1.8. EL *SOFTWARE* (LOS PROGRAMAS)

El *software* de una computadora es un conjunto de instrucciones de programa detalladas que controlan y coordinan los componentes *hardware* de una computadora y controlan las operaciones de un sistema informático. El auge de las computadoras el siglo pasado y en el actual siglo XXI, se debe esencialmente al desarrollo de sucesivas generaciones de *software* potentes y cada vez más *amistosas* ("fáciles de utilizar").

Las operaciones que debe realizar el *hardware* son especificadas por una lista de instrucciones, llamadas **programas**, o *software*. Un programa de *software* es un conjunto de **sentencias** o **instrucciones** a la computadora. El proceso de escritura o codificación de un programa se denomina **programación** y las personas que se especializan en esta actividad se denominan **programadores**. Existen dos tipos importantes de *software*: *software* del sistema y *software* de aplicaciones. Cada tipo realiza una función diferente.

El **software del sistema** es un conjunto generalizado de programas que gestiona los recursos de la computadora, tal como el procesador central, enlaces de comunicaciones y dispositivos periféricos. Los programadores que escriben *software* del sistema se llaman **programadores de sistemas**. El **software de aplicaciones** es el conjunto de programas escritos por empresas o usuarios individuales o en equipo y que instruyen a la computadora para que ejecute una tarea específica. Los programadores que escriben *software* de aplicaciones se llaman **programadores de aplicaciones**.

Los dos tipos de *software* están relacionados entre sí, de modo que los usuarios y los programadores pueden hacer así un uso eficiente de la computadora. En la Figura 1.12 se muestra una vista organizacional de una computadora donde se ven los diferentes tipos de *software* a modo de capas de la computadora desde su interior (el *hardware*) hasta su exterior (usuario). Las diferentes capas funcionan gracias a las instrucciones específicas (instrucciones máquina) que forman parte del *software* del sistema y llegan al *software* de aplicación, programado por los programadores de aplicaciones, que es utilizado por el usuario que no requiere ser un especialista.

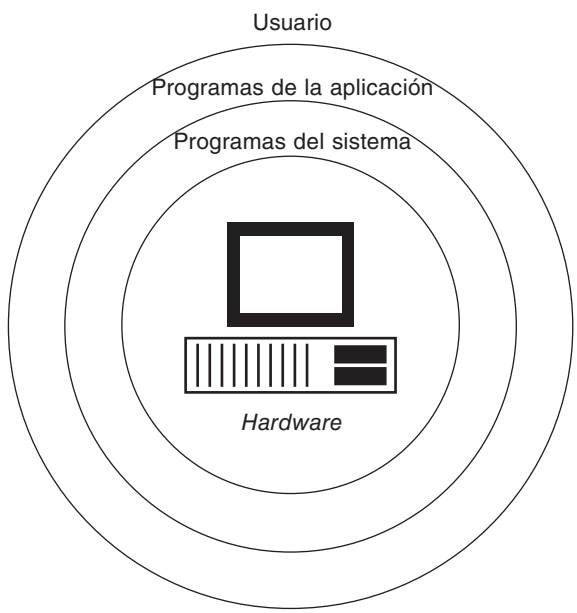

Figura 1.12. Relación entre programas de aplicación y programas del sistema.

1.8.1. Software del sistema

El *software* del sistema coordina las diferentes partes de un sistema de computadora y conecta e interactúa entre el *software* de aplicación y el *hardware* de la computadora. Otro tipo de *software* del sistema que gestiona, controla las actividades de la computadora y realiza tareas de proceso comunes, se denomina *utility* o **utilidades** (en algunas partes de Latinoamérica, **utilerías**). El *software* del sistema que gestiona y controla las actividades de la computadora se denomina **sistema operativo**. Otro *software* del sistema son los **programas traductores** o de traducción de lenguajes de computadora que convierten los lenguajes de programación, entendibles por los programadores, en lenguaje máquina que entienden las computadoras.

El ***software* del sistema** es el conjunto de programas indispensables para que la máquina funcione; se denominan también *programas del sistema*. Estos programas son, básicamente, *el sistema operativo, los editores de texto,* los *compiladores/intérpretes* (lenguajes de programación) y los *programas de utilidad*.

1.8.2. Software de aplicación

El *software* de aplicación tiene como función principal asistir y ayudar a un usuario de una computadora para ejecutar tareas específicas. Los programas de aplicación se pueden desarrollar con diferentes lenguajes y herramientas de *software*. Por ejemplo, una aplicación de procesamiento de textos (*word processing*) tal como Word o Word Perfect que ayuda a crear documentos, una hoja de cálculo tal como Lotus 1-2-3 o Excel que ayudan a automatizar tareas tediosas o repetitivas de cálculos matemáticos o estadísticos, a generar diagramas o gráficos, presentaciones visuales como PowerPoint, o a crear bases de datos como Access u Oracle que ayudan a crear archivos y registros de datos.

Los usuarios, normalmente, compran el *software* de aplicaciones en discos CD o DVD (antiguamente en disquetes) o los descargan (*bajan*) de la Red Internet y han de instalar el *software* copiando los programas correspondientes de los discos en el disco duro de la computadora. Cuando compre estos programas asegúrese de que son compatibles con su computadora y con su sistema operativo. Existe una gran diversidad de programas de aplicación para todo tipo de actividades tanto de modo personal, como de negocios, navegación y manipulación en Internet, gráficos y presentaciones visuales, etc.

Los *lenguajes de programación* sirven para escribir programas que permitan la comunicación usuario/máquina. Unos programas especiales llamados *traductores* (**compiladores** o **intérpretes**) convierten las instrucciones escritas en lenguajes de programación en instrucciones escritas en lenguajes máquina (0 y 1, *bits*) que ésta pueda entender.

Los *programas de utilidad*[22] facilitan el uso de la computadora. Un buen ejemplo es un *editor de* textos que permite la escritura y edición de documentos. Este libro ha sido escrito en un editor de textos o *procesador de palabras* (***"word procesor"***).

Los programas que realizan tareas concretas, nóminas, contabilidad, análisis estadístico, etc., es decir, los programas que podrá escribir en C, se denominan *programas de aplicación*. A lo largo del libro se verán pequeños programas de aplicación que muestran los principios de una buena programación de computadora.

Se debe diferenciar entre el acto de crear un programa y la acción de la computadora cuando ejecuta las instrucciones del programa. La creación de un programa se hace inicialmente en papel y a continuación se introduce en la computadora y se convierte en lenguaje entendible por la computadora. La ejecución de un programa requiere una aplicación de una entrada (*datos*) al programa y la obtención de una salida (*resultados*). La entrada puede tener una variedad de formas, tales como números o caracteres alfabéticos. La salida puede también tener formas, tales como datos numéricos o caracteres, señales para controlar equipos o robots, etc. (Figura 1.13).

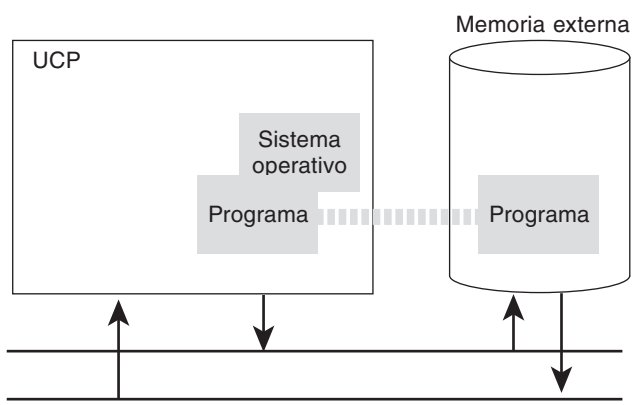

Figura 1.13. Ejecución de un programa.

[22] *Utility*: programa de utilidad o utilitería.

1.8.3. Sistema operativo

Un **sistema operativo SO** (*Operating System, OS*) es tal vez la parte más importante del software del sistema y es el software que controla y gestiona los recursos de la computadora. En la práctica el sistema operativo es la colección de programas de computadora que controla la interacción del usuario y el *hardware* de la computadora. El sistema operativo es el administrador principal de la computadora, y por ello a veces se la compara con el director de una orquesta ya que este software es el responsable de dirigir todas las operaciones de la computadora y gestionar todos sus recursos.

El sistema operativo asigna recursos, planifica el uso de recursos y tareas de la computadora, y monitoriza las actividades del sistema informático. Estos recursos incluyen memoria, dispositivos de E/S (Entrada/Salida), y la UCP (Unidad Central de Proceso). El sistema operativo proporciona servicios tales como asignar memoria a un programa y manipulación del control de los dispositivos de E/S tales como el monitor, el teclado o las unidades de disco. La Tabla 1.9 muestra algunos de los sistemas operativos más populares utilizados en enseñanza y en informática profesional.

Tabla 1.9. Sistemas operativos más utilizados en educación y en la empresa

Sistema operativo	*Características*
Windows Vista	Nuevo sistema operativo de Microsoft presentado a comienzos del año 2007.
Windows XP	Sistema operativo más utilizado en la actualidad, tanto en el campo de la enseñanza, como en la industria y negocios. Su fabricante es Microsoft.
Windows 98/ME/2000	Versiones anteriores de Windows pero que todavía hoy son muy utilizados.
UNIX	Sistema operativo abierto, escrito en C y todavía muy utilizado en el campo profesional.
Linux	Sistema operativo de software abierto, gratuito y de libre distribución, similar a UNIX, y una gran alternativa a Windows. Muy utilizado actualmente en servidores de aplicaciones para Internet.
Mac OS	Sistema operativo de las computadoras Apple Macintosh.
DOS y OS/2	Sistemas operativos creados por Microsoft e IBM respectivamente, ya poco utilizados pero que han sido la base de los actuales sistemas operativos.
CP/M	Sistema operativo de 8 bits para las primeras microcomputadoras nacidas en la década de los setenta.
Symbian	Sistema operativo para teléfonos móviles apoyado fundamentalmente por el fabricante de teléfonos celulares Nokia.
PalmOS	Sistema operativo para agendas digitales, PDA; del fabricante Palm.
Windows Mobile, CE	Sistema operativo para teléfonos móviles con arquitectura y apariencias similares a Windows XP. Las últimas versiones son: 5.0 y 6.0.

Cuando un usuario interactúa con una computadora, la interacción está controlada por el sistema operativo. Un usuario se comunica con un sistema operativo a través de una interfaz de usuario de ese sistema operativo. Los sistemas operativos modernos utilizan una **interfaz gráfica de usuario, IGU** (*Graphical User Interface*, **GUI**) que hace uso masivo de iconos, botones, barras y cuadros de diálogo para realizar tareas que se controlan por el teclado o el ratón (mouse), entre otros dispositivos.

Normalmente el sistema operativo se almacena de modo permanente en un chip de memoria de sólo lectura (ROM), de modo que esté disponible tan pronto la computadora se pone en marcha ("se enciende" o "se prende"). Otra parte del sistema operativo puede residir en disco, que se almacena en memoria RAM en la inicialización del sistema por primera vez en una operación que se llama *carga* del sistema (*booting*).

El sistema operativo dirige las operaciones globales de la computadora, instruye a la computadora para ejecutar otros programas y controla el almacenamiento y recuperación de archivos (programas y datos) de cintas y discos. Gracias al sistema operativo es posible que el programador pueda introducir y grabar nuevos programas, así como instruir a la computadora para que los ejecute. Los sistemas operativos pueden ser: *monousuarios* (un solo usuario) y *multiusuarios*, o tiempo compartido (diferentes usuarios), atendiendo al número de usuarios y *monocarga* (una sola tarea) o *multitarea* (múltiples tareas) según las tareas (procesos) que puede realizar simultáneamente.

Windows Vista

El 30 de enero de 2007, Microsoft presentó a nivel mundial su nuevo sistema operativo **Windows Vista**. Esta nueva versión en la que Microsoft llevaba trabajando desde hacía cinco años, en que presentó su hasta ahora, última versión, **Windows XP**, es un avance significativo en la nueva generación de sistemas operativos que se utilizarán en la próxima década.

Windows Vista contiene numerosas características nuevas y muchas otras actualizadas, algunas de las cuales son: una interfaz gráfica de usuario muy amigable, herramientas de creación de multimedia, potentes herramientas de comunicación entre computadoras, etc. También ha incluido programas que hasta el momento de su lanzamiento se comercializaban independientemente tales como programas de reproducción de música, vídeo, accesos a Internet, etc. Es de destacar que Vista ha mejorado notablemente la seguridad en el sistema operativo, ya que Windows XP y sus predecesores han sido muy vulnerables a virus, malware, y otros ataques a la seguridad del sistema y del usuario.

Existen cinco versiones comerciales: *Home Basic, Home Premium, Business, Ultimate* y *Enterprise*. Los requisitos que debe tener su computadora dependerá de la versión elegida y variará desde la más básica, recomendada para usuarios domésticos (512 MB de RAM mínima, procesador de 32 bits (x86) o de 64 bits (x64) a 1 GHz, 15 GB de espacio disponible en el disco duro, etc.) a *Ultimate* que incorpora todas las funcionalidades y ventajas contenidas en las demás versiones (ya se requiere al menos 1 GB de memoria, mayor capacidad de disco duro, etc.). A nivel de empresas y grandes corporaciones se recomienda *Enterprise,* diseñada para reducir los riesgos de seguridad y los enormes costes de este tipo de infraestructuras.

Tipos de sistemas operativos

Las diferentes características especializadas del sistema operativo permiten a las computadoras manejar muchas tareas diferentes, así como múltiples usuarios de modo simultáneo o en paralelo, bien de modo secuencial. En función de sus características específicas los sistemas operativos se pueden clasificar en varios grupos.

1.8.3.1. *Multiprogramación/Multitarea*

La multiprogramación permite a múltiples programas compartir recursos de un sistema de computadora en cualquier momento a través del uso concurrente una UCP. Sólo un programa utiliza realmente la UCP en cualquier momento dado, sin embargo las necesidades de entrada/salida pueden ser atendidas en el mismo momento. Dos o más programas están activos al mismo tiempo, pero no utilizan los recursos de la computadora simultáneamente. Con multiprogramación, un grupo de programas se ejecutan alternativamente y se alternan en el uso del procesador. Cuando se utiliza un sistema operativo de un único usuario, la multiprogramación toma el nombre de **multitarea**.

Multiprogramación

Método de ejecución de dos o más programas concurrentemente utilizando la misma computadora. La UCP ejecuta sólo un programa pero puede atender los servicios de entrada/salida de los otros al mismo tiempo.

1.8.3.2. *Tiempo compartido (múltiples usuarios,* time sharing*)*

Un sistema operativo multiusuario es un sistema operativo que tiene la capacidad de permitir que muchos usuarios compartan simultáneamente los recursos de proceso de la computadora. Centenas o millares de usuarios se pueden conectar a la computadora que asigna un tiempo de computador a cada usuario, de modo que a medida que se libera la tarea de un usuario, se realiza la tarea del siguiente, y así sucesivamente. Dada la alta velocidad de transferencia de las operaciones, la sensación es de que todos los usuarios están conectados simultáneamente a la UCP con cada usuario recibiendo únicamente un tiempo de máquina.

1.8.3.3. *Multiproceso*

Un sistema operativo trabaja en multiproceso cuando puede enlazar dos o más UCP para trabajar en paralelo en un único sistema de computadora. El sistema operativo puede asignar múltiples UCP para ejecutar diferentes instrucciones del mismo programa o de programas diferentes simultáneamente, dividiendo el trabajo entre las diferentes UCP.

La multiprogramación utiliza proceso concurrente con una UCP; el multiproceso utiliza proceso simultáneo con múltiples UCP.

1.9. LENGUAJES DE PROGRAMACIÓN

Como se ha visto en el apartado anterior, para que un procesador realice un proceso se le debe suministrar en primer lugar un algoritmo adecuado. El procesador debe ser capaz de *interpretar* el algoritmo, lo que significa:

- comprender las instrucciones de cada paso,
- realizar las operaciones correspondientes.

Cuando el procesador es una computadora, el algoritmo se ha de expresar en un formato que se denomina *programa,* ya que el pseudocódigo o el diagrama de flujo no son comprensibles por la computadora, aunque pueda entenderlos cualquier programador. Un programa se escribe en un *lenguaje de programación* y las operaciones que conducen a expresar un algoritmo en forma de programa se llaman *programación*. Así pues, los lenguajes utilizados para escribir programas de computadoras son los *lenguajes de programación* y **programadores** son los escritores y diseñadores de programas. El proceso de traducir un algoritmo en *pseudocódigo* a un lenguaje de programación se denomina **codificación**, y el algoritmo escrito en un lenguaje de programación se denomina **código fuente**.

En la realidad la computadora no entiende directamente los lenguajes de programación sino que se requiere un programa que traduzca el código fuente a otro lenguaje que sí entiende la máquina directamente, pero muy complejo para las personas; este lenguaje se conoce como **lenguaje máquina** y el código correspondiente **código máquina**. Los programas que traducen el código fuente escrito en un lenguaje de programación —tal como C++— a código máquina se denominan **traductores**. El proceso de conversión de un algoritmo escrito en pseudocódigo hasta un programa ejecutable comprensible por la máquina, se muestra en la Figura 1.14.

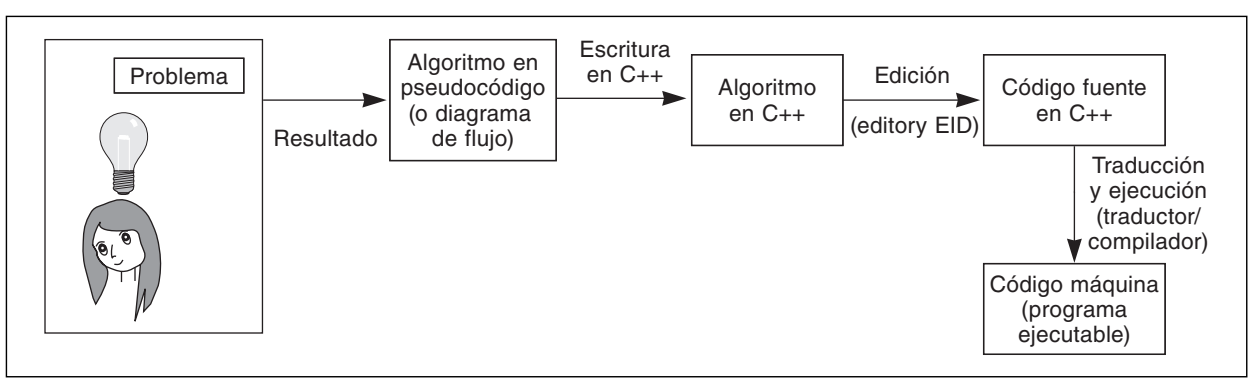

Figura 1.14. Proceso de transformación de un algoritmo en pseudocódigo en un programa ejecutable.

Hoy en día, la mayoría de los programadores emplean lenguajes de programación como C++, C, C#, Java, Visual Basic, XML, HTML, Perl, PHP, JavaScript..., aunque todavía se utilizan, sobre todo profesionalmente, los clásicos *COBOL*, *FORTRAN*, *Pascal* o el mítico BASIC. Estos lenguajes se denominan **lenguajes de alto nivel** y permiten a los profesionales resolver problemas convirtiendo sus algoritmos en programas escritos en alguno de estos lenguajes de programación.

Los **lenguajes de programación** se utilizan para escribir programas. Los programas de las computadoras modernas constan de secuencias de instrucciones que se codifican como secuencias de dígitos numéricos que podrán entender dichas computadoras. El sistema de codificación se conoce como **lenguaje máquina** que es el lenguaje nativo de una computadora. Desgraciadamente la escritura de programas en lenguaje máquina es una tarea tediosa y difícil ya que sus instrucciones son secuencias de 0 y 1 (*patrones de bit*, tales como 11110000, 01110011...) que son muy difíciles de recordar y manipular por las personas. En consecuencia, se necesitan lenguajes de programación "amigables con el programador" que permitan escribir los programas para poder "charlar" con facilidad

con las computadoras. Sin embargo, las computadoras sólo entienden las instrucciones en lenguaje máquina, por lo que será preciso traducir los programas resultantes a lenguajes de máquina antes de que puedan ser ejecutadas por ellas.

Cada lenguaje de programación tiene un conjunto o "juego" de instrucciones (acciones u operaciones que debe realizar la máquina) que la computadora podrá entender directamente en su código máquina o bien se traducirán a dicho código máquina. Las instrucciones básicas y comunes en casi todos los lenguajes de programación son:

- *Instrucciones de entrada/salida.* Instrucciones de transferencia de información entre dispositivos periféricos y la memoria central, tales como `"leer de..."` o bien `"escribir en..."`.
- *Instrucciones de cálculo.* Instrucciones para que la computadora pueda realizar operaciones aritméticas.
- *Instrucciones de control.* Instrucciones que modifican la secuencia de la ejecución del programa.

Además de estas instrucciones y dependiendo del procesador y del lenguaje de programación existirán otras que conformarán el conjunto de instrucciones y junto con las reglas de sintaxis permitirán escribir los programas de las computadoras. Los principales tipos de lenguajes de programación son:

- *Lenguajes máquina.*
- *Lenguajes de bajo nivel (ensambladores).*
- *Lenguajes de alto nivel.*

Figura 1.15. Diferentes sistemas operativos: Windows Vista (izquierda) y Red Hat Enterprise Linux 4.

1.9.1. Traductores de lenguaje: el proceso de traducción de un programa

El proceso de traducción de un programa fuente escrito en un lenguaje de alto nivel a un lenguaje máquina comprensible por la computadora, se realiza mediante programas llamados "traductores". Los **traductores de lenguaje** son programas que traducen a su vez los programas fuente escritos en lenguajes de alto nivel a código máquina. Los traductores se dividen en **compiladores** e **intérpretes**.

Intérpretes

Un *intérprete* es un traductor que toma un programa fuente, lo traduce y, a continuación, lo ejecuta. Los programas intérpretes clásicos como BASIC, prácticamente ya no se utilizan, más que en circunstancias especiales. Sin embargo, está muy extendida la versión interpretada del lenguaje Smalltalk, un lenguaje orientado a objetos puro. El sistema de traducción consiste en: traducir la primera sentencia del programa a lenguaje máquina, se detiene la traducción, se ejecuta la sentencia; a continuación, se traduce la siguiente sentencia, se detiene la traducción, se ejecuta la sentencia y así sucesivamente hasta terminar el programa (Figura 1.16).

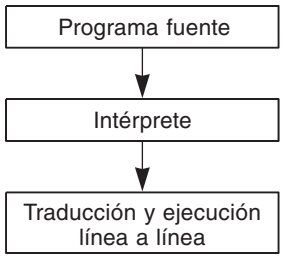

Figura 1.16. Intérprete.

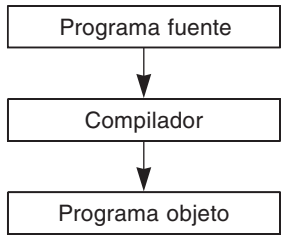

Figura 1.17. La compilación de programas.

Compiladores

Un *compilador* es un programa que traduce los programas fuente escritos en lenguaje de alto nivel a lenguaje máquina. La traducción del programa completo se realiza en una sola operación denominada **compilación** del programa; es decir, se traducen todas las instrucciones del programa en un solo bloque. El programa compilado y depurado (eliminados los errores del código fuente) se denomina *programa ejecutable* porque ya se puede ejecutar directamente y cuantas veces se desee; sólo deberá volver a compilarse de nuevo en el caso de que se modifique alguna instrucción del programa. De este modo el programa ejecutable no necesita del compilador para su ejecución. Los traductores de lenguajes típicos más utilizados son: **C**, **C++**, **Java**, **C#**, **Pascal**, **FORTRAN** y **COBOL** (Figura 1.17).

1.9.2. La compilación y sus fases

La *compilación* es el proceso de traducción de programas fuente a programas objeto. El programa objeto obtenido de la compilación ha sido traducido normalmente a código máquina.

Para conseguir el programa máquina real se debe utilizar un programa llamado *montador* o *enlazador* (*linker*). El proceso de montaje conduce a un programa en lenguaje máquina directamente ejecutable (Figura 1.18).

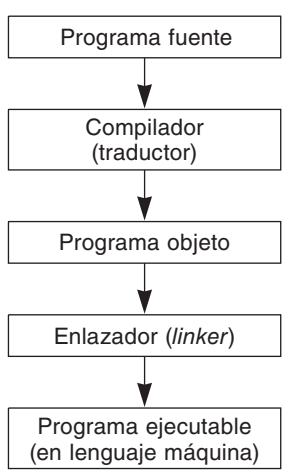

Figura 1.18. Fases de la compilación.

El proceso de ejecución de un programa escrito en un lenguaje de programación y mediante un compilador suele tener los siguientes pasos:

1. Escritura del *programa fuente* con un *editor* (programa que permite a una computadora actuar de modo similar a una máquina de escribir electrónica) y guardarlo en un dispositivo de almacenamiento (por ejemplo, un disco).
2. Introducir el programa fuente en memoria.

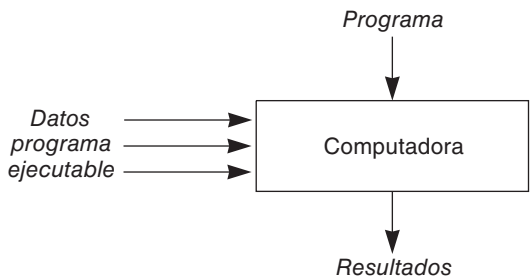

Figura 1.19. Ejecución de un programa.

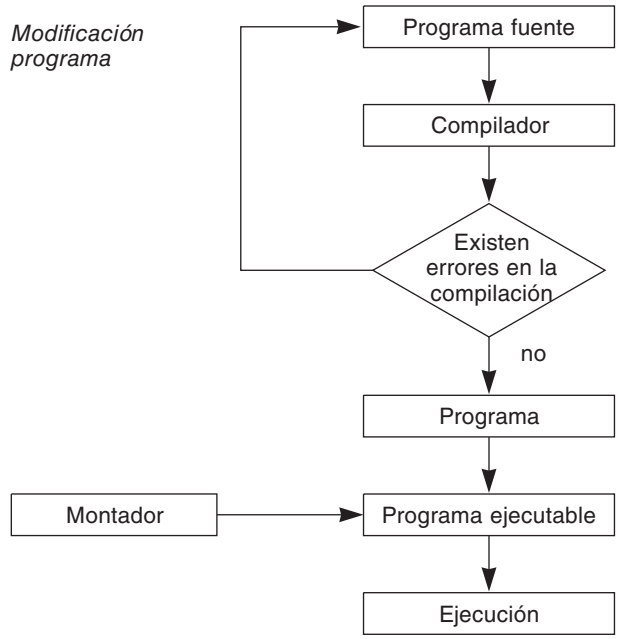

Figura 1.20. Fases de ejecución de un programa.

3. *Compilar* el programa con el compilador seleccionado.
4. *Verificar* y *corregir errores de compilación* (listado de errores).
5. Obtención del programa *objeto*.
6. El enlazador (*linker*) obtiene el *programa ejecutable*.
7. Se ejecuta el programa y, si no existen errores, se tendrá la salida del programa.

El proceso de ejecución se muestra en las Figuras 1.19 y 1.20.

1.9.3. Evolución de los lenguajes de programación

En la década de los cuarenta cuando nacían las primeras computadoras digitales el lenguaje que se utilizaba para programar era el *lenguaje máquina* que traducía directamente el código máquina (código binario) comprensible para las computadoras. Las instrucciones en lenguaje máquina dependían de cada computadora y debido a la dificultad de su escritura, los investigadores de la época simplificaron el proceso de programación desarrollando sistemas de notación en los cuales las instrucciones se representaban en formatos *nemónicos* (nemotécnicos) en vez de en formatos numéricos que eran más difíciles de recordar. Por ejemplo, mientras la instrucción

```
Mover el contenido del registro 4 al registro 8
```

se podía expresar en lenguaje máquina como

```
4048        o bien        0010 0000 0010 1000
```

en código *nemotécnico* podía aparecer como

```
MOV R5, R6
```

Para convertir los programas escritos en código nemotécnico a lenguaje máquina, se desarrollaron programas ensambladores (*assemblers*). Es decir, los **ensambladores** son programas que traducen otros programas escritos en código nemotécnico en instrucciones numéricas en lenguaje máquina que son compatibles y legibles por la máquina. Estos programas de traducción se llaman ensambladores porque su tarea es ensamblar las instrucciones reales de la

máquina con los nemotécnicos e identificadores que representan las instrucciones escritas en ensamblador. A estos lenguajes se les denominó de segunda generación, reservando el nombre de primera generación para los lenguajes de máquina.

En la década de los cincuenta y sesenta comenzaron a desarrollarse lenguajes de programación de tercera generación que diferían de las generaciones anteriores en que sus instrucciones o primitivas eran de alto nivel (comprensibles por el programador, como si fueran lenguajes naturales) e **independientes de la máquina**. Estos lenguajes se llamaron **lenguajes de alto nivel**. Los ejemplos más conocidos son **FORTRAN** (FORmula TRANslator) que fue desarrollado para aplicaciones científicas y de ingeniería, y **COBOL** (COmmon Business-Oriented Language), que fue desarrollado por la U.S. Navy de Estados Unidos, para aplicaciones de gestión o administración. Con el paso de los años aparecieron nuevos lenguajes tales como **Pascal**, **BASIC**, **C**, **C++**, **Ada**, **Java**, **C#**, **HTML**, **XML**...

Los lenguajes de programación de alto nivel se componen de un conjunto de instrucciones o primitivas más fáciles de escribir y recordar su función que los lenguajes máquina y ensamblador. Sin embargo, los programas escritos en un lenguaje de alto nivel, como C o Java necesitan ser traducidos a código máquina; para ello se requiere un programa denominado **traductor**. Estos programas de traducción se denominaron técnicamente, **compiladores**. De este modo existen compiladores de C, FORTRAN, Pascal, Java, etc.

También surgió una alternativa a los traductores compiladores como medio de implementación de lenguajes de tercera generación que se denominaron **intérpretes**[23]. Estos programas eran similares a los traductores excepto que ellos ejecutaban las instrucciones a medida que se traducían, en lugar de guardar la versión completa traducida para su uso posterior. Es decir, en vez de producir una copia de un programa en lenguaje máquina que se ejecuta más tarde (este es el caso de la mayoría de los lenguajes, C, C++, Pascal, Java...), un intérprete ejecuta realmente un programa desde su formato de alto nivel, instrucción a instrucción. Cada tipo de traductor tiene sus ventajas e inconvenientes, aunque hoy día prácticamente los traductores utilizados son casi todos compiladores por su mayor eficiencia y rendimiento.

Sin embargo, en el aprendizaje de programación se suele comenzar también con el uso de los lenguajes algorítmicos, similares a los lenguajes naturales, mediante instrucciones escritas en *pseudocódigo* (o *seudocógid*o) que son palabras o abreviaturas de palabras escritas en inglés, español, portugués, etc. Posteriormente se realiza la conversión al lenguaje de alto nivel que se vaya a utilizar realmente en la computadora, tal como C, C++ o Java. Esta técnica facilita la escritura de algoritmos como paso previo a la programación.

1.9.4. Paradigmas de programación

La evolución de los lenguajes de programación ha ido paralela a la idea de paradigma de programación: enfoques alternativos a los procesos de programación. En realidad un **paradigma de programación** representa fundamentalmente enfoques diferentes para la construcción de soluciones a problemas y por consiguiente afectan al proceso completo de desarrollo de software. Los paradigmas de programación clásicos son: *procedimental (o imperativo), funcional, declarativo y orientado a objetos.* En la Figura 1.21 se muestra la evolución de los paradigmas de programación y los lenguajes asociados a cada paradigma [BROOKSHEAR 04][24].

Lenguajes imperativos (procedimentales)

El **paradigma imperativo o procedimental** representa el enfoque o método tradicional de programación. Un lenguaje imperativo es un conjunto de instrucciones que se ejecutan una por una, de principio a fin, de modo secuencial excepto cuando intervienen instrucciones de salto de secuencia o control. Este paradigma define el proceso de programación como el desarrollo de una secuencia de órdenes (*comandos*) que manipulan los datos para producir los resultados deseados. Por consiguiente, el paradigma imperativo señala un enfoque del proceso de programación mediante la realización de un algoritmo que resuelve de modo manual el problema y a continuación expresa ese algoritmo como una secuencia de órdenes. En un lenguaje procedimental cada instrucción es una orden u órdenes para que la computadora realice alguna tarea específica.

---

[23] Uno de los intérpretes más populares en las décadas de los setenta y ochenta, fue **BASIC**.

[24] J. Glenn Brookshear, *Computer Science: An overview, Eigth edition*, Boston (EE.UU.): Pearson/Addison Wesley, 2005, p. 230. Obra clásica y excelente para la introducción a la informática y a las ciencias de la computación en todos sus campos fundamentales. Esta obra se recomienda a todos los lectores que deseen profundizar en los diferentes temas tratados en este capítulo y ayudará considerablemente al lector como libro de consulta en su aprendizaje en programación.

Los lenguajes de programación procedimentales, por excelencia, son **FORTRAN**, **COBOL**, **Pascal**, **BASIC**, **ALGOL**, **C** y **Ada** (aunque sus últimas versiones ya tienen un carácter completamente orientado a objetos).

Lenguajes declarativos

En contraste con el paradigma imperativo el **paradigma declarativo** solicita al programador que describa el problema en lugar de encontrar una solución algorítmica al problema; es decir, un lenguaje declarativo utiliza el principio del razonamiento lógico para responder a las preguntas o cuestiones consultadas. Se basa en la *lógica formal* y en el *cálculo de predicados de primer orden.* El razonamiento lógico se basa en la deducción. El lenguaje declarativo por excelencia es **Prolog**.

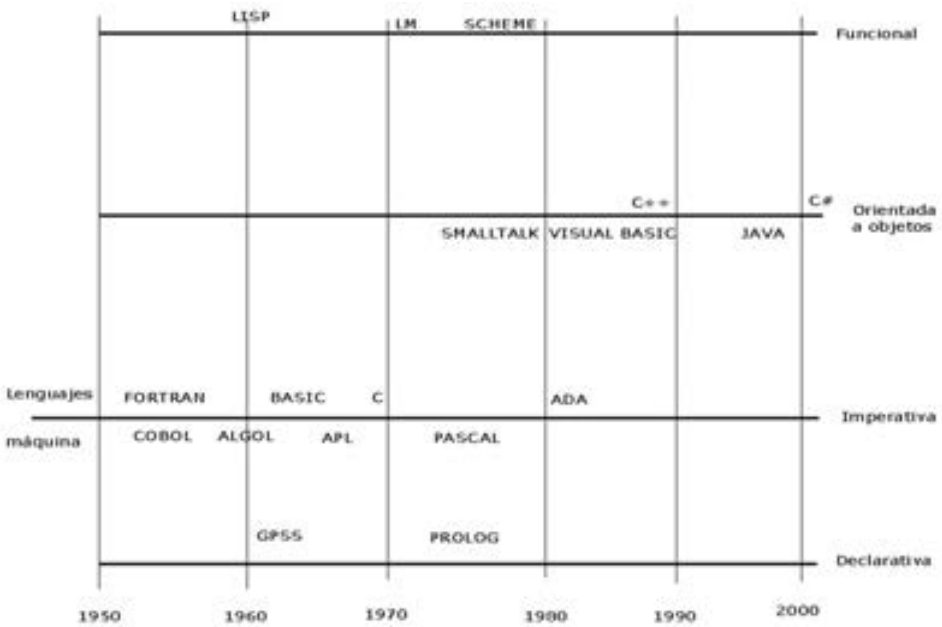

Figura 1.21. Paradigmas de programación (evolución de lenguajes).

Lenguajes orientados a objetos

El paradigma orientado a objetos se asocia con el proceso de programación llamado **programación orientada a objetos (POO)**[25] consistente en un enfoque totalmente distinto al proceso procedimental. El enfoque orientado a objetos guarda analogía con la vida real. El desarrollo de software OO se basa en el diseño y construcción de objetos que se componen a su vez de datos y operaciones que manipulan esos datos. El programador define en primer lugar los objetos del problema y a continuación los datos y operaciones que actuarán sobre esos datos. Las ventajas de la programación orientada a objetos se derivan esencialmente de la estructura modular existente en la vida real y el modo de respuesta de estos módulos u objetos a mensajes o eventos que se producen en cualquier instante.

Los orígenes de la POO se remontan a los Tipos Abstractos de Datos como parte constitutiva de una estructura de datos. En este libro se dedicará un capítulo completo al estudio del **TAD** como origen del concepto de programación denominado **objeto**.

C++ lenguaje orientado a objetos, por excelencia, es una extensión del lenguaje C y contiene las tres propiedades más importantes: *encapsulamiento, herencia y polimorfismo.* Smalltalk es otro lenguaje orientado a objetos muy potente y de gran impacto en el desarrollo del software orientado a objetos que se ha realizado en las últimas décadas.

Hoy día **Java** y **C#** son herederos directos de C++ y C, y constituyen los lenguajes orientados a objetos más utilizados en la industria del software del siglo XXI. **Visual Basic** y **VB.Net** son otros lenguajes orientados a objetos, no tan potentes como los anteriores pero extremadamente sencillos y fáciles de aprender.

---

[25] Si desea profundizar en este tipo de programación existen numerosos y excelentes libros que puede consultar en la Bibliografía.

1.10. BREVE HISTORIA DE LOS LENGUAJES DE PROGRAMACIÓN

La historia de la computación ha estado asociada indisolublemente a la aparición y a la historia de lenguajes de programación de computadoras[26]. La Biblia de los lenguajes ha sido una constante en el desarrollo de la industria del software y en los avances científicos y tecnológicos. Desde el año 1642 en que Blaise Pascal, inventó *La Pascalina*, una máquina que ayudaba a contar mediante unos dispositivos de ruedas, se han sucedido numerosos inventos que han ido evolucionando, a medida que se programaban mediante códigos de máquina, lenguajes ensambladores, hasta llegar a los lenguajes de programación de alto nivel en los que ya no se dependía del hardware de la máquina sino de la capacidad de abstracción del programador y de la sintaxis, semántica y potencia del lenguaje.

En la década de los cincuenta, IBM diseñó el primer lenguaje de programación comercial de alto nivel y concebido para resolver problemas científicos y de ingeniería (FORTRAN, 1954). Todavía hoy, muchos científicos e ingenieros siguen utilizando FORTRAN en sus versiones más recientes FORTRAN 77 y FORTRAN 90. En 1959, la doctora y almirante, Grace Hopper, lideró el equipo que desarrolló COBOL, el lenguaje por excelencia del mundo de la gestión y de los negocios hasta hace muy poco tiempo; aunque todavía el mercado sigue demandando programadores de COBOL ya que numerosas aplicaciones comerciales siguen corriendo en este lenguaje.

Una enumeración rápida de lenguajes de programación que han sido o son populares y los años en que aparecieron es la siguiente:

Década 50	Década 60	Década 70	Década 80	Década 90	Década 00
FORTRAN (1954)	BASIC (1964)	Pascal (1970)	C++ (1983)	Java (1997)	C# (2000)
ALGOL 58 (1958)	LOGO (1968)	C (1971)	Eiffel (1986)		
LISP (1958)	Simula 67 (1967)	Modula 2 (1975)	Perl (1987)		
COBOL (1959)	Smalltalk (1969)	Ada (1979)			

Programación de la Web

Si después o en paralelo de su proceso de aprendizaje en fundamentos y metodología de la programación desea practicar no sólo con un lenguaje tradicional como Pascal, C, C++, Java o C#, sino introducirse en lenguajes de programación para la Web, enumeramos a continuación los más empleados en este campo.

Los programadores pueden utilizar una amplia variedad de lenguajes de programación, incluyendo C y C++ para escribir aplicaciones Web. Sin embargo, algunas herramientas de programación son, particularmente, útiles para desarrollar aplicaciones Web:

- **HTML,** técnicamente es un lenguaje de descripción de páginas más que un lenguaje de programación. Es el elemento clave para la programación en la Web.
- **JavaScript**, es un lenguaje interpretado de guionado (scripting) que facilita a los diseñadores de páginas Web añadir guiones a páginas Web y modos para enlazar **esas páginas**.
- **VBScript**, la respuesta de Microsoft a JavaScript basada en VisualBasic.
- **Java,** lenguaje de programación, por excelencia, de la Web.
- **ActiveX**, lenguaje de Microsoft para simular a algunas de las características de Java.
- **C#,** el verdadero competidor de Java y creado por Microsoft.
- **Perl,** lenguaje interpretado de guionado (scripting) idóneo para escritura de texto.
- **XML**, lenguaje de marcación que resuelve todas las limitaciones de HTML y ha sido el creador de una nueva forma de programar la Web. Es el otro gran lenguaje de la Web.
- **AJAX,** es el futuro de la Web. Es una mezcla de JavaScript y XML. Es la espina dorsal de la nueva generación Web 2.0.

[26] Si desea una breve historia pero más detallada de los lenguajes de programación más utilizados por los programadores profesionales tanto para aprendizaje como para el desarrollo profesional puede consultarlo en la página web del libro: `www.mhe.es/joyanes`.

En el sitio Web de la editorial O'Reilly puede descargarse un póster (en PDF) con una magnífica y fiable Historia de los Lenguajes de Programación: `www.oreilly.com/news/graphics/prog_lang_poster.pdf`

RESUMEN

Una computadora es una máquina para procesar información y obtener resultados en función de unos datos de entrada.

Hardware: parte física de una computadora (dispositivos electrónicos).
Software: parte lógica de una computadora (programas).

Las computadoras se componen de:

- Dispositivos de Entrada/Salida (E/S).
- Unidad Central de Proceso (Unidad de Control y Unidad Lógica y Aritmética).
- Memoria central.
- Dispositivos de almacenamiento masivo de información (memoria auxiliar o externa).

El *software del sistema* comprende, entre otros, el sistema operativo **Windows**, **Linux**, en computadoras personales y los lenguajes de programación. Los *lenguajes de programación* de alto nivel están diseñados para hacer más fácil la escritura de programas que los lenguajes de bajo nivel. Existen numerosos lenguajes de programación cada uno de los cuales tiene sus propias características y funcionalidades, y normalmente son más fáciles de transportar a máquinas diferentes que los escritos en lenguajes de bajo nivel.

Los programas escritos en lenguaje de alto nivel deben ser traducidos por un compilador antes de que se puedan ejecutar en una máquina específica. En la mayoría de los lenguajes de programación se require un compilador para cada máquina en la que se desea ejecutar programas escritos en un lenguaje específico...

Los lenguajes de programación se clasifican en:

- *Alto nivel:* Pascal, FORTRAN, Visual Basic, C, Ada, Modula-2, C++, Java, Delphi, C#, etc.
- *Bajo nivel:* Ensamblador.
- *Máquina:* Código máquina.
- *Diseño de Web:* SMGL, HTML, XML, PHP...

Los programas traductores de lenguajes son:

- *Compiladores.*
- *Intérpretes.*