

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

**MODUL IV
SINGLY LINKED LIST**



Disusun Oleh :

NAMA : Muhamad Naufal Ammar

NIM : 103112430036

Dosen

WAHYU ANDI SAPUTRA

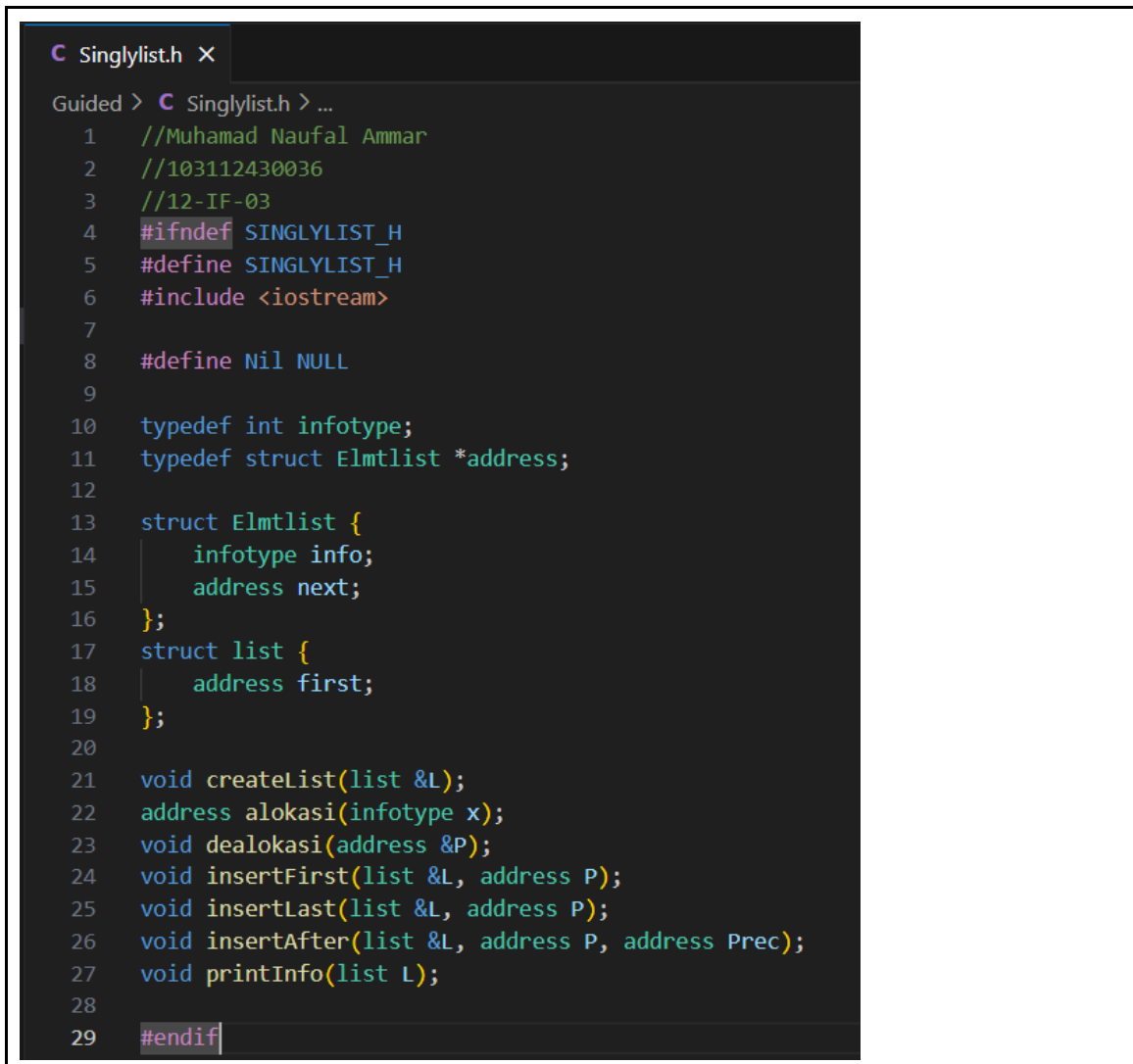
**PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

A. Dasar Teori

Konsep dasar dari program Single Linked List pada pengelolaan playlist lagu ini berlandaskan pada teori struktur data dinamis, di mana alokasi memori dilakukan secara fleksibel menggunakan pointer untuk membentuk hubungan antar-node. Setiap node dalam linked list menyimpan elemen data serta alamat node berikutnya, memungkinkan operasi penambahan dan penghapusan data dilakukan tanpa perlu menggeser elemen lain seperti pada array statis. Menurut penelitian yang dilakukan oleh Putra dan Wibowo (2021), penerapan struktur data linked list dapat meningkatkan efisiensi dalam pengelolaan data karena mendukung operasi insert dan delete dengan kompleksitas waktu yang relatif rendah. Struktur seperti ini juga mendukung pengembangan sistem informasi yang berbasis pada konsep abstraksi data, sebagaimana dijelaskan oleh Nugroho dan Prasetyo (2020), bahwa pemisahan antara definisi dan implementasi fungsi dapat mempermudah proses debugging dan kolaborasi dalam tim pengembang.

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

Guided 1



```
C Singlylist.h X
Guided > C Singlylist.h > ...
1 //Muhamad Naufal Ammar
2 //103112430036
3 //12-IF-03
4 #ifndef SINGLYLIST_H
5 #define SINGLYLIST_H
6 #include <iostream>
7
8 #define Nil NULL
9
10 typedef int infotype;
11 typedef struct Elmtlist *address;
12
13 struct Elmtlist {
14     infotype info;
15     address next;
16 };
17 struct list {
18     address first;
19 };
20
21 void createList(list &L);
22 address alokasi(infotype x);
23 void dealokasi(address &P);
24 void insertFirst(list &L, address P);
25 void insertLast(list &L, address P);
26 void insertAfter(list &L, address P, address Prec);
27 void printInfo(list L);
28
29 #endif
```

C Singlylist.h

G Singlylist.cpp X

Guided > G Singlylist.cpp > InsertLast(list &, address)

```
1 //Muhamad Naufal Ammar
2 //103112430036
3 //12-IF-03
4 #include "Singlylist.h"
5 #include <iostream>
6 void createlist(list &L) {
7     L.first = Nil;
8 }
9 address alokasi(infotype x) {
10     address P = new Elmtlist;
11     P->info = x;
12     P->next = Nil;
13     return P;
14 }
15 void dealokasi(address &P) {
16     delete P;
17 }
18 void insertFirst(list &L, address P) {
19     if (L.first == Nil) {
20         L.first = P;
21     } else {
22         P->next = L.first;
23         L.first = P;
24     }
25 }
26 void insertLast(list &L, address P) {
27     if (L.first == Nil) {
28         L.first = P;
29     } else {
30         address Q = L.first;
31         while (Q->next != Nil) {
32             Q = Q->next;
33         }
34         Q->next = P;
35     }
36 }
37 void insertAfter(list &L, address P, address Prec) {
38     if (Prec != Nil) {
39         P->next = Prec->next;
40         Prec->next = P;
41     }
42 }
43 void printInfo(list L) {
44     address P = L.first;
45     while (P != Nil) {
46         std::cout << P->info << " ";
47         P = P->next;
48     }
49     std::cout << std::endl;
50 }
```

```
Singlylist.h  Singlylist.cpp  main.cpp X
Guided > main.cpp > main()
1 //Muhamad Naufal Ammar
2 //103112430036
3 //12-IF-03
4 #include <iostream>
5 #include "Singlylist.h"
6 #include "Singlylist.cpp"
7
8 using namespace std;
9 int main() {
10     list L;
11     address P1, P2, P3, P4, P5 = Nil;
12     createList(L);
13
14     P1 = alokasi(2);
15     insertFirst(L,P1);
16
17     P2 = alokasi(0);
18     insertFirst(L,P2);
19
20     P3 = alokasi(8);
21     insertFirst(L,P3);
22
23     P4 = alokasi(12);
24     insertFirst(L,P4);
25
26     P5 = alokasi(9);
27     insertFirst(L,P5);
28     printInfo(L);
29
30     return 0;
31 }
```

Screenshots Output

```
PS C:\SMT 3\Struktur Data\modul-amer\103112430036_MuhamadNaufalAmmar_12-IF-03_Modul4> cd "c:\SMT 3\Struktur Data\modul-amer\103112430036_MuhamadNaufalAmmar_12-IF-03_Modul4\Guided\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
9 12 8 0 2
```

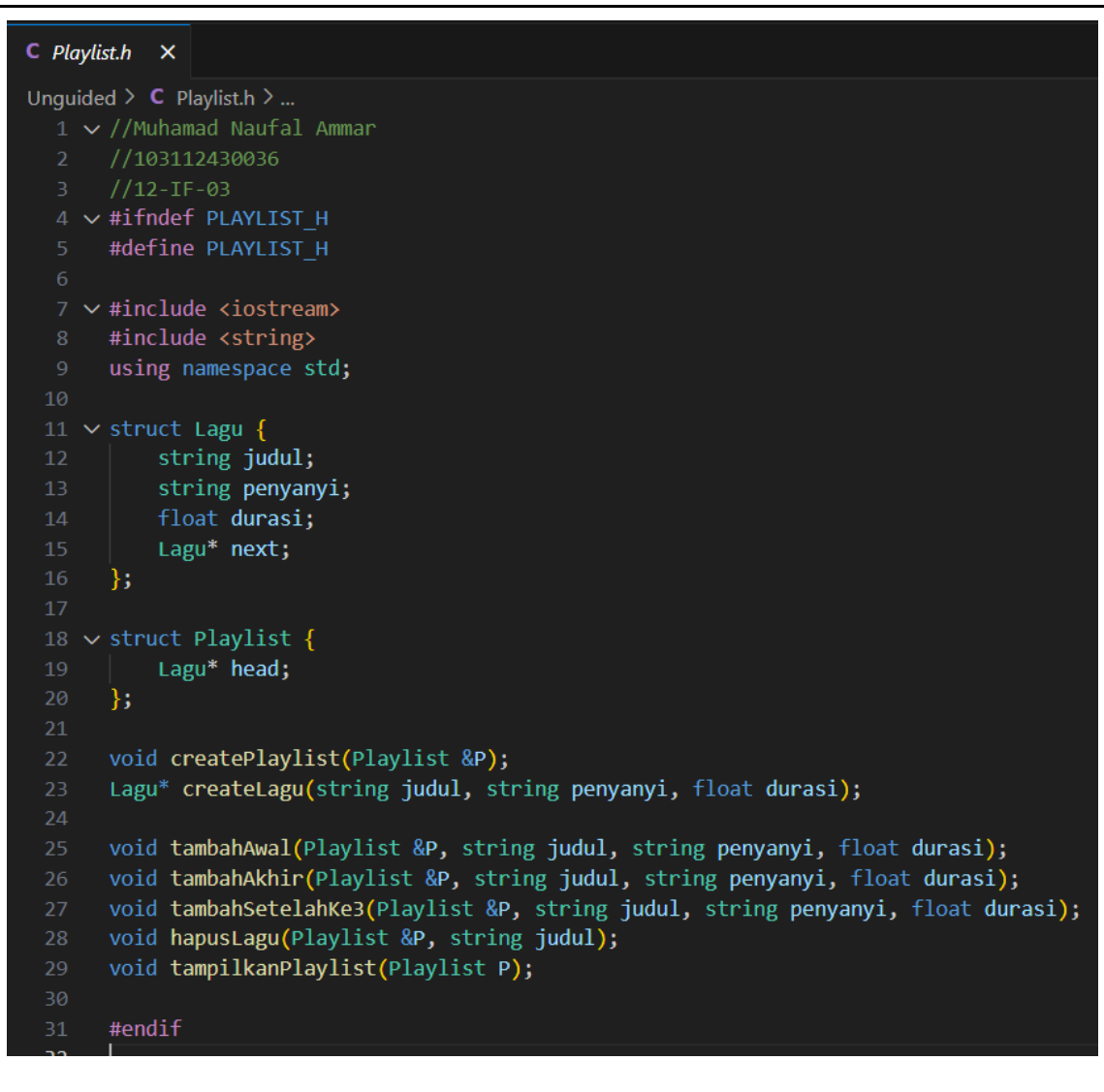
Deskripsi:

Program ini merupakan contoh penerapan dasar dari struktur data Single Linked List dalam bahasa C++, yang bertujuan untuk membantu memahami konsep penyimpanan data secara dinamis dengan memanfaatkan pointer. Program dibagi menjadi tiga komponen utama, yaitu Singlylist.h sebagai file header yang berisi deklarasi struktur node (Elmtlist) dan prototipe fungsi, Singlylist.cpp yang berisi implementasi fungsi-fungsi seperti pembuatan list, penambahan elemen di awal, akhir, atau setelah

node tertentu, serta penampilan isi list, dan main.cpp sebagai program utama yang mengeksekusi seluruh operasi tersebut. Pada bagian utama program, beberapa node dengan nilai 2, 0, 8, 12, dan 9 ditambahkan menggunakan fungsi insertFirst(), sehingga urutan data menjadi 9 12 8 0 2. Melalui implementasi ini, mahasiswa dapat mempelajari konsep fundamental dari linked list, memahami hubungan antar-elemen melalui pointer, serta mengamati bagaimana proses penyisipan dan penelusuran data dapat dilakukan dengan efisien tanpa perlu memindahkan seluruh elemen seperti yang terjadi pada struktur array.

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

Unguided 1



```
C Playlist.h X
Unguided > C Playlist.h > ...
1 //Muhamad Naufal Ammar
2 //103112430036
3 //12-IF-03
4 #ifndef PLAYLIST_H
5 #define PLAYLIST_H
6
7 #include <iostream>
8 #include <string>
9 using namespace std;
10
11 struct Lagu {
12     string judul;
13     string penyanyi;
14     float durasi;
15     Lagu* next;
16 };
17
18 struct Playlist {
19     Lagu* head;
20 };
21
22 void createPlaylist(Playlist &P);
23 Lagu* createLagu(string judul, string penyanyi, float durasi);
24
25 void tambahAwal(Playlist &P, string judul, string penyanyi, float durasi);
26 void tambahAkhir(Playlist &P, string judul, string penyanyi, float durasi);
27 void tambahSetelahKe3(Playlist &P, string judul, string penyanyi, float durasi);
28 void hapusLagu(Playlist &P, string judul);
29 void tampilkanPlaylist(Playlist P);
30
31 #endif
32
```

Playlist.cpp X

Unguided > G: Playlist.cpp > ...

```
1 //Muhamad Naufal Ammar
2 //103112430036
3 //12-IF-03
4 #include "Playlist.h"
5
6 void createPlaylist(Playlist &P) {
7     P.head = NULL;
8 }
9
10 Lagu* createLagu(string judul, string penyanyi, float durasi) {
11     Lagu* laguBaru = new Lagu;
12     laguBaru->judul = judul;
13     laguBaru->penyanyi = penyanyi;
14     laguBaru->durasi = durasi;
15     laguBaru->next = NULL;
16     return laguBaru;
17 }
18
19 void tambahAwal(Playlist &P, string judul, string penyanyi, float durasi) {
20     Lagu* laguBaru = createLagu(judul, penyanyi, durasi);
21     laguBaru->next = P.head;
22     P.head = laguBaru;
23 }
24
25 void tambahAkhir(Playlist &P, string judul, string penyanyi, float durasi) {
26     Lagu* laguBaru = createLagu(judul, penyanyi, durasi);
27     if (P.head == NULL) {
28         P.head = laguBaru;
29     } else {
30         Lagu* temp = P.head;
31         while (temp->next != NULL) {
32             temp = temp->next;
33         }
34         temp->next = laguBaru;
35     }
36 }
37
38 void tambahSetelahKe3(Playlist &P, string judul, string penyanyi, float durasi) {
39     Lagu* laguBaru = createLagu(judul, penyanyi, durasi);
40     Lagu* temp = P.head;
41     int count = 1;
42
43     while (temp != NULL && count < 3) {
44         temp = temp->next;
45         count++;
46     }
47
48     if (temp != NULL) {
49         laguBaru->next = temp->next;
50         temp->next = laguBaru;
51     } else {
52         cout << "Playlist kurang dari 3 lagu, tidak bisa menambah setelah lagu ke-3.\n";
53         delete laguBaru;
54     }
55 }
```

```

void hapusLagu(Playlist &P, string judul) {
    if (P.head == NULL) {
        cout << "Playlist kosong.\n";
        return;
    }

    Lagu* temp = P.head;
    Lagu* prev = NULL;

    if (temp != NULL && temp->judul == judul) {
        P.head = temp->next;
        delete temp;
        cout << "Lagu \"\" << judul << "\" berhasil dihapus.\n";
        return;
    }

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

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

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

void tampilkanPlaylist(Playlist P) {
    if (P.head == NULL) {
        cout << "Playlist kosong.\n";
        return;
    }

    Lagu* temp = P.head;
    int i = 1;

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

```

main.cpp X

Unguided > main.cpp > ...

```
1 //Muhamad Naufal Ammar
2 //103112430036
3 //12-IF-03
4 #include "Playlist.h"
5 #include "Playlist.cpp"
6 using namespace std;
7 int main() {
8     Playlist myPlaylist;
9     createPlaylist(myPlaylist);
10
11     int pilihan;
12     string judul, penyanyi;
13     float durasi;
14
15     do {
16         cout << "\n=== MENU PLAYLIST LAGU ===" << endl;
17         cout << "1. Tambah lagu di awal playlist" << endl;
18         cout << "2. Tambah lagu di akhir playlist" << endl;
19         cout << "3. Tambah lagu setelah playlist ke-3" << endl;
20         cout << "4. Hapus lagu berdasarkan judul" << endl;
21         cout << "5. Tampilkan seluruh lagu dalam playlist" << endl;
22         cout << "0. Keluar" << endl;
23         cout << "Pilih menu: ";
24         cin >> pilihan;
25         cin.ignore();
26
27         switch (pilihan) {
28             case 1:
29                 cout << "\nMasukkan Judul Lagu : ";
30                 getline(cin, judul);
31                 cout << "Masukkan Nama Penyanyi: ";
32                 getline(cin, penyanyi);
33                 cout << "Masukkan Durasi (menit): ";
34                 cin >> durasi;
35                 tambahAwal(myPlaylist, judul, penyanyi, durasi);
36                 cout << "Lagu berhasil ditambahkan di awal playlist.\n";
37                 break;
```



```

38
39     case 2:
40         cout << "\nMasukkan Judul Lagu : ";
41         getline(cin, judul);
42         cout << "Masukkan Nama Penyanyi: ";
43         getline(cin, penyanyi);
44         cout << "Masukkan Durasi (menit): ";
45         cin >> durasi;
46         tambahAkhir(myPlaylist, judul, penyanyi, durasi);
47         cout << "Lagu berhasil ditambahkan di akhir playlist.\n";
48         break;
49
50     case 3:
51         cout << "\nMasukkan Judul Lagu : ";
52         getline(cin, judul);
53         cout << "Masukkan Nama Penyanyi: ";
54         getline(cin, penyanyi);
55         cout << "Masukkan Durasi (menit): ";
56         cin >> durasi;
57         tambahSetelahKe3(myPlaylist, judul, penyanyi, durasi);
58         break;
59
60     case 4:
61         cout << "\nMasukkan Judul Lagu yang akan dihapus: ";
62         getline(cin, judul);
63         hapusLagu(myPlaylist, judul);
64         break;
65
66     case 5:
67         tampilkanPlaylist(myPlaylist);
68         break;
69
70     case 0:
71         cout << "\nTerima kasih! Program selesai.\n";
72         break;
73
74     default:
75         cout << "Pilihan tidak valid! Silakan coba lagi.\n";
76     }
77
78 } while (pilihan != 0);
79
80 return 0;
81 }
82

```

Screenshots Output

```
P5 C:\SMT 3\Struktur Data\modul-amer\103112430036_MuhamadNaufalAmmar_12-IF-03_Modul4\Guided> cd "c:\SMT 3\Struktur Data\modul-amer\103112430036_MuhamadNaufalAmmar_12-IF-03_Modul4\Unguided\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }

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

Masukkan Judul Lagu : Kangen
Masukkan Nama Penyanyi: Dewa
Masukkan Durasi (menit): 4
Lagu berhasil ditambahkan di awal playlist.

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

Masukkan Judul Lagu : Abadi
Masukkan Nama Penyanyi: Perunggu
Masukkan Durasi (menit): 5
Lagu berhasil ditambahkan di awal playlist.

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

Masukkan Judul Lagu : Pram
Masukkan Nama Penyanyi: Perunggu
Masukkan Durasi (menit): 3
Lagu berhasil ditambahkan di akhir playlist.
```

```

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

Masukkan Judul Lagu : Payphone
Masukkan Nama Penyanyi: Maroon5
Masukkan Durasi (menit): 6

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

Masukkan Judul Lagu yang akan dihapus: Abadi
Lagu "Abadi" berhasil dihapus.

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

Daftar Lagu dalam Playlist:
-----
1. Judul   : Kangen
   Penyanyi: Dewa
   Durasi  : 4 menit
-----
2. Judul   : Pram
   Penyanyi: Perunggu
   Durasi  : 3 menit
-----
3. Judul   : Payphone
   Penyanyi: Maroon5
   Durasi  : 6 menit
-----

```

Deskripsi:

Kode program yang terdiri dari tiga file, yaitu `Playlist.h`, `Playlist.cpp`, dan `main.cpp`, merupakan implementasi modular dari struktur data *Single Linked List* untuk mengelola playlist lagu secara dinamis. Pada file `Playlist.h`, didefinisikan struktur node bernama *Lagu* yang berisi atribut judul, penyanyi, dan durasi, serta deklarasi fungsi-fungsi utama yang digunakan untuk memanipulasi list. File `Playlist.cpp` berfungsi sebagai tempat implementasi seluruh fungsi tersebut, seperti pembuatan playlist baru, penambahan lagu di awal, akhir, maupun setelah posisi ke-3, penghapusan lagu berdasarkan judul, serta penampilan seluruh isi playlist. Sementara itu, file `main.cpp` bertindak sebagai program utama yang menyediakan menu interaktif bagi pengguna untuk menambahkan, menghapus, dan menampilkan lagu melalui input langsung. Pembagian kode menjadi tiga bagian ini mencerminkan prinsip *modular programming*, yang bertujuan untuk meningkatkan keteraturan, kemudahan pemeliharaan, serta memperjelas alur kerja program berbasis *linked list*.

C. Kesimpulan

Berdasarkan implementasi dan teori yang mendasari, program pengelolaan playlist menggunakan Single Linked List ini menunjukkan bagaimana struktur data dinamis dapat digunakan untuk mengatur data musik secara efisien dan fleksibel. Melalui penerapan prinsip abstraksi data dan pemrograman modular, program ini tidak hanya berfungsi dengan baik dalam melakukan operasi dasar pada list seperti menambah, menghapus, dan menampilkan data, tetapi juga mencerminkan penerapan praktik rekayasa perangkat lunak yang baik dalam konteks akademik maupun pengembangan aplikasi nyata. Dengan demikian, pemanfaatan linked list menjadi solusi ideal untuk sistem yang membutuhkan manipulasi data secara dinamis dan efisien dalam pemrograman C++.

D. Referensi

Nugroho, A., & Prasetyo, D. (2020). *Implementasi Struktur Data Linked List dalam Pengembangan Sistem Informasi Berbasis Objek*. Jurnal Teknologi dan Sistem Informasi, 8(2), 112–119.

Putra, R. A., & Wibowo, S. (2021). *Analisis Efisiensi Struktur Data Linked List dalam Pengelolaan Data Dinamis*. Jurnal Ilmiah Informatika dan Komputer, 15(3), 87–94.