

Introducción a la programación

04/10/2016

Andrei García

*2016*

# Conceptos previos

## Definición informática:

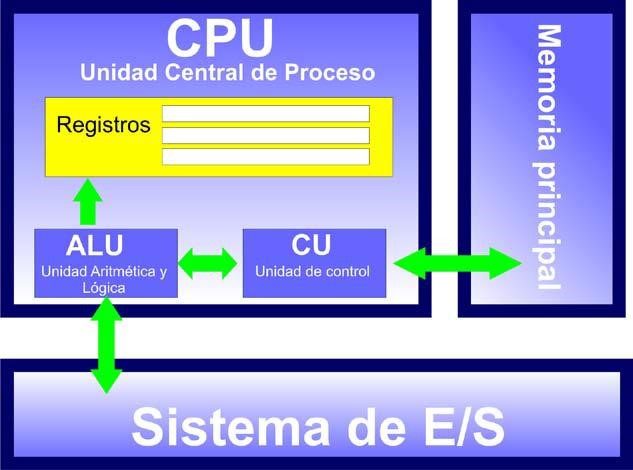
– Conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de computadoras electrónicas.

## Definición computadora:

- Máquina capaz de aceptar unos datos de entrada, efectuar con ellos operaciones lógicas y aritméticas, y proporcionar la información resultante a través de un medio de salida.

# 2. Arquitectura básica del PC

Basada en el concepto de Von Newmann en 1945: “Crear una máquina capaz de ejecutar instrucciones leídas de memoria y ejecutadas en la CPU comunicándose con el exterior a través de la Unidad de E/S”



* **Memoria Principal:** espacio de almacenamiento temporal dividido en celdas de igual tamaño destinado a almacenar instrucciones y datos.
* Unidad Central de Proceso (**CPU**): Se encarga de la ejecución de las instrucciones almacenadas en la memoria.
* **Unidad Aritmética**: encargada de realizar las operaciones aritméticas (sumas, restas, ...) y lógicas (and, or, not, ...).
* **Unidad de Control**: Su función es decodificar las instrucciones del programa en ejecución y generar todas las señales necesarias para que puedan ser ejecutadas.
* **Unidad de E/S**: permite la comunicación de la CPU y la memoria con el exterior: impresora, monitor, teclado, ...
* **Registros**: almacén temporal que se usa durante la ejecución de las instrucciones.

# Tipos de dispositivos

**Dispositivos de Entrada**: permiten al usuario introducir la información en el sistema. Los datos leídos se almacenan en memoria. Ejempos: teclado, ratón...

**Dispositivos de Salida**: son los encargados de mostrar los resultados obtenidos por la computadora al usuario. Ejemplos: monitor, altavoz, ...

**Dispositivos de Entrada/Salida** : permiten tanto la entrada de información en la computadora como la salida de la misma. Ejemplos: tarjeta de red, módem, etc.

# Sistemas operativos respecto a programación

Un Sistema Operativo es un programa o conjunto de programas que actúa como intermediario entre el usuario y el hardware del ordenador, ocultando su complejidad mediante una interfaz sencilla de utilizar. Sus objetivos son:

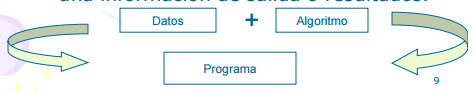
* Hacer cómoda la utilización de la computadora y facilitar la programación, sobretodo en lenguajes de alto nivel.
* Utilizar recursos de la computadora de forma eficiente.

## Funciones del sistema operativo

* Gestión de procesos
* Gestión de memoria
* Gestión de dispositivos de E/S
* Gestión del sistema de ficheros
* Gestión de la red
* Protección

La mayoría de estas funciones las realiza el **kernel**.

# El ordenador y los algoritmos

Un ordenador es una máquina que ejecuta algoritmos (Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema). La ejecución o procesamiento de un algoritmo supone la transformación de una información de salida o resultados. 

Una tarea ejecutable por un ordenador da lugar a un problema algorítmico. El algoritmo es la solución a ese problema (software).

Cuando un algoritmo es ejecutado por los circuitos de un ordenador (hardware) se origina un proceso que genera resultados a partir de los datos. Un proceso se caracteriza por una sucesión de estados de determinadas magnitudes que están almacenadas en la memoria RAM, y que llamaremos variables.

## Características de los algoritmos

Un algoritmo debe resolver el problema para el que fue formulado. Los algoritmos son independientes del ordenador. Los algoritmos se escriben para poder ser utilizados en cualquier máquina, ya que lo traduce un lenguaje compilador común, siendo estos precios y con cálculos exactos y finitos. Éstos algoritmos han de ser repetidos si se es requerido.

## Algoritmos: Tipos de datos

Los algoritmos se construyen utilizando elementos simples para que el lenguaje se parezca más al de las computadoras. Los datos pueden ser:

Números (**INT,FLOAT**) (10, 25, 5.32)

Textos (**STRING**) (“Hola mundo”)

Lógicos (**BOOLEAN**) (true, false)

Datos compuestos (**ARRAY**) (registros, listas)

## Algoritmos: Expresiones

**Variables**: Nombre sensible a mayúsculas y minúsculas que posee un valor de cualquier tipo.

**Operadores lógicos**: Condicionantes para comparar datos. (+, -, \*, >,<,<=,>=,/).

## C:\Users\Andrei\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_1.pngAlgoritmos: Pasos previos

**Diagramas de flujo**: Representación gráfica del proyecto, comando o función a programar.

If -> $nombreCliente has A

Execute -> “El nombre de cliente posee A”

Else -> $nombreCliente dones’t has A

Execute -> “El nombre de cliente no posee A”

**Pseudocódigo**: Representación textual simplificada del proyecto, comando o función a programar. Se usan las funciones del lenguaje de programación indicado, comúnmente: if, else, while, for, foreach, switch, case…

# Tipos de lenguajes de programación

## Según eje temporal

### Lenguajes de primera generación

A principio de los 50 se usaba el lenguaje máquina y el ensamblador.

### Lenguajes de segunda generación

A finales de los 50 principios de los 60 se amplía el uso de los lenguajes para las aplicaciones científico-militares y aparecen las aplicaciones de gestión. Fortran, Cobol y Algol

### Lenguajes de tercera generación

nacen con la programación estructurada y presentan potentes posibilidades de estructuración de datos y procedimientos. Pascal, C, etc.

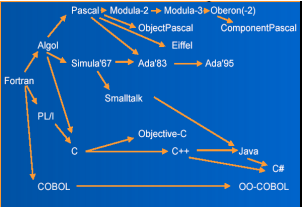
### Lenguajes de cuarta generación

Combinan características procedurales y no procedurales. SQL, OASIS, lenguajes de generación de informes, etc.

Según su proximidad al lenguaje máquina

Lenguaje máquina: cada instrucción se codifica como una secuencia de 1's y 0's, lo que dificulta el desarrollo de programas complejos. (**COMPLEJO**).

Lenguajes de bajo nivel: la única ventaja de estos lenguajes es que ocupan poco espacio en memoria y se ejecutan con mayor rapidez.

Lenguaje ensamblador: cada instrucción en ensamblador equivale a una instrucción en lenguaje máquina. Utiliza palabras mnemotécnicas en lugar de cadenas de bits. (**SENCILLO**).

Lenguaje de alto nivel: Independencia de la máquina, se aproxima al lenguaje natural, incluye funciones y comandos de frecuente repetición.

# Tipos de lenguajes de programación

## Según funcionalidad

**Para la educación**: Logo, Pilot, Eiffel, ...

**De propósito general**: concebidos no para una aplicación concreta, sino para ser eficaces en diversos campos. Pascal, C.

**Lenguajes de inteligencia artificial**: creados para la emulación de programas que emulan comportamientos inteligentes. Lisp, Prolog.

**Orientados a la gestión**: las aplicaciones de gestión se caracterizan por tener la necesidad de manejar grandes volúmenes de información de forma fiable. Cobol, Clipper, PL/SQL, ...

**Científicos**: se caracterizan por realizar pocas operaciones de E/S y por el contrario una gran cantidad de cálculos complejos. Fortran, Apl, ...

**Orientados a Internet**: son lenguajes transversales a la máquina y al sistema operativo. PHP, Java.

**Orientados a tiempo real**: pensados para dar respuesta a exigentes requisitos temporales.

## Según traducción a código máquina

* **Interpretados**: “Sentencia traducida, sentencia ejecutada”. Toma un programa fuente y lo va traduciendo y ejecutando simultáneamente. (PHP)

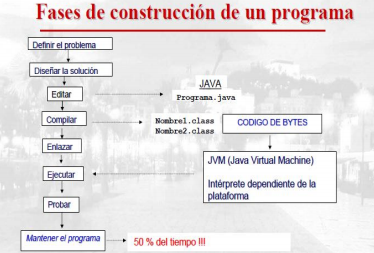
**Ventajas**: posee una sintaxis más elaborada que la de los ensambladores. Además la velocidad de traducción depende fundamentalmente de la sintaxis del lenguaje. Cuanto más simple, más rápido será el análisis de cada sentencia.

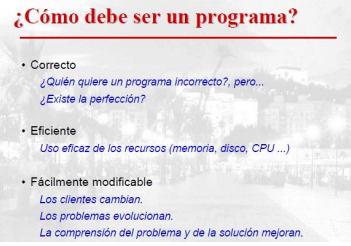
**Inconvenientes**: una sentencia que tenga que ejecutarse varias veces tendrá que traducirse también varias veces. La forma de evitar estos retardos consiste en adoptar una estrategia diferente a la traducción-ejecución. Consiste en traducir el programa completo y después ejecutarlo. Así surgen los compiladores.

* **Compilados**: un compilador es un traductor que genera un programa objeto (programa escrito en un lenguaje de bajo nivel) a partir de un programa fuente (programa en lenguaje de alto nivel). (C++)

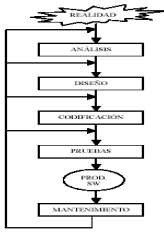
**Ventajas**: Los recursos son mejor administrados y está preparado para un sistema operativo específico, haciéndolo más rápido (si está programado eficientemente).

**Desventajas**: Has de incluir cada librería usada en cada archivo del proyecto, no es multiplataforma, es más complicado que los interpretados.





# Ciclo de vida de un programa

Las distintas fases del ciclo de vida de un programa se realizan secuencialmente, en cada una de las etapas se genera una documentación que servirá para iniciar la siguiente. A este proceso se le conoce como ciclo de vida clásico o en cascada.

En la elaboración de aplicaciones existen unas fases a las que se denominan Ciclo de vida, ya que una vez que llegamos a la última fase puede ser que sea necesario comenzar el ciclo de nuevo en otra fase anterior.

## Análisis

Estudio de la situación y requisitos existentes. Se establece la viabilidad del proyecto. Existen diversas técnicas para realizar el análisis estructurado:

* Diagramas de flujos de datos
* Modelos de datos
* Diccionarios de datos
* Definición de las interfaces de usuario

## Diseño

Se establece una solución óptima con los suficientes detalles para la realización física de la aplicación. Se pueden establecer las siguientes etapas:

1. Diseño externo
2. Diseño de datos
3. Diseño modular
4. Diseño procedimental (Al final de esta fase se obtiene el cuaderno de carga).

## Codificación

El objetivo de estas pruebas es garantizar que el sistema ha sido desarrollado correctamente, sin errores de diseño y/o programación. Es conveniente que sean planteadas al menos tanto a nivel de cada módulo (aislado del resto), como de integración del sistema y adaptación al idioma requerido.

## Explotación

Esta etapa tiene como objetivo la verificación de que el sistema desarrollado cumple con los requisitos expresados inicialmente por el cliente y que han dado lugar al presente proyecto. Se ha de formar a los usuarios (para la correcta utilización del sistema) y documentación necesaria para la explotación del sistema y el manual de uso o guía de la aplicación.

## Mantenimiento

“DEPURAR, CORREGIR, MEJORAR Y ADAPTAR la aplicación si no responde exactamente a la solución buscada”. El mantenimiento ha de ser:

* Correctivo
* Adaptativo
* Perfectivo

# Métodos de programación

* **Intérprete**: Toma un programa fuente escrito en un lenguaje de alto nivel y lo va traduciendo y ejecutando simultáneamente.
* **Compilador**: Es un traductor que genera un programa objeto a partir de un programa fuente escrito en un lenguaje de alto nivel.
* **Depuradores**: Es una herramienta que permite depurar o limpiar los errores de un programa fuente. Normalmente esta herramienta está integrada en el entorno de programación del lenguaje.
* **RAD** (Rapid Application Development): Se suele usar para referirnos a IDEs (Herramientas de Desarrollo Integrado) como Delphi, Foxpro, Eclipse, Visual Basic, etc.
* **CASE** (Computer Aided SW Engineering): Persigue proporcionar un conjunto de herramientas, bien integradas, que enlacen y automaticen todas las fases del ciclo de vida del SW.