

**LAPORAN PRAKTIKUM**  
**PERTEMUAN 3**



**Disusun Oleh:**  
**Naura Aisha Zahira (2311104078)**  
**S1SE-07-02**

**Dosen :**  
**Wahyu Andi Saputra, S.Pd., M.Eng**

**PROGRAM STUDI S1 SOFTWARE ENGINEERING**  
**FAKULTAS INFORMATIKA**  
**TELKOM UNIVERSITY**  
**PURWOKERTO**  
**2024**

## 1. Tujuan

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

## 2. Landasan Teori

ADT adalah TYPE dan sekumpulan PRIMITIF (operasi dasar) terhadap TYPE tersebut. Selain itu, dalam sebuah ADT yang lengkap, disertakan pula definisi invarian dari TYPE dan aksioma yang berlaku. ADT merupakan definisi STATIK.

Definisi type dari sebuah ADT dapat mengandung sebuah definisi ADT lain. Misalnya :

a. ADT waktu yang terdiri dari ADT JAM dan ADT DATE

b. Garis terdiri dari dua buah ADT POINT

SEGI4 yang terdiri dari pasangan dua buah POINT (Top,Left) dan (Bottom,Right)

TYPE diterjemahkan menjadi type terdefinisi dalam bahasa yang bersangkutan. Jika dalam bahasa C menggunakan struct PRIMITIF, dalam konteks prosedural, diterjemahkan menjadi fungsi atau prosedur. PRIMITIF dikelompokkan menjadi:

1. Konstruktor/Kreator, pemebentuk nilai type.Semua objek (variabel) bertype tersebut harus melalui konstruktor. Biasanya namanya diawali Make.
2. Selector, untuk mengakses tipe komponen(biasanya namanya diawali Get).
3. Prosedur pengubah nilai komponen (biasanya namanya diawali Get).
4. Tipe validator komponen, yang dipakai untuk mentest apakah dapat membentuk tipe sesuai dengan batasan.
5. Destruktor/Dealokator yaitu untuk “menghancurkan” nilai objek/variabel (sekalius memori penyimpanannya).
6. Baca/Tulis, untuk interface dengan input/output device.
7. Operator relasional, terhadap tipe tersebut untuk mendefinisikan lebih besar, lebih kecil, sama dengan dan sebagainya.
8. Aritmatika terhadap tipe tersebut, karena biasanya aritmatika dalam bahasa C hanya terdefinisi untuk bilangan numerik.
9. Konversi dari tipe tersebut ke tipe dasar dan sebaliknya.

ADT biasanya diimplementasikan menjadi dua buah modul utama dan 1 modul interface program utama (driver). Dua modul tersebut adalah sebagai berikut:

1. Definisi/Spesifikasi Type dan Primitif/Header fungsi (.h)

- Spesifikasi type sesuai dengan kaidah bahasa yang dipakai
- Spesifikasi dari primitif sesuai dengan kaidah dalam konteks prosedural,yaitu:
- Fungsi : nama, domain, range, dan prekondisi jika ada

- Prosedur : Initial state, Final state, dan proses yang dilakukan

## 2. Body/realisasi dari primitif (.c)

Berupa kode program dalam bahasa yang bersangkutan (dalam praktikum ini berarti dengan bahasa C++). Realisasi fungsi dan prosedur harus sedapat mungkin memanfaatkan selector dan konstruktor. Untuk memahami lebih jelas mengenai konsep ADT

## 3. Guided

Kode ini adalah program C++ yang digunakan untuk menghitung rata-rata dua nilai yang diinput oleh pengguna. Berikut adalah penjelasan kode secara detail:

### a. Deklarasi Header dan Namespace:

```
#include <iostream>

using namespace std;
```

Pada bagian ini, kita menginclude header *iostream* yang digunakan untuk input dan output data. Kemudian, kita menggunakan namespace *std* untuk mengakses fungsi-fungsi yang tersedia di dalamnya.

### b. Deklarasi Struktur Data:

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

Pada bagian ini, kita mendeklarasikan struktur data *mahasiswa* yang memiliki tiga atribut: *nim* (char array dengan panjang 10), *nilai1* (integer), dan *nilai2* (integer).

### c. Deklarasi Fungsi:

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

Pada bagian ini, kita mendeklarasikan dua fungsi: *inputMhs* dan *rata2*. Fungsi *inputMhs* digunakan untuk menginput data mahasiswa, sedangkan fungsi *rata2* digunakan untuk menghitung rata-rata dua nilai.

### d. Fungsi Main:

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

Pada bagian ini, kita mendefinisikan fungsi *main* yang merupakan entry point dari program. Fungsi ini melakukan tiga hal:

- Membuat objek *mhs* dari struktur data *mahasiswa*.
- Memanggil fungsi *inputMhs* untuk menginput data mahasiswa.
- Memanggil fungsi *rata2* untuk menghitung rata-rata dua nilai dan menampilkan hasilnya ke layar.

e. Fungsi InputMhs:

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

Pada bagian ini, kita mendefinisikan fungsi *inputMhs* yang digunakan untuk menginput data mahasiswa. Fungsi ini meminta pengguna untuk menginput NIM, nilai 1, dan nilai 2, kemudian menyimpannya ke dalam objek *m*.

f. Fungsi Rata2:

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

Pada bagian ini, kita mendefinisikan fungsi *rata2* yang digunakan untuk menghitung rata-rata dua nilai. Fungsi ini mengambil objek *m* sebagai parameter dan menghitung rata-rata dengan menjumlahkan nilai 1 dan nilai 2, kemudian membaginya dengan 2.0f (untuk menghasilkan nilai float).

#### 4. Unguided

- 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  $0.3 \cdot \text{uts} + 0.4 \cdot \text{uas} + 0.3 \cdot \text{tugas}$ .

Code:

```

1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  struct Mahasiswa {
7      string nama;
8      string nim;
9      float uts, uas, tugas, nilaiAkhir;
10 };
11
12 // Fungsi untuk menghitung nilai akhir
13 float hitungNilaiAkhir(float uts, float uas, float tugas) {
14     return (0.3 * uts) + (0.4 * uas) + (0.3 * tugas);
15 }
16
17 // Fungsi untuk memasukkan data mahasiswa
18 void inputMahasiswa(Mahasiswa &m) {
19     cout << "Input Nama: ";
20     cin.ignore(); // Membersihkan buffer agar getline bisa berfungsi
21     getline(cin, m.nama);
22
23     cout << "Input NIM: ";
24     cin >> m.nim;
25
26     cout << "Input nilai UTS: ";
27     cin >> m.uts;
28
29     cout << "Input nilai UAS: ";
30     cin >> m.uas;
31
32     cout << "Input nilai Tugas: ";
33     cin >> m.tugas;
34
35     m.nilaiAkhir = hitungNilaiAkhir(m.uts, m.uas, m.tugas);
36 }
37
38 // Fungsi untuk menampilkan data mahasiswa
39 void tampilMahasiswa(Mahasiswa m) {
40     cout << "\nNama: " << m.nama;
41     cout << "\nNIM: " << m.nim;
42     cout << "\nNilai UTS: " << m.uts;
43     cout << "\nNilai UAS: " << m.uas;
44     cout << "\nNilai Tugas: " << m.tugas;
45     cout << "\nNilai Akhir: " << m.nilaiAkhir << endl;
46 }
47
48 int main() {
49     const int MAX_MHS = 10; // Maksimal 10 mahasiswa
50     Mahasiswa mahasiswa[MAX_MHS];
51     int jumlahMahasiswa;
52
53     cout << "Masukkan jumlah mahasiswa (maks 10): ";
54     cin >> jumlahMahasiswa;
55
56     if(jumlahMahasiswa > MAX_MHS) {
57         cout << "Jumlah mahasiswa melebihi batas maksimal!" << endl;
58         return 1;
59     }
60
61     // Input data mahasiswa
62     for (int i = 0; i < jumlahMahasiswa; i++) {
63         cout << "\nData Mahasiswa ke-" << i + 1 << ": " << endl;
64         inputMahasiswa(mahasiswa[i]);
65     }
66
67     // Tampilkan data mahasiswa
68     cout << "\nDaftar Mahasiswa:\n";
69     for (int i = 0; i < jumlahMahasiswa; i++) {
70         cout << "\nMahasiswa ke-" << i + 1 << ": " << endl;
71         tampilMahasiswa(mahasiswa[i]);
72     }
73
74     return 0;
75 }
76

```

Output:

```
Data Mahasiswa ke-1:
Input Nama: naura
Input NIM: 2311104078
Input nilai UTS: 50
Input nilai UAS: 100
Input nilai Tugas: 90
```

```
Data Mahasiswa ke-2:
Input Nama: aisha
Input NIM: 2311104078
Input nilai UTS: 70
Input nilai UAS: 80
Input nilai Tugas: 65
```

```
Data Mahasiswa ke-3:
Input Nama: zahira
Input NIM: 2311104078
Input nilai UTS: 85
Input nilai UAS: 95
Input nilai Tugas: 75
```

```
Daftar Mahasiswa:
```

```
Mahasiswa ke-1:
```

```
Nama: naura
NIM: 2311104078
Nilai UTS: 50
Nilai UAS: 100
Nilai Tugas: 90
Nilai Akhir: 82
```

```
Mahasiswa ke-2:
```

```
Nama: aisha
NIM: 2311104078
Nilai UTS: 70
Nilai UAS: 80
Nilai Tugas: 65
Nilai Akhir: 72.5
```

```
Mahasiswa ke-3:
```

```
Nama: zahira
NIM: 2311104078
Nilai UTS: 85
Nilai UAS: 95
Nilai Tugas: 75
Nilai Akhir: 86
```

- b. 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);

    return 0;
}

```

Gambar 3-1 Main.cpp pelajaran

Contoh output hasil:

```

nama pelajaran : Struktur Data
nilai : STD

```

Gambar 3-2 output pelajaran

Code:

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  // Definisi ADT Pelajaran di file "pelajaran.h"
6  struct pelajaran {
7      string namamapel;
8      string kodepel;
9  };
10
11 // Fungsi untuk membuat objek Pelajaran baru
12 pelajaran create_pelajaran(string namapel, string kodepel) {
13     pelajaran pel;
14     pel.namamapel = namapel;
15     pel.kodepel = kodepel;
16     return pel;
17 }
18
19 // Prosedur untuk menampilkan detail Pelajaran
20 void tampil_pelajaran(pelajaran pel) {
21     cout << "nama pelajaran: " << pel.namamapel << endl;
22     cout << "nilai: " << pel.kodepel << endl;
23 }
24
25 int main() {
26     // Membuat objek Pelajaran
27     pelajaran pel = create_pelajaran("Struktur Data", "STD");
28
29     // Menampilkan detail Pelajaran
30     tampil_pelajaran(pel);
31
32     return 0;
33 }

```

Output:

nama pelajaran: Struktur Data  
nilai: STD

c. Buatlah program dengan ketentuan:

- 2 buah array 2D unteger 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
- Fungsi/prosedur yang akan menukarkan isi dari variabel yang ditunjuk oleh 2 buah pointer

Code:



```

1  #include <iostream>
2
3  using namespace std;
4
5  // Fungsi untuk menampilkan isi array 2D
6  void tampilArray(int arr[3][3]) {
7      for (int i = 0; i < 3; i++) {
8          for (int j = 0; j < 3; j++) {
9              cout << arr[i][j] << " ";
10             }
11             cout << endl;
12         }
13     }
14
15     // Fungsi untuk menukarkan dua nilai array pada posisi tertentu
16     void tukarArray(int arr1[3][3], int arr2[3][3], int baris, int kolom) {
17         int temp = arr1[baris][kolom];
18         arr1[baris][kolom] = arr2[baris][kolom];
19         arr2[baris][kolom] = temp;
20     }
21
22     // Fungsi untuk menukarkan isi variabel yang ditunjuk oleh dua pointer
23     void tukarPointer(int *ptr1, int *ptr2) {
24         int temp = *ptr1;
25         *ptr1 = *ptr2;
26         *ptr2 = temp;
27     }
28
29     int main() {
30         // Deklarasi 2 buah array 2D berukuran 3x3
31         int arr1[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
32         int arr2[3][3] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
33
34         // Deklarasi 2 pointer integer
35         int a = 10, b = 20;
36         int *ptr1 = &a;
37         int *ptr2 = &b;
38
39         // Menampilkan isi array sebelum ditukar
40         cout << "Array 1 sebelum ditukar:" << endl;
41         tampilArray(arr1);
42         cout << "Array 2 sebelum ditukar:" << endl;
43         tampilArray(arr2);
44
45         // Menukarkan isi array di posisi tertentu
46         tukarArray(arr1, arr2, 1, 1); // Tukar elemen di baris 1, kolom 1
47
48         // Menampilkan isi array setelah ditukar
49         cout << "\nArray 1 setelah ditukar:" << endl;
50         tampilArray(arr1);
51         cout << "Array 2 setelah ditukar:" << endl;
52         tampilArray(arr2);
53
54         // Menampilkan nilai pointer sebelum ditukar
55         cout << "\nNilai sebelum pointer ditukar:" << endl;
56         cout << "Nilai ptr1 (a): " << *ptr1 << endl;
57         cout << "Nilai ptr2 (b): " << *ptr2 << endl;
58
59         // Menukarkan isi dari variabel yang ditunjuk oleh pointer
60         tukarPointer(ptr1, ptr2);
61
62         // Menampilkan nilai pointer setelah ditukar
63         cout << "\nNilai setelah pointer ditukar:" << endl;
64         cout << "Nilai ptr1 (a): " << *ptr1 << endl;
65         cout << "Nilai ptr2 (b): " << *ptr2 << endl;
66
67         return 0;
68     }
69

```

Output:

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

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

Nilai sebelum pointer ditukar:
Nilai ptr1 (a): 10
Nilai ptr2 (b): 20

Nilai setelah pointer ditukar:
Nilai ptr1 (a): 20
Nilai ptr2 (b): 10
```

## 5. Kesimpulan

Dalam praktikum ini, dipelajari dan dipraktikkan konsep Abstract Data Type (ADT) dan penerapannya dalam C++. ADT memberikan kerangka yang memungkinkan pemrograman modular dan terstruktur dengan baik melalui pemisahan antara spesifikasi dan implementasi. Melalui implementasi struktur data seperti mahasiswa dan fungsi yang terkait, diperoleh pemahaman tentang penggunaan ADT untuk memanipulasi data. Program C++ yang dibuat berhasil mengimplementasikan ADT untuk menghitung rata-rata nilai, mengelola data mahasiswa, serta mengaplikasikan konsep ADT lainnya dalam tugas yang lebih kompleks.

Ini menunjukkan pentingnya ADT dalam membangun program yang lebih mudah dikelola, dimodifikasi, dan dipahami.

