LAPORAN PRAKTIKUM
Modul 3
ABSTRACT DATA TYPE (ADT)

Disusun Oleh:
Aulia Jasifa Br Ginting 2311104060
S1SE-07-02

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO

2024

1. **Tujuan**
    1. Memahami konsep Abstract Data Type (ADT) penggunaan dalam pemrograman.
2. **Landasan Teori**

    Abstract Data Type (ADT)

    Abstract Data Type (ADT) adalah sebuah tipe data yang didefinisikan oleh perilaku dan operasi yang dapat dilakukan pada data tersebut, bukan oleh implementasinya. ADT adalah koleksi data dan operasi yang dapat dilakukan pada data tersebut, tetapi tidak menjelaskan bagaimana data tersebut disimpan atau diimplementasikan. 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 bahasa C++). Realisasi fungsi dan prosedur harus sedapat mungkin memanfaatkan selector dan konstruktor. Untuk memahami lebih jelas mengenai konsep ADT.

3. Guided

Programnya

```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;
    }
```

Outputnya:

```
Input nim= 2311104060
input nilai =97
input nilai =98
rata-rata = 97
```

4. Unguided

1. 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.

   Programnya

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

using namespace std;

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

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

void inputDataMahasiswa(Mahasiswa &mhs) {
    cout << "Masukkan Nama: ";
    cin.ignore();
    getline(cin, mhs.nama);
    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.nilai_akhir = hitungNilaiAkhir(mhs.uts, mhs.uas, mhs.tugas);
}

void tampilkanDataMahasiswa(const Mahasiswa &mhs) {
    cout << setw(20) << left << mhs.nama;
    cout << setw(15) << left << mhs.nim;
    cout << setw(10) << left << mhs.uts;
    cout << setw(10) << left << mhs.uas;
    cout << setw(10) << left << mhs.tugas;
    cout << setw(10) << left << fixed << setprecision(2) << mhs.nilai_akhir << endl;
}

int main() {
    Mahasiswa daftarMahasiswa[10];
    int jumlahMahasiswa = 0;
    char pilihan;

    do {
        if (jumlahMahasiswa < 10) {
            cout << "\nMasukkan data mahasiswa ke-" << jumlahMahasiswa + 1 << endl;
            inputDataMahasiswa(daftarMahasiswa[jumlahMahasiswa]);
            jumlahMahasiswa++;
        } else {
            cout << "Jumlah maksimum mahasiswa (10) telah tercapai." << endl;
            break;
        }

        cout << "Apakah Anda ingin memasukkan data mahasiswa lagi? (y/n): ";
        cin >> pilihan;
    } while (pilihan == 'y' || pilihan == 'Y');

    cout << "\nDaftar Mahasiswa:\n";
    cout << setw(20) << left << "Nama";
    cout << setw(15) << left << "NIM";
    cout << setw(10) << left << "UTS";
    cout << setw(10) << left << "UAS";
    cout << setw(10) << left << "Tugas";
    cout << setw(10) << left << "Nilai Akhir" << endl;
    cout << string(75, '-') << endl;

    for (int i = 0; i < jumlahMahasiswa; i++) {
        tampilkanDataMahasiswa(daftarMahasiswa[i]);
    }

    return 0;
}
```

   Outputnya:

```
Masukkan data mahasiswa ke-1
Masukkan Nama:  Aulia Jasifa
Masukkan NIM: 2311104060
Masukkan nilai UTS: 98
Masukkan nilai UAS: 99
Masukkan nilai Tugas: 95
Apakah Anda ingin memasukkan data mahasiswa lagi? (y/n): n

Daftar Mahasiswa:
Nama                NIM            UTS       UAS       Tugas     Nilai Akhir
---------------------------------------------------------------------------
Aulia Jasifa        2311104060     98        99        95        97.50
```

2. 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;
}
```

Programnya
File Pelajaran.h

```cpp
#ifndef PELAJARAN_H
#define PELAJARAN_H

#include <string>

struct pelajaran {
    std::string namaMpel;
    std::string kodeMpel;
};

pelajaran create_pelajaran(std::string namapel, std::string kodepel);
void tampil_pelajaran(pelajaran pel);

#endif // PELAJARAN_H
```

File Pelajaran.cpp

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

pelajaran create_pelajaran(std::string namapel, std::string kodepel) {
    pelajaran pel;
    pel.namaMpel = namapel;
    pel.kodeMpel = kodepel;
    return pel;
}

void tampil_pelajaran(pelajaran pel) {
    std::cout << "nama pelajaran : " << pel.namaMpel << std::endl;
    std::cout << "kode : " << pel.kodeMpel << std::endl;
}
```

File Main.cpp

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

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

    pelajaran pel = create_pelajaran(namapel, kodepel);
    tampil_pelajaran(pel);

    return 0;
}
```

Outputnya:

```
nama pelajaran : Struktur Data
kode : STD
```

3. 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
   - Fungsi/ prosedur yang akan menukarkan isi dari variabel yang ditunjuk oleh 2 buah *pointer*

   Outputnya:

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

Isi Array 2:
  10  11  12
  13  14  15
  16  17  18

Menukar elemen pada posisi (1,1) antara Array 1 dan Array 2
Isi Array 1 setelah penukaran:
   1   2   3
   4  14   6
   7   8   9

Isi Array 2 setelah penukaran:
  10  11  12
  13   5  15
  16  17  18

Nilai yang ditunjuk pointer sebelum penukaran:
ptr1: 100
ptr2: 200

Menukar nilai yang ditunjuk oleh pointer
Nilai yang ditunjuk pointer setelah penukaran:
ptr1: 200
ptr2: 100
```

Programnya

```cpp
#include <iostream>
#include <iomanip>

using namespace std;

// Fungsi untuk menampilkan isi array 2D
void tampilkanArray(int arr[3][3], const string& namaArray) {
    cout << "Isi " << namaArray << ":" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << setw(4) << arr[i][j];
        }
        cout << endl;
    }
    cout << endl;
}

// Fungsi untuk menukar isi dari 2 array 2D pada posisi tertentu
void tukarElemen(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 menukar isi dari variabel yang ditunjuk oleh 2 buah pointer
void tukarPointer(int* ptr1, int* ptr2) {
    int temp = *ptr1;
    *ptr1 = *ptr2;
    *ptr2 = temp;
}

int main() {
    // Inisialisasi 2 buah array 2D integer berukuran 3x3
    int array1[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int array2[3][3] = {{10, 11, 12}, {13, 14, 15}, {16, 17, 18}};

    // Inisialisasi 2 buah pointer integer
    int nilai1 = 100, nilai2 = 200;
    int *ptr1 = &nilai1, *ptr2 = &nilai2;

    // Menampilkan isi awal array
    tampilkanArray(array1, "Array 1");
    tampilkanArray(array2, "Array 2");

    // Menukar elemen pada posisi (1,1) antara array1 dan array2
    cout << "Menukar elemen pada posisi (1,1) antara Array 1 dan Array 2" << endl;
    tukarElemen(array1, array2, 1, 1);

    // Menampilkan isi array setelah penukaran
    tampilkanArray(array1, "Array 1 setelah penukaran");
    tampilkanArray(array2, "Array 2 setelah penukaran");

    // Menampilkan nilai yang ditunjuk oleh pointer sebelum penukaran
    cout << "Nilai yang ditunjuk pointer sebelum penukaran:" << endl;
    cout << "ptr1: " << *ptr1 << endl;
    cout << "ptr2: " << *ptr2 << endl << endl;

    // Menukar nilai yang ditunjuk oleh pointer
    cout << "Menukar nilai yang ditunjuk oleh pointer" << endl;
    tukarPointer(ptr1, ptr2);

    // Menampilkan nilai yang ditunjuk oleh pointer setelah penukaran
    cout << "Nilai yang ditunjuk pointer setelah penukaran:" << endl;
    cout << "ptr1: " << *ptr1 << endl;
    cout << "ptr2: " << *ptr2 << endl;

    return 0;
}
```

5. **Kesimpulan**

   Dapat memahami bagaimana konsep ADT diterapkan dalam pemrograman, mempelajari cara memisahkan antara representasi data dengan operasinya, yang merupakan prinsip penting dalam pengembangan software yang modular dan mudah dipelihara.