

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN
ALGORITMA**

**MODUL VIII
SEARCHING**



Disusun Oleh :

Raka Andriy Shevchenko
2311102054

Dosen

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

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

A. Dasar Teori

Pengertian Algoritma Pencarian:

Algoritma pencarian adalah prosedur langkah demi langkah yang digunakan untuk menemukan data tertentu di antara beberapa data yang tersimpan

1. Binary Search

- a. Binary Search dapat lebih efisien dari sisi waktu:

Binary Search dapat lebih cepat dalam mencari data jika dibandingkan dengan Sequential Search, karena membagi array menjadi dua bagian dan mengecek data di tengah.

- b. Binary Search hanya dapat digunakan pada array yang telah diurutkan:

Binary Search hanya dapat digunakan jika array telah diurutkan secara ascending atau descending, karena algoritma ini membagi array menjadi dua bagian berdasarkan nilai tengah.

- c. Binary Search memerlukan data yang telah diurutkan:

Binary Search memerlukan data yang telah diurutkan sebelumnya, sehingga dapat membagi array menjadi dua bagian dan mengecek data di tengah.

2. Sequential Search

- a. Sequential Search memerlukan memori yang relatif kecil:

Sequential Search memerlukan memori yang relatif kecil karena hanya memerlukan ruang untuk menyimpan data yang sedang diproses.

- b. Sequential Search dapat digunakan pada array yang tidak diurutkan:

Sequential Search dapat digunakan pada array yang tidak diurutkan, karena membandingkan setiap elemen array secara berurutan.

- c. Sequential Search dapat mengalami masalah optimal lokal:

Sequential Search dapat mengalami masalah optimal lokal, karena hanya mencari data yang paling dekat dengan data yang dicari, bukan data yang paling optimal.

B. Guided

a. Guided 1

Source Code:

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

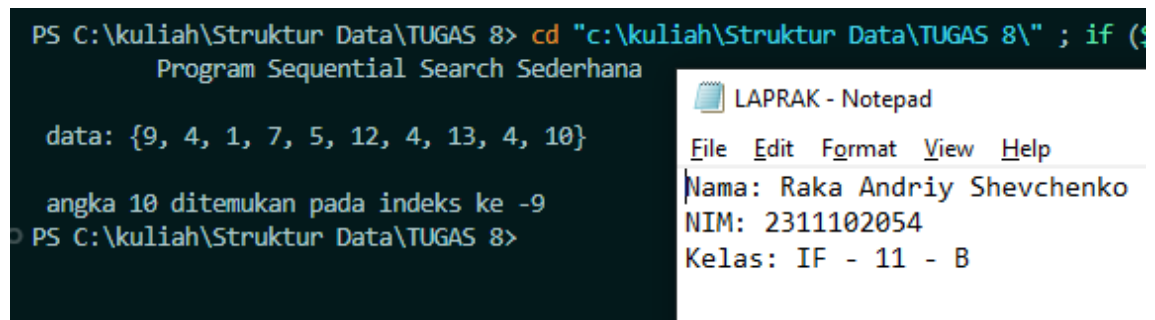
int main()
{
    int n = 10;
    int data[n] = {9, 4, 1, 7, 5, 12, 4, 13, 4, 10};
    int cari = 10;
    bool ketemu = false;
    int i;

    // 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;
}
```

Output:



```
PS C:\kuliah\Struktur Data\TUGAS 8> cd "c:\kuliah\Struktur Data\TUGAS 8\" ; if ($?) {
    Program Sequential Search Sederhana

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

    angka 10 ditemukan pada indeks ke -9
PS C:\kuliah\Struktur Data\TUGAS 8>
```

LAPRAK - Notepad

File Edit Format View Help

Nama: Raka Andriy Shevchenko
NIM: 2311102054
Kelas: IF - 11 - B

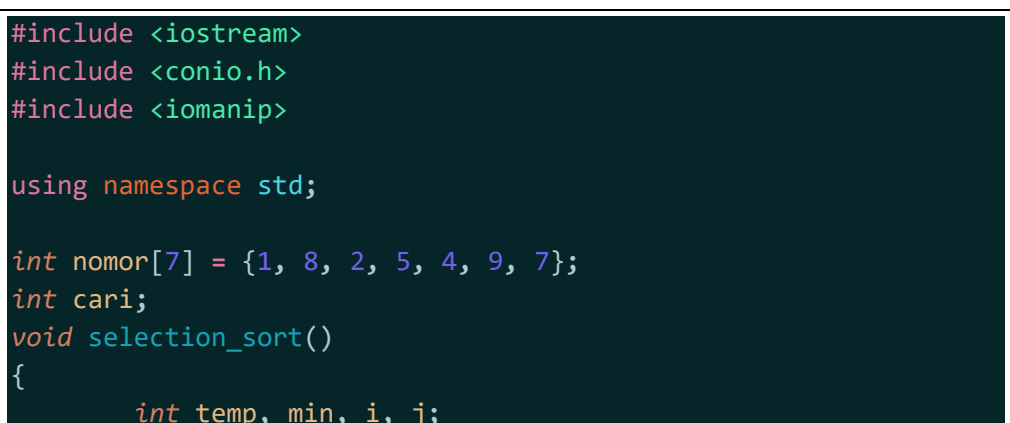
Deskripsi:

Program ini adalah implementasi sederhana dari algoritma Pencarian Berurutan. Program ini mencari nilai tertentu, cari, dalam larik data bilangan bulat. Larik berisi 10 elemen, dan program melakukan iterasi melalui setiap elemen untuk memeriksa apakah elemen tersebut cocok dengan nilai target. Jika ditemukan kecocokan, program akan menetapkan bendera boolean ketemu menjadi true dan keluar dari perulangan. Jika perulangan selesai tanpa menemukan kecocokan, ketemu tetap bernilai salah.

Program kemudian mencetak sebuah pesan yang menunjukkan apakah nilai target ditemukan atau tidak. Jika ditemukan, program ini juga mencetak indeks dari nilai dalam larik. Jika tidak ditemukan, program akan mencetak pesan yang menyatakan bahwa nilai tersebut tidak dapat ditemukan di dalam data.

b. Guided 2

Source Code:



```
#include <iostream>
#include <conio.h>
#include <iomanip>

using namespace std;

int nomor[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(nomor[j]<nomor[min])
                {
                    min=j;
                }
            }
            temp = nomor[i];
            nomor[i] = nomor[min];
            nomor[min] = temp;
        }
    }

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

int main()

```

```

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

```

Output:

The screenshot shows a terminal window with the following output:

```

PS C:\kuliah\Struktur Data\TUGAS 8> cd "c:\kuliah\Struktur Data\TUGAS 8\" ; if
BINARY SEARCH

nomor :   1   8   2   5   4   9   7

Masukkan nomor yang ingin Anda cari :9

nomor diurutkan :   1   2   4   5   7   8   9

nomor ditemukan pada index ke- 6

```

Overlaid on the terminal is a Notepad window titled "LAPRAK - Notepad" with the following text:

```

File Edit Format View Help
Nama: Raka Andriy Shevchenko
NIM: 2311102054
Kelas: IF - 11 - B

```

Deskripsi:

Program ini adalah kombinasi dari dua algoritma: Pengurutan Pilihan dan Pencarian Biner. Program ini pertama-tama mengurutkan larik bilangan bulat menggunakan algoritme Pengurutan Pilihan, lalu melakukan Pencarian Biner untuk menemukan nilai tertentu di dalam larik yang telah diurutkan.

Berikut ini adalah rincian programnya:

Fungsi `selection_sort` mengurutkan larik nomor menggunakan algoritma

Selection Sort. Fungsi ini mengulang-ulang larik, menemukan nilai minimum dan menukarnya dengan elemen saat ini, hingga seluruh larik terurut.

Fungsi `binarysearch` melakukan Pencarian Biner pada larik yang diurutkan untuk menemukan pencarian nilai tertentu. Fungsi ini menggunakan perulangan sementara untuk berulang kali membagi rentang pencarian menjadi dua dan memeriksa apakah elemen tengah cocok dengan nilai target. Jika ditemukan, ia akan mencetak indeks dari nilai dalam larik tersebut. Jika tidak ditemukan, program akan mencetak pesan yang menyatakan bahwa nilai tersebut tidak dapat ditemukan.

Pada fungsi utama, program pertama-tama menampilkan larik asli dan meminta pengguna untuk memasukkan nilai yang akan dicari. Kemudian memanggil fungsi `selection_sort` untuk mengurutkan larik, menampilkan larik yang telah diurutkan, dan akhirnya memanggil fungsi `binarysearch` untuk mencari nilai yang dimasukkan.

Program ini menggunakan pustaka `conio.h` untuk fungsi `_getche`, yang digunakan untuk menghentikan sementara program sebelum keluar.

C. Unguided

a. Unguided 1

Source code:

```
#include <iostream>
#include <iomanip>
#include <cstring>

using namespace std;

int partition(char arr[], int start, int end)
{
    int pivot = arr[start];
    int count = 0;
    for (int i = start + 1; i <= end; i++)
    {
        if (arr[i] <= pivot)
            count++;
    }

    int pivotIndex = start + count;
    swap(arr[pivotIndex], arr[start]);

    int i = start, j = end;

    while (i < pivotIndex && j > pivotIndex)
    {
        while (arr[i] <= pivot)
        {
            i++;
        }

        while (arr[j] > pivot)
        {
            j--;
        }

        if (i < pivotIndex && j > pivotIndex)
        {
            swap(arr[i++], arr[j--]);
        }
    }
}
```



```

        return pivotIndex;
    }

void quickSort(char arr[], int start, int end)
{
    if (start >= end)
        return;

    int p = partition(arr, start, end);

    quickSort(arr, start, p - 1);
    quickSort(arr, p + 1, end);
}

void binary_search(char data[], int n, char cari)
{
    int awal = 0, akhir = n - 1, tengah;
    bool found = false;
    while (!found && awal <= akhir)
    {
        tengah = (awal + akhir) / 2;
        if (data[tengah] == cari)
        {
            found = true;
        }
        else if (data[tengah] < cari)
        {
            awal = tengah + 1;
        }
        else
        {
            akhir = tengah - 1;
        }
    }
    if (found)
    {
        cout << "\n Data ditemukan pada index ke- " << tengah <<
endl;
    }
    else
    {
        cout << "\n Data tidak ditemukan\n";
    }
}

```

```

    }
}

int main()
{
    const int MAX_LENGTH = 100;
    char kalimat[MAX_LENGTH];
    char cleaned[MAX_LENGTH];
    char cari;

    cout << "\t BINARY SEARCH " << endl;
    cout << "\n Masukkan sebuah kalimat : ";
    cin.getline(kalimat, MAX_LENGTH);

    // Remove spaces
    int n = 0;
    for (int i = 0; kalimat[i] != '\0'; i++)
    {
        if (kalimat[i] != ' ')
        {
            cleaned[n++] = kalimat[i];
        }
    }
    cleaned[n] = '\0';

    cout << "\n Data diurutkan : ";
    quickSort(cleaned, 0, n - 1);

    // Tampilkan data setelah diurutkan
    for (int i = 0; i < n; i++)
    {
        cout << cleaned[i];
    }
    cout << endl;

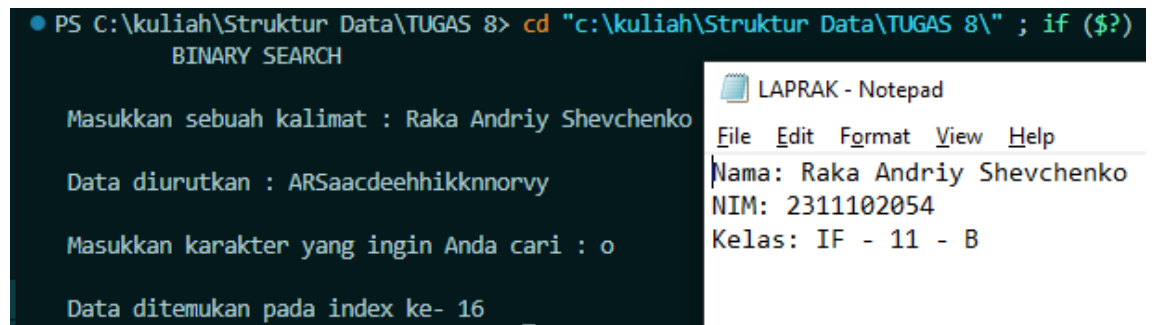
    cout << "\n Masukkan karakter yang ingin Anda cari : ";
    cin >> cari;

    binary_search(cleaned, n, cari);

    return EXIT_SUCCESS;
}

```

Output:



The screenshot shows a Windows command prompt window with a dark background. The prompt is at the C:\kuliah\Struktur Data\TUGAS 8 directory. The user has entered 'cd "c:\kuliah\Struktur Data\TUGAS 8\" ; if (\$?)' and the prompt has returned. Below the prompt, the program output is displayed in white text: 'BINARY SEARCH', 'Masukkan sebuah kalimat : Raka Andriy Shevchenko', 'Data diurutkan : ARSaacdeehhikknorvy', 'Masukkan karakter yang ingin Anda cari : o', and 'Data ditemukan pada index ke- 16'. To the right of the command prompt is a Notepad window titled 'LAPRAK - Notepad'. It has a menu bar with 'File', 'Edit', 'Format', 'View', and 'Help'. The text in the Notepad window is: 'Nama: Raka Andriy Shevchenko', 'NIM: 2311102054', and 'Kelas: IF - 11 - B'.

```
PS C:\kuliah\Struktur Data\TUGAS 8> cd "c:\kuliah\Struktur Data\TUGAS 8\" ; if ($?)
BINARY SEARCH

Masukkan sebuah kalimat : Raka Andriy Shevchenko

Data diurutkan : ARSaacdeehhikknorvy

Masukkan karakter yang ingin Anda cari : o

Data ditemukan pada index ke- 16
```

LAPRAK - Notepad
File Edit Format View Help
Nama: Raka Andriy Shevchenko
NIM: 2311102054
Kelas: IF - 11 - B

Deskripsi:

Program C++ ini adalah kombinasi dari dua algoritma: Pengurutan Cepat dan Pencarian Biner. Program ini pertama-tama mengurutkan larik karakter menggunakan algoritme Quick Sort, dan kemudian melakukan Pencarian Biner untuk menemukan karakter tertentu di dalam larik yang telah diurutkan.

Berikut ini adalah rincian programnya:

Fungsi partisi adalah fungsi pembantu untuk algoritma Quick Sort. Fungsi ini mempartisi larik di sekitar elemen pivot, memastikan bahwa semua elemen yang lebih kecil dari pivot berada di sisi kiri dan semua elemen yang lebih besar dari pivot berada di sisi kanan.

Fungsi quickSort mengimplementasikan algoritma Quick Sort. Fungsi ini secara rekursif memanggil dirinya sendiri untuk mengurutkan larik, menggunakan fungsi partisi untuk membagi larik menjadi sub-larik yang lebih kecil.

Fungsi binary_search melakukan Pencarian Biner pada larik yang telah diurutkan untuk menemukan karakter tertentu. Fungsi ini menggunakan perulangan sementara untuk berulang kali membagi rentang pencarian menjadi dua dan memeriksa apakah elemen tengah cocok dengan karakter target. Jika ditemukan, ia akan mencetak indeks karakter dalam larik. Jika tidak ditemukan, program akan mencetak pesan yang menyatakan bahwa karakter tersebut tidak dapat ditemukan.

Pada fungsi utama, program ini pertama-tama meminta pengguna untuk memasukkan sebuah kalimat. Kemudian menghapus semua spasi dari kalimat tersebut dan menyimpan kalimat yang telah dibersihkan dalam sebuah larik.

Program ini kemudian memanggil fungsi quickSort untuk mengurutkan kalimat yang telah dibersihkan, dan menampilkan kalimat yang telah diurutkan.

b. Unguided 2

Source Code:

```
#include <iostream>
#include <cstring>

using namespace std;

int partition(char arr[], int start, int end)
{
    int pivot = arr[start];
    int count = 0;
    for (int i = start + 1; i <= end; i++)
    {
        if (arr[i] <= pivot)
            count++;
    }

    int pivotIndex = start + count;
    swap(arr[pivotIndex], arr[start]);

    int i = start, j = end;

    while (i < pivotIndex && j > pivotIndex)
    {
        while (arr[i] <= pivot)
        {
            i++;
        }

        while (arr[j] > pivot)
        {
            j--;
        }
    }
}
```

```

    }

    if (i < pivotIndex && j > pivotIndex)
    {
        swap(arr[i++], arr[j--]);
    }
}

return pivotIndex;
}

void quickSort(char arr[], int start, int end)
{
    if (start >= end)
        return;

    int p = partition(arr, start, end);

    quickSort(arr, start, p - 1);
    quickSort(arr, p + 1, end);
}

void hitung(const char arr[], int n, int &countv, int &countk)
{
    countv = 0;
    countk = 0;

    for (int i = 0; i < n; i++)
    {
        if (arr[i] == 'a' || arr[i] == 'i' || arr[i] == 'u' ||
arr[i] == 'e' || arr[i] == 'o' ||
        arr[i] == 'A' || arr[i] == 'I' || arr[i] == 'U' ||
arr[i] == 'E' || arr[i] == 'O')
        {
            countv++;
        }
        else if (isalpha(arr[i]))
        {
            countk++;
        }
    }
}

```

```

int main()
{
    const int MAX_LENGTH = 100;
    char kalimat[MAX_LENGTH];
    char cleaned[MAX_LENGTH];
    int countv, countk;

    cout << "\n Masukkan sebuah kalimat : ";
    cin.getline(kalimat, MAX_LENGTH);

    // Remove spaces
    int n = 0;
    for (int i = 0; kalimat[i] != '\0'; i++)
    {
        if (kalimat[i] != ' ')
        {
            cleaned[n++] = kalimat[i];
        }
    }
    cleaned[n] = '\0';

    cout << "\n Data diurutkan : ";
    quickSort(cleaned, 0, n - 1);

    // Tampilkan data setelah diurutkan
    for (int i = 0; i < n; i++)
    {
        cout << cleaned[i];
    }
    cout << endl;

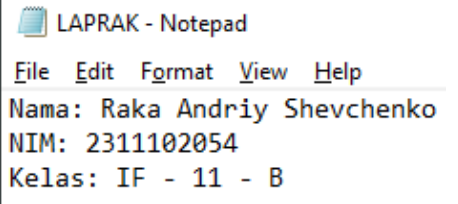
    hitung(cleaned, n, countv, countk);

    cout << "Jumlah Huruf Vokal : " << countv << endl;
    cout << "Jumlah Huruf Konsonan : " << countk << endl;
    return EXIT_SUCCESS;
}

```

Output:

```
PS C:\kuliah\Struktur Data\TUGAS 8> cd "c:\kuliah\Struktur Data\TUGAS 8\" : if ($?) {
Masukkan sebuah kalimat : Raka Andriy Shevchenko
Data diurutkan : ARSaacdeehhikknorvy
Jumlah Huruf Vokal : 7
Jumlah Huruf Konsonan : 13
```



Deskripsi:

Program C++ ini adalah kombinasi dari dua algoritma: Penyortiran Cepat dan sebuah fungsi untuk menghitung jumlah vokal dan konsonan dalam sebuah string. Program ini pertama-tama mengurutkan larik karakter menggunakan algoritme Quick Sort, kemudian menghitung jumlah vokal dan konsonan dalam larik yang telah diurutkan.

Berikut ini adalah rincian programnya:

Fungsi partisi adalah fungsi pembantu untuk algoritma Penyortiran Cepat. Fungsi ini mempartisi larik di sekitar elemen pivot, memastikan bahwa semua elemen yang lebih kecil dari pivot berada di sisi kiri dan semua elemen yang lebih besar dari pivot berada di sisi kanan.

Fungsi quickSort mengimplementasikan algoritma Quick Sort. Fungsi ini secara rekursif memanggil dirinya sendiri untuk mengurutkan larik, menggunakan fungsi partisi untuk membagi larik menjadi sub-larik yang lebih kecil.

Fungsi hitung menghitung jumlah huruf vokal dan konsonan dalam larik. Fungsi ini mengulang larik dan memeriksa setiap karakter. Jika karakter tersebut adalah vokal (baik huruf kecil atau huruf besar), maka fungsi ini akan menambah variabel hitungv. Jika karakter tersebut adalah konsonan (baik huruf kecil atau huruf besar), program akan menambah variabel countk.

Pada fungsi utama, program ini pertama-tama meminta pengguna untuk memasukkan sebuah kalimat. Kemudian menghapus semua spasi dari kalimat tersebut dan menyimpan kalimat yang telah dibersihkan dalam

sebuah array.

Program ini kemudian memanggil fungsi quickSort untuk mengurutkan kalimat yang telah dibersihkan, dan menampilkan kalimat yang telah diurutkan.

c. Unguided 3

Source Code:

```
#include <iostream>
#include <iomanip>
#include <cstring>

using namespace std;

int main()
{
    int n = 9;
    int data[n] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
    int a4 = 4;
    int j4 = 0;

    for (int i = 0; i <= n; i++)
    {
        if (data[i] == a4)
        {
            j4++;
        }
    }

    cout << "\tSequential Search." << endl;
    cout << "Data : [";
    for (int i = 0; i <= n; i++)
    {
        cout << data[i];
        if (i != n)
        {
            cout << ", ";
        }
    }
    cout << "]\n";
    cout << endl;
```



```

    cout << "Jumlah angka 4 : " << j4 << endl;

    return EXIT_SUCCESS;
}

```

Output:

```

PS C:\kuliah\Struktur Data\TUGAS 8> cd "c:\
Sequential Search.
Data : [9, 4, 1, 4, 7, 10, 5, 4, 12, 4]
Jumlah angka 4 : 4
PS C:\kuliah\Struktur Data\TUGAS 8> 

```

File Edit Format View Help
Nama: Raka Andriy Shevchenko
NIM: 2311102054
Kelas: IF - 11 - B

Deskripsi:

Program ini adalah implementasi sederhana dari algoritma Pencarian Berurutan. Program ini mencari nilai tertentu, a4, dalam larik data bilangan bulat. Larik berisi 10 elemen, dan program melakukan iterasi melalui setiap elemen untuk memeriksa apakah elemen tersebut cocok dengan nilai target. Jika ditemukan kecocokan, program akan menambah pencacah j4.

Berikut ini adalah rincian programnya:

Program mendefinisikan sebuah data larik dengan 10 elemen dan nilai target a4 yang disetel ke 4.

Program ini menggunakan perulangan for untuk mengulang setiap elemen dalam larik. Jika elemen saat ini sesuai dengan nilai target, maka program akan menambah pencacah j4.

Program kemudian mencetak larik dan hitungan nilai target.

Program ini menggunakan pustaka iostream untuk operasi input/output dan pustaka iomanip untuk memformat output.

D. Kesimpulan

a. Binary Search:

- Lebih efisien dari sisi waktu jika dibandingkan dengan Sequential Search.
- Hanya dapat digunakan pada array yang telah diurutkan.
- Memerlukan data yang telah diurutkan.

b. Sequential Search:

- Memerlukan memori yang relatif kecil.
- Dapat digunakan pada array yang tidak diurutkan.
- Dapat mengalami masalah optimal lokal.

E. Referensi

- [1] Trivusi, Algoritma Pencarian: Pengertian, Karakteristik, dan Jenis-Jenisnya.

Diakses dari

[Algoritma Pencarian: Pengertian, Karakteristik, dan Jenis-Jenisnya - Trivusi](#)

- [2] Ridwan Maulana, ALGORITMA PENCARIAN (SEQUENTIAL DAN BINARY SEARCH). Diakses dari

[ALGORITMA PENCARIAN \(SEQUENTIAL DAN BINARY SEARCH\)](#)
[\(researchgate.net\)](#)

- [3] Yoga Religia, ANALISIS ALGORITMA SEQUENTIAL SEARCH DAN BINARY SEARCH PADA BIG DATA. Diakses dari

[ANALISIS ALGORITMA SEQUENTIAL SEARCH DAN BINARY SEARCH](#)
[PADA BIG DATA | Pelita Teknologi \(pelitabangsa.ac.id\)](#)