

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

MODUL VII

MULTI LINKED LIST



Disusun Oleh :

NAMA : Muhammad Fachri Auravyano Saka
NIM : 103112430180

Dosen

FAHRUDIN MUKTI WIBOWO

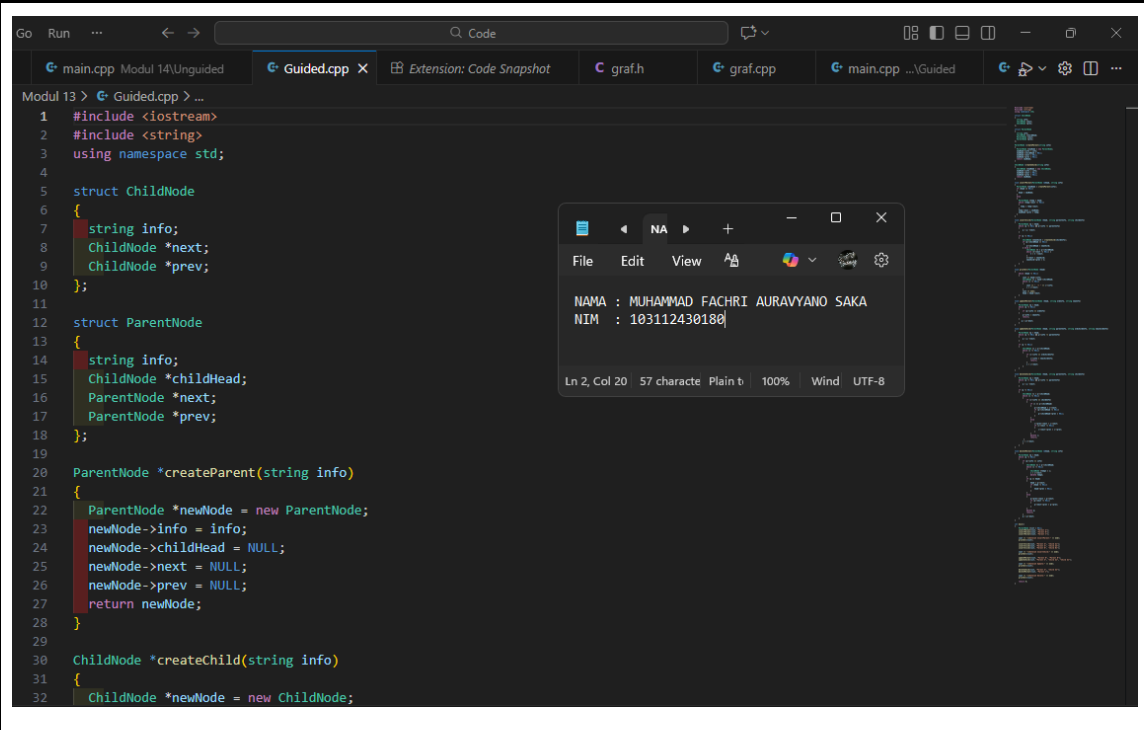
**PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

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



```
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;
```

File Edit View          

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

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

```
Go Run ... < -> Code
main.cpp Modul 14\Unguided Guided.cpp X Extension: Code Snapshot graf.h graf.cpp main.cpp ...\Guided
Modul 13 > Guided.cpp > ...
28 ParentNode *createParent(string info)
29
30 ChildNode *createChild(string info)
31 {
32     ChildNode *newNode = new ChildNode;
33     newNode->info = info;
34     newNode->next = NULL;
35     newNode->prev = NULL;
36     return newNode;
37 }
38
39 void insertParent(ParentNode *&head, string info)
40 {
41     ParentNode *newNode = createParent(info);
42     if (head == NULL)
43     {
44         head = newNode;
45     }
46     else
47     {
48         ParentNode *temp = head;
49         while (temp->next != NULL)
50         {
51             temp = temp->next;
52         }
53         temp->next = newNode;
54         newNode->prev = temp;
55     }
56 }
57
58 void insertClild(ParentNode *head, string parentInfo, string childInfo)
59 {
```

```
File Edit View A+ 100% Wind UTF-8
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 character Plain t
```

```
Go Run ... < -> Code
main.cpp Modul 14\Unguided Guided.cpp X Extension: Code Snapshot graf.h graf.cpp main.cpp ...\Guided
Modul 13 > Guided.cpp > ...
58 void insertClild(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;
```

```
File Edit View A+ 100% Wind UTF-8
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 character Plain t
```

```
Go Run ... < > Code
main.cpp Modul 14\Unguided Guided.cpp X Extension: Code Snapshot graf.h graf.cpp main.cpp ...\Guided

Modul 13 > Guided.cpp > ...
58 void insertChild(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
```

File Edit View

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

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

```
Go Run ... < > Code
main.cpp Modul 14\Unguided Guided.cpp X 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)
```

File Edit View

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

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

```
Go Run ... < -> Code
main.cpp Modul 14\Unguided Guided.cpp X Extension: Code Snapshot graf.h graf.cpp main.cpp ...\Guided

Modul 13 > Guided.cpp > ...
136 void deleteChild(ParentNode *head, string parentInfo, string childInfo)
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                 else
161                 {
162                     c->prev->next = c->next;
163                     if (c->next != NULL)
164                     {
165                         c->next->prev = c->prev;
166                     }
167                 }
168                 delete c;
169                 return;
170             }
171             c = c->next;
172         }
173     }
```

File Edit View A+ 100% Wind UTF-8

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

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

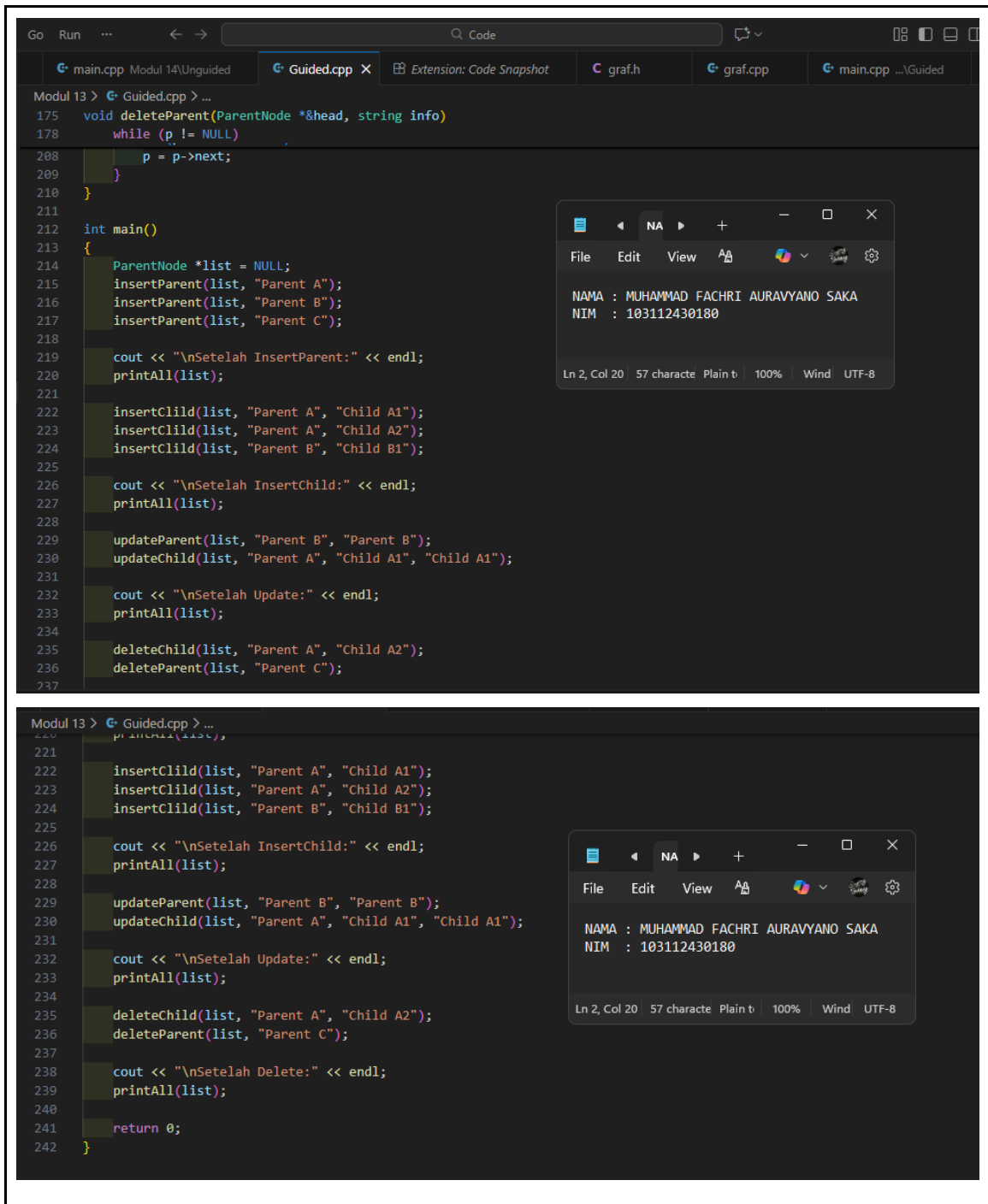
```
Go Run ... < -> Code
main.cpp Modul 14\Unguided Guided.cpp X Extension: Code Snapshot graf.h graf.cpp main.cpp ...\Guided

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         }
```

File Edit View A+ 100% Wind UTF-8

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

Ln 2, Col 20 57 character Plain t 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>
```

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

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 pointer-nya 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

```
Go Run ... < -> Code
main.cpp x
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         P = P->next;
35     } while (P->next != L.first);
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", '1', 4.0);
50     insertFirst(L, P1);
51
52     P1 = createData("Fahmi", "06", '1', 3.45);
53     insertLast(L, P1);
54
55     P1 = createData("Bobi", "02", '1', 3.71);
56     insertFirst(L, P1);
57
58     P1 = createData("Ali", "01", '1', 3.3);
59 }
```

NA

File Edit View

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

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

```
Go Run ... < -> Code
main.cpp x
Modul 13 > Unguided > main.cpp > sortList(List &L)
16 void sortList(List &L) {
25     do {
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         P = P->next;
35     } while (P->next != L.first);
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", '1', 4.0);
50     insertFirst(L, P1);
51
52     P1 = createData("Fahmi", "06", '1', 3.45);
53     insertLast(L, P1);
54
55     P1 = createData("Bobi", "02", '1', 3.71);
56     insertFirst(L, P1);
57
58     P1 = createData("Ali", "01", '1', 3.3);
59 }
```

NA

File Edit View

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

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

Ln 16, Col 25 Spaces: 4 UTF-8 CRLF {} C++ Win32 Prettier


```
Go Run ... < -> Code
main.cpp x
Modul 13 > Unguided > main.cpp > sortList(List &L)
53 P1 = createData("Fahmi", '06', '1', 3.45);
54 insertLast(L, P1);
55
56 P1 = createData("Bobi", "02", '1', 3.71);
57 insertFirst(L, P1);
58
59 P1 = createData("Ali", "01", '1', 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", '1', 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;
```

File Edit View

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

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

Ln 16, Col 25 Spaces: 4 UTF-8 CRLF C++ Win32 Prettier

circularlist.cpp

```
Go Run ... < -> Code
circularlist.cpp x
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(intotype 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     }
}
```

File Edit View

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

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

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF C++ Win32 Prettier

```
Go Run ... < -> Q Code
circularlist.cpp X
Modul 13 > Unguided > circularlist.cpp > ...
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 deleteLast(List &L, address &P) {
71     if (L.first != Nil) {
72         P = L.first;
73         if (P->next == L.first) {
74             L.first = Nil;
75         } else {
76             address Q = L.first;
77             while (Q->next != L.first) {
78                 Q = Q->next;
79             }
80             Q->next = Nil;
81         }
82     }
83 }
84
85 void deleteAfter(List &L, address Prec, address &P) {
86     if (Prec != Nil) {
87         P = Prec->next;
88         if (P == L.first && P->next == L.first) {
89             L.first = Nil;
90         } else {
91             Prec->next = P->next;
92             if (P == L.first) {
93                 L.first = P->next;
94             }
95         }
96         P->next = Nil;
97     }
98 }
99
100 void deleteBefore(List &L, address Prec, address &P) {
101     if (Prec != Nil) {
102         P = Prec;
103         if (P->next == L.first) {
104             L.first = Nil;
105         } else {
106             address Q = L.first;
107             while (Q->next != L.first) {
108                 Q = Q->next;
109             }
110             Q->next = Nil;
111         }
112     }
113 }
114
115 void deleteAll(List &L) {
116     L.first = Nil;
117 }
118
119 void printList(List &L) {
120     if (L.first == Nil) {
121         cout << "List is empty" << endl;
122     } else {
123         address P = L.first;
124         do {
125             cout << P->data << " ";
126             P = P->next;
127         } while (P != L.first);
128         cout << endl;
129     }
130 }
131
132 void main() {
133     List L;
134     L.first = Nil;
135     address P, Q, R;
136     P = new Node(1);
137     Q = new Node(2);
138     R = new Node(3);
139     insertLast(L, P);
140     insertLast(L, Q);
141     insertLast(L, R);
142     printList(L);
143     deleteFirst(L, P);
144     printList(L);
145     deleteLast(L, P);
146     printList(L);
147     deleteAfter(L, P, P);
148     printList(L);
149     deleteBefore(L, P, P);
150     printList(L);
151     deleteAll(L);
152     printList(L);
153 }
```

File Edit View A 100% Wind UTF-8

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

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

```
Go Run ... < -> Q Code
circularlist.cpp X
Modul 13 > Unguided > circularlist.cpp > ...
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 deleteLast(List &L, address &P) {
71     if (L.first != Nil) {
72         P = L.first;
73         if (P->next == L.first) {
74             L.first = Nil;
75         } else {
76             address Q = L.first;
77             while (Q->next != L.first) {
78                 Q = Q->next;
79             }
80             Q->next = Nil;
81         }
82     }
83 }
84
85 void deleteAfter(List &L, address Prec, address &P) {
86     if (Prec != Nil) {
87         P = Prec->next;
88         if (P == L.first && P->next == L.first) {
89             L.first = Nil;
90         } else {
91             Prec->next = P->next;
92             if (P == L.first) {
93                 L.first = P->next;
94             }
95         }
96         P->next = Nil;
97     }
98 }
99
100 void deleteBefore(List &L, address Prec, address &P) {
101     if (Prec != Nil) {
102         P = Prec;
103         if (P->next == L.first) {
104             L.first = Nil;
105         } else {
106             address Q = L.first;
107             while (Q->next != L.first) {
108                 Q = Q->next;
109             }
110             Q->next = Nil;
111         }
112     }
113 }
114
115 void deleteAll(List &L) {
116     L.first = Nil;
117 }
118
119 void printList(List &L) {
120     if (L.first == Nil) {
121         cout << "List is empty" << endl;
122     } else {
123         address P = L.first;
124         do {
125             cout << P->data << " ";
126             P = P->next;
127         } while (P != L.first);
128         cout << endl;
129     }
130 }
131
132 void main() {
133     List L;
134     L.first = Nil;
135     address P, Q, R;
136     P = new Node(1);
137     Q = new Node(2);
138     R = new Node(3);
139     insertLast(L, P);
140     insertLast(L, Q);
141     insertLast(L, R);
142     printList(L);
143     deleteFirst(L, P);
144     printList(L);
145     deleteLast(L, P);
146     printList(L);
147     deleteAfter(L, P, P);
148     printList(L);
149     deleteBefore(L, P, P);
150     printList(L);
151     deleteAll(L);
152     printList(L);
153 }
```

File Edit View A 100% Wind UTF-8

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

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

```
Go Run ... < > Code
circularlist.cpp x
Modul 13 > Unguided > circularlist.cpp > ...
86 void deleteLast(List &L, address &P) {
87     if (L.first != Nil) {
91     } else {
92         address Q = L.first;
93         while (Q->next->next != L.first) {
94             Q = Q->next;
95         }
96         P = Q->next;
97         Q->next = L.first;
98     }
99     P->next = Nil;
100 }
101 }
102
103 address findElm(List L, infotype x) {
104     if (L.first == Nil) {
105         return Nil;
106     }
107
108     address P = L.first;
109     do {
110         if (P->info.nim == x.nim) {
111             return P;
112         }
113         P = P->next;
114     } while (P != L.first);
115
116     return Nil;
117 }
118
119 void printInfo(List L) {
120     if (L.first == Nil) {
121         cout << "List kosong" << endl;
122     } else {
123         address P = L.first;
124         do {
125             cout << "Nama : " << P->info.nama << endl;
126             cout << "NIM : " << P->info.nim << endl;
127             cout << "L/P : " << P->info.jenis_kelamin << endl;
128             cout << "IPK : " << P->info.ipk << endl;
129             cout << endl;
130             P = P->next;
131         } while (P != L.first);
132     }
133 }
```

File Edit View A+ 100% Wind UTF-8

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

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

```
118
119 void printInfo(List L) {
120     if (L.first == Nil) {
121         cout << "List kosong" << endl;
122     } else {
123         address P = L.first;
124         do {
125             cout << "Nama : " << P->info.nama << endl;
126             cout << "NIM : " << P->info.nim << endl;
127             cout << "L/P : " << P->info.jenis_kelamin << endl;
128             cout << "IPK : " << P->info.ipk << endl;
129             cout << endl;
130             P = P->next;
131         } while (P != L.first);
132     }
133 }
```

File Edit View A+ 100% Wind UTF-8

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

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

circularlist.h

The image displays two screenshots of a C++ IDE, likely Visual Studio Code, showing the development of a circular linked list. The top screenshot shows the initial structure of the program, including the typedef for Elmlist and the struct List. The bottom screenshot shows the completion of the List struct and the implementation of various operations like insert, delete, and find.

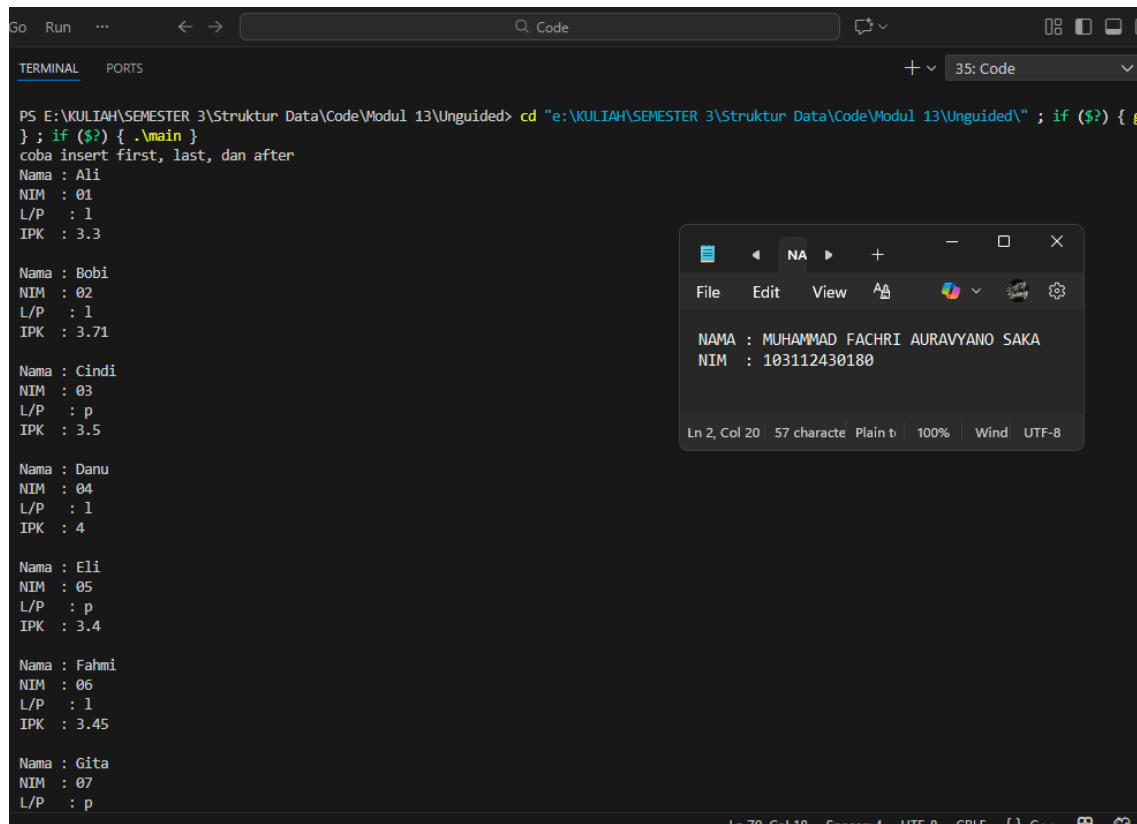
Top Screenshot:

```
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);
```

Bottom Screenshot:

```
1 #ifndef CIRCULARLIST_H_INCLUDED
2
3 struct List {
4     address first;
5 };
6
7 void createList(List &L);
8
9 address alokasi(infotype x);
10 void dealokasi(address &P);
11
12 void insertFirst(List &L, address P);
13 void insertAfter(List &L, address Prec, address P);
14 void insertLast(List &L, address P);
15
16 void deleteFirst(List &L, address &P);
17 void deleteAfter(List &L, address Prec, address &P);
18 void deleteLast(List &L, address &P);
19
20 address findElm(List L, infotype x);
21
22 void printInfo(List L);
23
24 #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 ($?) { g
}; if ($?) { .\main }
coba insert first, last, dan after
Nama : Ali
NIM : 01
L/P : 1
IPK : 3.3

Nama : Bobi
NIM : 02
L/P : 1
IPK : 3.71

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

Nama : Danu
NIM : 04
L/P : 1
IPK : 4

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

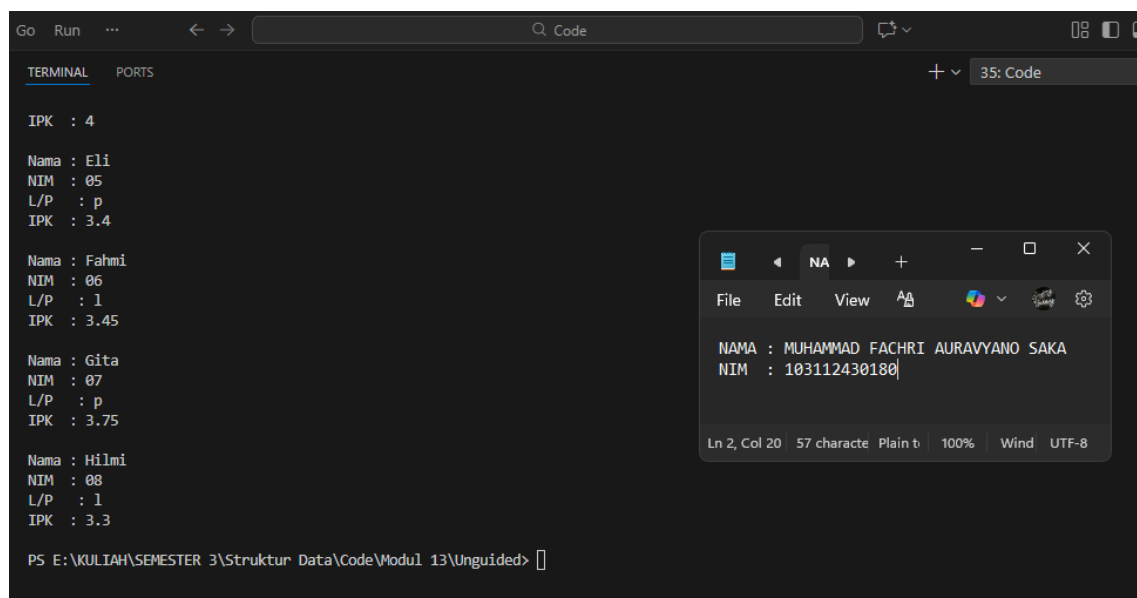
Nama : Fahmi
NIM : 06
L/P : 1
IPK : 3.45

Nama : Gita
NIM : 07
L/P : p
```

File Edit View A 100% Wind UTF-8

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

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



```
IPK : 4

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

Nama : Fahmi
NIM : 06
L/P : 1
IPK : 3.45

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

Nama : Hilmi
NIM : 08
L/P : 1
IPK : 3.3

PS E:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 13\Unguided> []
```

File Edit View A 100% Wind UTF-8

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

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

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](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.