

**LAPORAN PRAKTIKUM  
STRUKTUR DATA**

**MODUL IV  
SINGLY LINKED LIST**



**Disusun Oleh :**

NAMA : Muhammad Fachri Auravyano Saka  
NIM : 103112430180

**Dosen**

FAHRUDIN MUKTI WIBOWO

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

## A. Dasar Teori

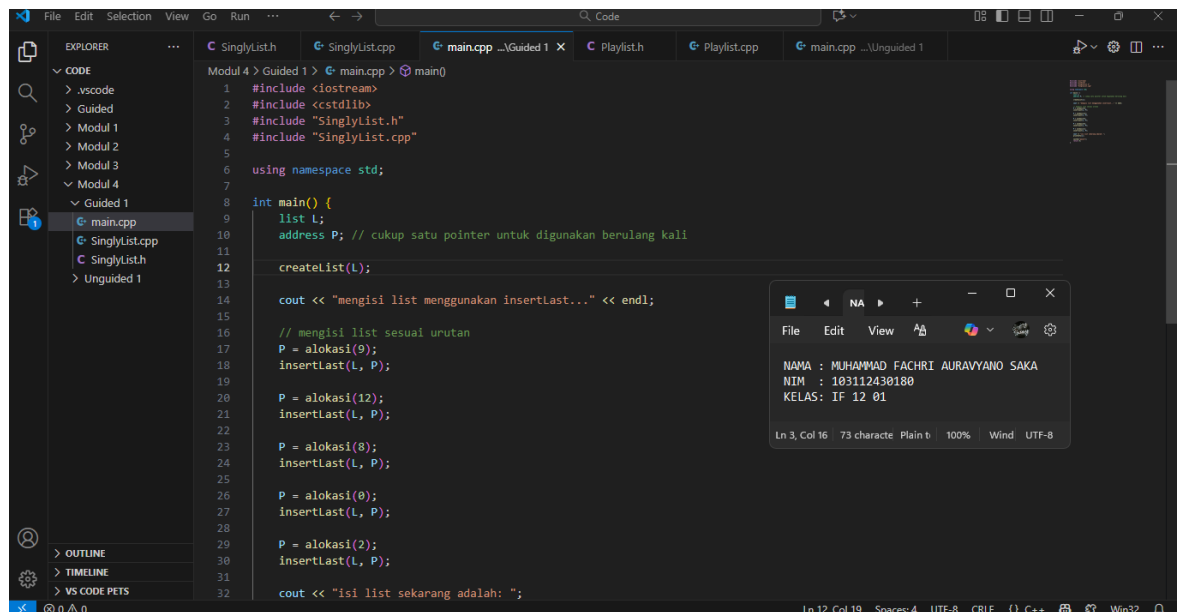
Linked List merupakan salah satu bentuk struktur data yang terdiri dari serangkaian elemen data yang saling terhubung. Berbeda dengan array yang bersifat statis, linked list bersifat dinamis dan fleksibel, artinya ukurannya dapat bertambah atau berkurang sesuai kebutuhan selama program berjalan. Karena sifatnya yang fleksibel dan saling terkait, implementasi linked list lebih cocok menggunakan pointer daripada array.

Salah satu model dasar dari linked list adalah Singly Linked List (SLL). Model ini memiliki karakteristik utama yaitu hanya memiliki satu arah pointer sebagai penghubung antar elemen, yang disebut successor atau next. Setiap elemen (juga disebut node) dalam SLL terdiri dari dua bagian utama: info, yang berisi data atau informasi utama, dan next, yaitu pointer yang menyimpan alamat dari elemen berikutnya dalam rangkaian. Elemen terakhir dalam SLL akan memiliki pointer next yang menunjuk ke NIL atau NULL, menandakan akhir dari list.

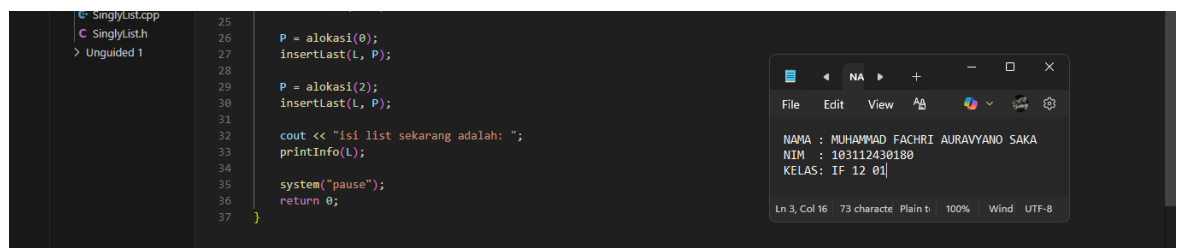
## B. Guided (berisi screenshot source code & output program disertai penjelasannya)

### Guided 1

#### Main.cpp



```
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: ";
```



```
25 P = alokasi(0);
26 insertLast(L, P);
27
28 P = alokasi(2);
29 insertLast(L, P);
30
31 cout << "isi list sekarang adalah: ";
32 printInfo(L);
33
34 system("pause");
35 return 0;
36 }
```

```
#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 insertLast..." << endl;

    // mengisi list sesuai urutan
    P = alokasi(9);
    insertLast(L, P);

    P = alokasi(12);
    insertLast(L, P);

    P = alokasi(8);
    insertLast(L, P);

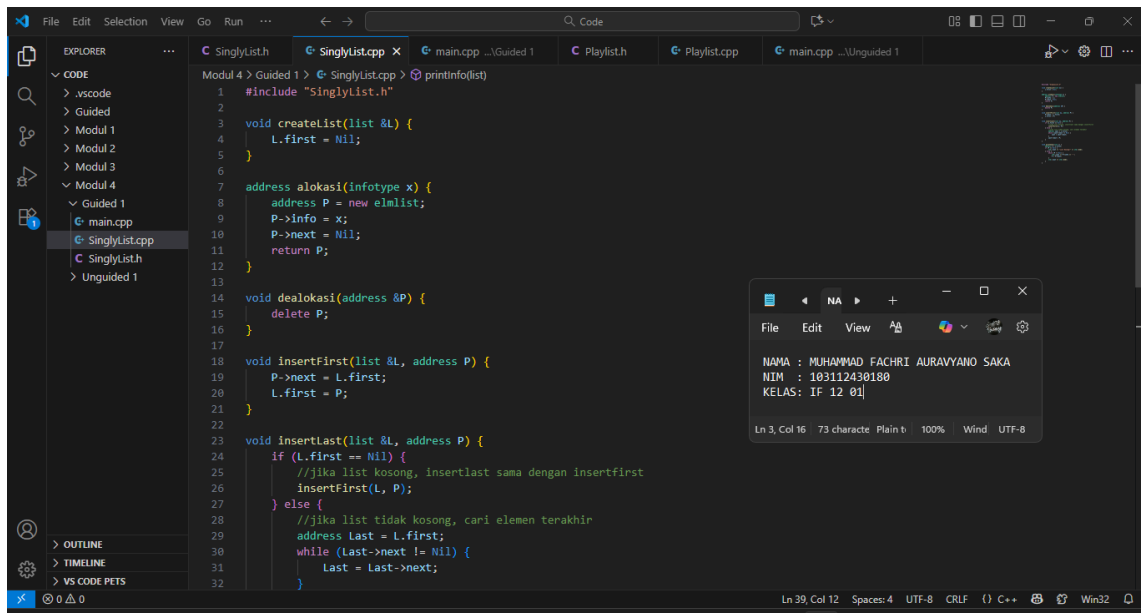
    P = alokasi(0);
    insertLast(L, P);

    P = alokasi(2);
    insertLast(L, P);


















    cout << "isi list sekarang adalah: ";
    printInfo(L);

    system("pause");
    return 0;
}

```

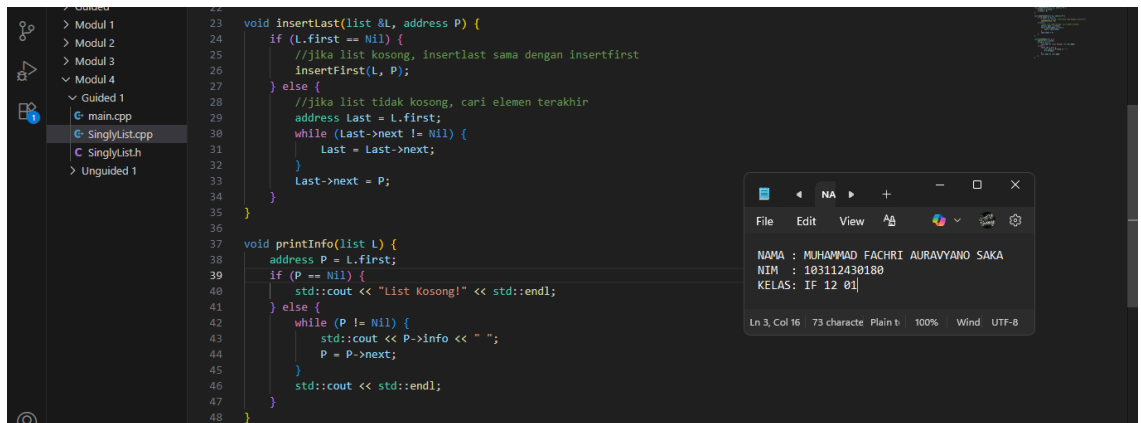


```
1 #include "SinglyList.h"
2
3 void createList(list &L) {
4     L.first = Nil;
5 }
6
7 address alokasi(infotype x) {
8     address P = new elmList;
9     P->info = x;
10    P->next = Nil;
11    return P;
12 }
13
14 void dealokasi(address &P) {
15     delete P;
16 }
17
18 void insertFirst(list &L, 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 elemen terakhir
29         address Last = L.first;
30         while (Last->next != Nil) {
31             Last = Last->next;
32         }
33         Last->next = P;
34     }
35 }
36
37 void printInfo(list L) {
38     address P = L.first;
39     if (P == Nil) {
40         std::cout << "List Kosong!" << std::endl;
41     } else {
42         while (P != Nil) {
43             std::cout << P->info << " ";
44             P = P->next;
45         }
46         std::cout << std::endl;
47     }
48 }
```


















File Edit View A                 

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180  
KELAS: IF 12 01

Ln 3, Col 16 73 character Plain t 100% Wind UTF-8



```
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 elemen terakhir
29         address Last = L.first;
30         while (Last->next != Nil) {
31             Last = Last->next;
32         }
33         Last->next = P;
34     }
35 }
36
37 void printInfo(list L) {
38     address P = L.first;
39     if (P == Nil) {
40         std::cout << "List Kosong!" << std::endl;
41     } else {
42         while (P != Nil) {
43             std::cout << P->info << " ";
44             P = P->next;
45         }
46         std::cout << std::endl;
47     }
48 }
```

File Edit View A                 

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180  
KELAS: IF 12 01

Ln 3, Col 16 73 character Plain t 100% Wind UTF-8

```
#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 elemen terakhir
        address Last = L.first;
        while (Last->next != Nil) {
            Last = Last->next;
        }
        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 << " ";

```

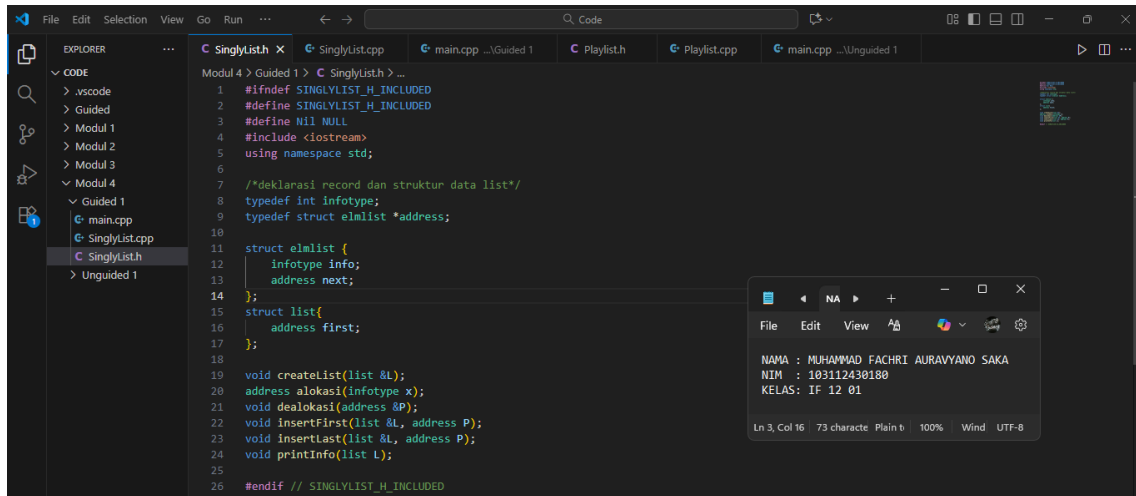
```

        P = P->next;
    }

    std::cout << std::endl;
}
}

```

## SinglyList.h



```

#ifndef SINGLYLIST_H_INCLUDED
#define SINGLYLIST_H_INCLUDED

#define Nil NULL

#include <iostream>

using namespace std;

/*deklarasi record dan struktur data list*/
typedef int infotype;
typedef struct elmlist *address;

struct elmlist {
    infotype info;
    address next;
};

struct list{
    address first;

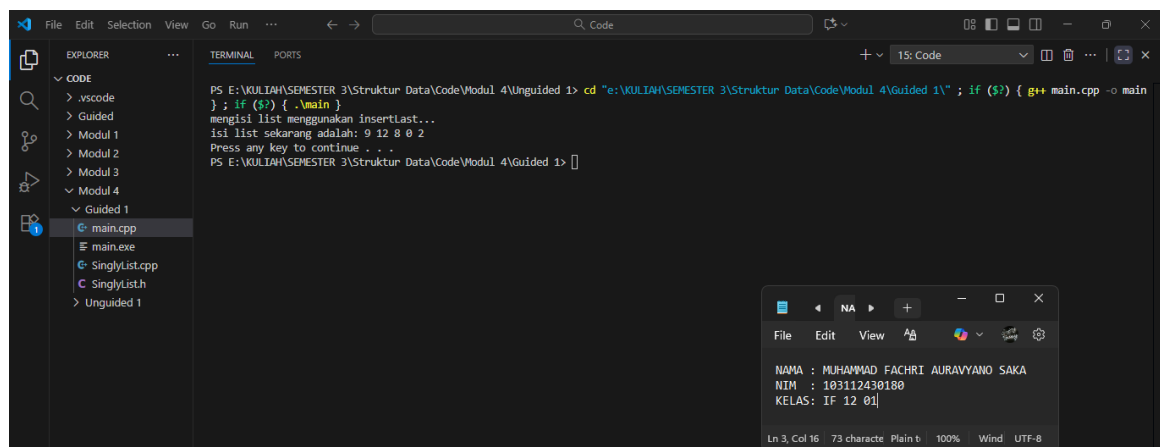
```

```
};

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_H_INCLUDED
```

## Screenshots Output



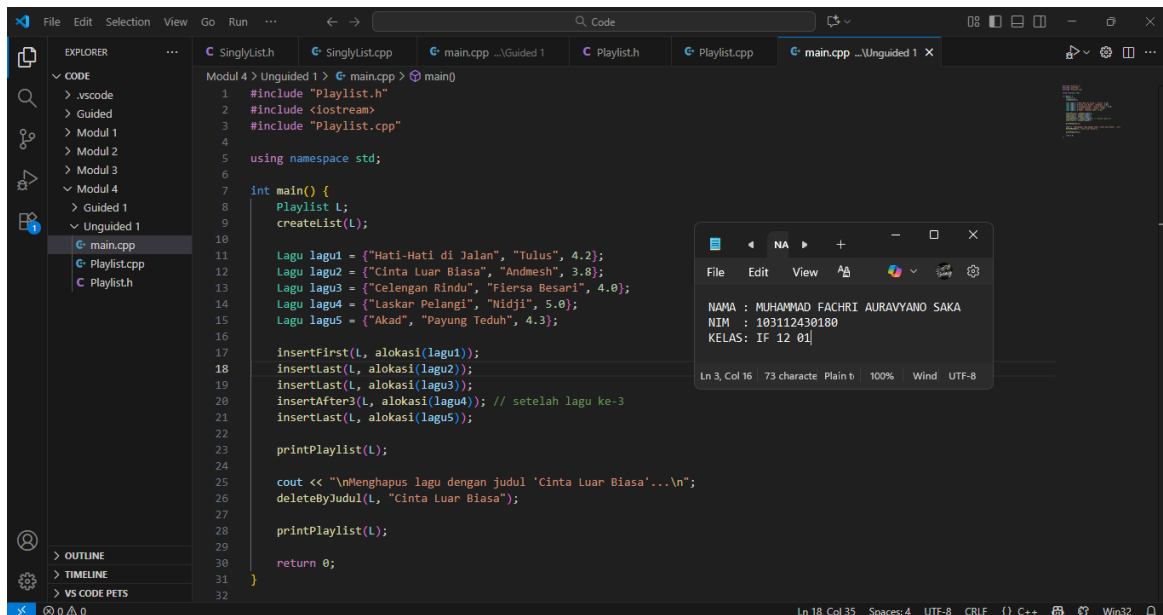
## Deskripsi:

Program yang kamu buat ini adalah implementasi dasar dari struktur data singly linked list. Intinya, program ini berfungsi untuk membuat sebuah "rantai" data, di mana setiap elemen (disebut node) menyimpan sebuah nilai integer dan menunjuk ke elemen berikutnya dalam urutan. Program utamamu (main.cpp) membuat sebuah list kosong, kemudian secara berurutan menambahkan beberapa angka (9, 12, 8, 0, dan 2) ke akhir list menggunakan fungsi insertLast. Setelah semua data dimasukkan, program akan mencetak seluruh isi list tersebut ke layar, menunjukkan data yang telah tersimpan secara berurutan. Semua logika untuk operasi list, seperti membuat list, mengalokasikan memori untuk elemen baru, dan menambahkan elemen, diatur dalam file SinglyList.cpp dan SinglyList.h.

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

Unguided 1

Main.cpp



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

using namespace std;

int main() {
    Playlist L;
    createList(L);

    Lagu lagu1 = {"Hati-Hati di Jalan", "Tulus", 4.2};
    Lagu lagu2 = {"Cinta Luar Biasa", "Andmesh", 3.8};
    Lagu lagu3 = {"Celengan Rindu", "Fiersa Besari", 4.0};
    Lagu lagu4 = {"Laskar Pelangi", "Nidji", 5.0};
    Lagu lagu5 = {"Akad", "Payung Teduh", 4.3};

    insertFirst(L, alokasi(lagu1));
    insertLast(L, alokasi(lagu2));
    insertLast(L, alokasi(lagu3));
    insertAfter3(L, alokasi(lagu4)); // setelah lagu ke-3
    insertLast(L, alokasi(lagu5));

    printPlaylist(L);

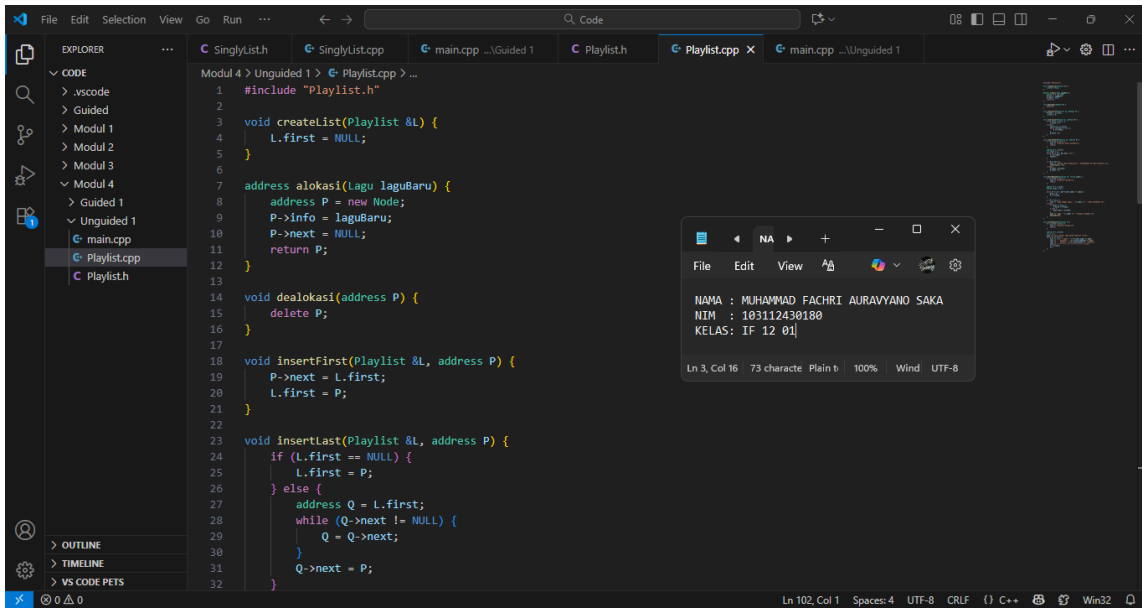
    cout << "\nMenghapus lagu dengan judul 'Cinta Luar
Biasa'...\n";
    deleteByJudul(L, "Cinta Luar Biasa");

    printPlaylist(L);

    return 0;
}
```

Playlist.cpp





```
1 #include "Playlist.h"
2
3 void createList(Playlist &L) {
4     L.first = NULL;
5 }
6
7 address alokasi(Lagu laguBaru) {
8     address P = new Node;
9     P->info = laguBaru;
10    P->next = NULL;
11    return P;
12 }
13
14 void dealokasi(address P) {
15     delete P;
16 }
17
18 void insertFirst(Playlist &L, address P) {
19     P->next = L.first;
20     L.first = P;
21 }
22
23 void insertLast(Playlist &L, address P) {
24     if (L.first == NULL) {
25         L.first = P;
26     } else {
27         address Q = L.first;
28         while (Q->next != NULL) {
29             Q = Q->next;
30         }
31         Q->next = P;
32 }
```

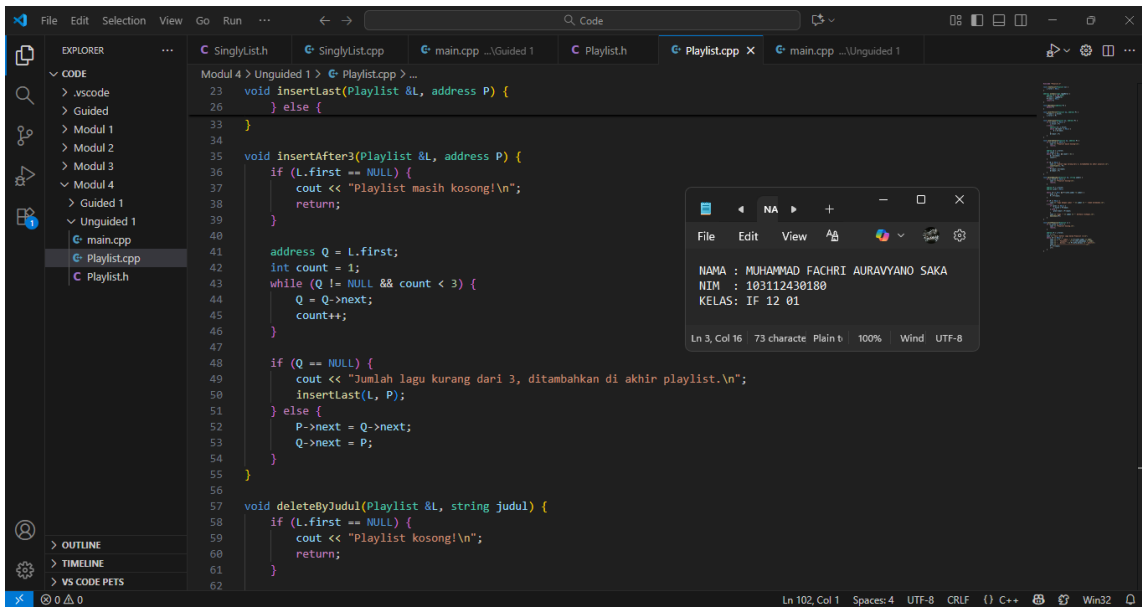
Modul 4 > Unguided 1 > Playlist.cpp

File Edit View

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180  
KELAS: IF 12 01

Ln 3, Col 16 73 character Plain b 100% Wind UTF-8

Ln 102, Col 1 Spaces:4 UTF-8 CRLF {} C++ Win32



```
23 void insertLast(Playlist &L, address P) {
24     } else {
25
26     }
27
28 void insertAfter3(Playlist &L, address P) {
29     if (L.first == NULL) {
30         cout << "Playlist masih kosong!\n";
31         return;
32     }
33
34     address Q = L.first;
35     int count = 1;
36     while (Q != NULL && count < 3) {
37         Q = Q->next;
38         count++;
39     }
40
41     if (Q == NULL) {
42         cout << "Jumlah lagu kurang dari 3, ditambahkan di akhir playlist.\n";
43         insertLast(L, P);
44     } else {
45         P->next = Q->next;
46         Q->next = P;
47     }
48
49 void deleteByJudul(Playlist &L, string judul) {
50     if (L.first == NULL) {
51         cout << "Playlist kosong!\n";
52         return;
53     }
54
55     address P = L.first;
56     address prev = NULL;
57
58     while (P != NULL && P->info.judul != judul) {
59         prev = P;
60         P = P->next;
61     }
62
63     if (P == NULL) {
64         cout << "Lagu dengan judul " << judul << " tidak ditemukan.\n";
65     } else {
66         if (prev == NULL) {
67             L.first = P->next;
68         } else {
69             prev->next = P->next;
70         }
71         cout << "Lagu " << judul << " berhasil dihapus.\n";
72         dealokasi(P);
73     }
74
75 void printPlaylist(Playlist L) {
76     if (L.first == NULL) {
77         cout << "Playlist kosong.\n";
78         return;
79     }
80
81     address P = L.first;
82     int i = 1;
83     cout << "\n== Daftar Lagu dalam Playlist ==\n";
```

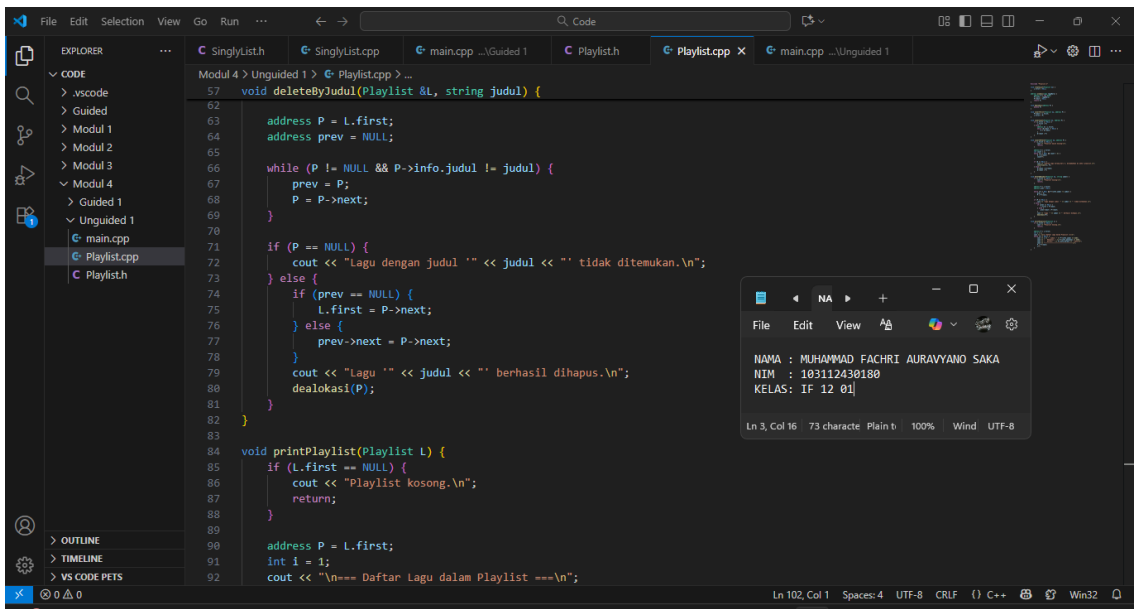
Modul 4 > Unguided 1 > Playlist.cpp

File Edit View

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180  
KELAS: IF 12 01

Ln 3, Col 16 73 character Plain b 100% Wind UTF-8

Ln 102, Col 1 Spaces:4 UTF-8 CRLF {} C++ Win32



```
57 void deleteByJudul(Playlist &L, string judul) {
58
59     address P = L.first;
60     address prev = NULL;
61
62     while (P != NULL && P->info.judul != judul) {
63         prev = P;
64         P = P->next;
65     }
66
67     if (P == NULL) {
68         cout << "Lagu dengan judul " << judul << " tidak ditemukan.\n";
69     } else {
70         if (prev == NULL) {
71             L.first = P->next;
72         } else {
73             prev->next = P->next;
74         }
75         cout << "Lagu " << judul << " berhasil dihapus.\n";
76         dealokasi(P);
77     }
78
79 void printPlaylist(Playlist L) {
80     if (L.first == NULL) {
81         cout << "Playlist kosong.\n";
82         return;
83     }
84
85     address P = L.first;
86     int i = 1;
87     cout << "\n== Daftar Lagu dalam Playlist ==\n";
```

Modul 4 > Unguided 1 > Playlist.cpp

File Edit View

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180  
KELAS: IF 12 01

Ln 3, Col 16 73 character Plain b 100% Wind UTF-8

Ln 102, Col 1 Spaces:4 UTF-8 CRLF {} C++ Win32



```
#include "Playlist.h"

void createList(Playlist &L) {
    L.first = NULL;
}

address alokasi(Lagu laguBaru) {
    address P = new Node;
    P->info = laguBaru;
    P->next = NULL;
    return P;
}

void dealokasi(address P) {
    delete P;
}

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

void insertLast(Playlist &L, address P) {
    if (L.first == NULL) {
        L.first = P;
    }
}
```

```

    } else {
        address Q = L.first;
        while (Q->next != NULL) {
            Q = Q->next;
        }
        Q->next = P;
    }
}

void insertAfter3(Playlist &L, address P) {
    if (L.first == NULL) {
        cout << "Playlist masih kosong!\n";
        return;
    }

    address Q = L.first;
    int count = 1;
    while (Q != NULL && count < 3) {
        Q = Q->next;
        count++;
    }

    if (Q == NULL) {
        cout << "Jumlah lagu kurang dari 3, ditambahkan di akhir playlist.\n";
        insertLast(L, P);
    } else {
        P->next = Q->next;
        Q->next = P;
    }
}

```

```

void deleteByJudul(Playlist &L, string judul) {
    if (L.first == NULL) {
        cout << "Playlist kosong!\n";
        return;
    }

    address P = L.first;
    address prev = NULL;

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

    if (P == NULL) {
        cout << "Lagu dengan judul '" << judul << "' tidak
ditemukan.\n";
    } else {
        if (prev == NULL) {
            L.first = P->next;
        } else {
            prev->next = P->next;
        }

        cout << "Lagu '" << judul << "' berhasil
dihapus.\n";
        dealokasi(P);
    }
}

void printPlaylist(Playlist L) {

```

```

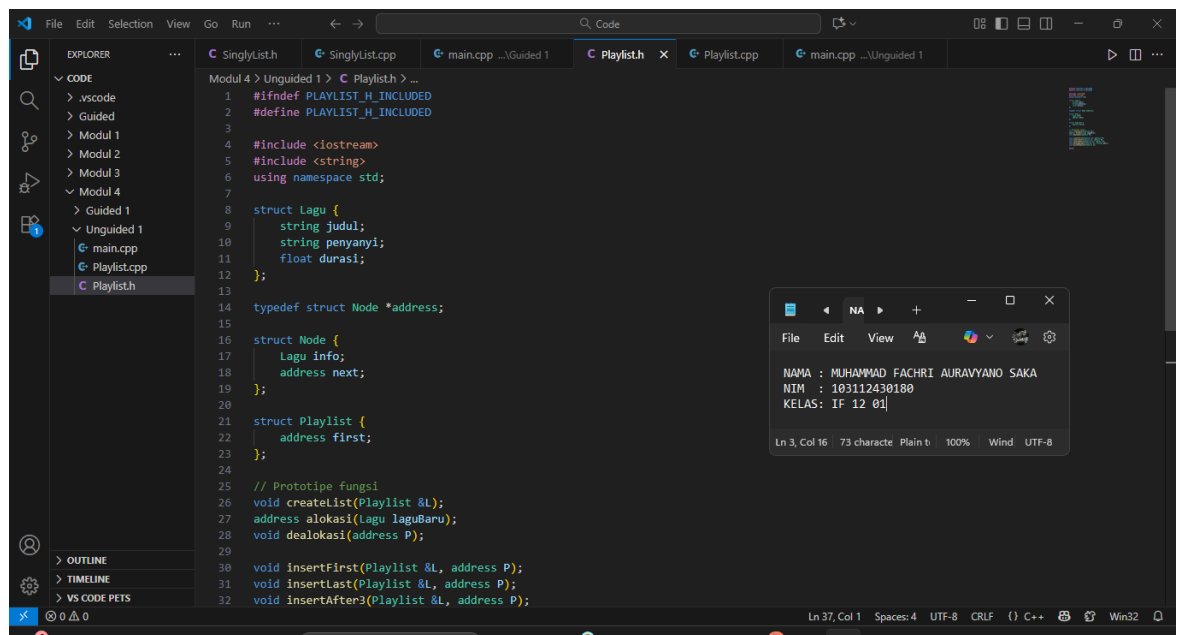
    if (L.first == NULL) {
        cout << "Playlist kosong.\n";
        return;
    }

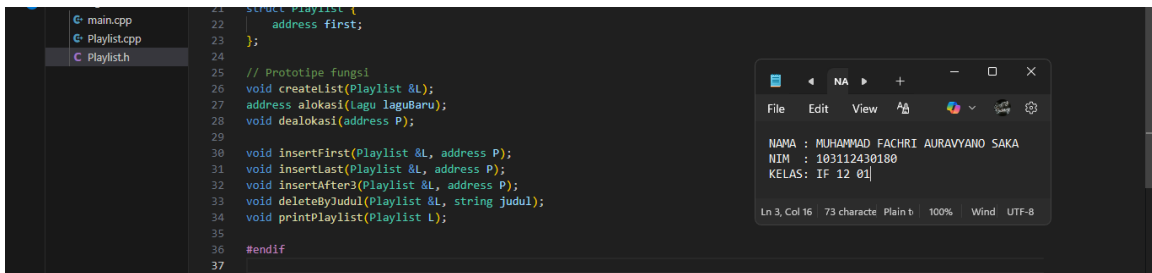
    address P = L.first;
    int i = 1;
    cout << "\n=== Daftar Lagu dalam Playlist ===\n";
    while (P != NULL) {
        cout << i << ". Judul: " << P->info.judul << endl;
        cout << "    Penyanyi: " << P->info.penyanyi << endl;
        cout << "        Durasi: " << P->info.durasi << "
menit\n";

        cout << "-----\n";
        P = P->next;
        i++;
    }
}

```

## 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;
};

// Prototipe fungsi
void createList(Playlist &L);
```

```

address alokasi(Lagu laguBaru);

void dealokasi(address P);

void insertFirst(Playlist &L, address P);
void insertLast(Playlist &L, address P);
void insertAfter3(Playlist &L, address P);
void deleteByJudul(Playlist &L, string judul);
void printPlaylist(Playlist L);

#endif

```

## Screenshot Output

```

PS E:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 4\Unguided 1> cd "e:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 4\Unguided 1\" ; if ($?) { g++ main.cpp -o mai
n } ; if ($?) { .\main }

=== Daftar Lagu dalam Playlist ===
1. Judul: Hati-Hati di Jalan
   Penyanyi: Tulus
   Durasi: 4.2 menit
-----
2. Judul: Cinta Luar Biasa
   Penyanyi: Andmesh
   Durasi: 3.8 menit
-----
3. Judul: Celengan Rindu
   Penyanyi: Fiersa Besari
   Durasi: 4 menit
-----
4. Judul: Laskar Pelangi
   Penyanyi: Nidji
   Durasi: 5 menit
-----
5. Judul: Akad
   Penyanyi: Payung Teduh
   Durasi: 4.3 menit
-----

Menghapus lagu dengan judul 'Cinta Luar Biasa'...
Lagu 'Cinta Luar Biasa' berhasil dihapus.

=== Daftar Lagu dalam Playlist ===
1. Judul: Hati-Hati di Jalan
   Penyanyi: Tulus
   Durasi: 4.2 menit
-----
2. Judul: Celengan Rindu
   Penyanyi: Fiersa Besari
   Durasi: 4 menit
-----
3. Judul: Laskar Pelangi
   Penyanyi: Nidji
   Durasi: 5 menit
-----
4. Judul: Akad
   Penyanyi: Payung Teduh
   Durasi: 4.3 menit
-----

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
KELAS: IF 12 01

```

```

PS E:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 4\Unguided 1>

Menghapus lagu dengan judul 'Cinta Luar Biasa'...
Lagu 'Cinta Luar Biasa' berhasil dihapus.

=== Daftar Lagu dalam Playlist ===
1. Judul: Hati-Hati di Jalan
   Penyanyi: Tulus
   Durasi: 4.2 menit
-----
2. Judul: Celengan Rindu
   Penyanyi: Fiersa Besari
   Durasi: 4 menit
-----
3. Judul: Laskar Pelangi
   Penyanyi: Nidji
   Durasi: 5 menit
-----
4. Judul: Akad
   Penyanyi: Payung Teduh
   Durasi: 4.3 menit
-----

```

## Deskripsi:

Program ini pada dasarnya adalah simulasi manajemen playlist musik sederhana menggunakan struktur data singly linked list di C++. Setiap lagu yang punya data judul,

penyanyi, dan durasi—disimpan dalam sebuah node atau simpul. Rangkaian node inilah yang membentuk playlist-nya.

Logika utamanya ada di file `main.cpp` yang bertindak sebagai "driver". Di sana, program pertama-tama membuat sebuah playlist kosong. Kemudian, beberapa lagu didefinisikan dan dimasukkan ke dalam playlist dengan berbagai cara: ada yang dimasukkan di awal (`insertFirst`), ada yang di akhir (`insertLast`), dan ada satu lagu yang disisipkan secara spesifik setelah lagu ketiga (`insertAfter3`). Setelah playlist awal terbentuk, program akan menampilkan seluruh isinya. Sebagai demonstrasi fitur tambahan, program kemudian mencari dan menghapus salah satu lagu berdasarkan judulnya (`deleteByJudul`), lalu menampilkan kembali isi playlist yang sudah diperbarui. Semua fungsi inti seperti alokasi memori, penyisipan, penghapusan, dan pencetakan didefinisikan secara terpisah di file `Playlist.cpp` dan `Playlist.h` agar kodenya lebih rapi dan terstruktur.

#### D. Kesimpulan

Linked list adalah struktur data dinamis yang memungkinkan alokasi memori yang fleksibel, di mana elemen dapat ditambah atau dikurangi sesuai kebutuhan. Implementasinya lebih efektif menggunakan pointer karena sifatnya yang dinamis, berbeda dengan array yang statis.

Operasi-operasi dasar pada SLL seperti Insert (penyisipan) , Delete (penghapusan) , dan Update (pengubahan data) sangat bergantung pada operasi fundamental lainnya, yaitu Searching (pencarian). Operasi pencarian berfungsi untuk menemukan node tertentu sebagai acuan sebelum melakukan modifikasi pada list.

#### E. Referensi

Castillo III, L. S. (2016). Interactive System for Visualization of Linked List Operations in C++.

Furcy, D. (2009). JHAVEPOP: Visualizing linked-list operations in C++ and Java. *Journal of Computing Sciences in Colleges*, 25(1), 32-41.