LAPORAN UJIAN TENGAH SEMESTER
STRUKTUR DATA

Tanggal Ujian : Senin, 21 Oktober 2024

Oleh:

Nama : Muhammad Ibnu Yaser
NIM : 2209076051
Program Studi : Teknik Elektro

FAKULTAS TEKNIK
UNIVERSITAS MULAWARMAN

SAMARINDA
2024

1. Latar Belakang

Struktur data merupakan komponen penting dalam pemrograman yang digunakan untuk mengorganisasi dan mengelola data secara efisien. Kemampuan memilih dan menggunakan struktur data yang tepat dapat memengaruhi kinerja dan efisiensi program dalam memproses data. Dalam mata kuliah Struktur Data, mahasiswa diperkenalkan dengan berbagai jenis struktur seperti array, pointer, stack, dan queue. Selain itu, pemahaman mengenai cara kerja file handling juga menjadi bekal penting agar program dapat menyimpan data secara persisten. Pada UTS ini, mahasiswa diminta untuk mengimplementasikan sejumlah program dengan menggunakan beberapa konsep penting dalam struktur data. Setiap soal berfokus pada penerapan teknik berbeda, mulai dari pengelolaan data mahasiswa dengan array dan pointer, hingga simulasi sistem perpustakaan yang menggabungkan berbagai struktur data. Penguasaan konsep-konsep ini sangat diperlukan karena berbagai situasi dunia nyata menuntut pemecahan masalah dengan metode penyimpanan dan pemrosesan data yang efisien.

2. Penjelasan Soal

Soal 1 : Array dan Pointer

Deskripsi dan Implementasi:

Pada soal ini, program dirancang untuk mengelola data mahasiswa menggunakan struct dan array of pointers. Setiap mahasiswa memiliki NIM, nama, dan IPK yang disimpan sebagai anggota struct. Data mahasiswa kemudian dikelola menggunakan array pointer agar program dapat menyimpan elemen secara dinamis. Algoritma bubble sort digunakan untuk mengurutkan data mahasiswa berdasarkan IPK dalam urutan descending.

Fitur Utama:

Menambah Data Mahasiswa – Input informasi mahasiswa baru ke dalam array pointer.

Menghapus Data Mahasiswa – Menghapus data mahasiswa terakhir dalam array.

Menampilkan Daftar Mahasiswa – Menampilkan semua data yang tersimpan.

Mengurutkan Berdasarkan IPK – Menggunakan bubble sort untuk mengurutkan mahasiswa berdasarkan IPK.

Codingan :

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

struct Mahasiswa {
    string NIM;
    string nama;
    float IPK;
};

void tambahMahasiswa(Mahasiswa* arr[], int& count) {
    if (count < 10) {
        arr[count] = new Mahasiswa();
        cout << "Masukkan NIM: "; cin >> arr[count]->NIM;
        cout << "Masukkan Nama: "; cin >> arr[count]->nama;
        cout << "Masukkan IPK: "; cin >> arr[count]->IPK;
        count++;
    } else {
        cout << "Data mahasiswa penuh!\n";
    }
}

void hapusMahasiswa(Mahasiswa* arr[], int& count) {
    if (count > 0) {
        delete arr[count - 1];
        count--;
        cout << "Mahasiswa terakhir berhasil dihapus.\n";
    } else {
        cout << "Tidak ada data untuk dihapus.\n";
    }
}

void tampilkanMahasiswa(Mahasiswa* arr[], int count) {
    for (int i = 0; i < count; i++) {
        cout << "NIM: " << arr[i]->NIM << ", Nama: " << arr[i]->nama
             << ", IPK: " << arr[i]->IPK << endl;
    }
}

void urutkanMahasiswa(Mahasiswa* arr[], int count) {
    for (int i = 0; i < count - 1; i++) {
        for (int j = 0; j < count - i - 1; j++) {
            if (arr[j]->IPK < arr[j + 1]->IPK) {
                swap(arr[j], arr[j + 1]);
            }
        }
    }
    cout << "Mahasiswa berhasil diurutkan berdasarkan IPK.\n";
}

int main() {
    Mahasiswa* mahasiswa[10];
    int count = 0;
    int pilihan;
    do {
        cout << "1. Tambah Mahasiswa\n2. Hapus Mahasiswa\n"
             <<  "3.  Tampilkan  Mahasiswa\n4.  Urutkan  Mahasiswa\n0.
Keluar\n";
```

```
        cout << "Pilih: "; cin >> pilihan;
        switch (pilihan) {
            case 1: tambahMahasiswa(mahasiswa, count); break;
            case 2: hapusMahasiswa(mahasiswa, count); break;
            case 3: tampilkanMahasiswa(mahasiswa, count); break;
            case 4: urutkanMahasiswa(mahasiswa, count); break;
        }
    } while (pilihan != 0);
    return 0;
}
```

Soal 2 : Struct dan File Handling

Deskripsi dan Implementasi:

Pada soal ini, program dibuat untuk mengelola inventaris peralatan laboratorium. Setiap peralatan dicatat dalam struct dengan beberapa atribut seperti kode, nama, jumlah, dan kondisi. Data disimpan secara permanen dalam file teks menggunakan fungsi ofstream dan ifstream. Program ini memastikan bahwa informasi tetap tersedia meskipun aplikasi ditutup, dengan file handling berperan penting untuk menyimpan dan membaca data.

Fitur Utama:

Menambah Peralatan Baru – Menambahkan data baru ke file inventaris.

Menampilkan Data Peralatan – Membaca dan menampilkan semua data dari file.

Codingan :

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

struct Peralatan {
    string kode;
    string nama;
    int jumlah;
    string kondisi;
};

void tambahPeralatan() {
    Peralatan p;
    ofstream file("inventaris.txt", ios::app);
    cout << "Masukkan kode: "; cin >> p.kode;
    cout << "Masukkan nama: "; cin >> p.nama;
    cout << "Masukkan jumlah: "; cin >> p.jumlah;
    cout << "Masukkan kondisi: "; cin >> p.kondisi;
    file << p.kode << " " << p.nama << " " << p.jumlah << " " << p.kondisi
<< endl;
    file.close();
}
```

```
void tampilkanPeralatan() {
    ifstream file("inventaris.txt");
    string line;
    while (getline(file, line)) {
        cout << line << endl;
    }
    file.close();
}

int main() {
    int pilihan;
    do {
        cout << "1. Tambah Peralatan\n2. Tampilkan Peralatan\n0.
Keluar\n";
        cout << "Pilih: "; cin >> pilihan;
        if (pilihan == 1) tambahPeralatan();
        else if (pilihan == 2) tampilkanPeralatan();
    } while (pilihan != 0);
    return 0;
}
```

Soal 3 : Kalkulator Postfix (Stack)

Deskripsi dan Implementasi:

Program ini memanfaatkan stack untuk menghitung ekspresi postfix. Dalam notasi postfix, operator ditempatkan setelah operand sehingga tidak memerlukan tanda kurung untuk prioritas operasi. Program memasukkan angka ke dalam stack, dan ketika menemukan operator, dua angka teratas dikeluarkan untuk dilakukan operasi sesuai dengan operator tersebut. Program mendukung operasi dasar seperti penjumlahan, pengurangan, perkalian, pembagian, dan pangkat.

Fitur Utama:

Push dan Pop – Menambahkan dan mengeluarkan nilai dari stack.

Evaluasi Ekspresi – Memproses ekspresi postfix sesuai urutan operand dan operator.

Operator Pendukung – +, -, *, /, dan ^.

Codingan :

```
#include <iostream>
#include <stack>
#include <cmath>
using namespace std;

int hitungPostfix(string ekspresi) {
    stack<int> s;
    for (char c : ekspresi) {
        if (isdigit(c)) {
            s.push(c - '0');
```

```
        } else {
            int b = s.top(); s.pop();
            int a = s.top(); s.pop();
            switch (c) {
                case '+': s.push(a + b); break;
                case '-': s.push(a - b); break;
                case '*': s.push(a * b); break;
                case '/': s.push(a / b); break;
                case '^': s.push(pow(a, b)); break;
            }
        }
    }
    return s.top();
}

int main() {
    string ekspresi;
    cout << "Masukkan ekspresi postfix: "; cin >> ekspresi;
    cout << "Hasil: " << hitungPostfix(ekspresi) << endl;
    return 0;
}
```

Soal 4 : Simulasi Antrian Bank (Queue)

Deskripsi dan Implementasi:

Soal ini menuntut simulasi antrian layanan bank menggunakan queue. Setiap pelanggan memiliki nomor antrian dan estimasi waktu layanan tertentu. Program ini menggunakan prinsip FIFO (First-In, First-Out) untuk memastikan bahwa pelanggan yang datang lebih awal dilayani terlebih dahulu. Selain itu, statistik seperti total waktu layanan dan jumlah pelanggan terlayani juga dihitung.

Fitur Utama:

Enqueue dan Dequeue – Menambahkan dan menghapus pelanggan dari antrian.

Statistik Layanan – Menghitung total waktu dan jumlah pelanggan terlayani.

Codingan :

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

struct Pelanggan {
    int nomor;
    int waktuLayanan;
};

int main() {
    queue<Pelanggan> antrian;
    int totalWaktu = 0, jumlahPelanggan = 0;
```

```
    // Menambah beberapa pelanggan ke antrian
    for (int i = 1; i <= 5; i++) {
        antrian.push({i, 5});
    }

    while (!antrian.empty()) {
        Pelanggan p = antrian.front();
        antrian.pop();
        cout << "Melayani pelanggan " << p.nomor << " selama "
             << p.waktuLayanan << " menit.\n";
        totalWaktu += p.waktuLayanan;
        jumlahPelanggan++;
    }

    cout << "Total waktu layanan: " << totalWaktu << " menit\n";
    cout << "Jumlah pelanggan terlayani: " << jumlahPelanggan << endl;
    return 0;
}
```

Soal 5 : Implementasi Gabungan (Manajemen Perpustakaan)

Deskripsi dan Implementasi:

Program ini adalah simulasi sistem perpustakaan yang menggabungkan berbagai struktur data. Array of pointers digunakan untuk menyimpan data buku, sementara stack mencatat riwayat peminjaman, dan queue mengelola antrian peminjaman buku. Fitur pencarian buku berdasarkan ISBN juga disediakan, sehingga program ini memberikan pengalaman manajemen perpustakaan yang lengkap.

Fitur Utama:

Manajemen Buku – Menambah dan mencari buku dalam koleksi perpustakaan.

Stack untuk Riwayat Peminjaman – Melacak buku yang dipinjam.

Queue untuk Antrian Peminjaman – Mengatur antrean pelanggan yang ingin meminjam buku.

Codingan :

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

// Struct untuk Buku
struct Buku {
    string ISBN;
    string judul;
    string pengarang;
```

```
    int tahunTerbit;
};

// Struct untuk Pelanggan
struct Pelanggan {
    string nama;
    string ISBNBuku;
};

// Deklarasi Array of Pointers untuk Buku
Buku* koleksiBuku[10];
int jumlahBuku = 0;

// Stack untuk Riwayat Peminjaman Buku
stack<string> riwayatPeminjaman;

// Queue untuk Antrian Peminjaman Buku
queue<Pelanggan> antrianPeminjaman;

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

// Fungsi Mencari Buku Berdasarkan ISBN
void cariBuku(string isbn) {
    bool ditemukan = false;
    for (int i = 0; i < jumlahBuku; i++) {
        if (koleksiBuku[i]->ISBN == isbn) {
            cout << "Buku Ditemukan:\n";
            cout << "ISBN: " << koleksiBuku[i]->ISBN << "\n"
                 << "Judul: " << koleksiBuku[i]->judul << "\n"
                 << "Pengarang: " << koleksiBuku[i]->pengarang << "\n"
                 << "Tahun Terbit: " << koleksiBuku[i]->tahunTerbit <<
"\n";
            ditemukan = true;
            break;
        }
    }
    if (!ditemukan) {
        cout << "Buku tidak ditemukan!\n";
    }
}

// Fungsi Menampilkan Semua Buku
void tampilkanBuku() {
```

```
    if (jumlahBuku == 0) {
        cout << "Tidak ada buku dalam koleksi.\n";
    } else {
        cout << "Daftar Buku:\n";
        for (int i = 0; i < jumlahBuku; i++) {
            cout << i + 1 << ". " << koleksiBuku[i]->judul << " (ISBN:
"
                 << koleksiBuku[i]->ISBN << ")\n";
        }
    }
}

// Fungsi Menambah Antrian Peminjaman Buku
void tambahAntrian() {
    Pelanggan pelanggan;
    cout << "Masukkan Nama Pelanggan: "; cin.ignore();
    getline(cin, pelanggan.nama);
    cout << "Masukkan ISBN Buku yang Ingin Dipinjam: "; cin >>
pelanggan.ISBNBuku;
    antrianPeminjaman.push(pelanggan);
    cout << "Pelanggan berhasil ditambahkan ke antrian.\n";
}

// Fungsi Memproses Peminjaman Buku (Dequeue)
void prosesPeminjaman() {
    if (!antrianPeminjaman.empty()) {
        Pelanggan pelanggan = antrianPeminjaman.front();
        antrianPeminjaman.pop();
        riwayatPeminjaman.push(pelanggan.ISBNBuku);
        cout << "Peminjaman untuk pelanggan " << pelanggan.nama
             << " berhasil diproses. Buku ISBN: " << pelanggan.ISBNBuku
<< "\n";
    } else {
        cout << "Tidak ada antrian peminjaman.\n";
    }
}

// Fungsi Menampilkan Riwayat Peminjaman
void tampilkanRiwayat() {
    if (riwayatPeminjaman.empty()) {
        cout << "Belum ada peminjaman.\n";
    } else {
        cout << "Riwayat Peminjaman Buku:\n";
        stack<string> temp = riwayatPeminjaman; // Salinan stack untuk
ditampilkan
        while (!temp.empty()) {
            cout << "ISBN: " << temp.top() << "\n";
            temp.pop();
        }
    }
}

// Fungsi Menu Utama
void menu() {
    int pilihan;
    do {
        cout << "\n=== Sistem Manajemen Perpustakaan ===\n";
        cout << "1. Tambah Buku\n";
        cout << "2. Cari Buku Berdasarkan ISBN\n";
```

```
        cout << "3. Tampilkan Semua Buku\n";
        cout << "4. Tambah Antrian Peminjaman\n";
        cout << "5. Proses Peminjaman Buku\n";
        cout << "6. Tampilkan Riwayat Peminjaman\n";
        cout << "0. Keluar\n";
        cout << "Pilih: "; cin >> pilihan;

        switch (pilihan) {
            case 1: tambahBuku(); break;
            case 2: {
                string isbn;
                cout << "Masukkan ISBN: "; cin >> isbn;
                cariBuku(isbn);
                break;
            }
            case 3: tampilkanBuku(); break;
            case 4: tambahAntrian(); break;
            case 5: prosesPeminjaman(); break;
            case 6: tampilkanRiwayat(); break;
            case 0: cout << "Keluar dari program.\n"; break;
            default: cout << "Pilihan tidak valid!\n";
        }
    } while (pilihan != 0);
}

int main() {
    menu();
    return 0;
}
```

3. Kesimpulan Akhir

Tugas UTS ini memberikan pemahaman mendalam tentang bagaimana setiap struktur data memiliki fungsi spesifik dan dapat diterapkan untuk memecahkan masalah yang berbeda. Array dan pointer memberikan fleksibilitas dalam pengelolaan data, stack dan queue membantu menyelesaikan masalah yang memerlukan urutan tertentu, dan file handling menjaga data tetap tersedia meskipun aplikasi berhenti. Implementasi gabungan dalam manajemen perpustakaan juga memperlihatkan bagaimana struktur data bekerja bersama untuk menciptakan solusi yang efektif.