Fundamentos de la programación

1 Computadoras y programación

Doble Grado en Matemáticas e informática

Ana Gil Luezas
(adaptadas del original de Luis Hernández Yáñez)

Facultad de Informática
Universidad Complutense

Índice

Informática y computadora R.A.E.

Informática (Ciencia de la computación)

Conjunto de conocimientos científicos y técnicos
que hacen posible el tratamiento automático
de la información por medio de ordenadores

Computadora

Máquina electrónica, analógica o digital,
dotada de una memoria de gran capacidad
y de métodos de tratamiento de la información,
capaz de resolver problemas matemáticos y lógicos
mediante la ejecución de programas informáticos

Computadoras

En todas partes y con muchas formas

Hardware y software

Hardware

Componentes que integran
la parte material
de una computadora

Software

Programas, instrucciones
y reglas informáticas
para ejecutar tareas
en una computadora

Programar

Indicar a la computadora qué es lo que tiene que hacer para realizar una tarea concreta.

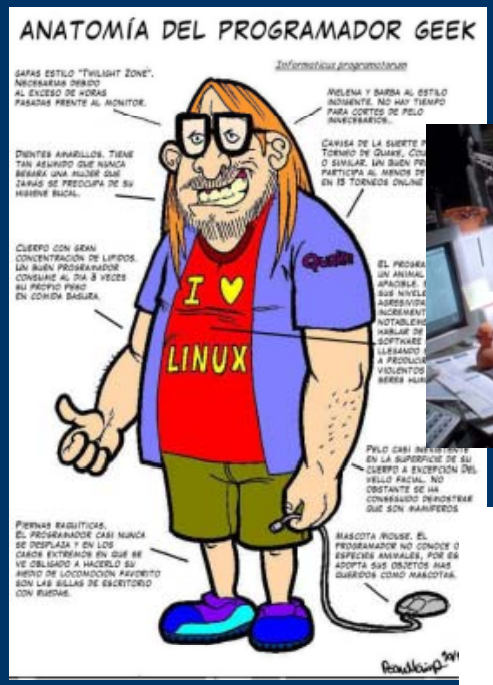
Algoritmo: Descripción precisa de una secuencia de instrucciones para resolver un problema.

Un programa es la codificación de un algoritmo en un lenguaje concreto:

- Secuencia de instrucciones
- Instrucciones que entiende la computadora

Proceso o cómputo: la ejecución de un programa, o una secuencia de instrucciones, en un ordenador.

Programadores



Parque Jurásico

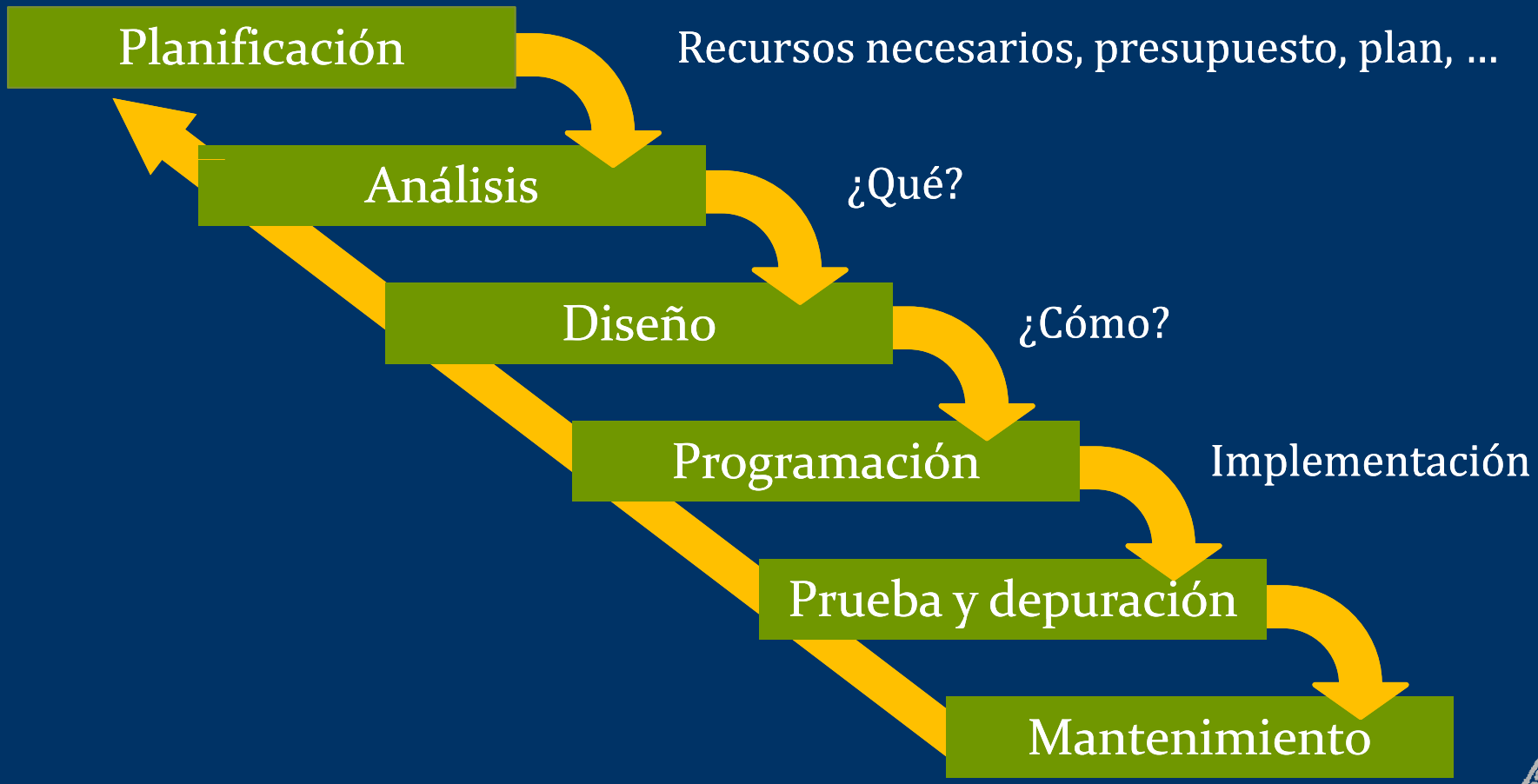
Trabajo en equipo.
Múltiples roles...

- ✓ Analistas
- ✓ Diseñadores
- ✓ Programadores
- ✓ Probadores
- ✓ Administradores de sistemas
- ✓ Etcétera...

La Ingeniería del Software

La programación es sólo una etapa del proceso de desarrollo

Modelo de desarrollo “en cascada”:



Un poco de historia

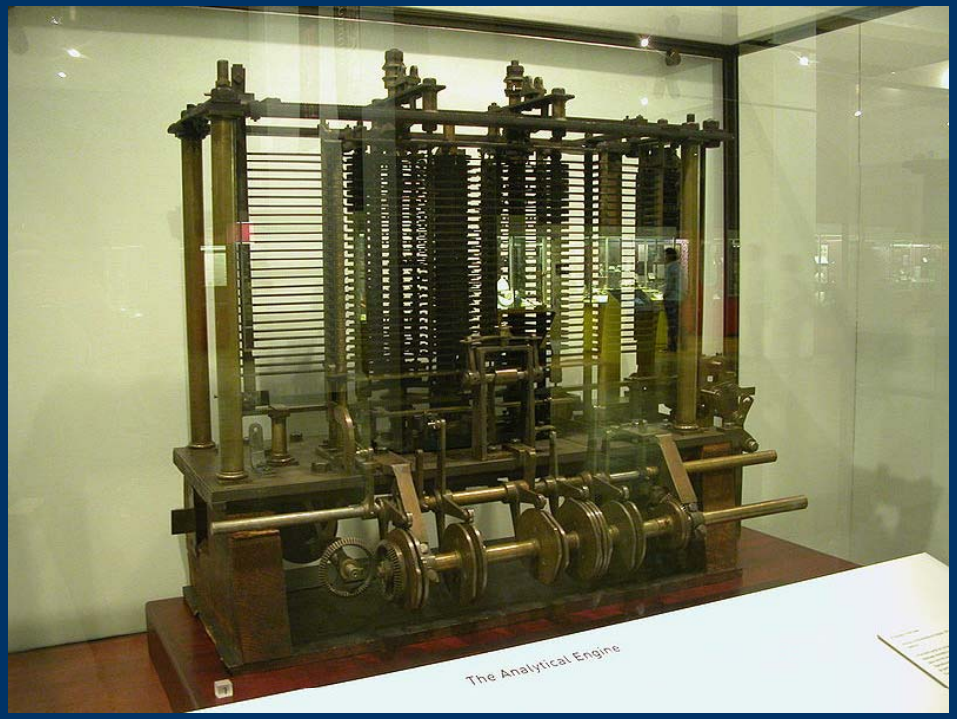
La *prehistoria*

El ábaco

(Wikipedia)

Siglo XIX

Máquina analítica de Charles Babbage



Lady Ada Lovelace es considerada la primera programadora

Un poco de historia

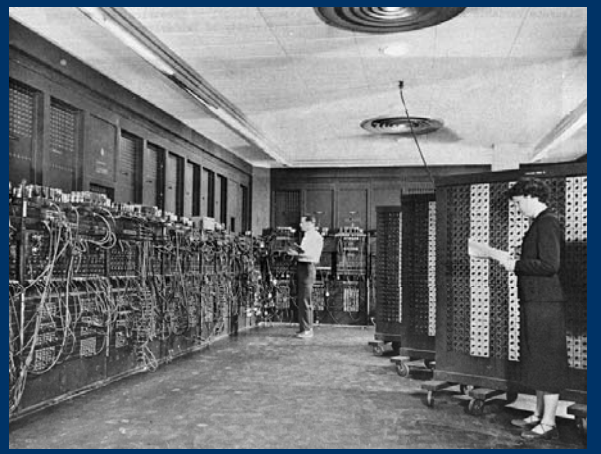
Siglo XX

1936 Máquina de Turing

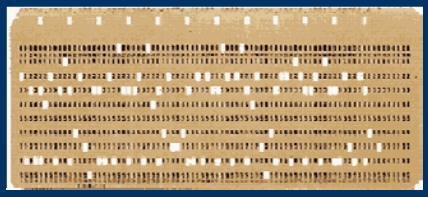
1946 ENIAC: Primera computadora digital de propósito general

1947 El transistor

1953 IBM 650: Primera computadora a gran escala



ENIAC (Wikipedia)



1966 ARPANET: Origen de Internet

1967 El *disquete*

1970 Sistema operativo UNIX

1972 Primer virus informático (*Creeper*)
Lenguaje de programación C

1974 Protocolo TCP. Primera red local

Un poco de historia

1975 Se funda Microsoft

1976 Se funda Apple

1979 Juego *Pacman*

1981 IBM PC
Sistema operativo MS-DOS

1983 Lenguaje de programación C++

1984 CD-ROM

1985 Windows 1.0

1990 Lenguaje HTML
World Wide Web

1991 Sistema operativo Linux

Linux

Apple II (Wikipedia)

IBM PC (Wikipedia)

Un poco de historia

1992 Windows 3.1

1995 Lenguaje de programación Java
DVD

1998 Se funda Google

1999 MSN Messenger

Siglo XXI

2001 Windows XP
Mac OS X

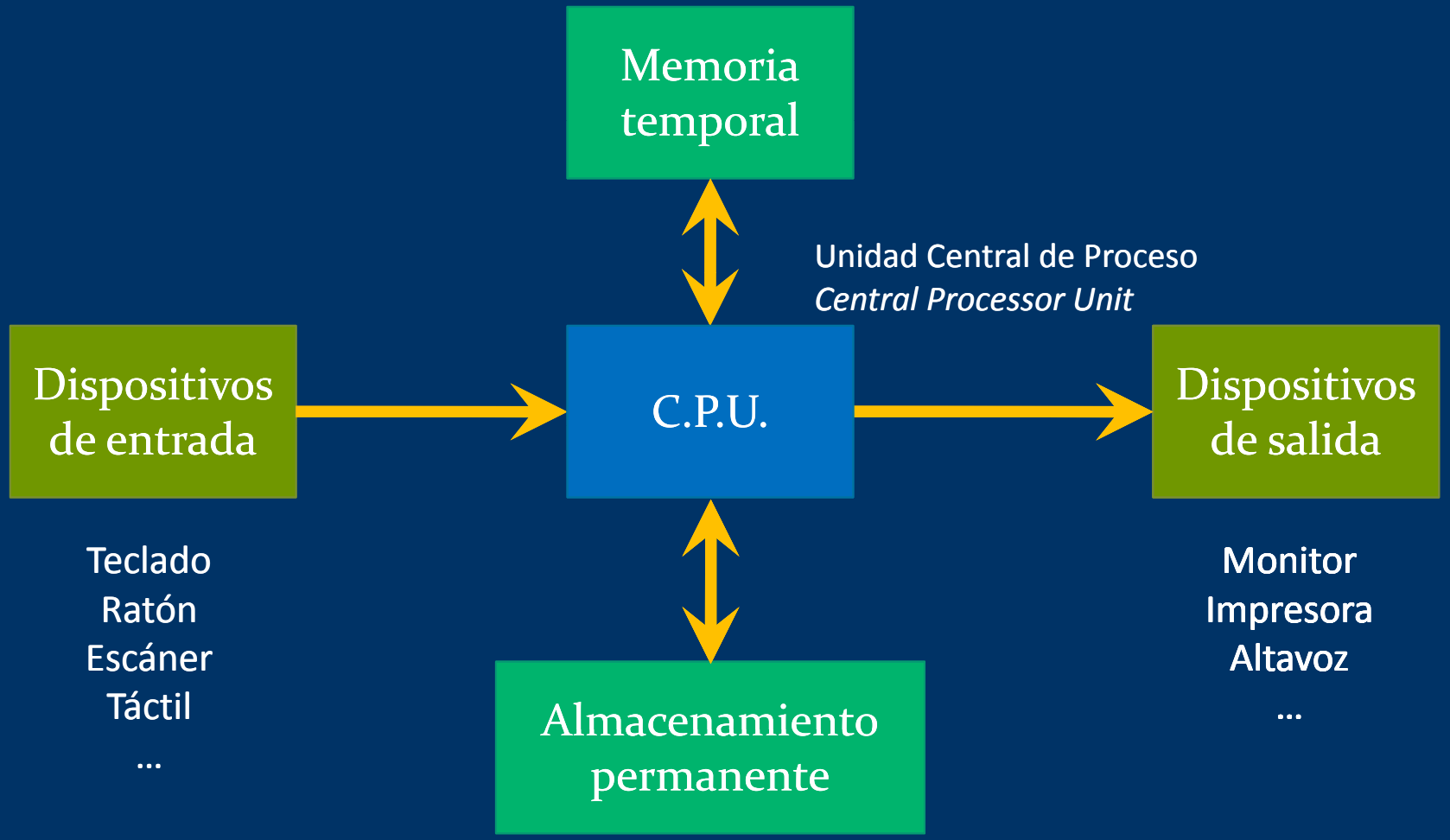
2002 Mozilla Firefox

2007 iPhone

2008 Android ...

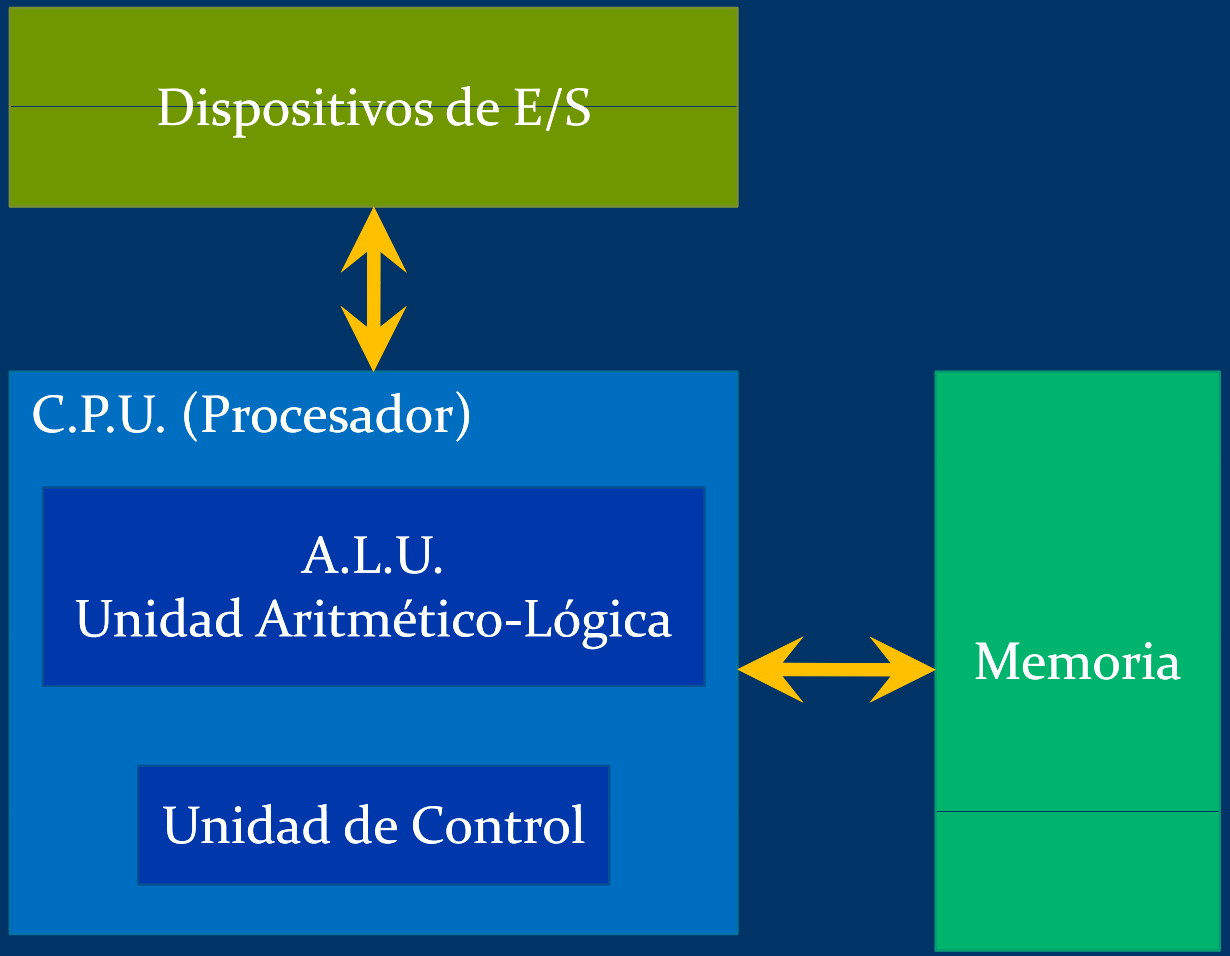
Computadoras

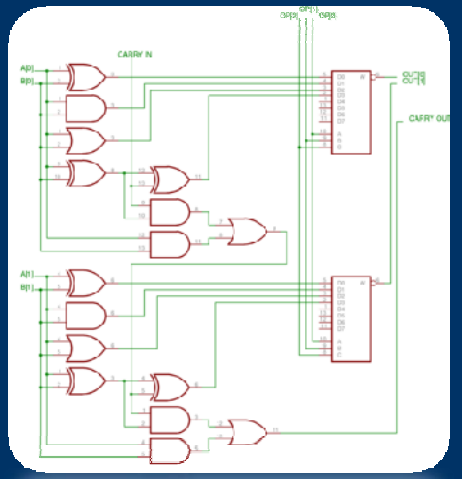
Esquema general



Computadoras

La arquitectura de Von Neumann

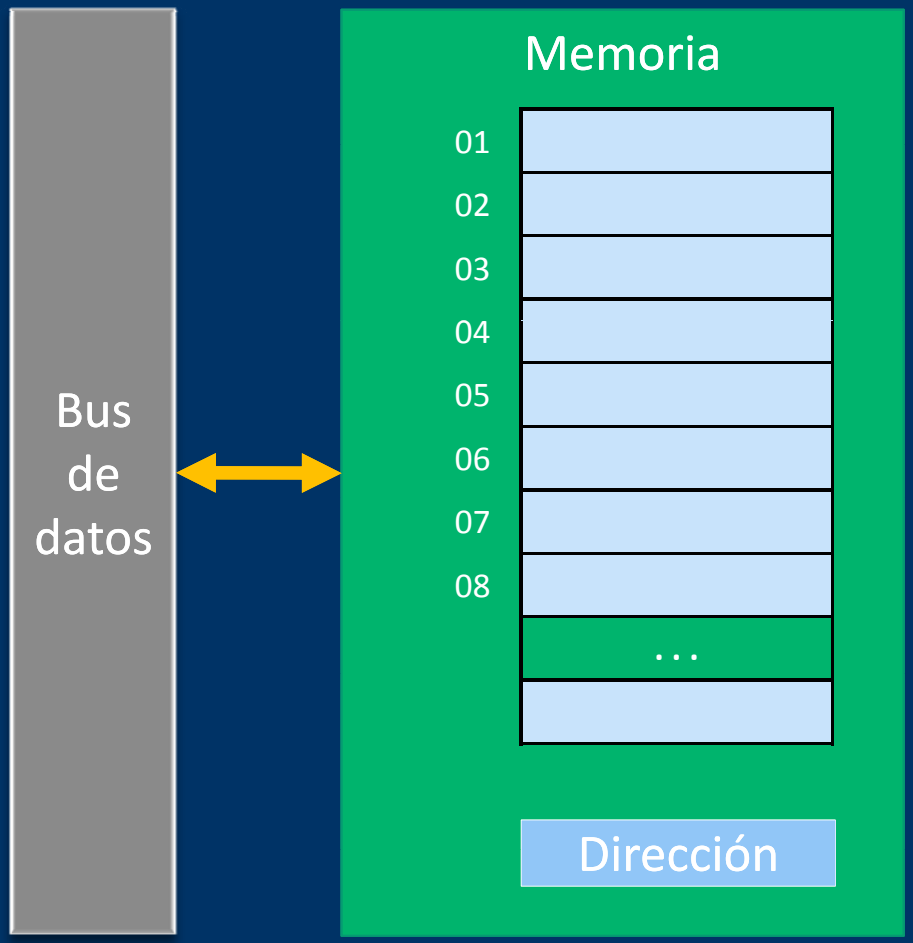




Una ALU de 2 bits (Wikipedia)

Computadoras

La memoria



Cada celda tiene una dirección asociada
Celdas de 8 / 16 / 32 / 64 bits

Información volátil

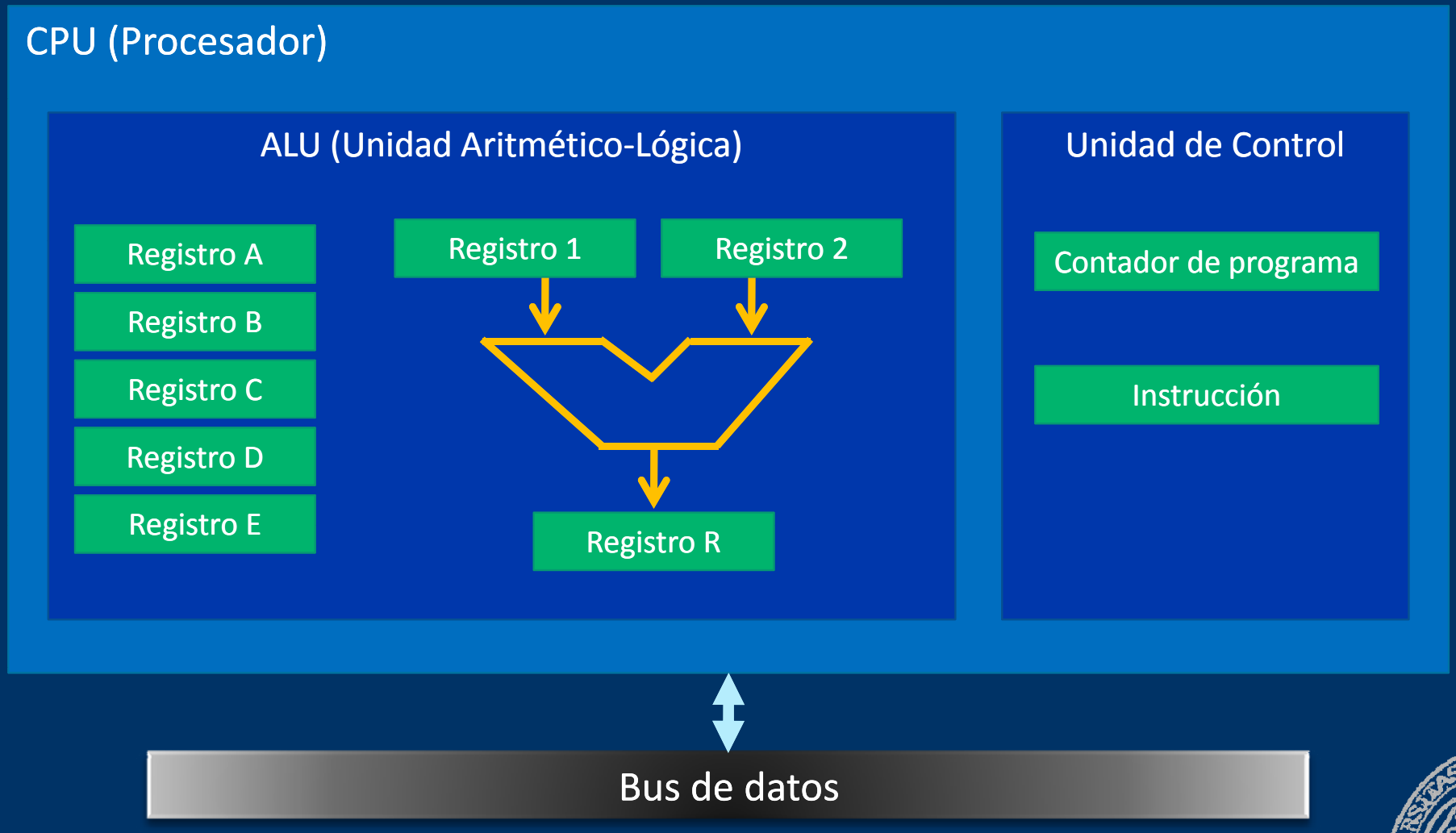
1 Bit = 0 / 1
1 Byte = 8 bits = 1 carácter / nº (0-255)
1 Kilobyte (KB) = 1024 Bytes
1 Megabyte (MB) = 1024 KB
1 Gigabyte (GB) = 1024 MB
1 Terabyte (TB) = 1024 GB
1 Petabyte (PB) = 1024 TB

$2^{10} = 1024 \approx 1000$

Computadoras

Unidad Central de Proceso (CPU)

Registros de 8 / 16 / 32 / 64 bits



Programación de computadoras

Los procesadores trabajan con ceros y unos (bits)

Unidad de memoria básica: *Byte* (8 bits)
(2 dígitos hexadecimales: 01011011 → 0101 1011 → 5B)

Lenguaje máquina

Códigos hexadecimales que representan instrucciones, registros de la CPU, direcciones de memoria o datos

Ejemplo de programa:

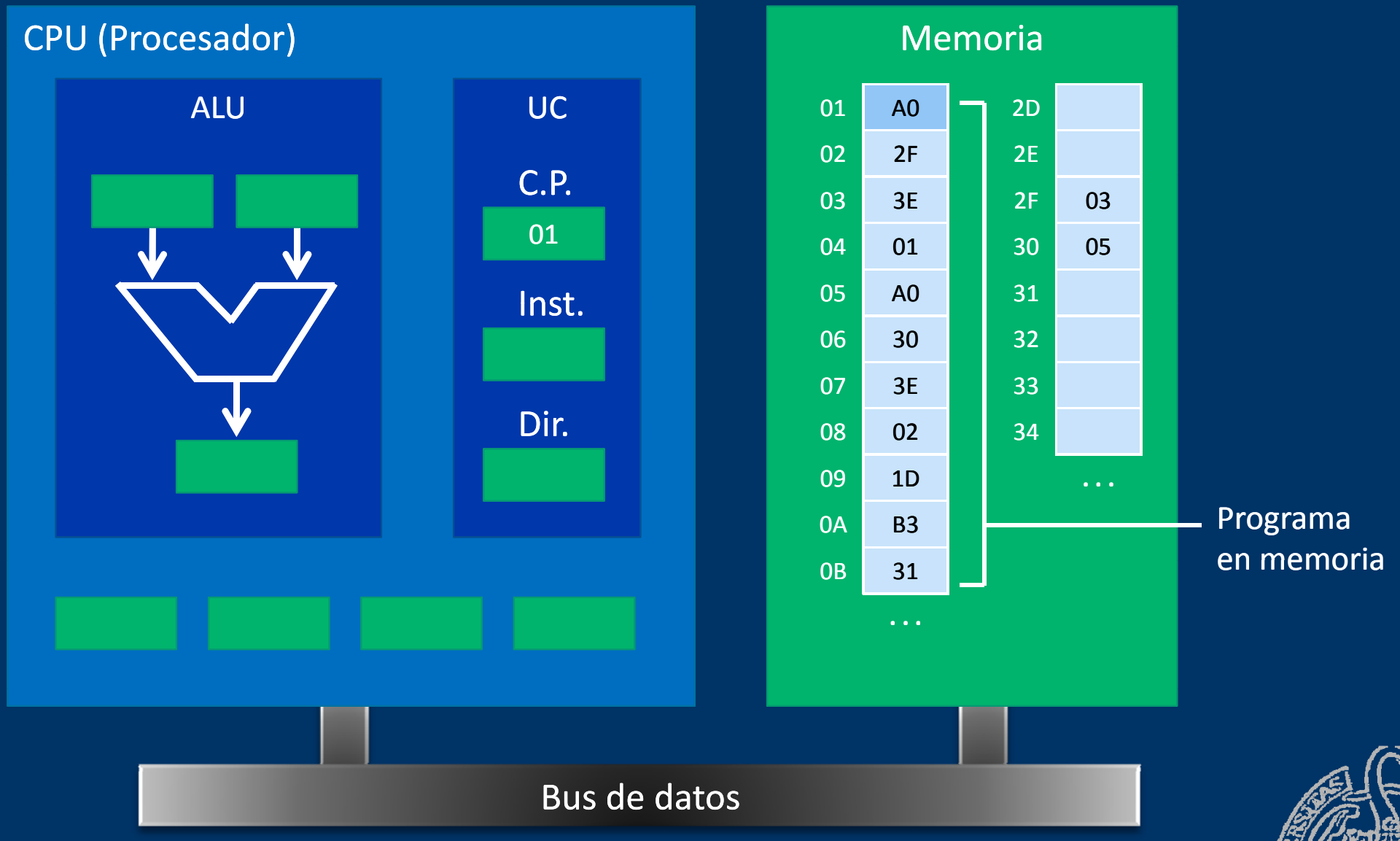
Instrucción	*Significado*
`A0 2F`	*Acceder a la posición de memoria 2F*
`3E 01`	*Copiar el dato en el registro 1 de la ALU*
`A0 30`	*Acceder a la posición de memoria 30*
`3E 02`	*Copiar el dato en el registro 2 de la ALU*
`1D`	*Sumar*
`B3 31`	*Guardar el resultado en la posición de memoria 31*

Lenguaje de bajo nivel

Dependiente de la máquina

Programación intrincada

El lenguaje máquina



Lenguaje ensamblador

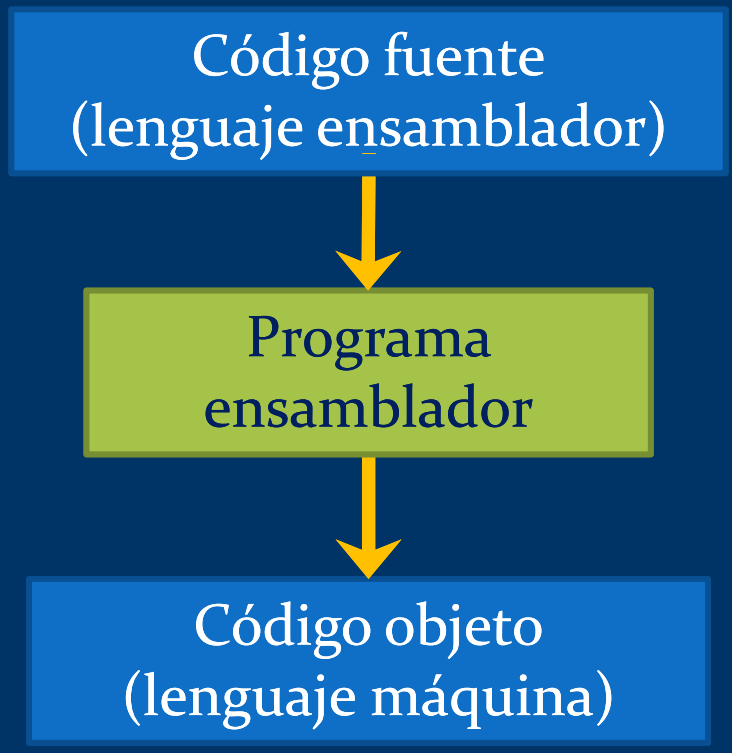
Nemotécnicos para los códigos hexadecimales:

`A0` → `READ` `3E` → `REG` `1D` → `ADD` …

Mayor legibilidad:

```
READ 2F
REG 01
READ 30
REG 02
ADD
WRITE 31
```

Lenguaje de nivel medio



Lenguajes de programación de alto nivel

- ✓ Más cercanos a los lenguajes natural y matemático

```
resultado = dato1 + dato2;
```

- ✓ Mayor legibilidad, mayor facilidad de codificación
- ✓ Estructuración de datos / abstracción procedimental

FORTRAN Python Prolog C#
C Pascal Cobol Lisp Ruby
BASIC Smalltalk Haskell Ada
Simula Java Eiffel C++
...

Lenguajes de programación de alto nivel

El sistema operativo:

Software básico encargado de manejar el hardware y facilitar el trabajo a los programas de aplicación, proporcionándoles un conjunto de servicios genéricos.

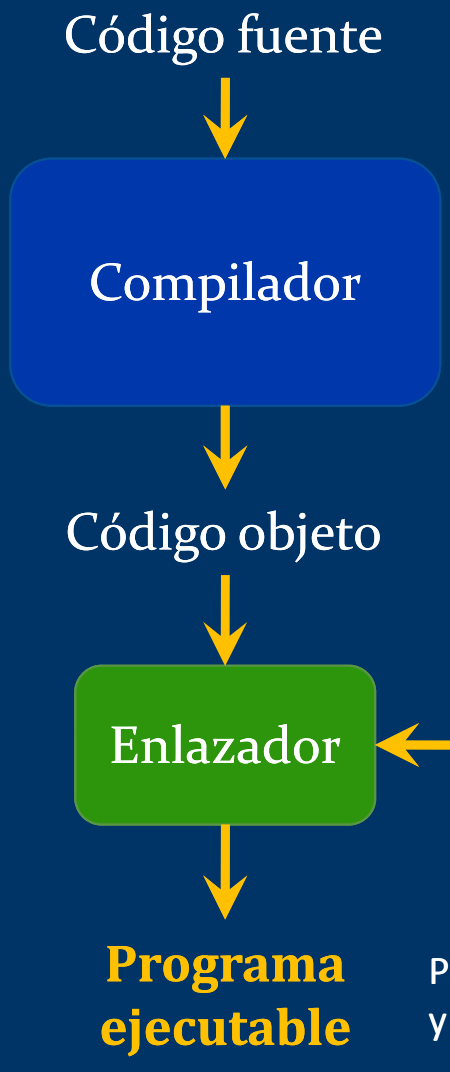
- Interfaz con el usuario.
- Asignación de tiempos de CPU.
- Control y asignación racional de los recursos de la computadora.
- Ejecución de programas.
- Administración de discos y dispositivos.

...

Lenguajes de programación de alto nivel

Modelo de ejecución basado en compilación:
Compilan y enlazan programas completos

Otros modelos:
Intérpretes,
Máquinas virtuales




```
#include <iostream>
using namespace std;

int main()
{
   cout << "Hola Mundo!" << endl;
   return 0;
}
```

01000101001110100111100...

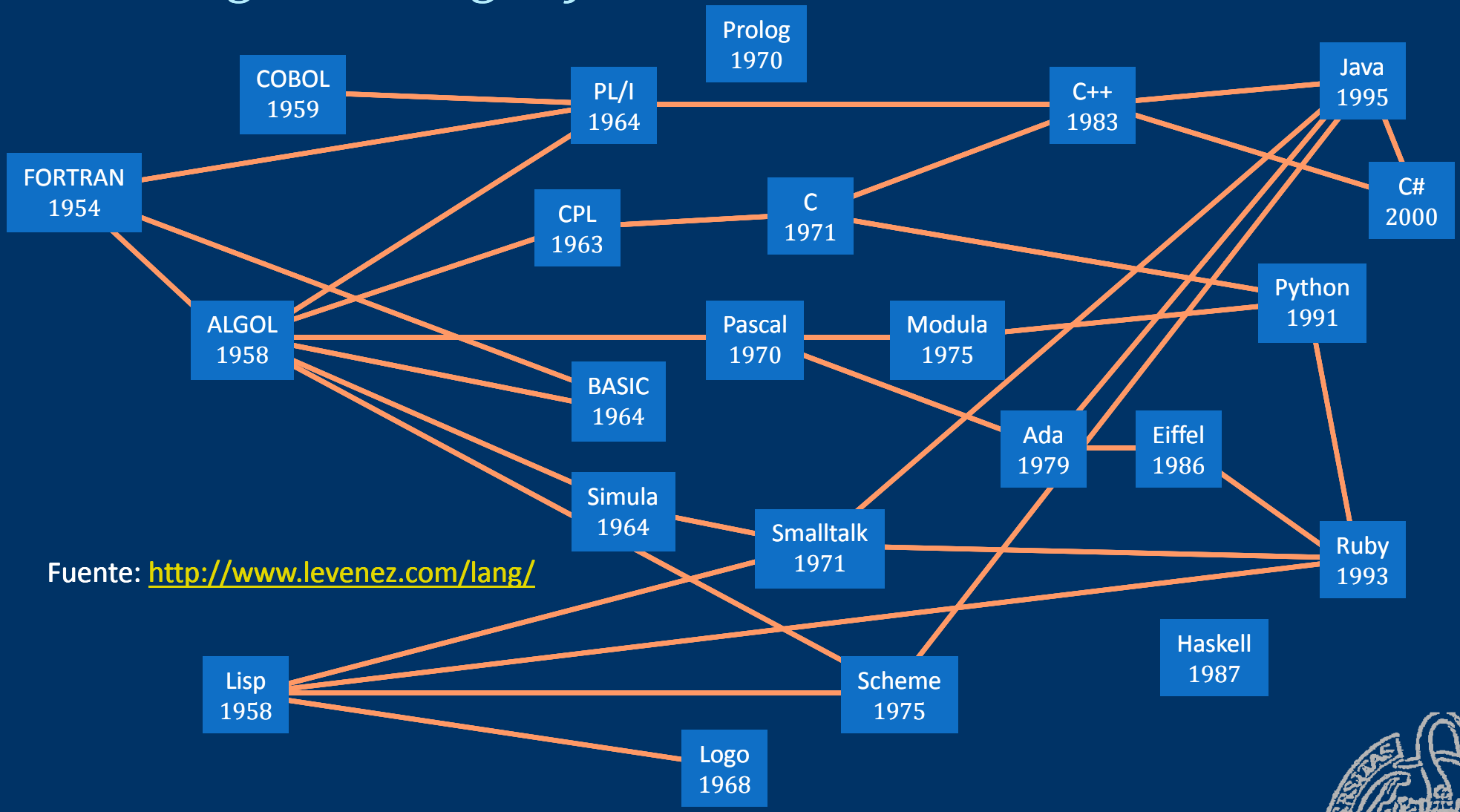
Código objeto de biblioteca

Para una arquitectura concreta y un Sistema Operativo concreto

Los lenguajes de programación de alto nivel

Genealogía de lenguajes

Versiones / Estándares



Fuente: http://www.levenez.com/lang/

¿Por qué C++?

Bjarne Stroustrup (1983)

- ✓ Para programar necesitamos un lenguaje
- ✓ Lenguaje muy ampliamente utilizado en las áreas de ingeniería
- ✓ Bien definido por un estándar (C++ 11)
- ✓ Disponible para casi cualquier computadora
- ✓ Los conceptos son trasladables a otros lenguajes

C++: Un mejor C

La base de C++: El lenguaje C

- ✓ Lenguaje creado por Dennis M. Ritchie en 1972
- ✓ Lenguaje de nivel medio:
  - Estructuras típicas de los lenguajes de alto nivel
  - Construcciones para control a nivel de máquina
- ✓ Lenguaje sencillo (pocas palabras reservadas)
- ✓ Lenguaje estructurado (no estrictamente)
  - Compartimentación de código y datos
  - Componente estructural básico: la función (subprograma)
- ✓ Programación modular

Los lenguajes de programación

Elementos de un lenguaje

- ✓ Instrucciones
- ✓ Datos (literales, variables, tipos)
- ✓ Subprogramas (funciones)
- ✓ Comentarios
- ✓ Directivas
- ✓ …

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hola Mundo!" << endl; // Muestra Hola Mundo!

    return 0;
}
```

Directiva
Subprograma
Dato
Comentario
Instrucción
Instrucción
Dato

Los lenguajes de programación

Sintaxis y semántica de los lenguajes

Sintaxis

- Reglas que determinan cómo se pueden construir y secuenciar los elementos del lenguaje

Semántica

- Significado de cada elemento del lenguaje ¿Para qué sirve?, ¿Qué hace?

Sintaxis de los lenguajes de programación

Especificación

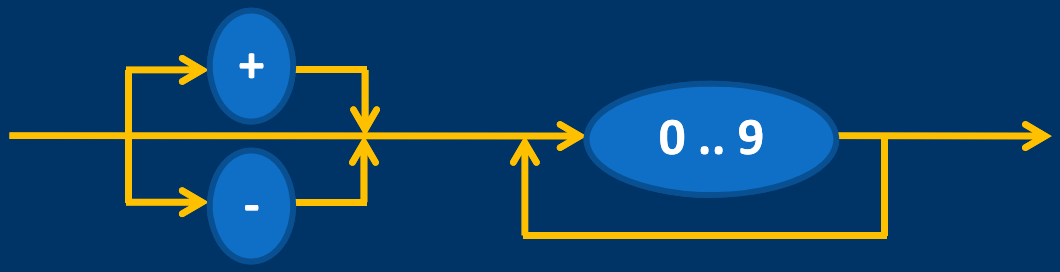
- ✓ Lenguajes (BNF)
- ✓ Diagramas

Ejemplo: Números enteros

BNF

<numero entero> ::= <signo opcional><secuencia de dígitos>
<signo opcional> ::= +|-|<nada>
<secuencia de dígitos> ::= <dígito>|<dígito><secuencia de dígitos>
<dígito> ::= 0|1|2|3|4|5|6|7|8|9
<nada> ::=

| significa ó



+23	✓
-159	✓
1374	✓
1-34	✗
3.4	✗
002	✓

Backus-Naur Form (BNF)

<numero entero> ::= <signo opcional><secuencia de dígitos>
<signo opcional> ::= +|-|<nada>
<secuencia de dígitos> ::= <dígito>|<dígito><secuencia de dígitos>
<dígito> ::= 0|1|2|3|4|5|6|7|8|9
<nada> ::=

+23

<numero entero> ::= <signo opcional><secuencia de dígitos>
::= **+**<secuencia de dígitos> ::= **+**<dígito><secuencia de dígitos>
::= **+2**<secuencia de dígitos> ::= **+2**<dígito> ::= **+23**

1374

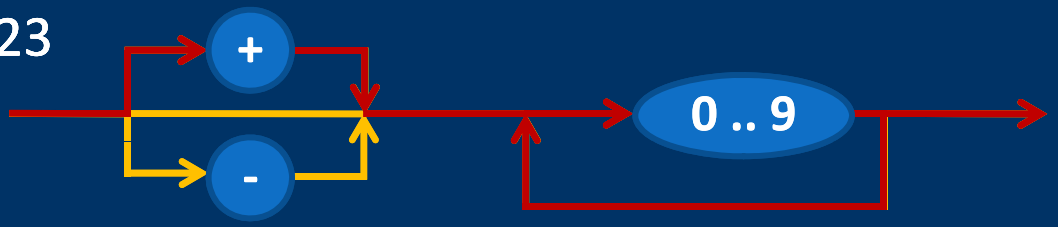
<numero entero> ::= <signo opcional><secuencia de dígitos>
::= <secuencia de dígitos> ::= <dígito><secuencia de dígitos>
::= **1**<secuencia de dígitos> ::= **1**<dígito><secuencia de dígitos>
::= **13**<secuencia de dígitos> ::= **13**<dígito><secuencia de dígitos>
::= **137**<secuencia de dígitos> ::= **137**<dígito> ::= **1374**

1-34

<numero entero> ::= <signo opcional><secuencia de dígitos>
::= <secuencia de dígitos> ::= <dígito><secuencia de dígitos>
::= **1**<secuencia de dígitos> ::= **ERROR** (**-** no es <dígito>)

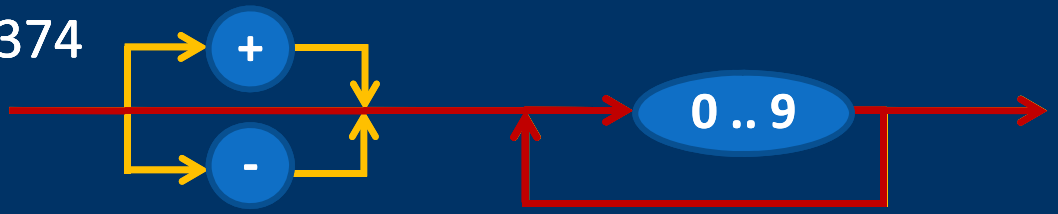
Diagramas de sintaxis

+23



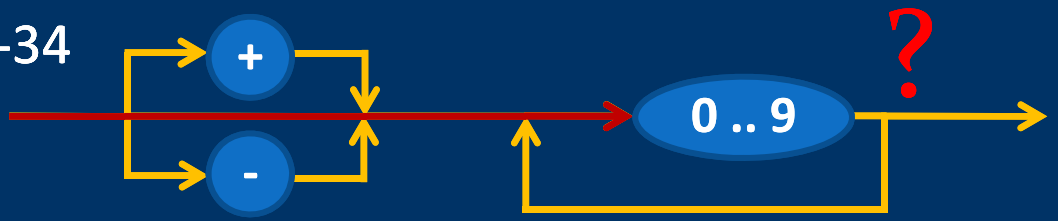
+23 ✓

1374



1374 ✓

1-34

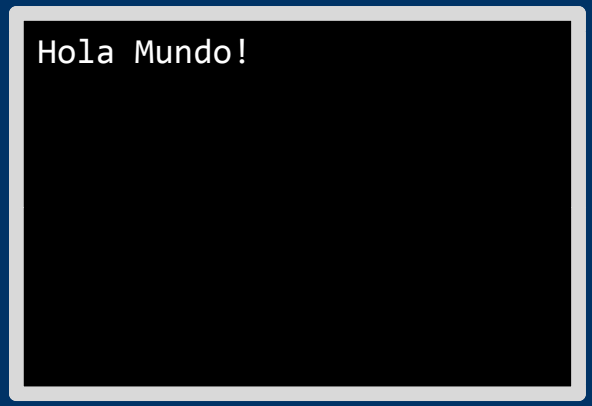


1- ✗

Un primer programa en C++

Hola Mundo!

Un programa que muestra un saludo en la pantalla:



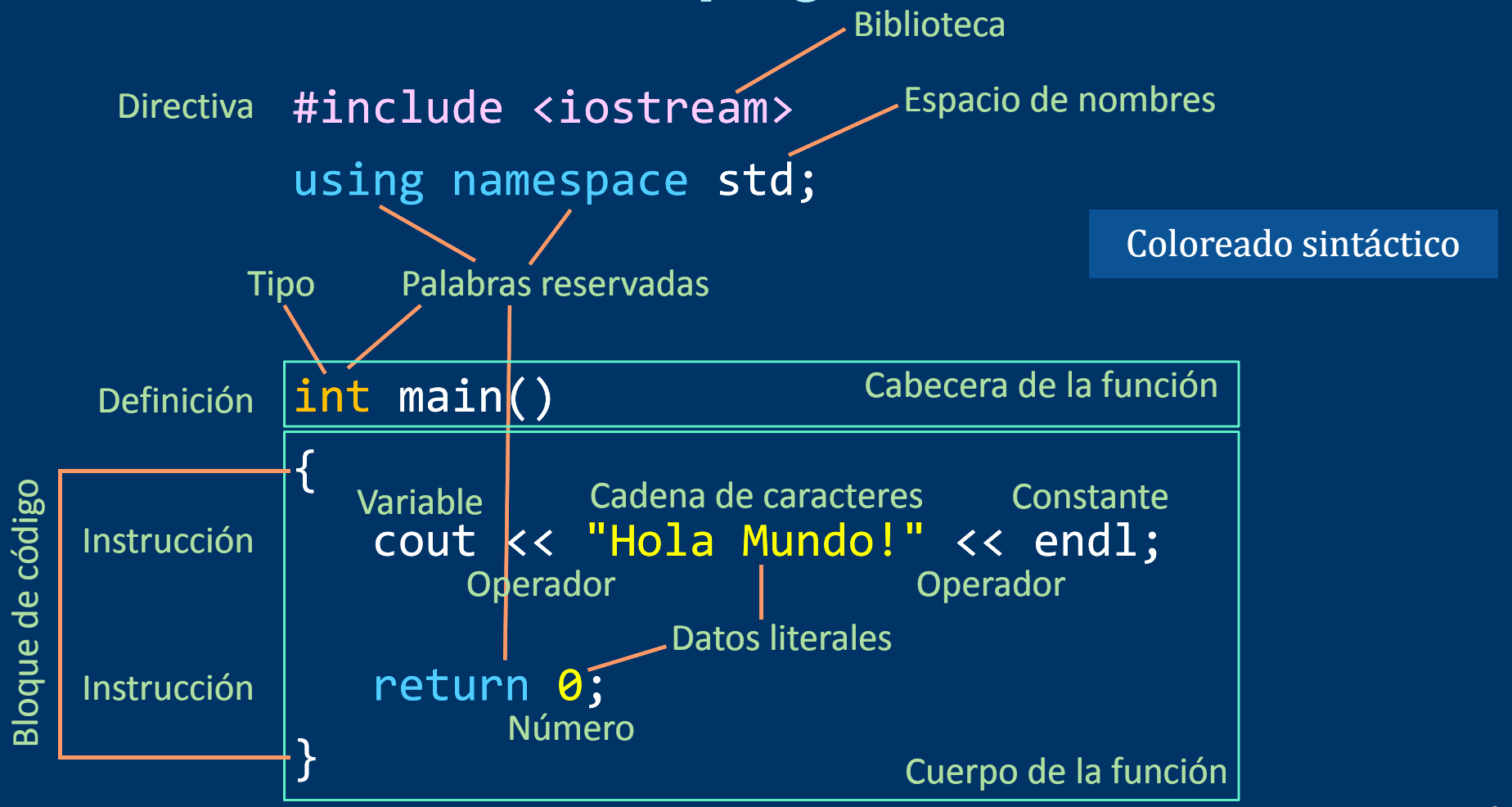

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hola Mundo!" << endl; // Muestra Hola Mundo!

    return 0;
}
```

Un primer programa en C++

Elementos sintácticos del programa



Las instrucciones terminan en ;

Un primer programa en C++: ejecución

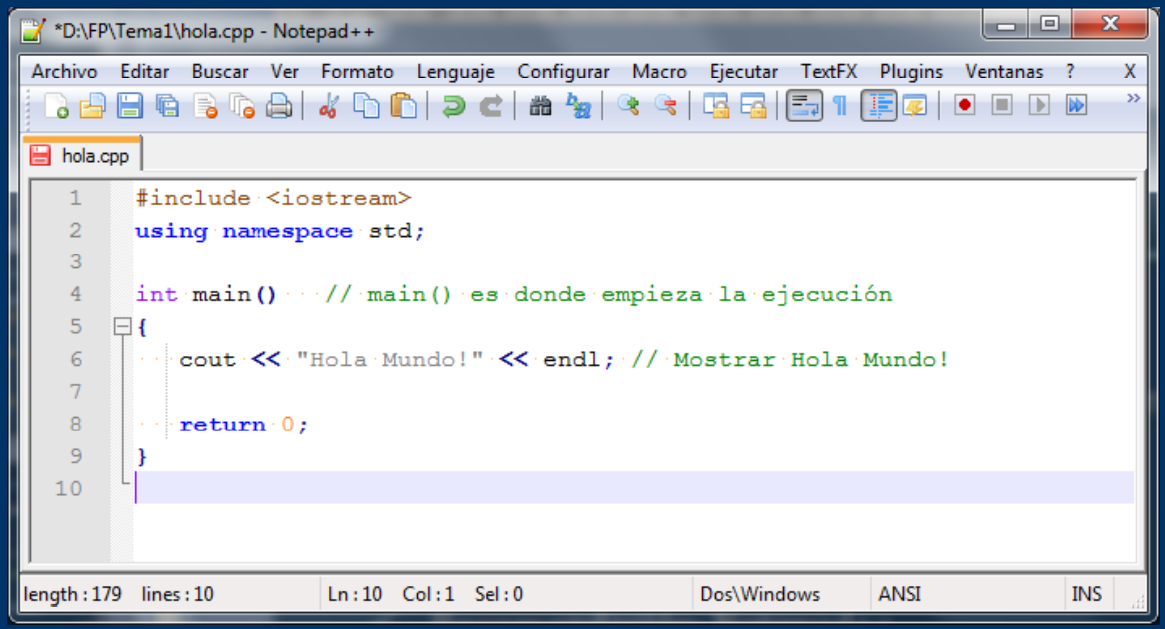
¿Qué hace el programa?

- ✓ La ejecución del programa siempre empieza en la función `main()`
- ✓ Se ejecutan las instrucciones en secuencia de principio a fin

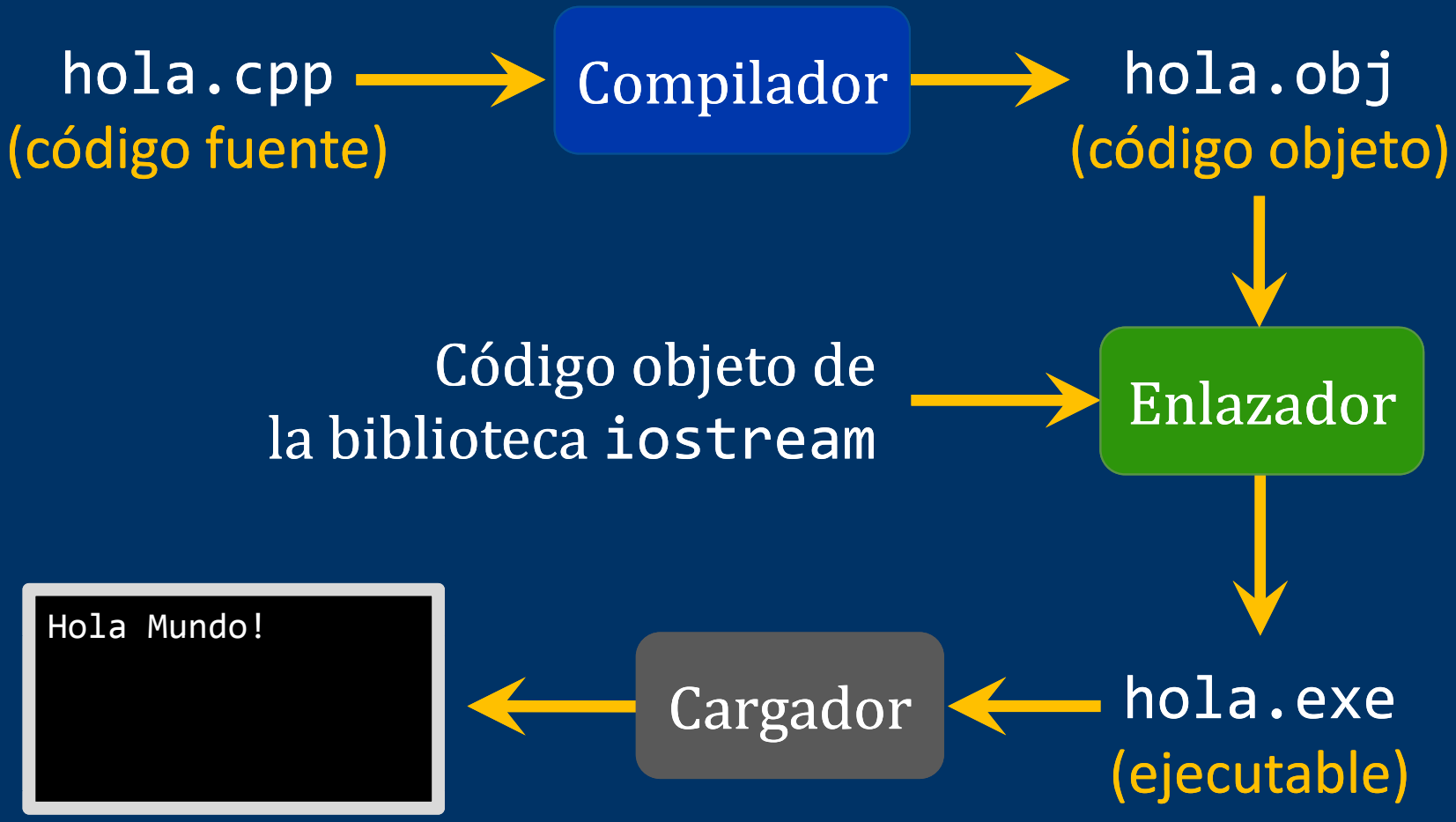
Herramientas de desarrollo

Editor

- ✓ Bloc de notas, Wordpad, Writer, Gedit, Kwrite, …
  (texto simple, sin formatos)
- ✓ Editores específicos, coloreado sintáctico: Emacs, Notepad++



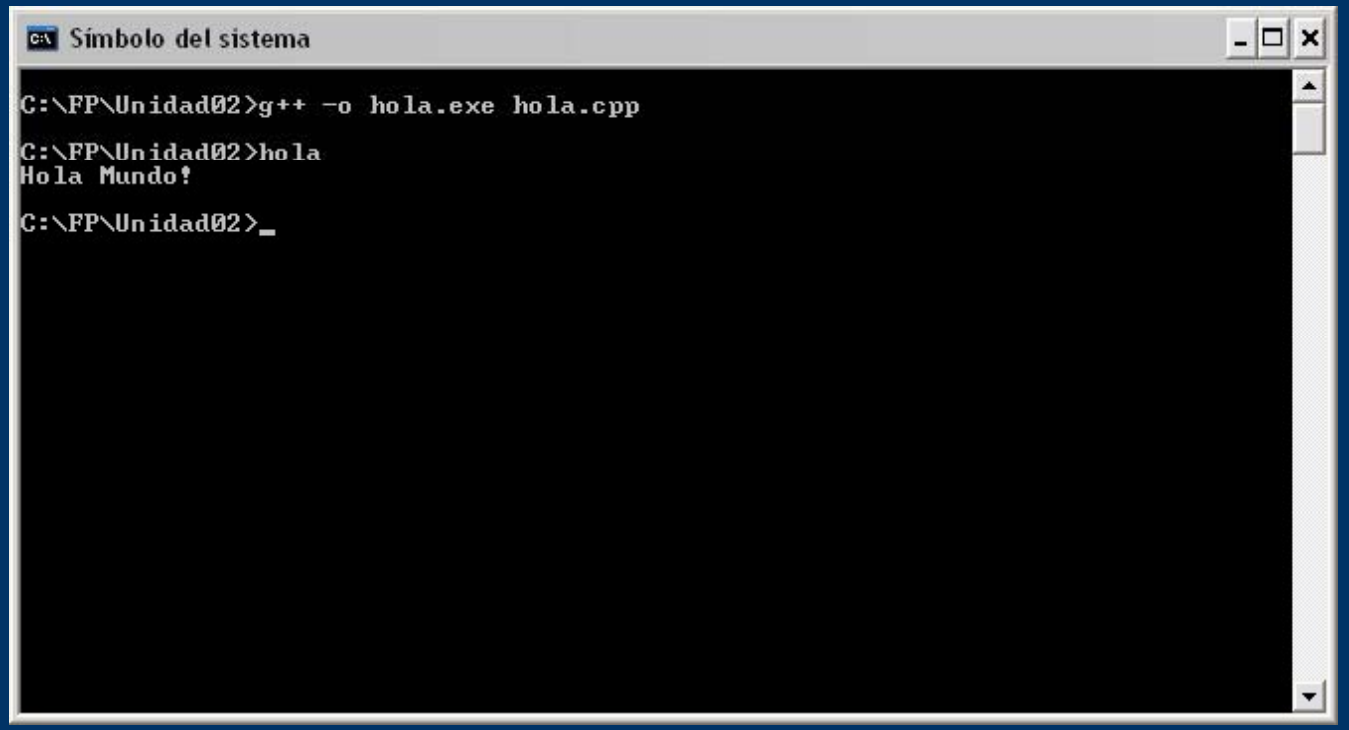
Compilación, enlace y ejecución



Más herramientas de desarrollo

Compilador

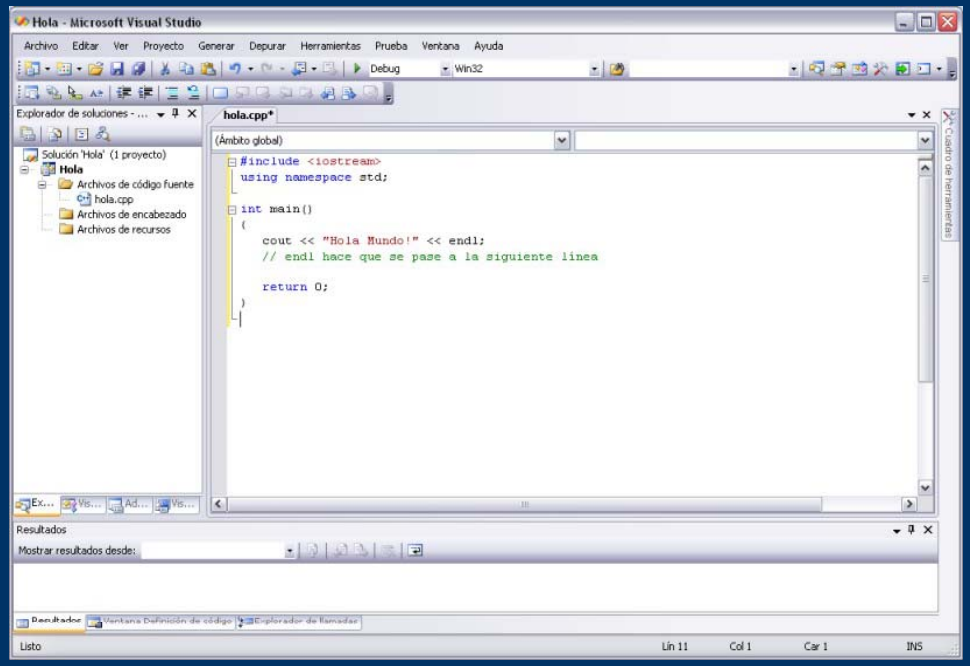
- ✓ Importante: C++ estándar (C++11)
  GNU G++ (*MinGW* en Windows),
  MS Visual Studio, Borland C++, …



Más herramientas de desarrollo

Entornos de desarrollo (IDE)

- ✓ Para editar, compilar y probar el código del programa
- ✓ Recomendaciones:
  - Windows: MS Visual Studio (MS Visual C++ Express) o Eclipse
  - Linux: Netbeans o Eclipse



Un ejemplo de programación

Un ejemplo de programación

Sintaxis y semántica de los lenguajes

Sintaxis

- Reglas que determinan cómo se pueden construir y secuenciar los elementos del lenguaje

Semántica

- Significado de cada elemento del lenguaje
  ¿Para qué sirve?, ¿Qué hace?

  Reglas que determinan el efecto da cada instrucción

Un ejemplo de programación

Una computadora de un coche

Coche que acepta programas que le indican una ruta.

Instrucciones que entiende:

<instrucción> ::= <inst> ;

<inst> ::= `Start` | `Stop` | <avanzar>

<avanzar> ::= `Go` <dirección> <num> `Blocks`

<dirección> ::= `North` | `East` | `South` | `West`

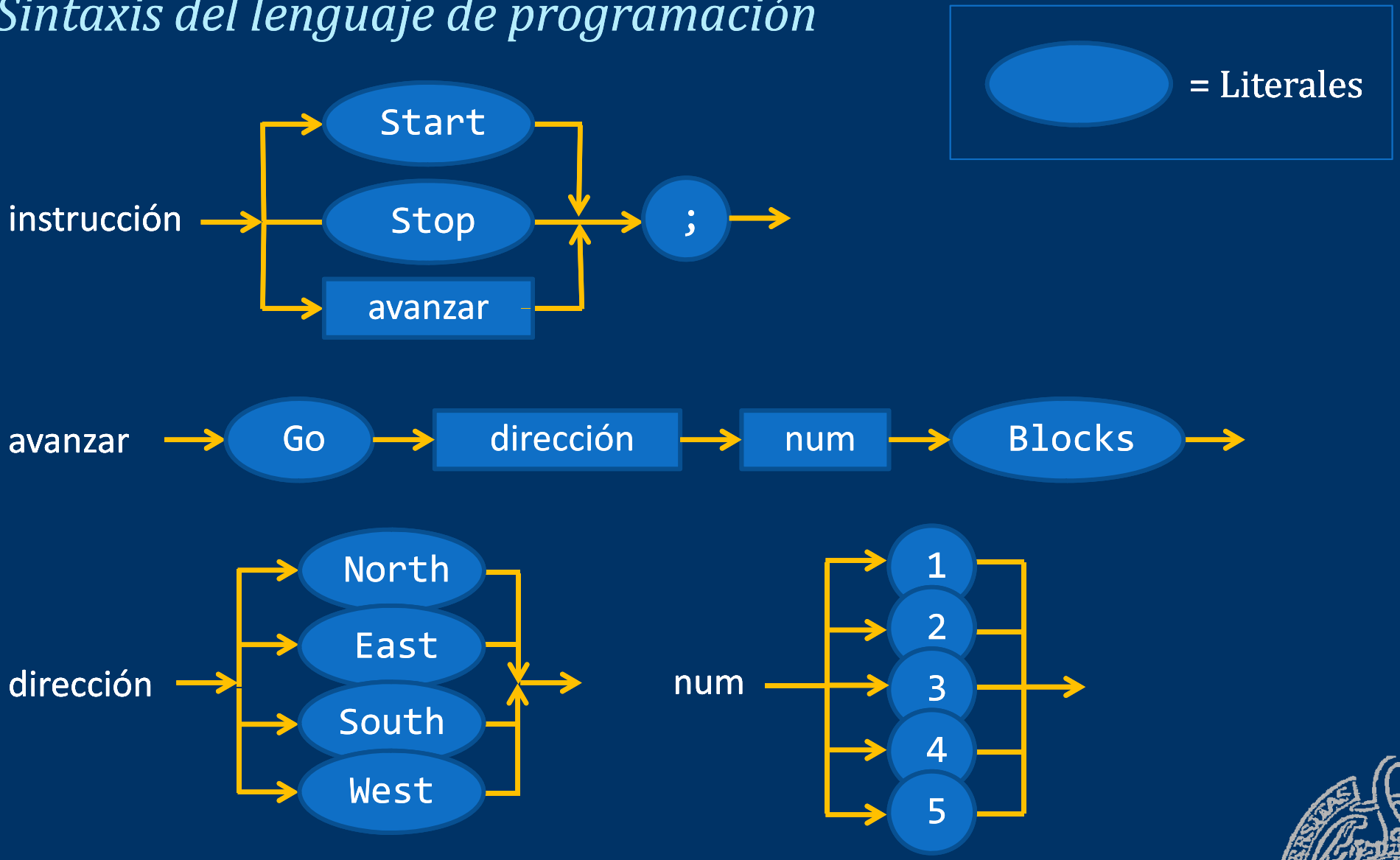
<num> ::= `1` | `2` | `3` | `4` | `5`

Ejemplos:

```
Start;
Go North 3 Blocks;
Stop;
```

Un ejemplo de programación

Sintaxis del lenguaje de programación



Un ejemplo de programación

El problema a resolver

Estando el coche en la posición A, conseguir llegar al Cine B.

¿Qué pasos hay que seguir ?

Arrancar

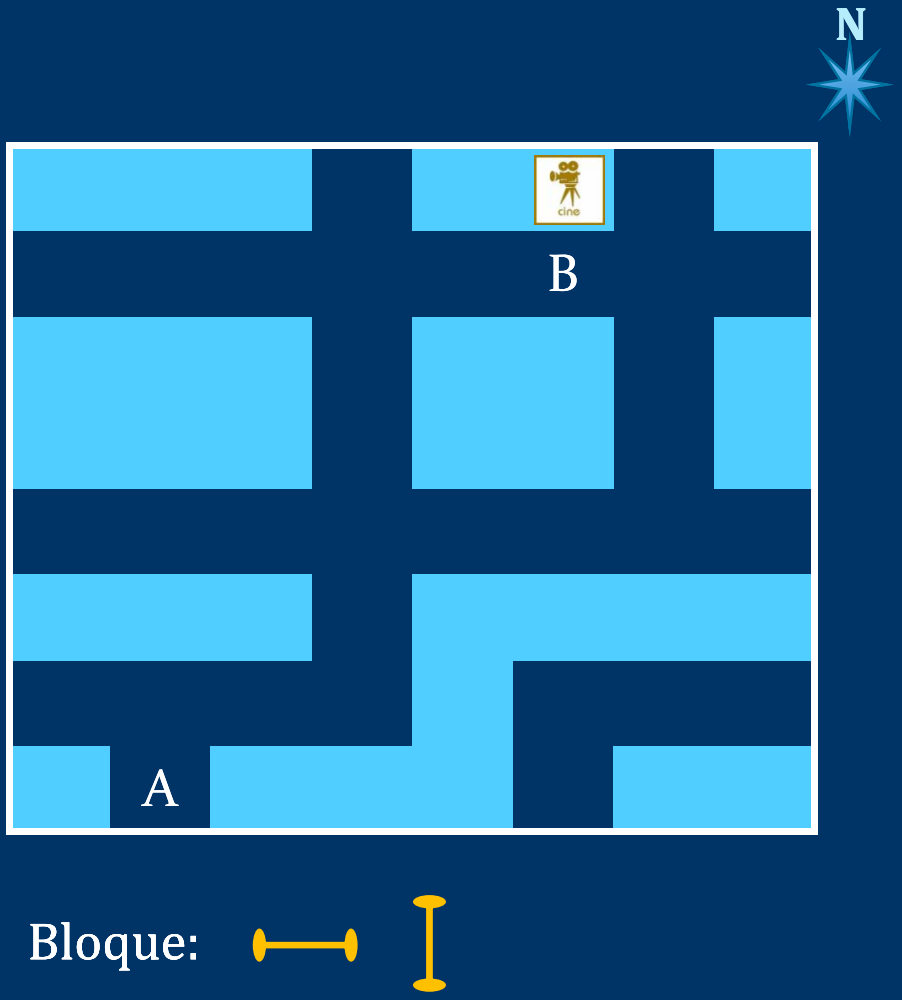
Ir un bloque al Norte

Ir dos bloques al Este

Ir cinco bloques al Norte

Ir dos bloques al Este

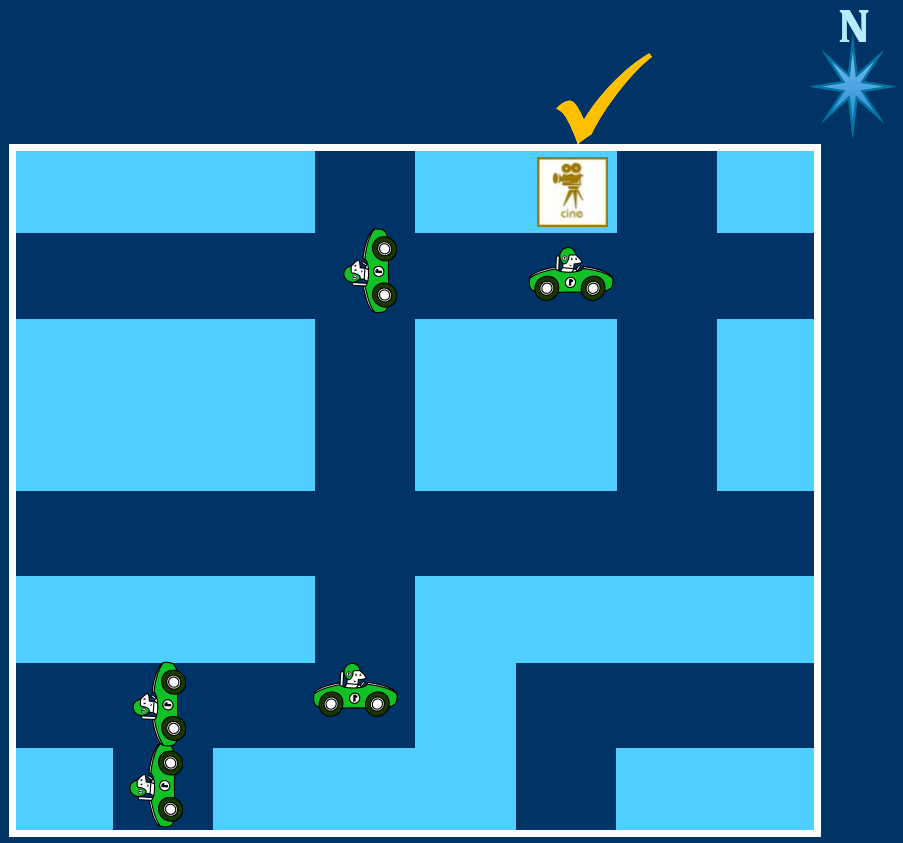
Parar



Un ejemplo de programación

El algoritmo

Secuencia de pasos que hay que seguir para resolver el problema.



1.- Arrancar

2.- Ir un bloque al Norte

3.- Ir dos bloques al Este

4.- Ir cinco bloques al Norte

5.- Ir dos bloques al Este

6.- Parar

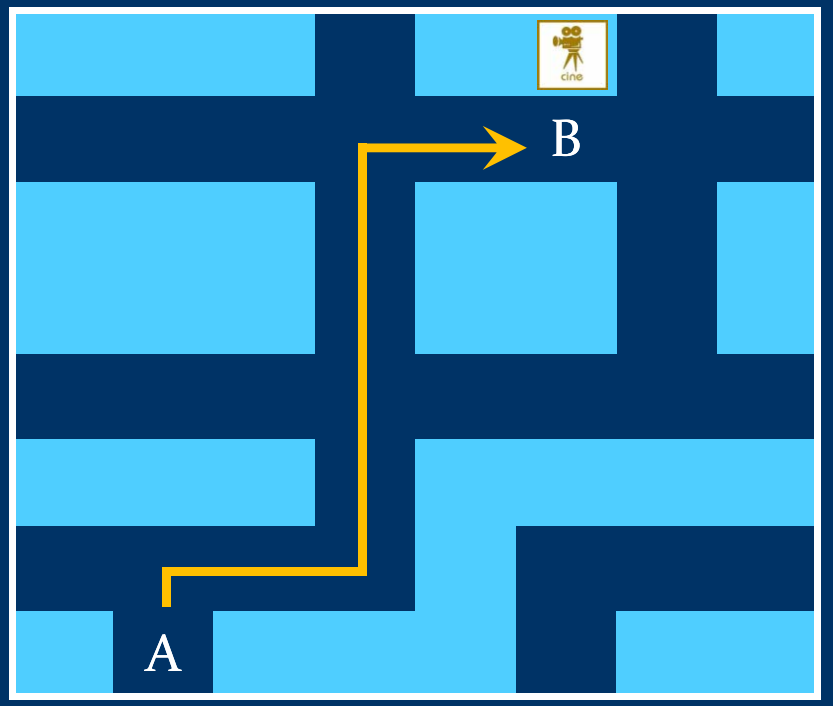
Estas instrucciones sirven tanto para una persona como para una computadora.

Un ejemplo de programación

El programa

Escribir el algoritmo en el lenguaje de programación.

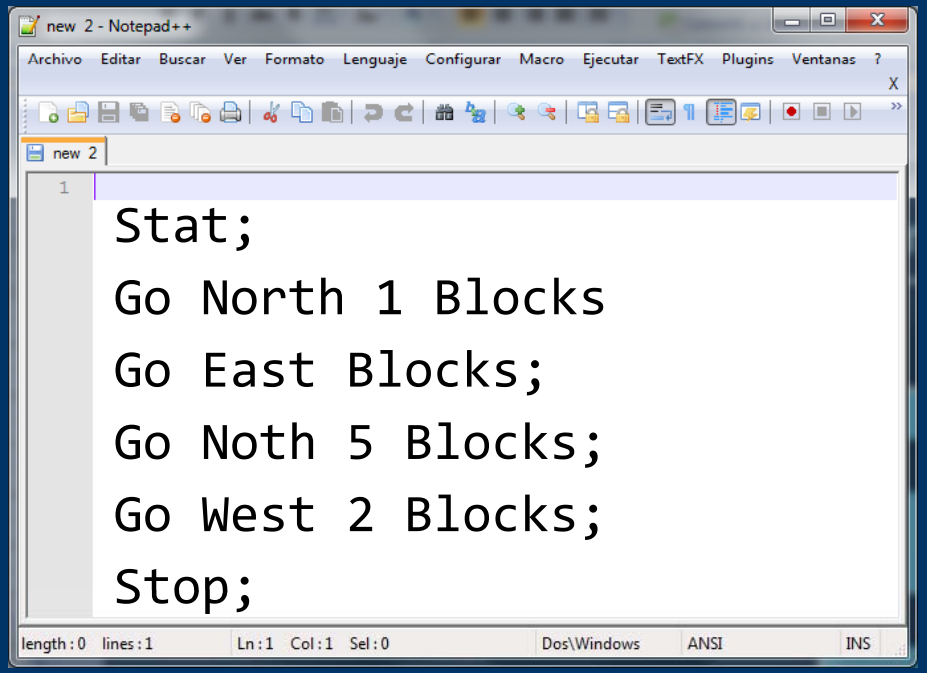
```
Start;
Go North 1 Blocks;
Go East 2 Blocks;
Go North 5 Blocks;
Go East 2 Blocks;
Stop;
```



Un ejemplo de programación

El programa

Escribimos el código del programa
en un editor y lo guardamos
en un documento `programa.prg`.



Copiamos el archivo
en una llave USB
y lo llevamos al coche.

Un ejemplo de programación

La compilación

Introducimos la llave USB en el coche y pulsamos el botón de ejecutar el programa:

```
Stat;
----^ Unknown word.
Go North 1 Blocks
-----------------^ ; missing.
Go East Blocks;
--------^ Number missing.
Go Noth 5 Blocks;
-------^ Unknown word.
Go West 2 Blocks;
Stop;
There are errors. Impossible to run the program.
```

Errores de sintaxis

Un ejemplo de programación

Depuración

Editamos el código para arreglar los errores de sintaxis.

```
Stat;
Go North 1 Blocks
Go East Blocks;
Go Noth 5 Blocks;
Go West 2 Blocks;
Stop;
```

```
Start;
Go North 1 Blocks;
Go East 3 Blocks;
Go North 5 Blocks;
Go West 2 Blocks;
Stop;
```

Un ejemplo de programación

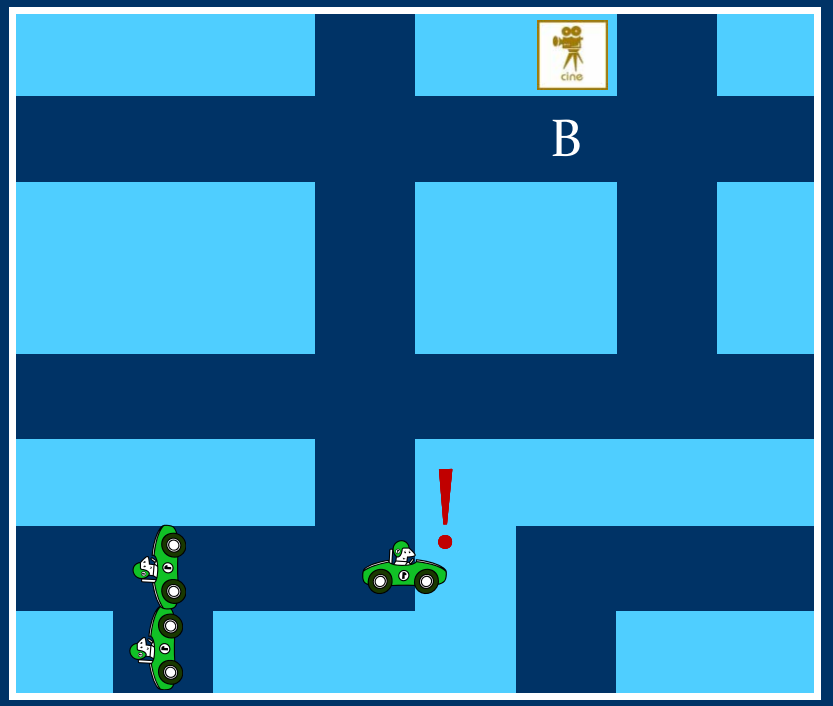
La ejecución

Se realiza lo que pide cada instrucción.

```
Start;

Go North 1 Blocks;

Go East 3 Blocks;
```



Error de ejecución

¡Una instrucción no se puede ejecutar!

Un ejemplo de programación

Depuración

Editamos el código para arreglar el error de ejecución.

```
Start;
Go North 1 Blocks;
Go East 3 Blocks;
Go North 5 Blocks;
Go West 2 Blocks;
Stop;
```

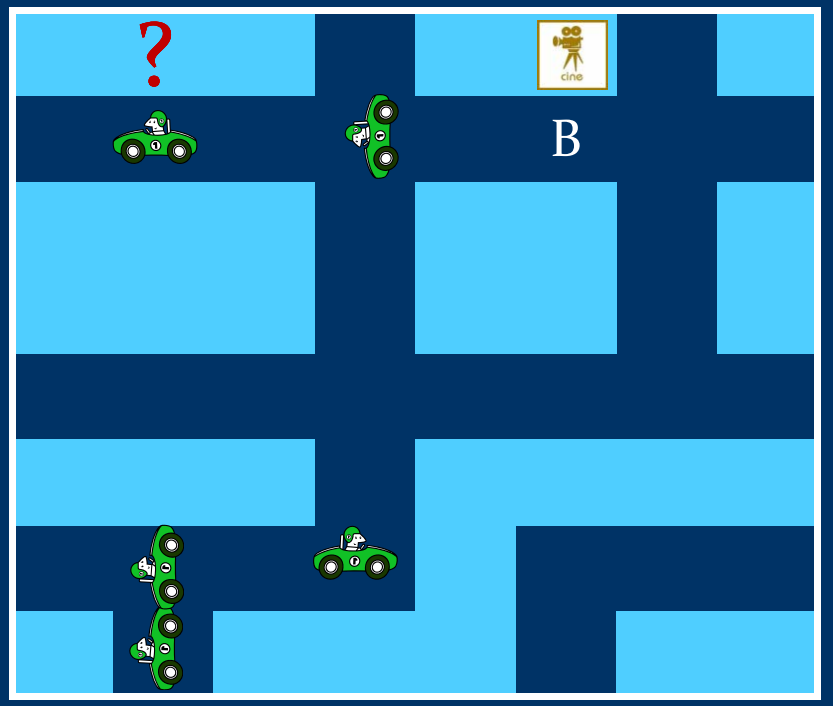
```
Start;
Go North 1 Blocks;
Go East 2 Blocks;
Go North 5 Blocks;
Go West 2 Blocks;
Stop;
```

Un ejemplo de programación

La ejecución

Se realiza lo que pide cada instrucción.

```
Start;
Go North 1 Blocks;
Go East 2 Blocks;
Go North 5 Blocks;
Go West 2 Blocks;
Stop;
```



Error lógico

¡El programa se ejecuta, pero no obtiene el resultado deseado!

Un ejemplo de programación

Depuración

Editamos el código para arreglar el error lógico.

```
Start;
Go North 1 Blocks;
Go East 2 Blocks;
Go North 5 Blocks;
Go West 2 Blocks;
Stop;
```

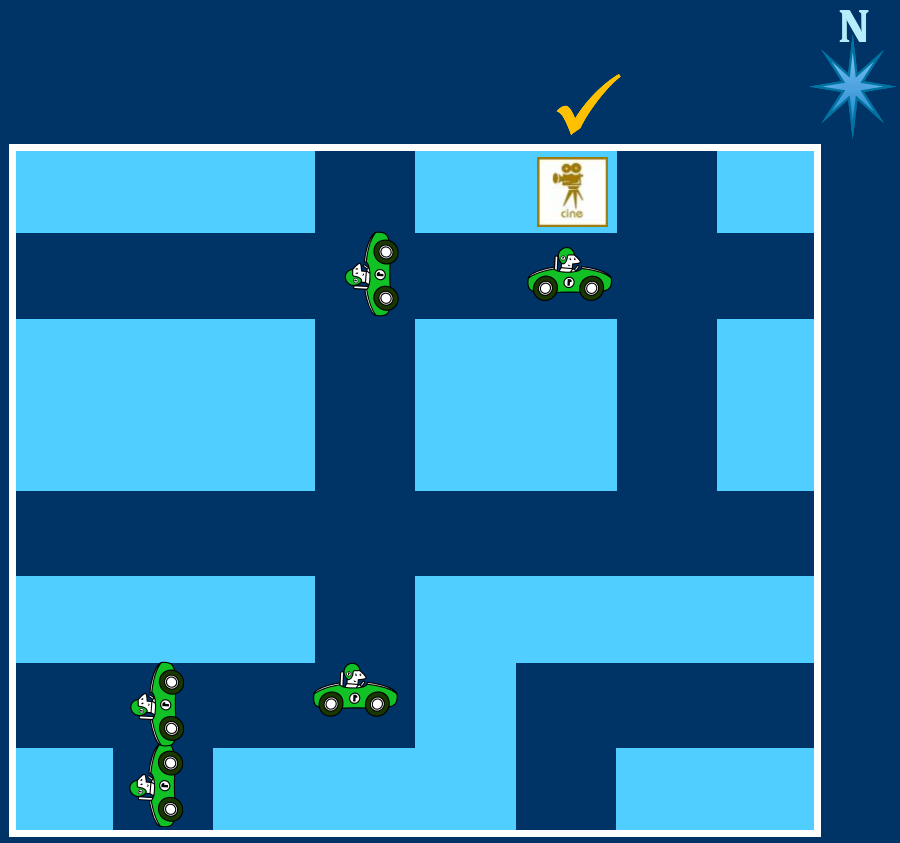
```
Start;
Go North 1 Blocks;
Go East 2 Blocks;
Go North 5 Blocks;
Go East 2 Blocks;
Stop;
```

Un ejemplo de programación

La ejecución

Se realiza lo que pide cada instrucción.

```
Start;
Go North 1 Blocks;
Go East 2 Blocks;
Go North 5 Blocks;
Go East 2 Blocks;
Stop;
```



¡Conseguido!

Un ejemplo de programación

Sintaxis del lenguaje de programación

Instrucciones que entiende:

<instrucción> ::= <inst> ;

<inst> ::= `Start` | `Stop` | <avanzar>

<avanzar> ::= `Go` <dirección> <num> `Blocks`

<dirección> ::= `North` | `East` | `South` | `West`

<num> ::= `1` | `2` | `3` | `4` | `5`

Semántica del lenguaje

¿Qué hace cada instrucción?: modifica el estado del coche

Variables que definen el estado:

arrancado, posición(x, y), chocado

Un ejemplo de programación

Semántica del lenguaje

¿Qué hace cada instrucción?: modifica el estado del coche

Variables que definen el estado:

arrancado, posición(x, y), chocado

Las coordenadas (x, y) sobre el mapa: (0,0) esquina sup. izq.

El estado, respecto a un mapa, queda definido por la tripla:

<arrancado, posición(x, y), chocado>

El mapa queda definido por una matriz N filas y M columnas:

Mapa(f, c) = 1 <-> calle

Mapa(f, c) = 0 <-> edificio

Un ejemplo de programación

Especificación del problema

El problema en estos términos quedaría especificado por:

¿Cómo pasar de un

- ✓ estado inicial <posición(FA, CA)> a un
  estado final <posición(FB, CB)> ?

- ✓ estado inicial <posición(FA, CA), chocado> a un
  estado final <posición(FB, CB)> ?

- ✓ estado inicial <posición(FA, CA), No chocado> a un
  estado final <No arrancado, posición(FB, CB), No chocado>

Un ejemplo de programación

Semántica de las instrucciones mediante
reglas de cambio de estado: EstA <inst> EstD

EstA: Estado antes de la ejecución de la instrucción
<inst> ejecución de una instrucción
EstD: Estado después de la ejecución de la instrucción

Para cada instrucción:

EstA: <posición(F, C), No chocado>
`Start`
EstD: <arrancado, posición(F, C), No chocado>

Un ejemplo de programación

Para cada instrucción: EstA <inst> EstD

- EstA: <posición(F, C), No chocado>

  `Start`

  EstD: <arrancado, posición(F, C), No chocado>

- EstA: <posición(F, C), No chocado>

  `Stop`

  EstD: <No arrancado, posición(F, C), No chocado>

- EstA: <[No]arrancado, posición(F, C), chocado>

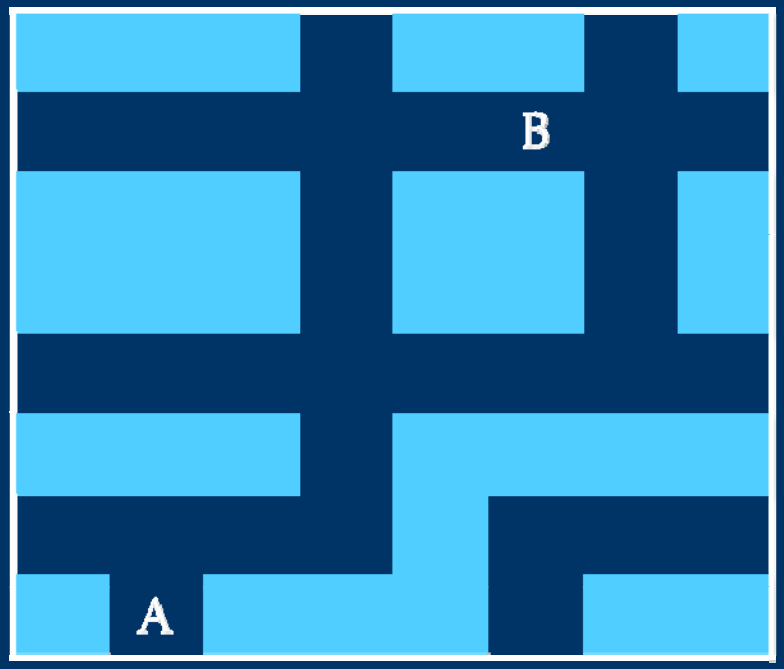
  `<inst>`

  EstD: <[No]arrancado, posición(F, C), chocado>

Un ejemplo de programación

El mapa, matriz de N filas y M columnas:
Mapa(F, C) = 1 <-> calle, Mapa(F, C) = 0 <-> edificio

- EstA: <arrancado, posición(F, C), No chocado>
  AND (F-n) >=0 AND (p.t. i:1…n, Mapa(F-i, C)=calle)
  `Go North n Blocks`
  EstD: <arrancado, posición(F-n, C), No chocado>

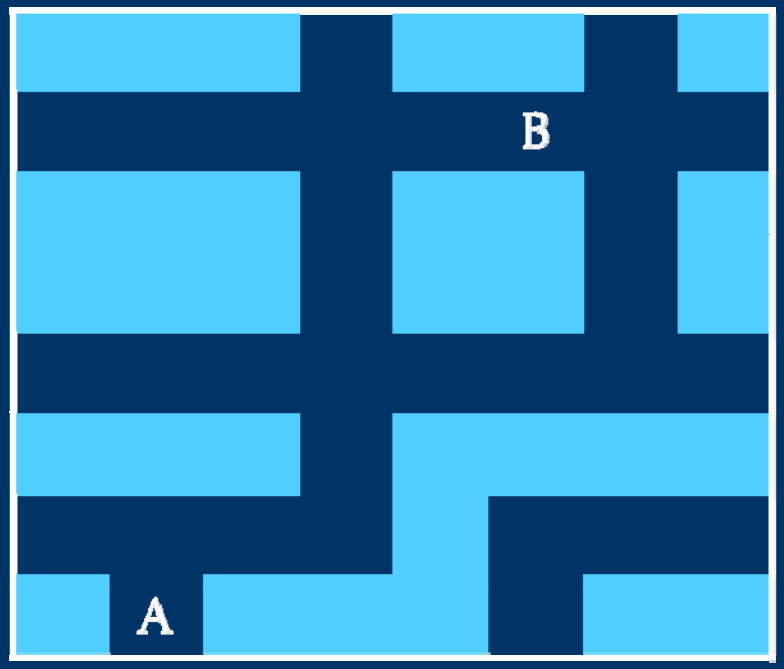


Análogo para `East`, `South` y `West`

Un ejemplo de programación

El mapa, matriz de N filas y M columnas:
Mapa(F, C) = 1 <-> calle, Mapa(F, C) = 0 <-> edificio

- EstA: <arrancado, posición(F, C), No chocado>
  AND (F-n <0) OR (p.a. i:1…n, Mapa(F-i, C)=edificio)
  `Go North n Blocks`
  EstD: <posición(F, C), chocado>



Análogo para `East`, `South` y `West`

Acerca de *Creative Commons*

Licencia CC (Creative Commons)

Este tipo de licencias ofrecen algunos derechos a terceras personas bajo ciertas condiciones.

Este documento tiene establecidas las siguientes:

- Reconocimiento (*Attribution*):
  En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.
- No comercial (*Non commercial*):
  La explotación de la obra queda limitada a usos no comerciales.
- Compartir igual (*Share alike*):
  La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Pulsa en la imagen de arriba a la derecha para saber más.