INTRODUCCIÓN AL LENGUAJE ADA

Ada es un lenguaje de programación que sigue los principios de la programación estructurada. Este lenguaje fue diseñado por Jean Ichbiah de CII Honeywell Bull para el Departamento de Defensa de Estados Unidos, en el año 1979, tras realizar un concurso público en los años 70 con el fin de adquirir un lenguaje dotado de ciertas características inexistentes en más de una veintena de lenguajes estudiados. Se puede afirmar que el diseño del lenguaje Ada tiene sus orígenes en Pascal, Algol y PL/1, con importantes extensiones tanto sintácticas como semánticas. El nombre del lenguaje se eligió en honor a Ada Augusta Byron, hija del misántropo, pero genial poeta, Lord George Byron. Ada Augusta Byron es considerada como la primera programadora, por sus trabajos con Charles Babbage, creador de la máquina analítica.

Ada pronto se convirtió en un estándar de ANSI en 1983 (ANSI/MIL-STD 1815A-1983) y de ISO en 1987 (ISO-8652:1987). Ada 83 presentaba nuevos conceptos frente a otros lenguajes como Pascal como el manejo de excepciones y la programación genérica. En esta primera versión, la encapsulación y la posibilidad de introducir parámetros genéricos constituían un grupo de características propias ya de un lenguaje OO.

Entre 1992 y 1995, Tucker Taft de Intermetrics diseña Ada 95 (ISO/IEC 8652:1995), una revisión con la que el lenguaje ya se puede considerar OO. La nueva versión incorporaba conceptos propios del paradigma orientado a objetos, tales como herencia, polimorfismo y ligadura dinámica. Ada fue el primer lenguaje de programación OO estandarizado por ISO.

Se espera que durante el año 2006, ISO publique la siguiente revisión del lenguaje, Ada 2005 (ISO/IEC 8652:2005), que no recoge cambios tan profundos como lo hacía Ada 95 respecto a Ada 83. A título informativo, entre las modificaciones que presenta Ada 2005 podemos citar: cambios en el modelo OO, cambios en los tipos puntero, inicializaciones por defecto, sintaxis especial para elevar excepciones con mensaje, ampliación de la biblioteca predefinida, nuevo tipo de caracteres, mejoras en tiempo real y concurrencia, etc. Para mayor información, el lector puede consultar la página http://es.wikibooks.org/wiki/.

Hasta 1997, el Departamento de Defensa de Estados Unidos exigía el uso de este lenguaje en los proyectos que contrataba. Algo parecido ocurría en otros ministerios de países de la OTAN. Actualmente, Ada está presente fundamentalmente en entornos académicos universitarios y se aplica para desarrollar software que precisa seguridad y fiabilidad, como la defensa y la aeronáutica. Por ejemplo, la empresa española Indra utiliza Ada en el software de gestión del tráfico aéreo, y en el ordenador principal de los aviones Tornado británicos también se emplea Ada.

A continuación, se presentan las principales características de Ada, con una breve referencia a su tratamiento en otros lenguajes como Pascal, Modula y C.

- Ada tiene una sintaxis inspirada en Pascal. Por ejemplo, las palabras clave de una instrucción condicional son *if* y *else*, el operador de asignación es := y el de igualdad =. Su sintaxis está orientada a obtener programas legibles, pensada para leerlos no para escribirlos como ocurre en el críptico lenguaje C.
- Al igual que Pascal, y a diferencia de C, se puede utilizar indistintamente mayúsculas y minúsculas para los identificadores y las palabras claves del lenguaje, esto es, es un lenguaje *case-insensitive*.
- Como ocurre en Modula, cada instrucción estructurada se cierra con la palabra clave *end*, para facilitar la lectura y evitar errores debidos a interpretaciones equivocadas en la ejecución de los bloques de instrucciones.
- Al contrario de lo que sucede en lenguajes como C, hay una clara distinción entre procedimiento y función. Esta distinción no es solo sintáctica, pues Ada impide que los parámetros de las funciones sean de entrada/salida (*in out* en Ada, *var* en Pascal), para evitar los conocidos efectos colaterales; Pascal carece de esta característica. Una función Ada puede devolver cualquier tipo, predefinido o definido por el programador, estructurado o escalar, a diferencia de lenguajes como Pascal que no permiten que una función devuelva un resultado no escalar.
- Ada es un lenguaje estructurado en bloques como Pascal y Modula. Además, estos lenguajes tienen en común la posibilidad de definir procedimientos y funciones como léxico local de otros procedimientos y funciones. Las reglas de ámbito y visibilidad de Ada son similares a las reglas de ámbito y visibilidad de Pascal.
- Una característica peculiar de Ada frente a lenguajes como Pascal y Modula es la sobrecarga en procedimientos y funciones. En Ada es posible definir dos procedimientos o funciones con el mismo nombre, pero con parámetros de diferente tipo. Además, también permite sobrecargar los operadores en notación infija para que, además de operar sobre los tipos predefinidos, con el significado habitual, actúen sobre tipos definidos por el programador. Entre éstos se encuentran los operadores aritméticos habituales '+', '*', '-', '/' y los operadores relacionales '=', '<=', '>', etcétera.
- En Ada se pueden definir módulos que presentan una interfaz pública y una implementación privada. Un lenguaje como Pascal adolece de ocultación de la información al definir unidades y aunque lenguajes como Modula sí lo permiten, el uso del mecanismo de punteros para dar cuenta de esta característica de diseño es algo artificial y oscurece la legibilidad del programa.
- En la línea de lenguajes como C, Pascal y Modula, Ada es un lenguaje tipado estáticamente. Sin embargo, al contrario que el lenguaje C, es fuertemente tipado para que el compilador favorezca la captura de mayor número errores, y lo hagan más adecuado para un curso de

iniciación a la programación. Ada también permite la comprobación de errores en tiempo de ejecución que pueden ser habilitados o deshabilitados en aras al rendimiento.

- Al igual que en C, el signo de puntuación ';' se utilizado como terminador de instrucciones y no como separador como ocurre en Pascal. Por tanto, cualquier instrucción precisa acabar en ';', aunque lleve después un final de bloque o sentencia.
- Como lenguaje pensado entre otras cosas para la interacción con dispositivos reales, Ada incluye en su estándar una serie de tipos y operaciones predefinidas que permiten el control adecuado de la ejecución simultánea (concurrente o paralela) de distintas tareas como parte de un mismo programa. En este estándar se incluyen operaciones propias de sistemas concurrentes, tales como la sincronización y comunicación entre distintas tareas y el seguimiento del tiempo real. En aquellos sistemas operativos que ofrecen la posibilidad de controlar con mayor detalle este tipo de características, los compiladores de Ada pueden implementar el anejo Sistemas de Tiempo Real (comentado más abajo) que establece un estándar para realizar un control más fino de este tipo de características.
- A diferencia de Pascal y Modula, Ada permite la entrada/salida de tipos enumerados (incluido el tipo booleano, que en Ada es un caso particular, aunque algo especial, de tipo enumerado).
- Ada ofrece un repertorio de operaciones estándar, agrupadas en paquetes genéricos, para gestionar la entrada y salida en archivos de memoria externa. Estas operaciones permiten tanto el acceso secuencial como directo.
- Además de éstas y otras características del lenguaje, el estándar ISO/IEC 8652:1995 de Ada'95 define una serie de anejos (llamados *Specialized-needs anexes*) que opcionalmente pueden implementar los compiladores del lenguaje que se adapten a esta norma estándar. A continuación se describen brevemente estos anejos, aunque como es evidente no serán objeto de estudio en este libro:
    - *Programación de Sistemas*: Define un estándar para las operaciones de acceso a la máquina a bajo nivel, tales como control sobre el código máquina o manejo de interrupciones.
    - *Sistemas de Tiempo Real*: Además de las características propias del núcleo del lenguaje para el manejo de tareas, define extensiones adicionales orientadas a la construcción de software para sistemas de tiempo real, tales como el control de las prioridades de las tareas, la política de asignación de tiempos a tareas concurrentes, etcétera. Estas características precisan que el sistema operativo ofrezca al programador la posibilidad de controlar estos aspectos de la programación. Un compilador que implemente las extensiones de este anejo debe también implementar las de *Programación de Sistemas*.

- *Sistemas Distribuidos*: Establece extensiones del lenguaje orientadas al diseño de programas que se ejecutan (o potencialmente pudieran hacerlo) de forma paralela en más de un procesador, posiblemente en equipos diferentes o incluso remotos.
- *Sistemas de Información*: Define extensiones del lenguaje útiles para la programación de aplicaciones de gestión, sistemas financieros, etcétera, consistentes, sobre todo, en la definición de tipos numéricos apropiados para representar de forma adecuada códigos de referencia, cantidades de dinero (con o sin céntimos de unidad), etcétera, así como las operaciones para el manejo y edición de datos de estos tipos.
- *Cálculo Numérico*: Este anejo establece un estándar de tipos y operaciones para la construcción de programas orientados al cálculo numérico intensivo. Entre estas extensiones se incluye el tipo *número complejo*, junto con un extenso repertorio de operaciones asociadas al mismo; instrucciones y directivas de compilador que posibilitan al programador un mayor control sobre ciertas características de las operaciones matemáticas en punto flotante o fijo, tales como la precisión, la secuencia de generación de valores pseudoaleatorios, etcétera.
- *Seguridad y Fiabilidad*: Incluye extensiones del lenguaje y directivas de compilador orientadas a la construcción de programas cuyo funcionamiento es especialmente crítico y precisan una alta fiabilidad, revisión del código generado, demostración de la corrección de sus características, restringir al programador el uso de determinadas posibilidades del lenguaje que podrían oscurecer los puntos anteriores, etcétera.

Para un libro de introducción a la programación, basta con mostrar un reducido número de conceptos básicos del lenguaje. El núcleo básico de Ada es similar al de otros lenguajes estructurados tradicionales como Pascal o Modula. De hecho, Ada comparte con estos lenguajes la mayor parte de las instrucciones estructuradas y conceptos básicos, además de presentar una sintaxis similar, por lo que el lector que tenga conocimientos en Pascal o Modula, no encontrará dificultades en el aprendizaje de este nuevo lenguaje.