

<p>IK131 Algoritma dan Pemrograman</p> <p>Searching (Pencarian)</p>



Disusun oleh:
Tim Pendamping Pak Jajang

2425 – Jajang Kusnendar, M.T.

PROGRAM STUDI PENDIDIKAN ILMU KOMPUTER

FAKULTAS PENDIDIKAN MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PENDIDIKAN INDONESIA

1. Identitas

Nama Mata Kuliah – Kode	:	IK131 – Algoritma dan Pemrograman
Materi	:	Searching (Pencarian)
Sub Materi	:	<ul style="list-style-type: none">- Sequential Search (Pencarian Berurutan)- Binary Search (Pencarian Biner)- Pencarian String
Beban (Waktu Pelaksanaan)	:	150 menit
Semester	:	1
Pengampu MK	:	Jajang Kusnendar, M.T

2. Capaian Praktikum

Capaian Praktikum
<ul style="list-style-type: none">- Memahami Konsep Pencarian- Mengimplementasi Sequential Search- Mengimplementasi Binary Search- Menganalisis perbandingan Keduanya- Menggunakan dalam Konteks Nyata

3. Perangkat yang Dibutuhkan

Perangkat lunak
<ul style="list-style-type: none">- Compiler bahasa C- Pada windows: DevC++, VisualStudio

Ringkasan Materi

Pencarian adalah teknik untuk menemukan elemen tertentu dalam sebuah koleksi data. Dalam pemrograman, ada dua metode umum untuk pencarian data:

A. Sequential Search (Pencarian Berurutan)

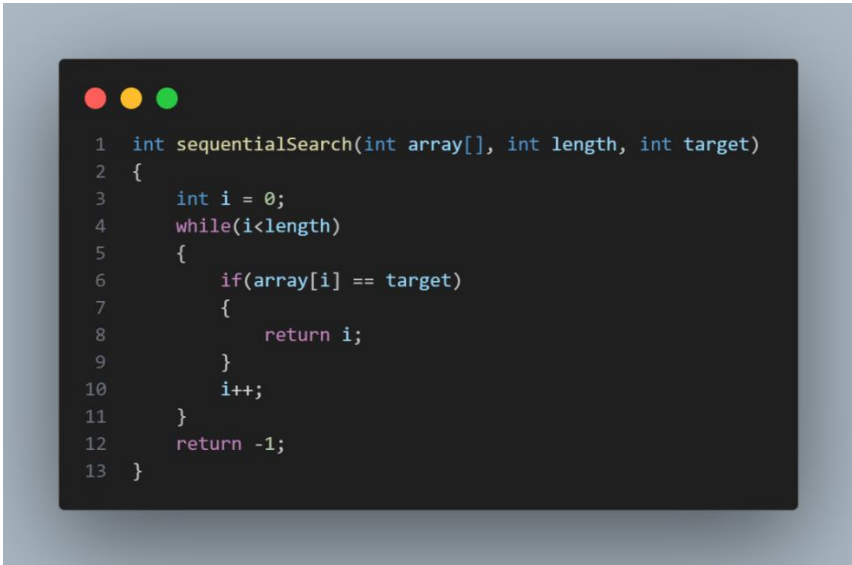
Sequential Search adalah algoritma pencarian yang memeriksa satu per satu elemen dari awal hingga akhir kumpulan data (array). Metode ini cocok digunakan untuk data yang tidak terurut.

Kelebihan:

- Sederhana dan mudah diimplementasikan.
- Dapat digunakan pada data yang tidak terurut.


Kekurangan:

- Kurang efisien untuk kumpulan data yang besar karena harus memeriksa setiap elemen satu per satu.

Penjelasan Kode Sequential Search

```
1  int sequentialSearch(int array[], int length, int target)
2  {
3      int i = 0;
4      while(i < length)
5      {
6          if(array[i] == target)
7          {
8              return i;
9          }
10         i++;
11     }
12     return -1;
13 }
```

- Fungsi sequentialSearch mengambil tiga parameter: array data, ukuran array (size), dan nilai yang ingin dicari (key).
- Fungsi ini menggunakan looping while, selama nilai i kurang dari panjang array (length).
- Jika elemen array pada indeks i sama dengan target, maka akan mengembalikan nilai i.
- Increment nilai i untuk memeriksa elemen berikutnya.
- Jika tidak ada elemen yang cocok hingga akhir, fungsi mengembalikan -1, yang berarti elemen tidak ditemukan.



```
1  int main()
2  {
3      int n, i, cari;
4      printf("Masukkan Jumlah Data: ");
5      scanf("%d", &n);
6
7      int arr[n];
8
9      for (i = 0; i < n; i++)
10     {
11         printf("Masukkan Data: ");
12         scanf("%d", &arr[i]);
13     }
14
15     printf("Masukkan Data yang Ingin Dicari: ");
16     scanf("%d", &cari);
17
```

- Di dalam fungsi main, meminta pengguna untuk memasukkan jumlah elemen (n)
- Loop untuk meminta pengguna mengisi elemen array arr. Setiap iterasi meminta input dan menyimpannya di arr[i]
- Meminta pengguna memasukkan nilai yang akan dicari dalam array.

```
1  int hasil = sequentialSearch(arr, n, cari);
2
3  if (hasil == -1)
4  {
5      printf("Data tidak ditemukan");
6  }
7  else
8  {
9      printf("Data terletak di indeks ke-%d", hasil);
10 }
11
12 return 0;
13 }
```

- Memanggil fungsi sequentialSearch dan menyimpan hasil pencarian dalam variabel hasil
- Jika hasil == -1, artinya pencarian tidak ditemukan dalam array, sehingga mencetak pesan "Data tidak ditemukan"
- Jika tidak (hasil == i), maka akan mencetak indeks tempat nilai cari ditemukan dalam array.

B. Binary Search (Pencarian Biner)

Binary Search adalah metode pencarian memerlukan data dalam keadaan terurut. Binary Search bekerja dengan membagi dua bagian data yang terurut secara berulang hingga elemen yang dicari ditemukan.

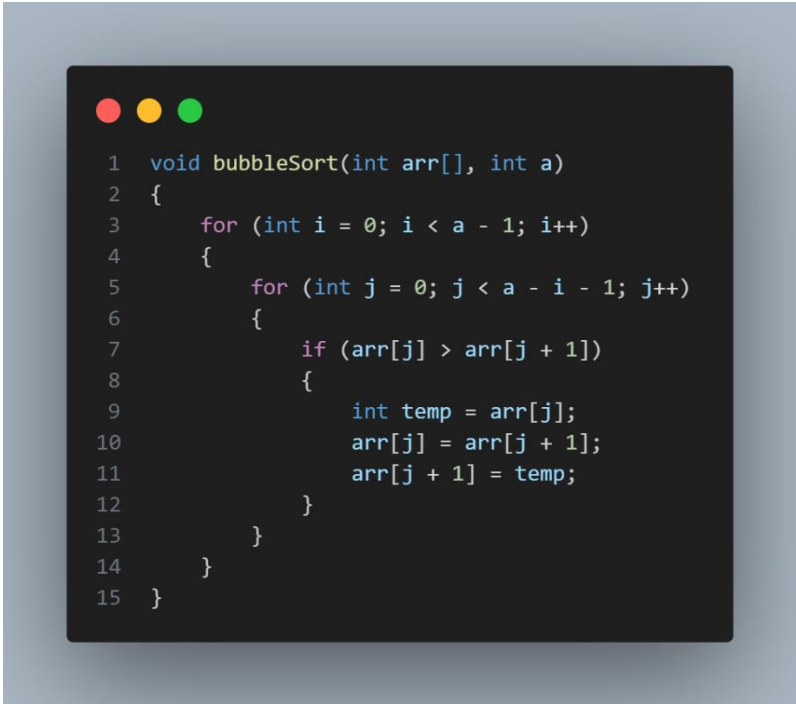
Kelebihan:

- Lebih cepat daripada Sequential Search, terutama untuk data yang besar.

Kekurangan:

- Hanya dapat digunakan pada data yang terurut.

Penjelasan kode binary search



```
1 void bubbleSort(int arr[], int a)
2 {
3     for (int i = 0; i < a - 1; i++)
4     {
5         for (int j = 0; j < a - i - 1; j++)
6         {
7             if (arr[j] > arr[j + 1])
8             {
9                 int temp = arr[j];
10                arr[j] = arr[j + 1];
11                arr[j + 1] = temp;
12            }
13        }
14    }
15 }
```

// Mengingatn lagi implementasi bubble sort

- Implementasi algoritma *Bubble Sort*. **Jika elemen pertama lebih besar dari elemen kedua, posisinya ditukar.**
- Menggunakan nested loop, loop yang pertama untuk mengatur iterasi yang sesuai dengan jumlah bilangan, loop yang kedua untuk membandingkan dan menukarkannya jika diperlukan.
- Menggunakan $a - 1$ pada loop i , karena pasti jika jumlah bilangan itu a , maka hanya membutuhkan $a - 1$ iterasi.
- Dengan menggunakan $a - i - 1$ pada loop j , kita mengurangi jumlah iterasi yang tidak perlu, sehingga mempercepat proses sorting.
- Fungsi code dalam if yaitu:
 1. Membandingkan dua elemen bersebelahan,
 2. Menukar posisi mereka jika tidak sesuai urutan.
 3. Menggerakkan elemen terbesar atau terkecil ke posisi yang benar pada setiap iterasi

```
1  int binarySearch(int arr[], int n, int target)
2  {
3      int low = 0;
4      int high = n - 1;
5
6      while (low <= high)
7      {
8          int mid = (high + low) / 2;
9
10         if (arr[mid] == target)
11         {
12             return mid;
13         }
14         else if (arr[mid] < target)
15         {
16             low = mid + 1;
17         }
18         else
19         {
20             high = mid - 1;
21         }
22     }
23
24     return -1;
25 }
```

- Selama nilai low lebih kecil atau sama dengan high, fungsi akan terus mencari elemen.
- Jika elemen tengah (arr[mid]) sama dengan target, fungsi mengembalikan indeks mid.
- Jika arr[mid] lebih kecil, cari di sisi kanan array (low = mid + 1).
- Jika lebih besar, cari di sisi kiri array (high = mid - 1).
- Jika elemen tidak ditemukan, fungsi mengembalikan nilai -1.

2425 – Jajang Kusnendar, M.T.

PROGRAM STUDI PENDIDIKAN ILMU KOMPUTER

FAKULTAS PENDIDIKAN MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PENDIDIKAN INDONESIA



```
1  int main()
2  {
3      int n, target;
4      printf("Masukkan Banyaknya data: ");
5      scanf("%d", &n);
6
7      int arr[n];
8
9      for (int i = 0; i < n; i++)
10     {
11         printf("Masukkan Data: ");
12         scanf("%d", &arr[i]);
13     }
14
15     bubbleSort(arr, n);
```

- Meminta input jumlah elemen array dari pengguna.
- Meminta pengguna memasukkan nilai-nilai elemen array satu per satu.
- Memanggil fungsi bubbleSort untuk mengurutkan array arr.


```
1 printf("Masukkan Angka yang ingin dicari: ");
2 scanf("%d", &target);
3
4 int hasil = binarySearch(arr, n, target);
5
6 if (hasil != -1)
7 {
8     printf("Nilai %d ditemukan di indeks ke-%d setelah diurutkan", target, hasil);
9 }
10 else
11 {
12     printf("Nilai %d tidak ditemukan", target);
13 }
14
15 return 0;
16 }
```

- Meminta pengguna memasukkan nilai target yang ingin dicari di dalam array.
- Memanggil fungsi `binarySearch` untuk mencari nilai target dalam array yang sudah terurut dan hasil pencarian disimpan dalam variabel `hasil`.
- Jika `binarySearch` menemukan target, tampilkan indeksnya dan jika tidak ditemukan (`hasil == -1`), tampilkan pesan bahwa target tidak ditemukan.

C. Pencarian String

- Pada dasarnya, pencarian string sama dengan pencarian integer, tapi dalam pencarian string kita menggunakan fungsi dari library `<string.h>` yaitu `strcmp()`;
- `strcmp()`; atau string compare ini berfungsi untuk membandingkan 2 string.
- Dengan `strcmp()`; kita membandingkan 2 string berdasarkan code ASCII.
- Jika string pertama lebih kecil maka akan mengembalikan nilai negative.
- Jika string pertama lebih besar maka akan mengembalikan nilai positif.
- Jika kedua string sama, maka akan mengembalikan nilai 0.

```
1 int sequentialSearch(char array[][50], int length, char target[]) {  
2     for (int i = 0; i < length; i++) {  
3         if (strcmp(array[i], target) == 0) { // Membandingkan string  
4             return i; // Mengembalikan indeks jika ditemukan  
5         }  
6     }  
7     return -1; // Mengembalikan -1 jika tidak ditemukan  
8 }
```

- Dalam string jangan lupa untuk memperhatikan inputnya. Karena karakter newline seringkali tersimpan di buffer. Jadi kita dapat menggunakan fungsi-fungsi di bawah ini:
 1. `getchar();` // Menghilangkan newline setelah input angka
 2. `(strcspn(str1, '\n') == '\0')` // Mengganti newline menjadi null.
 3. `fflush(stdin);` //Membersihkan input buffer (salah satunya newline);
- Kemudian jika ingin menginput string yang bisa menggunakan spasi dapat menggunakan:
 1. `fgets(buffer, sizeof(buffer), stdin)`
 2. `gets(buffer);`

Lembar Kerja Mahasiswa**1. Lengkapi kode program berikut**

```
#include <stdio.h>

int binarySearch(int arr[], int target, int n){

    while (low >= high){

        if (arr[mid] == target){
            //apa yang harus dikerjakan
        }
        else if (arr[mid] < target){
            //apa yang harus dikerjakan
        }
        else{
            //apa yang harus dikerjakan
        }
    }
    return -1;
}

int main() {

    int arr[] = {1, 3, 5, 7, 9, 11, 13};

    for(i = 0; i < 7; i++){
        printf("%d ", arr[i]);
    }

    printf("\nMasukkan angka yang ingin dicari: ");
    scanf("%d", &target);

    int result = binarySearch();

    if (result != -1) {
        printf("Elemen %d ditemukan pada indeks %d\n", target, result);
    } else {
        printf("Elemen %d tidak ditemukan\n", target);
    }
    return 0;
}
```

Input dan Output

```
1 3 5 7 9 11 13
Masukkan angka yang ingin dicari: 9
Elemen 9 ditemukan pada indeks 4
Press any key to continue . . .
```

2. Lengkapi kode program berikut

```
#include <stdio.h>

int sequentialSearch() {
    for (int i = 0; i < n; i++) {
        if () { //lengkapi kondisinya
            //apa yang harus dilakukan
        }
    }
    return -1;
}

int main() {

    printf("Masukkan Jumlah Data: ");
    scanf("%d", &n);

    int arr[n];

    for (i = 0; i < n; i++)
    {
        printf("Masukkan Data ke-%d: ", i+1);
        scanf("%d", &arr[i]);
    }

    printf("Masukkan Data yang Ingin Dicari: ");
    scanf("%d", &cari);

    int result = sequentialSearch();
    if () { //lengkapi kondisinya
        printf("Elemen %d ditemukan pada indeks %d\n", cari, result);
    } else {
        printf("Elemen %d tidak ditemukan\n", cari);
    }
    return 0;
}
```

Input dan Output

```
Masukkan Jumlah Data: 3
Masukkan Bilangan ke-1: 7
Masukkan Bilangan ke-2: 3
Masukkan Bilangan ke-3: 9
Masukkan Bilangan yang Ingin Dicari: 3
Bilangan 3 ditemukan pada indeks 1

Press any key to continue . . .
```

- Seorang guru ingin mengetahui siswa yang mendapatkan nilai tertentu dari daftar nilai yang ada di kelas. Untuk mempermudah pencarian, guru ingin menggunakan program untuk mencari siswa berdasarkan nilai yang dicari. Program ini akan menampilkan nama siswa dan indeksnya pada daftar jika ditemukan. Jika tidak, program akan memberikan pesan bahwa siswa dengan nilai tersebut tidak ditemukan.

Input dan Output

```
Masukkan jumlah siswa: 3
Siswa 1:
Nama: udin petot
Nilai: 85
Siswa 2:
Nama: maman racing
Nilai: 90
Siswa 3:
Nama: asep knalpot
Nilai: 70
Masukkan nilai yang ingin dicari: 90
Siswa dengan nilai 90 ditemukan di indeks 1, dengan Nama: maman racing

Press any key to continue . . .
```

- Petugas perpustakaan kesulitan mencari buku di rak karena banyaknya koleksi. Bnatulah petugas perpustakaan untuk mencari buku berdasarkan judul bukunya, dan program akan menampilkan judul buku jika ditemukan beserta kode bukunya, jika tidak program akan memberikan pesan bahwa Buku tidak ditemukan, sehingga dapat mempercepat proses pencarian dan menghemat waktu.

Input dan Output

```
Masukkan jumlah buku: 3
Buku 1:
Judul: algoritma pemrograman
Kode Buku: 112
Buku 2:
Judul: bahasa indonesia
Kode Buku: 556
Buku 3:
Judul: kalkulus
Kode Buku: 301
Masukkan nama buku yang ingin dicari: algoritma pemrograman
Buku ditemukan:
Judul: algoritma pemrograman
Kode Buku: 112
Press any key to continue . . .
```

5. Di sebuah toko bangunan, pemilik toko sering kesulitan mencari informasi tentang barang tertentu karena banyaknya barang yang tersedia. Setiap barang di toko memiliki **kode barang unik** dan **nama barang**. Pemilik ingin sebuah program yang dapat membantu mencari barang berdasarkan **kode barang** dan **nama lengkap barang**. Buatlah program untuk membantu pemilik toko mencari barang tersebut. Jika barang ditemukan, tampilkan **kode barang** dan **nama barang**. Jika tidak ditemukan, berikan pesan bahwa barang tidak ada.

Input dan Output

Cari berdasarkan kode

```
Masukkan jumlah barang: 3
Barang 1:
Kode Barang: 135
Nama Barang: semen
Barang 2:
Kode Barang: 225
Nama Barang: cangkul
Barang 3:
Kode Barang: 557
Nama Barang: paralon
Pilih metode pencarian:
1. Berdasarkan kode barang
2. Berdasarkan nama barang
Masukkan pilihan (1/2): 1
Masukkan kode barang: 225
Hasil pencarian:
Kode: 225, Nama: cangkul
Press any key to continue . . .
```

Cari berdasarkan nama

```
Masukkan jumlah barang: 3
Barang 1:
Kode Barang: 135
Nama Barang: semen
Barang 2:
Kode Barang: 225
Nama Barang: cangkul
Barang 3:
Kode Barang: 557
Nama Barang: paralon
Pilih metode pencarian:
1. Berdasarkan kode barang
2. Berdasarkan nama barang
Masukkan pilihan (1/2): 2
Masukkan nama barang: paralon
Hasil pencarian:
Kode: 557, Nama: paralon
Press any key to continue . . .
```