Generalidades de la programación en C++

1967: Fue desarrollado el lenguaje BCPL para escribir sistemas operativos y compiladores (Martin Richards).

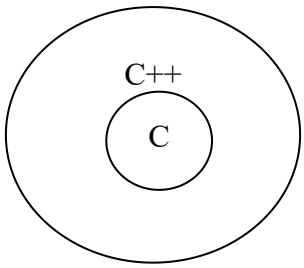
1970: BCPL fue retomado por Ken Thompson para escribir el lenguaje B, con el que desarrollaron las primeras versiones del sistema operativo UNIX, en los Laboratorios Bell (creados en 1925).

1972: Dennis Ritchie desarrolló, a partir del lenguaje B, el lenguaje C en los Laboratorios Bell. C es un lenguaje ampliamente utilizado para desarrollar software de sistemas operativos y compiladores, aunque también se utiliza para desarrollar aplicaciones (hojas electrónicas, gestores de bases de datos, editores, programas de manipulación de gráficos, etc.). Se utiliza para escribir programas bajo el paradigma de la programación estructurada. Los nombres de los archivos que contienen programas escritos en este lenguaje terminan con la extensión .c. Se convirtió en un estándar ANSI/ISO en 1990. Se han realizado más revisiones del estándar, como la de 1999, para adaptarlo a las exigencias del nuevo hardware, como por ejemplo, las computadoras de 64 bits.

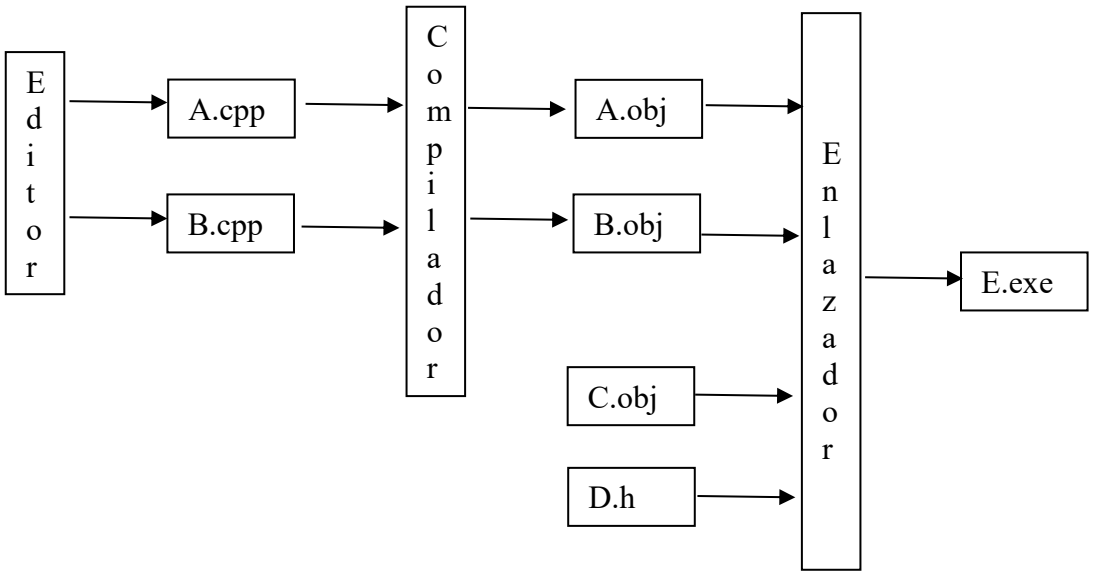
El lenguaje C vino a sustituir a otros lenguajes más didácticos como Pascal, creado por Niklaus Wirth, en Suiza. Pascal fue inicialmente creado para la enseñanza de la programación estructurada, pero rápidamente trascendió a otros ámbitos. La ventaja del lenguaje C sobre Pascal fue que permitía llegar a ciertas estructuras internas del hardware y que los programas ejecutables generados luego del proceso de compilación eran más pequeños. Esto le valió su aceptación, a pesar de que la elaboración de programas era menos didáctica que la de Pascal.

Una de las ventajas que mostró el lenguaje C es que sus funciones predefinidas venían clasificadas en librarías. Es decir, venían agrupadas en archivos, de acuerdo a la naturaleza de las tareas que resolvían. De esta forma el programador podía decidir qué librerías incluir en sus programas, lo que permitía generar programas ejecutables más pequeños luego del proceso de compilación. En contraste, los compiladores de otros lenguajes incluían todas las funciones del lenguaje durante el proceso de compilación, aunque no se fueran a necesitar, lo cual generaba programas ejecutables demasiado grandes. Esto, para la cantidad de RAM disponible en las computadoras de esa época era un factor crítico. Ejemplos de algunas librerías tradicionales de C bastante utilizadas, son: math.h, stdio.h, stdlib.h, time.h, string.h, etc. Las librerías en C++ son colecciones de clases y funciones, algunas de estas son: iostream, cmath, string, vector, fstream, complex.

Inicios de 1980: Bjarne Stroustrup desarrolló una extensión de C, a la que llamó **C++** (++ hace referencia a "un incremento de C"). Entre otras cosas, proporciona la capacidad de programar orientado a objetos. En realidad, un programa en C++ puede incluir características únicamente de programación estructurada o de ambos paradigmas, por lo que es multiparadigma, o, más bien, biparadigma, o híbrido. Los nombres de los archivos que contienen programas escritos en C++ generalmente terminan con la extensión .cpp y pueden contener programación C pura.



Existen muchos compiladores y entornos de desarrollo de programas en C++. Todos ellos generan programas ejecutables cuyo contenido son instrucciones en lenguaje de máquina, en código binario, que pueden ejecutarse directamente desde la línea de comandos del sistema operativo (la consola de Ubuntu o la ventana de comandos de Windows) o desde una ventana de un explorador del sistema, como Windows Explorer. El proceso de generación de un programa ejecutable en C++ puede verse como sigue:



Como puede verse en la figura, con ayuda de un editor, nosotros escribimos nuestros programas, luego los hacemos pasar por un proceso que se llama compilación, que revisa si los programas están bien escritos y cumplen las reglas impuestas por el lenguaje de programación. Si todo va bien, el compilador traduce a lenguaje de máquina nuestros programas. Luego, estos programas traducidos pasan por un proceso de enlazamiento que, entre otras cosas, unifica los diferentes módulos del programa y las librerías pre elaboradas del lenguaje. Así, se crea el programa ejecutable que será el que la computadora podrá ejectuar.

Antiguamente todos estos procesos se llevaban a cabo paso a paso, aun ahora puede hacerse así, pero los entornos de desarrollo, que antes no existían y que proveían funcionalidades mínimas en sus inicios, han evolucionado a través de las décadas y actualmente permiten hacer todos los procesos descritos, y más, de forma muy sencilla y aun desapercibida.

Hoy en día los entornos de desarrollo proveen de ventanas para realizar corridas del programa sin necesidad de abandonar el entorno, o de ventanas para realizar procedimientos de depuración del programa que se está creando, o de la habilidad de mostrar en diferentes colores los distintos tipos de elementos de un programa, etc. Pero, finalmente, en el caso de C/C++ lo que se creará es un programa ejecutable compilado.

Existen otro tipo de lenguajes y herramientas que vienen ligados a sus entornos de desarrollo, estos son, por lo general, lenguajes interpretados: interpretan y ejecutan línea por línea. Ejemplos de estos lenguajes, son: Prolog, Racket, Python, Octave, Matlab y SciLab. Algunos de estos entornos, con el tiempo, han desarrollado la opción de generar ejecutables, para correr el programa independientemente del entorno de desarrollo.

Existe otro tipo de lenguajes para el que su entorno de desarrollo genera cierto tipo de código intermedio, que para correr fuera del entorno de desarrollo necesita de una máquina o motor que lo interprete y ejecute, y que debe esta instalada en la computadora. Un ejemplo de estos es Java, con su java virtual machine.

Por lo demás, existen lenguajes y entornos que proveen de varias de las facilidades de las que se ha hablado arriba. Sobre lenguajes y desarrollo nunca se dirá la última palabra.