LAPORAN PRAKTIKUM
MODUL 6

DOUBLE LINKED LIST BAGIAN 1

Disusun Oleh:

Prajna paramitha - 2311104016

SE 07 01

Dosen :

Yudha Islami Sulistya, S.Kom., M.Cs

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2024

SOAL TP

Soal 1: Menambahkan Elemen di Awal dan Akhir DLL
Deskripsi Soal:
Buatlah program yang mengizinkan pengguna menambahkan elemen ke dalam Doubly Linked List di awal dan di akhir list.
Instruksi:
1. Implementasikan fungsi `insertFirst` untuk menambahkan elemen di awal list.
2. Implementasikan fungsi `insertLast` untuk menambahkan elemen di akhir list.
3. Tampilkan seluruh elemen dalam list dari depan ke belakang setelah penambahan dilakukan.
Contoh Input:
- Input: Masukkan elemen pertama = 10
- Input: Masukkan elemen kedua di awal = 5
- Input: Masukkan elemen ketiga di akhir = 20
Output:
- DAFTAR ANGGOTA LIST: 5 <-> 10 <-> 20

```cpp
// insertfirst -> di awal
// insertlast -> diakhir
// nampilin semua elemen dalam list dari depan sampai belakang
# include <iostream>

using namespace std;

// Struktur untuk node dalam double linked list
struct Node {
    int data;
    Node* next; // Pointer ke node berikutnya
    Node* prev; // Pointer ke node sebelumnya
    // ini sesuai dengan struktur dari DLL dia kan ada 3 bagian:
    // 1. depan(next) 2. info 3. blkg(prev)
};

// Struktur untuk double linked list
struct DoubleList {
    Node* head;             // Pointer ke node pertama
    Node* tail;             // Pointer ke node terakhir
};

// Fungsi untuk membuat list kosong
void createList_2311104016(DoubleList &L) {
    L.head = NULL;
    L.tail = NULL;
}

// Fungsi untuk membuat node baru
Node* createNode_2311104016(int angka) {
    Node* newNode = new Node();    // Alokasi memori untuk node baru
    newNode->data = angka;
    newNode->next = NULL;
    // istilah dia node pertama jadi blm ada prev sma next
    newNode->prev = NULL;
    return newNode;
}
void insertFirst_2311104016(DoubleList &L, Node* newNode){
    //jadi node baru akan menjadi prev dri node yang head(node pertama sebelumnya)

    if(L.head == NULL) {
        L.head = newNode;
        L.tail = newNode;
    } else {
        newNode->next = L.head;
        L.head->prev = newNode;
        L.head = newNode;
    }
}
```

Compiled successfully!

```cpp
void insertLast_2311104016(DoubleList &L, Node* newNode) {

    if (L.head == NULL) {
        L.head = newNode;
        L.tail = newNode;
    } else {
        newNode->prev = L.tail;
        L.tail->next = newNode;
        L.tail = newNode;
    }
}

void display_2311104016(DoubleList L) {
    Node* current = L.head;
    cout << "DAFTAR ANGGOTA LIST: ";
    while (current != NULL) {
        cout << current->data;
        if (current->next != NULL) {
            cout << " <-> ";
        }
        current = current->next;
    }
    cout << endl;
}

int main() {
    DoubleList list;
    createList_2311104016(list);

    int elem1, elem2, elem3;

    // Input sesuai contoh
    cout << "Input: Masukkan elemen pertama = ";
    cin >> elem1;

    cout << "Input: Masukkan elemen kedua di awal = ";
    cin >> elem2;

    cout << "Input: Masukkan elemen ketiga di akhir = ";
    cin >> elem3;

    // Membuat node baru dan menambahkan ke list
    Node* node1 = createNode_2311104016(elem1);
    Node* node2 = createNode_2311104016(elem2);
    Node* node3 = createNode_2311104016(elem3);
```

```cpp
// Menambahkan elemen sesuai urutan
insertFirst_2311104016(list, node1);  // Elemen pertama
insertFirst_2311104016(list, node2);  // Elemen kedua di awal
insertLast_2311104016(list, node3);   // Elemen ketiga di akhir

// Menampilkan hasil
display_2311104016(list);

return 0;
```

Compiled successfully!

```
Input: Masukkan elemen pertama = 5
Input: Masukkan elemen kedua di awal = 10
Input: Masukkan elemen ketiga di akhir = 20
DAFTAR ANGGOTA LIST: 10 <-> 5 <-> 20
PS C:\Users\ZBooK\OneDrive\Dokumen\Struktur data>
```

Penjelasan :

- Struct Node : Struktur node dalam double linked listnya. Jadi DLL itu punya 3 bagian yaitu prev data next. Untuk data di setting integer var data
- Untuk struct node pointer next dan prev ditulis Node* prev dan Node* next.
- Struct DoubleList : struktut untuk double linked list, jadi DLL ada head dan tail.
- Membuat fungsi create list dengan param double linkedlist. Membuat create diawali kosong jadi head dan tailnya NULL.
- Membuat fungsi create node, jadi pertama membuat objek nodenya. Setelah itu newNode objeknya akan diisi dengan angka (artinya data dari nodenya yang baru dibuat akan diisi dengan angka). Untuk next dan prev diisi null.
- Membuat fungsi insertfirst jadi memasukkan node dibaris paling pertama. Disini terjadi pengecekan kalau head nya kosong artinya node ini node pertama maka head dan tailnya newNode. Tapi jika headnya tidak kosong maka next dari newNodenya akan diubah menjadi headnya dan preview dari headnya menjadi newNode jadi newNode didepan head yang sebelumnya.
- Membuat fungsi insertlast untuk menambahkan diakhir. Jadi sama ada pengecekan seperti insertfirst. Konsepnya sama dengan insertfirst.
- Membuat fungsi display untuk menampilkan isi dari DLL nya.
- Membuat fungsi utama yaitu main yang akan dijalnkan pertama kali.
- Memanggil doublelinked list dan function createList yg sebelumnya telah dibuat.
- Lalu menerima inputan dari user yang disimpan masing” di elm1, elm2, elm3 dan akan memanggil fungsi create dan insert sesuai dengan paramnya. Dibaris ini:

```
Node* node1 = createNode_2311104016(elem1);
   Node* node2 = createNode_2311104016(elem2);
   Node* node3 = createNode_2311104016(elem3);

   // Menambahkan elemen sesuai urutan
   insertFirst_2311104016(list, node1);  // Elemen pertama
   insertFirst_2311104016(list, node2);  // Elemen kedua di awal
   insertLast_2311104016(list, node3);   // Elemen ketiga di akhir
```

lalu terakhir menampilkan dengan memanggil method display

Soal 2: Menghapus Elemen di Awal dan Akhir DLL

Deskripsi Soal:

Buatlah program yang memungkinkan pengguna untuk menghapus elemen pertama dan elemen terakhir dalam Doubly Linked List.

```cpp
# include <iostream>
using namespace std;

struct Node
{
    // membuat nodenya dulu, jadi DLL ada 3 bag data, prev sama next
    int data;
    Node* prev; //pointer prev elemen
    Node* next; //pointer next elemen
};

struct DoubleList
{
    // address
    Node* head;
    Node* tail;
};

//soal 2 menghapus elemen diawal dan diakhir
// delete first, delete last, menampilkan

// Fungsi untuk membuat list kosong
void createList_2311104016(DoubleList &L) {
    L.head = NULL;
    L.tail = NULL;
}

// Fungsi untuk membuat node baru
Node* createNode_2311104016(int angka) {
    Node* newNode = new Node();    // Alokasi memori untuk node baru
    newNode->data = angka;
    newNode->next = NULL;
    // istilah dia node pertama jadi blm ada prev sma next
    newNode->prev = NULL;
    return newNode;
}
void insertFirst_2311104016(DoubleList &L, Node* newNode){
    //jadi node baru akan menjadi prev dri node yang head(node pertama sebelumnya)

    if(L.head == NULL) {
        L.head = newNode;
        L.tail = newNode;
    } else {
        newNode->next = L.head;
        L.head->prev = newNode;
        L.head = newNode;
    }
}
// Fungsi untuk menambahkan buku di akhir list
void insertLast_2311104016(DoubleList &L, Node* newNode) {
    // address P == Node* newNode
    if (L.head == NULL) {
        L.head = newNode;
        L.tail = newNode;
    } else {
        newNode->prev = L.tail;
        L.tail->next = newNode;
        L.tail = newNode;
    }
}
```

```cpp
void DeleteFirst_2311104016(DoubleList &L) {
    if(L.head == NULL && L.tail == NULL) {
        cout << "Maaf cantik listnya kosong nih.." << endl;
    } else if (L.head == L.tail) {
        // klo node yang kita apus itu node satu satunya
        delete L.head;
        L.head = NULL;
        L.tail = NULL;
    } else {
        Node* temp = L.head;
        L.head = L.head->next;
        L.head->prev = NULL;
        delete temp;
    }
}

void DeleteLast_2311104016(DoubleList &L){
    if(L.head == NULL && L.tail == NULL) {
        cout << "Maaf cantik listnya kosong nih.." << endl;
    } else if (L.head == L.tail) {
        delete L.tail;
        L.head = NULL;
        L.tail = NULL;
    } else {
        Node* temp = L.tail;
        L.tail = L.tail->prev;
        L.tail->next = NULL;
        delete temp;
    }
}
void display_2311104016(DoubleList L) {
    Node* current = L.head;
    cout << "DAFTAR ANGGOTA LIST: ";
    while (current != NULL) {
        cout << current->data;
        if (current->next != NULL) {
            cout << " <-> ";
        }
        current = current->next;
    }
    cout << endl;
}

int main() {
    DoubleList list;
    createList_2311104016(list);
    int elem1, elem2, elem3;

    // Input sesuai contoh
    cout << "Input: Masukkan elemen pertama = ";
    cin >> elem1;

    cout << "Input: Masukkan elemen kedua di awal = ";
    cin >> elem2;

    cout << "Input: Masukkan elemen ketiga di akhir = ";
    cin >> elem3;
```

```cpp
    // Membuat node baru dan menambahkan ke list
    Node* node1 = createNode_2311104016(elem1);
    Node* node2 = createNode_2311104016(elem2);
    Node* node3 = createNode_2311104016(elem3);

    // Menambahkan elemen sesuai urutan
    insertFirst_2311104016(list, node1);  // Elemen pertama
    insertLast_2311104016(list, node2);  // Elemen kedua di awal
    insertLast_2311104016(list, node3);   // Elemen ketiga di akhir
    DeleteFirst_2311104016(list);
    DeleteLast_2311104016(list);

    // Menampilkan hasil
    display_2311104016(list);

    return 0;
}
```

```
Input: Masukkan elemen pertama = 10
Input: Masukkan elemen kedua di awal = 15
Input: Masukkan elemen ketiga di akhir = 20
DAFTAR ANGGOTA LIST: 15
PS C:\Users\ZBooK\OneDrive\Dokumen\Struktur data>
```

Penjelasan :

- Struct node : struktur nodenya berisikan data, pointer prev dan pointer next
- Struct doublelist : membuat struktur double listnya
- Membuat method createList dengan set L.head dan L.tailnya NULL
- Membuat methoc createNode, dan newNode datanya akan diisi oleh param angka dan pointer next prev null mengibaratkan merupakan node pertama
- Membuat method insertfirst untuk menambahkan node baru didepan dengan param doublelist, jadi jika headnya kosong maka newnode akan menjadi node pertama namun jika tidak maka aka nada perubahan pointer prev dan nextnya dan menjadikan node yang baru ditambahkan menjadi headnya
- Membuat method insertlast : kurang lebih logicnya sama seperti insertfirst hanya saja ada perbedaan di pointer prev dan nextnya.
- Membuat method delete first untuk menghapus node yang didepan, pertama dilakukan pengecekan jika head dan tailnya kosong maka tidak ada data yang dihapus. Namun jika node tersebut merupakan satu satunya node maka akan langsung dihapus tanpa ada perubahan pointer. Pengondisian terakhir Ketika node tersebut banyak tidak hanya ada satu. Maka ada perubahan di pointernya
- Membuat method delete last, sebenarnya untuk logic mirip" namun perbedaan ada dipointernya.
- Membuat method display untuk menampilkan list akhir Ketika sudah ditambahkan dan didelete
- Membuat main yaitu fungsi utama yang akan dijalankan, didalamnya menerima inputan dari user dan memanggil methodnya.

Soal 3: Menampilkan Elemen dari Depan ke Belakang dan Sebaliknya
Deskripsi Soal: Buatlah program yang memungkinkan pengguna memasukkan beberapa elemen ke dalam Doubly Linked List. Setelah elemen dimasukkan, tampilkan seluruh elemen dalam list dari depan ke belakang, kemudian dari belakang ke depan.
Instruksi:
1. Implementasikan fungsi untuk menampilkan elemen dari depan ke belakang.
2. Implementasikan fungsi untuk menampilkan elemen dari belakang ke depan.
3. Tambahkan 4 elemen ke dalam list dan tampilkan elemen tersebut dalam dua arah.
Contoh Input:
- Input: Masukkan 4 elemen secara berurutan: 1, 2, 3, 4
Output:
- Daftar elemen dari depan ke belakang: 1 <-> 2 <-> 3 <-> 4
- Daftar elemen dari belakang ke depan: 4 <-> 3 <-> 2 <-> 1

```cpp
#include <iostream>
using namespace std;

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

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

void createList_2311104016(DoubleList &L) {
    L.head = NULL;
    L.tail = NULL;
}

Node* createNode_2311104016(int angka) {
    Node* newNode = new Node();
    newNode->data = angka;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

void insertLast_2311104016(DoubleList &L, Node* newNode) {
    if (L.head == NULL) {
        L.head = newNode;
        L.tail = newNode;
    } else {
        newNode->prev = L.tail;
        L.tail->next = newNode;
        L.tail = newNode;
    }
}

void displayForward_2311104016(DoubleList L) {
    Node* current = L.head;
    cout << "Daftar elemen dari depan ke belakang: ";
    while (current != NULL) {
        cout << current->data;
        if (current->next != NULL) {
            cout << " <-> ";
        }
        current = current->next;
    }
    cout << endl;
}
```

```cpp
void displayBackward_2311104016(DoubleList L) {
    Node* current = L.tail;
    cout << "Daftar elemen dari belakang ke depan: ";
    while (current != NULL) {
        cout << current->data;
        if (current->prev != NULL) {
            cout << " <-> ";
        }
        current = current->prev;
    }
    cout << endl;
}

int main() {
    DoubleList list;
    createList_2311104016(list);

    cout << "Input: Masukkan 4 elemen secara berurutan:\n";
    for(int i = 1; i <= 4; i++) {
        int elem;
        cout << "Masukkan elemen ke-" << i << ": ";
        cin >> elem;
        Node* newNode = createNode_2311104016(elem);
        insertLast_2311104016(list, newNode);
    }

    cout << "\nOutput:\n";
    displayForward_2311104016(list);
    displayBackward_2311104016(list);

    return 0;
}
```

```
Input: Masukkan 4 elemen secara berurutan:
Masukkan elemen ke-1: 1
Masukkan elemen ke-2: 2
Masukkan elemen ke-3: 3
Masukkan elemen ke-4: 4

Output:
Daftar elemen dari depan ke belakang: 1 <-> 2 <-> 3 <-> 4
Daftar elemen dari belakang ke depan: 4 <-> 3 <-> 2 <-> 1
PS C:\Users\ZBooK\OneDrive\Dokumen\Struktur data>
```

Penjelasan

- Struct node : struktur nodenya berisikan data, pointer prev dan pointer next
- Struct doublelist : membuat struktur double listnya
- Membuat method createList dengan set L.head dan L.tailnya NULL
- Membuat methoc createNode, dan newNode datanya akan diisi oleh param angka dan pointer next prev null mengibaratkan merupakan node pertama
- Membuat method insertlast : untuk menambahkan data itu dibelakangnya, jadi terurut kebelakang.
- Membuat method displayfirward untuk menampilkan data urut dari depan ke belakang, jadi dia mengurutkannya next pinter to next
- Membuat method displaybackward kalua ini untuk menampilkan data dari belakang kedepan, ini menggunakan pointer prev jadi akan ditarik ke preview (kebelakang)
- Membuat main yaitu fungsi utama yang akan dijalankan, didalamnya menerima inputan dari user dan memanggil methodnya.