LAPORAN PRAKTIKUM

MODUL 3

“ABSTRACT DATA TYPE (ADT)”

Nama :
Alvin Bagus Firmansyah -
2311104070

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

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TUJUAN

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

II. Landasan Teori

Apa itu ADT (Abstract Data Type)?

Bayangkan Anda sedang membangun sebuah rumah. Anda membutuhkan berbagai jenis bahan bangunan seperti batu bata, semen, kayu, dan lain-lain. Setiap bahan bangunan ini memiliki sifat dan cara penggunaannya yang berbeda-beda.

Dalam pemrograman, ADT adalah seperti "bahan bangunan" yang sudah didefinisikan sebelumnya. ADT ini memiliki tipe data tertentu (misalnya, bilangan bulat, bilangan desimal, teks) dan sekumpulan operasi yang bisa dilakukan pada tipe data tersebut (misalnya, penjumlahan, pengurangan, perbandingan).

Contoh ADT:

- Bilangan bulat: Tipe data yang merepresentasikan bilangan bulat. Operasi yang bisa dilakukan: penjumlahan, pengurangan, perkalian, pembagian, dan lain-lain.
- String: Tipe data yang merepresentasikan teks. Operasi yang bisa dilakukan: penggabungan, pencarian, penggantian, dan lain-lain.
- List: Kumpulan data yang urut. Operasi yang bisa dilakukan: menambahkan elemen, menghapus elemen, mencari elemen, dan lain-lain.

Kenapa kita perlu ADT?

- Modularitas: ADT membuat kode program menjadi lebih terstruktur dan mudah dipahami.
- Reusabilitas: ADT dapat digunakan kembali dalam berbagai bagian program.
- Abstraksi: ADT memungkinkan kita untuk fokus pada apa yang bisa dilakukan dengan data, tanpa perlu memikirkan implementasi detailnya.

Bagian-bagian Utama ADT

1. Definisi Tipe: Menentukan jenis data yang akan digunakan dalam ADT.
2. Operasi: Menentukan operasi-operasi apa saja yang bisa dilakukan pada tipe data

tersebut.

- Konstruktor: Membuat objek baru dari tipe data tersebut.
- Selektor: Mengambil nilai dari komponen-komponen objek.
- Pengubah: Mengubah nilai dari komponen-komponen objek.
- Validator: Memeriksa apakah nilai yang diberikan valid untuk tipe data tersebut.
- Destruktor: Menghapus objek dari memori.
- Operator: Operasi seperti penjumlahan, pengurangan, perbandingan, dan lain-lain.

Contoh ADT: Titik (Point)

Misalkan kita ingin membuat ADT untuk merepresentasikan titik pada bidang kartesian.

- Definisi Tipe: Titik memiliki dua komponen: koordinat x dan koordinat y.
- Operasi:
  - Konstruktor: Membuat titik baru dengan koordinat x dan y yang diberikan.
  - Selektor: Mendapatkan nilai koordinat x atau y dari sebuah titik.
  - Pengubah: Mengubah nilai koordinat x atau y dari sebuah titik.
  - Operator: Menghitung jarak antara dua titik, memeriksa apakah dua titik sama, dan lain-lain.

Manfaat ADT

Dengan menggunakan ADT, kita dapat membuat program yang lebih modular, mudah dipelihara, dan mudah diperluas. Misalnya, jika kita ingin membuat ADT untuk segitiga, kita bisa menggunakan ADT Point sebagai komponen dasar segitiga.

Dalam konteks pemrograman berorientasi objek, konsep ADT sangat erat kaitannya dengan kelas dan objek. Kelas adalah blueprint untuk membuat objek, sedangkan objek adalah instansi dari kelas.

Semoga penjelasan ini membantu Anda memahami konsep ADT dengan lebih baik. Jika Anda memiliki pertanyaan lebih lanjut, jangan ragu untuk bertanya!

Apakah Anda ingin membahas contoh ADT lainnya atau konsep terkait ADT yang lebih spesifik?

Beberapa topik yang mungkin menarik untuk dibahas:

- Implementasi ADT dalam bahasa pemrograman lain (misalnya, Java, Python)
- Hubungan antara ADT dengan struktur data
- Penerapan ADT dalam algoritma dan pemrograman
- Konsep ADT dalam pemrograman berorientasi objek

III. GUIDE

1.Kode Pemorgamaan:

main.cpp

```cpp
#include <iostream>

using namespace std;

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

// Fungsi untuk menginput data mahasiswa
void inputMhs(mahasiswa &m);

// Fungsi untuk menghitung rata-rata nilai mahasiswa
float rata2(mahasiswa m);

int main() {
    mahasiswa mhs;
    // Memanggil fungsi untuk input data mahasiswa
    inputMhs(mhs);
    // Menampilkan rata-rata nilai
    cout << "Rata-rata = " << rata2(mhs) << endl;
    return 0;
}

// Implementasi fungsi untuk input data mahasiswa
void inputMhs(mahasiswa &m) {
    cout << "Input NIM: ";
    cin >> m.nim;
    cout << "Input nilai 1: ";
    cin >> m.nilai1;
    cout << "Input nilai 2: ";
    cin >> m.nilai2;
}

// Implementasi fungsi untuk menghitung rata-rata nilai
float rata2(mahasiswa m) {
    // Mengembalikan hasil rata-rata
    return (m.nilai1 + m.nilai2) / 2.0; // Perbaikan: gunakan 2.0 agar hasil pembagian menjadi float
}
```

Hasilnya:

```
"C:\Users\alvin\OneDrive\Dol
Input NIM: 2311104070
Input nilai 1: 100
Input nilai 2: 70
Rata-rata = 85

Process returned 0 (0x0)   execution time : 8.263 s
Press any key to continue.
```

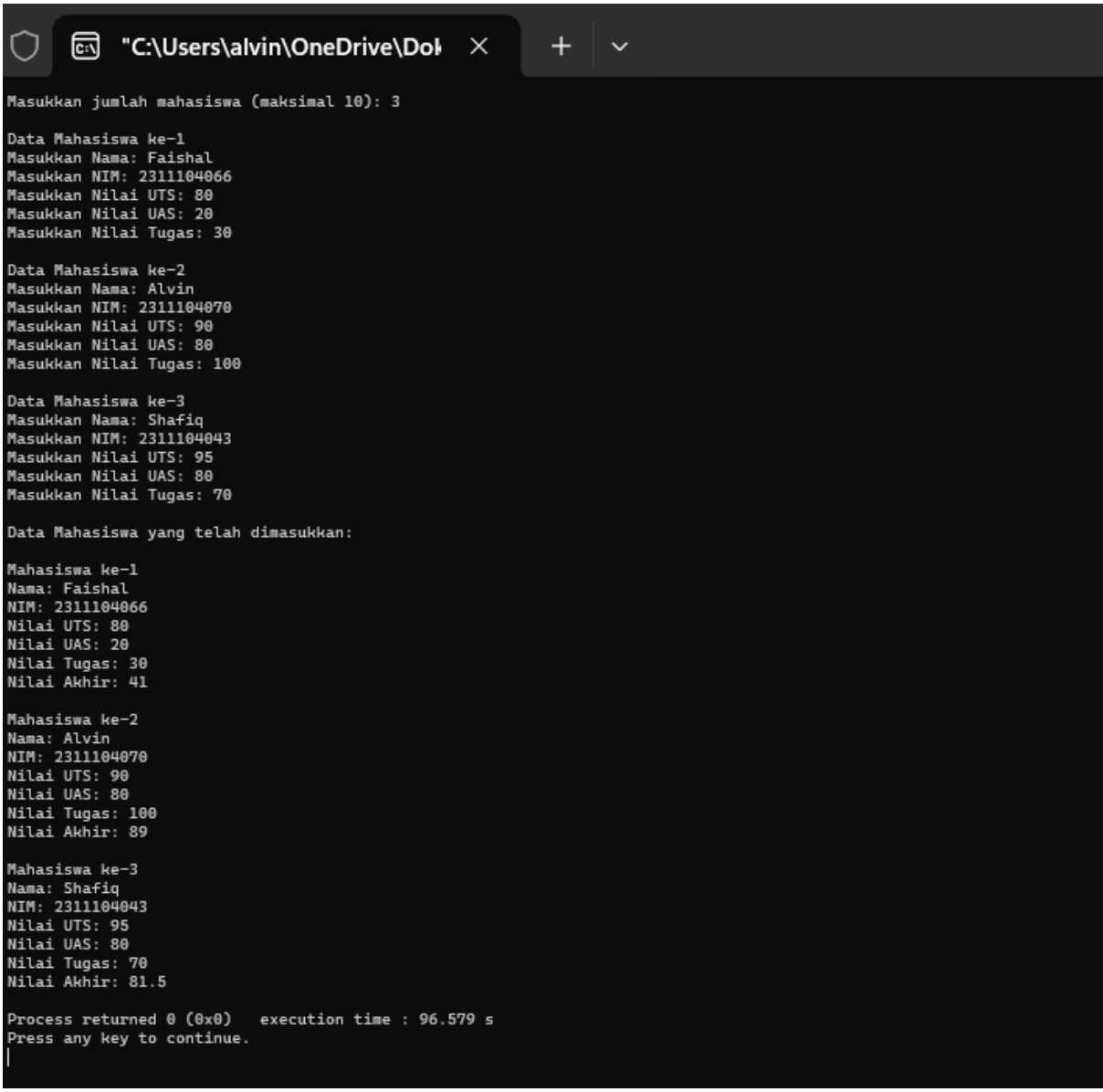
IV. UNGUIDED

1.

main.cpp

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

struct Mahasiswa {
    string nama;
    string nim;
    float uts;
    float uas;
    float tugas;
    float nilaiAkhir;
};

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

// Fungsi untuk input data mahasiswa
void inputMahasiswa(Mahasiswa &mhs) {
    cout << "Masukkan Nama: ";
    cin.ignore();
    getline(cin, mhs.nama);  // Untuk input string yang bisa mengandung spasi
    cout << "Masukkan NIM: ";
    cin >> mhs.nim;
    cout << "Masukkan Nilai UTS: ";
    cin >> mhs.uts;
    cout << "Masukkan Nilai UAS: ";
    cin >> mhs.uas;
    cout << "Masukkan Nilai Tugas: ";
    cin >> mhs.tugas;
    mhs.nilaiAkhir = hitungNilaiAkhir(mhs.uts, mhs.uas, mhs.tugas);
}

// Fungsi untuk menampilkan data mahasiswa
void tampilkanMahasiswa(const Mahasiswa &mhs) {
    cout << "\nNama: " << mhs.nama;
    cout << "\nNIM: " << mhs.nim;
    cout << "\nNilai UTS: " << mhs.uts;
    cout << "\nNilai UAS: " << mhs.uas;
    cout << "\nNilai Tugas: " << mhs.tugas;
    cout << "\nNilai Akhir: " << mhs.nilaiAkhir << endl;
}

int main() {
    Mahasiswa mahasiswa[10]; // Array untuk menyimpan data maksimal 10 mahasiswa
    int jumlahMahasiswa;

    cout << "Masukkan jumlah mahasiswa (maksimal 10): ";
    cin >> jumlahMahasiswa;

    if (jumlahMahasiswa > 10) {
        cout << "Jumlah mahasiswa melebihi batas maksimal!" << endl;
        return 1;
    }

    // Input data mahasiswa
    for (int i = 0; i < jumlahMahasiswa; i++) {
        cout << "\nData Mahasiswa ke-" << i + 1 << endl;
        inputMahasiswa(mahasiswa[i]);
    }

    // Menampilkan data mahasiswa
    cout << "\nData Mahasiswa yang telah dimasukkan:\n";
    for (int i = 0; i < jumlahMahasiswa; i++) {
        cout << "\nMahasiswa ke-" << i + 1;
        tampilkanMahasiswa(mahasiswa[i]);
    }

    return 0;
}
```

Hasil Output::



2. Pelajaran h

main.cpp | include\pelajaran.h | src\pelajaran.cpp | *pelajaran.h | *pelajaran.cpp | *pelajaran.h | *pelajaran.cpp

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

using namespace std;

// Implementasi fungsi create_pelajaran
pelajaran create_pelajaran(string namapel, string kodepel) {
    pelajaran pel;  // Membuat objek pelajaran
    pel.namaMapel = namapel;  // Mengisi namaMapel
    pel.kodeMapel = kodepel;  // Mengisi kodeMapel
    return pel;  // Mengembalikan objek pelajaran
}

// Implementasi prosedur tampil_pelajaran
void tampil_pelajaran(pelajaran pel) {
    cout << "nama pelajaran : " << pel.namaMapel << endl;
    cout << "nilai : " << pel.kodeMapel << endl;
}
```

Pelajaran.cpp

main.cpp | include\pelajaran.h | src\pelajaran.cpp | *pelajaran.h | *pelajaran.cpp | pelajaran.h | *pelajaran.cpp

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

using namespace std;

// Implementasi fungsi create_pelajaran
pelajaran create_pelajaran(string namapel, string kodepel) {
    pelajaran pel;  // Membuat objek pelajaran
    pel.namaMapel = namapel;  // Mengisi namaMapel
    pel.kodeMapel = kodepel;  // Mengisi kodeMapel
    return pel;  // Mengembalikan objek pelajaran
}

// Implementasi prosedur tampil_pelajaran
void tampil_pelajaran(pelajaran pel) {
    cout << "nama pelajaran : " << pel.namaMapel << endl;
    cout << "nilai : " << pel.kodeMapel << endl;
}
```

Main.cpp

main.cpp | include\pelajaran.h | src\pelajaran.cpp | *pelajaran.h | *pelajaran.cpp | pelajaran.h | *pelajaran.cpp

```cpp
#include <iostream>
#include "pelajaran.h" // Memasukkan file header pelajaran.h

using namespace std;

int main() {
    string namapel = "Struktur Data"; // Inisialisasi nama pelajaran
    string kodepel = "STD"; // Inisialisasi kode pelajaran

    pelajaran pel = create_pelajaran(namapel, kodepel); // Panggil fungsi create_pelajaran
    tampil_pelajaran(pel); // Panggil fungsi tampil_pelajaran

    return 0;
}
```

Output:

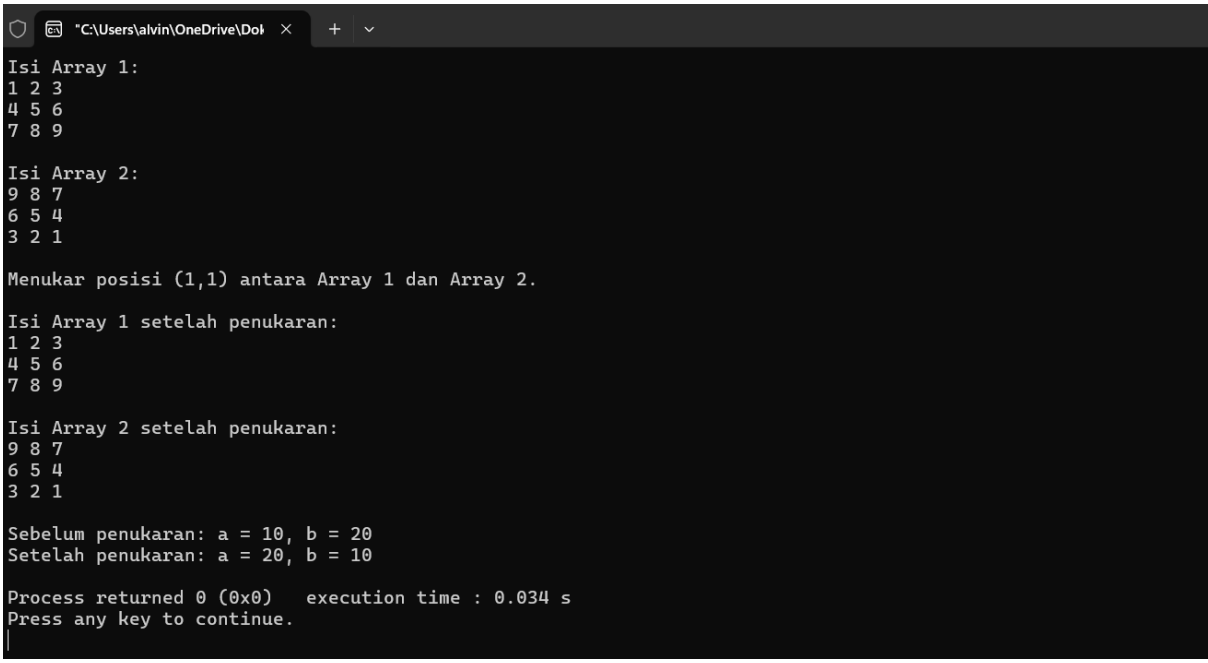
```
"C:\Users\alvin\OneDrive\Dok
nama pelajaran : Struktur Data
nilai : STD

Process returned 0 (0x0)   execution time : 0.027 s
Press any key to continue.
```

3.

main.cpp | include\pelajaran.h | src\pelajaran.cpp | *pelajaran.h | *pelajaran.cpp | pelajaran.h | *pelajaran.cpp | main.cpp

```cpp
#include <iostream>

using namespace std;


void tampilkanArray2D(int array[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << array[i][j] << " ";
        }
        cout << endl;
    }
}


void tukarArray2D(int array1[3][3], int array2[3][3], int baris, int kolom) {

    if (baris >= 0 && baris < 3 && kolom >= 0 && kolom < 3) {
        int temp = array1[baris][kolom];
        array1[baris][kolom] = array2[baris][kolom];
        array2[baris][kolom] = temp;
    } else {
        cout << "Indeks di luar batas!" << endl;
    }
}


void tukarPointer(int *ptr1, int *ptr2) {
    int temp = *ptr1;
    *ptr1 = *ptr2;
    *ptr2 = temp;
}

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}
    };


    cout << "Isi Array 1:" << endl;
    tampilkanArray2D(array1);

    cout << "\nIsi Array 2:" << endl;
    tampilkanArray2D(array2);


    int baris = 1, kolom = 1;
    cout << "\nMenukar posisi (" << baris << "," << kolom << ") antara Array 1 dan Array 2." << endl;
    tukarArray2D(array1, array2, baris, kolom);


    cout << "\nIsi Array 1 setelah penukaran:" << endl;
    tampilkanArray2D(array1);

    cout << "\nIsi Array 2 setelah penukaran:" << endl;
    tampilkanArray2D(array2);


    int a = 10, b = 20;
    int *ptr1 = &a;
    int *ptr2 = &b;

    cout << "\nSebelum penukaran: a = " << a << ", b = " << b << endl;
    tukarPointer(ptr1, ptr2);
    cout << "Setelah penukaran: a = " << a << ", b = " << b << endl;

    return 0;
}
```

Hasilnya:



V. KESIMPULAN

ADT itu seperti kotak khusus untuk menyimpan dan mengelola data di dalam program komputer.