PROGRAMACIÓN 1

Grado en Ingeniería Informática

Tema 1

Introducción

Dept. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Universitat d'Alacant
Universidad de Alicante

Índice

1. Representación de la información

- Los computadores representan la información usando dos dígitos: CODIFICACIÓN BINARIA (base 2)
- BIT: (*BInary digIT*: 0 ó 1) unidad de información mínima representable en un ordenador.
- BYTE: 8 bits.
- PALABRA (*WORD*): Es el número de bits que maneja una máquina de forma conjunta (como bloque). Es decir, es el tamaño (en bits) de los registros que hay en el procesador y que tienen capacidad de almacenar una palabra. Los tamaños más empleados son 32 y 64 bits.

1. Representación de la información

Representación de números

- Los números se representan mediante **sistemas numéricos**.
- En el lenguaje de programación C se pueden utilizar los siguientes sistemas numéricos:

Sistema numérico	Prefijo	Representación del número decimal 345
Decimal (Base 10)	*(ninguno)*	345
Binario (Base 2)	**0b**	**0b**101011001
Octal (Base 8)	**0**	**0**531
Hexadecimal (Base 16)	**0x**	**0x**159

1. Representación de la información

Representación de caracteres

- Un carácter se representa empleando un `byte`. El conjunto de caracteres codificable en un ordenador se denomina ***juego de caracteres***, y está compuesto por:
  - letras o caracteres alfabéticos
  - dígitos o caracteres numéricos
  - caracteres especiales y de puntuación
  - caracteres de control (salto de línea, etc.)

1. Representación de la información

Juego de caracteres

- **ASCII**. Permite definir 127 caracteres

Caracteres ASCII de control		
00	NULL	(carácter nulo)
01	SOH	(inicio encabezado)
02	STX	(inicio texto)
03	ETX	(fin de texto)
04	EOT	(fin transmisión)
05	ENQ	(consulta)

Caracteres ASCII imprimibles					
32	espacio	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e

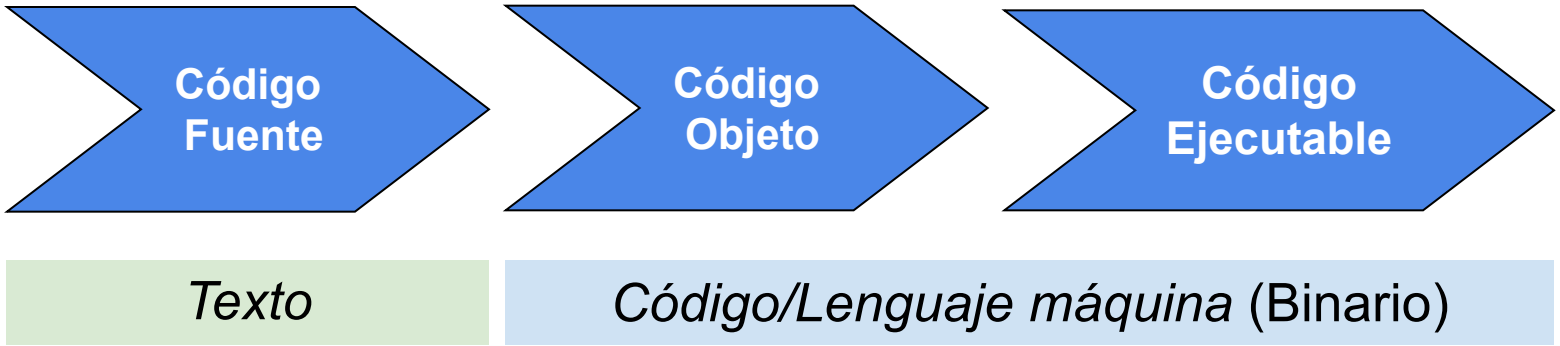
- **ASCII extendido**. Añade un bit más y permite representar vocales acentuadas, *ñ*, etc.

- **Unicode**. Evolución del ASCII. Abarca todos los caracteres de todas las ortografías del mundo.

2. Compiladores vs Intérpretes

COMPILADOR

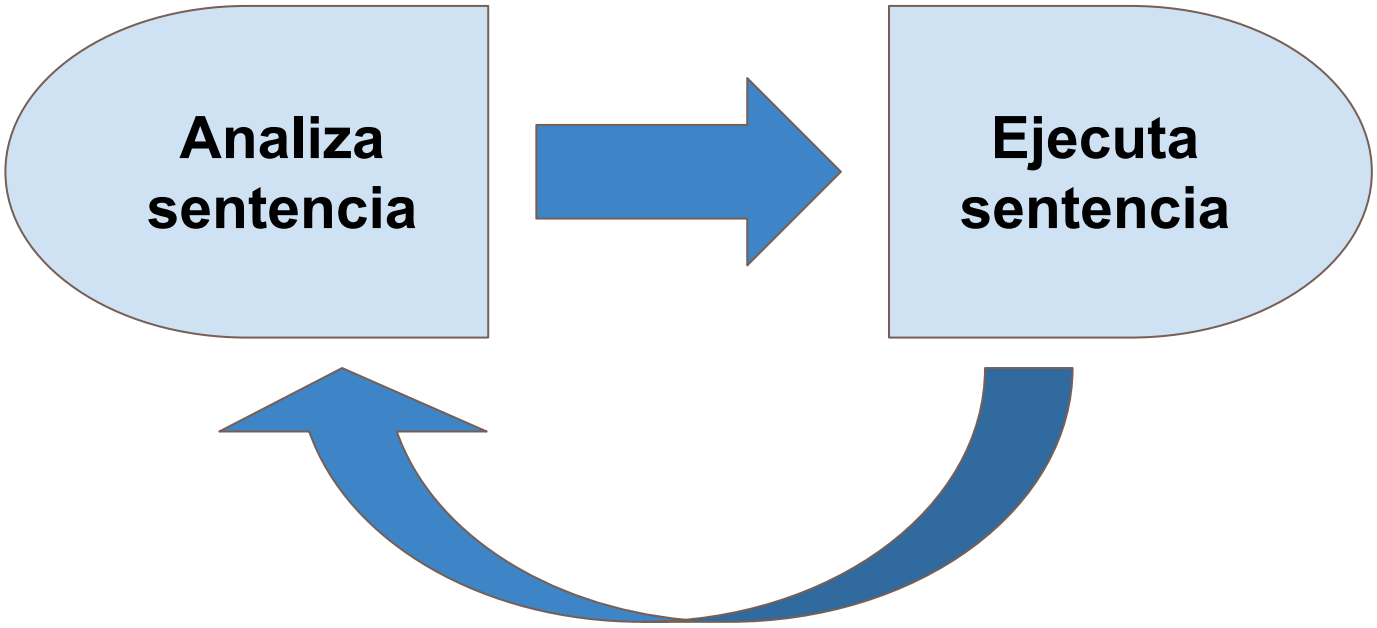
El compilador **analiza** nuestro programa comprobando su sintaxis e indicando los errores de escritura, y **genera** el programa en *código* o *lenguaje máquina*. Puede que el programa necesite un **enlazado** (linkado), en donde se le unen una serie de módulos de librería.



2. Compiladores vs Intérpretes

INTÉRPRETE

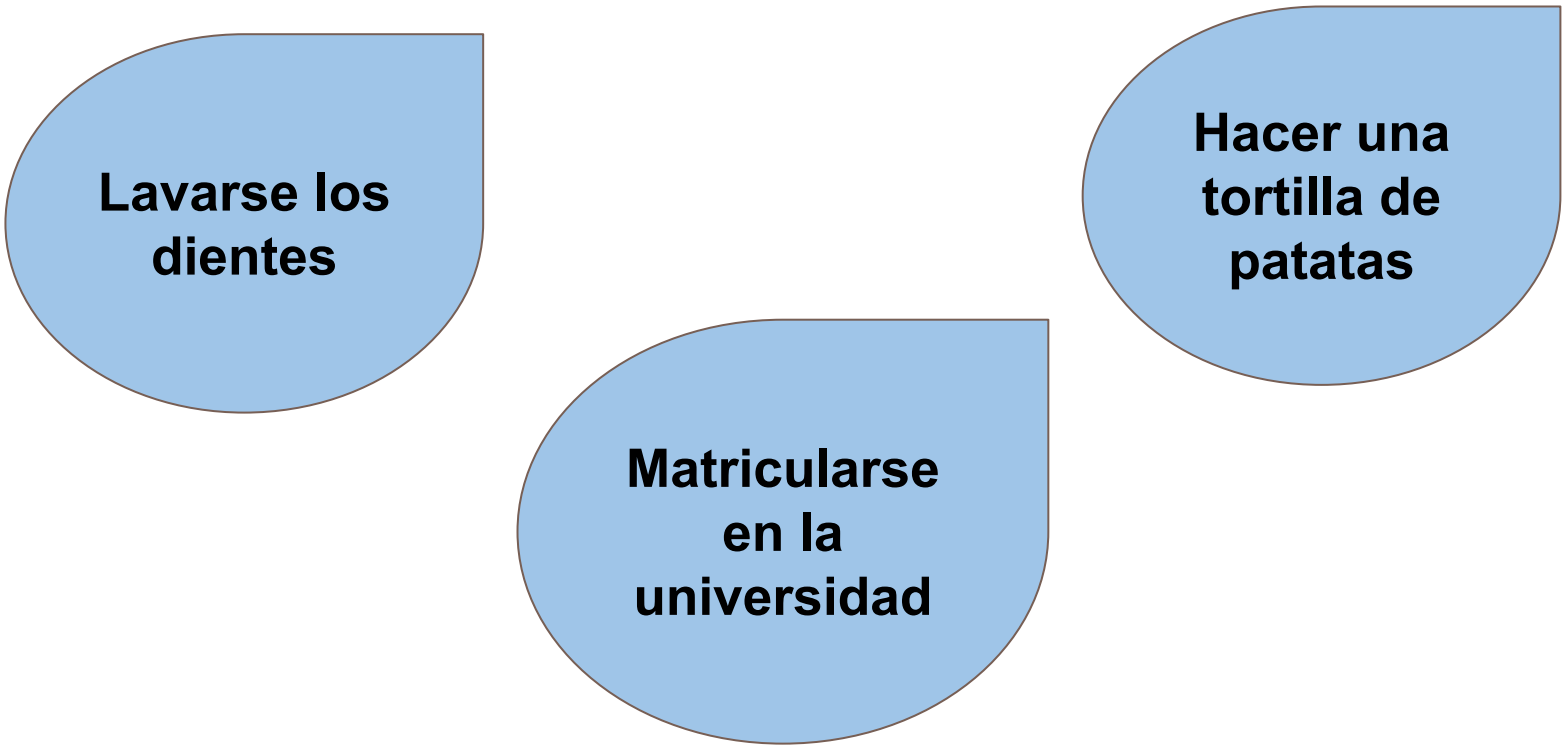
El intérprete analiza y ejecuta el código fuente de un programa sentencia a sentencia.



3. Algoritmo vs Programa informático

ALGORITMO

Secuencia de pasos finitos, bien definidos, que resuelven un problema.



3. Algoritmo vs Programa informático

PROGRAMA INFORMÁTICO

Conjunto de instrucciones ordenadas, escritas en un lenguaje de programación, para que un ordenador lleve a cabo una determinada tarea.

```c
#include<stdio.h>

int main(){

  int numero;

  printf("Dime un número entero: ");
  scanf("%d", &numero);

  if(numero % 2 == 0)
    printf("El número %d es PAR\n", numero);
  else
    printf("El número %d es IMPAR\n", numero);

  return 0;
}
```

4. ¿Cómo desarrollar un programa?

1. Comprender el problema
- ¿Qué hay que resolver?

2. Diseñar una solución
- ¿Cómo se va a resolver?

3. Implementar un programa
- Codificar en un lenguaje de programación

4. Verificar y depurar el programa
- Ejecutar el programa y corregir errores

4. ¿Cómo desarrollar un programa?

Ejemplo:

Problema: Cálculo del precio de la entrada del cine

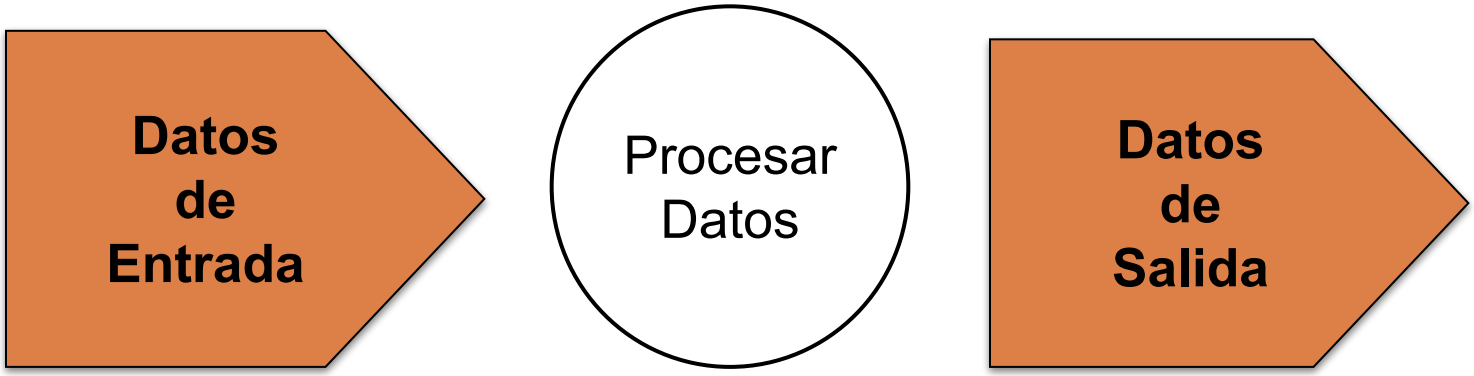
El precio de la entrada se calcula en base a la edad de la persona y a si compra palomitas y/o refresco. El precio de la entrada es de **7€**, pero si la persona tiene menos de 18 años o más de 65, su precio es de **4€**. Si pide sólo palomitas se suman **2€**. Si pide sólo refresco se suman **3€**. Y si pide palomitas y refresco se aplica un descuento y sólo se suman **4 €**.

4. ¿Cómo desarrollar un programa?

1. Comprender el problema

Analizar el problema y responder a la pregunta:

¿**QUÉ** es lo que hay que resolver?

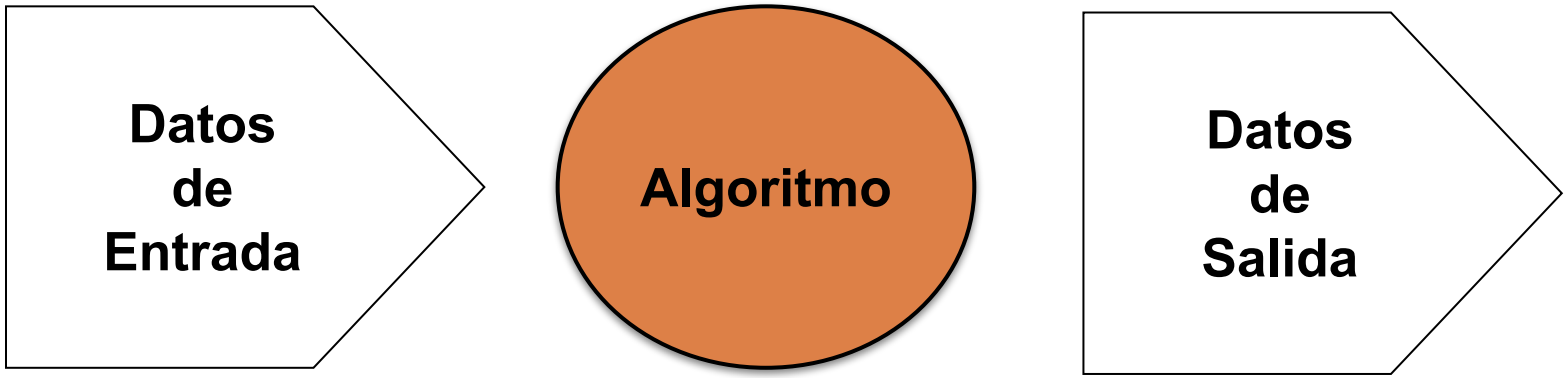


4. ¿Cómo desarrollar un programa?

2. Diseñar una solución

Proponer los pasos a seguir (***algoritmo***) para solucionar el problema y responder a la pregunta:

¿**CÓMO** se va a resolver?



4. ¿Cómo desarrollar un programa?

2. Diseñar una solución

Importancia del Análisis y diseño

- Es fundamental **comprender bien el problema** antes de pensar en la solución
- Antes de ponerse a escribir el programa (***implementación***) es necesario **tener claro cómo resolverlo**.

4. ¿Cómo desarrollar un programa?

2. Diseñar una solución

Solución algorítmica

Algoritmo:

```
- Pedir la edad de la persona
- Si (es menor de 18 ó mayor de 65)
  Entonces
    PRECIO_ENTRADA = 4€
  Sino
    PRECIO_ENTRADA = 7€
- Preguntar si quiere palomitas
- Si (quiere palomitas)
  Entonces
    PALOMITAS = true
  Sino
    PALOMITAS = false
- Preguntar si quiere refresco
- Si (quiere refresco)
  Entonces
    REFRESCO = true
  Sino
    REFRESCO = false
```

```
- Si (PALOMITAS Y REFRESCO)
  Entonces
    PRECIO_ENTRADA = PRECIO_ENTRADA + 4€
  Sino
    Si (PALOMITAS)
    Entonces
      PRECIO_ENTRADA = PRECIO_ENTRADA + 2€
    Sino
      Si (REFRESCO)
      Entonces
        PRECIO_ENTRADA = PRECIO_ENTRADA + 3€
```

4. ¿Cómo desarrollar un programa?

3. Implementar un programa

Codificar en un lenguaje de programación los pasos a seguir para resolver el problema:

a. Conocer la sintaxis del lenguaje de programación a utilizar
b. Escribir el programa con un editor de texto
c. Compilar y corregir errores sintácticos

4. ¿Cómo desarrollar un programa?


```c
#include <stdio.h>
#include <stdbool.h>

int main() {
  int edad, precio;
  char respuesta;
  bool palomitas, refresco;

  printf("Dime tu edad: ");
  scanf("%d", &edad);

  if(edad < 0)
    printf("ERROR\n");
  else {
    if(edad < 18 || edad > 65)
      precio = 4;
    else
      precio = 7;

    printf("¿Quieres palomitas? (S/N)");
    scanf(" %c", &respuesta);
    if(respuesta != 'S' && respuesta != 'N')
      printf("ERROR\n");
    else {
      if(respuesta == 'S')
        palomitas = true;
      else
        palomitas = false;
      printf("¿Quieres refresco? (S/N)");
      scanf(" %c", &respuesta);
      if(respuesta != 'S' && respuesta != 'N')
        printf("ERROR\n");
      else {
        if(respuesta == 'S')
          refresco = true;
        else
          refresco = false;
        if( palomitas && refresco )
          precio = precio + 4;
        else if( palomitas )
          precio = precio + 2;
        else if( refresco )
          precio = precio + 3;

        printf("TOTAL A PAGAR: %d€\n", precio);
      }
    }
  }

  return 0;
}
```

Programa en C para calcular el precio de la entrada

4. ¿Cómo desarrollar un programa?

4. Verificar y depurar el programa

Ejecutar el programa y corregir errores :

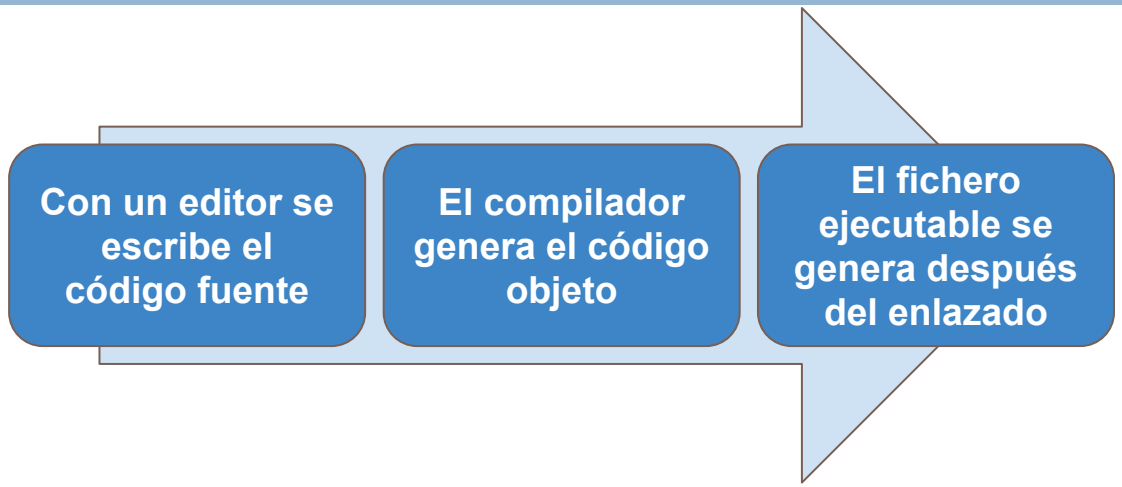
1. Verificar el programa (**Pruebas**)
   - Ejecutar el programa y **detectar errores**
2. Depurar el programa (**Depuración**)
   - **Corregir errores** de ejecución del programa

DATOS DE ENTRADA			DATOS DE SALIDA	RESULTADO OK
Edad	Palomitas	Refresco		
12	N	N	4	✓
36	N	S	10	✓
26	S	S	11	✓
18	X			✗
43	S	F		✗
67	S	N	6	✓
-34				✗
17	S	S	8	✓

5. ¿Por qué utilizamos el lenguaje C?

- Es un lenguaje de propósito general
- Muy utilizado en el mundo laboral
- Independiente del sistema operativo
- Facilita la programación estructurada y modular
- Opera a bajo y alto nivel
- Su aprendizaje permite conocer mejor otros lenguajes de programación

6. Cómo se crea un programa ejecutable



- Con un editor de texto (kate, gedit, sublime text, etc.) **se escribe** el programa, dando lugar al **código fuente.**
- **Se compila,** usando el correspondiente compilador, **para generar el ejecutable**. Usaremos el **gcc** en el sistema operativo Linux.
- Otra posibilidad es usar un IDE (Entorno Integrado de Desarrollo). Ejemplo: Dev-C++ (Windows), Eclipse, NetBeans. Los IDE incluyen el editor, el compilador, el enlazador y un depurador, además de otros elementos.

7. Estructura de un programa en C


```c
#include<stdio.h>

int main(){
    int num;

    printf("Dime un número entero: ");
    scanf("%d", &num);

    if(num % 2 == 0)
        printf("El número %d es PAR\n", num);
    else
        printf("El número %d es IMPAR\n", num);

    return 0;
}
```

Inclusión de ficheros auxiliares

Función **main**. Función principal del programa. Es la primera función que se ejecuta.

Sentencia de lectura (entrada) desde teclado

Sentencia de escritura (salida) a pantalla

Sentencia para finalizar la ejecución de la función y que ésta devuelva el valor 0