

# **LAPORAN PRAKTIKUM**

## **MODUL 13**

### **MULTI LINKED LIST**



**Disusun Oleh :**

Farhan Kurniawan (2311104073)

**Kelas:**

SE-07-2

**Dosen :**

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

**PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK**

**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**

Penggunaan Multi Linked List merupakan salah satu teknik pengolahan data yang efektif dalam struktur data. Secara konsep, Multi Linked List adalah pengembangan dari Linked List biasa, dimana setiap node memiliki lebih dari satu pointer yang mengarah ke node lainnya. Hal ini memungkinkan data disimpan dan diakses secara efisien dalam beberapa dimensi.

Dalam implementasinya, Multi Linked List dapat digunakan dalam berbagai studi kasus seperti pengolahan data kompleks, representasi graf, dan pengembangan database. Dengan memahami konsep dan penggunaan Multi Linked List, praktikan dapat mengembangkan kemampuan dalam merancang dan mengimplementasikan algoritma yang lebih efisien dan efektif. Oleh karena itu, praktikum ini bertujuan untuk memahami penggunaan Multi Linked List dan mengimplementasikannya dalam beberapa studi kasus nyata.

## **III. GUIDED**

## 1. Kode Program Guided 1:

```
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 }
```

```

1  void display() {
2      Node* current = head;
3      while (current != nullptr) {
4          cout << "Parent: " << current->data << " -> ";
5          Node* child = current->child;
6          while (child != nullptr) {
7              cout << child->data << " ";
8              child = child->next;
9          }
10         cout << endl;
11         current = current->next;
12     }
13 }
14
15 ~MultiLinkedList() {
16
17     while (head != nullptr) {
18         Node* temp = head;
19         head = head->next;
20
21         while (temp->child != nullptr) {
22             Node* childTemp = temp->child;
23             temp->child = temp->child->next;
24             delete childTemp;
25         }
26         delete temp;
27     }
28 }
29
30 };
31
32 int main() {
33     MultiLinkedList mList;
34
35     mList.addParent(1);
36     mList.addParent(2);
37     mList.addParent(3);
38
39     mList.addChild(1, 10);
40     mList.addChild(1, 11);
41     mList.addChild(2, 20);
42     mList.addChild(2, 20);
43     mList.addChild(3, 30);
44     mList.addChild(3, 30);
45     mList.display();
46
47     return 0;
48 }
49

```

Hasil Run:

```

PS C:\Users\Farhan Kurniawan\Downloads\data karyawan> cd "c:\
d1.cpp -o guided1 } ; if ($?) { .\guided1 }
Parent: 3 -> 30 30
Parent: 2 -> 20 20
Parent: 1 -> 11 10
PS C:\Users\Farhan Kurniawan\Downloads\data karyawan\output>

```

## 2. Kode Program Guided 2:

```

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

```

```

1 void display() {
2     EmployeeNode* current = head;
3     while (current != nullptr) {
4         cout << "Manager: " << current->name << " -> ";
5         EmployeeNode* sub = current->subordinate;
6         while (sub != nullptr) {
7             cout << sub->name << " ";
8             sub = sub->next;
9         }
10        cout << endl;
11        current = current->next;
12    }
13 }
14
15 ~EmployeeList() {
16
17     while (head != nullptr) {
18         EmployeeNode* temp = head;
19         head = head->next;
20
21         while (temp->subordinate != nullptr) {
22             EmployeeNode* subTemp = temp->subordinate;
23             temp->subordinate = temp->subordinate->next;
24             delete subTemp;
25         }
26         delete temp;
27     }
28 }
29
30 };
31
32 int main() {
33     EmployeeList emplList;
34
35     emplList.addEmployee("Alice");
36     emplList.addEmployee("Bob");
37     emplList.addEmployee("Charlie");
38
39     emplList.addSubordinate("Alice", "David");
40     emplList.addSubordinate("Alice", "Eve");
41     emplList.addSubordinate("Bob", "Frank");
42
43     emplList.addSubordinate("Charlie", "Frans");
44     emplList.addSubordinate("Charlie", "Brian");
45
46     emplList.display();
47
48     return 0;
49 }
50

```

Hasil Run:

```
PS C:\Users\Farhan Kurniawan\Downloads\data karyawan\output> cd
+ guided2.cpp -o guided2 } ; if ($?) { .\guided2 }
Manager: Charlie -> Brian Frans
Manager: Bob -> Frank
Manager: Alice -> Eve David
PS C:\Users\Farhan Kurniawan\Downloads\data karyawan\output>
```

3. Kode Program Guided 3:

```

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     // 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 manajer 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 manajer 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 }

```



Hasil Run:

```
PS C:\Users\Farhan Kurniawan\Downloads\data karyawan\output> cd
+ guided3.cpp -o guided3 } ; if ($?) { .\guided3 }
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:\Users\Farhan Kurniawan\Downloads\data karyawan\output>
```

## IV. UNGUIDED

### 1. Kode Program:

```
1 // Menampilkan daftar karyawan dan subordinate mereka
2 void display() {
3     EmployeeNode* current = head;
4     while (current != nullptr) {
5         cout << "Manager: " << current->name << " -> ";
6         EmployeeNode* sub = current->subordinate;
7         while (sub != nullptr) {
8             cout << sub->name << " ";
9             sub = sub->next;
10        }
11        cout << endl;
12        current = current->next;
13    }
14 }
15
16 ~EmployeeList() {
17     // Destructor untuk membersihkan memori
18     while (head != nullptr) {
19         EmployeeNode* temp = head;
20         head = head->next;
21
22         // Hapus semua subordinate dari node ini
23         while (temp->subordinate != nullptr) {
24             EmployeeNode* subTemp = temp->subordinate;
25             temp->subordinate = temp->subordinate->next;
26             delete subTemp;
27         }
28         delete temp;
29     }
30 }
31 };
32
33 int main() {
34     EmployeeList emplist;
35
36     emplist.addEmployee("Alice");
37     emplist.addEmployee("Bob");
38     emplist.addEmployee("Charlie");
39
40     emplist.addSubordinate("Alice", "David");
41     emplist.addSubordinate("Alice", "Eve");
42     emplist.addSubordinate("Bob", "Frank");
43
44     cout << "Initial employee list:" << endl;
45     emplist.display(); // Menampilkan isi daftar karyawan
46
47     emplist.deleteSubordinate("Alice", "David"); // Menghapus David dari Alice
48     emplist.deleteEmployee("Charlie"); // Menghapus Charlie
49
50     cout << "\nUpdated employee list:" << endl;
51     emplist.display(); // Menampilkan isi daftar setelah penghapusan
52
53     return 0;
54 }
55
```

```

1 // Fungsi untuk menghapus proyek tertentu dari pegawai
2 void hapusProyek(Pegawai* pegawai, string namaProyek) {
3     Proyek* temp = pegawai->headProyek;
4     Proyek* prev = NULL;
5
6     while (temp != NULL && temp->namaProyek != namaProyek) {
7         prev = temp;
8         temp = temp->next;
9     }
10
11     if (temp == NULL) {
12         cout << "Proyek \"" << namaProyek << "\" tidak ditemukan.\n";
13         return;
14     }
15
16     if (prev == NULL) { // proyek yang dihapus adalah proyek pertama
17         pegawai->headProyek = temp->next;
18     } else {
19         prev->next = temp->next;
20     }
21     delete temp;
22     cout << "Proyek \"" << namaProyek << "\" berhasil dihapus.\n";
23 }
24
25 // Fungsi untuk menampilkan data pegawai dan proyek mereka
26 void tampilkanData(Pegawai* headPegawai) {
27     Pegawai* tempPegawai = headPegawai;
28     while (tempPegawai != NULL) {
29         cout << "Pegawai: " << tempPegawai->namaPegawai
30             << " (ID: " << tempPegawai->idPegawai << ")\n";
31         Proyek* tempProyek = tempPegawai->headProyek;
32         if (tempProyek == NULL) {
33             cout << " Tidak ada proyek.\n";
34         } else {
35             while (tempProyek != NULL) {
36                 cout << " - Proyek: " << tempProyek->namaProyek
37                     << ", Durasi: " << tempProyek->durasi << " bulan\n";
38                 tempProyek = tempProyek->next;
39             }
40         }
41         tempPegawai = tempPegawai->next;
42         cout << endl;
43     }
44 }
45
46 int main() {
47     // Membuat daftar pegawai
48     Pegawai* headPegawai = buatPegawai("Andi", "P001");
49     headPegawai->next = buatPegawai("Budi", "P002");
50     headPegawai->next->next = buatPegawai("Citra", "P003");
51
52     // Menambahkan proyek ke pegawai
53     tambahProyek(headPegawai, "Aplikasi Mobile", 12); // Proyek untuk Andi
54     tambahProyek(headPegawai->next, "Sistem Akuntansi", 8); // Proyek untuk Budi
55     tambahProyek(headPegawai->next->next, "E-commerce", 10); // Proyek untuk Citra
56
57     // Menambahkan proyek baru ke Andi
58     tambahProyek(headPegawai, "Analisis Data", 6);
59
60     // Menghapus proyek "Aplikasi Mobile" dari Andi
61     hapusProyek(headPegawai, "Aplikasi Mobile");
62
63     // Menampilkan data pegawai dan proyek mereka
64     cout << "\nData Pegawai dan Proyek:\n";
65     tampilkanData(headPegawai);
66
67     return 0;
68 }
69

```

Hasil Run:

```
PS C:\Users\Farhan Kurniawan\Downloads\data karyawan> cd
ded1.cpp -o Unguided1 } ; if ($?) { .\Unguided1 }
Proyek "Aplikasi Mobile" berhasil dihapus.

Data Pegawai dan Proyek:
Pegawai: Andi (ID: P001)
  - Proyek: Analisis Data, Durasi: 6 bulan

Pegawai: Budi (ID: P002)
  - Proyek: Sistem Akuntansi, Durasi: 8 bulan

Pegawai: Citra (ID: P003)
  - Proyek: E-commerce, Durasi: 10 bulan

PS C:\Users\Farhan Kurniawan\Downloads\data karyawan\output>
```

2. Kode Program:



```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  // Struktur untuk Node Buku
6  struct Buku {
7      string judulBuku;
8      string tanggalPengembalian;
9      Buku* next; // Pointer ke buku berikutnya
10 };
11
12 // Struktur untuk Node Anggota
13 struct Anggota {
14     string namaAnggota;
15     string idAnggota;
16     Buku* headBuku; // Pointer ke buku pertama
17     Anggota* next; // Pointer ke anggota berikutnya
18 };
19
20 // Fungsi untuk membuat node anggota baru
21 Anggota* buatAnggota(string nama, string id) {
22     Anggota* anggotaBaru = new Anggota;
23     anggotaBaru->namaAnggota = nama;
24     anggotaBaru->idAnggota = id;
25     anggotaBaru->headBuku = NULL;
26     anggotaBaru->next = NULL;
27     return anggotaBaru;
28 }
29
30 // Fungsi untuk membuat node buku baru
31 Buku* buatBuku(string judul, string tanggal) {
32     Buku* bukuBaru = new Buku;
33     bukuBaru->judulBuku = judul;
34     bukuBaru->tanggalPengembalian = tanggal;
35     bukuBaru->next = NULL;
36     return bukuBaru;
37 }
38
39 // Fungsi untuk menambahkan buku ke anggota tertentu
40 void tambahBuku(Anggota* anggota, string judul, string tanggal) {
41     Buku* bukuBaru = buatBuku(judul, tanggal);
42     if (anggota->headBuku == NULL) {
43         anggota->headBuku = bukuBaru;
44     } else {
45         Buku* temp = anggota->headBuku;
46         while (temp->next != NULL) {
47             temp = temp->next; // Cari buku terakhir
48         }
49         temp->next = bukuBaru;
50     }
51 }
52
```

```

1 // Fungsi untuk menghapus anggota beserta semua buku yang dipinjam
2 void hapusAnggota(Anggota*& head, string id) {
3     Anggota* temp = head;
4     Anggota* prev = NULL;
5
6     while (temp != NULL && temp->idAnggota != id) {
7         prev = temp;
8         temp = temp->next;
9     }
10
11     if (temp == NULL) {
12         cout << "Anggota dengan ID \"< id << "\" tidak ditemukan.\n";
13         return;
14     }
15
16     // Hapus semua buku yang dipinjam oleh anggota
17     Buku* currentBuku = temp->headBuku;
18     while (currentBuku != NULL) {
19         Buku* toDelete = currentBuku;
20         currentBuku = currentBuku->next;
21         delete toDelete;
22     }
23
24     // Hapus anggota dari daftar
25     if (prev == NULL) {
26         head = temp->next; // Anggota pertama dihapus
27     } else {
28         prev->next = temp->next;
29     }
30
31     delete temp;
32     cout << "Anggota dengan ID \"< id << "\" berhasil dihapus beserta bukunya.\n";
33 }
34
35 // Fungsi untuk menampilkan seluruh data anggota dan buku yang dipinjam
36 void tampilkanData(Anggota* head) {
37     Anggota* tempAnggota = head;
38     while (tempAnggota != NULL) {
39         cout << "Anggota: " << tempAnggota->namaAnggota
40             << " (ID: " << tempAnggota->idAnggota << ")\n";
41         Buku* tempBuku = tempAnggota->headBuku;
42         if (tempBuku == NULL) {
43             cout << " Tidak ada buku yang dipinjam.\n";
44         } else {
45             while (tempBuku != NULL) {
46                 cout << " - Buku: " << tempBuku->judulBuku
47                     << ", Pengembalian: " << tempBuku->tanggalPengembalian << endl;
48                 tempBuku = tempBuku->next;
49             }
50         }
51         tempAnggota = tempAnggota->next;
52         cout << endl;
53     }
54 }
55
56 int main() {
57     // Membuat daftar anggota perpustakaan
58     Anggota* headAnggota = buatAnggota("Rani", "A001");
59     headAnggota->next = buatAnggota("Dito", "A002");
60     headAnggota->next->next = buatAnggota("Vina", "A003");
61
62     // Menambahkan buku yang dipinjam
63     tambahBuku(headAnggota, "Pemrograman C++", "01/12/2024"); // Untuk Rani
64     tambahBuku(headAnggota->next, "Algoritma Pemrograman", "15/12/2024"); // Untuk Dito
65
66     // Menambahkan buku baru untuk Rani
67     tambahBuku(headAnggota, "Struktur Data", "10/12/2024");
68
69     // Menghapus anggota Dito beserta bukunya
70     hapusAnggota(headAnggota, "A002");
71
72     // Menampilkan seluruh data anggota dan buku
73     cout << "\nData Anggota Perpustakaan dan Buku yang Dipinjam:\n";
74     tampilkanData(headAnggota);
75
76     return 0;
77 }
78

```

Hasil Run:

```
PS C:\Users\Farhan Kurniawan\Downloads\data karyawan> cd "
ded2.cpp -o Unguided2 } ; if ($?) { .\Unguided2 }
Anggota dengan ID "A002" berhasil dihapus beserta bukunya.

Data Anggota Perpustakaan dan Buku yang Dipinjam:
Anggota: Rani (ID: A001)
  - Buku: Pemrograman C++, Pengembalian: 01/12/2024
  - Buku: Struktur Data, Pengembalian: 10/12/2024
Anggota: Vina (ID: A003)
  Tidak ada buku yang dipinjam.

PS C:\Users\Farhan Kurniawan\Downloads\data karyawan\output>
```

## V. KESIMPULAN

Mempelajari Multi Linked List memberikan pemahaman mendalam tentang struktur data kompleks yang efektif. Dengan memahami konsep dan implementasi ini, mahasiswa dapat mengembangkan kemampuan dalam mengolah data kompleks, merancang algoritma efisien dan menganalisis masalah. Pemahaman ini juga membuka peluang aplikasi nyata dalam pengembangan perangkat lunak dan sistem informasi. Oleh karena itu, mempelajari Multi Linked List sangat penting bagi pengembangan kemampuan dalam bidang teknologi informasi.