LAPORAN PRAKTIKUM

MODUL 8
SEARCHING

Disusun oleh:
Bayu Kuncoro Adi
NIM: 2311102031

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

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

BAB I TUJUAN
PRAKTIKUM

A. TUJUAN PRAKTIKUM

1. Mahasiswa mampu menunjukkan beberapa algoritma dan searching.
2. Menunjukkan bahwa pencarian merupakan suatu persoalan yang bisa diselesaikan dengan beberapa algoritma yang berbeda.
3. Dapat memilih algoritma yang paling sesuai untuk menyelesaikan suatu permasalahan pemrograman.

BAB II DASAR

TEORI

A. DASAR TEORI

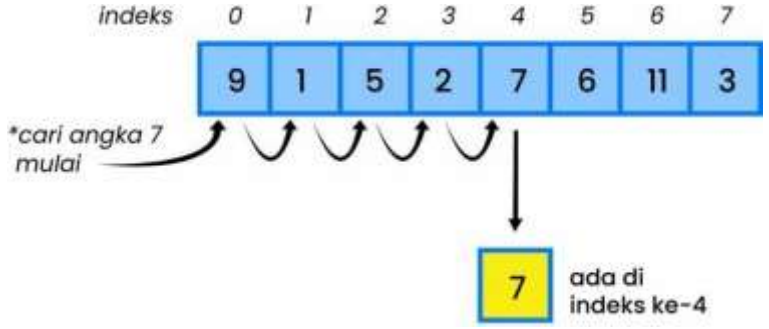
1. Pengertian Searching

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. 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:

a. Sequential Search
   Sequential Search merupakan salah satu algoritma pencarian data yang biasa digunakan untuk data yang berpola acak atau belum terurut. Sequential search juga merupakan teknik pencarian data dari array yang paling mudah, dimana data dalam array dibaca satu demi satu dan diurutkan dari index terkecil ke index terbesar, maupun sebaliknya. Konsep Sequential Search yaitu:
   - 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

   Algoritma pencarian berurutan dapat dituliskan sebagai berikut :
   1) i ← 0
   2) ketemu ← false
   3) Selama (tidak ketemu) dan (i <= N) kerjakan baris 4
   4) Jika (Data[i] = x) maka ketemu ← true, jika tidak i ← i + 1
   5) Jika (ketemu) maka i adalah indeks dari data yang dicari, jika tidak data tidak ditemukan.
   Di bawah ini merupakan fungsi untuk mencari data menggunakan pencarian sekuensial.

Contoh dari Sequential Search, yaitu: Int A[8] = {9,1,5,2,7,6,11,3}



Gambar 1. Ilustrasi Sequential Search

Misalkan, dari data di atas angka yang akan dicari adalah angka 7 dalam array A, maka proses yang akan terjadi yaitu:

- Pencarian dimulai pada index ke-0 yaitu angka 9, kemudian dicocokkan dengan angka yang akan dicari, jika tidak sama maka pencarian akan dilanjutkan ke index selanjutnya.
- Pada index ke-1, yaitu angka 1, juga bukan angka yang dicari, maka pencarian akan dilanjutkan pada index selanjutnya.
- Pada index ke-2 dan index ke-3 yaitu angka 5 dan 2, juga bukan angka yang dicari, sehingga pencarian dilanjutkan pada index selanjutnya.
- Pada index ke-4 yaitu angka 7 dan ternyata angka 7 merupakan angka yang dicari, sehingga pencarian akan dihentikan dan proses selesai

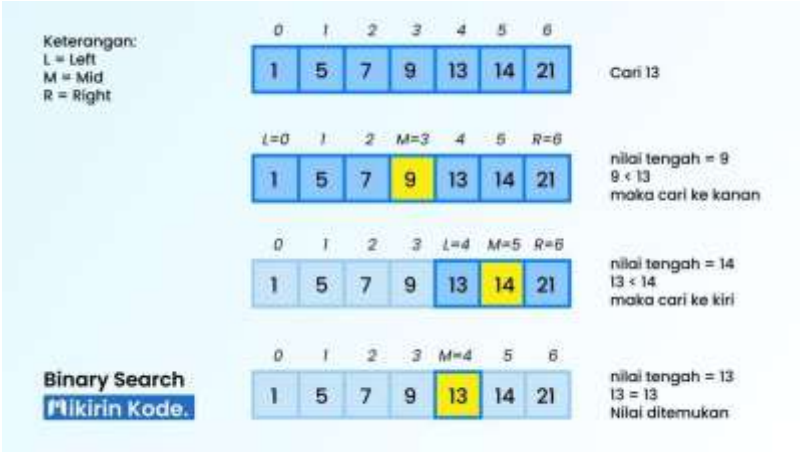
b. Binary Search

Binary Search termasuk ke dalam interval search, dimana algoritma ini merupakan algoritma pencarian pada array/list dengan elemen terurut. Pada metode ini, data harus diurutkan terlebih dahulu dengan cara data dibagi menjadi dua bagian (secara logika), untuk setiap tahap pencarian. Dalam penerapannya algoritma ini sering digabungkan dengan algoritma sorting karena data yang akan digunakan harus sudah terurut terlebih dahulu. Konsep 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 diposisi 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.

Algoritma pencarian biner dapat dituliskan sebagai berikut :
1) L 0
2) R N – 1
3) ketemu false
4) Selama (L <= R) dan (tidak ketemu) kerjakan baris 5 sampai dengan 8
5) m (L + R) / 2
6) Jika (Data[m] = x) maka ketemu true
7) Jika (x < Data[m]) maka R m – 1
8) Jika (x > Data[m]) maka L m + 1
9) Jika (ketemu) maka m adalah indeks dari data yang dicari, jika tidak data tidak ditemukan.

Contoh dari Binary Search, yaitu:



Gambar 2. Ilustrasi Binary Search

- Terdapat sebuah array yang menampung 7 elemen seperti ilustrasi di atas. Nilai yang akan dicari pada array tersebut adalah 13.
- Jadi karena konsep dari binary search ini adalah membagi array menjadi dua bagian, maka pertama tama kita cari nilai tengahnya dulu, total elemen dibagi 2 yaitu 7/2 = 4.5 dan kita bulatkan jadi 4.
- Maka elemen ke empat pada array adalah nilai tengahnya, yaitu angka 9 pada indeks ke 3.
- Kemudian kita cek apakah 13 > 9 atau 13 < 9?
- 13 lebih besar dari 9, maka kemungkinan besar angka 13 berada setelah 9 atau di sebelah kanan. Selanjutnya kita cari ke kanan dan kita dapat mengabaikan elemen yang ada di kiri.
- Setelah itu kita cari lagi nilai tengahnya, didapatlah angka 14 sebagai nilai tengah. Lalu, kita bandingkan apakah 13 > 14 atau 13 < 14?
- Ternyata 13 lebih kecil dari 14, maka selanjutnya kita cari ke kiri.
- Karna tersisa 1 elemen saja, maka elemen tersebut adalah nilai tengahnya.
- Setelah dicek ternyata elemen pada indeks ke-4 adalah elemen yang dicari, maka telah selesai proses pencariannya.

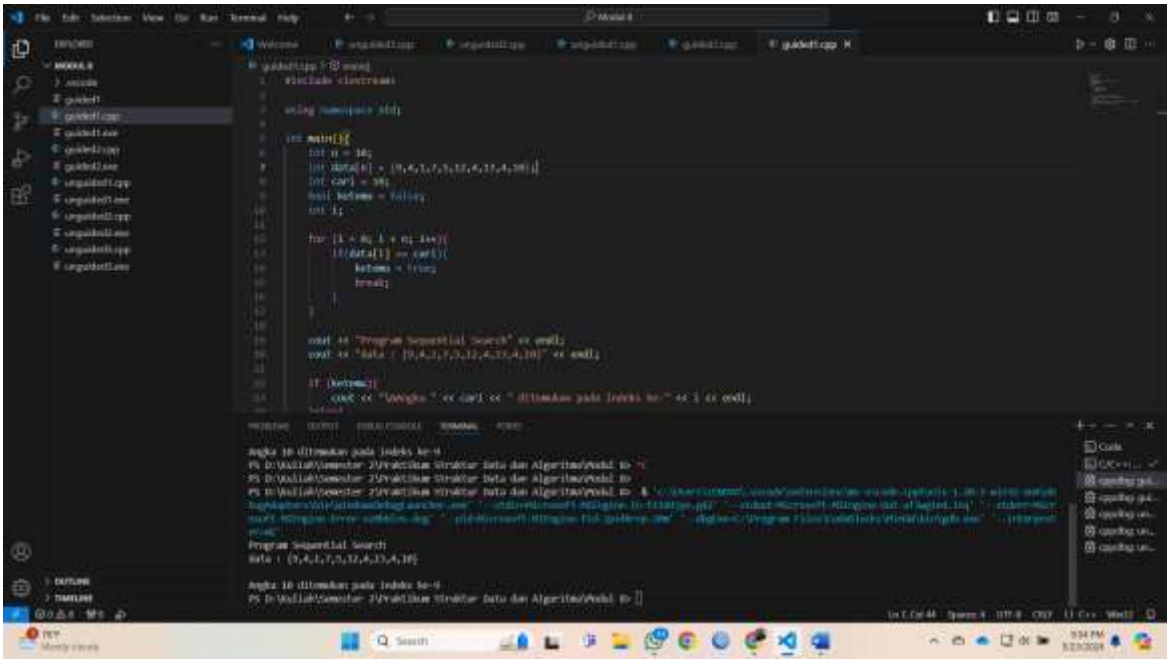
BAB III
GUIDED

1. Guided 1

Program:

```
#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;

    for (i = 0; i < n; i++){
        if(data[i] == cari){
            ketemu = true;
            break;
        }
    }

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

    if (ketemu){
        cout << "\nAngka " << cari << " ditemukan pada indeks ke-" << i
<< endl;
    }else{
        cout << "data tidak ditemukan" << endl;
    }

    return 0;
}
```

Screenshoot Program:



Deskripsi Program:

Program di atas adalah implementasi sederhana dari algoritma pencarian sekuensial (sequential search) dalam bahasa C++. Program ini mencari sebuah angka tertentu, yaitu `10`, dalam sebuah array yang berisi 10 elemen. Pertama, array `data` diinisialisasi dengan nilai-nilai tertentu, dan variabel `cari` diset dengan nilai `10` yang akan dicari dalam array tersebut. Variabel `ketemu` digunakan sebagai flag untuk menandai apakah angka yang dicari ditemukan atau tidak. Program ini kemudian melakukan iterasi melalui setiap elemen array `data` menggunakan loop `for`. Jika angka yang dicari ditemukan pada salah satu elemen array, flag `ketemu` diset menjadi `true` dan loop dihentikan. Setelah pencarian selesai, program mencetak hasil pencarian ke layar, yaitu apakah angka `10` ditemukan dan pada indeks ke berapa atau mencetak pesan bahwa data tidak ditemukan.

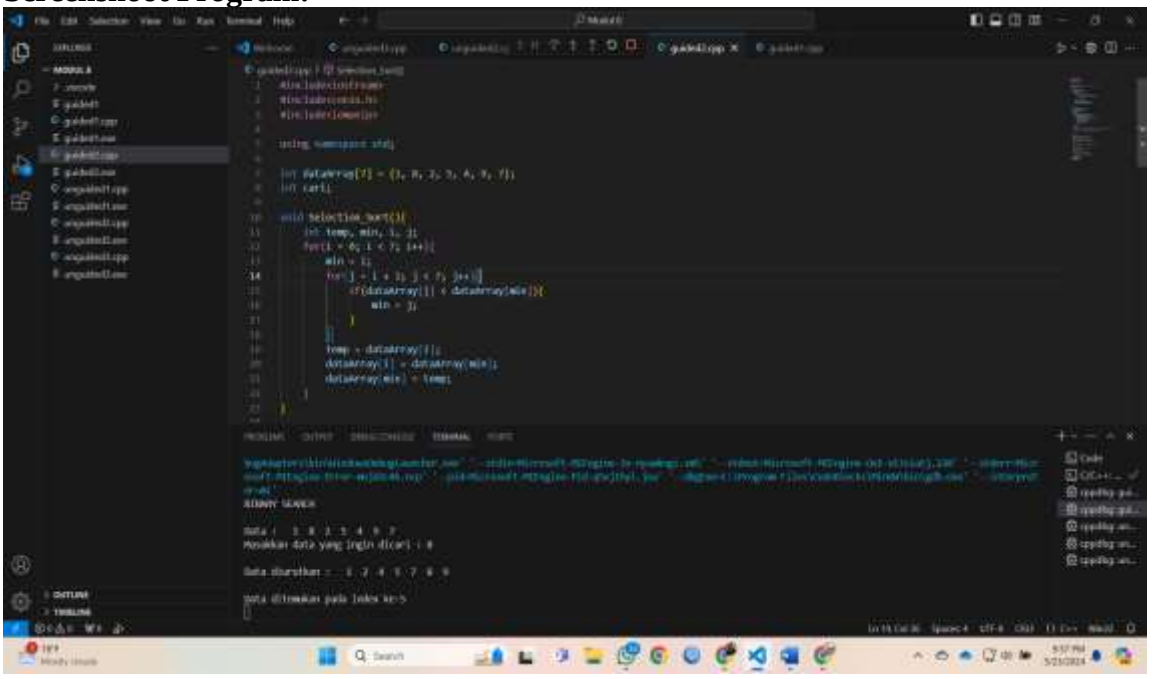
2. **Guided 2**

Program :

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

using namespace std;

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

void BinarySearch(){
    int awal, akhir, tengah;
    bool b_flag = false;
    awal = 0;
    akhir = 6;
    while(!b_flag && awal <= akhir){
        tengah = (awal + akhir)/2;
        if(dataArray[tengah] == cari){
            b_flag = true;
        } else if(dataArray[tengah] < cari){
            awal = tengah + 1;
        } else {
            akhir = tengah - 1;
        }
    }
    if(b_flag){
        cout << "\nData ditemukan pada index ke-" << tengah << endl;
    } else {
        cout << "\nData tidak ditemukan" << endl;
    }
}

int main(){
    cout << "BINARY SEARCH" << endl;
```

```
    cout << "\nData : ";
    for(int x = 0; x < 7; x++){
        cout << setw(3) << dataArray[x];
    }
    cout << endl;

    cout << "Masukkan data yang ingin dicari : ";
    cin >> cari;

    cout << "\nData diurutkan : ";
    Selection_Sort();

    for(int x = 0; x < 7; x++){
        cout << setw(3) << dataArray[x];
    }
    cout << endl;
    BinarySearch();
    _getche();
    return 0;
}
```

Screenshoot Program:



Deskripsi Program:

Program di atas mengimplementasikan pencarian biner dalam bahasa C++. Program ini pertama-tama mendeklarasikan sebuah array `dataArray` dengan 7 elemen integer yang tidak terurut. Pengguna diminta untuk memasukkan nilai yang ingin dicari dalam array tersebut. Program kemudian mengurutkan array menggunakan algoritma selection sort, dan menampilkan array yang sudah terurut. Setelah itu, dilakukan pencarian biner pada array yang sudah diurutkan untuk menemukan indeks elemen yang dicari. Jika elemen ditemukan, program akan mencetak indeks elemen tersebut; jika tidak ditemukan, program akan memberitahukan bahwa elemen tidak ada dalam array. Program menggunakan fungsi `_getche()` untuk menunggu input karakter sebelum mengakhiri eksekusi, sehingga hasil dapat dilihat oleh pengguna.

LATIHAN KELAS-UNGUIDED

1. Buatlah sebuah program untuk mencari sebuah huruf pada sebuah kalimat yangsudah di input dengan menggunakan Binary Search!

Program:

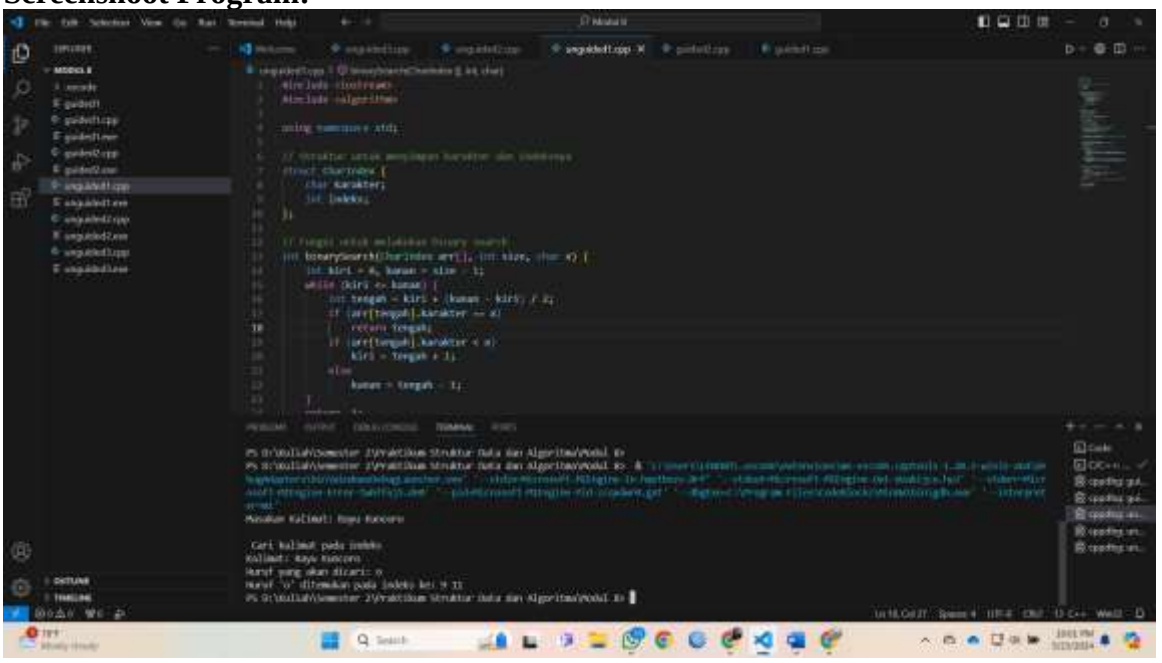
```
#include <iostream>
#include <algorithm>

using namespace std;

// Struktur untuk menyimpan karakter dan indeksnya
struct CharIndex {
    char karakter;
    int indeks;
};

// Fungsi untuk melakukan binary search
int binarySearch(CharIndex arr[], int size, char x) {
    int kiri = 0, kanan = size - 1;
    while (kiri <= kanan) {
        int tengah = kiri + (kanan - kiri) / 2;
        if (arr[tengah].karakter == x)
            return tengah;
        if (arr[tengah].karakter < x)
            kiri = tengah + 1;
        else
            kanan = tengah - 1;
    }
    return -1;
}

int main() {
    string kalimat, hurufDicari;
    cout << "Masukan Kalimat: ";
    getline(cin, kalimat);

    cout << "\n Cari kalimat pada indeks" << endl;
    cout << "Kalimat: " << kalimat << endl;

    cout << "Huruf yang akan dicari: ";
    cin >> hurufDicari;

    char huruf = hurufDicari[0];
    int length = kalimat.length();

    // Membuat array untuk menyimpan karakter dan indeksnya
    CharIndex* charIndexArray = new CharIndex[length];
```

```
    for (int i = 0; i < length; i++) {
        charIndexArray[i].karakter = kalimat[i];
```

```
        charIndexArray[i].indeks = i;
    }

    // Mengurutkan array berdasarkan karakter
    sort(charIndexArray, charIndexArray + length, [](CharIndex a,
CharIndex b) {
        return a.karakter < b.karakter;
    });

    // Mencari huruf yang dicari
    int index = binarySearch(charIndexArray, length, huruf);
    int* resultIndices = new int[length];
    int resultCount = 0;

    // Jika huruf ditemukan, kita cari semua indeks kemunculannya
    if (index != -1) {
        int tempIndex = index;
        // Cari ke kiri
        while (tempIndex >= 0 && charIndexArray[tempIndex].karakter ==
huruf) {
            resultIndices[resultCount++] =
charIndexArray[tempIndex].indeks;
            tempIndex--;
        }
        // Cari ke kanan
        tempIndex = index + 1;
        while (tempIndex < length && charIndexArray[tempIndex].karakter
== huruf) {
            resultIndices[resultCount++] =
charIndexArray[tempIndex].indeks;
            tempIndex++;
        }
    }

    // Mengurutkan indeks yang ditemukan
    sort(resultIndices, resultIndices + resultCount);

    // Menampilkan hasil
    if (resultCount > 0) {
        cout << "Huruf '" << huruf << "' ditemukan pada indeks ke: ";
        for (int i = 0; i < resultCount; i++) {
            cout << resultIndices[i] << " ";
        }
        cout << endl;
    } else {
        cout << "Huruf '" << huruf << "' tidak ditemukan dalam kalimat."
<< endl;
    }

    // Membersihkan memori
```

```
    delete[] charIndexArray;
    delete[] resultIndices;

    return 0;
}
```

Screenshoot Program:



Deskripsi Program:

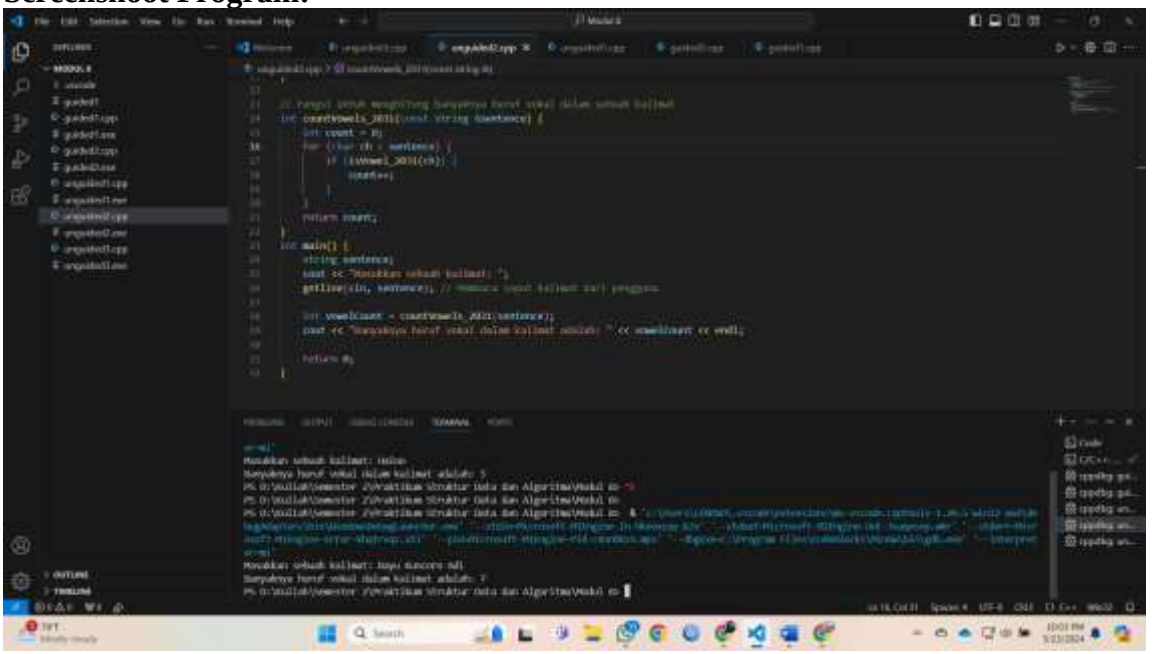
Program di atas meminta pengguna untuk memasukkan sebuah kalimat dan sebuah huruf yang ingin dicari dalam kalimat tersebut. Program ini kemudian menyimpan setiap karakter dari kalimat beserta indeksnya dalam array struct bernama `CharIndex`. Setelah itu, array tersebut diurutkan berdasarkan karakter menggunakan fungsi `sort` dari pustaka `<algorithm>`. Program kemudian menggunakan binary search untuk menemukan huruf yang dicari dalam array yang telah diurutkan. Jika huruf ditemukan, program akan mencari dan mencatat semua indeks kemunculan huruf tersebut, baik ke kiri maupun ke kanan dari posisi ditemukan pertama kali. Indeks-indeks yang ditemukan kemudian diurutkan dan ditampilkan kepada pengguna. Program mengakhiri dengan membersihkan memori yang dialokasikan secara dinamis.

2. Buatlah sebuah program yang dapat menghitung banyaknya huruf vocal dalamsebuah kalimat!

Program:

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

// Fungsi untuk memeriksa apakah sebuah karakter adalah vokal
bool isVowel_2031(char ch) {
    // Konversi karakter menjadi huruf kecil
    ch = tolower(ch);
    // Periksa apakah karakter adalah salah satu dari 'a', 'e', 'i',
'o', 'u'
    return (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch ==
'u');
}

// Fungsi untuk menghitung banyaknya huruf vokal dalam sebuah kalimat
int countVowels_2031(const string &sentence) {
    int count = 0;
    for (char ch : sentence) {
        if (isVowel_2031(ch)) {
            count++;
        }
    }
    return count;
}
int main() {
    string sentence;
    cout << "Masukkan sebuah kalimat: ";
    getline(cin, sentence); // Membaca input kalimat dari pengguna

    int vowelCount = countVowels_2031(sentence);
    cout << "Banyaknya huruf vokal dalam kalimat adalah: " <<
vowelCount << endl;

    return 0;
}
```

Screenshoot Program:



Deskripsi Program:

Program di atas adalah program C++ yang menghitung jumlah huruf vokal dalam sebuah kalimat yang dimasukkan oleh pengguna. Program ini pertama-tama mendefinisikan fungsi `isVowel_2031` untuk memeriksa apakah sebuah karakter adalah huruf vokal (a, e, i, o, u) dengan mengubah karakter tersebut menjadi huruf kecil dan membandingkannya dengan karakter vokal. Kemudian, fungsi `countVowels_2031` digunakan untuk menghitung jumlah huruf vokal dalam sebuah kalimat dengan menggunakan fungsi `isVowel_2031`. Dalam fungsi `main`, program meminta pengguna untuk memasukkan sebuah kalimat, membaca kalimat tersebut menggunakan `getline`, dan kemudian menghitung serta menampilkan jumlah huruf vokal yang terdapat dalam kalimat tersebut.

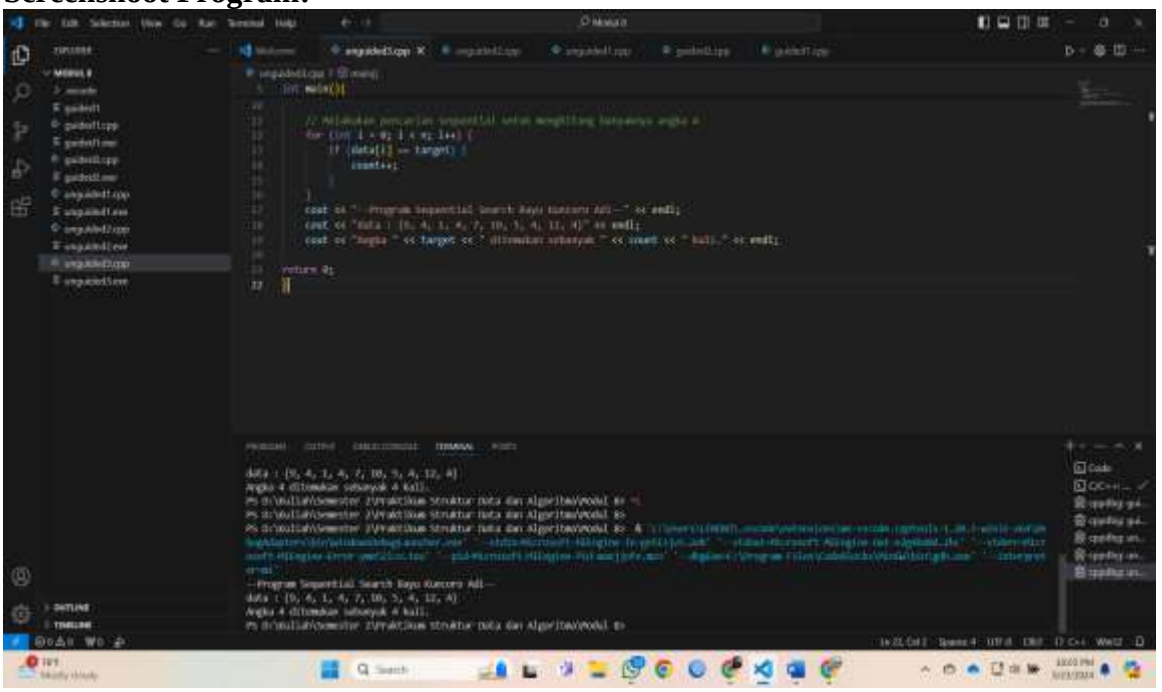
3. Diketahui data = 9, 4, 1, 4, 7, 10, 5, 4, 12, 4. Hitunglah berapa banyak angka 4 dengan menggunakan algoritma Sequential Search!

Program:

```
#include <iostream>

using namespace std;

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

    // Melakukan pencarian sequential untuk menghitung banyaknya angka
4
    for (int i = 0; i < n; i++) {
        if (data[i] == target) {
            count++;
        }
```

```
    }
    cout << "--Program Sequential Search Bayu Kuncoro Adi--" << endl;
    cout << "data : {9, 4, 1, 4, 7, 10, 5, 4, 12, 4}" << endl;
    cout << "Angka " << target << " ditemukan sebanyak " << count << "
kali." << endl;

return 0;
}
```

Screenshoot Program:



Deskripsi Program:

Program di atas adalah program pencarian sequential sederhana yang ditulis dalam bahasa C++. Program ini menginisialisasi sebuah array `data` dengan 10 elemen dan sebuah variabel `target` dengan nilai 4. Tujuan dari program ini adalah untuk menghitung berapa kali nilai `target` muncul dalam array `data`. Program menggunakan loop `for` untuk memeriksa setiap elemen array, dan setiap kali elemen yang diperiksa sama dengan nilai `target`, variabel `count` akan ditambah satu. Setelah loop selesai, program menampilkan jumlah kemunculan nilai `target` dalam array `data` melalui output standar. Program juga menampilkan informasi tentang data yang digunakan dan pesan deskriptif tentang tujuan program.

BAB IV
KESIMPULAN

Sequential Search adalah algoritma pencarian data yang paling sederhana dan mudah dipahami. Algoritma ini bekerja dengan cara membandingkan data yang dicari dengan setiap elemen dalam array secara berurutan. Jika data yang dicari ditemukan, maka proses pencarian akan dihentikan dan program akan menampilkan informasi tentang indeks data yang ditemukan.

Sequential Search juga dapat digunakan untuk menghitung jumlah kemunculan data tertentu dalam array. Hal ini dilakukan dengan cara menambahkan satu ke variabel pencacah setiap kali data yang dicari ditemukan. Setelah loop pencarian selesai, variabel pencacah akan berisi jumlah total kemunculan data yang dicari.

Binary Search adalah algoritma pencarian data yang efisien untuk digunakan pada array yang telah terurut. Algoritma ini bekerja dengan cara membagi array menjadi dua bagian secara berulang, dan membandingkan data yang dicari dengan nilai tengah dari array yang dibagi.

Program di atas menunjukkan contoh implementasi Sequential Search dalam bahasa C++. Program ini mencari angka 4 dalam array data dan menghitung berapa kali angka tersebut muncul. Program ini menggunakan loop for untuk melakukan pencarian dan variabel count untuk menampung jumlah kemunculan angka 4.

Sequential Search adalah algoritma pencarian yang mudah diimplementasikan dan efektif untuk mencari data dalam array yang tidak terurut. Algoritma ini juga dapat digunakan untuk menghitung jumlah kemunculan data tertentu dalam array.

DAFTAR PUSTAKA

Karumanchi, N. (2016). *Data Structures and algorithms made easy: Concepts, problems, Interview Questions*. CareerMonk Publications

Modul 8 Searching Praktikum Struktur Data dan Algoritma Pemrograman

Modul Searching https://elektro.um.ac.id/wp-content/uploads/2016/04/MODUL-9-SEARCHING.pdf

Struktur Data diakses dari https://github.com/yunusfebriansyah/struktur-data/tree/main/Circular%20Single%20Linked%20List