

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA**

**MODUL VIII
“ALGORITMA SEARCHING”**



Disusun Oleh :

Nama : Fitri Kusumaningtyas
NIM : 2311102068
Kelas : IF 11 B

DOSEN:

WAHYU ANDI SAPUTRA, S.Pd. , M.Eng

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

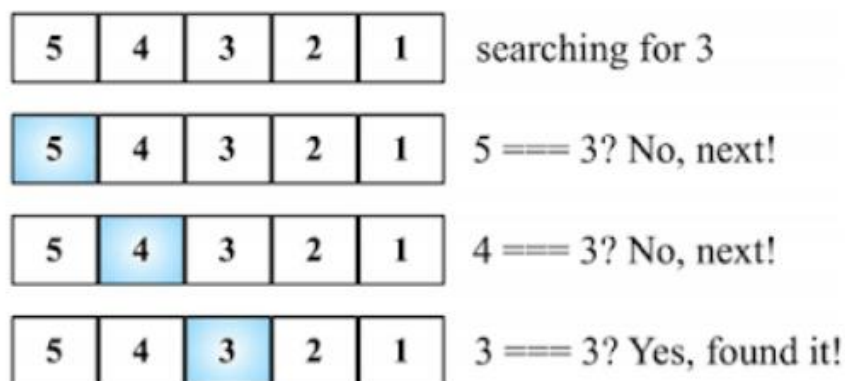
A. DASAR TEORI

Algoritma Searching adalah alat penting dalam ilmu komputer yang digunakan untuk menemukan elemen tertentu dalam kumpulan data. Algoritme ini dirancang untuk menavigasi struktur data secara efisien untuk menemukan informasi yang Anda perlukan, dan merupakan dasar dari berbagai aplikasi, seperti database dan mesin pencari web. Searching adalah proses dasar mencari item tertentu dalam kumpulan data. Kumpulan data ini dapat mengambil berbagai bentuk, seperti array, daftar, pohon, dan representasi terstruktur lainnya. Tujuan utama Searching adalah untuk menentukan apakah item yang menarik ada dalam data Anda, dan jika ada, temukan atau ambil item tersebut. Ini memainkan peran penting dalam berbagai tugas komputasi dan aplikasi dunia nyata, termasuk pengambilan informasi, analisis data, proses pengambilan keputusan, dan banyak lagi.

1. Sequential searching

Sequential Search Metode ini biasanya memiliki tujuan yang sama yaitu mencari data dalam sebuah array. Namun, Searching sekuensial jauh lebih mudah diimplementasikan secara terprogram dibandingkan Binary Search. Di bawah ini adalah beberapa proses cara kerja metode Searching sekuensial.

- Menentukan data yang akan dicari terlebih dahulu.
- Temukan data yang Anda cari dalam array satu per satu.
- Membandingkan data yang diambil dari data pertama dengan data terakhir dalam larik.
- Jika data ditemukan, pesan sukses ditampilkan.
- Jika tidak ada data yang ditemukan, pesan ``Data detail tidak ditemukan" juga ditampilkan.



2. Binary Search

Metode Searching biner ini berbeda dengan metode Searching sekuensial. Perbedaannya sangat jelas karena Searching sekuensial juga dapat mencari data secara acak tanpa proses yang rumit. Ini berbeda dengan metode Searching biner, yang mengharuskan data dalam array diurutkan dalam urutan menaik atau menurun. Oleh

karena itu, Searching biner memerlukan metode pengurutan. Di bawah ini adalah cara kerja algoritma atau Searching biner saat mencari data.

- Langkah pertama adalah mengulanginya untuk menentukan posisi terendah, yaitu posisi di mana indeks dapat ditampilkan sebagai yang terendah.
- Selanjutnya, tentukan posisi teratas dari array indeks.
- Dalam hal ini, gunakan algoritma seperti ini untuk mencari posisi tengah: $\text{Tengah} = (\text{posisi tertinggi} + \text{posisi terendah})/2$.
- Selanjutnya, bandingkan nilai yang Anda cari dengan nilai di tengah.
- Jika nilai atau data yang dicari cocok dengan nilai pusat atau larik data, operasi selesai.
- Jika nilai atau data yang dicari lebih kecil dari data di tengah, Searching dilanjutkan dengan elemen kiri array tengah.
- Syaratnya posisi tertinggi diubah ke posisi tengah - 1, dan posisi terendah tidak berubah.
- Jika nilai atau data yang diambil lebih besar dari data di tengah array.
- Oleh karena itu, Searching dilanjutkan ke sebelah kanan elemen array tengah.
- Syaratnya rank tertinggi tetap sama dan rank terendah menjadi rank tengah + 1.

Binary Search



	0	1	2	3	4	5	6	7	8	9
Search 23	2	5	8	12	16	23	38	56	72	91
	L=0	1	2	3	M=4	5	6	7	8	9
23 > 16 take 2 nd half	2	5	8	12	16	23	38	56	72	91
	0	1	2	3	4	L=5	6	M=7	8	H=9
23 < 56 take 1 st half	2	5	8	12	16	23	38	56	72	91
	0	1	2	3	4	L=5, M=5	H=6	7	8	9
Found 23, Return 5	2	5	8	12	16	23	38	56	72	91

B. GUIDED

1. Guided I

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;

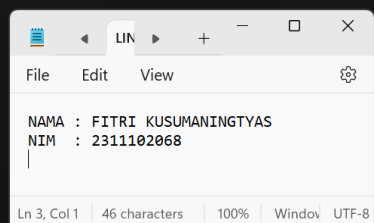
    if (ketemu){
        cout << "\nangka " << cari << " ditemukan pada
indeks ke-" << i << endl;
    } else {
        cout << cari << " tidak dapat ditemukan pada
data."<< endl;
    }
    return 0;
}
```

Ouput Program :

```
PS D:\strukdat> cd "d:\strukdat\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRu
nnerFile }
Program Sequential Search Sederhana

data : {9,4,1,7,5,12,4,13,4,10}

angka 10 ditemukan pada indeks ke-9
PS D:\strukdat> |
```



Deskripsi Program :

Kode di atas adalah program C++ yang melakukan Sequential Search pada array data.

- Deklarasi variabel n dengan nilai 10 yang menunjukkan jumlah elemen dalam array data.
- Deklarasi array data yang berisi 10 elemen yaitu {9, 4, 1, 7, 5, 12, 4, 13, 4, 10}.
- Deklarasi variabel Searching yang mengandung nilai 10.
- Menunjukkan nilai yang akan dicari dalam larik data.
- Deklarasi variabel dengan nilai 'false' ditemukan.
- Digunakan untuk menunjukkan apakah nilai Searching ditemukan.
- Deklarasi variabel i digunakan sebagai indeks untuk mengakses elemen dalam array data.

Algoritme Searching sequential dilakukan menggunakan loop for yang mengakses setiap elemen dalam array data. Jika nilai elemen array data cocok dengan nilai Searching, variabel yang ditemukan diubah menjadi true dan perulangan dihentikan dengan perintah break. Program menampilkan judul "Program Sequential Search Sederhana" dan isi larik data. Jika nilai Searching ditemukan dalam larik data, program akan menampilkan pesan berikut: "Nomor [nilai Searching] ditemukan pada indeks [indeks]". Jika nilai Searching tidak ditemukan dalam array data, program akan menampilkan pesan "[Nilai Searching] tidak ditemukan dalam data." Dalam contoh ini, program mencari nilai 10 dalam array data dan menampilkan hasilnya.

2. Guided II

Source Code

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

int dataa[7] = {1,8,2,5,4,9,7} ;
int cari ;

void selection_sort () {
    int temp, min , i , j;
    for ( i=0; i<7; i++){
        min =i;
        for (j = i+1 ; j<7; j++){
            if (dataa[j] < dataa[min] ) {
                min = j;
            }
        }
        temp = dataa[i];
        dataa[i] = dataa[min];
        dataa[min] = temp;
    }
}
```

```

        }
    }
    temp = dataa[i];
    dataa[i] = dataa[min] ;
    dataa[min] = temp;
}
}

void binarysearch (){
    //seacrhing
    int awal, akhir , tengah , b_flag = 0;
    awal = 0;
    akhir = 7;
    while (b_flag == 0 && awal <= akhir ){
        tengah = (awal + akhir)/2;
        if (dataa[tengah] == cari ){
            b_flag = 1;
            break;
        } else if (dataa[tengah] < cari )
            awal = tengah + 1;
        else
            akhir = tengah-1;
    }
    if (b_flag == 1)
        cout << "\n dataa ditemukan pada index ke-"<< tengah <<
endl;
    else
        cout << "\n dataa tidak ditemukan \n" ;
}

int main (){
    cout << "\t BINARY SEARCH" << endl;
    cout << "\n dataa : ";
    for (int x= 0; x<7; x++)
        cout << setw (3) << dataa[x];
    cout << endl;

    cout<< "\n Masukkan dataa yang ingin Anda cari :";
    cin>>cari;
    cout<< "\n dataa diurutkan : ";
    //urutkan dataa dengan selection sort
    selection_sort();
    //tampilkan dataa setelah diurutkan
    for(int x = 0; x<7;x++)
        cout<<setw(3)<<dataa[x];

    cout<<endl;
}

```

```

        binarysearch();
        _getche();
        return EXIT_SUCCESS;
    }
}

```

Output Program :

```

PS D:\strukdat> cd "d:\strukdat\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
BINARY SEARCH

dataa : 1 8 2 5 4 9 7

Masukkan dataa yang ingin Anda cari :7

dataa diurutkan : 1 2 4 5 7 8 9

dataa ditemukan pada index ke-4

```

Deskripsi Program :

Kode di atas merupakan program C++ yang menggunakan metode Binary Search untuk mencari data dalam array data yang diurutkan menggunakan metode seleksi.

- Deklarasi array data yang berisi tujuh elemen: {1, 8, 2, 5, 4, 9, 7}.
- Deklarasi variabel Searching untuk menyimpan data yang akan dicari.
- selection_sort() definisi fungsi.
- Urutkan array data menggunakan algoritma pengurutan selektif.
- Definisi fungsi binersearch() yang menggunakan penelusuran biner untuk mencari data dalam larik data.
- Di dalam fungsi main(), program menampilkan judul "BINARY SEARCH" dan isi array data.

Program ini memerlukan input pengguna untuk mengambil data. Data kemudian diurutkan menggunakan fungsi 'selection_sort()'. Program mengurutkan dan menampilkan data. Pengambilan data dilakukan menggunakan fungsi 'binarysearch()'. Setelah data ditemukan, program menampilkan indeks data. Jika tidak ada data yang ditemukan, program akan menampilkan pesan "Data Tidak Ditemukan". Program menunggu masukan pengguna sebelum keluar. Dalam contoh ini, program mencari data yang dimasukkan pengguna dalam array data yang diurutkan menggunakan pengurutan pilihan.

C. UNGUIDED

1. Unguided 1 Source Code

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

void selectionSort(string &huruf, int n)
{
    int a, b, min;
    for (a = 0; a < n - 1; a++)
    {
        min = a;
        for (b = a + 1; b < n; b++)
            if (huruf[b] < huruf[min])
                min = b;
        char temp = huruf[a];
        huruf[a] = huruf[min];
        huruf[min] = temp;
    }
}

int binarySearch(string huruf, int kiri, int kanan, char target)
{
    while (kiri <= kanan)
    {
        int mid = kiri + (kanan - kiri) / 2;
        if (huruf[mid] == target)
            return mid;
        if (huruf[mid] < target)
            kiri = mid + 1;
        else
            kanan = mid - 1;
    }
    return -1;
}

int main()
{
    string kalimat;
    char input;
    cout << "\n===== " << endl;
    cout << "          Mencari Huruf pada Kalimat          " << endl;
    cout << "===== " << endl;
    cout << "Masukkan kalimat: ";
    getline(cin, kalimat);
    cout << "Masukkan huruf yang ingin dicari: ";
    cin >> input;
```

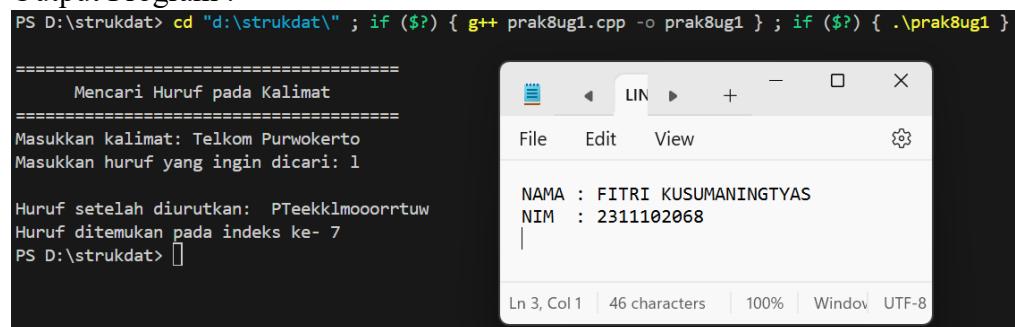


```

        cout << endl;
        selectionSort(kalimat, kalimat.size());
        int result = binarySearch(kalimat, 0, kalimat.size() - 1,
input);
        if (result == -1)
        {
            cout << "Huruf yang Anda cari tidak ditemukan!" <<
endl;
        }
        else
        {
            cout << "Huruf setelah diurutkan: " << kalimat << endl;
            cout << "Huruf ditemukan pada indeks ke- " << result
<< endl;
        }
        return 0;
    }
}

```

Output Program :



The screenshot shows a Windows Command Prompt window with the following text:

```

PS D:\strukdat> cd "d:\strukdat\" ; if ($?) { g++ prak8ug1.cpp -o prak8ug1 } ; if ($?) { .\prak8ug1 }

=====
Mencari Huruf pada Kalimat
=====
Masukkan kalimat: Telkom Purwokerto
Masukkan huruf yang ingin dicari: l

Huruf setelah diurutkan: PTeeklmooortuw
Huruf ditemukan pada indeks ke- 7
PS D:\strukdat>

```

Overlaid on the terminal is a small text editor window titled "LIN" with the following content:

```

NAMA : FITRI KUSUMANINGTYAS
NIM  : 2311102068

```

The status bar at the bottom of the editor shows "Ln 3, Col 1", "46 characters", "100%", "Window", and "UTF-8".

Deskripsi Program :

Kode di atas merupakan program C++ yang menggunakan algoritma selection sort untuk mengurutkan kalimat kemudian menggunakan Searching biner untuk mencari karakter dalam kalimat. fungsi selectionSort yang mengurutkan urutan karakter menggunakan algoritma pengurutan pilihan. Deklarasi fungsi "binarySearch" yang mencari karakter dalam string menggunakan metode Searching biner.

Pada fungsi utama,

- Program menampilkan judul "Mencari Huruf pada Kalimat" dan meminta pengguna memasukkan kalimat dan karakter yang akan dicari.
- Program ini menggunakan fungsi ``selectionSort" untuk mengurutkan kalimat.
- Program ini menggunakan fungsi 'binarySearch' untuk melakukan Searching karakter.

- Ketika karakter ditemukan, program akan mengurutkannya dan menampilkan indeks karakter dan kalimat.
- Jika surat tidak ditemukan, program akan menampilkan pesan "Huruf yang Anda cari tidak ditemukan! “

Dalam contoh ini, program menggunakan pengurutan pilihan untuk mencari karakter yang diketik oleh pengguna dalam teks yang diurutkan.

2. Unguided 2

Source Code

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

using namespace std;

int main() {
    string kalimat;
    int jumlah = 0;

    cout << "\n===== " << endl;
    cout << "    Perhitungan Huruf Vokal    " << endl;
    cout << "===== " << endl;

    cout << "Masukkan kalimat: ";
    getline(cin, kalimat);

    for (int i = 0; i < kalimat.length(); i++) {
        char a = kalimat[i];
        if (a == 'a' || a == 'i' || a == 'u' || a == 'e' || a
== 'o' ||
        a == 'A' || a == 'I' || a == 'U' || a == 'E' || a
== 'O') {
            jumlah++;
        }
    }

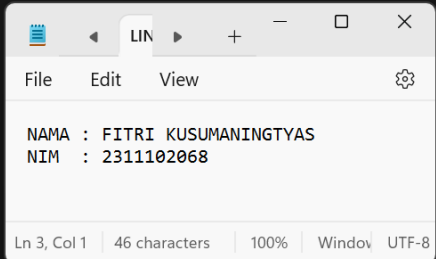
    cout << "Jumlah huruf vokal pada kalimat : " << jumlah << endl;

    return 0;
}
```

Output Program :

```
PS D:\strukdat> cd "d:\strukdat\" ; if ($?) { g++ prak8ug2.cpp -o prak8ug2 } ; if ($?) { .\prak8ug2 }

=====
Perhitungan Huruf Vokal
=====
Masukkan kalimat: Institut Teknologi Telkom Purwokerto
Jumlah huruf vokal pada kalimat : 13
PS D:\strukdat>
```



Deskripsi Program :

Kode di atas adalah program C++ yang digunakan untuk menghitung huruf vokal dalam sebuah kalimat.

- Deklarasi variabel kalimat untuk menyimpan pernyataan yang menghitung jumlah vokal.
- Deklarasi jumlah variabel untuk menyimpan jumlah vokal yang ditemukan.
- Program menampilkan judul "Perhitungan Huruf Vokal" dan meminta pengguna memasukkan kalimat.
- Program ini menggunakan perulangan for untuk mengulang setiap karakter dalam pernyataan.
- Di dalam perulangan, program menggunakan kondisi if untuk memeriksa apakah huruf saat ini adalah huruf vokal (a, i, u, e, o, atau huruf besar).
- Jika karakter saat ini adalah vokal, program menambahkan 1 ke variabel jumlah
- Setelah perulangan selesai, program akan menampilkan jumlah vokal yang ditemukan dalam kalimat.

Dalam contoh ini, program menghitung jumlah vokal dalam kalimat yang dimasukkan oleh pengguna.

3. Unguided 3

Source Code

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

int hitungAngka(const int array[], int size, int target)
{
    int jumlah = 0;
    for (int i = 0; i < size; i++)
    {
        if (array[i] == target)
        {
            jumlah++;
        }
    }
}
```

```

    }
}
return jumlah;
}

int main()
{
    const int size = 10;
    int array[size] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
    int target = 4;
    int jumlah = hitungAngka(array, size, target);
    cout << "\n===== Hitung Jumlah Angka =====" <<
endl;
    cout << "Data: ";
    for (int element : array)
    {
        cout << element << " ";
    }
    cout << "\nAngka yang dicari: " << target << endl;
    cout << "Ditemukan angka " << target << " dalam data
sebanyak: " << jumlah << endl;
    return 0;
}

```

Output Program :

The screenshot shows a terminal window with the following output:

```

PS D:\strukdat> cd "d:\strukdat\" ; if ($?) { g++ prak8ug3.cpp -o prak8ug3 } ; if ($?) { .\prak8ug3 }

===== Hitung Jumlah Angka =====
Data: 9 4 1 4 7 10 5 4 12 4
Angka yang dicari: 4
Ditemukan angka 4 dalam data sebanyak: 4
PS D:\strukdat>

```

Overlaid on the terminal is a Notepad++ window titled "LIN" containing the following text:

```

NAMA : FITRI KUSUMANINGTYAS
NIM  : 2311102068

```

The Notepad++ status bar at the bottom indicates "Ln 3, Col 1", "46 characters", "100%", "Window", and "UTF-8".

Deskripsi Program :

Program diatas merupakan program C++ yang digunakan untuk menghitung banyaknya kemunculan suatu bilangan tertentu (dalam hal ini bilangan 4) dalam suatu array dengan menggunakan algoritma Searching sekuensial.

- Deklarasi variabel ukuran untuk menyimpan jumlah elemen dalam array.
- Deklarasi Array Sebuah array berukuran, diinisialisasi dengan nilai yang ditentukan.
- Deklarasi variabel target untuk menyimpan nomor yang akan dicari.
- Deklarasi fungsi hitungAngka dengan parameter array, ukuran array, dan nomor target.

- Fungsi `hitungAngka` menggunakan variabel `count` untuk menyimpan jumlah kemunculan nomor target.
- Perulangan `for` digunakan untuk mengulang setiap elemen array.
- Di dalam loop, jika elemen array saat ini sama dengan nomor target, variabel numerik bertambah 1.
- Saat perulangan selesai, fungsi `'hitungAngka'` mengembalikan nilai variabel jumlah.
- Dalam fungsi utama, variabel `Jumlah` diinisialisasi dengan hasil pemanggilan fungsi `hitungAngka`.
- Selanjutnya, program menampilkan data dalam array, nomor target, dan jumlah kemunculan nomor target dalam array.

Dalam contoh ini, program menghitung jumlah kemunculan angka 4 dalam larik yang ditentukan. Hasilnya 4 kali.

D. KESIMPULAN

Mempelajari teori Searching di C++ sangat penting bagi programmer karena alasan berikut:

1. Efisiensi:
 - Mengetahui algoritma Searching yang tepat membantu programmer memecahkan masalah dengan cara yang paling efisien.
 - Searching sekuensial dan Searching biner adalah dua algoritma Searching yang umum digunakan dengan kompleksitas waktu yang berbeda.
 - Memahami kompleksitas waktu suatu algoritma Searching dapat membantu pemrogram memprediksi kinerja algoritma.
2. Ketepatan:
 - Algoritma Searching yang tepat membantu pemrogram menemukan apa yang mereka cari.
 - Searching sekuensial menemukan elemen Searching setiap kali elemen tersebut berada dalam array, sedangkan Searching biner menemukan elemen Searching hanya jika array diurutkan.
3. Pemahaman Algoritma:
 - Mempelajari teori Searching membantu pemrogram memahami cara kerja algoritma Searching dan bagaimana algoritma ini dapat diterapkan untuk pemecahan masalah.
 - Pemahaman yang baik tentang algoritma Searching dapat membantu pemrogram mengembangkan algoritma Searching mereka sendiri atau memodifikasi algoritma Searching yang ada.
4. Penerapan Praktis:
 - Algoritma Searching dapat digunakan dalam berbagai aplikasi, antara lain: Mencari data di database.
 - Mencari kata kunci dalam dokumen teks.
 - Menemukan nilai dalam array.
 - Temukan rute dalam diagram.

Mempelajari teori Searching di C++ adalah investasi berharga bagi programmer. Pemahaman teori Searching memungkinkan pemrogram memilih algoritma Searching yang tepat untuk menyelesaikan masalah secara efisien dan akurat.

E. REFERESNSI JURNAL

Paysan-Lafosse, T., Blum, M., Chuguransky, S., Grego, T., Pinto, B. L., Salazar, G. A., ... & Bateman, A. (2023). InterPro in 2022. *Nucleic acids research*, 51(D1), D418-D427.

Chen, Y., & Yao, S. (2017). Sequential search with refinement: Model and application with click-stream data. *Management Science*, 63(12), 4345-4365.

Robert, J., & Stahl, D. O. (1993). Informative price advertising in a sequential search model. *Econometrica: Journal of the Econometric Society*, 657-686.

Chandramohan, M., Xue, Y., Xu, Z., Liu, Y., Cho, C. Y., & Tan, H. B. K. (2016, November). Bingo: Cross-architecture cross-os binary search. In *Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering* (pp. 678-689).

Nowak, R. D. (2011). The geometry of generalized binary search. *IEEE Transactions on Information Theory*, 57(12), 7893-7906.