Telkom University Purwokerto

Aturan Praktikum Struktur Data

1. **Akun GitHub:** Setiap praktikan wajib memiliki akun GitHubyang aktif dan digunakan selama praktikum berlangsung.
2. **Invite Collaborator:** Setiap praktikan diwajibkan untuk menambahkan collaborator di setiap repository
    a. Asisten Praktikum: AndiniNH
    b. AsistenPraktikum:4ldiputra
3. **Repository Praktikum:** Setiap praktikan diwajibkan untuk membuat satu repository di GitHub yang akan digunakan untuk seluruh tugas dan laporan praktikum. Repository ini harus diatur dengan rapi dan sesuai dengan instruksi yang akan diberikan di lampiran.
4. **Penamaan Folder:** Penamaan folderdalam repositoryakan dibahas secara rinci di lampiran. Praktikan wajib mengikuti aturan penamaan yang telah ditentukan.

Nomor	Pertemuan	Penamaan
1	Penganalan Bahasa C++ Bagian Pertama	01_Pengenalan_CPP_Bagian_1
2	Pengenalan Bahasa C++ Bagian Kedua	02_Pengenalan_CPP_Bagian_2
3	Abstract Data Type	03_Abstract_Data_Type
4	Single Linked List Bagian Pertama	04_Single_Linked_List_Bagian_1
5	Single Linked List Bagian Kedua	05_Single_Linked_List_Bagian_2
6	Double Linked List Bagian Pertama	06_Double_Linked_List_Bagian_1
7	Stack	07_Stack
8	Queue	08_Queue
9	Assessment Bagian Pertama	09_Assessment_Bagian_1
10	Tree Bagian Pertama	10_Tree_Bagian_1
11	Tree Bagian Kedua	11_Tree_Bagian_2
12	Asistensi Tugas Besar	12_Asistensi_Tugas_Besar
13	Multi Linked List	13_Multi_Linked_List
14	Graph	14_Graph
15	Assessment Bagian Kedua	15_Assessment_Bagian_2
16	Tugas Besar	16_Tugas_Besar

5. **Jam Praktikum:**
    - Jam masuk praktikum adalah **1 jam lebih lambat** dari jadwal yang tercantum. Sebagai contoh, jika jadwalpraktikumadalah pukul 06.30 - 09.30, maka aturan praktikum akan diatur sebagai berikut:
        - **06.30 - 07.30**: Waktuini digunakan untuk **Tugas Praktikum dan Laporan Praktikum** yang dilakukan di luar laboratorium.
        - **07.30 - 0G.30**: Sesi ini mencakup **tutorial, diskusi, dan kasus problem- solving**. Kegiatan ini berlangsung di dalam laboratorium dengan alokasi waktu sebagai berikut:
            - **60 menit pertama**: Tugas terbimbing.
            - **60 menit kedua**: Tugas mandiri.
6. **Pengumpulan Tugasn Pendahuluan:** Tugas Pendahuluan (TP) wajib dikumpulkan melalui GitHub sesuai dengan format berikut:

    nama_repo/nama_pertemuan/TP_Pertemuan_Ke.md

    Sebagai contoh:

    STD_Yudha_Islalmi_Sulistya_XXXXXXXX/01_Running_Modul/TP_01.md

7. **Pengecekan Tugas Pendahuluan:** Pengumpulan laporan praktikum akan diperiksa **1 hari sebelum praktikum selanjutnya** dimulai. Pastikan tugas telah diunggah tepat waktu untuk menghindari sanksi.

LAPORAN PRAKTIKUM
MODUL 3
ABSTRACT DATA TYPE (ADT)

Disusun Oleh :

Izzaty Zahara Br Barus – 2311104052

Kelas :

SE-07-02

Dosen :

Wahyu Andi Saputra, S.pd,M.Eng

PROGRAM STUDI SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO

2024

I. TUJUAN

1. Memahami konsep Abstract Data Type (ADT) dan penggunaannya dalam program

II. LANDASAN TEORI

1. Abstract Data Type (ADT)

Abstract Data Type (ADT) adalah tipe data yang didefinisikan secara abstrak bersama dengan serangkaian operasi dasar (primitif) yang dapat dilakukan pada tipe tersebut. ADT mencakup definisi tipe, invarian, dan aksioma yang harus dipatuhi. ADT memungkinkan kita untuk memisahkan antara definisi tipe dan implementasi operasinya, sehingga implementasi menjadi modular.
Dalam contoh yang diberikan, ADT mahasiswa didefinisikan dengan tipe data seperti **nim**, **nilai1**, dan **nilai2**. Primitif yang digunakan termasuk **inputMhs()** untuk memasukkan data mahasiswa dan **rata2()** untuk menghitung rata-rata nilai mahasiswa. Implementasi ini dipisah menjadi file header (*.h*) yang berisi spesifikasi tipe dan file source (*.cpp*) yang berisi realisasi dari fungsi tersebut.

III. GUIDE

```cpp
#include<iostream>

using namespace std;

    struct mahasiswa {
        char nim[10];
        int nilai1,nilai2;
    };

    void inputMhs(mahasiswa &m);
    float rata2(mahasiswa m);

    int main () {
        mahasiswa mhs;
        inputMhs(mhs);
        cout << "rata-rata = " << rata2(mhs);
        return 0;
    }

    void inputMhs(mahasiswa &m){
        cout << "Input nim= ";
        cin >> (m).nim;
        cout <<"input nilai =";
        cin >> (m).nilai1;
        cout <<"input nilai =";
        cin >> (m).nilai2;
    }
```

```
float rata2(mahasiswa m) {
    return(m.nilai1+m.nilai2) /2;
}
```

Output

```
PS D:\STD.02\Modul.03\Modul.03\GUIDE\output> & .\'modul3.exe'
Input nim= 2311104052
input nilai =90
input nilai =90
rata-rata = 90
PS D:\STD.02\Modul.03\Modul.03\GUIDE\output>
```

Penjelasan

Program di atas mengelola data seorang mahasiswa dan menghitung rata-rata dari dua nilai yang dimasukkan. Berikut penjelasannya:

1. **Struktur mahasiswa**: Program menggunakan struktur data untuk menyimpan informasi mahasiswa, yaitu NIM (nim), dan dua nilai ujian (nilai1 dan nilai2).
2. **Fungsi inputMhs**: Fungsi ini digunakan untuk menerima input dari pengguna, yaitu NIM serta dua nilai ujian. Data tersebut dimasukkan ke dalam variabel bertipe mahasiswa.
3. **Fungsi rata2**: Fungsi ini menghitung rata-rata dari dua nilai ujian mahasiswa yang sudah diinput sebelumnya.
4. **Program Utama (main)**: Program akan meminta pengguna untuk memasukkan NIM dan dua nilai, lalu menghitung dan menampilkan rata-rata dari nilai-nilai tersebut.

Secara keseluruhan, program ini sederhana dan bertujuan untuk menampilkan rata-rata dua nilai dari satu mahasiswa.

IV. UNGUIDED

1. Task 1

```
#include <iostream>
#include <string>

using namespace std;

// Struktur data untuk mahasiswa
struct Mahasiswa {
    string nama;
    string nim;
```

```
    float nilaiUts;
    float nilaiUas;
    float nilaiTugas;
    float nilaiAkhir;
};

// Fungsi untuk menghitung nilai akhir
float hitungNilaiAkhir(float nilaiUts, float nilaiUas, float
nilaiTugas) {
    return 0.3 * nilaiUts + 0.4 * nilaiUas + 0.3 * nilaiTugas;
}

int main() {
    int jumlahMahasiswa;
    cout << "Masukkan jumlah mahasiswa (maksimal 5): ";
    cin >> jumlahMahasiswa;

    // Validasi input
    while (jumlahMahasiswa > 5) {
        cout << "Inputan terlalu banyak. Silakan input ulang: ";
        cin >> jumlahMahasiswa;
    }

    // Deklarasi array mahasiswa
    Mahasiswa mahasiswa[jumlahMahasiswa];

    // Input data mahasiswa
    for (int i = 0; i < jumlahMahasiswa; i++) {
        cout << "Masukkan nama mahasiswa " << i + 1 << ": ";
        cin >> mahasiswa[i].nama;

        cout << "Masukkan NIM mahasiswa " << i + 1 << ": ";
        cin >> mahasiswa[i].nim;

        cout << "Masukkan nilai UTS mahasiswa " << i + 1 << ": ";
        cin >> mahasiswa[i].nilaiUts;

        cout << "Masukkan nilai UAS mahasiswa " << i + 1 << ": ";
        cin >> mahasiswa[i].nilaiUas;

        cout << "Masukkan nilai tugas mahasiswa " << i + 1 << ":
";
        cin >> mahasiswa[i].nilaiTugas;

        // Hitung nilai akhir
                                    mahasiswa[i].nilaiAkhir       =
hitungNilaiAkhir(mahasiswa[i].nilaiUts,     mahasiswa[i].nilaiUas,
mahasiswa[i].nilaiTugas);
```

```
    }

    // Tampilkan data mahasiswa
    cout << "\nData Mahasiswa:\n";
    for (int i = 0; i < jumlahMahasiswa; i++) {
        cout << "Nama: " << mahasiswa[i].nama << endl;
        cout << "NIM: " << mahasiswa[i].nim << endl;
        cout << "Nilai UTS: " << mahasiswa[i].nilaiUts << endl;
        cout << "Nilai UAS: " << mahasiswa[i].nilaiUas << endl;
       cout << "Nilai Tugas: " << mahasiswa[i].nilaiTugas << endl;
       cout << "Nilai Akhir: " << mahasiswa[i].nilaiAkhir << endl;
        cout << "*\n";
    }

    return 0;
}
```

Output

```
PS D:\STD.02\Modul.03\Modul.03\UNGUIDED\output> & .\'TASK1.exe'
Masukkan jumlah mahasiswa (maksimal 5): 1
Masukkan nama mahasiswa 1: izza
Masukkan NIM mahasiswa 1: 2311104052
Masukkan nilai UTS mahasiswa 1: 90
Masukkan nilai UAS mahasiswa 1: 90
Masukkan nilai tugas mahasiswa 1: 90

Data Mahasiswa:
Nama: izza
NIM: 2311104052
Nilai UTS: 90
Nilai UAS: 90
Nilai Tugas: 90
Nilai Akhir: 90
*
```

Penjelasan

Program ini mengelola data mahasiswa dengan struktur Mahasiswa yang menyimpan informasi seperti **nama**, **NIM**, serta nilai **UTS**, **UAS**, dan **tugas**. Ada fungsi hitungNilaiAkhir untuk menghitung nilai akhir berdasarkan bobot tertentu.
Program melakukan:

1. **Input**: Jumlah mahasiswa (maksimal 5) dan detail tiap mahasiswa (nama, NIM, nilai).
2. **Perhitungan**: Menghitung nilai akhir untuk setiap mahasiswa.
3. **Output**: Menampilkan data mahasiswa lengkap dengan nilai akhir.

Program juga memastikan jumlah input mahasiswa tidak melebihi batas (5).

2. Task 2

```cpp
#include <iostream>
#include "TASK2_PELAJARAN.CPP"

using namespace std;

int main() {
    string namapel = "LOGIKA MATEMATIKA";
    string kodepel = "LOGMAT";
    pelajaran pel = create_pelajaran(namapel, kodepel);
    tampil_pelajaran(pel);
    return 0;
}
```

```cpp
#include <iostream>
#include <string>

using namespace std;

struct pelajaran {
    string namamapel;
    string kodepel;
};

pelajaran create_pelajaran(string namapel, string kodepel) {
    pelajaran pel;
    pel.namamapel = namapel;
    pel.kodepel = kodepel;
    return pel;
}

void tampil_pelajaran(pelajaran pel) {
    cout << "Nama Mata Kuliah : " << pel.namamapel << endl;
    cout << "Nilai : " << pel.kodepel << endl;
}
```

Output

```
PS D:\STD.02\Modul.03> cd 'd:\STD.02\Modul.03\Modul.03\UNGUIDED\output'
PS D:\STD.02\Modul.03\Modul.03\UNGUIDED\output> & .\'TASK2_MAIN.exe'
Nama Mata Kuliah : LOGIKA MATEMATIKA
Nilai : LOGMAT
PS D:\STD.02\Modul.03\Modul.03\UNGUIDED\output>
```

Penjelasan
Berikut adalah penjelasan singkat dari program di atas:
1. Library dan Namespace:
- `#include <iostream>`: Mengimpor pustaka untuk input/output.
- `#include <string>`: Mengimpor pustaka untuk menggunakan tipe data string.

- `using namespace std`: Menghindari penulisan `std::` setiap kali menggunakan fungsi dari standard library, seperti `cout` dan `cin`.
2. Struktur `pelajaran`:
- Struktur (`struct`) pelajaran memiliki dua atribut:
- `namamapel`: Menyimpan nama mata pelajaran.
- `kodepel`: Menyimpan kode mata pelajaran.
3. Fungsi `create_pelajaran`:
- Fungsi ini menerima dua parameter: namapel (nama pelajaran) dan kodepel (kode pelajaran).
- Fungsi membuat objek `pelajaran` baru, mengisi atributnya dengan nilai yang diterima, lalu mengembalikan objek tersebut.
4. Fungsi `tampil_pelajaran`:
- Fungsi ini menerima objek `pelajaran` sebagai parameter dan menampilkan nama dan kode pelajaran menggunakan `cout`.
5. File `main.cpp`:
- Program utama dimulai dengan mendeklarasikan dua variabel string: `namapel` (dengan nilai LOGIKA MATEMATIKA) dan `kodepel` (dengan nilai LOGMAT).
- Fungsi `create_pelajaran` dipanggil untuk membuat objek `pelajaran` dengan data tersebut.
- Fungsi `tampil_pelajaran` digunakan untuk menampilkan nama dan kode pelajaran ke layar.
- Program kemudian selesai dengan return `0`.
Secara keseluruhan, program ini mendefinisikan struktur pelajaran, membuat objek pelajaran dengan data tertentu, dan kemudian menampilkan datanya di layar.

3. Task 3

```
#include <iostream>

using namespace std;

void menampilkanArray(int arr[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }
}

void menukarArray(int arr1[3][3], int arr2[3][3], int posisi1[2],
int posisi2[2]) {
    int temp = arr1[posisi1[0]][posisi1[1]];
    arr1[posisi1[0]][posisi1[1]] = arr2[posisi2[0]][posisi2[1]] +
1;
    arr2[posisi2[0]][posisi2[1]] = temp + 2;
}
```

```
void menukarPointer(int* ptr1, int* ptr2) {
    int temp = *ptr1;
    *ptr1 = *ptr2 ;
    *ptr2 = temp ;
}

int main() {
    int arr1[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int arr2[3][3] = {{10, 11, 12}, {13, 14, 15}, {16, 17, 18}};

    int var1 = 20;
    int var2 = 30;
    int* ptr1 = &var1;
    int* ptr2 = &var2;

    cout << "Isi Array 1 : " << endl;
    menampilkanArray(arr1);
    cout << "Isi Array 2 : " << endl;
    menampilkanArray(arr2);
    cout << "" << endl;

    int posisi1[2] = {1,1};
    int posisi2[2] = {2,2};
    menukarArray(arr1, arr2, posisi1, posisi2);

    cout << "Isi Nilai Array 1 Setelah Ditukar:" << endl;
    menampilkanArray(arr1);
    cout << "Isi Nilai Array 2 Setelah Ditukar:" << endl;
    menampilkanArray(arr2);
    cout << "*" << endl;

    menukarPointer(ptr1, ptr2);

    cout << "Isi Nilai Variabel 1 Setelah Ditukar: " << *ptr1 <<
endl;
    cout << "Isi Nilai Variabel 2 Detelah Ditukar: " << *ptr2 <<
endl;
    cout << "*" << endl;

    return 0;
}
```

Output

```
PS D:\STD.02\Modul.03\Modul.03\UNGUIDED\output> & .\'TASK3.exe'
Isi Array 1 :
1 2 3
4 5 6
7 8 9
Isi Array 2 :
10 11 12
13 14 15
16 17 18

Isi Nilai Array 1 Setelah Ditukar:
1 2 3
4 18 6
7 8 9
Isi Nilai Array 2 Setelah Ditukar:
10 11 12
13 14 15
16 17 5
*
Isi Nilai Variabel 1 Setelah Ditukar: 30
Isi Nilai Variabel 2 Setelah Ditukar: 20
*
```

Penjelasan

Program di atas memiliki dua fungsi utama untuk menukar data dalam array 2D dan juga menggunakan pointer untuk menukar nilai variabel. Berikut penjelasannya:

1. Fungsi menampilkanArray: Fungsi ini digunakan untuk menampilkan isi dari sebuah array 2D ukuran 3x3 dengan cara mencetak setiap elemen array di konsol.
2. Fungsi menukarArray: Fungsi ini menukar elemen dari dua array 2D pada posisi tertentu. Pada contoh ini, elemen pada posisi [1,1] dari arr1 ditukar dengan elemen pada posisi [2,2] dari arr2.
3. Fungsi menukarPointer: Fungsi ini menukar nilai dari dua variabel menggunakan pointer. Variabel var1 dan var2 ditukar nilainya melalui pointer ptr1 dan ptr2.
4. Program Utama (main):
    - Program menampilkan isi dari dua array 3x3 (arr1 dan arr2).
    - Kemudian, elemen dari kedua array pada posisi tertentu ditukar.
    - Setelah itu, program menampilkan kembali kedua array untuk menunjukkan hasil pertukaran.
    - Program juga menukar nilai dari variabel var1 dan var2 menggunakan fungsi menukarPointer dan menampilkan hasilnya.

Secara keseluruhan, program ini menunjukkan cara menukar elemen pada array dan cara menukar nilai variabel menggunakan pointer.

V. KESIMPULAN

Kesimpulan dari laporan ini adalah:

1. Pemahaman tentang ADT: Praktikum ini membantu memahami konsep Abstract Data Type (ADT), bagaimana tipe data didefinisikan secara abstrak, dan bagaimana operasi-operasi dasar dapat dilakukan pada tipe tersebut.

2. Implementasi Modular: Dengan menggunakan ADT, implementasi program menjadi modular. Spesifikasi dan realisasi dipisah, memudahkan pengelolaan dan pembaruan kode program tanpa merusak keseluruhan sistem.
3. Penggunaan ADT dalam Program: Contoh program yang dibuat menunjukkan bagaimana struktur data, seperti mahasiswa atau pelajaran, dapat diatur menggunakan ADT. Operasi-operasi seperti menghitung nilai akhir mahasiswa dan menampilkan data pelajaran juga diimplementasikan melalui fungsi ADT.
4. Penerapan Operasi pada Array dan Pointer: Pada tugas terakhir, ditunjukkan bagaimana operasi penukaran nilai dapat diterapkan pada array dua dimensi dan pointer, memberikan gambaran tentang manipulasi data pada tingkat yang lebih rendah menggunakan C++.
   Secara keseluruhan, konsep ADT mempermudah manajemen tipe data dan operasinya, sekaligus menjaga keteraturan kode dalam proyek besar.