

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



Disusun Oleh:
Aji Prasetyo Nugroho - 2211104049
S1SE-07-2

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

A. Tujuan

1. Mahasiswa mampu memahami definisi dan konsep dasar Abstract Data Type (ADT) dalam pemrograman, termasuk pemisahan antara representasi data dan operasinya.
2. Mahasiswa mampu mengimplementasikan ADT pada program menggunakan bahasa pemrograman, dengan memisahkan deklarasi dan implementasi ADT melalui file header dan file implementasi.
3. Mahasiswa mampu membangun program yang modular dengan memanfaatkan ADT, sehingga program lebih terstruktur, mudah dipelihara, dan dikembangkan.
4. Mahasiswa mampu memanfaatkan ADT untuk meningkatkan abstraksi dalam program dengan memisahkan antara penggunaan data dan detail implementasi dari struktur data tersebut.
5. Mahasiswa memahami pentingnya enkapsulasi (encapsulation) dalam ADT untuk membatasi akses langsung ke elemen-elemen data, dan hanya memperbolehkan interaksi melalui fungsi/prosedur yang telah didefinisikan.

B. Landasan Teori

Abstract Data Type (ADT) adalah sebuah model matematis untuk tipe data yang ditentukan oleh perilaku dari data tersebut (operasi-operasi yang dapat dilakukan terhadap data) tanpa memikirkan bagaimana data tersebut diimplementasikan. Dalam ADT, kita hanya perlu memahami apa yang dapat dilakukan terhadap data (interface), tanpa memedulikan bagaimana data tersebut diimplementasikan (implementasi detail).

C. Guided

a) Abstract Data Type (ADT)

Abstract Data Type (ADT) adalah tipe data dan sekumpulan operasi dasar yang bisa dilakukan terhadap tipe tersebut, dilengkapi dengan invarian dan aksioma yang berlaku. ADT didefinisikan secara statik dan bisa mengandung ADT lain, misalnya ADT `waktu` terdiri dari ADT `jam` dan `tanggal`, atau ADT `garis` yang terdiri dari dua ADT `titik (POINT)`. Dalam C, ADT diimplementasikan dengan `struct` dan operasi dasar diterjemahkan menjadi fungsi atau prosedur. Primitif ADT mencakup konstruktor (membentuk nilai tipe), selektor (mengakses komponen), prosedur pengubah nilai, validator (memeriksa kesesuaian nilai), destruktur (menghapus objek dan memori), serta operasi baca/tulis, operator

relasional, aritmatika, dan konversi antar tipe. Implementasi ADT melibatkan modul spesifikasi (file `.h`) dan realisasi (file `.cpp` atau `.c`), serta satu modul driver untuk menguji ADT.

Source Code :

```
#include<iostream>

using namespace std;

struct mahasiswa{
    char nim[20];
    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 :

```
Input nim = 10
input nilai = 100
Input nilai = 100
rata-rata = 100
PS D:\Praktikum SD> █
```

D. Unguided

1. Buat program yang dapat menyimpan data mahasiswa (max. 10) ke dalam sebuah *array* dengan field nama, nim, uts, uas, tugas, dan nilai akhir. Nilai akhir diperoleh dari FUNGSI dengan rumus $0.3 \times \text{uts} + 0.4 \times \text{uas} + 0.3 \times \text{tugas}$.

Source code :

```
#include <iostream>
#include <iomanip>
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() {
    const int MAKS_MAHASISWA = 10;
    Mahasiswa mahasiswa[MAKS_MAHASISWA];
    int jumlahMahasiswa;

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

    if (jumlahMahasiswa > MAKS_MAHASISWA) {
        cout << "Jumlah mahasiswa melebihi batas maksimal (10). Program dihentikan." << endl;
        return 1;
    }

    for (int i = 0; i < jumlahMahasiswa; i++) {
        cout << "\nData mahasiswa ke-" << (i + 1) << endl;
        cout << "Nama      : ";
        cin.ignore();
        getline(cin, mahasiswa[i].nama);
        cout << "NIM       : ";
        getline(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;

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

    cout << "\nData Mahasiswa: \n";
    cout << "===== \n";
    cout << left << setw(5) << "No" << setw(20) << "Nama" << setw(15) << "NIM" << setw(10) << "UTS" << setw(10) <<
    "UAS" << setw(10) << "Tugas" << setw(10) << "Nilai Akhir" << endl;
    cout << "===== \n";
    for (int i = 0; i < jumlahMahasiswa; i++) {
        cout << left << setw(5) << (i + 1)
            << setw(20) << mahasiswa[i].nama
            << setw(15) << mahasiswa[i].nim
            << setw(10) << mahasiswa[i].uts
            << setw(10) << mahasiswa[i].uas
            << setw(10) << mahasiswa[i].tugas
            << setw(10) << fixed << setprecision(2) << mahasiswa[i].nilaiAkhir << endl;
    }

    return 0;
}
```

Output :

```
Masukkan jumlah mahasiswa (maksimal 10): 3

Data mahasiswa ke-1
Nama      : aji
NIM       : 49
Nilai UTS : 100
Nilai UAS : 100
Nilai Tugas: 100

Data mahasiswa ke-2
Nama      : yogi
NIM       : 61
Nilai UTS : 100
Nilai UAS : 100
Nilai Tugas: 100

Data mahasiswa ke-3
Nama      : rio
NIM       : 31
Nilai UTS : 100
Nilai UAS : 100
Nilai Tugas: 100

Data Mahasiswa:
=====
No  Nama      NIM      UTS      UAS      Tugas      Nilai Akhir
=====
1   aji       49       100      100      100      100.00
2   yogi      61       100.00   100.00   100.00   100.00
3   rio       31       100.00   100.00   100.00   100.00
PS D:\Praktikum STD>
```

Program di atas digunakan untuk mengelola data mahasiswa dengan kapasitas maksimum 10 mahasiswa. Setiap mahasiswa memiliki atribut nama, nim, uts, uas, tugas, dan nilaiAkhir yang disimpan dalam array mahasiswa berukuran 10. Program meminta pengguna untuk memasukkan jumlah mahasiswa dan datanya (nama, NIM, nilai UTS, nilai UAS, dan nilai tugas) satu per satu. Nilai akhir dihitung menggunakan fungsi hitungNilaiAkhir dengan rumus $0.3 * uts + 0.4 * uas + 0.3 * tugas$ dan disimpan dalam nilaiAkhir tiap mahasiswa. Setelah data dimasukkan, program menampilkan tabel yang berisi informasi mahasiswa beserta nilai akhirnya menggunakan setw untuk format output yang rapi. Jika jumlah mahasiswa yang dimasukkan melebihi batas maksimum, program akan memberikan pesan kesalahan dan berhenti.

2. Buatlah ADT pelajaran sebagai berikut di dalam file "pelajaran.h":

```
type pelajaran <
    namaMapel : string
    kodeMapel : string
>
fungsi create_pelajaran( namapel : string, kodepel : string ) →
    pelajaran
prosedur tampil_pelajaran( pel : pelajaran )
```

Buatlah implementasi ADT pelajaran pada file "pelajaran.cpp"

Cobalah hasil implementasi ADT pada file "main.cpp"

```
using namespace std;
int main(){
    string namapel = "Struktur Data";
    string kodepel = "STD";
    pelajaran pel = create_pelajaran(namapel, kodepel);
    tampil_pelajaran(pel);

    return 0;
}
```

Gambar 3-1 Main.cpp pelajaran

Contoh output hasil:

```
nama pelajaran : Struktur Data
nilai : STD
```

Source Code soal2pelajaran.h :

```
#ifndef PELAJARAN_H
#define PELAJARAN_H

#include <string>
using namespace std;


struct pelajaran {
    string namaMapel;
    string kodeMapel;
};

pelajaran create_pelajaran(string namapel, string kodepel);

void tampil_pelajaran(pelajaran pel);

#endif
```

Source Code soal2pelajaran.cpp :




```
#include "soal2pelajaran.h"
#include <iostream>
using namespace std;

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

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

Source Code soal2main.cpp :



```
#include <iostream>
#include "soal2pelajaran.h"
#include "soal2pelajaran.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 :

```
nama pelajaran : Struktur Data  
nilai : STD  
PS D:\Praktikum STD>
```

Program ini terdiri dari tiga file: `soal2main.cpp`, `soal2pelajaran.cpp`, dan `soal2pelajaran.h`, yang digunakan untuk mendefinisikan dan mengimplementasikan ADT (Abstract Data Type) `pelajaran`. File `soal2pelajaran.h` mendefinisikan struktur `pelajaran` dengan dua atribut `namaMapel` dan `kodeMapel`, serta mendeklarasikan dua fungsi `create_pelajaran` (untuk membuat objek `pelajaran`) dan `tampil_pelajaran` (untuk menampilkan informasi `pelajaran`). File `soal2pelajaran.cpp` berisi implementasi dari fungsi-fungsi tersebut: `create_pelajaran` menerima nama mata pelajaran dan kode pelajaran sebagai parameter, lalu mengembalikan objek `pelajaran` yang berisi data tersebut, sedangkan `tampil_pelajaran` mencetak nama dan kode pelajaran ke layar. File `soal2main.cpp` adalah file utama yang menguji ADT `pelajaran` dengan membuat objek `pelajaran` bernama "Struktur Data" dengan kode "STD" menggunakan `create_pelajaran` dan kemudian menampilkannya menggunakan `tampil_pelajaran`. Program ini menunjukkan bagaimana membangun ADT yang modular dan terpisah, sehingga memudahkan pengembangan dan pemeliharaan program.

3. Buatlah program dengan ketentuan :

- 2 buah array 2D integer berukuran 3x3 dan 2 buah pointer integer
- fungsi/prosedur yang menampilkan isi sebuah array integer 2D
- fungsi/prosedur yang akan menukarkan isi dari 2 array integer 2D pada posisi tertentu
- fungsi/prosedur yang akan menukarkan isi dari variabel yang ditunjuk oleh 2 buah pointer

Source Code :

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

void tampilArray(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 tukarArray(int array1[3][3], int array2[3][3], int baris, int kolom) {
    if (baris < 3 && kolom < 3) {
        int temp = array1[baris][kolom];
        array1[baris][kolom] = array2[baris][kolom];
        array2[baris][kolom] = temp;
    } else {
        cout << "Indeks baris atau kolom di luar batas!" << endl;
    }
}

void tukarPointer(int *p1, int *p2) {
    int temp = *p1;
    *p1 = *p2;
    *p2 = 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 a = 10, b = 20;
    int *p1 = &a;
    int *p2 = &b;

    cout << "Array 1 sebelum pertukaran: " << endl;
    tampilArray(array1);
    cout << "Array 2 sebelum pertukaran: " << endl;
    tampilArray(array2);

    tukarArray(array1, array2, 1, 1);

    cout << "\nArray 1 setelah pertukaran pada posisi (1,1): " << endl;
    tampilArray(array1);
    cout << "Array 2 setelah pertukaran pada posisi (1,1): " << endl;
    tampilArray(array2);

    cout << "\nNilai p1 dan p2 sebelum pertukaran: " << endl;
    cout << "p1: " << *p1 << " p2: " << *p2 << endl;

    tukarPointer(p1, p2);

    cout << "\nNilai p1 dan p2 setelah pertukaran: " << endl;
    cout << "p1: " << *p1 << " p2: " << *p2 << endl;

    return 0;
}
```

Output :

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

Array 1 setelah pertukaran pada posisi (1,1):
1 2 3
4 5 6
7 8 9
Array 2 setelah pertukaran pada posisi (1,1):
9 8 7
6 5 4
6 5 4
3 2 1

Nilai p1 dan p2 sebelum pertukaran:
p1: 10 p2: 20

Nilai p1 dan p2 setelah pertukaran:
p1: 20 p2: 10
PS D:\Praktikum STD>
```

Program di atas mendemonstrasikan penggunaan array 2D dan pointer dengan beberapa operasi pertukaran nilai. Program ini memiliki dua array 2D (`array1` dan `array2`) berukuran 3x3 dan dua pointer (`p1` dan `p2`) yang masing-masing menunjuk ke variabel `a` dan `b`. Program ini menyediakan tiga fungsi: `tampilArray` untuk menampilkan isi array 2D, `tukarArray` untuk menukar elemen dari dua array 2D pada posisi tertentu, dan `tukarPointer` untuk menukar nilai yang ditunjuk oleh dua pointer. Dalam `main()`, program pertama-tama menampilkan `array1` dan `array2` sebelum pertukaran. Kemudian, elemen pada posisi `(1,1)` dari `array1` dan `array2` ditukar, dan hasilnya ditampilkan kembali. Selanjutnya, program menampilkan nilai `p1` dan `p2` sebelum pertukaran, menukar nilai yang ditunjuk oleh `p1` dan `p2`, dan menampilkan hasil akhirnya. Program ini menunjukkan cara kerja pertukaran nilai pada array dan pointer, serta pentingnya indeks dalam operasi pada array 2D.