

# Prólogo a la segunda edición

## INTRODUCCIÓN

Esta segunda edición de *Programación en C++. Algoritmos, Estructura de datos y Objetos*, se ha escrito, al igual que la primera edición, pensando en que pudiera servir de referencia y guía de estudio para un primer *curso de introducción a la programación*, con una segunda parte que, a su vez, sirviera como continuación y de *introducción a las estructuras de datos y a la programación orientada a objetos*; todos ellos utilizando C++ como lenguaje de programación. Sin embargo, esta segunda edición ofrece novedades que confiamos mejoren sensiblemente a la primera edición. En particular, se ha tenido en cuenta la opinión de lectores, profesores y maestros, que han utilizado la obra, en su primera edición, como referencia, bien en forma de libro de texto, bien en forma de libro de consulta o simplemente como libro complementario y que sus aportaciones nos ha ayudado considerablemente. Al igual que en la primera edición, el objetivo final que se busca es, no sólo describir la sintaxis de C++, sino, y sobre todo, mostrar las características más sobresalientes del lenguaje a la vez que se enseñan técnicas de programación estructurada y orientada a objetos, pero en este caso utilizando el estándar **ANSI/ISO C++**. Por consiguiente, los objetivos fundamentales son:

- Énfasis fuerte en el análisis, construcción y diseño de programas.
- Un medio de resolución de problemas mediante técnicas de programación.
- Actualización de contenidos al último estándar ANSI/ISO C++, incluyendo las novedades más significativas (tales como espacio de nombres, archivo `iostream`...).
- Tutorial enfocado al lenguaje, mejorado con numerosos ejemplos, ejercicios y herramientas de ayuda al aprendizaje.
- Descripción detallada del lenguaje, con un énfasis especial en técnicas de programación actuales y eficientes.
- Reordenar el contenido en base a la experiencia adquirida en la primera edición.
- Una introducción a la informática, a las ciencias de la computación y a la programación, usando una herramienta de programación denominada C++.

En resumen, éste es un libro diseñado para enseñar a programar utilizando C++, no un libro diseñado para enseñar C++, aunque también pretende conseguirlo. No obstante, confiamos que los estudiantes y autodidactas que utilicen la obra se conviertan de un modo razonable en acérrimos seguidores y adeptos de C++, al igual que nos ocurre a casi todos los programadores que comenzamos a trabajar con este lenguaje. Así, se tratará de enseñar las técnicas clásicas y avanzadas de programación estructurada, junto con técnicas orientadas a objetos. La programación orientada a objetos no es la panacea universal de programador del siglo XXI, pero le ayudará a realizar tareas que, de otra manera, serían complejas y tediosas.

El libro cuenta con una página Web oficial ([www.mhe.es/joyanes](http://www.mhe.es/joyanes)) donde no sólo podrá consultar y descargarse códigos fuente de los programas del libro, sino también apéndices complementarios y cursos o tutoriales de programación en C++, y de C, especialmente pensados para los lectores sin conocimiento de este lenguaje.

## LA EVOLUCIÓN DE C++

El aprendizaje de C++ es una aventura de descubrimientos, en especial, porque el lenguaje se adapta muy bien a diferentes paradigmas de programación, incluyendo entre ellos, *programación orientada a objetos*, *programación genérica* y la tradicional *programación procedimental o estructurada*, tradicional. C++ ha ido evolucionando desde la publicación del libro de Stroustrup *C++. Manual de Referencia con anotaciones* (conocido como ARM y traducido al español por el autor de este libro y por el profesor Miguel Katrib de la Universidad de Cuba) hasta llegar a la actual versión estándar *ISO/ANSI C++ Standard, second edition* (2003) que suelen incorporar casi todos los compiladores y ya muy estabilizada.

Esta edición sigue el estándar ANSI/ISO, aunque en algunos ejemplos y ejercicios, se ha optado por mantener el antiguo estándar, a efectos de compatibilidad, con otras versiones antiguas que pueda utilizar el lector y para aquellos casos en que utilice un compilador no compatible con el estándar, antes mencionado.

Aprenderá muchas características de C++, e incluso las derivadas de C, a destacar:

- Clases y objetos.
- Herencia.
- Polimorfismo y funciones virtuales.
- Sobrecarga de funciones y de operadores.
- Variables referencia.
- Programación genérica, utilizando plantillas (templates) y la Biblioteca Plantillas estándar (STL, Standard Template Library).
- Mecanismo de excepciones para manejar condiciones de error.
- Espacio de nombres para gestionar nombres de funciones, clases y variables.

C++ se comenzó a utilizar como un «*C con clases*» y fue a principios de los ochenta cuando comenzó la revolución C++, aunque su primer uso comercial, por parte de una organización de investigación, comenzó en julio de 1983. Como Stroustrup cuenta en el prólogo de la 3.<sup>a</sup> edición de su citada obra, C++ nació con la idea de que el autor y sus colegas no tuvieran que programar en ensamblador ni en otros lenguajes al uso (véase Pascal, BASIC, FORTRAN...). La explosión del lenguaje en la comunidad informática hizo inevitable la estandarización, proceso que comenzó en 1987 [Stroustrup 94]. Así nació una primera fuente de estandarización, la ya citada obra : *The Annotated C++ Reference Manual* [Ellis 89]<sup>1</sup>. En diciembre de 1989 se reunió el comité X3J16 de ANSI, bajo el auspicio de Hewlett-Packard, y en junio de 1991 se realizó el primer esfuerzo de estandarización internacional de la mano de ISO, y así comenzó a nacer el estándar ANSI/ISO C++. En 1995 se publicó un borrador estándar para su examen público y en noviembre de 1997 fue finalmente aprobado el estándar C++ internacional, aunque fue en 1998 cuando el proceso se pudo dar por terminado (*ANSI/ISO C++ Draft Standard*) conocido como **ISO/IEC 14882: 1998** o simplemente **C++ Estándar** o **ANSI C++**. Stroustrup publicó en 1997 la tercera edición de su libro *The C++ Programming Language* [Stroustrup 97] y en 2000, una actualización que se publicó como *edición especial*<sup>2</sup> (traducida por un equipo de profesores de la Facultad de Informática de la Universidad Pontificia de Salamanca en Madrid, dirigidos por el autor de este libro). El libro que tiene en sus manos, sigue el estándar ANSI/ISO C++.

## COMPILADORES Y COMPILACIÓN DE PROGRAMAS EN C++

Existen numerosos compiladores de C++ en el mercado. Desde ediciones gratuitas y *descargables* a través de Internet hasta profesionales, con costes diferentes, comercializados por diferentes fabricantes. Es difícil dar una recomendación al lector porque casi todos ellos son buenos compiladores, muchos de

<sup>1</sup> Existe versión española de Addison-Wesley Díaz de Santos y traducida por los profesores Manuel Katrib y Luis Joyanes.

<sup>2</sup> Esta obra fue traducida por un equipo de profesores universitarios que dirigió y coordinó el profesor Luis Joyanes, co-autor de esta obra.

ellos con Entornos Integrados de Desarrollo (**EID**). Si usted es estudiante, tal vez la mejor decisión sea utilizar el compilador que le haya propuesto su profesor y que utilice en su Universidad, Instituto Tecnológico o cualquier otro Centro de Formación, donde estudie. Si usted es un lector autodidacta y está aprendiendo por su cuenta existen varias versiones gratuitas que puede descargar desde Internet. Algunos de los más reconocidos son: **Dev-C++** de **Bloodshed** que cumple fielmente el estándar ANSI/ISO C++ (uno de los compiladores utilizado por el autor del libro para editar y compilar los programas incluidos en el mismo) y que corre bajo entornos Windows; **GCC** de GNU que corre bajo los entornos Linux y Unix. Existen muchos otros compiladores gratuitos por lo que tiene donde elegir. Tal vez un consejo más: procure que sea compatible con el estándar ANSI/ISO C++.

Bjarne Stroustrup (creador e inventor de C++) en su página oficial<sup>3</sup> ha publicado el 19 de junio de 2006, «*una lista incompleta de compiladores de C++*», que le recomiendo lea y visite, aunque el mismo reconoce es imposible tener actualizada la lista de compiladores disponibles. Por su interés incluimos a continuación un breve extracto de su lista recomendada:

#### *Compiladores gratuitos*

- Apple C++.
- Bloodshed Dev-C++.
- Borland C++.
- Cygwin (GNU C++).
- Digital Mars C++.
- MINGW - «Minimalist GNU for Windows».
- DJ Delorie's C++ para desarrollo de sistemas DOS/Windows (GNU C++).
- GNU CC fuente.
- Intel C++ para Linux.
- The LLVM Compiler Infrastructure (basado en GCC).
- Microsoft Visual C++ Toolkit 2003.
- Sun Studio.

#### *Compiladores que requieren pago* (algunos permiten descargas gratuitas durante periodos de prueba)

- Borland C++.
- Comeau C++ para múltiples plataformas.
- Compaq C++.
- Green Hills C++ para múltiples plataformas de sistemas empotrados.
- HP C++.
- IBM C++.
- Intel C++ para Windows, Linux, y sistemas empotrados.
- Interstron C++.
- Mentor Graphics/Microtec Research C++ para sistemas empotrados.
- Microsoft C++.
- Paradigm C++, para sistemas empotrados x86.
- The Portland Group C++.
- SGI C++.
- Sun C++.
- WindRiver's Diab C++ utilizado en muchos sistemas empotrados.

---

<sup>3</sup> <http://public.research.att.com/~bs/compilers.html>. El artículo «*An incomplete list of C++ compilers*» lo suele modificar Stroustrup y en la cabecera indica la fecha de modificación. En nuestro caso, consultamos dicha página mientras escribíamos el prólogo en la segunda quincena de junio y la fecha de actualización es «19 de junio de 2006».

Stroustrup recomienda un sitio de compiladores gratuitos de C y C++ (**Compilers.net**: [www.compilers.net/Dir/Free/Compilers/CCpp.htm](http://www.compilers.net/Dir/Free/Compilers/CCpp.htm)).

### **Nota práctica de compatibilidad C++ estándar**

En el artículo antes citado de Stroustrup, recomienda que compile con su compilador, el sencillo programa fuente C++ siguiente. Si usted compila bien este programa, no tendrá problemas con C++ estándar, en caso contrario aconseja buscar otro compilador que sea compatible.

```
#include<iostream>
#include<string>

using namespace std;

int main( )
{
    string s;
    cout << «Por favor introduzca su nombre seguido por Intro \n»;
    cin >> s;
    cout << «Hola, « << s << '\n';
    return 0; // esta sentencia return no es necesaria
}
```

### **Nota práctica sobre compilación de programas fuente con el compilador Dev C++ y otros compiladores compatibles ANSI/ISO C++**

Muchos programas del libro han sido compilados y ejecutados en el compilador de *freeware* (de libre distribución) BloodShed **DEV-C++**<sup>4</sup>, versión 4.9.9.2. Dev C++ es un editor de múltiples ventanas integrado con un compilador de fácil uso que permite la compilación, enlace y ejecución de programas C o C++.

Las sentencias **system(«PAUSE»);** y **return EXIT\_SUCCES;** son incluidas por defecto por el entorno **Dev** en todos los programas escritos en C++. A efectos de compatibilidad práctica, si usted no utiliza el compilador Dev C++ o no desea que en sus listados aparezca estas sentencias, puede sustituirlas bien por otras equivalentes o bien quitar las dos y sustituirlas por **return 0;** como recomienda Stroustrup para terminar sus programas y que es el método habitualmente empleado en el libro.

**system(«PAUSE»);** detiene la ejecución del programa hasta que se pulsa una tecla. Las dos sentencias siguientes de C++ producen el mismo efecto:

```
cout << «Presione ENTER para terminar»;
cint.get();

return EXIT_SUCCES; devuelve el estado de terminación correcta del programa al sistema operativo. La siguiente sentencia de C++ estándar, produce el mismo efecto:

return 0;
```

<sup>4</sup> <http://www.bloodshed.net/>.

Si en el compilador con el cual trabaja no le funciona alguna de las sentencias anteriores puede sustituirlas por las indicadas en la Tabla P.1.

**Tabla P.1. Dev C++ versus otros compiladores de C++ (en compilación)**

Sustitución de sentencias	
DEV C++	C++
<code>system(«PAUSE»);</code>	<code>cout &lt;&lt; «Presione ENTER para terminar»; cint.get();</code>
<code>return EXIT_SUCCES;</code>	<code>return 0;</code>

## OBJETIVOS DEL LIBRO

C++ es un superconjunto de C y su mejor extensión. Este es un tópico conocido por toda la comunidad de programadores del mundo. Cabe preguntarse como hacen muchos autores, profesores, alumnos y profesionales: ¿se debe aprender primero C y luego C++? Stroustrup y una gran mayoría de programadores contestan así: No sólo es innecesario aprender primero C, sino que además es una mala idea. Nosotros no somos tan radicales y pensamos que se puede llegar a C++ procediendo de ambos caminos, aunque es lógico la consideración citada anteriormente, ya que efectivamente los hábitos de programación estructurada de C pueden retrasar la adquisición de los conceptos clave de C++, pero también es cierto que en muchas casos ayuda considerablemente en el aprendizaje.

Este libro supone que el lector no es programador de C, ni de ningún otro lenguaje, aunque también somos conscientes de que el lector que haya seguido un primer curso de programación en algoritmos o en algún lenguaje estructurado, llámese Pascal o cualquier otro, éste le ayudará favorablemente al correcto y rápido aprendizaje de la programación en C++ y obtendrá el máximo rendimiento de esta obra. Sin embargo, si ya conoce C, naturalmente no tendrá ningún problema en su aprendizaje, muy al contrario, bastará que lea con detalle las diferencias esenciales de los primeros capítulos (en caso de duda puede también consultar la página oficial del libro donde podrá encontrar numerosa documentación sobre C), de modo que irá integrando gradualmente los nuevos conceptos que irá encontrando a medida que avance en la obra con los conceptos clásicos de C. El libro pretende enseñar a programar utilizando tres conceptos fundamentales:

1. *Algoritmos* (conjunto de instrucciones programadas para resolver una tarea específica).
2. *Datos y Estructuras de Datos* (una colección de datos que se proporcionan a los algoritmos que se han de ejecutar para encontrar una solución: los datos se organizarán en *estructuras de datos*).
3. *Objetos* (el conjunto de datos y algoritmos que los manipulan, encapsulados en un tipo de dato nuevo conocido como objeto).

Los dos primeros aspectos, **algoritmos** y **datos**, han permanecido invariables a lo largo de la corta historia de la informática/computación, pero la interrelación entre ellos sí que ha variado y continuará haciéndolo. Esta interrelación se conoce como *paradigma de programación*.

En el paradigma de programación procedimental (*procedural* o por procedimientos) un problema se modela directamente mediante un conjunto de algoritmos. Por ejemplo, la nómina de una empresa o la gestión de ventas de un almacén, se representa como una serie de procedimientos que manipulan datos. Los datos se almacenan separadamente y se accede a ellos o bien mediante una posición global o mediante parámetros en los procedimientos. Tres lenguajes de programación clásicos, FORTRAN, Pascal y C, han representado el arquetipo de la programación *procedimental*, también relacionada estrechamente y, a veces, conocida como *programación estructurada*. La programación con soporte en C++, propor-

ciona el paradigma *procedimental* con un énfasis en funciones, plantillas de funciones y algoritmos genéricos.

En la década de los ochenta, el enfoque del diseño de programas se desplazó desde el paradigma *procedimental* al orientado a objetos apoyado en los tipos abstractos de datos (TAD). En este paradigma se modela un conjunto de abstracciones de datos. En C++ estas abstracciones se conocen como **clases**. Las clases contienen un conjunto de instancias o ejemplares de la misma que se denominan objetos, de modo que un programa actúa como un conjunto de objetos que se relacionan entre sí. La gran diferencia entre ambos paradigmas reside en el hecho de que los algoritmos asociados con cada clase se conocen como *interfaz pública* de la clase y los datos se almacenan privadamente dentro de cada objeto de modo que el acceso a los datos está oculto al programa general y se gestionan a través de la interfaz.

C++ es un lenguaje *multiparadigma*. Soporta ambos tipos de paradigmas. La gran ventaja es que se puede proporcionar la solución que mejor; resuelva cada problema con el paradigma más adecuado, dado que ningún paradigma resuelve definitivamente todos los problemas. El inconveniente es que el lenguaje se vuelve más grande y complejo, pero este inconveniente está quedando prácticamente resuelto por el citado proceso de estandarización, internacional a que ha sido sometido C++, lo que ha conseguido que esa implícita dificultad se haya convertido en facilidad de uso a medida que se controla y domina el lenguaje. También los fabricantes de compiladores han contribuido a ello y al núcleo fundamental del lenguaje le han añadido una serie de bibliotecas de funciones y de plantillas (tal como **STL**) que han simplificado notablemente las tareas de programación y han facilitado de sobremanera que éstas sean muy eficientes.

Así pues, en resumen, los objetivos fundamentales de esta obra son: *introducción a la programación estructurada, estructuras de datos y programación orientada a objetos* con el lenguaje estándar **ANSI/ISO C++**.

## EL LIBRO COMO HERRAMIENTA DOCENTE

La experiencia del autor desde hace muchos años con obras muy implantadas en el mundo universitario como *Programación en C++* (1.ª edición), *Fundamentos de programación* (en su 3.ª edición), *Programación en C* (en su 2.ª edición), *Programación en Pascal* (en su 4.ª edición), y *Programación en BASIC* (que alcanzó tres ediciones y numerosísimas reimpresiones en la década de los ochenta) nos ha llevado a mantener la estructura de esta obra, actualizándola a los contenidos que se prevén para los estudiantes del actual siglo XXI. Por ello en el contenido de la obra hemos tenido en cuenta no sólo las directrices de los planes de estudio españoles de ingeniería informática e ingeniería técnica en informática de sistemas y de gestión, y licenciaturas en ciencias de la computación, matemáticas, físicas..., sino también de ingenierías tales como industriales, telecomunicaciones, agrónomos o geodesia. Nuestro conocimiento del mundo educativo latinoamericano nos ha llevado a pensar también en las carreras de ingeniería de sistemas computacionales y las licenciaturas en informática y en sistemas de información, como se las conoce en aquel continente americano.

Por todo lo anterior, el contenido del libro intenta seguir un programa estándar de un primer curso de introducción a la programación y, según situaciones, un segundo curso de programación de nivel medio, en asignaturas tales como *Metodología de la programación*, *Fundamentos de programación*, *Introducción a la programación...* Asimismo, se ha buscado seguir las directrices emanadas de la ACM para curricula actuales y las vigentes en universidades latinoamericanas, muchas de las cuales conocemos y con las que tenemos relaciones profesionales.

El contenido del libro abarca los citados programas y comienza con la introducción a la computación y a la programación, para llegar a estructuras de datos y objetos. Por esta circunstancia la estructura del curso no ha de ser secuencial en su totalidad sino que el profesor/maestro y el alumno/lector podrán estudiar sus materias en el orden que consideren más oportuno. Esta es la razón principal por la cual el libro se ha organizado en cuatro partes con numerosos apéndices incluidos en la página *web* oficial del mismo, con el objeto de que el lector seleccione y se «baje» aquellos apéndices que considere de su interés, y de este modo no incrementar el número de páginas de la obra.

Se trata de describir los dos paradigmas más populares en el mundo de la programación: el *procedimental* y el *orientado a objetos*. Los cursos de programación en sus niveles inicial y medio están evolucionando para aprovechar las ventajas de nuevas y futuras tendencias en ingeniería de software y en diseño de lenguajes de programación, específicamente diseño y programación orientada a objetos. Algunas facultades y escuelas de ingenieros, junto con la nueva formación profesional (ciclos formativos de nivel superior) en España y en Latinoamérica, han introducido a sus alumnos en la programación orientada a objetos, inmediatamente después del conocimiento de la programación estructurada, e incluso —en ocasiones— antes o en paralelo. Por esta razón, una metodología que se podría seguir sería impartir un curso de fundamentos de programación seguido de estructuras de datos y luego seguir con un segundo nivel de programación avanzada y programación orientada a objetos que constituyen las cuatro partes del libro. Pensando en aquellos alumnos que necesiten profundizar en los temas tratados en el capítulo 1, se han escrito los Apéndices 1 y 2 y «subido» a la página web del libro con una ampliación de Introducción a las computadoras y una Introducción a la Programación con el lenguaje algorítmico, pseudocódigo, y ejemplos en C y C++.

## CARACTERÍSTICAS IMPORTANTES DEL LIBRO

**Programación en C++** utiliza los siguientes elementos clave para conseguir obtener el mayor rendimiento del material incluido en sus diferentes capítulos:

**Contenido.** Enumera los apartados descritos en el capítulo.

**Introducción.** Abre el capítulo con una breve revisión de los puntos y objetivos más importantes que se tratarán y todo aquello que se puede esperar del mismo.

**Conceptos clave.** Enumera los términos informáticos y de programación más notables que se tratarán en el capítulo.

**Descripción del capítulo.** Explicación usual de los apartados correspondientes del capítulo. En cada capítulo se incluyen ejemplos y ejercicios resueltos. Los listados de los programas completos o parciales se escriben en letra Courier con la finalidad principal de que puedan ser identificados fácilmente por el lector.

**Resumen del capítulo.** Revisa los temas importantes que los estudiantes y lectores deben comprender y recordar. Busca también ayudar a reforzar los conceptos clave que se han aprendido en el capítulo.

**Ejercicios.** Al final de cada capítulo se proporciona a los lectores una lista de ejercicios sencillos de modo que le sirvan de oportunidad para que puedan medir el avance experimentado mientras leen y siguen —en su caso— las explicaciones del profesor relativas al capítulo.

**Problemas.** Después del apartado Ejercicios, se añaden una serie de actividades y proyectos de programación que se le proponen al lector como tarea complementaria de los ejercicios y de un nivel de dificultad algo mayor.

**Ejercicios resueltos y problemas resueltos.** En estas secciones, el lector encontrará enunciados de ejercicios y problemas complementarios, cuyas soluciones encontrará en las dos fuentes siguientes:

1. **Programación en C++: Un enfoque práctico.** Madrid: McGraw-Hill, Colección *Schaum*, 2006, de Luis Joyanes y Lucas Sánchez.
2. Portal del libro: [www.mhe.es/joyanes](http://www.mhe.es/joyanes).

El libro *Programación en C++: Un enfoque práctico*, perteneciente a la prestigiosa Colección *Schaum*, de libros prácticos, ha sido escrito por el autor del libro y el profesor Lucas Sánchez, como un libro eminentemente práctico y complementario de éste que tiene usted en sus manos, por lo cual con-

tiene un gran número de ejercicios y problemas propuestos y resueltos. Los códigos fuente de los *Ejercicios resueltos* y *Problemas resueltos* de los 20 primeros capítulos de este libro los encontrará usted en las dos fuentes citadas anteriormente. De este modo, el lector que lo desee podrá verificar y comprobar su propia solución con la solución planteada en el citado libro de la colección Shaum o bien en la página web del libro.

**Recuadro.** Conceptos importantes que el lector debe considerar durante el desarrollo del capítulo.

**Consejo.** Ideas, sugerencias, recomendaciones... al lector, con el objetivo de obtener el mayor rendimiento posible del lenguaje y de la programación.

**Precaución.** Advertencia al lector para que tenga cuidado al hacer uso de los conceptos incluidos en el recuadro adjunto.

**Reglas.** Normas o ideas que el lector debe seguir preferentemente en el diseño y construcción de sus programas.

## ORGANIZACIÓN DEL LIBRO

El libro se divide en cuatro partes que unidas constituyen un curso completo de programación en C++. Dado que el conocimiento es acumulativo, los primeros capítulos proporcionan el fundamento conceptual para la comprensión y aprendizaje de C++ y una guía a los estudiantes a través de ejemplos y ejercicios sencillos, y los capítulos posteriores presentan de modo progresivo la programación en C++ en detalle, tanto en el paradigma procedimental como en el orientado a objetos. Los apéndices contienen un conjunto de temas importantes que incluyen desde guías de sintaxis de ANSI/ISO C++ hasta un glosario de términos o una biblioteca de funciones y clases, junto con una extensa bibliografía de algoritmos, estructura de datos, programación orientada a objetos y una amplia lista de sitios de Internet (URL) donde el lector podrá complementar, ampliar y profundizar en el mundo de la programación y en la introducción a la ingeniería de software.

### Parte I. Fundamentos de programación en C++

Esta parte es un primer curso de programación para alumnos principiantes en asignaturas de introducción a la programación. Esta parte sirve tanto para cursos de C++ como de C (en este caso con la ayuda de los Apéndices A y B). Esta parte comienza con una introducción a la informática y a las ciencias de la computación como a la programación. Describe los elementos básicos constitutivos de un programa y las herramientas de programación utilizadas tales como algoritmos, diagramas de flujo, etc. Asimismo se incluye un curso del lenguaje C++ y técnicas de programación que deberá emplear el lector en su aprendizaje de programación.

**Capítulo 1. *Introducción a la ciencia de la computación y a la programación.*** Proporciona una revisión de las características más importantes necesarias para seguir bien un curso de programación básico y avanzado en C++. Para ello se describe la organización física de una computadora junto con el concepto de algoritmo y los métodos más eficientes para la resolución de problemas con computadora. Se explican también los diferentes tipos de programación y una breve historia del lenguaje C++.

**Capítulo 2. *El lenguaje C++. Elementos básicos.*** Enseña los fundamentos de C++, organización y estructura general de un programa, función `main()`, ejecución, depuración y prueba de un programa, elementos de un programa, tipos de datos (el tipo de dato `bool`), constantes, variables y entradas/salidas de datos (`cin` y `cout`).

**Capítulo 3. *Operadores y expresiones.*** Se describen los conceptos y tipos de operadores y expresiones, conversiones y precedencias. Se destacan operadores especiales tales como manipulación

de bits, condicional, `sizeof`, `()`, `[]`, `::`, coma, etc., y se analizan los conceptos de conversión de tipos, prioridad y asociatividad entre operadores.

- Capítulo 4. Estructuras de selección: sentencias *if* y *switch*.** Introduce al concepto de estructura de control y, en particular, estructuras de selección, tales como `if`, `if-else`, `case` y `switch`. Expresiones condicionales con el operador `?:`, evaluación en cortocircuito de expresiones lógicas, errores frecuentes de programación y puesta a punto de programas.
- Capítulo 5. Estructuras repetitivas: bucles (*for*, *while* y *do-while*).** El capítulo introduce las estructuras repetitivas (`for`, `while` y `do-while`). Examina la repetición (iteración) de sentencias en detalle y compara los bucles controlados por centinela, bandera, etc. Explica precauciones y reglas de uso de diseño de bucles. Compara los tres diferentes tipos de bucles, así como el concepto de bucles anidados.
- Capítulo 6. Funciones.** Examina el diseño y construcción de módulos de programas mediante funciones. Se define la estructura de una función, prototipos y parámetros. El concepto de funciones en línea (*inline*), uso de bibliotecas de funciones, clases de almacenamiento, ámbitos, visibilidad de una función. Asimismo se introduce el concepto de recursividad y plantillas de funciones.
- Capítulo 7. Arrays/ Arreglos (listas y tablas).** Examina la estructuración de los datos en *arrays* (*arreglos*) o grupos de elementos dato del mismo tipo. El capítulo presenta numerosos ejemplos de arrays de uno, dos o múltiples índices. Se realiza una introducción a los algoritmos de ordenación y búsqueda de elementos en una lista.
- Capítulo 8. Estructuras y uniones.** Conceptos de estructuras, declaración, definición, iniciación, uso y tamaño. Acceso a estructuras. *Arrays* (arreglos) de estructuras y estructuras anidadas. Uniones y enumeraciones.
- Capítulo 9. Punteros (apuntadores).** Presenta una de las características más potentes y eficientes del lenguaje C++, los punteros. Este capítulo proporciona explicación detallada de los punteros, *arrays* de punteros, punteros de cadena, aritmética de punteros, punteros constantes, punteros como argumentos de funciones, punteros a funciones y a estructuras.
- Capítulo 10. Asignación dinámica de memoria.** En este capítulo se describe la gestión dinámica de la memoria junto con los operadores `new` y `delete`. Se dan reglas de funcionamiento de ambos operadores y se describe el concepto de arrays dinámicos y asignación de memoria para *arrays*. Se examinan los tipos de memoria en C++ así como los modos de asignación y liberación de memoria.
- Capítulo 11. Cadenas.** Se examina el concepto de cadena (`string`) así como las relaciones entre punteros, *arrays* y cadenas en C++. Se introducen conceptos básicos de manipulación de cadenas junto con operaciones básicas tales como longitud, concatenación, comparación, conversión y búsqueda de caracteres y cadenas.
- Capítulo 12. Ordenación y búsqueda.** Dos de las operaciones más importantes que se realizan en algoritmos y programas de cualquier entidad y complejidad son: ordenación de elementos de una lista o tabla y búsqueda de un elemento en dichas listas o tablas. Los métodos básicos más usuales de búsqueda y ordenación se describen en el capítulo. Asimismo, se explica el concepto de análisis de algoritmos de ordenación y búsqueda.

## Parte II. Programación orientada a objetos

En esta parte se describen las propiedades fundamentales de la programación orientada a objetos y los métodos de implementación de las mismas con el lenguaje C++. Así, se estudian y analizan: *clases y objetos*, *clases derivadas* y *herencia*, *polimorfismo* y la *genericidad* (*plantillas*, *templates*), soporte también del paradigma de programación genérica. Los tres capítulos que constituyen esta parte, se han diseñado como una introducción a la programación orientada a objetos; los Capítulos 21 y 22 que tratan sobre «sobrecarga de operadores» y «excepciones» están directamente relacionados con los tres capítulos de esta parte, ya que también son propiedades características del citado paradigma de programación.

- Capítulo 13. Clases y objetos.** Este capítulo muestra la forma de implementar la abstracción de datos, mediante tipos abstractos de datos, clases. Se describen los conceptos de clase y objeto, así como el sistema para definición de una clase. El capítulo examina el método de inicialización de objetos, junto con el concepto de constructores y destructores de una clase.
- Capítulo 14. Clases derivadas: herencia y polimorfismo.** Dos de las propiedades más importantes de la programación orientada a objetos son: *herencia* y *polimorfismo*. La herencia es un sistema de *reusabilidad* de software en el que nuevas clases se desarrollan rápida y fácilmente a partir de las clases existentes y añadiendo nuevas propiedades a las mismas. Este capítulo examina las nociones de clases base y clases derivadas, herencia *protegida*, *pública* y *privada* (`protected`, `public`, `private`), constructores y destructores en clases base y derivadas. Se describen los diferentes tipos de herencia: simple y múltiple. El polimorfismo y la ligadura dinámica se describen también a lo largo del capítulo.
- Capítulo 15. Genericidad: plantillas (templales).** Examina una de las incorporaciones más importantes de la evolución del lenguaje C++: las plantillas (*templates*). Se describen el concepto y la definición de las plantillas de funciones. Las plantillas de clases denominadas tipos *parametrizados* permiten definir tipos genéricos tales como una cola de enteros, una cola de reales (`float`), una cola de cadenas, etc. Se presenta una aplicación práctica y se realiza una comparativa de las plantillas y el polimorfismo.

### Parte III. Estructuras de datos

- Capítulo 16. Flujos y archivos: biblioteca estándar E/S.** Contiene una descripción abierta del tratamiento del nuevo estilo orientado a objetos de entrada/salida en C++. Este capítulo examina las diversas características de E/S de C++ que incluye la salida con el operador de inserción de flujos, la entrada con el operador de extracción de flujos, E/S con seguridad de tipos. Se analizan el uso y aplicación de los manipuladores e indicadores de formato. El concepto de archivo junto con su definición e implementación es motivo de estudio en este capítulo. Las operaciones usuales se estudian con detenimiento.
- Capítulo 17. Listas enlazadas.** Una lista enlazada es una estructura de datos que mantiene una colección de elementos, pero el número de ellos no se conoce por anticipado o varía en un amplio rango. La lista enlazada se compone de elementos que contienen un valor y un puntero. El capítulo describe los fundamentos teóricos y las operaciones que se pueden realizar en la lista enlazada. También se describen los distintos tipos de listas enlazadas.
- Capítulo 18. Pilas y colas.** Las ideas abstractas de pila y cola se describen en el capítulo. Pilas y colas se pueden implementar de diferentes maneras, bien con vectores (*arrays*) o con listas enlazadas.
- Capítulo 19. Recursividad.** El importante concepto de recursividad (propiedad de una función de llamarse a sí misma) se introduce en el capítulo junto con algoritmos complejos de ordenación y búsqueda en los que además se estudia su eficiencia. Entre los algoritmos importantes se explica el método de ordenación rápida *QuickSort*.
- Capítulo 20. Árboles.** Los árboles son otro tipo de estructura de datos dinámica y no lineal. Las operaciones básicas en los árboles junto con sus operaciones fundamentales se estudian en el capítulo. Se describen también los árboles binarios y árboles binarios de búsqueda como elementos clave en el diseño y construcción de estructura de datos complejas.

### Parte IV. Programación avanzada

En esta parte del libro se describen dos características importantes de C++ -también existen en otros lenguajes de programación orientados a objetos como Java y Ada- que se pueden considerar como propiedades de orientación a objetos o como propiedades complementarias, pero que son de gran uso en

programación avanzada y cuya correcta aplicación por el programador le permitirá diseñar y construir programas muy eficientes.

**Capítulo 21. Sobrecarga de operadores.** Es una de las características más populares de cualquier curso de programación en C++. La sobrecarga de operadores permite al programador indicar al compilador cómo utilizar operadores existentes con objetos de tipos nuevos. C++ utiliza la sobrecarga con tipos incorporados tales como enteros, reales y carácter. Un ejemplo típico es la concatenación de cadenas mediante el operador «+» que une una cadena a continuación de otra.

**Capítulo 22 Excepciones.** Se examina en este capítulo una de las mejoras más sobresalientes del lenguaje C++. El manejo de excepciones permite al programador escribir programas que sean más robustos, más tolerantes a fallos y más adecuados a entornos de misión y negocios críticos. El capítulo introduce a los fundamentos básicos del manejo de excepciones con bloques `try`, sentencias `throw` y bloques `catch`; indica cómo y cuándo se vuelve a lanzar una excepción y se incluyen aplicaciones prácticas de manejo de excepciones.

## APÉNDICES INCLUIDOS EN LA WEB (página oficial del libro: [www.mhe.es/joyanes](http://www.mhe.es/joyanes))

En todos los libros dedicados a la enseñanza y aprendizaje de técnicas de programación es frecuente incluir apéndices de temas complementarios a los explicados en los capítulos anteriores. Estos apéndices sirven de guía y referencia de elementos importantes del lenguaje y de la programación de computadoras. En esta edición se ha optado por colocarlos en la Web, de modo que el lector pueda «bajarlos» de ella cuando lo considere oportuno y en función de su lectura y aprendizaje.

**Apéndice A. C++ frente a C.** Estudio extenso de comparación de características de los lenguajes C y C++ cuyo objetivo es facilitar la migración de un lenguaje a otro con facilidad.

**Apéndice B. Guía de sintaxis de ANSI/ISO C++.** Descripción detallada de los elementos fundamentales del estándar C++.

**Apéndice C. Operadores (prioridad).** Tabla que contiene todos los operadores y el orden de prioridad en las operaciones cuando aparecen en expresiones.

**Apéndice D. Código de caracteres ASCII.** Listado del juego de caracteres del código ASCII utilizado en la actualidad en la mayoría de las computadoras.

**Apéndice E.** Listado por orden alfabético de las palabras reservadas en ANSI/ISO C++, al estilo del diccionario. Definición y uso de cada palabra reservada, con ejemplos sencillos de aplicación.

**Apéndice F. Biblioteca defunciones estándar ANSI/ISO C++.** Diccionario alfabético de las funciones estándar de la biblioteca estándar de ANSI/ISO C++, con indicación de la sintaxis del prototipo de cada función. Una descripción de su misión, junto con algunos ejemplos sencillos de la misma.

**Apéndice G. Biblioteca de clases de ANSI/ISO C++.** Diccionario de clases correspondientes a la biblioteca de clases estándar de C++.

**Apéndice H. Glosario.** Minidiccionario de términos importantes de programación en inglés, con su traducción al español y una breve descripción de los mismos.

**Apéndice I. Recursos (Libros / Revistas / URLs de la Web sobre C++).** Colección de recursos Web de interés para el lector, tanto para su etapa de aprendizaje de la programación en C++, como para su etapa profesional de programador.

**Apéndice J. Bibliografía.** Enumeración de los libros más sobresalientes empleados por el autor en la escritura de esta obra, así como otras obras importantes complementarias que ayuden al lector que desee profundizar o ampliar aquellos conceptos que considere necesario conocer con más detenimiento.

Pensando en los lectores que deseen profundizar en los conceptos introductorios explicados en el Capítulo 1 «Introducción a la computación y a la programación» se han colocado en el portal web del

libro dos apéndices complementarios a modo de pequeños cursos teórico-práctico de *introducción a las computadoras, algoritmos y programación*, con la ayuda de un lenguaje algorítmico o pseudocódigo.

**Apéndice 1.** *Introducción a las computadoras y a los lenguajes de programación.*

**Apéndice 2.** *Metodología de la programación y desarrollo de software.*

## AGRADECIMIENTOS EN LA PRIMERA EDICIÓN

Un libro nunca es fruto único del autor, sobre todo si el libro está concebido como libro de texto y autoaprendizaje, y pretende llegar a lectores y estudiantes de informática y de computación, y, en general, de ciencias e ingeniería, así como a autodidactas en asignaturas tales como programación (introducción, fundamentos, avanzada, etc.). Esta obra no es una excepción a la regla y son muchas las personas que me han ayudado a terminarla. En primer lugar, mis colegas de la Universidad Pontificia de Salamanca en el campus de Madrid, y en particular del Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software de la misma, que desde hace muchos años me ayudan y colaboran en la impartición de las diferentes asignaturas del departamento y, sobre todo, en la elaboración de los programas y planes de estudio de las mismas. A todos ellos les agradezco públicamente su apoyo y ayuda.

En especial deseo agradecer la revisión, comentarios, ideas, consejos y sugerencias, que sobre este libro, en particular, me han dado los siguientes profesores de universidades españolas:

**Ignacio Zahonero Martínez**, profesor de la Facultad de Informática de la Universidad Pontificia de Salamanca en el campus de Madrid.

**María Luisa Diez Platas**, profesora de la Facultad de Informática de la Universidad Pontificia de Salamanca en el campus de Madrid.

**Eduardo González Joyanes**, profesor de la Facultad de Ingeniería de la Universidad Católica de Ávila.

También he de agradecer a otros profesores su cuidada revisión de algunos capítulos de esta obra, así como la aportación de algunos gráficos y dibujos:

**Sergio Ríos Aguilar**, profesor de la Facultad de Informática de la Universidad Pontificia de Salamanca en el campus de Madrid.

**Francisco Mata Mata**, profesor de la Escuela Politécnica Superior de la Universidad de Jaén.

**Luis Rodríguez Baena**, profesor de la Facultad de Informática de la Universidad Pontificia de Salamanca en el campus de Madrid.

A los restantes profesores del grupo de Tecnologías Orientadas a Objetos de nuestra Facultad que ponen en marcha todas las asignaturas relacionadas con estas tecnologías: Héctor Castan Rodríguez, Salvador Sánchez Sánchez, Paloma Centenera Centenera, Francisco Morales Arcía, Rosa Hernández Prieto, Miguel Ángel Sicilia Urbán, María Borrego, Víctor Martín García, Rafael Ojeda Martín, Andrés Castillo y Antonio Reus Hungría.

Muchos otros profesores han ayudado en la concepción y realización de esta obra, de una u otra manera, apoyando con su continua colaboración y sugerencia de ideas la puesta en marcha de asignaturas del área de programación; con el riesgo de olvidar a alguno y pidiéndole, por anticipado, mi involuntaria omisión, he de citar de modo especial a los siguientes compañeros —y sin embargo amigos— en el Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software de la Facultad de Informática y Escuela Universitaria de Informática de la Universidad Pontificia de Salamanca en el campus de Madrid: Isabel Torralvo, Mercedes Vargas, Luis Villar, Óscar San Juan, Yago Sáez, Antonio Muñoz, Mónica Vázquez, Angela Carrasco, Matilde Fernández, Juan Antonio Riesco, Joaquín Aveger, Lucas Parra y Miguel Sánchez

Además de estos compañeros en docencia, no puedo dejar de agradecer, una vez más, a mi editora —y sin embargo amiga— **Concha Fernández Madrid**, las constantes muestras de afecto y comprensión que siempre tiene, y ésta no ha sido una excepción, hacia mi persona y mi obra. Sus continuos consejos,

sugerencias y recomendaciones, siempre son acertados y, además, fáciles de seguir; por si no fuera suficiente, siempre benefician a la obra.

Mi eterno agradecimiento a todas las personas anteriores. Naturalmente, no puedo dejar de agradecer a mis numerosos alumnos, estudiantes y lectores, en general, españoles y latinoamericanos, que, continuamente, me aconsejan, critican y me proporcionan ideas para mejoras continuas a mis libros. Sin todo lo que he aprendido, sigo aprendiendo y seguiré aprendiendo de ellos y su aliento continuo, sería prácticamente imposible para mí terminar nuevas obras y, en especial, este libro. También deseo expresar mi agradecimiento a tantos y tantos colegas de universidades españolas y latinoamericanas que apoyan mi labor docente y editorial.

Mi reconocimiento y agradecimiento eterno a todos: alumnos, lectores, colegas, profesores, maestros, monitores y editores. Gracias por vuestra ayuda.

En Carchelejo, Jaén (Andalucía), verano de 1999.

## AGRADECIMIENTOS EN LA SEGUNDA EDICIÓN

A mis compañeros del Departamento de Lenguajes y Sistemas Informáticos de la Facultad de Informática y Escuela Universitaria de Informática de la Universidad Pontificia de Salamanca en el *campus* de Madrid, que revisaron las primeras pruebas de esta obra y me dieron consejos sobre las mismas:

- **Ignacio Zahonero Martínez.**
- **Lucas Sánchez García.**
- **Matilde Fernández Azuela.**

Gracias, compañeros, y sin embargo, amigos.

A mi editor y amigo, **Carmelo Sánchez**, que una vez más, me ha aconsejado, me ha revisado y me ha acompañado con sus buenas artes, costumbres y su profesionalidad, en todo el camino que ha durado la edición de este libro.

De un modo muy especial y con mi agradecimiento eterno, a mis lectores, a los estudiantes de España y de Latinoamérica, que han estudiado o consultado la primera edición de esta obra, a mis colegas —profesores y maestros— de España y Latinoamérica que han tenido la amabilidad de consultar o seguir su contenido en sus clases y a todos aquellos que me han dado consejos, sugerencias, propuestas y revisiones. Mi reconocimiento más sincero y de nuevo «mi agradecimiento eterno»; son el aliento diario que me permite continuar con esta hermosa tarea que es la comunicación con todos ellos. Gracias.

En Carchelejo (Sierra Mágina), Andalucía (España), julio de 2006.

