

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN**

**MODUL 8  
“ ALGORITMA SEARCHING”**



**DISUSUN OLEH :**

**BINTANG YUDHISTIRA  
(2311102052)**

**DOSEN :**

**Wahyu Andi Saputra, S.Pd. , M.Eng**

**PROGRAM STUDI S1 INFORMATIKA  
FAKULTAS INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
PURWOKERTO  
2024**

## DASAR TEORI

### SEARCHING

Searching adalah metode pencarian informasi dalam suatu aplikasi dengan suatu kunci (key). Pencarian diperlukan untuk mencari informasi khusus dari table pada saat lokasi yang pasti dari informasi tersebut sebelumnya tidak diketahui.

Searching berarti mencari data yang dibutuhkan. Searching dalam pemrograman bisa dilakukan untuk mencari data yang ada di dalam memory komputer. Dalam kehidupan sehari-hari kita juga sering melakukan kegiatan searching seperti mencari data/informasi yang ada dalam internet.

Pencarian di perlukan untuk mencari informasi khusus dari tabel pada saat lokasi yang pasti dari informasi tersebut sebelumnya tidak diketahui. Pencarian selalu dinyatakan dengan referensi pada adanya sekelompok data yang tersimpan secara terorganisasi, kelompok data tersebut kita sebut tabel.

Array memungkinkan untuk menyimpan nilai yang bertipe sama. Operasi yang umum dalam array adalah Sequential Search dan Binary search. Perbedaan dari kedua teknik ini terletak pada keadaan data.

#### 1. Konsep Dasar Searching

Searching adalah proses mendapatkan (retrieve) informasi berdasarkan kunci tertentu dari sejumlah informasi yang telah disimpan

#### 2. Squential Search

Teknik ini merupakan teknik pengurutan data yang paling sederhana dan paling mudah dimengerti maupun diterapkan. Prinsip dasar dari teknik Insertion Sort ini, yaitu seolah-olah mengambil sebuah elemen dari tempat tertentu, kemudian menyisipkannya (insert) ke suatu tempat hingga elemen-elemen lain bergeser ke belakang.

Cara mengurutkannya adalah dicek satu persatu mulai dari yang kedua sampai dengan yang terakhir.

Apabila ditemukan data yang lebih kecil dari data sebelumnya, maka data tersebut disisipkan pada posisi yang sesuai.

```
void insertion_sort(int arr[], int length) {
    int i, j ,tmp;
    for (i = 1; i < length; i++) {
        j = i;
        while (j > 0 && arr[j - 1] > arr[j]) {
            tmp = arr[j];
            arr[j] = arr[j - 1];
            arr[j - 1] = tmp;
            j--;
        } //end of while loop
    } //end of for loop
}
```

### 3. Bubble sort

Cara mengurutkannya adalah membandingkan elemen yang sekarang dengan elemen yang berikutnya. Jika elemen sekarang > elemen berikutnya, maka tukar

```
void bubble_sort(int arr[], int length){
    bool not_sorted = true;
    int j=0,tmp;

    while (not_sorted){
        not_sorted = false;
        j++;
        for (int i = 0; i < length - j; i++){
            if (arr[i] > arr[i + 1]) {
                tmp = arr[i];
                arr[i] = arr[i + 1];
                arr[i + 1] = tmp;
                not_sorted = true;
            } //end of if
        } //end of for loop
    } //end of while loop
} //end of bubble_sort
```

### 4. Selection sort

Cara mengurutkannya adalah dengan membandingkan elemen sekarang dengan elemen yang berikutnya sampai terakhir. Jika ditemukan elemen paling kecil, kemudian ditukar dengan elemen sekarang.

```
void selectSort(int arr[], int n) {
    int pos_min,temp;
    for (int i=0; i < n-1; i++) {
        pos_min = i;
        for (int j=i+1; j < n; j++) {
            if (arr[j] < arr[pos_min])
                pos_min=j;
        }
        if (pos_min != i) {
            temp = arr[i];
            arr[i] = arr[pos_min];
            arr[pos_min] = temp;
        }
    }
}
```

## GUIDED LATIHAN KELAS

### 1. Latihan 1

#### SOURCE CODE

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

int main()
{
    int n = 10;
    int data[n] = {9, 4, 1, 7, 5, 12, 4, 13, 4, 10};
    int cari = 10;
    bool ketemu = false;
    int i;
    // algoritma sequential search
    for (i = 0; i < n; i++)
    {
        if (data[i] == cari)
        {
            ketemu = true;
            break;
        }
    }

    cout << "\t Program Sequential Search Sederhana\n " << endl;
    cout << " data: {9, 4, 1, 7, 5, 12, 4, 13, 4, 10}" << endl;

    if (ketemu)
    {
        cout << "\n angka " << cari << " ditemukan pada indeks ke - "
        << i << endl;
    }
    else
    {
        cout << cari << " tidak dapat ditemukan pada data."
    }
}
```

```

        << endl;
    }

    return 0;
}

```

## OUTPUTNYA :

The screenshot shows a C++ IDE with the following code in `Guided1.cpp`:

```

int main()
{
    int n = 10;
    int data[n] = {9, 4, 1, 7, 5, 12, 4, 13, 4, 10};
    int cari = 10;
    bool ketemu = false;
    int i;

```

The terminal output shows the execution of the program:

```

Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 8 Searching\ ; if ($?) { g++ Guided1.cpp -o Guided1 }
; if ($?) { .\Guided1 }
Program Sequential Search Sederhana

data: {9, 4, 1, 7, 5, 12, 4, 13, 4, 10}

angka 10 ditemukan pada indeks ke - 9
PS C:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan A

```

A small window titled "Nama" is open, showing the user's name and NIM:

```

Nama : Bintang Yudhistira
Nim : 2311102052

```

## DESKRIPSI PROGRAM :

Program ini adalah program yang dibuat untuk mengimplementasikan algoritma Searching yang dimana jika nilai cari sama dengan nilai yang ada dalam array, maka program akan menampilkan output yang menyatakan bahwa nilai tersebut ada dalam array indeks ke berapa. Sebaliknya jika nilai cari tidak ada yg sama dengan nilai dalam array, maka program akan menampilkan output yang menyatakann bahwa nilai tidak ditemukan dalam array.

## 2. Latihan 2

### SOURCE CODE

```
#include <iostream>
using namespace std;
#include <conio.h>
#include <iomanip>
int DATA[7] = {1, 8, 2, 5, 4, 9, 7};
int cari;
void selection_sort()
{
    int temp, min, i, j;
    for (i = 0; i < 7; i++)
    {
        min = i;
        for (j = i + 1; j < 7; j++)
        {
            if (DATA[j] < DATA[min])
            {
                min = j;
            }
        }
        temp = DATA[i];
        DATA[i] = DATA[min];
        DATA[min] = temp;
    }
}

void binarysearch()
{
    // searching
    int awal, akhir, tengah, b_flag = 0;
    awal = 0;
    akhir = 7;
    while (b_flag == 0 && awal <= akhir)
    {
        tengah = (awal + akhir) / 2;
        if (DATA[tengah] == cari)
        {
            b_flag = 1;
            break;
        }
        else if (DATA[tengah] < cari)
            awal = tengah + 1;
        else
            akhir = tengah - 1;
    }
    if (b_flag == 1)
        cout << "\n Data ditemukan pada index ke-" << tengah << endl;
    else
        cout << "\n Data tidak ditemukan\n";
}

int main()
{

```

```

cout << "\t BINARY SEARCH " << endl;
cout << "\n Data : ";
// tampilkan DATA awal
for (int x = 0; x < 7; x++)
    cout << setw(3) << DATA[x];
cout << endl;
cout << "\n Masukkan DATA yang ingin Anda cari :";
cin >> cari;
cout << "\n Data diurutkan : ";
// urutkan DATA dengan selection sort
selection_sort();
// tampilkan DATA setelah diurutkan
for (int x = 0; x < 7; x++)
    cout << setw(3) << DATA[x];
cout << endl;
binarysearch();
_getche();
return EXIT_SUCCESS;
}

```

## OUTPUTNYA :

```

1 #include <iostream>
2 using namespace std;
3 #include <conio.h>
4 #include <iomanip>
5 int DATA[7] = {1, 8, 2, 5, 4, 9, 7};
6 int cari;
7 void selection_sort()
8 {
9     int temp, min, i, j;
10    for (i = 0; i < 7; i++)
11    {

```

OUTPUT:

```

Prak 8 Searching

Data :  1  8  2  5  4  9  7

Masukkan DATA yang ingin Anda cari : 1

Data diurutkan :  1  2  4  5  7  8  9

Data ditemukan pada index ke-0

```

## DESKRIPSI PROGRAM :

Program ini dibuat untuk mengimplementasikan algoritma searching pada array. Sama halnya seperti tadi bedanya program ini mengharuskan kita untuk menginputkan nilai yang akan di cari, jika nilai yg di cari sama dengan nilai yang ada di dalam array maka program akan menampilkan bahwa data yang di cari itu ada di dalam array. Sebaliknya jika tidak ada maka program akan menyatakan data yang kita cari itu tidak ada di di dalam array.

Fungsi `selection_sort` mengurutkan array DATA menggunakan algoritma selection sort, di mana elemen terkecil dari bagian yang belum terurut dipilih dan ditukar dengan elemen pertama dari bagian tersebut. Proses ini diulang sampai seluruh array terurut. Fungsi `binarysearch` mencari nilai cari dalam array DATA yang sudah terurut menggunakan algoritma binary search. Algoritma ini membagi dua array dan menentukan apakah nilai yang dicari berada di bagian kiri atau kanan, kemudian mempersempit ruang pencarian ke bagian tersebut. Proses ini diulang sampai nilai ditemukan atau ruang pencarian habis.

Program dimulai dari fungsi main, yang menampilkan judul "BINARY SEARCH" dan kemudian menampilkan data awal dari array DATA. Pengguna diminta untuk memasukkan nilai yang ingin dicari (cari). Setelah itu, array DATA diurutkan menggunakan fungsi `selection_sort`, dan array yang sudah terurut ditampilkan. Program kemudian memanggil fungsi `binarysearch` untuk mencari nilai cari dalam array yang sudah terurut dan menampilkan hasil pencarian. Fungsi `getche()` digunakan untuk menunggu pengguna menekan tombol sebelum program berakhir. Program ini menunjukkan proses mengurutkan array dan kemudian mencari sebuah nilai dalam array yang sudah diurutkan menggunakan pencarian biner, yang lebih efisien daripada pencarian linier untuk array yang sudah terurut.

## UNGUIDED

### 1. Tugas 1

#### SOURCE CODE

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

void selection_sort(char data[], int n)
{
    for (int i = 0; i < n - 1; ++i)
    {
        int min_idx = i;
```



```

        for (int j = i + 1; j < n; ++j)
        {
            if (data[j] < data[min_idx])
            {
                min_idx = j;
            }
        }
        swap(data[i], data[min_idx]);
    }
}

bool binary_search(const char data[], int n, char target)
{
    int left = 0, right = n - 1;
    while (left <= right)
    {
        int middle = left + (right - left) / 2;
        if (data[middle] == target)
        {
            return true;
        }
        else if (data[middle] < target)
        {
            left = middle + 1;
        }
        else
        {
            right = middle - 1;
        }
    }
    return false;
}

int main()
{
    string input;
    char target;
    cout << "Masukkan sebuah kalimat: ";
    getline(cin, input);
    cout << "Masukkan huruf yang ingin dicari: ";
    cin >> target;

    // Mengubah kalimat menjadi array karakter
    int n = input.size();
    char data[n];
    for (int i = 0; i < n; ++i)
    {
        data[i] = input[i];
    }

    // Mengurutkan array karakter
    selection_sort(data, n);

```

```

// Menampilkan array karakter setelah diurutkan
cout << "Kalimat setelah diurutkan: ";
for (int i = 0; i < n; ++i)
{
    cout << data[i] << ' ';
}
cout << endl;

// Melakukan pencarian biner
if (binary_search(data, n, target))
{
    cout << "Huruf '" << target << "' ditemukan dalam kalimat." <<
endl;
}
else
{
    cout << "Huruf '" << target << "' tidak ditemukan dalam
kalimat." << endl;
}

return 0;
}

```

## OUTPUTNYA :

```

PS C:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 8 Searching> cd "c:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 8 Searching"; if ($?) { g++ Unguided1.cpp -o Unguided1 } ; if ($?) { .\Unguided1 }
Masukkan sebuah kalimat: yudhis
Masukkan huruf yang ingin dicari: y
Kalimat setelah diurutkan: d h i s u y
Huruf 'y' ditemukan dalam kalimat.
PS C:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 8 Searching>

```

## DESKRIPSI PROGRAM :

Kode ini membaca sebuah kalimat dan karakter dari pengguna, mengurutkan karakter dalam kalimat menggunakan algoritma selection sort, lalu mencari karakter tersebut dalam kalimat yang sudah diurutkan menggunakan algoritma binary search. Program

dimulai dengan memasukkan pustaka `iostream` untuk input dan output serta algoritma untuk fungsi-fungsi algoritma seperti `swap`. Fungsi `selection_sort` mengurutkan array karakter data yang berukuran `n` dengan cara mencari elemen terkecil di bagian yang belum terurut dan menukarnya dengan elemen pertama di bagian tersebut, diulang sampai seluruh array terurut. Fungsi `binary_search` mencari karakter target dalam array data yang sudah terurut dengan membagi dua array dan menentukan apakah karakter yang dicari berada di bagian kiri atau kanan dari tengah, mempersempit ruang pencarian ke bagian tersebut, diulang sampai karakter ditemukan atau ruang pencarian habis. Program kemudian meminta pengguna untuk memasukkan sebuah kalimat dan karakter yang ingin dicari dalam kalimat tersebut, mengubah kalimat menjadi array karakter data dengan ukuran `n` yang sama dengan panjang kalimat, mengurutkan array data menggunakan `selection_sort`, dan menampilkan kalimat yang sudah diurutkan sebagai array karakter yang diurutkan. Selanjutnya, program menggunakan `binary_search` untuk mencari karakter target dalam array data yang sudah diurutkan dan menampilkan hasil pencarian, yaitu apakah karakter ditemukan atau tidak. Program ini menunjukkan bagaimana mengurutkan array karakter dan kemudian mencari sebuah karakter dalam array tersebut menggunakan algoritma pencarian yang lebih efisien setelah data diurutkan.

## 2. Tugas 2

### SOURCE CODE

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

int main()
{

    string kalimat;
    int jumlah = 0;

    cout << "Jumlah Huruf Vokal" << endl; cout << "Masukkan kalimat : ";
    getline(cin, kalimat);

    for (int i = 0; i < kalimat.length(); i++) { char c =
    tolower(kalimat[i]);
    if (c == 'a' || c == 'i' || c == 'u' || c == 'e' || c == 'o')
    jumlah++;
    }

    cout << "\nKalimat yang dimasukkan memiliki " << jumlah << " huruf
```

```
vokal."; return 0;
}
```

## OUTPUTNYA :

The screenshot shows a C++ IDE with the following components:

- EXPLORER:** Lists files including `Guided1.cpp`, `Guided2.cpp`, `Unguided1.cpp`, and `Unguided2.cpp`.
- EDITOR:** Displays the code for `Unguided2.cpp`, which includes `<iostream>`, uses `std` namespace, and contains a `main` function. The function declares a `string kalimat` and an `int jumlah = 0`. It prompts the user to enter a sentence and iterates through each character, converting it to lowercase and checking if it is a vowel.
- TERMINAL:** Shows the output of the program:
 

```
PS C:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 8 Searching> .\Unguided2.exe
Jumlah Huruf Vokal
Masukkan kalimat : hahaha

Kalimat yang dimasukkan memiliki 3 huruf vokal.
PS C:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 8 Searching>
```

## DSKRIPSI PROGRAM :

Program ini menghitung jumlah huruf vokal dalam sebuah kalimat yang dimasukkan oleh pengguna. Dimulai dengan mengimpor pustaka `iostream` untuk fungsi input dan output, di dalam fungsi `main`, sebuah string `kalimat` dan variabel `jumlah` diinisialisasi untuk menyimpan kalimat yang dimasukkan dan jumlah huruf vokal dalam kalimat tersebut. Program menampilkan judul "Jumlah Huruf Vokal" dan meminta pengguna untuk memasukkan sebuah kalimat, menggunakan `getline` untuk membaca seluruh kalimat. Program kemudian menggunakan loop `for` untuk memeriksa setiap karakter dalam kalimat, mengubahnya menjadi huruf kecil dengan `tolower` untuk memastikan perbandingan tidak sensitif huruf besar/kecil. Jika karakter tersebut adalah salah satu huruf vokal ('a', 'i', 'u', 'e', atau 'o'), variabel `jumlah` ditambah satu. Setelah semua karakter diperiksa, program menampilkan jumlah huruf vokal yang ditemukan dalam kalimat yang dimasukkan. Terakhir, program mengembalikan nilai 0 untuk menunjukkan eksekusi selesai tanpa kesalahan.

### 3. Tugas 3

#### SOURCE CODE

```
#include <iostream>
#include <iomanip>
#include <conio.h>
using namespace std;

int DATA[10]={ 9, 4, 1, 4, 7, 10, 5, 4, 12, 4};

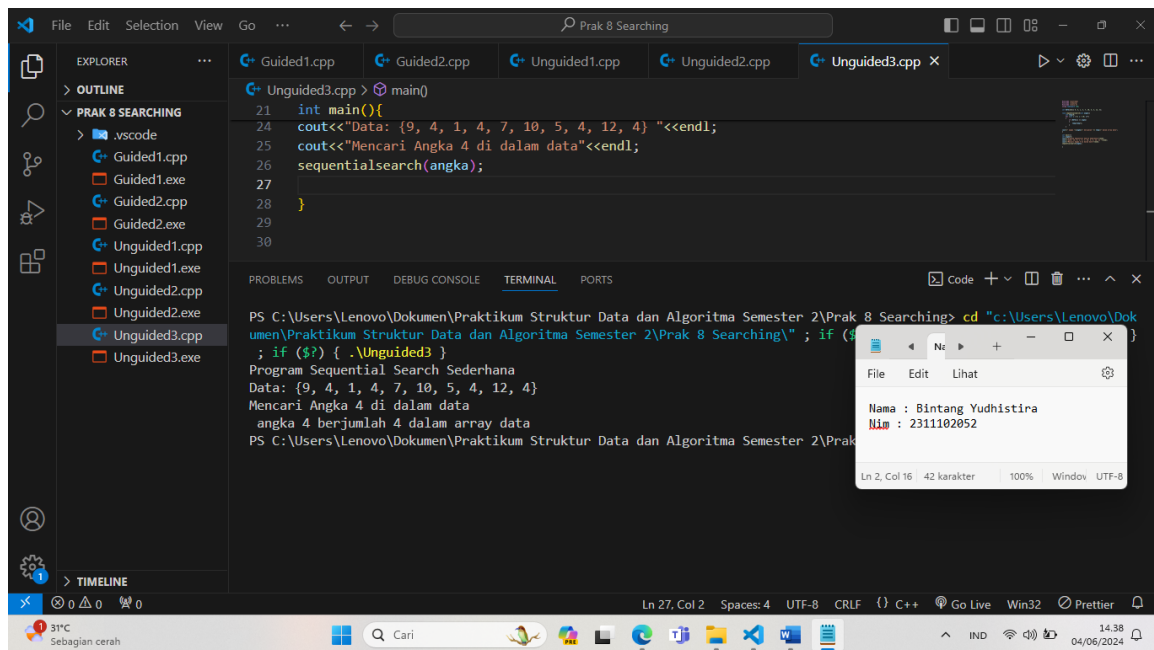
void sequentialsearch(int angka){
    int temp=0;
    for (int i = 0; i < 10; i++)
    {
        if (DATA[i] == angka)
        {
            temp=temp+1;
        }
    }
}

cout<<" angka "<<angka<<" berjumlah "<< temp<<" dalam array data";
}

int main(){
int angka=4;
cout<<"Program Sequential Search Sederhana"<<endl;
cout<<"Data: {9, 4, 1, 4, 7, 10, 5, 4, 12, 4} "<<endl;
cout<<"Mencari Angka 4 di dalam data"<<endl;
sequentialsearch(angka);

}
```

## OUTPUTNYA :



```
21 int main(){
24     cout<<"Data: {9, 4, 1, 4, 7, 10, 5, 4, 12, 4} "<<endl;
25     cout<<"Mencari Angka 4 di dalam data"<<endl;
26     sequentialsearch(angka);
27 }
28
29
30
```

```
PS C:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 8 Searching> cd "c:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 8 Searching"; if ($?) { .\Unguided3 }
Program Sequential Search Sederhana
Data: {9, 4, 1, 4, 7, 10, 5, 4, 12, 4}
Mencari Angka 4 di dalam data
angka 4 berjumlah 4 dalam array data
PS C:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 8 Searching>
```

Nama : Bintang Yudhistina  
Nim : 2311102052

## DESKRIPSI PROGRAM :

Program ini melakukan pencarian sekuensial untuk menghitung jumlah kemunculan sebuah angka dalam array menggunakan bahasa C++. Dimulai dengan mengimpor pustaka `iostream`, `iomanip`, dan `conio.h` untuk fungsi input/output dan manipulasi I/O, program ini mendefinisikan array `DATA` yang berisi sepuluh elemen yang diinisialisasi dengan nilai `{9, 4, 1, 4, 7, 10, 5, 4, 12, 4}`. Fungsi `sequentialsearch` menerima sebuah angka sebagai parameter dan menghitung jumlah kemunculan angka tersebut dalam array `DATA` dengan menggunakan variabel `temp` untuk menyimpan jumlah kemunculan. Dalam loop `for`, setiap elemen array diperiksa apakah sama dengan angka yang dicari, dan jika iya, `temp` ditambah satu. Setelah loop selesai, program menampilkan berapa kali angka tersebut muncul dalam array. Fungsi `main` menginisialisasi angka yang akan dicari sebagai 4, menampilkan beberapa informasi awal tentang program dan data yang digunakan, kemudian memanggil fungsi `sequentialsearch` untuk menghitung dan menampilkan jumlah kemunculan angka 4 dalam array `DATA`. Program ini diakhiri tanpa mengembalikan nilai apapun dari fungsi `main`.

## **KESIMPULAN**

Materi tentang searching melibatkan berbagai teknik untuk menemukan elemen tertentu dalam kumpulan data. Dalam pengembangan perangkat lunak, kemampuan untuk melakukan pencarian secara efisien sangatlah penting karena banyak algoritma dan aplikasi yang bergantung pada kemampuan ini. Dari berbagai teknik pencarian yang ada, beberapa di antaranya termasuk sequential search, binary search, hashing, dan interpolation search. Sequential search merupakan metode pencarian sederhana yang memeriksa setiap elemen satu per satu dari awal sampai akhir kumpulan data. Meskipun sederhana, metode ini kurang efisien untuk dataset besar karena memiliki kompleksitas waktu linier. Di sisi lain, binary search, yang efisien untuk kumpulan data yang sudah diurutkan, membagi kumpulan data menjadi dua bagian dan mencari secara rekursif di salah satu bagian yang berpotensi mengandung elemen yang dicari. Hashing menggunakan fungsi hash untuk mengonversi kunci pencarian menjadi alamat di mana data disimpan atau diharapkan ditemukan, dan memungkinkan pencarian dengan kompleksitas waktu konstan pada rata-rata, sangat efisien untuk dataset besar. Sedangkan interpolation search, serupa dengan binary search, namun lebih cocok untuk dataset yang didistribusikan secara merata, memperkirakan posisi elemen yang dicari berdasarkan nilai dan distribusi elemen di dalam kumpulan data. Pemahaman yang baik tentang teknik-teknik pencarian ini penting dalam pengembangan perangkat lunak untuk memilih metode pencarian yang tepat sesuai dengan sifat dataset dan kebutuhan aplikasi, serta untuk mengoptimalkan kinerja dan efisiensi aplikasi secara keseluruhan.

## REFERENSI

- [1] Asisten Praktikum, “Modul 8 Searching”, Learning Management System, 2024.
- [2] [Apa itu Coding? Penjelasan Untuk Pemula - Dicoding Blog](#)





