

HISTORIA Y  
EVOLUCIÓN  
DE LOS  
LENGUAJES  
DE  
PROGRAMACI  
ÓN

# INDICE

Índice.....	1
Introducción.....	2
Definiciones.....	3
Historia.....	4
Las tendencias de los lenguajes de programación.....	5
Clasificación de los lenguajes de programación.....	9
Algunos lenguajes de programación de alto nivel.....	14
Evolución de los lenguajes de programación.....	24
Bibliografía.....	30

## INTRODUCCIÓN

Los ordenadores no hablan nuestro idioma, son máquinas y como tales, necesitan un lenguaje específico pensado por el hombre para ellas. Además, necesitan constantemente interpretar todas las instrucciones que reciben. Dada la dificultad de comunicación insalvable entre el computador y el programador, pronto aparecieron lenguajes de programación que hacen posible la comunicación con el microprocesador, utilizando términos y símbolos relacionados con el tipo de problema que se debe resolver, mediante el empleo de herramientas que brinda la informática.

Estos lenguajes permiten, por un lado, escribir las operaciones que son necesarias realizar para resolver el problema de un modo parecido a como se escribiría convencionalmente (es decir, redactar adecuadamente el algoritmo de resolución del problema) y, por el otro, se encarga de traducir el algoritmo al lenguaje máquina (proceso conocido como compilación) con lo que se le confiere al programa la capacidad de correr (ser ejecutado) en el ordenador. El ordenador es en realidad tan sólo una máquina virtual, capaz de resolver todos los problemas que los usuarios seamos capaces de expresar mediante un algoritmo (programa).

En la actualidad hay muchos tipos de lenguajes de programación, cada uno de ellos con su propia gramática, su terminología especial y una sintaxis particular. Por ejemplo, existen algunos creados especialmente para aplicaciones científicas o matemáticas generales (BASIC, FORTRAN, PASCAL, etc.); otros, en cambio, se orientan al campo empresarial y al manejo de textos y ficheros, es decir, son en realidad fundamentalmente gestores de información (COBOL, PL/1, etc.), o muy relacionados con el lenguaje máquina del ordenador (como el C y el ASSEMBLER).

Los ordenadores se programaban en lenguaje máquina pero las dificultades que esto conllevaba, junto con la enorme facilidad de cometer errores, cuya localización era larga y compleja, hicieron concebir, en la década de los 40, la posibilidad de usar lenguajes simbólicos. Los primeros en aparecer fueron los ensambladores, fundamentalmente consistía en dar un nombre (mnemónico) a cada tipo de instrucción y cada dirección (etiqueta). Al principio se hacía el programa sobre papel y, después se traducía a mano con la ayuda de unas tablas, y se introducían en la máquina en forma numérica, pero pronto aparecieron programas que se ensamblaban automáticamente.

## DEFINICIONES

Es complicado definir qué es y qué no es un lenguaje de programación. Se asume generalmente que la traducción de las instrucciones a un código que comprende la computadora debe ser completamente sistemática. Normalmente es la computadora la que realiza la traducción.

A continuación, voy a redactar una serie de definiciones de los lenguajes de programación.

Un lenguaje de programación es una notación para escribir programas, a través de los cuales podemos comunicarnos con el hardware y dar así las ordenes adecuadas para la realización de un determinado proceso. Un lenguaje está definido por una gramática o conjunto de reglas que se aplican a un alfabeto constituido por el conjunto de símbolos utilizados. Los distintos niveles de programación existentes nos permiten acceder al hardware, de tal forma que según utilicemos un nivel u otro, así tendremos que utilizar un determinado lenguaje ligado a sus correspondientes traductores.

Conjunto de normas “lingüísticas” (palabras y símbolos) que permiten escribir un programa y que éste sea entendido por el ordenador y pueda ser trasladado a ordenadores similares para su funcionamiento en otros sistemas.

Conjunto de instrucciones, ordenes y símbolos reconocibles por autómatas, a través de su unidad de programación, que le permite ejecutar la secuencia de control deseada. Al conjunto de todas estas instrucciones, ordenes y símbolos que están disponibles se le llaman lenguajes de programación del autómata. El programa está formado por un conjunto de instrucciones, sentencias, bloques funcionales y grafismos que indican las operaciones a realizar. Las instrucciones representan la tarea más elemental de un programa: leer una entrada, realizar una operación, activar una salida, etc. La sentencia representa el mínimo conjunto de instrucciones o sentencias que realizan una tarea o función compleja: encontrar el valor de una función lógica en combinación de varias variables, consultar un conjunto de condiciones, etc. El bloque funcional es el conjunto de instrucciones o sentencias que realizan una tarea o función compleja: contadores,

registros de desplazamientos, transferencias de información, etc. Todos estos elementos están relacionados entre sí mediante los símbolos o grafismos.

Es un conjunto de palabras y símbolos que permiten al usuario generar comandos e instrucciones para que la computadora los ejecute. Los lenguajes de programación deben tener instrucciones que pertenecen a las categorías ya familiares de entrada/salida, cálculo/manipulación, de textos, lógica/comparación, y almacenamiento/recuperación.

## HISTORIA

Los primeros lenguajes de programación surgieron de la idea de Charles Babbage, la cual se le ocurrió a este hombre a mediados del siglo XIX. Era un profesor matemático de la universidad de Cambridge e inventor inglés, que a principios del siglo XIX predijo muchas de las teorías en que se basan los actuales ordenadores. Consistía en lo que él denominaba la máquina analítica, pero que por motivos técnicos no pudo construirse hasta mediados del siglo XX. Con él colaboró Ada Lovelace, la cual es considerada como la primera programadora de la historia, pues realizó programas para aquella supuesta máquina de Babbage, en tarjetas perforadas. Como la máquina no llegó nunca a construirse, los programas de Ada, lógicamente, tampoco llegaron a ejecutarse, pero sí suponen un punto de partida de la programación, sobre todo si observamos que en cuanto se empezó a programar, los programadores utilizaron las técnicas diseñadas por Charles Babbage, y Ada, que consistían entre otras, en la programación mediante tarjetas perforadas. A pesar de ello, Ada ha permanecido como la primera programadora de la historia. Se dice por tanto que estos dos genios de antaño, se adelantaron un siglo a su época, lo cual describe la inteligencia de la que se hallaban dotados.

En 1823 el gobierno Británico lo apoyó para crear el proyecto de una máquina de diferencias, un dispositivo mecánico para efectuar sumas repetidas. Pero Babbage se dedicó al proyecto de la máquina analítica, abandonando la máquina de diferencias, que se pudiera programar con tarjetas perforadas, gracias a la creación de Charles Jacquard (francés). Este hombre era un fabricante de tejidos y había creado un telar que podía reproducir automáticamente patrones de tejidos, leyendo la información codificada en patrones de agujeros perforados en tarjetas de papel rígido. Entonces Babbage intentó crear la máquina que se pudiera programar con tarjetas perforadas para efectuar cualquier cálculo con una precisión de 20 dígitos. Pero la tecnología de la época no bastaba para hacer realidad sus ideas. Si bien las ideas de Babbage no llegaron a materializarse de forma definitiva, su contribución es decisiva, ya que los ordenadores actuales responden a un esquema análogo al de la máquina analítica. En su diseño, la máquina constaba de cinco unidades básicas: 1) Unidad de entrada, para introducir datos e instrucciones; 2) Memoria, donde se almacenaban datos y resultados intermedios; 3) Unidad de control, para regular la secuencia de ejecución de las operaciones; 4) Unidad Aritmético-Lógica, que efectúa las operaciones; 5) Unidad de salida, encargada de comunicar al exterior los resultados. Charles Babbage, conocido

como el "padre de la informática" no pudo completar en aquella época la construcción del computador que había soñado, dado que faltaba algo fundamental: la electrónica. El camino señalado de Babbage, no fue nunca abandonado y siguiéndolo, se construyeron los primeros computadores.

Cuando surgió el primer ordenador, el famoso ENIAC (Electronic Numerical Integrator And Calculator), su programación se basaba en componentes físicos, o sea, que se programaba, cambiando directamente el Hardware de la maquina, exactamente lo que se hacia era cambiar cables de sitio para conseguir así la programación de la maquina. La entrada y salida de datos se realizaba mediante tarjetas perforadas.

## LAS TENDENCIAS DE LOS 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 este trabajo, intentare 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 más parecidos al castellano o inglés en su potencia expresiva y funcionalidad están en el nivel más alto respecto a los otros. Son fundamentalmente lenguajes de ordenes, dominados por sentencias que expresan "Lo que hay que hacer" en vez 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 más rápidamente el lenguaje y usarlo en su trabajo, sin necesidad de programadores o prácticas de programación.

Los lenguajes de "Alto Nivel" son los más 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 máquinas de forma que un programa puede ser fácilmente "Transportado" (Transferido) de una máquina a otra sin una revisión sustancial. En ese sentido se llama "Independientes de la máquina". 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 Máquina" son dependientes de la máquina. Cada tipo de máquina, tal como VAX de digital, tiene su propio lenguaje máquina distinto y su lenguaje ensamblador asociado. El lenguaje Ensamblador es simplemente una representación simbólica del lenguaje máquina 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 máquina, ensambladores de alto nivel:

Como muestra este ejemplo, a más bajo nivel de lenguaje más cerca está de las características de un tipo de máquina particular y más 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 máquina 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 máquina.

La programación de un lenguaje de alto nivel o en un lenguaje ensamblador requiere, por tanto, algún tipo de interfaz con el lenguaje máquina para que el programa pueda ejecutarse. Las tres interfaces más comunes: un "ensamblador", un "compilador" y un "interprete". El ensamblador y el compilador traducen 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 más 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).

## 1

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 más 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 aprendiendo 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 reciente 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 máquinas 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 cálculo lambda 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 influyó 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.

# CLASIFICACION DE LOS LENGUAJES DE PROGRAMACIÓN

## **LENGUAJE MÁQUINA:**

El lenguaje máquina es el único que entiende directamente la computadora, ya que esta escrito en lenguajes directamente inteligibles por la máquina (computadora), utiliza el alfabeto binario, que consta de los dos únicos símbolos 0 y 1, denominados bits (abreviatura inglesa de dígitos binarios). Sus instrucciones son cadenas binarias (cadenas o series de caracteres de dígitos 0 y 1) que especifican una operación y, las posiciones (dirección) de memoria implicadas en la operación se denominan instrucciones de máquina o código maquina. Fue el primer lenguaje utilizado en la programación de computadoras, pero dejó de utilizarse por su dificultad y complicación, siendo sustituido por otros lenguajes más fáciles de aprender y utilizar, que además reducen la posibilidad de cometer errores. El lenguaje máquina es el conocido código binario. Generalmente, en la codificación de los programas se empleaba el sistema hexadecimal para simplificar el trabajo de escritura. Todas las instrucciones preparadas en cualquier lenguaje máquina tienen por lo menos dos partes. La primera es el comando u operación, que dice a las computadoras cual es la función que va a realizar. Todas las computadoras tienen un código de operación para cada una de las funciones. La segunda parte de la instrucción es el operando, que indica a la computadora donde hallar o almacenar los datos y otras instrucciones que se van a manipular, el número de operandoos de una instrucción varía en distintas computadoras.

Ventajas del lenguaje máquina: posibilidad de cargar (transferir un programa a la memoria) sin necesidad de traducción posterior, lo que supone una velocidad de ejecución superior a cualquier otro lenguaje de programación.

Desventajas del lenguaje máquina: dificultad y lentitud en la codificación. Poca fiabilidad. Gran dificultad para verificar y poner a punto los programas. Los programas solo son ejecutables en el mismo procesador (CPU). En la actualidad, las desventajas superan a las ventajas, lo que hace prácticamente no recomendables a los lenguajes máquina.

## **LENGUAJES DE BAJO NIVEL (ensamblador):**

Son más fáciles de utilizar que los lenguajes máquina, pero al igual que ellos, dependen de la máquina en particular. El lenguaje de bajo nivel por excelencia es el ensamblador. El lenguaje ensamblador es el primer intento de sustituir el lenguaje máquina por otro más similar a los utilizados por las personas. Este intenta desflexibilizar la representación de los diferentes campos. Esa flexibilidad se consigue no escribiendo los campos en binario y aproximando la escritura al lenguaje. A principios de la década de los 50 y con el fin de facilitar la labor de los programadores, se desarrollaron códigos mnemotécnicos para las operaciones y direcciones simbólicas. Los códigos mnemotécnicos son los símbolos alfabéticos del lenguaje máquina. La computadora sigue utilizando el lenguaje máquina para procesar los datos, pero los programas ensambladores traducen antes los símbolos de código de operación especificados a sus equivalentes en el lenguaje máquina. En la actualidad los programadores no asignan números de dirección reales a los datos simbólicos, simplemente especifican donde quieren que se coloque la primera localidad del programa y el programa ensamblador se encarga de lo demás, asigna localidades tanto para las instrucciones como los datos. Estos programas de ensamble o ensambladores también permiten a la computadora convertir las instrucciones en lenguaje ensamblador del programador en su propio código máquina. Un programa de instrucciones escrito en lenguaje ensamblador por un programador se llama programa fuente. Después de que el ensamblador convierte el programa fuente en código máquina a este se le denomina programa objeto. Para los programadores es más fácil escribir instrucciones en un lenguaje ensamblador que en código de lenguaje máquina pero es posible que se requieran dos corridas de computadora antes de que se puedan utilizar las instrucciones del programa fuente para producir las salidas deseadas.

El lenguaje de bajo nivel es el lenguaje de programación que el ordenador puede entender a la hora de ejecutar programas, lo que aumenta su velocidad de ejecución, pues no necesita un intérprete que traduzca cada línea de instrucciones.

Visto a muy bajo nivel, los microprocesadores procesan exclusivamente señales electrónicas binarias. Dar una instrucción a un microprocesador supone en realidad enviar series de unos y ceros espaciadas en el tiempo de una forma determinada. Esta secuencia de señales se denomina código máquina. El código representa normalmente datos y números e instrucciones para manipularlos. Un modo más fácil de comprender el código máquina es dando a cada instrucción un mnemónico, como por ejemplo STORE, ADD o JUMP. Esta abstracción da como resultado el ensamblador, un lenguaje de muy bajo nivel que es específico de cada microprocesador.

Los lenguajes de bajo nivel permiten crear programas muy rápidos, pero que son, a menudo, difíciles de aprender. Más importante es el hecho de que los programas escritos en un bajo nivel sean altamente específicos de cada procesador. Si se lleva el programa a otra máquina se debe reescribir el programa desde el principio.

Ventajas del lenguaje ensamblador frente al lenguaje máquina: mayor facilidad de codificación y, en general, su velocidad de cálculo, ahorran tiempo y requieren menos atención a detalles. Se incurren en menos errores y los que se cometen son más fáciles de localizar. Tanto el lenguaje máquina como el ensamblador gozan de la ventaja de mínima ocupación de memoria y mínimo tiempo de ejecución en comparación con el resultado de la compilación del programa equivalente escrito en otros lenguajes. Los

programas en lenguaje ensamblador son más fáciles de modificar que los programas en lenguaje máquina.

Desventajas del lenguaje ensamblador: dependencia total de la máquina lo que impide la transportabilidad de los programas (posibilidad de ejecutar un programa en diferentes máquinas). El lenguaje ensamblador del PC es distinto del lenguaje ensamblador del Apple Machintosh. La formación de los programadores es más compleja que la correspondiente a los programadores de alto nivel, ya que exige no solo las técnicas de programación, sino también el conocimiento del interior de la máquina. El programador ha de conocer perfectamente el hardware del equipo, ya que maneja directamente las posiciones de memoria, registros del procesador y demás elementos físicos. Todas las instrucciones son elementales, es decir, en el programa se deben describir con el máximo detalle todas las operaciones que se han de efectuar en la máquina para la realización de cualquier proceso.

Los lenguajes ensamblador tienen sus aplicaciones muy reducidas, se centran básicamente en aplicaciones de tiempo real, control de procesos y de dispositivos electrónicos.

### **LENGUAJES DE ALTO NIVEL:**

Estos lenguajes son los más utilizados por los programadores. Están diseñados para que las personas escriban y entiendan los programas de un modo mucho más fácil que los lenguajes máquina y ensamblador. Un programa escrito en lenguaje de alto nivel es independiente de la máquina (las instrucciones no dependen del diseño del hardware o de una computadora en particular), por lo que estos programas son portables o transportables. Los programas escritos en lenguaje de alto nivel pueden ser ejecutados con poca o ninguna modificación en diferentes tipos de computadoras. Son lenguajes de programación en los que las instrucciones enviadas para que el ordenador ejecute ciertas órdenes son similares al lenguaje humano. Dado que el ordenador no es capaz de reconocer estas órdenes, es necesario el uso de un intérprete que traduzca el lenguaje de alto nivel a un lenguaje de bajo nivel que el sistema pueda entender.

Por lo general se piensa que los ordenadores son máquinas que realizan tareas de cálculos o procesamiento de texto. La descripción anterior es sólo una forma muy esquemática de ver una computadora. Hay un alto nivel de abstracción entre lo que se pide a la computadora y lo que realmente comprende. Existe también una relación compleja entre los lenguajes de alto nivel y el código máquina.

Los lenguajes de alto nivel son normalmente fáciles de aprender porque están formados por elementos de lenguajes naturales, como el inglés. En BASIC, el lenguaje de alto nivel más conocido, los comandos como "IF CONTADOR=10 THEN STOP" pueden utilizarse para pedir a la computadora que pare si CONTADOR es igual a diez. Por desgracia para muchas personas esta forma de trabajar es un poco frustrante, dado que a pesar de que las computadoras parecen comprender un lenguaje natural, lo hacen en realidad de una forma rígida y sistemática.

Los lenguajes de alto nivel, también denominados lenguajes evolucionados, surgen con posterioridad a los anteriores (lenguaje máquina, lenguajes de bajo nivel o ensamblador) con los siguientes objetivos, entre otros:

Lograr independencia de la maquina, pudiendo utilizar un mismo programa en diferentes equipos con la única condición de disponer de un programa traductor o compilador, que es suministrado por el fabricante, para obtener el programa ejecutable en lenguaje binario de la maquina que se trate. Además, no se necesita conocer el hardware específico de dicha maquina. 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 maquina, ya que se utilizan palabras (en ingles) en lugar de cadenas de símbolos sin ningún significado aparente. Incluir rutinas de uso frecuente, como las de entrada / salida, funciones matemáticas, manejo de tablas, etc., que figuran en una especie de librería del lenguaje, de manera que se puedan utilizar siempre que se quiera sin necesidad de programarlas cada vez.

Ventajas de los lenguajes de alto nivel: el tiempo de formación de los programadores es relativamente corto comparado con otros lenguajes. La escritura de programas se basa en reglas sintácticas similares a los lenguajes humanos, nombres de las instrucciones tales como READ, WRITE, PRINT, OPEN, etc. Las modificaciones y puestas a punto de los programas son más fáciles. Reducción del costo de los programas. Transportabilidad. Permiten tener una mejor documentación. Son más fáciles de mantener.

Desventajas de los lenguajes de alto nivel: incremento del tiempo de puesta a punto al necesitarse diferentes traducciones del programa fuente para conseguir el programa definitivo. No se aprovechan los recursos internos de la maquina que se explotan mucho mejor en lenguajes máquina y ensambladores. Aumento de la ocupación de memoria. El tiempo de ejecución de los programas es mucho mayor.

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, además de las diferentes versiones o dialectos que se han desarrollado de algunos de ellos. Es difícil establecer una clasificación general de los mismos, ya que en cualquiera que se realice habrá lenguajes que pertenezcan a más de uno de los grupos establecidos. Una clasificación muy extendida, atendiendo a la forma de trabajar de los programas y a la filosofía con que fueron concebidos, es la siguiente:

- **Lenguajes imperativos.** Utilizan instrucciones como unidad de trabajo de los programas (Cobol, Pascal, C, Ada).
- **Lenguajes declarativos.** Los programas se construyen mediante descripciones de funciones o expresiones lógicas (Lisp, Prolog).
- **Lenguajes orientados a objetos.** El diseño de los programas se basa más en los datos y su estructura. La unidad de proceso es el objeto y en él se incluyen los datos (variables) y las operaciones que actúan sobre ellos (Smalltalk, C++).
- **Lenguajes orientados al problema.** Diseñados para problemas específicos, principalmente de gestión, suelen ser generadores de aplicaciones.
- **Lenguajes naturales.** Están desarrollándose nuevos lenguajes con el principal objetivo de aproximar el diseño y construcción de programas al lenguaje de las personas.

Otra clasificación que se puede hacer es la de atendiendo al desarrollo de los lenguajes desde la aparición de las computadoras, que sigue un cierto paralelismo con las generaciones establecidas en la evolución de las mismas:

- **Primera generación.** Lenguajes maquina y ensambladores.
- **Segunda generación.** Primeros lenguajes de alto nivel imperativo (FROTRAN, COBOL).
- **Tercera generación.** Lenguajes de alto nivel imperativo. Son los mas utilizados y siguen vigentes en la actualidad (ALGOL 8, PL/I, PASCAL, MODULA).
- **Cuarta generación.** Orientados básicamente a las aplicaciones de gestión y al manejo de bases de datos (NATURAL, SQL).
- **Quinta generación.** Orientados a la inteligencia artificial y al procesamiento de los lenguajes naturales (LISP, PROLOG).

Para la mejor comprensión se harán unas definiciones:

**Programa:** es un conjunto de instrucciones escritas en un lenguaje de programación que indican a la computadora la secuencia de pasos, para resolver un problema.

**Código fuente:** esta creado en algún lenguaje de alto nivel, por lo que es entendido 100% por el ser humano. Este debe estar complementado por su documentación o manuales donde se indica el desarrollo lógico del mismo.

**Código objeto:** es creado por los compiladores y nos sirve como enlace entre el programa fuente y el ejecutable.

## ALGUNOS LENGUAJES DE PROGRAMACIÓN DE ALTO NIVEL

A continuación se presentan varios de los más conocidos y utilizados, lenguajes de alto nivel.

### **FORTRAN**

Abreviatura de FORMula TRANslator (traductor de formulas), fue definido alrededor del año 1955 en Estados Unidos por la compañía IBM. Es el más antiguo de los lenguajes de alto nivel. Antes de él, todos los programas se escribían en lenguaje ensamblador o en lenguaje máquina. Es un lenguaje especializado en aplicaciones técnicas y científicas. Se caracteriza por su potencia en los cálculos matemáticos, pero está limitado en las aplicaciones de gestión, manejo de archivos, tratamiento de cadenas de caracteres y edición de informes. Es un lenguaje notorio, por la facilidad con que permite expresar una ecuación. Muchas de sus características fueron incorporadas más tarde en el primer lenguaje BASIC. Una de sus ventajas es que es un lenguaje compacto y es también ampliamente utilizado para aplicaciones en los negocios que no requieren manejo de grandes archivos de datos. Hasta 1961 se mantuvo como monopolio de IBM, pero posteriormente se fue implementando en ordenadores de otros fabricantes. A lo largo de su existencia han aparecido diferentes versiones, entre las que destaca la adoptada en 1966 por el ANSI (American National Standards Institute), en la que se definieron nuevas reglas del lenguaje y se logró la independencia del mismo con respecto a la máquina; es decir, comenzó la portabilidad del lenguaje. Esta versión se denominó FORTRAN IV o FORTRAN 66, y el idioma se hizo tan popular en los años 60, que FORTRAN 66 se volvió el primer idioma en ser regularizado oficialmente en 1972. En 1977 apareció una nueva versión más evolucionada que se llamó FORTRAN V o FORTRAN 77. Está reflejada en el documento ANS 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. 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. A mediados de los años setenta se proporcionaron virtualmente cada computadora, mini o mainframe, con un sistema FORTRAN 66 normal. Era por

consiguiente posible escribir programas en FORTRAN en cualquier sistema y estar bastante seguro que estos pudieran moverse para trabajar en cualquier otro sistema bastante fácil. Esto, y el hecho que pudieran procesarse programas de FORTRAN muy eficazmente. La última normalización del lenguaje, FORTRAN 90, se encuentra en el documento ANS X3.198-1991 en la que se incluyen características como la recursividad, tratamiento paralelo de tablas y uso de memoria dinámica. Permite expresar los programas de maneras que se satisfacen más a un ambiente de la informática moderna y han quedado obsoletos muchos de los mecanismos que eran apropiados en FORTRAN 77. En FORTRAN 90 algunos rasgos de FORTRAN 77 han sido reemplazados por rasgos mejores, más seguros y más eficaces, muchos de estos fueron quitados del idioma FORTRAN 95. El FORTRAN tiene la ventaja de ser un lenguaje compacto que sirve muy bien para satisfacer las necesidades de los científicos y los estadísticos de los negocios.

### **COBOL**

Es el lenguaje más utilizado en las aplicaciones de gestión, creado en 1960 por un comité denominado CODASYL (COference on DATA SYstems Languages), patrocinado por el Departamento de Defensa de Estados Unidos, a fin de disponer de un lenguaje universal para aplicaciones comerciales, como expresa su nombre (COmmon Business Oriented Language).

A lo largo de su existencia ha sufrido diversas actualizaciones. Su primer estándar fue aprobado por el ANSI en 1968. Posteriormente, en 1974, se adopta la norma ANS X3.23-1974, que ha perdurado hasta su última versión, COBOL ANS-85, que facilita el diseño estructurado de los programas.

Sus características más destacables son las siguientes: se asemeja al lenguaje natural (inglés), es autodocumentado y ofrece grandes facilidades en el manejo de archivos, así como en la edición de informes escritos. Puede emplear términos comúnmente utilizados en los negocios.

Entre sus inconvenientes están sus rígidas reglas de formatos de escritura, la necesidad de escribir todos los elementos al máximo detalle, la extensión excesiva en sus sentencias, e incluso duplicación en algunos casos, y la inexistencia de funciones matemáticas.

No obstante, se puede afirmar que en la actualidad continúa siendo el lenguaje más utilizado en las aplicaciones de gestión.

### **PL/I**

Fue creado a comienzos de los años sesenta por IBM para ser usado en sus equipos del sistema 360. Inspirándose en los lenguajes ALGOL, COBOL y FORTRAN se desarrolló el PL/I (Programming Language/I) tomando las mejores características de los anteriores y añadiendo algunas nuevas, con el objetivo de obtener un lenguaje lo más general posible en cuanto a su implementación, útil para aplicaciones técnico-científicas, comerciales, de proceso de textos, de bases de datos y de programación de sistemas. Se trata de un lenguaje de programación complejo. Compilado y estructurado, es capaz de gestionar errores y de procesar multitareas, y se emplea en entornos académicos y de investigación.

Entre sus novedades está su gran libertad en el formato de escritura de los programas: soporta la programación estructurada y diseño modular. Es un lenguaje flexible y sofisticado. No obstante, no ha superado a sus progenitores en sus aplicaciones

específicas, debido en parte a su amplitud y, por ello, al tamaño de su compilador que hasta ahora solo se podía instalar en grandes equipos. El elemento básico de este programa es el enunciado que termina en punto y coma. Los enunciados se combinan en procedimientos. Un procedimiento puede representar por completo a un programa pequeño o un “bloque de construcción” o módulo de un programa más complejo.

## **BASIC**

El lenguaje BASIC fue diseñado por los profesores John G. Kemeny y Thomas E. Kurtz del Dartmouth College (Estados Unidos) en 1965, con el objetivo principal de proporcionar a los principiantes un lenguaje fácil de aprender, como se indica en su nombre Beginner's All-purpose Symbolic Instruction Code (Código de instrucciones simbólico de propósito general para principiantes). Es un lenguaje interactivo muy popular que tiene una aceptación debido a la facilidad de su uso, es un idioma simple para aprender y fácil de traducir. Que sé interactivo, permite la comunicación directa entre el usuario y el sistema de computo durante la preparación y uso de los programas. Entre sus principales novedades están las de ser un lenguaje interpretado y de uso conversacional, útil para aplicaciones técnicas y de gestión. Esto, unido a la popularización de las microcomputadoras y computadoras personales, ha hecho que su utilización sea haya extendido enormemente, a la vez que ha propiciado el surgimiento de una gran diversidad de diversiones que extienden y se adaptan a necesidades particulares el lenguaje original. Existen multitud de interpretes y compiladores del lenguaje.

## **PASCAL**

Fue creado por el matemático suizo Nicklaus Wirth en 1970, basándose en el lenguaje ALGOL, en cuyo diseño había participado en los años sesenta. Su nombre proviene del filósofo y matemático francés del siglo XVII, Blaise Pascal, que invento la primera maquina tipo mecánico para sumar. Fue el primer gran lenguaje creado después de haber sido ampliamente diseminados los conceptos asociados con la programación estructurada.

Aunque en principio la idea del diseñador era proporcionar un lenguaje adecuado para la enseñanza de los conceptos y técnicas de programación, con el tiempo ha llegado a ser un lenguaje ampliamente utilizado en todo tipo de aplicaciones, que posee grandes facilidades para la programación de sistemas y diseño grafico.

Aporta los conceptos de tipo de datos, programación estructurada y diseño descendente, entre otros, además de haberse convertido en predecesor de otros lenguajes más modernos, como MODULA-2 y ADA.

## **C**

Este lenguaje fue creado en 1972 por Dennis Ritchie a partir del trabajo elaborado por su colega de los laboratorios Bell Telephone, Ken Thompson. Estos habían diseñado con anterioridad el sistema operativo UNIX, y su intención al desarrollar el lenguaje C fue la de conseguir un lenguaje idóneo para la programación de sistemas que fuese independiente de la maquina, con el cual escribir su sistema UNIX.

Aunque, como acabo de decir, fue diseñado inicialmente para la programación de sistemas, posteriormente su uso se ha extendido a ablaciones técnico-científicas, de bases de datos, de proceso de textos, etc.

En 1980 Bjarne Stroustrup, inspirado en el lenguaje Simula67 (adicionó las características de la programación orientada a objetos incluyendo la ventaja de una biblioteca de funciones orientadas a objetos) y lo denominó C con clases. Para 1983 dicha denominación cambió a la de C++. Con este nuevo enfoque surge la nueva metodología que aumenta las posibilidades de la programación bajo nuevos conceptos. La utilización óptima de este lenguaje se consigue dentro de su entorno natural, que es el sistema operativo UNIX, y entre sus características destaca el uso de programación estructurada para resolver tareas de bajo nivel, así como la amplia librería de rutinas de que dispone. El lenguaje C reúne características de programación intermedia entre los lenguajes ensambladores y los lenguajes de alto nivel; con gran poderío basado en sus operaciones a nivel de bits (propias de ensambladores) y la mayoría de los elementos de la programación estructurada de los lenguajes de alto nivel, por lo que resulta ser el lenguaje preferido para el desarrollo de software de sistemas y aplicaciones profesionales de la programación de computadoras.

### **MODULA-2**

El lenguaje MODULA fue diseñado en 1977 bajo la dirección de Niklaus Wirth, creador también del lenguaje PASCAL, con la intención de incluir las necesidades de la programación de sistemas y dar respuestas a las críticas recibidas respecto de las carencias del lenguaje PASCAL. En 1979 se realizó una versión que pasó a denominarse MODULA-2 y que perdura en la actualidad.

Además de incluir las características de su predecesor, este nuevo lenguaje incorpora las principales carencias de aquel, como la posibilidad de compilación separada, creación de librerías, programación concurrente, mejora del manejo de cadenas de caracteres, los procedimientos de entrada/salida y la gestión de la memoria, etc. Además, posee grandes facilidades para la programación de sistemas.

También, debido a sus cualidades didácticas, ha sido ampliamente aceptado por la comunidad universitaria como herramienta idónea para la enseñanza de la programación.

### **ADA**

Es el último intento de obtener un único lenguaje para todo tipo de aplicaciones, e incluso los últimos avances de técnicas de programación. Su diseño fue encargado por el Departamento de Defensa de Estados Unidos, para su uso en servicios militares, 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 estos, 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. Como principal inconveniente presenta su gran extensión. Los escritores lo llamaron inflexible e ineficiente, en tanto que sus favorecedores lo consideraban un gran avance en la tecnología del software.

### **LISP**

En informática, acrónimo de List Processing. Un lenguaje de programación para ordenadores o computadoras orientado a la generación de listas, desarrollado en 1959-1960 por John McCarthy y usado principalmente para manipular listas de datos o de símbolos. El lenguaje LISP constituyó un cambio radical con respecto a los lenguajes procedurales (FORTRAN, ALGOL) que se desarrollaban por entonces. El LISP es un lenguaje interpretado, en el que cada expresión es una lista de llamadas a funciones. Este lenguaje se sigue utilizando con frecuencia en investigación y en círculos académicos, y fue considerado durante mucho tiempo el lenguaje modelo para la investigación de la inteligencia artificial (IA), aunque el Prolog ha ganado terreno durante los últimos años.

### **LOGO**

En informática, lenguaje de programación de ordenadores o computadoras, desarrollado en 1968 por Seymour Papert en el MIT, que se usa frecuentemente en la enseñanza de lenguaje de programación a niños. Una característica importante de Logo son los gráficos de tortuga, que permiten al programador hacer dibujos simples dirigiendo los movimientos de la tortuga en la pantalla hacia adelante, hacia la derecha o la izquierda. Una vez que dominan el entorno sencillo del dibujo, el programador (normalmente un niño o una niña) empieza a descubrir las características más sofisticadas del lenguaje, que están basadas fundamentalmente en el lenguaje de programación LISP. Logo está considerado como un lenguaje para la formación, a pesar de que algunas empresas intentaron que tuviera una mayor aceptación en los círculos profesionales de programación.

### **RPG**

Report Program Operator fue introducido en 1960 como un lenguaje para duplicar rápidamente el enfoque de proceso utilizado con un equipo de tarjeta perforada. Este lenguaje fue desarrollado por IBM en 1964. Su uso está aun limitado sobre todo para las aplicaciones de negocios que son procesadas en pequeñas computadoras, generar informes comerciales o de negocios. Como su nombre lo sugiere, el RPG está diseñado para generar los reportes de salida que resultan del proceso de aplicaciones de negocios. A pesar de las aplicaciones de actualización de archivos, el RPG es un lenguaje de propósito limitado porque los programas objeto generados por el compilador de RPG siguen sin desviación, un ciclo de procesamiento básico.

Una ventaja del RPG es la relativa facilidad para aprenderlo y usarlo. Dado que la lógica de la programación es fija, existen menos reglas formales que en otros lenguajes.

### **ALGOL**

El ALGOL (ALGOritmic Language) fue presentado en 1958. Fue el primer lenguaje de programación de proceso estructurado de alto nivel. Fue orientado al uso de quienes participan en proyectos científicos y matemáticos. Un grupo internacional de matemáticos europeos y americanos, pretendían crear un lenguaje común normalizado que les permitiera el intercambio de algoritmos, aunque esta en desuso, fue el primero que incorporó conceptos claves para la programación actual.

## **APL**

Sus siglas significan (A Programming Language). Un Lenguaje de Programación. Este programa fue desarrollado por Kenneth Inverson en el año 1961 para resolver problemas matemáticos. Este lenguaje se caracteriza por su brevedad y por su capacidad de generación de matrices y se utiliza en el desarrollo de modelos matemáticos.

## **PILOT**

Programmend Inquiry Language Or Teaching (Consulta, lenguaje o aprendizaje de investigación programada) creado en 1969.

Este lenguaje de programación es utilizado fundamentalmente para crear aplicaciones destinadas a instrucciones asistidas por computadoras. Se caracteriza por utilizar un mínimo de sintaxis.

## **SMALLTALK**

SMALLTALK, Lenguaje de Programación orientado a objetos integrados con un entorno de desarrollo multiventana. SMALLTALK no es solo un hermoso lenguaje de computación orientado a objetos. El entorno de desarrollo merece similar valoración y ha sido copiado muchas veces, desde el Sistema Operativo de Apple MS Windows y Borland Pascal (en una memoria extensión). Muchos conceptos de SMALLTALK como los browsers y las técnicas de browsing han encontrado hoy su rumbo en muchas herramientas de desarrollo de la generación X, desarrollado por SMALLTALK poseen un factor “divertido-de-usar”. Los cambios se graban instantáneamente y los mismos pueden probarse rápidamente.

SMALLTALK fue desarrollado dentro del Grupo de Investigación del Aprendizaje en el Centro de Investigación de Xerox en Palo Alto a comienzos de los 70. Las principales ideas de SMALLTALK se le atribuyen generalmente a Alan Kay con raíces en Simula, LISP y SketchPad. Dan Ingalls escribió el código de las primeras ventanas solapables, los pop-up menús y la clase BitBlit. Adele Goldberg y Dave Robson escribieron los manuales de referencia para SMALLTALK y fueron miembros clave del equipo de desarrollo. Un programa de licenciamiento de Xerox y Xerox Special Information Systems. Sin embargo la distribución generalizada a la comunidad de desarrollo no sucedió hasta la fundación de una nueva compañía llamada ParcPlace Systems Inc. , Dirigida por Adele Goldberg.

Un segundo SMALLTALK (SMALLTALK 4) fue desarrollado por Digital en los Angeles California. Este SMALLTALK estaba dirigido a cubrir la necesidad de un producto pequeño, de alta velocidad, basado en PC.

Object Technology International Inc. (OTI) desarrolló un conjunto de herramientas para proveer el control de inversiones y el manejo de configuraciones en grandes proyectos. IBM desarrolló la familia de productos VisualAge para SMALLTALK en colaboración con Object Technology (antiguamente ParcPlace-Digital) e IBM permanecen como los distribuidores dominantes de entornos de desarrollos en SMALLTALK. Algunos nuevos SMALLTALK se hallan en etapa de desarrollo.

## **FORTH**

Lenguaje de cuarta generación, creado en 1970, es un lenguaje estructurado e interpretado de fácil ampliación y ofrece una alta funcionalidad en un espacio reducido.

Es un lenguaje de alto nivel del cual derivan en la actualidad casi todos los lenguajes empleados en los robots.

### **LENGUAJE C++**

Se pronuncia “ce plus plus”. Fue desarrollada por Bjarne Stroustrup en los Bell Laboratories a principios de la década de los 80. C++ introduce la programación orientada al objeto en C. Es un lenguaje extremadamente poderoso y eficiente. C++ es un super conjunto de C, para aprender C++ significa aprender todo de C, luego aprender programación orientada al objeto y el uso de éstas con C++.

### **DELPHI**

Es un entorno de programación visual orientado a objetos para desarrollo rápido de aplicaciones (RAD) de propósito general, incluyendo aplicaciones cliente/servidor.

Delphi es la versión de Delphi para 32 bits (delphi 3), es decir son casi los mismos, con la única diferencia que Delphi 3 es mucho más mejorado, por ejemplo contiene un TeeChart, que sirve para los gráficos de negocio.

Delphi tiene las siguiente características:

- Rendimiento - con el mejor y más rápido compilador del mundo.
- Empresa e Internet - soluciones cliente y servicio
- Desarrollo de aplicaciones rápidas (RAD).
- Reusabilidad de componentes, un verdadero entorno orientado a objetos.
- Manejo de Base de Datos escalables.
- Arquitectura multinivel abierta y dimensionable.
- Diseminación de información de base de datos en la Web a una gran velocidad.

### **JAVA**

Es un lenguaje de programación para crear programas seguros, portátiles, orientados a objetos interactivos, para mejorar la entrega de información a través de Internet, etc.

### **JAVASCRIPT**

Este lenguaje de programación originalmente fue llamado LIVESCRIPT, pero luego fue renombrado con el nombre de JAVASCRIPT, con la idea de capitalizar la fama de Java, lenguaje desarrollado por Sun Microsystems. Éste es un complemento ideal del lenguaje HTML, al permitir a la página realizar algunas tareas por si misma, sin necesidad de estar sobrecargando el servidor del cual depende; JAVASCRIPT es un lenguaje diseñado especialmente para ejecutarlo en internet.

Entre estas tareas, puede estar, por ejemplo, realizar algunos cálculos simples, formatear un texto para que sea leído por distintas personas de manera distinta, proveer de un medio de configurar la visualización de una página, realizar un prechequeo de validación en formulario antes de enviarlo, etc.

### **HTML**

El lenguaje HTML, sirve para realizar esas atractivas páginas Web. Se trata de un sistema de marcas que permite enlazar al mismo tiempo texto, sonidos y gráficos dentro del mismo documento, con otros dentro del servidor o incluso con otros servidores

WWW. Es decir, es un editor para combinar textos, imágenes e incluso sonido y ahora también imágenes en movimiento. Es, en definitiva, la forma de manejar y presentar la información en la red.

Para escribir documentos de hipertexto se ha desarrollado un nuevo formato de datos o lenguaje llamado Hyper Text Markup Language (HTML). Este lenguaje permite dar indicaciones precisas al programa cliente de cómo debe presentarse el documento en pantalla o al ser impreso.

El lenguaje HTML es el usado actualmente para escribir textos Hypermediales en el web.

Tres normas fundamentales:

#### 1.- HTML simplemente texto

Lo primero es saber que un documento HTML es un archivo de texto simple, luego, se puede editar con cualquier editor de textos.

#### 2.- No importan los Tabs ni los saltos de línea

Los interpretes HTML no toman en cuenta las tabulaciones, los saltos de líneas ni los espacios en blanco extra. Esto tiene ventajas o desventajas. La principal ventaja es que permite obtener resultados uniformes y de buena presentación de manera bastante fácil. La principal desventaja es que un documento HTML, por lo menos se debe usar los comandos `<P>... </P>` o `<BR>` para evitar que quede todo el texto en una sola línea.

#### 3.- Existen 3 caracteres especiales:

- `<` menor que, se usa para indicar el comienzo de un comando HTML
- `>` mayor que, se usa para indicar el término de un comando HTML.
- `&` Ampersand, se usa para escribir caracteres especiales (símbolos matemáticos, comerciales, así como el signo menor que y el mayor que entre otros) en un documento.

Lo primero es conocer los comandos que debe contener todo documento HTML de más de una línea de largo:

#### EL COMANDO PÁRRAFO:

El comando `<P>... </P>` se utiliza como un delimitador de párrafo en HTML. Inserta automáticamente un quiebre de línea al final del párrafo, y produce un espaciado conveniente entre los diferentes párrafos de un documento. También en forma adicional permite alinear el texto al centro, a la izquierda o a la derecha.

#### EL COMANDO QUIEBRE DE LÍNEAS:

El comando `< BR>` permite hacer un quiebre (salto)de línea.

#### LOS COMANDOS DE ENCABEZADO:

Los textos en HTML poseen seis niveles de encabezado. Por ejemplo el nivel 1 se usa para las divisiones mayores de texto, el nivel de encabezado 6 se usa para las divisiones más chicas de texto.

#### ESTRUCTURA DE HIPERTEXTO:

Existen dos partes fundamentales de un documento HTML.

#### **ENCABEZADO:**

Se inicia mediante el comando <HEAD> y se termina con </HEAD>. Por lo general se incluyen aquí el título del documento, mediante el comando <TITLE>... </TITLE>.

#### **CUERPO:**

Se inicia mediante el comando <BODY> y se termina con el comando. Dentro del cuerpo del documento se incluyen cualquier carácter imprimible. Además es importante incluir el comando <ADDRESS>... </ADDRESS> al final del cuerpo pero dentro de él. Dentro del ADDRESS se escribe el nombre del autor del documento, la organización a la que pertenece, su dirección del correo electrónico y otra información que se considere relevante.

#### **HYPERTALK**

"HyperTalk" es el lenguaje desarrollado por Dan Winkler para Bill Atkinson, el creador del "HyperCard" para Apple-Macintosh. Está orientado a la creación de aplicaciones conforme al sistema de "hiperarchivos" (sistemas de fichas interrelacionadas donde se facilita el "navegar" de un archivo a otro).

HyperTalk es un buen ejemplo de lenguaje orientado a objetos. Este tipo de lenguaje combina la lógica declarativa con los algoritmos (Vea "PROLOG"). Un programa ya no es una secuencia de instrucciones sino un conjunto de objetos agrupados en conjuntos, definidos mediante atributos y a los cuales pueden asociarse instrucciones. Así, en HyperCard, existen archivos ("stacks" o "pilas") que agrupan fichas ("cards"), y cada una de éstas contiene campos de datos y botones. Todos son "objetos" que -si bien mantienen entre sí una relación jerárquica- tienen asociados paquetes de instrucciones ("scripts") independientes unos de otros. Cada objeto pertenece a un conjunto (como fichas o botones) que tiene "atributos" propios comunes a todos sus miembros, y cada atributo tendrá un valor común o específico para cada caso. Para dar o buscar dicho valor intervienen "facetas" que son instrucciones (procedimientos) asociadas.

#### **Perl**

Es un lenguaje especializado en el procesamiento de textos, particularmente extraer y validar las respuestas a cuestionarios incluidos en páginas web.

#### **PHP**

Lenguaje que se acopla al HTML (páginas web) para definir procedimientos que ha de realizar el servidor de web, por ejemplo procesar un formulario, enviar o extraer datos de una base de datos (acoplándose también con un lenguaje de tipo SQL), enviar una u otra página web según determinadas condiciones prefijadas por el programador, etc.

#### **PROLOG**

Los primeros años de la década del 70 son conocidos como un período de "crisis del software", en que se descubrió que la creación de buenos programas involucraba costos mayores que los del hardware que los ejecuta. También se hacía patente una creciente necesidad de procesar "conocimientos" (algo mucho más amplio y complejo que los datos cuantitativos o meras "secuencias de caracteres" a los cuales se reducen muchos lenguajes de programación). Esta crisis llevó a investigar numerosas alternativas, entre

las cuales nuevos lenguajes no basados en instrucciones algorítmicas o procedimientos. Si el hombre "procesa" más información por inferencia lógica que por cálculo, ¿no podría la máquina hacer lo mismo?

PROLOG ("PROgramación en LOGica") es una respuesta a esta crisis, producto del avance de la lógica moderna (de tipo funcional). Lo crearon A. Colmenauer y Ph. Roussel, con la colaboración de R. Kowalski, simultáneamente en la Universidad de Aix-Marseille (Francia) y Edimburgo (Gran Bretaña). Se basa en el cálculo de predicados de primer orden y en el principio de resolución de Robinson. En vez de ser algorítmico ("procedural" en inglés, término sin traducción), es decir concebido como un conjunto de instrucciones que la máquina debe ejecutar en forma secuencial, es "declarativo", es decir basado en definiciones (de "hechos" o "reglas", como se explica más abajo).

### **SQL**

Lenguaje desarrollado especialmente para facilitar la consulta de bases de datos (BD), acotando progresivamente la búsqueda (de ahí el nombre de "Sequential Query Language").

Existen hoy numerosas aplicaciones de administración de bases de datos que recurren al SQL (Las más conocidas, potentes - y caras - son Oracle e Informix).

Hoy se pueden acoplar las bases de datos a hipertextos (páginas web), para lo cual las buenas aplicaciones ya traen módulos que hacen la conexión. El lenguaje PHP del cual hablamos más arriba también sirve para definir procedimientos de inserción y de consulta de datos en BD que funcionan con SQL.

### **ASP**

Su nombre es Active Server Pages. Es un lenguaje independiente, diseñado por Microsoft para la codificación eficiente de los scripts de los servidores, que fueron diseñados para ser ejecutados por un servidor Web en respuesta a la petición de un URL de un usuario. Los scripts de ASP son similares a otros scripts de servidores con los que puedes estar familiarizado, que son utilizados en otras plataformas, como Perl, Python, etc.

## EVOLUCION DE LOS LENGUAJES DE PROGRAMACIÓN

periodo	Influencias	Lenguajes
1950 - 55	Ordenadores primitivos	Lenguajes ensamblador
		Lenguajes experimentales
		de alto nivel
1956 - 60	Ordenadores pequeños,	FORTRAN
	caros y lentos	ALGOL 58 y 60
	Cintas magnéticas	COBOL
	Compiladores e interpretes	LISP
	Optimización del código	
1961 - 65	Ord. grandes y caros	FORTRAN IV
	Discos Magnéticos	COBOL 61 Extendido
	Sistemas operativos	ALGOL 60 Revisado
	Leng. de propósito general	SNOBOL
		APL ( como notación sólo)
1966 - 70	Ordenadores de diferentes	PL/I
	tamaños, velocidades, costes	FORTRAN 66 (standard)
	Sistemas de almacenamiento	COBOL 65 (standard)
	masivo de datos (caros)	ALGOL 68
	S.O. multitarea e	SNOBOL4
	interactivos	SIMULA 67

	Compil. con optimización	BASIC
	Leng. estandar ,	APL/360
	flexibles y generales	
1971 - 75	Micro ordenadores	
	Sistemas de almacenamiento	PASCAL
	masivo de datos pequeños	COBOL 74
	y baratos	PL /I
	Progr. estructurada	
	Ingeniería del software	
	Leng. sencillos	
1976 - 80	Ord. baratos y potentes	ADA
	Sistemas distribuidos	FORTRAN 77
	Prog. tiempo-real	PROLOG
	Prog. interactiva	C
	Abstracción de datos	
	Prog. con fiabilidad	
	y fácil mantenimiento	

Todo este desarrollo de las computadoras y de los lenguajes de programación, suele divisarse por generaciones y el criterio que se determinó para determinar el cambio de generación no está muy bien definido, pero resulta aparente que deben cumplirse al menos los siguientes requisitos:

- La forma en que están construidas.
- Forma en que el ser humano se comunica con ellas.

### **Primera Generación**

En esta generación había un gran desconocimiento de las capacidades de las computadoras, puesto que se realizó un estudio en esta época que determinó que con veinte computadoras se saturaría el mercado de los Estados Unidos en el campo de procesamiento de datos.

Esta generación abarco la década de los cincuenta. Y se conoce como la primera generación. Estas máquinas tenían las siguientes características:

- Estas máquinas estaban construidas por medio de tubos de vacío.
- Eran programadas en lenguaje de máquina.

1

En esta generación las máquinas son grandes y costosas (de un costo aproximado de ciento de miles de dólares). En 1951 aparece la UNIVAC (NIVersAl Computer), fue la primera computadora comercial, que disponía de mil palabras de memoria central y

podían leer cintas magnéticas, se utilizó para procesar el censo de 1950 en los Estados Unidos.

En las dos primeras generaciones, las unidades de entrada utilizaban tarjetas perforadas, retomadas por Herman Hollerith (1860 - 1929), quien además fundó una compañía que con el paso del tiempo se conocería como IBM (International Business Machines).

Después se desarrolló por IBM la **IBM 701** de la cual se entregaron 18 unidades entre 1953 y 1957.

Posteriormente, la compañía Remington Rand fabricó el modelo 1103, que competía con la 701 en el campo científico, por lo que la IBM desarrolló la 702, la cual presentó problemas en memoria, debido a esto no duró en el mercado.

La computadora más exitosa de la primera generación fue la IBM 650, de la cual se produjeron varios cientos. Esta computadora que usaba un esquema de memoria secundaria llamado tambor magnético, que es el antecesor de los discos actuales.

Otros modelos de computadora que se pueden situar en los inicios de la segunda generación son: la UNIVAC 80 y 90, las IBM 704 y 709, Burroughs 220 y UNIVAC 1105.

### **Segunda Generación**

Cerca de la década de 1960, las computadoras seguían evolucionando, se reducía su tamaño y crecía su capacidad de procesamiento. También en esta época se empezó a definir la forma de comunicarse con las computadoras, que recibía el nombre de programación de sistemas.

Las características de la segunda generación son las siguientes:

- Están construidas con circuitos de transistores.
- Se programan en nuevos lenguajes llamados lenguajes de alto nivel.

En esta generación las computadoras se reducen de tamaño y son de menor costo. Aparecen muchas compañías y las computadoras eran bastante avanzadas para su época como la serie 5000 de Burroughs y la ATLAS de la Universidad de Manchester.

Algunas de estas computadoras se programaban con cintas perforadas y otras más por medio de cableado en un tablero. Los programas eran hechos a la medida por un equipo de expertos: analistas, diseñadores, programadores y operadores que se manejaban como una orquesta para resolver los problemas y cálculos solicitados por la administración. El usuario final de la información no tenía contacto directo con las computadoras. Esta situación en un principio se produjo en las primeras computadoras personales, pues se requería saberlas "programar" (alimentarle instrucciones) para obtener resultados; por lo tanto su uso estaba limitado a aquellos audaces pioneros que gustaran de pasar un buen número de horas escribiendo instrucciones, "corriendo" el programa resultante y verificando y corrigiendo los errores o bugs que aparecieran. Además, para no perder el "programa" resultante había que "guardarlo" (almacenarlo) en una grabadora de cassette, pues en esa época no había discos flexibles y mucho menos discos duros para las PC; este procedimiento podía tomar de 10 a 45 minutos, según el programa. El panorama se modificó totalmente con la aparición de las computadoras personales con mejores circuitos, más memoria, unidades de disco flexible y sobre todo con la aparición de programas de aplicación general en donde el usuario compra el programa y se pone a trabajar. Aparecen los programas procesadores de palabras como el célebre Word Star, la impresionante hoja de cálculo (spreadsheet) Visicalc y otros más que de la noche a la mañana cambian la imagen de la PC. El software empieza a

tratar de alcanzar el paso del hardware. Pero aquí aparece un nuevo elemento: el usuario.

El usuario de las computadoras va cambiando y evolucionando con el tiempo. De estar totalmente desconectado a ellas en las máquinas grandes pasa la PC a ser pieza clave en el diseño tanto del hardware como del software. Aparece el concepto de human interface que es la relación entre el usuario y su computadora. Se habla entonces de hardware ergonómico (adaptado a las dimensiones humanas para reducir el cansancio), diseños de pantallas antirreflejos y teclados que descansen la muñeca. Con respecto al software se inicia una verdadera carrera para encontrar la manera en que el usuario pase menos tiempo capacitándose y entrenándose y más tiempo produciendo. Se ponen al alcance programas con menús (listas de opciones) que orientan en todo momento al usuario (con el consiguiente aburrimiento de los usuarios expertos); otros programas ofrecen toda una artillería de teclas de control y teclas de funciones (atajos) para efectuar toda suerte de efectos en el trabajo (con la consiguiente desorientación de los usuarios novatos). Se ofrecen un sinnúmero de cursos prometiendo que en pocas semanas hacen de cualquier persona un experto en los programas comerciales. Pero el problema "constante" es que ninguna solución para el uso de los programas es "constante". Cada nuevo programa requiere aprender nuevos controles, nuevos trucos, nuevos menús. Se empieza a sentir que la relación usuario-PC no está acorde con los desarrollos del equipo y de la potencia de los programas. Hace falta una relación amistosa entre el usuario y la PC.

Las computadoras de esta generación fueron: la Philco 212 (esta compañía se retiró del mercado en 1964) y la UNIVAC M460, la Control Data Corporation modelo 1604, seguida por la serie 3000, la IBM mejoró la 709 y sacó al mercado la 7090, la National Cash Register empezó a producir máquinas para proceso de datos de tipo comercial, introdujo el modelo NCR 315.

La Radio Corporation of America introdujo el modelo 501, que manejaba el lenguaje COBOL, para procesos administrativos y comerciales. Después salió al mercado la RCA 601.

### **Tercera generación**

Con los progresos de la electrónica y los avances de comunicación con las computadoras en la década de los 1960, surge la **tercera generación** de las computadoras. Se inaugura con la IBM 360 en abril de 1964.<sup>3</sup>

Las características de esta generación fueron las siguientes:

- Su fabricación electrónica esta basada en circuitos integrados.
- Su manejo es por medio de los lenguajes de control de los sistemas operativos.

La IBM produce la serie 360 con los modelos 20, 22, 30, 40, 50, 65, 67, 75, 85, 90, 195 que utilizaban técnicas especiales del procesador, unidades de cinta de nueve canales, paquetes de discos magnéticos y otras características que ahora son estándares (no todos los modelos usaban estas técnicas, sino que estaba dividido por aplicaciones).

El sistema operativo de la serie 360, se llamó OS que contaba con varias configuraciones, incluía un conjunto de técnicas de manejo de memoria y del procesador que pronto se convirtieron en estándares.

2

En 1964 CDC introdujo la serie 6000 con la computadora 6600 que se consideró durante algunos años como la más rápida.

En la década de 1970, la IBM produce la serie 370 (modelos 115, 125, 135, 145, 158, 168). UNIVAC compite con los modelos 1108 y 1110, máquinas en gran escala; mientras que CDC produce su serie 7000 con el modelo 7600. Estas computadoras se caracterizan por ser muy potentes y veloces.

A finales de esta década la IBM de su serie 370 produce los modelos 3031, 3033, 4341. Burroughs con su serie 6000 produce los modelos 6500 y 6700 de avanzado diseño, que se reemplazaron por su serie 7000. Honey - Well participa con su computadora DPS con varios modelos.

A mediados de la década de 1970, aparecen en el mercado las computadoras de tamaño mediano, o *minicomputadoras* que no son tan costosas como las grandes (llamadas también como *mainframes* que significa también, gran sistema), pero disponen de gran capacidad de procesamiento. Algunas minicomputadoras fueron las siguientes: la PDP - 8 y la PDP - 11 de Digital Equipment Corporation, la VAX (Virtual Address eXtended) de la misma compañía, los modelos NOVA y ECLIPSE de Data General, la serie 3000 y 9000 de Hewlett - Packard con varios modelos el 36 y el 34,

la Wang y Honey - Well -Bull, Siemens de origen alemán, la ICL fabricada en Inglaterra. En la Unión Soviética se utilizó la US (Sistema Unificado, Ryad) que ha pasado por varias generaciones.

#### **Cuarta Generación**

Aquí aparecen los *microprocesadores* que es un gran adelanto de la microelectrónica, son circuitos integrados de alta densidad y con una velocidad impresionante. Las microcomputadoras con base en estos circuitos son extremadamente pequeñas y baratas, por lo que su uso se extiende al mercado industrial. Aquí nacen las computadoras personales que han adquirido proporciones enormes y que han influido en la sociedad en general sobre la llamada "*revolución informática*".

En 1976 Steve Wozniak y Steve Jobs inventan la primera microcomputadora de uso masivo y más tarde forman la compañía conocida como la Apple que fue la segunda compañía más grande del mundo, antecedida tan solo por IBM; y esta por su parte es aún de las cinco compañías más grandes del mundo.

En 1981 se vendieron 800 000 computadoras personales, al siguiente subió a 1 400 000. Entre 1984 y 1987 se vendieron alrededor de 60 millones de computadoras personales, por lo que no queda duda que su impacto y penetración han sido enormes.

Con el surgimiento de las computadoras personales, el software y los sistemas que con ellas se manejan han tenido un considerable avance, porque han hecho más interactiva la comunicación con el usuario. Surgen otras aplicaciones como los procesadores de palabra, las hojas electrónicas de cálculo, paquetes gráficos, etc. También las industrias del Software de las computadoras personales crecen con gran rapidez, Gary Kildall y William Gates se dedicaron durante años a la creación de sistemas operativos y métodos para lograr una utilización sencilla de las microcomputadoras (son los creadores de CP/M y de los productos de Microsoft).

No todo son microcomputadoras, por su puesto, las minicomputadoras y los grandes sistemas continúan en desarrollo. De hecho las máquinas pequeñas rebasaban por mucho la capacidad de los grandes sistemas de 10 o 15 años antes, que requerían de instalaciones costosas y especiales, pero sería equivocado suponer que las grandes computadoras han desaparecido; por el contrario, su presencia era ya ineludible en prácticamente todas las esferas de control gubernamental, militar y de la gran industria.

Las enormes computadoras de las series CDC, CRAY, Hitachi o IBM por ejemplo, eran capaces de atender a varios cientos de millones de operaciones por segundo.

### **Quinta Generación**

En vista de la acelerada marcha de la microelectrónica, la sociedad industrial se ha dado a la tarea de poner también a esa altura el desarrollo del software y los sistemas con que se manejan las computadoras. Surge la competencia internacional por el dominio del mercado de la computación, en la que se perfilan dos líderes que, sin embargo, no han podido alcanzar el nivel que se desea: la capacidad de comunicarse con la computadora en un lenguaje más cotidiano y no a través de códigos o lenguajes de control especializados.

Japón lanzó en 1983 el llamado "programa de la quinta generación de computadoras", con los objetivos explícitos de producir máquinas con innovaciones reales en los criterios mencionados. Y en los Estados Unidos ya está en actividad un programa en desarrollo que persigue objetivos semejantes, que pueden resumirse de la siguiente manera:

- Procesamiento en paralelo mediante arquitecturas y diseños especiales y circuitos de gran velocidad.
- Manejo de lenguaje natural y sistemas de inteligencia artificial.

El futuro previsible de la computación es muy interesante, y se puede esperar que esta ciencia siga siendo objeto de atención prioritaria de gobiernos y de la sociedad en conjunto.

## **BIBLIGRAFIA**

Los documentos empleados para la realización de este trabajo han sido obtenidos de:

- [WWW.ELRICONDELVAGO.ES](http://WWW.ELRICONDELVAGO.ES)
- [WWW.GEOCITIES.COM](http://WWW.GEOCITIES.COM)
- [WWW.LYCOS.ES](http://WWW.LYCOS.ES)
- [WWW.LAWEBDELPROGRAMADOR.ES](http://WWW.LAWEBDELPROGRAMADOR.ES)
- [WWW.GOOGLE.COM](http://WWW.GOOGLE.COM)
- [WWW.PROGRAMANDO.COM](http://WWW.PROGRAMANDO.COM)
- [WWW.TERRA.ES](http://WWW.TERRA.ES)
- [WWW.YA.COM](http://WWW.YA.COM)
- INFORMATICA BASICA
- CURSO DE PROGRAMACIÓN COBOL