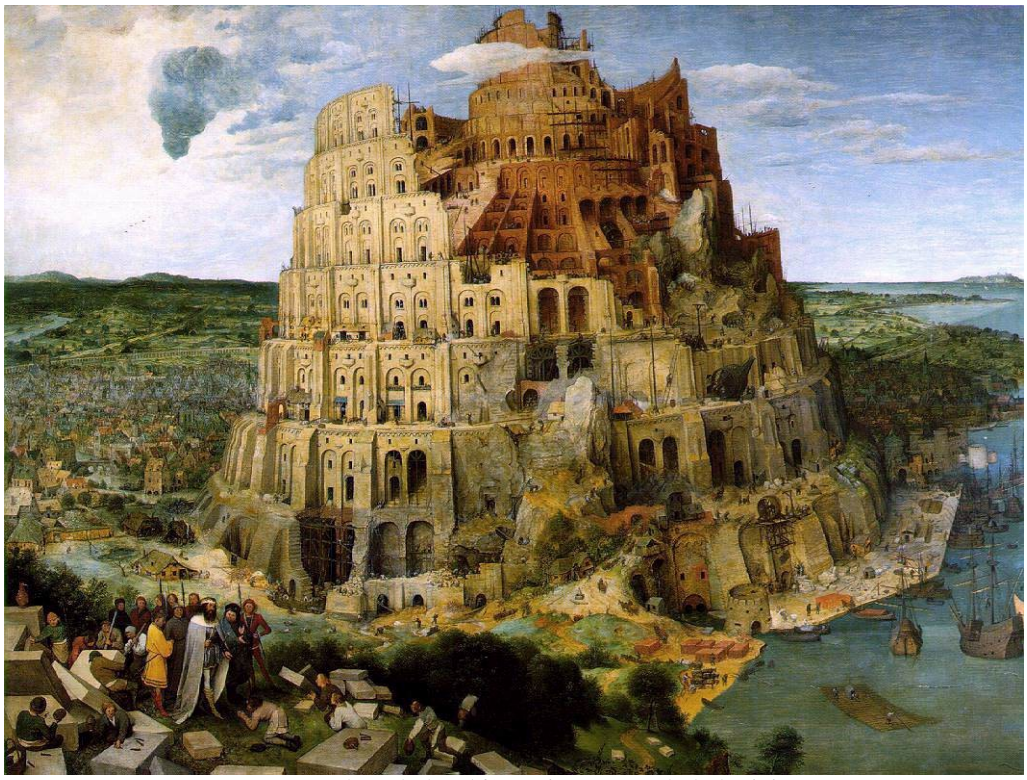
Lenguajes de programación

Última modificación 2008/10



La Torre de Babel (1563) - Pieter Brueghel el Viejo

Realizado a partir de información de la Wikipedia y trabajo propio.

Lenguajes de programación

Contenido

INTRODUCCIÓN....3
DEFINICIÓN....4
CLASIFICACIÓN....4
- SEGÚN EL NIVEL DE ABSTRACCIÓN....4
  - Lenguajes de máquina y de bajo nivel....4
  - Lenguajes de medio nivel....4
  - Lenguajes de alto nivel y de muy alto nivel....4
- SEGÚN LA FORMA DE EJECUCIÓN....5
  - Diferencias entres lenguajes compilados e interpretados....5
- SEGÚN EL PARADIGMA DE PROGRAMACIÓN....6

INTRODUCCIÓN

Con el objeto de facilitar la interacción de las personas con los computadores, los sistemas operativos hacen una aparición discreta y bastante simple a principios de 1950, con conceptos tales como el monitor residente, el proceso por lotes y el almacenamiento temporal.

En las primeras máquinas, como la ENIAC (*Electronic Numerical Integrator And Calculator*) que se terminó de construir a finales de 1945, la programación se hacía manualmente conectando cables y pulsando interruptores. Los datos se suministraban en tarjetas perforadas. La programación para los cálculos normales requería de media hora a un día entero.

Pocos años después la programación se hace en base a instrucciones, que son secuencias de unos y ceros que representan si una llave debe estar activa o inactiva (1948: *Small Scale Experimental Machine* o "*The Baby*"). A esta forma de programar se le llamó "lenguaje de máquina".

A comienzos de 1950 se desarrollaron los primeros lenguajes simbólicos nacidos de la necesidad de recordar secuencias de programación para las acciones usuales. A estas acciones se les denominó con nombres fáciles de memorizar y asociar: ADD (sumar), SUB (restar), MUL (multiplicar), CALL (ejecutar subrutina), etc. A este conjunto de instrucciones se le llamó "lenguaje ensamblador".

A finales de los años cincuenta y comienzos de los sesenta se desarrollaron los primeros lenguajes de alto nivel con su propio vocabulario más o menos limitado, su gramática más o menos estricta y su semántica, que se asimilan relativamente al lenguaje humano. Estos lenguajes se denominan de "alto nivel" porque para poder ser utilizados deben pasarse por un traductor que los convierta a un lenguaje de nivel inferior (código máquina o ensamblador). A este proceso se le llama "compilación" o "interpretación" y lo realiza, curiosamente, otro programa[1]. El código escrito por el programador es lo que se conoce como "código fuente" y el código traducido es lo que se conoce como "código binario".

El primero fue FORTRAN (*FORmula TRANslator*) en 1954, y su creación se debe a John Backus. Otros lenguajes de alto nivel son COBOL (*Common Business-Oriented Language*) desarrollado en 1960 por Grace Hopper y otros; Pascal, desarrollado por Niklaus Wirth en 1970 y C desarrollado por Ken Thompson y Dennis Ritchie con el objeto de crear un nuevo sistema operativo (UNIX).

Basándose en los trabajos publicados por Robinson en 1965, en 1972, Kowalski publica las primeras ideas acerca de cómo la lógica de primer orden podría ser usada como un lenguaje de programación. Poco después Colmerauer lleva a la practica estas ideas con la implementación del lenguaje PROLOG (PROgramming in LOGic), el primer y más difundido lenguaje que utiliza un nuevo paradigma de programación, la programación lógica.

Los conceptos de la programación orientada a objetos tienen origen en "Simula 67", un lenguaje diseñado para hacer simulaciones, creado por Ole-Johan Dahl y Kristen Nygaard del Centro de Cómputo Noruego en Oslo. Estos principios fueron refinados más tarde en "Smalltalk", diseñado para ser un sistema completamente dinámico en el cual los objetos se podrían crear y modificar "sobre la marcha" en lugar de tener un sistema basado en programas estáticos.

La programación orientada a objetos se fue convirtiendo en dominante a mediados de los años ochenta, en gran parte debido a la influencia de C++, una extensión del lenguaje de programación C. Su dominación fue consolidada gracias al auge de las Interfaces Gráficas de Usuario, para las cuales la programación orientada a objetos está particularmente bien adaptada. En este caso, se habla también de programación dirigida por eventos.
Las características de orientación a objetos son utilizadas por lenguajes específicamente diseñados, pero también han sido agregadas a muchos lenguajes existentes anteriormente.

1 Para una información más específica ver más adelante la sección sobre lenguajes compilados y lenguajes interpretados.

DEFINICIÓN

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones, y es utilizado para controlar el comportamiento físico y lógico de una máquina.

Aunque muchas veces se usan los términos 'lenguaje de programación' y 'lenguaje informático' como si fuesen sinónimos, no es del todo correcto, ya que los lenguajes informáticos engloban a los lenguajes de programación y a otros más, como por ejemplo HTML que es un lenguaje para el marcado de páginas web.

Un lenguaje de programación permite especificar de manera precisa sobre qué datos debe operar una computadora, cómo estos datos deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural, tal como sucede con el lenguaje Léxico.

CLASIFICACIÓN

Los lenguajes de programación se pueden clasificar atendiendo a varios criterios, los principales son:

- Según el nivel de abstracción
- Según la forma de ejecución
- Según el paradigma de programación que poseen cada uno de ellos

SEGÚN EL NIVEL DE ABSTRACCIÓN

Lenguajes de máquina y de bajo nivel

Los lenguajes de máquina están escritos en códigos (código máquina) directamente inteligibles por la máquina (computadora), siendo sus instrucciones cadenas binarias (0 y 1).
“Lenguaje de máquina” hace referencia al lenguaje específico de una computadora, mientras que “código máquina” hace referencia al modo en que se escriben los diferentes lenguajes de máquina.

Los lenguajes de bajo nivel son lenguajes de programación que se acercan al funcionamiento de una computadora. Los lenguajes de más bajo nivel son los lenguajes de máquinas. A éste nivel le sigue el lenguaje ensamblador, ya que al programar en ensamblador se trabajan con los registros de memoria de la computadora de forma directa.

La programación en un lenguaje de bajo nivel tiene como ventajas una mayor adaptación al equipo, además de la posibilidad de obtener la máxima velocidad con el mínimo uso de memoria.
Sin embargo tiene importantes inconvenientes, como la imposibilidad de escribir código independiente de la máquina y la mayor dificultad en la programación y en la comprensión de los programas.

Lenguajes de medio nivel

Minoritariamente en algunos textos se diferencian algunos lenguajes como de medio nivel, como el lenguaje C, ya que tienen ciertas características que los acercan a los lenguajes de bajo nivel, como gestión de punteros de memoria y registros, pero con sintaxis, vocabulario y gramática de alto nivel.

Lenguajes de alto nivel y de muy alto nivel

Los lenguajes de programación de alto nivel se caracterizan por expresar los algoritmos de una manera adecuada a la capacidad cognitiva humana, en lugar de estar orientados a su ejecución en las máquinas.
Los lenguajes de alto y bajo nivel requieren de conocimientos específicos de programación y del lenguaje concreto (vocabulario, gramática y sintaxis) para realizar las secuencias de instrucciones lógicas.
Los lenguajes de muy alto nivel se crearon para que el usuario común pudiese solucionar ciertos problemas sencillos de procesamiento de datos de una manera más fácil y rápida.

SEGÚN LA FORMA DE EJECUCIÓN

Los procesadores usados en las computadoras son capaces de entender y actuar según lo indican programas escritos en un lenguaje fijo para cada arquitectura, llamado lenguaje de máquina. Todo programa escrito en un lenguaje de alto nivel puede ser ejecutado de dos maneras:

- **Lenguajes compilados**: Antes de poder utilizarse el programa debe utilizarse un traductor llamado "compilador" que se encarga de traducir ("compilar") el programa original ("código fuente") al programa equivalente escrito en lenguaje de máquina o ensamblador ("binario"). Los binarios son los programas ejecutables y los únicos necesarios para el funcionamiento del programa.
- **Lenguajes interpretados**: Cada vez que se usa el programa debe utilizarse un traductor llamado "intérprete" que se encarga de traducir ("interpretar") las instrucciones del programa original ("código fuente") a código máquina según van siendo utilizadas. Para el funcionamiento del programa siempre es necesario disponer del código original y del intérprete.

Diferencias entres lenguajes compilados e interpretados

- Los lenguajes compilados se compilan una vez y se utilizan cuantas veces se desee sin necesidad de volver a utilizar el compilador. Los lenguajes interpretados son interpretados, valga la redundancia, cada vez que se ejecutan y necesitan siempre del intérprete.
- Los compiladores analizan todo el programa y no generan resultados si no es correcto todo el código. Los intérpretes analizan las instrucciones según las necesitan y pueden iniciar la ejecución de un programa con errores e incluso terminar correctamente una ejecución de un programa con errores siempre que no haya sido necesario el uso de las instrucciones que contienen dichos errores.
- Un compilador traduce cada instrucción una sola vez. Un intérprete debe traducir una instrucción cada vez que la encuentra.
- Los binarios son compilados para una arquitectura específica y no pueden ser utilizados en otras arquitecturas no compatibles (aunque pueden existir distintos compiladores para generar binarios para diferentes arquitecturas). Un lenguaje interpretado puede ser utilizado en cualquier arquitectura que disponga de un intérprete sin necesidad de cambios.
- Los lenguajes compilados son más eficientes que los interpretados y además permiten distribuir el programa en forma confidencial mediante binarios.
- Es más sencillo empaquetar lenguajes interpretados dentro de otros lenguajes, como JavaScript dentro de HTML.

Para obtener las ventajas de ambos tipos de lenguajes algunos utilizan una aproximación en dos fases. Primero el programa original (código fuente) es precompilado a un binario confidencial, portable e interpretable. En una segunda fase el binario precompilado es interpretado en cada arquitectura. Ésta aproximación es la que realiza por ejemplo Java.

Hay que hacer notar que algunas aplicaciones permiten ser programadas con lenguajes. Estos lenguajes no tienen por objeto solicitar acciones a la computadora sino solicitar acciones a la aplicación sobre la que se ejecutan. Por tanto aunque algunos de estos lenguajes son lenguajes de programación, no son lenguajes de programación de computadoras y por tanto no necesitan ser traducidos a código máquina.
Es el caso por ejemplo de SQL, un lenguaje declarativo de cuarta generación diseñado para trabajar con bases de datos. Este lenguaje SQL es interpretado por el motor de la Base de Datos, no por la CPU.

SEGÚN EL PARADIGMA DE PROGRAMACIÓN

Un paradigma de programación representa un enfoque particular o filosofía para la construcción del software. Si bien puede seleccionarse la forma pura de estos paradigmas a la hora de programar, en la práctica es habitual que se mezclen, dando lugar a la programación multiparadigma.

Los diferentes paradigmas de programación son:

- **Algorítmico, Imperativo o Por procedimientos**. El más común y está representado, por ejemplo, por C o por BASIC.
  Describe la programación en términos del estado del programa y sentencias que cambian dicho estado. Los programas imperativos son un conjunto de instrucciones que le indican al computador cómo realizar una tarea. La implementación de hardware de la mayoría de computadores es imperativa ya que el *hardware* está diseñado para ejecutar código de máquina que es imperativo.
- **Declarativo o Predicativo**. Basado en la utilización de predicados lógicos (lógico) o funciones matemáticas (funcional), su objetivo es conseguir lenguajes expresivos en los que no sea necesario especificar cómo resolver el problema (programación convencional imperativa), sino qué problema se desea resolver. Los interpretes de los lenguajes declarativos tienen incorporado un motor de inferencia genérico que resuelve los problemas a partir de su especificación.
  - **Lógico**. Un ejemplo es PROLOG. El mecanismo de inferencia genérico se basa en los procedimientos de deducción de formulas válidas en un sistema axiomático
  - **Funcional**. Representado por la familia de lenguajes LISP (en particular Scheme), ML o Haskell. El mecanismo de inferencia genérico se basa en la reducción de una expresión funcional a otra equivalente simplificada.
- **Orientado a Objetos**. Cada vez más utilizado, sobre todo en combinación con el imperativo. De hecho los lenguajes orientados a objetos permiten la programación imperativa. Algunos ejemplos de lenguajes orientados a objetos son C++, Java, Python. Usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo y encapsulamiento.