Programación

Introducción

Sistemas Informáticos

Cuando el procesamiento de los datos y la información en un **Sistema de Información** lo realiza un ordenador lo llamamos **Sistema Informático**.

En un Sistema Informático el algoritmo se describe mediante un programa.

Programa: Conjunto ordenado de instrucciones que se dan al ordenador indicándole las operaciones o tareas que ha de realizar para resolver un problema.

Para escribir programas utilizamos **lenguajes de programación**

Programación

Un **programador** es una persona que resuelve problemas, y para llegar a ser un programador eficaz se **necesita aprender a resolver problemas de un modo riguroso y sistemático**:

- *Definición o análisis del problema*: consiste en el estudio detallado del problema. Se debe identificar los datos de entrada, de salida y la descripción del problema.
- *Diseño del algoritmo*: que describe la secuencia ordenada de pasos que conduce a la solución de un problema dado: **algoritmo**.
- *Transformación del algoritmo en un programa (codificación)*: Se expresa el algoritmo como un programa en un **lenguaje de programación**.
- *Ejecución y validación del programa*.

Desarrollo de programas - Etapas

- **Análisis**: Entender el problema.
- **Diseño**: Creamos el algoritmo: pseudocódigo.
- **Codificación**: Escribir el algoritmo en un lenguaje de programación (**Código fuente**).
- **Ejecución y validación**: Comprobamos que el programa resuelve el problema planteado.

Análisis del problema

Para realizar el análisis del problema realizamos varios pasos:

- **Definición del problema**: Tenemos que entender y comprender la naturaleza del problema, tenemos que dominar los conceptos con los que estamos trabajando y conocer que operaciones hay que realizar para solucionar el problema y obtener la información de salida.
- **Especificación de los datos de entrada**: Hay que determinar que datos de entrada necesitamos para resolver el problema.
- **Especificación de los datos de salida**: También hay que determinar los datos de salida que van a ofrecer la solución del problema.

Diseño de algoritmos

A partir de los requerimientos, resultados del análisis, empieza la etapa de **diseño** donde tenemos que construir un **algoritmo** que resuelva el problema.

Un **algoritmo** es un conjunto de acciones que especifican la secuencia de operaciones realizar, en orden, para resolver un problema.

Los algoritmos son independientes tanto del lenguaje de programación como del ordenador que los ejecuta.

Las *características* de los algoritmos son:

- Un algoritmo debe ser preciso e indicar el orden de realización de cada paso.
- Un algoritmo debe estar definido. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.
- Un algoritmo debe ser finito. Si se sigue un algoritmo, se debe terminar en algún momento; o sea, debe tener un número finito de pasos.

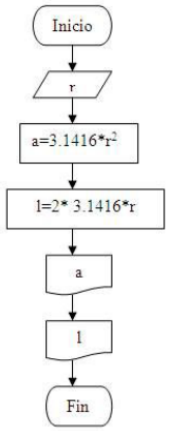
Herramientas de representación de algoritmos

- Un **diagrama de flujo** es una de las técnicas de representación gráfica de
algoritmos más antiguas. Ventajas: permite altos niveles de estructuración y modularización y es fácil de usar. Desventajas: son difíciles de actualizar y se complican cuando el algoritmo es grande.
- El **pseudocódigo**, nos permite una aproximación del algoritmo al lenguaje natural y por tanto un a redacción rápida del mismo.

Fórmulas:
$a = 3.1416\, r^2$
$l = 2*3.1416*r$

Pseudocódigo y Diagrama de flujo

Inicio
  Leer r
  $a = 3.1416*r^2$
  $l = 2*3.1416*r$
  Escribir a
  Escribir l
Fin algoritmo

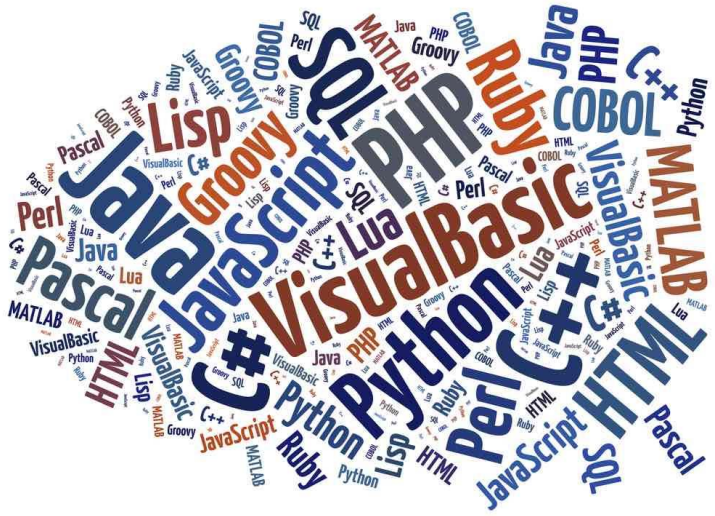


Lenguajes de programación

Para que un ordenador realice un proceso, se le debe suministrar en primer lugar un algoritmo adecuado, que llamamos **programa**.

El procesador debe ser capaz de interpretar el algoritmo, lo que significa:

- Comprender las instrucciones de cada paso.
- Realizar las operaciones correspondientes.



Tipos de lenguajes de programación

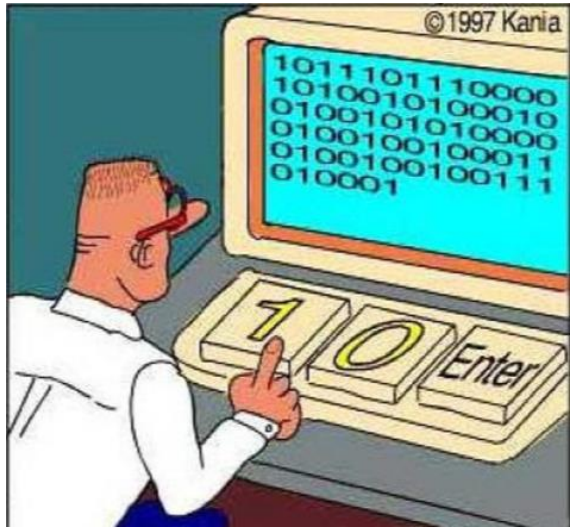
- Lenguaje Máquina
- Lenguaje de bajo nivel (Ensamblador)
- Lenguaje de alto nivel
- Programas traductores
  - Compiladores
  - Intérpretes
  - Máquina virtual

Lenguaje Máquina

Son aquellos que están escritos en lenguajes que directamente entiende el ordenador, ya que sus instrucciones son cadenas binarias (secuencias de
ceros y unos) que especifican una operación y las posiciones (dirección) de memoria implicadas en la operación. Se denominan instrucciones de máquina o **código máquina**.

Características:

- Las instrucciones en lenguaje máquina dependen del hardware del ordenador y por tanto serán diferentes de un ordenador a otro.
- Se puede transferir un programa a memoria sin necesidad de traducción posterior, lo que supone una mayor velocidad de ejecución a cualquier otro lenguaje.
- Dificultad y lentitud en la codificación.
- Conjunto de instrucciones reducido (operaciones muy elementales)



Lenguaje de bajo nivel (Ensamblador)

Los lenguajes de bajo nivel son más fáciles de utilizar que los lenguajes máquina, pero, al igual que ellos, dependen de la máquina en particular.

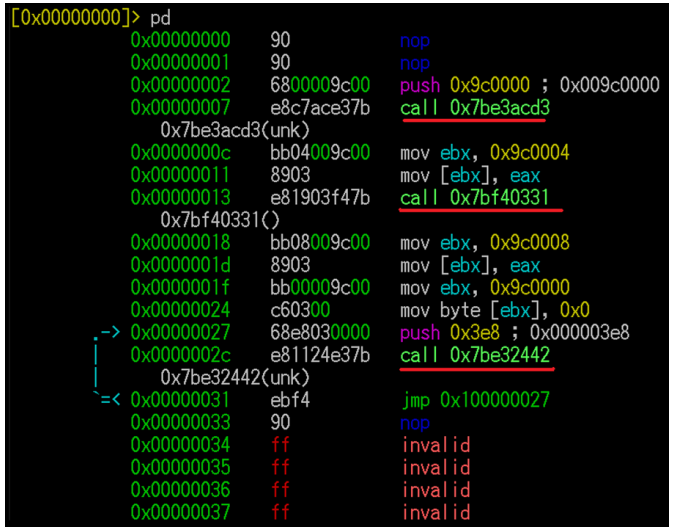
El lenguaje de bajo nivel por excelencia es el **ensamblador**.

Características:

- Cada línea de código se traduce en una instrucción para la máquina
- Repertorio de instrucciones reducido (operaciones muy elementales
- Programas difíciles de entender

Las instrucciones en lenguaje ensamblador son instrucciones conocidas como ***mnemónicos***.

Un programa escrito en lenguaje ensamblador no puede ser ejecutado directamente por la máquina, sino que requiere una fase de traducción al lenguaje máquina. El programa original escrito en lenguaje ensamblador se denomina **programa fuente** y el programa traducido en lenguaje máquina se conoce como **programa objeto**, ya directamente entendible por el ordenador.



Lenguaje de alto nivel

Los lenguajes de alto nivel son los mas utilizados por los programadores. Están diseñados para que las personas escriban y entiendan los programas de un modo mucho más fácil que los lenguajes máquina y ensambladores. Otra razón es que un programa escrito en un lenguaje de alto nivel es independiente de la máquina, o sea, las instrucciones del programa del ordenador no dependen del diseño hardware de un ordenador en particular. Por lo tanto los programas escritos en lenguajes de alto nivel son portables o transportables, lo que significa la posibilidad de poder ser ejecutados con poca o ninguna modificación en diferentes tipos de ordenadores.

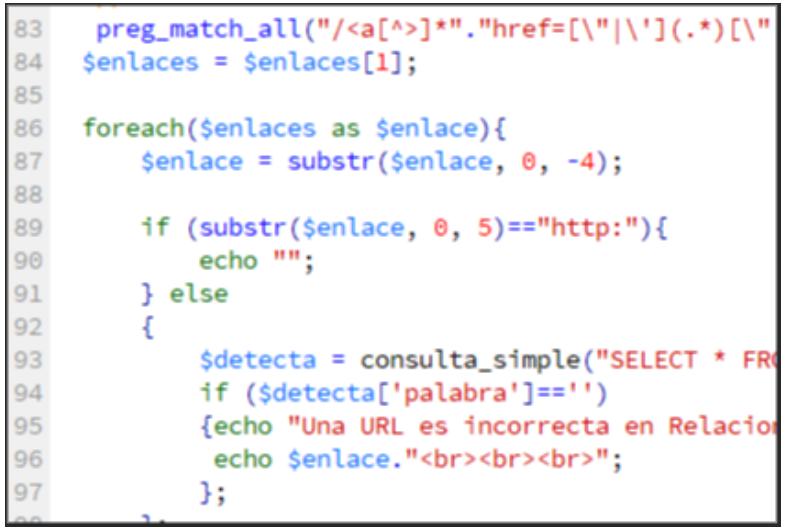
Características:

- Independientes del modelo de ordenador
- Proporcionan un mayor nivel de abstracción

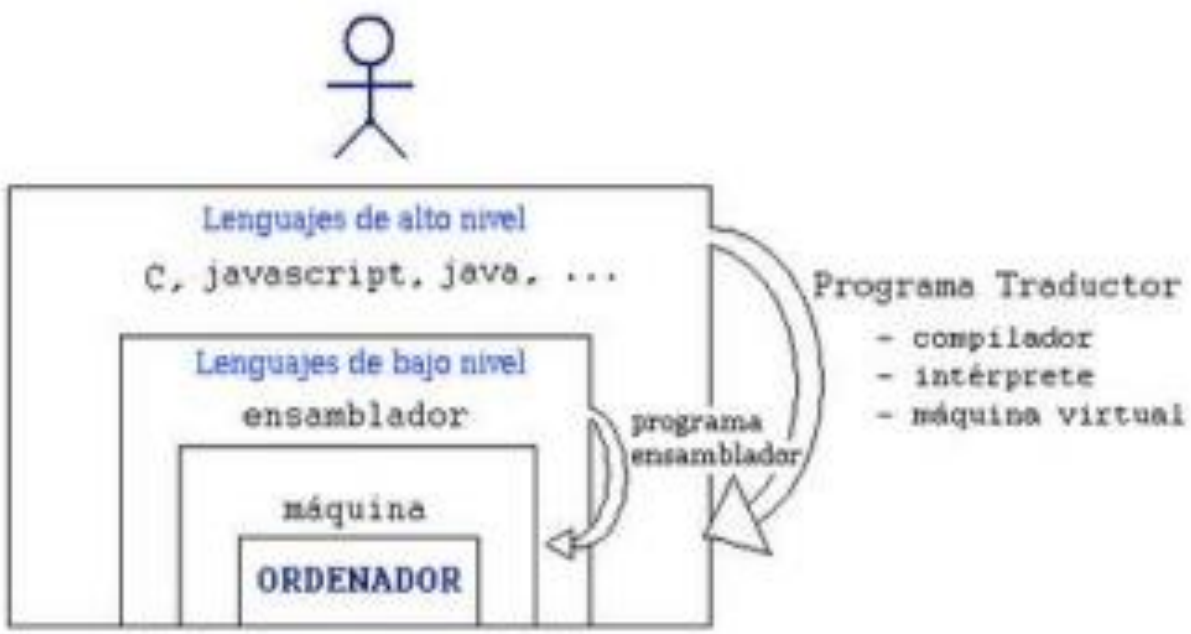
Ejemplos de lenguajes de programación de alto nivel:

BASIC, COBOL, PASCAL, C, VISUAL BASIC, JAVA, PYTHON, PERL, GO, PHP, RUBY,…

Al igual que sucede con los lenguajes ensambladores, los programas fuente tienen que ser traducidos por **programas traductores**, llamados en este caso compiladores o intérpretes.



Programas traductores



Los traductores transforman programas escritos en un lenguaje de alto nivel en programas escritos en código máquina. Podemos indicar distintos tipos:

Compiladores

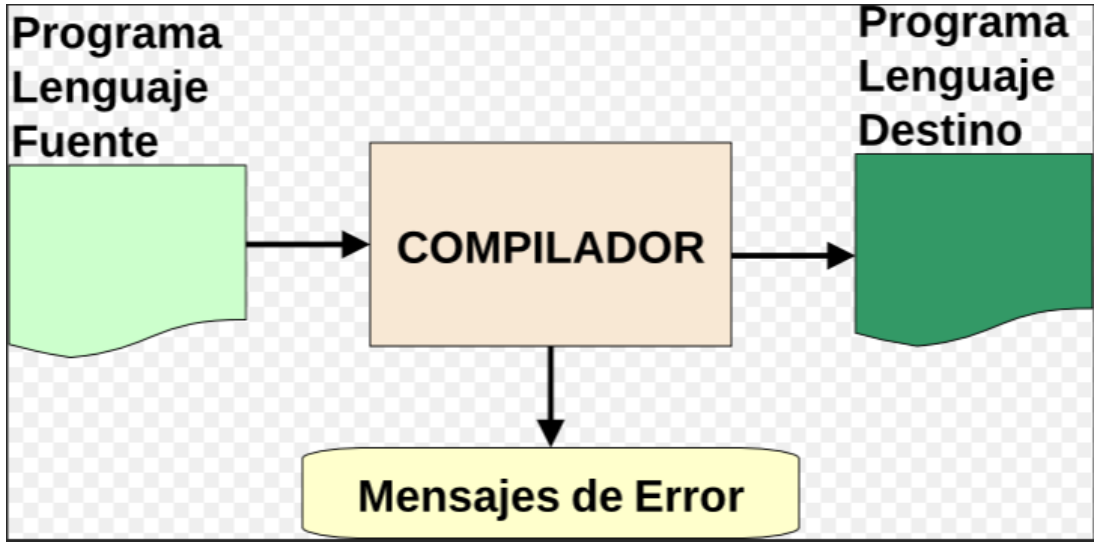
Intérpretes

Máquina virtual

Compilador

- Convierte un programa escrito en alto nivel (código fuente) a un programa máquina (código ejecutable).
- Para generar el código ejecutable el código no debe tener errores de sintaxis.
- Necesitamos un compilador para cada arquitectura y sistema operativo.
- Los programas ejecutables no son compatibles entre plataformas.
- Una vez generado el programa ejecutable, no es necesario tener el código fuente.

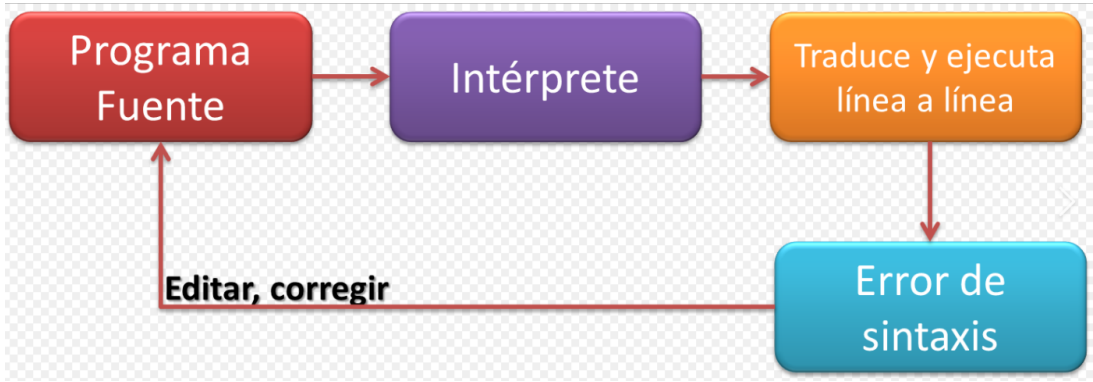
Ejemplos: C, Pascal,…



Intérprete

- La traducción y ejecución de código fuente a código máquina se hace línea por línea.
- Los errores de sintaxis aparecen cuando se interpreta la instrucción con error.
- Necesitamos el código fuente para ejecutar el programa.
- Los lenguajes interpretados suelen ser más lentos en su ejecución

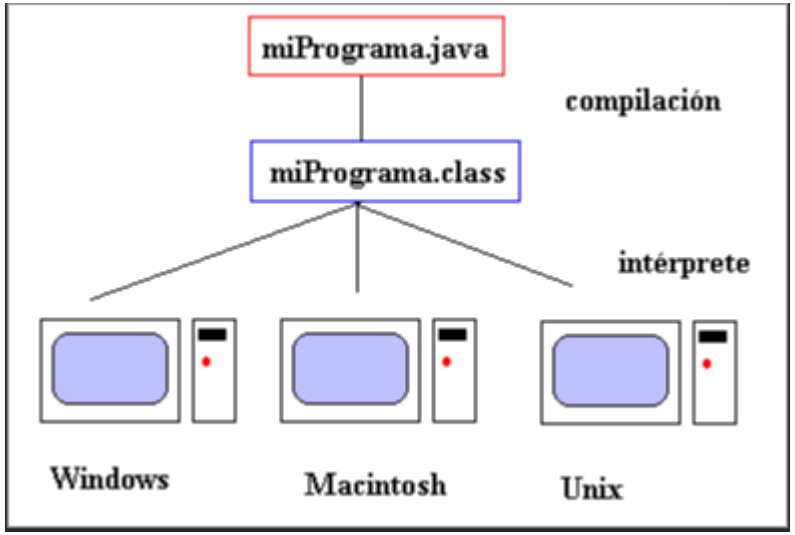
Ejemplos: Python, PHP, …



Máquina virtual

- La traducción se hace en dos pasos.
- Primero se compila el código fuente a un código intermedio (bytecode).
- Segundo, este bytecode se interpreta y ejecuta por una “máquina virtual”.
- El bytecode es multiplataforma.
- Necesito una “máquina virtual” para cada plataforma.
- No necesito el código fuente.

Ejemplo: Java, C#, …



Ejemplo - Compilación y ejecución de un lenguaje compilado: C++

- Lo primero es escribir el código fuente (adivina.cpp)
- Cuando compilamos y ejecutamos el código fuente se genera un programa ejecutable (adivina.bin) que se puede ejecutar desde la línea de comandos
- La compilación incluye varios pasos, el más importante de ellos es la creación de un fichero objeto intermedio. Que ha continuación se enlaza y produce el fichero ejecutable.
- Cuando el código es grande no hay un solo fichero fuente sino muchos, compilar individualmente estos “módulos” permite, por ejemplo, ahorrar mucho tiempo en la modificación y compilación de un solo componente.
- El fichero ejecutable depende de la arquitectura (tipo de procesador, sistema operativo,…) donde se ha compilado.

$ ./adivina.bin

```cpp
#include <iostream>
#include <cstdlib>
using namespace std;

int main(int argc, char *argv[]) {
    int num_secreto,numero,intentos;
    intentos=1;
    num_secreto=rand() % 100;
    cout << "Introduce un número:";
    cin >> numero;
    while (numero!=num_secreto)
    {
        if(numero>num_secreto)
        {
            cout << "El número introducido es mayor\n";
        }
        else
        {
            cout << "El número introducido es menor\n";
        }
        intentos++;
        cout << "Introduce un número:";
        cin >> numero;
    }
    cout << "Lo has acertado en " << intentos << " intentos.";
    return 0;
}
```

Ejemplo - Compilación e interpretación de un programa Java

A la hora de instalar Java tenemos que tener en cuenta que tenemos dos componentes diferentes:

- **Java Runtime Environment (JRE)** son el conjunto de aplicaciones que se instalan en un equipo para que puedan ejecutarse en él aplicaciones java. Los dos componentes principales de un JRE son:
  - Java Virtual Machine: Aplicación que ejecuta el código java en bytecode y que está adaptada a la plataforma sobre la que opera.
  - Bibliotecas Java
- **Java Development Kit (JDK)** son el conjunto de programas para desarrollar aplicaciones y entre otros incluye el compilador javac que convierte un programa fuente java a bytecode. Al instalar el JDK se instala también el componente JRE.
- Utilizando un editor de texto creamos el fichero adivina.java y al compilar la aplicación se crea un fichero bytecode adivina.class que es un fichero portable que se puede ejecutar tanto en Linux como en Windows

  $ javac adivina.java

  java adivina

```java
import java.util.Scanner;
public class Adivina {
     public static void main(String[] args) {
         int num_secreto,numero,intentos;
         Scanner sc = new Scanner(System.in);
        intentos=1;
        num_secreto=(int) (Math.random()*100+1);
        System.out.println("Introduce un número:");
        numero= sc.nextInt();
        while (numero!=num_secreto)
        {
            if(numero>num_secreto)
            {
                System.out.println("El número introducido es mayor\n");
            }
            else
            {
                System.out.println("El número introducido es menor\n");
            }
            intentos++;
            System.out.println("Introduce un número:");
            numero= sc.nextInt();
        }
        System.out.println("Lo has acertado en "+intentos +" intentos.");
     }
}
```

Ejemplo - Ejecución de programas interpretados con Python

Un Lenguaje interpretado es aquel cuyo código no necesita ser preprocesado mediante un compilador, eso significa que el ordenador es capaz de ejecutar la sucesión de instrucciones dadas por el programador sin necesidad de leer y traducir exhaustivamente todo el código.

Por lo tanto necesitamos el código fuente para ejecutar el programa utilizando un interprete que tendremos instalado en nuestras distintas plataformas (Linux, Windows, …)

Python es un lenguaje interpretado, que está instalado por defecto en la mayoría de las distribuciones Linux. Para instalarlo en Windowsse seguirán las instrucciones que hay en su página de descargas.

Creamos el fichero adivina.py y para ejecutar el código fuente, desde cualquier sistema operativo donde tengamos el interprete de python, ejecutaríamos la siguiente instrucción:

$ python3 adivina.py

```
#!/usr/bin/env python
import random

a=random.randrange(0, 100)
intentos=1
b=int(input("Introduce un número:"))
while a!=b:
        if b>a:
                print("El número introducido es mayor")
        else:
                print("El número introducido es menor")
        intentos=intentos+1
        b=int(input("Introduce un número:"))
print("Has acertado en %d intentos." % intentos)
```