Introducción

Que nos impulsa a programar

Podríamos definir que el motivo que impulsa a las personas a aprender lenguajes de programación y técnicas para el mismo es la de emplear a las computadoras como una herramienta para resolver problemas.

Podríamos distinguir los siguientes pasos para poder definir una posible solución:

Etapa 1	Etapa 2	Etapa 3	Etapa 4
Definición o análisis del problema.	Diseño del algoritmo	Transformación del algoritmo en un programa.	Ejecución y validación del programa.

Introducción a C++

C++: es un lenguaje de programación de propósito general y de alto nivel con facilidades para la programación a bajo nivel. Tiene un modelo de tipos de dato estáticos y es un lenguaje que soporta tres paradigmas de programación:

- Programación procedimental.
- Programación orientada a objetos.
- Programación genérica.

Cuenta con abstracción de datos e identificación de tipo de dato en tiempo de ejecución. Desde 1990, C++ ha sido uno de los lenguajes comerciales de programación más populares.

Beneficios de trabajar con C++

- C ++ es un lenguaje altamente portátil y, a menudo, es la tecnología elegida para el desarrollo de aplicaciones multiplataforma.
- C ++ es un lenguaje de programación orientado a objetos e incluye clases, herencia, polimorfismo, abstracción de datos y encapsulación.
- C ++ tiene una rica biblioteca de funciones.
- C ++ permite el manejo de excepciones y la sobrecarga de funciones que no son posibles en C.
- C ++ es un lenguaje potente, eficiente y rápido. Encuentra una amplia gama de aplicaciones, desde aplicaciones de GUI hasta gráficos en 3D para juegos y simulaciones matemáticas en tiempo real.

Historia

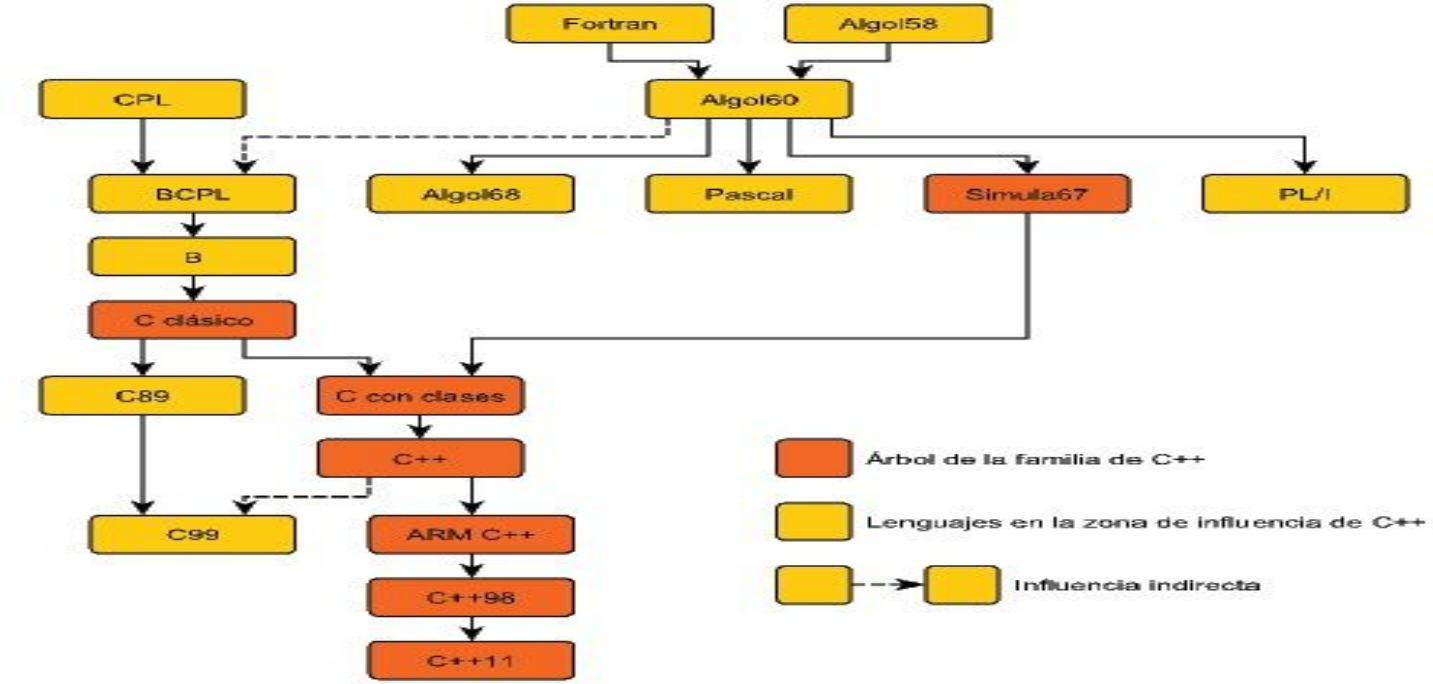
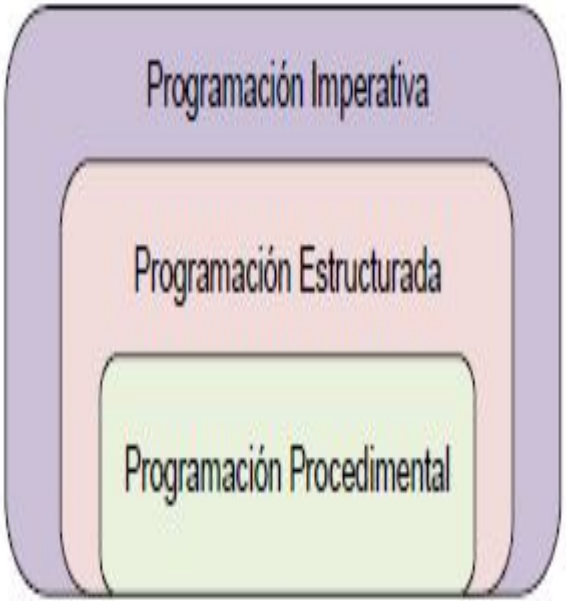


Figura 1. Genealogía del lenguaje de programación C++

Programación Procedimental

La programación por procedimientos es un paradigma de la programación. Muchas veces es aplicable tanto en lenguajes de programación de bajo nivel como en lenguajes de alto nivel. En el caso de que esta técnica se aplique en lenguajes de alto nivel, recibirá el nombre de Programación funcional. Esta técnica consiste en basarse en un número muy bajo de expresiones repetidas, englobadas todas en un procedimiento o función y llamarlo cada vez que tenga que ejecutarse.



Programación Orientada a Objetos.

La programación Orientada a objetos se define como un paradigma de la programación, una manera de programar específica, donde se organiza el código en unidades denominadas clases, de las cuales se crean objetos que se relacionan entre sí para conseguir los objetivos de las aplicaciones.

Podemos entender la programación Orientada a objetos (POO) como una forma especial de programar, más cercana a cómo expresaríamos las cosas en la vida real que otros tipos de programación, que permite diseñar mejor las aplicaciones, llegando a mayores cotas de complejidad, sin que el código se vuelva inmanejable.

Programación orientada a objetos.

Programación Genérica

La programación genérica está mucho más centrada en los algoritmos que en los datos y su postulado fundamental puede sintetizarse en una palabra: generalización. Significa que, en la medida de lo posible, los algoritmos deben ser parametrizados al máximo y expresados de la forma más independiente posible de detalles concretos, permitiendo así que puedan servir para la mayor variedad posible de tipos y estructuras de datos.

Por ejemplo, la idea de ordenación "Sort" se repite infinidad de veces en la programación, aunque los objetos a ordenar y los criterios de ordenación varíen de un caso a otro. Alrededor de esta idea surgió un nuevo paradigma denominado programación genérica o funcional.

Lenguaje de Programación

Podríamos definir apresuradamente que se trata de un conjunto de palabras que se organizan semánticamente para constituir un algoritmo.

Este lenguaje tiende a representar el lenguaje natural humano. Esto no siempre fue así.

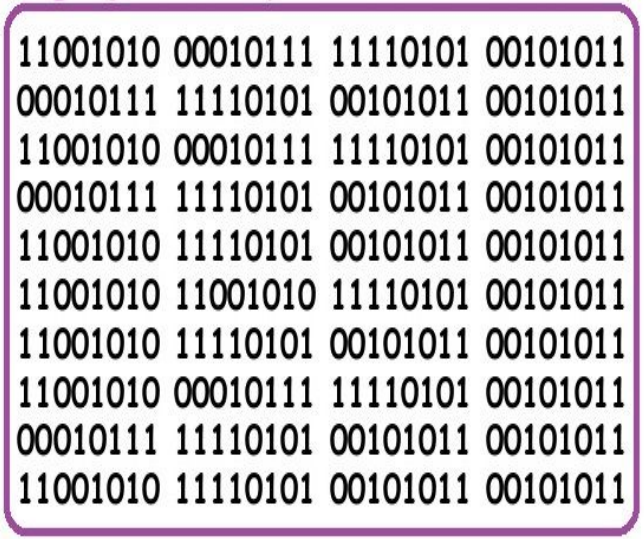
Podríamos indicar existen en un principio:

- Lenguaje Máquina
- Lenguaje Assembler o Bajo Nivel
- Lenguaje de Alto Nivel
- Lenguajes Gráficos.

Lenguaje Máquina

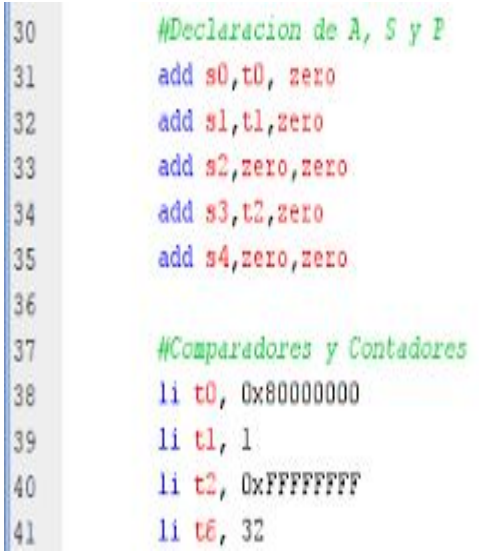
Es el lenguaje de programación más primitivo, ya que su funcionamiento se define a partir de mapas binarios o bits. En este sentido, el ordenador lee y reconoce secuencias numéricas de “0” y “1”, siendo éste, el lenguaje reconocido de manera nativa por cualquier computadora.

Lenguaje de máquina - tabla binaria.



Lenguaje Assembler o Bajo Nivel

Es un lenguaje de programación creado a partir de la necesidad de hacer el lenguaje de programación, algo más entendible y razonable para el ser humano. El lenguaje ensamblador es redactado y almacenado en forma de texto (al igual que como ocurre en los softwares de alto nivel). Además, el computador reconoce las instrucciones programadas a través de un procesador.



Lenguaje de Alto Nivel

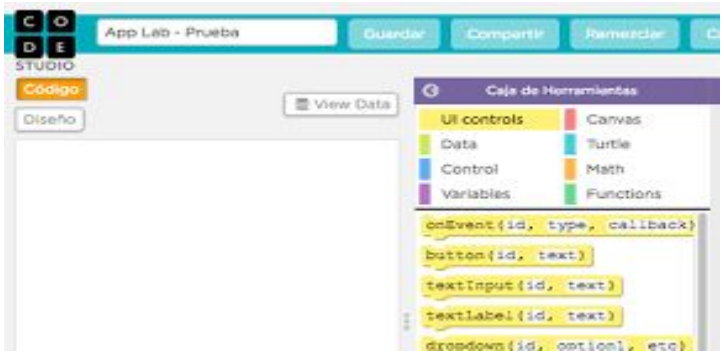
El lenguaje de alto nivel es el término empleado para referirse a cualquier lenguaje de programación cuyo objetivo sea ofrecer una serie de normas sintácticas y semánticas más sencillas de entender y escribir para los programadores.

```cpp
#include <iostream>
#include <stdio.h>  // Para sscanf
// https://parzibyte.me/blog/2019/06/19/namespaces-cpp-using-namespace/
using namespace std;

int main() {
  int numero = 0;
  const char *numeroComoCadena = "2811";
  // Lo convertimos usando sscanf, que
  // escanea datos a partir de una cadena
  sscanf(numeroComoCadena, "%d", &numero);
  std::cout << "El número es " << numero << " y el doble del mismo es "
            << numero * 2;
}
```

Lenguajes Gráficos.

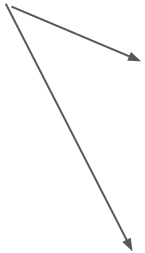
Desde hace un tiempo se incursiona en emplear elementos gráficos para constituir algoritmos. Esto permite que el programador quizás se desentienda de estar memorizando instrucciones. Emplear gráficos hace que sea más rápida la interpretación y más fácil su estudio.



Traducción

Las computadoras no reconocen el lenguaje humano. Es más, los Microprocesadores tienen un conjunto de instrucciones definidas por el fabricante llamado “Set de Instrucciones” y dependiendo de la arquitectura del Microprocesador tendrán distinto conjunto de instrucciones.

Arquitecturas

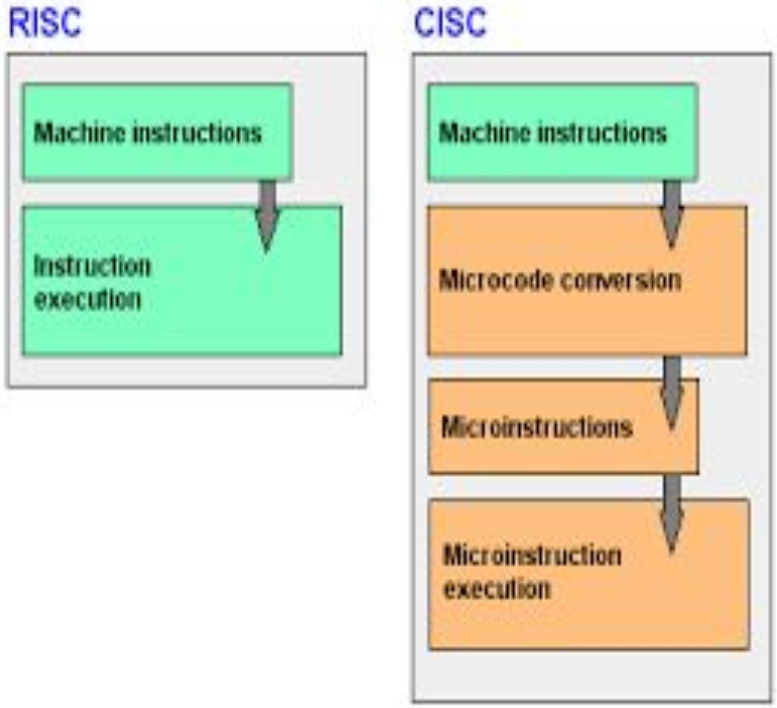


RISC(Equipos con un consumo mesurado de energía).

CISC(Equipos de gran consumo de energía).

CISC VS RISC

Cisc y Risc nota de voz



NXP s08 (CISC)	ARM Cortex M (RISC)

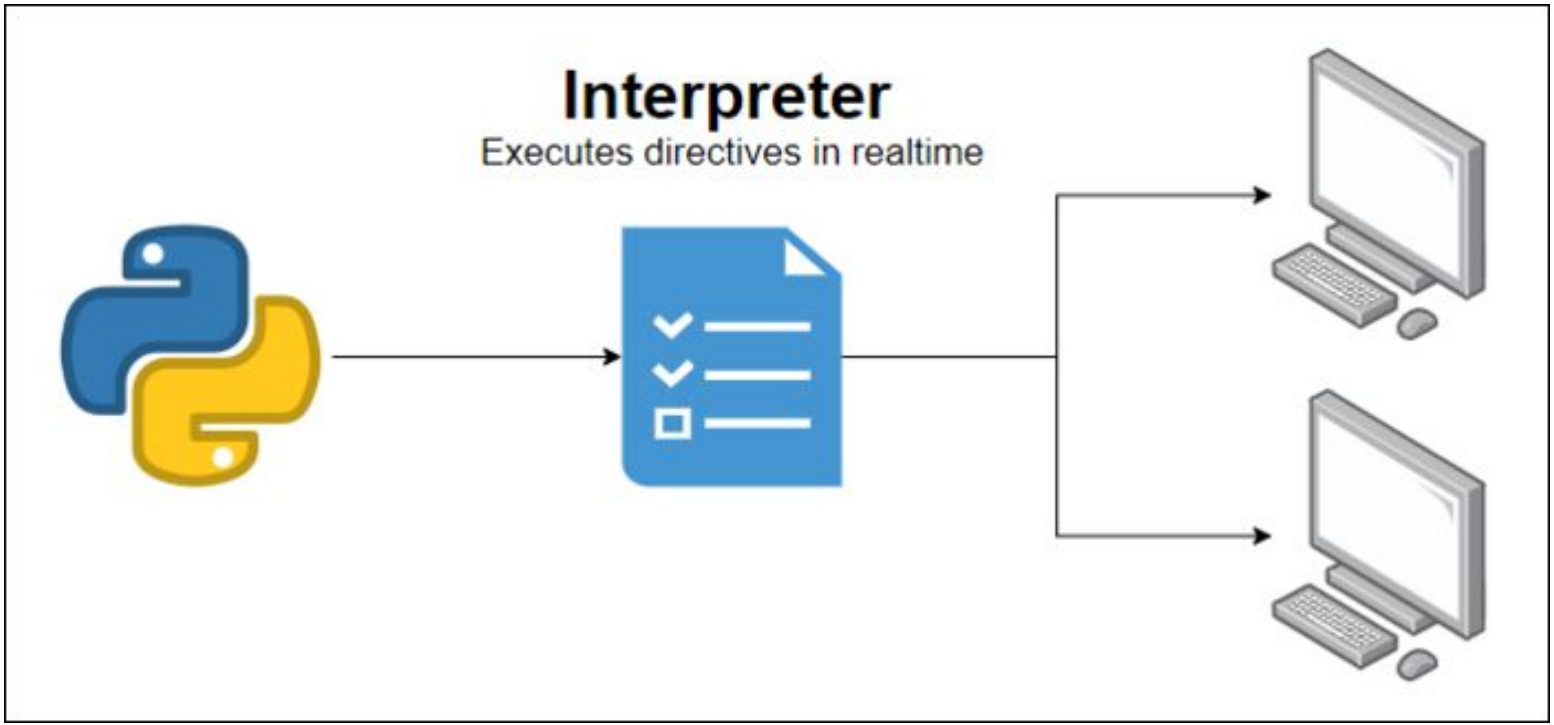
INC counter	LDR R0, =counter
	LDR R1, [R0]
	ADD R1, #1
	STR R1, [R0]

Interpretes VS Compiladores

Intérpretes

Un intérprete es un programa informático que procesa el código fuente de un proyecto de software durante su tiempo de ejecución, es decir, mientras el software se está ejecutando, y actúa como una interfaz entre ese proyecto y el procesador. Un intérprete siempre procesa el código línea por línea, de modo que lee, analiza y prepara cada secuencia de forma consecutiva para el procesador. Este principio también se aplica a las secuencias recurrentes, que se ejecutan de nuevo cada vez que vuelven a aparecer en el código. Para procesar el código fuente del software, el intérprete recurre a sus propias bibliotecas internas: en cuanto una línea de código fuente se ha traducido a los correspondientes comandos legibles por máquina, esta se envía directamente al procesador.

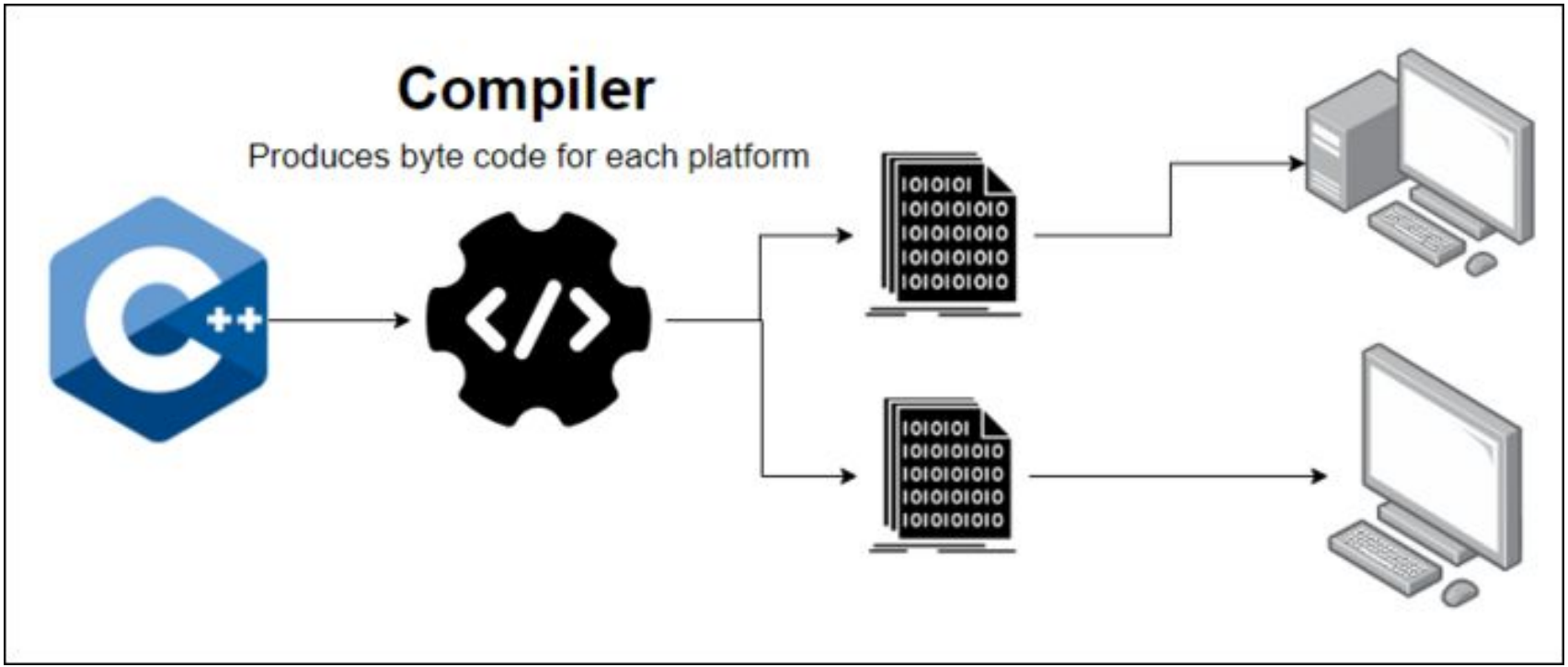
Intérpretes



Compiladores

Un compilador es un programa informático que traduce todo el código fuente de un proyecto de software a código máquina antes de ejecutarlo. Solo entonces el procesador ejecuta el software, obteniendo todas las instrucciones en código máquina antes de comenzar. De esta manera, el procesador cuenta con todos los componentes necesarios para ejecutar el software, procesar las entradas y generar los resultados. No obstante, en muchos casos, durante el proceso de compilación tiene lugar un paso intermedio fundamental: antes de generar la traducción final en código máquina, la mayoría de los compiladores suelen convertir el código fuente en un código intermedio (también llamado código objeto) que, a menudo, es compatible con diversas plataformas y que, además, también puede ser utilizado por un intérprete.

Compiladores



Compilador e intérprete: diferencias, en resumen

	Intérprete	Compilador
Momento en que se traduce el código fuente	Durante el tiempo de ejecución del software	Antes de ejecutar el software
Procedimiento de traducción	Línea por línea	Siempre todo el código
Presentación de errores de código	Después de cada línea	En conjunto, después de toda la compilación
Velocidad de traducción	Alta	Baja
Eficiencia de traducción	Baja	Alta
Coste de desarrollo	Bajo	Alto
Lenguajes típicos	PHP, Perl, Python, Ruby, BASIC	C, C++, Pascal

Fases de la compilación

Programa Fuente	Compilador	Programa Objeto	Montador	Programa Ejecutable
Codificación del programador en el IDE integrado. Nace el archivo Fuente.	Traducción de Archivo Fuente en Archivo Objeto	Este es un estado intermedio que permite obtener una sintaxis estudiada y corregida por el compilador	Montador o Enlazador(linker). Genera el archivo que interpreta la máquina.	Archivo de Ejecución.

Bibliografía.

https://sites.google.com/site/fernandoagomezf/programacion-en-c/tips-de-programador-c/editorial-de-c/introduccion-a-c

https://recluit.com/cual-es-la-historia-de-c/#.YsSVhnbMLrc

http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro32/112_programacin_procedimental.html

https://desarrolloweb.com/articulos/499.php

https://liukin.es/que-es-la-compilacion-just-in-time-jit-cloudsavvy-it/

https://conceptoabc.com/lenguaje-de-programacion/

http://aulavirtualrctdr.blogspot.com/2016/02/historia-breve-del-lenguaje-c.html