiliki penunjuk ke satu arah, karena dapat mempermudah akses data dari kedua sisi list. Setiap elemen dalam double linked list terhubung ke elemen sebelum dan sesudahnya, sehingga setiap perubahan pada elemen, seperti penambahan atau penghapusan, akan melibatkan penyesuaian penunjuk pada elemen-elemen terkait. Elemen baru dapat ditambahkan atau dihapus di berbagai posisi dalam list, baik di awal, di akhir, maupun di antara dua elemen lainnya. Struktur ini mendukung tiga operasi dasar: insert (menyisipkan), delete (menghapus), dan pencarian elemen berdasarkan kondisi tertentu.

- Operasi Insert

Operasi insert pada double linked list digunakan untuk menambahkan elemen baru ke dalam list pada posisi tertentu:

- **Insert First:** Menambahkan elemen baru di awal list, sehingga elemen tersebut menjadi elemen pertama (First). Elemen yang sebelumnya berada di posisi pertama akan memiliki penunjuk prev yang diarahkan ke elemen baru.
- **Insert Last:** Menambahkan elemen baru di akhir list. Elemen ini akan menjadi elemen terakhir (Last), dengan penunjuk next dari elemen yang sebelumnya terakhir diarahkan ke elemen baru tersebut.
- **Insert After:** Menyisipkan elemen baru setelah elemen tertentu dalam list. Operasi ini mengatur ulang penunjuk next dan prev pada elemen terkait agar elemen baru dapat ditempatkan di posisi yang diinginkan.

- Operasi Delete

Operasi delete pada double linked list digunakan untuk menghapus elemen pada posisi tertentu di dalam list:

- **Delete First:** Menghapus elemen pertama dalam list. Setelah elemen ini dihapus, elemen kedua dalam list akan menjadi elemen pertama yang baru, dengan penunjuk prev yang diatur menjadi NULL.

- **Delete Last:** Menghapus elemen terakhir dari list. Setelah elemen ini dihapus, elemen kedua terakhir akan menjadi elemen terakhir yang baru, dan penunjuk next dari elemen tersebut akan diatur menjadi NULL.
- **Delete After:** Menghapus elemen yang berada setelah elemen tertentu dalam list. Operasi ini mengatur ulang penunjuk next dan prev pada elemen terkait sehingga elemen yang dihapus tidak lagi terhubung dengan list.

Dengan adanya operasi-operasi ini, double linked list memberikan fleksibilitas tinggi untuk mengelola data. Struktur ini memungkinkan penambahan atau penghapusan data dengan mudah di berbagai posisi, serta mempermudah akses data melalui penunjuk next dan prev. Fleksibilitas ini menjadikan double linked list sangat berguna dalam aplikasi yang memerlukan manipulasi data dua arah dalam jumlah besar, seperti dalam sistem manajemen data atau pengelolaan antrian yang kompleks.

3. Guided

- **Array**

1. **CODE :**

```

1  #include <iostream>
2  using namespace std;
3
4  class Node {
5  public:
6      int data;
7      Node* prev;
8      Node* next;
9  };
10
11 class DoublyLinkedList {
12 public:
13     Node* head;
14     Node* tail;
15
16     // Constructor untuk inisialisasi head dan tail
17     DoublyLinkedList() {
18         head = nullptr;
19         tail = nullptr;
20     }
21
22     // Fungsi untuk menambahkan elemen di depan list
23     void insert(int data) {
24         Node* newNode = new Node;
25         newNode->data = data;
26         newNode->prev = nullptr;
27         newNode->next = head;

```

```

27     newNode->next = head;
28
29     if (head != nullptr) {
30         head->prev = newNode;
31     } else {
32         tail = newNode; // Jika list kosong, tail juga mengarah ke node baru
33     }
34     head = newNode;
35 }
36
37 // Fungsi untuk menghapus elemen dari depan list
38 void deleteNode() {
39     if (head == nullptr) {
40         return; // Jika list kosong
41     }
42     Node* temp = head;
43     head = head->next;
44     if (head != nullptr) {
45         head->prev = nullptr;
46     } else {
47         tail = nullptr; // Jika hanya satu elemen di list
48     }
49     delete temp; // Hapus elemen
50 }
51
52 // Fungsi untuk mengupdate data di list
53 bool update(int oldData, int newData) {

```

```

main.cpp x include\doublelist.h x src\doublelist.cpp x main.cpp x
53 bool update(int oldData, int newData) {
54     Node* current = head;
55     while (current != nullptr) {
56         if (current->data == oldData) {
57             current->data = newData;
58             return true; // Jika data ditemukan dan diupdate
59         }
60         current = current->next;
61     }
62     return false; // Jika data tidak ditemukan
63 }
64
65 // Fungsi untuk menghapus semua elemen di list
66 void deleteAll() {
67     Node* current = head;
68     while (current != nullptr) {
69         Node* temp = current;
70         current = current->next;
71         delete temp;
72     }
73     head = nullptr;
74     tail = nullptr;
75 }
76
77 // Fungsi untuk menampilkan semua elemen di list
78 void display() {
79     Node* current = head;

```

```
main.cpp x include\doublelist.h x src\doublelist.cpp x main.cpp x
79     Node* current = head;
80     while (current != nullptr) {
81         cout << current->data << " ";
82         current = current->next;
83     }
84     cout << endl;
85 }
86 };
87
88 int main() {
89     DoublyLinkedList list;
90     while (true) {
91         cout << "1. Add data" << endl;
92         cout << "2. Delete data" << endl;
93         cout << "3. Update data" << endl;
94         cout << "4. Clear data" << endl;
95         cout << "5. Display data" << endl;
96         cout << "6. Exit" << endl;
97
98         int choice;
99         cout << "Enter your choice: ";
100        cin >> choice;
101
102        switch (choice) {
103            case 1: {
104                int data;
105                cout << "Enter data to add: ";
```

```
105                cout << "Enter data to add: ";
106                cin >> data;
107                list.insert(data);
108                break;
109            }
110            case 2: {
111                list.deleteNode();
112                break;
113            }
114            case 3: {
115                int oldData, newData;
116                cout << "Enter old data: ";
117                cin >> oldData;
118                cout << "Enter new data: ";
119                cin >> newData;
120                bool updated = list.update(oldData, newData);
121                if (!updated) {
122                    cout << "Data not found" << endl;
123                }
124                break;
125            }
126            case 4: {
127                list.deleteAll();
128                break;
129            }
130            case 5: {
131                list.display();
```

```

120         bool updated = list.update(oldData, newData);
121         if (!updated) {
122             cout << "Data not found" << endl;
123         }
124         break;
125     }
126     case 4: {
127         list.deleteAll();
128         break;
129     }
130     case 5: {
131         list.display();
132         break;
133     }
134     case 6: {
135         return 0;
136     }
137     default: {
138         cout << "Invalid choice" << endl;
139         break;
140     }
141 }
142
143 return 0;
144 }
145

```

Penjelasan :

Program di atas adalah implementasi dari double linked list yang mendukung berbagai operasi seperti menambahkan data, menghapus data, memperbarui data, menghapus semua data, dan menampilkan data. Double linked list ini memungkinkan penyimpanan elemen-elemen yang terhubung dalam dua arah, yaitu maju dan mundur. Berikut penjelasan dari setiap bagian dan operasi yang ada dalam program ini:

Struktur dan Inisialisasi Double Linked List

Program ini menggunakan dua kelas utama:

1. **Node:** Setiap elemen dalam double linked list direpresentasikan sebagai node. Setiap node memiliki tiga komponen: data, prev (penunjuk ke elemen sebelumnya), dan next (penunjuk ke elemen berikutnya).
2. **DoublyLinkedList:** Kelas ini merepresentasikan double linked list secara keseluruhan dan memiliki dua penunjuk utama, yaitu head dan tail.
 - head menunjuk ke elemen pertama dalam list.
 - tail menunjuk ke elemen terakhir dalam list.

Kedua penunjuk ini membantu mengatur list dan memudahkan navigasi, baik dari awal ke akhir maupun dari akhir ke awal list.

Operasi yang Didukung

1. Menambahkan Elemen di Depan List (Insert)

- Operasi ini menambahkan elemen baru di depan list. Jika list kosong, elemen baru ini menjadi satu-satunya elemen dan ditunjuk oleh head dan tail.
- Jika list tidak kosong, elemen baru akan dihubungkan sebagai elemen pertama, dan penunjuk head akan diarahkan ke elemen baru tersebut.
- Setelah penambahan, penunjuk next dari elemen baru diarahkan ke elemen yang sebelumnya merupakan elemen pertama, dan penunjuk prev dari elemen yang sebelumnya pertama diarahkan ke elemen baru.

2. Menghapus Elemen dari Depan List (Delete)

- Operasi ini menghapus elemen pertama dari list. Jika hanya ada satu elemen di list, head dan tail akan diset ke nullptr setelah penghapusan.
- Jika terdapat lebih dari satu elemen, elemen kedua akan menjadi elemen pertama setelah penghapusan, dan penunjuk prev dari elemen baru pertama akan diatur menjadi nullptr.

3. Memperbarui Data di List (Update)

- Operasi ini mencari elemen dalam list yang memiliki nilai data tertentu. Jika elemen tersebut ditemukan, nilai datanya diperbarui dengan nilai baru yang diberikan.
- Jika elemen yang dicari tidak ditemukan, operasi ini akan menampilkan pesan bahwa data tidak ditemukan.

4. Menghapus Semua Elemen di List (Delete All)

- Operasi ini menghapus seluruh elemen dalam list satu per satu, mulai dari elemen pertama hingga elemen terakhir. Setelah penghapusan, list akan kosong, dan head serta tail diset ke nullptr.
- Operasi ini berguna untuk membersihkan list sepenuhnya tanpa menyisakan data.

5. Menampilkan Semua Elemen di List (Display)

- Operasi ini menampilkan semua elemen di dalam list dari awal hingga akhir. Program akan mencetak data dari setiap elemen secara berurutan.

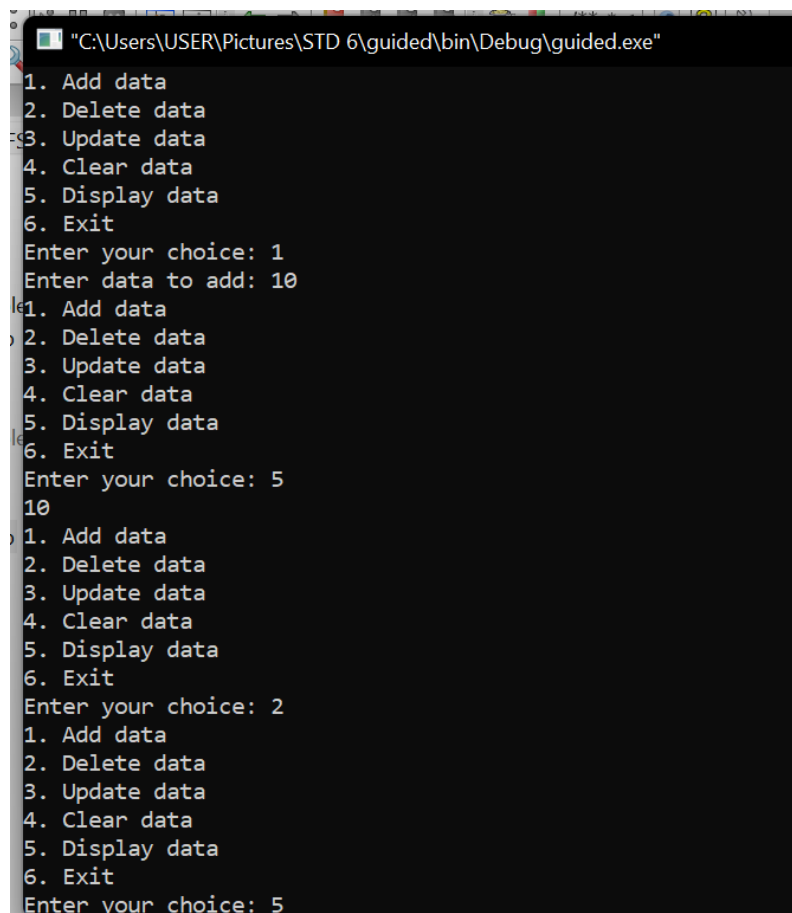
- Jika list kosong, operasi ini tidak akan mencetak apapun, dan hanya akan menampilkan baris kosong.

6. Antarmuka Pengguna (Menu)

- Program menggunakan antarmuka berbasis teks untuk menerima input pengguna. Pengguna dapat memilih operasi yang ingin dijalankan dengan memasukkan angka yang sesuai dari menu.
- Menu ini menyediakan opsi-opsi untuk menambahkan data, menghapus data, memperbarui data, membersihkan list, menampilkan data, atau keluar dari program.

Program ini terus berjalan dalam loop hingga pengguna memilih opsi untuk keluar. Setiap pilihan yang dimasukkan oleh pengguna dieksekusi oleh program sesuai dengan operasi yang dipilih, dan program akan menampilkan hasil dari setiap operasi (misalnya menampilkan data atau menunjukkan pesan jika data yang dicari tidak ditemukan).

OUTPUT :



```

"C:\Users\USER\Pictures\STD 6\guided\bin\Debug\guided.exe"
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: 10
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
10
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 2
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5

```



```

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

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

Process returned 0 (0x0)   execution time : 38.081 s
Press any key to continue.

```

4. Unguided

1. CODE : doublelist.h

```

1 // doublelist.h
2 #ifndef DOUBLELIST_H
3 #define DOUBLELIST_H
4 #include <string>
5 using namespace std;
6
7 struct infotype {
8     string nopol;
9     string warna;
10    int thnBuat;
11 };
12
13 typedef struct ElmList *address;
14
15 struct ElmList {
16     infotype info;
17     address next;
18     address prev;
19 };
20
21 struct List {
22     address First;
23     address Last;
24 };
25
26 void CreateList(List &L);
27 address alokasi(infotype x);

```

```

13 typedef struct ElmList *address;
14
15 struct ElmList {
16     infotype info;
17     address next;
18     address prev;
19 };
20
21 struct List {
22     address First;
23     address Last;
24 };
25
26 void CreateList(List &L);
27 address alokasi(infotype x);
28 void dealokasi(address &P);
29 void printInfo(List L);
30 void insertLast(List &L, address P);
31 bool isKendaraanExist(List L, string nopol);
32 address findElm(List L, string nopol); // Task 2 function
33 void deleteFirst(List &L, address &P); // Task 3 function
34 void deleteLast(List &L, address &P); // Task 3 function
35 void deleteAfter(List &L, address Prec, address &P); // Task 3 function
36
37 #endif
38

```

Penjelasan :

Header file doublelist.h di atas adalah deklarasi struktur dan fungsi-fungsi yang digunakan dalam implementasi double linked list untuk menyimpan informasi tentang kendaraan. File ini berisi tipe data, struktur, dan fungsi-fungsi yang dibutuhkan untuk mengelola double linked list, seperti menambahkan, mencari, dan menghapus elemen dalam list. Berikut adalah penjelasan dari setiap bagian secara rinci:

Struktur Data

1. infotype

Struktur infotype digunakan untuk menyimpan informasi terkait kendaraan.

infotype memiliki tiga atribut:

- nopol: Merupakan nomor polisi kendaraan (tipe string).
- warna: Warna kendaraan (tipe string).
- thnBuat: Tahun pembuatan kendaraan (tipe int).

Struktur ini adalah tipe data untuk menyimpan informasi kendaraan yang akan digunakan sebagai isi dari setiap elemen dalam double linked list.

2. address

address adalah tipe data pointer yang menunjuk ke struktur ElmList. Tipe ini mendefinisikan sebuah pointer ke elemen list yang digunakan untuk navigasi di dalam linked list, memungkinkan tiap elemen dalam list terhubung dengan elemen lainnya.

3. ElmList

ElmList adalah struktur yang merepresentasikan satu elemen atau node dalam double linked list. Struktur ini terdiri dari:

- info: Tipe infotype yang menyimpan informasi kendaraan pada elemen tersebut.
- next: Tipe address yang menunjuk ke elemen berikutnya dalam list.
- prev: Tipe address yang menunjuk ke elemen sebelumnya dalam list.

Dengan adanya next dan prev, double linked list memungkinkan navigasi dua arah dari satu elemen ke elemen lainnya.

4. **List**

List adalah struktur utama yang menyimpan informasi tentang list itu sendiri, yaitu:

- First: Pointer yang menunjuk ke elemen pertama dalam list.
- Last: Pointer yang menunjuk ke elemen terakhir dalam list.

Dengan adanya First dan Last, list dapat dengan mudah mengakses elemen pertama dan terakhir, yang berguna dalam berbagai operasi seperti penambahan atau penghapusan elemen dari ujung list.

Fungsi dan Prosedur

1. **CreateList**

Fungsi ini bertanggung jawab untuk menginisialisasi sebuah list baru. Fungsi ini akan memastikan bahwa list dimulai dalam keadaan kosong dengan menyetel First dan Last ke NULL.

2. **alokasi**

Fungsi ini berfungsi untuk membuat elemen baru (node) dalam list. Fungsi ini menerima data tipe infotype sebagai parameter dan mengalokasikan memori untuk elemen baru, lalu mengatur informasi di dalamnya. Fungsi ini mengembalikan alamat elemen yang baru saja dibuat.

3. **dealokasi**

Fungsi ini digunakan untuk menghapus elemen dari list dengan membebaskan memori yang digunakan oleh elemen tersebut. Ini penting untuk mencegah kebocoran memori setelah elemen dihapus.

4. **printInfo**

Fungsi ini digunakan untuk menampilkan informasi semua elemen dalam list. Fungsi ini mencetak nomor polisi, warna, dan tahun pembuatan setiap kendaraan yang ada dalam list, dimulai dari elemen pertama hingga terakhir.

5. **insertLast**

Fungsi ini bertugas menambahkan elemen baru di akhir list. Fungsi ini akan menghubungkan elemen baru dengan elemen terakhir di dalam list, mengatur next dan prev untuk memastikan elemen tersebut ditempatkan di akhir, dan memperbarui penunjuk Last.

6. **isKendaraanExist**

Fungsi ini digunakan untuk mengecek apakah kendaraan dengan nomor polisi tertentu sudah ada dalam list. Fungsi ini menerima nomor polisi sebagai parameter dan memeriksa setiap elemen dalam list untuk mencocokkan nomor polisi tersebut. Fungsi ini mengembalikan nilai true jika ditemukan, dan false jika tidak.

7. **findElm**

Fungsi ini mencari elemen dalam list berdasarkan nomor polisi yang diberikan. Fungsi ini akan mengembalikan alamat elemen yang sesuai jika nomor polisi ditemukan, atau NULL jika elemen dengan nomor polisi tersebut tidak ada di dalam list. Fungsi ini digunakan dalam operasi-operasi yang memerlukan pencarian data tertentu.

8. **deleteFirst**

Fungsi ini menghapus elemen pertama dari list. Fungsi ini mengatur First untuk

menunjuk ke elemen kedua setelah elemen pertama dihapus, dan mengatur prev dari elemen baru pertama menjadi NULL. Jika list hanya memiliki satu elemen, maka First dan Last akan diset menjadi NULL setelah penghapusan.

9. **deleteLast**

Fungsi ini menghapus elemen terakhir dari list. Fungsi ini mengatur Last untuk menunjuk ke elemen kedua terakhir setelah elemen terakhir dihapus, dan mengatur next dari elemen baru terakhir menjadi NULL. Jika list hanya memiliki satu elemen, maka First dan Last akan diset menjadi NULL setelah penghapusan.

10. **deleteAfter**

Fungsi ini menghapus elemen yang berada setelah elemen tertentu (diberikan sebagai parameter Prec). Fungsi ini berguna untuk menghapus elemen di tengah list. Fungsi ini mengatur next dari elemen sebelumnya (Prec) dan prev dari elemen berikutnya agar terhubung satu sama lain setelah elemen yang dihapus.

Header file doublelist.h menyediakan struktur data dan deklarasi fungsi yang diperlukan untuk mengelola double linked list yang menyimpan informasi kendaraan. Dengan fungsi-fungsi seperti insertLast, deleteFirst, deleteLast, dan deleteAfter, list ini memungkinkan penambahan, pencarian, dan penghapusan elemen dengan mudah. Struktur List dengan pointer First dan Last memberikan kemudahan dalam akses data dari ujung list, sedangkan struktur ElmList dengan penunjuk dua arah (next dan prev) memungkinkan list ini diakses dari dua arah untuk memudahkan manipulasi data.

2. CODE : doublelist.cpp

```
1 // doublelist.cpp
2 #include "doublelist.h"
3 #include <iostream>
4 using namespace std;
5
6 void CreateList(List &L) {
7     L.First = NULL;
8     L.Last = NULL;
9 }
10
11 address alokasi(infotype x) {
12     address P = new ElmList;
13     P->info = x;
14     P->next = NULL;
15     P->prev = NULL;
16     return P;
17 }
18
19 void dealokasi(address &P) {
20     delete P;
21 }
22
23 void printInfo(List L) {
24     address P = L.First;
25     cout << "\nDATA LIST 1" << endl;
26     while (P != NULL) {
27         cout << "no polisi : " << P->info.nopol << endl;
```

```

27         cout << "no polisi : " << P->info.nopol << endl;
28         cout << "warna      : " << P->info.warna << endl;
29         cout << "tahun      : " << P->info.thnBuat << endl;
30         P = P->next;
31     }
32 }
33
34 void insertLast(List &L, address P) {
35     if (L.First == NULL) {
36         L.First = P;
37         L.Last = P;
38     } else {
39         P->prev = L.Last;
40         L.Last->next = P;
41         L.Last = P;
42     }
43 }
44
45 bool isKendaraanExist(List L, string nopol) {
46     address P = L.First;
47     while (P != NULL) {
48         if (P->info.nopol == nopol) {
49             return true;
50         }
51         P = P->next;
52     }
53     return false;

```

```

53     return false;
54 }
55
56 // Task 2: Find element based on nomor polisi
57 address findElm(List L, string nopol) {
58     address P = L.First;
59     while (P != NULL) {
60         if (P->info.nopol == nopol) {
61             return P;
62         }
63         P = P->next;
64     }
65     return NULL;
66 }
67
68 // Task 3: Delete first element
69 void deleteFirst(List &L, address &P) {
70     if (L.First != NULL) {
71         P = L.First;
72         if (L.First == L.Last) {
73             L.First = NULL;
74             L.Last = NULL;
75         } else {
76             L.First = L.First->next;
77             L.First->prev = NULL;
78         }
79         P->next = NULL;

```

```

79         P->next = NULL;
80     }
81 }
82
83 // Task 3: Delete last element
84 void deleteLast(List &L, address &P) {
85     if (L.Last != NULL) {
86         P = L.Last;
87         if (L.First == L.Last) {
88             L.First = NULL;
89             L.Last = NULL;
90         } else {
91             L.Last = L.Last->prev;
92             L.Last->next = NULL;
93         }
94         P->prev = NULL;
95     }
96 }
97
98 // Task 3: Delete after a specified element
99 void deleteAfter(List &L, address Prec, address &P) {
100     if (Prec != NULL && Prec->next != NULL) {
101         P = Prec->next;
102         if (P == L.Last) {
103             deleteLast(L, P);
104         } else {
105             Prec->next = P->next;
106             P->next->prev = Prec;
107             P->next = NULL;
108             P->prev = NULL;
109         }
110     }
111 }
112
113 // Task 3: Delete after a specified element
114 void deleteAfter(List &L, address Prec, address &P) {
115     if (Prec != NULL && Prec->next != NULL) {
116         P = Prec->next;
117         if (P == L.Last) {
118             deleteLast(L, P);
119         } else {
120             Prec->next = P->next;
121             P->next->prev = Prec;
122             P->next = NULL;
123             P->prev = NULL;
124         }
125     }
126 }

```

Penjelasan :

File doublelist.cpp ini adalah implementasi dari berbagai fungsi untuk mengelola double linked list yang menyimpan data kendaraan. Setiap fungsi bertanggung jawab untuk operasi tertentu, seperti membuat list, mengalokasikan dan menghapus elemen, menambahkan dan menghapus elemen dari posisi tertentu, serta menampilkan data. Berikut penjelasan dari masing-masing fungsionalitas tanpa penjelasan per kode atau fungsi secara teknis.

Fungsi dan Fungsionalitas

1. **Inisialisasi List (CreateList)** Fungsi ini bertugas untuk menginisialisasi list kosong. Ketika list pertama kali dibuat, First dan Last disetel ke NULL, menunjukkan bahwa list masih kosong dan belum ada elemen di dalamnya.
2. **Alokasi Elemen (alokasi)** Fungsi ini bertujuan untuk membuat elemen baru dalam list. Ketika informasi kendaraan diberikan (berupa nomor polisi, warna, dan tahun pembuatan), fungsi ini akan membuat elemen (node) baru dan menyimpan data tersebut di dalam elemen tersebut.

Elemen baru ini nantinya bisa ditempatkan di mana saja dalam list sesuai kebutuhan (misalnya di awal atau di akhir).

3. **Dealokasi Elemen (dealokasi)** Fungsi ini digunakan untuk menghapus elemen dari memori setelah elemen tersebut tidak lagi diperlukan. Hal ini penting untuk mencegah kebocoran memori, terutama ketika elemen dihapus dari list, sehingga ruang yang digunakan elemen tersebut dapat dibebaskan.
4. **Menampilkan Informasi List (printInfo)** Fungsi ini akan menampilkan semua elemen yang ada di dalam list. Setiap elemen dicetak satu per satu, menunjukkan informasi kendaraan seperti nomor polisi, warna, dan tahun pembuatan. Fungsi ini memulai dari elemen pertama (First) dan berlanjut ke elemen berikutnya hingga mencapai akhir list (NULL).
5. **Menambahkan Elemen di Akhir List (insertLast)** Fungsi ini menambahkan elemen baru di akhir list. Jika list kosong, elemen tersebut menjadi elemen pertama dan terakhir sekaligus. Jika sudah ada elemen dalam list, elemen baru ini ditambahkan setelah elemen terakhir, dan pointer Last diperbarui untuk menunjuk ke elemen baru tersebut. Dengan demikian, elemen baru selalu ditempatkan di akhir list.
6. **Pengecekan Keberadaan Kendaraan (isKendaraanExist)** Fungsi ini melakukan pengecekan apakah ada kendaraan dengan nomor polisi tertentu dalam list. Fungsi ini berguna untuk memastikan apakah kendaraan sudah ada sebelum menambahkan atau memodifikasi data kendaraan tersebut, menghindari duplikasi data.
7. **Mencari Elemen Berdasarkan Nomor Polisi (findElm)** Fungsi ini mencari elemen dalam list yang memiliki nomor polisi tertentu. Fungsi ini digunakan untuk menemukan elemen tertentu dalam list, terutama jika kita ingin mengakses atau mengubah data spesifik berdasarkan nomor polisi kendaraan.
8. **Menghapus Elemen Pertama (deleteFirst)** Fungsi ini menghapus elemen pertama dari list. Jika list hanya memiliki satu elemen, maka First dan Last akan diatur ke NULL, menandakan bahwa list kembali kosong. Jika ada lebih dari satu elemen, elemen pertama dihapus, dan elemen kedua menjadi elemen pertama yang baru.
9. **Menghapus Elemen Terakhir (deleteLast)** Fungsi ini menghapus elemen terakhir dari list. Jika list hanya memiliki satu elemen, maka list akan menjadi kosong setelah elemen ini dihapus. Jika ada lebih dari satu elemen, elemen terakhir dihapus, dan elemen sebelumnya menjadi elemen terakhir yang baru.
10. **Menghapus Elemen Setelah Elemen Tertentu (deleteAfter)** Fungsi ini menghapus elemen yang berada setelah elemen tertentu (diberikan sebagai parameter). Fungsi ini berguna untuk menghapus elemen di tengah list. Jika elemen setelah elemen yang ditentukan adalah elemen terakhir, fungsi deleteLast akan dipanggil. Jika bukan, fungsi ini akan menghubungkan elemen sebelum dan sesudah elemen yang dihapus, memastikan list tetap terhubung dengan benar.

File doublelist.cpp mengimplementasikan berbagai fungsi yang diperlukan untuk mengelola double linked list dengan informasi kendaraan. Dengan fungsi-fungsi ini, list dapat diinisialisasi, elemen-elemen dapat ditambahkan dan dihapus dari posisi yang berbeda dalam list, dan informasi dapat dicetak serta dicari. Fungsi-fungsi ini menyediakan cara yang fleksibel untuk mengelola data kendaraan dalam bentuk double linked list, memungkinkan manipulasi data secara efisien.

3. CODE : main.cpp

```

1  // main.cpp
2  #include "doublelist.h"
3  #include <iostream>
4  using namespace std;
5
6  int main() {
7      List L;
8      infotype kendaraan;
9      address P;
10
11     CreateList(L);
12
13     // Insert sample data
14     while (true) {
15         cout << "masukkan nomor polisi: ";
16         cin >> kendaraan.nopol;
17
18         if (isKendaraanExist(L, kendaraan.nopol)) {
19             cout << "nomor polisi sudah terdaftar" << endl;
20             continue;
21         }
22
23         cout << "masukkan warna kendaraan: ";
24         cin >> kendaraan.warna;
25         cout << "masukkan tahun kendaraan: ";
26         cin >> kendaraan.thnBuat;
27
28         P = alokasi(kendaraan);
29         insertLast(L, P);
30
31         if (L.First != NULL && L.First->next != NULL && L.First->next->next != NULL) {
32             break;
33         }
34     }
35
36     // Display the list
37     printInfo(L);
38
39     // Task 2: Find element by nomor polisi
40     cout << "\nMasukkan Nomor Polisi yang dicari: ";
41     string nopolCari;
42     cin >> nopolCari;
43     P = findElm(L, nopolCari);
44     if (P != NULL) {
45         cout << "Nomor Polisi : " << P->info.nopol << endl;
46         cout << "Warna       : " << P->info.warna << endl;
47         cout << "Tahun        : " << P->info.thnBuat << endl;
48     } else {
49         cout << "Data dengan nomor polisi " << nopolCari << " tidak ditemukan." << endl;
50     }
51
52     // Task 3: Delete element based on user input
53     cout << "\nMasukkan Nomor Polisi yang ingin dihapus: ";

```



```

53     cout << "\nMasukkan Nomor Polisi yang ingin dihapus: ";
54     string nopolHapus;
55     cin >> nopolHapus;
56
57     P = findElem(L, nopolHapus);
58     if (P != NULL) {
59         address deletedNode = NULL;
60
61         if (P == L.First) {
62             deleteFirst(L, deletedNode);
63         } else if (P == L.Last) {
64             deleteLast(L, deletedNode);
65         } else {
66             deleteAfter(L, P->prev, deletedNode);
67         }
68
69         dealokasi(deletedNode);
70         cout << "Data dengan nomor polisi " << nopolHapus << " berhasil dihapus." << endl;
71     } else {
72         cout << "Data dengan nomor polisi " << nopolHapus << " tidak ditemukan, sehingga tidak dapat dihapus." << endl;
73     }
74
75     // Display the list after deletion
76     printInfo(L);
77
78     return 0;
79 }

```

Penjelasan :

Program main.cpp ini adalah implementasi dari operasi-operasi dasar pada double linked list yang menyimpan informasi kendaraan. Program ini mengelola daftar kendaraan yang mencakup beberapa fitur, seperti menambah data kendaraan, mencari data berdasarkan nomor polisi, dan menghapus data tertentu. Berikut penjelasan dari setiap bagian fungsionalitas program secara rinci.

Inisialisasi dan Pengisian Data Awal

1. Inisialisasi List

Program dimulai dengan membuat list kosong menggunakan fungsi CreateList, yang mengatur pointer First dan Last ke NULL. Ini berarti bahwa list belum berisi elemen apapun pada tahap ini.

2. Menambahkan Data Kendaraan

Program meminta pengguna untuk memasukkan data kendaraan, yaitu nomor polisi, warna, dan tahun pembuatan. Sebelum data ditambahkan, program memeriksa apakah nomor polisi yang dimasukkan sudah ada di dalam list menggunakan fungsi isKendaraanExist. Jika sudah ada, pengguna diminta untuk memasukkan nomor polisi lain. Jika belum ada, data kendaraan tersebut dialokasikan menjadi elemen baru dalam list dan ditempatkan di akhir list dengan insertLast.

Program ini mengulangi proses penambahan hingga terdapat setidaknya tiga elemen di dalam list.

Menampilkan Daftar Kendaraan

Setelah beberapa data kendaraan dimasukkan, program menampilkan seluruh data yang ada di dalam list. Setiap elemen dalam list dicetak satu per satu, dimulai dari elemen pertama hingga terakhir, dan menampilkan informasi nomor polisi, warna, dan tahun pembuatan untuk setiap kendaraan.

Mencari Kendaraan Berdasarkan Nomor Polisi

Program kemudian meminta pengguna untuk memasukkan nomor polisi kendaraan yang ingin dicari. Fungsi `findElm` digunakan untuk mencari elemen dalam list yang memiliki nomor polisi tersebut. Jika elemen ditemukan, program menampilkan informasi kendaraan terkait. Jika elemen tidak ditemukan, program memberikan pesan bahwa data dengan nomor polisi yang dicari tidak ada di dalam list.

Menghapus Kendaraan Berdasarkan Nomor Polisi

Selanjutnya, program memberikan opsi kepada pengguna untuk menghapus data kendaraan berdasarkan nomor polisi. Pengguna memasukkan nomor polisi kendaraan yang ingin dihapus, dan program mencari elemen terkait dengan `findElm`. Jika elemen ditemukan:

- Jika elemen tersebut adalah elemen pertama dalam list, maka `deleteFirst` digunakan untuk menghapusnya.
- Jika elemen tersebut adalah elemen terakhir, `deleteLast` digunakan.
- Jika elemen berada di tengah, maka `deleteAfter` digunakan, yang menghapus elemen di posisi tertentu setelah elemen tertentu dalam list.

Setelah elemen berhasil dihapus, program membebaskan memori yang digunakan elemen tersebut dengan fungsi dealokasi. Jika elemen dengan nomor polisi yang dimasukkan tidak ditemukan, program memberikan pesan bahwa penghapusan tidak dapat dilakukan karena data tidak ditemukan.

Menampilkan Daftar Kendaraan Setelah Penghapusan

Terakhir, program menampilkan daftar kendaraan yang tersisa setelah operasi penghapusan. Semua data kendaraan yang tersisa dicetak kembali untuk memastikan bahwa elemen yang diminta telah dihapus dari list.

OUTPUT :

1. Implementasi ADT *Double Linked list* pada file “doublelist.cpp” hasil implementasi ADT pada file “main.cpp”.

```

"C:\Users\USER\Pictures\STD 6\unguided1\bin\Debug\unguided1.exe"
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: D004
masukkan warna kendaraan: kuning
masukkan tahun kendaraan: 90

DATA LIST 1
no polisi : D001
warna      : hitam
tahun      : 90
no polisi : D003
warna      : putih
tahun      : 70
no polisi : D004
warna      : kuning
tahun      : 90

```

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

```

"C:\Users\USER\Pictures\STD 6\unguided1\bin\Debug\unguided1.exe"
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: D004
masukkan warna kendaraan: kuning
masukkan tahun kendaraan: 90

DATA LIST 1
no polisi : D001
warna      : hitam
tahun      : 90
no polisi : D003
warna      : putih
tahun      : 70
no polisi : D004
warna      : kuning
tahun      : 90

Masukkan Nomor Polisi yang dicari: D001
Nomor Polisi : D001
Warna        : hitam
Tahun        : 90

```

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)`

```
"C:\Users\USER\Pictures\STD 6\unguided1\bin\Debug\unguided1.exe"

DATA LIST 1
no polisi : D001
warna     : hitam
tahun     : 90
no polisi : D003
warna     : putih
tahun     : 70
no polisi : D004
warna     : kuning
tahun     : 90

Masukkan Nomor Polisi yang dicari: D001
Nomor Polisi : D001
Warna        : hitam
Tahun        : 90

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

DATA LIST 1
no polisi : D001
warna     : hitam
tahun     : 90
no polisi : D004
warna     : kuning
tahun     : 90

Process returned 0 (0x0)   execution time : 65.522 s
Press any key to continue.
```

5. Kesimpulan

Praktikum ini membahas konsep dan implementasi Double Linked List, yaitu struktur data yang memungkinkan elemen-elemen saling terhubung dua arah menggunakan pointer next dan prev. Double Linked List lebih fleksibel dibandingkan Single Linked List karena mempermudah penambahan, penghapusan, dan pencarian data dari kedua arah. Pada struktur ini, head menunjuk ke elemen pertama dan tail menunjuk ke elemen terakhir, membuat navigasi data lebih efisien. Implementasi Double Linked List dalam praktikum ini mencakup beberapa fungsi utama, seperti:

1. Penambahan dan Penghapusan Elemen: Mendukung penambahan di awal, akhir, atau setelah elemen tertentu, serta penghapusan elemen pertama, terakhir, atau setelah elemen tertentu.
2. Pencarian dan Pengelolaan Memori: Memungkinkan pencarian elemen berdasarkan kriteria tertentu dan pengelolaan memori melalui fungsi alokasi dan dealokasi, yang penting untuk menghindari kebocoran memori.
3. Penggunaan Header File: Memisahkan deklarasi dan implementasi ke dalam file header (doublelist.h), yang menjaga modularitas dan kerapian kode.

