"Programming and Problem Solving with Ada" de Nell Dale, Chip Weems y John McCormick, D.C. Heath and Company, 1994, 975 pp.

SpAda, Número 26, 1994.

Hace unos meses dábamos por concluido nuestro ciclo de comentarios sobre libros de texto de Programación y/o Ingeniería de Software *soportados* por Ada. Sin embargo, en 1994 han visto la luz dos nuevas obras que debemos añadir a las hasta entonces comentadas: la segunda edición de la obra de Skansholm "Programming from the beginning" (que incluye ahora material perteneciente a Ada 9X) y la que hoy traemos hasta estas páginas, "Programming and Problem Solving with Ada" de Nell Dale, Chip Weems y John McCormick. Se trata ésta de una obra concebida como un *premier* de programación y, en principio, pretende ser una actualización (más que una mera traducción a Ada) del texto de los propios Dale y Weems "Introduction to Pascal and Structured Design", texto que apareció en su primera edición en 1983 y que en su momento disfrutó de un considerable éxito. El libro se estructura en diecisiete capítulos, a los que se añaden nueve apéndices, un glosario y la colección de las respuestas a los ejercicios seleccionados en cada tema.

Como es ya casi norma, presentaremos el libro desde la doble perspectiva de su *método pedagógico* y de su *contenido técnico*. Empecemos entonces: *Pedagogical Features*. Así es como denominan los autores al conjunto de procedimientos docentes empleados a lo largo de todo el texto (recordemos, a título anecdótico si se quiere, que la obra de Feldman y Koffman —sin duda la más interesante en este aspecto— los denominaba *Pedagogical Tools*). Está compuesto por los siguientes:

- *Goals*: brevemente, en la portada de cada capítulo señalan qué se pretende aprender en él y dan una referencia rápida y completa de su contenido. Estas materias son repasadas al final del capítulo y su práctica es el objetivo de los ejercicios propuestos.

- *Special Sections*: engloban a cuatro:

  - *Theoretical Foundations*: describen de manera sencilla y, consecuentemente, accesible los principios teóricos en los que se fundamentan algunos conceptos empleados habitualmente en el mundo de los computadores. Destacaremos con entusiasmo el presente en el capítulo 5 y titulado *The Magnitude of Work* que constituye una magnífica introducción a la complejidad algorítmica.
  - *Software Engineering Tips*: no es lo que parece anunciar (recordemos de nuevo que se trata de un libro de introducción a la Programación, no a la Ingeniería de Software): recogen "consejos" (más que técnicas o métodos) para conseguir que los programas sean más fiables, robustos y eficientes. Por ejemplo cómo elegir una estructura de datos o cómo garantizar la *portabilidad* de los programas.
  - *Matters of Style*: cómo codificar los programas para conseguir su fácil legibilidad.
  - *Background Information*: "culturilla general informática", de lo mejor: por ejemplo, cómo se gestó el lenguaje Ada, o quién era ella (la que le dio el nombre) o Charles Babbage; o cuestiones al margen, como por ejemplo disquisiciones sobre la representación de los números en coma fija.

- *Problem-Solving Case Studies*: ya se sabe, con el gerundio se practica el infinitivo: programando, programar. Los autores hacen uso los "casos de estudio" para practicar los conceptos aparecidos en el correspondiente capítulo. Se plantea un problema (de lo más dispares y divertidos: una tienda de skís de fondo, *The Pasta House*, un sistema de control de vuelo simplificado, etc). En general, se aplica una técnica de *Top-Down* hasta obtener una solución escrita en lenguaje natural. Luego se aplica una técnica de diseño modular hasta que se llega a la codificación en Ada. Ya por último se muestran unos ejemplos de juegos de datos para comprobar el correcto funcionamiento del programa.

- *Testing and Debugging*: siguiendo a los casos de estudio presentes en cada capítulo, estas secciones consideran en profundidad las implicaciones del material presentado en el capítulo con repecto a la comprobación de los programas. La sección concluye con una lista de *testing and debugging hints* que resume una serie de, otra vez, consejos para conseguir la depuración de los programas. Los contenidos presentes en las secciones bajo este epígrafe son quizás excesivamente *ad hoc* para los problemas propuestos, aunque destacan por la sensación de veteranía que transmiten.

- *Quick Checks*: aparecen al final de cada capítulo y son un test rápido para el alumno sobre los puntos más importantes contenidos en aquél. Si el alumno falla en la respuesta, aparece el número de la página donde se encuentra el correspondiente concepto.
- *Exam Preparation Exercises*: eso: preguntas que ayudan al alumno a prepararse para los exámenes. Las respuestas correspondientes a unos cuantos seleccionados se encuentran a final del libro (las restantes, como comentaremos más adelante, aparecen en la *Instructor's Guide*).
- *Programming Warm-up Exercises*: permiten escribir trozos de código o subprogramas que ponen en práctica uno de los nuevos conceptos estudiados sin la necesidad de escribir un programa completo.
- *Programming Problems*: especificaciones de problemas que requieren escribir programas completos. Los problemas, como ocurría con los "casos de estudio", pertenecen a un rango de aplicaciones de lo más variopinto.

Aparte, la editorial suministra una serie de material destinado, fundamentalmente, al profesor: *Instructor's Guide*: contiene explicaciones adicionales a las contenidas en el texto que pueden ser útiles para la enseñanza de la materia, la respuesta a todos los ejercicios y, por cada capítulo, la resolución completa de uno de los problemas de programación, que se discute en profundidad; *Transparency Masters*: las consabidas transparencias para presentar los temas en clase; *Test Item File*: ¡más de 1.700 preguntas de test!; *Program Disk*: un disquete que contiene todos los programas presentes en el libro; *Ada Compiler*: dicen vender el compilador de Meridian al precio más bajo del mercado.

Yendo al otro aspecto, el contenido, en el libro aparece todo el Ada menos, primero, y como es habitual en este tipo de textos, la concurrencia y la interacción con el hardware, y, segundo, y eso sí que constituye una novedad, los punteros y las estructuras dinámicas de datos (conceptos que, alegan, raramente llegan a ser explicados en un primer curso de programación). Por lo demás, incluye materias que no son habituales en los textos para este nivel. Por ejemplo, sorprende, quizás, el tratamiento que se hace del tipo fichero (al que se le dedica la mitad del capítulo 9), o el capítulo 16 dedicado en exclusiva a los tipos numéricos (probablemente, de las obras hasta ahora comentadas, la que trata con mayor profundidad el tema). Dejando a un lado el primer capítulo, encargado de introducir conceptos básicos sobre los computadores y la programación (una muy buena introducción para quienes se acercan por primera vez al mundo de la informática), el resto de los temas van presentando todos los elementos del lenguaje en un orden que no está guiado por éste, sino por los diferentes aspectos de las técnicas de programación que se van mostrando de manera incremental. Así nos encontramos con el capítulo 5 de título *Looping* o con el 6, *Subprograms*. Precisamente éstos, los subprogramas, aparecían por primera vez en el capítulo 3, junto con una aproximación a la metodología *Top-Down*, mientras que no es hasta ese capítulo 6 cuando se enseña cómo escribirlos. A esas alturas ya se han descrito los tipos enumerados (aunque no exhaustivamente), pero no los subtipos, que se encuentran en el siguiente, capítulo 7. De la misma manera, los *arrays* son presentados en dos capítulos, el 12 dedicado a los de una dimensión y el 13 a los de dos o más. La recursividad es la materia que aparece en exclusiva en el capítulo 17.

Refiramos, por último, aparte, los contenidos del libro que tienen que ver con lo que conocemos como *Programming in the Large* (pero que, insistamos, no son tratados desde ese punto de vista: éste es sólo un libro de introducción a la programación). Los paquetes aparecen por primera vez en el capítulo 2, donde se hace uso de procedimientos de entrada/salida presentes en el paquete *Text_IO*, y se explican en detalle en el capítulo 11. La generación de ejemplares de paquetes genéricos aparece también tempranamente, capítulos 2 y 4, mientras que éstos se describen por completo en el 15. En lo que se refiere a las excepciones, se mencionan por primera vez en el capítulo 2, al tratar de los errores de compilación y de ejecución, se enseña cómo escribir "manejadores" en el capítulo 9, y, por último, la definición de excepciones por parte del programador se encuentra en el 15.

Otro excelente libro, alegría.