

**LAPORAN PRAKTIKUM  
STRUKTUR DATA**

**MODUL IV  
SINGLY LINKED LIST**



**Disusun Oleh :**

Nama : Besthian Guido Rafael Simbolon  
NIM : 103112430258

**Dosen**

FAHRUDIN MUKTI WIBOWO

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

## A. Dasar Teori

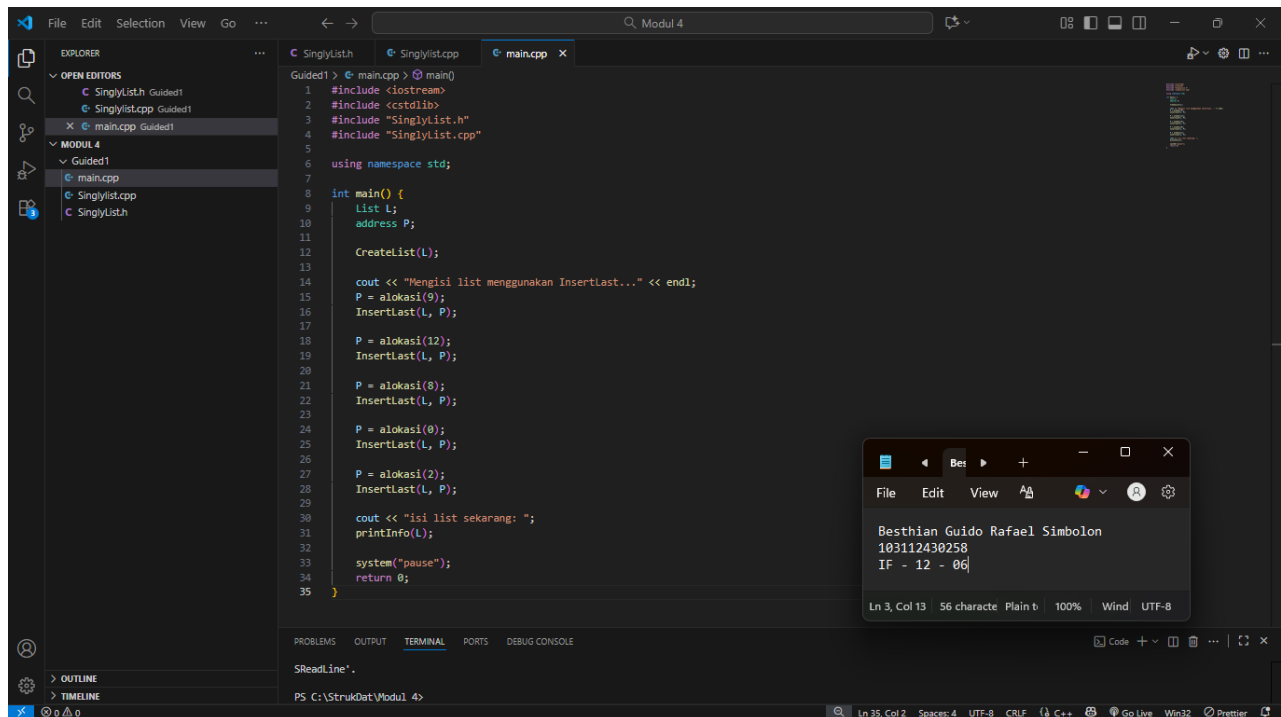
Singly Linked List adalah salah satu bentuk struktur data dinamis yang terdiri dari beberapa elemen atau node yang saling terhubung satu arah menggunakan pointer. Setiap node memiliki dua bagian utama, yaitu data untuk menyimpan informasi dan pointer yang menunjuk ke node berikutnya. Node pertama disebut head, sedangkan node terakhir menunjuk ke NULL karena tidak ada elemen setelahnya. Berbeda dengan array yang bersifat statis, linked list bersifat fleksibel sehingga ukuran datanya bisa berubah sesuai kebutuhan program. Struktur ini banyak digunakan karena mempermudah penambahan dan penghapusan data tanpa perlu menggeser elemen seperti pada array.

Dalam penggunaannya, singly linked list memiliki beberapa operasi dasar, seperti membuat list baru (create list), menambah data (insert), menghapus data (delete), menampilkan isi list (view), dan memperbarui data (update). Semua operasi tersebut dilakukan dengan memanfaatkan pointer untuk menghubungkan antar node. Kelebihan dari linked list adalah efisiensinya dalam memanipulasi data, namun kekurangannya terletak pada cara akses yang harus dilakukan secara berurutan dari awal list. Dengan memahami konsep singly linked list, kita dapat mengelola data yang jumlahnya tidak tetap sekaligus memahami cara kerja memori dinamis dan pointer dalam pemrograman.

## B. Guided

### 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;
11
12    CreateList(L);
13
14    cout << "Mengisi list menggunakan InsertLast..." << endl;
15    P = alokasi(9);
16    InsertLast(L, P);
17
18    P = alokasi(12);
19    InsertLast(L, P);
20
21    P = alokasi(8);
22    InsertLast(L, P);
23
24    P = alokasi(0);
25    InsertLast(L, P);
26
27    P = alokasi(2);
28    InsertLast(L, P);
29
30    cout << "Isi list sekarang: ";
31    printInfo(L);
32
33    system("pause");
34    return 0;
35 }
```

Besthian Guido Rafael Simbolon  
103112430258  
IF - 12 - 06

Code:

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

using namespace std;

int main() {
    List L;
    address P;

    CreateList(L);

    cout << "Mengisi list menggunakan InsertLast..." << endl;
    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: ";
```

```

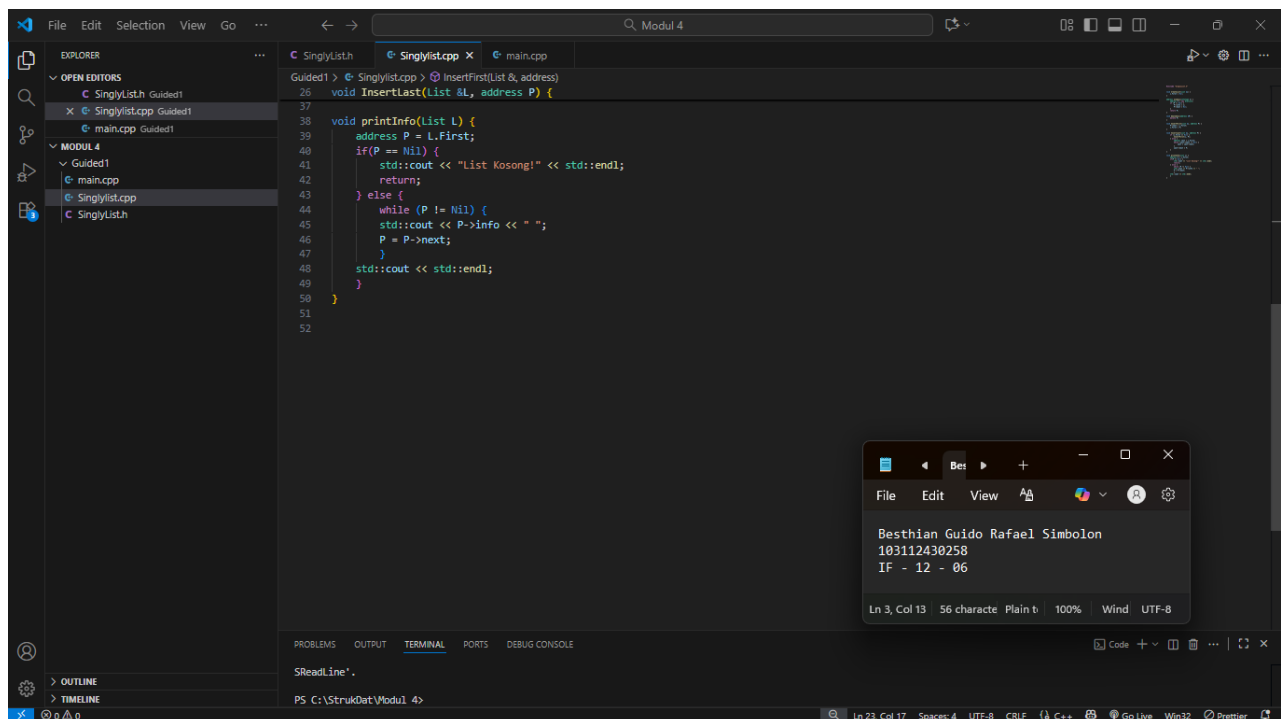
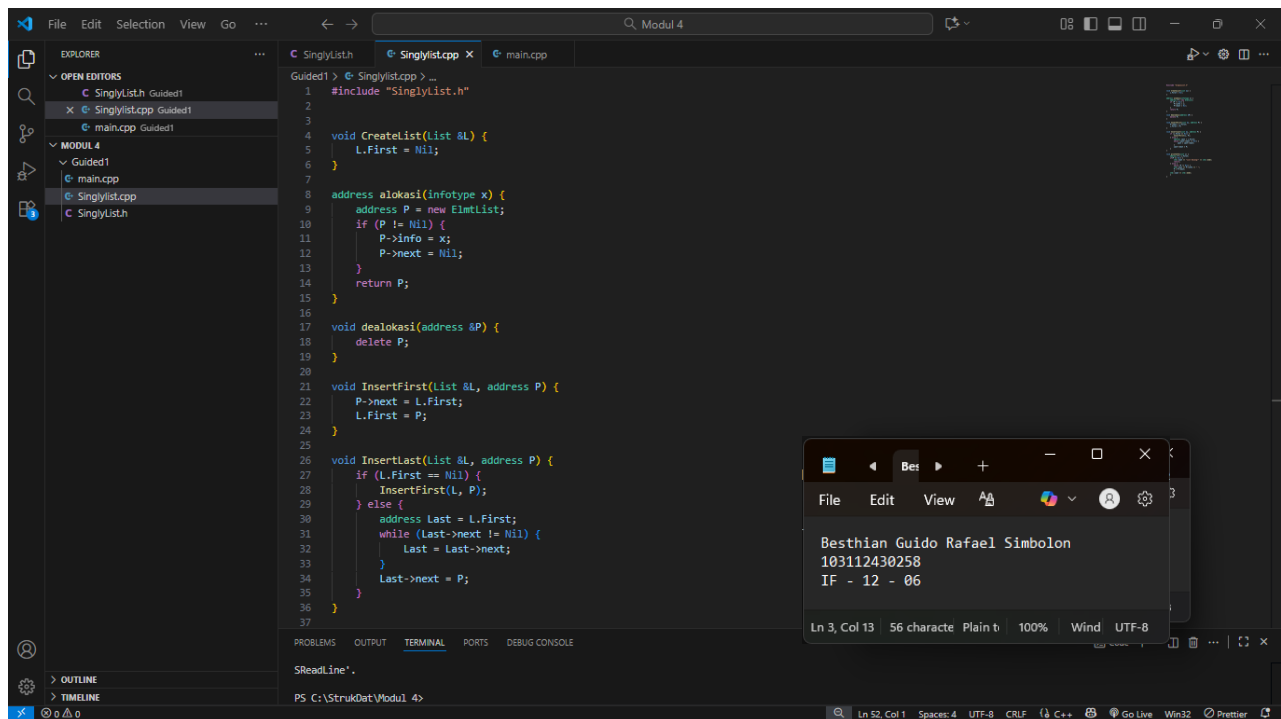
printInfo(L);

system("pause");

return 0;
}

```

## Singlylist.cpp



Code:

```
#include "SinglyList.h"

void CreateList(List &L) {
    L.First = Nil;
}

address alokasi(infotype x) {
    address P = new ElmtList;
    if (P != Nil) {
        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;
}

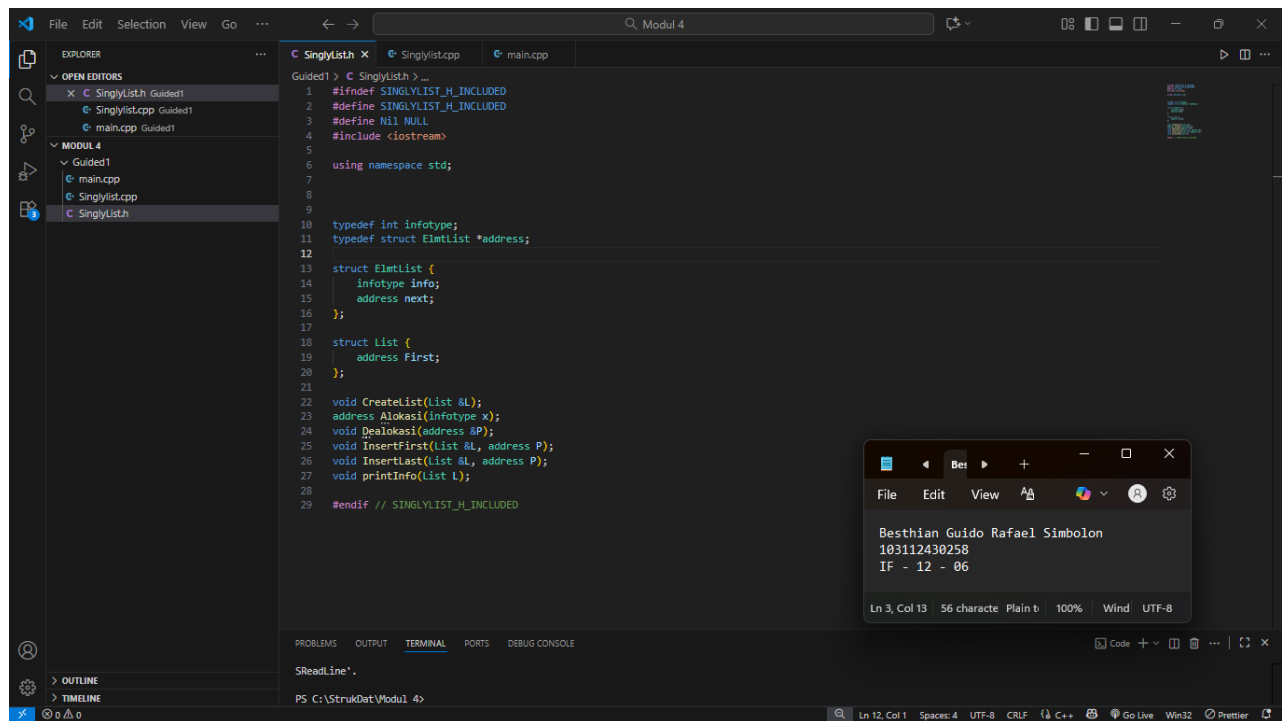
void InsertLast(List &L, address P) {
    if (L.First == Nil) {
        InsertFirst(L, P);
    } else {
        address Last = L.First;
        while (Last->next != Nil) {
            Last = Last->next;
        }
        Last->next = P;
    }
}
```

```

void printInfo(List L) {
    address P = L.First;
    if(P == Nil) {
        std::cout << "List Kosong!" << std::endl;
        return;
    } else {
        while (P != Nil) {
            std::cout << P->info << " ";
            P = P->next;
        }
        std::cout << std::endl;
    }
}

```

## Singlylist.h



Code :

```

#ifndef SINGLYLIST_H_INCLUDED
#define SINGLYLIST_H_INCLUDED
#define Nil NULL

```

```
#include <iostream>

using namespace std;

typedef int infotype;
typedef struct ElmtList *address;

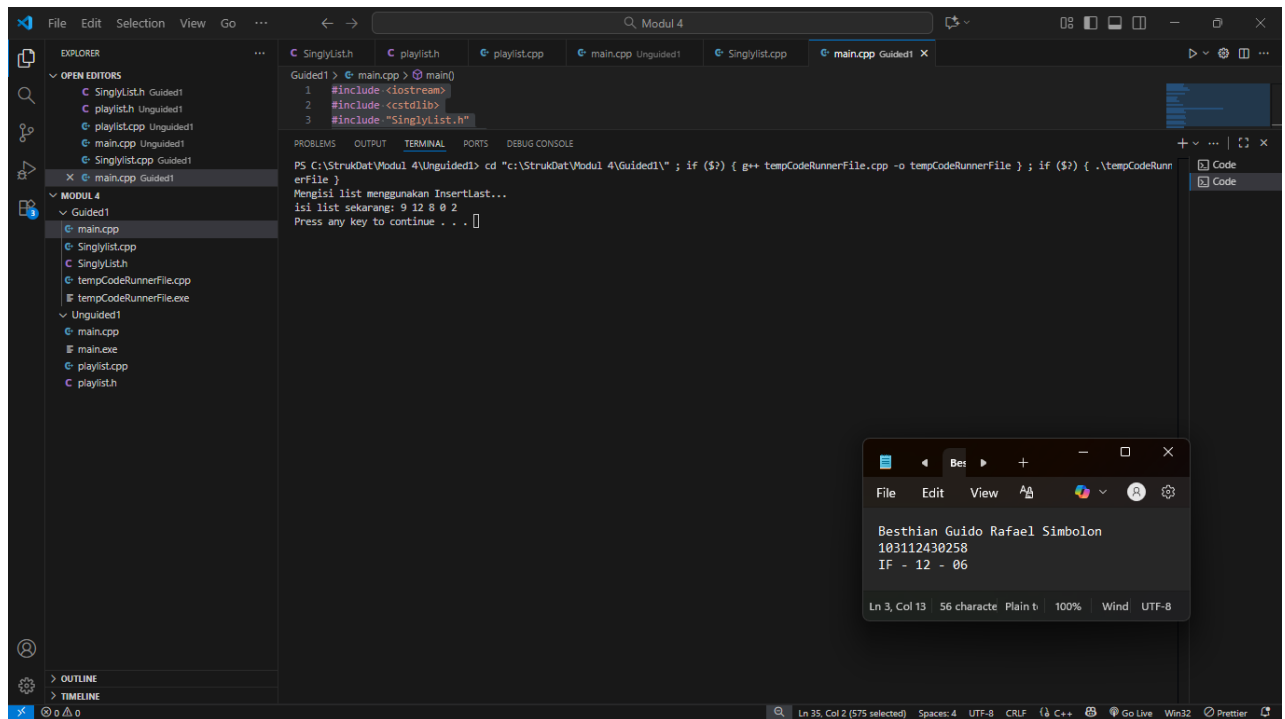
struct ElmtList {
    infotype info;
    address next;
};

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 di atas adalah contoh penerapan struktur data Singly Linked List menggunakan bahasa C++. Program ini menyimpan sekumpulan data integer dalam bentuk node yang saling terhubung satu arah. Setiap node memiliki dua bagian, yaitu info untuk menyimpan data dan next untuk menunjuk ke node berikutnya. Proses pembentukan list diawali dengan fungsi `CreateList` yang membuat list kosong. Kemudian, fungsi alokasi digunakan untuk membuat node baru, dan `InsertLast` berfungsi menambahkan node di bagian akhir list.

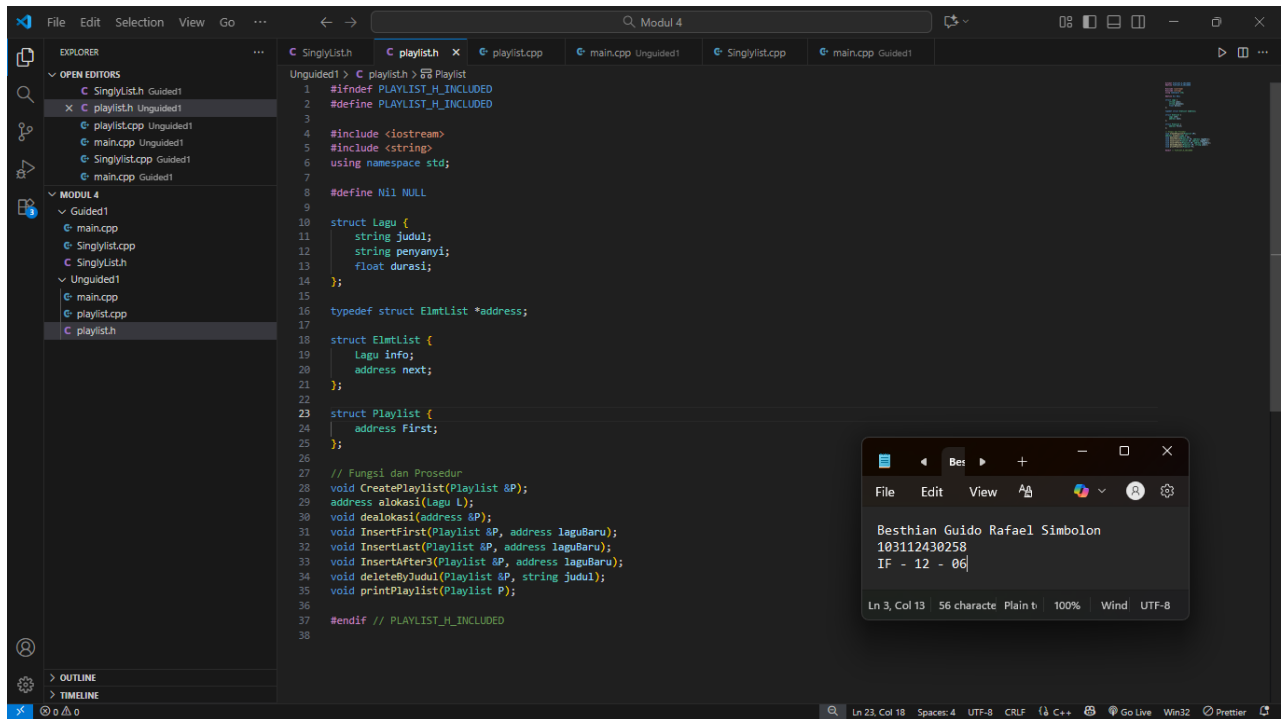
Dalam fungsi `main`, program membuat list kosong, lalu menambahkan beberapa data (9, 12, 8, 0, dan 2) ke dalamnya menggunakan `InsertLast`. Setelah semua data dimasukkan, fungsi `printInfo` dipanggil untuk menampilkan isi list ke layar. Hasil akhirnya, program akan menampilkan data secara berurutan sesuai urutan saat dimasukkan. Secara sederhana, program ini menunjukkan bagaimana cara menyimpan, menambah, dan menampilkan data menggunakan konsep pointer pada struktur Singly Linked List



## C. Unguided/Latihan

### Unguided 1

#### Playlist.h



```
#ifndef PLAYLIST_H_INCLUDED

#define PLAYLIST_H_INCLUDED

#include <iostream>

#include <string>

using namespace std;

#define Nil NULL

struct Lagu {

    string judul;

    string penyanyi;
```

```
        float durasi;

};

typedef struct ElmtList *address;

struct ElmtList {

    Lagu info;

    address next;

};

struct Playlist {

    address First;

};

// Fungsi dan Prosedur

void CreatePlaylist(Playlist &P);

address alokasi(Lagu L);

void dealokasi(address &P);

void InsertFirst(Playlist &P, address laguBaru);

void InsertLast(Playlist &P, address laguBaru);

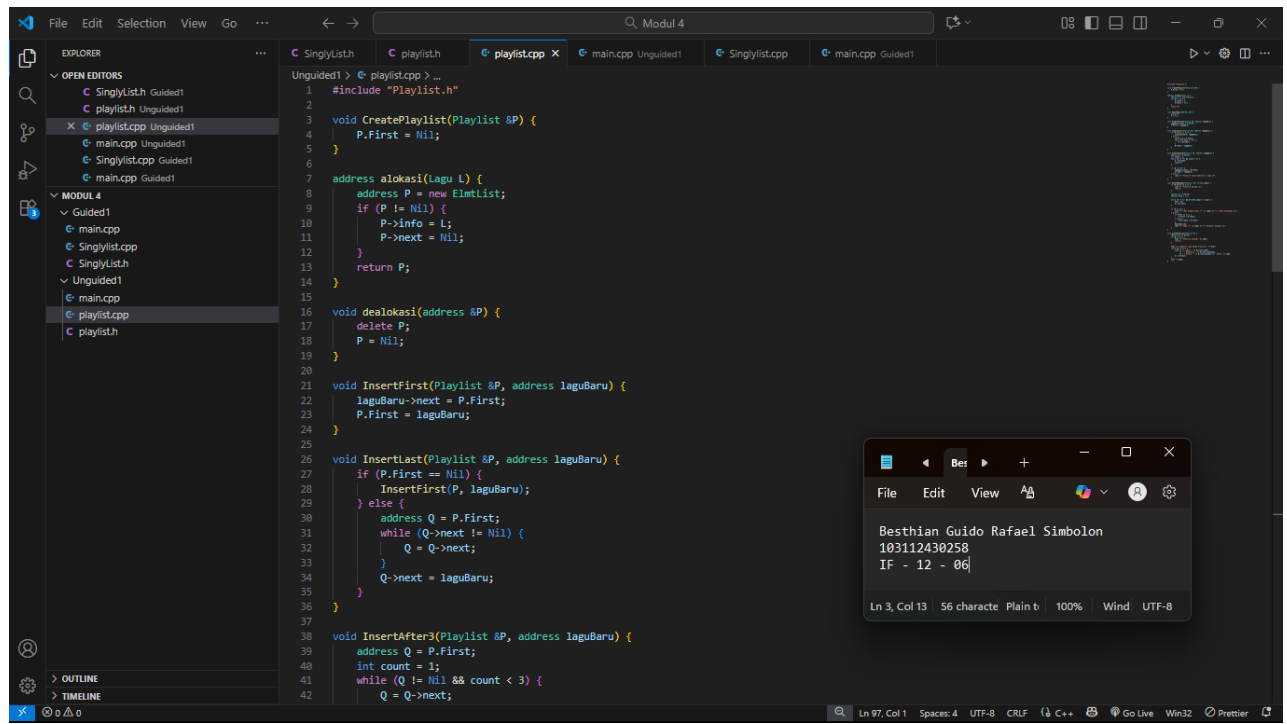
void InsertAfter3(Playlist &P, address laguBaru);

void deleteByJudul(Playlist &P, string judul);

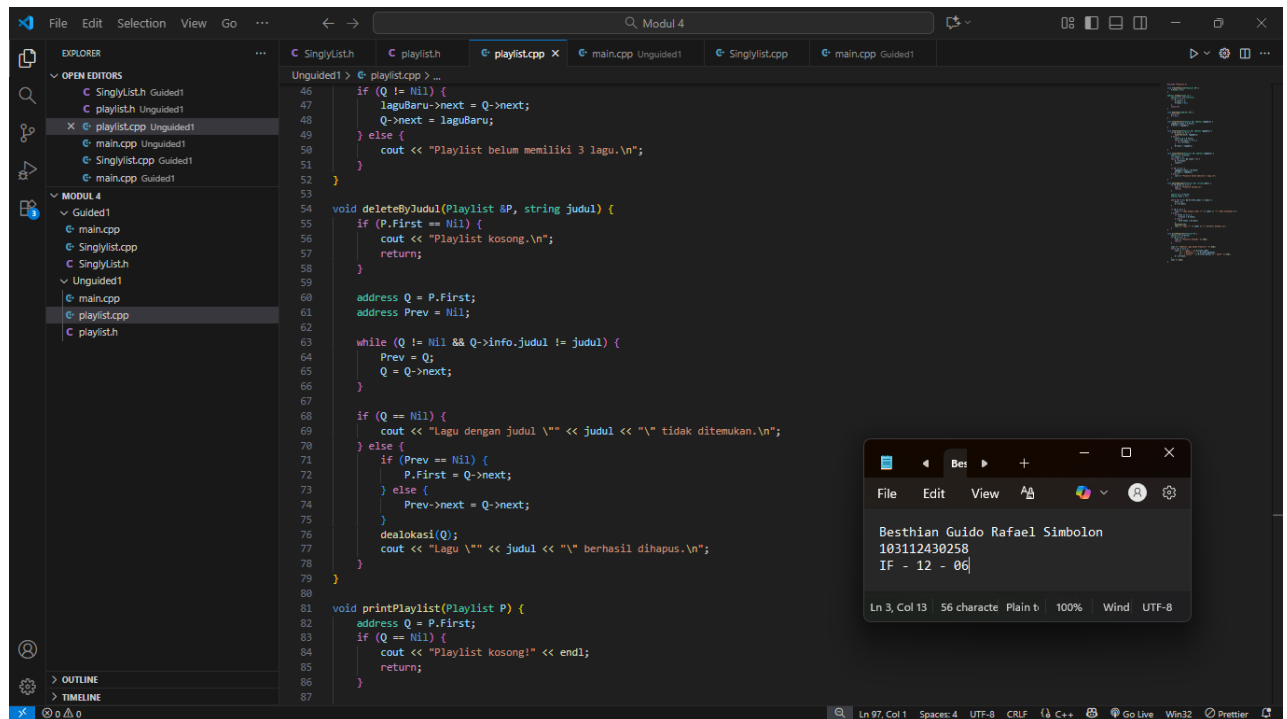
void printPlaylist(Playlist P);

#endif // PLAYLIST_H_INCLUDED
```

## Playlist.cpp



```
1 #include "Playlist.h"
2
3 void CreatePlaylist(Playlist &P) {
4     P.First = Nil;
5 }
6
7 address alokasi(Lagu L) {
8     address P = new ElmtList;
9     if (P != Nil) {
10         P->info = L;
11         P->next = Nil;
12     }
13     return P;
14 }
15
16 void dealokasi(address &P) {
17     delete P;
18     P = Nil;
19 }
20
21 void InsertFirst(Playlist &P, address laguBaru) {
22     laguBaru->next = P.First;
23     P.First = laguBaru;
24 }
25
26 void InsertLast(Playlist &P, address laguBaru) {
27     if (P.First == Nil) {
28         InsertFirst(P, laguBaru);
29     } else {
30         address Q = P.First;
31         while (Q->next != Nil) {
32             Q = Q->next;
33         }
34         Q->next = laguBaru;
35     }
36 }
37
38 void InsertAfter3(Playlist &P, address laguBaru) {
39     address Q = P.First;
40     int count = 1;
41     while (Q != Nil && count < 3) {
42         Q = Q->next;
```



```
46 if (Q != Nil) {
47     laguBaru->next = Q->next;
48     Q->next = laguBaru;
49 } else {
50     cout << "Playlist belum memiliki 3 lagu.\n";
51 }
52 }
53
54 void deleteByJudul(Playlist &P, string judul) {
55     if (P.First == Nil) {
56         cout << "Playlist kosong.\n";
57         return;
58     }
59
60     address Q = P.First;
61     address Prev = Nil;
62
63     while (Q != Nil && Q->info.judul != judul) {
64         Prev = Q;
65         Q = Q->next;
66     }
67
68     if (Q == Nil) {
69         cout << "Lagu dengan judul \"" << judul << "\" tidak ditemukan.\n";
70     } else {
71         if (Prev == Nil) {
72             P.First = Q->next;
73         } else {
74             Prev->next = Q->next;
75         }
76         dealokasi(Q);
77         cout << "Lagu \"" << judul << "\" berhasil dihapus.\n";
78     }
79 }
80
81 void printPlaylist(Playlist P) {
82     address Q = P.First;
83     if (Q == Nil) {
84         cout << "Playlist kosong!" << endl;
85         return;
86     }
87 }
```

Code: #include "Playlist.h"

```
void CreatePlaylist(Playlist &P) {
```

```

        P.First = Nil;
    }

address alokasi(Lagu L) {
    address P = new ElmtList;

    if (P != Nil) {
        P->info = L;
        P->next = Nil;
    }

    return P;
}

void dealokasi(address &P) {
    delete P;

    P = Nil;
}

void InsertFirst(Playlist &P, address laguBaru) {
    laguBaru->next = P.First;

    P.First = laguBaru;
}

void InsertLast(Playlist &P, address laguBaru) {
    if (P.First == Nil) {
        InsertFirst(P, laguBaru);
    } else {

```

```

        address Q = P.First;

        while (Q->next != Nil) {

            Q = Q->next;

        }

        Q->next = laguBaru;

    }
}

void InsertAfter3(Playlist &P, address laguBaru) {

    address Q = P.First;

    int count = 1;

    while (Q != Nil && count < 3) {

        Q = Q->next;

        count++;

    }

    if (Q != Nil) {

        laguBaru->next = Q->next;

        Q->next = laguBaru;

    } else {

        cout << "Playlist belum memiliki 3 lagu.\n";

    }

}

void deleteByJudul(Playlist &P, string judul) {

    if (P.First == Nil) {

```

```

        cout << "Playlist kosong.\n";

        return;
    }

    address Q = P.First;

    address Prev = Nil;

    while (Q != Nil && Q->info.judul != judul) {

        Prev = Q;

        Q = Q->next;

    }

    if (Q == Nil) {

        cout << "Lagu dengan judul \"" << judul << "\" tidak
ditemukan.\n";

    } else {

        if (Prev == Nil) {

            P.First = Q->next;

        } else {

            Prev->next = Q->next;

        }

        dealokasi(Q);

        cout << "Lagu \"" << judul << "\" berhasil
dihapus.\n";

    }

}

```

```

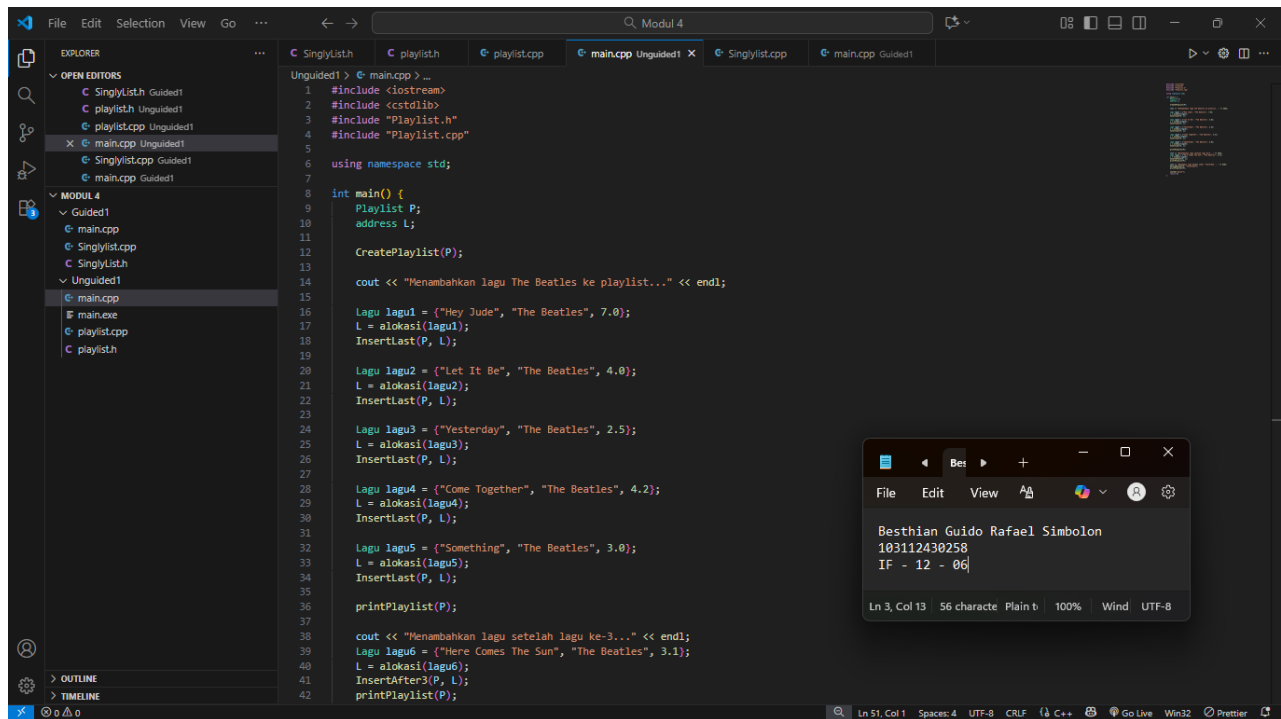
void printPlaylist(Playlist P) {
    address Q = P.First;

    if (Q == Nil) {
        cout << "Playlist kosong!" << endl;
        return;
    }

    cout << "\nDaftar Lagu dalam Playlist:" << endl;
    while (Q != Nil) {
        cout << "- Judul: " << Q->info.judul
            << " | Penyanyi: " << Q->info.penyanyi
            << " | Durasi: " << Q->info.durasi << " menit"
        << endl;
        Q = Q->next;
    }
    cout << endl;
}

```

Main.cpp



Code :

```
#include <iostream>

#include <cstdlib>

#include "Playlist.h"

#include "Playlist.cpp"

using namespace std;

int main() {

    Playlist P;

    address L;

    CreatePlaylist(P);

    cout << "Menambahkan lagu The Beatles ke playlist..." << endl;
```



```
Lagu lagu1 = {"Hey Jude", "The Beatles", 7.0};  
  
L = alokasi(lagu1);  
  
InsertLast(P, L);  
  
  
Lagu lagu2 = {"Let It Be", "The Beatles", 4.0};  
  
L = alokasi(lagu2);  
  
InsertLast(P, L);  
  
  
Lagu lagu3 = {"Yesterday", "The Beatles", 2.5};  
  
L = alokasi(lagu3);  
  
InsertLast(P, L);  
  
  
Lagu lagu4 = {"Come Together", "The Beatles", 4.2};  
  
L = alokasi(lagu4);  
  
InsertLast(P, L);  
  
  
Lagu lagu5 = {"Something", "The Beatles", 3.0};  
  
L = alokasi(lagu5);  
  
InsertLast(P, L);  
  
  
printPlaylist(P);  
  
  
cout << "Menambahkan lagu setelah lagu ke-3..." << endl;  
  
Lagu lagu6 = {"Here Comes The Sun", "The Beatles", 3.1};  
  
L = alokasi(lagu6);
```

```

        InsertAfter3(P, L);

        printPlaylist(P);

        cout << "Menghapus lagu dengan judul 'Yesterday'..." << endl;

        deleteByJudul(P, "Yesterday");

        printPlaylist(P);

        system("pause");

        return 0;

}

```

## Screenshot Output

```

PS C:\StrukDat\Modul 4\Unguided1> cd "c:\StrukDat\Modul 4\Unguided1"; if ($?) { g++ main.cpp -o main }; if ($?) { .\main
}
Menambahkan lagu The Beatles ke playlist...

Daftar Lagu dalam Playlist:
- Judul: Hey Jude | Penyanyi: The Beatles | Durasi: 7 menit
- Judul: Let It Be | Penyanyi: The Beatles | Durasi: 4 menit
- Judul: Yesterday | Penyanyi: The Beatles | Durasi: 2.5 menit
- Judul: Come Together | Penyanyi: The Beatles | Durasi: 4.2 menit
- Judul: Something | Penyanyi: The Beatles | Durasi: 3 menit

Menambahkan lagu setelah lagu ke-3...

Daftar Lagu dalam Playlist:
- Judul: Hey Jude | Penyanyi: The Beatles | Durasi: 7 menit
- Judul: Let It Be | Penyanyi: The Beatles | Durasi: 4 menit
- Judul: Yesterday | Penyanyi: The Beatles | Durasi: 2.5 menit
- Judul: Here Comes The Sun | Penyanyi: The Beatles | Durasi: 3.1 menit
- Judul: Come Together | Penyanyi: The Beatles | Durasi: 4.2 menit
- Judul: Something | Penyanyi: The Beatles | Durasi: 3 menit

Menghapus lagu dengan judul 'Yesterday'...
Lagu "Yesterday" berhasil dihapus.

Daftar Lagu dalam Playlist:
- Judul: Hey Jude | Penyanyi: The Beatles | Durasi: 7 menit
- Judul: Let It Be | Penyanyi: The Beatles | Durasi: 4 menit
- Judul: Here Comes The Sun | Penyanyi: The Beatles | Durasi: 3.1 menit
- Judul: Come Together | Penyanyi: The Beatles | Durasi: 4.2 menit
- Judul: Something | Penyanyi: The Beatles | Durasi: 3 menit

Press any key to continue . . .

```

## Deskripsi

Program ini adalah contoh penerapan Single Linked List untuk mengelola playlist lagu “The Beatles”. Setiap lagu disimpan dalam satu node yang berisi judul, penyanyi, dan durasi. Program diawali dengan membuat playlist kosong, lalu menambahkan beberapa lagu menggunakan fungsi InsertLast. Setelah itu, program menambahkan satu lagu baru

setelah lagu ke-3 dengan fungsi InsertAfter3, dan juga menghapus lagu berdasarkan judul menggunakan deleteByJudul. Akhirnya, semua lagu dalam playlist ditampilkan ke layar melalui fungsi printPlaylist. Program ini menunjukkan bagaimana linked list digunakan untuk menyimpan, menambah, menghapus, dan menampilkan data secara dinamis.

#### D. Referensi

Juhana, A. (2020). *Algoritma dan Pemrograman Dasar Menggunakan Bahasa C++*. Rumah Publikasi Indonesia.

Munir, R. (2019). *Algoritma dan Struktur Data dalam Bahasa C dan C++ (Edisi Kedua)*. Informatika Bandung.

Sugiarto, A. (2021). *Struktur Data dan Algoritma dengan C++*. Penerbit Andi.

Wahana Komputer. (2018). *Pemrograman C++ untuk Pemula dan Profesional*. Elex Media Komputindo.

Nugroho, A. (2017). *Struktur Data dan Pemrograman Berorientasi Objek dengan C++*. Graha Ilmu.