por Nacho Iborra

Entornos de Desarrollo
Bloque 1
Tema 1: Programas, lenguajes y compiladores

1.1.1. Software y programas

Si buscamos por Internet definiciones del concepto *software* podemos encontrar varias, aunque básicamente todas son equivalentes. Entendemos por **software** el conjunto de programas, documentación y datos relacionados con un sistema informático.

Cada producto software es (puede ser) distinto al resto, ya que se desarrolla para un cliente diferente, o para cumplir una finalidad diferente. Así, construirlo implica entender lo que se tiene que hacer, realizar un diseño previo e implementarlo, como veremos más adelante. En ese sentido, no podemos comparar el desarrollo de software con una producción industrial (como la fabricación de teclados, por ejemplo), donde todo es más mecánico y automatizado. El desarrollo requiere de la creación de un proyecto software, y de un equipo trabajando de forma coordinada. Además, el software, a diferencia de los productos físicos, no se estropea, aunque sí puede llegar a deteriorarse su rendimiento con las sucesivas actualizaciones y ampliaciones que se le hagan.

1.1.1.1. Componentes del software

De la definición anterior de software, podemos entender que está compuesto por tres elementos:

- **Programas**: son conjuntos de instrucciones que proporcionan la funcionalidad deseada. Están escritos en lenguajes de programación específicos.
- **Datos**: los programas necesitan datos con los que trabajar y manipular. Estos datos pueden ser obtenidos y/o almacenados en bases de datos o archivos.
- **Documentos**: en los documentos del software se explica cómo utilizar el programa, especificaciones de cómo se ha realizado (para poder hacer cambios posteriormente), etc.

1.1.1.2. Tipos de software

Existen fundamentalmente dos tipos de software:

- **Software de aplicaciones**: proporciona servicios al cliente, como por ejemplo aplicaciones de tratamiento de textos, hojas de cálculo, contabilidad, etc. Dentro de este

tipo de software podemos encontrar, además, subcategorías, tales como el software de gestión (nóminas, contabilidad, inventarios...), software de ingeniería o científico (programas de dibujo asistido (CAD), simuladores...), software de red o Internet (navegadores, clientes FTP...), etc.

- **Software de sistemas**: se utiliza para operar y mantener un sistema informático. Básicamente son programas que ayudan a otros programas. En este grupo entrarían, por ejemplo, los sistemas operativos o los compiladores.

1.1.2. Lenguajes y compiladores

Hemos visto que los programas son conjuntos de instrucciones que se proporcionan a un ordenador para realizar una tarea. Estas instrucciones se escriben en un **lenguaje** de programación que elijamos, generando unos archivos de código llamados **código fuente**, escritos en el lenguaje escogido.

1.1.2.1. Tipos de lenguajes

A la hora de elegir o valorar un lenguaje de programación, distinguimos principalmente entre lenguajes de **alto nivel** (más parecidos al lenguaje humano, y por tanto, más fáciles de comprender por las personas que los utilizan), y de **bajo nivel** (más cercanos al código máquina, y por tanto más difíciles de entender por las personas, pero más rápidos de procesar y traducir por parte de los ordenadores).

- Entre los lenguajes de **alto nivel**, que son los que se utilizan habitualmente, existe un amplio abanico donde elegir, dependiendo del tipo de aplicación que vayamos a desarrollar. Hablamos de lenguajes como C, C++, C#, Java, Javascript, PHP, Python, etc.
- Entre los lenguajes de **bajo nivel**, el más popular es el lenguaje ensamblador, con un conjunto de instrucciones muy definido que se traducen fácilmente a lenguaje máquina.

Aquí vemos un sencillo programa escrito en lenguaje Java, que simplemente muestra el mensaje "Hola" por pantalla:

```
public class Prueba
{
        public static void main(String[] args)
        {
                System.out.println("Hola");
        }
}
```

El mismo programa escrito en lenguaje C quedaría así:

```c
#include <stdio.h>

int main()
{
        printf("Hola");
        return 0;
}
```

1.1.2.2. Compiladores y otros procesadores de lenguajes

El ordenador no es capaz por sí mismo de entender ninguno de los lenguajes de programación que los humanos utilizamos para desarrollar programas. Para hacerlos funcionar, esas instrucciones se deben traducir a un lenguaje que el ordenador sea capaz de entender, denominado *código máquina*, compuesto por bits (ceros y unos).

Para hacer una traducción de un lenguaje de programación determinado al código máquina entendible por un ordenador, se utilizan unas herramientas llamadas **compiladores**, aunque esta definición no es del todo correcta. En general, existen distintos procesadores de lenguajes de programación:

- **Compiladores**: se encargan de traducir el código escrito en el lenguaje de programación escogido en código ejecutable (código máquina), generando para ello un archivo ejecutable en el sistema operativo en cuestión. Por ejemplo, si compilamos un programa hecho en C++ bajo Windows, se generará un archivo con extensión *EXE* que podremos ejecutar directamente.
- **Intérpretes**: se encargan de traducir "al vuelo" el código del lenguaje de programación, sin generar ningún archivo ejecutable, de modo que si queremos volver a ejecutar el programa, necesitamos disponer del código fuente original para volverlo a interpretar. Es habitual que lenguajes como PHP o Python tengan intérpretes en lugar de compiladores. Su tiempo de respuesta es algo más lento a la hora de traducir el código cada vez, pero da más independencia de la plataforma sobre la que se está ejecutando.
- **Máquinas virtuales**: una vía intermedia entre la compilación y la interpretación es la que utilizan lenguajes como por ejemplo Java. Los programas Java no se compilan a código máquina nativo (no se genera un archivo *EXE* en Windows al compilar un programa Java, por ejemplo), pero tampoco se interpretan cada vez que se ejecutan. A cambio, Java compila el código fuente en un código intermedio propio, y pone en marcha una máquina virtual (JVM, *Java Virtual Machine*) que se encarga de interpretar y ejecutar ese código intermedio cada vez que se necesite. Así, no necesitamos disponer del código fuente Java original cada vez que queramos ejecutar el programa, ni tampoco dependemos de una plataforma concreta (Windows, Linux, etc.), sino que basta con tener una máquina virtual Java en marcha para poder ejecutar el programa compilado. Una idea similar es la que sigue el lenguaje C# con su plataforma virtual *.NET*.

> Ejercicios propuestos:

> 1.1.2.1. Existen diversos estudios y análisis que tratan de clasificar los lenguajes de programación por su popularidad o uso actual. Quizá uno de los más completos sea el que realiza la web RedMonk, basándose en un cruce de datos entre la principal web/repositorio de código (GitHub) y la principal web para consulta de dudas (StackOverflow). Aquí puedes consultar uno de sus más recientes análisis. Examinadla en clase, valorad los resultados y contestad a las siguientes preguntas: ¿Cuántos de los primeros 20 lenguajes del ranking conocías? ¿Echas en falta alguno de los lenguajes que conoces entre esos 20 primeros?

> 1.1.2.2. Vamos a instalar un compilador que utilizaremos después durante el módulo en multitud de ocasiones: el compilador de Java. Puedes descargar el kit de desarrollo de Java (JDK) desde esta web (recomendamos instalar la versión 8 al menos). Descarga el paquete adecuado para tu plataforma e instálalo. Después, abre un terminal y escribe el comando `javac -version`. Deberá aparecerte en el terminal algo parecido a esto (la versión variará dependiendo de cuál instales):

```
javac 1.8.0_181
```

> Después, crea un archivo llamado "Prueba.java" en tu carpeta de trabajo, escribe dentro el código del ejemplo anterior que escribe "Hola" por pantalla en Java, y compílalo desde el terminal, escribiendo este comando desde la misma carpeta donde esté el archivo fuente:

```
javac Prueba.java
```

> Se habrá creado en la misma carpeta un archivo llamado "Prueba.class", que es el archivo compilado para la máquina virtual Java. Para ejecutar este programa, debes escribir este comando (también desde la misma carpeta donde esté el archivo compilado):

```
java Prueba
```

> Deberás ver por pantalla el mensaje "Hola", indicando que todo ha ido correctamente.

1.1.3. Algunos lenguajes populares

Veamos a continuación las características principales de algunos de los lenguajes más populares en la actualidad, para hacernos una idea de su origen, repercusión y posibilidades.

1.1.3.1. El lenguaje C

Si tuviéramos que ponerle una etiqueta al lenguaje C probablemente sería la de *el lenguaje*, ya que sin duda fue el precursor de muchos de los lenguajes que surgieron después. Es una evolución de un lenguaje previo, llamado B, pero que no alcanzó la misma popularidad.

Antes de este lenguaje, se solía programar a base de tarjetas perforadas u otros dispositivos físicos o, en el mejor de los casos, empleando lenguajes de bajo nivel, difíciles de entender, escribir y mantener.

Así, puede entenderse que la aparición del lenguaje C en torno a 1970 fuese todo un *boom*. A pesar de ser un lenguaje con estructuras que lo hacen de alto nivel, también posee algunas características que lo acercan al bajo nivel, como el control o acceso a posiciones de memoria mediante punteros.

En la actualidad, no se utiliza demasiado para el desarrollo de aplicaciones, pero sí para la implementación de sistemas operativos, librerías, o herramientas como compiladores de otros lenguajes. Es muy apreciado por la eficiencia del código que produce.

1.1.3.2. C++

Como siguiente paso evolutivo al lenguaje C anterior estaría C++, que apareció a finales de los años 70, y que permitió extender el lenguaje C para trabajar con clases y objetos, siendo por tanto un lenguaje *híbrido* (permitía tanto la programación orientada a objetos como la tradicional basada en C).

En la actualidad, quizá su principal ámbito de aplicación es el mundo de los videojuegos, tanto para desarrollar motores de videojuegos (Unreal Engine, por ejemplo), como para programar videojuegos empleando diversas librerías (la propia Unreal, Cocos2D, SDL, etc.). Esto es así debido a que, al ser una extensión del lenguaje C, continúa disponiendo de mecanismos de bajo nivel que dan un acceso rápido al hardware (la tarjeta gráfica, en este caso), y porque la orientación a objetos y la multitud de librerías disponibles hacen que sea un lenguaje muy rico y con muchas opciones para el desarrollo.

1.1.3.3. Java

Java es un lenguaje de programación orientado a objetos, nacido a principios de los años 90 a través de la empresa *Sun Microsystems*. Como hemos comentado antes, dispone de una máquina virtual propia que ejecuta sus programas compilados, lo que hace que los programas Java sean independientes de la plataforma sobre la que se están ejecutando, pudiendo hacerse en Linux, Windows, Mac y otros sistemas. Esto es así porque, inicialmente, el lenguaje se concibió para poder programar todo tipo de aparatos electrónicos, incluyendo electrodomésticos. Pero su popularidad fue tal que rápidamente se centró en el desarrollo de aplicaciones para ordenador.

Con Java y otros lenguajes de esa época se empieza a abandonar definitivamente esa "puerta abierta" al bajo nivel, y podemos hablar ya de lenguajes totalmente de alto nivel. El acceso a la memoria y a dispositivos físicos ya está mucho más limitado, y el propio lenguaje proporciona, además, otras herramientas como el *garbage collector* (recolector de basura), que se encarga de revisar la memoria y eliminar los elementos que ya no estemos utilizando, ahorrando así al programador esa tarea, tan necesaria en lenguajes previos como C o C++.

El ámbito de aplicación del lenguaje Java hoy en día es bastante amplio. Se desarrollan tanto aplicaciones de escritorio (empleando librerías gráficas como Swing o JavaFX), como aplicaciones móviles (Android utiliza lenguaje Java), webs (empleando lenguaje JSP y servlets), etc.

1.1.3.4. C#

El lenguaje C# apareció en torno al año 2000, como una nueva extensión de sus hermanos mayores C y C++. Sin embargo, también bebe de otros lenguajes previos muy populares, como por ejemplo, y sobre todo, Java. Si examináis estos dos lenguajes, podéis comprobar que su sintaxis es muy similar, y por tanto aprender uno de ellos es muy sencillo habiendo aprendido el otro.

También adaptó de Java el hecho de disponer de su propia máquina virtual, la plataforma *.NET*, que se encarga de ejecutar los programas C# en diferentes sistemas (de forma directa en sistemas Windows, y algo más elaborada en otros sistemas).

El mercado de C# hoy en día también es muy amplio: desde aplicaciones de escritorio con Windows Forms o WPF, hasta aplicaciones web (ASP .NET), videojuegos (Unity), etc.

1.1.3.5. Javascript

Abandonamos los lenguajes multipropósito como Java o C#, con los que podemos desarrollar muy diversos tipos de aplicaciones, para centrarnos en otro que, en principio, se concibió para el desarrollo web: Javascript. Este lenguaje interpretado apareció a mediados de los años 90, y debe su nombre a la popularidad del lenguaje Java, aunque no tienen ninguna relación entre sí.

Inicialmente se concebió como un lenguaje de programación web en el lado cliente, es decir, en el navegador. De esta forma, se podía dar contenido dinámico a las páginas, quitar y poner elementos en la web sin tener que llamar al servidor. Con la tecnología AJAX se permitió, posteriormente, hacer llamadas al servidor desde las propias páginas del cliente, obtener el resultado y recargar únicamente la parte de la página que lo necesitara, sin tener que recargar toda la página por completo.

En los últimos años, Javascript ha dado pasos gigantescos en su nicho de mercado, y ya no sólo se emplea para desarrollar aplicaciones web en el lado del cliente. También podemos desarrollarlas en el lado servidor, equiparándose a lenguajes como PHP, JSP o ASP.NET, a través del framework Node.js. Y no sólo queda ahí la cosa: podemos desarrollar aplicaciones de escritorio gracias a frameworks como Electron, y aplicaciones móviles mediante tecnologías híbridas como Ionic.

1.1.3.6. PHP

PHP es otro de los lenguajes concebidos para desarrollo web, y que actualmente abarca la gran mayoría del desarrollo de aplicaciones web en entorno servidor. Concebido también a mediados de los años 90, su evolución no ha sido tan explosiva como la de Javascript, y su nicho de mercado se ha centrado en el desarrollo web. Pero su facilidad de uso y sus posibilidades lo convierten en el lenguaje preferido para esa tarea, desde hace varios años.

1.1.3.7. Otros lenguajes

Los lenguajes vistos hasta ahora son, quizá, los que más popularidad y cota de mercado tienen. Pero no podemos dejar de prestar atención a otros que tienen también su nicho de mercado, por diferentes motivos. Por ejemplo:

- **Python**, un lenguaje interpretado que tiene entre sus muchas ventajas la facilidad de aprendizaje (suele ser el lenguaje elegido por muchas instituciones educativas para enseñar a programar). El código resultante en este lenguaje suele ser mucho más compacto y directo que el de otros, para realizar una misma tarea. También se pueden desarrollar distintos tipos de scripts para sistemas operativos empleando Python, así como videojuegos (mediante librerías como *PyGame*) o webs en el lado servidor (mediante frameworks como Django).
- **Go** es un lenguaje inspirado en C, compilado, y desarrollado por Google en 2009. Permite tanto programación estructurada tradicional como orientada a objetos, e incorpora muchas facilidades de lenguajes como Python, lo que lo hacen a priori apetecible de aprender. Teniendo en cuenta la empresa que lo soporta, no es de extrañar que su importancia vaya en aumento.
- **Swift** es un lenguaje desarrollado por Apple en 2014, y que ha venido a sustituir al lenguaje Objective-C en el desarrollo de aplicaciones para el mundo Apple (Mac OSX e iOS), por lo que también se prevé, salvo que irrumpa otro lenguaje alternativo, que su importancia sea alta.

> Ejercicios propuestos:

> 1.1.3.1. Busca algún editor de código online que acepte diversos lenguajes. Aquí tienes un : Ideone. Averigua cómo escribir un programa que diga "Hola" en los lenguajes vistos en este apartado, o algún otro que se te ocurra. Anota tus soluciones en un archivo de texto y guárdalo.