

## **LAB. DESAIN DAN ANALISIS ALGORITMA**



*Disusun Oleh:*

ACHMAD RIYADH SEMIT

NPM: 232310040

**FAKULTAS INFORMATIKA & PARIWISATA**

**PRODI TEKNOLOGI INFORMASI**

**INSTITUT BISNIS DAN INFORMATIKA KESATUAN**

*Jl. Rangka Gading No.01, Gudang, Kecamatan Bogor Tengah, Kota Bogor, Jawa*

*Barat 16123*

## DAFTAR ISI

<b>DAFTAR ISI</b> .....	i
<b>BAB I PENDAHULUAN</b> .....	1
A. Latar Belakang .....	1
B. Tujuan.....	
<b>BAB II SORTING</b> .....	2
A. Pengertian Sorting .....	2
B. Jenis-Jenis Sorting .....	3
<b>BAB III SEARCHING</b> .....	14
A. Pengertian.....	14
B. Jenis-Jenis Searching .....	14
<b>BAB IV CLASS</b> .....	20
A. Pengertian Class .....	20
B. Setter dan Getter pada Class.....	21
C. Konsep OOP pada Class.....	22
<b>BAB IV PENUTUP</b> .....	<b>Error! Bookmark not defined.</b>
A. Kesimpulan.....	25
<b>DAFTAR PUSTAKA</b> .....	26

# **BAB I**

## **PENDAHULUAN**

### **A. Latar Belakang**

Pada era digital seperti saat ini, pemrosesan data merupakan bagian integral dari hampir semua bidang, termasuk rekayasa perangkat lunak, ilmu komputer, dan kecerdasan buatan. Dalam pengembangan perangkat lunak, konsep seperti sorting, searching, serta penggunaan class dan objek dalam paradigma pemrograman berorientasi objek (OOP) sangatlah penting. Dalam makalah ini, kami akan membahas secara detail tentang penggunaan konsep-konsep tersebut dalam bahasa pemrograman C++.

### **B. Tujuan**

Tujuan utama dari makalah ini adalah:

1. Menguraikan Konsep Sorting dan Searching: Membahas algoritma-algoritma dasar untuk mengurutkan dan mencari data, serta memberikan contoh implementasi dalam bahasa C++.
2. Memperkenalkan Konsep Class dan OOP: Menjelaskan konsep dasar kelas dan objek dalam pemrograman berorientasi objek (OOP) dan bagaimana cara mengimplementasikannya dalam bahasa C++.
3. Mengilustrasikan Penggunaan Setter dan Getter: Menunjukkan bagaimana menggunakan metode setter dan getter untuk mengatur dan mengambil nilai dari variabel-variabel dalam kelas.
4. Memberikan Contoh Implementasi: Dapat memahami dengan jelas cara mengaplikasikan konsep-konsep yang dibahas dalam makalah ini.

## **BAB II**

### **SORTING**

#### **A. Pengertian Sorting**

Sorting berasal dari kata dasar "sort", dalam bahasa Indonesia berarti mengurutkan. Jadi, metode sorting pada C++ disini adalah mengurutkan data berdasarkan ascending (dari nilai terkecil ke nilai terbesar) ataupun descending (dari nilai terbesar ke nilai terkecil). Penggunaan sorting ini sangat berguna ketika di dalam program kita terdapat metode searching, dimana salah satu metode searching yang bernama binary searching. Membutuhkan metode sorting agar metode binary searching dapat berjalan dengan semestinya.

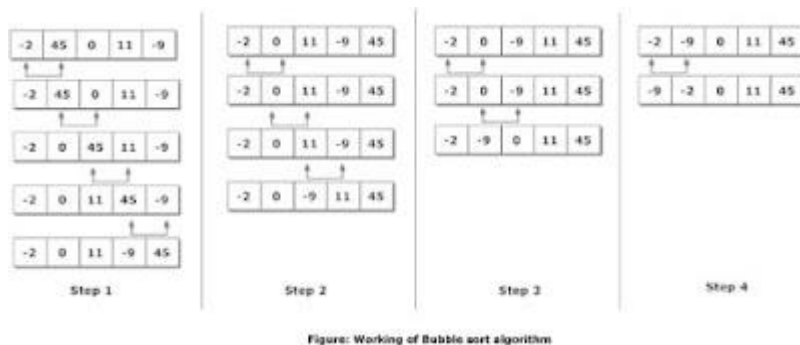
Di sekitar kita, banyak sekali contohnya. Seperti kamus bahasa, diurutkan berdasarkan abjad mulai dari A hingga Z. Selain itu, folder atau file pada komputer kita dimana diurutkan berdasarkan atribut tanggal, nama, hingga ukuran file atau folder tersebut. Jadi, metode sorting ini tentunya sangat berguna untuk pembuatan program dari bahasa C++. Menggunakan metode sorting ini dapat mempersingkat waktu kita dalam membuat program dan menghemat baris kode

#### **B. Jenis-Jenis Metode Sorting**

Jenis - jenis metode sorting ini ada banyak sekali. Lebih banyak di banding jenis - jenis metode searching. Dibawah ini beberapa jenis - jenis metode sorting yang dapat diketahui :

- Bubble Sort
- Selection Sort
- Insertion Sort
- Quick Sort
- Merge Sort

## Metode Bubble Sort



Metode ini terinspirasi dari gelembung air, jadi metode ini dinamakan bubble sort. Cara kerjanya seperti di atas, dimana data pertama dan kedua akan dibandingkan terlebih dahulu. Apabila dari kedua lebih besar maka, tidak terjadi pergeseran dan akan dilanjutkan dengan perbandingan data ketiga dan selanjutnya. Apabila data yang dibandingkan lebih kecil dari data yang berada disampingnya. Maka data terkecil hasil perbandingan akan digeser pada sisi kiri dan perbandingan akan terus berlanjut.

## Contoh Code implementasi Bubble Sort

```
1 #include <iostream>
2 using namespace std;
3
4 void bubbleSort(int arr[], int n) {
5     for (int i = 0; i < n-1; i++) {
6         for (int j = 0; j < n-i-1; j++) {
7             if (arr[j] > arr[j+1]) {
8                 int temp = arr[j];
9                 arr[j] = arr[j+1];
10                arr[j+1] = temp;
11            }
12        }
13    }
14 }
15
16 int main() {
17     int arr[] = {64, 34, 25, 12, 22, 11, 90};
18     int n = sizeof(arr)/sizeof(arr[0]);
19     bubbleSort(arr, n);
20     cout << "Sorted array: \n";
21     for (int i=0; i < n; i++) {
22         cout << arr[i] << " ";
23     }
24     cout << endl;
25     return 0;
26 }
27
```

Sorted array:  
11 12 22 25 34 64 90

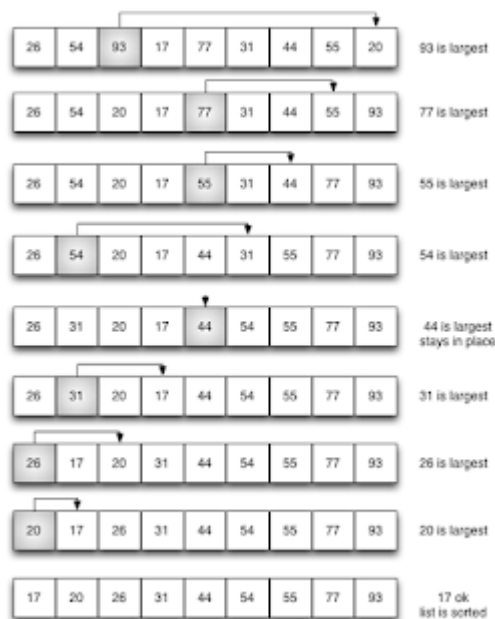
Process exited after 0.0584 seconds with return value 0  
Press any key to continue . . .

Report Window  
Compiler Resources Compile Log Debug Find Results Close  
Abort Compilation - Compilation Time: 1,20s

Line: 7 Col: 37 Sel: 0 Lines: 27 Length: 615 Insert Done parsing in 0,219 seconds

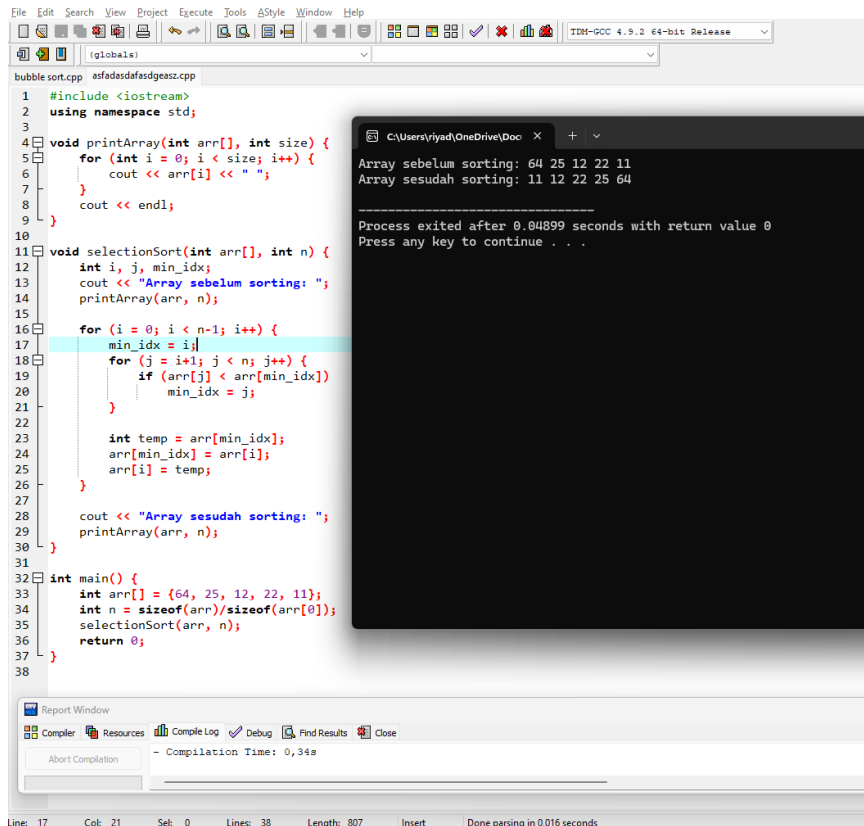
- Fungsi bubbleSort menerima array integer dan panjangnya sebagai argumen.
- Dua loop bersarang digunakan untuk membandingkan setiap pasangan elemen dalam array.
- Jika elemen pertama dari pasangan lebih besar dari elemen kedua, mereka ditukar.
- Proses ini diulangi sampai seluruh array terurut.
- Di dalam main(), array yang tidak terurut diberikan sebagai input ke fungsi bubbleSort, kemudian array yang sudah terurut dicetak.

### Metode Selection Sort



Kalau untuk metode selection sort ini, berbeda dengan bubble sort. Perbedaannya terletak pada dimana data terbesar berhenti. Metode selection sort, data 1 pertama akan di bandingkan dengan data kedua, apabila data tersebut bernilai lebih besar di banding data pertama. Maka, data tersebut akan bergeser dan akan berhenti hingga data  $n+1$  lebih besar dari data tersebut

## Contoh Code menggunakan Selection Sort



The screenshot shows a C++ IDE with a file named `bubble sort.cpp` (though the code is Selection Sort). The code defines a `printArray` function, a `selectionSort` function, and a `main` function. The `main` function initializes an array `arr = {64, 25, 12, 22, 11}` and calls `selectionSort(arr, n)`. The `selectionSort` function uses two nested loops to find the minimum element and swap it with the first element of the unsorted portion. The `printArray` function prints the array elements. The output window shows the array before and after sorting, and the process exit message.

```
1 #include <iostream>
2 using namespace std;
3
4 void printArray(int arr[], int size) {
5     for (int i = 0; i < size; i++) {
6         cout << arr[i] << " ";
7     }
8     cout << endl;
9 }
10
11 void selectionSort(int arr[], int n) {
12     int i, j, min_idx;
13     cout << "Array sebelum sorting: ";
14     printArray(arr, n);
15
16     for (i = 0; i < n-1; i++) {
17         min_idx = i;
18         for (j = i+1; j < n; j++) {
19             if (arr[j] < arr[min_idx])
20                 min_idx = j;
21         }
22
23         int temp = arr[min_idx];
24         arr[min_idx] = arr[i];
25         arr[i] = temp;
26     }
27
28     cout << "Array sesudah sorting: ";
29     printArray(arr, n);
30 }
31
32 int main() {
33     int arr[] = {64, 25, 12, 22, 11};
34     int n = sizeof(arr)/sizeof(arr[0]);
35     selectionSort(arr, n);
36     return 0;
37 }
38
```

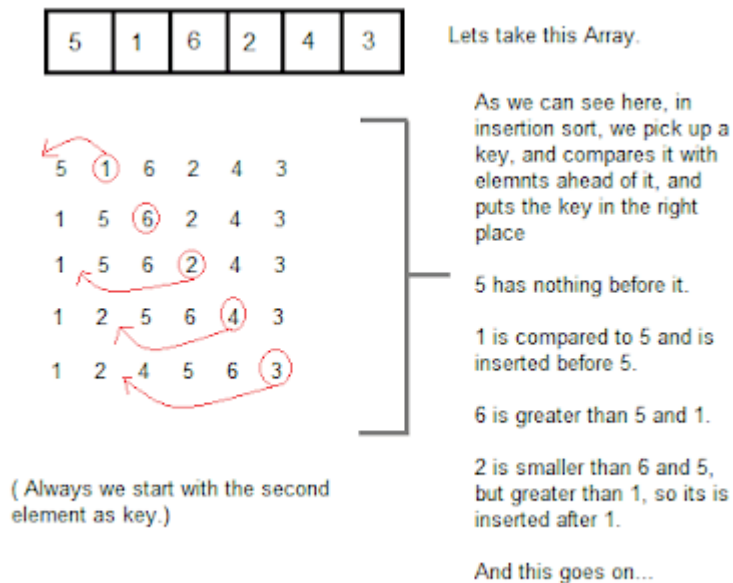
Output:

```
Array sebelum sorting: 64 25 12 22 11
Array sesudah sorting: 11 12 22 25 64

Process exited after 0.04899 seconds with return value 0
Press any key to continue . . .
```

- Fungsi `selectionSort` menerima array integer dan panjangnya sebagai argumen.
- Di dalamnya, dua loop digunakan.
- Loop pertama digunakan untuk melintasi seluruh array.
- Pada setiap iterasi loop pertama, loop kedua digunakan untuk mencari elemen minimum dalam array yang belum diurutkan.
- Setelah menemukan elemen minimum, elemen tersebut ditukar dengan elemen pertama dari array yang belum diurutkan. Proses ini diulangi sampai seluruh array terurut.
- Di dalam `main()`, array yang tidak terurut diberikan sebagai input ke fungsi `selectionSort`, kemudian array yang sudah terurut dicetak.
- Fungsi `printArray` digunakan untuk mencetak array.
- Sebelum proses sorting dimulai, array dicetak menggunakan fungsi `printArray`.
- Setelah proses sorting selesai, array dicetak lagi untuk menampilkan hasil sorting.

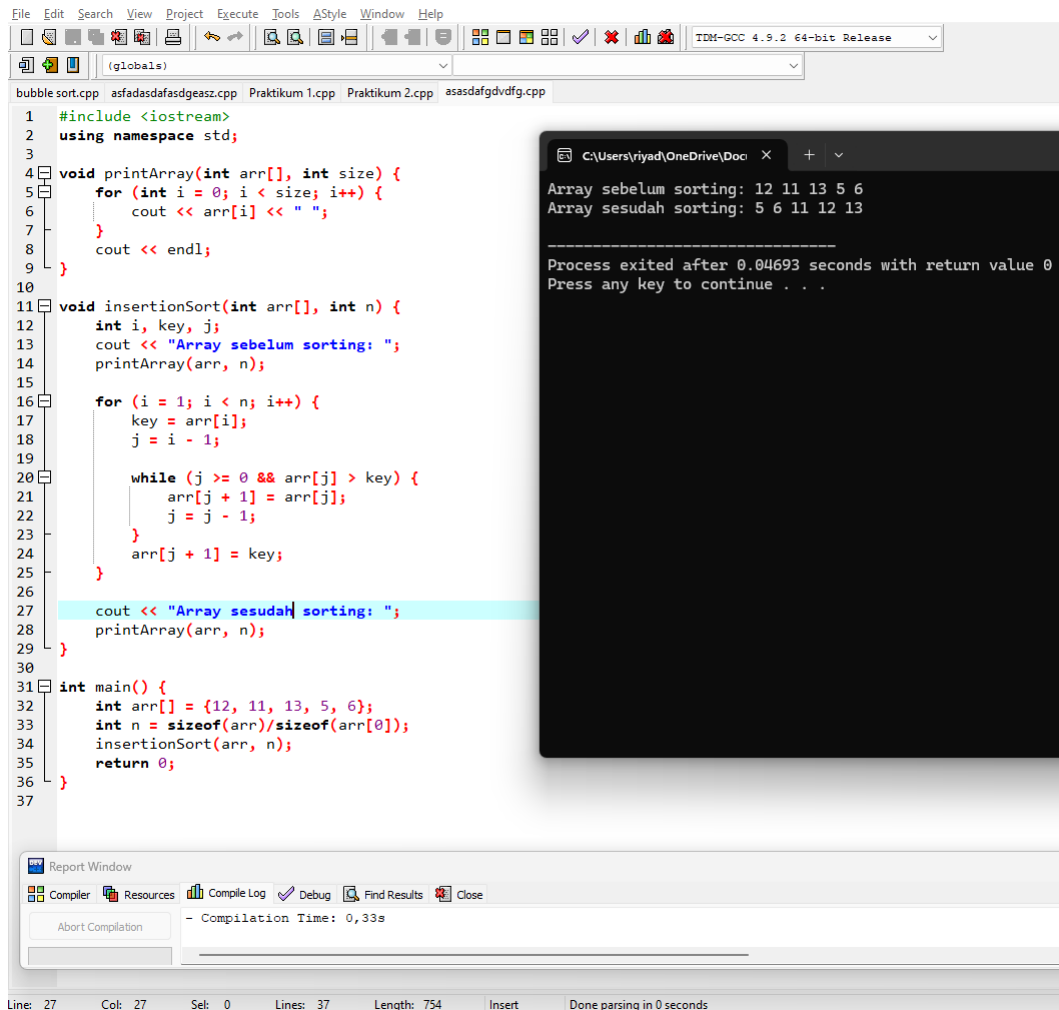
## Metode Insertion Sort



Metode insertion sort ini cara kerjanya, seperti gambar di atas. Dimana data terkecil akan di kelompokkan ke sisi kiri sehingga terjadi pergeseran pada sisi kanan. Data ini selalu dibandingkan setelah pergeseran. Contoh, Pada gambar terdapat angka 2 dimana akan dibandingkan dengan 6 selanjutnya terjadi pergeseran. Setelah itu, dibandingkan kembali dengan 5 dan 6 lalu hasilnya lebih kecil maka terjadi pergeseran dimana 5 lebih besar dari 2. Selanjutnya, 2 dibandingkan dengan 1 dan 5 nah di sini sudah tidak ada data lagi karena data terkecil adalah 1. Jadi, 2 akan berhenti setelah angka 1.



## Contoh Code menggunakan Insertion Sort



The screenshot shows a C++ IDE with a source code editor on the left and a terminal window on the right. The source code implements an Insertion Sort algorithm. It includes `<iostream>` and uses the `std` namespace. A `printArray` function is defined to print an array. The `insertionSort` function takes an array and its size, prints the array before sorting, then iterates from index 1 to `n-1`. For each element, it finds the correct position in the sorted part of the array using a `while` loop and shifts elements to the right. Finally, it prints the array after sorting. The `main` function initializes an array `{12, 11, 13, 5, 6}` and calls `insertionSort`. The terminal window shows the output: 'Array sebelum sorting: 12 11 13 5 6', 'Array sesudah sorting: 5 6 11 12 13', and a message indicating the process exited after 0.04693 seconds.

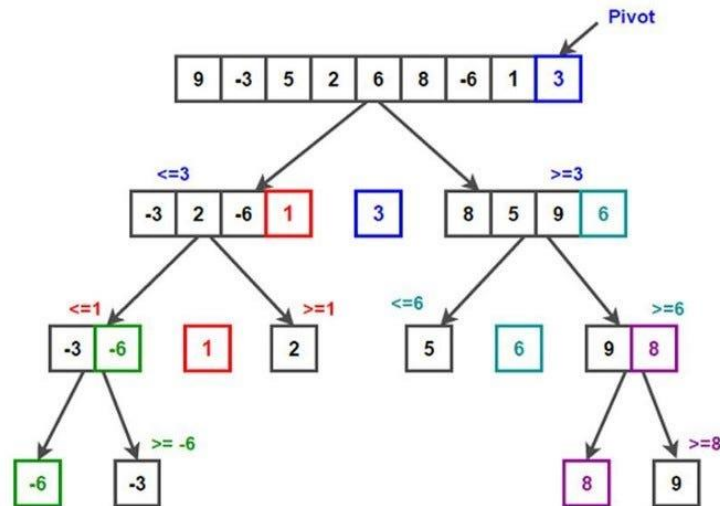
```
1 #include <iostream>
2 using namespace std;
3
4 void printArray(int arr[], int size) {
5     for (int i = 0; i < size; i++) {
6         cout << arr[i] << " ";
7     }
8     cout << endl;
9 }
10
11 void insertionSort(int arr[], int n) {
12     int i, key, j;
13     cout << "Array sebelum sorting: ";
14     printArray(arr, n);
15
16     for (i = 1; i < n; i++) {
17         key = arr[i];
18         j = i - 1;
19
20         while (j >= 0 && arr[j] > key) {
21             arr[j + 1] = arr[j];
22             j = j - 1;
23         }
24         arr[j + 1] = key;
25     }
26
27     cout << "Array sesudah sorting: ";
28     printArray(arr, n);
29 }
30
31 int main() {
32     int arr[] = {12, 11, 13, 5, 6};
33     int n = sizeof(arr)/sizeof(arr[0]);
34     insertionSort(arr, n);
35     return 0;
36 }
37
```

Report Window  
Compiler Resources Compile Log Debug Find Results Close  
- Compilation Time: 0,33s  
Line: 27 Col: 27 Sel: 0 Lines: 37 Length: 754 Insert Done parsing in 0 seconds

- Langkah pertama adalah mencetak array sebelum proses sorting dimulai dengan menggunakan fungsi `printArray`.
- Iterasi dimulai dari indeks kedua ( $i = 1$ ) hingga indeks terakhir ( $n - 1$ ). Pada setiap iterasi, elemen yang dipilih (`key`) disimpan dalam variabel `key`.
- Kemudian, dengan menggunakan loop `while`, algoritma mencari posisi yang tepat untuk menyisipkan `key` dalam bagian array yang sudah diurutkan. Jika elemen yang dibandingkan lebih besar dari `key`, elemen tersebut digeser satu posisi ke kanan.
- Setelah menemukan posisi yang tepat untuk `key`, `key` dimasukkan ke dalam posisi tersebut.
- Setelah iterasi selesai, array yang telah diurutkan dicetak menggunakan fungsi `printArray`.

## Metode Quick Sort

Quick Sort adalah algoritma pengurutan yang berdasarkan pada pendekatan "divide and conquer" (bagi dan kuasai). Ia memilih elemen yang disebut sebagai "pivot" dan membagi array menjadi dua bagian: satu dengan elemen yang lebih kecil dari pivot, dan yang lain dengan elemen yang lebih besar dari pivot. Kemudian, algoritma dijalankan secara rekursif untuk kedua bagian tersebut



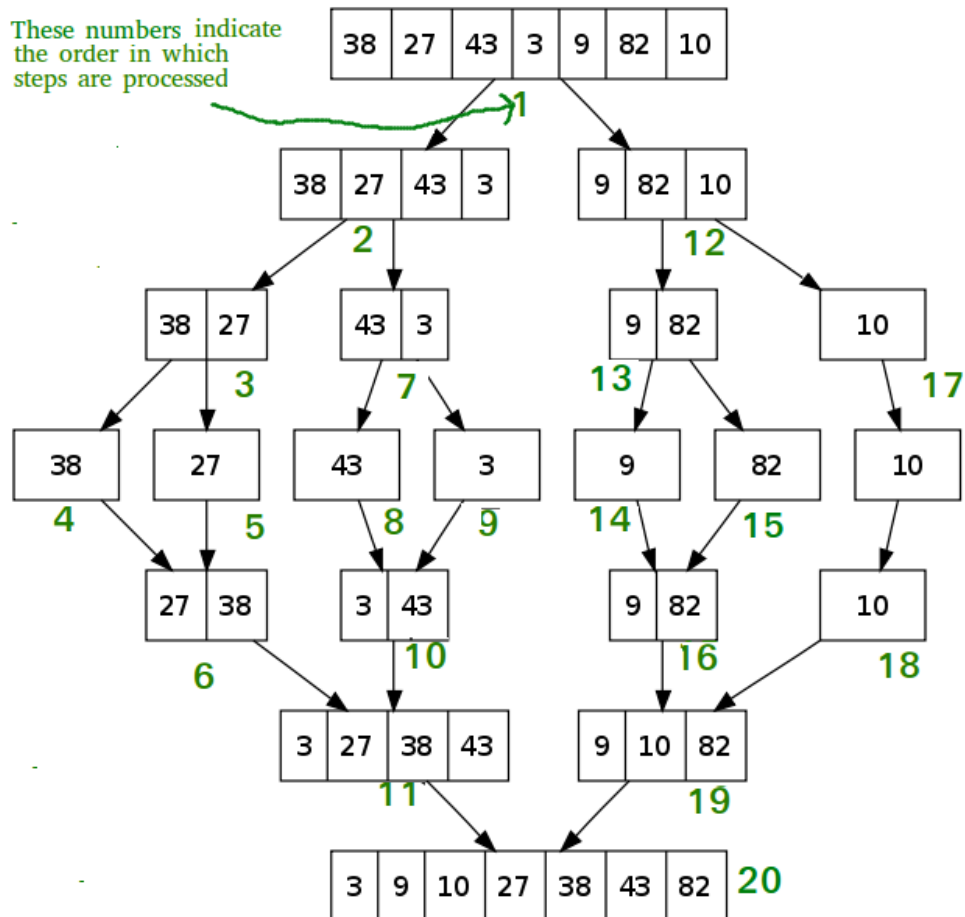
## Contoh Code Menggunakan Quick Sort

```
1 #include <iostream>
2 using namespace std;
3
4 // void swap(int &a, int &b) {
5 //     int temp = a;
6 //     a = b;
7 //     b = temp;
8 // }
9
10 int partition(int arr[], int l, int h) {
11     int pivot = arr[h];
12     int i = l - 1;
13
14     for (int j = l; j < h; j++) {
15         if (arr[j] >= pivot) {
16             i++;
17             swap(arr[i], arr[j]);
18         }
19     }
20     swap(arr[i + 1], arr[h]);
21     return i + 1;
22 }
23
24 void quickSort(int arr[], int l, int h) {
25     if (l < h) {
26         int pivotIndex = partition(arr, l, h);
27         quickSort(arr, l, pivotIndex - 1);
28         quickSort(arr, pivotIndex + 1, h);
29     }
30 }
31
32 void view(int arr[], int size) {
33     for (int i = 0; i < size; i++) {
34         cout << arr[i] << " ";
35     }
36     cout << endl;
37 }
38
39 int main() {
40     int arr[] = {67, 90, 87, 34, 11, 16, 42, 51, 7};
41     int n = sizeof(arr) / sizeof(arr[0]);
42     cout << "Data yang akan di sort: ";
43     view(arr, n);
44
45     cout << "Quick Sort: ";
46     quickSort(arr, 0, n - 1);
47     view(arr, n);
48     cout << "Data setelah diurutkan: ";
49     view(arr, n);
50     return 0;
51 }
```

- Fungsi swap digunakan untuk menukar nilai dua variabel. Di sini, kami menggunakan referensi untuk variabel a dan b, sehingga perubahan yang terjadi pada variabel tersebut akan tercermin di luar fungsi.
- Fungsi partition digunakan untuk membagi array ke dalam dua bagian berdasarkan pivot yang dipilih. Di sini, kami menggunakan pivot sebagai elemen terakhir dari array. Fungsi ini mengembalikan indeks pivot setelah membagi array.
- Fungsi quickSort merupakan implementasi rekursif algoritma Quick Sort. Di sini, kami memeriksa apakah indeks awal l lebih kecil dari indeks akhir h. Jika ya, maka kami memanggil partition untuk membagi array dan kemudian secara rekursif memanggil quickSort untuk kedua bagian array tersebut.
- Fungsi view digunakan untuk mencetak isi array.
- Di dalam fungsi main, array awal diinisialisasi dan kemudian dicetak sebelum pengurutan. Kemudian, dilakukan pengurutan array menggunakan algoritma Quick Sort dengan cara iteratif, dimana setiap iterasi array diurutkan hingga elemen ke-i. Setelah selesai, array yang sudah diurutkan dicetak.

## Metode Merge Sort

Merge Sort juga adalah algoritma "divide and conquer". Ia membagi array menjadi dua bagian, mengurutkan masing-masing bagian secara terpisah, dan kemudian menggabungkan kembali dua bagian yang telah diurutkan tersebut.



## Contoh Code menggunakan Merge Sort

```
1 #include <iostream>
2 using namespace std;
3
4 void merge(int arr[], int kiri, int tengah, int kanan) {
5     int n1 = tengah - kiri + 1;
6     int n2 = kanan - tengah;
7
8     int l[n1], r[n2];
9
10    for (int i = 0; i < n1; i++)
11        l[i] = arr[kiri + i];
12    for (int j = 0; j < n2; j++)
13        r[j] = arr[tengah + 1 + j];
14
15    int i = 0, j = 0, k = kiri;
16    while (i < n1 && j < n2) {
17        if (l[i] <= r[j]) {
18            arr[k] = l[i];
19            i++;
20        } else {
21            arr[k] = r[j];
22            j++;
23        }
24        k++;
25    }
26    while (i < n1) {
27        arr[k] = l[i];
28        i++;
29        k++;
30    }
31    while (j < n2) {
32        arr[k] = r[j];
33        j++;
34        k++;
35    }
36 }
37
38 int partisi(int arr[], int kiri, int kanan) {
39     int tengah = kiri + (kanan - kiri) / 2;
40
41     cout << "Memecah : ";
42     for (int i = kiri; i <= tengah; i++) {
43         cout << arr[i] << " ";
44     }
45     cout << endl;
46     mergeSort(arr, kiri, tengah);
47
48     cout << "Menggabungkan : ";
49     for (int i = tengah + 1; i <= kanan; i++) {
50         cout << arr[i] << " ";
51     }
52     cout << endl;
53     mergeSort(arr, tengah + 1, kanan);
54
55     cout << "Menggabungkan : ";
56     merge(arr, kiri, tengah, kanan);
57     for (int i = kiri; i <= kanan; i++) {
58         cout << arr[i] << " ";
59     }
60     cout << endl;
61 }
62
63 int main() {
64     int arr[] = {10, 12, 84, 24, 1, 55, 25, 67, 78};
65     int n = sizeof(arr) / sizeof(arr[0]);
66
67     cout << "Data awal : ";
68     for (int i = 0; i < n; i++) {
69         cout << arr[i] << " ";
70     }
71     cout << endl;
72
73     cout << "Proses Sort:\n";
74     mergeSort(arr, 0, n-1);
75
76     cout << "Data setelah diurutkan : ";
77     for (int i = 0; i < n; i++) {
78         cout << arr[i] << " ";
79     }
80     cout << endl;
81 }
```

Output Console:

```
Data awal: 10 12 84 24 1 55 25 67 78 96
Proses Sort:
Memecah : 10 12 84 24 1
Memecah : 55 25 67 78 96
Menggabungkan: 10 12
Memecah : 84 24
Menggabungkan: 60 46 12
Memecah : 24 1
Menggabungkan: 34 1
Menggabungkan: 60 46 24 12 1
Memecah : 55 25 67 78 96
Menggabungkan: 56 25 67
Memecah : 56 25
Menggabungkan: 56 25
Memecah : 56 25
Menggabungkan: 56 25
Memecah : 78 96
Menggabungkan: 67 56 25
Memecah : 78 96
Menggabungkan: 67 56 25
Memecah : 78 96
Menggabungkan: 56 78
Menggabungkan: 56 78 67 56 25
Menggabungkan: 56 78 67 56 25 60 46 24 12 1
Data setelah diurutkan: 10 12 67 60 56 46 25 24 12 1
Process exited after 0.00053 seconds with return value 0
```

- Fungsi swap digunakan untuk menukar nilai dua variabel. Di sini, kami menggunakan referensi untuk variabel a dan b, sehingga perubahan yang terjadi pada variabel tersebut akan tercermin di luar fungsi.
- Fungsi partition digunakan untuk membagi array ke dalam dua bagian berdasarkan pivot yang dipilih. Di sini, kami menggunakan pivot sebagai elemen terakhir dari array. Fungsi ini mengembalikan indeks pivot setelah membagi array.
- Fungsi quickSort merupakan implementasi rekursif algoritma Quick Sort. Di sini, kami memeriksa apakah indeks awal l lebih kecil dari indeks akhir h. Jika ya, maka kami memanggil partition untuk membagi array dan kemudian secara rekursif memanggil quickSort untuk kedua bagian array tersebut.

- Di dalam fungsi main, array awal diinisialisasi dan kemudian dicetak sebelum pengurutan. Kemudian, dilakukan pengurutan array menggunakan algoritma Quick Sort dengan cara iteratif, dimana setiap iterasi array diurutkan hingga elemen ke-i. Setelah selesai, array yang sudah diurutkan dicetak.

## **BAB III**

### **SEARCHING**

#### **A. Pengertian Searching**

Searching merupakan kegiatan mencari data yang akan dibutuhkan. Searching dalam pemrograman dapat dilakukan untuk mencari data yang berada pada memory komputer. Dalam kehidupan sehari-hari kita sering melakukan Searching seperti pada saat mencari data maupun informasi yang ada pada internet. Terdapat macam-macam metode yang dapat digunakan dalam Searching, antara lain:

- Sequential Search
- Binary Search
- 

#### **B. Jenis-Jenis Searching**

##### **Sequential Search**

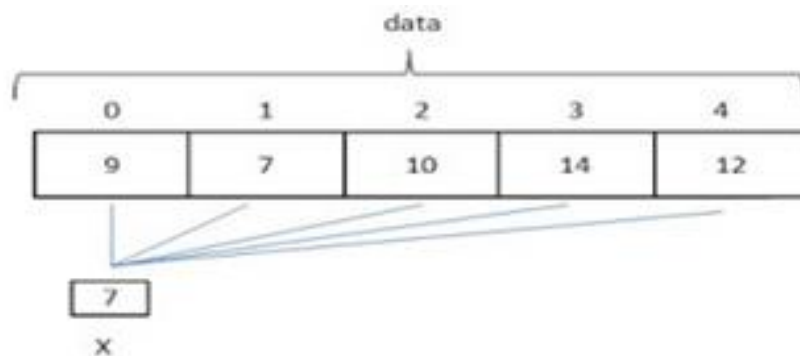
Sequential Search adalah sebuah metode pencarian data dalam array dengan cara membandingkan data yang akan dicari dengan data yang terdapat dalam array secara berurutan. Pencarian data dengan menggunakan metode Sequential Search lebih efektif untuk mencari data dalam posisi tidak terurut atau acak.

Proses Sequential Search dapat dijelaskan sebagai berikut:

1. Menentukan data yang akan dicari.
2. Membaca data array satu per satu secara sekuensial
3. Membaca data mulai dari data pertama sampai dengan data terakhir,

kemudian data yang dicari akan dibandingkan dengan masing-masing data yang terdapat dalam array.

- A. Apabila data yang dicari ditemukan maka kita dapat membuat pernyataan bahwa data telah ditemukan.
- B. Jika data yang dicari tidak ditemukan maka kita dapat membuat pernyataan bahwa data tidak ditemukan



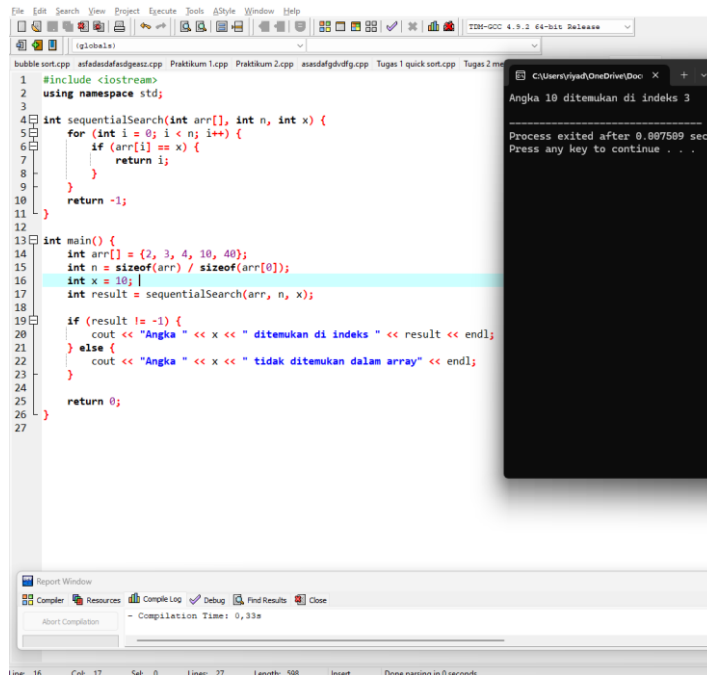
Data yang dicari yaitu 7, lalu disimpan dalam variabel x, kemudian akan dibandingkan satu per satu secara kuensial terhadap data yang terdapat dalam array. Jika ditemukan data pada array yang sama dengan data yang dicari maka data ditemukan.

Kelebihan dan Kekurangan Sequential Searching yaitu:

- Kelebihan Sequential Searching adalah Sequential Searching lebih mudah dalam proses implementasi dalam pemrograman.
- Kekurangan Sequential Searching adalah data yang terdapat dalam suatu array jumlahnya sangat banyak, maka waktu yang diperlukan dalam membandingkan data yang dicari dengan jumlah data dalam suatu array membutuhkan waktu yang lebih lama.



## Contoh Code menggunakan Sequential Search



The screenshot shows a C++ IDE with a source code window and a console window. The source code implements a sequential search function and a main function. The console output shows the result of the search.

```
1 #include <iostream>
2 using namespace std;
3
4 int sequentialSearch(int arr[], int n, int x) {
5     for (int i = 0; i < n; i++) {
6         if (arr[i] == x) {
7             return i;
8         }
9     }
10    return -1;
11 }
12
13 int main() {
14     int arr[] = {2, 3, 4, 10, 40};
15     int n = sizeof(arr) / sizeof(arr[0]);
16     int x = 10;
17     int result = sequentialSearch(arr, n, x);
18
19     if (result != -1) {
20         cout << "Angka " << x << " ditemukan di indeks " << result << endl;
21     } else {
22         cout << "Angka " << x << " tidak ditemukan dalam array" << endl;
23     }
24
25     return 0;
26 }
27
```

Console Output:

```
Angka 10 ditemukan di indeks 3
Process exited after 0.007589 seconds
Press any key to continue . . .
```

Fungsi sequentialSearch menerima array, panjang array, dan angka yang ingin dicari sebagai argumen.

Di dalamnya, kita melakukan iterasi melalui array dan memeriksa setiap elemen.

Jika elemen yang dicari ditemukan, kita mengembalikan indeksnya.

Jika tidak ditemukan, kita mengembalikan -1.

Di dalam main(), kita menggunakan array yang telah diinisialisasi, mencari angka 10, dan mencetak hasil pencarian.

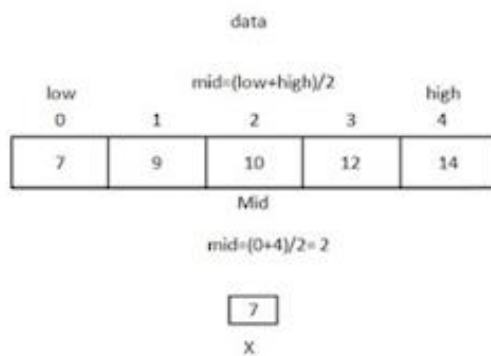
## Binary Search

Metode Binary Search adalah suatu metode pencarian data dengan cara mengelompokkan array menjadi bagian-bagian. Binary Search hanya dapat diterapkan pada data yang telah terurut baik ascending maupun descending dalam suatu array.

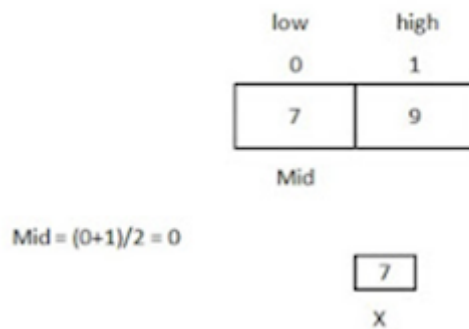
Proses Binary Search dengan data ascending dan telah urut:

1. Membuat perulangan kemudian menentukan posisi low yaitu posisi yang menandakan index paling rendah kemudian menentukan posisi high, setelah itu mencari posisi mid atau tengah =  $(\text{high} + \text{low})/2$

2. Membandingkan data yang dicari dengan nilai yang berada di posisi mid atau tengah.
3. Jika data yang dicari sama dengan nilai yang ada di posisi mid atau tengah berarti data ditemukan.
4. Jika data yang dicari lebih kecil dari nilai yang terdapat pada mid maka pencarian data akan dilakukan dibagian kiri mid dengan melakukan pembandingan. Dengan kondisiposisi high berubah yaitu (mid - 1) dan posisi low tetap.
5. Jika data yang dicari lebih besar dari nilai yang terdapat pada mid maka pencarian data akan dilakukan di bagian kanan dari mid dengan posisi low yang berubah yaitu (mid + 1) dan posisi high tetap.



Setelah menentukan low dan high lalu menentukan mid. perhitungan  $\text{mid} = (\text{low} + \text{high}) / 2$  jadi  $\text{mid} = (0 + 4) / 2$ , artinya mid berada pada data (2). Data yang dicari kemudian ditampung di variabel x lalu dibandingkan dengan data atau nilai yang berada di mid. data yang dicari adalah  $7 < 10$  kemudian pencarian data dilakukan di bagian kiri mid.

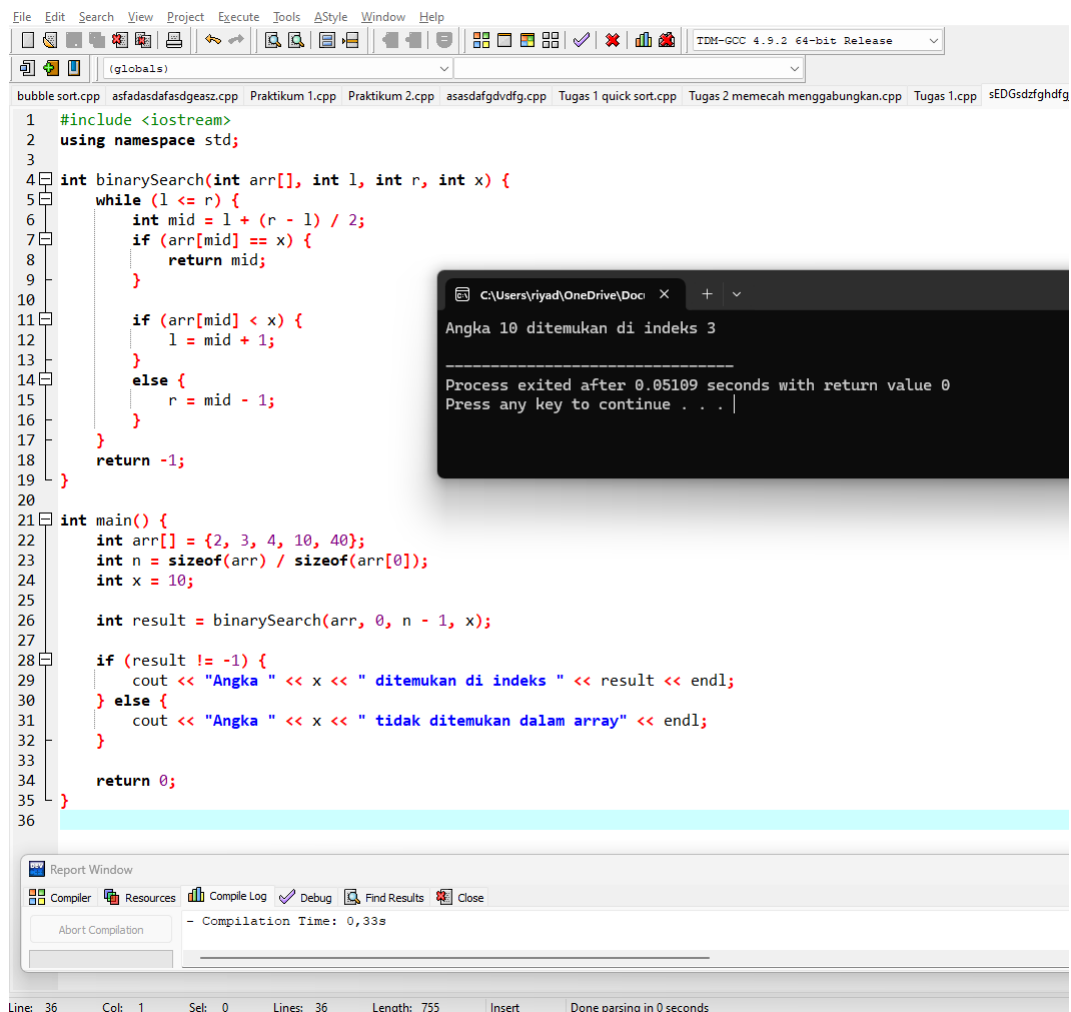


Pencarian di bagian kiri mid masih dalam perulangan yang sama, namun indeksinya dipersempit. Karena data yang dicari lebih kecil dari data mid yang sebelumnya maka posisi high akan berubah yaitu (mid - 1) yang sebelumnya dan low tetap pada posisi yang sama. Kemudian menentukan  $\text{mid} = (0+1)/2 = 0$  artinya mid sekarang terletak di data (0). lalu data yang dicari dibandingkan dengan mid.  $7=7$  artinya data telah ditemukan

#### Kelebihan dan Kekurangan Binary Search:

1. Kelebihan Binary Search adalah tidak perlu membandingkan data yang dicari dengan seluruh data array yang ada, hanya cukup melalui titik tengah kemudian kita bias menentukan ke mana selanjutnya mencari data yang akan dicari.
2. Kekurangan Binary Search adalah penerapannya agak sedikit lebih rumit dikarenakan tidak dapat digunakan pada data array yang masih acak, sehingga harus melakukan sorting terlebih dahulu dalam penerapannya.

## Contoh Code menggunakan Binary Search



```
1 #include <iostream>
2 using namespace std;
3
4 int binarySearch(int arr[], int l, int r, int x) {
5     while (l <= r) {
6         int mid = l + (r - l) / 2;
7         if (arr[mid] == x) {
8             return mid;
9         }
10
11         if (arr[mid] < x) {
12             l = mid + 1;
13         }
14         else {
15             r = mid - 1;
16         }
17     }
18     return -1;
19 }
20
21 int main() {
22     int arr[] = {2, 3, 4, 10, 40};
23     int n = sizeof(arr) / sizeof(arr[0]);
24     int x = 10;
25
26     int result = binarySearch(arr, 0, n - 1, x);
27
28     if (result != -1) {
29         cout << "Angka " << x << " ditemukan di indeks " << result << endl;
30     } else {
31         cout << "Angka " << x << " tidak ditemukan dalam array" << endl;
32     }
33
34     return 0;
35 }
36
```

Angka 10 ditemukan di indeks 3

Process exited after 0.05109 seconds with return value 0  
Press any key to continue . . .

Report Window

Compiler Resources Compile Log Debug Find Results Close

Abort Compilation

Compilation Time: 0,33s

Line: 36 Col: 1 Sel: 0 Lines: 36 Length: 755 Insert Done parsing in 0 seconds

Fungsi `binarySearch` menerima array terurut, indeks kiri, indeks kanan, dan angka yang ingin dicari sebagai argumen.

Pada setiap iterasi, kita mencari elemen tengah dari array.

Jika elemen tengah adalah elemen yang dicari, kita mengembalikan indeksinya.

Jika elemen tengah lebih kecil dari angka yang dicari, kita cari di sebelah kanan.

Jika elemen tengah lebih besar dari angka yang dicari, kita cari di sebelah kiri.

Jika elemen tidak ditemukan, kita mengembalikan -1.

Di dalam `main()`, kita menggunakan array yang telah diinisialisasi, mencari angka 10, dan mencetak hasil pencarian.

## **BAB IV**

### **CLASS**

#### **A. Pengertian Class**

Class adalah cetak biru atau blueprint dari object. Class digunakan hanya untuk membuat kerangka dasar. Yang akan kita pakai nanti adalah hasil cetakan dari class, yakni object.

Sebagai analogi, class bisa diibaratkan dengan laptop atau notebook. Kita tahu bahwa laptop memiliki ciri-ciri seperti merk, memiliki keyboard, memiliki processor, dan beberapa ciri khas lain yang menyatakan sebuah benda tersebut adalah laptop. Selain memiliki ciri-ciri, sebuah laptop juga bisa dikenakan tindakan, seperti: menghidupkan laptop atau mematikan laptop.

Class dalam analogi ini adalah gambaran umum tentang sebuah benda. Di dalam pemrograman nanti, contoh class seperti User, Item, Siswa, Validate, dll.

Di dalam C++, penulisan class diawali dengan keyword class, kemudian diikuti dengan nama dari class. Aturan penulisan nama class sama seperti aturan penulisan variabel dalam C++ (lebih tepatnya aturan identifier), yakni tidak boleh diawali angka dan tidak boleh mengandung spasi.

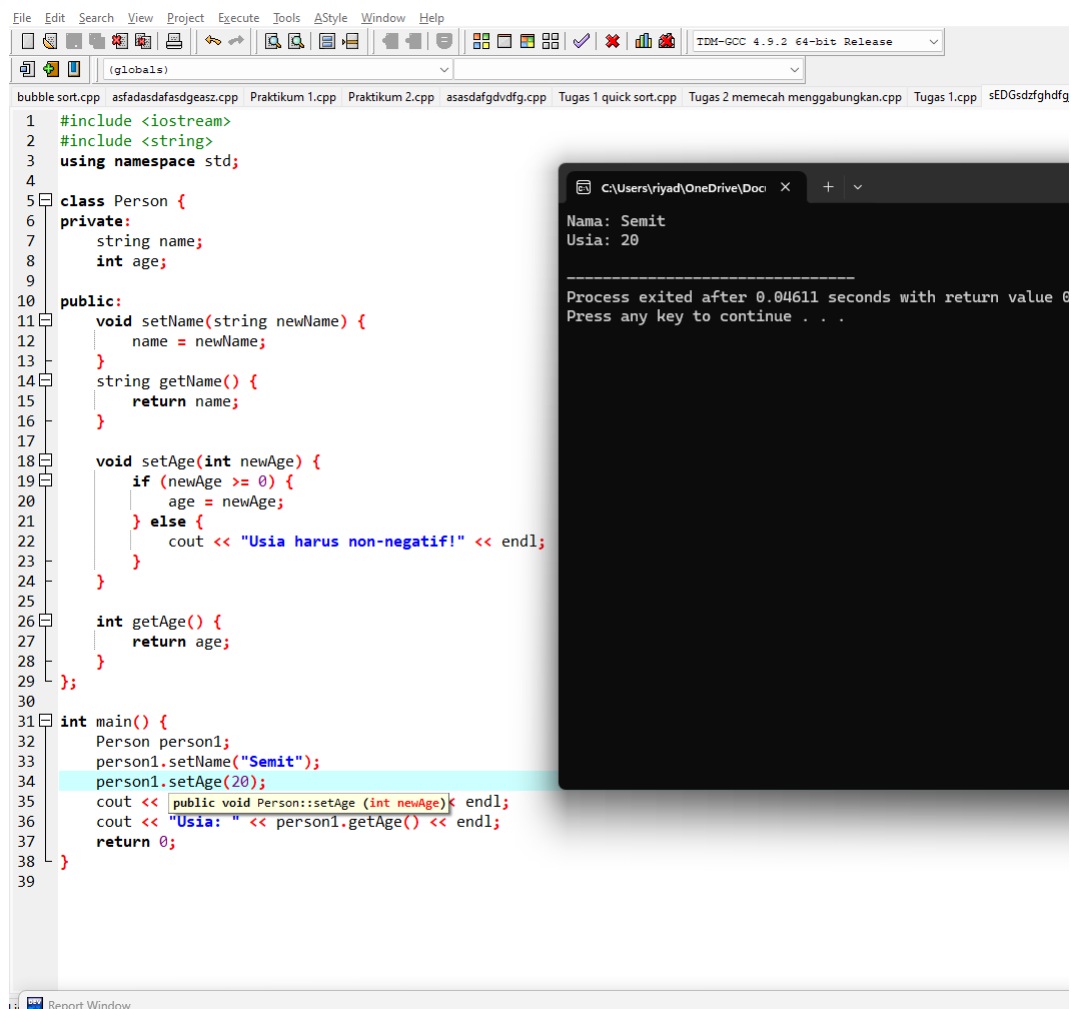
## B. Setter dan Getter pada Class

Setter adalah member function atau method yang dipakai untuk memberikan nilai ke dalam sebuah data member. Sedangkan Getter adalah member function yang dipakai untuk menampilkan nilai data member.

Perlunya penggunaan setter dan getter berangkat dari sebuah tips atau saran bahwa semua data member suatu class harusnya di set sebagai private (bukan public).

Jika semua data member di set sebagai private, maka kita tidak bisa lagi memberikan nilai dan mengakses nilai data member secara langsung.

### Contoh Code Class Menggunakan Setter dan Getter



```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 class Person {
6 private:
7     string name;
8     int age;
9
10 public:
11     void setName(string newName) {
12         name = newName;
13     }
14     string getName() {
15         return name;
16     }
17
18     void setAge(int newAge) {
19         if (newAge >= 0) {
20             age = newAge;
21         } else {
22             cout << "Usia harus non-negatif!" << endl;
23         }
24     }
25
26     int getAge() {
27         return age;
28     }
29 };
30
31 int main() {
32     Person person1;
33     person1.setName("Semit");
34     person1.setAge(20);
35     cout << "Nama: " << person1.getName() << endl;
36     cout << "Usia: " << person1.getAge() << endl;
37     return 0;
38 }
39
```

Process exited after 0.04611 seconds with return value 0  
Press any key to continue . . .

- Kelas Person memiliki dua data anggota pribadi: name dan age.
- Metode setName digunakan untuk mengatur nilai name, sedangkan metode getName digunakan untuk mendapatkan nilai name.
- Metode setAge digunakan untuk mengatur nilai age dengan pengecekan bahwa usia harus non-negatif, sedangkan metode getAge digunakan untuk mendapatkan nilai age.
- Di dalam main(), kita membuat objek dari kelas Person dan menggunakan metode setter untuk mengatur nilai nama dan usia. Kemudian, kita menggunakan **metode** getter untuk mendapatkan dan mencetak nilai nama dan usia.

### C. Konsep OOP dalam Class

#### 1. Enkapsulasi

Enkapsulasi (encapsulation) adalah sebuah metode untuk mengatur struktur class dengan cara menyembunyikan alur kerja dari class tersebut.

Struktur class yang dimaksud adalah data member dan member function. Dengan enkapsulasi, kita bisa membuat pembatasan hak akses kepada data member dan member function, sehingga hanya data member dan member function tertentu saja yang bisa diakses dari luar class. Enkapsulasi ini juga dikenal dengan istilah information hiding.

Dengan menghalangi kode program lain untuk mengubah data member atau member function tertentu, class menjadi lebih terintegrasi dan menghindari kesalahan ketika seseorang mencoba mengubahnya.

Pembatasan hak akses seperti ini sangat berguna ketika membuat program dengan tim. Setiap anggota tim bisa membuat nama data member dan member function yang sama, selama di class berbeda. Programmer yang merancang class kemudian menyediakan member function khusus yang memang ditujukan untuk diakses dari luar class.

Melanjutkan analogi tentang class Laptop, perusahaan pembuat Laptop telah menyediakan member function khusus untuk menghidupkan Laptop, yakni dengan cara menekan tombol on.

Di dalam Laptop sendiri, banyak member function lain yang dijalankan ketika kita menyalakan Laptop, seperti mengirim sinyal booting ke processor, mengirim data dari processor ke memory, dan mengirim sinyal listrik ke LED di monitor, dst. Akan tetapi, ini adalah member function internal Laptop, dimana kita sebagai user tidak perlu memahaminya untuk menghidupkan Laptop.

## 2.Pewarisan

Pewarisan atau inheritance mendefinisikan suatu kelas dan kemudian menggunakannya untuk memabangun hirarki kelas turun yang mana masing – masing turunan mewarisi semua akses kode maupun data kelas dasarnya

Sifat-sifat Inheritance :

- Public  
Penentuan akses berbasis public menyebabkan anggota dari public sebuah kelas utama akan menjadi anggota public kelas turunan.
- Private  
Penentuan akses berbasis private menyebabkan anggota public dari kelas utama akan menjadi anggota protect kelas turunan dan menyebabkan anggota kelas utama menjadi protect kelas turunan.
- Protected  
Penentu akses berbasis protected menyebabkan anggota dari anggota protect dan public dari kelas utama akan menjadi anggota private dari kelas turunan. Anggota private dari kelas utama selalu menjadi anggota private kelas utama.



### 3. Polimorpisme

Dari asal katanya, Polymorphism diambil dari bahasa Yunani, Poly artinya banyak dan Morph artinya bentuk. Polymorphism dalam OOP dapat diartikan Object yang memiliki operasi yang sama tapi perilaku yang berbeda beda.

Misalnya: Burung dan Pinguin yang sama sama memiliki atribut sayap untuk operasi bergerak atau berpindah tempat.

Cara atau perilaku burung untuk berpindah tempat dengan pinguin itu berbeda.

Burung dapat berpindah tempat dengan terbang menggunakan sayapnya, sedangkan pinguin berpindah tempat dengan berenang menggunakan sayapnya.

Contoh lain misalnya dalam menghitung luas sebuah bidang. Misalkan kita punya 2 bidang yakni segitiga dan segiempat. Keduanya memiliki operasi hitung luas, tapi cara menghitung luas segitiga tentu beda dengan hitung luas segiempat.

## **BAB IV**

### **PENUTUP**

#### **A. Kesimpulan**

Makalah ini membahas konsep-konsep dasar dalam pemrograman C++, termasuk sorting, searching, class, dan penggunaan setter dan getter.

Pertama-tama, dalam bagian Sorting, konsep sorting dijelaskan sebagai proses mengurutkan data dalam urutan tertentu. Beberapa metode sorting yang dibahas meliputi Bubble Sort, Selection Sort, Insertion Sort, Quick Sort, dan Merge Sort. Setiap metode diilustrasikan dengan contoh kode program dan penjelasan tentang cara kerjanya.

Selanjutnya, pada bagian Searching, konsep pencarian data dalam array dibahas dengan dua metode, yaitu Sequential Search dan Binary Search. Penjelasan meliputi proses dan contoh penggunaan kedua metode pencarian tersebut.

Kemudian, pada bagian Class, konsep dasar class dalam pemrograman berorientasi objek (OOP) dijelaskan. Diperkenalkan pula penggunaan setter dan getter dalam class untuk mengatur dan mengakses nilai variabel-variabel private dalam class.

Dalam keseluruhan makalah, konsep-konsep tersebut diilustrasikan dengan contoh kode program dan penjelasan yang mendalam. Melalui pembahasan ini, pembaca diharapkan dapat memahami dan mengimplementasikan konsep-konsep dasar dalam pemrograman C++ dengan lebih baik.

## DAFTAR PUSTAKA

*<http://www.sarjanapedia.com/2019/02/metode-sorting-pada-cpp.html>*  
*<https://sinaualpro.blogspot.com/2018/12/searching-dalam-bahasa-pemrograman-c.html>*  
*<https://www.duniailkom.com/tutorial-belajar-oop-c-plus-plus-pengertian-setter-dan-getter/>*  
*<https://oopcpp.wordpress.com/2017/05/26/konsep-polymorphism-dalam-c/>*  
*<https://bahan--ajar-esaunggul-ac-id.webpkgcache.com/doc/-/s/bahan-ajar.esaunggul.ac.id/cio220/wp-content/uploads/sites/1537/2019/12/PPT-UEU-Bahasa-Pemrograman-Pertemuan-14.pptx>*