LAPORAN PRAKTIKUM

PERTEMUAN 3

03_ABSTRACT DATA TYPE (ADT)

Nama :

Ilham Lii Assidaq (2311104068)

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. TOOL

Dev C++

III. DASAR TEORI

ADT (Abstract Data Type) adalah jenis data yang didefinisikan bersama dengan operasi-operasi dasar yang dapat dilakukan pada jenis data tersebut. Selain itu, ADT yang lengkap juga mencakup definisi kondisi atau aturan tetap dari tipe data tersebut, serta aksioma-aksioma yang mengatur operasinya. ADT bersifat statis, artinya definisinya tidak berubah.

Definisi type dari sebuah ADT dapat mengandung sebuah definisi ADT lain. Misalnya :
3. ADT waktu yang terdiri dari ADT JAM dan ADT DATE
4. Garis terdiri dari duah buah ADT POINT
SEGI4 yang terdiri dari pasangan dua buah POINT (Top,Left) dan (Bottom,Right)

Dalam bahasa C, *type* merujuk pada tipe data yang didefinisikan, seperti *struct*, dan *primitif* dalam pemrograman prosedural berupa fungsi atau prosedur, yang kemudian dikelompokkan berdasarkan kategori tertentu, seperti:

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. 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)
Kode program ditulis menggunakan bahasa yang sesuai, dalam hal ini C++. Implementasi fungsi dan prosedur sebaiknya memaksimalkan penggunaan selector dan konstruktor

IV. GUIDED

1. Code

```cpp
#include <iostream>

using namespace std;

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

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

int main(){
   mahasiswa Mahasiswa;
   inputMahasiswa(Mahasiswa);
   cout << "Average = " << rata2(Mahasiswa) << endl;
   return 0;
}

void inputMahasiswa(mahasiswa &m){
   cout << "Input NIM : ";
   cin >> (m).nim;
   cout << "Input Nilai 1 : ";
   cin >> (m).nilai1;
   cout << "Input Nilai 2 : ";
   cin >> (m).nilai2;
}

float rata2(mahasiswa m){
   return((m).nilai1 + (m).nilai2) / 2;
}
```

2. Output

```
D:\ITT SM 3\03_ABSTRAKSI\r
Input NIM : 2311104068
Input Nilai 1 : 90
Input Nilai 2 : 77
Average = 83

--------------------------------
Process exited after 22.21 seconds with return value 0
Press any key to continue . . .
```

V. UNGUIDED

1. Menghitung Nilai

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

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

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

int main() {
    Mahasiswa mahasiswa[10];
    int jumlah;

    cout << "Masukkan jumlah MHS Max:10: ";
    cin >> jumlah;

    for (int i = 0; i < jumlah; i++) {
        cout << "\nMahasiswa " << i + 1 << "\nNama: "; cin >> mahasiswa[i].nama;
        cout << "NIM: "; cin >> mahasiswa[i].nim;
        cout << "UTS: "; cin >> mahasiswa[i].uts;
        cout << "UAS: "; cin >> mahasiswa[i].uas;
        cout << "Tugas: "; cin >> mahasiswa[i].tugas;

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

    cout << "\nData Mahasiswa:\n";
    for (int i = 0; i < jumlah; i++)
        cout << mahasiswa[i].nama << " (" << mahasiswa[i].nim << ") - Nilai Akhirnya adalah: " << mahasiswa[i].nilai_akhir << endl;

    return 0;
}
```

```
D:\ITT SM 3\03_ABSTRAKSI\U
Masukkan jumlah MHS Max:10: 3

Mahasiswa 1
Nama: yangbenengdor
NIM: 234
UTS: 90
UAS: 80
Tugas: 70

Mahasiswa 2
Nama: gilaaloo
NIM: 2345
UTS: 60
UAS: 50
Tugas: 40

Mahasiswa 3
Nama: spontanuhuy
NIM: 30
UTS: 20
UAS: 10
Tugas: 30

Data Mahasiswa:
yangbenengdor (234) - Nilai Akhirnya adalah: 80
gilaaloo (2345) - Nilai Akhirnya adalah: 50
spontanuhuy (30) - Nilai Akhirnya adalah: 19
```

2. Pelajaran.h.cpp

```cpp
#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.cpp

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

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 << "Kode pelajaran: " << 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

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

3. Menukar Array

```
4
5   void tampil_array(int arr[3][3]) {
6       for (int i = 0; i < 3; ++i) {
7           for (int j = 0; j < 3; ++j) {
8               cout << arr[i][j] << " ";
9           }
10          cout << endl;
11      }
12  }
13
14  void tukar_array(int arr1[3][3], int arr2[3][3], int x, int y) {
15      if (x < 0 || x >= 3 || y < 0 || y >= 3) {
16          cout << "Posisi error." << endl;
17          return;
18      }
19      int temp = arr1[x][y];
20      arr1[x][y] = arr2[x][y];
21      arr2[x][y] = temp;
22  }
23
24  int main() {
25
26      int array1[3][3] = {
27          {1, 2, 3},
28          {4, 5, 6},
29          {7, 8, 9}
30      };
31
32      int array2[3][3] = {
33          {9, 8, 7},
34          {6, 5, 4},
35          {3, 2, 1}
36      };
37
38      int *ptr1 = &array1[0][0];
39      int *ptr2 = &array2[0][0];
40
41      cout << "Array 1:" << endl;
42      tampil_array(array1);
43
44      cout << "Array 2:" << endl;
45      tampil_array(array2);
46
47      tukar_array(array1, array2, 1, 1);
48
49      cout << "\nSetelah menukar elemen pada posisi (1, 1):" << endl;
50      cout << "Array 1:" << endl;
51      tampil_array(array1);
52
53      cout << "Array 2:" << endl;
54      tampil_array(array2);
55
56      return 0;
57  }
```

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

Setelah menukar elemen pada posisi (1, 1):
Array 1:
1 2 3
4 5 6
7 8 9
Array 2:
9 8 7
6 5 4
3 2 1

--------------------------------
Process exited after 0.1181 seconds with return value 0
Press any key to continue . . .
```

VI. KESIMPULAN

Pada praktikum modul 3 ini, mempelajari bahwa ADT merupakan jenis data yang mendefinisikan struktur dan operasi dasar yang dapat dilakukan terhadap data tersebut, serta pentingnya penerapan konsep modular dalam pemrograman. Pada praktikum ini, implementasi ADT berhasil dalam menghitung nilai dan menukar elemen array, sehingga meningkatkan pemahaman tentang struktur data dan keterampilan pemrograman praktis yang dapat diterapkan dalam pemrograman-pemrograman lainnya.