

# Programación 1

## Tema 2

---

### Lenguaje de programación y ejecución de un programa



Escuela de  
Ingeniería y Arquitectura  
**Universidad Zaragoza**

## Información sobre protección de datos de carácter personal en el tratamiento de gestión de grabaciones de docencia

Sesión con grabación



**Tratamiento:** Gestión de grabaciones de docencia

**Finalidad:** Grabación y tratamiento audiovisual de docencia y su evaluación

**Base Jurídica:** Art. 6.1.b), c) y d) Reglamento General de Protección de Datos

**Responsable:** Universidad de Zaragoza.

**Ejercicio de Derechos** de acceso, rectificación, supresión, portabilidad, limitación u oposición al tratamiento ante el gerente de la Universidad conforme a <https://protecciondatos.unizar.es/procedimiento-seguir>

**Información completa en:**

[https://protecciondatos.unizar.es/sites/protecciondatos.unizar.es/files/user/s/lopd/gdocencia\\_extensa.pdf](https://protecciondatos.unizar.es/sites/protecciondatos.unizar.es/files/user/s/lopd/gdocencia_extensa.pdf)

**Propiedad intelectual:** Queda prohibida la difusión, distribución o divulgación de la grabación y particularmente su compartición en redes sociales o servicios dedicados a compartir apuntes. La infracción de esta prohibición puede generar responsabilidad disciplinaria, administrativa y de índole civil o penal.

Fuente de las imágenes: <https://pixabay.com/es>



## Información sobre protección de datos de carácter personal en el tratamiento de gestión de grabaciones de docencia

---

- ❑ Se recuerda que la grabación de las clases por medios distintos a los usados por el profesor o por personas diferentes al profesor sin su autorización expresa no está permitida, al igual que la difusión de esas imágenes o audios.



# Índice

---

- Lenguaje de programación
  - Símbolos
  - Sintaxis
  - Semántica
- Computador
- Ejecución de un programa
- Sistema operativo, entorno de programación

# Expresión de un algoritmo

---

- Lenguaje natural
- Notación algorítmica
- Notación gráfica
  - Diagramas de flujo
- Lenguaje de programación
  - Ada, Pascal, Módulo-2, C
  - **C++**, Java
  - Lisp, Prolog
  - Fortran, Cobol

# Elementos de un programa

---

- **Símbolos**
  - Palabras clave y directivas
  - Identificadores
  - Operadores
  - Separadores
  - Constantes
- **Sintaxis**
- **Semántica**

# Ejemplo de programa

```
#include <iostream>

/*
 * Programa que escribe en la pantalla el mensaje
 * «Bienvenidos a la Universidad».
 */
int main() {
    // primera instrucción
    std::cout << "Bienvenidos a la Universidad" << std::endl;

    // segunda instrucción
    return 0;
}
```

# Comentarios

```
#include <iostream>

/*
 * Programa que escribe en la pantalla el mensaje
 * «Bienvenidos a la Universidad»
 */
int main() {
    // primera instrucción
    std::cout << "Bienvenidos a la Universidad" << std::endl;

    // segunda instrucción
    return 0;
}
```



# Símbolos

```
#include <iostream>

int main() {
    std::cout
        << "Bienvenidos a la Universidad"
        << std::endl;
    return 0;
}
```

# Palabras clave y directivas

```
#include <iostream>

int main() {
    std::cout
        << "Bienvenidos a la Universidad"
        << std::endl;
    return 0;
}
```

# Palabras clave en C++ 17

<code>alignas</code>	<code>continue</code>	<code>friend</code>	<code>register</code>	<code>true</code>
<code>alignof</code>	<code>decltype</code>	<code>goto</code>	<code>reinterpret_cast</code>	<code>try</code>
<code>asm</code>	<code>default</code>	<code>if</code>	<code>return</code>	<code>typedef</code>
<code>auto</code>	<code>delete</code>	<code>inline</code>	<code>short</code>	<code>typeid</code>
<code>bool</code>	<code>do</code>	<code>int</code>	<code>signed</code>	<code>typename</code>
<code>break</code>	<code>double</code>	<code>long</code>	<code>sizeof</code>	<code>union</code>
<code>case</code>	<code>dynamic_cast</code>	<code>mutable</code>	<code>static</code>	<code>unsigned</code>
<code>catch</code>	<code>else</code>	<code>namespace</code>	<code>static_assert</code>	<code>using</code>
<code>char</code>	<code>enum</code>	<code>new</code>	<code>static_cast</code>	<code>virtual</code>
<code>char16_t</code>	<code>explicit</code>	<code>noexcept</code>	<code>struct</code>	<code>void</code>
<code>char32_t</code>	<code>export</code>	<code>nullptr</code>	<code>switch</code>	<code>volatile</code>
<code>class</code>	<code>extern</code>	<code>operator</code>	<code>template</code>	<code>wchar_t</code>
<code>const</code>	<code>false</code>	<code>private</code>	<code>this</code>	<code>while</code>
<code>constexpr</code>	<code>float</code>	<code>protected</code>	<code>thread_local</code>	
<code>const_cast</code>	<code>for</code>	<code>public</code>	<code>throw</code>	

# Directivas en C++ 17

<b>#</b>	<b>#if</b>	<b>#elif</b>	<b>#pragma</b>
<b>##</b>	<b>#ifdef</b>	<b>#endif</b>	<b>#undef</b>
<b>#define</b>	<b>#ifndef</b>	<b>#line</b>	
<b>#include</b>	<b>#else</b>	<b>#error</b>	

# Símbolos

```
#include <iostream>

int main() {
    std::cout
        << "Bienvenidos a la Universidad"
        << std::endl;
    return 0;
}
```

# Identificadores

```
#include <iostream>

int main() {
    std::cout
        << "Bienvenidos a la Universidad"
        << std::endl;
    return 0;
}
```

# Símbolos

```
#include <iostream>

int main() {
    std::cout
        << "Bienvenidos a la Universidad"
        << std::endl;
    return 0;
}
```

# Operadores

```
#include <iostream>

int main() {
    std::cout
        << "Bienvenidos a la Universidad"
        << std::endl;
    return 0;
}
```



# Algunos operadores en C++

---

□	<	<=	>	>=	==	!=	
□	+	-	*	/	%	++	--
□	&&		!				
□	=	+=	-=	*=	/=	%=	
□	()	[]	*	&	( <i>tipo</i> )	<b>sizeof</b>	
	::						

# Separadores y finalizadores

```
#include <iostream>
int main() {
    → std::cout
    → → → → << "Bienvenidos a la Universidad"
    → → → → << std::endl;
    → return 0;
}
```

# Separadores y finalizadores en C++

---

- Separadores
  - Blancos (espacios, tabuladores, fin de línea)
  - Coma (,)
- Finalizadores
  - Punto y coma (;)
- Delimitadores
  - Paréntesis: ( )
  - Corchetes: [ ]
  - Llaves: { }

# Constantes

```
#include <iostream>

int main() {
    std::cout
        << "Bienvenidos a la Universidad"
        << std::endl;
    return 0;
}
```

# Elementos de un programa

---

## □ **Símbolos**

- Palabras clave
- Identificadores
- Operadores
- Separadores
- Constantes

## □ **Sintaxis**

## □ **Semántica**

# Notación de Backus-Naur

---

- Notación BNF (*Backus-Naur form*)
  - Definición de reglas sintácticas para definir lenguajes
  - Descripción de la organización de estructuras de datos secuenciales

# Notación de Backus-Naur

- **Metasímbolos** utilizados:
  - Definición de una regla  
**<nombre\_regla> ::= expresión**
  - Sustitución de la expresión  
**<nombre\_regla>**
  - Literal  
**"Prog1f"**
  - Alternativa  
**expresión1 | expresión2**
  - Agrupación sin repetición  
**( expresión )**
  - Agrupación con repetición (cero, una o más veces)  
**{ expresión }**
  - Agrupación con opcionalidad (cero o una veces)  
**[ expresión ]**

# Notación Backus-Naur

$::=$	Definición de regla sintáctica
$\langle \quad \rangle$	Delimitadores de nombre de regla sintáctica
$\text{“} \quad \text{”}$	Carácter o secuencia de caracteres literal (en ocasiones, los omitiremos)
$ $	Separador de alternativas
$( \quad )$	Agrupador sin repetición
$\{ \quad \}$	Agrupador con repetición (0, 1 o más veces)
$[ \quad ]$	Agrupador opcional (0 o 1 vez)



# Identificadores

```
<identificador> ::=  
    ( <letra> | _ ){ <letra> | <dígito> | _ }  
<letra> ::= <mayúscula> | <minúscula>  
<mayúscula> ::= A | B | C | D | E | F | G | H  
    | I | J | K | L | M | N | O | P | Q | R | S  
    | T | U | V | W | X | Y | Z  
<minúscula> ::= a | b | c | d | e | f | g | h  
    | i | j | k | l | m | n | o | p | q | r | s  
    | t | u | v | w | x | y | z  
<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7  
    | 8 | 9
```

## Sintaxis. Ejemplo

```
<instrucciónCondicional> ::=  
“if” “(” <condición> “)”  
  (<instrucción> | <bloque>)  
  [“else” (<instrucción> | <bloque>)]
```

```
<bloque> ::= “{” {<instrucción>} “}”
```

```
<condición> ::= ...
```

```
<instrucción> ::= ...
```

## Semántica. Ejemplo

```
if (x >= 0) {  
    cout << x << endl;  
}  
else {  
    cout << -x << endl;  
}
```



# Índice

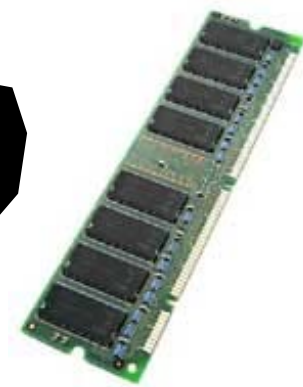
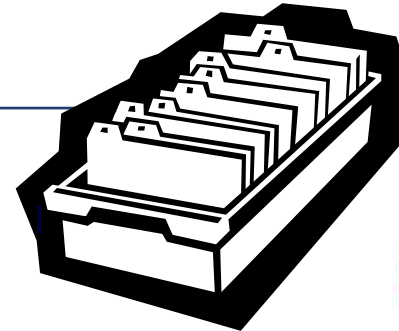
---

- Lenguaje de programación
  - Símbolos
  - Sintaxis
  - Semántica
- Computador
- Ejecución de un programa
- Sistema operativo, entorno de programación

# Computador

## □ Memoria

- Datos e instrucciones

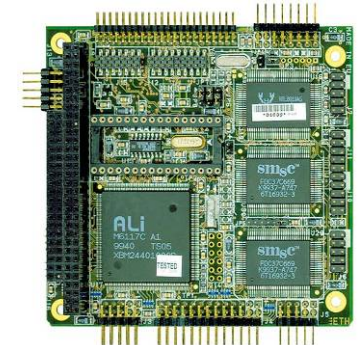
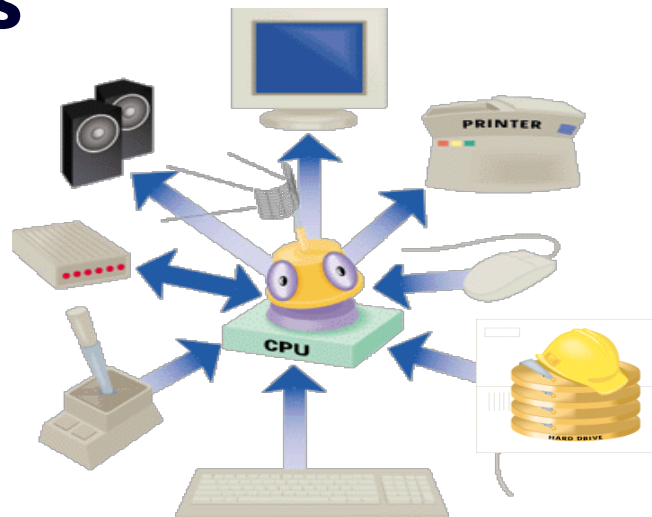


## □ Unidad central de proceso (CPU)

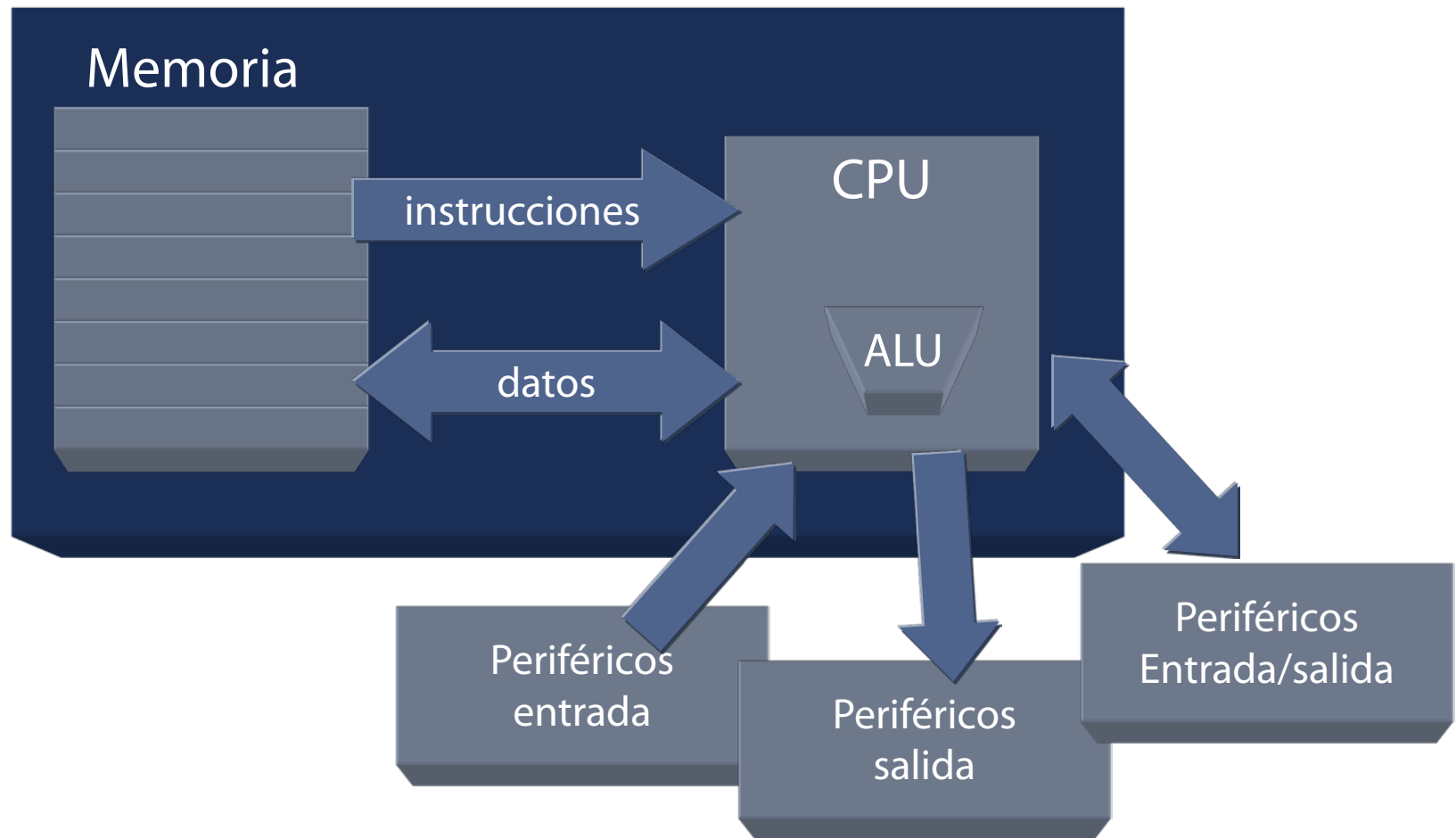
- Ejecuta acciones

## □ Periféricos

- Entrada
- Salida

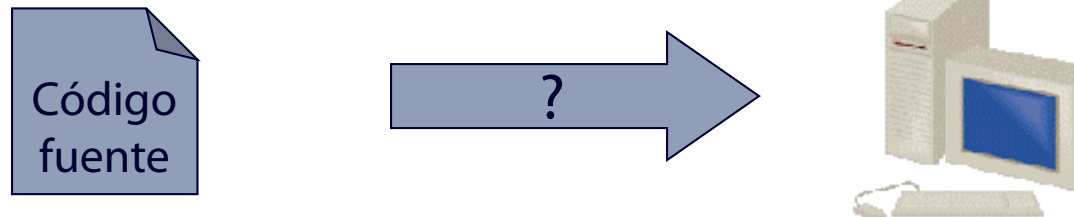


# Computador



# Ejecución de un programa

---



- Ejecución interpretada
  - Un **intérprete** (en memoria del computador) analiza y ejecuta cada instrucción del código fuente
- Ejecución con compilación previa
  - Un **compilador** genera un **programa ejecutable** que se carga en memoria y se ejecuta

# Sistema operativo.

## Entorno de programación

---

- Sistema operativo
  - Conjunto de programas
    - Facilitan la utilización del sistema
    - Controlan el funcionamiento de la máquina
- Entorno de programación
  - Facilita el trabajo de desarrollo de programas utilizando un lenguaje determinado



# Resumen

---

- Lenguaje de programación
  - Símbolos
  - Sintaxis
  - Semántica
- Computador
- Ejecución de un programa
- Sistema operativo, entorno de programación