Bagian 4

4 Searching

“Something invisible doesn’t mean it doesn’t exist. You just need to searh”

- Me

Bab ini membahas tentang pencarian sekumpulan data menggunakan 2 metode yaitu *sequential* dan *binary*. Pembahasan tiap metode disertai contoh kasus dan ilustrasi secara visual untuk menggambarkan tiap prosesnya. Di akhir pembahasan juga diperkenalkan *standard library* C++ untuk proses pencarian.

4.1 Pengertian

Searching atau pencarian adalah salah satu aktivitas yang sering dilakukan dalam kehidupan sehari – hari. Contoh yang mudah adalah pencarian rumah yang berdasar pada alamat yang tercantum dalam kartu nama. Pada saat melakukan pencarian rumah maka akan di lihat satu persatu nomor tiap rumah apakah sesuai dengan yang tertulis pada kartu nama. Proses pencarian tersebut akan terus dilakukan sampai nomor rumah yang dicari sesuai atau tidak ditemukan sama sekali.

Contoh lain adalah mencari sebuah kartu dalam kumpulan kartu remi yang tersusun secara acak. Pencarian akan terus dilakukan satu demi satu dari tumpukan paling atas sampai kartu yang dimaksud ditemukan. Bila sampai pada kartu yang terakhir belum ada yang sesuai maka dapat dikatakan bahwa kartu yang dimaksud tidak ada.

Dalam pemrograman masalah pencarian adalah salah satu hal yang penting. Pencarian ini berkaitan dengan data yang biasanya tersimpan secara teruntun baik dalam bentuk array atau struktur yang lain.

4.2 Metode

Secara garis besar ada 2 metode dalam pencarian data yaitu pencarian beruntun (*sequential search*) dan pencarian biner (*binary search*). Kedua metode tersebut sebenarnya mengadopsi dari cara pencarian dalam kehidupan sehari – hari.

4.2.1 Sequential Search

Metode Sequential Search dapat digunakan untuk melakukan pencarian data baik pada array yang sudah terurut maupun yang belum terurut. Proses yang terjadi pada metode pencarian ini adalah sebagai berikut:

1. Menentukan data yang dicari.
2. Data yang dicari akan dibandingkan dengan data dalam array satu persatu mulai dari data ke-0 sampai index terakhir, tetapi
3. Jika perbandingan antara data yang dicari dengan data dalam *array* cocok maka proses dihentikan.

Sebagai ilustrasi ada sebuah array berukuran 7 bertipe **int** dengan nama **ar** berisi data {5, 9, 2, 7, 8, 1, 6} . Data yang akan dicari adalah 7 yang dinyatakan sebagai **x**. Berikut ini adalah contoh proses pencarian yang bisa digunakan.

```
bool found{false};
for(auto i = 0; i < 7; ++i)
{
    if(x == arr[i])
    {
        found = true;
        break;
    }
}
```

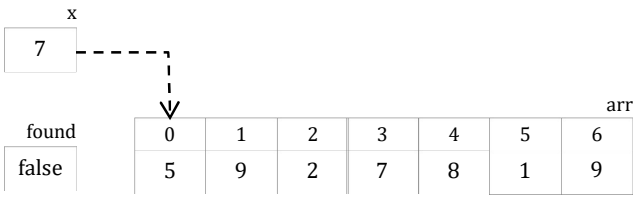
Pada potongan kode di atas terdapat variabel x sebagai penampung data yang di cari yaitu 7 dan variabel k sebagai penanda apakah data yang dicari ketemu. Sebagai nilai awal variabel `found` adalah *false* sebagai penanda bahwa data belum ditemukan.

Pada saat proses perulangan ke-i nilai `x` akan selalu dibandingkan dengan nilai `arr[i]`. Jika perbandingan sesuai maka nilai `found` akan diset menjadi *true* dan perulangan dihentikan.

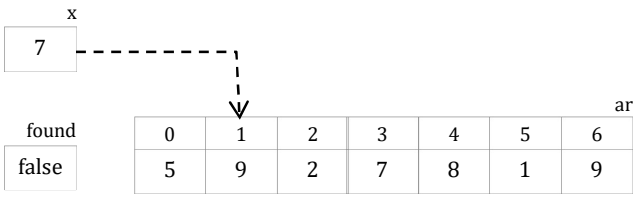
Akhir proses dari contoh di atas akan didapat nilai k adalah 3, sehingga pencarian dikatakan berhasil yaitu data ketemu pada index yang ke-3.

Ilustrasi secara visual contoh di atas adalah sebagai berikut.

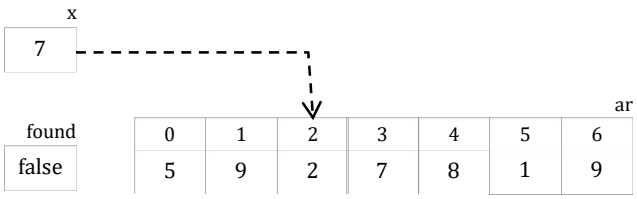
Perulangan i = 0



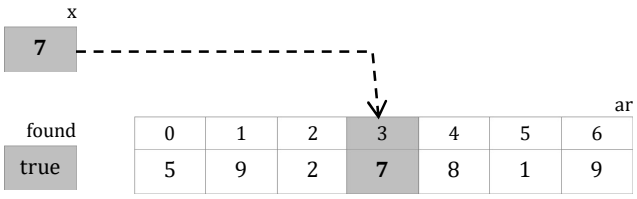
Perulangan i = 1



Perulangan i = 2



Perulangan i = 3



4.2.2 Binary Search

Metode Binary Search hanya digunakan untuk pencarian data yang sudah terurut. Proses yang terjadi pada pencarian dengan metode ini adalah sebagai berikut:

1. Menentukan data yang akan dicari.
2. Menentukan elemen tengah dari array.
3. Jika nilai elemen tengah sama dengan data yang dicari maka pencarian selesai, tetapi
4. Jika nilai elemen tengah tidak sama dengan data yang dicari maka:
    a. Jika nilai elemen tengah lebih besar daripada data yang dicari maka pencarian dilakukan pada setengah array pertama, tetapi
    b. Jika nilai elemen tengah lebih kecil daripada data yang dicari maka pencarian dilakukan pada setengah array berikutnya.

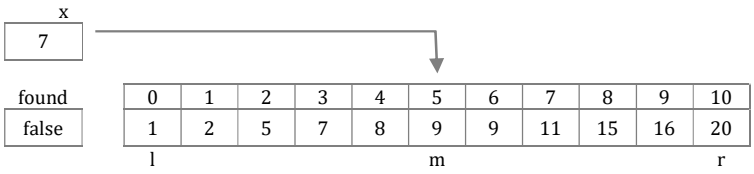
Sebagai ilustrasi terdapat *array* berukuran 11 bertipe **`int`** dengan nama **`ar`** berisi data terurut naik {1, 2, 5, 7, 8, 9, 9, 11, 15, 16, 20}. Sebagai contoh data yang akan dicari adalah 5 dinyatakakan sebagai **`x`** dan penanda bila data ketemu adalah **`found`**. Proses pencarian dapat dilakukan seperti berikut.

```
int l{0};
int r{10};
bool found{false};

while(l <= r && !found)
{
    m = l + (r - l) / 2;

    if(x == arr[m])
        found = true;

    if(x < arr[m])
        r = m - 1;
    else if(x > arr[m])
        l = m + 1;
}
```

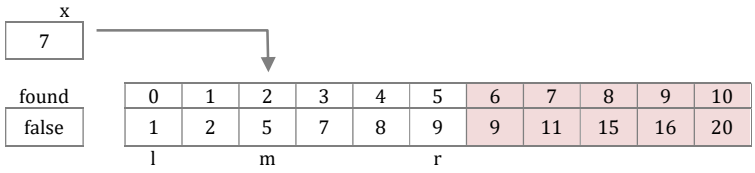
Dari potongan kode di atas terdapat 3 variabel lagi yaitu **`l`**, **`m`**, dan **`r`** masing – masing digunakan sebagai penanda posisi kiri (**`l`**), tengah (**`m`**), dan kanan (**`r`**) sebagai batas pencarian. Proses pencarian akan dilakukan selama posisi index yang ditunjuk **`l`** <= **`r`**. Proses pencocokan selalu mengacu terhadap data pada posisi **`m`** di mana **`x`** == **`arr[m]`**.

Berdasar ilustrasi di atas maka data ditemukan pada index yang ke-3. Ilustrasi visual dari proses pencarian di atas adalah sebagai berikut.

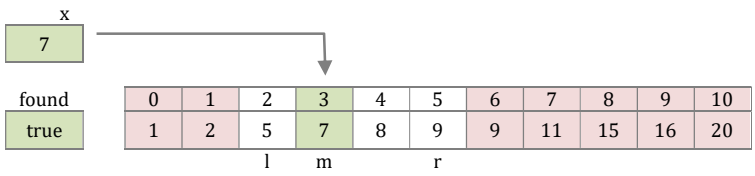
Perulangan pertama



Perulangan Kedua



Perulangan Ketiga



4.2.2.1 Binary Search dengan Rekursi

Selain menggunakan perulangan seperti ilustrasi di atas binary search dapat pula dibuat dengan rekursi. Berikut ini adalah contoh fungsi rekursif untuk binary search.

```
bool binsearch(int n[], int x, int l, int r)
{
    int m = l + (r - l) / 2;

    if(x == n[m])
       return true;

    if(x < n[m])
       return binsearch(x, l, m - 1);

    if(x > n[m])
       return binsearch(x, m + 1, r);

    if(l > r)
       return false;
}
```

4.3 Standard Library `std::find()`

Dalam praktek pemrograman sesungguhnya seorang pemrogram C++ tidak harus mengimplementasikan sendiri algoritma untuk pencarian. C++ menyediakan beberapa fungsi dalam *standard library* untuk keperluan pencarian, diantaranya yang sering digunakan dan relevan dengan pembahasan di bab ini adalah **`std::find()`** dan **`std::find_if()`**. Definisi kedua fungsi tersebut ada dalam *header* **`<algorithm>`**.

Pembahasan lebih lanjut mengenai definisi dan penggunaan kedua fungsi di atas akan diuraikan pada bab berikutnya tentang *Standard Template Library* (*STL*) (§10.4)

4.4 Contoh Kasus

Terdapat sebuah array dengan ukuran maksimal 20. Elemen array diisi dengan input oleh pengguna saat program berjalan. Program harus bisa melakukan pencarian terhadap elemen array dan bila ketemu maka harus menampilkan semua posisi index-nya bila terdapat lebih dari satu elemen yang sama nilainya.

Penyelesaian

```
/**
 * program sequential search
 * dapat menampilkan semua index data yang ditemukan
 *
*/

#include <iostream>

using namespace std;

const int MAX_SIZE{20}

int data[MAX_SIZE];   // array data
int idx[MAX_SIZE];    // array untuk menyimpan index elemen yang ditemukan
int count{0};         // counter, menghitung ada berapa banyak data yang ditemukan

void search(int x);

int main(void) {
	int n;

	cout << "jumlah data: ";
	cin << n;

	for(auto i = 0; i < n; ++i) {
		cout << "data ke-<< i << "i: ";
		cin >> data[i];
	}

	int x;
	cout << "cari: ";
	cin >> x;

	search(x);
	// jika counter > 0, berarti ada data yang ditemukan
	if(count > 0)
	{
		cout << "ditemukan pada index: ";
		for(auto i = 0; i < count; ++i)
		{
			cout << idx[i] << ", ";
		}
	}
	else
	{
		Cout << "data tidak ditemukan\n";
	}

	return 0;
}

void search(int x) {
	for(auto i = 0; i < n; ++i)
```

```
        {
                // jika x ditemukan pada data[i]
                if(x == data[i])
                {
                        // simpan index i ke array idx
                        idx[count++] = i;
                }
        }
}
```

4.5 Latihan

1. Pencarian nomor halaman buku dapat dilakukan dengan metode binary search. Jelaskan dengan contoh.
2. Buatlah program *phonebook* yang berisi data nama dan nomor telepon. *Phonebook* memiliki kapasitas maksimal 50.
   Ketentuan:
   Program memiliki 2 *array*, 1 *array* untuk menyimpan nama dan 1 lagi untuk menyimpan *nomor* telepon. Di dalam program terdapat fungsi untuk melakukan penambahan dan pencarian data.