**Laporan Hasil Praktikum**

Linked List

****

**Oleh :**

Tiyyasha Ananda Mufti Hanifah (5223600011)

**Program Studi D4 Teknologi Game**

**Departemen Teknologi Multimedia Kreatif**

**Politeknik Elektronika Negeri Surabaya**

**2024**

1. **Dasar Teori**

Linked list adalah struktur data fundamental yang terdiri dari node-node (simpul) yang terhubung secara linier. Setiap node berisi dua komponen utama:

1. Data: Nilai atau item yang disimpan dalam node. Data ini dapat berupa tipe data dasar (seperti int, float, char) atau struktur data kompleks (seperti objek kelas).
2. Penunjuk (pointer) next: Referensi ke node berikutnya dalam urutan linked list. Penunjuk ini menampung alamat memori dari node berikutnya, memungkinkan navigasi melalui linked list.

Linked list memiliki berbagai operasi dasar yang penting untuk manipulasi penggunaanya:

1. Penambahan (Insertion): Menambahkan node baru ke linked list di posisi tertentu, seperti di awal, akhir, atau di antara node-node yang ada.
2. Penghapusan (Deletion): Menghapus node tertentu dari linked list, memperbarui penunjuk yang relevan untuk mempertahankan integritas struktur.
3. Pencarian (Search): Mencari node yang berisi data tertentu dalam linked list, mengembalikan posisi atau referensi ke node tersebut.
4. Traversal (Penelusuran): Mengunjungi setiap node dalam linked list secara berurutan, sering kali untuk memproses data atau melakukan operasi lainnya.
5. **Alat dan Bahan**

Alat yang digunakan pada percobaan menggambar kali ini adalah :

1. Laptop/PC
2. C++ compiler
3. Internet
4. **Langkah Percobaan**
5. Menyisipkan sebagai simpul ujung(awal) dari linked list.

/\*

Nama : Tiyyasha Ananda Mufti Hanifah

NRP : 5223600011

\*/

#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(10);

newData(20);

newData(30);

list();

return 0;

}

Analisa: Program mendefinisikan struktur Node untuk mewakili setiap simpul dalam linked list. Setiap Node memiliki dua anggota:

* data: Menyimpan nilai integer untuk node tersebut.
* next: Sebuah pointer yang menunjuk ke node berikutnya dalam linked list.

1. Mengakses Elemen Struktur

/\*

Nama : Tiyyasha Ananda Mufti Hanifah

NRP : 5223600011

\*/

#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(10);

newData(20);

newData(30);

list();

return 0;

}

Analisa: Membuat pointer temp dan menginisialisasinya dengan bagian atas daftar. Iterasi daftar menggunakan perulangan while hingga temp menjadi NULL. Di setiap iterasi, mencetak nilai data yang disimpan di node saat ini. Memindahkan temp ke node berikutnya dalam daftar dengan menetapkan temp = temp->next. Mencetak karakter baris baru terakhir.

1. Mencari sebuah simpul tertentu.

/\*

Nama : Tiyyasha Ananda Mufti Hanifah

NRP : 5223600011

\*/

#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 NULL;

}

int main() {

newData(1);

newData(2);

newData(3);

newData(4);

newData(5);

list();

int key = 3;

Node\* foundNode = search(key);

if (foundNode == NULL) {

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

}

else {

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

}

return 0;

}

Analisa: Program ini mendeklarasikan linked list dengan fungsi untuk menambahkan node (newData), mencetak nilai (list), dan mencari node tertentu berdasarkan nilai datanya (search).

1. Menyisipkan sebagai simpul terakhir

/\*

Nama : Tiyyasha Ananda Mufti Hanifah

NRP : 5223600011

\*/

#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 << endl;

temp = temp->next;

}

cout << endl;

}

int main() {

newData(10);

newData(20);

newData(30);

newData(40);

list();

return 0;

}

Analisa: Program ini menggunakan fungsi head untuk mendeklarasikan titik awal linked list. Fungsi newData menelusuri list untuk menemukan node terakhir dan menyisipkan node baru di akhir. Fungsi list melakukan iterasi melalui list dan mencetak isinya.

1. Menghapus simpul tertentu.

/\*

Nama : Tiyyasha Ananda Mufti Hanifah

NRP : 5223600011

\*/

#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 << endl;

temp = temp->next;

}

cout << endl;

}

// Fungsi untuk menghapus data pada posisi tertentu

void deleteNode(int value) {

// Handle empty list case

if (head == NULL) {

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

return;

}

// Mengahapus data pada posisi pertama

if (head->data == value) {

Node\* temp = head;

head = head->next;

delete temp;

return;

}

// 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 << "Node with value " << value << " not found." << endl;

return;

}

// Menghapus node

prev->next = temp->next;

delete temp;

}

int main() {

newData(10);

newData(20);

newData(30);

newData(40);

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

list();

cout << "Menghapus data 20:" << endl;

deleteNode(20);

cout << "List baru:" << endl;

list();

return 0;

}

Analisa: deleteNode(int value): menghapus node yang berisi nilai yang ditentukan.

1. Menyisipkan setelah simpul tertentu.

/\*

Nama : Tiyyasha Ananda Mufti Hanifah

NRP : 5223600011

\*/

#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 << "The given previous node cannot be NULL" << 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 awal:" << endl;

list();

// Menambahkan data setelah 20

int new\_data = 55;

Node\* prev\_node = head;

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

prev\_node = prev\_node->next;

}

if (prev\_node == NULL) {

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

} else {

newNode(prev\_node, new\_data);

cout << "List setelah ditambahkan data baru:" << endl;

list();

}

return 0;

}

Analisa: Program ini membuat struktur dengan fungsi untuk menambahkan node:

* newData(int data): Membuat node baru dengan data yang diberikan, menambahkannya ke akhir daftar jika daftar tidak kosong, dan mengubah head pointer ke node baru jika daftar kosong.
* newNode(Node\* prev\_node, int new\_data): Menyisipkan node baru dengan data yang diberikan setelah prev\_node yang ditentukan. Ini mengembalikan pesan kesalahan jika prev\_node adalah NULL.
* Fungsi list(): melakukan iterasi dan mencetak data setiap node.

1. Menyisipkan sebelum simpul tertentu.

/\*

Nama : Tiyyasha Ananda Mufti Hanifah

NRP : 5223600011

\*/

#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) {

// Create the new node

Node\* newNode = new Node;

newNode->data = newData;

// Check if the list is empty or if the target data is the head

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

newNode->next = head;

head = newNode;

return;

}

// Otherwise, traverse the list to find the node before the target

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

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

prev = curr;

curr = curr->next;

}

// Handle the case where the target data is not found

if (curr == NULL) {

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

return;

}

// Insert the new node

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 = 15;

insertBefore(targetData, newDataToInsert);

cout << "List setelan ditambahkan " << newDataToInsert << " sebelum " << targetData << ": " << endl;

list();

return 0;

}

Analisa: Program ini membuat struktur dengan fungsi untuk menambahkan node:

* data(int data): Fungsi ini membuat node baru, menetapkan `data` yang disediakan ke node tersebut, dan menyisipkannya di akhir list yang ada.
* insertBefore(int targetData, int newData): Fungsi ini menyisipkan node baru dengan `newData` sebelum node yang berisi `targetData` dalam list.
* list(): Fungsi ini mencetak data setiap node dalam daftar tertaut, dimulai dari kepala dan melakukan iterasi hingga mencapai node dengan `berikutnya` disetel ke `NULL`.

1. **Kesimpulan**

Linked list adalah struktur data dinamis dengan kelebihan berikut:

1. Memori Dinamis: Node dapat dialokasikan secara dinamis saat runtime, sesuai kebutuhan, sehingga lebih efisien untuk data yang ukurannya tidak diketahui sebelumnya atau dapat berubah sewaktu-waktu.
2. Penyisipan dan Penghapusan Efisien: Operasi penyisipan dan penghapusan pada linked list umumnya lebih efisien daripada array.
3. Ukuran Fleksibel: Linked list tidak memiliki batasan ukuran tetap, sehingga dapat tumbuh atau menyusut secara dinamis sesuai kebutuhan program.

Namun, linked list juga memiliki beberapa kelemahan:

1. Akses Acak Lambat
2. Overhead Memori Tambahan

Oleh karena itu, pemilihan antara linked list dan array bergantung pada persyaratan spesifik kasus penggunaan dan pertimbangan trade-off antara keunggulan dan kelemahan masing-masing struktur.