

## BAB 3

### Searching (Pencarian)

#### 1. Tujuan Instruksional Umum

- Mahasiswa mampu melakukan perancangan aplikasi menggunakan Struktur Searching (Pencarian).
- Mahasiswa mampu melakukan analisis pada algoritma Searching yang dibuat.
- Mahasiswa mampu mengimplementasikan algoritma Searching pada sebuah aplikasi secara tepat dan efisien

#### 2. Tujuan Instruksional Khusus

- Mahasiswa mampu menjelaskan mengenai algoritma Searching.
- Mahasiswa mampu membuat dan mendeklarasikan struktur algoritma Searching.
- Mahasiswa mampu menerapkan dan mengimplementasikan algoritma Searching.

### Sequential Search (Linear Search)

Teknik pencarian data dari array yang paling mudah adalah dengan cara *sequential search*, dimana data dalam array dibaca 1 demi satu, diurutkan dari index terkecil ke index terbesar, maupun sebaliknya.

Contoh :

Array :

`int a[5] = {0,3,6,10,1}` (index array pada bahasa C++ dimulai dari index ke 0 !!!) jika kita ingin mencari bilangan 6 dalam array tersebut, maka proses yang terjadi kita mencari

- dari array index ke-0, yaitu 0, dicocokkan dengan bilangan yang akan dicari, jika tidak sama, maka mencari ke index berikutnya
- pada array index ke-1, juga bukan bilangan yang dicari, maka kita mencari lagi pada index berikutnya
- pada array index ke-2, ternyata bilangan yang kita cari ada ditemukan, maka kita keluar dari looping pencarian.

Contoh source :

```
d:\Kuliah\ukdw\asiste-1\strukt-1\source\search-1\sequential.cpp
#include<stdio.h>
void main(){
int array_a[5]={0,3,6,10,1};
int i, cari, flag=0;
printf("Masukkan data yang ingin dicari : ");
scanf("%i",&cari);
for(i=0;i<5;i++){
    if(array_a[i]==cari){
        flag=1;
        break;
    }
}
if(flag==1) printf("data yang anda cari ditemukan pada index ke-%i",i);
else printf("data yang anda cari tidak ditemukan");
}
```

Output:

```
(Inactive D:\KULIAHWUKD\ASISTE-1\STRUKT-1\SOU...
Masukkan data yang ingin dicari : 6
data yang anda cari ditemukan pada index ke-2
```


## *Binary search*

Metode pencarian yang kedua adalah *binary search*, pada metode pencarian ini, **data harus diurutkan terlebih dahulu**. Pada metode pencarian ini, data dibagi menjadi dua bagian (secara logika), untuk setiap tahap pencarian.

Algoritma binary search :

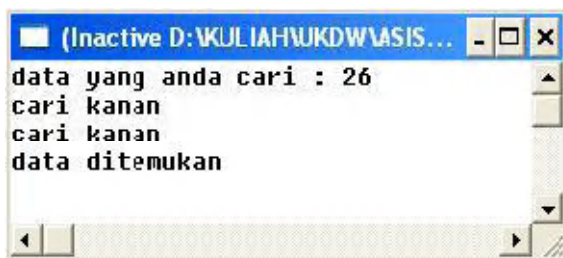
1. Data diambil dari posisi 1 sampai posisi akhir N
2. Kemudian cari posisi data tengah dengan rumus:  $(\text{posisi awal} + \text{posisi akhir}) / 2$
3. Kemudian data yang dicari dibandingkan dengan data yang di tengah, apakah sama atau lebih kecil, atau lebih besar?
4. Jika lebih besar, maka proses pencarian dicari dengan posisi awal adalah posisi tengah + 1
5. Jika lebih kecil, maka proses pencarian dicari dengan posisi akhir adalah posisi tengah - 1
6. Jika data sama, berarti ketemu.

Contoh source binary search :



```
d:\...\lasiste-1\strukt-1\source\search-1\binary.cpp
#include<stdio.h>
void main(){
int array_a[10]={0,2,5,7,11,12,14,22,26,31};
int awal, tengah, akhir, cari, flag=0;
awal=0;
akhir=9; //didapat dari banyak_data-1
printf("data yang anda cari : ");
scanf("%i",&cari);
while(awal<=akhir && flag==0){
    tengah=(awal+akhir)/2;
    if(array_a[tengah]==cari){
        flag=1;
        break;
    }
    else if(array_a[tengah]<cari){
        awal=tengah+1;
        printf("cari kanan\n");
    }
    else{
        akhir=tengah-1;
        printf("cari kiri\n");
    }
}
if(flag==1) printf("data ditemukan");
else printf("data tidak ditemukan");
}
```

Output:



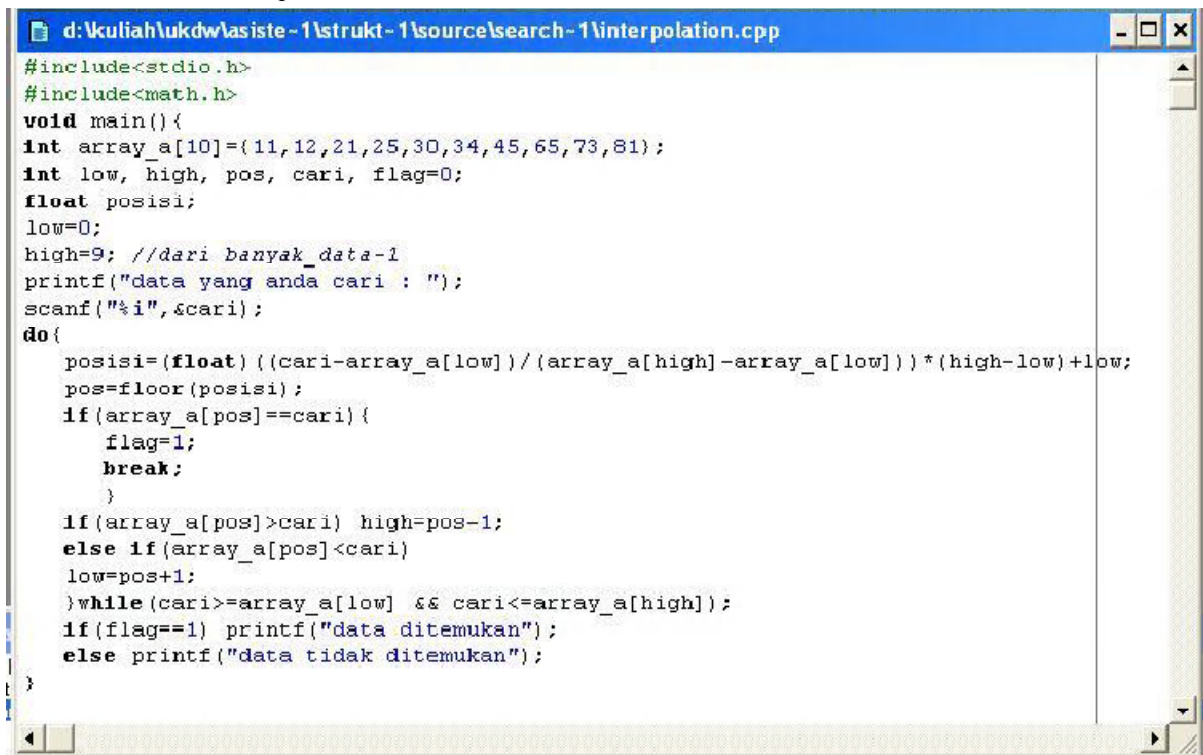
```
(Inactive D:\KULIAH\UKDW\ASIS... - [X]
data yang anda cari : 26
cari kanan
cari kanan
data ditemukan
```

## Interpolation Search

*Interpolation search* merupakan salah satu metode pencarian yang dapat digunakan. Seperti pada *binary search*, **data yang harus diurutkan terlebih dahulu**, sebelum dapat dilakukan pencarian dengan metode ini. Pada metode pencarian ini, kita mencoba menebak letak data yang kita cari, dengan perhitungan

- Jika  $\text{data}[\text{posisi}] > \text{data yg dicari}$ ,  $\text{high} = \text{pos} - 1$
- Jika  $\text{data}[\text{posisi}] < \text{data yg dicari}$ ,  $\text{low} = \text{pos} + 1$

Contoh source code interpolation search :



```
d:\Kuliah\ukdw\asiste-1\strukt-1\source\search-1\interpolation.cpp
#include<stdio.h>
#include<math.h>
void main(){
int array_a[10]={11,12,21,25,30,34,45,65,73,81};
int low, high, pos, cari, flag=0;
float posisi;
low=0;
high=9; //dari banyak_data-1
printf("data yang anda cari : ");
scanf("%i", &cari);
do{
    posisi=(float) ((cari-array_a[low]) / (array_a[high]-array_a[low])) * (high-low)+low;
    pos=floor(posisi);
    if(array_a[pos]==cari){
        flag=1;
        break;
    }
    if(array_a[pos]>cari) high=pos-1;
    else if(array_a[pos]<cari)
        low=pos+1;
}while(cari>=array_a[low] && cari<=array_a[high]);
if(flag==1) printf("data ditemukan");
else printf("data tidak ditemukan");
}
```

Output:



```
(Inactive D:\KULIAHWUKDWWASISTE...
data yang anda cari : 34
data ditemukan
```

Tambahan materi :

***break ;***

digunakan untuk keluar dari suatu blok perintah

***continue;***

digunakan untuk mem *by-pass* satu iterasi pada perulangan

contoh kode : (dijalankan dan dipelajari cara kerjanya)

```
#include<stdio.h>
void main(){
    for(int i=0;i<10;i++){
        for(int j=10;j>0;j--){
            if(i+j==10) break;
            //if(i+j==10) continue; //hilangkan tanda slash di depan if, dan beri
            slash 2 di depan if yang atas untuk mencoba kode continue
            printf("%i ",i+j);
            printf("pass");
        }
        printf("\n");
    }
}
```

### Latihan 1

1. Buatlah sebuah program yang dapat menerima inputan data kedalam sebuah array.
2. Lanjutan dari nomor 1, gunakan sequensial search untuk mencari sebuah nilai dari array tersebut, dan gantilah nilainya.
3. Coba gunakan metode pencarian lainnya
4. Buatlah menu untuk program tersebut (1. Sequential search, 2. Binary Search, 3. Interpolation Search)

### Latihan 2

1. Buatlah sebuah program yang dapat mencari dan menampilkan suatu bilangan yang dicari beserta indexnya.

Contoh :

isi array : 12, 14, 15, 12, 5

data yang dicari : 12

output: data 12 ditemukan pada index ke 0 dan 3

petunjuk :

- o coba tampilkan index array yang ditemukan pada sebuah array baru
- o cara 2, langsung tampilkan index array jika data ditemukan

2. Buatlah program untuk mencari data pada array 2 dimensi (bisa ditambahkan kode program untuk member inputan data dan ukuran array).

Contoh :

data array :

1	3	2
10	5	8
15	24	10

yang dicari : 24

output : data 24 berada pada posisi [2][1]

yang dicari : 2

output : data 2 berada pada posisi [0][2]

petunjuk : gunakan sequential search, karena data tidak diurutkan, terdapat 2 looping untuk proses pencarian.

### contoh

Menggunakan binary search dengan memasukkan jumlah data 10

```
# include <stdio.h>
void main()
{
    int A[10], index[10], i, j, k, tkr, top, bottom, middle, tm;
    for (i=0; i<10; i++)
    {
        printf("Data ke-%d=", i+1);
        scanf("%i", & A[i]);
    }
    printf("Masukkan data yang akan Anda cari:");
    scanf("%i", &k);

    for (i=0; i<10; i++)
    {
        for (j=i+1; j<10; j++)
        {
            if (A[i]>A[j])
            {
                tkr=A[i] ;
                A[i]=A[j];
                A[j]=tkr;
            }
        }
    }
    //proses pencarian data
    tm=0;
    top=9;
    bottom=0;
    while (top>=bottom)
    {
        middle=(top+bottom)/2;
        if (A[middle]==k)
        {
            tm++;
        }
        if (A[middle]<k)
        {
            bottom=middle+1;
        }
        else
        {
            top=middle-1;
        }
    }
    if (tm>0)
    printf("Data %d yang dicari ada dalam array \n",k);

    else
    printf("Data tidak ditemukan dalam array\n");
}
```

---



Untuk menghitung banyaknya huruf vocal dan konsonan.

```
// Menghitung jumlah vokal konsonan angka
# include <stdio.h>
# include <ctype.h>
# include <string.h>
void main()
{
    char kata[100],b=0,c=0,d=0;
    gets(kata);
    for (int a=0;kata[a];a++)
    {
        if (toupper(kata[a])== 'A' || toupper(kata[a])== 'I' | toupper(kata[a])== 'U'
        || toupper(kata[a])== 'E' || toupper(kata[a])== 'O' )
        {
            b++;
        }
        else if (isdigit (kata[a]))
        {
            c++;
        }
        else if (isspace (kata[a])){}
        else
            d++;
    }
    printf("Jumlah vokal=%i\t ",b);
    printf("Jumlah konsonan=%i\t",d);
    printf("Jumlah numerik=%i\t",c);
}
```

### LATIHAN 3

1. Buat program untuk mencari angka yang diinputkan, banyak data inputan user, lalu cari di indeks keberapa, dan berapa banyak data yang ditemukan. (Array 1 dimensi)
2. Buat program input number secara random, kemudian lakukan searching banyak bilangan yang genap dan yang ganjil.

### TUGAS

1. Buat program untuk mencari suatu data dan inputan berupa kalimat, hitung konsonan,vokal,numerik.

Seperti contoh:

```
Input : aku dan aka
Output :
Vokal = 5 = a u a a a
Konsonan = 4 = k d n k
Masukkan data yang akan Anda cari:s
```

Data tidak ada...

2. Buat searching number dg input random, jika input data tidak urut, maka program secara otomatis akan menggunakan metode Linear Search. Jika input data sudah terurut, akan muncul pilihan menggunakan metode binary atau interpolation search .
-

**Masukkan Jumlah Data yang digunakan: 100000**

**Tampilkan Data (Y/T): T**

**Urutkan Data (Y/T): Y**

**Masukkan data yang dicari: 9**

**Pilih Metode Pencarian:**

- 1. Binnary Search**
- 2. Interpolation Search**

**Pilih metode(1/2): 2**

**Data tidak ditemukan..Ulangi(Y/T): t**