

Prólogo

C++ es heredero directo del lenguaje C que a su vez se deriva del lenguaje B. El lenguaje de programación C fue desarrollado por **Dennis Ritchie** de AT&T Bell Laboratories y se utilizó para escribir y mantener el sistema operativo UNIX. C es un lenguaje de propósito general que se puede utilizar para escribir cualquier tipo de programa, pero su éxito y popularidad está especialmente relacionado con el sistema operativo UNIX. Los sistemas operativos son los programas que gestionan (administran) los *recursos de la computadora*. Ejemplos bien conocidos de sistemas operativos además de UNIX son MS/DOS, OS Mac, OS/2, MVS, Linux, Windows 95/98, NT, 2000, XP 2000, o el recientemente presentado **Vista** de Microsoft que vendrá a sustituir al actual Windows XP.

La especificación formal del lenguaje C es un documento escrito por Ritchie, titulado *The C Reference Manual*. En 1997, **Ritchie** y **Brian Kernighan**, ampliaron ese documento y publicaron un libro referencia del lenguaje *The C Programming Language* (también conocido por el K&R). Aunque C es un lenguaje muy potente y sigue siendo muy utilizado en el mundo universitario y también en el profesional, el hecho de haber sido diseñado al principio de los setenta y que la naturaleza de la programación ha cambiado radicalmente en la década de los ochenta y de los noventa, exigía una actualización para subsanar sus “deficiencias”.

Bjarne Stroustrup de AT&T Bell Laboratories desarrolló C++ al principio de la década de los ochenta. Stroustrup diseñó C++ como un mejor C. En general, C estándar es un subconjunto de C++ y la mayoría de los programas C son también programas C++ (la afirmación inversa no es verdadera). C++ además de añadir propiedades a C, presenta características y propiedades de *programación orientada a objetos*, que es una técnica de programación muy potente y que se verá en la segunda parte de este libro.

Se han presentado varias versiones de C++ y su evolución ha incorporado nuevas y potentes propiedades: herencia, genericidad, plantillas, funciones virtuales, excepciones, etc. C++ ha ido evolucionando año a año y como su autor ha explicado: «*evolucionó siempre para resolver problemas encontrados por los usuarios y como consecuencia de conversaciones entre el autor, sus amigos y sus colegas*»¹.

C++ comenzó su proyecto de estandarización ante el comité ANSI y su primera referencia es *The Annotated C++ Reference Manual* [Ellis 89]². En diciembre 1989 se reunió el comité X3J16 del ANSI por iniciativa de Hewlett Packard. En junio de 1991, la estandarización de ANSI pasó a formar parte de un esfuerzo de estandarización ISO. En 1995 se publicó un borrador de estándar para su examen y en 1998 se aprobó el estándar C++ internacional por las organizaciones ANSI e ISO, conocido como **ISO/IEC 14882** o simplemente **C++ Estándar** o **ANSI C++**. Stroustrup publicó en 1997 la tercera edición de su libro *The C++ Programming Language* y, en 1998, una actualización que se publicó como *edición especial*³. El libro que tiene en sus manos, sigue el estándar ANSI/ISO C++.

C++ en el aprendizaje de algoritmos y programación orientada a objetos

Todas las carreras universitarias de *Ciencias e Ingeniería*, así como los estudios de *Formación Profesional* (sobre todo en España los ciclos superiores) requieren un curso básico de *algoritmos y de programación* con un lenguaje potente y profesional pero que sea simple y fácil de utilizar, así como un curso de *programación orientada a objetos*.

¹ [Stroustrup 98], p. 12

² Existe versión española de Addison-Wesley/Díaz de Santos y traducida por los profesores Manuel Katrib y Luis Joyanes.

³ Esta obra fue traducida por un equipo de profesores universitarios que dirigió y coordinó el profesor Luis Joyanes, coautor de esta obra.

Los doce primeros capítulos constituyen un curso inicial de algoritmos y programación y puede ser utilizado en asignaturas tales como *Introducción a la programación*, *Fundamentos de programación* o *Metodología de la programación* o con otros nombres tales como *Algoritmos*, *Programación I*, etc. Los restantes capítulos (12 al 20) pueden ser utilizados para cursos de introducción a estructuras de datos y a programación orientada a objetos, y puede ser utilizado en asignaturas tales como *Estructuras de datos I*, *Introducción a la programación orientada a objetos* o similares. C++ es uno de los lenguajes universales más utilizado y recomendado en planes de estudio de universidades y centros de formación de todo el mundo. Organizaciones como **ACM**, **IEEE**, colegios profesionales, siguen recomendando la necesidad del conocimiento en profundidad de técnicas y de lenguajes de programación estructurada con el objetivo de “acomodar” la formación del estudiante a la concepción, diseño y construcción de algoritmos y de estructuras de datos. El conocimiento profundo de algoritmos unido a técnicas fiables, rigurosas y eficientes de programación preparan al estudiante o al autodidacta para un alto rendimiento en programación y la preparación para asumir los retos de la programación orientada a objetos en una primera fase y las técnicas y métodos inherentes a ingeniería de software en otra fase más avanzada.

La mejor manera para aprender a programar una computadora es pensar y diseñar el algoritmo que resuelve el problema, codificar en un lenguaje de programación (C++ en nuestro caso) y depurar el programa una y otra vez hasta entender la gramática y sus reglas de sintaxis, así como la lógica del programa. Nunca mejor dicho: aprender practicando. El lenguaje C++ se presta a escribir, compilar, ejecutar y verificar errores. Por esta razón hemos incluido en la estructura del libro las introducciones teóricas imprescindibles con el apoyo de numerosos ejemplos, luego hemos incorporado numerosos ejercicios y problemas de programación con un análisis del problema y sus códigos fuente, y en numerosas ocasiones se presenta la salida o ejecución de los respectivos programas.

La estructura del libro en la colección *Schaum*

Esta edición ha sido escrita dentro de la prestigiosa colección *Schaum* de McGraw-Hill, como un manual práctico para la enseñanza de la programación de computadoras estudiando con el lenguaje de programación C++. Debido a los objetivos que tiene esta antigua colección, el enfoque es eminentemente práctico con el necesario estudio teórico que permita avanzar de modo rápido y eficaz al estudiante en su aprendizaje de la programación en C++. Pensando en la colección los dos autores hemos escrito este libro con un planteamiento eminentemente teórico-práctico como son todos los pertenecientes a esta colección con el objetivo de ayudar a los lectores a superar sus exámenes y pruebas prácticas en sus estudios de formación profesional o universitarios, y mejorar su aprendizaje de modo simultáneo pero con unos planteamientos prácticos: analizar los problemas, escribir los códigos fuente de los programas y depurar estos programas hasta conseguir el funcionamiento correcto y adecuado.

Hemos añadido un complemento práctico de ayuda al lector. En la página oficial del libro, encontrará el lector todos los códigos fuente incluidos y no incluidos en la obra y que podrá descargar de Internet. Pretendemos no sólo evitar su escritura desde el teclado para que se centre en el estudio de la lógica del programa y su posterior depuración (edición, compilación, ejecución, verificación y pruebas) sino también para que pueda contrastar el avance adecuado de su aprendizaje. También en la página web encontrará otros recursos educativos que confiamos le ayudarán a progresar de un modo eficaz y rápido.

¿Qué necesita para utilizar este libro?

Programación en C++. Un enfoque práctico, está pensado y diseñado para enseñar métodos de escritura de programas en C++, tanto estructurados como orientados a objetos, útiles y eficientes y como indica el subtítulo de la obra, de un modo totalmente práctico. Se pretende también enseñar tanto la sintaxis como el funcionamiento del lenguaje de programación C++ junto con las técnicas de programación y los fundamentos de construcción de algoritmos básicos, junto el diseño y construcción de clases y restantes propiedades fundamentales de la programación orientada a objetos. El contenido se ha escrito pensando en un lector que tiene conocimientos básicos de algoritmos y de programación en C/C++, y que desea aprender a conocer de modo práctico técnicas de programación, tanto estructuradas como orientadas a objetos. La gran profusión de ejemplos resueltos permitirá al lector sin conocimientos de programación básica y del lenguaje C/C++ seguir el curso incluido en el libro, aunque en este caso le recomendamos no afronte la resolución de los ejercicios y problemas hasta tanto no haya estudiado y asimilado bien los conceptos teóricos y prácticos de cada capítulo.

El libro es eminentemente práctico con la formación teórica necesaria para obtener el mayor rendimiento en su aprendizaje. Pretende que el lector utilice el libro para aprender de un modo práctico las técnicas de programación en C++, necesarias para convertirle en un buen programador de este lenguaje.

Para utilizar este libro y obtener el máximo rendimiento, es conveniente utilizar en paralelo con el aprendizaje una computadora que tenga instalados un compilador de C++ y un editor de texto para preparar sus archivos de código fuente. Normalmente, hoy día, la mayoría de los compiladores vienen con un Entorno Integrado de Desarrollo (EID) que contiene un compilador, un editor y un depurador de puesta a punto de sus programas. Existen numerosos compiladores de C++ en el mercado y también numerosas versiones *shareware* (libres de costes) disponibles en Internet. Idealmente, se debe elegir un compilador que sea compatible con la versión estándar de C++ del American National Standards Institute (ANSI), **ANSI C++**, que es la versión empleada en la escritura de este libro. La mayoría de los actuales compiladores disponibles de C++ , comerciales o de dominio público, soportan el estándar citado. Los autores hemos utilizado el compilador **Dev C++** por considerar que es uno de los más utilizados hoy día, tanto por alumnos autodidactas como por alumnos de carreras de ciencias y de ingeniería, en sus prácticas regladas o en sus hogares, al ser un compilador gratuito, de libre disposición, que funciona en entornos Windows y totalmente compatible con ANSI C++.

Aunque el libro está concebido como un libro de problemas, se ha incluido la teoría necesaria para que el libro pueda ser considerado también un libro de teoría con los conocimientos mínimos necesarios para conseguir alcanzar un nivel medio de programación. No obstante si necesita o desea adquirir un nivel más profundo de teoría de programación en C++, tanto de algoritmos, como de programación estructurada u orientada a objetos, le proponemos un libro complementario de éste, y de uno de sus coautores (Luis Joyanes) *Programación en C++: Algoritmos, estructura de datos y objetos*, 1.ª edición, publicado en el año 2000. En la actualidad el autor trabaja en la 2.ª edición que espera estar publicada en el primer semestre de 2006. Esta nueva edición tendrá, además de las actualizaciones correspondientes, el enunciado de todos los ejercicios y problemas, resueltos y propuestos en esta obra de la *Colección Schaum* de modo que ambos libros se configuran como un conjunto complementario teórico-práctico para el aprendizaje de programación en C++.

En cualquier forma *si usted sigue un curso reglado, el mejor método para estudiar este libro es seguir los consejos de su maestro y profesor tanto para su formación teórica como para su formación práctica*. Si usted es un autodidacta o estudia de modo autónomo, la recomendación entonces será que a medida que vaya leyendo el libro, compile, ejecute y depure de errores sus programas, tanto los resueltos y propuestos como los que usted diseñe, tratando de entender la lógica del algoritmo y la sintaxis del lenguaje en cada ejercicio que realice.

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 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++ (utilizado por los autores del libro para editar y compilar todos 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 oficial⁴ ha publicado el 9 de febrero de 2006, “*una lista incompleta de compiladores de C++*”, que le recomiendo lea y visite. Por su interés incluimos, a continuación, un breve extracto de su lista recomendada:

Compiladores gratuitos

Apple C++
Borland C++
Dev-C++ de Bloodshed
GNUIntel C++ para Linux
Microsoft Visual C++ Toolkit 2003
Sun Studio...

⁴ <http://public.research.att.com/~bs/compiler.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 primera quincena de marzo, y la fecha incluida es la de 9 de febrero de 2006.

Compiladores comerciales

Borland C++
Compaq C++
HP C++
IBM C++
Intel C++ para Windows, Linux y algunos sistemas empotrados
Microsoft C++

Stroustrup recomienda un sitio de compiladores gratuitos de C y C++ (Compilers.net).

NOTA PRÁCTICA DE COMPATIBILIDAD C++ ESTÁNDAR

En el artículo antes citado de Stroustrup, recomienda que compile con su compilador el siguiente simple programa fuente C++. 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";
    return 0; // esta sentencia return no es necesaria
}
```

¿Cómo está organizado el libro?

Todos los capítulos siguen una estructura similar: Breve *introducción* al capítulo; *fundamentos teóricos básicos* necesarios para el aprendizaje con numerosos ejemplos; enunciados de *Ejercicios* y *Problemas*, y a continuación, la *Solución de los ejercicios* y la *Solución de los problemas*, donde se incluyen el análisis del problema y la codificación en C++; por último, todos los capítulos contienen una colección de ejercicios y problemas propuestos, cuyo objetivo es facilitar al lector la medición de su aprendizaje.

Con el objetivo de facilitar la edición, ejecución y puesta a punto de los programas al lector, esta edición incluye una novedad: todos los códigos fuente del libro, tanto de los numerosos ejemplos como de ejercicios y problemas se han incluido en la página web oficial del libro (www.mhe.es/joyanes). Por esta razón, numerosos listados de códigos fuente de los ejercicios y problemas resueltos de diferentes capítulos, sólo se han incluido en la página Web, y no se han editado en el libro. Se pretende con esta estructura dos cosas: primera, que el libro no resultara voluminoso en número de páginas; segunda, que el lector pudiera comprobar cuando considerara oportuno la solución propuesta, sin más que bajar dicho código de la página web.

Capítulo 1. Programación orientada a objetos versus programación estructurada: C++ y algoritmos. Explica y describe los conceptos fundamentales de algoritmos, de programación estructurada y de programación orientada a objetos, junto con las propiedades más significativas de C++.

Capítulo 2. Conceptos básicos de los programas en C++. Se introduce al lector en la estructura general y elementos básicos de un programa en C++. Se describen los tipos de datos, constantes, variables y la entrada/salida; así como el nuevo concepto de espacio de nombres (*namespaces*).

Capítulo 3. Operadores y expresiones. Se aprende el uso de los operadores aritméticos, relacionales y lógicos para la manipulación de operaciones y expresiones en C. Se estudian también operadores especiales y conversiones de tipos, junto con reglas de prioridad y *asociatividad* de los operadores en las expresiones y operaciones matemáticas.

Capítulo 4. Estructuras de control selectivas (*if*, *if-else*, *switch*). Introduce a las sentencias de selección fundamentales en cualquier programa. Se examina el uso de sentencias compuestas o bloques así como el uso de operadores condicionales y evaluación de expresiones lógicas.

Capítulo 5. Estructuras de control repetitivas (*while*, *for*, *do-while*). Se aprende el concepto de bucle o lazo y el modo de controlar la ejecución de un programa mediante las sentencias *for*, *while* y *do-while*. También se explica el concepto de anidamiento de bucles y bucles vacíos; se proporcionan ejemplos útiles para el diseño eficiente de bucles.

Capítulo 6. Funciones y módulos. Examina las funciones en C++, una parte importante de la programación y enseña la programación estructurada —un método de diseño de programas que enfatiza en el enfoque descendente para la resolución de problemas mediante la descomposición del problema grande en problemas de menor nivel que se implementan a su vez con funciones. Se introduce al lector en el concepto de funciones de biblioteca, junto con la propiedad de sobrecarga y plantillas de funciones.

Capítulo 7. Arrays o arreglos (listas y tablas). Se explica un método sencillo pero potente de almacenamiento de datos. Se aprende cómo agrupar datos similares en *arrays* o “*arreglos*” (listas y tablas) numéricas.

Capítulo 8. Registros (estructuras y uniones). Se describen conceptos básicos de estructuras, uniones y enumeraciones: declaración, definición, iniciación, uso y tamaño. El concepto de tipo de dato definido por el usuario (*typedef*) y de campos de bit, se explican también en este capítulo.

Capítulo 9. Cadenas. Se describe 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. Se describen las funciones más notables de la biblioteca *string.h*.

Capítulo 10. Punteros (apuntadores). Presenta una de las características más potentes y eficientes del lenguaje C++, los *punteros*. Se describe el concepto, los punteros NULL y void y los diferentes punteros de cadena, como argumentos de funciones, punteros a funciones y punteros a estructuras, junto con la aritmética de punteros.

Capítulo 11. Gestión dinámica de la memoria. Se describe la gestión dinámica de la memoria y los operadores asociados a esta tarea: *new* y *delete*. Se proporcionan reglas de funcionamiento de estos operadores y reglas para asignación de memoria.

Capítulo 12. Ordenación y búsqueda. Se describen en el capítulo métodos básicos de búsqueda de información (secuencial y binaria) y de ordenación de listas y vectores (*burbuja*, selección, inserción y *shell*).

Capítulo 13. Clases y objetos. Los conceptos fundamentales de la programación orientada a objetos se estudian en este capítulo: clases, objetos, constructores, destructores y clases compuestas.

Capítulo 14. Herencia y polimorfismo. Las propiedades de herencia y polimorfismo constituyen, junto con las clases, la espina dorsal de la programación orientada a objetos.

Capítulo 15. Plantillas, excepciones y sobrecarga de operadores. La propiedad de genericidad se implementa mediante plantillas (*templates*). Las excepciones son un mecanismo muy potente para programación avanzada en C++ y en otros lenguajes. Por último se describe la propiedad ya estudiada de sobrecarga; en este caso, la sobrecarga de operadores.

Capítulo 16. Flujos y archivos. Las estructuras de datos básicas de almacenamiento en dispositivos externos, flujos y archivos, son objeto de estudio en este capítulo. Las clases *istream*, *ostream*, *ifstream* y *ofstream*, se analizan y describen prácticamente.

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 rango amplio. La lista enlazada se compone de elementos que contienen un valor y un puntero. El capítulo describe los fundamentos teóricos, tipos de listas y operaciones que se pueden realizar en la lista enlazada.

Capítulo 18. Pilas y colas. Las estructuras de datos más utilizadas desde el punto de vista de abstracción e implementación son las pilas y colas. Su estructura, diseño y manipulación de los algoritmos básicos se explican en el capítulo.

Capítulo 19. Recursividad. La naturaleza de la recursividad y su comparación con la iteración se muestran en el capítulo. Esta propiedad es una de las más potentes existentes en programación. Su comprensión y buen uso es muy importante para cualquier programador.

Capítulo 20 Árboles. Las estructuras de datos no lineales y dinámicas son muy utilizadas en programación. Los árboles son una de las estructuras más conocidas en algoritmia y en programación ya que son la base para las técnicas de programación avanzada.

APÉNDICES EN LA WEB

En el sitio oficial del libro, www.mhe.es/joyanes, podrá encontrar los siguientes apéndices que podrá descargar libremente:

- A. C++ frente a (*versus*) C
- B. Guía de sintaxis de ANSI/ISO C++
- C. Operadores y expresiones (prioridad)
- D. Código de caracteres ASCII
- E. Palabras reservadas ANSI/ISO C++
- F. Biblioteca estándar de funciones ANSI/ISO C++
- G. Biblioteca estándar de clases ANSI/ISO C++
- H. Glosario
- I. Recursos (Libros/Revistas/Tutoriales) de programación en la Web
- J. Bibliografía

AGRADECIMIENTOS

A nuestro editor Carmelo Sánchez que con sus sabios consejos técnicos y editoriales siempre contribuye a la mejor edición de nuestros libros. Nuestro agradecimiento eterno, amigo y editor. También y como siempre a todos nuestros compañeros del Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software de la Facultad de Informática y de la Escuela Universitaria de Informática de la Universidad Pontificia de Salamanca en el campus de Madrid que en esta ocasión y como siempre nos animan, aconsejan y asesoran en la distribución de los temas y nos dan sus opiniones para mejora de nuestras obras. Gracias colegas y amigos.

Naturalmente a nuestros lectores, razón de ser de nuestro libro. Confiamos no defraudar la confianza depositada en esta obra y aspiramos a que su progresión en el aprendizaje de la programación en C++ sea todo lo rápida y eficiente que deseamos. Así mismo deseamos destacar de modo muy especial, a todos nuestros lectores, profesores, maestros y alumnos de Latinoamérica, que utilizan nuestros libros y nos ayudan con sus comentarios continuos para mejorar cada una de nuestras obras y ediciones. Por último, un agradecimiento especial a todos los profesores y maestros que han utilizado nuestra obra *Programación en C++*; muchos nos habéis animado a escribir este nuevo libro de modo complementario y que sirviera tanto de modo independiente como unido al libro citado en talleres y prácticas de programación. Nuestro agradecimiento más sincero a todos y nuestra disponibilidad total, si así lo consideran oportuno.

En Carhelejo (Jaén) y en Madrid, marzo de 2006.

LOS AUTORES