

**LAPORAN PRAKTIKUM**

**MODUL 13**

**MULTI LINKED LIST**



**Disusun Oleh:**

**Rizaldy Aulia Rachman (2311104051)**

**S1SE-07-02**

**Dosen :**

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

**PROGRAM STUDI S1 SOFTWARE ENGINEERING**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## **I. TUJUAN**

1. Memahami penggunaan *Multi Linked list*.
2. Mengimplementasikan *Multi Linked list* dalam beberapa studi kasus.

## **II. LANDASAN TEORI**

### **2.1 Multi Linked List**

*Linked list 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 .

### **2.2 Kelebihan Multi Linked List**

1. Fleksibilitas Tinggi
  - o Memungkinkan representasi hubungan kompleks antara data, seperti relasi hierarki, graf, atau tabel multidimensi.
2. Efisiensi dalam Menyimpan Data Terkait
  - o Struktur ini memungkinkan setiap node untuk menyimpan beberapa referensi ke data yang relevan atau terkait.
3. Akses Relasi Lebih Cepat
  - o Jika data memiliki banyak relasi atau koneksi, multi linked list memungkinkan navigasi lebih cepat melalui pointer tambahan.
4. Cocok untuk Struktur Data Kompleks
  - o Digunakan dalam representasi graf atau multidimensional arrays, di mana konektivitas antar elemen sangat penting.

### **2.3 Kekurangan Multi Linked List**

1. Pengelolaan Pointer Lebih Rumit
  - o Karena setiap node memiliki banyak pointer, pengelolaan memori dan operasi seperti *insert* atau *delete* menjadi lebih kompleks.
2. Penggunaan Memori Lebih Besar
  - o Adanya lebih dari satu pointer per node menyebabkan kebutuhan memori meningkat dibandingkan *single linked list*.
3. Sulit untuk Diimplementasikan
  - o Dibutuhkan pemahaman lebih dalam tentang struktur data dan algoritma karena strukturnya lebih kompleks.

#### 4. Proses Traversal Rumit

- Traversal pada *multi linked list* memerlukan perhatian ekstra untuk menghindari pointer yang tidak valid atau perulangan tanpa akhir.

### III. GUIDED

#### 1. Guided1

Code:

```
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         while (head != nullptr) {
62             Node* temp = head;
63             head = head->next;
64
65             while (temp->child != nullptr) {
66                 Node* childTemp = temp->child;
67                 temp->child = temp->child->next;
68                 delete childTemp;
69             }
70             delete temp;
71         }
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
```

Output:

```
Parent: 3 -> 30 30
Parent: 2 -> 20 20
Parent: 1 -> 11 10
PS C:\Praktikum Struktur data\pertemuan10\output>
```

## 2. Guided2

Code:

```

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             while (temp->subordinate != nullptr) {
67                 EmployeeNode* subTemp = temp->subordinate;
68                 temp->subordinate = temp->subordinate->next;
69                 delete subTemp;
70             }
71             delete temp;
72         }
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

```

Output:

```
Manager: Charlie -> Brian Frans  
Manager: Bob -> Frank  
Manager: Alice -> Eve David  
PS C:\Praktikum Struktur data\pertemuan10\output>
```

### 3. Guided3

Code:

```

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
13 EmployeeNode(string empName) : name(empName), next(nullptr), subordinate(nullptr) {}
14
15 // Kelas untuk daftar 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 (tambah)
24     void addEmployee(string name) {
25         EmployeeNode* newEmployee = new EmployeeNode(name);
26         EmployeeNode* next = head; // Menyimpan pointer ke karyawan sebelumnya
27         head = newEmployee; // Menjadikan 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             manager->subordinate = newSubordinate; // Menyambahkan ke subordinate sebelumnya
39             manager->subordinate->next = newSubordinate; // Menambahkan subordinate
40         } else {
41             cout << "Manager not found!" << endl;
42         }
43     }
44
45     // Menghapus karyawan (hapus)
46     void deleteEmployee(string name) {
47         EmployeeNode* current = head;
48         while (current != nullptr && (current->name != name)) {
49             current = current->next;
50         }
51         if (current != nullptr) { // Jika karyawan ditemukan
52             EmployeeNode* toDelete = current;
53             *current = *current->next;
54
55             // hapus semua subordinate dari node ini
56             while (toDelete->subordinate != nullptr) {
57                 EmployeeNode* subTemp = toDelete->subordinate;
58                 toDelete->subordinate = toDelete->subordinate->next;
59                 delete subTemp;
60             }
61             delete toDelete;
62             cout << "Employee " << name << " deleted." << endl;
63         } else {
64             cout << "Employee not found!" << endl;
65         }
66     }
67
68     // Menghapus subordinate dari karyawan tertentu
69     void deleteSubordinate(string managerName, string subordinateName) {
70         EmployeeNode* manager = head;
71         while (manager != nullptr && manager->name != managerName) {
72             manager = manager->next;
73         }
74         if (manager != nullptr) { // Jika manager ditemukan
75             EmployeeNode* currentSub = manager->subordinate;
76             while (currentSub != nullptr && (currentSub->name != subordinateName)) {
77                 currentSub = currentSub->next;
78             }
79             if (currentSub != nullptr) { // Jika subordinate ditemukan
80                 EmployeeNode* toDelete = currentSub;
81                 *currentSub = *currentSub->next; // Menghapus dari list
82                 delete toDelete; // Menghapus node subordinate
83                 cout << "Subordinate " << subordinateName << " deleted from " << managerName << "." << endl;
84             } else {
85                 cout << "Subordinate not found!" << endl;
86             }
87         } else {
88             cout << "Manager not found!" << endl;
89         }
90     }
91
92     // Menampilkan daftar karyawan dan subordinate mereka
93     void display() {
94         EmployeeNode* current = head;
95         while (current != nullptr) {
96             cout << "Manager: " << current->name << " -> ";
97             EmployeeNode* sub = current->subordinate;
98             while (sub != nullptr) {
99                 cout << sub->name << " ";
100                 sub = sub->next;
101             }
102             cout << endl;
103             current = current->next;
104         }
105     }
106
107 ~EmployeeList() {
108     // Destructor untuk membebaskan memori
109     while (head != nullptr) {
110         EmployeeNode* temp = head;
111         head = head->next;
112         delete temp;
113     }
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 list 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 list daftar setelah penghapusan
145
146     return 0;
147 }
148

```

Output:



```
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 C:\Praktikum Struktur data\pertemuan10\output>
```

#### IV. UNGUIDED

##### 1. Manajemen Data Pegawai dan Proyek

Buatlah program menggunakan Multi Linked List untuk menyimpan data pegawai dan proyek yang dikelola setiap pegawai.

- Setiap pegawai memiliki data: Nama Pegawai dan ID Pegawai.
- Setiap proyek memiliki data: Nama Proyek\*\* dan \*\*Durasi (bulan).

Instruksi:

##### 1. Masukkan data pegawai berikut:

- Pegawai 1: Nama = "Andi", ID = "P001".
- Pegawai 2: Nama = "Budi", ID = "P002".
- Pegawai 3: Nama = "Citra", ID = "P003".

##### 2. Tambahkan proyek ke pegawai:

- Proyek 1: Nama = "Aplikasi Mobile", Durasi = 12 bulan (Untuk Andi).
- Proyek 2: Nama = "Sistem Akuntansi", Durasi = 8 bulan (Untuk Budi).
- Proyek 3: Nama = "E-commerce", Durasi = 10 bulan (Untuk Citra).

##### 3. Tambahkan proyek baru:

- Proyek 4: Nama = "Analisis Data", Durasi = 6 bulan (Untuk Andi).

##### 4. Hapus proyek "Aplikasi Mobile" dari Andi.

##### 5. Tampilkan data pegawai dan proyek mereka.

Jawaban:

Code:

```

1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 struct Proyek {
6     string namaProyek;
7     int durasi;
8     Proyek* nextProyek;
9 };
10
11 struct Pegawai {
12     string namaPegawai;
13     string idPegawai;
14     Proyek* headProyek;
15     Pegawai* nextPegawai;
16 };
17
18 Pegawai* headPegawai = nullptr;
19
20 void tambahPegawai(string nama, string id) {
21     Pegawai* pegawaiBaru = new Pegawai;
22     pegawaiBaru->namaPegawai = nama;
23     pegawaiBaru->idPegawai = id;
24     pegawaiBaru->headProyek = nullptr;
25     pegawaiBaru->nextPegawai = headPegawai;
26     headPegawai = pegawaiBaru;
27 }
28
29 void tambahProyek(string idPegawai, string namaProyek, int durasi) {
30     Pegawai* currPegawai = headPegawai;
31     while (currPegawai != nullptr) {
32         if (currPegawai->idPegawai == idPegawai) {
33             Proyek* proyekBaru = new Proyek;
34             proyekBaru->namaProyek = namaProyek;
35             proyekBaru->durasi = durasi;
36             proyekBaru->nextProyek = currPegawai->headProyek;
37             currPegawai->headProyek = proyekBaru;
38             return;
39         }
40         currPegawai = currPegawai->nextPegawai;
41     }
42     cout << "Pegawai dengan ID " << idPegawai << " tidak ditemukan!\n";
43 }
44
45 void hapusProyek(string idPegawai, string namaProyek) {
46     Pegawai* currPegawai = headPegawai;
47     while (currPegawai != nullptr) {
48         if (currPegawai->idPegawai == idPegawai) {
49             Proyek* currProyek = currPegawai->headProyek;
50             Proyek* prevProyek = nullptr;
51             while (currProyek != nullptr) {
52                 if (currProyek->namaProyek == namaProyek) {
53                     if (prevProyek == nullptr) {
54                         currPegawai->headProyek = currProyek->nextProyek;
55                     } else {
56                         prevProyek->nextProyek = currProyek->nextProyek;
57                     }
58                     delete currProyek;
59                     return;
60                 }
61                 prevProyek = currProyek;
62                 currProyek = currProyek->nextProyek;
63             }
64         }
65         currPegawai = currPegawai->nextPegawai;
66     }
67     cout << "Proyek " << namaProyek << " tidak ditemukan untuk pegawai dengan ID " << idPegawai << "!\n";
68 }
69
70 void tampilkanData() {
71     Pegawai* currPegawai = headPegawai;
72     while (currPegawai != nullptr) {
73         cout << "Pegawai " << currPegawai->namaPegawai << " (ID: " << currPegawai->idPegawai << ")\n";
74         Proyek* currProyek = currPegawai->headProyek;
75         while (currProyek != nullptr) {
76             cout << "  Proyek: " << currProyek->namaProyek << " (Durasi: " << currProyek->durasi << " bulan)\n";
77             currProyek = currProyek->nextProyek;
78         }
79         currPegawai = currPegawai->nextPegawai;
80     }
81 }
82
83 int main() {
84     int pilihan;
85     do {
86         cout << "\nMenu:\n";
87         cout << "1. Tambah Pegawai\n";
88         cout << "2. Tambah Proyek\n";
89         cout << "3. Hapus Proyek\n";
90         cout << "4. Tampilkan Data\n";
91         cout << "5. Keluar\n";
92         cout << "Pilihan: ";
93         cin >> pilihan;
94         cin.ignore();
95
96         if (pilihan == 1) {
97             string nama, id;
98             cout << "Masukkan Nama Pegawai: ";
99             getline(cin, nama);
100             cout << "Masukkan ID Pegawai: ";
101             getline(cin, id);
102             tambahPegawai(nama, id);
103         } else if (pilihan == 2) {
104             string id, namaProyek;
105             int durasi;
106             cout << "Masukkan ID Pegawai: ";
107             getline(cin, id);
108             cout << "Masukkan Nama Proyek: ";
109             getline(cin, namaProyek);
110             cout << "Masukkan Durasi Proyek (bulan): ";
111             cin >> durasi;
112             cin.ignore();
113             tambahProyek(id, namaProyek, durasi);
114         } else if (pilihan == 3) {
115             string id, namaProyek;
116             cout << "Masukkan ID Pegawai: ";
117             getline(cin, id);
118             cout << "Masukkan Nama Proyek yang akan dihapus: ";
119             getline(cin, namaProyek);
120             hapusProyek(id, namaProyek);
121         } else if (pilihan == 4) {
122             tampilkanData();
123         } else if (pilihan != 5) {
124             cout << "Pilihan tidak valid!\n";
125         }
126     } while (pilihan != 5);
127
128     return 0;
129 }
130

```

## 1. Output intruksi 1:

```
Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 1
Masukkan Nama Pegawai: Andi
Masukkan ID Pegawai: P001

Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 1
Masukkan Nama Pegawai: Budi
Masukkan ID Pegawai: P002

Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 1
Masukkan Nama Pegawai: Citra
Masukkan ID Pegawai: P003
```

## 2. Output intruksi 2:

```
Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 2
Masukkan ID Pegawai: P001
Masukkan Nama Proyek: Aplikasi Mobile
Masukkan Durasi Proyek (bulan): 12

Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 2
Masukkan ID Pegawai: P002
Masukkan Nama Proyek: Sistem Akuntansi
Masukkan Durasi Proyek (bulan): 8

Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 2
Masukkan ID Pegawai: P003
Masukkan Nama Proyek: E-commerce
Masukkan Durasi Proyek (bulan): 10
```

## 3. Output intruksi 3:

```

Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 2
Masukkan ID Pegawai: P001
Masukkan Nama Proyek: Analisis Data
Masukkan Durasi Proyek (bulan): 6

```

4. Output intruksi 4:

```

Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 3
Masukkan ID Pegawai: P001
Masukkan Nama Proyek yang akan dihapus: Aplikasi Mobile

```

5. Output intruksi 5:

```

Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 4
Pegawai: Andi (ID: P001)
    Proyek: Analisis Data (Durasi: 6 bulan)
Pegawai: Citra (ID: P003)
    Proyek: E-commerce (Durasi: 10 bulan)
Pegawai: Budi (ID: P002)
    Proyek: Sistem Akuntansi (Durasi: 8 bulan)

```

## 2. Sistem Manajemen Buku Perpustakaan

Gunakan Multi Linked List untuk menyimpan data anggota perpustakaan dan daftar buku yang dipinjam.

- Setiap anggota memiliki data: Nama Anggota dan ID Anggota.
- Setiap buku memiliki data: Judul Buku dan Tanggal Pengembalian.

Instruksi:

1. Masukkan data anggota berikut:

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

2. Tambahkan 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).
3. Tambahkan buku baru:
- Buku 3: Judul = "Struktur Data", Pengembalian = "10/12/2024" (Untuk Rani).
4. Hapus anggota Dito beserta buku yang dipinjam.
5. Tampilkan seluruh data anggota dan buku yang dipinjam.

Jawaban:

Code:

```

1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 struct Buku {
6     string judulBuku;
7     string tanggalPengembalian;
8     Buku* nextBuku;
9 };
10
11 struct Anggota {
12     string namaAnggota;
13     string idAnggota;
14     Buku* headBuku;
15     Anggota* nextAnggota;
16 };
17
18 Anggota* headAnggota = nullptr;
19
20 void tambahAnggota(string nama, string id) {
21     Anggota* anggotaBaru = new Anggota;
22     anggotaBaru->namaAnggota = nama;
23     anggotaBaru->idAnggota = id;
24     anggotaBaru->headBuku = nullptr;
25     anggotaBaru->nextAnggota = headAnggota;
26     headAnggota = anggotaBaru;
27 }
28
29 void tambahBuku(string idAnggota, string judulBuku, string tanggalPengembalian) {
30     Anggota* currAnggota = headAnggota;
31     while (currAnggota != nullptr) {
32         if (currAnggota->idAnggota == idAnggota) {
33             Buku* bukuBaru = new Buku;
34             bukuBaru->judulBuku = judulBuku;
35             bukuBaru->tanggalPengembalian = tanggalPengembalian;
36             bukuBaru->nextBuku = currAnggota->headBuku;
37             currAnggota->headBuku = bukuBaru;
38             return;
39         }
40         currAnggota = currAnggota->nextAnggota;
41     }
42     cout << "Anggota dengan ID " << idAnggota << " tidak ditemukan!\n";
43 }
44
45 void hapusAnggota(string idAnggota) {
46     Anggota* currAnggota = headAnggota;
47     Anggota* prevAnggota = nullptr;
48     while (currAnggota != nullptr) {
49         if (currAnggota->idAnggota == idAnggota) {
50             if (prevAnggota == nullptr) {
51                 headAnggota = currAnggota->nextAnggota;
52             } else {
53                 prevAnggota->nextAnggota = currAnggota->nextAnggota;
54             }
55             // Hapus semua buku yang dipinjam anggota
56             Buku* currBuku = currAnggota->headBuku;
57             while (currBuku != nullptr) {
58                 Buku* temp = currBuku;
59                 currBuku = currBuku->nextBuku;
60                 delete temp;
61             }
62             delete currAnggota;
63             return;
64         }
65         prevAnggota = currAnggota;
66         currAnggota = currAnggota->nextAnggota;
67     }
68     cout << "Anggota dengan ID " << idAnggota << " tidak ditemukan!\n";
69 }
70
71 void tampilkanData() {
72     Anggota* currAnggota = headAnggota;
73     while (currAnggota != nullptr) {
74         cout << "Anggota: " << currAnggota->namaAnggota << " (ID: " << currAnggota->idAnggota << ")\n";
75         Buku* currBuku = currAnggota->headBuku;
76         while (currBuku != nullptr) {
77             cout << "  Buku: " << currBuku->judulBuku << " (Pengembalian: " << currBuku->tanggalPengembalian << ")\n";
78             currBuku = currBuku->nextBuku;
79         }
80         currAnggota = currAnggota->nextAnggota;
81     }
82 }
83
84 int main() {
85     int pilihan;
86     do {
87         cout << "\nMenu:\n";
88         cout << "1. Tambah Anggota\n";
89         cout << "2. Tambah Buku\n";
90         cout << "3. Hapus Anggota\n";
91         cout << "4. Tampilkan Data\n";
92         cout << "5. Keluar\n";
93         cout << "Pilihan: ";
94         cin >> pilihan;
95         cin.ignore();
96
97         if (pilihan == 1) {
98             string nama, id;
99             cout << "Masukkan Nama Anggota: ";
100             getline(cin, nama);
101             cout << "Masukkan ID Anggota: ";
102             getline(cin, id);
103             tambahAnggota(nama, id);
104         } else if (pilihan == 2) {
105             string id, judulBuku, tanggalPengembalian;
106             cout << "Masukkan ID Anggota: ";
107             getline(cin, id);
108             cout << "Masukkan Judul Buku: ";
109             getline(cin, judulBuku);
110             cout << "Masukkan Tanggal Pengembalian: ";
111             getline(cin, tanggalPengembalian);
112             tambahBuku(id, judulBuku, tanggalPengembalian);
113         } else if (pilihan == 3) {
114             string id;
115             cout << "Masukkan ID Anggota yang akan dihapus: ";
116             getline(cin, id);
117             hapusAnggota(id);
118         } else if (pilihan == 4) {
119             tampilkanData();
120         } else if (pilihan != 5) {
121             cout << "Pilihan tidak valid.\n";
122         }
123     } while (pilihan != 5);
124
125     return 0;
126 }
127

```

1. Output intruksi 1:

```
Menu:
1. Tambah Anggota
2. Tambah Buku
3. Hapus Anggota
4. Tampilkan Data
5. Keluar
Pilihan: 1
Masukkan Nama Anggota: Rani
Masukkan ID Anggota: A001

Menu:
1. Tambah Anggota
2. Tambah Buku
3. Hapus Anggota
4. Tampilkan Data
5. Keluar
Pilihan: 1
Masukkan Nama Anggota: Dito
Masukkan ID Anggota: A002

Menu:
1. Tambah Anggota
2. Tambah Buku
3. Hapus Anggota
4. Tampilkan Data
5. Keluar
Pilihan: 1
Masukkan Nama Anggota: Vina
Masukkan ID Anggota: A003
```

2. Output intruksi 2:

```
Menu:
1. Tambah Anggota
2. Tambah Buku
3. Hapus Anggota
4. Tampilkan Data
5. Keluar
Pilihan: 2
Masukkan ID Anggota: A001
Masukkan Judul Buku: Pemrograman C++
Masukkan Tanggal Pengembalian: 01/12/2024

Menu:
1. Tambah Anggota
2. Tambah Buku
3. Hapus Anggota
4. Tampilkan Data
5. Keluar
Pilihan: 2
Masukkan ID Anggota: A002
Masukkan Judul Buku: Algoritma Pemrograman
Masukkan Tanggal Pengembalian: 15/12/2024
```

3. Output intruksi 3:

```
Menu:
1. Tambah Anggota
2. Tambah Buku
3. Hapus Anggota
4. Tampilkan Data
5. Keluar
Pilihan: 2
Masukkan ID Anggota: A001
Masukkan Judul Buku: Struktur Data
Masukkan Tanggal Pengembalian: 10/12/2024
```

4. Output intruksi 4:

```
Menu:
1. Tambah Anggota
2. Tambah Buku
3. Hapus Anggota
4. Tampilkan Data
5. Keluar
Pilihan: 3
Masukkan ID Anggota yang akan dihapus: A002
```

5. Output intruksi 5:

```
Menu:
1. Tambah Anggota
2. Tambah Buku
3. Hapus Anggota
4. Tampilkan Data
5. Keluar
Pilihan: 4
Anggota: Vina (ID: A003)
Anggota: Rani (ID: A001)
Buku: Struktur Data (Pengembalian: 10/12/2024)
Buku: Pemrograman C++ (Pengembalian: 01/12/2024)
```