Departamento de Lenguajes y Sistemas Informáticos

521069 FUNDAMENTOS DE INFORMÁTICA

EQUIPO DOCENTE:
Dra. COVADONGA RODRIGO SAN JUAN
D. JOSÉ LUIS DELGADO LEAL

Tema 2. Conceptos básicos de Hardware y Software: familiarizarse con el ordenador

Objetivos Generales del Tema

- Cómo funcionan los computadoras
- Conocer e identificar los componentes hardware
- Conocer e identificar los componentes software
- Conocer en qué consiste la Ingeniería del Software

Indice

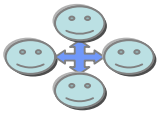
2.1 Concepto de Informática

El origen del computador se debe a la necesidad de realizar cálculos de forma automática. Sin embargo, el procesamiento numérico no es su única utilidad. La posibilidad de realizar operaciones lógicas le dota de la capacidad de ser usado para el procesamiento general de datos de información. La informática, es precisamente el cuerpo de conocimiento que se encarga de todo lo relacionado con el desarrollo y uso de los sistemas basados en computador.

Todos hemos escuchado alguna vez la frase que dice que vivimos en la sociedad de la información. La tecnología afecta nuestra vida diaria, en casa, en el trabajo, en los centros educativos,.... Las nuevas tecnologías hacen posible que tengamos disponible una gran cantidad de información a nuestra disposición y además nos facilitan los medios de comunicarnos, la automatización de un montón de procesos, etc

Por ejemplo, los computadores nos permiten:

- Disponer de electrodomésticos inteligentes que controlan el gasto energético, etc
- Mantener un sofisticado sistema de seguridad en nuestro domicilio
- Encontrar todo tipo de información a través de Internet.
- Organizar la información de nuestro ordenador personal, fácil de acceder, de forma organizada, etc.
- Comunicarnos rápida y fácilmente con otras personas, a través de correo electrónico, mensajería por teléfono, sitios web, etc.



Actividad 2.1. Piense y enumere tres actividades concretas de su vida diaria o profesional que se hayan visto afectadas o hayan mejorado por el uso de las nuevas tecnologías informáticas.

<Envíe un mensaje al foro general de consultas para intercambiar impresiones con sus compañeros del Campus –e o de la tutoría de la asignatura en el centro asociado>

2.2 Definir el término Computador

Un computador es, por tanto, un dispositivo que procesa datos y los convierte en información útil para las personas. Para funcionar necesita ser programado, es decir, saber lo que tiene que hacer. Ello se consigue mediante un juego de instrucciones, que al ponerlas de forma ordenada se convierten en programas y generan las órdenes que debe cumplir el computador. Estas órdenes, son secuencias de pasos muy estrictas, qué deben estar perfectamente definidas y muy

bien organizadas, y es lo que más adelante en este tema se denominará parte "software" del computador .

El término computador digital, simplemente se utiliza porque los computadores trabajan generalmente con "dígitos" o números. El computador no trabaja con palabras o con el lenguaje que utilizamos los humanos, y por eso existe un proceso bastante complicado - que afortunadamente es transparente para los usuarios - en el que las órdenes que los programadores dan a los computadores se convierten en secuencias de dígitos 1 y 0 que es el "lenguaje" que realmente comprenden los microchips.

Todos estos sistemas o "aparatos" operan básicamente bajo los mismos principios, están fabricados con similares componentes básicos y requieren de un juego de instrucciones particulares para poder ser programados y funcionar. Todos disponen de un pequeño microcomputador, un microchip, que hace las veces de "cerebro".

Definición:

Computador: aparato electrónico capaz de interpretar y ejecutar comandos programados para operaciones de entrada, salida, cálculo y lógica. Los computadores:

1. Reciben entradas. La entrada son los datos que se capturan en un sistema de computación para su procesamiento.
2. Producen salidas. La salida es la presentación de los resultados del procesamiento.
3. Procesan información
4. Almacenan información

2.3 Sistemas basados en computador

Al desarrollar los primeros computadores se planteó la necesidad de programarlos y de almacenar en su memoria la información sobre la tarea que iban a ejecutar. Los primeros prototipos se usaban como calculadoras simples indicando uno por uno los pasos de cálculo. John von Neumann desarrolló tras la Segunda Guerra Mundial el modelo que lleva su nombre para describir este concepto de programa almacenado. Este modelo consistía en dos partes: una memoria, que guardaba instrucciones y datos, y una unidad central, que procesaba los contenidos de la memoria.

Las máquinas modernas tienen tanto en común con las ideas de von Neumann y sus colegas que se les hace referencia como "máquinas (o arquitectura) von Neumann". Paulatinamente, se han desarrollado en el tiempo los lenguajes que las programan. Primero surgieron los lenguajes ensambladores (muy cercanos a los lenguajes de las máquinas), con posterioridad los lenguajes de programación imperativos (C, Pascal, etc) hasta evolucionar hacia los más actuales que siguen el paradigma de la orientación a objetos (Ada, Java, C++, entre otros muchos). Tanto los imperativos como los orientados a objetos reciben también el nombre de ***lenguajes de alto nivel*** porque las órdenes se dan en forma de sentencias se aproximan bastante a nuestro lenguaje natural (al menos al mundo anglosajón ya que muchas palabras clave son palabras directas en inglés). Pero el proceso no termina hasta que las instrucciones escritas en el lenguaje de alto nivel se convierten en lenguaje máquina, utilizando para ello una herramienta informática que se denomina compilador.

Definición:

Un sistema informático es básicamente un sistema basado en computador, es decir, un conjunto de entidades que interrelacionan para un fin común y cuyo control depende de un computador o de un conjunto de ellas. Cada entidad realiza un procesamiento diferente de la información.

Actividad 2.2. Piense en dos inventos tecnológicos recientes que sean ejemplos de sistemas basados en computador e intente adivinar que funciones realiza el microchip que contienen.

<Envíe un mensaje al foro general de consultas para intercambiar impresiones con sus compañeros del Campus –e o de la tutoría de la asignatura en el centro asociado>

Uno de los mejores ejemplos de sistema basado en un computador es precisamente el **Ordenador Personal**, el cual puede tener la forma de un ordenador de sobremesa, un portátil, un reproductor de MP3, una PDA, etc. (ver figura 1).

Figura 1. Ordenadores Personales

Una primera clasificación de ordenadores personales puede realizarse en base al sistema operativo que utilizan, denominándose generalmente:

- **PC** – los que utilizan el sistema operativo Microsoft Windows o una distribución Linux
- **Mac** – usan el sistema operativo de Macintosh

También se pueden clasificar atendiendo a su tamaño, finalidad de uso, número de usuarios concurrentes en el tiempo, etc (ver tabla adjunta).

Computador Personal (PC)	Uno	Cabe en una mesa o incluso en una mano	Utilizado por individuos bien para organizar información, para crear productos o para entretenerse.
Minicomputador	Varios o cientos	Ocupa parte de una sala	Utilizado por PYMES, organizaciones académicas, etc
Mainframe (super computador)	Miles	Tamaño de una sala	Utilizado en grandes corporaciones y agencias gubernamentales

2.4. Una mirada al interior del sistema computador

Todos los sistemas informáticos o sistemas basados en computador disponen de dos partes claramente diferenciadas: hardware y software, términos anglosajones ampliamente aceptados en la comunidad informática y que hay que aprender a utilizar con fluidez.

2.4.1. Representación de la información y su almacenamiento

La información es un conjunto de datos procesados y organizados, por tanto, implica tanto un conjunto de datos como su interrelación. Para representar los datos existen dos formas de hacerlo: la representación analógica y la digital. En los ordenadores toda la información está almacenada digitalmente, desde los números al texto pasando por imágenes de vídeo o registros de audio.

Es importante conocer las siguientes *definiciones* (utilizando su denominación anglosajona):

bit
La unidad más pequeña de datos. Su valor es 0 ó 1. También se denomina dígito binario.

byte
Bloque compuesto por 8 bits. Generalmente, es el mínimo bloque de información que fluye por el computador. Por ejemplo, cada letra del lenguaje natural es representada por un byte.

Generalizando,
un **Kilobyte (Kb)** son 2^{10} or 1,024 bytes.
un **Megabyte (Mb)** son 1,048,576 bytes.

2.4.2 Definir el término Hardware

Se denomina **hardware** o **soporte físico** al conjunto de elementos materiales que componen un ordenador. Hardware también son los componentes físicos de una computador tales como el disco duro, CD-ROM o DVD, disquetera (floppy), etc.. En dicho conjunto se incluyen los dispositivos electrónicos y electromecánicos, circuitos, cables, tarjetas, placa madre, armarios o cajas, periféricos de todo tipo y otros elementos físicos.

El hardware se refiere a todos los componentes físicos (que se pueden tocar) de la computadora: discos, unidades de disco, monitor, teclado, ratón, impresora, placas, chips y demás periféricos.

2.4.3 Tipos de hardware

Se clasifica generalmente en básico y complementario, entendiendo por básico todo aquel dispositivo necesario para iniciar el ordenador, y el complementario como su nombre lo dice sirve para realizar funciones específicas o más allá de las básicas.

Definiciones:

Unidad central de procesamiento (CPU)

Es la computador real, la "inteligencia" de un sistema de computación. Los microchips utilizados – Intel Pentium, AMD Athlon, etc - realizan funciones básicas hasta millones de veces por segundo. Cada vez que la CPU recibe y ejecuta una instrucción, completa un ciclo de reloj.

Memoria y dispositivos de almacenamiento

En la placa madre del ordenador está situado el microchip y varios chips de memoria que componen la memoria primaria del ordenador (RAM, caché, etc). Esta memoria guarda la memoria "más inmediata" que utilizada el microchip, es decir, la de antes y después del procesamiento.

Las unidades externas de almacenamiento, componen la llamada memoria secundaria, y son también bien conocidos: discos duros, disquetes o floppy, ZIP, memorias de pequeño tamaño tipo flash, etc.

Periféricos de entrada

Son los que permiten que el usuario aporte información exterior. Entre ellos podemos encontrar: teclado, ratón (mouse), escáner, cámara digital, micrófono, SAI (Sistema de Alimentación Ininterrumpida), etc.

Periféricos de salida

Son los que muestran al usuario el resultado de las operaciones realizadas por el PC. En este grupo podemos encontrar: monitor, impresora, altavoces, etc.

Periféricos de comunicación

Son los dispositivos que pueden aportar simultáneamente información exterior al PC y al usuario. Aquí se encuentran: módem, tarjetas de red local, tarjetas de conexión inalámbrica,...

A continuación se presentan una serie de consideraciones importantes que el alumno debe conocer sobre parte del hardware básico de un ordenador.

Velocidad de proceso de la CPU

Se denomina velocidad de cómputo o de proceso de la CPU al número de ciclos completados por unidad de tiempo.
Así, los ciclos se miden en:

- Megahercios (MHz) = millones (10^6) de ciclos por segundo
- Gigahercios (GHz) = Mill millones (10^9) de ciclos por segundo

Memoria del ordenador

El almacén de datos se realiza sobre una gran diversidad de dispositivos de memoria. A este respecto hay dos consideraciones importantes a destacar:

- **Distinción entre memoria principal y secundaria**

La memoria primaria se puede considerar como la más inmediata y de carácter volátil (es decir, al apagar el ordenador, esta memoria se vacía). En realidad está compuesta por varios chips específicos situados en la plaza madre (RAM, memoria caché) o zonas reservadas del disco duro (memoria virtual), cada uno de los cuales tiene distinta función y propiedades, pero cuya explicación se sale fuera de las competencias de este curso introductorio de la informática.

Como ya se ha mencionado antes, las unidades externas de almacenamiento, componen la llamada memoria secundaria, y son también bien conocidos por el usuario: discos duros, disquetes o floppy, ZIP, memorias de pequeño tamaño tipo flash, etc.

- **Medidas de memoria**

Las unidades básicas de medida del almacén de datos son:
- Kilobyte (KB) = 1024 bytes
- Megabyte (MB) = 1048576 bytes (1024 bytes x 1024 bytes)

- **Distinción entre memoria RAM y ROM**

Tipo de memoria	¿Qué función realiza?	¿Cuándo se usa?
Read-only memory (ROM)	Almacena información permanente, cómo por ejemplo, quién es el ordenador y que componentes tiene al arrancar (soy un Pentium y tengo un monitor LCD, grabadora de DVD, etc.)	Al encender el ordenador
Random-access memory (RAM)	Almacena información temporal (por ejemplo, copia de la carta que estamos escribiendo con un procesador de textos)	Al arrancar cualquier programa

Actividad 2.3. A partir de las definiciones presentadas, encontrar la equivalencia en bytes de un Gigabyte (GB) y un Terabyte (TB) .

<Envíe un mensaje al foro general de consultas para intercambiar impresiones con sus compañeros del Campus –e o de la tutoría de la asignatura en el centro asociado>

Práctica 2.1. Conocer el Ordenador Personal En esta práctica el estudiante debe identificar las distintas partes hardware que componen su ordenador personal o el de su trabajo. Como orientación, el estudiante debe al menos reconocer los componentes marcados con círculo rojo en la figura 2.

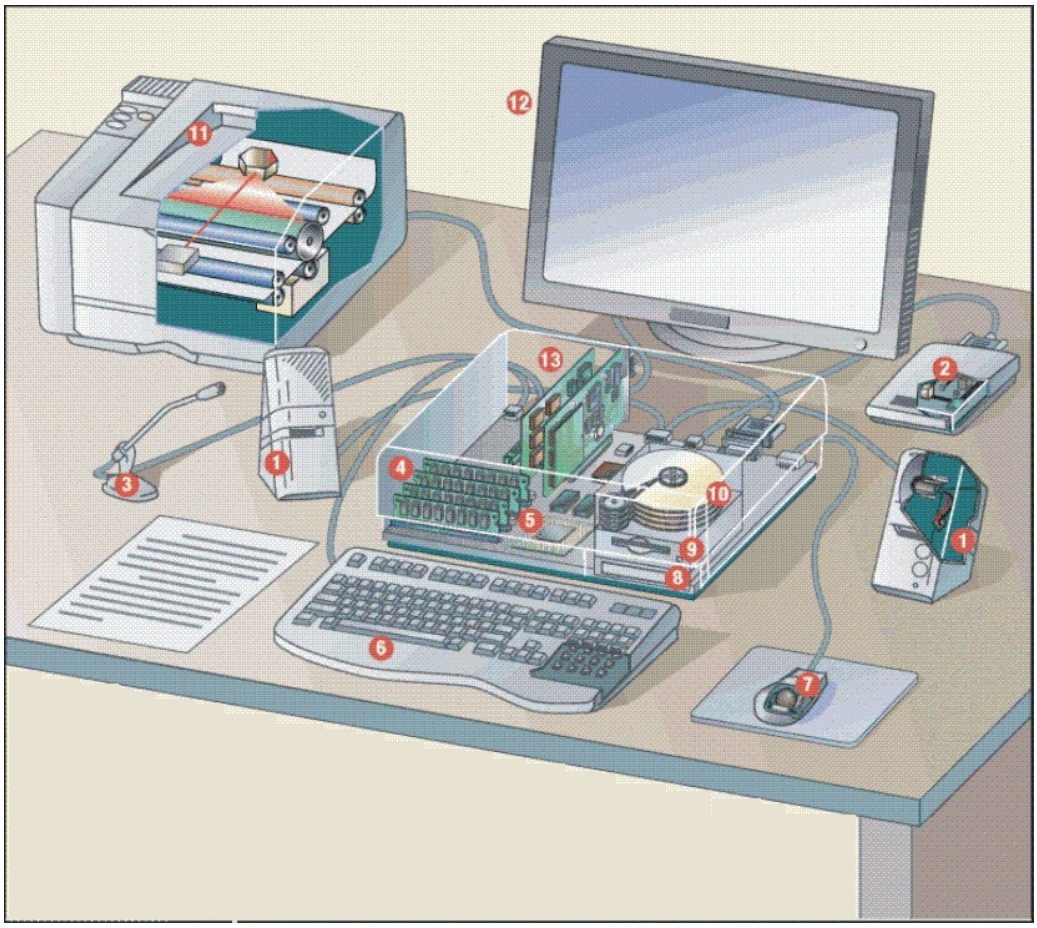


Figura 2. Componentes hardware de un Ordenador Personal

<Envíe un mensaje al foro general de consultas para intercambiar impresiones con sus compañeros del Campus –e o de la tutoría de la asignatura en el centro asociado>

Práctica 2.2. Reconocer tipos de hardware. En esta práctica el estudiante debe identificar los distintos tipos de hardware que aparecen en la siguiente ficha técnica de un anuncio comercial.

Ficha Técnica **Marca:** PEPITO GRILLO
Procesador: Intel® Core™ Duo T2250
Velocidad del Procesador: 1,73 Ghz
Memoria Caché: 2 MB de caché de nivel 2
Memoria RAM: 1024 (2 x 512 MB) MB.
Tipo de memoria RAM: DDR2 a 667 MHz
Ampliación Memoria RAM: 2 GB.
Disco Duro: 120 GB.
Controladora Disco Duro: SATA a 5.400 rpm
Unidad Óptica: Grabadora de DVD Super Multi (+/-R +/-RW) con soporte para doble capa
Tarjeta Gráfica: NVIDIA® GeForce™ Go 7400 con 256 MB de memoria de vídeo TurboCache™ incluida 128 MB de memoria de vídeo dedicada
Audio:
- Sonido 3D de 16 bits compatible con Sound Blaster Pro integrado
- Altavoces Altec Lansing®
Tarjeta de Red: Ethernet 10/100BT integrado
Conexiones inalámbricas: LAN inalámbrica integrada Intel® PRO/Wireless 3945 802.11a/b/g
Módem / Fax: Módem 56K de alta velocidad
Conexión a Internet: Conexión fácil a Internet con los principales proveedores.
Tipo de Batería: Ion de litio (Li-Ion) de 6 celdas
Dispositivos Entrada / Salida:
- 1 puerto VGA
- 3 puertos USB 2.0
- 1 IEEE-1394
- 1 conector de módem RJ 11
- 1 conector RJ 45 Ethernet
- Salida de televisión S-video
- 2 puertos de salida de auriculares con SPDIF audio digital
- 1 puerto de entrada de micrófono
- Conector de conexión de cables
Puertos USB: 3 USB 2.0
Número de Slots PCMCIA: 1 x Ranura PC Card Tipo I ó Tipo II
Tamaño de Pantalla: 15.4"
Tipo de Pantalla: Pantalla panorámica de alta definición BrightView, WXGA
Resolución máxima: 1.280 x 800
Tipo de Teclado: Teclado de tamaño completo de 101 teclas
Tipo de Ratón: Touchpad
Dimensiones: 237 x 257 x 25,4/39,6 mm.
Peso: 2,99 Kg.
Alimentación: Adaptador de corriente de 90 W
Sistema Operativo preinstalado: Windows XP Media Center Edition 2005 original con Update Rollup 2
Software preinstalado:
- Microsoft® Works 8.0
- Microsoft® Internet Explorer 6.0
- Microsoft® Outlook Express
- Adobe® Reader 7.0
- Sonic™ Digital MediaPlus: Sonic™ RecordNow
- Sonic™ MyDVD
- Sonic™ Easy Archive
- Sonic™ Express Labeler
- Microsoft® Windows® Media Player
- Microsoft® MovieMaker
- HP Photosmart Premier
- DVD Play (con Cyberlink)
- QuickPlay Direct y QuickPlay para Windows
Garantía: 2 años. Ver condiciones en el menú de Ayuda.
Más información:
- WebCam HP Pavilion con micrófono integrado
- Lector Digital Media Reader integrado "5 en 1" para tarjetas Secure Digital, MultiMedia, Memory Stick, Memory Stick Pro o xD Picture

<Envíe un mensaje al foro general de consultas para intercambiar impresiones con sus compañeros del Campus –e o de la tutoría de la asignatura en el centro asociado>

Práctica 2.3. Reconocer características del hardware. Tomando la ficha técnica presentada en la práctica 2.2., el estudiante debe identificar la velocidad de proceso de la CPU, y aglutinar la lista de componentes hardware que forman la memoria principal y memoria secundaria.

<Envíe un mensaje al foro general de consultas para intercambiar impresiones con sus compañeros del Campus –e o de la tutoría de la asignatura en el centro asociado>

2.4.4. Definir el término Software

Se denomina **software** (también **soporte lógico**) a todos los componentes intangibles de un ordenador o computadora, es decir, al conjunto de programas y procedimientos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema (hardware). Esto incluye aplicaciones informáticas tales como un procesador de textos, que permite al usuario realizar una tarea, y software de sistema como un sistema operativo, que permite al resto de programas funcionar adecuadamente, facilitando la interacción con los componentes físicos y el resto de aplicaciones.

El software es intangible, existe como ideas, conceptos, símbolos, pero no tiene sustancia. Una buena metáfora sería un libro: las páginas y la tinta son el hardware, mientras que las palabras, oraciones, párrafos y el significado del texto son el software. Un computador sin software sería tan inútil como un libro con páginas en blanco.

Probablemente la definición más formal de software es la atribuida a la IEEE en su estándar 729: «la suma total de los programas de cómputo, procedimientos, reglas documentación y datos asociados que forman parte de las operaciones de un sistema de cómputo. Bajo esta definición el concepto de software va más allá de los programas de cómputo en sus distintas formas: código fuente o código máquina, binario o ejecutable, además de su documentación: es decir, todo lo intangible.

Comúnmente se considera el software como sinónimo de programa, y sin embargo, en informática es un concepto mucho más general que abarca no sólo el código generado sino toda la documentación asociada. Existen dos grandes grupos, a saber: software de sistemas y de aplicación. El primero es el software utilizado por otro software, como los sistemas operativos, que es el programa que permite comunicar con los elementos hardware. El segundo, es el software destinado a ser utilizado por un usuario: una hoja de cálculo, un procesador de textos, un juego, etc.

La comunicación del usuario con el software se realiza hoy en día de forma generalizada a través de las denominadas interfaces gráficas de usuario. En éstas, las capacidades del programa están representadas por iconos o símbolos gráficos a los que el usuario accede mediante un dispositivo apuntador. Con un buen diseño, es fácil independizar interfaz y parte funcional del programa con lo que se consigue un mejor mantenimiento y mayor reutilización de sus elementos.

2.4.5. Tipología de software

El software se puede clasificar según la propuesta siguiente, aunque el criterio no sea siempre claro y evidente.

Definiciones:

- **Software de sistema**, que permite funcionar al hardware. Su objetivo es aislar tanto como sea posible al programador de aplicaciones de los detalles del computador particular que se use, especialmente de las características físicas de la memoria, dispositivos de comunicaciones, impresoras, pantallas, teclados, etcetera. Incluye entre otros:
  - Sistemas operativos
  - Controladores de dispositivo
  - Herramientas de diagnóstico
  - Servidores
  - Sistemas de ventanas
  - Utilidades
- **Software de programación**, que proporciona herramientas para ayudar al programador a escribir programas informáticos y a usar diferentes lenguajes de programación de forma práctica. Incluye entre otros:
  - Editores de texto
  - Compiladores
  - Intérpretes
  - Enlazadores
  - Depuradores
  - Los entornos integrados de desarrollo (IDE) agrupan estas herramientas de forma que el programador no necesite introducir múltiples comandos para compilar, interpretar, depurar, etcétera, gracias a que habitualmente cuentan con una interfaz gráfica de usuario (GUI) avanzada.
- **Software de aplicación**, que permite a los usuarios llevar a cabo una o varias tareas más específicas, en cualquier campo de actividad susceptible de ser automatizado o asistido, con especial énfasis en los negocios. Incluye entre otros:
  - Aplicaciones de automatización industrial
  - Aplicaciones ofimáticas
  - Software educativo
  - Software médico
  - Bases de datos
  - Videojuegos

Práctica 2.2. En esta práctica el estudiante debe identificar los distintos programas y sistema operativo que utiliza habitualmente en su ordenador personal (ver ejemplo en figura 2). ¿A qué tipo de software pertenece el compilador jGRASP que va a utilizar para realizar la práctica de esta asignatura?

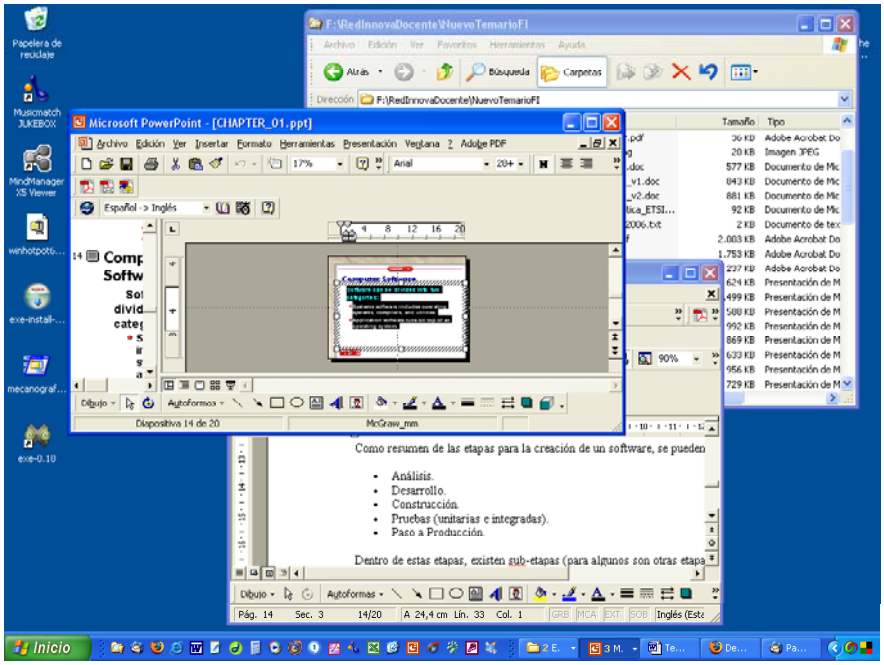


Figura 3 . Diversos programas en la pantalla de un computador.

<Envíe un mensaje al foro general de consultas para intercambiar impresiones con sus compañeros del Campus –e o de la tutoría de la asignatura en el centro asociado>

2.5 Elementos de Programación y Lenguajes

Una fuente inicial de problemas para los pioneros de los lenguajes de programación provino del propio hardware de los ordenadores. Según Wulf (1981), los ordenadores de los años 50 estaban más orientados a los lenguajes de bajo nivel (como el lenguaje ensamblador) que los de un nivel más alto, lo que daba lugar a tres tipos de problemas: en primer lugar, la falta de uniformidad en las instrucciones (algunas veces, las mismas operaciones se llevaban a cabo en registros y otras en memoria). En segundo lugar, las operaciones que formaban parte de los nuevos lenguajes de alto nivel no estaban bien soportadas. En tercer y último lugar, era difícil mantener la integridad de programas y datos, porque esta característica, que se consigue a través de pruebas contra error, no existía al no haber sido valorada tanto como la velocidad de operación a la hora de desarrollar el hardware.

Un programa es una serie de instrucciones que indican de forma precisa y exacta al computador qué tiene que hacer. Un computador se puede entender como una máquina virtual, capaz de realizar una serie de tareas genéricas pero no concretada hacia ninguna tarea específica. Es siempre, por tanto, necesario un programa qué, usando un lenguaje entendible por la máquina, le indique qué tiene que hacer en cada momento. En la figura 4 se pueden ver las fases típicas de la generación código.

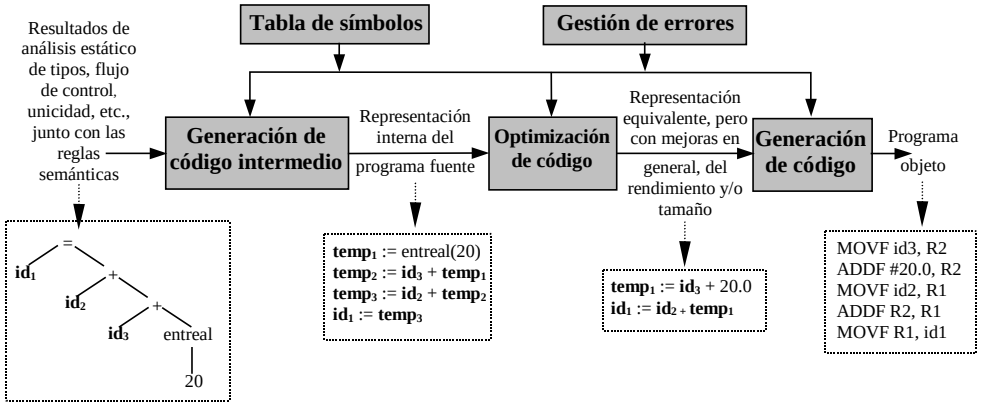


Figura 4. Fases de la generación de código

Una vez que se ha producido un código intermedio, un compilador puede llevar a cabo un proceso de optimización del código para mejorar la velocidad de ejecución del código de máquina que saldrá al final de proceso de compilación y/o su tamaño. Hay que notar que no hay ninguna garantía de que el código que salga de este proceso realmente se ejecute más rápidamente, aunque en general es así. Además, la gestión de errores en el proceso de compilación de programas es una

fase importante en el proceso de programar. Eso quiere decir que el compilador tiene que ser capaz de detectar el error en la capa de procesamiento adecuado (errores sintácticos, semánticos, etc.) y presentarlo al programador de una manera que le ayuda a depurar los programas.

Por lo general, se han vinculado el nivel de abstracción de un lenguaje y la fecha de su aparición con una generación (Parra, 1997). A continuación se van a considerar las peculiaridades de cada generación que puede darnos una indicación de cómo caracterizar cada lenguaje y cómo entender su utilización.

1ª generación: Los lenguajes de esta generación se caracterizan por poseer una codificación muy compleja, lo cual los hace difíciles de aprender, entender y aplicar. Más que lenguajes en el sentido de aportar algún grado de abstracción, son más bien códigos de máquina y no requieren traducción alguna porque el ordenador es capaz de leerlos directamente.

2ª generación: Se llama lenguajes de segunda generación a los lenguajes ensamblador. Requieren una traducción, aunque ésta es muy simple porque cada instrucción corresponde a un código solamente. Son mucho más fáciles de manejar para los profesionales que los códigos hexadecimales, por lo que se les considera lenguajes codificados y a cada palabra le corresponde una instrucción del microprocesador.

3ª generación: Estos son los más utilizados y están diseñados para ser usados por programadores profesionales. Se caracterizan por su transportabilidad, es decir, que están implementados sobre varias máquinas de forma que un programa puede ser fácilmente llevado de una máquina a otra sin una revisión sustancial. Aunque no son fundamentalmente declarativos, estos lenguajes permiten que los algoritmos se expresen en un nivel y estilo de escritura fácilmente legible y comprensible por otros programadores. Los códigos pueden ser difíciles de leer, entender, mantener y depurar.

4ª generación: Según el IEEE (1990), se puede definir un lenguaje de esta generación como aquél desarrollado para mejorar la productividad de un lenguaje de la tercera generación, que muchas veces, permite a los no-programadores aprovechar la funcionalidad disponible en el ordenador sin escribir programas complejos. Los rasgos de un lenguaje de esta generación son: un sistema de gestión de base de datos integrado, un lenguaje de consulta, un generador de informes y gráficos, la posibilidad de producir modelos financieros, la funcionalidad de una hoja de cálculo y algunas funciones estadísticas.

5ª generación: Según el IEEE (1990), se puede definir un lenguaje de esta generación como el que incorpora los conceptos de sistemas basados en el conocimiento, sistemas expertos y el procesamiento del lenguaje natural. Todavía no existe ningún lenguaje que realmente se pueda definir como de quinta generación de propósito general.

2.6 Principales paradigmas de programación

En esta sección se resumen los principales paradigmas de programación y se realiza una breve caracterización de los mecanismos de algunos lenguajes de programación. A la hora de describir las características de un lenguaje de programación, compararlo con otros o situarlo dentro de un paradigma, se hace referencia a los mecanismos que tiene. En algunos casos (como en el paradigma imperativo), muchos de los mecanismos vienen impuestos directamente por la arquitectura de la máquina y en otros casos (como en el paradigma funcional), los mecanismos vienen impuestos por una filosofía. Esto no quiere decir que la mayoría de los lenguajes de programación no tengan la mayoría de los mecanismos, sino solamente que el grado de acceso a ellos que tiene el programador, y su visibilidad y naturaleza implícita o explícita, dependen de cada uno.

Hay que tener en cuenta es que la lista de mecanismos no ha sido siempre fija, sino que ha ido evolucionando y creciendo. Y no hay ninguna razón para pensar que no va a seguir expandiéndose con avances en investigación y la implantación de nuevos lenguajes e incluso nuevos paradigmas basados en nuevas metáforas de computación.

2.6.1 Programación imperativa

La programación imperativa nació junto con los primeros ordenadores y lenguajes en los años 50 y 60 y reflejaba directamente la arquitectura moderna del ordenador, basada en lo que propusó von Neumann. Se puede caracterizar un lenguaje como imperativo si dispone de las siguientes propiedades (Sethi, 1992; Tucker y Noonan, 2002): el concepto básico es el estado de la máquina; un programa consiste en un conjunto de instrucciones que se ejecuta y que cambia el valor de alguna variable o posición de memoria; tipos de datos para números reales, caracteres, cadenas, boléanos y sus operadores; enunciados de control de flujo, como por ejemplo: *for*, *while*, *case* e *if*; estructuras de datos que se puede asignar; comandos para controlar la entrada y salida de datos; punteros y procedimientos y funciones.

Casi todos los lenguajes de propósito general son de este tipo. Como ejemplos representativos en esta categoría se pueden destacar: FORTRAN, Algol, COBOL, Pascal, Modula 2, PL/I, C y Ada. A continuación se presentará un breve resumen de los aspectos más notables, basado en las varias fuentes bibliográficas (Pratt y Zewkowitz, 2001; Sethi, 1992; Wilson y Clark, 2001; Sebesta, 2002):

> **COBOL**: *Common Business Oriented Language* (lenguaje común orientado a los negocios) fue desarrollado a finales de los años 50 y aparecieron compiladores para él a principios de los años 60. Difería considerablemente

de otros LP de la época como FORTRAN y Algol, por estar muy orientado a los negocios.

Pascal y **Modula 2**: Pascal fue desarrollado a finales de los años 60 y principios de los 70 por Niklaus Wirth, basado en su experiencia con Algol W. Quiso producir un LP que se pudiera implementar eficientemente en la mayoría de los ordenadores y que sirviera para la enseñanza de los LP. Wirth desarrolló **Modula** y luego **Modula 2** a partir de Pascal, con una mejora en la sintaxis y semántica respecto a Pascal, donde la mayor extensión consistía en la existencia de módulos para facilitar la reutilización del código del lenguaje.

C: Solucionó los problemas de resistencia a lenguajes de alto nivel por parte de los programadores de sistemas. Aunque originalmente estaba muy vinculado al sistema operativo UNIX, en los años 80 salieron versiones de compiladores para otros sistemas y quizás hoy en día sea el lenguaje con compiladores para la mayor gama de SO y máquinas.

Ada: El Departamento de Defensa de los EEUU decidió que quería un nuevo lenguaje que se pudiera usar para controlar sistemas complejos, grandes, con cierto grado de paralelismo, y que cambiaron con el tiempo. La definición del nuevo lenguaje, llamado Ada, apareció en 1983 y es el único en la historia que fue un estándar antes de la aparición del primer compilador (que apareció en 1986). Fue la primera vez que aparecieron pruebas oficiales para un compilador, pero años más tarde, la empresa Sun hizo algo parecido para el Java, pero con un pequeño cambio de énfasis: lo hizo para comprobar qué programas no utilizan elementos que forman parte de la versión oficial del lenguaje.

2.6.2 Programación orientada a objetos

En los años 70 había un acuerdo sobre la necesidad de facilitar la reutilización de código. Y como resultado hubo un esfuerzo para extender el concepto de la abstracción de los programas imperativos (abstracción de procedimientos y funciones) para incluir datos y tipos de datos abstractos. Se consiguió proporcionar abstracción de datos con un mecanismo de encapsulamiento que limitó tanto la visibilidad como el ámbito de los datos y las funciones definidas por ellos. Aun así, con los módulos y paquetes, todavía hubo problemas porque no existían mecanismos para la inicialización y terminación automática de instancias de estos tipos, ni tampoco para extender los tipos de datos abstractos para añadir nuevas funciones. La programación orientada a objetos (a partir de ahora POO) nació de

las ideas originales del lenguaje Simula (tratado a continuación) y el concepto de "clase" formaba una base para solucionar los problemas anteriores.

La POO recibió mucho interés en los años 80 y se convirtió en el paradigma dominante a finales de los años 90 (Booch, 1994). Al principio había muchos lenguajes orientados a objetos (a partir de ahora, LPOO), muchos de los cuales incorporaron aspectos de la programación funcional. A continuación se hará un resumen de algunos de los principales LPOO, es decir, Simula, Smalltalk, Eiffel, C++ y Java:

Simula: Al principio de los años 60 un grupo en Noruega estaba trabajando en un lenguaje para llevar a cabo simulaciones de sistemas. Eventualmente su trabajo dio a luz Simula 67. Se puede decir que este lenguaje fue el padre de lo que es el paradigma de la programación orientada a objetos (a partir de ahora, OO) porque definió el concepto de clase como un objeto que empaqueta tanto los datos como los procedimientos y las funciones que los manipulan.

Smalltalk: fue el resultado del trabajo de Alan Kay y sus colegas para producir un ordenador personal para no expertos, llevado a cabo al principio de los años 70. El lenguaje en sí formaba parte de un entorno de desarrollo visual y era difícil distinguir entre los límites del lenguaje y el entorno. Smalltalk seguía el desarrollo del concepto de clase de Simula aunque recibió mucha influencia también de Lisp. Hasta la mitad de los años 80 cuando la programación OO empezó a ser muy popular, Smalltalk era el LPOO por defecto.

Eiffel: Fue diseñado por Betrand Meyer a mediados de los años 80 con el objetivo de ser un lenguaje que apoyara la creación de sistemas grandes y robustos. La gran diferencia con Smalltalk era su sistema de tipos estáticos y la comprobación de tipos durante el proceso de compilación, algo importante para producir sistemas robustos.

C++: Fue desarrollado por Bjarne Stroustrup desde 1979 a 1985 para añadir el mecanismo de clases de Simula al lenguaje C. Ha tenido un impacto muy importante en la evolución de C y en la forma de compilar programas escritos en C en un compilador de C++ con pocos cambios. Ha servido como un puente entre el mundo de la programación OO y el mundo industrial. No es un lenguaje completamente OO, pero ha resultado uno de los más utilizados.

Java: Fue desarrollado por Sun Microsystems para programar dispositivos electrónicos en 1990/1. Como lenguaje es más sencillo y pequeño que C++, y se desarrolló sobre la base de lo que los diseñadores pensaban que eran las peores características de C++ (en el sentido de causar problemas a la hora de

programar). Como características importantes del lenguaje, adicionales a las de la orientación a objetos, se pueden destacar dos: en primer lugar, el papel central otorgado a las bibliotecas del LP a través de la API (*Application Programming Interface*) de Java. En segundo lugar, se puede destacar el hecho de que no se compila un programa en este lenguaje a código de máquina, sino a un código intermedio que se puede considerar como un lenguaje de máquina para una máquina virtual de Java.

2.7 Entornos de desarrollo integrados

Los entornos de trabajo facilitan un editor de código con acceso a las bibliotecas del lenguaje, un compilador y un depurador, todos integrados de manera que desde el editor se puede dar el comando para compilar el código y depurar simbólicamente (es decir, ir desde el editor línea por línea en el programa, inspeccionando las variables y estructuras de datos: lo que se llama IDE [*Integrated Development Environment* – entorno integrado de desarrollo]).

El gran adelanto que tuvo lugar en los años 90 con estos entornos de desarrollo fue la inclusión de herramientas visuales para el diseño de la interfaz gráfica de la aplicación. Así que un programador podía dibujar la interfaz que quería, con botones, cajas de textos, menús despegables, etc., y a la vez el entorno produciría el código fuente. Se llamó a dichos entornos *herramientas de desarrollo rápido* (RAD – *rapid application development*) y hoy en día casi siempre es la herramienta de elección por parte de los programadores. Hay que tener un poco de cuidado con ellos, porque en el caso de las herramientas (y lenguajes) como Delphi, Visual BASIC y Visual C++, los API son para el SO Microsoft Windows y por lo tanto los programas generados (especialmente la interfaz gráfica) no funcionará en otras plataformas. Esto no ocurre con las herramientas para Java porque, como ya se ha dicho, Java fue desarrollado para funcionar en muchas plataformas, incluso con la misma interfaz gráfica.

El entorno de desarrollo recomendado por el equipo docente en esta asignatura, **jGRASP**, es sencillo y versátil (admite incorporar compiladores de distintos lenguajes, no sólo Java) además de tener un interfaz de usuario muy amigable para cualquier usuario de entornos de ventanas (tipo Windows, Linux, Mac, etc).

2.8 Ingeniería del Software

En la primera época de las computadoras el desarrollo del software era puramente artesanal. Pero a mediados de los años setenta en el siglo XX el hardware ofrecía cada vez mayores posibilidades. En los nuevos sistemas software que se comenzaban a desarrollar, la complejidad era ya un factor a tener en cuenta y las técnicas artesanales ya no servían y era necesario hacer una planificación de cara a rentabilizar el esfuerzo. Pero la falta de control sobre el desarrollo de estos sistemas desembocó en lo que hoy se conoce como crisis del software (Gibbs, 1994). El problema alcanzó tal magnitud que en la reunión del Comité Científico de la OTAN que tuvo lugar en 1968 se propuso una solución que consistía en tratar el desarrollo de software con las técnicas ingenieriles utilizadas por las industrias fabricantes de productos "físicos". Así, la ingeniería del software se puede definir como la aplicación de una aproximación sistemática, disciplinada y cuantificable al desarrollo, operación y mantenimiento del software (Pressman, 2002).

El proceso de construcción del software requiere, como cualquier otra ingeniería, identificar las tareas que se han de realizar sobre el software y aplicar esas tareas de una forma ordenada y efectiva. Adicionalmente el desarrollo del software se debe realizar por un conjunto coordinado de personas simultáneamente, y por lo tanto sus esfuerzos deben estar dirigidos por una misma metodología que permita estructurar las diferentes fases del desarrollo. Algunas empresas y universidades han desarrollado a lo largo del tiempo varias metodologías, es decir, de modos sistemáticos de producir software. Todas ellas tienen en común la intención de automatizar el proceso de desarrollo para que el producto resultante cumpla una serie de requisitos (tiempo de desarrollo, calidad, eficiencia, precio, mantenibilidad). Las metodologías dividen la realización de un proyecto en una serie de etapas que son: especificación, diseño, codificación, pruebas y mantenimiento. Las distintas formas de ordenar el esfuerzo a lo largo del tiempo son los ciclos de vida.

En concreto, el software adopta varias formas en distintos momentos de su ciclo de vida:

- Código fuente: escrito por programadores. Contiene el conjunto de instrucciones destinadas a la computadora.
- Código objeto: resultado de compilar el código fuente, como si fuera una traducción de éste último. El código objeto no es directamente inteligible por el ser humano, pero tampoco es directamente entendible por la computadora.

- Código ejecutable: resultado de enlazar uno o varios fragmentos de código objeto. Constituye un archivo binario con un formato tal que el sistema operativo es capaz de cargarlo en la memoria de un ordenador, y proceder a su ejecución. El código ejecutable es directamente inteligible por la computadora.

Por último, cabe mencionar dos tipos de herramientas que se han producido que marcan nuevos tipos de sistemas de programación que podrían ser muy importantes en el futuro de la programación. En primer lugar, las herramientas que permiten a un usuario no programador construir programas utilizando una herramienta visual y paletas de componentes que se pueden conectar para especificar el flujo del programa sin escribir una línea de código (por ejemplo, JavaStudio). En segundo lugar, las herramientas que permiten el diseño de programas utilizando **UML** (*Unified Modelling Language* o Lenguaje Unificado de Modelado) y después producen automáticamente el código fuente del programa. Estas herramientas ofrecen la posibilidad de mantener el problema del desarrollo de programas a un nivel abstracto donde es posible llevar a cabo análisis sobre el diseño sin bajar al nivel del código.

Como resumen de las etapas para la creación de un software, se pueden mencionar:

- Análisis.
- Desarrollo.
- Construcción.
- Pruebas (unitarias e integradas).
- Paso a Producción.

Dentro de estas etapas, existen sub-etapas (para algunos son otras etapas, como por ejemplo, paso a ambiente beta/rc).

Recursos Didácticos Complementarios - Referencias

DOCUMENTOS	FORMATO
Guía de Estudio Rodrigo, C. y Delgado, J.L. "521069 Fundamentos de Informática. Guía de Estudio" (2006)	PDF
LIBROS	
Introducción a la Computación Norton, P. Ed. MgGraw-Hill (2006) ISBN: 970-10-5108-4.	LIBRO
Introducción a la Ciencia de la Computación Forouzan, B.A. Ed. Thomson (2003) ISBN: 970-686-285-4.	LIBRO
RECURSOS WEB	
Wikipedia [URL] http://www.wikipedia.org/	URL

EVALUACIÓN

Cuestionario de opinión sobre el Tema 2

ASPECTOS A VALORAR	PUNTUACIÓN (1-10)
ORGANIZACIÓN DEL DOCUMENTO	
Enfoque Práctico del Documento	
Claridad en la exposición	
Utilidad de los contenidos	
Adecuación a los objetivos	
Tamaño	
ORGANIZACIÓN DE LAS EPÍGRAFES	
Estructuración de los contenidos	
Dinamismo en la presentación de contenidos	
Utilidad práctica de los contenidos	
Organización del material didáctico, actividades y prácticas	
Adecuación a objetivos	
OTROS DATOS DE INTERÉS	
Extensión total del tema	
Valoración global del tema	
COMENTARIOS	

Por favor, enviar por correo electrónico al Equipo docente: fundinfor@lsi.uned.es