1. Computadores y programación

Informática y computadora (RAE)

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.*

Hardware y Software (RAE)

Hardware

Componentes que integran la parte física de una computadora.

Software

Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

El Sistema Operativo

• Es el software básico para manejar el hardware y proporcionar un conjunto de servicios genérico al resto de programas, las aplicaciones.

• Es la primera capa de software que tiene un computadora, que establece un puente entre el hardware y el resto del software (aplicaciones)... pero en definitiva es un programa

• Sistemas operativos más usuales: Linux, Windows (en sus distintas versiones), Mac OS, Solaris, Unix, Android...

• En FP nos interesa aprender a diseñar aplicaciones. No vemos los sistemas operativos a fondo.

Programas, algoritmo, cómputo

Algoritmo

Descripción precisa y ordenada de una secuencia (finita) de operaciones que permite hallar la solución de un problema.

Conocemos algún algoritmo?
Al ser una secuencia finita de instrucciones, siempre terminará... es cierto? NO

Programa

Codificación de un algoritmo en un lenguaje de programación concreto (C#, C++, Java, ...), mediante una secuencia de instrucciones que entiende la computadora.

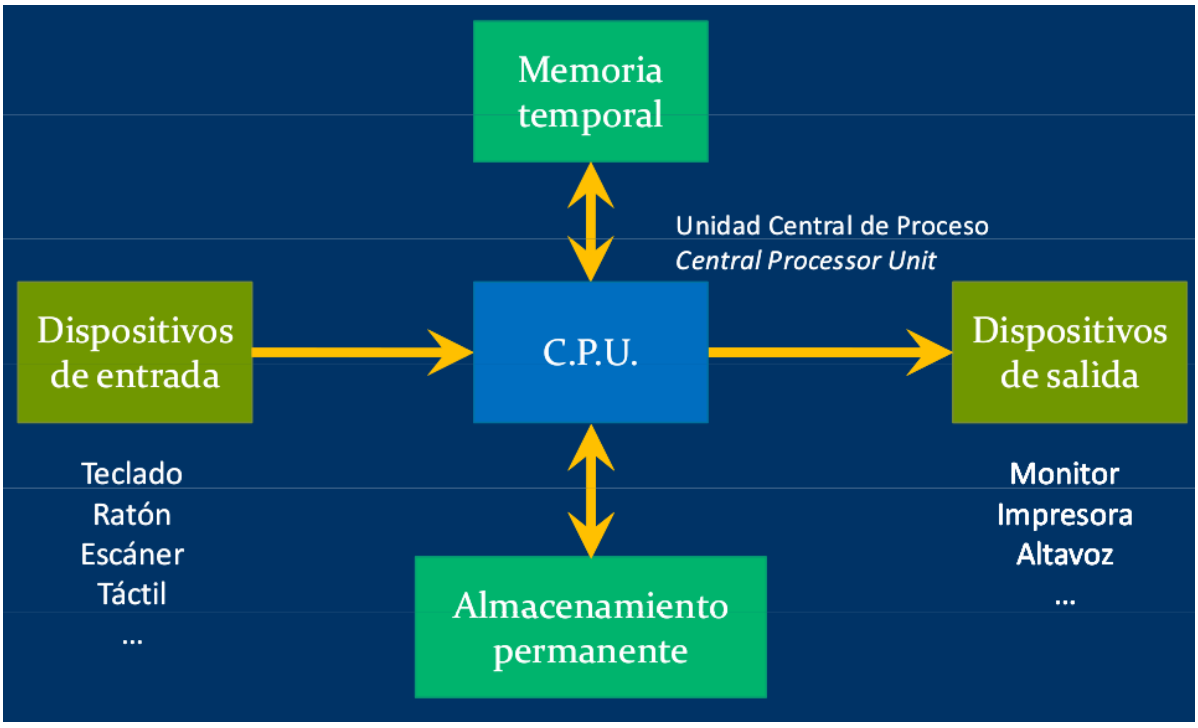
Son distintos programa y algoritmo? Conocemos algún programa?

Cómputo

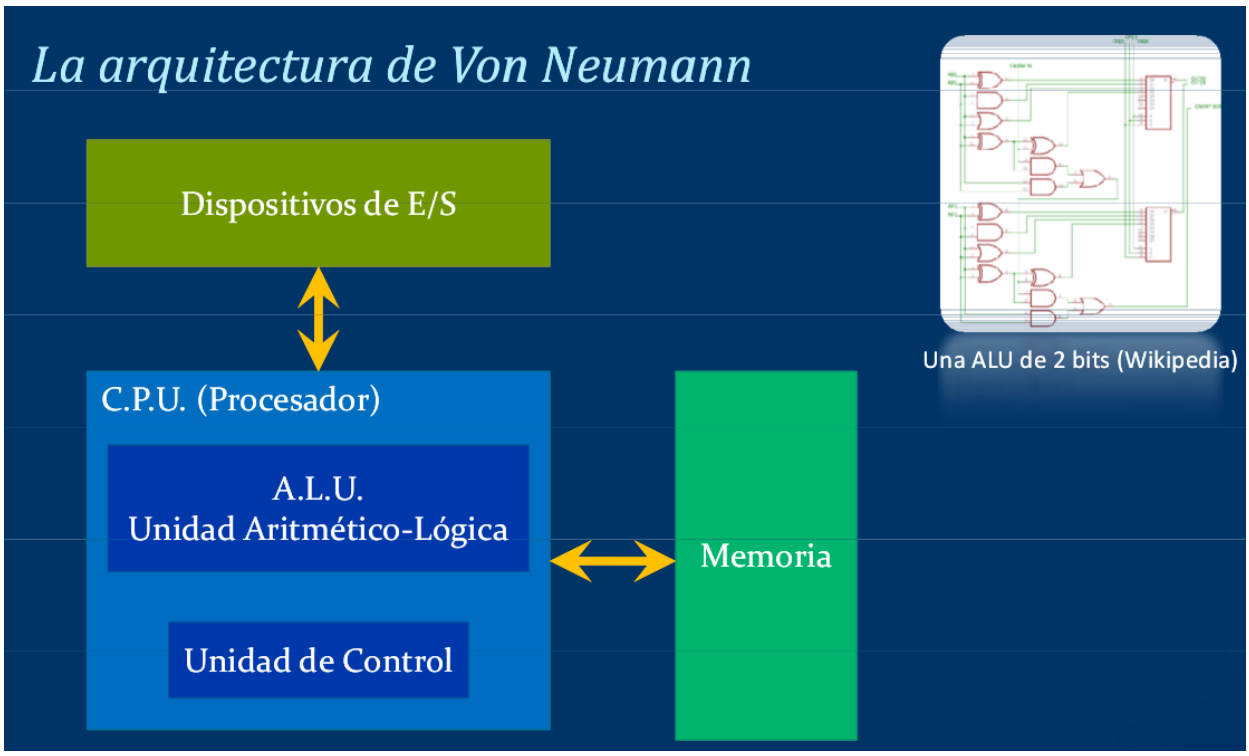
Ejecución de un programa en un ordenador.

Cuantos cómputos puede hacer un programa?

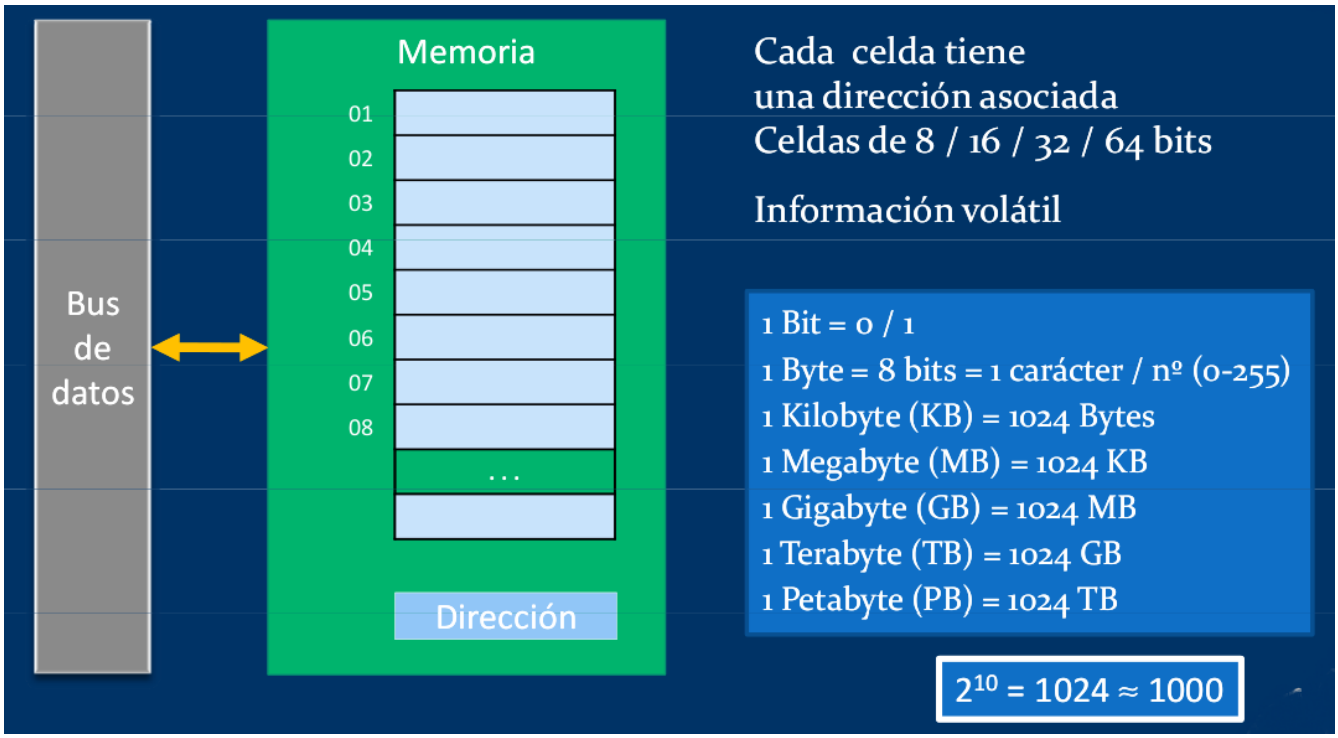
Arquitectura de una computadora



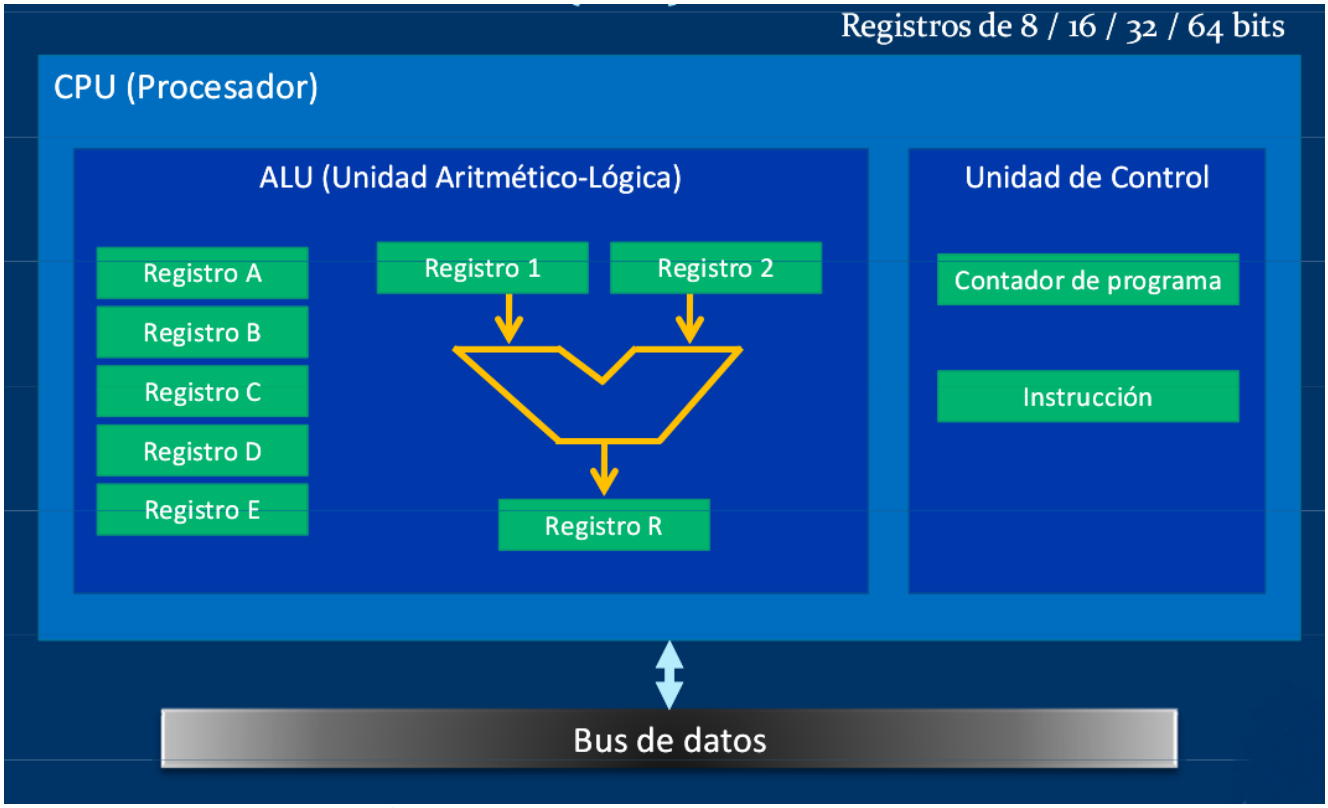
Arquitectura elemental



Memoria Principal



Unidad Central de Proceso (CPU)



Programas a bajo nivel: el lenguaje *máquina*

• La CPU trabaja en sistema binario, con ceros y unos (bits, unidad mínima de información/dato)
Byte: grupo de 8 bits (unidad básica de acceso a memoria)

En el lenguaje máquina TODO se representa en binario: instrucciones del programa, datos, direcciones de memoria, etc.

- Es habitual utilizar representación en sistema hexadecimal
  Así, el byte 01011011 se representa como:

$$\underbrace{0101}_{5}\underbrace{1011}_{B}$$

01011011 binario $\equiv 5B$ hexadecimal

Lenguaje máquina

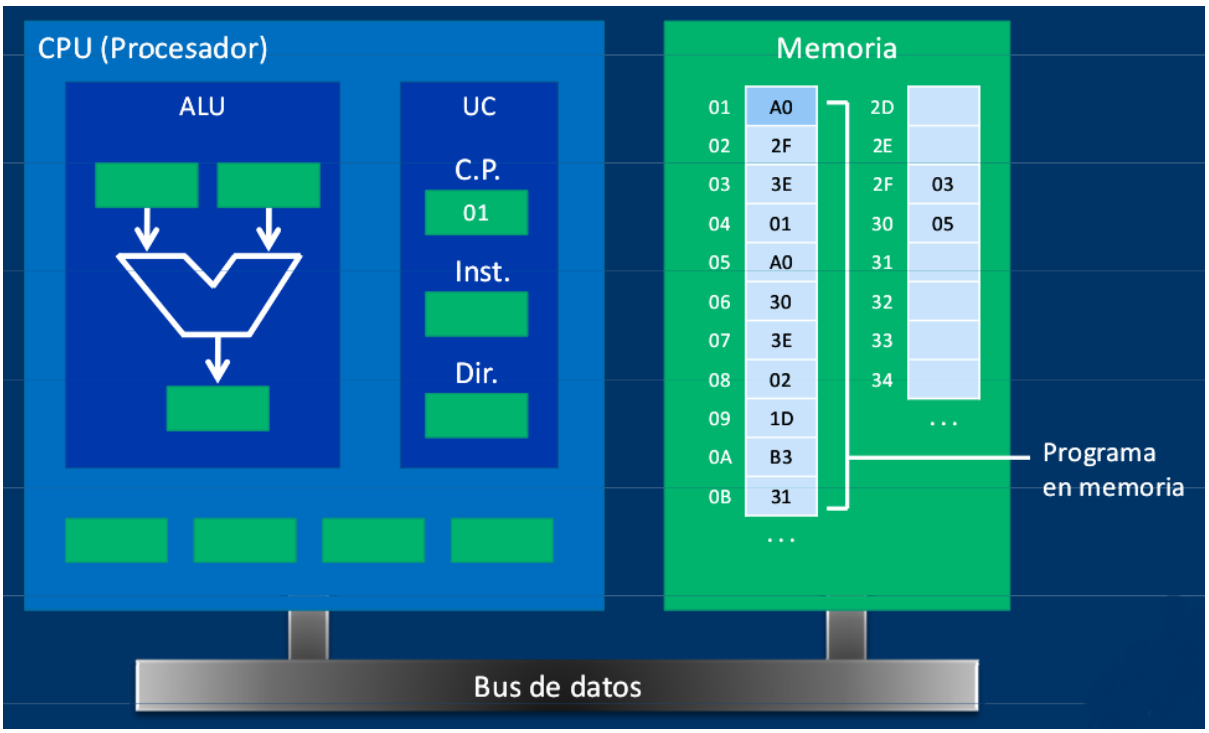
Un ejemplo de programa en lenguaje máquina para sumar dos números

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 (registros 1 y 2)
B3 31	Guardar el resultado en la posición de memoria 31

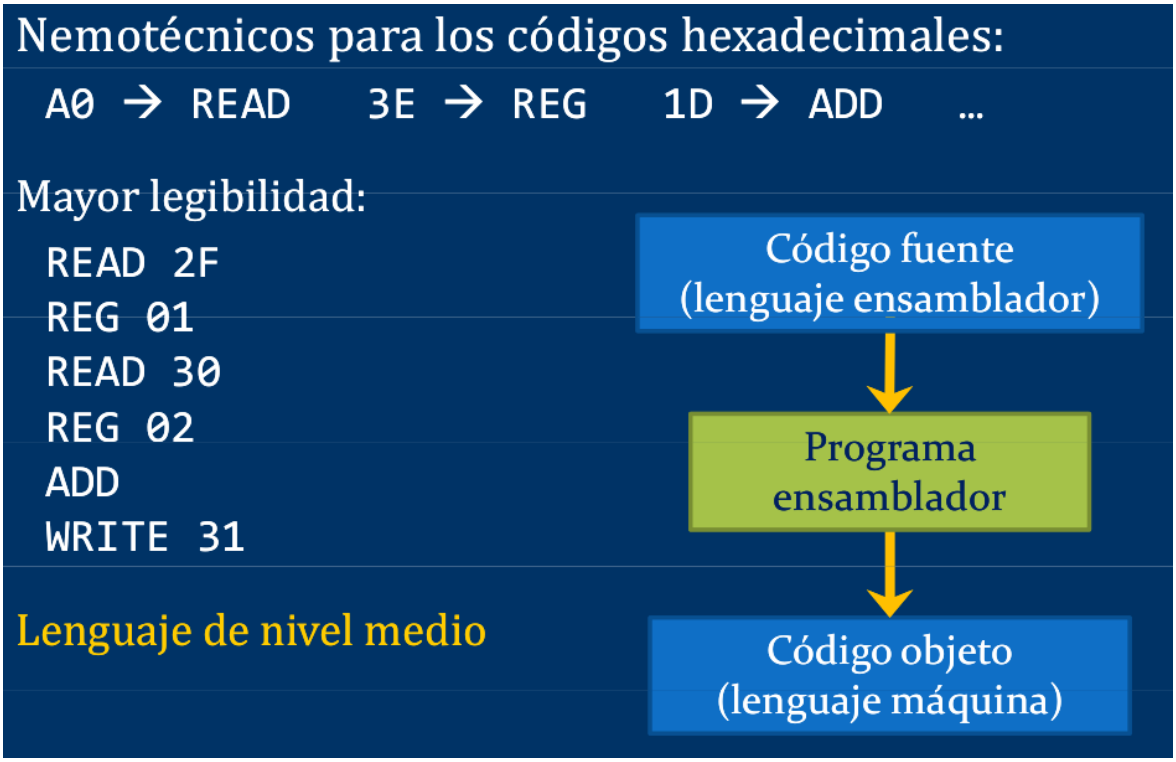
Este lenguaje es dependiente de la computadora concreta (de la CPU de la misma). Es distinto para un PC o un MAC... incluso es (ligeramente) distinto para cada CPU de esas arquitecturas.

Muy difícil programar directamente en lenguaje máquina.

Lenguaje máquina (II)



Lenguaje ensamblador. Un poco más fácil



Pero sigue siendo un lenguaje de bajo nivel, difícil de utilizar.

El siguiente paso: los lenguajes de alto nivel

- Más próximos al lenguaje natural y matemático

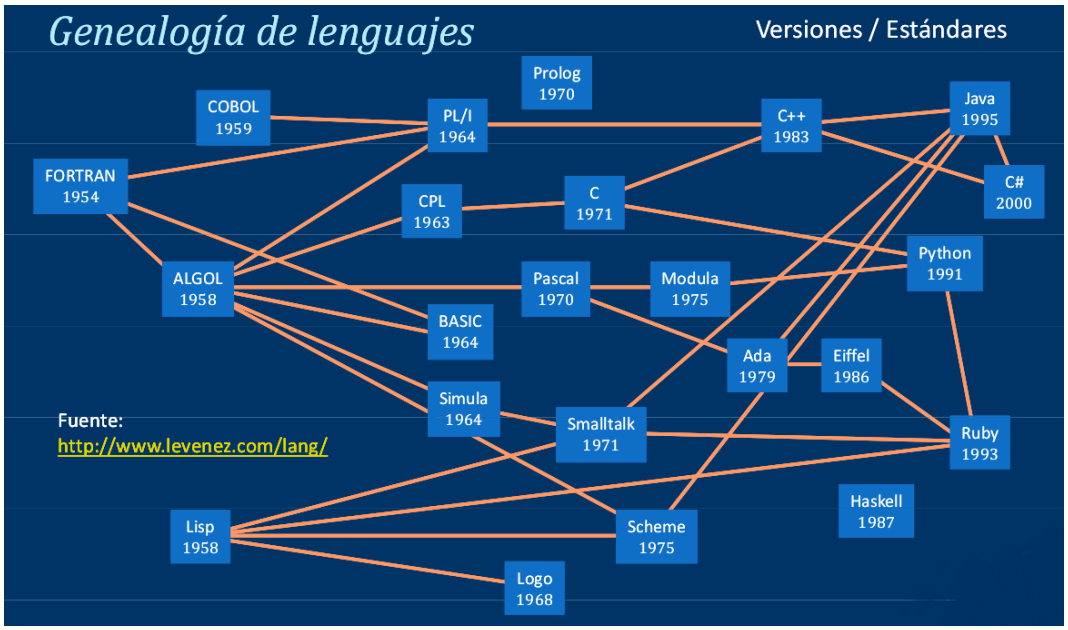
$$resultado = dato1 + dato2;$$

- Programas mucho más fáciles de escribir (y de leer)
- Después vino la programación estructurada, la abstracción procedimental, la estructuración de datos, la programación orientada a objetos...

  $\rightsquigarrow$ capas de abstracción para acercar el lenguaje a la persona y facilitar el diseño de programas.

- Infinidad de lenguajes: C++, Java, Python, Fortran, Prolog, Haskell, Pascal, Cobol, Lisp, Smalltalk, C#,...

Muchos lenguajes de programación



En `https://es.wikipedia.org/wiki/Anexo:Lenguajes_de_programacion` hay una lista muy incompleta con más de 600 lenguajes de programación.

¿Por qué C#?

- Lenguaje moderno (2000)
- Influido por lenguajes consolidados Java, C++, Eiffel, Modula-3, Pascal... incorporando lo mejor de ellos.
- Elecciones de diseño muy acertadas, diseñado por programadores experimentados como Anders Hejlsberg

  es elegante, simple, con tipos seguros...
- Completamente integrado en la plataforma .NET de MS Windows, pero multiplataforma
- Descriptores (Wikipedia) Multiparadigma: estructurado, imperativo, orientado a objetos, dirigido por eventos, funcional, genérico, reflexivo

Por qué se llama C#? # (sostenido musical, un semitono por arriba: superior a C). Es *otra* evolución de *C*.

¿Por qué C# en el grado de videojuegos?

- Porque se integra perfectamente con el motor Unity
- Es un lenguaje de propósito general... no sirve solo para videojuegos, sino para cualquier tipo de aplicación.
- Es un lenguaje con futuro y es perfecto para el currículum de un alumno de este grado.
- Es un buen lenguaje de iniciación a la programación... pero además es un lenguaje profesional!

Por otro lado:

- El objetivo prioritario del curso NO es aprender C#... es aprender programación estructurada: C# es muy razonable.
- Un programador debe conocer distintos (y variados) lenguajes de programación y aprender constantemente otros nuevos.
- ¿Cuál es el mejor lenguaje de programación?

  Cada programador tiene su(s) favorito(s). Lo ideal es dominar muchos y utilizar uno u otro dependiendo de la aplicación.

Sintaxis y semántica de los lenguajes

Un lenguaje (formal) queda definido (formalmente) por dos aspectos esenciales:

- Sintaxis: reglas que determinan las construcciones válidas del lenguaje. ¿Está bien escrito?
- Semántica: significado que se atribuye a las construcciones válidas del lenguaje. ¿Qué hace?

Un lenguaje de programación es un lenguaje formal.

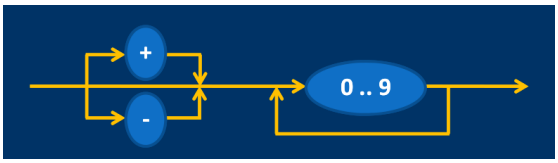
- La sintaxis determina lo que es un *programa* válido.
- La semántica determina lo que hace un programa válido (el resultado que produce su ejecución).

En los sucesivo nos centramos en lenguajes de programación.

Sintaxis de los lenguajes de programación

Se define mediante Diagramas de flujo o Gramáticas

• Diagramas de flujo: representación gráfica de la forma de construcción de elementos del lenguaje. Por ejemplo, para los números enteros:



De acuerdo con esto, ¿están bien formados los siguientes números?

$$23, -159, +5, 1 - 34, 3.14, 002$$

Sintaxis de los lenguajes de programación. Gramáticas

La forma más habitual de definir la sintaxis es con

• Gramáticas BNF (Bakus-Naur Form) o EBNF (Extended Bakus-Naur Form): más formal, más potente, más precisa. Por ejemplo, para los enteros:

entero	→	signoOp secDigitos
signo	→	*+* | *-* | ϵ
secDigitos	→	digito | digito secDigitos
digito	→	*0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9*

- Cada línea es una *producción gramatical*
- Los símbolos no terminales son los que tienen una producción asociada o más (los que aparecen a la izquierda de →)
- Los símbolos terminales son el resto.
- | significa *alternativa* (ó)
- ϵ significa *secuencia vacía*

Gramáticas. Reconocimiento, generación.

$$
\begin{array}{lcl}
\text{entero} & \rightarrow & \text{signoOp secDigitos} \\
\text{signoOp} & \rightarrow & + \mid - \mid \epsilon \\
\text{secDigitos} & \rightarrow & \text{digito} \mid \text{digito secDigitos} \\
\text{digito} & \rightarrow & 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{array}
$$

De acuerdo con esto, ¿están bien formados los números 23, -159, +5, 1-34, 3.14, 002?

Por ejemplo, para el 23:

$$
\begin{array}{lcl}
\text{entero} & \rightarrow & \underline{\text{signoOp}}\ \text{secDigitos} \\
 & \rightarrow & \epsilon\ \text{secDigitos} \equiv \underline{\text{secDigitos}} \\
 & \rightarrow & \underline{\text{digito}}\ \text{secDigitos} \\
 & \rightarrow & 2\ \underline{\text{secDigitos}} \\
 & \rightarrow & 2\ \underline{\text{digito}} \\
 & \rightarrow & 2\ 3
\end{array}
$$