Liceo Compu-Market

Catedratico: Erick Gomes

Catedra: Programacion

Historia de los Lenguajes de Programación. Con la idea de facilitarnos las tareas que debemos de desempeñar los humanos, hemos venido inventado diversas herramientas a lo largo de nuestra historia, que nos permiten tener una mejor calidad de vida. Los ordenadores son uno más de los inventos del hombre, aunque debemos decir que las tecnologías para su fabricación y explotación han tenido un desarrollo sorprendente a partir de la segunda mitad del siglo XX. Esta herramienta por sí sola no es capaz de efectuar ninguna tarea, es tan sólo un conjunto de cables y circuitos que necesitan recibir instrucción por parte de los humanos para desempeñar alguna tarea. El problema entonces, se puede fijar en ¿cómo vamos a poder hacer que un conjunto de circuitos desempeñen una determinada tarea y nos entreguen los resultados que nosotros esperamos?, es decir, ¿de qué manera se puede lograr la comunicación entre el hombre y el ordenador?. Así pues, tratando de dar una solución al problema planteado, surgieron los lenguajes de programación, que son como un lenguaje cualquiera, pero simplificado y con ciertas normas, para poder trasmitir nuestros deseos al ordenador. Por otro lado, como se sabe, un conjunto de circuitos no entendería ningún lenguaje que nosotros conozcamos, por más sencillo que éste parezca. Los circuitos en todo caso, sólo reconocen presencia o ausencia de energía, es decir que debemos hablarle a la máquina en su propio lenguaje (presencia y ausencia de energía, 0 y 1), o nuestro lenguaje deberá de ser traducido a un lenguaje binario cuyo alfabeto es el 0 y el 1, mediante las herramientas desarrolladas para llevar a cabo esta tarea, las cuales reciben el nombre de traductores, y como veremos más adelante, los hay de muchos tipos, dependiendo de características más específicas del lenguaje a traducir y de la manera de llevar a cabo su traducción. Como ya habréis entendido, para crear un lenguaje de programación, deberemos crear la herramienta que lo traduce, y es justamente de ellas, de las que hablaremos a continuación, para describir como han ido evolucionando en los últimos 50 años [BYTE 95]. • 1946: Konrad Zuse , un ingeniero Alemán mientras trabajaba en los Alpes de Bavaria, desarrolló el lenguaje Plankalkul, el cual, fue aplicado entre otras cosas para jugar al ajedrez. • 1949: Aparece Short Code, que viene a ser el primer lenguaje que fue usado en un dispositivo de cómputo electrónico, aunque se debe decir que se trata de un lenguaje traducido a mano. • 1951: Grace Hopper , trabajando para Remington Rand, comenzó el trabajo de diseño del primer compilador conocido ampliamente, el A-0, el cual, al ser liberado por la compañía en 1957, lo hizo con el nombre de MATH-MATIC. • 1952: Alick E. Glennie, durante su tiempo libre en la Universidad de Manchester, concibe un sistema de programación llamado AUTOCODE, que viene a ser un compilador muy rudimentario. • 1957: aparece FORTRAN (FORmula TRANslating) sistema traductor de fórmulas matemáticas. Fue desarrollado por un equipo, al frente del cual se encontraba John Backus quien después vendría a contribuir en el desarrollo del compilador para el lenguaje ALGOL y de la notación usada para la especificación sintáctica de los lenguajes, conocida como BNF (Backus Naur Form). A partir de los años sesenta, empiezan a surgir diferentes lenguajes de programación, atendiendo a diversos enfoques, características y propósitos, que más adelante describiremos. Por lo pronto, puede decirse, que actualmente existen alrededor de 2000 lenguajes de programación [KINNERSLEY 95] y continuamente, están apareciendo otros más nuevos, que prometen hacer mejor uso de los recursos computacionales y facilitar el trabajo de los programadores. Tratando de resumir un poco, presentaremos los siguientes cuadros evolutivos, donde aparecen los lenguajes que por su uso y comercialización, han resultado ser los más populares a lo largo de este medio siglo. [LABRA 98] [RUS 01]

**Manual**

Nombres : Ian Gutierrez, Ivo Suarez, Carlos Aguilar, Ademy

Grado: 5to Bachillerato

Fecha: 17/08/2017

Lógica en Programacion

La programación lógica, junto con la funcional, forma parte de lo que se conoce como programación declarativa. En los lenguajes tradicionales, la programación consiste en indicar cómo resolver un problema mediante sentencias; en la programación lógica, se trabaja de una forma descriptiva, estableciendo relaciones entre entidades, indicando no cómo, sino qué hacer. La ecuación de Robert Kowalski (Universidad de Edimburgo) establece la idea esencial de la programación lógica: algoritmos = lógica + control. Es decir, un algoritmo se construye especificando conocimiento en un lenguaje formal (lógica de primer orden), y el problema se resuelve mediante un mecanismo de inferencia (control) que actúa sobre aquél.

Prolog: El lenguaje Prolog, principal representante del paradigma, se basa en un subconjunto de la lógica de primer orden (restricción de la forma clausal de la lógica denominada cláusulas de Horn). Philippe Roussel y Alain Colmerauer (Universidad de AixMarseille) lo crearon en 1972, y su base teórica se debe en gran parte a Kowalski.

Estructuras básicas: Prolog cuenta con dos tipos de estructuras: términos y sentencias. Los términos pueden ser constantes, variables o functores:

> Las constantes, representadas por una cadena de caracteres, pueden ser números o cualquier cadena que comience en minúscula.

> Las variables son cadenas que comienzan con una letra mayúscula

> Los functores son identificadores que empiezan con minúscula, seguidos de una lista de parámetros (términos) entre paréntesis, separados por comas. Las sentencias son reglas o cláusulas. Hay hechos, reglas con cabeza y cola, y consultas.

> Un hecho establece una relación entre objetos, y es la forma más sencilla de sentencia. Por ejemplo:

humano (socrates).

ama (juan,maría)

Se establece que Sócrates es humano y que Juan ama a María.

> Una regla permite definir nuevas relaciones a partir de otras ya existentes. Si queremos establecer que todo humano es mortal, en lógica estándar escribiríamos V(x)(humano(x)=>mortal(x)), mientras que en Prolog escribimos:

mortal(X):-humano(X).

Esto se lee: X (variable) es mortal si X es humano. El símbolo :- significa “si” o, si lo leemos de derecha a izquierda, entonces o implica. En esta regla, mortal(X) es la cabeza, y humano(X) es el cuerpo.

> Para entender el concepto de consulta, veamos un ejemplo. En lógica estándar:

V(x)(humano(x)=>mortal(x)) > humano(socrates) > entonces mortal (socrates)

Partiendo de que los humanos son mortales y de que Sócrates es humano, deducimos que Sócrates es mortal. Para realizar esa deducción en Prolog, hay que preguntar si es mortal Sócrates, o quién es mortal. Si del programa lógico (conjunto de hechos y reglas) se deduce que Sócrates es mortal, entonces ésa será la respuesta que obtendremos. Veamos el programa:

mortal(X):-humano(X).

humano(Sócrates).

Para preguntar interactivamente, los ambientes de Prolog tienen un listener, un intérprete de línea de comando cuyo prompt es un signo de pregunta. Se introduce una sentencia (eventualmente con variables), y Prolog intentará demostrarla (usando un algoritmo de inferencia llamado SLD-Resolution, basado en la regla de resolución de Robinson), buscando además constantes que puedan reemplazar las variables de la pregunta. Las preguntas de nuestro ejemplo serían:

?- mortal(Sócrates).

Yes.

?- mortal(X)

X = socrates´

Las segundas líneas son las respuestas de Prolog: primero, afirmando que sí, Sócrates es mortal; después, contestando Sócrates al preguntar quién es mortal. Si agregamos más conocimiento a nuestro programa lógico (por ejemplo, que Platón es mortal), podemos pedir más respuestas. Si luego de obtener Sócrates escribimos un punto y coma (así “pedimos más”), nos responderá Platón. Cuando Prolog no tenga más respuestas, nos dirá que No; esa negativa no significa que lo que preguntemos sea falso, sino que no lo conoce o no puede demostrarlo. Es decir que si preguntamos, por ejemplo, si Arquímedes es mortal, la respuesta será que no, pero porque nuestro programa no cuenta con el conocimiento suficiente. Esto se denomina negación por falla

?- mortal(X)

X = socrates;

X = platon;

No

Este modo interactivo es muy útil para prueba y prototipación, pero dependiendo de la implementación Prolog en particular, es posible realizar invocaciones desde otros ambientes, o ejecutar un programa Prolog que tenga un predicado main (que será el que se intentará probar). Para facilitar el desarrollo de aplicaciones, Prolog cuenta con un conjunto de características que “ensucian” de alguna manera su pureza en lo que hace al paradigma, pero que lo vuelven utilizable. Hablamos, por ejemplo, de escritura por pantalla, pedido de datos al usuario y otros elementos específicos de control del mecanismo de inferencia (el predicado de corte es el más conocido).

**Listas** Prolog manipula listas con una facilidad sintáctica que simplifica la programación: una lista es un par ordenado, donde un elemento es un término (la cabeza), y el otro puede ser un término, una lista o la lista vacía (la cola). Las listas van entre corchetes, y la cabeza se separa de la cola con el símbolo pipe; la lista vacía se indica con dos corchetes separados por un espacio. Haciendo un abuso de notación, también es posible enumerar todos los elementos de la lista separándolos con comas. Por ejemplo, la lista 1, 2, socrates, a, platon se escribe en Prolog de estas dos maneras:

[1|[2|[socrates|[a|[platon|[]]]]]]

[1, 2, socrates, a, platon]

Para comprender un poco mejor el poder de la programación declarativa, comparemos un algoritmo que determina si un elemento se encuentra en una lista en un lenguaje declarativo (por ejemplo, Pascal) y en Prolog. En Pascal:

function member(item:Integer; L:Array

of Integer):Boolean;

var

i:Integer;

begi n

i:=0;

while((i <= length(L)) and (L[i] <> item)) do

begi n

i := i+1;

end;

Result := item = L[i];

end;

Para saber si un elemento es parte de una lista, en Prolog sólo declaramos que es su cabeza o es parte de su cola.

member(X,[X|Xs])

. member(X,[Y|Ys]) :- member(X,Ys).

El predicado member (parte del estándar de Prolog) dice que un elemento (representado por la variable X) es miembro de una lista si es la cabeza de ella (la cabeza de la lista [X|Xs] es X), y también que un elemento es miembro de una lista si es miembro de la cola (la cola de la lista [Y|Ys] es Ys, que es

Operadores

MATEMATICOS RELACIONALES

> Mayor que

< Menor que

>= Mayor o igual que

=< Menor o igual que

=:= Aritméticamente igual

=\= Aritméticamente diferente

+ Suma

- Resta

\* Multiplicación

/ División (retorna siempre en punto flotante)

// División entera (trunca)

mod Resto de división

\*\* Potenciación

Programación es la acción de programar que implica **ordenar, estructurar o componer una serie de acciones cronológicas para cumplir un objetivo**. La programación puede ser aplicado para eventos sociales, a medios de comunicación y al mundo informático de las computadoras.

En informática, la programación es el uso de lenguajes informáticos para imprimir en un sistema computacional funciones y procesos deseados. La **programación de una computadora** es la forma de indicar a la computadora qué es lo que tiene que hacer.

Un **lenguaje de programación** es, en la ciencia de la computación, la herramienta para automatizar informaciones y acciones a través de una computadora. Los lenguajes de programación más conocidos son: Basic (1964), C++ (1983), Phyton (1991), Java (1995), C# (2000), entre otros.

La programación es una de las etapas para el desarrollo de un programa o software. La programación especifica la estructura y el comportamiento de un programa verificando si está funcionando adecuadamente o no.

La programación incluye la especificación del algoritmo definida como la secuencia de pasos y operaciones que el programa debe realizar para resolver un problema. Para que el algoritmo funcione, el programa debe estar implementado en un lenguaje compatible y correcto.

Qué es Algoritmo:

Como algoritmo denominamos un **conjunto ordenado y finito de operaciones simples a través del cual podemos hallar la solución a un problema**.

La palabra, como tal, proviene del latín tardío *alborarismus*, y este a su vez es una abreviación del árabe clásico *ḥisābu lḡubār*, que significa ‘cálculo mediante cifras arábigas’.

Los algoritmos **nos permiten ejecutar una acción o resolver un problema mediante una serie de instrucciones definidas**, ordenadas y finitas. Así, dado un estado inicial y una entrada, y siguiendo los sucesivos pasos indicados, se llega al estado final y se obtiene una solución.

Aunque es un término habitual en áreas como las matemáticas, la informática, la lógica y demás disciplinas relacionadas, lo cierto es que en la vida cotidiana también usamos algoritmos para solucionar cuestiones.

**Ejemplos de algoritmos**, pues, no solo son programas informáticos, sino también aquel manual que nos explica, paso a paso, cómo armar la biblioteca o activar el celular que compramos., incluso una receta de cocina es un algoritmo.

**En matemáticas**, algunos ejemplos de algoritmos lo constituyen la **multiplicación**, en donde seguimos una secuencia de operaciones para obtener el producto; la **división**, que nos permite determinar el cociente de dos números, así como el **algoritmo de Euclides**, con el cual sacamos el máximo común divisor de dos enteros positivos.

Asimismo, un algoritmo puede trazarse, por ejemplo, en un diagrama de flujo donde se especifique cada una de las tareas a realizar, con sus acciones y sus posibles alternativas, hasta el cumplimiento final de la tarea.

La programación también se refiere a un conjunto de eventos, actividades o programas de televisión agrupados en un día o momento específico. La programación generalmente se encuentra en formato escrito y ordenado cronológicamente.

Por ejemplo, en la programación de un evento esta puede incluir discursos o bandas musicales que actuarán a una hora específica, así como una programación de la televisión informará qué programas estarán en el aire a determinada hora y día.

## Qué es Informática:

Informática es el **tratamiento automático de la información**. Como tal, designa a un conjunto de conocimientos teóricos y prácticos, relativos al ámbito de la [ciencia](https://www.significados.com/ciencia/) y de la [tecnología](https://www.significados.com/tecnologia/), que se combinan para posibilitar el tratamiento racional y automático de la información mediante **sistemas informáticos o computadoras**, cuyas tareas principales son **almacenar, procesar y transmitir la información**.

Según la Real Academia Española de la Lengua, la palabra **informática**pasa al español a través del francés informatique, contracción de las palabras information, que traduce ‘información’, y automatique, ‘automática’, aunque su origen se registra en el alemán informatik.

Como disciplina de estudio, la **ciencia de la** **informática**o**ciencia de la computación**estudia científicamente los límites físicos y teóricos de las computadoras, su procesamiento y el almacenamiento de la información.

La **ingeniería informática**por otro lado aplica la teoría de la ciencia de la computación en métodos, [técnicas](https://www.significados.com/tecnica/), [procesos](https://www.significados.com/proceso/), desarrollo y aplicación de la misma.

La informática nos posibilita el **manejo rápido y eficiente de enormes volúmenes de datos** y es clave en el desarrollo de las tecnologías de la comunicación y el [internet](https://www.significados.com/internet/). Su grado de utilidad es tal que hoy día no hay prácticamente actividad humana o disciplina del conocimiento que no se sirva de ella. La encontramos en el ámbito **empresarial, industrial, comercial, educativo,** así como en la **medicina, los transportes o los videojuegos**. Su potencial, en este sentido, es ilimitado.

En la informática existe el área de **seguridad informática** que protege la infraestructura física y toda la información que se esconde dentro de un sistema informático para que no sea violado. Se vuelve crucial por el inevitable intercambio de datos e información.

Informática también se refiere a lo que es **perteneciente o relativo a la informática**: “Javier es un experto en seguridad **informática**”. Por extensión, también es empleada, tanto en masculino como en femenino, para designar aquella **persona que trabaja en informática**.

La **Olimpiada Internacional de Informática** (IOI) fue creado por la UNESCO y celebrado por primera vez en Bulgaria en el año 1989. Es una de las 5 olimpiadas internacionales de ciencias. Las otras categorías son: física, biología, química y matemática. La próxima IOI se celebrará en Agosto del 2016 en Kazan, Rusia.

En la actualidad, la noción de programación se encuentra muy asociada a la creación de aplicaciones [**informáticas**](http://definicion.de/informatica) y videojuegos; es el proceso por el cual una persona desarrolla un programa valiéndose de una herramienta que le permita escribir el código (el cual puede estar en uno o varios lenguajes, tales como C++, Java y Python) y de otra que sea capaz de “traducirlo” a lo que se conoce como **lenguaje de máquina**, el cual puede ser entendido por un microprocesador.

Este último paso se conoce como **compilación** y es necesario para que el código pueda ser ejecutado por la plataforma para la cual haya sido creado, que puede ser un **[ordenador](http://definicion.de/ordenador)**, una tableta, una consola o un teléfono móvil, por ejemplo. Existe también una forma de traducir el código denominada **interpretación**, que consiste en analizar línea a línea, hasta que se traduzca lo suficiente como para poder realizar una tarea. Cabe mencionar que los lenguajes de programación se dividen en dos grandes grupos, donde los que pueden ser compilados no pueden ser interpretados, y viceversa.

La totalidad del proceso de desarrollo abarca varias etapas y requiere del trabajo de diferentes especialistas. En principio, partiendo de la base de un **[proyecto](http://definicion.de/proyecto)** bien organizado, es necesario dar con una idea atractiva, interesante, que justifique los meses o años de esfuerzo que vendrán. Tan sólo esta primera parte puede tomar mucho tiempo, dado que lo que comienza como un *producto perfecto* puede convertirse, luego de cuestionarlo y observarlo desde diferentes ángulos, en un *fracaso seguro*.

Una vez hallada la idea, se debe establecer el **[diseño](http://definicion.de/diseno)** de la misma; en otras palabras, se trata de formalizar todo aquello que se haya discutido durante la búsqueda inicial. Si bien cada equipo trabaja a su manera, dado que no se puede estructurar la creación de una forma rígida, es probable que el siguiente paso lógico sea comenzar a experimentar a través de la programación, para facilitar a los diseñadores un contacto directo e interactivo con su idea.

Los programadores tienen en sus manos el poder de dar vida a un sistema, a una aplicación, a un videojuego. Es importante notar que todos estos **[productos](http://definicion.de/producto)** pueden ser definidos en absoluto detalle en un documento, incluyendo imágenes y gráficos que expliquen cómo funciona cada milímetro de los mismos; sin embargo, hasta que un desarrollador de software no entra en acción, no es posible verlos en movimiento, probarlos, pasar de la teoría a la práctica.

En un plano más técnico, la programación se realiza mediante el uso de **algoritmos**, que son secuencias finitas, ordenadas y no ambiguas de instrucciones que deben seguirse para resolver un **[problema](http://definicion.de/problema)**. Algunas de ellas pueden agruparse y recibir un nombre para poder ser invocadas con facilidad tantas veces como sea necesario. Del mismo modo que los seres humanos necesitamos respirar constantemente, una aplicación informática necesita conocer la posición del puntero del ratón a cada momento, así como su actividad (si se ha hecho clic y con qué botón, si se ha soltado o si se mantiene presionado, etcétera).

¿Que es un lenguaje de programación?

Un lenguaje de programación es: un idioma artificial diseñado para expresar procesos que pueden

ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que

controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con

precisión, o como modo de comunicación humana.1 Está formado por un conjunto de símbolos y

reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y

expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el

código fuente de un programa informático se le llama programación.

Algunos de los lenguajes de programación más importantes por año son:

1943 - Plankalkül (Konrad Zuse)

1943 - ENIAC

1949 - C-10

1951 - Regional Assembly Language

1952 - Autocode

1954 - FORTRAN

1958 - LISP

1958 - ALGOL

1959 - COBOL

1962 - APL

1962 - Simula

1964 - BASIC

1964 - PL/I

1966 - BCPL

1969 - B

1970 - Pascal

1972 - C

1972 - Smalltalk

1972 - Prolog

1973 - ML

1974 - Comandos Guardados

1978 - SQL

1983 - Ada

1983 - C++

1985 - Eiffel

1987 - Perl

1989 - FL (Backus)

1990 - Haskell

1990 - Python

1991 - Java

1993 - Ruby

1995 - D

2000 - C#

2007 - UAP

2010 - Chavira



Historia de los Lenguajes de Programación. Con la idea de facilitarnos las tareas que debemos de desempeñar los humanos, hemos venido inventado diversas herramientas a lo largo de nuestra historia, que nos permiten tener una mejor calidad de vida. Los ordenadores son uno más de los inventos del hombre, aunque debemos decir que las tecnologías para su fabricación y explotación han tenido un desarrollo sorprendente a partir de la segunda mitad del siglo XX. Esta herramienta por sí sola no es capaz de efectuar ninguna tarea, es tan sólo un conjunto de cables y circuitos que necesitan recibir instrucción por parte de los humanos para desempeñar alguna tarea. El problema entonces, se puede fijar en ¿cómo vamos a poder hacer que un conjunto de circuitos desempeñen una determinada tarea y nos entreguen los resultados que nosotros esperamos?, es decir, ¿de qué manera se puede lograr la comunicación entre el hombre y el ordenador?. Así pues, tratando de dar una solución al problema planteado, surgieron los lenguajes de programación, que son como un lenguaje cualquiera, pero simplificado y con ciertas normas, para poder trasmitir nuestros deseos al ordenador. Por otro lado, como se sabe, un conjunto de circuitos no entendería ningún lenguaje que nosotros conozcamos, por más sencillo que éste parezca. Los circuitos en todo caso, sólo reconocen presencia o ausencia de energía, es decir que debemos hablarle a la máquina en su propio lenguaje (presencia y ausencia de energía, 0 y 1), o nuestro lenguaje deberá de ser traducido a un lenguaje binario cuyo alfabeto es el 0 y el 1, mediante las herramientas desarrolladas para llevar a cabo esta tarea, las cuales reciben el nombre de traductores, y como veremos más adelante, los hay de muchos tipos, dependiendo de características más específicas del lenguaje a traducir y de la manera de llevar a cabo su traducción. Como ya habréis entendido, para crear un lenguaje de programación, deberemos crear la herramienta que lo traduce, y es justamente de ellas, de las que hablaremos a continuación, para describir como han ido evolucionando en los últimos 50 años [BYTE 95]. • 1946: Konrad Zuse , un ingeniero Alemán mientras trabajaba en los Alpes de Bavaria, desarrolló el lenguaje Plankalkul, el cual, fue aplicado entre otras cosas para jugar al ajedrez. • 1949: Aparece Short Code, que viene a ser el primer lenguaje que fue usado en un dispositivo de cómputo electrónico, aunque se debe decir que se trata de un lenguaje traducido a mano. • 1951: Grace Hopper , trabajando para Remington Rand, comenzó el trabajo de diseño del primer compilador conocido ampliamente, el A-0, el cual, al ser liberado por la compañía en 1957, lo hizo con el nombre de MATH-MATIC. • 1952: Alick E. Glennie, durante su tiempo libre en la Universidad de Manchester, concibe un sistema de programación llamado AUTOCODE, que viene a ser un compilador muy rudimentario. • 1957: aparece FORTRAN (FORmula TRANslating) sistema traductor de fórmulas matemáticas. Fue desarrollado por un equipo, al frente del cual se encontraba John Backus quien después vendría a contribuir en el desarrollo del compilador para el lenguaje ALGOL y de la notación usada para la especificación sintáctica de los lenguajes, conocida como BNF (Backus Naur Form). A partir de los años sesenta, empiezan a surgir diferentes lenguajes de programación, atendiendo a diversos enfoques, características y propósitos, que más adelante describiremos. Por lo pronto, puede decirse, que actualmente existen alrededor de 2000 lenguajes de programación [KINNERSLEY 95] y continuamente, están apareciendo otros más nuevos, que prometen hacer mejor uso de los recursos computacionales y facilitar el trabajo de los programadores. Tratando de resumir un poco, presentaremos los siguientes cuadros evolutivos, donde aparecen los lenguajes que por su uso y comercialización, han resultado ser los más populares a lo largo de este medio siglo. [LABRA 98] [RUS 01]

La programación Orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación.

Con la POO tenemos que aprender a pensar las cosas de una manera distinta, para escribir nuestros programas en términos de objetos, propiedades, métodos y otras cosas que veremos rápidamente para aclarar conceptos y dar una pequeña base que permita soltarnos un poco con este tipo de programación.

## Motivación

Durante años, los programadores se han dedicado a construir aplicaciones muy parecidas que resolvían una y otra vez los mismos problemas. Para conseguir que los esfuerzos de los programadores puedan ser utilizados por otras personas se creó la POO. Que es una serie de normas de realizar las cosas de manera que otras personas puedan utilizarlas y adelantar su trabajo, de manera que consigamos que el código se pueda reutilizar.

La POO no es difícil, pero es una manera especial de pensar, a veces subjetiva de quien la programa, de manera que la forma de hacer las cosas puede ser diferente según el programador. Aunque podamos hacer los programas de formas distintas, no todas ellas son correctas, lo difícil no es programar orientado a objetos sino programar bien. Programar bien es importante porque así nos podemos aprovechar de todas las ventajas de la POO.

## Cómo se piensa en objetos

Pensar en términos de objetos es muy parecido a cómo lo haríamos en la vida real. Por ejemplo vamos a pensar en un coche para tratar de modelizarlo en un esquema de POO. Diríamos que el coche es el elemento principal que tiene una serie de características, como podrían ser el color, el modelo o la marca. Además tiene una serie de funcionalidades asociadas, como pueden ser ponerse en marcha, parar o aparcar.

Pues en un esquema POO el coche sería el objeto, las propiedades serían las características como el color o el modelo y los métodos serían las funcionalidades asociadas como ponerse en marcha o parar.

Por poner otro ejemplo vamos a ver cómo modelizaríamos en un esquema POO una fracción, es decir, esa estructura matemática que tiene un numerador y un denominador que divide al numerador, por ejemplo 3/2.

La fracción será el objeto y tendrá dos propiedades, el numerador y el denominador. Luego podría tener varios métodos como simplificarse, sumarse con otra fracción o número, restarse con otra fracción, etc.

Estos objetos se podrán utilizar en los programas, por ejemplo en un programa de matemáticas harás uso de objetos fracción y en un programa que gestione un taller de coches utilizarás objetos coche. Los programas Orientados a objetos utilizan muchos objetos para realizar las acciones que se desean realizar y ellos mismos también son objetos. Es decir, el taller de coches será un objeto que utilizará objetos coche, herramienta, mecánico, recambios, etc.

## Clases en POO

Las clases son declaraciones de objetos, también se podrían definir como abstracciones de objetos. Esto quiere decir que la definición de un objeto es la clase. Cuando programamos un objeto y definimos sus características y funcionalidades en realidad lo que estamos haciendo es programar una clase. En los ejemplos anteriores en realidad hablábamos de las clases coche o fracción porque sólo estuvimos definiendo, aunque por encima, sus formas.

**Propiedades en clases**   
Las propiedades o atributos son las características de los objetos. Cuando definimos una propiedad normalmente especificamos su nombre y su tipo. Nos podemos hacer a la idea de que las propiedades son algo así como variables donde almacenamos datos relacionados con los objetos.

**Métodos en las clases**   
Son las funcionalidades asociadas a los objetos. Cuando estamos programando las clases las llamamos métodos. Los métodos son como funciones que están asociadas a un objeto.

## Objetos en POO

Los objetos son ejemplares de una clase cualquiera. Cuando creamos un ejemplar tenemos que especificar la clase a partir de la cual se creará. Esta acción de crear un objeto a partir de una clase se llama instanciar (que viene de una mala traducción de la palabra instace que en inglés significa ejemplar). Por ejemplo, un objeto de la clase fracción es por ejemplo 3/5. El concepto o definición de fracción sería la clase, pero cuando ya estamos hablando de una fracción en concreto 4/7, 8/1000 o cualquier otra, la llamamos objeto.

Para crear un objeto se tiene que escribir una instrucción especial que puede ser distinta dependiendo el lenguaje de programación que se emplee, pero será algo parecido a esto.

miCoche = new Coche()

Con la palabra new especificamos que se tiene que crear una instancia de la clase que sigue a continuación. Dentro de los paréntesis podríamos colocar parámetros con los que inicializar el objeto de la clase coche.

**Estados en objetos**   
Cuando tenemos un objeto sus propiedades toman valores. Por ejemplo, cuando tenemos un coche la propiedad color tomará un valor en concreto, como por ejemplo rojo o gris metalizado. El valor concreto de una propiedad de un objeto se llama estado.

Para acceder a un estado de un objeto para ver su valor o cambiarlo se utiliza el operador punto.

miCoche.color = rojo

El objeto es miCoche, luego colocamos el operador punto y por último el nombre e la propiedad a la que deseamos acceder. En este ejemplo estamos cambiando el valor del estado de la propiedad del objeto a rojo con una simple asignación.

**Mensajes en objetos**   
Un mensaje en un objeto es la acción de efectuar una llamada a un método. Por ejemplo, cuando le decimos a un objeto coche que se ponga en marcha estamos pasándole el mensaje ponte en marcha.

Para mandar mensajes a los objetos utilizamos el operador punto, seguido del método que deseamos invocar.

miCoche.ponerseEnMarcha()

En este ejemplo pasamos el mensaje ponerseEnMarcha(). Hay que colocar paréntesis igual que cualquier llamada a una función, dentro irían los parámetros.

.

## Ejemplo concreto de programación orientada a objetos

Para conseguir un ejemplo concreto de lo que es la programación orientada a objetos, podemos entrar en el [Manual de PHP 5](http://www.desarrolloweb.com/manuales/58/). Realmente este manual explica las características de orientación a objetos de PHP 5 y ofrece ejemplos concretos de creación de clases con características como herencia, polimorfismo, etc.

podemos afirmar que un **lenguaje de programación** es aquella estructura que, con una cierta base sintáctica y semántica, imparte distintas instrucciones a un programa de [**computadora**](http://definicion.de/computadora).

A la hora de establecer el origen del lenguaje de programación tenemos que hacer referencia, sin lugar a dudas, a Ada Lovelace que está considerada como la primera programadora de computadoras conocida en todo el mundo. De ahí, curiosamente que se hablara en su honor del lenguaje de programación Ada. Y es que dicha figura llevó a cabo no sólo la manipulación de una serie de símbolos para una máquina del científico británico Charles Babbage sino también la consecución del establecimiento de las instrucciones necesarias para que un computador pudiera realizar una serie de cálculos iniciales.

Dentro de lo que es el lenguaje de programación es muy importante subrayar que los profesionales que se dedican a desarrollar este trabajan con un conjunto de elementos que son los que dan forma y sentido al mismo, los que permiten que aquellos funcionen y logren sus objetivos. Entre los mismos se encontrarían, por ejemplo, las variables, los vectores, los bucles, los condicionantes, la sintaxis o la semántica estática.

Las **secuencias de programación** para las acciones más usuales fueron asociadas para ser denominadas con nombres fáciles de memorizar (como **ADD** o **MUL**). Al conjunto de instrucciones se lo denomina **lenguaje ensamblador**.

Cabe diferenciar entre el lenguaje de programación y el **lenguaje informático**. Existen lenguajes informáticos que no son, en realidad, lenguajes de programación, como es el caso del [**HTML**](http://definicion.de/html/) (un **lenguaje de marcas**).

El lenguaje de programación tiene la capacidad de especificar, de forma precisa, cuáles son los datos que debe trabajar un equipo informático, de qué modo deben ser conservados o transferidos dichos datos y qué instrucciones debe poner en marcha la computadora ante ciertas circunstancias.

Existen diversos lenguajes de programación, lo que ha llevado al desarrollo de **intérpretes** (programas que adaptan las instrucciones encontradas en otro lenguaje) y **compiladores** (aquellos programas que traducen de un lenguaje a otro).

De acuerdo a su nivel de abstracción, se habla de **lenguaje de máquina** (son las cadenas binarias que pueden ser legibles de manera directa por la computadora), **lenguaje de bajo nivel** (el lenguaje de programación que se acerca al funcionamiento de una computadora), **lenguaje de medio nivel**(comparte características con los lenguajes de bajo nivel pero también con los más avanzados) o **lenguaje de alto nivel** (formado por elementos del lenguaje humano).

Muchos son los lenguajes que existen actualmente en el mercado de la tecnología y la informática. No obstante, entre los más importantes podemos citar a Pascal, Visual Basic, SQL, Delphi, Lingo, Cobol, HTML o Java. Este último, por ejemplo, se caracteriza porque fue desarrollado en el año 1995 por el científico de la computación James Gosling y porque está orientado a objetos.

Aunque sea difícil creerlo estos dos números lograron revolucionar al mundo y fueron la base para lo que hoy conocemos en cuanto a dispositivos electrónicos se refiere, el 1 y el 0 conforman el denominado Sistema Binario y es el lenguaje usado por las computadoras, el “Lenguaje Maquina”.

Los dispositivos electrónicos trabajan con dos niveles de voltaje -Encendido y Apagado- -1 y 0- respectivamente. Pero ¿Que es en si el Lenguaje Maquina?.

El Lenguaje Maquina es el conjunto de datos que la parte física de la computadora (Hardware) es capaz de comprender e interpretar “El Código Binario” comprendido por los Valores 0 y 1 con tensiones comprendidas entre 0 y 4 Voltios y 4 y 5 Voltios respectivamente, la secuencias de estos valores formaran cadenas de información para que se realice una instrucción.

Este Lenguaje fue el primero empleado por el hombre en la programación de las primeras computadoras, con secuencias como esta *01101100101001001111*prácticamente se le decía a las computadoras que hacer, esto sera muy fácil de comprender para la maquina debido a que le hablamos en su propio “idioma” pero es muy difícil de comprender para nosotros. Indicarle a una Maquina lo que debe hacer es muy distinto que indicarle a un ser Humano lo que debe hacer, aunque sea la misma orden.

En los tiempos de desarrollo de la “Maquina” el Programador debía usar esta serie de combinaciones binarias para dictar ordenes siendo él mismo el traductor, poco después se pensó ¿Si se usara la misma maquina para traducir su propio código? con esto surgieron los llamados “Compiladores” y lo que se considera el segundo lenguaje de programación el Lenguaje “Ensamblador o Assambly”

El Lenguaje Ensamblador (Assambly) esta diseñado para una computadora especifica, es decir, en los lenguajes de alto nivel no se necesita conocer la arquitectura de una computadora para programar, en el Ensamblador si, se programara de manera mas correcta para una determinada familia de Microprocesadores.

Se paso de las complicadas cadenas de dígitos por palabras claves en nuestro idioma que despues serian traducidas por la misma computadora. Ejemplo.

¿Que seria mas fácil de comprender para ti?

Esto:

01000010010100100100010101001111010001110100000101001110010001110100000101001100

O esto:

BreoganGal

Sin lugar a dudas la segunda, seria de mas fácil compresión sin mencionar que la cadena es mas corta pero antes de que pronunciemos BreoganGal ya al computadora habrá procesado esa cadena ¡y quien sabe cuantas mas!.

El lenguaje Assambly permitió el avance y compresión de las computadoras preparando el camino para que posteriormente salieran a la luz los llamados Lenguajes de alto Nivel que facilitarían aun mas el trabajo, pero el Assamby sigue siendo un lenguaje muy util y directo a la hora conectarse con la computadora, indispensable para crear los Sistemas operativos y diversas piezas que conforman el Hardware.

Un **lenguaje de programación** es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo.   
  
Por otro lado, el término "lenguaje natural" define un medio de comunicación compartido por un grupo de personas (por ejemplo: inglés o francés).   
  
Los lenguajes que los equipos usan para comunicarse entre ellos no tienen nada que ver con los lenguajes de programación; se los conoce como [protocolos de comunicación](http://es.ccm.net/contents/internet/protocol.php3). Se trata de dos conceptos totalmente diferentes. Un lenguaje de programación es muy estricto: 

A CADA instrucción le corresponde UNA acción de procesador.

El lenguaje utilizado por el [procesador](http://es.ccm.net/contents/pc/processeur.php3) se denomina **lenguaje máquina**. Se trata de instrucciones que llegan al procesador consistentes en una serie de [datos binarios](http://es.ccm.net/contents/base/binaire.php3) (ceros y unos).   
  
El lenguaje máquina, por lo tanto, no es comprensible para los seres humanos, razón por la cual se han desarrollado lenguajes intermediarios comprensibles para el hombre. El código escrito en este tipo de lenguaje se transforma en código máquina para que el procesador pueda interpretarlo.   
  
El **ensamblador** fue el primer lenguaje de programación utilizado. Es muy similar al lenguaje máquina, pero solo los desarrolladores pueden comprenderlo. A pesar de su parecido con lenguaje máquina, depende estrictamente del tipo de procesador utilizado (cada tipo de procesador puede tener su propio lenguaje máquina). Así, un programa desarrollado para un equipo no puede ser "portado" a otro tipo de equipo. El término **portabilidad** describe la capacidad de usar un programa de *software* en diferentes tipos de equipos. Para poder utilizar un programa de *software* escrito en un código ensamblador en otro tipo de equipo, ¡a veces será necesario volver a escribir todo el programa!   
  
Por lo tanto, un lenguaje de programación tiene varias ventajas: es mucho más fácil de comprender que un lenguaje máquina, y permite mayor portabilidad, es decir que puede adaptarse fácilmente para ejecutarse en diferentes tipos de equipos. Los lenguajes de programación generalmente se dividen en dos grupos principales en base al procesamiento de sus comandos: lenguajes imperativos y lenguajes funcionales.

### Lenguaje de programación imperativo

Un **lenguaje imperativo** programa mediante una serie de comandos, agrupados en bloques y compuestos de órdenes condicionales que permiten al programa retornar a un bloque de comandos si se cumple la condición. Estos fueron los primeros lenguajes de programación en uso y aún hoy muchos lenguajes modernos usan este principio

Los lenguajes de programación pueden, en líneas generales, dividirse en dos categorías: lenguajes interpretados y lenguajes compilados.

### Lenguaje interpretado

Un lenguaje de programación es, por definición, diferente al lenguaje máquina. Por lo tanto, debe traducirse para que el procesador pueda comprenderlo. Un programa escrito en un lenguaje interpretado requiere de un programa auxiliar (el intérprete), que traduce los comandos de los programas según sea necesario.

### Lenguaje compilado

Un programa escrito en un lenguaje **compilado** se traduce a través de un programa anexo llamado **compilador** que, a su vez, crea un nuevo archivo independiente que no necesita ningún otro programa para ejecutarse a sí mismo. Este archivo se llama **ejecutable**.   
  
Un programa escrito en un lenguaje compilado posee la ventaja de no necesitar un programa anexo para ser ejecutado una vez que ha sido compilado. Además, como sólo es necesaria una traducción, la ejecución se vuelve más rápida. Sin embargo, no es tan flexible como un programa escrito en lenguaje interpretado, ya que cada modificación del archivo fuente (el archivo comprensible para los seres humanos: el archivo a compilar) requiere de la compilación del programa para aplicar los cambio

En [matemáticas](https://es.wikipedia.org/wiki/Matem%C3%A1ticas), [lógica](https://es.wikipedia.org/wiki/L%C3%B3gica), [ciencias de la computación](https://es.wikipedia.org/wiki/Ciencias_de_la_computaci%C3%B3n) y disciplinas relacionadas, un **algoritmo** (del griego y latín, *dixit algorithmus* y este a su vez del matemático persa [Al-Juarismi](https://es.wikipedia.org/wiki/Al-Juarismi))[1](https://es.wikipedia.org/wiki/Algoritmo#cite_note-Brassard-1)​ es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite llevar a cabo una actividad mediante pasos sucesivos que no generen dudas a quien deba hacer dicha actividad.[2](https://es.wikipedia.org/wiki/Algoritmo#cite_note-rae-2)​ Dados un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a un estado final y se obtiene una solución. Los algoritmos son el objeto de estudio de la **algoritmia**.[1](https://es.wikipedia.org/wiki/Algoritmo#cite_note-Brassard-1)​

En la vida cotidiana, se emplean algoritmos frecuentemente para resolver problemas. Algunos ejemplos son los manuales de usuario, que muestran algoritmos para usar un aparato, o las instrucciones que recibe un trabajador de su [patrón](https://es.wikipedia.org/wiki/Empleador). Algunos ejemplos en [matemática](https://es.wikipedia.org/wiki/Matem%C3%A1tica) son el [algoritmo de multiplicación](https://es.wikipedia.org/wiki/Algoritmo_de_multiplicaci%C3%B3n), para [calcular](https://es.wikipedia.org/wiki/C%C3%A1lculo) el producto, el algoritmo de la [división](https://es.wikipedia.org/wiki/Divisi%C3%B3n_(matem%C3%A1ticas)) para calcular el cociente de dos números, el [algoritmo de Euclides](https://es.wikipedia.org/wiki/Algoritmo_de_Euclides) para obtener el [máximo común divisor](https://es.wikipedia.org/wiki/M%C3%A1ximo_com%C3%BAn_divisor) de dos [enteros](https://es.wikipedia.org/wiki/N%C3%BAmeros_enteros) positivos, o el [método de Gauss](https://es.wikipedia.org/wiki/Eliminaci%C3%B3n_de_Gauss-Jordan) para resolver un [sistema de ecuaciones lineales](https://es.wikipedia.org/wiki/Sistema_de_ecuaciones_lineales).

Por [algoritmo](https://es.wikipedia.org/wiki/es:algoritmo), se entiende a un conjunto *finito* de instrucciones que se deben seguir para resolver un problema. No obstante, desde el punto de vista de la programación de ordenadores, la definición del algoritmo como la especificación de una serie de pasos, es incompleta. Debe observarse que los ordenadores son equipos que tienen limitaciones físicas en cuanto a capacidad de almacenamiento y procesamiento. Por consiguiente debemos refinar un poco más nuestra definición de algoritmo para hacerla aplicable de manera efectiva en el ámbito de la informática.

El algoritmo es un conjunto de pasos, instrucciones o acciones que se deben seguir para resolver un problema. Existen una gran cantidad de algoritmos, hay que escoger el más efectivo. Hay dos tipos de algoritmos que son los ***cualitativos*** y ***cuantitativos***, ***cualitativos*** son todos aquellos pasos o instrucciones descritos por medio de palabras que sirven para llegar a la obtencion de una respuesta o solucion de un problema, y ***cuantitativos*** son todos aquellos pasos o instrucciones que involucran calculos numéricos para llegar a un resultado satisfactorio.

Los algoritmos son el fundamento de la programación de computadoras, para que la computadora pueda ejecutar una tarea es necesario que primero se diseñe el algoritmo correspondiente, es decir, especificar las operaciones necesarias para transformar los datos de entrada en datos de salida. Una vez diseñado y probado el algoritmo, se trasforma en el programa correspondiente. El programa esta compuesto por el algoritmo, la especificación de los datos y las instrucciones que permiten [la comunicación](http://www.monografias.com/trabajos/lacomunica/lacomunica.shtml) entre los usuarios del programa y la máquina.

**Definición de algoritmo**: Un algoritmo es un conjunto finito de instrucciones cuyo fin es realizar una tarea; este conjunto finito de instrucciones debe también ser preciso y determinístico.

* **Preciso**: el algoritmo debe ejecutar la tarea para el cual fue diseñado.
* **Determinístico**: significa que el resultado debe depender estrictamente de los datos suministrados, siempre que el algoritmo se ejecute con un mismo conjunto de datos de entrada, el resultado debe ser siempre el mismo.

Son ejemplos de algoritmos los [métodos](http://www.monografias.com/trabajos11/metods/metods.shtml) utilizados en aritmética para sumar, restar, multiplicar y dividir cantidades; la aplicación de la fórmula cuadrática para encontrar las raíces de un polinomio de segundo grado En todos ellos se cumplen las tres características anteriores.

Un algoritmo eficiente y confiable es el [producto](http://www.monografias.com/trabajos12/elproduc/elproduc.shtml) de un análisis exhaustivo del problema, para determinar la mejor alternativa de solución.

## 

## Elementos básicos en el *[diseño](http://www.monografias.com/trabajos13/diseprod/diseprod.shtml)* de algoritmos

* HERRAMIENTAS PARA EL DISEÑO DE ALGORITMOS

Las dos [herramientas](http://www.monografias.com/trabajos11/contrest/contrest.shtml) más utilizadas para diseñar algoritmos son el Pseudocódigo y el [Diagrama de flujo](http://www.monografias.com/trabajos12/diflu/diflu.shtml).

* **1. Pseudocódigo**

Es un lenguaje simplificado para describir un algoritmo utilizando una mezcla de frases en lenguaje común, y palabras claves que indican el inicio y el fin del algoritmo y las instrucciones específicas a realizar.

Por ejemplo, el siguiente pseudocódigo corresponde al algoritmo para calcular el área del rectángulo: el algoritmo calcula el área (a) de un rectángulo cualquiera, si se le suministra la longitud de de la base (b) y la longitud de la altura (h).

Los algoritmos son el fundamento de la programación de computadoras, para que la computadora pueda ejecutar una tarea es necesario que primero se diseñe el algoritmo correspondiente, es decir, especificar las operaciones necesarias para transformar los datos de entrada en datos de salida. Una vez diseñado y probado el algoritmo, se trasforma en el programa correspondiente. El programa esta compuesto por el algoritmo, la especificación de los datos y las instrucciones que permiten [la comunicación](http://www.monografias.com/trabajos/lacomunica/lacomunica.shtml) entre los usuarios del programa y la máquina.

**Definición de algoritmo**: Un algoritmo es un conjunto finito de instrucciones cuyo fin es realizar una tarea; este conjunto finito de instrucciones debe también ser preciso y determinístico.

* **Preciso**: el algoritmo debe ejecutar la tarea para el cual fue diseñado.
* **Determinístico**: significa que el resultado debe depender estrictamente de los datos suministrados, siempre que el algoritmo se ejecute con un mismo conjunto de datos de entrada, el resultado debe ser siempre el mismo.

Son ejemplos de algoritmos los [métodos](http://www.monografias.com/trabajos11/metods/metods.shtml) utilizados en aritmética para sumar, restar, multiplicar y dividir cantidades; la aplicación de la fórmula cuadrática para encontrar las raíces de un polinomio de segundo grado En todos ellos se cumplen las tres características anteriores.

Un algoritmo eficiente y confiable es el [producto](http://www.monografias.com/trabajos12/elproduc/elproduc.shtml) de un análisis exhaustivo del problema, para determinar la mejor alternativa de solución.

Los algoritmos son el fundamento de la programación de computadoras, para que la computadora pueda ejecutar una tarea es necesario que primero se diseñe el algoritmo correspondiente, es decir, especificar las operaciones necesarias para transformar los datos de entrada en datos de salida. Una vez diseñado y probado el algoritmo, se trasforma en el programa correspondiente. El programa esta compuesto por el algoritmo, la especificación de los datos y las instrucciones que permiten [la comunicación](http://www.monografias.com/trabajos/lacomunica/lacomunica.shtml) entre los usuarios del programa y la máquina.

**Definición de algoritmo**: Un algoritmo es un conjunto finito de instrucciones cuyo fin es realizar una tarea; este conjunto finito de instrucciones debe también ser preciso y determinístico.

* **Preciso**: el algoritmo debe ejecutar la tarea para el cual fue diseñado.
* **Determinístico**: significa que el resultado debe depender estrictamente de los datos suministrados, siempre que el algoritmo se ejecute con un mismo conjunto de datos de entrada, el resultado debe ser siempre el mismo.

Son ejemplos de algoritmos los [métodos](http://www.monografias.com/trabajos11/metods/metods.shtml) utilizados en aritmética para sumar, restar, multiplicar y dividir cantidades; la aplicación de la fórmula cuadrática para encontrar las raíces de un polinomio de segundo grado En todos ellos se cumplen las tres características anteriores.

Un algoritmo eficiente y confiable es el [producto](http://www.monografias.com/trabajos12/elproduc/elproduc.shtml) de un análisis exhaustivo del problema, para determinar la mejor alternativa de solución.