Algoritmos y Estructuras de Datos I

Trabajo Práctico N°3	Unidad 3.3
Modalidad: Semi -Presencial	**Estratégica Didáctica:** Trabajo grupal
Metodología de Desarrollo: Det. docente	**Metodología de Corrección:** Via Classroom.
Carácter de Trabajo: Obligatorio – Con Nota	**Fecha Entrega:** A confirmar por el Docente.

MARCO TEÓRICO

Responder el siguiente cuestionario en función de la bibliografía Obligatoria.

Unidad 4

1. ¿Por qué necesitaría usar Archivos en una aplicación?

Los archivos en una aplicación es necesario para guardar la información en el dispositivo..
Es una secuencia de bytes almacenados en un dispositivo. Un archivo es identificado por un nombre y una extensión que determinan su Tipo.
Los archivos o ficheros son la forma en la que C++ permite el acceso al disco.

2. ¿Explicar las diferencias entre archivos de texto y binarios? (ampliar sobre lo visto en clase)

Los dos tipos de archivos pueden tener el mismo aspecto en la superficie, pero codifican datos de forma diferente.
Los archivos de texto: Contiene solamente datos de texto. Son archivos normales que contienen información almacenada en formato ASCII y que el usuario puede leer.
Los archivos binarios: Contienen tanto datos binarios y de texto. Podrían ser archivos ejecutables que indican al sistema que ha de realizar un trabajo. Los mandatos y los programas se almacenan en archivos binarios ejecutables. Los programas de compilación especial convierten texto ASCII en código binario.

3. ¿Qué entiende por “Flujos” en la lectura y escritura?

Flujo de información del código de llamada a una función y de la función al código de llamada. El flujo es una secuencia de bytes, realiza operaciones de entradas, los bytes fluyen de un dispositivo a una memoria.

4. ¿Explicar la librería y los objetos (la secuencia) que se utilizan para leer un archivo?

La librería es la que define las clases necesarias para la utilización de ficheros, estos pueden ser <ifstream>, <ofstream>, <fstream> que derivan del <istream> y <ostream>.
Para abrir el fichero se debe definir el objeto, que puede ser <ifstream>, <ofstream> o <fstream>.
Dependiendo de lo que deseemos hacer con el fichero, usaremos objetos de las clases:
·ifstream (input file stream), clase orientada para la lectura
·ofstream (output file stream). clase orientada para la escritura ·fstream (file stream), cuando deseemos alternativamente leer o escribir del mismo fichero en el mismo programa.

1) Se debe incluir la librería
   <iostream> //entrada -salida
   <fstream> //incluye y maneja los archivos
   #include <string> //permite usar cadena de textos
2) seguimos con la escritura o lectura del operador.

que puede ser
<ifstream> //lectura
<ofstream> //escritura
<fstream> //hace ambos, lee y escribe

3) se abre el archivo “nombreDelArchivo. txt”
4) se elige al operados de la lectura de archivo “variable”, getline ”cadena del archivo”
5) se cierra el archivo

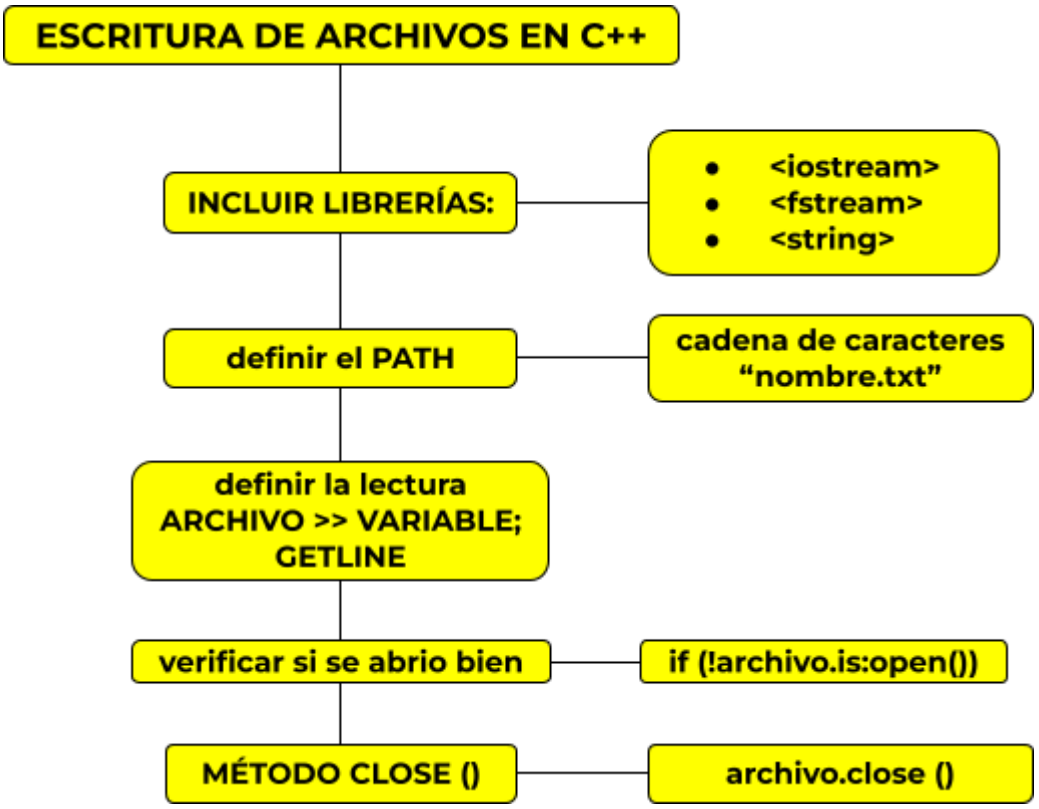
Uso de archivos

Si se quiere que un programa use el archivo I/O, se tienen que hacer cuatro cosas:

1. Solicitar al preprocesador que incluya el archivo de encabezado `fstream`.
2. Usar las sentencias de declaración para declarar los flujos de archivo que se van a usar.
3. Preparar cada archivo para lectura o escritura al usar una función llamada `open`.
4. Especificar el nombre del flujo de archivo en cada sentencia de entrada o salida.

5. ¿Cómo deberemos poner el camino o “Path” del archivo a leer o escribir?
El Path o camino se debe situar entre comillas, para la cadena de caracteres que indica la ubicación de un archivo. un ejemplo seria “salida.txt”.

6. Explicar en un Diagrama la secuencia de comandos usados en la Escritura de Archivo



7. Dar Ejemplos de código “Descriptivamente Comentado” de Lectura de Archivos.

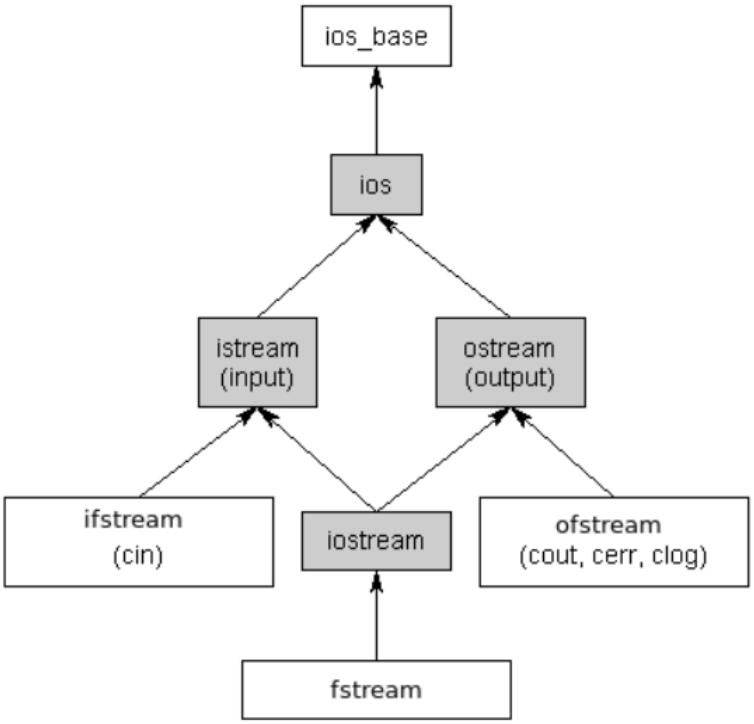
```
#include <iostream>                 // Incluimos la librería para manejo de archivos
#include <fstream>
#include <string>

 int main() {
 std::string ruta = "datos.txt";    // Definimos la ruta o nombre del archivo a leer

 std::ifstream archivo;             // Creamos un objeto ifstream para la lectura del archivo

archivo.open(ruta);                 // Abrimos el archivo especificado por la ruta

if (!archivo.is_open()) {           // Verificamos si el archivo se abrió correctamente

std::cerr << "Error al abrir el archivo: " << ruta << std::endl;      // Mostramos un mensaje de error si no se pudo abrir el archivo

return 1;                           // Salimos del programa con código de error
}

int numero;                         // Variables para almacenar datos leídos
std::string linea;
archivo >> numero;

 std::cout << "Número leído: " << numero << std::endl;          // Ejemplo de lectura de un número del archivo

archivo.ignore();                           // Consumir el carácter de salto de línea pendiente

std::getline(archivo, linea);       // Ejemplo de lectura de una línea completa del archivo
td::cout << "Línea leída: " << linea << std::endl;

 archivo.close();                           // Cerramos el archivo después de terminar la lectura

std::cout << "Lectura de archivo completada." << std::endl;

return 0;
}
```

8. Dar Ejemplos de código “Descriptivamente Comentado” de Escritura de Archivos.

```
#include <iostream>
#include <fstream>                  // Incluimos la librería para manejo de archivos
#include <string>

int main() {
std::string ruta = "salida.txt";    // Definimos la ruta o nombre del archivo a escribir

 std::ofstream archivo;             // Creamos un objeto ofstream para la escritura del archivo

 archivo.open(ruta);                // Abrimos el archivo especificado por la ruta

if (!archivo.is_open()) {           // Verificamos si el archivo se abrió correctamente
                                    // Mostramos un mensaje de error si no se pudo abrir el archivo

std::cerr << "Error al abrir el archivo: " << ruta << std::endl;

return 1;                           // Salimos del programa con código de error
 }

                                    // Ejemplo de escritura en el archivo
archivo << "Escribiendo en un archivo desde C++!" << std::endl;
archivo << 42 << std::endl;

archivo.close();                    // Cerramos el archivo después de terminar la escritura

 std::cout << "Escritura en archivo completada." << std::endl;

return 0;                           // Salida exitosa del programa
}
```

MARCO OPCIONAL TEÓRICO

Opcional 1: Para los que quieren Ampliar les copio el "Modelo de Objetos" que representa la FamiliaFStream.



Opcional 2 ¿Cómo puedo leer y analizar archivos CSV en C++?

Los archivos CSV, utilizan librerías estándar como <iostream> o <fstream>;
y <string> (para manipular las cadenas). abrimos el archivo .CSV usando el <ifstream>;
cada línea está separada por comas y almacena vectores, vectores en cadenas. Si se realiza correctamente se puede manipular flujos de entradas de archivos y procesar datos de manera segura, pero extensa.

- Abrir el archivo CSV utilizando std::ifstream
- Leer el archivo línea por línea, utilizando std::getline()
- Dividir cada línea en campos utilizando un delimitador (generalmente una coma), por ejemplo, con std::stringstream
- Almacenar los campos en una estructura de datos adecuada, como un vector de vectores de strings (std::vector<std::vector<std::string>>).
- Procesar y analizar los datos según tus necesidades.

Opcional 3: ¿Qué son los Archivos Binarios?

Los archivos binarios son aptos para maquinas, son usados para almacenar los datos de manera eficiente.

Opcional 4 : ¿Qué tipos de Archivos se pueden leer de forma binaria?

Algunos tipos de archivos que pueden ser leídos de forma binaria incluyen:
Imágenes (JPG, PNG, GIF, BMP)
Archivos de audio (MP3, WAV, FLAC)
Archivos de video (MP4, AVI)
Bases de datos (SQLite, MySQL)
Formatos de datos estructurados (XML, JSON, YAML)

Opcional 5: ¿Qué tipos de Librerías de Objetos se deben usar?

Se recomienda el uso de librerías
<istream> Para leer datos del flujo de entrada;
<ifstream> Para leer datos de un archivo de entrada;
<iostream> Para manejar flujos de entrada y salida;
<iomanip> Para dar formato a los datos de entrada y salida.

Opcional 6: Dar ejemplo de Apertura y cierre de Archivos Binarios

```
#include <iostream>
#include <fstream>
#include <vector>

struct Datos {
int id;
double valor;
};

int main() {
std::ofstream archivo_salida("datos.bin", std::ios::binary);          // Abrir archivo binario para escritura

if (!archivo_salida.is_open()) {
std::cerr << "Error al abrir el archivo binario para escritura." << std::endl;
 return 1;
}

std::vector<Datos> datos = {{1, 3.14}, {2, 6.28}};                  // Ejemplo de escritura de datos en el archivo binario

archivo_salida.write(reinterpret_cast<const char*>(&datos[0]), datos.size() * sizeof(Datos));

archivo_salida.close(); // Cerrar archivo binario

std::cout << "Datos escritos en archivo binario correctamente." << std::endl;

return 0;
}
```

MARCO PRÁCTICO

Desarrollar en C++ los siguientes programas:
(Se recomienda utilizar Funciones, Tipos definidos x el usuario y Enumeradores)
(Actividad Anterior, Leer y Reflexionar sobre las frases de Bjarne Stroustrup que se usarán en el archivo de
Texto.)

Resolver y Realizar en c++

a. Crea un programa que vaya leyendo las frases que el usuario teclea y las guarde en un fichero de texto llamado “frasesDeBjarme.txt”. Terminará cuando la frase introducida sea "fin" (esa frase no deberá guardarse en el fichero).
b. Para el ejercicio anterior, Abrir el archivo y mostrar las frases x pantalla, con cada frase en una línea, con Interlineado.
c. Para el Archivo del ejercicio (a) contar la cantidad de palabras que hay x línea, luego mostrar la cantidad total y el promedio de palabras x línea.
d. Para el archivo de “frasesDeBjarme.txt” indicar cual es la palabra mas Larga.