# Tema 1

## Introducción a la programación

Este material ha sido desarrollado en su totalidad por Román Ginés Martínez Ferrández (rgmf@riseup.net) y Gonzalo Ruiz Arenas (ruizgon@gmail.com) salvo referencias al pie de página.


## Sumario

# 1. Algoritmo

Un algoritmo es un conjunto ordenado y finito de operaciones que permite obtener la solución a un problema.

> **Curiosidad**
> La palabra “algoritmo” procede de la deformación de la forma latina del nombre del matemático persa Abu Ja’far Mohamed ibn Musa al **Khowarizmi** (825 d.C.) que hizo constar en sus escritos referencias a lo indicado.

Para representar algorimos se utiliza un lenguaje próximo al lenguaje natural en el que se aliminan ambigüedades mediante el uso de un determinado vocabulario.

Por ejemplo, veamos cómo representar un algoritmo que nos resuelva el problema de “ir al cine”:

```
1. Inicio
2. Leer cartelera
3. Seleccionar película
4. Si llueve
       4.1. Coger paraguas
5. Ir al cine
6. Si hay entradas
       6.1. Comprar entrada
       6.2. Ver película
7. Si no
       7.1. Seleccionar otra película
       7.2. Ir al punto 5
8. Fin
```

## 1.1. Diagramas de flujo

Con los diagramas de flujo realizamos una representación gráfica de los algoritmos, es una forma de esquematizar y nos permiten ver de un vistazo cómo funciona el algoritmo.

Para realizar esta representación gráfica se emplean una serie de símbolos o iconos estándar, que pueden verse en la Ilustración 1.

En la Ilustración 2 puedes ver cómo podemos diagramar el algoritmo que describíamos arriba que muestra los pasos para “ir al cine”.

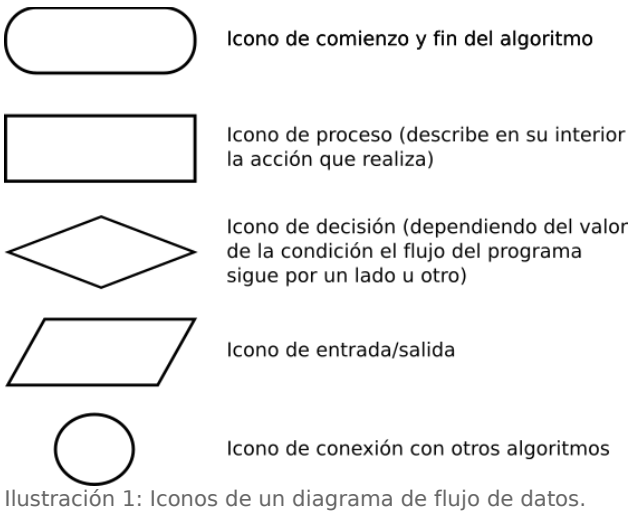


Ilustración 1: Iconos de un diagrama de flujo de datos.

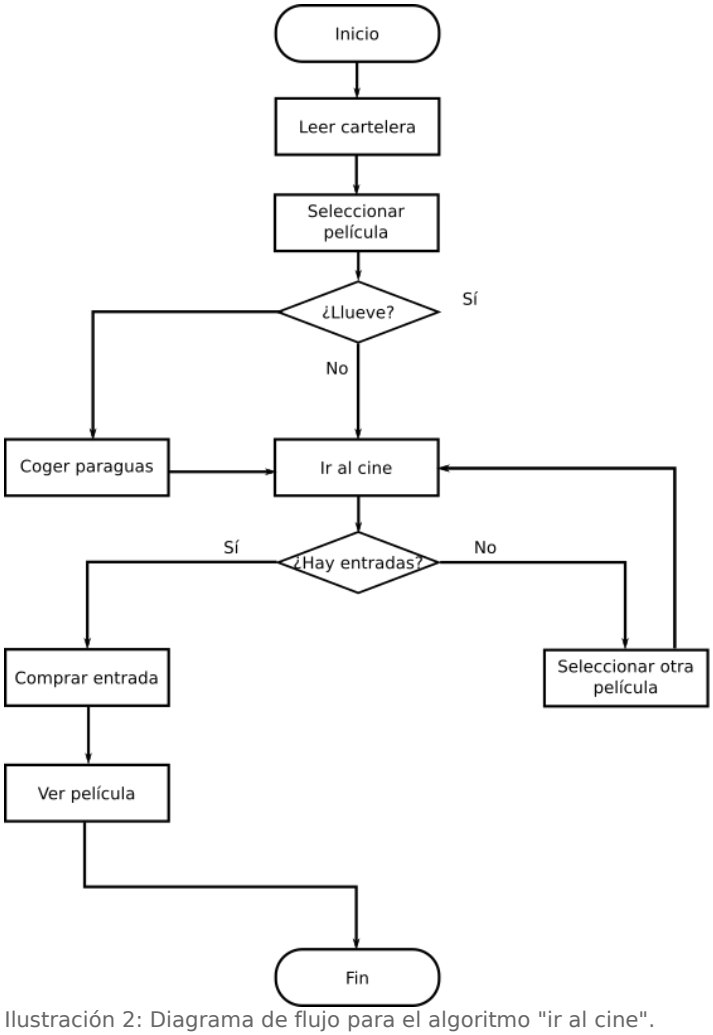


Ilustración 2: Diagrama de flujo para el algoritmo "ir al cine".

# 2. Programa

Un programa informático es un conjunto de instrucciones ejecutables por un ordenador. El programa, por tanto, indica a la máquina:

- lo que debe hacer,
- en qué orden debe hacerlo y
- con qué datos ha de trabajar.

Los programas se escriben utilizando lenguajes de programación.

## 2.1. Lenguajes de programación

Los lenguajes de programación se pueden dividir en 4 generaciones.

### a) 1ª generación de lenguajes: lenguaje máquina

Utilizan una notación binaria (todo son 0s y 1s). Estos lenguajes fueron los primeros en usarse allá por la Segunda Guerra Mundial y la complejidad era bastante grande aunque los programas eran sencillos.

Veamos un ejemplo de lenguaje máquina sencillo, basado en un ordenador imaginario:

| Código | Operación |
|---|---|
| 001 | Cargar |
| 010 | Sumar |
| 011 | Restar |
| 100 | Bifurcar |
| 101 | Comparar |
| 110 | Almacenar |
| 111 | Parar |

A continuación se muestra un programa en lenguaje máquina que resta un número de otro y el resultado lo almacena en una posición de memoria utilizando nuestro ordenador imaginario:

| Memoria Central | | | |
|---|---|---|---|
| Dirección | | | Contenido |
| 0 | 00000 | 00100101 | ------- |
| 1 | 00001 | 01100110 | programa |
| 2 | 00010 | 11000100 | |
| 3 | 00011 | 11100000 | ------- |
| 4 | 00100 | 00000000 | datos |
| 5 | 00101 | 00000111 | |
| 6 | 00110 | 00000010 | ------- |
| 7 | 00111 | 00000000 | memoria libre |
| 8 | 01000 | 00000000 | |

## b) 2ª generación de lenguajes: lenguaje ensamblador

Los lenguajes de programación fueron evolucionando a la vez que las aplicaciones que se requerían eran más complejas. Así fue como se desarrolló el lenguaje ensamblador que no es más que una traducción del lenguaje máquina a un lenguaje algo más textual.

> **Curiosidad**
> Hoy en día el lenguaje ensamblador se sigue usando en programas de sistemas como es un Sistema Operativo.

Veamos un ejemplo de lenguaje ensamblador sencillo, basado en nuestro ordenador imaginario. Este sería el lenguaje ensamblador equivalente al lenguaje máquina visto anteriormente:

| Código binario | Código ensamblador | Operación |
|---|---|---|
| 001 | CAR | Cargar |
| 010 | ALM | Sumar |
| 011 | SUM | Restar |
| 100 | RES | Bifurcar |
| 101 | BC | Comparar |
| 110 | BNC | Almacenar |
| 111 | PAR | Parar |

Ahora se muestra el programa que escribimos anteriormente en binario escrito en ensamblador. Se resta el número 2 al número 7, es decir 7 - 2:

```
                    CAR       A
                    RES       B
                    ALM       C
                    PAR

C:                  0
A:                  7
B:                  2
```

Ahora se utilizan nemotécnicos como CAR, RES, ALM y PAR, en vez de los códigos binarios, y variables como A, B y C, en vez de posiciones de memoria. Estos mnemónicos y variables hacen mucho más fácil la programación pues se abstraen ciertas complejidades hardware.

## c) 3ª generación de lenguajes: lenguajes de alto nivel

Con el paso de los años se había puesto de manifiesto que la mayor limitación de los ordenadoress se encontraba a nivel software y no a nivel hardware, ya que escribir programas en lenguajes de bajo nivel, como el lenguaje máquina y el lenguaje ensamblador, era una tarea larga, difícil y cara.

Los lenguajes de esta generación destacan por tener una sitaxis próxima al lenguaje natural, o más bien al lenguaje matemático, lo que reduce la complejidad de la programación, agiliza mucho el proceso de creación de programas y se acortan exponencialmente los tiempos en el desarrollo de los mismos.

Dentro de esta generación de lenguajes podemos destacar el lenguaje C.

> **Curiosidad**
> Aunque el lenguaje C fue inicialmente desarrollado por Dennis M. Ritchie en 1969, no fue hasta 1978 cuando fue publicada la primera edición del libro *El lenguaje de programación C*, también conocido como *La Biblia de C*, por Dennis M. Ritchie y Brian Kernighan.

El lenguaje de programación C fue creado para escribir sistemas operativos, y en concreto el archiconocido sistema operativo UNIX.

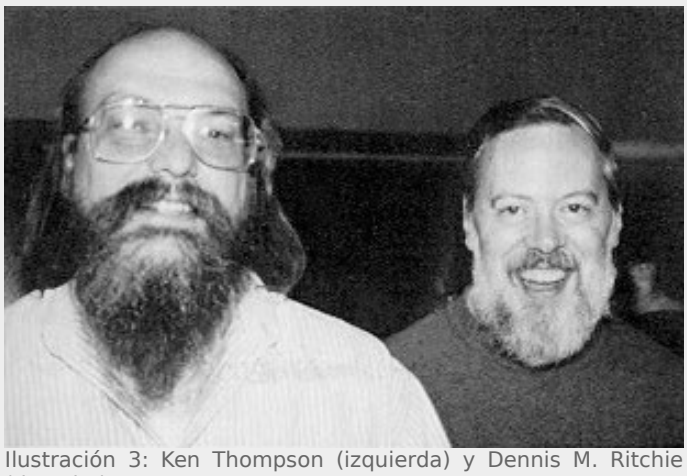

Ilustración 3: Ken Thompson (izquierda) y Dennis M. Ritchie (derecha).

Ken Thompson fue quien diseñó e implementó el sistema operativo UNIX.

Dennis M. Ritchie participó en UNIX y creó el lenguaje de programación C.

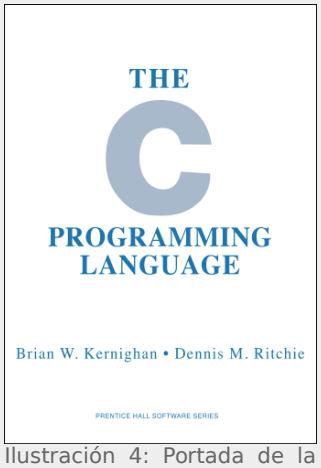


Ilustración 4: Portada de la primera edición del libro El lenguaje de programación C.

Descargada de la Wikipedia, autor Julesmazur con licencia CC by-sa 3.0 Unported.

Hoy en día encontramos programas hechos en C por todas partes: Linux, Windows, MacOs, etc.

A continuación se muestra el programa que resta los números 7 y 2 escrito en el lenguaje de programación C:

```
#include <stdio.h>

int
main (int argc, char **argv)
{
      int num1 = 7;
      int num2 = 2;
      int resultado = num1 + num2;
      printf (“Resultado de 7 – 2 = %d\n”, resultado);
      return 0;
}
```

Aunque no entiendas nada, este fragmento de código no te resultará tan extraño como los dos programas anteriores gracias al mayor nivel de abstracción que incluye.

## d) 4ª generación de lenguajes: lenguajes orientados a objetos

Aunque los lenguajes de programación orientados a objectos no son exactamente lenguajes de programación de 4ª generación se suelen incluir aquí.

La 4ª generación de lenguajes incluye aquellos lenguajes que nos permiten crear programas sin necesidad de escribir casi nada de código fuente. Algo parecido a lo que tenemos con Scratch, donde a golpe de ratón construimos programas sin mucho esfuerzo y conocimiento.

Su uso es limitado a algunas áreas como puede ser la gestión y tratamiento de bases de datos.

## 3. Intérpretes y compiladores

Un programa escrito con un lenguaje de programación no puede ser directamente ejecutado por un ordenador porque, como ya sabrás, un ordenador solo entiende de 1s y de 0s. Se hace necesario traducir el programa a lenguaje máquina convirtiéndolo en un programa ejecutable.

Esta traducción se lleva a cabo por medio de un **compilador** o un **intérprete**.

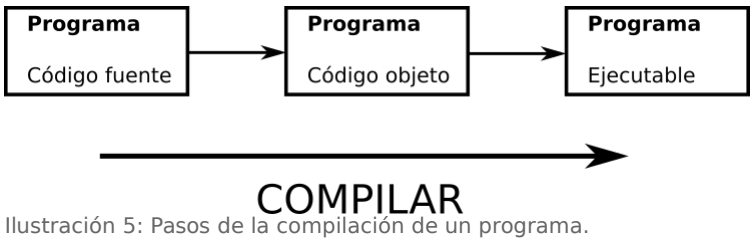


Ilustración 5: Pasos de la compilación de un programa.

- **Código fuente**: es el texto que escribe el programador utilizando un lenguaje de programación determinado.
- **Código objecto**: es el código traducido a lenguaje máquina al que le faltan algunas librerías y adaptaciones para poder ser ejecutado en un ordenador.
- **Ejecutable**: es el código objeto preparado para ser ejecutado en un ordenador.

Aunque un compilador y un intérprete no es lo mismo, se utilizan para convertir un programa escrito en un lenguaje de programación a uno ejecutable. En clase estudiaremos las diferencias.

En la siguiente ilustración puedes ver un programa en lenguaje máquina para el microprocesador Intel 8088 (un microprocesador de los años 80):

```
-u 100 1a
0CFD:0100 BA0B01      MOV     DX,010B
0CFD:0103 B409        MOV     AH,09
0CFD:0105 CD21        INT     21
0CFD:0107 B400        MOV     AH,00
0CFD:0109 CD21        INT     21
-d 10b 13f
0CFD:0100                                       48 6F 6C 61 2C          Hola,
0CFD:0110  20 65 73 74 65 20 65 73-20 75 6E 20 70 72 6F 67   este es un prog
0CFD:0120  72 61 6D 61 20 68 65 63-68 6F 20 65 6E 20 61 73   rama hecho en as
0CFD:0130  73 65 6D 62 6C 65 72 20-70 61 72 61 20 6C 61 20   sembler para la
0CFD:0140  57 69 6B 69 70 65 64 69-61 24                     Wikipedia$
```

Ilustración 6: Lenguaje de máquina del Intel 8088. El código de máquina en hexadecimal se resalta en rojo, el equivalente en lenguaje ensamblador en magenta, y las direcciones de memoria donde se encuentra el código, en azul. Abajo se ve un texto en hexadecimal y ASCII.