

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

**MODUL III
ABSTRACT DATA TYPE (ADT)**



Disusun Oleh :

NAMA : Azzahra Fareluka Esti Ning Tyas
NIM : 103112430023

Dosen

FAHRUDIN MUKTI WIBOWO

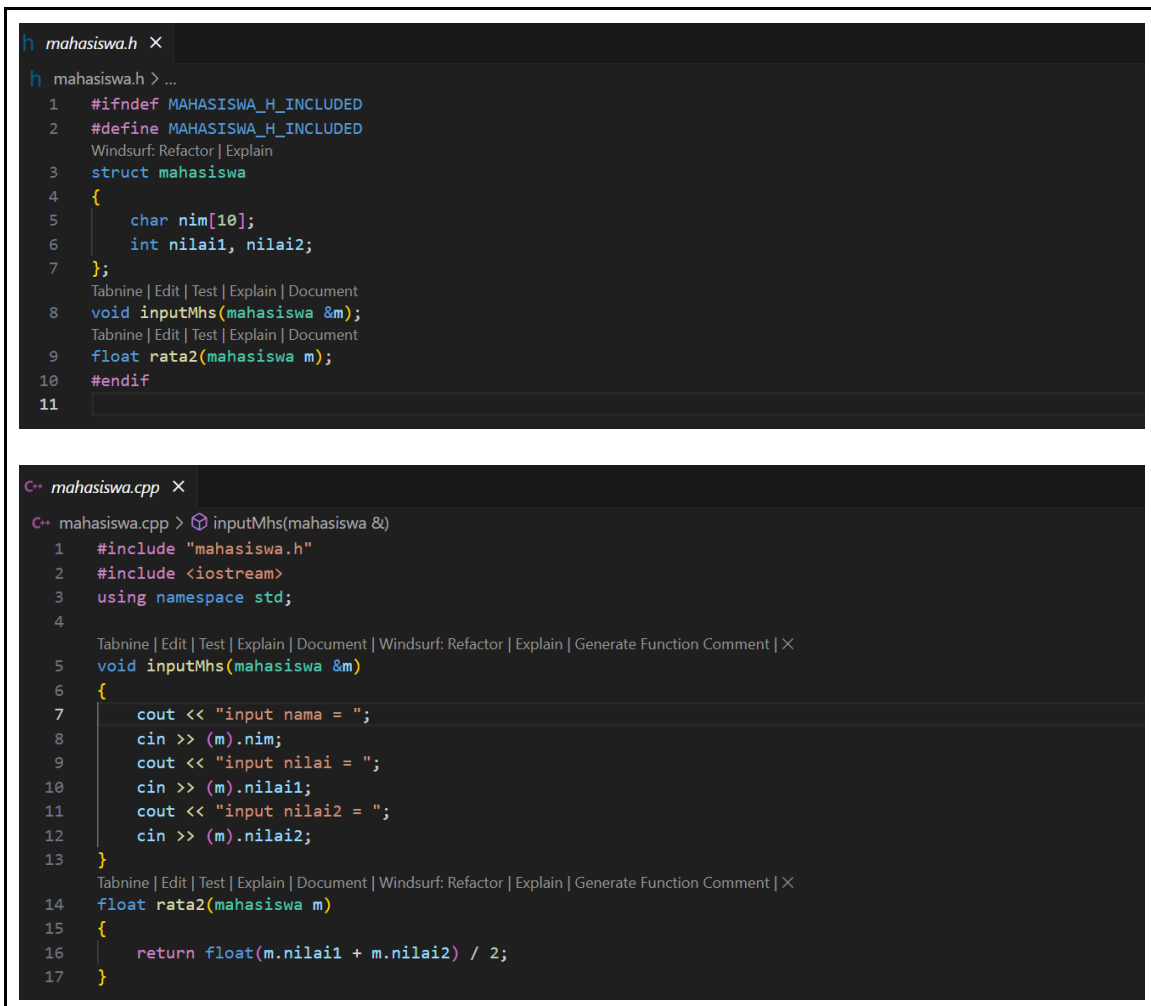
**PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

A. Dasar Teori

Abstract Data Type (ADT) atau Tipe Data Abstrak merupakan konsep dalam pemrograman yang mendefinisikan suatu tipe data berdasarkan nilai-nilai yang dapat disimpan dan operasi yang dapat dilakukan terhadap nilai tersebut, tanpa memperhatikan bagaimana cara implementasinya. Dalam bahasa C++, ADT biasanya diwujudkan melalui kelas (class) yang menggabungkan data dan fungsi dalam satu kesatuan, dengan menerapkan prinsip enkapsulasi untuk menyembunyikan detail implementasi dari pengguna. Dengan adanya ADT, pengembang hanya perlu mengetahui cara menggunakan operasi yang disediakan tanpa harus memahami strukturnya. Contohnya, pada ADT Stack, pengguna cukup mengetahui operasi seperti push (menambahkan data), pop (menghapus data), dan peek (melihat data teratas), tanpa perlu tahu bagaimana data disimpan di memori. Konsep ini membantu menciptakan program yang lebih modular, mudah dipelihara, dan terstruktur dengan baik, karena perubahan pada implementasi internal tidak memengaruhi bagian program lain yang menggunakan ADT tersebut.

B. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1



```
h mahasiswa.h X
h mahasiswa.h > ...
1  #ifndef MAHASISWA_H_INCLUDED
2  #define MAHASISWA_H_INCLUDED
3  struct mahasiswa
4  {
5      char nim[10];
6      int nilai1, nilai2;
7  };
8  void inputMhs(mahasiswa &m);
9  float rata2(mahasiswa m);
10 #endif
11

C++ mahasiswa.cpp X
C++ mahasiswa.cpp > inputMhs(mahasiswa &m)
1  #include "mahasiswa.h"
2  #include <iostream>
3  using namespace std;
4
5  void inputMhs(mahasiswa &m)
6  {
7      cout << "input nama = ";
8      cin >> (m).nim;
9      cout << "input nilai = ";
10     cin >> (m).nilai1;
11     cout << "input nilai2 = ";
12     cin >> (m).nilai2;
13 }
14 float rata2(mahasiswa m)
15 {
16     return float(m.nilai1 + m.nilai2) / 2;
17 }
```

```
main.cpp x
C++ main.cpp > main()
1  #include <iostream>
2  #include "mahasiswa.h"
3  #include "mahasiswa.cpp"
4  using namespace std;
5
6  Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
7  int main()
8  {
9      mahasiswa mhs;
10     inputMhs(mhs);
11     cout << "rata - rata = " << rata2(mhs);
12     return 0;
}
```

Screenshots Output

```
PS D:\StrukturData> cd "d:\StrukturData\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
input nama = Azzahra
input nilai = 100
input nilai2 = 100
rata - rata = 100
```

Deskripsi:

Program C++ ini menerapkan konsep Abstract Data Type (ADT) dengan memisahkan data dan fungsi ke dalam beberapa file. Struktur mahasiswa menyimpan NIM serta dua nilai, lalu fungsi `inputMhs()` digunakan untuk memasukkan data dan `rata2()` untuk menghitung rata-ratanya. Program utama di `main.cpp` memanggil kedua fungsi tersebut untuk menampilkan hasil rata-rata. Dengan ADT, program menjadi lebih teratur dan mudah dipahami.

C. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Unguided 1

```
Soal1.cpp X
Modul03 > Unguided > C++ Soal1.cpp > 58 Mahasiswa > uts

1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 struct Mahasiswa {
6     string nama;
7     string nim;
8     float uts, uas, tugas, nilaiAkhir;
9 };
10
11 float hitungNilaiAkhir(float uts, float uas, float tugas) {
12     return (0.3 * uts) + (0.4 * uas) + (0.3 * tugas);
13 }
14
15 void input(Mahasiswa mhs[], int jumlah) {
16     cin.ignore();
17     for (int i = 0; i < jumlah; i++) {
18         cout << "\nData Mahasiswa ke-" << i + 1 << endl;
19         cout << "Nama      : ";
20         getline(cin, mhs[i].nama);
21         cout << "NIM       : ";
22         getline(cin, mhs[i].nim);
23         cout << "Nilai UTS   : ";
24         cin >> mhs[i].uts;
25         cout << "Nilai UAS   : ";
26         cin >> mhs[i].uas;
27         cout << "Nilai Tugas : ";
28         cin >> mhs[i].tugas;
29
30         mhs[i].nilaiAkhir = hitungNilaiAkhir(mhs[i].uts, mhs[i].uas, mhs[i].tugas);
31         cin.ignore();
32     }
33 }
34
35 void tampilkanData(Mahasiswa mhs[], int jumlah) {
36     for (int i = 0; i < jumlah; i++) {
37         cout << "\nData Mahasiswa ke-" << i + 1 << endl;
38         cout << "Nama      : " << mhs[i].nama << endl;
39         cout << "NIM       : " << mhs[i].nim << endl;
40         cout << "Nilai UTS   : " << mhs[i].uts << endl;
41         cout << "Nilai UAS   : " << mhs[i].uas << endl;
42         cout << "Nilai Tugas : " << mhs[i].tugas << endl;
43         cout << "Nilai Akhir : " << mhs[i].nilaiAkhir << endl;
44     }
45 }
46
47 int main() {
48     int jumlah;
49     Mahasiswa mhs[10];
50
51     cout << "Masukkan jumlah mahasiswa (1-10): ";
52     cin >> jumlah;
53
54     if (jumlah >= 1 && jumlah <= 10) {
55         input(mhs, jumlah);
56         tampilkanData(mhs, jumlah);
57     } else {
58         cout << "Jumlah mahasiswa harus antara 1 sampai 10!" << endl;
59     }
60
61     return 0;
62 }
```

Screenshots Output

```
● Masukkan jumlah mahasiswa (1-10): 2

Data Mahasiswa ke-1
Nama      : Azzahra
NIM       : 12345
Nilai UTS : 100
Nilai UAS : 99
Nilai Tugas : 98

Data Mahasiswa ke-2
Nama      : Farelika
NIM       : 54321
Nilai UTS : 99
Nilai UAS : 99
Nilai Tugas : 99

Data Mahasiswa ke-1
Nama      : Azzahra
NIM       : 12345
Nilai UTS : 100
Nilai UAS : 99
Nilai Tugas : 98
Nilai Akhir : 99

Data Mahasiswa ke-2
Nama      : Farelika
NIM       : 54321
Nilai UTS : 99
Nilai UAS : 99
Nilai Tugas : 99
Nilai Akhir : 99
```

Deskripsi:

Program ini berfungsi untuk menginput dan menampilkan data mahasiswa menggunakan struct dan fungsi terpisah. Struct Mahasiswa menyimpan data seperti nama, NIM, nilai UTS, UAS, tugas, dan nilai akhir. Fungsi `hitungNilaiAkhir()` menghitung nilai akhir berdasarkan rumus $0.3 \times \text{uts} + 0.4 \times \text{uas} + 0.3 \times \text{tugas}$. Fungsi `input()` meminta data dari pengguna, lalu fungsi `tampilkanData()` menampilkan hasilnya. Di dalam `main()`, pengguna diminta memasukkan jumlah mahasiswa (maksimal 10). Jika valid, data diolah dan ditampilkan; jika tidak, muncul pesan kesalahan. Program ini menerapkan konsep array of struct dan fungsi modular.

Unguided 2

```
h Soal2_pelajaran.h X
Modul03 > Unguided > h Soal2_pelajaran.h > create_pelajaran(string, string)
1  #ifndef SOAL2_PELAJARAN_H
2  #define SOAL2_PELAJARAN_H
3  #include <string>
4  using namespace std;
5
6  Windsurf: Refactor | Explain
7  struct pelajaran {
8      string namaMapel;
9      string kodeMapel;
10 };
11
12 Tabnine | Edit | Test | Explain | Document
13 pelajaran create_pelajaran(string namapel, string kodepel);
14 void tampil_pelajaran(pelajaran pel);
15 #endif

C++ Soal2_pelajaran.cpp X
Modul03 > Unguided > C++ Soal2_pelajaran.cpp > ...
1  #include <iostream>
2  #include "Soal2_pelajaran.h"
3  using namespace std;
4
5  Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
6  pelajaran create_pelajaran(string namapel, string kodepel) {
7      pelajaran p;
8      p.namaMapel = namapel;
9      p.kodeMapel = kodepel;
10     return p;
11 }
12
13 Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
14 void tampil_pelajaran(pelajaran pel) {
15     cout << "nama pelajaran : " << pel.namaMapel << endl;
16     cout << "nilai : " << pel.kodeMapel << endl;
17 }

C++ Soal2_main.cpp X
Modul03 > Unguided > C++ Soal2_main.cpp > ...
1  #include <iostream>
2  #include "Soal2_pelajaran.h"
3  #include "Soal2_pelajaran.cpp"
4  using namespace std;
5
6  Tabnine | Edit | Test | Explain | Document | Windsurf: Refactor | Explain | Generate Function Comment | X
7  int main() {
8      string namapel = "Struktur Data";
9      string kodepel = "STD";
10
11     pelajaran pel = create_pelajaran(namapel, kodepel);
12     tampil_pelajaran(pel);
13
14     return 0;
15 }
```

Screenshots Output

```
PS D:\StrukturData> cd "d:\StrukturData\Modul03\Unguided\" ; if ($?) { g++ Soal2_main.cpp -o Soal2_main } ; if ($?) { .\Soal2_main }
• nama pelajaran : Struktur Data
  nilai : STD
```

Deskripsi:

Program ini menerapkan konsep Abstract Data Type (ADT) dengan memisahkan kode menjadi tiga file: pelajaran.h, pelajaran.cpp, dan main.cpp. File pelajaran.h berisi definisi struktur pelajaran yang memiliki dua atribut, yaitu namaMapel dan kodeMapel, serta deklarasi dua fungsi: create_pelajaran() dan tampil_pelajaran(). File pelajaran.cpp berisi implementasi fungsi-fungsi tersebut: create_pelajaran() digunakan untuk membuat objek pelajaran dengan mengisi atributnya, sedangkan tampil_pelajaran() menampilkan data pelajaran ke layar. File main.cpp berfungsi sebagai program utama yang membuat objek pelajaran dengan nama “Struktur Data” dan kode “STD”, lalu menampilkan hasilnya. Secara keseluruhan, program ini menunjukkan cara memisahkan definisi, implementasi, dan pemanggilan ADT agar kode lebih terstruktur dan mudah dipelihara.

Unguided 3

```
C++ Soal3.cpp X
Modul03 > Unguided > C++ Soal3.cpp > ...
1 #include <iostream>
2 using namespace std;
3
4 void tampilArray(int arr[3][3]) {
5     for (int i = 0; i < 3; i++) {
6         for (int j = 0; j < 3; j++) {
7             cout << arr[i][j] << " ";
8         }
9         cout << endl;
10    }
11 }
12
13 void tukarArray(int arr1[3][3], int arr2[3][3], int baris, int kolom) {
14     int temp = arr1[baris][kolom];
15     arr1[baris][kolom] = arr2[baris][kolom];
16     arr2[baris][kolom] = temp;
17 }
18
19 void tukarPointer(int *p1, int *p2) {
20     int temp = *p1;
21     *p1 = *p2;
22     *p2 = temp;
23 }
24
25 int main() {
26     int A[3][3] = {
27         {1, 2, 3},
28         {4, 5, 6},
29         {7, 8, 9}
30     };
31
32     int B[3][3] = {
33         {10, 11, 12},
34         {13, 14, 15},
35         {16, 17, 18}
36     };
37
38     int *p1, *p2;
39     int x = 100, y = 200;
40
41     cout << "Array A:" << endl;
42     tampilArray(A);
43     cout << "\nArray B:" << endl;
44     tampilArray(B);
45
46     cout << "\nMenukar elemen A[1][1] dengan B[1][1]...\n";
47     tukarArray(A, B, 1, 1);
48
49     cout << "\nArray A setelah ditukar:" << endl;
50     tampilArray(A);
51     cout << "\nArray B setelah ditukar:" << endl;
52     tampilArray(B);
53
54     p1 = &x;
55     p2 = &y;
56
57     cout << "\nSebelum tukar pointer:" << endl;
58     cout << "x = " << x << ", y = " << y << endl;
59
60     tukarPointer(p1, p2);
61
62     cout << "Setelah tukar pointer:" << endl;
63     cout << "x = " << x << ", y = " << y << endl;
64
65     return 0;
66 }
```

Screenshots Output

```
PS D:\StrukturData> cd "d:\StrukturData\Modul03\Unguided\" ; if ($?) { g++ Soal3.cpp -o Soal3 } ; if ($?) { .\Soal3 }
Array A:
1      2      3
4      5      6
7      8      9

Array B:
10     11     12
13     14     15
16     17     18

Menukar elemen A[1][1] dengan B[1][1]...

Array A setelah ditukar:
1      2      3
4      14     6
7      8      9

Array B setelah ditukar:
10     11     12
13     5      15
16     17     18

Sebelum tukar pointer:
x = 100, y = 200
Setelah tukar pointer:
x = 200, y = 100
```

Deskripsi:

Program ini menampilkan dan memanipulasi data menggunakan array 2 dimensi dan pointer. Fungsi `tampilArray()` digunakan untuk menampilkan isi array 3×3 ke layar. Fungsi `tukarArray()` menukar elemen pada posisi tertentu antara dua array 2D yang berbeda. Fungsi `tukarPointer()` menukar nilai dari dua variabel melalui pointer. Dalam fungsi `main()`, terdapat dua array (A dan B) berukuran 3×3 serta dua variabel integer (x dan y) yang dihubungkan dengan pointer. Program pertama menampilkan isi array, lalu menukar elemen `A[1][1]` dengan `B[1][1]`, menampilkan hasil pertukaran, kemudian menukar nilai x dan y menggunakan pointer. Program ini menunjukkan penggunaan fungsi, array 2D, dan konsep pointer secara sederhana.

D. Kesimpulan

Melalui praktikum Modul III tentang Abstract Data Type (ADT), dapat disimpulkan bahwa ADT merupakan konsep penting dalam pemrograman yang memungkinkan pemisahan antara definisi tipe data dan implementasinya, sehingga program menjadi lebih modular, terstruktur, dan mudah dipelihara. Dengan menerapkan ADT menggunakan struct dan fungsi terpisah, seperti pada contoh data mahasiswa dan pelajaran, kode menjadi lebih rapi dan fleksibel terhadap perubahan. Selain itu, penggunaan array, pointer, dan fungsi modular membantu memperkuat pemahaman tentang cara pengelolaan data di dalam memori serta penerapan prinsip encapsulation dalam C++. Secara keseluruhan, penerapan ADT mendukung pengembangan program yang efisien, mudah dibaca, dan dapat dikembangkan lebih lanjut.

E. Referensi

Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2014). Data Structures and Algorithms in C++. Wiley.

Wirth, N. (1976). Algorithms + Data Structures = Programs. Prentice Hall.

Deitel, P. J., & Deitel, H. M. (2017). C++ How to Program (10th Edition). Pearson.

Carrano, F. M., & Henry, T. (2015). Data Abstraction and Problem Solving with C++: Walls and Mirrors. Pearson.