LAPORAN PRAKTIKUM
ALGORITMA & STRUKTUR DATA
MODUL 6

SEARCHING

Oleh:

Noor Khalisa NIM. 2410817220012

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 : Noor Khalisa
NIM : 24101817220012

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

```
#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:

```
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 Source Code Soal

```
 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(const vector<int> &nums, int
    target){
11      int count = 0;
12
13      cout << "\nMenjalankan Sequential Search..." <<
    endl;
14
15      for (int i = 0; i < nums.size(); ++i) {
16          if (nums[i] == target) {
17              cout << "Angka " << target << " ditemukan
    pada indeks ke-" << i << "." << endl;
18              count++;
19          }
20      }
21
22      if (count == 0) {
23          cout << "Angka " << target << " tidak
    ditemukan dalam daftar." << endl;
24      } else {
25          cout << "Total kemunculan: " << count <<
    endl;
26      }
27  }
28
29  void binarySearch(const vector<int> &nums, int
    target){
30      vector<int> sortedNums = nums;
31      sort(sortedNums.begin(), sortedNums.end());
32
33      cout << "\nMenjalankan Binary Search..." << endl;
```

```
34      cout << "Daftar angka yang diurutkan:\n";
35      for (int i = 0; i < sortedNums.size(); ++i) {
36          cout << "[" << i << "] " << sortedNums[i] <<
" ";
37          if ((i + 1) % 10 == 0) cout << endl;
38      }
39
40      int left = 0;
41      int right = sortedNums.size() - 1;
42      bool found = false;
43
44      while (left <= right) {
45          int mid = left + (right - left) / 2;
46          if (sortedNums[mid] == target) {
47              cout << "\nAngka " << target << "
ditemukan pada indeks ke-" << mid << " (dalam daftar
yang diurutkan)." << endl;
48              found = true;
49              break;
50          } else if (sortedNums[mid] < target) {
51              left = mid + 1;
52          } else {
53              right = mid - 1;
54          }
55      }
56
57      if (!found) {
58          cout << "\nAngka " << target << " tidak
ditemukan dalam daftar." << endl;
59      }
60  }
61
62  /// use this to clear screen, change if necessary
63  void clearScreen(){
64      system("cls");
65  }
66
67  // use this to  print your answer
68  void explain(){
69      cout << "\n--- Perbedaan Sequential Searching dan
Binary Searching ---" << endl;
70      cout << "Sequential Searching (Pencarian
Berurutan):" << endl;
```

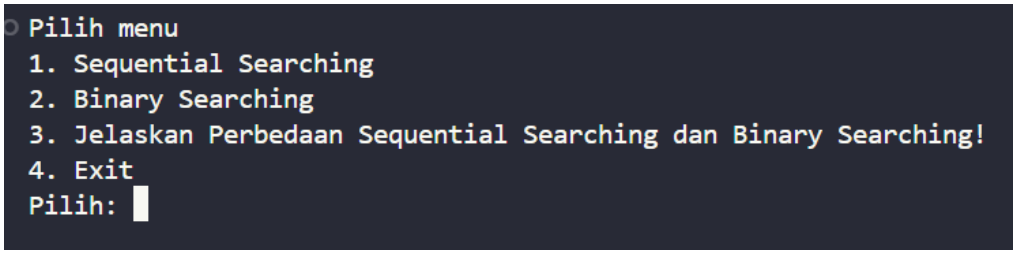
```
71      cout << "1. Cara Kerja: Metode ini mencari elemen secara berurutan, satu per satu, dari awal hingga akhir daftar." << endl;
72      cout << "2. Persyaratan Data: Tidak memerlukan data yang diurutkan. Dapat bekerja pada daftar angka yang tidak terurut." << endl;
73      cout << "3. Kecepatan: Relatif lebih lambat, terutama untuk daftar yang besar, karena dalam kasus terburuk harus memeriksa setiap elemen." << endl;
74      cout << "4. Kompleksitas Waktu: Memiliki kompleksitas waktu rata-rata O(n), di mana 'n' adalah jumlah elemen dalam daftar." << endl;
75      cout << "5. Kegunaan: Cocok untuk daftar yang kecil atau ketika data tidak perlu diurutkan." << endl;
76      cout << "\nBinary Searching (Pencarian Biner):" << endl;
77      cout << "1. Cara Kerja: Metode ini bekerja dengan membagi daftar menjadi dua bagian pada setiap langkah. Ini membandingkan elemen target dengan elemen tengah, dan kemudian memutuskan apakah akan mencari di paruh kiri atau paruh kanan." << endl;
78      cout << "2. Persyaratan Data: Wajib memerlukan data yang sudah diurutkan. Jika data tidak diurutkan, Binary Search tidak akan memberikan hasil yang benar." << endl;
79      cout << "3. Kecepatan: Jauh lebih cepat daripada Sequential Search untuk daftar yang besar, karena setiap langkah mengurangi ruang pencarian menjadi setengahnya." << endl;
80      cout << "4. Kompleksitas Waktu: Memiliki kompleksitas waktu rata-rata O(log n), yang berarti waktu pencarian meningkat secara logaritmik dengan ukuran daftar." << endl;
81      cout << "5. Kegunaan: Sangat efisien untuk daftar yang besar dan sudah terurut, seperti dalam database atau kamus." << endl;
82      cout << "\n--- Kesimpulan ---" << endl;
83      cout << "Pilihan antara Sequential Search dan Binary Search tergantung pada apakah data Anda sudah diurutkan dan seberapa besar daftar yang akan dicari." << endl;
84  }
85
86
```

```
 87 int main() {
 88     int opt, target;
 89     do {
 90         cout << "Pilih menu" << endl;
 91         cout << "1. Sequential Searching" << endl;
 92         cout << "2. Binary Searching" << endl;
 93         cout << "3. Jelaskan Perbedaan Sequential
    Searching dan Binary Searching!" << endl;
 94         cout << "4. Exit" << endl;
 95         cout << "Pilih: ";
 96         cin >> opt;
 97 
 98         switch (opt) {
 99             case 1: {
100                 vector<int> nums (100);
101                 mt19937_64 rng(random_device{}());
102                 uniform_int_distribution<int>
    dist(1, 50);
103 
104                 for (auto &val: nums) {
105                     val = (dist(rng));
106                 }
107 
108                 // give message here
109                 // print the number list
110                 cout << "Daftar angka: " << endl;
111                 for (int i = 0; i < nums.size(); ++i)
    {
112                     cout << "[" << i << "] " <<
    nums[i] << " ";
113                     if ((i + 1) % 10 == 0) cout <<
    endl;
114                 }
115 
116                 cout << "\nMasukkan angka yang ingin
    dicari: "; cin >> target;
117                 sequentialSearch(nums, target);
118                 break;
119             }
120 
121             case 2: {
122                 int size;
123                 cout << "Masukkan ukuran vector: ";
124                 cin >> size;
125 
```

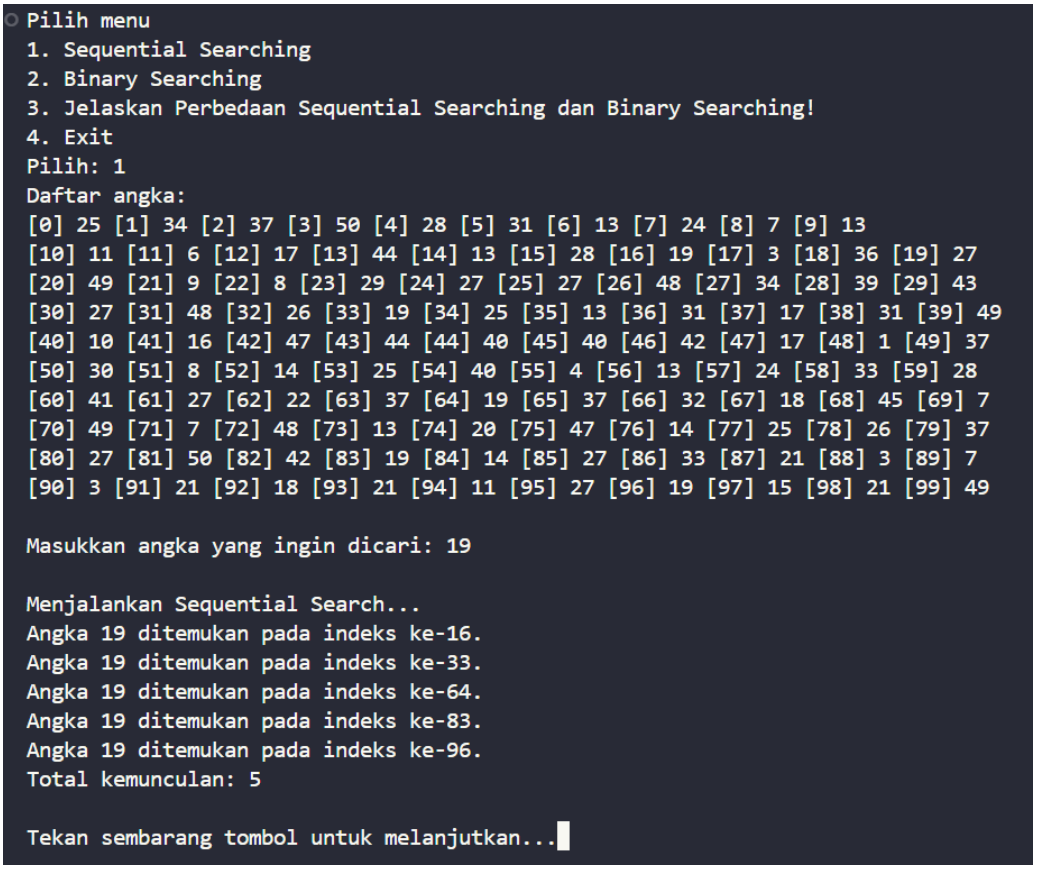
```
126                 vector<int> nums(size);
127                 mt19937_64 rng(random_device{}());
128                 uniform_int_distribution<int>
    dist(1, 100);
129
130                 for (auto &val: nums) {
131                     val = (dist(rng));
132                 }
133
134                 // give message here
135                 // print the number list
136                 cout << "Daftar angka:\n";
137                 for (int i = 0; i < nums.size(); ++i)
    {
138                     cout << "[" << i << "] " <<
    nums[i] << " ";
139                     if ((i + 1) % 10 == 0) cout <<
    endl;
140                 }
141
142                 cout << "\nMasukkan angka yang ingin
    dicari: "; cin >> target;
143                 binarySearch(nums, target);
144                 break;
145             }
146
147             case 3:
148                 explain();
149                 break;
150
151             case 4:
152                 cout << "\nTERIMA KASIH\n";
153                 cout << "Programme was made by Noor
    Khalisa (2410817220012)" << endl;
154                 break;
155
156             default:
157                 cout << "Opsi tidak terdefinisi,
    mohon masukkan ulang opsi" << endl;
158                 break;
159         }
160
161         if (opt != 4) {
162             cout << "\nTekan sembarang tombol untuk
    melanjutkan...";
```

```
163                 getch();
164                 clearScreen();
165             }
166
167         } while (opt != 4);
168
169         return 0;
170 }
```

B. Output



Gambar 1 Tampilan Menu



Gambar 2 Sequential Searching

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

 Masukkan angka yang ingin dicari: 19

 Menjalankan Sequential Search...
 Angka 19 tidak ditemukan dalam daftar.

 Tekan sembarang tombol untuk melanjutkan...
```

Gambar 3 Sequential Searching Ketika Angka Tidak Ditemukan Dalam Daftar

```
Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih: 2
Masukkan ukuran vector: 100
Daftar angka:
[0] 73 [1] 48 [2] 16 [3] 37 [4] 12 [5] 37 [6] 84 [7] 5 [8] 5 [9] 82
[10] 44 [11] 85 [12] 53 [13] 87 [14] 24 [15] 46 [16] 86 [17] 34 [18] 9 [19] 31
[20] 41 [21] 13 [22] 82 [23] 85 [24] 28 [25] 79 [26] 51 [27] 10 [28] 97 [29] 4
[30] 74 [31] 65 [32] 46 [33] 5 [34] 22 [35] 48 [36] 51 [37] 88 [38] 16 [39] 92
[40] 6 [41] 47 [42] 78 [43] 91 [44] 54 [45] 6 [46] 13 [47] 37 [48] 61 [49] 8
[50] 21 [51] 81 [52] 81 [53] 51 [54] 69 [55] 70 [56] 8 [57] 16 [58] 73 [59] 73
[60] 54 [61] 96 [62] 23 [63] 48 [64] 44 [65] 50 [66] 51 [67] 56 [68] 85 [69] 67
[70] 7 [71] 44 [72] 51 [73] 48 [74] 53 [75] 30 [76] 31 [77] 7 [78] 80 [79] 2
[80] 28 [81] 34 [82] 90 [83] 75 [84] 54 [85] 96 [86] 52 [87] 1 [88] 56 [89] 23
[90] 97 [91] 6 [92] 88 [93] 21 [94] 10 [95] 47 [96] 45 [97] 55 [98] 39 [99] 64

Masukkan angka yang ingin dicari: 25

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

Angka 25 tidak ditemukan dalam daftar.

Tekan sembarang tombol untuk melanjutkan...
```

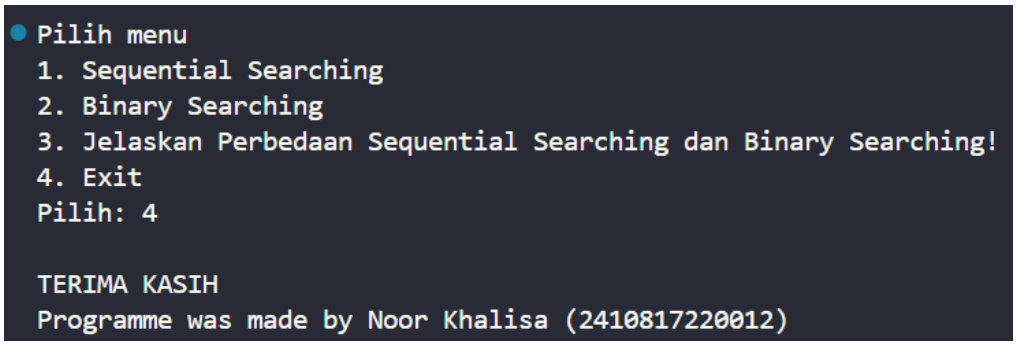
Gambar 4 Binary Searching

```
Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih: 2
Masukkan ukuran vector: 100
Daftar angka:
[0] 63 [1] 92 [2] 37 [3] 17 [4] 25 [5] 75 [6] 74 [7] 87 [8] 83 [9] 1
[10] 39 [11] 28 [12] 42 [13] 95 [14] 54 [15] 28 [16] 8 [17] 88 [18] 31 [19] 14
[20] 2 [21] 89 [22] 100 [23] 27 [24] 16 [25] 76 [26] 87 [27] 62 [28] 65 [29] 24
[30] 64 [31] 38 [32] 63 [33] 11 [34] 12 [35] 68 [36] 55 [37] 72 [38] 81 [39] 31
[40] 54 [41] 44 [42] 42 [43] 21 [44] 100 [45] 10 [46] 49 [47] 62 [48] 14 [49] 61
[50] 87 [51] 17 [52] 31 [53] 74 [54] 70 [55] 48 [56] 40 [57] 69 [58] 98 [59] 9
[60] 11 [61] 83 [62] 19 [63] 98 [64] 25 [65] 66 [66] 16 [67] 44 [68] 45 [69] 70
[70] 76 [71] 79 [72] 30 [73] 47 [74] 46 [75] 52 [76] 79 [77] 25 [78] 100 [79] 17
[80] 99 [81] 99 [82] 67 [83] 49 [84] 31 [85] 69 [86] 74 [87] 45 [88] 90 [89] 27
[90] 12 [91] 22 [92] 98 [93] 2 [94] 41 [95] 2 [96] 51 [97] 31 [98] 57 [99] 45

Masukkan angka yang ingin dicari: 19

Menjalankan Binary Search...
Daftar angka yang diurutkan:
[0] 1 [1] 2 [2] 2 [3] 2 [4] 8 [5] 9 [6] 10 [7] 11 [8] 11 [9] 12
[10] 12 [11] 14 [12] 14 [13] 16 [14] 16 [15] 17 [16] 17 [17] 17 [18] 19 [19] 21
[20] 22 [21] 24 [22] 25 [23] 25 [24] 25 [25] 27 [26] 27 [27] 28 [28] 28 [29] 30
[30] 31 [31] 31 [32] 31 [33] 31 [34] 31 [35] 37 [36] 38 [37] 39 [38] 40 [39] 41
[40] 42 [41] 42 [42] 44 [43] 44 [44] 45 [45] 45 [46] 45 [47] 46 [48] 47 [49] 48
[50] 49 [51] 49 [52] 51 [53] 52 [54] 54 [55] 54 [56] 55 [57] 57 [58] 61 [59] 62
[60] 62 [61] 63 [62] 63 [63] 64 [64] 65 [65] 66 [66] 67 [67] 68 [68] 69 [69] 69
[70] 70 [71] 70 [72] 72 [73] 74 [74] 74 [75] 74 [76] 75 [77] 76 [78] 76 [79] 79
[80] 79 [81] 81 [82] 83 [83] 83 [84] 87 [85] 87 [86] 87 [87] 88 [88] 89 [89] 90
[90] 92 [91] 95 [92] 98 [93] 98 [94] 98 [95] 99 [96] 99 [97] 100 [98] 100 [99] 100

Angka 19 ditemukan pada indeks ke-18 (dalam daftar yang diurutkan).

Tekan sembarang tombol untuk melanjutkan...
```

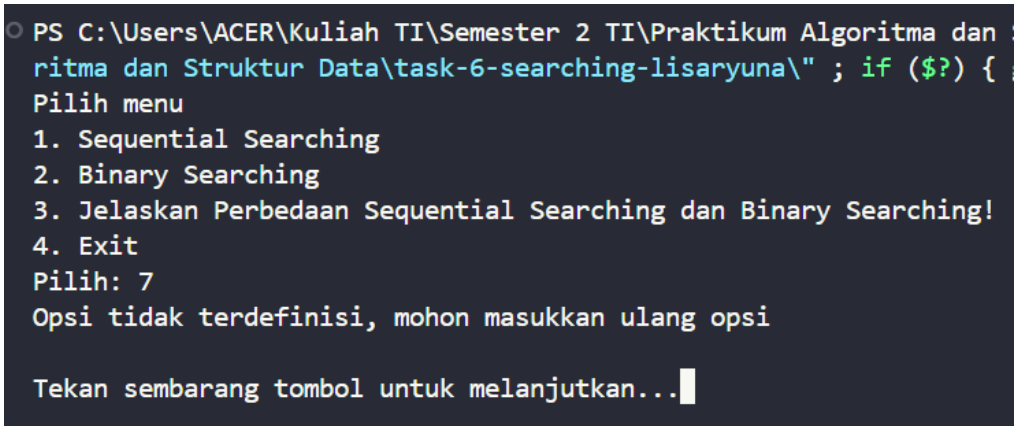
Gambar 5 Binary Searching Ketika Angka Tidak Ditemukan Dalam Daftar

```
PS C:\Users\ACER\Kuliah TI\Semester 2 TI\Praktikum Algoritma dan Struktur Data\task-6-searching-lisaryuna> cd "c:\Users\ACER\Kuliah TI\Semester 2 TI\Praktikum Algo
ritma dan Struktur Data\task-6-searching-lisaryuna\" ; if ($?) { g++ searching.cpp -o searching } ; if ($?) { .\searching }
Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih: 3

--- Perbedaan Sequential Searching dan Binary Searching ---
Sequential Searching (Pencarian Berurutan):
1. Cara Kerja: Metode ini mencari elemen secara berurutan, satu per satu, dari awal hingga akhir daftar.
2. Persyaratan Data: Tidak memerlukan data yang diurutkan. Dapat bekerja pada daftar angka yang tidak terurut.
3. Kecepatan: Relatif lebih lambat, terutama untuk daftar yang besar, karena dalam kasus terburuk harus memeriksa setiap elemen.
4. Kompleksitas Waktu: Memiliki kompleksitas waktu rata-rata O(n), di mana 'n' adalah jumlah elemen dalam daftar.
5. Kegunaan: Cocok untuk daftar yang kecil atau ketika data tidak perlu diurutkan.

Binary Searching (Pencarian Biner):
1. Cara Kerja: Metode ini bekerja dengan membagi daftar menjadi dua bagian pada setiap langkah. Ini membandingkan elemen target dengan elemen tengah, dan kemudian
memutuskan apakah akan mencari di paruh kiri atau paruh kanan.
2. Persyaratan Data: Wajib memerlukan data yang sudah diurutkan. Jika data tidak diurutkan, Binary Search tidak akan memberikan hasil yang benar.
3. Kecepatan: Jauh lebih cepat daripada Sequential Search untuk daftar yang besar, karena setiap langkah mengurangi ruang pencarian menjadi setengahnya.
4. Kompleksitas Waktu: Memiliki kompleksitas waktu rata-rata O(log n), yang berarti waktu pencarian meningkat secara logaritmik dengan ukuran daftar.
5. Kegunaan: Sangat efisien untuk daftar yang besar dan sudah terurut, seperti dalam database atau kamus.

--- Kesimpulan ---
Pilihan antara Sequential Search dan Binary Search tergantung pada apakah data Anda sudah diurutkan dan seberapa besar daftar yang akan dicari.

Tekan sembarang tombol untuk melanjutkan...
```

Gambar 6 Perbedaan Sequential Searching dan Binary Searching



Gambar 7 Exit



Gambar 8 Opsi Tidak Terdefinisi

C. Pembahasan

Struktur Kode Umum

Header Files (#include) adalah pustaka yang berisi kode-kode siap pakai.

- <iostream> digunakan untuk input dan output standar (seperti cout untuk mencetak ke konsol dan cin untuk membaca input dari konsol).
- <conio.h> adalah library yang menyediakan fungsi-fungsi konsol input/output, seperti getch(), yang digunakan untuk mendapatkan karakter dari keyboard tanpa perlu menekan enter.
- <random> menyediakan alat untuk menghasilkan angka acak. Library ini menyediakan engine (seperti mt19937_64) dan distribusi (seperti uniform_int_distribution) untuk mengontrol karakteristik angka acak yang dihasilkan.

- <vector> menyediakan kelas std::vector, yang merupakan array dinamis. Ini memungkinkan untuk membuat daftar angka yang ukurannya bisa berubah saat runtime.
- <algorithm> berisi berbagai algoritma standar, termasuk std::sort yang digunakan untuk mengurutkan koleksi elemen.

using namespace std; adalah pernyataan yang memungkinkan agar bisa menggunakan elemen-elemen dari namespace std (standard) secara langsung, seperti cout, cin, vector, dan sort, tanpa perlu menambahkan std:: di depannya.

Fungsi sequentialSearch

Fungsi sequentialSearch berfungsi untuk mencari satu atau lebih kemunculan sebuah angka target dalam sebuah daftar (vector) secara berurutan, dari awal hingga akhir.

Parameter:

- const vector<int> &nums menerima referensi konstan ke vector yang berisi bilangan bulat. const berarti fungsi tidak akan mengubah vector asli, dan & berarti fungsi bekerja langsung pada vector asli untuk efisiensi.
- int target menerima angka bilangan bulat yang akan dicari.

Cara Kerja:

1. Inisialisasi Penghitung: sebuah variabel int count diinisialisasi ke 0. Variabel ini akan melacak berapa kali target ditemukan di dalam vector.
2. Jelajah Daftar: fungsi ini menggunakan sebuah loop for yang akan berulang dari indeks 0 hingga nums.size() – 1. Artinya, setiap elemen dalam vector akan diperiksa secara individual. nums.size() adalah metode dari objek vector yang mengembalikan jumlah elemen saat ini dalam vector.
3. Perbandingan Elemen: di setiap iterasi loop, nums[i] (elemen pada indeks i) dibandingkan dengan target menggunakan operator ==.
4. Jika Ditemukan:

- Jika nums[i] == target, program akan mencetak pesan yang yang menunjukkan bahwa target ditemukan pada indeks i.
- Variabel count akan dinaikkan sebanyak satu (count++).

5. Hasil Akhir:
    - Setelah loop selesai, fungsi memeriksa nilai count.
    - Jika count masih 0, ini berarti target tidak ditemukan sama sekali dalam vector, dan pesan yang sesuai akan dicetak.
    - Jika count lebih besar dari 0, ini berarti target ditemukan setidaknya sekali, dan program akan mencetak total kemunculan target yang ditemukan.

Fungsi binarySearch

Fungsi binarySearch berfungsi untuk mencari sebuah angka target dalam daftar yang sudah terurut dengan cara membagi rentang pencarian menjadi dua pada setiap langkah. Ini jauh lebih efisien daripada Sequential Search untuk daftar yang besar.

Parameter:

- const vector<int> &nums menerima referensi konstan ke vector yang berisi bilangan bulat. const berarti fungsi tidak akan mengubah vector asli, dan & berarti fungsi bekerja langsung pada vector asli untuk efisiensi.
- int target menerima angka bilangan bulat yang akan dicari.

Cara Kerja:

1. Pengurutan Data:
    - vector<int> sortedNums = nums adalah sebuah salinan dari vector nums dibuat dan disimpan ke dalam sortedNums. Ini dilakukan karena binarySearch memerlukan data yang terurut, dan kita tidak ingin mengubah vector asli nums (yang mungkin belum terurut).
    - sort(sortedNums.begin(), sortedNums.end()) adalah fungsi std::sort dari <algorithm> dipanggil untuk mengurutkan sortedNums dari yang

terkecil hingga terbesar. sortedNums.begin() dan sortedNums.end() adalah iterator yang menandai awal dan akhir vector yang akan diurutkan.

- sortedNums kemudian ditampilkan, sehingga user dapat melihat daftar yang sudah terurut.

2. Inisialisasi Batas:
    - int left = 0; left diatur ke indeks pertama (0) dari sortedNums, menandai batas bawah (paling kiri) dari rentang pencarian.
    - int right = sortedNums.size() - 1; right diatur ke indeks terakhir dari sortedNums, menandai batas atas (paling kanan) dari rentang pencarian.
    - bool found = false, variabel boolean ini diinisialisasi ke false dan akan diatur menjadi true jika target berhasil ditemukan.
3. Loop Pencarian (while): while (left <= right), loop ini akan terus berjalan selama left kurang dari atau sama dengan right. Ini menunjukkan bahwa masih ada rentang (segmen) vector yang valid untuk diperiksa. Jika left melebihi right, berarti target tidak ditemukan dalam rentang saat ini.
4. Tentukan Titik Tengah: int mid = left + (right - left) / 2 menghitung indeks tengah dari rentang left dan right saat ini. Metode ini (left + (right - left) / 2) lebih aman daripada (left + right) / 2 untuk menghindari potensi integer overflow jika left dan right keduanya adalah bilangan positif yang sangat besar.
5. Perbandingan dengan Target:
    - Kasus 1: Ditemukan (sortedNums[mid] == target)
        - Jika elemen di mid sama dengan target, berarti angka yang dicari telah ditemukan.
        - Program mencetak pesan yang menunjukkan lokasi target (indeks mid) dalam daftar yang sudah terurut.
        - found diatur menjadi true.
        - break, loop dihentikan karena target sudah berhasil ditemukan.
    - Kasus 2: Target Lebih Besar (sortedNums[mid] < target)

    - Jika elemen di mid lebih kecil dari target, ini berarti target (jika ada) harus berada di paruh kanan dari mid.
    - Batas bawah pencarian (left) diperbarui menjadi mid + 1 untuk melanjutkan pencarian di paruh kanan.
- Kasus 3: Target Lebih Kecil (sortedNums[mid] > target)
    - Jika elemen di mid lebih besar dari target, ini berarti target (jika ada) harus berada di paruh kiri dari mid.
    - Batas atas pencarian (right) diperbarui menjadi mid - 1 untuk melanjutkan pencarian di paruh kiri.

6. Penyelesaian Loop: loop berakhir ketika target ditemukan atau ketika left melebihi right (menandakan bahwa rentang pencarian telah habis dan target tidak ditemukan).
7. Hasil Akhir: setelah loop selesai, jika variabel found masih false, ini berarti target tidak ada dalam daftar yang diurutkan, dan pesan yang sesuai akan dicetak.

Fungsi clearScreen

Fungsi clearScreen berfungsi untuk membersihkan tampilan layar terminal.

Cara Kerja: system("cls") memangggil perintah sistem operasi cls. Ini adalah perintah clear screen untuk sistem operasi Windows.

Fungsi explain

Fungsi explain berfungsi untuk menjelaskan perbedaan antara algoritma Sequential Searching dan Binary Searching.

Cara Kerja:

1. Pencetakan Informasi: fungsi ini hanya berisi serangkaian pernyataan cout yang mencetak teks penjelasan informatif.
2. Detail Penjelasan: teks tersebut merinci perbedaan utama antara Sequential Search dan Binary Search, mencakup aspek-aspek berikut:

- Cara Kerja: menjelaskan proses langkah demi langkah dari setiap algoritma.
- Persyaratan Data: menekankan kondisi data yang harus dipenuhi (misalnya, kebutuhan data terurut untuk Binary Search).
- Kecepatan (Efisiensi): membandingkan seberapa cepat (atau lambat) kedua algoritma beroperasi, terutama pada kumpulan data yang besar.
- Kompleksitas Waktu: menyebutkan notasi O besar (O(n) untuk Sequential Search dan O(logn) untuk Binary Search), yang merupakan cara standar untuk mengukur efisiensi algoritma dalam kaitannya dengan ukuran input.
- Kegunaan: memberikan konteks kapan masing-masing algoritma lebih cocok digunakan dalam skenario praktis.

Fungsi main

Fungsi main adalah titik awal eksekusi program. Fungsi ini mengelola alur program secara keseluruhan, berinteraksi dengan user, dan memanggil fungsi-fungsi lain yang diperlukan.

Alur kerja dalalm main():

1. Inisialisasi Variabel:
    - int opt mendeklarasikan variabel integer opt yang akan menyimpan pilihan menu yang dimasukkan oleh user.
    - int targer mendeklarasikan variabel integer target yang akan menyimpan angka yang ingin dicari oleh user.
2. Loop Utama Program (do-while):

    do {...} while (opt != 4), program memulai dengan blok do {…}, yang berarti semua kode di dalamnya akan dieksekusi setidaknya satu kali. Setelah eksekusi pertama, loop akan terus berulang selama kondisi opt != 4 (pilihan user bukan 4, yaitu "Exit") terpenuhi. Ini memastikan program terus berjalan dan menampilkan menu hingga user secara eksplisit memilih untuk keluar.

3. Tampilan Menu: di setiap awal perulangan do-while, beberapa perintah cout akan menampilkan daftar opsi menu.
4. Menerima Pilihan User: cin >> opt akan menunggu user mengetikkan angka yang dipilih dan menekan enter. Angka yang dimasukkan akan disimpan ke dalam variabel opt.
5. Memilih dan Menjalankan Algoritma (Blok switch):
    - switch (opt) {...}, struktur switch-case ini digunakan untuk mengeksekusi blok kode yang berbeda secara selektif berdasarkan nilai variabel opt.
    - case 1 (Sequential Searching):
        - vector<int> nums (100) membuat sebuah vector baru bernama nums yang diinisialisasi dengan ukuran 100 elemen.
        - mt19937_64 rng(random_device{}()) menginisialisasi sebuah objek generator angka acak (rng) menggunakan engine Mersenne Twister 64-bit. random_device{}() digunakan untuk menghasilkan seed yang non-deterministik (bervariasi setiap kali program dijalankan), sehingga angka acak yang dihasilkan akan berbeda setiap saat.
        - uniform_int_distribution<int> dist(1, 50) mendefinisikan sebuah distribusi angka acak yang akan menghasilkan bilangan bulat (int) secara seragam (merata) dalam rentang inklusif dari 1 hingga 50.
        - for (auto &val: nums) adalah range-based for loop yang mengiterasi melalui setiap elemen dalam nums. &val berarti val adalah referensi ke elemen vector tersebut, memungkinkan modifikasi langsung pada elemen aslinya.
        - val = (dist(rng)), di setiap iterasi, sebuah angka acak baru dihasilkan oleh distribusi dist (menggunakan rng sebagai sumber keacakan) dan diberikan kepada val (elemen vector saat ini).
        - Daftar nums yang baru dibuat dan diisi angka acak ini kemudian ditampilkan.

    - Program meminta user untuk memasukkan target yang ingin dicari.
    - Akhirnya, fungsi sequentialSearch dipanggil dengan nums dan target sebagai argumen.
    - break mengakhiri eksekusi blok switch.
- case 2 (Binary Searching):
    - Program meminta user untuk memasukkan ukuran vector yang diinginkan.
    - Sebuah vector nums baru dibuat dengan ukuran yang ditentukan.
    - Mirip dengan case 1, nums diisi dengan angka acak, tetapi kali ini dalam rentang 1 hingga 100.
    - nums ditampilkan.
    - Program meminta target dari user.
    - Fungsi binarySearch dipanggil dengan nums dan target.
    - break mengakhiri eksekusi blok switch.
- case 3 (Jelaskan Perbedaan):
    - Memanggil fungsi explain() untuk menampilkan penjelasan.
    - break mengakhiri eksekusi blok switch.
- case 4 (Exit):
    - Program mencetak pesan "TERIMA KASIH" dan informasi pembuat program.
    - Loop do-while akan segera berakhir setelah ini.
    - break mengakhiri eksekusi blok switch.
- default:

  Jika user memasukkan nilai opt yang tidak cocok dengan case manapun (misalnya, angka di luar 1-4), blok default ini akan dieksekusi, menampilkan pesan kesalahan "Opsi tidak terdefinisi...".

6. Jeda dan Bersihkan Layar:
    - if (opt != 4) {...} adalah kondisi untuk memastikan bahwa blok kode di dalamnya hanya dieksekusi jika user belum memilih opsi "Exit" (4).

- cout << "\nTekan sembarang tombol untuk melanjutkan..." menampilkan pesan yang meminta user untuk menekan tombol.
- getch(), fungsi ini (dari <conio.h>) menunggu user menekan tombol apa pun. Input tidak akan ditampilkan.
- clearScreen() memanggil fungsi clearScreen() untuk membersihkan tampilan, mempersiapkan layar untuk menu atau output berikutnya agar lebih rapi.

7. Mengakhiri Program:

   return 0, ketika loop do-while berakhir (yaitu, user memilih opt = 4), fungsi main akan mengembalikan nilai 0. Dalam C++, mengembalikan 0 dari main secara konvensional menandakan bahwa program telah selesai dieksekusi dengan sukses tanpa kesalahan.