

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

**MODUL IV DAN V
SINGLY LINKED LIST**



Disusun Oleh :

NAMA : Balawan Satria Lhaksana Putra M.

NIM : 103112430004

Dosen

FAHRUDIN MUKTI WIBOWO

**PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

A. Dasar Teori

Single Linked List merupakan salah satu bentuk struktur data dinamis yang tersusun dari elemen-elemen (node) yang saling terhubung menggunakan pointer. Setiap node dalam linked list biasanya terdiri dari dua bagian utama: data (yang menyimpan nilai atau informasi) dan pointer next (yang menunjuk ke node berikutnya). Berbeda dengan array, linked list tidak memerlukan alokasi memori statis, sehingga ukuran datanya dapat bertambah atau berkurang sesuai kebutuhan program.

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

Guided 1 (Singlylist.h)

```
#ifndef SINGLYLIST_H_INCLUDED
#define SINGLYLIST_H_INCLUDED

#include <iostream>

#define Nil NULL

typedef int 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 printInfo(list L);

#endif
```

Guided 1 (Singlylist.cpp)

```
#include "Singlylist.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){
        insertFirst(L, P);
    } else{
        address Last = L.first;
        while(Last->next != Nil){
            Last = Last->next;
        }
        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;
    }
}
}

```

Guided 1 (main.cpp)

```

#include "Singlylist.h"
#include "Singlylist.cpp"

```

```

#include <iostream>
#include <cstdlib>

using namespace std;

int main() {
    list L;
    address P;

    createList(L);
    cout << "Mengisi list menggunakan insertLast" << endl;

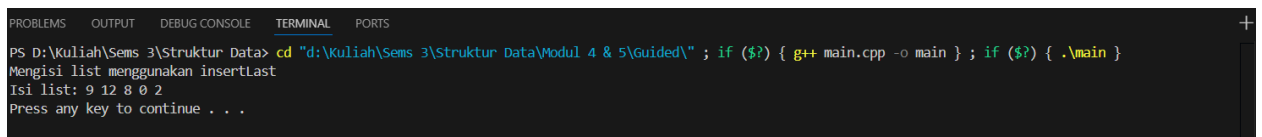
    //Mengisi list sesuai urutan
    P = alokasi(9);
    insertLast(L, P);
    P = alokasi(12);
    insertLast(L, P);
    P = alokasi(8);
    insertLast(L, P);
    P = alokasi(0);
    insertLast(L, P);
    P = alokasi(2);
    insertLast(L, P);

    cout << "Isi list: ";
    printInfo(L);

    system("pause");
    return 0;
}

```

Screenshots Output



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Kuliah\Sems 3\Struktur Data> cd "d:\Kuliah\Sems 3\Struktur Data\Modul 4 & 5\Guided\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
Mengisi list menggunakan insertLast
Isi list: 9 12 8 0 2
Press any key to continue . . .

```

Deskripsi:

Program ini merupakan implementasi Singly Linked List menggunakan tiga file terpisah, yaitu header (Singlylist.h), implementasi (Singlylist.cpp), dan program utama (main.cpp). Pada file header, didefinisikan struktur data `elmList` yang berisi info bertipe `int` dan pointer `next`, serta struktur `list` yang menyimpan pointer `first` sebagai penanda elemen pertama dalam list.

File implementasi berisi fungsi-fungsi utama seperti `createList()` untuk menginisialisasi

list kosong, alokasi() untuk membuat node baru, dealokasi() untuk menghapus node, insertFirst() untuk menambahkan elemen di awal list, insertLast() untuk menambahkan elemen di akhir list, dan printInfo() untuk menampilkan seluruh isi list.

Pada program utama, list L dibuat kosong terlebih dahulu, kemudian lima data (9, 12, 8, 0, dan 2) ditambahkan ke dalam list menggunakan fungsi insertLast(). Hasil akhirnya, program menampilkan seluruh isi list secara berurutan di layar.

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

Unguided 1 (Playlist.h)

```
#ifndef PLAYLIST_H_INCLUDED
#define PLAYLIST_H_INCLUDED

#include <iostream>
#include <string>
using namespace std;

struct Lagu {
    string judul;
    string penyanyi;
    float durasi;
    Lagu* next;
};

struct Playlist {
    Lagu* first;
};

void createPlaylist(Playlist &P);
Lagu* alokasi(string judul, string penyanyi, float durasi);
void insertFirst(Playlist &P, Lagu* L);
void insertLast(Playlist &P, Lagu* L);
void insertAfter3(Playlist &P, Lagu* L);
void deleteByJudul(Playlist &P, string judul);
void printPlaylist(Playlist P);

#endif
```

Unguided 1 (Playlist.cpp)

```
#include "Playlist.h"

// Inisialisasi playlist kosong
void createPlaylist(Playlist &P) {
    P.first = NULL;
}
```

```

// Buat node lagu baru
Lagu* alokasi(string judul, string penyanyi, float durasi) {
    Lagu* L = new Lagu;
    L->judul = judul;
    L->penyanyi = penyanyi;
    L->durasi = durasi;
    L->next = NULL;
    return L;
}

// Tambah lagu di awal
void insertFirst(Playlist &P, Lagu* L) {
    L->next = P.first;
    P.first = L;
}

// Tambah lagu di akhir
void insertLast(Playlist &P, Lagu* L) {
    if (P.first == NULL) {
        insertFirst(P, L);
    } else {
        Lagu* temp = P.first;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = L;
    }
}

// Tambah lagu setelah lagu ke-3
void insertAfter3(Playlist &P, Lagu* L) {
    Lagu* temp = P.first;
    int count = 1;
    while (temp != NULL && count < 3) {
        temp = temp->next;
        count++;
    }
    if (temp != NULL) {
        L->next = temp->next;
        temp->next = L;
    } else {
        cout << "Playlist kurang dari 3 lagu!" << endl;
    }
}

// Hapus lagu berdasarkan judul
void deleteByJudul(Playlist &P, string judul) {
    Lagu *temp = P.first, *prev = NULL;

```

```

        while (temp != NULL && temp->judul != judul) {
            prev = temp;
            temp = temp->next;
        }

        if (temp == NULL) {
            cout << "Lagu dengan judul \"" << judul << "\" tidak ditemukan!"
<< endl;
            return;
        }

        if (prev == NULL) {
            P.first = temp->next;
        } else {
            prev->next = temp->next;
        }

        delete temp;
        cout << "Lagu \"" << judul << "\" berhasil dihapus.\n";
    }

    // Menampilkan seluruh lagu dalam playlist
    void printPlaylist(Playlist P) {
        if (P.first == NULL) {
            cout << "Playlist kosong!\n";
        } else {
            Lagu* temp = P.first;
            int i = 1;
            cout << "\n=== Daftar Lagu dalam Playlist ===\n";
            while (temp != NULL) {
                cout << i++ << ". Judul    : " << temp->judul << endl;
                cout << "    Penyanyi: " << temp->penyanyi << endl;
                cout << "    Durasi   : " << temp->durasi << " menit\n";
                temp = temp->next;
            }
            cout << endl;
        }
    }
}

```

Unguided 1 (main.cpp)

```

#include <iostream>
#include "Playlist.h"
#include "Playlist.cpp"
using namespace std;

int main() {
    Playlist P;
    createPlaylist(P);
}

```

```

int pilihan;
string judul, penyanyi;
float durasi;

do {
    cout << "\n=== MENU PLAYLIST LAGU ===\n";
    cout << "1. Tambah lagu di awal\n";
    cout << "2. Tambah lagu di akhir\n";
    cout << "3. Tambah lagu setelah lagu ke-3\n";
    cout << "4. Hapus lagu berdasarkan judul\n";
    cout << "5. Tampilkan semua lagu\n";
    cout << "0. Keluar\n";
    cout << "Pilih menu: ";
    cin >> pilihan;
    cin.ignore();

    switch (pilihan) {
        case 1:
            cout << "Masukkan judul lagu   : ";
            getline(cin, judul);
            cout << "Masukkan nama penyanyi: ";
            getline(cin, penyanyi);
            cout << "Masukkan durasi (menit): ";
            cin >> durasi;
            insertFirst(P, alokasi(judul, penyanyi, durasi));
            break;

        case 2:
            cout << "Masukkan judul lagu   : ";
            getline(cin, judul);
            cout << "Masukkan nama penyanyi: ";
            getline(cin, penyanyi);
            cout << "Masukkan durasi (menit): ";
            cin >> durasi;
            insertLast(P, alokasi(judul, penyanyi, durasi));
            break;

        case 3:
            cout << "Masukkan judul lagu   : ";
            getline(cin, judul);
            cout << "Masukkan nama penyanyi: ";
            getline(cin, penyanyi);
            cout << "Masukkan durasi (menit): ";
            cin >> durasi;
            insertAfter3(P, alokasi(judul, penyanyi, durasi));
            break;
    }
}

```



```

        case 4:
            cout << "Masukkan judul lagu yang ingin dihapus: ";
            getline(cin, judul);
            deleteByJudul(P, judul);
            break;

        case 5:
            printPlaylist(P);
            break;

        case 0:
            cout << "Terima kasih! Program selesai.\n";
            break;

        default:
            cout << "Pilihan tidak valid!\n";
    }
} while (pilihan != 0);

return 0;
}

```

Screenshots Output

Tampilan Awal

The screenshot shows the initial output of the program. The terminal window displays the menu options for the playlist application. The user has selected option 1, 'Tambah lagu di awal'.

```

D:\Kuliah\Sems 3\Struktur Data\Modul 4 & 5\Unguided>.\main

=== MENU PLAYLIST LAGU ===
1. Tambah lagu di awal
2. Tambah lagu di akhir
3. Tambah lagu setelah lagu ke-3
4. Hapus lagu berdasarkan judul
5. Tampilkan semua lagu
0. Keluar
Pilih menu: █

```

Tambah

The screenshot shows the program's output after adding two songs. The user has selected option 1 twice, entering the song title, artist, and duration for each.

```

Pilih menu: 1
Masukkan judul lagu   : Blue Jeans
Masukkan nama penyanyi: Gangga
Masukkan durasi (menit): 4.00

=== MENU PLAYLIST LAGU ===
1. Tambah lagu di awal
2. Tambah lagu di akhir
3. Tambah lagu setelah lagu ke-3
4. Hapus lagu berdasarkan judul
5. Tampilkan semua lagu
0. Keluar
Pilih menu: 1
Masukkan judul lagu   : Terbuang Dalam Waktu
Masukkan nama penyanyi: Barasuara
Masukkan durasi (menit): 4.41

```

Tampilkan

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
=== MENU PLAYLIST LAGU ===
1. Tambah lagu di awal
2. Tambah lagu di akhir
3. Tambah lagu setelah lagu ke-3
4. Hapus lagu berdasarkan judul
5. Tampilkan semua lagu
0. Keluar
Pilih menu: 5

=== Daftar Lagu dalam Playlist ===
1. Judul : Terbuang Dalam Waktu
   Penyanyi: Barasuara
   Durasi : 4.41 menit
2. Judul : Blue Jeans
   Penyanyi: Gangga
   Durasi : 4 menit
```

Hapus

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
=== MENU PLAYLIST LAGU ===
1. Tambah lagu di awal
2. Tambah lagu di akhir
3. Tambah lagu setelah lagu ke-3
4. Hapus lagu berdasarkan judul
5. Tampilkan semua lagu
0. Keluar
Pilih menu: 4
Masukkan judul lagu yang ingin dihapus: Blue Jeans
Lagu "Blue Jeans" berhasil dihapus.

=== MENU PLAYLIST LAGU ===
1. Tambah lagu di awal
2. Tambah lagu di akhir
3. Tambah lagu setelah lagu ke-3
4. Hapus lagu berdasarkan judul
5. Tampilkan semua lagu
0. Keluar
Pilih menu: 5

=== Daftar Lagu dalam Playlist ===
1. Judul : Terbuang Dalam Waktu
   Penyanyi: Barasuara
   Durasi : 4.41 menit
```

Deskripsi:

Program ini merupakan implementasi Single Linked List dalam bahasa C++ yang digunakan untuk mengelola playlist lagu secara dinamis. Data setiap lagu disimpan dalam node yang memiliki atribut judul, penyanyi, dan durasi, serta pointer next yang menunjuk ke lagu berikutnya. Program ini dibagi menjadi tiga file, yaitu Playlist.h untuk deklarasi struktur dan fungsi, Playlist.cpp untuk implementasi fungsi, dan main.cpp sebagai program utama.

Pada file **Playlist.cpp**, terdapat beberapa operasi utama, yaitu:

- `createPlaylist()` untuk membuat playlist kosong,
- `alokasi()` untuk membuat node baru berisi data lagu,
- `insertFirst()` menambah lagu di awal playlist,
- `insertLast()` menambah lagu di akhir playlist,
- `insertAfter3()` menambah lagu setelah lagu ke-3,
- `deleteByJudul()` menghapus lagu berdasarkan judul, dan
- `printPlaylist()` menampilkan seluruh lagu dalam playlist.

Sedangkan pada file `main.cpp`, program menampilkan menu interaktif yang

memungkinkan pengguna menambah lagu, menghapus lagu, atau melihat seluruh daftar lagu. Program berjalan dalam loop hingga pengguna memilih keluar. Dengan struktur ini, program menggambarkan penggunaan pointer dan linked list untuk menyimpan data secara fleksibel tanpa batasan jumlah elemen yang tetap.

D. Kesimpulan

Dari percobaan ini dapat disimpulkan bahwa Single Linked List merupakan struktur data yang efisien untuk mengelola kumpulan data yang bersifat dinamis. Melalui penerapan pada program Playlist Lagu, konsep dasar seperti alokasi memori dinamis, penggunaan pointer, serta operasi penambahan dan penghapusan node dapat dipahami dengan baik. Penggunaan tiga file terpisah (header, implementasi, dan main program) juga membantu dalam modularisasi kode, sehingga program menjadi lebih terstruktur, mudah dibaca, dan dapat dikembangkan lebih lanjut.

E. Referensi

- https://daismabali.com/artikel_detail/54/1/Mengenal-Single-Linked-List-dalam-Struktur-Data.html
- <https://www.trivusi.web.id/2022/07/struktur-data-linked-list.html>
- <https://terapan-ti.vokasi.unesa.ac.id/post/memahami-konsep-dan-jenis-jenis-linked-list-dalam-struktur-data>