

**LAPORAN PRAKTIKUM**  
**Modul 3**  
**“ABSTRACT DATA TYPE (ADT)”**



**Disusun Oleh:**  
**Ganes Gemi Putra - 2311104075**  
**SE – 07 - 02**

**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.

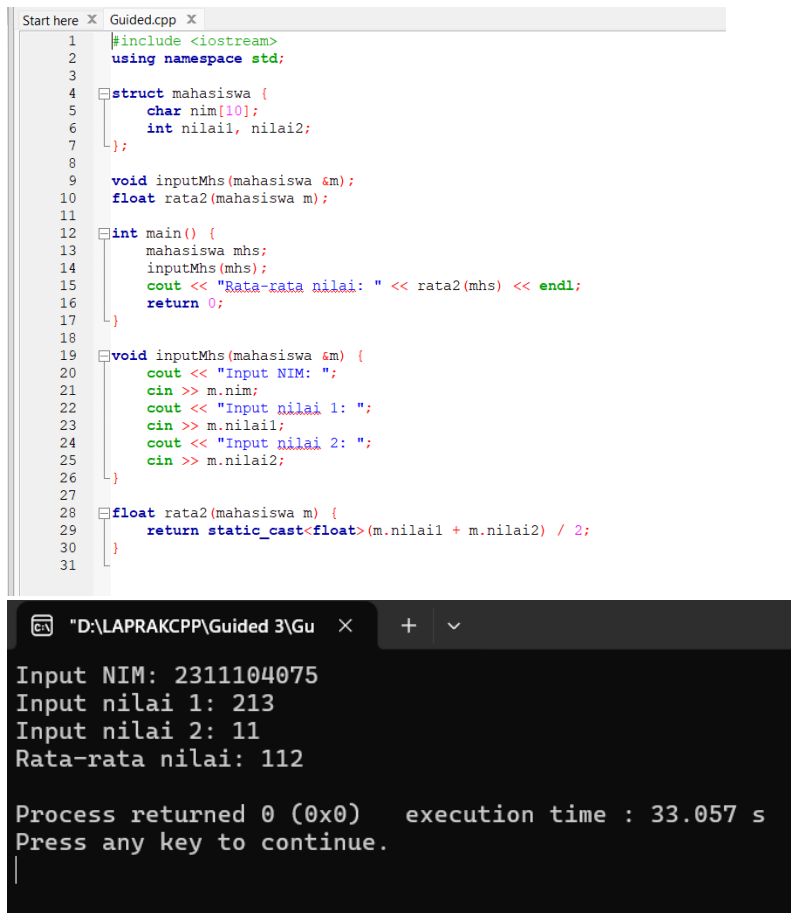
**2. Landasan Teori**

**Abstract Data Type (ADT) adalah tipe data yang didefinisikan oleh perilaku yang bisa dilakukan oleh operasi-operasi yang ada pada tipe tersebut. Konsep ini melibatkan operasi dasar yang disebut primitif, seperti konstruktor, selector, dan validator. ADT digunakan untuk memisahkan definisi tipe data dari**

implementasinya. Implementasi ADT biasanya terbagi menjadi dua modul utama: definisi/spesifikasi tipe dan primitif, serta realisasi dari primitif dalam kode program.

Implementasi ADT membantu dalam pembuatan struktur data yang lebih terstruktur, seperti yang dijelaskan dalam contoh kode program yang menggunakan bahasa C++.

### 3. Guided



The image shows a C++ program in a code editor and its execution output in a terminal window. The code defines a `mahasiswa` struct with `nim` and `nilai` fields, and functions to input data and calculate the average. The terminal shows the program running with input values 2311104075, 213, and 11, resulting in an average of 112.

```
1 #include <iostream>
2 using namespace std;
3
4 struct mahasiswa {
5     char nim[10];
6     int nilai1, nilai2;
7 };
8
9 void inputMhs(mahasiswa &m);
10 float rata2(mahasiswa m);
11
12 int main() {
13     mahasiswa mhs;
14     inputMhs(mhs);
15     cout << "Rata-rata nilai: " << rata2(mhs) << endl;
16     return 0;
17 }
18
19 void inputMhs(mahasiswa &m) {
20     cout << "Input NIM: ";
21     cin >> m.nim;
22     cout << "Input nilai 1: ";
23     cin >> m.nilai1;
24     cout << "Input nilai 2: ";
25     cin >> m.nilai2;
26 }
27
28 float rata2(mahasiswa m) {
29     return static_cast<float>(m.nilai1 + m.nilai2) / 2;
30 }
31
```

```
"D:\LAPRAKCPP\Guided 3\Gu  ×  +  v
Input NIM: 2311104075
Input nilai 1: 213
Input nilai 2: 11
Rata-rata nilai: 112

Process returned 0 (0x0)   execution time : 33.057 s
Press any key to continue.
|
```

#### 4. Unguided

##### Program untuk menyimpan data mahasiswa

```
Guided.cpp X main.cpp X
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 struct Mahasiswa {
7     string nama;
8     string nim;
9     float uts;
10    float uas;
11    float tugas;
12    float nilai_akhir;
13 };
14
15 // Fungsi untuk menghitung nilai akhir
16 float hitungNilaiAkhir(float uts, float uas, float tugas) {
17     return (0.3 * uts) + (0.4 * uas) + (0.3 * tugas);
18 }
19
20 // Fungsi untuk input data mahasiswa
21 void inputMahasiswa(Mahasiswa &mhs) {
22     cout << "Masukkan Nama: ";
23     getline(cin, mhs.nama);
24     cout << "Masukkan NIM: ";
25     getline(cin, mhs.nim);
26     cout << "Masukkan Nilai UTS: ";
27     cin >> mhs.uts;
28     cout << "Masukkan Nilai UAS: ";
29     cin >> mhs.uas;
30     cout << "Masukkan Nilai Tugas: ";
31     cin >> mhs.tugas;
32     cin.ignore(); // Menghilangkan newline karakter dari buffer
33     mhs.nilai_akhir = hitungNilaiAkhir(mhs.uts, mhs.uas, mhs.tugas);
34 }
35
36 // Fungsi untuk menampilkan data mahasiswa
37 void tampilkanMahasiswa(const Mahasiswa &mhs) {
38     cout << "Nama: " << mhs.nama << endl;
39 }
```

Hasil :

```
D:\LAPRAKCPP\Unguided\bin X + v
Masukkan jumlah mahasiswa (max 10): 10

Data Mahasiswa ke-1:
Masukkan Nama: Ganes
Masukkan NIM: 2311104075
Masukkan Nilai UTS: 100
Masukkan Nilai UAS: 100
Masukkan Nilai Tugas: 100

Data Mahasiswa ke-2:
Masukkan Nama: |
```

##### Implementasi ADT pelajaran

##### Program dengan array 2D dan pointer

```

92
93 #include <iostream>
94
95 using namespace std;
96
97 void tampilkanArray(int arr[3][3]) {
98     for (int i = 0; i < 3; ++i) {
99         for (int j = 0; j < 3; ++j) {
100             cout << arr[i][j] << " ";
101         }
102         cout << endl;
103     }
104 }
105
106 void tukarArray(int arr1[3][3], int arr2[3][3], int baris, int kolom) {
107     int temp = arr1[baris][kolom];
108     arr1[baris][kolom] = arr2[baris][kolom];
109     arr2[baris][kolom] = temp;
110 }
111
112 void tukarPointer(int *a, int *b) {
113     int temp = *a;
114     *a = *b;
115     *b = temp;
116 }
117
118 int main() {
119     int array1[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
120     int array2[3][3] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
121     int a = 10, b = 20;
122
123     cout << "Array 1 sebelum penukaran:" << endl;
124     tampilkanArray(array1);
125
126     cout << "\nArray 2 sebelum penukaran:" << endl;
127     tampilkanArray(array2);
128
129     tukarArray(array1, array2, 0, 0);
130     tukarPointer(&a, &b);
131
132     cout << "Array 1 setelah penukaran:" << endl;
133     tampilkanArray(array1);
134
135     cout << "\nArray 2 setelah penukaran:" << endl;
136     tampilkanArray(array2);
137
138     cout << "Nilai a dan b sebelum penukaran: a = 10, b = 20\n";
139     cout << "Nilai a dan b setelah penukaran: a = 20, b = 10\n";
140
141     return 0;
142 }

```

## Hasil:

```

D:\LAPRAKCPP\Unguided\Un...
+ v

Array 1 sebelum penukaran:
1 2 3
4 5 6
7 8 9

Array 2 sebelum penukaran:
9 8 7
6 5 4
3 2 1

Array 1 setelah penukaran:
1 2 3
4 5 6
7 8 9

Array 2 setelah penukaran:
9 8 7
6 5 4
3 2 1

Nilai a dan b sebelum penukaran: a = 10, b = 20
Nilai a dan b setelah penukaran: a = 20, b = 10

Process returned 0 (0x0)   execution time : 0.050 s
Press any key to continue.

```

## 5. Kesimpulan

konsep ADT sangat penting dalam pengembangan program yang modular dan terstruktur. ADT memungkinkan pemrogram untuk memisahkan antara apa yang dilakukan oleh suatu tipe data dan bagaimana tipe data tersebut diimplementasikan, yang pada akhirnya

meningkatkan keterbacaan kode, pemeliharaan, dan fleksibilitas pengembangan program.