

**LAPORAN PRAKTIKUM  
PERTEMUAN 13  
MULTI LINKED LIST**



**Nama :**

**Haza Zaidan Zidna Fann**

**(2311104056)**

**Dosen :**

**Wahyu Andi Saputra**

**PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK  
FAKULTAS INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO**

**2024**

**I. TUJUAN**

Memahami penggunaan *Multi Linked list*.

Mengimplementasikan *Multi Linked list* dalam beberapa studi kasus.

**II. LANDASAN TEORI**

Multi-linked list adalah struktur data yang menghubungkan elemen-elemen dalam bentuk daftar, di mana setiap elemen dapat memiliki lebih dari satu pointer yang mengarah ke elemen lainnya. Setiap node dalam multi-linked list dapat menyimpan referensi ke beberapa node, memungkinkan hubungan yang lebih kompleks antara data. Struktur ini berguna untuk mengorganisir data dengan hubungan yang lebih dinamis dan fleksibel dibandingkan dengan daftar biasa.

### **III. GUIDED**

```

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
67             while (temp->child != nullptr) {
68                 Node* childTemp = temp->child;
69                 temp->child = temp->child->next;
70                 delete childTemp;
71             }
72             delete temp;
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
95

```

```
Parent: 3 -> 30 30
Parent: 2 -> 20 20
Parent: 1 -> 11 10
```

### **Struktur Data Node:**

Menyimpan data induk/anak.

Pointer next menghubungkan elemen dalam daftar induk.

Pointer child menghubungkan elemen anak dari setiap induk.

### **Kelas MultiLinkedList:**

**addParent:** Menambahkan elemen induk di awal daftar.

**addChild:** Menambahkan elemen anak ke induk yang sudah ada.

**display:** Menampilkan semua induk beserta daftar anaknya.

**Destruktor:** Membersihkan semua memori yang dialokasikan.

### **Main Function:**

Menambahkan tiga induk (1, 2, 3).

Menambahkan beberapa anak ke setiap induk.

Menampilkan hasil berupa daftar induk dan anak.

```

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 empList;
79
80     empList.addEmployee("Alice");
81     empList.addEmployee("Bob");
82     empList.addEmployee("Charlie");
83
84     empList.addSubordinate("Alice", "David");
85     empList.addSubordinate("Alice", "Eve");
86     empList.addSubordinate("Bob", "Frank");
87
88     empList.addSubordinate("Charlie", "Frans");
89     empList.addSubordinate("Charlie", "Brian");
90
91     empList.display();
92
93     return 0;
94 }
95

```

```
Manager: Charlie -> Brian Frans  
Manager: Bob -> Frank  
Manager: Alice -> Eve David
```

### **Struktur Data EmployeeNode:**

Menyimpan nama pegawai (name).

Pointer next menghubungkan pegawai dalam daftar utama.

Pointer subordinate menghubungkan daftar bawahan dari masing-masing pegawai.

### **Kelas EmployeeList:**

**addEmployee:** Menambahkan pegawai baru di awal daftar.

**addSubordinate:** Menambahkan bawahan untuk pegawai tertentu (induk).

**display:** Menampilkan semua pegawai beserta daftar bawahan mereka.

**Destruktor:** Menghapus semua pegawai dan bawahan untuk membebaskan memori.

### **Fungsi Utama (main):**

Menambahkan tiga pegawai utama: *Alice, Bob, Charlie*.

Menambahkan beberapa bawahan untuk masing-masing pegawai.

Menampilkan daftar pegawai beserta bawahan mereka.

```

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 empName) : name(empName), 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     // Menambatkan 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; // Menghapus dari list
85
86                 delete toDelete; // Menghapus node subordinate
87                 cout << "Subordinate " << subordinateName << " deleted from " << managerName << "." << endl;
88             } else {
89                 cout << "Subordinate not found!" << endl;
90             }
91         } else {
92             cout << "Manager not found!" << endl;
93         }
94     }
95
96     // Menampilkan daftar karyawan dan subordinate mereka
97     void display() {
98         EmployeeNode* current = head;
99         while (current != nullptr) {
100             cout << "Manager: " << current->name << " -> ";
101             EmployeeNode* sub = current->subordinate;
102             while (sub != nullptr) {
103                 cout << sub->name << " -> ";
104                 sub = sub->next;
105             }
106             cout << endl;
107             current = current->next;
108         }
109     }
110
111     ~EmployeeList() {
112         // Destructor untuk membersihkan memori
113         while (head != nullptr) {
114             EmployeeNode* temp = head;
115             head = head->next;
116
117             // Hapus semua subordinate dari node ini
118             while (temp->subordinate != nullptr) {
119                 EmployeeNode* subTemp = temp->subordinate;
120                 temp->subordinate = temp->subordinate->next;
121                 delete subTemp;
122             }
123             delete temp;
124         }
125     }
126 };
127
128 int main() {
129     EmployeeList emplist;
130
131     emplist.addEmployee("Alice");
132     emplist.addEmployee("Bob");
133     emplist.addEmployee("Charlie");
134
135     emplist.addSubordinate("Alice", "David");
136     emplist.addSubordinate("Alice", "Ive");
137     emplist.addSubordinate("Bob", "Frank");
138
139     cout << "Initial employee list:" << endl;
140     emplist.display(); // Menampilkan isi daftar karyawan
141
142     emplist.deleteSubordinate("Alice", "David"); // Menghapus David dari Alice
143     emplist.deleteEmployee("Charlie"); // Menghapus Charlie
144
145     cout << "Updated employee list:" << endl;
146     emplist.display(); // Menampilkan isi daftar setelah penghapusan
147
148     return 0;
149 }
150

```

```
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
```

**EmployeeNode:**

Menyimpan nama karyawan (name), pointer ke karyawan berikutnya (next), dan pointer ke daftar bawahan (subordinate).

**Fitur Utama:****addEmployee:**

Menambahkan karyawan baru sebagai induk di awal daftar.

**addSubordinate:**

Menambahkan bawahan ke karyawan tertentu (induk).

**deleteEmployee:**

Menghapus karyawan dari daftar, beserta seluruh bawahannya.

**deleteSubordinate:**

Menghapus bawahan dari karyawan tertentu tanpa menghapus karyawan itu sendiri.

**display:**

Menampilkan semua karyawan dan daftar bawahannya.

**Destruktor:**

Menghapus seluruh karyawan dan bawahan untuk membebaskan memori.

**Main Function:**

Menambahkan tiga karyawan: *Alice*, *Bob*, dan *Charlie*.

Menambahkan bawahan ke masing-masing karyawan.

Menampilkan daftar awal.

Menghapus bawahan *David* dari *Alice* dan menghapus karyawan *Charlie*.

Menampilkan daftar karyawan yang diperbarui.

#### IV. UNGUIDED



```

1 #include <iostream>
2 #include <conio.h>
3 #include <string>
4 #include <string.h>
5 #include <stdlib.h>
6
7 // Struktur untuk proyek
8 struct Project {
9     string projectName;
10    int duration;
11    Project* nextProject;
12 };
13
14 // Struktur untuk pegawai
15 struct Employee {
16     string employeeName;
17     string employeeID;
18     Project* projectHead;
19     Employee* nextEmployee;
20 };
21
22 // Fungsi untuk membuat pegawai baru
23 Employee* createEmployee(string name, string id) {
24     Employee* newEmployee = new Employee;
25     newEmployee->employeeName = name;
26     newEmployee->employeeID = id;
27     newEmployee->projectHead = nullptr;
28     newEmployee->nextEmployee = nullptr;
29     return newEmployee;
30 }
31
32 // Fungsi untuk membuat proyek baru
33 Project* createProject(string name, int duration) {
34     Project* newProject = new Project;
35     newProject->projectName = name;
36     newProject->duration = duration;
37     newProject->nextProject = nullptr;
38     return newProject;
39 }
40
41 // Menambahkan proyek ke pegawai
42 void addProject(Employee* employee, string projectName, int duration) {
43     Project* newProject = createProject(projectName, duration);
44     if (employee->projectHead == nullptr) {
45         employee->projectHead = newProject;
46     } else {
47         Project* temp = employee->projectHead;
48         while (temp->nextProject != nullptr) {
49             temp = temp->nextProject;
50         }
51         temp->nextProject = newProject;
52     }
53 }
54
55 // Menghapus proyek berdasarkan nama
56 void removeProject(Employee* employee, string projectName) {
57     Project* temp = employee->projectHead;
58     Project* prev = nullptr;
59     while (temp != nullptr && temp->projectName != projectName) {
60         prev = temp;
61         temp = temp->nextProject;
62     }
63     if (temp == nullptr) {
64         cout << "Proyek tidak ditemukan.\n";
65         return;
66     }
67     if (prev == nullptr) {
68         employee->projectHead = temp->nextProject;
69     } else {
70         prev->nextProject = temp->nextProject;
71     }
72     delete temp;
73     cout << "Proyek " << projectName << " berhasil dihapus.\n";
74 }
75
76 // Menampilkan data pegawai dan proyek mereka dalam bentuk tabel
77 void displayEmployeesAndProjects() {
78     cout << left << setw(15) << "Nama Pegawai" << setw(10) << "ID" << "Proyek dan Durasi\n";
79     cout << "-----\n";
80     Employee* tempEmployee = head;
81     while (tempEmployee != nullptr) {
82         cout << left << setw(15) << tempEmployee->employeeName << setw(10) << tempEmployee->employeeID;
83         if (tempEmployee->projectHead != nullptr) {
84             Project* tempProject = tempEmployee->projectHead;
85             if (tempProject != nullptr) {
86                 cout << " " << tempProject->projectName << " (" << tempProject->duration << " bulan)\n";
87                 tempProject = tempProject->nextProject;
88             } else {
89                 cout << " " << tempProject->projectName << " (" << tempProject->duration << " bulan)\n";
90                 tempProject = tempProject->nextProject;
91             }
92         }
93         tempEmployee = tempEmployee->nextEmployee;
94     }
95     cout << endl;
96 }
97
98 // Fungsi untuk menambahkan pegawai ke daftar
99 void addEmployee(Employee* head, string name, string id) {
100    if (head == nullptr) {
101        head = createEmployee(name, id);
102    } else {
103        Employee* temp = head;
104        while (temp->nextEmployee != nullptr) {
105            temp = temp->nextEmployee;
106        }
107        temp->nextEmployee = createEmployee(name, id);
108    }
109 }
110
111 int main() {
112     Employee* head = nullptr;
113
114     int choice;
115     string name, id, projectName;
116     int duration;
117
118     do {
119         cout << "Sistem Manajemen Data Proyek dan Pegawai\n";
120         cout << "1. Tampilkan Pegawai\n";
121         cout << "2. Tampilkan Proyek ke Pegawai\n";
122         cout << "3. Tampilkan Proyek Baru\n";
123         cout << "4. Tampilkan Pegawai\n";
124         cout << "5. Tampilkan Data Pegawai\n";
125         cout << "6. Keluar\n";
126         cout << "Pilih opsi: ";
127         cin >> choice;
128
129         switch (choice) {
130             case 1:
131                 int numEmployees;
132                 cout << "Masukkan jumlah pegawai yang ingin ditampilkan: ";
133                 cin >> numEmployees;
134                 for (int i = 0; i < numEmployees; i++) {
135                     cout << "Masukkan Nama Pegawai: ";
136                     cin.ignore();
137                     getline(cin, name);
138                     cout << "Masukkan ID Pegawai: ";
139                     cin >> id;
140                     addEmployee(head, name, id);
141                 }
142                 break;
143             case 2:
144                 cout << "Masukkan ID Pegawai: ";
145                 cin >> id;
146                 cout << "Masukkan Nama Proyek: ";
147                 cin.ignore();
148                 getline(cin, projectName);
149                 cout << "Masukkan Durasi Proyek (bulan): ";
150                 cin >> duration;
151                 {
152                     Employee* temp = head;
153                     while (temp != nullptr && temp->employeeID != id) {
154                         temp = temp->nextEmployee;
155                     }
156                     if (temp != nullptr) {
157                         addProject(temp, projectName, duration);
158                     } else {
159                         cout << "Pegawai dengan ID " << id << " tidak ditemukan.\n";
160                     }
161                 }
162                 break;
163             case 3:
164                 cout << "Masukkan Nama Pegawai: ";
165                 cin.ignore();
166                 getline(cin, name);
167                 cout << "Masukkan Nama Proyek: ";
168                 cin.ignore();
169                 getline(cin, projectName);
170                 cout << "Masukkan Durasi Proyek (bulan): ";
171                 cin >> duration;
172                 {
173                     Employee* temp = head;
174                     while (temp != nullptr && temp->employeeName != name) {
175                         temp = temp->nextEmployee;
176                     }
177                     if (temp != nullptr) {
178                         addProject(temp, projectName, duration);
179                     } else {
180                         cout << "Proyek " << projectName << " berhasil ditambahkan ke pegawai dengan nama " << name << ".\n";
181                     }
182                     cout << "Pegawai dengan nama " << name << " tidak ditemukan.\n";
183                 }
184                 break;
185             case 4:
186                 cout << "Masukkan ID Pegawai: ";
187                 cin >> id;
188                 cout << "Masukkan Nama Proyek yang akan dihapus: ";
189                 cin.ignore();
190                 getline(cin, projectName);
191                 {
192                     Employee* temp = head;
193                     while (temp != nullptr && temp->employeeID != id) {
194                         temp = temp->nextEmployee;
195                     }
196                     if (temp != nullptr) {
197                         removeProject(temp, projectName);
198                     } else {
199                         cout << "Pegawai dengan ID " << id << " tidak ditemukan.\n";
200                     }
201                 }
202                 break;
203             case 5:
204                 displayEmployeesAndProjects();
205                 break;
206             case 6:
207                 cout << "Keluar dari program.\n";
208                 break;
209             default:
210                 cout << "Pilihan tidak valid.\n";
211         }
212     } while (choice != 6);
213     return 0;
214 }

```

Sistem Manajemen Data Proyek dan Pegawai:

1. Tambahkan Pegawai
2. Tambahkan Proyek ke Pegawai
3. Tambahkan Proyek Baru
4. Hapus Proyek
5. Tampilkan Data Pegawai
6. Keluar

Pilih opsi: 1

Masukkan jumlah pegawai yang ingin ditambahkan: 1

Masukkan Nama Pegawai: John Doe

Masukkan ID Pegawai: JD001

Sistem Manajemen Data Proyek dan Pegawai:

1. Tambahkan Pegawai
2. Tambahkan Proyek ke Pegawai
3. Tambahkan Proyek Baru
4. Hapus Proyek
5. Tampilkan Data Pegawai
6. Keluar

Pilih opsi: 2

Masukkan ID Pegawai: JD001

Masukkan Nama Proyek: Proyek A

Masukkan Durasi Proyek (bulan): 6

Sistem Manajemen Data Proyek dan Pegawai:

1. Tambahkan Pegawai
2. Tambahkan Proyek ke Pegawai
3. Tambahkan Proyek Baru
4. Hapus Proyek
5. Tampilkan Data Pegawai
6. Keluar

Pilih opsi: 5

Nama Pegawai	ID	Proyek dan Durasi
John Doe	JD001	- Proyek A (6 bulan)

Sistem Manajemen Data Proyek dan Pegawai:

1. Tambahkan Pegawai
2. Tambahkan Proyek ke Pegawai
3. Tambahkan Proyek Baru
4. Hapus Proyek

```

Pilih opsi: 5
Nama Pegawai  ID      Proyek dan Durasi
-----
John Doe      JD001    - Proyek A (6 bulan)

Sistem Manajemen Data Proyek dan Pegawai:
1. Tambahkan Pegawai
2. Tambahkan Proyek ke Pegawai
3. Tambahkan Proyek Baru
4. Hapus Proyek
5. Tampilkan Data Pegawai
6. Keluar
Pilih opsi: 4
Masukkan ID Pegawai: JD001
Masukkan Nama Proyek yang akan dihapus: Proyek A
Proyek Proyek A berhasil dihapus.
Sistem Manajemen Data Proyek dan Pegawai:
1. Tambahkan Pegawai
2. Tambahkan Proyek ke Pegawai
3. Tambahkan Proyek Baru
4. Hapus Proyek
5. Tampilkan Data Pegawai
6. Keluar
Pilih opsi: 5
Nama Pegawai  ID      Proyek dan Durasi
-----
John Doe      JD001    -

Sistem Manajemen Data Proyek dan Pegawai:
1. Tambahkan Pegawai
2. Tambahkan Proyek ke Pegawai
3. Tambahkan Proyek Baru
4. Hapus Proyek
5. Tampilkan Data Pegawai
6. Keluar

```

### Struktur Data:

#### Project:

Menyimpan informasi proyek, seperti nama dan durasi.

Pointer nextProject menghubungkan proyek berikutnya.

#### Employee:

Menyimpan informasi pegawai, seperti nama dan ID.

Pointer projectHead menghubungkan daftar proyek yang dikerjakan pegawai.

Pointer nextEmployee menghubungkan pegawai berikutnya dalam daftar.

#### Fitur Utama:

createEmployee dan createProject:

Membuat node baru untuk pegawai atau proyek.

addEmployee:

Menambahkan pegawai baru ke dalam daftar.

addProject:

Menambahkan proyek ke pegawai tertentu.

removeProject:

Menghapus proyek dari daftar proyek pegawai tertentu berdasarkan nama proyek.

displayEmployees:

Menampilkan daftar pegawai beserta proyek-proyek mereka dalam format tabel.

Fungsi Utama (main):

Menu Interaktif:

Tambahkan Pegawai: Menambah pegawai baru.

Tambahkan Proyek ke Pegawai: Menambah proyek ke pegawai berdasarkan ID.

Tambahkan Proyek Baru: Menambah proyek ke pegawai berdasarkan nama.

Hapus Proyek: Menghapus proyek dari pegawai tertentu.

Tampilkan Data Pegawai: Menampilkan seluruh pegawai beserta proyek mereka.

Keluar: Mengakhiri program.

```

1 #include <iostream>
2 #include <iomanip>
3 #include <string>
4 using namespace std;
5
6 // Struktur untuk Buku
7 struct Book {
8     string title;
9     string returnDate;
10    Book* nextBook;
11 };
12
13 // Struktur untuk Anggota
14 struct Member {
15     string memberName;
16     string memberID;
17     Book* bookHead;
18     Member* nextMember;
19 };
20
21 // Fungsi untuk membuat anggota baru
22 Member* createMember(string name, string id) {
23     Member* newMember = new Member;
24     newMember->memberName = name;
25     newMember->memberID = id;
26     newMember->bookHead = nullptr;
27     newMember->nextMember = nullptr;
28     return newMember;
29 }
30
31 // Fungsi untuk membuat buku baru
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 // Menambahkan buku ke anggota
41 void addBook(Member* member, string title, string returnDate) {
42     Book* newBook = createBook(title, returnDate);
43     if (member->bookHead == nullptr) {
44         member->bookHead = newBook;
45     } else {
46         Book* temp = member->bookHead;
47         while (temp->nextBook != nullptr) {
48             temp = temp->nextBook;
49         }
50         temp->nextBook = newBook;
51     }
52 }
53
54 // Menghapus anggota beserta buku yang dipinjam
55 void removeMember(Member* head, string id) {
56     Member* temp = head;
57     Member* prev = nullptr;
58
59     while (temp != nullptr && temp->memberID != id) {
60         prev = temp;
61         temp = temp->nextMember;
62     }
63
64     if (temp == nullptr) {
65         cout << "Anggota dengan ID " << id << " tidak ditemukan.\n";
66         return;
67     }
68
69     if (prev == nullptr) {
70         head = temp->nextMember;
71     } else {
72         prev->nextMember = temp->nextMember;
73     }
74
75     // Hapus semua buku yang dipinjam anggota
76     Book* bookTemp = temp->bookHead;
77     while (bookTemp != nullptr) {
78         Book* toDelete = bookTemp;
79         bookTemp = bookTemp->nextBook;
80         delete toDelete;
81     }
82
83     delete temp;
84     cout << "Anggota dengan ID " << id << " beserta buku yang dipinjam berhasil dihapus.\n";
85 }
86
87 // Menampilkan data anggota dan buku yang dipinjam
88 void displayMembers(Member* head) {
89     Member* tempMember = head;
90     int bookNumber = 1;
91     while (tempMember != nullptr) {
92         Book* tempBook = tempMember->bookHead;
93         while (tempBook != nullptr) {
94             cout << "Buku " << bookNumber << ": Judul = \"<tempBook->title << "\", Pengembalian = \"<tempBook->returnDate << "\" (Untuk " << tempMember->memberName << ").\n";
95             tempBook = tempBook->nextBook;
96             bookNumber++;
97         }
98         tempMember = tempMember->nextMember;
99     }
100 }
101
102 int main() {
103     Member* head = nullptr;
104
105     // Menambahkan data anggota
106     head = createMember("Rani", "A001");
107     head->nextMember = createMember("Dito", "A002");
108     head->nextMember->nextMember = createMember("Vina", "A003");
109
110     cout << "Data Awal Anggota Perpustakaan:\n\n";
111     cout << "Anggota 1: Nama = \"Rani\", ID = \"A001\".\n";
112     cout << "Anggota 2: Nama = \"Dito\", ID = \"A002\".\n";
113     cout << "Anggota 3: Nama = \"Vina\", ID = \"A003\".\n\n";
114
115     cout << "Setelah menambahkan buku yang dipinjam:\n\n";
116
117     // Menambahkan buku yang dipinjam
118     addBook(head, "Pemrograman C++", "01/12/2024");
119     addBook(head->nextMember, "Algoritma Pemrograman", "15/12/2024");
120     displayMembers(head);
121
122     // Menambahkan buku baru untuk Rani
123     cout << "Setelah menambahkan buku baru:\n\n";
124     addBook(head, "Struktur Data", "18/12/2024");
125     displayMembers(head);
126
127     // Menghapus anggota Dito beserta buku yang dipinjam
128     cout << "Setelah menghapus anggota Dito:\n\n";
129     removeMember(head, "A002");
130     displayMembers(head);
131
132     return 0;
133 }

```

```

o Data Awal Anggota Perpustakaan:

Anggota 1: Nama = "Rani", ID = "A001".
Anggota 2: Nama = "Dito", ID = "A002".
Anggota 3: Nama = "Vina", ID = "A003".

Setelah menambahkan buku yang dipinjam:

- Buku 1: Judul = "Pemrograman C++", Pengembalian = "01/12/2024" (Untuk Rani).
- Buku 2: Judul = "Algoritma Pemrograman", Pengembalian = "15/12/2024" (Untuk Dito).

Setelah menambahkan buku baru:

- Buku 1: Judul = "Pemrograman C++", Pengembalian = "01/12/2024" (Untuk Rani).
- Buku 2: Judul = "Struktur Data", Pengembalian = "10/12/2024" (Untuk Rani).
- Buku 3: Judul = "Algoritma Pemrograman", Pengembalian = "15/12/2024" (Untuk Dito).

Setelah menghapus anggota Dito:

Anggota dengan ID A002 beserta buku yang dipinjam berhasil dihapus.
- Buku 1: Judul = "Pemrograman C++", Pengembalian = "01/12/2024" (Untuk Rani).
- Buku 2: Judul = "Struktur Data", Pengembalian = "10/12/2024" (Untuk Rani).
PS C:\pertemuan 13>

```

## Struktur Data:

**Book:** Menyimpan informasi buku seperti judul dan tanggal pengembalian, serta pointer ke buku berikutnya.

**Member:** Menyimpan informasi anggota (nama, ID) dan pointer ke daftar buku yang dipinjam (linked list dari Book), serta pointer ke anggota berikutnya.

## Fungsi Utama:

**createMember:** Membuat anggota baru dengan nama dan ID.

**createBook:** Membuat buku baru dengan judul dan tanggal pengembalian.

**addBook:** Menambahkan buku baru ke daftar buku anggota.

**removeMember:** Menghapus anggota beserta semua buku yang dipinjam.

**displayMembers:** Menampilkan semua anggota dan buku yang dipinjam.

## Alur Program:

Program dimulai dengan membuat tiga anggota (Rani, Dito, dan Vina).

Buku ditambahkan ke anggota-anggota tersebut.

Buku baru ditambahkan untuk anggota Rani.

Anggota Dito dihapus bersama buku-buku yang dipinjam.

## Hasil Output:

Program menampilkan status anggota dan buku yang dipinjam pada setiap tahap: setelah penambahan buku, penambahan buku baru untuk Rani, dan setelah

penghapusan anggota Dito.

## **V. KESIMPULAN**

Multi Linked List adalah struktur data yang menghubungkan berbagai tipe data menggunakan pointer yang saling terkait, memungkinkan pengelolaan data yang lebih kompleks, seperti anggota perpustakaan dan buku yang mereka pinjam. Kelebihannya adalah fleksibilitas dalam menyimpan data, namun lebih memerlukan memori dan pengelolaan yang lebih rumit.