LAPORAN PRAKTIKUM
Tugas Pendahuluan Modul 06
“Double Linked List”

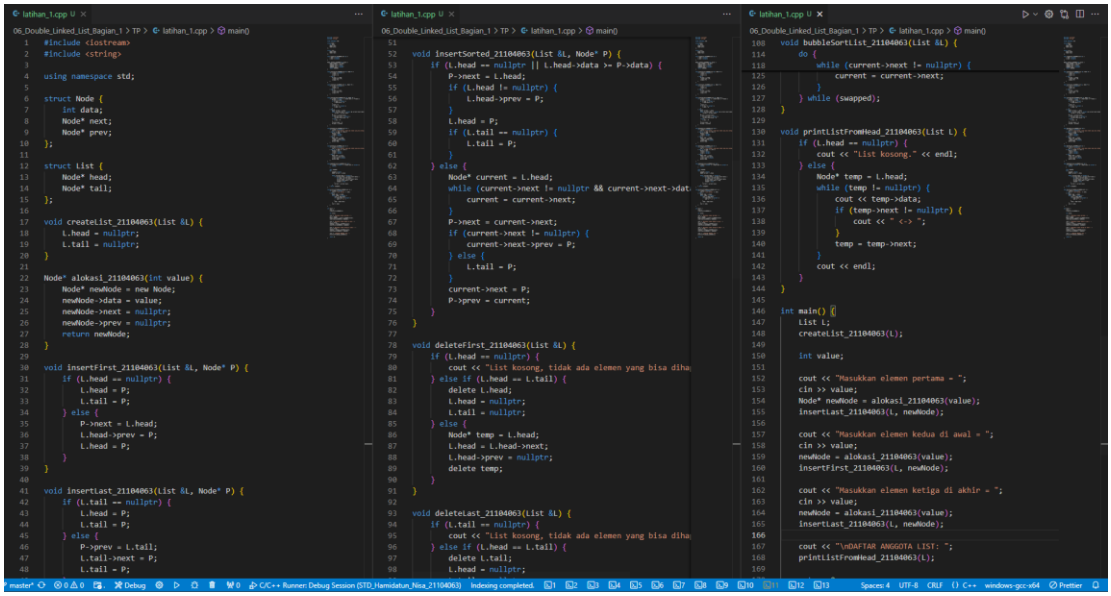
Disusun Oleh:
Hamidatun Nisa - 21104063
Struktur Data SE07-01

Dosen :
Yudha Islami Sulistya

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM PURWOKERTO
2024

1. Menambahkan Elemen di Awal dan Akhir DLL

   Code :

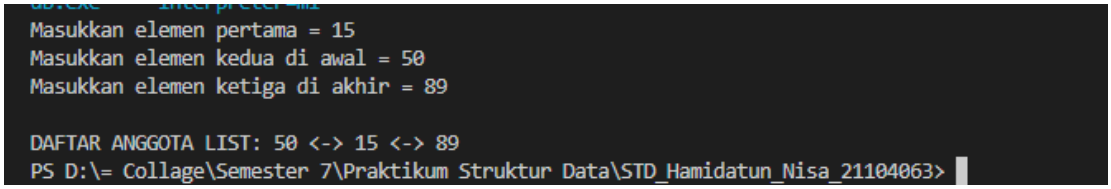


```
#include <iostream>
#include <string>

using namespace std;

struct Node {
    int data;
    Node* next;
    Node* prev;
};

struct List {
    Node* head;
    Node* tail;
};

void createList_21104063(List &L) {
    L.head = nullptr;
    L.tail = nullptr;
}

Node* alokasi_21104063(int value) {
    Node* newNode = new Node;
    newNode->data = value;
    newNode->next = nullptr;
    newNode->prev = nullptr;
    return newNode;
}

void insertFirst_21104063(List &L, Node* P) {
    if (L.head == nullptr) {
        L.head = P;
        L.tail = P;
    } else {
        P->next = L.head;
        L.head->prev = P;
        L.head = P;
    }
}

void insertLast_21104063(List &L, Node* P) {
    if (L.tail == nullptr) {
        L.head = P;
        L.tail = P;
    } else {
        P->prev = L.tail;
        L.tail->next = P;
        L.tail = P;
```

```
}

void insertSorted_21104063(List &L, Node* P) {
    if (L.head == nullptr || L.head->data >= P->data) {
        P->next = L.head;
        if (L.head != nullptr) {
            L.head->prev = P;
        }
        L.head = P;
        if (L.tail == nullptr) {
            L.tail = P;
        }
    } else {
        Node* current = L.head;
        while (current->next != nullptr && current->next->dat
            current = current->next;
        }
        P->next = current->next;
        if (current->next != nullptr) {
            current->next->prev = P;
        } else {
            L.tail = P;
        }
        current->next = P;
        P->prev = current;
    }
}

void deleteFirst_21104063(List &L) {
    if (L.head == nullptr) {
        cout << "List kosong, tidak ada elemen yang bisa diha
    } else if (L.head == L.tail) {
        delete L.head;
        L.head = nullptr;
        L.tail = nullptr;
    } else {
        Node* temp = L.head;
        L.head = L.head->next;
        L.head->prev = nullptr;
        delete temp;
    }
}

void deleteLast_21104063(List &L) {
    if (L.tail == nullptr) {
        cout << "List kosong, tidak ada elemen yang bisa diha
    } else if (L.head == L.tail) {
        delete L.tail;
        L.head = nullptr;
```

```
void bubbleSortList_21104063(List &L) {
    do {
        while (current->next != nullptr) {
            current = current->next;
        }
    } while (swapped);
}

void printListFromHead_21104063(List L) {
    if (L.head == nullptr) {
        cout << "List kosong." << endl;
    } else {
        Node* temp = L.head;
        while (temp != nullptr) {
            cout << temp->data;
            if (temp->next != nullptr) {
                cout << " <-> ";
            }
            temp = temp->next;
        }
        cout << endl;
    }
}

int main() {
    List L;
    createList_21104063(L);

    int value;

    cout << "Masukkan elemen pertama = ";
    cin >> value;
    Node* newNode = alokasi_21104063(value);
    insertLast_21104063(L, newNode);

    cout << "Masukkan elemen kedua di awal = ";
    cin >> value;
    newNode = alokasi_21104063(value);
    insertFirst_21104063(L, newNode);

    cout << "Masukkan elemen ketiga di akhir = ";
    cin >> value;
    newNode = alokasi_21104063(value);
    insertLast_21104063(L, newNode);

    cout << "\nDAFTAR ANGGOTA LIST: ";
    printListFromHead_21104063(L);
```

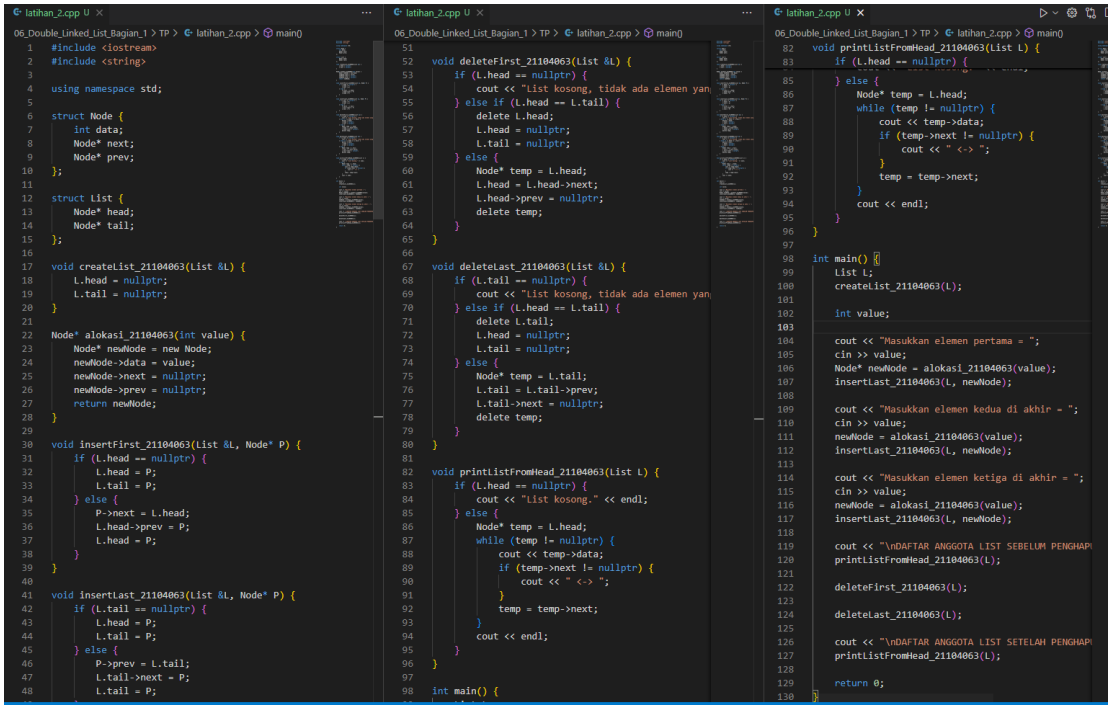
   Output :



```
Masukkan elemen pertama = 15
Masukkan elemen kedua di awal = 50
Masukkan elemen ketiga di akhir = 89

DAFTAR ANGGOTA LIST: 50 <-> 15 <-> 89
PS D:\= Collage\Semester 7\Praktikum Struktur Data\STD_Hamidatun_Nisa_21104063>
```

2. Menghapus Elemen di Awal dan Akhir DLL

Code :



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

using namespace std;

struct Node {
    int data;
    Node* next;
    Node* prev;
};

struct List {
    Node* head;
    Node* tail;
};

void createList_21104063(List &L) {
    L.head = nullptr;
    L.tail = nullptr;
}

Node* alokasi_21104063(int value) {
    Node* newNode = new Node;
    newNode->data = value;
    newNode->next = nullptr;
    newNode->prev = nullptr;
    return newNode;
}

void insertFirst_21104063(List &L, Node* P) {
    if (L.head == nullptr) {
        L.head = P;
        L.tail = P;
    } else {
        P->next = L.head;
        L.head->prev = P;
        L.head = P;
    }
}

void insertLast_21104063(List &L, Node* P) {
    if (L.tail == nullptr) {
        L.head = P;
        L.tail = P;
    } else {
        P->prev = L.tail;
        L.tail->next = P;
        L.tail = P;
```

```cpp
void deleteFirst_21104063(List &L) {
    if (L.head == nullptr) {
        cout << "List kosong, tidak ada elemen yan
    } else if (L.head == L.tail) {
        delete L.head;
        L.head = nullptr;
        L.tail = nullptr;
    } else {
        Node* temp = L.head;
        L.head = L.head->next;
        L.head->prev = nullptr;
        delete temp;
    }
}

void deleteLast_21104063(List &L) {
    if (L.tail == nullptr) {
        cout << "List kosong, tidak ada elemen yan
    } else if (L.head == L.tail) {
        delete L.tail;
        L.head = nullptr;
        L.tail = nullptr;
    } else {
        Node* temp = L.tail;
        L.tail = L.tail->prev;
        L.tail->next = nullptr;
        delete temp;
    }
}

void printListFromHead_21104063(List L) {
    if (L.head == nullptr) {
        cout << "List kosong." << endl;
    } else {
        Node* temp = L.head;
        while (temp != nullptr) {
            cout << temp->data;
            if (temp->next != nullptr) {
                cout << " <-> ";
            }
            temp = temp->next;
        }
        cout << endl;
    }
}

int main() {
```

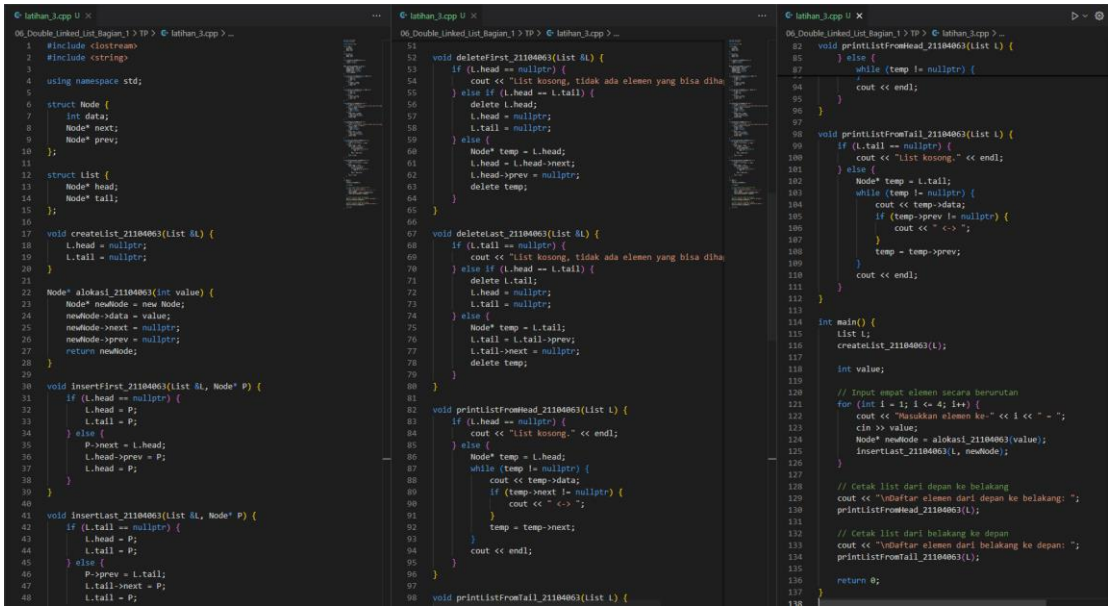
```cpp
void printListFromHead_21104063(List L) {
    if (L.head == nullptr) {
    } else {
        Node* temp = L.head;
        while (temp != nullptr) {
            cout << temp->data;
            if (temp->next != nullptr) {
                cout << " <-> ";
            }
            temp = temp->next;
        }
        cout << endl;
    }
}

int main() {
    List L;
    createList_21104063(L);

    int value;

    cout << "Masukkan elemen pertama = ";
    cin >> value;
    Node* newNode = alokasi_21104063(value);
    insertLast_21104063(L, newNode);

    cout << "Masukkan elemen kedua di akhir = ";
    cin >> value;
    newNode = alokasi_21104063(value);
    insertLast_21104063(L, newNode);

    cout << "Masukkan elemen ketiga di akhir = ";
    cin >> value;
    newNode = alokasi_21104063(value);
    insertLast_21104063(L, newNode);

    cout << "\nDAFTAR ANGGOTA LIST SEBELUM PENGHAP
    printListFromHead_21104063(L);

    deleteFirst_21104063(L);

    deleteLast_21104063(L);

    cout << "\nDAFTAR ANGGOTA LIST SETELAH PENGHAP
    printListFromHead_21104063(L);

    return 0;
}
```

Output :

```
Masukkan elemen pertama = 90
Masukkan elemen kedua di awal = 12
Masukkan elemen ketiga di akhir = 17

DAFTAR ANGGOTA LIST SEBELUM PENGHAPUSAN: 12 <-> 90 <-> 17

DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: 90
PS D:\= Collage\Semester 7\Praktikum Struktur Data\STD_Hamidatun_Nisa_21104063>
```

3. Menampilkan Elemen dari Depan ke Belakang dan Sebaliknya

Code :



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

using namespace std;

struct Node {
    int data;
    Node* next;
    Node* prev;
};

struct List {
    Node* head;
    Node* tail;
};

void createList_21104063(List &L) {
    L.head = nullptr;
    L.tail = nullptr;
}

Node* alokasi_21104063(int value) {
    Node* newNode = new Node;
    newNode->data = value;
    newNode->next = nullptr;
    newNode->prev = nullptr;
    return newNode;
}

void insertFirst_21104063(List &L, Node* P) {
    if (L.head == nullptr) {
        L.head = P;
        L.tail = P;
    } else {
        P->next = L.head;
        L.head->prev = P;
        L.head = P;
    }
}

void insertLast_21104063(List &L, Node* P) {
    if (L.tail == nullptr) {
        L.head = P;
        L.tail = P;
    } else {
        P->prev = L.tail;
        L.tail->next = P;
        L.tail = P;
    }
}

void deleteFirst_21104063(List &L) {
    if (L.head == nullptr) {
        cout << "List kosong, tidak ada elemen yang bisa dihapus." << endl;
    } else if (L.head == L.tail) {
        delete L.head;
        L.head = nullptr;
        L.tail = nullptr;
    } else {
        Node* temp = L.head;
        L.head = L.head->next;
        L.head->prev = nullptr;
        delete temp;
    }
}

void deleteLast_21104063(List &L) {
    if (L.tail == nullptr) {
        cout << "List kosong, tidak ada elemen yang bisa dihapus." << endl;
    } else if (L.head == L.tail) {
        delete L.tail;
        L.head = nullptr;
        L.tail = nullptr;
    } else {
        Node* temp = L.tail;
        L.tail = L.tail->prev;
        L.tail->next = nullptr;
        delete temp;
    }
}

void printListFromHead_21104063(List L) {
    if (L.head == nullptr) {
        cout << "List kosong." << endl;
    } else {
        Node* temp = L.head;
        while (temp != nullptr) {
            cout << temp->data;
            if (temp->next != nullptr) {
                cout << " <-> ";
            }
            temp = temp->next;
        }
        cout << endl;
    }
}

void printListFromTail_21104063(List L) {
    if (L.tail == nullptr) {
        cout << "List kosong." << endl;
    } else {
        Node* temp = L.tail;
        while (temp != nullptr) {
            cout << temp->data;
            if (temp->prev != nullptr) {
                cout << " <-> ";
            }
            temp = temp->prev;
        }
        cout << endl;
    }
}

int main() {
    List L;
    createList_21104063(L);

    int value;

    // Input empat elemen secara berurutan
    for (int i = 1; i <= 4; i++) {
        cout << "Masukkan elemen ke-" << i << " = ";
        cin >> value;
        Node* newNode = alokasi_21104063(value);
        insertLast_21104063(L, newNode);
    }

    // Cetak list dari depan ke belakang
    cout << "\nDaftar elemen dari depan ke belakang: ";
    printListFromHead_21104063(L);

    // Cetak list dari belakang ke depan
    cout << "\nDaftar elemen dari belakang ke depan: ";
    printListFromTail_21104063(L);

    return 0;
}
```

Output :

```
Masukkan elemen ke-1 = 90
Masukkan elemen ke-2 = 54
Masukkan elemen ke-3 = 35
Masukkan elemen ke-4 = 75

Daftar elemen dari depan ke belakang: 90 <-> 54 <-> 35 <-> 75

Daftar elemen dari belakang ke depan: 75 <-> 35 <-> 54 <-> 90
PS D:\= Collage\Semester 7\Praktikum Struktur Data\STD_Hamidatun_Nisa_21104063>
```