

LAPORAN PRAKTIKUM

PERTEMUAN 13

Multi Linked List



Nama :

Resita Istania Purwanto (2311104037)

Dosen :

Yudha Islami Sulistya

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. GUIDED

a. multilist.h

```
#ifndef MULTILIST_H
#define MULTILIST_H

#include "circularlist.h" // Memanfaatkan struktur circularlist
#include <iostream>
#include <string>

using namespace std;

// Struktur untuk multi list
Codeium: Refactor | Explain
struct Child {
    string nama; // Nama entitas child
    Child* next;
};

Codeium: Refactor | Explain
struct Parent {
    string nama; // Nama entitas parent
    Parent* next;
    Child* firstChild; // Pointer ke anak pertama
};

// Operasi pada parent dan child
Codeium: Refactor | Explain | X
void createParent(Parent* &P, string nama);
void createChild(Child* &C, string nama);
void insertChild(Parent* &P, Child* C);
void printMultilist(Parent* P);

#endif
```

b. multilist.cpp

```
#include "multilist.h"

Codeium: Refactor | Explain | Generate Function Comment | X
void createParent(Parent* &P, string nama) {
    P = new Parent;
    P->nama = nama;
    P->next = nullptr;
    P->firstChild = nullptr;
}

Codeium: Refactor | Explain | Generate Function Comment | X
void createChild(Child* &C, string nama) {
    C = new Child;
    C->nama = nama;
    C->next = nullptr;
}

Codeium: Refactor | Explain | Generate Function Comment | X
void insertChild(Parent* &P, Child* C) {
    if (P->firstChild == nullptr) {
        P->firstChild = C;
    } else {
        Child* temp = P->firstChild;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = C;
    }
}

Codeium: Refactor | Explain | Generate Function Comment | X
void printMultilist(Parent* P) {
    while (P != nullptr) {
        cout << "Parent: " << P->nama << endl;
    }
}
```

```

32     Child* C = P->firstChild;
33     while (C != nullptr) {
34         cout << "   Child: " << C->nama << endl;
35         C = C->next;
36     }
37
38     P = P->next;
39 }
40 }
41
42

```

c. circularlist.h

```

#ifndef CIRCULARLIST_H
#define CIRCULARLIST_H

#include <string>
using namespace std;

Codeium: Refactor | Explain
struct mahasiswa {
    string nama;
    string nim;
    char jenis_kelamin;
    float ipk;
};

typedef mahasiswa infotype;

Codeium: Refactor | Explain
struct ElmList {
    infotype info;
    ElmList *next;
};

typedef ElmList* address;
typedef struct {
    address first;
} List;

// Fungsi dan prosedur untuk Circular List
Codeium: Refactor | Explain | X
void createlist(List &L);
address alokasi(infotype x);
void dealokasi(address &P);
void insertFirst(List &L, address P);
void insertAfter(List &L, address Prec, address P);

32 void insertLast(List &L, address P);
33 void deleteFirst(List &L, address &P);
34 void deleteAfter(List &L, address Prec, address &P);
35 void deletelast(List &L, address &P);
36 address findElm(List L, infotype x);
37 void printInfo(List L);
38 address createData(string nama, string nim, char jenis_kelamin, float ipk);
39
40 #endif // CIRCULARLIST_H

```

d. circularlist.cpp

```

1  #include <iostream>
2  #include "circularlist.h"
3
4  using namespace std;
5
6  Codeium: Refactor | Explain | Generate Function Comment | ✕
7  void createlist(List &L) {
8      L.first = nullptr;
9  }
10
11  Codeium: Refactor | Explain | Generate Function Comment | ✕
12  address alokasi(infotype x) {
13      address P = new Elmlist;
14      if (P != nullptr) {
15          P->info = x;
16          P->next = nullptr;
17      }
18      return P;
19  }
20
21  Codeium: Refactor | Explain | Generate Function Comment | ✕
22  void dealokasi(address &P) {
23      delete P;
24  }
25
26  Codeium: Refactor | Explain | Generate Function Comment | ✕
27  void insertFirst(List &L, address P) {
28      if (L.first == nullptr) {
29          L.first = P;
30          P->next = P;
31      } else {
32          address last = L.first;
33          while (last->next != L.first) {
34              last = last->next;
35          }
36          last->next = P;
37          P->next = L.first;
38          L.first = P;
39      }
40  }
41
42  Codeium: Refactor | Explain | Generate Function Comment | ✕
43  void insertAfter(List &L, address Prec, address P) {
44      P->next = Prec->next;
45      Prec->next = P;
46  }
47
48  Codeium: Refactor | Explain | Generate Function Comment | ✕
49  void insertLast(List &L, address P) {
50      if (L.first == nullptr) {
51          L.first = P;
52          P->next = P;
53      } else {
54          address last = L.first;
55          while (last->next != L.first) {
56              last = last->next;
57          }
58          last->next = P;
59          P->next = L.first;
60      }
61  }
62
63  Codeium: Refactor | Explain | Generate Function Comment | ✕
64  void deleteFirst(List &L, address &P) {
65      P = L.first;

```

```

66  }
67
68  Codeium: Refactor | Explain | Generate Function Comment | ✕
69  void deleteLast(List &L, address &P) {
70      if (L.first == nullptr) {
71          P = nullptr;
72      } else {
73          address last = L.first;
74          while (last->next != L.first) {
75              last = last->next;
76          }
77          last->next = nullptr;
78          P = last;
79      }
80  }
81
82  Codeium: Refactor | Explain | Generate Function Comment | ✕
83  void deleteAll(List &L) {
84      L.first = nullptr;
85  }
86
87  Codeium: Refactor | Explain | Generate Function Comment | ✕
88  void displayCircularList(List &L) {
89      if (L.first == nullptr) {
90          cout << "Circular List is empty." << endl;
91      } else {
92          address P = L.first;
93          do {
94              cout << P->info << " ";
95              P = P->next;
96          } while (P != L.first);
97          cout << endl;
98      }
99  }
100
101  Codeium: Refactor | Explain | Generate Function Comment | ✕
102  int main() {
103      List L;
104      createlist(L);
105      alokasi(10);
106      alokasi(20);
107      alokasi(30);
108      insertFirst(L, alokasi(40));
109      insertAfter(L, L.first, alokasi(50));
110      insertLast(L, alokasi(60));
111      displayCircularList(L);
112      deleteFirst(L, L.first);
113      deleteLast(L, L.first);
114      deleteAll(L);
115      displayCircularList(L);
116      return 0;
117  }

```

```

59     if (P != nullptr) {
60         if (P->next == L.first) {
61             L.first = nullptr;
62         } else {
63             address last = L.first;
64             while (last->next != L.first) {
65                 last = last->next;
66             }
67             L.first = P->next;
68             last->next = L.first;
69         }
70         P->next = nullptr;
71     }
72 }
73

```

Codeium: Refactor | Explain | Generate Function Comment | X

```

74 void deleteAfter(List &L, address Prec, address &P) {
75     P = Prec->next;
76     if (P != nullptr) {
77         Prec->next = P->next;
78         P->next = nullptr;
79     }
80 }
81

```

Codeium: Refactor | Explain | Generate Function Comment | X

```

82 void deleteLast(List &L, address &P) {
83     if (L.first != nullptr) {
84         address last = L.first;
85         while (last->next->next != L.first) {
86             last = last->next;
87         }
88         P = last->next;
89         last->next = L.first;
90

```

```

90         P->next = nullptr;
91     }
92 }
93

```

Codeium: Refactor | Explain | Generate Function Comment | X

```

94 address findElm(List L, infotype x) {
95     address P = L.first;
96     if (P != nullptr) {
97         do {
98             if (P->info.nim == x.nim) {
99                 return P;
100             }
101             P = P->next;
102         } while (P != L.first);
103     }
104     return nullptr;
105 }
106

```

Codeium: Refactor | Explain | Generate Function Comment | X

```

107 void printInfo(List L) {
108     address P = L.first;
109     if (P != nullptr) {
110         do {
111             cout << "Nama: " << P->info.nama << endl;
112             cout << "NIM : " << P->info.nim << endl;
113             cout << "L/P : " << P->info.jenis_kelamin << endl;
114             cout << "IPK : " << P->info.ipk << endl << endl;
115             P = P->next;
116         } while (P != L.first);
117     }
118 }

```

```

Codeium: Refactor | Explain | Generate Function Comment | X
120 address createData(string nama, string nim, char jenis_kelamin, float ipk) {
121     infotype x = {nama, nim, jenis_kelamin, ipk};
122     return alokasi(x);
123 }
124

```

e. main.cpp

```

#include <iostream>
#include "circularlist.h"

using namespace std;

Codeium: Refactor | Explain | Generate Function Comment | X
int main() {
    List L;
    address P1, parag2;
    infotype x;

    createlist(L);

    cout << "coba insert first, last, dan after" << endl;

    P1 = createData("Danu", "04", 'l', 4.0);
    insertFirst(L, P1);

    P1 = createData("Fahmi", "06", 'l', 3.45);
    insertLast(L, P1);

    P1 = createData("Bobi", "02", 'l', 3.71);
    insertFirst(L, P1);

    P1 = createData("Ali", "01", 'l', 3.3);
    insertFirst(L, P1);

    P1 = createData("Gita", "07", 'p', 3.75);
    insertLast(L, P1);

    x.nim = "07";
    P1 = findElm(L, x);
    P2 = createData("Cindi", "03", 'p', 3.5);

    insertAfter(L, P1, P2);

    x.nim = "02";
    P1 = findElm(L, x);
    P2 = createData("Hilmi", "08", 'p', 3.3);
    insertAfter(L, P1, P2);

    x.nim = "04";
    P1 = findElm(L, x);
    P2 = createData("Eli", "05", 'p', 3.4);
    insertAfter(L, P1, P2);

    printInfo(L);

    return 0;
}

```

Output:

```
coba insert first, last, dan after
```

```
Nana: Ali
```

```
NIM : 01
```

```
L/P : 1
```

```
IPK : 3.3
```

```
Nana: Bobi
```

```
NIM : 02
```

```
L/P : 1
```

```
IPK : 3.71
```

```
Nana: Hilmi
```

```
NIM : 08
```

```
L/P : p
```

```
IPK : 3.3
```

```
Nana: Danu
```

```
NIM : 04
```

```
L/P : 1
```

```
IPK : 4
```

```
Nana: Eli
```

```
NIM : 05
```

```
L/P : p
```

```
IPK : 3.4
```

```
Nana: Fahmi
```

```
NIM : 06
```

```
L/P : 1
```

```
IPK : 3.45
```

```
Nana: Gita
```

```
NIM : 07
```

```
L/P : p
```

```
IPK : 3.75
```

```
Nana: Cindi
```

```
NIM : 03
```

```
L/P : p
```

```
IPK : 3.5
```

Penjelasan:

- Multilist: Struktur data yang menghubungkan entitas induk dan anak. Tersedia kemampuan untuk membuat orang tua dan anak, serta menambahkan anak ke orang tua.
- Circular list: Mencantumkan struktur data yang elemennya terhubung secara melingkar. Berisi fungsionalitas untuk menambah, menghapus, dan mencari item dalam daftar.
- Dalam fungsi main(), data siswa dimasukkan ke dalam daftar melingkar menggunakan berbagai operasi (insertFirst, insertLast, insertAfter) dan informasi siswa ditampilkan menggunakan printInfo().

II. UNGUIDED

soal 1

a. pegawai.h


```

#ifndef PEGAWAI_H
#define PEGAWAI_H

#include <string>
using namespace std;

// Struktur untuk Proyek
Codeium: Refactor | Explain
struct Proyek {
    string namaProyek;
    int durasi;
    Proyek* nextProyek;
};

// Struktur untuk Pegawai
Codeium: Refactor | Explain
struct Pegawai {
    string namaPegawai;
    string idPegawai;
    Proyek* proyek;
    Pegawai* nextPegawai;
};

// Fungsi untuk membuat pegawai baru
Codeium: Refactor | Explain | X
Pegawai* buatPegawai(string nama, string id);

// Fungsi untuk menambahkan proyek ke pegawai
Codeium: Refactor | Explain | X
void tambahProyek(Pegawai* pegawai, string namaProyek, int durasi);

// Fungsi untuk menghapus proyek dari pegawai
Codeium: Refactor | Explain | X
void hapusProyek(Pegawai* pegawai, string namaProyek);

28 // Fungsi untuk menghapus proyek dari pegawai
Codeium: Refactor | Explain | X
29 void hapusProyek(Pegawai* pegawai, string namaProyek);
30
31 // Fungsi untuk menampilkan data pegawai dan proyek mereka
Codeium: Refactor | Explain | X
32 void tampilkanPegawai(Pegawai* head);
33
34 #endif

```

b. pegawai.cpp


```

1  #include "Pegawai.h"
2  #include <iostream>
3  using namespace std;
4
5  Codeium: Refactor | Explain | Generate Function Comment | X
6  Pegawai* buatPegawai(string nama, string id) {
7      Pegawai* pegawai = new Pegawai;
8      pegawai->namaPegawai = nama;
9      pegawai->idPegawai = id;
10     pegawai->proyek = nullptr;
11     pegawai->nextPegawai = nullptr;
12     return pegawai;
13 }
14
15 Codeium: Refactor | Explain | Generate Function Comment | X
16 void tambahProyek(Pegawai* pegawai, string namaProyek, int durasi) {
17     Proyek* proyek = new Proyek;
18     proyek->namaProyek = namaProyek;
19     proyek->durasi = durasi;
20     proyek->nextProyek = pegawai->proyek;
21     pegawai->proyek = proyek;
22 }
23
24 Codeium: Refactor | Explain | Generate Function Comment | X
25 void hapusProyek(Pegawai* pegawai, string namaProyek) {
26     Proyek* proyek = pegawai->proyek;
27     Proyek* prev = nullptr;
28
29     while (proyek != nullptr && proyek->namaProyek != namaProyek) {
30         prev = proyek;
31         proyek = proyek->nextProyek;
32     }
33
34     if (proyek != nullptr) {
35         if (prev == nullptr) {
36             pegawai->proyek = proyek->nextProyek;
37         } else {
38             prev->nextProyek = proyek->nextProyek;
39         }
40         delete proyek;
41     }
42 }
43
44 Codeium: Refactor | Explain | Generate Function Comment | X
45 void tampilkanPegawai(Pegawai* head) {
46     Pegawai* pegawai = head;
47     while (pegawai != nullptr) {
48         cout << "Nama Pegawai: " << pegawai->namaPegawai << ", ID: " << pegawai->idPegawai << endl;
49         Proyek* proyek = pegawai->proyek;
50         while (proyek != nullptr) {
51             cout << " - Proyek: " << proyek->namaProyek << ", Durasi: " << proyek->durasi << " bulan" << endl;
52             proyek = proyek->nextProyek;
53         }
54         pegawai = pegawai->nextPegawai;
55     }
56 }
57

```

c. Main.cpp

```

1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 Codeium: Refactor | Explain
6 struct Proyek {
7     string namaProyek;
8     int durasi;
9     Proyek* nextProyek;
10 };
11
12 Codeium: Refactor | Explain
13 struct Pegawai {
14     string namaPegawai;
15     string idPegawai;
16     Proyek* proyek;
17     Pegawai* nextPegawai;
18 };
19
20 Codeium: Refactor | Explain | Generate Function Comment | X
21 Pegawai* buatPegawai(string nama, string id) {
22     Pegawai* pegawai = new Pegawai;
23     pegawai->namaPegawai = nama;
24     pegawai->idPegawai = id;
25     pegawai->proyek = nullptr;
26     pegawai->nextPegawai = nullptr;
27     return pegawai;
28 }
29
30 Codeium: Refactor | Explain | Generate Function Comment | X
31 void tambahProyek(Pegawai* pegawai, string namaProyek, int durasi) {
32     Proyek* proyek = new Proyek;
33     proyek->namaProyek = namaProyek;
34     proyek->durasi = durasi;

```

```

31     proyek->nextProyek = pegawai->proyek;
32     pegawai->proyek = proyek;
33 }
34
35 Codeium: Refactor | Explain | Generate Function Comment | X
36 void hapusProyek(Pegawai* pegawai, string namaProyek) {
37     Proyek* proyek = pegawai->proyek;
38     Proyek* prev = nullptr;
39
40     while (proyek != nullptr && proyek->namaProyek != namaProyek) {
41         prev = proyek;
42         proyek = proyek->nextProyek;
43     }
44
45     if (proyek != nullptr) {
46         if (prev == nullptr) {
47             pegawai->proyek = proyek->nextProyek;
48         } else {
49             prev->nextProyek = proyek->nextProyek;
50         }
51         delete proyek;
52     }
53 }
54
55 Codeium: Refactor | Explain | Generate Function Comment | X
56 void tampilkanPegawai(Pegawai* head) {
57     Pegawai* pegawai = head;
58     while (pegawai != nullptr) {
59         cout << "Nama Pegawai: " << pegawai->namaPegawai << ", ID: " << pegawai->idPegawai << endl;
60         Proyek* proyek = pegawai->proyek;
61         while (proyek != nullptr) {
62             cout << " - Proyek: " << proyek->namaProyek << ", Durasi: " << proyek->durasi << " bulan" << endl;
63             proyek = proyek->nextProyek;

```

```

62     }
63     pegawai = pegawai->nextPegawai;
64 }
65 }
66
Codeium: Refactor | Explain | Generate Function Comment | X
67 int main() {
68     Pegawai* head = nullptr;
69     Pegawai* andi = buatPegawai("Andi", "P001");
70     Pegawai* budi = buatPegawai("Budi", "P002");
71     Pegawai* citra = buatPegawai("Citra", "P003");
72
73     head = andi;
74     andi->nextPegawai = budi;
75     budi->nextPegawai = citra;
76
77
78     tambahProyek(andi, "Aplikasi Mobile", 12);
79     tambahProyek(budi, "Sistem Akuntansi", 8);
80     tambahProyek(citra, "E-commerce", 10);
81
82
83     tambahProyek(andi, "Analisis Data", 6);
84
85
86     hapusProyek(andi, "Aplikasi Mobile");
87
88     tampilkanPegawai(head);
89
90     return 0;
91 }

```

Output:

```

Nama Pegawai: Andi, ID: P001
- Proyek: Analisis Data, Durasi: 6 bulan
Nama Pegawai: Budi, ID: P002
- Proyek: Sistem Akuntansi, Durasi: 8 bulan
Nama Pegawai: Citra, ID: P003
- Proyek: E-commerce, Durasi: 10 bulan

```

Penjelasan:

- Pegawai.h (header untuk struktur data pegawai dan proyek)
- Pegawai.cpp (implementasi fungsi-fungsi terkait pegawai dan proyek)
- Main.cpp (file utama untuk menjalankan program)

Soal 2

a. Anggota.h

```

1  #ifndef ANGOTA_H
2  #define ANGOTA_H
3
4  #include <string>
5  using namespace std;
6
Codeium: Refactor | Explain
7  struct Buku {
8      string judulBuku;
9      string tanggalPengembalian;
10     Buku* nextBuku;
11 };
12
Codeium: Refactor | Explain
13 struct Anggota {
14     string namaAnggota;
15     string idAnggota;
16     Buku* buku;
17     Anggota* nextAnggota;
18 };
19
20 Anggota* buatAnggota(string nama, string id);
21
22 void tambahBuku(Anggota* anggota, string judulBuku, string tanggalPengembalian);
23
24 void hapusAnggota(Anggota*& head, string idAnggota);
25
26 void tampilkanAnggota(Anggota* head);
27
28 #endif

```

b. Anggota.cpp

```

1  #include "Anggota.h"
2  #include <iostream>
3  using namespace std;
4  Anggota* buatAnggota(string nama, string id) {
5      Anggota* anggota = new Anggota;
6      anggota->namaAnggota = nama;
7      anggota->idAnggota = id;
8      anggota->buku = nullptr;
9      anggota->nextAnggota = nullptr;
10     return anggota;
11 }
12
13 void tambahBuku(Anggota* anggota, string judulBuku, string tanggalPengembalian) {
14     Buku* buku = new Buku;
15     buku->judulBuku = judulBuku;
16     buku->tanggalPengembalian = tanggalPengembalian;
17     buku->nextBuku = anggota->buku;
18     anggota->buku = buku;
19 }
20
21 void hapusAnggota(Anggota*& head, string idAnggota) {
22     Anggota* anggota = head;
23     Anggota* prev = nullptr;
24
25     while (anggota != nullptr && anggota->idAnggota != idAnggota) {
26         prev = anggota;
27         anggota = anggota->nextAnggota;
28     }
29
30     if (anggota != nullptr) {
31         if (prev == nullptr) {
32             head = anggota->nextAnggota;
33         } else {
34             prev->nextAnggota = anggota->nextAnggota;
35         }
36
37         Buku* buku = anggota->buku;
38         while (buku != nullptr) {
39             Buku* temp = buku;
40             buku = buku->nextBuku;
41             delete temp;
42         }
43
44         delete anggota;
45     }
46 }
47 void tampilkanAnggota(Anggota* head) {
48     Anggota* anggota = head;
49     while (anggota != nullptr) {
50         cout << "Nama Anggota: " << anggota->namaAnggota << ", ID: " << anggota->idAnggota << endl;
51         Buku* buku = anggota->buku;
52         while (buku != nullptr) {
53             cout << " - Buku: " << buku->judulBuku << ", Tanggal Pengembalian: " << buku->tanggalPengembalian << endl;
54             buku = buku->nextBuku;
55         }
56         anggota = anggota->nextAnggota;
57     }
58 }
59

```

c. Main.cpp

```

1  #include <iostream>
2  #include "Anggota.h"
3  using namespace std;
4
5  Codeium: Refactor | Explain | Generate Function Comment | X
6  int main() {
7      // Membuat anggota
8      Anggota* head = nullptr;
9      Anggota* rani = buatAnggota("Rani", "A001");
10     Anggota* dito = buatAnggota("Dito", "A002");
11     Anggota* vina = buatAnggota("Vina", "A003");
12
13     head = rani;
14     rani->nextAnggota = dito;
15     dito->nextAnggota = vina;
16
17     tambahBuku(rani, "Pemrograman C++", "01/12/2024");
18     tambahBuku(dito, "Algoritma Pemrograman", "15/12/2024");
19
20     tambahBuku(rani, "Struktur Data", "10/12/2024");
21
22     hapusAnggota(head, "A002");
23
24     tampilkanAnggota(head);
25
26     return 0;
27 }

```

Output:

```

Nama Anggota: Rani, ID: A001
- Buku: Struktur Data, Tanggal Pengembalian: 10/12/2024
- Buku: Pemrograman C++, Tanggal Pengembalian: 01/12/2024
Nama Anggota: Vina, ID: A003

```

Penjelasan:

Anggota.h (header untuk struktur data anggota dan buku)

Anggota.cpp (implementasi fungsi-fungsi terkait anggota dan buku)

Main.cpp (file utama untuk menjalankan program)