

Praktikum Algoritma dan Struktur Data

Doubly Linked List



Oleh :

Rayhan Elmo Athalah Saputra (5223600027)

Program Studi Sarjana Terapan Teknologi Game

Departemen Teknologi Multimedia Kreatif

Politeknik Elektronika Negeri Surabaya

2024

Doubly linked list dibentuk dengan Menyusun sejumlah elemen sehingga pointer next menunjuk ke elemen yang mengikutinya dan pointer back menunjuk ke elemen yang mendahuluinya. Doubly linked list terdiri dari elemen – elemen individu, Dimana masing – masing dihubungkan dengan dua pointer. Masing – masing elemen terdiri dari 3 bagian, yaitu sebuah data dan sebuah pointer yang berisi alamat data berikutnya disebut dengan next dan pointer yang berisi alamat data sebelumnya di sebut before.

PERCOBAAN

1. Implementasikan operasi dasar Double linked list : Menyisipkan sebagai simpul ujung(awal) dari linked list.
2. Implementasikan operasi dasar Double linked list : Membaca atau menampilkan
3. Implementasikan operasi dasar Double linked list : Mencari sebuah simpul tertentu. Tambahkan kondisi jika yang dicari adalah data yang paling depan.
4. Implementasikan operasi dasar Double linked list : Menghapus simpul tertentu.
5. Tambahkan kondisi jika yang dihapus adalah data yang paling depan atau data yang paling terakhir.
6. Gabungkan semua operasi di atas dalam sebuah Menu Pilihan.

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

// Mendefinisi Struktur Linked List
struct Node {
    int data; // Data yang disimpan dalam node
    Node* prev; // Pointer ke node sebelumnya
    Node* next; // Pointer ke node berikutnya
};

// Menyisipkan simpul baru di awal
void insertNode(Node*& head, Node*& tail, int data) {
    Node* n = new Node(); // Membuat node baru
    n->data = data; // Menyimpan data ke dalam node baru
    if (head == nullptr) { // Jika LinkedList masih kosong
        n->next = nullptr;
        n->prev = nullptr;
        head = n; // Node baru menjadi node pertama
        tail = n; // Node baru juga menjadi node terakhir
    }
```

```

    } else { // Jika LinkedList sudah berisi node
        n->next = head; // Node baru menunjuk ke node pertama yang
lama
        head->prev = n; // Node pertama yang lama menunjuk ke node
baru
        head = n; // Node baru menjadi node pertama
    }
}

// Mencari node dengan nilai tertentu dalam LinkedList
void search(Node* head, int data) {
    Node* temp = head; // Pointer sementara yang digunakan untuk
traversal
    int position = 1; // Variabel untuk menyimpan posisi node dalam
LinkedList
    bool found = false; // Variabel untuk menandai apakah data
ditemukan

    // Melakukan traversal LinkedList untuk mencari data
    while (temp != nullptr) {
        if (temp->data == data) { // Jika data ditemukan
            found = true; // Set found menjadi true
            break; // Keluar dari loop
        }
        temp = temp->next; // Pindah ke node berikutnya
        position++; // Tambahkan 1 ke posisi
    }

    // Menampilkan pesan sesuai dengan hasil pencarian
    if (found) {
        cout << "Data " << data << " ditemukan pada node ke-" <<
position << endl;
    } else {
        cout << "Data " << data << " tidak ditemukan dalam
LinkedList." << endl;
    }
}

// Menghapus node dengan nilai tertentu dari LinkedList
void deleteNode(Node*& head, Node*& tail, int data) {
    Node* temp = head; // Pointer sementara untuk traversal

    // Melakukan traversal LinkedList untuk mencari data yang akan
dihapus
    while (temp != nullptr) {
        if (temp->data == data) { // Jika data ditemukan

```

```

        if (temp == head) { // Jika data yang ditemukan berada di
node pertama
            head = head->next; // Node berikutnya menjadi node
pertama
            if (head != nullptr) {
                head->prev = nullptr; // Jika masih ada node lain,
prev dari node pertama diatur menjadi nullptr
            } else {
                tail = nullptr; // Jika tidak ada node lain, tail
diatur menjadi nullptr
            }
        } else if (temp == tail) { // Jika data yang ditemukan
berada di node terakhir
            tail = tail->prev; // Node sebelumnya menjadi node
terakhir
            tail->next = nullptr; // Next dari node terakhir
diatur menjadi nullptr
        } else { // Jika data yang ditemukan berada di tengah
LinkedList
            temp->prev->next = temp->next; // Node sebelumnya
menunjuk ke node setelahnya
            temp->next->prev = temp->prev; // Node setelahnya
menunjuk ke node sebelumnya
        }
        delete temp; // Menghapus node yang mengandung data yang
sesuai
        return; // Keluar dari fungsi setelah menghapus node
    }
    temp = temp->next; // Pindah ke node berikutnya
}

// Jika data tidak ditemukan dalam LinkedList
cout << "Data " << data << " tidak ditemukan dalam LinkedList." <<
endl;
}

// Menampilkan LinkedList
void display(Node* head) {
    Node* temp = head; // Pointer sementara untuk traversal
    while (temp != nullptr) { // Melakukan traversal hingga mencapai
akhir LinkedList
        cout << temp->data << " "; // Menampilkan data dari setiap
node
        temp = temp->next; // Pindah ke node berikutnya
    }
    cout << endl; // Baris baru setelah menampilkan semua data
}

```

```

// Menu utama program
int main() {
    Node* head = nullptr; // Pointer ke node pertama dalam LinkedList
    Node* tail = nullptr; // Pointer ke node terakhir dalam LinkedList
    int choice, data; // Variabel untuk menyimpan pilihan pengguna dan
    data yang dimasukkan

    // Loop utama untuk menampilkan menu dan memproses pilihan
    pengguna
    do {
        // Menampilkan menu pilihan
        cout << "\nMenu Pilihan:\n";
        cout << "1. Sisipkan Node di Awal\n";
        cout << "2. Cari Node\n";
        cout << "3. Hapus Node (berdasarkan data)\n";
        cout << "4. Tampilkan Node\n";
        cout << "5. Keluar\n";
        cout << "Masukkan pilihan Anda: ";
        cin >> choice;

        // Memproses pilihan pengguna
        switch (choice) {
            case 1:
                cout << "Masukkan data yang akan disisipkan: ";
                cin >> data;
                insertNode(head, tail, data); // Memanggil fungsi
                untuk menyisipkan node di awal
                break;
            case 2:
                cout << "Masukkan data yang akan dicari: ";
                cin >> data;
                search(head, data); // Memanggil fungsi untuk mencari
                node dengan data tertentu
                break;
            case 3:
                cout << "Masukkan data yang akan dihapus: ";
                cin >> data;
                deleteNode(head, tail, data); // Memanggil fungsi
                untuk menghapus node dengan data tertentu
                break;
            case 4:
                cout << "LinkedList: ";
                display(head); // Memanggil fungsi untuk menampilkan
                semua node dalam LinkedList
                break;
            case 5:

```

```

        cout << "Keluar dari program.\n";
        break;
    default:
        cout << "Pilihan tidak valid! Silakan coba lagi.\n";
    }
    } while (choice != 5); // Melanjutkan loop sampai pengguna memilih
    untuk keluar

    return 0; // Keluar dari program
}

```

Hasil

main.cpp

Run

Output

Clear

```

1  #include <iostream>
2  using namespace std;
3
4  // Mendefinisikan Struktur Linked List
5  struct Node {
6      int data; // Data yang disimpan dalam node
7      Node* prev; // Pointer ke node sebelumnya
8      Node* next; // Pointer ke node berikutnya
9  };
10
11 // Menyisipkan simpul baru di awal
12 void insertNode(Node*& head, Node*& tail, int data) {
13     Node* n = new Node(); // Membuat node baru
14     n->data = data; // Menyimpan data ke dalam node baru
15     if (head == nullptr) { // Jika LinkedList masih kosong
16         n->next = nullptr;
17         n->prev = nullptr;
18         head = n; // Node baru menjadi node pertama
19         tail = n; // Node baru juga menjadi node terakhir
20     } else { // Jika LinkedList sudah berisi node
21         n->next = head; // Node baru menunjuk ke node

```

```

/tmp/uFDeFFPEd8.o
Menu Pilihan:
1. Sisipkan Node di Awal
2. Cari Node
3. Hapus Node (berdasarkan data)
4. Tampilkan Node
5. Keluar
Masukkan pilihan Anda:

```

Output

/tmp/V0IoV6W3Gx.o

Menu Pilihan:

1. Sisipkan Node di Awal
2. Cari Node
3. Hapus Node (berdasarkan data)
4. Tampilkan Node
5. Keluar

Masukkan pilihan Anda: 1

Masukkan data yang akan disisipkan: 23

Menu Pilihan:

1. Sisipkan Node di Awal
2. Cari Node
3. Hapus Node (berdasarkan data)
4. Tampilkan Node
5. Keluar

Masukkan pilihan Anda: 2

Masukkan data yang akan dicari: 23

Data 23 ditemukan pada node ke-1

Menu Pilihan:

1. Sisipkan Node di Awal
2. Cari Node
3. Hapus Node (berdasarkan data)
4. Tampilkan Node
5. Keluar

Masukkan pilihan Anda: 3

Masukkan data yang akan dihapus: 23

Menu Pilihan:

1. Sisipkan Node di Awal
2. Cari Node
3. Hapus Node (berdasarkan data)
4. Tampilkan Node
5. Keluar

Masukkan pilihan Anda: 4

LinkedList:

Menu Pilihan:

1. Sisipkan Node di Awal
2. Cari Node
3. Hapus Node (berdasarkan data)
4. Tampilkan Node
5. Keluar

Masukkan pilihan Anda: 5

Keluar dari program.

=== Code Execution Successful ===|