Nama : Aini Azzah
NIM : 21091397006
Kelas : 2021 B

Selection Sort

(Metode pemilihan / seleksi**)**

Metode mengurutkan elemen dengan cara menyeleksi satu persatu angka pada elemen array, kemudian menentukan nilai minimum/maksimum pada array yang kemudian selanjutnya nilai maksimum/minimum tersebut ditukarkan dengan nilai elemen pertama dari array.

1. Maximum sort

Didasarkan pada pemilihan elemen maksimum. Konsepnya memilih elemen maksimum, kemudian mempertukarkan elemen maksimum tersebut dengan elemen paling akhir untuk urutan menaik (dari terkecil ke terbesar) atau ditukarkan dengan elemen paling awal untuk urutan menurun (dari terbesar ke terkecil). Setelah ditukarkan elemen awal/akhir tersebut di-isolasi (tidak digunakan) untuk tahapan selanjutnya. Kemudian dipilih kembali elemen yang bernilai maksimum tanpa menyertakan elemen yang telah terisolasi, dan ditukarkan dengan elemen paling awal /akhir. Proses tersebut diulang-ulang hingga terbentuknya urutan (sorting).

2. Minimum sort

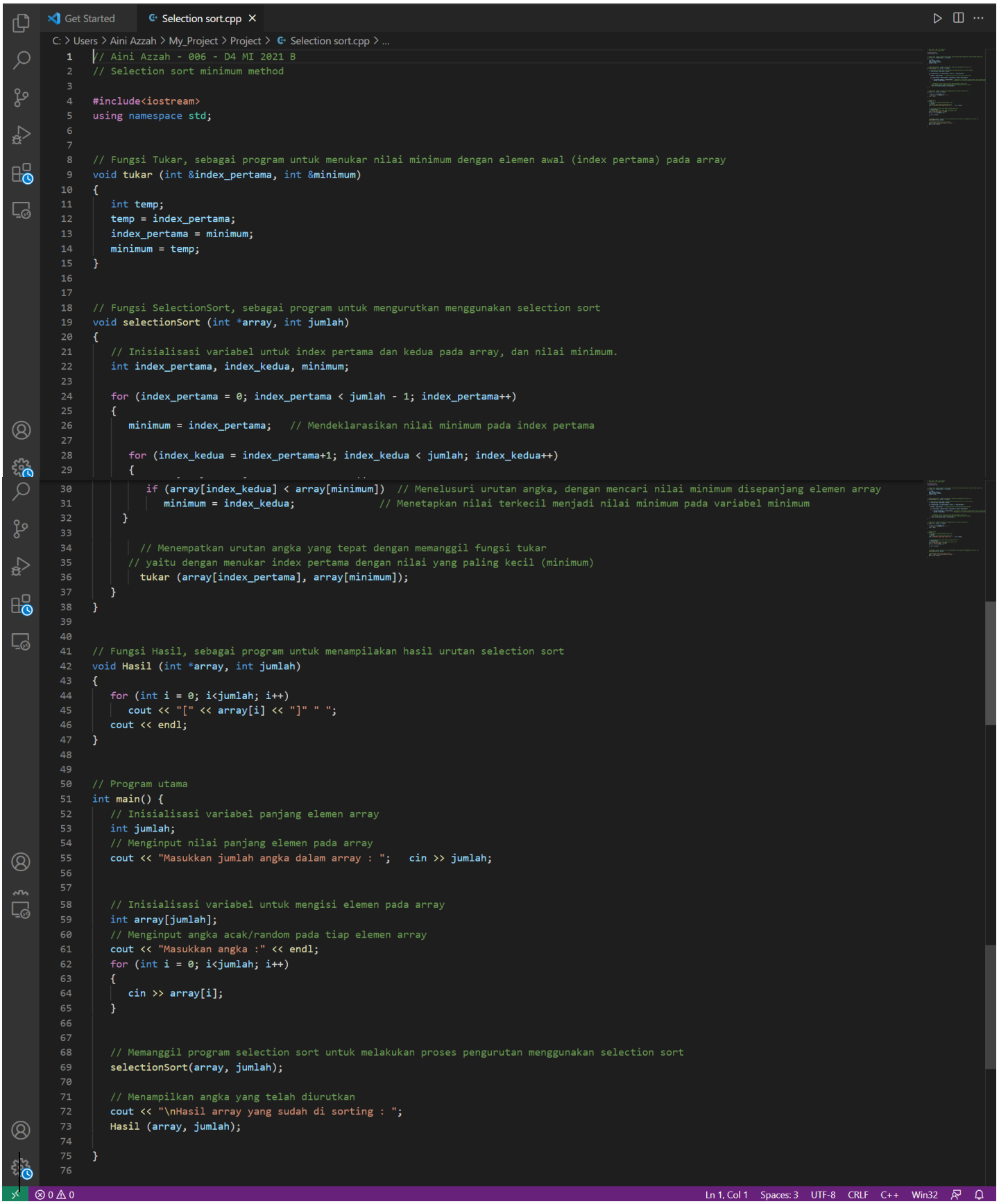
Dengan mendapatkan nilai elemen paling kecil, kemudian ditukarkan dengan elemen paling awal (untuk urutan menaik) atau ditukarkan dengan elemen paling akhir untuk urutan menurun (dari terbesar ke terkecil). Dan proses berulang seperti yang telah dijelaskan pada selection sort metode masksimum.

❖ Cara Kerja Selection Sort

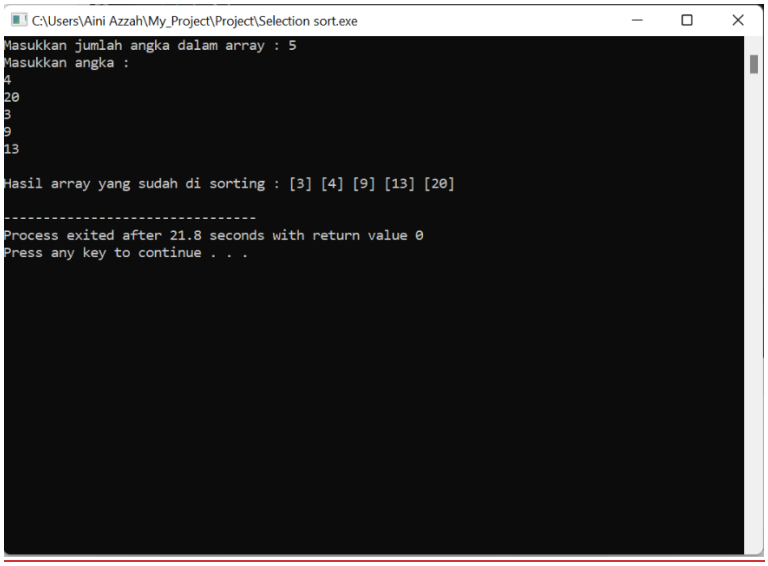
4	20	3	9	13
3*	20	4	9	13
3	4*	20	9	13
3	4	9*	20	13
3	4	9	13*	20

(*) Mewakili nilai minimum

❖ Berikut kode program yang saya gunakan untuk melakukan *sorting* (pengurutan) menggunakan Selection sort.



❖ Berikut hasil output dari kode Selection Sort di atas.



❖ Penjelasan Prinsip kerja Program Selection Sort di Atas

1. Perintah cout, untuk menginputkan jumlah angka/elemen yang akan diurutkan dalam array.
2. Perintah pada looping for, untuk menginputkan angka sebanyak jumlah elemen yang telah diinputkan.
3. Melakukan pengurutan menggunakan program selection sort yang telah dibuat pada fungsi SelectionSort.
    3.1. Looping untuk mecari nilai elemen terkecil (minimum) disepanjang elemen array, kemudian menukarnya dengan elemen pertama pada array menggunakan program *swap* (tukar) yang telah dibuat pada fungsi tukar.
    3.2. Looping untuk mengulangi proses tersebut hingga semua elemen terururt mulai dari nilai terkecil hingga terbesar (urutan menaik).
4. Menampilkan hasil array yang telah diurutkan dengan memanggil fungsi Hasil yang telah dibuat untuk menampilkan hasil pengurutan (*Sorting*).

❖ Analisis Kompleksitas / Big O Selection Sort

Sesuai dengan prinsip kerjanya, pengurutan menggunakan Selection Sort membutuhkan dua loop for bersarang satu sama lain utnuk menyelesaikan program pengurutannya. Satu for loop untuk menelusuri semua elemen dalam array hingga menemukan elemen minimum yang menggunakan for loop lain yang bersarang di dalam for loop. Oleh karena itu, mengingat ukuran ‘jumlah’ dari array inputan, algoritma pengurutan memiliki kompleksitas ruang dan waktu sebagai berikut.

1. Kompleksitas Waktu (*Time Complecity*)

    Jumlah perbandingan

Elemen yang belum diurutkan	Perbandingan untuk menemukan nilai min
n	n – 1
n – 1	n – 2

-	-
3	2
2	1
1	0
	n (n – 1) / 2

Running time untuk input n

$$\sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-1} (n-1-(i+1)+1)$$

$$= \sum_{i=0}^{n-1} (n-i+1)$$

$$= n \sum_{i=0}^{n-1} 1 - \sum_{i=0}^{n-1} i$$

$$= n^2 - n - \frac{(n-1)n}{2}$$

$$= O(n^2)$$

n = 1	
n = 5	
n = 10	

Berdasarkan operasi perbandingan elemennya :

(n – 1) + (n – 2) + (n – 3) + …….. + 1 = n (n – 1) / 2

Sehingga mendekati sama dengan n^2.
Dan juga, Selection sort menggunakan dua loop dalam programnya, sehingga sederhananya n * n = n^2

Terbaik	$O(n^{2)}$
Terburuk	$O(n^{2)}$
Rata-rata	$O(n^2)$

- Kompleksitas kasus terbaik = $O(n^2)$
  Terjadi ketika array sudah di urutkan.
- Kompleksitas kasus terburuk = $O(n^2)$
  Kasus terburuk dapat terjadi, jika ingin mengurutkan dalam urutan menaik (dari kecil ke besar) akan tetapi array yang diinputkan dalam urutan menurun (dari besar ke kecil)
- Kompleksitas kasus rata-rata = $O(n^2)$
  Kompleksitas kasus rata-rata terjadi jika, ketika elemen-elemen array yang telah diinputkan berada dalam urutan acak (tidak naik atau turun).

Kompleksitas waktu dari jenis selection sort sama dalam semua kasus (terbaik, terburuk, dan rata-rata). Pada setiap langkah harus menemukan elemen minimum dan meletakkannya di

tempat yang tepat. Dan elemen minimum tidak akan diketahui sampai akhir dari elemen array telah ditelusuri.

2. Kompleksitas Ruang (*Space Complexity*)
   Kompleksitas ruang dari selection sort adalah O(1) karena menggunakan *extra variable* (variable tambahan) yaitu “temp” untuk membantu menjalankan program pengurutannya.

Dengan mengetahui komplelksitas baik waktu dan ruang dari selection sort, dapat diambil kesimpulan mengenai kelebihan dan kekurangan dari metode pengurutan menggunakan selection sort ini, jika dibandingkan dengan metode pengurutan (*sorting*) yang lainnya (Insertion sort, Bubble sort, Merge sort, dan shell sort) yang telah di jelaskan rekan kelompok saya.

Kelebihan :

- Implementasi sederhana dan efisien untuk kumpulan data kecil.
- Lebih efisien jika dibandingkan dengan pengurutan kuadrat lainnya ($O(n^2)$), yaitu Bubble sort atau Insertion sort
- Efisien untuk kumpulan data yang sudah diurutkan. Kompleksitas waktunnya yaitu ($O(n^2)$), karena ada dua perulangan (*looping*) bersarang.
- Tidak mengubah urutan relative elemen yang sudah diurutkan. Kompleksitas ruangnya yaitu ($O(1)$).

Kekurangan :

- Selection sort jauh kurang efisien pada daftar yang besar, jika dibandingkan dengan algoritma quick sort, atau merge sort. Karena akan menyebabkan kompleksitas yang lebih tinggi dan kurang praktis.

Jadi semakin banyak elemen yang diinputkan pada array, maka semakin banyak waktu yang dibutuhkan untuk menyelesaikan program pengurutan menggunakan Selection Sort,