LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 4
SINGLE LINKED LIST

Disusun Oleh :
NAMA : DEVI YULIANA
NIM : 103112400151

Dosen
FAHRUDIN MUKTI WIBOWO
STRUKTUR DATA

PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

- Linked List adalah struktur data yang berisi kumpulan elemen (node) yang saling terhubung lewat pointer. Tiap node menyimpan data dan alamat node berikutnya.
- Kenapa pakai pointer? Karena pointer membuat ukuran list bisa berubah-ubah (dinamis), jadi lebih fleksibel dibanding array yang ukurannya tetap.
- Singly Linked List adalah jenis linked list yang paling sederhana — tiap node hanya punya satu arah (dari depan ke belakang), dan node terakhir menunjuk ke NULL.
- Bagian-bagian penting:
    1. Head / First : menunjuk ke node pertama.
    2. Next : penghubung ke node berikutnya.
    3. Node : tempat menyimpan data.

- Operasi dasar:
    1. Create List : buat list kosong.
    2. Insert : tambah elemen (bisa di awal, akhir, atau setelah node tertentu).
    3. Delete : hapus elemen (awal, akhir, atau setelah node tertentu).
    4. View : tampilkan isi list.
    5. Update & Dealokasi : ubah atau hapus data dari memori.

- Kelebihan: fleksibel, mudah menambah atau menghapus data.
- Kekurangan: hanya bisa dibaca maju, dan pencarian data agak lambat karena harus ditelusuri satu per satu.

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

Guided 1

singlylist.cpp


LAPRAK WEEK 4 > singlylist.cpp > insertFirst(List &L, address)
#include "singlylist.h"
using namespace std;
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) {
// jika list kosong insert last sama dengan insert first
insertFirst(L, P);
} else {
// Jika list tidak kosong, cari elemen terakhir
address Last = L.first;
while (Last->next != Nil) {
Last = Last->next;
}
// sambungkan elemen terakhir ke elemen baru (p)
Last->next = P;
}
}
void printInfo(List L) {
address P = L.first;
if (P == Nil) {
std::cout << "List Kosong!" << std::endl;
} else {
while (P != Nil) {
std::cout << P->info << " ";
P = P->next;
}
std::cout << std::endl;
}
}
NAMA : DEVI YULIANA
NIM : 103112400151
KELAS : S1IF-12-06

singlylist.h

LAPRAK WEEK 4 > C singlylist.h > ...

```
1   #ifndef SINGLYLIST_H_INCLUDED
2   #define SINGLYLIST_H_INCLUDED
3
4   #include <iostream>
5
6   #define Nil NULL
7
8   typedef int infotype;
9   typedef struct Elmist *address;
10
11  struct Elmist {
12      infotype info;
13      address next;
14  };
15
16  struct List {
17      address first;
18  };
19
20  // Deklarasi Prosedur dan Fungsi Primitif
21  void CreateList(List &L);
22  address alokasi(infotype x);
23  void dealokasi(address &P);
24  void insertFirst(List &L, address P);
25  void printInfo(List L);
26
27  #endif
```

NAMA : DEVI YULIANA
NIM : 103112400151
KELAS : S1IF-12-06

main.cpp

```cpp
#include<iostream>
#include<cstdlib>
#include "singlylist.h"
#include "singlylist.cpp"
using namespace std;
int main(){
    List L;
    CreateList(L);
    cout<<"Mengisi List menggunakan interLast..."<<endl;
    address P;

    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;
}
```

NAMA : DEVI YULIANA
NIM : 103112400151
KELAS : S1IF-12-06

Screenshots Output

```
PS C:\Users\musli\Downloads\SEMESTER 3\SEMESTER3> & 'c:\Users\musli\.vscode\extensions\ms-vscode.cpptools-1.28.3-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-rfgykpfi.y4r' '--stdout=Microsoft-MIEngine-Out-t3s5fxoy.gww' '--stderr=Microsoft-MIEngine-Error-3pz1tx3x.g3x' '--pid=Microsoft-MIEngine-Pid-sqp0ep1b.vog' '--dbgExe=C:\Users\musli\mingw32\bin\gdb.exe' '--interpreter=mi'
Mengisi List menggunakan interLast...
Isi List sekarang adalah : 9 12 8 0 2
Press any key to continue . . .
```

NAMA : DEVI YULIANA
NIM : 103112400151
KELAS : S1IF-12-06

Deskripsi :

Program ini berfungsi untuk menyimpan data secara dinamis, di mana tiap elemen (node) terhubung lewat pointer tanpa batasan ukuran tetap seperti array.

Struktur datanya terdiri dari ElmList (yang menyimpan data dan alamat elemen berikutnya) dan List (yang menyimpan alamat elemen pertama). Program juga punya beberapa fungsi dasar seperti CreateList buat inisialisasi, alokasi buat membuat node baru, insertLast buat nambah data di akhir list, dan printInfo buat menampilkan isi list.

Di bagian utama (main), program membuat list kosong, lalu menambahkan beberapa data (9, 12, 8, 0, dan 2). Setelah semua elemen dimasukkan, program menampilkan seluruh isi list ke layar. Singkatnya, program ini menunjukkan cara membuat, menambah, dan menampilkan data menggunakan struktur linked list.

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

Unguided 1


```h
#ifndef PLAYLIST_H_INCLUDED
#define PLAYLIST_H_INCLUDED
#include <iostream>
#include <string>
using namespace std;

// Struktur data lagu
struct Lagu {
    string judul;
    string penyanyi;
    float durasi; // dalam menit
};

// Node untuk setiap lagu
struct Node {
    Lagu data;
    Node *next;
};

// Playlist (linked list)
struct Playlist {
    Node *head;
};

// Fungsi dasar
void buatPlaylist(Playlist &P);
bool cekKosong(Playlist P);
Node* buatNode(Lagu laguBaru);
void hapusNode(Node* node);

// Operasi tambah lagu
void tambahDepan(Playlist &P, Node* node);
void tambahBelakang(Playlist &P, Node* node);
void tambahSetelah(Playlist &P, Node* node, int posisi);

// Operasi hapus lagu
void hapusBerdasarkanJudul(Playlist &P, string judul);

// Operasi tampilkan
void tampilkanPlaylist(Playlist P);

#endif
```

week 4 > playlist.cpp > ...

```cpp
#include "Playlist.h"

// Membuat playlist kosong
void buatPlaylist(Playlist &P) {
    P.head = NULL;
}

// Mengecek apakah playlist kosong
bool cekKosong(Playlist P) {
    return (P.head == NULL);
}

// Membuat node baru
Node* buatNode(Lagu LaguBaru) {
    Node* node = new Node;
    node->data = LaguBaru;
    node->next = NULL;
    return node;
}

// Menghapus node dari memori
void hapusNode(Node* node) {
    delete node;
}

// Menambah lagu di awal playlist
void tambahDepan(Playlist &P, Node* node) {
    if (cekKosong(P)) {
        P.head = node;
    } else {
        node->next = P.head;
        P.head = node;
    }
}

// Menambah lagu di akhir playlist
void tambahBelakang(Playlist &P, Node* node) {
    if (cekKosong(P)) {
        P.head = node;
    } else {
        Node* bantu = P.head;
        while (bantu->next != NULL) {
            bantu = bantu->next;
        }
        bantu->next = node;
    }
}

// Menambah lagu setelah posisi tertentu
void tambahSetelah(Playlist &P, Node* node, int posisi) {
    if (cekKosong(P)) {
        cout << "Playlist masih kosong!\n";
        return;
    }

    Node* bantu = P.head;
    int i = 1;
    while (bantu != NULL && i < posisi) {
```

NAMA : DEVI YULIANA
NIM : 103112400151
KELAS : S1IF-12-06

```cpp
// Menambah lagu setelah posisi tertentu
void tambahSetelah(Playlist &P, Node* node, int posisi) {
    if (cekKosong(P)) {
        cout << "Playlist masih kosong!\n";
        return;
    }

    Node* bantu = P.head;
    int i = 1;
    while (bantu != NULL && i < posisi) {
        bantu = bantu->next;
        i++;
    }

    if (bantu != NULL) {
        node->next = bantu->next;
        bantu->next = node;
    } else {
        cout << "Posisi tidak ditemukan!\n";
    }
}

// Menghapus lagu berdasarkan judul
void hapusBerdasarkanJudul(Playlist &P, string judul) {
    if (cekKosong(P)) {
        cout << "Playlist kosong.\n";
        return;
    }

    Node *hapus = P.head, *sebelum = NULL;
    while (hapus != NULL && hapus->data.judul != judul) {
        sebelum = hapus;
        hapus = hapus->next;
    }

    if (hapus == NULL) {
        cout << "Lagu \"" << judul << "\" tidak ditemukan.\n";
        return;
    }

    if (sebelum == NULL) { // hapus di awal
        P.head = hapus->next;
    } else {
        sebelum->next = hapus->next;
    }
```

NAMA : DEVI YULIANA
NIM : 103112400151
KELAS : S1IF-12-06

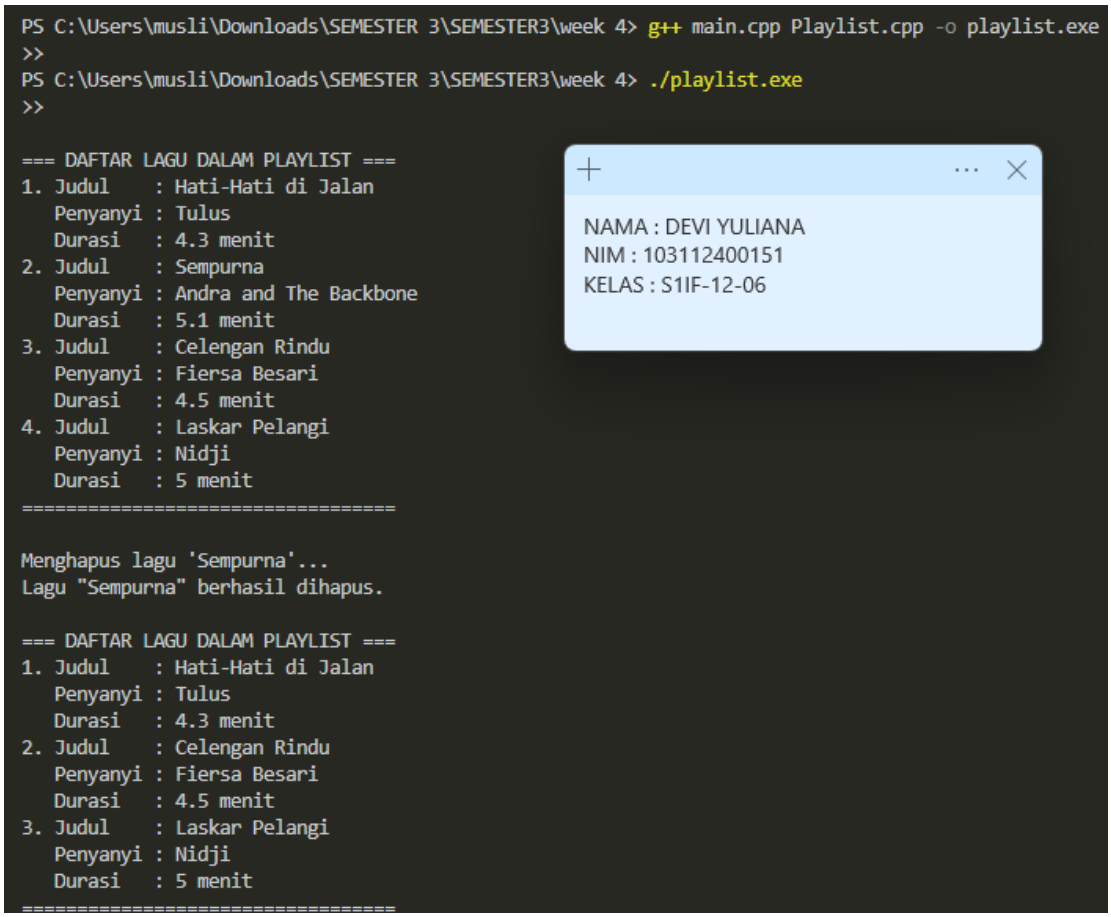
```
    hapusNode(hapus);
    cout << "Lagu \"" << judul << "\" berhasil dihapus.\n";
}

// Menampilkan seluruh playlist
void tampilkanPlaylist(Playlist P) {
    if (cekKosong(P)) {
        cout << "Playlist kosong.\n";
    } else {
        Node* bantu = P.head;
        int i = 1;
        cout << "\n=== DAFTAR LAGU DALAM PLAYLIST ===\n";
        while (bantu != NULL) {
            cout << i << ". Judul    : " << bantu->data.judul << endl;
            cout << "   Penyanyi : " << bantu->data.penyanyi << endl;
            cout << "   Durasi   : " << bantu->data.durasi << " menit\n";
            bantu = bantu->next;
            i++;
        }
        cout << "================================\n";
    }
}
```

NAMA : DEVI YULIANA
NIM : 103112400151
KELAS : S1IF-12-06

```
#include "Playlist.h"

int main() {
    Playlist playlistSaya;
    buatPlaylist(playlistSaya);

    Lagu l1 = {"Hati-Hati di Jalan", "Tulus", 4.3};
    Lagu l2 = {"Sempurna", "Andra and The Backbone", 5.1};
    Lagu l3 = {"Celengan Rindu", "Fiersa Besari", 4.5};
    Lagu l4 = {"Laskar Pelangi", "Nidji", 5.0};

    // Tambah lagu sesuai perintah soal
    tambahDepan(playlistSaya, buatNode(l1));     // di awal
    tambahBelakang(playlistSaya, buatNode(l2));  // di akhir
    tambahBelakang(playlistSaya, buatNode(l3));  // di akhir
    tambahSetelah(playlistSaya, buatNode(l4), 3); // setelah lagu ke-3

    tampilkanPlaylist(playlistSaya);

    cout << "\nMenghapus lagu 'Sempurna'...\n";
    hapusBerdasarkanJudul(playlistSaya, "Sempurna");

    tampilkanPlaylist(playlistSaya);

    return 0;
}
```

NAMA : DEVI YULIANA
NIM : 103112400151
KELAS : S1IF-12-06

Screenshots Output



Deskripsi:

Jadi, playlist ini disusun dari beberapa node yang saling terhubung, di mana setiap node berisi data lagu seperti judul, penyanyi, dan durasi. Program ini bisa menambah lagu di bagian awal, akhir, atau di posisi tertentu dalam playlist, serta bisa menghapus lagu berdasarkan judulnya.

Awalnya, program membuat playlist kosong, lalu menambahkan beberapa lagu contoh seperti “Hati-Hati di Jalan”, “Sempurna”, “Celengan Rindu”, dan “Laskar Pelangi”. Setelah lagu-lagu itu dimasukkan, program akan menampilkan daftar lagu yang ada di playlist. Kemudian, lagu berjudul “Sempurna” dihapus dari daftar, dan playlist ditampilkan lagi untuk menunjukkan hasil akhirnya.

Secara sederhana, program ini membantu kita memahami gimana cara kerja linked list dalam bentuk yang lebih nyata, yaitu daftar lagu. Dengan cara ini, kita bisa nambah, hapus, atau lihat lagu-lagu di playlist tanpa ribet menggeser-geser data seperti kalau pakai array.

D. Kesimpulan

Singly Linked List adalah struktur data yang efisien dan fleksibel untuk menyimpan data yang jumlahnya bisa berubah-ubah. Dengan konsep node dan pointer, kita bisa menambah, menghapus, atau menampilkan data dengan lebih mudah dibanding array. Walau cara kerjanya butuh ketelitian dalam mengatur pointer, konsep ini sangat penting untuk memahami struktur data tingkat lanjut seperti stack, queue, dan tree.

E. Referensi

Astuti, I. K. (2019). *STRUKTUR DATA LINKED LIST*.

Soetanto, H. (2022). *Struktur Data*.