Elementos de desarrollo del Software

ÍNDICE

1. Programa informático

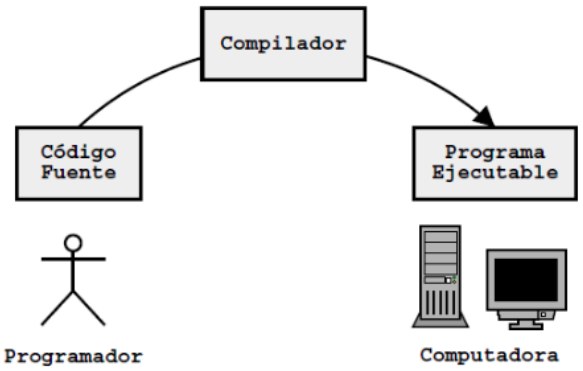
Un **programa informático** es un conjunto de instrucciones escritas en un lenguaje de programación que aplicadas sobre un conjunto de datos resuelven un problema o parte del mismo.

2. Creación de un programa

Cuando nos disponemos a codificar un programa lo hacemos en un determinado lenguaje cercano a nosotros (entendible por nosotros) pero que la máquina no entiende. Con ello obtenemos el código fuente de nuestro programa.

Una vez acabado, deberemos transformar ese código en un lenguaje que comprenda el ordenador y obtener el código ejecutable.

Para ello pasará por distintas fases.



- **Código fuente:** conjunto de instrucciones escritas en un lenguaje de programación determinado. Es el código en el que nosotros escribimos nuestro programa.

- **Código objeto:** el código objeto es el código resultante de compilar el código fuente. Si se trata de un lenguaje de programación compilado, el código objeto será código máquina, mientras que, si se trata de un lenguaje de programación virtual, será código bytecode.

- **Código ejecutable:** el código ejecutable es el resultado obtenido de enlazar nuestro código objeto con las librerías. Este código ya es nuestro programa ejecutable, programa que se ejecutará directamente en nuestro sistema o sobre una máquina virtual en el caso de los lenguajes de programación virtuales.

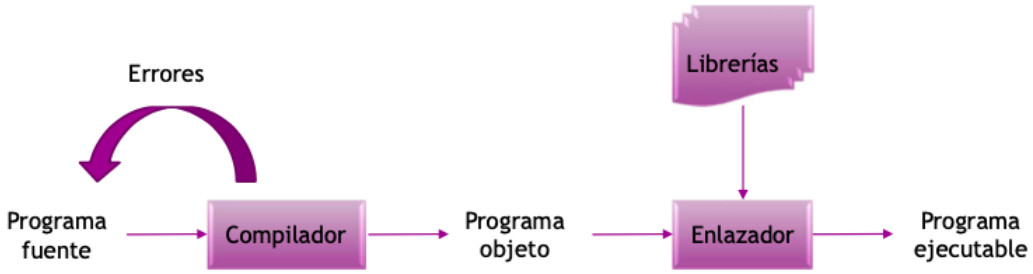
2.1. Obtención de código ejecutable

Para obtener código binario ejecutable tenemos 2 opciones:

- Compilar
- Interpretar

2.2. Proceso de compilación/interpretación

Aunque el proceso de obtener nuestro código ejecutable pase tanto por un compilador como por un enlazador, se suele llamar al proceso completo "**compilación**".



La compilación/interpretación del código fuente se lleva a cabo en varias fases:

- **1. Análisis léxico:**

  Se asegura que el conjunto de **símbolos** esté dentro de los permitidos.

  Ejemplo: El coche de maría. → CORRECTO

  El coche ed maría → INCORRECTO

  ¿Los lexemas o tockens que obtenemos son o no correctos en la lengua castellana?

- **2. Análisis sintáctico:**

  Se asegura que se aplican las **normas** de construcción permitidas de los símbolos del lenguaje.

  Ejemplo: El compilar. error, desarrollo puede grave entornos

  CORRECTO LÉXICAMENTE – INCORRECTO SINTÁCTICAMENTE

  ¿Se utilizan correctamente las reglas de la lengua castellana? Los signos de puntuación no se usan de forma correcta

- **3. Análisis semántico**

  Se asegura que el significado de construcción es coherente para realizar acciones válidas.

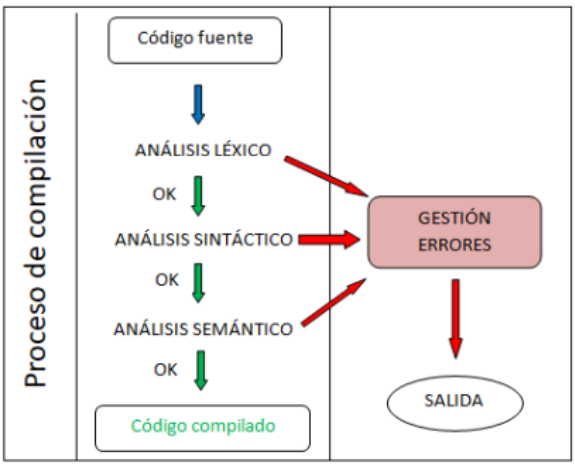
  Ejemplo: La abeja compila las montañas del garbanzo

  CORRECTO LÉXICAMENTE y SINTÁCTICAMENTE– INCORRECTO SEMÁNTICAMENTE

  ¿Tiene sentido lo que se expresa? No

- **4. Generador de código objeto**

  Si no existen errores se genera el código objeto.



- **5. Enlazador.**

  Enlaza con las librerías para generar **código ejecutable.**

- **Conclusión**

Si no existen errores, se genera el código objeto correspondiente/se ejecuta la línea

→ Un código fuente correctamente escrito no significa que funcione según lo deseado.

2.3. Lenguajes compilados

C, C++, Java, Cobol, Pascal, Basic, Fortran

- **Principal ventaja**: Ejecución muy eficiente.
- **Principal desventaja**: Es necesario compilar cada vez que el código fuente es modificado.

2.4. Lenguajes interpretados

PHP, Javascript, Script de Shell

- **Principal ventaja**: El código fuente se interpreta directamente.
- **Principal desventaja**: Ejecución menos eficiente.

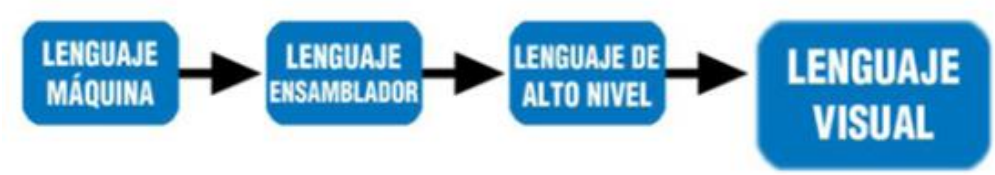
3. Lenguajes de programación

Se puede definir un **Lenguaje de Programación** como un idioma creado de forma artificial para obtener un código que el hardware de la computadora pueda entender y ejecutar. De esta manera podemos indicarle lo que deseamos que haga.

Existen diversos tipos de lenguajes de programación que según las necesidades del proyecto hace que unos sean más idóneos que otros. Es en la etapa de diseño cuando típicamente se elige el lenguaje de programación a utilizar y en la fase de desarrollo cuando se hace uso de ellos.

En otras palabras, es muy importante tener muy clara la función de los lenguajes de programación. Son los instrumentos que tenemos para que el ordenador realice las tareas que necesitamos.

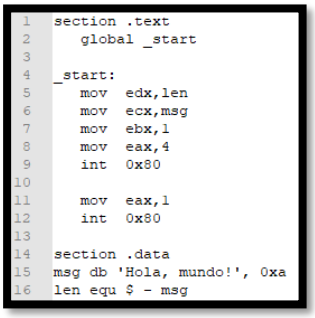
Los lenguajes de programación han sufrido su propia evolución, como se puede apreciar en la siguiente figura siguiente:



Lenguaje máquina:

- Fue el primer lenguaje utilizado.
- Sus instrucciones son combinaciones de unos y ceros.
- Es el único lenguaje que entiende directamente el ordenador. No necesita traducción.
- Es único para cada procesador (no es portable entre arquitecturas hardware).

Lenguaje ensamblador:



```
section .text
   global _start

_start:
   mov  edx,len
   mov  ecx,msg
   mov  ebx,1
   mov  eax,4
   int  0x80

   mov  eax,1
   int  0x80

section .data
msg db 'Hola, mundo!', 0xa
len equ $ - msg
```

- Sustituyó al lenguaje máquina para facilitar la labor de programación.
- En lugar de unos y ceros se programa usando mnemotécnicos (instrucciones complejas).
- Necesita traducción al lenguaje máquina para poder ejecutarse.

Lenguaje de alto nivel:

- Sustituyeron al lenguaje ensamblador para facilitar más la labor de programación.
- En lugar de mnemotécnicos, se utilizan sentencias y órdenes derivadas del idioma inglés.
- Necesita traducción al lenguaje máquina.

Lenguaje visual:

- En lugar de sentencias escritas, se programa gráficamente usando el ratón y directamente la apariencia del software.
- Se genera un código de alto nivel equivalente de forma automática.

4. Clasificación de los lenguajes de programación

Podemos clasificar los distintos tipos de Lenguajes de Programación en base a distintos criterios:

A. Según lo cerca que esté del lenguaje humano

1. **Lenguajes de programación de alto nivel.**
   - Por su esencia, están más próximos al razonamiento humano.
   - Son independientes a la arquitectura del ordenador, por lo que para su ejecución precisan traducción al lenguaje máquina, pueden ser ejecutados en cualquier plataforma.
2. **Lenguajes de programación de bajo nivel.**

   Están más próximos al funcionamiento interno de la computadora. Tienen muy buenos rendimientos. Tenemos:

   - **Lenguaje máquina.** Se programan sus registros directamente con 0s y 1s. Difícil de programar y resolver errores.
   - **Lenguaje ensamblador.** Se trata de un primer nivel de abstracción respecto al lenguaje máquina, aunque conceptualmente está mucho más cercano al equipo que al razonamiento humano.

- **B. Lenguajes interpretados vs compilados**

1. **Lenguaje compilado**.
   - El programa se traduce con un programa llamado compilador, que crea un fichero binario ejecutable particular para un tipo de máquina.
   - Este ejecutable ya no necesita ningún otro programa para ser utilizado.
   - Por ejemplo, C++.
2. **Lenguaje interpretado.**
   - El programa es transformado a un formato acordado (bytecode), no interpretable directamente por ningún tipo de máquina.
   - Posteriormente se utilizarán diferentes intérpretes (dependiendo de la plataforma donde se ejecuta), para conseguir el código máquina correspondiente.
   - Por ejemplo, Java.

- **C. Lenguajes imperativos vs declarativos**

1. **En la programación imperativa**:
   - Se describe paso a paso un conjunto de instrucciones que deben ejecutarse para variar el estado del programa y hallar la solución, es decir, un algoritmo en el que se describen los pasos necesarios para solucionar el problema.
   - Ejemplo: Basic, C, C++, Fortran, Pascal, Perl, PHP, Java, lenguajes ensamblador.
2. **En la programación declarativa:**
   - Las sentencias que se utilizan describen el problema que se quiere solucionar, pero no las instrucciones necesarias para hacerlo.
   - Algunos ejemplos son:
     - Lógicos: prolog.
     - Algebraicos: SQL.
     - Funcionales: Haskell.

- **C. Según la técnica de programación utilizada**

1. **Lenguajes de programación estructurados:**
   - Usan la técnica de programación estructurada (sentencias secuenciales, condicionales y repetitivas).
   - Ejemplos: Pascal, C, etc.

2. **Lenguajes de programación orientados a objetos (POO).**

   - Usan la técnica de programación orientada a objetos.
   - Ejemplos: C++ (C plus plus), Java, Ada, C#, .Net, etc.

3. **Lenguajes de programación visuales.**

   - Basados en las técnicas anteriores, permiten programar gráficamente, posteriormente se obtiene un código equivalente de forma automática.
   - Ejemplos: Visual Basic.Net, C#, .Net, etc.

La programación estructurada fue de gran éxito por su sencillez a la hora de construir y leer programas. Luego llegó la programación modular, seguida de la programación orientada a objetos y finalmente la programación visual.

Veamos con más detalle las características de los lenguajes de programación estructurada, modular y orientados a objetos en los siguientes puntos.

4.1. Lenguajes de programación estructurada

La programación estructurada se define como una técnica que permite sólo el uso de tres tipos de sentencias o estructuras de control:

- Sentencias secuenciales.
- Sentencias selectivas (condicionales). Ejemplo if-else
- Sentencias repetitivas (iteraciones o bucles). Ejemplo for

Los lenguajes de programación que se basan en la programación estructurada reciben el nombre de lenguajes de programación estructurados.

Ventajas

- Los programas son fáciles de leer, sencillos y rápidos. La estructura del programa es sencilla y clara.
- El mantenimiento de los programas es sencillo.

Inconvenientes

- Todo el programa se concentra en un único bloque (si se hace demasiado grande es difícil manejarlo).
- No permite reutilización eficaz de código, ya que todo va "en uno".
- La programación estructurada evolucionó hacia la programación modular, que divide el programa en trozos de código llamados módulos con una funcionalidad concreta, que podrán ser reutilizables.

4.2. Lenguajes de programación modular

- Son aquellos que permiten dividir los programas grandes en trozos más pequeños (siguiendo la conocida técnica "divide y vencerás").

- La programación estructurada evolucionó hacia la programación modular, que divide el programa en trozos de código llamados módulos con una funcionalidad concreta, que podrán ser reutilizables.

4.3. Lenguajes Orientados a Objetos

En la **P.O.O.** los programas se componen de objetos independientes entre sí que colaboran para realizar acciones.

Los **lenguajes de programación orientados a objetos** tratan a los programas no como un conjunto ordenado de instrucciones (tal como sucedía en la programación estructurada) sino como un conjunto de objetos que colaboran entre ellos para realizar acciones.

- **Características de la programación OO:**

  - Se define **clase** como una colección de objetos con características similares (Atributos).

    *Si en nuestro programa interactúan un conjunto de personas, la clase a implementar será Persona y los **objetos** serán personas en particular (Noa, Valeria, Mila ...). Las clases definen una serie de atributos, que caracterizan a cada objeto persona. Por ejemplo, el atributo edad tendrá el valor 4 para Noa, 7 para Valeria ....*

  - Las **clases** definen una serie de métodos que corresponden a las acciones que pueden llevar a cabo los **objetos**.

    Por ejemplo, en la clase Persona se puede definir el método Saludar, que puede ser utilizado por Mila para dar los buenos días. Estos métodos contendrán el código que da respuesta a las acciones que implementan. Mediante llamadas a los métodos, unos objetos se comunican con otros produciéndose un cambio de estado de los mismos. Esto se denomina envió de mensajes.

  - El código es más reutilizable.

- **Términos relativos a la programación OO:**

  - Clase
  - Objeto
  - Mensaje
  - Método
  - Evento (sistema)
  - Propiedad o atributo
  - Estado interno
  - Abstracción
  - Encapsulamiento
  - Polimorfismo
  - Herencia

- **Lenguajes orientados a objetos son:**

  - Ada, C++ (C plus plus), VB.NET, C#, Java, PowerBuilder, etc.

5. JAVA

5.1. Características

- Lenguajes compilado e interpretado.
- El código fuente Java se compila y se obtiene un código binario intermedio denominado bytecode.
- Puede considerarse código objeto, pero destinado a la máquina virtual de Java en lugar de código objeto nativo.
- Después este bytecode se interpreta para ejecutarlo.

5.2. Ventajas y desventajas

- Ventajas:
  - Estructurado y Orientado a Objetos
  - Relativamente fácil de aprender
  - Buena documentación para aprenderlo y base de usuarios
- Desventajas:
  - Menos eficiente que los lenguajes compilados

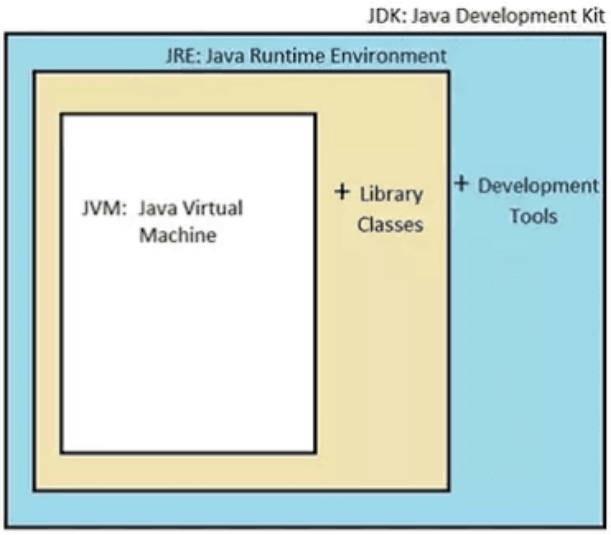
5.3. JRE, JDK, JVM

Cuando hablamos de Java solemos hacer referencia a tres conceptos que, en algunos casos, nos pueden presetar confusión.

- **JVM** (Java Virtual Machine): es responsable de ejecutar el programa Java (bytecodes) línea por línea, por lo que también se lo conoce como intérprete.
- **JRE** (Java Runtime Environment): proporciona los requisitos mínimos para ejecutar una aplicación Java; consiste en Java Virtual Machine (JVM), clases principales y archivos auxiliares.
- **JDK** (Java Development Kit): es un entorno de desarrollo de software utilizado para desarrollar aplicaciones y applets de Java. Incluye Java Runtime Environment (JRE), un intérprete/cargador (Java), un compilador (javac), un archivador (jar), un generador de documentación (Javadoc) y otras herramientas necesarias para el desarrollo de Java.

  En resumen, es un kit que proporciona el entorno para desarrollar y ejecutar el programa Java que incluye dos cosas:

  - Herramientas que proporcionan un entorno de desarrollar para programas Java.
  - JRE para ejecutar los programas Java desarrollados.



JDK = JRE + Development Tools
JRE = JVM + Library Classes

5.4. Compilador JIT