**LAPORAN PRAKTIKUM STRUKTUR DATA**

**MODUL**

**SINGLE LINKED LIST**



**Disusun Oleh :**

NAMA : Ayu Setyaning Tyas

NIM : 103112430119

**Dosen**

FAHRUDIN MUKTI WIBOWO

**PROGRAM STUDI STRUKTUR DATA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2025**

1. Dasar Teori

Single Linked List (Daftar Berantai Tunggal) adalah sebuah struktur data linear dinamis yang sangat berbeda dari array. Alih-alih menyimpan data di lokasi memori yang berdekatan, SLL tersusun dari unit-unit terpisah yang disebut node. Setiap node punya dua bagian: data (misalnya, informasi lagu) dan sebuah pointer (petunjuk alamat) yang mengarahkan ke node berikutnya secara berurutan. Node pertama selalu diakses melalui Head, dan rantai ini berakhir ketika node terakhir menunjuk ke nilai NULL. Manfaat utamanya, seperti yang dijelaskan Wikipedia, adalah kemampuannya untuk mengubah ukuran secara fleksibel dan melakukan operasi penyisipan (menambah) serta penghapusan (menghilangkan) data dengan sangat efisien hanya dengan memanipulasi ulang pointer, sebuah proses yang ideal untuk mengelola daftar yang sering berubah seperti playlist musik.

1. Guided (berisi screenshot source code & output program disertai penjelasannya) [

Guided 1

Code *singlylist.h*

#ifndef SINGLYLIST\_H\_INCLUDE

#define SINGLYLIST\_H\_INCLUDE

#include <iostream>

#define Nill NULL

typedef int infotype;

typedef struct Elmtlist \*address;

struct Elmtlist {

    infotype info;

    address next;

};

struct List {

    address First;

};

//deklarasi prosedur dan fungsi primitif

void CreateList(List &L);

address Alokasi(infotype x);

void Dealokasi(address &P);

void InsertFirst(List &L, address P);

void InsertLast(List &L, address P);

void printInfo(List L);

#endif

Code *singlylist.cpp*

#include "singlylist.h"

void CreateList(List &L) {

    L.First = Nill;

}

address Alokasi(infotype x) {

    address P = new Elmtlist;

        P->info = x;

        P->next = Nill;

    return P;

}

void Dealokasi(address &P) {

    delete P;

}

void InsertFirst(List &L, address P) {

    P->next = L.First;

    L.First = P;

}

void InsertLast(List &L, address P) {

    if (L.First == Nill) {

        //jika list kosong, insertlast sama dengan insertfirst

       InsertFirst(L, P);

    } else {

        // jika list tidak kosong, cari elemen terakhir

        address Last = L.First;

        while (Last->next != Nill) {

            Last = Last->next;

        }

        // sambungkan elemen terakhir dengan elemen baru (p)

        Last->next = P;

    }

}

void printInfo(List L) {

    address P = L.First;

    if (P == Nill) {

        std::cout << "List kosong! " << std::endl;

    } else {

        while (P != Nill) {

            std::cout << P->info << " ";

            P = P->next;

        }

        std::cout << std::endl;

    }

}

Code *main.cpp*

#include <iostream>

#include <cstdlib>

#include "singlylist.h"

#include "singlylist.cpp"

using namespace std;

int main() {

    List L;

    address P; // cukup satu pointer untuk digunakan berulang kali

    CreateList(L);

    cout << "mengisi list menggunakan insertlast..." << endl;

    // mengisi list sesuai urutan

    P = Alokasi(9);

    InsertLast(L, P);

    P = Alokasi(12);

    InsertLast(L, P);

    P = Alokasi(8);

    InsertLast(L, P);

    P = Alokasi(0);

    InsertLast(L, P);

    P = Alokasi (2);

    InsertLast(L, P);

    cout << "isi list sekarang adalah: ";

    printInfo(L);

    system ("pause");

    return 0;

}

A screenshot of a computer

AI-generated content may be incorrect.Screenshots Output

Deskripsi:

Program ini dibuat menggunakan Single Linked List. Program ini dibuat bertujuan untuk mempelajari bagaimana cara menambahkan dan menghampus tanpa bantuan array.

1. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Unguided 1

Code *playlist.h*

#ifndef PLAYLIST\_H\_INCLUDED

#define PLAYLIST\_H\_INCLUDED

#include <iostream>

#include <string>

using namespace std;

#define Nil NULL

struct Lagu {

    string judul;

    string penyanyi;

    float durasi;

};

typedef Lagu infotype;

typedef struct elmlist \*address;

struct elmlist {

    infotype info;

    address next;

};

struct List {

    address first;

};

void createList(List &L);

address alokasi(infotype x);

void dealokasi(address P);

void insertFirst(List &L, address P);

void insertLast(List &L, address P);

void insertAfterN(List &L, address P, int posisi);

void deleteByTitle(List &L, string judul);

void printInfo(List L);

#endif

Code *Playlist.cpp*

#include "playlist.h"

void createList(List &L) {

    L.first = Nil;

}

address alokasi(infotype x) {

    address P = new elmlist;

    P->info = x;

    P->next = Nil;

    return P;

}

void dealokasi(address P) {

    delete P;

}

void insertFirst(List &L, address P) {

    P->next = L.first;

    L.first = P;

}

void insertLast(List &L, address P) {

    if (L.first == Nil) {

        L.first = P;

    } else {

        address Q = L.first;

        while (Q->next != Nil) {

            Q = Q->next;

        }

        Q->next = P;

    }

}

void insertAfterN(List &L, address P, int posisi) {

    address Q = L.first;

    int count = 1;

    while (Q != Nil && count < posisi) {

        Q = Q->next;

        count++;

    }

    if (Q != Nil) {

        P->next = Q->next;

        Q->next = P;

    } else {

        cout << "Posisi tidak valid. Lagu ditambahkan di akhir playlist.\n";

        insertLast(L, P);

    }

}

void deleteByTitle(List &L, string judul) {

    if (L.first == Nil) {

        cout << "Playlist kosong.\n";

        return;

    }

    address P = L.first;

    address Prec = Nil;

    while (P != Nil && P->info.judul != judul) {

        Prec = P;

        P = P->next;

    }

    if (P == Nil) {

        cout << "Lagu dengan judul \"" << judul << "\" tidak ditemukan.\n";

    } else {

        if (Prec == Nil) {

            L.first = P->next;

        } else {

            Prec->next = P->next;

        }

        dealokasi(P);

        cout << "Lagu \"" << judul << "\" berhasil dihapus.\n";

    }

}

void printInfo(List L) {

    if (L.first == Nil) {

        cout << "Playlist kosong.\n";

    } else {

        address P = L.first;

        int i = 1;

        cout << "=== Daftar Lagu dalam Playlist ===\n";

        while (P != Nil) {

            cout << i++ << ". Judul  : " << P->info.judul << endl;

            cout << "   Penyanyi : " << P->info.penyanyi << endl;

            cout << "   Durasi   : " << P->info.durasi << " menit\n";

            cout << "-----------------------------------\n";

            P = P->next;

        }

    }

}

Code *main.cpp*

#include "playlist.h"

int main() {

    List L;

    createList(L);

    infotype lagu;

    address P;

    lagu = {"Happy Birthday", "Back Number", 4.18};

    P = alokasi(lagu);

    insertFirst(L, P);

    lagu = {"Dear.My Darling", "Boynextdoor", 1.40};

    P = alokasi(lagu);

    insertFirst(L, P);

    lagu = {"Beautiful Life", "Xdinary Heroes", 4.25};

    P = alokasi(lagu);

    insertLast(L, P);

    lagu = {"Happy Ending", "Back Number", 5.14};

    P = alokasi(lagu);

    insertAfterN(L, P, 3);

    printInfo(L);

    cout << endl;

    deleteByTitle(L, "Beautiful Life");

    cout << endl << "Setelah penghapusan:\n";

    printInfo(L);

    return 0;

}

A screenshot of a computer

AI-generated content may be incorrect.Screenshots Output

A screenshot of a computer

AI-generated content may be incorrect.

Deskripsi:

Program Single Linked List Playlist Lagu dibuat untuk memahami, mengimplementasikan, dan mempraktikkan struktur data Linked List dalam bentuk aplikasi nyata pengelolaan playlist lagu.  
Selain itu, program ini juga melatih konsep pointer, operasi penyisipan/penghapusan node, dan pemrograman modular di C++.

1. Kesimpulan

Program Single Linked List dalam C++ ini dirancang sebagai sarana pembelajaran praktis untuk menguasai konsep dasar struktur data Linked List melalui implementasi sistem pengelolaan playlist lagu. Mulai dari operasi dasar seperti insertLast(), program kemudian dikembangkan untuk menyimpan data lagu (judul, penyanyi, durasi) dalam setiap node dan memungkinkan pengguna melakukan berbagai operasi, yaitu penyisipan (di awal, akhir, dan setelah node tertentu), penghapusan berdasarkan judul, serta penelusuran dan penampilan seluruh playlist. Tujuannya adalah membantu pembelajar memahami penyimpanan data $\mathbf{dinamis}$ tanpa array, mengaplikasikan pointer dan pengelolaan memori C++, menerapkan operasi Linked List esensial, dan melatih pemrograman modular dengan memisahkan kode ke dalam file Playlist.h, Playlist.cpp, dan main.cpp, sekaligus melihat penerapan nyata Linked List pada aplikasi pemutar musik modern.

1. Referensi

<https://en.wikipedia.org/wiki/Linked_list>

<https://www.google.com/search?q=https://en.wikipedia.org/wiki/Linked_list%23Singly-linked_list>

<https://en.wikipedia.org/wiki/Linked_list>