# Programación avanzada en C++
# C++11 / 14 / 17

**Los objetivos de este curso son:**

- Conocer las principales novedades que ofrece los estándares C++11, C++14 y C++17, y cómo pueden usarse para mejorar el desarrollo de software.
- Comprender el impacto de las novedades sobre el rendimiento y la facilidad de mantenimiento del software.
- Comprender qué novedades son útiles para desarrolladores de aplicaciones y cuáles para desarrolladores de bibliotecas.
- Comprender las oportunidades que ofrece C++11 para el desarrollo de aplicaciones concurrentes de forma portable, así como sus limitaciones.
- Obtener una visión inicial de los próximos cambios previstos en las revisiones del estándar de C++ (C++20 y especificaciones técnicas).

## Programa del Curso

### Parte I: Lenguaje

#### Generalidades y Sistema de tipos

1. Preprocesador.
2. Aserciones en tiempo de compilación.
3. Nuevos tipos primitivos.
4. Puntero nulo.
5. Caracteres Unicode.
6. Inferencia de tipos.
7. Bucles basados en rango.
8. Nuevos enumerados.
9. Nuevas conversiones contextuales.
10. Separadores de dígitos en literales.
11. Literales binarios.

#### Iniciación

1. Sintaxis uniforme de iniciación.
    1.1. Iniciación no uniforme.
    1.2. Iniciación uniforme.
    1.3. Semántica de iniciación uniforme.
    1.4. Variaciones sintácticas.
    1.5. Iniciación y estrechamiento.
2. Listas de iniciación.
    2.1. Extensión de iniciación uniforme.
    2.2. Constructores de lista de iniciación.
    2.3. Listas de iniciación como parámetro.
    2.4. Listas de iniciación y sobrecarga.
    2.5. Listas de iniciación y auto.

#### Desarrollo de clases

1. Semántica de movimiento.
2. Especificación *noexcept*.
3. Constantes en tiempo de compilación: *constexpr*.
4. Requisitos relajados para *constexpr*.
5. Funciones miembro por defecto y eliminadas.
6. Iniciación de miembros no estáticos.
7. Iniciación de miembros agreagados.
8. Constructores delegados y constructores heredados.
9. Operadores explícitos de conversión.
10. Control de herencia: *override* y *final*.
11. Atributo *[[deprecated]]*.

#### Novedades globales

1. Espacios de nombre en línea.
2. Sintaxis sufija de funciones.
3. *decltype* y *declval*.
4. Deducción de tipo de retorno y *decltype(auto)*.
5. Literales definidos por el usuario.
6. Uniones generalizadas.
7. Alienamiento.
8. Asignación de memoria optimizada.
9. Liberación de memoria con tamaño.

#### Soporte a programación genérica

1. Expresiones lambda.
2. Capturas con iniciación.
3. Expresiones lambda genéricas.
4. Plantillas externas.
5. Alias de plantilla y alias de tipos.
6. Plantillas de variable.
7. Clases locales y argumentos de plantilla.
8. Plantillas con número variable de argumentos.

### PARTE II: La biblioteca estándar

#### Soporte para metaprogramación

1. Funciones sobre tipos.
2. *Type traits* y soporte para metaprogramación.
3. *Transformation traits* mejorados.
4. Evaluación de constantes integrales.
5. Secuencias enteras en tiempo de compilación.

#### Utilidades

1. *Smart pointers*.
2. Tuplas.
3. Acceso a tuplas por tipo.
4. Alias para metafunciones.
5. Evoltorios de funciones.
6. *bind()*.
7. Biblioteca de tiempo.
8. Biblioteca de excepciones y errores del sistema.
9. Literales definidos por el usuario en la biblioteca estándar.

### Mejoras a la STL

1. Contenedores
2. Iteradores.
3. Algoritmos.
4. Algoritmos no modificantes robustos.
5. Objetos función mejorados.
6. Búsqueda heterogénea en contenedores asociativos.

### Tratamiento de cadenas

1. Expresiones regulares.
2. Conversiones de cadenas/numéricas.
3. Entrada/salida delimitada.

## Parte III: Concurrencia

### Introducción a la concurrencia

1. Concurrencia en C++.
2. Hilos.
3. Acceso a datos compartidos.
4. Mecanismos de espera.
5. Futuros y promesas.
6. Tareas asíncronas.

### Modelo de memoria y tipos atómicos

1. Modelo de memoria.
2. Tipos atómicos.
3. Relaciones de ordenación.
4. Modelos de consistencia.
5. Barreras.

### Hilos

1. La clase *thread*.
2. Construcción e identidad.
3. Terminación de hilos.
4. Espacio de nombres *this_thread*.
5. Almacenamiento local al hilo.

### Exclusión mutua

1. Objetos *mutex*.
2. Utilidades para *mutex*.
3. Adquisición múltiple.
4. Variables condición.
5. Cerrojos compartidos.

### Futuros y promesas

1. Promesas.
2. Tareas empaquetadas.
3. Futuros y futuros compartidos.
4. Ejecución asíncrona de tareas.

### Parte IV: Introducción a C++17

1. Cambios menores.
   1.1. Trigrafos.
   1.2. Palabra reservada *register*.
   1.3. Incremento de booleanos.
   1.4. Especificaciones dinámicas de excepciones.
   1.5. Clase *auto_ptr*.
   1.6. Reglas para iniciación.
   1.7. *static_assert.*
2. Aclaraciones.
   2.1. Orden de evaluación.
   2.2. Elusión de copia.
   2.3. Excepciones y punteros a función.
   2.4. Memoria dinámica sobre-alineada.
3. Programación genérica.
   3.1. Deducción de argumentos de plantilla de clase.
   3.2. Deducción de valores en plantillas.
   3.3. Plegamiento de expresiones.
   3.4. *If* en tiempo de compilación.
   3.5. Otros cambios en programación genérica.
4. Atributos.
   4.1. Atributos en C++11/14.
   4.2. Nuevos atributos.
   4.3. Otras modificaciones sobre atributos.
5. Simplificaciones.
   5.1. Vinculación estructurada.
   5.2. Iniciación en *if* y *switch*.
   5.3. Variables *inline*.
6. Algoritmos paralelos.
   6.1. Políticas de ejecución.
   6.2. Visión general.
   6.3. Algoritmos no numéricos.
   6.4. Algoritmos numéricos.
7. Acceso al sistema de ficheros.
8. Nuevos tipos de la biblioteca.
   8.1. Valores generalizados.
   8.2. Valores opcionales.
   8.3. Registros con variantes.
   8.4. Vistas sobre cadenas.