

LAPORAN PRAKTIKUM

Modul 3

Abstract Data Type (ADT)



Disusun Oleh:

Ryan Gabriel Togar Simamora (2311104045)

Kelas : SE0702

Dosen :

Wahyu Andi Saputra

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. Tujuan

Memahami konsep Abstract Data Type (ADT) dan penggunaannya dalam pemrograman.

II. Landasan Teori

ADT (Abstract Data Type) adalah konsep yang menggambarkan tipe data dan sekumpulan operasi yang dapat dilakukan terhadap tipe data tersebut, tanpa memperhatikan bagaimana implementasinya secara internal. ADT memberikan abstraksi yang memisahkan detail implementasi dari cara penggunaannya, sehingga fokus pengguna lebih pada fungsi dan operasional yang disediakan daripada cara kerja di baliknya.

Dalam ADT, kita mendefinisikan tipe data yang mencakup struktur data, serta operasi dasar (primitif) yang dapat dilakukan terhadap data tersebut. Operasi ini bisa berupa pembentukan, pengubahan, penghapusan, serta interaksi dengan data melalui input dan output. ADT juga bisa mencakup aturan-aturan atau kondisi tertentu yang harus dipenuhi, seperti invarian dan aksioma yang mengatur bagaimana tipe data itu berfungsi dalam berbagai situasi.

Dalam konteks C++, tipe data di dalam ADT sering diterjemahkan menggunakan struktur (struct) atau kelas (class). Operasi primitif diimplementasikan melalui fungsi atau prosedur yang berperan sebagai konstruktor, selektor, atau prosedur pengubah, tergantung pada fungsinya.

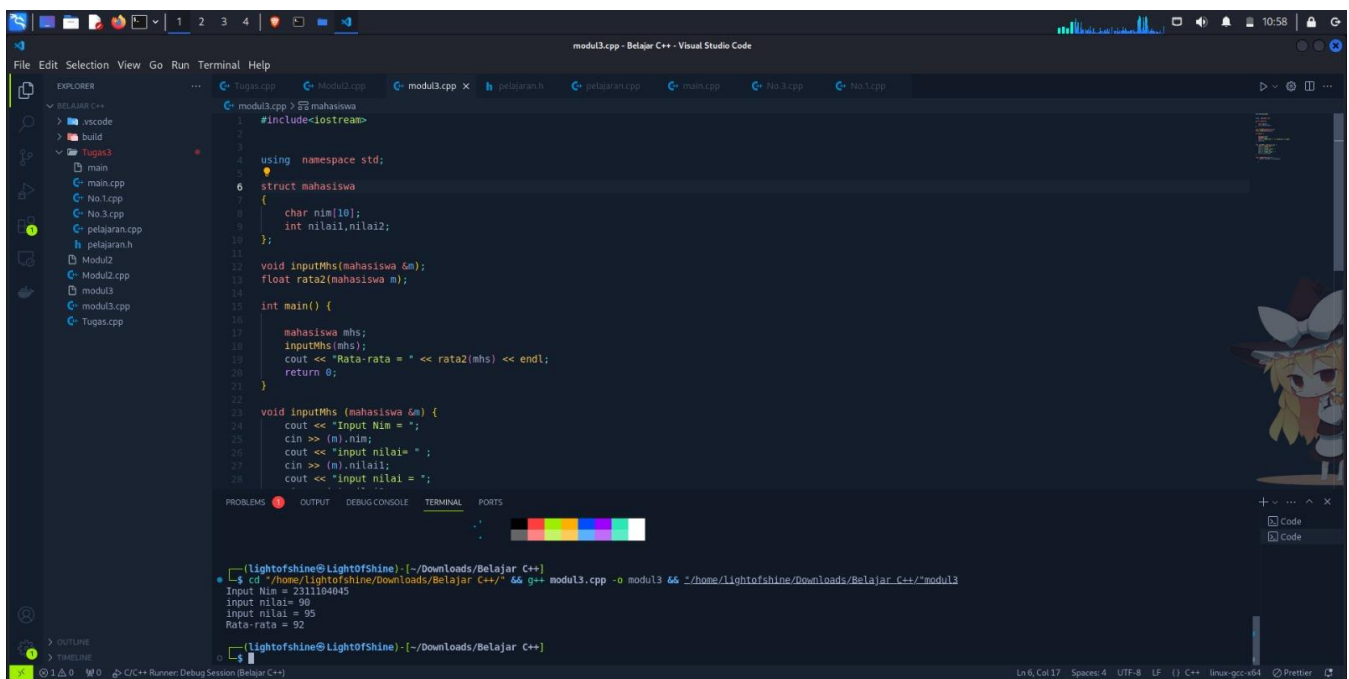
Ada beberapa jenis operasi yang umum dalam ADT, antara lain:

- ❖ Konstruktor/Kreator: Operasi yang bertanggung jawab membentuk atau menginisialisasi objek dari tipe data tersebut.
- ❖ Selektor: Operasi untuk mengakses elemen-elemen atau komponen-komponen dari tipe data tersebut.
- ❖ Pengubah: Operasi untuk memodifikasi nilai elemen atau komponen dari objek bertipe ADT.
- ❖ Validator: Digunakan untuk memeriksa apakah data sesuai dengan aturan-aturan yang telah ditetapkan.
- ❖ Destruktor/Dealokator: Operasi yang digunakan untuk membersihkan atau menghapus objek yang telah diciptakan, serta melepaskan memori yang digunakan.
- ❖ Input/Output: Operasi yang menangani antarmuka untuk membaca dan menulis data dari atau ke perangkat input/output.
- ❖ Operator relasional: Operasi yang memungkinkan perbandingan antar objek, seperti membandingkan apakah objek lebih besar, lebih kecil, atau sama.
- ❖ Aritmatika: Operasi yang mendukung perhitungan pada tipe data tertentu, terutama untuk data numerik.
- ❖ Konversi: Operasi yang mengubah tipe data dari ADT ke tipe dasar atau sebaliknya.

Dalam implementasinya, ADT biasanya dipisahkan ke dalam dua bagian utama:

- ❖ Spesifikasi/Definisi Tipe dan Operasi: Ini biasanya ditulis dalam file header (.h), yang mencakup deklarasi tipe data dan deskripsi dari operasi-operasi yang berlaku. Spesifikasi ini hanya menyatakan apa yang akan dilakukan oleh operasi, tanpa menunjukkan bagaimana operasi tersebut diimplementasikan.
- ❖ Implementasi Primitif: Bagian ini ditulis dalam file terpisah (.cpp) yang berisi kode program aktual untuk mewujudkan operasi-operasi yang telah didefinisikan. Di sinilah detail dari bagaimana operasi bekerja secara internal dijelaskan.

Contoh :



```
modul3.cpp > 83 mahasiswa
1 #include <iostream>
2
3
4 using namespace std;
5
6 struct mahasiswa
7 {
8     char nim[10];
9     int nilai1, nilai2;
10 };
11
12 void inputMhs(mahasiswa &m);
13 float rata2(mahasiswa m);
14
15 int main() {
16
17     mahasiswa mhs;
18     inputMhs(mhs);
19     cout << "Rata-rata = " << rata2(mhs) << endl;
20     return 0;
21 }
22
23 void inputMhs (mahasiswa &m) {
24     cout << "Input Nim = ";
25     cin >> (m).nim;
26     cout << "input nilai1 = ";
27     cin >> (m).nilai1;
28     cout << "input nilai2 = ";
29     cin >> (m).nilai2;
30 }
```

```
lightofshine@lightofshine:~/Downloads/Belajar C++$ cd "/home/lightofshine/Downloads/Belajar C++/" && g++ modul3.cpp -o modul3 && ./modul3
Input Nim = 2311104045
Input nilai1 = 90
Input nilai2 = 95
Rata-rata = 92
lightofshine@lightofshine:~/Downloads/Belajar C++$
```

Untuk Source codenya lebih jelas dibawah ini :

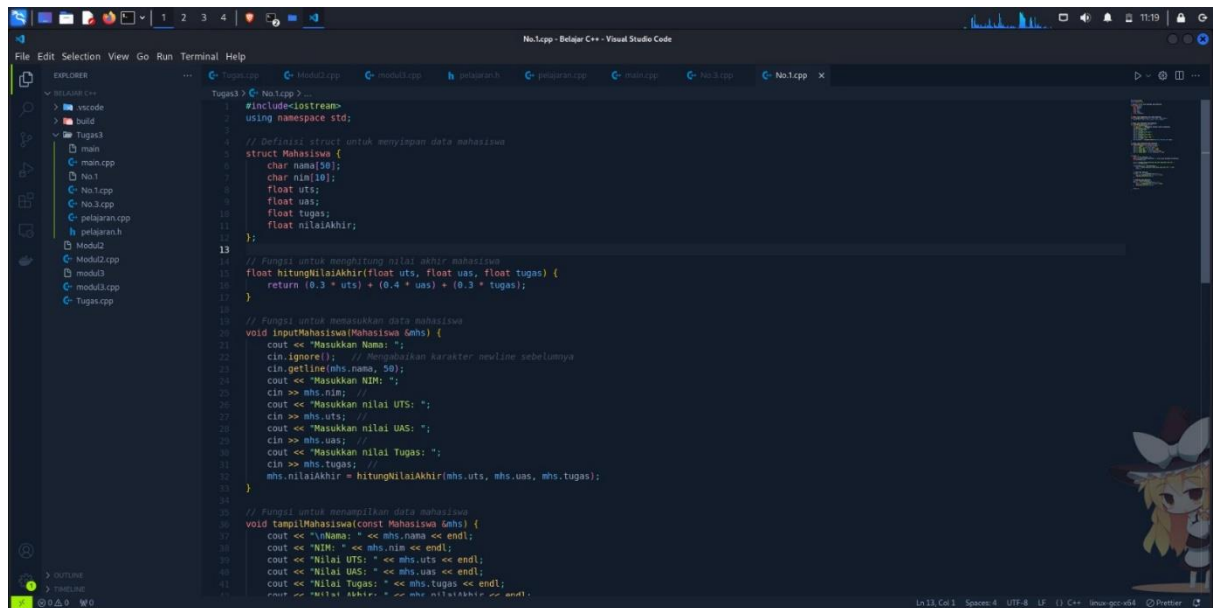
```
1  #include<iostream>
2
3
4  using namespace std;
5
6  struct mahasiswa
7  {
8      char nim[10];
9      int nilail,nilai2;
10 };
11
12 void inputMhs(mahasiswa &m);
13 float rata2(mahasiswa m);
14
15 int main() {
16
17     mahasiswa mhs;
18     inputMhs(mhs);
19     cout << "Rata-rata = " << rata2(mhs) << endl;
20     return 0;
21 }
22
23 void inputMhs (mahasiswa &m) {
24     cout << "Input Nim = ";
25     cin >> (m).nim;
26     cout << "input nilai= " ;
27     cin >> (m).nilail;
28     cout << "input nilai = ";
29     cin >> (m).nilai2;
30 }
31
32 float rata2(mahasiswa m) {
33     return( m.nilail + m.nilai2)/2;
34 }
35
36
```

Dengan ADT, kita dapat membuat program yang lebih modular dan mudah dipahami, karena kita bisa memisahkan antara definisi tipe data dan operasionalnya, sehingga perubahan dalam implementasi internal tidak akan mempengaruhi penggunaan di luar modul tersebut.

III. Unguided

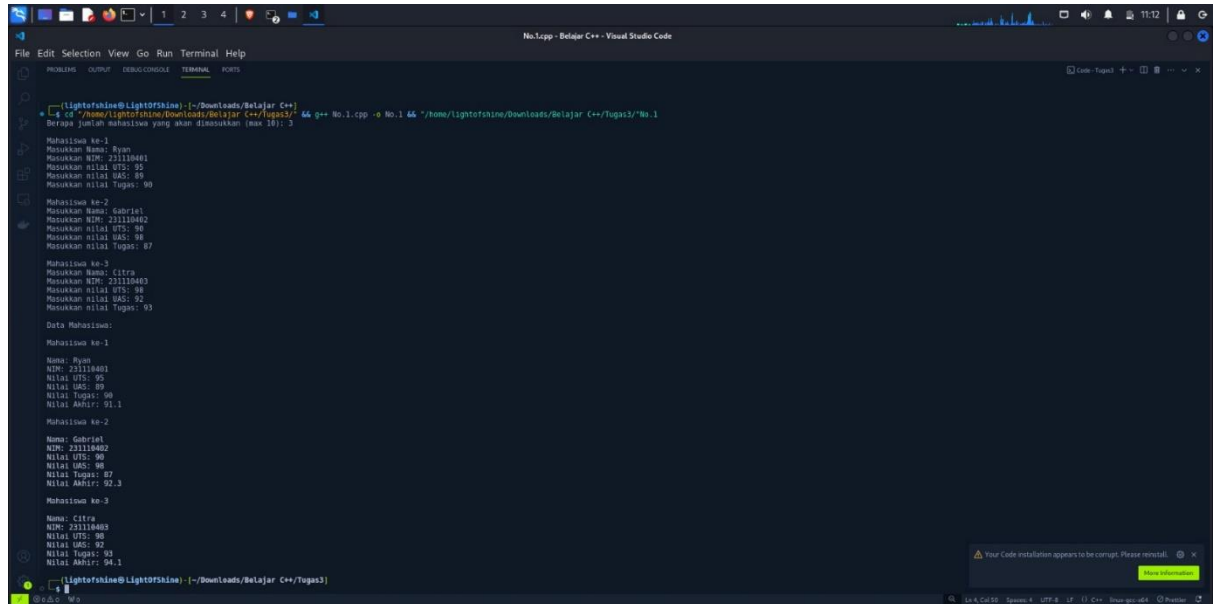
1. Buat program yang dapat menyimpan data mahasiswa (max. 10) ke dalam sebuah *array* dengan field nama, nim, uts, uas, tugas, dan nilai akhir. Nilai akhir diperoleh dari FUNGSI dengan rumus $0.3*uts+0.4*uas+0.3*tugas$.

Jawab :



```
1 #include<iostream>
2 using namespace std;
3
4 // Definisi struct untuk menyimpan data mahasiswa
5 struct Mahasiswa {
6     char nama[50];
7     char nim[10];
8     float uts;
9     float uas;
10    float tugas;
11    float nilaiAkhir;
12 };
13
14 // Fungsi untuk menghitung nilai akhir mahasiswa
15 float hitungNilaiAkhir(float uts, float uas, float tugas) {
16     return (0.3 * uts) + (0.4 * uas) + (0.3 * tugas);
17 }
18
19 // Fungsi untuk memasukkan data mahasiswa
20 void inputMahasiswa(Mahasiswa &mhs) {
21     cout << "Masukkan Nama: ";
22     cin.ignore(); // Mengabaikan karakter newline sebelumnya
23     cin.getline(mhs.nama, 50);
24     cout << "Masukkan NIM: ";
25     cin >> mhs.nim; //
26     cout << "Masukkan nilai UTS: ";
27     cin >> mhs.uts; //
28     cout << "Masukkan nilai UAS: ";
29     cin >> mhs.uas; //
30     cout << "Masukkan nilai Tugas: ";
31     cin >> mhs.tugas; //
32     mhs.nilaiAkhir = hitungNilaiAkhir(mhs.uts, mhs.uas, mhs.tugas);
33 }
34
35 // Fungsi untuk menampilkan data mahasiswa
36 void tampilMahasiswa(const Mahasiswa &mhs) {
37     cout << "Nama: " << mhs.nama << endl;
38     cout << "NIM: " << mhs.nim << endl;
39     cout << "Nilai UTS: " << mhs.uts << endl;
40     cout << "Nilai UAS: " << mhs.uas << endl;
41     cout << "Nilai Tugas: " << mhs.tugas << endl;
42     cout << "Nilai Akhir: " << mhs.nilaiAkhir << endl;
43 }
```

Output :



```
Lightofshin@Lightofshin:~/Downloads/Belajar C++/Tugas3$ g++ No.1.cpp -o No.1.exe
Berapa jumlah mahasiswa yang akan dimasukkan (max 10): 3

Mahasiswa ke-1
Masukkan Nama: Ryan
Masukkan NIM: 23110401
Masukkan nilai UTS: 95
Masukkan nilai UAS: 98
Masukkan nilai Tugas: 90

Mahasiswa ke-2
Masukkan Nama: Gabriel
Masukkan NIM: 23110402
Masukkan nilai UTS: 98
Masukkan nilai UAS: 98
Masukkan nilai Tugas: 87

Mahasiswa ke-3
Masukkan Nama: Citra
Masukkan NIM: 23110403
Masukkan nilai UTS: 98
Masukkan nilai UAS: 92
Masukkan nilai Tugas: 93

Data Mahasiswa:
Mahasiswa ke-1
Nama: Ryan
NIM: 23110401
Nilai UTS: 95
Nilai UAS: 98
Nilai Tugas: 90
Nilai Akhir: 91.1
Mahasiswa ke-2
Nama: Gabriel
NIM: 23110402
Nilai UTS: 98
Nilai UAS: 98
Nilai Tugas: 87
Nilai Akhir: 92.3
Mahasiswa ke-3
Nama: Citra
NIM: 23110403
Nilai UTS: 98
Nilai UAS: 92
Nilai Tugas: 93
Nilai Akhir: 94.1
```

Untuk Source codenya lebih jelas dibawah ini :

```
1  #include<iostream>
2  using namespace std;
3
4  // Definisi struct untuk menyimpan data mahasiswa
5  struct Mahasiswa {
6      char nama[50];
7      char nim[10];
8      float uts;
9      float uas;
10     float tugas;
11     float nilaiAkhir;
12 };
13
14 // Fungsi untuk menghitung nilai akhir mahasiswa
15 float hitungNilaiAkhir(float uts, float uas, float tugas) {
16     return (0.3 * uts) + (0.4 * uas) + (0.3 * tugas);
17 }
18
19 // Fungsi untuk memasukkan data mahasiswa
20 void inputMahasiswa(Mahasiswa &mhs) {
21     cout << "Masukkan Nama: ";
22     cin.ignore(); // Mengabaikan karakter newline sebelumnya
23     cin.getline(mhs.nama, 50);
24     cout << "Masukkan NIM: ";
25     cin >> mhs.nim; //
26     cout << "Masukkan nilai UTS: ";
27     cin >> mhs.uts; //
28     cout << "Masukkan nilai UAS: ";
29     cin >> mhs.uas; //
30     cout << "Masukkan nilai Tugas: ";
31     cin >> mhs.tugas; //
32     mhs.nilaiAkhir = hitungNilaiAkhir(mhs.uts, mhs.uas, mhs.tugas);
33 }
34
35 // Fungsi untuk menampilkan data mahasiswa
36 void tampilMahasiswa(const Mahasiswa &mhs) {
37     cout << "\nNama: " << mhs.nama << endl;
38     cout << "NIM: " << mhs.nim << endl;
39     cout << "Nilai UTS: " << mhs.uts << endl;
40     cout << "Nilai UAS: " << mhs.uas << endl;
41     cout << "Nilai Tugas: " << mhs.tugas << endl;
42     cout << "Nilai Akhir: " << mhs.nilaiAkhir << endl;
43 }
44
45 int main() {
46     const int MAX_MAHASISWA = 10;
47     Mahasiswa mhsArray[MAX_MAHASISWA]; // Array untuk menyimpan 10 mahasiswa
48     int jumlahMahasiswa;
49
50     cout << "Berapa jumlah mahasiswa yang akan dimasukkan (max 10): ";
51     cin >> jumlahMahasiswa;
52
53     if(jumlahMahasiswa > MAX_MAHASISWA) {
54         cout << "Jumlah mahasiswa tidak boleh lebih dari 10!" << endl;
55         return 1;
56     }
57
58     // Input data mahasiswa
59     for(int i = 0; i < jumlahMahasiswa; i++) {
60         cout << "\nMahasiswa ke-" << (i+1) << endl;
61         inputMahasiswa(mhsArray[i]);
62     }
63
64     // Tampilkan data mahasiswa
65     cout << "\nData Mahasiswa:\n";
66     for(int i = 0; i < jumlahMahasiswa; i++) {
67         cout << "\nMahasiswa ke-" << (i+1) << endl;
68         tampilMahasiswa(mhsArray[i]);
69     }
70
71     return 0;
72 }
```

2. Buatlah ADT pelajaran sebagai berikut di dalam *file* “pelajaran.h”:

```
tipe 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);
}
```

Gambar 3-1 Main.cpp pe

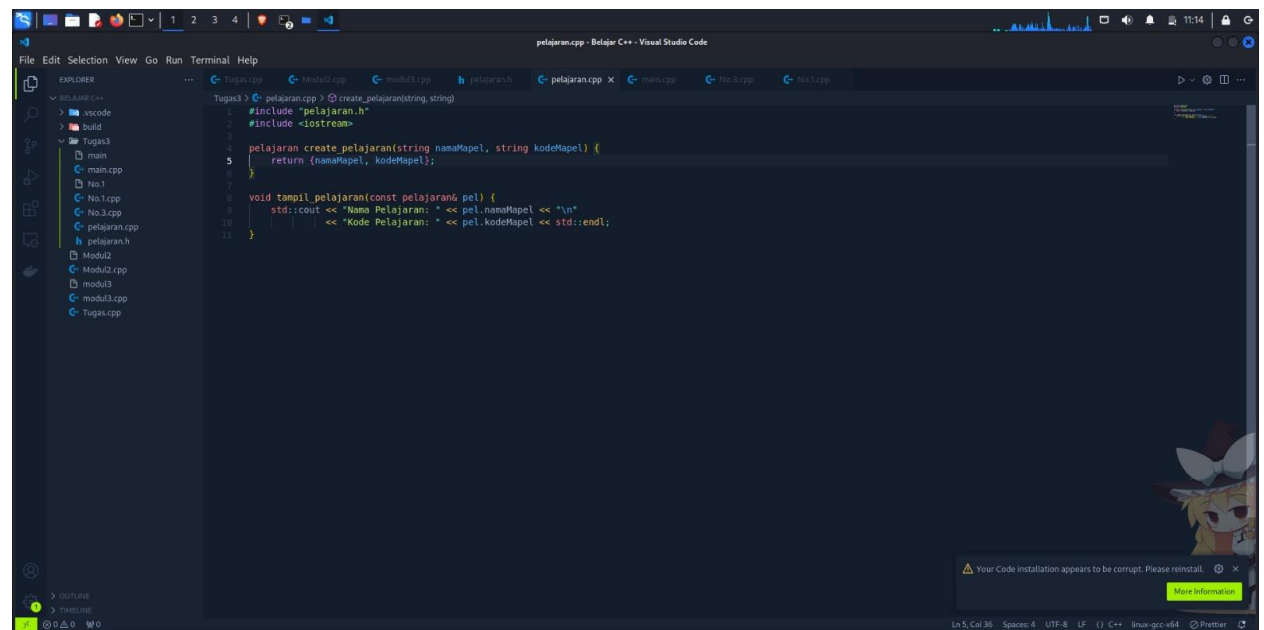
Contoh *output* hasil:

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

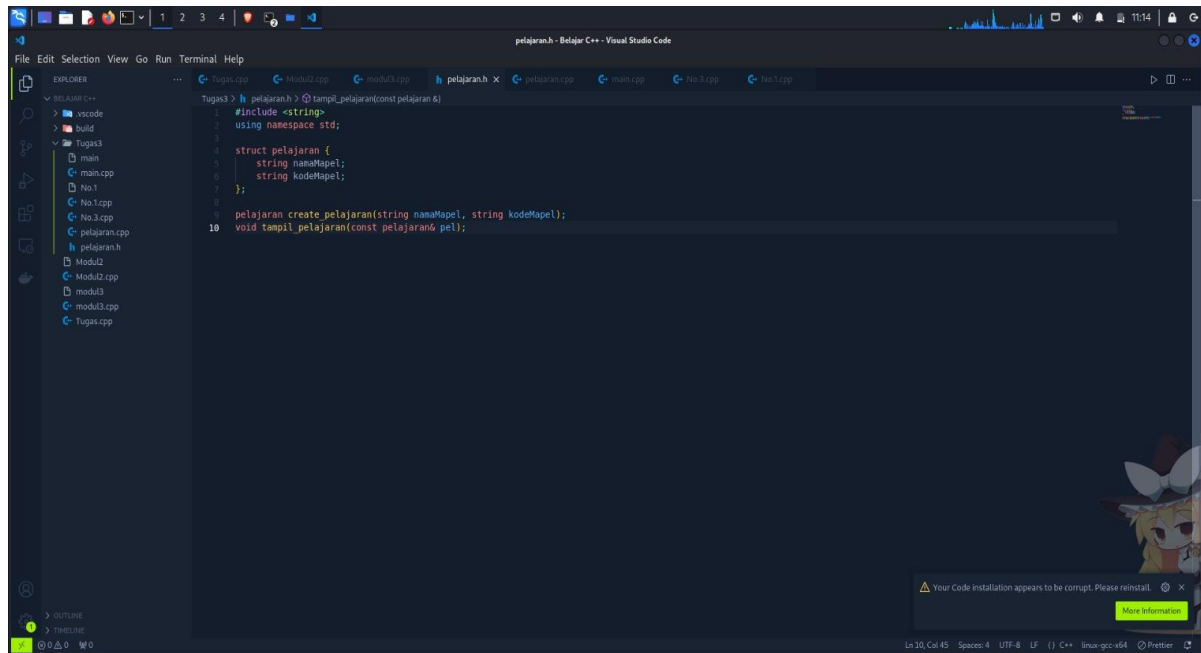
Gambar 3-2 output pelajaran

Jawab :

File pelajaran.cpp

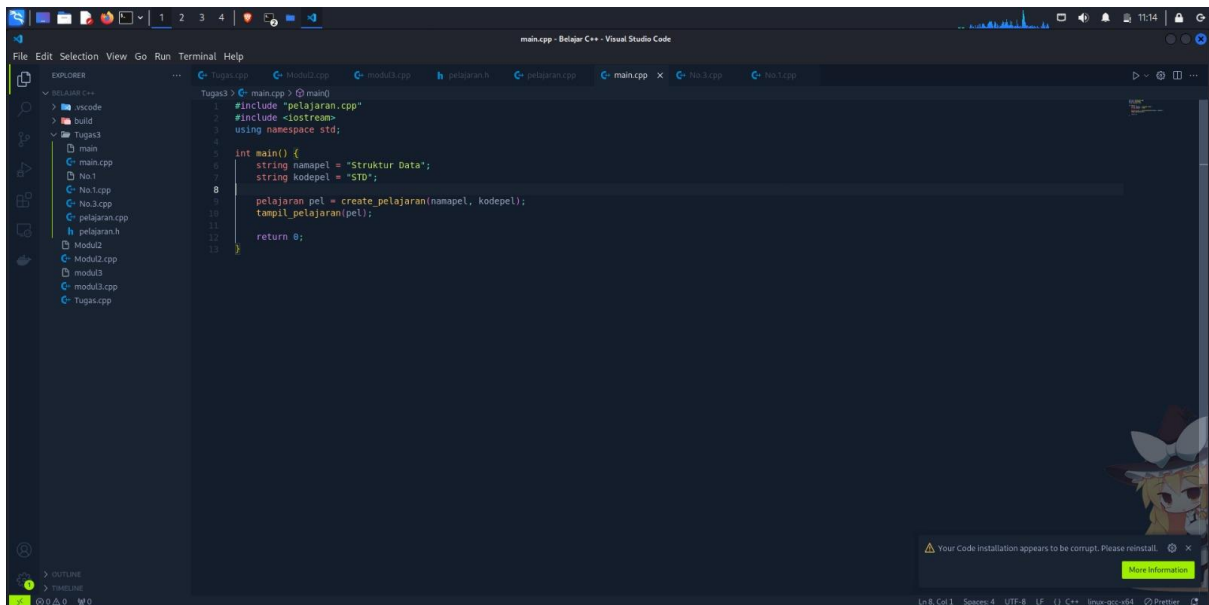


File pelajaran.h



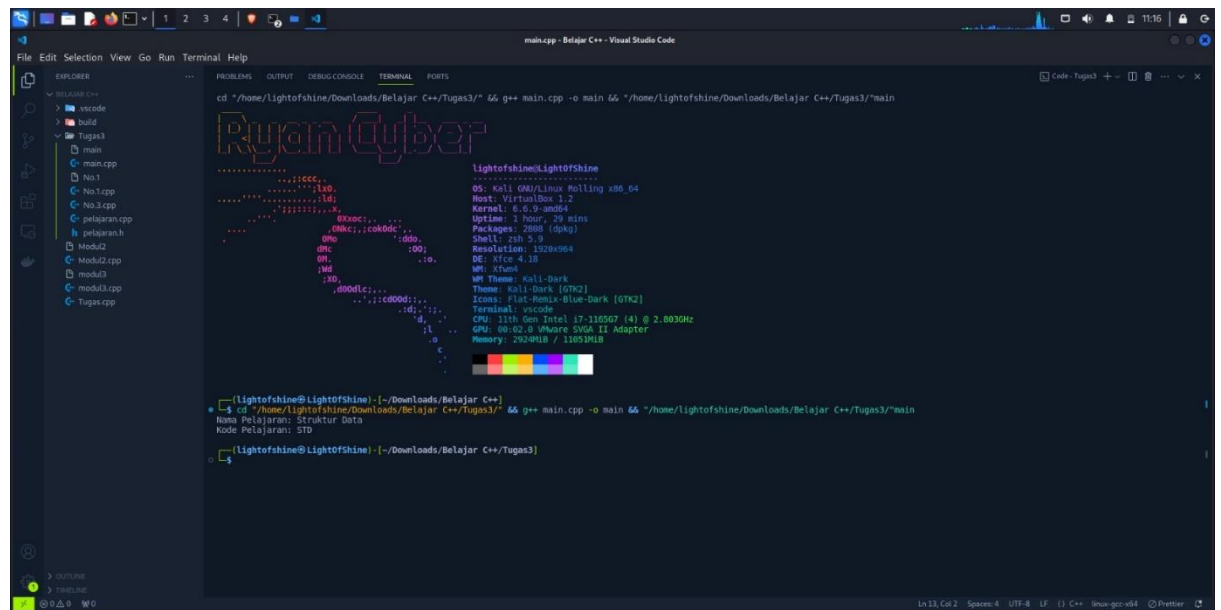
```
1 #ifndef PELAJARAN_H
2 #define PELAJARAN_H
3
4 struct pelajaran {
5     string namaMapel;
6     string kodeMapel;
7 };
8
9 pelajaran create_pelajaran(string namaMapel, string kodeMapel);
10 void tampil_pelajaran(const pelajaran& pel);
11
12 #endif
```

File main.cpp



```
1 #include "pelajaran.h"
2 #include <iostream>
3 using namespace std;
4
5 int main() {
6     string namaMapel = "Struktur Data";
7     string kodeMapel = "STD";
8
9     pelajaran pel = create_pelajaran(namaMapel, kodeMapel);
10    tampil_pelajaran(pel);
11
12    return 0;
13 }
```


Output :



```
cd "/home/Lightofshine/Downloads/Belajar C++/Tugas3/" && g++ main.cpp -o main && "/home/Lightofshine/Downloads/Belajar C++/Tugas3/main"

Ryan Cyber

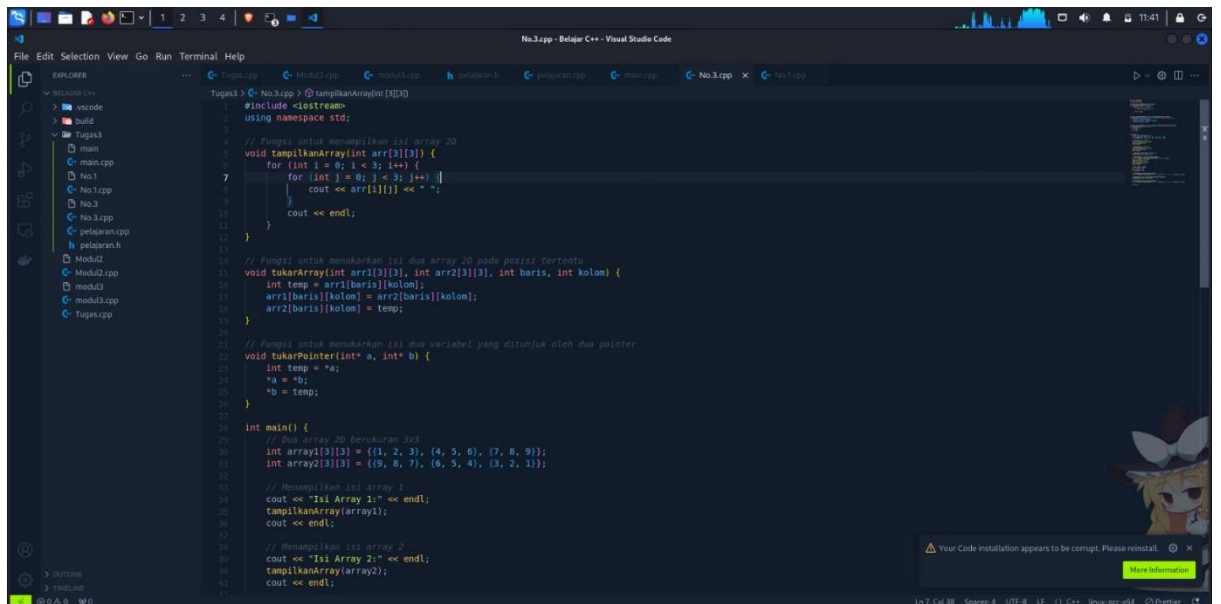
Lightofshine@Lightofshine
-----
OS: Kali OM/Linux Rolling x86_64
Host: VirtualBox 1.2
Kernel: 6.8.9-amd64
Uptime: 3 hour, 29 mins
Packages: 2088 (dpkg)
Shell: zsh 5.9
Resolution: 1920x964
DE: Xfce 4.18
WM: Xfce
WM Theme: Kali-Dark
Theme: Kali-Dark [GTK2]
Icons: Flat-Remix-Blue-Dark [GTK2]
Terminal: vscode
CPU: 11th Gen Intel i7-116507 (4) @ 2.803GHz
GPU: 00:02.0 VMware SVGA II Adapter
Memory: 2924MB / 1165MB

[Lightofshine@Lightofshine]~/Downloads/Belajar C++
$ cd "/home/Lightofshine/Downloads/Belajar C++/Tugas3/" && g++ main.cpp -o main && "/home/Lightofshine/Downloads/Belajar C++/Tugas3/main"
Name Pelajaran: Struktur Data
Kode Pelajaran: STD
[Lightofshine@Lightofshine]~/Downloads/Belajar C++/Tugas3]
```

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

Jawab :



```
No.3.cpp - Belajar C++ - Visual Studio Code

Tugas3 > C: No.3.cpp > @ tampilanArray(int [3][3])
1 #include <iostream>
2 using namespace std;
3
4 // Fungsi untuk menampilkan isi array 2D
5 void tampilanArray(int arr[3][3]) {
6     for (int i = 0; i < 3; i++) {
7         for (int j = 0; j < 3; j++) {
8             cout << arr[i][j] << " ";
9         }
10        cout << endl;
11    }
12 }
13
14 // Fungsi untuk menukarkan isi dua array 2D pada posisi tertentu
15 void tukarArray(int arr1[3][3], int arr2[3][3], int baris, int kolom) {
16     int temp = arr1[baris][kolom];
17     arr1[baris][kolom] = arr2[baris][kolom];
18     arr2[baris][kolom] = temp;
19 }
20
21 // Fungsi untuk menukarkan isi dua variabel yang ditunjuk oleh dua pointer
22 void tukarPointer(int* a, int* b) {
23     int temp = *a;
24     *a = *b;
25     *b = temp;
26 }
27
28 int main() {
29     // Dua array 2D berukuran 3x3
30     int array1[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
31     int array2[3][3] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
32
33     // Menampilkan isi array 1
34     cout << "Isi Array 1:" << endl;
35     tampilanArray(array1);
36     cout << endl;
37
38     // Menampilkan isi array 2
39     cout << "Isi Array 2:" << endl;
40     tampilanArray(array2);
41     cout << endl;
42 }
```

Untuk Source codenya lebih jelasnya dibawah ini :

```
1  #include <iostream>
2  using namespace std;
3
4  // Fungsi untuk menampilkan isi array 2D
5  void tampilkanArray(int arr[3][3]) {
6      for (int i = 0; i < 3; i++) {
7          for (int j = 0; j < 3; j++) {
8              cout << arr[i][j] << " ";
9          }
10         cout << endl;
11     }
12 }
13
14 // Fungsi untuk menukarkan isi dua array 2D pada posisi tertentu
15 void tukarArray(int arr1[3][3], int arr2[3][3], int baris, int kolom) {
16     int temp = arr1[baris][kolom];
17     arr1[baris][kolom] = arr2[baris][kolom];
18     arr2[baris][kolom] = temp;
19 }
20
21 // Fungsi untuk menukarkan isi dua variabel yang ditunjuk oleh dua pointer
22 void tukarPointer(int* a, int* b) {
23     int temp = *a;
24     *a = *b;
25     *b = temp;
26 }
27
28 int main() {
29     // Dua array 2D berukuran 3x3
30     int array1[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
31     int array2[3][3] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
32
33     // Menampilkan isi array 1
34     cout << "Isi Array 1:" << endl;
35     tampilkanArray(array1);
36     cout << endl;
37
38     // Menampilkan isi array 2
39     cout << "Isi Array 2:" << endl;
40     tampilkanArray(array2);
41     cout << endl;
42
43     // Menukarkan elemen pada posisi (1, 1)
44     tukarArray(array1, array2, 1, 1);
45
46     // Menampilkan isi array setelah penukaran
47     cout << "Setelah Tukar Elemen pada Posisi (1, 1):" << endl;
48     cout << "Array 1:" << endl;
49     tampilkanArray(array1);
50     cout << "Array 2:" << endl;
51     tampilkanArray(array2);
52     cout << endl;
53
54     // Dua pointer integer
55     int a = 10, b = 20;
56     int* pointer1 = &a;
57     int* pointer2 = &b;
58
59     // Menampilkan nilai sebelum penukaran
60     cout << "Sebelum Tukar Pointer: a = " << *pointer1 << ", b = " << *pointer2 << endl;
61
62     // Menukarkan nilai yang ditunjuk oleh ptrA dan ptrB
63     tukarPointer(pointer1, pointer2);
64
65     // Menampilkan nilai setelah penukaran
66     cout << "Setelah Tukar Pointer: a = " << *pointer1 << ", b = " << *pointer2 << endl;
67
68     return 0;
69 }
```

Output :

The image shows a Visual Studio Code editor window with a C++ project named 'No.3.cpp - Belajar C++'. The code is a program to swap elements in an array. It includes a header file 'swap.h' and a main function that tests the swap function with two arrays. The terminal output shows the program's execution, including system information and the results of the swap function. The code is as follows:

```
#include <iostream>
#include <vector>
#include "swap.h"

using namespace std;

int main()
{
    // Array 1
    int a[5] = {1, 2, 3, 4, 5};
    // Array 2
    int b[5] = {6, 7, 8, 9, 10};

    // Swap elements at index 1 and 2
    swap(a, b, 1, 2);

    // Print the arrays after swap
    for (int i = 0; i < a.size(); i++)
        cout << a[i] << " ";
    cout << endl;

    for (int i = 0; i < b.size(); i++)
        cout << b[i] << " ";
    cout << endl;

    return 0;
}
```

The terminal output shows the program's execution, including system information and the results of the swap function. The output is as follows:

```
lightfshine@lightfshine:~$ ./No.3.cpp
1 2 3 4 5
6 7 8 9 10
Setelah Tukar Elemen pada Posisi (1, 2):
Array 1:
6 7 8 9 10
Array 2:
1 2 3 4 5
Setelah Tukar Pointer: a = 10, b = 20
Setelah Tukar Pointer: a = 20, b = 10
```

IV . Kesimpulan

Abstract Data Type (ADT) merupakan konsep penting dalam pemrograman yang memisahkan antara definisi logis dan implementasi teknis suatu tipe data. Dengan menggunakan ADT, kita dapat mendefinisikan tipe data secara abstrak beserta operasi-operasi dasar yang bisa dilakukan, tanpa perlu memikirkan bagaimana tipe data tersebut diimplementasikan. Pendekatan ini memungkinkan kita untuk membuat program yang lebih terstruktur, modular, dan mudah dikelola.

Dalam praktikum ini, ADT diimplementasikan melalui dua bagian utama:

1. Spesifikasi Type dan Operasi (header .h) yang mendefinisikan struktur tipe data dan operasi dasar.
2. Body atau Implementasi (source .cpp) yang menguraikan detail dari fungsi-fungsi atau prosedur yang digunakan untuk memanipulasi tipe data tersebut.

Dengan menerapkan ADT, kita dapat:

1. Mempermudah pengelolaan kode.
2. Memisahkan antara apa yang dilakukan (spesifikasi) dan bagaimana itu dilakukan (implementasi).
3. Meningkatkan fleksibilitas dan keterbacaan kode.

Secara keseluruhan, penggunaan ADT dalam pengembangan program membantu menciptakan kode yang lebih rapi, terstruktur, dan dapat diperluas di masa mendatang, sehingga sangat bermanfaat dalam skala program yang lebih besar.