LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL III
ABSTRACT DATA TYPE (ADT)

Disusun Oleh :
NAMA : Ridha Akifah
NIM : 103112400132

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Abstract Data Type (ADT) adalah definisi statik yang mengkombinasikan sebuah TYPE dan sekumpulan PRIMITIF (operasi dasar) terhadap *type* tersebut. Konsep intinya adalah memisahkan spesifikasi (*apa* yang dilakukan) dari implementasi (*bagaimana* dilakukan). Primitif pada ADT dikelompokkan menjadi beberapa jenis, seperti Konstruktor/Kreator untuk membentuk nilai tipe, Selector untuk mengakses komponen, Prosedur Pengubah Nilai, dan Destruktor/Dealokator. Implementasi ADT biasanya dipisahkan menjadi dua modul utama: *file* Header (.h) yang berisi definisi tipe (struct dalam C) dan spesifikasi fungsi/prosedur, serta *file* Body/Realisasi (.c atau .cpp) yang berisi kode program dari primitif tersebut. Pemisahan ini menjaga prinsip modularitas, di mana program utama (*driver*) hanya berinteraksi dengan *file header* untuk menggunakan ADT.

B. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1

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

void inputMhs(mahasiswa &m)
{
    cout << "input nama = ";
    cin >> (m).nim;
    cout << "input nilai = ";
    cin >> (m).nilai1;
    cout << "input nilai2 = ";
    cin >> (m).nilai2;
}
float rata2(mahasiswa m)
{
```

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

Deskripsi:

File ini berperan sebagai Spesifikasi dari Abstract Data Type (ADT) Mahasiswa. Bagian ini mendefinisikan TYPE mahasiswa menggunakan struct yang berisi komponen-komponen data yang relevan (misalnya: nama, NIM, dan nilai). Selain itu, *file* ini memuat deklarasi (prototipe) dari semua primitif yang tersedia untuk ADT, seperti Konstruktor (cth: create_mahasiswa) dan primitif I/O (cth: tampil_mahasiswa). Secara fungsi, *file* ini menentukan *interface* publik ADT.

Guided 2

```
#ifndef MAHASISWA_H_INCLUDED
#define MAHASISWA_H_INCLUDED
struct mahasiswa
{
    char nim[10];
    int nilai1, nilai2;
};
void inputMhs(mahasiswa &m);
float rata2 (mahasiswa m);
#endif
```

Deskripsi:

File ini menyediakan Realisasi atau Implementasi kode penuh dari setiap primitif yang dideklarasikan di mahasiswa.h. Fungsi Konstruktor diimplementasikan untuk mengalokasikan dan menginisialisasi objek mahasiswa dengan nilai *input*. Sementara itu, prosedur I/O dan primitif lain (seperti Selector atau Mutator, jika ada) diimplementasikan untuk melakukan operasi yang sesungguhnya pada data mahasiswa. *File* ini menyembunyikan detail *bagaimana* data dikelola dari program utama.

Guided 3

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

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

Screenshoot Output

```
PS D:\C++> cd "d:\C++\Modul 3\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
input nama = dioo
input nilai = 101
input nilai2 = 100
rata-rata = 100.5
```

Deskripsi:

File ini berfungsi sebagai Program Utama (Driver) yang bertindak sebagai pengguna ADT Mahasiswa. Dengan hanya menyertakan (include) mahasiswa.h, *driver* ini memanggil Konstruktor untuk membuat objek mhs baru, dan kemudian menggunakan primitif ADT lainnya, seperti prosedur I/O, untuk memproses atau menampilkan data objek tersebut. *File* ini menunjukkan demonstrasi kerja dari ADT Mahasiswa secara keseluruhan, sekaligus menegaskan prinsip enkapsulasi karena tidak mengakses struktur internal ADT secara langsung.

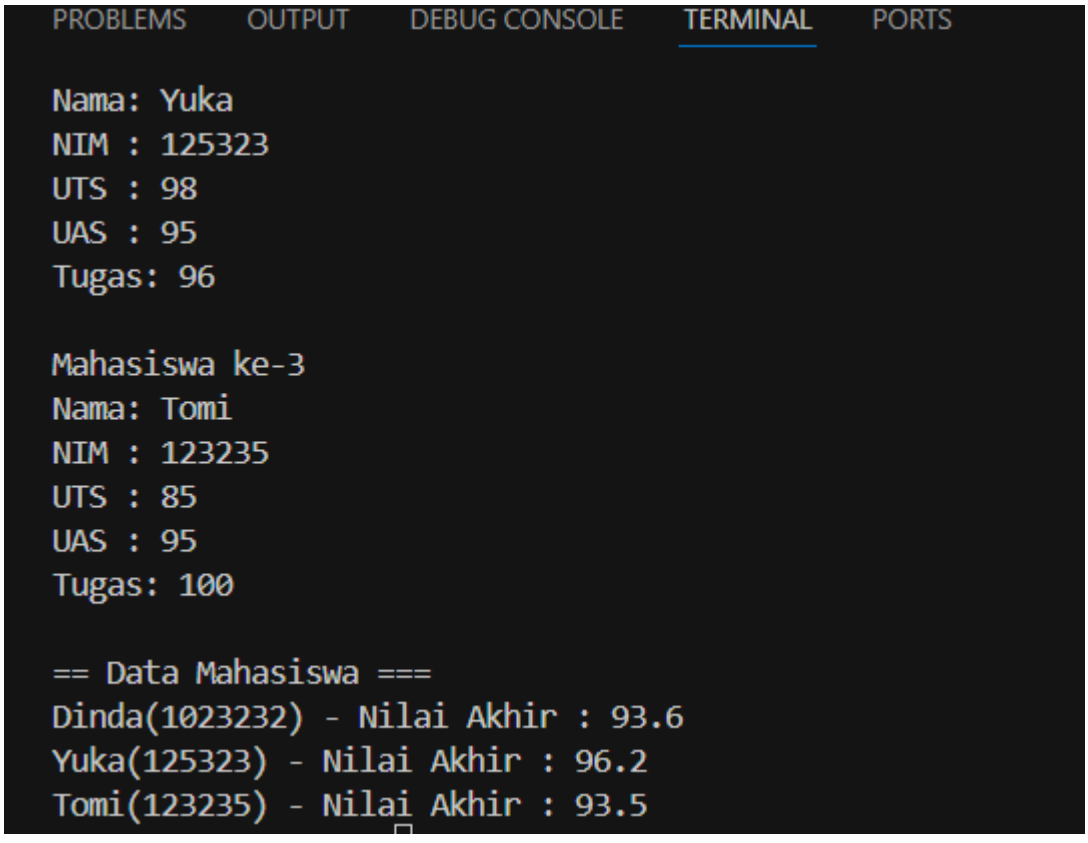
C. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Unguided 1

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

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

int main() {
    struct mahasiswa {
        string nama, nim;
        float uts, uas, tugas, nilaiAkhir;
    } mhs[10];

    int n;
    cout << "Jumlah mahasiswa (max 10): ";
    cin >> n;
    cin.ignore();

    for (int i = 0; i < n; i++) {
        cout << "\nMahasiswa ke-" << i + 1 << endl;
        cout << "Nama: "; getline(cin, mhs[i].nama);
        cout << "NIM : "; getline(cin, mhs[i].nim);
        cout << "UTS : "; cin >> mhs[i].uts;
        cout << "UAS : "; cin >> mhs[i].uas;
        cout << "Tugas: "; cin >> mhs[i].tugas;
```

```
            mhs[i].nilaiAkhir  =  HitungNilaiAkhir(mhs[i].uts,
mhs[i].uas, mhs[i].tugas);
        cin.ignore();
    }

    cout << "\n== Data Mahasiswa ===\n";
    for (int i = 0; i < n; i++)
         cout << mhs[i].nama << "(" << mhs [i].nim << ") - Nilai
Akhir : "
             << mhs[i].nilaiAkhir << endl;

    return 0;
  }
```

Screenshots Output



Deskripsi:

Program ini adalah sistem sederhana untuk menginput dan menghitung nilai akhir mahasiswa. Ia menggunakan sebuah struktur data untuk menyimpan nama, NIM, dan tiga komponen nilai (UTS, UAS, Tugas), serta sebuah fungsi yang menghitung Nilai Akhir dengan pembobotan 30% (UTS), 40% (UAS), dan 30% (Tugas). Setelah data semua mahasiswa diinput, program akan menampilkan ringkasan yang berisi nama, NIM, dan Nilai Akhir dari setiap mahasiswa.

Unguided 2

pelajaran.h

```h
#ifndef PELAJARAN_H
#define PELAJARAN_H
#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.cpp

```cpp
#include <iostream>
#include "pelajaran.h"
using namespace std;
Pelajaran create_pelajaran(string namaMapel, string kodeMapel)
{
```

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

main2.cpp

```
#include <iostream>
#include "pelajaran.h"
#include "pelajaran.cpp"
using namespace std;
int main()
{
    string namaMapel = "Struktur Data";
    string kodeMapel = "STD";
    Pelajaran pel = create_pelajaran(namaMapel, kodeMapel);
    tampil_pelajaran(pel);
    return 0;
}
```

Screenshoot Output

```
PS D:\C++\Modul 3> cd "d:\C++\Modul 3\" ; if ($?) { g++ main2.cpp -o main2 } ; if ($?) { .\main2 }
nama pelajaran : Struktur Data
nilai : STD
```

Deskripsi:

ini mendemonstrasikan implementasi Abstract Data Type (ADT) sederhana untuk mengelola data pelajaran. Implementasi ini menggunakan struct untuk mendefinisikan tipe data, dilengkapi dengan fungsi-fungsi dasar seperti fungsi pembentuk (untuk inisialisasi atau *create*) dan fungsi penampil (*display*). Tujuan utamanya adalah menyimpan dan memanipulasi informasi berupa nama dan kode mata pelajaran.

Unguided 3

```cpp
#include <iostream>
using namespace std;
void tampil(int a[3][3])
{
    for (int i = 0; i < 3; i++, cout << endl)
        for (int j = 0; j < 3; j++)
            cout << a[i][j] << " ";
}
void tukarArray(int a[3][3], int b[3][3], int i, int j)
{
    int t = a[i][j];
    a[i][j] = b[i][j];
    b[i][j] = t;
}
void tukarPtr(int *p1, int *p2)
{
    int t = *p1;
    *p1 = *p2;
    *p2 = t;
```

```
}
int main()
{
    int A[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int B[3][3] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
    int x = 10, y = 20, *p1 = &x, *p2 = &y;
    cout << "A sebelum tukar:\n";
    tampil(A);
    cout << "\nB sebelum tukar:\n";
    tampil(B);
    tukarArray(A, B, 1, 2);
    cout << "\n\nSetelah tukar [1][2]:\nA:\n";
    tampil(A);
    cout << "\nB:\n";
    tampil(B);
    cout << "\n\nSebelum tukar pointer: x=" << x << ", y=" << y;
    tukarPtr(p1, p2);
     cout << "\nSetelah tukar pointer: x=" << x << ", y=" << y <<
endl;
}
```

Screenshots Output

Deskripsi:

Program ini mendemonstrasikan pertukaran nilai (*swapping*) menggunakan dua metode: array dua dimensi dan pointer. Program menginisialisasi dua matriks 3×3 (A dan B) dan dua variabel (x dan y). Fungsi tukarArray menukar elemen spesifik antar matriks, sedangkan fungsi tukarPtr menukar nilai x dan y menggunakan *pointer* (*call by reference*). Tujuannya adalah memperlihatkan bagaimana nilai dapat diubah baik langsung melalui array maupun secara tidak langsung melalui alamat memori.

D. Kesimpulan

Secara keseluruhan, praktikum ini menyimpulkan bahwa Abstract Data Type (ADT) penting untuk menciptakan program yang modular dan terenkapsulasi dengan memisahkan definisi data (struct) dari operasinya (.h dan .cpp). Konsep array digunakan untuk penyimpanan data sejenis yang berurutan, sementara pointer terbukti esensial untuk mengakses dan memanipulasi data melalui alamat memori, terutama dalam implementasi mekanisme *call by reference* untuk mengubah nilai variabel asli di luar fungsi. Penerapan seluruh konsep ini memungkinkan pengelolaan data yang efisien dan pemecahan masalah komputasi secara terstruktur.

E. Referensi

https://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B

https://learn.microsoft.com/id-id/cpp/cpp/references-cpp?view=msvc-170

https://medium.com/@mohamedeissabay/understanding-c-a-deep-dive-into-core-concepts-48a560679cdd