LAPORAN PRAKTIKUM

PERTEMUAN 3

Disusun Oleh:

Muhammad Ikhsan Al Hakim (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 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. 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:

```cpp
#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:

```cpp
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:

```cpp
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:

```cpp
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

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:

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

using namespace std;

struct Mahasiswa {
    string nama;
    string nim;
    float uts, uas, tugas, 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 memasukkan data mahasiswa
void inputMahasiswa(Mahasiswa &m) {
    cout << "Input Nama: ";
    cin.ignore(); // Membersihkan buffer agar getline bisa berfungsi
    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;

    m.nilaiAkhir = hitungNilaiAkhir(m.uts, m.uas, m.tugas);
}

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

int main() {
    const int MAX_MHS = 10;  // Maksimal 10 mahasiswa
    Mahasiswa mahasiswa[MAX_MHS];
    int jumlahMahasiswa;

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

    if(jumlahMahasiswa > MAX_MHS) {
        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]);
    }

    // Tampilkan data mahasiswa
    cout << "\nDaftar Mahasiswa:\n";
    for (int i = 0; i < jumlahMahasiswa; i++) {
        cout << "\nMahasiswa ke-" << i + 1 << ":" << endl;
        tampilMahasiswa(mahasiswa[i]);
    }

    return 0;
}

```

Output:

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

Data Mahasiswa ke-1:
Input Nama: ikhsan
Input NIM: 2311104064
Input nilai UTS: 80
Input nilai UAS: 95
Input nilai Tugas: 90

Data Mahasiswa ke-2:
Input Nama: al hakim
Input NIM: 2311104064
Input nilai UTS: 90
Input nilai UAS: 90
Input nilai Tugas: 80
```

```
Daftar Mahasiswa:

Mahasiswa ke-1:

Nama: ikhsan
NIM: 2311104064
Nilai UTS: 80
Nilai UAS: 95
Nilai Tugas: 90
Nilai Akhir: 89

Mahasiswa ke-2:

Nama: al hakim
NIM: 2311104064
Nilai UTS: 90
Nilai UAS: 90
Nilai Tugas: 80
Nilai Akhir: 87
```

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:

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

// Definisi ADT Pelajaran di file "pelajaran.h"
struct pelajaran {
    string namamapel;
    string kodepel;
};

// Fungsi untuk membuat objek Pelajaran baru
pelajaran create_pelajaran(string namapel, string kodepel) {
    pelajaran pel;
    pel.namamapel = namapel;
    pel.kodepel = kodepel;
    return pel;
}

// Prosedur untuk menampilkan detail Pelajaran
void tampil_pelajaran(pelajaran pel) {
    cout << "nama pelajaran: " << pel.namamapel << endl;
    cout << "nilai: " << pel.kodepel << endl;
}

int main() {
    // Membuat objek Pelajaran
    pelajaran pel = create_pelajaran("Struktur Data", "STD");

    // Menampilkan detail Pelajaran
    tampil_pelajaran(pel);

    return 0;
}
```

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:

```cpp
#include <iostream>

using namespace std;

// Fungsi untuk menampilkan isi array 2D
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;
    }
}

// Fungsi untuk menukarkan dua nilai array pada posisi tertentu
void tukarArray(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;
}

// Fungsi untuk menukarkan isi variabel yang ditunjuk oleh dua pointer
void tukarPointer(int *ptr1, int *ptr2) {
    int temp = *ptr1;
    *ptr1 = *ptr2;
    *ptr2 = temp;
}

int main() {
    // Deklarasi 2 buah array 2D berukuran 3x3
    int arr1[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int arr2[3][3] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};

    // Deklarasi 2 pointer integer
    int a = 10, b = 20;
    int *ptr1 = &a;
    int *ptr2 = &b;

    // Menampilkan isi array sebelum ditukar
    cout << "Array 1 sebelum ditukar:" << endl;
    tampilArray(arr1);
    cout << "Array 2 sebelum ditukar:" << endl;
    tampilArray(arr2);

    // Menukarkan isi array di posisi tertentu
    tukarArray(arr1, arr2, 1, 1); // Tukar elemen di baris 1, kolom 1

    // Menampilkan isi array setelah ditukar
    cout << "\nArray 1 setelah ditukar:" << endl;
    tampilArray(arr1);
    cout << "Array 2 setelah ditukar:" << endl;
    tampilArray(arr2);

    // Menampilkan nilai pointer sebelum ditukar
    cout << "\nNilai sebelum pointer ditukar:" << endl;
    cout << "Nilai ptr1 (a): " << *ptr1 << endl;
    cout << "Nilai ptr2 (b): " << *ptr2 << endl;

    // Menukarkan isi dari variabel yang ditunjuk oleh pointer
    tukarPointer(ptr1, ptr2);

    // Menampilkan nilai pointer setelah ditukar
    cout << "\nNilai setelah pointer ditukar:" << endl;
    cout << "Nilai ptr1 (a): " << *ptr1 << endl;
    cout << "Nilai ptr2 (b): " << *ptr2 << endl;

    return 0;
}
```

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.