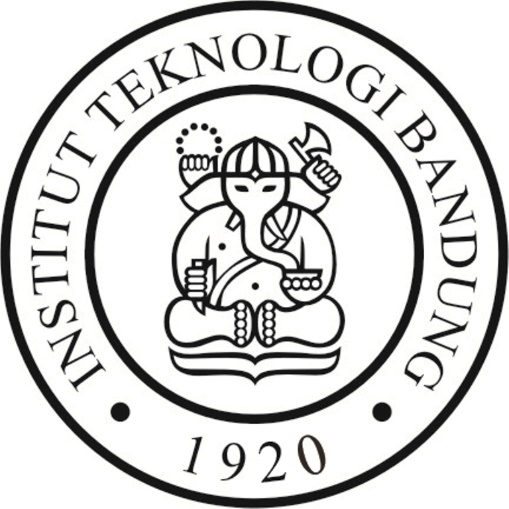
**TUGAS KECIL 1**

**IF2211 STRATEGI ALGORITMA**

***CONVEX HULL ALGORITHM***

***BRUTE FORCE***



**Disusun oleh:**

**Jones Napoleon**

**13518086**

**Mata Kuliah Strategi Algoritma**

**Semester 2 Tahun Ajaran 2019/2020**

**Prodi Teknik Informatika**

**Sekolah Teknik Elektro dan Informatika**

**Institut Teknologi Bandung 2020**

**DAFTAR ISI**

DAFTAR ISI

1. [**CONVEX HULL DAN ALGORITMA BRUTE FORCE** 2](#_Toc30549302)

[**1.1. *OVERVIEW*** 2](#_Toc30549303)

[**1.2. STRUKTUR DATA** 2](#_Toc30549304)

[**1.3. ALGORITMA** 2](#_Toc30549305)

[**2. IMPLEMENTASI KODE PROGRAM DALAM BAHASA C** 4](#_Toc30549306)

[**2.1. HEADER *POINT.H*** 4](#_Toc30549307)

[**2.2. HEADER *ARRAY.H*** 4](#_Toc30549308)

[**2.3. *MAIN PROGRAM*** 6](#_Toc30549309)

[**3. *SCREENSHOT* HASIL PROGRAM** 7](#_Toc30549311)

[**3.1. INPUT BERJUMLAH 5** 7](#_Toc30549312)

[**3.2. INPUT BERJUMLAH 10** 8](#_Toc30549313)

[**3.3. INPUT BERJUMLAH 20** 8](#_Toc30549314)

[**4. LAIN-LAIN** 9](#_Toc30549316)

[**4.1. PENILAIAN INDIVIDU** 9](#_Toc30549317)

[**4.2. SPESIFIKASI PERANGKAT KERAS YANG DIGUNAKAN** 9](#_Toc30549318)

**BAB I**

# **CONVEX HULL DAN ALGORITMA BRUTE FORCE**

## **1.1. *Overview***

*Convex Hull* merupakan himpunan *convex* terkecil yang mengandung atau mengenkapsulasi semua himpunan titik yang ada, biasanya merupakan titik-titik yang berada pada “lapisan terluar” himpunan titik tersebut. Pada program ini, semua antar-titik diasumsi dihubungkan hanya dengan garis lurus.

Secara dasar, program menerima *input* berupa jumlah titik (anggap *N* buah), kemudian program akan menggenerasikan titik-titik sebanyak *N* kali. Setelah itu, program akan memproses benar tidaknya masing-masing kombinasi dua titik (satu garis) merupakan titik yang sesuai dengan *Convex Hull*. Kumpulan titik-titik yang sesuai tersebut kemudian dikembalikan.

## **1.2. Struktur data**

Secara intuitif, program membutuhkan model berupa *point* (titik) yang mempunyai struktur data *absis* (X) dan *ordinat* (Y) – keduanya bertipe *float*. Selain itu, program juga memerlukan model berupa *array of point* yang akan menampung semua titik baik setelah digenerasi secara acak maupun setelah diproses dan siap dikembalikan. Mengikuti konvensi Institut Teknologi Bandung, indeks di *array* dimulai dari 1.

|  |  |
| --- | --- |
|  |  |
| Gambar 1. Struktur Data *POINT* | Gambar 2. Struktur Data *array of POINT* |

## **1.3. Algoritma**

Program setelah menyimpan kumpulan titik pada suatu *array* (*array\_init*) dengan cara digenerasi secara acak, selanjutnya akan melakukan iterasi secara kombinatorial. Misalkan ada *N* buah titik. Program akan mulai dari titik di *array* berindeks 1 dengan melakukan pengecekan titik di *array* berindeks 1 dengan titik di *array* berindeks 2. Kemudian titik di *array* berindeks 1 dengan titik di *array* berindeks 3, hingga titik di *array* berindeks 1 dengan titik di *array* berindeks N. Satu iterasi elemen *array* selesai. Kemudian titik di *array* berindeks 2 dengan titik di *array* berindeks 3, sampai dengan titik di *array* berindeks *N*, hingga akhirnya kombinasi titik di *array* berindeks *N*-1 dengan titik di *array* berindeks *N*.

Pengecekan dua titik diatas dilakukan dengan melakukan persamaan garis:

***y = b + m (x – a)***

, dimana ***m***merupakan gradien kedua titik tersebut dengan ***b*** dan ***a*** secara berturut-turut merupakan ordinat dan absis dari salah satu titik tersebut.

Program kemudian mengiterasi semua titik selain kedua titik percobaan tersebut dengan cara disubstitusikan nilai absisnya ke persamaan tersebut. Jika nilai ***y*** yang dihasilkan lebih besar dari nilai ordinat titik tersebut, maka program akan mengembalikan ***true***, atau ***false*** jika sebaliknya (konvensi penulis). Nilai ***false*** atau ***true*** ini kemudian akan di-*append* ke suatu *array* baru hingga semua titik lainnya telah diuji dengan persamaan garis tersebut. Akhirnya kedua titik tersebut dikatakan titik-titik pada *convex hull* jika dan hanya jika semua *array* *boolean* tersebut memiliki nilai *boolean* yang sama.

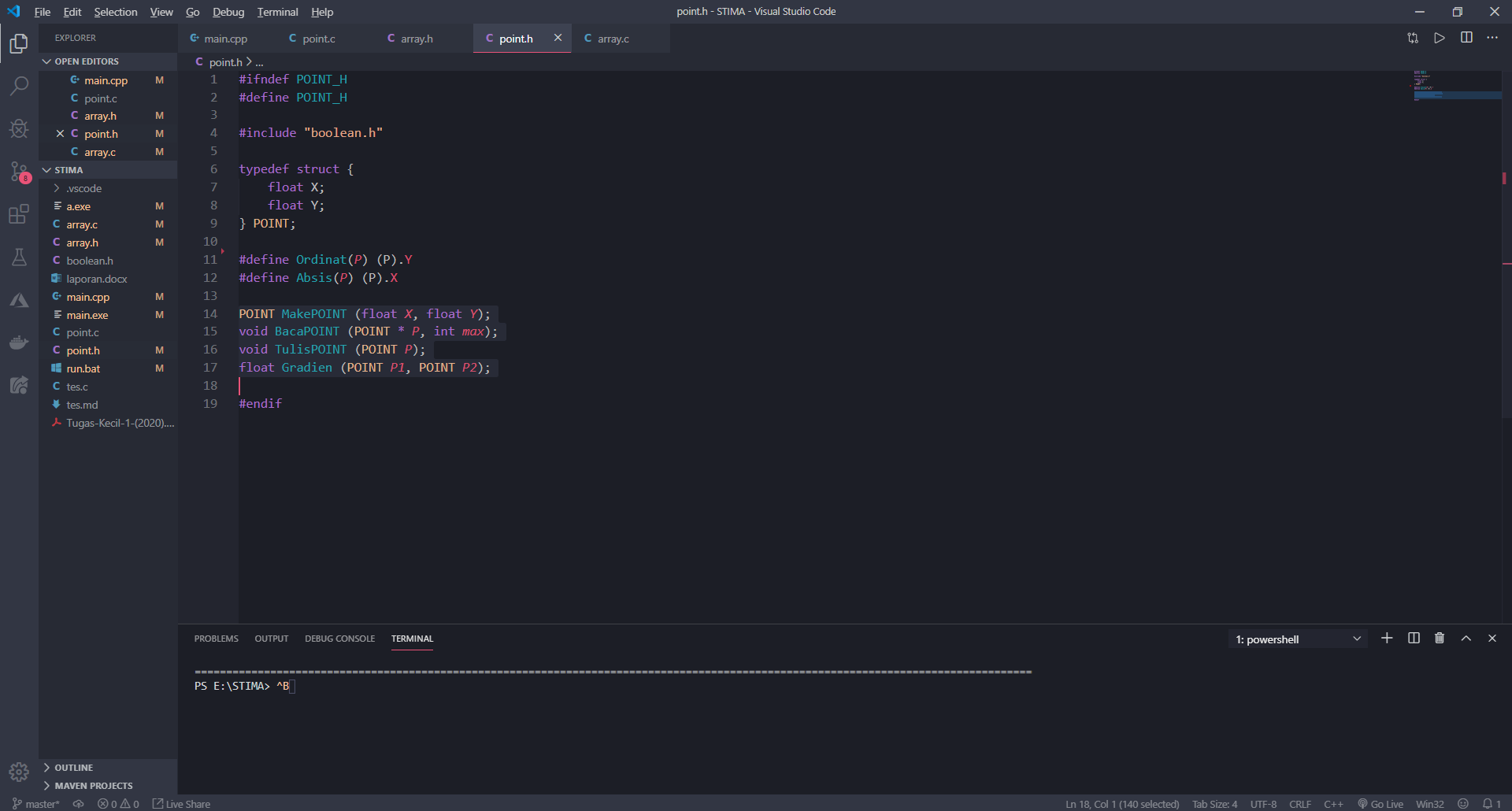
Dari paragraf pertama dilakukannya iterasi titik secara kombinatorial, kompleksitasnya *big-O-*nya telah berupa karena untuk setiap titik dilakukan pemasangan dengan titik-titik lainnya. Kemudian dari percobaan masing-masing titik dengan persamaan garis tersebut meningkatkan kompleksitasnya sebesar , sehingga kompleksitas akhir programnya adalah .

**BAB II**

# **IMPLEMENTASI KODE PROGRAM DALAM BAHASA C**

## **2.1. HEADER *POINT.H***

Di bawah ditampilkan keempat fungsi dan prosedur yang ada pada header *point.h*



Gambar 3. Fungsi dan prosedur pada header *point.h*

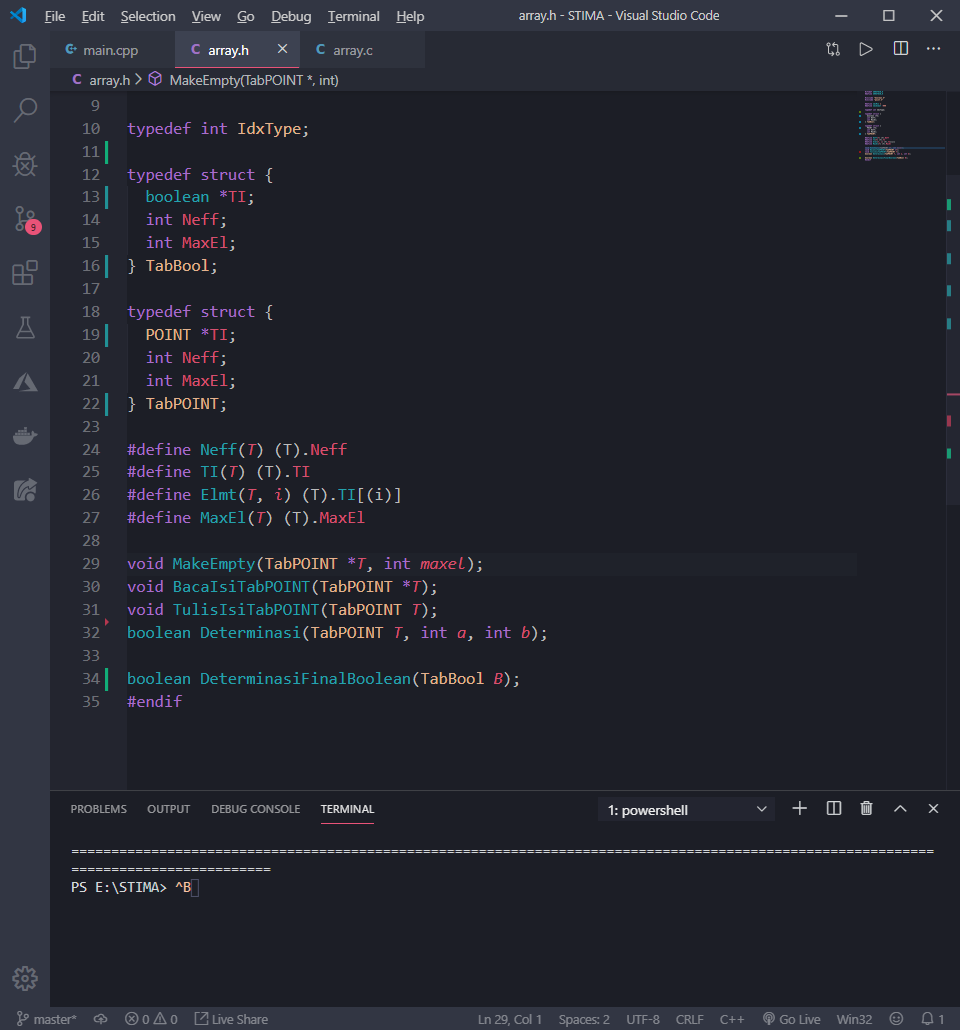
Fungsi **MakePOINT** mengembalikan sebuah **POINT** dari input parameter absis (*X*) dan ordinat (*Y*). Prosedur **TulisPOINT** menuliskan sebuah **POINT**dalam bentuk “(<**absis**>, <**ordinat**>)”. Kedua fungsi tidak ditampilkan lagi *screenshot* metodenya mengingat abstraksinya terlalu rendah.

Di bawah ditampilkan prosedur **BacaPOINT** yang membentuk suatu *point* dengan acak dan batasan nilai ***max*** pada kedua absis dan ordinatnya. Fungsi **Gradien** yang menerima *input* dua titik juga ditampilkan dibawah. Mengingat gradient membagi perbedaan ordinat dengan perbedaan absis, maka jika kedua titik memiliki absis yang sama, maka secara matematika akan menghasilkan nilai tak terhingga. Untuk mengkompensasi hal tersebut, dilakukan pengecekan kesamaan nilai absis terlebih dahulu dan dikembalikan suatu nilai relatif besar (9999) untuk mengatasi *error* tersebut.

|  |  |
| --- | --- |
|  |  |
| Gambar 4.Implementasi prosedur **BacaPOINT** | Gambar 5. Implementasi fungsi **Gradien** |

## **2.2. HEADER *ARRAY.H***

Sesuai metode yang ditulis diatas, program membutuhkan beberapa metode dasar untuk operasi *array* sebagai berikut.

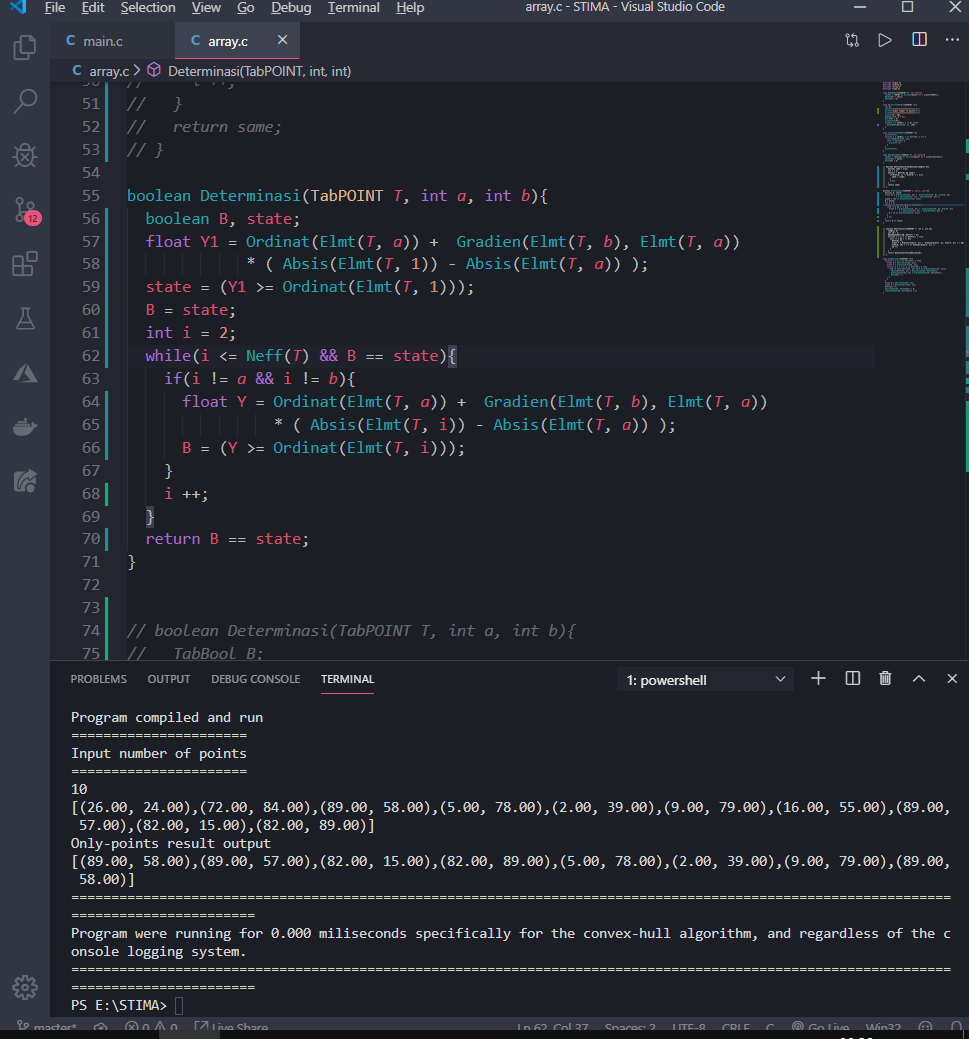


Gambar 6. Fungsi dan prosedur pada header *array.h*

**MakeEmpty**, **BacaIsiTabPOINT** dan **TulisIsiTabPOINT** merupakan prosedur *generic* yang hanya melakukan metode seperti bagaimana semestinya (tidak ditampilkan lagi karena merupakan bentuk abstraksi biasa dari program pada mata kuliah Algoritma dan Struktur Data).

**MakeEmpty** menerima *pointer array* dan nilai ***maxel*** sebagai parameternya dan menginisiasi *array baru TABPOINT* dengan jumlah ***maxel***. **BacaIsiTabPOINT**awalnya membaca *input* *user* dengan suatu nilai ***N***, dan kemudianmengisi *array* yang disediakan pada parameternya dengan memanggil *procedure* **BacaPOINT** dari header *point.h* sebanyak ***N*** iterasi. Di sisi lain, **TulisIsiTabPOINT** memanggil *procedure* **TulisPOINT** untuk iterasi setiap elemennya, dipisahkan dengan sebuah koma (,), dan dimulai dan diakhiri dengan [].

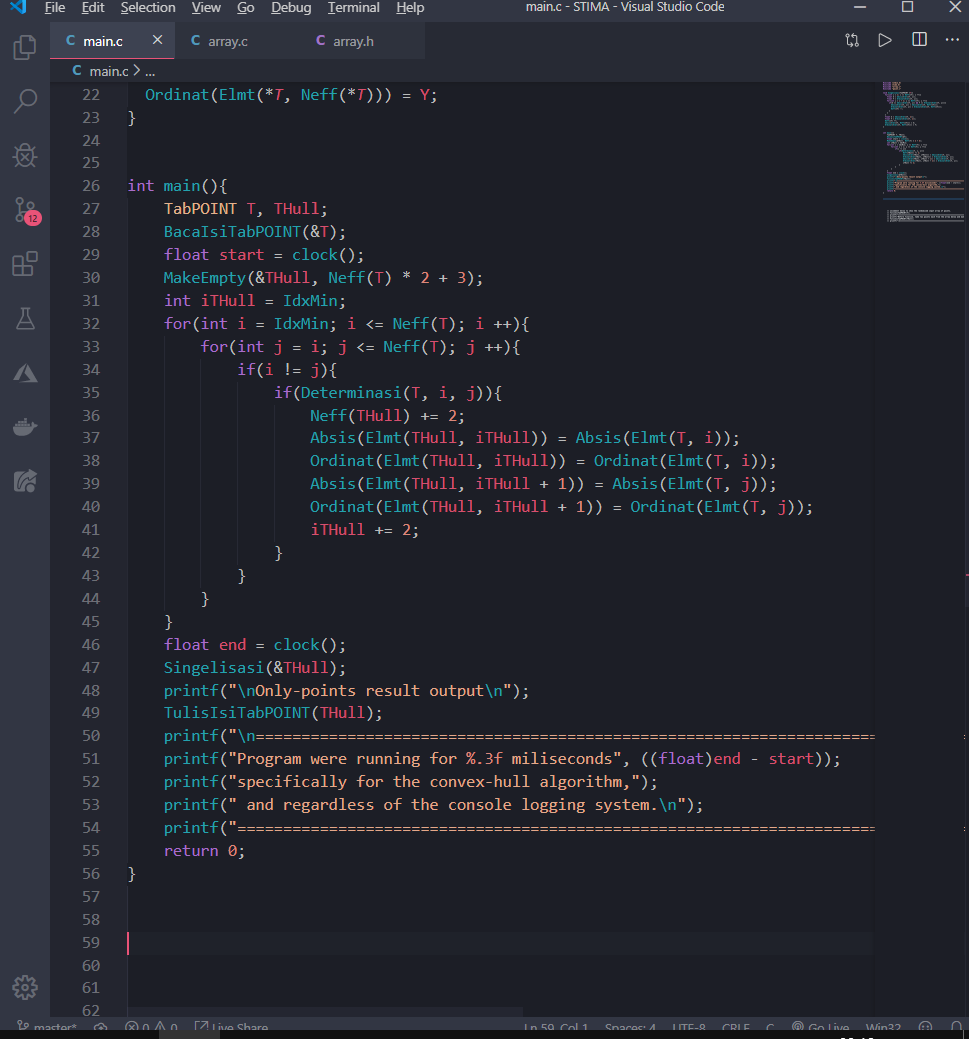
Fungsi **Determinasi** menerima *array of POINT* (**T**), indeks elemen ke-***a*** pada *array* **T** (titik pertama) dan indeks elemen ke-***b*** pada *array* **T** (titik kedua) sebagai parameternya. Kemudian, untuk setiap titik selain elemen *array berindeks-****a*** dan elemen *array berindeks-****b***, dilakukan perbandingan sesuai yang telah dijelaskan di **Bagian 2.2**



Gambar 7. Implementasi fungsi **Determinasi**

## **2.3. *MAIN PROGRAM***

Program utama berjalan sebagai berikut. Dimulai dengan membaca *array of POINT* (**BacaIsiTabPOINT**), menginisiasi sebuah array **THull** dan mengiterasikan semua kombinasi pasangan *POINT* dengan semua titik selain pasangan tersebut, dan men-**DETERMINASI**-kan benar tidaknya pasangan *POINT* tersebut merupakan pasangan titik (garis) *convex hull*. Sampai sini, *array* **THull**akan berisi semua *POINT* sebanyak dua kali. Oleh karena itu, dibuatlah fungsi **Singelisasi**yang mengembalikan *array* **THull** dengan kondisi semua elemen tidak ada yang ganda, sebelum akhirnya menampilkan **THull** final. Berikut ditampilkan program utama (**main**) dan fungsi **Singelisasi**. Program juga akhirnya akan menampilkan waktu pemrosesan algoritma *convex hull*. Akan tetapi, untuk skala kecil (1 <= ***N*** <=100), waktu pemrosesan tidak biasanya diatas 5 milidetik.



Gambar 8. Program utama

## 

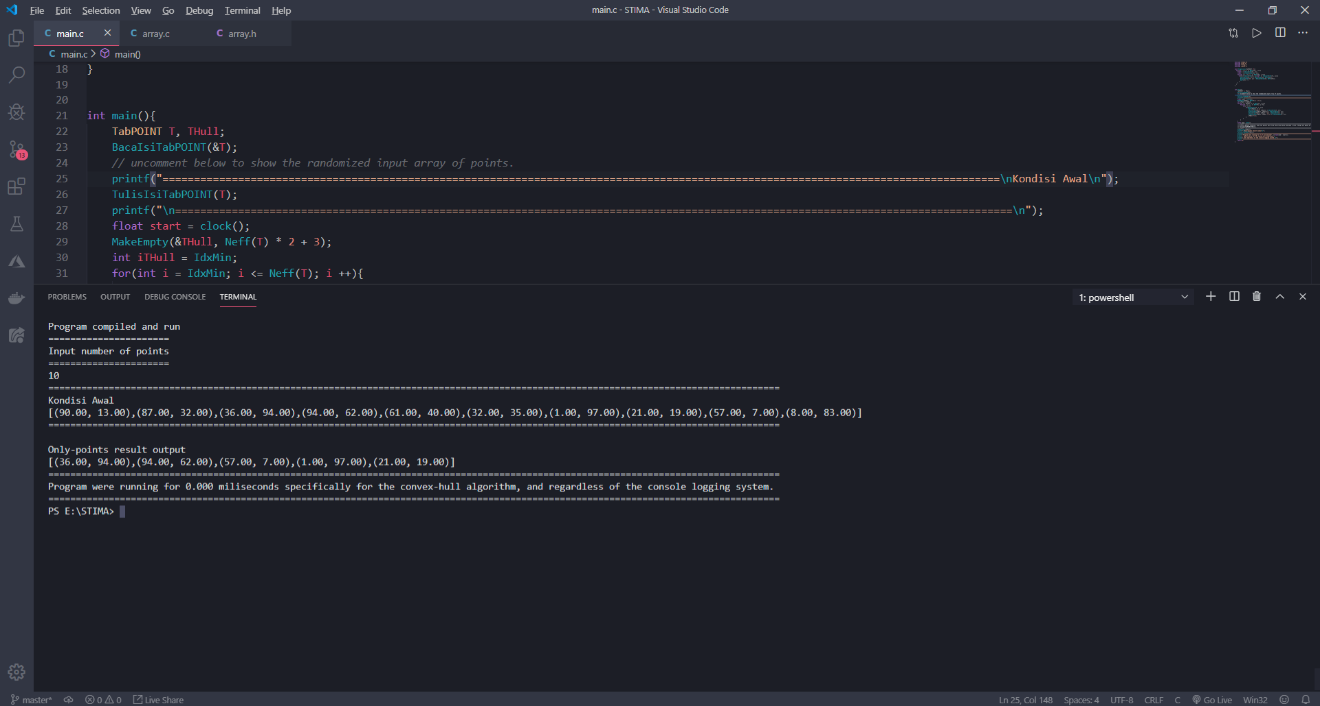
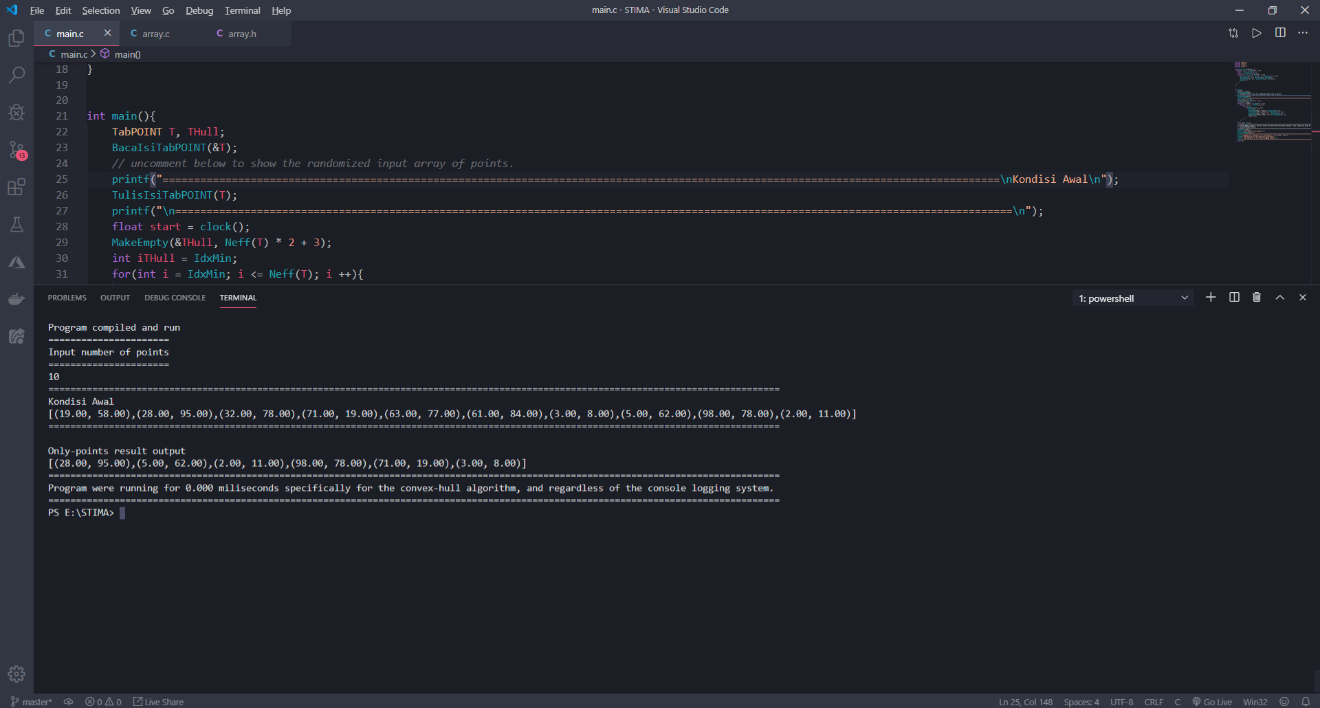
Gambar 9. Fungsi **Singelisasi**

**BAB III**

# ***SCREENSHOT* HASIL PROGRAM**

## **3.1. input berjumlah 5**

## **3.2. input berjumlah 10**



## **3.3. input berjumlah 20**

## 

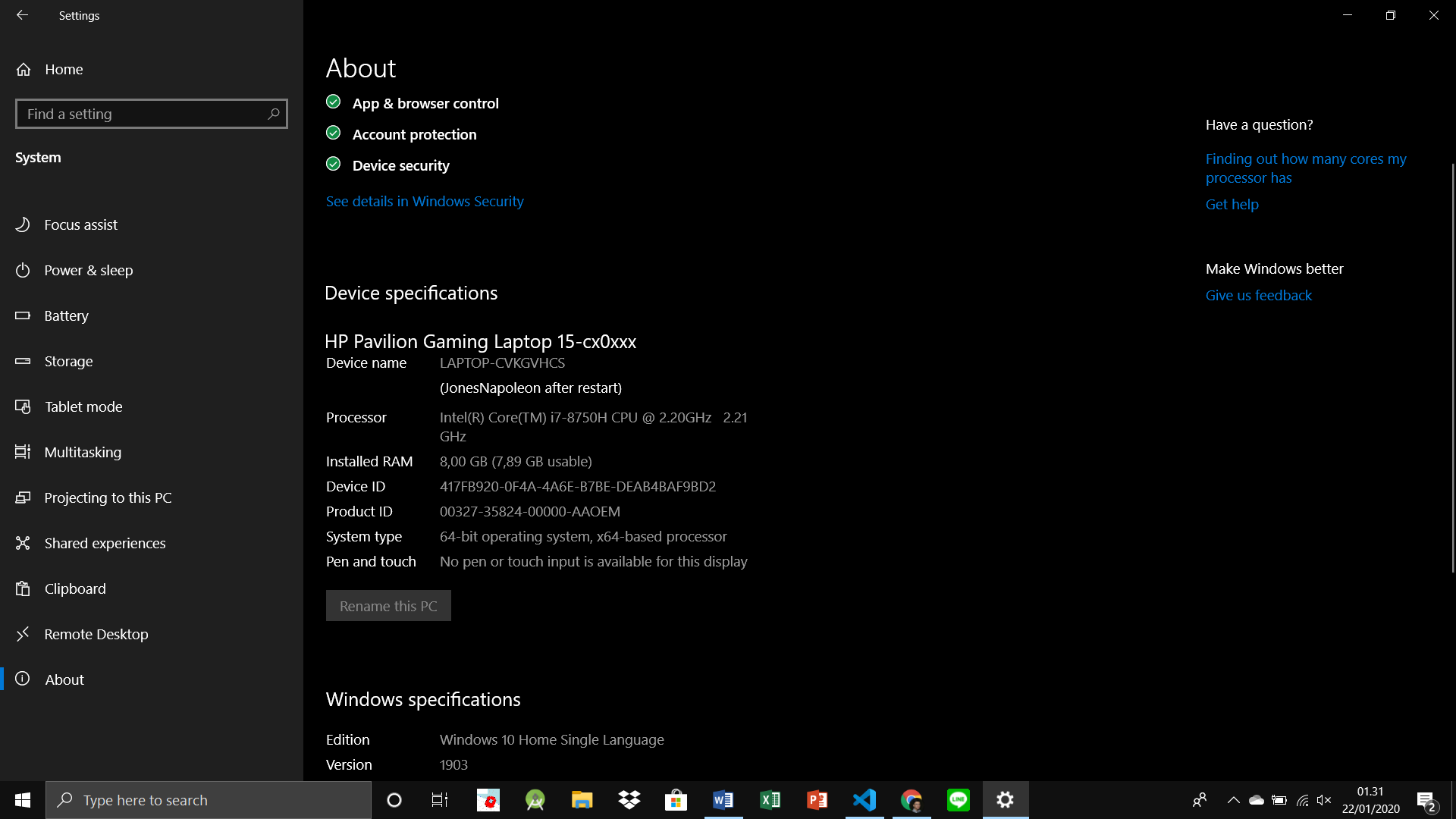
**BAB IV**

# **LAIN-LAIN**

## **4.1. penILAIAN INDIVIDU**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Poin | Ya | Tidak |
| 1. | Program berhasil dikompilasi | √ |  |
| 2. | Program berhasil running | √ |  |
| 3. | Program dapat menerima input dan menuliskan output. | √ |  |
| 4. | Luaran sudah benar untuk semua n | √ |  |

## **4.2. SPESIFIKASI perangkat keras yang digunakan**



Gambar 10. Spesifikasi Laptop