Praktikum Algoritma dan Struktur Data

Linked List



# Oleh:

## Rangga Raisha Syaputra / 5223600007

**Program Studi STr Teknologi Game Departemen Teknologi Multimedia Kreatif Politeknik Elektronika Negeri Surabaya 2023/2024**

Pada praktikum ini, saya akan mengerjakan soal – soal yang telah disediakan supaya untuk dikerjakan dalam Bahasa pemrograman C++. Berikut soal – soal yang akan dikerjakan,

Operasi Pada Linked list Terdapat beberapa Operasi yang penting pada linked list, yaitu:

1. Menyisipkan sebagai simpul ujung(awal) dari linked list.

2. Membaca atau menampilkan

3. Mencari sebuah simpul tertentu

4. Menyisipkan sebagai simpul terakhir

5. Menghapus simpul tertentu

6. Menyisipkan setelah simpul tertentu

7. Menyisipkan sebelum simpul tertentu

**HASIL**

**1. Menyisipkan sebagai simpul ujung(awal) dari linked list.**

#include <iostream>

using namespace std;

struct Node {

int data;

Node \*next;

};

Node \*head = NULL;

void newData(int data) {

Node \*newNode = new Node;

newNode->data = data;

newNode->next = head;

head = newNode;

}

void list() {

Node \*temp = head;

while (temp != NULL) {

cout << temp->data << endl;

temp = temp->next;

}

cout << endl;

}

int main() {

newData(30);

newData(20);

newData(10);

list();

return 0;

}

**2. Membaca atau menampilkan**

#include <iostream>

using namespace std;

struct Node {

int data;

Node \*next;

};

Node \*head = NULL;

void newData(int data) {

Node \*newNode = new Node;

newNode->data = data;

newNode->next = head;

head = newNode;

}

void list() {

Node \*temp = head;

while (temp != NULL) {

cout << temp->data << endl;

temp = temp->next;

}

cout << endl;

}

int main() {

newData(30);

newData(20);

newData(10);

list();

return 0;

}

**3. Mencari sebuah simpul tertentu**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* head = NULL;

void newData(int data) {

Node\* newNode = new Node;

newNode->data = data;

newNode->next = NULL;

if (head == NULL) {

head = newNode;

}

else {

Node\* temp = head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

}

}

void list() {

Node\* temp = head;

while (temp != NULL) {

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

Node\* search(int data) {

Node\* temp = head;

while (temp != NULL) {

if (temp->data == data) {

return temp;

}

temp = temp->next;

}

return 0;

}

int main() {

newData(1);

newData(2);

newData(3);

newData(4);

newData(5);

list();

int key = 0;

Node\* foundNode = search(key);

if (foundNode == NULL) {

cout << "Data " << key << " tidak ditemukan." << endl;

}

else {

cout << "Data " << key << " ditemukan." << endl;

}

return 0;

}

**4. Menyisipkan sebagai simpul terakhir**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* head = NULL;

void baruData(int data) {

Node\* baruNode = new Node;

baruNode->data = data;

baruNode->next = NULL;

if (head == NULL) {

head = baruNode;

}

else {

Node\* temp = head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = baruNode;

}

}

void list() {

Node\* temp = head;

while (temp != NULL) {

cout << temp->data << endl;

temp = temp->next;

}

cout << endl;

}

int main() {

baruData(10);

baruData(20);

baruData(30);

baruData(40);

list();

return 0;

}

**5. Menghapus simpul tertentu**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* head = NULL;

void baruData(int data) {

Node\* baruNode = new Node;

baruNode->data = data;

baruNode->next = NULL;

if (head == NULL) {

head = baruNode;

} else {

Node\* temp = head;

while (temp->next != NULL) {

temp = temp->next; // Pemindahan ke simpul berikutnya

}

temp->next = baruNode; // Penyisipan di akhir linked list

}

}

void list() {

Node\* temp = head;

while (temp != NULL) {

cout << temp->data << endl;

temp = temp->next;

}

cout << endl;

}

// Fungsi untuk menghapus data pada posisi tertentu

void hapusNode(int value) {

// Handle empty list case

if (head == NULL) {

cout << "Tidak bisa menghapus data" << endl;

}

// Menghapus data pada posisi pertama

if (head->data == value) {

Node\* temp = head;

head = head->next;

delete temp;

}

// Mencari node

Node\* temp = head;

Node\* prev = NULL;

while (temp != NULL && temp->data != value) {

prev = temp;

temp = temp->next;

}

// Mencari data untuk dihapus

if (temp == NULL) {

cout << "Data tidak ada yang dihapus karena Node dengan nilai " << value << " tidak tersedia" << endl;

}

// Menghapus node

prev->next = temp->next;

delete temp;

}

int main() {

baruData(10);

baruData(20);

baruData(30);

baruData(40);

cout << "List data sebelum ada data yang dihapus: " << endl;

list();

hapusNode(30);

cout << "List setelah ada data yang dihapus:" << endl;

list();

return 0;

}

**6. Menyisipkan setelah simpul tertentu**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* head = NULL;

void newData(int data) {

Node\* newNode = new Node;

newNode->data = data;

newNode->next = NULL;

if (head == NULL) {

head = newNode;

} else {

Node\* temp = head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

}

}

//Fungsi menambahkan node

void newNode(Node\* prev\_node, int new\_data) {

if (prev\_node == NULL) {

cout << "Node tidak boleh kosong" << endl;

return;

}

Node\* new\_node = new Node;

new\_node->data = new\_data;

new\_node->next = prev\_node->next;

prev\_node->next = new\_node;

}

void list() {

Node\* temp = head;

while (temp != NULL) {

cout << temp->data << endl;

temp = temp->next;

}

cout << endl;

}

int main() {

newData(10);

newData(20);

newData(30);

newData(40);

cout << "List data awal sebelum disisipkan:" << endl;

list();

// Menambahkan data setelah 20

int new\_data = 50;

Node\* prev\_node = head;

while (prev\_node != NULL && prev\_node->data != 20) {

prev\_node = prev\_node->next;

}

if (prev\_node == NULL) {

cout << "Node yang ingin ditambahkan tidak tersedia" << endl;

} else {

newNode(prev\_node, new\_data);

cout << "List data setelah disisipkan 50 setelah data 20:" << endl;

list();

}

return 0;

}

**7. Menyisipkan sebelum simpul tertentu**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* head = NULL;

void newData(int data) {

Node\* newNode = new Node;

newNode->data = data;

newNode->next = NULL;

if (head == NULL) {

head = newNode;

} else {

Node\* temp = head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

}

}

void insertBefore(int targetData, int newData) {

// membuat node baru

Node\* newNode = new Node;

newNode->data = newData;

if (head == NULL || head->data == targetData) {

newNode->next = head;

head = newNode;

return;

}

// menemukan node pada sebelum

Node\* prev = head, \*curr = head->next;

while (curr != NULL && curr->data != targetData) {

prev = curr;

curr = curr->next;

}

// Jika data tidak tersedia

if (curr == NULL) {

cout << "Target data not found in the list." << endl;

return;

}

// Insert node baru

newNode->next = curr;

prev->next = newNode;

}

void list() {

Node\* temp = head;

while (temp != NULL) {

cout << temp->data << endl;

temp = temp->next;

}

cout << endl;

}

int main() {

newData(10);

newData(20);

newData(30);

newData(40);

cout << "List awal: " << endl;

list();

int targetData = 20, newDataToInsert = 70;

insertBefore(targetData, newDataToInsert);

cout << "List data setelah disisipkan " << newDataToInsert << " sebelum data " << targetData << ": " << endl;

list();

return 0;

}