

# Prólogo

Dos de las disciplinas clásicas en todas las carreras relacionadas con la Informática y las Ciencias de la Computación son: *Estructuras de Datos* y *Algoritmos* o bien una sola disciplina, si ambas se estudian integradas en *Algoritmos y Estructuras de Datos*. El estudio de estructuras de datos y de algoritmos es tan antiguo como el nacimiento de la programación y se ha convertido en estudio obligatorio en todos los currículos desde finales de los años sesenta y sobre todo en la década de los setenta cuando apareció el Lenguaje Pascal de la mano del profesor suizo Niklaus Wirth, y posteriormente en la década de los ochenta con la aparición de su obra —ya clásica— *Algorithms and Data Structures* en 1986. ***Estructuras de Datos en C++*** trata sobre el estudio de las estructuras de datos dentro del marco de trabajo de los tipos abstractos de datos (**TAD**) y objetos, bajo la óptica del análisis, diseño de algoritmos y programación, realizando las implementaciones de los algoritmos en C++.

C++ es un *superconjunto* y una extensión de C, 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 —creador y padre de C++— junto con una gran mayoría de programadores contesta así: “No sólo no 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. En el caso de un libro de Estructuras de Datos como el que Vd. tiene en sus manos, la problemática es la misma, por lo que se puede aprender a analizar y diseñar estructuras de datos directamente desde C++ . Pero en cualquier forma y en apoyo de nuestra teoría anterior, hemos introducido en los primeros capítulos los conceptos básicos necesarios para seguir el contenido de la obra tanto si usted ya es programador de C++ como si procediese de C y no tuviese esa formación específica en C++.

¿Porqué en C++ y no en Java, porqué no en C/C++ o en Visual Basic/C# ? Muchas Facultades y Escuelas de Ciencias y de Ingeniería, así como Institutos Tecnológicos y Centros de Formación Profesional, comienzan sus cursos de Estructuras de Datos con el soporte de C y muchas otras con el soporte de C++ o Java. De hecho, en nuestra propia universidad, en asignaturas relacionadas con esta disciplina se aprende a diseñar y construir estructuras de datos utilizando C, C++ o Java, a veces, indistintamente. ¿Existe una solución ideal? Evidentemente, consideramos que no y cada una de ellas tiene sus ventajas y sus inconvenientes, y es la decisión del maestro y profesor, responsable de su formación, quien debe elegir aquella que considera más recomendable para sus alumnos teniendo en cuenta el entorno y contexto donde se desarrolla su labor, ya que siempre pensará en su mejor futuro y por esta razón siempre la encajará dentro del currículo específico de su carrera en el lugar que considere más oportuno.

El primer problema que se suele presentar al estudiante de *algoritmos y estructuras de datos* que, probablemente, procederá de un curso de nivel básico, medio o avanzado de *introducción o fundamentos de programación* o bien de *iniciación de algoritmos*, es precisamente el modo de afrontar información compleja desde el principio. Al permitir C++ el empleo de los dos paradigmas clásicos de programación: *procedimental* y *orientada a objetos*, el aprendizaje de la disciplina le será más fácil ya que podrá adaptarse en función de su formación. Pensando en aquellos lectores que no hayan seguido ningún curso de programación orientada a objetos (POO) y dado que la POO es una herramienta de programación y organización muy potente y con grandes ventajas para la enseñanza y posterior tarea profesional, se han incluido capítulos específicos de clases, objetos, clases derivadas, herencia, polimorfismo y plantillas, que pese a no conformar un curso de introducción a POO si se han escrito pensando en servir de fundamentos básicos a modo de introducción, recordatorio o nueva formación.

Por otra parte, la mayoría de los estudiantes de informática, ciencias de la computación, ingeniería de sistemas o de telecomunicaciones, o cualesquiera otra carrera de ciencias o ingeniería, o de estudios de formación profesional de nivel superior, requieren conocer bien el flujo C-C++ y viceversa. El enfoque orientado a objetos permite la implementación de las estructuras de datos con clases y objetos aportando a las estructuras de datos su verdadera potencialidad, si bien hay que reconocer que el paradigma estructurado para aprender la idea de algoritmo y estructuras de datos está muy extendido entre múltiples profesores, alumnos, etc. El libro está soportando la comprensión del tipo abstracto de datos (TAD) con un estilo que permite la formación de las estructuras de datos orientadas a objetos.

Además de estas ventajas, existen otras, que si bien se pueden considerar menores, no por ello menos importantes y son de gran incidencia en la formación en esta materia. Por ejemplo, algunas de las funciones de Entrada/Salida (tan importantes en programación) son más fáciles en C++ que en C (véase el caso de números enteros), otros tipos de datos tales como cadenas y números reales se pueden formatear más fácilmente en C. Otro factor importante para los principiantes es el conjunto de mensajes de error y advertencias proporcionadas por un compilador durante el desarrollo del programa.

Se estudian estructuras de datos con un objetivo fundamental: aprender a escribir programas más eficientes. También cabe aquí hacerse la pregunta ¿Por qué se necesitan programas más eficientes cuando las nuevas computadoras son más rápidas cada año (en el momento de escribir este prólogo, las frecuencias de trabajo de las computadoras personales domésticas son de 3 GHz o superiores —o bien 1,6 a 1,8 GHz en el caso de procesadores de doble núcleo tanto de AMD como de Intel—, y las memorias centrales de 1 GB a 4 GB, son prácticamente usuales en la mayoría de las PCs y claro está son el nivel de partida en profesionales). La razón tal vez resida en el hecho de que nuestras metas no se amplían a medida que se aumentan las características de las computadoras. La potencia de cálculo y las capacidades de almacenamiento aumentan la eficacia y ello conlleva un aumento de los resultados de las máquinas y de los programas desarrollados para ellas.

La búsqueda de la eficiencia de un programa no debe chocar con un buen diseño y una codificación clara y legible. La creación de programas eficientes tiene poco que ver con “trucos de programación” sino al contrario se basan en una buena organización de la información y buenos algoritmos. Un programador que no domine los principios básicos de diseños claros y limpios probablemente no escribirá programas eficientes. A la inversa, programas claros requieren organizaciones de datos claras y algoritmos claros, precisos y transparentes.

La mayoría de los departamentos informáticos reconocen que las destrezas de buena programación requieren un fuerte énfasis en los principios básicos de ingeniería de software. Por consiguiente, una vez que un programador ha aprendido los principios para diseñar e imple-

mentar programas claros y precisos, el paso siguiente es estudiar los efectos de las organizaciones de datos y los algoritmos en la eficiencia de un programa.

## EL ENFOQUE DEL LIBRO

En esta obra se muestran numerosas técnicas de representación de datos. El contexto de las mismas se engloban en los siguientes principios:

1. Cada estructura de datos tiene sus costes y sus beneficios. Los programadores y diseñadores necesitan una comprensión rigurosa y completa de cómo evaluar los costes y beneficios para adaptarse a los nuevos retos que afronta la construcción de la aplicación. Estas propiedades requieren un conocimiento o comprensión de los principios del análisis de algoritmos y también una consideración práctica de los efectos significativos del medio físico empleado (p.e. datos almacenados en un disco frente a memoria principal).
2. Los temas relativos a costes y beneficios se consideran dentro del concepto de elemento de compensación. Por ejemplo, es bastante frecuente reducir los requisitos de tiempo en beneficio de un incremento de requisitos de espacio en memoria o viceversa.
3. Los programadores no deben reinventar la rueda continuamente. Por consiguiente, los estudiantes necesitan aprender las estructuras de datos utilizadas junto con los algoritmos correspondientes.
4. Los datos estructurados siguen a las necesidades. Los estudiantes deben aprender a evaluar primero las necesidades de la aplicación, a continuación, encontrar una estructura de datos en correspondencia con sus funcionalidades.

Esta edición, fundamentalmente, describe *estructuras de datos*, métodos de organización de grandes cantidades de datos y *algoritmos* junto con el *análisis de los mismos*, en esencia estimación del tiempo de ejecución de algoritmos. A medida que las computadoras se vuelven más y más rápidas, la necesidad de programas que pueden manejar grandes cantidades de entradas se vuelve más críticas y su eficiencia aumenta a medida que estos programas pueden manipular más y mejores organizaciones de datos. Analizando un algoritmo antes de que se codifique realmente, los estudiantes pueden decidir si una determinada solución será factible y rigurosa. Por ejemplo se pueden ver cómo diseños e implementaciones cuidadas pueden reducir los costes en tiempo y memoria de algoritmos. Por esta razón, se dedica un capítulo en exclusiva a tratar los conceptos fundamentales de *análisis de algoritmos*, y en un gran número de algoritmos se incluyen explicaciones de tiempos de ejecución para poder medir la complejidad y eficiencia de los mismos.

El método didáctico que sigue nuestro libro ya lo hemos seguido en otras obras nuestras y busca preferentemente enseñar al lector a pensar en la resolución de un problema siguiendo un determinado método ya conocido o bien creado por el propio lector, una vez esbozado el método, se estudia el algoritmo correspondiente junto con las etapas que pueden resolver el problema. A continuación se escribe el algoritmo, en ocasiones en *pseudocódigo* que al ser en español facilitará el aprendizaje al lector, y siempre en C++ para que el lector pueda verificar su programa antes de introducirlos en la computadora; se incluyen a veces la salida en pantalla resultante de la ejecución correspondiente en la máquina.

Uno de los objetivos fundamentales del libro es enseñar al estudiante, simultáneamente, buenas reglas de programación y análisis de algoritmos de modo que puedan desarrollar los programas con la mayor eficiencia posible.

## EL LIBRO COMO TEXTO DE REFERENCIA UNIVERSITARIA Y PROFESIONAL

El estudio de *Algoritmos* y de *Estructuras de Datos* son disciplinas académicas que se incorporan a todos los planes de estudios universitarios de Ingeniería e Ingeniería Técnica en Informática, Ingeniería de Sistemas Computacionales y Licenciaturas en Informática, así como a los planes de estudio de Informática en Formación Profesional y en institutos politécnicos. Suele considerarse también a estas disciplinas como ampliaciones de las asignaturas de Programación, en cualquiera de sus niveles.

En el caso de España, los actuales planes de estudios y los futuros —contemplados en la Declaración de Bolonia (EEES, Espacio Europeo de Educación Superior)—, de Ingeniería Técnica en Informática e Ingeniería Informática, contemplan materias troncales relativas tanto a Algoritmos como a Estructuras de Datos. Igual sucede en los países iberoamericanos donde también es común incluir estas disciplinas en los *currículum* de carreras de Ingeniería de Sistemas y Licenciaturas en Informática. ACM, la organización profesional norteamericana más prestigiosa a nivel mundial, incluye en las recomendaciones de sus diferentes currículos y carreras relacionadas con informática el estudio de materias de algoritmos y estructuras de datos. En el conocido *Computing Curricula* de 1992 se incluyen descriptores recomendados de *Programación y Estructura de Datos*, y en los últimos currículos publicados, *Computing Curricula* 2001 y 2005, se incluyen en el área **PF** de *Fundamentos de Programación (Programming Fundamentals, PF1 a PF4)*, **AL** de *Algoritmos y Complejidad (Algorithms and Complexity, AL1 a AL3)*. En este libro se ha incluido los descriptores más importantes tales como *Algoritmos y Resolución de Problemas, Estructuras de datos fundamentales, Recursión, Análisis de algoritmos básicos y estrategias de algoritmos*. Además se incluyen un estudio de algoritmos de estructuras discretas tan importantes como *Árboles y Grafos*.

## ORGANIZACIÓN DEL LIBRO

El libro está concebido como libro didáctico y teórico pero con un enfoque muy práctico, por lo que se incluyen gran número de ejemplos y ejercicios resueltos. Se pretende enseñar los principios básicos requeridos para seleccionar o diseñar los algoritmos y las estructuras de datos que ayudarán a resolver mejor los problemas que no a memorizar una gran cantidad de implementaciones. Los lectores deben tener conocimientos a nivel de iniciación o nivel medio en programación. Es deseable, haber cursado al menos un curso de un semestre de introducción a los algoritmos y a la programación, con ayuda de alguna herramienta de programación, preferentemente en lenguaje C++, pero podría bastar un curso de introducción a los algoritmos y programación; pensando en estos lectores en la página web oficial del curso podrá encontrar guías didácticas de introducción al lenguaje C++. El libro busca de modo prioritario enseñar al lector técnicas de programación de algoritmos y estructuras de datos. Se pretende aprender a programar practicando el análisis de los problemas y su codificación en C++.

El libro está pensado para un curso completo anual o bien dos semestres, para ser estudiado de modo independiente —por esta razón se incluyen las explicaciones y conceptos básicos de la teoría de algoritmos y estructuras de datos— o bien de modo complementario, exclusivamente como apoyo de otros libros de teoría o simplemente del curso impartido por el maestro o profesor en su aula de clase. Para aquellos lectores que deseen contrastar o comparar el diseño de algoritmos y estructuras de datos en otros lenguajes tales como Pascal y C, desde un enfoque práctico, le enumeramos otras obras nuestras: Joyanes, L.; Centenera, P.; Sánchez, L.,

Zahonero I.; Fernández, M. *Estructuras de datos. Libro de problemas*. McGraw-Hill, 1999, con un enfoque en Pascal, Joyanes, L.; Sánchez, L.; Zahonero, I.; Fernández, M. *Estructuras de datos en C*. McGraw-Hill/Schaum, 2005, con un enfoque en C).

## Contenido

El contenido del libro sigue los programas clásicos de las disciplinas *Estructura de Datos y/o Estructuras de Datos y de la Información* respetando las directrices emanadas de los currículos del 91 y las actualizadas del 2001 y 2005 de ACM/IEEE, así como de los planes de estudio de Ingeniero Informático e Ingenieros Técnicos en Informática de España y los de Ingenieros de Sistemas y Licenciados en Informática de muchas universidades latinoamericanas.

Aunque no se ha realizado una división formal del libro en partes, se puede considerar desde el punto de vista práctico que su contenido se agrupa en cuatro grandes partes. La Parte I: **Abstracción y desarrollo del software con C++** presenta los importantes conceptos de algoritmo, tipo abstracto de datos, clases, objetos, plantillas (templates) y *genericidad*, y describe las estructuras de datos más simples tales como los *arrays* (*arreglos*) cadenas o estructuras. La Parte II: **Análisis y diseño de algoritmos (recursividad, ordenación y búsqueda)** describe el importante concepto de análisis de un algoritmo y las diferentes formas de medir su complejidad y eficiencia examina los algoritmos más utilizados en la construcción de cualquier programa tales como los relativos a búsqueda y ordenación, así como las potentes técnicas de manipulación de la recursividad. La Parte III: **Estructuras de datos lineales (abstracción de datos, listas, pilas, colas, colas de prioridad, tablas hash, la biblioteca STL, contenedores e iteradores)** constituyen una de las partes avanzadas del libro y que suele formar parte de cursos de nivel medio/alto en organización de datos. Por último, la Parte IV: **Estructuras de datos no lineales (árboles, grafos y sus algoritmos)** constituyen también una de las partes avanzadas del libro; su conocimiento y manipulación permitirán al programador obtener el máximo aprovechamiento en el diseño y construcción de sus programas. La descripción más detallada de los capítulos correspondientes se reseñan a continuación:

**Capítulo 1. Desarrollo de software. Tipos abstractos de datos.** Los tipos de datos y necesidad de su organización en estructuras de datos es la parte central de este capítulo. El tratamiento de la abstracción de datos, junto con el reforzamiento de los conceptos de algoritmos y programas, y su herramienta de representación más característica, el *pseudocódigo* completa el capítulo.

**Capítulo 2. Clases y objetos.** La programación orientada a objetos es hoy día, el eje fundamental de la programación de computadoras. Su núcleo esencial son los conceptos de clases y objetos. En el capítulo se consideran los conceptos teóricos de encapsulación de datos y tipos abstractos de datos como soporte de una clase y de un objeto; también se analizan el modo de construcción y destrucción de objetos, así como conceptos tan importantes como las funciones amiga y los miembros estáticos de una clase.

**Capítulo 3. Tipos de datos básicos. Arrays, cadenas, estructuras y tipos enumerados.** Se revisan en este capítulo los conceptos básicos de tipos de datos como fundamentos para el diseño y construcción de las clases explicadas en el Capítulo 2. Los diferentes tipos de *arrays* (*arreglos*) se describen y detallan junto con la introducción a la importante clase *string*.

**Capítulo 4. Clases derivadas: herencia y polimorfismo.** Uno de los conceptos más empleados en programación orientada a objetos y que ayudará al programador de un modo eficiente al diseño de estructura de datos son las clases derivadas. La propiedad de herencia, junto con el polimorfismo ayudan a definir con toda eficacia las clases derivadas. Otro término fundamental en POO son las clases abstractas que permitirán la construcción de clases deriva-

das. C++, es uno de los pocos lenguajes de programación orientada a objetos que soporta herencia simple y herencia múltiple, aunque en este caso particular el lector deberá estudiar con detenimiento este concepto ya que a sus grandes posibilidades también se pueden añadir grandes problemas de diseño.

**Capítulo 5. Genericidad: plantillas (*templates*).** Una de las propiedades más destacadas de C++ es el soporte de la *genericidad* y, por consiguiente, la posibilidad de ejecutar programación genérica. La definición y buen uso de las plantillas (*templates*) es uno de los objetivos de este capítulo. Diferenciar y hacer buen uso de las plantillas de clases y las plantillas de funciones son otro de los objetivos de este capítulo.

**Capítulo 6. Análisis y eficiencias de algoritmos.** El estudio de algoritmos es uno de los objetivos más ambiciosos de esta obra. El capítulo hace una revisión de sus propiedades más importantes; representación, eficiencia y exactitud. Se describe la notación *O grande* utilizada preferentemente en el análisis de algoritmos.

**Capítulo 7. Algoritmos recursivos.** La recursividad es una de las características más sobresalientes en cualquier tipo de programación, Los algoritmos recursivos abundan en la vida ordinaria y el proceso de abstracción que identifica estos algoritmos debe conducir a un buen diseño de dichos algoritmos. Los algoritmos más sobresalientes y de mayor difusión en el mundo de la programación se explican con detalle en el capítulo. Así se describen algoritmos como: *mergesort* (ordenación por mezclas), *backtracking* (vuelta atrás) y otros.

**Capítulo 8. Algoritmos de ordenación y búsqueda.** Las operaciones más frecuentes en el proceso de estructura de datos, son: la ordenación y búsqueda de datos específicos. Los algoritmos más populares y eficientes de proceso de estructuras de datos internas se describen en el capítulo junto con un análisis de su complejidad.

**Capítulo 9. Algoritmos de ordenación de archivos.** Los archivos (ficheros) de datos son, posiblemente, las estructuras de datos más diseñadas y utilizadas por los programadores de aplicaciones y programadores de sistemas. Los conceptos de flujos y archivos de C++ junto con los métodos clásicos y eficientes de ordenación de archivos se describen en profundidad en el capítulo.

**Capítulo 10. Listas.** Una lista enlazada es una estructura de datos lineal de gran uso en la vida diaria de las personas y de las organizaciones. Su implementación mediante listas enlazadas es el objetivo central de este capítulo. Variantes de las listas enlazadas simples como doblemente enlazadas y circulares, son también, motivo de estudio en el capítulo.

**Capítulo 11. Pilas.** La pila es una estructura de datos simple, y cuyo concepto forma también parte, como las listas, en un elevado porcentaje, de la vida diaria de las personas y organizaciones. El tipo de dato *Pila* se puede implementar con arrays o con listas enlazadas y describe ambos algoritmos y sus correspondientes implementaciones en C++.

**Capítulo 12. Colas.** Al igual que las pilas, las colas conforman otra estructura que abunda en la vida ordinaria. La implementación del **TAD** (*Tipo Abstracto de Dato*) cola se puede hacer con arrays (arreglos), listas enlazadas e incluso listas circulares. Así mismo se analiza también en el capítulo el concepto de *bicola* o cola de doble entrada..

**Capítulo 13. Cola de prioridades y montículos.** Un tipo especial de cola, la cola de prioridades, utilizado en situaciones especiales, para resolución de problemas, junto con el concepto de montículo (*heap*, en inglés) se analizan detalladamente, junto con un método de ordenación por montículos muy eficiente, sobre todo en situaciones complejas y difíciles. Asimismo se analiza en el capítulo el concepto de montículo binomial.

**Capítulo 14. Tablas de dispersión: Funciones *hash*.** Las tablas aleatorias *hash* junto con los problemas de resolución de colisiones y los diferentes tipos de direccionamiento conforman este capítulo.

**Capítulo 15. Biblioteca estándar de plantillas STL.** El importante concepto de biblioteca de plantillas de clases se estudia en este capítulo. En particular, la biblioteca **STL** de C++. Los *contenedores e iteradores* son dos términos importantes para la programación genérica con plantillas y su conocimiento y diseño son muy importantes en la formación del programador.

**Capítulo 16. Árboles. Árboles binarios y árboles ordenados.** Los árboles son estructuras de datos no lineales y jerárquicas muy notables. Estas estructuras son notablemente de gran importancia en programación avanzada. Los árboles binarios y los árboles binarios de búsqueda se describen con rigor y profundidad por su importancia en el mundo actual de la programación tanto tradicional (*fuera de línea*) como en la Web (*en línea*).

**Capítulo 17. Árboles de búsqueda equilibrados. Árboles B.** Este capítulo se dedica a la programación avanzada de árboles de búsqueda equilibrada y árboles B. Estas estructuras de datos son complejas y su diseño y construcción requiere de estrategias y métodos eficientes para su implementación; sin embargo su uso puede producir grandes mejoras al diseño y construcción de programas que sería muy difícil por otros métodos.

**Capítulo 18. Grafos.** Los grafos son una de las herramientas más empleadas en matemáticas, estadística, investigación operativa y en numerosos campos científicos. El estudio de la teoría de Grafos se realiza fundamentalmente como elemento de *Matemática Discreta o Matemática Aplicada*. El conocimiento profundo de la teoría de grafos junto con los algoritmos de implementación es fundamental para conseguir el mayor rendimiento de las operaciones con datos, sobre todo si estos son complejos en su organización. Un programador de alto nivel no puede dejar de conocer en toda su profundidad la teoría de grafos y sus operaciones

Los **Anexos A, B, C y D** los puede encontrar el lector en la página web oficial del libro y forman parte de *Lecturas recomendadas* a todos aquellos lectores que deseen profundizar en programación avanzada de Árboles y Grafos.

**Anexo A.** Eliminación de árboles AVL (*Lectura recomendada del Capítulo 17*).

**Anexo B.** Eliminación de árboles B (*Lectura recomendada del Capítulo 17*).

**Anexo C.** Listas de adyacencia. Puntos de articulación de un grafo (*Lectura recomendada del Capítulo 18*).

**Anexo D.** Grafos para redes de flujo (*Lectura recomendada del Capítulo 18*).

## CÓDIGO C++ DISPONIBLE

Los códigos en C++ C de todos los programas de este libro están disponibles en la Web (Internet) —en formato Word para que puedan ser utilizados directamente y evitar su “teclado” en el caso de los programas largos, o bien simplemente, para seleccionar, recortar, modificar... por el lector a su conveniencia, a medida que avanza en su formación—. Estos códigos fuente se encuentran en la página oficial del libro <http://www.mch.es/joyanes>.

## AGRADECIMIENTOS

Muchos profesores y colegas españoles y latinoamericanos nos han alentado a escribir esta obra, continuación/complemento de nuestra antiguas y todavía disponibles en librería, *Estructura de Datos* cuyo enfoque era en el clásico lenguaje Pascal y Algoritmos y estructuras de datos una perspectiva en C. A todos ellos queremos mostrarles nuestro agradecimiento y como siempre brindarles nuestra colaboración si así lo desean.

A los muchos instructores, maestros y profesores tanto amigos como anónimos de Universidades e Institutos Tecnológicos y Politécnicos de España y Latinoamérica que siempre apoyan nuestras obras y a los que desgraciadamente nunca podremos agradecer individualmente ese apoyo; al menos que conste en este humilde homenaje, nuestro eterno agradecimiento y reconocimiento por ese cariño que siempre prestan a nuestras obras. Como saben aquellos que nos conocen siempre estamos a su disposición en la medida que, físicamente, nos es posible. Gracias a todos, esta obra es posible, en un porcentaje muy alto, por vuestra ayuda y colaboración.

Y como no, a los estudiantes, a los lectores autodidactas y no autodidactas, que siguen nuestras obras. Su apoyo es un gran acicate para seguir nuestra tarea. También gracias queridos lectores.

Pero si importantes son en esta obra, nuestros colegas y lectores españoles y latinoamericanos, no podemos dejar de citar al equipo humano que desde la editorial siempre cuida nuestras obras y sobre todo nos dan consejos, sugerencias, propuestas, nos “soportan” nuestros retrasos, nuestros “cambios” en la redacción, etc. **A Carmelo Sánchez** nuestro editor —y sin embargo amigo— de McGraw-Hill, que en esta ocasión, para no ser menos, nos ha vuelto a asesorar tanto en la fase de realización como en todo el proceso editorial hasta su publicación final. Por último a nuestro nuevo editor y amigo José Luis García Jurado que se ha hecho cargo de la fase final de la edición de este libro y que en tan breve espacio de tiempo nos ha prestado todo su apoyo, comprensión y *saber hacer*.

Madrid, marzo de 2007.

LOS AUTORES