INTRODUCCIÓN A LA PROGRAMACIÓN

ANEXO 1

Lenguajes de programación

HISTORIA Y EVOLUCIÓN DE LOS LENGUAJES DE PROGRAMACIÓN

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 sé hacia 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, se redactan 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 órdenes 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, órdenes y símbolos reconocibles por autómata, a través de su unidad de programación, que le permite ejecutar la secuencia de control deseada. Al conjunto de total de estas instrucciones, órdenes y

símbolos que están disponibles, se le llama 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, calculo/manipulación, de textos, lógica/comparación, y almacenamiento/recuperación.

HISTORIA

Los primeros lenguajes de programación surgieron de la idea de Charles Babagge, 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 al principio del siglo XIX predijo muchas de las teorías en que se basan los actuales ordenadores.

Consistía en lo que él denominaba la máquina analítica, que por motivos técnicos no pudo construirse sino hasta mediados del siglo XX. Con él colaboró Ada Lovelace, quien es considerada como la primera programadora de la historia, pues realizó programas para la máquina de Babagge, empleando tarjetas perforadas. Como la máquina no llegó a construirse, los programas de Ada, lógicamente, tampoco llegaron a ejecutarse, pero si 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 Babagge, y Ada, que consistían (entre otras), en la programación mediante tarjetas perforadas. Se dice 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. No obstante, Babagge se dedicó al proyecto de la máquina analítica, abandonando la máquina de diferencias, basándose en la programación con tarjetas perforadas a través de 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 Babagge intentó crear una máquina que programara, con tarjetas perforadas, y efectuase cualquier cálculo con una precisión de 20 dígitos. Pero la tecnología de la época no bastaba para hacer realidad sus ideas. Si bien estas ideas no llegaron a materializarse de forma definitiva, su contribución fue 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, pero sus ideas no fueron abandonadas, siendo la base para la construcción de los primeros computadores.

Máquina analítica

Máquina diferencial

Cuando surgió el primer ordenador, el famoso ENIAC (Electronic Numerical Integrator And Calculator), su programación se basaba en componentes físicos, es decir, que se programaba cambiando directamente el hardware de la máquina. Exactamente lo que sé hacia era cambiar cables de sitio para conseguir la programación de la máquina. La entrada y salida de datos se realizaba mediante tarjetas perforadas.

LAS TENDENCIAS DE LOS LENGUAJES DE PROGRAMACIÓN

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

1. 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 órdenes, dominados por sentencias que expresan "lo que hay que hacer" en vez de "cómo hacerlo". Ejemplos de esto tenemos a 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 rápidamente el lenguaje y usarlo en su trabajo, sin necesidad de programadores o prácticas de programación.

2. Los **lenguajes de alto nivel** son los más utilizados como lenguaje de programación. Aunque no son fundamentalmente declarativos, éstos permiten que los algoritmos se expresen en un nivel y estilo de escritura, de fácil lectura y comprensión por otros programadores. Además, los lenguajes de alto nivel tienen normalmente las características de "transportabilidad". Es decir, están implementados 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 llaman "independientes de la máquina". Ejemplos de estos lenguajes tenemos están 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).

3. 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 y su lenguaje ensamblador asociado. El lenguaje ensamblador es, simplemente, una representación simbólica del lenguaje de 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 de lenguajes.

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

Como muestra este ejemplo, a más bajo nivel de lenguaje, más cerca 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 es que los lenguajes ensambladores 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 de un lenguaje ensamblador, requiere de 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 "intérprete". El ensamblador y el compilador traducen el programa a otro equivalente en el lenguaje X de la máquina "residente" como un paso separado antes de la ejecución. Por otra parte, el intérprete 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 máquina. Esto se debe principalmente a que las sentencias dentro de un "bucle" deben ser reinterpretadas cada vez que se ejecutan por un intérprete. Con un compilador, cada sentencia es interpretada y luego traducida a lenguaje máquina solo una vez.

Algunos lenguajes son interpretados, como APL, PROLOG y LISP. El resto de los lenguajes – Pascal, FORTRAN, COBOL, PL/I, SNOBOL, C, Ada y Modula−2 – son lenguajes compilados. En algunos casos, un compilador estará utilizable alternativamente para un lenguaje interpretado (tal como LISP) e inversamente (tal como el intérprete 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 máquinas cuyos lenguajes son ellos mismos lenguajes de alto nivel. Como ejemplo de estos están las nuevas máquinas LISP, que han sido diseñadas recientemente por Symbolics y Xerox Corporations.

Los lenguajes de programación son tomados de diferentes perspectivas. Es importante para un programador decidir cuáles conceptos emitir o cuáles 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:

- Orientada a Objetos.
- Imperativa: entrada, procesamiento y salidas de Datos.
- Funcional: (funciones) los datos son funciones, los resultados pueden ser un valor o una función.
- Lógica: {T, F} + operaciones lógicas (Inteligencia Artificial).
- Concurrente: aún está 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 qué presentan características diferentes. Por ejemplo, los lenguajes "más 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 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 ejercen 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 diseño de lenguajes y la mayoría de las aplicaciones con las que se enfrenta el programador.

Existen varios lenguajes que 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.

Finalmente, la notación algebraica ordinaria influyó fuertemente en el diseño de FORTRAN y ALGOL. Por otra parte, las construcciones de largas sentencias en el idioma inglés influyeron 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. Estas máquinas 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 está entre los objetivos de este texto, pero es importante mencionarlo aquí debido a su fundamental influencia en la evolución de los primeros lenguajes de programación, y por su estado en el núcleo de la teoría de la computadora. Adicional a este punto cabe mencionar lo siguiente: 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.

EVOLUCIÓN DE LOS LENGUAJES DE PROGRAMACIÓN

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, tenemos lo siguiente:

- ✓ 1ª generación. Lenguajes máquina y ensambladores.
- ✓ 2ª generación. Primeros lenguajes de alto nivel imperativo (FROTRAN, COBOL).
- ✓ 3ª generación. Lenguajes de alto nivel imperativo. Son los más utilizados y siguen vigentes en la actualidad (ALGOL 8, PL/I, PASCAL, MODULA)
- ✓ 4ª generación. Orientados básicamente a las aplicaciones de gestión y al manejo de bases de datos (NATURAL, SQL).
- ✓ 5ª generación. Orientados a la inteligencia artificial y al procesamiento de los lenguajes naturales (LISP, PROLOG).

Para una mejor compresión del tema, se definirán los siguientes términos:

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

Código fuente: está 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.

Evolución de los lenguajes de programación

PERIODO	INFLUENCIAS	LENGUAJES
1950 – 55	Ordenadores primitivos	Lenguajes ensamblador Lenguajes experimentales de alto nivel
1956 – 60	Ordenadores pequeños caros y lentos Cintas magnéticas Compiladores e interpretes Optimización del código	FORTRAN ALGOL 58 y 60 COBOL LISP
1961 – 65	Ord. grandes y caros Discos Magnéticos Sistemas operativos Leng. de propósito general	FORTRAN IV COBOL 61 Extendido ALGOL 60 Revisado SNOBOL APL (como notación sólo)
1966 – 70	Ordenadores de diferentes tamaños, velocidades, costes Sistemas de almacenamiento masivo de datos (caros) S.O. multitarea e interactivo Compil. con optimización Leng. estandard flexibles y generales	PL/I FORTRAN 66 (estándar) COBOL 65 (estándar) ALGOL 68 SNOBOL4 SIMULA 67 BASIC APL/360
1971 – 75	Micro ordenadores	

	Sistemas de almacenamiento masivo de datos pequeños y baratos Progr. Estructurada Ingeniería del software Leng. sencillos	PASCAL COBOL 74 PL /I
1976 – 80	Comp. baratas y potentes Sistemas distribuidos Prog. tiempo–real Prog. interactiva Abstracción de datos Prog. con fiabilidad y fácil mantenimiento	ADA FORTRAN 77 PROLOG C

Todo este desarrollo de las computadoras y de los lenguajes de programación, suele divisarse por generaciones y el criterio que se determinó para indicar el cambio de generación no está muy bien definido (debido a que no hubo un cambio significativo en diseño estructural del software sino más bien fue paulatino), 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.