LAPORAN

MODUL 4

SINGLY LINKED LIST (BAGIAN PERTAMA)

Disusun Oleh :

NAMA :

Devi Nurliana

Dosen :

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. DASAR TEORI

Linked list adalah struktur data yang bisa berubah sesuai kebutuhan, yang menyimpan informasi dalam bentuk node atau simpul. Setiap simpul terdiri dari data dan sebuah tanda arah (pointer) yang menghubungkannya ke simpul berikutnya. Berbeda dengan array yang ukurannya tetap, linked list bisa mengubah jumlah datanya saat program sedang berjalan, sehingga lebih mudah dalam mengelola informasi. Singly linked list adalah jenis linked list di mana setiap simpul hanya memiliki satu tanda arah yang menuju simpul berikutnya. Untuk mencari data, kita harus mulai dari simpul pertama hingga simpul terakhir, di mana simpul terakhir memiliki tanda arah kosong (NULL). Pengelolaan singly linked list dilakukan dengan beberapa operasi seperti membuat daftar, mengalokasikan dan melepas memori, menyisipkan data, serta menampilkan data.

B. GUIDED

1. Guide 1
   a. Source Code
      - singlylist.h

```
#ifndef SINGLYLIST_H_INCLUDED
#define SINGLYLIST_H_INCLUDED

#include <iostream>
#define Nil NULL

typedef int infotype;
typedef struct Elmist *address;

struct Elmist {
infotype info;
address next; //pointer
};

struct List {
address first; //digunakan untuk menunjuk ke node pertama
};

//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
```

      - singlylist.cpp

```
#include "singlylist.h" //jangan lupa include headerny
```

```
void CreateList(List &L) { //mendefinisikan atau membuat list pertama yang masih kosong
   L.first = Nil;
}

address alokasi(infotype x) { //digunakan untuk membuat alamat baru di sebuah memori
   address P = new Elmist; //digunakan untuk mengaokasikan memori untuk node baru
   P->info = x; //menyimpan data ke dalam x ke dalam file info nantinya
   P->next = Nil;
   return P; //mengembalikan alamat note baru
}

void dealokasi(address &P) {
   delete P; //menghapus atau mengembalikan ke note system
}

void insertFirst(List &L, address P) { //menyisipkan note baru di list yang sudah dibuat dibagian awal
   P->next = L.first; //note baru p dihubungkan ke note pertama, secara otomatis note pertama diganti ke P
   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 sistem 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;
```

```
        }
}
```

- main.cpp

```
#include<iostream>
#include<cstdlib>
#include "singlylist.h"
#include "singlylist.cpp"


using namespace std;

int main(){
   List L; //berisi pointer utama
   address P; //Cukup satu pointer untuk digunakan berulang
kali, menginisialisasi alamat memori pertama

   CreateList(L); //untuk memanggil prosedur list yang sudah
didefinisikan di bagian body

   cout << "Mengisi List menggunakan insertLast..." << endl;

   //Mengisi list sesuai urutan
   P = alokasi(9); //untuk nilai yang pertama 9 menggunakan
insert last
   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); //memanggil prosedur yang sudah didefinisikan
ke nilai nilai mengggunakan insert last

   system("pause");
   return 0;
}
```

b. Screenshot Output

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

c. Deskripsi

Program ini berfungsi untuk mengimplementasikan struktur data Singly Linked List menggunakan bahasa C++, dengan operasi pembuatan list, alokasi memori, penyisipan data di akhir list, dan penampilan isi list. Langkah kerja program adalah sebagai berikut:

1. Pendefinisian struktur dan ADT (singlylist.h)
   Program mendefinisikan tipe data dasar untuk singly linked list, yaitu infotype bertipe integer dan address sebagai pointer ke node. Setiap node (Elmist) memiliki dua bagian, yaitu info untuk menyimpan data dan next sebagai pointer ke node berikutnya. Selain itu, didefinisikan struktur List yang memiliki pointer first untuk menunjuk node pertama. Pada file ini juga dideklarasikan prosedur dan fungsi primitif seperti CreateList, alokasi, dealokasi, insertFirst, insertLast, dan printInfo.

2. Implementasi fungsi dan prosedur (singlylist.cpp)
   Program mengimplementasikan seluruh fungsi yang telah dideklarasikan pada file header. Prosedur CreateList digunakan untuk membuat list kosong dengan mengatur first bernilai NULL. Fungsi alokasi digunakan untuk mengalokasikan memori baru dan menyimpan data ke dalam node. Prosedur dealokasi digunakan untuk menghapus node dari memori. Prosedur insertFirst menyisipkan node di awal list, sedangkan insertLast menyisipkan node di akhir list dengan cara menelusuri node terakhir terlebih dahulu. Prosedur printInfo digunakan untuk menampilkan seluruh isi list dari node pertama hingga terakhir.

3. Proses utama program (main.cpp)
   Pada fungsi main, program membuat sebuah list kosong dengan memanggil CreateList. Selanjutnya program menampilkan pesan dan mengisi list menggunakan operasi insertLast. Data yang dimasukkan secara berurutan adalah 9, 12, 8, 0, dan 2. Setiap data dialokasikan terlebih dahulu menggunakan fungsi alokasi, kemudian disisipkan ke dalam list. Setelah seluruh data dimasukkan, program memanggil printInfo untuk menampilkan isi list ke layar.

4. Selesai
   Setelah menampilkan isi list, program menunggu input pengguna dengan *system(\"pause\"), kemudian program berakhir dan mengembalikan nilai 0.*

C. UNGUIDED

1. Unguided 1
   a. Source Code
      - 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;
};

typedef struct Node *address;

struct Node {
   Lagu info;
   address next;
};

struct Playlist {
   address first;
};

void createPlaylist(Playlist &P);
address alokasi(Lagu L);
void dealokasi(address &P);

void insertFirst(Playlist &P, address Q);
void insertLast(Playlist &P, address Q);
void insertAfter3(Playlist &P, address Q);

void deleteByTitle(Playlist &P, string judul);
void printPlaylist(Playlist P);

#endif
```

      - playlist.cpp

```
#include "playlist.h"

void createPlaylist(Playlist &P) {
   P.first = NULL;
}

address alokasi(Lagu L) {
   address P = new Node;
   P->info = L;
```

```
    P->next = NULL;
    return P;
}

void dealokasi(address &P) {
    delete P;
}

void insertFirst(Playlist &P, address Q) {
    Q->next = P.first;
    P.first = Q;
}

void insertLast(Playlist &P, address Q) {
    if (P.first == NULL) {
        insertFirst(P, Q);
    } else {
        address last = P.first;
        while (last->next != NULL) {
            last = last->next;
        }
        last->next = Q;
    }
}

void insertAfter3(Playlist &P, address Q) {
    if (P.first == NULL) {
        // jika kosong, langsung jadi first
        insertFirst(P, Q);
        return;
    }

    address temp = P.first;
    int count = 1;

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

    if (temp != NULL) {
        Q->next = temp->next;
        temp->next = Q;
    } else {
        insertLast(P, Q);
    }
}

void deleteByTitle(Playlist &P, string judul) {
    if (P.first == NULL) {
```

```
        cout << "Playlist kosong!\n";
        return;
    }

    address temp = P.first;
    address prev = NULL;
    if (temp->info.judul == judul) {
        P.first = temp->next;
        dealokasi(temp);
        cout << "Lagu \"" << judul << "\" berhasil dihapus!\n";
        return;
    }

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

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

    prev->next = temp->next;
    dealokasi(temp);
    cout << "Lagu \"" << judul << "\" berhasil dihapus!\n";
}

void printPlaylist(Playlist P) {
    if (P.first == NULL) {
        cout << "Playlist kosong!\n";
        return;
    }

    address temp = P.first;
    int nomor = 1;

    while (temp != NULL) {
        cout << nomor++ << ". "
             << temp->info.judul << " - "
             << temp->info.penyanyi
             << " (" << temp->info.durasi << " menit)\n";
        temp = temp->next;
    }
}
```

- main.cpp

```
#include <iostream>
#include "playlist.h"
#include "playlist.cpp"
```

```
using namespace std;

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

    Lagu L;
    address Q;

    cout << "Tambahkan Lagu pada Bagian Awal\n";
    L = {"Sambel Kecombrang", "Ayu Kentongan", 5.30f};
    Q = alokasi(L);
    insertLast(P, Q);

    L = {"Tak Ingin Sate", "Keisya Lemleman", 6.20f};
    Q = alokasi(L);
    insertLast(P, Q);

    L = {"Sisa Uang", "Mahalbanget", 3.40f};
    Q = alokasi(L);
    insertLast(P, Q);

    printPlaylist(P);
    cout << endl;

    cout << "Tambahkan Lagu pada Bagian Awal\n";
    L = {"Bahasa Monyet", "Raiso", 3.50f};
    Q = alokasi(L);
    insertFirst(P, Q);

    printPlaylist(P);
    cout << endl;

    cout << "Tambahkan Lagu Setelah Lagu Ke-3\n";
    L = {"Janji Palsu", "Tiara Masihdini", 3.20f};
    Q = alokasi(L);
    insertAfter3(P, Q);

    printPlaylist(P);
    cout << endl;

    cout << "Menghapus Lagu Berdasarkan Judul\n";
    deleteByTitle(P, "Tak Ingin Sate");

    printPlaylist(P);

    return 0;
}
```

b. Screenshot Output

```
Tambahkan Lagu pada Bagian Awal
1. Sambel Kecombrang - Ayu Kentongan (5.3 menit)
2. Tak Ingin Sate - Keisya Lemleman (6.2 menit)
3. Sisa Uang - Mahalbanget (3.4 menit)

Tambahkan Lagu pada Bagian Awal
1. Bahasa Monyet - Raiso (3.5 menit)
2. Sambel Kecombrang - Ayu Kentongan (5.3 menit)
3. Tak Ingin Sate - Keisya Lemleman (6.2 menit)
4. Sisa Uang - Mahalbanget (3.4 menit)

Tambahkan Lagu Setelah Lagu Ke-3
1. Bahasa Monyet - Raiso (3.5 menit)
2. Sambel Kecombrang - Ayu Kentongan (5.3 menit)
3. Tak Ingin Sate - Keisya Lemleman (6.2 menit)
4. Janji Palsu - Tiara Masihdini (3.2 menit)
5. Sisa Uang - Mahalbanget (3.4 menit)

Menghapus Lagu Berdasarkan Judul
Lagu "Tak Ingin Sate" berhasil dihapus!
1. Bahasa Monyet - Raiso (3.5 menit)
2. Sambel Kecombrang - Ayu Kentongan (5.3 menit)
3. Janji Palsu - Tiara Masihdini (3.2 menit)
4. Sisa Uang - Mahalbanget (3.4 menit)
```

c. Deskripsi

Program ini berfungsi untuk mengelola playlist lagu menggunakan struktur data singly linked list, di mana setiap node menyimpan data lagu berupa judul, penyanyi, dan durasi. Program menyediakan fitur penambahan lagu, penghapusan lagu berdasarkan judul, serta penampilan isi playlist.

1. Pendefinisian struktur dan ADT
   Program mendefinisikan struktur lagu, node, dan playlist, serta mendeklarasikan fungsi untuk membuat playlist, alokasi dan dealokasi memori, penyisipan lagu di awal, di akhir, setelah lagu ke-3, penghapusan lagu berdasarkan judul, dan menampilkan playlist.
2. Proses utama program
   Program membuat playlist kosong, kemudian menambahkan beberapa lagu ke dalam playlist. Lagu dapat ditambahkan di awal, di akhir, maupun setelah lagu ke-3. Program juga menghapus lagu berdasarkan judul tertentu dan menampilkan isi playlist setelah setiap operasi.
3. Selesai
   Setelah seluruh proses dijalankan, program berakhir dan mengembalikan nilai 0.

D. KESIMPULAN

Berdasarkan hasil praktikum yang dilakukan, dapat disimpulkan bahwa struktur data singly linked list memungkinkan pengelolaan data secara dinamis karena jumlah elemen bisa bertambah atau berkurang selama program berjalan. Dengan menggunakan konsep node dan pointer, data dapat disimpan dan dihubungkan secara berurutan tanpa harus berada di lokasi memori yang dekat.

Dari implementasi program singly linked list dan playlist lagu, terlihat bahwa operasi dasar seperti membuat list, mengalokasikan dan melepaskan memori, menyisipkan data, menghapus data, serta menampilkan isi list bisa diterapkan dengan baik menggunakan bahasa C++.
Praktikum ini membantu memahami cara kerja pointer dan linked list serta penerapannya dalam mengelola data yang terstruktur dan fleksibel.

E. REFERNSI

- Modul Praktikum Struktur Data. *Modul 4: Singly Linked List*. Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Telkom, Tahun 2025.
- Weiss, M. A. (2014). *Data Structures and Algorithm Analysis in C++*. Pearson Education.
- Malik, D. S. (2013). *C++ Programming: From Problem Analysis to Program Design*. Cengage Learning.
- Kadir, A. (2012). *Dasar Pemrograman C++*. Andi Publisher.