CAPITULO 1

INTRODUCCION A LA PROGRAMACION

1.1 Introducción. Objetivos.

La principal razón por la que las personas aprenden a programar, es para poder utilizar un ordenador como una herramienta para resolver sus problemas.

La Informática, como ciencia para el tratamiento de la información, ha evolucionado de forma espectacular en los últimos años. Esto ha traído consigo una revolución sociocultural que afecta a todos. La resolución de problemas que hace unos años eran largos y costosos o bien inabordables, ahora se han simplificado enormemente gracias a los ordenadores. El avance en la microelectrónica y la evolución de los sistemas operativos que controlan los dispositivos, han facilitado el uso de los ordenadores y el diseño de programas más eficaces.

Teniendo en cuenta las dos consideraciones anteriores, empezaremos esta asignatura con un primer capítulo que tiene como fin presentar los conceptos básicos de la programación, introduciendo los términos, entornos, materiales y finalidades del curso. Básicamente los objetivos que se pretenden son:

- El reconocimiento de los componentes de los sistemas de información.
- Conocer las etapas del ciclo de vida de una aplicación informática.
- La identificación de los elementos que constituyen o se relacionan con un programa.

1.2 Conceptos de ordenador y de sistema operativo.

Un **ordenador** o computador es un dispositivo electrónico que permite procesar información y obtener resultados. La información y los datos se introducen al ordenador por dispositivos de entrada, se almacenan en una memoria, y posteriormente se procesan para producir unos resultados que se presentan por dispositivos de salida.

En general un computador está compuesto de una parte material o física (**hardware**) y de una parte lógica o programable (**software**). El hardware lo componen todos los dispositivos y elementos materiales. El software lo componen los programas y aplicaciones diseñados y escritos para un ordenador.

Dentro del software, existe un tipo de software fundamental llamado **sistema operativo**. Éste permite controlar los dispositivos hardware, la ejecución de aplicaciones y la relación o comunicación de los usuarios con el ordenador. Las aplicaciones y los usuarios no tratan o se comunican con los dispositivos directamente, lo hacen a través del sistema operativo.

A nivel funcional, un ordenador está compuesto de elementos de proceso, almacenamiento, entrada y salida. Los primeros se encargan de controlar y ejecutar los programas. Los sistemas de almacenamiento permiten registrar en diferentes soportes información. Los elementos de entrada y salida se encargan de introducir y extraer información del ordenador.

1.3 Sistemas de proceso.

Históricamente, el hombre ha necesitado manejar grandes volúmenes de información, bien creándola o transmitiéndola. Por ello su empeño en crear máquinas y métodos para procesar dicha información. De esta

forma nace la **Informática** como ciencia encargada de este estudio y desarrollo. El fin de la Informática es ayudar a las personas en los trabajos rutinarios y repetitivos, generalmente de cálculo y gestión, así como la recogida de datos, elaboración de la información y distribución de la misma.

Existen muchas definiciones del término Informática. Una de ellas podría ser ésta: *"ciencia para el tratamiento racional de la información de forma automática"*. Los ordenadores son los elementos que realizan este tratamiento. Un **ordenador** se podría definir como una *"máquina que es capaz de recibir datos, procesarlos de forma automática y proporcionar unos resultados, gobernada por un programa"*. Vemos que el ordenador debe estar dirigido por lo que llamamos **programa**: *"conjunto de órdenes encaminadas a resolver un determinado problema"*.

A nivel físico y de forma muy general, un ordenador dispone de una Unidad Central de Proceso (CPU) encargada de capturar, analizar, descodificar y ejecutar las instrucciones que compone un programa. Para realizar estas tareas, la CPU dispone de una Unidad de Control (UC), la cual coordina las actividades del ordenador, las operaciones a realizar y su orden; y de una Unidad Aritmético-Lógica (ALU), encargada de las operaciones de cálculo y de comparación.

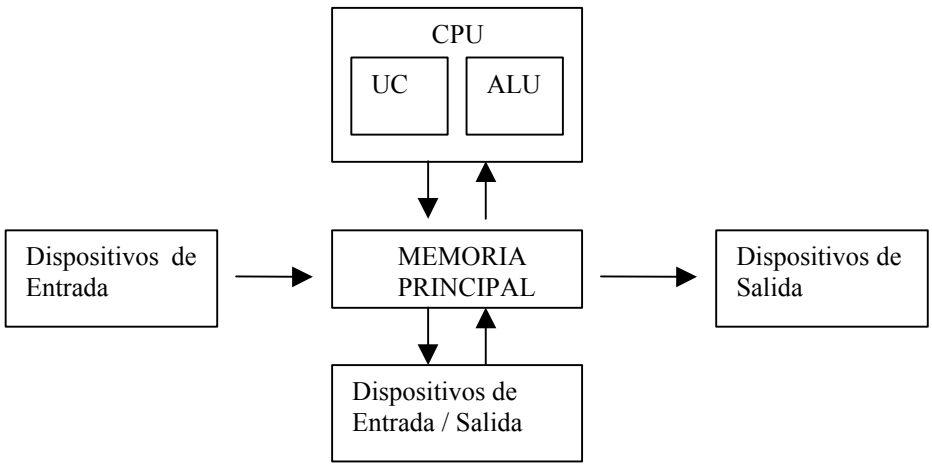
Los datos que necesite un programa, y las instrucciones que forman dicho programa deben residir en una parte de la memoria interna del ordenador llamada RAM (memoria de acceso aleatorio). La memoria ROM (memoria de solo lectura) es otra parte de la memoria interna en la que se almacenan programas de forma permanente, cuya funciones principales son el arranque y la configuración del ordenador.

La memoria del ordenador se organiza en millones de unidades de almacenamiento individuales llamadas celdas. En cada celda se almacenan un cierto número de bits, normalmente ocho. Las celdas se referencian o identifican por una dirección numérica. Este número expresa la posición relativa de la celda en la memoria.

Para que los datos puedan introducirse en la RAM, se necesitan una serie de dispositivos llamados de Entrada. Teclados, ratones, escaners, etc. nos permiten introducir datos y codificarlos en el sistema de representación binario que usan los ordenadores actuales.

Los resultados e informaciones obtenidos por el proceso de datos, deben presentarse de forma adecuada para que las personas podamos manejarlos. Los monitores, impresoras, trazadores de gráficos, etc. son dispositivos de salida.

Para que los datos e informaciones se puedan registrar y almacenar de una forma permanente, se necesitan sistemas de almacenamiento no volátiles. Los discos y cintas con sus diferentes tecnologías, tipos y formatos, son los dispositivos de almacenamiento más usados. El siguiente esquema representa la organización física de un ordenador general.



1.4 Algoritmos.

Un programador debe resolver problemas mediante programas. Resolver problemas no es fácil, pero la metodología de la programación que veremos, nos ayudará a resolver problemas mediante programas de una forma rigurosa y sistemática.

Los programas necesitan, antes de construirse, diseñarse detalladamente. El programador, antes que nada, debe encontrar un método para resolver el problema en cuestión. Es decir, debe encontrar las acciones y operaciones que resuelvan el problema: el algoritmo. Un **algoritmo** se podría definir como las acciones y operaciones detalladas necesarias a tomar para la resolución de un problema.

Los pasos para la resolución de un problema son:

1. Diseño del algoritmo que describe la secuencia ordenada de pasos que conducen a la solución.
2. Expresar el algoritmo como un programa en un lenguaje de programación.
3. Ejecución y validación del programa en el ordenador.

Un algoritmo debe ser **preciso** en cuanto al orden o secuencia de las operaciones, **finito** en cuanto al número de operaciones, y **definido** en el sentido de que se sigue dos veces el algoritmo se debe obtener el mismo resultado cada vez. En general un mismo problema puede ser resuelto por diferentes algoritmos.

Los algoritmos se pueden representar de diferentes formas: **gráfica** usando los ordinogramas y organigramas (diagramas de flujo), y mediante **lenguaje** a través de pseudocódigo o lenguaje natural.

Las representaciones gráficas de algoritmos son más claras cuando la solución es corta o pequeña, pero si el algoritmo es un poco largo, su representación ocupa mucho espacio y se vuelve engorrosa de seguir. El pseudocódigo intenta representar la solución utilizando un lenguaje natural, pero utilizando una sintaxis y estructura parecida a las de los lenguajes de programación. De esta forma, una vez obtenida la solución algorítmica representada en pseudocódigo, el paso a un lenguaje programación es "casi" inmediato. Usaremos pseudocódigo en nuestro curso ya que también es el método más aceptado en la programación estructurada.

1.5 Aplicación informática. Ciclo de vida de una aplicación.

Una **aplicación informática** es un conjunto de programas enlazados e interdependientes que resuelven un problema complejo. Gestión de nóminas, gestión de pedidos, control de almacenes, calculo matricial, gestión de centros educativos, etc. , son ejemplos de aplicaciones informáticas.

El desarrollo de una aplicación informática contempla una serie de etapas bien diferenciadas tanto en diseño y construcción como en los recursos humanos y técnicos. Se denomina **ciclo de vida** de un sistema informático al conjunto de etapas de desarrollo desde que se diseña hasta que muere (deja de tener validez). El número y nombre de estas etapas difieren según la metodología que se use en el diseño o en los autores de estas metodologías.

Según las actividades que se realizan, las etapas de desarrollo podrían ser:

1. **Identificación de necesidades**. En esta fase se pretende conocer a fondo el problema, definiéndolo sin ambigüedades, delimitando el entorno al que afecta y alcance del problema.
2. **Análisis de requerimientos**. Se pretende identificar que es lo que se necesita para resolver el problema. El resultado es un diseño general del sistema usando métodos de representación gráficos.
3. **Estudio de viabilidad**. Se exponen las diferentes soluciones según los recursos económicos y técnicos disponibles.
4. **Diseño**. Una vez conocido lo que el sistema debe hacer, tenemos que llegar a saber cómo ha de hacerse. Para ello se debe dar la solución algorítmica indicándose los procesos y datos que intervienen. Normalmente esta fase se descompone en dos etapas: diseño externo, donde se elaboran los diseños de pantallas, menús, informes, listados, etc. y diseño interno, estableciéndose los archivos o bases de datos necesarios para almacenar los datos y sus relaciones entre ellos.
5. **Codificación**. Una vez diseñada la solución algorítmica, se traduce al lenguaje de programación.
6. **Pruebas**. Se comprueba el correcto funcionamiento de los módulos y la integración de todos ellos.

7. **Implantación**. Se realizan los trabajos de instalación de la aplicación.
8. **Mantenimiento**. Se modifica, actualizan y depuran los módulos que componen la aplicación.

Hemos dejado fuera de estas etapas, las que están relacionadas con la planificación, control y seguimiento del proyecto.

1.6 Codificación y errores.

La codificación es una de las etapas más importantes del ciclo de vida de un proyecto. Para que conseguir una ejecución sin errores y en funcionamiento se requiere un gran trabajo y métodos apropiados. Los lenguajes de programación son las herramientas que se usan para codificar programas. Ellos nos permiten pasar la solución algorítmica al lenguaje máquina que entiende el hardware del ordenador.

Existen multitud de lenguajes de programación. Muchos se han construido a partir de otros, y en general están construidos o diseñados para aplicaciones de diferentes tipos. Hay lenguajes de tipo científico, gráfico, de gestión, etc. Los lenguajes de propósito general son los que permiten codificar problemas de muy diverso tipo, como son los casos de C y C++, lenguajes que aprenderemos en este curso.

Otra tarea del programador es controlar y corregir los errores que se producen durante la codificación y ejecución de los programas que componen la aplicación. Podemos clasificar los errores en dos tipos:

Los **errores de codificación** son errores producidos por el programador al usar instrucciones que están escritas de forma incorrecta. Errores léxicos, de sintaxis o semánticos son típicos, sobre todo al comienzo del estudio de un lenguaje de programación. Los compiladores o traductores del lenguaje detectan estos errores y los señalan.

Los **errores de lógica** son aquellos que se producen cuando el programa no realiza lo que inicialmente se preveía. Su origen puede estar en un mal diseño del algoritmo, o en el paso del sistema de representación del algoritmo al lenguaje escogido para codificarlo. Los depuradores de código pueden ayudar a localizar estos errores. Los depuradores permiten ejecutar el programa paso a paso, investigar las llamadas a otros programas, conocer en cada momento el contenido de las variables, etc.

1.7 Lenguajes de programación. Tipos.

Un **lenguaje de programación** es una notación para escribir programas que permitan comunicar al hardware las órdenes que se realizan en un proceso. La tarea principal de un programador es traducir un algoritmo a un lenguaje de programación.

Como ya hemos comentado, los lenguajes intentan aprovechar el desarrollo del hardware mediante la diversificación, especializándose según el tipo de aplicación a programar: propósito general, científico, gestión, edición, etc.

Los pasos o acciones de una algoritmo, se expresan en los programas como **instrucciones**. Luego un programa consta de una secuencia de instrucciones, cada una de las cuales especifica ciertas operaciones que debe ejecutar el ordenador. Es evidente que para elaborar un programa se debe conocer el repertorio de instrucciones del lenguaje elegido.

1.7.1 Tipos de lenguajes

Los lenguajes de programación se suelen clasificar, desde el punto de vista de programación de las aplicaciones en diferentes niveles: bajo nivel, intermedios, alto nivel.

§ Lenguaje máquina

Es el lenguaje binario que entiende el hardware. En este lenguaje todas las instrucciones están formadas por tiras de bits con un significado para el procesador. El código binario es dependiente de la máquina. Así, el repertorio de instrucciones máquina que entiende un procesador Intel o compatible es diferente al código

binario de un PowerPC, Motorola, Digital , etc. Programar en máquina es muy complicado, lento y produce muchos errores. Sus únicas ventajas son que el código resultante es muy reducido y rápido en ejecución.

§ Lenguaje ensamblador

Es la primera evolución del lenguaje máquina hacia el lenguaje humano. Usa palabras mnemotécnicas que sustituyen a las cadenas de bits. No existe un único lenguaje ensamblador. Hay tantos lenguajes ensambladores como tipos de procesadores. No obstante los lenguajes ensambladores de una misma familia de procesadores suelen ser compatibles. Así un programa escrito en el ensamblador del procesador Intel 80386, también es ejecutable en los procesadores 80486, Pentium, Pentium II, etc. ,y compatibles con éstos de otros fabricantes.

El lenguaje ensamblador no es ejecutable directamente por la máquina, necesita de una traducción al lenguaje máquina. Es complejo, se necesita conocer muy bien el hardware, todas las instrucciones son elementales, pero es muy eficaz en rendimiento y ocupación de memoria.

§ Lenguajes de alto nivel

Son el resultado de la evolución de los lenguajes hacia el humano facilitando la escritura y entendimiento de los programas. Los objetivos que se alcanzan son:

1. Independencia de la máquina. Se permite el uso del mismo programa en diferentes equipos con la condición de tener los traductores apropiados al lenguaje máquina de cada equipo. Con esto conseguimos que los programas sean transportables de una máquina a otra, cosa que no sucedía con los lenguajes ensambladores y máquina.

2. Facilitar el uso y aprendizaje al ser parecidos al lenguaje natural inglés.

3. Reducir el coste de los programas al ser más fáciles de modificar y emplear menos tiempo en la codificación.

Los lenguajes de alto nivel deben traducirse al lenguaje máquina, que es lo que realmente entiende el ordenador. Actualmente existen miles de lenguajes que complican un poco la programación. Además, los programas creados con estos lenguajes consumen mucha más memoria y son más lentos de ejecutar. Entre 1954 y 1960 surgen los primeros lenguajes de alto nivel: Fortran, Cobol y Algol. Muchos de los lenguajes actuales se han diseñado a partir de estos primeros.

1.7.2 Traductores.

Los lenguajes simbólicos (no-máquina) necesitan traducirse a lenguaje máquina, el que entiende el hardware. Así, un traductor es un programa encargado de traducir código escrito en un lenguaje simbólico a código binario o máquina. Cada lenguaje necesita su traductor. Existen dos tipos de traductores:

- **Compiladores**: Traduce un programa escrito en un lenguaje intermedio o de alto nivel a lenguaje máquina. El programa escrito en lenguaje simbólico que se va a traducir se denomina **programa fuente**. El programa equivalente como producto de la traducción se llama **programa objeto**. Si hay errores se genera un listado de todos los detectados en la traducción. Los compiladores de programas escritos en lenguaje ensamblador (*assembly lenguage)*, se denominan **ensambladores** *(assemblers)*. No hay que confundir el lenguaje con su traductor; en este caso, la traducción al español de ambos términos ha originado la misma palabra: ensamblador. Los ensambladores generan una sola instrucción máquina por cada instrucción escrita en ensamblador. En los compiladores de lenguajes de alto nivel, por cada instrucción de alto nivel se genera varias instrucciones máquina

- **Intérpretes**: Se realiza una traducción y ejecución posterior, de cada una de las instrucciones del programa escrito en lenguaje de alto nivel. Si al traducir una instrucción hay errores en ésta, se para la ejecución del programa. La principal ventaja de los intérpretes es que es fácil detectar errores de ejecución, pero la ejecución del programa es lenta debido a que cada vez que se ejecuta hay que traducir el código.

1.7.3 Etapas en el proceso de compilación.

1. **Edición.** Consiste en la escritura del programa en un lenguaje de programación y su posterior grabación en un soporte de almacenamiento (normalmente en disco magnético). Para editar el programa se necesita un programa editor, que puede o no ser parte del compilador utilizado como una herramienta más. En esta fase obtenemos el programa o código fuente.

2. **Compilación.** En esta fase se traduce el programa fuente en su equivalente máquina obteniéndose un programa o código objeto. Si se producen errores, el compilador los mostrará mediante mensajes para indicarnos dónde están y que tipos de errores hemos cometido. En este caso, deberemos editar de nuevo el programa fuente, subsanar los errores y volver a compilar.

3. **Enlace.** Esta fase se llama también montaje o *linkado*. Consiste en unir o enlazar el programa objeto producto de la compilación con determinadas rutinas o módulos propios del lenguaje. Si la aplicación consta de varios módulos objetos, también deberemos enlazarlos para obtener un código o programa ejecutable. En el proceso de montaje también pueden aparecer errores, normalmente relacionados con la ubicación de los módulos a enlazar.

4. **Ejecución.** En esta fase se carga el programa en memoria y se comprueba que la ejecución es correcta. Mediante juegos de prueba se realizan comprobaciones de ejecución, básicamente consistentes en comprobar si son correctos los datos obtenidos en función de unos determinados datos de entrada.

1.8 Documentación.

La documentación de programas que constituyen una aplicación, es una de las tareas más importantes tanto en el desarrollo de la aplicación como en el posterior mantenimiento. Una documentación correcta nos permitirá posteriormente reutilizar parte de los programas en otros desarrollos software. La documentación sirve para explicar el significado de los procesos y de las operaciones e instrucciones que realiza el programa. Esto permite que otras personas, e incluso el mismo autor de la aplicación pasado un tiempo, conozca la finalidad del código escrito. Desgraciadamente, es una de las tareas que el programador suele simplificar o dejar incompleta.

La documentación de los programas debe comenzar desde el inicio del ciclo de vida de un proyecto informático, sin esperar a tener finalizada la construcción de todos los programas que lo componen.

Al final del proyecto, se deben tener básicamente los siguientes documentos: una guía técnica, una guía de uso y una guía de instalación.

1.8.1 Guía técnica.

Es el documento donde queda especificado el diseño del proyecto, la codificación de los programas y las pruebas realizadas para comprobar su funcionamiento. Los destinatarios de esta guía son los analistas y programadores (personal técnico).

La finalidad de una guía técnica es la de facilitar el desarrollo, corrección y mantenimiento de los programas de una forma rápida y precisa. La guía técnica está formada por los siguientes documentos:

1. **Cuaderno de carga.** Documento donde se refleja el diseño de la aplicación según las necesidades del usuario y como resultado del análisis de requerimientos realizado por los analistas. El documento está destinado a los programadores. Su contenido se clasifica en diferentes partes; generalmente contienen:

   - Una descripción general de la aplicación.
   - Una relación y descripción de los datos que se usan.
   - Las especificaciones y diseño de entradas y salidas. (Pantallas, formularios, controles.).
   - La estructura y descripción de los módulos principales de la aplicación.
   - Una relación de todas las especificaciones por cada uno de los programas (descripción del programa, diagrama modular, variables de uso, módulos que lo forman y algoritmo.

2. **Programa fuente.** Es el donde documento donde se incluye la codificación de los programas según el lenguaje y compilador seleccionado. El programa fuente debe contener comentarios y utilizar un estilo claro y normalizado que será de utilidad para el mantenimiento y explotación del programa.

3. **Juego de pruebas.** Las pruebas se pueden realizar en cualquiera de las etapas de la aplicación. Los tipos de pruebas son:

   - Pruebas unitarias para cada módulo dentro de un programa.
   - Pruebas de interconexión para cada uno de los programas con todos sus módulos.
   - Pruebas de integración del sistema para todos los programas que componen la aplicación.

1.8.2 Guía de uso.

Es el manual de usuario donde se incluye toda la información necesaria para que los usuarios puedan utilizar correctamente la aplicación. Se debe presentar de forma que el usuario la comprenda con claridad, evitando todas las referencias a la parte técnica y centrándose sobre todo en las entradas y salidas que maneja la aplicación.

Debe estar redactada con estilo claro, conciso y sin ambigüedades. Si es necesario debe ir acompañada con un glosario que explique aquellos términos informáticos desconocidos por los usuarios. Una buena guía de uso debería contener las siguientes informaciones:

- Forma de uso de la guía.
- Descripción de la aplicación.
- Descripción del sistema hardware existente.
- Forma de lanzar la ejecución de la aplicación.
- Orden en que se desarrollan los procesos y su descripción.
- Descripción de los formatos de los documentos de entrada de datos, descripción de los formatos de salida en pantalla e impresora y de los controles de los datos que necesitan estas salidas.
- Ejemplos de uso de los programas.

1.8.3 Guía de instalación.

También llamado manual de explotación, es el documento que contiene la información que se necesita para poner en marcha el sistema y las normas de explotación.

Para la puesta en marcha del sistema se debe seguir una serie de normas relacionadas con la implantación de la aplicación en el sistema físico, la importación de datos desde el sistema anterior al nuevo sistema y la realización de pruebas del nuevo sistema. Cuando las pruebas sean satisfactorias, se podrá sustituir el nuevo sistema por el anterior.

Las normas de explotación son las normas del uso normal de la aplicación. Se establecen las normas para:

- Los operadores de consola.
- Las respuestas a los mensajes de error.
- La prioridad y periodicidad de los procesos.
- La administración de los datos.
- La forma de administrar la red de comunicación.
- La utilización de programas de utilidad.
- La seguridad física del sistema y copias de seguridad.
- La seguridad de acceso a la información.