**Universidad Peruana de Ciencias Aplicadas**

**UPC**

“Año del diálogo y la reconciliación nacional”



Trabajo Final

Curso: Algoritmos y Estructura de Datos

Sección: SW31

Alumnos:

Rodrigo Toconas, ……..

Ricardo More, u201420831

Monterrico, junio 2019

Índice:

[Introducción 3](#_Toc12515002)

[Problema 4](#_Toc12515003)

[Objetivos 4](#_Toc12515004)

[Marco Conceptual 4](#_Toc12515005)

[Diagrama de Clases 6](#_Toc12515006)

[Definir Requisitos 6](#_Toc12515007)

[Ejecución 7](#_Toc12515008)

[Implementación de Clases 7](#_Toc12515009)

[Conclusiones 14](#_Toc12515010)

[Referencias 14](#_Toc12515011)

# Introducción

En este informe presentaremos el desarrollo de la estructura conocida como Dataframe y sus diferentes manipulaciones, en el lenguaje de programación C++ , el cual será presentado como el Informe del Trabajo Final del curso de Algoritmos y Estructura de Datos.

El trabajo está pensado para crear una biblioteca accesible para cualquier usuario que necesite manejar las estructuras Dataframe. En este informe hemos empleado el desarrollo de la investigación de un Dataframe con sus respectivas manipulaciones y aplicaciones.

# Problema

El problema es que no se sabia que herramientas usar para crear la Biblioteca en C++ para el manejo de estructuras DataFrame

# Objetivos

Investigar sobre diferentes herramientas y bibliotecas para poder implementar la Biblioteca en C++ para el manejo de Dataframes

# Marco Conceptual

Para la creación de esta biblioteca hemos usado bibliotecas alternas como lo son : vector,String, fstream y structuras templates.

VECTOR.h

Los vectores son contenedores de secuencia que representan matrices que pueden cambiar de tamaño.   
Al igual que los arrays, los vectores utilizan ubicaciones de almacenamiento contiguas para sus elementos, lo que significa que también se puede acceder a sus elementos mediante el uso de desplazamientos en punteros regulares a sus elementos, y con la misma eficacia que en los arreglos. Pero, a diferencia de los arreglos, su tamaño puede cambiar dinámicamente, ya que su almacenamiento se maneja automáticamente por el contenedor.   
Internamente, los vectores utilizan una matriz asignada dinámicamente para almacenar sus elementos. Es posible que sea necesario reasignar esta matriz para que aumente su tamaño cuando se insertan nuevos elementos, lo que implica asignar una nueva matriz y mover todos los elementos a ella. Esta es una tarea relativamente costosa en términos de tiempo de procesamiento y, por lo tanto, los vectores no se reasignan cada vez que se agrega un elemento al contenedor.  
Sus funciones miembro mas representativos son.

* Begin
* End
* Size
* Max\_size
* Resize
* Capacity
* Empty
* Reserve
* At
* Front
* Back
* Data
* Push\_back
* Pop\_back
* Insert
* Erase
* Swap
* clear

<String>

Esta biblioteca lo que hace es introducer en el Proyecto los tipo string, caracteres y algunas funciones de conversion. Entre sus funciones miembros mas representativos estan:

* stoi
* stol
* stoul
* stoll
* stoull
* stof
* stod
* stold
* atoi

<fstream>

Esta biblioteca lo que hace es introducer en el Proyecto los tipo string, caracteres y algunas funciones de conversion Esta clase deriva del <istream> y <ostream> . Entre sus funciones mas representativas estan:

* getchar
* getline
* is\_open
* close

Templates

Los templates son caracteres potentes de C++ que le permiten escribir progreamas genéricos. Con los templates puedes crear una sola function o clase y poder trabajarla en diferentes tipos de datos, utilizando estos como plantilla.   
Los template se pueden usar de dos maneras diferentes: Templates de funciones y templates de clases  
Templates de funciones: Estos funcionan de forma similar a una funcion normal, con una pequeña diferencia.  
Un template de una función puede hacer que esta trabaje con diferentes tipos de datos a la vez, algo que si no usaras templates no podrías lograr y tendrías que estar reescribiendo código para cada tipo de dato.

**template** <**class** T>

T someFunction(T arg)

{

... .. ...

}

Templates de Clase: Al igual que las de función , sirven para crear plantilals de una clase para realizar operaciones geneéricas. Ya que, a vecs se necesta una implementación de clases que sean iguales para diferentestipos de datos, pues usamos los templates y asi podremos reusar el código para diferentes tipos de datos.

**template** <**class** T>

class className

{

... .. ...

public:

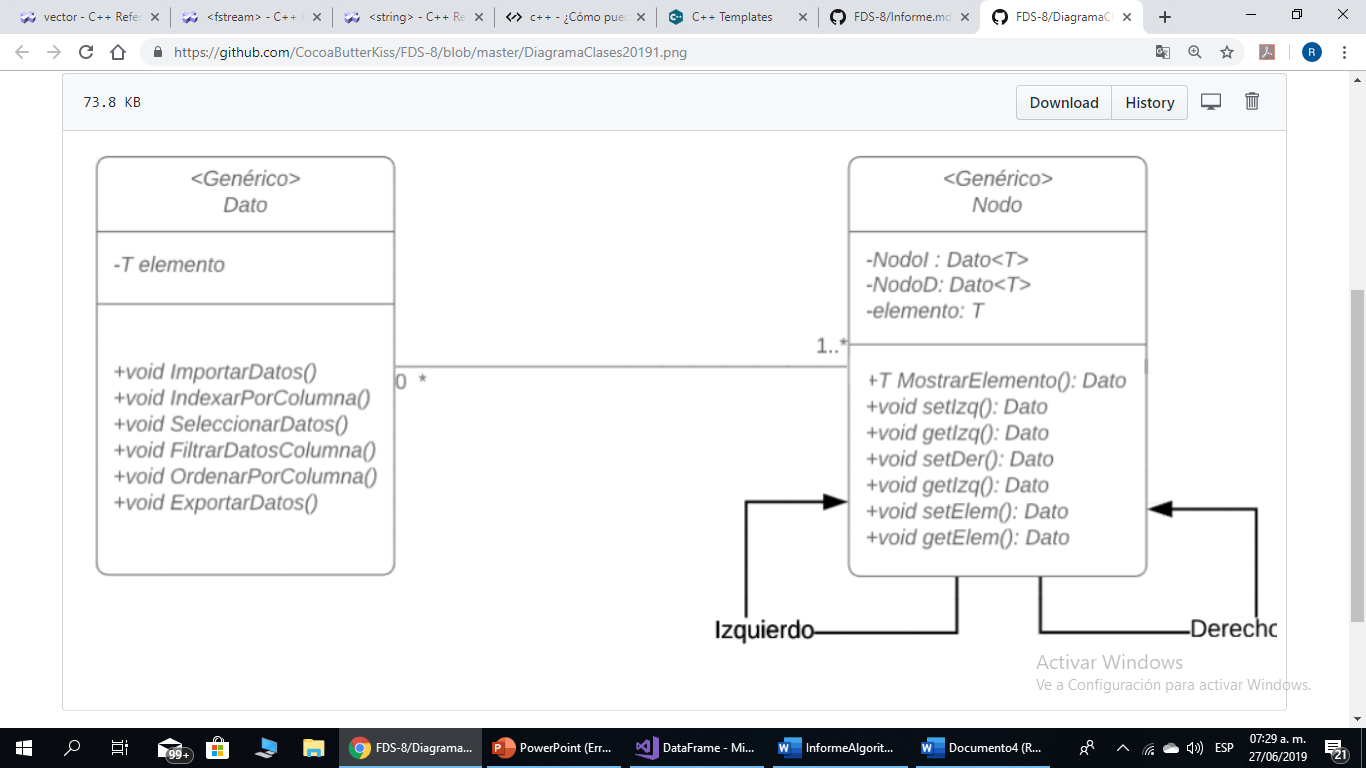
T var;

T someOperation(T arg);

... .. ...

};

# Diagrama de Clases



# Definir Requisitos

Requisito 01:

Se necesita de una base de datos en forma de fichero.

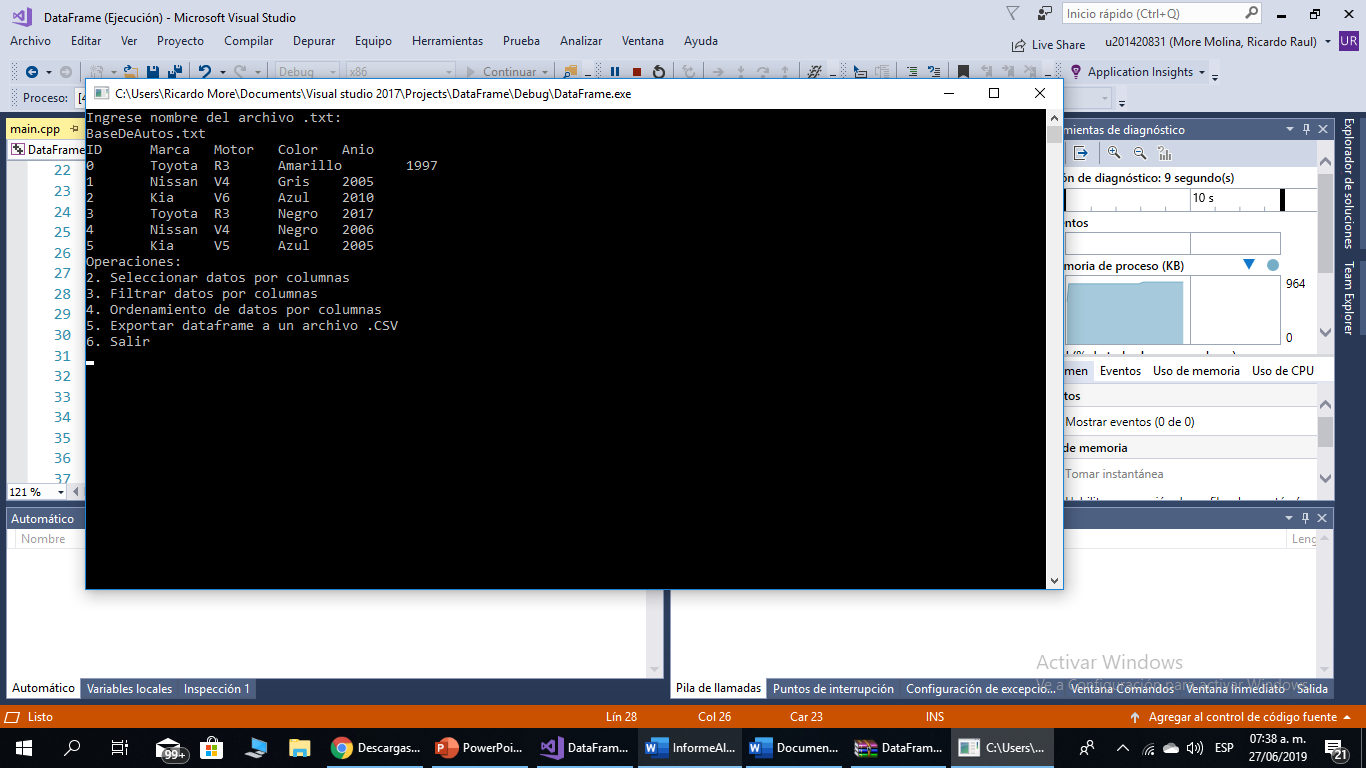
Requisito 02:

Los ficheros de entrada deben ser csv y terminar cada línea en un punto(.) .

Requisito 03:

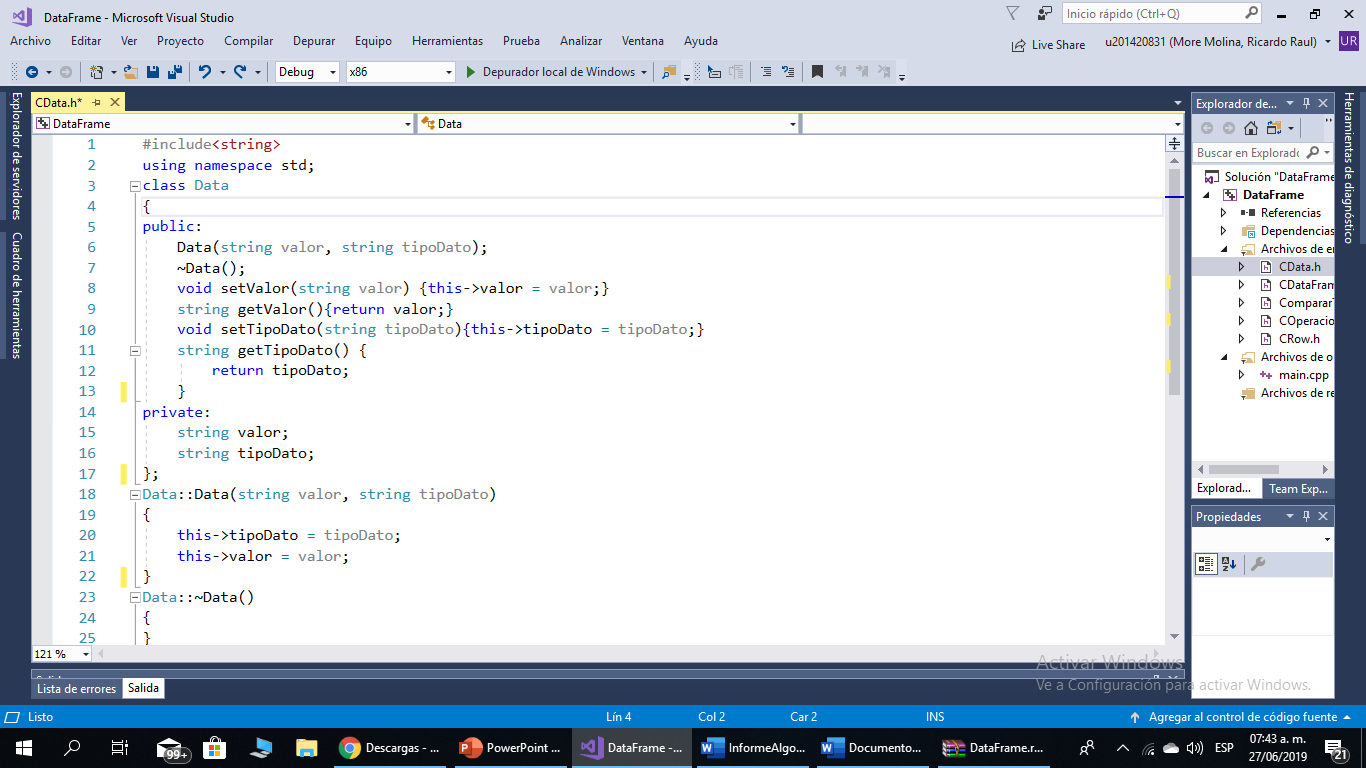
Los datos deben estar ordenados en forma horizontal.

# Ejecución

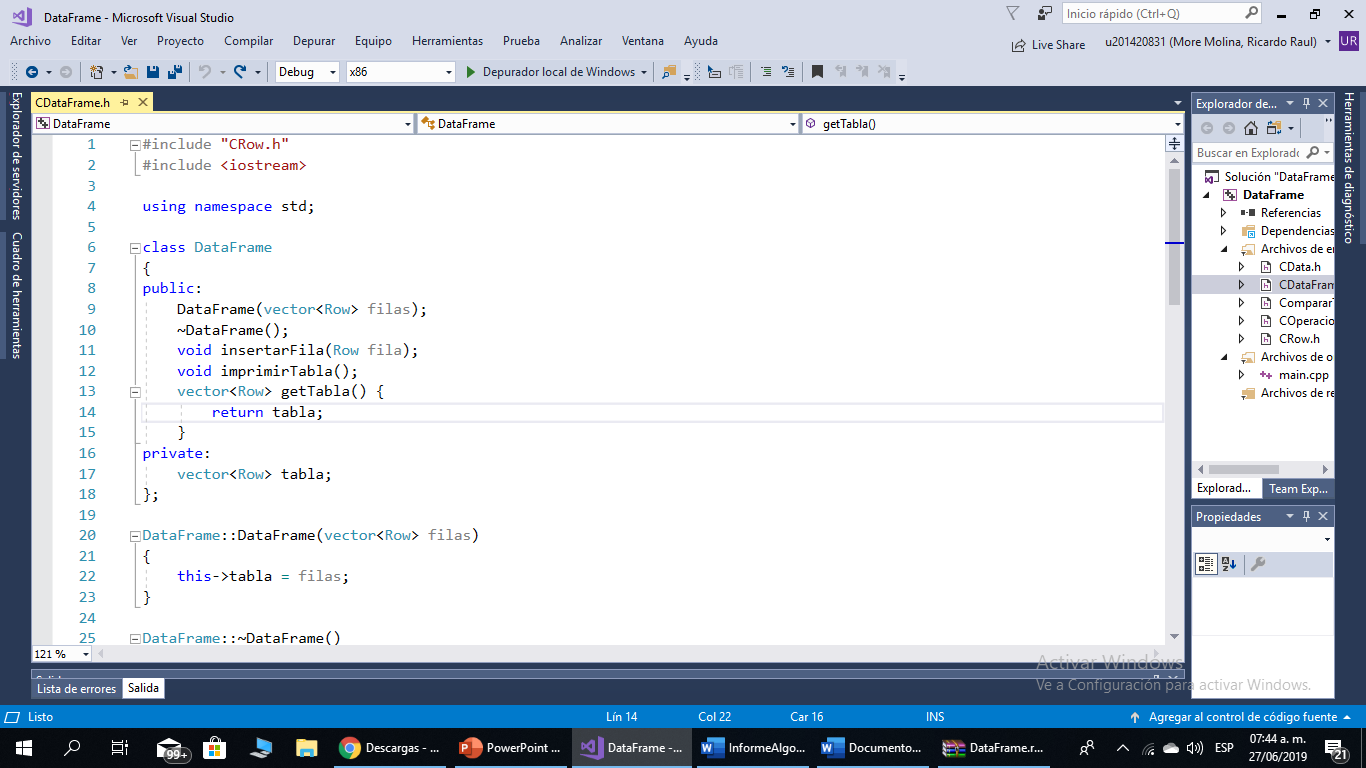


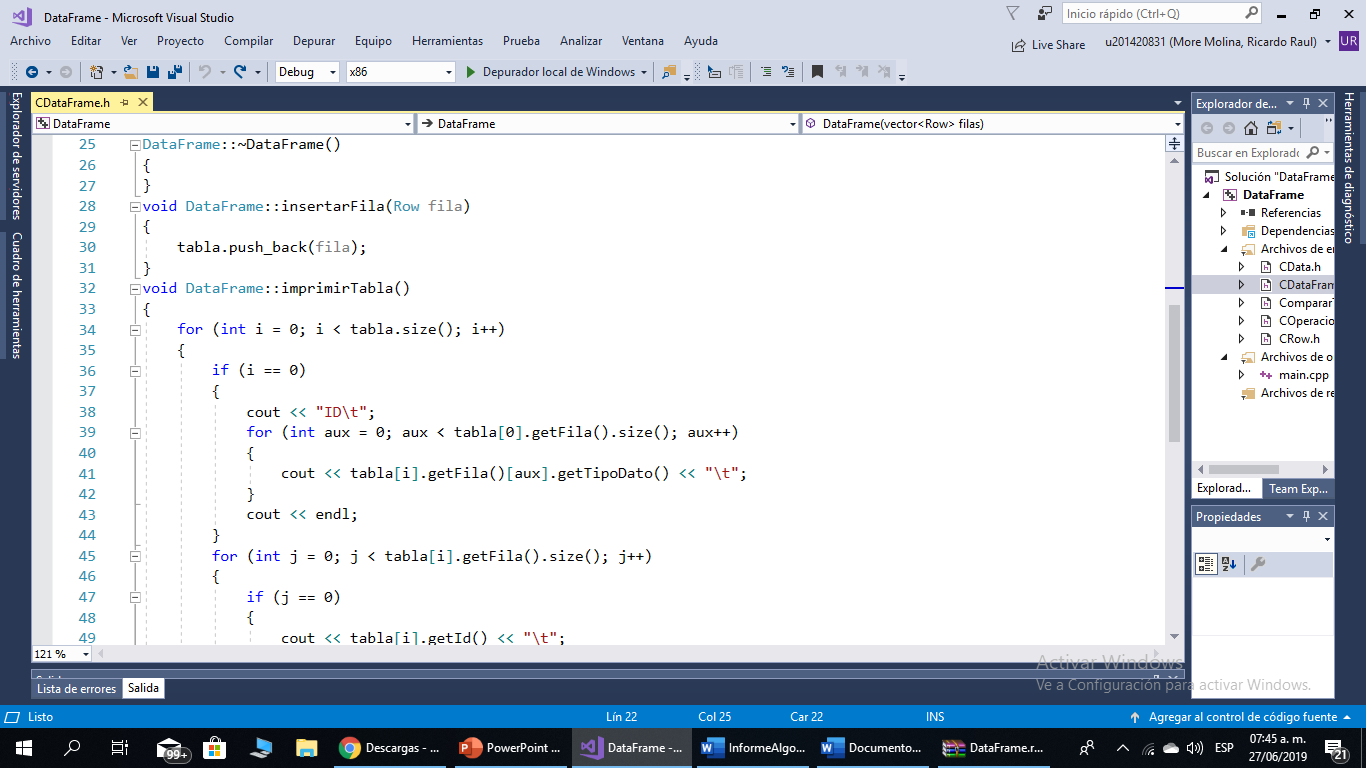
# Implementación de Clases

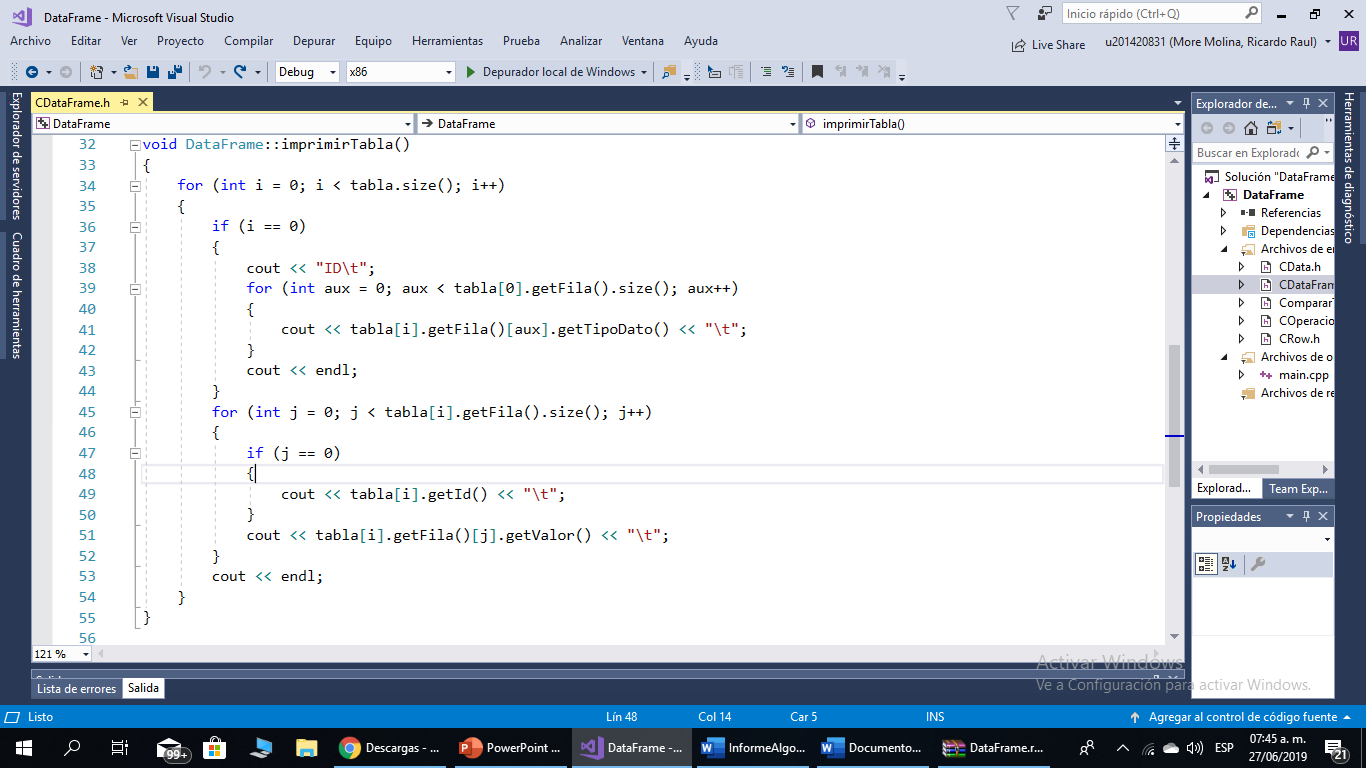
CDATA



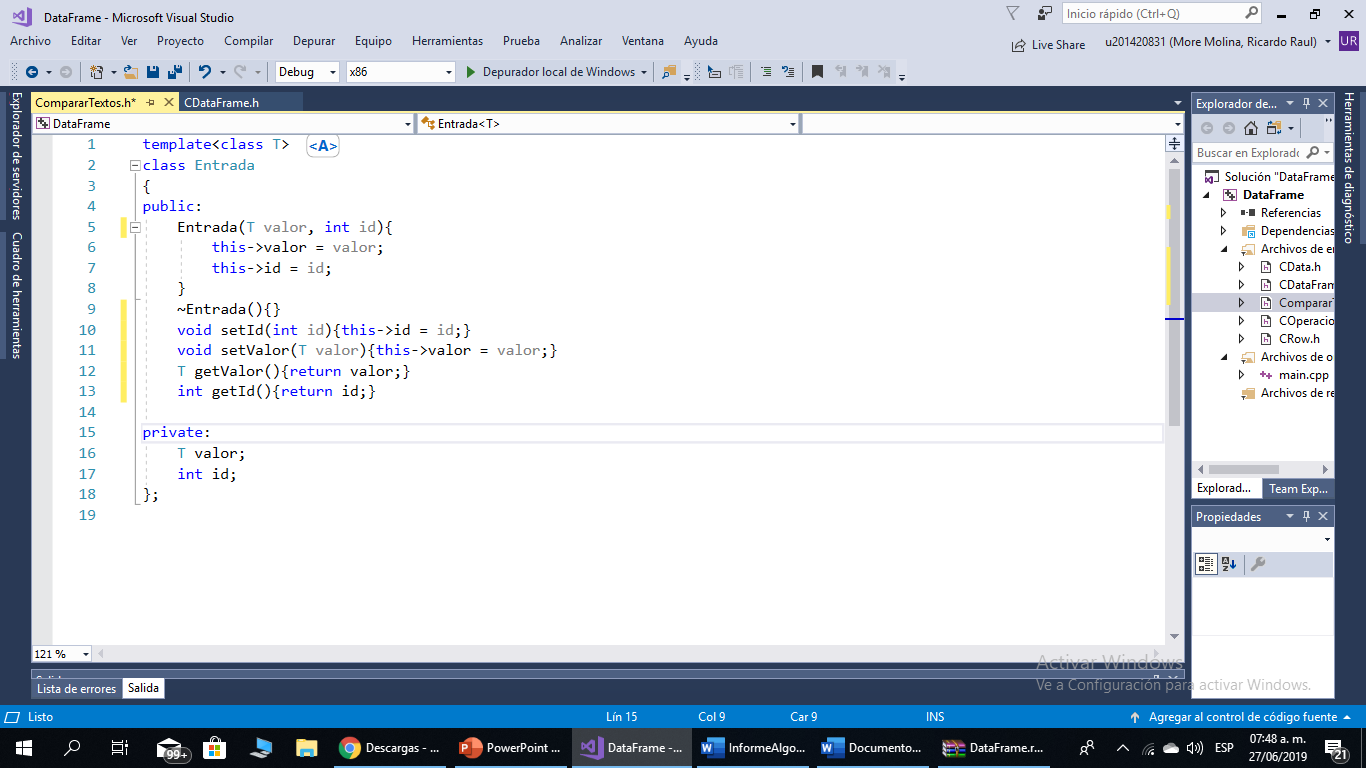
CDATAFRAME



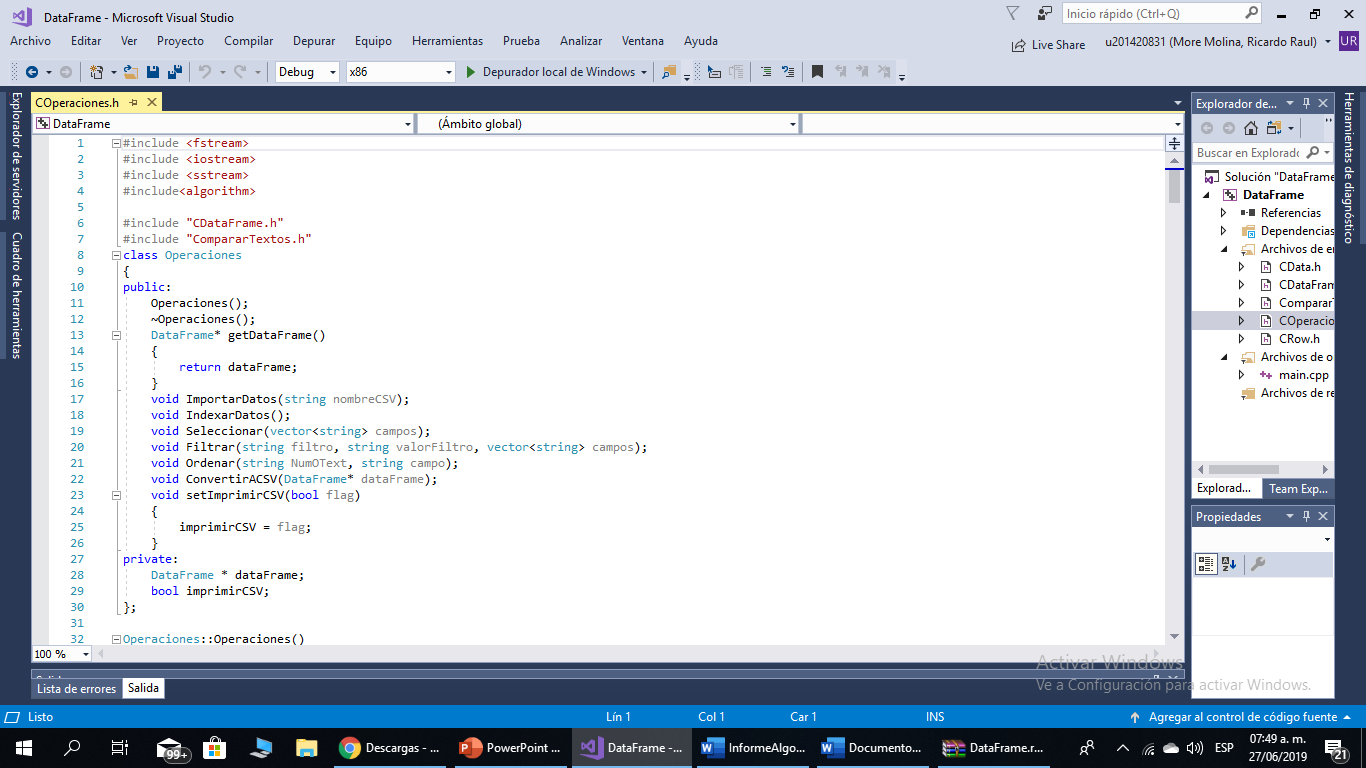


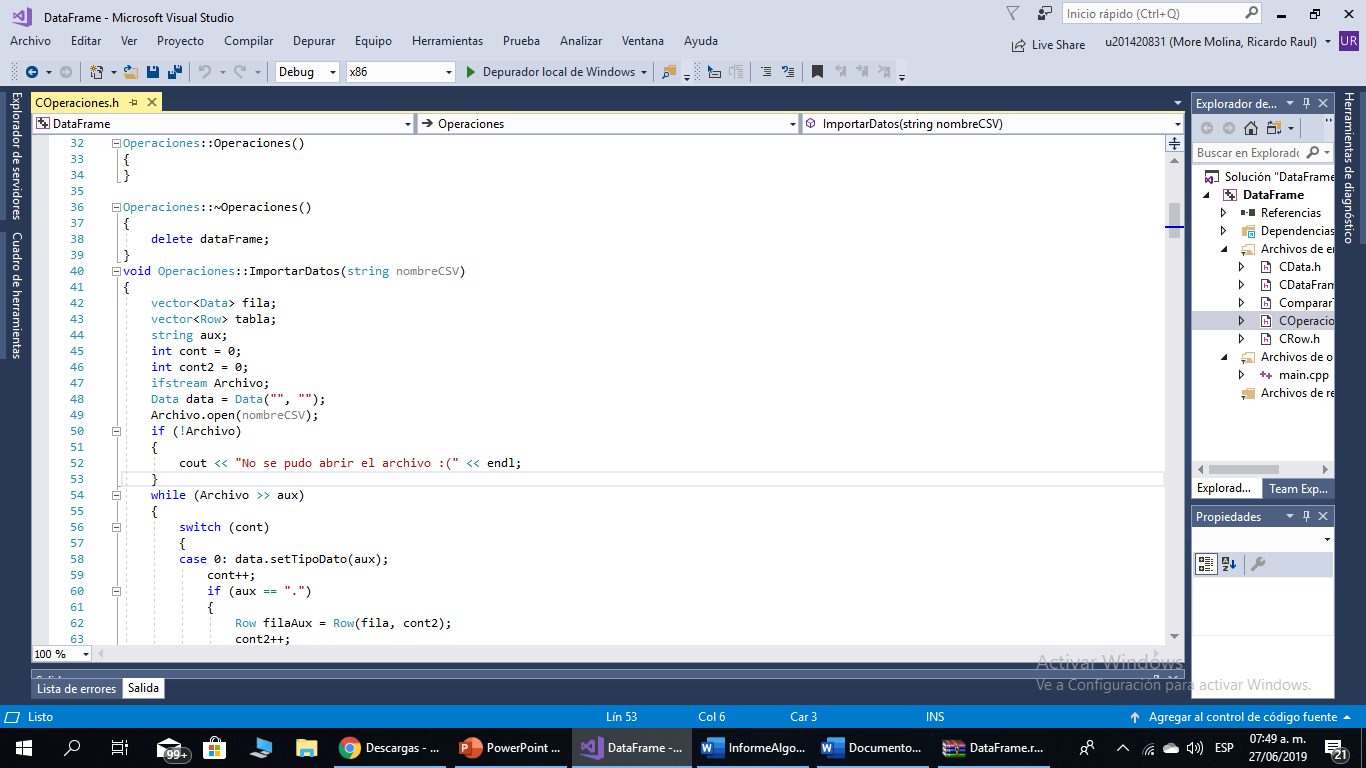


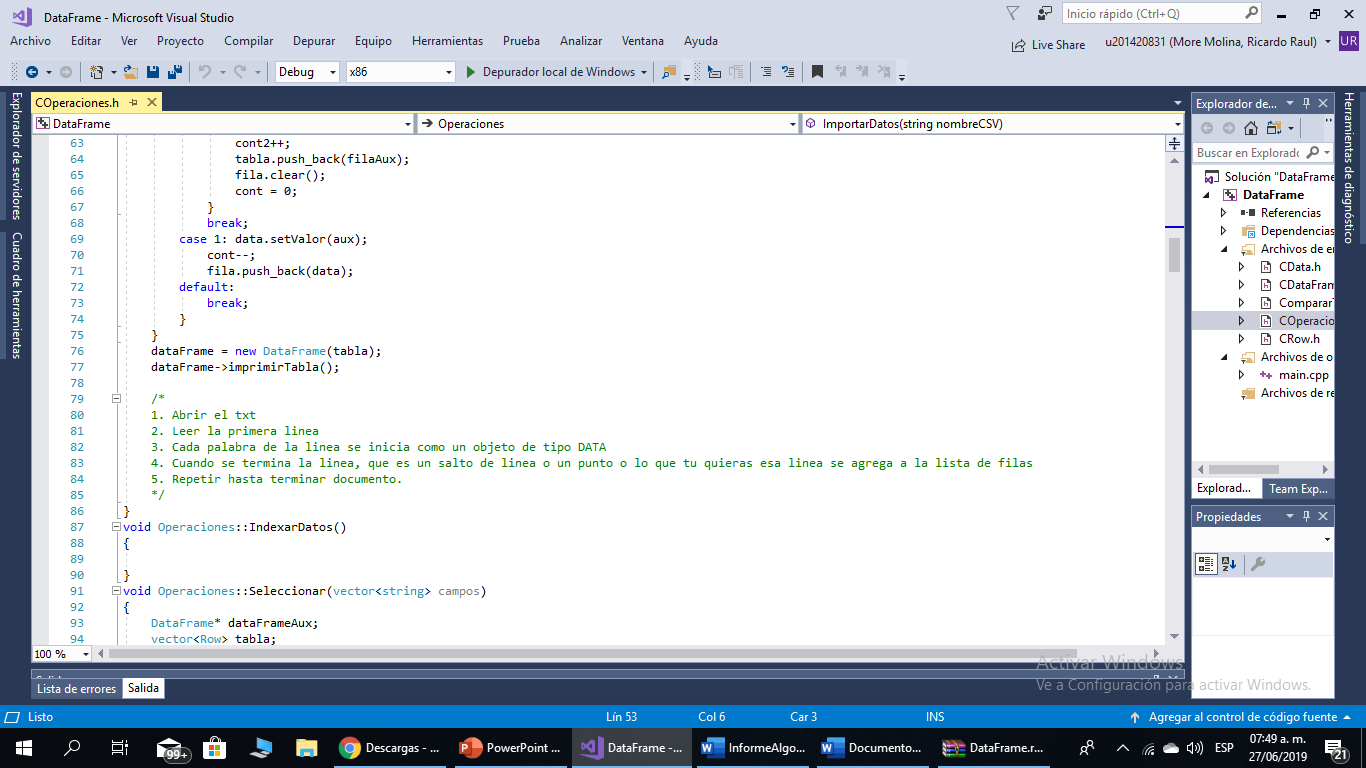
CCOMPARARTEXTOS

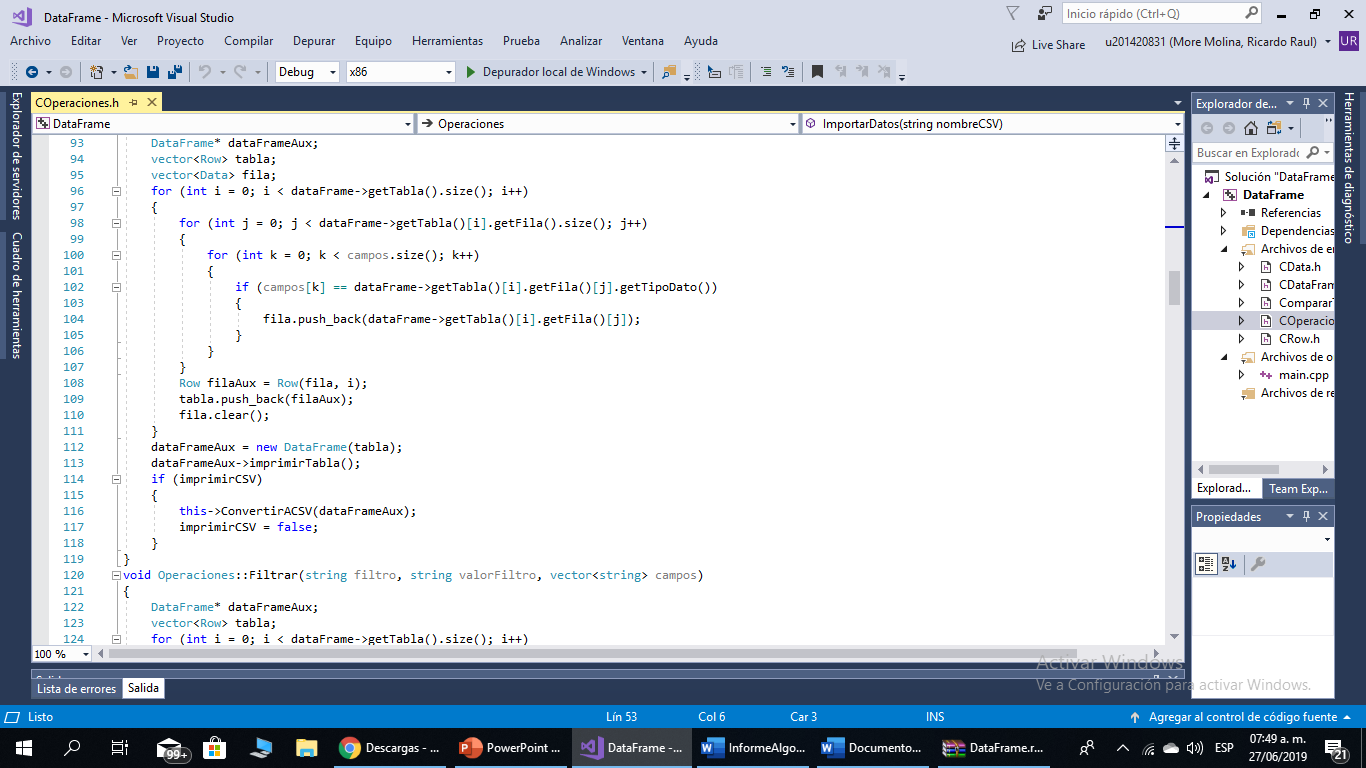


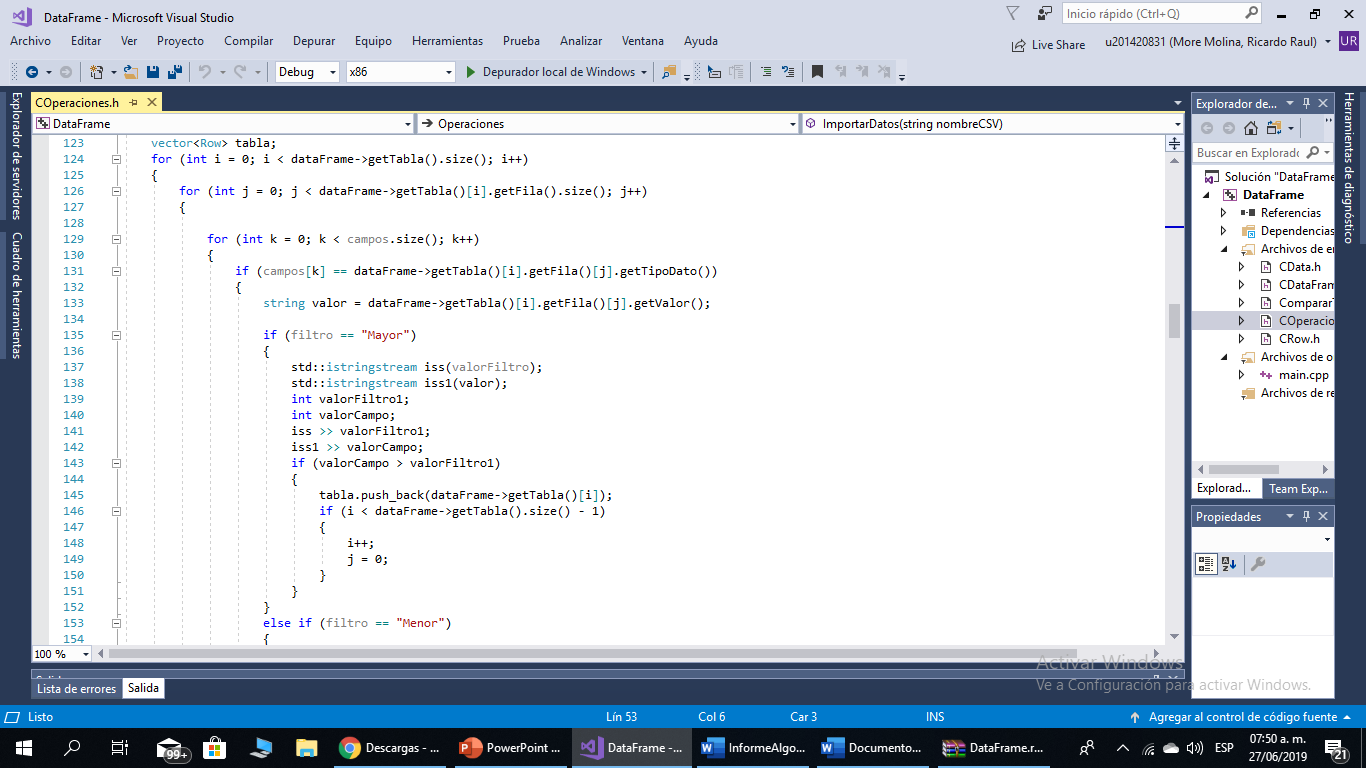
COPERACIONES

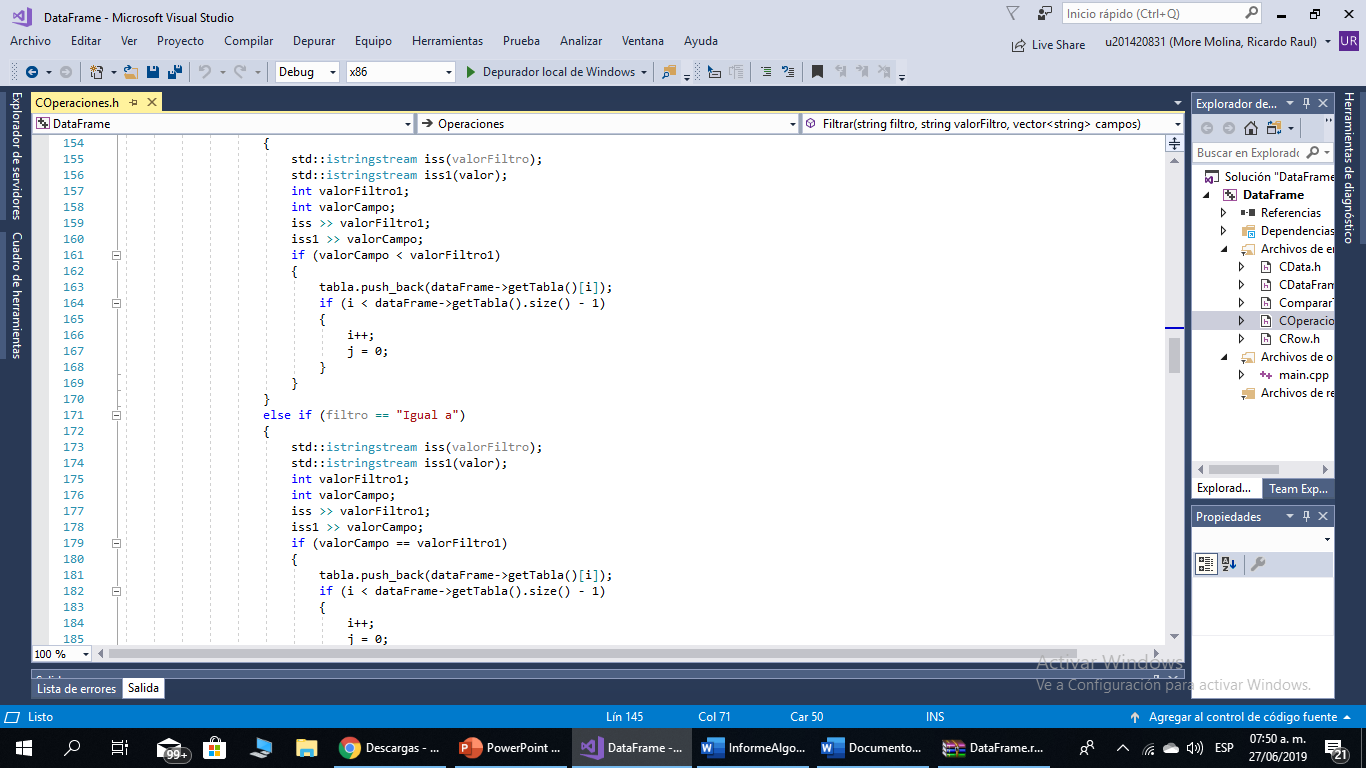


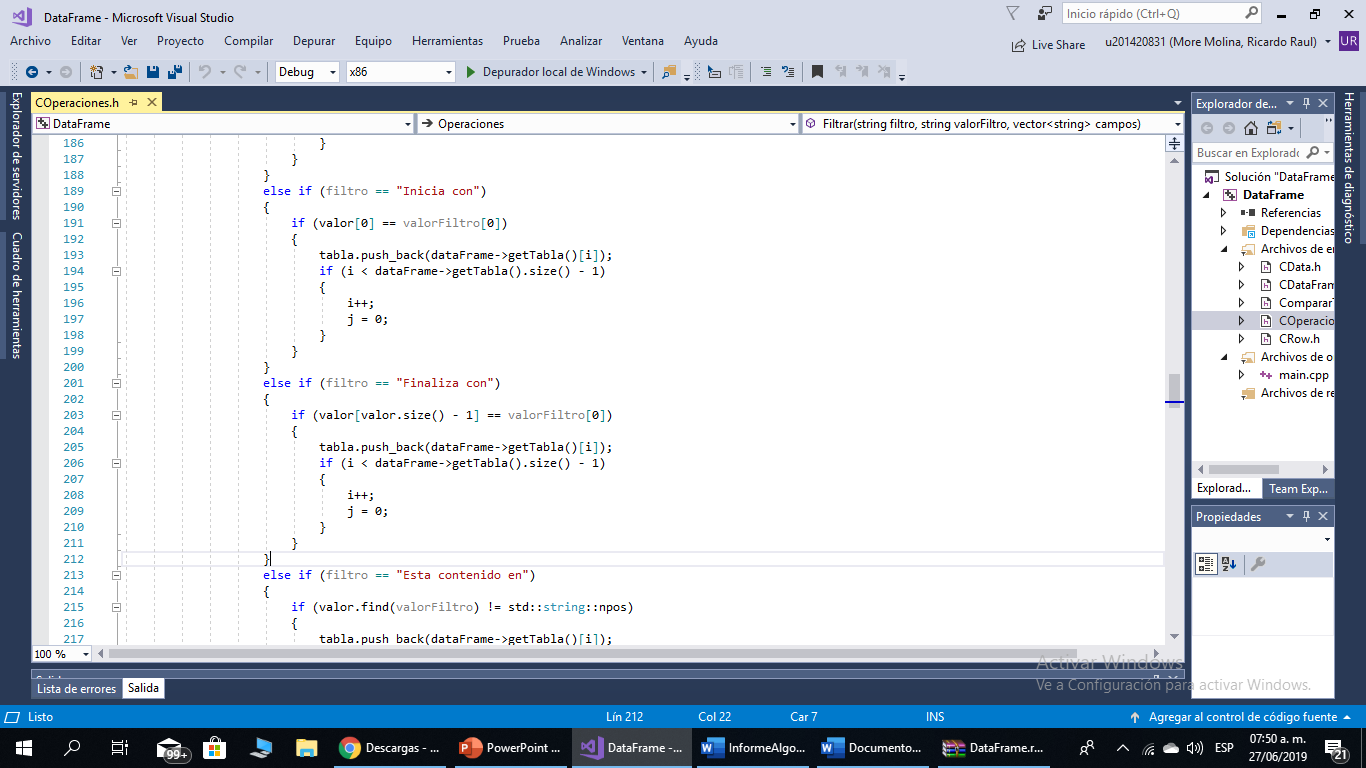


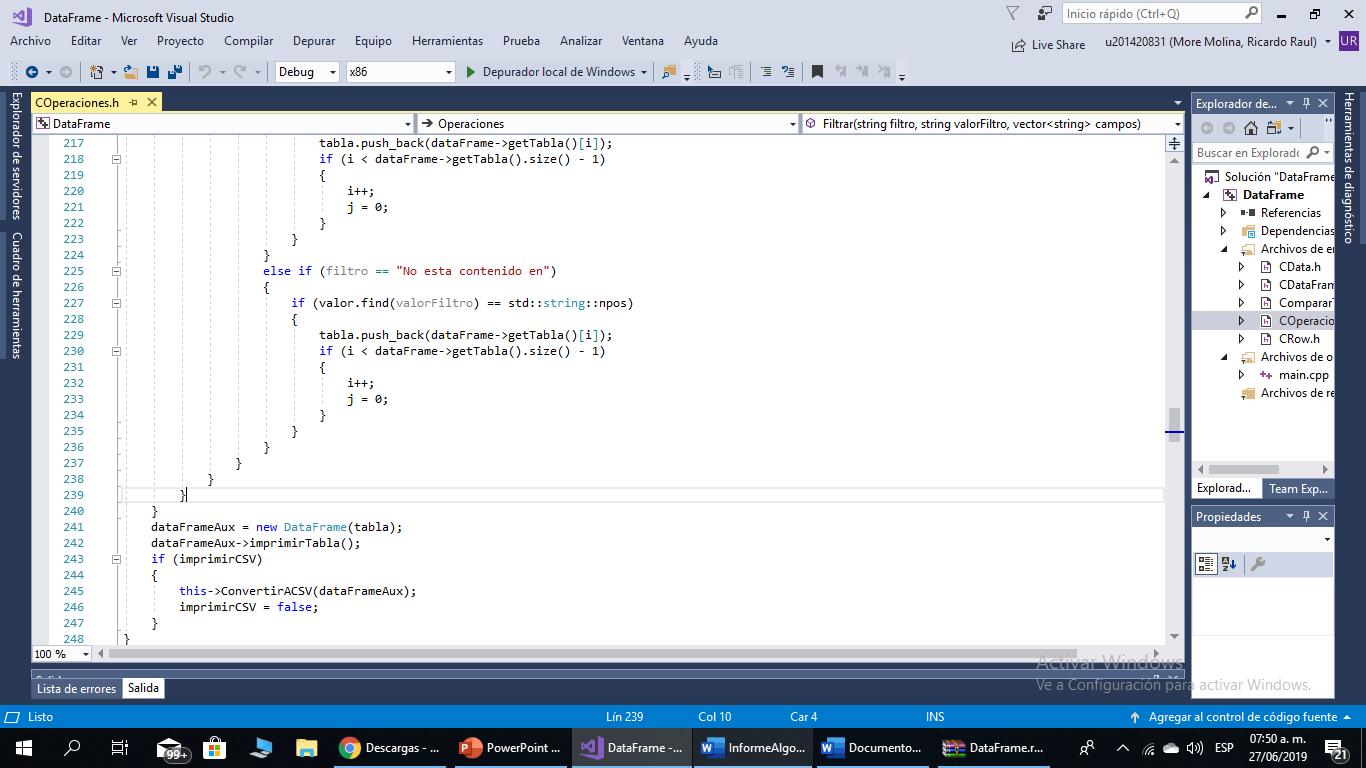


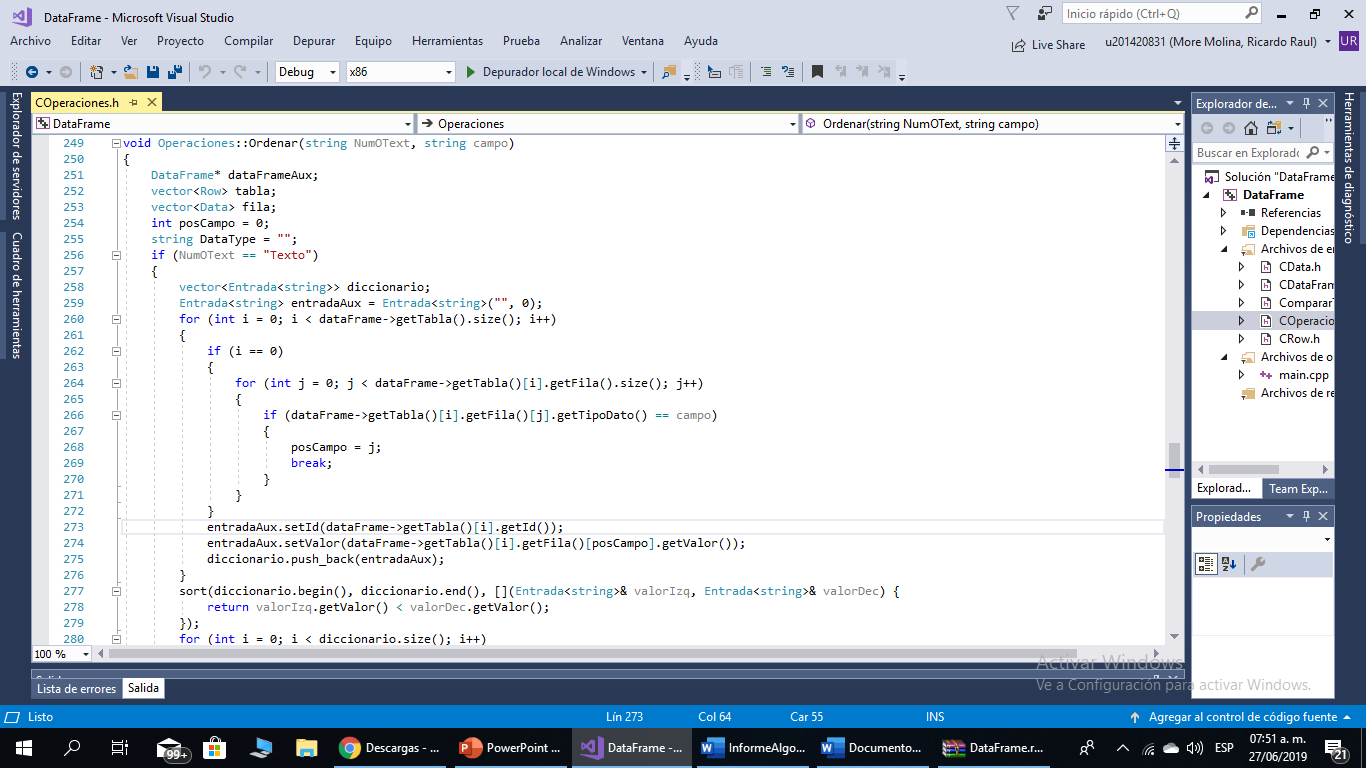


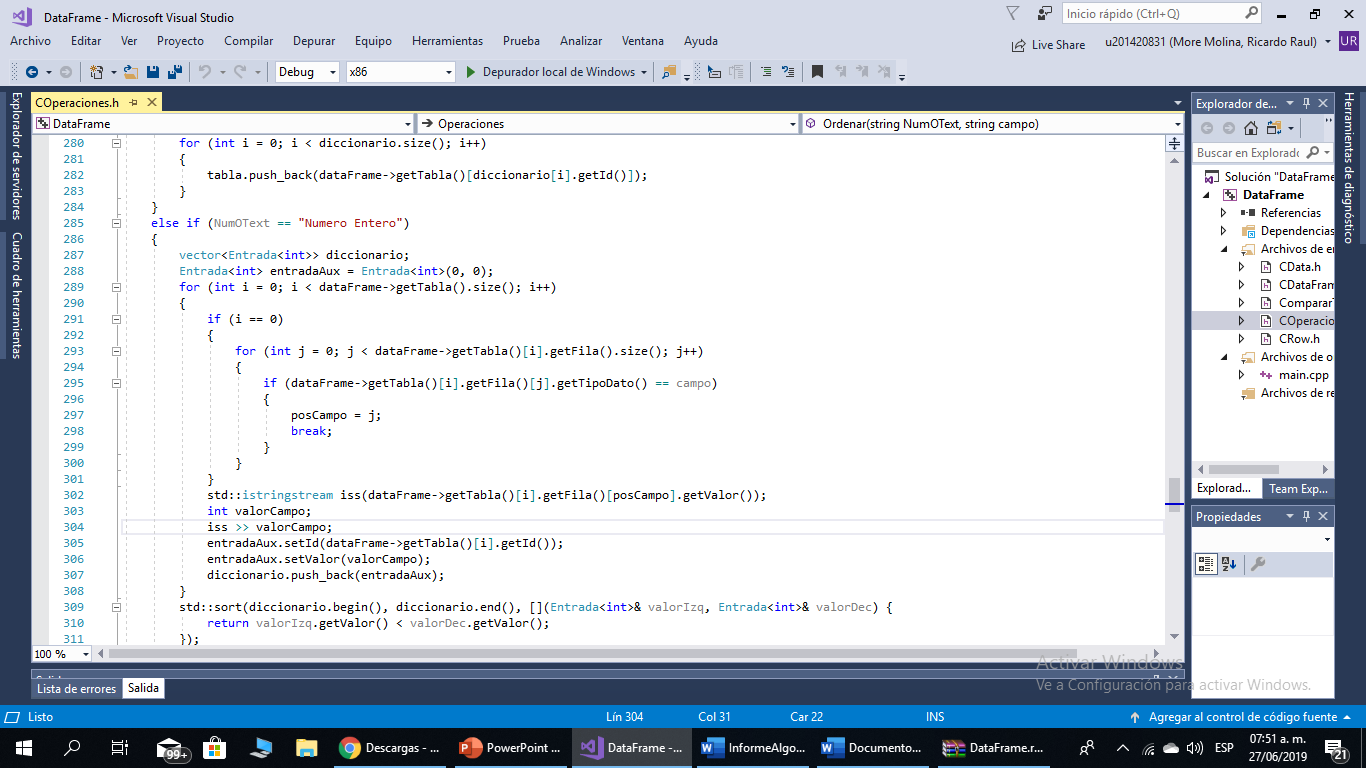


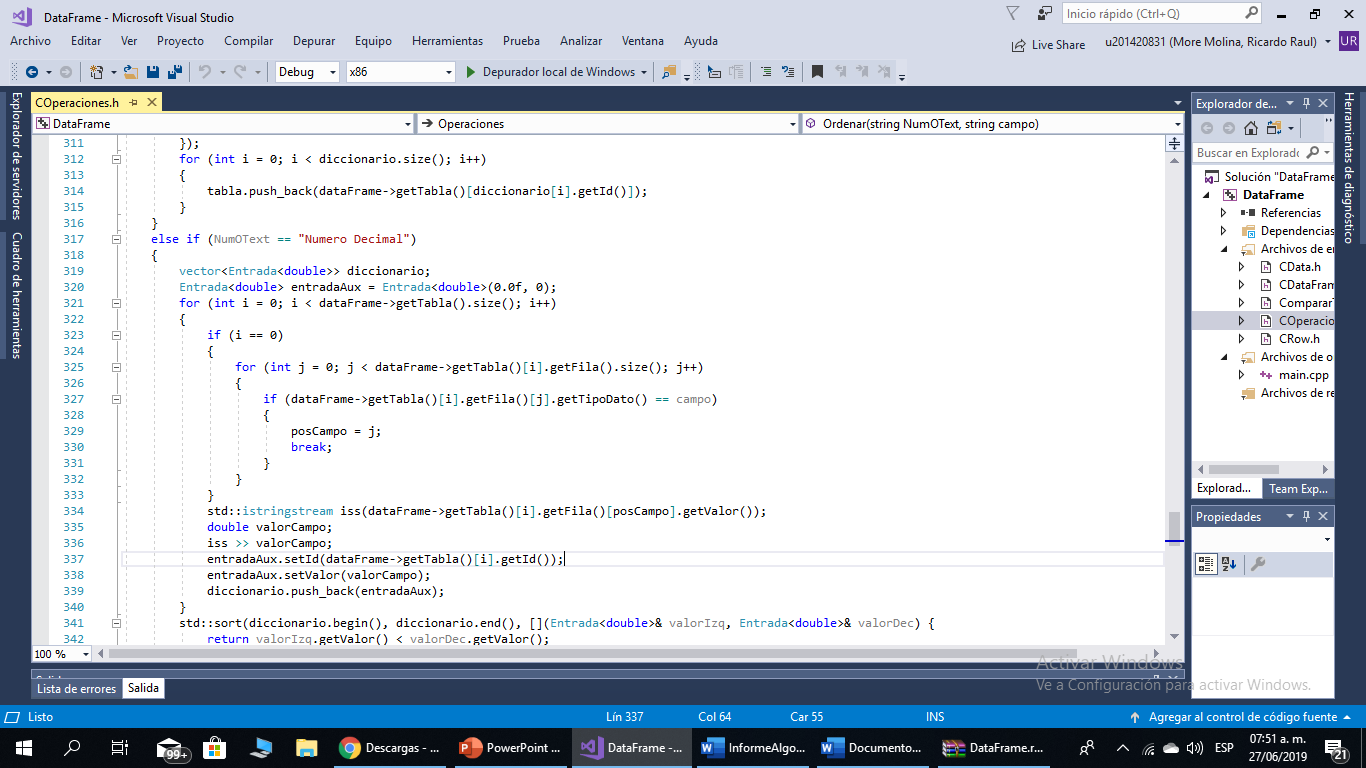


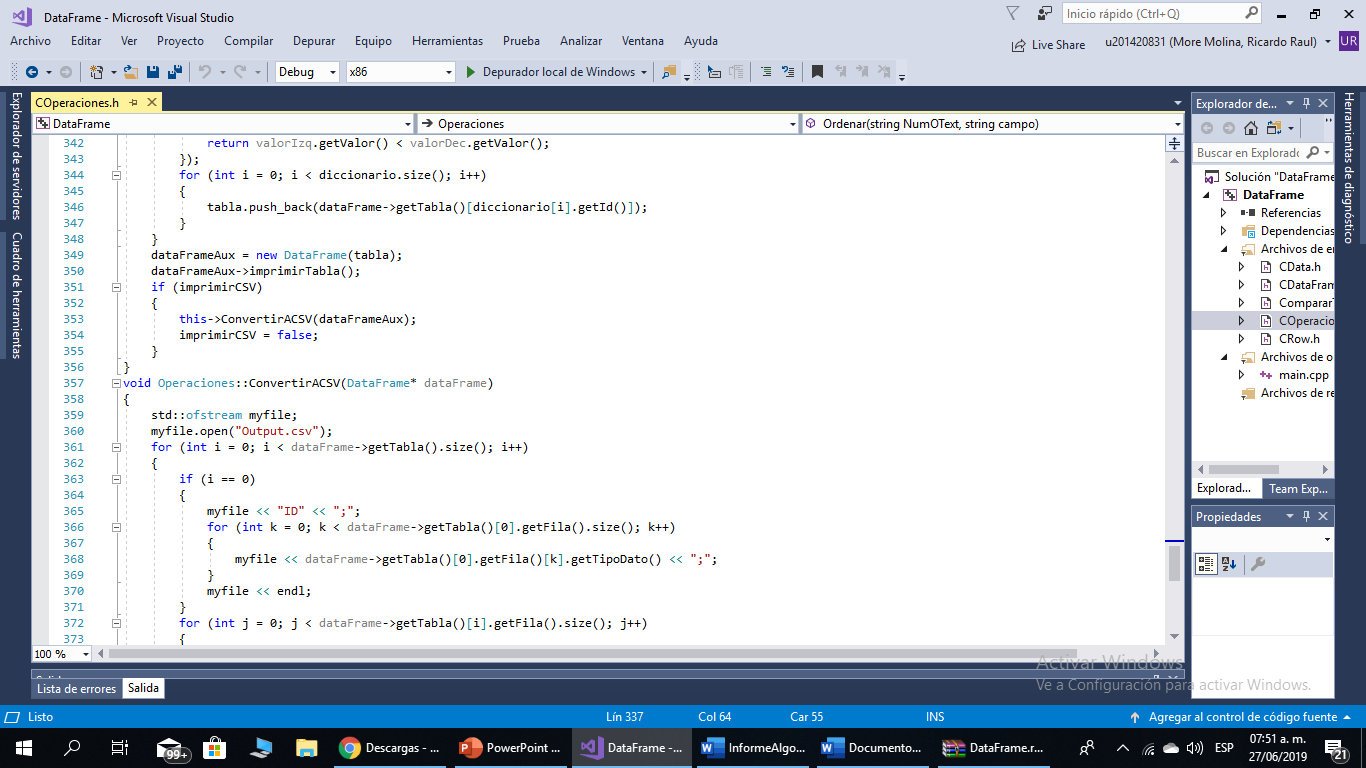


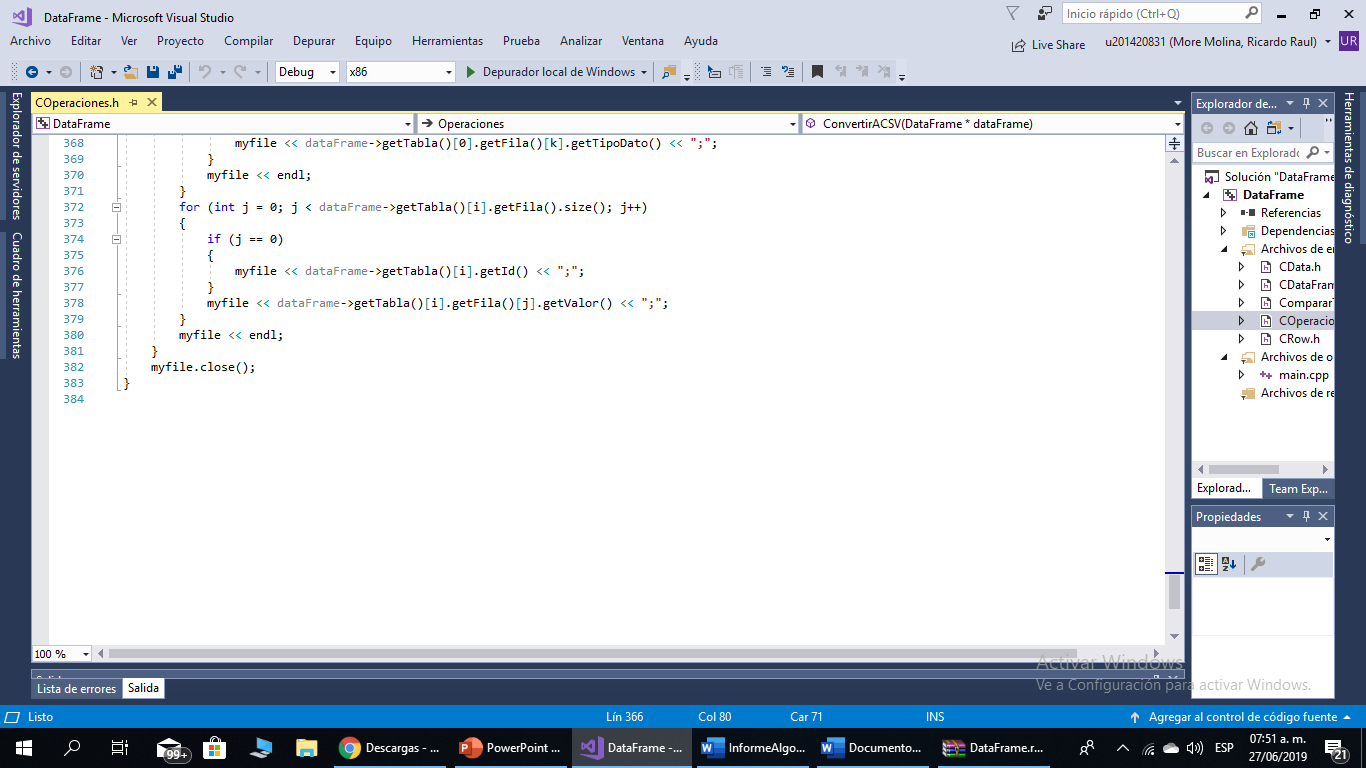




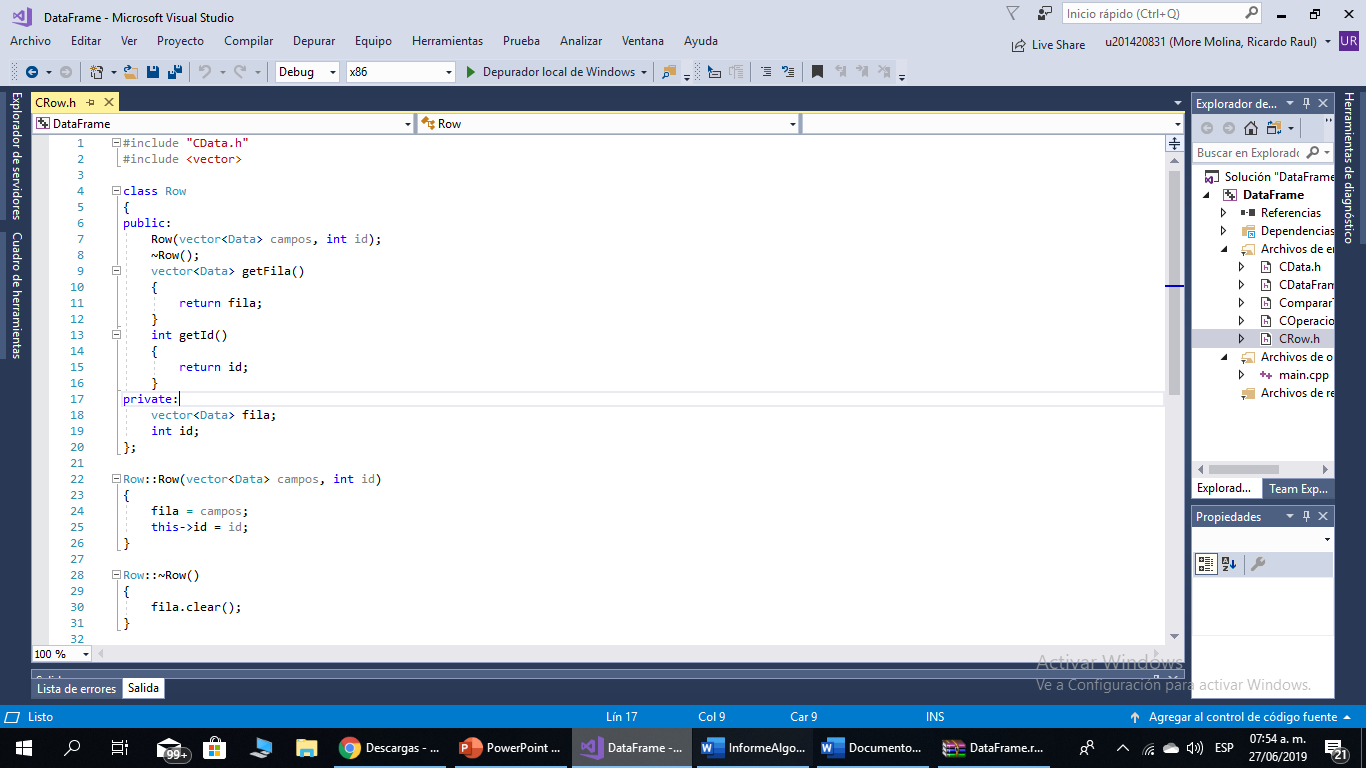








CROW



# Conclusiones

Pudimos concluir que crear una biblioteca para Dataframes se puede hacer de muchas formas, desde hacerlos con el contendor <maps> , con arrays o como en nuestro caso que elegimos usar vectores. Nos parece una estructura mas accesible de usar para este tipo de implementaciones

# Referencias

* Cplusplus (2000). *Vector.* Recuperado de <http://www.cplusplus.com/reference/vector/vector/> [Consulta: 26 de junio de 2019].
* Cplusplus (2000).*String.* Recuperado de <http://www.cplusplus.com/reference/string/> [Consulta: 26 de junio de 2019].