**LAPORAN PRAKTIKUM**

**STRUKTUR DATA DAN ALGORITMA**

**ANALISIS ASIMTOTIK**



**DISUSUN OLEH**

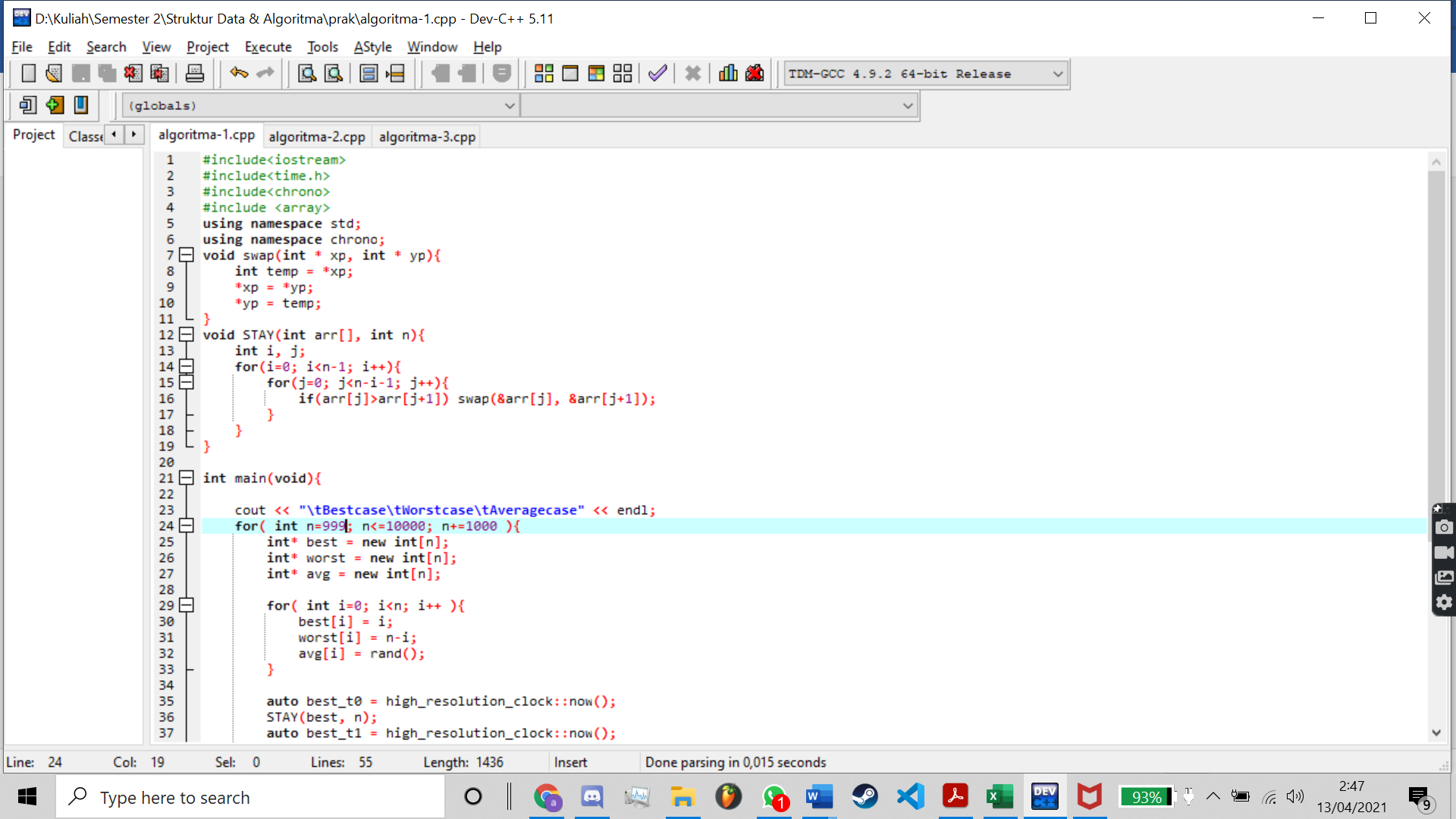
**Muhammad Alwiza Ansyar M0520051**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM UNIVERSITAS SEBELAS MARET**

**2021**

1. **Algoritma-1.cpp**
2. Source code:



1. Analisa Big O

Terdapat dua fungsi yaitu fungsi **swap** dan fungsi **STAY**. Fungsi **swap** melakukan pertukaran value antara dua variable, sedangkan fungsi **STAY** melakukan pengurutan elemen array dari kecil ke besar dengan cara memanggil **swap** apabila ditemukan elemen yang tidak urut (dengan cara membandingkan nilai dari elemen)

Fungsi **swap**:

* Memiliki tige baris (8-10) yang masing-masing merupakan konstan sehingga memiliki O(1)

Fungsi **STAY**:

* Baris 13 merupakan konstan sehingga memiliki O(1)
* Baris 14 merupakan loop menggunakan for statement dengan iterasi i++ sehingga memiliki O(n)
* Baris 15 merupakan loop menggunakan for statement yang juga merupakan inner loop dari loop sebelumnya, menggunakan iterasi j++ sehingga memiliki O(n)
* Baris 16 berisi pertukaran yang eksekusinya tergantung dari isi array

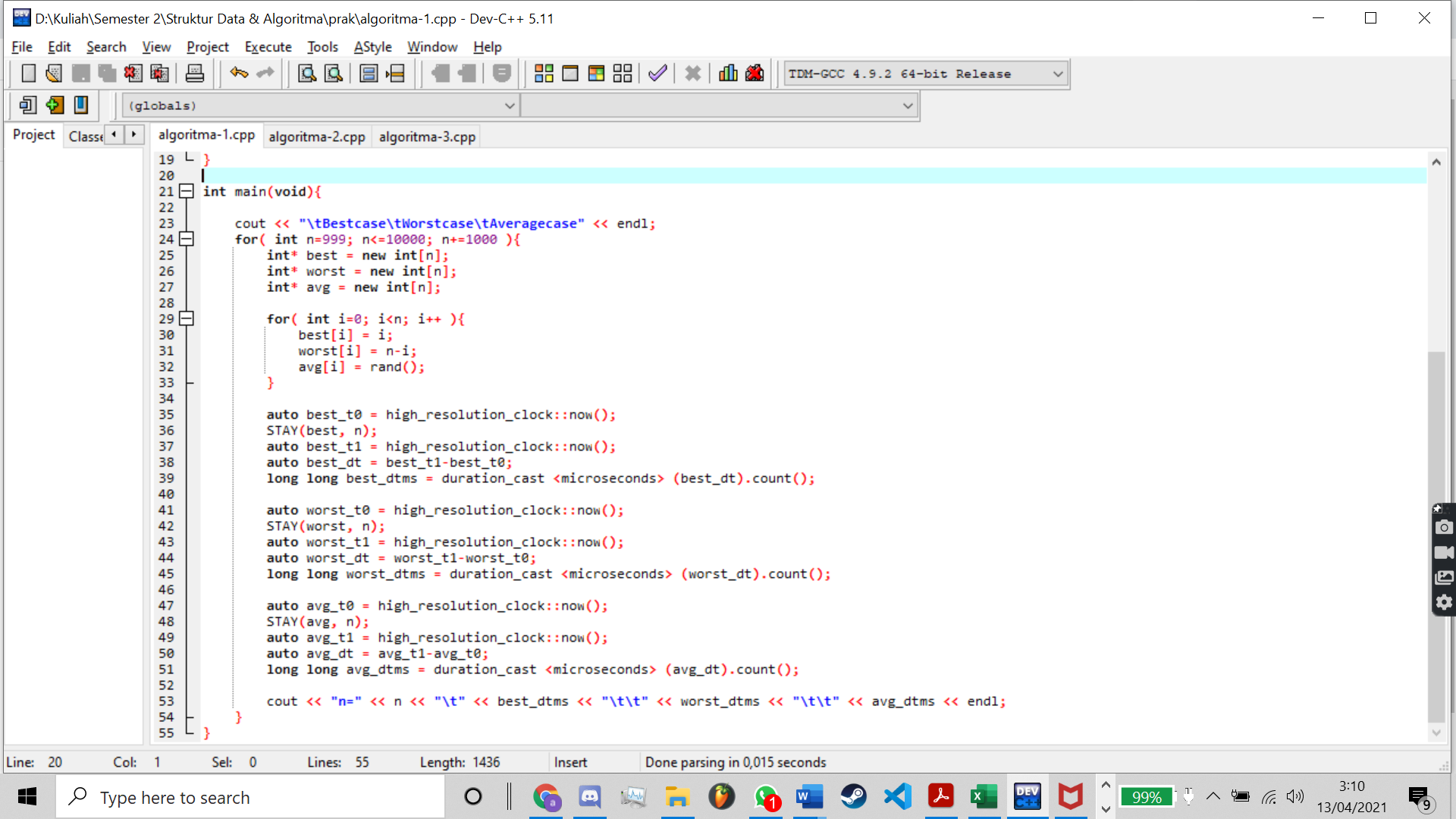
1. Analisa kasus

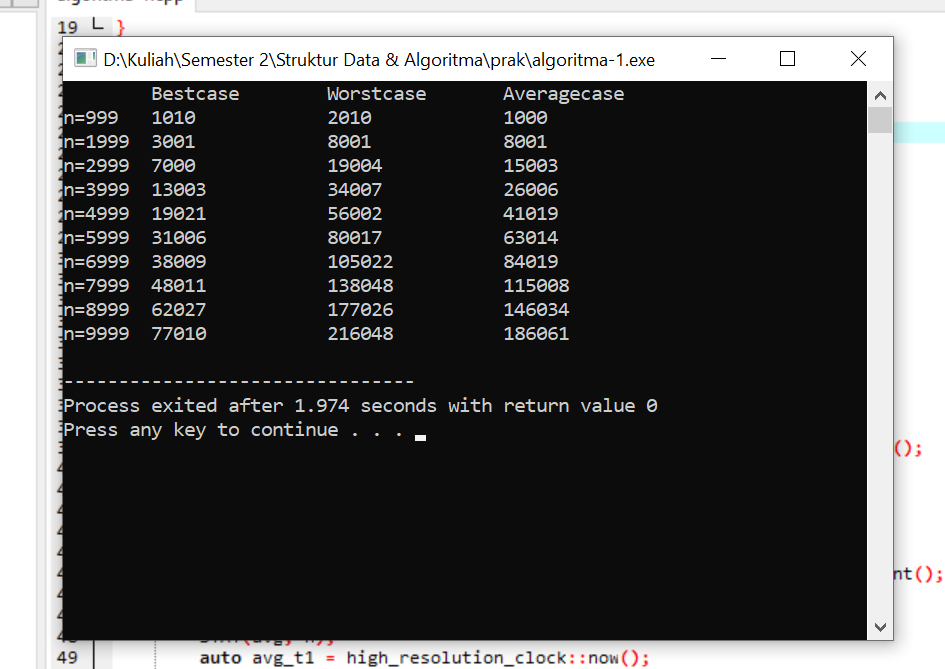
Isi dari array akan mempengaruhi kompleksitas waktu dari program. Jika ditemukan elemen [j] lebih besar dari elemen [j+1], maka akan dipanggil fungsi **swap**. Namun, fungsi **swap** hanya memiliki O(1) sehingga perbedaan antar kasus tidak signifikan

* Bestcase 🡪 Array dengan elemen urut dari kecil ke besar, tidak dilakukan **swap** sama sekali 🡪 O(n2)
* Worstcase 🡪 Array dengan elemen urut dari besar ke kecil, dilakukan **swap** sebanyak n-1 kali 🡪 O(n2)
* Averagecase 🡪 Array dengan elemen yang acak, **swap** mengikuti elemen 🡪 O(n2)

1. Testing

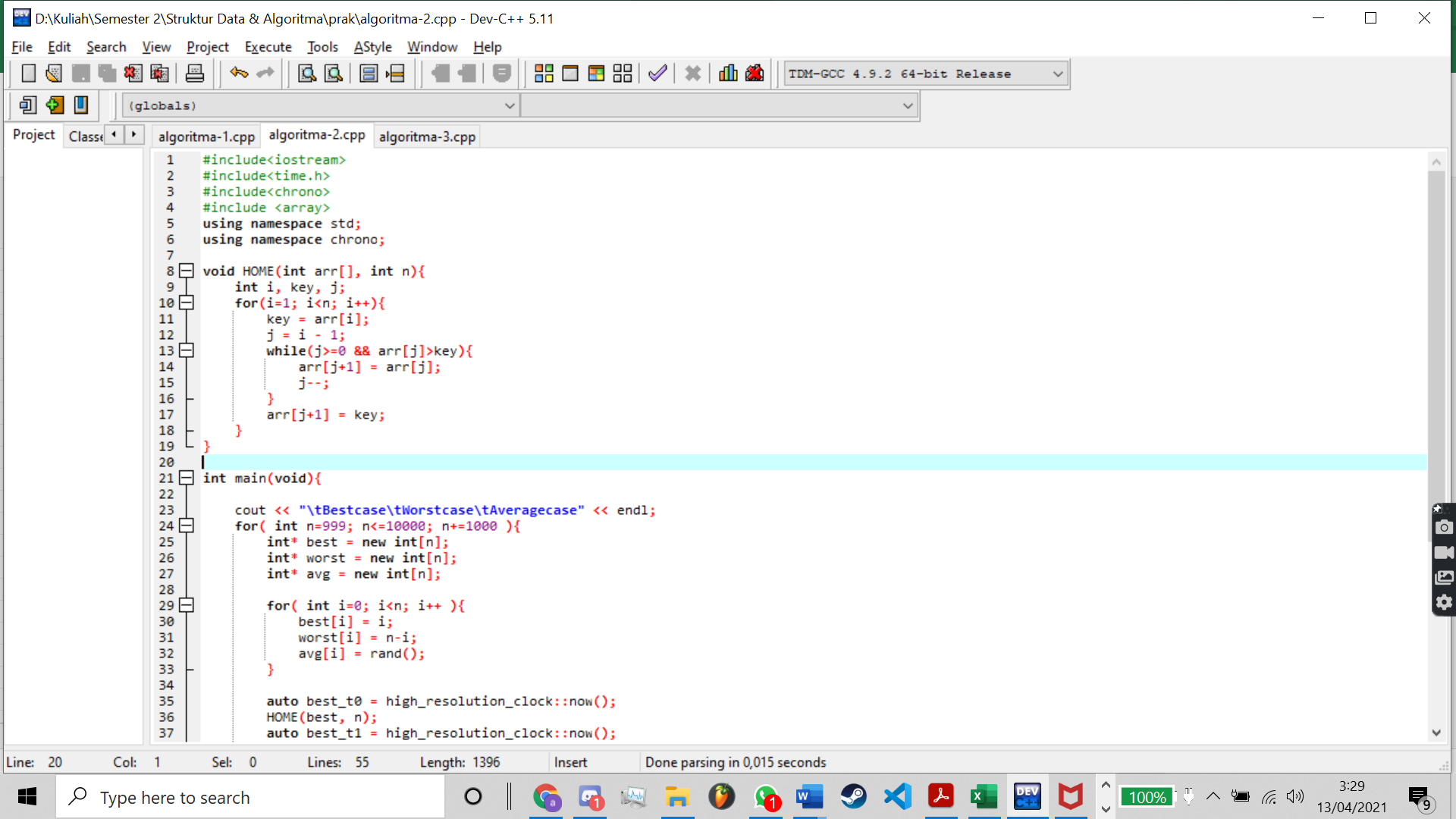
Pada test ini digunakan n dengan initial value 999, batas n<=10000, dan iterasi n=n+1000





1. Grafik

1. **Algoritma-2.cpp**
2. Source code:



1. Analisa Big O

Terdapat fungsi **HOME** yang melakukan pengurutan array dari kecil ke besar. Apabila ditemukan elemen yang tidak urut, maka akan masuk pada while statement yang melakukan pertukaran elemen array

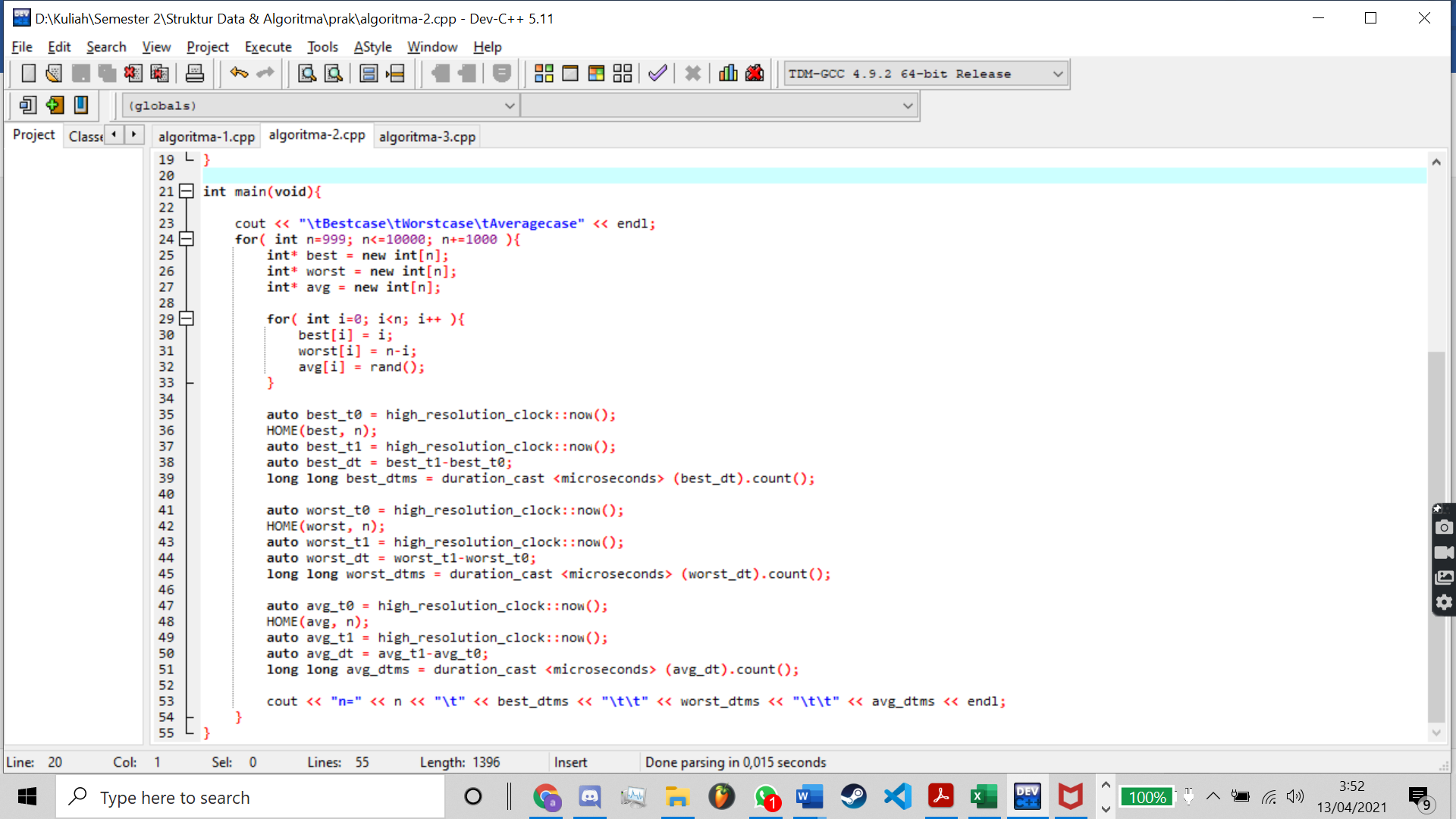
* Baris 9, 11, 12, 14, 15, dan 17 merupakan konstan sehingga memiliki O(1)
* Baris 10 memiliki loop menggunakan for statement dan iterasi i++ sehingga memiliki O(n)
* Baris 13 memiliki loop menggunakan while statement dan iterasi j— sehingga memiliki O(n)

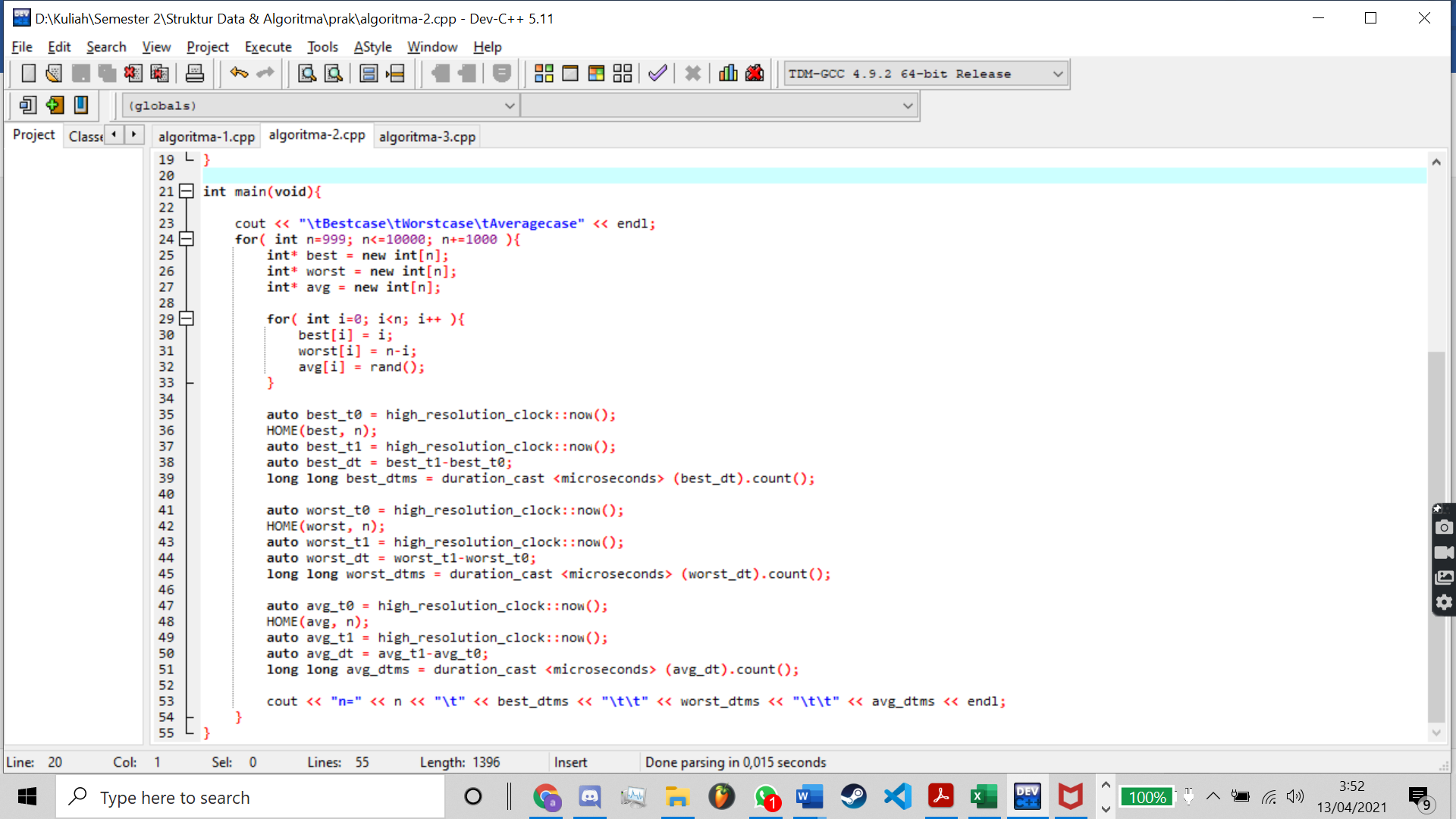
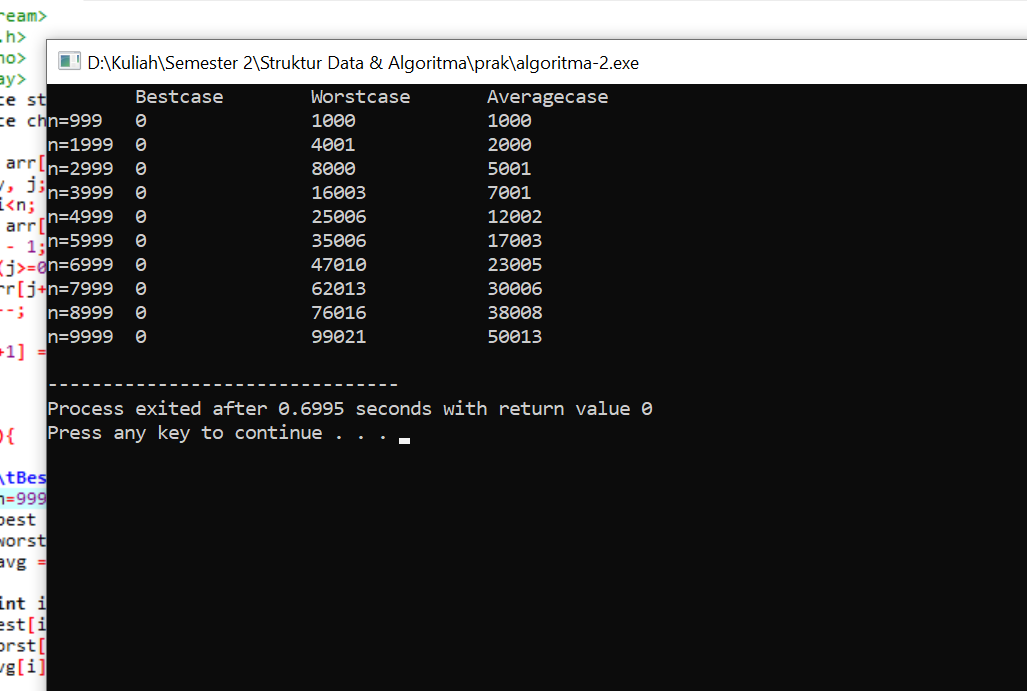
1. Analisa kasus

Isi dari array akan mempengaruhi kompleksitas waktu. Jika ditemukan elemen [j] lebih besar dari [j+1] maka akan dieksekusi while loop yang melakukan pertukaran. While loop memiliki O(n) sehingga akan ada perbedaan signifikan pada kasus

* Bestcase 🡪 Array dengan elemen urut dari kecil ke besar, tidak dieksekusi while loop sama sekali 🡪 O(n)
* Worstcase 🡪 Array dengan elemen urut dari besar ke kecil, dieksekusi while loop sebanyak n-1 kali 🡪 O(n2)
* Averagecase 🡪 Array dengan elemen yang acak, while loopmengikuti elemen 🡪 O(n2)

1. Testing

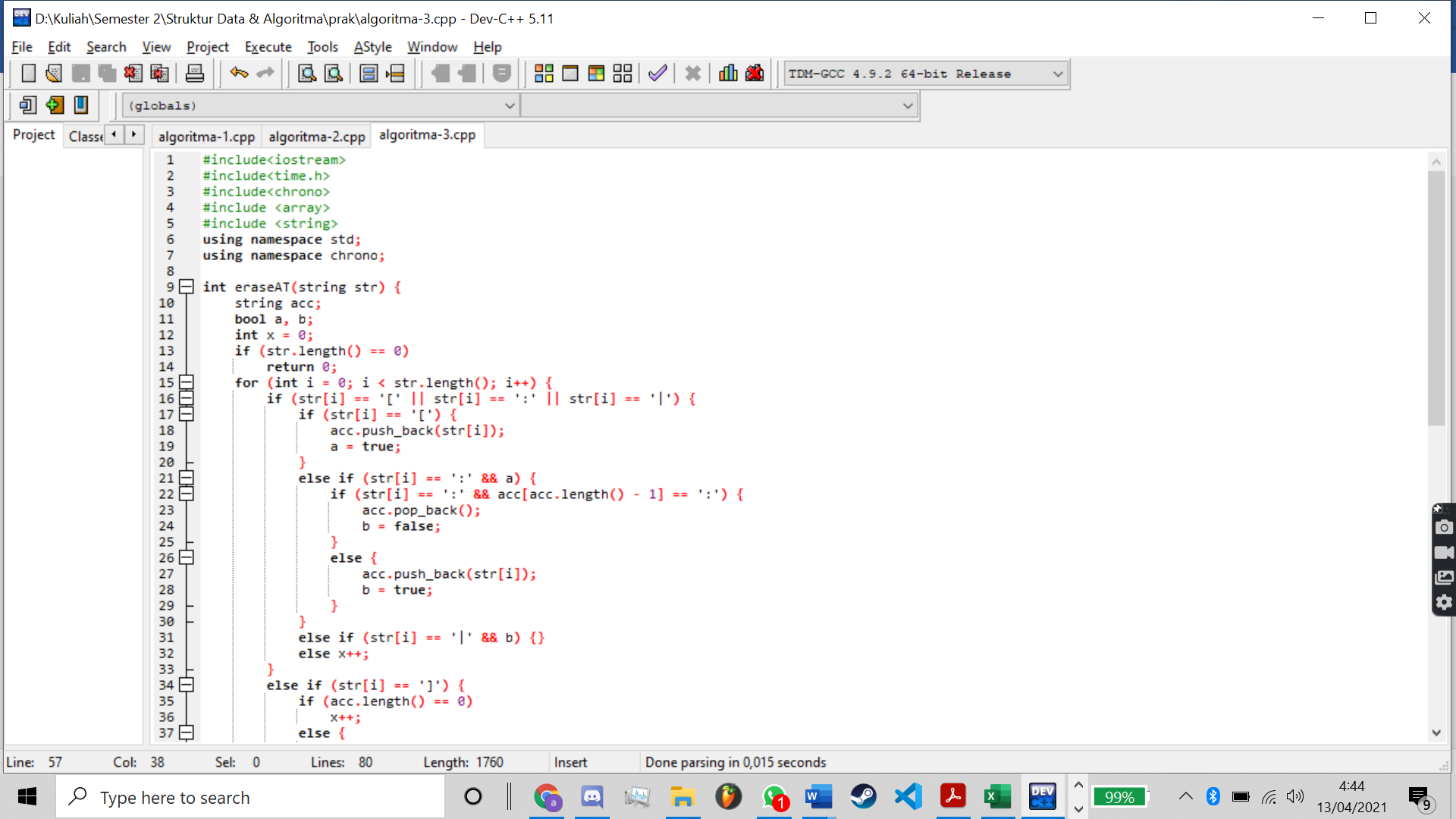
Pada test ini digunakan n dengan initial value 999, batas n<=10000, dan iterasi n=n+1000

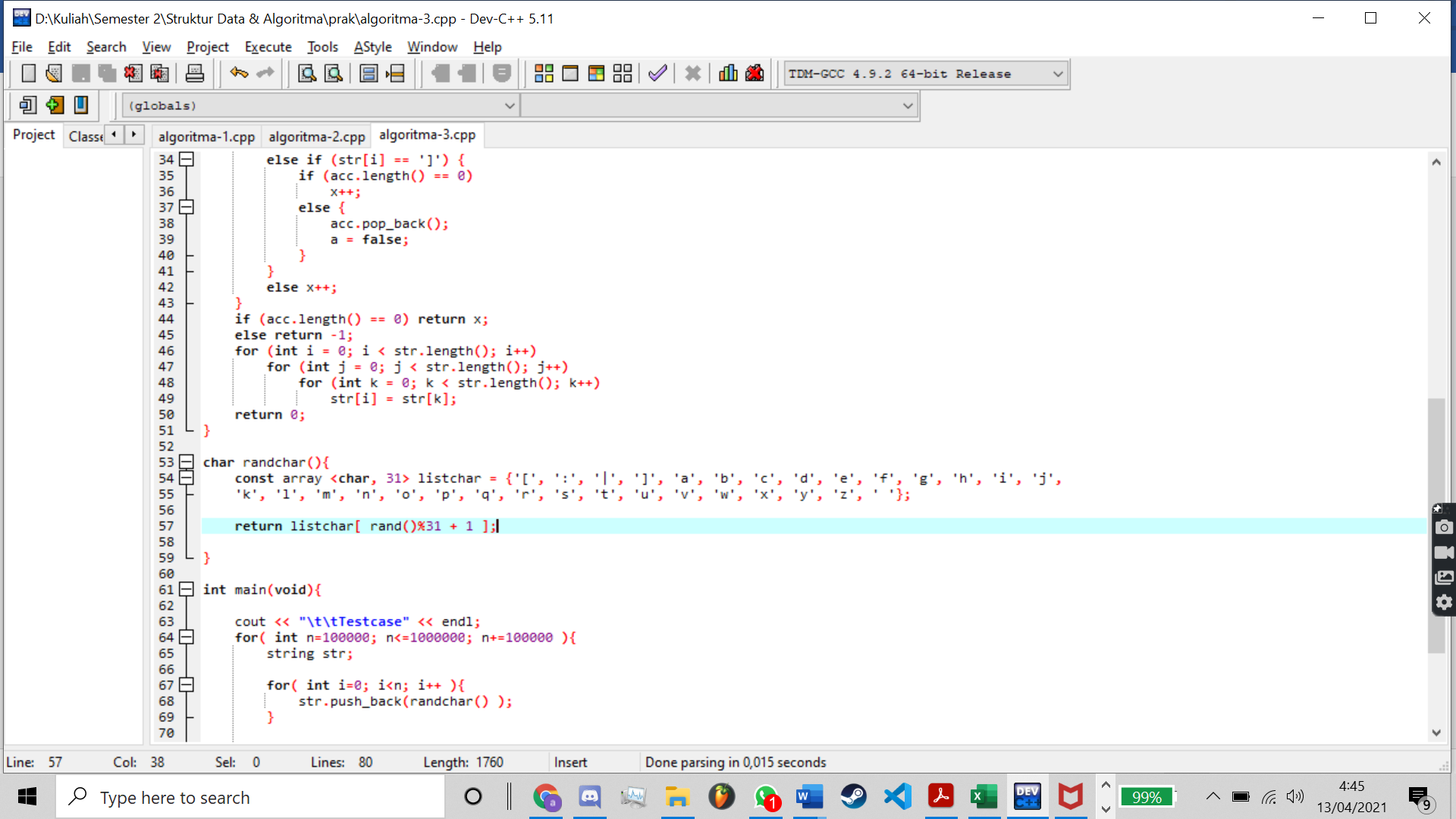


Note: runtime dari bestcase bernilai sangatlah kecil yang mana masih dianggap 0 dalam skala mikrosecond

1. Grafik

1. **Algoritma-3.cpp**
2. Source code:





1. Analisa Big O

Terdapat fungsi **eraseAT** yang menghitung jumlah karakter dari sebuah string dengan pengecualian pada empat karakter yaitu “[“, “:”, “|”, dan “]” yang memiliki mekanisme khusus. Mekanisme tersebut didukung oleh variabel pendukung a dan b yang bertipe boolean.

* Baris 15 memiliki loop menggunakan for statement dan iterasi i++ sehingga memiliki O(n)
* Selain loop, semua merupakan konstan sehingga memiliki O(1)

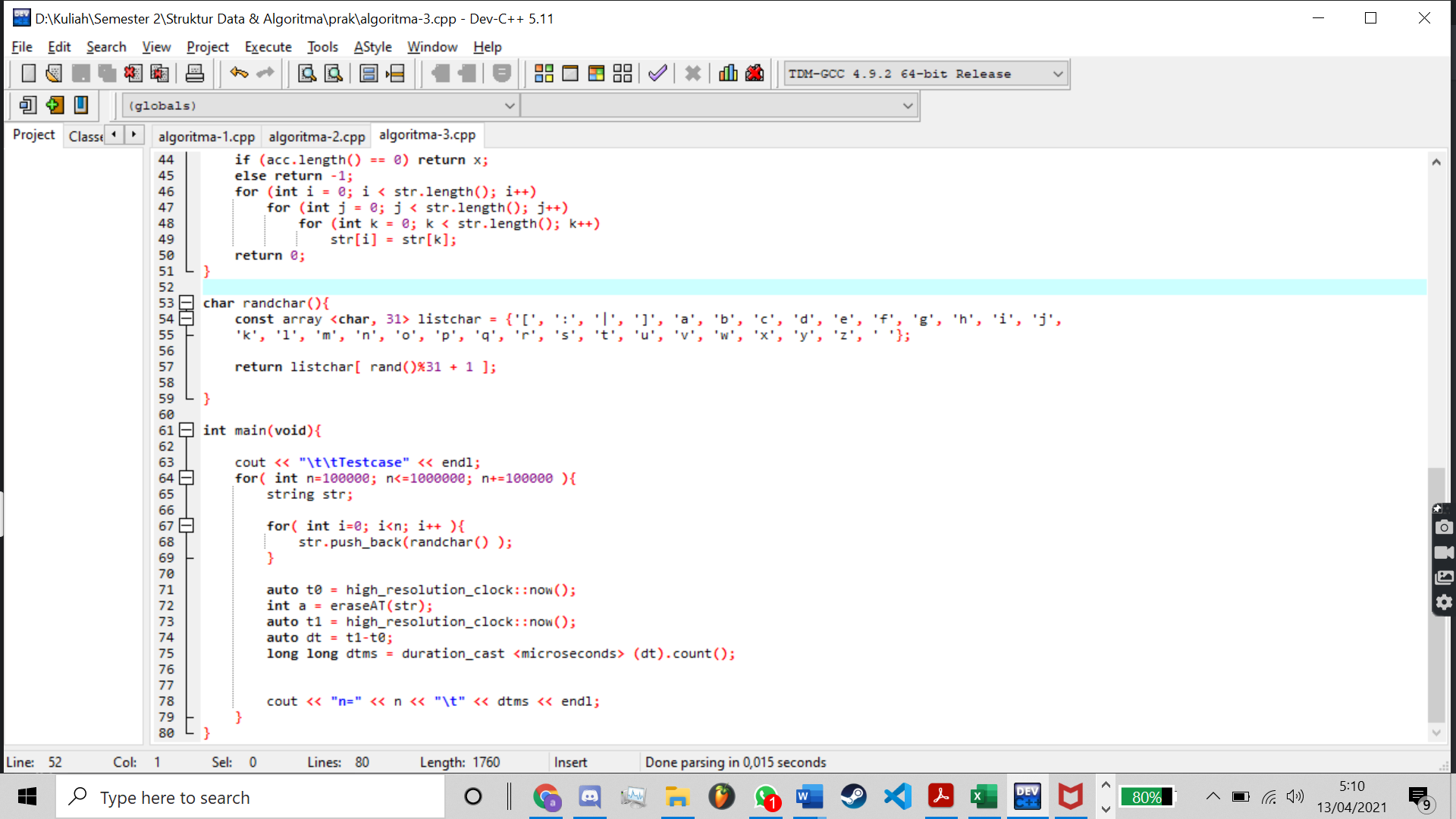
Terdapat perintah yang *unreachable* yaitu pada baris 46-50 dikarenakan baris tersebut terletak sesudah if else statement yang keduanya melakukan return.

1. Analisa kasus

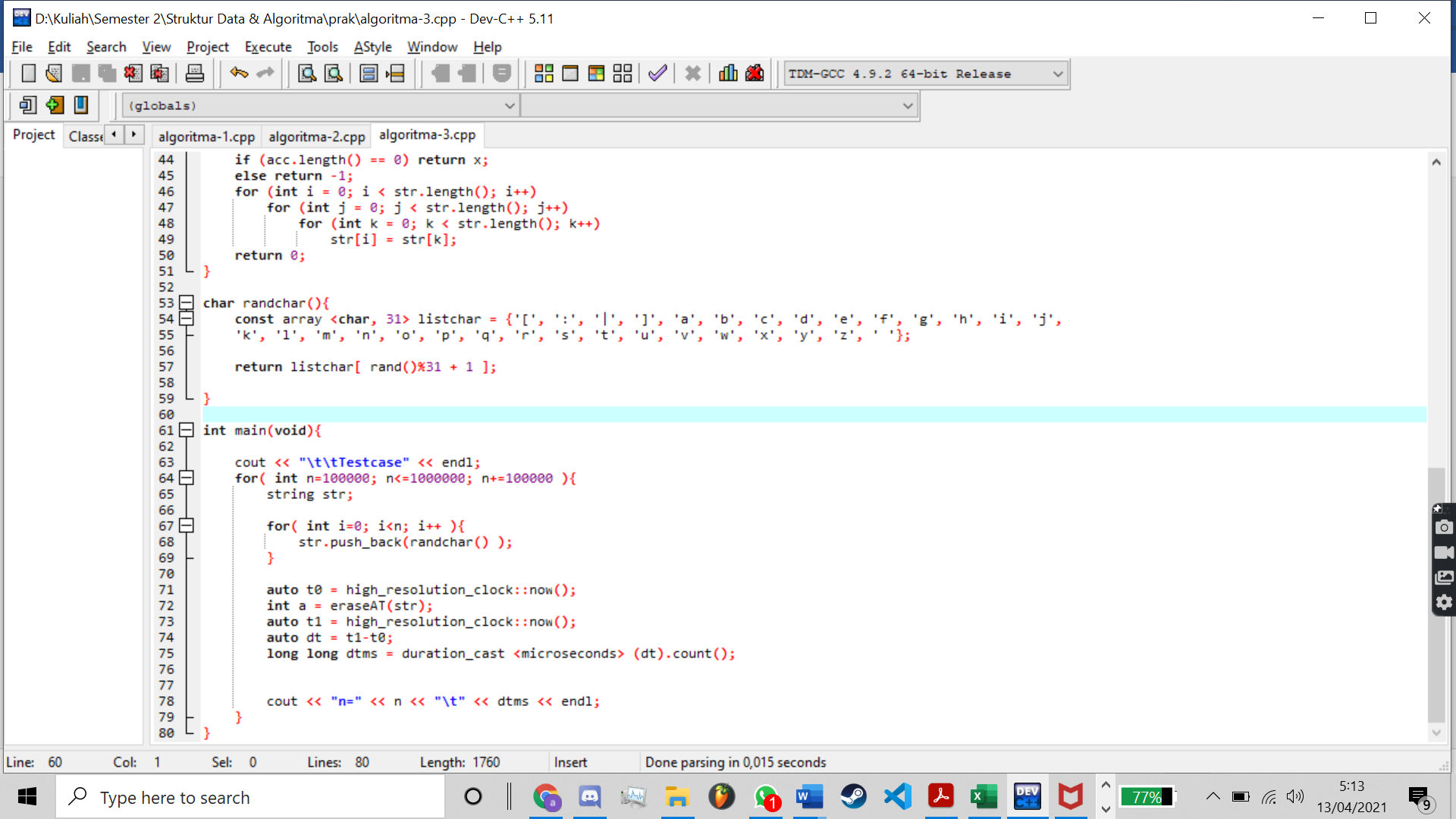
Fungsi **eraseAT** memiliki perbedaan T(n) yang sangat kecil sehingga fungsi **eraseAT** dapat dianggap tidak memiliki kasus bestcase, worstcase, dan averagecase

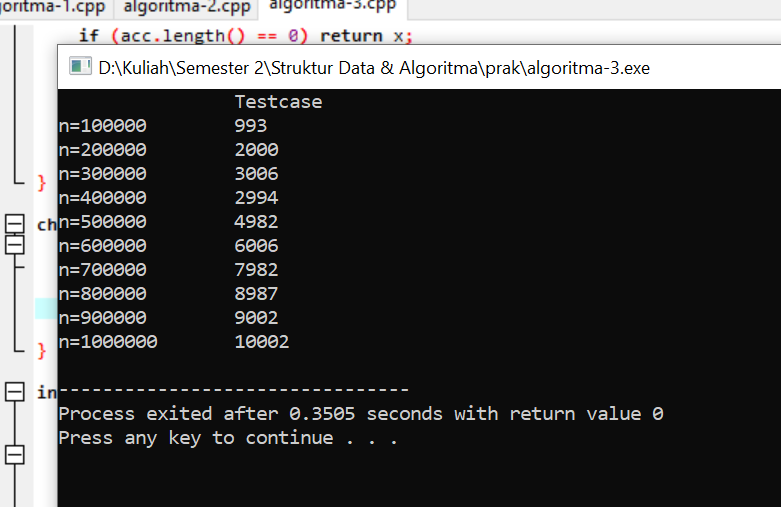
* Semua kasus 🡪 O(n)

1. Testing

Untuk men-*generate* string dengan karakter yang acak, dibutuhkan fungsi pendukung yaitu fungsi **randchar**. Karakter yang akan dirandom adalah lowercase alphabet ditambah dengan “[“, “:”, “|”, dan “]” (total ada 31 karakter)

Pada test ini digunakan n dengan initial value 100000, batas n<= 1000000, dan iterasi n=n+ 100000





1. Grafik

**Kesimpulan**

Algoritma-1 dan Algoritma-2 memiliki task yang sama yaitu mengurutkan array dari kecil ke besar. Dapat dilihat bahwa Algoritma-2 memiliki runtime yang lebih cepat daripada Algoritma-1 sehingga dapat disimpulkan bahwa Algoritma-2 lebih bagus daripada Algoritma-1