LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VII

MULTI LINKED LIST

Disusun Oleh :

NAMA : Muhammad Fachri Auravyano Saka

NIM : 103112430180

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

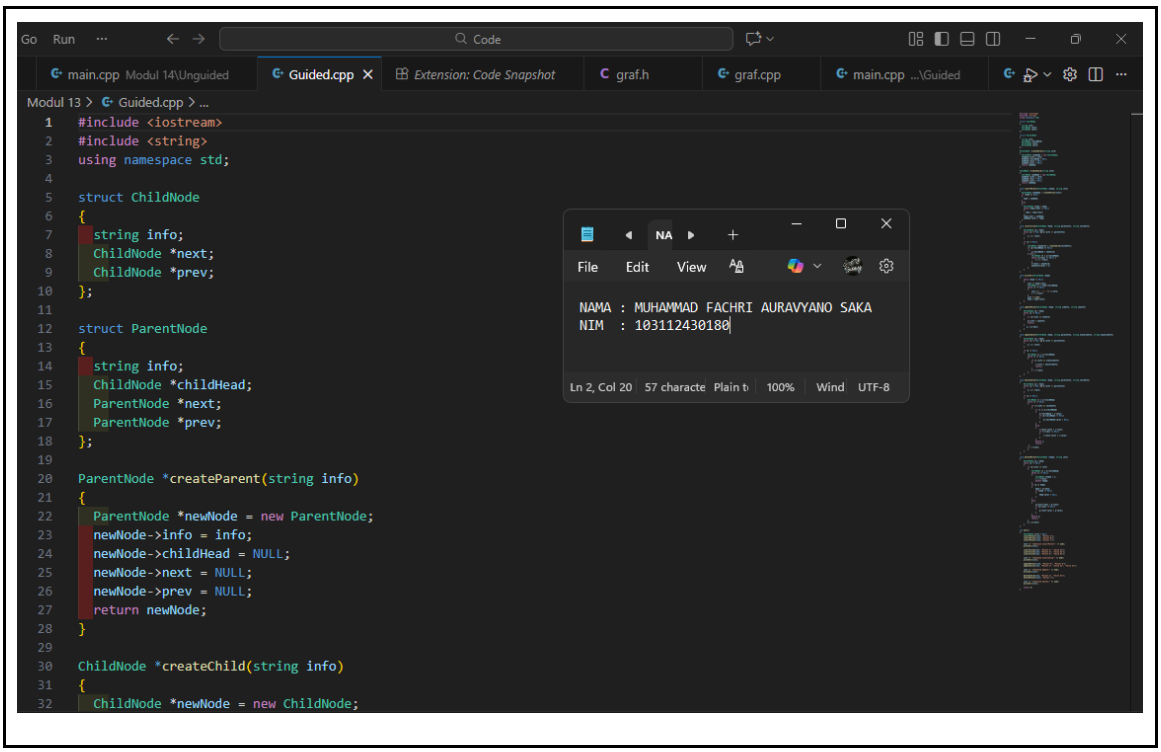
2025

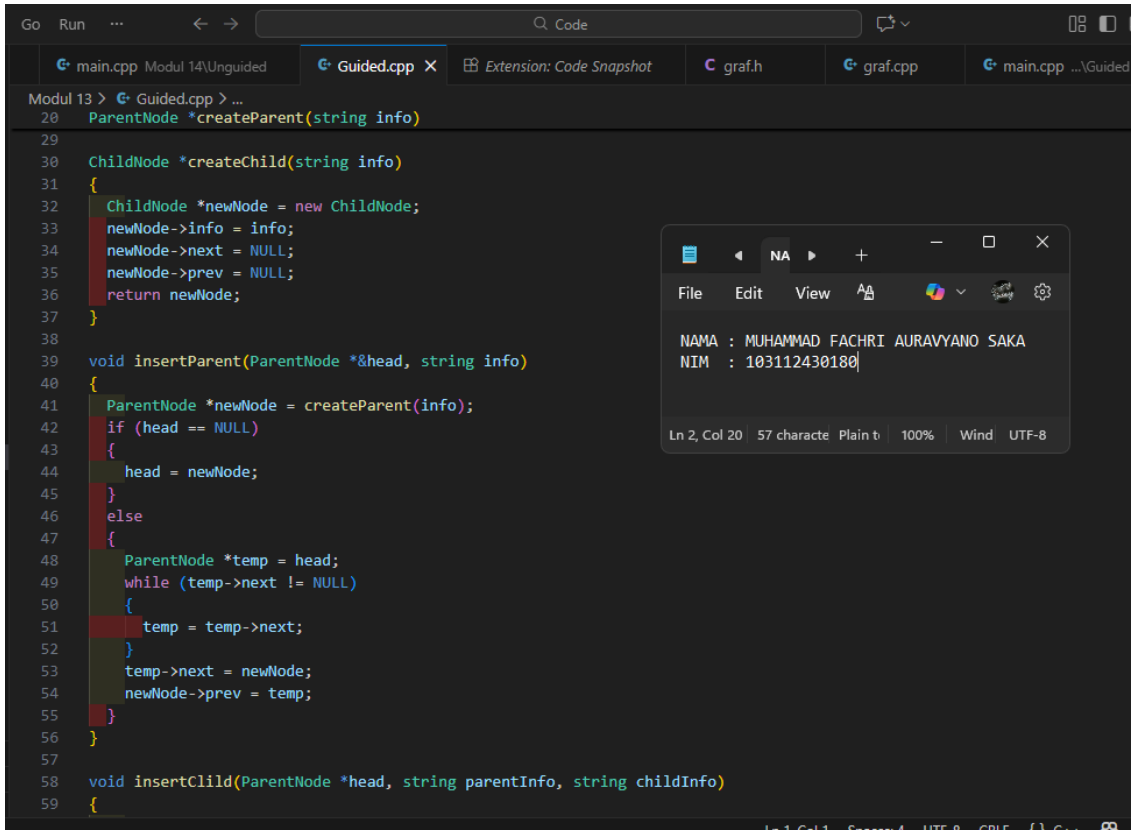
A. Dasar Teori

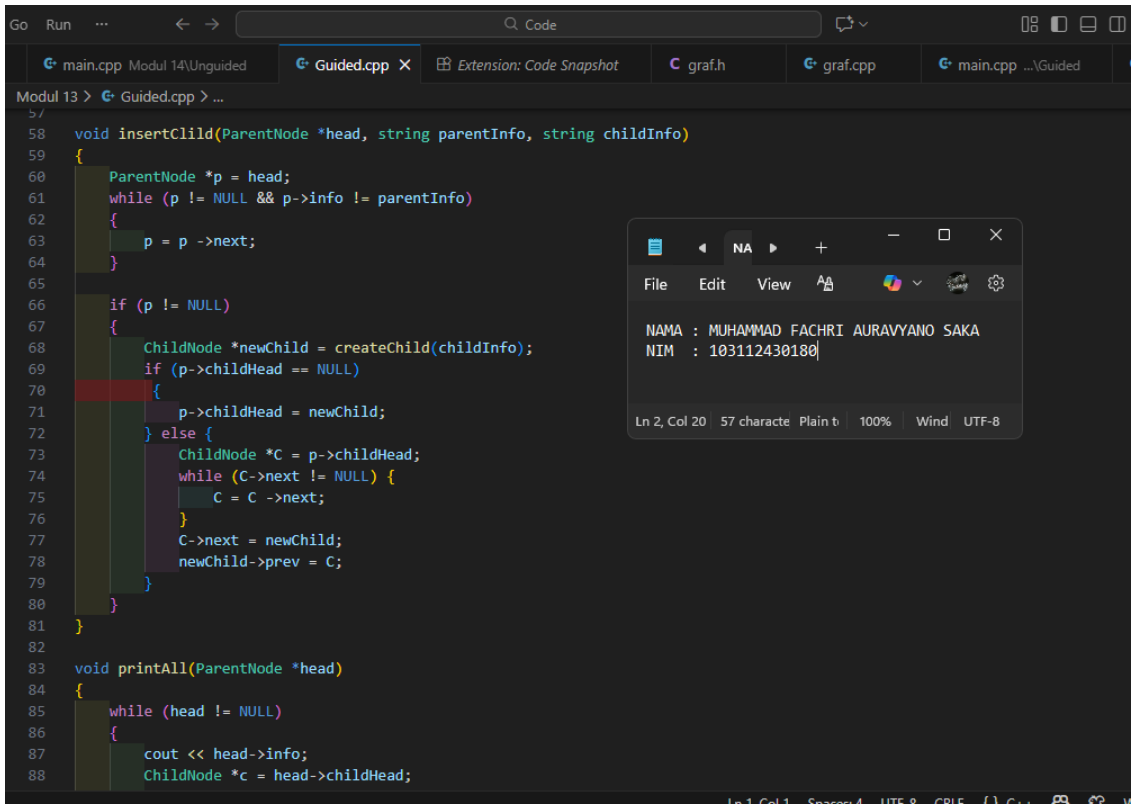
Linked List adalah struktur data dinamis yang terdiri dari sekumpulan node yang saling terhubung melalui pointer. Salah satu pengembangan dari struktur ini adalah Multi-Linked List, yang memungkinkan pengelolaan data yang memiliki relasi hierarkis atau satu-ke-banyak (one-to-many). Dalam implementasinya, struktur ini sering menggunakan Doubly Linked List di mana setiap node memiliki pointer next dan prev untuk navigasi dua arah. Keunikan dari Multi-Linked List terletak pada adanya pointer tambahan pada node induk (Parent) yang menunjuk ke alamat node pertama dari senarai anak (Child List). Hal ini menciptakan struktur seperti pohon (tree) atau graf yang memungkinkan pengelompokan data, di mana satu kategori (Parent) dapat menampung berbagai sub-item (Child) secara dinamis tanpa batasan ukuran array statis.

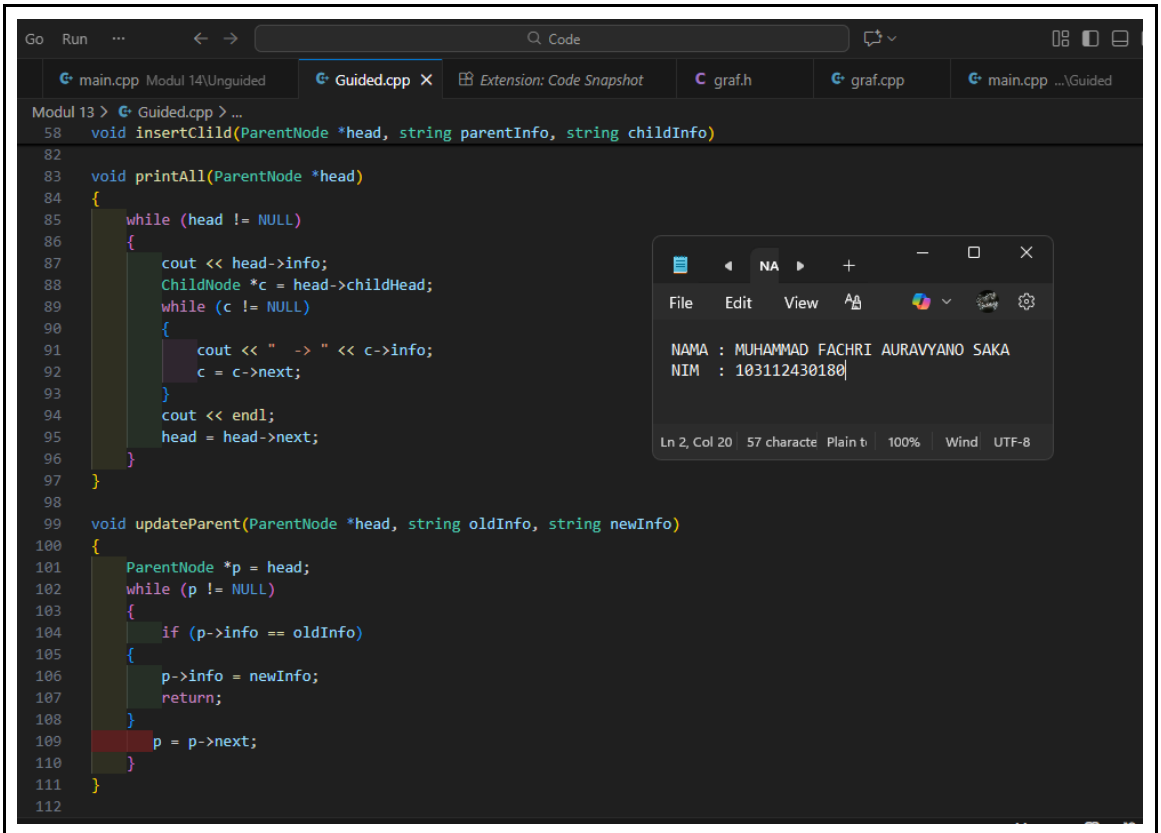
B. Guided (berisi screenshot source code & output program disertai penjelasannya)

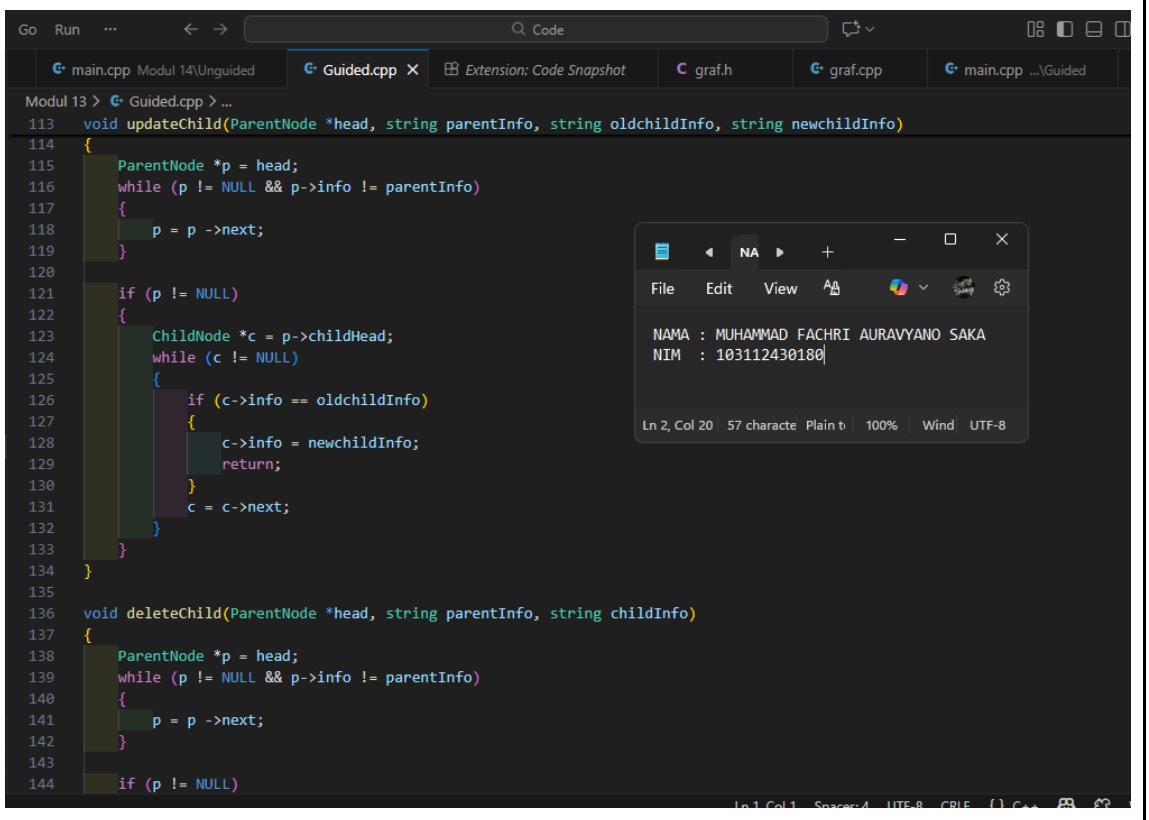
Guided 1

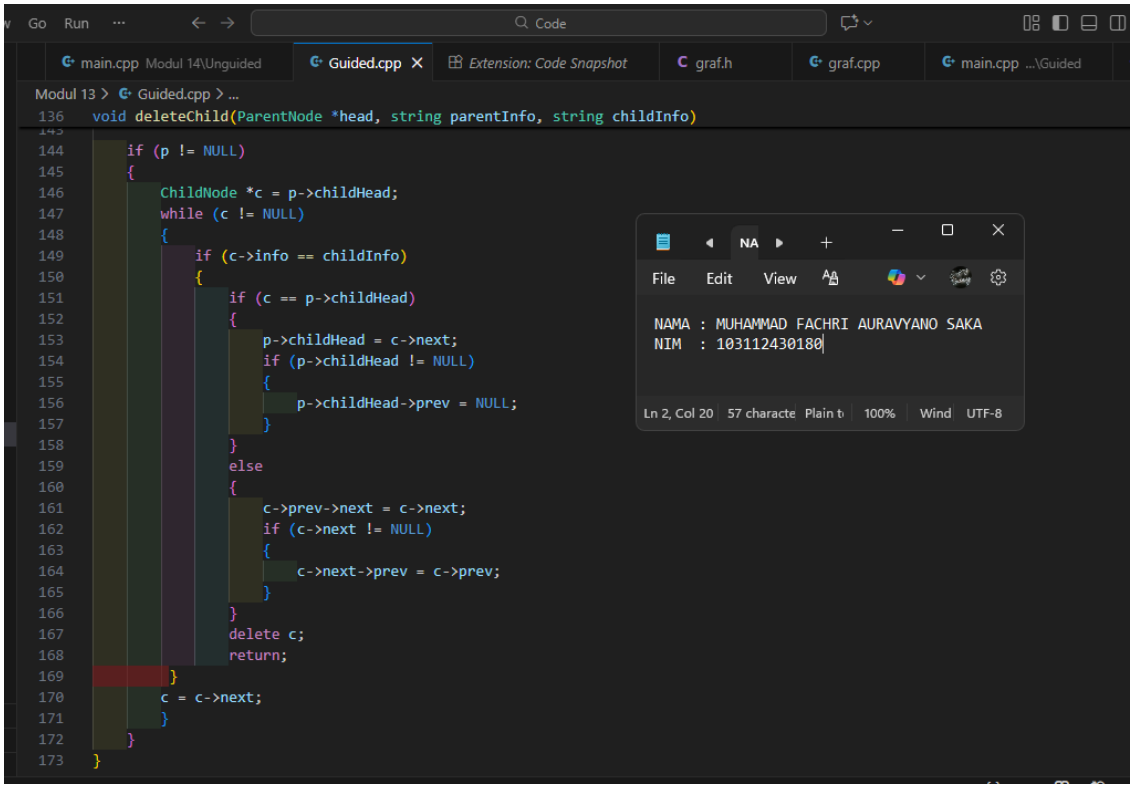


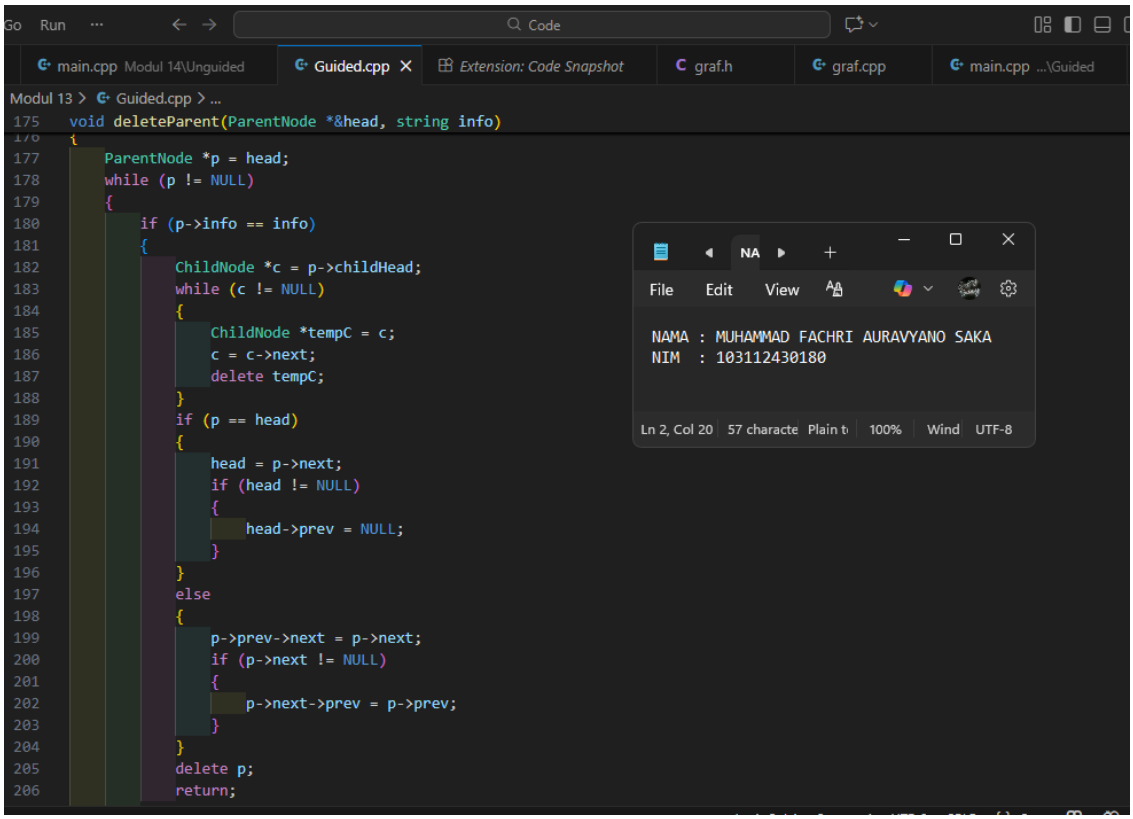

Go Run
Code
main.cpp Modul 14\Unguided
Guided.cpp
Extension: Code Snapshot
graf.h
graf.cpp
main.cpp ...\Guided
Modul 13 > Guided.cpp > ...
ParentNode *createParent(string info)
ChildNode *createChild(string info)
{
ChildNode *newNode = new ChildNode;
newNode->info = info;
newNode->next = NULL;
newNode->prev = NULL;
return newNode;
}
void insertParent(ParentNode *&head, string info)
{
ParentNode *newNode = createParent(info);
if (head == NULL)
{
head = newNode;
}
else
{
ParentNode *temp = head;
while (temp->next != NULL)
{
temp = temp->next;
}
temp->next = newNode;
newNode->prev = temp;
}
}
void insertClild(ParentNode *head, string parentInfo, string childInfo)
{
NA
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8

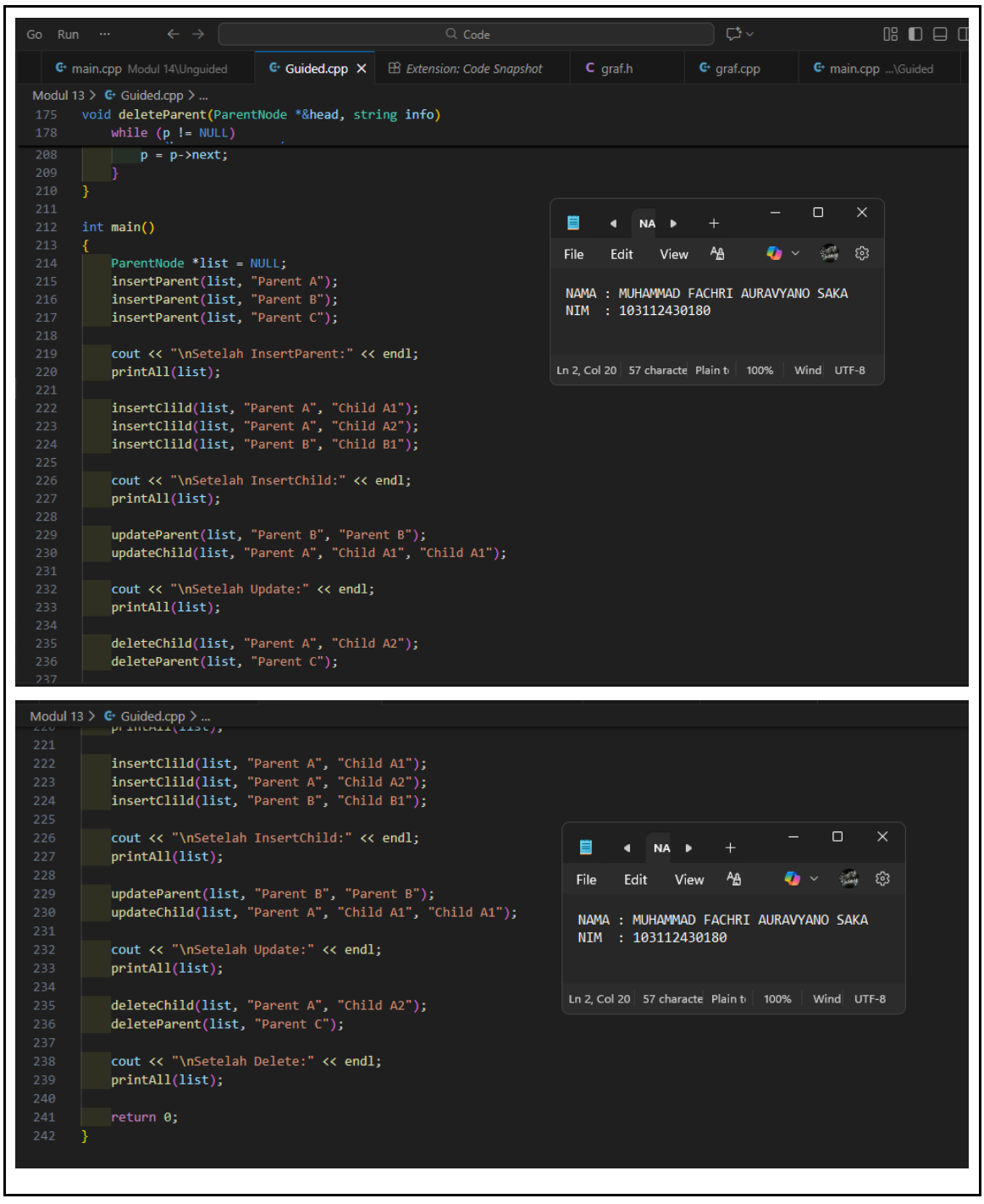

Go Run
Code
main.cpp Modul 14\Unguided
Guided.cpp
Extension: Code Snapshot
graf.h
graf.cpp
main.cpp ...\Guided
Modul 13 > Guided.cpp > ...
void insertClild(ParentNode *head, string parentInfo, string childInfo)
{
ParentNode *p = head;
while (p != NULL && p->info != parentInfo)
{
p = p ->next;
}
if (p != NULL)
{
ChildNode *newChild = createChild(childInfo);
if (p->childHead == NULL)
{
p->childHead = newChild;
} else {
ChildNode *C = p->childHead;
while (C->next != NULL) {
C = C ->next;
}
C->next = newChild;
newChild->prev = C;
}
}
}
void printAll(ParentNode *head)
{
while (head != NULL)
{
cout << head->info;
ChildNode *c = head->childHead;
NA
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8


Go Run
Code
main.cpp Modul 14\Unguided
Guided.cpp
Extension: Code Snapshot
graf.h
graf.cpp
main.cpp ...\Guided
Modul 13 > Guided.cpp > ...
58 void insertClild(ParentNode *head, string parentInfo, string childInfo)
82
83 void printAll(ParentNode *head)
84 {
85 while (head != NULL)
86 {
87 cout << head->info;
88 ChildNode *c = head->childHead;
89 while (c != NULL)
90 {
91 cout << " -> " << c->info;
92 c = c->next;
93 }
94 cout << endl;
95 head = head->next;
96 }
97 }
98
99 void updateParent(ParentNode *head, string oldInfo, string newInfo)
100 {
101 ParentNode *p = head;
102 while (p != NULL)
103 {
104 if (p->info == oldInfo)
105 {
106 p->info = newInfo;
107 return;
108 }
109 p = p->next;
110 }
111 }
112
NA
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8

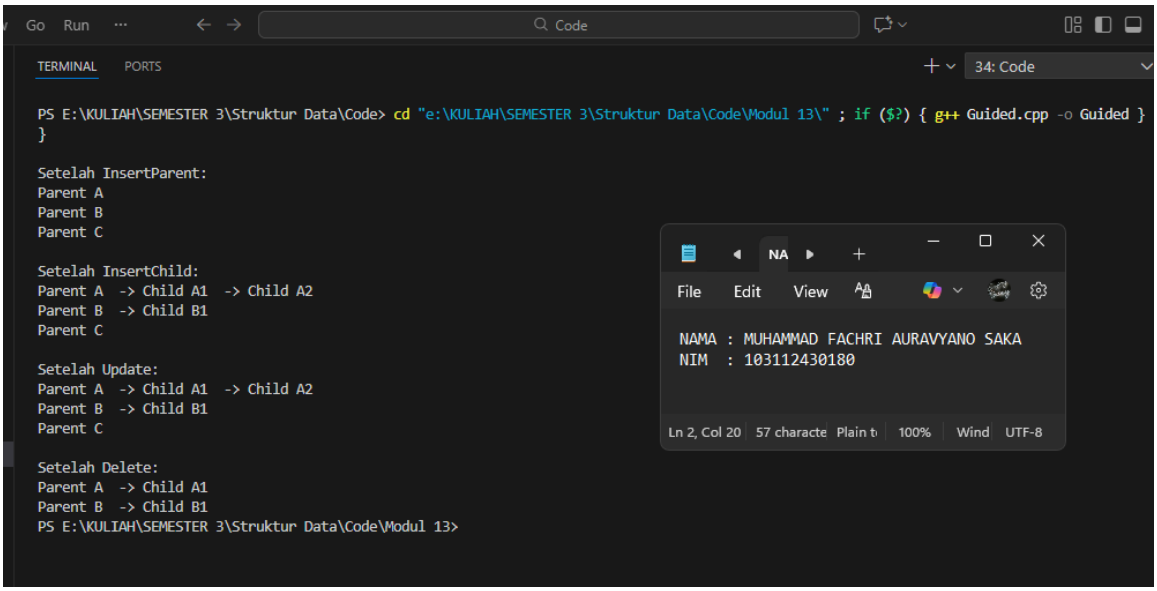

Go Run
Code
main.cpp Modul 14\Unguided
Guided.cpp
Extension: Code Snapshot
graf.h
graf.cpp
main.cpp ...\Guided
Modul 13 > Guided.cpp > ...
113 void updateChild(ParentNode *head, string parentInfo, string oldchildInfo, string newchildInfo)
114 {
115 ParentNode *p = head;
116 while (p != NULL && p->info != parentInfo)
117 {
118 p = p ->next;
119 }
120
121 if (p != NULL)
122 {
123 ChildNode *c = p->childHead;
124 while (c != NULL)
125 {
126 if (c->info == oldchildInfo)
127 {
128 c->info = newchildInfo;
129 return;
130 }
131 c = c->next;
132 }
133 }
134 }
135
136 void deleteChild(ParentNode *head, string parentInfo, string childInfo)
137 {
138 ParentNode *p = head;
139 while (p != NULL && p->info != parentInfo)
140 {
141 p = p ->next;
142 }
143
144 if (p != NULL)
NA
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8


Go Run
Code
main.cpp Modul 14\Unguided
Guided.cpp
Extension: Code Snapshot
graf.h
graf.cpp
main.cpp ...\Guided
Modul 13 > Guided.cpp > ...
136 void deleteChild(ParentNode *head, string parentInfo, string childInfo)
144 if (p != NULL)
145 {
146 ChildNode *c = p->childHead;
147 while (c != NULL)
148 {
149 if (c->info == childInfo)
150 {
151 if (c == p->childHead)
152 {
153 p->childHead = c->next;
154 if (p->childHead != NULL)
155 {
156 p->childHead->prev = NULL;
157 }
158 }
159 else
160 {
161 c->prev->next = c->next;
162 if (c->next != NULL)
163 {
164 c->next->prev = c->prev;
165 }
166 }
167 delete c;
168 return;
169 }
170 c = c->next;
171 }
172 }
173 }
NA
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8


Go Run
Code
main.cpp Modul 14\Unguided
Guided.cpp
Extension: Code Snapshot
graf.h
graf.cpp
main.cpp ...\Guided
Modul 13 > Guided.cpp > ...
175 void deleteParent(ParentNode *&head, string info)
177 ParentNode *p = head;
178 while (p != NULL)
179 {
180 if (p->info == info)
181 {
182 ChildNode *c = p->childHead;
183 while (c != NULL)
184 {
185 ChildNode *tempC = c;
186 c = c->next;
187 delete tempC;
188 }
189 if (p == head)
190 {
191 head = p->next;
192 if (head != NULL)
193 {
194 head->prev = NULL;
195 }
196 }
197 else
198 {
199 p->prev->next = p->next;
200 if (p->next != NULL)
201 {
202 p->next->prev = p->prev;
203 }
204 }
205 delete p;
206 return;
NA
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8



Screenshots Output



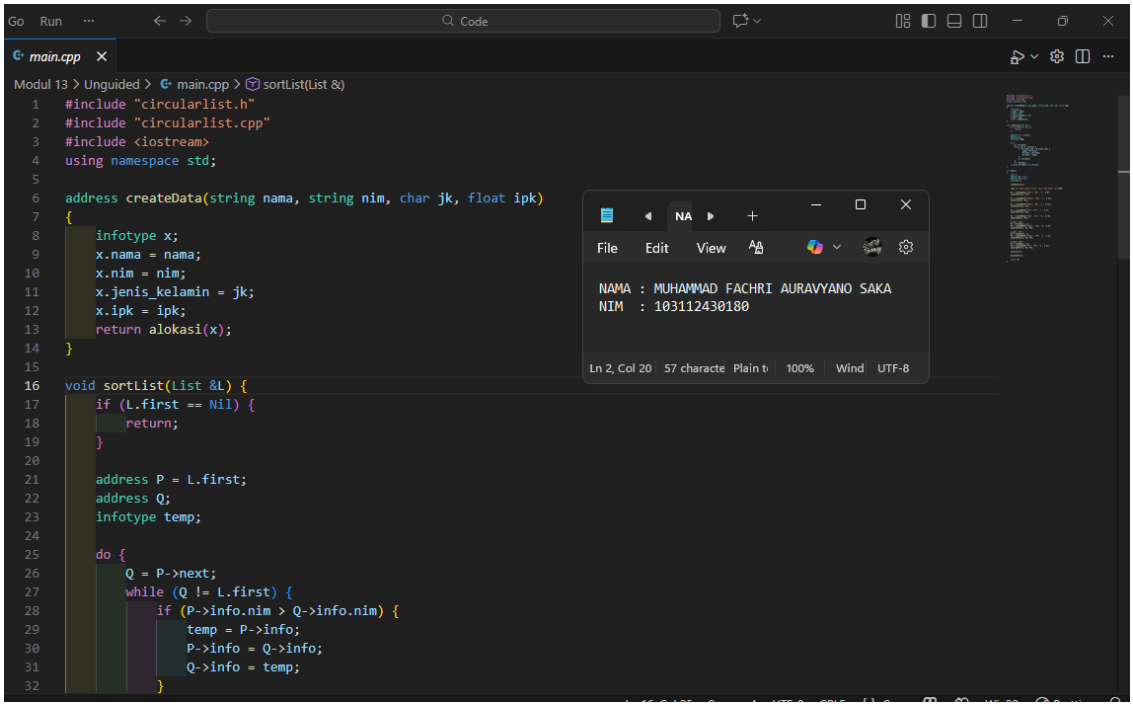
Deskripsi:

Program ini adalah implementasi sederhana dari struktur data Multi-Linked List (atau List of Lists) menggunakan C++, di mana terdapat dua jenis node yaitu ParentNode (induk) dan ChildNode (anak). Setiap node induk terhubung satu sama lain dalam doubly linked list, dan uniknya, setiap node induk memiliki pointernya sendiri yang menunjuk ke daftar anak yang juga berbentuk doubly linked list. Program ini mencakup operasi dasar (CRUD) yang memungkinkan pengguna untuk menambahkan parent dan child, menampilkan seluruh hierarki data, memperbarui informasi pada node tertentu, serta menghapus node parent atau child dengan menangani perubahan pointer agar rantai data tidak putus. Pada fungsi main, kode mendemonstrasikan alur lengkap mulai dari penyisipan data, pembaruan, hingga penghapusan untuk menunjukkan bagaimana relasi antar node dikelola dalam memori.

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

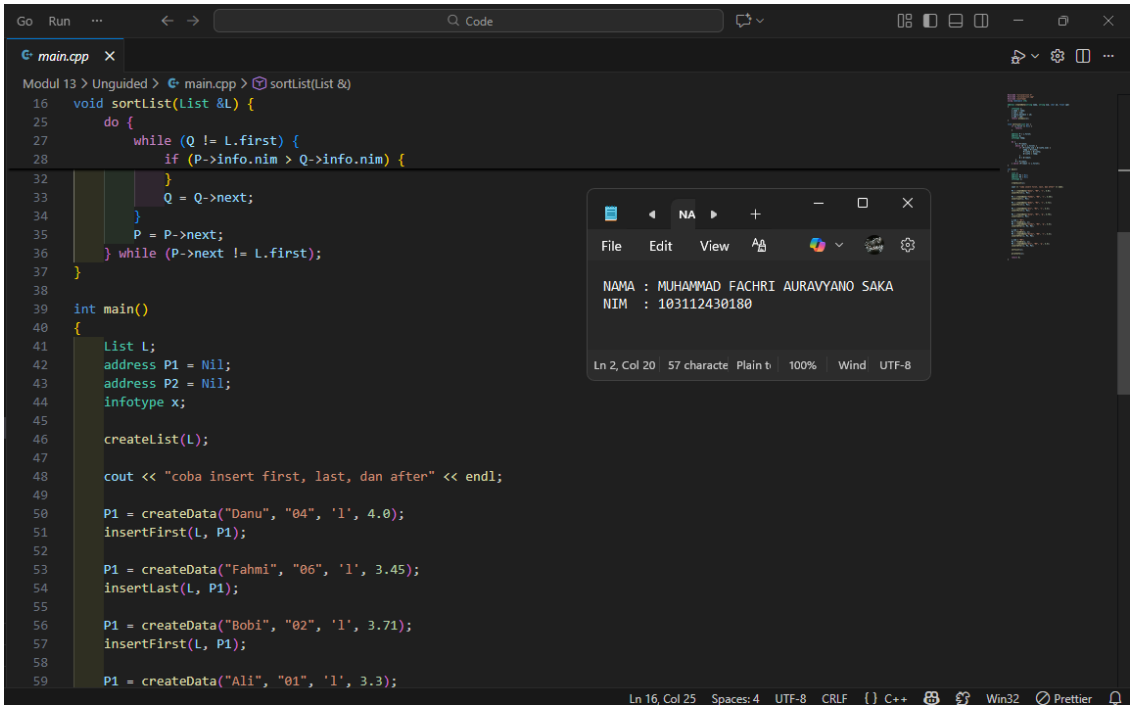
Unguided 1

Main.cpp



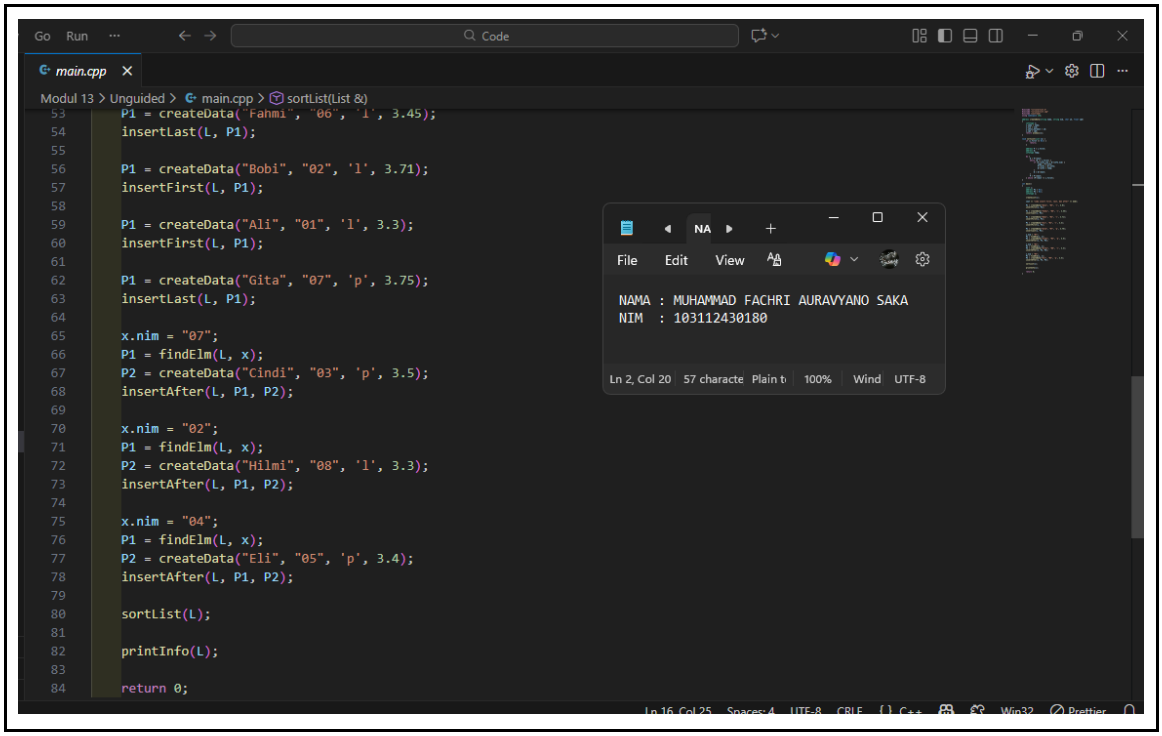
```
Modul 13 > Unguided > main.cpp > sortList(List &)
#include "circularlist.h"
#include "circularlist.cpp"
#include <iostream>
using namespace std;

address createData(string nama, string nim, char jk, float ipk)
{
    infotype x;
    x.nama = nama;
    x.nim = nim;
    x.jenis_kelamin = jk;
    x.ipk = ipk;
    return alokasi(x);
}

void sortList(List &L) {
    if (L.first == Nil) {
        return;
    }

    address P = L.first;
    address Q;
    infotype temp;

    do {
        Q = P->next;
        while (Q != L.first) {
            if (P->info.nim > Q->info.nim) {
                temp = P->info;
                P->info = Q->info;
                Q->info = temp;
            }
```

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM  : 103112430180
```

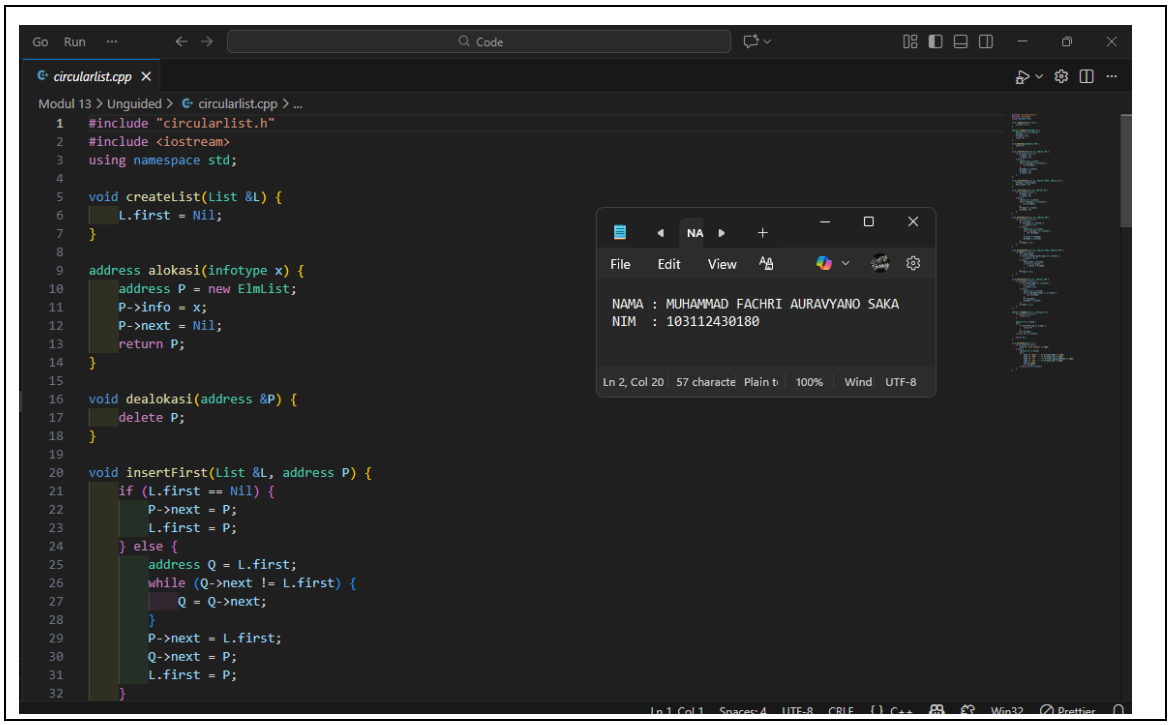


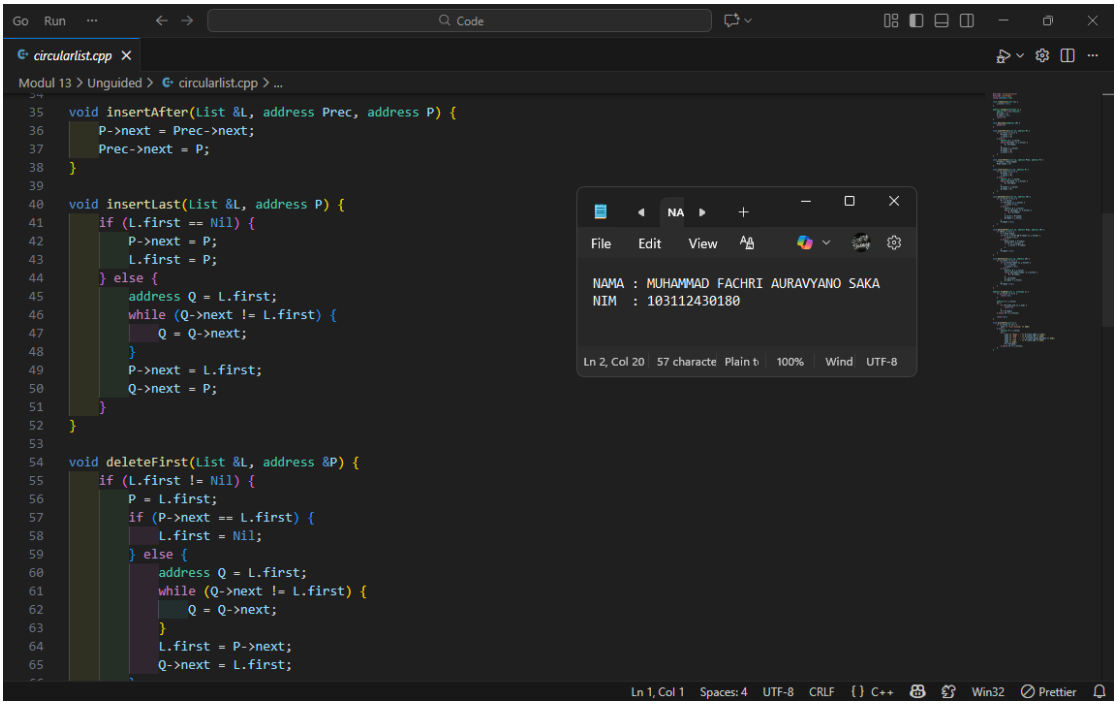
```
Modul 13 > Unguided > main.cpp > sortList(List &)
void sortList(List &L) {
    do {
        while (Q != L.first) {
            if (P->info.nim > Q->info.nim) {
            }
            Q = Q->next;
        }
        P = P->next;
    } while (P->next != L.first);
}

int main()
{
    List L;
    address P1 = Nil;
    address P2 = Nil;
    infotype x;

    createList(L);

    cout << "coba insert first, last, dan after" << endl;

    P1 = createData("Danu", "04", 'l', 4.0);
    insertFirst(L, P1);

    P1 = createData("Fahmi", "06", 'l', 3.45);
    insertLast(L, P1);

    P1 = createData("Bobi", "02", 'l', 3.71);
    insertFirst(L, P1);

    P1 = createData("Ali", "01", 'l', 3.3);
```

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM  : 103112430180
```



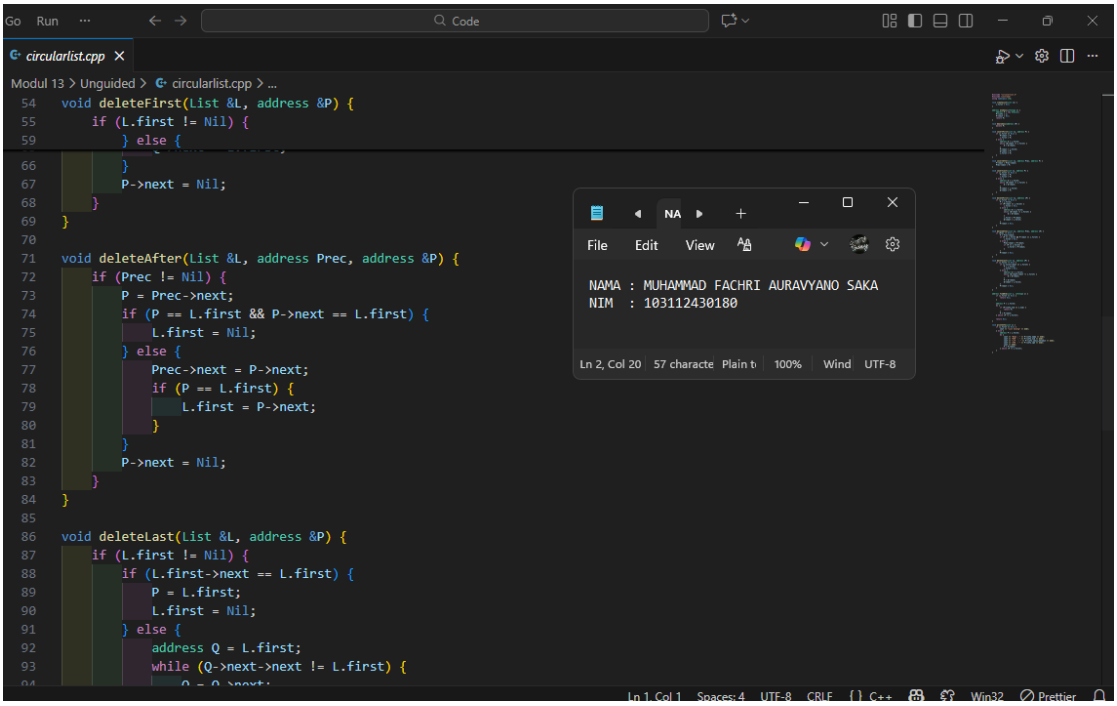
circularlist.cpp





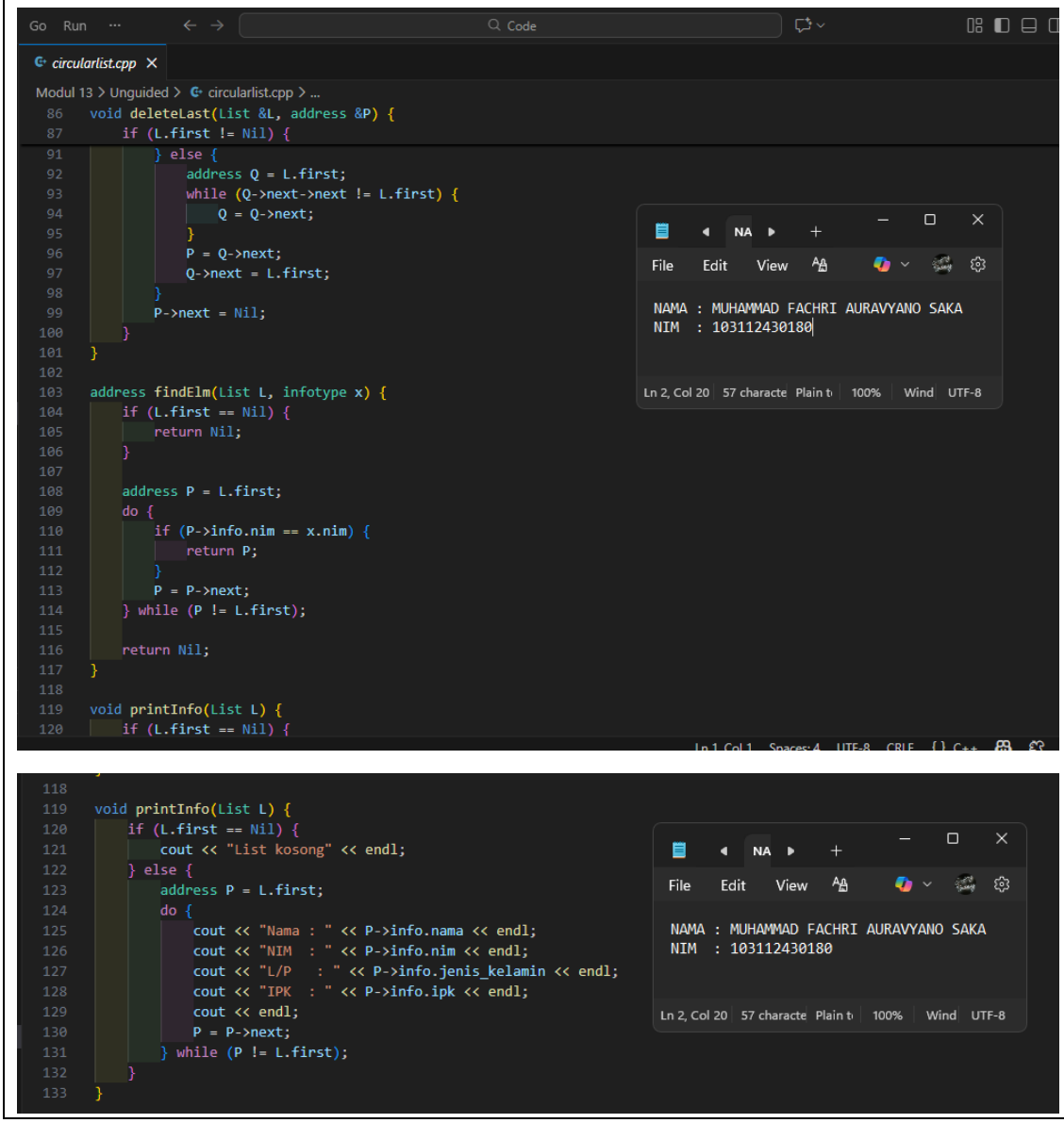
```cpp
void insertAfter(List &L, address Prec, address P) {
    P->next = Prec->next;
    Prec->next = P;
}

void insertLast(List &L, address P) {
    if (L.first == Nil) {
        P->next = P;
        L.first = P;
    } else {
        address Q = L.first;
        while (Q->next != L.first) {
            Q = Q->next;
        }
        P->next = L.first;
        Q->next = P;
    }
}

void deleteFirst(List &L, address &P) {
    if (L.first != Nil) {
        P = L.first;
        if (P->next == L.first) {
            L.first = Nil;
        } else {
            address Q = L.first;
            while (Q->next != L.first) {
                Q = Q->next;
            }
            L.first = P->next;
            Q->next = L.first;
```

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM  : 103112430180
```

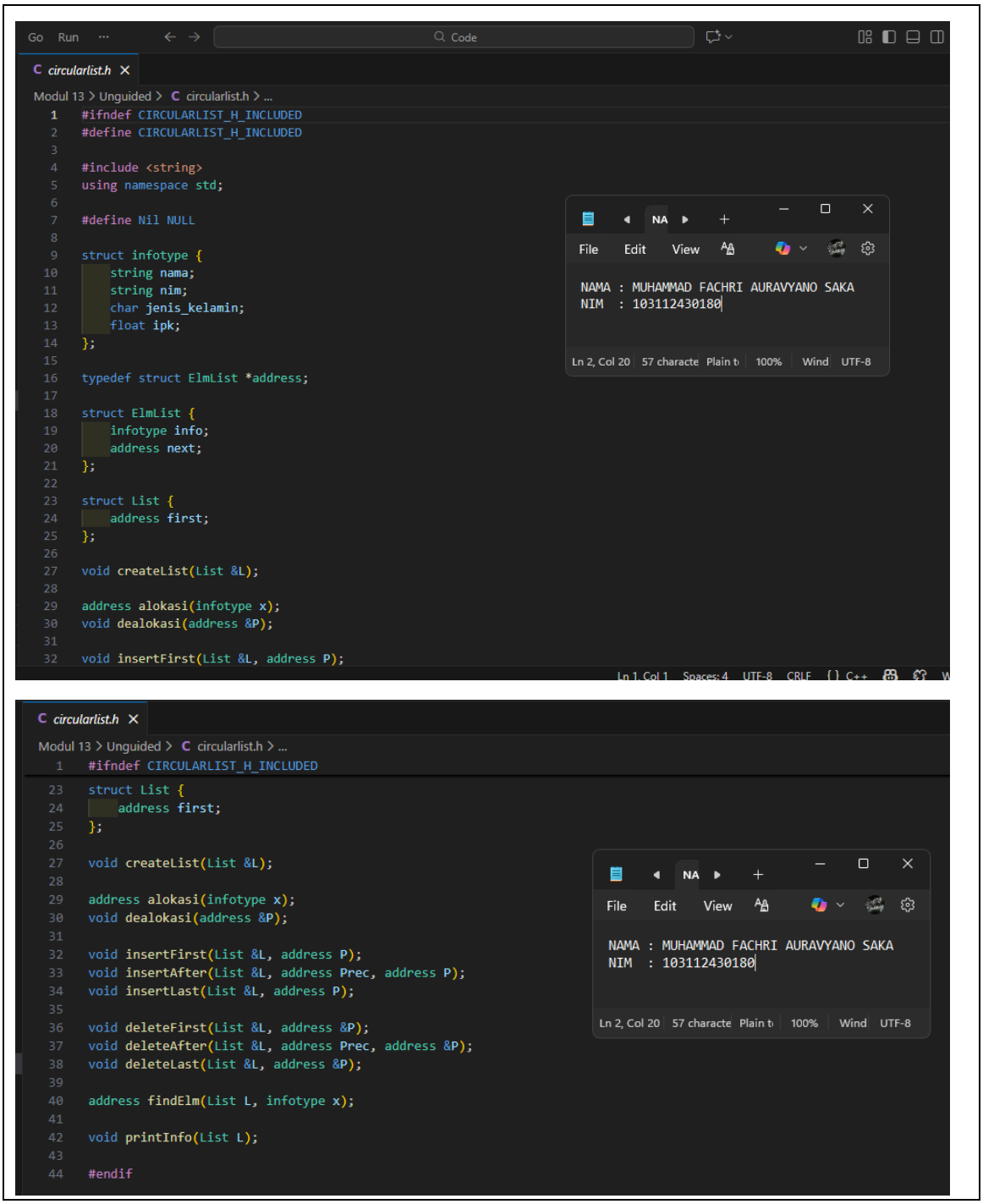


```cpp
void deleteFirst(List &L, address &P) {
    if (L.first != Nil) {
        } else {
        }
        P->next = Nil;
    }
}

void deleteAfter(List &L, address Prec, address &P) {
    if (Prec != Nil) {
        P = Prec->next;
        if (P == L.first && P->next == L.first) {
            L.first = Nil;
        } else {
            Prec->next = P->next;
            if (P == L.first) {
                L.first = P->next;
            }
        }
        P->next = Nil;
    }
}

void deleteLast(List &L, address &P) {
    if (L.first != Nil) {
        if (L.first->next == L.first) {
            P = L.first;
            L.first = Nil;
        } else {
            address Q = L.first;
            while (Q->next->next != L.first) {
```

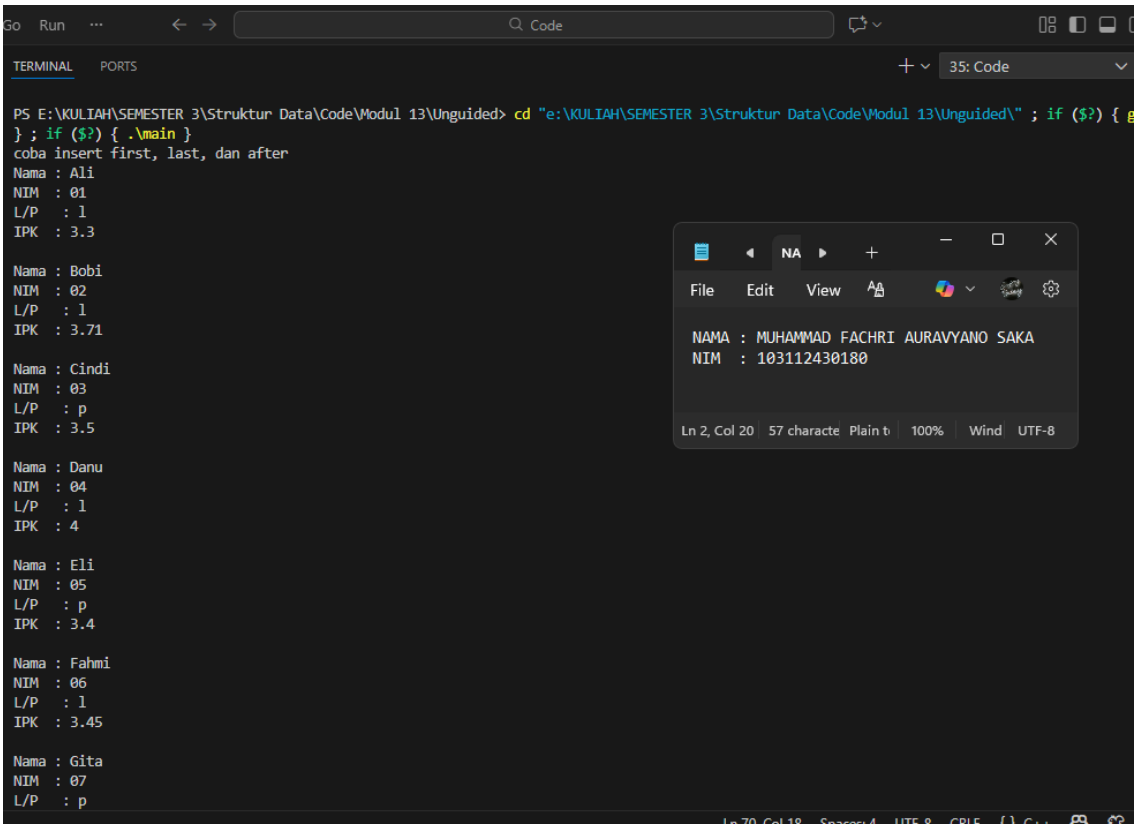
```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM  : 103112430180
```

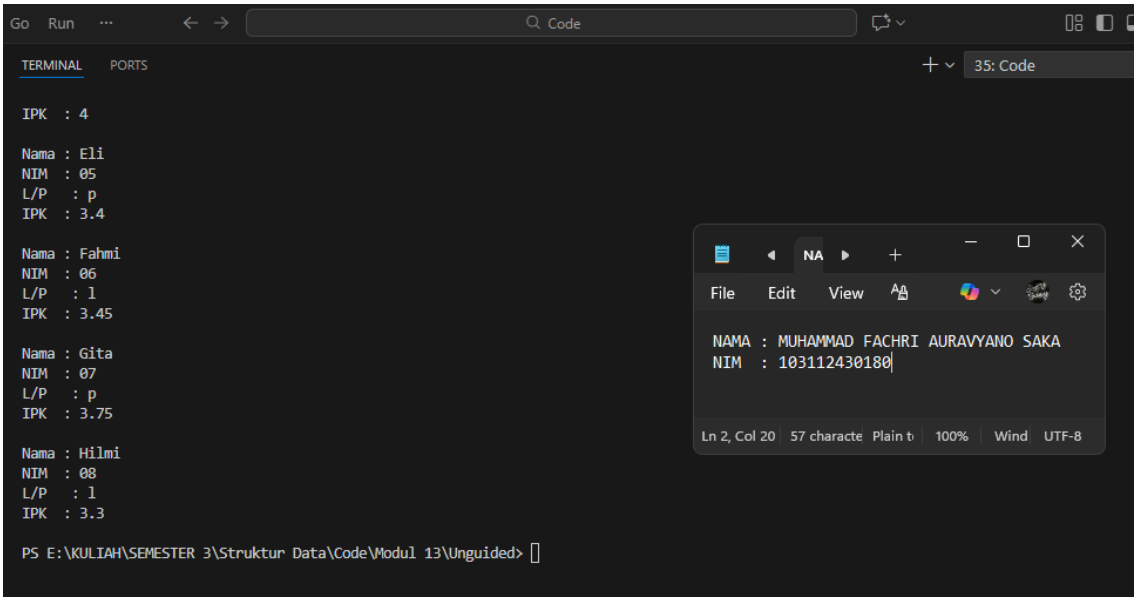


circularlist.h



Screenshot Output





Deskripsi:

Program ini adalah implementasi struktur data Circular Single Linked List menggunakan C++ yang dirancang untuk mengelola data mahasiswa (mencakup Nama, NIM, Jenis Kelamin, dan IPK). Berbeda dengan linked list biasa, node terakhir dalam struktur ini memiliki pointer next yang menunjuk kembali ke node pertama (first), sehingga membentuk siklus tertutup tanpa akhir NULL. Kode ini dibagi menjadi file header, implementasi, dan main untuk modularitas, menyediakan operasi dasar seperti alokasi memori, penyisipan node (di awal, akhir, atau setelah node tertentu), penghapusan, serta pencarian data. Pada fungsi main, program mendemonstrasikan penginputan berbagai

data mahasiswa secara acak yang kemudian diurutkan (sorting) berdasarkan NIM menggunakan algoritma sederhana sebelum seluruh daftar ditampilkan ke layar.

D. Kesimpulan

Berdasarkan praktikum yang telah dilakukan, dapat disimpulkan bahwa penggunaan Linked List memberikan fleksibilitas tinggi dalam manajemen memori dibandingkan struktur data statis. Implementasi Multi-Linked List terbukti efektif untuk merepresentasikan data yang bersifat hierarkis dan kompleks, di mana relasi antar entitas (Parent dan Child) dapat dikelola secara terstruktur menggunakan pointer ganda. Sementara itu, Circular Linked List menunjukkan keunggulan dalam menangani data yang membutuhkan siklus kontinu tanpa nilai null di akhir. Kedua implementasi ini menegaskan pentingnya pemahaman mendalam mengenai manipulasi pointer, alokasi memori dinamis, serta logika algoritma untuk menjaga integritas data saat dilakukan operasi penyisipan, penghapusan, maupun pengurutan data.

E. Referensi

Muñoz, D. F. (2024, June). A C++ library for fast simulation of queues and some experimental results. In *AIP Conference Proceedings* (Vol. 3094, No. 1, p. 110002). AIP Publishing LLC.

Goponenko, A., & Carroll, S. (2019). A C++ implementation of a lock-free priority queue based on Multi-Dimensional Linked List. *Link: https://www. researchgate. net/publication/337020321_A_C_Implementation_of_a_Lock-Free _Priority_Queue_Based_on_Multi-Dimensional_Linked_List*.

Malik, D. S. (2010). *Data structures using C++*. USA.