

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

**MODUL 4 & 5
SINGLY LINKED LIST**



Disusun Oleh :

NAMA : M.IRFAN ADIB

NIM : 103112400257

Dosen

FAHRUDIN MUKTI WIBOWO

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

A. Dasar Teori

Linked list (senarai berkait) merupakan struktur data fleksibel yang terdiri dari serangkaian elemen data yang saling terhubung, memungkinkan pertumbuhan dan penyusutan dinamis, berbeda dengan array yang statis. Implementasi linked list sering menggunakan pointer karena sifat dinamisnya cocok untuk data yang saling bergantung. Operasi dasarnya mencakup pembuatan list, penyisipan, penghapusan, dan penelusuran elemen. Model yang dibahas adalah Singly Linked List, sebuah ADT di mana setiap elemen, disebut Node (simpul), memiliki dua bagian: data utama dan pointer 'next' yang menunjuk ke alamat elemen berikutnya. Pointer khusus, 'first' atau 'head', menunjuk ke elemen pertama, dan list berakhir ketika elemen terakhir memiliki pointer 'next' yang menunjuk ke NULL (Nil).

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

Guided 1

```
#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;
};

// 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
```

```
#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;
    }
}

```

```
#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 interLast..." << 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

```
Mengisi list menggunakan insertLast...  
Isi list sekarang adalah: 9 12 8 0 2  
Press any key to continue . . .
```

Deskripsi:

Program C++ ini mengimplementasikan struktur data Singly Linked List untuk mengelola urutan data bilangan bulat (infotype). Struktur ini didefinisikan melalui Node (ElmList) yang berisi data (info) dan pointer ke elemen berikutnya (next), serta struktur list yang hanya menyimpan pointer ke elemen pertama (first). Kode ini menyediakan fungsi-fungsi dasar (primitif) seperti membuat list kosong (createList), mengalokasikan Node baru (alokasi), membebaskan memori Node (dealokasi), serta menambahkan elemen di awal (insertFirst) dan di akhir list (insertLast). Fungsi printInfo digunakan untuk menampilkan seluruh isi list. Dalam fungsi utama (main), list dibuat, lalu diisi dengan nilai 9, 12, 8, 0, dan 2 secara berurutan menggunakan insertLast, dan kemudian hasilnya dicetak ke konsol.

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

Unguided 1

```
#ifndef PLAYLST_H
#define PLAYLST_H

#include <string>
#include <iostream>

using namespace std;

struct Node {
    string judul;
    string penyanyi;
    float durasi;

    Node* next;
};

class Playlist {
private:
    Node* head;
public:
    Playlist();
    void tambahLaguDiAwal(string judul, string penyanyi, float durasi);
    void tambahLaguDiAkhir(string judul, string penyanyi, float durasi);
    void tambahLaguSetelahKetiga(string judul, string penyanyi, float durasi);
    void hapusLagu(string judul);
    void tampilkanPlaylist();
};

#endif
```

```

#include "Playlist.h"

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

void Playlist::tambahLaguDiAwal(string judul, string penyanyi,
float durasi) {
    Node* newNode = new Node;

    newNode->judul = judul;
    newNode->penyanyi = penyanyi;
    newNode->durasi = durasi;

    newNode->next = head;
    head = newNode;

    cout << "Berhasil menambah '" << judul << "' di awal
playlist.\n";
}

void Playlist::tambahLaguDiAkhir(string judul, string penyanyi,
float durasi) {
    Node* newNode = new Node;
    newNode->judul = judul;
    newNode->penyanyi = penyanyi;
    newNode->durasi = durasi;
    newNode->next = nullptr;

    if (head == nullptr) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
    }

    cout << "Berhasil menambah '" << judul << "' di akhir
playlist.\n";
}

void Playlist::tambahLaguSetelahKetiga(string judul, string

```

```

    penyanyi, float durasi) {
        Node* temp = head;
        int counter = 1;

        while (counter < 3 && temp != nullptr) {
            temp = temp->next;
            counter++;
        }

        if (temp == nullptr) {
            cout << "Gagal! Playlist hanya memiliki " << counter-1 <<
" lagu, tidak bisa menambah setelah lagu ke-3.\n";
        } else {
            Node* newNode = new Node;
            newNode->judul = judul;
            newNode->penyanyi = penyanyi;
            newNode->durasi = durasi;

            newNode->next = temp->next;
            temp->next = newNode;

            cout << "Berhasil menambah '" << judul << "' setelah lagu
ke-3.\n";
        }
    }

void Playlist::hapusLagu(string judul) {
    if (head == nullptr) {
        cout << "Playlist kosong, tidak ada yang bisa dihapus.\n";
        return;
    }

    if (head->judul == judul) {
        Node* temp = head;
        head = head->next;
        delete temp;
        cout << "Berhasil menghapus '" << judul << "' dari
playlist.\n";
        return;
    }

    Node* previous = head;
    Node* current = head->next;

```

```

        while (current != nullptr && current->judul != judul) {
            previous = current;
            current = current->next;
        }

        if (current == nullptr) {
            cout << "Lagu '" << judul << "' tidak ditemukan di
playlist.\n";
        } else {
            previous->next = current->next;

            delete current;
            cout << "Berhasil menghapus '" << judul << "' dari
playlist.\n";
        }
    }
}

void Playlist::tampilkanPlaylist() {
    Node* temp = head;

    if (temp == nullptr) {
        cout << "Playlist saat ini kosong." << endl;
        return;
    }

    cout << "\n===== ISI PLAYLIST ANDA =====\n";
    int nomor = 1;
    while (temp != nullptr) {
        cout << nomor << ". Judul      : " << temp->judul << "\n";
        cout << "    Penyanyi : " << temp->penyanyi << "\n";
        cout << "    Durasi   : " << temp->durasi << " menit\n";
        cout << "-----\n";

        temp = temp->next;
        nomor++;
    }
    cout << "===== \n";
}

```

```

#include "Playlist.h"
#include <iostream>
#include <string>

using namespace std;

void tampilkanMenu() {
    cout << "\n=== MENU MANAJEMEN 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 Playlist\n";
    cout << "6. Keluar\n";
    cout << "=====\n";
    cout << "Masukkan pilihan Anda (1-6): ";
}

int main() {
    Playlist playlistSaya;
    int pilihan = 0;
    string judul, penyanyi;
    float durasi;

    do {
        tampilkanMenu();

        while (!(cin >> pilihan)) {
            cout << "Input tidak valid. Harap masukkan angka
(1-6): ";
            cin.clear();
            cin.ignore(1024, '\n');
        }

        cin.ignore(1024, '\n');

        switch (pilihan) {
            case 1:
                cout << "Masukkan Judul: ";
                getline(cin, judul);
                cout << "Masukkan Penyanyi: ";
                getline(cin, penyanyi);
                cout << "Masukkan Durasi (menit, cth: 3.5): ";

```

```

        cin >> durasi;
        playlistSaya.tambahLaguDiAwal(judul, penyanyi,
durasi);

        break;

    case 2:
        cout << "Masukkan Judul: ";
        getline(cin, judul);
        cout << "Masukkan Penyanyi: ";
        getline(cin, penyanyi);
        cout << "Masukkan Durasi (menit, cth: 3.5): ";
        cin >> durasi;
        playlistSaya.tambahLaguDiAkhir(judul, penyanyi,
durasi);

        break;

    case 3:
        cout << "Masukkan Judul: ";
        getline(cin, judul);
        cout << "Masukkan Penyanyi: ";
        getline(cin, penyanyi);
        cout << "Masukkan Durasi (menit, cth: 3.5): ";
        cin >> durasi;
        playlistSaya.tambahLaguSetelahKetiga(judul,
penyanyi, durasi);

        break;

    case 4:
        cout << "Masukkan Judul Lagu yang ingin dihapus:
";

        getline(cin, judul);
        playlistSaya.hapusLagu(judul);
        break;

    case 5:
        playlistSaya.tampilkanPlaylist();
        break;

    case 6:
        cout << "Terima kasih telah menggunakan program
ini. Sampai jumpa!\n";
        break;

```

```

        default:
            cout << "Pilihan tidak valid! Silakan pilih angka
dari 1 sampai 6.\n";
            break;
    }

    if (pilihan != 6) {
        cout << "\nTekan Enter untuk kembali ke menu...";
        if (pilihan == 1 || pilihan == 2 || pilihan == 3) {
            cin.ignore(1024, '\n');
        }
        cin.get();
    }

    } while (pilihan != 6);

    return 0;
}

```

Screenshots Output

```

=== MENU MANAJEMEN 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 Playlist
6. Keluar
=====
Masukkan pilihan Anda (1-6): 1
Masukkan Judul: halo halo bandung
Masukkan Penyanyi: budi
Masukkan Durasi (menit, cth: 3.5): 120
Berhasil menambah 'halo halo bandung' di awal playlist.

Tekan Enter untuk kembali ke menu...

```

```

=== MENU MANAJEMEN 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 Playlist
6. Keluar
=====
Masukkan pilihan Anda (1-6): 2
Masukkan Judul: balonku
Masukkan Penyanyi: asep
Masukkan Durasi (menit, cth: 3.5): 200
Berhasil menambah 'balonku' di akhir playlist.

Tekan Enter untuk kembali ke menu...

```

```
=== MENU MANAJEMEN 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 Playlist
6. Keluar
=====
Masukkan pilihan Anda (1-6): 3
Masukkan Judul: cicak di dinding
Masukkan Penyanyi: abdul
Masukkan Durasi (menit, cth: 3.5): 100
Gagal! Playlist hanya memiliki 2 lagu, tidak bisa menambah setelah lagu ke-3.

Tekan Enter untuk kembali ke menu...
```

```
=== MENU MANAJEMEN 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 Playlist
6. Keluar
=====
Masukkan pilihan Anda (1-6): 4
Masukkan Judul Lagu yang ingin dihapus: cicak di dinding
Lagu 'cicak di dinding' tidak ditemukan di playlist.

Tekan Enter untuk kembali ke menu...
```

```
=== MENU MANAJEMEN 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 Playlist
6. Keluar
=====
Masukkan pilihan Anda (1-6): 5

===== ISI PLAYLIST ANDA =====
1. Judul      : halo halo bandung
   Penyanyi   : budi
   Durasi     : 120 menit
-----
2. Judul      : balonku
   Penyanyi   : asep
   Durasi     : 200 menit
-----
=====

Tekan Enter untuk kembali ke menu...
```

```
=== MENU MANAJEMEN 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 Playlist
6. Keluar
=====
Masukkan pilihan Anda (1-6): 6
Terima kasih telah menggunakan program ini. Sampai jumpa!
```

Deskripsi:

Program C++ ini mengimplementasikan sebuah sistem manajemen Playlist menggunakan struktur data Singly Linked List untuk menyimpan lagu, di mana setiap Node (Node) menyimpan informasi detail lagu (judul, penyanyi, dan durasi). Kelas Playlist mengelola operasi-operasi dasar Linked List seperti inisialisasi list, penambahan lagu di awal (`tambahLaguDiAwal`), penambahan di akhir (`tambahLaguDiAkhir`), penambahan di posisi tertentu (setelah lagu ketiga, `tambahLaguSetelahKetiga`), penghapusan lagu berdasarkan judul (`hapusLagu`), dan menampilkan seluruh isi playlist (`tampilkanPlaylist`). Fungsi main menyediakan antarmuka menu interaktif bagi pengguna untuk berinteraksi dengan playlist, memungkinkan pengguna untuk memilih dan menjalankan fungsi-fungsi manajemen lagu yang telah disediakan.

D. Kesimpulan

Berdasarkan praktikum yang telah dilakukan, dapat disimpulkan bahwa Singly Linked List (SLL) adalah struktur data dinamis di mana setiap elemen, yang disebut Node, terdiri dari data dan pointer successor (next) yang mengarah ke elemen berikutnya. Keseluruhan list dapat diakses melalui pointer first atau head yang menunjuk ke elemen pertama, dan list berakhir ketika pointer elemen terakhir menunjuk ke NULL. Implementasi SLL ini berhasil diterapkan dalam dua konteks: pertama, melalui program Guided untuk mendemonstrasikan operasi dasar seperti insertFirst pada data integer; dan kedua, melalui program Unguided yang mengaplikasikan SLL dalam sistem manajer playlist interaktif yang lebih kompleks, memungkinkan pengelolaan data majemuk (string dan float) serta mendukung berbagai operasi penambahan di awal, akhir, tengah, dan penghapusan data.

E. Referensi

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms* (4th ed.). The MIT Press.

Stroustrup, B. (2014). *Programming: Principles and Practice Using C++* (2nd ed.). Addison-Wesley Professional.