LAPORAN PRAKTIKUM

MODUL VIII
ALGORITMA SEARCHING

Disusun oleh:
Geranada Saputra Priambudi
NIM: 2311102008

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. 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

Algoritma pencarian adalah serangkaian langkah atau instruksi yang digunakan untuk mencari suatu data atau informasi tertentu dalam struktur data seperti array, list, tree, atau database. Tujuan dari algoritma pencarian adalah untuk menemukan posisi atau keberadaan data yang dicari dalam struktur data dengan cara yang efisien dan efektif.

Algoritma pencarian digunakan pada berbagai jenis aplikasi, seperti mesin pencari, permainan, dan aplikasi bisnis. Pemahaman tentang algoritma pencarian sangat penting dalam ilmu komputer karena dapat membantu dalam pemecahan masalah dan pengembangan aplikasi.

Jenis Algoritma Pencarian

Ada beberapa jenis algoritma pencarian yang umum digunakan di antaranya:

1. Linear Search
2. Binary Search
3. Interpolation Search
4. Hash Search
5. Exponential Search
6. Jump Search
7. Fibonacci Search
8. Ternary Search

BAB III

GUIDED

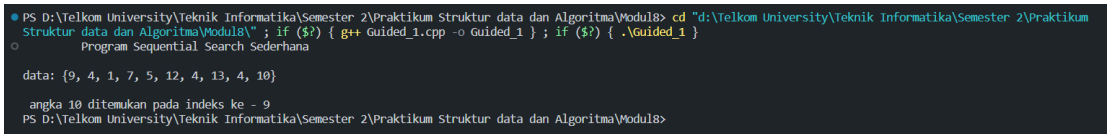
1. Guided 1

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

Program ini adalah implementasi sederhana dari algoritma Sequential Search dalam bahasa C++. Program ini mencari angka tertentu (dalam hal ini angka 10) dalam sebuah array dan menampilkan hasil pencarian.

2. Guided 2

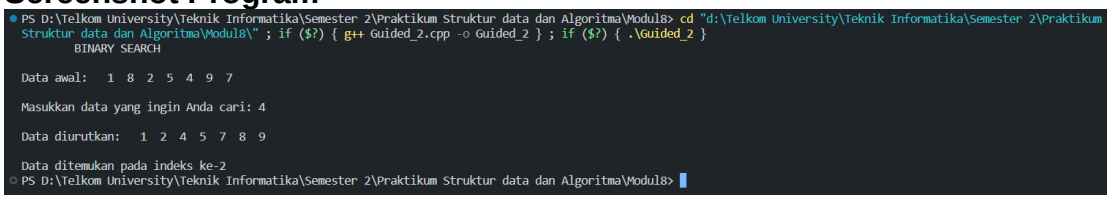
Source Code

```
#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;

// 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;
}
```

Screenshot Program



Deskripsi program

Program ini adalah implementasi gabungan dari algoritma Selection Sort dan Binary Search dalam bahasa C++. Program ini mengurutkan sebuah array menggunakan Selection Sort, kemudian mencari elemen tertentu dalam array yang sudah diurutkan menggunakan Binary Search.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

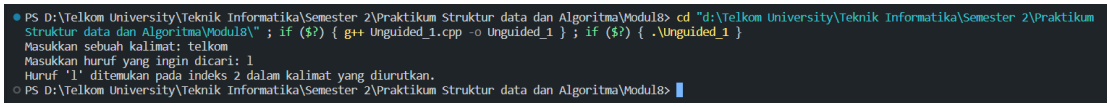
Source code

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

using namespace std;

// Fungsi untuk melakukan binary search
int binarySearch(const string &str, char target) {
    int left = 0;
    int right = str.length() - 1;

    while (left <= right) {
        int middle = left + (right - left) / 2;

        // Jika target ditemukan di tengah
        if (str[middle] == target) {
            return middle;
        }
        // Jika target lebih kecil dari elemen tengah, maka cari
di kiri
        else if (str[middle] > target) {
            right = middle - 1;
        }
        // Jika target lebih besar dari elemen tengah, maka cari
di kanan
        else {
            left = middle + 1;
        }
    }
```

```
    // Jika tidak ditemukan
    return -1;
}

int main() {
    string sentence;
    char target;

    // Input kalimat
    cout << "Masukkan sebuah kalimat: ";
    getline(cin, sentence);

    // Input huruf yang dicari
    cout << "Masukkan huruf yang ingin dicari: ";
    cin >> target;

    // Mengubah seluruh kalimat dan huruf target ke huruf kecil
untuk menghilangkan sensitivitas huruf besar-kecil
    transform(sentence.begin(),                 sentence.end(),
sentence.begin(), ::tolower);
    target = tolower(target);

    // Menghilangkan spasi dari kalimat
    sentence.erase(remove(sentence.begin(),  sentence.end(),  '
'), sentence.end());

    // Mengurutkan kalimat
    sort(sentence.begin(), sentence.end());

    // Mencari huruf menggunakan binary search
    int result = binarySearch(sentence, target);

    if (result != -1) {
```

```
        cout << "Huruf '" << target << "' ditemukan pada indeks
" << result << " dalam kalimat yang diurutkan." << endl;
    } else {
        cout << "Huruf '" << target << "' tidak ditemukan dalam
kalimat." << endl;
    }

    return 0;
}
```

Screenshoot program



Deskripsi program

Program ini adalah implementasi dari algoritma Binary Search untuk mencari karakter tertentu dalam sebuah kalimat yang diberikan oleh pengguna. Program ini bekerja dalam beberapa langkah utama: menerima input, menghilangkan spasi, mengubah ke huruf kecil, mengurutkan karakter, dan kemudian melakukan pencarian binary.

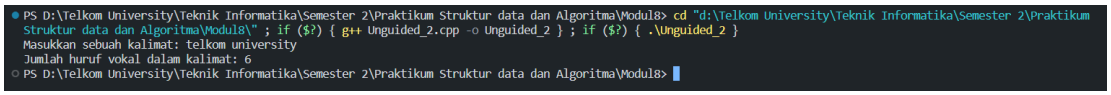
2. Unguided 2

Source code

```
#include <iostream>
#include <string>
#include <cctype>

using namespace std;

// Fungsi untuk memeriksa apakah sebuah karakter adalah huruf
vokal
bool isVowel(char ch) {
    ch = tolower(ch); // Mengubah karakter ke huruf kecil untuk
pemeriksaan
```

```
    return ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' ||
ch == 'u';
}

int main() {
    string sentence;
    int vowelCount = 0;

    // Input kalimat
    cout << "Masukkan sebuah kalimat: ";
    getline(cin, sentence);

    // Menghitung jumlah huruf vokal
    for (char ch : sentence) {
        if (isVowel(ch)) {
            vowelCount++;
        }
    }

    // Menampilkan hasil
    cout << "Jumlah huruf vokal dalam kalimat: " << vowelCount
<< endl;

    return 0;
}
```

Screenshot program



Deskripsi program

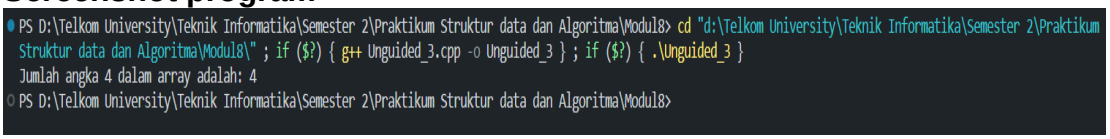
Program diatas adalah program sederhana dalam bahasa C++ untuk menghitung jumlah huruf vokal dalam sebuah kalimat yang diberikan oleh pengguna.

3. Unguided 3

Source code

```
#include <iostream>

using namespace std;

int countOccurrences(int arr[], int size, int target) {
    int count = 0;
    for (int i = 0; i < size; ++i) {
        if (arr[i] == target) {
            count++;
        }
    }
    return count;
}

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

    int count = countOccurrences(data, size, target);

    cout << "Jumlah angka " << target << " dalam array adalah:
" << count << endl;

    return 0;
}
```

Screenshot program



Deskripsi program

Program diatas adalah implementasi algoritma pencarian sekuensial (Sequential Search) untuk menghitung berapa kali sebuah angka tertentu muncul dalam sebuah array.

BAB IV

KESIMPULAN

Algoritma searching adalah algoritma yang digunakan untuk mencari elemen data tertentu dalam suatu kumpulan data. Algoritma searching memiliki berbagai macam jenis, seperti sequential search, binary search, dan hash table search. Masing-masing algoritma searching memiliki kelebihan dan kekurangannya sendiri, dan pilihan algoritma yang tepat tergantung pada jenis data dan ukuran kumpulan data. Tujuan praktikum struktur data algoritma searching adalah untuk memahami konsep dan implementasi berbagai macam algoritma searching.

Algoritma searching adalah alat yang penting untuk menyelesaikan berbagai macam masalah pemrograman. Praktikum struktur data algoritma searching memberikan pemahaman yang mendalam tentang konsep dan implementasi berbagai macam algoritma searching. Dengan memahami konsep dan implementasi algoritma searching, mahasiswa dapat memilih algoritma searching yang tepat untuk menyelesaikan masalah pemrograman dengan lebih efektif.