

LAPORAN PRAKTIKUM STRUKTUR DATA

PERTEMUAN 10

MULTI LINKED LIST



Nama :

Reyner Atira Prasetyo (2311104057)

S1SE-07-02

Dosen :

Wahyu Andi Saputra, S.Pd., M.Eng.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TUJUAN

- a. Memahami konsep penggunaan fungsi rekursif.
- b. Mengimplementasikan bentuk-bentuk fungsi rekursif.
- c. Mengaplikasikan struktur data tree dalam sebuah kasus pemrograman.
- d. Mengimplementasikan struktur data tree, khususnya Binary Tree.
- e. Mengimplementasikan struktur data tree, khususnya Binary Tree.

TOOL

1. Visual Studio Code
2. GCC

II. DASAR TEORI

Multi List merupakan sekumpulan list yang berbeda yang memiliki suatu keterhubungan satu sama lain. Tiap elemen dalam multi link list dapat membentuk list sendiri. Biasanya ada yang bersifat sebagai list induk dan list anak.

III. GUIDED

1. Guided1

```

1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6
7  struct Node {
8      int data;
9      Node* next;
10     Node* child;
11
12     Node(int val) : data(val), next(nullptr), child(nullptr) {}
13 };
14
15
16 class MultilinkedList {
17 private:
18     Node* head;
19
20 public:
21     MultilinkedList() : head(nullptr) {}
22
23
24     void addParent(int data) {
25         Node* newNode = new Node(data);
26         newNode->next = head;
27         head = newNode;
28     }
29
30
31     void addChild(int parentData, int childData) {
32         Node* parent = head;
33         while (parent != nullptr && parent->data != parentData) {
34             parent = parent->next;
35         }
36         if (parent != nullptr) {
37             Node* newChild = new Node(childData);
38             newChild->next = parent->child;
39             parent->child = newChild;
40         } else {
41             cout << "Parent not found!" << endl;
42         }
43     }
44
45
46     void display() {
47         Node* current = head;
48         while (current != nullptr) {
49             cout << "Parent: " << current->data << " -> ";
50             Node* child = current->child;
51             while (child != nullptr) {
52                 cout << child->data << " ";
53                 child = child->next;
54             }
55             cout << endl;
56             current = current->next;
57         }
58     }
59
60     ~MultilinkedList() {
61
62         while (head != nullptr) {
63             Node* temp = head;
64             head = head->next;
65
66             while (temp->child != nullptr) {
67                 Node* childTemp = temp->child;
68                 temp->child = temp->child->next;
69                 delete childTemp;
70             }
71             delete temp;
72         }
73     }
74 };
75
76
77 int main() {
78     MultilinkedList mList;
79
80     mList.addParent(1);
81     mList.addParent(2);
82     mList.addParent(3);
83
84     mList.addChild(1, 10);
85     mList.addChild(1, 11);
86     mList.addChild(2, 20);
87     mList.addChild(2, 20);
88     mList.addChild(3, 30);
89     mList.addChild(3, 30);
90     mList.display();
91
92     return 0;
93 }
94

```

Hasil Run :

```
IEngine-Error-xe1m0bxn.fxf' '--pid=Microsof
Parent: 3 -> 30 30
Parent: 2 -> 20 20
Parent: 1 -> 11 10
PS D:\PRAKTIKUM\Struktur Data\pertemuan10>
```

2. Guided2

```

1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6
7  struct EmployeeNode {
8      string name;
9      EmployeeNode* next;
10     EmployeeNode* subordinate;
11
12     EmployeeNode(string empName) : name(empName), next(nullptr), subordinate(nullptr) {}
13 };
14
15
16 class EmployeeList {
17 private:
18     EmployeeNode* head;
19
20 public:
21     EmployeeList() : head(nullptr) {}
22
23
24     void addEmployee(string name) {
25         EmployeeNode* newEmployee = new EmployeeNode(name);
26         newEmployee->next = head;
27         head = newEmployee;
28     }
29
30
31     void addSubordinate(string managerName, string subordinateName) {
32         EmployeeNode* manager = head;
33         while (manager != nullptr && manager->name != managerName) {
34             manager = manager->next;
35         }
36         if (manager != nullptr) {
37             EmployeeNode* newSubordinate = new EmployeeNode(subordinateName);
38             newSubordinate->next = manager->subordinate;
39             manager->subordinate = newSubordinate;
40         } else {
41             cout << "Manager not found!" << endl;
42         }
43     }
44
45
46     void display() {
47         EmployeeNode* current = head;
48         while (current != nullptr) {
49             cout << "Manager: " << current->name << " -> ";
50             EmployeeNode* sub = current->subordinate;
51             while (sub != nullptr) {
52                 cout << sub->name << " ";
53                 sub = sub->next;
54             }
55             cout << endl;
56             current = current->next;
57         }
58     }
59
60     ~EmployeeList() {
61
62         while (head != nullptr) {
63             EmployeeNode* temp = head;
64             head = head->next;
65
66
67             while (temp->subordinate != nullptr) {
68                 EmployeeNode* subTemp = temp->subordinate;
69                 temp->subordinate = temp->subordinate->next;
70                 delete subTemp;
71             }
72             delete temp;
73         }
74     }
75 };
76
77 int main() {
78     EmployeeList emplList;
79
80     emplList.addEmployee("Alice");
81     emplList.addEmployee("Bob");
82     emplList.addEmployee("Charlie");
83
84     emplList.addSubordinate("Alice", "David");
85     emplList.addSubordinate("Alice", "Eve");
86     emplList.addSubordinate("Bob", "Frank");
87
88     emplList.addSubordinate("Charlie", "Frans");
89     emplList.addSubordinate("Charlie", "Brian");
90
91     emplList.display();
92
93     return 0;
94 }
95

```

Hasil Run :

```
Engine-Error-hp4yjp0g.nj3' '--pid=Microsof  
Manager: Charlie -> Brian Frans  
Manager: Bob -> Frank  
Manager: Alice -> Eve David  
PS D:\PRAKTIKUM\Struktur Data\pertemuan10>
```

3. Guided3

```

1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 // Struktur untuk node karyawan
7 struct EmployeeNode {
8     string name; // Nama karyawan
9     EmployeeNode* next; // Pointer ke karyawan berikutnya
10    EmployeeNode* subordinate; // Pointer ke subordinate pertama
11
12    EmployeeNode(string name) : name(name), next(nullptr), subordinate(nullptr) {}
13 };
14
15 // Kelas untuk Multi-Linked List Karyawan
16 class EmployeeList {
17 private:
18     EmployeeNode* head; // Pointer ke kepala list
19
20 public:
21     EmployeeList() : head(nullptr) {}
22
23     // Menambahkan karyawan (induk)
24     void addEmployee(string name) {
25         EmployeeNode* newEmployee = new EmployeeNode(name);
26         newEmployee->next = head; // Menyambungkan ke karyawan sebelumnya
27         head = newEmployee; // Memperbarui head
28     }
29
30     // Menambahkan subordinate ke karyawan tertentu
31     void addSubordinate(string managerName, string subordinateName) {
32         EmployeeNode* manager = head;
33         while (manager != nullptr && manager->name != managerName) {
34             manager = manager->next;
35         }
36         if (manager != nullptr) { // Jika manager ditemukan
37             EmployeeNode* newSubordinate = new EmployeeNode(subordinateName);
38             newSubordinate->next = manager->subordinate; // Menyambungkan ke subordinate sebelumnya
39             manager->subordinate = newSubordinate; // Memperbarui subordinate
40         } else {
41             cout << "Manager not found!" << endl;
42         }
43     }
44
45     // Menghapus karyawan (induk)
46     void deleteEmployee(string name) {
47         EmployeeNode* current = head;
48         while (current != nullptr && (current->name != name) {
49             current = (current->next);
50         }
51
52         if (current != nullptr) { // Jika karyawan ditemukan
53             EmployeeNode* toDelete = current;
54             current = (current->next);
55
56             // Hapus semua subordinate dari node ini
57             while (toDelete->subordinate != nullptr) {
58                 EmployeeNode* subTemp = toDelete->subordinate;
59                 toDelete->subordinate = toDelete->subordinate->next;
60                 delete subTemp;
61             }
62             delete toDelete;
63             cout << "Employee " << name << " deleted." << endl;
64         } else {
65             cout << "Employee not found!" << endl;
66         }
67     }
68
69     // Menghapus subordinate dari karyawan tertentu
70     void deleteSubordinate(string managerName, string subordinateName) {
71         EmployeeNode* manager = head;
72         while (manager != nullptr && manager->name != managerName) {
73             manager = manager->next;
74         }
75
76         if (manager != nullptr) { // Jika manager ditemukan
77             EmployeeNode* currentSub = (manager->subordinate);
78             while (currentSub != nullptr && (currentSub->name != subordinateName) {
79                 currentSub = (currentSub->next);
80             }
81
82             if (currentSub != nullptr) { // Jika subordinate ditemukan
83                 EmployeeNode* toDelete = currentSub;
84                 currentSub = (currentSub->next); // Hapus dari list
85                 delete toDelete; // Hapus node subordinate
86             } else {
87                 cout << "Subordinate " << subordinateName << " deleted from " << managerName << "." << endl;
88             }
89         } else {
90             cout << "Manager not found!" << endl;
91         }
92     }
93
94     // Menampilkan daftar karyawan dan subordinate mereka
95     void display() {
96         EmployeeNode* current = head;
97         while (current != nullptr) {
98             cout << "Manager: " << current->name << " -> ";
99             EmployeeNode* sub = current->subordinate;
100             while (sub != nullptr) {
101                 cout << sub->name << " ";
102                 sub = sub->next;
103             }
104             cout << endl;
105             current = current->next;
106         }
107     }
108
109     ~EmployeeList() {
110         // Destructor untuk membersihkan memori
111         while (head != nullptr) {
112             EmployeeNode* temp = head;
113             head = head->next;
114
115             // Hapus semua subordinate dari node ini
116             while (temp->subordinate != nullptr) {
117                 EmployeeNode* subTemp = temp->subordinate;
118                 temp->subordinate = temp->subordinate->next;
119                 delete subTemp;
120             }
121             delete temp;
122         }
123     }
124 };
125
126 int main() {
127     EmployeeList empList;
128
129     empList.addEmployee("Alice");
130     empList.addEmployee("Bob");
131     empList.addEmployee("Charlie");
132
133     empList.addSubordinate("Alice", "David");
134     empList.addSubordinate("Alice", "Eve");
135     empList.addSubordinate("Bob", "Frank");
136
137     cout << "Initial employee list:" << endl;
138     empList.display(); // Menampilkan daftar karyawan
139
140     empList.deleteSubordinate("Alice", "David"); // Menghapus David dari Alice
141     empList.deleteEmployee("Charlie"); // Menghapus Charlie
142
143     cout << "Updated employee list:" << endl;
144     empList.display(); // Menampilkan daftar setelah penghapusan
145
146     return 0;
147 }

```

Hasil Run :

```
IEEngine-Error-5peacqoh.aeb' '--pid=Microsof
Initial employee list:
Manager: Charlie ->
Manager: Bob -> Frank
Manager: Alice -> Eve David
Subordinate David deleted from Alice.
Employee Charlie deleted.

Updated employee list:
Manager: Bob -> Frank
Manager: Alice -> Eve
PS D:\PRAKTIKUM\Struktur Data\pertemuan10>
```

IV. UNGUIDED

1. Unguided1.cpp


```

1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 struct Project {
6     string projectName;
7     int duration;
8     Project() {}
9 };
10
11 struct Employee {
12     string employeeName;
13     string employeeID;
14     Project* projectHead;
15     Employee* nextEmployee;
16 };
17
18 // Head of the Employee list
19 Employee* employeeHead = nullptr;
20
21 // Function to create a new employee node
22 Employee* createEmployee(string name, string id) {
23     Employee* newEmployee = new Employee;
24     newEmployee->employeeName = name;
25     newEmployee->employeeID = id;
26     newEmployee->projectHead = nullptr;
27     newEmployee->nextEmployee = nullptr;
28     return newEmployee;
29 }
30
31 // Function to create a new project node
32 Project* createProject(string name, int duration) {
33     Project* newProject = new Project;
34     newProject->projectName = name;
35     newProject->duration = duration;
36     newProject->nextProject = nullptr;
37     return newProject;
38 }
39
40 // Function to add an employee to the list
41 void addEmployee(string name, string id) {
42     Employee* newEmployee = createEmployee(name, id);
43     if (employeeHead == nullptr) {
44         employeeHead = newEmployee;
45     } else {
46         Employee* temp = employeeHead;
47         while (temp->nextEmployee != nullptr) {
48             temp = temp->nextEmployee;
49         }
50         temp->nextEmployee = newEmployee;
51     }
52     cout << "Employee : " << name << " with ID : " << id << " added.\n";
53 }
54
55 // Function to add a project to an employee
56 void addProjectToEmployee(string employeeID, string projectName, int duration) {
57     Employee* temp = employeeHead;
58     while (temp != nullptr && temp->employeeID != employeeID) {
59         temp = temp->nextEmployee;
60     }
61
62     if (temp == nullptr) {
63         cout << "Employee with ID : " << employeeID << " not found.\n";
64         return;
65     }
66
67     Project* newProject = createProject(projectName, duration);
68     if (temp->projectHead == nullptr) {
69         temp->projectHead = newProject;
70     } else {
71         Project* pTemp = temp->projectHead;
72         while (pTemp->nextProject != nullptr) {
73             pTemp = pTemp->nextProject;
74         }
75         pTemp->nextProject = newProject;
76     }
77     cout << "Project : " << projectName << " with duration : " << duration << " months added to employee : " << temp->employeeName << ".\n";
78 }
79
80 // Function to delete a project from an employee
81 void deleteProjectFromEmployee(string employeeID, string projectName) {
82     Employee* temp = employeeHead;
83     while (temp != nullptr && temp->employeeID != employeeID) {
84         temp = temp->nextEmployee;
85     }
86
87     if (temp == nullptr) {
88         cout << "Employee with ID : " << employeeID << " not found.\n";
89         return;
90     }
91
92     Project* pTemp = temp->projectHead;
93     Project* prev = nullptr;
94
95     while (pTemp != nullptr && pTemp->projectName != projectName) {
96         prev = pTemp;
97         pTemp = pTemp->nextProject;
98     }
99
100     if (pTemp == nullptr) {
101         cout << "Project : " << projectName << " not found for employee : " << temp->employeeName << ".\n";
102         return;
103     }
104
105     if (prev == nullptr) {
106         temp->projectHead = pTemp->nextProject;
107     } else {
108         prev->nextProject = pTemp->nextProject;
109     }
110
111     delete pTemp;
112     cout << "Project : " << projectName << " removed from employee : " << temp->employeeName << ".\n";
113 }
114
115 // Function to display all employees and their projects
116 void displayEmployees() {
117     Employee* temp = employeeHead;
118     while (temp != nullptr) {
119         cout << "Employee Name : " << temp->employeeName << ", ID : " << temp->employeeID << ".\n";
120         Project* pTemp = temp->projectHead;
121         while (pTemp != nullptr) {
122             cout << "Project : " << pTemp->projectName << ", duration : " << pTemp->duration << " months.\n";
123             pTemp = pTemp->nextProject;
124         }
125         temp = temp->nextEmployee;
126         cout << "\n";
127     }
128 }
129
130 int main() {
131     // Adding employees
132     addEmployee("Andi", "0001");
133     addEmployee("Budi", "0002");
134     addEmployee("Citra", "0003");
135     cout << "\n";
136     displayEmployees();
137     cout << "-----\n";
138
139     // Adding projects to employees
140     addProjectToEmployee("0001", "Aplikasi Pabrik", 12);
141     addProjectToEmployee("0002", "Sistem Monitoring", 8);
142     addProjectToEmployee("0003", "E-commerce", 10);
143     cout << "\n";
144     displayEmployees();
145     cout << "-----\n";
146
147     // Adding a new project to Andi
148     addProjectToEmployee("0001", "Analisis Data", 6);
149     cout << "\n";
150     displayEmployees();
151     cout << "-----\n";
152
153     // Deleting a project from Budi
154     deleteProjectFromEmployee("0002", "Aplikasi Pabrik");
155     cout << "\n";
156     displayEmployees();
157
158     return 0;
159 }

```

Penjelasan :

Struktur Data

- Project: Representasi proyek dengan atribut:
 - projectName: Nama proyek.
 - duration: Durasi proyek (dalam bulan).
 - nextProject: Pointer ke proyek berikutnya.
- Employee: Representasi karyawan dengan atribut:
 - employeeName: Nama karyawan.
 - employeeID: ID karyawan.
 - projectHead: Pointer ke daftar proyek milik karyawan.
 - nextEmployee: Pointer ke karyawan berikutnya.

Fungsi Utama:

- createEmployee(): Membuat node karyawan baru.
- createProject(): Membuat node proyek baru.
- addEmployee(): Menambahkan karyawan baru ke daftar.
- addProjectToEmployee(): Menambahkan proyek baru ke karyawan tertentu berdasarkan ID.
- deleteProjectFromEmployee(): Menghapus proyek tertentu dari karyawan berdasarkan ID.
- displayEmployees(): Menampilkan daftar karyawan beserta semua proyek mereka.

Fungsi main():

- Menambahkan karyawan: Andi, Budi, Citra.
- Menambahkan proyek ke masing-masing karyawan.
- Menampilkan daftar karyawan dan proyek.
- Menambahkan proyek tambahan untuk Andi dan menghapus proyek

"Aplikasi Mobile" dari Andi.

Hasil Run :

```
Engine-Error=0c41kde.sur --pid=Microsoft=Engine-Pid=00221a5.n3a --dbgexe=C:\msys64\ucrt64\bin
Employee : Andi with ID : P001 added.
Employee : Budi with ID : P002 added.
Employee : Citra with ID : P003 added.

Employee Name : Andi, ID : P001

Employee Name : Budi, ID : P002

Employee Name : Citra, ID : P003

-----
Project : Aplikasi Mobile with duration : 12 months added to employee : Andi.
Project : Sistem Akuntansi with duration : 8 months added to employee : Budi.
Project : E-commerce with duration : 10 months added to employee : Citra.

Employee Name : Andi, ID : P001
Project : Aplikasi Mobile, Duration : 12 months

Employee Name : Budi, ID : P002
Project : Sistem Akuntansi, Duration : 8 months

Employee Name : Citra, ID : P003
Project : E-commerce, Duration : 10 months

-----
Project : Analisis Data with duration : 6 months added to employee : Andi.

Employee Name : Andi, ID : P001
Project : Aplikasi Mobile, Duration : 12 months
Project : Analisis Data, Duration : 6 months

Employee Name : Budi, ID : P002
Project : Sistem Akuntansi, Duration : 8 months

Employee Name : Citra, ID : P003
Project : E-commerce, Duration : 10 months

-----
```

```
-----
Project : Aplikasi Mobile removed from employee : Andi.

Employee Name : Andi, ID : P001
Project : Analisis Data, Duration : 6 months

Employee Name : Budi, ID : P002
Project : Sistem Akuntansi, Duration : 8 months

Employee Name : Citra, ID : P003
Project : E-commerce, Duration : 10 months
```

2. Unguided2.cpp

```

1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 struct Book {
6     string title;
7     string returnDate;
8     Book* nextBook;
9 };
10
11 struct Member {
12     string name;
13     string id;
14     Book* bookHead;
15     Member* nextMember;
16 };
17
18 // Head of the Member list
19 Member* memberHead = nullptr;
20
21 // Function to create a new member node
22 Member* createMember(string name, string id) {
23     Member* newMember = new Member;
24     newMember->name = name;
25     newMember->id = id;
26     newMember->bookHead = nullptr;
27     newMember->nextMember = nullptr;
28     return newMember;
29 }
30
31 // Function to create a new book node
32 Book* createBook(string title, string returnDate) {
33     Book* newBook = new Book;
34     newBook->title = title;
35     newBook->returnDate = returnDate;
36     newBook->nextBook = nullptr;
37     return newBook;
38 }
39
40 // Function to add a member to the list
41 void addMember(string name, string id) {
42     Member* newMember = createMember(name, id);
43     if (memberHead == nullptr) {
44         memberHead = newMember;
45     } else {
46         Member* temp = memberHead;
47         while (temp->nextMember != nullptr) {
48             temp = temp->nextMember;
49         }
50         temp->nextMember = newMember;
51     }
52     cout << "Member : " << name << " with ID : " << id << " added.\n";
53 }
54
55 // Function to add a book to a member
56 void addBookToMember(string memberId, string title, string returnDate) {
57     Member* temp = memberHead;
58     while (temp != nullptr && temp->id != memberId) {
59         temp = temp->nextMember;
60     }
61     if (temp == nullptr) {
62         cout << "Member with ID " << memberId << " not found.\n";
63         return;
64     }
65     Book* newBook = createBook(title, returnDate);
66     if (temp->bookHead == nullptr) {
67         temp->bookHead = newBook;
68     } else {
69         Book* bTemp = temp->bookHead;
70         while (bTemp->nextBook != nullptr) {
71             bTemp = bTemp->nextBook;
72         }
73         bTemp->nextBook = newBook;
74     }
75     temp->nextBook = newBook;
76     cout << "Book : " << title << " with return date : " << returnDate << " added to member : " << temp->name << ".\n";
77 }
78
79 // Function to delete a member and all their books
80 void deleteMember(string memberId) {
81     Member* temp = memberHead;
82     Member* prev = nullptr;
83
84     while (temp != nullptr && temp->id != memberId) {
85         prev = temp;
86         temp = temp->nextMember;
87     }
88     if (temp == nullptr) {
89         cout << "Member with ID " << memberId << " not found.\n";
90         return;
91     }
92     if (prev == nullptr) {
93         memberHead = temp->nextMember;
94     } else {
95         prev->nextMember = temp->nextMember;
96     }
97
98     // Delete all books of the member
99     Book* bTemp = temp->bookHead;
100     while (bTemp != nullptr) {
101         Book* toDelete = bTemp;
102         bTemp = bTemp->nextBook;
103         delete toDelete;
104     }
105     delete temp;
106     cout << "Member with ID : " << memberId << " deleted.\n";
107 }
108
109 // Function to display all members and their books
110 void displayMembers() {
111     Member* temp = memberHead;
112     while (temp != nullptr) {
113         cout << "Member Name : " << temp->name << ", ID : " << temp->id << ".\n";
114         Book* bTemp = temp->bookHead;
115         while (bTemp != nullptr) {
116             cout << "Book Title : " << bTemp->title << ", Return Date : " << bTemp->returnDate << ".\n";
117             bTemp = bTemp->nextBook;
118         }
119         temp = temp->nextMember;
120     }
121     cout << "\n";
122 }
123
124 int main() {
125     // Adding members
126     addMember("Ravi", "A001");
127     addMember("Dilip", "A002");
128     addMember("Vijay", "A003");
129     cout << "\n";
130     displayMembers();
131     cout << "-----\n";
132
133     // Adding books to members
134     addBookToMember("A001", "Panchangama C++", "15/12/2024");
135     addBookToMember("A003", "Algorithm Programming", "15/12/2024");
136     cout << "\n";
137     displayMembers();
138     cout << "-----\n";
139
140     // Adding a new book to Ravi
141     addBookToMember("A001", "Structure Data", "18/12/2024");
142     cout << "\n";
143     displayMembers();
144     cout << "-----\n";
145
146     // Deleting member Dijo
147     deleteMember("A002");
148     cout << "\n";
149     displayMembers();
150     return 0;
151 }

```

Penjelasan :

Struktur Data:

- Book:
 - title: Nama buku.
 - returnDate: Tanggal pengembalian.
 - nextBook: Pointer ke buku berikutnya dalam daftar buku anggota.
- Member:
 - name: Nama anggota.
 - id: ID anggota.
 - bookHead: Pointer ke daftar buku yang dipinjam oleh anggota.
 - nextMember: Pointer ke anggota berikutnya dalam daftar anggota.

Fungsi Utama:

- createMember(): Membuat node anggota baru.
- createBook(): Membuat node buku baru.
- addMember(): Menambahkan anggota ke daftar.
- addBookToMember(): Menambahkan buku ke anggota tertentu berdasarkan ID.
- deleteMember(): Menghapus anggota beserta semua buku yang dipinjamnya.
- displayMembers(): Menampilkan daftar anggota dan buku yang dipinjam masing-masing anggota.

Fungsi main():

- Menambahkan anggota: Rani, Dito, dan Vina.
- Menambahkan buku ke anggota:
- Rani meminjam "Pemrograman C++".
- Dito meminjam "Algoritma Pemrograman".

- Menampilkan daftar anggota dan buku mereka.
- Menambahkan buku tambahan untuk Rani ("Struktur Data").
- Menghapus anggota Dito beserta semua buku yang dipinjamnya.

Hasil Run :

```
Member : Rani with ID : A001 added.
Member : Dito with ID : A002 added.
Member : Vina with ID : A003 added.

Member Name: Rani, ID: A001

Member Name: Dito, ID: A002

Member Name: Vina, ID: A003

-----
Book : Pemrograman C++ with return date : 01/12/2024 added to member : Rani.
Book : Algoritma Pemrograman with return date : 15/12/2024 added to member : Dito.

Member Name: Rani, ID: A001
  Book Title: Pemrograman C++, Return Date: 01/12/2024

Member Name: Dito, ID: A002
  Book Title: Algoritma Pemrograman, Return Date: 15/12/2024

Member Name: Vina, ID: A003

-----
Book : Struktur Data with return date : 10/12/2024 added to member : Rani.

Member Name: Rani, ID: A001
  Book Title: Pemrograman C++, Return Date: 01/12/2024
  Book Title: Struktur Data, Return Date: 10/12/2024

Member Name: Dito, ID: A002
  Book Title: Algoritma Pemrograman, Return Date: 15/12/2024

Member Name: Vina, ID: A003

-----
Member with ID : A002 deleted.

Member Name: Rani, ID: A001
  Book Title: Pemrograman C++, Return Date: 01/12/2024
  Book Title: Struktur Data, Return Date: 10/12/2024

Member Name: Vina, ID: A003
```

V. KESIMPULAN

Pada praktikum ini, telah berhasil dibuat dan diimplementasikan struktur data Multi Linked List. Multi Linked List merupakan salah satu variasi dari linked list yang mendukung koneksi antara elemen dalam lebih dari satu dimensi, seperti menghubungkan elemen berdasarkan kategori utama dan sub-kategori. Struktur ini sering digunakan untuk merepresentasikan data hierarkis atau relasional.

Dari hasil praktikum, dapat disimpulkan beberapa hal:

Fleksibilitas Struktur Data: Multi Linked List memungkinkan pengelolaan data yang lebih kompleks dibandingkan single atau double linked list, terutama dalam kasus di mana elemen data memiliki banyak hubungan atau relasi.

Operasi Dasar: Operasi seperti penambahan, penghapusan, dan pencarian elemen memerlukan pemahaman yang mendalam tentang struktur dan pointer, karena elemen dapat memiliki banyak hubungan antar node.

Efisiensi: Meski memberikan fleksibilitas, Multi Linked List cenderung lebih rumit untuk diimplementasikan dan memerlukan lebih banyak memori dibandingkan linked list sederhana, karena setiap node harus menyimpan lebih dari satu pointer.

Aplikasi Praktis: Struktur ini dapat digunakan dalam berbagai aplikasi nyata, seperti sistem manajemen basis data, representasi graf, atau penyimpanan data relasional dalam hierarki.

Kesulitan utama dalam implementasi Multi Linked List adalah menjaga konsistensi relasi antar node, terutama saat terjadi operasi modifikasi pada list. Oleh karena itu, penting untuk merancang algoritma yang efisien dan memastikan setiap perubahan diperiksa secara menyeluruh.

Secara keseluruhan, praktikum ini memberikan pemahaman yang baik mengenai konsep dasar dan implementasi Multi Linked List, serta pentingnya manajemen memori dan pointer dalam struktur data kompleks.