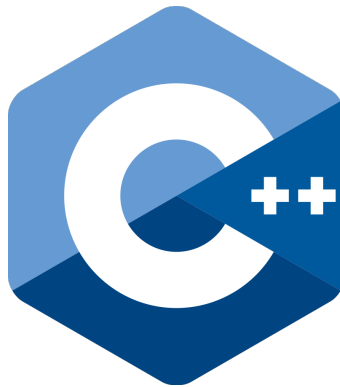




# Metodología de la Programación

Curso 2023/2024



## Guion de prácticas

*Ficheros de texto*



# Contents

<b>1 Descripción</b>	<b>5</b>
<b>2 Objetivos</b>	<b>5</b>
<b>3 Manejo de ficheros de texto</b>	<b>5</b>
3.1 Procedimiento para leer datos desde un fichero . . . . .	5
3.2 Procedimiento para escribir datos en un fichero . . . . .	6
3.3 Gestión básica de errores con ficheros . . . . .	8
<b>4 Videotutoriales</b>	<b>8</b>



## 1 Descripción

Por ahora los únicos datos con los que pueden trabajar nuestros programas son los que introducimos desde el teclado o los que mostramos por la pantalla, pero en programas reales esto no es suficiente, sino que se hace necesario almacenar de forma permanente tanto los datos de entrada como los de salida de un mismo programa, para lo que se recurre al uso de ficheros en memoria masiva. En esta práctica introduciremos muy brevemente el protocolo de manejo de ficheros de texto, dejando los ficheros binarios y los detalles más avanzados del uso de ficheros para su introducción en temas posteriores.

## 2 Objetivos

- Conocer la estructura de los ficheros de texto.
- Leer datos de un fichero de texto.
- Escribir datos en un fichero de texto.

## 3 Manejo de ficheros de texto

Los ficheros de texto contienen datos que han sido codificados como texto usando un esquema como ISO8859-1<sup>1</sup> o UTF<sup>2</sup> y, a diferencia de los ficheros binarios, aparecen exactamente como en la pantalla del ordenador, es decir, como una secuencia de caracteres. Por tanto un fichero de texto puede abrirse y editarse con programas editores como `notepad` en Windows o `gedit` en Ubuntu Linux.

### 3.1 Procedimiento para leer datos desde un fichero

Los datos de un fichero se pueden procesar de forma muy similar a como se leen datos desde el teclado o como se escriben datos en pantalla con los operadores de extracción `>>` y de inserción `<<` en un flujo de datos. En el Cuadro 1 se pueden ver dos programas que leen exactamente los mismos valores para cada variable con la diferencia de que el primero los lee desde el teclado y el segundo los lee desde un fichero llamado "datos.txt".

Se puede ver que el procedimiento para leer los datos desde un fichero es muy sencillo y consiste en los siguientes pasos.

1. Incluir el fichero de cabeceras `fstream` para poder manejar ficheros.
2. Crear un flujo de datos de entrada `ifstream` que proviene de un fichero cuyo nombre es `fentrada` y asociarlo al fichero `datos.txt` del que se van a leer los datos.

---

<sup>1</sup>Codificación ISO8859-1 ([Abrir →](#))

<sup>2</sup>Codificación UTF-8 ([Abrir →](#))


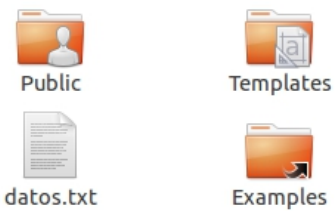
Código C++	Datos introducidos desde el teclado
<pre>#include &lt;iostream&gt; using namespace std; int main() {     int i;     double d;     char c[64];      cin &gt;&gt; i;     cin &gt;&gt; d;     cin &gt;&gt; c;     return 0; }</pre>	 10   3.14   Pepe
Código C++	Datos en el fichero datos.txt
<pre>#include &lt;iostream&gt; #include &lt;fstream&gt; using namespace std; int main() {     int i;     double d;     char c[64];     ifstream fentrada;      fentrada.open("datos.txt");     fentrada &gt;&gt; i;     fentrada &gt;&gt; d;     fentrada &gt;&gt; c;     fentrada.close();     return 0; }</pre>	 10   3.14   Pepe

Tabla 1: Dos programas y sus datos asociados, que leen exactamente los mismos valores para cada variable.

3. Leer los datos con el operador de extracción >> exactamente igual que si se leyesen desde `cin`.
4. Cuando hemos terminado de leer los datos cerramos el flujo de entrada desde el fichero.

### 3.2 Procedimiento para escribir datos en un fichero

Para escribir datos en un fichero el procedimiento es muy similar y aparece ilustrado en los ejemplos del Cuadro 2.

En este caso, el procedimiento para guardar o escribir los datos en un fichero también es muy sencillo y consiste en los siguientes pasos:

1. Incluir el fichero de cabeceras `fstream` para poder manejar ficheros.
2. Crear un flujo de datos de salida `ofstream` hacia un fichero cuyo nombre es `fsalida` y asociarlo al fichero `datos.txt` en el que se van a escribir los datos. Si el fichero no existe, se crea, y si ya existe, se borran sus datos antes de empezar a escribir en él.


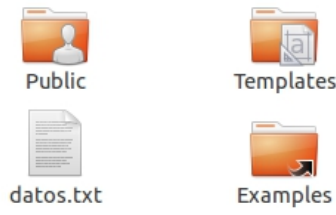



Código C++	Datos visualizados en pantalla
<pre>#include &lt;iostream&gt; using namespace std; int main() {     int i=10;     double d=3.14;     char c[64]="Pepe";      cout &lt;&lt; i &lt;&lt; " ";     cout &lt;&lt; d &lt;&lt; " ";     cout &lt;&lt; c &lt;&lt; endl;     return 0; }</pre>	 <p>10 3.14 Pepe</p>
Código C++	Datos guardados en el fichero datos.txt
<pre>#include &lt;iostream&gt; #include &lt;fstream&gt; using namespace std; int main() {     int i=10;     double d=3.14;     char c[64]="Pepe";     ofstream fsalida;      fsalida.open("datos.txt");     fsalida &lt;&lt; i &lt;&lt; " ";     fsalida &lt;&lt; d &lt;&lt; " ";     fsalida &lt;&lt; c &lt;&lt; endl;     fsalida.close();     return 0; }</pre>	 <p>Public</p>  <p>Templates</p>  <p>datos.txt</p>  <p>Examples</p> <p>10 3.14 Pepe</p>

Tabla 2: Dos programas que escriben exactamente los mismos valores para cada variable en la pantalla (el primero) y en el fichero "datos.txt" (el segundo).

3. Escribir los datos en el fichero con el operador de inserción << exactamente igual que si se mostrasen por pantalla con cout.
4. Cuando hemos terminado de escribir los datos cerramos el flujo de salida.




### 3.3 Gestión básica de errores con ficheros

Son muchas las casuísticas que se pueden presentar cuando se manejan datos desde o hacia un fichero. Se deben comprobar los posibles errores en la apertura y en el flujo de lectura / escritura de un fichero. El primer error se produce cuando se intenta abrir un fichero que no existe o cuando no se tienen privilegios para abrir el fichero o para escribir en él. El segundo error puede ocurrir cuando las lecturas / escrituras no se han realizado con éxito. Aunque la apertura y lectura/escritura de datos haya sido la correcta, se pueden producir errores adicionales en la interpretación de los datos leídos en casos de incompatibilidad de datos o de cantidad excesiva de datos. A continuación se muestra un ejemplo de la comprobación de errores asociados a las operaciones con ficheros:

```
#include<iostream>
#include<fstream>
using namespace std;

int main(){
    int i;
    double d;
    char c[64];
    ifstream fentrada;
    fentrada.open("datos.txt");
    if (fentrada){
        fentrada >> i;
        fentrada >> d;
        fentrada >> c;
        // if (fentrada.eof()){
        if (!fentrada){
            cerr << "error_de_lectura_del_fichero\n";
        }
        fentrada.close();
    }
    else{
        cerr << "error_de_apertura_del_fichero\n";
    }
}
```

## 4 Videotutoriales

1. Entrada de datos desde un fichero  [\(Abrir →\)](#)
2. Salida de datos hacia un fichero  [\(Abrir →\)](#)
3. Gestión de errores  [\(Abrir →\)](#)