

Nama : Aisyah Aqilah Rian Vania  
NIM : 21091397002  
Kelas : 2021 B / D4 Manajemen Informatika

## Laporan Individu

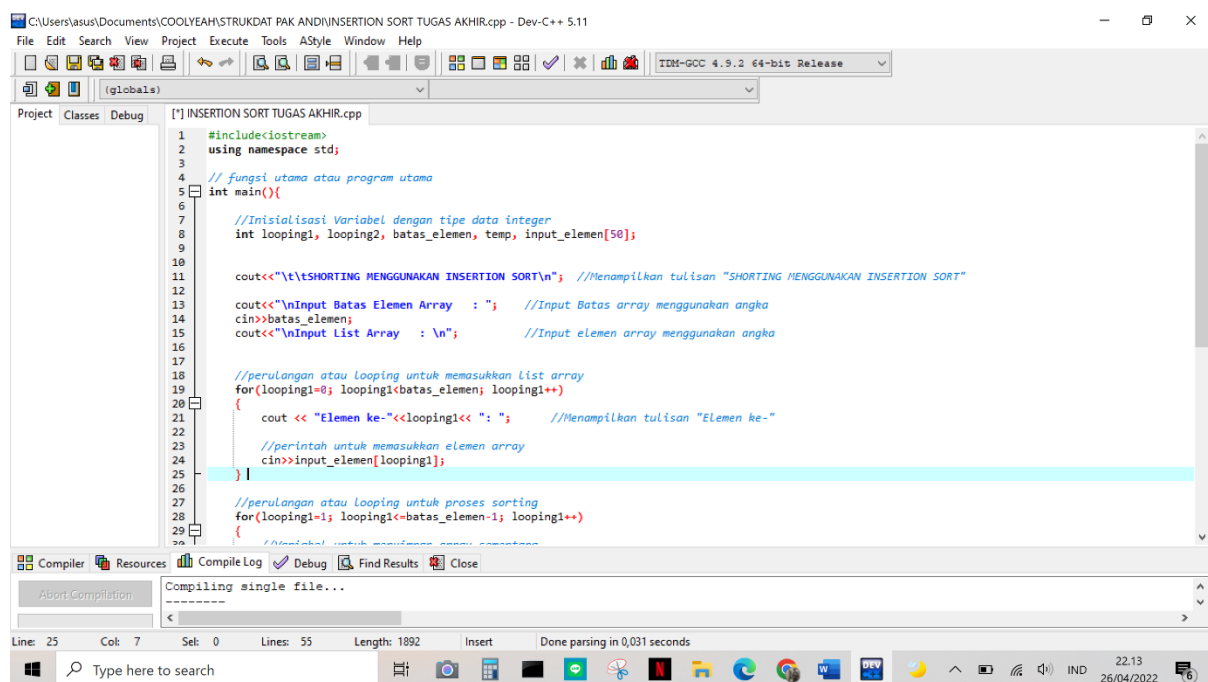
Insertion sort merupakan algoritma yang paling sederhana. Insertion sort adalah sebuah metode pengurutan data dengan menempatkan setiap elemen data pada pisisnya dengan cara melakukan perbandingan dengan data – data yang ada.

Metode insertion sort ini dapat dianalogikan sama seperti mengurutkan kartu, dimana jika suatu kartu dipindah tempatkan menurut posisinya, maka kartu yang lain akan bergeser mundur atau maju sesuai kondisi pemindahanan kartu tersebut. Dalam pengurutan data, metode ini dipakai bertujuan untuk menjadikan bagian sisi kiri array terurutkan sampai dengan seluruh array diurutkan.

Algoritma insertion sort :

1. Pengecekan dimulai dari data ke-1 sampai dengan data ke-n
2. Pengurutan dilakukan dengan cara membandingkan data ke-l (dimulai dari data ke-2 sampai dengan data terakhir)
3. Bandingkan data ke-l tersebut dengan data sebelumnya (i-l). jika lebih kecil maka data tersebut dpat disisipkan ke data awal (depan) sesuai dengan posisi yang seharusnya
4. Lakukan langkah ke-2 dan ke-3 untuk bilangan selanjutnya (  $l = l + 1$  ) sampai didapatkan urutan yang optimal.

## ❖ Laporan Coding



```
C:\Users\asus\Documents\COOLYEAH\STRUKDAT PAK ANDI\INSERTION SORT TUGAS AKHIR.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug [1] INSERTION SORT TUGAS AKHIR.cpp
1 #include<iostream>
2 using namespace std;
3
4 // fungsi utama atau program utama
5 int main()
6 {
7     //Inisialisasi Variabel dengan tipe data integer
8     int looping1, looping2, batas_elemen, temp, input_elemen[50];
9
10
11     cout<<"\t\tSHORTING MENGGUNAKAN INSERTION SORT\n"; //Menampilkan tulisan "SHORTING MENGGUNAKAN INSERTION SORT"
12
13     cout<<"\nInput Batas Elemen Array : "; //Input Batas array menggunakan angka
14     cin>>batas_elemen;
15     cout<<"\nInput List Array : \n"; //Input elemen array menggunakan angka
16
17     //perulangan atau looping untuk memasukkan list array
18     for(looping1=0; looping1<batas_elemen; looping1++)
19     {
20         cout << "Elemen ke-"<<looping1<< " : "; //Menampilkan tulisan "Elemen ke-"
21         //perintah untuk memasukkan elemen array
22         cin>>input_elemen[looping1];
23     }
24
25     //perulangan atau looping untuk proses sorting
26     for(looping1=1; looping1<batas_elemen-1; looping1++)
27     {
28         //Inisialisasi variabel untuk proses sorting
29         looping2 = looping1;
30         while(looping2 > 0 && input_elemen[looping2] < input_elemen[looping2-1])
31         {
32             temp = input_elemen[looping2];
33             input_elemen[looping2] = input_elemen[looping2-1];
34             input_elemen[looping2-1] = temp;
35             looping2--;
36         }
37     }
38
39     cout<<"\nArray yang sudah diurutkan : \n";
40     for(looping1=0; looping1<batas_elemen; looping1++)
41     {
42         cout << input_elemen[looping1] << " ";
43         if(looping1 % 10 == 9) cout << "\n";
44     }
45     cout << "\n";
46     return 0;
47 }
```

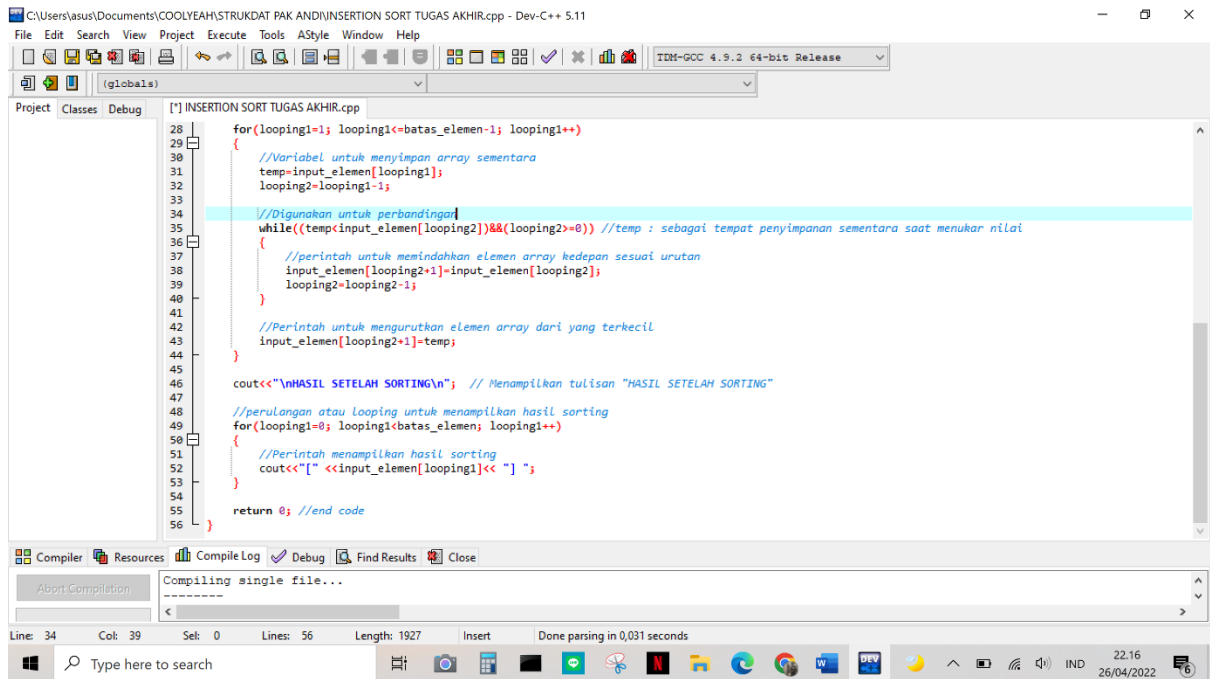
Compiler Resources Compile Log Debug Find Results Close

Compiling single file...

Line: 25 Col: 7 Sel: 0 Lines: 55 Length: 1892 Insert Done parsing in 0,031 seconds

Type here to search

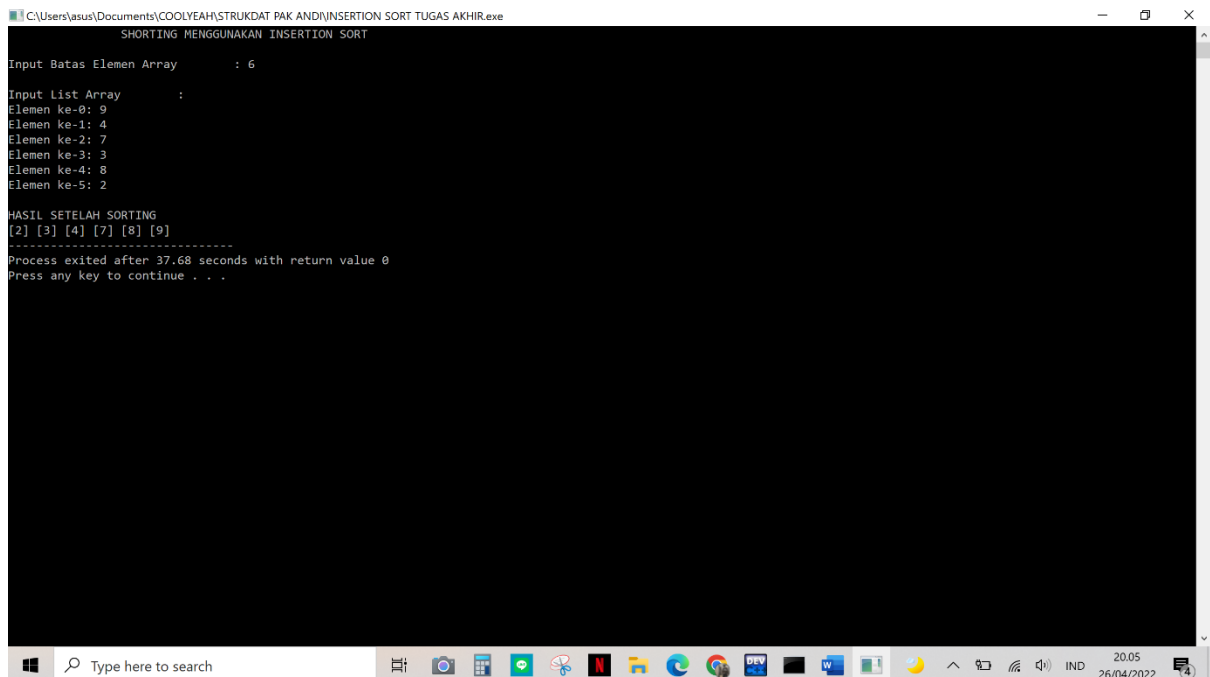
22.13 26/04/2022



```
28 for(looping1=1; looping1<batas_elemen-1; looping1++)
29 {
30     //Variabel untuk menyimpan array sementara
31     temp=input_elemen[looping1];
32     looping2=looping1-1;
33
34     //Digunakan untuk perbandingan
35     while((temp<input_elemen[looping2])&&(looping2>=0)) //temp : sebagai tempat penyimpanan sementara saat menukar nilai
36     {
37         //Perintah untuk memindahkan elemen array kedepan sesuai urutan
38         input_elemen[looping2+1]=input_elemen[looping2];
39         looping2=looping2-1;
40     }
41     //Perintah untuk mengurutkan elemen array dari yang terkecil
42     input_elemen[looping2+1]=temp;
43 }
44
45 cout<<"\\nHASIL SETELAH SORTING\\n"; // Menampilkan tulisan "HASIL SETELAH SORTING"
46
47 //perulangan atau looping untuk menampilkan hasil sorting
48 for(looping1=0; looping1<batas_elemen; looping1++)
49 {
50     //Perintah menampilkan hasil sorting
51     cout<< "[" << input_elemen[looping1] << " ] ";
52 }
53
54 return 0; //end code
55 }
```

Line: 34 Col: 39 Sel: 0 Lines: 56 Length: 1927 Insert Done parsing in 0,031 seconds

## Hasil run coding (output)



```
SHORTING MENGGUNAKAN INSERTION SORT
Input Batas Elemen Array      : 6
Input List Array              :
Elemen ke-0: 9
Elemen ke-1: 4
Elemen ke-2: 7
Elemen ke-3: 3
Elemen ke-4: 8
Elemen ke-5: 2

HASIL SETELAH SORTING
[2] [3] [4] [7] [8] [9]
-----
Process exited after 37.68 seconds with return value 0
Press any key to continue . . .
```

## Proses shorting insertion sort

9	4	7	3	8	2
---	---	---	---	---	---

Pada tabel diatas elemen list array masih belum tersorting

9	4	7	3	8	2
---	---	---	---	---	---

Elemen kedua bertukar dengan elemen pertama karena elemen kedua lebih kecil dari elemen pertama

4	9	7	3	8	2
---	---	---	---	---	---

Elemen ketiga bertukar dengan elemen kedua karena elemen ketiga lebih kecil dari elemen kedua

4	7	9	3	8	2
---	---	---	---	---	---

Elemen keempat bertukar dengan elemen pertama karena elemen keempat lebih kecil dari elemen pertama, kedua, dan ketiga

3	4	7	9	8	2
---	---	---	---	---	---

Elemen kelima bertukar dengan elemen keempat karena elemen kelima lebih kecil dari elemen keempat

3	4	7	8	9	2
---	---	---	---	---	---

Elemen keenam bertukar dengan elemen pertama karena elemen keenam lebih kecil dari elemen pertama, kedua, ketiga dan seterusnya

2	3	4	7	8	9
---	---	---	---	---	---

Pada tabel diatas elemen list array sudah tersortir

## ❖ Kelebihan & Kekurangan

### •Kelebihan Insertion Sort :

1. Sederhana dalam penerapannya.
2. Mangkus dalam data yang kecil.
3. Jika list sudah terurut atau sebagian terurut maka Insertion Sort akan lebih cepat dibandingkan dengan Quicksort.
4. Mangkus dalam data yang sebagian sudah terurut.
5. Lebih mangkus dibanding Bubble Sort dan Selection Sort.
6. Loop dalam pada Inserion Sort sangat cepat, sehingga membuatnya salah satu algoritma pengurutan tercepat pada jumlah elemen yang sedikit.
7. Stabil.

### •Kekurangan Insertion Sort :

1. Banyaknya operasi yang diperlukan dalam mencari posisi yang tepat untuk elemen larik.
2. Untuk larik yang jumlahnya besar ini tidak praktis.
3. Jika list terurut terbalik sehingga setiap eksekusi dari perintah harus memindai dan mengganti seluruh bagian sebelum menyisipkan elemen berikutnya.
4. Membutuhkan waktu  $O(n^2)$  pada data yang tidak terurut, sehingga tidak cocok dalam pengurutan elemen dalam jumlah besar.