

# Índice

Lenguajes de programación .....	3
1. Un poco de historia.....	8
2. ¿Qué son los lenguajes de programación?.....	9
3. Tipos de lenguaje de alto nivel:.....	9
-Listas:.....	11
Historia.....	15
Críticas:.....	17
Ejemplo de programa en Algol.....	18
Algol W.....	19
Algol 68.....	19
Características.....	21
Empleo.....	22

# LENGUAJES DE ALTO NIVEL

---

## NOMBRES:

CRISTINA DIAZ- MAROTO

ESTEFANIA HUEDO NOGUEROL

NATALIA DE MOYA ROMERO

# LENGUAJES DE PROGRAMACIÓN

El estudio de los lenguajes de programación agrupa tres intereses diferentes; el del programador profesional, el del diseñador del lenguaje y del Implementador del lenguaje.

Además, estos tres trabajos han de realizarse dentro de las ligaduras y capacidades de la organización de una computadora y de las limitaciones fundamentales de la propia "calculabilidad". El termino "el programador" es un tanto amorfo, en el sentido de que camufla importantes diferencias entre distintos niveles y aplicaciones de la programación. Claramente el programador que ha realizado un curso de doce semanas en COBOL y luego entra en el campo del procesamiento de datos es diferente del programador que escribe un compilador en Pascal, o del programador que diseña un experimento de inteligencia artificial en LISP, o del programador que combina sus rutinas de FORTRAN para resolver un problema de ingeniería complejo, o del programador que desarrolla un sistema operativo multiprocesador en ADA.

En esta investigación, intentaremos clarificar estas distinciones tratando diferentes lenguajes de programación en el contexto de cada área de aplicación diferente. El "diseñador del lenguaje" es también un termino algo nebuloso. Algunos lenguajes (como APL y LISP) fueron diseñados por una sola persona con un concepto único, mientras que otros (FORTRAN y COBOL) son el producto de desarrollo de varios años realizados por comités de diseño de lenguajes.

El "Implementador del lenguaje" es la persona o grupo que desarrolla un compilador o interprete para un lenguaje sobre una maquina particular o tipos de maquinas. Mas frecuentemente, el primer compilador para el lenguaje Y sobre la maquina X es desarrollada por la corporación que manufactura la maquina X . Por ejemplo, hay varios compiladores de Fortran en uso; uno desarrollado por IBM para una maquina IBM, otro desarrollado por DEC para una maquina DEC, otro por CDC, y así sucesivamente. Las compañías de software también desarrollan compiladores y también lo hacen los grupos de investigación de las universidades. Por ejemplo, la universidad de Waterloo desarrolla compiladores para FORTRAN Y PASCAL, los cuales son útiles en un entorno de programación de estudiantes debido a su superior capacidad de diagnostico y velocidad de compilación.

Hay también muchos aspectos compartidos entre los programadores, diseñadores de un lenguaje implementadores del mismo. Cada uno debe comprender las necesidades y ligaduras que gobiernan las actividades de los otros dos.

Hay, al menos, dos formas fundamentales desde las que pueden verse o clasificarse los lenguajes de programación: por su nivel y por principales aplicaciones. Además, estas visiones están condicionadas por la visión histórica por la que ha transcurrido el lenguaje. Además, hay cuatro niveles distintos de lenguaje de programación.

Los "Lenguajes Declarativos" son los mas parecidos al castellano o ingles en su potencia expresiva y funcionalidad están en el nivel mas alto respecto a los otros. Son fundamentalmente lenguajes de ordenes, dominados por sentencias que expresan "Lo que hay que hacer" en ves de "Como hacerlo". Ejemplos de estos lenguajes son los lenguajes estadísticos como SAS y SPSS y los lenguajes de búsqueda en base de datos, como NATURAL e IMS. Estos lenguajes se desarrollaron con la idea de que los profesionales pudieran asimilar mas rápidamente el lenguaje y usarlo en su trabajo, sin necesidad de programadores o practicas de programación.

Los lenguajes de "Alto Nivel" son los mas utilizados como lenguaje de programación. Aunque no son fundamentalmente declarativos, estos lenguajes permiten que los algoritmos se expresen en un nivel y estilo de escritura fácilmente legible y comprensible por otros programadores. Además, los lenguajes de alto nivel tienen normalmente las características de "Transportabilidad". Es decir, están implementadas sobre varias maquinas de forma que un programa puede ser fácilmente "Transportado" (Transferido) de una maquina a otra sin una revisión sustancial. En ese sentido se llama "Independientes de la maquina". Ejemplos de estos lenguajes de alto nivel son PASCAL, APL y FORTRAN (para aplicaciones científicas), COBOL (para aplicaciones de procesamiento de datos), SNOBOL (para aplicaciones de procesamiento de textos), LISP y PROLOG (para aplicaciones de inteligencia artificial), C y ADA (para aplicaciones de programación de sistemas) y PL/I (para aplicaciones de propósitos generales).

Los "Lenguajes Ensambladores" y los "Lenguajes Maquina" son dependientes de la maquina. Cada tipo de maquina, tal como VAX de digital, tiene su propio lenguaje maquina distinto y su lenguaje ensamblador asociado. El lenguaje Ensamblador es simplemente una representación simbólica del lenguaje maquina asociado, lo

cual permite una programación menos tediosa que con el anterior. Sin embargo, es necesario un conocimiento de la arquitectura mecánica subyacente para realizar una programación efectiva en cualquiera de estos niveles lenguajes.

Los siguientes tres segmentos del programa equivalentes exponen las distinciones básicas entre lenguajes maquina, ensambladores de alto nivel:

Como muestra este ejemplo, a mas bajo nivel de lenguaje mas cerca esta de las características de un tipo e maquina particular y mas alejado de ser comprendido por un humano ordinario. Hay también una estrecha relación ( correspondencia 1:1 ) entre las sentencias en lenguaje ensamblador y sus formas en lenguaje maquina codificada. La principal diferencia aquí es que los lenguajes ensambladores se utilizan símbolos (X,Y,Z,A para " sumar", M para "multiplicar"), mientras que se requieren códigos numéricos (OC1A4, etc.) para que lo comprenda la maquina.

La programación de un lenguaje de alto nivel o en un lenguaje ensamblador requiere, por tanto, algún tipo de interfaz con el lenguaje maquina para que el programa pueda ejecutarse. Las tres interfaces mas comunes: un "ensamblador" , un "compilador" y un "interprete". El ensamblador y el compilador traduce el programa a otro equivalente en el lenguaje X de la maquina "residente" como un paso separado antes de la ejecución. Por otra parte, el interprete ejecuta directamente las instrucciones en un lenguaje Y de alto nivel, sin un paso de procesamiento previo.

La compilación es, en general, un proceso mas eficiente que la interpretación para la mayoría de los tipos de maquina. Esto se debe principalmente a que las sentencias dentro de un "bucle" deben ser reinterpretadas cada vez que se ejecutan por un interprete. Con un compilador. Cada sentencia es interpretada y luego traducida a lenguaje maquina solo una vez.

Algunos lenguajes son lenguajes principalmente interpretados, como APL, PROLOG y LISP. El resto de los lenguajes -- Pascal, FORTRAN, COBOL, PL/I, SNOBOL, C, Ada y Modula-2 – son normalmente lenguajes compilados. En algunos casos, un compilador estará utilizable alternativamente para un lenguaje interpretado (tal como LISP) e inversamente (tal como el interprete SNOBOL4 de los laboratorios Bell). Frecuentemente la interpretación es preferible a la compilación en un entorno de programación experimental o de educación, donde cada nueva ejecución de un programa implicado un cambio en el propio texto del programa. La calidad de diagnosis y depuración que soportan los lenguajes interpretados es generalmente mejor que la de los

lenguajes compilados, puesto que los mensajes de error se refieren directamente a sentencias del texto del programa original. Además, la ventaja de la eficiencia que se adjudica tradicionalmente a los lenguajes compilados frente a los interpretados puede pronto ser eliminado, debido a la evolución de las maquinas cuyos lenguajes son ellos mismos lenguajes de alto nivel. Como ejemplo de estos están las nuevas maquinas LISP, las cuales han sido diseñadas recientemente por Symbolics y Xerox Corporations.

Los lenguajes de Programación son tomados de diferentes perspectivas. Es importante para un programador decidir cuales conceptos emitir o cuales incluir en la programación. Con frecuencia el programador es osado a usar combinaciones de conceptos que hacen al lenguaje "DURO" de usar, de entender e implementar. Cada programador tiene en mente un estilo particular de programación, la decisión de incluir u omitir ciertos tipos de datos que pueden tener una significativa influencia en la forma en que el Lenguaje es usado, la decisión de usar u omitir conceptos de programación o modelos.

Existen cinco estilos de programación y son los siguientes:

Orientados a Objetos.

Imperativa : Entrada, procesamiento y salidas de Datos.

Funcional : "Funciones", los datos son funciones, los resultados pueden ser un valor o una función.

Lógico : {T,F} + operaciones lógicas (Inteligencia Artificial).

Concurrente : Aún esta en proceso de investigación.

El programador, diseñador e implementador de un lenguaje de programación deben comprender la evolución histórica de los lenguajes para poder apreciar por que presentan características diferentes. Por ejemplo, los lenguajes "mas jóvenes" desaconsejan (o prohíben) el uso de las sentencias GOTO como mecanismo de control inferior, y esto es correcto en el contexto de las filosofías actuales de ingeniería del software y programación estructurada. Pero hubo un tiempo en que la GOTO, combinada con la IF, era la única estructura de control disponible; el programador no dispone de algo como la construcción WHILE o un IF-THEN-ELSE para elegir. Por tanto, cuando se ve un lenguaje como FORTRAN, el cual tiene sus raíces en los comienzos de la historia de los lenguajes de programación, uno no debe sorprenderse de ver la antigua sentencia GOTO dentro de su repertorio.

Lo mas importante es que la historia nos permite ver la evolución de familias de lenguajes de programación, ver la influencia que ejercer las arquitecturas y aplicaciones de las computadoras sobre el

diseño de lenguajes y evitar futuros defectos de diseño aprendido las lecciones del pasado. Los que estudian se han elegido debido a su mayor influencia y amplio uso entre los programadores, así como por sus distintas características de diseño e implementación. Colectivamente cubren los aspectos más importantes con los que ha de enfrentarse el diseñador de lenguajes y la mayoría de las aplicaciones con las que se enfrenta el programador. Para los lectores que estén interesados en conocer con más detalle la historia de los lenguajes de programación recomendamos las actas de una recién conferencia (1981) sobre este tema, editadas por Richard Wexelblat. Vemos que FORTRAN I es un ascendente directo de FORTRAN II, mientras que FORTRAN, COBOL, ALGO 60, LISP, SNOBOL y los lenguajes ensambladores, influyeron en el diseño de PL/I.

También varios lenguajes están prefijados por las letras ANS. Esto significa que el American National Standards Institute ha adoptado esa versión del lenguaje como el estándar nacional. Una vez que un lenguaje está estandarizado, las maquinas que implementan este lenguaje deben cumplir todas las especificaciones estándares, reforzando así el máximo de transportabilidad de programas de una máquina a otra. La policía federal de no comprar máquinas que no cumplan la versión estándar de cualquier lenguaje que soporte tiende a "fortalecer" el proceso de estandarización, puesto que el gobierno es, con mucho, el mayor comprador de computadoras de la nación.

Finalmente, la notación algebraica ordinaria, por ejemplo, influyó fuertemente en el diseño de FORTRAN y ALGOL. Por otra parte, el inglés influyó en el desarrollo del COBOL. El lambda cálculo de Church dio los fundamentos de la notación funcional de LISP, mientras que el algoritmo de Markov motivó el estilo de reconocimiento de formas de SNOBOL. La arquitectura de computadoras de Von Neumann, la cual fue una evolución de la máquina más antigua de Turing, es el modelo básico de la mayoría de los diseños de computadoras de las últimas tres décadas. Esta máquina no solo influyeron en los primeros lenguajes sino que también suministraron el esqueleto operacional sobre el que evolucionó la mayoría de la programación de sistemas.

Una discusión más directa de todos estos primeros modelos no están entre los objetivos de este texto. Sin embargo, es importante apuntar aquí debido a su fundamental influencia en la evolución de los primeros lenguajes de programación, por una parte, y por su estado en el núcleo de la teoría de la computadora, por otra. Mas sobre este punto, cualquier algoritmo que pueda describirse en

inglés o castellano puede escribirse igualmente como una máquina de Turing (máquina de Von Neumann), un algoritmo de Markov o una función recursiva. Esta sección, conocida ampliamente como "tesis de Church", nos permite escribir algoritmos en distintos estilos de programación (lenguajes) sin sacrificar ninguna medida de generalidad, o potencia de programación, en la transición.

## 1. UN POCO DE HISTORIA

El lenguaje de programación Pascal es un *lenguaje de alto nivel* y propósito general (aplicable a una gran cantidad de aplicaciones diversas) desarrollado por el profesor suizo **Niklaus Wirth** (Instituto tecnológico de Zurich, Suiza). El propósito de Wirth era crear un lenguaje para la enseñanza de técnicas de programación a estudiantes universitarios. Pero a medida que pasaban los años, Pascal se iba convirtiendo en un estándar en el mundo de la programación.

Una versión preliminar del lenguaje apareció en **1968** y el primer compilador totalmente completo apareció a finales de **1970**. Desde entonces, muchos compiladores han sido construidos y están disponibles para diferentes máquinas. Durante muchos años, el libro *Pascal User Manual and Report*, publicado por Wirth y Kathleen Jensen en **1974**, ha servido *de facto* como estándar de todas las versiones.

Las diferentes versiones ofrecían interpretaciones ligeramente diferentes que impedían la compatibilidad entre ellas. Por estas razones, diferentes proyectos se iniciaron para producir una definición estándar del lenguaje y culminaron en dos estándares: uno de la *International Standard Organization (ISO)* en 1982 y otro por un comité conjunto del *American National Standards Institute (ANSI)* y del *Institute of Electrical and Electronics Engineers (IEEE)*. Estas dos versiones o definiciones se conocen como **ISO Pascal** y **ANSI/IEEE Pascal**, y difieren en algunos aspectos no especialmente significativos. Sin embargo, una versión no estándar se ha popularizado considerablemente: **Turbo Pascal** (marca registrada por *Borland International, Inc.*). Esta versión ha contribuido en gran medida a la popularización del lenguaje Pascal.

## **2. ¿QUÉ SON LOS LENGUAJES DE PROGRAMACIÓN?**

Los lenguajes de alto nivel logran la independencia del tipo de máquina y se aproximan al lenguaje natural. Se puede decir que el principal problema que presentan los lenguajes de alto nivel es la gran cantidad de ellos que existen actualmente en uso.

Los lenguajes de alto nivel, también denominados lenguajes evolucionados, surgen con posterioridad a los anteriores, con los siguientes objetivos, entre otros:

- Lograr independencia de la máquina, pudiendo utilizar un mismo programa en diferentes equipos con la única condición de disponer de un programa traductor o compilador, que lo suministra el fabricante, para obtener el programa ejecutable en lenguaje binario de la máquina que se trate. Además, no se necesita conocer el hardware específico de dicha máquina.
- Aproximarse al lenguaje natural, para que el programa se pueda escribir y leer de una forma más sencilla, eliminando muchas de las posibilidades de cometer errores que se daban en el lenguaje máquina, ya que se utilizan palabras (en inglés) en lugar de cadenas de símbolos sin ningún significado aparente.
- Incluir rutinas de uso frecuente como son las de entrada/salida, funciones matemáticas, manejo de tablas, etc, que figuran en una especie de librería del lenguaje, de tal manera que se pueden utilizar siempre que se quieran sin necesidad de programarlas cada vez.

Se puede decir que el principal problema que presentan los lenguajes de alto nivel es la gran cantidad de ellos que existen actualmente en uso (FORTRAN, LISP, ALGOL, COBOL, APL, SNOBOL, PROLOG, MODULA2, ALGOL68, PASCAL, SIMULA67, ADA, C++, LIS, EUCLID, BASIC), además de las diferentes versiones o dialectos que se han desarrollado de algunos de ellos.

## **3. TIPOS DE LENGUAJE DE ALTO NIVEL:**

### **A) PASCAL:**

El lenguaje de programación en Pascal, es un lenguaje de alto nivel, y de propósito general, lo cual quiere decir que se puede utilizar para cualquier tipo de propósitos. El lenguaje de programación en Pascal se considera un lenguaje estructurado, sencillo y práctico para todos aquellos usuarios que se inician en el mundo de la programación, ya que fue creado con fines de aprendizaje.

Al ser un Pascal lenguaje estructurado, sirve de base para cualquier otro lenguaje de alto nivel, por estas características es utilizado en las universidades e institutos de educación para inicializar a los futuros ingenieros en sistemas o informática.

El lenguaje de programación Pascal, es idóneo en el estudio y definición de las estructuras de datos, su fácil definición lo hace manejable para un programador novato.

Con la programación en Pascal, se pueden realizar desde programas formales, rutinas, utilitarios, hasta cualquier clase de video juegos.

Programación en Pascal es un lenguaje de sintaxis sencilla, muy estructurado y que comprueba exhaustivamente todo tipo de datos.

El mejor de los propósitos de programación en Pascal es que enseña buenas formas de programación, con lo cual se utiliza mucho en la enseñanza, por todos los motivos nominados anteriormente, por su sencillez, su estructuración y su facilidad de lectura y entendimiento.

Existen varios dialectos locales de programación en Pascal, entre ellas el Turbo Pascal, el cual acepta instrucciones de Pascal.

## **B) FORTRAN**

Abreviatura de **F**órmula **T**ranslator (traductor de fórmulas), fue definido alrededor del año 1955 en los Estados Unidos por la

compañía IBM. Es el más antiguo de los lenguajes de alto nivel, pues antes de su aparición todos los programas se escribían en lenguaje ensamblador o en lenguaje máquina.

Es un lenguaje especializado en aplicaciones técnicas y científicas, caracterizándose por su potencia en los cálculos matemáticos, pero estando limitado en las aplicaciones de gestión, manejo de archivos, tratamiento de cadenas de caracteres y edición de informes.

A lo largo de su existencia han aparecido diferentes versiones, entre las que destaca la realizada en 1966 por ANSI (American National Standard Institute) en la que se definieron nuevas reglas del lenguaje y se efectuó la independencia del mismo con respecto a la máquina, es decir, comenzó la transportabilidad del lenguaje. Esta versión se denominó FORTRAN IV o FORTRAN 66. En 1977, apareció una nueva versión más evolucionada que se llamó FORTRAN V o FORTRAN 77, esta versión está reflejada en el documento «ANSI X3.9-1978: Programming Language FORTRAN» y define dos niveles del lenguaje denominados FORTRAN 77 completo y FORTRAN 77 básico, siendo el segundo un subconjunto del primero. Esta última versión incluye además instrucciones para el manejo de cadenas de caracteres y de archivos, así como otras para la utilización de técnicas de programación estructurada. Estas características hacen que el lenguaje también sea válido para determinadas aplicaciones de gestión.

### **C) LISP:**

**Lisp** es el segundo lenguaje de programación, después de Fortran, de alto nivel. Lisp es de tipo declarativo y fue creado por John McCarthy y sus colaboradores en el MIT.

#### **-Listas:**

El elemento fundamental en Lisp es la lista, en el sentido más amplio del término, pues tanto los datos como los programas son listas. De ahí viene su nombre, pues Lisp es un acrónimo de "LIStProcessing".

Los lenguajes de este tipo se llaman "aplicativos" o "funcionales", porque se basan en la aplicación de funciones a sus datos.

En Lisp se distinguen dos tipos fundamentales de elementos:

- Átomos: son datos elementales y pueden pertenecer a varios tipos: números, caracteres, cadenas de caracteres y símbolos.
- Listas: son secuencias de átomos o de listas encerradas entre paréntesis. Además, existe una lista especial, "nil", que es la lista nula, que no tiene ningún elemento.

En Lisp, una función se expresa como una lista.

Algunas de las funciones predefinidas de Lisp tienen símbolos familiares (+ para la suma, \* para el producto), pero otras son más exóticas, especialmente dos que sirven precisamente para manipular listas, descomponiéndolas en sus componentes. Sus nombres ("car" y "cdr") son un poco extraños(1), reliquias de tiempos pasados y de la estructura de los ordenadores de segunda generación, "car" devuelve la cabeza de una lista y "cdr" su cola o resto.

Lisp sigue una filosofía de tratamiento no-destructivo de los parámetros, de modo que la mayoría de las funciones devuelven una lista resultado de efectuar alguna transformación sobre la que recibieron, pero sin alterar esta última.

Uno de los motivos por los que Lisp es especialmente adecuado para la IA es el hecho de que el código y los datos tengan el mismo tratamiento (como listas); esto hace especialmente sencillo escribir programas capaces de escribir otros programas según las circunstancias.

Lisp fue uno de los primeros lenguajes de programación a incluir manejo de excepciones con las primitivas *catch* y *throw*.

Derivado del Lisp es el lenguaje de programación Logo. Sin entrar en detalles, podría decirse que Logo es Lisp sin paréntesis y con operadores aritméticos infijos.

## **D) JAVA:**

### **El lenguaje de programación Java**

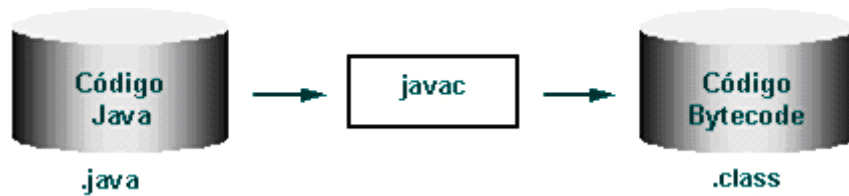
Podemos empezar diciendo que el lenguaje Java es de alto nivel (\*) y sus características más importantes son:

- Lenguaje orientado a objetos.
- Java es un lenguaje sencillo.
- Independiente de plataforma
- Brinda un gran nivel de seguridad
- Capacidad multihilo
- Gran rendimiento
- Creación de aplicaciones distribuidas
- Su robustez o lo integrado que tiene el protocolo TCP/IP lo que lo hace un lenguaje ideal para Internet.

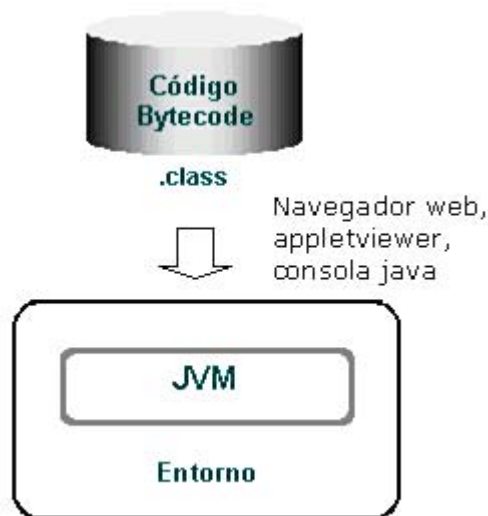
Tradicionalmente se han dividido los lenguajes en compilados e interpretados. Los primeros necesitan ser traducidos por un programa llamado compilador al lenguaje máquina, que es el que entiende el ordenador. Como ejemplo de estos lenguajes podríamos citar a C, C++, Visual Basic, Clipper, etc. Los interpretados, en cambio, son traducidos mientras se ejecutan, por ejemplo HTML, WML o XML, por lo cual no necesitan ser compilados.

Así pues la diferencia entre estos lenguajes radica en la manera de ejecutarlos. Mientras que los compilados sólo se compilan una vez y lo hacen pasando todo el programa a código máquina (si da un error aunque sea en la última línea no podríamos ejecutar nada de nada), en el momento que lo hemos compilado correctamente se genera un archivo .exe que se puede ejecutar tantas veces como queramos sin tener que volver a compilar. Los interpretados en cambio, cada vez que los queramos ejecutar tendremos que interpretarlos línea a línea, es más lento, pero puede ocurrir un error en la última línea y a diferencia de los compilados, el programa se ejecuta justo hasta la línea que produce el error.

Java está diseñado para que un programa escrito en este lenguaje sea ejecutado independientemente de la plataforma (hardware, software y sistema operativo) en la que se esté actuando. Esta portabilidad se consigue haciendo de Java un lenguaje medio interpretado medio compilado. ¿Cómo se come esto? Pues se coge el código fuente, se compila a un lenguaje intermedio cercano al lenguaje máquina pero independiente del ordenador y el sistema operativo en que se ejecuta (llamado en el mundo Java bytecodes).



Finalmente, se interpreta ese lenguaje intermedio por medio de un programa denominado máquina virtual de Java (JVM), que sí depende de la plataforma.



Los java bytecodes permiten el ya conocido “write once, run anywhere” (compila una sola vez y ejecútalo donde quieras). Podemos compilar nuestros programas a bytecodes en cualquier plataforma que tenga el compilador Java. Los bytecodes luego pueden ejecutarse en cualquier implementación de la máquina virtual de Java (JVM). Esto significa que mientras el ordenador tenga un JVM, el mismo programa escrito en Java puede ejecutarse en Windows, Solaris, iMac, Linux, etc.

### La plataforma Java

Con plataforma nos referimos al ambiente de hardware y software en donde el programa se ejecuta, por ejemplo, plataformas como Linux, Solaris, Windows 2003 y MacOS. En casi todos los casos las plataformas son descritas como la combinación del sistema operativo y el hardware. La plataforma Java se diferencia de estas plataformas, es que es una plataforma sólo de software y se ejecuta sobre las otras plataformas de hardware.

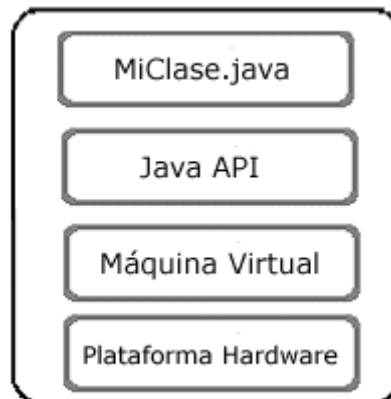
La plataforma Java tiene 2 componentes:

- La máquina virtual de Java (JVM)
- El Java API (Application Programming Interface)

Ya hemos visto algo de la máquina virtual de Java (JVM); es la base de la plataforma Java y es llevada a diferentes plataformas de hardware.

El Java API es una gran colección de componentes de software que proporcionan muchas utilidades para el programador, por ejemplo, los API's para las interfases gráficas. Los API's de Java están agrupados en librerías de ciertas Clases e interfaces, estas librerías son conocidas como paquetes.

El siguiente gráfico describe un programa que se está ejecutando sobre la plataforma Java. Como vemos, el Java API y la máquina virtual aíslan al programa del hardware.



(\*) Lenguajes de alto nivel = son aquellos en los que las instrucciones o sentencias son escritas con palabras similares a las de los lenguajes humanos (mayormente en Inglés). Esto facilita la escritura y comprensión del código al programador.

## Historia

### Orígenes

La plataforma Java y el lenguaje Java empezaron como un proyecto interno de Sun Microsystems en diciembre de 1990. Patrick Naughton, ingeniero de Sun, estaba decepcionado con el estado de C++ y la API de C y sus herramientas. Mientras consideraba migrar

a NeXT, Naughton recibió la oferta de trabajar en una nueva tecnología, y así comenzó el proyecto *Stealth*.

El Proyecto Stealth fue rebautizado, es decir, vuelto a bautizar, como *Green Project* (o *Proyecto Verde*) cuando James Gosling y Mike Sheridan se unieron a Naughton. Con la ayuda de otros ingenieros, empezaron a trabajar en una pequeña oficina en Sand Hill Road en Menlo Park, California. Intentaban desarrollar una nueva tecnología para programar la siguiente generación de *dispositivos inteligentes*, en los que Sun veía un campo nuevo a explotar.

El equipo pensó al principio usar C++, pero se descartó por varias razones. Al estar desarrollando un sistema empotrado con recursos limitados, C++ no es adecuado por necesitar mayor potencia además de que su complejidad conduce a errores de desarrollo. La ausencia de un *recolector de basura* (*garbage collector*) obligaba a los programadores a manejar manualmente el sistema de memoria, una tarea peligrosa y proclive a fallos. El equipo también se encontró con problemas por la falta de herramientas portables en cuanto a seguridad, programación distribuida, y programación concurrente. Finalmente abogaban por una plataforma que fuese fácilmente portable a todo tipo de dispositivo.

Bill Joy había concebido un nuevo lenguaje que combinase lo mejor de *Mesa* y C. En un escrito titulado *Further* (más lejos), proponía a Sun que sus ingenieros crearan un entorno *orientado a objetos* basado en C++. Al principio Gosling intentó modificar y ampliar C++, a lo que llamó C++ ++ --, pero pronto descartó la idea para crear un lenguaje completamente nuevo, al que llamó *Oak*, en referencia al roble que tenía junto a su oficina.

El equipo dedicó largas horas de trabajo y en el verano de 1992 tuvieron lista algunas partes de la plataforma, incluyendo el Sistema Operativo Green, el lenguaje Oak, las librerías y el hardware. La primera prueba, llevada a cabo el 3 de Septiembre de 1992, se centró en construir una PDA (*Personal Digital Assistant* o Asistente Digital Personal) llamada *Star7*<sup>[1]</sup>, que contaba con una interfaz gráfica y un asistente apodado "Duke" para guiar al usuario.

En noviembre de ese mismo año, el Proyecto Verde se convirtió en **FirstPerson, Inc**, una división propiedad de Sun Microsystems, y el equipo se trasladó a Palo Alto (California). El interés se centró entonces en construir dispositivos interactivos, hasta que Time

Warner publicó una solicitud de oferta para un adaptador de televisión. Es decir, un aparato que se sitúa entre la televisión y una fuente de señal externa y que adapta el contenido de ésta (video, audio, páginas Web, etc.) para verse en la pantalla. Entonces, FirstPerson cambió de idea y envió a Warner una propuesta para el dispositivo que deseaban. Sin embargo, la industria del cable consideró que esa propuesta daba demasiado control al usuario, con lo que FirstPerson perdió la puja a favor de Silicon Graphics Incorporated. Un trato con la empresa 3DO para el mismo tipo de dispositivo tampoco llegó a buen puerto. Viendo que no había muchas posibilidades en la industria de la televisión, la compañía volvió al seno de Sun.

### **Críticas:**

Harold dijo en 1995 que Java fue creado para abrir una nueva vía en la gestión de software complejo, y es por regla general aceptado que se ha comportado bien en ese aspecto. Sin embargo no puede decirse que Java no tenga grietas, ni que se adapta completamente a todos los estilos de programación, todos los entornos, o todas las necesidades.

### **E) ADA:**

Es el último intento de obtener un único lenguaje para todo tipo de aplicaciones e incluye los últimos avances en técnicas de programación. Su diseño fue encargado por el Departamento de Defensa de los Estados Unidos a la empresa Honeywell-Bull después de una selección rigurosa entre varias propuestas realizadas sobre una serie de requerimientos del lenguaje y de haber evaluado negativamente veintitrés lenguajes existentes. De éstos se seleccionaron como base para la creación del nuevo lenguaje el PASCAL, el ALGOL y el PL/I. La estandarización del lenguaje se publicó en 1983 con el nombre de ADA en honor de la considerada primera programadora de la historia Augusta Ada Byron, condesa de Lovelace.

Entre las características del lenguaje se encuentran la compilación separada, los tipos abstractos de datos, programación concurrente,

programación estructurada, libertad de formatos de escritura, etc., presentando como principal inconveniente su gran extensión.

## F) ALGOL

### De Wikipedia, la enciclopedia libre

Saltar a [navegación](#), [búsqueda](#)

Se denomina **ALGOL** (o **Algol**) a un lenguaje de programación. La voz es un acrónimo de las palabras inglesas ***Algorithmic Language*** (lenguaje algorítmico).

Fue muy popular en las universidades durante los años 60, pero no llegó a cuajar como lenguaje de utilización comercial.

Sin embargo, Algol influyó profundamente en varios lenguajes posteriores que sí alcanzaron gran difusión, como Pascal, C y Ada.

Hacia 1965 dos corrientes se distinguieron sobre el tema de un sucesor para Algol. Como resultado se definieron los lenguajes Algol W que es un lenguaje *minimalista*, rápidamente implementado y distribuido y, por otra parte, Algol 68 que para la época está en la frontera entre un lenguaje para programar en él y un lenguaje para investigar sobre él.

### Ejemplo de programa en Algol

```
procedure Absmax(a) Dimensiones:(n, m) Resultado:(y)  
Subíndices:(i, k);  
value n, m; array a; integer n, m, i, k; real y;  
comment De la matriz a se toma el elemento con el valor  
absoluto mayor y se coloca en y.  
Los subíndices del elemento se colocan en i y k;  
begin integer p, q;  
y := 0; i := k := 1;  
for p := 1 step 1 until n do  
for q := 1 step 1 until m do  
if abs(a[p, q]) > y then  
begin y := abs(a[p, q]);
```

```
i := p; k := q  
end  
end Absmax
```

## **Algol W**

Lenguaje elaborado diseñado por Niklaus Wirth y Tony Hoare a partir de los trabajos del grupo ALGOL de la IFIP. Se trata de un lenguaje conciso, simple de implementar, que evita todos los defectos conocidos del lenguaje Algol e incluye sus propias características adicionales. Sin embargo, el grupo Algol no lo adoptó como sucesor de Algol prefiriendo en su lugar al que terminó siendo Algol 68. Algol W fue utilizado por gran cantidad de usuarios y sembró el camino para el nacimiento del lenguaje Pascal.

Entre las características del lenguaje se destacan: Aritmética de doble precisión, números complejos, Strings y estructuras de datos dinámicas, evaluación por valor, pasaje de parámetros por valor, valor resultado o resultado.

## **Algol 68**

La definición del lenguaje fue presentada en la reunion del comité ALGOL de la IFIP en 1965. Luego de varios años de revisión del diseño se llegó a una versión definitiva en 1968. Al principal autor es Adriaan Van Wijngarden.

Los objetivos principales de ALGOL 68 son el permitir comunicar algoritmos, el permitir una eficiente ejecución de los mismos en diferentes arquitecturas y el de servir como herramienta para la enseñanza.

Una característica interesante de ALGOL 68 es que su semántica fue definida formalmente antes de ser implementado en base al formalismo llamado gramáticas de dos niveles.

## **G)APL:**

**APL. A Programming Language** o conocido generciamente como *lenguaje de programación A*. Es un intérprete, desarrollado por IBM

a finales de los años 60. Fue definido por Kenneth Iverson. Es un lenguaje muy conciso, con una sintaxis muy sencilla. Está orientado a trabajos con matrices, con la que se pueden hacer todo tipo de operaciones lógicas, aritméticas,... Incluso se pueden inventar las operaciones que se quieren hacer con las matrices.

Es de una potencia tremenda. Una sola sentencia puede traducirse en miles de ellas en otros lenguajes, como por ejemplo Fortran.

Un ejemplo, un lenguaje de simulación de circuitos, SIAL desarrollado por D. Manuel Alfonseca ocupaba cerca de 25 000 sentencias en Fortran-Assembler y en APL todo el programa cabía en dos folios.

A pesar de ser un lenguaje de tan alto nivel también es capaz de manipular a escala de bits.

Tiene la propiedad de que desde una rutina se puede reescribir otra, lo que lo hace muy apropiado para la fabricación de compiladores.

Sus problemas radican en que: 1) Necesita teclado especial para poner los operadores lógicos y simbólicos. 2) Es tan potente que es difícil de documentar.

Más recientemente Kenneth Iverson, estuvo al frente del desarrollo del lenguaje de programación sucesor de APL llamado J. Una de las características particulares de J es lo que se ha dado en denominar programación funcional tácita; donde se considera que para expresar programas no es necesario nombrar variables, ni parámetros a funciones (Estos conceptos de programación tácita han sido incorporados al lenguaje Logo en la biblioteca LogoFE). En J, la variedad de los que en APL se llaman operadores es mucho mayor.

## H) COBOL:

El lenguaje COBOL (acrónimo de **CO**mmon **B**usiness -**O**riented **L**anguage, *Lenguaje Común Orientado a Negocios*) fue creado en el año 1960 con el objetivo de crear un lenguaje de programación

universal que pudiera ser usado en cualquier ordenador, ya que en los años 1960 existían numerosos modelos de ordenadores incompatibles entre sí, y que estuviera orientado principalmente a los negocios, es decir, a la llamada informática de gestión..

## **Características**

COBOL fue dotado por diseño de unas excelentes capacidades de autodocumentación, una buena gestión de archivos y una excelente gestión de los tipos de datos para la época, a través de la conocida sentencia PICTURE para la definición de campos estructurados. Para evitar errores de redondeo en los cálculos que se producen al convertir los números a binario y que son inaceptables en temas comerciales, COBOL puede emplear y emplea por defecto números en base diez. Para facilitar la creación de programas en COBOL, la sintaxis del mismo fue creada de forma que fuese parecida al idioma inglés, evitando el uso de símbolos que se impusieron en lenguajes de programación posteriores.

Pese a esto, a comienzos de los ochenta se fue quedando anticuado respecto a los nuevos paradigmas de programación y a los lenguajes que los implementaban. En la revisión de 1985 se solucionó, incorporando a COBOL variables locales, recursividad, reserva de memoria dinámica y programación estructurada.

En la revisión de 2002 se le añadió orientación a objetos, aunque desde la revisión de 1974 se podía crear un entorno de trabajo similar a la orientación a objetos, y un método de generación de pantallas gráficas estandarizado.

Antes de la inclusión de las nuevas características en el estándar oficial, muchos fabricantes de compiladores las añadían de forma no estándar. En la actualidad este proceso se está viendo con la integración de COBOL con Internet. Existen varios compiladores que permiten emplear COBOL como lenguaje de scripting y de servicio web. También existen compiladores que permiten generar código COBOL para la plataforma .NET y EJB.

## **Programa Hola mundo**

IDENTIFICATION DIVISION.  
Program-Id. Hola-Mundo.

ENVIRONMENT DIVISION.

DATA DIVISION.

PROCEDURE DIVISION.

Main.

DISPLAY "¡Hola Mundo!".

STOP RUN.

## **Empleo**

Pese a que muchas personas creen que el lenguaje COBOL está en desuso, la realidad es que casi todos los sistemas que requieren gran capacidad de procesamiento batch, tanto de los bancos como en otras grandes empresas con sistemas mainframes, utilizan COBOL. Esto permite garantizar la compatibilidad de los sistemas antiguos con los más modernos, así como tener la seguridad de que el lenguaje es perfectamente estable y probado. Según un informe de Gartner Group de 2005, el 75% de los datos generados por negocios son procesados por programas creados en COBOL, y en otro informe de 1997 estima que el 80% de los 300.000 millones de líneas de código existentes están creados en COBOL, escribiéndose 5.000 millones de líneas nuevas de COBOL cada año. Con todo eso, hoy por hoy, la programación en COBOL es uno de los negocios más rentables del mundo de la informática.

## **I) BASIC**

PROVIENE DE LAS SIGLAS BEGINNER'S ALL-PURPOSE SYMBOLIC INSTRUCTION CODE (CÓDIGO DE INSTRUCCIÓN SIMBÓLICO DE USO GENERAL PARA PRINCIPIANTES). SE CREO EN EL DARTMOUTH COLLEGE EN COMBINACIÓN CON EL PRIMER SISTEMA DE TIEMPO COMPARTIDO EN EL MUNDO.

AL PASO DE LOS AÑOS, EL LENGUAJE BASIC SE HA CONVERTIDO EN UNO DE LOS LENGUAJES DE PROGRAMACIÓN MÁS POPULARES Y DE MÁS FÁCIL ACCESO PARA ALMACENES ESPECIALIZADOS. DEBIDO A QUE RESULTA FÁCIL DE UTILIZAR Y QUE LAS NECESIDADES DE

ALMACENAMIENTO DE SU TRADUCTOR DE LENGUAJES SON PEQUEÑAS, TRABAJA CON EFICIENCIA EN CASI TODAS LAS COMPUTADORAS PERSONALES. EXISTEN MUCHAS VERSIONES DEL LENGUAJE BASIC.

## **CARACTERÍSTICAS**

CASI TODA CARACTERÍSTICA PRINCIPAL SE RELACIONA CON SU FACILIDAD DE USO PARA LOS PRINCIPIANTES. ENTRE ESTAS CARACTERÍSTICAS SE ENCUENTRAN LA NOMINACIÓN SIMPLIFICADA DE VARIABLES, EL FORMATO OPCIONAL, EL MODO DE PROGRAMACIÓN CONVERSACIONAL Y EL FÁCIL DIAGNOSTICO.

## **LIMITACIONES**

LAS VERSIONES DE BASIC NO ESTÁN DISEÑADAS PARA FACILITAR LOS PROGRAMAS ESTRUCTURADOS. EXISTEN TANTAS VERSIONES DE BASIC, UN PROGRAMA DESARROLLADO EN UNA COMPUTADORA PUEDE REQUERIR MODIFICACIONES SUSTANCIALES PARA EJECUTARSE EN OTRA MÁQUINA.

## **J) LENGUAJE PL / 1**

Traduce lenguaje de programación 1. Nació alrededor de los años 50. Es un lenguaje de reciente desarrollo orientado hacia los procedimientos.

Elaborado originalmente por la IBM, puede considerarse un lenguaje de aplicación múltiple. Su formato fue

diseñado para incluir las mejores características tanto de FORTRAN como de COBOL.

Es un lenguaje muy poderoso. Permite a los programadores manejar grandes cantidades de datos, realizar análisis estadísticos complicados o reorganizar archivos de datos alfanuméricos.

PL/1 requiere de un computador grande y por lo tanto, de una fracción apreciable de memoria principal.







































- Java no ha aportado capacidades estándares para aritmética en punto flotante. El estándar IEEE 754 para “Estándar para Aritmética Binaria en Punto Flotante” apareció en 1985, y desde entonces es el estándar para la industria. Y aunque la aritmética flotante de Java se basa en gran medida en la norma del IEEE, no soporta aún algunas características. Más información al respecto puede encontrarse en la sección final de enlaces externos.
- A raíz de la naturaleza propietaria de Java, la supuesta inflexibilidad para cambiar, y un creciente estrechamiento de líneas alrededor del sector corporativo, se dice que Java es “el nuevo COBOL”. Aunque puede ser una afirmación exagerada, hace referencia a una preocupación real sobre el panorama de futuro de Java.
- El recolector de basura de Java sólo gestiona la memoria, pero el instante en que tiene lugar su tarea no puede controlarse manualmente. Por tanto, aquellos objetos que reservan recursos externos deben ser desalojados de memoria a mano (usando el mecanismo del lenguaje “finally”), o deben mantenerse hasta que acabe la ejecución del programa.

