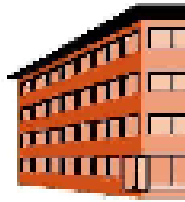
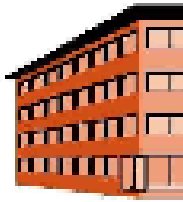
UT1:Conceptos básicos sobre programación

Abraham Pérez Barrera



ÍNDICE.



1. Los sistemas de procesamiento de la información.

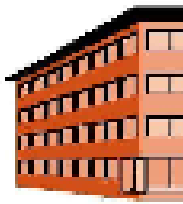
¿Qué es la información?

- Es todo aquello que posea algún interés para nosotros. Ejemplo: qué tiempo hace hoy, el precio del pan, el IVA que se aplica a un producto.
- Los **datos** son una representación simbólica de una variable.
- Un **dato** por sí mismo no constituye **información**, es el **procesamiento** de los **mismos** lo que nos proporciona **información**.

1. Los sistemas de procesamiento de la información.

Sistemas de procesamiento de la información.

- Un **sistema** se define de forma general como un conjunto de componentes conectados que interactúan, que tiene un propósito.
- Los **sistemas de procesamiento de información** son sistemas que transforman los **datos** en **información** organizada, significativa y útil.
- Los componentes de un **sistema de proceso de información** son tres: **entrada**, **proceso** y **salida**.



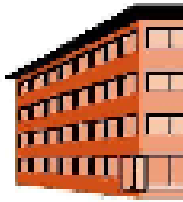
1. Los sistemas de procesamiento de la información.

- La **entrada** son los **datos** que el **sistema** requiere. Los proporciona el usuario al **sistema**.
- El **procesamiento** de los **datos** lo lleva a cabo el procesador del ordenador.
- La **salida** es la **información** que nos suministra el procesador después de haber procesado los datos.

1. Los sistemas de procesamiento de la información.

Representación de la información.

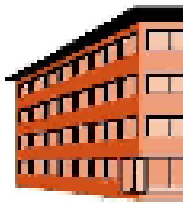
- Independientemente del **sistema** que procese la **información**, debemos saber que la información está **representada por símbolos.** Según éstos podemos clasificar la información así:
  - **Datos numéricos**: 0 al 9.
  - Información de tipo **alfabético**: letras.
  - **Datos alfanuméricos**: combinación de los anteriores y caracteres especiales.



1. Los sistemas de procesamiento de la información.

Representación de la información.

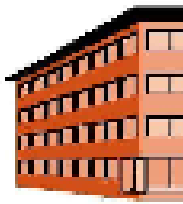
- Dependiendo del tratamiento que se le dé a la **información** esta puede ser:
  - **Analógica**: la **información** se representa por un número infinito de valores.
  - **Digital**: la **información** se representa por un número finito de valores. Un **dígito** es un número que puede expresarse usando una sola cifra.
- Ejemplos:
- Dígitos con 10 estados se llaman dígitos decimales y la representación se llama decimal. Es la que usa el hombre.
- Dígitos con 2 estados se llaman dígitos binarios y la representación se llama binaria. Es la que usa el **ORDENADOR** internamente.



1. Los sistemas de procesamiento de la información.

Tratamiento automático de la información.

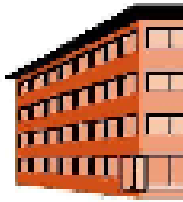
- Con el tiempo el hombre se ha dado cuenta de que los pasos para realizar la mayoría de sus actividades son siempre los mismos, lo que varía es la información que se procesa. Estos pasos son : recoger la **información**, tratarla, **procesarla**, elaborar resultados y utilizarlos.
- Por tanto, **todos los procesos pueden ser automatizados.**
- Ventajas:
  - Ahorro eficiente de recursos humanos y técnicos.
  - Obtención de resultados fiables.
  - Cometer el menor número posible de errores.



1. Los sistemas de procesamiento de la información.

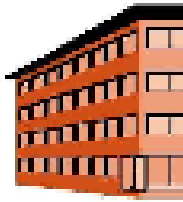
Tratamiento automático de la información.

- Surge el concepto de **informática:** es un conjunto de técnicas que tienden a facilitar el tratamiento automático de la información y el **ordenador** es la máquina capaz de realizar ese tratamiento.
- Fundamentalmente, un ordenador trabaja con dos tipos de informaciones:
  - Unas **instrucciones** que le dicen qué debe hacer.
  - Unos **datos** sobre los que opera. Pueden ser: numéricos, alfabéticos y alfanuméricos.



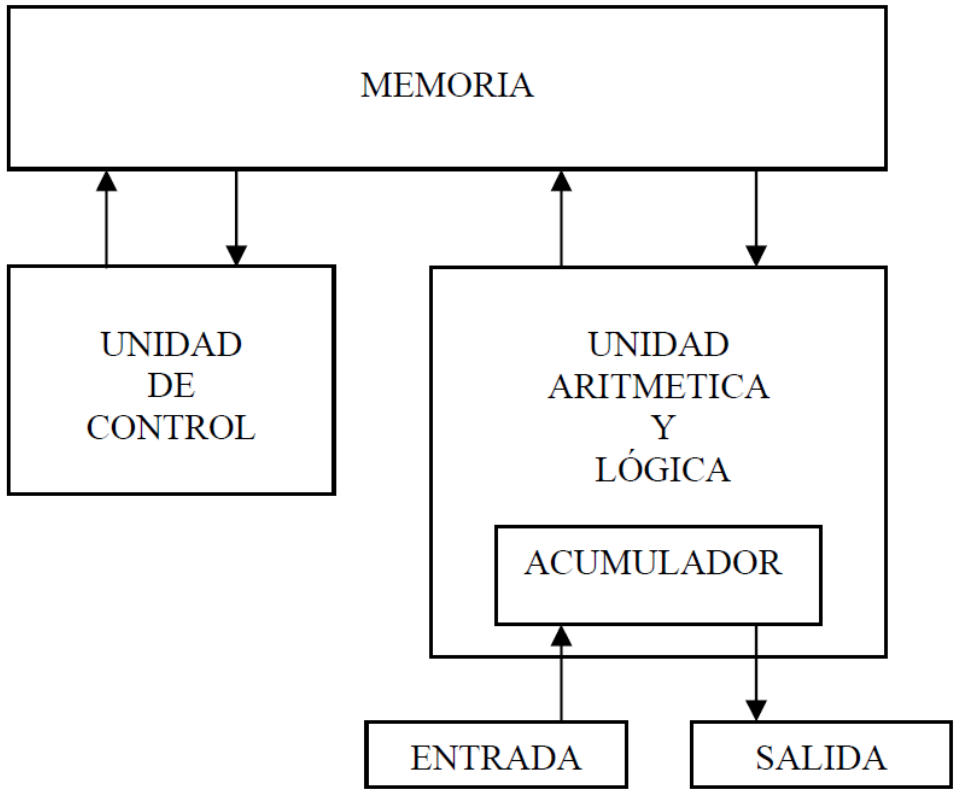
Elementos básicos de un ordenador: modelo Von Neumann.

Abraham Pérez Barrera



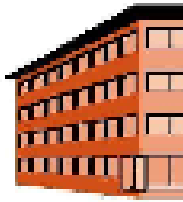
2. Elementos básicos de un ordenador: modelo Von Neumann.

- La máquina original de **Von Neumann** respondía al siguiente esquema:



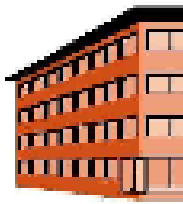
2. Elementos básicos de un ordenador: modelo Von Neumann.

- La mayoría de las computadoras de hoy en día están basadas en el modelo diseñado por John Von Neumann.
- La máquina de Von Neumann consta de cinco partes básicas:
  - La **Memoria**.
  - La **unidad aritmética lógica**.
  - La **unidad de Control**.
  - La **unidad** de **Entrada**.
  - La **unidad** de **Salida**.



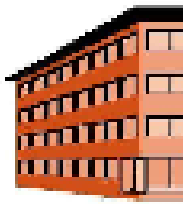
2. Elementos básicos de un ordenador: modelo Von Neumann.

- La **unidad de control (UC)** toma las **instrucciones** de la **unidad de memoria** de una en una y las interpreta. Después envía las señales apropiadas a todas las otras unidades para que se ejecute la **instrucción** especificada.
- La **unidad aritmética-lógica (UAL)** realiza todas las operaciones **aritméticas** y **lógicas** con los **datos** y envía los resultados a la **unidad de memoria** para que se almacenen.

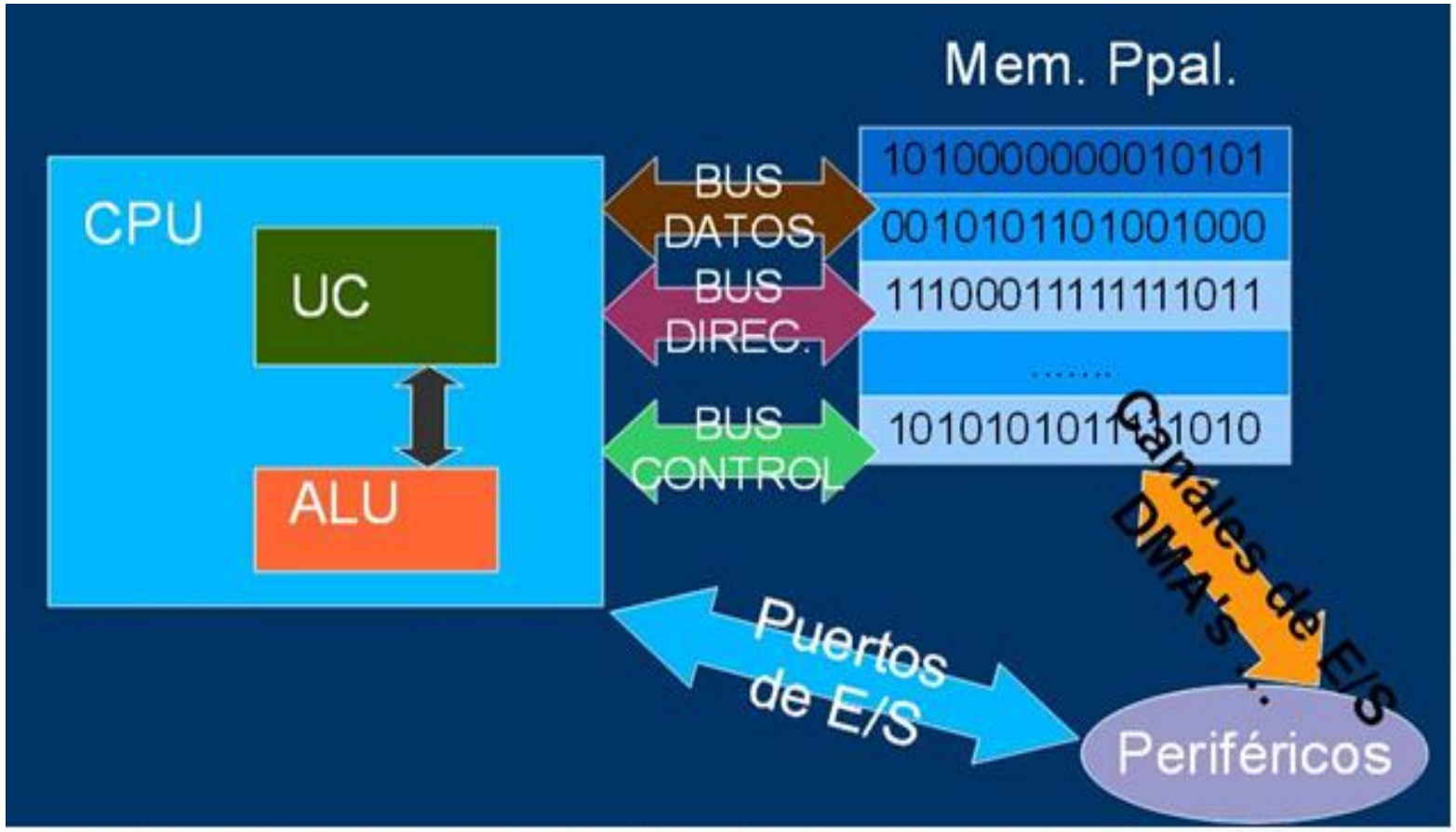


2. Elementos básicos de un ordenador: modelo Von Neumann.

- La **memoria** almacena las **instrucciones** y los **datos** recibidos de la **unidad** de **entrada** y guarda los **datos** que provienen de la **unidad aritmética-lógica**.
- La **unidad de entrada** almacena los **datos** y las **instrucciones**. Por lo regular, **datos** e **instrucciones** ingresan en la **unidad de memoria** por ratón o teclado.
- La **unidad de salida** toma los **datos** de la **memoria** para mostrárselos al usuario (generalmente por pantalla).

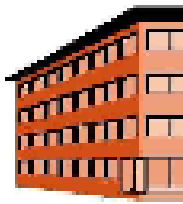


2. Elementos básicos de un ordenador: modelo Von Neumann.



2. Elementos básicos de un ordenador: modelo Von Neumann.

- Los **datos** se mueven entre los diferentes componentes del ordenador a través del **bus de datos**.
- Otros **buses** importantes son el **bus de direcciones** (que establece la dirección de **memoria** del **dato** que se transfiere) y el de **control** (que gobierna el acceso al de **datos** y al de **direcciones**).
- **Puertos de E/S**: permite realizar transferencias de información entre el exterior y la CPU.
- Otro elemento importante en un ordenador son los **periféricos**, que permiten que el usuario introduzca (entrada) o lea **datos** (salida). Tres tipos de **periféricos**:
  - De **entrada**. Ejemplo: el ratón.
  - De **salida**. Ejemplo: la pantalla.
  - De **entrada/salida**. Ejemplo: el disco duro.
- Canales de E/S, DMA's: son rutas del sistema usados por muchos dispositivos para transferir información directamente a la memoria en ambos sentidos.

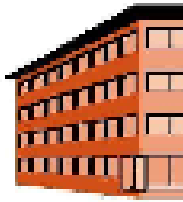


3. Sistemas de Numeración.

- Un **sistema de numeración** es un conjunto de **símbolos** y **reglas de generación** que permiten construir todos los números válidos en el sistema. Ejemplos de numeración son el decimal y el binario, de los que ya hemos hablado.
- Un **sistema de numeración es <u>posicional</u>** si el valor de final de un número depende de la posición de los **símbolos**. Por ejemplo, los números 654 y 456 utilizan los mismos dígitos pero son números diferentes porque la posición que ocupa cada uno de ellos no es la misma. El 4 no vale lo mismo en el primero, donde representa 4 **unidades**, que en el segundo, donde representa 4 **centenas**.

3. Sistemas de Numeración.

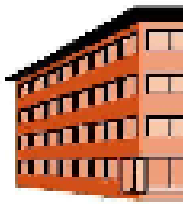
- El número de **símbolos** permitidos en un sistema de **numeración posicional** se conoce como **base** del **sistema de numeración**. Ejemplo: el **sistema de numeración decimal** es de **base** 10 porque utiliza 10 **símbolos** (0, 1, 2, 3, 4, 5, 6, 7, 8 y 9).
- ¿Cuál es la base del **sistema binario**? 2: los **símbolos** son el 0 y el 1.



3. Sistemas de Numeración.

- La forma en la que se construyen los números en un **sistema de numeración posicional** se rige por el **teorema fundamental de la numeración.**
- La fórmula general para construir un número *N*, con un número finito de decimales, en un **sistema de numeración posicional** de **base** *b* es la siguiente:

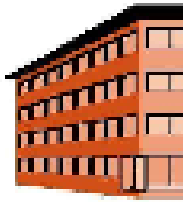
$$N = \begin{cases} \langle d_{(n-1)} \dots d_1 d_0, d_{-1} \dots d_{-k} \rangle = \sum_{i=-k}^{n} d_i b^i \\ N = d_n b^n + \dots + d_1 b^1 + d_0 b^0 + d_{-1} b^{-1} + \dots + d_{-k} b^{-k} \end{cases}$$



3. Sistemas de Numeración.

- El significado de cada uno de los elementos de la fórmula es el siguiente:
  - N→ número válido en el **sistema de numeración**.
  - b→ **base** del **sistema de numeración**. Número de **símbolos** permitidos en el sistema.
  - d_i→ un **símbolo** cualquiera de los permitidos en el **sistema de numeración**.
  - n→ número de **dígitos** de la parte entera.
  - ,→coma fraccionaria, **símbolo** utilizado para separar la **parte entera** de un número de su **parte fraccionaria**.
  - k→ número de **dígitos** de la **parte decimal**.

$$N = \begin{cases} \langle d_{(n-1)} \dots d_1 d_0, d_{-1} \dots d_{-k} \rangle = \sum_{i=-k}^{n} d_i b^i \\ N = d_n b^n + \dots + d_1 b^1 + d_0 b^0 + d_{-1} b^{-1} + \dots + d_{-k} b^{-k} \end{cases}$$



3. Sistemas de Numeración.

- Un ejemplo en el **sistema decimal**. Recordemos que la **base** era 10.

  El **teorema fundamental de la numeración** se aplicaría como

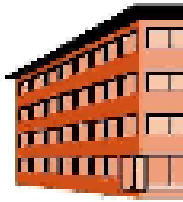
$$N = d_n \dots d_1 d_0, d_{-1} \dots d_{-k} =$$

$$d_n \cdot 10^n + \dots + d_1 \cdot 10^1 + d_0 \cdot 10^0, + d_{-1} \cdot 10^{-1} + \dots + d_{-k} \cdot 10^{-k} =$$

$$N = \sum_{i=-k}^{n} d_i \cdot 10^i$$

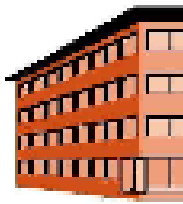
- El número 1492,36; se expresaría como

$$1492,36 = 1 \cdot 10^3 + 4 \cdot 10^2 + 9 \cdot 10^1 + 2 \cdot 10^0, + 3 \cdot 10^{-1} + 6 \cdot 10^{-2}$$



3. Sistemas de Numeración.

- Como trabajar con cadenas de ceros y unos es muy engorroso, agrupamos los **bits** (los **dígitos** del **sistema binario** reciben el nombre de **bits**):
  - De TRES EN TRES -> tenemos el **sistema octal** o **base** 8, el nº de símbolos 8 (del 0 al 7)
  - De CUATRO EN CUATRO -> **sistema hexadecimal o base** 16, el nº de símbolos 15 (del 0 al 9, A,B,C,D,E,F).

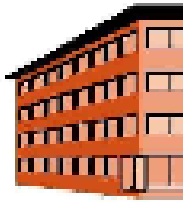


3. Sistemas de Numeración.

NOTA:

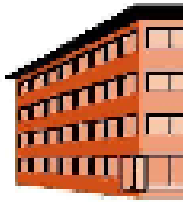
Hemos hablado de la representación de números naturales pero también existe lo siguiente:

- Representación de números negativos. Ejemplo: complemento a 1, complemento a 2,…
- Representación de números reales. Ejemplo: punto fijo, coma flotante,…
- Representación alfanumérica. Ejemplo: ascii, unicode,…



4. Programas y Algoritmos.

- Al salir de fábrica, las CPU's sólo saben realizar por sí mismas un número limitado de operaciones **lógicas** y **aritméticas**.
- Nosotros deseamos realizar operaciones más complicadas y tenemos que hacerlo utilizando únicamente las operaciones que sepa realizar la CPU. Así que tenemos que decirle muy explícitamente lo que queremos que haga .
- Podemos definir **algoritmo** como los pasos que hay que dar para resolver un problema. En nuestro caso, utilizamos los **algoritmos** para decirle al ordenador lo que tiene que hacer.
- Cuando describimos esos pasos en un **lenguaje de programación** concreto, obtenemos un **programa**.



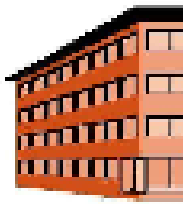
4. Programas y Algoritmos.

¿Qué es un ALGORITMO?:

Conjunto de pasos que seguimos para resolver un problema.

- Definición más formal: secuencia precisa, finita y bien definida de instrucciones que nos llevan a resolver un problema dado.
  - Precisa: Sin ambigüedad. Debe indicar el orden de realización de cada paso.
  - Finita: tiene un punto de finalización.
  - Bien definida: todas las veces que se siga el **algoritmo** con los mismos datos, se debe obtener el mismo resultado.

Ejemplos: pasos para la resolución de una ecuación de 2º grado, una receta de cocina.



4. Programas y Algoritmos.

Características de un ALGORITMO.

- La definición de un algoritmo debe describir tres partes:
  - Entrada. Ejemplo receta: ingredientes y utensilios.
  - Proceso. Ejemplo receta: proceso de elaboración de la receta.
  - Salida. Ejemplo receta: el plato cocinado.
- Una **vez** realizado el **algoritmo**, se traducirá a un programa comprensible por la máquina que vaya a ejecutar el **algoritmo**.

4. Programas y Algoritmos.

- Son independientes del **lenguaje de programación** en que se expresan y del ordenador que los ejecuta. Ejemplo: receta de un plato de cocina, los pasos son los mismos independientemente del idioma en el que esté y del cocinero que la realice.

4. Programas y Algoritmos.

Programa.

- Un **programa** es una secuencia ordenada de **INSTRUCCIONES**, comprensibles por una computadora, que manipulan **DATOS** para realizar una función determinada.

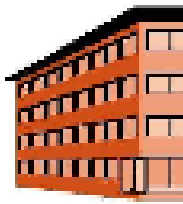
- En memoria tendremos:

  - **Datos**: de entrada, salida e intermedios.

  - **Instrucciones**: operaciones que debe realizar la CPU, sobre qué datos debe hacerlos y dónde almacenar los resultados.

Programación.

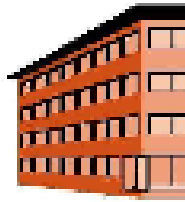
Proceso que consiste en escribir, probar, analizar y definir el código de un **programa**.



4. Programas y Algoritmos.

- Por ejemplo: los paso para realizar un sandwich de jamón y queso podrían ser:

1. Coger una rebanada de pan.
2. Colocar encima de la rebanada un loncha de queso.
3. Colocar encima de la loncha de queso una de jamón.
4. Colocar encima de la loncha de jamón otra de queso.
5. Colocar encima de la loncha de queso otra rebanada de pan.
6. Calentar dos minutos en el microondas a media potencia.



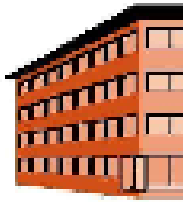
4. Programas y Algoritmos.

Actividad 1: Realizar un **algoritmo** para freír un huevo.

5. Lenguajes de Programación.

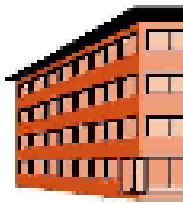
¿Qué es un lenguaje de programación?

- El idioma es el lenguaje con el que nos comunicamos los seres humanos.
- Un **lenguaje de programación** es el "idioma" que sirve para comunicarse a los seres humanos con el ordenador.
- Permite expresar las **instrucciones** que el **programador** desea que la computadora ejecute.



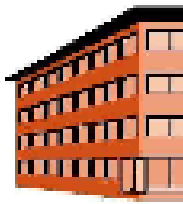
5. Lenguajes de Programación.

- Los **lenguajes de programación** se pueden clasificar atendiendo a varios criterios.
- Los principales criterios son: el **nivel de abstracción** y la **forma de ejecución**.
- Según su nivel de abstracción se dividen en lenguajes de **bajo nivel** y lenguajes de **alto nivel**.
- Según la forma de ejecución, se dividen en **compilados** e **interpretados**.



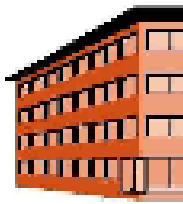
5. Lenguajes de Programación.

- Los **lenguajes de bajo nivel** aprovechan al máximo las condiciones de la máquina en la que se van a ejecutar.
- Los lenguajes de más bajo nivel son el **lenguaje máquina** y el **lenguaje ensamblador**.
- Los lenguajes de **alto nivel** se caracterizan por traducir los **algoritmos** a un lenguaje más fácil de entender por el programador.



5. Lenguajes de Programación.

- Para ejecutar un programa hay que traducirlo al **lenguaje máquina**. Según como se realice la traducción tendremos,
  - **Lenguajes compilados**: se traducen a través de un programa que se llama **compilador**, que genera un fichero en **lenguaje máquina**. Si el programa no presenta errores, la traducción sólo se lleva a cabo una vez.
  - **Lenguajes interpretados**: un programa **intérprete** traduce el programa cada vez que hay que ejecutarlo. No genera un fichero, por eso hay que traducirlo cada vez.



5. Lenguajes de Programación.

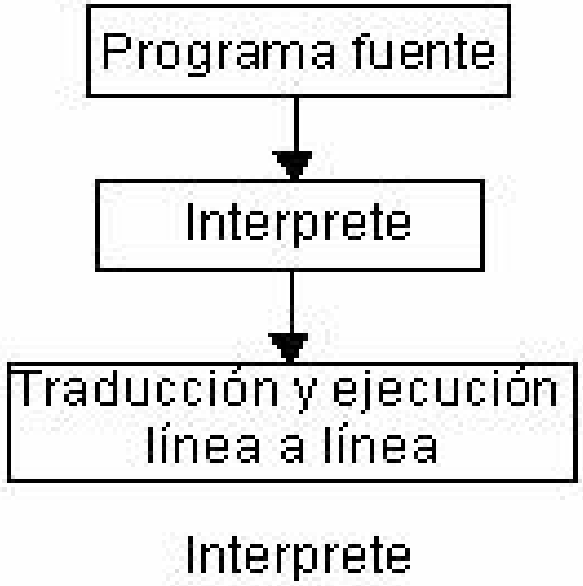
- Un **compilador** toma todas las **instrucciones** del lenguaje del nivel más alto y obtiene todo el código del lenguaje de nivel más bajo. Al compilar, se obtiene un lenguaje ensamblador, así que necesitamos un programa llamado MONTADOR O ENLAZADOR O LINKER para convertirlo a lenguaje máquina. Un ejemplo de lenguajes compilado es C.

Fases de la compilación

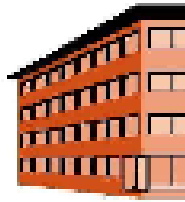
5. Lenguajes de Programación.

- El **intérprete** toma las instrucciones una a una, las traduce y ejecuta.

Ejemplos de lenguajes interpretados son Basic, JavaScript, PHP,...



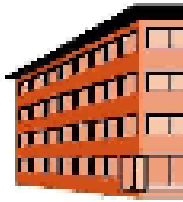
Interprete



5. Lenguajes de Programación.

Diferencia fundamental entre un compilador y un intérprete:

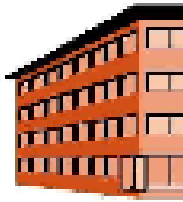
- Los compiladores realizan la traducción en tiempo de desarrollo. Es decir, el programa aún no se está ejecutando. El compilador recibe todo el código fuente, lo analiza, lo optimiza generando un fichero ejecutable pero escrito en lenguaje máquina listo para su ejecución.
- Los intérpretes realizan la traducción del programa cada vez que se quiera ejecutar (en tiempo de ejecución). Es decir, a medida que el programa se va ejecutando, el intérprete va traduciendo instrucciones al lenguaje máquina. Estos no generan un fichero ejecutable por eso hay que realizar la traducción cada vez que se ejecute el programa.



5. Lenguajes de Programación.

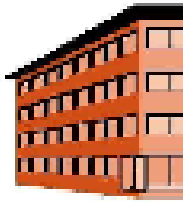
Ventajas e inconvenientes:

- La ventaja de los compilados es que no hace falta traducir el programa cada vez pero su inconveniente es que no podremos ejecutar el programa hasta que no haya ningún error.
- Los intérpretes van traduciendo y ejecutando instrucción a instrucción, lo cual hace que se pueda ejecutar el programa aunque haya errores. Cuando llegue a la instrucción errónea, se parará el programa.



5. Lenguajes de Programación.

- Un ejemplo de la diferencia entre un compilador y un intérprete visto desde el punto de vista de los idiomas, sería:
  - Compilación: traducir todo un texto de inglés a español.
  - Intérprete: imaginamos tres personas, una habla inglés, otra español y otra ambos idiomas (el intérprete). El intérprete va traduciendo al momento.

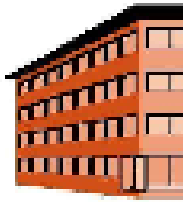


5. Lenguajes de Programación.

Java obtiene las ventajas de los lenguajes compilados y de los lenguajes interpretados de la siguiente forma:

1º) Realiza la fase de compilación traduciendo el programa fuente a un lenguaje intermedio. Dicha traducción se guarda en un fichero.

2º) Dicho fichero es traducido al **lenguaje máquina** cada vez que se quiera ejecutar. Sería la fase de **interpretación**.

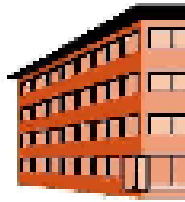


5. Lenguajes de Programación.

Actividad 2: ¿qué entiendes por **compilador**?

Actividad 3: ¿cómo explicarías qué es un **intérprete**?

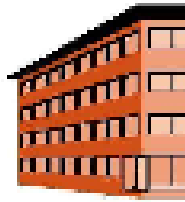
Actividad 4: ¿qué es un **enlazador** o **linker**?



5. Lenguajes de Programación.

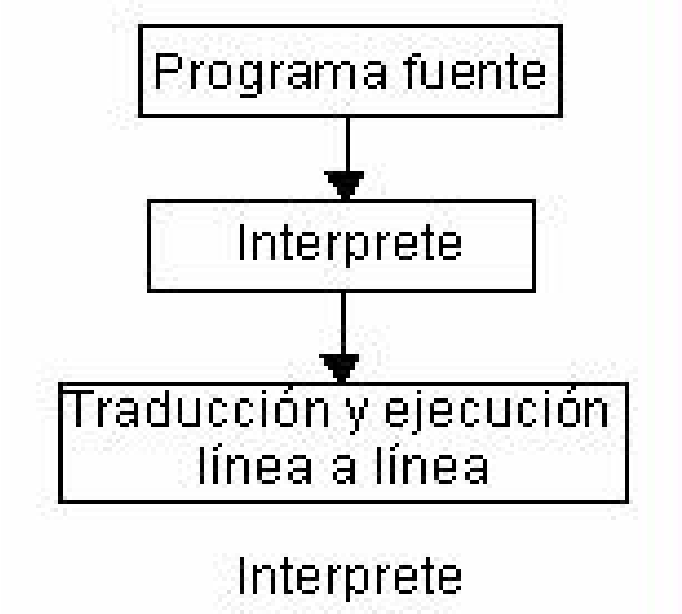
Actividad 5: explica el diagrama de la diapositiva 32.

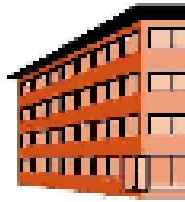
Fases de la compilación



5. Lenguajes de Programación.

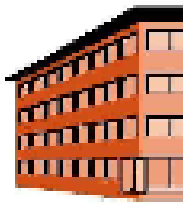
Actividad 6: explica el diagrama de la diapositiva 33.





5.1. Generaciones de los lenguajes de programación.

- PRIMERA GENERACIÓN: LENGUAJES MÁQUINA.
  - Son lenguajes de bajo nivel.
  - Los lenguajes de esta generación son los **lenguajes máquina**.
    - Las **instrucciones** dependen del hardware, por lo que cada modelo de ordenador tiene su propio código. Esto supone que no se pueden transportar de una máquina a otra.
    - Las **instrucciones** están formadas por secuencias de ceros y unos que «entiende» el ordenador pero no el usuario. Por lo tanto, es muy difícil de desarrollar y mantener.
  - Un programa escrito en **lenguaje máquina** puede ser manejado directamente por el ordenador.

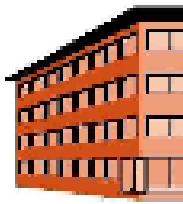


5.1. Generaciones de los lenguajes de programación.

- SEGUNDA GENERACIÓN: LENGUAJES ENSAMBLADORES.
  - Son lenguajes de bajo nivel.
  - Los lenguajes de esta generación son los ENSAMBLADORES.
    - Las **instrucciones** son conocidas como MNEMOTÉCNICOS.

      Ejemplos de mnemotécnicos: SUM, RES, DIV, etc.

      Ejemplo de instrucción: SUM M, N, P
  - Un programa escrito en **lenguaje ensamblador** no puede ser ejecutado directamente por el ordenador. Debe traducirse primero por un **traductor** a **lenguaje máquina**.



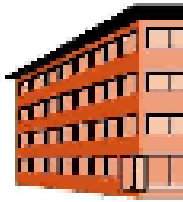
5.1. Generaciones de los lenguajes de programación.

- Dependen del ordenador, por lo que no se pueden transportar de una máquina a otra.
- Difícil desarrollo porque lleva mucho tiempo formar a los programadores para trabajar con una máquina en concreto cada vez.
- Son difíciles de mantener.

5.1. Generaciones de los lenguajes de programación.

➢ TERCERA GENERACIÓN: LENGUAJES DE ALTO NIVEL.

- Son los más usados porque están diseñados para que las personas escriban y entiendan los programas de un modo sencillo usando instrucciones más parecidas al lenguaje del usuario (READ, WRITE, PRINT, OPEN).
- Un programa escrito en **lenguaje de alto nivel** es independiente de la máquina, por lo que se pueden ejecutar en diferentes tipos de ordenadores.
- Al ser el desarrollo más simple, el tiempo de formación de los programadores es más corto.



5.1. Generaciones de los lenguajes de programación.

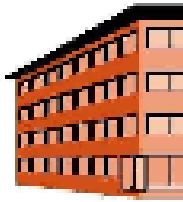
- El mantenimiento es más sencillo, por lo que se reduce el coste de los programas.
- Un programa escrito en lenguaje de **alto nivel** no puede ser ejecutado directamente por el ordenador. Debe traducirse primero a **lenguaje máquina** que es el lenguaje que entiende el ordenador por un **traductor**.
- Inconvenientes:
  - Aumento de ocupación en memoria.
  - Tiempo de ejecución mayor.
- Ejemplos: PASCAL, C, C++, JAVA.

5.1. Generaciones de los lenguajes de programación.

- CUARTA GENERACIÓN.

Características:

- Permiten elaborar programas en menor tiempo, lo que conlleva a un aumento de la productividad.
- El personal que elabora software sufre menos agotamiento, ya que generalmente requiere escribir menos.
- Mantenimiento más simple.
- Alta transportabilidad.
- Ejemplos: Focus, SQL, PowerBuilder.



5.1. Generaciones de los lenguajes de programación.

Actividad 7: explica a qué nos referimos cuando hablamos de la **transportabilidad** de los programas.

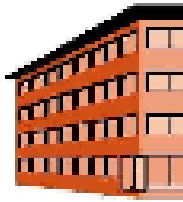
Actividad 8: busca en Internet qué es un lenguaje de **quinta generación**.

Actividad 9: busca en Internet 5 ejemplos de lenguajes de **alto nivel**.

Actividad 10: ¿a qué tipo de lenguaje podría pertenecer una instrucción del tipo «MOV AL,61h»?

6. Programación. Tipos. Calidad en los programas.

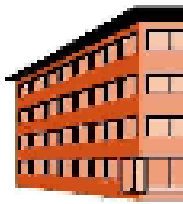
- Tipos de programación (entre otros):
  - **Estructurada**: utiliza únicamente subrutinas y tres estructuras (secuencial, selectiva e iterativa).
  - **Modular**: consiste en dividir un programa en módulos o subprogramas con el fin de hacerlo más legible y manejable.
  - **Orientada a objetos**: usa los objetos y sus interacciones para diseñar aplicaciones y programas informáticos.
  - **Concurrente**: simultaneidad en la ejecución de múltiples tareas interactivas.



6. Programación. Tipos. Calidad en los programas.

Las características generales que debe reunir un programa son las siguientes:

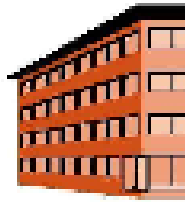
- **Legibilidad**: el programa debe estar escrito de forma clara y sencilla, de tal forma que facilite su lectura y comprensión.
- **Fiabilidad**: el programa debe controlar todo tipo de situaciones que puedan ocurrir durante la ejecución.
- **Portabilidad**: debe poder ejecutarse en cualquier máquina con diferentes sistemas operativos, es decir, en cualquier plataforma.
- **Modularidad**: el programa debe estar formado por partes.
- **Eficiencia**: se deben aprovechar al máximo los recursos del ordenador, minimizando la memoria utilizada y el tiempo de ejecución.
- **Documentación**: todo programa debe estar debidamente documentado tanto de forma externa como interna.



7. Ciclo de vida de una aplicación informática.

¿Qué es una aplicación informática?

- Es un programa diseñado para facilitar al usuario la realización de un determinado trabajo.
- Son parte del software de una computadora y se ejecutan sobre el sistema operativo.
- Ejemplos: Word, aplicación de nóminas, aplicación para inscribirse a un curso online, aplicación para hacer la matrícula del curso.
- En general, es un programa escrito en cualquier **lenguaje de programación**.



7. Ciclo de vida de una aplicación informática.

Ciclo de vida de una aplicación informática.

Son las etapas por las que pasa desde que se plantea su creación hasta que se termina.

1ª) Análisis (¿QUÉ?):

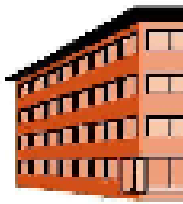
- Definición del problema.
- Estudio de viabilidad.
- Obtener un modelo detallado.

2ª) Diseño: ¿CÓMO?

3ª) Implantación: codificación del programa.

4ª) Pruebas: comprobar que el programa hace lo que debe.

5ª) Explotación y mantenimiento: instalar el programa y mantenerlo (parches).



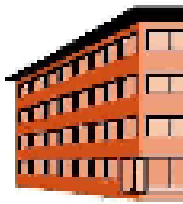
8. Errores en el desarrollo del software.

- Las pruebas del software son una tarea que consiste en comprobar si un programa funciona correctamente.
- Los errores pueden ser de 4 tipos:
  - Errores de **CODIFICACIÓN**: son errores de sintaxis. Ejemplo: palabras reservadas mal escritas, instrucciones incompletas, etc. Se producen en la fase de compilación.
  - Errores de **EJECUCIÓN**: se dan cuando se le pide al programa que haga algo imposible y falla durante su ejecución. Ejemplo: dividir por 0, raíz cuadrada de números negativos, etc.
  - Errores **LÓGICOS**: son errores que se producen cuando el programa no hace lo que tiene que hacer. Se detectan haciendo varias pruebas y se comparan los resultados.
  - Errores de **ESPECIFICACIÓN**: son errores producidos por un mal entendimiento de cuáles eran los requisitos que debía de cumplir el programa.

9. Documentación de los programas.

- Es muy importante que todos los programas vayan acompañados de una documentación. Dos tipos:
  - **Externa**: donde se describe,
    - La actividad que se ha informatizado.
    - Las entradas del programa
    - Las salidas del programa.
    - El código fuente.
  - **Interna**: son los llamados comentarios que se ponen dentro del programa con el fin de hacer todas las aclaraciones necesarias para dar claridad al código.

- Gracias a la **documentación**, se consigue que el mantenimiento y el uso del programa sean mucho más fáciles.



Actividad 11: Busca en Internet tres lenguajes de programación que usen programación orientada a objetos.

Actividad 12: ¿Qué ventajas tiene la programación modular?

Actividad 13: ¿En qué fase valorarías si existe el hardware necesario para realizar una aplicación?

Actividad 14: ¿En qué fase se llevarían a cabo las actualizaciones de la aplicación?

Actividad 15: ¿Ante qué tipos de errores se comportan de forma diferente un compilador y un intérprete?

Actividad 16: ¿Qué tipo de documentación sería un manual de usuario?