

Programación 1

Tema 2

Lenguaje de programación y ejecución de un programa



Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza



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, Módula-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>

/*
 * Pre:  ---
 * Post: Escribe por 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>

/*
 * Pre:   ---
 * Post:  Escribe por 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++

<code>alignas</code>	<code>const</code>	<code>for</code>	<code>private</code>	<code>throw</code>
<code>alignof</code>	<code>constexpr</code>	<code>friend</code>	<code>protected</code>	<code>true</code>
<code>and</code>	<code>const_cast</code>	<code>goto</code>	<code>public</code>	<code>try</code>
<code>and_eq</code>	<code>continue</code>	<code>if</code>	<code>register</code>	<code>typedef</code>
<code>asm</code>	<code>decltype</code>	<code>inline</code>	<code>reinterpret_cast</code>	<code>typeid</code>
<code>auto</code>	<code>default</code>	<code>int</code>	<code>return</code>	<code>typename</code>
<code>bitand</code>	<code>delete</code>	<code>long</code>	<code>short</code>	<code>union</code>
<code>bitor</code>	<code>do</code>	<code>mutable</code>	<code>signed</code>	<code>unsigned</code>
<code>bool</code>	<code>double</code>	<code>namespace</code>	<code>sizeof</code>	<code>using</code>
<code>break</code>	<code>dynamic_cast</code>	<code>new</code>	<code>static</code>	<code>virtual</code>
<code>case</code>	<code>else</code>	<code>noexcept</code>	<code>static_assert</code>	<code>void</code>
<code>catch</code>	<code>enum</code>	<code>not</code>	<code>static_cast</code>	<code>volatile</code>
<code>char</code>	<code>explicit</code>	<code>not_eq</code>	<code>struct</code>	<code>wchar_t</code>
<code>char16_t</code>	<code>export</code>	<code>nullptr</code>	<code>switch</code>	<code>while</code>
<code>char32_t</code>	<code>extern</code>	<code>operator</code>	<code>template</code>	<code>xor</code>
<code>class</code>	<code>false</code>	<code>or</code>	<code>this</code>	<code>xor_eq</code>
<code>compl</code>	<code>float</code>	<code>or_eq</code>	<code>thread_local</code>	

Directivas en C++

#	#if	#elif	#pragma
#define	#ifdef	#endif	
#undef	#ifndef	#line	
#include	#else	#error	

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;
}
```

Identificadores en C++

```
<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
```

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;
}
```

Operadores en C++

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

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)

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;  
}
```

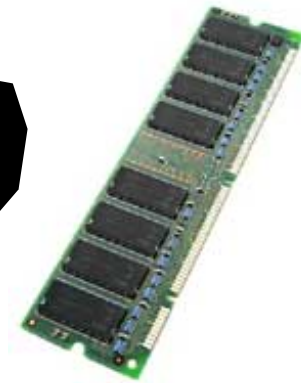
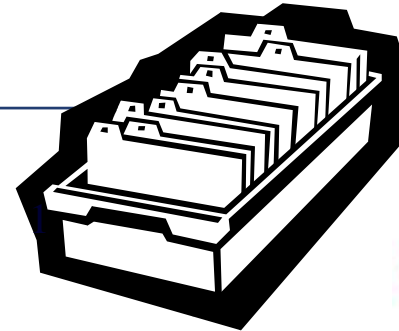
Índice

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

Computador

□ Memoria

- Datos e instrucciones

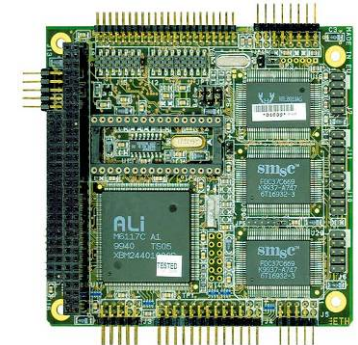
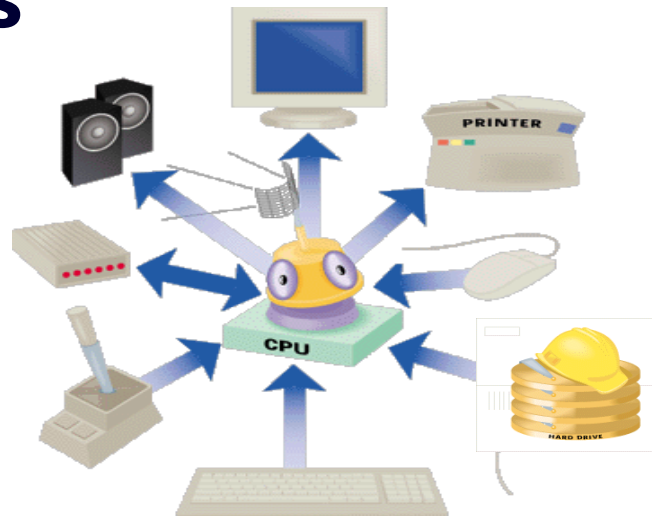


□ Unidad central de proceso (CPU)

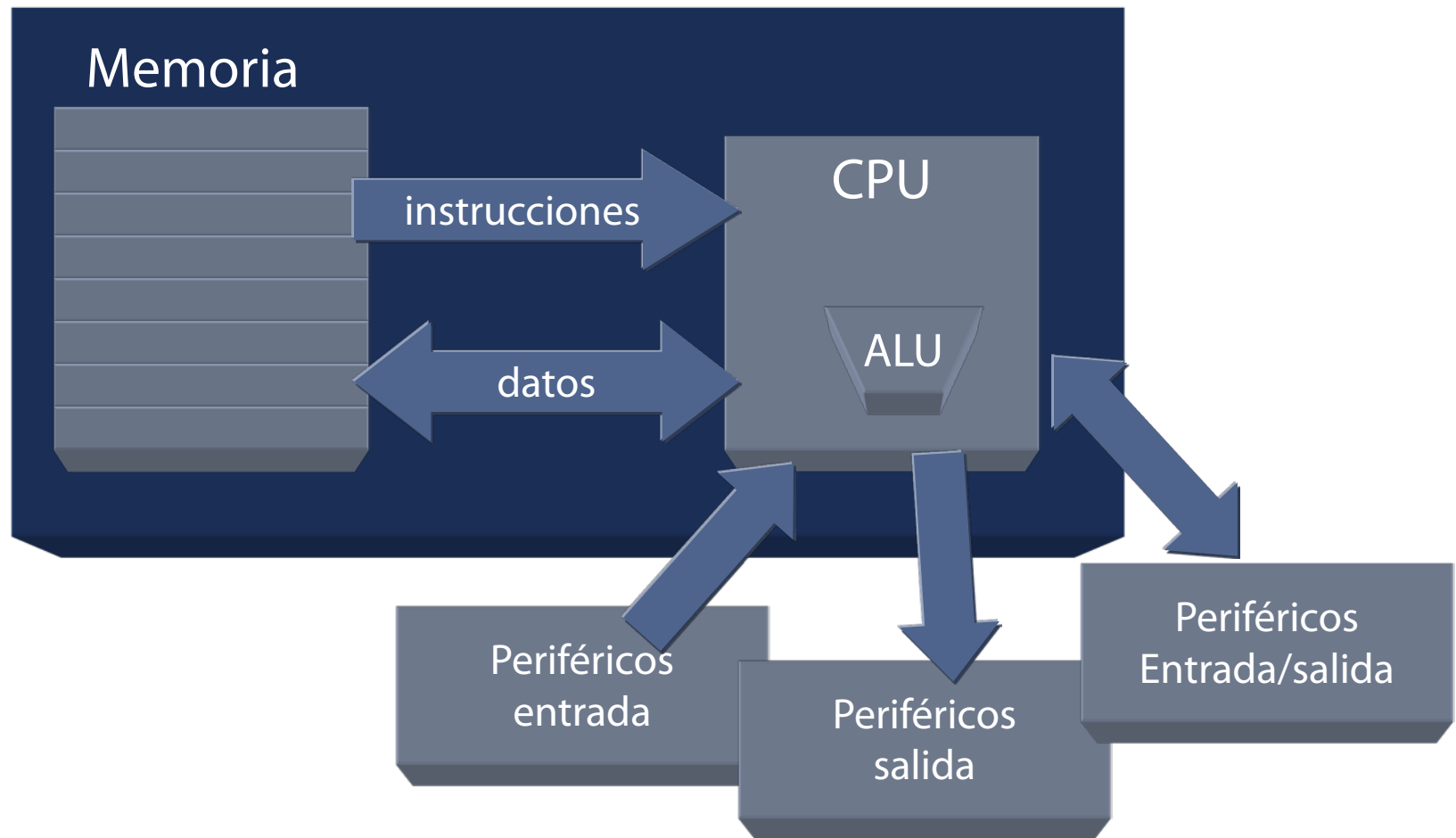
- Ejecuta acciones

□ Periféricos

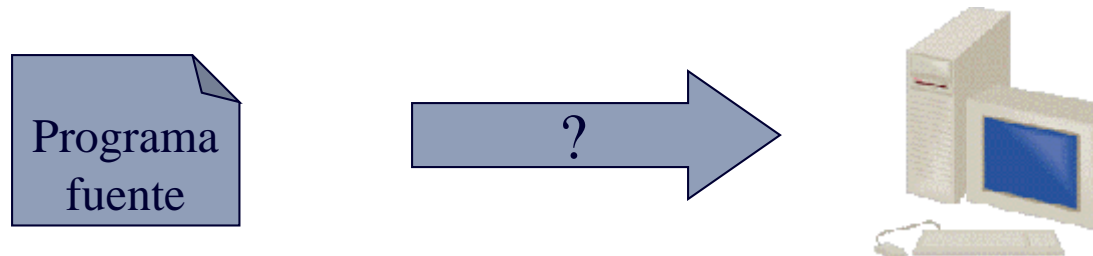
- Entrada
- Salida



Computador



Ejecución de un programa



- Ejecución interpretada
 - Un **intérprete** (en memoria del computador) analiza y ejecuta cada instrucción del programa 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