

LAPORAN PRAKTIKUM

MODUL VIII ALGORITMA SEARCHING



Disusun oleh:
Muhammad Agha Zulfadhli
NIM: 2311102015

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. Mampu memahami searching pada struktur data dan algoritma
- b. Mampu mengimplementasikan operasi-operasi pada searching
- c. Mampu memecahkan permasalahan dengan solusi searching

BAB II

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:

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) $i \leftarrow 0$
- 2) $ketemu \leftarrow false$
- 3) Selama (tidak ketemu) dan $(i \leq N)$ kerjakan baris 4
- 4) Jika $(Data[i] = x)$ maka $ketemu \leftarrow true$, jika tidak $i \leftarrow i + 1$
- 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.

```

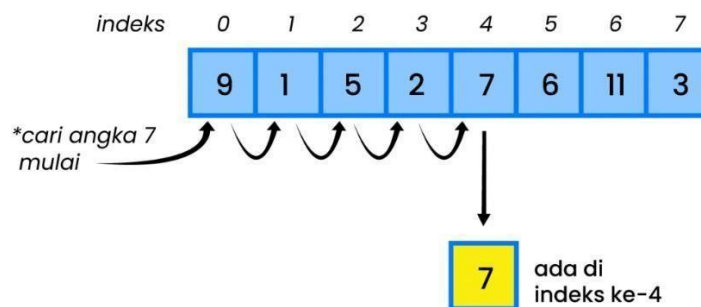
int SequentialSearch (int x)
{
    int i = 0;
    bool ketemu = false;
    while ((!ketemu) && (i < Max)){
        if (Data[i] == x)
            ketemu = true;
        else
            i++;
    }
    if (ketemu)
        return i;
    else
        return -1;
}

```

Fungsi diatas akan mengembalikan indeks dari data yang dicari. Apabila data tidak ditemukan maka fungsi diatas akan mengembalikan nilai -1.

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

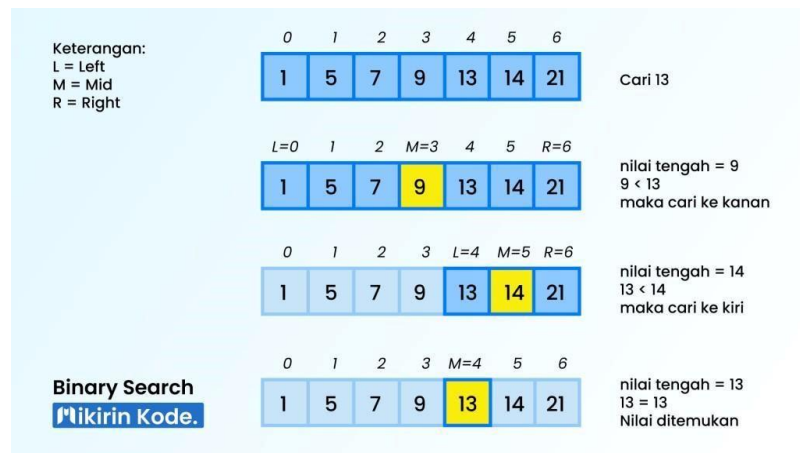
Binary Search termasuk ke dalam interval search, dimana algoritma ini merupakan algoritma pencarian pada array/list dengan elemen terurut. Pada metode ini, data harus diurutkan terlebih dahulu dengan cara data dibagi menjadi dua bagian (secara logika), untuk setiap tahap pencarian. Dalam penerapannya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

Algoritma pencarian biner dapat dituliskan sebagai berikut :

- 1) $L = 0$
- 2) $R = N - 1$
- 3) ketemu false
- 4) Selama $(L \leq R)$ dan (tidak ketemu) kerjakan baris 5 sampai dengan 8
- 5) $m = (L + R) / 2$
- 6) Jika $(\text{Data}[m] = x)$ maka ketemu true
- 7) Jika $(x < \text{Data}[m])$ maka $R = m - 1$
- 8) Jika $(x > \text{Data}[m])$ 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

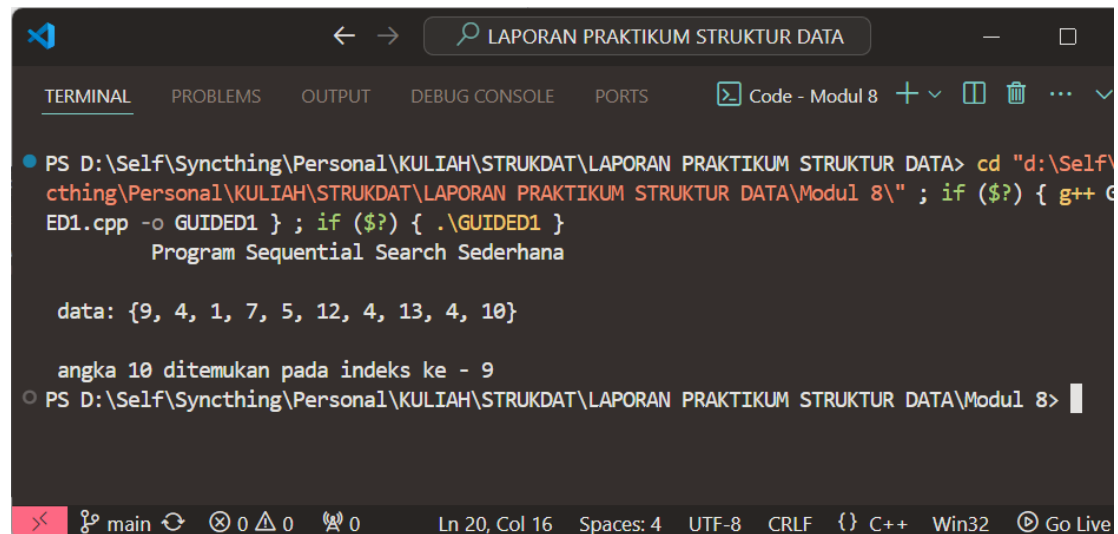
GUIDED

1. Guided 1

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;
}
```


Screenshoot program



```
VS LAPORAN PRAKTIKUM STRUKTUR DATA
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE PORTS Code - Modul 8
PS D:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA> cd "d:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA\Modul 8\" ; 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 D:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA\Modul 8>
```

Deskripsi program

Program di atas adalah implementasi sederhana dari algoritma pencarian sekuensial (sequential search). Tujuan dari program ini adalah untuk mencari sebuah angka tertentu dalam sebuah array. Berikut adalah penjelasan detail tentang bagaimana program ini bekerja:

Program dimulai dengan mendeklarasikan sebuah variabel `n` yang diinisialisasi dengan nilai 10, yang menunjukkan jumlah elemen dalam array `data`. Array `data` tersebut berisi sepuluh bilangan bulat, yaitu `{9, 4, 1, 7, 5, 12, 4, 13, 4, 10}`. Selain itu, ada variabel `cari` yang diinisialisasi dengan nilai 10, yang merupakan angka yang akan dicari dalam array `data`. Sebuah variabel boolean `ketemu` diinisialisasi dengan nilai `false`, yang akan digunakan untuk menandai apakah angka yang dicari ditemukan atau tidak. Variabel `i` juga dideklarasikan untuk digunakan sebagai indeks dalam proses pencarian.

Program kemudian menggunakan loop ``for`` untuk menjalankan algoritma pencarian sekuensial. Loop ini akan berjalan dari indeks 0 hingga 9 (total 10 elemen). Di dalam loop, terdapat sebuah pernyataan ``if`` yang memeriksa apakah elemen pada indeks ``i`` dari array ``data`` sama dengan angka yang dicari (``cari``). Jika kondisi ini terpenuhi, variabel ``ketemu`` diubah menjadi ``true``, dan loop dihentikan dengan perintah ``break``.

Setelah loop selesai, program akan mencetak hasil pencarian. Pertama, program mencetak judul "Program Sequential Search Sederhana" dan daftar elemen dari array ``data``. Kemudian, jika angka yang dicari ditemukan (nilai ``ketemu`` adalah ``true``), program akan mencetak pesan bahwa angka tersebut ditemukan pada indeks ke-``i``. Sebaliknya, jika angka tidak ditemukan (nilai ``ketemu`` tetap ``false``), program akan mencetak pesan bahwa angka tersebut tidak dapat ditemukan dalam array.

Akhirnya, program mengembalikan nilai 0 untuk menandakan bahwa program telah selesai dieksekusi tanpa kesalahan.

2. Guided 2

Source code

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

int data1[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 (data1[j] < data1[min])
            {
                min = j;
            }
        }
        temp = data1[i];
        data1[i] = data1[min];
        data1[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 (data1[tengah] == cari)
        {
            b_flag = 1;
            break;
        }
        else if (data1[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) << data1[x];
    cout << endl;
    cout << "\n Masukkan data yang ingin Anda cari :";
    cin >> cari;
    cout << "\n Data diurutkan : ";
    // urutkan data1 dengan selection sort
    selection_sort();
    // tampilkan data1 setelah diurutkan
    for (int x = 0; x < 7; x++)
        cout << setw(3) << data1[x];
    cout << endl;
    binarysearch();
    _getche();
    return EXIT_SUCCESS;
}

```

Screenshoot program

The screenshot shows a Visual Studio Code terminal window with the following content:

```

IDED2.cpp -o GUIDED2 } ; if ($?) { .\GUIDED2 }
BINARY SEARCH

Data :  1  8  2  5  4  9  7

Masukkan data yang ingin Anda cari :8

Data diurutkan :  1  2  4  5  7  8  9

Data ditemukan pada index ke- 5
a
PS D:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA\Modul 8>

```

The terminal window has a title bar that says "LAPORAN PRAKTIKUM STRUKTUR DATA". The status bar at the bottom shows "Ln 44, Col 32", "Spaces: 4", "UTF-8", "CRLF", "{ } C++", "Win32", and "Go Live".

Deskripsi program

Program di atas adalah implementasi dari algoritma pengurutan (selection sort) dan pencarian biner (binary search) dalam bahasa C++. Tujuan dari program ini adalah untuk mencari sebuah angka tertentu dalam array setelah mengurutkannya terlebih dahulu. Berikut adalah penjelasan rinci tentang cara kerja program ini:

Program dimulai dengan menyertakan pustaka `<iostream>` untuk operasi input/output dan `<iomanip>` untuk manipulasi output. Pustaka `<conio.h>` juga disertakan untuk menggunakan fungsi `_getche()`, yang memungkinkan program menunggu input dari pengguna sebelum keluar.

Array `data1` berisi tujuh bilangan bulat, yaitu `{1, 8, 2, 5, 4, 9, 7}`. Variabel `cari` dideklarasikan untuk menyimpan angka yang akan dicari oleh pengguna.

Fungsi `selection_sort` mengimplementasikan algoritma selection sort untuk mengurutkan array `data1`. Dalam algoritma ini, elemen terkecil dalam bagian yang belum diurutkan dari array ditemukan dan ditukar dengan elemen pertama dari bagian tersebut. Proses ini diulang hingga seluruh array terurut.

Fungsi `binarysearch` mengimplementasikan algoritma pencarian biner untuk mencari angka yang dimasukkan pengguna dalam array `data1` yang sudah diurutkan. Algoritma ini bekerja dengan membagi array menjadi dua bagian dan membandingkan elemen tengah dengan angka yang dicari. Jika elemen tengah adalah angka yang dicari, pencarian berhenti. Jika tidak, pencarian berlanjut di separuh array yang sesuai.

Proses ini berulang hingga angka ditemukan atau bagian array yang dicari habis.

Dalam fungsi ``main``, program pertama-tama mencetak judul "BINARY SEARCH" dan menampilkan data awal dari array ``data1``. Kemudian, pengguna diminta untuk memasukkan angka yang ingin dicari. Setelah itu, program mengurutkan array menggunakan fungsi ``selection_sort`` dan menampilkan array yang sudah diurutkan. Selanjutnya, fungsi ``binarysearch`` dipanggil untuk mencari angka dalam array yang sudah diurutkan dan menampilkan hasil pencarian, yaitu indeks dari angka tersebut jika ditemukan, atau pesan bahwa angka tidak ditemukan jika tidak ada di dalam array.

Akhirnya, program menunggu input dari pengguna sebelum keluar menggunakan ``_getche()``, dan mengembalikan nilai ``EXIT_SUCCESS`` untuk menandakan bahwa program telah selesai dieksekusi tanpa kesalahan.

LATIHAN KELAS - UNGUIDED

Unguided 1

Source code

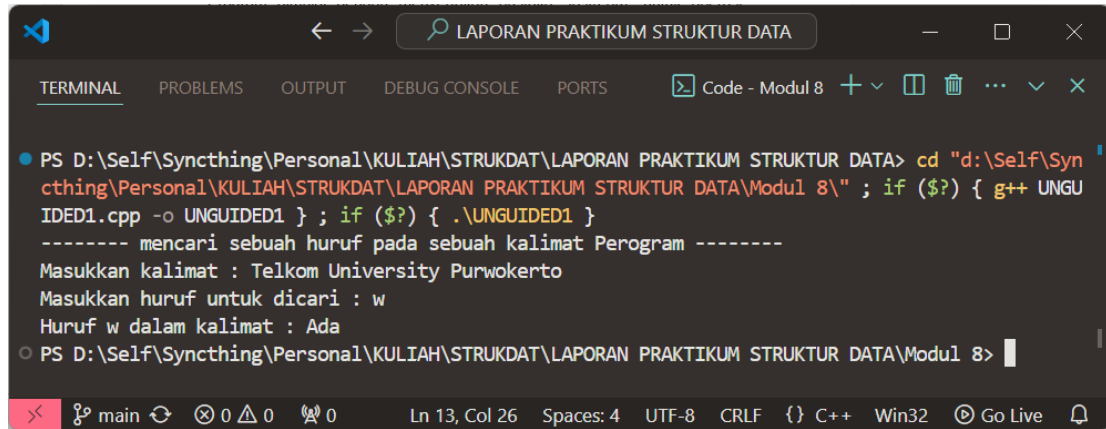
```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;

bool binarySearch(string kalimat, char key) {
    // searching
    sort(kalimat.begin(), kalimat.end());
    int awal = 0, akhir = kalimat.size() , tengah;
    bool found;
    while (!found && awal <= akhir) {
        tengah = (awal + akhir) / 2;
        if (kalimat[tengah] == key) {
            found = true;
            break;
        }
        else if (kalimat[tengah] < key) awal = tengah + 1;
        else akhir = tengah - 1;
    }
    return found;
}

int main() {
    string kalimat;
    char inputKata;
    cout << "----- mencari sebuah huruf pada sebuah kalimat
Perogram -----\\n";
    cout << "Masukkan kalimat : ";
    getline(cin, kalimat);
    cout << "Masukkan huruf untuk dicari : ";
    cin >> inputKata;

    cout << "Huruf " << inputKata << " dalam kalimat : " <<
(binarySearch(kalimat, inputKata) ? "Ada" : "Tidak ada");
}
```

Screenshoot program



```
PS D:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA> cd "d:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA\Modul 8\" ; if ($?) { g++ UNGUIDED1.cpp -o UNGUIDED1 } ; if ($?) { .\UNGUIDED1 }
----- mencari sebuah huruf pada sebuah kalimat Perogram -----
Masukkan kalimat : Telkom University Purwokerto
Masukkan huruf untuk dicari : w
Huruf w dalam kalimat : Ada
PS D:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA\Modul 8>
```

Deskripsi program

Program di atas adalah implementasi dari algoritma pencarian biner (binary search) untuk mencari sebuah karakter dalam sebuah kalimat yang diberikan oleh pengguna. Berikut adalah penjelasan rinci tentang cara kerja program ini:

Program dimulai dengan menyertakan pustaka ``iostream`` untuk operasi input/output dan ``bits/stdc++.h`` untuk mengakses berbagai fungsi dan pustaka standar C++.

Fungsi ``binarySearch`` mengambil dua parameter: sebuah string ``kalimat`` dan sebuah karakter ``key`` yang akan dicari. Pertama, fungsi ini mengurutkan string ``kalimat`` menggunakan fungsi ``sort`` dari pustaka standar. Kemudian, variabel ``awal``, ``akhir``, dan ``tengah`` diinisialisasi untuk menentukan batas pencarian biner. Variabel ``awal`` diatur ke indeks 0, dan ``akhir`` diatur ke ukuran string ``kalimat``.

Pencarian biner dilakukan dalam loop ``while``, yang terus berlanjut selama ``found`` adalah ``false`` dan ``awal`` lebih kecil atau sama dengan ``akhir``. Pada

setiap iterasi, indeks tengah dihitung dan karakter pada indeks tersebut dibandingkan dengan `key`. Jika karakter pada indeks tengah sama dengan `key`, `found` diatur menjadi `true` dan loop berhenti. Jika karakter pada indeks tengah lebih kecil dari `key`, variabel `awal` diperbarui ke `tengah + 1` untuk mencari di separuh bagian kanan. Sebaliknya, jika karakter pada indeks tengah lebih besar dari `key`, variabel `akhir` diperbarui ke `tengah - 1` untuk mencari di separuh bagian kiri. Fungsi mengembalikan `true` jika karakter ditemukan, dan `false` jika tidak.

Dalam fungsi `main`, program meminta pengguna untuk memasukkan sebuah kalimat dan karakter yang akan dicari. Pengguna memasukkan kalimat menggunakan `getline` untuk memungkinkan input dengan spasi. Karakter yang akan dicari dimasukkan menggunakan `cin`. Program kemudian memanggil fungsi `binarySearch` dengan kalimat dan karakter yang dimasukkan, dan mencetak hasil pencarian. Jika karakter ditemukan, program mencetak "Ada"; jika tidak ditemukan, program mencetak "Tidak ada".

Unguided 2

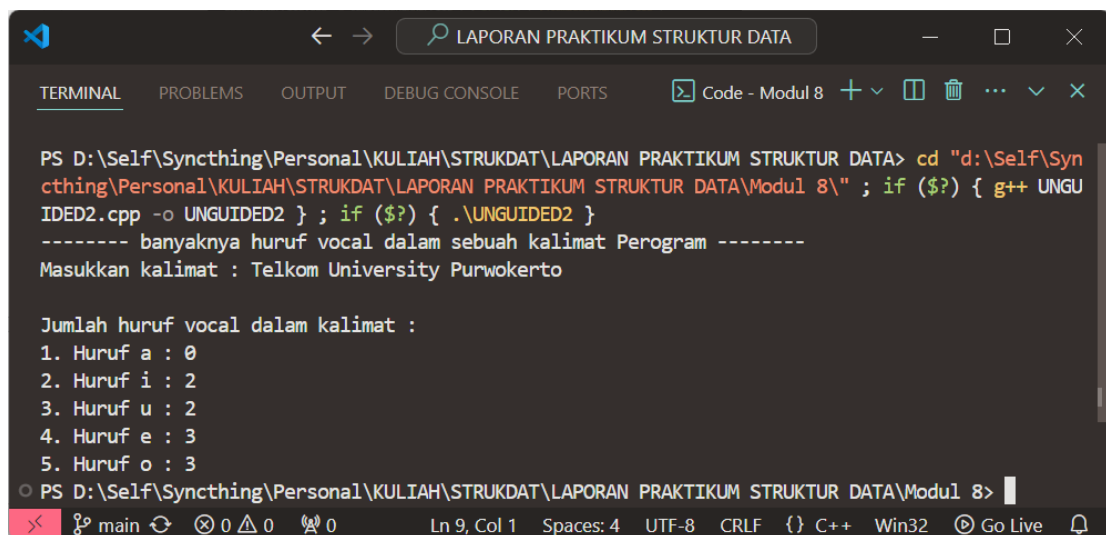
Source code

```
#include <iostream>
using namespace std;
int main() {
    string kalimat;
    int vocal[5] = {0,0,0,0,0};
    cout << "----- banyaknya huruf vocal dalam sebuah kalimat
Perogram -----\\n";
    cout << "Masukkan kalimat : ";
    getline(cin, kalimat);

    for (int i = 0; i < kalimat.size() ; i++) {
        if (kalimat[i] == 'a' || kalimat[i] == 'A') {vocal[0]++;}
        if (kalimat[i] == 'i' || kalimat[i] == 'I') {vocal[1]++;}
        if (kalimat[i] == 'u' || kalimat[i] == 'U') {vocal[2]++;}
        if (kalimat[i] == 'e' || kalimat[i] == 'E') {vocal[3]++;}
        if (kalimat[i] == 'o' || kalimat[i] == 'O') {vocal[4]++;}
    }

    cout << "\\nJumlah huruf vocal dalam kalimat : \\n"
    "1. Huruf a : " << vocal[0] << endl <<
    "2. Huruf i : " << vocal[1] << endl <<
    "3. Huruf u : " << vocal[2] << endl <<
    "4. Huruf e : " << vocal[3] << endl <<
    "5. Huruf o : " << vocal[4] << endl;
}
```

Screenshoot program



```
PS D:\Self\Syncting\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA> cd "d:\Self\Syncting\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA\Modul 8\" ; if ($?) { g++ UNGUIDED2.cpp -o UNGUIDED2 } ; if ($?) { .\UNGUIDED2 }
----- banyaknya huruf vocal dalam sebuah kalimat Perogram -----
Masukkan kalimat : Telkom University Purwokerto

Jumlah huruf vocal dalam kalimat :
1. Huruf a : 0
2. Huruf i : 2
3. Huruf u : 2
4. Huruf e : 3
5. Huruf o : 3
PS D:\Self\Syncting\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA\Modul 8>
```

Deskripsi program

Program di atas adalah implementasi sederhana untuk menghitung jumlah kemunculan setiap huruf vokal dalam sebuah kalimat yang diberikan oleh pengguna. Berikut adalah penjelasan rinci tentang cara kerja program ini:

Program dimulai dengan menyertakan pustaka `<iostream>` untuk operasi input/output. Di dalam fungsi `main`, sebuah array `vocal` dengan ukuran 5 diinisialisasi dengan nilai nol. Array ini digunakan untuk menyimpan jumlah kemunculan masing-masing huruf vokal ('a', 'i', 'u', 'e', 'o') dalam kalimat.

Program kemudian mencetak judul "----- banyaknya huruf vokal dalam sebuah kalimat Program -----" dan meminta pengguna untuk memasukkan sebuah kalimat. Kalimat ini dibaca menggunakan `getline(cin, kalimat)`, yang memungkinkan pengguna untuk memasukkan kalimat yang mengandung spasi.

Setelah kalimat dimasukkan, program memulai loop `for` yang berjalan dari indeks 0 hingga panjang kalimat. Pada setiap iterasi, program memeriksa karakter pada indeks `i` dari string `kalimat`. Jika karakter tersebut adalah 'a', nilai pada indeks pertama `vocal` ditingkatkan. Jika karakter tersebut adalah 'i', nilai pada indeks kedua `vocal` ditingkatkan, dan seterusnya untuk huruf 'u', 'e', dan 'o'.

Setelah loop selesai, program mencetak jumlah kemunculan masing-masing huruf vokal dengan menggunakan array `vocal`. Setiap elemen dari array `vocal` mewakili jumlah kemunculan satu huruf vokal: `vocal[0]` untuk 'a', `vocal[1]` untuk 'i', `vocal[2]` untuk 'u', `vocal[3]` untuk 'e', dan `vocal[4]` untuk 'o'.

Dengan demikian, program ini menghitung dan menampilkan jumlah setiap huruf vokal dalam kalimat yang dimasukkan oleh pengguna, memberikan informasi yang jelas tentang berapa kali masing-masing vokal muncul dalam kalimat tersebut.

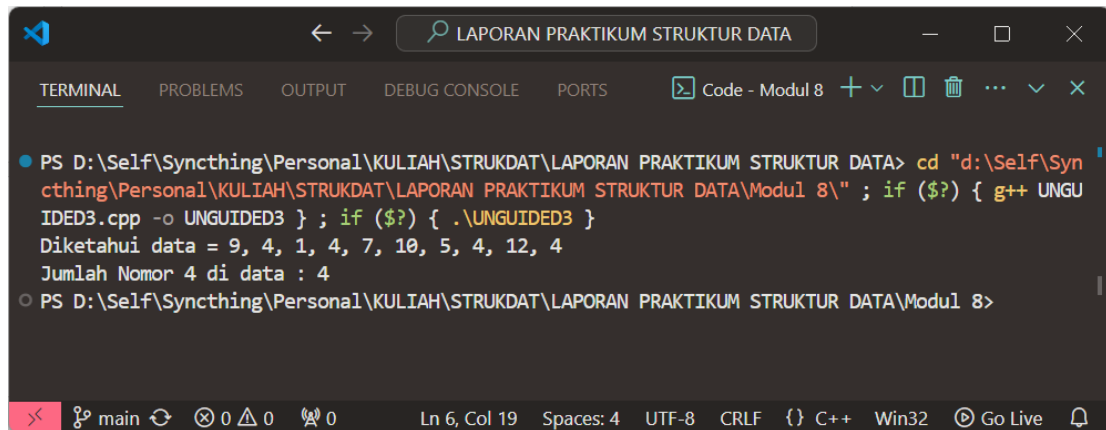
Unguided 3

Source code

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

int main() {
    int dataNum[] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
    int num4 = 0 ;
    for (int i = 0; i < 10; i++) if (dataNum[i] == 4) num4++;
    cout << "Diketahui data = 9, 4, 1, 4, 7, 10, 5, 4, 12, 4\n"
    "Jumlah Nomor 4 di data : " << num4;
}
```

Screenshoot program



```
PS D:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA> cd "d:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA\Modul 8\" ; if ($?) { g++ UNGUIDED3.cpp -o UNGUIDED3 } ; if ($?) { .\UNGUIDED3 }
Diketahui data = 9, 4, 1, 4, 7, 10, 5, 4, 12, 4
Jumlah Nomor 4 di data : 4
PS D:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA\Modul 8>
```

Deskripsi program

Program di atas adalah sebuah program sederhana yang menghitung jumlah kemunculan angka 4 dalam sebuah array data.

Program dimulai dengan mendeklarasikan sebuah array dataNum yang berisi sepuluh bilangan bulat. Array ini berfungsi sebagai data yang akan diperiksa untuk menghitung jumlah kemunculan angka 4.

Selanjutnya, program menggunakan loop for untuk mengiterasi melalui setiap elemen dalam array dataNum. Pada setiap iterasi, program memeriksa apakah nilai elemen tersebut sama dengan 4. Jika iya, variabel num4 (yang digunakan untuk menghitung jumlah kemunculan angka 4) akan ditambah satu.

BAB IV

KESIMPULAN

Pencarian data (searching) adalah proses menemukan nilai tertentu dalam kumpulan data yang dapat menghasilkan tiga kemungkinan: data ditemukan, data ditemukan lebih dari satu, atau data tidak ditemukan. Terdapat dua metode utama dalam algoritma pencarian: Sequential Search dan Binary Search. Berikut adalah kesimpulan dari penjelasan di atas:

1. Sequential Search, Metode ini efektif untuk data yang tidak terurut dan melibatkan pencarian data dengan memeriksa setiap elemen dalam array satu per satu dari indeks pertama hingga terakhir. Pencarian akan berhenti ketika data ditemukan atau setelah seluruh elemen diperiksa.
2. Binary Search, Metode ini digunakan untuk data yang sudah terurut. Algoritma ini membagi array menjadi dua bagian dan membandingkan elemen tengah dengan data yang dicari. Proses ini berlanjut dengan mempersempit pencarian ke separuh array yang relevan, hingga data ditemukan atau pencarian selesai.
3. Sequential Search lebih sederhana dan mudah diimplementasikan, namun kurang efisien untuk dataset besar karena memeriksa setiap elemen. Sebaliknya, Binary Search lebih efisien untuk dataset besar karena mengurangi jumlah elemen yang diperiksa secara signifikan melalui proses pembagian.
4. Implementasi Algoritma Kedua metode pencarian ini dapat diimplementasikan dengan algoritma yang jelas dan terstruktur, yang masing-masing memiliki kegunaan dan keunggulan tergantung pada kondisi dan karakteristik data yang sedang diolah.

DAFTAR PUSTAKA

- [1] Learning Management System ,MODUL 8 ALGORITMA SEARCHING.
- [2] Search algorithm, Wikipedia.
https://en.wikipedia.org/wiki/Search_algorithm