  
Trabajo Final: Hito 3  
  
  
**Curso:** Algoritmos y Estructura de Datos

**Profesor:** Wilder Adan Namay Zevallos

**Ciclo:** 2019-01

**Sección:** SV31

**Grupo engendrado por:**

* Marco Manrique Acha (U20171C084)
* Guillermo Rosales (U201613518)

Universidad Peruana de Ciencias Aplicadas (UPC)

Chorrillos, 2019.

**Introducción**

El presente informe tiene como finalidad detallar el proyecto final que se desarrollará para el curso de Algoritmos y Estructuras de Datos. Dicho proyecto consta en la elaboración de una librería en el lenguaje de C++ que permita crear, manipular y realizar operaciones en un Dataframe. Entiéndase este último término como una estructura de datos tabular bidimensional con ejes dispuestos en filas y columnas.

**Problema**

La correcta toma de decisiones sobre las estructuras de datos y algoritmos a utilizar. A su vez, la correcta implementación de lo mencionado para lograr la efectividad del Dataframe.

**Objetivos**

El objetivo principal del proyecto es la elaboración una librería “Dataframe.h” que permitirá crear, manipular y efectuar operaciones con Dataframes. Las operaciones que incluirá la librería serán: importación de datos, indexado de datos por columna, selección y filtrado de datos por columna, ordenamiento y exportación de datos. La implementación de la librería se hará en el entorno gráfico de C++ haciendo uso de Windows Forms. Por último, como consideraciones adicionales, el programa será construido haciendo uso de la programación orientada a objetos (POO) y será alojado en la herramienta de administración de repositorios Git, Github.

**Alcance de Proyecto**

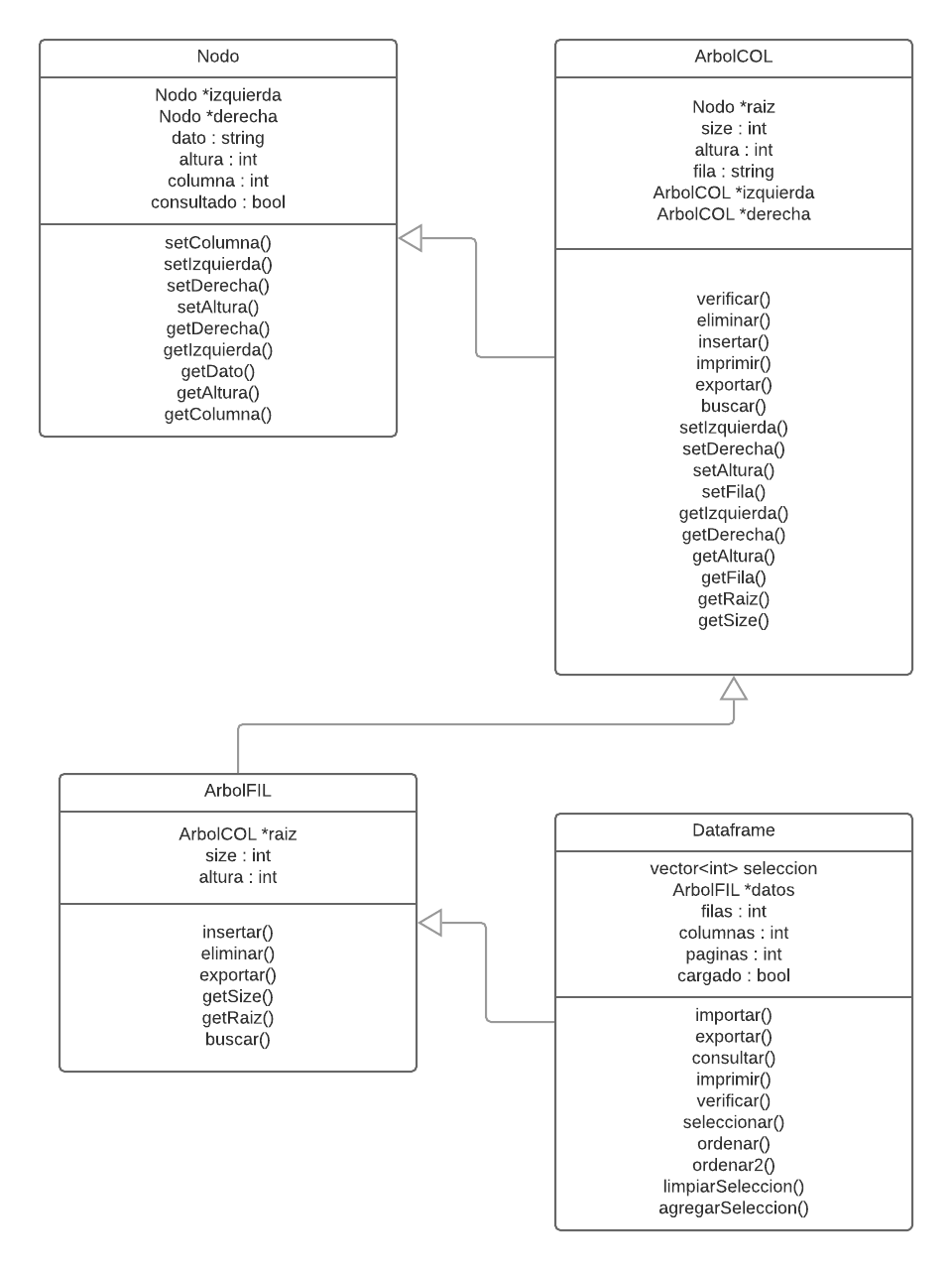
* Programación de las estructuras de datos haciendo uso de Templates.
* Codificación del algoritmo de ordenamiento.
* Implementación de las Clases con sus respectivos atributos y métodos.

**Marco conceptual**

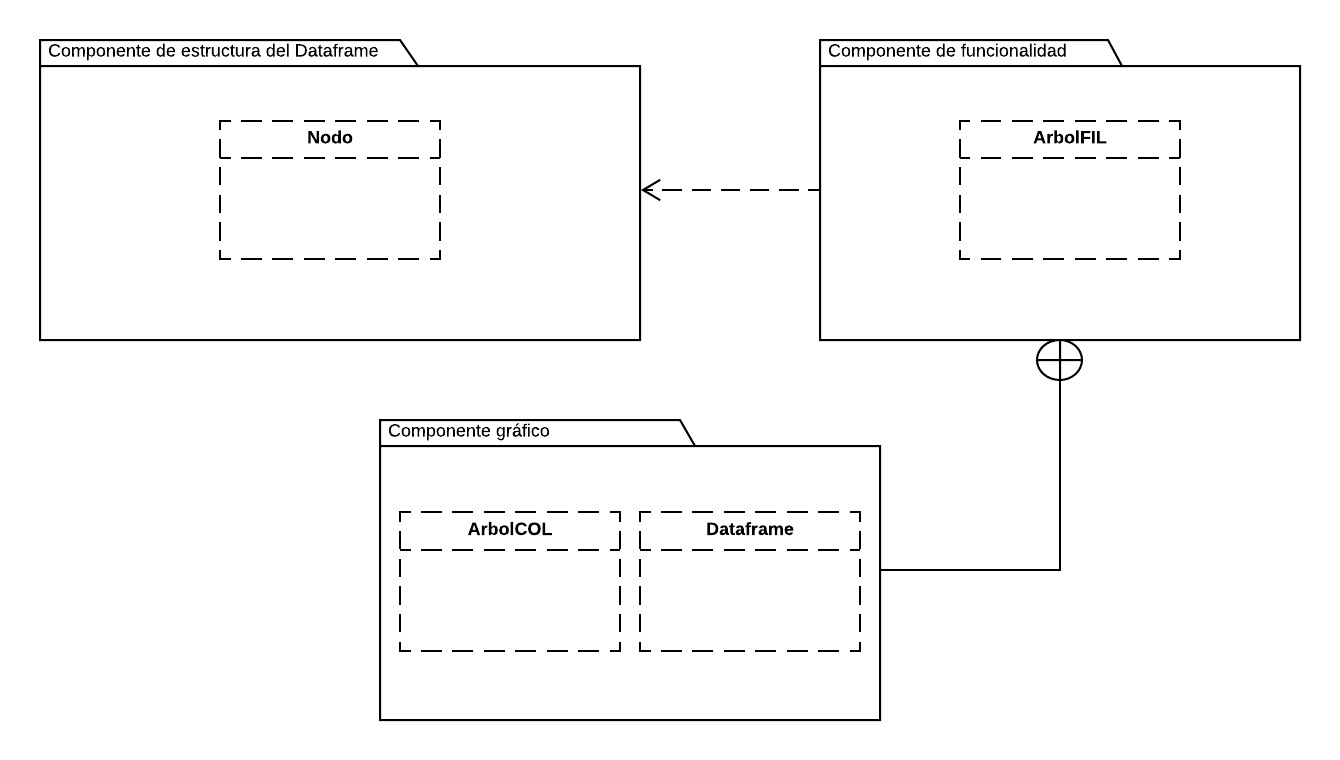
Para la elaboración del proyecto final se emplearán los siguientes conceptos:

* Clases y Objetos (POO): administración de parámetros para cada dataframe.
* Plantillas (templates): parametrizar clases para adaptarlas a cualquier tipo de dato.
* Archivos: lectura y escritura de archivos para la importacion y exportacion de datos respectivamente.
* Lista Doblemente Enlazada (EDD): almacenamiento de objetos.
* Estructura de Árbol AVL: indexado de filas (listas).
* Algoritmos de ordenamiento: para la organización de datos.
* Interfaz de aplicación gráfica (api): Windows Forms.

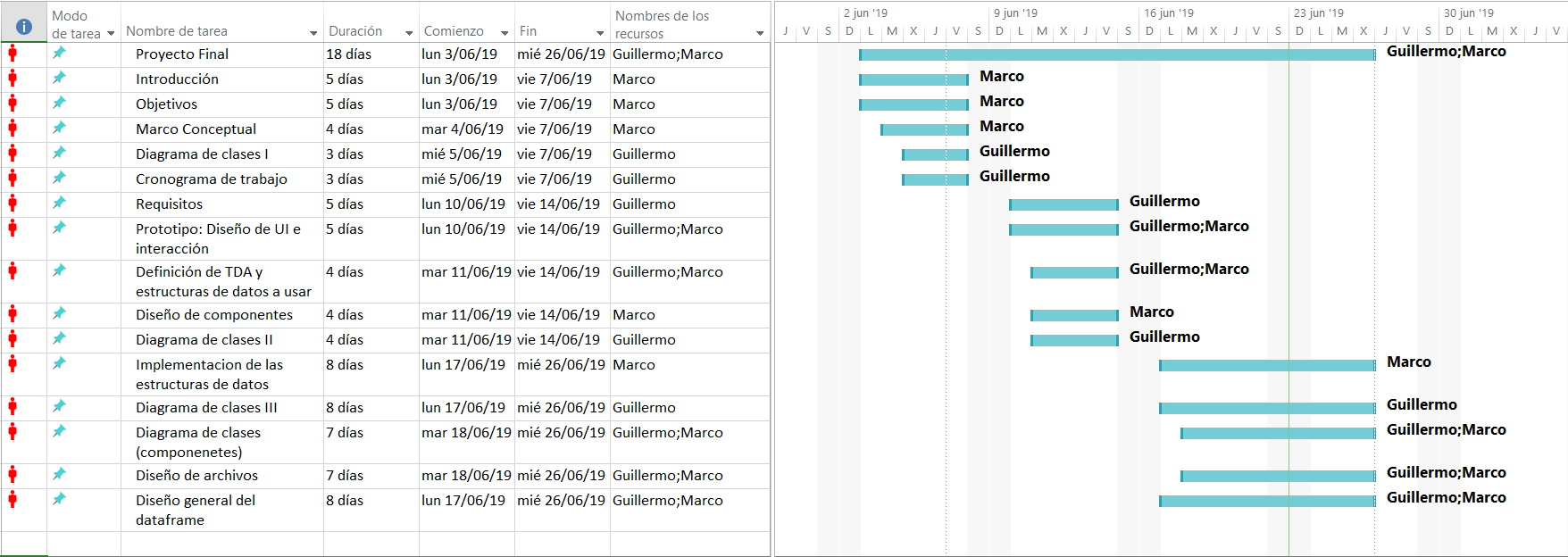
**Diagrama de clases III**



**Diagrama de componentes**



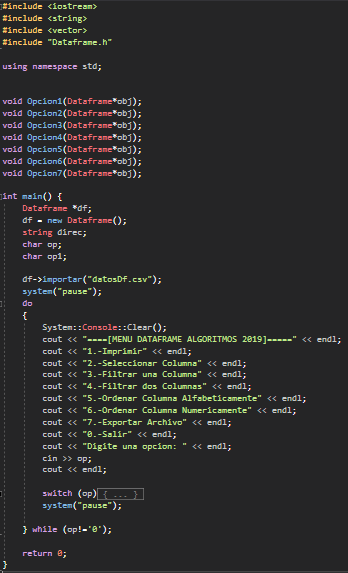
**Cronograma de trabajo**



**Requisitos**

El trabajo final del curso consiste en la construcción de un dataframe con las siguientes características mínimas.

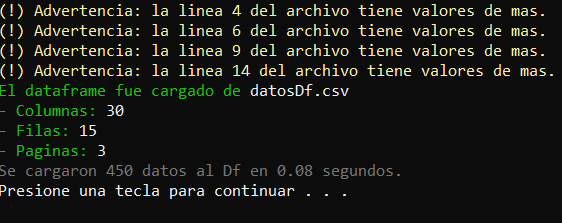
* El proyecto consiste en un Dataframe, una clase de objetos especial en lenguaje R, donde se generan informaciones en horizontal(sujetos) y vertical(variables).
* Una vez generado el Dataframe, este debe seguir permaneciendo durante toda la ejecución del proyecto; de modo que si el usuario desea editar y/o guardar el archivo, al compilar de nuevo el archivo debe estar como lo dejó anteriormente.
* La estructura de un Dataframe es muy similar a la de una matriz, lo que los diferencia es que en una matriz sólo admite valores numéricos, mientras que en un Dataframe se puede incluir también datos alfanuméricos.
* El usuario debe poder editar, cargar y guardar los archivos en el Dataframe.
* El usuario debe poder ordenar toda la información de dicha columna, de menor a mayor o viceversa.



**Ejecución**

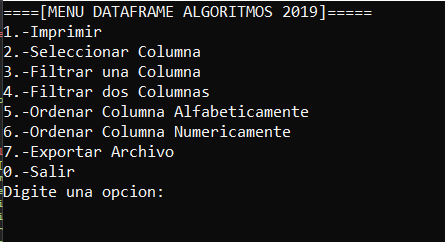
Según lo establecido en nuestra librería Dataframe, la forma de ejecución sería similar a la ejemplificada en las imágenes anteriores. Esta ejecución se alternaria una vez se deje de desarrollar para consola y se opte por una interfaz gráfica. La implementación 1 presenta la creación, importación y los datos del dataframe en si. La implementación 2 es similar solo que antes de imprimir, ejecuta un algoritmo para ordenar el dataframe según el criterio de la Columna C. Por último, cabe mencionar que los elementos de ejecución presentados pueden llegar a ser modificados.

**Diseño de interfaz de usuario**



La imagen anterior será el diseño que tendrá la interfaz de usuario de nuestro programa.

**Diseño de interacción**



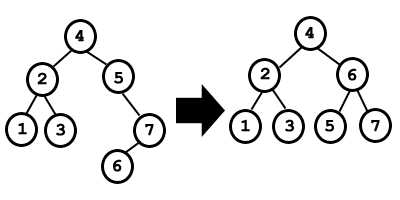
El usuario podrá seleccionar cada columna a su gusto para poder realizar las operaciones que desee. Consta de tres operaciones: Filtrado, Ordenamiento y Exportación.

**Tipos de datos abstractos**

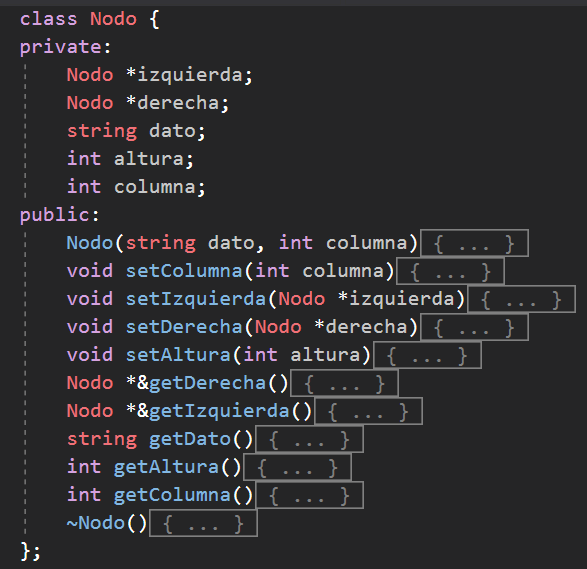
Un Tipo de dato abstracto (en adelante TDA) es una agrupación de datos u objetos al cual se le asocian operaciones. El TDA provee de una interfaz con la cual es posible realizar las operaciones permitidas, abstrayéndose de la manera en cómo están implementadas dichas operaciones. Como resultado de este proceso de abstracción un mismo TDA puede ser implementado utilizando diferentes estructuras de datos y proveer la misma funcionalidad.

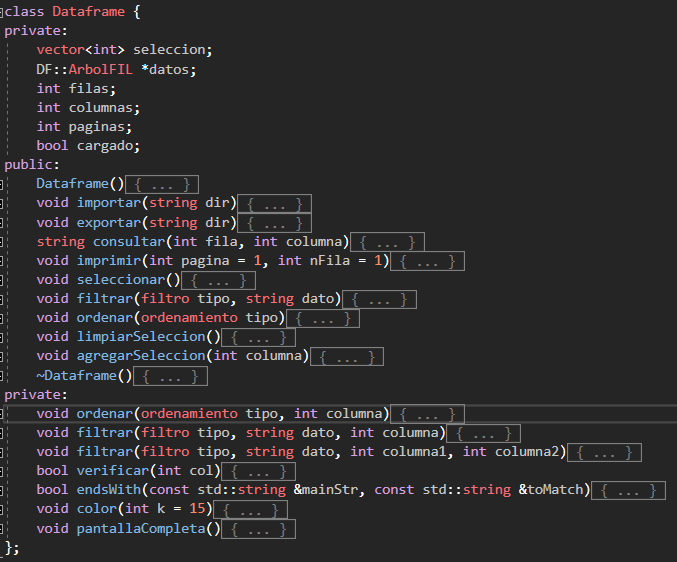
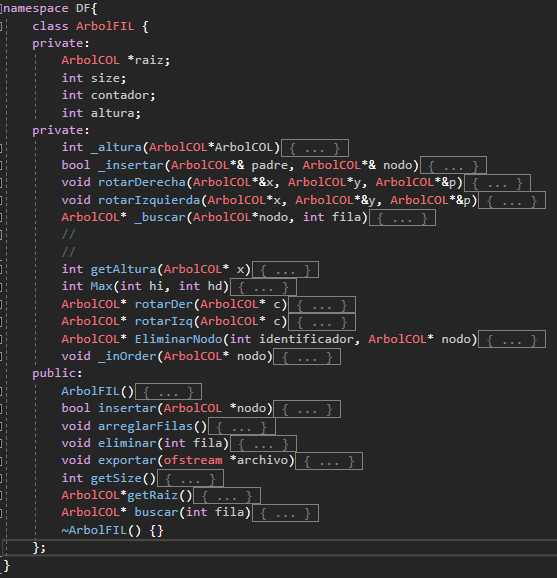
\*Los tipos de datos abstractos que se han implementado son:

**TDA Árbol AVL:** es un árbol binario de búsqueda que cumple con la condición de que la diferencia entre las alturas de los subárboles de cada uno de sus nodos sea como mucho 1 (Con esto el Árbol está balanceado). Además, cada nodo cumple con que todos los nodos de su subárbol izquierdo son menores que la raíz y todos los nodos del subárbol derecho son mayores que la raíz.



**Implementación de las estructuras de datos**

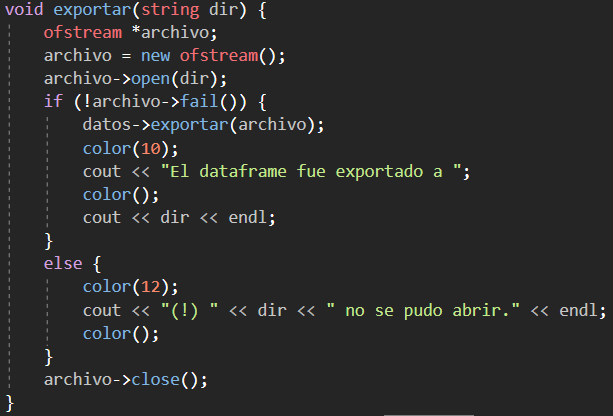




**Diseño de archivos**

Para poder abrir un fichero, leer y escribir se debe incluir la librería <fstream>. Por un lado, para abrir un fichero para lectura, debe crear un objeto “ifstream”(open for reading) que se usará como un “cin”. Por otro lado, para crear un fichero de escritura, se crea un objeto “ofstream”(open for writing) que se comporta como un “cout”. Una vez que tiene abierto el fichero puede leer o escribir en él como si usara cualquier objeto <iostream>.

Una de funciones las más útiles de la librería iostream es getline(), que permite leer una línea (terminada en nueva línea) y guardarla en un objeto string. El primer argumento es el objeto ifstream del que se va a leer la información y el segundo argumento es el objeto string. Cuando termina la llamada a la función, el objeto string contiene la línea capturada.



**Conclusiones**

La gran cantidad de información generada al momento de ejecutarse el programa plantea problemas de manejo de datos y su correcta recuperación. La eficacia en tiempo y espacio en el acceso a diversos datos refleja que se deben hacer el uso de diversas Estructura de datos sean éstas creadas o implementadas al momento de que se requiera de una.