Modul 3 ABSTRACT DATA TYPE (ADT)

TUJUAN PRAKTIKUM
1. Memahami konsep Abstract Data Type (ADT) dan penggunaannya dalam pemrograman.

3.1 *Abstract Data Type (ADT)*

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 :

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*)

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, perhatikan ilustrasi berikut.

Algoritma	C++

Program coba_ADT
 	
Type
mahasiswa <
nim : char[10]
nilai1,nilai2 : integer
 	
Kamus
mhs : mahasiswa
 	
procedure inputMhs(input/output
m : mahasiswa)
function rata2(input:
 	
m : mahasiswa) : real
 	
Algoritma
inputMhs(mhs)
output(rata2(mhs))
 	
procedure inputMhs(input/output
m : mahasiswa)
kamus
algoritma
input(m.nim, m.nilai1, m.nilai2)
 	
function rata2(input:
m : mahasiswa) : real
kamus
algoritma
→ (m.nilai1 + m.nilai1) / 2
	#include <iostream>
	#include <conio.h>
	#include <stdlib.h>
	
	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 nama = ";
	cin >> (m).nim;
	cout << "input nilai = ";
	cin >> (m).nilai1;
	cout << "input nilai2 = ";
	cin >> (m).nilai2;
	}
	float rata2(mahasiswa m){
	return (m.nilai1+m.nilai2)/2;
	}

Definisi/ Spesifikasi Type dan Primitif / Header fungsi (&.h)

Body/ relisasi dari primitif (&.c)

Untuk menerapkan konsep ADT, kita harus memisah deklarasi tipe, variabel, dan fungsi dari program ke dalam sebuah file.h dan memisah definisi fungsi dari program ke sebuah file.cpp. Sehingga jika kita menerapkan konsep ADT berdasarkan contoh program di atas, bentuk code program akan dipisah menjadi seperti berikut.

Algoritma	C++

Program coba_ADT
 	
Type
mahasiswa <
nim : char[10]
nilai1,nilai2 : integer
 	
Kamus
mhs : mahasiswa
 	
procedure inputMhs(i/o
m : mahasiswa)
function rata2(input:
m : mahasiswa) : real
 	
Algoritma
inputMhs(mhs)
output(rata2(mhs))
procedure inputMhs(input/output
m : mahasiswa)
kamus
algoritma
input(m.nim, m.nilai1, m.nilai2)
	mahasiswa.h
	
	#ifndef MAHASISWA_H_INCLUDED
	#define MAHASISWA_H_INCLUDED
	struct mahasiswa{
	char nim[10];
	int nilai1, nilai2;
	};
	
	void inputMhs(mahasiswa &m);
	float rata2(mahasiswa m);
	#endif // MAHASISWA_H_INCLUDED
	
	mahasiswa.cpp
	
	void inputMhs(mahasiswa &m){
	cout << "*input* nama = ";
	cin >> (m).nim;
	cout << "*input* nilai = ";
	cin >> (m).nilai1;
	cout << "*input* nilai2 = ";
	cin >> (m).nilai2;
	}

```
function rata2(input:
   m : mahasiswa) : real
kamus
algoritma
   → (m.nilai1 + m.nilai1) / 2
```

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

main.cpp

```
#include <iostream>
#include <conio.h>
#include <stdlib.h>
#include "mahasiswa.cpp"

using namespace std;

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

3.2 *Latihan*

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.

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;
}
```

Gambar 3-1 Main.cpp pelajaran

Contoh *output* hasil:

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

Gambar 3-2 *output* pelajaran

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*