

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

MODUL VII

MULTI LINKED LIST



Disusun Oleh :

NAMA : Muhammad Fachri Auravyano Saka
NIM : 103112430180

Dosen
FAHRUDIN MUKTI WIBOWO

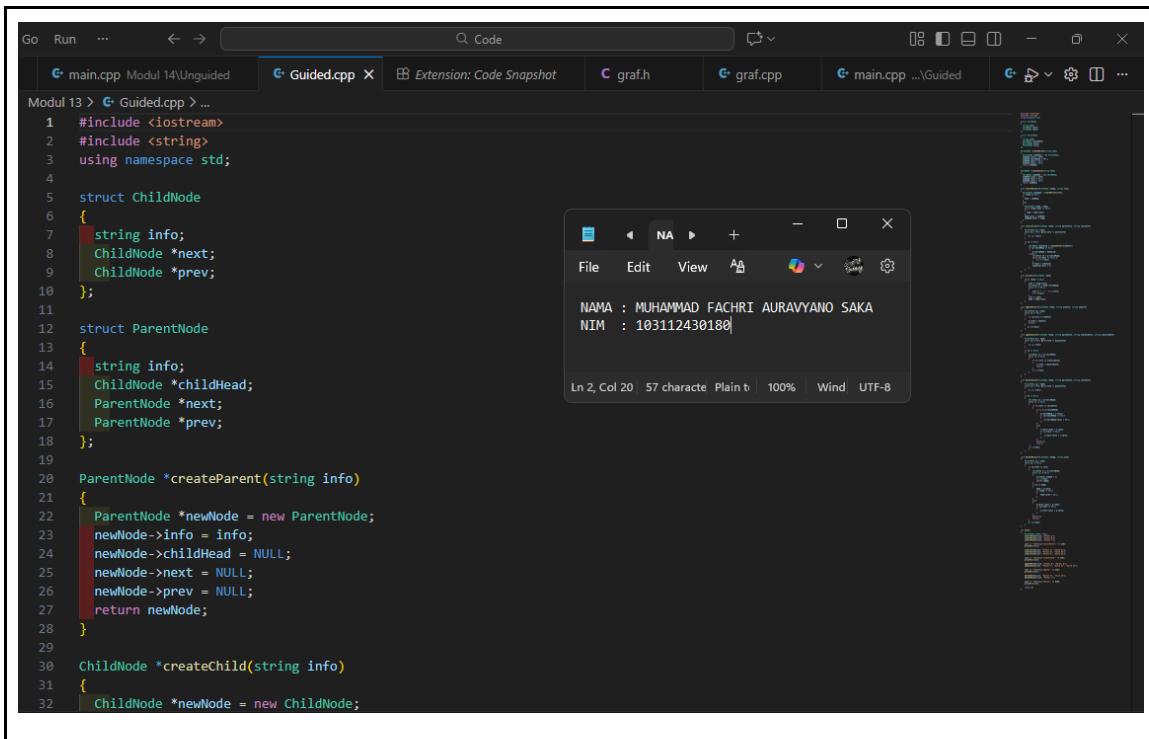
**PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

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.

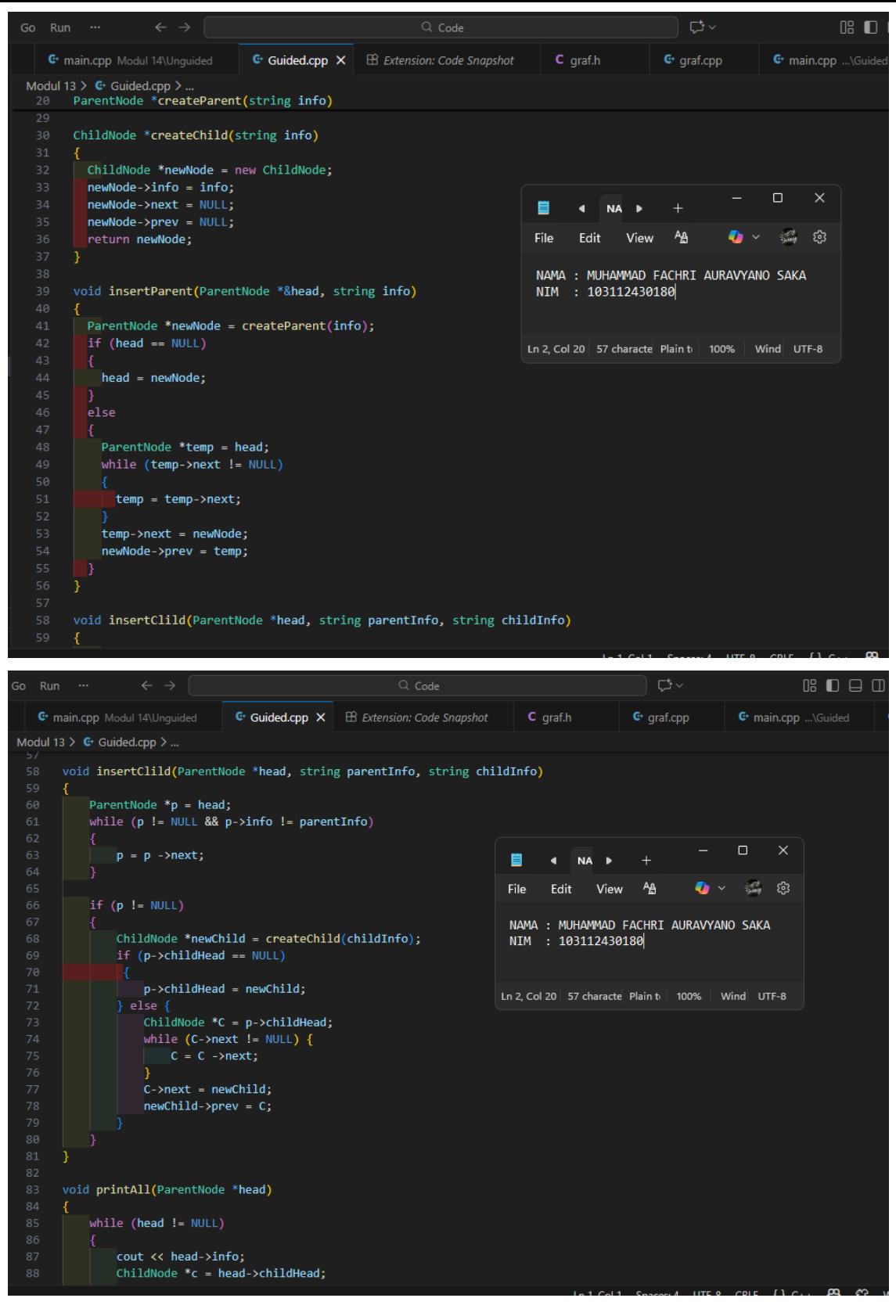
B. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1



```
Go Run ... ← → Q: Code
Modul 13 > Guided.cpp ...
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 struct ChildNode
6 {
7     string info;
8     ChildNode *next;
9     ChildNode *prev;
10 };
11
12 struct ParentNode
13 {
14     string info;
15     ChildNode *childHead;
16     ParentNode *next;
17     ParentNode *prev;
18 };
19
20 ParentNode *createParent(string info)
21 {
22     ParentNode *newNode = new ParentNode;
23     newNode->info = info;
24     newNode->childHead = NULL;
25     newNode->next = NULL;
26     newNode->prev = NULL;
27     return newNode;
28 }
29
30 ChildNode *createChild(string info)
31 {
32     ChildNode *newNode = new ChildNode;
```

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180



Code Editor Screenshot:

Top Bar: Go Run ... ← → ⌂ Q Code Extension: Code Snapshot

Left Window (Modul 13):

```
20 ParentNode *createParent(string info)
21
22 ChildNode *createChild(string info)
23 {
24     ChildNode *newNode = new ChildNode;
25     newNode->info = info;
26     newNode->next = NULL;
27     newNode->prev = NULL;
28     return newNode;
29 }
30
31 void insertParent(ParentNode *&head, string info)
32 {
33     ParentNode *newNode = createParent(info);
34     if (head == NULL)
35     {
36         head = newNode;
37     }
38     else
39     {
40         ParentNode *temp = head;
41         while (temp->next != NULL)
42         {
43             temp = temp->next;
44         }
45         temp->next = newNode;
46         newNode->prev = temp;
47     }
48 }
49
50 void insertChild(ParentNode *head, string parentInfo, string childInfo)
51 {
```

Right Window (Code Snapshot):

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

Bottom Bar: Line 2, Col 20 | 57 characters | Plain text | 100% | Wind | UTF-8

Code Editor Screenshot:

Top Bar: Go Run ... ← → ⌂ Q Code Extension: Code Snapshot

Left Window (Modul 14):

```
58 void insertChild(ParentNode *head, string parentInfo, string childInfo)
59 {
60     ParentNode *p = head;
61     while (p != NULL && p->info != parentInfo)
62     {
63         p = p ->next;
64     }
65
66     if (p != NULL)
67     {
68         ChildNode *newChild = createChild(childInfo);
69         if (p->childHead == NULL)
70         {
71             p->childHead = newChild;
72         } else {
73             ChildNode *C = p->childHead;
74             while (C->next != NULL) {
75                 C = C ->next;
76             }
77             C->next = newChild;
78             newChild->prev = C;
79         }
80     }
81 }
82
83 void printAll(ParentNode *head)
84 {
85     while (head != NULL)
86     {
87         cout << head->info;
88         ChildNode *c = head->childHead;
```

Right Window (Code Snapshot):

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

Bottom Bar: Line 2, Col 1 | 57 characters | Plain text | 100% | Wind | UTF-8

Code Editor Screenshot:

Modul 13 > Guided.cpp > ...

```
58 void insertChild(ParentNode *head, string parentInfo, string childInfo)
59
60 void printAll(ParentNode *head)
61 {
62     while (head != NULL)
63     {
64         cout << head->info;
65         ChildNode *c = head->childHead;
66         while (c != NULL)
67         {
68             cout << " -> " << c->info;
69             c = c->next;
70         }
71         cout << endl;
72         head = head->next;
73     }
74 }
75
76 void updateParent(ParentNode *head, string oldInfo, string newInfo)
77 {
78     ParentNode *p = head;
79     while (p != NULL)
80     {
81         if (p->info == oldInfo)
82         {
83             p->info = newInfo;
84             return;
85         }
86         p = p->next;
87     }
88 }
```

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)
```

Code Snapshot Modal:

File Edit View AA NA + - X

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

Ln 2, Col 20 | 57 characters | Plain text | 100% | Window | UTF-8

Modul 13 > Guided.cpp > ...

```
136 void deleteChild(ParentNode *head, string parentInfo, string childInfo)
143 {
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                     delete c;
167                     return;
168                 }
169             }
170             c = c->next;
171         }
172     }
173 }
```

Modul 13 > Guided.cpp > ...

```
175 void deleteParent(ParentNode *head, string info)
176 {
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;
207         }
208     }
209 }
```

Modul 13 > Guided.cpp > ...

```

175 void deleteParent(ParentNode *&head, string info)
176     while (p != NULL)
177         {
178             p = p->next;
179         }
180     }
181
182 int main()
183 {
184     ParentNode *list = NULL;
185     insertParent(list, "Parent A");
186     insertParent(list, "Parent B");
187     insertParent(list, "Parent C");
188
189     cout << "\nSetelah InsertParent:" << endl;
190     printAll(list);
191
192     insertChild(list, "Parent A", "Child A1");
193     insertChild(list, "Parent A", "Child A2");
194     insertChild(list, "Parent B", "Child B1");
195
196     cout << "\nSetelah InsertChild:" << endl;
197     printAll(list);
198
199     updateParent(list, "Parent B", "Parent B");
200     updateChild(list, "Parent A", "Child A1", "Child A1");
201
202     cout << "\nSetelah Update:" << endl;
203     printAll(list);
204
205     deleteChild(list, "Parent A", "Child A2");
206     deleteParent(list, "Parent C");
207
208     cout << "\nSetelah Delete:" << endl;
209     printAll(list);
210
211     return 0;
212 }
```

Modul 13 > Guided.cpp > ...

```

220
221     insertChild(list, "Parent A", "Child A1");
222     insertChild(list, "Parent A", "Child A2");
223     insertChild(list, "Parent B", "Child B1");
224
225     cout << "\nSetelah InsertChild:" << endl;
226     printAll(list);
227
228     updateParent(list, "Parent B", "Parent B");
229     updateChild(list, "Parent A", "Child A1", "Child A1");
230
231     cout << "\nSetelah Update:" << endl;
232     printAll(list);
233
234     deleteChild(list, "Parent A", "Child A2");
235     deleteParent(list, "Parent C");
236
237     cout << "\nSetelah Delete:" << endl;
238     printAll(list);
239
240     return 0;
241 }
```

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

Ln 2, Col 20 57 character Plain text 100% Wind UTF-8

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

Ln 2, Col 20 57 character Plain text 100% Wind UTF-8

Screenshots Output

```
PS E:\KULIAH\SEMESTER 3\Struktur Data\Code> cd "e:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 13\" ; if ($?) { g++ Guided.cpp -o Guided }

Setelah InsertParent:
Parent A
Parent B
Parent C

Setelah InsertChild:
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 E:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 13>
```

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 &L)
1 #include "circularlist.h"
2 #include "circularlist.cpp"
3 #include <iostream>
4 using namespace std;
5
6 address createData(string nama, string nim, char jk, float ipk)
7 {
8     infotype x;
9     x.nama = nama;
10    x.nim = nim;
11    x.jenis_kelamin = jk;
12    x.ipk = ipk;
13    return alokasi(x);
14 }
15
16 void sortList(List &L) {
17     if (L.first == Nil) {
18         return;
19     }
20
21     address P = L.first;
22     address Q;
23     infotype temp;
24
25     do {
26         Q = P->next;
27         while (Q != L.first) {
28             if (P->info.nim > Q->info.nim) {
29                 temp = P->info;
30                 P->info = Q->info;
31                 Q->info = temp;
32             }
33         }
34     } while (P->next != L.first);
35
36 }
37
38 int main()
39 {
40     List L;
41     address P1 = Nil;
42     address P2 = Nil;
43     infotype x;
44
45     createList(L);
46
47     cout << "coba insert first, last, dan after" << endl;
48
49     P1 = createData("Danu", "04", 'l', 4.0);
50     insertFirst(L, P1);
51
52     P1 = createData("Fahmi", "06", 'l', 3.45);
53     insertLast(L, P1);
54
55     P1 = createData("Bobi", "02", 'l', 3.71);
56     insertFirst(L, P1);
57
58     P1 = createData("Ali", "01", 'l', 3.3);
59 }
```

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

```
Modul 13 > Unguided > main.cpp > sortList(List &L)
16 void sortlist(List &L) {
17     do {
18         while (Q != L.first) {
19             if (P->info.nim > Q->info.nim) {
20                 }
21             Q = Q->next;
22         }
23         P = P->next;
24     } while (P->next != L.first);
25
26 }
27
28 int main()
29 {
30     List L;
31     address P1 = Nil;
32     address P2 = Nil;
33     infotype x;
34
35     createList(L);
36
37     cout << "coba insert first, last, dan after" << endl;
38
39     P1 = createData("Danu", "04", 'l', 4.0);
40     insertFirst(L, P1);
41
42     P1 = createData("Fahmi", "06", 'l', 3.45);
43     insertLast(L, P1);
44
45     P1 = createData("Bobi", "02", 'l', 3.71);
46     insertFirst(L, P1);
47
48     P1 = createData("Ali", "01", 'l', 3.3);
49 }
```

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

```
main.cpp
Modul 13 > Unguided > main.cpp > sortList(List &L)
53     P1 = createData("Fahmi", "06", 'I', 3.45);
54     insertLast(L, P1);
55
56     P1 = createData("Bobi", "02", 'I', 3.71);
57     insertFirst(L, P1);
58
59     P1 = createData("Ali", "01", 'I', 3.3);
60     insertFirst(L, P1);
61
62     P1 = createData("Gita", "07", 'P', 3.75);
63     insertLast(L, P1);
64
65     x.nim = "07";
66     P1 = findElm(L, x);
67     P2 = createData("Cindi", "03", 'P', 3.5);
68     insertAfter(L, P1, P2);
69
70     x.nim = "02";
71     P1 = findElm(L, x);
72     P2 = createData("Hilmi", "08", 'I', 3.3);
73     insertAfter(L, P1, P2);
74
75     x.nim = "04";
76     P1 = findElm(L, x);
77     P2 = createData("Eli", "05", 'P', 3.4);
78     insertAfter(L, P1, P2);
79
80     sortList(L);
81
82     printInfo(L);
83
84     return 0;

```

circularlist.cpp

```
circularlist.cpp
Modul 13 > Unguided > circularlist.cpp > ...
1 #include "circularlist.h"
2 #include <iostream>
3 using namespace std;
4
5 void createList(List &L) {
6     L.first = Nil;
7 }
8
9 address alokasi(infotype x) {
10     address P = new ElmList;
11     P->info = x;
12     P->next = Nil;
13     return P;
14 }
15
16 void dealokasi(address &P) {
17     delete P;
18 }
19
20 void insertFirst(List &L, address P) {
21     if (L.first == Nil) {
22         P->next = P;
23         L.first = P;
24     } else {
25         address Q = L.first;
26         while (Q->next != L.first) {
27             Q = Q->next;
28         }
29         P->next = L.first;
30         Q->next = P;
31         L.first = P;
32     }

```

The image shows two screenshots of a code editor interface, likely Visual Studio Code, demonstrating the implementation of a circular linked list.

Screenshot 1: The left pane displays the `circularlist.cpp` file with C++ code. The right pane shows a terminal window with the following output:

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

Screenshot 2: The left pane displays the `circularlist.cpp` file with C++ code. The right pane shows a terminal window with the following output:

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

Both screenshots show the same code and terminal output, indicating a successful run of the program.

```
35 void insertAfter(List &L, address Prec, address P) {
36     P->next = Prec->next;
37     Prec->next = P;
38 }
39
40 void insertLast(List &L, address P) {
41     if (L.first == Nil) {
42         P->next = P;
43         L.first = P;
44     } else {
45         address Q = L.first;
46         while (Q->next != L.first) {
47             Q = Q->next;
48         }
49         P->next = L.first;
50         Q->next = P;
51     }
52 }
53
54 void deleteFirst(List &L, address &P) {
55     if (L.first != Nil) {
56         P = L.first;
57         if (P->next == L.first) {
58             L.first = Nil;
59         } else {
60             address Q = L.first;
61             while (Q->next != L.first) {
62                 Q = Q->next;
63             }
64             L.first = P->next;
65             Q->next = L.first;
66         }
67     }
68 }
69
70 void deleteAfter(List &L, address Prec, address &P) {
71     if (Prec != Nil) {
72         P = Prec->next;
73         if (P == L.first && P->next == L.first) {
74             L.first = Nil;
75         } else {
76             Prec->next = P->next;
77             if (P == L.first) {
78                 L.first = P->next;
79             }
80         }
81         P->next = Nil;
82     }
83 }
84
85 void deletelast(List &L, address &P) {
86     if (L.first != Nil) {
87         if (L.first->next == L.first) {
88             P = L.first;
89             L.first = Nil;
90         } else {
91             address Q = L.first;
92             while (Q->next->next != L.first) {
93                 Q = Q->next;
94             }
95             P = Q->next;
96             Q->next = Nil;
97         }
98     }
99 }
```

```
Modul 13 > Unguided > circularlist.cpp > ...
86 void deleteLast(List &L, address &P) {
87     if (L.first != Nil) {
88         } else {
89             address Q = L.first;
90             while (Q->next->next != L.first) {
91                 Q = Q->next;
92             }
93             P = Q->next;
94             Q->next = L.first;
95         }
96         P->next = Nil;
97     }
98 }
99
100 }

101 address findElm(List L, infotype x) {
102     if (L.first == Nil) {
103         return Nil;
104     }
105
106     address P = L.first;
107     do {
108         if (P->info.nim == x.nim) {
109             return P;
110         }
111         P = P->next;
112     } while (P != L.first);
113
114     return Nil;
115 }
116
117 }

118 void printInfo(List L) {
119     if (L.first == Nil) {
120         cout << "List kosong" << endl;
121     } else {
122         address P = L.first;
123         do {
124             cout << "Nama : " << P->info.nama << endl;
125             cout << "NIM : " << P->info.nim << endl;
126             cout << "L/P : " << P->info.jenis_kelamin << endl;
127             cout << "IPK : " << P->info.ipk << endl;
128             cout << endl;
129             P = P->next;
130         } while (P != L.first);
131     }
132 }
```

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

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

circularlist.h

circularlist.h

```

Modul 13 > Unguided > C circularlist.h > ...
1 #ifndef CIRCULARLIST_H_INCLUDED
2 #define CIRCULARLIST_H_INCLUDED
3
4 #include <string>
5 using namespace std;
6
7 #define Nil NULL
8
9 struct infotype {
10     string nama;
11     string nim;
12     char jenis_kelamin;
13     float ipk;
14 };
15
16 typedef struct ElmList *address;
17
18 struct ElmList {
19     infotype info;
20     address next;
21 };
22
23 struct List {
24     address first;
25 };
26
27 void createList(List &L);
28
29 address alokasi(infotype x);
30 void dealokasi(address &P);
31
32 void insertFirst(List &L, address P);

```

circularlist.cpp

```

Modul 13 > Unguided > C circularlist.h > ...
1 #ifndef CIRCULARLIST_H_INCLUDED
23 struct List {
24     address first;
25 };
26
27 void createList(List &L);
28
29 address alokasi(infotype x);
30 void dealokasi(address &P);
31
32 void insertFirst(List &L, address P);
33 void insertAfter(List &L, address Prec, address P);
34 void insertLast(List &L, address P);
35
36 void deleteFirst(List &L, address &P);
37 void deleteAfter(List &L, address Prec, address &P);
38 void deleteLast(List &L, address &P);
39
40 address findElm(List L, infotype x);
41
42 void printInfo(List L);
43
44 #endif

```

Screenshot Output

```

PS E:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 13\Unguided> cd "e:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 13\Unguided\" ; if ($?) { echo "coba insert first, last, dan after" }
coba insert first, last, dan after
Nama : Ali
NIM : 01
L/P : l
IPK : 3.3

Nama : Bobi
NIM : 02
L/P : l
IPK : 3.71

Nama : Cindi
NIM : 03
L/P : p
IPK : 3.5

Nama : Danu
NIM : 04
L/P : l
IPK : 4

Nama : Eli
NIM : 05
L/P : p
IPK : 3.4

Nama : Fahmi
NIM : 06
L/P : l
IPK : 3.45

Nama : Gita
NIM : 07
L/P : p
IPK : 3.75

Nama : Hilmi
NIM : 08
L/P : l
IPK : 3.3

IPK : 4

Nama : Eli
NIM : 05
L/P : p
IPK : 3.4

Nama : Fahmi
NIM : 06
L/P : l
IPK : 3.45

Nama : Gita
NIM : 07
L/P : p
IPK : 3.75

Nama : Hilmi
NIM : 08
L/P : l
IPK : 3.3

```

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

Ln 2, Col 20 | 57 character | Plain t | 100% | Wind | UTF-8

LN 70, Col 18 | Spaces 4 | UTF-8 | CR LF | { } | C | B | S | C |

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.