# LAPORAN PRAKTIKUM
## Modul 03
## “ABSTRACT DATA TYPE (ADT)”


**Disusun Oleh:**

**Dimastian Aji Wibowo (2311104058)**

**SE-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.
    - Mengetahui cara penerapan ADT pada bahasa pemrograman C++
2. **Landasan Teori**

**Absract Data Type (ADT)**

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:

1. ADT waktu yang terdiri dari ADT JAM dan ADT DATE
2. Garis terdiri dari duah 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 dikelompokan 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 (sekaligus 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.

**3. Guided**

1. Menuliskan header yang diperlukan dan menuliskan namespace std.
2. Mendefinisikan struktur bernama mahasiswa yang berisi char nim[10] yang merupakan sebuah array karakter untuk menyimpan ID mahasiswa (NIM) hingga 9 karakter ditambah karakter null sebagai penanda akhir dan int nilai1, nilai2 yang merupakan dua variabel integer untuk menyimpan nilai mahasiswa.
3. Mendeklarasikan fungsi inputMhs(mahasiswa &m) untuk memasukkan data mahasiswa dengan menggunakan referensi dan fungsi rata2(mahasiswa m) untuk menghitung rata-rata dari dua nilai.
4. Membuat fungsi utama diikuti dengan membuat sebuah variabel mahasiswa bernama mhs.
5. Fungsi inputMhs dipanggil untuk memasukkan data untuk mhs dan fungsi rata2 dipanggil untuk menghitung dan menampilkan rata-rata dari dua nilai dengan program diakhiri dengan return 0; yang menunjukkan bahwa eksekusi berhasil.
6. Mendefinisikan fungsi input untuk mengambil referensi dari struktur mahasiswa dan meminta pengguna untuk memasukkan: ID mahasiswa (nim), Nilai pertama (nilai1), dan Nilai kedua (nilai2).
7. Membuat fungsi perhitungan rata – rata dengan mengambil struktur mahasiswa sebagai argumen dan menghitung rata-rata dari nilai1 dan nilai2 dan dihitung menggunakan float.

```cpp
#include <iostream>
using namespace std;

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

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


int main(){
    mahasiswa mhs;
    inputMhs(mhs);
    cout<<"Rata - rata = "<<rata2(mhs);
    return 0;
}
void inputMhs(mahasiswa &m){
    cout<<"Input nim = ";
    cin>>(m).nim;
    cout<<"Input nilai = ";
    cin>>(m).nilai1;
    cout<<"Input nilai = ";
    cin>>(m).nilai2;
}
float rata2(mahasiswa m){
    return(m.nilai1+m.nilai2)/2;
}
```

8. Berikut merupakan output dari program tersebut.

```
Input nim = 2311104058
Input nilai = 98
Input nilai = 80
Rata - rata = 89
```

## 4. Unguided

### A. Program Data Mahasiswa

1. Membuat struktur Mahasiswa yang memiliki atribut: nama (string), nim (string), uts (float), uas (float), tugas (float), dan nilaiAkhir (float).
2. Membuat fungsi yang menerima tiga parameter: nilai UTS, nilai UAS, dan nilai tugas dengan formula nilai UTS, nilai UAS, dan nilai tugas masing – masing 0.3, 0.4, dan 0,3 dengan nilai akhir hasil perhitungan dikembalikan (return) ke program utama.
3. Mendeklarasikan array, menentukan jumlah maksimal mahasiswa yang bisa disimpan (10 mahasiswa), dan membuat variabel yang akan digunakan untuk menyimpan jumlah mahasiswa yang akan diinput.

```
#include <iostream>
#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) {
    return (0.3 * uts) + (0.4 * uas) + (0.3 * tugas);
}
int main() {
    const int MAX_MAHASISWA = 10;
    Mahasiswa mahasiswa[MAX_MAHASISWA];
    int jumlahMahasiswa;
    cout << "Masukkan jumlah mahasiswa (maksimal 10): ";
    cin >> jumlahMahasiswa;
    if (jumlahMahasiswa > MAX_MAHASISWA) {
        cout << "Jumlah mahasiswa melebihi batas maksimal!" << endl;
        return 1;
    }
```

4. Membuat perulangan for yang digunakan untuk menginput data mahasiswa satu per satu hingga jumlah mahasiswa yang diinput tercapai lalu setiap mahasiswa akan diinput dengan nama, NIM, nilai UTS, nilai UAS, dan nilai tugas.
5. Setelah semua data dimasukkan, program menghitung nilai akhir menggunakan fungsi hitungNilaiAkhir dan menyimpannya di mahasiswa[i].nilaiAkhir.
6. Setelah semua data mahasiswa dimasukkan, program menampilkan data masing-masing mahasiswa (nama, NIM, nilai UTS, UAS, tugas, dan nilai akhir) menggunakan loop for.
7. Menuliskan return 0 untuk yang menunjukkan bahwa program berjalan dengan

sukses.

```
    for (int i = 0; i < jumlahMahasiswa; i++) {
        cout << "\nMasukkan data mahasiswa ke-" << i + 1 << ":" << endl;
        cout << "Nama: ";
        cin >> ws;
        getline(cin, mahasiswa[i].nama);
        cout << "NIM: ";
        cin >> mahasiswa[i].nim;
        cout << "Nilai UTS: ";
        cin >> mahasiswa[i].uts;
        cout << "Nilai UAS: ";
        cin >> mahasiswa[i].uas;
        cout << "Nilai Tugas: ";
        cin >> mahasiswa[i].tugas;

        mahasiswa[i].nilaiAkhir = hitungNilaiAkhir(mahasiswa[i].uts, mahasiswa[i].uas, mahasiswa[i].tugas);
    }
    cout << "\nData Mahasiswa:\n";
    for (int i = 0; i < jumlahMahasiswa; i++) {
        cout << "\nMahasiswa ke-" << i + 1 << ":" << endl;
        cout << "Nama: " << mahasiswa[i].nama << endl;
        cout << "NIM: " << mahasiswa[i].nim << endl;
        cout << "Nilai UTS: " << mahasiswa[i].uts << endl;
        cout << "Nilai UAS: " << mahasiswa[i].uas << endl;
        cout << "Nilai Tugas: " << mahasiswa[i].tugas << endl;
        cout << "Nilai Akhir: " << mahasiswa[i].nilaiAkhir << endl;
    }
    return 0;
}
```

8. Berikut merupakan output dari program tersebut.

```
Masukkan jumlah mahasiswa (maksimal 10): 2

Masukkan data mahasiswa ke-1:
Nama: Dimastian Aji Wibowo
NIM: 2311104058
Nilai UTS: 100
Nilai UAS: 100
Nilai Tugas: 90

Masukkan data mahasiswa ke-2:
Nama: Zhafir Zaidan Avail
NIM: 2311104059
Nilai UTS: 60
Nilai UAS: 65
Nilai Tugas: 70

Data Mahasiswa:

Mahasiswa ke-1:
Nama: Dimastian Aji Wibowo
NIM: 2311104058
Nilai UTS: 100
Nilai UAS: 100
Nilai Tugas: 90
Nilai Akhir: 97

Mahasiswa ke-2:
Nama: Zhafir Zaidan Avail
NIM: 2311104059
Nilai UTS: 60
Nilai UAS: 65
Nilai Tugas: 70
Nilai Akhir: 65
```

## B. Program mata pelajaran

1. Membuat file header dan mendefinisikannya.
2. Menuliskan library yang diperlukan.
3. Ini mendefinisikan tipe data Pelajaran menggunakan struct, yang

mengelompokkan beberapa data terkait dalam satu unit yaitu namaMapel bertipe string yang menyimpan nama mata pelajaran dan kode mapel yang menyimpan kode mata pelajaran.

4. Mendeklarasikan fungsi yang menerima dua parameter bertipe string yaitu namapel dan kodepel, yang berfungsi untuk membuat sebuah objek Pelajaran lalu fungsi ini akan mengembalikan objek dari tipe Pelajaran, yang menciptakan dan menginisialisasi nilai namaMapel dan kodeMapel pada struct tersebut.
5. Mendeklarasikan prosedur tampil_pelajaran yang mana prosedur ini digunakan untuk menampilkan informasi dari objek Pelajaran dengan parameter pel yang merupakan objek bertipe Pelajaran yang diteruskan ke prosedur ini untuk ditampilkan datanya.
6. Menuliskan penutupan header.

```
#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
```

7. Membuat file baru.
8. Menuliskan library include file header sebelumnya dan library lain yang diperlukan.
9. Membuat fungsi create_pelajaran yang bertugas untuk membuat dan menginisialisasi objek Pelajaran dan fungsi ini menerima dua parameter bertipe string yang digunakan untuk mengisi atribut namaMapel dan kodeMapel dari objek Pelajaran dengan parameter namapel yang mengisi nama mata pelajaran dan kodepel yang engisi kode mata pelajaran.
10. Menuliskan return pel untuk mengembalikan objek pel yang sudah diisi dengan data namaMapel dan kodeMapel ke pemanggil fungsi.
11. Membuat fungsi tampil_pelajaran untuk menampilkan data yang ada dalam objek Pelajaran dengan parameter pel yang menerima objek Pelajaran yang berisi

data mata pelajaran lalu pel.namaMapel: Mengakses dan menampilkan nama mata pelajaran dari objek pel dan pel.kodeMapel: Mengakses dan menampilkan kode mata pelajaran dari objek pel.

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

using namespace std;

// Implementasi fungsi create_pelajaran
Pelajaran create_pelajaran(string namapel, string kodepel) {
    Pelajaran pel;
    pel.namaMapel = namapel;
    pel.kodeMapel = kodepel;
    return pel;
}

// Implementasi prosedur tampil_pelajaran
void tampil_pelajaran(Pelajaran pel) {
    cout << "Nama Mata Pelajaran: " << pel.namaMapel << endl;
    cout << "Kode Mata Pelajaran: " << pel.kodeMapel << endl;
}
```

12. Membuat file main.
13. Menuliskan library include file header dan library lain yang diperlukan.
14. Membuat fungsi main().
15. Membuat objek Pelajaran dan create_pelajaran dimana Pelajaran pel ini akan mendeklarasikan sebuah variabel pel bertipe Pelajaran dan Pelajaran adalah tipe data yang didefinisikan dalam file header pelajaran.h.
16. Fungsi create_pelajaran dipanggil dengan dua parameter: "Matematika" sebagai namapel dan "MTH101" sebagai kodepel. Fungsi ini mengembalikan objek Pelajaran yang berisi data tersebut. Objek ini kemudian disimpan dalam variabel pel. Di dalam fungsi create_pelajaran, atribut namaMapel akan diisi dengan "Matematika" dan kodeMapel dengan "MTH101", yang nantinya menjadi data di dalam objek pel.
17. Menampilkan data dengan membuat tampil_pelajaran(pel) dimana prosedur ini dipanggil dengan objek pel yang sudah berisi data mata pelajaran.
18. Menuliskan return 0 untuk menunjukkan bahwa program selesai dieksekusi dengan sukses.

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

int main() {
    Pelajaran pel = create_pelajaran("Matematika", "MTH101");

    tampil_pelajaran(pel);

    return 0;
}
```

19. Berikut merupakan output dari program tersebut.

```
Nama Mata Pelajaran: Matematika
Kode Mata Pelajaran: MTH101
```

## C. Array 2D

1. Membuat fungsi tampilkanArray yang bertugas untuk menampilkan elemen-elemen dari array 2D bertipe int[3][3] ke layar dengan memanfaatkan perulangan for dengan i adalah indeks baris dari 0 hingga 2 dan j adalah indeks kolom dari 0 hingga 2 untuk setiap elemen array (arr[i][j]), nilai elemen tersebut ditampilkan dengan cout, dan dipisahkan dengan spasi.
2. Membuat fungsi tukarArray2D yang menukarkan elemen dari dua array 2D pada posisi yang sama (pada baris dan kolom yang diberikan).
3. Membuat fungsi tukarPointer yang bertujuan menukarkan nilai yang ditunjuk oleh dua pointer integer.

```cpp
#include <iostream>
using namespace std;

void tampilkanArray(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 tukarArray2D(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* ptr1, int* ptr2) {
    int temp = *ptr1;
    *ptr1 = *ptr2;
    *ptr2 = temp;
}
```

4. Membuat fungsi main() .
5. Deklarasi array 2D dan pointer dengan dua array 2D, array1 dan array2,

dideklarasikan dengan ukuran 3x3, masing-masing berisi angka – angka tertentu dan dua pointer ptr1 dan ptr2 masing-masing menunjuk ke elemen pertama dari array1 dan array2.

6. Menampilkan array sebelum penukaran.
7. Menukarkan elemen pada Posisi (1,1) dari array1 dan array2 dengan memanggil tukarArray2D untuk menukarkan elemen pada posisi baris ke-1 dan kolom ke-1 dari array1 dan array2.
8. Menampilkan array setelah penukaran elemen.
9. Menukarkan nilai yang ditunjuk pointer dengan memanggil tukarPointer untuk menukarkan nilai yang ditunjuk oleh ptr1 dan ptr2.
10. Menampilkan nilai yang ditunjuk oleh ptr1 dan ptr2 setelah penukaran.
11. Menuliskan return 0.

```
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 = &array1[0][0];
    int* ptr2 = &array2[0][0];

    cout << "Array 1 Sebelum Penukaran:" << endl;
    tampilkanArray(array1);
    cout << "Array 2 Sebelum Penukaran:" << endl;
    tampilkanArray(array2);

    tukarArray2D(array1, array2, 1, 1);

    cout << "\nArray 1 Setelah Penukaran (1,1) dengan Array 2:" << endl;
    tampilkanArray(array1);
    cout << "Array 2 Setelah Penukaran (1,1) dengan Array 1:" << endl;
    tampilkanArray(array2);

    tukarPointer(ptr1, ptr2);

    cout << "\nNilai yang ditunjuk oleh ptr1 setelah penukaran: " << *ptr1 << endl;
    cout << "Nilai yang ditunjuk oleh ptr2 setelah penukaran: " << *ptr2 << endl;

    return 0;
}
```

12. Berikut merupakan output dari program tersebut.

```
Array 1 Sebelum Penukaran:
1 2 3
4 5 6
7 8 9
Array 2 Sebelum Penukaran:
9 8 7
6 5 4
3 2 1

Array 1 Setelah Penukaran (1,1) dengan Array 2:
1 2 3
4 5 6
7 8 9
Array 2 Setelah Penukaran (1,1) dengan Array 1:
9 8 7
6 5 4
3 2 1

Nilai yang ditunjuk oleh ptr1 setelah penukaran: 9
Nilai yang ditunjuk oleh ptr2 setelah penukaran: 1
```

## 5. Kesimpulan

Praktikum kali ini yang membahas tentang ADT (Abstract Data Type) adalah bahwa ADT memungkinkan kita untuk mendefinisikan tipe data abstrak secara terstruktur dan terpisah antara deklarasi dan implementasi. Dengan menggunakan ADT, kita dapat memisahkan logika pembuatan dan pengelolaan data dari detail implementasinya, sehingga mempermudah pengembangan, pemeliharaan, dan pemahaman program. Implementasi ADT seperti yang dilakukan pada tipe data Pelajaran menunjukkan bagaimana fungsi dan prosedur digunakan untuk membuat dan menampilkan data, memberikan contoh bagaimana konsep abstraksi diterapkan dalam pemrograman berorientasi objek.