

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

MODUL III

ABSTRACT DATA TYPE (ADT)



Disusun Oleh :

Muhammad Fathammubina

NIM : 10 3112430188

Dosen

WAHYU ANDI SAPUTRA

**PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

A. Dasar Teori

Abstract Data Type (ADT) Tipe data yang didefinisikan bersama dengan sekumpulan operasi (fungsi/prosedur) yang bisa dilakukan terhadapnya. Singkatnya, ADT merupakan data + operasi yang boleh dilakukan dan tidak fokus pada cara penyimpanan data, tapi pada apa yang bisa dilakukan. Komponen dalam ADT meliputi :

1. Type — struktur data (misal struct Mahasiswa)
2. Primitif (operasi dasar), meliputi:
 - Konstruktor (Make...) : membuat objek
 - Selector (Get...) : mengambil nilai komponen
 - Mutator (Set...) : mengubah nilai komponen
 - Validator : memeriksa validitas data
 - Destructor : menghapus atau melepaskan memori
 - Baca/Tulis (I/O) : interaksi dengan input/output
 - Operator relasional / aritmatika
 - Konversi tipe

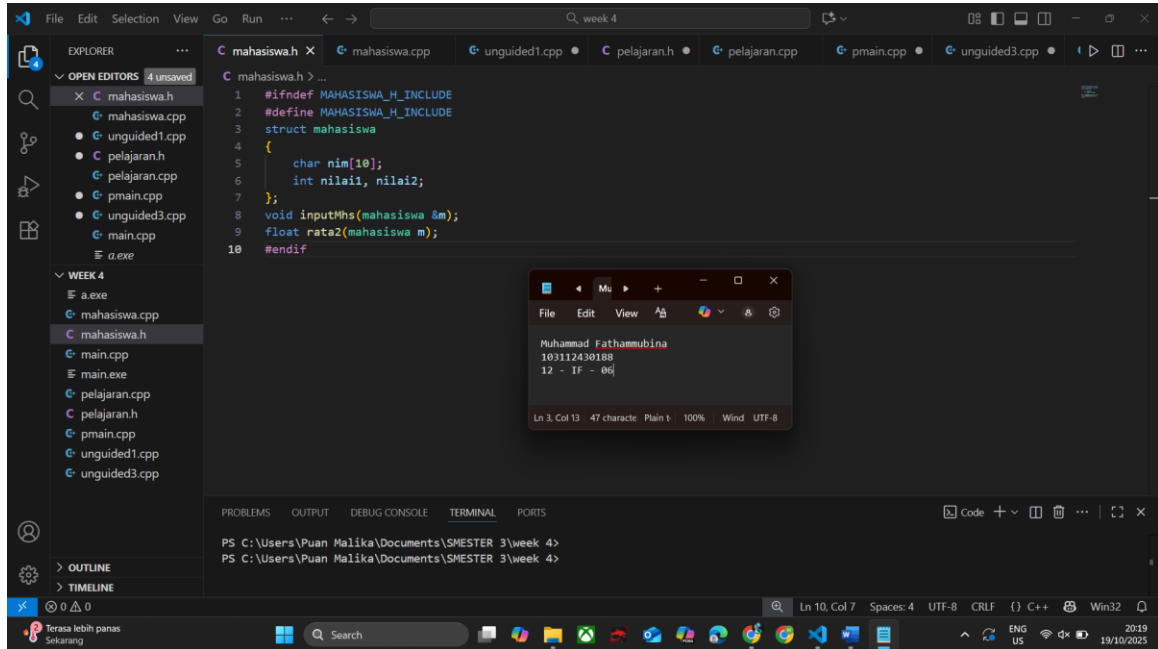
Struktur file ADT memiliki 3 file utama, yaitu :

- .h = Spesifikasi / header yang berisi struct dan deklarasi fungsi
- .cpp = Implementasi yang berisi fungsi-fungsi dari header
- Main.cpp = Program utama yang memakai ADT

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

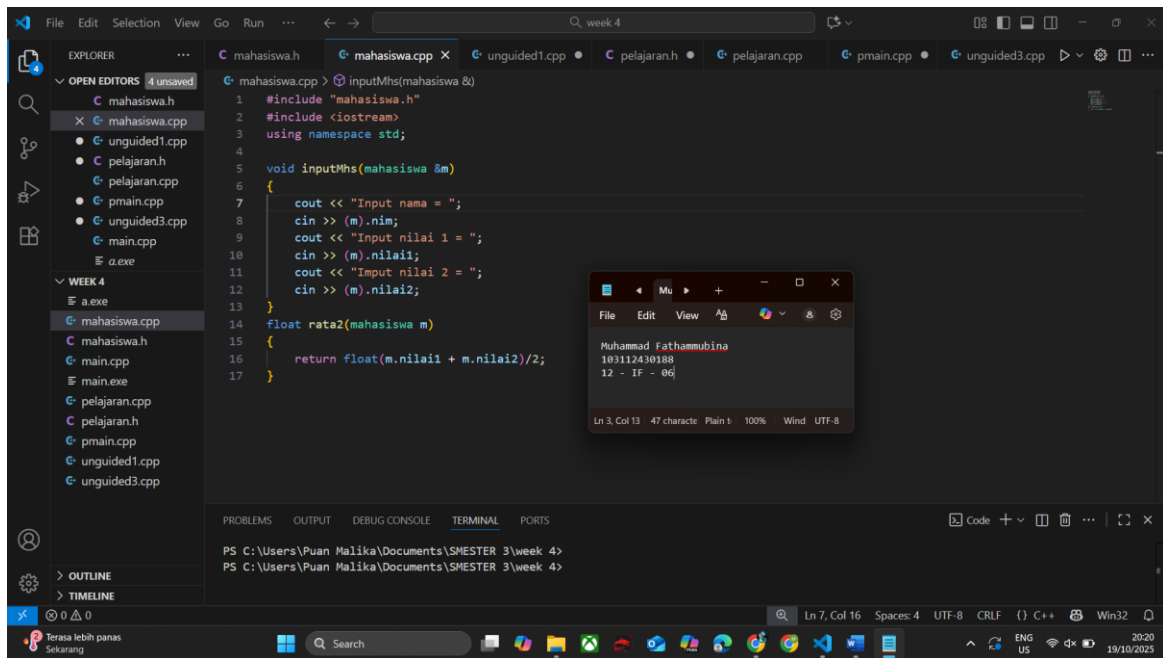
Guided 1

mahasiswa.h



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

mahasiswa.cpp

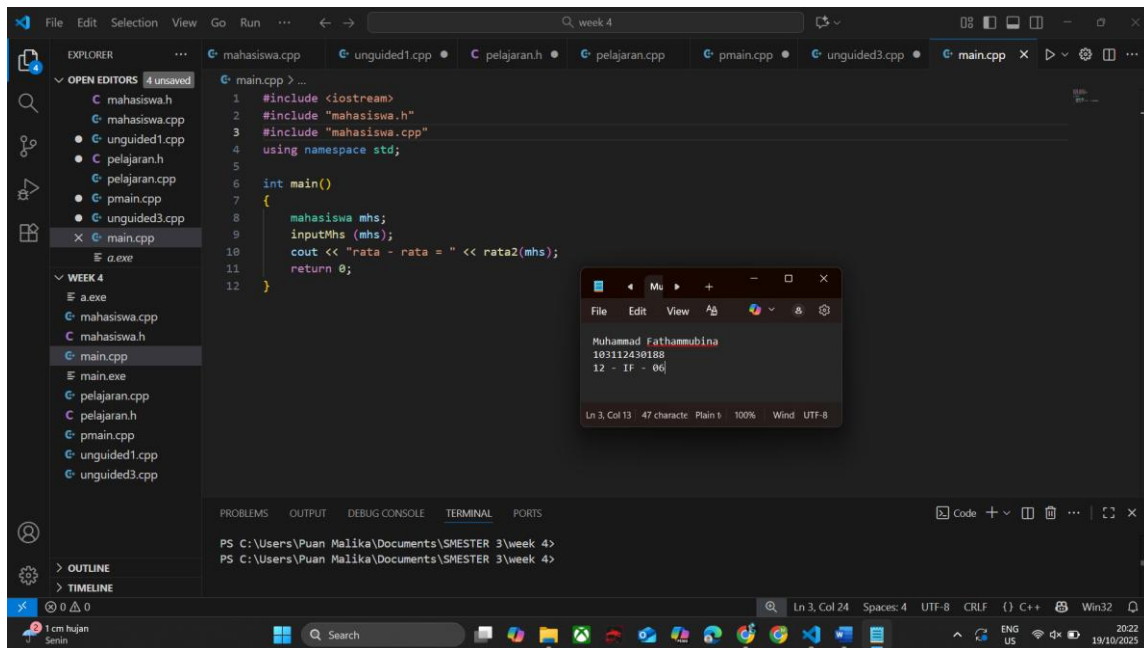


```
#include "mahasiswa.h"
#include <iostream>
using namespace std;

void inputMhs(mahasiswa &m)
{
    cout << "Input nama = ";
    cin >> (m).nim;
    cout << "Input nilai 1 = ";
    cin >> (m).nilai1;
    cout << "Input nilai 2 = ";
    cin >> (m).nilai2;
}

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

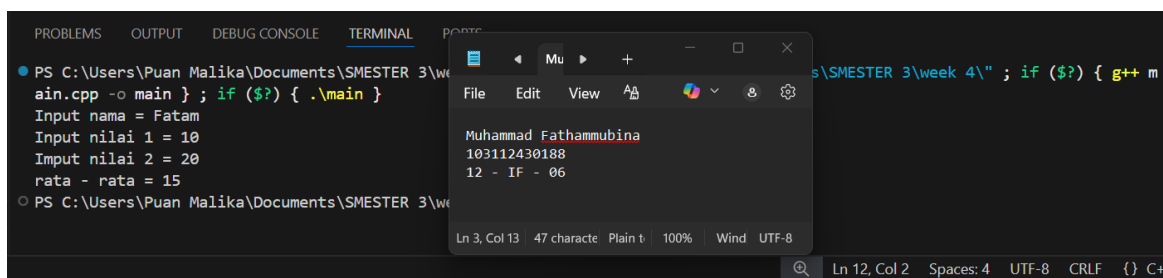
main.cpp



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

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

Screenshots Output



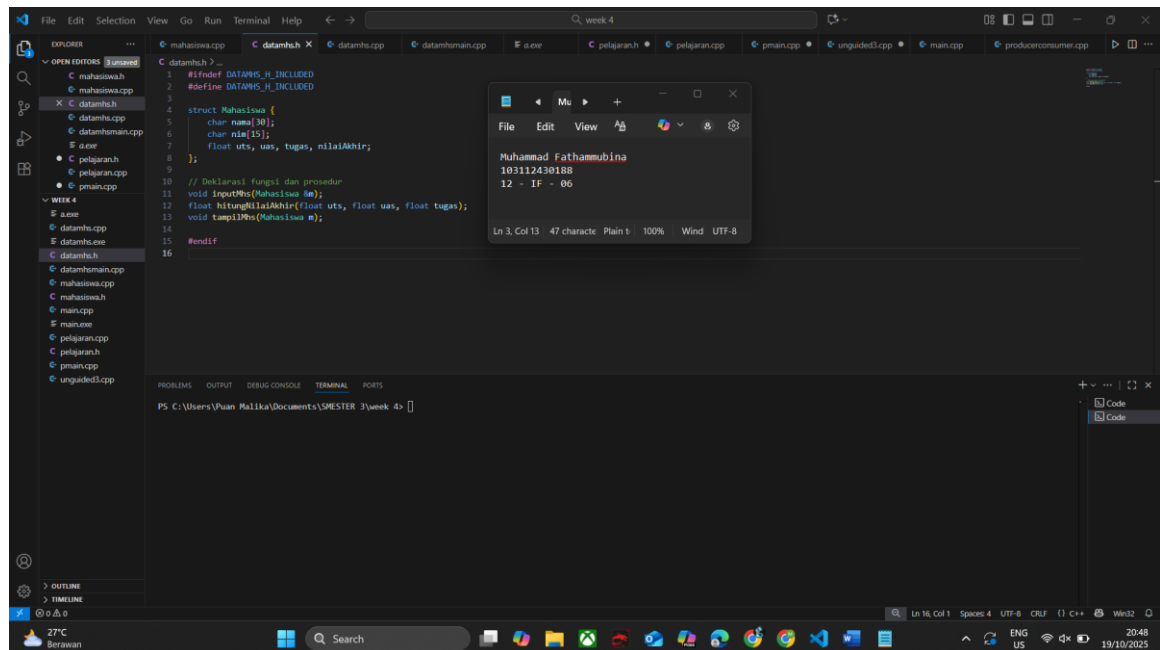
Deskripsi:

Program di atas merupakan penerapan konsep Abstract Data Type (ADT) dalam C++ yang menunjukkan bagaimana data dan operasi dipisahkan secara modular. Langkah pembuatannya diawali dengan membuat file header mahasiswa.h yang berisi definisi

struktur mahasiswa dan deklarasi fungsi inputMhs() serta rata2(). Selanjutnya, file mahasiswa.cpp dibuat untuk mengisi atau mengimplementasikan fungsi-fungsi tersebut, di mana inputMhs() berfungsi meminta input data mahasiswa (NIM dan dua nilai), sedangkan rata2() menghitung rata-rata dari kedua nilai tersebut. Terakhir, file main.cpp digunakan sebagai program utama yang memanggil fungsi-fungsi dari ADT untuk menerima data, menghitung nilai rata-rata, dan menampilkannya ke layar. Dengan pembagian ini, program menjadi lebih terstruktur, mudah dibaca, dan sesuai dengan prinsip modularitas ADT.

Unguided 1

datamhs.h



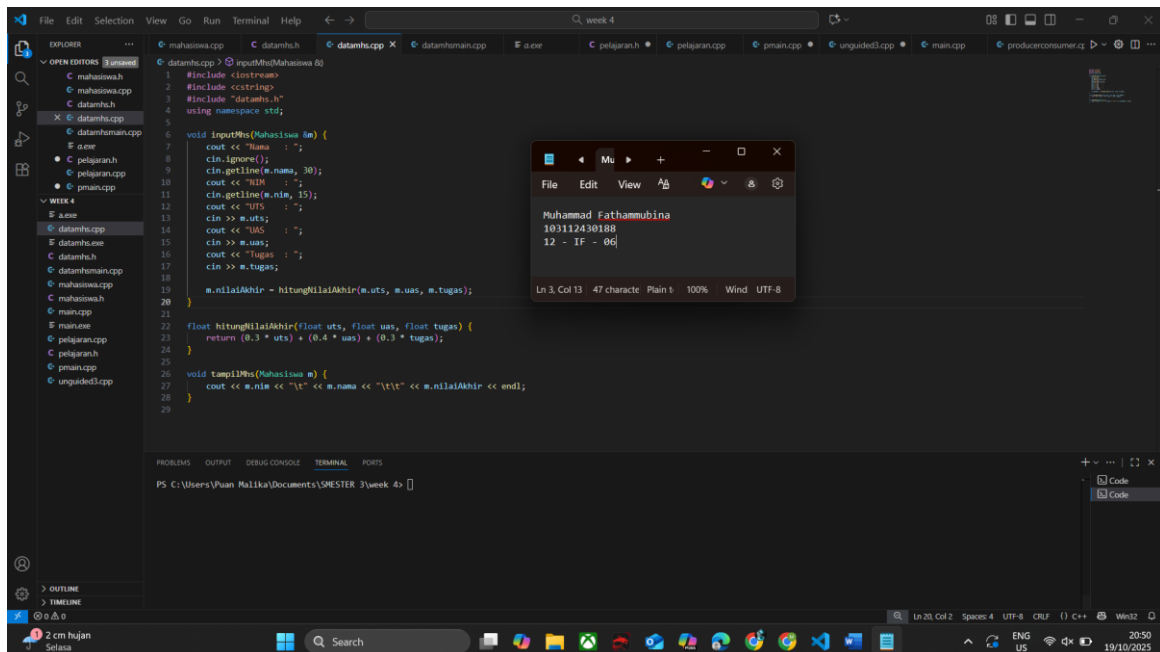
```
#ifndef DATAMHS_H_INCLUDED
#define DATAMHS_H_INCLUDED

struct Mahasiswa {
    char nama[30];
    char nim[15];
    float uts, uas, tugas, nilaiAkhir;
};

// Deklarasi fungsi dan prosedur
void inputMhs(Mahasiswa &m);
float hitungNilaiAkhir(float uts, float uas, float tugas);
void tampilMhs(Mahasiswa m);

#endif
```

datamhs.cpp



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

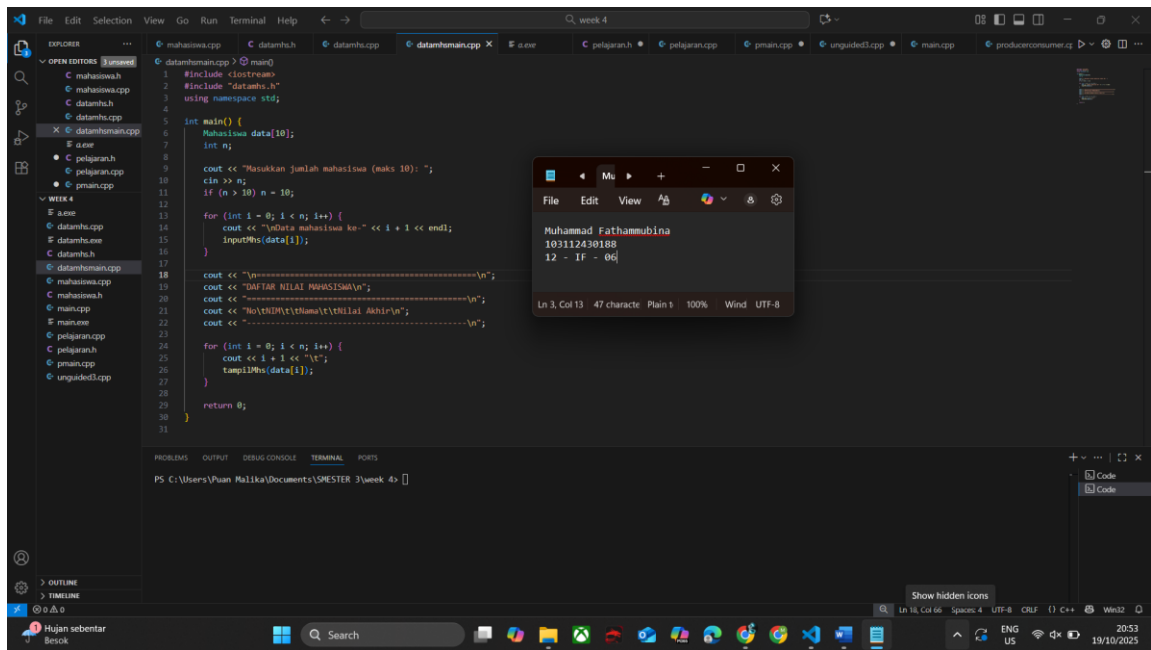
void inputMhs(Mahasiswa &m) {
    cout << "Nama : ";
    cin.ignore();
    cin.getline(m.nama, 30);
    cout << "NIM : ";
    cin.getline(m.nim, 15);
    cout << "UTS : ";
    cin >> m.uts;
    cout << "UAS : ";
    cin >> m.uas;
    cout << "Tugas : ";
    cin >> m.tugas;

    m.nilaiAkhir = hitungNilaiAkhir(m.uts, m.uas, m.tugas);
}

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

void tampilMhs(Mahasiswa m) {
    cout << m.nim << "\t" << m.nama << "\t\t" << m.nilaiAkhir << endl;
}
```

datamhsmain.cpp



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

int main() {
    Mahasiswa data[10];
    int n;

    cout << "Masukkan jumlah mahasiswa (maks 10): ";
    cin >> n;
    if (n > 10) n = 10;

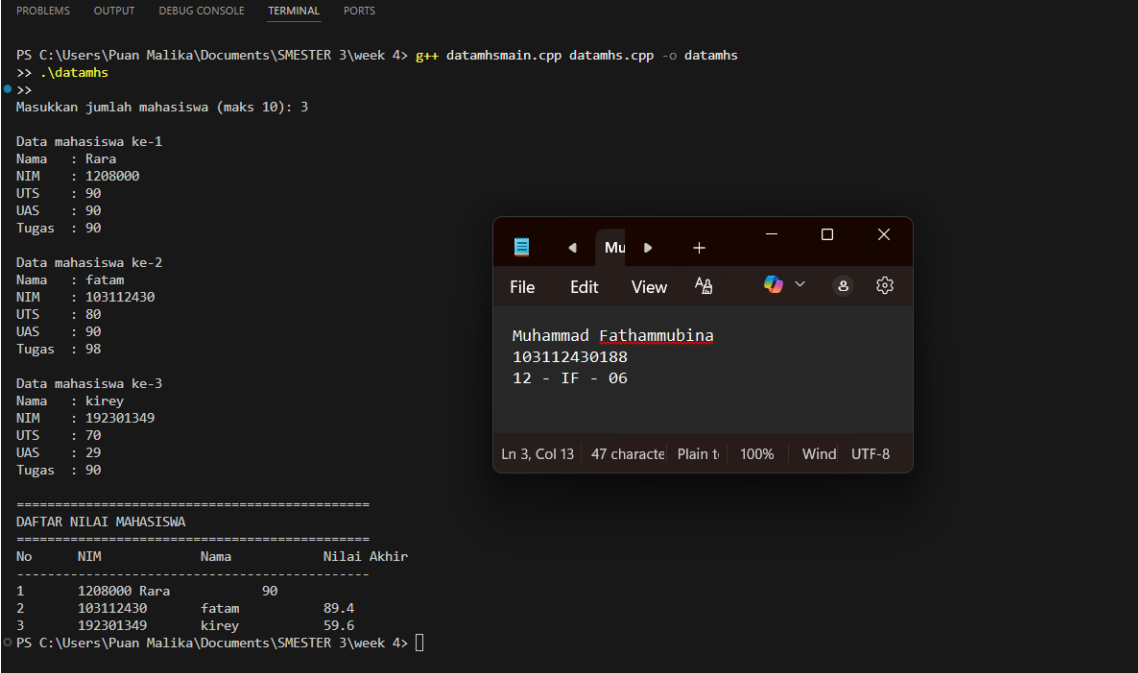
    for (int i = 0; i < n; i++) {
        cout << "\\nData mahasiswa ke-" << i + 1 << endl;
        inputMhs(data[i]);
    }

    cout << "\\n=====\\n";
    cout << "DAFTAR NILAI MAHASISWA\\n";
    cout << "=====\\n";
    cout << "No\\tNIM\\t\\tNama\\t\\tNilai Akhir\\n";
    cout << "-----\\n";

    for (int i = 0; i < n; i++) {
        cout << i + 1 << "\\t";
        tampilMhs(data[i]);
    }

    return 0;
}
```


Screenshots Output



```
PS C:\Users\Puan Malika\Documents\SEMESTER 3\week 4> g++ datamhsmain.cpp datamhs.cpp -o datamhs
>> .\datamhs
>>
Masukkan jumlah mahasiswa (maks 10): 3

Data mahasiswa ke-1
Nama : Rara
NIM : 1208000
UTS : 90
UAS : 90
Tugas : 90

Data mahasiswa ke-2
Nama : fatam
NIM : 103112430
UTS : 80
UAS : 90
Tugas : 98

Data mahasiswa ke-3
Nama : kirey
NIM : 192301349
UTS : 70
UAS : 29
Tugas : 90

=====
DAFTAR NILAI MAHASISWA
=====
No      NIM      Nama      Nilai Akhir
-----
1       1208000  Rara      90
2       103112430  fatam      89.4
3       192301349  kirey      59.6
© PS C:\Users\Puan Malika\Documents\SEMESTER 3\week 4>
```

Inset Notepad++ content:

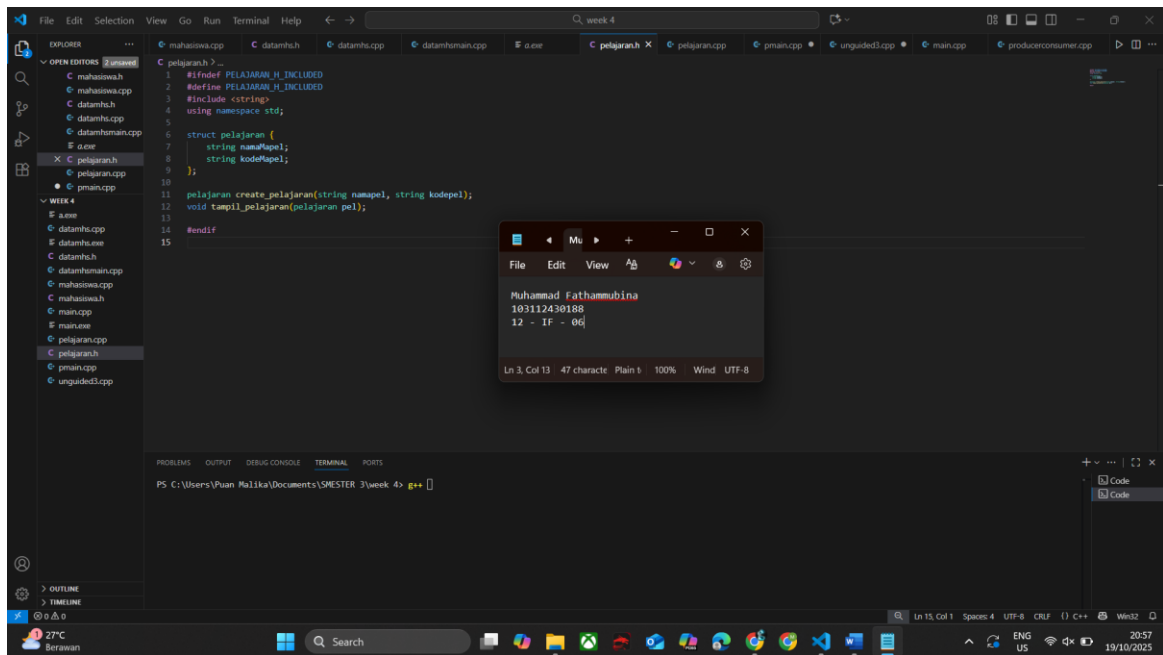
```
Muhammad Fathammubina
103112430188
12 - IF - 06
```

Deskripsi:

Program ini merupakan implementasi lengkap dari Abstract Data Type (ADT) bernama `datamhs`, yang digunakan untuk mengelola data nilai mahasiswa secara modular dalam bahasa C++. File `datamhs.h` berfungsi sebagai header file yang mendefinisikan struktur Mahasiswa dan mendeklarasikan fungsi-fungsi yang akan digunakan. File `datamhs.cpp` berisi implementasi fungsi-fungsi tersebut, yaitu `inputMhs()` untuk menerima input data mahasiswa, `hitungNilaiAkhir()` untuk menghitung nilai akhir berdasarkan rumus $0.3 * UTS + 0.4 * UAS + 0.3 * tugas$, serta `tampilMhs()` untuk menampilkan hasil data mahasiswa ke layar. Sementara itu, file `main.cpp` menjadi program utama yang memanggil fungsi-fungsi dari ADT untuk mengisi, menghitung, dan menampilkan data nilai mahasiswa. Pemisahan kode menjadi tiga file ini menunjukkan penerapan prinsip ADT yang menekankan modularitas, keterbacaan, dan kemudahan pemeliharaan program.

Unguided 2

`pelajaran.h`



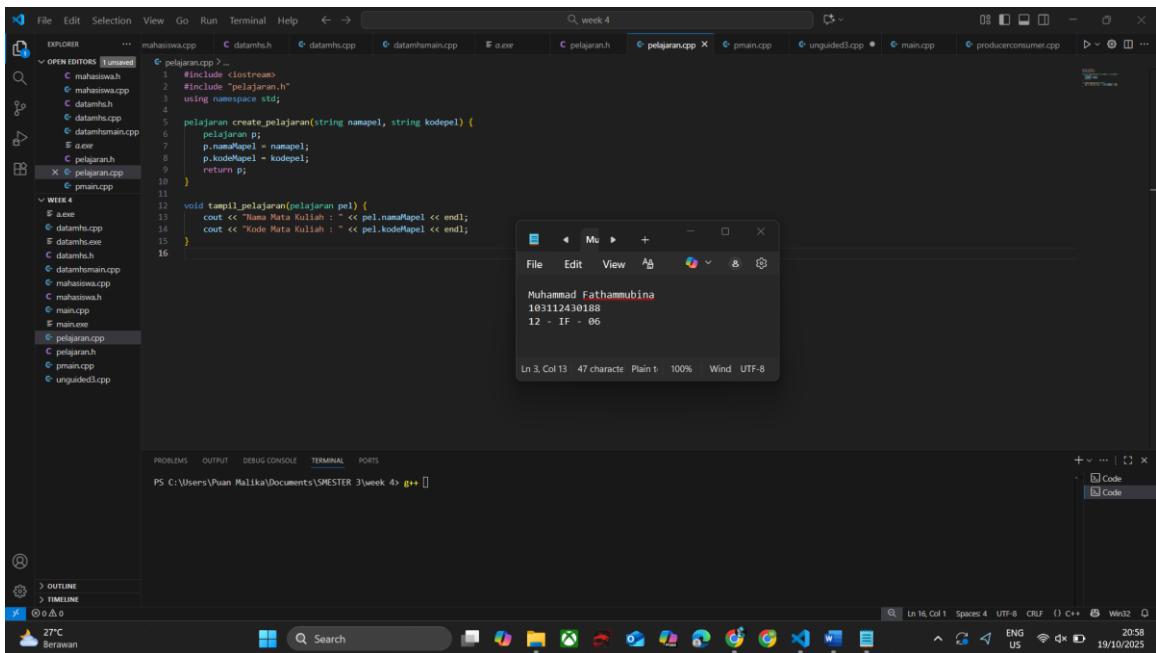
```
#ifndef PELAJARAN_H_INCLUDED
#define PELAJARAN_H_INCLUDED
#include <string>
using namespace std;

struct pelajaran {
    string namaMapel;
    string kodeMapel;
};

pelajaran create_pelajaran(string namapel, string kodepel);
void tampil_pelajaran(pelajaran pel);

#endif
```

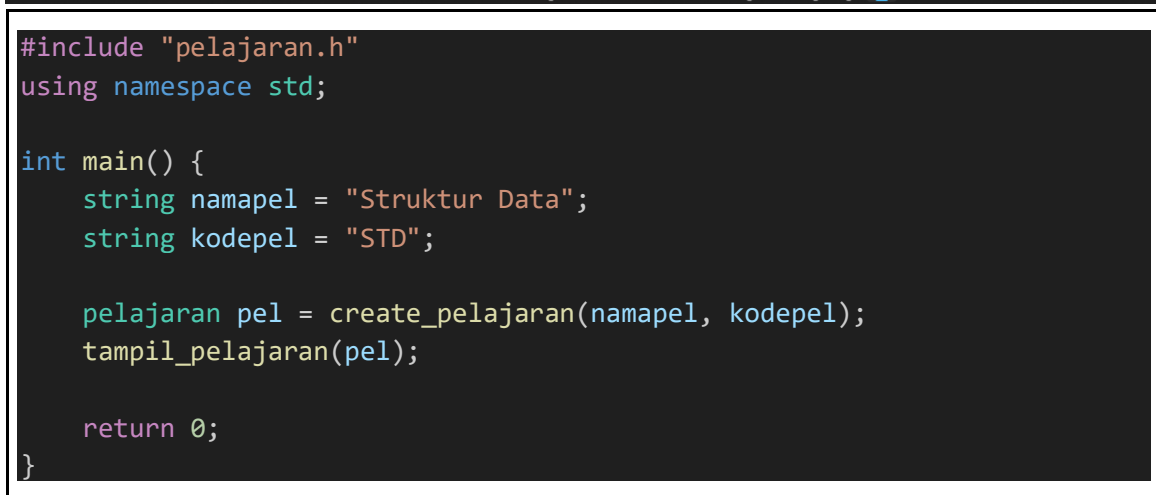
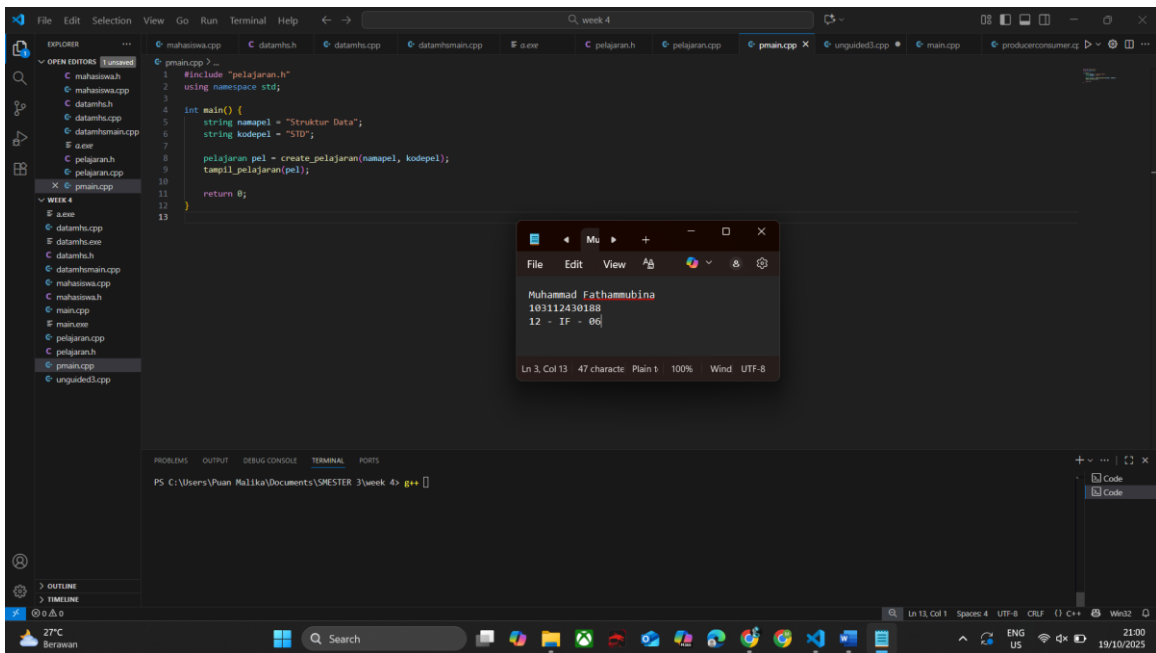
pelajaran.cpp



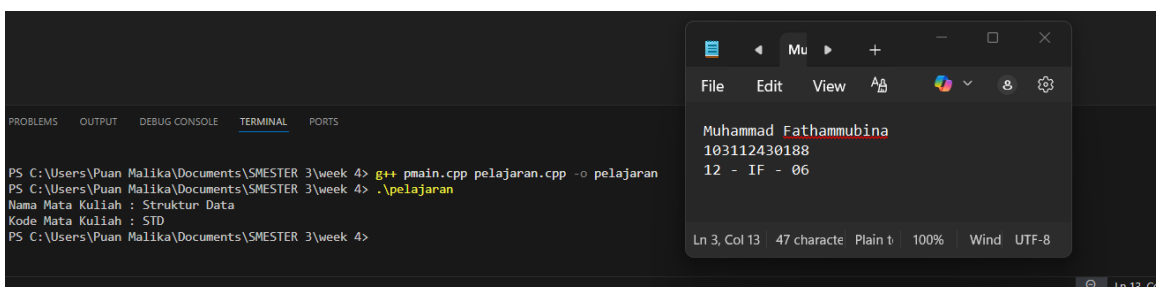
```
#include <iostream>
#include "pelajaran.h"
using namespace std;

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

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



Screenshots Output



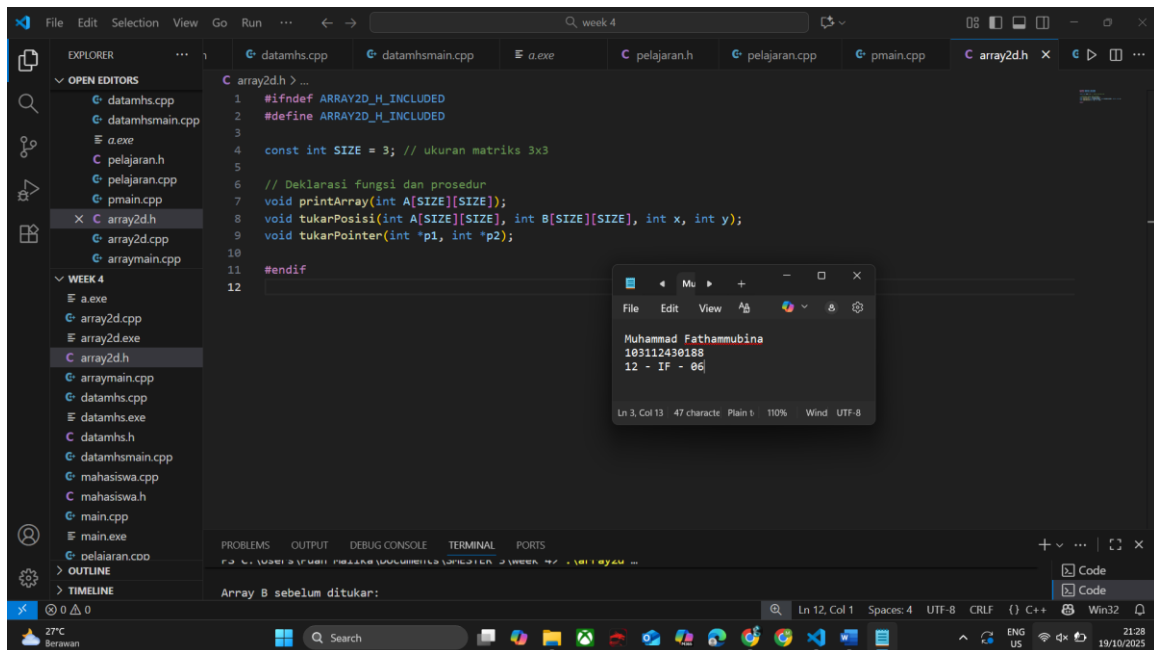
Deskripsi:

Program ini merupakan contoh penerapan Abstract Data Type (ADT) bernama pelajaran, yang dirancang untuk menyimpan dan menampilkan informasi mata kuliah secara terstruktur. File pelajaran.h berfungsi sebagai header file yang mendefinisikan struktur pelajaran dengan dua atribut, yaitu namaMapel untuk nama mata kuliah dan kodeMapel untuk kode mata kuliah, serta mendeklarasikan dua fungsi utama: create_pelajaran() dan

tampil_pelajaran(). File pelajaran.cpp berisi implementasi kedua fungsi tersebut, di mana create_pelajaran() digunakan untuk membuat dan mengisi objek pelajaran berdasarkan input nama dan kode, sedangkan tampil_pelajaran() menampilkan data mata kuliah ke layar. Sementara itu, file main.cpp berfungsi sebagai program utama yang membuat objek pelajaran, mengisinya dengan data contoh (“Struktur Data”, “STD”), dan menampilkannya. Pemisahan kode menjadi tiga file ini menggambarkan penerapan prinsip modularitas dan enkapsulasi dalam ADT, sehingga program lebih rapi, mudah dibaca, dan mudah dikembangkan.

Unguided 3

array2d.h



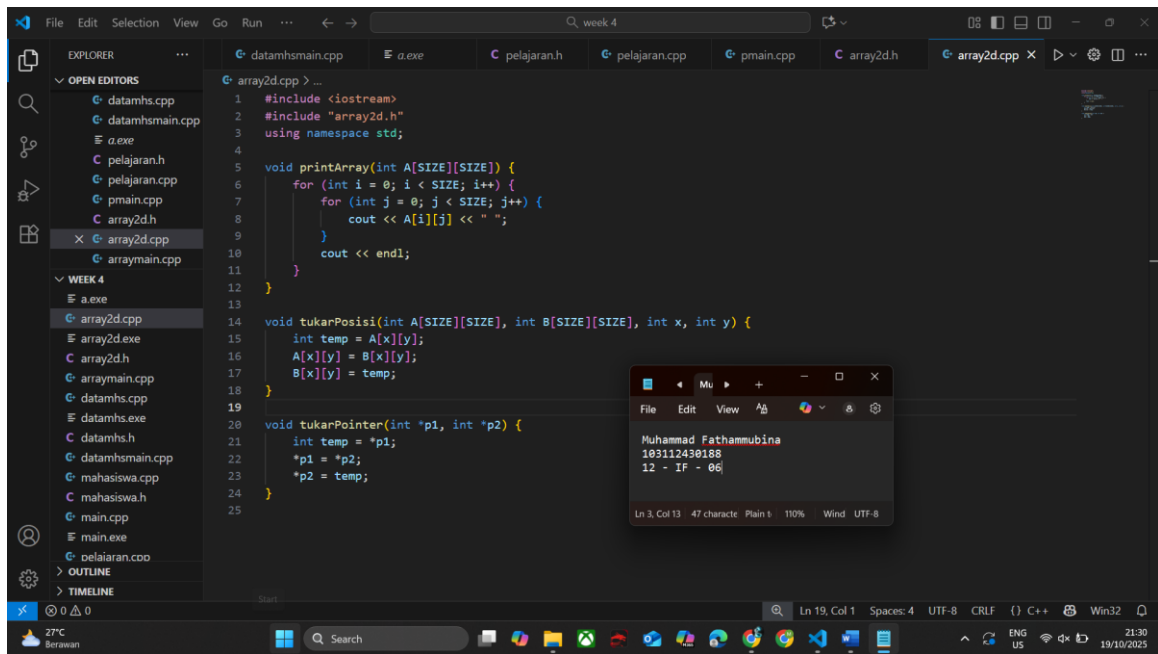
```
#ifndef ARRAY2D_H_INCLUDED
#define ARRAY2D_H_INCLUDED

const int SIZE = 3; // ukuran matriks 3x3

// Deklarasi fungsi dan prosedur
void printArray(int A[SIZE][SIZE]);
void tukarPosisi(int A[SIZE][SIZE], int B[SIZE][SIZE], int x, int y);
void tukarPointer(int *p1, int *p2);

#endif
```

array2d.cpp



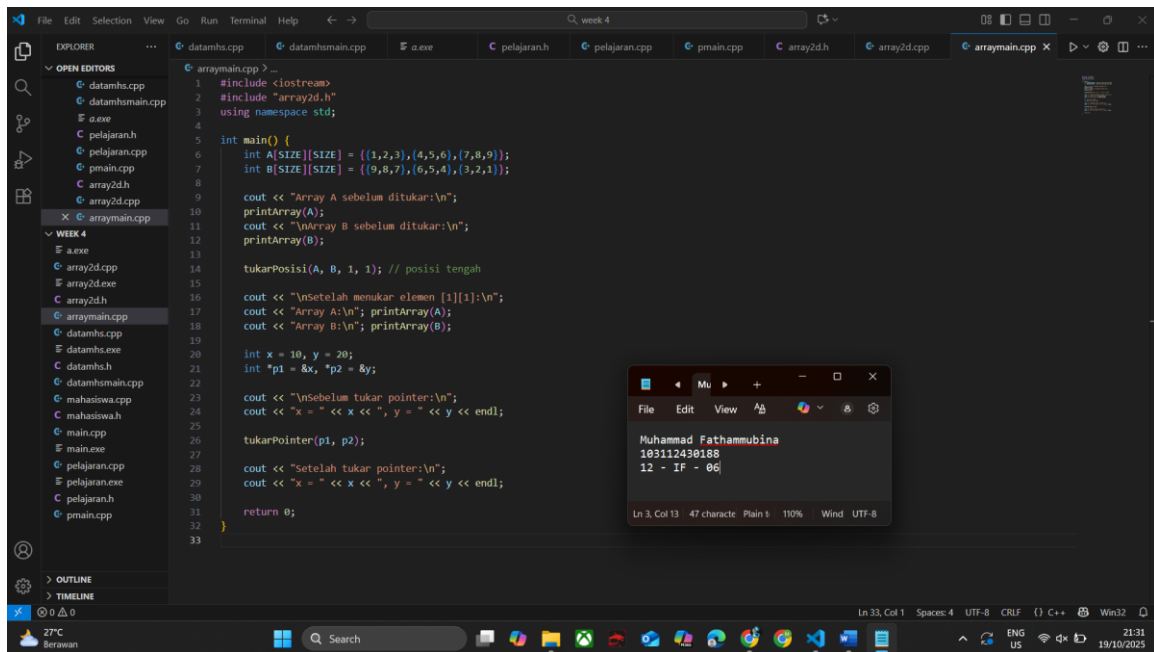
```
#include <iostream>
#include "array2d.h"
using namespace std;

void printArray(int A[SIZE][SIZE]) {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            cout << A[i][j] << " ";
        }
        cout << endl;
    }
}

void tukarPosisi(int A[SIZE][SIZE], int B[SIZE][SIZE], int x, int y) {
    int temp = A[x][y];
    A[x][y] = B[x][y];
    B[x][y] = temp;
}

void tukarPointer(int *p1, int *p2) {
    int temp = *p1;
    *p1 = *p2;
    *p2 = temp;
}
```

arraymain.cpp



```
#include <iostream>
#include "array2d.h"
using namespace std;

int main() {
    int A[SIZE][SIZE] = {{1,2,3},{4,5,6},{7,8,9}};
    int B[SIZE][SIZE] = {{9,8,7},{6,5,4},{3,2,1}};

    cout << "Array A sebelum ditukar:\n";
    printArray(A);
    cout << "\nArray B sebelum ditukar:\n";
    printArray(B);

    tukarPosisi(A, B, 1, 1); // posisi tengah

    cout << "\nSetelah menukar elemen [1][1]:\n";
    cout << "Array A:\n"; printArray(A);
    cout << "Array B:\n"; printArray(B);

    int x = 10, y = 20;
    int *p1 = &x, *p2 = &y;

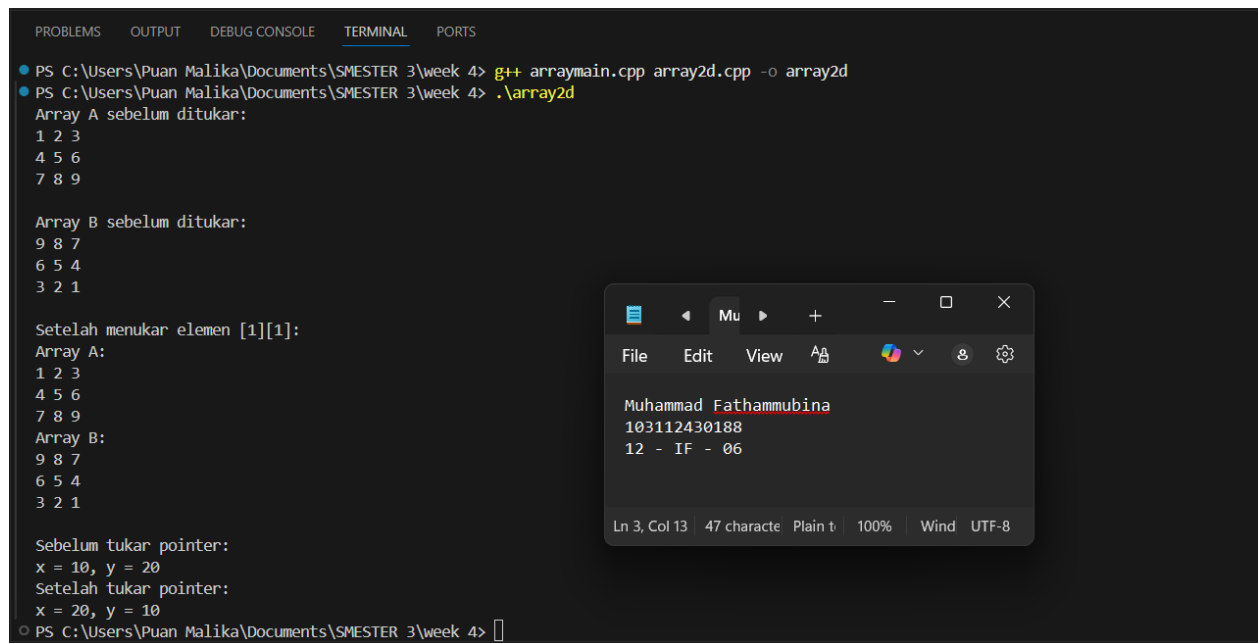
    cout << "\nSebelum tukar pointer:\n";
    cout << "x = " << x << ", y = " << y << endl;

    tukarPointer(p1, p2);

    cout << "Setelah tukar pointer:\n";
    cout << "x = " << x << ", y = " << y << endl;
```

```
    return 0;
}
```

Screenshots Output



```
PS C:\Users\Puan Malika\Documents\SMESTER 3\week 4> g++ arraymain.cpp array2d.cpp -o array2d
PS C:\Users\Puan Malika\Documents\SMESTER 3\week 4> .\array2d

Array A sebelum ditukar:
1 2 3
4 5 6
7 8 9

Array B sebelum ditukar:
9 8 7
6 5 4
3 2 1

Setelah menukar elemen [1][1]:
Array A:
1 2 3
4 5 6
7 8 9
Array B:
9 8 7
6 5 4
3 2 1

Sebelum tukar pointer:
x = 10, y = 20
Setelah tukar pointer:
x = 20, y = 10
PS C:\Users\Puan Malika\Documents\SMESTER 3\week 4>
```

Deskripsi:

Program ini merupakan implementasi Abstract Data Type (ADT) bernama array2d, yang digunakan untuk memanipulasi dan menampilkan data dalam bentuk array dua dimensi berukuran 3×3 serta mendemonstrasikan konsep pointer. File array2d.h berfungsi sebagai header file yang berisi deklarasi konstanta SIZE dan tiga fungsi utama: printArray() untuk menampilkan isi array 2D, tukarPosisi() untuk menukar elemen antara dua array pada posisi tertentu, dan tukarPointer() untuk menukar nilai dua variabel melalui pointer. File array2d.cpp berisi implementasi ketiga fungsi tersebut, di mana setiap fungsi menjalankan operasi sesuai perannya terhadap array atau pointer yang diberikan. Sementara itu, file main.cpp menjadi program utama yang membuat dua array 3×3 (A dan B), menampilkan isinya sebelum dan sesudah penukaran elemen, serta menguji fungsi tukarPointer() dengan dua variabel integer. Pembagian kode ini menunjukkan penerapan prinsip modularitas dan abstraksi data pada ADT, sehingga program lebih mudah dibaca, diuji, dan dikembangkan.

C. Kesimpulan

Dari hasil praktikum Modul III tentang Abstract Data Type (ADT), dapat disimpulkan bahwa ADT merupakan konsep penting dalam pemrograman terstruktur yang memisahkan antara definisi data dan operasi yang dapat dilakukan terhadap data tersebut. Melalui implementasi beberapa latihan seperti mahasiswa, datamhs, pelajaran, dan array2d, dapat dipahami bahwa penggunaan ADT membantu membuat program menjadi

lebih modular, mudah dipahami, dan mudah dikembangkan. Dengan memisahkan kode ke dalam tiga bagian utama yaitu file header (.h), file implementasi (.cpp), dan file utama (main.cpp), struktur program menjadi lebih rapi dan mendukung prinsip enkapsulasi serta pemeliharaan kode yang baik. Selain itu, penerapan fungsi dan prosedur dalam ADT menunjukkan bagaimana logika program dapat dibangun secara sistematis untuk mengelola data secara efisien.

D. Referensi

Muliono, R. (2017). Abstract Data Type (ADT). Universitas Multimedia Nusantara.

Steven, J, Z. (2019). Menerapkan ADT di Kelas C++. Universitas Old Domino.