# Clase 1: Introducción a C

Mauricio Gómez García

# Índice

# Introducción al curso

El objetivo de este curso es enseñar la lógica de programación básica y necesaria para poder utilizar el lenguaje de programación C, sin embargo muchas de las metodologías aplicadas aquí se podrán utilizar en otros lenguajes de igual manera. Con la información que se proveerá dentro de este curso se busca dar a conocer la manera en la que se trabaja con el lenguaje y las mejores prácticas al momento de realizar código en C.

Adicional, se busca enseñar las bases del lenguaje y la manera en la que funciona para así poder entender mejor cómo funciona el lenguaje con mayor profundidad. C es una herramienta muy útil, por lo que entender su diseño y función es necesario para poder aprovechar su utilidad al máximo.

Cada sesión se estructurará en secciones, los cuales tendrán sus propios temas e información, y los documentos (como el actual) se proveerán para que los interesados puedan revisar la información y fuentes a más detalle. En caso necesario, un breve receso o descanso se podrá dar entre secciones, dependiendo de cada sesión.

Es importante notar que durante el curso será necesario utilizar la línea de comandos para realizar varias acciones, por lo que tener permisos adecuados en la máquina será necesario para poder seguir el curso.

# Sección 1

La primera sección se enfocará en datos generales acerca del lenguaje de programación, su historia, características principales y la instalación de un entorno en donde se podrá editar, compilar y ejecutar código en C. Se proveerán fuentes de información en casos que sean relevantes.

## Objetivo

Esta sesión tiene como objetivo dar a conocer la información básica del lenguaje de programación C, una breve historia de cómo se desarrolló, cómo empezar a trabajar en el, y un análisis de un programa básico realizado en C, para conocer su funcionalidad a grandes rasgos. En sesiones futuras se darán a conocer más a detalle las funciones, métodos y otros datos importantes del lenguaje.

Se tomará un enfoque particular en configurar un entorno para la compilación y ejecución de código, asumiendo que el interesado no cuente con ningún entorno prévio a este curso. Esto es para poder ofrecer una configuración básica que permita trabajar con el lenguaje y facilitar conocer la herramienta, pero es importante notar que existen otros flujos de trabajo para desarrollo de software en C.

# Introducción

C es un lenguaje de programación multi-propósito utilizado en muchos ambientes y para muchos fines, y dada su versatilidad es uno de los lenguajes más populares actualmente[1].

Los lenguajes de programación se pueden categorizar en dos grupos grandes: los lenguajes compilados y los interpretados. C es un lenguaje compilado, lo que significa que es necesario primero compilar el código fuente y crear un archivo ejecutable ántes de poder utilizar la programación realizada[2].

Otro dato importante a conocer de C es que es un lenguaje de más bajo nivel a comparación de otros lenguajes como Python o Java. Esto significa que su función es más cercana a la estructura de instrucciones que interpreta la computadora, y tiene accesos a hardware más sencillos que lenguajes de más alto nivel[3]. Esto significa que implementar métodos y funciones puede resultar más eficiente, pero también implica cierto riesgo si la programación se realiza de manera incorrecta.

Por último, es importante conocer que el lenguaje tiene una orientación de programación funcional, en contraste con la orientación a objetos. Esto significa que el paradigma de programación se centra en crear funciones y conexiones entre dichas, enfocándose en el procesamiento de datos entre las funciones para lograr los objetivos deseados[4].

---

[1] C (Programming Language): https://en.wikipedia.org/wiki/C_(programming_language)

[2] Features and Characteristics of Compiled Languages: https://www.sqa.org.uk/e-learning/ClientSide01CD/page_14.htm

[3] What is a Low Level Language?: https://www.geeksforgeeks.org/what-is-a-low-level-language/

[4] Functional Programming: https://en.wikipedia.org/wiki/Functional_programming

## Historia

El lenguaje nace en conjunto con el sistema operativo Unix, y es el sucesor al lenguaje B. Tal como su predecesor, fué inicialmente desarrollado por Bell Labs para uso en Unix, y a partir de su implementación en dicho sistema operativo se fué cimentado como el estandar para el desarrollo de software[5].

C nace dada la necesidad de la implementación de nuevas funciones, metodologías y maneras de guardar información al momento de manejar la programación, los cuales no estaban disponibles en B. Dennis Ritchie y Ken Thompson fuerons los primeros en trabajar en el desarrollo del lenguaje[6].

Actualmente, el estandar para el lenguaje se mantiene en ISO/IEC 9899:2024, la cual coloquialmente se conoce como C23[7]. Existen otros estándares, la más popular siendo C99, la cuál se estableció en 1990. Los estándares para el lenguaje dictaminan ciertos paradigmas, métodos y tipos de dato que se pueden manejar utilizando el lenguaje.

[5] C (Programming Language) History: https://en.wikipedia.org/wiki/C_(programming_language)#History

[6] C (Programming Language) History: https://en.wikipedia.org/wiki/C_(programming_language)#History

[7] C23: https://en.wikipedia.org/wiki/C23_(C_standard_revision)

## Herramientas

En este apartado se verán las herramientas con las que se estarán trabajando durante el curso. Dichas herramientas dependerán del sistema operativo en el que el interesado esté trabajando.

La herramienta más importante con la que se trabajará será el compilador, el cual nos sirve para crear los ejecutables a partir de nuestro código fuente. Será importante instalar un compilador para el sistema operativo en la cual el interesado trabaje. En todo caso, se trabajará desde la línea de comandos, pero habrán diferencias ligeras dependiendo del sistema en donde se opere.

En caso posible, se preferirá trabajar en un entorno basado en Linux, dado que es más fácil realizar el proceso de instalación, compilación y ejecución de código que en otros sistemas como Windows o MacOS.

Adicional al compilador, será necesario un editor de texto o IDE, el cual queda a discreción del interesado para elegir. Únicamente será importante que el editor pueda modificar texto, para poder escribir y editar el código en el cual trabajaremos, ya que la compilación se realizará manualmente mediante la línea de comandos.

Siguiente se enlistan algunos editores disponibles para algunos sistemas operativos:

- *Linux*:
  - Kate
  - Nano
  - Vim
- *Windows*:
  - Notepad++
  - Visual Studio Code (con extensiones para C)
- *MacOS*
  - Xcode
  - TextEdit
  - Vim

Siguiente, se revisarán compiladores para cada sistema operativo.

### Linux

Las distribuciones de Linux usualmente ofrecen un compilador de C en sus repositorios oficiales. El compilador más popular para las plataformas basadas en Linux es GNU Compiler Collection (GCC)[8]. Dependiendo de la distribución específica, el método de instalación será distinta. Siguiente se comparten los métodos de instalación para las distribuciones más populares:

*Debian/Ubuntu y derivados*[9]

```
sudo apt update
sudo apt install build-essential
```

*Fedora y derivados*[10]

```
sudo dnf install gcc
```

*Arch y derivados*[11]

```
sudo pacman -Sy gcc
```

[8] GNU Compiler Collection: https://en.wikipedia.org/wiki/GNU_Compiler_Collection
[9] Installing GCC: https://shape.host/resources/debian-12-installing-gcc-compiler-detailed-walkthrough
[10] C Installation and Usage: https://developer.fedoraproject.org/tech/languages/c/c_installation.html
[11] Arch Wiki: GNU Compiler Collection: https://wiki.archlinux.org/title/GNU_Compiler_Collection

### Windows

Existen múltiples herramientas para compilar el lenguaje C para Windows. Aquí se instalará Cygwin[12], ya que es una herramienta de software libre y su instalación es más sencillo que varias de sus contrapartes.

Será necesario visitar la página oficial de Cygwin para poder descargar la herramienta de instalación. Una vez descargada la herramienta de instalación (disponible aquí) será necesario ejecutar el programa para instalar la herramienta. Esto se puede lograr simplemente dando doble click en el archivo desde el navegador de archivos.

Al ejecutar el programa, será necesario seleccionar las herramientas con las que estaremos trabajando, específicamente será necesario seleccionar mínimo `gcc`, el cual usaremos para compilar nuestro código. Una vez instalada la herramienta, se podrá utilizar desde la línea de comandos.

[12] Cygwin: https://cygwin.com/

### MacOS

MacOS ofrece Xcode[13], la herramienta proprietaria de Apple para desarrollo en en su sistema operativo. Dicha herramienta será necesaria para poder realizar compilación de código C en el sistema. Para instalarlo, será necesario abrir la terminal y ejecutar el siguiente comando:

```
xcode-select --install
```

Una vez instalado Xcode, será necesario habilitar herramientas de línea de comando desde la aplicación. Desde el menú, será necesario abrir “Preferencias”, y seleccionar el pánel de “Descargas”. Una vez se esté en esta sección, seleccionar “Herramientas de línea de comandos” y se instalarán.

[13] Xcode: https://developer.apple.com/xcode/

# Sección 2

Durante esta sección se revisará un ejemplo básico de código en C, se diseccionarán sus partes principales y se verán algunos conceptos básicos acerca del lenguaje. Se entrará en detalle a cada una de las partes vistas en sesiones futuras, ya que el objetivo de la sesión es observar la funcionalidad general del lenguaje y código.

## Código

Siguiente se comparte un ejemplo de código realizado en C. Éste código tiene como funcionalidad única imprimir texto en la pantalla, y es uno de los ejemplos más básicos que existen en cualquier lenguaje de programación.

```c
#include <stdio.h>

int main() {
    printf("Hola, Mundo!\n");
    printf("Este es mi primer programa hecho en C\n");
    return 0;
}
```

## Análisis

El código que hemos visualizado préviamente cuenta con varios componentes, los cuales revisaremos ahora. En primer lugar, es importante conocer que cada línea cuenta con una “declaración”, lo cual indica una instrucción que realizará el código. Cada declaración se separa por un punto y coma (;), y es una buena práctica escribir una declaración por línea, sin embargo C permite tener varias declaraciones por línea.

```
#include <stdio.h>
```

Esta línea es importante, ya que estamos declarando el uso de la libreria `<stdio.h>`. La declaración `#include` es una declaración especial de compilador, el cual establece qué librerias, cabeceras u objetos externos se utilizarán. Este concepto es extenso, por lo que, por el momento, únicamente es necesario conocer que `stdio.h` es la librería estandar para entrada y salida de datos para el programa.

```
int main() {}
```

Esta línea declara una función de nombre `main`. Las funciones nos permiten establecer series de instrucciones a seguir dentro de nuestro programa, y es *obligatorio* contar con la función `main`, ya que nos indica la función principal a ejecutar dentro de nuestro programa. Es posible establecer múltiples funciones, y se pueden utilizar más de una vez en un programa.

Prévio al nombre de la función, la palabra clave `int` indica el tipo de dato que devolverá la función. Las funciones usualmente devuelven un valor (con excepción a las funciones de tipo `void`, las cuales no devuelven ningún valor), y es necesario declarar el tipo de valor que devolverá, independientemente si se utilizará dicho dato o no. Esta palabra clave se conoce como un “datatype”, o tipo de dato. Existen muchos tipos de datos, los cuales se verán más a profundidad en sesiones futuras.

Los paréntesis `()` indican los parámetros que utilizará la función. Los parámetros de una función son valores que se utilizarán en conjunto con las instrucciones que ejecutará la función, por lo que es necesario establecer qué parámetros (si es que hay algunos) dentro de los paréntesis. En este caso, no hay ningún parámetro de entrada, por lo que los paréntesis quedan en blanco.

Los corchetes `{}` indican el inicio y fin del cuerpo de la función, y entre ellas se encuentran todas las instrucciones que se ejecutarán al momento de llamar a la función.

```
printf("Hola, Mundo!\n");
printf("Este es mi primer programa hecho en C\n");
```

Éstas líneas son las instrucciones que realiza el programa. Más específicamente, son las instrucciones que se ejecutan al llamar a la función `main()`. Como se mencionó préviamente, cada instrucción o declaración se debe terminar por un punto y comma `;`.

La declaración `printf()` es una función declarada dentro de la librería `<stdio.h>` que hemos importado préviamente, y nos permite imprimir texto en la consola. El contenido de los paréntesis indica los parámetros a utilizar, y en este caso tenemos cadenas de texto (`string`):

- `"Hola, Mundo!\n"`
- `"Este es mi primer programa hecho en C\n"`

La función entonces toma estos parámetros y ejecuta las instrucciones que se les ha asignado préviamente, las cuales están dentro de la librería que hemos importado, con la finalidad de imprimir el texto en la consola.

```
return 0;
```

Esta instrucción indica el valor que devolverá la función. Como se mencionó préviamente, las funciones devuelven datos, y es importante indicar el valor que se devolverá al terminar su ejecución. En este caso, al terminar de ejecutar el programa se devolverá un valór numérico `0`. Es posible devolver datos distintos, dependiendo del flujo de instrucciones definidas en la función.

Es importante notar que las instrucciones se ejecutan secuencialmente, es decir, se ejecutan en el orden en el que están escritas. Esto hace necesario indicar los pasos en el orden en el que se desean realizar.

## Ejecución

Ya que se ha analizado el código, procederemos a compilar y ejecutarlo para observar su funcionamiento.

Se recomienda crear un directorio o carpeta en donde trabajaremos, con el objetivo de no modificar datos o archivos externos al ejemplo con el cual trabajaremos.

En primer lugar, crearemos el archivo que se compilará. Desde su editor de texto de preferencia, cree un archivo de nombre `fuente.c`. Dentro del archivo inserte el código fuente que se ha compartido préviamente y guarde el archivo.

Una vez creado el archivo, será necesario trabajar en la línea de comandos para compilar y ejecutar el programa. Para empezar, será necesario cambiar de directorio, para así trabajar dentro del cual creamos nuestro archivo.

```
cd [Ruta al directorio]
```

Al cambiar al directorio, podremos utilizar nuestro compilador (que hemos instalado préviamente) para compilar nuestro código fuente y después ejecutarlo:

```
cc fuente.c -o ejecutable # MacOS
gcc fuente.c -o ejecutable # Linux
gcc fuente.c -o ejecutable.exe # Windows
```

Este comando utiliza GCC para compilar el archivo creado préviamente (**`fuente.c`**), y utilizando la bandera **`-o`** establecemos el nombre del archivo de salida (**`ejecutable`**).

Una vez que se ha compilado el ejecutable, podremos correr el programa para observar la salida:

```
./ejecutable # Linux y MacOS
./ejecutable.exe # Windows
```

Al ejecutar nuestro código, la salida debe ser la siguiente:

```
Hola, Mundo!
Este es mi primer programa hecho en C
```

# Conclusión

El lenguaje de programación C es muy versátil, y conocer a profundidad cómo funciona y la manera en la que el lenguaje trabaja es necesario para poder crear programas funcionales, para la finalidad que requiera el desarrollador.

Existen muchos casos de uso y puntos importantes a conocer al momento de desarrollar herramientas con C, pero tener bases sólidas de cómo funciona el lenguaje, la lógica de programación y cómo utilizar herramientas para desarrollar software en C son escenciales para cualquier desarrollador de software que esté interesado en el lenguaje.

Como se demostró con el ejemplo de código, existen muchos componentes que interactúan entre sí, y conocer su función y sintaxis es importante para crear programas funcionales y aprovechar al máximo el lenguaje de programación. Las funciones, librerias, declaraciones y demás componentes tienen varias interacciones, las cuales profundizaremos en sesiones futuras.

Como tarea opcional para la siguiente sesión, intente crear su propio programa que imprima más texto y líneas, e investigue qué significa la cadena `"\n"` dentro del código que se demostró.