Introducción a la programación

Aprende a programar desde cero

Juan José Lozano Gómez – j2logo.com

2

Qué es programar. Qué es un programa

Índice de contenidos

Introducción

Este tema es una introducción a los conceptos básicos de la programación: qué es programar, qué es un algoritmo, qué es un programa, conocer los distintos tipos de lenguajes de programación y entender cómo funciona un programa.

Además, repasaremos muy brevemente los orígenes del maravilloso mundo de la programación. Esto nos ayudará a entender los primeros programas que realizaremos con *PseInt*.

Qué es programar

Formalmente, **programar consiste en ordenar y/o ejecutar una serie de acciones para la consecución de un fin**.

Por ejemplo: cuando programamos una alarma en nuestro teléfono móvil o establecemos un recordatorio en el calendario, estamos indicando una serie de acciones cuyo fin es alertarnos de un evento.

Sin embargo, a mí me gusta definir programar del siguiente modo:

> ***Programar es el arte de solucionar problemas***

Y, efectivamente, en esto consiste programar, en resolver problemas. Resolver problemas siguiendo un método. Un método que diferencia a un buen programador de uno del montón. Este método se fundamenta en tres pasos que explicaré a lo largo del tema, pero que se resumen en:

1. **Analizar el problema**. Tenemos un problema y hay que entender en qué consiste, dónde quiero llegar.
2. **Resolver**. Lo hacemos por medio de un algoritmo. Sé cómo solucionar el problema. No siempre hace falta que esté escrito.
3. **Implementar**. Convertir el algoritmo en un programa. Traducirlo utilizando un lenguaje de programación concreto.

Entre otras, **una característica fundamental de un buen programador es que nunca se salta el *paso 2***.

La programación está basada en la *algoritmia*

Desde épocas antiguas, los científicos y matemáticos han tratado de representar gráficamente cómo se realizaban los cómputos y cálculos para resolver determinados problemas. Se dieron cuenta de que los problemas matemáticos podían ser representados de manera gráfica usando una secuencia de cálculos más simples. Esto dio lugar a la *algoritmia*.

Básicamente, la algoritmia consiste en la búsqueda de soluciones a problemas concretos.

Una definición más formal es la siguiente:

La algoritmia es la ciencia que estudia los algoritmos

En la siguiente sección veremos qué es un algoritmo y por qué la algoritmia y los algoritmos son la base de la programación.

Qué es un algoritmo

En el apartado anterior vimos cómo la algoritmia trata de buscar soluciones a problemas concretos a través de los algoritmos.

> **Un algoritmo es un conjunto ordenado, no ambiguo y finito de operaciones que permite encontrar la solución a un problema.**

De ahí viene el pensamiento *raro* y *friki* de los informáticos, jajaja. Los programadores pensamos imaginando algoritmos. Es broma.

Y es que los algoritmos nos rodean por todas partes.

Cuando le preguntas a tu madre cómo se hace una comida, cuando sigues las indicaciones para montar un mueble, cuando realizas una multiplicación o cuando pretendes cambiar de canal en el televisor, en todas esas situaciones estás siguiendo un algoritmo.

Sigamos el ejemplo de la receta de cocina.

Como te indiqué anteriormente, un algoritmo es una secuencia de instrucciones ordenadas, no ambiguas y finitas que dan la solución a un problema.

Toda receta tiene como objetivo preparar un plato (el problema). Para ello, necesitamos unos ingredientes y una serie de pasos e indicaciones que especifican cómo manipular dichos ingredientes para finalmente obtener la comida deseada. Los pasos que debemos seguir hay que ejecutarlos uno a continuación del otro y siempre del mismo modo.

Como puedes observar, una receta cumple con la definición de algoritmo.

Qué es un programa

Una vez visto qué es un algoritmo, vamos a definir qué es un programa. Pero antes repasaremos cómo surgió la programación.

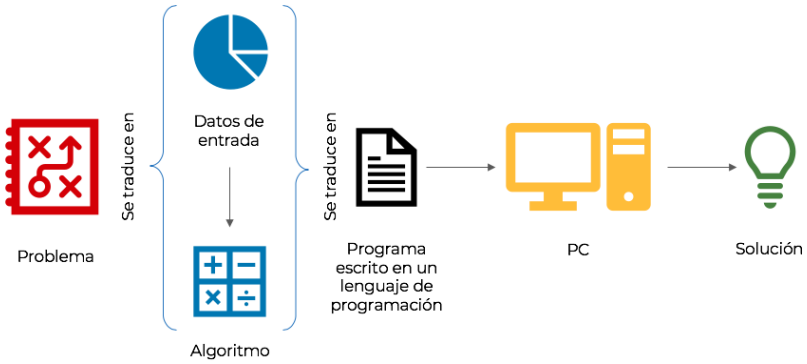
La programación tiene sus orígenes en la computación, a comienzos del siglo XX, como ciencia que estudia los cómputos. Un cómputo (en el sentido matemático) es la obtención de una solución a partir de unos datos de entrada, utilizando para ello un algoritmo.

Por eso, en informática, programar es el proceso de escribir un algoritmo que entiende un ordenador. Entonces, ¿qué es un programa?

Un programa es un conjunto de instrucciones que manipulan unos datos de entrada y generan un resultado. Además, para los mismos datos de entrada, siempre se obtiene la misma solución.

No sé si tanta definición te ha quedado del todo clara.

La idea final que quiero que interiorices es la siguiente: **Programar consiste en escribir un algoritmo como un conjunto de instrucciones que lee y ejecuta un ordenador**. Dichas instrucciones manipulan unos datos de entrada para obtener la solución a un problema. Y siempre para los mismos datos de entrada se obtiene la misma solución.



El algoritmo (las instrucciones y los datos) se tiene que escribir usando un lenguaje específico que entienda el ordenador. Es lo que se conoce como *lenguaje de programación.*

Lenguajes de programación

Al igual que las personas utilizamos un lenguaje para comunicarnos, un lenguaje de programación es la forma que tenemos de comunicarnos con el ordenador. Para que una computadora sepa interpretar y pueda ejecutar un programa, este hay que escribirlo empleando un lenguaje de programación.

Realmente, los ordenadores solo entienden un lenguaje formado por ceros y unos. Todo en la informática actual está basado en 0's y 1's. Los datos y las instrucciones de un programa se representan como secuencias de 0's y 1's. Es lo que se conoce como sistema binario.

Para que te hagas una idea, los números que utilizamos en nuestro día a día están basados en el sistema de numeración decimal. Las cantidades se representan utilizando como base aritmética las potencias del número diez. La informática, sin embargo, está basada en el sistema binario.

Por ejemplo, el número 5 en binario se representa como 00000101 y el número 8 como 00001000.

Un 0 o un 1 es lo que se conoce como bit. Una secuencia de ocho bits es un byte, 1024 bytes son un kilobyte, etc.

No voy a entrar en detalle en esto, pero te servirá de base para que entiendas todo lo que vamos a ver en esta sección y la siguiente.

Pero entonces, ¿los programas se escriben con 0's y 1's? Sí y no. En lo que resta del tema te explicaré el proceso completo desde el problema y la concepción de la idea hasta que finalmente un ordenador ejecuta un programa.

A partir de aquí vamos a tomar como referencia un problema muy simple: sumar dos números cualesquiera.

Hemos visto que los algoritmos se utilizan para representar gráficamente la solución a un problema. Los algoritmos se escriben utilizando unas anotaciones y símbolos conocidas como *pseudocódigo*.

El pseudocódigo ayuda a escribir un algoritmo sin utilizar un lenguaje de programación en concreto. Está pensado para ser entendido únicamente por las personas (no por los ordenadores) y se utiliza como paso previo a escribir un programa.

Nuestro problema de sumar dos números lo podemos representar como un algoritmo en pseudocódigo de la siguiente manera:

```
Algoritmo:

      x <- Leer número
      y <- Leer número

      resultado <- x + y

      Escribir 'El resultado es', resultado

Fin Algoritmo
```

¡Hemos escrito nuestro primer algoritmo! Por el momento no hace falta que entiendas los símbolos ni la estructura que hemos definido, pero seguro que al leerlo intuías más o menos lo que hacía. El algoritmo pide dos números al usuario, los suma y, finalmente, muestra el resultado.

El pseudocódigo es una descripción informal de un problema. No sigue ninguna regla ni existe una definición de qué es lo correcto y qué no. Tampoco define qué símbolos y caracteres utilizar. Cada uno puede escribir el pseudocódigo como quiera.

Por su parte, **un lenguaje de programación es una definición formal de símbolos y reglas gramaticales y semánticas.**

Los programas y aplicaciones que usas habitualmente se programan utilizando un lenguaje de programación. Quizá te suenen los nombres de algunos lenguajes como *C, JavaScript, Java*, *Python, PHP o Go*, aunque existen muchísimos más lenguajes.

Para que veas la diferencia entre pseudocódigo y lenguaje de programación, vamos a escribir el algoritmo anterior utilizando el lenguaje *Python*. Esto ya será un programa que puede ejecutarse en un ordenador. De nuevo, no tienes por qué entender nada:

```
x = int(input('Introduce un número > '))
y = int(input('Introduce un número > '))

resultado = x + y

print(f'{x} + {y} = {resultado}')
```

¡Hemos escrito nuestro primer programa en Python!

Como te he indicado anteriormente, hay cientos de lenguajes y de muy diverso tipo. Estos, se suelen clasificar atendiendo a diferentes criterios.

Así, tenemos **lenguajes de programación de alto nivel y de bajo nivel**.

El lenguaje ensamblador (no es un único lenguaje, hay muchos) es un ejemplo de lenguaje de bajo nivel, ya que está más cerca de programar como realmente entiende un ordenador, esto es, con 0's y 1's, que como entiende un humano.

Por su parte, todos los lenguajes que he mencionado anteriormente (*C, Java, Python*, ...) son lenguajes de alto nivel. Normalmente, estos lenguajes son de propósito general, es decir, permiten escribir todo tipo de programas y utilizan un conjunto de símbolos e instrucciones más cercanos al lenguaje utilizado por las personas.

Otra forma de clasificar los lenguajes es por paradigmas. Los paradigmas de programación distinguen entre los distintos estilos de estructurar y organizar las tareas que debe realizar un programa. Dos de los paradigmas más conocidos son: la *programación imperativa* y la *programación orientada a objetos.*

La programación imperativa es la forma de programación más usada y antigua. Se basa en dar instrucciones al ordenador de cómo hacer las cosas en forma de algoritmos.

En cambio, la programación orientada a objetos, que está basada en el paradigma imperativo, encapsula datos y comportamiento en entes denominados objetos.

Muchos lenguajes de programación, como *Python*, soportan diferentes paradigmas a la vez. Son conocidos como lenguajes multi-paradigma.

En este curso nos vamos a centrar en aprender la programación imperativa, dado que es el tipo de programación que soportan la mayoría de lenguajes.

La última clasificación que veremos (hay otras clasificaciones) es la que distingue entre *lenguajes compilados o interpretados.*

Como te adelanté al comienzo de esta sección, el único lenguaje que entiende un ordenador es aquél que está formado por 0's y 1's. Sin embargo, las personas programamos, generalmente, utilizando un lenguaje de programación de alto nivel, más próximo a nuestra

forma de hablar. Entonces, ¿cómo se traduce un programa escrito en un lenguaje de programación al lenguaje que entiende la computadora (lenguaje máquina)?

Para ello se siguen dos filosofías: *lenguajes compilados* vs *lenguajes interpretados*.

En los lenguajes compilados, una vez que hemos escrito un programa en el lenguaje de nuestra elección, hay que traducirlo al lenguaje máquina. Este proceso de traducción y transformación se conoce como *compilación* y lo realiza un programa llamado *compilador*. *C* y *C++* son ejemplos de lenguajes compilados. El resultado de compilar un programa suele ser un ejecutable. Un ejecutable es un programa escrito en binario y este es el que finalmente interpreta y ejecuta un ordenador.

Por su parte, en los programas escritos con lenguajes interpretados no se genera un ejecutable en binario. Estos programas se ejecutan por medio de un intérprete que es el que realiza la transformación del lenguaje de programación al lenguaje máquina a medida que es necesario. *Python* y *PHP* son ejemplos de lenguajes interpretados.

El siguiente diagrama muestra todo lo que hemos visto en esta sección: problema, algoritmo escrito en pseudocódigo, programa escrito en lenguaje de programación, compilación o interpretación y, finalmente, ejecución.

Cómo funciona un programa

Como hemos visto, al final, cualquier programa está formado únicamente por dos elementos: ***datos*** e ***instrucciones*** que manipulan dichos datos.

La estructura básica que sigue todo programa es la siguiente:

```
PROGRAMA:

    DATOS:

        (DEFINICIÓN DE LOS DATOS)
        DATO 1
        DATO 2
        …

    ALGORITMO:

        INSTRUCCIÓN 1
        INSTRUCCIÓN 2
        …
    FIN ALGORITMO

FIN PROGRAMA
```

Como ves, un programa define una sección para declarar los datos que se van a utilizar y una sección

para escribir las instrucciones que manipulan dichos datos.

Cuando tengas experiencia y sepas programar quizá pienses que tus programas se parecen poco a esto, sin embargo, de un modo u otro, será la estructura básica que realmente sigan.

Una vez que sabemos que un programa escrito en un lenguaje de programación, al final, se traduce a lenguaje máquina, que es lo que realmente entiende un ordenador, nos podemos preguntar: *¿Cómo funciona un programa? ¿Cómo lo ejecuta un ordenador?*

Voy a intentar explicártelo de manera muy resumida, desde el punto de vista de la programación, porque estas cuestiones dan para varios temas que se salen del ámbito de este curso.

Como sabrás, dos de los principales elementos de cualquier PC son la memoria RAM y, especialmente, la CPU.

Generalmente, los datos de un programa residen en la memoria RAM del ordenador. Concretamente, cada programa, cuando se ejecuta, tiene un espacio

reservado de memoria RAM que le asigna el sistema operativo.

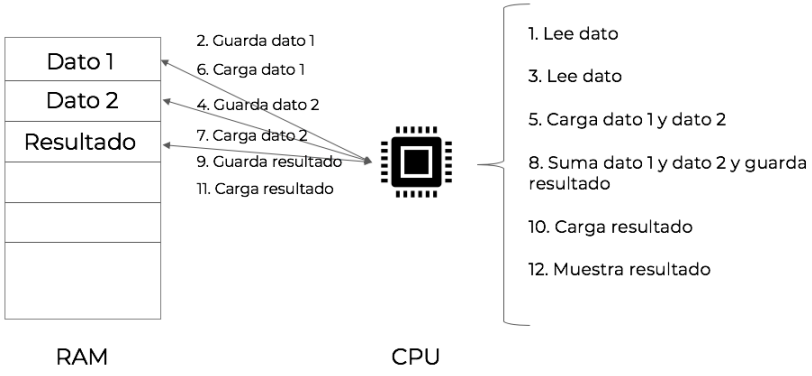
Por su parte, la CPU o procesador es el que ejecuta las instrucciones que manipulan los datos.

Los procesadores implementan un conjunto de instrucciones muy reducido si las comparamos con las instrucciones que podemos escribir en un lenguaje de programación de alto nivel. Estas instrucciones se podrían simplificar en: sumar, restar, multiplicar, dividir, leer un dato de memoria RAM, escribir un dato en la memoria RAM, ...

Por eso, generalmente, se programa utilizando los lenguajes de alto nivel y no el lenguaje ensamblador o lenguaje máquina. Además de que estos últimos son menos intuitivos, nos permiten expresar menos cosas. No obstante, las instrucciones escritas en un lenguaje de programación son finalmente traducidas a las instrucciones que realmente implementa un procesador, ya sea compilando el programa o interpretándolo.

Continuando con el ejemplo que vimos en la sección anterior de sumar dos valores, un resumen de cómo se

ejecutaría dicho programa sería el siguiente (sigue el orden de la numeración):



Introducción a PseInt

PseInt es un entorno de programación orientado al aprendizaje. No se pueden desarrollar grandes aplicaciones con él, pero sí que sirve para dar los primeros pasos en el mundo de la programación.

Lo bueno de *PseInt* es que define un lenguaje de programación que se parece más al pseudocódigo que a un lenguaje de verdad.

Es por esto por lo que se utiliza, fundamentalmente, para enseñar a programar. Y por eso mismo, lo utilizaremos en este curso. Además, el entorno y el lenguaje están en español (normalmente los lenguajes de programación utilizan el inglés).

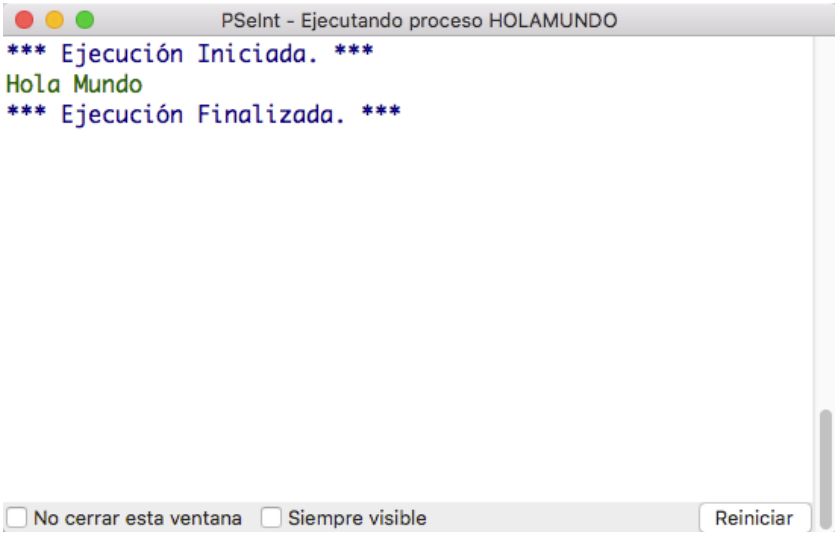
En este tema no vamos a ver el lenguaje en profundidad. Lo iremos aprendiendo poco a poco a lo largo de los siguientes módulos.

Sin embargo, es una tradición cuando se está aprendiendo un lenguaje nuevo, crear un programa conocido como ***Hola Mundo***. Este programa nos da una ligerísima idea de cómo es una aplicación, muy sencilla, escrita en dicho lenguaje y así familiarizarnos con el entorno.

Abre *PseInt* y escribe el siguiente código en el editor:

```
Algoritmo HolaMundo
 Escribir 'Hola Mundo';
FinAlgoritmo
```

A continuación, ejecuta el programa. Verás como el resultado es una ventana con el mensaje *Hola Mundo:*



Como te decía, no tienes que entender nada por el momento. A lo largo de los siguientes temas irás aprendiendo el lenguaje.

Para terminar esta unidad, vamos a escribir el programa de la suma en el lenguaje de *PseInt*.

Escribe el código siguiente en el editor y haz clic sobre el botón de ejecutar.

```
Algoritmo suma
   Definir a Como Entero;
   Definir b Como Entero;
   Definir resultado Como Entero;

   Escribir 'Introduce un numero';
   Leer a;
   Escribir 'Introduce un numero';
   Leer b;
   resultado <- a + b;
   Escribir a, ' + ', b, ' = ', resultado;
FinAlgoritmo
```

¡Acabas de implementar tu primer programa a partir de un algoritmo!

¡Genial!, ¿no? Pues lo mejor está por llegar...