

SOC

Servei d'Ocupació
de Catalunya



Generalitat
de Catalunya



Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur



MÓDULO 1. MF0951_2

INTEGRAR COMPONENTES SOFTWARE EN PÁGINAS WEB

UNIDAD FORMATIVA 1.

UF1305 INTEGRAR COMPONENTES SOFTWARE EN PÁGINAS WEB.



Formació



Formació
online



Gestió de
Bonificacions



Desenvolupament
de Persones



Selecció



Tecnologies de
la Informació



Millora
Contínua



Servei de
Prevenció Aliè





1 Metodología de la programación

1.1 Introducción

- _Descripción
- _Características
- _Librerías de Javascript
- _¿Qué es un IDE en programación?
- _Instalación IDE VSC
- _Extensiones IDE VSC HTML

1.2 Lógica de programación.

- _Descripción y utilización de operaciones lógicas.
- _Secuencias y partes de un programa.



1.3

Ordinogrames/Diagrama de flujo

- _Descripción de un ordinograma.
- _Elementos de un ordinograma.
- _Operaciones en un programa.
- _Implementación de elementos y operaciones en un ordinograma.

1.4

Pseudocódigos.

- _Descripción de pseudocódigo.
- _Creación del pseudocódigo.



1.5

Objetos.

- _Descripción de objetos.
- _Funciones de los objetos.
- _Comportamientos de los objetos.
- _Atributos de los objetos.
- _Creación de objetos.

1.6

Ejemplos de códigos en diferentes lenguajes.

- _Códigos en lenguajes estructurales.
- _Códigos en lenguajes scripts.
- _Códigos en lenguajes orientados a objetos.

Examen**Bonus track**

"Hay sólo dos clases de lenguajes de programación: aquellos de los que la gente está siempre quejándose y aquellos que nadie usa"

-- Bjarne Stroustrup



Bjarne Stroustrup, es un científico de la computación y catedrático de Ciencias de la Computación en la Universidad A&M de Texas. Es reconocido principalmente por el desarrollo del lenguaje de programación C++.

Metodología de la programación

1.

1.1.

Introducción JS.

JavaScript es un lenguaje de programación **interpretado**, dialecto del estándar **ECMAScript**. Se define como **orientado a objetos**, basado en **prototipos**, **imperativo**, débilmente **tipado** y **dinámico**.

- ✓ Diseñado por: Netscape Communications, **Fundación Mozilla**
- ✓ Última versión estable: ECMAScript 2021 (01 de junio de 2021)
- ✓ Sistema de tipos: Débil, dinámico
- ✓ Apareció en: 4 de diciembre de 1995
- ✓ Dialectos: ECMAScript



«Javascript es la tercera pieza fundamental del desarrollo web»

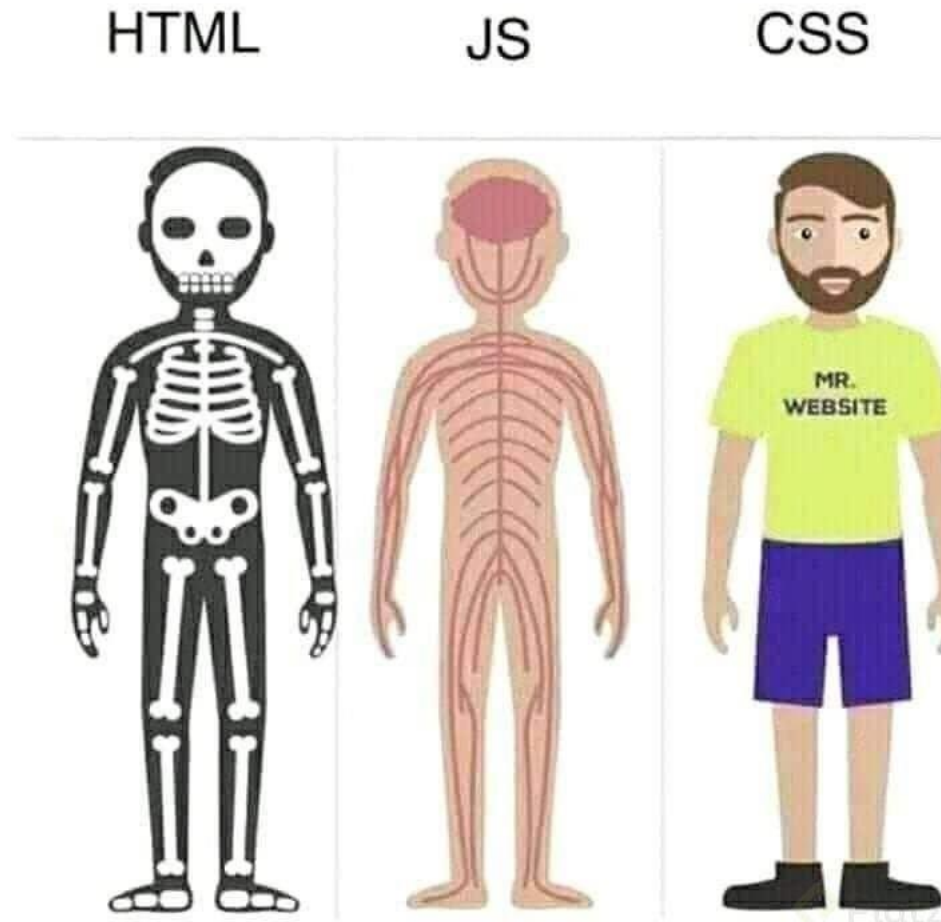
ECMAScript es una especificación de lenguaje de programación publicada por ECMA International.

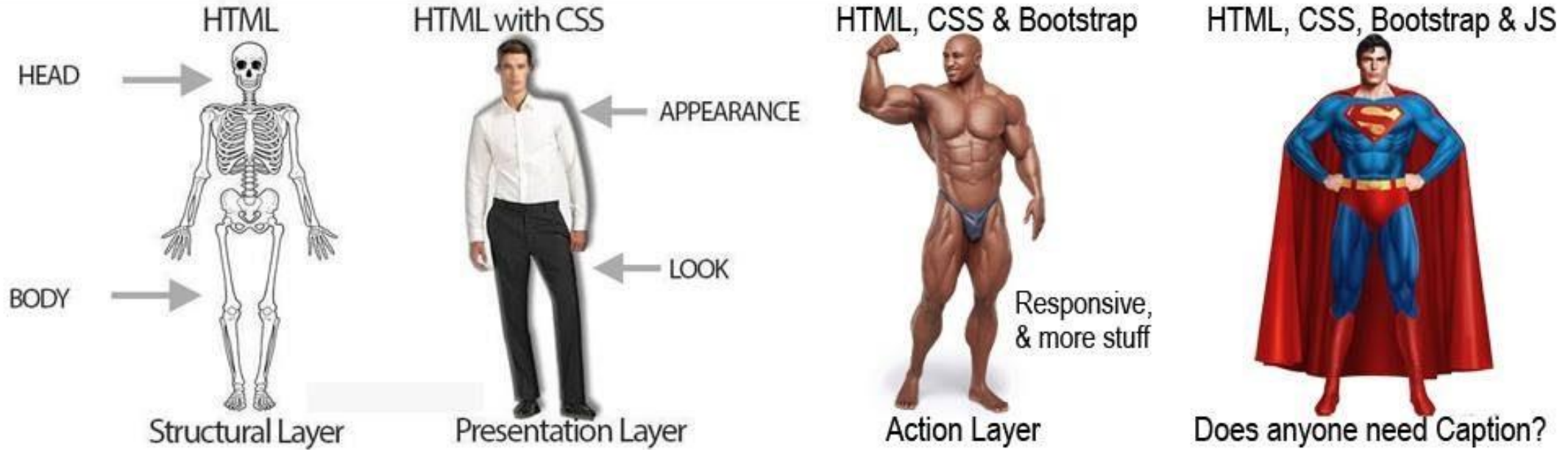
Prototípico de la herencia es una forma de objeto-orientado a la reutilización de código. javascript es uno de los pocos [principales] lenguajes orientados a objetos para el uso prototípico de la herencia.



¿Cuáles son las otras dos?







1.1. Introducció_característiques

Lenguaje del lado del cliente: Se ejecuta en la máquina del propio cliente a través de un navegador.

Lenguaje orientado a objetos: utiliza clases y objetos como estructuras que permiten organizarse de forma simple y son reutilizables durante todo el desarrollo. Otros lenguajes orientados a objetos son Java, Python o C++.

De tipado débil o no tipado: No es necesario especificar el tipo de dato al declarar una variable. Lenguajes de tipado fuerte son C++ o Java.

De alto nivel: Su sintaxis es fácilmente comprensible por su similitud al lenguaje de las personas. Se le llama de “alto nivel” porque **su sintaxis se encuentra alejada del nivel máquina**, es decir, del código que procesa una computadora para ejecutar lo que nosotros programamos.

Un lenguaje de alto nivel como Javascript permite que su **barrera de entrada y su curva de aprendizaje se acorte drásticamente**.

¿ **Cómo sería un condicional? Si ...**



Lenguaje interpretado: permite convertir las líneas de código en el lenguaje de la máquina. Esto tiene un gran número de ventajas como la **reducción del procesamiento** en servidores web al **ejecutarse directamente en el navegador del usuario**, o que es apto para **múltiples plataformas** permitiendo usar el mismo código. C#, Ruby, Java o Python

Según un [estudio de requisitos solicitados](#) en las ofertas de empleo en el año 2022 realizado por la universidad de Boston Northeastern, Javascript es el **segundo lenguaje más demandado** sólo por detrás de Python.



TOP 10 Popular Programming Languages in 2020	
1	Python
2	JavaScript
3	Java
4	C#
5	C
6	C++
7	GO
8	R
9	Swift
10	PHP

WWW.NORTHEASTERN.EDU/GRADUATE



1.1. Introducció_ Librerías de Javascript

jQuery



La librería jQuery es una de las librerías **más conocidas** para programar en javascript, y cuenta con una gran comunidad de usuarios y desarrolladores. Una de sus principales características es que se trata de una librería open source, es decir, de **código abierto**. La filosofía de jQuery se basa en realizar órdenes de codificación simples y escuetas, **programando en una o dos líneas lo que en javascript llevaría 20 líneas**.

¿Qué es un framework? Y cual conocéis?



1.1. Introducció_ Librerías de Javascript

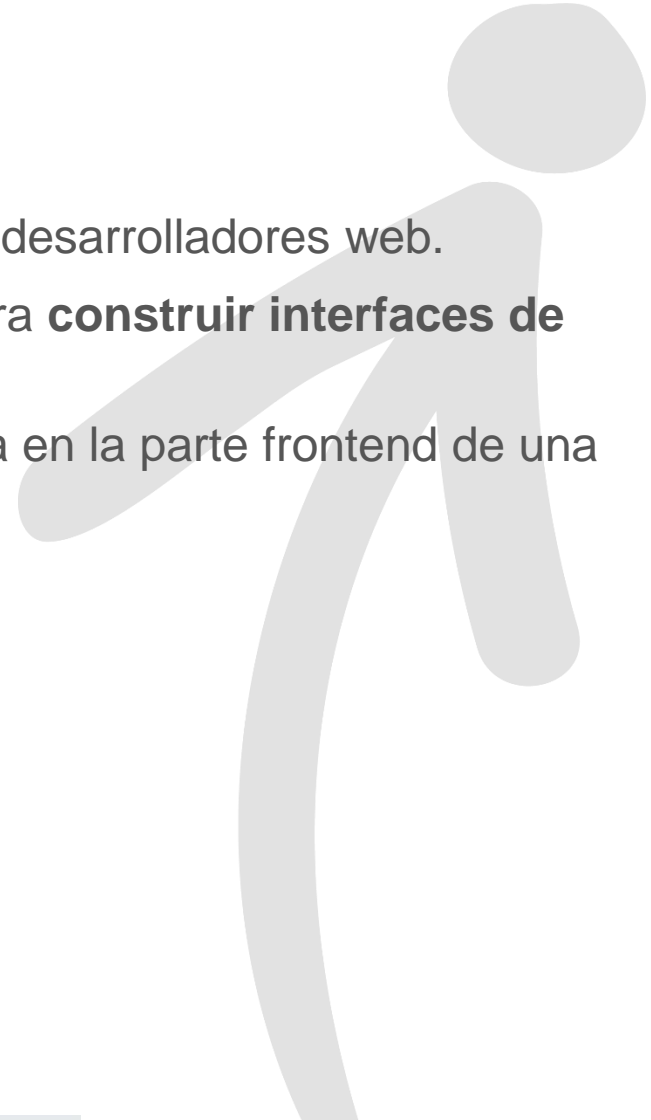
React



Junto con jQuery, React JS es otra librería clave de Javascript para los desarrolladores web.

React fue creada por **Facebook en 2011** y planeada explícitamente para **construir interfaces de usuario dinámicas, rápidas e interactivas**.

Al igual que jQuery, también es una librería de **código abierto** centrada en la parte frontend de una aplicación.



1.1 Introducció_ Librerías de Javascript

AngularJS

AngularJS es un framework desarrollado por **Google en 2009** y de **código abierto**.

Al igual que React, esta librería **se centra en el desarrollo frontend**.

AngularJS utiliza una versión de Javascript llamada **Typescript**.

Es una de las **librerías más utilizadas junto con React o Vue** por sus sencillas implementaciones y su multitud de herramientas, así como su integración y utilización del HTML evitando muchos quebraderos de cabeza a los desarrolladores.



Vue.js

Con total seguridad, Vue es el **framework de Javascript que más ha crecido** en popularidad.

Su flexibilidad y sencillez han convertido a Vue en la librería preferida de muchos desarrolladores amateur y profesionales al enfrentarse a un proyecto de desarrollo web.

Al igual que las librerías anteriores, Vue es de **código abierto** y se creó en 2014 por el mismo que desarrolló el framework AngularJS en **Google** unos años antes.



Node js

Node.js es la librería **opensource** más utilizada para el desarrollo **backend** con millones de desarrolladores en todo el mundo.

Este framework surgió como respuesta a la necesidad de **ejecutar aplicaciones con javascript no sólo en un navegador si no también en una máquina.**



1.1 Introducció_¿Qué es un IDE en programación?

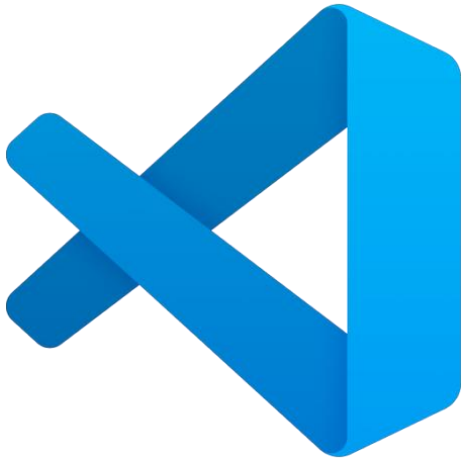
IDE es el acrónimo del término inglés *Integrated Development Environment*
Entorno de Desarrollo Integrado.

Es el escenario digital utilizado en programación.

Editor de código.

Compilador.

Depurador o *debugger*.




Atom
Notepad++
Sublime Text

Visual Studio Code.

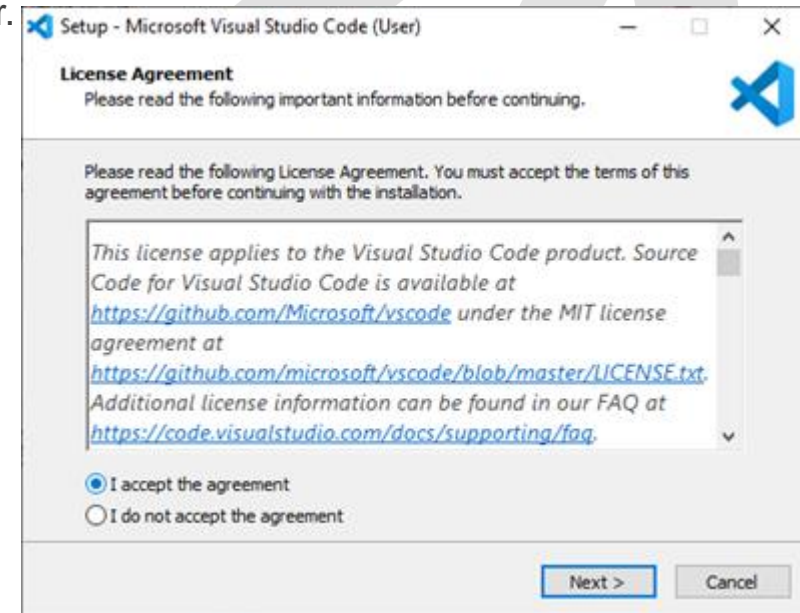
Eclipse
NetBeans
Xcode



- ✓ **Paso 1:** Ve a la pàgina de Microsoft Visual Studio Code en Academic Software y haz clic en el botón 'Descargar Visual Studio Code' para descargar el archivo de instalación.
- ✓ **Paso 2:** Abre el archivo de instalación .exe en tu carpeta de descargas para iniciar la instalación.

 VSCodeUserSetup-x64-1.63.2.exe

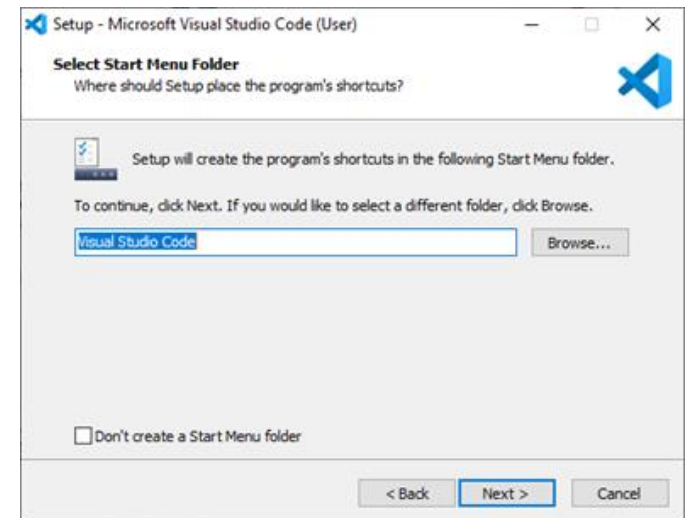
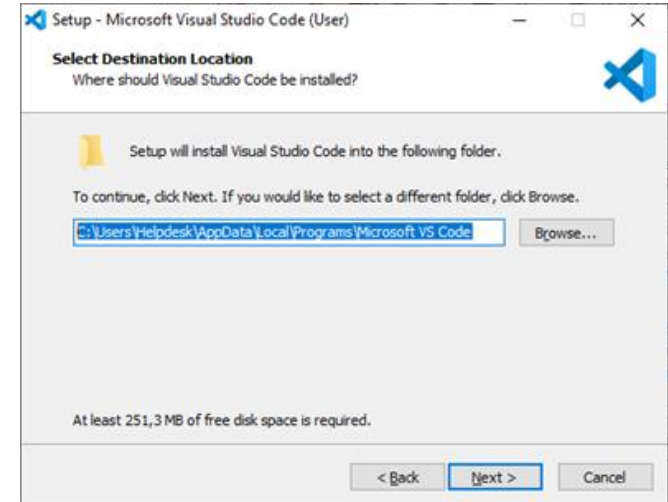
Paso 3: Lee y acepta el acuerdo de licencia. Haz clic en Next para continuar.



0. Introducció instal·lació IDE VSC

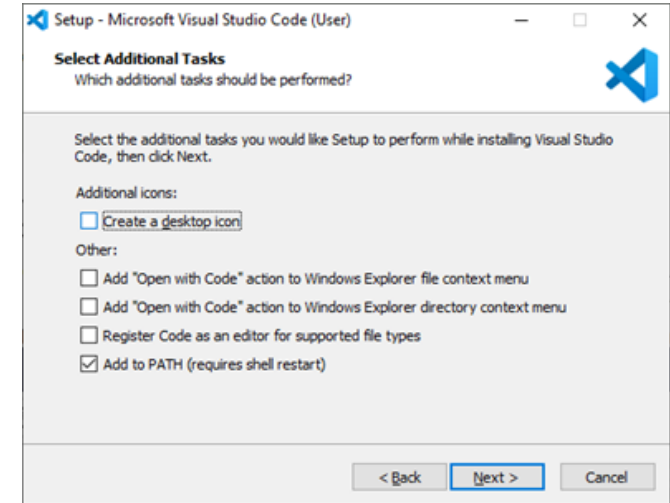
- ✓ **Paso 4:** Puedes cambiar la ubicación de la carpeta de instalación o mantener la configuración predeterminada. Haz clic en Next para continuar.

Paso 5: Elige si deseas cambiar el nombre de la carpeta de accesos directos en el menú Inicio o si no deseas instalar accesos directos en absoluto. Haz clic en Next.

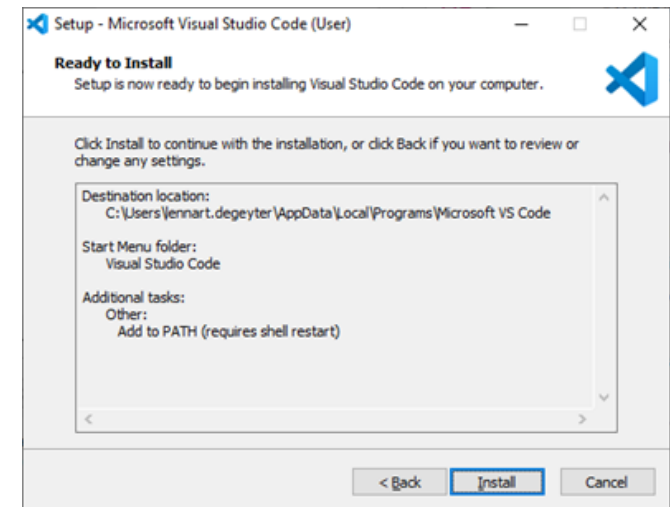


1. Introducció instal·lació IDE VSC

- ✓ **Paso 6:** Selecciona las tareas adicionales, por ej. crear un icono en el escritorio o añadir opciones al menú contextual de Windows Explorer. Haz clic en Next.

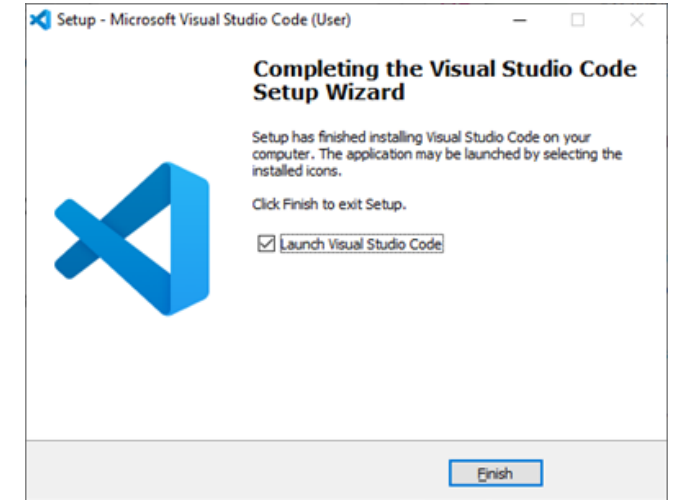
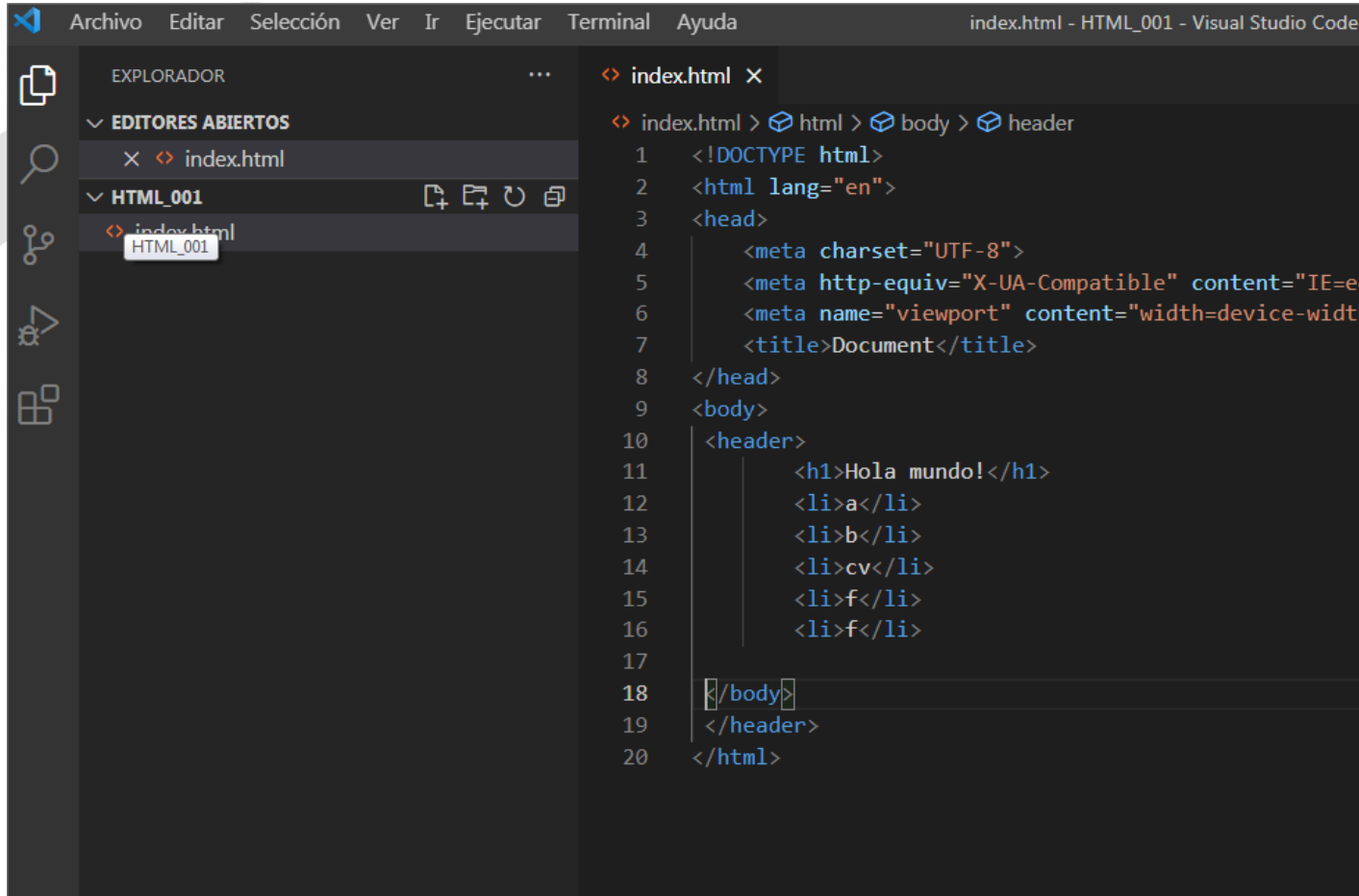


Paso 7: Haz clic en Install para iniciar la instalación.



1.1. Introducció instal·lació IDE VSC

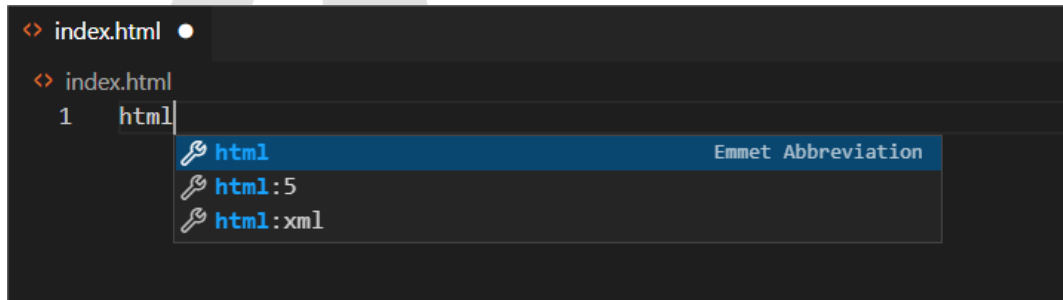
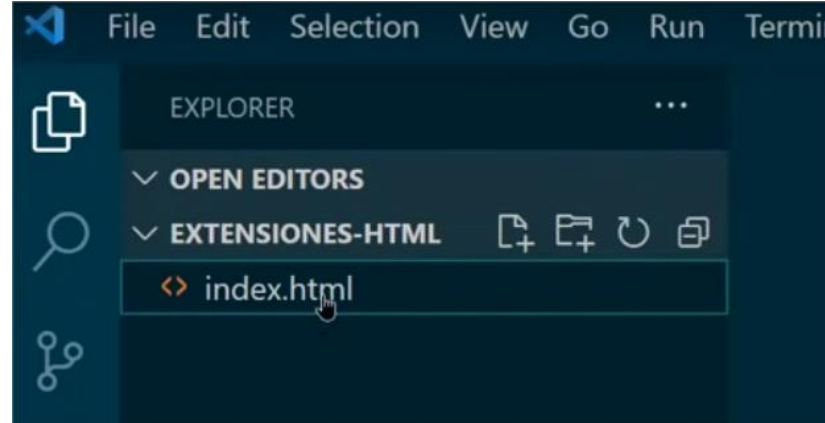
- ✓ **Paso 8:** El programa està instal·lat i està llest per usar. Fes clic en Finish per finalitzar la instal·lació i llançar el programa.



1.1. Introducció_extensioes VSC

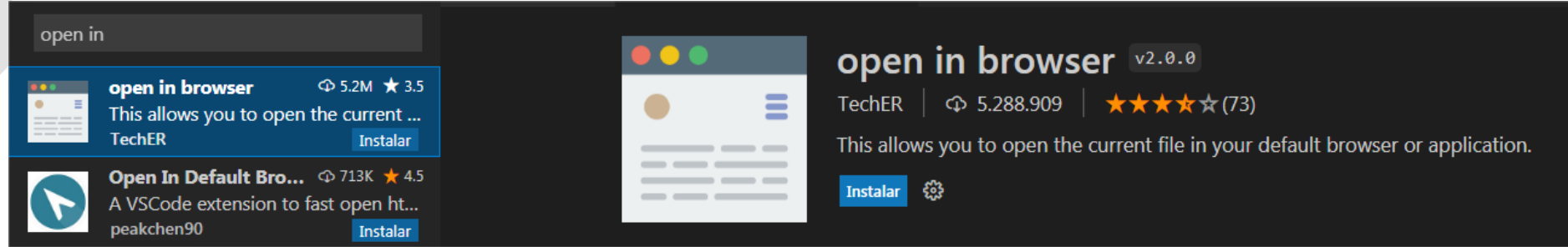
✓ Extensions HTML

- Creamos carpeta
- Creamos archivo index.html
- Creamos estructura html:5

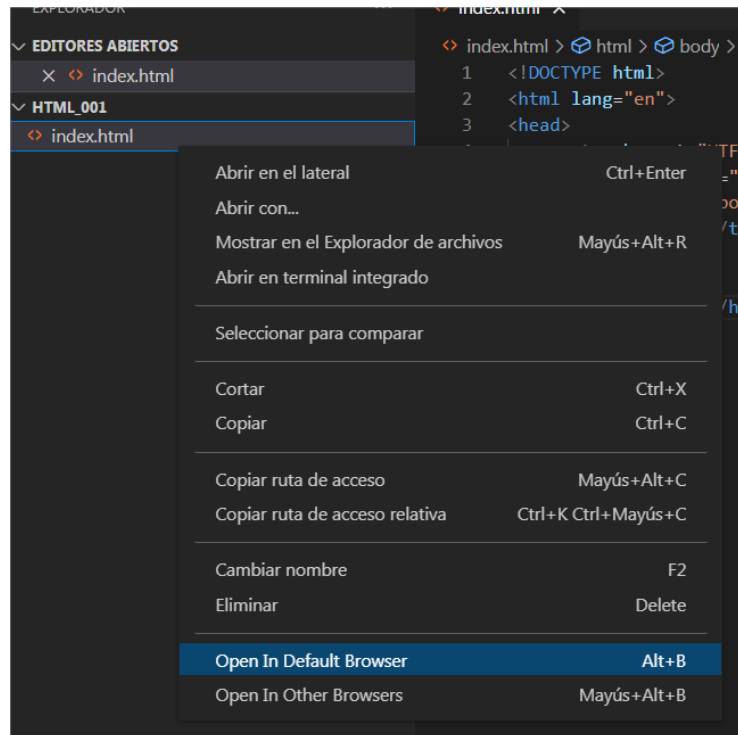


1.1. Introducció_extensióes VSC HTML

✓ Open in browser

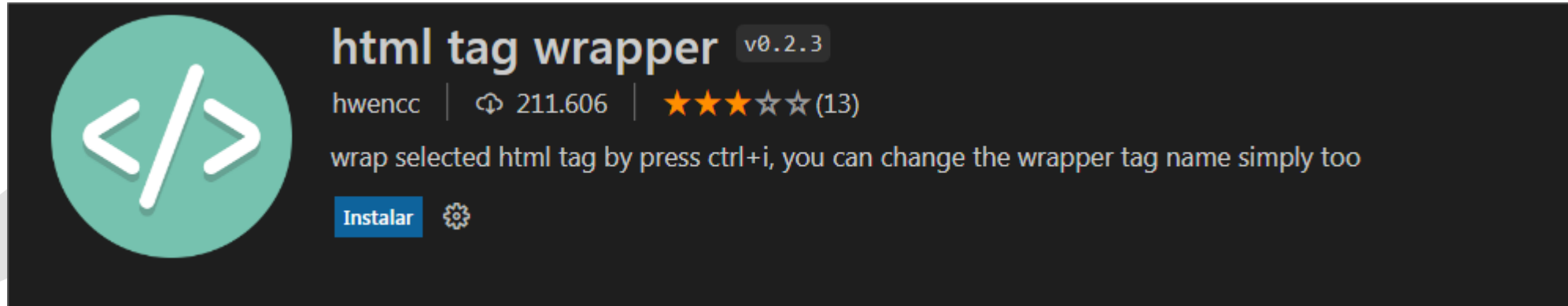


- Alt + B



1.1. Introducció_extensióes VSC HTML

✓ HTML TAG WRAPPER



- CTRL + i

```
</head>
<body>
  <h1>Hola mundo!</h1>
  li*5
</body>
</html>
```

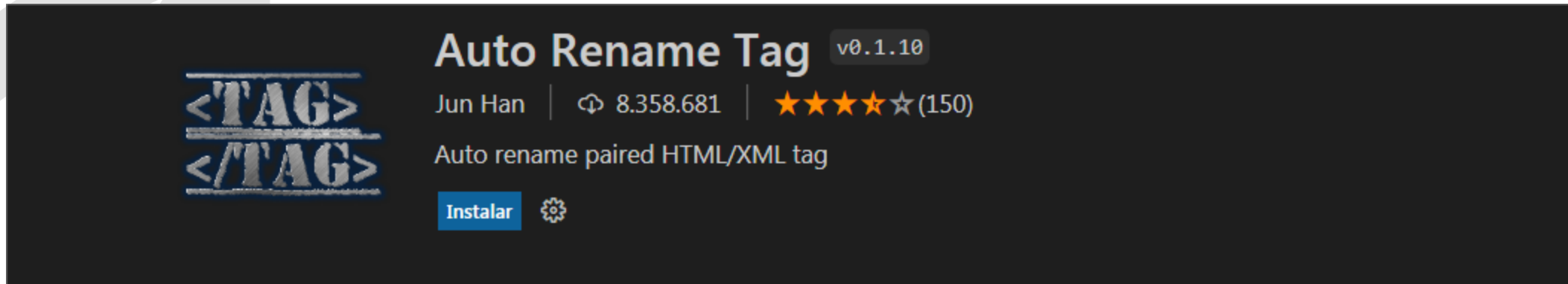
```
<body>
  <h1>Hola mundo!</h1>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
</body>
</html>
```

```
<body>
  <h1>Hola mundo!</h1>
  <div>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
  </div>
</body>
```



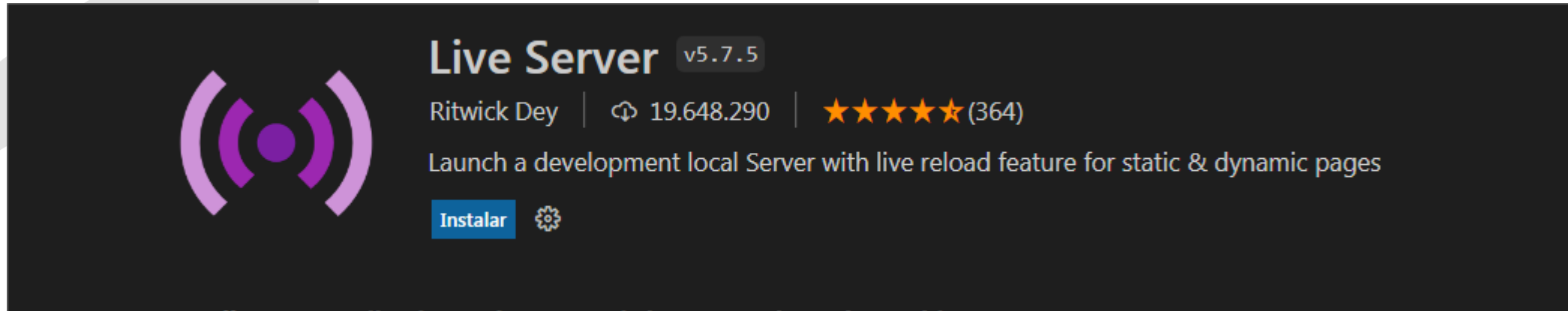
1.1. Introducció_extensióes VSC HTML

- ✓ AUTO RENAME TAG
- ✓ Cambiar las dos etiquetas a la vez

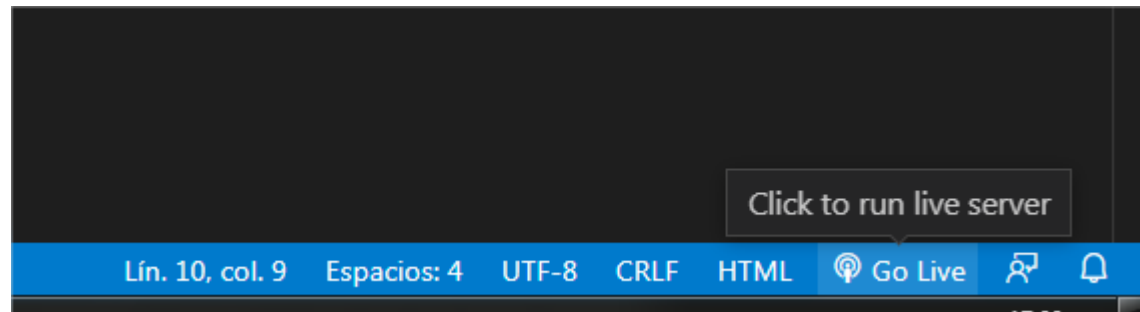
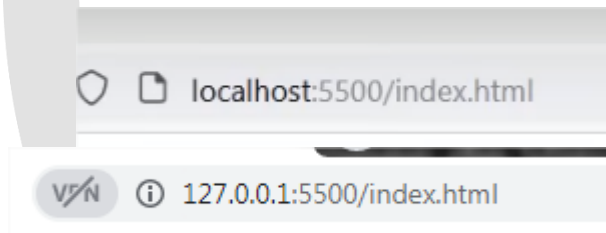


1.1. Introducció_extensiones VSC HTML


- ✓ LIVE SERVER
- ✓ Auto recargar el navegador/live reload



Arrancamos servidor local. Port 5500



1.1. Introducció_extensióes VSC JS




JavaScript Debugger (Nightly)

v2022.3.717 [Vista Previa](#)

Microsoft | 483.055 | ★★★★★ (5)

An extension for debugging Node.js programs and Chrome.

[Instalar](#) ⚙️



ESLint

v2.2.2

Microsoft | 19.110.220 | ★★★★★ (188)

Integrates ESLint JavaScript into VS Code.

[Instalar](#) ▼ ⚙️

★ Esta extensión se recomienda en función de los archivos abiertos recientemente.



1.1. Introducció_extensióes VSC JS



Better Comments v2.1.0

Aaron Bond | 2.820.070 | ★★★★★ (127)

Improve your code commenting by annotating with alert, informational, TODOs, and more!

Deshabilitar



Desinstalar



JavaScript (ES6) code snippets v1.8.0

charalampos karypidis | 7.757.032 | ★★★★★ (33)

Code snippets for JavaScript in ES6 syntax

Instalar



Import Cost v2.15.0

Wix | 1.592.353 | ★★★★★☆ (44)

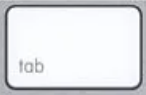
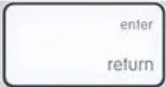


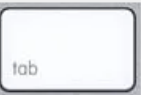
Display import/require package size in the editor

Instalar



1.1. Introducció Atajos HTML VSC JS

- Crear estructura
 - Carpeta /javascript
 - Carpeta /hola-mundo
 - Crear index.html (! +[TAB])
 - p+[tab]
 - h2+[tab]
 - img+[tab]
 - lorem+[tab]
 - lorem500+[tab]

TECLA	RESULTADO
p + 	<p></p>
p + 	<p></p>
h2 + 	<h2></h2>
img + 	
! + 	<pre> <!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Document</title> </head> <body> </body> </html> </pre>



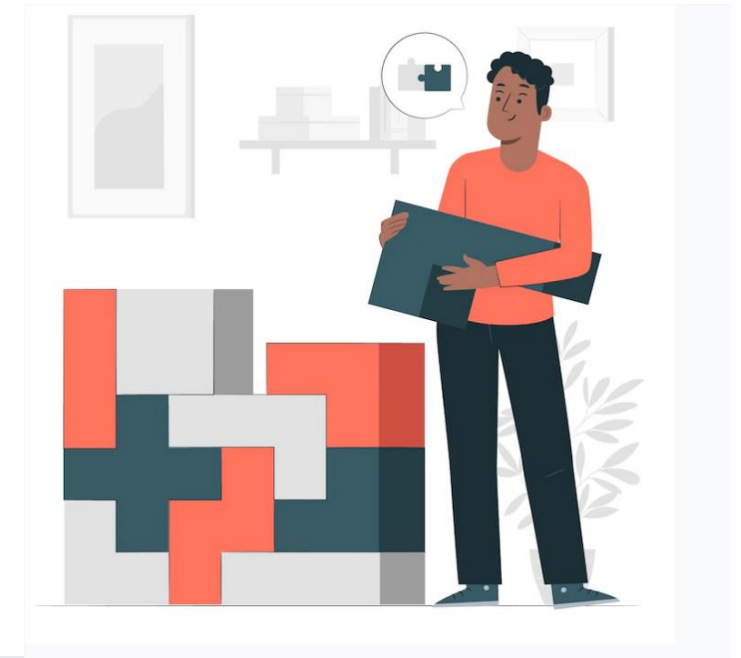
Lógica de programación

1. Lógica de programación

La metodología de programación es el enfoque teórico - práctico que **hace posible encontrar soluciones a problemas complejos** partiendo del análisis y apoyándose en la planificación.

Un **programa informático** o programa de computadora es una **secuencia de instrucciones**, escritas para realizar una tarea específica en un computador.

Un **programador** es aquella persona que **elabora programas** de computadora, es decir **escribe, depura y mantiene el código fuente** de un programa informático, que ejecuta el hardware de una computadora, para realizar una tarea determinada.



1.2. Lògica de programació

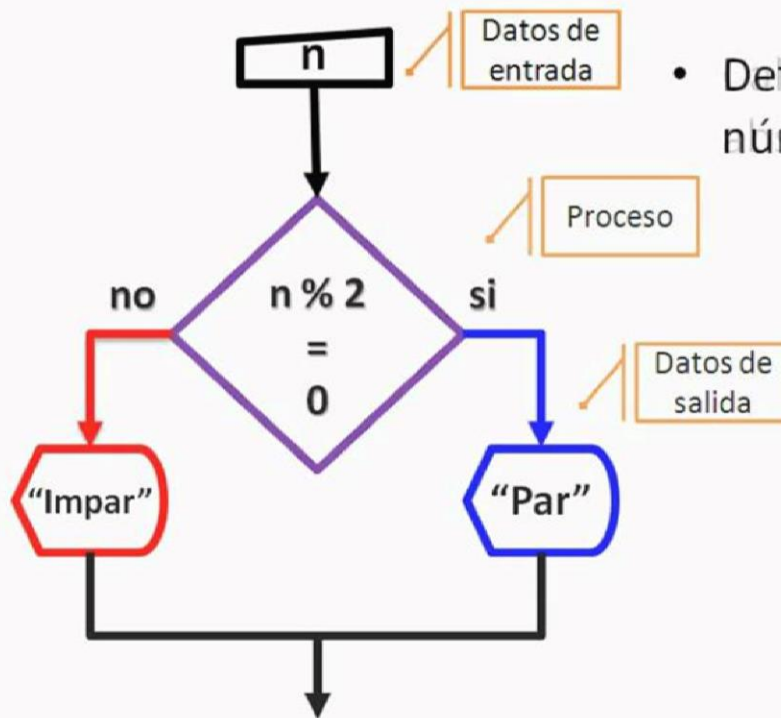
Visualiza la següent situació, tu necessites **hacer un pastel**:

1. Seleccionar los ingredientes de la receta;
2. Seleccionar recipiente;
3. Colocar harina, de acuerdo con la medida;
4. Seleccionar los huevos;
5. Colocar mantequilla y azúcar al gusto;
6. Colocar la leche;
7. Mezclar todos los ingredientes en el recipiente;
8. Despejar la masa en el molde;
9. Llevar al horno;
10. Esperar 40 minutos;
11. Retirar del horno;
12. Servir el pastel.



¿Quién sabe
cocinar?

Ejemplo

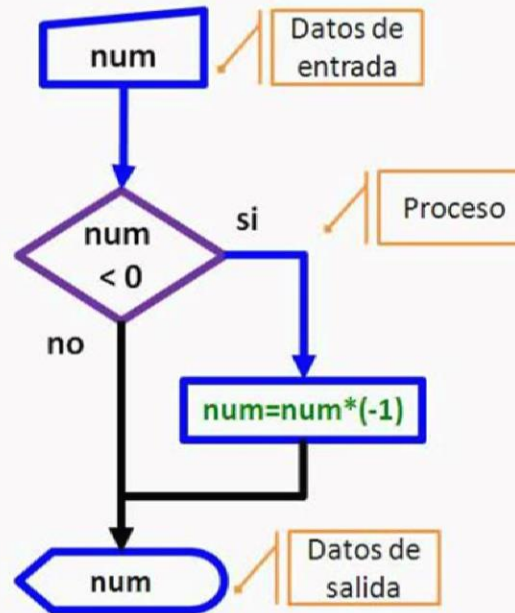


- Determinar si un número es par o impar

	entrada	salida
Prueba	num	Resultado
1	18	Par
2	17	Impar
3	14	Par
4	21	Impar
5	8	Par
6	7	Impar
7	23	Impar
8	0	Par
9	3	Impar
10	20	Par

Ejemplo

- Calcular el valor absoluto de un número.

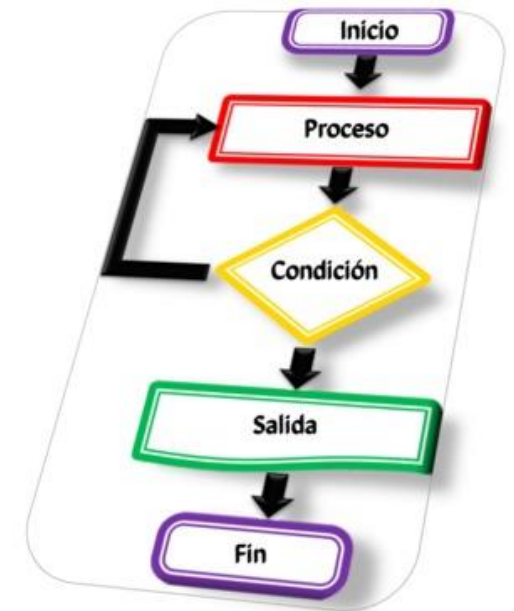


	entrada	salida	
Prueba	num	num	
1	5	5	
2	8	8	
3	3	3	
4	-8	8	←
5	9	9	
6	6	6	
7	-5	5	←
8	-3	3	←
9	5	5	
10	-10	10	←

El **valor absoluto de un número**, en otras palabras, es el **valor** que resulta de eliminar el signo correspondiente a este. Es decir, el **valor absoluto de un número** positivo es este mismo **número**

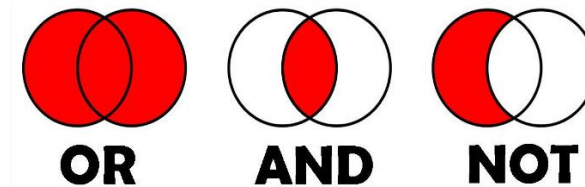
1.2 Lògica de programació

- Ordenadores = impulsos elèctrics = 1 bit (mínima quantitat d'informació / unitat bàsica) 1/0
- Circuits lògics d'un microprocessador (àlgebra booleana) cert/fals
- Javascript = Llenguatge alt nivell (comprensible per a l'humà)
- S'utilitza gramàtica i vocabulari.



La lògica proposicional està composta de:

- Proposició simple: Es una oració o judici que afirma, nega o describe el estado de las cosas.
- Proposició composta: Es la oración que empalma dos o más proposiciones simples, por medio de conectivos lógicos Y, O...



Ejemplo de **proposició simple**:

- Dylan tiene 3 años
- Deneb es la estrella más brillante

Ejemplo de **proposició composta**:

- La ballena vive en el mar **y** es un mamífero
- Yo iré a estudiar **o** iré a jugar fútbol

Tabla verdad AND

Entrada A	Entrada B	Salida $A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Tabla verdad OR

Entrada A	Entrada B	Salida $A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Tabla verdad XOR

Entrada A	Entrada B	Salida $A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

AND (conjunció)

a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

OR (disjunció)

a	b	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

XOR (disjunció exclusiva)

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

NOT (negació)

a	NOT a
0	1
1	0

$6 > 5$

AND

$8 < 7$

→ Falso

$6 > 5$

OR

$8 < 7$

→ Verdadero

$6 > 5$

XOR

$8 < 7$

→ Verdadero

NOT($6 > 5$) → Falso

NOT($8 < 7$)

→ Verdadero



1.2 Lógica de programación__Secuencias y partes de un programa.

Las **partes principales de un programa** están relacionadas con dos bloques: **declaraciones e instrucciones**.

En las **instrucciones** podemos diferenciar tres partes fundamentales

Entrada de Datos:

La constituyen todas las instrucciones que toman los datos de entrada desde un dispositivo externo y los almacena en la memoria principal para que puedan ser procesados.

Proceso o algoritmo: Está formado por las instrucciones que modifican los objetos a partir de su estado inicial (datos de entrada) hasta el estado final (resultados) dejando los objetos que lo contiene disponibles en la memoria principal.

Salida de resultados: Conjunto de instrucciones que toman los datos finales (resultado) de la memoria principal y los envían a los dispositivos externos.

1.2 Lògica de programació__Secuencias y partes de un programa.

El teorema de **Böhm y Jacopini** (1966) dice **que un programa** propio puede ser escrito utilizando sólo **tres tipos de estructuras de control**:

1. Estructura secuencial

Una estructura de programa es secuencial si las instrucciones se ejecutan una tras otra, a modo de secuencia lineal, es decir que una instrucción no se ejecuta hasta que finaliza la anterior, ni se bifurca el flujo del programa.

El teorema de **Böhm y Jacopini**

2. Estructura selectiva o de selecció IF

La estructura selectiva permite que la **ejecución del programa se bifurque a una instrucción** (o conjunto) u otra/s, según un criterio o condición lógica establecida, sólo uno de los caminos en la bifurcación será el tomado para ejecutarse.

- Estructura de control selectiva simple
- Estructura de control selectiva doble
- Estructura de control selectiva multiple

3. Estructura de control cíclica o repetitiva

- Estructura de control desde **FOR**
- Estructura de control mientras **WHILE**

Operadores de relación	
Mayor a	>
Menor a	<
Mayor o igual a	>=
Menor o igual a	<=
Diferente a	~=

Operadores lógicos	
Y (and)	&
O (or)	
No (not)	~
Exclusivo o (or)	xor

para esta estructuras son necesarios los operadores logicos y de relación.

El **proceso de programación** es, por consiguiente, un **proceso de solución de problemas** y el **desarrollo de un programa** requiere las siguientes fases:

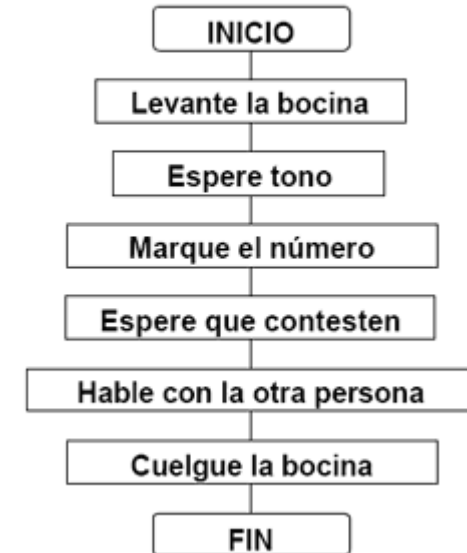
- 1.- Definición y análisis del problema;
- 2.- Diseño de **algoritmos**:
Diagrama de flujo
Pseudocódigo
- 3.- Codificación del programa;
- 4.- Depuración y verificación del programa;
- 5.- Documentación;
- 6.- Mantenimiento.

Pseudocódigo:

```

INICIO
  Levante la bocina
  Espere tono
  Marque el número
  Espere que contesten
  Hable con la otra persona
  Cuelgue la bocina
FIN
  
```

Diagrama de flujos:



1.2.

Algoritmo



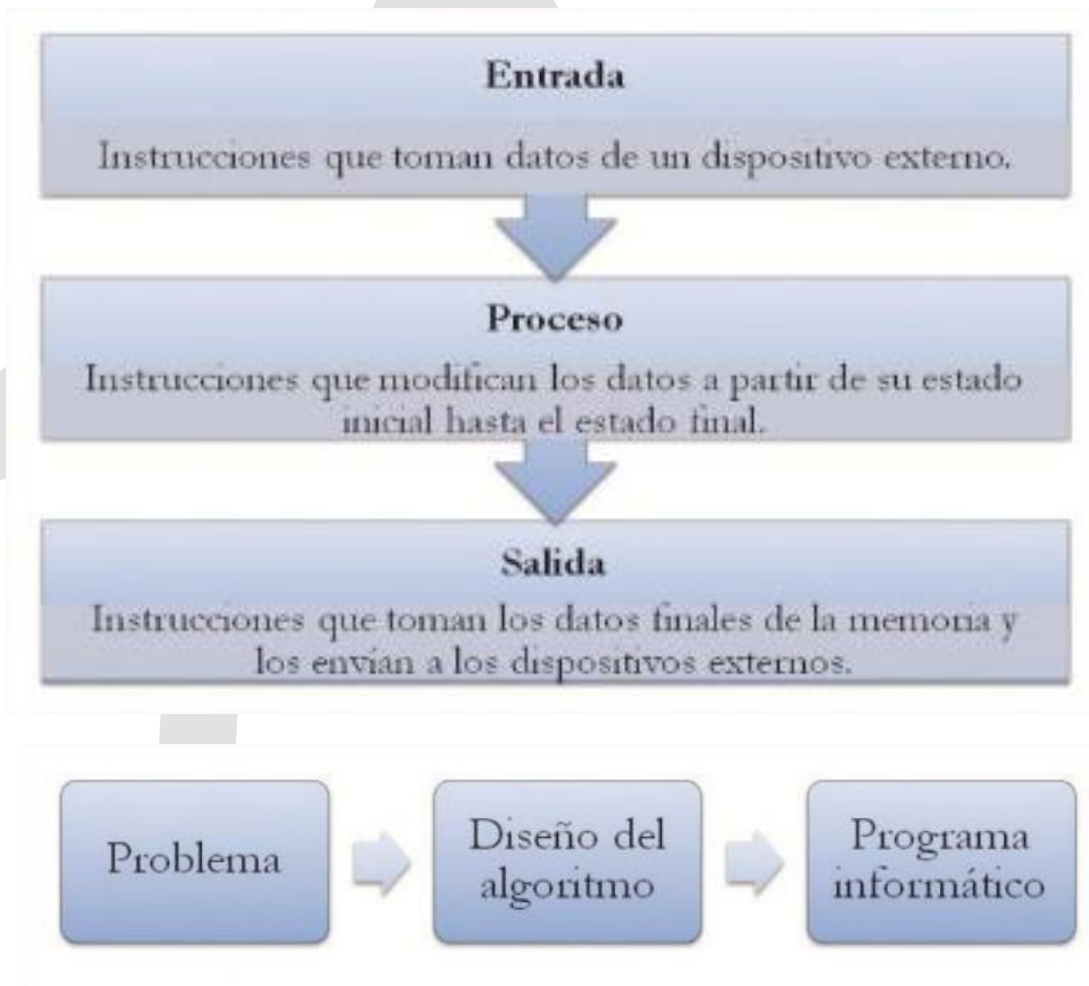
1.2 Lógica de programación__Secuencias y partes de un programa.

Un **algoritmo** informático. Conjunto de instrucciones definidas, ordenadas y acotadas para resolver un problema, realizar un cálculo o desarrollar una tarea.

En programación, un algoritmo supone el **paso previo a ponerse a escribir el código**.

Las **tres partes de un algoritmo** son:

- **Entrada de Datos:** La constituyen todas las instrucciones que toman los datos de entrada desde un dispositivo externo y los almacena en la memoria principal para que puedan ser procesados.
- **Proceso o algoritmo:** Está formado por las instrucciones que modifican los objetos a partir de su estado inicial (datos de entrada) hasta el estado final (resultados) dejando los objetos que lo contiene disponibles en la memoria principal.
- **Salida de resultados:** Conjunto de instrucciones que toman los datos finales (resultado) de la memoria principal y los envían a los dispositivos externos.



Deseas saber el número telefónico de una persona, entonces el algoritmo para resolver este problema sería:

1. Primero buscar la guía telefónica.
2. Luego abrirla en la mitad.
3. Luego ver en qué mitad está el nombre (si el nombre empieza con "b" entonces está en la primera mitad)
4. Luego se toma esta mitad y se vuelve a partir y vuelves a ver en cual nueva mitad está el nombre, y así se va repitiendo (eso es un bucle) hasta que encuentras la página.
5. Luego buscas en la página de arriba hacia abajo el nombre.
6. Luego que lo encuentras vas a columna del número.
7. Lo lees.
8. Y cierras la guía.

Cuando te vistes por la mañana.

1. Te despiertas.
2. Te lavas la cara.
3. Te duchas
4. Escoges la ropa que te vas a poner.
5. Escoges los zapatos.
6. Te quitas la pijama.
7. Te pones la ropa.
8. Luego los zapatos.
9. Te peinas
10. Listo.

1.2 Lògica de programació__Secuencias y partes de un programa.

- Algoritmo **Secuencial**: una acción (instrucción) sigue otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso.
- Algoritmo **Iterativo**: se ejecutan mediante ciclos. Estos algoritmos son muy útiles al momento de realizar tareas repetitivas (como recorrer un **arreglo/array** de datos). **La opción al uso de algoritmos iterativos es el uso de la recursividad en funciones. FOR**
- Algoritmo **Selectivo**: se utilizan para TOMAR DECISIONES. Lo que se hace es EVALUAR una condición, y, a continuación, en función del resultado, se lleva a cabo una opción u otra. **IF**
- Algoritmo **Repetición**: Es un mecanismo de lazo. Permite repetir varias veces un grupo de pasos, hasta que se satisfaga esta condición. **WHILE**

```
{Algoritmo MENU a base de 'si ... entonces ... sino'}
Declaración de variables
.....ENTEROS: opción
fin declaración de variables
inicio
.....mostrar por pantalla 'menú de opciones:'
.....mostrar por pantalla '1. Diccionario de sinónimos'
.....mostrar por pantalla '2. Diccionario de antónimos'
.....mostrar por pantalla '3. Buscar palabra'
.....mostrar por pantalla '4. Salir'
.....leer del teclado la variable opción
.....SI opción = 1 ENTONCES {lo que toque a esta opción}
.....SINO, ENTONCES .....
.....fin del SI

.....SI opción = 2 ENTONCES {lo que toque a esta opción}
.....SINO, ENTONCES .....
.....fin del SI

.....SI opción = 3 ENTONCES {lo que toque a esta opción}
.....SINO, ENTONCES .....
.....fin del SI

.....SI opción = 4 ENTONCES {lo que toque a esta opción}
.....SINO, ENTONCES Por pantalla 'opción incorrecta'
.....fin del SI
fin
```

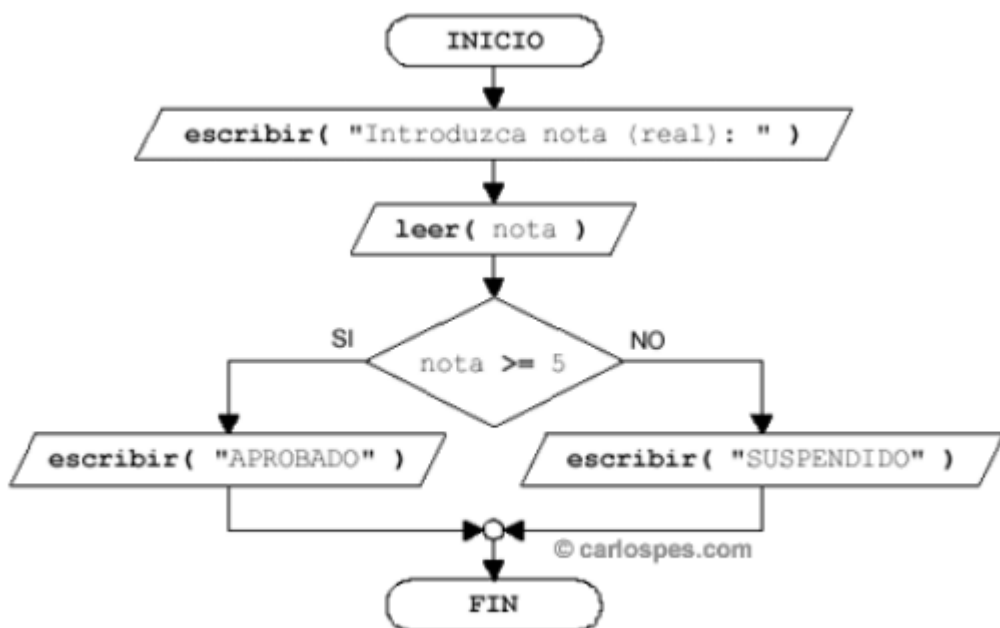
¿Hacemos una
menú?



Diagramas de flujo



En programación, los algoritmos (además de un pseudocódigo) también se pueden representar, gráficamente, por medio de diagramas de flujo.



El lenguaje unificado de modelado (**UML**, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el Object Management Group (OMG).

Es un lenguaje gráfico para **visualizar, especificar, construir y documentar un sistema**. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

1.3 Ordinogramas/diagramas de flujo. _Elementos de un ordinograma.

Todo ordinograma debe estar compuesto de:

- Un símbolo de inicio de ejecución del programa





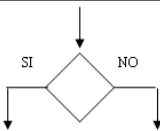
INICIO

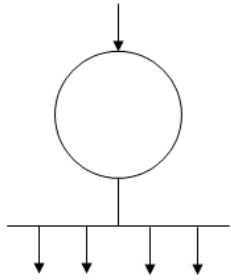


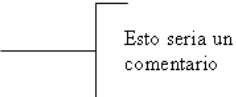
- La secuencia de operaciones necesarias para el correcto funcionamiento del programa. Las operaciones seguirán un orden (de arriba abajo y de izquierda a derecha).

- Un símbolo que indique el final del programa.

FIN

Símbolos utilizados

Símbolo	Descripción
	Para inicio/fin o para una parada indeterminada
	Símbolo de entrada/salida genérico
	Representa una operación o proceso general con datos de memoria.
	Símbolo de subprograma o subrutina. Se utiliza para realizar una llamada a un modulo del programa.
	Símbolo de decisión para realizar una pregunta con dos posibles respuestas. Es lo que llamamos símbolo de selección simple.

	<p>Símbolo de selección múltiple</p>
	<p>Símbolo de bucle definido.</p>
	<p>Conector. Se utiliza para agrupar varias líneas de flujo que salen del mismo origen.</p>
	<p>Símbolo para poner comentarios</p>

Draw.io

Wireflow

Lucidchart

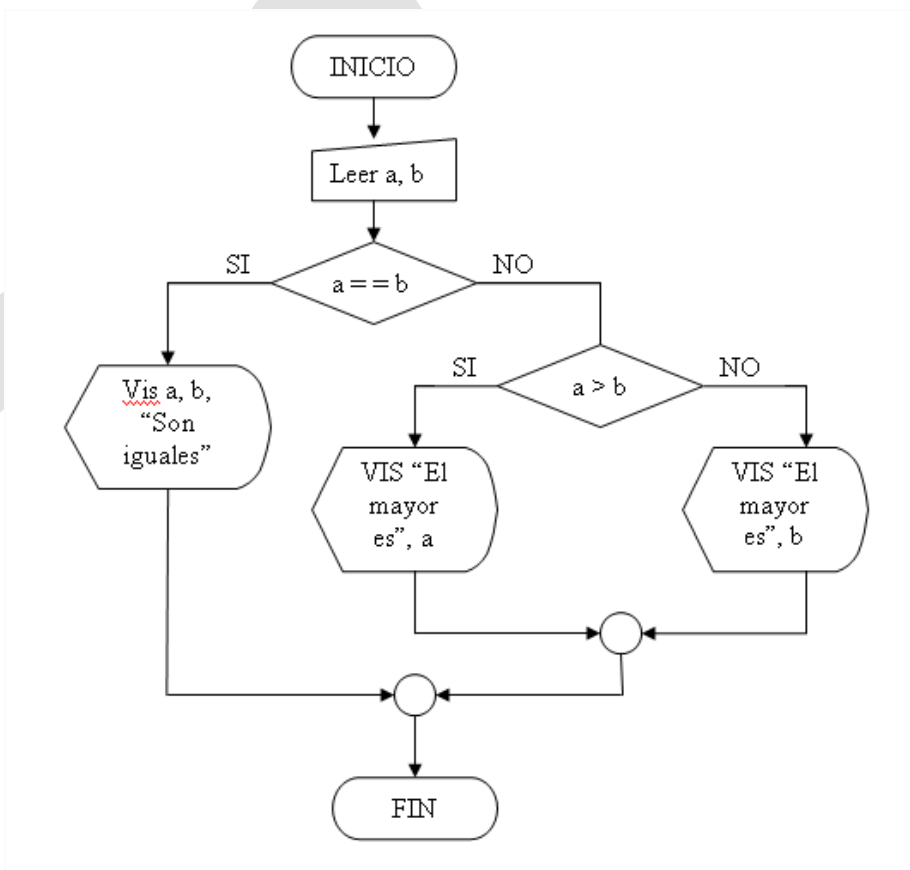
Creately

Google Drawings

Reglas a la hora de hacer ordinogramas

- Todos los símbolos utilizados deben estar unidos por líneas de flujo.
- No se pueden cruzar las líneas de flujo
- A un símbolo de proceso pueden llegarle varias líneas de flujo pero solo puede salir una de él.
- Al símbolo de inicio no puede llegarle ninguna línea de flujo
- De un símbolo de fin no puede salir ninguna línea de flujo pero si le pueden llegar varias.

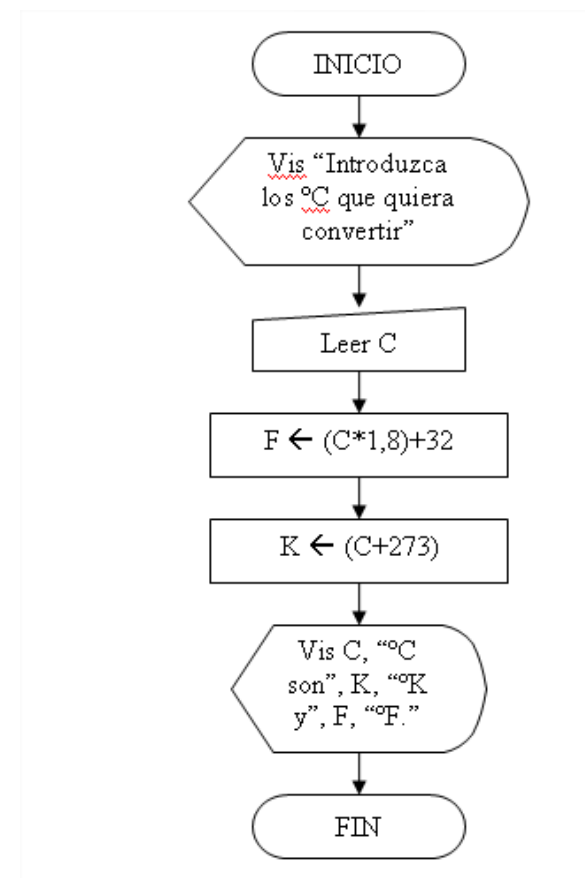
1.3 Ordinogramas/diagramas de flujo. _operaciones de un programa.



mayor de dos números

$$F = (C * 1,8) + 32$$

$$K = (C + 273)$$



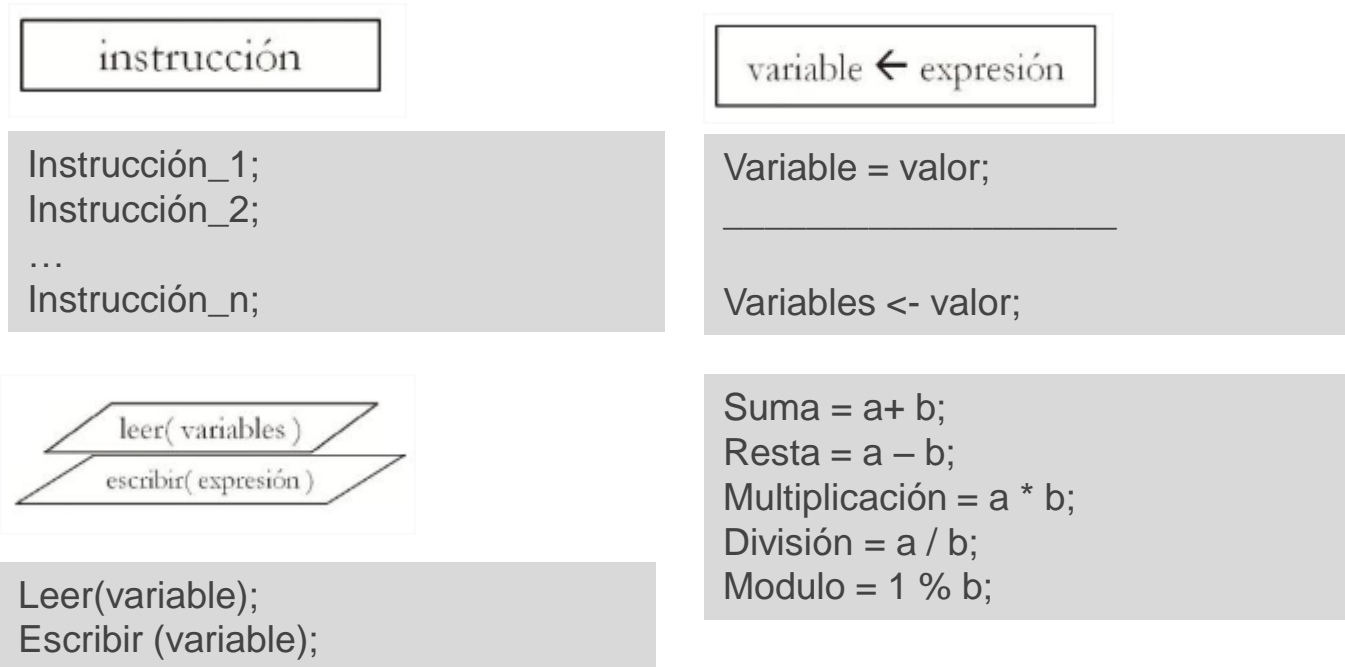
devuelve los grados introducidos en celsius a Kelvin y Fahrenheit.

Pseudocódigo

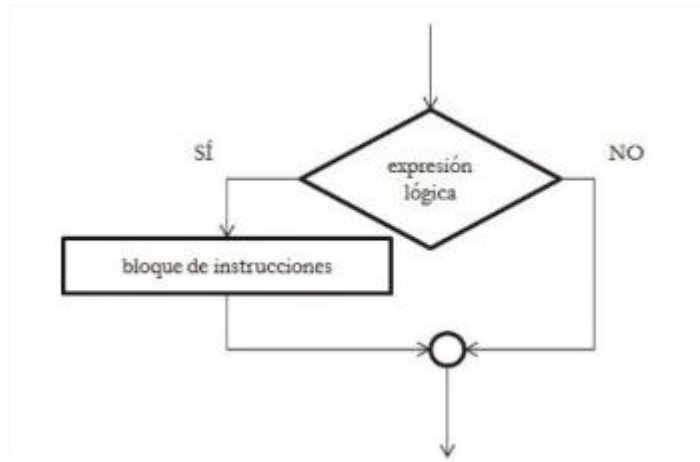
Lenguaje intermedio entre nuestro lenguaje y el lenguaje de programación.

Pseudocódigo = Casi código

El principal objetivo del pseudocódigo es el de representar la solución a un algoritmo de la forma más detallada posible, y a su vez lo más parecida posible al lenguaje que posteriormente se utilizara para la codificación del mismo.

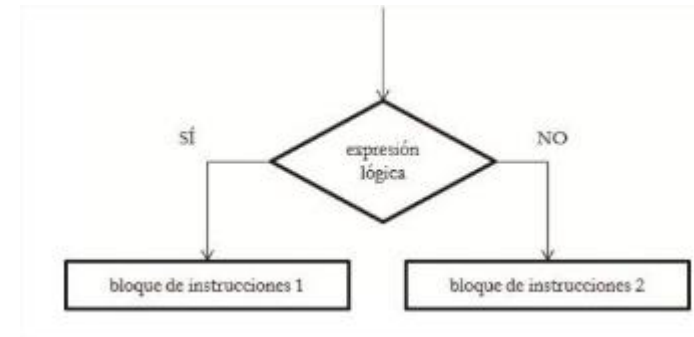


Estructura alternativa simple



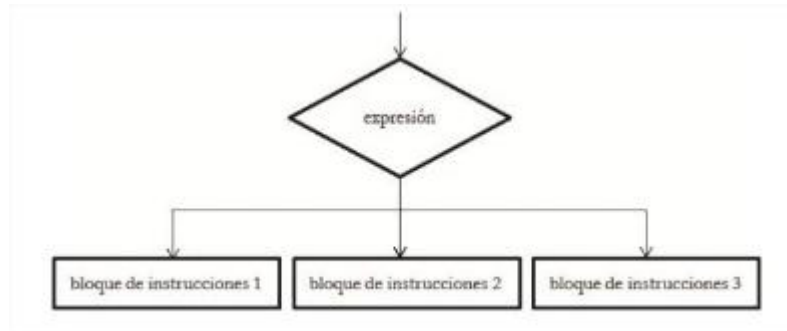
Si (expresión lógica) Entonces
instrucciones;
Fin Si

Estructura alternativa simple



Si (expresión lógica) Entonces
instrucciones_1;
Si no
instrucciones_2;
Fin Si

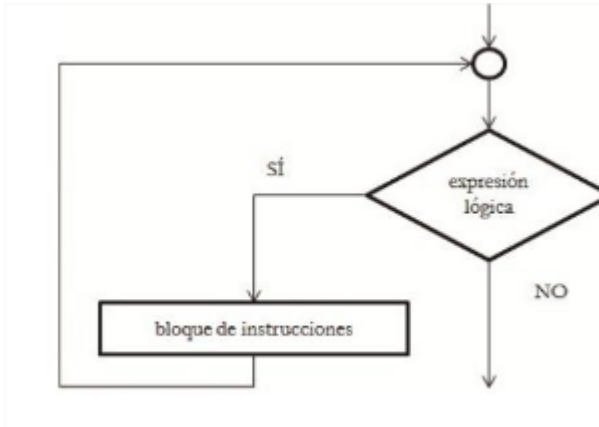
Selección multiple



IF

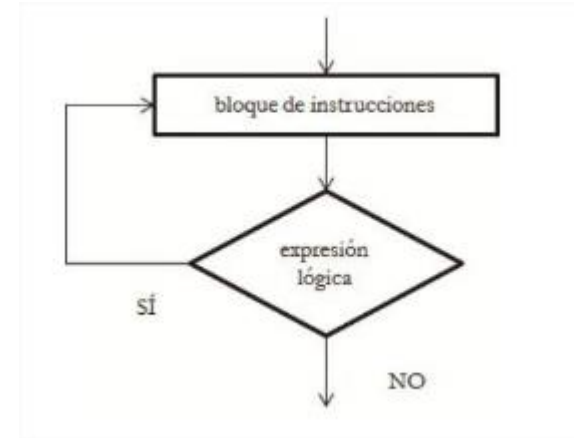
```
Según (variable) Hacer
  Caso Valor_1
    instrucciones_1;
  Fin Según
  Caso Valor_2
    instrucciones_1;
  Fin Según
  ...
  Caso Valor_n
    instrucciones_n;
  Fin Según
Por defecto
  instrucciones_defecto;
Fin Según
```

Estructura iterativas o de repetición



WHILE

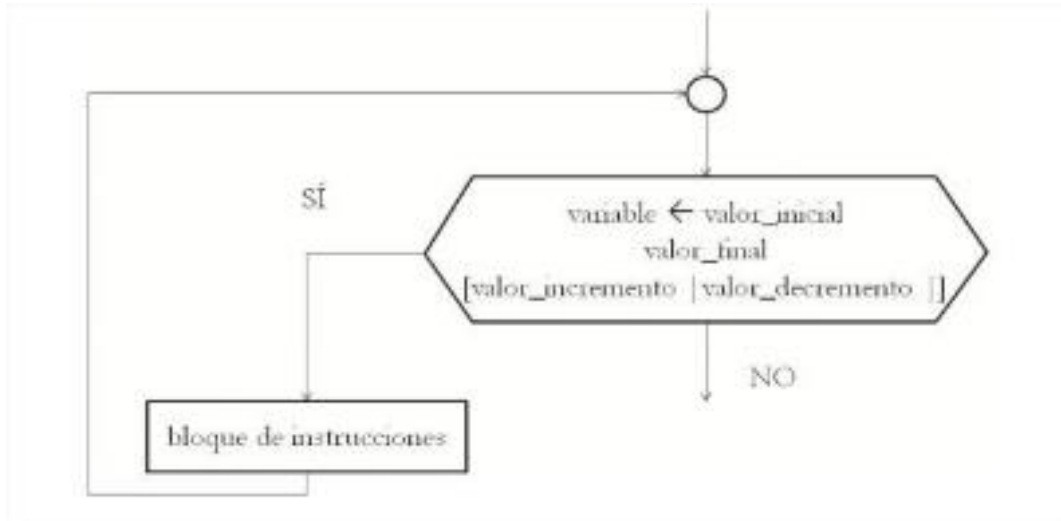
Mientras (expresión lógica) hacer instrucciones;
Fin Mientras;



DO WHILE

Hacer instrucciones;
Mientras (expresión lógica)

Estructura iterativas o de repetición



FOR

Para i=x Hasta y (con Paso Z) Hacer

Mientras (expresión lógica) Hacer
 Si (expresión lógica) Entonces
 instrucciones;
 Fin Si
 Fin Mientras

Estructura a seguir en su realización:

Cabecera:

Programa:

Modulo:

Tipos de datos:

Constantes:

Variables:

Cuerpo:

Inicio

Instrucciones

Fin

Función Potencia (a, b)

Si $b == 0$ Entonces

$p \leftarrow 1;$

Si no

$p \leftarrow a;$

Para $i \leftarrow 2$ Hasta b con paso 1 Hacer

*$p \leftarrow p * a;$*

Fin Para

Fin Si

Devolver $p;$

Fin Función

Para comentar en pseudocódigo se le antepone al comentario dos asteriscos ()

1.4 Pseudocódigos_Creación de pseudocódigo.

Ejemplos

* Programa que calcula el área de un cuadrado a partir de un lado dado por teclado.

Programa: area_cuadrado

Modulo: main **(también se puede llamar principal)

Variables:

lado: natural

area: natural

Inicio

Visualizar "Introduce el lado del cuadrado"

Leer lado

Area<- lado * lado


Visualizar "El área del cuadrado es", area

Fin

1.4 Pseudocódigos_Creación de pseudocódigo.



* Programa que visualice la tabla de multiplicar del numero introducido por teclado



```
Programa: Tabla multiplicar
Modulo: main
Variables:
t: entero
num : entero
Inicio
  Visualizar "Introduce un número"
  Leer num
  Desde t=1 hasta t=10 repetir
    Visualizar num, " X", t, "=", num*t
  Fin desde
Fin
```

Una vez que tenemos preparado un **diagrama de flujos (ordinograma u organigrama)** y un **pseudocódigo** ya podemos comenzar con **la codificación del programa** en nuestro ordenador. A partir de aquí todo varía dependiendo del lenguaje de programación que utilicemos, pero en todos los programas tendremos que definir los tipos de datos que utilizaremos.

1.5.

Objetos POO

POO/OOP significa Programación Orientada a Objetos.

La programación procedimental se trata de escribir **procedimientos o funciones que realizan operaciones en los datos**

La programación orientada a objetos se trata de **crear objetos que contienen tanto datos como funciones.**

La programación orientada a objetos tiene varias ventajas sobre la programación procedimental:

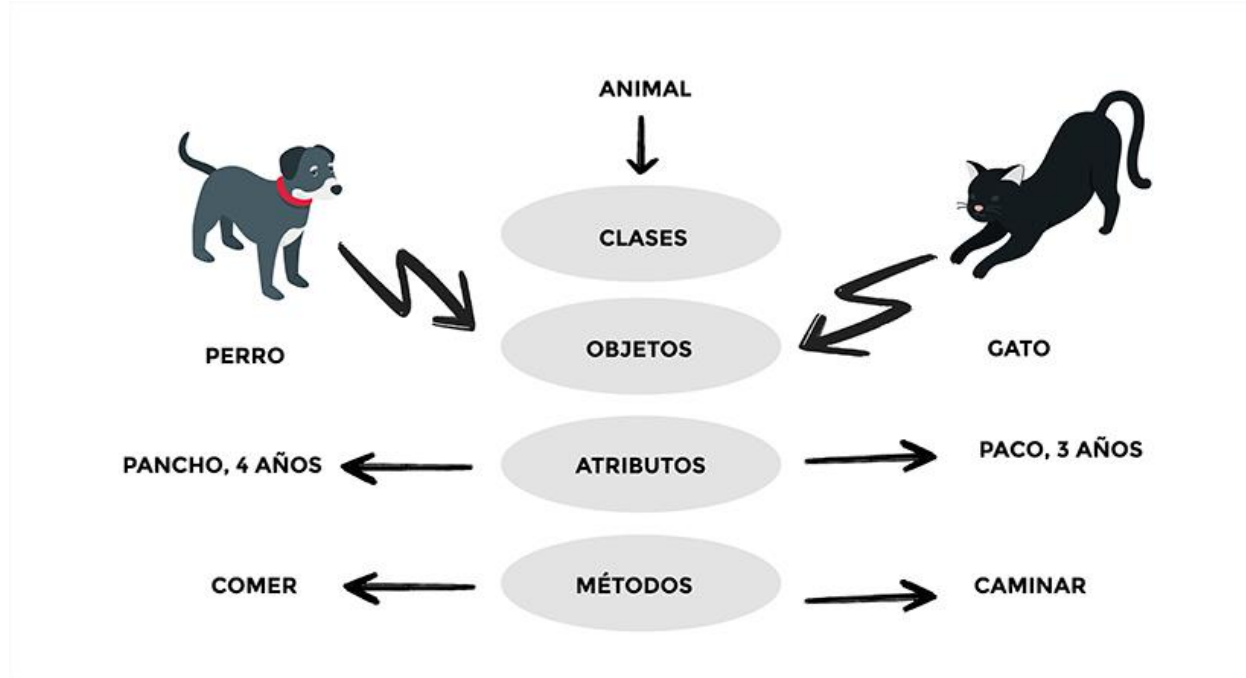
- **Reutilización** del código.
- Convierte cosas complejas en **estructuras simples reproducibles.**
- **Evita la duplicación de código.**
- Permite **trabajar en equipo** gracias al encapsulamiento ya que minimiza la posibilidad de duplicar funciones cuando varias personas trabajan sobre un mismo objeto al mismo tiempo.
- Al estar la clase bien estructurada permite la **corrección de errores** en varios lugares del código.
- **Protege la información** a través de la encapsulación, ya que solo se puede acceder a los datos del objeto a través de propiedades y métodos privados.
- La abstracción nos permite **construir sistemas más complejos** y de una forma más sencilla y organizada.

1.5 Programación Orientada a Objetos_descripción de clases, objetos e instancias

Las **clases** y los objetos son los dos aspectos principales de la programación orientada a objetos

Un **objeto** se puede definir como un campo de datos que tiene atributos y comportamiento únicos.

Clase	Coche	Fruta	Marcas
Objeto	Volvo	Manzana	HP
Objeto	Audi	Piña	Epson
Objeto	Seat	Pera	Samsung



Con la clase se pueden crear instancias de un objeto, cada uno de ellos con sus atributos definidos de forma independiente. Con esto podríamos crear un gato llamado *Paco*, con 3 años de edad, y otro animal, este tipo perro y llamado *Pancho*, con una de edad de 4 años.

Los dos están **definidos por la clase animal**, pero son dos instancias distintas. Por lo tanto, llamar a sus métodos puede tener resultados diferentes. Los dos comparten la lógica, pero cada uno tiene su estado de forma independiente.

Terminología

Clase Define las características del Objeto.

Objeto Una instancia de una Clase.

Propiedad Una característica del Objeto, como el color.

Método Una capacidad del Objeto, como caminar.

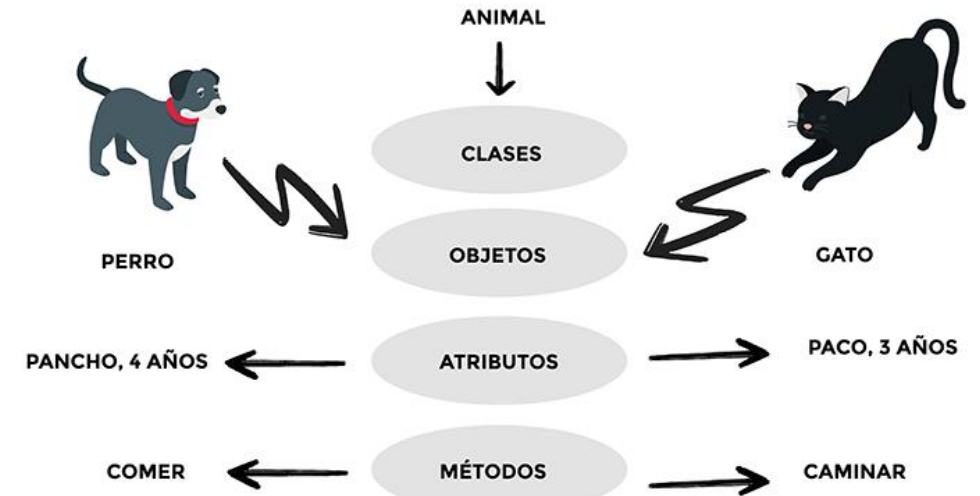
Constructor Es un método llamado en el momento de la creación de instancias.

Herencia Una Clase puede heredar características de otra Clase.

Encapsulamiento Una Clase sólo define las características del Objeto, un Método sólo define cómo se ejecuta el Método.

Abstracción La conjunción de herencia compleja, métodos y propiedades que un objeto debe ser capaz de simular en un modelo de la realidad.

Polimorfismo Diferentes Clases podrían definir el mismo método o propiedad.



Encapsulación. La implementación y el estado de cada objeto se mantienen de forma privada dentro de un límite definido o clase. Otros objetos no tienen acceso a esta clase o la autoridad para realizar cambios, pero pueden llamar a una lista de funciones o métodos públicos. Esta característica de ocultación de datos proporciona una mayor seguridad al programa y evita la corrupción de datos no intencionada.

Abstracción. Los objetos solo revelan mecanismos internos que son relevantes para el uso de otros objetos, ocultando cualquier código de implementación innecesario. Este concepto ayuda a los desarrolladores a realizar cambios y adiciones más fácilmente a lo largo del tiempo.

Herencia. Se pueden asignar relaciones y subclases entre objetos, lo que permite a los desarrolladores reutilizar una lógica común sin dejar de mantener una jerarquía única. Esta propiedad de OOP obliga a un análisis de datos más completo, reduce el tiempo de desarrollo y asegura un mayor nivel de precisión.

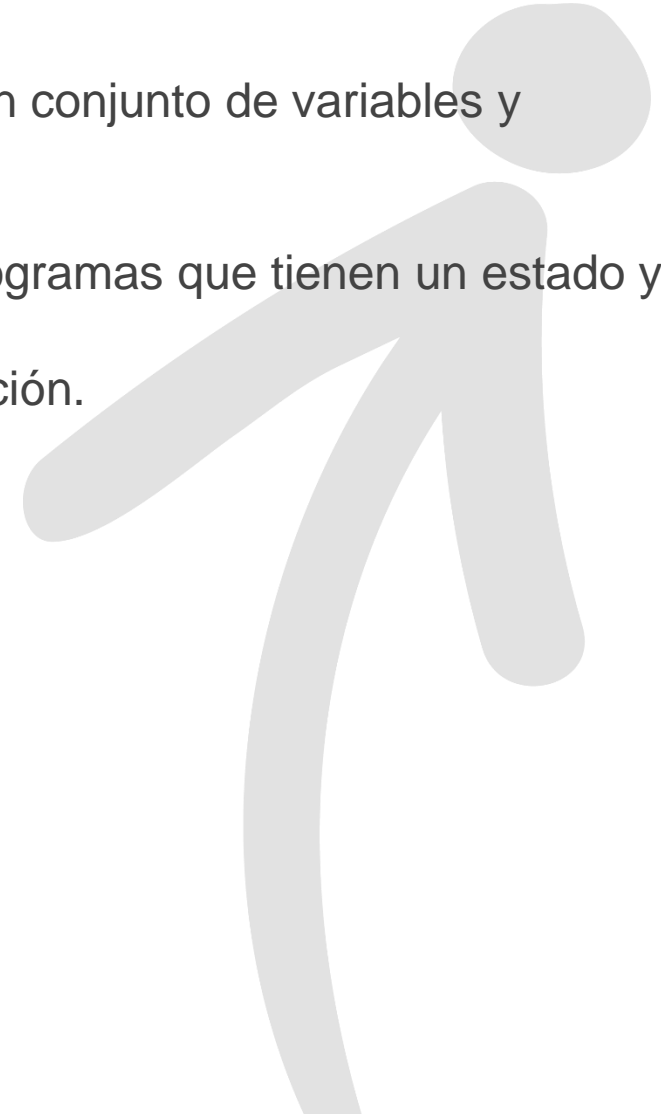
Polimorfismo. Los objetos pueden adoptar más de una forma según el contexto. El programa determinará qué significado o uso es necesario para cada ejecución de ese objeto, reduciendo la necesidad de duplicar código.

Clase: las clases son un pilar fundamental de la POO y representan un conjunto de variables y métodos para operar con datos.

Objeto: en el paradigma de programación orientada a objetos, son programas que tienen un estado y un comportamiento, conteniendo datos almacenados y tareas realizables durante su ejecución.

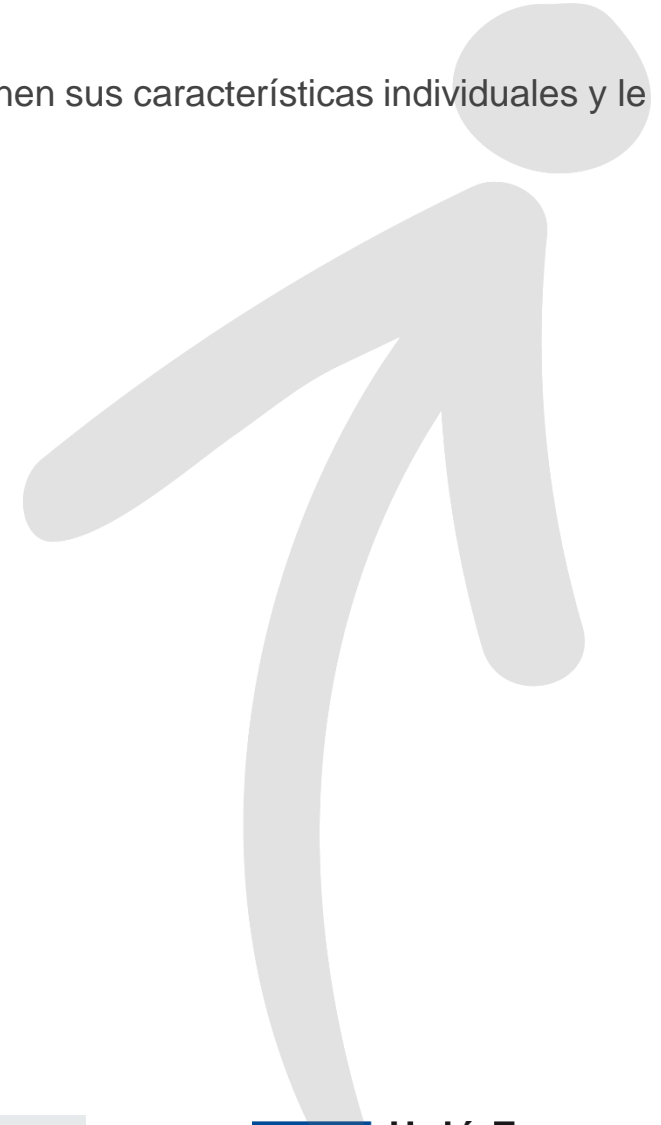
```
class Coche{  
    marca: string;  
    modelo: string;  
    color: string;  
    antigüedad: number;  
}
```

```
var coche1 = Coche("Seat", "Toledo", "azul", 1988);  
var coche2 = Coche("Audi", "S6", "rojo", 1997);  
var coche3 = Coche("Citroen", "C3", "verde", 2020);
```



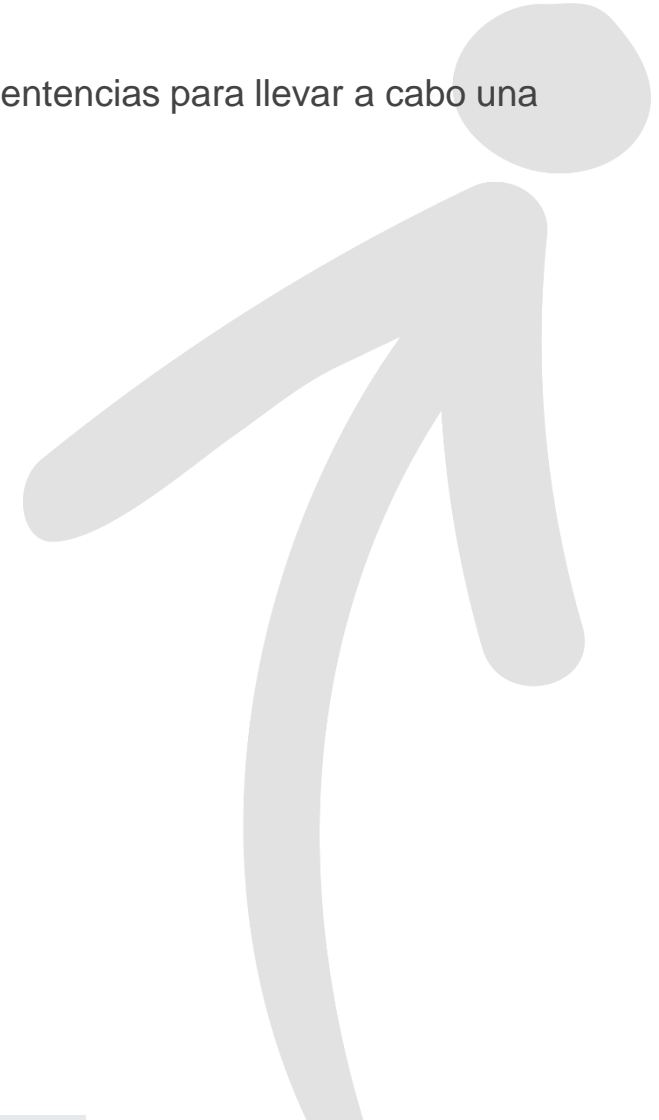
Atributos o propiedades: en POO cada objeto dispone de una serie de atributos que definen sus características individuales y le permiten diferenciarse de otros (apariencia, estado, etc).

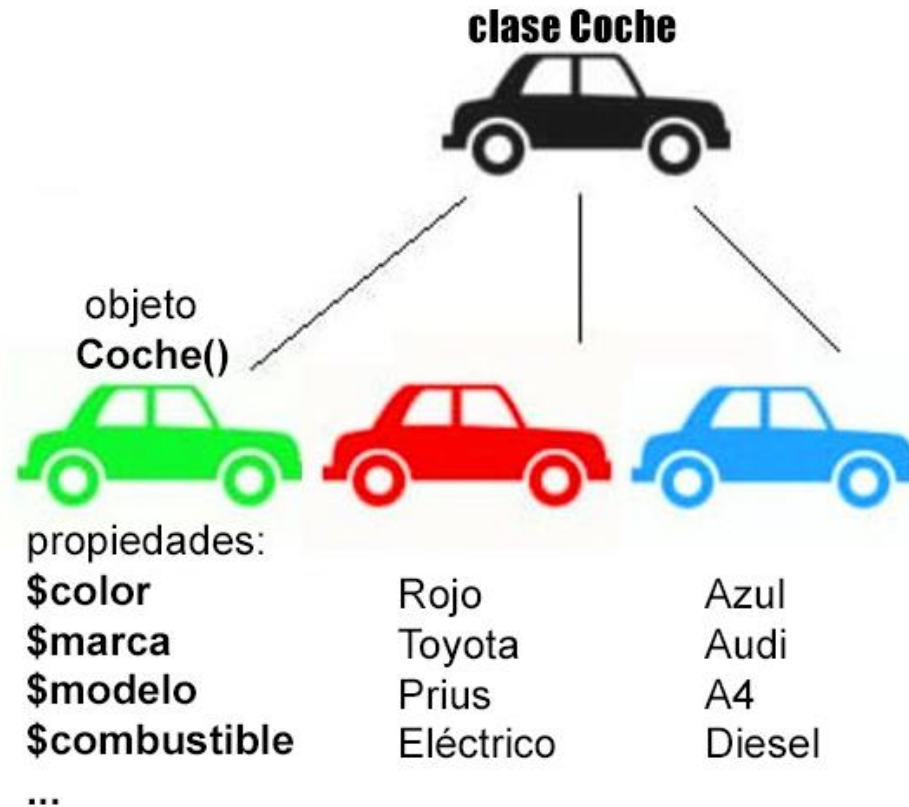
```
class Coche{  
    marca: string;  
    modelo: string;  
    color: string;  
    antigüedad: number;  
}
```



Método: es una subrutina que puede pertenecer a una clase u objeto, y son una serie de sentencias para llevar a cabo una acción.

```
class Coche{  
    marca: string;  
    modelo: string;  
    color: string;  
    antigüedad: number;  
  
    mostrarColor(){}  
    arrancar(){}  
    frenar(){}  
    girarDerecha(){}  
    girarIzquierda(){}  
}
```





MÉTODOS

llenarDeposito()
arrancarMotor()
frenar()
acelerar()
tocarClaxon()
...



```
var camiseta_1 = new Camiseta(); //objeto vacio
```

```
camiseta_1.color = "Rojo";  
camiseta_1.marca = "Nike";  
camiseta_1.modelo = "Silver88";  
camiseta_1.talla = "XL";  
camiseta_1.precio = 55;
```

```
var camiseta_2 = new Camiseta(); //objeto vacio
```

```
camiseta_2.color = "Azul";  
camiseta_2.marca = "Adidas";  
camiseta_2.modelo = "Verano";  
camiseta_2.talla = "L";  
camiseta_2.precio = 47;
```

```
console.log(camiseta_1, camiseta_2);
```

```
// camiseta_1 = Camiseta {color: "rojo", modelo: "Silver88", marca: "Nike", talla: "XL", precio: "55"}
```

```
// camiseta_2 = Camiseta {color: "azul", modelo: "verano", marca: "Adidas", talla: "L", precio: "47"}
```

Ejemplos de código en diferentes lenguajes

<http://helloworldcollection.de/>



El **teorema del programa estructurado** es la base teórica sobre la que se construyó esta nueva forma de programar, ya que nos da la característica fundamental de la programación estructurada. Postula que, **simplemente con la combinación de tres estructuras básicas, es suficiente para expresar cualquier función computable.**

- Los programas desarrollados con la programación estructurada **son más sencillos de entender,**
- Los programas tendrán una **estructura clara**
- La **fase de prueba y depuración** de los programas **se optimiza**
- El coste del **mantenimiento** de los programas que usan la programación estructurada es **más reducido.**
- Son **más rápidos de crear** y los programadores aumentan su rendimiento.

Las 3 estructuras básicas

- Secuencia.
- Selección o condicional.
- Iteración (ciclo o bucle).



Los lenguajes de scripts son un **tipo específico de lenguajes** informáticos que se pueden utilizar para **dar instrucciones a otro software**, como un **navegador** web, un **servidor** o una **aplicación** independiente.

Muchos de los lenguajes de scripts más populares de hoy en día son lenguajes de programación, como **JavaScript, PHP, Ruby, Python**, y varios otros.

Los lenguajes de scripts pueden realizar diferentes acciones dentro de un entorno de tiempo de ejecución particular, como automatizar la ejecución de tareas, mejorar la funcionalidad del software principal, realizar configuraciones, extraer datos de conjuntos de datos y otros.

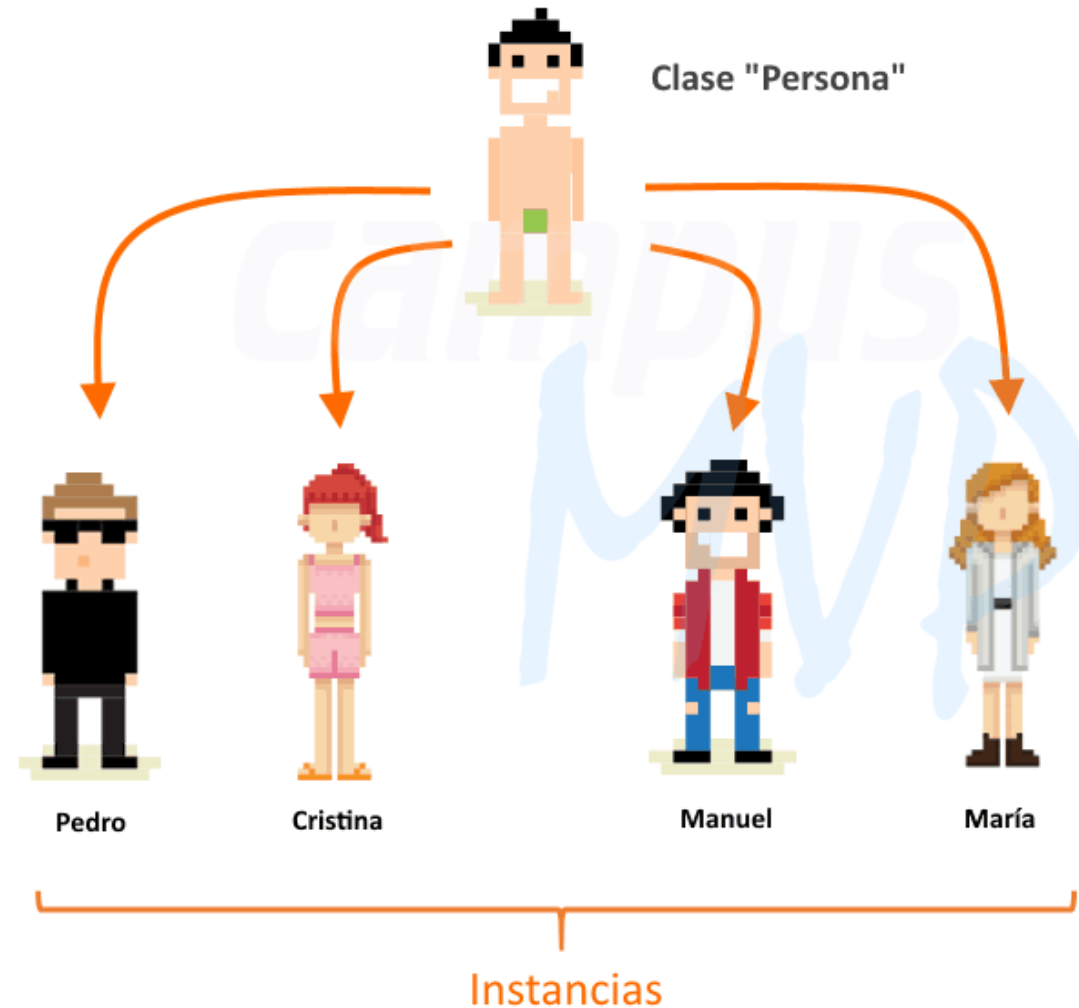

```
<HTML>
<HEAD>
<TITLE>Ejemplo01.htm</TITLE>

<SCRIPT LANGUAGE="JavaScript">
  //Visualizar un mensaje de bienvenida
  alert(";Bienvenido a nuestra página!");
</SCRIPT>

</HEAD>
<BODY>
<a href='Ejemplo02.html'>Ir al siguiente ejemplo...</a>
</BODY>
</HTML>
```

La Programación Orientada a Objetos (POO, en español; OOP, según sus siglas en inglés) es un paradigma de programación que parte del concepto de "objetos" como base, los cuales contienen información en forma de campos (a veces también referidos como atributos o propiedades) y código en forma de métodos.

Algunas características clave de la programación orientada a objetos son herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento.







Barcelona

Francesc Tàrraga 14
08027 Barcelona
93 351 78 00

Madrid

Campanar 12
28028 Madrid
91 502 13 40

Reus

Alcalde Joan Bertran 34-38
43202 Reus
977 31 24 36

info@grupcief.com

www.grupcief.com

