LAPORAN PRAKTIKUM
STURKTUR DATA DAN ALGORITMA

MODUL VIII
ALGORITMA SEARCHING

Disusun Oleh :
Muhammad Dani Ayubi
2311102003

Dosen Pengmapu:
Wahyu Andi Syahputra, S.pd.,M.Eng

LABORATORIUM MULTIMEDIA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO

PURWOKERTO
2024

BAB I
TUJUAN PRAKTIKUM

a. Menunjukkan beberapa algoritma dalam Pencarian.
b. Menunjukkan bahwa pencarian merupakan suatu persoalan yang bisa diselesaikan dengan beberapa algoritma yang berbeda.
c. Dapat memilih algoritma yang paling sesuai untuk menyelesaikan suatu permasalahan pemrograman

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:

1. 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 dimana data dalam array dibaca satu demi satuan 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

2. 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
    - 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

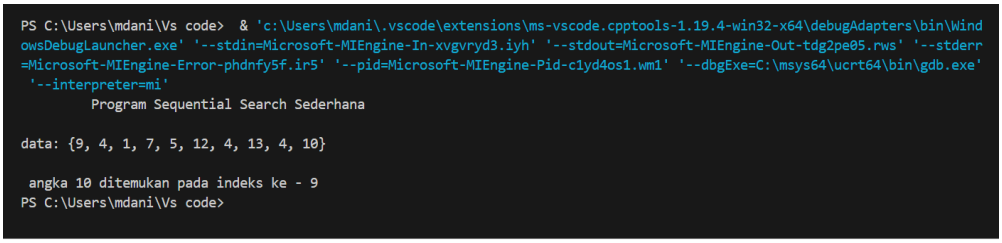
GUIDED

1. Guided 1

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



Deskripsi program

Kode di atas adalah implementasi konsep Sequential search program sequential search akan melakukan looping sebanyak n sampai data yang di cari ketemu disini kita mencari angka 10 maka akan di cari dari array indeks pertama sampai akhir

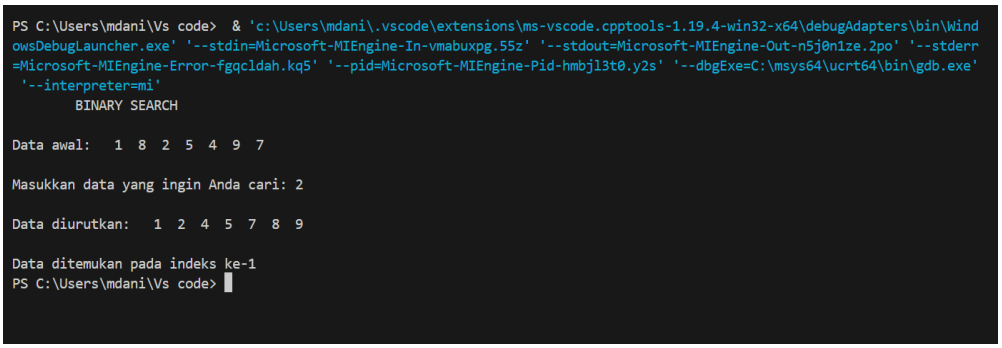
2. Guided 2

Source code

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

// Deklarasi array dan variabel untuk pencarian
int arrayData[7] = {1, 8, 2, 5, 4, 9, 7};
int cari;

// Fungsi selection sort untuk mengurutkan array
void selectionSort(int arr[], int n) {
    int temp, minIndex;

    for (int i = 0; i < n - 1; i++) {
        minIndex = i;

        for (int j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }

        // Tukar elemen
        temp = arr[i];
        arr[i] = arr[minIndex];
        arr[minIndex] = temp;
    }
}

// Fungsi binary search untuk mencari data dalam array yang
telah diurutkan
void binarySearch(int arr[], int n, int target) {
    int start = 0, end = n - 1, middle, found = 0;

    while (start <= end && found == 0) {
        middle = (start + end) / 2;

        if (arr[middle] == target) {
            found = 1;
        } else if (arr[middle] < target) {
            start = middle + 1;
        } else {
```

```
            end = middle - 1;
        }
    }

    if (found == 1) {
        cout << "\nData ditemukan pada indeks ke-" << middle <<
endl;
    } else {
        cout << "\nData tidak ditemukan\n";
    }
}

int main() {
    cout << "\tBINARY SEARCH" << endl;

    cout << "\nData awal: ";
    // Menampilkan data awal
    for (int i = 0; i < 7; i++) {
        cout << setw(3) << arrayData[i];
    }
    cout << endl;

    cout << "\nMasukkan data yang ingin Anda cari: ";
    cin >> cari;

    // Mengurutkan data dengan selection sort
    selectionSort(arrayData, 7);

    cout << "\nData diurutkan: ";
    // Menampilkan data setelah diurutkan
    for (int i = 0; i < 7; i++) {
        cout << setw(3) << arrayData[i];
    }
    cout << endl;

    // Melakukan binary search
    binarySearch(arrayData, 7, cari);

    return 0;
}
```

Screenshoot program



Deskripsi program

Kode diatas adalah contoh dari implementasi algoritma binary search, berbeda dengan algoritma sequential search yang mengecek data sampai ke n yang dilakukan binary search pertama kali adalah mengurutkan data nya terlebih dahulu baru kemduain mencari data yang ingin dicari

LATIHAN KELAS - UNGUIDED

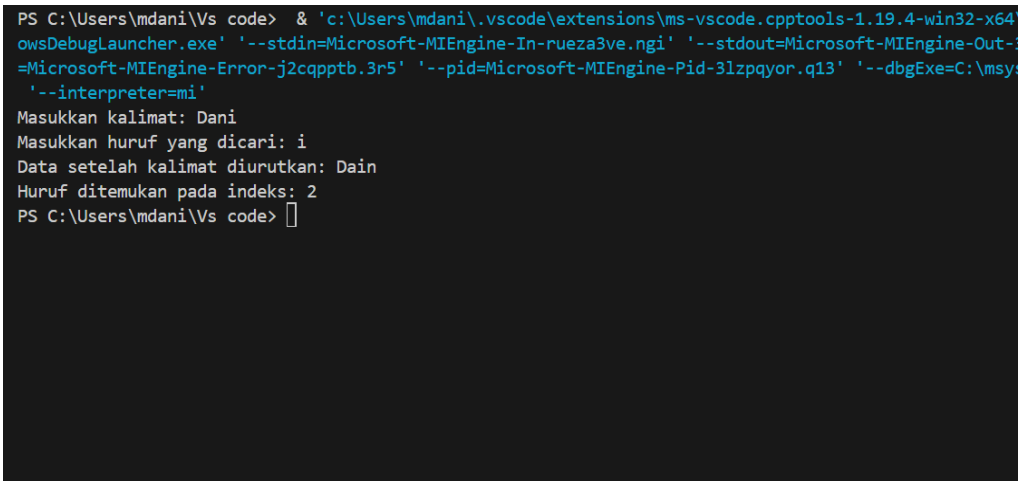
1. **Unguided 1**

   Source code

```cpp
#include <iostream>
using namespace std;
void selectionSort(string &arr, int n)
{
    int i, j, min;
    for (i = 0; i < n - 1; i++)
    {
        min = i;
        for (j = i + 1; j < n; j++)
        {
            if (arr[j] < arr[min])
            {
                min = j;
            }
        }
        char temp = arr[i];
        arr[i] = arr[min];
        arr[min] = temp;
    }
}
int binarySearch(string arr, int left, int right, char target)
{
    while (left <= right)
    {
        int mid = left + (right - left) / 2;
        if (arr[mid] == target)
            return mid;
        if (arr[mid] < target)
            left = mid + 1;
        else
            right = mid - 1;
    }
    return -1;
}
int main()
{
    string kalimat;
    char input;
```

```
    cout << "Masukkan kalimat: ";
    getline(cin, kalimat);
    cout << "Masukkan huruf yang dicari: ";
    cin >> input;
    selectionSort(kalimat, kalimat.size());
    int result = binarySearch(kalimat, 0, kalimat.size() - 1,
input);
    if (result == -1)
    {
        cout << "Huruf tidak ditemukan" << endl;
    }
    else
    {
        cout << "Data setelah kalimat diurutkan: " << kalimat
<< endl;
        cout << "Huruf ditemukan pada indeks: " << result
             << endl;
    }
    return 0;
}
```

Screenshoot program



Deskripsi program

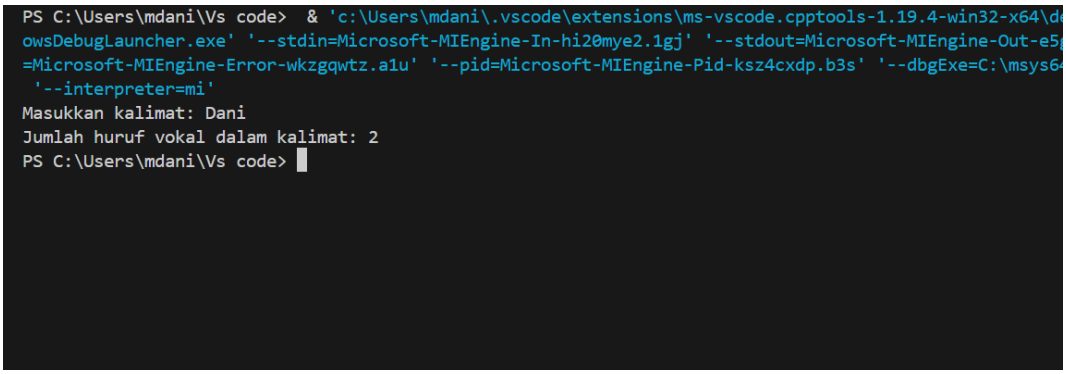
Program diatas adalah contoh implementasi dari alogritma binary search memungkinkan kita memasukkan sebuah kalimat dan huruf yang ingin dicari, kemudian mencari huruf tersebut dalam kalimat menggunakan algoritma Binary Search. Jika huruf ditemukan, program akan mencetak indeks pertama di mana huruf tersebut ditemukan, serta indeks tambahan jika ada huruf target yang berulang. Jika huruf tidak ditemukan, program akan mencetak pesan "Huruf tidak ditemukan."

2. Unguided 2

Source code

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

int countVowels(const char* sentence) {
int count = 0;
int length = strlen(sentence);

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

    }
    return count;
}

    int main() {
        const int MAX_SIZE = 100;
        char sentence[MAX_SIZE];
        cout << "Masukkan kalimat: ";
        cin.getline(sentence, MAX_SIZE);

        int vowelCount = countVowels(sentence);
        cout << "Jumlah huruf vokal dalam kalimat: " <<
vowelCount <<
        endl;

return 0;
}
```

Screenshoot program



Deskripsi program

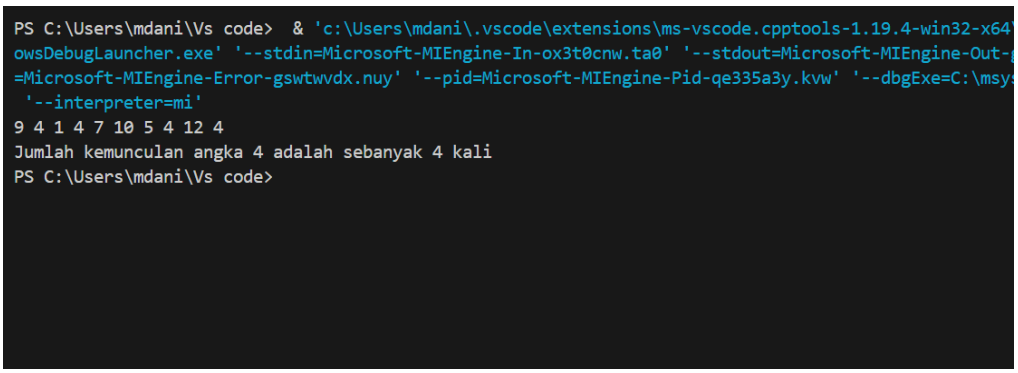
Fungsi ini digunakan untuk menghitung jumlah huruf vokal dalam kalimat. Fungsi menerima parameter sentence yang merupakan pointer ke array karakter yang berisi kalimat. Program menginisialisasi variabel count sebagai penghitung jumlah huruf vokal. Kemudian, program melakukan iterasi melalui setiap karakter dalam kalimat menggunakan loop for. Di dalam loop ini, program menggunakan fungsi tolower untuk mengubah huruf menjadi huruf kecil dan memeriksa apakah huruf tersebut adalah huruf vokal (a, e, i, o, atau u). Jika huruf adalah huruf vokal, program meningkatkan nilai count. Pada akhirnya, fungsi mengembalikan nilai count yang merupakan jumlah huruf vokal dalam kalimat.

3. Unguided 3

Source code

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

int sequentialSearch(int arr[], int leng, int target) {
    int count = 0;
    for (int i=0; i<leng; i++){
        if(arr[i] == target){
            count++;
            }

        }
return count;
}
int main() {
    int data[] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
    int leng = sizeof(data) / sizeof(*data);
    int target = 4;

    for(int i=0; i<leng; i++) {
        cout << data[i] << " ";
        }
        cout << endl;

        int result = sequentialSearch(data, leng, target);
        cout << "Jumlah kemunculan angka " << target << "
adalah sebanyak " << result << " kali" << endl;
}
```

Screenshoot program



Deskripsi program

Pada program diatas adalah contoh implementasi dari alogritma sequential search pada fungsi sequentialSearch: Ini adalah fungsi yang mengimplementasikan algoritma Sequential Search. Fungsi ini menerima array arr, ukuran array size, dan angka target yang ingin dicari target. Program melakukan iterasi melalui setiap elemen array menggunakan loop for. Di setiap iterasi, program memeriksa apakah elemen array saat ini sama dengan angka target. Jika sama, program meningkatkan nilai count yang merupakan penghitung jumlah kemunculan angka target. Pada akhirnya, fungsi mengembalikan nilai count.

BAB IV
KESIMPULAN

Berikut adalah kesimpulan yang dapat diambil dari laporan praktikum kali ini:

1) Algoritma searching digunakan untuk mencari elemen tertentu dalam kumpulan data atau struktur data.
2) Terdapat beberapa algoritma searching yang umum digunakan, antara lain Sequential Search, Binary Search, dan Interpolation Search.
3) Sequential Search adalah algoritma sederhana yang bekerja dengan memeriksa setiap elemen dalam urutan sekuensial. Cocok digunakan untuk kumpulan data yang belum terurut atau berukuran kecil.
4) Binary Search adalah algoritma efisien yang hanya dapat digunakan pada kumpulan data yang sudah terurut secara menaik atau menurun. Beroperasi dengan membagi kumpulan data menjadi dua bagian secara berulang.