LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 11
MULTI LINKED LIST

Disusun Oleh :

NAMA : Muhammad Azzam Satria

NIM : 103112400112

Dosen

FAHRUDIN MUKTI WIBOWO S. Kom., M. Eng

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

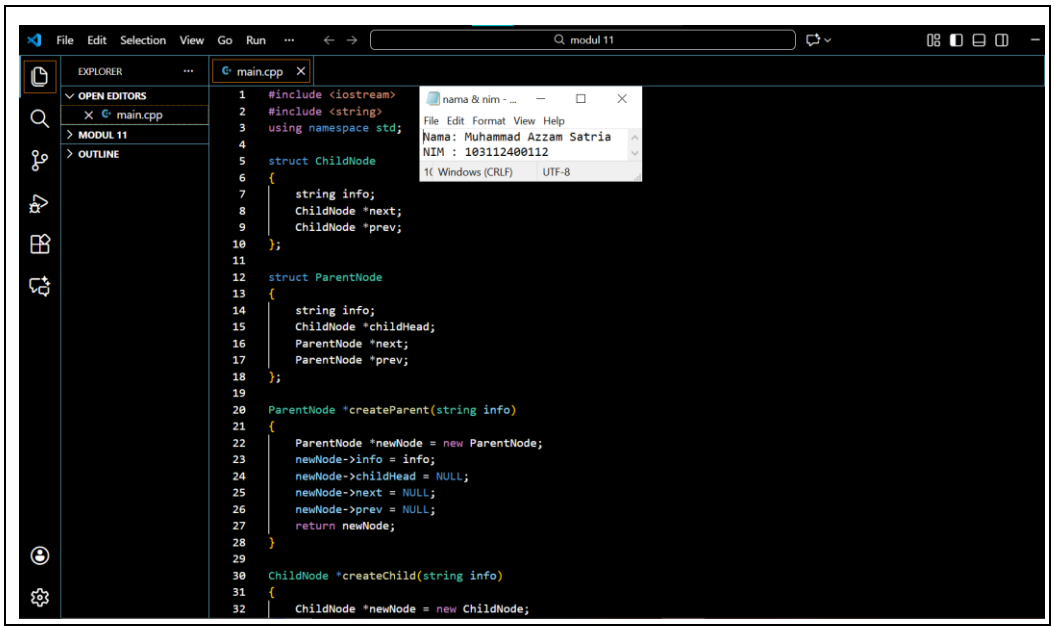
A. Dasar Teori

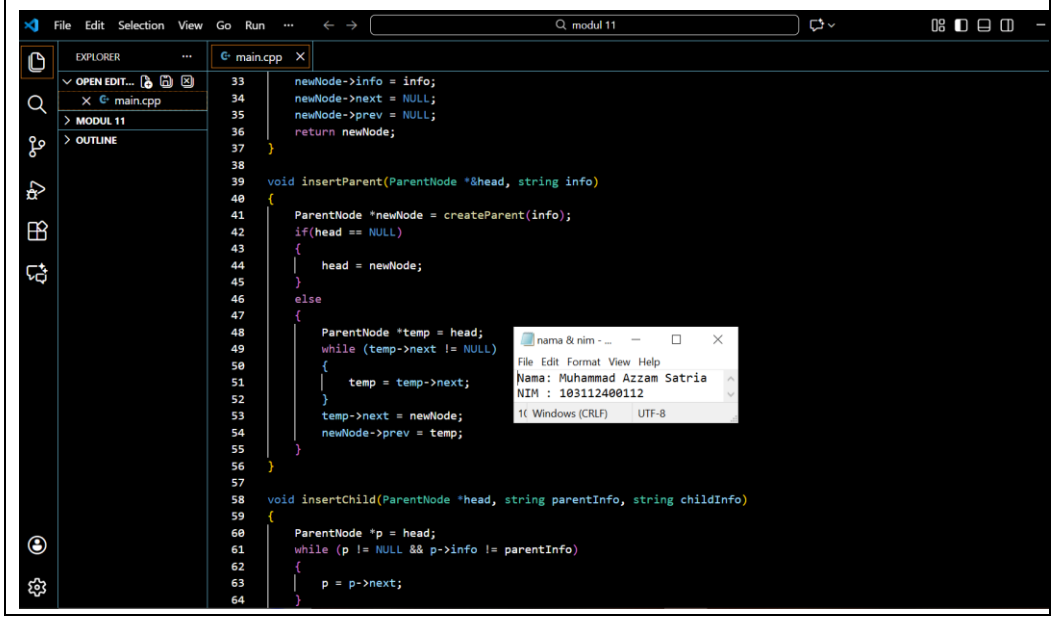
Struktur data multi linked list adalah pengembangan dari linked list biasa yang digunakan untuk menyimpan data bertingkat antara data induk dan data anak. Setiap node induk mempunyai pointer tambahan yang mengarah pada list anak sehingga satu elemen dapat terhubung dengan banyak elemen lain dibawahnya. Struktur data ini membuat pemodelan data menjadi lebih fleksibel ketika satu kategori memiliki banyak subkategori. Multi linked list mempunyai proses insert dan delete yang dilakukan di dua level yang berbeda sehingga pengelolaan pointer harus diperhatikan secara teliti. Selain itu, struktur data ini memiliki keunggulan yaitu mepermudah penyimpanan data hierarki yang tidak mempunyai ukuran tetap dan berubah kapanpun.

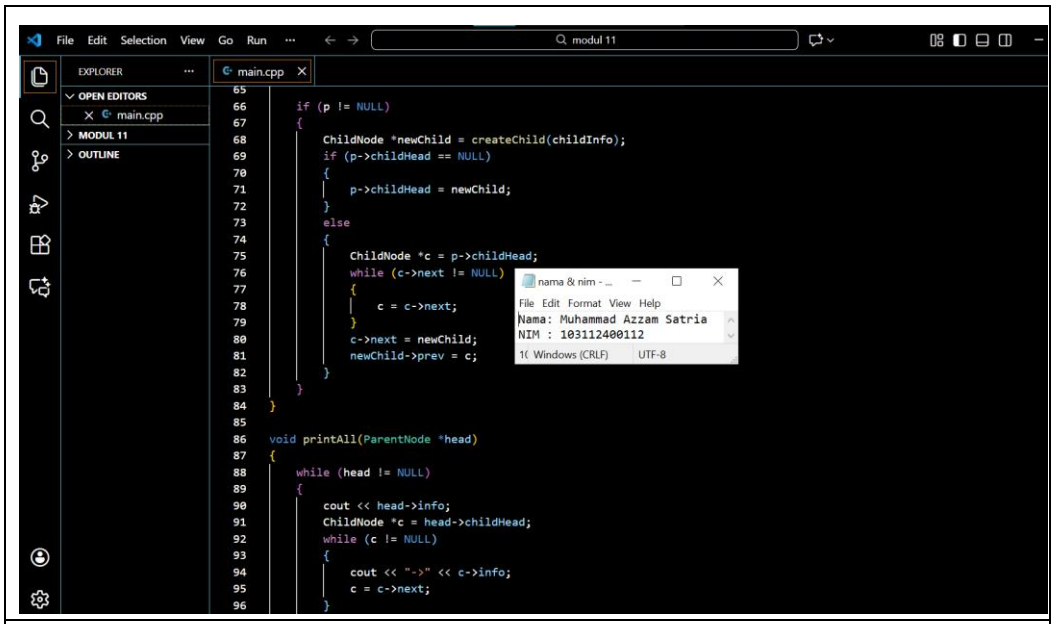
Proses traversal pada multi linked list dilakukan dengan menelusuri node induk, lalu dilanjut dengan menelusuri seluruh node anak yang ada di bawah induk tersebut. Pengaturan alokasi dan dealokasi juga dilakukan secara terpisah antara induk dan anak agar memastikan tidak ada memori yang tertinggal. Struktur data ini sangat tepat digunakan pada suatu kasus yang membutuhkan hubungan satu kategori ke banyak subkategori, seperti halnya dafta kelas dengan mahasiswa. Dengan sifatnya yang dinamis dan bertingkat, multi linked list memberikan gambaran detail pada suatu hubungan data yang saling berkaitan.

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

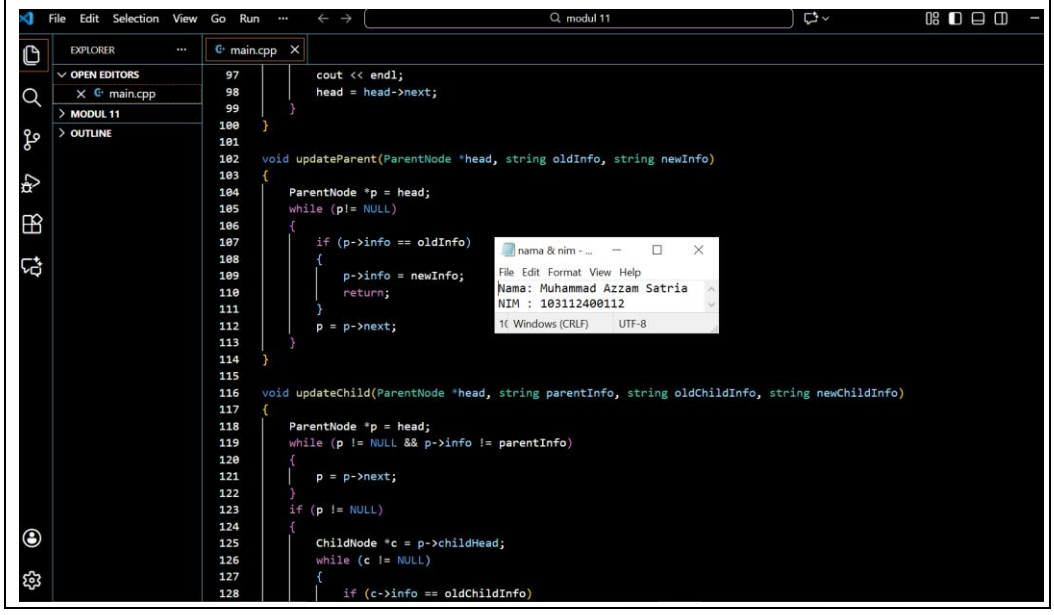
Guided 1


File Edit Selection View Go Run
modul 11
EXPLORER
main.cpp
OPEN EDITORS
main.cpp
MODUL 11
OUTLINE
nama & nim - ...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Ln 1, Col 1 Windows (CRLF) UTF-8
#include <iostream>
#include <string>
using namespace std;
struct ChildNode
{
string info;
ChildNode *next;
ChildNode *prev;
};
struct ParentNode
{
string info;
ChildNode *childHead;
ParentNode *next;
ParentNode *prev;
};
ParentNode *createParent(string info)
{
ParentNode *newNode = new ParentNode;
newNode->info = info;
newNode->childHead = NULL;
newNode->next = NULL;
newNode->prev = NULL;
return newNode;
}
ChildNode *createChild(string info)
{
ChildNode *newNode = new ChildNode;

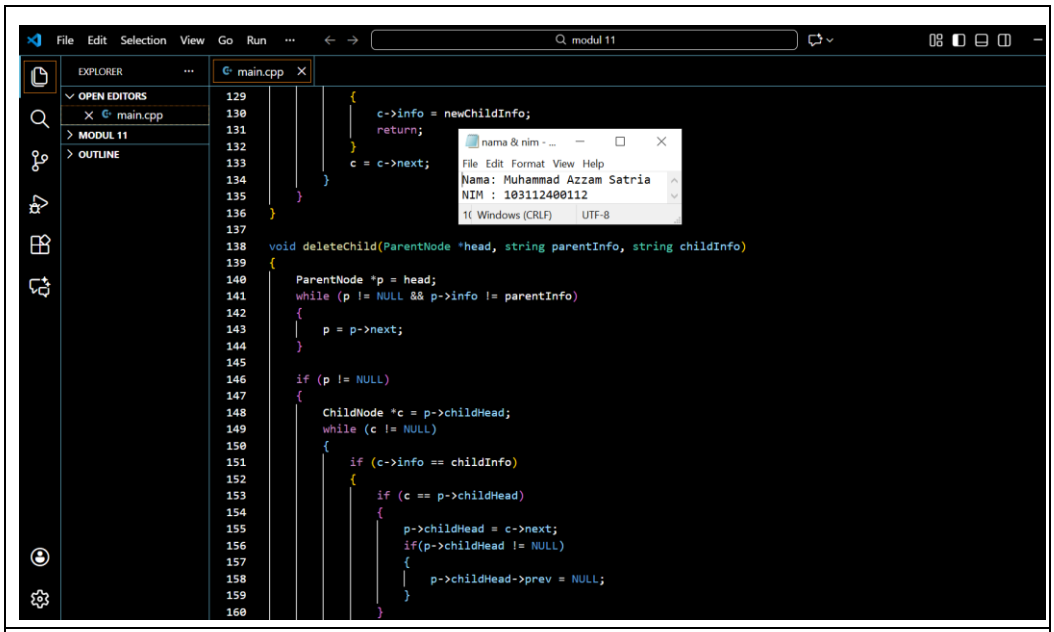

File Edit Selection View Go Run
modul 11
EXPLORER
main.cpp
OPEN EDIT...
main.cpp
MODUL 11
OUTLINE
nama & nim - ...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Ln 1, Col 1 Windows (CRLF) UTF-8
newNode->info = info;
newNode->next = NULL;
newNode->prev = NULL;
return newNode;
}
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 insertChild(ParentNode *head, string parentInfo, string childInfo)
{
ParentNode *p = head;
while (p != NULL && p->info != parentInfo)
{
p = p->next;
}

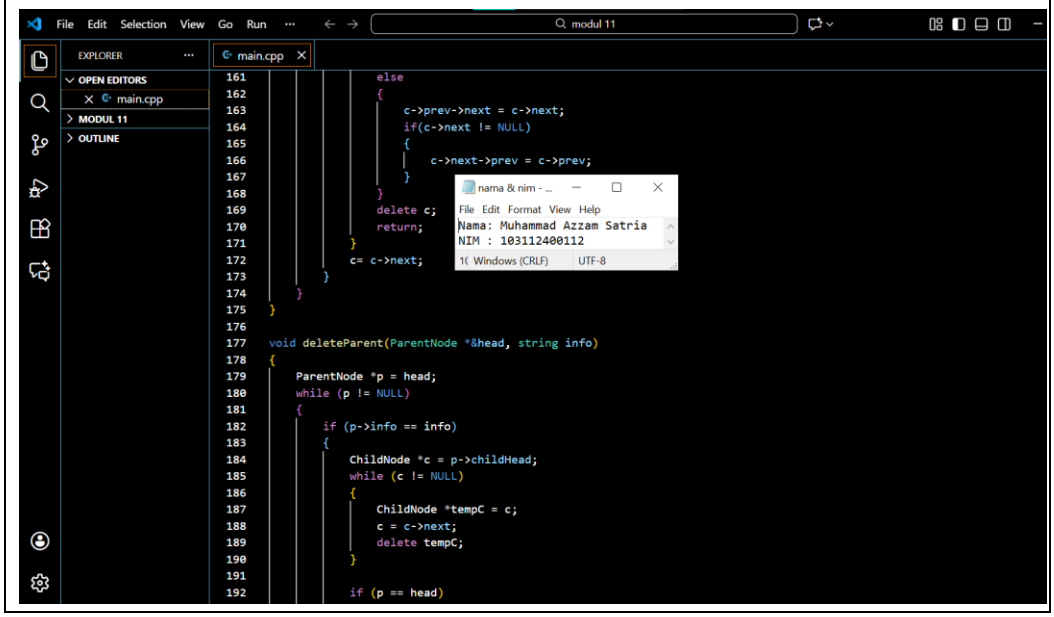


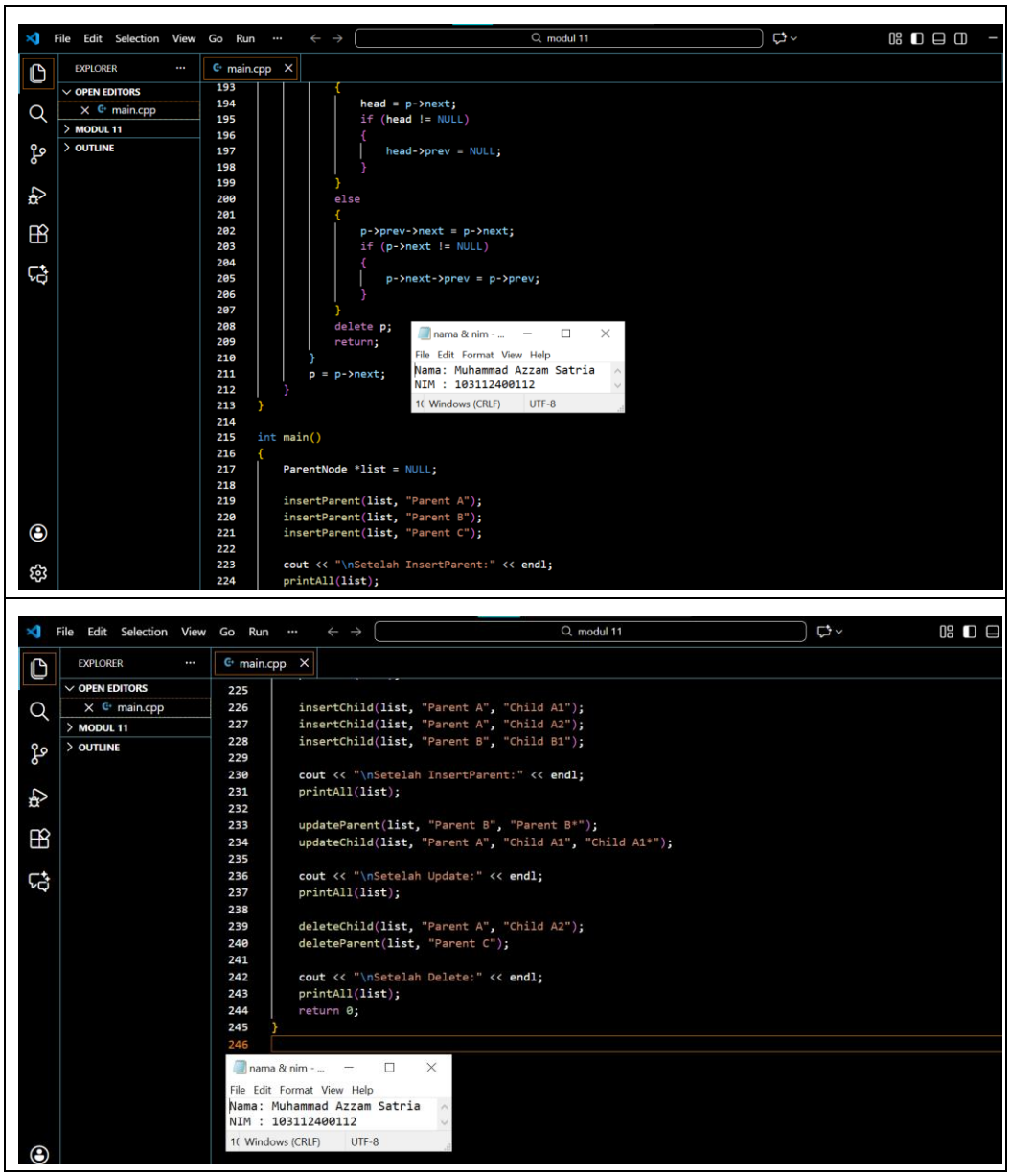

```
65
66      if (p != NULL)
67      {
68          ChildNode *newChild = createChild(childInfo);
69          if (p->childHead == NULL)
70          {
71              p->childHead = newChild;
72          }
73          else
74          {
75              ChildNode *c = p->childHead;
76              while (c->next != NULL)
77              {
78                  c = c->next;
79              }
80              c->next = newChild;
81              newChild->prev = c;
82          }
83      }
84  }
85
86  void printAll(ParentNode *head)
87  {
88      while (head != NULL)
89      {
90          cout << head->info;
91          ChildNode *c = head->childHead;
92          while (c != NULL)
93          {
94              cout << "->" << c->info;
95              c = c->next;
96          }
```



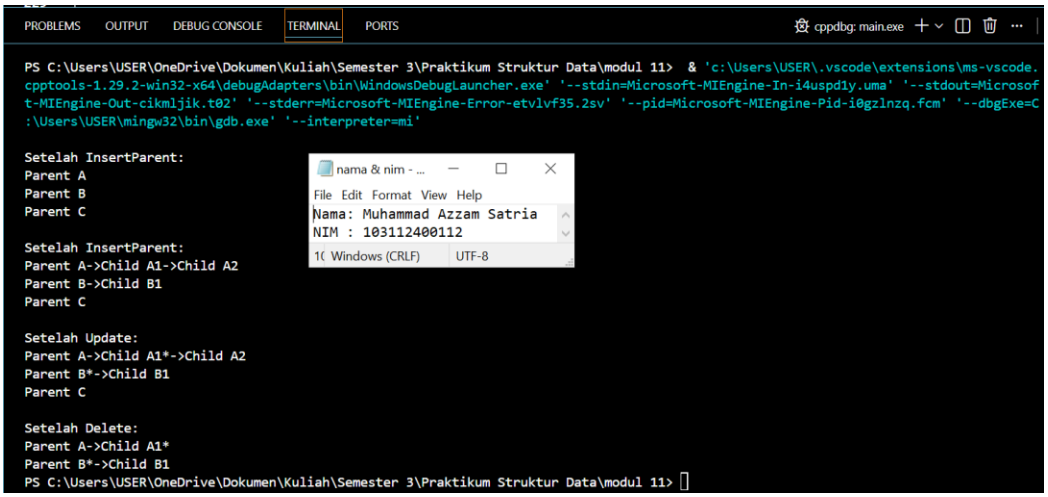

```
97          cout << endl;
98          head = head->next;
99      }
100 }
101
102 void updateParent(ParentNode *head, string oldInfo, string newInfo)
103 {
104     ParentNode *p = head;
105     while (p!= NULL)
106     {
107         if (p->info == oldInfo)
108         {
109             p->info = newInfo;
110             return;
111         }
112         p = p->next;
113     }
114 }
115
116 void updateChild(ParentNode *head, string parentInfo, string oldChildInfo, string newChildInfo)
117 {
118     ParentNode *p = head;
119     while (p != NULL && p->info != parentInfo)
120     {
121         p = p->next;
122     }
123     if (p != NULL)
124     {
125         ChildNode *c = p->childHead;
126         while (c != NULL)
127         {
128             if (c->info == oldChildInfo)
```


File Edit Selection View Go Run ...
modul 11
EXPLORER
main.cpp
OPEN EDITORS
main.cpp
MODUL 11
OUTLINE
nama & nim - ...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Ln 1, Col 1 Windows (CRLF) UTF-8
129 {
130 c->info = newChildInfo;
131 return;
132 }
133 c = c->next;
134 }
135 }
136 }
137
138 void deleteChild(ParentNode *head, string parentInfo, string childInfo)
139 {
140 ParentNode *p = head;
141 while (p != NULL && p->info != parentInfo)
142 {
143 p = p->next;
144 }
145
146 if (p != NULL)
147 {
148 ChildNode *c = p->childHead;
149 while (c != NULL)
150 {
151 if (c->info == childInfo)
152 {
153 if (c == p->childHead)
154 {
155 p->childHead = c->next;
156 if(p->childHead != NULL)
157 {
158 p->childHead->prev = NULL;
159 }
160 }


File Edit Selection View Go Run ...
modul 11
EXPLORER
main.cpp
OPEN EDITORS
main.cpp
MODUL 11
OUTLINE
nama & nim - ...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Ln 1, Col 1 Windows (CRLF) UTF-8
161 else
162 {
163 c->prev->next = c->next;
164 if(c->next != NULL)
165 {
166 c->next->prev = c->prev;
167 }
168 }
169 delete c;
170 return;
171 }
172 c= c->next;
173 }
174 }
175 }
176
177 void deleteParent(ParentNode *&head, string info)
178 {
179 ParentNode *p = head;
180 while (p != NULL)
181 {
182 if (p->info == info)
183 {
184 ChildNode *c = p->childHead;
185 while (c != NULL)
186 {
187 ChildNode *tempC = c;
188 c = c->next;
189 delete tempC;
190 }
191
192 if (p == head)



Screenshots Output

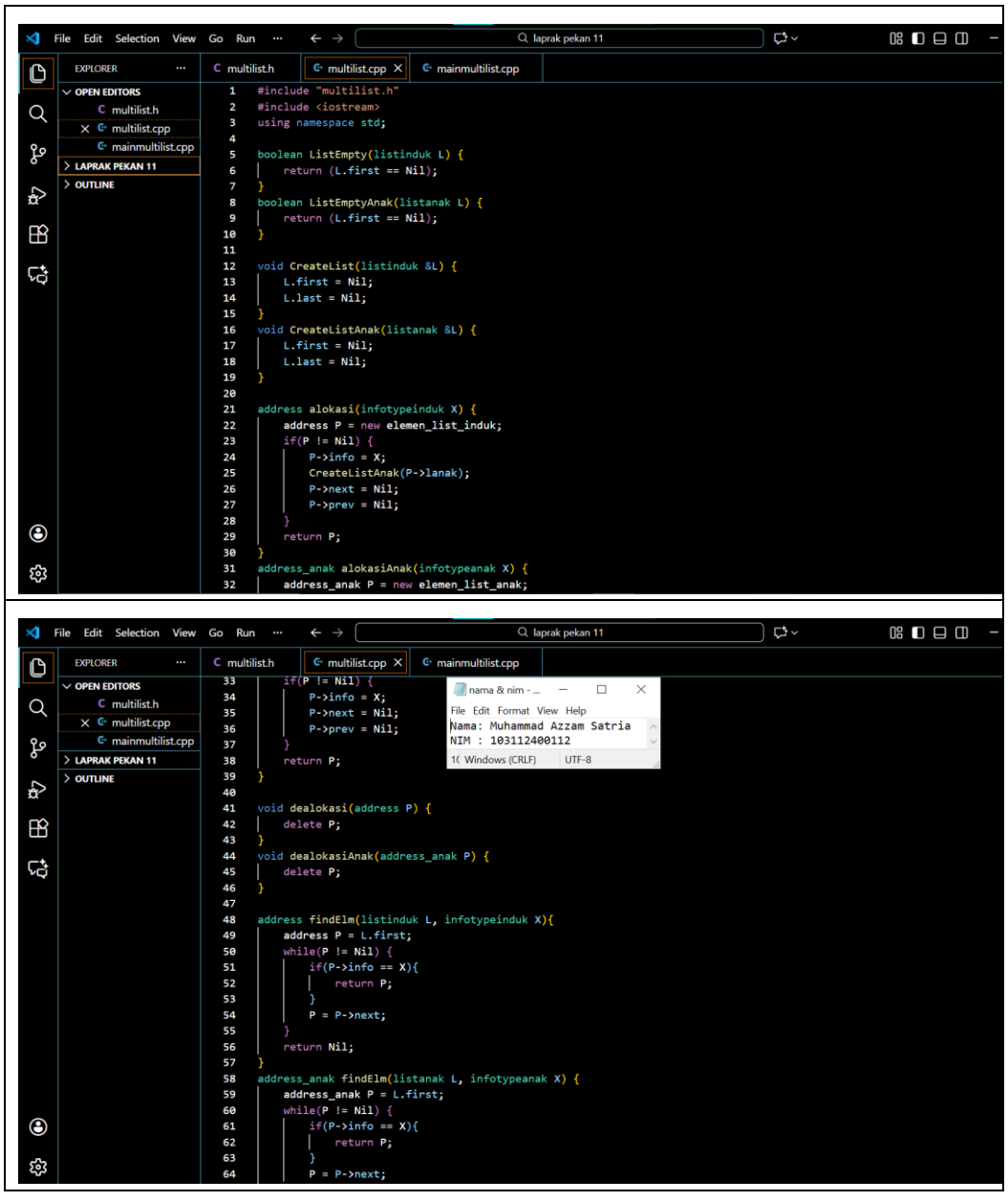


Deskripsi :

Program ini merupakan penerapan dari struktur data multi linked list, yang mana setiap parent mempunyai daftar child dengan bentuk linked list juga. Pada program ini struct ParentNode dan ChildNode didefinisikan dengan pointer next dan prev, setiap parent mempunyai pointer tambahan yaitu childHead untuk menghubungkan list anak. Fungsi createParent dan createChild yaitu untuk membuat node baru dengan menetapkan pointer kosong. Prosedur insertParent fungsinya yaitu untuk menyisipkan parent dengan menelusuri list hingga node terakhir sebelum menambahkan parent baru di bagian ujung list, sedangkan prosedur insertChild fungsinya yaitu menyisipkan child dengan mencari parent berdasarkan nama lalu menambah child di posisi terakhir pada daftar anak.

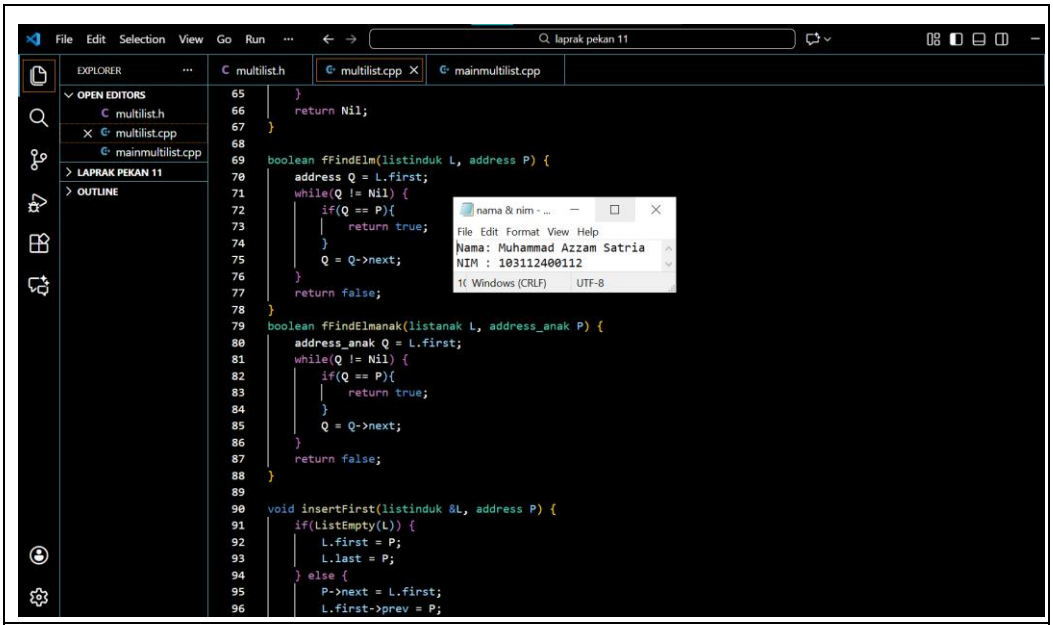
Fungsi dari prosedur printAll yaitu untuk menampilkan semua parent dan daftar child dengan menelusuri kedua list tersebut. Untuk mengubah suatu data, prosedur updateParent akan mencari node parent yang sesuai kemudian mengganti isi node tersebut dengan string baru, sedangkan prosedur updateChild fungsinya yaitu untuk mencari parent tertentu dengan menelusuri child list hingga mendapatkan child yang ingin diubah. Prosedur deletParent mempunyai fungsi sebagai menghapus node child dengan memperbaiki pointer prev dan next sesuai dengan posisi child yang dihapus. Selanjutnya terdapat prosedur deleteParent yang berfungsi sebagai menghapus semua child pada parent sebelum menghapus node parent nya dari list. Di dalam fungsi main, program menjalankan semua perintah seperti insert, update, delete dan print untuk menampilkan semua data pada program tersebut.

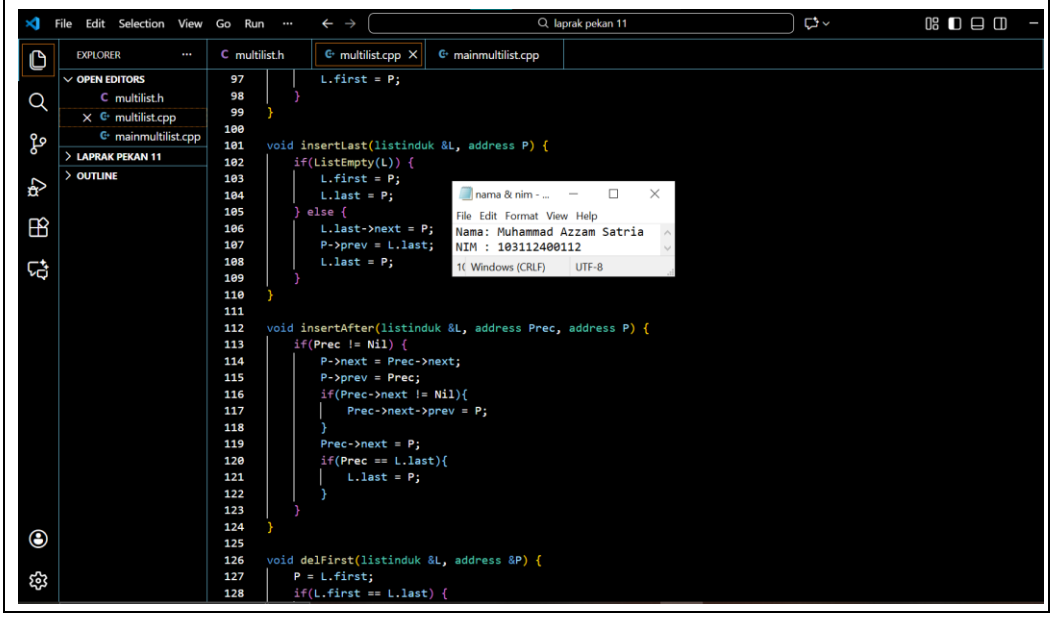
C. Unguided 1

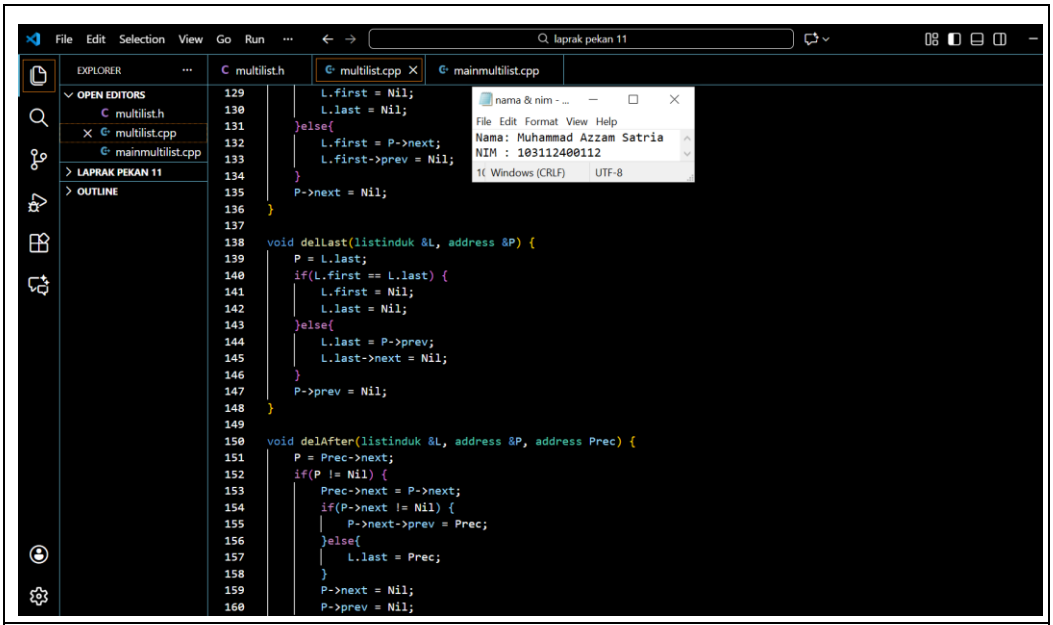


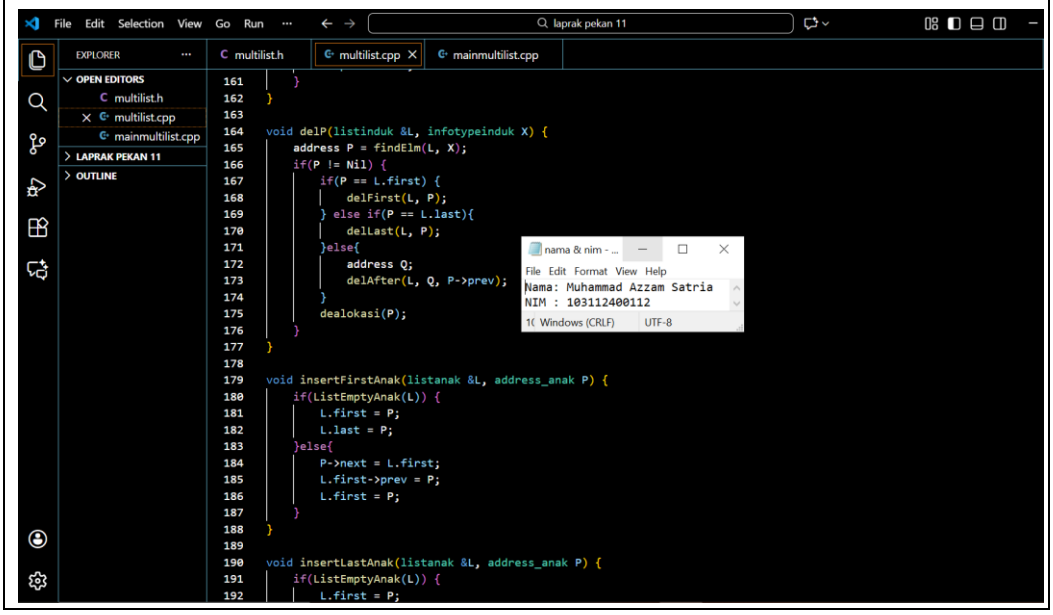
```
#include "multilist.h"
#include <iostream>
using namespace std;

boolean ListEmpty(listinduk L) {
    return (L.first == Nil);
}
boolean ListEmptyAnak(listanak L) {
    return (L.first == Nil);
}

void CreateList(listinduk &L) {
    L.first = Nil;
    L.last = Nil;
}
void CreateListAnak(listanak &L) {
    L.first = Nil;
    L.last = Nil;
}

address alokasi(infotypeinduk X) {
    address P = new elemen_list_induk;
    if(P != Nil) {
        P->info = X;
        CreateListAnak(P->lanak);
        P->next = Nil;
        P->prev = Nil;
    }
    return P;
}
address_anak alokasiAnak(infotypeanak X) {
    address_anak P = new elemen_list_anak;
    if(P != Nil) {
        P->info = X;
        P->next = Nil;
        P->prev = Nil;
    }
    return P;
}

void dealokasi(address P) {
    delete P;
}
void dealokasiAnak(address_anak P) {
    delete P;
}

address findElm(listinduk L, infotypeinduk X){
    address P = L.first;
    while(P != Nil) {
        if(P->info == X){
            return P;
        }
        P = P->next;
    }
    return Nil;
}
address_anak findElm(listanak L, infotypeanak X) {
    address_anak P = L.first;
    while(P != Nil) {
        if(P->info == X){
            return P;
        }
        P = P->next;
```

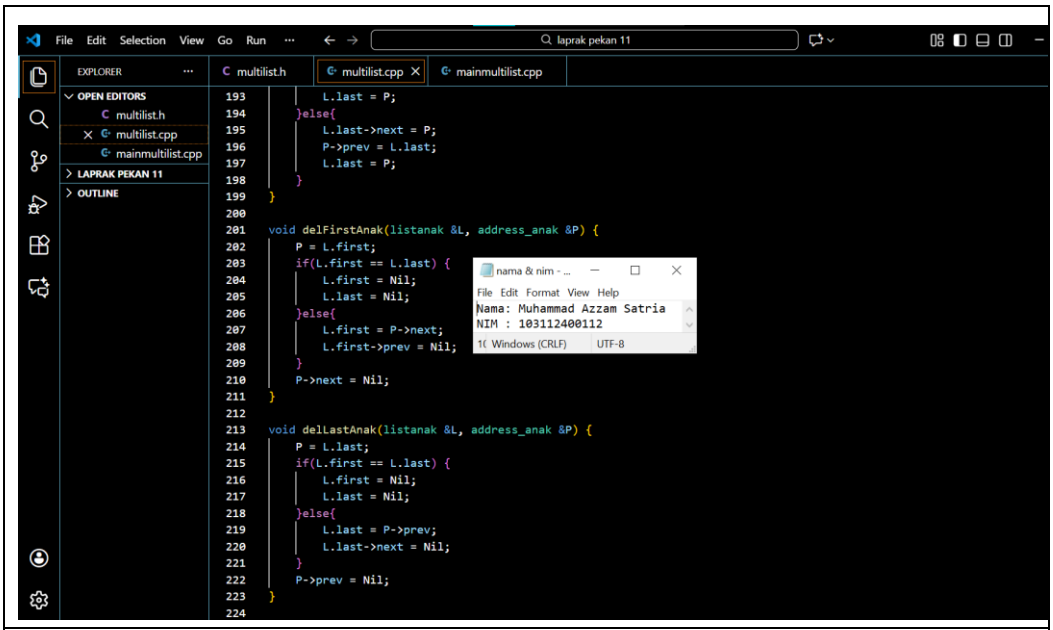
```
nama & nim - ...
File  Edit  Format  View  Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Ln 1, Col 1   Windows (CRLF)   UTF-8
```

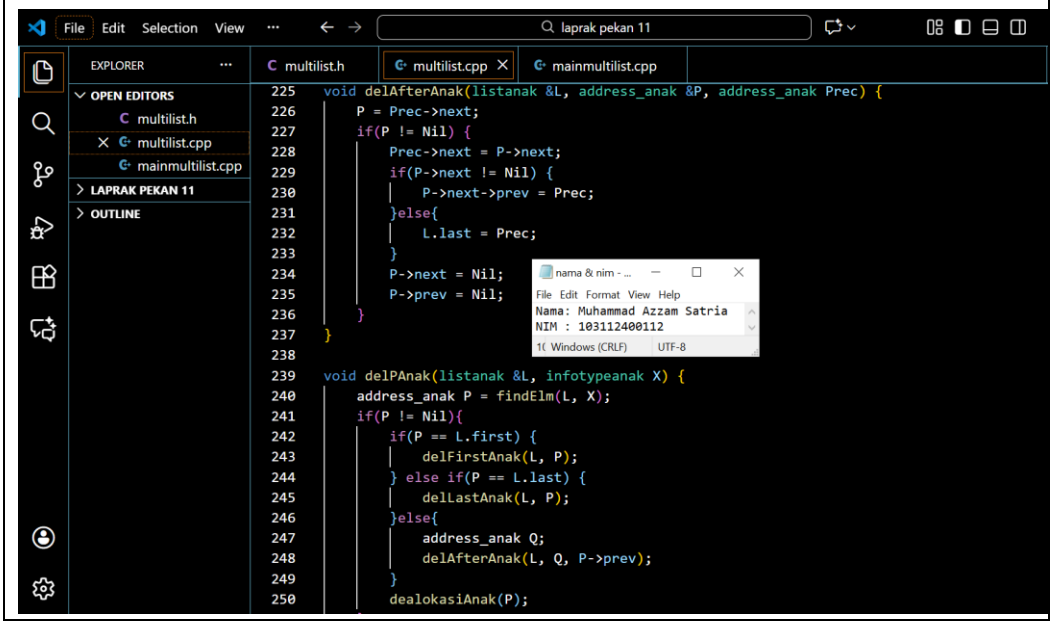

File Edit Selection View Go Run
laprak pekan 11
EXPLORER
multilist.h
multilist.cpp
mainmultilist.cpp
OPEN EDITORS
LAPRAK PEKAN 11
OUTLINE
nama & nim - ...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Windows (CRLF)
UTF-8

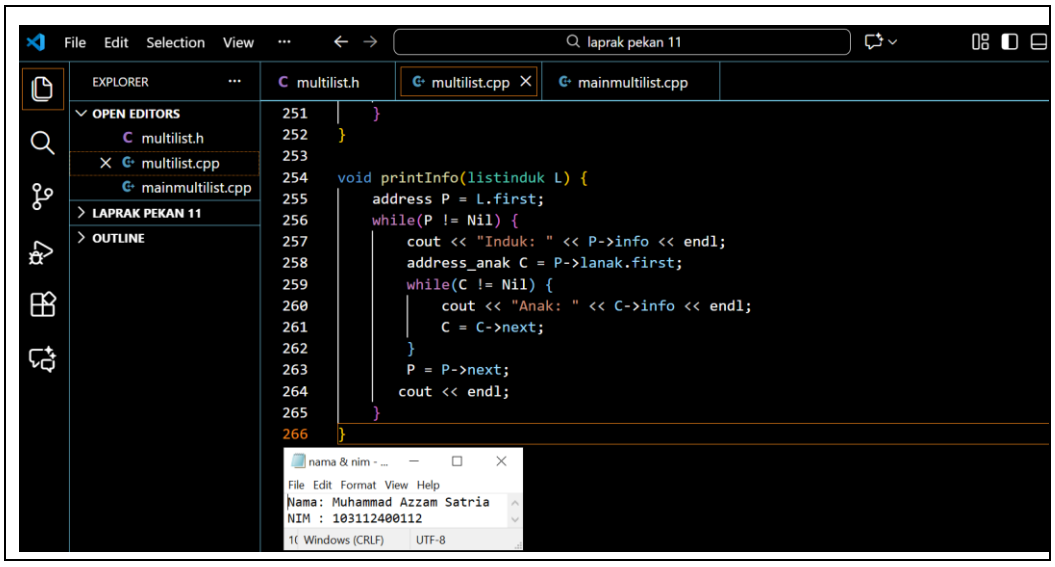

File Edit Selection View Go Run
laprak pekan 11
EXPLORER
multilist.h
multilist.cpp
mainmultilist.cpp
OPEN EDITORS
LAPRAK PEKAN 11
OUTLINE
nama & nim - ...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Windows (CRLF)
UTF-8


File Edit Selection View Go Run
laprak pekan 11
EXPLORER
multilist.h
multilist.cpp
mainmultilist.cpp
OPEN EDITORS
LAPRAK PEKAN 11
OUTLINE
nama & nim - ...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Windows (CRLF)
UTF-8
129 L.first = Nil;
130 L.last = Nil;
131 }else{
132 L.first = P->next;
133 L.first->prev = Nil;
134 }
135 P->next = Nil;
136 }
137
138 void delLast(listinduk &L, address &P) {
139 P = L.last;
140 if(L.first == L.last) {
141 L.first = Nil;
142 L.last = Nil;
143 }else{
144 L.last = P->prev;
145 L.last->next = Nil;
146 }
147 P->prev = Nil;
148 }
149
150 void delAfter(listinduk &L, address &P, address Prec) {
151 P = Prec->next;
152 if(P != Nil) {
153 Prec->next = P->next;
154 if(P->next != Nil) {
155 P->next->prev = Prec;
156 }else{
157 L.last = Prec;
158 }
159 P->next = Nil;
160 P->prev = Nil;

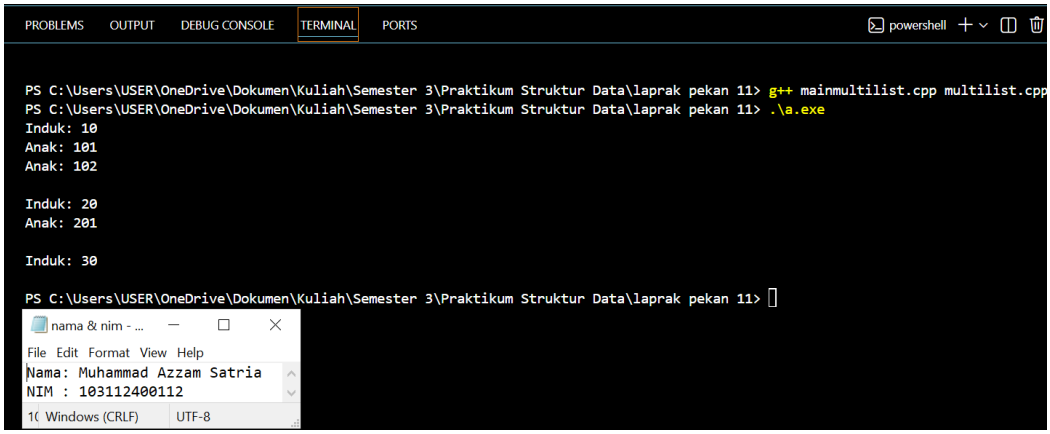

File Edit Selection View Go Run
laprak pekan 11
EXPLORER
multilist.h
multilist.cpp
mainmultilist.cpp
OPEN EDITORS
LAPRAK PEKAN 11
OUTLINE
nama & nim - ...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Windows (CRLF)
UTF-8
161 }
162 }
163
164 void delP(listinduk &L, infotypeinduk X) {
165 address P = findElm(L, X);
166 if(P != Nil) {
167 if(P == L.first) {
168 delFirst(L, P);
169 } else if(P == L.last){
170 delLast(L, P);
171 }else{
172 address Q;
173 delAfter(L, Q, P->prev);
174 }
175 dealokasi(P);
176 }
177 }
178
179 void insertFirstAnak(listanak &L, address_anak P) {
180 if(ListEmptyAnak(L)) {
181 L.first = P;
182 L.last = P;
183 }else{
184 P->next = L.first;
185 L.first->prev = P;
186 L.first = P;
187 }
188 }
189
190 void insertLastAnak(listanak &L, address_anak P) {
191 if(ListEmptyAnak(L)) {
192 L.first = P;


File Edit Selection View Go Run
laprak pekan 11
EXPLORER
multilist.h
multilist.cpp
mainmultilist.cpp
OPEN EDITORS
LAPRAK PEKAN 11
OUTLINE
L.last = P;
}else{
L.last->next = P;
P->prev = L.last;
L.last = P;
}
}
void delFirstAnak(listanak &L, address_anak &P) {
P = L.first;
if(L.first == L.last) {
L.first = Nil;
L.last = Nil;
}else{
L.first = P->next;
L.first->prev = Nil;
}
P->next = Nil;
}
void delLastAnak(listanak &L, address_anak &P) {
P = L.last;
if(L.first == L.last) {
L.first = Nil;
L.last = Nil;
}else{
L.last = P->prev;
L.last->next = Nil;
}
P->prev = Nil;
}
nama & nim - ...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Windows (CRLF)
UTF-8


File Edit Selection View
laprak pekan 11
EXPLORER
multilist.h
multilist.cpp
mainmultilist.cpp
OPEN EDITORS
LAPRAK PEKAN 11
OUTLINE
void delAfterAnak(listanak &L, address_anak &P, address_anak Prec) {
P = Prec->next;
if(P != Nil) {
Prec->next = P->next;
if(P->next != Nil) {
P->next->prev = Prec;
}else{
L.last = Prec;
}
P->next = Nil;
P->prev = Nil;
}
}
void delPAnak(listanak &L, infotypeanak X) {
address_anak P = findElm(L, X);
if(P != Nil){
if(P == L.first) {
delFirstAnak(L, P);
} else if(P == L.last) {
delLastAnak(L, P);
}else{
address_anak Q;
delAfterAnak(L, Q, P->prev);
}
dealokasiAnak(P);
nama & nim - ...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Windows (CRLF)
UTF-8



Screenshots Output

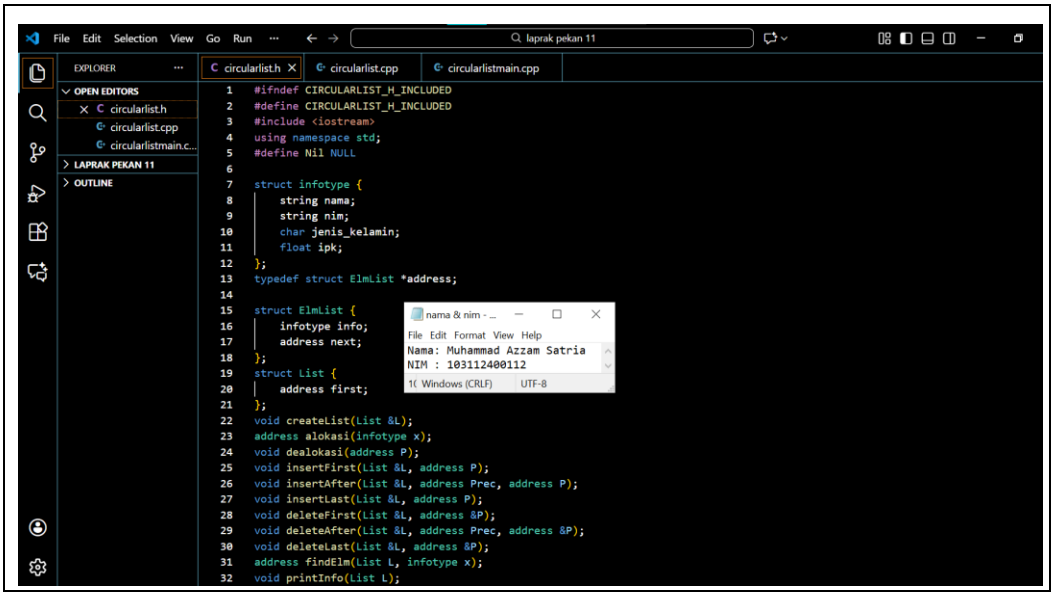


Deskripsi:

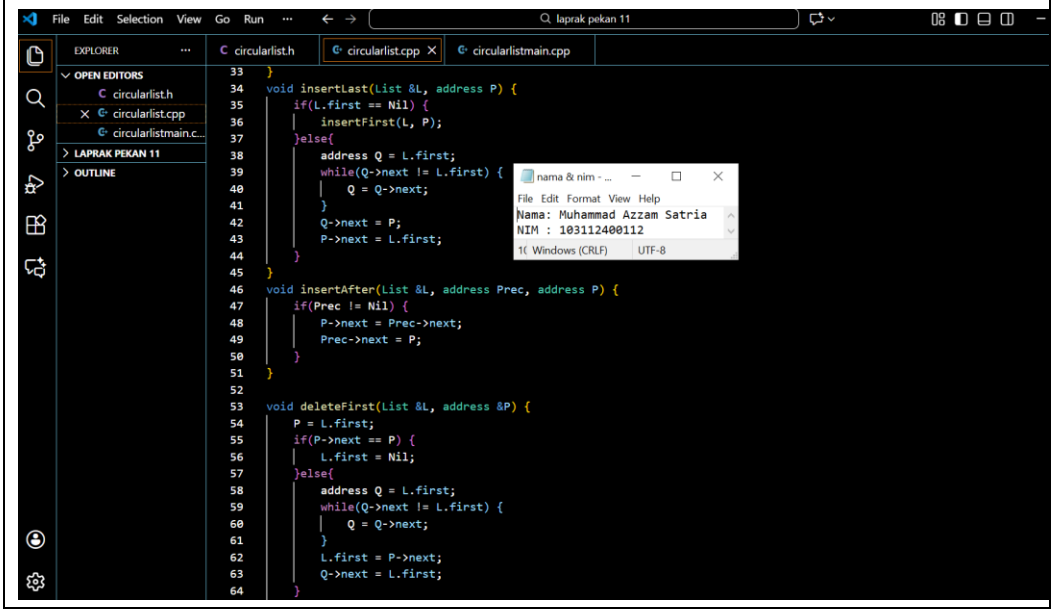
Pada soal nomor 2, program menyusun kembali semua fungsi yang ada di dalam file header (multilist.h) ke dalam file multilist.cpp. Pada bagian awal program terdapat fungsi untuk mengecek kondisi list apakah list induk atau list anak masih kosong. Jika kosong maka program akan menjalankan fungsi CreateList dan CreateListAnak untuk menginisialisasi list dengan nilai pointer first dan last bernilai kosong. Kemudian terdapat juga fungsi alokasi dan dealokasi yang cara kerjanya yaitu menyiapkan node baru dan mengembalikan memori node yang sudah tidak terpakai. Fungsi findElm yaitu untuk mencari elemen list induk dan list anak dengan mengecek node satu per satu menggunakan pointer next.

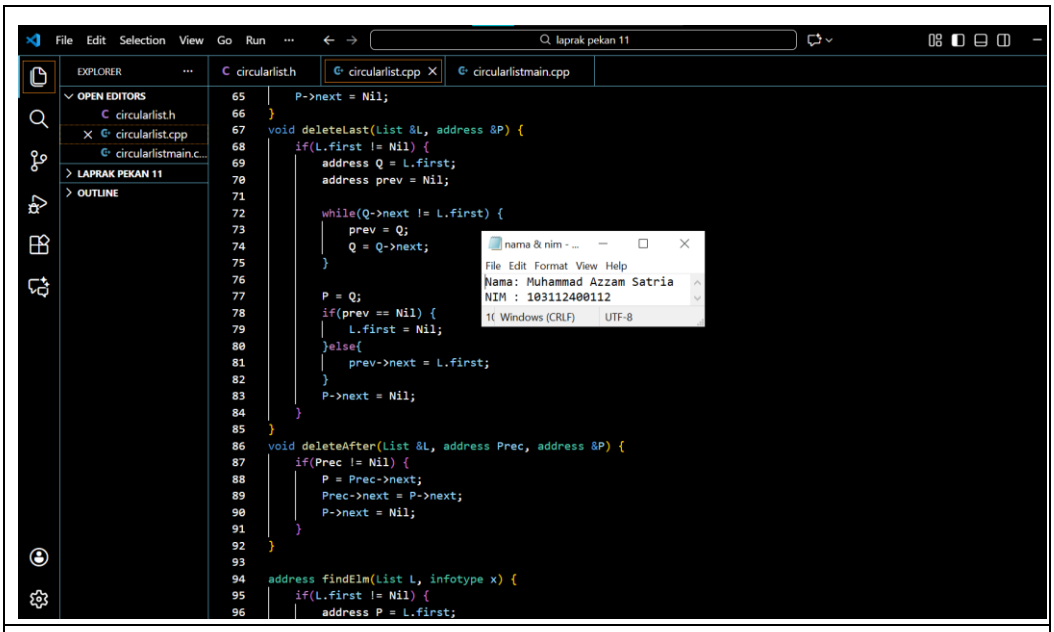
Selain memiliki fungsi didalamnya, program juga mempunyai beberapa prosedur diantaranya yaitu insertFirst, insertLast dan insertAfter untuk menambah elemen dengan mengatur ulang pointer next dan prev agar tetap mempunyai hubungan antar node yang terhubung dengan benar. Lalu terdapat juga prosedur penghapusan elemen seperti delFirst, delLast dan delAfter yang fungsinya yaitu mengambil node yang berada di posisi tertentu kemudian menghapus hubungannya dari list. Kemudian terdapat juga prosedur delP dan delPAnak untuk menhapus elemen berdasarkan nilai info sehingga tidak memerlukan alamat node secara langsung Prosedur yang terakhir yaitu printInfo yang cara kerjanya adalah menampilkan seluruh data induk dan anak yang dimilikinya dengan menelusuri list dari awal sampai akhir.

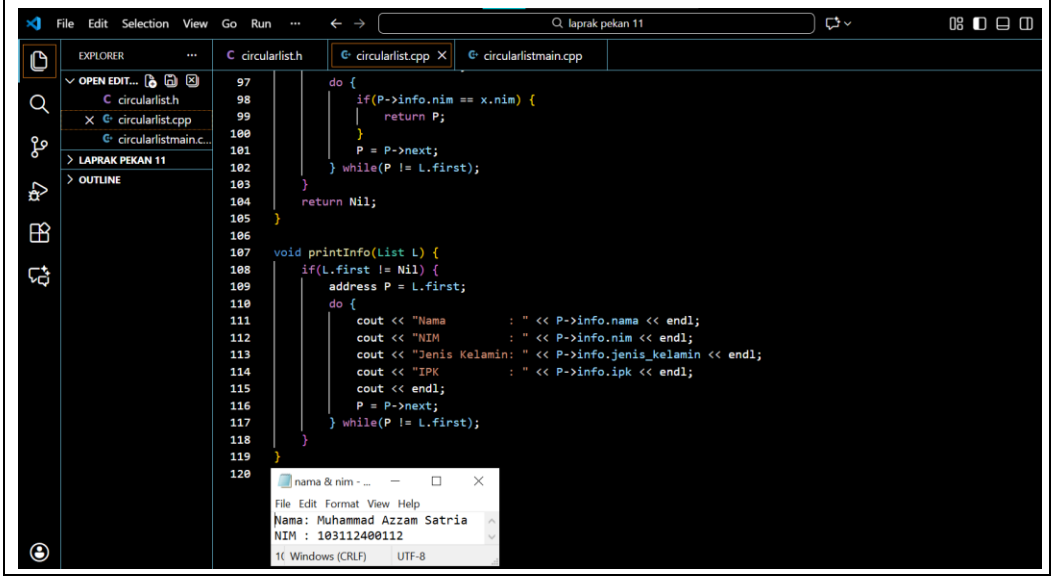
Unguided 2

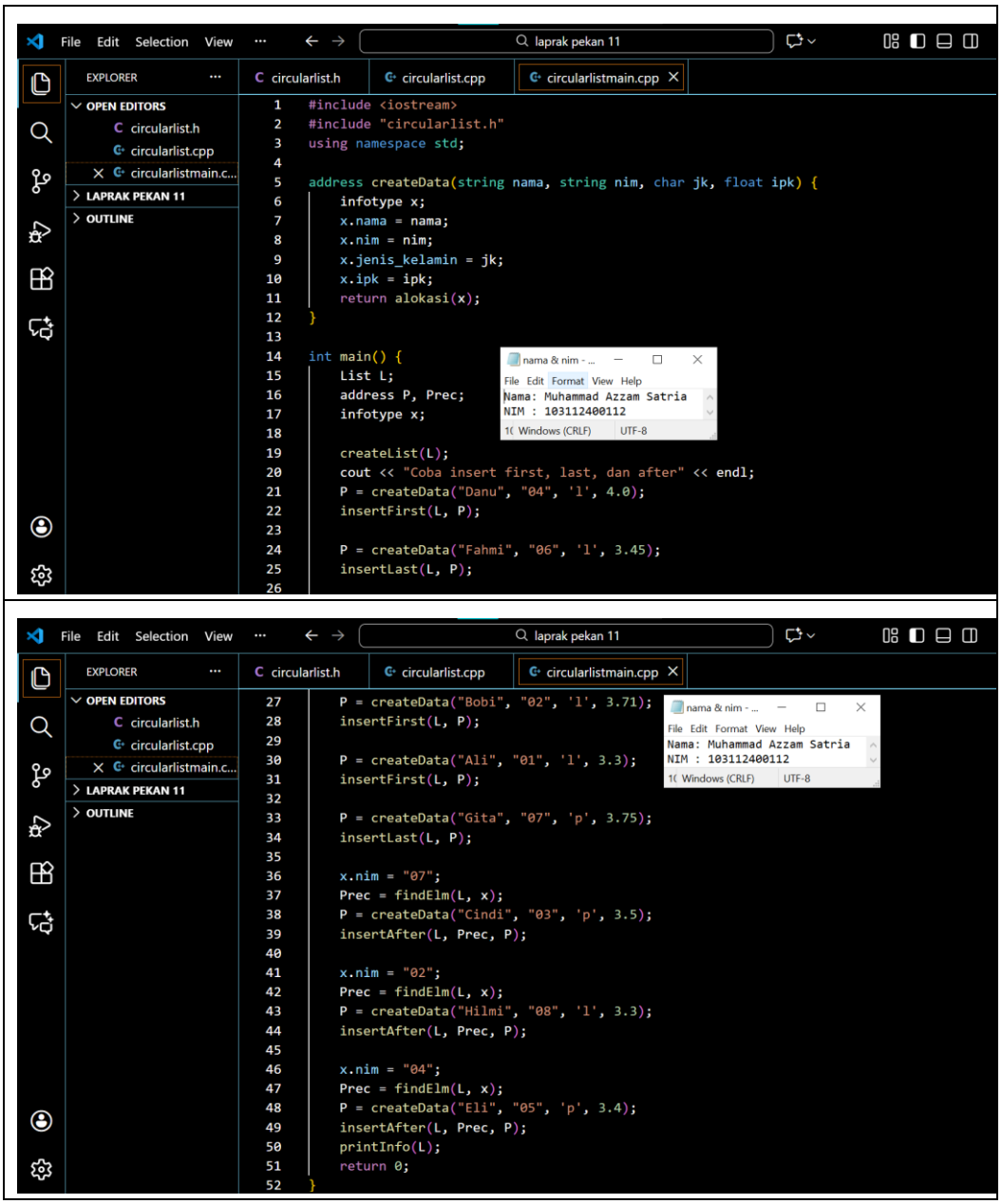



File Edit Selection View Go Run
laprak pekan 11
EXPLORER
OPEN EDITORS
circularlist.h
circularlist.cpp
circularlistmain.c...
LAPRAK PEKAN 11
OUTLINE
circularlist.h
circularlist.cpp
circularlistmain.cpp
#include "circularlist.h"
using namespace std;
void createList(List &L) {
L.first = Nil;
}
address alokasi(infotype x) {
address P = new ElmList;
if(P != Nil) {
P->info = x;
P->next = P;
}
return P;
}
void dealokasi(address P) {
delete P;
}
void insertFirst(List &L, address P) {
if(L.first == Nil) {
L.first = P;
P->next = P;
}else{
address Q = L.first;
while(Q->next != L.first) {
Q = Q->next;
}
P->next = L.first;
Q->next = P;
L.first = P;
}
nama & nim - ...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Windows (CRLF)
UTF-8

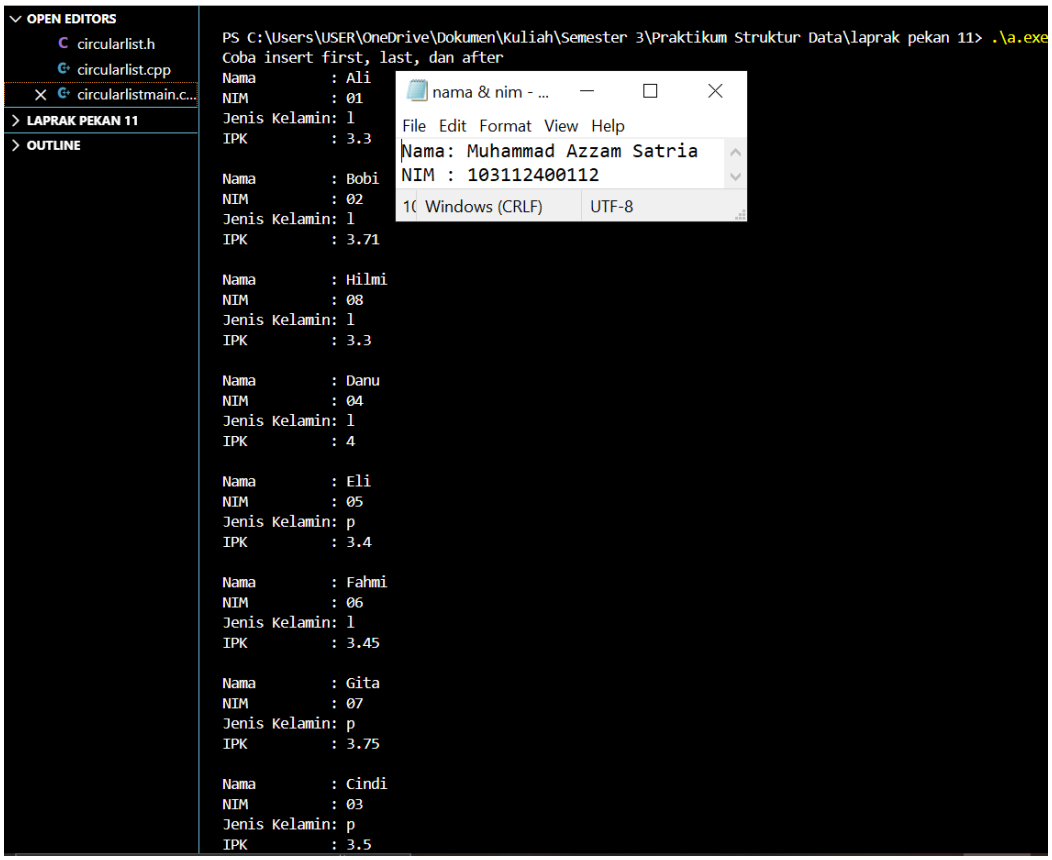

File Edit Selection View Go Run
laprak pekan 11
EXPLORER
OPEN EDITORS
circularlist.h
circularlist.cpp
circularlistmain.c...
LAPRAK PEKAN 11
OUTLINE
circularlist.h
circularlist.cpp
circularlistmain.cpp
}
void insertLast(List &L, address P) {
if(L.first == Nil) {
insertFirst(L, P);
}else{
address Q = L.first;
while(Q->next != L.first) {
Q = Q->next;
}
Q->next = P;
P->next = L.first;
}
}
void insertAfter(List &L, address Prec, address P) {
if(Prec != Nil) {
P->next = Prec->next;
Prec->next = P;
}
}
void deleteFirst(List &L, address &P) {
P = L.first;
if(P->next == P) {
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 & nim - ...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Windows (CRLF)
UTF-8


File Edit Selection View Go Run
laprak pekan 11
EXPLORER
circularlist.h
circularlist.cpp
circularlistmain.cpp
OPEN EDITORS
LAPRAK PEKAN 11
OUTLINE
P->next = Nil;
}
void deleteLast(List &L, address &P) {
if(L.first != Nil) {
address Q = L.first;
address prev = Nil;
while(Q->next != L.first) {
prev = Q;
Q = Q->next;
}
P = Q;
if(prev == Nil) {
L.first = Nil;
}else{
prev->next = L.first;
}
P->next = Nil;
}
}
void deleteAfter(List &L, address Prec, address &P) {
if(Prec != Nil) {
P = Prec->next;
Prec->next = P->next;
P->next = Nil;
}
}
address findElm(List L, infotype x) {
if(L.first != Nil) {
address P = L.first;
nama & nim - ...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Windows (CRLF)
UTF-8


File Edit Selection View Go Run
laprak pekan 11
EXPLORER
circularlist.h
circularlist.cpp
circularlistmain.cpp
LAPRAK PEKAN 11
OUTLINE
do {
if(P->info.nim == x.nim) {
return P;
}
P = P->next;
} while(P != L.first);
}
return Nil;
}
void printInfo(List L) {
if(L.first != Nil) {
address P = L.first;
do {
cout << "Nama : " << P->info.nama << endl;
cout << "NIM : " << P->info.nim << endl;
cout << "Jenis Kelamin: " << P->info.jenis_kelamin << endl;
cout << "IPK : " << P->info.ipk << endl;
cout << endl;
P = P->next;
} while(P != L.first);
}
}
nama & nim - ...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Windows (CRLF)
UTF-8



Screenshosts output



Deskripsi:

Pada soal nomor 3 program mendefinisikan struktur circular linked list dengan file header yang berisi tipe data dan deklarasi fungsi. Tipe data infotype berguna untuk menampung data mahasiswa berupa nama, nim, jenis kelamin dan ipk. Struct ElmList memiliki fungsi sebagai node yang menyimpan satu data mahasiswa dan pointer next yang mengarag ke elemen selanjutnya. Struct List hanya mempunyai pointer first yang yang fungsinya sebagai penanda awal list. Pada file heade terdapat juga beberapa fungsi dan prosedur, diantaranya yaitu prosedur CreateList sebagai inisialisasi list, fungsi alokasi untuk membuat node baru yang dihubungkan ke list, prosedur dealokasi dipakai ketika suatu node dihapus, prosedur insertFirst, insertLast dan insertAfter untuk menyisipkan node di lokasi tertentu, prosedur deleteFirst, deleteLast dan deleteAfter untuk menghapus node di lokasi tertentu, fungsi findElm untuk mencari node berdasarkan nim mahasiswa. Terakhir prosedur printInfo untuk menampilkan seluruh isi list pada program secara berurutan.

Pada file circularlist.cpp, CreateList mengatur pointer first menjadi nil agar menjadi tanda list tersebut masih kosong. Fungsi alokasi akan membuat node baru dan mengisi semua atribut mahasiswa. Pointer next akan mengarah ke dirinya sendiri agar membentuk circular. Pada prosedur insertFirst, node baru ditempatkan di depan list dengan cara mencari node terakhir agar pointer next terakhir dapat diarahkan ke node baru. Prosedur insertLast menyimpan node di posisi paling akhir dengan menelusuri node hingga menemukan node yang diberikan melalui pointer Prec dan cara penghubungannya yaitu menganti pointer next dari Prec. Prosedur deleteFirst menghapus node awal dengan mencari node terakhir dan mengarahkan pointer last ke node kedua. Prosedur deleteLast menghapus node terakhir dengan mencari node sebelum last dan mengubah pointer next nya kembali ke first. Prosedur deleteAfter menggunakan pointerPrec sebagai patokan untuk menghapus node yang berada di setelahnya. Fungsi findElm yaitu untuk menelusuri semua node dengan struktur do while agar semua node dikunjungi dalam bentuk circular dan dari setiap iterasi akan mencockkan nim dan mengembalikan pointer jika data didapatkan.

Pada file circularmain.cpp program akan memanggil CreateList untuk membuat list kosong. Kemudian data mahasiswa beberapa dibuat dengan createData agar setiap atribut dikumpulkan sebelum diubah menjadi node baru. Penyisipan data dilakukan menggunakan insertFirst dan insertLast untuk membuat susunan circular list dan sebagai dasar dari proses selanjutnya. Fungsi findElm akan mencari node dengan nim tertentu sebagai patokan penyisipan data berikutnya. Hasil pencarian node akan dipakai di insertAfter untuk menempatkan node baru tepat setelah node yang ditemukan. Proses dijalankan berulang kali untuk mengisi list di posis yang berbeda dengan tidak mengubah node yang ada. Setelah penyisipan selesai, program akan menjalankan printInfo untuk menampilkan seluruh data mahasiswa yang telah diinputkan.

D. Kesimpulan

Berdasarkan modul, guided dan latihan soal dari struktur data multi linked list, saya dapat menjadi lebih paham mengenai struktur data ini. Multi linked list digunakan pada data yang mempunyai hubungan bertingkat antara elemen induk dan anak. Setiap elemen induk terhubung dengan list anaknya sendiri. Dengan konsep seperti ini saya belajar bagaimana menyisipkan, mencari dan menghapus data tanpa merusak hubungan antar node. Dalam implementasi program, setiap

operasi perlu adanya penelusuran pointer yang berurutan agar struktur induk dan anak tetap terhubung dengan benar.

Pada bagian latihan circular linked list, diperlihatkan bagaimana node terakhir harus diarahkan kembali ke node pertama agar tetap circular. Penyisipan node dilakukan dengan menyesuaikan pointer tergantung dimana posisi data ditambahkan. Operasi delete juga menjaga agar pointer tetap menyatu setelah satu node dilepas. Terakhir operasi pencarian menggunakan do while karena circular list tidak memiliki node akhir

E. Referensi

Wijoyo, A., Yuliza, M. F., Permatasari, G. A., & Ningsih, K. (2024). *Penggunaan algoritma doubly linked list untuk insertion dan deletion*. Jurnal Riset Informatika dan Inovasi (JRIIN), 1(12), 1–8.

Mbejo, M. T., Oktaviani, F., Simarmata, J., & Hutapea, Y. (2025). *Analisis struktur data linked list dalam pengolahan data mahasiswa*. Jurnal Sains Informatika Terapan (JSIT), 2**(**2), 1–10.

Sahid, S., Purba, D. R., & Silitonga, M. (2025). *Implementasi queue berbasis linked list pada aplikasi web manajemen antrian print mahasiswa*. Jurnal Ilmu Komputer dan Informatika, 6(1), 20–28.