



**Universidad
Europea de Madrid**

LAUREATE INTERNATIONAL UNIVERSITIES

INTRODUCCIÓN A COMPILADORES Y LENGUAJES FORMALES

FUNDAMENTOS DE COMPILADORES

Cartagena99

**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70**

© Todos los derechos de propiedad intelectual de esta obra pertenecen en exclusiva a la Universidad Europea de Madrid, S.L.U. Queda terminantemente prohibida la reproducción, puesta a disposición del público y en general cualquier otra forma de explotación de toda o parte de la misma.

La utilización no autorizada de esta obra, así como los perjuicios ocasionados en los derechos de propiedad intelectual e industrial de la Universidad Europea de Madrid, S.L.U., darán lugar al ejercicio de las acciones que legalmente le correspondan y, en su caso, a las responsabilidades que de dicho

**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70**

Cartagena99

Índice

| | |
|---|----|
| Presentación | 4 |
| ¿Por qué son necesarios los compiladores? | 5 |
| ¿Cómo nos independizamos de la máquina? | 7 |
| Herramientas necesarias para los compiladores | 9 |
| Proceso de compilación | 13 |
| Fases de la compilación | 16 |
| ¿Cómo diseñamos un compilador? | 20 |
| ¿Cómo diseñamos un compilador? | 21 |
| Compilador cruzado | 22 |
| ¿Por qué es importante entender los compiladores? | 23 |
| Resumen | 24 |



Cartagena99

**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70**

Presentación

El objetivo de este tema es que el estudiante comprenda la problemática asociada a los **compiladores**, cuál fue su origen, por qué son necesarios y cómo han evolucionado, además de presentar una visión global que permita identificar las distintas fases que los componen.

Los objetivos a conseguir en este tema son:

- Saber por qué son necesarios los compiladores.
- Conocer cómo nos independizamos de la máquina.
- Conocer los tipos de compiladores y las fases de un compilador.
- Reconocer cómo se diseña un compilador.
- Motivaciones.

Veremos, por tanto, una reseña histórica para entender por qué era importante **independizarse de la máquina**, así como las herramientas necesarias para compilar un programa y por qué son necesarias. Debe verse el **proceso de desarrollo de un compilador** como un caso especial de ingeniería del software.



CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

¿Por qué son necesarios los compiladores?

Con la llegada de **John Von Neumann** al proyecto **ENIAC** (1945), y tras sufrir la problemática de tener que cablear el ordenador para cada nueva tarea, decide resolver este problema. Lo consigue en 1949, escribiendo **secuencias de código o programas** que hacen que estos ordenadores realizaran los cálculos deseados. Estos programas se escribían utilizando códigos numéricos que representaban las operaciones que se iban a realizar. El lenguaje utilizado para construir estos programas se denominó **lenguaje máquina** porque estaba totalmente relacionado con el hardware de la máquina.

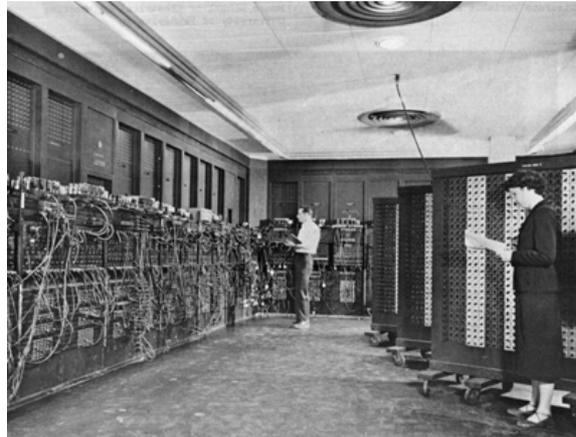
Como se puede ver en el siguiente ejemplo, esta forma de escribir un programa es difícil y tediosa, por lo que pronto se dio nombre a los códigos de las operaciones y a las direcciones de memoria.



Código máquina vs código

Ejemplo

ensamblador



Máquina ENIAC (Electronic Numerical Integrator and Computer) en Philadelphia, Pennsylvania (US Army)

Con el lenguaje ensamblador se mejoró enormemente la **velocidad y exactitud**

con la que se escribían los programas. De hecho, todavía se encuentra en uso en situaciones donde se necesita velocidad y se tiene poco espacio para el código.

De todas formas, el lenguaje ensamblador no es perfecto. Sus principales **desventajas** son:

- No es fácil de escribir.
- Es difícil de leer y entender.
- Depende de la máquina para la que se ha escrito.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Ejemplo**Ejemplo de código máquina vs código ensamblador**

Un ejemplo de este código máquina es el siguiente:

| | Código operación | Dirección |
|----------------|------------------|-----------|
| Código máquina | 00010101 | 10000011 |
| Ensamblador | LOAD | X |

The logo for 'Cartagena99' features the text 'Cartagena99' in a stylized, blue, serif font. The '99' is significantly larger and more prominent than the 'Cartagena' part. The text is set against a light blue, abstract background that resembles a stylized 'C' or a wave. Below the text, there is a horizontal orange and yellow gradient bar.

**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70**

¿Cómo nos independizamos de la máquina?

Como ya se ha indicado, el ensamblador depende totalmente de la máquina para la que se ha escrito, y esto implica **reescribir** otra vez el programa si se va a ejecutar en otra máquina. Por tanto, una vez superado el problema del cableado de la máquina para cada nuevo programa, ahora el objetivo a conseguir era independizarse de la máquina.

Es decir, se trataba de generar un código intermedio, denominado **código objeto**, que nos **ahorrara recordar las direcciones de memoria**, así como otros aspectos totalmente ligados con la máquina. Se pensaba que no iba a ser fácil, además de poco eficiente. Aquí vemos un ejemplo de lo que se estaba buscando:

| | Código operación | Dirección |
|-------------|------------------|-----------|
| Ensamblador | MOV X, 5 | 1000011 |
| Código | X=5 | X |

Había que conseguir crear un lenguaje que nos permitiera realizar acciones de tal forma que fuera **fácil de manejar y aprender** por una persona. Este tipo de lenguajes se denominó **de alto nivel**, en contraposición con los de bajo nivel (código máquina y ensamblador) totalmente dependientes de la máquina.

De esta forma, se llegó al que se denomina el **primer compilador** realizado por **Grace Hopper**, el A-0. Se empezó a trabajar en él en el otoño de 1951 y la primera rutina "compilada" se probó con éxito en un **UNIVAC** en la primavera de 1952 (Hopper & Mauchly, 1953). El concepto de compilador de estos momentos consistía en la aceptación del pseudocódigo, su decodificación, la búsqueda de la subrutina apropiada, la asignación de las direcciones de memoria a las subrutinas en el programa, la selección e inserción de las posiciones de memoria de los argumentos y resultados, la organización de la transferencia del control y la escritura del programa terminado en una cinta. Esto era más un cargador o enlazador que el concepto actual de compilador, pero fue la primera vez que se usó la palabra **compiler**.

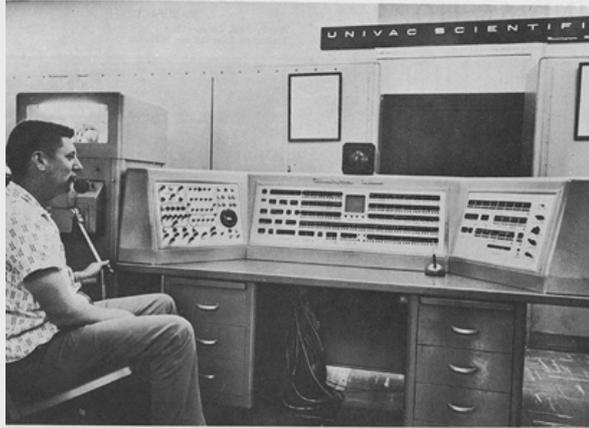
Por otro lado, en 1954, John Backus comenzó el desarrollo de un compilador de **FORTRAN** para IBM, concretamente para el IBM 704, que le llevó dos años y medio y 18 hombres para realizarlo. Consistió en dos componentes: el lenguaje FORTRAN y el traductor para el IBM 704. Este

**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70**

Cartagena99

UNIVAC



Máquina UNIVAC (US Army)

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Herramientas necesarias para los compiladores

A continuación se indican las **herramientas software necesarias** para que el proceso de compilación se realice de una forma adecuada y completa:



CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
 LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
 CALL OR WHATSAPP:689 45 44 70



CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
 LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
 CALL OR WHATSAPP:689 45 44 70

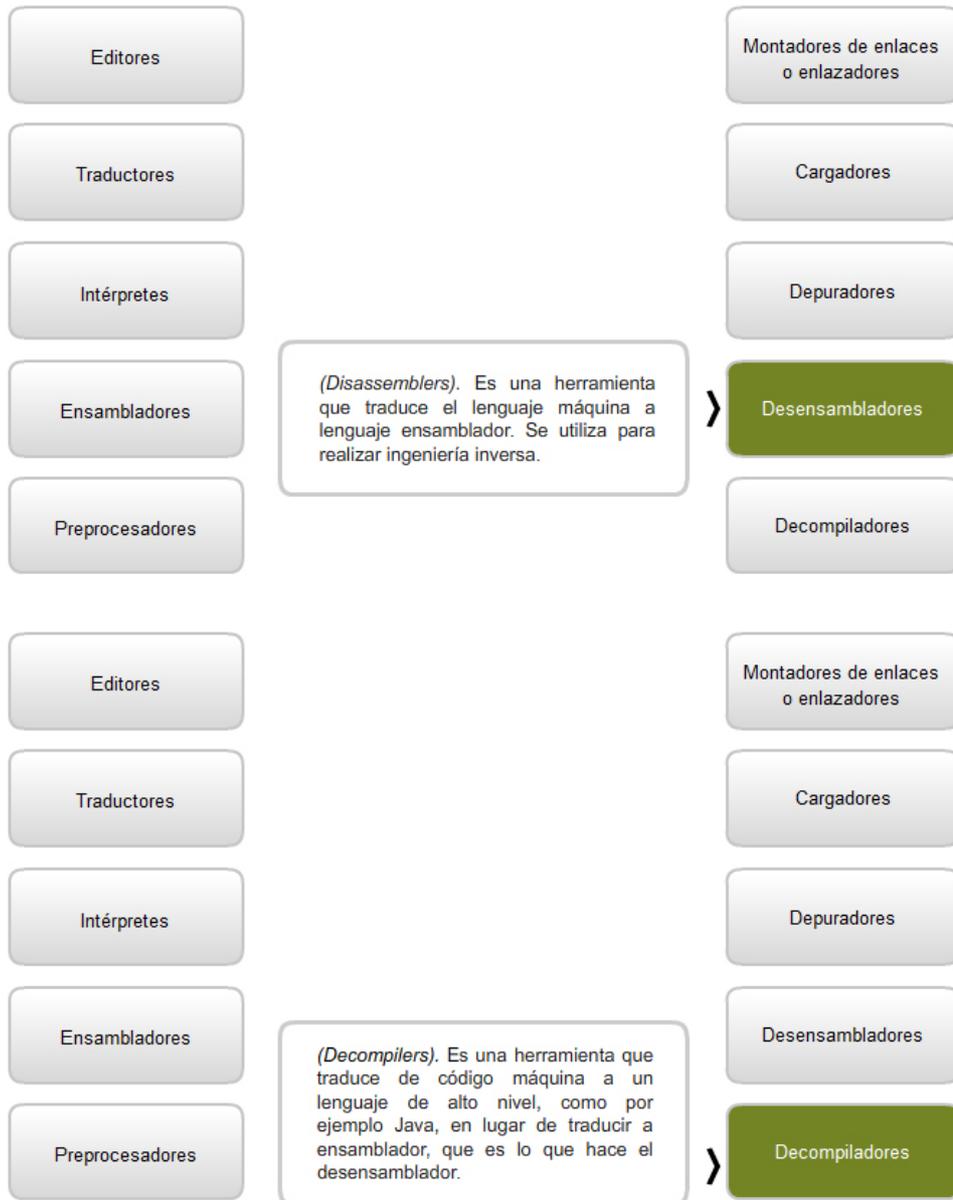




CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
 LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
 CALL OR WHATSAPP:689 45 44 70



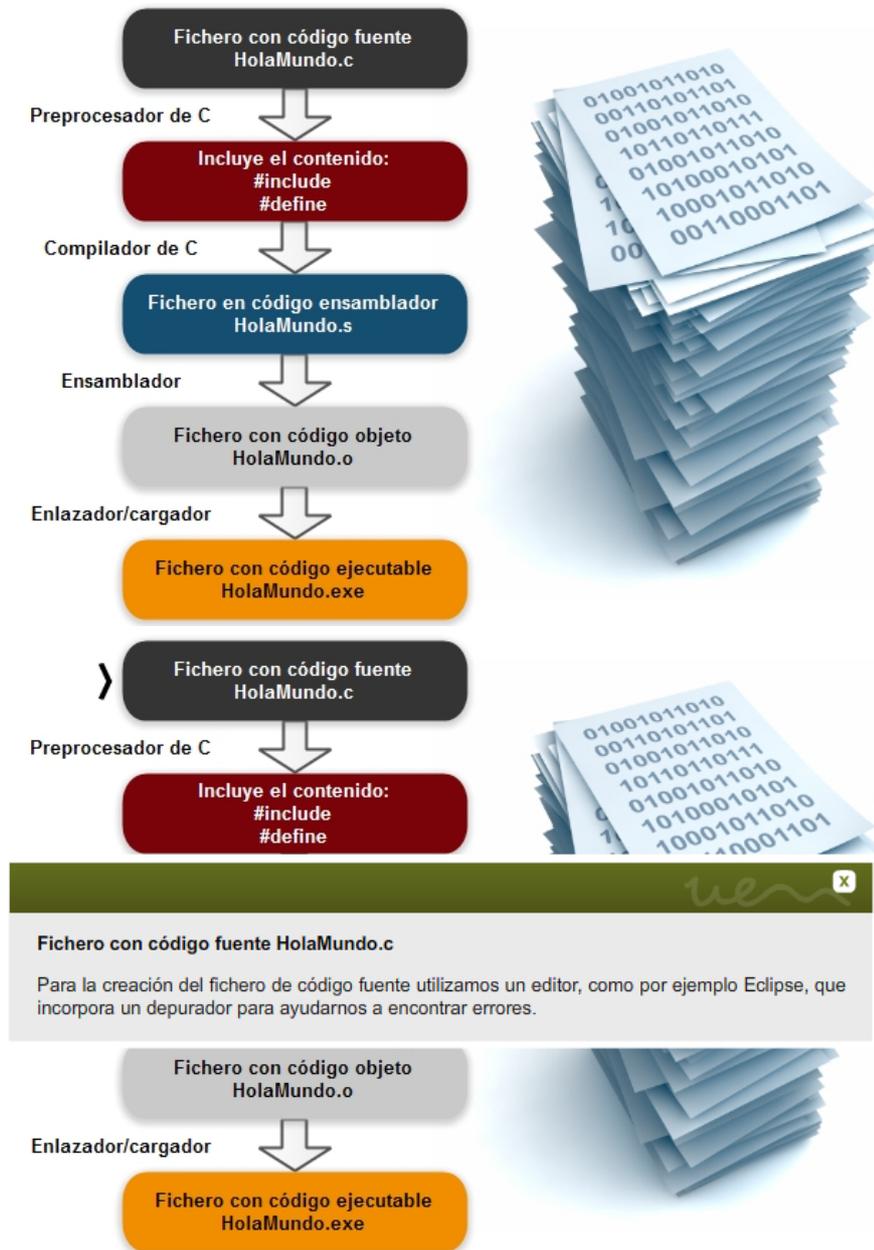


CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
 LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
 CALL OR WHATSAPP:689 45 44 70

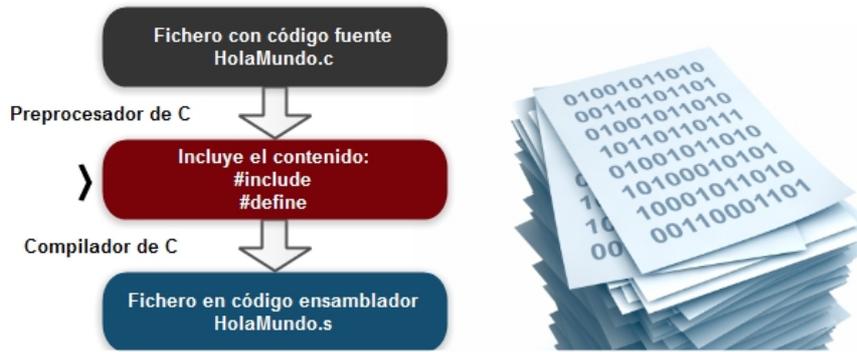
Proceso de compilación

La figura muestra el **proceso de compilación**, para, a partir de un fichero con el código fuente, obtener un fichero con el código ejecutable.



CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
 LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
 CALL OR WHATSAPP:689 45 44 70



Incluye el contenido: #include #define

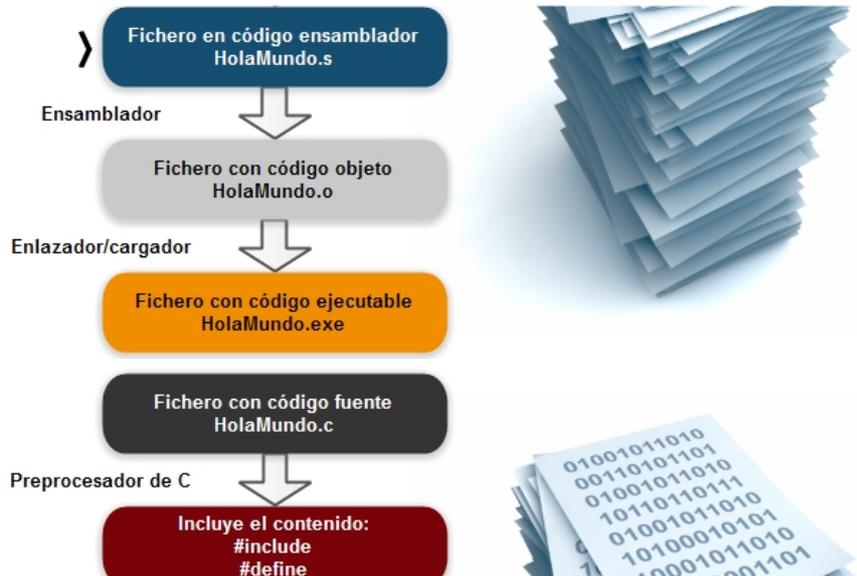
El preprocesador de C (cpp) incluye el contenido de un fichero (Ej: #include <stdio.h>) a través de las directivas #include, #define o #if.

Fichero con código ejecutable
HolaMundo.exe

Fichero con código fuente
HolaMundo.c

Fichero en código ensamblador HolaMundo.s

Traduce el código fuente una vez lo ha analizado a código ensamblador. Este proceso lo explicaremos detalladamente en la siguiente pantalla.



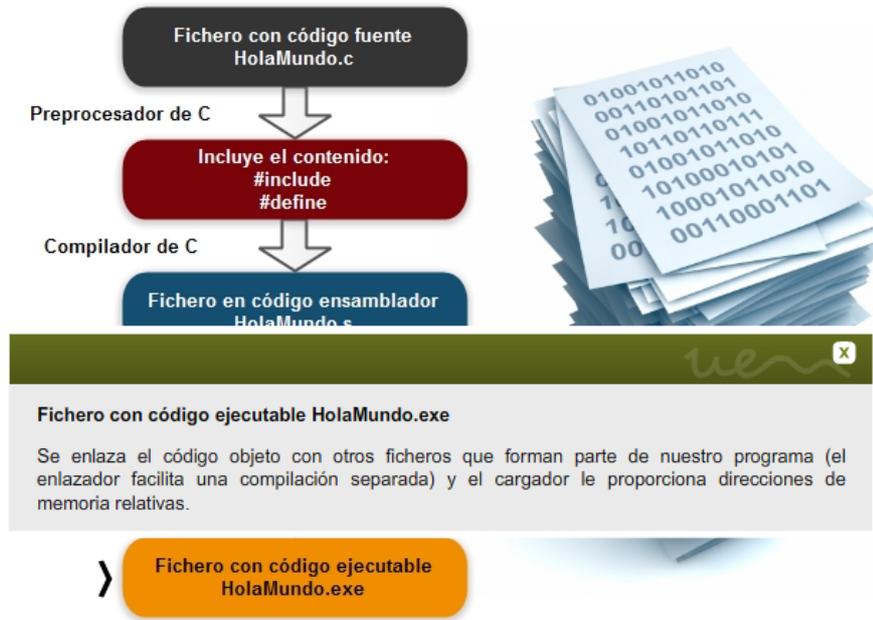
Fichero en código ejecutable HolaMundo.exe

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Fichero con código ejecutable
HolaMundo.exe





**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70**

Fases de la compilación

En esta figura vemos las **fases por las que pasa un programa fuente** desde que es recibido por el analizador léxico hasta que se genera el código objeto y que va al cargador/enlazador para su ejecución.

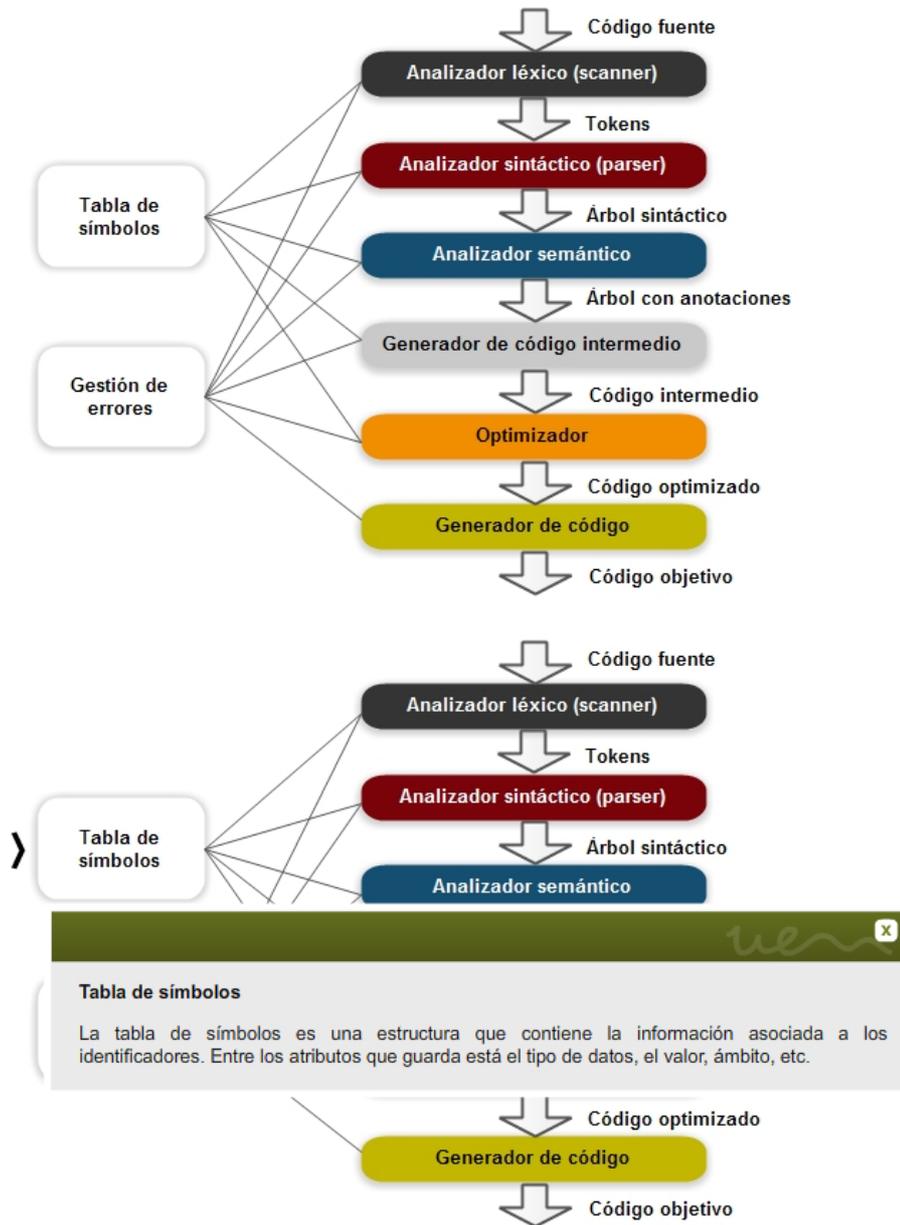
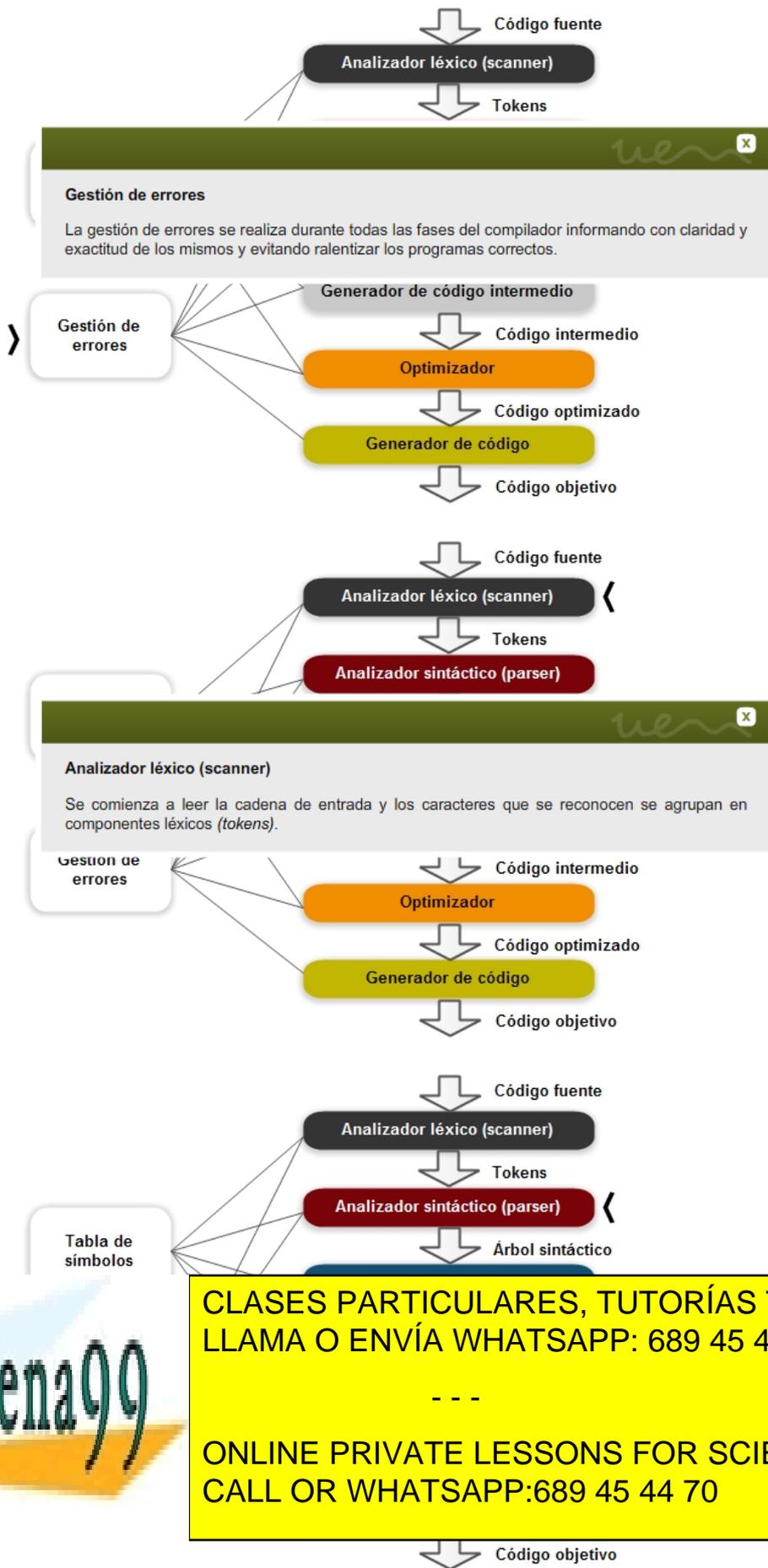


Tabla de símbolos
 La tabla de símbolos es una estructura que contiene la información asociada a los identificadores. Entre los atributos que guarda está el tipo de datos, el valor, ámbito, etc.



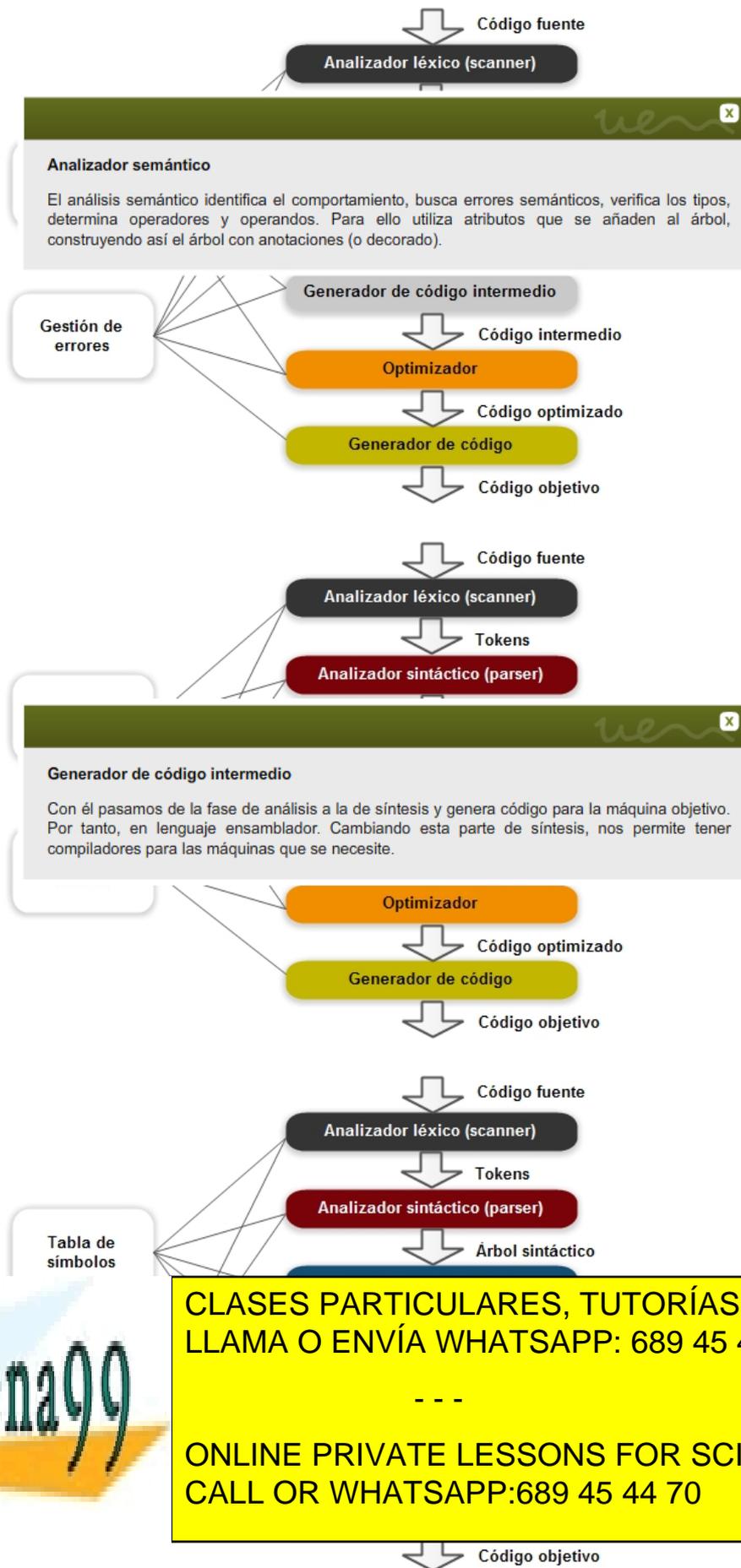
**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
 LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
 CALL OR WHATSAPP:689 45 44 70**



**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
 LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
 CALL OR WHATSAPP:689 45 44 70**

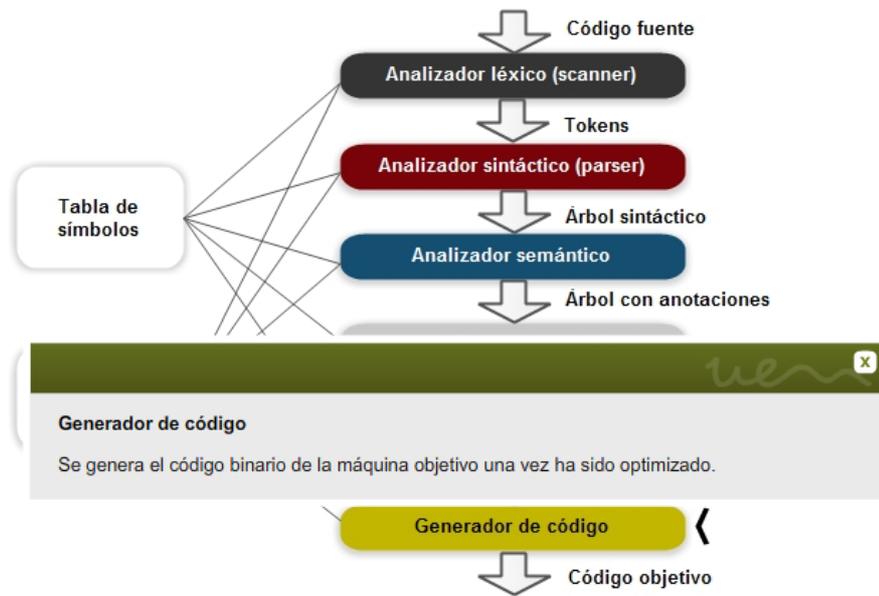


CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
 LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
 CALL OR WHATSAPP:689 45 44 70

Código objetivo





Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

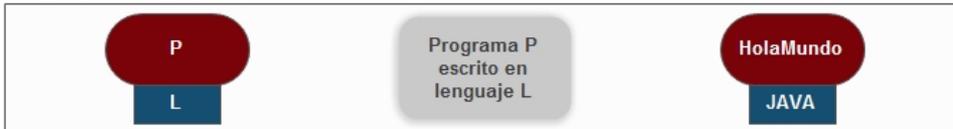
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

¿Cómo diseñamos un compilador?

Hasta ahora hemos partido de un lenguaje fuente y se trata de obtener el lenguaje objeto, pero no hemos tenido en cuenta el **lenguaje de implementación del propio compilador**, que es necesario para proporcionar la visión global necesaria para el diseño del mismo.

A la hora de diseñar un compilador utilizamos los diagramas de Tombstone, que son un conjunto de piezas de puzzle muy útiles para esta tarea y que a su vez tienen unas reglas:

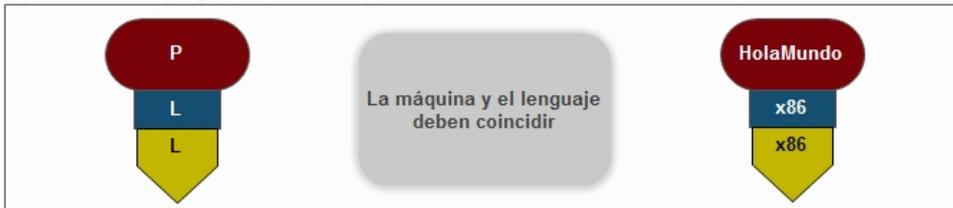
Diagramas para programas



Diagramas para máquinas



Ejecución de un programa en una máquina

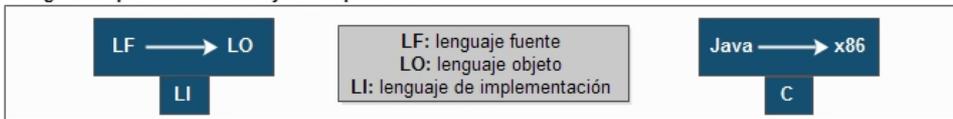


CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
 LLAMA O ENVÍA WHATSAPP: 689 45 44 70

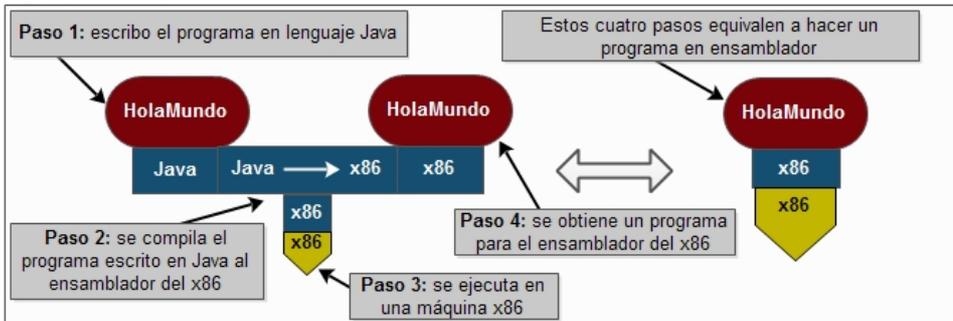
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
 CALL OR WHATSAPP:689 45 44 70

¿Cómo diseñamos un compilador?

Diagramas para traductores y/o compiladores



Ejemplo de compilación



Este proceso se sintetiza y se generaliza para cualquier programa escrito en Java para una máquina x86, utilizando la notación en T:



CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

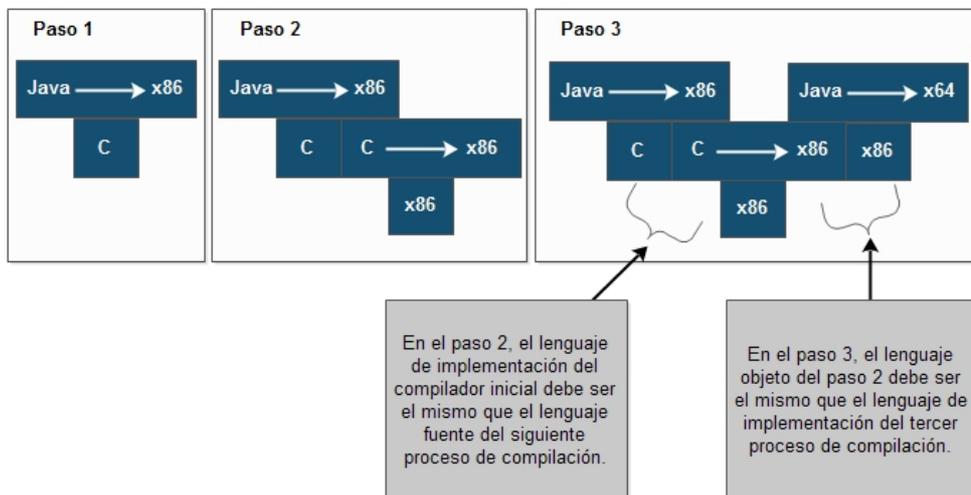
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

Compilador cruzado

Los primeros compiladores se escribieron directamente para la máquina en la que se iban a ejecutar, es decir, para su código máquina. Esto, evidentemente, ya no es necesario hacerlo y si queremos que un compilador sea **portable** a otra arquitectura de máquina (por ejemplo, para cuando cambia el procesador) debemos resolver el problema: tenemos un compilador escrito en un lenguaje que queremos que se ejecute en una máquina distinta de la máquina objetivo. A esto se denomina obtener un **compilador cruzado**.

Partimos de un compilador de Java (lenguaje fuente), escrito en C (lenguaje de implementación) para una máquina x86 (lenguaje objeto), y queremos que se ejecute en una máquina de 64 bits (x64). ¿Cómo lo resolvemos?



CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

¿Por qué es importante entender los compiladores?

Hay varios motivos que hacen que esta materia sea un componente esencial de una Ingeniería en Informática:

- Permite concentrar en una sola materia los conocimientos de Arquitectura de ordenadores, Ingeniería del software, Algoritmia, Teoría de lenguajes formales y Lenguajes de programación.
- El entender cómo se construyen los compiladores permite realizar programas más eficientes y correctos.
- Nos permite conocer mejor las decisiones de diseño que llevan a unos lenguajes a tener determinadas características: sobrecarga de operadores, polimorfismo, tipificación estática o dinámica y un largo etcétera.
- Nos permite aplicar las técnicas y herramientas a otros campos: formateadores de texto (LaTeX, Tex, etc.), intérpretes gráficos (PovRAY, GIF, Postcript, etc.), lenguajes de simulación (MSDL, CBML, GPSS, etc.).
- Antes de la aparición de las técnicas que vamos a aprender, los compiladores eran muy poco eficientes.



**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70**

Cartagena99

Resumen

En este tema hemos dado un repaso a la historia de los compiladores, desde los inicios con Grace Hooper y John Von Neumann y su interés por independizarse de la máquina y pasar primero por los lenguajes de bajo nivel como ensamblador, para llegar a los de alto nivel, como FORTRAN con John Backus.

Hemos pasado por las herramientas necesarias para realizar la compilación de una forma adecuada y completa: editores, traductores, intérpretes, ensambladores, preprocesadores, enlazadores, cargadores, depuradores, desensambladores y decompiladores. Además, hemos aprendido el proceso de compilación.

Por otro lado, hemos aprendido las fases de la compilación:

- **Análisis:** léxico, sintáctico y semántico.
- **Síntesis:** generación de código intermedio, optimización y generación de código.

También hemos aprendido, por medio de los diagramas de **Tombstone**, a diseñar un compilador para otra máquina distinta de la que originalmente fue diseñado, denominándose **compilador cruzado**.

Por último, hemos comprendido por qué es necesario entender los compiladores: concentrar varias materias en una permite realizar programas más eficientes y correctos, tomar decisiones de diseño de una forma más completa y aplicar las técnicas en el desarrollo de otros tipos de herramientas relacionadas con los procesadores de lenguaje.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99