

# Prefacio a la cuarta edición

La informática y las ciencias de la computación en los primeros años del siglo XXI vienen marcadas por los avances tecnológicos de la pasada década. Los más de veinte años de vida de la computadora personal (**PC**) y los más de cincuenta años de la informática/computación tradicional vienen acompañados de cambios rápidos y evolutivos en las disciplinas clásicas. El rápido crecimiento del mundo de las redes y, en consecuencia, la **World Wide Web** hacen revolucionarios a estos cambios y afectan al cuerpo de conocimiento de los procesos educativos y profesionales.

Así, como declara ACM en su informe final (15 de diciembre de 2001) CC2001 **Computer Science**, la formación en carreras de informática, ciencias de la computación o ingeniería de sistemas deberá prestar especial importancia a temas tales como:

- **Algoritmos y estructuras de datos.**
- **La World Wide Web y sus aplicaciones.**
- **Las tecnologías de red y en especial aquellas basadas en TCP/IP.**
- **Gráficos y multimedia.**
- **Sistemas empotrados.**
- **Bases de datos relacionales.**
- **Inteoperabilidad.**
- **Programación orientada a objetos.**
- **Interacción Persona-Máquina.**
- ...

ACM, velando porque sus miembros —al fin y al cabo, representantes de la comunidad informática mundial— sigan los progresos científicos y, en consecuencia, culturales y sociales derivados de las innovaciones tecnológicas, ha trabajado durante muchos años en un nuevo modelo curricular de la carrera de ingeniero informático o ingeniero de sistemas (*computer sciences*) y a finales del 2001 publicó su anteproyecto de currículo profesional (informe CC2001). El cuerpo de conocimiento incluido en este informe contempla una estructura con 14 grupos de conocimiento que van desde las Estructuras Discretas a la Ingeniería de Software pasando por Fundamentos de Programación (*Programming Fundamentals*, **PF**). En nuestro caso, y para reconocimiento de las citadas palabras, nos cabe el honor de que nuestra obra, que ha cumplido recientemente VEINTE AÑOS DE VIDA, tenga el mismo título en español (**Fundamentos de Programación, Programming Fundamentals**) que uno de los 14 grupos de conocimiento ahora recomendados como disciplina fundamental por la ACM dentro de su currículo de *Computer Science*. Así, en el citado currículo se incluyen descriptores tales como: *construcciones de programación fundamentales, algoritmos y resolución de problemas, estructuras de datos fundamentales y recursividad o recursión*.

También los planes de estudios de ingeniería informática en España (superior y técnicas de sistemas o de gestión) y de otras ingenierías (telecomunicaciones, industriales, etc.) y de ingeniería de sistemas en Latinoamérica, incluyen asignaturas tales como *Metodología de la Programación, Fundamentos de Programación* o *Introducción a la Programación*.

Del estudio comparado de las citadas recomendaciones curriculares, así como de planes de estudios conocidos de carreras de ingeniería de sistemas y licenciaturas en informática de universidades latinoamericanas, hemos llegado a la consideración de que la iniciación de un estudiante de ingeniería informática o de ingeniería de sistemas en las técnicas de programación del siglo XXI requiere no sólo del aprendizaje clásico del diseño de algoritmos y de la comprensión de las técnicas orientadas a objetos, sino un **método de transición hacia tecnologías de Internet**. Por

otra parte, un libro dirigido a los primeros cursos de introducción a la programación exige no sólo la elección de un lenguaje de programación adecuado si no, y sobre todo, «proporcionar al lector las herramientas para desarrollar programas correctos, eficientes, bien estructurados y con estilo, que sirvan de base para la construcción de unos fundamentos teóricos y prácticos que le permitan continuar con éxito sus estudios de los cursos superiores de su carrera, así como su futura especialización en ciencias e ingeniería». En consecuencia y de modo global, la obra pretende enseñar técnicas de **análisis, diseño y construcción de algoritmos, estructuras de datos y objetos**, así como reglas para la escritura de **programas, eficientes tanto estructurados, fundamentalmente, como orientados a objetos**. De modo complementario, y no por ello menos importante, se busca también enseñar al alumno técnicas de abstracción que le permitan resolver los problemas de programación del modo más sencillo y racional pensando no sólo en el aprendizaje de reglas de sintaxis y construcción de programas, sino, y sobre todo, aprender a pensar para conseguir la resolución del problema en cuestión de forma clara, eficaz y fácil de implementar en un lenguaje de programación y su ejecución posterior en una computadora u ordenador.

## OBJETIVOS DEL LIBRO

El libro pretende enseñar a programar utilizando conceptos fundamentales, tales como:

1. *Algoritmos* (conjunto de instrucciones programadas para resolver una tarea específica).
2. *Datos* (una colección de datos que se proporcionan a los algoritmos que se han de ejecutar para encontrar una solución: los datos se organizarán en *estructuras de datos*).
3. *Objetos* (conjunto de datos y algoritmos que los manipulan, encapsulados en un tipo de dato conocido como **objeto**).
4. *Clases* (tipos de objetos con igual estado y comportamiento, o dicho de otro modo, los mismos atributos y operaciones).
5. *Estructuras de datos* (conjunto de organizaciones de datos para tratar y manipular eficazmente datos homogéneos y heterogéneos).
6. *Temas avanzados* (recursividad, métodos avanzados de ordenación y búsqueda, relaciones entre clases, etc.).

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 representan como una serie de funciones 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 —normalmente— conocida como **programación estructurada**. La programación con soporte en C y Pascal proporciona el paradigma *procedimental* tradicional con un énfasis en funciones, plantillas de funciones y algoritmos genéricos.

En la década de los ochenta, el enfoque del diseño de programas se desplazó desde el paradigma *procedimental* al *orientado a objetos* apoyado en los tipos abstractos de datos (TAD). Desde entonces conviven los dos paradigmas. En el paradigma orientado a objetos un problema se modela un conjunto de abstracciones de datos (tipos de datos) conocidos 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.

Así pues, en resumen, los objetivos fundamentales de esta obra son: aprendizaje y formación en *algoritmos* y *programación estructurada*, *estructuras de datos* y *programación orientada a objetos*. Evidentemente, la mejor forma de aprender a programar es con la ayuda de un lenguaje de programación. Apoyados en la experiencia de nuestras tres primeras ediciones y en los resultados conseguidos con nuestros alumnos y lectores, hemos seguido apostando por utilizar un lenguaje algorítmico —**pseudolenguaje**— que apoyado en un *pseudocódigo* (*seudocódigo*) en español nos permitiera enseñar al alumno las técnicas y reglas de programación y que su aprendizaje fuese rápido y gradual. Naturalmente, además del **pseudocódigo** hemos utilizado las otras herramientas de programación clásicas y probadas como los **diagramas de flujo** o los **diagramas N-S**.

En esta cuarta edición hemos seguido utilizando el mismo lenguaje algorítmico al que ya se le añadió las construcciones y estructuras necesarias para incorporar en sus especificaciones las técnicas orientadas a objetos. Así, hemos seguido utilizando nuestro lenguaje UPSAM inspirado en los lenguajes estructurados por excelencia, C, Pascal y FORTRAN, y le hemos añadido las propiedades de los lenguajes orientados a objetos tales como C++, Java y C#, en la mejor armonía posible y con unas especificaciones que hemos incluido en los Apéndices I a V del sitio de Internet del libro ([www.mhe.es/joyanes](http://www.mhe.es/joyanes)), como ya hiciéramos también en las tres primeras ediciones.

## EL LIBRO COMO HERRAMIENTA DOCENTE

En el contenido de la obra hemos tenido en cuenta no sólo las directrices de los planes de estudio españoles de ingeniería informática (antigua licenciatura en informática), ingeniería técnica en informática y licenciatura en ciencias de la computación, sino también de ingenierías, tales como industriales, telecomunicaciones, agrónomos o minas, o las más jóvenes, como ingeniería en geodesia, ingeniería química o ingeniería telemática. Nuestro conocimiento del mundo educativo latinoamericano nos ha llevado a pensar también en las carreras de ingeniería de sistemas computacionales y las licenciaturas en informática y en sistemas de información, como se las conoce en Latinoamérica.

El contenido del libro se ha escrito pensando en un posible desarrollo de dos cuatrimestres o semestres o en un año completo, y siguiendo los descriptores (temas centrales) recomendados en las directrices del Ministerio de Educación y Ciencia español para los planes de estudio de **Ingeniería en Informática**, **Ingeniería Técnica en Informática** e **Ingeniería Técnica en Informática**, y los planes de estudios de **Ingeniería de sistemas** y **Licenciaturas en Informática** con el objeto de poder ser utilizado también por los estudiantes de primeros semestres de estas carreras. Igualmente pretende seguir las directrices que contiene el Libro Blanco de Informática en España para la futura carrera **Grado en Ingeniería Informática** adaptada al futuro Espacio Europeo de Educación Superior (*Declaración de Bolonia*). Desde el punto de vista de currículum, se pretende que el libro pueda servir para asignaturas tales como *Introducción a la Programación*, *Fundamentos de Programación* y *Metodología de la Programación*, y también, si el lector o el maestro/profesor lo consideran oportuno, para cursos de *Introducción a Estructuras de Datos* y/o *Programación Orientada a Objetos*.

No podíamos dejar de lado las recomendaciones de la más prestigiosa organización de informáticos del mundo, ACM, anteriormente citada. Se estudió en su momento los *Curricula de Computer Science* vigentes durante el largo periodo de elaboración de esta obra (68, 78 y 91), pero siguiendo la evolución del último *curricula*, por lo que tras su publicación el 15 de diciembre de 2001 del «*Computing Curricula 2001 Computer Science*» estudiamos el citado currículum y durante la escritura de nuestra tercera edición consideramos cómo introducir también sus directrices más destacadas; como lógicamente no se podía seguir todas las directrices de su cuerpo de conocimiento al pie de la letra, analizamos en profundidad las unidades más acordes con nuestros planes de estudios: *Programming Fundamentals (PF)*, *Algorithms and Complexity (AL)* y *Programming Languages (PL)*. Por suerte nuestra obra incorporaba la mayoría de los temas importantes recomendados en las tres citadas unidades de conocimiento, en gran medida de la unidad **PF**, y temas específicos de programación orientado a objetos y de análisis de algoritmos de las unidades **PL** y **AL**.

El contenido del libro abarca los citados programas y comienza con la introducción a los algoritmos y a la programación, para llegar a estructuras de datos y programación orientada a objetos. Por esta circunstancia la estructura del curso no ha de ser secuencial en su totalidad sino que el profesor/maestro y el alumno/lector podrán estudiar sus materias en el orden que consideren más oportuno. Esta es la razón principal por la cual el libro se ha organizado en cuatro partes y en cuatro apéndices y ocho apéndices en Internet.

Se trata de describir los *dos paradigmas* más populares en el mundo de la programación: el *procedimental* y el *orientado a objetos*. Los cursos de programación en sus niveles inicial y medio están evolucionando para aprovechar las ventajas de nuevas y futuras tendencias en ingeniería de software y en diseño de lenguajes de programación, específicamente diseño y programación orientada a objetos. Numerosas facultades y escuelas de ingenieros, junto con la **nueva formación profesional** (*ciclos formativos de nivel superior*) en España y en Latinoamérica, están introduciendo a sus alumnos en la programación orientada a objetos, inmediatamente después del conocimiento de la programación estructurada, e incluso —en ocasiones antes—. Por esta razón, una metodología que se podría seguir sería impartir un curso de *algoritmos e introducción a la programación* (parte I) seguido de *estructuras de datos* (parte II) y luego seguir con un segundo nivel de *programación avanzada y programación orientada a objetos* (partes II, III y IV) que constituyen las cuatro partes del libro. De modo complementario, si el alumno o el profesor/maestro lo desea se puede practicar con algún lenguaje de programación estructurado u orientado a objetos, ya que pensando en esa posibilidad se incluyen en la página web del libro, apéndices con guías de sintaxis de los lenguajes de programación más populares hoy día, C, C++, Java y C# e incluso se han mantenido resúmenes de guías de sintaxis de Pascal, FORTRAN y Modula-2.

Uno de los temas más debatidos en la educación en informática o en ciencias de la computación (*Computer Sciences*) es el rol de la programación en el currículo introductorio. A través de la historia de la disciplina —como fielmente reconoce en la introducción del Capítulo 7 relativo a cursos de introducción, ACM en su Computing Curricula 2001 [ACM91]— la mayoría de los cursos de introducción a la informática se han centrado principalmente en el desarrollo de habilidades o destrezas de programación. La adopción de un curso de introducción a la programación proviene de una serie de factores prácticos e históricos e incluyen los siguientes temas:

- La programación es una técnica esencial que debe ser dominada por cualquier estudiante de informática. Su inserción en los primeros cursos de la carrera asegura que los estudiantes tengan la facilidad necesaria con la programación para cuando se matriculan en los cursos de nivel intermedio y avanzado.
- La informática no se convirtió en una disciplina académica hasta después que la mayoría de las instituciones ha desarrollado un conjunto de cursos de programación introductorias que sirvan a una gran audiencia.
- El modelo de aprendizaje en programación siguió desde el principio las tempranas recomendaciones del Currículo 68 [ACM68] que comenzaba con un curso denominado «**Introducción a la Computación**», en la que la abrumadora mayoría de los temas estaban relacionados con la programación. Con posterioridad, y en el currículum del 78 de la ACM [ACM78] definía a estos cursos como «**Introducción a la Programación**» y se les denominó CS1 y CS2; hoy día se le sigue denominando así por la mayoría de los profesores y universidades que seguimos criterios emanados de la ACM.

La fluidez en un lenguaje de programación es prerequisite para el estudio de las ciencias de la computación. Ya en 1991 el informe CC1991 de la ACM reconocía la exigencia del conocimiento de un lenguaje de programación.

- Los programas de informática deben enseñar a los estudiantes cómo usar al menos bien un lenguaje de programación. Además, recomendamos que los programas en informática deben enseñar a los estudiantes a ser competentes en lenguajes y que hagan uso de al menos dos paradigmas de programación. Como consecuencia de estas ideas, el currículo 2001 de la ACM contempla la necesidad de conceptos y habilidades que son fundamentales en la práctica de la programación con independencia del paradigma subyacente. Como resultado de este pensamiento, el área de Fundamentos de Programación incluye unidades sobre conceptos de programación, estructuras de datos básicas y procesos algorítmicos. Además de estas unidades fundamentales se requieren conceptos básicos pertenecientes a otras áreas, como son Lenguajes de Programación (**PL**, *Programming Languages*) y de Ingeniería de Software (**SE**, *Software Engineering*).

Los temas fundamentales que considera el currículo 2001 son:

- construcciones fundamentales de programación,
- algoritmos y resolución de problemas,
- estructuras de datos fundamentales,
- recursión o recursividad,
- programación controlada por eventos,

de otras áreas, Lenguajes de Programación (PL), destacar:

- revisión de lenguajes de programación,
- declaraciones y tipos,
- mecanismos de abstracción,
- programación orientada a objetos,

y de Ingeniería de Software (SE):

- diseño de software,
- herramientas y entornos de software,
- requisitos y especificaciones de software.

Este libro se ha escrito pensando en que pudiera servir de referencia y guía de estudio para un primer curso de *introducción a la programación*, con una segunda parte que, a su vez, sirviera como continuación y para un posible segundo curso, de *estructuras de datos y programación orientada a objetos*. El objetivo final que busca es, no sólo

describir la sintaxis de un lenguaje de programación, sino, y sobre todo, mostrar las características más sobresalientes del lenguaje algorítmico y a la vez enseñar técnicas de programación estructurada y orientada a objetos. Así pues, los objetivos fundamentales son:

- Énfasis fuerte en el análisis, construcción y diseño de programas.
- Un medio de resolución de problemas mediante técnicas de programación.
- Una introducción a la informática y a las ciencias de la computación usando una herramienta de programación denominada **pseudocódigo**.
- Aprendizaje de técnicas de construcción de programas estructurados y una iniciación a los programas orientados a objetos.

Así se tratará de enseñar las técnicas clásicas y avanzadas de programación estructurada, junto con técnicas orientadas a objetos. La programación orientada a objetos no es la panacea universal de un programador del siglo XXI, pero le ayudará a realizar tareas que, de otra manera, serían complejas y tediosas.

El contenido del libro trata de proporcionar soporte a un año académico completo (dos semestres o cuatrimestres), alrededor de 24 a 32 semanas, dependiendo lógicamente de su calendario y planificación. Los diez primeros capítulos pueden comprender el primer semestre y los restantes capítulos pueden impartirse en el segundo semestre. Lógicamente la secuencia y planificación real dependerá del maestro o profesor que marcará y señalará semana a semana la progresión que él considera lógica. Si usted es un estudiante autodidacta, su propia progresión vendrá marcada por las horas que dedique al estudio y al aprendizaje con la computadora, aunque no debe variar mucho del ritmo citado al principio de este párrafo.

## EL LENGUAJE ALGORÍTMICO DE PROGRAMACIÓN UPSAM 2.0

Los cursos de introducción a la programación se apoyan siempre en un lenguaje de programación o en un **pseudolenguaje** sustentado sobre un **pseudocódigo**. En nuestro caso optamos por esta segunda opción: el pseudolenguaje con la herramienta del pseudocódigo en español o castellano. Desde la aparición de la primera edición, allá por finales de los años ochenta, apostamos fundamentalmente por el pseudocódigo para que junto con los diagramas de flujo y diagramas N-S nos ayudara a explicar técnicas de programación a nuestros alumnos. Con ocasión de la publicación de la segunda edición (año 1996) no sólo seguimos apostando otra vez, y fundamentalmente, por el pseudocódigo sino que juntamos los trabajos de todos los profesores del antiguo Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software de la Facultad de Informática y Escuela Universitaria de Informática de la Universidad Pontificia de Salamanca en el campus de Madrid y le dimos forma a la versión 1.0 del lenguaje UPSAM<sup>1</sup>.

La versión 1.0 de UPSAM se apoyaba fundamentalmente en los lenguajes de programación Pascal (Turbo Pascal) y C, con referencias a FORTRAN, COBOL y BASIC, y algunas ideas del lenguaje C++ que comenzaba ya a ser un estándar en el mundo de la programación. Java nació en 1995 mientras el libro estaba en imprenta, aunque la edición de nuestra obra se publicó en 1996. Por ello el lenguaje algorítmico sólo contenía reglas de sintaxis y técnicas de programación estructurada. Sin embargo, en la segunda mitad de los noventa, C++ se acabó por imponer como estándar en el mundo de la programación, y Java iba asentándose como lenguaje de programación para la Web e Internet, por lo que la nueva especificación algorítmica debería contemplar estos lenguajes emergentes, consolidado en el caso de C++.

En los primeros años del siglo XXI, Java se terminó de implantar como lenguaje universal de Internet y también como lenguaje estándar para el aprendizaje de programación de computadoras; pero además, y sobre todo, para cursos de programación orientada a objetos y estructuras de datos. De igual modo, C#, el lenguaje creado y presentado por Microsoft a comienzos del año 2000, como competidor de Java y extensión de C/C++, también ha pasado a ser una realidad del mundo de construcción de software y aunque no ha tenido tanta aceptación en el mundo del aprendizaje educativo de la programación, sí se ha implantado como lenguaje de desarrollo principalmente para plataformas .Net. Por todo ello, la versión 2.0 del lenguaje algorítmico UPSAM que presentamos ya en la tercera edición de esta obra, se apoyó fundamentalmente en C y C++, así como en Java y C#, aunque hemos intentado no olvidar sus viejos orígenes apoyados en Pascal y Turbo Pascal, y algo menos en FORTRAN y COBOL.

Así pues, la versión 2.0 del lenguaje UPSAM presentada en 2003 y hoy revisada con la actualización 2.1, se apoyaba en los quince años de vida (originalmente dicha versión se conocía como UPS; hoy hace ya veinte años) y

---

<sup>1</sup> En los prólogos de la segunda y tercera edición, se referencian los nombres de todos los profesores que intervinimos en la redacción de la especificación de la versión 1.0 y 2.0 y, por consiguiente, figuran como autores de dicha especificación.

se construyó como mejora de la versión 1.0 presentada en la primera y segunda edición de esta obra. Para completar la versión algorítmica, en el Apéndice D, se incluye una guía rápida de referencia del lenguaje C, “padre de todos los lenguajes modernos”, y en el portal del libro ([www.mhe.es/joyanes](http://www.mhe.es/joyanes)) se incluyen guías de todos los lenguajes de programación considerados en la especificación UPSAM 2.1.

En la versión 2.0 trabajamos todos los profesores, de aquel entonces, del área de programación de la Universidad Pontificia de Salamanca en el campus de Madrid, pero de un modo muy especial los profesores compiladores de todas las propuestas de nuestros compañeros, además del autor de esta obra. Especialmente quiero destacar a los profesores **Luis Rodríguez Baena**, **Víctor Martín García**, **Lucas Sánchez García**, **Ignacio Zahonero Martínez** y **Matilde Fernández Azuela**. Cabe destacar también la gran contribución y revisión de la versión 2.0 del profesor Joaquín Abeger, y restantes compañeros del entonces departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software de la Facultad de Informática y Escuela Universitaria de Informática de la Universidad Pontificia de Salamanca en el campus de Madrid, cuyas aportaciones han sido fundamentales para que la versión 2.0 viera la luz.

Deseo resaltar de modo muy especial la gran aportación práctica y de investigación a todas las versiones de UPSAM realizada por los profesores de Procesadores de Lenguajes (Compiladores), **María Luisa Díez Plata** y **Enrique Torres Franco**, que durante numerosos cursos académicos han implementado partes del lenguaje en un traductor, en las prácticas y proyectos de compiladores diseñados por y para los alumnos de la citada asignatura de cuarto curso (séptimo y octavo semestre), en algunos casos funcionando con prototipos. De igual forma, deseo expresar mi agradecimiento a la profesora de la Universidad de Alicante, **Rosana Latorre Cuerda**, que no sólo ha apoyado continuamente esta obra sino que además ha utilizado el lenguaje algorítmico UPSAM en sus clases de compiladores y de programación.

Muchos —innumerables diría yo— son los profesores, colegas, y sin embargo amigos, que me han apoyado, revisado y dado ideas para mejorar las sucesivas ediciones del lenguaje algorítmico. Es imposible enumerarlos a todos aquí —y asumo el enorme riesgo de haber olvidado a muchos, a los que ya pido y presento mis disculpas— por lo que trataré de enumerar al menos a aquellos que me vienen a la memoria en este instante —que los duendes de las imprentas me exigen sea ya— y reitero mis disculpas a todos los que ahora no figuran (y trataré de que estén en el sitio oficial del libro y en próximas ediciones). Gracias infinitas a todos: María Eugenia Valesani (Universidad Nacional del Nordeste, Argentina), Héctor Castán Rodríguez (Universidad Pontificia de Salamanca, campus de Madrid), Vidal Alonso Secades, Alfonso López Rivero y Miguel Ángel Sánchez Vidales (Universidad Pontificia de Salamanca, campus de Salamanca), David La Red (Universidad Nacional del Nordeste, Argentina), Óscar Sanjuán Martínez y Juan Manuel Cueva Lovelle (Universidad de Oviedo), Darwin Muñoz (Unibe, República Dominicana), Ricardo Moreno (Universidad Tecnológica de Pereira, Colombia), Miguel Cid (INTEC, República Dominicana), Julio Perrier (Universidad Autónoma de Santo Domingo, República Dominicana), Quinta Ana Pérez (ITLA, República Dominicana), Jorge Torres (Tecnológico de Monterrey, campus de Querétaro, México), Augusto Bernuy (UTP, Lima, Perú), Juan José Moreno (Universidad Católica de Uruguay), Manuel Pérez Cota (Universidad de Vigo), Ruben González (Universidad Pontificia de Salamanca, campus de Madrid), José Rafael García-Bermejo Giner (Universidad de Salamanca), Víctor Hugo Medina García y Giovanni Tarazona (Universidad Distrital, Bogotá, Colombia), Luz Mayela Ramírez y mi querido Jota Jota (Universidad Católica de Colombia), Alveiro (de la Universidad Cooperativa de Colombia), Marcelo (de la Universidad de Caldas), Sergio Ríos (Universidad Pontificia de Salamanca, campus de Salamanca), Rosana Latorre Cuerda (Universidad de Salamanca)... bueno, son tantos que necesitaría quizá un tiempo infinito para nombrarlos a todos. A los ausentes, mis disculpas; y a todos, gracias: esta obra es, en gran parte, vuestra, con todas vuestras ayudas, críticas, apoyos y con todo cuanto el corazón pueda hablar.

Espero que la versión 2.1, la actual, y la futura, 3.0, sean capaces de recoger todo el trabajo de nuestro grupo de investigación y de todos los profesores de universidades amigas.

## CARACTERÍSTICAS IMPORTANTES DEL LIBRO

*Fundamentos de programación, cuarta edición*, utiliza en cada capítulo los siguientes elementos clave para conseguir obtener el mayor rendimiento del material incluido:

- **Objetivos.** Enumera los conceptos y técnicas que el lector y los estudiantes aprenderán en el capítulo. Su lectura ayudará a los estudiantes a determinar si se han cumplido estos objetivos después de terminar el capítulo.
- **Contenido.** Índice completo del capítulo que facilita la lectura y la correcta progresión en la lectura y comprensión de los diferentes temas que se exponen posteriormente.
- **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.

- **Descripción del capítulo.** Explicación usual de los apartados correspondientes del capítulo. En cada capítulo se incluyen ejemplos y ejercicios resueltos. Los listados de los programas completos o parciales se escriben en letra “courier” con la finalidad principal de que puedan ser identificados fácilmente por el lector. Todos ellos han sido probados para facilitar la práctica del lector/alumno.
- **Conceptos clave.** Enumera los términos de computación, informáticos y de programación (terminología) más notables que se han descrito o tratado 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.** En muchos capítulos se incluyen enunciados de problemas propuestos para realizar por el alumno y que presentan una mayor dificultad que los ejercicios antes planteados. Se suelen incluir una serie de actividades y proyectos de programación que se le proponen al lector como tarea complementaria de los ejercicios.

A lo largo de todo el libro se incluyen una serie de recuadros —sombreados o no— que ofrecen al lector consejos, advertencias y reglas de uso del lenguaje y de técnicas de programación, con la finalidad de que puedan ir asimilando conceptos prácticos de interés que les ayuden en el aprendizaje y construcción de programas eficientes y de fácil lectura.

- **Recuadro.** Conceptos importantes que el lector debe considerar durante el desarrollo del capítulo.
- **Consejo.** Ideas, sugerencias, recomendaciones... al lector, con el objetivo de obtener el mayor rendimiento posible del lenguaje y de la programación.
- **Precaución.** Advertencia al lector para que tenga cuidado al hacer uso de los conceptos incluidos en el recuadro adjunto.
- **Nota.** Normas o ideas que el lector debe seguir preferentemente en el diseño y construcción de sus programas.

## ORGANIZACIÓN DEL LIBRO

El libro se ha dividido en cuatro partes a efectos de organización para su lectura y estudio gradual. Dado que el conocimiento es acumulativo, se comienza en los primeros capítulos con conceptos conceptuales y prácticos, y se avanza de modo progresivo hasta llegar a las técnicas avanzadas y a una introducción a la ingeniería de software que intentan preparar al lector/estudiante para sus estudios posteriores. Como complemento y ayuda al lector durante sus estudios y para su posterior formación profesional, se han incluido una gran cantidad de apéndices que incluyen fundamentalmente guías de sintaxis de los lenguajes de programación más populares con el objetivo de facilitar la implementación de los algoritmos en el lenguaje de programación elegido para «dialogar» con la computadora. Así mismo, y para ayudar a la preparación del aprendizaje, lecturas y estudios futuros, se ha incluido una amplia guía de recursos de programación (libros, revistas y sitios Web «URLs») que hemos consultado en la elaboración de nuestra obra y seguimos consultando también en nuestra vida profesional.

## PARTE I. ALGORITMOS Y HERRAMIENTAS DE PROGRAMACIÓN

Esta parte es un primer curso de programación para alumnos principiantes en asignaturas de introducción a la programación en lenguajes estructurados y sirve tanto para cursos introductorios de carácter semestral, tales como *Introducción a la Programación*, *Metodología de la Programación* o *Fundamentos de programación*, en primeros cursos de carreras de ingeniería informática, ingeniería de sistemas y licenciatura en informática o en sistemas de información, y asignaturas de programación de ingeniería y de ciencias. Contiene esta parte los fundamentos teóricos y prácticos relativos a la organización de una computadora y los lenguajes de programación, así como la descripción de las herramientas de programación más frecuentemente utilizadas en el campo de la programación. Se incluyen también en esta parte los elementos básicos constitutivos de un programa y las herramientas de programación utilizadas, tales como algoritmos, diagramas de flujo, etc. La segunda mitad de esta primera parte es una descripción teórico-práctica de las estructuras utilizadas para controlar el flujo de instrucciones de un programa y una descripción detallada del importante concepto de subprograma (procedimiento/función), piedra angular de la programación modular y estructurada.

**Capítulo 1. Introducción a las computadoras y los lenguajes de programación.** Las computadoras son herramientas esenciales en muchas áreas de la vida: profesional, industrial, empresarial, académica,... en realidad, en casi todos los campos de la sociedad. Las computadoras funcionan correctamente con la ayuda de los programas. Los programas se escriben mediante lenguajes de programación que previamente se han escrito en algoritmos u otras herramientas, tales como diagramas de flujo. Este capítulo introductorio describe la organización de una computadora y sus diferentes partes junto con el concepto de programa y de lenguaje de programación. Así mismo y al objeto de que el lector pueda entender los fundamentos teóricos en que se asienta la programación, se incluye una breve historia de los lenguajes de programación más influyentes y que, en el caso de la tercera edición, han servido de inspiración para la nueva versión del pseudocódigo **UPSAM 2.0**: es decir, C, C++, Java y C#, con referencias lógicas al histórico Pascal.

**Capítulo 2. Metodología de la programación y desarrollo de software.** En este capítulo se describen métodos para la resolución de problemas con computadora y con un lenguaje de programación (en nuestro caso el pseudolenguaje o lenguaje algorítmico UPSAM 2.0). Se explican las fases de la resolución de un problema junto con las técnicas de programación modular y estructurada. Se inicia en este capítulo la descripción del concepto, función y uso de algoritmo. Uno de los objetivos más importantes de este libro es el aprendizaje, diseño y construcción de algoritmos.

**Capítulo 3. Estructura general de un programa.** Enseña la organización y estructura general de un programa así como su creación y proceso de ejecución. Se describen los elementos básicos de un programa: tipos de datos, constantes, variables y entradas/salidas de datos. También se introduce al lector en la operación de asignación así como en el concepto de función interna. De igual forma se estudian los importantes conceptos de expresiones y operaciones junto con sus diferentes tipos.

**Capítulo 4. Flujo de control I: Estructuras selectivas.** Introduce al concepto de estructura de control y, en particular, estructuras de selección, tales como *si-entonces* ("if-then"), *según sea/caso de* ("switch/case"). Se describen también las estructuras de decisión anidadas. Así mismo se explica también la «denostada» sentencia *ir\_a* (goto), cuyo uso no se recomienda pero sí el conocimiento de su funcionamiento,

**Capítulo 5. Flujo de control II: Estructuras repetitivas.** El capítulo introduce las estructuras repetitivas (*mientras* ("while"), *hacer-mientras* ("do-while"), *repetir* ("repeat"), *desde/para* ("for")). Examina la repetición (*iteración*) de sentencias en detalle y compara los bucles controlados por centinela, bandera, etc. Explica precauciones y reglas de uso de diseño de bucles. Compara los tres diferentes tipos de bucles, así como el concepto de bucles anidados.

**Capítulo 6. Subprogramas (subalgoritmos): Funciones.** La resolución de problemas complejos se facilita considerablemente si se divide en problemas más pequeños (subproblemas). La resolución de estos problemas se realiza con *subalgoritmos* (subprogramas) que a su vez se dividen en dos grandes categorías: *funciones* y *procedimientos*.

## PARTE II. ESTRUCTURA DE DATOS

Esta parte es clave en el aprendizaje de técnicas de programación. Tal es su importancia que los planes de estudio de cualquier carrera de ingeniería informática o de ciencias de la computación incluye una asignatura troncal denominada *Estructura de datos*.

**Capítulo 7. Estructuras de datos I (arrays y estructuras).** Examina la estructuración de los datos en *arrays* o grupos de elementos dato del mismo tipo. El capítulo presenta numerosos ejemplos de *arrays* de uno, dos o múltiples índices. También se explican los otros tipos de estructuras de datos básicas: estructuras y registros. Estas estructuras de datos permiten encapsular en un tipo de dato definido por el usuario otros datos heterogéneos. Así mismo se describe el concepto de arrays de estructuras y arrays de registros.

**Capítulo 8. Las cadenas de caracteres.** Se examina el concepto de carácter y de cadena (*String*) junto con su declaración e inicialización. Se introducen conceptos básicos de manipulación de cadenas: lectura y asignación junto con operaciones básicas, tales como longitud, concatenación, comparación, conversión y búsqueda de caracteres y cadenas. Las operaciones de tratamiento de caracteres y cadenas son operaciones muy usuales en todo tipo de programas.

**Capítulo 9. Archivos (ficheros).** El concepto de archivo junto con su definición e implementación es motivo de estudio en este capítulo. Los tipos de archivos más usuales junto con las operaciones básicas de manipulación se estudian con detenimiento.

**Capítulo 10. Ordenación, búsqueda e intercalación.** Las computadoras emplean una gran parte de su tiempo en operaciones de búsqueda, clasificación y mezcla de datos. Los archivos se sitúan adecuadamente en dispositivos de almacenamiento externo que son más lentos que la memoria central pero que, por el contrario, tienen la ventaja de almacenamiento permanente después de apagar la computadora. Se describen los algoritmos de los métodos más utilizados en el diseño e implementación de programas.

**Capítulo 11. Ordenación, búsqueda y fusión externa (archivos).** Normalmente los datos almacenados de modo permanente en dispositivos externos requieren para su procesamiento el almacenamiento en la memoria central. Por esta circunstancia, las técnicas de ordenación y búsqueda sobre arrays y vectores comentados en el capítulo anterior necesitan una profundización en cuanto a técnicas y métodos. Una de las técnicas más importantes es la fusión o mezcla. En el capítulo se describen las técnicas de manipulación externa de archivos.

**Capítulo 12. Estructuras dinámicas lineales de datos (pilas, colas y listas enlazadas).** Una lista enlazada es una estructura de datos que mantiene una colección de elementos, pero el número de ellos no se conoce por anticipado o varía en un amplio rango. La lista enlazada se compone de elementos que contienen un valor y un puntero. El capítulo describe los fundamentos teóricos y las operaciones que se pueden realizar en la lista enlazada. También se describen los distintos tipos de listas enlazadas, tales como doblemente enlazadas y circulares. Las ideas abstractas de pila y cola se describen en el capítulo. Pilas y colas se pueden implementar de diferentes maneras, bien con vectores (arrays) o con listas enlazadas.

**Capítulo 13. Estructuras de datos no lineales (árboles y grafos).** Los árboles son otro tipo de estructura de datos dinámica y no lineal. Se estudian las operaciones básicas en los árboles junto con sus operaciones fundamentales.

**Capítulo 14. Recursividad.** El importante concepto de recursividad (propiedad de una función de llamarse a sí misma) se introduce en el capítulo junto con algoritmos complejos de ordenación y búsqueda en los que además se estudia su eficiencia.

### PARTE III. PROGRAMACIÓN ORIENTADA A OBJETOS Y UML 2.1.

Nuestra experiencia en la enseñanza de la programación orientada a objetos a estudiantes universitarios data de finales de la década de los ochenta. En este largo periodo, los primitivos y básicos conceptos de orientación a objetos se siguen manteniendo desde el punto de vista conceptual y práctico, tal y como se definieron hace treinta años. Hoy la programación orientada a objetos es una clara realidad y por ello cualquier curso de introducción a la programación aconseja, al menos, incluir un pequeño curso de orientación a objetos que puede impartirse como un curso independiente, como complemento de la Parte II o como parte de un curso completo de introducción a la programación que comienza en el Capítulo 1.

**Capítulo 15. Tipos abstractos de datos, objetos y modelado con UML 2.1.** Este capítulo describe los conceptos fundamentales de la orientación a objetos: clases, objetos y herencia. La definición y declaración de una clase junto con su organización y estructura se explican detenidamente. Se describen también otros conceptos importantes, tales como polimorfismo, ligadura dinámica y sobrecarga y un resumen de la terminología orientada a objetos.

**Capítulo 16. Diseño de clases y objetos: representaciones gráficas en UML.** Una de las tareas fundamentales de un programador es el diseño y posterior implementación de una clase y de un objeto en un lenguaje de programación. Para realizar esta tarea con eficiencia se exige el uso de una herramienta gráfica. UML es el lenguaje de modelado unificado estándar en el campo de la ingeniería de software y en el capítulo se describen las notaciones gráficas básicas de clases y objetos.

**Capítulo 17. Relaciones entre clases: Delegaciones, asociaciones, agregaciones, herencia.** En este capítulo se introducen los conceptos fundamentales de las relaciones entre clases: asociación, agregación y generalización/especialización. Se describen todas estas relaciones así como las notaciones gráficas que las representan en el lenguaje de modelado UML.

## PARTE IV. METODOLOGÍA DE LA PROGRAMACIÓN Y DESARROLLO DE SOFTWARE

En esta parte se describen reglas prácticas para la resolución de problemas mediante programación y a su posterior desarrollo de software. Estas reglas buscan proporcionar al lector reglas de puesta a punto de programas junto con directrices de metodología de programación que faciliten al lector la tarea de diseñar y construir programas con calidad y eficiencia junto con una introducción a la ingeniería de software.

**Capítulo 18. Resolución de problemas y desarrollo de software: Metodología de la programación.** En el capítulo se analiza el desarrollo de un programa y sus diferentes fases: análisis, diseño, codificación, depuración, pruebas y mantenimiento. Estos principios básicos configuran la ingeniería de software como ciencia que pretende la concepción, diseño y construcción de programas eficientes.

## APÉNDICES

En todos los libros dedicados a la enseñanza y aprendizaje de técnicas de programación es frecuente incluir apéndices de temas complementarios a los explicados en los capítulos anteriores. Estos apéndices sirven de guía y referencia de elementos importantes del lenguaje y de la programación de computadoras.

**Apéndice A. Especificaciones del lenguaje algorítmico UPSAM 2.0.** Se describen los elementos básicos del lenguaje algorítmico en su versión 2.0 junto con la sintaxis de todos los componentes de un programa. Asimismo se especifican las palabras reservadas y símbolos reservados. En relación con la versión 1.0 de UPSAM es de destacar una nueva guía de especificaciones de programación orientada a objetos, novedad en esta versión y que permitirá la traducción del pseudocódigo a los lenguajes orientados a objetos tales como C++, Java o C#.

**Apéndice B. Prioridad de operadores.** Tabla que contiene todos los operadores y el orden de prioridad y asociatividad en las operaciones cuando aparecen en expresiones.

**Apéndice C. Código ASCII y Unicode.** Tablas de los códigos de caracteres que se utilizan en programas de computadoras. El código ASCII es el más universal y empleado de modo masivo por programadores de todo el mundo y, naturalmente, es el que utilizan la mayoría de las computadoras actuales. Unicode es un lenguaje mucho más amplio que utilizan las computadoras personales para realizar programas y aplicaciones en cualquier tipo de computadora y en Internet. Unicode proporciona un número único para cada carácter, sin importar la plataforma, sin importar el programa, sin importar el idioma. La importancia de Unicode reside, entre otras cosas, en que está avalado por líderes de la industria tales como Apple, HP, IBM, Microsoft, Oracle, Sun, entre otros. También es un requisito para los estándares modernos, tales como XML, Java, C#, etc.

**Apéndice D. Guía de sintaxis del lenguaje C.** En este apéndice se hace una descripción de la sintaxis, gramática y especificaciones más importantes del lenguaje C.

**Bibliografía y recursos de programación: Libros, Revistas, Web, Compiladores.** Enumeración de los libros más sobresalientes empleados por el autor en la escritura de esta obra, así como otras obras importantes de referencia que ayuden al lector que desee profundizar o ampliar aquellos conceptos que considere necesario conocer con más detenimiento. Listado de sitios Web de interés para la formación en Java, tanto profesionales como medios de comunicación, especialmente revistas especializadas.

## APÉNDICES EN EL SITIO WEB ([www.mhe.es/joyanes](http://www.mhe.es/joyanes))

**Apéndice I. Guía de sintaxis de Pascal y Turbo Pascal.** Aunque ya está muy en desuso en la enseñanza de la programación el lenguaje Pascal, su sintaxis y estructura sigue siendo un modelo excelente de aprendizaje y es ésta la razón de seguir incluyendo esta guía de sintaxis en la obra.

**Apéndice II. Guía de sintaxis del lenguaje ANSI C.** Especificaciones, normas de uso y reglas de sintaxis del lenguaje de programación C en su versión estándar ANSI/ISO.

**Apéndice III. Guía de sintaxis del lenguaje ANSI/ISO C++ estándar.** Especificaciones, normas de uso y reglas de sintaxis del lenguaje de programación C++ en su versión estándar ANSI/ISO.

**Apéndice IV. *Guía de sintaxis del lenguaje Java 2.*** Descripción detallada de los elementos fundamentales del estándar Java: especificaciones, reglas de uso y de sintaxis.

**Apéndice V. *Guía de sintaxis del lenguaje C#.*** Descripción detallada de los elementos fundamentales del lenguaje C#: especificaciones, reglas de uso y de sintaxis.

**Apéndice VI. *Palabras reservadas: C++, Java y C#.*** Listados de palabras reservadas (clave) de los lenguajes de programación C++, Java y C#. Se incluye también una tabla comparativa de las palabras reservadas de los tres lenguajes de programación.

**Apéndice VII. *Glosario de palabras reservadas de C/C++.*** Glosario terminológico de las palabras reservadas del lenguaje de programación C/C++ con una breve descripción y algunos ejemplos de uso de cada palabra.

**Apéndice VIII. *Glosario de palabras reservadas de C#.*** Glosario terminológico de las palabras reservadas del lenguaje de programación C# con una breve descripción y algunos ejemplos de uso de cada palabra.

## AGRADECIMIENTOS

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

Esta cuarta edición ha sido leída y revisada con toda minuciosidad y —fundamentalmente— rigurosidad por los siguientes profesores de la Universidad Pontificia de Salamanca en el campus de Madrid: **Matilde Fernández Azuela, Lucas Sánchez García e Ignacio Zahonero Martínez** (mi eterno agradecimiento).

Como ya comenté anteriormente, muchos otros profesores españoles y latinoamericanos me han ayudado en la concepción y realización de esta obra y de otras muchas, de una u otra manera, apoyándome con su continua colaboración y sugerencia de ideas para la puesta en marcha de asignaturas del área de programación, en temas tan variados como *Fundamentos de Programación, Lenguajes de programación tales como BASIC, Visual Basic, Pascal, C, C++, Java o Delphi*. Citar a todos ellos me llevaría páginas completas. Sí, al menos, y como reconocimiento silencioso, decir que además de España, se incluyen todos los países latinoamericanos desde México, Perú, Venezuela o Colombia a Argentina, Uruguay y Chile en el cono sur, pasando por Guatemala o República Dominicana en Centroamérica. En cualquier forma, sí quería destacar de modo especial a la profesora M.<sup>a</sup> Eugenia Valesany de la Universidad Nacional del Nordeste de Corrientes en Argentina, por la labor de rigurosa revisión que realizó sobre la segunda edición y la gran cantidad de sugerencias y propuestas que me ha hecho para esta nueva edición motivada fundamentalmente por su extraordinaria y valiosa investigación al mundo de los algoritmos y de la programación. **A todos ellos y a todos nuestros lectores y alumnos de España y Latinoamérica, una vez más, nuestro agradecimiento eterno.**

Además de a nuestros compañeros en la docencia y a nuestros alumnos, no puedo dejar de agradecer, una vez más, a mi editor —y sin embargo amigo— José Luis García Jurado, que inició y puso en marcha todo el proyecto de esta 4.<sup>a</sup> edición, y también a mi nueva editora, Cristina Sánchez, que ha terminado dicho proyecto, las constantes muestras de afecto y comprensión que han tenido con mi obra. Esta ocasión, como no era menos, tampoco ha sido una excepción. Sin embargo, ahora he de resaltar esa gran amistad que nos une. La elaboración de esta obra por mil circunstancias ha entrañado, más que nunca, tal vez muchas más dificultades que otras obras nuestras. De nuevo y con gran paciencia, me han ayudado, comprendido y tolerado mis mil y una duda, sugerencias, retrasos, etc. No puedo por menos de expresar mi infinito reconocimiento y agradecimiento. Sin esta comprensión y su apoyo continuo posiblemente hoy todavía no habría visto la luz esta obra, debido a mis grandes retrasos en la entrega del original y posteriores revisiones de imprenta. Con el corazón en la mano, mi eterno agradecimiento. Pero en esta ocasión también deseo agradecer las muchas atenciones que mis editores de McGraw-Hill México dedican siempre a mis obras. Sus consejos, ideas y sugerencias siempre son un enorme aliciente y una ayuda inestimable. Sus consejos, ideas y sugerencias, unido a su gran paciencia y comprensión con el autor por sus muchos retrasos en la entrega de originales, han hecho que la obra haya sido mejorada considerablemente en el proceso de edición.

Naturalmente —y aunque ya los he citado anteriormente—, no puedo dejar de agradecer a nuestros numerosos alumnos, estudiantes y lectores, en general, españoles y latinoamericanos, que continuamente me aconsejan, critican y proporcionan ideas para mejoras continuas de mis obras. Sin todo lo que hemos aprendido, seguimos aprendiendo y seguiremos aprendiendo de ellos y sin su aliento continuo me sería prácticamente imposible terminar mis nuevas obras y, en especial, este libro. De modo muy especial deseo reiterar mi agradecimiento a tantos y tantos colegas de universidades españolas y latinoamericanas que apoyan nuestra labor docente y editorial. Mi más sincero reconocimiento y agradecimiento, una vez más, a todos: alumnos, lectores, colegas, profesores, maestros, monitores y editores. Muy bien sé que siempre estaré en deuda con vosotros. Mi único consuelo es que vuestro apoyo me sigue dando fuerza en esta labor académica y que, allá por donde mis derroteros profesionales me llevan, siempre está presente ese inmenso e impagable agradecimiento a esa enorme ayuda que me prestáis. **Gracias, una vez más, por vuestra ayuda.**

*En Carchelejo (Jaén) y en Madrid, otoño de 2007.*

***El autor***