Universidad Nacional Experimental de Guayana

Vicerrectorado Académico

Coordinación General de Pregrado

Proyecto de Carrera: Ingeniería Informática

Cátedra: Estructura de Datos

Sección: 1

**Laboratorio- Métodos de Ordenamiento**

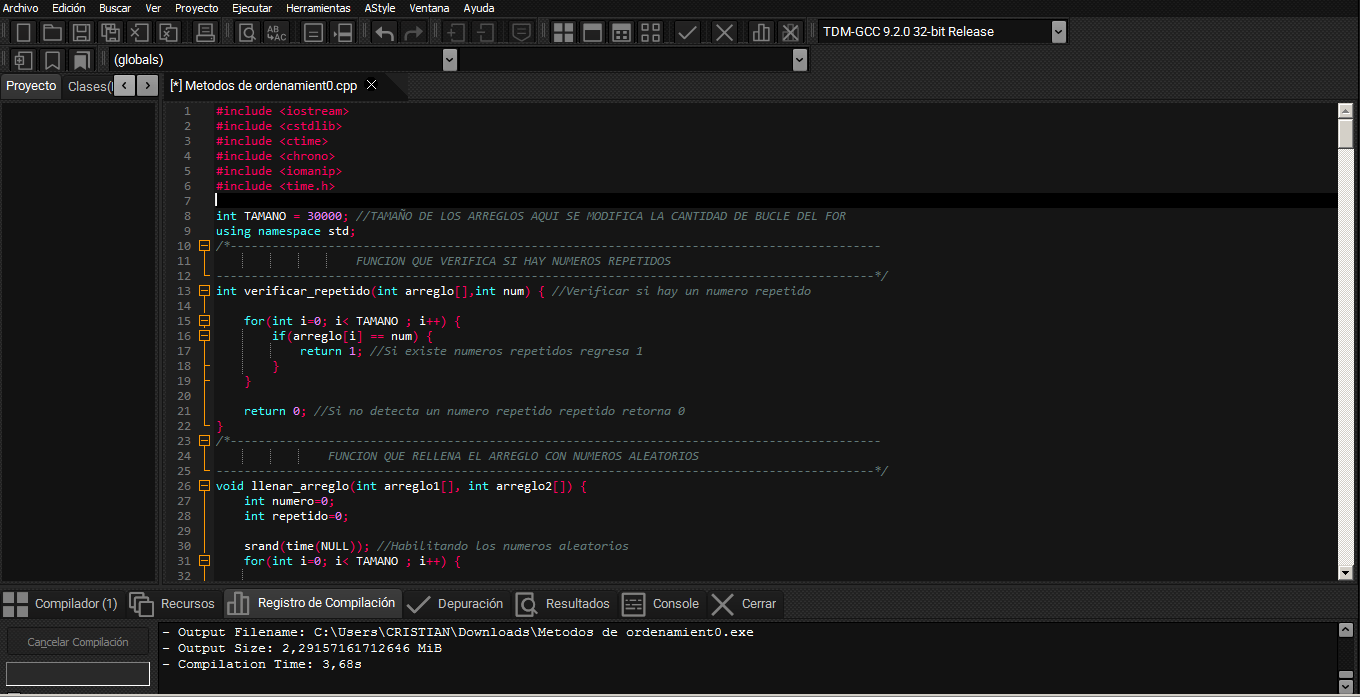
Profesor: Alumno:

Carlos Abaffy Lorenzo Parra CI: V-28727109

Ciudad Guayana, Junio 2023

**OBSERVACIONES**

* Se observó que en distintas maquinas la ejecución del programa arroja diferentes tiempos de ejecución, debido a que la gran carga de elementos ubicado en los arreglos. La velocidad para poder procesar tanta información respecto a los métodos de ordenamiento variará dependiendo del equipo utilizado, es previsible ya que el computador utiliza todos los recursos disponibles para realizar la ejecución del programa.
* Se utiliza el compilador Embarcadero Dev-C++ 6.3.
* Se compila y ejecuta con TDM-GCC 9.2.0 32-bit Release. El tipo de ejecución dependerá del arquetipo de la maquina ya que si se ejecuta una versión de 64-bit en un computador de 32-bit habrá errores de compilación.



**ANALISIS**

* **Ordenamiento de 30.000 elementos en un arreglo**

1. Arreglo desordenado: Cuando los arreglos están desordenados los métodos más eficientes son QuickSort y ShellShort con una diferencia de 0.01segundos. Los más ineficientes son el método burbuja con una diferencia de 8.60 segundo y burbuja modificado con 9.04 segundos.
2. Arreglo ordenado: Cuando los arreglos están ordenado existen 3 métodos que aproximadamente duran la misma cantidad y estos son el Ordenamiento por inserción, burbuja modificada y ShellShort.
3. Arreglo invertido: Cuando el arreglo está ordenado e invertido los mejores tiempos son ShellShort con un tiempo de 0.1 segundos, el método de ordenamiento Selección con una diferencia de 3.13 segundos y Quicksort con un tiempo de 3.31 segundos de diferencia.

* **Ordenamiento de 3000 elementos en un arreglo**

1. Arreglo desordenado: Los tiempos de cada método de ordenamiento se diferencian en promedio de 1 segundo a 2.Los que ordenan más rápido siguen siendo QuickSort,ShellShort y ahora los métodos burbuja y burbuja modificada son eficaces debido a que el tiempo de ordenamiento se redujo drásticamente.
2. Arreglo ordenado: Los métodos Insercion, Burbuja.M y Shellshort poseen una diferencia de microsegundos el cual el primer puesto lo ocupa Inserción.
3. Arreglo invertido: Cuando el arreglo está invertido el más eficiente es Shellshort con una diferencia con el método de Selección y el método de QuickSort de 0.03 segundos. Los métodos burbuja ocupan la última posición con una diferencia aproximada de 0.06 segundos.

Se debe resaltar, el tiempo de ordenamiento cuando esta desordenado, ordenado he invertido dependerá de los elementos que existan en cada ejecución del programa ya que no serán los mismos arreglos y en la maquina donde se ejecute el programa ya que habrá diferencias de tiempos drásticos.

**CONCLUSIONES**

La eficiencia de los métodos de ordenamiento dependerá del tamaño del arreglo. Un método de ordenamiento será mejor que otro en la situación en que sea usado, es decir, mientras más grande sea el tamaño del arreglo el tiempo de ordenamiento será mayor, Los mejores métodos cuando están desordenados son QuickSort y ShellShort.

Cuando el arreglo está ordenado, los métodos de ordenamiento por inserción, burbuja modificada y ShellSort son los más eficientes. Sin embargo, sus diferencias de tiempo de ejecución son mínimas y pueden variar por el los elementos existentes en el arreglo y el tamaño del mismo.

En el caso de arreglos invertidos, los métodos más eficientes son ShellSort, Selección y QuickSort. Estos métodos muestran tiempos de ejecución más bajos en comparación con los métodos de burbuja.

Es importante tener en cuenta que el tamaño del arreglo también influye en los tiempos de ejecución. A medida que se aumenta el tamaño del arreglo, los métodos más eficientes siguen siendo QuickSort y ShellSort, pero los métodos de burbuja y burbuja modificada también se vuelven más eficientes en comparación con el ordenamiento de arreglos más pequeños.