

**LAPORAN PRAKTIKUM  
STRUKTUR DATA**

**MODUL 4**



**Disusun Oleh :**

NAMA : IBTIDA ZADA UTOMO

NIM : 103112430037

**Dosen**

FAHRUDIN MUKTI WIBOWO

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

## A. Dasar Teori

Struktur data merupakan cara penyimpanan dan pengorganisasian data agar dapat digunakan secara efisien. Salah satu struktur data yang sering digunakan adalah *linked list*. *Linked list* adalah kumpulan elemen data yang disebut *node*, di mana setiap *node* berisi dua bagian utama: data itu sendiri dan pointer yang menunjuk ke *node* berikutnya. Salah satu jenis *linked list* adalah *single linked list*, yaitu daftar berantai di mana setiap *node* hanya memiliki satu pointer yang menunjuk ke *node* berikutnya. Berbeda dengan array yang menggunakan indeks untuk mengakses elemen, *linked list* memungkinkan penambahan dan penghapusan data secara dinamis tanpa harus menggeser elemen lainnya.

Dalam *single linked list*, terdapat beberapa operasi dasar seperti pembuatan list kosong, penambahan data di awal atau di akhir list, penyisipan data di posisi tertentu, penghapusan node, serta pencetakan isi list. Implementasi *single linked list* banyak digunakan dalam berbagai aplikasi, seperti manajemen antrian, sistem navigasi, dan juga pengelolaan playlist lagu. Pada program ini, *single linked list* digunakan untuk mengelola playlist lagu, di mana setiap lagu disimpan sebagai satu *node* dengan atribut berupa judul, penyanyi, dan durasi. Struktur ini memungkinkan pengguna untuk menambah, menghapus, dan menampilkan daftar lagu secara fleksibel dan efisien tanpa batasan ukuran tetap seperti pada array.

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

### Guided 1

singlyst.h

```
#ifndef SINGLYLISH_H_INLCLUDED
#define SINGLYLISH_H_INLCLUDED
#include <iostream>
#define NIL NULL

typedef int infotype;
typedef struct ElmList *address;

struct ElmList {
    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

```

singlyst.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) {
        // Jika list kosong, insertLast sama dengan insertFirst
        insertFirst(L, P);
    } else {

```

```

        // Jika list tidak kosong, cari element terakhir
        address Last = L.first;
        while (Last->next != NIL) {
            Last = Last->next;
        }
        // Sambungkan elemen terakhir dengan elemen terbaru (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;
    }
}
}

```

## 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 dignakan 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);

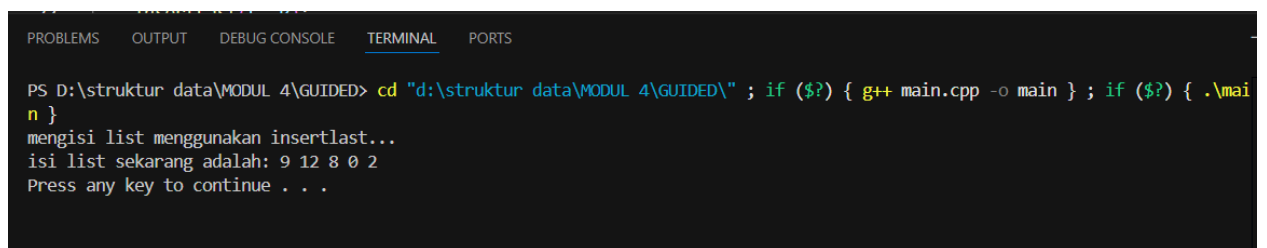
    P = alokasi(2);
    insertLast(L, P);

    cout << "isi list sekarang adalah: ";
    printInfo(L);

    system("pause");
    return 0;
}

```

## Screenshots Output



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\struktur data\MODUL 4\GUIDED> cd "d:\struktur data\MODUL 4\GUIDED\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\mai
n }
mengisi list menggunakan insertlast...
isi list sekarang adalah: 9 12 8 0 2
Press any key to continue . . .

```

Deskripsi: program diatas adalah program untuk implementasi dari struktur data single linked list tujuannya untuk menyimpan, menambah, dan menampilkan data menggunakan konsep linked list

## Unguided 1

### playlist.h

```

#ifndef PLAYLIST_H
#define PLAYLIST_H

```

```

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

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

class Playlist {
private:
    Lagu* head;

public:
    Playlist();
    ~Playlist();

    void tambahDepan(string judul, string penyanyi, float durasi);
    void tambahBelakang(string judul, string penyanyi, float
durasi);
    void tambahSetelahKe3(string judul, string penyanyi, float
durasi);
    void hapusLagu(string judul);
    void tampilkan();
};

#endif

```

#### playlist.cpp

```

#include "Playlist.h"

Playlist::Playlist() {
    head = nullptr;
}

Playlist::~~Playlist() {
    Lagu* current = head;

```

```

        while (current != nullptr) {
            Lagu* temp = current;
            current = current->next;
            delete temp;
        }
    }

void Playlist::tambahDepan(string judul, string penyanyi, float
durasi) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, nullptr};
    baru->next = head;
    head = baru;
}

void Playlist::tambahBelakang(string judul, string penyanyi, float
durasi) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, nullptr};
    if (head == nullptr) {
        head = baru;
        return;
    }

    Lagu* temp = head;
    while (temp->next != nullptr)
        temp = temp->next;
    temp->next = baru;
}

void Playlist::tambahSetelahKe3(string judul, string penyanyi,
float durasi) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, nullptr};

    Lagu* temp = head;
    int count = 1;
    while (temp != nullptr && count < 3) {
        temp = temp->next;
        count++;
    }

    if (temp == nullptr) {
        cout << "Playlist kurang dari 3 lagu.\n";
        delete baru;
        return;
    }
}

```

```

    }

    baru->next = temp->next;
    temp->next = baru;
}

void Playlist::hapusLagu(string judul) {
    if (head == nullptr) {
        cout << "Playlist kosong!\n";
        return;
    }

    if (head->judul == judul) {
        Lagu* hapus = head;
        head = head->next;
        delete hapus;
        cout << "Lagu \"" << judul << "\" berhasil dihapus.\n";
        return;
    }

    Lagu* temp = head;
    while (temp->next != nullptr && temp->next->judul != judul)
        temp = temp->next;

    if (temp->next == nullptr) {
        cout << "Lagu tidak ditemukan.\n";
        return;
    }

    Lagu* hapus = temp->next;
    temp->next = hapus->next;
    delete hapus;
    cout << "Lagu \"" << judul << "\" berhasil dihapus.\n";
}

void Playlist::tampilkan() {
    if (head == nullptr) {
        cout << "Playlist kosong!\n";
        return;
    }

    Lagu* temp = head;
    int i = 1;

```



```

        cout << "\nDaftar Lagu dalam Playlist:\n";
        while (temp != nullptr) {
            cout << i++ << ". " << temp->judul << " - " <<
temp->penyanyi
            << " (" << temp->durasi << " menit)\n";
            temp = temp->next;
        }
    }
}

```

main.cpp

```

#include "Playlist.h"
#include "playlist.cpp"

int main() {
    Playlist myPlaylist;
    int pilihan;
    string judul, penyanyi;
    float durasi;

    do {
        cout << "\n=== MENU PLAYLIST ===\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 seluruh lagu\n";
        cout << "0. Keluar\n";
        cout << "Pilih: ";
        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;
                myPlaylist.tambahDepan(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;
            myPlaylist.tambahBelakang(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;
            myPlaylist.tambahSetelahKe3(judul, penyanyi,
durasi);

            break;

        case 4:
            cout << "Masukkan judul lagu yang akan dihapus: ";
            getline(cin, judul);
            myPlaylist.hapusLagu(judul);
            break;

        case 5:
            myPlaylist.tampilkan();
            break;

        case 0:
            cout << "Keluar dari program.\n";
            break;

        default:
            cout << "Pilihan tidak valid!\n";

    }
} while (pilihan != 0);
return 0;
}

```

## Screenshots Output

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\struktur data\MODUL 4\UNGUIDED> cd "d:\struktur data\MODUL 4\UNGUIDED\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }

=== MENU PLAYLIST ===
1. Tambah lagu di awal
2. Tambah lagu di akhir
3. Tambah lagu setelah lagu ke-3
4. Hapus lagu berdasarkan judul
5. Tampilkan seluruh lagu
0. Keluar
Pilih: 1
Masukkan judul lagu: monokrom
Masukkan nama penyanyi: tulus
Masukkan durasi (menit): 3

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

Daftar Lagu dalam Playlist:
1. monokrom - tulus (3 menit)

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

Deskripsi: program diatas adalah program playlist lagu, jadi nanti kita bisa menambahkan lagu, dan tidak hanya menambahkan saja ada beberapa menu nya seperti tambah lagu di awal, tambah lagu di akhir, tambah lagu setelah lagu ketiga, hapus lagu berdasarkan judul, tampilkan seluruh lagi, dan keluar, jadi misal kita ingin menambahkan lagu makam pilih menu pertama nanti kita disuruh memasukan judul, lalu penyanyi nya , dan berapa menit durasi nya, dan jika memilih menu nomer 5 maka program akan menampilkan seluruh data lagu yang pernah dimasukan ke dalam data

### C. Kesimpulan

Dari percobaan yang dilakukan, dapat disimpulkan bahwa *single linked list* merupakan struktur data yang efisien untuk menyimpan dan mengelola data secara dinamis. Dengan menggunakan *linked list*, proses penambahan maupun penghapusan data dapat dilakukan tanpa harus memindahkan elemen lain seperti pada array. Program playlist lagu yang dibuat menunjukkan penerapan nyata dari konsep *single linked list*, di mana pengguna dapat menambahkan lagu di awal, di akhir, atau setelah lagu ke-3, serta menghapus lagu berdasarkan judul dan menampilkan seluruh daftar lagu. Implementasi ini membantu memahami cara kerja pointer dan hubungan antar-node dalam membentuk suatu rangkaian data yang saling terhubung.

#### D. Referensi

- ["ISO/IEC TS 19841:2015"](#). International Organization for Standardization. [Archived](#) from the original on 15 January 2019. Retrieved 15 February 2019.
- ["ISO/IEC TS 19568:2015"](#). International Organization for Standardization. [Archived](#) from the original on 15 January 2019. Retrieved 15 February 2019.
- ["ISO/IEC TS 19217:2015"](#). International Organization for Standardization. [Archived](#) from the original on 15 January 2019. Retrieved 15 February 2019.