# Librerías comunes de C++ y su sintaxis

Javier Morales Rosas

# Librerías

Dentro del entorno de C ++, además de sus compiladores, se incluyen ciertos archivos llamados bibliotecas más comúnmente librerías. Las librerías contienen el código objeto de muchos programas que nos permiten realizar cosas comunes, como leer el teclado, escribir en la pantalla, manejar números, realizar funciones matemáticas, etc.

Las librerías están clasificadas por el tipo de trabajos que hacen, como pueden ser de entrada y salida, matemáticas, de manejo de memoria, de manejo de textos y como imaginarás existen muchísimas librerías disponibles y todas con una función específica.

# ¿Qué es una librería?

Las librerías son archivos (no siempre externos) que nos permiten llevar a cabo diferentes tareas sin necesidad de preocuparnos por cómo se hacen sino simplemente entender cómo usarlas.

Las librerías en C + + permiten hacer nuestros programas más modulares y reutilizables, facilitando además crear programas con funcionalidades bastante complejas en unas pocas líneas de código.

# Librerías más comunes de C++

**fstream**: Flujos hacia/desde ficheros. Permite la manipulación de archivos desde el programar, tanto leer como escribir en ellos.

**iosfwd**: Contiene declaraciones adelantadas de todas las plantillas de flujos y sus typedefs estándar. Por ejemplo ostream.

**iostream**: Parte del a STL que contiene los algoritmos estándar, es quizá la más usada e importante (aunque no indispensable).

**list**: Parte de la STL relativa a contenedores tipo list; listas doblemente enlazadas

**math**: Contiene los prototipos de las funciones y otras definiciones para el uso y manipulación de funciones matemáticas.

**memory**: Utilidades relativas a la gestión de memoria, incluyendo asignadores y punteros inteligentes (auto_ptr).

"auto_ptr" es una clase que conforma la librería memory y permite un fácil manejo de punteros y su destrucción automáticamente.

**Librería new**: Manejo de memoria dinámica

**numeric**: Parte de la librería numérica de la STL relativa a operaciones numéricas.

**ostream**: Algoritmos estándar para los flujos de salida.

**queue**: Parte de la STL relativa a contenedores tipo queue (colas de objetos).

**Librería stdio**: Contiene los prototipos de las funciones, macros, y tipos para manipular datos de y salida.

**Librería stdlib**: Contiene los prototipos de las funciones, macros, y tipos para utilidades de uso general.

**string**: Parte de la STL relativa a contenedores tipo string; una generalización de las cadenas alfanuméricas para albergar cadenas de objetos. Muy útil para el fácil uso de las cadenas de caracteres, pues elimina muchas de las dificultades que generan los char

**typeinfo**: Mecanismo de identificación de tipos en tiempo de ejecución

**vector**: Parte de la STL relativa a los contenedores tipo vector; una generalización de las unidimensionales C/C++

**forward_list**: Esta librería es útil para implementar con gran facilidad listas enlazadas simples.

**List**:Permite implementar listas doblemente enlazadas (listas enlazadas dobles) fácilmente.

**Iterator**: Proporciona un conjunto de clases para iterar elementos.

**Regex**: Proporciona fácil acceso al uso de expresiones regulares.

**Thread**: Útil para trabajar programación multihilos y crear múltiples hilos en nuestra aplicación.

# ¿Cuál es la sintaxis para declarar una librería?

Es muy simple realizar la declaración de librerías, en C++, está se debe hacer al principio de todo nuestro código, antes de declarar de cualquier función o línea de código.

Debemos indicarle al compilador que librerías usar, para saber qué términos están correctos en la escritura de nuestro código y cuáles no.

La sintaxis son las siguientes, cualquiera es válida:

- **#include** *<nombre de la librería>*
- **#include** *"nombre de la librería"*

# ¿Qué es “using namespace std”?

Todas las librerías estándar de C++ contienen una declaración del espacio de nombre std, es decir que todas las librerías que hacen parte del estándar de C++ colocan entidades dentro de este espacio de nombre.

Por esta razón cuando declaramos el uso del espacio de nombre std por medio de "using namespace std;", podemos evitar estar escribiendo std::cout o std::cin, etc en nuestro código.

Facilita la escritura de éste al momento de usar las entidades de las librerías estándar. Sin embargo si vamos a hacer uso de una o varias librerías estándar de C++ es recomendable que declaremos el namespace std, para no tener que estar constantemente escribiendo cosas similares a las que puse hace unas líneas como std::cin o similares, dado que únicamente se puede acceder a la entidades de las librerías estándar por medio del espacio nombre std.