

LAPORAN PRAKTIKUM
Modul 3
“ABSTRACT DATA TYPE (ADT)”



Disusun Oleh:
Benedictus Qsota Noventino Baru - 2311104029
S1SE07A

Dosen :
Yudha Islami Sulistya, S.Kom., M.Cs

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM
PURWOKERTO
2024

LATIHAN (UNGUIDED)

Nomor 1

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

// Struktur untuk menyimpan data mahasiswa
struct Mahasiswa {
    string nama;
    string nim;
    float uts;
    float uas;
    float tugas;
    float nilai_akhir;
};

// Fungsi untuk menghitung nilai akhir
float hitung_nilai_akhir(float uts, float uas, float tugas) {
    return (0.3 * uts) + (0.4 * uas) + (0.3 * tugas);
}

// Fungsi untuk menambah data mahasiswa
void tambah_mahasiswa(Mahasiswa& mhs) {
    cout << "Nama      : ";
    cin.ignore();
    getline(cin, mhs.nama);
    cout << "NIM       : ";
    getline(cin, mhs.nim);
    cout << "Nilai UTS  : ";
    cin >> mhs.uts;
    cout << "Nilai UAS  : ";
    cin >> mhs.uas;
    cout << "Nilai Tugas: ";
    cin >> mhs.tugas;

    // Hitung nilai akhir menggunakan fungsi
    mhs.nilai_akhir = hitung_nilai_akhir(mhs.uts, mhs.uas, mhs.tugas);
}

// Fungsi untuk menampilkan data mahasiswa
void tampilkan_mahasiswa(const Mahasiswa mhs[], int jumlah_mahasiswa) {
    for (int i = 0; i < jumlah_mahasiswa; i++) {
        cout << "\nData Mahasiswa " << i + 1 << endl;
        cout << "Nama      : " << mhs[i].nama << endl;
        cout << "NIM       : " << mhs[i].nim << endl;
        cout << "Nilai UTS  : " << mhs[i].uts << endl;
        cout << "Nilai UAS  : " << mhs[i].uas << endl;
        cout << "Nilai Tugas: " << mhs[i].tugas << endl;
        cout << "Nilai Akhir: " << mhs[i].nilai_akhir << endl;
        cout << "-----" << endl;
    }
}

int main() {
    const int MAX_MAHASISWA = 10;
    Mahasiswa mahasiswa[MAX_MAHASISWA];
    int jumlah_mahasiswa = 0;
    char lanjut;

    // Input data mahasiswa
    do {
        if (jumlah_mahasiswa < MAX_MAHASISWA) {
            cout << "\nInput data mahasiswa ke-" << jumlah_mahasiswa + 1 << endl;
            tambah_mahasiswa(mahasiswa[jumlah_mahasiswa]);
            jumlah_mahasiswa++;

            cout << "Ingin menambah data mahasiswa lagi? (y/n): ";
            cin >> lanjut;
        } else {
            cout << "Data mahasiswa sudah mencapai maksimal." << endl;
            break;
        }
    } while (lanjut == 'y' || lanjut == 'Y');

    // Tampilkan data mahasiswa
    tampilkan_mahasiswa(mahasiswa, jumlah_mahasiswa);

    return 0;
}
```

```
Input data mahasiswa ke-1
Nama      : Noven
NIM       : 2311104029
Nilai UTS : 90
Nilai UAS : 90
Nilai Tugas: 90
Ingin menambah data mahasiswa lagi? (y/n): n

Data Mahasiswa 1
Nama      : oven
NIM       : 2311104029
Nilai UTS : 90
Nilai UAS : 90
Nilai Tugas: 90
Nilai Akhir: 90
-----
```

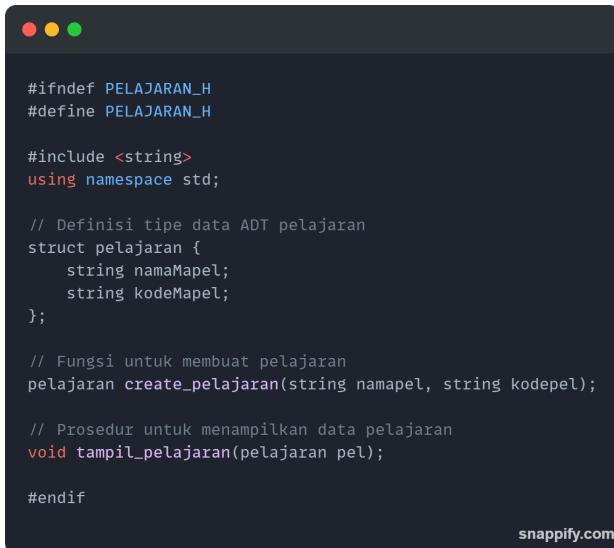
Penjelasan:

1. **Struct Mahasiswa**: Digunakan untuk menyimpan data setiap mahasiswa, termasuk nama, nim, uts, uas, tugas, dan nilai akhir.
2. **Fungsi hitung_nilai_akhir**: Menghitung nilai akhir berdasarkan rumus yang diberikan ($0.3 * UTS + 0.4 * UAS + 0.3 * Tugas$).
3. **Fungsi tambah_mahasiswa**: Mengambil input dari pengguna untuk setiap mahasiswa dan menyimpannya ke dalam array.
4. **Fungsi tampilkan_mahasiswa**: Menampilkan data mahasiswa yang telah diinput.
5. **Array mahasiswa**: Menyimpan data mahasiswa dengan maksimal 10 elemen.

Soal 2

File pelajaran.h

Fungsi: File ini berfungsi sebagai **header file** yang mendeklarasikan tipe data **pelajaran**, fungsi **create_pelajaran**, dan prosedur **tampil_pelajaran**.



```
#ifndef PELAJARAN_H
#define PELAJARAN_H

#include <string>
using namespace std;

// Definisi tipe data ADT pelajaran
struct pelajaran {
    string namaMapel;
    string kodeMapel;
};

// Fungsi untuk membuat pelajaran
pelajaran create_pelajaran(string namaapel, string kodepel);

// Prosedur untuk menampilkan data pelajaran
void tampil_pelajaran(pelajaran pel);

#endif
```

Penjelasan:

- **Preprocessor Directives (#ifndef, #define, #endif):** Mencegah multiple inclusion, yaitu ketika file **pelajaran.h** di-include lebih dari sekali, untuk menghindari error.
- **struct pelajaran:** Mendefinisikan tipe data custom yang disebut **pelajaran**. Tipe data ini memiliki dua field, yaitu:
 - **namaMapel:** String yang menyimpan nama mata pelajaran.
 - **kodeMapel:** String yang menyimpan kode mata pelajaran.
- **Deklarasi Fungsi dan Prosedur:**
 - **create_pelajaran:** Fungsi ini akan digunakan untuk membuat dan mengembalikan objek **pelajaran**.
 - **tampil_pelajaran:** Prosedur ini akan menampilkan isi dari objek **pelajaran**.

File pelajaran.cpp

Fungsi: File ini adalah **implementasi** dari fungsi-fungsi yang dideklarasikan di file header **pelajaran.h**.

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

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

snappify.com

Penjelasan:

- **#include "pelajaran.h"**: Meng-include header **pelajaran.h** agar implementasi fungsi yang dideklarasikan di sana bisa ditulis di sini.
- **create_pelajaran**: Fungsi ini bertugas untuk membuat objek **pelajaran** dan mengisi nilai **namaMapel** dan **kodeMapel**. Ini adalah fungsi untuk membentuk sebuah instance dari **struct pelajaran**:
 - Argumen input: **namaPel** dan **kodePel** (masing-masing bertipe string).
 - Fungsi ini akan mengembalikan sebuah objek **pelajaran** dengan nama dan kode pelajaran yang sudah diisi.
- **tampil_pelajaran**: Prosedur ini menampilkan informasi mata pelajaran dengan cara mencetak **namaMapel** dan **kodeMapel** dari objek **pelajaran**.

File main.cpp

Fungsi: Ini adalah file utama yang menjalankan program. File ini menggunakan fungsi yang diimplementasikan di file `pelajaran.cpp` untuk membuat objek pelajaran dan menampilkan informasinya.

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

using namespace std;

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

    // Membuat objek pelajaran menggunakan fungsi create_pelajaran
    pelajaran pel = create_pelajaran(namapel, kodepel);

    // Menampilkan data pelajaran
    tampil_pelajaran(pel);

    return 0;
}
```

snappify.com

Penjelasan:

- **#include "pelajaran.h"**: Meng-include header agar program bisa menggunakan tipe data `pelajaran` dan fungsi-fungsi yang sudah dideklarasikan di `pelajaran.h`.
- **Fungsi main**: Ini adalah fungsi utama yang mengeksekusi program.
 - `namapel` dan `kodepel` diinisialisasi dengan nilai "Struktur Data" dan "STD".
 - `create_pelajaran` dipanggil untuk membuat sebuah objek `pelajaran` bernama `pel` dengan data `namapel` dan `kodepel`.
 - `tampil_pelajaran` dipanggil untuk menampilkan data mata pelajaran yang tersimpan di dalam objek `pel`.

Output:

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

Soal 3

```
#include <iostream>

using namespace std;

// Prosedur untuk menampilkan isi array 2D berukuran 3x3
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;
    }
}

// Prosedur untuk menukarkan isi dua array 2D pada posisi tertentu
void tukarIsiArray(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;
}

// Prosedur untuk menukarkan isi dari variabel yang ditunjuk oleh dua pointer
void tukarPointer(int *p1, int *p2) {
    int temp = *p1;
    *p1 = *p2;
    *p2 = temp;
}

int main() {
    // Deklarasi dua array 2D berukuran 3x3
    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}};

    // Dua pointer integer
    int a = 10;
    int b = 20;
    int *p1 = &a;
    int *p2 = &b;

    // Menampilkan isi array sebelum pertukaran
    cout << "Isi Array 1 sebelum pertukaran:" << endl;
    tampilArray(array1);
    cout << "Isi Array 2 sebelum pertukaran:" << endl;
    tampilArray(array2);

    // Menukarkan isi array pada posisi tertentu (misal posisi [1][1])
    tukarIsiArray(array1, array2, 1, 1);

    // Menampilkan isi array setelah pertukaran
    cout << "\nIsi Array 1 setelah pertukaran di posisi [1][1]: " << endl;
    tampilArray(array1);
    cout << "Isi Array 2 setelah pertukaran di posisi [1][1]: " << endl;
    tampilArray(array2);

    // Menampilkan nilai yang ditunjuk oleh pointer sebelum pertukaran
    cout << "\nNilai yang ditunjuk pointer sebelum pertukaran:" << endl;
    cout << "p1 menunjuk nilai: " << *p1 << endl;
    cout << "p2 menunjuk nilai: " << *p2 << endl;

    // Menukarkan nilai yang ditunjuk oleh kedua pointer
    tukarPointer(p1, p2);

    // Menampilkan nilai yang ditunjuk oleh pointer setelah pertukaran
    cout << "\nNilai yang ditunjuk pointer setelah pertukaran:" << endl;
    cout << "p1 menunjuk nilai: " << *p1 << endl;
    cout << "p2 menunjuk nilai: " << *p2 << endl;

    return 0;
}
```

Prosedur `tampilArray`:

- Menampilkan isi array 2D (3x3) dengan mencetak elemen-elemen array dalam bentuk matriks.

Prosedur `tukarIsiArray`:

- Menukarkan nilai pada **posisi tertentu** (baris dan kolom yang ditentukan) antara dua array 2D.

Prosedur `tukarPointer`:

- Menukarkan nilai yang ditunjuk oleh dua buah pointer integer.

Fungsi `main`:

- Mendeklarasikan dua array 2D (`array1` dan `array2`) berukuran 3x3.
- Mendeklarasikan dua pointer (`p1` menunjuk nilai `a` dan `p2` menunjuk nilai `b`).
- Menampilkan array dan pointer sebelum dan sesudah dilakukan pertukaran.

SOAL TP

1. Apa yang dimaksud dengan pointer?

Pointer adalah variabel yang menyimpan alamat memori dari variabel lain. Pointer digunakan untuk mengakses dan memanipulasi data di lokasi memori yang berbeda, memungkinkan efisiensi dalam pengelolaan memori dan pengiriman parameter dalam fungsi.

2. Bagaimana cara menampilkan alamat memori dari suatu variabel dalam program C++? Berikan contoh!

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

int main() {
    int a = 10;
    cout << "Alamat memori dari variabel a: " << &a << endl;
    return 0;
}
```

snappify.com

Alamat memori dari variabel a: 0x7ffde0c5a5bc (contoh alamat, bisa berbeda di setiap run)

3. Bagaimana cara menggunakan pointer dalam program C++? Berikan contoh cara menampilkan nilai yang tersimpan pada suatu alamat melalui pointer!

Untuk menggunakan pointer, kita mendeklarasikan pointer dengan tipe data yang sesuai dan menggunakan operator dereference `*` untuk mengakses nilai yang ditunjuk oleh pointer. Berikut adalah contoh:

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

int main() {
    int a = 20;
    int *p = &a; // p menunjuk ke alamat a

    cout << "Nilai a: " << a << endl;
    cout << "Nilai yang ditunjuk oleh pointer p: " << *p << endl; // mengakses nilai melalui pointer
    return 0;
}
```

snappify.com

Nilai a: 20

Nilai yang ditunjuk oleh pointer p: 20

4. Apa yang dimaksud dengan Abstract Data Type (ADT)?

Abstract Data Type (ADT) adalah model data yang mendefinisikan tipe data berdasarkan perilaku dari operasi yang dapat dilakukan pada tipe tersebut, tanpa memberikan detail implementasi. ADT fokus pada apa yang dapat dilakukan dengan data, bukan bagaimana data tersebut diimplementasikan.

5. Berikan contoh ilustrasi sederhana di dalam dunia nyata, tetapi di luar konteks pemrograman!

KULKAS

ADT: Penyimpanan Makanan

Operasi: Menyimpan, mengeluarkan, dan mendinginkan makanan.

Penjelasan: Dalam kulkas, kita bisa menyimpan berbagai makanan tanpa perlu tahu tentang komponen dan proses pendinginan yang terjadi di dalamnya.

6. Tulis ADT dari bangun ruang kerucut dalam bahasa C++!

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

class Kerucut {
private:
    double jariJari;
    double tinggi;

public:
    // Constructor
    Kerucut(double r, double t) : jariJari(r), tinggi(t) {}

    // Fungsi untuk menghitung volume
    double volume() {
        return (1.0 / 3) * M_PI * pow(jariJari, 2) * tinggi;
    }

    // Fungsi untuk menghitung luas permukaan
    double luasPermukaan() {
        double s = sqrt(pow(jariJari, 2) + pow(tinggi, 2)); // Panjang garis pelukis
        return M_PI * jariJari * (jariJari + s);
    }

    // Fungsi untuk menampilkan informasi kerucut
    void info() {
        cout << "Jari-Jari: " << jariJari << endl;
        cout << "Tinggi: " << tinggi << endl;
        cout << "Volume: " << volume() << endl;
        cout << "Luas Permukaan: " << luasPermukaan() << endl;
    }
};

int main() {
    Kerucut k(5.0, 10.0); // Membuat objek kerucut
    k.info(); // Menampilkan informasi kerucut
    return 0;
}
```

Penjelasan Kode:

- **Class Kerucut:** Mewakili ADT kerucut dengan atribut `jariJari` dan `tinggi`.
- **Fungsi `volume` dan `luasPermukaan`:** Menghitung volume dan luas permukaan kerucut.
- **Fungsi `info`:** Menampilkan informasi tentang kerucut.
- **`main`:** Membuat objek `Kerucut` dan memanggil metode `info` untuk menampilkan data.