

LAPORAN PRAKTIKUM
Modul 3
“Abstract Data Type (ADT)”



Disusun Oleh:

Ahmad Al - Farizi - 2311104054

Kelas :

S1SE-07-02

Dosen :

Wahyu Andi Saputra, S.Pd, M.Eng

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

1. Tujuan

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

2. Landasan Teori

2.1. Abstract Data Type

Abstract Data Type (ADT) adalah model matematika yang mendefinisikan tipe data dengan mengaitkannya pada operasi-operasi yang dapat dilakukan terhadap data tersebut. Konsep ini berfungsi untuk memisahkan struktur penyimpanan dari perilaku tipe data, sehingga pemrogram tidak perlu mengetahui rincian implementasi internal dari tipe data tersebut. ADT memiliki beberapa tujuan dan manfaat, seperti modularitas, yang memungkinkan pengembangan sistem secara terpisah tanpa saling mengganggu, serta pengkapsulan, yang menyembunyikan informasi internal dari pengguna. Dengan demikian, perubahan pada implementasi tidak akan mempengaruhi program yang menggunakan ADT tersebut. Selain itu, ADT juga memberikan abstraksi, sehingga pemrogram dapat fokus pada perilaku objek tanpa harus memperhatikan detail implementasi.

Dalam struktur ADT terdapat dua komponen utama: definisi tipe, yang menyediakan spesifikasi tentang bagaimana tipe data diorganisir, dan operasi dasar, yaitu kumpulan fungsi atau prosedur yang dapat digunakan untuk memanipulasi data dalam tipe tersebut. ADT dapat dibedakan menjadi beberapa jenis berdasarkan sifat datanya, seperti ADT homogen yang hanya menampung variabel dengan tipe data yang sama, dan ADT heterogen yang dapat menampung variabel dengan berbagai tipe data. Sebagai contoh implementasi, dalam bahasa C terdapat struktur `Jam` yang mendefinisikan waktu dan dilengkapi dengan fungsi `MakeJam` serta `TulisJam` untuk membuat dan menampilkan waktu. Secara keseluruhan, ADT merupakan konsep fundamental dalam pemrograman yang membantu dalam pengembangan perangkat lunak yang lebih terstruktur dan mudah dikelola, memungkinkan pemrogram menciptakan solusi yang lebih efisien dan modular.

3. Guided

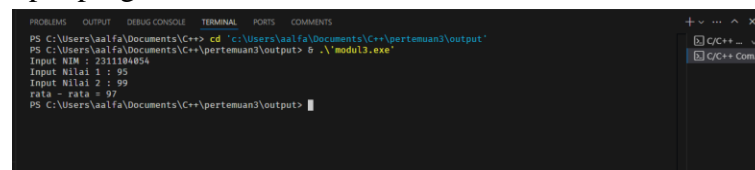
3.1. ADT

Kode program ini digunakan untuk menghitung rata – rata nilai dua mata pelajaran dari seorang mahasiswa. Program ini terdiri dari struct mahasiswa yang memiliki atribut NIM, nilai 1, dan nilai 2. Fungsi inputMahasiswa adalah untuk menginputkan data mahasiswa, sedangkan fungsi rata2 menghitung rata – rata nilai. Pada fungsi main, program memanggil kedua fungsi tersebut dan menampilkan hasil rata – rata untuk nilai mahasiswa.

Kode program :

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

Output program :



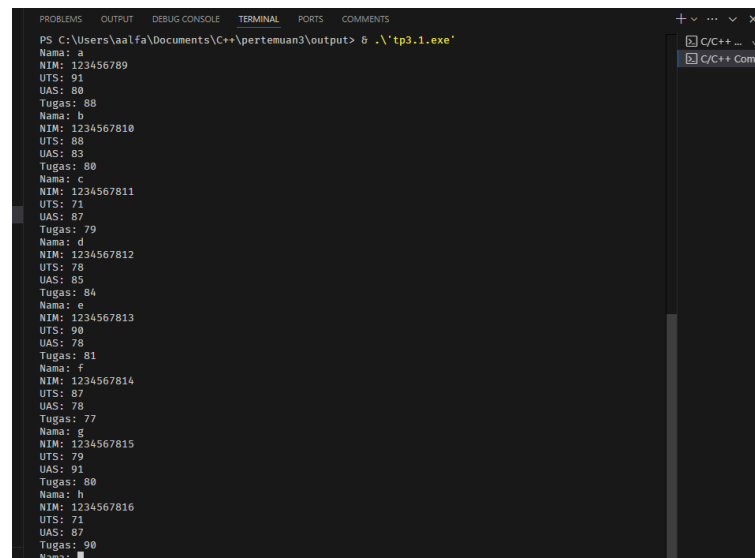
```
PS C:\Users\aa1fa\Documents\C++> cd 'C:\Users\aa1fa\Documents\C++\pertemuan3\output'
PS C:\Users\aa1fa\Documents\C++\pertemuan3\output> .\modul3.exe
Input NIM : 2311104034
Input Nilai 1 : 95
Input Nilai 2 : 99
rata - rata = 97
PS C:\Users\aa1fa\Documents\C++\pertemuan3\output>
```

4. Unguided

4.1. Kode program :

```
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 const int MAX_MAHASISWA = 10;
7
8 struct Mahasiswa {
9     string nama;
10    int nim;
11    float uts;
12    float uas;
13    float tugas;
14    float nilaiAkhir;
15 };
16
17 float hitungNilaiAkhir(float uts, float uas, float tugas) {
18     return 0.3 * uts + 0.4 * uas + 0.3 * tugas;
19 }
20
21 int main() {
22     Mahasiswa mahasiswa[MAX_MAHASISWA];
23     int jumlahMahasiswa = 0;
24
25     for (int i = 0; i < MAX_MAHASISWA; i++) {
26         cout << "Nama: ";
27         cin.ignore();
28         getline(cin, mahasiswa[i].nama);
29         cout << "NIM: ";
30         cin >> mahasiswa[i].nim;
31         cout << "UTS: ";
32         cin >> mahasiswa[i].uts;
33         cout << "UAS: ";
34         cin >> mahasiswa[i].uas;
35         cout << "Tugas: ";
36         cin >> mahasiswa[i].tugas;
37         mahasiswa[i].nilaiAkhir = hitungNilaiAkhir(mahasiswa[i].uts, mahasiswa[i].uas, mahasiswa[i].tugas);
38         jumlahMahasiswa++;
39     }
40
41     cout << "Data Mahasiswa:" << endl;
42     for (int i = 0; i < jumlahMahasiswa; i++) {
43         cout << "Nama: " << mahasiswa[i].nama << endl;
44         cout << "NIM: " << mahasiswa[i].nim << endl;
45         cout << "UTS: " << mahasiswa[i].uts << endl;
46         cout << "UAS: " << mahasiswa[i].uas << endl;
47         cout << "Tugas: " << mahasiswa[i].tugas << endl;
48         cout << "Nilai Akhir: " << mahasiswa[i].nilaiAkhir << endl;
49     }
50
51     return 0;
52 }
```

Output dari kode program :



```
PS C:\Users\aalfa\Documents\C++\pertemuan3\output> g++ tp3.1.exe
Nama: a
NIM: 123456789
UTS: 91
UAS: 80
Tugas: 88
Nama: b
NIM: 1234567810
UTS: 88
UAS: 83
Tugas: 80
Nama: c
NIM: 1234567811
UTS: 71
UAS: 87
Tugas: 79
Nama: d
NIM: 1234567812
UTS: 78
UAS: 85
Tugas: 84
Nama: e
NIM: 1234567813
UTS: 90
UAS: 78
Tugas: 81
Nama: f
NIM: 1234567814
UTS: 87
UAS: 78
Tugas: 77
Nama: g
NIM: 1234567815
UTS: 79
UAS: 91
Tugas: 80
Nama: h
NIM: 1234567816
UTS: 71
UAS: 87
Tugas: 90
Nama: 
```

4.2. Kode program :

paragraf.h :

```

paragraf.h  X  main.cpp  X  pelajaran.cpp  X
1  #ifndef PELAJARAN_H_INCLUDED
2  #define PELAJARAN_H_INCLUDED
3
4  #include <string>
5  using namespace std;
6
7  struct Pelajaran {
8      string namaMapel;
9      string kodeMapel;
10 };
11
12 Pelajaran create_pelajaran(string namaapel, string kodepel);
13 void tampil_pelajaran(Pelajaran pel);
14
15 #endif // PELAJARAN_H_INCLUDED
16
17

```

paragraf.cpp

```

paragraf.h  X  main.cpp  X  pelajaran.cpp  X
1  #include "pelajaran.h"
2  #include <iostream>
3
4  using namespace std;
5
6  Pelajaran create_pelajaran(string namaapel, string kodepel) {
7      Pelajaran pel;
8      pel.namaMapel = namaapel;
9      pel.kodeMapel = kodepel;
10     return pel;
11 }
12
13 void tampil_pelajaran(Pelajaran pel) {
14     cout << "Nama Pelajaran : " << pel.namaMapel << endl;
15     cout << "Kode Pelajaran : " << pel.kodeMapel << endl;
16 }
17

```

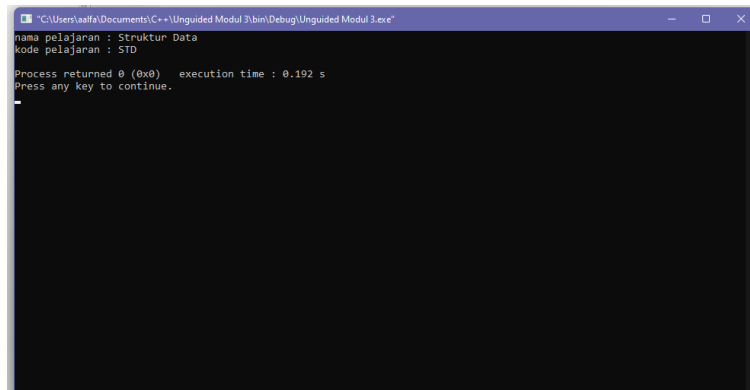
main.cpp

```

paragraf.h  X  main.cpp  X  pelajaran.cpp  X
1  #include "paragraf.h"
2  #include <iostream>
3
4  using namespace std;
5
6  int main()
7  {
8      Pelajaran pel = create_pelajaran("Struktur Data", "STD");
9      tampil_pelajaran(pel);
10     return 0;
11 }
12

```

Output dari kode program main.cpp :

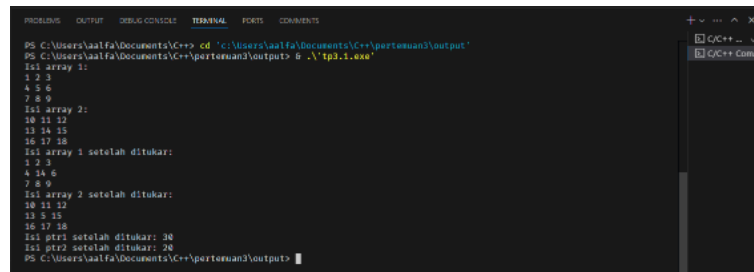


```
C:\Users\ialfa\Documents\C++\Unguided Modul 3\bin\Debug\Unguided Modul 3.exe
nama pelajaran : Struktur Data
kode pelajaran : STD
Process returned 0 (0x0) execution time : 0.192 s
Press any key to continue.
```

4.3. Kode program :

```
1  #include <iostream>
2
3  void tampilkanArray(int arr[3][3]) {
4      for (int i = 0; i < 3; i++) {
5          for (int j = 0; j < 3; j++) {
6              std::cout << arr[i][j] << " ";
7          }
8          std::cout << std::endl;
9      }
10 }
11
12 void tukarArray(int arr1[3][3], int arr2[3][3], int posisi1, int posisi2) {
13     int temp = arr1[posisi1][posisi2];
14     arr1[posisi1][posisi2] = arr2[posisi1][posisi2];
15     arr2[posisi1][posisi2] = temp;
16 }
17
18 void tukarPointer(int* ptr1, int* ptr2) {
19     int temp = *ptr1;
20     *ptr1 = *ptr2;
21     *ptr2 = temp;
22 }
23
24 int main() {
25
26     int arr1[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
27     int arr2[3][3] = {{10, 11, 12}, {13, 14, 15}, {16, 17, 18}};
28
29     int* ptr1 = new int;
30     int* ptr2 = new int;
31     *ptr1 = 20;
32     *ptr2 = 30;
33
34     std::cout << "Isi array 1:" << std::endl;
35     tampilkanArray(arr1);
36     std::cout << "Isi array 2:" << std::endl;
37     tampilkanArray(arr2);
38
39     tukarArray(arr1, arr2, 1, 1);
40     std::cout << "Isi array 1 setelah ditukar:" << std::endl;
41     tampilkanArray(arr1);
42     std::cout << "Isi array 2 setelah ditukar:" << std::endl;
43     tampilkanArray(arr2);
44
45     tukarPointer(ptr1, ptr2);
46     std::cout << "Isi ptr1 setelah ditukar: " << *ptr1 << std::endl;
47     std::cout << "Isi ptr2 setelah ditukar: " << *ptr2 << std::endl;
48
49     delete ptr1;
50     delete ptr2;
51
52     return 0;
53 }
```

Output dari kode program :



```
PS C:\Users\aa1fa\Documents\C++> cd 'C:\Users\aa1fa\Documents\C++\pertemuan3\output'
PS C:\Users\aa1fa\Documents\C++\pertemuan3\output> g++ tp3.1.exe
Isi array 1:
1 2 3
4 5 6
7 8 9
Isi array 2:
10 11 12
13 14 15
16 17 18
Isi array 1 setelah ditukar:
1 2 3
4 14 6
7 8 9
Isi array 2 setelah ditukar:
10 11 12
13 5 15
16 17 18
Isi ptr1 setelah ditukar: 30
Isi ptr2 setelah ditukar: 20
PS C:\Users\aa1fa\Documents\C++\pertemuan3\output>
```

5. Kesimpulan

ADT adalah model matematika yang mendefinisikan tipe data dengan operasi-operasi yang dapat dilakukan terhadap data tersebut. ADT memisahkan struktur penyimpanan dari perilaku tipe data, memungkinkan modularitas, pengkapsulan, dan abstraksi. ADT terdiri dari definisi tipe dan operasi dasar, dan dapat dibedakan menjadi beberapa jenis berdasarkan sifat datanya.

