

**LAPORAN PRAKTIKUM  
STRUKTUR DATA**

**MODUL IV & V  
SINGLY LINKED LIST**



**Disusun Oleh :**

NAMA : Azzahra Fareluka Esti Ning Tyas  
NIM : 103112430023

**Dosen**

FAHRUDIN MUKTI WIBOWO

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

## A. Dasar Teori

Linked list merupakan struktur data dinamis yang terdiri dari sejumlah elemen yang disebut node, di mana setiap node berisi data dan pointer yang menunjuk ke node berikutnya. Tidak seperti array yang memiliki ukuran tetap, linked list dapat bertambah atau berkurang sesuai kebutuhan program (Goodrich, Tamassia, & Goldwasser, 2014).

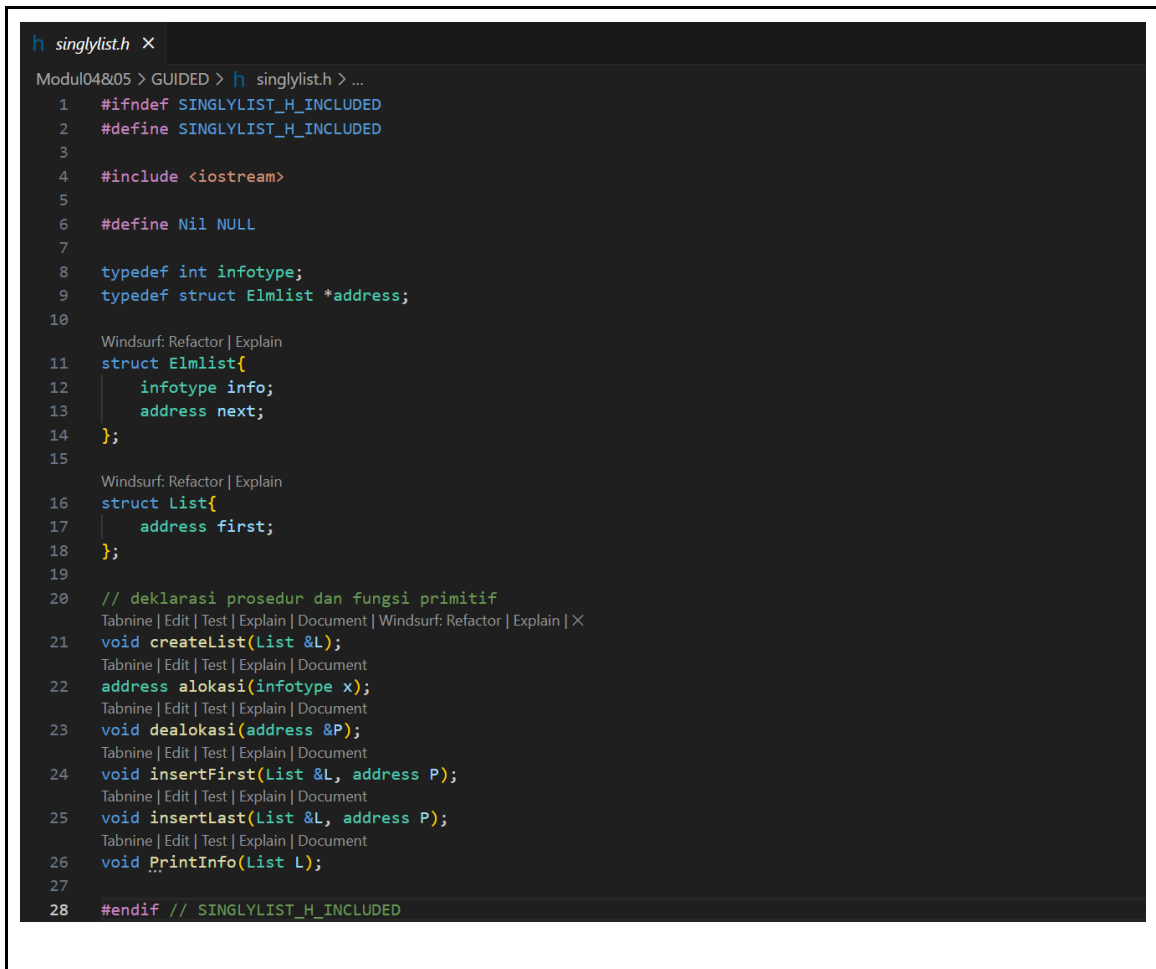
Singly Linked List adalah bentuk paling sederhana dari linked list yang hanya memiliki satu arah pointer. Setiap node terdiri dari dua bagian, yaitu data field dan next pointer, sedangkan node terakhir menunjuk ke NULL untuk menandakan akhir list (Lafore, 2002). Karena hanya memiliki satu arah, pengaksesan data dilakukan secara sekuensial dari awal hingga akhir list.

Operasi dasar pada singly linked list meliputi pembuatan list (create list), penambahan data (insert first, insert last, insert after), penghapusan data (delete first, delete last, delete after), penelusuran (traversal), dan pencarian data (searching). Operasi-operasi tersebut memungkinkan manipulasi data secara efisien tanpa harus menggeser elemen lain seperti pada array (Sedgewick & Wayne, 2011).

Kelebihan singly linked list adalah fleksibilitas ukuran dan efisiensi penyisipan atau penghapusan data. Namun, kelemahannya adalah akses data yang hanya satu arah dan tidak mendukung random access (Malik, 2018). Struktur ini menjadi dasar penting bagi pengembangan struktur data lain seperti doubly linked list, stack, queue, dan tree (Wirth, 1976).

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

Guided 1



```
h singlylist.h x
Modul04&05 > GUIDED > h singlylist.h > ...
1  #ifndef SINGLYLIST_H_INCLUDED
2  #define SINGLYLIST_H_INCLUDED
3
4  #include <iostream>
5
6  #define Nil NULL
7
8  typedef int infotype;
9  typedef struct Elmlist *address;
10
11  struct Elmlist{
12      infotype info;
13      address next;
14  };
15
16  struct List{
17      address first;
18  };
19
20  // deklarasi prosedur dan fungsi primitif
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 PrintInfo(List L);
27
28  #endif // SINGLYLIST_H_INCLUDED
```

```

C++ singlylist.cpp x
Modul04&05 > GUIDED > C++ singlylist.cpp > printInfo(List)
1  #include "Singlylist.h"
2
3  Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
4  void createList(List &L){
5      L.first = Nil;
6  }
7
8  Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
9  address alokasi(infotype x){
10     address P = new Elmlist;
11     P->info = x;
12     P->next = Nil;
13     return P;
14 }
15
16 Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
17 void dealokasi(address &P){
18     delete P;
19 }
20
21 Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
22 void insertFirst(List &L, address P){
23     P->next = L.first;
24     L.first = P;
25 }
26
27 Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
28 void insertLast(List &L, address P){
29     if (L.first == Nil) {
30         // jika list kosong, insertlast sama dengan insertfirst
31         insertFirst(L, P);
32     } else {
33         // jika list tidak kosong, cari elemen terakhir
34         address Last = L.first;
35         while (Last->next != Nil) {
36             Last = Last->next;
37         }
38         // sambungkan elemen terakhir ke elemen baru (P)
39         Last->next = P;
40     }
41 }
42
43 Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
44 void printInfo(List L){
45     address P = L.first;
46     if (P == Nil) {
47         std::cout << "List kosong" << std::endl;
48     } else {
49         while (P != Nil) {
50             std::cout << P->info << " ";
51             P = P->next;
52         }
53         std::cout << std::endl;
54     }
55 }
56
57

```

```
C++ main.cpp x
Modul04&05 > GUIDED > C++ 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      List L;
10     address P; //cukup satu pointer untuk digunakan berulang kali
11
12     createlist(L);
13
14     cout << "Mengisi list menggunakan insertlast..." << endl;
15
16     //mengisi list sesuai urutan
17     P = alokasi(9);
18     insertLast(L, P);
19
20     P = alokasi(12);
21     insertLast(L, P);
22
23     P = alokasi(8);
24     insertLast(L, P);
25
26     P = alokasi(0);
27     insertLast(L, P);
28
29     P = alokasi(2);
30     insertLast(L, P);
31
32     cout << "Isi list sekarang adalah: ";
33     printInfo(L);
34
35     system("pause");
36     return 0;
37 }
```

## Screenshots Output

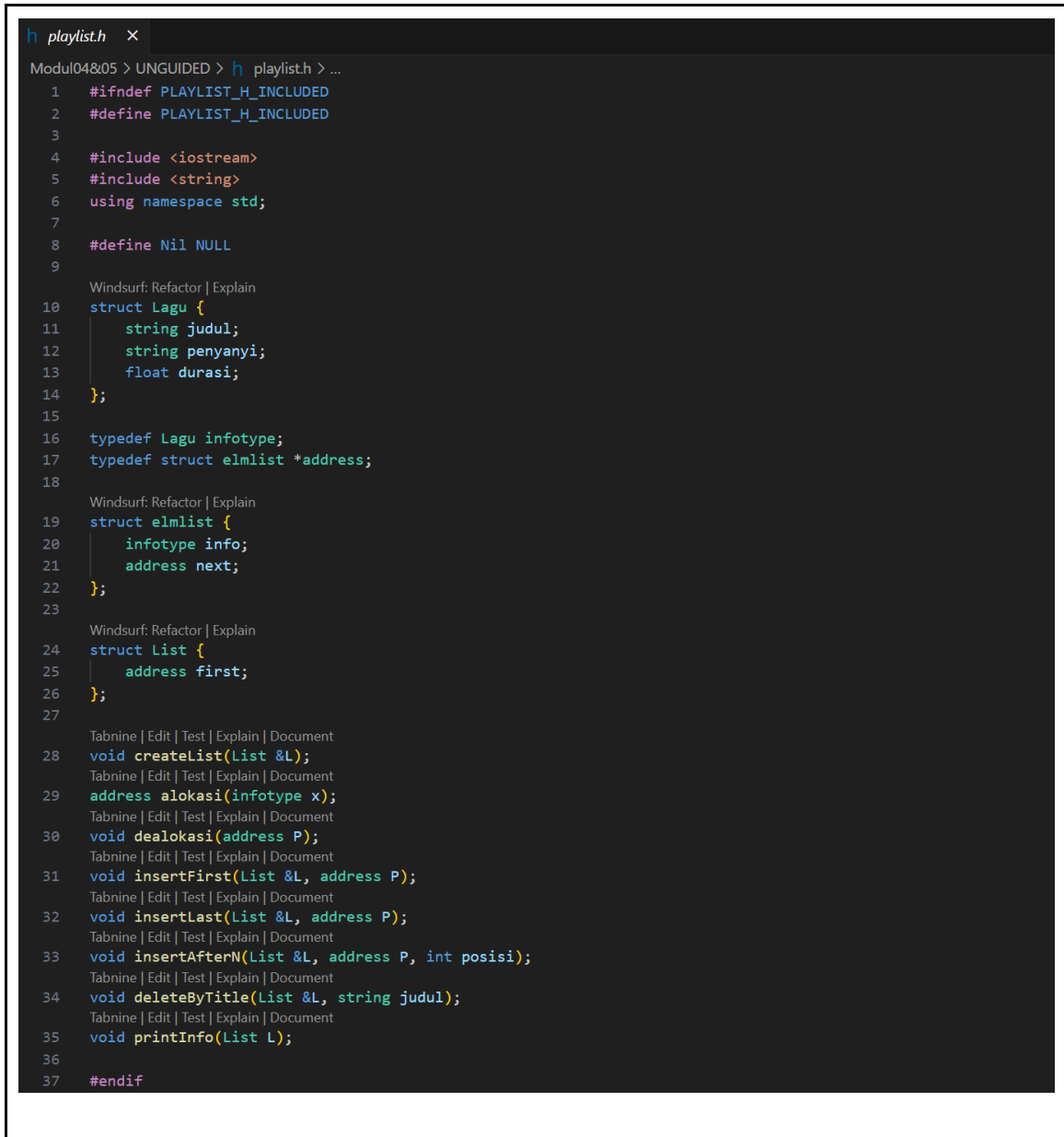
```
PS D:\StrukturData> cd "d:\StrukturData\Modul04&05\GUIDED\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
● Mengisi list menggunakan insertlast...
Isi list sekarang adalah: 9 12 8 0 2
Press any key to continue . . .
```

## Deskripsi:

Program ini merupakan implementasi Singly Linked List dalam C++ yang menyimpan data bertipe integer secara dinamis. File Singlylist.h berisi deklarasi struktur dan fungsi dasar, Singlylist.cpp mengimplementasikan operasi seperti membuat list, menambah elemen di awal atau akhir, serta menampilkan isi list, sedangkan main.cpp digunakan untuk menguji fungsi-fungsi tersebut. Program menambahkan beberapa elemen menggunakan insertLast dan menampilkan hasilnya dengan printInfo, sehingga menggambarkan cara kerja dasar linked list dalam penyimpanan dan penelusuran data.

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

Unguided 1



```
h playlist.h x
Modul04&05 > UNGUIDED > h playlist.h > ...
1  #ifndef PLAYLIST_H_INCLUDED
2  #define PLAYLIST_H_INCLUDED
3
4  #include <iostream>
5  #include <string>
6  using namespace std;
7
8  #define Nil NULL
9
10 struct Lagu {
11     string judul;
12     string penyanyi;
13     float durasi;
14 };
15
16 typedef Lagu infotype;
17 typedef struct elmlist *address;
18
19 struct elmlist {
20     infotype info;
21     address next;
22 };
23
24 struct List {
25     address first;
26 };
27
28 void createList(List &L);
29 address alokasi(infotype x);
30 void dealokasi(address P);
31 void insertFirst(List &L, address P);
32 void insertLast(List &L, address P);
33 void insertAfterN(List &L, address P, int posisi);
34 void deleteByTitle(List &L, string judul);
35 void printInfo(List L);
36
37 #endif
```

```

C++ playlist.cpp X
Modul04&05 > UNGUIDED > C++ playlist.cpp > insertFirst(List &, address)

1  #include "playlist.h"
2
3  Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
4  void createList(List &L) {
5      L.first = Nil;
6  }
7
8  Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
9  address alokasi(infotype x) {
10     address P = new elmlist;
11     P->info = x;
12     P->next = Nil;
13     return P;
14 }
15
16 Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
17 void dealokasi(address P) {
18     delete P;
19 }
20
21 Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
22 void insertFirst(List &L, address P) {
23     P->next = L.first;
24     L.first = P;
25 }
26
27 Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
28 void insertLast(List &L, address P) {
29     if (L.first == Nil) {
30         L.first = P;
31     } else {
32         address Q = L.first;
33         while (Q->next != Nil) {
34             Q = Q->next;
35         }
36         Q->next = P;
37     }
38 }
39
40 Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
41 void insertAfterN(List &L, address P, int posisi) {
42     address Q = L.first;
43     int count = 1;
44     while (Q != Nil && count < posisi) {
45         Q = Q->next;
46         count++;
47     }
48     if (Q != Nil) {
49         P->next = Q->next;
50         Q->next = P;
51     } else {
52         cout << "Posisi tidak valid. Lagu ditambahkan di akhir playlist.\n";
53         insertLast(L, P);
54     }
55 }
56
57 Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
58 void deleteByTitle(List &L, string judul) {
59     if (L.first == Nil) {
60         cout << "Playlist kosong.\n";
61         return;
62     }
63
64     address P = L.first;
65     address Prec = Nil;
66
67     while (P != Nil && P->info.judul != judul) {
68         Prec = P;
69         P = P->next;
70     }
71
72     if (P == Nil) {
73         cout << "Lagu dengan judul \"" << judul << "\" tidak ditemukan.\n";
74     } else {
75         if (Prec == Nil) {
76             L.first = P->next;
77         } else {
78             Prec->next = P->next;
79         }
80         dealokasi(P);
81         cout << "Lagu \"" << judul << "\" berhasil dihapus.\n";
82     }
83 }
84
85 Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
86 void printInfo(List L) {
87     if (L.first == Nil) {
88         cout << "Playlist kosong.\n";
89     } else {
90         address P = L.first;
91         int i = 1;
92         cout << "=== Daftar Lagu dalam Playlist ===\n";
93         while (P != Nil) {
94             cout << i++ << ". Judul : " << P->info.judul << endl;
95             cout << "    Penyanyi : " << P->info.penyanyi << endl;
96             cout << "    Durasi : " << P->info.durasi << " menit\n";
97             cout << "===== \n";
98             P = P->next;
99         }
100     }
101 }

```

C++ main.cpp X

Modul04&05 > UNGUIDED > C++ main.cpp > ...

```
1  #include "playlist.h"
2  #include "playlist.cpp"
3
4  Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
5  int main() {
6      List L;
7      createList(L);
8
9      infotype lagu;
10     address P;
11
12     lagu = {"Komeng", "Fafa", 2.12};
13     P = alokasi(lagu);
14     insertFirst(L, P);
15
16     lagu = {"Que Sera Sera", "Sara", 3.20};
17     P = alokasi(lagu);
18     insertFirst(L, P);
19
20     lagu = {"Only", "Fersa", 2.45};
21     P = alokasi(lagu);
22     insertLast(L, P);
23
24     lagu = {"La Vita La Avanti", "Kai", 3.14};
25     P = alokasi(lagu);
26     insertAfterN(L, P, 3);
27
28     printInfo(L);
29
30     cout << endl;
31
32     deleteByTitle(L, "Only");
33
34     cout << endl << "Setelah penghapusan:\n";
35     printInfo(L);
36
37     return 0;
38 }
```



## Screenshots Output

```
PS D:\StrukturData> cd "d:\StrukturData\Modul04&05\UNGUIDED\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
=== Daftar Lagu dalam Playlist ===
1. Judul : Que Sera Sera
   Penyanyi : Sara
   Durasi : 3.2 menit
=====
2. Judul : Komeng
   Penyanyi : Fafa
   Durasi : 2.12 menit
=====
3. Judul : Only
   Penyanyi : Fersa
   Durasi : 2.45 menit
=====
4. Judul : La Vita La Avanti
   Penyanyi : Kai
   Durasi : 3.14 menit
=====

Lagu "Only" berhasil dihapus.

Setelah penghapusan:
=== Daftar Lagu dalam Playlist ===
1. Judul : Que Sera Sera
   Penyanyi : Sara
   Durasi : 3.2 menit
=====
2. Judul : Komeng
   Penyanyi : Fafa
   Durasi : 2.12 menit
=====
3. Judul : La Vita La Avanti
   Penyanyi : Kai
   Durasi : 3.14 menit
=====
```

## Deskripsi:

Program ini merupakan implementasi Singly Linked List dalam bahasa C++ untuk mengelola playlist lagu secara dinamis. Setiap lagu disimpan sebagai node yang berisi judul, penyanyi, dan durasi, dengan pointer yang menghubungkan antar-node. Program memiliki fungsi untuk menambah lagu di awal, akhir, atau setelah posisi tertentu, menghapus lagu berdasarkan judul, serta menampilkan seluruh isi playlist. Melalui operasi seperti insertFirst, insertLast, insertAfterN, deleteByTitle, dan printInfo, program ini menunjukkan cara kerja linked list dalam menyimpan dan memanipulasi data secara fleksibel sesuai konsep pada modul struktur data.

## D. Kesimpulan

Dari praktikum modul IV dan V tentang Singly Linked List dapat disimpulkan bahwa linked list merupakan struktur data dinamis yang memungkinkan pengelolaan data secara fleksibel menggunakan pointer. Melalui implementasi dalam bahasa C++, dapat dipahami bagaimana proses pembentukan list, penyisipan elemen di awal, akhir, maupun posisi tertentu, serta penghapusan dan penelusuran data dilakukan secara efisien tanpa perlu pergeseran elemen seperti pada array. Pada program guided, implementasi dilakukan untuk menyimpan data bertipe integer, sedangkan pada program unguided dikembangkan menjadi pengelolaan playlist lagu yang menunjukkan penerapan nyata konsep linked list. Dengan demikian, praktikum ini membantu mahasiswa memahami prinsip kerja dasar struktur data dinamis dan penggunaannya dalam pemrograman.

## E. Referensi

Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2014). Data Structures and Algorithms in C++ (2nd ed.). Wiley.

Lafore, R. (2002). Data Structures and Algorithms in C++. Sams Publishing.

Malik, D. S. (2018). C++ Programming: Program Design Including Data Structures (8th ed.). Cengage Learning.

Sedgewick, R., & Wayne, K. (2011). Algorithms (4th ed.). Addison-Wesley.

Wirth, N. (1976). Algorithms + Data Structures = Programs. Prentice-Hall.