

**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 & Struktur 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

LEMBAR PENGESAHAN	ii
DAFTAR ISI.....	iii
DAFTAR GAMBAR	iv
DAFTAR TABEL.....	v
SOAL	6
A. Source Code.....	8
B. Output	12
C. Pembahasan	14
TAUTAN GIT.....	19

DAFTAR GAMBAR

Gambar 1 Tampilan Menu	12
Gambar 2 Tampilan Menu Sequential Searching	12
Gambar 3 Menu Sequential Searching (Tidak Ada Angka yang Dicari).....	12
Gambar 4 Tampilan Menu Binary Searching	13
Gambar 5 Menu Binary Searching (Tidak Ada Angka yang Dicari).....	13
Gambar 6 Tampilan Menu Penjelasan	13
Gambar 7 Tampilan Menu Exit	14

DAFTAR TABEL

Tabel 1 Tabel Source Code Searching	8
---	---

SOAL

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

- Sequential Searching

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <time.h>
4
5  using namespace std;
6
7  int random(int bil)
8  {
9      int jumlah = rand() % bil;
10     return jumlah;
11 }
12
13 void randomize()
14 {
15     srand(time(NULL));
16 }
17
18 void clrscr()
19 {
20     system("cls");
21 }
22
23 int main()
24 {
25     clrscr();
26     int data[100];
27     int cari = 20;
28     int counter = 0;
29     int flag = 0;
30     int save;
31     randomize();
32     printf("generating 100 number . . .\n");
33     for (int i = 0; i < 100; i++)
34     {
35         data[i] = random(100) + 1;
36         printf("%d ", data[i]);
37     }
38     printf("\ndone.\n");
39
40     for (int i = 0; i < 100; i++)
41     {
42         if (data[i] == cari)
43         {
44             counter++;
45             flag = 1;
46             save = i;
47         }
48     }
49
50     if (flag == 1)
51     {
52         printf("Data ada, sebanyak %d!\n", counter);
53         printf("pada indeks ke-%d", save);
54     }
55     else
56     {
57         printf("Data tidak ada!\n");
58     }
59 }
60
```

- Binary Searching

```

1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n, kiri, kanan, tengah, temp, key;
7      bool ketemu = false;
8
9      cout << "Masukan jumlah data? ";
10     cin >> n;
11     int angka[n];
12     for (int i = 0; i < n; i++)
13     {
14         cout << "Angka ke - [" << i << "] : ";
15         cin >> angka[i];
16     }
17
18     for (int i = 0; i < n; i++)
19     {
20         for (int j = 0; j < n - 1; j++)
21         {
22             if (angka[j] > angka[j + 1])
23             {
24                 temp = angka[j];
25                 angka[j] = angka[j + 1];
26                 angka[j + 1] = temp;
27             }
28         }
29     }
30     cout << "-----\n";
31     cout << "Data yang telah diurutkan adalah:\n";
32     for (int i = 0; i < n; i++)
33     {
34         cout << angka[i] << " ";
35     }
36     cout << "\n-----\n";
37     cout << "Masukan angka yang dicari: ";
38     cin >> key;
39
40     kiri = 0;
41     kanan = n - 1;
42     while (kiri <= kanan)
43     {
44         tengah = (kiri + kanan) / 2;
45         if (key == angka[tengah])
46         {
47             ketemu = true;
48             break;
49         }
50         else if (key < angka[tengah])
51         {
52             kanan = tengah - 1;
53         }
54         else
55         {
56             kiri = tengah + 1;
57         }
58     }
59     if (ketemu == true)
60     {
61         cout << "Angka ditemukan! ";
62     }
63     else
64     {
65         cout << "Angka tidak ditemukan!";
66     }
67     return 0;
68 }

```

- Tampilan Menu Program

```

Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih :

```

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
36	di indeks ke-" << "[" << i << "]" << ".\n";
37	found = true;
38	}
39	}
40	if (!found) {
41	cout << "Angka " << target << " tidak
42	ditemukan dalam daftar.\n";
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         mtl937_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

```

=====
MENU
SEARCHING ALGORITHM
=====
| 1. Sequential Searching
| 2. Binary Searching
| 3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
| 4. Exit
=====
Pilih: 

```

Gambar 1 Tampilan Menu

```

=====
MENU
SEARCHING ALGORITHM
=====
| 1. Sequential Searching
| 2. Binary Searching
| 3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
| 4. Exit
=====
Pilih: 1

Daftar angka:
[0]19 [1]49 [2]46 [3]17 [4]26 [5]34 [6]30 [7]12 [8]32 [9]19 [10]37 [11]6 [12]13 [13]48 [14]26 [15]21 [16]48 [17]37 [18]20 [19]31 [20]17 [21]9 [
22]48 [23]17 [24]25 [25]37 [26]43 [27]35 [28]13 [29]11 [30]10 [31]9 [32]26 [33]23 [34]6 [35]45 [36]35 [37]8 [38]38 [39]16 [40]43 [41]39 [42]5 [
43]6 [44]27 [45]47 [46]22 [47]6 [48]47 [49]39 [50]23 [51]22 [52]9 [53]30 [54]25 [55]46 [56]30 [57]30 [58]14 [59]27 [60]26 [61]9 [62]5 [63]17 [6
4]9 [65]27 [66]42 [67]40 [68]43 [69]35 [70]43 [71]11 [72]46 [73]34 [74]38 [75]46 [76]41 [77]35 [78]40 [79]23 [80]20 [81]34 [82]28 [83]38 [84]50
[85]5 [86]10 [87]16 [88]10 [89]7 [90]32 [91]16 [92]6 [93]2 [94]5 [95]22 [96]37 [97]1 [98]17 [99]35

Masukkan angka yang ingin dicari : 26

Hasil Pencarian :
Angka 26 ditemukan di indeks ke-[4].
Angka 26 ditemukan di indeks ke-[14].
Angka 26 ditemukan di indeks ke-[32].
Angka 26 ditemukan di indeks ke-[60].

Tekan sembarang tombol untuk melanjutkan...

```

Gambar 2 Tampilan Menu Sequential Searching

```

=====
MENU
SEARCHING ALGORITHM
=====
| 1. Sequential Searching
| 2. Binary Searching
| 3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
| 4. Exit
=====
Pilih: 1

Daftar angka:
[0]45 [1]39 [2]4 [3]28 [4]29 [5]11 [6]25 [7]41 [8]15 [9]44 [10]12 [11]21 [12]4 [13]31 [14]16 [15]27 [16]31 [17]28 [18]33 [19]40 [20]31 [21]47 [
22]47 [23]48 [24]11 [25]24 [26]30 [27]29 [28]9 [29]16 [30]33 [31]17 [32]12 [33]11 [34]39 [35]9 [36]18 [37]21 [38]48 [39]27 [40]41 [41]45 [42]30
[43]17 [44]38 [45]3 [46]15 [47]41 [48]45 [49]4 [50]50 [51]20 [52]10 [53]37 [54]35 [55]2 [56]35 [57]33 [58]25 [59]14 [60]14 [61]10 [62]42 [63]3
0 [64]16 [65]49 [66]14 [67]24 [68]25 [69]25 [70]49 [71]38 [72]3 [73]34 [74]30 [75]10 [76]10 [77]31 [78]13 [79]20 [80]34 [81]25 [82]6 [83]33 [84
]49 [85]6 [86]20 [87]38 [88]14 [89]41 [90]37 [91]43 [92]44 [93]21 [94]11 [95]12 [96]35 [97]24 [98]8 [99]34

Masukkan angka yang ingin dicari : 65

Hasil Pencarian :
Angka 65 tidak ditemukan dalam daftar.

Tekan sembarang tombol untuk melanjutkan...

```

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

```

=====
MENU
SEARCHING ALGORITHM
=====
| 1. Sequential Searching
| 2. Binary Searching
| 3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
| 4. Exit
=====
Pilih: 2
Masukkan ukuran vector yang ingin dibuat : 50

Daftar angka setelah diurutkan:
[0]1 [1]1 [2]1 [3]4 [4]5 [5]5 [6]7 [7]11 [8]11 [9]16 [10]18 [11]18 [12]19 [13]21 [14]23 [15]24 [16]26 [17]30 [18]31 [19]32 [20]34 [21]36 [22]40
[23]41 [24]46 [25]48 [26]51 [27]57 [28]58 [29]61 [30]63 [31]63 [32]67 [33]68 [34]69 [35]70 [36]71 [37]72 [38]72 [39]76 [40]77 [41]79 [42]79 [4
3]84 [44]84 [45]85 [46]88 [47]94 [48]96 [49]99

Masukkan angka yang ingin dicari : 4

Hasil Pencarian :
Angka 4 ditemukan di indeks ke-[3] (dalam daftar terurut).

Tekan sembarang tombol untuk melanjutkan...

```

Gambar 4 Tampilan Menu Binary Searching

```

=====
MENU
SEARCHING ALGORITHM
=====
| 1. Sequential Searching
| 2. Binary Searching
| 3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
| 4. Exit
=====
Pilih: 2
Masukkan ukuran vector yang ingin dibuat : 50

Daftar angka setelah diurutkan:
[0]1 [1]8 [2]10 [3]10 [4]13 [5]15 [6]17 [7]17 [8]19 [9]21 [10]22 [11]28 [12]38 [13]39 [14]44 [15]46 [16]46 [17]47 [18]51 [19]54 [20]54 [21]55 [
22]58 [23]59 [24]62 [25]66 [26]69 [27]71 [28]71 [29]72 [30]72 [31]75 [32]77 [33]78 [34]80 [35]80 [36]82 [37]83 [38]84 [39]86 [40]88 [41]88 [42]
92 [43]93 [44]93 [45]94 [46]98 [47]98 [48]100 [49]100

Masukkan angka yang ingin dicari : 50

Hasil Pencarian :
Angka 50 tidak ditemukan dalam daftar.

Tekan sembarang tombol untuk melanjutkan...

```

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

```

=====
MENU
SEARCHING ALGORITHM
=====
| 1. Sequential Searching
| 2. Binary Searching
| 3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
| 4. Exit
=====
Pilih: 3

=== Penjelasan ===
1. Sequential Search (Pencarian Sekuensial):
- Metode pencarian yang dilakukan dengan memeriksa setiap elemen secara berurutan dari awal hingga akhir.
- Tidak memerlukan data dalam kondisi terurut, sehingga dapat diterapkan pada data acak.
- Cocok digunakan untuk dataset kecil atau jika data tidak disortir.
- Mudah diimplementasikan dan tidak membutuhkan struktur data tambahan.
- Kompleksitas waktu dalam kasus terburuk adalah  $O(n)$ , di mana  $n$  adalah jumlah elemen.
- Kekurangannya: Tidak efisien untuk dataset yang besar karena memerlukan banyak perbandingan.

2. Binary Search (Pencarian Biner):
- Metode pencarian yang bekerja dengan cara membagi dua bagian data yang sudah terurut.
- Setiap langkah membandingkan elemen tengah dengan elemen yang dicari:
  - Jika elemen tengah lebih kecil, pencarian dilanjutkan ke separuh kanan,
  - jika lebih besar, ke separuh kiri.
- Sangat efisien untuk dataset besar yang sudah dalam kondisi terurut.
- Kompleksitas waktu dalam kasus terburuk adalah  $O(\log n)$ .
- Kekurangannya: Hanya dapat digunakan jika data sudah terurut, jika belum maka perlu proses
  sorting terlebih dahulu.
- Implementasi sedikit lebih kompleks dibandingkan Sequential Search.

Tekan sembarang tombol untuk melanjutkan...

```

Gambar 6 Tampilan Menu Penjelasan

```
=====
      MENU
    SEARCHING ALGORITHM
=====
| 1. Sequential Searching
| 2. Binary Searching
| 3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
| 4. Exit
=====
Pilih: 4

TERIMA KASIH!
Programme was made by Muhammad Irgi Fahrezha (2410817210005)
PS C:\Users\Acer\Documents\2. SEMESTER 2\PRAKTIKUM ALGORITMAA DASAR DAN STRUKTUR DATA\task-6-sea
rching-LavalaMofuMofu> █
```

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>