

TUGAS KELOMPOK
ALGORITMA DAN PEMROGRAMAN
A11.4218



REKURSIF

Dosen Pengampu :
Danang Wahyu Utomo, M.Kom

Disusun Oleh :
Farras Adhani Zayn
A11.2022.14698

UNIVERSITAS DIAN NUSWANTORO
MEI 2023

PENDAHULUAN

A. Pengertian *Abstract* atau Abstraksi

Pada setiap bahasa pemrograman konvensional, sering kali ditemukan bahwa para pengembang menawarkan sebuah bantuan yang kuat terhadap hal yang berhubungan dengan abstraksi dimana contohnya adalah seperti Fungsi dan Prosedur. Ketika seorang *programmer* membuat sebuah fungsi atau prosedur, hal yang mereka fokuskan pada saat itu (atau seharusnya) hanyalah apa yang terjadi pada lingkup fungsi atau prosedur tersebut. (1974, B. Liskov & Stephen Z).

Maka jika dia hanya seharusnya fokus kepada fungsi atau prosedur yang ia buat, hal ini berarti dia tidak seharusnya fokus ke pada program inti atau yang lebih dikenal fungsi `main()` pada C++.

Jika kita menarik kesimpulan pada pernyataan tersebut, dapat diketahui bahwa *Abstract* atau Abstraksi adalah dimana kita hanya fokus kepada hal yang kita lakukan atau tidak memikirkan hal yang lain.

B. Pengertian *Abstract Data Type (ADT)*

Jika sebelumnya kita sudah mengetahui arti Abstraksi, maka jika terdapat problematika apa itu *Abstract Data Type*. Kesimpulan yang dapat kita Tarik adalah bahwasanya sebuah Data Type memiliki titik fokusnya tersendiri seperti contoh dimana ketika kita memiliki *Integer* ataupun *Integer Array*, maka yang akan kita lakukan terhadap variabel dengan tipe data tersebut adalah yaitu proses Aritmetika (Pada umumnya).

Menurut Barbara Liskov dan Stephen Zilles pada Artikel mereka yang berjudul *Programming with Abstract Data Types (ADT)*, mereka menyatakan bahwasanya *ADT* tidak memedulikan bagaimana objek data tersebut direpresentasikan, dan mereka menganggap objek tersebut sebagai operasi atomik meskipun beberapa mesin instruksi mungkin diperlukan untuk melakukannya.

PEMBAHASAN

A. Bagaimana Konsep *ADT* Berjalan?

Pada sebuah pemrograman, terdapat beberapa tipe data yang berbeda jenis dan kegunaannya. Dasarnya, tidak mungkin kita melakukan aritmetika dengan variabel bertipe data *string*. `String 1 + String 1` tentu tidak akan sama dengan 2, begitu juga apabila berbeda tipe data. Maka untuk mengatasi hal ini, terdapat konsep *ADT* yang dimana ia mengatur bahwasanya setiap tipe data memiliki fokusnya masing-masing.

Ketika kita hendak mendapatkan hasil dari 30 dibagi dengan 2, maka kita seharusnya menggunakan tipe data *Integer*. Karena tidak mungkin kita membagi *string* 30 dengan *string* 2, jikalau *string 1 + string 2* adalah 12. Maka hal ini tentu akan menimbulkan masalah.

B. Mengapa Harus Ada *ADT*?

Tentu saja hal ini dikarenakan sebuah problematika dimana tipe data *char* tidak mungkin dan tidak lazim untuk dijumlahkan dengan tipe data *float*. Begitu juga dengan sebuah tipe data *int** yang dijumlahkan dengan tipe data *int*, jika hal ini dilakukan pada Bahasa C++. Maka yang timbul hanyalah sebuah *error* dan *warning*.

Maka untuk menghindari semua ini, harus diterapkan yang namanya konsep *ADT* agar kita menggunakan tipe data sesuai dengan kebutuhan kita.

C. Apa Hubungannya *ADT* dan OOP

Jika *ADT* memiliki hubungan erat dengan sebuah tipe data, maka jika kita memperhatikan OOP dengan seksama. Maka dapat kita tarik sebuah kesimpulan.

Yaitu adalah, *ADT* berhubungan sangat erat dengan fungsi dan prosedur yang terdapat pada sebuah *Class*. Melihat dari untuk mengapa kita membuat fungsi tersebut, maka sudah sepastinya kita menerapkan *ADT* agar tidak terjadi sebuah

problematika yang lain. Contohnya adalah, jika kita membuat fungsi akar() maka sudah seharusnya kita menggunakan tipe data float dikarenakan akar tidak selalu menghasilkan Integer. Begitu juga ketika kita hendak mengembalikan nilai *true* atau *false*, maka sudah selazimnya kita menggunakan tipe data bool daripada tipe data string. Karena keduanya memiliki nilai yang tentu saja berbeda.

D. Contoh Penerapan ADT dengan Perancangan Program Sederhana

Dalam penerapan ADT dengan perancangan program sederhana, kita dapat membuat sebuah program yang menampilkan nama pengguna dan umurnya berdasarkan tahun lahir.

Pertama-tama kita harus membuat kamus variabel terlebih dahulu pada *class guessPerson*. Jika terdapat file *header.h*, maka *class* di deklarasi pada file *header.h* dan bukan pada *pustaka.cpp* seperti pada gambar 1.1 berikut.



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

class guessPerson {
protected:
    // Declare variabel yang hanya bisa digunakan pada class dan turunannya
    // Variabel tahun sekarang
    int currentYear = 2023;
    // Variabel umur
    int personAge;

public:
    // Declare variabel yang dapat digunakan oleh program inti
    // Variabel nama orang
    string personName;
    // Variabel tahun lahir
    int bornYear;

    int getAge();
};
```

Tentu ketika kita mendeklarasikan variabel, tentu perlu diingat bahwasanya tipe data dari variabel harus sesuai dengan operasi apa yang akan ia lakukan. Karena hal tersebutlah yang membuat OOP memiliki hubungan dengan konsep ADT.

Kemudian kita buat fungsi-fungsi yang kita butuhkan (Perlu diingat sesuai dengan kebutuhan dan nilai yang akan dikembalikan). Jika *class* di declare pada *header.h*, maka fungsi di declare pada *pustaka.cpp* dengan cara seperti gambar 1.2 berikut



```
#include "header.h"

// Fungsi untuk menghitung umur
int guessPerson::getAge() {
    this->personAge = this->currentYear - this->bornYear;
    return this->personAge;
}
```

Ketika sudah kita *declare* sebagai berikut, maka kita tinggal menambahkan dan memanggilnya pada file *main.cpp* seperti pada gambar 1.3 berikut.



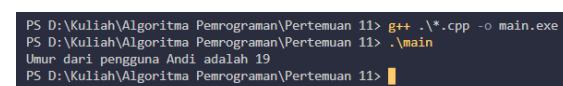
```
#include "header.h"

int main() {
    guessPerson Person;

    Person.personName = "Andi";
    Person.bornYear = 2004;

    cout << "Umur dari pengguna " << Person.personName <<
    " adalah " << Person.getAge() << endl;
    return 0;
}
```

Dengan menerapkan konsep ADT, kita dapat dengan aman dan tenang dalam melakukan operasi-operasi pada program tersebut. Sehingga kita dapat meminimalisir *error* yang disebabkan oleh perbedaan tipe data dan perbedaan hasil. Hasil dari program tersebut dapat dilihat pada gambar 1.4 berikut.



```
PS D:\Kuliah\Algoritma Pemrograman\Pertemuan 11> g++ *.cpp -o main.exe
PS D:\Kuliah\Algoritma Pemrograman\Pertemuan 11> .\main
Umur dari pengguna Andi adalah 19
PS D:\Kuliah\Algoritma Pemrograman\Pertemuan 11>
```

DAFTAR PUSTAKA

1. Riskov Barbara & Zilles Stephen. (1974). *Programming with Abstract Data Types (ADT)*. ACM SIGPLAN Notices. 9(4). 50-59.
2. Santoso T, J. (2021). *Struktur dan Algoritma*. Universitas STEKOM. Semarang.
3. Jouannaud J. & Okada M., *Abstract Data Systems*. Theoretical Computer Science.