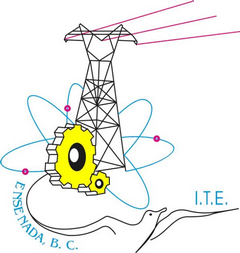
Instituto Tecnológico De Ensenada

Metodos de Ordenamiento de Datos

Inteligencia Artificial



Ingeniería en Sistemas Computacionales

Francisco de Jesus Diaz Amador

Jose Manuel Gonzalez Lara

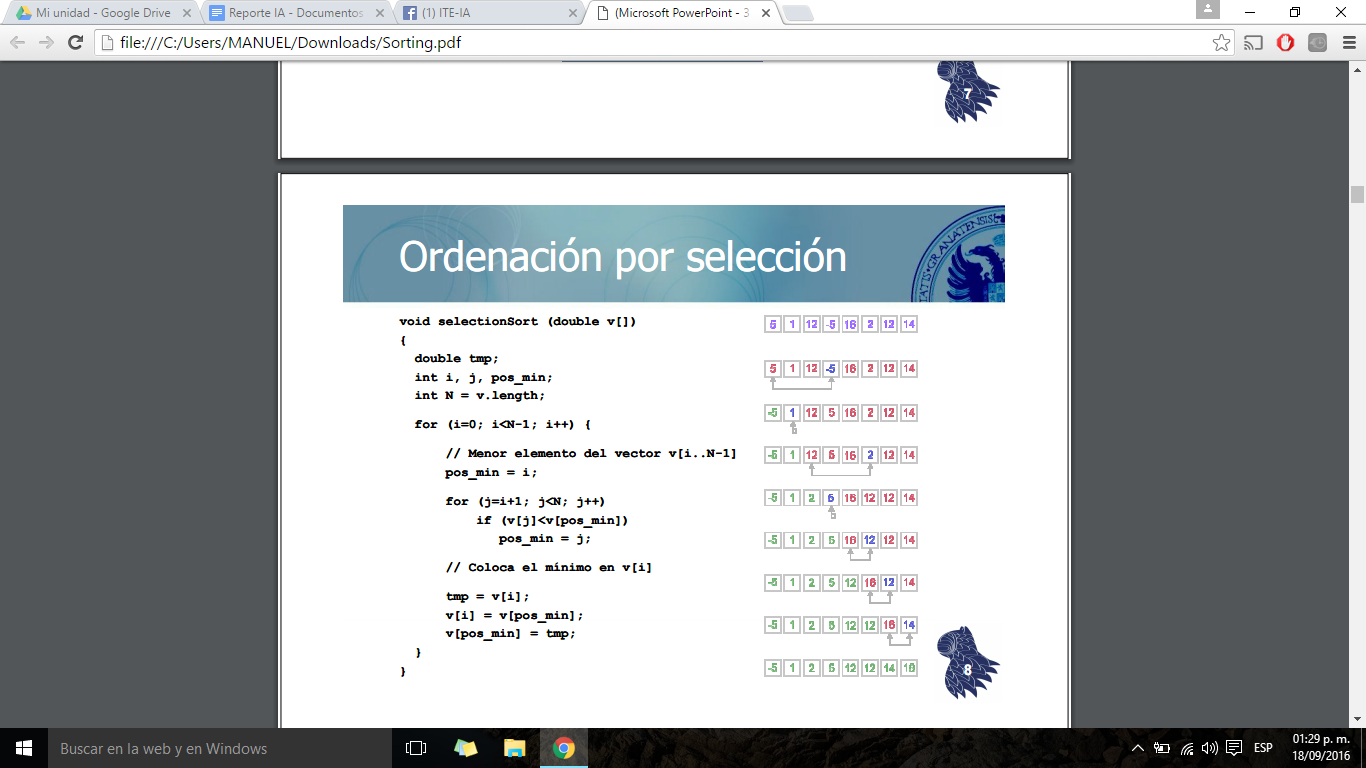
Noveno Semestre

Ensenada B.C a 22 de Septiembre del 2016

**Metodos de ordenamiento de datos**

**Ordenación por selección**

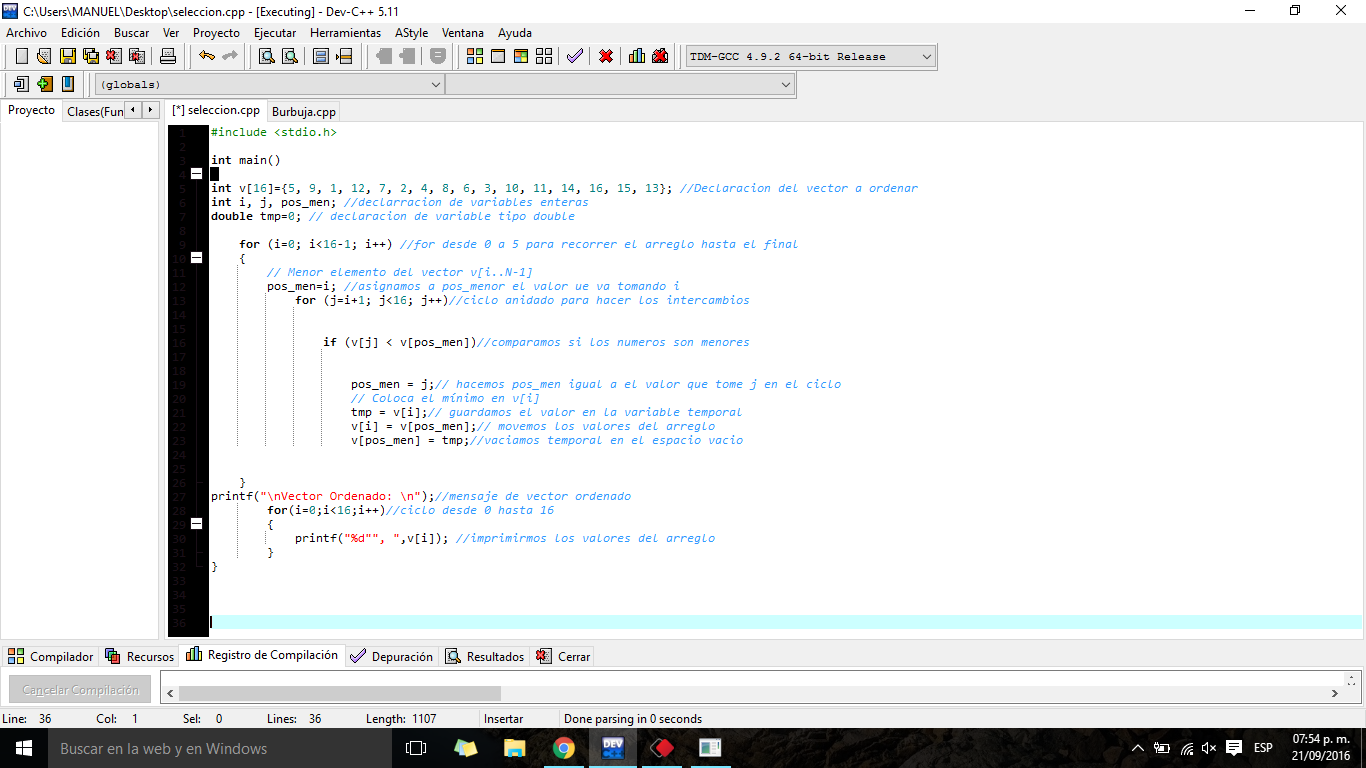
En este metodo de ordenacion, se toma el numero menor de un vector de datos no ordenados, intercambiando este número por el que se encuentra en la primera posición del vector como se muestra a continuación:



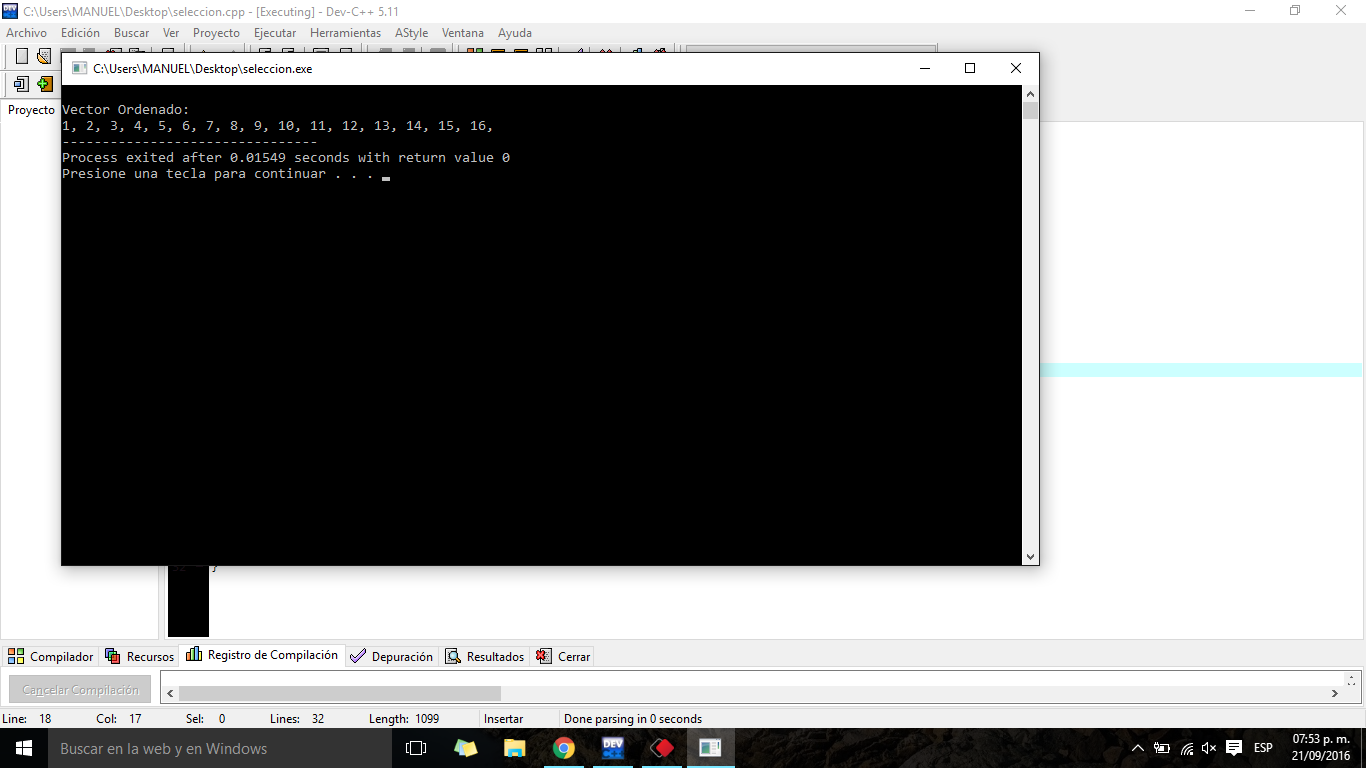
**Diagrama de Flujo**



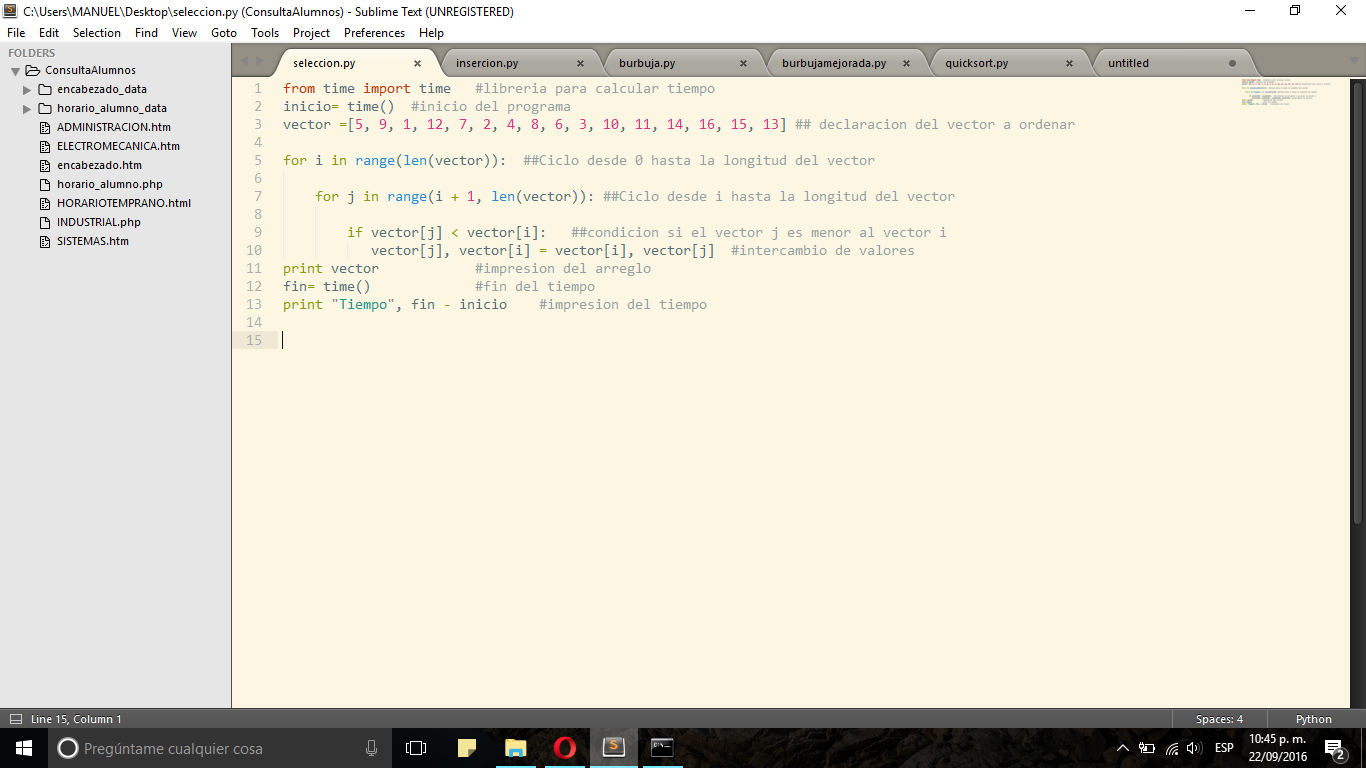
**Código en C**



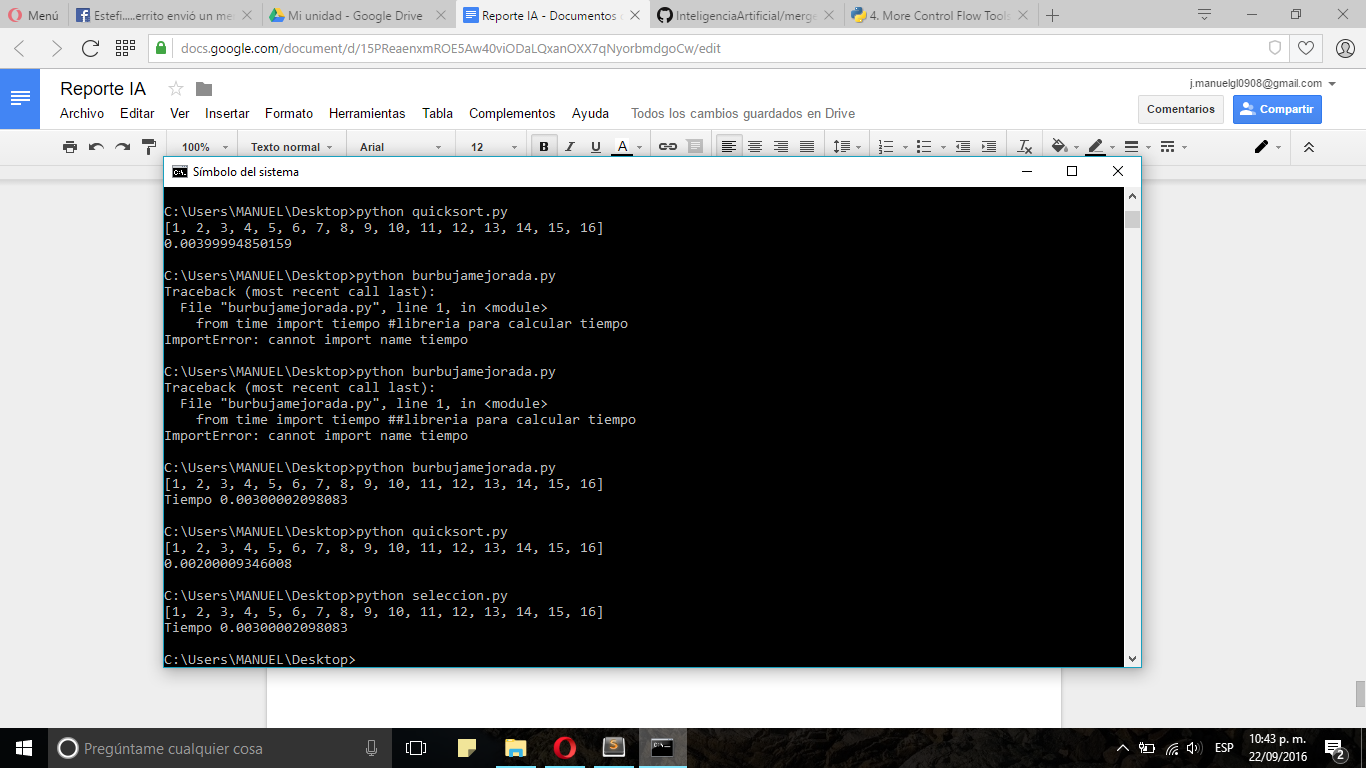
**Ejecución**



**Código en python**



**Ejecución**



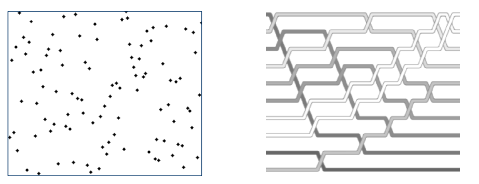
**Conclusión:**

El método de selección suele ser uno de los más comunes dado a que es lo que normalmente una persona realiza con la vista para ordenar una lista de números dibujada, por lo tanto es muy fácil de comprender su funcionamiento

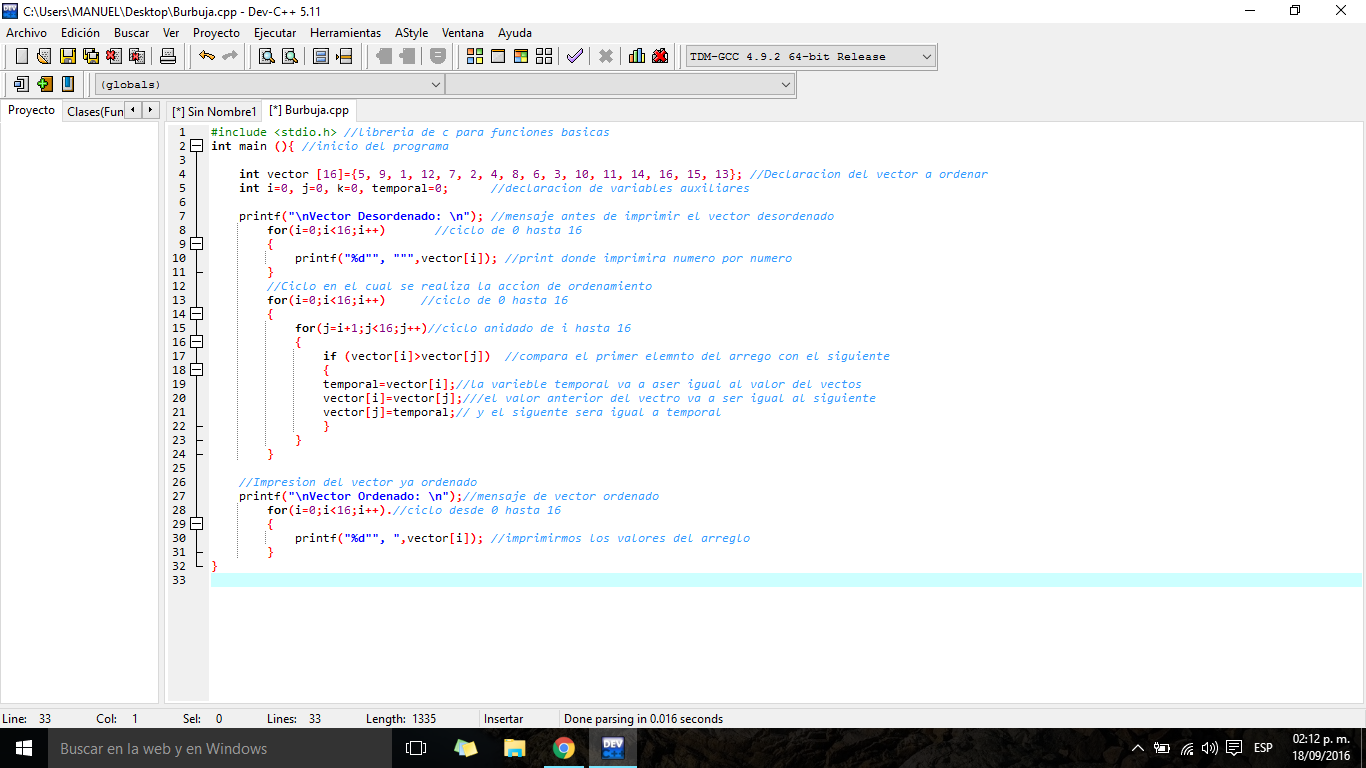
**Método de la burbuja**

**Descripción**

El procedimiento de la burbuja es el siguiente: Ir comparando desde la casilla 0 número tras número hasta encontrar uno mayor, si este es realmente el mayor de todo el vector se llevará hasta la última casilla, si no es así, será reemplazado por uno mayor que él. Este procedimiento seguirá así hasta que haya ordenado todas las casillas del vector. Una de las deficiencias del algoritmo es que ya cuando a ordenado parte del vector vuelve a compararlo cuando esto ya no es necesario.



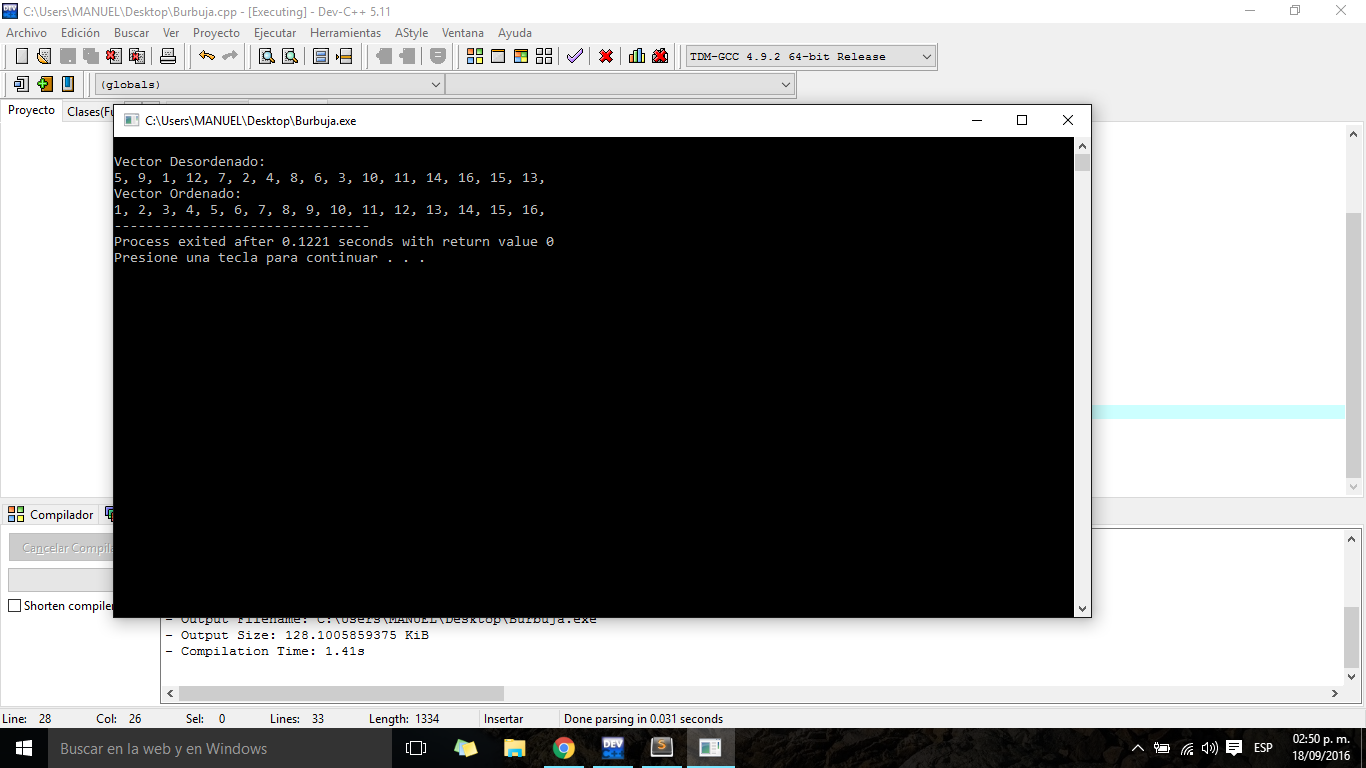
**Código en C**



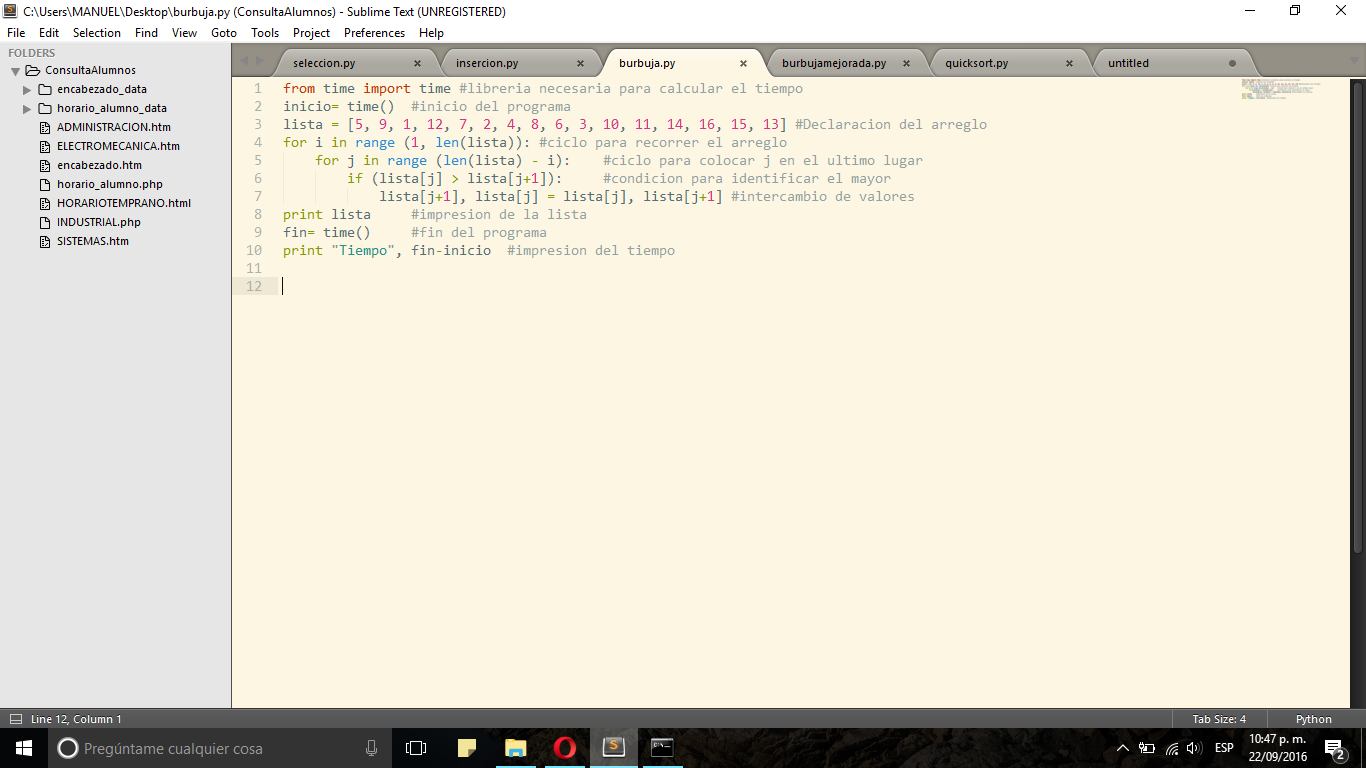
**Diagrama de flujo**

Untitled Diagram.png

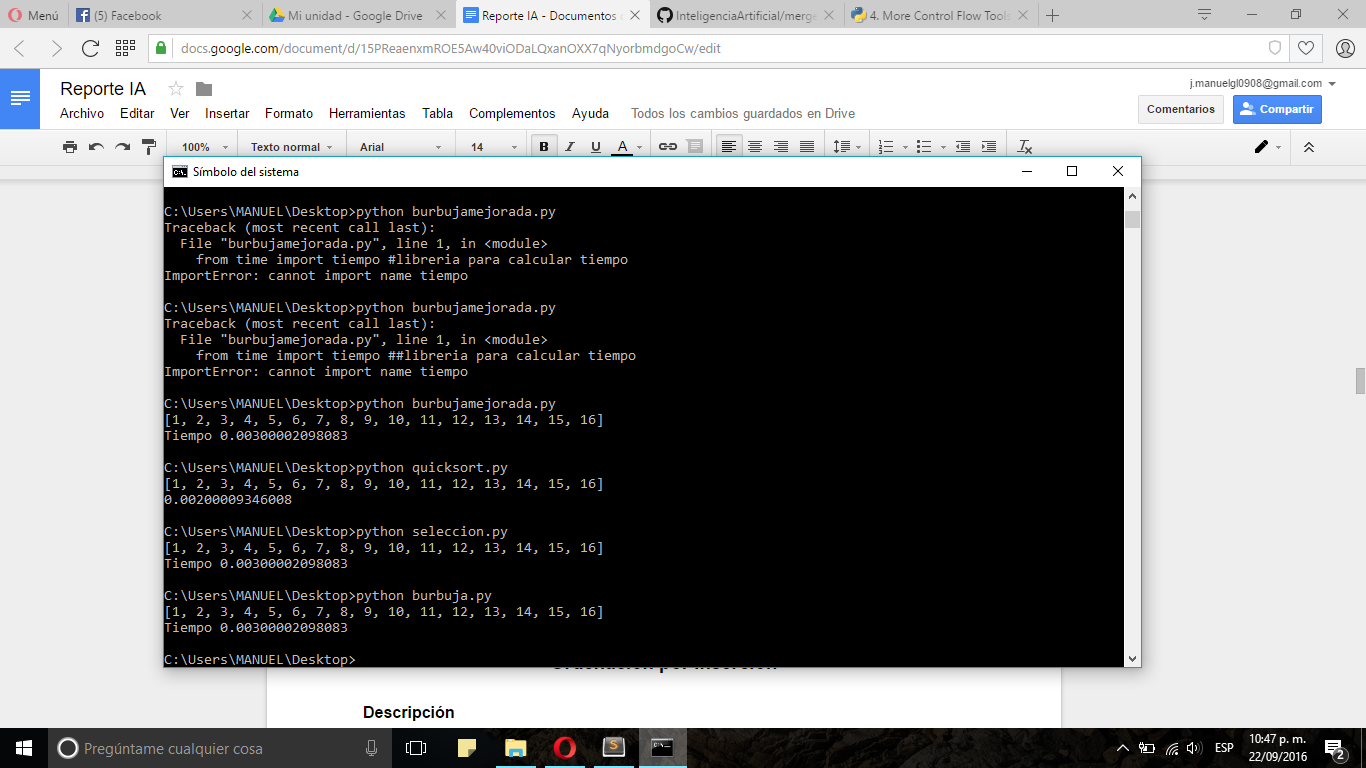
**Ejecución**



**Código en python**



**Ejecución**



**Conclusión:**

El método de la burbuja es muy bueno y confiable para proyectos pequeños donde siempre se tenga que ordenar una misma cantidad de números , ya que en una situación diferente donde encuentra mayor cantidad de números el proceso sería bastante lento ya que su algoritmo no es óptimo para ello.

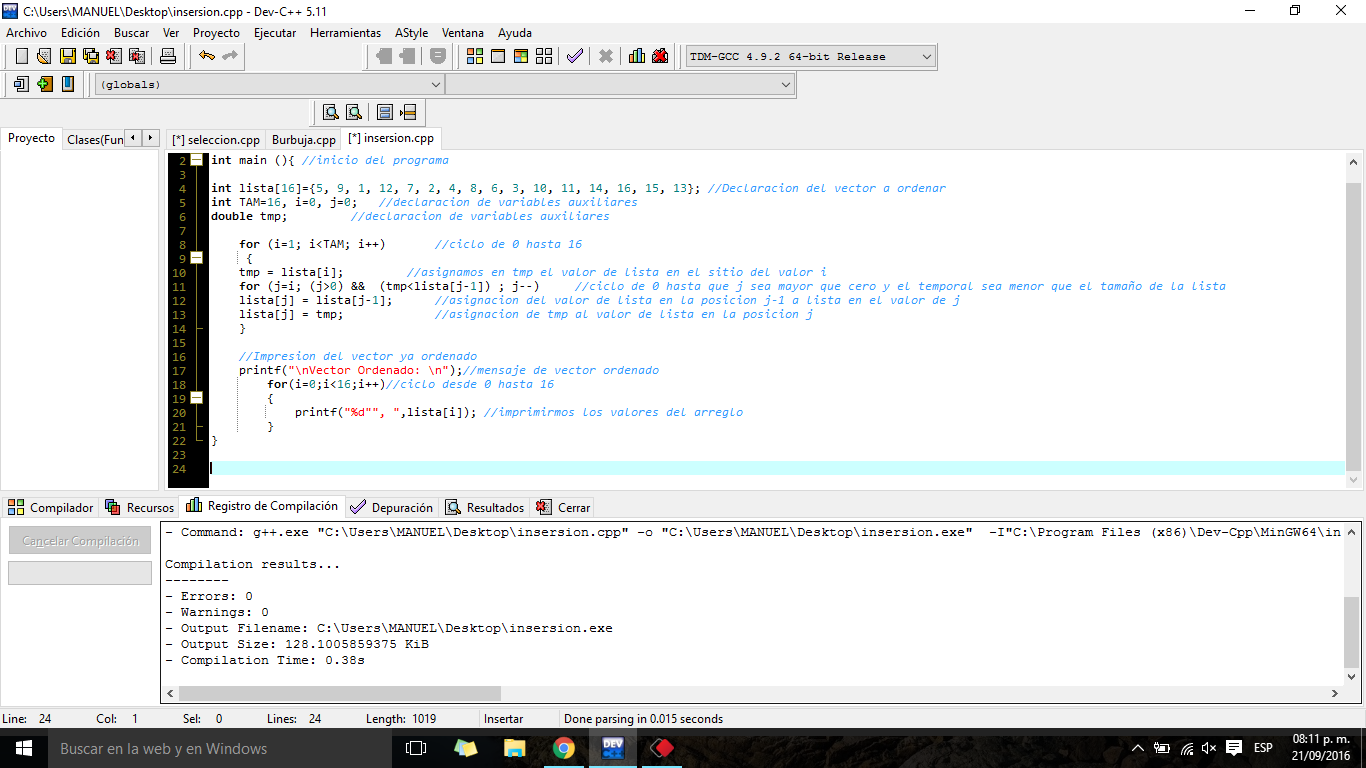
**Ordenación por inserción**

**Descripción**

La idea de este algoritmo de ordenación consiste en ir insertando un elemento de la lista ó un arreglo en la parte ordenada de la misma, asumiendo que el primer elemento es la parte ordenada, el algoritmo irá comparando un elemento de la parte desordenada de la lista con los elementos de la parte ordenada, insertando el elemento en la posición correcta dentro de la parte ordenada, y así sucesivamente hasta obtener la lista ordenada.



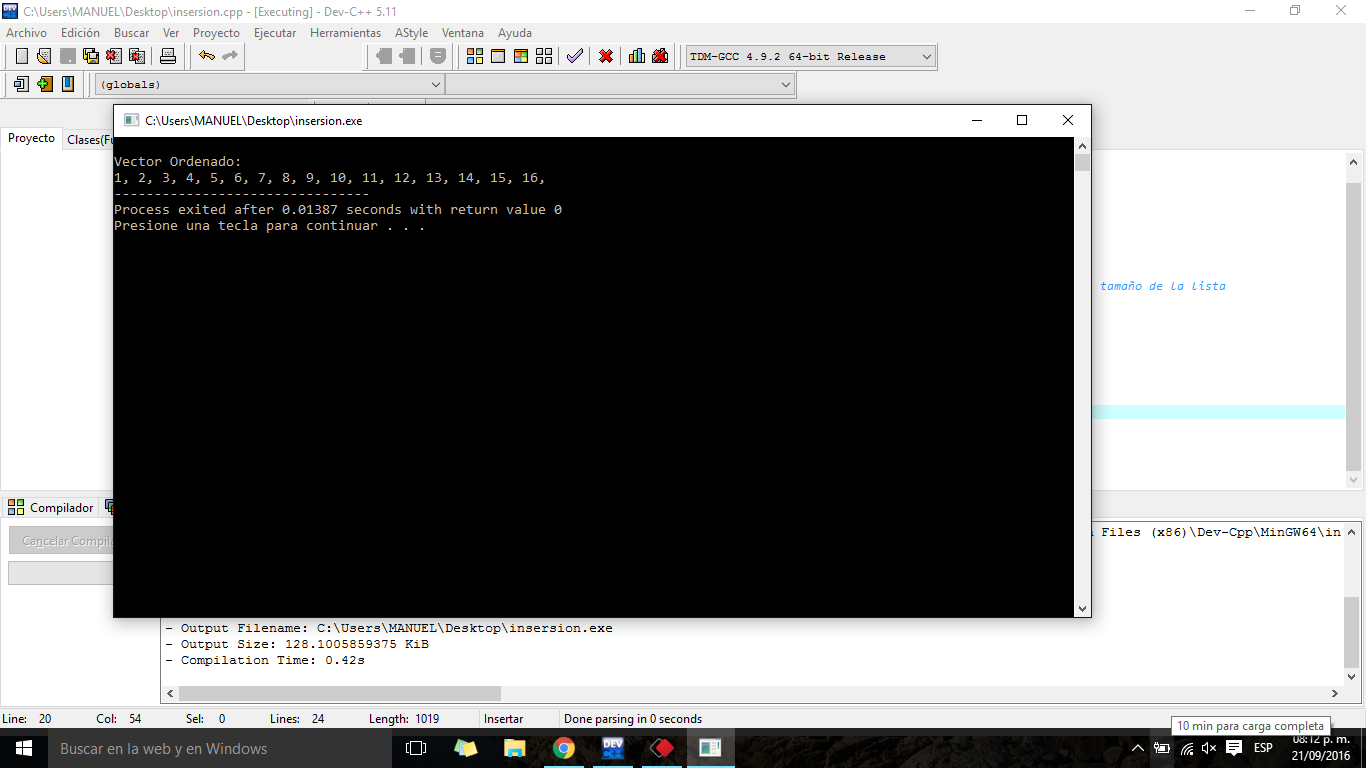
**Código en C**



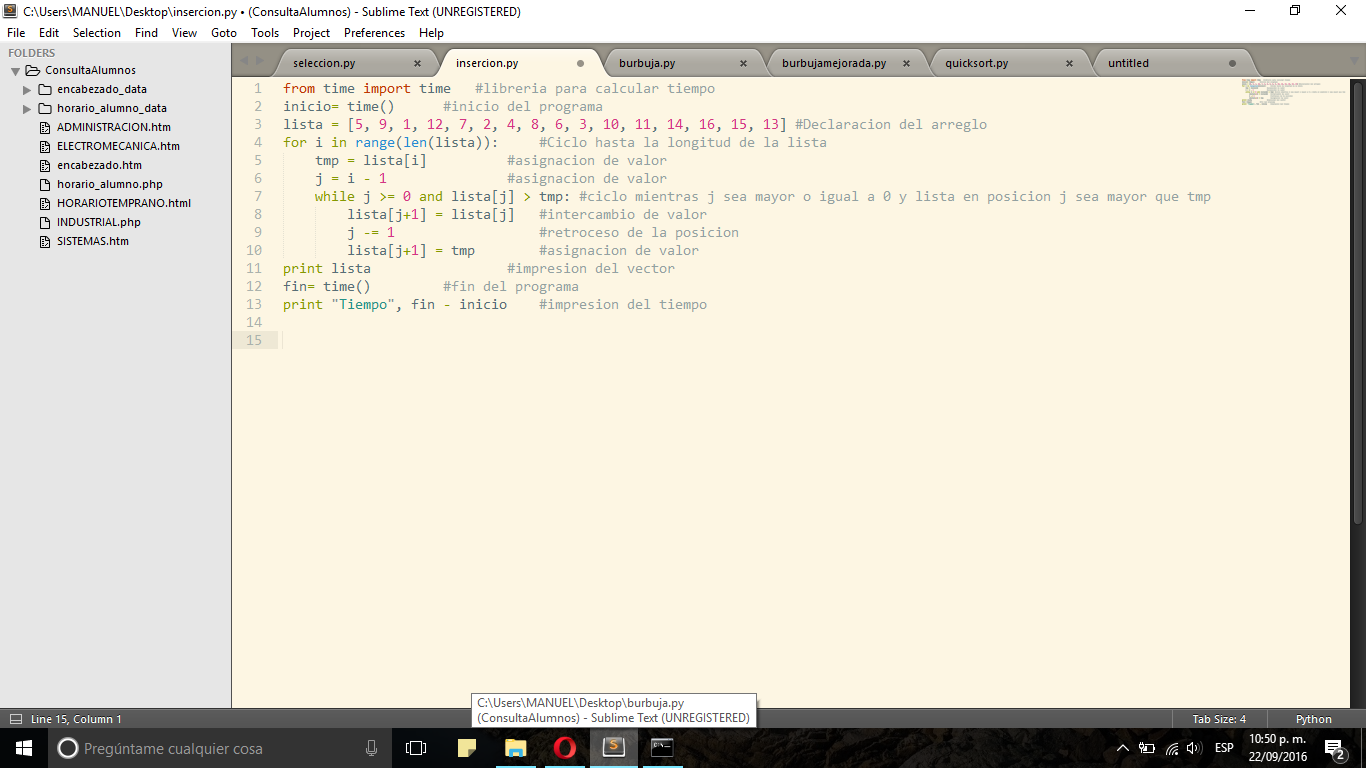
**Diagrama de Flujo**



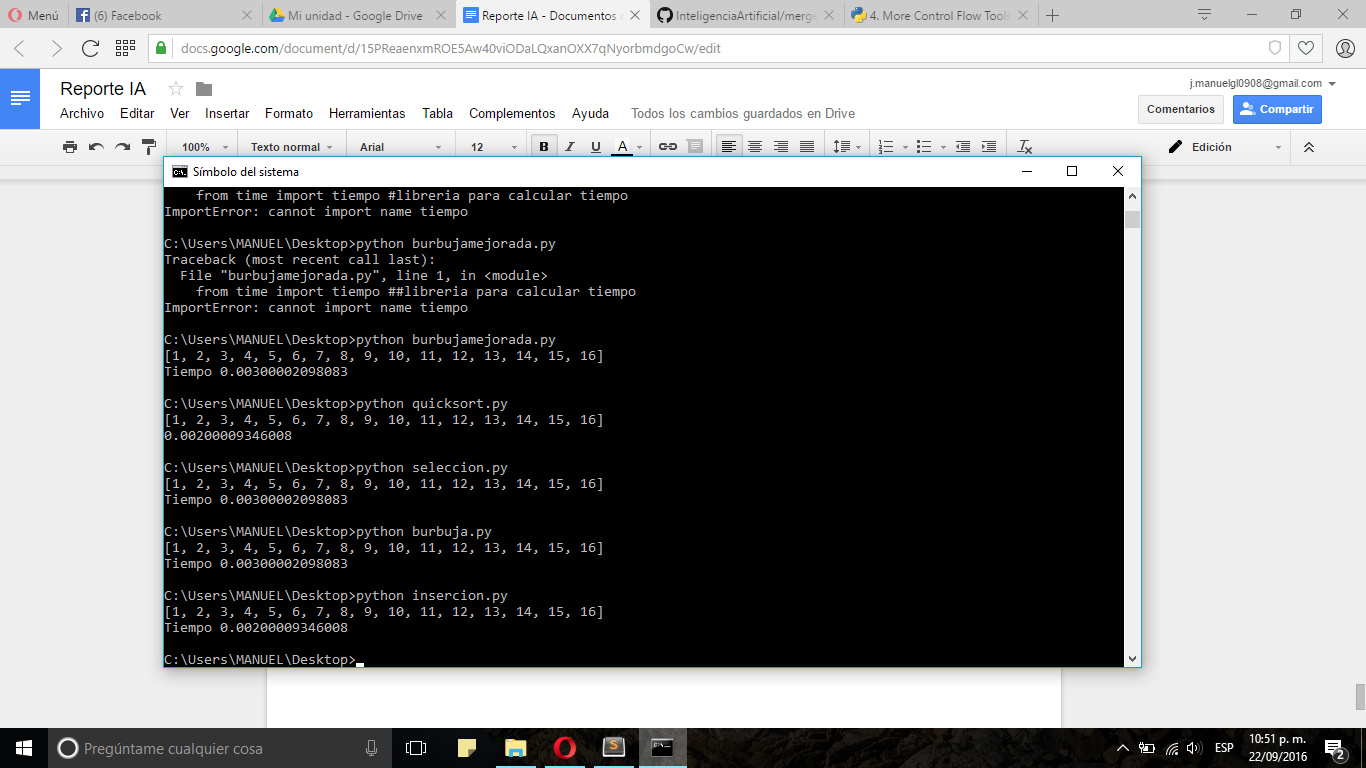
**Ejecución**



**Código en python**



**Ejecución**



**Conclusión:**

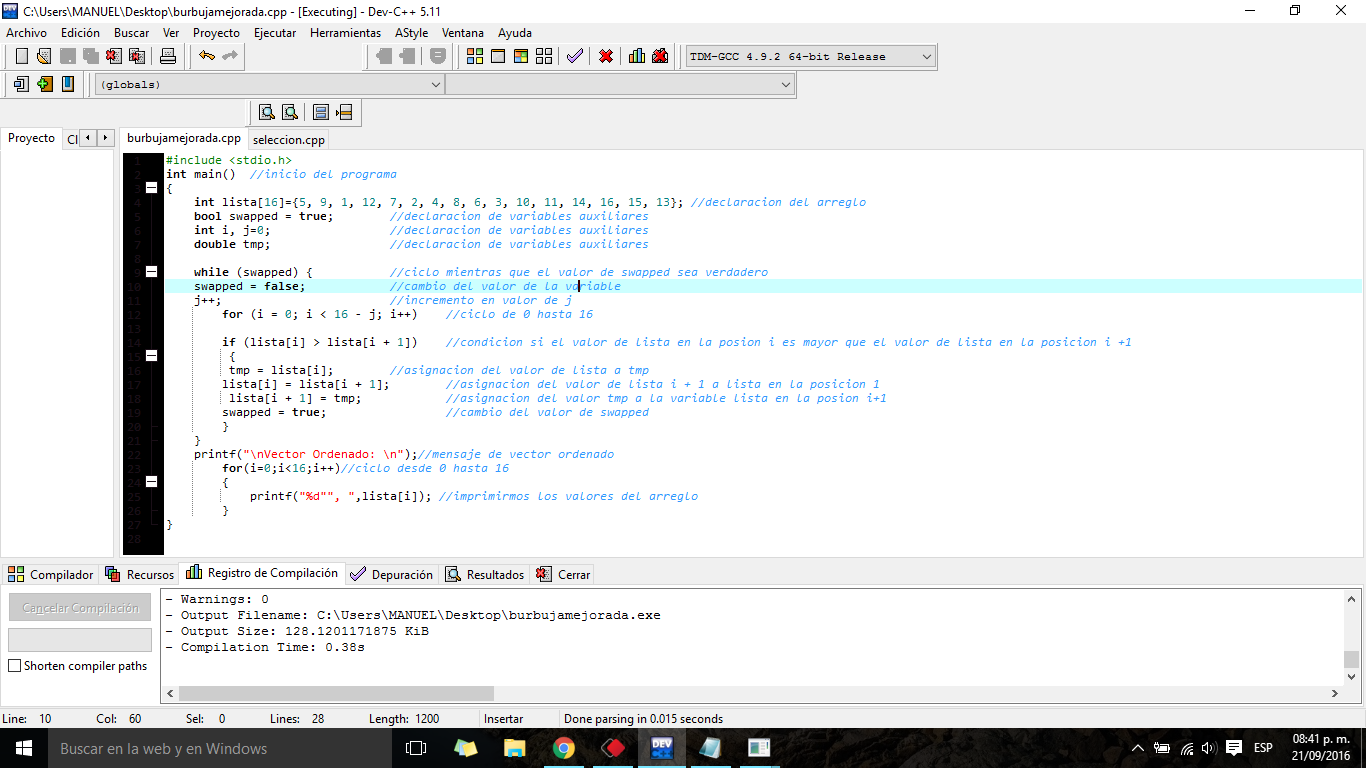
El método de inserción es bastante aplicable en proyectos donde hay que hacer un intercambio directo y eficaz ya que el ciclo se recorre menos veces porque se va insertando al paso que lo va recorriendo , por eso mismo fue uno de los algoritmos que menos tiempo tardó en su ordenamiento

**Burbuja Mejorada**

**Descripción**

Usa una centinela y termina cuando en una iteración el bucle anidado no se produzcan intercambios

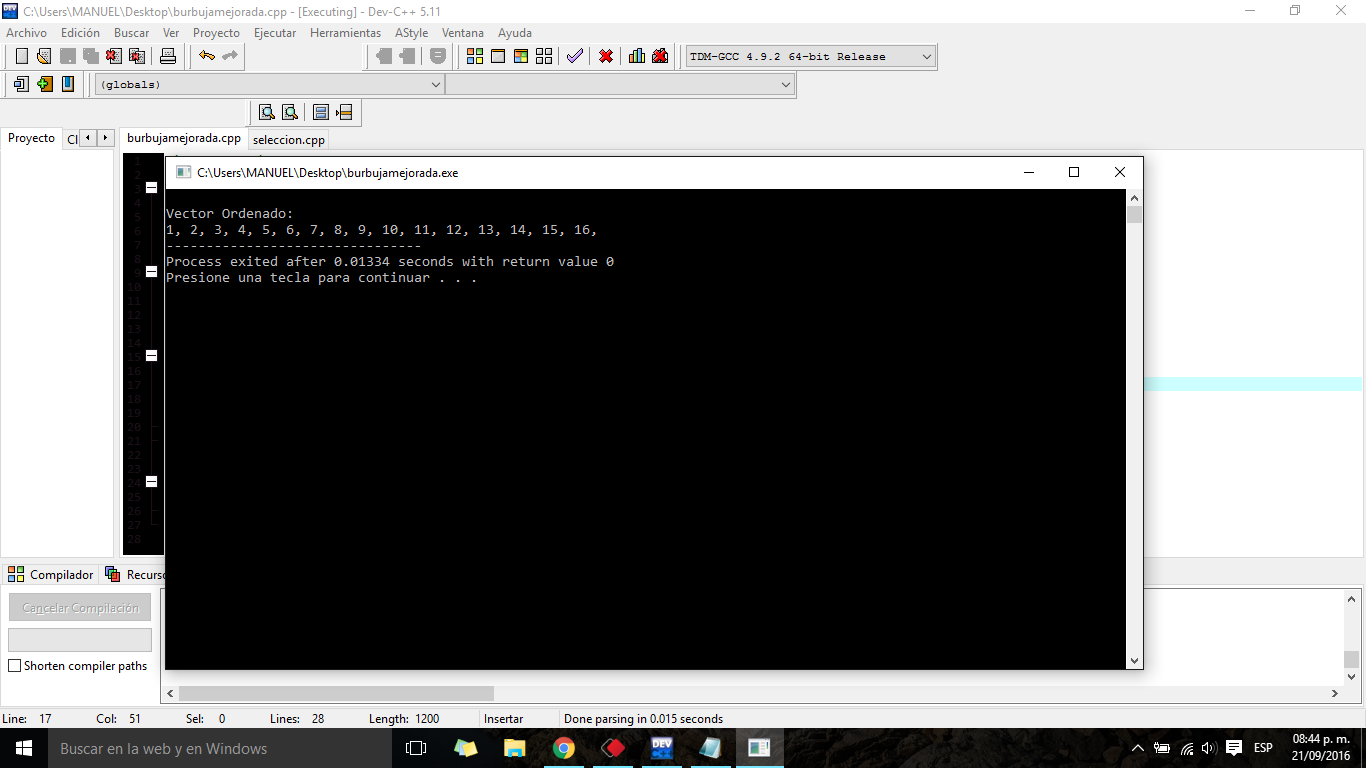
**Código en C**



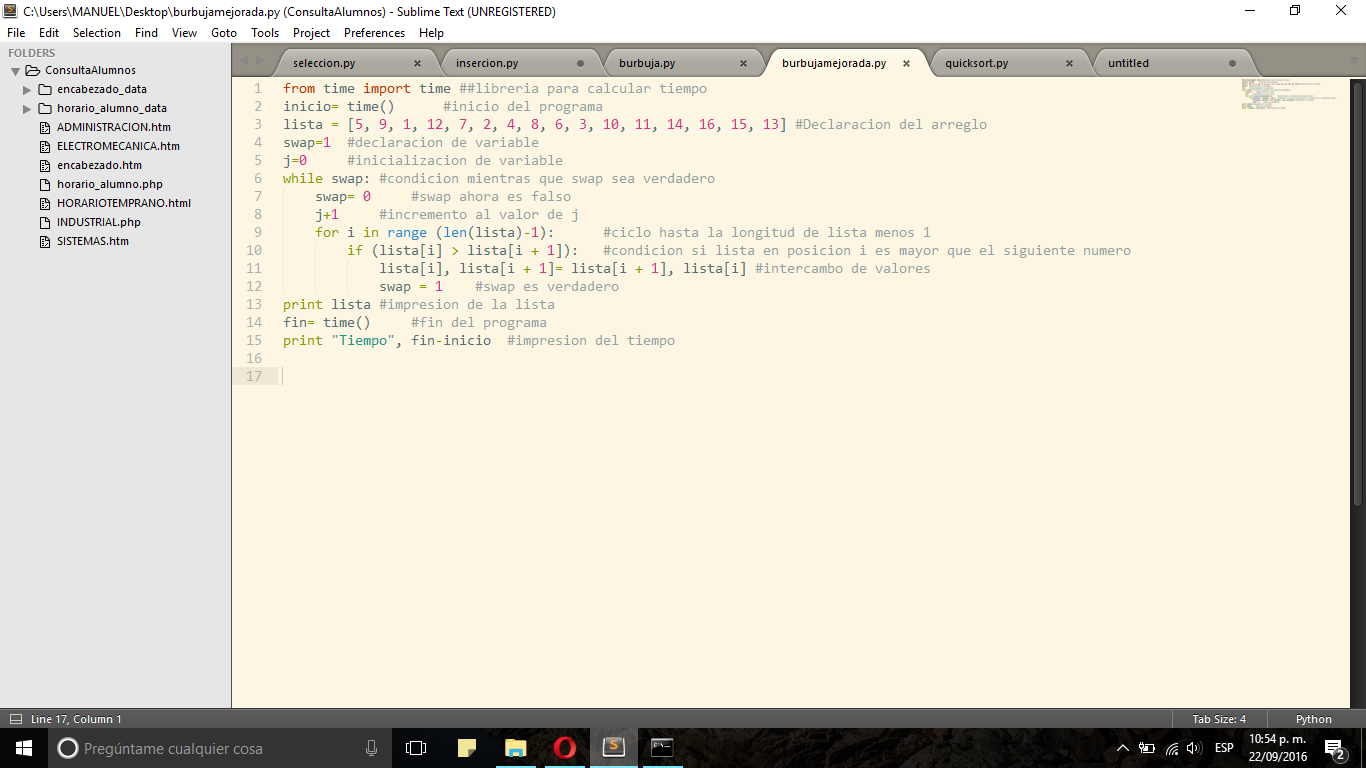
**Diagrama de flujo**



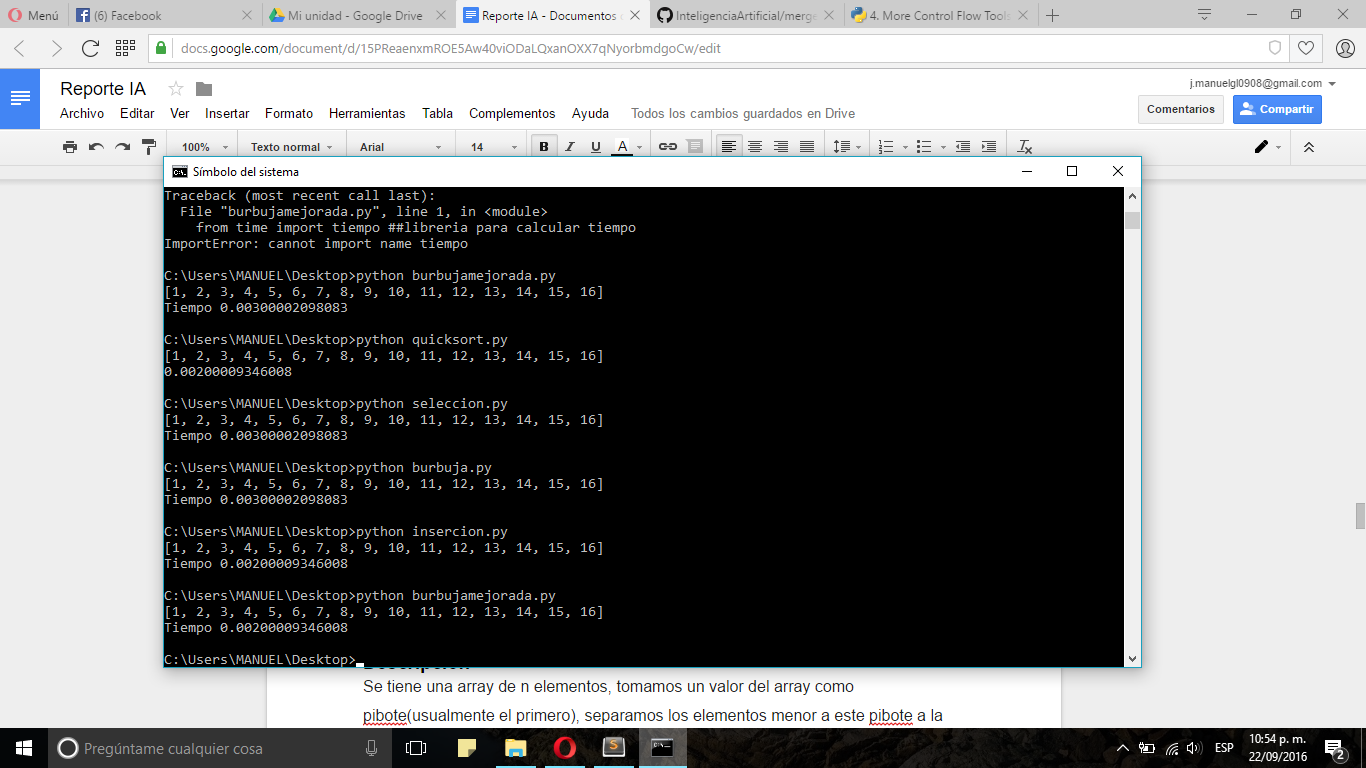
**Ejecución**



**Código en python**



**Ejecución**



Conclusión:

A diferencia de la burbuja común este método es bastante más rápido ya que ordena en una menor cantidad de iteraciones por lo cual se le pueden aplicar quizá una doble cantidad de datos más que al burbuja común realizandolo en el mismo tiempo, es una buena opción para proyectos de mediano tamaño.

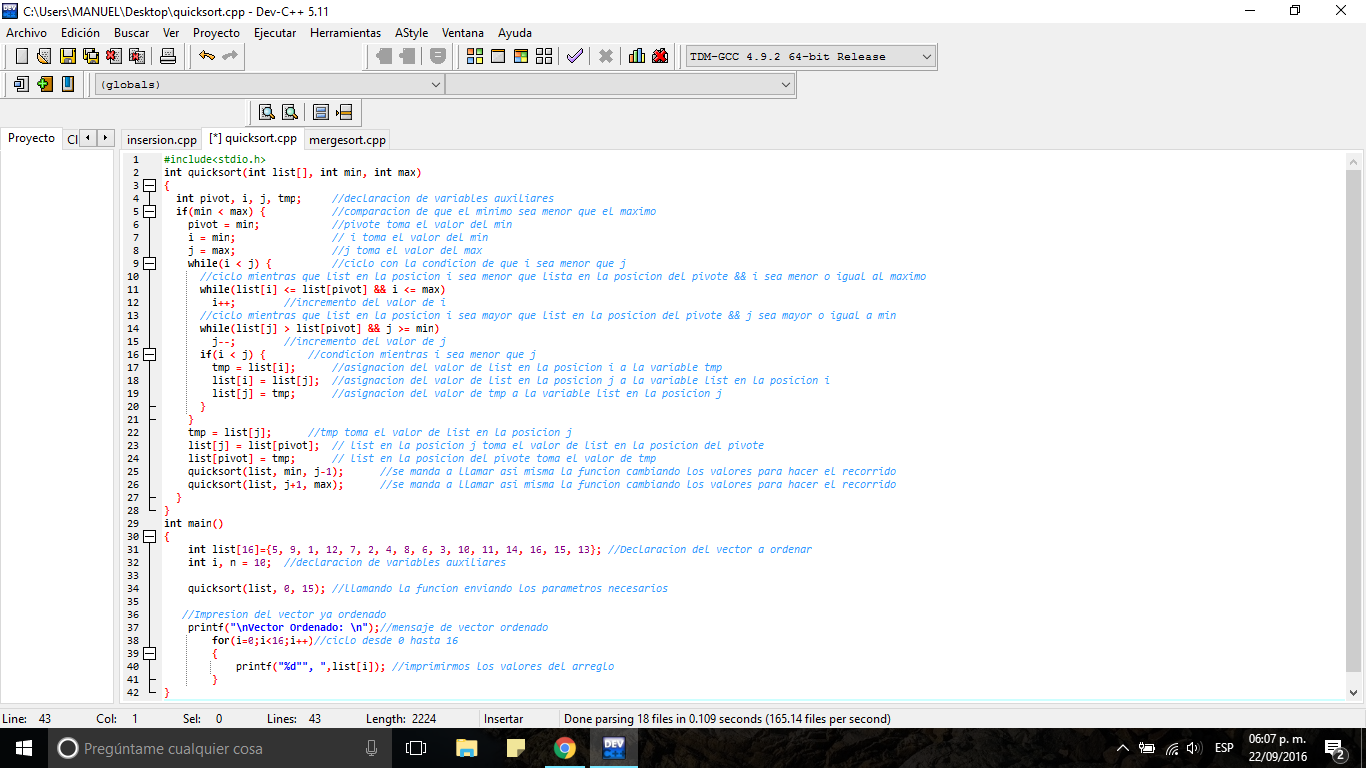
**QuickSort**

**Descripción**

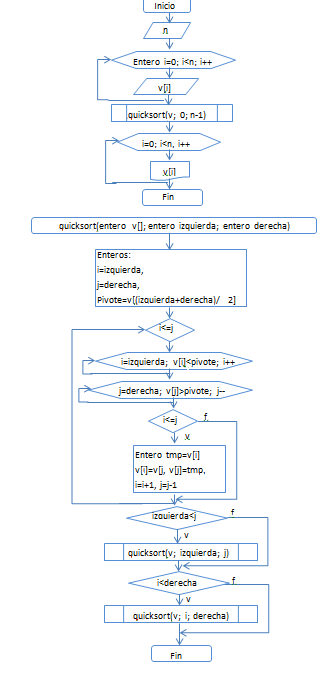
Se tiene una array de n elementos, tomamos un valor del array como pivote(usualmente el primero), separamos los elementos menor a este pivote a la izquierda y los mayores a la derecha, es decir, dividimos el array en 2 subarrays.

Con estos subarrays se repite el mismo proceso de forma recursiva hasta que estos tengan más de 1 elemento. Por lo tanto la función quicksort quedaría de la siguiente manera:

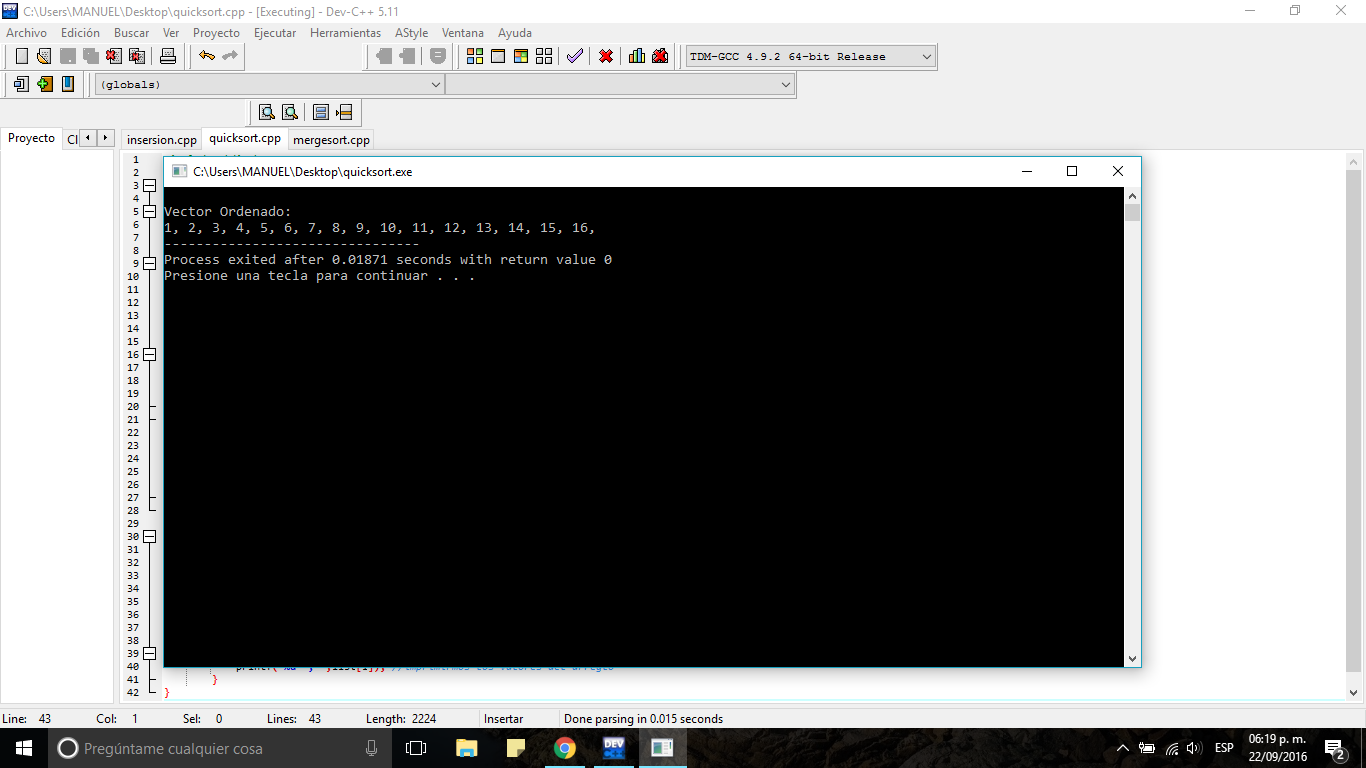
**Codigo en C**



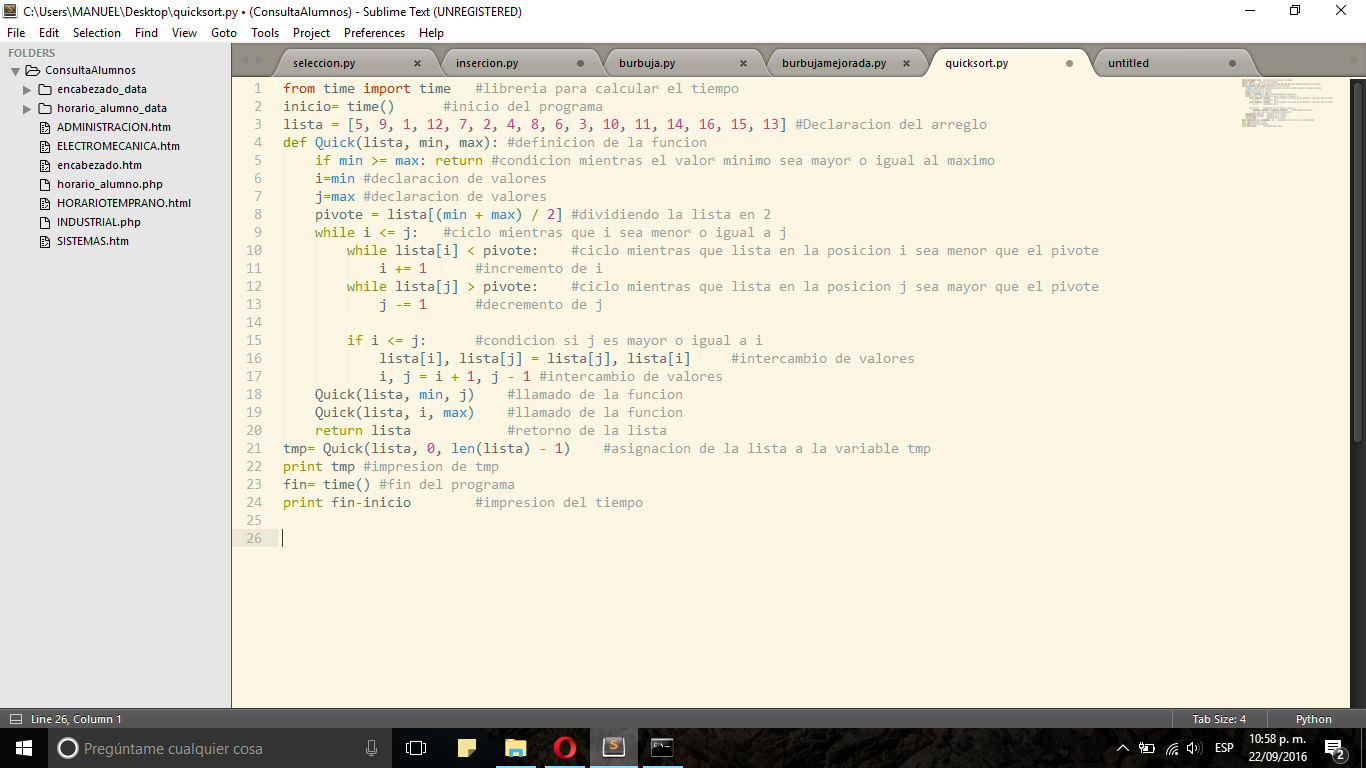
**Diagrama de Flujo**



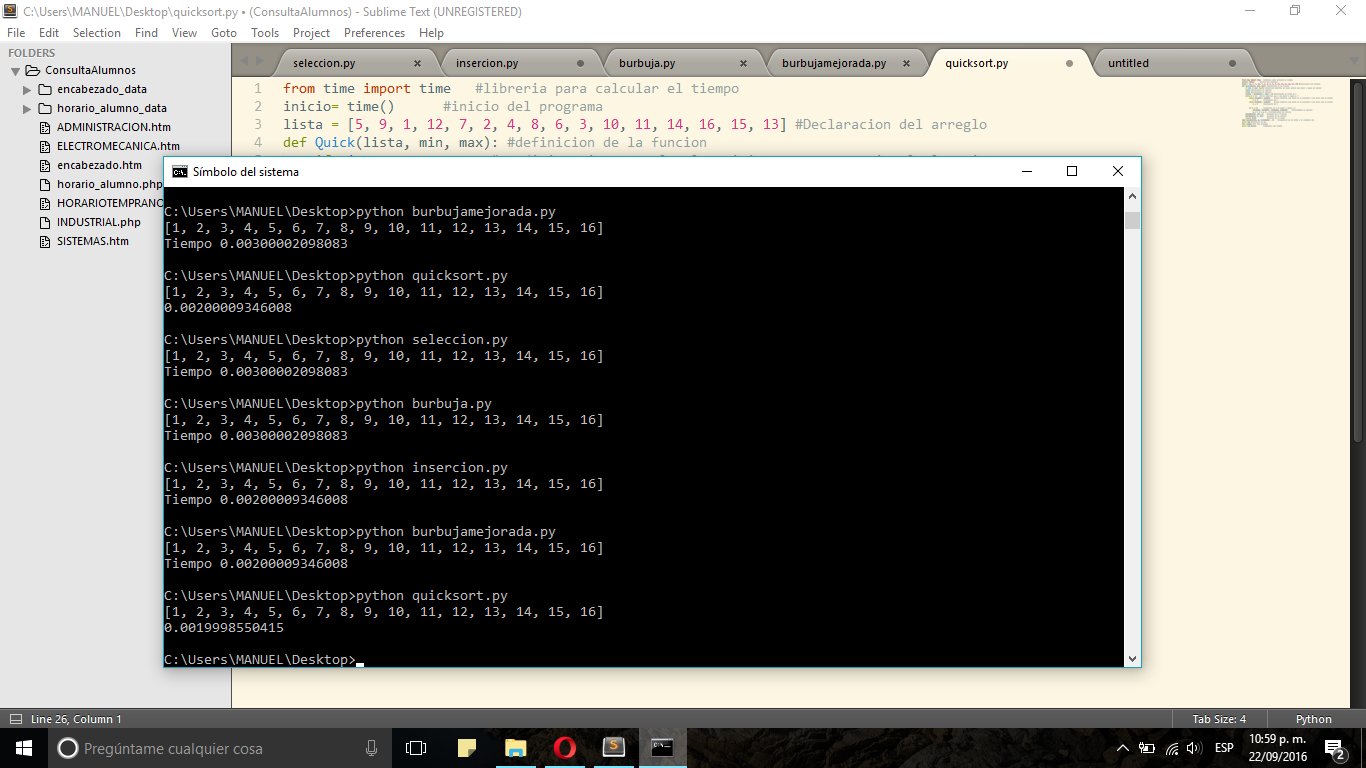
**Ejecución**



**Código en Python**



Ejecución



**Conclusión :**

Analizando quicksort en el mejor y peor caso se puede ver que es uno de los mejores métodos de ordenación, su implementación a pesar de no ser muy sencilla tampoco es complicada haciendo de este un algoritmo interesante y elegante estructura y de mucha eficiencia.

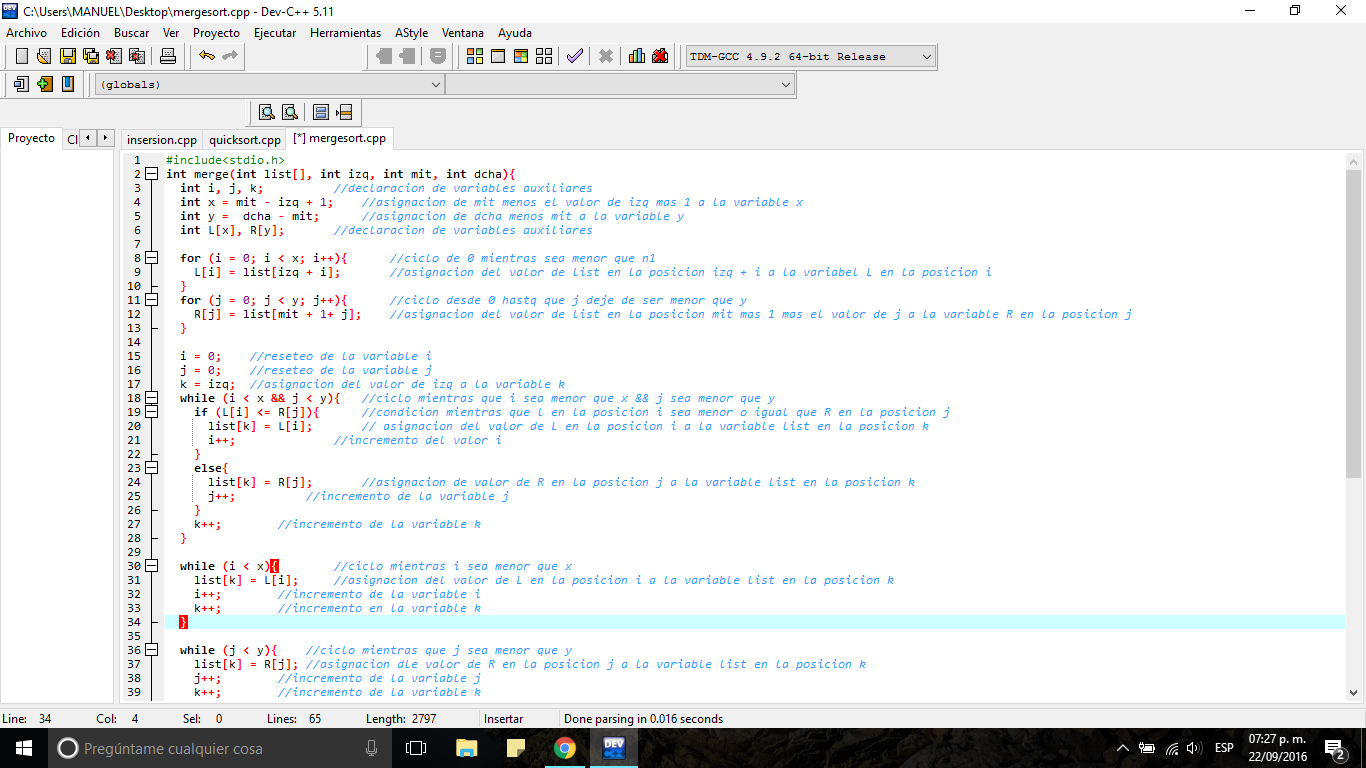
Este algoritmo de ordenación es un ejemplo claro de que el método divide y vencerás es efectivo cuando tienes cantidades grandes de datos por trabajar y necesitas ahorrar tiempo y recursos

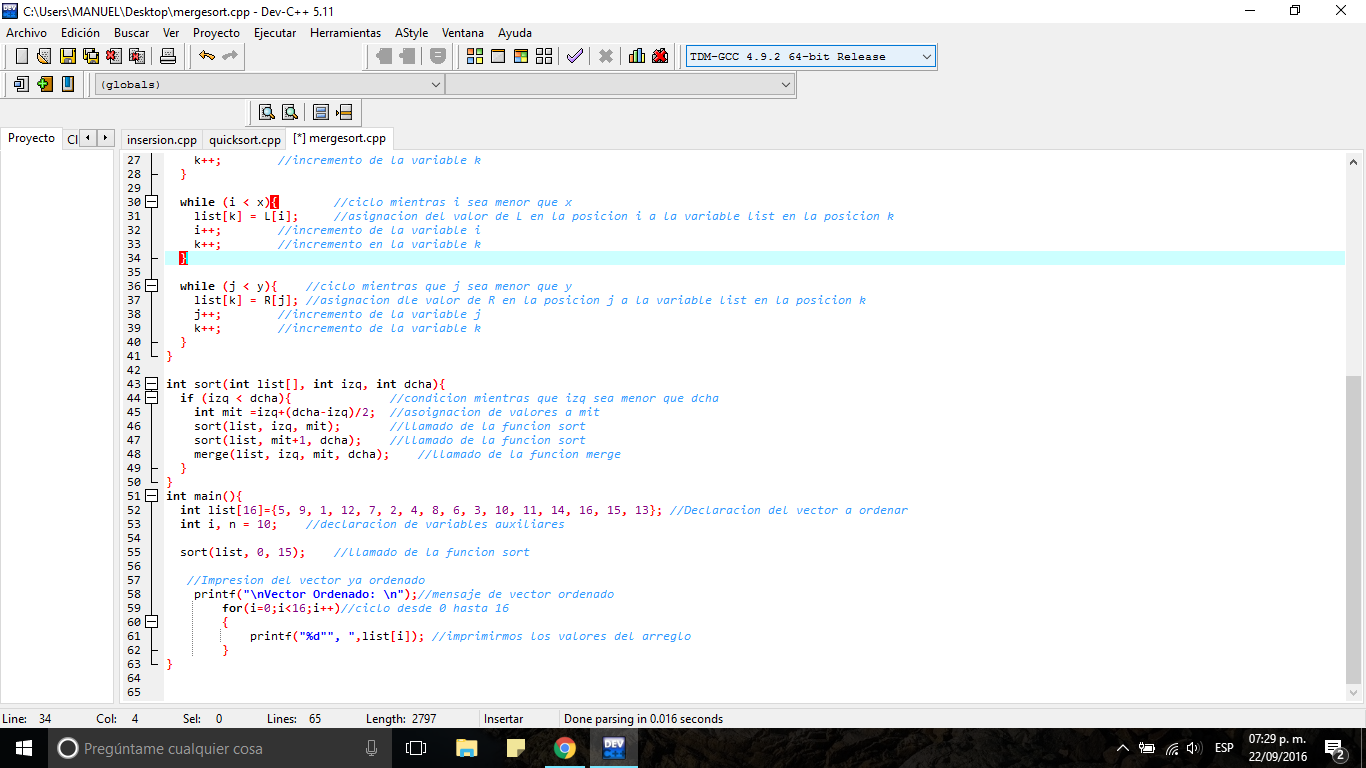
**MergeSort**

**Descripción**

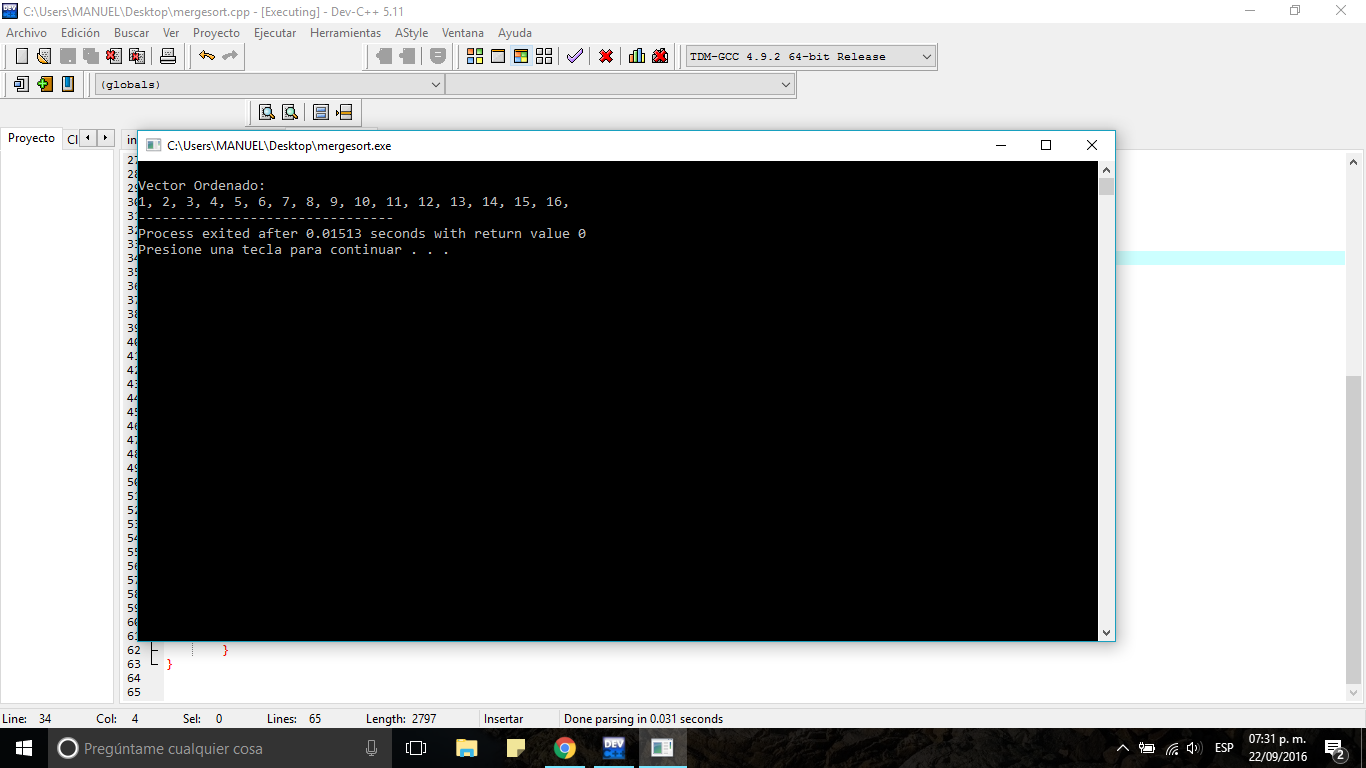
El algoritmo de Mergesort es un ejemplo clásico de algoritmo que utiliza el principio de dividir para conquistar. Si el vector tiene más de dos elementos se lo divide en dos mitades, se invoca recursivamente al algoritmo y luego se hace una intercalación de las dos mitades ordenadas.

**Código en C**





**Ejecución**



**Conclusión:**

Mergesort es una excelente opción , de hecho una de las mejores para el ordenamiento de datos ya que utiliza la técnica de “divide y vencerás”

así como implementa el uso de la recursividad haciendo efectivo y rápido rápido el ordenamiento , siendo este el más eficaz en las pruebas de rendimiento en comparación con los demás algoritmos.

**TABLA COMPARATIVA**

|  |  |  |
| --- | --- | --- |
| **Método** | **C** | **Python** |

|  |  |  |
| --- | --- | --- |
| **Metodo Insercion** | 0.01387 | 0.0020 |
| **Metodo Seleccion** | 0.01549 | 0.0030 |
| **Metodo Burbuja** | 0.1221 | 0.0030 |
| **Metodo Burbuja mejorada** | 0.01334 | 0.0020 |
| **Metodo Quicksort** | 0.01871 | 0.0019 |
| **Método MergeSort** | 0.01513 |  |