

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

**MODUL III
ABSTRACT DATA TYPE**



Disusun oleh :

Junadil Muqorobin (103112400281)

Dosen

Fahrudin Mukti Wibowo S.Kom., M.Eng

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

A. Dasar Teori

Abstract Data Type (ADT) atau Tipe Data Abstrak merupakan salah satu konsep fundamental dalam bidang struktur data yang berfungsi untuk memisahkan antara *definisi logis* suatu tipe data dengan *implementasi fisiknya*. ADT mendefinisikan kumpulan nilai beserta operasi yang dapat dilakukan terhadap nilai-nilai tersebut tanpa memperhatikan bagaimana data disimpan atau diimplementasikan dalam memori komputer. Dengan adanya pemisahan ini, programmer dapat fokus pada cara penggunaan data tanpa perlu memikirkan detail teknis penyimpanannya. Dalam bahasa pemrograman C++, ADT umumnya diimplementasikan menggunakan *struct* atau *class* yang dilengkapi dengan sekumpulan fungsi atau prosedur seperti konstruktor, selektor, mutator, serta operator relasional dan aritmetika sesuai kebutuhan manipulasi datanya.

Penerapan ADT dalam pengembangan perangkat lunak sangat penting karena mendukung prinsip *modular programming*, yaitu pemisahan antara bagian spesifikasi dan bagian implementasi. Biasanya, bagian spesifikasi diletakkan pada file header (*.h*) sedangkan bagian implementasinya berada pada file sumber (*.cpp*). Pendekatan ini membuat program lebih terstruktur, mudah dipelihara, serta memungkinkan pengujian modul secara terpisah. Sebagai contoh, dalam praktikum ini diperlihatkan bagaimana ADT “mahasiswa” didefinisikan untuk menyimpan data seperti *nim*, *nilai1*, dan *nilai2*, yang kemudian dioperasikan melalui fungsi untuk mengisi data dan menghitung rata-rata nilai. Pendekatan tersebut memperlihatkan bahwa ADT tidak hanya mempermudah

pengorganisasian data, tetapi juga menjaga integritas serta konsistensi struktur data di dalam program. ADT juga memungkinkan pembentukan tipe data yang lebih kompleks melalui kombinasi beberapa ADT yang lebih sederhana. Misalnya, sebuah ADT “waktu” dapat terdiri dari ADT “jam” dan “tanggal”, atau ADT “garis” yang dibangun dari dua ADT “titik” (point). Kemampuan ini sangat bermanfaat dalam pengembangan sistem berskala besar karena setiap tipe data dapat dikembangkan, diuji, dan dimodifikasi secara independen. Dalam konteks C++, pendekatan ini menjadi dasar dari konsep *object-oriented programming* (OOP), di mana data dan fungsi yang terkait disatukan dalam satu entitas yang disebut *class*.

Menurut Srihith, Donald, Srinivas, Anjali, dan Varaprasad (2022), konsep ADT merupakan tulang punggung dari struktur data modern karena menyediakan kerangka kerja yang efisien dalam penyimpanan dan manipulasi data. Mereka menegaskan bahwa pemahaman mendalam terhadap ADT dapat meningkatkan efisiensi algoritma dan membantu pengembang dalam mengoptimalkan kinerja sistem komputasi. Pandangan ini sejalan dengan hasil penelitian Nkweteyim (2017), yang menjelaskan bahwa pemilihan struktur data dan algoritma yang tepat berperan penting dalam efisiensi proses komputasi, termasuk dalam pencarian, pengurutan, serta pengelolaan data dalam berbagai aplikasi. Dengan demikian, penerapan ADT dalam pemrograman tidak hanya memberikan manfaat dari sisi desain modular, tetapi juga berkontribusi secara langsung terhadap peningkatan performa dan keandalan perangkat lunak.

B. Guided

1. Implementasi konsep ADT

Source code mahasiswa.h

```
#ifndef MAHASISWA_H_INCLUDED
#define MAHASISWA_H_INCLUDED

struct mahasiswa
{
    char nim[10];
    int nilai1, nilai2;
};

void inputMhs(mahasiswa &m);
float rata2(mahasiswa m);
#endif
```

Source code mahasiswa.cpp

```
#include "mahasiswa.h"
#include <iostream>

using namespace std;

void inputMhs(mahasiswa &m){
    cout << "input nama : ";
    cin >> (m).nim;
    cout << "input nilai : ";
    cin >> (m).nilai1;
    cout << "input nilai2 : ";
    cin >> (m).nilai2;
}

float rata2(mahasiswa m){
    return float(m.nilai1 + m.nilai2) / 2;
}
```

Source code main.cpp

```
#include <iostream>
#include "mahasiswa.h"

using namespace std;

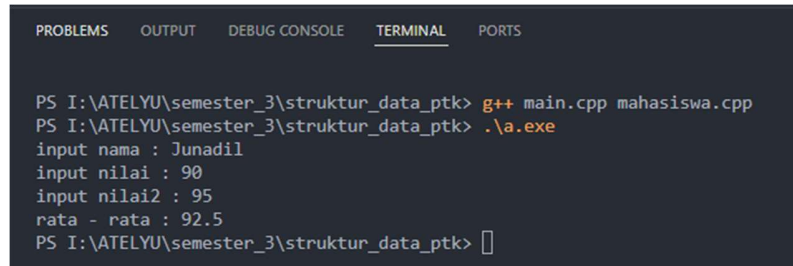
int main(){
```

```

mahasiswa mhs;
inputMhs(mhs);
cout << "rata - rata : " << rata2(mhs);
return 0;
}

```

Screenshoot Output:



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS I:\ATELYU\semester_3\struktur_data_ptk> g++ main.cpp mahasiswa.cpp
PS I:\ATELYU\semester_3\struktur_data_ptk> .\a.exe
input nama : Junadil
input nilai : 90
input nilai2 : 95
rata - rata : 92.5
PS I:\ATELYU\semester_3\struktur_data_ptk> 

```

Deskripsi:

Ketiga file program diatas adalah satu kesatuan penerapan konsep ADT untuk program penghitung rata-rata nilai. dengan langkah kerja sebagai berikut:

- Source code mahasiswa.h
 - File mahasiswa.h merupakan file header yang berfungsi sebagai spesifikasi atau deklarasi Abstract Data Type (ADT). Di dalamnya terdapat definisi *struct* mahasiswa serta deklarasi fungsi-fungsi yang beroperasi terhadap tipe data tersebut.
 - Nim dengan tipe data `char[10]` digunakan untuk menyimpan NIM.
 - `nilai1` dan `nilai2` dengan tipe data `int` digunakan untuk menyimpan dua nilai mahasiswa.
 - `Void inputMhs(mahasiswa &m)` digunakan untuk mengisi data ke dalam variabel bertipe mahasiswa.

- Float rata2(mahasiswa m) digunakan untuk menghitung rata-rata dari dua nilai mahasiswa.
- Source code mahasiswa.cpp

File mahasiswa.cpp merupakan file implementasi dari ADT mahasiswa. File ini mengandung definisi logika dari fungsi-fungsi yang sebelumnya dideklarasikan di mahasiswa.h.

 - inputMhs(mahasiswa &m) digunakan untuk menerima input data mahasiswa dari pengguna melalui terminal, parameter menggunakan &m agar data yang diinput langsung tersimpan ke dalam variabel asli. Fungsi ini memiliki tiga input yaitu nim, nilai1, dan nilai2 kemudian menyimpan pada anggota struct mahasiswa.
 - rata2(mahasiswa m) digunakan untuk menghitung dan mengembalikan nilai rata-rata dari dua nilai mahasiswa (nilai1 dan nilai2) dengan rumus $\text{rata2} = (\text{nilai1} + \text{nilai2}) / 2$ dan mengembalikan hasil dengan tipe data float agar bisa desimal.
 - Variabel y kemudian diisi dengan nilai yang ditunjuk oleh px menggunakan operator dereferensi *.
- Source code main.cpp

File main.cpp merupakan program utama yang bertugas menjalankan fungsi-fungsi dari ADT

mahasiswa. Program melakukan beberapa tahapan sebagai berikut:

- Mendeklarasikan variable mhs bertipe mahasiswa.
- Memanggil fungsi inputMhs(mhs) untuk meminta pengguna mengisi data mahasiswa yaitu nama dan nilai.
- Memanggil fungsi rata2(mhs) untuk menghitung rata-rata dua nilai yang telah dimasukkan pengguna.
- Kemudian fungsi-fungsi akan mengembalikan nilai yang telah di dapat dari perhitungan dan ditampilkan melalui main.cpp

C. Unguided

1. Menyimpan data mahasiswa

Source code mahasiswa.h

```
#ifndef MAHASISWA_H_INCLUDED
#define MAHASISWA_H_INCLUDED

#include <string>
using namespace std;

struct Mahasiswa {
    string nama;
    string nim;
    float uts;
    float uas;
    float tugas;
    float nilaiAkhir;
};
```

```

void inputMahasiswa(Mahasiswa &m);
float hitungNilaiAkhir(Mahasiswa m);
void tampilMahasiswa(Mahasiswa m);

#endif

```

Source code mahasiswa.cpp

```

#include "mahasiswa.h"
#include <iostream>
using namespace std;

void inputMahasiswa(Mahasiswa &m) {
    cout << "Masukkan Nama    : ";
    getline(cin >> ws, m.nama);
    cout << "Masukkan NIM      : ";
    cin >> m.nim;
    cout << "Masukkan Nilai UTS   : ";
    cin >> m.uts;
    cout << "Masukkan Nilai UAS   : ";
    cin >> m.uas;
    cout << "Masukkan Nilai Tugas : ";
    cin >> m.tugas;

    m.nilaiAkhir = hitungNilaiAkhir(m);
}

float hitungNilaiAkhir(Mahasiswa m) {
    return (0.3 * m.uts) + (0.4 * m.uas) + (0.3 * m.tugas);
}

void tampilMahasiswa(Mahasiswa m) {
    cout << "\nNama          : " << m.nama << endl;
    cout << "NIM           : " << m.nim << endl;
    cout << "UTS          : " << m.uts << endl;
    cout << "UAS          : " << m.uas << endl;
    cout << "Tugas        : " << m.tugas << endl;
    cout << "Nilai Akhir  : " << m.nilaiAkhir << endl;
}

```


Source code main.cpp

```
#include <iostream>
#include "mahasiswa.h"
using namespace std;

int main() {
    Mahasiswa data[10];
    int jumlah;

    cout << "Masukkan jumlah mahasiswa (maks 10): ";
    cin >> jumlah;

    if (jumlah > 10) jumlah = 10;

    cout << "\n=== Input Data Mahasiswa ===\n";
    for (int i = 0; i < jumlah; i++) {
        cout << "\nData ke-" << i + 1 << endl;
        inputMahasiswa(data[i]);
    }

    cout << "\n=== Daftar Nilai Mahasiswa ===\n";
    for (int i = 0; i < jumlah; i++) {
        cout << "\nMahasiswa ke-" << i + 1;
        tampilMahasiswa(data[i]);
    }

    return 0;
}
```

Screenshot Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS I:\ATELYU\semester_3\struktur_data_ptk\modul3\soal1> .\a.exe
Masukkan jumlah mahasiswa (maks 10): 2

=== Input Data Mahasiswa ===

Data ke-1
Masukkan Nama : Juna
Masukkan NIM : 103112400281
Masukkan Nilai UTS : 87
Masukkan Nilai UAS : 90
Masukkan Nilai Tugas : 100

Data ke-2
Masukkan Nama : Zaza
Masukkan NIM : 103112400217
Masukkan Nilai UTS : 80
Masukkan Nilai UAS : 95
Masukkan Nilai Tugas : 90

=== Daftar Nilai Mahasiswa ===

Mahasiswa ke-1
Nama : Juna
NIM : 103112400281
UTS : 87
UAS : 90
Tugas : 100
Nilai Akhir : 92.1

Mahasiswa ke-2
Nama : Zaza
NIM : 103112400217
UTS : 80
UAS : 95
Tugas : 90
Nilai Akhir : 89
PS I:\ATELYU\semester_3\struktur_data_ptk\modul3\soal1> 
```

Deskripsi:

Program pertama digunakan untuk menyimpan data maksimal 10 mahasiswa yang terdiri dari nama, NIM, nilai UTS, UAS, tugas, dan nilai akhir. Program ini menerapkan konsep Abstract Data Type (ADT) yang membagi kode menjadi tiga bagian, yaitu *header* (mahasiswa.h), *implementasi fungsi* (mahasiswa.cpp), dan *program utama* (main.cpp). Program berjalan dengan Langkah kerja sebagai berikut:

- Deklarasi konstanta dan variable
Langkah pertama yaitu pembuatan tipe data struct Mahasiswa yang berisi nama, nim, uts, uas, tugas, dan nilaiAkhir. Selain itu, dibuat array data[10] untuk menyimpan maksimal 10 data mahasiswa. Fungsi inputMahasiswa(), hitungNilaiAkhir(), dan

tampilMahasiswa() juga dideklarasikan di file header mahasiswa.h sebagai bagian dari ADT.

- **Input data mahasiswa**
Program meminta pengguna untuk memasukkan jumlah mahasiswa yang akan diinput (maksimal 10). Melalui perulangan for, fungsi inputMahasiswa() dipanggil untuk menerima data setiap mahasiswa, yang meliputi nama, NIM, dan nilai UTS, UAS, serta tugas.
- **Perhitungan nilai akhir**
Setiap kali data mahasiswa dimasukkan, fungsi hitungNilaiAkhir() dipanggil untuk menghitung nilai akhir menggunakan rumus: $\text{nilaiAkhir} = 0.3 * \text{uts} + 0.4 * \text{uas} + 0.3 * \text{tugas}$; Hasil perhitungan disimpan dalam field nilaiAkhir milik struct mahasiswa yang bersangkutan.
- **Menampilkan data nilai**
Setelah semua data selesai diinput dan dihitung, program menampilkan hasilnya dengan memanggil fungsi tampilMahasiswa(). Fungsi ini menampilkan nama, NIM, nilai UTS, UAS, Tugas, serta Nilai Akhir setiap mahasiswa. Proses ini dilakukan untuk semua data mahasiswa yang tersimpan di dalam array.

2. Implementasi ADT Pelajaran

Source code pelajaran.h

```
#ifndef PELAJARAN_H_INCLUDED
#define PELAJARAN_H_INCLUDED

#include <string>
using namespace std;

struct Pelajaran {
    string namaMapel;
    string kodeMapel;
};
```

```
Pelajaran create_pelajaran(string namaPel, string kodePel);  
void tampil_pelajaran(Pelajaran pel);  
  
#endif
```

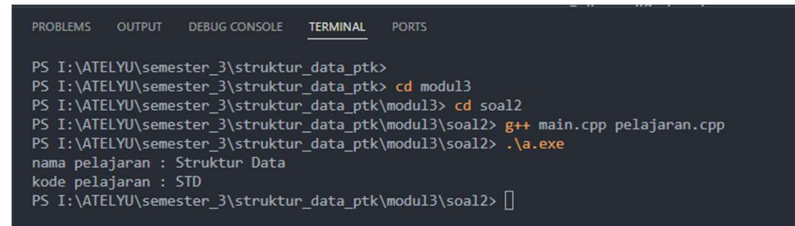
Source code pelajaran.cpp

```
#include "pelajaran.h"  
#include <iostream>  
using namespace std;  
  
Pelajaran create_pelajaran(string namaPel, string kodePel) {  
    Pelajaran p;  
    p.namaMapel = namaPel;  
    p.kodeMapel = kodePel;  
    return p;  
}  
  
void tampil_pelajaran(Pelajaran pel) {  
    cout << "nama pelajaran : " << pel.namaMapel << endl;  
    cout << "kode pelajaran : " << pel.kodeMapel << endl;  
}
```

Source code main.cpp

```
#include <iostream>  
#include "pelajaran.h"  
using namespace std;  
  
int main() {  
    string namapel = "Struktur Data";  
    string kodepel = "STD";  
  
    Pelajaran pel = create_pelajaran(namelapel, kodepel);  
  
    tampil_pelajaran(pel);  
  
    return 0;  
}
```

Screenshot Output:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS I:\ATELYU\semester_3\struktur_data_ptk>
PS I:\ATELYU\semester_3\struktur_data_ptk> cd modul3
PS I:\ATELYU\semester_3\struktur_data_ptk\modul3> cd soa12
PS I:\ATELYU\semester_3\struktur_data_ptk\modul3\soa12> g++ main.cpp pelajaran.cpp
PS I:\ATELYU\semester_3\struktur_data_ptk\modul3\soa12> .\a.exe
nama pelajaran : Struktur Data
kode pelajaran : STD
PS I:\ATELYU\semester_3\struktur_data_ptk\modul3\soa12> █
```

Deskripsi:

Program diatas berfungsi untuk membuat dan menampilkan data mata pelajaran menggunakan konsep Abstract Data Type (ADT). Tipe data struct Pelajaran didefinisikan di file pelajaran.h, berisi dua atribut, yaitu namaMapel dan kodeMapel. Program juga memiliki dua operasi utama: fungsi `create_pelajaran()` sebagai pembentuk data pelajaran, serta prosedur `tampil_pelajaran()` untuk menampilkan hasilnya.. Langkah kerja program diatas sebagai berikut:

- deklarasi variabel dan pointer
Program dimulai dengan pembuatan struct Pelajaran yang berisi dua atribut, yaitu namaMapel untuk menyimpan nama mata pelajaran dan kodeMapel untuk menyimpan kode pelajaran. Di file header pelajaran.h, juga dideklarasikan dua operasi:
`create_pelajaran()` sebagai fungsi pembentuk objek, dan `tampil_pelajaran()` untuk menampilkan hasilnya.
- Membuat objek pelajaran
Di dalam program utama (main.cpp), dua variabel string dideklarasikan, yaitu `namapel` dan `kodepel`. Keduanya diisi dengan nilai awal, misalnya "Struktur Data" dan "STD". Nilai ini kemudian dikirim ke fungsi `create_pelajaran()` untuk membentuk objek Pelajaran baru.
- Menampilkan data pelajaran

Setelah objek pel terbentuk, fungsi tampil_pelajaran() dipanggil untuk menampilkan hasil ke layar. Hasilnya berupa dua baris keluaran yang menampilkan nama pelajaran dan kode pelajaran.

3. implementasi array dua dimensi dengan ADT

Source code arrayDuaD.h

```
#ifndef ARRAY2D_H_INCLUDED
#define ARRAY2D_H_INCLUDED

void tampilArray(int arr[3][3]);
void tukarArray(int arr1[3][3], int arr2[3][3], int baris, int
kolom);
void tukarPointer(int *p1, int *p2);

#endif
```

Source code arrayDuaD.cpp

```
#include "arrayDuaD.h"
#include <iostream>
using namespace std;

void tampilArray(int arr[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << arr[i][j] << "\t";
        }
        cout << endl;
    }
}

void tukarArray(int arr1[3][3], int arr2[3][3], int baris, int
kolom) {
    int temp = arr1[baris][kolom];
    arr1[baris][kolom] = arr2[baris][kolom];
    arr2[baris][kolom] = temp;
}
```

```

}

void tukarPointer(int *p1, int *p2) {
    int temp = *p1;
    *p1 = *p2;
    *p2 = temp;
}

```

Source code main.cpp

```

#include <iostream>
#include "arrayDuaD.h"
using namespace std;

int main() {
    int arr1[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    int arr2[3][3] = {
        {9, 8, 7},
        {6, 5, 4},
        {3, 2, 1}
    };

    cout << "=== Array 1 Sebelum Ditukar ===\n";
    tampilArray(arr1);

    cout << "\n=== Array 2 Sebelum Ditukar ===\n";
    tampilArray(arr2);

    tukarArray(arr1, arr2, 1, 1);

    cout << "\n=== Setelah Menukar Elemen (1,1) ===\n";
    cout << "Array 1:\n";
    tampilArray(arr1);

    cout << "\nArray 2:\n";
    tampilArray(arr2);
}

```

```

int a = 10, b = 20;
int *p1 = &a;
int *p2 = &b;

cout << "\nNilai sebelum ditukar melalui pointer:\n";
cout << "a = " << a << ", b = " << b << endl;

tukarPointer(p1, p2);

cout << "Nilai setelah ditukar melalui pointer:\n";
cout << "a = " << a << ", b = " << b << endl;

return 0;
}

```

Screenshot Output:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS I:\ATELYU\semester_3\struktur_data_ptk\modul3\soal3> g++ main.cpp arrayDuaD.cpp
PS I:\ATELYU\semester_3\struktur_data_ptk\modul3\soal3> .\a.exe
=== Array 1 Sebelum Ditukar ===
1      2      3
4      5      6
7      8      9

=== Array 2 Sebelum Ditukar ===
9      8      7
6      5      4
3      2      1

=== Setelah Menukar Elemen (1,1) ===
Array 1:
1      2      3
4      5      6
7      8      9

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

Nilai sebelum ditukar melalui pointer:
a = 10, b = 20
Nilai setelah ditukar melalui pointer:
a = 20, b = 10
PS I:\ATELYU\semester_3\struktur_data_ptk\modul3\soal3> 

```

Deskripsi:

Program di atas digunakan untuk memanipulasi dua buah array 2 dimensi berukuran 3×3 serta dua buah pointer integer, dengan menerapkan prinsip Abstract Data Type (ADT) untuk pengelolaan fungsi-fungsinya.

Program ini dibagi menjadi tiga bagian: file *header* (arrayDuaD.h), file implementasi (arrayDuaD.cpp), dan program utama (main.cpp).. Langkah kerja program sebagai berikut:

- Deklarasi array dan pointer
Program mendefinisikan dua array dua dimensi `arr1[3][3]` dan `arr2[3][3]`, masing-masing berukuran 3×3 , serta dua pointer integer `p1` dan `p2`. Fungsi-fungsi ADT yang digunakan adalah `tampilArray()`, `tukarArray()`, dan `tukarPointer()`.
- Menampilkan isi array awal
Sebelum dilakukan pertukaran, program menampilkan isi kedua array menggunakan fungsi `tampilArray()`. Fungsi ini menggunakan dua perulangan bersarang (`for`) untuk mencetak setiap elemen array dalam bentuk matriks 3×3 .
- Menukar isi dua array pada posisi tertentu
Program memanggil fungsi `tukarArray(arr1, arr2, baris, kolom)` untuk menukar nilai antara dua array pada indeks yang ditentukan, misalnya (1,1). Proses pertukaran dilakukan menggunakan variabel sementara `temp` untuk menyimpan nilai sementara agar data tidak hilang.
- Menukar nilai melalui pointer
Setelah operasi array selesai, dua variabel `a` dan `b` diinisialisasi (misalnya `a = 10` dan `b = 20`). Pointer `p1` dan `p2` kemudian diarahkan ke alamat variabel tersebut. Fungsi `tukarPointer(p1, p2)` digunakan untuk menukar nilai variabel yang ditunjuk oleh pointer menggunakan dereferensi (`*p1` dan `*p2`).
- Menampilkan hasil akhir
Program menampilkan kembali isi kedua array setelah pertukaran, serta menampilkan nilai variabel `a` dan `b` sebelum dan sesudah proses tukar pointer. Hasil menunjukkan bahwa elemen array dan nilai variabel berhasil ditukar sesuai posisi dan alamat memorinya.

D. Kesimpulan

Dari praktikum yang telah dilakukan, dapat disimpulkan bahwa konsep Abstract Data Type (ADT) berperan penting dalam membuat program yang terstruktur, efisien, dan mudah dikembangkan. Dengan memisahkan antara definisi tipe data, implementasi fungsi, serta penggunaannya dalam program utama, kita dapat membangun kode yang lebih rapi dan mudah dikelola. Penerapan ADT pada berbagai studi kasus seperti pengelolaan data mahasiswa, pembuatan tipe data pelajaran, hingga operasi pada array dua dimensi dan pointer menunjukkan bahwa abstraksi data mempermudah proses pengembangan tanpa harus memahami detail implementasi setiap fungsi. Selain itu, pendekatan modular ini meningkatkan fleksibilitas dan keterbacaan kode, karena setiap bagian memiliki tanggung jawab yang jelas. Secara keseluruhan, praktikum ini memberikan pemahaman nyata tentang bagaimana prinsip abstraksi dan enkapsulasi dalam ADT mampu mendukung pembuatan program yang lebih sistematis dan profesional.

E. Referensi

Srihith, I. D., Donald, A., Srinivas, T. A. S., Anjali, D., & Varaprasad, R. (2022). *The backbone of computing: An exploration of data structures. International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*.

