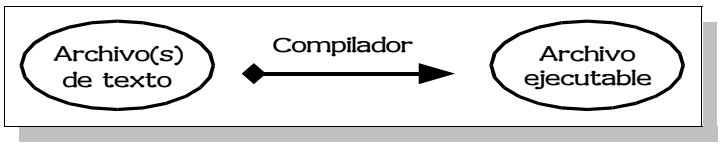
Conceptos generales

Creación de un programa

Simplificando un poco, se puede decir que los programas son **archivos ejecutables**. Internamente, están formados por órdenes para el microprocesador, que sólo éste entiende; las órdenes están en **código máquina**.

Crear programas directamente así es extremadamente difícil (aunque posible), por lo que se han inventado métodos que simplifican la tarea. Lo más habitual es escribir el programa usando un lenguaje que entiendan las personas y posteriormente usar otro programa para que haga la traducción a código máquina.

Los programadores escriben sus programas siguiendo unas determinadas reglas, y lo almacenan en uno o más archivos de texto, fácilmente manipulables. Estos archivos constituyen el **código fuente** del programa, o, sencillamente, *los fuentes*. Un programa llamado **compilador** examina el código fuente; si encuentra errores en la aplicación de las reglas, los indica; si no encuentra errores, produce el archivo ejecutable. Esquemáticamente, éste es el proceso:



Lenguajes de programación

Un lenguaje de programación es un conjunto de reglas que debe seguir el programador para escribir el código fuente de un programa. Existen miles de lenguajes distintos, aunque los más importantes no son más que unas decenas.

Nivel del lenguaje

Algunos lenguajes son más fáciles de entender por las personas y otros son más fáciles de entender por los compiladores. Atendiendo a esta característica se habla de lenguajes de programación de diferentes niveles:

- Lenguajes de **bajo nivel**. Difíciles de entender por los humanos, muy fáciles para los compiladores. El ejemplo más característico es el lenguaje **Ensamblador**. Estos lenguajes se utilizan en las partes más críticas de los programas, aquellas que deben ejecutarse con mucha rapidez y seguridad, ya que permiten manipular de modo muy cercano las características del microprocesador.
- Lenguajes de **medio nivel**. Permiten un acceso fácil a aspectos de bajo nivel y también crear estructuras de alto nivel. Estos lenguajes son muy versátiles y se utilizan muy ampliamente. Ejemplos típicos: **C** y **FORTH**. Con ellos se escriben sistemas operativos, software de comunicaciones, etc.
- Lenguajes de **alto nivel**. Permiten a los programadores concentrarse en aspectos muy generales y abstractos, despreocupándose de las características del microprocesador. Ejemplos: **C++**, **Pascal**, **BASIC**. Se utilizan para crear aplicaciones de tipo general, como procesadores de texto, hojas de cálculo, etc.

Lenguajes importantes

Además de los lenguajes citados anteriormente como ejemplos, que han sido elegidos por su relevancia, es conveniente conocer al menos los nombres de algunos otros lenguajes muy importantes:

- **FORTRAN**. Es un lenguaje científico, con el que hay escrito gran cantidad de software matemático crítico, usado en modelos de simulación, en control de sistemas, etc.
- **Cobol**. El típico lenguaje de programación usado en medios financieros; poco a poco está siendo desbancado por lenguajes más modernos, pero sigue funcionando gran cantidad de código escrito en este lenguaje.

Lenguajes interpretados

Existen algunos lenguajes de programación que no requieren ni de la aplicación directa del compilador ni de la creación de un ejecutable, sino que se pueden ejecutar directamente a partir del códi-

go fuente. Se llaman lenguajes interpretados por la presencia de un componente llamado **intérprete** que es el que se encarga de traducir a código máquina el código fuente, cosa que hace cada vez que se ejecuta el programa. Estos lenguajes resultan muy cómodos de utilizar y son especialmente buenos en tareas que no requieren gran capacidad de cálculo pero sí continuas adaptaciones. Como ejemplo de lenguajes interpretados hay que citar **Perl** y **Python**.

Partes de un programa

Simplificando mucho, se puede pensar que un programa consta de dos partes: la interfaz de usuario y las manipulaciones de datos. Con el interfaz de usuario se pide a la persona que usa el programa los datos de partida y se le entrega el resultado de las operaciones. La parte interna del programa procesa los datos iniciales para obtener las soluciones. Es fácil separar las dos partes, y es una buena práctica de programación hacerlo así.

Métodos de programación

Además de un lenguaje de programación, se suele seguir un determinado método escribiendo los programas. Muchas veces el lenguaje determina el método, pero casi siempre son independientes. Los métodos y los lenguajes han ido evolucionando juntos en la breve historia de la informática (poco más de cincuenta años).

Primeros métodos

Los primeros lenguajes eran muy simples y no permitían mucha flexibilidad. Los programas eran muy propensos a errores y muy difíciles de modificar. El lenguaje Ensamblador, FORTH y los primeros BASIC, son buenos ejemplos de ello. Seguir el funcionamiento del código era tan difícil como seguir el rastro de un hilo de *spaghetti* en un plato lleno de ellos. Precisamente se denomina *código spaghetti* al código que se solía escribir así.

Programación estructurada

Para poner orden en el código se inventaron lenguajes como **ALGOL**, que permitieron escribir mejor código en menos tiempo. En la programación estructurada se prohiben los saltos de una parte a otra del código, y esto supone una gran ventaja. Prácticamente todos los lenguajes modernos son estructurados.

Programación orientada al objeto

Una abstracción muy importante fue unir en un todo, llamado objeto, un conjunto de datos y los métodos necesarios para manipularlos. Esta abstracción permitió la creación más sencilla de software reutilizable, es decir, que se escribe una vez para un programa y se puede usar posteriormente en otros programas. El primer lenguaje así fue **Smalltalk**, los lenguajes **Ada** y **Eiffel** aportaron buenas ideas, apareció el C++, que incorpora a la potencia del C la facilidad del nuevo método, y en estos momentos los lenguajes de moda son **C#** y **Java**. (El simpático logotipo de Java es *Duke*).

Programación visual

Crear buenas interfaces de usuario consume mucho tiempo y sin embargo es muy fácil de estandarizar. Existen muchas herramientas que permiten crear fácilmente estas interfaces; se suelen llamar herramientas **visuales**. De momento, son inútiles para crear la parte fundamental de los programas, la que transforma los datos. Son muy populares las herramientas Visual Basic, Delphi, Kylix, Visual C++ y KDevelop.

Las librerías

Cuando un equipo de programación acomete un nuevo proyecto, no tiene por qué volver a programar todo por primera vez, sino que puede (y debe) utilizar partes que ya hayan sido creadas. En inglés se llaman *librarys* a las partes ya programadas que se pueden incorporar a un programa nuevo. Aunque la traducción correcta de este término inglés es “biblioteca”, lo cierto es que la mala traducción “librería” ha resultado ser la que se ha incorporado al lenguaje informático español.

Existen gran cantidad de librerías disponibles. Los lenguajes ya incorporan las suyas propias, imprescindibles para la creación de cualquier programa por sencillo que sea. Pero también existen enormes librerías que dan resueltos multitud de problemas. Un buen programador debe conocer tanto los lenguajes como las librerías disponibles.