

Módulo 3. Elementos y lógica de programación.

En este módulo conoceremos los elementos fundamentales de la programación desde su concepción teórica, para entender en detalle el comportamiento de cada uno frente a distintas implementaciones.

Conocer cada uno de dichos elementos es la base fundamental para todo programador, ya que nos permite plasmar directamente en código, la lógica de ejecución que vayamos elaborando.

Es imperativo para todos aquellos que quieran iniciarse en el mundo de la programación, entender, probar y aplicar todos los elementos básicos de la programación, comunes a todos los lenguajes.

Como futuros programadores, nuestro objetivo es diseñar y desarrollar distintas soluciones mediante la confección de aplicaciones informáticas.

Una definición que podemos encontrar en primera instancia sobre el desarrollo de una aplicación es: confeccionar, probar y buscar errores en un programa informático, nacido de una necesidad de solución a un determinado problema.

Cuando nos proponemos aprender a desarrollar y programar aplicaciones o sistemas, lo hacemos para cubrir determinadas necesidades, ya sean personales o de terceros, y así obtener un ingreso económico a cambio de nuestro trabajo.

Uno de los pasos fundamentales que debemos efectuar antes de comenzar es aprender la **programación lógica**.

Esto es importante porque, si bien los lenguajes de programación tienen sus particularidades, las soluciones lógicas son analizadas de un solo modo.

De esta manera, conocer este tema claramente nos permitirá migrar a todos los lenguajes que queramos.

Aprender a desarrollar aplicaciones nos ofrece muchas posibilidades, ya que podremos realizar programas en cualquier plataforma, ya sea para la Web, Windows, Linux o Macintosh; incluso, para móviles, televisión inteligente, etc.

El propósito principal es tener la base lógica de programación, y luego elegir cuál es el lenguaje en el que deseamos poner nuestro mayor esfuerzo. Puede ser el que esté latente en el mercado, uno específico de un

área (como para los trabajos científicos) o, simplemente, aquel en el que nos sintamos más cómodos para trabajar.

Al adquirir estos conocimientos, podremos tomar cualquier modelo de negocio o problema funcional de una organización, y resolverlo mediante la programación de una aplicación.


Resolver problemas.

Nuestra tarea principal será realizar una aplicación para resolver un problema en particular, o tal vez lo hagamos solo por diversión.


Por ejemplo, podemos crear un programa para llevar en nuestro teléfono móvil una agenda que nos informe los días de estreno de nuestras series favoritas de televisión. También podemos aplicarlo en el trabajo, para agilizar la toma de decisiones y digitalizar la información referida al desempeño de los empleados. Ambos son modelos de negocios distintos que plantean un problema, y nosotros debemos encontrar una solución. Estas necesidades pueden surgir desde distintos ámbitos:

- **Personal:** realizar pequeñas o amplias aplicaciones para un fin que nos beneficie. Por ejemplo: elegir una aplicación que nos indique el consumo de Internet en nuestro teléfono móvil o programar una página web personal.
- **Empresarial:** realizar sistemas informáticos, partes o módulos que tenemos que programar; incluso, arreglar un código que haya sido confeccionado por otro. Por ejemplo: utilizar nuestros conocimientos para mejorar un sistema de inventario o realizar una página web para una organización que cuenta con un módulo de ventas online.

Tengamos en cuenta que el ámbito empresarial es más duro, ya que requiere seguir ciertas pautas y criterios que veremos en los próximos capítulos. En cambio, cuando las metas son personales, podemos dedicarnos a desarrollar de manera **freelance**.



EL PLANTEO DE
METAS ES UN PUNTO
EXCLUYENTE EN EL
DESARROLLO DE
APLICACIONES



Las metas empresariales son estrictas y, en general, nos afectan, ya que, por ejemplo, nos imponen un límite de tiempo específico que debemos cumplir.

Dentro del desarrollo de aplicaciones, una meta empresarial que debe influir en nuestros objetivos personales es absorber los conocimientos del grupo de trabajo, para luego aplicarlos a los nuevos desafíos que vayamos afrontando más adelante.

El planteo de metas es un punto excluyente en el desarrollo de aplicaciones, porque tener en claro hacia dónde queremos llegar nos motivará a nivel personal a seguir investigando, buscando y probando.

Al mismo tiempo, nos ayudará a plantearnos los objetivos buscados sobre los desarrollos a realizar. De esta forma, algo que parece tan sencillo como plantearse una meta y conocer los objetivos nos permitirá organizar y optimizar el desarrollo.

Tipos de aplicaciones.

En el mercado informático actual, nos encontramos con diferentes soportes de hardware que albergan variados tipos de aplicaciones, ya sea exclusivas de Internet, del sistema operativo o de un aplicativo en particular.

Así como antes comenzamos a formar el concepto de desarrollo de una aplicación, ahora vamos a reforzarlo haciendo un repaso de las aplicaciones existentes, de modo de tener una idea gráfica de qué podemos considerar para nuestro trabajo.

Aplicaciones web

Las aplicaciones web son herramientas muy comunes en organizaciones que desean ampliar las fronteras de sus modelos de negocios o, simplemente, alcanzar la autogestión para empleados, alumnos, docentes, etcétera.

Hay una amplia variedad de sitios web destinados a distintos rubros, como puede ser el automotriz, en donde es posible personalizar o armar autos a gusto, elegir colores, definir agregados, etc.

Esto nos demuestra la variedad de trabajo que se puede realizar en los desarrollos para la Web.

Aplicaciones de escritorio

Las aplicaciones de escritorio son aquellas que funcionan sobre un sistema operativo de PC (computadora personal) o notebook.

Los desarrollos en este ámbito también son enormes, y podemos encontrarnos con algunos muy costosos, utilizados por grandes empresas; y con otros gratuitos y útiles que pueden servirnos para diferentes tareas.

Veremos que muchos de estos programas cuentan con un tipo de distribución llamado **trial**.

Se trata de una instalación de prueba, generalmente por un máximo de 30 días a partir de su instalación, que suele tener funcionalidades limitadas.

Otras versiones de prueba gratuitas pueden ser **shareware** o **freeware**, que podemos instalar y utilizar en los equipos que queramos.

Aplicaciones móviles

Son aplicaciones que se utilizan en equipos móviles, como teléfonos celulares o tabletas. Suelen ser muy similares a las de escritorio, ya que permiten realizar las mismas tareas, aunque el ingreso de datos es táctil o por voz.

Para visualizar algunos ejemplos, podemos visitar la página del mercado de Android, donde hay una infinidad de opciones gratuitas y pagas:

<https://play.google.com/store>.

Entrada – Proceso – Salida.

La **entrada** es el ingreso o comando de datos que vamos a realizar sobre un dispositivo, como, por ejemplo: tocar la pantalla, escribir, mover el puntero del mouse, hacer el movimiento con un joystick, etc.

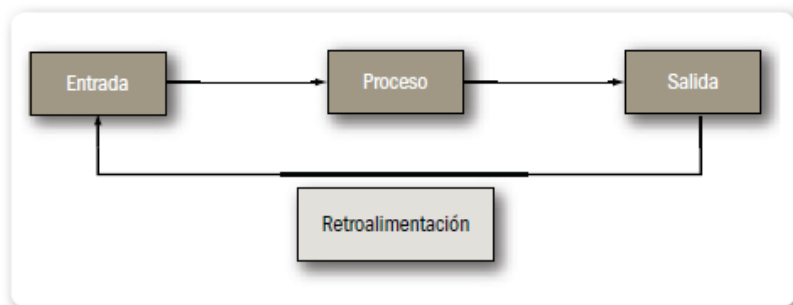
Por lo tanto, toda entrada se hará por medio de un dispositivo, como puede ser una pantalla táctil, un teclado, una webcam o un mouse.

El **proceso** es el trabajo, la interpretación y el cálculo de la información ingresada. Esta información puede ser un movimiento del mouse, una tecla pulsada, datos para calcular enviados, y otros.

Fundamentalmente, en el proceso ya entran en juego el procesador y la memoria de un dispositivo.

La **salida** es el resultado de las acciones que se efectúan sobre la información. Por lo tanto, si pulsamos el botón del mouse, se ejecutará una aplicación (pulsar el botón Enviar de un correo), se realizará una acción en un juego (como disparar), se devolverá el resultado de un cálculo, se ejecutará un video, y otras opciones más.

Este proceso de retroalimentación nos dará los mismos resultados, presionando ya sea uno o varios botones del teclado.

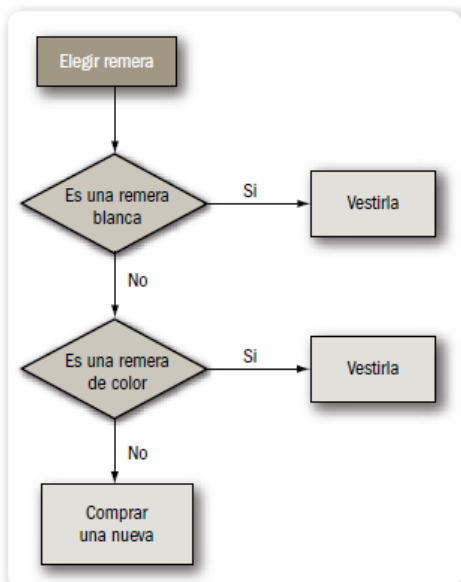


Algoritmo.

Si bien encontraremos múltiples definiciones de lo que es un algoritmo, nosotros trabajaremos con la genérica que toma la RAE, en la que se hace referencia a un conjunto ordenado y finito de operaciones que permite hallar la solución de un problema.

Nosotros, como seres humanos, tenemos incorporado un “**algoritmo**” de decisiones.

Por ejemplo, si deseamos vestir una remera, realizamos un proceso de selección de cuál o tal queremos, y terminamos por hacer la selección deseada. En un conjunto ordenado y finito de operaciones, podríamos representar, a través de un algoritmo, este proceso de selección y solución.



De esta manera, podemos definir el algoritmo como una serie de pasos ordenados que debemos seguir para lograr, finalmente, la resolución de una situación o problema.

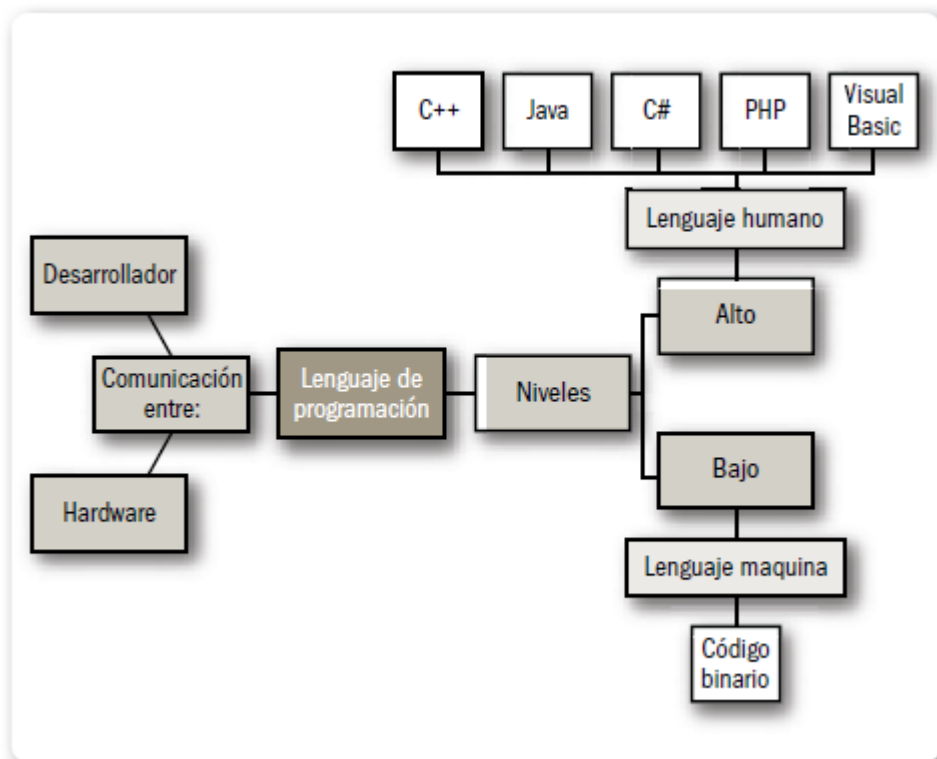
En el desarrollo, para poder ejecutar una aplicación, tenemos que traducir esto a sentencias ordenadas de código que se ejecuten línea a línea.

Lenguaje de programación.

Anteriormente presentamos la comunicación que existe entre un software y el hardware.

Ahora vamos a conocer la comunicación que debemos establecer nosotros, como desarrolladores, frente a nuestro hardware, para lograr que este ejecute las tareas o procesos que deseamos.

Para este fin, necesitaremos como herramienta primordial un **lenguaje de programación**.



Existen muchos lenguajes de programación que nos permiten desarrollar, por medio de un código (protocolo), sentencias algorítmicas que luego son traducidas a lenguaje máquina.

Estos cumplen la función de intermediarios entre el desarrollador y el hardware.

Teniendo en cuenta esta diversidad, veremos que hay dos grupos generales. Por un lado, se encuentran los lenguajes más próximos a la arquitectura del hardware, denominados lenguajes de bajo nivel (son más rígidos y complicados de aprender). Por otro lado, están aquellos más cercanos a los programadores y usuarios, denominados lenguajes de alto nivel (son más comprensibles para el lenguaje humano).

En distintos escritos se consideran lenguajes de **bajo nivel** a algunos como: FORTRAN, ASSEMBLER y C. Como lenguajes de **alto nivel** podemos mencionar: Visual Basic, Visual C++ y Python.

Si bien podemos encontrar categorizaciones más finas al respecto, que describan diferentes tipos de lenguajes, recordemos que, en términos generales, siempre se habla de lenguajes de alto nivel y de bajo nivel.

| ▼ LENGUAJE | ▼ DESCRIPCIÓN |
|--------------------|---|
| Máquina | Código interpretado directamente por el procesador. Las invocaciones a memoria, como los procesos aritmético-lógicos, son posiciones literales de conmutadores físicos del hardware en su representación booleana. Estos lenguajes son literales de tareas. Ver su ejemplo en la Figura 10. |
| Bajo nivel | Instrucciones que ensamblan los grupos de conmutadores necesarios para expresar una mínima lógica aritmética; están íntimamente vinculados al hardware. Estos lenguajes están orientados a procesos compuestos de tareas, y la cantidad de instrucciones depende de cómo haya sido diseñada la arquitectura del hardware. Como norma general, están disponibles a nivel firmware, CMOS o chipset. Ver su ejemplo en la Figura 11. |
| Medio nivel | Son aquellos que, basándose en los juegos de instrucciones disponibles (chipset), permiten el uso de funciones a nivel aritmético; pero a nivel lógico dependen de literales en ensamblador. Estos lenguajes están orientados a procedimientos compuestos de procesos. Este tema se verá a continuación, en el ejemplo Código C y Basic. |
| Alto nivel | Permiten mayor flexibilidad al desarrollador (a la hora de abstraerse o de ser literal), y un camino bidireccional entre el lenguaje máquina y una expresión casi oral entre la escritura del programa y su posterior compilación. Estos lenguajes están orientados a objetos. Ver su ejemplo en la Figura 12. |

Ciclo de vida (etapas) de una resolución de un problema.

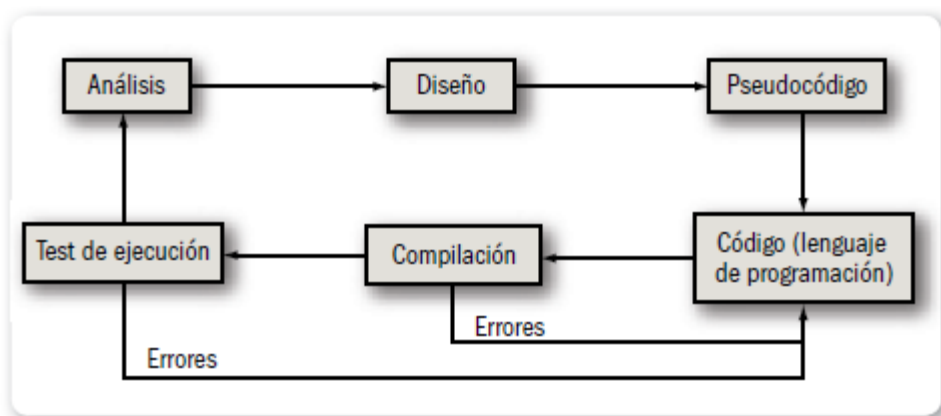
Ahora que conocemos las herramientas involucradas en el desarrollo de aplicaciones, es conveniente revisar qué tareas generales debemos considerar para llevar a cabo esta labor.

Como seres humanos, tenemos incorporada intuitivamente la resolución de problemas cotidianos gracias a nuestra experiencia, y para intentar afrontar un inconveniente, solemos hacer un proceso rápido de selección e intentamos buscar la opción más favorable.

En el ámbito laboral, y más aún en el desarrollo de aplicaciones, debemos ser cautelosos al momento de resolver alguna tarea o proceso.

Por eso, nos será de gran utilidad aplicar una herramienta del lenguaje de programación que nos permita confeccionar un programa y, así, resolver dicha situación.

Si bien este esquema nos será útil para la resolución de un desarrollo sencillo, en caso de trabajar con sistemas amplios, deberemos incluir ciertas técnicas de ingeniería del software.



En el proceso que vemos en el gráfico de la figura, puede suceder que debamos retroceder y volver a analizar o replantear algunas de las acciones.

Revisemos los pasos expuestos en el esquema (y en los próximos capítulos veremos cómo se desarrolla una aplicación basada en él).

Los siguientes aspectos son pasos que seguiremos como desarrolladores para resolver una situación:

- Analizar el problema que vamos a resolver.
- Diseñar una solución.
- Traducir la solución a pseudocódigo.
- Implementar en un lenguaje de programación todo lo analizado.
- Compilar el programa.
- Realizar pruebas de ejecución.
- Corregir los errores que haya.