LAPORAN UTS STRUKTUR DATA

Program Studi Teknik Elektro

Fakultas Teknik, Universitas Mulawarman

Judul:

Implementasi Struktur Data dalam Berbagai Studi Kasus

Mata Kuliah:

Struktur Data

Dosen Pengampu:

Dr. Eng. Ir. Aji Ery Burhandenny, S.T., M.AIT.

DAFTAR ISI

1. Pendahuluan

Tugas Ujian Tengah Semester (UTS) ini bertujuan untuk menguji pemahaman mahasiswa terkait implementasi berbagai struktur data seperti array, pointer, stack, queue, dan struct menggunakan bahasa pemrograman C++. Tugas ini bersifat take-home, di mana mahasiswa diwajibkan untuk mengumpulkan solusi kode dan penjelasan melalui repositori GitHub.

Setiap soal yang diberikan dalam UTS ini memiliki konteks studi kasus yang berbeda. Oleh karena itu, laporan ini akan membahas implementasi program secara terstruktur berdasarkan tiap soal, dilengkapi dengan kode program, penjelasan alur, dan hasil uji coba.

2. Pembahasan Setiap Soal

Soal 1: Array dan Pointer

Pada soal ini, program menggunakan array of pointers untuk menyimpan data mahasiswa dalam bentuk struct. Fitur yang diimplementasikan:

a. Menambah, menghapus, dan menampilkan data mahasiswa.
b. Mengurutkan data berdasarkan IPK menggunakan Bubble Sort.

Kode Program:

Kode lengkap terdapat pada Lampiran 1.

Penjelasan Alur Program:

a. Program meminta input berupa NIM, nama, dan IPK mahasiswa.
b. Data disimpan dalam array of pointers dan dapat dihapus berdasarkan NIM.
c. Data mahasiswa diurutkan dengan algoritma Bubble Sort dan ditampilkan.

Soal 2: Struct dan File Handling

Program ini mengelola inventaris peralatan laboratorium menggunakan struct dan file handling. Fitur:

a. Menambah, mengubah, dan menghapus data peralatan.
b. Membaca dan menyimpan data di file teks inventaris.txt.
c. Menampilkan data peralatan yang diurutkan berdasarkan kode peralatan.

Kode Program:

Kode lengkap dapat dilihat pada Lampiran 2.

Penjelasan Alur Program:

a. Setiap peralatan disimpan dengan kode, nama, jumlah, dan kondisi.
b. Data disimpan di file teks dan diurutkan berdasarkan kode saat ditampilkan.

Soal 3: Stack

Soal ini mengimplementasikan stack untuk membuat kalkulator notasi postfix. Program menangani:

a. Operasi aritmatika dasar (+, -, *, /) dan operator pangkat (^).
b. Menampilkan langkah-langkah evaluasi ekspresi postfix.

Kode Program:

Lihat Lampiran 3 untuk kode lengkap.

Penjelasan Alur Program:

a. Setiap angka atau operator dalam ekspresi postfix dibaca.
b. Operator diterapkan pada dua operan teratas di stack, dan hasilnya disimpan kembali di stack.
c. Hasil akhir ditampilkan setelah semua operator dievaluasi.

Soal 4: Queue

Program ini mensimulasikan antrian layanan pelanggan di bank menggunakan queue. Fitur yang diimplementasikan:

a. Menambah pelanggan ke antrian (enqueue) dan melayani pelanggan (dequeue).
b. Menghitung rata-rata waktu tunggu dan menampilkan jumlah pelanggan terlayani serta sisa antrian.

Kode Program:

Kode lengkap tersedia di Lampiran 4.

Penjelasan Alur Program:

a. pelanggan memiliki nomor antrian dan waktu layanan.
b. Terdapat 3 loket layanan yang melayani pelanggan secara bergilir.
c. Program menghitung rata-rata waktu tunggu dan jumlah pelanggan terlayani.

Soal 5: Implementasi Gabungan

Program ini menggabungkan array, pointer, stack, dan queue untuk membuat sistem manajemen perpustakaan sederhana. Fitur:

a. Menambah dan mencari buku berdasarkan ISBN.
b. Menyimpan riwayat peminjaman menggunakan stack.
c. Mengelola antrian peminjaman dengan queue.

Kode Program:

Lihat Lampiran 5 untuk kode lengkap.

Penjelasan Alur Program:

Buku disimpan dalam array of pointers dan dapat dicari dengan ISBN.

Peminjaman dan pengembalian buku dikelola dengan stack dan queue.

3. Kesimpulan

Tugas UTS ini memberikan pemahaman mendalam mengenai implementasi berbagai struktur data dalam pemrograman C++. Setiap soal menguji kemampuan mahasiswa untuk menyelesaikan permasalahan praktis dengan struktur data yang tepat. Dengan demikian, mahasiswa diharapkan dapat lebih terampil dalam memanfaatkan array, pointer, stack, queue, dan file handling dalam pengembangan aplikasi.

4. Lampiran

Lampiran 1: Kode Program Soal 1

```
#include <iostream>
#include <string>
using namespace std;

// Struct untuk menyimpan data mahasiswa
struct Mahasiswa {
    string NIM;
    string nama;
    float IPK;
};

// Fungsi untuk menambah data mahasiswa
void tambahMahasiswa(Mahasiswa* mahasiswa[], int& jumlah) {
    if (jumlah < 10) {
        mahasiswa[jumlah] = new Mahasiswa;
        cout << "Masukkan NIM: ";
        cin >> mahasiswa[jumlah]->NIM;
        cout << "Masukkan Nama: ";
        cin.ignore();
        getline(cin, mahasiswa[jumlah]->nama);
        cout << "Masukkan IPK: ";
        cin >> mahasiswa[jumlah]->IPK;
        jumlah++;
    } else {
        cout << "Array penuh!\n";
    }
}

// Fungsi untuk menghapus data mahasiswa berdasarkan NIM
void hapusMahasiswa(Mahasiswa* mahasiswa[], int& jumlah, const string&
NIM) {
    for (int i = 0; i < jumlah; i++) {
        if (mahasiswa[i]->NIM == NIM) {
            delete mahasiswa[i];
            for (int j = i; j < jumlah - 1; j++) {
                mahasiswa[j] = mahasiswa[j + 1];
            }
            mahasiswa[jumlah - 1] = nullptr;
            jumlah--;
            cout << "Data mahasiswa dengan NIM " << NIM << " telah
dihapus.\n";
            return;
        }
    }
    cout << "Mahasiswa dengan NIM tersebut tidak ditemukan.\n";
}

// Fungsi untuk menampilkan data mahasiswa
void tampilkanMahasiswa(Mahasiswa* mahasiswa[], int jumlah) {
    for (int i = 0; i < jumlah; i++) {
        cout << "NIM: " << mahasiswa[i]->NIM
             << ", Nama: " << mahasiswa[i]->nama
             << ", IPK: " << mahasiswa[i]->IPK << endl;
    }
}
```

```
// Fungsi untuk mengurutkan mahasiswa berdasarkan IPK
void urutkanMahasiswa(Mahasiswa* mahasiswa[], int jumlah) {
    for (int i = 0; i < jumlah - 1; i++) {
        for (int j = 0; j < jumlah - i - 1; j++) {
            if (mahasiswa[j]->IPK > mahasiswa[j + 1]->IPK) {
                swap(mahasiswa[j], mahasiswa[j + 1]);
            }
        }
    }
}

int main() {
    Mahasiswa* mahasiswa[10] = {nullptr};
    int jumlah = 0;
    int pilihan;
    string NIM;

    do {
        cout << "1. Tambah Mahasiswa\n2. Hapus Mahasiswa\n3. Tampilkan
Mahasiswa\n4. Urutkan Mahasiswa\n0. Keluar\n";
        cout << "Masukkan pilihan: ";
        cin >> pilihan;

        switch (pilihan) {
            case 1:
                tambahMahasiswa(mahasiswa, jumlah);
                break;
            case 2:
                cout << "Masukkan NIM mahasiswa yang ingin dihapus: ";
                cin >> NIM;
                hapusMahasiswa(mahasiswa, jumlah, NIM);
                break;
            case 3:
                tampilkanMahasiswa(mahasiswa, jumlah);
                break;
            case 4:
                urutkanMahasiswa(mahasiswa, jumlah);
                break;
        }
    } while (pilihan != 0);

    // Membersihkan memori heap
    for (int i = 0; i < jumlah; i++) {
        delete mahasiswa[i];
    }

    return 0;
}
```

Lampiran 2: Kode Program Soal 2

```
#include <iostream>
#include <fstream>
#include <vector>
#include <algorithm>
using namespace std;

// Struct untuk menyimpan data peralatan
struct Peralatan {
    string kode;
    string nama;
    int jumlah;
    string kondisi;
};

// Fungsi untuk menambah data peralatan
void tambahPeralatan(vector<Peralatan>& inventaris) {
    Peralatan p;
    cout << "Masukkan kode peralatan: ";
    cin >> p.kode;
    cout << "Masukkan nama peralatan: ";
    cin.ignore();
    getline(cin, p.nama);
    cout << "Masukkan jumlah: ";
    cin >> p.jumlah;
    cout << "Masukkan kondisi: ";
    cin.ignore();
    getline(cin, p.kondisi);
    inventaris.push_back(p);
}

// Fungsi untuk mengubah data peralatan berdasarkan kode
void ubahPeralatan(vector<Peralatan>& inventaris) {
    string kode;
    cout << "Masukkan kode peralatan yang ingin diubah: ";
    cin >> kode;

    for (auto& p : inventaris) {
        if (p.kode == kode) {
            cout << "Masukkan nama baru: ";
            cin.ignore();
            getline(cin, p.nama);
            cout << "Masukkan jumlah baru: ";
            cin >> p.jumlah;
            cout << "Masukkan kondisi baru: ";
            cin.ignore();
            getline(cin, p.kondisi);
            cout << "Data berhasil diubah.\n";
            return;
        }
    }
    cout << "Peralatan dengan kode tersebut tidak ditemukan.\n";
}

// Fungsi untuk menghapus data peralatan berdasarkan kode
void hapusPeralatan(vector<Peralatan>& inventaris) {
    string kode;
    cout << "Masukkan kode peralatan yang ingin dihapus: ";
```

```
    cin >> kode;

    auto it = remove_if(inventaris.begin(), inventaris.end(),
                        [&kode](const Peralatan& p) { return p.kode ==
kode; });

    if (it != inventaris.end()) {
        inventaris.erase(it, inventaris.end());
        cout << "Data berhasil dihapus.\n";
    } else {
        cout << "Peralatan dengan kode tersebut tidak ditemukan.\n";
    }
}

// Fungsi untuk menyimpan data ke file
void simpanKeFile(const vector<Peralatan>& inventaris) {
    ofstream file("inventaris.txt");
    for (const auto& p : inventaris) {
        file << p.kode << ',' << p.nama << ',' << p.jumlah << ',' <<
p.kondisi << '\n';
    }
    file.close();
}

// Fungsi untuk membaca data dari file
void bacaDariFile(vector<Peralatan>& inventaris) {
    ifstream file("inventaris.txt");
    if (!file) {
        cout << "File tidak ditemukan.\n";
        return;
    }

    Peralatan p;
    string line;
    while (getline(file, line)) {
        size_t pos = 0;
        pos = line.find(',');
        p.kode = line.substr(0, pos);
        line.erase(0, pos + 1);

        pos = line.find(',');
        p.nama = line.substr(0, pos);
        line.erase(0, pos + 1);

        pos = line.find(',');
        p.jumlah = stoi(line.substr(0, pos));
        line.erase(0, pos + 1);

        p.kondisi = line;
        inventaris.push_back(p);
    }
    file.close();
}

// Fungsi untuk menampilkan laporan inventaris
void tampilkanInventaris(const vector<Peralatan>& inventaris) {
    for (const auto& p : inventaris) {
        cout << "Kode: " << p.kode
             << ", Nama: " << p.nama
```

```
             << ", Jumlah: " << p.jumlah
             << ", Kondisi: " << p.kondisi << '\n';
    }
}

// Fungsi untuk mengurutkan inventaris berdasarkan kode
void urutkanInventaris(vector<Peralatan>& inventaris) {
    sort(inventaris.begin(), inventaris.end(),
         [](const Peralatan& a, const Peralatan& b) { return a.kode <
b.kode; });
}

int main() {
    vector<Peralatan> inventaris;
    bacaDariFile(inventaris);

    int pilihan;
    do {
        cout << "\n1. Tambah Peralatan\n2. Ubah Peralatan\n3. Hapus
Peralatan\n";
        cout << "4. Tampilkan Inventaris\n5. Simpan ke File\n6.
Urutkan Inventaris\n0. Keluar\n";
        cout << "Masukkan pilihan: ";
        cin >> pilihan;

        switch (pilihan) {
            case 1:
                tambahPeralatan(inventaris);
                break;
            case 2:
                ubahPeralatan(inventaris);
                break;
            case 3:
                hapusPeralatan(inventaris);
                break;
            case 4:
                tampilkanInventaris(inventaris);
                break;
            case 5:
                simpanKeFile(inventaris);
                break;
            case 6:
                urutkanInventaris(inventaris);
                cout << "Inventaris berhasil diurutkan.\n";
                break;
        }
    } while (pilihan != 0);

    return 0;
}
```

Lampiran 3: Kode Program Soal 3

```
#include <iostream>
#include <stack>
#include <cmath>   // untuk fungsi pow()
#include <sstream> // untuk parsing input
using namespace std;

// Fungsi untuk mengevaluasi ekspresi postfix
double evaluasiPostfix(const string& ekspresi) {
    stack<double> st;  // Stack untuk menyimpan operan
    stringstream ss(ekspresi);  // Stringstream untuk memisahkan token
    string token;

    // Parsing setiap token dalam ekspresi postfix
    while (ss >> token) {
        if (isdigit(token[0])) {  // Jika token adalah angka
            st.push(stod(token)); // Konversi string ke double dan
push ke stack
        } else {  // Jika token adalah operator
            double operan2 = st.top(); st.pop();  // Ambil operan kedua
            double operan1 = st.top(); st.pop();   // Ambil operan
pertama

            double hasil = 0;
            switch (token[0]) {
                case '+': hasil = operan1 + operan2; break;
                case '-': hasil = operan1 - operan2; break;
                case '*': hasil = operan1 * operan2; break;
                case '/': hasil = operan1 / operan2; break;
                case '^': hasil = pow(operan1, operan2); break;
                default:
                    cout << "Operator tidak valid.\n";
                    return 0;
            }
            st.push(hasil);  // Push hasil ke stack
            cout << "Langkah: " << operan1 << " " << token << " " <<
operan2
                 << " = " << hasil << endl;
        }
    }

    // Hasil akhir adalah elemen terakhir di stack
    return st.top();
}

int main() {
    string ekspresi;
    cout << "Masukkan ekspresi postfix: ";
    getline(cin, ekspresi);  // Membaca seluruh ekspresi

    double hasil = evaluasiPostfix(ekspresi);
    cout << "Hasil akhir: " << hasil << endl;

    return 0;
}
```

Lampiran 4: Kode Program Soal 4

```
#include <iostream>
#include <queue>
#include <vector>
using namespace std;

// Struct untuk menyimpan data pelanggan
struct Pelanggan {
    int nomorAntrian;
    int waktuLayanan;  // Waktu layanan dalam menit
};

// Class Queue untuk mengelola antrian pelanggan
class Queue {
private:
    queue<Pelanggan> q;  // Queue dari STL
public:
    // Menambah pelanggan ke dalam antrian
    void enqueue(int nomor, int waktu) {
        q.push({nomor, waktu});
    }

    // Mengeluarkan pelanggan dari antrian
    Pelanggan dequeue() {
        if (!q.empty()) {
            Pelanggan p = q.front();
            q.pop();
            return p;
        } else {
            return {-1, 0};   // Return pelanggan dummy jika queue
kosong
        }
    }

    // Mengecek apakah antrian kosong
    bool isEmpty() const {
        return q.empty();
    }

    // Mendapatkan ukuran antrian
    int size() const {
        return q.size();
    }
};

// Fungsi untuk menjalankan simulasi antrian dengan 3 loket layanan
void simulasiAntrian(Queue& antrian) {
    int totalWaktuTunggu = 0;
    int totalPelanggan = 0;
    int waktuLoket[3] = {0, 0, 0};  // Waktu layanan untuk 3 loket

    while (!antrian.isEmpty()) {
        for (int i = 0; i < 3; i++) {   // Setiap loket melayani 1
pelanggan
            if (!antrian.isEmpty()) {
                Pelanggan p = antrian.dequeue();
                totalWaktuTunggu += waktuLoket[i];   // Hitung waktu
tunggu
```

```
                waktuLoket[i] += p.waktuLayanan;     // Tambah waktu
layanan ke loket
                totalPelanggan++;
                cout << "Pelanggan " << p.nomorAntrian
                     << " dilayani di loket " << i + 1
                     << " selama " << p.waktuLayanan << " menit.\n";
            }
        }
    }

    // Hitung statistik akhir
    cout << "\nStatistik:\n";
    cout << "Total pelanggan terlayani: " << totalPelanggan << endl;
    cout << "Rata-rata waktu tunggu: "
         << (totalWaktuTunggu / (double)totalPelanggan) << " menit\n";
    cout << "Sisa antrian: " << antrian.size() << endl;
}

int main() {
    Queue antrian;
    int jumlahPelanggan, waktu;

    cout << "Masukkan jumlah pelanggan: ";
    cin >> jumlahPelanggan;

    // Menambahkan pelanggan ke dalam antrian
    for (int i = 1; i <= jumlahPelanggan; i++) {
        cout << "Masukkan waktu layanan untuk pelanggan " << i << ":
";
        cin >> waktu;
        antrian.enqueue(i, waktu);
    }

    // Menjalankan simulasi antrian
    simulasiAntrian(antrian);

    return 0;
}
```

Lampiran 5: Kode Program Soal 5

```
#include <iostream>
#include <string>
#include <stack>
#include <queue>
using namespace std;

// Struct untuk menyimpan data buku
struct Buku {
    string ISBN;
    string judul;
    string pengarang;
    int tahunTerbit;
};

// Stack untuk menyimpan riwayat peminjaman buku
stack<string> riwayatPeminjaman;

// Queue untuk mengelola antrian peminjaman buku
queue<string> antrianPeminjaman;

// Array of pointers untuk menyimpan data buku
Buku* perpustakaan[10] = {nullptr};  // Maksimal 10 buku
int jumlahBuku = 0;

// Fungsi untuk menambah buku
void tambahBuku() {
    if (jumlahBuku < 10) {
        perpustakaan[jumlahBuku] = new Buku;
        cout << "Masukkan ISBN: ";
        cin >> perpustakaan[jumlahBuku]->ISBN;
        cout << "Masukkan Judul: ";
        cin.ignore();
        getline(cin, perpustakaan[jumlahBuku]->judul);
        cout << "Masukkan Pengarang: ";
        getline(cin, perpustakaan[jumlahBuku]->pengarang);
        cout << "Masukkan Tahun Terbit: ";
        cin >> perpustakaan[jumlahBuku]->tahunTerbit;
        jumlahBuku++;
        cout << "Buku berhasil ditambahkan.\n";
    } else {
        cout << "Perpustakaan penuh!\n";
    }
}

// Fungsi untuk mencari buku berdasarkan ISBN
void cariBuku(const string& ISBN) {
    for (int i = 0; i < jumlahBuku; i++) {
        if (perpustakaan[i]->ISBN == ISBN) {
            cout << "Buku ditemukan: " << perpustakaan[i]->judul
                 << " oleh " << perpustakaan[i]->pengarang
                 << ", Tahun: " << perpustakaan[i]->tahunTerbit <<
endl;
            return;
        }
    }
    cout << "Buku dengan ISBN tersebut tidak ditemukan.\n";
}
```

```
// Fungsi untuk menampilkan semua buku
void tampilkanBuku() {
    cout << "\nDaftar Buku:\n";
    for (int i = 0; i < jumlahBuku; i++) {
        cout << "ISBN: " << perpustakaan[i]->ISBN
             << ", Judul: " << perpustakaan[i]->judul
             << ", Pengarang: " << perpustakaan[i]->pengarang
             << ", Tahun Terbit: " << perpustakaan[i]->tahunTerbit <<
endl;
    }
}

// Fungsi untuk meminjam buku
void pinjamBuku(const string& ISBN) {
    antrianPeminjaman.push(ISBN);
    cout << "Buku dengan ISBN " << ISBN << " masuk ke antrian
peminjaman.\n";
}

// Fungsi untuk memproses peminjaman buku
void prosesPeminjaman() {
    if (!antrianPeminjaman.empty()) {
        string ISBN = antrianPeminjaman.front();
        antrianPeminjaman.pop();
        riwayatPeminjaman.push(ISBN);
        cout << "Buku dengan ISBN " << ISBN << " telah dipinjam.\n";
    } else {
        cout << "Tidak ada buku dalam antrian peminjaman.\n";
    }
}

// Fungsi untuk menampilkan riwayat peminjaman
void tampilkanRiwayat() {
    cout << "\nRiwayat Peminjaman Buku:\n";
    stack<string> temp = riwayatPeminjaman;   // Salin stack untuk
ditampilkan
    while (!temp.empty()) {
        cout << "ISBN: " << temp.top() << endl;
        temp.pop();
    }
}

int main() {
    int pilihan;
    string ISBN;

    do {
        cout << "\nMenu Perpustakaan:\n";
        cout << "1. Tambah Buku\n2. Cari Buku\n3. Tampilkan Buku\n";
        cout << "4. Pinjam Buku\n5. Proses Peminjaman\n6. Tampilkan
Riwayat\n0. Keluar\n";
        cout << "Masukkan pilihan: ";
        cin >> pilihan;

        switch (pilihan) {
            case 1:
                tambahBuku();
                break;
```

```
            case 2:
                cout << "Masukkan ISBN buku yang dicari: ";
                cin >> ISBN;
                cariBuku(ISBN);
                break;
            case 3:
                tampilkanBuku();
                break;
            case 4:
                cout << "Masukkan ISBN buku yang ingin dipinjam: ";
                cin >> ISBN;
                pinjamBuku(ISBN);
                break;
            case 5:
                prosesPeminjaman();
                break;
            case 6:
                tampilkanRiwayat();
                break;
        }
    } while (pilihan != 0);

    // Membersihkan memori heap
    for (int i = 0; i < jumlahBuku; i++) {
        delete perpustakaan[i];
    }

    return 0;
}
```