LEMBAR ASISTENSI

PRAKTIKUM STRUKTUR DATA

LABORATORIUM TEKNIK KOMPUTER

JURUSAN TEKNIK ELEKTRO FAKULTAS TEKNIK

UNIVERSITAS LAMPUNG



Judul Praktikum : SORTING

Praktikan : Hanzuel Akbar Evansyah (2415061060)

Asisten : Alya Nayra Syafiqa (2315061001)

M. Sidiq Firdaus (2315061118)

M.Azmi Edfa Alhafizh (2315061115)

Riski Jaya Putra (2315061065)

M. Favian Rizki (2315061067)

Kelas : PSTI B

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Catatan** | **Tanggal** | **Paraf** |
| 1.  2.  3. |  |  |  |

Bandar Lampung, September

Alya Nayra Syafiqa

NPM. 2315061001

## JUDUL PERCOBAAN

SORTING

## TUJUAN PERCOBAAN

Adapun tujuan dari percobaan ini adalah sebagai berikut.

1. Mahasiswa dapat memaham fungsi dan jenis jenis sorting
2. Mahasiswa dapat menggunakan berbagai jenis sorting

## III. TEORI DASAR

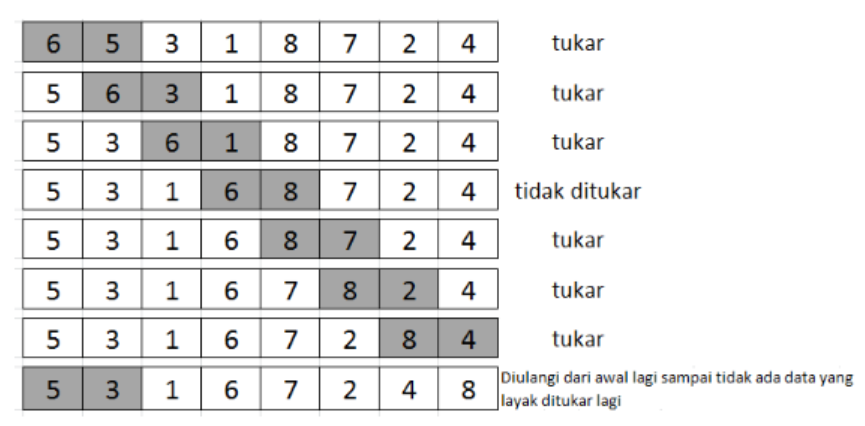
Struktur data merujuk pada metode penyimpanan, pengelompokan, serta penataan informasi di media penyimpanan komputer agar informasi itu bisa dimanfaatkan dengan efisien.1 Sementara itu, informasi merupakan gambaran dari realitas kehidupan sehari-hari. Informasi atau penjelasan mengenai fakta yang disimpan, dicatat atau digambarkan melalui bentuk tulisan, audio, visual, isyarat atau tanda. Struktur Data berperan dalam menyusun informasi sehingga implementasi atau perawatan kerangka program menjadi lebih rapi, pembaca pun bisa memahami arti serta prosedur pembuatan program serta mengenal berbagai jenis informasi. Penggunaan struktur data yang sesuai dalam tahap pemrograman, akan menghasilkan Algoritma yang transparan dan akurat sehingga membuat program secara total lebih mudah Array adalah elemen dari struktur data yang tergolong ke dalam struktur data dasar yang bisa dijelaskan sebagai reservasi tempat memori sementara di komputer. Struktur data yang pas akan sangat memengaruhi kinerja sebuah program, baik dari aspek waktu maupun kapasitas memori. Pilihan struktur data yang sesuai keperluan amat krusial sebab tiap macam struktur data punya ciri khas dan mekanisme operasi yang beragam. Contohnya, pemanfaatan array sesuai untuk menyimpan informasi dengan besaran yang telah diketahui dan pengaksesan langsung ke masing-masing unsur berdasarkan posisi indeks. Akan tetapi, bagi informasi yang kerap berubah dengan penambahan dan pengurangan, pemanfaatan struktur data semisal linked list atau stack bisa lebih cocok lantaran lebih adaptif dalam penanganan unsur-unsur informasinya.

3.1 Pengertian Sorting

Menurut Yahya (2014:135), Sorting merupakan proses penyusunan ulang data yang awalnya tersusun secara acak atau tidak beraturan menjadi rapi dan terstruktur berdasarkan kriteria khusus. Umumnya, penyusunan ini dibedakan menjadi dua jenis, yakni Ascending (penyusunan dari karakter/angka kecil menuju karakter/angka besar) serta Descending (penyusunan dari karakter/angka besar menuju karakter/angka kecil). Menurut Saputra dkk (2010:1) pula, algoritma sorting didefinisikan sebagai prosedur pengurutan sejumlah data berdasarkan nilai kunci spesifik. Penyusunan bisa dilakukan dari nilai terendah ke tertinggi (ascending) atau kebalikannya (descending). Pengurutan data (sorting) adalah suatu proses mengubah susunan data yang acak menjadi teratur sesuai pola tertentu, sehingga menjadi tersusun secara rapi berdasarkan aturan yang ditentukan. Proses pengurutan ini bisa diterapkan melalui cara Ascending dan Descending, serta dimanfaatkan untuk menyusun data berjenis angka maupun karakter.

3.2 Bubble Sort

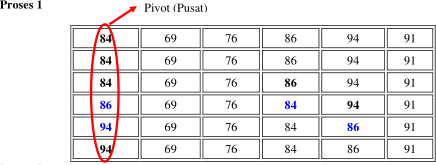
Bubble Sort merupakan teknik pengurutan yang terinspirasi dari sifat gelembung, yakni mengapung yang berarti mengambil nilai terbesar dan menempatkannya di posisi paling ujung kanan. Bubble Sort menyusun data melalui proses membandingkan elemen saat ini dengan elemen selanjutnya. Apabila elemen saat ini lebih besar daripada elemen selanjutnya, maka keduanya ditukar posisi (untuk pengurutan ascending). Sebaliknya, jika elemen saat ini lebih kecil daripada elemen selanjutnya, maka pertukaran dilakukan (untuk pengurutan descending). Algoritma Bubble Sort ini melibatkan perbandingan antar setiap elemen, diikuti dengan penukaran jika ditemukan elemen yang tidak pada urutan yang benar atau keliru. Proses perbandingan akan berlanjut secara berulang hingga tidak terjadi lagi pertukaran antar data.

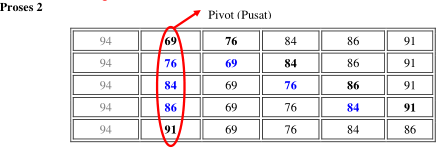
****

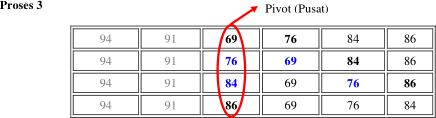
Gambar 3.2 Visualisasi Implementasi Bubble Sort

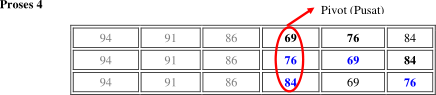
3.3 Exchange Sort

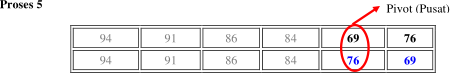
Exchange Sort, yang sering disebut juga sebagai Bubble Sort, merupakan algoritma penyusunan data yang membandingkan seluruh elemen dalam array serta menukarnya apabila susunannya tidak benar. Proses perbandingan ini akan diulang secara berulang hingga tidak diperlukan lagi pertukaran. Pada Exchange Sort, terdapat elemen yang secara tetap berperan sebagai elemen tengah (pivot). Yang membedakan Exchange Sort dari Bubble Sort adalah pendekatan perbandingan antar elemen. Dalam Bubble Sort, elemen di posisi awal atau akhir akan dibandingkan dengan elemen sebelum atau sesudahnya, kemudian elemen tersebut dijadikan pusat (pivot) untuk perbandingan berikutnya.











Gambar 3.3 Visualisasi Implementasi Bubble Sort

3.4 Selection Sort

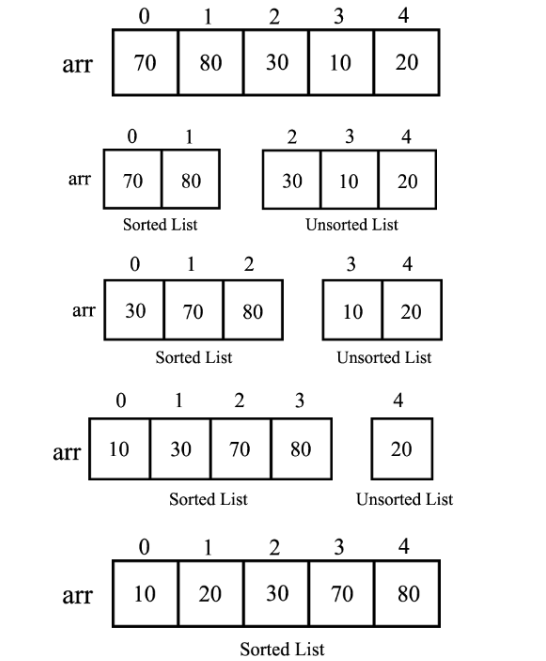
Selection Sort merupakan teknik pengurutan yang membandingkan elemen saat ini dengan elemen-elemen selanjutnya hingga mencapai elemen terakhir. Apabila ditemukan elemen yang lebih kecil daripada elemen saat ini, maka posisinya dicatat dan segera ditukar. Pendekatan Selection Sort melibatkan pemilihan nilai terkecil dari seluruh array, kemudian menukarnya dengan elemen di posisi awal, diikuti dengan perbandingan elemen saat ini terhadap elemen-elemen berikutnya hingga akhir array. Proses perbandingan ini dilanjutkan secara berulang hingga tidak terjadi lagi pertukaran antar elemen.

****

Gambar 3.4 Visualisasi Implementasi Selection Sort

3.5 Insertion Sort

Insertion Sort merupakan algoritma dasar yang cukup efektif untuk menyusun daftar yang sudah hampir rapi. Algoritma ini juga dapat dimanfaatkan sebagai elemen pendukung dalam algoritma yang lebih rumit (Traju, 2010:3). Mekanisme kerjanya melibatkan pengambilan elemen dari daftar secara bertahap dan penyisipan ke posisi yang sesuai, sesuai dengan julukannya. Dalam array, daftar yang sedang dibentuk dan elemen-elemen lainnya bisa berbagi ruang memori, meskipun prosedurnya terbilang agak kompleks. Demi mengoptimalkan penggunaan memori, implementasinya menerapkan pengurutan in-place, yakni membandingkan elemen saat ini dengan elemen-elemen sebelumnya yang telah tersusun, kemudian menggesernya secara berulang hingga mencapai posisi yang pas. Proses ini dilanjutkan secara kontinu hingga seluruh elemen input telah diproses (Traju, 2010:3). Metode Insertion Sort adalah proses penyusunan data yang dimulai dengan membandingkan dua elemen awal, diikuti perbandingan dengan elemen-elemen yang sudah tersusun, dan perbandingan tersebut diulang hingga tidak ada lagi elemen yang tersisa. Mekanisme Insertion Sort sesuai dengan namanya. Pada awalnya, dilakukan iterasi di mana setiap iterasi memindahkan nilai elemen, lalu menyisipkannya secara berulang hingga ke lokasi yang tepat. Proses ini diteruskan demikian. Melalui rangkaian iterasi tersebut, secara alami terbentuklah bagian yang telah tersortir dan bagian yang masih acak.Perhatikan gambar:

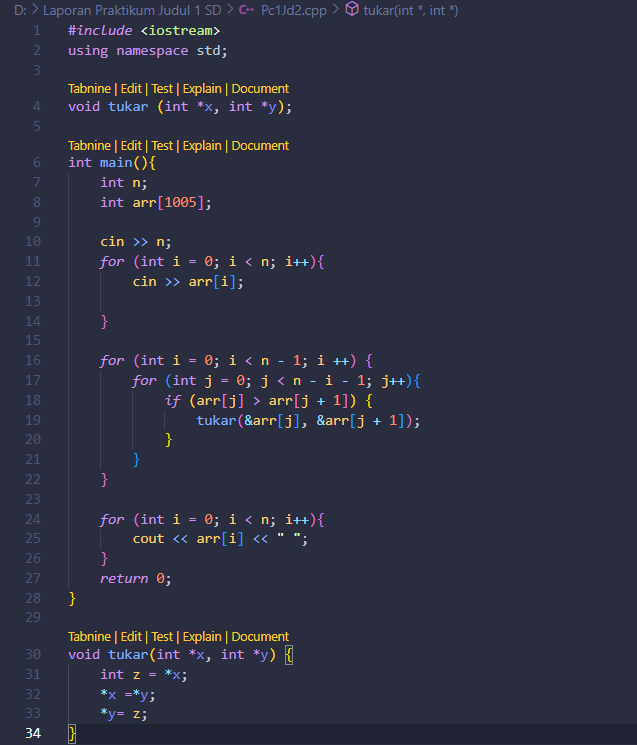


Gambar 3.5 Visualisasi Implementasi Insertion Sort

## IV. PROSEDUR PERCOBAAN

Adapun prosedur dari percobaan ini adalah sebagai berikut:

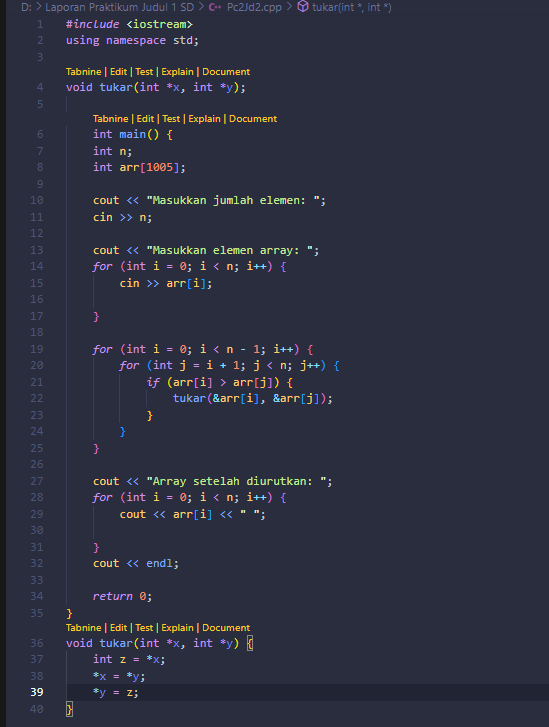
### 4.1 Percobaan 1: Bubble Sort

**

*Gambar 4.1. Percobaan 1: Bubble Sort*

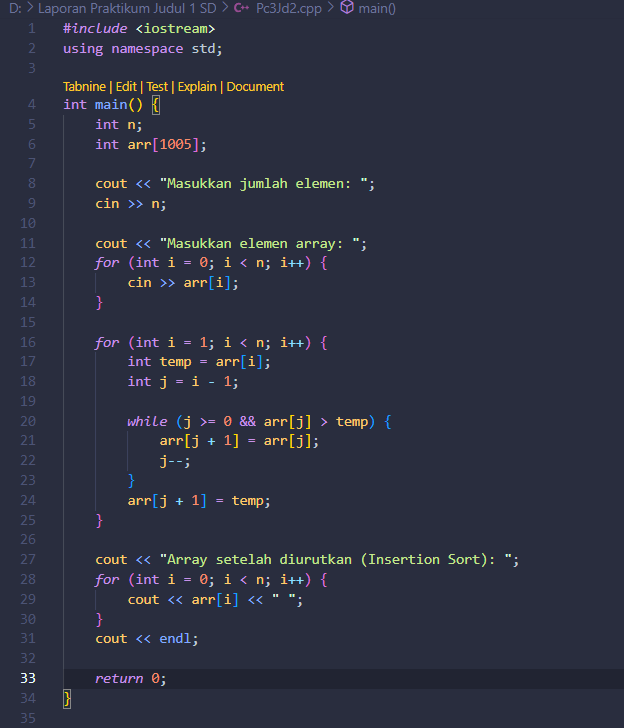
#### 

### 4.2 Percobaan 2 : Exchange Sort



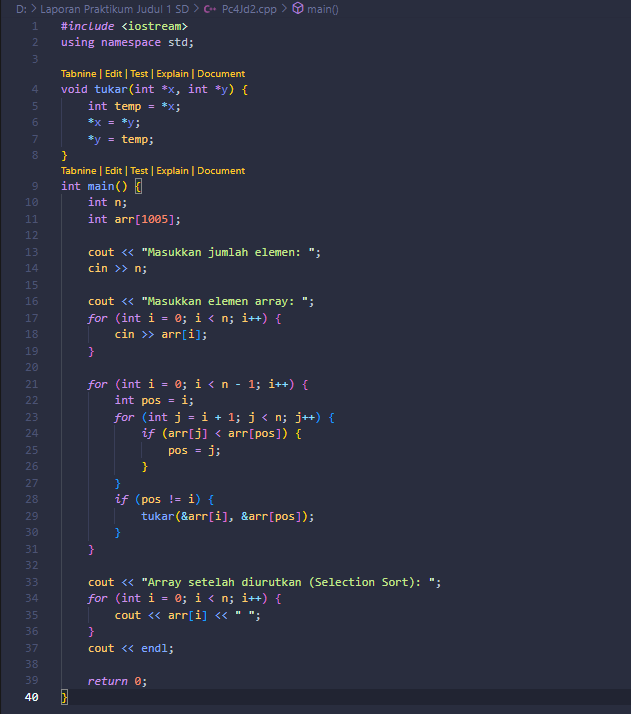
*Gambar 4.2 Percobaan 2: Exchange Sort*

### 4.3 Percobaan 3 : Insertion Sort



*Gambar 4.3 Percobaan 3 : Insertion Sort*

### 4.4 Percobaan 4 : Selection Sort



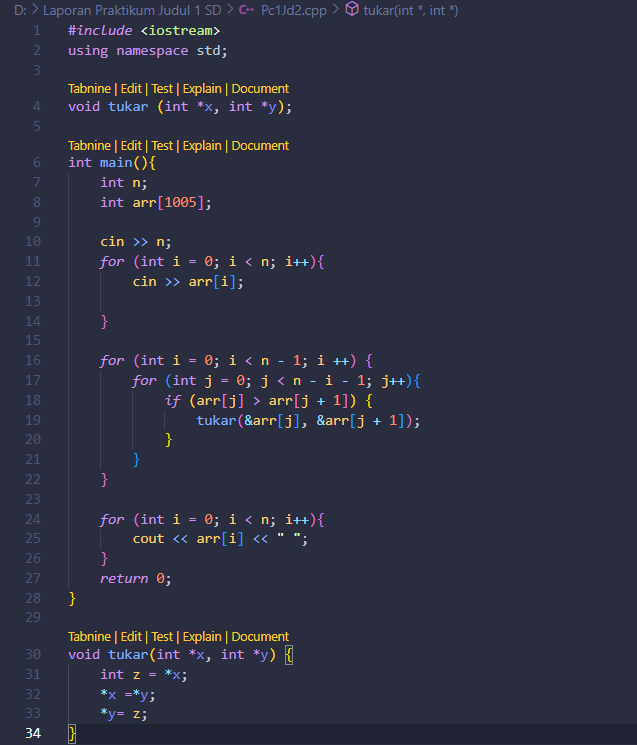
*Gambar 4.4 Percobaan 4 : Selection Sort*

## V. PEMBAHASAN

Adapun pembahasan dari percobaan ini adalah sebagai berikut:

### 5.1 Percobaan 1: Bubble Sort

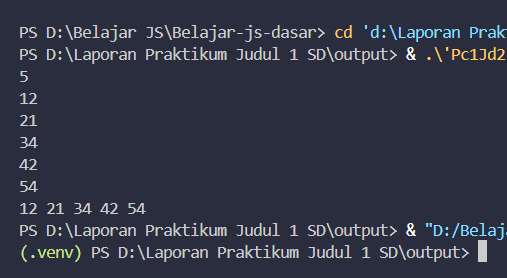
5.1.a Source Code Percobaan 1 : Bubble Sort

**

*Gambar 5.1..a Source Code Percobaan 1: Bubble Sort*

Berdasarkan gambar 5.1.a Source Code Percobaan 1: Bubble Sort. Program C++ ini mengimplementasikan algoritma Bubble Sort untuk mengurutkan daftar angka. Kode diawali dengan menyertakan header iostream untuk fungsionalitas input/output (cin dan cout), dan menggunakan namespace std untuk kemudahan penulisan. Fungsi kunci tukar dideklarasikan lebih awal, bertugas menukar nilai dua variabel melalui alamat mereka (menggunakan pointer \*x dan \*y). Eksekusi dimulai di dalam fungsi main(). Program mula-mula meminta pengguna memasukkan jumlah elemen (n) dan kemudian membaca sebanyak n angka ke dalam array arr melalui loop input. Bagian utama pengurutan adalah dua loop bersarang (nested loop). Loop luar mengontrol berapa kali pengulangan pengurutan harus dilakukan, dan loop dalam membandingkan setiap pasangan elemen yang bersebelahan. Jika suatu elemen (arr[j]) lebih besar dari elemen di depannya (arr[j + 1]), mereka dianggap tidak berurutan, dan fungsi tukar dipanggil. Pemanggilan ini mengirimkan alamat kedua elemen, memungkinkan fungsi tukar mengubah nilai asli mereka secara langsung. Proses ini secara berulang memindahkan elemen terbesar ke akhir array pada setiap iterasi. Setelah pengurutan selesai, loop terakhir mencetak semua elemen arr yang kini sudah terurut ke layar. Terakhir, fungsi tukar diimplementasikan menggunakan variabel sementara z untuk memastikan pertukaran nilai dua pointer (\*x dan \*y) berjalan dengan benar, menyelesaikan program, dan mengembalikan nilai 0.

##### 5.1.b Output Percobaan 1: Bubble Sort

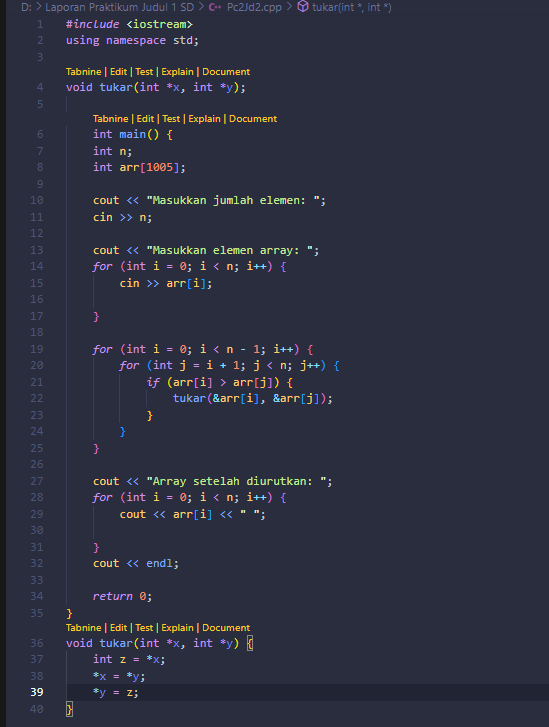


*Gambar* 5.1.b Output Percobaan 1: Bubble Sort

Berdasarkan gambar 5.1.b Output Percobaan 1: Bubble Sort. Output pada terminal ini merekam proses eksekusi program Bubble Sort yang dimulai dengan memasukkan data oleh pengguna. Awalnya, pengguna memberikan input 5, yang menentukan jumlah elemen yang akan diurutkan. Setelah itu, lima angka berikutnya, yaitu 12, 21, 34, 42, 54, dimasukkan satu per satu untuk mengisi array. Secara kebetulan, array input ini sudah terurut dalam urutan ascending. Program kemudian menjalankan algoritma Bubble Sort yang dirancang untuk mengurutkan dari terkecil ke terbesar. Karena kondisi perbandingan (arr[j] > arr[j + 1]) tidak pernah terpenuhi untuk array yang sudah terurut ini, tidak ada pertukaran yang dilakukan. Terakhir, program mencetak hasil akhir dari array tersebut. Output final berupa barisan angka 12 21 34 42 54 menunjukkan bahwa program telah berhasil memproses input dan mencetak array yang terurut (walaupun dalam kasus ini urutan tersebut tidak berubah dari input awal).

### 5.2 Percobaan 2 : Exchange Sort

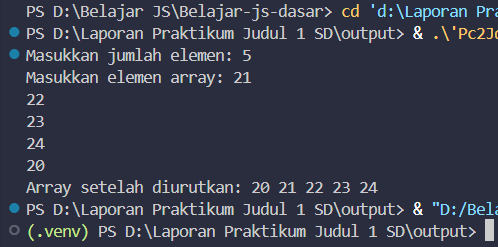
##### 5.2.a Source Code Percobaan 2 : Exchange Sort



*Gambar 5.2.a Source Code Percobaan 2 : Exchange Sort*

Berdasarkan gambar 5.2.a Source Code Percobaan 2 : Exchange Sort. Kode C++ ini berfungsi mengimplementasikan algoritma Exchange Sort untuk mengurutkan daftar angka. Program dibuka dengan mengimpor library iostream dan menggunakan namespace std untuk memudahkan penulisan syntax input dan output. Terdapat fungsi pembantu tukar yang dideklarasikan dan didefinisikan di bagian akhir. Fungsi tukar ini sangat penting karena ia menerima alamat (pointer) dua variabel (\*x dan \*y) dan menggunakan variabel sementara z untuk menukar nilai asli keduanya. Pelaksanaan program dimulai di fungsi main(). Program meminta pengguna untuk memasukkan jumlah elemen (n) dan dilanjutkan dengan meminta input untuk elemen-elemen array arr. Inti dari program terletak pada dua for loop bersarang (nested loop) yang menjalankan Exchange Sort. Outer loop (i) menunjuk pada posisi saat ini yang harus diisi dengan nilai terkecil. Sementara itu, inner loop (j) berfungsi untuk membandingkan elemen yang ditunjuk oleh i (arr[i]) dengan sisa elemen di depannya (arr[j]). Jika ditemukan bahwa elemen arr[i] lebih besar dari elemen yang sedang diperiksa (arr[j]), maka kedua elemen tersebut segera ditukar menggunakan fungsi tukar. Proses penukaran ini terjadi berulang kali dalam satu iterasi i setiap kali elemen yang lebih kecil ditemukan. Hasilnya, elemen terkecil akan terus "merambat" hingga menempati posisi i yang benar. Setelah seluruh array selesai diurutkan, program kemudian mencetak pesan "Array setelah diurutkan:" diikuti dengan mencetak semua elemen arr yang kini telah berurutan dari terkecil ke terbesar. Program diakhiri dengan mengembalikan nilai 0, menandakan eksekusi yang sukses.

5.2.b Output Percobaan 2-2 : Exchange Sort

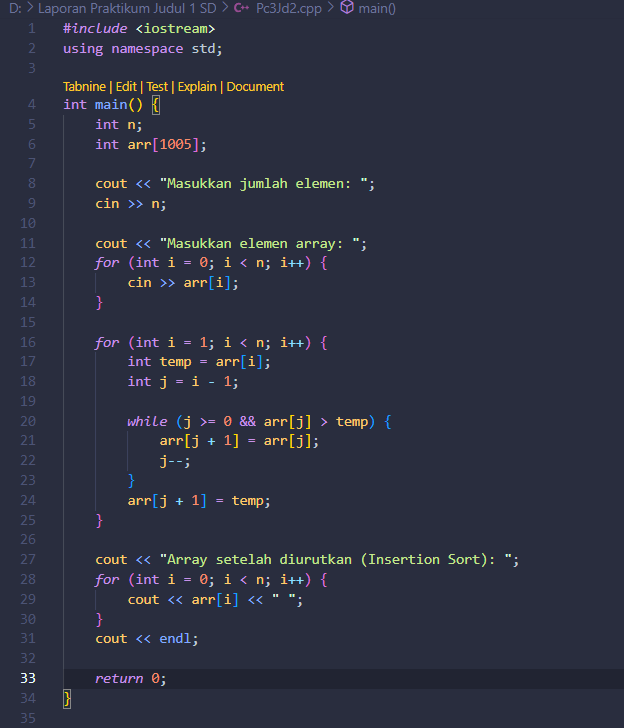


*Gambar 5.2.b Output Percobaan 2-2 : Exchange Sort*

Berdasarkan gambar 5.2.b Output Percobaan 2: Exchange Sort. Tampilan terminal ini merangkum proses input, pengurutan, dan output dari program Exchange Sort. Proses dimulai ketika pengguna menetapkan jumlah elemen (n) sebanyak 5. Selanjutnya, array diisi dengan urutan angka awal: 21, 22, 23, 24, 20. Program kemudian memproses array yang belum terurut ini menggunakan algoritma Exchange Sort. Algoritma tersebut secara berulang membandingkan dan segera menukar elemen, memastikan bahwa pada setiap langkah, elemen terkecil yang ditemukan ditarik ke depan array (ke posisi yang ditunjuk oleh outer loop). Meskipun prosesnya melibatkan beberapa pertukaran (misalnya, menukar 21 dengan 20, lalu 22 dengan 21, dan seterusnya), output akhir yang ditampilkan setelah proses pengurutan menunjukkan bahwa array telah berhasil disusun dalam urutan ascending (naik). Hasil cetak akhirnya adalah barisan angka 20 21 22 23 24, yang merupakan bukti keberhasilan program dalam mengurutkan data yang dimasukkan. Iterasi i=0 (Posisi 21): Program membandingkan 21 dengan semua elemen lain (22, 23, 24, 20). Ketika 21 dibandingkan dengan 20 (pada j=4), kondisi 21 > 20 terpenuhi. Keduanya ditukar. Array menjadi [20, 22, 23, 24, 21]. Setelah iterasi i=0 selesai, elemen terkecil (20) sudah berada di posisi awal. Iterasi i=1 (Posisi 22): Program membandingkan 22 dengan elemen sisa (23, 24, 21). Ketika 22 dibandingkan dengan 21 (pada j=4), kondisi 22 > 21 terpenuhi. Keduanya ditukar. Array menjadi [20, 21, 23, 24, 22]. Iterasi i=2 (Posisi 23): Program membandingkan 23 dengan elemen sisa (24, 22). Ketika 23 dibandingkan dengan 22 (pada j=4), kondisi 23 > 22 terpenuhi. Keduanya ditukar. Array menjadi [20, 21, 22, 24, 23]. Iterasi i=3 (Posisi 24): Program membandingkan 24 dengan elemen sisa (23). Kondisi 24 > 23 terpenuhi. Keduanya ditukar. Array menjadi [20, 21, 22, 23, 24]. Setelah semua iterasi selesai, program mencetak pesan Array setelah diurutkan: (Baris 27) dan dilanjutkan dengan mencetak semua elemen array yang kini sudah terurut. 20 21 22 23 24: Ini adalah output akhir yang menunjukkan bahwa algoritma Exchange Sort telah berhasil mengurutkan array dari nilai terkecil ke terbesar, sesuai dengan implementasi kode Anda.

### 5.3 Percobaan 3 : Insertion Sort

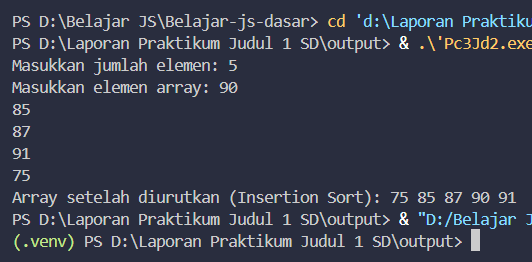
##### 5.3.a Source Code Percobaan 3 : Insertion Sort



*Gambar 5.3.a Source Code Percobaan 3 : Insertion Sort*

Berdasarkan gambar 5.3.a Source Code Percobaan 3 : Insertion Sort. Kode C++ ini mengimplementasikan algoritma Insertion Sort, sebuah metode pengurutan yang bekerja seperti menyusun kartu di tangan, dengan membangun array yang sudah terurut satu elemen pada satu waktu. Program dimulai dengan mempersiapkan library iostream dan namespace std. Fungsi utama, main(), mendeklarasikan variabel untuk jumlah elemen n dan array arr berkapasitas 1005. Program kemudian berinteraksi dengan pengguna, pertama meminta input nilai n, dan dilanjutkan dengan membaca n elemen array ke dalam arr melalui sebuah for loop input. Inti dari proses pengurutan dimulai pada for loop utama yang berinisiasi dari elemen kedua (i = 1), sebab elemen pertama dianggap sudah terurut. Di setiap iterasi, nilai elemen yang akan disisipkan (arr[i]) disimpan dalam variabel sementara temp. Kemudian, variabel j disiapkan untuk menunjuk ke elemen terakhir dari sub-array yang sudah terurut (i - 1). Proses penyisipan terjadi dalam while loop yang terus berjalan selama indeks j masih valid (j >= 0) dan elemen yang ditunjuk oleh j lebih besar dari temp (arr[j] > temp). Jika kedua kondisi ini benar, elemen di posisi j akan digeser satu posisi ke kanan (arr[j + 1] = arr[j]), dan j dimundurkan (j--) untuk melanjutkan perbandingan. Setelah while loop selesai (ketika posisi yang tepat ditemukan), nilai temp disisipkan ke posisi yang benar (arr[j + 1] = temp), menyelesaikan penyisipan dan memperluas sub-array yang terurut. Akhirnya, setelah seluruh array terurut, program mencetak pesan "Array setelah diurutkan (Insertion Sort):" diikuti dengan for loop yang mencetak semua elemen arr yang sudah terurut ke layar, dipisahkan oleh spasi. Program ditutup dengan mencetak baris baru (endl) dan mengembalikan nilai 0.

##### 5.3.b Output 5.3.b Percobaan : Insertion Sort

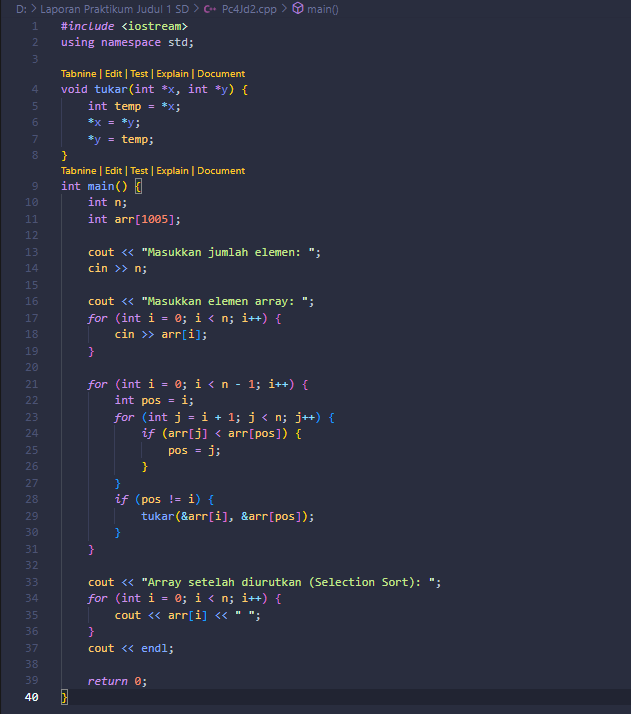


*Gambar 5.3.b Output Percobaan 3 : Insertion Sort*

Berdasarkan gambar 5.3.b Output Percobaan 3 : Insertion Sort. Tampilan hasil eksekusi program mencerminkan proses pengurutan menggunakan algoritma Insertion Sort. Program pertama-tama mencatat input dari pengguna, yaitu jumlah elemen sebanyak 5. Array awal yang dimasukkan adalah [90, 85, 87, 91, 75], yang merupakan data tidak terurut. Algoritma Insertion Sort kemudian dijalankan, mengambil setiap elemen dan menyisipkannya ke posisi yang tepat dalam sub-array di sebelah kiri yang sudah terurut, sambil menggeser elemen yang lebih besar ke kanan. Proses ini berulang, secara efektif memindahkan angka terkecil 75 dari posisi terakhir ke posisi pertama, dan menempatkan elemen lainnya sesuai urutan. Hasil akhirnya, yang dicetak setelah label "Array setelah diurutkan (Insertion Sort):", adalah barisan angka 75 85 87 90 91. Output ini memvalidasi bahwa program telah berhasil mengurutkan array input secara ascending (dari terkecil ke terbesar). oses pengurutan Insertion Sort dijalankan pada array input awal [90, 85, 87, 91, 75] berdasarkan nilai n=5. Insertion Sort bekerja dengan menganggap elemen pertama (90) sudah terurut, dan kemudian menyisipkan elemen-elemen sisanya satu per satu ke dalam sub-array yang terurut tersebut. Iterasi 1 (Menyisipkan 85) Pada iterasi pertama, nilai 85 (elemen pada indeks i=1) diambil sebagai temp. Angka 85 dibandingkan dengan elemen yang sudah terurut di kirinya (90). Karena 85 < 90, angka 90 digeser ke kanan. Kemudian, 85 disisipkan ke posisi kosong di depan. Array Setelah Iterasi 1: [85, 90, 87, 91, 75]. Iterasi 2 (Menyisipkan 87) Selanjutnya, nilai 87 (elemen pada indeks i=2) diambil. Angka 87 dibandingkan dengan sub-array terurut [85, 90]. Karena 87 < 90, angka 90 digeser ke kanan. 87 kemudian dibandingkan dengan 85. Karena 87 > 85, posisi ditemukan. 87 disisipkan di antara 85 dan 90. Array Setelah Iterasi 2: [85, 87, 90, 91, 75]. Iterasi 3 (Menyisipkan 91) Nilai 91 (elemen pada indeks i=3) diambil. Angka 91 dibandingkan dengan sub-array terurut [85, 87, 90]. Karena 91 lebih besar dari semua elemen yang sudah terurut, tidak ada pergeseran yang terjadi, dan 91 tetap berada di posisinya saat ini. Array Setelah Iterasi 3: [85, 87, 90, 91, 75]. Iterasi 4 (Menyisipkan 75) Terakhir, nilai 75 (elemen pada indeks i=4) diambil. Angka 75 adalah elemen terkecil. Ia dibandingkan dengan 91, 90, 87, dan 85. Semua elemen ini lebih besar dari 75, sehingga mereka digeser satu per satu ke kanan. Setelah pergeseran, 75 disisipkan ke posisi awal (indeks 0). Array Setelah Iterasi 4: [75, 85, 87, 90, 91] Setelah semua iterasi selesai, proses pengurutan berhasil menata array menjadi 75 85 87 90 91, sesuai dengan output program yang ditampilkan.

### 5.4 Percobaan 4 : Selection Sort

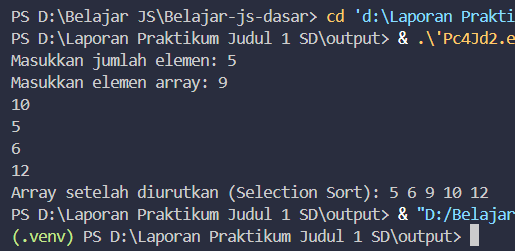
##### 5.4.a Source Code Percobaan 4 : Selection Sort



*Gambar 5.4.a Source Code Percobaan 4 : Selection Sort*

Berdasarkan gambar 5.4.a Source Code Percobaan 4 : Selection Sort. Kode program C++ ini mengimplementasikan algoritma Selection Sort, sebuah metode pengurutan yang bekerja dengan berulang kali menemukan elemen terkecil dari array yang belum terurut dan menukarnya ke posisi terdepan. Program diawali dengan mendefinisikan fungsi tukar, yang menggunakan variabel sementara temp untuk menukar nilai dua variabel melalui alamat (pointer) yang dikirimkan. Di dalam fungsi main(), program mendeklarasikan variabel n untuk jumlah elemen dan arr sebagai array data. Program berinteraksi dengan pengguna, meminta input n, dan diikuti dengan pembacaan n elemen untuk mengisi array. Inti dari pengurutan berada pada dua for loop bersarang. Outer loop (i) bertanggung jawab untuk iterasi melalui array dan menandai posisi terdepan yang harus diisi. Di awal setiap iterasi i, sebuah variabel pos diinisiasi dengan i; variabel ini akan menyimpan indeks dari elemen terkecil yang ditemukan. Inner loop (j) bertugas untuk menjelajahi sisa array (dari i+1 hingga n) untuk mencari elemen terkecil. Jika elemen yang sedang diperiksa (arr[j]) lebih kecil dari elemen terkecil saat ini (arr[pos]), maka pos diperbarui menjadi j. Setelah inner loop selesai, variabel pos akan menunjuk ke indeks elemen terkecil di sisa array. Kemudian, program menggunakan kondisi if (pos != i) untuk memeriksa apakah elemen terkecil benar-benar ditemukan di luar posisi saat ini. Jika iya, fungsi tukar dipanggil untuk menukar elemen di posisi i dengan elemen terkecil (arr[pos]). Dengan ini, elemen terkecil dipastikan berada di posisi terurutnya. Setelah semua iterasi selesai, program mencetak pesan "Array setelah diurutkan (Selection Sort):" dan menampilkan semua elemen arr yang kini sudah terurut. Program diakhiri dengan return 0.

##### 5.4.b Output Percobaan 4 : Selection Sort



*Gambar 5.4.b Output Percobaan 4 : Selection Sort*

Berdasarkan gambar 5.4.b Output Percobaan 4 : Selection Sort. Tampilan output ini merekam hasil dari eksekusi program Selection Sort. Program memulai dengan menerima 5 elemen array, yaitu [9, 10, 5, 6, 12]. Algoritma Selection Sort kemudian bekerja dengan mencari elemen terkecil di seluruh array yang belum terurut dan menukarnya ke posisi terdepan pada setiap iterasi. Pada Iterasi 1, program mengidentifikasi 5 sebagai nilai terkecil dan menukarnya dengan elemen di posisi pertama (9), menghasilkan array [5, 10, 9, 6, 12]. Kemudian, pada Iterasi 2, elemen terkecil berikutnya yang tersisa adalah 6, yang ditukar dengan elemen di posisi kedua (10), menghasilkan array [5, 6, 9, 10, 12]. Pada iterasi selanjutnya (Iterasi 3 dan 4), elemen terkecil sudah berada di posisi yang tepat, sehingga tidak ada pertukaran yang terjadi. Hasil akhir yang dicetak oleh program, yaitu 5 6 9 10 12, memvalidasi bahwa proses Selection Sort telah berhasil mengurutkan array input secara ascending.

## 

## VI. KESIMPULAN

Adapun kesimpulan dari percobaan ini adalah sebagai berikut:

1. Berdasarkan Pada Percobaan 5.1 Percobaan 1 : Bubble Sort. lgoritma ini berhasil mengurutkan data melalui proses berulang membandingkan dan menukar elemen-elemen yang bersebelahan. Efek dari proses ini adalah elemen dengan nilai terbesar secara bertahap berpindah ke posisi akhirnya di sisi kanan array pada setiap putaran iterasi.
2. Berdasarkan Pada Percobaan 5.2 Percobaan 2 : Exchange Sort Algoritma ini melaksanakan pengurutan dengan menetapkan satu elemen sebagai nilai pembanding (*pivot*) dan membandingkannya dengan seluruh elemen yang belum terurut. Pertukaran terjadi segera setelah ditemukan elemen yang lebih kecil, yang bertujuan memastikan posisi awal *array* ditempati oleh elemen terkecil setelah satu kali iterasi penuh.
3. Berdasarkan Pada Percobaan 5.3 Percobaan 3 : Insertion Sort, Algoritma ini bekerja secara bertahap dengan membangun *array* yang terurut. Setiap elemen yang belum terurut diambil satu per satu dan ditempatkan ke posisi yang benar di dalam *sub-array* yang sudah terurut. Penempatan ini dilakukan dengan cara menggeser elemen-elemen lain yang nilainya lebih besar.
4. Berdasarkan Pada Percobaan 5.4 Percobaan 4 : Selection Sort, Pengurutan dilakukan secara konsisten dengan mencari elemen bernilai terkecil dari seluruh bagian *array* yang belum terurut. Setelah elemen terkecil tersebut ditemukan, elemen tersebut ditukar ke posisi awalnya. Metode ini efisien karena hanya melakukan satu kali pertukaran per iterasi utama, sehingga meminimalkan jumlah total operasi pertukaran.

## 

## DAFTAR PUSTAKA

Hariyanto, Bambang. 2000. *Struktur Data.* Bandung

Setyantoro, D., & Hasibuan, R. A. (2020). Analisis dan perbandingan kompleksitas algoritma Exchange Sort dan Insertion Sort untuk pengurutan data menggunakan Python. *TEKINFO, 21*(1), 48-56.

Sjukani ,Moh. 2012. *Struktur Data (Algoritma dan Struktur Data dengan C,C++*. Jakarta : Mitra Wacana Media

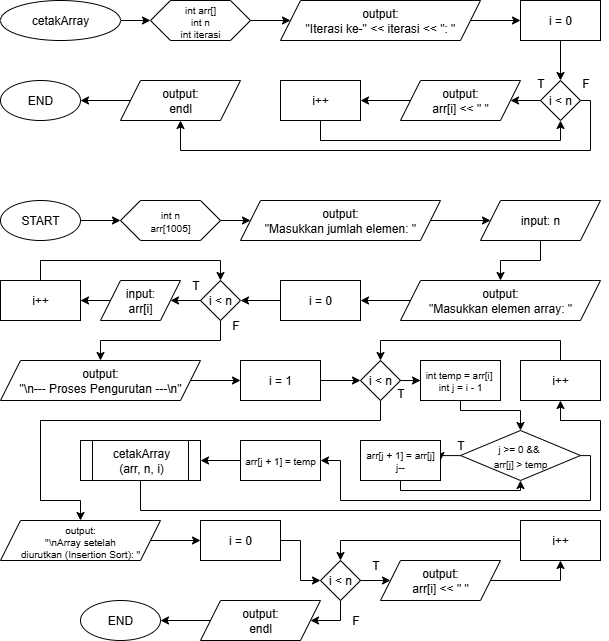
Rahayuningsih, P. A. (2016). Analisis perbandingan kompleksitas algoritma pengurutan nilai (sorting). *Jurnal Evolusi, 4*(2), 64.<https://evolusi.bsi.ac.id>

## TUGAS AKHIR

### 1. Kalimat Deskriptif

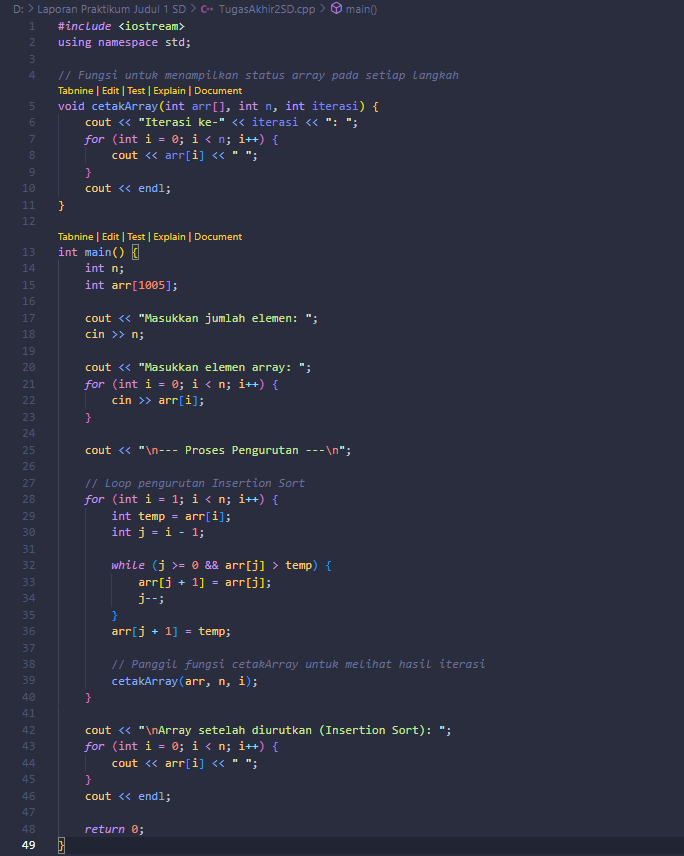
Program ini mengimplementasikan Insertion Sort dalam bahasa C++ untuk mengurutkan elemen-elemen dalam array secara ascending (menaik). Program dimulai dengan menerima input jumlah elemen array dari pengguna, diikuti dengan input setiap elemen array tersebut. Setelah menerima input, program menggunakan algoritma Insertion Sort untuk mengurutkan array. Dalam setiap iterasi, program mengambil elemen yang belum terurut dan menyisipkannya ke posisi yang tepat dalam sub-array di sebelah kiri yang sudah terurut. Elemen yang akan disisipkan disimpan dalam variabel sementara. Kemudian, elemen-elemen di sub-array terurut yang lebih besar akan digeser ke kanan untuk menciptakan ruang. Proses penggeseran dan penyisipan ini terus diulang hingga seluruh array terurut dari yang terkecil hingga terbesar. Program ini mengandalkan loop luar untuk mengiterasi elemen yang akan disisipkan dan loop dalam (while loop) untuk membandingkan dan menggeser elemen di sub-array terurut. Setelah proses pengurutan selesai, program menampilkan array yang sudah diurutkan sebagai output.

### 2. Flowchart Insertion Sort

**

*Gambar 2 Flowchart Insertion Sort Tugas Akhir*

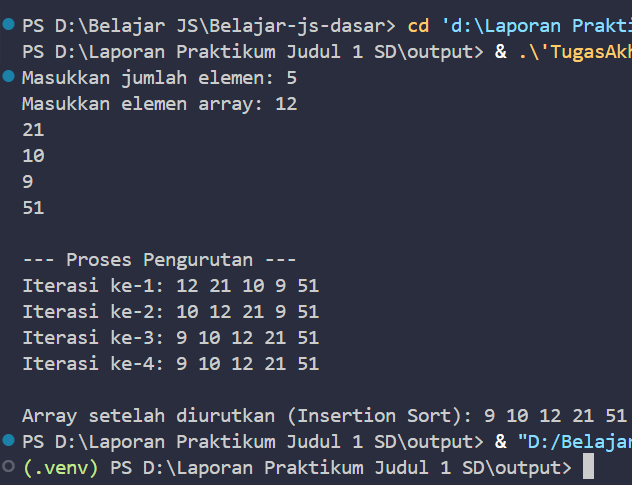
### 3. Source Insertion Sort



*Gambar 3 Source Code Insertion Sort Tugas Akhir*

Berdasarkan gambar 3 Source Code Insertion Sort Tugas Akhir. Kode program C++ ini mengimplementasikan algoritma Insertion Sort yang dirancang untuk mengurutkan elemen *array* secara *ascending*, namun dengan tambahan fitur visualisasi iterasi untuk keperluan analisis. Program diawali dengan pemuatan *library* iostream dan penggunaan namespace std. Terdapat fungsi pembantu bernama cetakArray yang didefinisikan di awal (Baris 5-11). Fungsi ini menerima *array*, ukuran elemen (n), dan nomor iterasi, lalu mencetak status *array* saat itu dengan *header* yang jelas. Fungsi ini bertujuan untuk mendokumentasikan status *array* setelah setiap langkah pengurutan. Eksekusi utama program dimulai di fungsi main(). Program mendeklarasikan variabel n (jumlah elemen) dan arr (array) dan berinteraksi dengan pengguna untuk menerima input nilai n serta elemen-elemen *array*. Inti dari pengurutan berada pada *loop* Insertion Sort (Baris 27-37). *Loop* utama (i) mengiterasi dari elemen kedua (i=1). Di setiap iterasi, nilai elemen yang akan disisipkan disimpan dalam variabel sementara temp, dan *while loop* (j) digunakan untuk membandingkan temp dengan elemen-elemen di *sub-array* terurut. Elemen-elemen yang lebih besar dari temp digeser ke kanan (arr[j + 1] = arr[j]) hingga posisi yang tepat untuk temp ditemukan. Setelah penyisipan (arr[j + 1] = temp;), fungsi cetakArray dipanggil (Baris 39) untuk menampilkan status *array* secara keseluruhan, memungkinkan pelacakan visual proses pengurutan. Akhirnya, setelah semua elemen selesai diurutkan, program mencetak pesan "Array setelah diurutkan (Insertion Sort):" dan menampilkan hasil akhir *array* yang sudah terurut, diikuti dengan return 0 untuk mengakhiri eksekusi program. Modifikasi ini menjadikan program sangat informatif untuk tujuan praktikum karena menyajikan proses pengurutan secara bertahap.

### 4. Output Insertion Sort



### *Gambar4 Output Insertion Sort Tugas Akhir*

Berdasarkan gambar 4 Output Insertion Sort Tampilan output ini merekam proses eksekusi program Insertion Sort secara menyeluruh, yang dilengkapi dengan fitur visualisasi langkah demi langkah. Program dimulai dengan menerima 5 elemen, membentuk array awal [12, 21, 10, 9, 51]. Setelah input diterima, program memasuki fase Proses Pengurutan. Dalam Iterasi ke-1, elemen 21 disisipkan dan, karena lebih besar dari 12, array tetap sama. Pada Iterasi ke-2, elemen 10 disisipkan, yang menyebabkan pergeseran 21 dan 12, menghasilkan urutan sementara [10, 12, 21, 9, 51]. Proses berlanjut pada Iterasi ke-3, di mana elemen 9 diidentifikasi sebagai yang terkecil saat itu, dan semua elemen yang lebih besar di depannya digeser, menempatkan 9 di posisi awal. Array menjadi [9, 10, 12, 21, 51]. Pada Iterasi ke-4 (elemen 51), karena array sudah terurut, tidak terjadi pergeseran dan array tetap. Proses iteratif ini secara visual menunjukkan bagaimana array diurutkan dari kiri ke kanan. Hasil akhirnya, yang ditampilkan sebagai "Array setelah diurutkan (Insertion Sort): 9 10 12 21 51", memvalidasi bahwa algoritma telah berhasil mengurutkan seluruh data secara ascending (menaik).

5. Link GitHub

<https://github.com/Hanzuel/Laporan-Praktikum-Struktur-Data-2.git>