

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



**Disusun Oleh:**

Ahmad Al - Farizi - 2311104054

**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). Searching, 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

Source code berikut adalah implementasi double linked list dalam C++. Kelas `Node` menyimpan data dan dua pointer, `prev` dan `next`, yang menghubungkan node saat ini dengan node sebelumnya dan berikutnya. Kelas `DoublyLinkedList` memiliki pointer `head` untuk node pertama dan `tail` untuk node terakhir. Metode `insert` menambahkan node baru di depan list. Jika list kosong, `tail` diatur ke node baru. Metode `deleteNode` menghapus node pertama, dan jika list kosong setelah penghapusan, `tail` juga diatur ke `nullptr`. Metode `update` mencari data tertentu, dan jika ditemukan, menggantinya dengan data baru. Fungsi `deleteAll` menghapus semua node dalam list, mengatur ulang `head` dan `tail` menjadi `nullptr`. Metode `display` menampilkan semua data dari awal hingga akhir list. Dalam fungsi `main`, terdapat menu interaktif untuk menambahkan, menghapus, memperbarui, membersihkan, dan menampilkan data dalam list. Program berjalan terus hingga pengguna memilih opsi `Exit` untuk mengakhiri program.

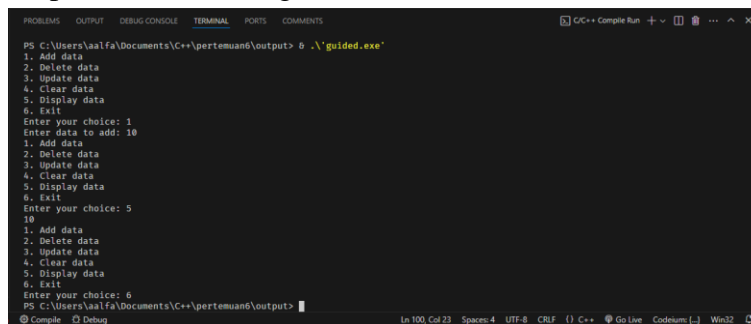
Kode Program :

```
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;
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     }
```

```
1 // Fungsi untuk mengupdate data di list
2 bool update(int oldData, int newData) {
3     Node* current = head;
4     while (current != nullptr) {
5         if (current->data == oldData) {
6             current->data = newData;
7             return true; // Jika data ditemukan dan diupdate
8         }
9         current = current->next;
10    }
11    return false; // Jika data tidak ditemukan
12 }
13
14 // Fungsi untuk menghapus semua elemen di list
15 void deleteAll() {
16     Node* current = head;
17     while (current != nullptr) {
18         Node* temp = current;
19         current = current->next;
20         delete temp;
21     }
22     head = nullptr;
23     tail = nullptr;
24 }
25
26 // Fungsi untuk menampilkan semua elemen di list
27 void display() {
28     Node* current = head;
29     while (current != nullptr) {
30         cout << current->data << " ";
31         current = current->next;
32     }
33     cout << endl;
34 }
35 };
36
37 int main() {
38     DoublyLinkedList list;
39     while (true) {
40         cout << "1. Add data" << endl;
41         cout << "2. Delete data" << endl;
42         cout << "3. Update data" << endl;
43         cout << "4. Clear data" << endl;
44         cout << "5. Display data" << endl;
45         cout << "6. Exit" << endl;
46
47         int choice;
48         cout << "Enter your choice: ";
49         cin >> choice;
50
51         switch (choice) {
52             case 1: {
53                 int data;
54                 cout << "Enter data to add: ";
55                 cin >> data;
56                 list.insert(data);
57                 break;
58             }
59             case 2: {
60                 list.deleteNode();
61                 break;
62             }
63             case 3: {
64                 int oldData, newData;
65                 cout << "Enter old data: ";
66                 cin >> oldData;
67                 cout << "Enter new data: ";
68                 cin >> newData;
69                 bool updated = list.update(oldData, newData);
70                 if (!updated) {
71                     cout << "Data not found" << endl;
72                 }
73                 break;
74             }
75         }
76     }
77 }
```

```
1      case 4: {
2          list.deleteAll();
3          break;
4      }
5      case 5: {
6          list.display();
7          break;
8      }
9      case 6: {
10         return 0;
11     }
12     default: {
13         cout << "Invalid choice" << endl;
14         break;
15     }
16 }
17 }
18 return 0;
19 }
```

Output dari Kode Program :



```
PS C:\Users\aa1fa\Documents\C++\pertemuan6\output> .\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: 6
PS C:\Users\aa1fa\Documents\C++\pertemuan6\output>
```

## 4. Unguided

### 4.1. Code Program:

#### doublelist.h

```
doublelist.h doublelist.h main.cpp X
1 #ifndef DOUBLELIST_H
2 #define DOUBLELIST_H
3
4 #include <iostream>
5 #include <string>
6 using namespace std;
7
8 struct infotype {
9     string nopol;
10    string warna;
11    int thnBuat;
12}
13
14 struct tLinkList;
15 typedef tLinkList* address;
16
17 struct tLinkList {
18     infotype info;
19     address next;
20     address prev;
21}
22
23 struct List {
24     address First;
25     address Last;
26}
27
28 void CreateList(List &L);
29 address alokasi(infotype x);
30 void dealokasi(address &P);
31 void printInfo(List L);
32 void insertFirst(List &L, address P);
33 address findMinList(L, infotype x);
34
35 #endif
```

#### doublelist.cpp

```
doublelist.cpp doublelist.cpp main.cpp X
1 #include "doublelist.h"
2
3 void CreateList(List &L) {
4     L.First = nullptr;
5     L.Last = nullptr;
6 }
7
8 address alokasi(infotype x) {
9     address P = new infotype;
10    P->nopol = x.nopol;
11    P->warna = x.warna;
12    P->thnBuat = x.thnBuat;
13    return P;
14}
15
16 void dealokasi(address &P) {
17     delete P;
18     P = nullptr;
19}
20
21 void printInfo(List L) {
22     address P = L.First;
23     while (P != nullptr) {
24         cout << "No Polisi : " << P->nopol << endl;
25         cout << "Warna : " << P->warna << endl;
26         cout << "Tahun : " << P->thnBuat << endl;
27         P = P->next;
28     }
29 }
30
31 void insertFirst(List &L, address P) {
32     if (L.First == nullptr) {
33         L.First = P;
34         L.Last = P;
35     } else {
36         P->next = L.First;
37         L.First->prev = P;
38         L.First = P;
39     }
40 }
41
42 address findMinList(L, infotype x) {
43     address P = L.First;
44     while (P != nullptr) {
45         if (P->nopol == x.nopol) {
46             return P;
47         }
48         P = P->next;
49     }
50     return nullptr;
51 }
52
```

#### main.cpp

```
doublelist.cpp doublelist.cpp main.cpp X
1 #include "doublelist.h"
2
3 int main() {
4     List L;
5     CreateList(L);
6
7     int jumlah;
8     cout << "Masukkan jumlah kendaraan yang akan diinput : ";
9     cin >> jumlah;
10
11     for (int i = 0; i < jumlah; i++) {
12         infotype kendaraan;
13         cout << "Masukkan nomor polisi : ";
14         cin >> kendaraan.nopol;
15         cout << "Masukkan warna kendaraan : ";
16         cin >> kendaraan.warna;
17         cout << "Masukkan tahun kendaraan : ";
18         cin >> kendaraan.thnBuat;
19
20         // Cek apakah nomor polisi sudah terdaftar
21         if (findMinList(L, kendaraan) == nullptr) {
22             insertFirst(L, alokasi(kendaraan)); // Memasukkan insertFirst
23         } else {
24             cout << "Nomor polisi sudah terdaftar!\n";
25         }
26     }
27
28     // Menampilkan data list setelah input selesai
29     cout << "List DATA LIST :\n";
30     printInfo(L);
31
32     return 0;
33 }
34
```

### Output dari Code Program:

```
C:\Users\asfa\Documents>C++\Unguided Modul 4\Unguided Modul 4.exe
Masukkan jumlah kendaraan yang akan diinput: 4
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
List DATA LIST 1
No Polisi : D004
Warna : kuning
Tahun : 90
No Polisi : D003
Warna : putih
Tahun : 70
No Polisi : D001
Warna : hitam
Tahun : 90
```

## 4.2. Kode Program: doublelist.h

```
doublelist.h doublelist.h main.cpp X
1 #ifndef DOUBLELIST_H
2 #define DOUBLELIST_H
3
4 #include <iostream>
5 #include <string>
6 using namespace std;
7
8 struct infoType {
9     string nopol;
10    string warna;
11    int thdBus;
12};
13
14 struct kmlist;
15 typedef struct kmlist address;
16
17 struct kmlist {
18     infoType info;
19     address next;
20     address prev;
21};
22
23 struct list {
24     address first;
25     address last;
26};
27
28 void CreateList(list &l);
29 void alokasi(infoType &i);
30 void deakasi(address &a);
31 void printInfo(list l);
32 void insertFirst(list &l, address &a); // Menambah insertLast dan insertFirst
33 address findMin(list l, infoType &i);
34
35 #endif
```

## doublelist.cpp

```
doublelist.cpp doublelist.cpp main.cpp X
1 #include "doublelist.h"
2
3 list l;
4 int main() {
5     CreateList(l);
6
7     // Data yang sudah dimasukkan sebelumnya
8     infoType benderaan = {"2001", "hijau", 90};
9     infoType benderaan2 = {"2002", "hijau", 90};
10    infoType benderaan3 = {"2003", "hijau", 90};
11
12    insertFirst(l, alokasi(benderaan));
13    insertFirst(l, alokasi(benderaan2));
14    insertFirst(l, alokasi(benderaan3));
15
16    // Menampilkan data list awal
17    cout << "DATA LIST AWAL" << endl;
18    printInfo(l);
19
20    // Input nomor polisi yang akan dicari
21    infoType benderaanCari;
22    cout << "Masukkan Nomor Polisi yang dicari: ";
23    cin >> benderaanCari.nopol;
24
25    // Cari data berdasarkan nomor polisi
26    address hasilCari = findMin(l, benderaanCari);
27
28    if (hasilCari != nullptr) {
29        cout << "No Polisi : " << hasilCari->info.nopol << endl;
30        cout << "Warna : " << hasilCari->info.warna << endl;
31        cout << "Tahun : " << hasilCari->info.thdBus << endl;
32    } else {
33        cout << "Tidak ada data polisi " << benderaanCari.nopol << " yang ditemukan." << endl;
34    }
35
36    return 0;
37}
```

## main.cpp

```
doublelist.cpp doublelist.cpp main.cpp X
1 #include "doublelist.h"
2
3 list l;
4 int main() {
5     CreateList(l);
6
7     // Data yang sudah dimasukkan sebelumnya
8     infoType benderaan = {"2001", "hijau", 90};
9     infoType benderaan2 = {"2002", "hijau", 90};
10    infoType benderaan3 = {"2003", "hijau", 90};
11
12    insertFirst(l, alokasi(benderaan));
13    insertFirst(l, alokasi(benderaan2));
14    insertFirst(l, alokasi(benderaan3));
15
16    // Menampilkan data list awal
17    cout << "DATA LIST AWAL" << endl;
18    printInfo(l);
19
20    // Input nomor polisi yang akan dicari
21    infoType benderaanCari;
22    cout << "Masukkan Nomor Polisi yang dicari: ";
23    cin >> benderaanCari.nopol;
24
25    // Cari data berdasarkan nomor polisi
26    address hasilCari = findMin(l, benderaanCari);
27
28    if (hasilCari != nullptr) {
29        cout << "No Polisi : " << hasilCari->info.nopol << endl;
30        cout << "Warna : " << hasilCari->info.warna << endl;
31        cout << "Tahun : " << hasilCari->info.thdBus << endl;
32    } else {
33        cout << "Tidak ada data polisi " << benderaanCari.nopol << " yang ditemukan." << endl;
34    }
35
36    return 0;
37}
```

## Ouput dari Kode Program:

```
DATA LIST 1
No Polisi : 0004
Warna : kuning
Tahun : 90
No Polisi : 0003
Warna : putih
Tahun : 70
No Polisi : 0001
Warna : hitam
Tahun : 90

Masukkan Nomor Polisi yang dicari: 0001
No Polisi : 0001
Warna : hitam
Tahun : 90
```



### 4.3. Kode Program: doublelist.h

```
doublelist.h X doublelist.h X main.cpp X
1 #ifndef DOUBLELIST_H
2 #define DOUBLELIST_H
3
4 #include <iostream>
5 #include <string>
6 using namespace std;
7
8 struct infoType {
9     string nama;
10    string alamat;
11    int umur;
12};
13
14 struct Node {
15    infoType info;
16    Node* next;
17};
18
19 struct List {
20    Node* first;
21    Node* last;
22};
23
24 void createList(List &l);
25 void deleteList(List &l, infoType i);
26 void insertList(List &l, infoType i);
27 void deleteFirstList(List &l);
28 void deleteLastList(List &l);
29 void deleteAllList(List &l);
30 void printList(List &l);
31
32#endif
```

### doublelist.cpp

```
doublelist.cpp X doublelist.cpp X main.cpp X
1 #include "doublelist.h"
2
3 void createList(List &l) {
4     l.first = nullptr;
5     l.last = nullptr;
6 }
7
8 void deleteList(List &l, infoType i) {
9     Node* p = new Node;
10    p->info = i;
11    p->next = nullptr;
12    return p;
13}
14
15 void deleteFirstList(List &l) {
16     if (l.first == nullptr) return;
17     l.first = l.first->next;
18 }
19
20 void deleteLastList(List &l) {
21     if (l.first == nullptr) return;
22     Node* p = l.first;
23     while (p->next != nullptr) {
24         p = p->next;
25     }
26     p->next = nullptr;
27 }
28
29 void deleteAllList(List &l) {
30     if (l.first == nullptr) return;
31     Node* p = l.first;
32     while (p != nullptr) {
33         delete p;
34         p = p->next;
35     }
36     l.first = nullptr;
37     l.last = nullptr;
38 }
39
40 void printList(List &l) {
41     if (l.first == nullptr) return;
42     Node* p = l.first;
43     while (p != nullptr) {
44         cout << "Nama: " << p->info.nama << endl;
45         cout << "Alamat: " << p->info.alamat << endl;
46         cout << "Umur: " << p->info.umur << endl;
47         p = p->next;
48     }
49 }
50
51 void insertList(List &l, infoType i) {
52     if (l.first == nullptr) {
53         l.first = new Node(i);
54         l.last = l.first;
55     } else {
56         l.last->next = new Node(i);
57         l.last = l.last->next;
58     }
59 }
60
61 void deleteList(List &l, infoType i) {
62     if (l.first == nullptr) return;
63     Node* p = l.first;
64     while (p != nullptr) {
65         if (p->info == i) {
66             if (p == l.first) {
67                 deleteFirstList(l);
68             } else {
69                 Node* prev = nullptr;
70                 while (prev->next != p) {
71                     prev = prev->next;
72                 }
73                 prev->next = p->next;
74                 delete p;
75             }
76         }
77         p = p->next;
78     }
79 }
80
81 void deleteFirstList(List &l) {
82     if (l.first == nullptr) return;
83     l.first = l.first->next;
84 }
85
86 void deleteLastList(List &l) {
87     if (l.first == nullptr) return;
88     Node* p = l.first;
89     while (p->next != nullptr) {
90         p = p->next;
91     }
92     p->next = nullptr;
93 }
94
95 void deleteAllList(List &l) {
96     if (l.first == nullptr) return;
97     Node* p = l.first;
98     while (p != nullptr) {
99         delete p;
100        p = p->next;
101    }
102    l.first = nullptr;
103    l.last = nullptr;
104}
```

## main.cpp

```

1  #include "doubleList.h"
2
3  int main() {
4      List L;
5      CreateList(L);
6
7      infoType kenderaaan = {"2001", "Bismar", 90};
8      infoType kenderaan2 = {"2002", "Bismar", 70};
9      infoType kenderaan3 = {"2003", "Bismar", 80};
10
11      insertFirst(L, alokasi(kenderaaan));
12      insertFirst(L, alokasi(kenderaan2));
13      insertFirst(L, alokasi(kenderaan3));
14
15      cout << "DATA LIST 1st\n";
16      printInfo(L);
17
18      infoType kenderaanhapus;
19      cout << "Masukkan Nomor Polisi yang akan dihapus: ";
20      cin >> kenderaanhapus.noPolisi;
21
22      address P = findIn(L, kenderaanhapus);
23
24      if (P == nullptr) {
25          cout << "Data tidak ditemukan\n";
26      } else if (P == L.First) {
27          deleteLast(L, P);
28      } else if (P == L.Last) {
29          deleteLast(L, P);
30      } else {
31          address Prev = P->prev;
32          address Next = P->next;
33          Prev->next = Next;
34          Next->prev = Prev;
35          cout << "Data dengan nomor polisi " << kenderaanhapus.noPolisi << " berhasil dihapus.\n";
36      }
37      cout << "Data dengan nomor polisi " << kenderaanhapus.noPolisi << " tidak ditemukan.\n";
38      printInfo(L);
39
40      return 0;
41  }

```

## Output dari Kode Program:

```

DATA LIST 1
No Polisi : 0004
nama      : kuning
tahun     : 90
No Polisi : 0003
nama      : putih
tahun     : 90
No Polisi : 0001
nama      : hitam
tahun     : 90

Masukkan Nomor Polisi yang akan dihapus: 0003
Data dengan nomor polisi 0003 berhasil dihapus.

DATA LIST 1
No Polisi : 0004
nama      : kuning
tahun     : 90
No Polisi : 0001
nama      : hitam
tahun     : 90

```

## **5. Kesimpulan**

Double linked list adalah struktur data dinamis yang memungkinkan navigasi dua arah antar elemen melalui dua penunjuk, yaitu prev dan next, serta memiliki pointer first dan last untuk mengakses elemen pertama dan terakhir. Double linked list mendukung berbagai operasi utama, seperti penyisipan (insert), penghapusan (delete), pembaruan data (update), penampilan data (view), dan pencarian data (searching). Dengan kemampuannya untuk menambahkan, menghapus, dan mencari elemen dari kedua arah, double linked list menawarkan fleksibilitas yang lebih tinggi dibandingkan single linked list, sehingga cocok untuk aplikasi yang memerlukan manipulasi data dua arah

