LAPORAN PRAKTIKUM

Modul 3

Abstract Data Type

Disusun Oleh:

Yogi Hafidh Maulana - 2211104061

SE06-02

Dosen :

Wahyu Andi

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

PURWOKERTO

2024

1. Tujuan

- Memahami Konsep Abstract Data Type (ADT)
- Mengidentifikasi Elemen-elemen pada ADT
- Menerapkan ADT dalam Pemrograman

2. Landasan Teori

- Abstract Data Type
  Abstract Data Type (ADT) adalah konsep dasar dalam ilmu komputer yang mengacu pada tipe data yang ditentukan secara abstrak oleh perilaku (operasi) yang dapat dilakukan terhadap data tersebut, bukan oleh implementasi spesifik yang mendasarinya. ADT memberikan cara untuk mengorganisir dan menyembunyikan detail implementasi struktur data sehingga pengguna hanya perlu mengetahui apa yang bisa dilakukan oleh tipe data tersebut, tanpa harus memahami bagaimana cara kerjanya secara internal.

  Konsep Dasar ADT ADT terdiri dari dua komponen utama yaitu:
  1. Definisi Tipe Data: Merupakan deskripsi formal dari struktur data, termasuk variabel-variabel yang membentuknya.
  2. Kumpulan Operasi: Sekumpulan operasi yang dapat dilakukan pada tipe data tersebut, seperti penambahan, penghapusan, modifikasi, serta pencarian elemen.

  Keuntungan Penggunaan ADT
  1. Abstraksi: ADT memisahkan antara definisi tipe data dan implementasi, sehingga memungkinkan pengguna fokus pada penggunaan tipe data tanpa perlu mengetahui detail implementasi.
  2. Reusabilitas: ADT memungkinkan kode lebih modular dan dapat digunakan kembali pada berbagai situasi dengan sedikit atau tanpa perubahan.
  3. Konsistensi: Dengan ADT, program lebih konsisten karena operasi-operasi pada tipe data tertentu diatur dalam satu modul.
  4. Pemeliharaan Mudah: Perubahan pada implementasi ADT tidak akan mempengaruhi kode program yang menggunakan ADT tersebut selama antarmuka (interface) tidak berubah.

3. Guided

a) Abstract Data Type

Code:

.vscode > guided.cpp

```cpp
#include<iostream>

using namespace std;

struct mahasiswa{
    char nim[20];
    int nilai1,nilai2;
};

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

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

```cpp
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;
}
```

Output:

```
Input nim = 2211104061
input nilai = 95
Input nilai  = 90
rata-rata = 92
PS D:\PROJECT\C++ Project\Laprak 3>
```

Deskripsi Program:
Kode di atas menggunakan konsep Abstract Data Type (ADT) dengan mendefinisikan struktur data bernama `mahasiswa`, yang menyimpan informasi penting seperti nomor induk mahasiswa (NIM) dan dua nilai. Fungsi `inputMhs` memungkinkan pengguna untuk memasukkan data mahasiswa melalui input, sedangkan fungsi `rata2` menghitung rata-rata dari dua nilai yang diberikan. Dalam fungsi `main`, program pertama-tama meminta pengguna untuk mengisi data mahasiswa dan kemudian menghitung serta menampilkan rata-rata nilai tersebut. Penggunaan ADT di sini membuat kode lebih terorganisir, modular, dan mudah dipahami, karena data yang relevan dikelompokkan bersama, dan perubahan pada satu bagian kode tidak akan mempengaruhi keseluruhan program.

4. Unguided

a) 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.

Code:

.vscode > soal01.cpp

```cpp
#include <iostream>
#include <string>

using namespace std;

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

void inputMahasiswa(Mahasiswa &m);
float hitungNilaiAkhir(float uts, float uas, float tugas);

int main() {
    const int maxMahasiswa = 10;
    Mahasiswa mhs[maxMahasiswa];
    int jumlahMahasiswa;

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

    if (jumlahMahasiswa > maxMahasiswa) {
        cout << "Jumlah mahasiswa tidak boleh lebih dari 10!" << endl;
        return 1;
    }

    for (int i = 0; i < jumlahMahasiswa; i++) {
        cout << "Data Mahasiswa ke-" << (i + 1) << ":" << endl;
        inputMahasiswa(mhs[i]);
```

```
        mhs[i].nilai_akhir = hitungNilaiAkhir(mhs[i].uts, mhs[i].uas, mhs[i].tugas);
    }

    cout << "\nData Mahasiswa:\n";
    for (int i = 0; i < jumlahMahasiswa; i++) {
        cout << "Nama: " << mhs[i].nama << endl;
        cout << "NIM: " << mhs[i].nim << endl;
        cout << "UTS: " << mhs[i].uts << endl;
        cout << "UAS: " << mhs[i].uas << endl;
        cout << "Tugas: " << mhs[i].tugas << endl;
        cout << "Nilai Akhir: " << mhs[i].nilai_akhir << endl;
        cout << "--------------------" << endl;
    }

    return 0;
}

```

```
void inputMahasiswa(Mahasiswa &m) {
    cout << "Input nama: ";
    cin.ignore(); // Membersihkan newline di buffer
    getline(cin, m.nama);
    cout << "Input NIM: ";
    cin >> m.nim;
    cout << "Input nilai UTS: ";
    cin >> m.uts;
    cout << "Input nilai UAS: ";
    cin >> m.uas;
    cout << "Input nilai tugas: ";
    cin >> m.tugas;
}
```

```
float hitungNilaiAkhir(float uts, float uas, float tugas) {
    return (0.3 * uts) + (0.4 * uas) + (0.3 * tugas);
}

```

Output:

```
Masukkan jumlah mahasiswa (max. 10): 2
Data Mahasiswa ke-1:
Input nama: Samsudin
Input NIM: 2211104035
Input nilai UTS: 80
Input nilai UAS: 80
Input nilai tugas: 90
Data Mahasiswa ke-2:
Input nama: Tarso
Input NIM: 2211104034
Input nilai UTS: 90
Input nilai UAS: 95
Input nilai tugas: 100
```

```
Data Mahasiswa:
Nama: Samsudin
NIM: 2211104035
UTS: 80
UAS: 80
Tugas: 90
Nilai Akhir: 83
--------------------
Nama: Tarso
NIM: 2211104034
UTS: 90
UAS: 95
Tugas: 100
Nilai Akhir: 95
--------------------
```

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;
}
```

Contoh output hasil:

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

Code:

Pelajaran.h

```
#ifndef PELAJARAN_H
#define PELAJARAN_H

#include <string>
using namespace std;

struct pelajaran {
    string namaMapel;
    string kodeMapel;
};

pelajaran create_pelajaran(string namapel, string kodePel);
void tampil_pelajaran(pelajaran pel);

#endif // PELAJARAN_H

```

pelajaran.cpp

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

using namespace std;

pelajaran create_pelajaran(string namapel, string kodePel) {
    pelajaran pel;
    pel.namaMapel = namapel;
    pel.kodeMapel = kodePel;
    return pel;
}

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

```

main.cpp

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

using namespace std;

int main() {
    string namapel = "Struktur Data";
    string kodepel = "STD";
    pelajaran pel = create_pelajaran(namapel, kodepel);
    tampil_pelajaran(pel);

    return 0;
}

```

Output:

```
PS D:\PROJECT\C++ Project\Laprak 3\.vscode> g++ -o main pelajaran.cpp main.cpp
>>
PS D:\PROJECT\C++ Project\Laprak 3\.vscode> ./main
nama pelajaran : Struktur Data
nilai : STD
PS D:\PROJECT\C++ Project\Laprak 3\.vscode>
```

c) 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 STRUKTUR DATA 46

- fungsi/prosedur yang akan menukarkan isi dari variabel yang ditunjuk oleh 2 buah pointer

Code:

```cpp
#include <iostream>

using namespace std;

// Fungsi untuk menampilkan isi array 2D
void tampilkanArray2D(int arr[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }
}

// Fungsi untuk menukarkan isi dua array 2D pada posisi tertentu
void tukarArray2D(int arr1[3][3], int arr2[3][3], int x, int y) {
    int temp = arr1[x][y];
    arr1[x][y] = arr2[x][y];
    arr2[x][y] = temp;
}

// Fungsi untuk menukarkan isi dua variabel yang ditunjuk oleh pointer
void tukarPointer(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

```cpp
int main() {
    // Inisialisasi 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}
    };

    // Menampilkan isi array sebelum pertukaran
    cout << "Array 1 sebelum pertukaran:" << endl;
    tampilkanArray2D(array1);
    cout << "Array 2 sebelum pertukaran:" << endl;
    tampilkanArray2D(array2);

    // Menukarkan elemen pada posisi (1, 1)
    tukarArray2D(array1, array2, 1, 1);

    // Menampilkan isi array setelah pertukaran
    cout << "\nArray 1 setelah pertukaran:" << endl;
    tampilkanArray2D(array1);
```

```
    cout << "Array 2 setelah pertukaran:" << endl;
    tampilkanArray2D(array2);

    // Inisialisasi dua pointer integer
    int* ptr1 = &array1[0][0]; // Pointer menunjuk ke elemen pertama array1
    int* ptr2 = &array2[2][2]; // Pointer menunjuk ke elemen terakhir array2

    // Menampilkan nilai sebelum pertukaran menggunakan pointer
    cout << "\nNilai sebelum pertukaran melalui pointer:" << endl;
    cout << "Nilai yang ditunjuk ptr1: " << *ptr1 << endl;
    cout << "Nilai yang ditunjuk ptr2: " << *ptr2 << endl;

    // Menukarkan nilai yang ditunjuk oleh pointer
    tukarPointer(ptr1, ptr2);

    // Menampilkan nilai setelah pertukaran menggunakan pointer
    cout << "\nNilai setelah pertukaran melalui pointer:" << endl;
    cout << "Nilai yang ditunjuk ptr1: " << *ptr1 << endl;
    cout << "Nilai yang ditunjuk ptr2: " << *ptr2 << endl;

    return 0;
}
```

Output:

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

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

Nilai sebelum pertukaran melalui pointer:
Nilai yang ditunjuk ptr1: 1
Nilai yang ditunjuk ptr2: 1

Nilai setelah pertukaran melalui pointer:
Nilai yang ditunjuk ptr1: 1
Nilai yang ditunjuk ptr2: 1
PS D:\PROJECT\C++ Project\Laprak 3>
```

5. Kesimpulan

Mempelajari Abstract Data Type (ADT) memberikan pemahaman mendalam tentang cara mendefinisikan dan mengimplementasikan tipe data secara abstrak yang terpisah dari detail implementasinya. ADT mengabstraksikan struktur data dan operasi, sehingga memungkinkan programmer untuk fokus pada perilaku data tanpa harus memahami bagaimana data tersebut diimplementasikan secara internal. Hal ini mendukung penerapan konsep modularitas dan reusabilitas, di mana tipe data dan operasinya dipisahkan ke dalam modul-modul yang terstruktur dan dapat digunakan kembali di berbagai konteks program dengan sedikit modifikasi. Selain itu, dengan memisahkan definisi (header) dari implementasi (implementation file), ADT mempermudah proses pemeliharaan dan pengembangan program karena perubahan pada implementasi tidak akan mempengaruhi program utama selama antarmuka tetap konsisten. Penggunaan ADT juga meningkatkan keamanan dan abstraksi data, sehingga pengguna hanya dapat mengakses data melalui operasi yang telah didefinisikan, memastikan bahwa data tidak diubah secara sembarangan. Dengan demikian, mempelajari ADT memberikan fondasi yang kuat dalam desain dan implementasi struktur data yang efektif dan efisien, yang sangat penting dalam pemrograman terstruktur dan berorientasi objek.