"Ada: Problem Solving and Program Design" de Michael B. Feldman y Elliot B. Koffman, Addison-Wesley, 1992, 744 pp.

SpAda, Número 16, 1992.

Aunque sea síntoma de falta de confianza en mis propios recursos docentes, tengo que reconocer que siempre me he sentido muy intrigado por saber cómo hacen los demás cuando quieren que alguien aprenda algo. Me temo, entonces, que el poder descubrir los "secretos" pedagógicos de Mike Feldman y Elliot Koffman, dos renombrados profesores universitarios, sea la razón por la que he disfrutado tanto estudiando el texto que hoy nos ocupa. Koffman, actualmente en la Universidad de Temple, fue uno de los fundadores del SIGCSE (*Special Interest Group in Computer Science Education*) de ACM y es autor de varios libros de éxito sobre diferentes lenguajes de programación. Por otra parte, es difícil que alguien que pertenezca al mundo Ada no haya oído hablar alguna vez de Mike Feldman (por lo que sé, "demasiado" conocido para los adictos a la lectura de las *news*). Es un activo miembro del *Education Commitee* del ACM SIGAda (y es fácil encontrar colaboraciones suyas en cualquier número de *Ada Letters* en el que se traten cuestiones relativas a la enseñanza de Ada), fue uno de los pioneros en el empleo de este lenguaje en la docencia universitaria, y es autor de dos libros sobre estructuras de datos en Ada y en Modula-2, respectivamente.

Cuando, todavía oliendo a tinta fresca, pude hojear "Ada: Problem Solving and Program Design" saqué dos conclusiones que a pesar de ser precipitadas son aún (quizás por eso de que las primeras impresiones son las que cuentan) lo que "a bote pronto" se me ocurre decir como presentación de esta obra: que es un libro con una clara (y original) orientación didáctica; y que, en contra de lo que su título parece indicar, se trata más de un libro *en* Ada, que de un libro *sobre* Ada. Los propios autores parecen confirmar ambas apreciaciones cuando anuncian que su obra está pensada para servir como "libro de texto" en un primer curso de programación, un semestre en el sistema educativo norteamericano, y que el lenguaje Ada aparece (sólo) como un buen medio para "poner en práctica" los conceptos estudiados.

Comenzando con el *qué* se cuenta, antes que detallar los conceptos contenidos en el libro, parece mucho más interesante comentar la soprendente distribución de éstos a través de los trece capítulos de la obra. Tras los dos capítulos iniciales, dedicados, el primero de ellos, a conceptos elementales sobre los computadores y la programación (dado el nivel para el que está concebido el texto, no se suponen conocimientos previos sobre estas materias), y, el segundo, a una introducción a Ada, aparecen, así de sopetón, dos temas que podemos considerar "avanzados" como son el diseño *top-down* y los paquetes. En el prefacio de la obra, los autores justifican estas y otras osadas decisiones, motivadas todas ellas por unos "principios programáticos" que marcan la orientación global del texto. Los dos conceptos mencionados son introducidos en este capítulo inicial con el propósito de que, desde un principio, se adquiera la costumbre de diseñar los programas pensando en la *reusabilidad*. Aparecen, así, abundantes ejemplos en los que se emplean paquetes que exportan funciones y procedimientos reusables. De hecho, no es hasta el capítulo 6 cuando se enseña cómo escribir los cuerpos de los procedimientos, retrasándose hasta ese momento la programación de procedimientos locales. Esta táctica tiene además beneficios laterales. Por ejemplo, hacer que los estudiantes no adquieran malas costumbres como la de referenciar variables globales desde los cuerpos de los procedimientos. Insistiendo en este plateamiento, es en el capítulo 9 cuando se entra de lleno en las técnicas para el desarrollo de los procedimientos que permiten la implementación de un diseño. Adicionalmente, aparecen en este tema algunas consideraciones sobre cómo validar la implementación realizada.

Otra de las pautas del planteamiento general de la obra es hacer especial hincapié en conceptos que están encuadrados dentro de los que se ha dado en llamar *Programming in the Large*. Entre otros objetivos, se pretende hacer consciente al alumno de la dimensión de los problemas en el mundo real y de su problemática adicional. Así, aparte de los elementos anteriormente citados, aparece, ya en el capítulo 6, la gestión de excepciones, sin duda, otro de los ingredientes claves en la construcción de sistemas de grandes dimensiones. Los capítulos 9 y 10 están dedicados expresamente al *Programming in the Large*. Recordemos que en el 9 aparecen técnicas para el desarrollo de programas, y, por su parte, el capítulo 10 presenta los tipos abstractos de datos. Abundando aún más en este tema, en el capítulo 12 se explica cómo construir unidades genéricas.

Los capítulos no mencionados hasta ahora comprenden conceptos que prodríamos considerar más clásicos: por ejemplo, el 5 las repeticiones en programas (bucles) y el 11 los *arrays* de varias dimensiones y a los registros con parte variante. El capítulo 13 y último está dedicado íntegramente a la recursión.

En lo referente al *cómo* se cuenta, recalcar que, bajo mi perspectiva, es lo más interesante de la obra. Ésta se estructura en capítulos (hasta ahí nada nuevo) y éstos a su vez en un conjunto de secciones encabezadas por epígrafes que se repiten a lo largo del texto:

Problem Solving: técnicas para el diseño de programas, construcción de algoritmos, etc.

Control Structures: construcciones fudamentales como sentencias condicionales, bucles, etc.

Data Structures: los diferentes *tipos* de datos.

System Structures: elementos para "programming in the large": paquetes, excepciones, etc.

Tricks of the Trade: técnicas informales para hacer las cosas mejor: "truquis".

Case Studies: desarrollo completo de programas a partir de la especificación de requerimientos y mediante la aplicación de una metodología de diseño.

Estas secciones aparecen con un título que especifica el caso particular que se trata. Vaya como ejemplo un fragmento del índice del capítulo 5 *Repetitions in Programs*:

Problem Solving: Repeating a Program Body

Case Study: Multiple-Employee Payroll Program

Control Structures: Nested Loops

Data Structures: Subtypes of Scalar Types

Tricks of the Trade: Debugging and Testing Programs

System Structures: An Useful Functions Package and the Overloading Principle

Distribuidos por las distintas secciones, aparecen lo que los autores denominan *teaching devices*:

Complete compilable programs: programas que corresponden a la solución de algún *Case Study*.

Syntax Displays: breve descripción de una nueva estructura Ada y un comentario sobre su significado.

Programming Style Displays: un poco de coquetería.

End-of-Section Exercises, End-of-Chapter Exercises: preguntas y pequeños ejercicios de programación.

Error Discussions and Chapter Review: aparecen al final de cada capítulo y comentan los errores más corrientes que se cometen en la programación de los nuevos conceptos estudiados, a la vez que se repasan éstos y se hace un resumen de las nuevas estructuras de Ada que han aparecido en el capítulo.

Destacar además que, aunque no aparezcan explícitamente como elementos pedagógicos, todo el texto está abundantemente "regado" con pequeños trozos de código que son ejemplo de la estructura Ada en cuestión. También hay que mencionar el loable esfuerzo de los autores por emplear desde un principio programas "listos para compilar", lo que, entre otras virtudes, representa un buen estímulo para el alumno (los fuentes de estos programas se distribuyen en un disquete que se adjunta con el libro). Y por último, sería injusto no señalar la originalidad y "sabrosura" de los *Tricks of Trade* y de los *Programming Style Displays*, algo así como "consejos de colega". No se los pierda.