

LAPORAN PRAKTIKUM
PERTEMUAN 13
MULTI LINKED LIST



Nama :

Salman Alfarisi (2311104036)

Dosen :

Yudha Islami Sulistya, S.Kom.,
M.Cs.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024

1. LATIHAN 1

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5
6  struct Proyek {
7      string nama_proyek;
8      int durasi;
9  };
10
11
12  struct Pegawai {
13      string nama_pegawai;
14      string id_pegawai;
15  };
16
17
18  struct ElmProyek {
19      Proyek info;
20      ElmProyek* next;
21  };
22
23
24  struct ElmPegawai {
25      Pegawai info;
26      ElmPegawai* next;
27      ElmProyek* firstProyek;
28  };
29
30
31  struct List {
32      ElmPegawai* first;
33  };
34
35
36  void createList(List &L) {
37      L.first = nullptr;
38  }
39
40
41  ElmPegawai* allocatePegawai(const Pegawai &data) {
42      ElmPegawai* P = new ElmPegawai;
43      P->info = data;
44      P->next = nullptr;
45      P->firstProyek = nullptr;
46      return P;
47  }
48
49
50  ElmProyek* allocateProyek(const Proyek &data) {
51      ElmProyek* P = new ElmProyek;
52      P->info = data;
53      P->next = nullptr;
54      return P;
55  }
56
57
58  void insertPegawai(List &L, ElmPegawai* P) {
59      if (L.first == nullptr) {
60          L.first = P;
61      } else {
62          P->next = L.first;
63          L.first = P;
64      }
65  }
66
67
68  void insertProyek(ElmPegawai* pegawai, ElmProyek* proyek) {
69      if (pegawai->firstProyek == nullptr) {
70          pegawai->firstProyek = proyek;
71      } else {
72          ElmProyek* Q = pegawai->firstProyek;
73          while (Q->next != nullptr) {
74              Q = Q->next;
75          }
76          Q->next = proyek;
77      }
78  }
79
80
```

```

80
81 void removeProyek(ElmPegawai* pegawai, const string &nama_proyek) {
82     if (pegawai->firstProyek == nullptr) return;
83
84     ElmProyek* P = pegawai->firstProyek;
85     ElmProyek* prev = nullptr;
86
87     while (P != nullptr) {
88         if (P->info.nama_proyek == nama_proyek) {
89             if (prev == nullptr) {
90                 pegawai->firstProyek = P->next;
91             } else {
92                 prev->next = P->next;
93             }
94             delete P;
95             cout << "Proyek " << nama_proyek << " berhasil dihapus!\n";
96             return;
97         }
98         prev = P;
99         P = P->next;
100     }
101     cout << "Proyek " << nama_proyek << " tidak ditemukan!\n";
102 }
103
104
105 void printList(const List &L) {
106     ElmPegawai* P = L.first;
107     while (P != nullptr) {
108         cout << "Pegawai: " << P->info.nama_pegawai << " (ID: " << P->info.id_pegawai << ")\n";
109         ElmProyek* Q = P->firstProyek;
110         while (Q != nullptr) {
111             cout << "    Proyek: " << Q->info.nama_proyek << ", Durasi: " << Q->info.durasi << " bulan\n";
112             Q = Q->next;
113         }
114         cout << endl;
115         P = P->next;
116     }
117 }
118
119 int main() {
120     List L;
121     createList(L);
122
123
124     Pegawai p1 = {"Andi", "P001"};
125     Pegawai p2 = {"Budi", "P002"};
126     Pegawai p3 = {"Citra", "P003"};
127
128
129     ElmPegawai* pegawai1 = allocatePegawai(p1);
130     ElmPegawai* pegawai2 = allocatePegawai(p2);
131     ElmPegawai* pegawai3 = allocatePegawai(p3);
132
133     insertPegawai(L, pegawai1);
134     insertPegawai(L, pegawai2);
135     insertPegawai(L, pegawai3);
136
137
138     Proyek proj1 = {"Aplikasi Mobile", 12};
139     Proyek proj2 = {"Sistem Akuntansi", 8};
140     Proyek proj3 = {"E-commerce", 10};
141     Proyek proj4 = {"Analisis Data", 6};
142
143
144     insertProyek(pegawai1, allocateProyek(proj1));
145     insertProyek(pegawai2, allocateProyek(proj2));
146     insertProyek(pegawai3, allocateProyek(proj3));
147     insertProyek(pegawai1, allocateProyek(proj4));
148
149     removeProyek(pegawai1, "Aplikasi Mobile");
150
151     cout << "Data Pegawai dan Proyek:\n";
152     printList(L);
153
154     return 0;
155 }
156

```

OUTPUT

```
Proyek Aplikasi Mobile berhasil dihapus!  
Data Pegawai dan Proyek:  
Pegawai: Citra (ID: P003)  
    Proyek: E-commerce, Durasi: 10 bulan  
  
Pegawai: Budi (ID: P002)  
    Proyek: Sistem Akuntansi, Durasi: 8 bulan  
  
Pegawai: Andi (ID: P001)  
    Proyek: Analisis Data, Durasi: 6 bulan
```

Struktur Multi-Linked List digunakan dalam program untuk mengelola data dan proyek individu yang bekerja di dalamnya. Setiap pegawai memiliki data seperti nama dan ID pegawai yang disimpan dalam node ElmEmployee. Setiap node pegawai juga memiliki pointer firstProject, yang menunjuk ke daftar proyek yang sedang dikerjakan oleh pegawai tersebut. Proyek ini memiliki informasi seperti nama proyek dan durasi proyek (dalam bulan), yang disimpan dalam node ElmProject. Struktur ini memungkinkan setiap pegawai memiliki daftar proyek mereka sendiri. Dengan menggunakan program, Anda dapat menambah pegawai baru (menambahkan Pegawai), menambah proyek ke daftar pegawai (menambahkan Proyek), menghapus pegawai dengan proyek mereka (hapus Pegawai), dan menampilkan data semua pegawai dan proyek yang mereka kelola (printList). Program ini membantu mengelola informasi proyek secara terstruktur dengan mengatur data hierarkis pekerja dan proyek.

2. LATIHAN 2

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  struct Buku {
6      string judulBuku;
7      string tanggalPengembalian;
8  };
9
10 struct Anggota {
11     string namaAnggota;
12     string idAnggota;
13 };
14
15 struct ElmBuku {
16     Buku info;
17     ElmBuku* next;
18 };
19
20 struct ElmAnggota {
21     Anggota info;
22     ElmBuku* firstBuku;
23     ElmAnggota* next;
24 };
25
26 struct List {
27     ElmAnggota* first;
28 };
29
30 void createList(List &L) {
31     L.first = nullptr;
32 }
33
34 ElmAnggota* allocateAnggota(const Anggota &data) {
35     ElmAnggota* P = new ElmAnggota;
36     P->info = data;
37     P->next = nullptr;
38     P->firstBuku = nullptr;
39     return P;
40 }
41
42 ElmBuku* allocateBuku(const Buku &data) {
43     ElmBuku* P = new ElmBuku;
44     P->info = data;
45     P->next = nullptr;
46     return P;
47 }
48
49 void insertAnggota(List &L, ElmAnggota* P) {
50     if (L.first == nullptr) {
51         L.first = P;
52     } else {
53         P->next = L.first;
54         L.first = P;
55     }
56 }
57
58 void insertBuku(ElmAnggota* anggota, ElmBuku* buku) {
59     if (anggota->firstBuku == nullptr) {
60         anggota->firstBuku = buku;
61     } else {
62         ElmBuku* Q = anggota->firstBuku;
63         while (Q->next != nullptr) {
64             Q = Q->next;
65         }
66         Q->next = buku;
67     }
68 }
69
```

```

70 void removeAnggota(List &L, const string &idAnggota) {
71     ElmAnggota* P = L.first;
72     ElmAnggota* prev = nullptr;
73
74     while (P != nullptr) {
75         if (P->info.idAnggota == idAnggota) {
76             ElmBuku* buku = P->firstBuku;
77             while (buku != nullptr) {
78                 ElmBuku* temp = buku;
79                 buku = buku->next;
80                 delete temp;
81             }
82
83             if (prev == nullptr) {
84                 L.first = P->next;
85             } else {
86                 prev->next = P->next;
87             }
88             delete P;
89             return;
90         }
91         prev = P;
92         P = P->next;
93     }
94 }
95
96 void printList(const List &L) {
97     ElmAnggota* P = L.first;
98     while (P != nullptr) {
99         cout << "Anggota: " << P->info.namaAnggota << " (ID: " << P->info.idAnggota << ")\n";
100        ElmBuku* Q = P->firstBuku;
101        while (Q != nullptr) {
102            cout << "    Buku: " << Q->info.judulBuku << ", Pengembalian: " << Q->info.tanggalPengembalian << "\n";
103            Q = Q->next;
104        }
105        cout << endl;
106        P = P->next;
107    }
108 }
109
110 int main() {
111     List L;
112     createlist(L);
113
114     Anggota a1 = {"Rani", "A001"};
115     Anggota a2 = {"Dito", "A002"};
116     Anggota a3 = {"Vina", "A003"};
117
118     ElmAnggota* anggota1 = allocateAnggota(a1);
119     ElmAnggota* anggota2 = allocateAnggota(a2);
120     ElmAnggota* anggota3 = allocateAnggota(a3);
121
122     insertAnggota(L, anggota1);
123     insertAnggota(L, anggota2);
124     insertAnggota(L, anggota3);
125
126     Buku b1 = {"Pemrograman C++", "01/12/2024"};
127     Buku b2 = {"Algoritma Pemrograman", "15/12/2024"};
128     Buku b3 = {"Struktur Data", "10/12/2024"};
129
130     insertBuku(anggota1, allocateBuku(b1));
131     insertBuku(anggota2, allocateBuku(b2));
132     insertBuku(anggota1, allocateBuku(b3));
133
134     removeAnggota(L, "A002");
135
136     printList(L);
137
138     return 0;
139 }
140

```

OUTPUT

```
Anggota: Vina (ID: A003)
```

```
Anggota: Rani (ID: A001)
```

```
Buku: Pemrograman C++, Pengembalian: 01/12/2024
```

```
Buku: Struktur Data, Pengembalian: 10/12/2024
```

Data anggota perpustakaan dan daftar buku yang mereka pinjam disimpan dalam program melalui struktur Multi-Linked List. Dalam node ElmAnggota, informasi seperti nama anggota dan ID anggota disimpan. Node ini juga memiliki pointer firstBook yang mengarah ke daftar buku yang dipinjam oleh anggota.

Buku mengandung data seperti judul buku dan tanggal pengembalian, yang disimpan dalam node ElmBuku. Dengan struktur ini, setiap anggota memiliki daftar buku sendiri, yang diimplementasikan sebagai daftar terhubung (linked list) yang berbeda. Program dapat menambah anggota (insertAnggota), menambah buku ke daftar anggota (insertBuku), menghapus anggota dan semua buku yang dipinjam (removeAnggota), dan menampilkan data semua anggota dan buku dalam daftar. Program ini memungkinkan manajemen data perpustakaan yang efisien dan hierarkis.