

**LAPORAN PRAKTIKUM  
STRUKTUR DATA**

**MODUL 4  
SINGLY LINKED LIST (1)**



**Disusun Oleh :**

NAMA : Faris Walid Awwal Aidi

NIM : 103112430133

**Dosen**

FAHRUDIN MUKTI WIBOWO

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

## A. Dasar Teori

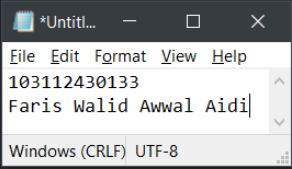
Linked list (senarai berkait) adalah sebuah struktur data yang terdiri dari serangkaian elemen data yang saling berhubungan atau berkait. Struktur ini bersifat fleksibel, memungkinkannya untuk tumbuh dan mengerut secara dinamis sesuai kebutuhan, tidak seperti array yang bersifat statis. Implementasi linked list menggunakan pointer lebih disukai karena sifat dinamisnya sangat cocok dengan bentuk data linked list yang saling bergandengan. Operasi dasar yang umum dilakukan pada linked list meliputi penciptaan list, penyisipan elemen, penghapusan elemen, dan penelusuran (view) elemen.

Model yang dipelajari adalah Singly Linked List, yang merupakan ADT (Abstract Data Type) linked list dengan pointer yang bergerak satu arah saja. Setiap elemen dalam list ini disebut Node (simpul), yang terdiri dari dua bagian: Data sebagai penyimpan informasi utama, dan Successor (pointer 'next') yang berfungsi sebagai penghubung yang menunjuk ke alamat elemen berikutnya. Sebuah pointer khusus bernama 'first' (atau 'head') digunakan untuk menunjuk alamat elemen pertama dalam list. Rangkaian list dianggap berakhir ketika elemen terakhir memiliki pointer 'next' yang menunjuk ke NULL (Nil), yang artinya tidak ada lagi elemen setelahnya.

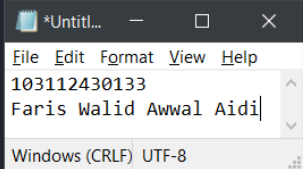
B. Guided (berisi screenshot source code & output program disertai penjelasannya)

1. Guided

```
guided1 > C singlyList.h > alokasi(infotype)
1  #ifndef SINGLYLIST_H_INCLUDED
2  #define SINGLYLIST_H_INCLUDED
3  #include <iostream>
4  #define NIL NULL
5
6  typedef int infotype;
7  typedef struct Elmlist *address;
8
9  struct Elmlist {
10     infotype info;
11     address next;
12 };
13
14 struct list {
15     address first;
16 };
17
18 // Deklarasi Prosedur dan Fungsi Primitif
19 void createList(list &L);
20 address alokasi(infotype x);
21 void dealokasi(address &P);
22 void insertFirst(list &L, address P);
23 void insertLast(list &L, address P);
24 void printInfo(list L);
25
26 #endif
```



```
guided1 > C singlyList.cpp > alokasi(infotype)
1  #include "singlyList.h"
2
3  void createList(list &L) {
4     L.first = NIL;
5  }
6
7  address alokasi(infotype x) {
8     address P = new Elmlist;
9     P->info = x;
10    P->next = NIL;
11    return P;
12 }
13
14 void dealokasi(address &P) {
15     delete P;
16 }
17
18 void insertFirst(list &L, address P) {
19     P->next = L.first;
20     L.first = P;
21 }
22
23 void insertLast(list &L, address P) {
24     if (L.first == NIL) {
25         // Jika list kosong, insertLast sama dengan insertFirst
26         insertFirst(L, P);
27     } else {
28         // Jika list tidak kosong, cari element terakhir
29         address Last = L.first;
30         while (Last->next != NIL) {
31             Last = Last->next;
32         }
33         // Sambungkan element terakhir dengan element terbaru (p)
34         Last->next = P;
35     }
36 }
```



```

38 void printInfo(List L) {
39     address P = L.first;
40     if (P == NIL) {
41         std::cout << "List kosong" << std::endl;
42     } else {
43         while (P != NIL) {
44             std::cout << P->info << " ";
45             P = P->next;
46         }
47         std::cout << std::endl;
48     }
49 }

```

```

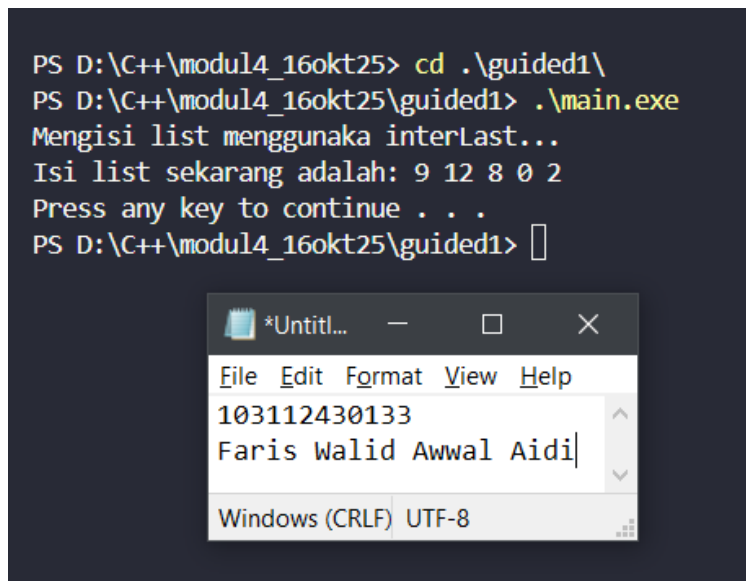
guided1 > main.cpp > main()
1  #include <iostream>
2  #include <cstdlib>
3  #include "singlyList.h"
4  #include "singlyList.cpp"
5
6  using namespace std;
7
8  int main()
9  {
10     List L;
11     address P; // Cukup satu pointer untuk digunakan berulang kali
12
13     createList(L);
14
15     cout << "Mengisi list menggunakan insertLast..." << endl;
16
17     // Mengisi list sesuai urutan
18     P = alokasi(9);
19     insertLast(L, P);
20
21     P = alokasi(12);
22     insertLast(L, P);
23
24     P = alokasi(8);
25     insertLast(L, P);
26
27     P = alokasi(0);
28     insertLast(L, P);
29
30     P = alokasi(2);
31     insertLast(L, P);
32
33     cout << "Isi list sekarang adalah: " ;
34     printInfo(L);
35
36     system("pause");
37     return 0;
38 }

```

\*Untitl... — □ ×

File	Edit	Format	View	Help
103112430133				
Faris Walid Awwal Aidi				
Windows (CRLF) UTF-8				

## Screenshots Output



```
PS D:\C++\modul4_16okt25> cd .\guided1\  
PS D:\C++\modul4_16okt25\guided1> .\main.exe  
Mengisi list menggunakan interLast...  
Isi list sekarang adalah: 9 12 8 0 2  
Press any key to continue . . .  
PS D:\C++\modul4_16okt25\guided1> 
```

The screenshot also shows a text editor window titled '\*Untitl...' with the following content:

```
File Edit Format View Help  
103112430133  
Faris Walid Awwal Aidi|  
Windows (CRLF) UTF-8
```

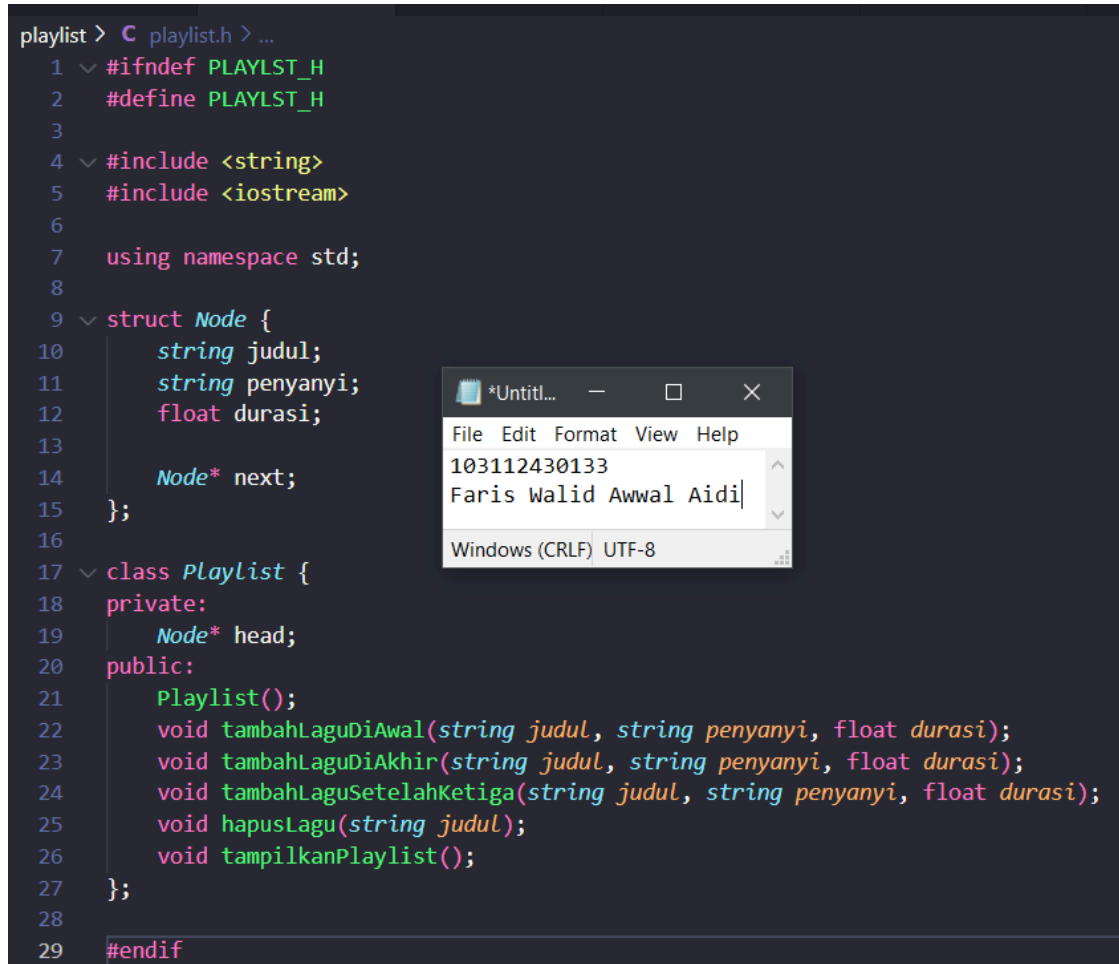
## Deskripsi:

Program ini merupakan implementasi dasar dari Abstract Data Type (ADT) Singly Linked List. Program ini terdiri dari file header (Singlylist.h) yang mendefinisikan struktur data untuk elemen list (node) dan list itu sendiri, serta prototipe fungsi-fungsi dasarnya. File implementasi (Singlylist.cpp) menyediakan logika untuk fungsi-fungsi tersebut, seperti alokasi untuk membuat node baru, insertFirst untuk menambah data di awal list, dan printInfo untuk menampilkan data. Program utama (main.cpp) bertugas mendemonstrasikan fungsionalitas dasar ini dengan cara menginisialisasi list, mengalokasikan lima node dengan data integer, dan memasukkan kelimaanya secara berurutan menggunakan insertFirst. Akhirnya, program mencetak isi list ke layar, yang menampilkan urutan data secara terbalik dari urutan pemasukannya (LIFO) karena penggunaan insertFirst.

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

1. Unguided I

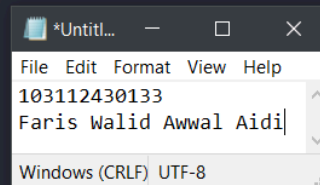
- playlist.h



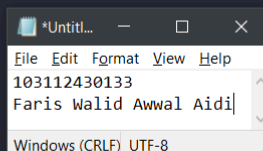
```
playlist > C playlist.h > ...
1  √ #ifndef PLAYLST_H
2  #define PLAYLST_H
3
4  √ #include <string>
5  #include <iostream>
6
7  using namespace std;
8
9  √ struct Node {
10     string judul;
11     string penyanyi;
12     float durasi;
13
14     Node* next;
15 };
16
17 √ class Playlist {
18 private:
19     Node* head;
20 public:
21     Playlist();
22     void tambahLaguDiAwal(string judul, string penyanyi, float durasi);
23     void tambahLaguDiAkhir(string judul, string penyanyi, float durasi);
24     void tambahLaguSetelahKetiga(string judul, string penyanyi, float durasi);
25     void hapusLagu(string judul);
26     void tampilkanPlaylist();
27 };
28
29 #endif
```

- playlist.cpp

```
playlist > G playlist.cpp > tampilkanPlaylist()
1  #include "Playlist.h"
2
3  Playlist::Playlist() {
4      head = nullptr;
5  }
6
7  void Playlist::tambahLaguDiAwal(string judul, string penyanyi, float durasi) {
8      Node* newNode = new Node;
9
10     newNode->judul = judul;
11     newNode->penyanyi = penyanyi;
12     newNode->durasi = durasi;
13
14     newNode->next = head;
15     head = newNode;
16
17     cout << "Berhasil menambah '" << judul << "' di awal playlist.\n";
18 }
19
20 void Playlist::tambahLaguDiAkhir(string judul, string penyanyi, float durasi) {
21     Node* newNode = new Node;
22     newNode->judul = judul;
23     newNode->penyanyi = penyanyi;
24     newNode->durasi = durasi;
25     newNode->next = nullptr;
26
27     if (head == nullptr) {
28         head = newNode;
29     } else {
30         Node* temp = head;
31         while (temp->next != nullptr) {
32             temp = temp->next;
33         }
34         temp->next = newNode;
35     }
36     cout << "Berhasil menambah '" << judul << "' di akhir playlist.\n";
37 }
38
```



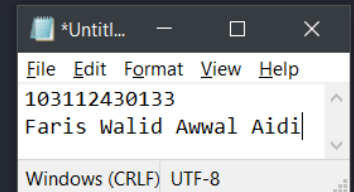
```
39 void Playlist::tambahLaguSetelahKetiga(string judul, string penyanyi, float durasi) {
40     Node* temp = head;
41     int counter = 1;
42
43     while (counter < 3 && temp != nullptr) {
44         temp = temp->next;
45         counter++;
46     }
47
48     if (temp == nullptr) {
49         cout << "Gagal! Playlist hanya memiliki " << counter-1 << " lagu, tidak bisa menambah setelah lagu ke-3.\n";
50     } else {
51         Node* newNode = new Node;
52         newNode->judul = judul;
53         newNode->penyanyi = penyanyi;
54         newNode->durasi = durasi;
55
56         newNode->next = temp->next;
57         temp->next = newNode;
58
59         cout << "Berhasil menambah '" << judul << "' setelah lagu ke-3.\n";
60     }
61 }
```



```

62
63 void Playlist::hapusLagu(string judul) {
64     if (head == nullptr) {
65         cout << "Playlist kosong, tidak ada yang bisa dihapus.\n";
66         return;
67     }
68
69     if (head->judul == judul) {
70         Node* temp = head;
71         head = head->next;
72         delete temp;
73         cout << "Berhasil menghapus '" << judul << "' dari playlist.\n";
74         return;
75     }
76
77     Node* previous = head;
78     Node* current = head->next;
79
80     while (current != nullptr && current->judul != judul) {
81         previous = current;
82         current = current->next;
83     }
84
85     if (current == nullptr) {
86         cout << "Lagu '" << judul << "' tidak ditemukan di playlist.\n";
87     } else {
88         previous->next = current->next;
89
90         delete current;
91         cout << "Berhasil menghapus '" << judul << "' dari playlist.\n";
92     }
93 }
94

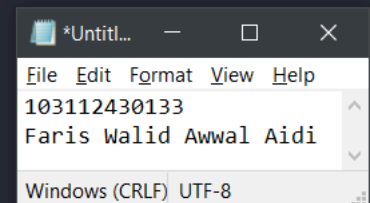
```



```

95 void Playlist::tampilkanPlaylist() {
96     Node* temp = head;
97
98     if (temp == nullptr) {
99         cout << "Playlist saat ini kosong." << endl;
100         return;
101     }
102
103     cout << "\n===== ISI PLAYLIST ANDA =====\n";
104     int nomor = 1;
105     while (temp != nullptr) {
106         cout << nomor << ". Judul      : " << temp->judul << "\n";
107         cout << "    Penyanyi : " << temp->penyanyi << "\n";
108         cout << "    Durasi   : " << temp->durasi << " menit\n";
109         cout << "-----\n";
110
111         temp = temp->next;
112         nomor++;
113     }
114     cout << "===== \n";
115 }

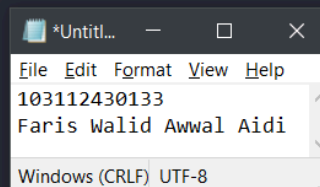
```



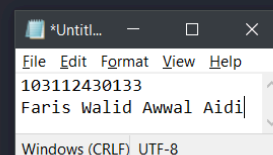


- main.cpp

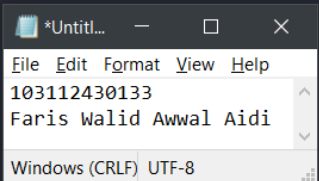
```
playlist > main.cpp > ...
1  #include "Playlist.h"
2  #include <iostream>
3  #include <string>
4
5  using namespace std;
6
7  void tampilkanMenu() {
8      cout << "\n=== MENU MANAJEMEN PLAYLIST ===\n";
9      cout << "1. Tambah Lagu di Awal\n";
10     cout << "2. Tambah Lagu di Akhir\n";
11     cout << "3. Tambah Lagu Setelah Lagu ke-3\n";
12     cout << "4. Hapus Lagu (berdasarkan judul)\n";
13     cout << "5. Tampilkan Seluruh Playlist\n";
14     cout << "6. Keluar\n";
15     cout << "=====\n";
16     cout << "Masukkan pilihan Anda (1-6): ";
17 }
18
19 int main() {
20     Playlist playlistSaya;
21     int pilihan = 0;
22     string judul, penyanyi;
23     float durasi;
24
25     do {
26         tampilkanMenu();
27
28         while (!(cin >> pilihan)) {
29             cout << "Input tidak valid. Harap masukkan angka (1-6): ";
30             cin.clear();
31             cin.ignore(1024, '\n');
32         }
33 }
```



```
36     switch (pilihan) {
37         case 1:
38             cout << "Masukkan Judul: ";
39             getline(cin, judul);
40             cout << "Masukkan Penyanyi: ";
41             getline(cin, penyanyi);
42             cout << "Masukkan Durasi (menit, cth: 3.5): ";
43             cin >> durasi;
44             playlistSaya.tambahLaguDiAwal(judul, penyanyi, durasi);
45             break;
46
47         case 2:
48             cout << "Masukkan Judul: ";
49             getline(cin, judul);
50             cout << "Masukkan Penyanyi: ";
51             getline(cin, penyanyi);
52             cout << "Masukkan Durasi (menit, cth: 3.5): ";
53             cin >> durasi;
54             playlistSaya.tambahLaguDiAkhir(judul, penyanyi, durasi);
55             break;
56
57         case 3:
58             cout << "Masukkan Judul: ";
59             getline(cin, judul);
60             cout << "Masukkan Penyanyi: ";
61             getline(cin, penyanyi);
62             cout << "Masukkan Durasi (menit, cth: 3.5): ";
63             cin >> durasi;
64             playlistSaya.tambahLaguSetelahKetiga(judul, penyanyi, durasi);
65             break;
66
67         case 4:
68             cout << "Masukkan Judul Lagu yang ingin dihapus: ";
69             getline(cin, judul);
70             playlistSaya.hapusLagu(judul);
71             break;
72 }
```



```
72
73     case 5:
74         playlistSaya.tampilkanPlaylist();
75         break;
76
77     case 6:
78         cout << "Terima kasih telah menggunakan program ini. Sampai jumpa!\n";
79         break;
80
81     default:
82         cout << "Pilihan tidak valid! Silakan pilih angka dari 1 sampai 6.\n";
83         break;
84 }
85
86 if (pilihan != 6) {
87     cout << "\nTekan Enter untuk kembali ke menu...";
88     if (pilihan == 1 || pilihan == 2 || pilihan == 3) {
89         cin.ignore(1024, '\n');
90     }
91     cin.get();
92 }
93
94 } while (pilihan != 6);
95
96 return 0;
97 }
```



## 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: Kacamata
Masukkan Penyanyi: Afghan
Masukkan Durasi (menit, cth: 3.5): 3
Berhasil menambah 'Kacamata' 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: Bukan Cinta Biasa
Masukkan Penyanyi: Afghan
Masukkan Durasi (menit, cth: 3.5): 3.5
Berhasil menambah 'Bukan Cinta Biasa' 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: I Would Like
Masukkan Penyanyi: JKAY
Masukkan Durasi (menit, cth: 3.5): 3
Berhasil menambah 'I Would Like' 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: Wajahmu Mengalihkan Duniaku
Berhasil menghapus 'Wajahmu Mengalihkan Duniaku' dari 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      : Jodoh Pasti Bertemu
   Penyanyi   : Afghan
   Durasi     : 2.5 menit
-----
2. Judul      : Kacamata
   Penyanyi   : Afghan
   Durasi     : 3 menit
-----
3. Judul      : I Would Like
   Penyanyi   : JKAY
   Durasi     : 3 menit
-----
4. Judul      : Bukan Cinta Biasa
   Penyanyi   : Afghan
   Durasi     : 3.5 menit
-----
=====

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

### Deskripsi:

Program ini adalah aplikasi Singly Linked List yang lebih kompleks, dirancang sebagai sistem manajemen playlist lagu. Program ini mengimplementasikan struktur data majemuk pada setiap node, yang mampu menyimpan tiga atribut sekaligus: judul lagu (string), nama penyanyi (string), dan durasi (float). Logika program dibagi menjadi tiga file: `Playlist.h` sebagai file header, `Playlist.cpp` untuk implementasi kelas `Playlist`, dan `main.cpp` sebagai program utama. Kelas `Playlist` menyediakan beragam operasi, termasuk menambah lagu di awal, menambah di akhir, menambah setelah posisi ketiga, menghapus lagu berdasarkan judul, dan menampilkan seluruh playlist. Program utama (`main.cpp`) dirancang agar sepenuhnya interaktif, menampilkan menu berbasis teks dalam satu lingkaran `do-while` yang memungkinkan pengguna untuk memilih dan menjalankan fungsi-fungsi manajemen playlist tersebut secara berulang hingga pengguna memilih opsi untuk keluar.

#### D. Kesimpulan

Berdasarkan praktikum yang telah dilakukan, dapat disimpulkan bahwa Singly Linked List (SLL) adalah sebuah struktur data dinamis di mana setiap elemen (disebut Node) terdiri dari data dan sebuah pointer successor ('next') yang menunjuk ke elemen berikutnya. Seluruh rangkaian list ini diakses melalui pointer 'first' (atau 'head') yang menunjuk ke elemen pertama, dan diakhiri oleh elemen yang pointer-nya menunjuk ke NULL. Implementasi SLL ini telah berhasil diterapkan melalui dua program: program Guided yang mendemonstrasikan operasi dasar seperti *insertFirst* pada data integer, dan program Unguided yang mengaplikasikannya dalam program manajer playlist interaktif yang lebih kompleks, yang mampu mengelola data majemuk (string dan float) serta melakukan berbagai operasi seperti penambahan di awal, akhir, tengah, dan penghapusan data.

#### Referensi

- Raharjo, Budi. 2025. *Buku Pemrograman C++ Mudah dan Cepat Menjadi Master C*.  
*Wikipedia contributors*. (2024, 8 Mei). C++. *Wikipedia, Ensiklopedia Bebas*.  
Diakses pada 2 Oktober 2025, dari <https://id.wikipedia.org/wiki/C%2B%2B>