LAPORAN PRAKTIKUM
Modul 3
Single Linked List(1)

Disusun Oleh:
Jauhar Fajar Zuhair
2311104072
S1SE-07-2

Dosen :
Wahyu Andri Saputra, S.Pd., M.Eng.

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO

2024

Tujuan Praktikum

1. Memahami penggunaan linked list dengan pointer dan operator-operator dalam program.
2. Memahami operasi-operasi dasar dalam linked list.
3. Membuat program dengan menggunakan linked list sesuai dengan prototype yang ada.

Landasan Teori

Linked list merupakan salah satu bentuk struktur data yang terdiri dari serangkaian elemen data yang saling terhubung. Struktur ini bersifat fleksibel, dapat bertambah dan berkurang sesuai kebutuhan. Data yang disimpan dalam linked list dapat berupa data tunggal (seperti string) atau data majemuk (seperti record dengan berbagai tipe data).

Implementasi linked list dapat menggunakan array atau pointer, namun penggunaan pointer lebih disukai karena beberapa alasan:

1. Pointer bersifat dinamis, sedangkan array statis.
2. Linked list dengan pointer lebih mudah diimplementasikan karena datanya saling terhubung.
3. Sifat fleksibel linked list lebih sesuai dengan karakteristik pointer.
4. Pointer lebih efisien dalam menangani linked list dibandingkan array.
5. Array lebih cocok untuk data dengan jumlah elemen maksimum yang telah diketahui sejak awal.

Dalam implementasinya, akses elemen pada linked list dengan pointer dapat menggunakan operator panah (->) atau titik (.).

Beberapa model ADT (Abstract Data Type) Linked list yang umum dipelajari meliputi:

1. Single Linked list
2. Double Linked list
3. Circular Linked list
4. Multi Linked list
5. Stack (Tumpukan)
6. Queue (Antrian)

7. Tree

8. Graph

Operasi-operasi dasar pada ADT Linked list mencakup:

1. Pembuatan dan inisialisasi list (Create List)

2. Penyisipan elemen list (Insert)

3. Penghapusan elemen list (Delete)

4. Penelusuran dan penampilan elemen list (View)

5. Pencarian elemen list (Searching)

6. Pengubahan isi elemen list (Update)

Single Linked list merupakan model ADT Linked list yang hanya memiliki satu arah pointer. Komponen utama dalam single linked list adalah:

- Elemen: segmen data dalam list

- Data: informasi utama yang tersimpan dalam elemen

- Successor: bagian elemen yang berfungsi sebagai penghubung antar elemen

Karakteristik Single Linked list:

1. Hanya memerlukan satu pointer

2. Node terakhir menunjuk ke Nil (kecuali pada list circular)

3. Hanya dapat melakukan pembacaan maju

4. Pencarian sekuensial dilakukan jika data tidak terurut

5. Lebih mudah dalam melakukan penyisipan atau penghapusan di tengah list

Guided

Guided 1: Implementasi Single Linked List untuk Data Mahasiswa

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

// Deklarasi Struct untuk mahasiswa
struct mahasiswa {
    char nama[30];
    char nim[10];
};

// Deklarasi Struct Node
struct Node {
    mahasiswa data;
    Node *next;
};

Node *head;
Node *tail;

// Inisialisasi List
void init() {
    head = nullptr;
    tail = nullptr;
}

// Pengecekan apakah list kosong
bool isEmpty() {
    return head == nullptr;
}

// Tambah Depan
void insertDepan(const mahasiswa &data) {
    Node *baru = new Node;
    baru->data = data;
    baru->next = nullptr;
    if (isEmpty()) {
        head = tail = baru;
    } else {
        baru->next = head;
        head = baru;
    }
}

// Tambah Belakang
void insertBelakang(const mahasiswa &data) {
    Node *baru = new Node;
    baru->data = data;
    baru->next = nullptr;
    if (isEmpty()) {
        head = tail = baru;
    } else {
        tail->next = baru;
        tail = baru;
    }
}

// Hitung Jumlah List
int hitungList() {
    Node *current = head;
    int jumlah = 0;
    while (current != nullptr) {
        jumlah++;
        current = current->next;
    }
    return jumlah;
}

// Hapus Depan
void hapusDepan() {
    if (!isEmpty()) {
        Node *hapus = head;
        head = head->next;
        delete hapus;
        if (head == nullptr) {
            tail = nullptr; // Jika list menjadi kosong
        }
    } else {
        cout << "List kosong!" << endl;
    }
}

// Hapus Belakang
void hapusBelakang() {
    if (!isEmpty()) {
        if (head == tail) {
            delete head;
            head = tail = nullptr; // List menjadi kosong
        } else {
            Node *bantu = head;
            while (bantu->next != tail) {
                bantu = bantu->next;
            }
            delete tail;
            tail = bantu;
            tail->next = nullptr;
        }
    } else {
        cout << "List kosong!" << endl;
    }
}

// Tampilkan List
void tampil() {
    Node *current = head;
    if (!isEmpty()) {
        while (current != nullptr) {
            cout << "Nama: " << current->data.nama << ", NIM: " << current->data.nim << endl;
            current = current->next;
        }
    } else {
        cout << "List masih kosong!" << endl;
    }
}

// Hapus List
void clearList() {
    Node *current = head;
    while (current != nullptr) {
        Node *hapus = current;
        current = current->next;
        delete hapus;
    }
    head = tail = nullptr;
    cout << "List berhasil terhapus!" << endl;
}

// Main Function
int main() {
    init();

    // Contoh data mahasiswa
    mahasiswa m1 = {nama:"Alice", nim:"123456"};
    mahasiswa m2 = {nama:"Bob", nim:"654321"};
    mahasiswa m3 = {nama:"Charlie", nim:"112233"};

    // Menambahkan mahasiswa ke dalam list
    insertDepan(m1);
    tampil();
    insertBelakang(m2);
    tampil();
    insertDepan(m3);
    tampil();

    // Menghapus elemen dari list
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();

    // Menghapus seluruh list
    clearList();

    return 0;
}
```

Program ini mengimplementasikan single linked list untuk menyimpan dan mengelola data mahasiswa. Berikut adalah penjelasan singkat tentang komponen-komponen utama program:

1. Struktur Data:
   - Struct `mahasiswa`: Menyimpan data nama dan NIM mahasiswa.
   - Struct `Node`: Menyimpan data mahasiswa dan pointer ke node berikutnya.

2. Fungsi-fungsi Utama:
   - `init()`: Inisialisasi list kosong.
   - `isEmpty()`: Mengecek apakah list kosong.
   - `insertDepan()`: Menambahkan node di awal list.
   - `insertBelakang()`: Menambahkan node di akhir list.
   - `hitungList()`: Menghitung jumlah node dalam list.
   - `hapusDepan()`: Menghapus node pertama.
   - `hapusBelakang()`: Menghapus node terakhir.
   - `tampil()`: Menampilkan seluruh isi list.
   - `clearList()`: Menghapus seluruh isi list.

3. Fungsi `main()`:
   - Mendemonstrasikan penggunaan fungsi-fungsi yang telah dibuat.
   - Menambahkan beberapa data mahasiswa ke dalam list.
   - Menampilkan isi list setelah setiap operasi.
   - Menghapus elemen dari list.
   - Membersihkan seluruh list di akhir program.

Program ini memberikan contoh praktis penggunaan single linked list untuk mengelola data mahasiswa, termasuk operasi dasar seperti penambahan, penghapusan, dan penampilan data.

Guided 2: Implementasi SingleLinked List dengan Operasi Dasar

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

// Definisi struktur untuk elemen list
struct Node {
    int data;       // Menyimpan nilai elemen
    Node* next;     // Pointer ke elemen berikutnya
};

// Fungsi untuk mengalokasikan memori untuk node baru
Node* alokasi(int value) {
    Node* newNode = new Node; // Alokasi memori untuk elemen baru
    if (newNode != nullptr) { // Jika alokasi berhasil
        newNode->data = value; // Mengisi data node
        newNode->next = nullptr; // Set next ke nullptr
    }
    return newNode; // Mengembalikan pointer node baru
}

// Fungsi untuk dealokasi memori node
void dealokasi(Node* node) {
    delete node; // Mengembalikan memori yang digunakan oleh node
}

// Pengecekan apakah list kosong
bool isListEmpty(Node* head) {
    return head == nullptr; // List kosong jika head adalah nullptr
}

// Menambahkan elemen di awal list
void insertFirst(Node* &head, int value) {
    Node* newNode = alokasi(value); // Alokasi memori untuk elemen baru
    if (newNode != nullptr) {
        newNode->next = head;     // Menghubungkan elemen baru ke elemen pertama
        head = newNode;           // Menetapkan elemen baru sebagai elemen pertama
    }
}

// Menambahkan elemen di akhir list
void insertLast(Node* &head, int value) {
    Node* newNode = alokasi(value); // Alokasi memori untuk elemen baru
    if (newNode != nullptr) {
        if (isListEmpty(head)) {  // Jika list kosong
            head = newNode;       // Elemen baru menjadi elemen pertama
        } else {
            Node* temp = head;
            while (temp->next != nullptr) { // Mencari elemen terakhir
                temp = temp->next;
            }
            temp->next = newNode; // Menambahkan elemen baru di akhir list
        }
    }
}

// Menampilkan semua elemen dalam list
void printList(Node* head) {
    if (isListEmpty(head)) {
        cout << "List kosong!" << endl;
    } else {
        Node* temp = head;
        while (temp != nullptr) {  // Selama belum mencapai akhir list
            cout << temp->data << " "; // Menampilkan data elemen
            temp = temp->next; // Melanjutkan ke elemen berikutnya
        }
        cout << endl;
    }
}

// Menghitung jumlah elemen dalam list
int countElements(Node* head) {
    int count = 0;
    Node* temp = head;
    while (temp != nullptr) {
        count++;           // Menambah jumlah elemen
        temp = temp->next; // Melanjutkan ke elemen berikutnya
    }
    return count;          // Mengembalikan jumlah elemen
}

// Menghapus semua elemen dalam list dan dealokasi memori
void clearList(Node* &head) {
    while (head != nullptr) {
        Node* temp = head;  // Simpan pointer ke node saat ini
        head = head->next;  // Pindahkan ke node berikutnya
        dealokasi(temp);    // Dealokasi node
    }
}

int main() {
    Node* head = nullptr;  // Membuat list kosong

    // Menambahkan elemen ke dalam list
    insertFirst(&head, value:10); // Menambahkan elemen 10 di awal list
    insertLast(&head, value:20);  // Menambahkan elemen 20 di akhir list
    insertLast(&head, value:30);  // Menambahkan elemen 30 di akhir list

    // Menampilkan isi list
    cout << "Isi List: ";
    printList(head);

    // Menampilkan jumlah elemen
    cout << "Jumlah elemen: " << countElements(head) << endl;

    // Menghapus semua elemen dalam list
    clearList(&head);

    // Menampilkan isi list setelah penghapusan
    cout << "Isi List setelah penghapusan: ";
    printList(head);

    return 0;
}
```

Program ini mendemonstrasikan implementasi dasar single linked list dengan fokus pada operasi-operasi fundamental. Berikut adalah komponen-komponen utama program:

1. Struktur Data:

   - Struct `Node`: Menyimpan data integer dan pointer ke node berikutnya.

2. Fungsi-fungsi Utama:

   - `alokasi()`: Mengalokasikan memori untuk node baru.
   - `dealokasi()`: Membebaskan memori yang digunakan oleh node.
   - `isListEmpty()`: Memeriksa apakah list kosong.
   - `insertFirst()`: Menyisipkan elemen di awal list.
   - `insertLast()`: Menyisipkan elemen di akhir list.
   - `printList()`: Menampilkan seluruh elemen dalam list.
   - `countElements()`: Menghitung jumlah elemen dalam list.
   - `clearList()`: Menghapus seluruh elemen dan membebaskan memori.

3. Fungsi `main()`:

   - Mendemonstrasikan penggunaan fungsi-fungsi yang telah dibuat.
   - Membuat list kosong dan menambahkan beberapa elemen.
   - Menampilkan isi list dan jumlah elemennya.
   - Membersihkan list dan menampilkan hasilnya.

Program ini memberikan pemahaman dasar tentang cara kerja single linked list, termasuk alokasi dan dealokasi memori, penyisipan elemen, penghitungan elemen, dan pembersihan list. Ini merupakan fondasi penting untuk memahami struktur data linked list dan operasinya.

Kedua program guided ini memberikan contoh praktis implementasi single linked list dengan berbagai operasi dasar, yang sesuai dengan tujuan praktikum yang telah ditetapkan.