

Introducción

José Otero

¹Departamento de informática
Universidad de Oviedo

6 de octubre de 2008

Índice

- 1 Concepto de Programa
- 2 Estructura de un computador
- 3 Representación de la información
- 4 Fases en el desarrollo de un programa

Las instrucciones para realizar un cálculo pueden expresarse empleando el lenguaje natural.

Ejemplo: resolver la ecuación de segundo grado

$$ax^2 + bx + c = 0$$

- Si $d = b^2 - 4ac$ es mayor o igual que cero las soluciones son reales y las da la fórmula $x = \frac{-b \pm \sqrt{d}}{2a}$
- En caso contrario las soluciones son complejas y las da la fórmula $\frac{-b}{2a} \pm \frac{\sqrt{-d}}{2a}i$

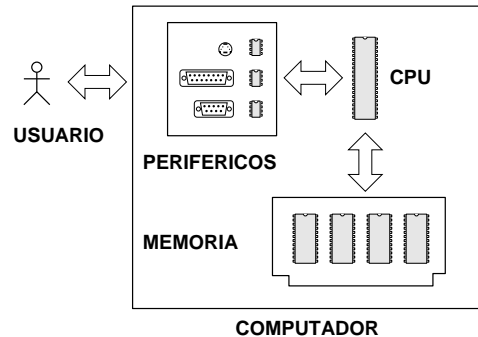
- Para *redactar* la descripción anterior, una persona *necesita*:
 - Conocer el proceso de cálculo.
 - Conocer el lenguaje en el que se expresa.
 - Saber describir el proceso en ese lenguaje.
- Para *aplicarla* una persona *necesita*
 - Conocer el lenguaje.
 - Conocer las operaciones involucradas

Informalmente, una descripción de un cálculo es un algoritmo. *Informalmente*, un programa es un algoritmo escrito en un lenguaje de programación, ejecutándose en un ordenador. Vamos a aprender a escribir programas.

El circuito electrónico capaz de ejecutar una secuencia de órdenes se denomina microprocesador.

Por si mismo es poco útil.

Arquitectura Von Neuman:



El microprocesador es gobernado por un reloj.

- Cada instrucción tarda en ejecutarse un número determinado de ciclos de reloj.
- Entre ciclos de reloj no se puede ejecutar ninguna instrucción (más o menos).

Cuando los parámetros de los procesos en los que se utiliza el ordenador varían rápidamente hay que tener en cuenta:

- Los cálculos tardan en realizarse.
- Cualquier suceso que dure menos que un ciclo de reloj puede pasar inadvertido.

Todos los elementos del computador presentan *dos* estados estables diferenciables.

La memoria también.

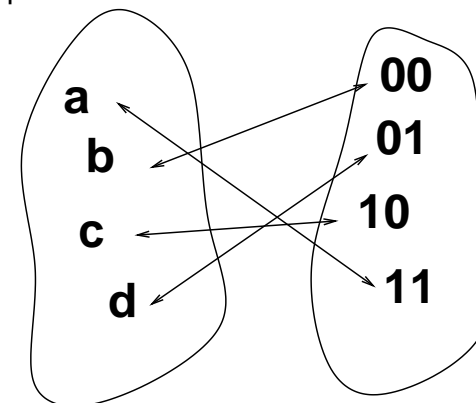
El sistema binario permite representar números naturales mediante 1 y 0.

Se puede representar el 1 con un estado (alto/cerrado) y el cero con otro (bajo/abierto).

Cualquier información tiene que representarse utilizando este sistema.

Una codificación puede definirse como una biyección entre el conjunto de informaciones a representar y un conjunto de cadenas de 1 y 0.

La biyección podría ser arbitraria



Una cadena tamaño 1 es un bit (*Binary Digit*).
Es la cantidad mínima de información.

- Ocho bits se denominan *Byte*.
- 2^{10} es un Kilo (K).
- 2^{20} es un Mega (M).
- 2^{30} es un Giga (G).
- 2^{40} es un Tera (T).

Cuanto mayor es el cardinal del conjunto de la izquierda, mayor es la longitud de las cadenas.

Con cadenas de longitud n se representan 2^n informaciones diferentes.

Como la memoria es finita, cuando el cardinal del conjunto de la izquierda es infinito:

- Hay que acotar el rango de valores.
 - Con 16 bits se pueden representar los enteros del 0 al 65535 o de -32768 a 32767.
- Hay que discretizar el conjunto.
 - No todos los números reales tienen representación en un ordenador.

Para algunos tipos de información existe un algoritmo para deducir la codificación.

- Números Naturales: binario natural.
- Números Enteros: complemento a 2.
- Números Reales: representación en coma flotante.

Para otros la codificación es arbitraria.

- Caracteres: ASCII.

En un sistema de numeración posicional los dígitos de un número tienen un peso igual a la base del sistema elevado a la posición del dígito dentro del número.

Ejemplos:

$$1436_{10} = 1 \times 10^3 + 4 \times 10^2 + 3 \times 10^1 + 6 \times 10^0$$

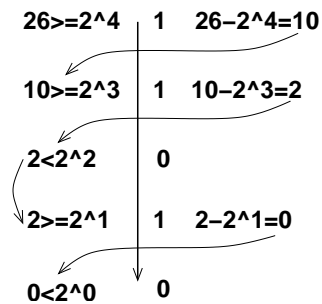
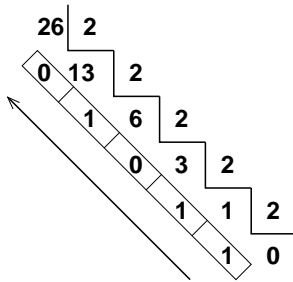
$$11011_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 =$$

$$(16 + 8 + 2 + 1)_{10} = 27_{10}$$

En general:

$$(d_n d_{n-1} d_{n-2} \dots d_2 d_1 d_0)_b = \sum_{i=0}^{i=n} d_i b^i$$

Existen dos formas de convertir números naturales a binario.



No solo se codifican en binario los datos y resultados.

- Los programas también son información y deben almacenarse en la memoria.
- Por lo tanto también deben codificarse en binario.
- Es difícil para un humano escribirlos directamente en binario.
- Es más fácil escribirlos utilizando un *lenguaje de programación*.

Especificación de requisitos

Texto en el que se detalle de forma clara y no ambigua la tarea que se desea que realice el programa, indicando:

- La finalidad del proceso
- Cual es el conjunto al que pertenecen los datos que procesará el algoritmo.
- Cual es el conjunto al que pertenecen los de los datos que se obtendrán como resultado.

Diseño

Consiste en construir un algoritmo que solucione el problema que se ha especificado.

- Seleccionar varias acciones elementales.
 - No necesariamente definidas en un lenguaje de programación.
 - Equivalentes a composiciones de estas.
- Organizarlas en el tiempo para conseguir el resultado.

Análisis

Estudiar lo eficaz que éste es resolviendo el problema.

- Existen muchos algoritmos distintos que resuelven el mismo problema
- No todos ellos tardan el mismo tiempo.

Codificación

Convertir al lenguaje de programación que se desee el algoritmo.

- Se elige una representación para los datos que el programa manejará.
- Se reemplazan las acciones del algoritmo por acciones primitivas válidas en el lenguaje de programación.
- El programa se guarda en uno o varios ficheros de texto. Es indiferente el editor que se utilice.

Verificación

Consta de tres aspectos:

- Demostración de la corrección del algoritmo o verificación formal.
 - Puede hacerse antes de la codificación.
- Prueba del programa.
 - Elegir un conjunto representativo de datos.
 - Comprobar el resultado.
 - Sólo es una prueba si se usan todas las combinaciones de datos posibles.
 - Si algún resultado es incorrecto el programa es incorrecto.
 - En caso contrario no sabemos nada.
- Depuración
 - Detectar y corregir los errores cometidos en la fase de codificación.
 - Existen herramientas para ejecutar el programa paso a paso y examinar los resultados intermedios.

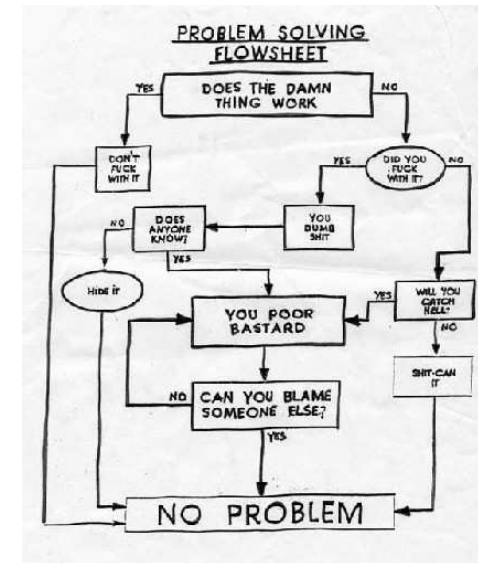
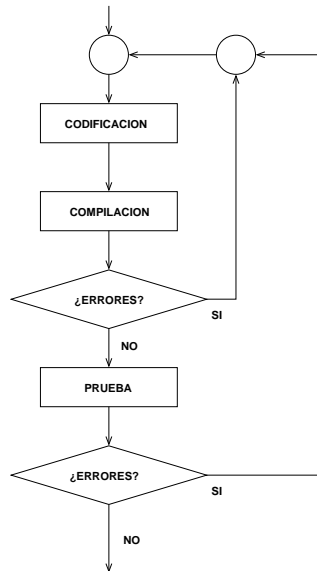
En el caso de algunos lenguajes, después de la codificación, los programas se ejecutan en otro programa denominado *intérprete*.

- Por ejemplo Matlab.

En otros casos, es necesario compilarlos antes:

- El proceso lo realiza otro programa (compilador) a partir del fichero o ficheros *fuentes* del programa.
- Si no existen errores en el *uso* del lenguaje, se produce un *ejecutable*.

Gráficamente



- Si el programa se ejecuta en un intérprete, puede correr en ordenadores de distinta arquitectura (PC con XP, PC con Linux, Mac,...).
 - Siempre que exista intérprete para esa arquitectura.
- Si el programa se ha escrito en un lenguaje que necesita ser compilado:
 - Los ejecutables sólo corren en la arquitectura para la que se han compilado.
 - La máquina en la que corren puede ser virtual (Java) y puede existir máquina virtual para cualquier arquitectura.
 - Los fuentes pueden compilarse para cualquier arquitectura, si se dispone del compilador adecuado. Esta característica se denomina portabilidad.