

**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 daftar 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

```
#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;
    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;
    }
}
```

```
File Edit Selection View Go Run ... ← → Q: modul 11
EXPLORER ...
OPEN EDITORS X main.cpp
MODUL 11
OUTLINE

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

File Edit Selection View Go Run ... ← → Q: modul 11
EXPLORER ...
OPEN EDITORS X main.cpp
MODUL 11
OUTLINE

97     cout << endl;
98     head = head->next;
99 }
100 }

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)
```

The image shows two screenshots of a code editor interface, likely Visual Studio Code, displaying C++ code for a linked list manipulation program. Both screenshots have a dark theme.

Screenshot 1: This screenshot shows the implementation of the `deleteChild` function. The code iterates through a list of parent nodes to find the one whose child node matches the specified `childInfo`. Once found, it iterates through the child node's list to find the specific child node to delete. If found, it updates the parent node's child head pointer to skip the child node. The code uses `prev` and `next` pointers to manage the list structure.

```
129     {
130         c->info = newChildInfo;
131         return;
132     }
133     c = c->next;
134 }
135 }
136 }

137 void deleteChild(ParentNode *head, string parentInfo, string childInfo)
138 {
139     ParentNode *p = head;
140     while (p != NULL && p->info != parentInfo)
141     {
142         p = p->next;
143     }
144
145     if (p != NULL)
146    {
147        ChildNode *c = p->childHead;
148        while (c != NULL)
149        {
150            if (c->info == childInfo)
151            {
152                if (c == p->childHead)
153                {
154                    p->childHead = c->next;
155                    if(p->childHead != NULL)
156                    {
157                        p->childHead->prev = NULL;
158                    }
159                }
160            }
161        }
162    }
163 }
```

Screenshot 2: This screenshot shows the implementation of the `deleteParent` function. The code iterates through a list of parent nodes to find the one whose info matches the specified `info`. Once found, it iterates through the parent node's child list to find the specific child node to delete. If found, it updates the parent node's child head pointer to skip the child node. The code uses `prev` and `next` pointers to manage the list structure.

```
161     else
162    {
163        c->prev->next = c->next;
164        if(c->next != NULL)
165        {
166            c->next->prev = c->prev;
167        }
168        delete c;
169        return;
170    }
171    c= c->next;
172 }

173 }
174 }

175 }

176 void deleteParent(ParentNode *&head, string info)
177 {
178     ParentNode *p = head;
179     while (p != NULL)
180     {
181         if (p->info == info)
182         {
183             ChildNode *c = p->childHead;
184             while (c != NULL)
185             {
186                 ChildNode *tempC = c;
187                 c = c->next;
188                 delete tempC;
189             }
190         }
191     }
192     if (p == head)
```

The screenshots show two instances of a code editor displaying the same C++ file, `main.cpp`, which implements a linked list structure with parent and child nodes.

Top Window Content:

```
193     {
194         head = p->next;
195         if (head != NULL)
196         {
197             head->prev = NULL;
198         }
199     }
200     else
201     {
202         p->prev->next = p->next;
203         if (p->next != NULL)
204         {
205             p->next->prev = p->prev;
206         }
207     }
208     delete p;
209     return;
210 }
211 p = p->next;
212 }
```

```
213 }
```

```
214
215 int main()
216 {
217     ParentNode *list = NULL;
218
219     insertParent(list, "Parent A");
220     insertParent(list, "Parent B");
221     insertParent(list, "Parent C");
222
223     cout << "\nSetelah InsertParent:" << endl;
224     printAll(list);
```

Bottom Window Content:

```
225     insertChild(list, "Parent A", "Child A1");
226     insertChild(list, "Parent A", "Child A2");
227     insertChild(list, "Parent B", "Child B1");
228
229     cout << "\nSetelah InsertParent:" << endl;
230     printAll(list);
231
232
233     updateParent(list, "Parent B", "Parent B*");
234     updateChild(list, "Parent A", "Child A1", "Child A1*");
235
236     cout << "\nSetelah Update:" << endl;
237     printAll(list);
238
239     deleteChild(list, "Parent A", "Child A2");
240     deleteParent(list, "Parent C");
241
242     cout << "\nSetelah Delete:" << endl;
243     printAll(list);
244
245 }
```

```
246 }
```

Output Window (Top Window):

nama & nim - ... — □ ×

File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112

1\ Windows (CRLF) UTF-8

Output Window (Bottom Window):

nama & nim - ... — □ ×

File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112

1\ Windows (CRLF) UTF-8

Screenshots Output

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
cppdbg: main.exe + ... |
```

```

PS C:\Users\USER\OneDrive\Kuliah\Semester 3\Praktikum Struktur Data\modul 11> & 'c:\Users\USER\.vscode\extensions\ms-vscode.cpptools-1.29.2-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-i4uspdly.uma' '--stdout=Microsoft-MIEngine-Out-cikmljik.t02' '--stderr=Microsoft-MIEngine-Error-etvlvf35.2sv' '--pid=Microsoft-MIEngine-Pid-i0gzlnzq.fcm' '--dbgExe=C:\Users\USER\mingw32\bin\gdb.exe' '--interpreter=mi'

Setelah InsertParent:
Parent A
Parent B
Parent C

Setelah InsertParent:
Parent A->Child A1->Child A2
Parent B->Child B1
Parent C

Setelah Update:
Parent A->Child A1*->Child A2
Parent B*->Child B1
Parent C

Setelah Delete:
Parent A->Child A1*
Parent B*->Child B1
PS C:\Users\USER\OneDrive\Kuliah\Semester 3\Praktikum Struktur Data\modul 11>
```

The terminal window shows the execution of a C++ program using the Microsoft MIEngine interface via GDB. It displays four stages of a linked list operation:

- Setelah InsertParent:** Shows three parent nodes (A, B, C) each with an empty child list.
- Setelah InsertParent:** Shows Parent A with two children (A1, A2) and Parent B with one child (B1).
- Setelah Update:** Shows Parent A's child A1 has been updated to point to Child A2, and Parent B's child B1 has been updated to point to Parent C.
- Setelah Delete:** Shows Parent A's child A1 has been deleted, and Parent B's child B1 has been deleted.

A small application window titled "nama & nim -" is overlaid on the terminal, showing the student's name and NIM.

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 deleteParent 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

The image displays two side-by-side code editor windows, likely from a C++ IDE like Dev-C++ or Code::Blocks.

Left Window (multilist.cpp):

```
1 #include "multilist.h"
2 #include <iostream>
3 using namespace std;
4
5 boolean ListEmpty(listinduk L) {
6     return (L.first == Nil);
7 }
8 boolean ListEmptyAnak(listanak L) {
9     return (L.first == Nil);
10 }
11
12 void CreateList(listinduk &L) {
13     L.first = Nil;
14     L.last = Nil;
15 }
16 void CreateListAnak(listanak &L) {
17     L.first = Nil;
18     L.last = Nil;
19 }
20
21 address alokasi(infotypeinduk X) {
22     address P = new elemen_list_induk;
23     if(P != Nil) {
24         P->info = X;
25         CreateListAnak(P->lanak);
26         P->next = Nil;
27         P->prev = Nil;
28     }
29     return P;
30 }
31 address_anak alokasiAnak(infotypeanak X) {
32     address_anak P = new elemen_list_anak;
```

Right Window (multilist.h):

```
33     if(P != Nil) {
34         P->info = X;
35         P->next = Nil;
36         P->prev = Nil;
37     }
38     return P;
39 }
40
41 void dealokasi(address P) {
42     delete P;
43 }
44 void dealokasiAnak(address_anak P) {
45     delete P;
46 }
47
48 address findElm(listinduk L, infotypeinduk X){
49     address P = L.first;
50     while(P != Nil) {
51         if(P->info == X){
52             return P;
53         }
54         P = P->next;
55     }
56     return Nil;
57 }
58 address_anak findElm(listanak L, infotypeanak X) {
59     address_anak P = L.first;
60     while(P != Nil) {
61         if(P->info == X){
62             return P;
63         }
64         P = P->next;
65     }
66 }
```

A tooltip is visible in the right window, showing the following content:

nama & nim -> □ ×
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
10 Windows (CRLF) UTF-8

File Edit Selection View Go Run ... ← → Q laprak pekan 11

EXPLORER ... C multilist.h C multilist.cpp X C mainmultilist.cpp

OPEN EDITORS C multilist.h X C multilist.cpp C mainmultilist.cpp

LAPRAK PEKAN 11

OUTLINE

```
65     }
66     return Nil;
67 }
68
69 boolean ffindElm(listinduk L, address P) {
70     address Q = L.first;
71     while(Q != Nil) {
72         if(Q == P){
73             return true;
74         }
75         Q = Q->next;
76     }
77     return false;
78 }
79 boolean ffindElmanak(listanak L, address_anak P) {
80     address_anak Q = L.first;
81     while(Q != Nil) {
82         if(Q == P){
83             return true;
84         }
85         Q = Q->next;
86     }
87     return false;
88 }
89 void insertFirst(listinduk &L, address P) {
90     if(listEmpty(L)) {
91         L.first = P;
92         L.last = P;
93     } else {
94         P->next = L.first;
95         L.first->prev = P;
96     }
97     L.first = P;
98 }
99 }
```

File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
1(C Windows (CRLF) UTF-8

File Edit Selection View Go Run ... ← → Q laprak pekan 11

EXPLORER ... C multilist.h C multilist.cpp X C mainmultilist.cpp

OPEN EDITORS C multilist.h X C multilist.cpp C mainmultilist.cpp

LAPRAK PEKAN 11

OUTLINE

```
100
101 void insertLast(listinduk &L, address P) {
102     if(listEmpty(L)) {
103         L.first = P;
104         L.last = P;
105     } else {
106         L.last->next = P;
107         P->prev = L.last;
108         L.last = P;
109     }
110 }
111 void insertAfter(listinduk &L, address Prec, address P) {
112     if(Prec == Nil) {
113         P->next = Prec->next;
114         P->prev = Prec;
115         if(Prec->next != Nil){
116             Prec->next->prev = P;
117         }
118         Prec->next = P;
119         if(Prec == L.last){
120             L.last = P;
121         }
122     }
123 }
124 }
125 void delFirst(listinduk &L, address &P) {
126     P = L.first;
127     if(L.first == L.last) {
```

File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
1(C Windows (CRLF) UTF-8

File Edit Selection View Go Run ... ← → Q: laprak pekan 11

EXPLORER ... C: multilist.h C: multilist.cpp X C: mainmultilist.cpp

OPEN EDITORS C: multilist.h X C: multilist.cpp C: mainmultilist.cpp

LAPRAK PEKAN 11

OUTLINE

```

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 void dellast(listinduk &L, address &P) {
138     P = L.last;
139     if(L.first == L.last) {
140         L.first = Nil;
141         L.last = Nil;
142     }else{
143         L.last = P->prev;
144         L.last->next = Nil;
145     }
146     P->prev = Nil;
147 }

148 void delAfter(listinduk &L, address &P, address Prec) {
149     P = Prec->next;
150     if(P != Nil) {
151         Prec->next = P->next;
152         if(P->next != Nil) {
153             P->next->prev = Prec;
154         }else{
155             L.last = Prec;
156         }
157         P->next = Nil;
158         P->prev = Nil;
159     }
160 }
```

File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
16 Windows (CRLF) UTF-8

File Edit Selection View Go Run ... ← → Q: laprak pekan 11

EXPLORER ... C: multilist.h C: multilist.cpp X C: mainmultilist.cpp

OPEN EDITORS C: multilist.h X C: multilist.cpp C: mainmultilist.cpp

LAPRAK PEKAN 11

OUTLINE

```

161 }
162 }
163 void delP(listinduk &L, infotypeinduk X) {
164     address P = findElm(L, X);
165     if(P != Nil) {
166         if(P == L.first) {
167             delFirst(L, P);
168         } else if(P == L.last){
169             dellast(L, P);
170         }else{
171             address Q;
172             delAfter(L, Q, P->prev);
173         }
174         dealokasi(P);
175     }
176 }
177 }

178 void insertFirstAnak(listanak &L, address_anak P) {
179     if(ListEmptyAnak(L)) {
180         L.first = P;
181         L.last = P;
182     }else{
183         P->next = L.first;
184         L.first->prev = P;
185         L.first = P;
186     }
187 }

188 void insertLastAnak(listanak &L, address_anak P) {
189     if(ListEmptyAnak(L)) {
190         L.first = P;
191     }
192 }
```

File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
16 Windows (CRLF) UTF-8

Screenshot of a code editor showing a tooltip for a variable. The tooltip displays the variable's name and value: Nama: Muhammad Azzam Satria and NIM: 103112400112. The code in the editor is related to a linked list manipulation function.

```
193     L.last = P;
194 } else{
195     L.last->next = P;
196     P->prev = L.last;
197     L.last = P;
198 }
199 }

200 void delFirstAnak(listanak &L, address_anak &P) {
201     P = L.first;
202     if(L.first == L.last) {
203         L.first = Nil;
204         L.last = Nil;
205     } else{
206         L.first = P->next;
207         L.first->prev = Nil;
208     }
209     P->next = Nil;
210 }
211 }

212 void dellastAnak(listanak &L, address_anak &P) {
213     P = L.last;
214     if(L.first == L.last) {
215         L.first = Nil;
216         L.last = Nil;
217     } else{
218         L.last = P->prev;
219         L.last->next = Nil;
220     }
221     P->prev = Nil;
222 }
223 }
```

Screenshot of a code editor showing a tooltip for a variable. The tooltip displays the variable's name and value: Name: Muhammad Azzam Satria and NIM : 103112400112. The code in the editor is related to a linked list manipulation function.

```
225 void delAfterAnak(listanak &L, address_anak &P, address_anak Prec) {
226     P = Prec->next;
227     if(P != Nil) {
228         Prec->next = P->next;
229         if(P->next != Nil) {
230             P->next->prev = Prec;
231         } else{
232             L.last = Prec;
233         }
234         P->next = Nil;
235         P->prev = Nil;
236     }
237 }

238 void delPAnak(listanak &L, infotypeanak X) {
239     address_anak P = findElm(L, X);
240     if(P != Nil){
241         if(P == L.first) {
242             delFirstAnak(L, P);
243         } else if(P == L.last) {
244             dellastAnak(L, P);
245         } else{
246             address_anak Q;
247             delAfterAnak(L, Q, P->prev);
248         }
249         dealokasiAnak(P);
250 }
```

The screenshot shows the Visual Studio Code interface. In the center, the code editor displays a C++ file named `multilist.cpp`. The code implements a multilist structure with functions for printing information and managing nodes. Below the code editor is a terminal window showing the command-line interface for running the program.

```

File Edit Selection View ... < > laprak pekan 11
EXPLORER multilist.h multilist.cpp mainmultilist.cpp
OPEN EDITORS
LAPRAK PEKAN 11
OUTLINE
251 }
252 }
253
254 void printInfo(listinduk L) {
255     address P = L.first;
256     while(P != Nil) {
257         cout << "Induk: " << P->info << endl;
258         address_anak C = P->anak.first;
259         while(C != Nil) {
260             cout << "Anak: " << C->info << endl;
261             C = C->next;
262         }
263         P = P->next;
264     }
265 }
266
  nama & nim - ...
  File Edit Format View Help
  Nama: Muhammad Azzam Satria
  NIM : 103112400112
  1C Windows (CRLF) UTF-8

```

Screenshots Output

The terminal window shows the execution of the program. It starts with the command `g++ mainmultilist.cpp multilist.cpp`, followed by the output of the program itself. The program prints the information for each node in the list, including the value of the node and its child node.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL powershell + 
PS C:\Users\USER\OneDrive\Dokumen\Kuliah\Semester 3\Praktikum Struktur Data\laprak pekan 11> g++ mainmultilist.cpp multilist.cpp
PS C:\Users\USER\OneDrive\Dokumen\Kuliah\Semester 3\Praktikum Struktur Data\laprak pekan 11> .\a.exe
Induk: 10
Anak: 101
Anak: 102

Induk: 20
Anak: 201

Induk: 30

PS C:\Users\USER\OneDrive\Dokumen\Kuliah\Semester 3\Praktikum Struktur Data\laprak pekan 11> []
  nama & nim - ...
  File Edit Format View Help
  Nama: Muhammad Azzam Satria
  NIM : 103112400112
  1C Windows (CRLF) UTF-8

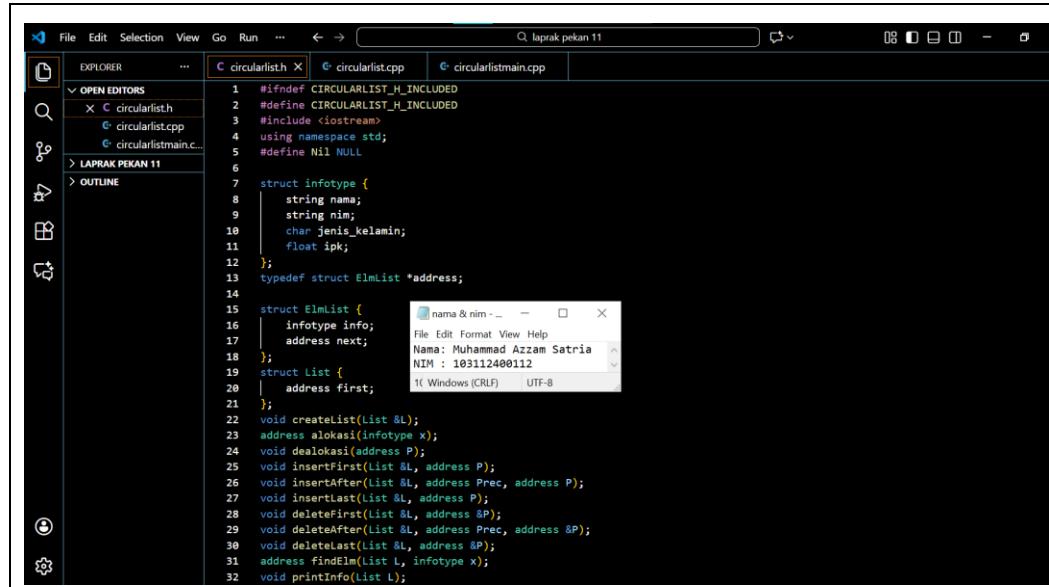
```

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



The screenshot shows a code editor interface with two files open: circularlist.cpp and circularlistmain.cpp. The circularlist.cpp file contains C++ code for a circular linked list. The code defines an infotype struct with fields for name, NIM, gender, and IPK. It also defines ElmList and List structures, and various functions for creating the list, inserting nodes at the beginning or after a specific node, deleting nodes from the beginning or after a specific node, and printing the information of a node. A tooltip window is visible over the circularlistmain.cpp file, showing the name and NIM of a student named Muhammad Azzam Satria with NIM: 103112400112. The code editor has a dark theme and includes an Explorer sidebar on the left.

```
#ifndef CIRCULARLIST_H_INCLUDED
#define CIRCULARLIST_H_INCLUDED
#include <iostream>
using namespace std;
#define Nil NULL

struct infotype {
    string nama;
    string nim;
    char jenis_kelamin;
    float ipk;
};

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 insertAfter(List &L, address Prec, address P);
void insertLast(List &L, address P);
void deleteFirst(List &L, address &P);
void deleteAfter(List &L, address Prec, address &P);
void deleteLast(List &L, address &P);
address findElm(List L, infotype x);
void printInfo(List L);
```

The image shows two side-by-side windows of a code editor, likely Visual Studio Code, displaying C++ code for a circular linked list. Both windows have the same title bar: "laprak pekan 11".

Top Window (Editor 1):

```
#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;
    }
}
```

A small terminal window is open in the bottom right corner of this window, showing the output of a command-line application. It displays:

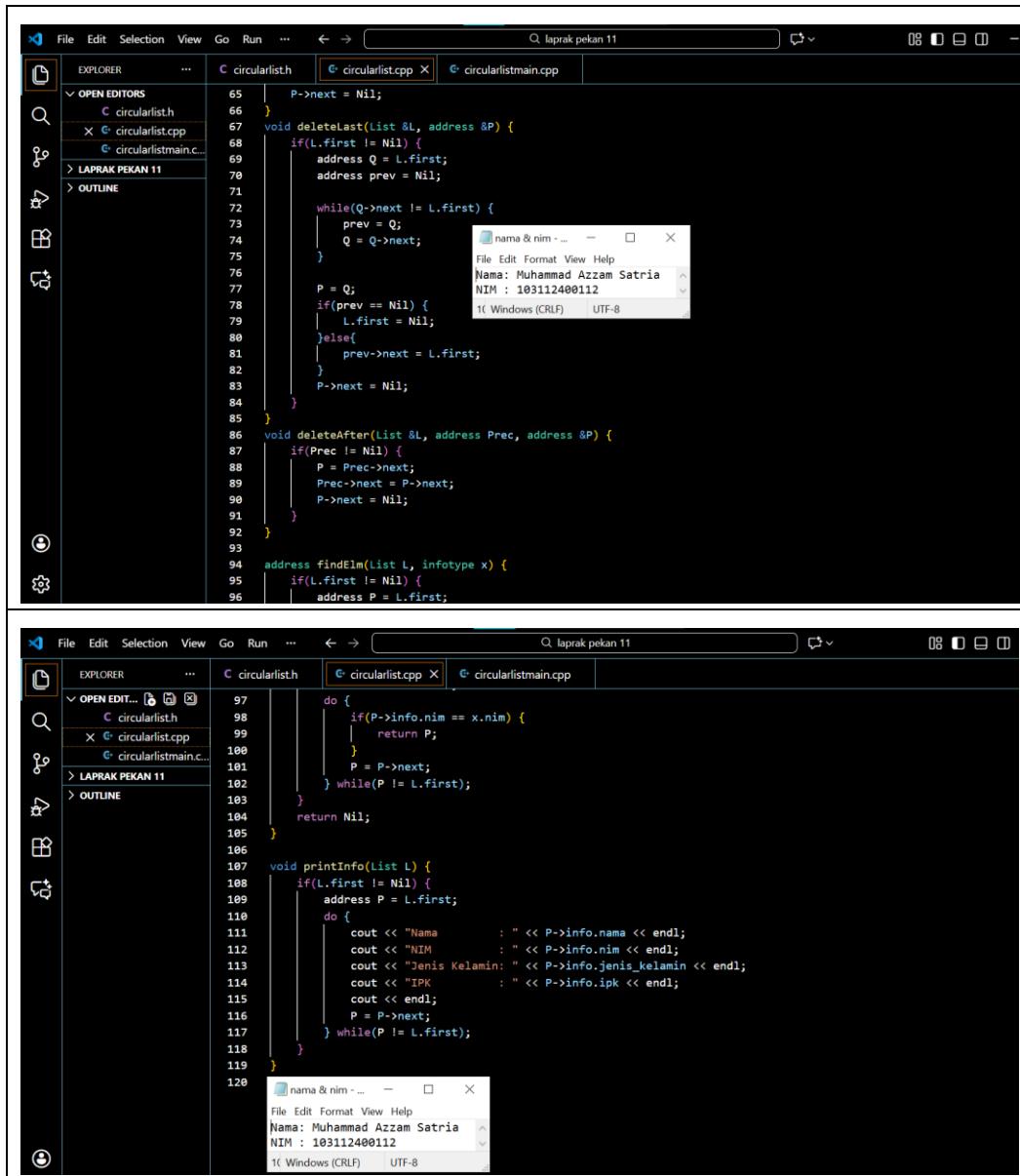
```
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
1(C:\Windows (CRLF)) UTF-8
```

Bottom Window (Editor 2):

```
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;
    }
}
```

A small terminal window is open in the bottom right corner of this window, showing the output of a command-line application. It displays:

```
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
1(C:\Windows (CRLF)) UTF-8
```



File Edit Selection View Go Run ... ← → Q: laprak pekan 11

EXPLORER OPEN EDITORS circularlisth circularlist.cpp circularlistmain.c...

LAPRAK PEKAN 11 OUTLINE

```
65     P->next = Nil;
66 }
67 void deleteLast(List &L, address &P) {
68     if(L.first != Nil) {
69         address Q = L.first;
70         address prev = Nil;
71
72         while(Q->next != L.first) {
73             prev = Q;
74             Q = Q->next;
75         }
76
77         P = Q;
78         if(prev == Nil) {
79             L.first = Nil;
80         } else{
81             prev->next = L.first;
82         }
83         P->next = Nil;
84     }
85 }
86 void deleteAfter(List &L, address Prec, address &P) {
87     if(Prec != Nil) {
88         P = Prec->next;
89         Prec->next = P->next;
90         P->next = Nil;
91     }
92 }
93 address findElm(List L, infotype x) {
94     if(L.first != Nil) {
95         address P = L.first;
```

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

File Edit Selection View Go Run ... ← → Q: laprak pekan 11

EXPLORER OPEN EDIT... circularlisth circularlist.cpp circularlistmain.cpp...

LAPRAK PEKAN 11 OUTLINE

```
97     do {
98         if(P->info.nim == x.nim) {
99             return P;
100        P = P->next;
101    } while(P != L.first);
102
103    return Nil;
104 }
105
106 void printInfo(List L) {
107     if(L.first != Nil) {
108         address P = L.first;
109         do {
110             cout << "Nama : " << P->info.nama << endl;
111             cout << "NIM : " << P->info.nim << endl;
112             cout << "Jenis Kelamin: " << P->info.jenis_kelamin << endl;
113             cout << "IPK : " << P->info.ipk << endl;
114             cout << endl;
115             P = P->next;
116         } while(P != L.first);
117     }
118 }
119
120
```

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

laprak pekan 11

circularlist.h circularlist.cpp circularlistmain.cpp

```

1 #include <iostream>
2 #include "circularlist.h"
3 using namespace std;
4
5 address createData(string nama, string nim, char jk, float ipk) {
6     infotype x;
7     x.nama = nama;
8     x.nim = nim;
9     x.jenis_kelamin = jk;
10    x.ipk = ipk;
11    return alokasi(x);
12 }

13
14 int main() {
15     List L;
16     address P, Prec;
17     infotype x;
18
19     cout << "Coba insert first, last, dan after" << endl;
20     P = createData("Danu", "04", 'l', 4.0);
21     insertFirst(L, P);
22
23     P = createData("Fahmi", "06", 'l', 3.45);
24     insertLast(L, P);
25
26 }

```


laprak pekan 11

circularlist.h circularlist.cpp circularlistmain.cpp

```

27 P = createData("Bobi", "02", 'l', 3.71);
28 insertFirst(L, P);
29
30 P = createData("Ali", "01", 'l', 3.3);
31 insertFirst(L, P);
32
33 P = createData("Gita", "07", 'p', 3.75);
34 insertLast(L, P);
35
36 x.nim = "07";
37 Prec = findElm(L, x);
38 P = createData("Cindi", "03", 'p', 3.5);
39 insertAfter(L, Prec, P);
40
41 x.nim = "02";
42 Prec = findElm(L, x);
43 P = createData("Hilmi", "08", 'l', 3.3);
44 insertAfter(L, Prec, P);
45
46 x.nim = "04";
47 Prec = findElm(L, x);
48 P = createData("Eli", "05", 'p', 3.4);
49 insertAfter(L, Prec, P);
50 printInfo(L);
51
52 }

```

Screenshots output

```

PS C:\Users\USER\OneDrive\Kuliah\Semester 3\Praktikum Struktur Data\laprak pekan 11> .\a.exe
Coba insert first, last, dan after
Nama : Ali
NIM : 01
Jenis Kelamin: l
IPK : 3.3
Nama : Bobi
NIM : 02
Jenis Kelamin: l
IPK : 3.71

Nama : Hilmi
NIM : 08
Jenis Kelamin: l
IPK : 3.3

Nama : Danu
NIM : 04
Jenis Kelamin: l
IPK : 4

Nama : Eli
NIM : 05
Jenis Kelamin: p
IPK : 3.4

Nama : Fahmi
NIM : 06
Jenis Kelamin: l
IPK : 3.45

Nama : Gita
NIM : 07
Jenis Kelamin: p
IPK : 3.75

Nama : Cindi
NIM : 03
Jenis Kelamin: p
IPK : 3.5

  File Edit Format View Help
  Nama: Muhammad Azzam Satria
  NIM : 103112400112
  1( Windows (CRLF)  UTF-8

```

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 mengarah ke elemen selanjutnya. Struct List hanya mempunyai pointer first yang fungsinya sebagai penanda awal list. Pada file header 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 mencocokkan 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 posisi 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.