**LAPORAN PRAKTIKUM**

**Modul 3**

**Abstract Data Type (ADT)**



**Disusun Oleh:**

**Ryan Gabriel Togar Simamora (2311104045)**

**Kelas : SE0702**

**Dosen :**

**Wahyu Andi Saputra**

**PROGRAM STUDI S1 SOFTWARE ENGINEERING**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

1. **Tujuan**

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

1. **Landasan Teori**

ADT (Abstract Data Type) adalah konsep yang menggambarkan tipe data dan sekumpulan operasi yang dapat dilakukan terhadap tipe data tersebut, tanpa memperhatikan bagaimana implementasinya secara internal. ADT memberikan abstraksi yang memisahkan detail implementasi dari cara penggunaannya, sehingga fokus pengguna lebih pada fungsi dan operasional yang disediakan daripada cara kerja di baliknya.

Dalam ADT, kita mendefinisikan tipe data yang mencakup struktur data, serta operasi dasar (primitif) yang dapat dilakukan terhadap data tersebut. Operasi ini bisa berupa pembentukan, pengubahan, penghapusan, serta interaksi dengan data melalui input dan output. ADT juga bisa mencakup aturan-aturan atau kondisi tertentu yang harus dipenuhi, seperti invarian dan aksioma yang mengatur bagaimana tipe data itu berfungsi dalam berbagai situasi.

Dalam konteks C++, tipe data di dalam ADT sering diterjemahkan menggunakan struktur (struct) atau kelas (class). Operasi primitif diimplementasikan melalui fungsi atau prosedur yang berperan sebagai konstruktor, selektor, atau prosedur pengubah, tergantung pada fungsinya.

**Ada beberapa jenis operasi yang umum dalam ADT, antara lain:**

* Konstruktor/Kreator: Operasi yang bertanggung jawab membentuk atau menginisialisasi objek dari tipe data tersebut.
* Selektor: Operasi untuk mengakses elemen-elemen atau komponen-komponen dari tipe data tersebut.
* Pengubah: Operasi untuk memodifikasi nilai elemen atau komponen dari objek bertipe ADT.
* Validator: Digunakan untuk memeriksa apakah data sesuai dengan aturan-aturan yang telah ditetapkan.
* Destruktor/Dealokator: Operasi yang digunakan untuk membersihkan atau menghapus objek yang telah diciptakan, serta melepaskan memori yang digunakan.
* Input/Output: Operasi yang menangani antarmuka untuk membaca dan menulis data dari atau ke perangkat input/output.
* Operator relasional: Operasi yang memungkinkan perbandingan antar objek, seperti membandingkan apakah objek lebih besar, lebih kecil, atau sama.
* Aritmatika: Operasi yang mendukung perhitungan pada tipe data tertentu, terutama untuk data numerik.
* Konversi: Operasi yang mengubah tipe data dari ADT ke tipe dasar atau sebaliknya.

**Dalam implementasinya, ADT biasanya dipisahkan ke dalam dua bagian utama:**

* Spesifikasi/Definisi Tipe dan Operasi: Ini biasanya ditulis dalam file header (.h), yang mencakup deklarasi tipe data dan deskripsi dari operasi-operasi yang berlaku. Spesifikasi ini hanya menyatakan apa yang akan dilakukan oleh operasi, tanpa menunjukkan bagaimana operasi tersebut diimplementasikan.
* Implementasi Primitif: Bagian ini ditulis dalam file terpisah (.cpp) yang berisi kode program aktual untuk mewujudkan operasi-operasi yang telah didefinisikan. Di sinilah detail dari bagaimana operasi bekerja secara internal dijelaskan.

Contoh : A screenshot of a computer program

Description automatically generated

Untuk Source codenya lebih jelas dibawah ini :

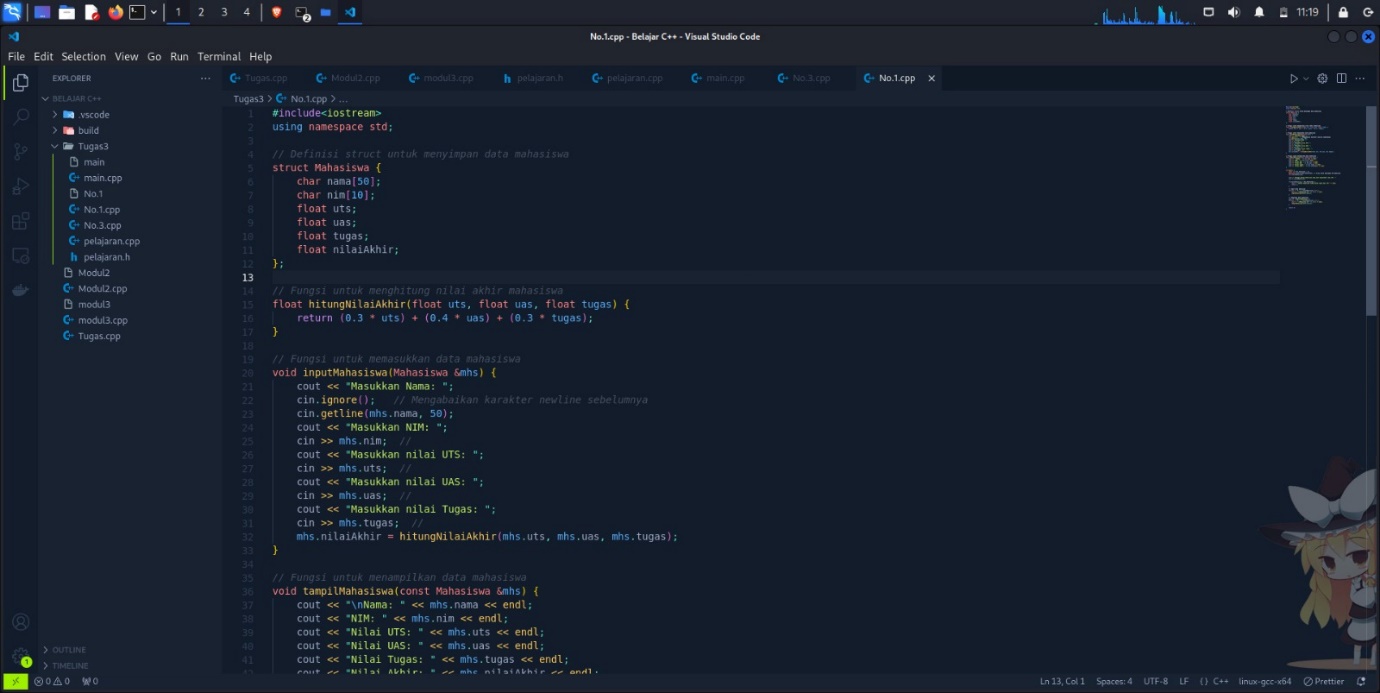
A screen shot of a computer code

Description automatically generated

Dengan ADT, kita dapat membuat program yang lebih modular dan mudah dipahami, karena kita bisa memisahkan antara definisi tipe data dan operasionalnya, sehingga perubahan dalam implementasi internal tidak akan mempengaruhi penggunaan di luar modul tersebut.

1. **Unguided**
2. 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.

Jawab :



Output :

A screenshot of a computer

Description automatically generated

Untuk Source codenya lebih jelas dibawah ini :

A screen shot of a computer code

Description automatically generated

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;**

**}**

Contoh *output* hasil:



Gambar 3-2 output pelajaran

Jawab :

A screenshot of a computer program

Description automatically generatedFile pelajaran.cpp

Gambar 3-1 Main.cpp pe

File pelajaran.h

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

File main.cpp

Output :

A screen shot of a computer

Description automatically generated

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

Jawab :

**A screenshot of a computer program

Description automatically generated**

Untuk Sorce codenya lebih jelasnya dibawah ini :

**A screen shot of a computer code

Description automatically generated**

Output :

**A screenshot of a computer program

Description automatically generated**

**IV . Kesimpulan**

Abstract Data Type (ADT) merupakan konsep penting dalam pemrograman yang memisahkan antara definisi logis dan implementasi teknis suatu tipe data. Dengan menggunakan ADT, kita dapat mendefinisikan tipe data secara abstrak beserta operasi-operasi dasar yang bisa dilakukan, tanpa perlu memikirkan bagaimana tipe data tersebut diimplementasikan. Pendekatan ini memungkinkan kita untuk membuat program yang lebih terstruktur, modular, dan mudah dikelola.

**Dalam praktikum ini, ADT diimplementasikan melalui dua bagian utama:**

1. Spesifikasi Type dan Operasi (header .h) yang mendefinisikan struktur tipe data dan operasi dasar.

2. Body atau Implementasi (source .cpp) yang menguraikan detail dari fungsi-fungsi atau prosedur yang digunakan untuk memanipulasi tipe data tersebut.

**Dengan menerapkan ADT, kita dapat:**

1. Mempermudah pengelolaan kode.

2. Memisahkan antara apa yang dilakukan (spesifikasi) dan bagaimana itu dilakukan (implementasi).

3. Meningkatkan fleksibilitas dan keterbacaan kode.

Secara keseluruhan, penggunaan ADT dalam pengembangan program membantu menciptakan kode yang lebih rapi, terstruktur, dan dapat diperluas di masa mendatang, sehingga sangat bermanfaat dalam skala program yang lebih besar.