

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

**MODUL III
ABSTRACT DATA TYPE (ADT)**



Disusun Oleh :

Nama : Besthian Guido Rafael Simbolon
NIM : 103112430258

Dosen

FAHRUDIN MUKTI WIBOWO

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

A. Dasar Teori

Abstract Data Type (ADT) atau tipe data abstrak adalah konsep penting dalam pemrograman yang digunakan untuk mendefinisikan suatu tipe data beserta operasi-operasi dasarnya, tanpa memperlihatkan detail cara kerjanya. Jadi, ADT lebih fokus pada apa yang dilakukan daripada bagaimana caranya dilakukan. Misalnya, kita bisa membuat ADT “mahasiswa” yang memiliki data seperti NIM dan nilai, serta operasi seperti inputMhs() untuk memasukkan data dan rata2() untuk menghitung nilai rata-rata, tanpa harus tahu detail proses di dalam fungsinya.

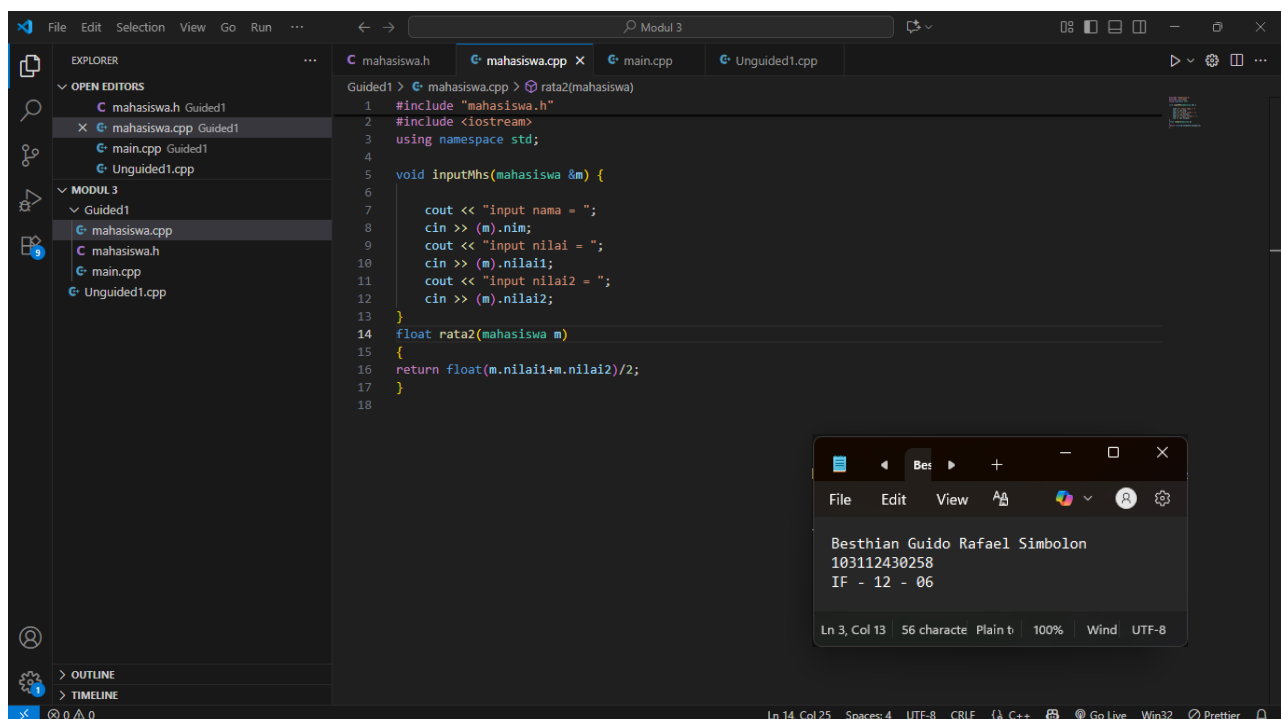
Dalam penerapannya, ADT biasanya terdiri dari dua bagian utama, yaitu definisi tipe dan fungsi (header) yang berisi deklarasi struktur data dan nama fungsi, serta implementasi fungsi (body) yang berisi kode program sebenarnya. Dalam bahasa C++ biasanya file header diberi ekstensi .h dan implementasinya di .cpp. Pemisahan ini bertujuan agar kode lebih rapi, mudah dibaca, dan bisa digunakan ulang di program lain.

Selain itu, ADT juga memiliki beberapa jenis operasi dasar, seperti konstruktor (untuk membuat data baru), selektor (untuk mengambil data), mutator (untuk mengubah data), destruktur (untuk menghapus data), serta operasi tambahan seperti baca/tulis dan perbandingan. Dengan menerapkan ADT, program menjadi lebih terstruktur, mudah dikembangkan, dan logika program lebih jelas karena setiap tipe data memiliki tanggung jawab dan fungsi masing-masing.

B. Guided

Guided 1

Mahasiswa.cpp



```
Guided1 > C:\mahasiswa.cpp > rata2(mahasiswa)
1  #include "mahasiswa.h"
2  #include <iostream>
3  using namespace std;
4
5  void inputMhs(mahasiswa &m) {
6
7      cout << "input nama = ";
8      cin >> (m).nim;
9      cout << "input nilai = ";
10     cin >> (m).nilai1;
11     cout << "input nilai2 = ";
12     cin >> (m).nilai2;
13 }
14 float rata2(mahasiswa m)
15 {
16     return float(m.nilai1+m.nilai2)/2;
17 }
18
```

Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06

Ln 3, Col 13 | 56 character | Plain text | 100% | Wind | UTF-8

Code:

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

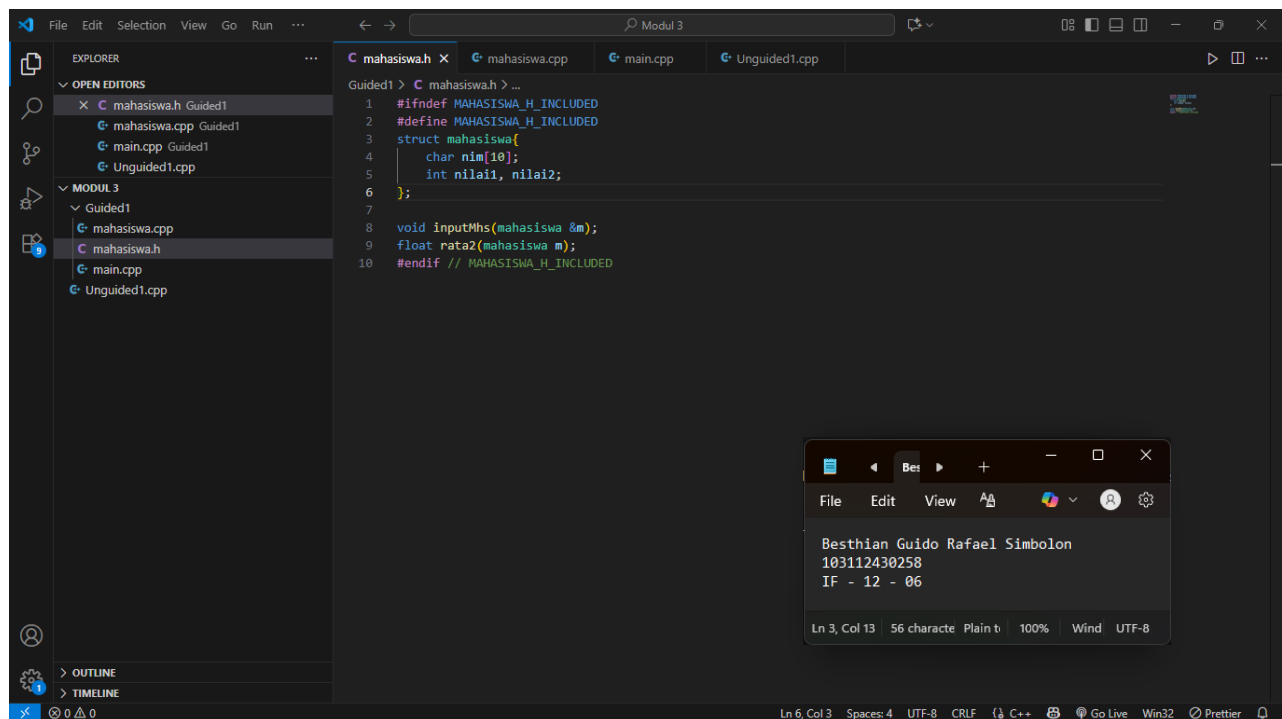
void inputMhs(mahasiswa &m) {

    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;
}
```

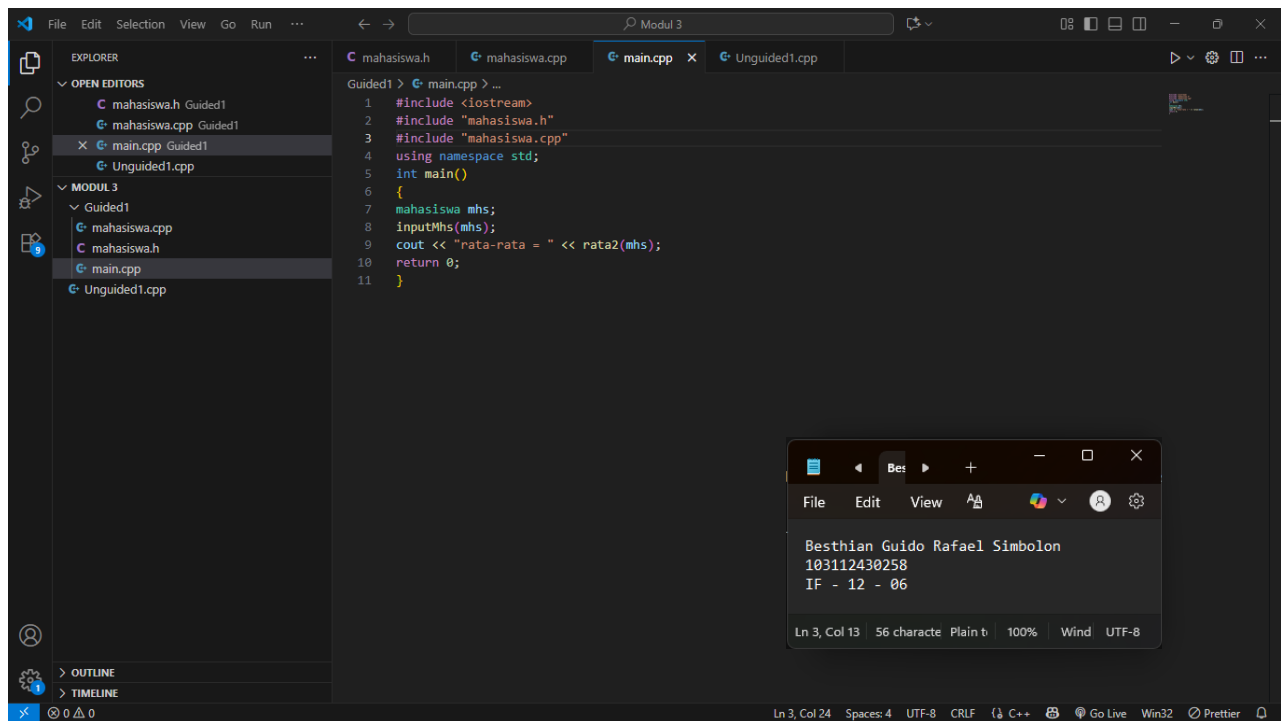
Mahasiswa.h



```
#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
```

main.cpp



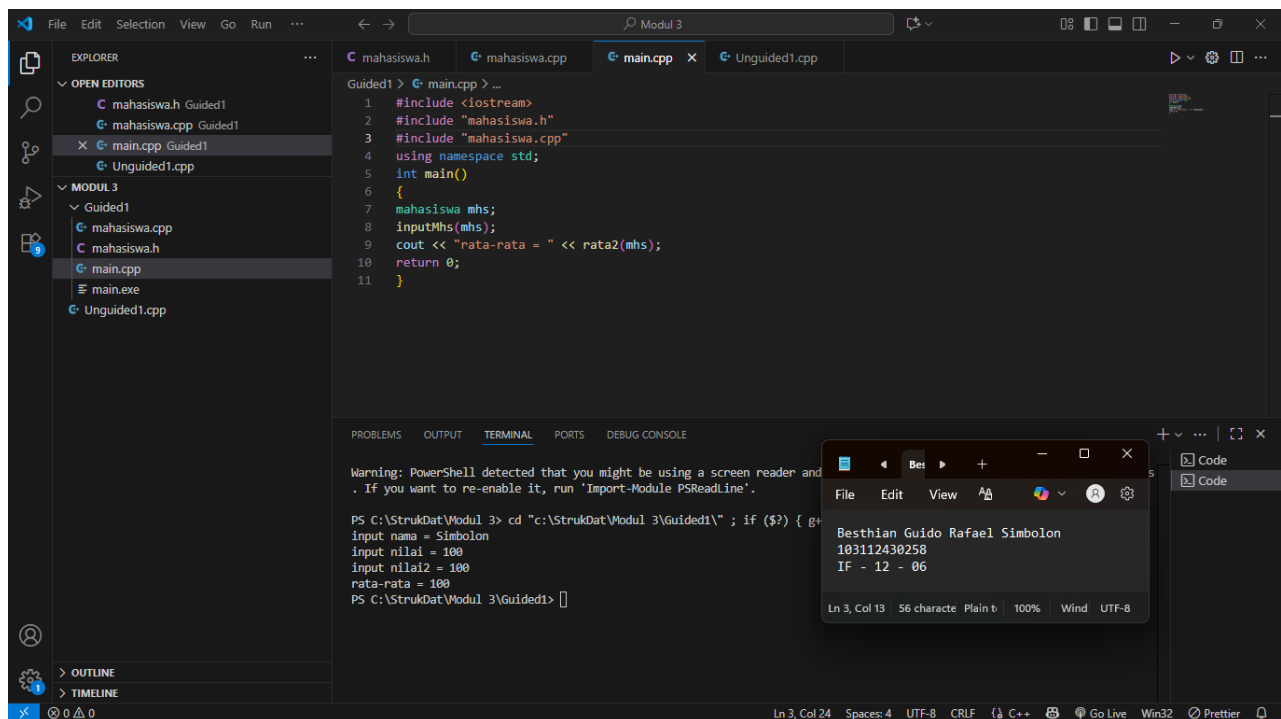
```

#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



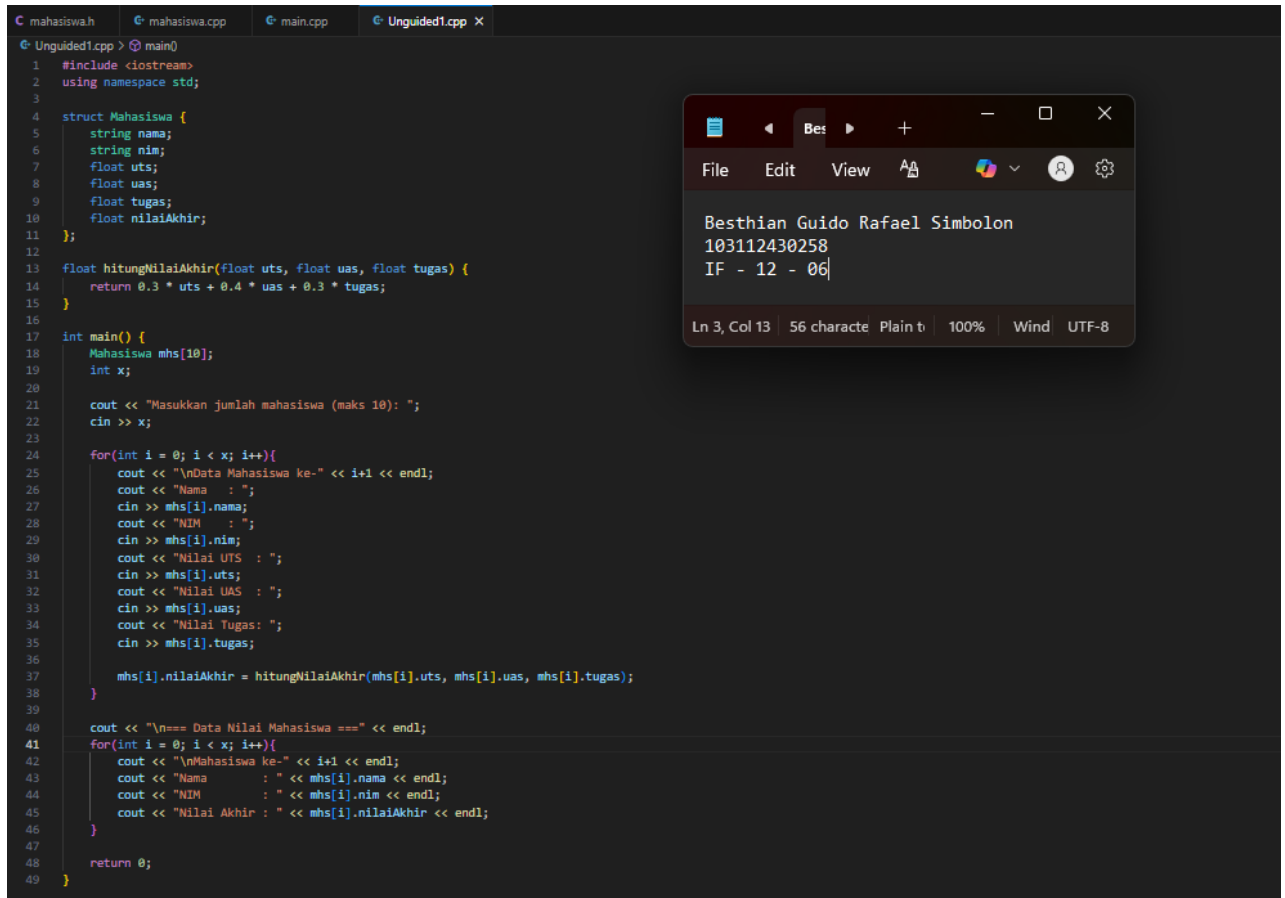
Deskripsi

Kode di atas merupakan contoh penerapan Abstract Data Type (ADT) dalam C++ yang digunakan untuk menginput data mahasiswa dan menghitung nilai rata-ratanya. Program ini terdiri dari tiga file: `mahasiswa.h` berisi struktur data mahasiswa serta deklarasi fungsi `inputMhs()` dan `rata2()`, `mahasiswa.cpp` berisi implementasi fungsi untuk mengisi data dan menghitung rata-rata dua nilai, dan `main.cpp` sebagai program utama yang memanggil fungsi-fungsi tersebut. Secara sederhana, program ini meminta pengguna memasukkan NIM serta dua nilai mahasiswa, lalu menampilkan hasil rata-rata nilainya, dengan konsep

ADT yang memisahkan antara definisi tipe data, implementasi fungsi, dan logika utama program agar lebih terstruktur dan mudah dikelola.

C. Unguided/Latihan

Unguided 1



```
1 #include <iostream>
2 using namespace std;
3
4 struct Mahasiswa {
5     string nama;
6     string nim;
7     float uts;
8     float uas;
9     float tugas;
10    float nilaiAkhir;
11};
12
13 float hitungNilaiAkhir(float uts, float uas, float tugas) {
14     return 0.3 * uts + 0.4 * uas + 0.3 * tugas;
15 }
16
17 int main() {
18     Mahasiswa mhs[10];
19     int x;
20
21     cout << "Masukkan jumlah mahasiswa (maks 10): ";
22     cin >> x;
23
24     for(int i = 0; i < x; i++){
25         cout << "\nData Mahasiswa ke-" << i+1 << endl;
26         cout << "Nama   : ";
27         cin >> mhs[i].nama;
28         cout << "NIM    : ";
29         cin >> mhs[i].nim;
30         cout << "Nilai UTS : ";
31         cin >> mhs[i].uts;
32         cout << "Nilai UAS : ";
33         cin >> mhs[i].uas;
34         cout << "Nilai Tugas: ";
35         cin >> mhs[i].tugas;
36
37         mhs[i].nilaiAkhir = hitungNilaiAkhir(mhs[i].uts, mhs[i].uas, mhs[i].tugas);
38     }
39
40     cout << "\n=== Data Nilai Mahasiswa ===" << endl;
41     for(int i = 0; i < x; i++){
42         cout << "\nMahasiswa ke-" << i+1 << endl;
43         cout << "Nama       : " << mhs[i].nama << endl;
44         cout << "NIM        : " << mhs[i].nim << endl;
45         cout << "Nilai Akhir : " << mhs[i].nilaiAkhir << endl;
46     }
47
48     return 0;
49 }
```

```
#include <iostream>

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 x;

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

    for(int i = 0; i < x; i++){
        cout << "\nData Mahasiswa ke-" << 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 << "\n=== Data Nilai Mahasiswa ===" << endl;

    for(int i = 0; i < x; i++){

        cout << "\nMahasiswa ke-" << i+1 << endl;

        cout << "Nama          : " << mhs[i].nama << endl;

        cout << "NIM           : " << mhs[i].nim << endl;

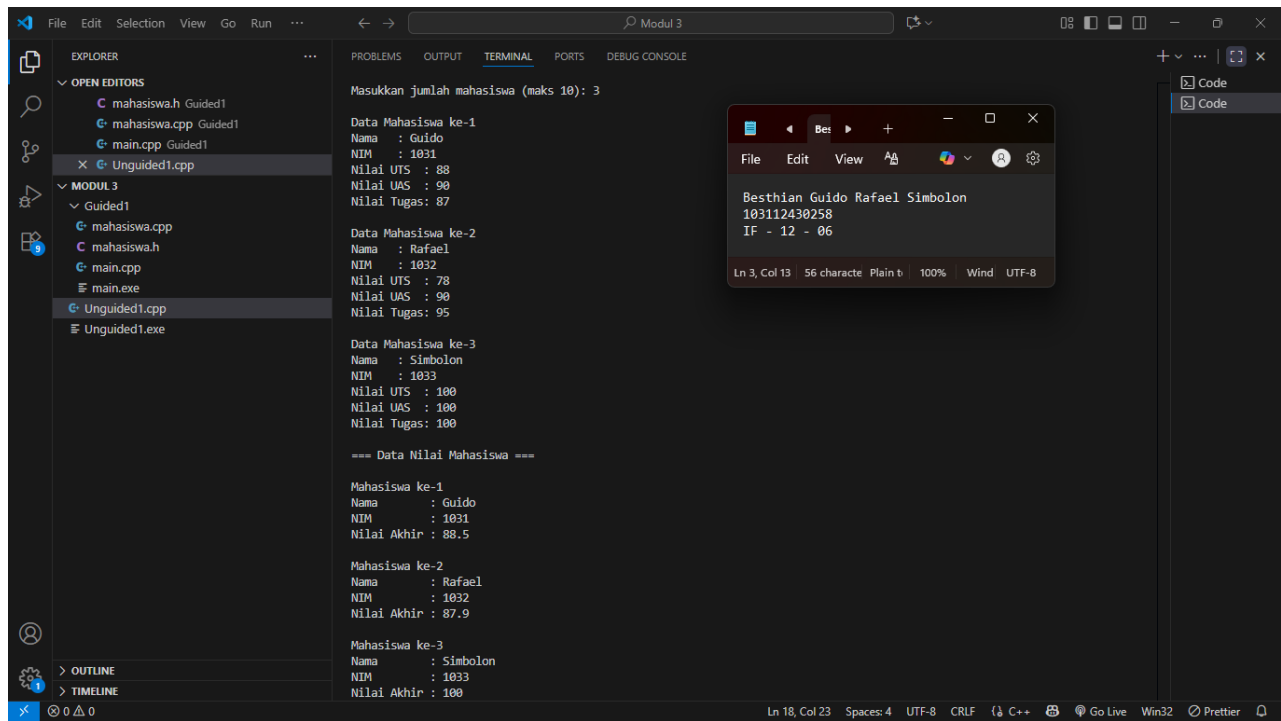
        cout << "Nilai Akhir : " << mhs[i].nilaiAkhir <<
endl;

    }

    return 0;

}
```


Screenshot Output



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

Data Mahasiswa ke-1
Nama : Guido
NIM : 1031
Nilai UTS : 88
Nilai UAS : 90
Nilai Tugas : 87

Data Mahasiswa ke-2
Nama : Rafael
NIM : 1032
Nilai UTS : 78
Nilai UAS : 90
Nilai Tugas : 95

Data Mahasiswa ke-3
Nama : Simbolon
NIM : 1033
Nilai UTS : 100
Nilai UAS : 100
Nilai Tugas : 100

=== Data Nilai Mahasiswa ===

Mahasiswa ke-1
Nama : Guido
NIM : 1031
Nilai Akhir : 88.5

Mahasiswa ke-2
Nama : Rafael
NIM : 1032
Nilai Akhir : 87.9

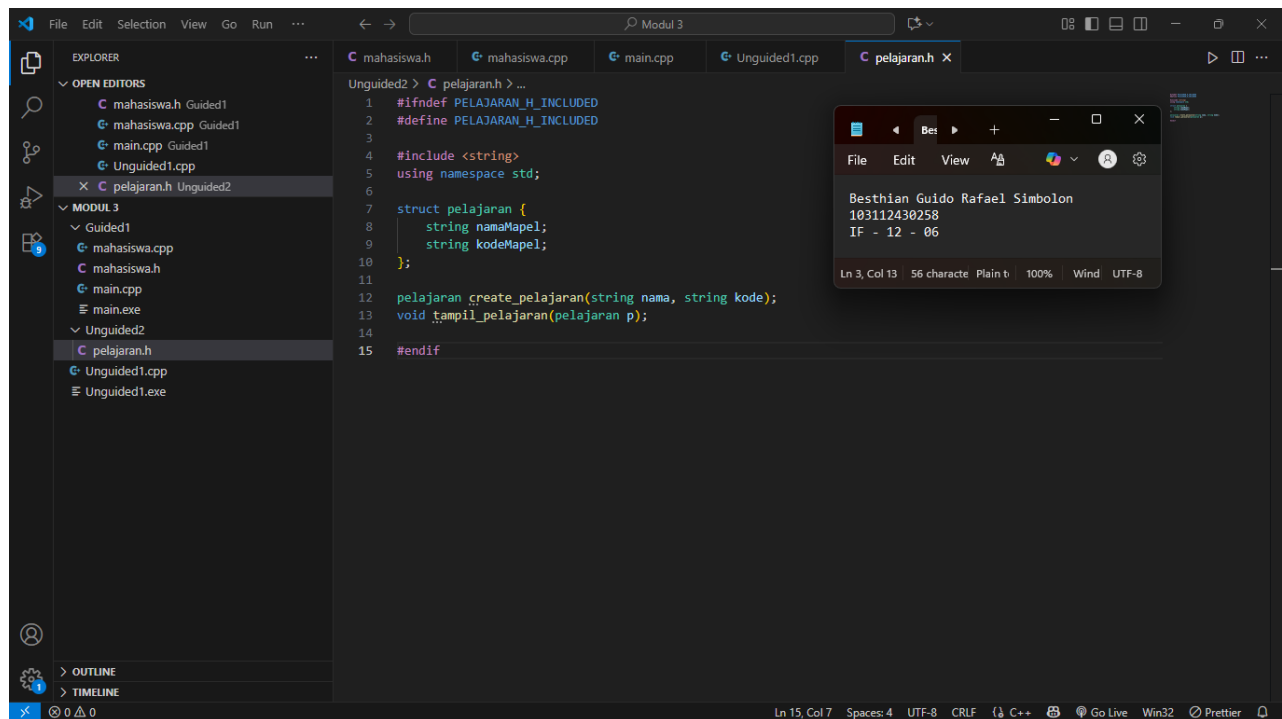
Mahasiswa ke-3
Nama : Simbolon
NIM : 1033
Nilai Akhir : 100
```

Deskripsi

Program di atas adalah contoh sederhana untuk menginput dan menghitung nilai akhir beberapa mahasiswa menggunakan konsep struktur data (struct) dalam C++. Di dalam struct Mahasiswa, terdapat beberapa atribut seperti nama, nim, uts, uas, tugas, dan nilaiAkhir yang digunakan untuk menyimpan data setiap mahasiswa. Fungsi `hitungNilaiAkhir()` digunakan untuk menghitung nilai akhir berdasarkan rumus $0.3 * uts + 0.4 * uas + 0.3 * tugas$. Pada bagian `main()`, program meminta pengguna menentukan jumlah mahasiswa (maksimal 10), lalu menginput data setiap mahasiswa satu per satu. Setelah semua data dimasukkan, program menampilkan hasil akhir berupa nama, NIM, dan nilai akhir masing-masing mahasiswa. Secara singkat, program ini membantu menghitung dan menampilkan nilai akhir mahasiswa dengan cara yang terstruktur dan mudah dipahami.

Unguided 2

Pelajaran.h



```
#ifndef PELAJARAN_H_INCLUDED
#define PELAJARAN_H_INCLUDED

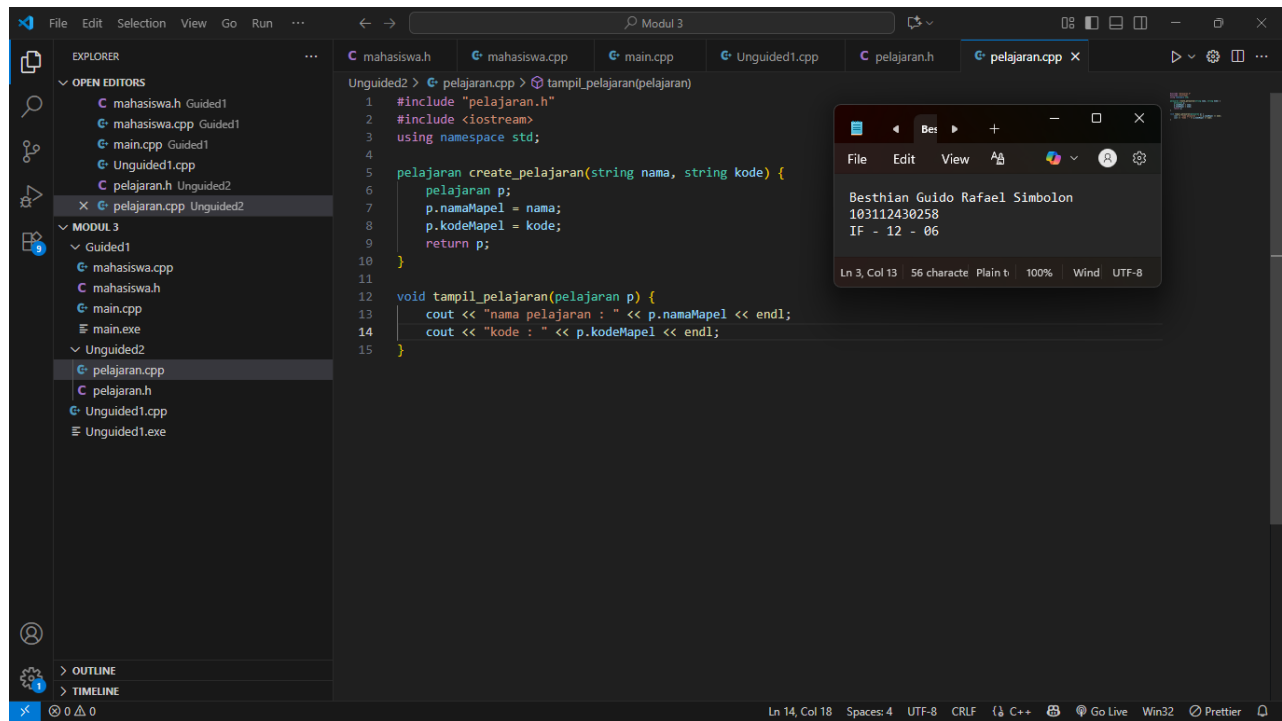
#include <string>
using namespace std;

struct pelajaran {
    string namaMapel;
    string kodeMapel;
};

pelajaran create_pelajaran(string nama, string kode);
void tampil_pelajaran(pelajaran p);

#endif
```

Pelajaran.cpp



```
#include "pelajaran.h"

#include <iostream>

using namespace std;

pelajaran
create_pelajaran(string
nama, string kode) {

    pelajaran p;

    p.namaMapel = nama;
    p.kodeMapel = kode;

    return p;

}

void
tampil_pelajaran(pelajaran
p) {
```

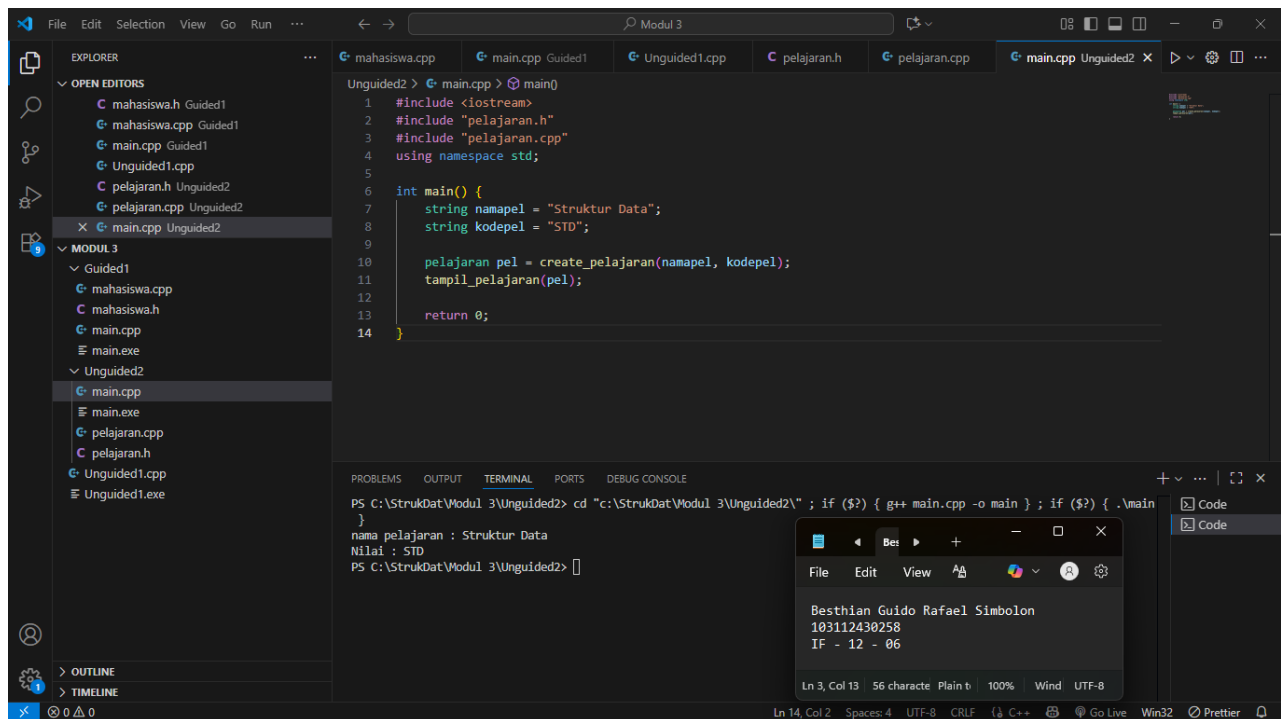
```

        cout << "nama
pelajaran : " <<
p.namaMapel << endl;

        cout << "kode : " <<
p.kodeMapel << endl;
    }

```

Main.cpp



```

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

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

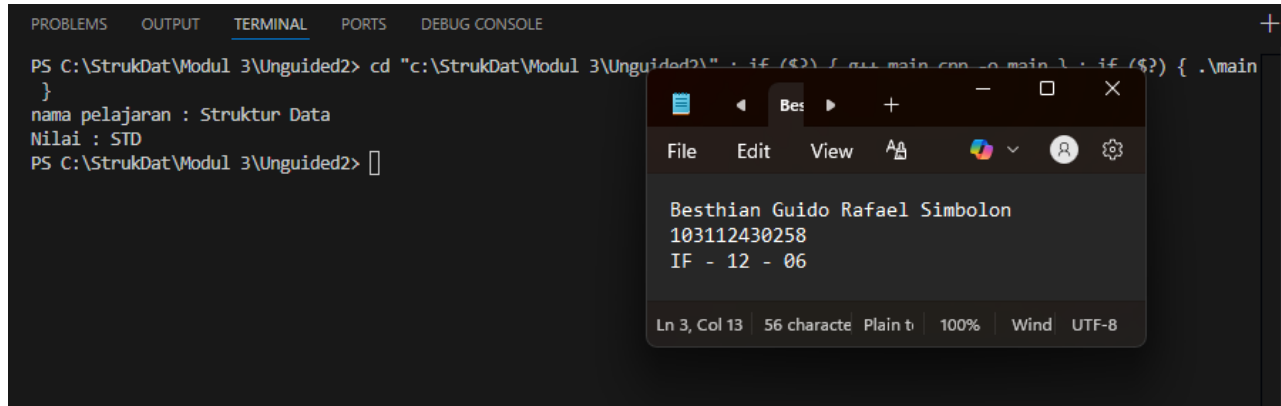
    pelajaran pel = create_pelajaran(namel, kodepel);
    tampil_pelajaran(pel);

    return 0;
}

```

```
}
```

Screenshots Output



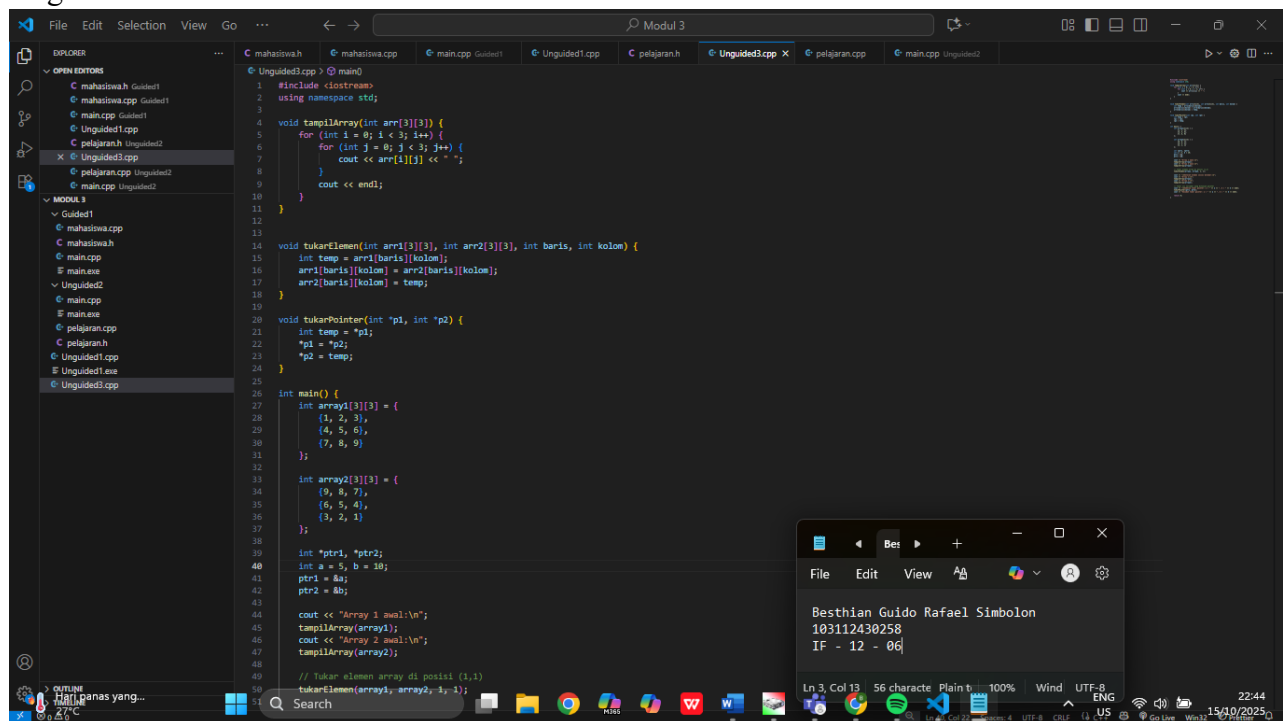
```
PS C:\StrukDat\Modul 3\Unguided2> cd "c:\StrukDat\Modul 3\Unguided2\" : if ($?) { gcc main.cpp -o main } : if ($?) { .\main
}
nama pelajaran : Struktur Data
Nilai : STD
PS C:\StrukDat\Modul 3\Unguided2> 
```

Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06

Deskripsi:

Kode di atas adalah contoh penerapan Abstract Data Type (ADT) untuk tipe data bernama pelajaran dalam bahasa C++. Program ini dibagi menjadi tiga file agar lebih terstruktur: pelajaran.h, pelajaran.cpp, dan main.cpp. Pada file pelajaran.h, terdapat deklarasi struktur pelajaran yang menyimpan dua data, yaitu namaMapel dan kodeMapel, serta deklarasi dua fungsi: create_pelajaran() untuk membuat data pelajaran dan tampil_pelajaran() untuk menampilkan data tersebut. Di file pelajaran.cpp, kedua fungsi itu diimplementasikan — create_pelajaran() berfungsi mengisi nilai atribut namaMapel dan kodeMapel, sedangkan tampil_pelajaran() menampilkan isinya ke layar. Terakhir, di main.cpp, program utama membuat satu objek pelajaran dengan nama "Struktur Data" dan kode "STD", lalu memanggil fungsi tampil_pelajaran() untuk menampilkan hasilnya. Secara sederhana, program ini menunjukkan bagaimana konsep ADT digunakan untuk memisahkan struktur data, fungsi pembuat, dan fungsi penampil agar kode lebih rapi dan mudah dipahami.

Unguided 3



```
#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 tukarElemen(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 *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 *ptr1, *ptr2;
    int a = 5, b = 10;
    ptr1 = &a;
    ptr2 = &b;

    cout << "Array 1 awal:\n";
    tampilArray(array1);
    cout << "Array 2 awal:\n";
    tampilArray(array2);
}

```

```

// Tukar elemen array di posisi (1,1)
tukarElemen(array1, array2, 1, 1);

cout << "\nSetelah elemen [1][1] ditukar:\n";
cout << "Array 1:\n";
tampilArray(array1);
cout << "Array 2:\n";
tampilArray(array2);

// Tukar isi variabel yang ditunjuk pointer
cout << "\nSebelum tukar pointer: a = " << a << ", b = " << b << endl;
tukarPointer(ptr1, ptr2);
cout << "Sesudah tukar pointer: a = " << a << ", b = " << b << endl;

return 0;
}

```

Screenhot Output

```

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
Setelah elemen [1][1] ditukar:
Array 1:
1 2 3
4 5 6
7 8 9
Array 2:
9 8 7
6 5 4
3 2 1

Sebelum tukar pointer: a = 5, b = 10
Sesudah tukar pointer: a = 10, b = 5
PS C:\StrukDat\Modul 3>

```

Deskripsi

Program di atas adalah contoh sederhana penggunaan array 2 dimensi dan pointer dalam C++. Tujuan program ini adalah untuk menampilkan isi dua buah array 3x3, menukar elemen tertentu di dalamnya, dan juga menukar nilai dua variabel menggunakan pointer. Fungsi `tampilArray()` digunakan untuk menampilkan isi array 3x3 ke layar. Fungsi `tukarElemen()` berfungsi untuk menukar elemen pada posisi tertentu (dalam contoh ini baris ke-1 dan kolom ke-1) antara dua array, sedangkan fungsi `tukarPointer()` digunakan

untuk menukar nilai dari dua variabel yang ditunjuk oleh pointer. Pada bagian main(), terdapat dua array (array1 dan array2) berukuran 3x3 yang diisi dengan angka berbeda, serta dua variabel a dan b yang masing-masing bernilai 5 dan 10. Setelah program dijalankan, hasilnya akan menunjukkan perubahan nilai elemen array setelah ditukar dan perubahan nilai variabel a serta b setelah proses pertukaran menggunakan pointer. Secara sederhana, program ini membantu memahami konsep fungsi, array 2D, dan pointer dalam pengolahan data di C++.

.

D. Kesimpulan

Abstract Data Type (ADT) membantu dalam membuat program yang lebih terstruktur, rapi, dan mudah dikelola dengan cara memisahkan antara definisi data, implementasi fungsi, dan logika utama program. Melalui contoh ADT seperti mahasiswa dan pelajaran, kita bisa melihat bagaimana data dan operasinya dikemas dalam satu kesatuan yang jelas, sehingga kode lebih modular dan mudah digunakan kembali.

Selain itu, contoh program dengan array 2D dan pointer menunjukkan penerapan konsep dasar struktur data dan manipulasi memori dalam C++. Penggunaan fungsi untuk menukar elemen array dan nilai pointer memperkuat pemahaman tentang bagaimana data dapat diakses dan diubah melalui referensi memori. Secara keseluruhan, modul ini mengajarkan pentingnya perancangan data dan fungsi secara sistematis untuk menghasilkan program yang efisien dan mudah dipahami.

.

E. Referensi

Juhana, A. (2020). *Algoritma dan Pemrograman Dasar Menggunakan Bahasa C++*. Rumah Publikasi Indonesia.

Wantoro, J., & Sukirman. (2017). *Algoritma dan Struktur Data dalam Bahasa C/C++*. Muhammadiyah University Press.

Kadir, A. (2013). *Algoritma dan Pemrograman Menggunakan C dan C++*. Andi Publisher.

Rachmawati, D., & Raharjo, B. (2016). *Pemrograman Terstruktur dengan C dan C++*. Informatika Bandung.