

# **LAPORAN PRAKTIKUM**

## **MODUL VI ALGORITMA SEARCHING**



**Disusun oleh:  
Didik Setiawan  
NIM: 2311102030**

**Dosen Pengampu:**  
Wahyu Andi Saputra, S.Pd., M.Eng.

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
PURWOKERTO  
2023**

## **BAB II**

### **TUJUAN PRAKTIKUM**

1. Mampu Menunjukkan beberapa algoritma dalam Pencarian.
2. Menunjukkan bahwa pencarian merupakan suatu persoalan yang bisa diselesaikan dengan beberapa algoritma yang berbeda.
3. cDapat memilih algoritma yang paling sesuai untuk menyelesaikan suatu permasalahan pemrograman.

## **BAB II**

### **DASAR TEORI**

#### **A. Pengertian Searching**

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.

#### **B. Metode Searching**

Terdapat 2 metode pada algoritma Searching

- **Sequential Search**
  - Membandingkan setiap elemen pada array satu per satu secara berurut.
  - Proses pencarian dimulai dari indeks pertama hingga indeks terakhir.
  - Proses pencarian akan berhenti apabila data ditemukan. Jika hingga akhir array data masih juga tidak ditemukan, maka proses pencarian tetap akan dihentikan.
  - Proses perulangan pada pencarian akan terjadi sebanyak jumlah N elemen pada array.
- **Binary Search**
  - Data diambil dari posisi 1 sampai posisi akhir N.
  - Kemudian data akan dibagi menjadi dua untuk mendapatkan posisi data tengah.
  - Selanjutnya data yang dicari akan dibandingkan dengan data yang berada di posisi tengah, apakah lebih besar atau lebih kecil.
  - Apabila data yang dicari lebih besar dari data tengah, maka dapat dipastikan bahwa data yang dicari kemungkinan berada di sebelah kanan dari data tengah. Proses pencarian selanjutnya akan dilakukan pembagian data menjadi dua bagian pada bagian kanan dengan acuan posisi data tengah akan menjadi posisi awal untuk pembagian tersebut.
  - Apabila data yang dicari lebih kecil dari data tengah, maka dapat dipastikan bahwa data yang dicari kemungkinan berada di sebelah kiri dari data tengah. Proses pencarian selanjutnya akan dilakukan pembagian data menjadi dua bagian pada

bagian kiri. Dengan acuan posisi data tengah akan menjadi posisi akhir untuk pembagian selanjutnya.

- Apabila data belum ditemukan, maka pencarian akan dilanjutkan dengan kembali membagi data menjadi dua.
- Namun apabila data bernilai sama, maka data yang dicari langsung ditemukan dan pencarian dihentikan

## BAB III

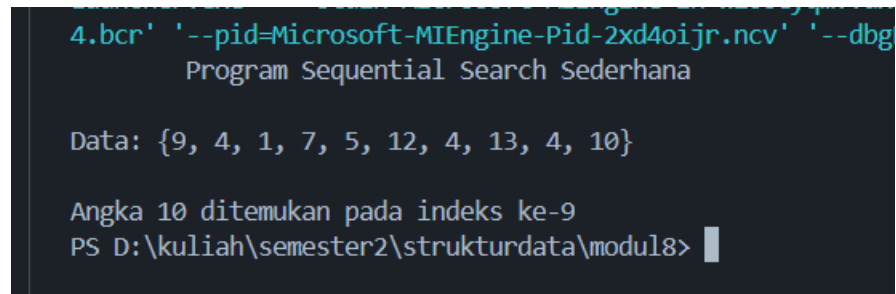
### GUIDED

#### 1. Guided 1

```
2. #include <iostream>
3. using namespace std;
4.
5. int main() {
6.     int n = 10;
7.     int data[n] = {9, 4, 1, 7, 5, 12, 4, 13, 4, 10};
8.     int cari = 10;
9.     bool ketemu = false;
10.    int i;
11.
12.    // Algoritma Sequential Search
13.    for (i = 0; i < n; i++) {
14.        if(data[i] == cari) {
15.            ketemu = true;
16.            break;
17.        }
18.    }
19.
20.    cout << "\tProgram Sequential Search
    Sederhana\n" << endl;
21.    cout << "Data: {9, 4, 1, 7, 5, 12, 4, 13, 4, 10}"
    << endl;
22.
23.    if (ketemu) {
24.        cout << "\nAngka " << cari << " ditemukan
    pada indeks ke-" << i << endl;
25.    } else {
26.        cout << cari << " tidak dapat ditemukan pada
    data." << endl;
27.    }
```

```
28.  
29.         return 0;  
30.     }
```

### Screenshoot program



```
4.bcr' '--pid=Microsoft-MIEngine-Pid-2xd4oijr.ncv' '--dbg  
Program Sequential Search Sederhana  
  
Data: {9, 4, 1, 7, 5, 12, 4, 13, 4, 10}  
  
Angka 10 ditemukan pada indeks ke-9  
PS D:\kuliah\semester2\strukturdata\modul8> |
```

### Deskripsi program

Algoritma Sequential Search dipakai, yang berarti program ini memeriksa setiap angka dalam daftar secara berurutan, dimulai dari awal, sampai menemukan angka 10 atau mencapai akhir daftar. Jika angka 10 ditemukan, program akan menampilkan lokasinya di daftar. Jika tidak ditemukan, program akan menyatakan bahwa angka 10 tidak ada dalam daftar.

### guided 2

#### 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;
}

```

**Screenshoot program**



```
BINARY SEARCH

Data awal:  1  8  2  5  4  9  7

Masukkan data yang ingin Anda cari: 7

Data diurutkan:  1  2  4  5  7  8  9

Data ditemukan pada index ke-4
PS D:\kuliah\semester2\strukturdata\modul8> |
```

### Deskripsi program

Algoritma ini membagi array menjadi dua bagian secara berulang untuk mempersempit pencarian.

Data diurutkan dulu dengan Selection Sort agar Binary Search bisa bekerja.

Binary Search kemudian membagi array menjadi dua bagian. Jika data target ada di bagian pertama, pencarian dilanjutkan di sana. Jika tidak, pencarian dilanjutkan di bagian kedua.

Proses ini terus diulang hingga data target ditemukan atau seluruh array diperiksa.

Jika data target ditemukan, lokasinya ditampilkan. Jika tidak, program menyatakan bahwa data target tidak ada.

## 1. Unguided 1

### Source code

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

using namespace std;

char inputString[100];
char targetChar;

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

```
        arr[min] = temp;
    }
}

void searchData(char arr[], int n)
{
    int start, end, mid;
    bool found = false;
    start = 0;
    end = n - 1;
    while (!found && start <= end)
    {
        mid = (start + end) / 2;
        if (arr[mid] == targetChar)
        {
            found = true;
        }
        else if (arr[mid] < targetChar)
        {
            start = mid + 1;
        }
        else
        {
            end = mid - 1;
        }
    }
    if (found)
    {
        cout << "\nData ditemukan pada index ke-" << mid << endl;
    }
    else
    {
        cout << "\nData tidak ditemukan" << endl;
    }
}
```

```
int main()
{
    cout << "===== Searching Huruf =====" << endl;

    cout << "Masukkan kalimat: ";
    cin.getline(inputString, 100);

    int length = strlen(inputString);

    cout << "Masukkan karakter yang ingin dicari: ";
    cin >> targetChar;

    cout << "\nData diurutkan: ";
    sortData(inputString, length);

    for (int x = 0; x < length; x++)
    {
        cout << setw(3) << inputString[x];
    }
    cout << endl;

    searchData(inputString, length);

    _getche();
    return 0;
}
```

**Screenshoot program**

```
3.313 C:\Program Files\Microsoft Visual Studio\2019\Community>cd ..
```

```
===== Searching Huruf =====  
Masukkan kalimat: halo  
Masukkan karakter yang ingin dicari: a  
  
Data diurutkan:  a h l o  
  
Data ditemukan pada index ke-0
```

### Deskripsi program

meminta user untuk memasukkan kalimat dan karakter yang ingin dicari, lalu mengurutkan kalimat tersebut menggunakan Selection Sort dan mencari karakter yang dicari menggunakan Binary Search. Hasilnya, program akan menampilkan kalimat yang telah diurutkan dan menunjukkan apakah karakter yang dicari ditemukan atau tidak.

## 2. Unguided 2

### Source code

```
#include <iostream>  
#include <string>  
  
using namespace std;  
  
int countVocal(string inputText, char targetChar) // fungsi untuk  
mencari jumlah huruf vokal (sequential search)  
{  
    int count = 0;  
    for (int i = 0; i < inputText.length(); i++) // Sequential  
search dilakukan dengan iterasi melalui setiap karakter dalam  
string  
    {
```

```

        if (inputText[i] == targetChar) // Memeriksa apakah
karakter pada indeks saat ini sama dengan huruf yang dicari
        {
            count++; // Jika ditemukan huruf yang sesuai,
jumlahnya ditambah
        }
    }
    return count; // Mengembalikan jumlah huruf yang ditemukan
}

void displayVocalCount(string inputText) // fungsi untuk
menampilkan jumlah huruf vokal
{
    int count;
    char vocalCharacters[10] = {'a', 'i', 'u', 'e', 'o', 'A',
'I', 'U', 'E', 'O'};
    for (int i = 0; i < 10; i++)
    {
        count = countVocal(inputText, vocalCharacters[i]);
        cout << "Jumlah huruf " << vocalCharacters[i] << " : "
<< count << endl;
    }
}

int main() // fungsi utama
{
    cout << "===== Menghitung Jumlah Huruf Vokal ====="
<< endl;
    string inputText;
    cout << "Masukkan kata : ";
    cin >> inputText;
    displayVocalCount(inputText);
    return 0;
}

```

**Screenshoot program**

```

===== Menghitung Jumlah Huruf Vokal =====
Masukkan kata : makanenak
Jumlah huruf a : 3
Jumlah huruf i : 0
Jumlah huruf u : 0
Jumlah huruf e : 1
Jumlah huruf o : 0
Jumlah huruf A : 0
Jumlah huruf I : 0
Jumlah huruf U : 0
Jumlah huruf E : 0
Jumlah huruf O : 0

```

### Deskripsi program

menggunakan fungsi countVocal untuk melakukan pencarian huruf vokal secara sequential dalam string input, dan fungsi displayVocalCount untuk menampilkan jumlah huruf vokal yang ditemukan untuk setiap huruf vokal (a, i, u, e, o, A, I, U, E, O).

### 3. Unguided 3

#### Source code

```

#include <iostream>

using namespace std;

int sequentialSearch(int inputArray[], int arraySize, int
targetNumber) // Fungsi untuk mencari jumlah kemunculan suatu
angka dalam array
{
    int count = 0;
    for (int i = 0; i < arraySize; ++i)
    {
        if (inputArray[i] == targetNumber)
        {
            count++;
        }
    }
}

```

```

        return count;
    }

    int main() // fungsi utama
    {
        cout << "===== Searching Data Angka 4 =====" <<
endl;
        int inputArray[] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4}; // Data
yang akan dicari
        int arraySize = sizeof(inputArray) / sizeof(inputArray[0]);
// Menghitung jumlah elemen dalam array
        int targetNumber = 4; // Angka yang dicari
        int count = sequentialSearch(inputArray, arraySize,
targetNumber); // Memanggil fungsi sequentialSearch
        cout << "Jumlah angka 4: " << count << endl;
        return 0;
    }

```

### Screenshoot program

```

===== Searching Data Angka 4 =====
Jumlah angka 4: 4
PS D:\kuliah\semester2\strukturdata\modul8>

```

### Deskripsi program

fungsi `sequentialSearch` yang menerima tiga parameter, yaitu array, panjang array, dan angka yang dicari. Fungsi ini melakukan iterasi melalui setiap elemen dalam array dan menambah jumlah kemunculan angka yang dicari jika ditemukan.



## KESIMPULAN

Kode pertama adalah program C++ yang mengimplementasikan algoritma Selection Sort untuk mengurutkan data berupa kalimat yang diinput oleh user, dan kemudian menggunakan algoritma Binary Search untuk mencari karakter tertentu dalam kalimat yang telah diurutkan. Sedangkan kode kedua adalah program C++ yang mengimplementasikan algoritma Sequential Search untuk mencari jumlah kemunculan suatu angka dalam array. Kode tersebut menggunakan algoritma pencarian yang berbeda, yaitu Selection Sort dan Binary Search untuk kode pertama, dan Sequential Search untuk kode kedua. Kode tersebut juga menggunakan array untuk menyimpan data yang akan dicari, dan menggunakan fungsi untuk mengimplementasikan algoritma pencarian. Kode pertama bertujuan untuk mencari karakter tertentu dalam kalimat yang telah diurutkan, sedangkan kode kedua bertujuan untuk mencari jumlah kemunculan suatu angka dalam array. Kode tersebut dapat digunakan untuk mengurutkan data dan mencari data tertentu dalam data yang telah diurutkan, sehingga dapat membantu dalam pengolahan data.

## DAFTAR PUSTAKA

Asisten Praktikum. (2024). MODUL 8 Searching. Learning Managament

**Badri, I. (2019). Pencarian Searching C Disertai Contoh.** <https://pintarkom.com/searching-pada-c-plus/>. Diakses pada 5 Juni 2024.