# LAPORAN PRAKTIKUM

# STRUKTUR DATA

## MODUL III

## *ABSTRACT DATA TYPE*


**Disusun Oleh :**

Nama : Izzah Minkhotun Fannisa

NIM : 103112400198

**Dosen**

FAHRUDIN MUKTI WIBOWO

**PROGRAM STUDI STRUKTUR DATA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**


**2025**

## A. Dasar Teori

C++ merupakan bahasa pemrograman yang pertama kali dikembangkan oleh Bjarne Stroustrop, seorang ilmuwan komputer asal Denmark. Bahasa pemrograman C++ disempurnakan dari bahasa C yang di temukan oleh Dennis Ritchie pada awal 1970-an. Lalu resmi dirilis pada tahun 1985. C++ cukup banyak digunakan karena kemampuannya dalam memberikan kecepatan eksekusi tinggi serta dukungan terhadap manajemen memori tingkat rendah.

*Abstract Data Type (ADT)* adalah konsep dalam pemrograman yang memisahkan antara definisi data dan cara penggunaannya. Dengan ADT, pengguna hanya perlu mengetahui operasi apa yang dapat dilakukan terhadap data tanpa harus memahami detail implementasinya.

Di C++, ADT bisa dibuat dengan menggunakan struct atau class yang berisi data dan fungsi-fungsi pendukung. Konsep ini membantu dalam menerapkan abstraksi dan modularisasi, sehingga membuat program lebih rapi dan mudah dikembangkan.

Dalam praktikum ini, ADT mahasiswa dibuat untuk menyimpan data seperti NIM dan nilai, serta memiliki beberapa operasi dasar seperti fungsi inputMhs() untuk memasukkan data dan fungsi rata2() untuk menghitung rata-rata nilai.

Dengan penerapan ini, terlihat bahwa ADT berperan penting dalam menyederhanakan pengelolaan data dan meningkatkan efisiensi program.

## B. Guide

### 1. Guide 1

#### a. Source Code

mahasiswa.h

```h
#ifndef MAHASISWA_H_INCLUDED
#define MAHASISWA_H_INCLUDED

struct mahasiswa {
    char nim[10];
    int nilai1;
    int nilai2;
};

void inputMhs(mahasiswa &m);
float rata2(mahasiswa m);

#endif
```

mahasiswa.cpp

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

void inputMhs(mahasiswa &m) {
```

```
    cout << "Input nama: ";
    cin >> m.nim;
    cout << "Input nilai1: ";
    cin >> m.nilai1;
    cout << "Input nilai2: ";
    cin >> m.nilai2;
}

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

main.cpp

```
#include <iostream>
#include "mahasiswa.h"

using namespace std;

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

**b. Screenshot Output**

```
PS D:\Mingguketiga> .\main.exe
Input nama: Minkho
Input nilai1: 90
Input nilai2: 100
rata-rata = 95
```

**c. Deskripsi Program**

Program tersebut menggunakan penerapan konsep Abstract Data Type (ADT). ADT yang dibuat diberi nama mahasiswa, yang berfungsi untuk menyimpan data seorang mahasiswa berupa NIM, nilai pertama, dan nilai kedua. Selain itu, ADT ini juga menyediakan dua operasi utama, yaitu fungsi untuk menginput data mahasiswa dan fungsi untuk menghitung nilai rata-rata dari dua nilai yang dimasukkan.

Struktur data mahasiswa didefinisikan dalam file mahasiswa.h, yang berisi deklarasi tipe data dan fungsi.

Implementasi atau definisi dari fungsi-fungsi tersebut ditulis pada file mahasiswa.cpp, sedangkan proses pemanggilan dan pengujian dilakukan melalui file main.cpp.

Dengan pembagian seperti ini, program menjadi modular, terstruktur, dan mudah dipelihara, karena antara spesifikasi tipe data, implementasi fungsi, dan program utama dipisahkan dengan jelas.

Secara umum, alur kerja program adalah sebagai berikut:

program utama (main.cpp) memanggil prosedur inputMhs() untuk meminta pengguna mengisi data mahasiswa (NIM, nilai 1, dan nilai 2).
Setelah data dimasukkan, fungsi rata2() akan menghitung nilai rata-rata dari dua nilai yang telah diinputkan, kemudian hasilnya ditampilkan ke layar.

Dengan adanya ADT mahasiswa, program ini menunjukkan bagaimana konsep abstraksi data dapat digunakan untuk menyembunyikan detail implementasi dari pengguna program, sehingga fokus utama berada pada apa yang dilakukan, bukan bagaimana cara melakukannya.

## C. Unguide

### 1. Unguide 1

#### a. Source Code

mahasiswa.h

```
#ifndef MAHASISWA_H_INCLUDED
#define MAHASISWA_H_INCLUDED

#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);
void inputMahasiswa(Mahasiswa &m);
void tampilkanMahasiswa(Mahasiswa m);

#endif
```

mahasiswa.cpp

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

float hitungNilaiAkhir(float uts, float uas, float tugas) {
```

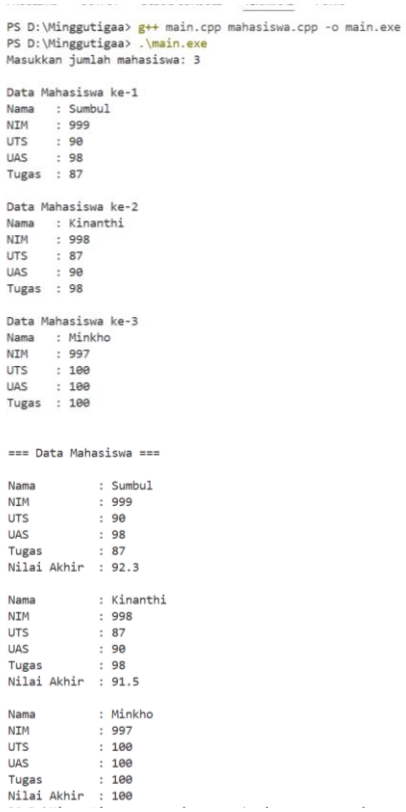
```
    return (0.3 * uts) + (0.4 * uas) + (0.3 * tugas);
}

void inputMahasiswa(Mahasiswa &m) {
    cout << "Nama    : ";
    getline(cin, m.nama);
    cout << "NIM     : ";
    getline(cin, m.nim);
    cout << "UTS     : ";
    cin >> m.uts;
    cout << "UAS     : ";
    cin >> m.uas;
    cout << "Tugas   : ";
    cin >> m.tugas;
    cin.ignore();
    m.nilaiAkhir = hitungNilaiAkhir(m.uts, m.uas, m.tugas);
}

void tampilkanMahasiswa(Mahasiswa m) {
    cout << "\nNama         : " << m.nama << endl;
    cout << "NIM          : " << m.nim << endl;
    cout << "UTS          : " << m.uts << endl;
    cout << "UAS          : " << m.uas << endl;
    cout << "Tugas        : " << m.tugas << endl;
    cout << "Nilai Akhir  : " << m.nilaiAkhir << endl;
}
```

main.cpp

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

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

    cout << "Masukkan jumlah mahasiswa: ";
    cin >> n;
    cin.ignore();

    for (int i = 0; i < n; i++) {
        cout << "\nData Mahasiswa ke-" << i + 1 << endl;
        inputMahasiswa(daftar[i]);
    }

    cout << "\n=== Data Mahasiswa ===\n";
    for (int i = 0; i < n; i++) {
```

```
        tampilkanMahasiswa(daftar[i]);
    }

    return 0;
}
```

**b. Screenshot Output**



```
PS D:\Minggutigaa> g++ main.cpp mahasiswa.cpp -o main.exe
PS D:\Minggutigaa> .\main.exe
Masukkan jumlah mahasiswa: 3

Data Mahasiswa ke-1
Nama   : Sumbul
NIM    : 999
UTS    : 90
UAS    : 98
Tugas  : 87

Data Mahasiswa ke-2
Nama   : Kinanthi
NIM    : 998
UTS    : 87
UAS    : 90
Tugas  : 98

Data Mahasiswa ke-3
Nama   : Minkho
NIM    : 997
UTS    : 100
UAS    : 100
Tugas  : 100


=== Data Mahasiswa ===

Nama        : Sumbul
NIM         : 999
UTS         : 90
UAS         : 98
Tugas       : 87
Nilai Akhir : 92.3

Nama        : Kinanthi
NIM         : 998
UTS         : 87
UAS         : 90
Tugas       : 98
Nilai Akhir : 91.5

Nama        : Minkho
NIM         : 997
UTS         : 100
UAS         : 100
Tugas       : 100
Nilai Akhir : 100
```

**c. Deskripsi Program**

Program tersebut digunakan untuk menyimpan dan menampilkan data mahasiswa sebanyak maksimal 10 orang dengan data nama, NIM, nilai UTS, nilai UAS, nilai tugas, dan nilai akhir yang diperoleh dari fungsi 0.3*uts+0.4*uas+0.3*tugas. Program ini menggunakan konsep Abstract Data Type (ADT) dengan 3 pembagian file (mahasiswa.h, mahasiswa.cpp, dan main.cpp). Cara kerja program:

1) Struct Mahasiswa digunakan untuk menyimpan semua data mahasiswa dalam satu tipe data yang terorganisir,
2) Fungsi hitungnilaiAkhir() untuk menerima tiga nilai (UTS, UAS, Tugas) dan mengembalikan nilai akhir hasil perhitungan,
3) inputMahasiswa() akan meminta user untuk menginput nama, NIM, dan Nilai-nilai mahasiswa. Setelah semua data diinputkan, nilai akhir dihitung dengan hitungNilaiAkhir() dan disimpan dalam struct,

4) tampilkanMahasiswa() akan menampilkan data yang telah disimpan di array satu persatu beserta nilai akhir yang sudah dihitung oleh fungsi,
5) Lalu, program utama (main.cpp) akan meminta user memasukkan jumlah mahasiswa (maksimal 10) dan program tersebut akan mengulangi proses input, kemudian menampilkan seluruh data mahasiswa lengkap dengan nilai akhirnya.

## 2. Unguide 2

### a. Source Code

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 pelajaran : " << pel.namaMapel << endl;
   cout << "kode pelajaran : " << pel.kodeMapel << endl;
}
```

main.cpp

```
#include <iostream>
```

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

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

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

    return 0;
}
```

**b. Screenshot Output**

```
PS D:\Minggutigaaa> .\main.exe
nama pelajaran : Struktur Data
kode pelajaran : STD
PS D:\Minggutigaaa>
```

**c. Deskripsi Program**

Program tersebut dibuat dengan menerapkan konsep Abstract Data Type (ADT). Dalam program ini, ADT bernama "pelajaran" memiliki dua atribut utama: yaitu namaMapel untuk menyimpan nama mata pelajaran dan kodeMapel untuk menyimpan kode dari mata pelajaran tersebut. Cara kerja program:

1) Tipe data pelajaran.h dibuat dengan struct yang berisi dua tipe atribut bertipe string:
   a) create_pelajaran() untuk membuat data baru.
   b) tampil_pelajaran() untuk menampilkan data pelajaran.
2) Fungsi dan prosedur pelajaran.cpp
   a) Fungsi create_pelajaran() menerima dua parameter (namapel dan kodepel), lalu membuat sebuah variabel bertipe pelajaran, mengisi kedua atributnya, dan mengembalikannya sebagai hasil.
   b) Prosedur tampil_pelajaran() digunakan untuk menampilkan nilai atribut dari objek pelajaran ke layar.
3) Fungsi main.cpp
   a) Mendeklarasikan dua variabel string, namapel dan kodepel.
   b) Variabel tersebut dimasukkan ke fungsi create_pelajaran() untuk membentuk objek baru bertipe pelajaran.
   c) Objek tersebut kemudian dikirim ke prosedur tampil_pelajaran() agar hasilnya dapat ditampilkan di layar.
4) Setelah program dijalankan, layar akan menampilkan nama dan kode pelajaran sesuai nilai yang dimasukkan ke dalam fungsi.

## 3. Unguide 3

**a. Source Code**

**array.h**

```
#ifndef ARRAY_H_INCLUDED
#define ARRAY_H_INCLUDED

#include <iostream>
using namespace std;

struct arrayInt {
    int data[3][3];
};

void isiArray(arrayInt &A);
void tampilArray(const arrayInt &A);
void tukarArray(arrayInt &A, arrayInt &B, int baris, int
kolom);
void tukarPointer(int *p1, int *p2);

#endif
```

**array.cpp**

```
#include "array.h"

void isiArray(arrayInt &A) {
    cout << "Masukkan elemen array (3x3):" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << "Elemen [" << i << "][" << j << "] = ";
            cin >> A.data[i][j];
        }
    }
}

void tampilArray(const arrayInt &A) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << A.data[i][j] << "\t";
        }
        cout << endl;
    }
    cout << endl;
}

void tukarArray(arrayInt &A, arrayInt &B, int baris, int kolom)
{
    int temp = A.data[baris][kolom];
    A.data[baris][kolom] = B.data[baris][kolom];
```

```
    B.data[baris][kolom] = temp;
}

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

**main.cpp**

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

int main() {
    arrayInt A, B;

    cout << "=== Input Array A ===" << endl;
    isiArray(A);
    cout << "=== Input Array B ===" << endl;
    isiArray(B);

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

    int baris = 1, kolom = 2;
    tukarArray(A, B, baris, kolom);

    cout << "\nArray A setelah ditukar:\n";
    tampilArray(A);
    cout << "Array B setelah ditukar:\n";
    tampilArray(B);

    // Menukar nilai variabel melalui pointer
    int x = 10, y = 20;
    cout << "Nilai sebelum ditukar: x = " << x << ", y = " << y
<< endl;
    tukarPointer(&x, &y);
    cout << "Nilai setelah ditukar:  x = " << x << ", y = " <<
y << endl;

    return 0;
}
```

## b. Screenshot Output

```
PS D:\Mingguketigaaaa> g++ main.cpp array.cpp -o main.exe
PS D:\Mingguketigaaaa> .\main.exe
=== Input Array A ===
Masukkan elemen array (3x3):
Elemen [0][0] = 7
Elemen [0][1] = 3
Elemen [0][2] = 1
Elemen [1][0] = 9
Elemen [1][1] = 5
Elemen [1][2] = 2
Elemen [2][0] = 4
Elemen [2][1] = 6
Elemen [2][2] = 8
=== Input Array B ===
Masukkan elemen array (3x3):
Elemen [0][0] = 9
Elemen [0][1] = 1
Elemen [0][2] = 2
Elemen [1][0] = 3
Elemen [1][1] = 4
Elemen [1][2] = 5
Elemen [2][0] = 5
Elemen [2][1] = 6
Elemen [2][2] = 7


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

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


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

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

Nilai sebelum ditukar: x = 10, y = 20
Nilai setelah ditukar:  x = 20, y = 10
```

## c. Deskripsi Program

Program dimulai dengan mendeklarasikan dua buah variabel bertipe arrayInt, yaitu A dan B, yang masing-masing berisi data array dua dimensi berukuran 3×3. Selanjutnya, user diminta untuk mengisi setiap elemen dari kedua array tersebut melalui pemanggilan fungsi isiArray(). Setelah kedua array terisi, program menampilkan seluruh isi elemen dari array A dan B dengan menggunakan prosedur tampilArray().

Langkah berikutnya adalah melakukan pertukaran nilai antar kedua array pada posisi tertentu yang ditentukan oleh indeks baris dan kolom. Proses ini dilakukan dengan memanggil fungsi tukarArray(A, B, baris, kolom), di mana nilai pada posisi yang sama di kedua array akan ditukar menggunakan variabel sementara. Setelah pertukaran dilakukan, program kembali menampilkan isi kedua array untuk menunjukkan perubahan yang terjadi.

Selain itu, program juga menguji fungsi tukarPointer(), yaitu fungsi yang digunakan untuk menukar nilai dua variabel integer melalui pointer. Dua variabel, misalnya x dan y, dideklarasikan dan masing-masing ditunjuk oleh pointer &x dan &y. Fungsi tukarPointer() kemudian dipanggil untuk menukar

nilai kedua variabel tersebut secara langsung di memori. Setelah proses selesai, program menampilkan nilai variabel sebelum dan sesudah pertukaran.

Secara keseluruhan, algoritma program ini menunjukkan penerapan prinsip abstraksi data dan modularisasi fungsi, di mana setiap operasi terhadap data array dan pointer dikemas ke dalam fungsi terpisah sehingga program menjadi lebih terstruktur, mudah dipahami, dan efisien dalam pengelolaan data.

## D. Kesimpulan

Dari hasil pembuatan dan pengujian program ADT mahasiswa, saya dapat menyimpulkan bahwa penerapan konsep Abstract Data Type (ADT) sangat membantu dalam memisahkan antara definisi tipe data, implementasi fungsi, dan program utama. Dengan membagi program ke dalam beberapa file, yaitu mahasiswa.h, mahasiswa.cpp, dan main.cpp, saya menjadi lebih memahami bagaimana sebuah program dapat dibuat secara modular dan terstruktur sehingga lebih mudah dibaca, diperbaiki, serta dikembangkan.

Melalui fungsi inputMhs() dan rata2(), saya juga belajar bagaimana sebuah ADT dapat memiliki operasi dasar yang bekerja secara terpisah tetapi tetap terhubung dalam satu kesatuan tipe data. Dari praktikum ini, saya memahami bahwa penggunaan ADT tidak hanya mempermudah proses pengelolaan data, tetapi juga membantu saya berpikir lebih sistematis dalam membangun program sesuai prinsip pemrograman terstruktur dan abstraksi data.

## E. Referensi


GeeksforGeeks. (2023). Abstract Data Types in C++. Diakses pada 16 Oktober 2025, dari https://www.geeksforgeeks.org/abstract-data-types-in-c/

Tutorialspoint. (2024). C++ Data Abstraction. Diakses pada 16 Oktober 2025, dari https://www.tutorialspoint.com/cplusplus/cpp_data_abstraction.htm

Wibowo, A. (2022). Pemrograman C++ Dasar dan Lanjut. Yogyakarta: Andi Publisher.

Munir, R. (2019). Algoritma dan Pemrograman dalam Bahasa C. Bandung: Informatika Bandung.

Wirth, N. (1976). Algorithms + Data Structures = Programs. Englewood Cliffs, NJ: Prentice-Hall.