

LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA

MODUL VIII ALGORITMA SEARCHING



Disusun Oleh :
NAUFAL THORIQ MUZHAFAR
2311102078

Dosen
WAHYU ANDI SAPUTRA, S.Pd., M.Eng

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

A. Dasar Teori

Pencarian (Searching) yaitu proses menemukan suatu nilai tertentu pada kumpulan data. Hasil pencarian adalah salah satu dari tiga keadaan ini: data ditemukan, data ditemukan lebih dari satu, atau data tidak ditemukan. Searching juga dapat dianggap sebagai proses pencarian suatu data di dalam sebuah array dengan cara mengecek satu persatu pada setiap index baris atau setiap index kolomnya dengan menggunakan teknik perulangan untuk melakukan pencarian data. Terdapat 2 metode pada algoritma Searching, yaitu:

Pencarian Linier

Ini adalah teknik pencarian paling dasar dan juga lebih mudah diterapkan. Dalam pencarian linier, kunci yang ingin dicari dibandingkan secara linier dengan setiap elemen kumpulan data. Teknik ini bekerja secara efektif pada struktur data linier.

32	12	6	23	54	10	28
0	1	2	3	4	5	6

Di atas adalah susunan tujuh elemen. Jika kita ingin mencari key = 23, maka mulai dari elemen ke-0 akan dibandingkan nilai kuncinya pada masing-masing elemen. Setelah elemen kunci cocok dengan elemen dalam array, lokasi tertentu tersebut akan dikembalikan. Dalam hal ini lokasi, 4 akan dikembalikan karena nilai kunci cocok dengan nilai di lokasi tersebut.

Implementasi C++

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    int myarray[10] = {21,43,23,54,75,13,5,8,25,10};
    int key,loc;
    cout<<"The input array is"<<endl;
    for(int i=0;i<10;i++){
        cout<<myarray[i]<<" ";
```

```

    }
    cout<<endl;
    cout<<"Enter the key to be searched : "; cin>>key;
    for (int i = 0; i< 10; i++)
    {
        if(myarray[i] == key)
        {
            loc = i+1;
            break;
        }
        else
            loc = 0;
    }
    if(loc != 0)
    {
        cout<<"Key found at position "<<loc<<" in the array";
    }
    else
    {
        cout<<"Could not find given key in the array";
    }
}

```

Output

```

The input array is
21 43 23 54 75 13 5 8 25 10
Enter the key to be searched : 4
Could not find given key in the array

```

```

The input array is
21 43 23 54 75 13 5 8 25 10
Enter the key to be searched : 21
Key found at position 1 in the array

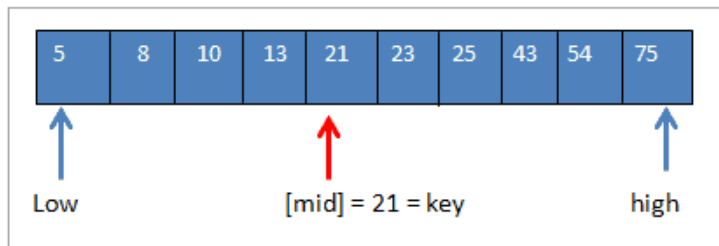
```

Pencarian Biner

Sebagai contoh, mari kita ambil array 10 elemen yang diurutkan berikut ini.

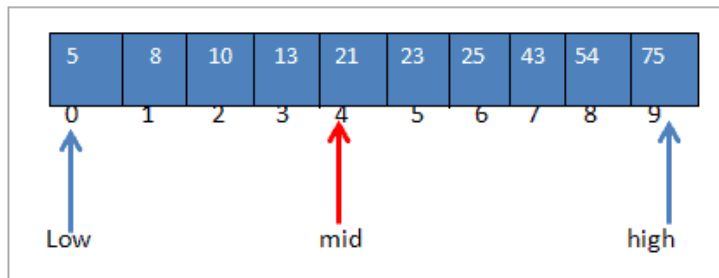
Sebagai contoh, mari kita ambil array 10 elemen yang diurutkan berikut ini.

Pertama, kita akan membandingkan nilai key dengan elemen [mid]. Kami menemukan bahwa nilai elemen di $\text{mid} = 21$.

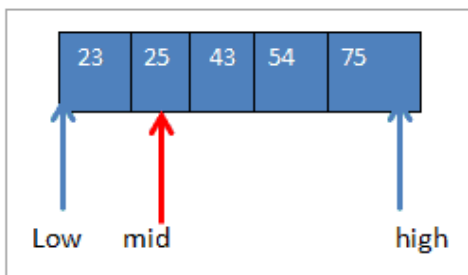


Jadi kita menemukan key itu = [mid]. Oleh karena itu kuncinya ditemukan.

Key = 25



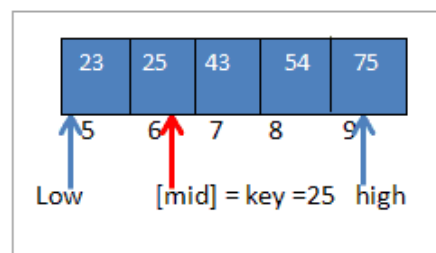
Pertama-tama kita bandingkan nilai key dengan nilai mid. Jadi ($21 < 25$), kita akan langsung mencari key di bagian atas array.



Sekarang lagi kita akan menemukan titik mid untuk paruh atas array.

$$\text{Mid} = 5 + 9 / 2 = 7$$

Nilai di lokasi [mid] = 25



Sekarang kita bandingkan elemen key dengan elemen mid. Jadi ($25 == 25$), maka kita sudah menemukan key di lokasi [mid].

Kami berulang kali membagi array dan dengan membandingkan elemen key dengan bagian mid, kami memutuskan di bagian mana untuk mencari key.

Implementasi C++

```
#include <iostream>
#include <string>
using namespace std;
int binarySearch(int myarray[], int beg, int end, int key)
{
    int mid;
    if(end >= beg) {
        mid = (beg + end)/2;
        if(myarray[mid] == key)
        {
            return mid+1;
        }
        else if(myarray[mid] < key) {
            return binarySearch(myarray,mid+1,end,key);
        }
        else {
            return binarySearch(myarray,beg,mid-1,key);
        }
    }
    return -1;
}
int main ()
{
    int myarray[10] = {5,8,10,13,21,23,25,43,54,75};
    int key, location=-1;
    cout<<"The input array is"<<endl;
    for(int i=0;i<10;i++){
        cout<<myarray[i]<<" ";
    }
    cout<<endl;
    cout<<"Enter the key that is to be searched:"; cin>>key;
    location = binarySearch(myarray, 0, 9, key);
    if(location != -1) {
```

```
        cout<<"Key found at location "<<location;
    }
    else    {
        cout<<"Requested key not found";
    }
}
```

Output

```
The input array is
5 8 10 13 21 23 25 43 54 75
Enter the key that is to be searched:21
Key found at location 5
```

Pencarian biner lebih efisien dari segi waktu dan kebenaran. Teknik pencarian linier jarang digunakan karena lebih rumit dan lambat. Pencarian biner jauh lebih cepat jika dibandingkan dengan pencarian linier.

Kesimpulan

Teknik pencarian membantu kita mencari informasi yang disimpan di komputer sehingga pengguna dapat melanjutkan tugas pemrosesan informasi lainnya. Teknik pencarian linier sederhana dan mudah tetapi tidak digunakan secara luas.

Teknik pencarian biner jauh lebih cepat dan efisien sehingga digunakan secara luas.

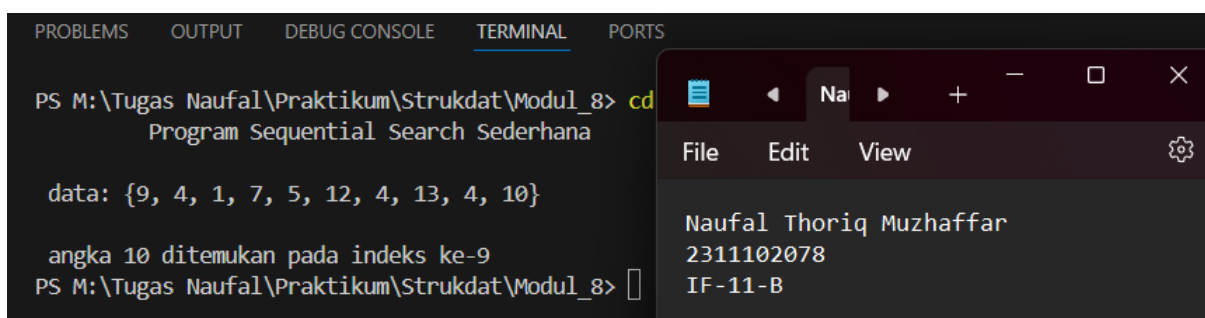
B. Guided

Guided 1 (Sequential Search)

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;
}
```

Sreenshots Output



The screenshot shows a terminal window with the following output:

```
PS M:\Tugas Naufal\Praktikum\Strukdat\Modul_8> cd
Program Sequential Search Sederhana

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

angka 10 ditemukan pada indeks ke-9
PS M:\Tugas Naufal\Praktikum\Strukdat\Modul_8>
```

The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active. The output shows the program's execution, including the data array and the search result for the number 10 at index 9.

Deskripsi Program

Program ini mendefinisikan sebuah array berisi 10 elemen dan mencari nilai tertentu (10) dalam array tersebut. Dengan menggunakan loop for, program memeriksa setiap elemen dalam array dan menentukan apakah nilai yang dicari ada di dalam array. Jika ditemukan, program mencetak indeks di mana nilai tersebut berada; jika tidak, program mencetak pesan bahwa nilai tidak ditemukan.

Guided 2 (Binary Sort)

Source Code

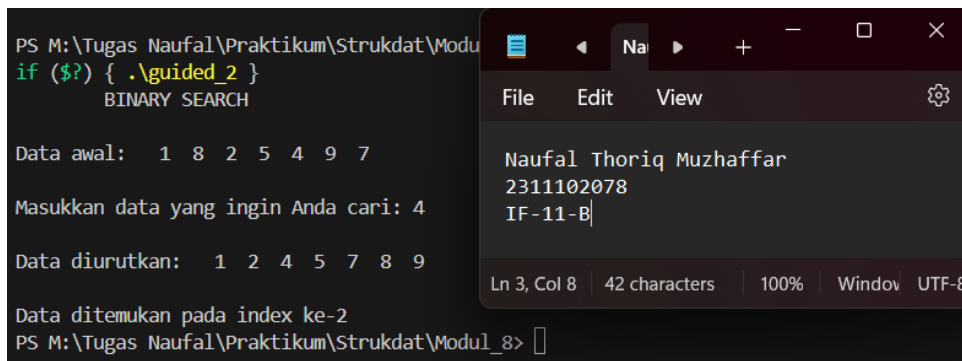
```
#include <iostream>
#include <iomanip>
using namespace std;
// Deklarasi array dan variabel untuk pencarian
int arrayData[7] = {1, 8, 2, 5, 4, 9, 7};
int cari;
void selection_sort(int arr[], int n)
{
    int temp, min;
    for (int i = 0; i < n - 1; i++)
    {
        min = i;
        for (int j = i + 1; j < n; j++)
        {
            if (arr[j] < arr[min])
            {
                min = j;
            }
        }
        // Tukar elemen
        temp = arr[i];
        arr[i] = arr[min];
        arr[min] = temp;
    }
}
void binary_search(int arr[], int n, int target)
{
    int awal = 0, akhir = n - 1, tengah, b_flag = 0;
    while (b_flag == 0 && awal <= akhir)
    {
        tengah = (awal + akhir) / 2;
        if (arr[tengah] == target)
        {
            b_flag = 1;
        }
    }
}
```

```

        break;
    }
    else if (arr[tengah] < target)
    {
        awal = tengah + 1;
    }
    else
    {
        akhir = tengah - 1;
    }
}
if (b_flag == 1)
    cout << "\nData ditemukan pada index ke-" << tengah << endl;
else
    cout << "\nData tidak ditemukan\n";
}
int main()
{
    cout << "\tBINARY SEARCH" << endl;
    cout << "\nData awal: ";
    // Tampilkan data awal
    for (int x = 0; x < 7; x++)
    {
        cout << setw(3) << arrayData[x];
    }
    cout << endl;
    cout << "\nMasukkan data yang ingin Anda cari: ";
    cin >> cari;
    // Urutkan data dengan selection sort
    selection_sort(arrayData, 7);
    cout << "\nData diurutkan: ";
    // Tampilkan data setelah diurutkan
    for (int x = 0; x < 7; x++)
    {
        cout << setw(3) << arrayData[x];
    }
    cout << endl;
    // Lakukan binary search
    binary_search(arrayData, 7, cari);
    return 0;
}

```

Screenshots Output



```
PS M:\Tugas Naufal\Praktikum\Strukdat\Modul_8> if ($?) { .\guided_2 }  
BINARY SEARCH  
  
Data awal: 1 8 2 5 4 9 7  
Masukkan data yang ingin Anda cari: 4  
  
Data diurutkan: 1 2 4 5 7 8 9  
  
Data ditemukan pada index ke-2  
PS M:\Tugas Naufal\Praktikum\Strukdat\Modul_8>
```

Deskripsi Program

Program digunakan untuk mencari sebuah angka dalam array dengan menggunakan metode pengurutan Selection Sort diikuti dengan pencarian Binary Search. Pertama, program mendeklarasikan sebuah array `arrayData` yang berisi tujuh elemen dan sebuah variabel `cari` untuk menyimpan angka yang akan dicari. Setelah pengguna memasukkan angka yang ingin dicari, program mengurutkan array menggunakan fungsi `selection_sort`. Setelah array terurut, program menampilkan array yang telah diurutkan dan kemudian menggunakan fungsi `binary_search` untuk mencari angka yang dimasukkan oleh pengguna. Jika angka ditemukan, program menampilkan indeks tempat angka tersebut ditemukan; jika tidak, program menampilkan pesan bahwa angka tidak ditemukan. Fungsi `selection_sort` bekerja dengan menemukan elemen terkecil dan menukarnya dengan elemen pertama, kedua, dan seterusnya, sedangkan fungsi `binary_search` mencari angka dengan membagi array terurut menjadi dua bagian secara iteratif sampai angka ditemukan atau seluruh array sudah diperiksa.

C. Unguided

Unguided 1

Source Code

```
#include <iostream>  
#include <conio.h>  
#include <iomanip>  
  
using namespace std;  
  
string sentence;  
char c;  
  
void toLower() {  
    string temp;
```

```

        for (int i = 0; i < sentence.length(); i++) {
            temp += tolower(sentence[i]);
        }
        sentence = temp;
    }

void selection_sort()
{
    int min, i, j;
    char temp;
    toLower();
    for (i = 0; i < sentence.length(); i++)
    {
        min = i;
        for (j = i + 1; j < sentence.length(); j++)
        {
            if (sentence[j] < sentence[min])
            {
                min = j;
            }
        }
        temp = sentence[i];
        sentence[i] = sentence[min];
        sentence[min] = temp;
    }
}

void binarysearch()
{
    int awal, akhir, tengah, b_flag = 0;
    awal = 0;
    akhir = sentence.length();
    while (b_flag == 0 && awal <= akhir)
    {
        tengah = (awal + akhir) / 2;
        if (sentence[tengah] == c)
        {
            b_flag = 1;
            break;
        }
        else if (sentence[tengah] < c)
            awal = tengah + 1;
        else
            akhir = tengah - 1;
    }
    if (b_flag == 1)
        cout << "\n Karakter '" << c << "' ditemukan pada index ke - " <<
tengah << endl;
    else

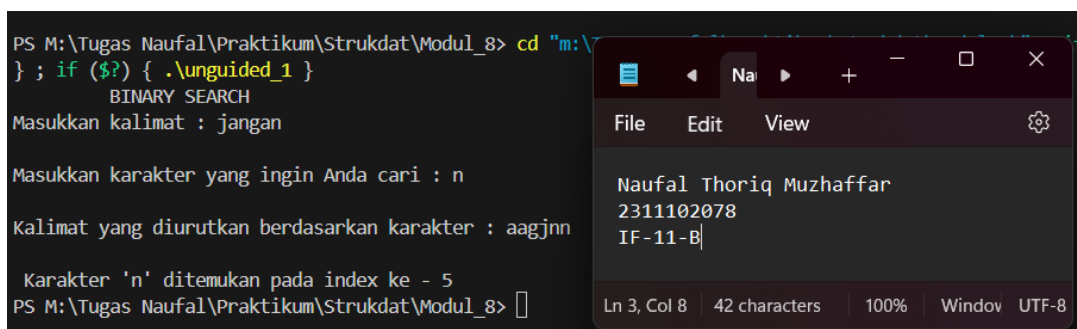
```

```

        cout << "\nData tidak ditemukan\n";
    }
int main()
{
    cout << "\t BINARY SEARCH " << endl;
    cout << "Masukkan kalimat : ";
    getline(cin, sentence);
    cout << "\nMasukkan karakter yang ingin Anda cari : ";
    cin >> c;
    c = tolower(c);
    cout << "\nKalimat yang diurutkan berdasarkan karakter : ";
    selection_sort();
    for (int x = 0; x < sentence.length(); x++)
        cout << sentence[x];
    cout << endl;
    binarysearch();
    return EXIT_SUCCESS;
}

```

Screenshots Output



The screenshot shows a Windows command prompt window with the following text:

```

PS M:\Tugas Naufal\Praktikum\Strukdat\Modul_8> cd "m:\
}; if ($?) { .\unguided_1 }
        BINARY SEARCH
Masukkan kalimat : jangan
Masukkan karakter yang ingin Anda cari : n
Kalimat yang diurutkan berdasarkan karakter : aagjnn
Karakter 'n' ditemukan pada index ke - 5
PS M:\Tugas Naufal\Praktikum\Strukdat\Modul_8>

```

Overlaid on the command prompt is a Notepad++ window titled 'Naufal Thoriq Muzhaffar'. The text in the Notepad++ window is:

```

Naufal Thoriq Muzhaffar
2311102078
IF-11-B

```

The Notepad++ status bar at the bottom indicates 'Ln 3, Col 8 | 42 characters | 100% | Window UTF-8'.

Deskripsi Program

Program digunakan untuk mencari sebuah karakter dalam sebuah kalimat menggunakan metode pengurutan Selection Sort diikuti dengan pencarian Binary Search. Pertama, pengguna memasukkan kalimat dan karakter yang ingin dicari. Program mengubah semua karakter dalam kalimat menjadi huruf kecil menggunakan fungsi `toLowerCase` untuk memastikan pencarian tidak bersifat case-sensitive. Kemudian, kalimat diurutkan berdasarkan karakter menggunakan fungsi `selection_sort`, yang menemukan dan menukar elemen terkecil ke posisi yang sesuai. Setelah itu, program mencari karakter yang dimasukkan pengguna dalam kalimat yang telah diurutkan menggunakan fungsi `binarysearch`. Jika karakter ditemukan, program menampilkan indeks tempat karakter tersebut ditemukan; jika tidak, program menampilkan pesan bahwa data tidak ditemukan. Hasil dari proses pengurutan dan pencarian ini ditampilkan di layar.

Unguided 2

Source Code

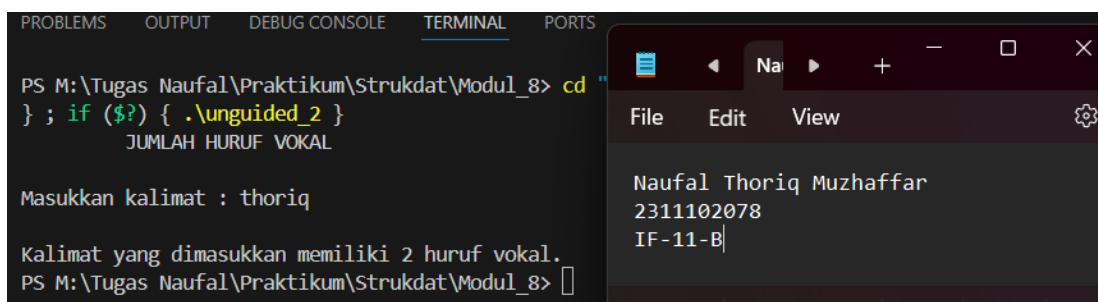
```
#include <iostream>
using namespace std;
int main()
{
    string s;
    int count = 0;

    cout << "\t JUMLAH HURUF VOKAL" << endl;
    cout << "\nMasukkan kalimat : ";
    getline(cin, s);

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

    cout << "\nKalimat yang dimasukkan memiliki " << count << " huruf vokal.";
    return 0;
}
```

Screenshots Output



Deskripsi Program

Program ini bertujuan untuk menghitung jumlah huruf vokal dalam sebuah kalimat yang dimasukkan oleh pengguna. Program dimulai dengan mendeklarasikan sebuah string `s` dan variabel `count` untuk menyimpan jumlah huruf vokal. Pengguna diminta untuk memasukkan sebuah kalimat, yang kemudian dibaca menggunakan `getline(cin, s)`. Dalam sebuah loop `for`, setiap karakter dalam kalimat diubah menjadi huruf kecil menggunakan `tolower`, kemudian diperiksa apakah karakter tersebut adalah salah satu dari huruf vokal ('a', 'i', 'u', 'e', 'o'). Jika ya,

count akan ditambah satu. Setelah loop selesai, program mencetak jumlah huruf vokal yang ditemukan dalam kalimat tersebut.

Unguided 3

Source Code

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

int arr[] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
int arrLength = sizeof(arr)/sizeof(arr[0]);
int cari;

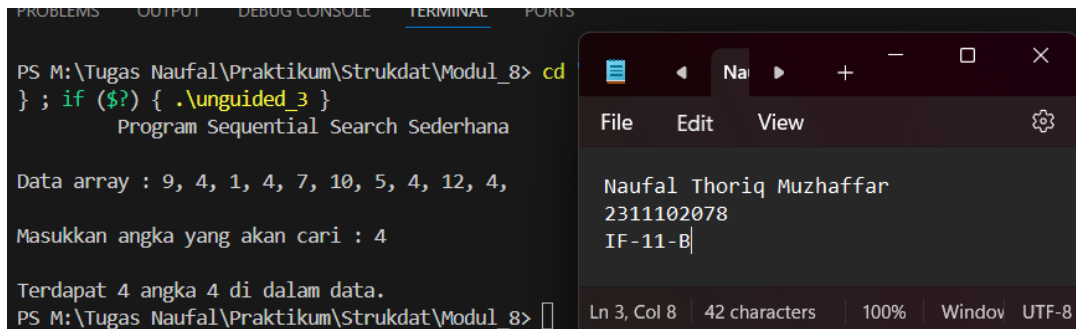
int seqSearch() {
    int count = 0;
    for (int i = 0; i < arrLength; i++)
    {
        if (arr[i] == cari)
        {
            count++;
        }
    }
    return count;
}

int main()
{
    // algoritma Sequential Search

    cout << "\t Program Sequential Search Sederhana\n" << endl;
    cout << "Data array : ";
    for (int x : arr) {
        cout << x << ", ";
    }
    cout << "\n\nMasukkan angka yang akan cari : ";
    cin >> cari;

    cout << "\nTerdapat " << seqSearch() << " angka " << cari << " di dalam data.";
    return 0;
}
```

Screenshots Output



```
PS M:\Tugas Naufal\Praktikum\Strukdat\Modul_8> cd
} ; if ($?) { .\unguided_3 }
Program Sequential Search Sederhana

Data array : 9, 4, 1, 4, 7, 10, 5, 4, 12, 4,

Masukkan angka yang akan cari : 4

Terdapat 4 angka 4 di dalam data.
PS M:\Tugas Naufal\Praktikum\Strukdat\Modul_8>
```

Deskripsi Program

Program ini menerima sebuah array yang sudah didefinisikan sebelumnya, kemudian meminta pengguna untuk memasukkan angka yang ingin dicari. Setelah itu, program akan mencari keberadaan angka tersebut dalam array menggunakan algoritma sequential search. Setiap kali angka yang dicari ditemukan dalam array, variabel count akan bertambah satu. Akhirnya, program akan mencetak jumlah kemunculan angka yang dicari dalam array. Ini adalah contoh penggunaan sederhana dari algoritma pencarian sekuensial untuk mencari nilai tertentu dalam sebuah array.

D. Kesimpulan

Algoritma searching merupakan serangkaian langkah logis yang digunakan untuk menemukan suatu nilai atau elemen tertentu dalam sebuah kumpulan data. Terdapat berbagai jenis algoritma searching yang dapat digunakan, seperti sequential search dan binary search. Sequential search melakukan pencarian satu per satu dari awal hingga akhir kumpulan data, sementara binary search membagi data menjadi dua bagian dan mencari di salah satu bagian berdasarkan perbandingan dengan nilai tengah. Pemilihan algoritma searching yang tepat akan sangat memengaruhi kinerja dan efisiensi pencarian data. Dengan pemahaman yang baik tentang karakteristik dan keunggulan masing-masing algoritma searching, pengembang dapat memilih algoritma yang sesuai untuk aplikasi mereka, meningkatkan efisiensi, dan mengoptimalkan kinerja pencarian data.

E. Referensi

- [1] Asprak “Modul 8 Queue”. Learning Management System 2024.
- [2] Softwaretestinghelp – Searching Algorithms in cpp. Diakses pada 2 Juni 2024
<https://www.softwaretestinghelp.com/searching-algorithms-in-cpp/>