## 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++

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

# 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++

- `<`, `<=`, `>`, `>=`, `==`, `!=`
- `+`, `-`, `*`, `/`, `%`, `++`, `--`
- `&&`, `||`, `!`
- `=`, `+=`, `-=`, `*=`, `/=`, `%=`
- `()`, `[]`, `*`, `&`, (*tipo*), `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 |
| < > | Delimitadores de nombre de regla sintáctica |
| “ ” | Carácter o secuencia de caracteres literal<br>(en ocasiones, los omitiremos) |
| \| | Separador de alternativas |
| ( ) | Agrupador sin repetición |
| { } | Agrupador con repetición<br>(0, 1 o más veces) |
| [ ] | Agrupador opcional (0 o 1 vez) |

# Sintaxis. Ejemplo

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

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

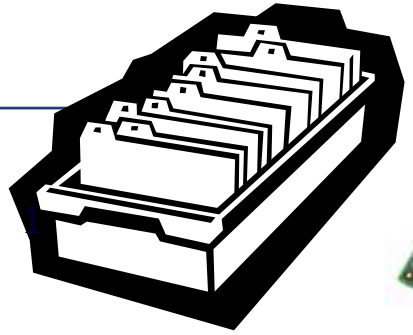
# Semántica. Ejemplo

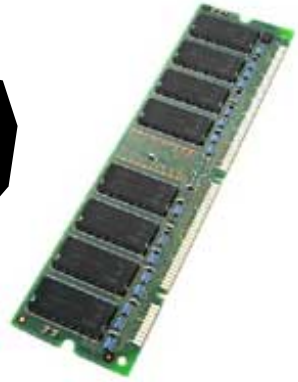
```
if (x >= 0) {
  cout << x << endl;
}
else {
  cout << -x << endl;
}
```

# Índice

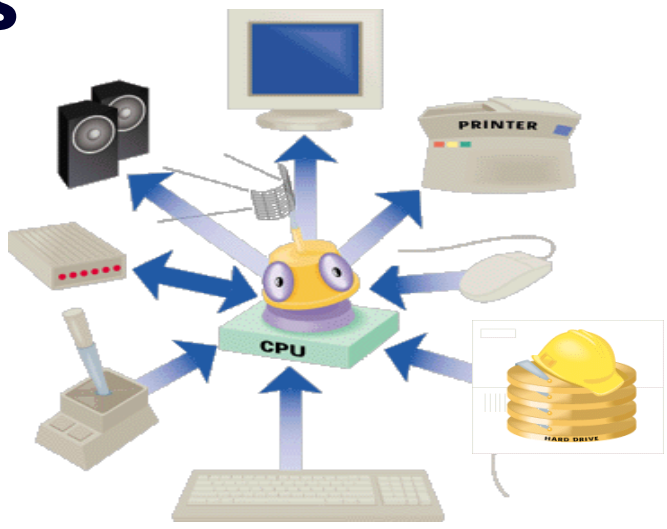
- Lenguaje de programación
  - Símbolos
  - Sintaxis
  - Semántica
- Computador
- Ejecución de un programa
- Sistema operativo, entorno de programación

# Computador

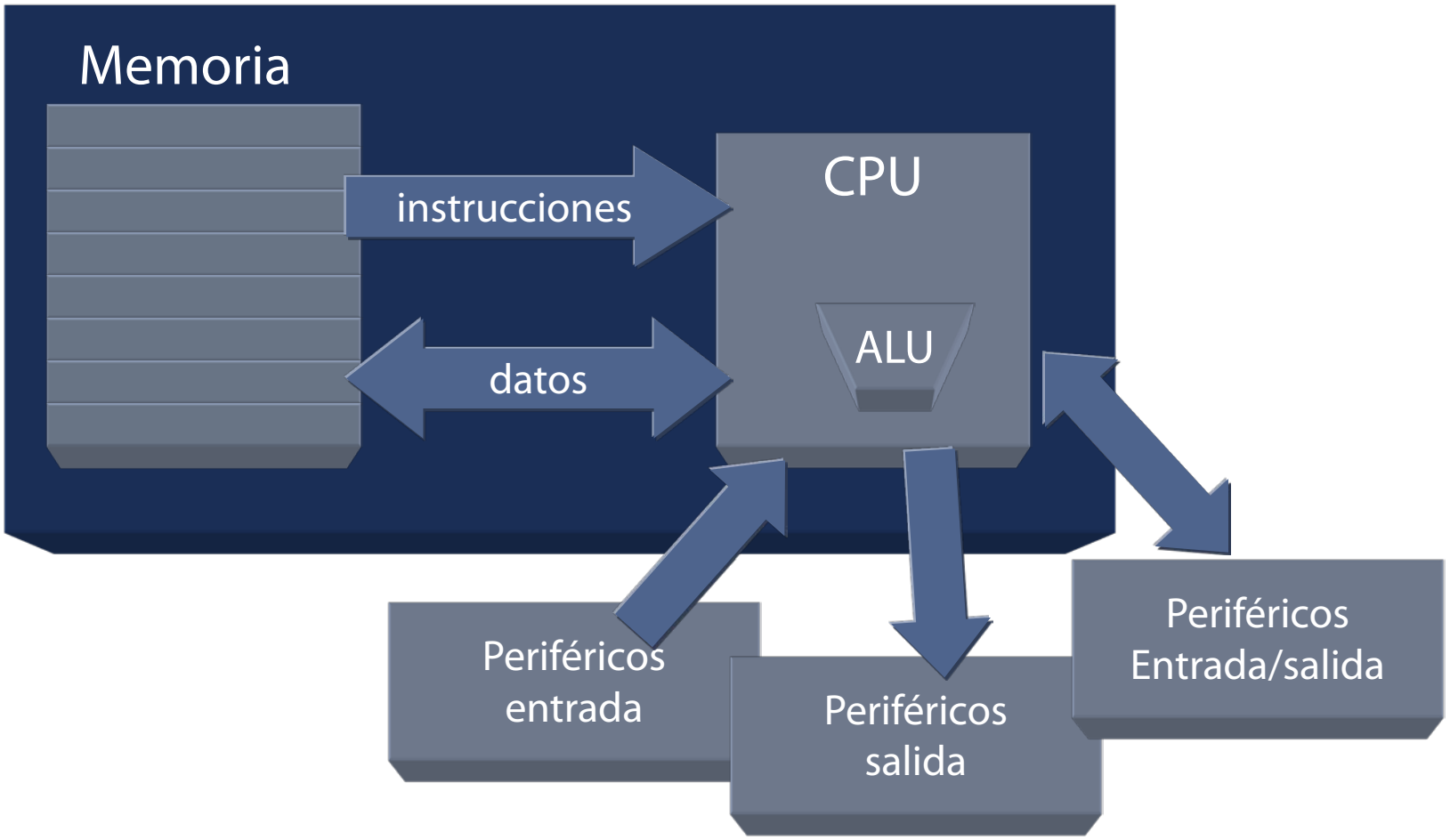




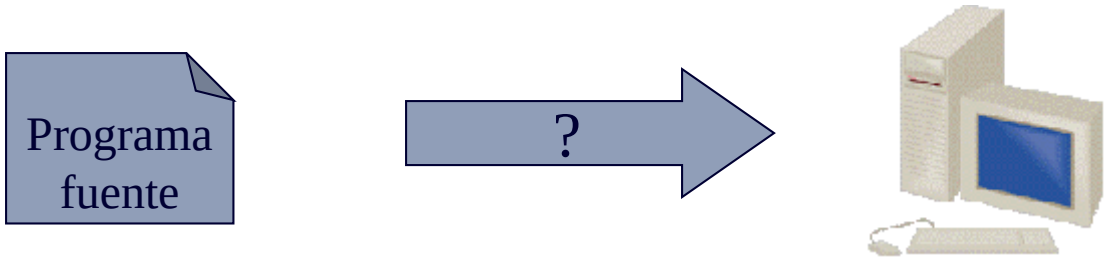
- **Memoria**
  - Datos e instrucciones
- **Unidad central de proceso** (CPU)
  - Ejecuta acciones
- **Periféricos**
  - Entrada
  - Salida



# Computador


Memoria
instrucciones
datos
CPU
ALU
Periféricos
entrada
Periféricos
salida
Periféricos
Entrada/salida

# 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