LAPORAN PRAKTIKUM
MODUL 3
“ABSTRACT DATA TYPE (ADT)”

Disusun Oleh:

Dimas Abhipraya Ramansyah **(**2311104069)

Kelas SISE-07-02

Dosen :

Wahyu Andi Saputra, S.pd, M.eng

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

PURWOKERTO

2024

1. **Tujuan**

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

2. **Landasan Teori**

1. Abstract Data Type

Abstract Data Type (ADT) adalah model matematika yang mendefinisikan tipe data dengan mengaitkannya pada operasi-operasi yang dapat dilakukan terhadap data tersebut. Konsep ini berfungsi untuk memisahkan struktur penyimpanan dari perilaku tipe data, sehingga pemrogram tidak perlu mengetahui rincian implementasi internal dari tipe data tersebut. ADT memiliki beberapa tujuan dan manfaat, seperti modularitas, yang memungkinkan pengembangan sistem secara terpisah tanpa saling mengganggu, serta pengkapsulan, yang menyembunyikan informasi internal dari pengguna. Dengan demikian, perubahan pada implementasi tidak akan mempengaruhi program yang menggunakan ADT tersebut. Selain itu, ADT juga memberikan abstraksi, sehingga pemrogram dapat fokus pada perilaku objek tanpa harus memperhatikan detail implementasi.

Dalam struktur ADT terdapat dua komponen utama: definisi tipe, yang menyediakan spesifikasi tentang bagaimana tipe data diorganisir, dan operasi dasar, yaitu kumpulan fungsi atau prosedur yang dapat digunakan untuk memanipulasi data dalam tipe tersebut. ADT dapat dibedakan menjadi beberapa jenis berdasarkan sifat datanya, seperti ADT homogen yang hanya menampung variabel dengan tipe data yang sama, dan ADT heterogen yang dapat menampung variabel dengan berbagai tipe data. Sebagai contoh implementasi, dalam bahasa C terdapat struktur Jam yang mendefinisikan waktu dan dilengkapi dengan fungsi MakeJam serta TulisJam untuk membuat dan mendampilkan waktu. Secara keseluruhan, ADT merupakan konsep fundamental dalam pemrograman yang membantu dalam pengembangan perangkat lunak yang lebih terstruktur dan mudah dikelola, memungkinkan pemrogram menciptakan solusi yang lebih efisien dan modular.

3. **Guided**

1.

```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;
}
```

Dari input di atas akan mendapatkan output:

```
input nim= 2311104063
input nilai = 90
input nilai = 85
rata-rata = 87
```

Penjelasan kode program:

Program ini menggunakan struktur mahasiswa untuk menyimpan data mahasiswa, yang dideklarasikan melalui struct mahasiswa sebagai cetak biru. Data mahasiswa diisi melalui fungsi inputMhs yang meminta pengguna untuk memasukkan informasi yang diperlukan. Selanjutnya, fungsi rata2 menghitung rata-rata nilai dari data yang dimasukkan, dan hasil perhitungan ini ditampilkan di layar dalam bentuk float. Seluruh alur program dijalankan dalam fungsi utama main, yang mengoordinasikan input, perhitungan rata-rata, dan output hasil ke layar.

Fungsi inputMhs(mahasiswa &m):

- Fungsi ini digunakan untuk menerima input dari user berupa NIM dan dua nilai.
- Simbol & menandakan bahwa argumen yang diterima oleh fungsi ini adalah referensi dari variabel mahasiswa, sehingga perubahan yang dilakukan dalam fungsi ini juga memengaruhi variabel aslinya.
- Fungsi ini meminta pengguna untuk memasukkan NIM, nilai pertama, dan nilai kedua.

Fungsi rata2(mahasiswa m):

- Fungsi ini digunakan untuk menghitung rata-rata dari dua nilai yang dimiliki oleh mahasiswa.
- Mengembalikan hasil perhitungan rata-rata dari nilail dan nilai2.

Fungsi main():

- Fungsi lama program.
- Membuat sebuah variabel mhs bertipe mahasiswa.
- Memanggil fungsi inputMhs() untuk meminta input data mahasiswa.
- Memanggil fungsi rata2() untuk menghitung dan menampilkan rata-rata dua nilai mahasiswa.

4. Unguided

1. Kode program:

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

using namespace std;

const int MAX_MAHASISWA = 10;

struct Mahasiswa {
    string nama;
    string nim;
    double uts, uas, tugas, nilaiAkhir;
};

// Fungsi untuk menghitung nilai akhir
double hitungNilaiAkhir(double uts, double uas, double tugas) {
    return 0.3 * uts + 0.4 * uas + 0.3 * tugas;
}

int main() {
    Mahasiswa mahasiswa[MAX_MAHASISWA];
    int jumlahMahasiswa;

    cout << "Masukkan jumlah mahasiswa (maksimal " << MAX_MAHASISWA << "): ";
    cin >> jumlahMahasiswa;

    for (int i = 0; i < jumlahMahasiswa; i++) {
        cout << "\nData Mahasiswa ke-" << i+1 << endl;
        cout << "Nama: ";
        cin.ignore(); // Membersihkan buffer input
        getline(cin, mahasiswa[i].nama);
        cout << "NIM: ";
        cin >> mahasiswa[i].nim;
        cout << "Nilai UTS: ";
        cin >> mahasiswa[i].uts;
        cout << "Nilai UAS: ";
        cin >> mahasiswa[i].uas;
        cout << "Nilai Tugas: ";
        cin >> mahasiswa[i].tugas;

        // Hitung nilai akhir
        mahasiswa[i].nilaiAkhir = hitungNilaiAkhir(mahasiswa[i].uts, mahasiswa[i].uas, mahasiswa[i].tugas);
    }

    cout << "\nData Mahasiswa:" << endl;
    for (int i = 0; i < jumlahMahasiswa; i++) {
        cout << "Nama: " << mahasiswa[i].nama << endl;
        cout << "NIM: " << mahasiswa[i].nim << endl;
        cout << "Nilai Akhir: " << mahasiswa[i].nilaiAkhir << endl << endl;
    }

    return 0;
}
```

Outputnya:

```
Masukkan jumlah mahasiswa (maksimal 10): 2

Data Mahasiswa ke-1
Nama: Daniel
NIM: 2311104063
Nilai UTS: 90
Nilai UAS: 95
Nilai Tugas: 80

Data Mahasiswa ke-2
Nama: Shafiq
NIM: 2311103043
Nilai UTS: 89
Nilai UAS: 90
Nilai Tugas: 95

Data Mahasiswa:
Nama: Daniel
NIM: 2311104063
Nilai Akhir: 89

Nama: Shafiq
NIM: 2311103043
Nilai Akhir: 91.2
```

2. **1. Pelajaran.h**

```h
#ifndef pelajaran_h
#define peljaran_h

#include <string>
using namespace std;

struct pelajaran {
    string namaMapel;
    string kodeMapel;
};

// Fungsi untuk membuat pelajaran
pelajaran create_pelajaran(string namapel, string kodepel);

// Prosedur untuk menampilkan pelajaran
void tampil_pelajaran(pelajaran pel);

#endif
```

2.pelajaran.cpp

```cpp
#include "pelajaran.h"
#include <iostream>

using namespace std;

// Implementasi fungsi create_pelajaran
pelajaran create_pelajaran(string namapel, string kodepel) {
    pelajaran pel;  // Membuat objek pelajaran
    pel.namaMapel = namapel;  // Mengisi namaMapel
    pel.kodeMapel = kodepel;  // Mengisi kodeMapel
    return pel;  // Mengembalikan objek pelajaran
}

// Implementasi prosedur tampil_pelajaran
void tampil_pelajaran(pelajaran pel) {
    cout << "nama pelajaran : " << pel.namaMapel << endl;
    cout << "nilai : " << pel.kodeMapel << endl;
}
```

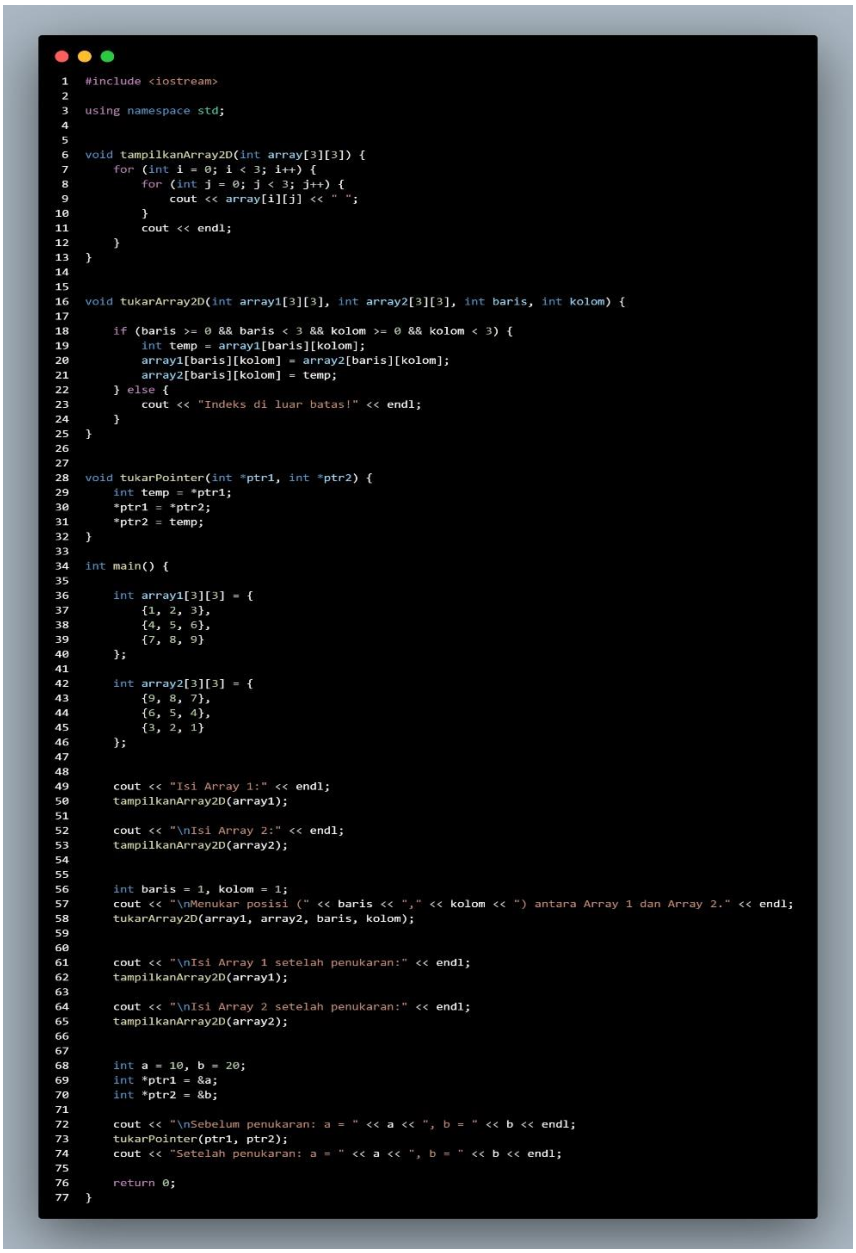
3.main.cpp

```cpp
include <iostream>
#include "pelajaran.h"
#include "pelajaran.cpp"

using namespace std;

int main() {
    string namapel = "Struktur Data";
    string kodepel = "STD";

    pelajaran pel = create_pelajaran(namapel, kodepel);  // Panggil fungsi create_pelajaran
    tampil_pelajaran(pel);  // Tampilkan hasil

    return 0;
}
```

Output dari program:

```
Nama Pelajaran : Struktur Data
Nilai : STD
```

3. **Kode Program**



```cpp
#include <iostream>

using namespace std;


void tampilkanArray2D(int array[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << array[i][j] << " ";
        }
        cout << endl;
    }
}


void tukarArray2D(int array1[3][3], int array2[3][3], int baris, int kolom) {

    if (baris >= 0 && baris < 3 && kolom >= 0 && kolom < 3) {
        int temp = array1[baris][kolom];
        array1[baris][kolom] = array2[baris][kolom];
        array2[baris][kolom] = temp;
    } else {
        cout << "Indeks di luar batas!" << endl;
    }
}


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

int main() {

    int array1[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    int array2[3][3] = {
        {9, 8, 7},
        {6, 5, 4},
        {3, 2, 1}
    };


    cout << "Isi Array 1:" << endl;
    tampilkanArray2D(array1);

    cout << "\nIsi Array 2:" << endl;
    tampilkanArray2D(array2);


    int baris = 1, kolom = 1;
    cout << "\nMenukar posisi (" << baris << "," << kolom << ") antara Array 1 dan Array 2." << endl;
    tukarArray2D(array1, array2, baris, kolom);


    cout << "\nIsi Array 1 setelah penukaran:" << endl;
    tampilkanArray2D(array1);

    cout << "\nIsi Array 2 setelah penukaran:" << endl;
    tampilkanArray2D(array2);


    int a = 10, b = 20;
    int *ptr1 = &a;
    int *ptr2 = &b;

    cout << "\nSebelum penukaran: a = " << a << ", b = " << b << endl;
    tukarPointer(ptr1, ptr2);
    cout << "Setelah penukaran: a = " << a << ", b = " << b << endl;

    return 0;
}
```

Output:

```
Isi Array 1:
1 2 3
4 5 6
7 8 9

Isi Array 2:
9 8 7
6 5 4
3 2 1

Menukar posisi (1,1) antara Array 1 dan Array 2.

Isi Array 1 setelah penukaran:
1 2 3
4 5 6
7 8 9

Isi Array 2 setelah penukaran:
9 8 7
6 5 4
3 2 1

Sebelum penukaran: a = 10, b = 20
Setelah penukaran: a = 20, b = 10
```

5. **Kesimpulan**

   Dari praktikum ini mempelajari Abstract Data Type (ADT) memisahkan struktur data dari operasinya, memungkinkan pengembangan program yang lebih modular, terstruktur, dan mudah dikelola. Dengan ADT, pemrogram dapat menyembunyikan detail implementasi, sehingga perubahan internal tidak memengaruhi bagian lain dari program. Penerapan ADT, seperti pada contoh struktur mahasiswa, memudahkan pengorganisasian data dan memisahkan fungsi untuk input, perhitungan, serta output. ADT mendukung efisiensi dan fleksibilitas dalam pengembangan perangkat lunak.