Historia y evolución de los lenguajes de programación

Vicente Trigo Aranda

El conjunto de órdenes e instrucciones que se dan al ordenador para que resuelva un problema o ejecute una determinada misión, recibe el nombre de programa. En los primeros tiempos de la informática, la programación se efectuaba en el único lenguaje que entiende el microprocesador: su propio código binario, también denominado lenguaje máquina o código máquina.

Pero la programación en lenguaje máquina resulta muy lenta y tediosa, pues los datos e instrucciones se deben introducir en sistema binario y, además, obliga a conocer las posiciones de memoria donde se almacenan los datos. Como puede imaginar, este tipo de programación conlleva gran número de errores y la tarea de depuración exige bastante tiempo y dedicación.

Por este motivo, a principios de los 50 se creó una notación simbólica, denominada código de ensamblaje (ASSEMBLY), que utiliza una serie de abreviaturas mnemotécnicas para representar las operaciones (figura 1): ADD (sumar), STORE (copiar), etc.. Al principio, la traducción del código de ensamblaje al código máquina se realizaba manualmente, pero enseguida se vio que el ordenador también podía encargarse de esa traducción; se desarrolló así un programa traductor, llamado ensamblador[1] (ASSEMBLER).

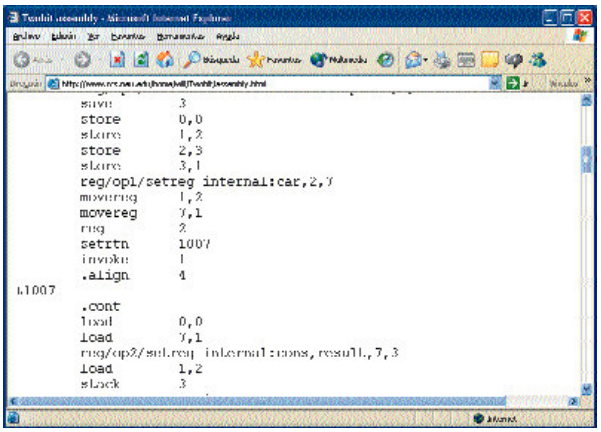


Figura 1. Ejemplo de programa en código de ensamblaje

Conforme los ordenadores fueron introduciéndose en el mundo empresarial y académico, aquellos primitivos lenguajes fueron sustituidos por otros más sencillos de aprender y más cómodos de emplear. Estos lenguajes, llamados de alto nivel, tienen una estructura que se adapta más al pensamiento humano que a la forma de trabajar del ordenador. Por ejemplo, seguro que le suenan lenguajes como BASIC, PASCAL, C, etc.

[1] En la actualidad, se acostumbra identificar el ensamblador, que es el programa traductor, con el código de ensamblaje.

¿Y cuántos lenguajes de programación existen? Pues sucede algo así como con los lenguajes humanos: existen centenares, si bien sólo unos pocos de ellos son ampliamente utilizados. En este artículo me voy a centrar en los lenguajes de programación más difundidos, siguiendo un criterio cronológico, y lo finalizaré presentándole una somera relación de otros lenguajes de programación más minoritarios o que ya han caído en desuso.

INTÉRPRETES Y COMPILADORES

Antes de pasar a ver los principales lenguajes de programación, debemos hacer un alto para comentar una característica común a todos ellos: las órdenes dadas en cualquier lenguaje siempre deben traducirse al código binario del ordenador, que es el único que realmente comprende su unidad central. Esta labor de traducción se lleva a cabo mediante un intérprete o un compilador. ¿Y qué diferencia hay entre estas dos cosas? Retomemos el símil de los lenguajes humanos y todo quedará aclarado.

Imagine que no sabe nada de inglés y necesita conversar con alguien que sólo conoce ese idioma. La forma más sencilla de establecer comunicación es conseguir una persona que ejerza de intérprete. Cuando diga una frase en castellano, su intérprete la traducirá al inglés y, de esta forma, podrá entenderla aquella persona con la que esté conversando; análogo proceso se seguirá para traducir del inglés al castellano. En resumen, mientras esté presente su intérprete, la conversación es posible.

El intérprete informático realiza, más o menos, el mismo papel. Traduce instrucción a instrucción y, de esta forma, favorece la interactividad, la depuración y puesta a punto del programa, la ejecución inmediata de una orden, etc. Por ejemplo, entre los lenguajes que suelen ser interpretados, se encuentran BASIC, LOGO, etc.

No obstante, existe otro tipo de traducción, la escrita, que presenta diferencias con respecto a la traducción oral. Suponga, por ejemplo, que uno de sus libros vaya a ser editado en Gran Bretaña. Desde luego, no tiene sentido que, quien compre su libro, tenga un intérprete cerca cada vez que desee leer su libro. Resulta más lógico traducir el libro al inglés, ¿no? Cuando el traductor haya finalizado su trabajo, se tendrán dos copias del libro; el original, en castellano, y su traducción, en inglés, que ya puede ser leída por cualquier persona que conozca el idioma de Shakespeare, sin necesidad de intérprete a su lado.

El equivalente informático de esta modalidad de traductor se denomina compilador[2]. Observe que, en contraste con el intérprete, que traduce las instrucciones una a una, el compilador traduce todo el programa de golpe, dejándolo listo para ser ejecutado[3]. De esta forma, se logra mayor rapidez en la ejecución y, además, se liberan recursos de la memoria, pues el programa, una vez compilado, no exige que el traductor esté residente en memoria, como sucede con los intérpretes. Por ejemplo, entre los lenguajes que siempre son compilados se pueden destacar PASCAL, FORTRAN, COBOL, etc.

Sin embargo, no todo son ventajas en los lenguajes compilados. Así, la depuración del programa resulta más cómoda con un intérprete, ya que el compilador no informa de los posibles errores hasta el momento de la compilación. Por otra parte, cada vez que se modifica algo en el programa es preciso volver a compilarlo de nuevo[4].

Una vez aclarada la diferencia entre intérprete y compilador, vayamos ya con el primer lenguaje de alto nivel de amplia difusión.

FORTRAN

Al comienzo de la década de los 50, John Backus estaba trabajando con SSEC (*Selective Sequence Electronic Calculator*), uno de los primeros ordenadores de IBM, y desarrolló el programa SPEEDCODING para él. Tomando éste como base, se emprendió, en otoño de 1954, la creación de un lenguaje para añadirle más prestaciones al modelo IBM 704, que iba a salir pronto al mercado.

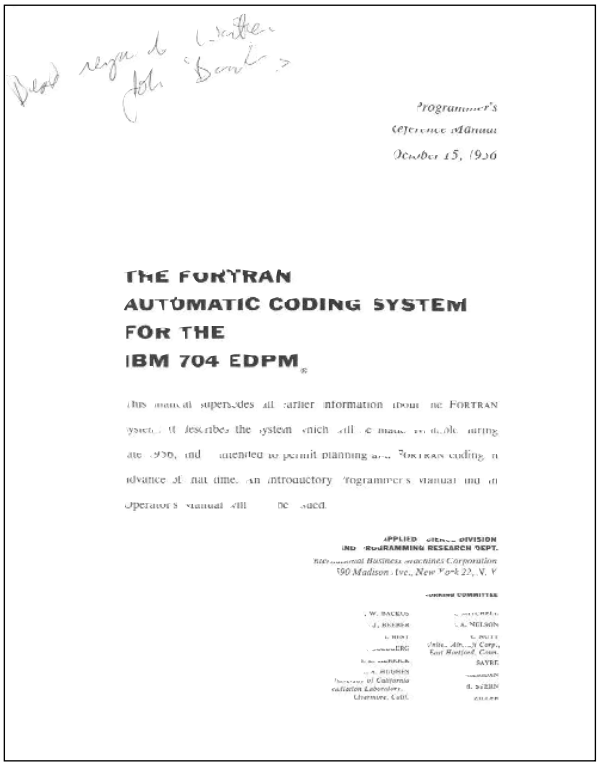
---

[2] La introducción del término compilador (*compiler*, en inglés) en informática, se debe a la pionera Grace Hopper, que designó con ese nombre a su programa traductor A-O (1949). Más tarde, desarrolló el compilador B-O para UNIVAC.

[3] Se llama programa fuente al programa original y programa objeto al programa resultante de la traducción.

[4] Debido a que los intérpretes ocupan menos memoria que los compiladores y, además, siempre deben estar en memoria, en los microordenadores (¿recuerda el ZX-Spectrum o el VIC-20?) se incluía, a modo de sistema operativo, un lenguaje interpretado (BASIC, generalmente).

En 1956 se terminó el compilador FORTRAN (*FORmula TRANslator*) y se incluyó en el IBM 704, junto con un manual de 51 páginas (figura 2).

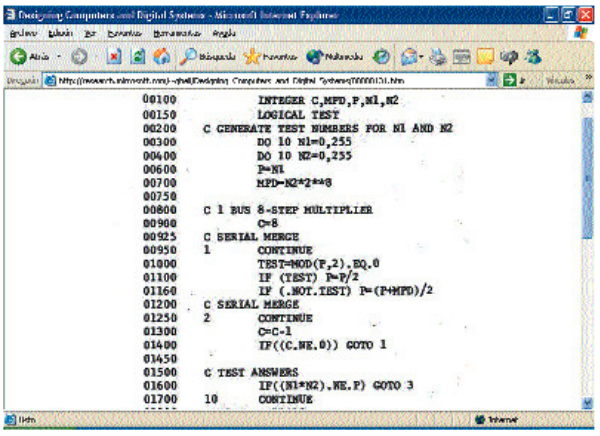


Programmer's
Reference Manual
October 15, 1956

THE FORTRAN
AUTOMATIC CODING SYSTEM
FOR THE
IBM 704 EDPM

Figura 2. Primera página del primer manual de FORTRAN, con la firma de Backus

Como su nombre indica, FORTRAN estaba (y está) destinado a la resolución de problemas científico-técnicos, resultando relativamente sencillo de aprender si se domina la notación matemática.



```
00100           INTEGER C,MPD,P,N1,N2
00150           LOGICAL TEST
00200   C GENERATE TEST NUMBERS FOR N1 AND N2
00300           DO 10 N1=0,255
00400           DO 10 N2=0,255
00600           P=N1
00700           MPD=N2*2**8
00750
00800   C 1 BUS 8-STEP MULTIPLIER
00900           C=8
00925   C SERIAL MERGE
00950   1       CONTINUE
01000           TEST=MOD(P,2).EQ.0
01100           IF (TEST) P=P/2
01160           IF (.NOT.TEST) P=(P+MPD)/2
01200   C SERIAL MERGE
01250   2       CONTINUE
01300           C=C-1
01400           IF((C.NE.0)) GOTO 1
01450
01500   C TEST ANSWERS
01600           IF((N1*N2).NE.P) GOTO 3
01700   10      CONTINUE
```

Figura 3. Programa en FORTRAN

Aunque ha ido perfeccionándose a lo largo del tiempo (con sus versiones II, IV, 77 y 90), lo cierto es que se ha visto superado por otros muchos lenguajes, ya que sus programas carecen de estructuración y son difíciles de seguir (figura 3). Sin embargo, todavía se sigue utilizando, sobre todo en el ámbito universitario. ¿Por qué? La respuesta radica en la gran biblioteca de subrutinas y funciones que se ha ido creando en sus más de treinta años de existencia.

COBOL

A finales de los 50, el Departamento de Defensa USA estaba bastante preocupado con los lenguajes de programación existentes, especialmente por dos razones: los programas no podían llevarse de un ordenador a otro y resultaban bastante difíciles de leer y modificar. Para solventar estos inconvenientes, patrocinó una conferencia sobre lenguajes (CODASYL, *COnference on DAta SYstems Languages*), que tuvo lugar en 1959 y en la que participaron las grandes empresas del sector (IBM, Sperry Rand, Honey Well, etc.). Como era previsible, formaba parte del comité la gran dama de la informática, Grace Hopper (figura 4).

Figura 4. Hopper junto al monumento a COBOL

Fruto de aquella conferencia fueron las especificaciones para desarrollar COBOL (*COmmon Business Oriented Language*), un lenguaje orientado hacia funciones administrativas, de gran portabilidad y legibilidad. Su primera versión apareció al año siguiente y, con el paso de los años, surgieron nuevas actualizaciones: COBOL 74, COBOL 85, etc.

Ya que se buscaba su facilidad de lectura, COBOL tiene una sintaxis muy similar al inglés común (figura 5), cuya terminología aparece continuamente: verbos, párrafos, frases, etc. Así, los programas se estructuran en cuatro divisiones (*Identification*, *Enviroment*, *Data*, *Procedure*), que se subdividen en secciones y éstas, a su vez, en párrafos, que constan de frases e instrucciones.

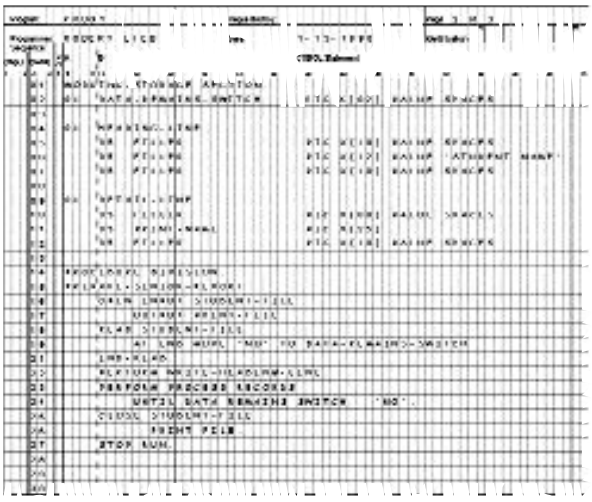


Figura 5. Antigua hoja de codificación en COBOL

En la actualidad, COBOL se utiliza casi exclusivamente en algunos grandes sistemas informáticos (entidades bancarias, sobre todo), si bien más para mantener el código existente que para desarrollar nuevas aplicaciones.

BASIC

John G. Kemeny y Thomas E. Kurtz eran profesores del Dartmouth College (New Hampshire) y, en 1964, diseñaron un nuevo lenguaje que permitiera introducirse a sus estudiantes en los sistemas de tiempo compartido. Ese lenguaje, al que llamaron BASIC por su sencillez[5], es, sin duda, el más difundido, aplicándose tanto en tareas de gestión como en aplicaciones científicas (figura 6).

Figura 6. Programa BASIC que simula un sorteo de la Primitiva

¿Y a qué se debe la gran popularidad del BASIC? Lo cierto es que no era el mejor lenguaje ni el más potente, pero tenía dos ventajas a su favor: se trataba de un lenguaje sencillo de aprender y, además, su intérprete ocupaba poca memoria. Por estos motivos, cuando se creó el primer ordenador personal (Altair de MITS), no es extraño que se desarrollase un BASIC para él. ¿Y sabe qué empresa lo diseñó? ¡Exacto! Microsoft.

Más tarde, Microsoft adaptó su BASIC a los productos de Apple, a los microordenadores y, lo más importante, al PC de IBM; de hecho, el sistema operativo MS-DOS incluía la versión GW-BASIC. En resumen, mucha gente aprendió a programar en BASIC con su ZX-Spectrum o su primer PC y, una vez dominado un lenguaje, es comprensible una cierta reticencia al cambio.

Además de GW-BASIC, hubo otras versiones que tuvieron cierta difusión en los 80, como Turbo BASIC (de Borland) y QuickBASIC (de Microsoft). De hecho, hasta Kemeny y Kurtz intentaron aprovechar el éxito de su creación y, en 1983, crearon True BASIC (figura 7), cuya comercialización no resultó muy fructífera[6].

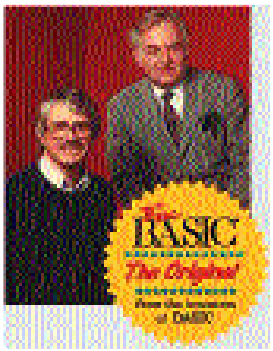


Figura 7. True BASIC, casi desconocido

BASIC ha sabido adaptarse a las necesidades del mercado en el transcurso de los años. Así, las primeras versiones eran interpretadas y sus programas resultaban un tanto ilegibles; en cambio, las actuales incorporan bastante estructuración y son compiladas. El exponente máximo de los modernos BASIC es Visual BASIC... también de Microsoft, claro está.

[5] BASIC son las siglas de *Beginners All Purpose Symbolic Instruction Code* (código de instrucciones simbólicas multiuso para principiantes), aunque, en mi opinión, se trata de uno más de los juegos de palabras tan comunes en informática. Lo más probable es que primero surgiera el nombre y, luego, se buscase la forma de encajarlo como siglas de algo, porque lo cierto es que el larguísimo nombre oficial se las trae.

[6] True BASIC todavía sigue vigente. Si lo desea, puede visitar su Web:

http://www.truebasic.com/

LOGO

En 1964, Seymour Papert se incorporó al MIT, tras haber permanecido cinco años en Suiza, colaborando con el pedagogo Jean Piaget (1896-1980). Tres años después, Papert comenzó a diseñar un lenguaje que sirviera para introducir en el mundo de la programación al alumnado de menor edad (figura 8)… "*¡Que los niños programen a los ordenadores y no los ordenadores a los niños!*"

Figura 8. Papert con unos niños programadores (1984)

Poco a poco, LOGO fue poniéndose a punto y cuando, en 1980, Papert lo divulgó en todo el mundo con su libro "*Mindstorms: Children Computers and Powerful Ideas*" (figura 9), fue muy bien acogido en los ámbitos educativos, especialmente en enseñanza primaria y secundaria.

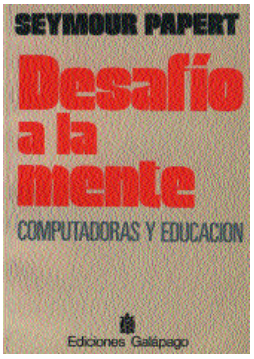


Figura 9. Versión en castellano de "Mindstorms"

Teniendo en cuenta los pocos conocimientos matemáticos de sus potenciales usuarios, LOGO introduce al mundo de la programación de una forma gráfica, mediante la geometría de la tortuga. En los primeros tiempos, con LOGO se controlaba un pequeño robot con ruedas, motor y un lápiz retráctil, que admitía órdenes sencillas e intuitivas (Avanza, Retrocede, Gira a la derecha, etc.) y al desplazarse por el papel iba trazando el dibujo ordenado. Como el robot tenía forma abombada (figura 10) se le llamó tortuga[7] (también influyó en el nombre el hecho de que era bastante lento).

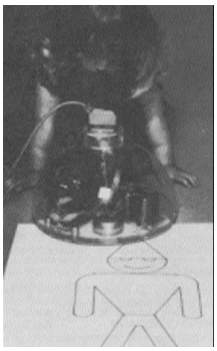


Figura 10. Una de las tortugas de la primera época

Sin embargo, tras un impulso inicial muy ilusionante, LOGO ha ido desapareciendo de los centros de enseñanza españoles. ¿Las causas? Por un lado, resulta que el lenguaje LOGO no es nada sencillo cuando se quiere ir más allá de la tortuga gráfica, ya que se basa en la utilización continua de listas y procedimientos recursivos, que no son fáciles de manejar. Por otra parte, la informática educativa ha ido perdiendo su componente formativa y creativa (la programación) y ha sido sustituida por una informática de usuario (manejo de aplicaciones ofimáticas).

C

En los Laboratorios Bell (New Jersey) trabajaron dos de los investigadores más conocidos de la moderna informática, Kenneth Thompson y Dennis Ritchie (figura 11), creadores del sistema operativo UNIX, en 1969.

Figura 11. Thompson y Ritchie recibiendo la Medalla National Technology en 1999

[7] En los ordenadores actuales, la tortuga se representa en pantalla mediante un pequeño triángulo.

En 1970, Thompson desarrolló un lenguaje experimental, al que llamó B. Dos años después, Ritchie se basó en B para crear un nuevo lenguaje de propósito general, que denominó C[8]. Como no depende de la arquitectura del hardware, C es uno de los lenguajes más portables del mercado y, como además ofrece amplias prestaciones, su difusión es amplísima.

A principios de los 80, Bjarne Stroustrup (figura 12) diseñó una ampliación de C y, en 1984, la convirtió en un compilador que llamó C++[9], especialmente enfocado a la programación orientada a objetos.

Figura 12. La dirección es http://www.research.att.com/~bs/homepage.html

PASCAL

A principios de los 70, el profesor suizo Niklaus Wirth (figura 13), del Instituto Politécnico Federal de Zurich, emprendió la creación de un nuevo lenguaje (PASCAL) que permitiera introducirse en la programación de una forma fácil pero a la vez potente y, sobre todo, siguiendo unas pautas estructuradas. De hecho, PASCAL es el lenguaje más sencillo que posibilita el acceso a la informática teórica: descomposición modular, recursividad, punteros, etc.

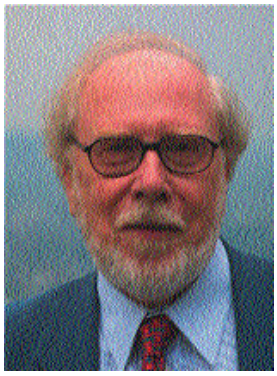


Figura 13. Niklaus Wirth

PASCAL, que surgió como una derivación de ALGOL (ver último apartado), fue definido en el libro "*PASCAL - User Manual and Report*" (1974), escrito por Kathleen Jensen y Niklaus Wirth (figura 14). En 1980 sufrió la primera formalización y se estandarizó en 1983[10]. Al poco tiempo, Borland lanzó al mercado su compilador PASCAL, cuyo nombre se precedía con la palabra Turbo, para recalcar su rapidez. Su éxito fue tan grande[11] que vendió casi medio millón de copias de su compilador sólo en 1985.

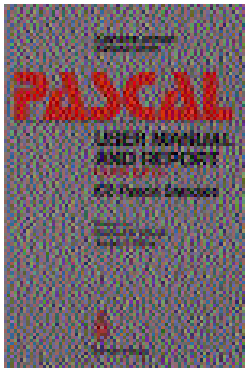


Figura 14. Portada del libro que divulgó el PASCAL

Durante más de un década, Turbo PASCAL ha sido sinónimo de PASCAL, pero, por desgracia, en el año 2000, Borland dejó de darle soporte técnico y su presencia es cada día menor en el ámbito de la programación, sobreviviendo a duras penas en el mundo univer-

[8] Posteriormente, se reescribió UNIX en C. También están desarrollados en C el propio compilador C y la mayoría de sistemas operativos.

[9] En C el operador ++ equivale a incrementar la variable. Como el nuevo lenguaje incrementaba la potencialidad de C con la programación orientada a objetos, se optó por el nombre C++.

[10] Poco después, para solventar todos los inconvenientes que presentaba PASCAL, Wirth diseñó el lenguaje MODULA-2 (*MODUlar LAnguaje number 2*), que está basado en el manejo de módulos, como bloques independientes.

[11] Microsoft, que había lanzando su propia versión de PASCAL (Quick PASCAL), acabó retirándola del mercado.

sitario. Sin embargo, en 1995 surgió una nueva versión, DELPHI, que amplía PASCAL a la programación visual e intenta hacerle la competencia a Visual BASIC.

PROLOG

En 1972, Robert Kowalski (universidad de Edimburgo) y Alain Colmerauer y Phillippe Roussell (universidad de Aix-Marseille) expusieron la revolucionaria idea de que la lógica podía emplearse como lenguaje de programación. Siguiendo esta línea, al año siguiente, el grupo de inteligencia artificial de la universidad de Aix-Marseille comenzó a diseñar ese lenguaje, al que se llamó PROLOG (*PROgramation LOGique*).

Figura 15. Alain Colmerauer

PROLOG es el prototipo de lenguaje declarativo por excelencia. ¿Y qué es eso de declarativo? Todos los lenguajes que hemos ido viendo hasta ahora son algorítmicos; es decir, los ordenadores se consideran máquinas de Turing a las que debemos indicar todos y cada uno de los pasos a seguir para realizar una cierta tarea. Frente a estos lenguajes imperativos, los declarativos no están basados en órdenes sino en descripciones. En otras palabras, en los programas PROLOG se proporcionan al ordenador una serie de conocimientos sobre un tema, junto con una serie de reglas, y el programa nos contestará todas aquellas preguntas que deseemos hacerle sobre el tema… siempre que las respuestas puedan deducirse lógicamente de los conocimientos dados al inicio.

Como es fácil suponer, PROLOG no está destinado al cálculo científico. Su aplicación en el campo de la inteligencia artificial, definiendo objetos y estableciendo relaciones, permite resolver problemas lógicos, desarrollar sistemas expertos, investigar en la comprensión del lenguaje humano, etc.

ADA

Quince años después de intentar uniformizar los lenguajes con COBOL, el Departamento de Defensa USA percibió que su objetivo no se había cumplido, ni mucho menos[12]. Por ese motivo, en 1975 formó un grupo de trabajo para evaluar los lenguajes existentes en aquel entonces[13] y ver si alguno de ellos podía adaptarse a las necesidades del Departamento.

¿Y qué condiciones debía cumplir el lenguaje deseado? En principio, debía permitir el diseño de programas modulares y estructurados, de modo que fuesen fáciles de leer y de depurar. También era necesario que, como debía controlar instrumentos militares de todo tipo, gestionase sin problemas cualquier periférico. Además, tenía que aceptar el trabajo en paralelo, de modo que varios procesos se ejecutaran de forma simultánea o cuasisimultánea.

¿Qué lenguaje eligió el grupo de trabajo? Lo cierto es que ninguno le convenció y en su informe final propuso la creación de un nuevo lenguaje, recomendando que se basara en PASCAL, PL/I (ver último apartado) y ALGOL 68, ya que eran los más apropiados de los evaluados[14].

Poco después, se convocó un concurso para desarrollar un nuevo lenguaje que se ajustara a los requerimientos del Departamento de Defensa. Se presentaron 17 propuestas que, tras una selección previa, quedaron

[12] Se estima que, a comienzos de los ochenta, sus ordenadores manejaban del orden de cuatrocientos lenguajes distintos, si bien muchos de ellos eran dialectos creados específicamente para determinados equipos militares. Teniendo en cuenta, además, que los programas tenían un promedio de cien mil líneas de código, no es extraño que hubiera un cierto caos informático.

[13] Los lenguajes evaluados fueron: ALGOL 60, ALGOL 68, CMS-2, COBOL, CORAL 66, CS-4, ECL, EUCLID, FORTRAN, HAL/S, JOVIAL J-3B, JOVIAL J-73, LIS, LTR, MORAL, RTL/2, PASCAL, PDL/2, PEARL, PL/I, SIMULA 67, SPL/1, TACPOL.

[14] *The unanimous recommendations of the evaluation committee, adopted unanimously by the HOLWG were:*

- *That none of the evaluated languages met the requirements to such an extent as to be selected with little or no modifications for a DoD-wide standard;*
- *That it appeared feasible within the state-of-the-art to construct a single language to meet essentially all of the requirements;*
- *That the construction of such a language would most likely be done by modification (albeit substantial) of an existing language. The approaches recommended as a basis for further development were the language families of PL/I, Pascal, and ALGOL-68.*

reducidas a cuatro, a las que se asignó los nombres clave de *Red*, *Green*, *Yellow* y *Blue*, por aquello de preservar el anonimato.

Finalmente, *Green* fue el lenguaje elegido. Propuesto por Honeywell-Bull (Francia), fue diseñado por un equipo encabezado por Jean Ichbiah (figura 16)[15]. En un primer momento, se le dio el nombre de DoD-1 pero acabó cambiándose por ADA, en honor de Ada Lovelace.

Figura 16. Jean Ichbiah

Sin embargo, a pesar de los años transcurridos desde entonces y a las mejoras que han ido introduciéndose en él (figura 17), lo cierto es que ADA no es un lenguaje popular, salvo por su nombre. Se le reprocha ser un tanto complejo, bastante estricto y sólo apropiado para el desarrollo de grandes programas.

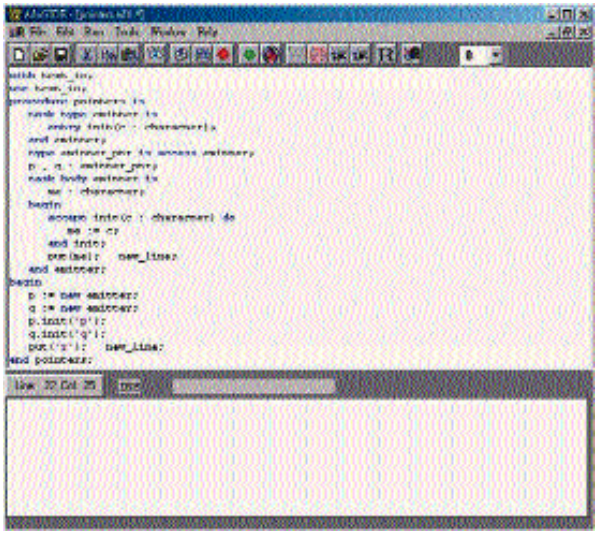


Figura 17. Programa realizado con el compilador GNAT ADA 95

JAVA

Este lenguaje[16], hoy en día ampliamente utilizado en Internet, fue desarrollado en 1990 por James Gosling (figura 18), de Sun Microsystems, basándose en C y C++. ¿Un lenguaje para Internet cuando, en aquella época, la Red estaba casi circunscrita al ámbito universitario? En realidad, el objetivo de Sun no tenía nada que ver con Internet; era crear un interfaz atractivo e intuitivo para electrónica de consumo (calculadoras, televisión interactiva, etc.).

Figura 18. James Gosling

Sin embargo, la electrónica de consumo no evolucionó como se esperaba y, durante unos años, el lenguaje de Gosling permaneció aparcado, hasta que Bill Joy (cofundador de Sun) consideró que podía ser interesante para Internet y propuso modificarlo para el nuevo medio. En agosto de 1995, ya con el nombre de JAVA, se presentó en sociedad[17].

A pesar de que JAVA resulta un tanto lento en su ejecución, cada día es más popular. Por un lado, es relativamente sencillo y bastante potente; además, es válido para cualquier plataforma y, sobre todo, muy fiable y seguro, manteniendo alejado a los virus.

ALGUNOS OTROS LENGUAJES

Como le indicaba al comienzo del artículo, además de los lenguajes de programación vistos anteriormente,

[15] Los otros tres proyectos correspondían a Intermetrics [Benjamin M. Brosgol] (*Red*), SofTech [John Goodenough] (*Blue*) y SRI International [Jay Spitzen] (*Yellow*). El nombre que aparece entre corchetes es el del director de cada equipo.

[16] Su nombre era inicialmente OAK (roble, en inglés), debido a la abundancia de ese árbol en los alrededores de las oficinas donde se estaba desarrollando.

[17] El nombre OAK estaba registrado por otra empresa y durante una reunión del equipo, en una cafetería, se propuso llamar a su lenguaje JAVA... el nombre de una marca de café.

se han ido desarrollando otros muchos, si bien su ámbito de aplicación es más reducido o ya han dejado de utilizarse. A modo de ejemplo, le muestro una somera relación de algunos que han tenido cierto interés y/o influencia.

PLANKALKÜL

Podríamos decir que es el antepasado de los modernos lenguajes de programación. Fue creado por Konrad Zuse, a mediados de los 40, para su serie de máquinas Z. Su nombre es una combinación de las palabras *Plan* y *Kalkül*, así que podría traducirse por "plan de cálculo".

SHORT CODE

Basándose en las ideas de John W. Mauchly, William F. Schmitt creó este lenguaje interpretado en 1950 y fue utilizado en la primera serie de UNIVAC. Es considerado el precursor de los lenguajes de alto nivel.

FLOWMATIC

El primer lenguaje de programación destinado al tratamiento de aplicaciones de gestión. Desarrollado por el equipo de Grace Hopper en 1957, este lenguaje compilado sólo fue implementado en UNIVAC.

LISP

Durante un encuentro sobre inteligencia artificial celebrado en el verano de 1956, H. A. Simon, A. Newell y J. C. Shaw describieron su lenguaje IPL (*Information Processing Language*), creado para el ordenador JOHNIAC. Inspirándose en ese lenguaje, en 1958 John McCarthy creó el LISP (*LISt Processing language*) como parte de un proyecto de inteligencia artificial del MIT, teniendo como soporte un equipo IBM 704[18]. Se trata de un lenguaje conciso e interactivo, basado en el tratamiento de listas (de ahí su nombre), ya que tanto los programas como los datos se estructuran mediante listas.

ALGOL

Con vistas a obtener un lenguaje universal, que no dependiera de la máquina donde se implementara, se formó un comité internacional, formado por la ACM (*Association for Computing Machinery*) y la GAMM (siglas alemanas de la Sociedad para las Matemáticas aplicadas), que, en 1958, publicó en Zurich un informe dando carta de nacimiento al IAL (*International Algebraic Language*), posteriormente denominado ALGOL 58 (*ALGOritmic Language*). Su versión operativa se presentó en Paris en 1960 y, más adelante, fue perfeccionada (ALGOL 68).

Aunque ha caído en desuso, su influencia ha sido decisiva en el desarrollo de los lenguajes de programación posteriores, ya que muchos de los más importantes (PASCAL, C, ADA, JAVA, etc.) descienden, directa o indirectamente, de ALGOL.

JOVIAL

Su nombre son las siglas de "*Jules' Own Version of the International Algorithmic Language*" y fue desarrollado en 1959, partiendo de IAL (de ahí el nombre[19]), para Air Force USA, que deseaba un lenguaje válido tanto para usos científicos como de gestión... y JOVIAL todavía sigue en activo (figura 19).

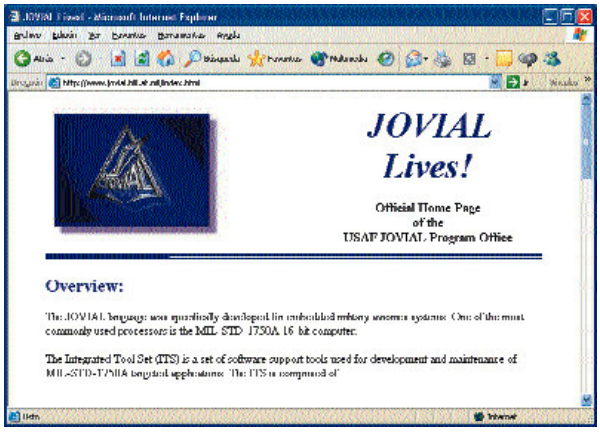


Figura 19. La dirección es: http://www.jovial.hill.af.mil

[18] Para conocer más a fondo la historia de LISP, según John McCarthy, vaya a la siguiente dirección:

http://www-formal.stanford.edu/jmc/history/lisp/lisp.html

[19] El nombre original del lenguaje era OVIAL (*Our Own Version of the International Algebraic Language*) pero se le añadió la J inicial porque, según palabras de Jules Schwartz, director del proyecto: "*In the late 1950s, society wasn't quite as free thinking as it is today. The name OVIAL seemed to have a connotation relative to the birth process that did not seem acceptable to some people*".

APL

El profesor Kenneth E. Iverson ideó una notación para describir, sin ambigüedad y con concisión, algoritmos matemáticos y la dio a conocer en su libro "*A Programming Language*" (cuyas siglas corresponden al nombre del lenguaje), publicado en 1962. Partiendo de esa notación, IBM desarrolló el lenguaje APL, orientado a usos científicos. Todavía se sigue utilizando y sus programas se reconocen visualmente por su brevedad y la inclusión de caracteres especiales (figura 20).

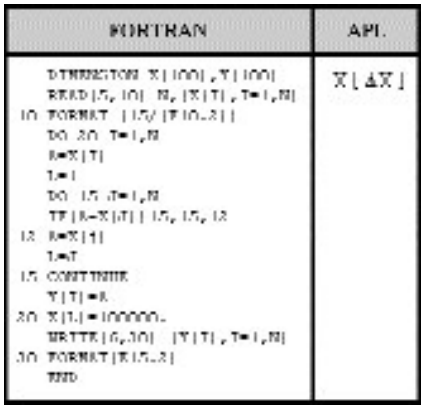


Figura 20. Ordenación de una lista en FORTRAN y APL. La concisión de éste último es innegable

PL/I

Este lenguaje fue desarrollado por IBM, a partir de 1963, que deseaba un lenguaje polivalente, en el sentido de que podía aplicarse tanto a gestión como al ámbito científico. Buscando aunar las ventajas de COBOL, FORTRAN y ALGOL, PL/I[20] resultó un lenguaje muy flexible y potente, por lo que todavía sigue en uso.

RPG

A principios de los 60, IBM comenzó a desarrollar un lenguaje orientado la obtención de informes (ventas, pagos, etc.) en el ámbito de gestión (RPG son las siglas de *Report Program Generator*). En 1964 salió al mercado con la serie IBM 360 y, desde entonces, ha sufrido diversas actualizaciones: II, III, 400, IV, Visual RPG. Es un lenguaje sencillo de aprender, si bien su versatilidad no es mucha. En las cuatro secciones en que se estructura cada programa, se deben indicar los archivos y dispositivos a emplear, fijar las especificaciones de entrada, determinar las operaciones a realizar y establecer los formatos de salida. A partir de la versión IV se añadió la sección de subprocedimientos.

SIMULA

Basado en ALGOL, se trata del primer lenguaje orientado a objetos. Fue desarrollado por los noruegos Ole-Johan Dahl y Kristen Nygaard (figura 21) que buscaban un lenguaje adecuado para la simulación de eventos discretos (su nombre es una contracción de *Simulation Languaje*). Su primer compilador estuvo disponible en 1964, para la serie 1100 de UNIVAC, si bien hasta 1967 no adquirió una amplia funcionalidad. En la actualidad hay disponibles diversas versiones freeware de su compilador[21].

Figura 21. Nygaard y Dahl, ya fallecidos, recibieron el premio Turing de la ACM en el año 2001

FORTH

Este lenguaje fue creado a finales a los 60 por Charles H. Moore, para controlar los radiotelescopios de Kitt Peak y procesar sus datos. Se trata de un lenguaje[22] funcional e interactivo que ha ido evolucionan-

[20] PL/I son las siglas de *Programming Language/I*, donde *I* representa el uno… ¿El primer lenguaje de programación? ¿El número uno de los lenguajes de programación?

[21] Por ejemplo, en:

http://www.simula-c.de/download.htm

http://www.ifi.uio.no/~cim/sim_index.shtml

[22] El origen de su nombre es ciertamente curioso. Moore consideraba que su lenguaje era tan avanzado que debía pertenecer a la cuarta (*fourth*, en inglés) generación de ordenadores (en aquel entonces estaban en la tercera). Pero como su implementación tuvo lugar en un IBM 1130 que sólo admitía identificadores de cinco letras, perdió la U y se quedó en FORTH.

do con el paso del tiempo. Debido a la poca memoria que ocupa y a su rapidez, fue uno de los primeros en difundirse entre los microordenadores (figura 22).

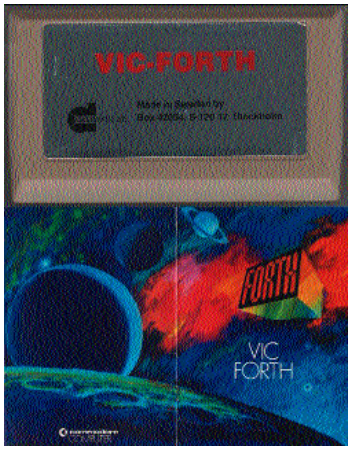


Figura 22. FORTH para Commodore

LSE

Con objeto de que el profesorado francés de secundaria fuese capaz de crear sus propios materiales educativos informáticos, se diseñó el lenguaje este lenguaje (*Lenguage Symbolique d'Enseignement*) en 1971, que no tuvo mucho éxito[23].

SMALLTALK

Creado por Alan Kay en el Centro de Investigaciones Xerox de Palo Alto, en los primeros 70, es un lenguaje muy influenciado por SIMULA, estando también orientado a objetos. Tuvo sucesivas versiones (72, 76 y 80) y ofrece un entorno completo para el desarrollo de programas.

COMAL

Destinado a la informática educativa en los países escandinavos, fue desarrollado por Benedict Loefstedt y Borge Christensen en 1973, combinando las ventajas de BASIC y PASCAL (sus siglas corresponden a *COMmon Algorithmic Language*). En 1980 se estandarizó y todavía sigue siendo utilizado, sobre todo en la Europa del norte.

¿Se ha quedado con ganas de conocer más lenguajes? Si la respuesta es afirmativa, puede darse un paseo por la Web *Dictionary of Programming Languages* (figura 23), donde encontrará referencias y comentarios sobre un centenar y medio de lenguajes.

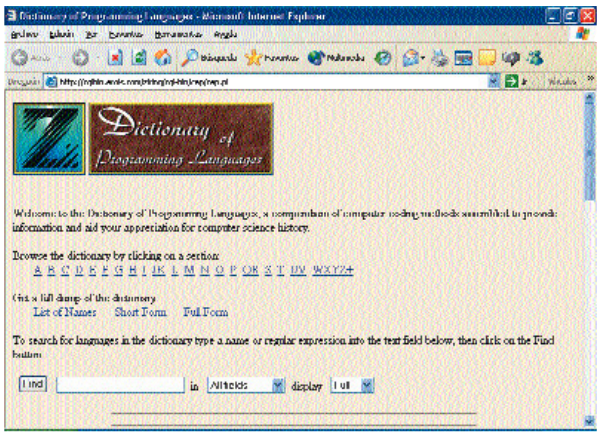


Figura 23. Su dirección es http://cgibin.erols.com/ziring/cgi-bin/cep/cep.pl

[23] En la Escuela Superior de Electricidad se desarrollaron, entre 1968 y 1976, varios lenguajes con una filosofía similar: LSD, LSE, LSG y LST.