LAPORAN PRAKTIKUM
MODUL 3
ABSTRACT DATA TYPE (ADT)

Disusun Oleh:

Satria Putra Dharma Prayudha - 21104036

SE07-02

Dosen :

Wahyu Andi Saputra, S.Pd., M.Eng

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

A. Tujuan

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

B. Landasan Teori

Landasan teori ini berdasarkan pada modul pembelajaran praktikum struktur data kali ini

2.1 Abstract Data Type (ADT)

*Abstract Data Type (*ADT) merupakan konsep penting dalam pemrograman yang terdiri dari tipe data dan sekumpulan operasi dasar yang dapat dilakukan terhadap tipe tersebut. ADT mendefinisikan tipe secara abstrak tanpa mengungkapkan implementasi internalnya, memungkinkan pemrogram untuk menggunakan tipe data tersebut tanpa perlu tahu bagaimana cara kerjanya di balik layar. Dalam implementasinya, ADT terdiri dari beberapa elemen:

- Konstruktor: Fungsi yang digunakan untuk membentuk nilai dari tipe data.
- Selector: Digunakan untuk mengakses komponen tipe data.
- Prosedur Pengubah: Prosedur yang bertugas mengubah nilai komponen.
- Validator: Memeriksa apakah tipe data valid sesuai dengan batasan yang ditetapkan.

ADT dapat diimplementasikan dengan memisahkan spesifikasi dan implementasinya dalam dua modul, yaitu:

- Modul Spesifikasi: Berupa *file header* (.h) yang mendefinisikan tipe data dan operasinya.
- Modul Implementasi: Berupa *file .cpp* yang berisi realisasi dari operasi yang didefinisikan.

Dalam ADT yang dipelajari, kita menerapkan konsep ini untuk tipe data seperti mahasiswa dan pelajaran, yang menunjukkan bagaimana data disusun dalam bentuk struktur, dan operasi-operasi terkait dapat dilakukan untuk

memanipulasi data tersebut.

C. Guided

a. *ADT*

Code :

```
#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 nilai mahasiswa adalah: " <<
rata2(mhs);
    return 0;
}

void inputMhs(mahasiswa &m){
    cout << "Masukkan NIM: ";
    cin >> (m).nim;
    cout << "Masukkan nilai 1: ";
    cin >> (m).nilai1;
    cout << "Masukkan nilai 2: ";
    cin >> (m).nilai2;
}

float rata2 (mahasiswa m){
    return (m.nilai1 + m.nilai2) / 2.0;
}
```

Output :

```
PS D:\Kuliah\Struktur Data\Github\03_Abstract_Data_Type\Unguided>
$?) { .\tempCodeRunnerFile }
Masukkan NIM: 21104036
Masukkan nilai 1: 100
Masukkan nilai 2: 80
Rata-rata nilai mahasiswa adalah: 90
```

Penjelasan : Kode di atas merupakan implementasi sederhana dari Abstract Data Type (ADT) menggunakan contoh yaitu mahasiswa. Struktur data mahasiswa digunakan untuk menyimpan data NIM dan dua nilai ujian. Program ini memiliki dua fungsi utama, yaitu inputMhs() yang bertugas menerima input dari pengguna untuk mengisi data NIM dan nilai mahasiswa, serta rata2() yang menghitung rata-rata dari dua nilai yang dimasukkan. Program ini mencetak hasil perhitungan rata-rata nilai mahasiswa ke layar, menunjukkan konsep dasar ADT di mana operasi (*input* dan perhitungan) dilakukan pada struktur data mahasiswa.

D. Unguided

a. Program Penyimpanan Data Mahasiswa

Code:

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

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

float hitungNilaiAkhir(float uts, float uas, float
tugas) {
    return (0.3 * uts) + (0.4 * uas) + (0.3 * tugas);
}

int main() {
    Mahasiswa mhs[10];
    int jumlahMahasiswa;

    cout << "Masukkan jumlah mahasiswa (maks 10): ";
    cin >> jumlahMahasiswa;

    for (int i = 0; i < jumlahMahasiswa; i++) {
        cout << "Mahasiswa " << i + 1 << endl;
        cout << "Nama: ";
        cin >> mhs[i].nama;
        cout << "NIM: ";
        cin >> mhs[i].nim;
        cout << "Nilai UTS: ";
        cin >> mhs[i].uts;
        cout << "Nilai UAS: ";
        cin >> mhs[i].uas;
        cout << "Nilai Tugas: ";
        cin >> mhs[i].tugas;

        mhs[i].nilaiAkhir =
hitungNilaiAkhir(mhs[i].uts, mhs[i].uas, mhs[i].tugas);
        cout << "Nilai Akhir: " << mhs[i].nilaiAkhir <<
endl << endl;
    }

    return 0;
}
```

Output:

```
PS D:\Kuliah\Struktur Data\Github\03_Abstract_Data_Type\Guided> cd
} ; if ($?) { .\Unguided_01_NilaiAkhir }
Masukkan jumlah mahasiswa (maks 10): 3
Mahasiswa 1
Nama: Satria
NIM: 21104036
Nilai UTS: 100
Nilai UAS: 95
Nilai Tugas: 89
Nilai Akhir: 94.7

Mahasiswa 2
Nama: Yudha
NIM: 21104099
Nilai UTS: 90
Nilai UAS: 80
Nilai Tugas: 85
Nilai Akhir: 84.5

Mahasiswa 3
Nama: Kuro
NIM: 21104056
Nilai UTS: 90
Nilai UAS: 100
Nilai Tugas: 95
Nilai Akhir: 95.5
```

Penjelasan: Program ini bertujuan untuk menyimpan data mahasiswa (maksimal 10 mahasiswa) ke dalam *array* struktur data yang berisi *field* seperti nama, NIM, nilai UTS, nilai UAS, dan nilai tugas. Nilai akhir setiap mahasiswa dihitung menggunakan formula tertentu, yaitu 30% dari nilai UTS, 40% dari nilai UAS, dan 30% dari nilai tugas. Hasil perhitungan nilai akhir ini kemudian ditampilkan untuk setiap mahasiswa. Program ini menunjukkan penerapan fungsi untuk perhitungan nilai dan penggunaan *array* dalam menyimpan data mahasiswa.

b. **ADT Pelajaran**

Code:

a. Pelajaran.h

```h
#ifndef PELAJARAN_H_INCLUDED
#define PELAJARAN_H_INCLUDED
#include <string>
using namespace std;

struct Pelajaran {
    string namaMapel;
    string kodeMapel;
};

Pelajaran create_pelajaran(string namaMapel, string
kodeMapel);
void tampil_pelajaran(Pelajaran pel);

#endif // PELAJARAN_H_INCLUDED
```

b. Pelajaran.cpp

```cpp
#include <iostream>
#include "pelajaran.h"
using namespace std;

Pelajaran create_pelajaran(string namaMapel, string
kodeMapel) {
    Pelajaran pel;
    pel.namaMapel = namaMapel;
    pel.kodeMapel = kodeMapel;
    return pel;
}

void tampil_pelajaran(Pelajaran pel) {
    cout << "Nama Mata Pelajaran: " << pel.namaMapel <<
endl;
    cout << "Kode Mata Pelajaran: " << pel.kodeMapel <<
endl;
}
```

c. Main.cpp

```cpp
#include <iostream>
#include "pelajaran.cpp"
using namespace std;

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

    Pelajaran pel = create_pelajaran(namapel, kodepel);
    tampil_pelajaran(pel);

    return 0;
}
```

Output:

```
PS D:\Kuliah\Struktur Data\Github\03_Abstract_Data_Type\Unguided> cd
unnerFile } ; if ($?) { .\tempCodeRunnerFile }
Nama Mata Pelajaran: Struktur Data
Kode Mata Pelajaran: STD
```

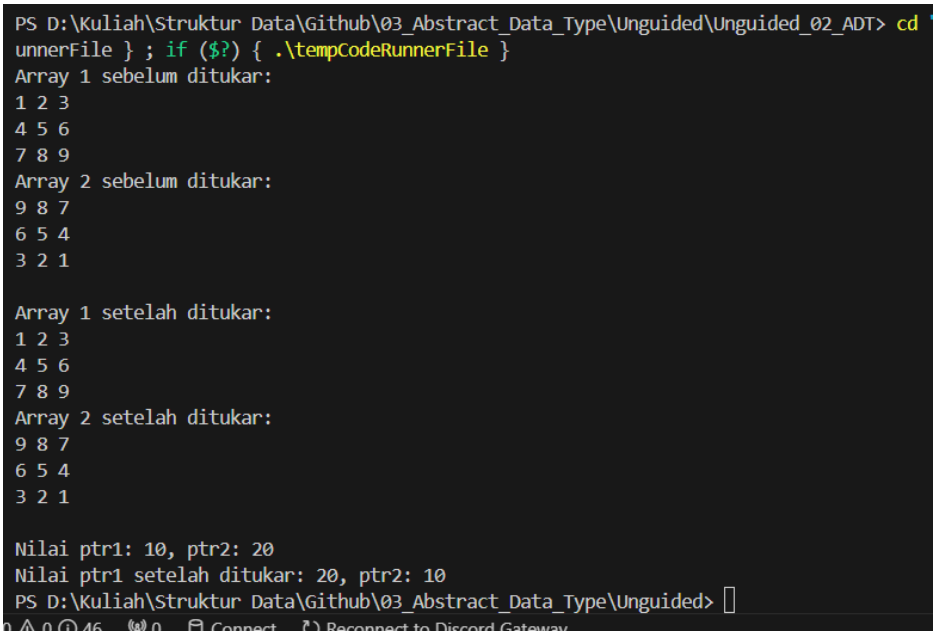
Penjelasan: Pada kode diatas berisikan *Abstract Data Type* (ADT) Pelajaran yang terdiri dari dua atribut utama: namaMapel (nama mata pelajaran) dan kodeMapel (kode mata pelajaran). ADT ini diimplementasikan dalam dua file terpisah, yaitu pelajaran.h untuk spesifikasi tipe dan deklarasi fungsinya, serta pelajaran.cpp untuk realisasi atau implementasi fungsi. Fungsi yang digunakan adalah create_pelajaran(), yang bertugas membuat objek pelajaran dengan memasukkan nama dan kode pelajaran, dan tampil_pelajaran(), yang bertugas menampilkan informasi pelajaran tersebut di layar. Setelah ADT diimplementasikan, program utama (main.cpp) akan memanggil fungsi-fungsi ini untuk membuat dan menampilkan data pelajaran.

c. **Pointer dan Array 2D**

Code:

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

void tampilArray(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 tukarArray(int arr1[3][3], int arr2[3][3], int baris, int kolom) {
    int temp = arr1[baris][kolom];
    arr1[baris][kolom] = arr2[baris][kolom];
    arr2[baris][kolom] = temp;
}

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}};

    int *ptr1, *ptr2;
    int var1 = 10, var2 = 20;

    ptr1 = &var1;
    ptr2 = &var2;

    // Menampilkan array sebelum ditukar
    cout << "Array 1 sebelum ditukar:" << endl;
    tampilArray(array1);

    cout << "Array 2 sebelum ditukar:" << endl;
    tampilArray(array2);

    // Menukar elemen pada posisi tertentu (misal posisi [1][1])
    tukarArray(array1, array2, 1, 1);

    // Menampilkan array setelah ditukar
    cout << "\nArray 1 setelah ditukar:" << endl;
    tampilArray(array1);

    cout << "Array 2 setelah ditukar:" << endl;
    tampilArray(array2);

    // Menampilkan nilai pointer sebelum ditukar
    cout << "\nNilai ptr1: " << *ptr1 << ", ptr2: " << *ptr2 << endl;

    // Menukar nilai yang ditunjuk oleh pointer
    tukarPointer(ptr1, ptr2);

    // Menampilkan nilai pointer setelah ditukar
    cout << "Nilai ptr1 setelah ditukar: " << *ptr1 << ", ptr2: " << *ptr2
<< endl;

    return 0;
}
```

Output:



Penjelasan: Kode di atas merupakan program yang mengimplementasikan dua *array* 2D berukuran 3x3 dan menunjukkan operasi penukaran elemen antara *array* tersebut serta penukaran nilai variabel melalui *pointer*. Fungsi tampilArray() digunakan untuk menampilkan isi dari *array* 2D dengan iterasi melalui setiap elemen, sehingga memudahkan visualisasi data sebelum dan setelah penukaran. Fungsi tukarArray() bertanggung jawab untuk menukar elemen pada posisi tertentu antara dua *array*, dalam hal ini, elemen pada posisi [1][1]. Fungsi tukarPointer() memungkinkan penukaran nilai yang ditunjuk oleh dua *pointer*, yang menunjukkan bagaimana pointer dapat digunakan untuk memanipulasi nilai variabel di memori. Dalam fungsi main(), dua array (array1 dan array2) diinisialisasi dengan nilai yang berbeda. Program pertama-tama menampilkan isi kedua *array* sebelum penukaran, kemudian melakukan penukaran elemen di posisi tertentu dan menampilkan hasilnya. Selanjutnya, nilai dua variabel (var1 dan var2) ditunjukkan sebelum dan sesudah ditukar, menunjukkan fleksibilitas *pointer* dalam memanipulasi nilai di memori.

E. Kesimpulan

Dalam praktikum ini, kita telah mempelajari dan menerapkan berbagai konsep penting dalam pemrograman menggunakan bahasa C++, khususnya yang berkaitan dengan *Abstract Data Type* (ADT), *array*, dan *pointer*. Implementasi ADT, seperti pada data mahasiswa dan pelajaran, menunjukkan bagaimana kita dapat mendefinisikan struktur data yang kompleks dan memisahkan spesifikasi dari implementasi, sehingga menghasilkan kode yang lebih terstruktur dan mudah dipahami.

Penggunaan *array* satu dimensi dan dua dimensi dalam program ini membantu kita memahami cara menyimpan dan mengakses kumpulan data yang terorganisir. Latihan yang dilakukan, seperti perhitungan rata-rata nilai mahasiswa dan penukaran elemen dalam array 2D, memberikan wawasan mendalam tentang manipulasi data menggunakan array. Selain itu, pengenalan pointer memberikan pengetahuan tentang interaksi program dengan memori, memungkinkan kita untuk melakukan manipulasi data secara lebih efisien.

Fungsi dan prosedur yang diterapkan dalam praktikum ini berkontribusi pada pengembangan kode yang modular, meningkatkan kemampuan pemrograman terstruktur. Secara keseluruhan, modul ini memberikan dasar yang kuat untuk pemrograman lanjutan dalam bahasa C++, sekaligus meningkatkan pemahaman kita tentang pengelolaan data dan penggunaan teknik pemrograman yang baik.