Prólogo

Introducción

Bienvenido a ***Programación en C, C++, Java y UML*** . Hemos escrito este libro pensando en aquellas personas que se adentran a la programación a nivel universitario o de forma profesional, con poco o ningún conocimiento de programación de computadoras, de algoritmos o de objetos.

Para Latinoamérica, pensamos en los estudiantes de ingeniería de sistemas, ingeniería electrónica, ingeniería de telecomunicaciones, licenciaturas de informática o de ciencias que desean introducirse en el área de programación. Desde el punto de vista español o portugués, se ha pensado en los nuevos Planes de Estudio acogidos al EEES (Espacio Europeo de Educación Superior) conocidos también como estudios de la Declaración de Bolonia, por los que todos los países de la Unión Europea, y otros europeos, han decidido utilizar estos contenidos comunes.

Todos los estudios de computación y de informática, tanto de Latinoamérica como de Europa, suelen contemplar asignaturas tales como Introducción, Fundamentos o Metodología de Programación, junto con estudios de Estructuras de Datos y de Programación Orientada a Objetos. Según la institución académica y, sobre todo, el profesor, los estudios se realizan con base en algoritmos y utilizan un lenguaje de programación. C, C++ y Java son los lenguajes de programación por excelencia. Son muchos los profesores que optan por comenzar con C, como lenguaje de programación tradicional e histórico; luego siguen con C++ y concluyen con Java. Otros enseñan simultáneamente C++ y Java, y aquellos que incorporan ideas más innovadoras se inician directamente en Java, para llevar a sus alumnos directamente por el camino de la Web. Nuestra experiencia de más de dos décadas de enseñanza de programación de computadoras y muchas obras escritas en este campo nos dice que todas las experiencias son positivas y que es la decisión del profesor, la mejor elección y, sin duda, la más acertada.

Por estas razones, surgió entre nosotros una idea *multidisciplinar* y *multilenguaje*, y esta obra es la plasmación de esa idea. Decidimos abordar los tres lenguajes de programación, primero independientemente, luego interrelacionados y después en paralelo, de modo que el profesor pudiera elegir el orden, la secuencia, el lenguaje o los lenguajes, en la forma que mejor se adaptase a sus clases. Así lo planteamos a nuestra editorial y así lo realizamos. La obra que tiene usted en su poder, recoge estas ideas, fruto de nuestros muchos años de docencia y de investigación.

Las carreras universitarias de ciencias e ingeniería requieren un curso básico de algoritmos y de programación con un lenguaje potente y profesional pero que sea simple y fácil de utilizar. C es idóneo para aprender a programar directamente las técnicas algorítmicas y de programación, o bien en paralelo con asignaturas tales como Introducción, Fundamentos o Metodología de la Programación cuando se utiliza un lenguaje algorítmico o un lenguaje de programación estructurada. C sigue siendo el lenguaje universal 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. Por estas razones, las partes I y II constituyen con sus quince capítulos, un **curso de fundamentos de programación en C**.

Hoy día, es indispensable en el estudio de las carreras en sistemas e informática, estudiar objetos y modelado de sistemas orientado a objetos. Por esta razón, la Parte III está constituida por una **introducción a UML**, el lenguaje unificado de modelado, por excelencia, y sin duda, el más utilizado no sólo en universidades e institutos tecnológicos, sino cada vez más en escuelas de negocios, como herramienta de modelado en ambientes de negocios y de empresas.

Si el profesor ha decidido tomar como base C, se encontrará con que la próxima decisión será elegir seguir sus clases ya en siguientes semestres, en C++ y/o en Java, por esta razón las partes IV y V son **cursos de programación orientada a objetos y clases** con C++ y Java, que se pueden estudiar en paralelo o de modo secuencial.

Por último y una vez que nuestros alumnos ya conocen los tres lenguajes de programación más utilizados en enseñanza, es el momento de iniciarse en estructuras de datos y por esta razón la parte VI es una ***introducción a estructuras de datos en C, en C++ y en Java,*** organizados secuencialmente, de modo que el profesor decida su estudio en paralelo o de modo secuencial.

El contenido del libro se ha pensado para dos o tres semestres. Su objetivo es servir, en primer lugar, para asignaturas típicas de *Algoritmos*, *Introducción*, *Fundamentos* o *Metodología de la Programación*; en segundo lugar, para asignaturas de *Programación Orientada a Objetos* y por último, pero no necesariamente de modo secuencial, para asignaturas de *Estructura de Datos*. Todo ello utilizando C, C++ y/o Java, unido al lenguaje unificado de modelado UML.

¿Qué necesita para utilizar este libro?

Esta obra se ha diseñado para para que el lector conozca los métodos de escritura de programas de una manera rápida y fácil; el alumno aprenderá tanto la sintaxis y funcionamiento del lenguaje de programación como las técnicas de programación y los fundamentos de construcción de algoritmos básicos. El contenido se ha escrito pensando en la posibilidad de que el lector sea:

- Una persona novata en la programación que desea aprender acerca de la programación y escritura de programas en C, C++ y Java desde el principio.
- Una persona con conocimientos básicos de programación que ha seguido cursos de iniciación en algoritmos, como pueden ser nuestras obras: *Metodología de la programación* o *Fundamentos de programación*
- Una persona con conocimientos básicos de lenguajes de programación tales como C, C++ o Java, pero que necesita interrelacionar los tres lenguajes con el objeto de llegar a adquirir un conocimiento profundo de ambos con el objeto de aplicar eficientemente las características fundamentales de cada uno de ellos y utilizar de modo profesional aquel lenguaje que considera más idóneo para el desarrollo de su aplicación.

El libro es eminentemente didáctico para enseñanza reglada de la programación de computadoras, pero no presupone ningún conocimiento previo de programación, por lo que puede ser también utilizado por lectores autodidactas con o sin formación en informática o en ciencias computacionales.

Para utilizar este libro y obtener el máximo rendimiento, se necesitará una computadora con un compilador de C/C++ y posteriormente de Java; es deseable tener instalada una biblioteca de funciones de modo que se puedan ejecutar los ejemplos del libro y un editor de texto para preparar sus archivos de código fuente. Existen numerosos compiladores de C/C++ y Java en el mercado y también numerosas versiones *shareware* (libres de costo) disponibles en Internet. Idealmente, se debe elegir un compilador que sea compatible con la versión estándar de C/C++ del American Nacional Standards Institute (ANSI) 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 C, por lo que ésta puede ser una opción muy recomendable. En el caso de Java, las últimas versiones (5 y 6) de compiladores pueden descargarse del sitio de Sun MicroSystems (`java.com`), por lo que siempre tendrá la seguridad de utilizar un estándar. Más adelante se incluyen recomendaciones sobre los compiladores y fabricantes más populares, así como la mejor forma de descargarse versiones gratuitas de la Web.

Se puede utilizar cualquier editor de texto, tal como Notepad o Vi, para crear sus archivos de programas fuente, aunque será mucho mejor utilizar un editor específico para editar código, como los que

suelen venir con los entornos integrados de desarrollo, para Windows o Linux. Sin embargo, no se deberá utilizar un procesador de textos, tipo Microsoft Word, ya que normalmente los procesadores de texto o de tratamiento de textos comerciales, incrustan o "embeben" códigos de formatos en el texto que no serán entendidos por su compilador.

Si usted es alumno, de cualquier nivel de enseñanza, y 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 compile, ejecute y depure (limpie) de errores sus programas, tanto los propuestos en el libro, como los que usted diseñe, a medida que vaya leyendo el libro, tratando de entender la lógica del algoritmo y la sintaxis del lenguaje en cada ejercicio que realice.

El objetivo final que buscamos es, no sólo describir la sintaxis de C, C++ y Java, sino, y sobre todo, mostrar las características más sobresalientes de cada lenguaje a la vez que se enseñan técnicas de programación estructurada y orientada a objetos y posteriormente las técnicas básicas de estructura de datos. Por consiguiente, las características fundamentales de esta obra 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/C++ y Java 5/6,** incluyendo las últimas novedades de los tres lenguajes de programación.
- Tutorial enfocado a los tres lenguajes, incluyendo numerosos ejemplos, ejercicios y herramientas de ayuda al aprendizaje.
- Descripción detallada del lenguaje respectivo, con un énfasis especial en técnicas de programación actuales y eficientes. Dado que hemos optado por iniciar el aprendizaje con un curso completo de C, dedicamos dos capítulos específicos para explicar las diferencias clave de C++ y Java comparadas con C (capítulos que hemos rotulado como "De C a C++" y "De C/C++ a Java") antes de iniciar el aprendizaje ya más en profundidad de ambos lenguajes.
- El contenido se ha estructurado en diferentes partes siguiendo nuestra experiencia docente en el mundo de la programación y en nuestras obras similares, y sobre todo en una secuencia que consideramos beneficiará al alumno en su formación progresiva: **Programación en C**, **Introducción a UML** (que nos facilite el estudio de la programación orientada a objetos), **Programación en C++**, **Programación en Java** y **Estructura de Datos** (en este caso hemos optado por explicar los conceptos clave también en modo secuencial, y en la medida de lo posible en paralelo, de modo que el lector pueda comparar los algoritmos y diseño de clases y vea la eficiencia de cada lenguaje comparado con los otros dos).
- Una introducción a la informática, a las ciencias de la computación y a la programación, usando una herramienta de programación denominada C/C++ o Java.

En resumen, éste es un libro diseñado para enseñar a programar utilizando un lenguaje de programación y no un libro específico diseñado para enseñar C/C++ o Java, aunque también pretende conseguirlo. No obstante, confiamos que los estudiantes y autodidactas que utilicen la obra puedan conocer los tres lenguajes de programación y los conocimientos clave de UML, de modo que aprendan y conozcan profesionalmente, tanto las técnicas clásicas y avanzadas de programación estructurada como las técnicas orientadas a objetos y el diseño y construcción de estructura de datos. 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 y le facilitará el tránsito a los caminos que le conducirán a la programación de lenguajes de programación para la Web, más específicos, como C++ y aquéllos utilizados en las nuevas tecnologías de la Web 2.0, AJAX y Web semántica, tales como JavaScript, XML o Ruby.

Objetivos

El libro pretende enseñar a programar una computadora utilizando varios lenguajes de programación estándares —C, C++ y Java—, introducir al lector en el lenguaje de modelado de sistemas UML, enseñar a programar con el paradigma estructurado y el orientado a objetos, así como introducir al lector en el fundamental campo de las estructuras de datos.

Para ello se apoyará en los siguientes conceptos fundamentales:

1. *Algoritmos* (conjunto de instrucciones programadas para resolver una tarea específica).
2. *Sintaxis de los lenguajes de programación C, C++ y Java,* en sus versiones estándares en el caso de C/C++ y en las últimas versiones 5.0 y 6 de Java.
3. *Orientación a Objetos* (datos y algoritmos que manipulan esos datos, encapsulados en un tipo de dato conocido como objeto)
4. *Estructuras de Datos* (colecciones 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).*
5. *UML 2.1, notaciones y relaciones gráficas de clases y objetos* (iniciación al modelado de aplicaciones con una herramienta universal, como es UML, lenguaje unificado de modelado)
6. *Técnicas de programación en C, C++ y Java* (aprendizaje fiable y eficiente del desarrollo de programas de aplicaciones)

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++, proporciona el paradigma *procedimental c*on un énfasis en funciones, plantillas de funciones y algoritmos genéricos.

En la década de 1980, el enfoque del diseño de programas se desplazó del *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*, es decir, admite ambos tipos de paradigmas. La gran ventaja es que se puede proporcionar la solución mejor; resuelve cada problema con el paradigma más adecuado, dado que ningún paradigma soluciona 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++.

La evolución de C

C fue una evolución de los lenguajes BCPL y B, desarrollados en la década de los sesenta por Martin Richards como lenguajes para escribir sistemas operativos y compiladores. Posteriormente Kein Thompson utilizó B para crear las primeras versiones del sistema operativo UNIX en Bell Laborato-

rios, sobre una computadora DEC PDP-7 de Digital. El lenguaje C, como tal, fue una evolución directa del lenguaje B y fue creado en 1972 por Dennis Ritchie, con la colaboración de Ken Thompson también en Bell Laboratories y se implementó originalmente en la mítica computadora DEC PDP-11.

En 1978, Ritchie y Brian Kernighan, publicaron la primera edición de *El lenguaje de programación C,* primera especificación no formal del lenguaje C y que se ha conocido tradicionalmente como el "**K&R C**" (el **C** de *Kernighan y Ritchie*). En 1983, el American Nacional Standards Institute (ANSI) creó el comité X3J11 cuyo objetivo era definir una especificación estándar de C. En 1989, fue aprobado el estándar ANSI X3159:1989. Esta versión se conoce como **ANSI C** o bien **C89**.

En 1990, el International Standar Organization (**ISO**) adoptó como estándar el ANSI C, con la denominación de estándar ISO/IEC 9899: 1990, que se le conoce como **C90** o **ANSI/ISO C**, aunque ambas versiones, ANSI C y ANSI/ISO C son formalmente las mismas.

El lenguaje C siguió evolucionando y en 1999 —ante el avance de C++— fue revisado y actualizado, publicándose un nuevo estándar INCITS/ISO/IEC 9899:1999. Este estándar es conocido como C99 y su especificación completa se puede consultar y adquirir en: `webstore.ansi.org/ansidocstore`

Las actas del grupo WG14 de estandarización de C se pueden encontrar en el sitio oficial del estándar **C99**: `www.open-std.org/jtc1/sc22/wg14/www/docs/n1250.pdf`

C99 no ha tenido tan gran aceptación como el ANSI/ISO C, aunque poco a poco, la gran mayoría de constructores de compiladores lo incluyen en la actualización de sus versiones y comienzan a ser comercializadas conformes a la norma estándar.

Los compiladores gratuitos más populares, tales como: Microsoft Visual C++ 5.0, GCC de GNU y su entorno integrado de desarrollo Dev-C++ 4.9.9.2 de Bloodshed, así como los compiladores de Borland contienen casi todas las funcionalidades y características del estándar C99.

Desde 2007, se estudian nuevas revisiones del estándar. Así en la reunión del grupo de trabajo de estandarización WG14/INCITS J11, conocido como WG14, celebrada en abril de 2007, hubo acuerdo general de trabajar en una nueva revisión del estándar C que fue etiquetada con el nombre de **C9X**. Posteriormente, en una reunión posterior del grupo WG14, en Londres, se acordó seguir adelante con el proyecto pero se le cambió el nombre a **C1X,** que es el nombre por el que en estos momentos se le conoce.

En la enciclopedia Wikipedia puede encontrar el lector una extensa historia del lenguaje C, así como numerosas notas y referencias bibliográficas, unido a una excelente lista de enlaces externos.

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 *programación orientada a objetos, programación genérica* y la tradicional *programación procedimental* o *estructurada*. C++ ha ido evolucionando desde la publicación del libro de Bjarne 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,* que suelen incorporar casi todos los compiladores.

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.

Se revisarán 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 de plantillas estándar (STL **o** 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 tercera 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 (como Pascal, Basic, Fortan, entre otros). La explosión del lenguaje en la comunidad informática hizo inevitable la estandarización, proceso que comenzó en 1987 [Stroustruo 94]. Así nació una primera fuente de estandarización, la ya citada obra: *The Annotated C++ Reference Manual* [**Ellis,** 89][1]. 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, 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[2] *(*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++.

La evolución de Java: Java 5 y Java 6

James Gosling y otros desarrolladores de Sun Microsystems, trabajaban, a primeros de los noventa, en un proyecto que deseaba diseñar un lenguaje de computadoras que pudiera ser utilizado para dispositivos de consumo tales como aparatos de televisión y otros componentes electrónicos. Los requisitos eran simples: un lenguaje portable que pudiera ser utilizado en cualquier máquina que tuviera el interprete adecuado. El proyecto se llamaba "*Green*" y el lenguaje pretendía generar código intermedio para una máquina hipotética. Este código intermedio debía poder utilizarse en cualquier máquina que tuviera el interprete correcto. El proyecto utilizaba una máquina virtual, pero los ingenieros procedían de trabajar en entornos UNIX con el lenguaje C++ y por esta razón era un lenguaje orientado a objetos y no a procedimientos. En paralelo a los desarrollos de Sun, la World Wide Web, desde 1993, estaba creciendo a marchas agigantadas. La clave para la web fue el navegador que traducía la página del hipertexto a la pantalla. En 1994, la mayoría de los usuarios de Internet utilizaban Mosaic, un navegador web no comercial.

El proyecto Green se transformó en "First Person, Inc"; se presentaron algunos resultados de éxito, pero se disolvió en 1994; sin embargo el lenguaje que primero se llamó "Oak" y posteriormente Java comenzó a utilizarse en el diseño de navegadores, en concreto en uno llamado HotJava que fue escrito en Java para mostrar toda su potencia. Se construyeron aplicaciones que llamaron "*applets*" e hicieron el navegador capaz de ejecutar código en el interior de páginas web. Esta experiencia se mostró en el SunWorld '95 y fue el comienzo de la historia de Java y el inicio de la leyenda de este lenguaje para desarrollo en Internet.

Sun lanzó la primera versión de Java a principios de 1996 y se le denominó Java 1.0 y desde entonces ha lanzado siete importantes revisiones del Java Development Kit (JDK) y la interfaz de programación de aplicaciones (API) ha crecido desde alrededor de 200 clases a más de 3 000 clases. La API genera ahora áreas tan diversas como construcción de interfaces de usuario, gestión de bases de datos, internacionalización, seguridad y procesamiento XML. En 1997, se lanzó la primera revisión, 1.1 que ya contenía clases internas. En diciembre de 1998 con ocasión de la conferencia JavaOne se presentó la versión 1.2 y se cambió el nombre comercial a *Java 2 Standard Edition Software Development Kit Versión 1.2* (ya se le conoció como Java 2, hoy sigue siendo muy utilizada).

Las versiones 1.3 (en el año 2000) y 2.4 (en el año 2004) de la Standard Edition eran mejoras incrementales sobre la versión inicial de Java 2. A finales de 2004 se presentó la versión Java 5.0 (ori-

[1] Existe versión española de Addison-Wesley Díaz de Santos y traducida por los profesores Manuel Katrib y Luis Joyanes.

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

ginalmente se le llamó versión 1.5) que es la primera versión que actualiza significativamente el lenguaje Java desde la versión 1.1 (se le añadió, por ejemplo, "clases genéricas", el bucle "`for each`", varargs, autoboxin, metadatos, enumeración e importación estática). Las clases genéricas eran similares a las plantillas (*templates*) de C++. y las características "`bucle for each`", "autoboxing" y metadatos, eran características que había incorporado C++.

Por último, a finales de 2006, se lanzó la versión 6 (sin el sufijo .0) que no trajo mejoras en el lenguaje sino en prestaciones y en bibliotecas. Las versiones actuales se conocen como J2SE 6.0 (Standard Edition, v. 1.6.0)

Compiladores y compilación de programas

C y C++

Existen numerosos compiladores de C++ en el mercado. Desde ediciones gratuitas y *descargables* a través de Internet hasta profesionales, con costos y fabricantes diferentes. 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 o centro de estudios. Si usted es un lector autodidacta, existen varias versiones gratuitas que puede descargar de 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 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 publicó, en su página oficial[3] "una lista incompleta de compiladores de C++", que le recomendamos leer, aunque aclara que es imposible tener actualizada la lista de compiladores disponibles. A continuación incluimos 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 multiples plataformas de sistemas empotrados
- HP C++
- IBM C++
- Intel C++ para Windows, Linux, y sistemas empotrados.

[3] El artículo se titula: "An Incomplete List of C++ Compilers"y su autor suele modificarlo (en la cabecera indica la fecha de modificación).

- 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.

Asimismo, Stroustrup recomienda un sitio de compiladores gratuitos de C y C++: `compilers.net` y también el Ddirectorio público: The Open Directory Project

Instalación de Visual C++ 5.0

Otra herramienta muy popular y gratuita es el compilador que ofrece Microsoft C++, la versión 5.5 , Visual C++ 2005 Express Edition y que puede descargar de

```
msdn.microsoft.com/vstudio/express/visualc
msdn.microsoft.com/vstudio/express/support.faq
```

Microsoft proporciona un foro dedicado al uso de la versión Express Edition y que se puede descargar en: `forums.microsoft.com/msdn/ShowForum.aspx?ForumID=24`

Instalación del compilador GCC de GNU y el IDE Dev-C++

El compilador GCC de GNU y el entorno Dev-C++ son uno de los programas más populares en la comunidad de programadores C/C++, puede descargarse e instalar el entorno y el compilador en su última versión IDE 4.9.9.2

```
www.bloodshed.net/devcpp.html
```

Nota práctica de C++ estándar (Bjarne Stroustrup)

En el caso de C/C++, su inventor, Stroustrup, recomienda en su página web una regla de compatibilidad de compiladores y que nosotros lógicamente también aconsejamos. 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";
  // en el original ""Please enter your first name followed by a // newline\n";
  cin >> s;
  cout << "Hola, " << s << ´\n´;
    // en el original "Hello, " y "this return statement isn't
    // necessary"
  return 0; // esta sentencia return no es necesaria
}
```

Java

El lector debe comenzar por aprender a instalar el kit de desarrollo de Java (JDK, *Java Development Kit*) que le permitirá compilar y ejecutar diferentes tipos de programas: programas de consola, aplicaciones gráficas y *applets*. Se pueden ejecutar las herramientas JDK tecleando órdenes en una ventana

shell. Sin embargo, la mayoría de los programadores prefieren el confort del entorno integrado de desarrollo. A continuación, le indicamos como puede instalar y utilizar gratuitamente el entorno de desarrollo.

Las versiones más completas y actualizadas de JDK están disponibles en Sun Microsystems para sistemas operativos Solaris, Linux y Windows; para entornos de Macintosh y otras plataformas, es posible, conseguir también versiones de otros distribuidores.

Si usted utiliza Solaris, Linux o Windows, puede descargar el entorno JDK y su documentación en:

```
java.sun.com/javase/downloads/index.jsp    java.sun.com/javase/6/download.jsp
```

Sun proporciona el Java Center, que es un sitio excelente con instrucciones de todo tipo para instalación en Windows, Linux y Mac OS X y Leopard.

```
Java.sun.com/developer/onlineTraining/new2java/index.html
```

Existen numerosos entornos integrados de desarrollo (IDE, Integrated Development Environment) de diferentes fabricantes que admiten todo el proceso de desarrollo de software, incluyendo editores para escribir y editar programas, además de depuradores para localizar y corregir errores lógicos. Algunos de los IDE más populares son los siguientes:

NetBeans (EID de Sun)	`www.netbeans.org`
JBuilder, del fabricante Borland	`www.norland.com`
BlueJ (EID gratuito y muy popular)	`www.bluej.org`
Eclipse	`www.eclipse.org`

El libro como herramienta docente

La experiencia de los autores desde hace muchos años con obras muy implantadas en el mundo universitario como *Programación en C++* (primera edición), *Fundamentos de programación* (en su cuarta edición), *Programación en C* (en su segunda edición), *Programación en Pascal* (en su cuarta 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 siglo XXI. Por ello en el contenido de la obra hemos tenido en cuenta no sólo las directrices de los planes de estudio europeos (Declaración de Bolonia) sino también los latinoamericanos.

El contenido del libro sigue un programa estándar de un primer curso de introducción a la programación y, 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 la currícula actual y vigente en universidades latinoamericanas, muchas de las cuales conocemos y con las que tenemos relaciones profesionales.

Características importantes del libro

Programación en C, C++, Java y UML 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.
- **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.
- **Recuadros**. 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.

¿Cómo está organizado el libro?

Todos los capítulos siguen una estructura similar; cada uno comienza con un extracto del contenido, una breve introducción al capítulo y una lista de los conceptos y términos más importantes del capítulo. La descripción teórica y práctica se acompaña de numerosos ejemplos y ejercicios prácticos con el objetivo fundamental de que el lector/alumno vaya aprendiendo a la par que sigue y estudia su contenido. A continuación se incluye un resumen con un recordatorio de los conceptos teóricos y/o prácticos más importantes del capítulo. Por último se incluyen una serie de descripciones de ejercicios de nivel de iniciación y medio, junto con otra lista de enunciados de problemas propuestos, con el objetivo fundamental que el lector/alumno practique y vea la progresión realizada con el seguimiento del libro así como de sus clases teórico-prácticas en su centro de enseñanza.

Capítulo 1. Introducción a las computadoras, a Internet (Web) y a los lenguajes de programación. Explica y describe los conceptos fundamentales de la computación y de los lenguajes de programación. El conocimiento completo del contenido del capítulo no es requisito imprescindible para el conocimiento y aprendizaje de los restantes capítulos aunque si es fundamental en su formación de programación. Por estas circunstancias el lector/alumno puede optar por su estudio en una sola vez o bien graduar el aprendizaje a lo largo de todo su curso de programación. Siempre se cuenta también con el profesor y maestro que podrá orientar al alumno en su progresión docente. Para los lectores que no hayan recibido ningún curso de introducción a las computadoras o a la informática les recomendamos su lectura antes de pasar al siguiente capítulo y luego una relectura en el momento y forma que ellos mismos consideren. Este capítulo ha sido actualizado totalmente y se han introducido tanto los conceptos fundamentales y genéricos tradicionales como las últimas innovaciones tecnológicas que más afectan al mundo de la programación.

Capítulo 2. Metodología de la programación: algoritmos y ciclo de vida del software. Se introduce al lector en los conceptos fundamentales de algoritmos y sus herramientas de representación. Así mismo se describen los tipos clásicos de programación con especial énfasis en la programación estructurada soporte del lenguaje C. Una breve introducción a la ingeniería de software pretende que el lector se mentalice desde el comienzo de su formación en programación en la importancia de la ingeniería de software como disciplina académica y sobre todo como actividad de ingeniería. 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); comienza con una introducción a la informática y a las ciencias de la computación así 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 3. El lenguaje C. Elementos básicos. Introduce a la estructura y los componentes principales de un programa en C. Aprende los significados de los elementos fundamentales de todo programa, tales como datos, constantes, variables y las operaciones básicas de entrada/salida.

Capítulo 4. 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 5. Estructuras de selección: sentencias `if` y `switch`. Introduce a las sentencias de selección básicas y 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 6. Estructuras de control: bucles. 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 7. Funciones. Examina las funciones en C, una parte importante de la programación. Aprende programación estructurada, un método de diseño de programas que enfatiza en el enfoque descendente para la resolución de problemas mediante la descomposición de un problema grande en problemas de menor nivel que se implementan a su vez con funciones.

Capítulo 8. Arreglos (*arrays*) Explica un método sencillo pero potente de almacenamiento de datos. Se aprende como agrupar datos similares en *arrays* o "**arreglos**" (listas y tablas numéricas).

Capítulo 9. Ordenación y búsqueda. Enseña los métodos para ordenar listas y tablas, así cómo búsqueda de datos en listas y tablas. Se estudian los algoritmos clásicos más sencillos y eficientes tanto de ordenación como de búsqueda

Capítulo 10. Estructuras y uniones. Se describen conceptos básicos de estructuras, uniones y enumeraciones: declaración, definición, iniciación, uso y tamaño. Las operaciones fundamentales de acceso a estructuras, arreglos de estructuras y estructuras anidadas se analizan también en este capítulo. En el capítulo se muestra de modo práctico como usar estructuras y uniones para conseguir las necesidades del programa; se explican las diferencias entre estructuras y uniones, así como el uso de la palabra reservada `typedef`.

Capítulo 11. Apuntadores (Punteros). Presenta una de las características más potentes y eficientes del lenguaje C, los *apuntadores.* Este capítulo proporciona explicación detallada de los punteros, arreglos de punteros, punteros de cadena, aritmética de punteros, punteros constantes, punteros como argumentos de funciones, punteros a funciones y a estructuras. De un modo práctico aprende el modo de utilizar punteros a punteros y cómo se pueden utilizar los arreglos de punteros para manipular las cadenas, que se estudiarán en profundidad en el capítulo 14.

Capítulo 12. Asignación dinámica de memoria. Se describe la gestión dinámica de la memoria y las funciones asociadas para esas tareas: `alloc( )`, `free( )`, `calloc( )` y `realloc( )`. Se proporcionan reglas de funcionamiento de esas funciones y reglas para asignación de memoria.

Capítulo 13. Cadenas. Se describe el concepto de cadena (*string)* así como las relaciones entre apuntadores, arreglos 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 14. Recursividad. La recursividad o propiedad de una función o expresión de llamarse a sí misma es una de las técnicas más importantes en la construcción de algoritmos. Por esta razón se dedica un capítulo completo al aprendizaje de las funciones recursivas.

Capítulo 15. Entrada y salida por archivos. Se estudia el concepto de flujo *(stream)* y los diferentes métodos de apertura de archivos, junto con los conceptos de archivos binarios y funciones para el acceso aleatorio. Muestra de un modo práctico como C utiliza los flujos, examina los flujos predefinidos y el modo práctico de trabajar con la pantalla, la impresora y el teclado.

Capítulo 16. Tipos abstractos de datos, objetos y modelado con UML 2.0. Con este capítulo se inicia el estudio del Lenguaje Unificado de Modelado (**UML**). Se describen las características fundamentales de la programación orientada a objetos y como se modelan e identifican los objetos, así como las propiedades fundamentales de la orientación a objetos.

Capítulo 17. Diseño de clases y objetos. Los conceptos fundamentales de clases y objetos, junto con constructores y destructores y como se representan gráficamente las clases y objetos en **UML 2.1**. El concepto de recolección de basura implementado en Java es otra de las propiedades clásicas que se describen en el capítulo.

Capítulo 18. Relaciones entre clases: delegaciones, asociaciones, agregaciones y herencia. Uno de los conceptos clave en el modelado orientado a objetos es la relación. Las relaciones entre clases y sus diferentes tipos (generalización, especialización, delegación, agregación, asociación, etc.) se estudian con ejemplos prácticos y sus respectivas notaciones gráficas en **UML 2.1**.

Capítulo 19. De C a C++. 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`). Se describen los conceptos fundamentales del lenguaje C++, haciendo énfasis en las diferencias clave con los conceptos aprendidos de C. En esencia, el capítulo se ha escrito como un breve curso de introducción a la programación en C++, y que debe servir de fundamentos para el aprendizaje de los siguientes capítulos que se centran esencialmente en *programación orientada a objetos* y en propiedades específicas de C++ como *excepciones* y *sobrecarga de operadores.*

Capítulo 20. 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 21. 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 22. 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.

Capítulo 23. 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 24 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.

Capítulo 25. De C/C++ a Java 5/6. Este capítulo constituye, al igual que el capítulo 19 para C++, un breve curso de introducción al lenguaje Java describiendo los elementos fundamentales de un programa en Java.

Capítulo 26. Programación orientada a objetos en Java: Clases y objetos. En este capítulo se describen los conceptos ya conocidos de clases y objetos, y como se implementan en Java, juntos con los conceptos nuevos de paquetes y recolección de objetos.

Capítulo 27. Programación orientada a objetos en Java: Herencia y Polimorfismo. Las ya conocidas clases derivadas se vuelven a estudiar en este capítulo desde el enfoque Java. Asimismo, se describen las propiedades clásicas de herencia y polimorfismo.

Capítulo 28. Colecciones. Colección es un concepto importante en Java junto con iteradores, conjuntos, mapas y diccionarios. Los iteradores e interfaces son otros conceptos importantes en Java y que se estudian en el capítulo.

Capítulo 29. *Applets* y *swing*. Diseño de componentes gráficos. Los *applets* como aplicaciones de Internet y la biblioteca de gráficos, *swing*, popularizada en las últimas versiones de Java 5 y Java 6, se explican en este capítulo.

Capítulo 30. Multitarea (Hilos) y excepciones. Dos propiedades muy importantes se describen en este capítulo: el ya conocido concepto de excepción y la propiedad que posibilita la multitarea mediante hilos de ejecución.

Capítulo 31. Organización de datos dentro de un archivo en C. La organización clásica de los datos en archivos es el tema central de este capítulo. Se hace una breve introducción a la ordenación y clasificación de archivos.

Capítulo 32. Listas, pilas y colas en C. El estudio de las estructuras básicas de datos: listas, pilas y colas, son los conceptos clave explicados en este capítulo. Se realiza una introducción básica a dichas estructuras y se estudia su codificación en C.

Capítulo 33. Flujos y archivos en C++. El diseño e implementación de los flujos (*stream)* y archivos en C++ es el motivo central de este capítulo.

Capítulo 34. Listas, Pilas y Colas en C++. La implementación de las estructuras de datos básicas en C++ se describen en el capítulo.

Capítulo 35. Archivos y flujos en Java. La codificación de los archivos y flujos en Java constituyen la parte fundamental del capítulo.

Capítulo 36. Listas, Pilas y Colas en Java. Los importantes conceptos de listas, pilas y colas, ya estudiados anteriormente, se estudian, y sus algoritmos y clases se codifican en Java.

Apéndices. Se incluyen *códigos de numeración*, *códigos ASCII* y *Unicode* y reglas de *prioridad y asociatividad de los operadores* en los tres lenguajes de programación clásicos y estudiados en el libro: C, C++ y Java 5/6. Asimismo se incluye una amplia bibliografía de algoritmos, estructuras de datos, orientación a objetos, estructuras de datos y, como ayuda complementaria al lector tanto en el aspecto docente como en el profesional, una amplia lista de recursos Web.

Apéndices incluidos en la Web

`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 de su contenido. Estos apéndices sirven de guía y referencia de elementos importantes del lenguaje y de la programación de computadoras. En la página oficial del libro en la Web, se han incluido numerosos apéndices, además de los incluidos en la edición en papel, de modo que el lector pueda "bajarlos" de ella cuando lo considere oportuno y en función de su lectura y aprendizaje. Se incluyen esencialmente:

Guías de sintaxis de C, C++ y Java
Guía rápida de referencia de UML 2.1
Biblioteca de funciones de C, C++ y Java 5/6
Biblioteca de clases de C++ y Java
Biblioteca estándar STL de C++
Glosario de términos
Cursos de introducción a la computación y a la programación
Cursos de algoritmos y estructuras de datos
Tutoriales
Guía de buenas prácticas de programación en C/C++ y Java 5/6
Bibliografía y Recursos Web actualizados
Hemeroteca

Agradecimientos

A nuestros compañeros 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 nos dieron consejos sobre las mismas:

Dra. Matilde Fernández Azuela
Dr. Lucas Sánchez García

Gracias, compañeros y, sin embargo, amigos.

En esta ocasión hemos de destacar un agradecimiento especial a nuestros editores mexicanos: **Pablo Roig** —editor *sponsor*—, **Ana Delgado** —editora de desarrollo—, **Marcela Rocha** —coordinadora editorial— y **Ricardo del Bosque** —director editorial— que en la distancia nos han ayudado facilitando nuestra tarea; nos han ido asesorando y dándonos consejos que han fructificado en esta obra. Desde aquellos días de finales de octubre donde en sesiones de trabajo en México, D.F. con mis buenos amigos **Miguel Ángel Toledo, Ramón Orduña** y **Pablo Roig,** y muchos otros miembros del equipo editorial y comercial, pusimos en marcha esta obra, han pasado muchos meses de trabajo duro, gracias a ellos y muchos otras personas de la editorial en México, esta obra hoy ve la luz. Gracias amigos.

Sin embargo, no podemos dejar de acordarnos y agradecer, como siempre, el apoyo de nuestra editorial en Madrid; **Manuel Cejudo** y **Cristina Sánchez**, como editores nos han prestado todo tipo de ayuda para que el proceso editorial tuviera éxito. Ellos, como en otras ocasiones, nos han ayudado y sobre todo nos han facilitado el *camino virtual Madrid-México DF*, y sus consejos también nos han ayudado considerablemente.

De un modo muy especial y con nuestro agradecimiento eterno, a nuestros lectores, a los estudiantes de Hispanoamérica, Brasil, Portugal —y otros países de habla portuguesa— y España, que han estudiado o consultado otras obras nuestras, y como no podía ser menos, nuestro agradecimiento más sincero a profesores y maestros de Latinoamérica y España, que han tenido la amabilidad de consultar o seguir sus contenidos en sus clases y a todos aquellos que nos han dado consejos, sugerencias, propuestas para que escribiéramos una obra nueva con un enfoque de multilenguaje. Nuestro reconocimiento más sincero y de nuevo "nuestro agradecimiento eterno"; son el aliento diario que nos permite continuar con esta hermosa tarea que es la comunicación con todos ellos. Y como no podía ser menos, nuestro sincero y eterno agradecimiento a nuestros alumnos y lectores que han confiado en nuestras obras. Simplemente, gracias a todos.

En Carchelejo (Sierra Mágina, Andalucía), en Lupiana (Guadalajara,Castilla La Mancha) y en México, D.F., febrero de 2009.