LAPORAN PRAKTIKUM

BAB (13)

Disusun Guna Memenuhi Tugas Mata Kuliah Struktur Data
Dosen Pengampu: Asep Jamaludin, S.Si., M.Kom.

Disusun Oleh:

(Dzulfikar Fathin A)

(2410631170135)

KELAS (2 C)

INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS SINGAPERBANGSA KARAWANG

2025

SOAL PRAKTIKUM

Latihan Pertemuan 13

Buat sistem dimana mencakup hal yang dapat mencatat data buku dan anggota, proses peminjaman buku, pengembalian buku, pencarian buku, serta pemetaan lokasi buku berdasarkan rak. Metode yang digunakan:

- Struct
- Pointer
- Stack
- Queue
- Tree
- Array
- Graph (adjacency matrix).

Lalu buatlah hasil pengerjaan dalam bentuk laporan dan di jelaskan hasil pengerjaan tersebut. Diharapkan Menjelaskan Hasil Source Code Yang Sudah di Buat Tidak Hanya Menampilkan Source Code di Laporan.

LANGKAH PENYELESAIAN

main.cpp

```cpp
#include <iostream>
#include <string>
#include <stack>
#include <queue>
#include <map>
#include <vector>
#include <iomanip>
using namespace std;

const int MAX_BUKU = 100;
const int MAX_ANGGOTA = 100;
const int MAX_RAK = 100;

struct Buku {
    int id;
    string judul;
    string lokasiRak;
};

struct Anggota {
    int id;
    string nama;
};

struct Peminjaman {
    int idBuku;
    int idAnggota;
};

struct TreeNode {
    Buku data;
    TreeNode* left;
    TreeNode* right;
};

TreeNode* root = nullptr;

Buku daftarBuku[MAX_BUKU];
Anggota daftarAnggota[MAX_ANGGOTA];
Peminjaman daftarPinjam[MAX_BUKU];
int jumlahBuku = 0, jumlahAnggota = 0, jumlahPinjam = 0;
```

main.cpp

```cpp
stack<Peminjaman> riwayatPengembalian;
queue<Peminjaman> antreanPeminjam;

// === GRAPH DENGAN ADJACENCY MATRIX ===
map<string, int> rakToIndex;
string indexToRak[MAX_RAK];
int adjacencyMatrix[MAX_RAK][MAX_RAK] = {0};
int jumlahRak = 0;

void tambahRakJikaBaru(string rak) {
    if (rakToIndex.find(rak) == rakToIndex.end()) {
        rakToIndex[rak] = jumlahRak;
        indexToRak[jumlahRak] = rak;
        jumlahRak++;
    }
}
```

```
void updateGraphRak(string rak) {
    tambahRakJikaBaru(rak);
    int idx = rakToIndex[rak];
    adjacencyMatrix[idx][idx] = 1; // Dummy: koneksi ke diri sendiri
}

void tambahBuku() {
    Buku b;
    cout << "Masukkan ID Buku: ";
    cin >> b.id;
    cin.ignore();
    cout << "Masukkan Judul Buku: ";
    getline(cin, b.judul);
    cout << "Masukkan Lokasi Rak: ";
    cin >> b.lokasiRak;

    daftarBuku[jumlahBuku++] = b;

    TreeNode* newNode = new TreeNode{b, nullptr, nullptr};

    if (!root) root = newNode;
    else {
        TreeNode* cur = root;
        while (true) {
            if (b.judul < cur->data.judul) {
                if (cur->left) cur = cur->left;
                else {
                    cur->left = newNode;
                    break;
                }
            } else {
                if (cur->right) cur = cur->right;
                else {
                    cur->right = newNode;
                    break;
                }
            }
        }
    }
```

```
void tambahBuku() {

    updateGraphRak(b.lokasiRak);
    cout << "Buku berhasil ditambahkan!\n";
}

void tambahAnggota() {
    Anggota a;
    cout << "Masukkan ID Anggota: ";
    cin >> a.id;
    cin.ignore();
    cout << "Masukkan Nama Anggota: ";
    getline(cin, a.nama);
    daftarAnggota[jumlahAnggota++] = a;
    cout << "Anggota berhasil ditambahkan!\n";
}

void pinjamBuku() {
    int idBuku, idAnggota;
    cout << "Masukkan ID Buku: ";
    cin >> idBuku;
    cout << "Masukkan ID Anggota: ";
    cin >> idAnggota;

    Peminjaman p{idBuku, idAnggota};
    daftarPinjam[jumlahPinjam++] = p;
    antreanPeminjam.push(p);

    cout << "Peminjaman dicatat.\n";
}
```

```
void kembalikanBuku() {
    if (antreanPeminjam.empty()) {
        cout << "Tidak ada buku yang sedang dipinjam.\n";
        return;
    }

    Peminjaman p = antreanPeminjam.front();
    antreanPeminjam.pop();
    riwayatPengembalian.push(p);

    cout << "Buku berhasil dikembalikan!\n";
}

TreeNode* cariBuku(TreeNode* node, const string& judul) {
    if (!node) return nullptr;
    if (node->data.judul == judul) return node;
    if (judul < node->data.judul) return cariBuku(node->left, judul);
    else return cariBuku(node->right, judul);
}

void cariDanTampilkanBuku() {
    string judul;
    cout << "Masukkan judul buku yang dicari: ";
    cin.ignore();
    getline(cin, judul);

    TreeNode* hasil = cariBuku(root, judul);
    if (hasil) {
        cout << "Buku ditemukan:\n";
        cout << "ID: " << hasil->data.id << ", Judul: " << hasil->data.judul
             << ", Lokasi Rak: " << hasil->data.lokasiRak << endl;
    } else {
        cout << "Buku tidak ditemukan.\n";
    }
}
```

```
void tampilkanGraphRak() {
    cout << "\nPeta Lokasi Rak (Adjacency Matrix):\n\n";

    const int lebarKolom = 10; // lebar per kolom agar rata

    // Header kolom
    cout << setw(lebarKolom) << " ";
    for (int i = 0; i < jumlahRak; ++i) {
        cout << setw(lebarKolom) << indexToRak[i];
    }
    cout << "\n";

    // Garis bawah header
    cout << string(lebarKolom * (jumlahRak + 1), '-') << "\n";

    // Isi matrix
    for (int i = 0; i < jumlahRak; ++i) {
        cout << setw(lebarKolom) << indexToRak[i];
        for (int j = 0; j < jumlahRak; ++j) {
            cout << setw(lebarKolom) << adjacencyMatrix[i][j];
        }
        cout << "\n";
    }
}
```

```
void tampilkanSemuaBukuBerdasarkanRak() {
    if (jumlahBuku == 0) {
        cout << "Belum ada buku yang tersedia.\n";
        return;
    }

    map<string, vector<Buku>> bukuPerRak;
    for (int i = 0; i < jumlahBuku; ++i) {
        bukuPerRak[daftarBuku[i].lokasiRak].push_back(daftarBuku[i]);
    }

    cout << "Daftar Buku Berdasarkan Rak:\n";
    for (const auto& rak : bukuPerRak) {
        cout << "Rak " << rak.first << ":\n";
        for (const auto& buku : rak.second) {
            cout << "  - ID: " << buku.id << ", Judul: " << buku.judul << endl;
        }
    }
}

void tampilkanMenu() {
    cout << "\n===== SISTEM PERPUSTAKAAN =====\n";
    cout << "1. Tambah Buku\n";
    cout << "2. Tambah Anggota\n";
    cout << "3. Pinjam Buku\n";
    cout << "4. Kembalikan Buku\n";
    cout << "5. Cari Buku\n";
    cout << "6. Tampilkan Peta Rak (Graph)\n";
    cout << "7. Tampilkan Semua Buku Berdasarkan Rak\n";
    cout << "8. Keluar\n";
    cout << "Pilih menu: ";
}
```

```
int main() {
    int pilihan;
    do {
        tampilkanMenu();
        cin >> pilihan;
        switch (pilihan) {
            case 1: tambahBuku(); break;
            case 2: tambahAnggota(); break;
            case 3: pinjamBuku(); break;
            case 4: kembalikanBuku(); break;
            case 5: cariDanTampilkanBuku(); break;
            case 6: tampilkanGraphRak(); break;
            case 7: tampilkanSemuaBukuBerdasarkanRak(); break;
            case 8: cout << "Terima kasih!\n"; break;
            default: cout << "Pilihan tidak valid.\n";
        }
    } while (pilihan != 8);

    return 0;
}
```

Penjelasan Code

1. Library dan Namespace

```cpp
#include <iostream>
#include <string>
#include <stack>
#include <queue>
#include <map>
#include <vector>
#include <iomanip>
using namespace std;
```

- #include digunakan untuk menyertakan pustaka standar:
  - iostream: input/output (cin, cout)
  - string: manipulasi teks
  - stack: struktur data tumpukan
  - queue: struktur data antrian
  - map: struktur data key-value (asosiatif)
  - vector: array dinamis
  - iomanip: manipulasi output (misalnya, setw)
- using namespace std: agar tidak perlu menulis std:: sebelum objek dari pustaka standar.

2. Konstanta dan Struktur Data

```cpp
const int MAX_BUKU = 100;
const int MAX_ANGGOTA = 100;
const int MAX_RAK = 100;

struct Buku {
    int id;
    string judul;
    string lokasiRak;
};

struct Anggota {
    int id;
    string nama;
};

struct Peminjaman {
    int idBuku;
    int idAnggota;
};

struct TreeNode {
    Buku data;
    TreeNode* left;
    TreeNode* right;
};
```

- Mendefinisikan batas maksimal data buku, anggota, dan rak.
- Struktur Buku menyimpan ID, judul, dan lokasi rak.
- Struktur Anggota menyimpan ID dan nama anggota.
- Struktur Peminjaman menyimpan hubungan antara buku dan anggota.
- Struktur untuk **Binary Search Tree** berdasarkan judul buku.

3. Variabel Global

```
TreeNode* root = nullptr;

Buku daftarBuku[MAX_BUKU];
Anggota daftarAnggota[MAX_ANGGOTA];
Peminjaman daftarPinjam[MAX_BUKU];
int jumlahBuku = 0, jumlahAnggota = 0, jumlahPinjam = 0;

stack<Peminjaman> riwayatPengembalian;
queue<Peminjaman> antreanPeminjam;

```

- Akar dari BST yang menyimpan buku.
- Array dan penghitung data untuk buku, anggota, dan peminjaman.
- queue: menyimpan antrean peminjam.
- stack: menyimpan riwayat pengembalian.

4. Graph Adjacency Matrix

```

map<string, int> rakToIndex;
string indexToRak[MAX_RAK];
int adjacencyMatrix[MAX_RAK][MAX_RAK] = {0};
int jumlahRak = 0;

```

- Untuk menyimpan relasi antar rak (meskipun hanya self-loop digunakan dalam kode ini).

5. Fungsi Graph Rak

```
void tambahRakJikaBaru(string rak) {
    if (rakToIndex.find(rak) == rakToIndex.end()) {
        rakToIndex[rak] = jumlahRak;
        indexToRak[jumlahRak] = rak;
        jumlahRak++;
    }
}
```

- Menambahkan entri rak baru jika belum ada dalam rakToIndex.
- Menambahkan node ke graf dan membuat koneksi ke dirinya sendiri.

6. Fungsi Tambah Buku

```
66  void tambahBuku() {
67      Buku b;
68      cout << "Masukkan ID Buku: ";
69      cin >> b.id;
70      cin.ignore();
71      cout << "Masukkan Judul Buku: ";
72      getline(cin, b.judul);
73      cout << "Masukkan Lokasi Rak: ";
74      cin >> b.lokasiRak;
75
76      daftarBuku[jumlahBuku++] = b;
77
78      TreeNode* newNode = new TreeNode{b, nullptr, nullptr};
79
80      if (!root) root = newNode;
81      else {
82          TreeNode* cur = root;
83          while (true) {
84              if (b.judul < cur->data.judul) {
85                  if (cur->left) cur = cur->left;
86                  else {
87                      cur->left = newNode;
88                      break;
89                  }
90              } else {
91                  if (cur->right) cur = cur->right;
92                  else {
93                      cur->right = newNode;
94                      break;
95                  }
96              }
97          }
98      }
99
100     updateGraphRak(b.lokasiRak);
101     cout << "Buku berhasil ditambahkan!\n";
102 }
```

- Input data buku
- Simpan ke array
- Masukkan ke BST berdasarkan judul
- Tambahkan rak ke graf

7. Fungsi Tambah Anggota

```
104 void tambahAnggota() {
105     Anggota a;
106     cout << "Masukkan ID Anggota: ";
107     cin >> a.id;
108     cin.ignore();
109     cout << "Masukkan Nama Anggota: ";
110     getline(cin, a.nama);
111     daftarAnggota[jumlahAnggota++] = a;
112     cout << "Anggota berhasil ditambahkan!\n";
113 }
```

- Input data anggota, simpan ke array.

8. Fungsi Pinjam Buku

```
115 void pinjamBuku() {
116     int idBuku, idAnggota;
117     cout << "Masukkan ID Buku: ";
118     cin >> idBuku;
119     cout << "Masukkan ID Anggota: ";
120     cin >> idAnggota;
121
122     Peminjaman p{idBuku, idAnggota};
123     daftarPinjam[jumlahPinjam++] = p;
124     antreanPeminjam.push(p);
125
126     cout << "Peminjaman dicatat.\n";
127 }
```

- Input ID buku dan anggota
- Simpan ke array dan tambahkan ke antrean peminjam

9. Fungsi kembalikan Buku

```
void kembalikanBuku() {
    if (antreanPeminjam.empty()) {
        cout << "Tidak ada buku yang sedang dipinjam.\n";
        return;
    }

    Peminjaman p = antreanPeminjam.front();
    antreanPeminjam.pop();
    riwayatPengembalian.push(p);

    cout << "Buku berhasil dikembalikan!\n";
}
```

- Ambil data dari antrean depan
- Masukkan ke stack riwayat pengembalian

10. Pencarian Buku (BST)

```
TreeNode* cariBuku(TreeNode* node, const string& judul) {
    if (!node) return nullptr;
    if (node->data.judul == judul) return node;
    if (judul < node->data.judul) return cariBuku(node->left, judul);
    else return cariBuku(node->right, judul);
}
```

- Fungsi rekursif mencari buku berdasarkan judul di BST

11. Fungsi Cari dan tampilkan Buku

```
void cariDanTampilkanBuku() {
    string judul;
    cout << "Masukkan judul buku yang dicari: ";
    cin.ignore();
    getline(cin, judul);

    TreeNode* hasil = cariBuku(root, judul);
    if (hasil) {
        cout << "Buku ditemukan:\n";
        cout << "ID: " << hasil->data.id << ", Judul: " << hasil->data.judul
             << ", Lokasi Rak: " << hasil->data.lokasiRak << endl;
    } else {
        cout << "Buku tidak ditemukan.\n";
    }
}
```

- Input judul, tampilkan hasil pencarian

12. Menampilkan Peta Lokasi Rak (Graph)

```
void tampilkanGraphRak() {
    cout << "\nPeta Lokasi Rak (Adjacency Matrix):\n\n";

    const int lebarKolom = 10;

    cout << setw(lebarKolom) << " ";
    for (int i = 0; i < jumlahRak; ++i) {
        cout << setw(lebarKolom) << indexToRak[i];
    }
    cout << "\n";

    cout << string(lebarKolom * (jumlahRak + 1), '-') << "\n";

    for (int i = 0; i < jumlahRak; ++i) {
        cout << setw(lebarKolom) << indexToRak[i];
        for (int j = 0; j < jumlahRak; ++j) {
            cout << setw(lebarKolom) << adjacencyMatrix[i][j];
        }
        cout << "\n";
    }
}
```

- Tampilkan adjacency matrix sebagai peta lokasi rak

13. Menampilkan Buku per Rak

```
void tampilkanSemuaBukuBerdasarkanRak() {
    if (jumlahBuku == 0) {
        cout << "Belum ada buku yang tersedia.\n";
        return;
    }

    map<string, vector<Buku>> bukuPerRak;
    for (int i = 0; i < jumlahBuku; ++i) {
        bukuPerRak[daftarBuku[i].lokasiRak].push_back(daftarBuku[i]);
    }

    cout << "Daftar Buku Berdasarkan Rak:\n";
    for (const auto& rak : bukuPerRak) {
        cout << "Rak " << rak.first << ":\n";
        for (const auto& buku : rak.second) {
            cout << "  - ID: " << buku.id << ", Judul: " << buku.judul << endl;
        }
    }
}
```

- Mengelompokkan buku berdasarkan rak dan menampilkannya

14. Menu Utama

```
void tampilkanMenu() {
    cout << "\n===== SISTEM PERPUSTAKAAN =====\n";
    cout << "1. Tambah Buku\n";
    cout << "2. Tambah Anggota\n";
    cout << "3. Pinjam Buku\n";
    cout << "4. Kembalikan Buku\n";
    cout << "5. Cari Buku\n";
    cout << "6. Tampilkan Peta Rak (Graph)\n";
    cout << "7. Tampilkan Semua Buku Berdasarkan Rak\n";
    cout << "8. Keluar\n";
    cout << "Pilih menu: ";
}
```

- Menampilkan pilihan menu ke layer

15. Fungsi main()

```
int main() {
    int pilihan;
    do {
        tampilkanMenu();
        cin >> pilihan;
        switch (pilihan) {
            case 1: tambahBuku(); break;
            case 2: tambahAnggota(); break;
            case 3: pinjamBuku(); break;
            case 4: kembalikanBuku(); break;
            case 5: cariDanTampilkanBuku(); break;
            case 6: tampilkanGraphRak(); break;
            case 7: tampilkanSemuaBukuBerdasarkanRak(); break;
            case 8: cout << "Terima kasih!\n"; break;
            default: cout << "Pilihan tidak valid.\n";
        }
    } while (pilihan != 8);

    return 0;
}
```

- Perulangan menu
- nput pilihan user
- Menjalankan fungsi sesuai pilihan hingga user memilih keluar

Output Program

1. Menu Utama & Menu Tambah Buku (1)

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

macbookair@MacBooks-MacBook-Air bab11 % g++ -std=c++11 main.cpp -o main && ./main

===== SISTEM PERPUSTAKAAN =====
1. Tambah Buku
2. Tambah Anggota
3. Pinjam Buku
4. Kembalikan Buku
5. Cari Buku
6. Tampilkan Peta Rakç (Graph)
7. Tampilkan Semua Buku Berdasarkan Rak
8. Keluar
Pilih menu: 1
Masukkan ID Buku: 001
Masukkan Judul Buku: Surat Cinta Untuk Starla
Masukkan Lokasi Rak: Rak_A
Buku berhasil ditambahkan!
```

2. Menu Tambah Anggota (2)

```
===== SISTEM PERPUSTAKAAN =====
1. Tambah Buku
2. Tambah Anggota
3. Pinjam Buku
4. Kembalikan Buku
5. Cari Buku
6. Tampilkan Peta Rakç (Graph)
7. Tampilkan Semua Buku Berdasarkan Rak
8. Keluar
Pilih menu: 2
Masukkan ID Anggota: 012345
Masukkan Nama Anggota: Dzulfikar Fathin
Anggota berhasil ditambahkan!
```

3. Menu Pinjam Buku (3)

```
===== SISTEM PERPUSTAKAAN =====
1. Tambah Buku
2. Tambah Anggota
3. Pinjam Buku
4. Kembalikan Buku
5. Cari Buku
6. Tampilkan Peta Rakç (Graph)
7. Tampilkan Semua Buku Berdasarkan Rak
8. Keluar
Pilih menu: 3
Masukkan ID Buku: 001
Masukkan ID Anggota: 012345
Peminjaman dicatat.
```

4. Menu Kembalikan Buku (4)

```
===== SISTEM PERPUSTAKAAN =====
1. Tambah Buku
2. Tambah Anggota
3. Pinjam Buku
4. Kembalikan Buku
5. Cari Buku
6. Tampilkan Peta Rakç (Graph)
7. Tampilkan Semua Buku Berdasarkan Rak
8. Keluar
Pilih menu: 4
Buku berhasil dikembalikan!
```

5. Menu Cari Buku (5)

```
===== SISTEM PERPUSTAKAAN =====
1. Tambah Buku
2. Tambah Anggota
3. Pinjam Buku
4. Kembalikan Buku
5. Cari Buku
6. Tampilkan Peta Rakç (Graph)
7. Tampilkan Semua Buku Berdasarkan Rak
8. Keluar
Pilih menu: 5
Masukkan judul buku yang dicari: Surat Cinta Untuk Starla
Buku ditemukan:
ID: 1, Judul: Surat Cinta Untuk Starla, Lokasi Rak: Rak_A
```

6. Menampilkan Peta Rak (6)

```
===== SISTEM PERPUSTAKAAN =====
1. Tambah Buku
2. Tambah Anggota
3. Pinjam Buku
4. Kembalikan Buku
5. Cari Buku
6. Tampilkan Peta Rakç (Graph)
7. Tampilkan Semua Buku Berdasarkan Rak
8. Keluar
Pilih menu: 6

Peta Lokasi Rak (Adjacency Matrix):

               Rak_A     Rak_B     Rak_C
-----------------------------------------
     Rak_A         1         0         0
     Rak_B         0         1         0
     Rak_C         0         0         1
```

7. Tampilkan Semua Buku Berdasarkan Rak (7)

```
===== SISTEM PERPUSTAKAAN =====
1. Tambah Buku
2. Tambah Anggota
3. Pinjam Buku
4. Kembalikan Buku
5. Cari Buku
6. Tampilkan Peta Rakç (Graph)
7. Tampilkan Semua Buku Berdasarkan Rak
8. Keluar
Pilih menu: 7
Daftar Buku Berdasarkan Rak:
Rak Rak_A:
  - ID: 1, Judul: Surat Cinta Untuk Starla
Rak Rak_B:
  - ID: 2, Judul: I love you 3000
Rak Rak_C:
  - ID: 3, Judul: Phiscology of Money
```