LAPORAN PRAKTIKUM
ALGORITMA & STRUKTUR DATA
MODUL 6

SEARCHING

Oleh:

Muhammad Irgi Fahrezha NIM. 2410817210005

PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS LAMBUNG MANGKURAT

JUNI 2025

LEMBAR PENGESAHAN

LAPORAN PRAKTIKUM ALGORITMA & STRUKTUR DATA

MODUL 6

Laporan Praktikum Algoritma & Sturktur Data Modul 6 : Searching ini disusun sebagai syarat lulus mata kuliah Praktikum Algoritma & Struktur Data. Laporan Praktikum ini dikerjakan oleh :

Nama Praktikan : Muhammad Irgi Fahrezha

NIM : 2410817210005

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Muhammad Fauzan Ahsani NIM. 2310817310009	Muti'a Maulida, S.Kom., M.TI. NIP. 198810272019032013

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

SOAL

Ketikkan source code berikut pada program IDE bahasa pemrograman C++ (**Gabungkan 2 code berikut menjadi 1 file (menu)**) :

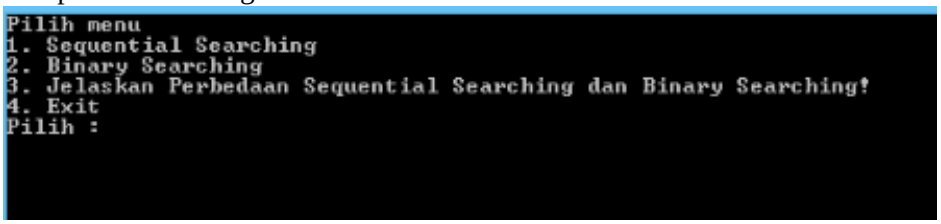
- Sequential Searching

```cpp
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

using namespace std;

int random(int bil)
{
    int jumlah = rand() % bil;
    return jumlah;
}

void randomize()
{
    srand(time(NULL));
}

void clrscr()
{
    system("cls");
}

int main()
{
    clrscr();
    int data[100];
    int cari = 20;
    int counter = 0;
    int flag = 0;
    int save;
    randomize();
    printf("generating 100 number . . .\n");
    for (int i = 0; i < 100; i++)
    {
        data[i] = random(100) + 1;
        printf("%d ", data[i]);
    }
    printf("\ndone.\n");

    for (int i = 0; i < 100; i++)
    {
        if (data[i] == cari)
        {
            counter++;
            flag = 1;
            save = i;
        }
    }

    if (flag == 1)
    {
        printf("Data ada, sebanyak %d!\n", counter);
        printf("pada indeks ke-%d", save);
    }
    else
    {
        printf("Data tidak ada!\n");
    }
}
```

- Binary Searching

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

int main()
{
    int n, kiri, kanan, tengah, temp, key;
    bool ketemu = false;

    cout << "Masukan jumlah data? ";
    cin >> n;
    int angka[n];
    for (int i = 0; i < n; i++)
    {
        cout << "Angka ke - [" << i << "] : ";
        cin >> angka[i];
    }

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n - 1; j++)
        {
            if (angka[j] > angka[j + 1])
            {
                temp = angka[j];
                angka[j] = angka[j + 1];
                angka[j + 1] = temp;
            }
        }
    }
    cout << "------------------------------------------------------------------------\n";
    cout << "Data yang telah diurutkan adalah:\n";
    for (int i = 0; i < n; i++)
    {
        cout << angka[i] << " ";
    }
    cout << "\n------------------------------------------------------------------------\n";
    cout << "Masukan angka yang dicari: ";
    cin >> key;

    kiri = 0;
    kanan = n - 1;
    while (kiri <= kanan)
    {
        tengah = (kiri + kanan) / 2;
        if (key == angka[tengah])
        {
            ketemu = true;
            break;
        }
        else if (key < angka[tengah])
        {
            kanan = tengah - 1;
        }
        else
        {
            kiri = tengah + 1;
        }
    }
    if (ketemu == true)
    {
        cout << "Angka ditemukan! ";
    }
    else
        cout << "Angka tidak ditemukan!";
    return 0;
}
```

- Tampilan Menu Program



Jelaskan perbedaan Sequential Searching dan Binary Searching beserta kelebihan dan kekurangan masing-masing!

A. Source Code

Tabel 1 Tabel Source Code Searching

```
 1  #include <iostream>
 2  #include <conio.h>
 3  #include <random>
 4  #include <vector>
 5  #include <algorithm>
 6
 7  using namespace std;
 8
 9
10  void sequentialSearch(){
11      int target;
12      vector<int> nums(100);
13          mt19937_64 rng(random_device{}());
14          uniform_int_distribution<int> dist(1, 50);
15          for (auto &val: nums)
16          {
17              val = dist(rng);
18          }
19
20
21      cout << "\nDaftar angka:\n";
22      for (int i = 0; i < nums.size(); ++i) {
23          cout << "[" << i << "]" << nums[i] << " ";
24      }
25      cout << "\n\n";
26
27      cout << "Masukkan angka yang ingin dicari : ";
28      cin >> target;
29      cout << endl;
30      cout << "Hasil Pencarian : " << endl;
31
32      bool found = false;
33      for (int i = 0; i < nums.size(); ++i) {
34          if (nums[i] == target) {
35              cout << "Angka " << target << " ditemukan
    di indeks ke-" << "[" << i << "]" << ".\n";
36              found = true;
37          }
38      }
39
40      if (!found) {
41          cout << "Angka " << target << " tidak
    ditemukan dalam daftar.\n";
42      }
43  }
44
45  void binarySearch(){
46      int target, size, temp;
47      cout << "Masukkan ukuran vector yang ingin dibuat
    : ";
```

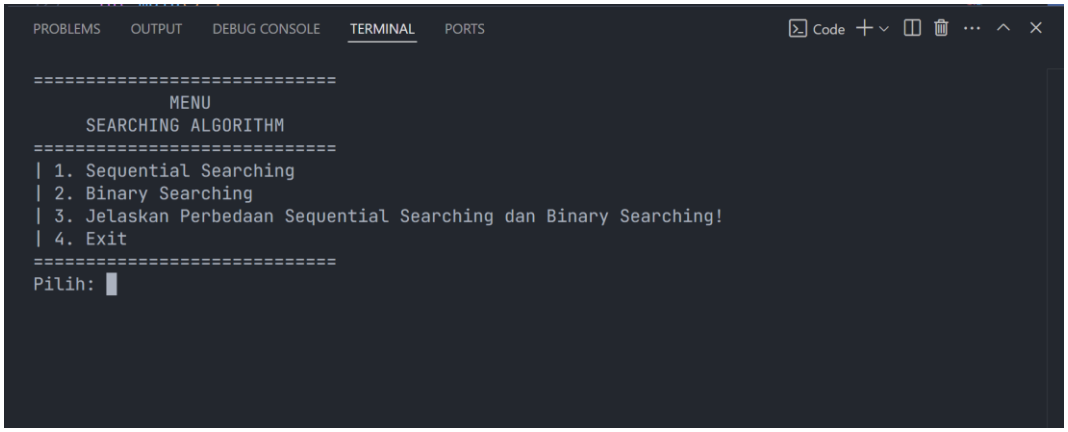
```
48      cin >> size;
49      vector<int> nums(size);
50          mt19937_64 rng(random_device{}());
51          uniform_int_distribution<int> dist(1, 100);
52          for (auto &val: nums)
53          {
54              val = dist(rng);
55          }
56
57      vector<int> angka = nums;
58      for(int i = 0; i < size; i++){
59          for(int j = 0; j < size - 1 ; j++){
60              if (angka[j] > angka[j + 1]){
61                  temp = angka[j];
62                  angka[j] = angka[j + 1];
63                  angka[j + 1] = temp;
64              }
65          }
66      }
67
68      cout << "\nDaftar angka setelah diurutkan:\n";
69      for (int i = 0; i < size; i++) {
70          cout << "[" << i << "]" << angka[i] << " ";
71      }
72      cout << "\n\n";
73
74      cout << "Masukkan angka yang ingin dicari : ";
75      cin >> target;
76      cout << endl;
77
78      cout << "Hasil Pencarian : " << endl;
79      int kiri = 0, kanan = size - 1;
80      bool found = false;
81      while (kiri <= kanan) {
82          int tengah = (kiri + kanan) / 2;
83          if (angka[tengah] == target) {
84              cout << "Angka " << target << " ditemukan
    di indeks ke-" << "[" << tengah << "]"
85                  << " (dalam daftar terurut).\n";
86              found = true;
87              break;
88          } else if (angka[tengah] < target) {
89              kiri = tengah + 1;
90          } else {
91              kanan = tengah - 1;
92          }
93      }
94
95      if (!found) {
96          cout << "Angka " << target << " tidak
    ditemukan dalam daftar.\n";
97      }
98  }
```

```
 99
100 void clearScreen(){
101     system("cls"); // Gunakan "clear" jika di
    Linux/Mac
102 }
103
104 void explain(){
105     cout << "\n=== Penjelasan ===\n";
106
107     cout << "1. Sequential Search (Pencarian
    Sekuensial):\n";
108     cout << "   - Metode pencarian yang dilakukan
    dengan memeriksa setiap elemen secara berurutan dari
    awal hingga akhir.\n";
109     cout << "   - Tidak memerlukan data dalam kondisi
    terurut, sehingga dapat diterapkan pada data acak.\n";
110     cout << "   - Cocok digunakan untuk dataset kecil
    atau jika data tidak disortir.\n";
111     cout << "   - Mudah diimplementasikan dan tidak
    membutuhkan struktur data tambahan.\n";
112     cout << "   - Kompleksitas waktu dalam kasus
    terburuk adalah O(n), di mana n adalah jumlah
    elemen.\n";
113     cout << "   - Kekurangannya: Tidak efisien untuk
    dataset yang besar karena memerlukan banyak
    perbandingan.\n\n";
114
115     cout << "2. Binary Search (Pencarian Biner):\n";
116     cout << "   - Metode pencarian yang bekerja dengan
    cara membagi dua bagian data yang sudah terurut.\n";
117     cout << "   - Setiap langkah membandingkan elemen
    tengah dengan elemen yang dicari:\n";
118     cout << "     Jika elemen tengah lebih kecil,
    pencarian dilanjutkan ke separuh kanan,\n";
119     cout << "     jika lebih besar, ke separuh
    kiri.\n";
120     cout << "   - Sangat efisien untuk dataset besar
    yang sudah dalam kondisi terurut.\n";
121     cout << "   - Kompleksitas waktu dalam kasus
    terburuk adalah O(log n).\n";
122     cout << "   - Kekurangannya: Hanya dapat digunakan
    jika data sudah terurut, jika belum maka perlu
    proses\n";
123     cout << "     sorting terlebih dahulu.\n";
124     cout << "   - Implementasi sedikit lebih kompleks
    dibandingkan Sequential Search.\n";
125 }
126
127 int main() {
128     int opt;
129     do {
130         cout << "============================== " <<
    endl;
```

```
131         cout << "              MENU              " <<
    endl;
132         cout << "     SEARCHING ALGORITHM      " <<
    endl;
133         cout << "============================== " <<
    endl;
134         cout << "| 1. Sequential Searching" << endl;
135         cout << "| 2. Binary Searching" << endl;
136         cout << "| 3. Jelaskan Perbedaan Sequential
    Searching dan Binary Searching!" << endl;
137         cout << "| 4. Exit" << endl;
138         cout << "============================== " <<
    endl;
139         cout << "Pilih: ";
140         cin >> opt;
141
142         switch (opt) {
143             case 1: {
144                 sequentialSearch();
145                 break;
146             }
147
148             case 2: {
149                 binarySearch();
150                 break;
151             }
152
153             case 3:
154                 explain();
155                 break;
156
157             case 4:
158                 cout << "\nTERIMA KASIH!" << endl;
159                 cout << "Programme was made by
    Muhammad Irgi Fahrezha (2410817210005)" << endl;
160                 break;
161
162             default:
163                 cout << endl;
164                 cout << "Opsi tidak terdefinisi, mohon
    masukkan ulang opsi." << endl;
165                 break;
166         }
167
168         if (opt != 4) {
169             cout << "\nTekan sembarang tombol untuk
    melanjutkan...";
170             getch();
171             clearScreen();
172         }
173
174     } while (opt != 4);
175
```

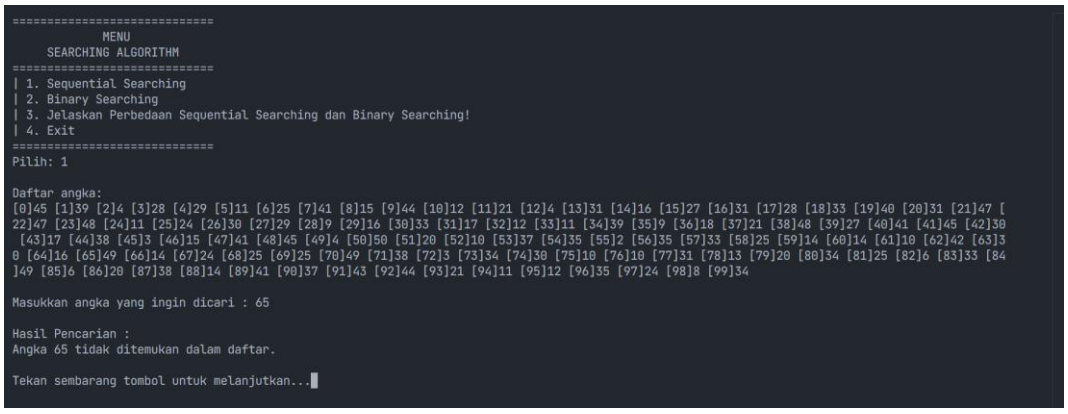
```
176         return 0;
177 }
```

B. Output

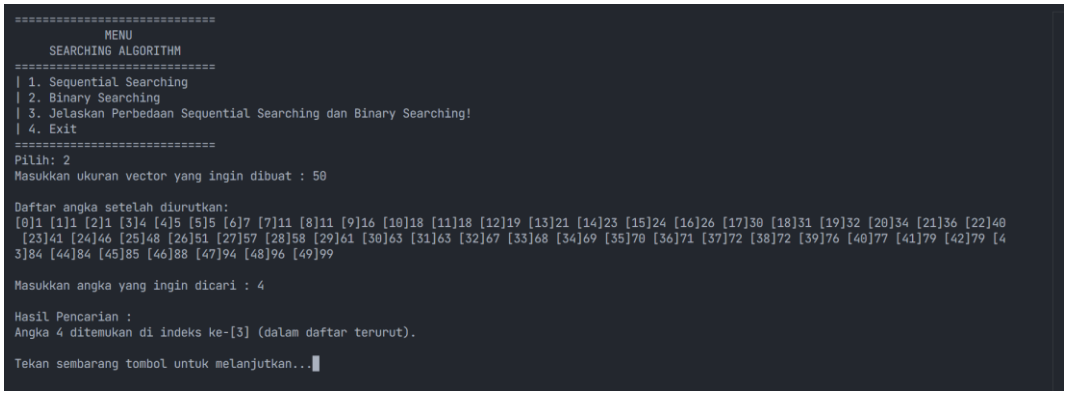


Gambar 1 Tampilan Menu

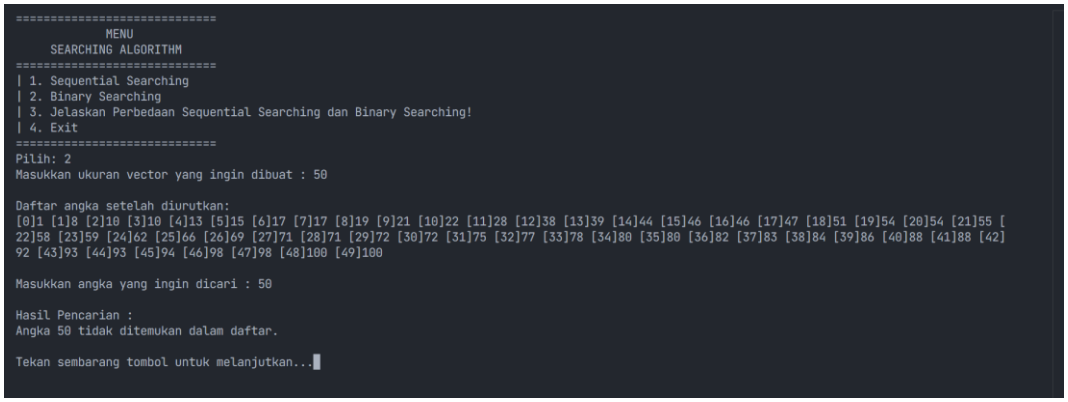
Gambar 2 Tampilan Menu Sequential Searching



Gambar 3 Menu Sequential Searching (Tidak Ada Angka yang Dicari)

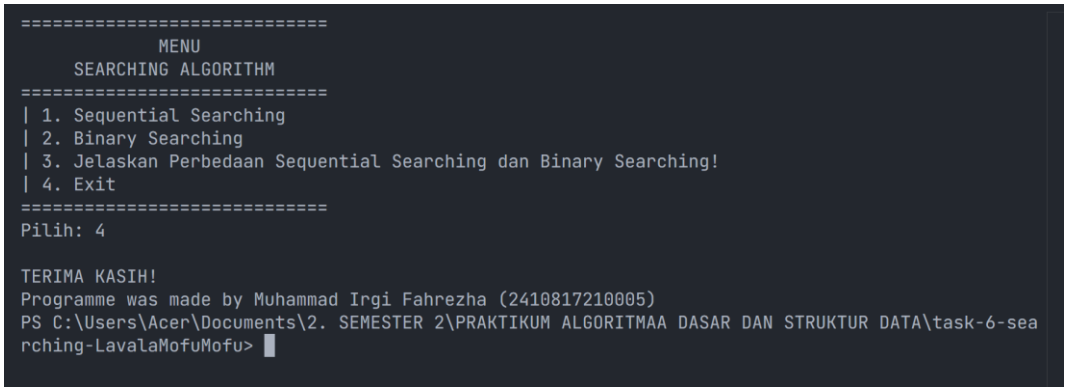


Gambar 4 Tampilan Menu Binary Searching



Gambar 5 Menu Binary Searching (Tidak Ada Angka yang Dicari)

Gambar 6 Tampilan Menu Penjelasan



Gambar 7 Tampilan Menu Exit

C. Pembahasan

Bagian Header

```
#include <iostream>
#include <conio.h>
#include <random>
#include <vector>
#include <algorithm>
using namespace std;
```

Bagian ini mengimpor pustaka-pustaka yang diperlukan:

- `iostream`: untuk input dan output standar (cin, cout).
- `conio.h`: menyediakan fungsi getch() untuk menunggu input tombol
- `random`: untuk membuat angka acak dengan generator modern (mt19937_64).
- `vector`: untuk menggunakan struktur data vector.
- `algorithm`: meskipun tidak digunakan eksplisit di sini, biasanya diperlukan untuk algoritma STL seperti sort.

`using namespace std` digunakan agar tidak perlu menulis std:: di depan fungsi-fungsi standar.

Fungsi Sequential Search

```
void sequentialSearch(){
    int target;
    vector<int> nums(100);
```
// Membuat vector berisi 100 elemen

```
mt19937_64 rng(random_device{}()); // Inisialisasi RNG
uniform_int_distribution<int> dist(1, 50); // Rentang
```
angka 1-50
```
for (auto &val: nums) {
    val = dist(rng); // Mengisi vector dengan angka acak
}
```
Vektor nums diisi dengan 100 angka acak dari 1 sampai 50. Angka dihasilkan dengan generator acak mt19937_64.

```
cout << "\nDaftar angka:\n";
    for (int i = 0; i < nums.size(); ++i) {
        cout << "[" << i << "]" << nums[i] << " ";
    }
```
Menampilkan isi vektor beserta indeksnya ke layar.

```
cout << "\n\nMasukkan angka yang ingin dicari : ";
cin >> target;
cout << endl;
cout << "Hasil Pencarian : " << endl;
Pengguna diminta memasukkan angka yang ingin dicari.
bool found = false;
    for (int i = 0; i < nums.size(); ++i) {
        if (nums[i] == target) {
            cout << "Angka " << target << " ditemukan
di indeks ke-" << "[" << i << "]" << ".\n";
            found = true;
        }
    }
```
Loop mencocokkan satu per satu elemen dalam vektor. Jika ditemukan, indeksnya ditampilkan dan found diubah menjadi true.

```
if (!found) {
```

```
            cout << "Angka " << target << " tidak
ditemukan dalam daftar.\n";
      }
}
```

Jika angka tidak ditemukan, ditampilkan pesan bahwa angka tidak ditemukan dalam daftar.

Fungsi Binary Search

```
void binarySearch(){
     int target, size, temp;
     cout << "Masukkan ukuran vector yang ingin dibuat : ";
     cin >> size;
     vector<int> nums(size);
```

Pengguna memasukkan ukuran vektor. Vektor nums dibuat dengan ukuran tersebut.

```
mt19937_64 rng(random_device{}());
uniform_int_distribution<int> dist(1, 100);
for (auto &val: nums) {
            val = dist(rng);
      }
```

Vektor diisi dengan angka acak dari 1 hingga 100.

```
vector<int> angka = nums;
      for(int i = 0; i < size; i++){
            for(int j = 0; j < size - 1 ; j++){
                  if (angka[j] > angka[j + 1]){
                        temp = angka[j];
                        angka[j] = angka[j + 1];
                        angka[j + 1] = temp;
                  }
            }
      }
```

Vektor angka adalah salinan dari nums. Data disortir menggunakan Bubble Sort.

```
cout << "\nDaftar angka setelah diurutkan:\n";
     for (int i = 0; i < size; i++) {
          cout << "[" << i << "]" << angka[i] << " ";
     }
```

Menampilkan vektor yang telah diurutkan.

```
cout << "\n\nMasukkan angka yang ingin dicari : ";
cin >> target;
```

Pengguna diminta memasukkan angka yang akan dicari.

```
 int kiri = 0, kanan = size - 1;
      bool found = false;
      while (kiri <= kanan) {
            int tengah = (kiri + kanan) / 2;
            if (angka[tengah] == target) {
                  cout << "Angka " << target << " ditemukan
di indeks ke-" << "[" << tengah << "]"
                        << " (dalam daftar terurut).\n";
                  found = true;
                  break;
            } else if (angka[tengah] < target) {
                  kiri = tengah + 1;
            } else {
                  kanan = tengah - 1;
            }
      }
```

Pencarian biner dilakukan: membandingkan nilai tengah dengan target, dan mempersempit ruang pencarian ke kiri atau kanan.

```
if (!found) {
            cout << "Angka " << target << " tidak
ditemukan dalam daftar.\n";
```

```
    }
}
```

Jika tidak ditemukan, ditampilkan pesan bahwa angka tidak ada.

Fungsi Clear Screen

```
void clearScreen(){
    system("cls");
}
```

Membersihkan layar. Khusus Windows (cls). Untuk Linux/Mac bisa diganti system("clear").

Fungsi Explain

```
void explain(){
    cout << "\n=== Penjelasan ===\n";
    ...
}
```

Menampilkan penjelasan perbedaan antara Sequential Search dan Binary Search, mulai dari metode, kompleksitas waktu, syarat data, serta kelebihan dan kekurangan masing-masing.

Fungsi Main

Fungsi utama main() menampilkan menu interaktif kepada pengguna dengan empat pilihan: menjalankan Sequential Search, menjalankan Binary Search, membaca penjelasan perbedaan kedua metode pencarian, dan keluar dari program. Fungsi ini menggunakan perulangan do-while untuk terus menampilkan menu hingga pengguna memilih opsi keluar. Setiap input pengguna diproses menggunakan switch-case, dan setelah menjalankan salah satu fungsi (selain keluar), program akan menunggu input tombol dari pengguna sebelum membersihkan layar dan kembali ke menu. Fungsi ini juga mencantumkan nama pembuat program pada akhir sesi.

TAUTAN GIT

https://github.com/DSA25-ULM/task-6-searching-LavalaMofuMofu