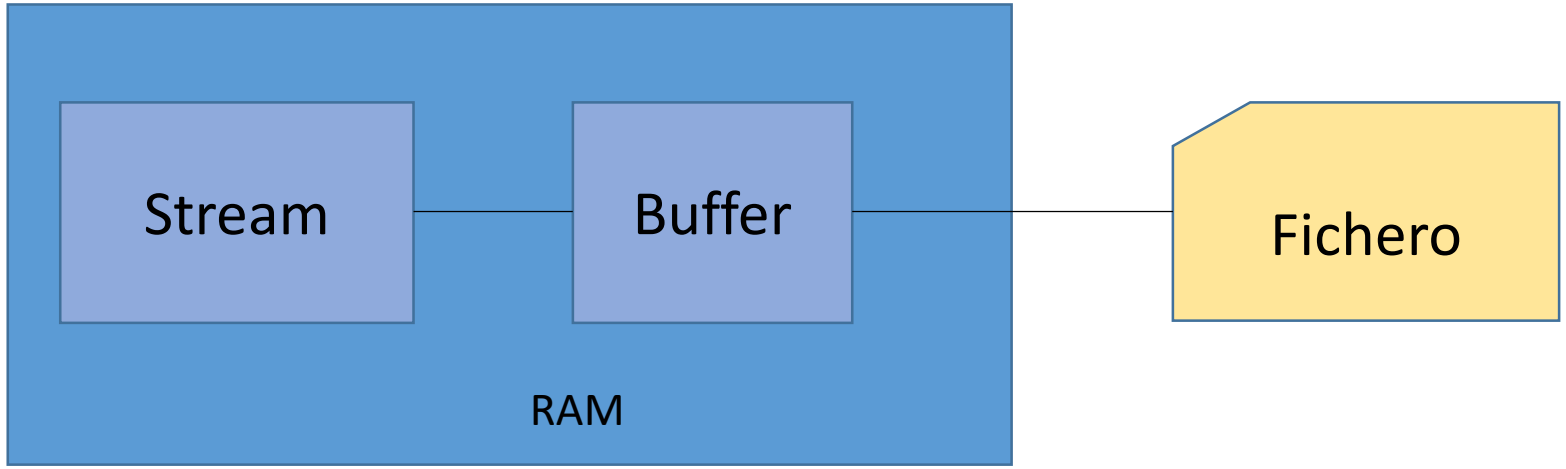
# FICHEROS o ARCHIVOS

Los archivos son medios que facilita el lenguaje para almacenar los datos en forma permanente, normalmente en los dispositivos de almacenamiento estándar.

En C++, se utilizan streams (flujos) para gestionar la lectura y escritura de datos. Ya conocemos dos flujos estándar: **cin** y **cout.**

En definitiva, **abrir un fichero** significa **definir un stream**. Dicho stream permite la transferencia de datos entre el programa y el fichero en disco.



El archivo de cabecera **fstream.h** define las clases **ifstream**, **ofstream** y **fstream** para operaciones de lectura, escritura y lectura/escritura respectivamente en archivos. Para trabajar con archivos debemos crear objetos de éstas clases

```
//  Escritura básica en archivo
#include <iostream>
#include <fstream>

using namespace std ;
int main()
{
   //ofstream archivo;  // Crea objeto de la clase ofstream
   //archivo.open("datos.txt"); //Llama método open

   ofstream archivo("datos.txt“);  // constructora de ofstream (salida) no hace falta open

   archivo << "Primera línea de texto" << endl;
   archivo << "Segunda línea de texto" << endl;
   archivo << "Última línea de texto" << endl;

   archivo.close();
   return 0;
}
```

```
// Lectura básica en archivo
#include <iostream>
#include <fstream>
#include <string>

using namespace std ;
int main()
{
  string linea;

  ifstream archivo("datos.txt");
  while (! archivo.eof())
  {
            getline(archivo,linea);// Leer también espacios blanco
            cout << linea <<endl ;
  }
   archivo.close();
   return 0;
}
```

## Modos de apertura de archivo

**ios::app** Si existe el fichero añade datos y sino lo crea. Ej. ***ofstream archivo("datos.txt“, ios::app);***

**ios::ate** Coloca el apuntador del archivo al final del mismo.

**ios::in** Operaciones de lectura. Esta es la opción por defecto para objetos de la clase ifstream.

**ios::out** Operaciones de escritura. Esta es la opción por defecto para objetos de la clase ofstream.

**ios::nocreate** Si el archivo no existe se suspende la operación.

**ios::noreplace** Crea un archivo, si existe uno con el mismo nombre la operación se suspende.

**ios::trunc** Crea un archivo, si existe uno con el mismo nombre lo borra.

**ios::binary** Operaciones binarias.

# ARCHIVOS BINARIOS

Los archivos binarios almacenan la información “byte a byte” en lugar de carácter por carácter, esto es muy útil cuando se necesita almacenar o recuperar información que no es un texto, por ejemplo: una imagen, un archivo ejecutable, un objeto, o datos numéricos de punto flotante.

**Apertura**: Modo binary o salida o entrada. Si el fichero no existe da error

```
fstream fichero(“nombrefichero",ios::binary|ios::out|ios::in)
```

**Escritura**

```
fichero.write((char*) &var , sizeof(var));
```

**Lectura**

```
fichero.read((char*) &var , sizeof(var));
```

**Saltar a una posición del fichero**

```
fichero.seekg(saltar_num_bytes , posición_salto);
```

Opcional

ios::beg salta desde principio
ios::cur salta desde la posición actual
Ios::end salta desde el final

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

using namespace std ;
int main(){
    int i,N;

    fstream f("ejemplo04.dat",ios::binary|ios::out|ios::in);
    if (!f)
   {
                cout<<"Error el fichero no existe";
                return 0;
   }
  for (i=1;i<=10;i++)
                f.write((char*)&i,sizeof(int)); // Almacena los 10 primeros enteros

  f.seekg(4*sizeof(int)); // se posiciona al principio del quinto entero
  f.read((char*)&N,sizeof(int)); // Lee dicho entero
  cout <<endl << "Quinto= " << N << endl; // visualiza el valor 5

   f.seekg(4*sizeof(int)); // se posiciona de nuevo en el quinto  pues el cursor avanza.
   i=100;
  f.write((char*)&i,sizeof(int)); // Modifica el valor 5 por el valor 100;
  f.seekg(0*sizeof(int)); // se posiciona de nuevo al principio del fichero
  for (i=1;i<=10;i++)
  {
                f.read((char*)&N,sizeof(int));
                cout <<endl << i << "= " << N << endl; // Se visualiza el contenido
  }

  cout<<f.tellg()<<" Bytes tamaño del fichero"<<endl;
  f.seekg(0,ios::beg); //ir al principio del fichero
  cout<<f.tellg()<<" Posición inicial"<<endl;
  f.seekg(4,ios::beg);
  f.seekg(2,ios::cur);
  cout<<f.tellg()<<“posición actual";
  f.close();
  return 0;
}
```