
	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	1/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Guía práctica de estudio 06: Entorno y fundamentos del lenguaje C



Elaborado por	Actualizado por:	Revisado por:
Ing. Jorge A. Solano Gálvez M.C. Laura Sandoval Montaño	Ing. Julio De León Razo M.T. Hugo Zúñiga Barragán	M.C. Cintia Quezada Reyes M.C. Laura Sandoval Montaño Ing. Jorge A. Solano Gálvez

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	2/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Guía práctica de estudio 06: Entorno y fundamentos del lenguaje C

Objetivo:

El alumno elaborará programas en lenguaje C utilizando las instrucciones de control de tipo *secuencia*, para realizar la declaración de variables de diferentes tipos de datos, así como efectuar llamadas a funciones externas de entrada y salida para asignar y mostrar valores de variables y expresiones.

Actividades:


- Crear un archivo de texto (utilizando algún editor) y escribir un programa en lenguaje C que contenga variables de diferentes tipos, asignación de valores (por lectura desde la entrada estándar o asignación directa) y escritura del valor de las variables en la salida estándar
- Compilar un código fuente y ejecutarlo.
- Modificar y actualizar un programa usando un editor.
- Elaborar expresiones relacionales/lógicas en un programa en C y mostrar el resultado de su evaluación.

Introducción

Una vez que un problema dado ha sido analizado (se identifican los datos de entrada y la salida deseada), que se ha diseñado un algoritmo que lo resuelva de manera eficiente (procesamiento de datos), y que se ha representado el algoritmo de manera gráfica o escrita (diagrama de flujo o pseudocódigo) se puede proceder a la etapa de codificación.

La codificación se puede realizar en cualquier lenguaje de programación estructurada. En este curso se aprenderá el uso del lenguaje de programación C.

Dentro del ciclo de vida del software, la implementación de un algoritmo se encuentra en la etapa de *codificación* del problema. (Figura 1)

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	3/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

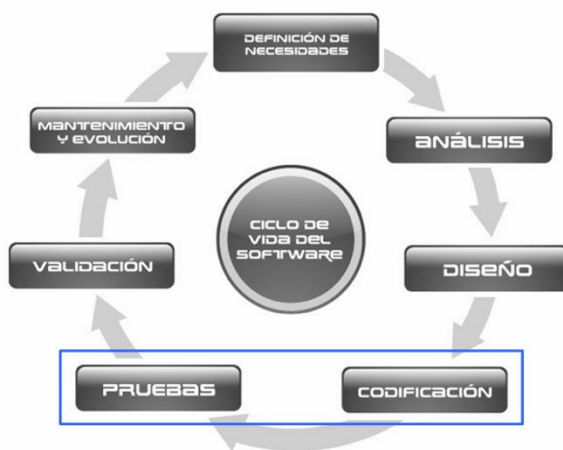



Figura 1: Ciclo de vida del software.

Entorno de C

Un lenguaje de programación permite expresar una serie de instrucciones que podrán ser realizadas por una computadora. Uno de los lenguajes de programación mayormente difundidos es el lenguaje C.

Una característica importante del lenguaje C es que es muy poderoso ya que combina las características de un lenguaje de alto nivel (facilidad de programación), con uno de bajo nivel (manejo más preciso de una máquina); por lo que se han creado variantes que permiten programar miles de dispositivos electrónicos en el mundo con sus respectivos compiladores.

Un programa en C se elabora describiendo cada una de las instrucciones de acuerdo con las reglas definidas en este lenguaje en un archivo de texto para después ser procesadas en un compilador. Un compilador es un programa que toma como entrada un archivo de texto y tiene como salida un programa ejecutable, éste tiene instrucciones que pueden ser procesadas por el hardware de la computadora en conjunto con el sistema operativo que corre sobre ella. Se tiene como ventaja que un programa escrito en lenguaje C, siguiendo siempre su estándar, puede ejecutarse en cualquier máquina siempre y cuando exista un compilador de C. A esto también se le conoce como multiplataforma.

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	4/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Editores

Un programa en C debe ser escrito en un editor de texto para después generar un programa ejecutable en la computadora por medio de un compilador. Tanto el editor de texto como el compilador van de la mano con el sistema operativo y si posee o no interfaz gráfica, por lo que son factores que se deben tomar en cuenta a la hora de elegir el entorno para desarrollar programas en C.

Es importante señalar que no es lo mismo un editor de texto que un procesador de texto. El primero edita un texto plano que puede tener muchas utilidades como guardar una configuración, tener escrito un programa, etcétera, y será interpretado hasta que se haga una lectura de éste. Un procesador de texto permite dar formato al texto, a la hoja donde está escrito, incrustar imágenes, entre otros, su salida puede ser un archivo de texto plano que contiene etiquetas que señalan el formato que se le dio al texto o algo un poco más complejo. A continuación, se presentan algunos de los editores más comunes.


Editor Visual Interface de GNU/Linux (vi)

El editor *vi* (visual interface) es el editor más común en cualquier distribución de sistemas operativos con núcleo basado en UNIX. Está disponible en línea de comandos y si el sistema operativo tiene entorno gráfico se puede acceder a él desde *la terminal*. *vi* es un editor que puede resultar difícil de usar en un inicio. Aunque existen editores más intuitivos en su uso; en muchas ocasiones *vi* es el único disponible.

Para iniciar *vi*, debe teclearse desde la línea de comandos:

```
vi nombre_archivo[.ext]
```

Donde *nombre_archivo* es el nombre del archivo a editar o el nombre de un archivo nuevo que se creará con *vi*, y *[.ext]* se refiere a la *extensión* que indica que el texto es una programa escrito en algún lenguaje o es texto plano, por ejemplo. Es válido incluir la ruta donde se localiza o localizará el archivo. Existen más métodos de apertura para usuarios más avanzados.

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	5/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Modo comando (Figura 2)

Es el modo por defecto de *vi* cuando se abre. Las teclas presionadas ejecutan diversas acciones predeterminadas y no se puede editar el texto libremente. Los comandos son sensitivos a las mayúsculas y a las minúsculas. Algunos ejemplos son:

- \uparrow o *k* mueve el cursor hacia arriba.
- \downarrow o *j* mueve el cursor hacia abajo.
- \leftarrow o *h* mueve el cursor hacia la izquierda.
- \rightarrow o *l* mueve el cursor hacia la derecha.
- *1G* lleva el cursor al comienzo de la primera línea.
- *G* lleva el cursor al comienzo de la última línea.
- *x* borra el carácter marcado por el cursor.
- *dd* borra o corta la línea donde está el cursor.
- *ndd* donde *n* es la cantidad de líneas que se borrarán o cortarán después del cursor.
- *D* borra o corta desde la posición de cursor hasta el final de la línea.
- *dw* borra o corta desde la posición del cursor hasta el final de una palabra.
- *yy* copia la línea donde está el cursor.
- *p* pega un contenido copiado o borrado.
- *u* deshace el último cambio.

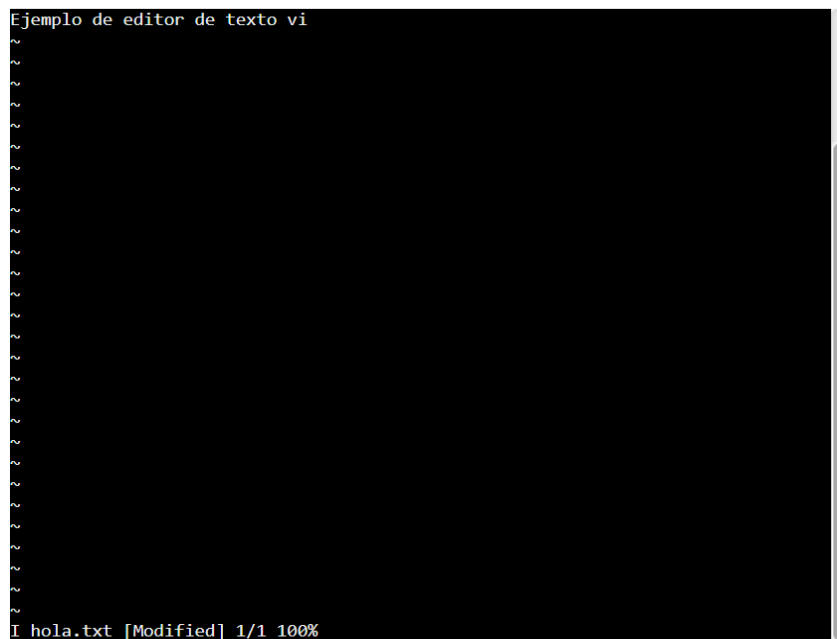



Figura 2: *vi* en modo comando.

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	6/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Modo de última línea

Se puede acceder a él utilizando el modo comando pero los comandos no tendrán efecto hasta que se presiona la tecla Enter además de que se visualizará el comando en la última línea del editor. Es posible cancelar el comando con la tecla Esc. Los comandos de última línea se caracterizan porque inician con /, ? o :. Algunos ejemplos son:

- **/texto** donde la cadena **texto** será buscada hacia delante de donde se encuentra el cursor.
- **?texto** donde la cadena **texto** será buscada hacia atrás de donde se encuentra el cursor.
- **:q** para salir de *vi* sin haber editado el texto desde la última vez que se guardó.
- **:q!** para salir de *vi* sin guardar los cambios.
- **:w** para guardar los cambios sin salir de *vi*.
- **:w archivo** para realizar la orden “guardar como”, siendo **archivo** el nombre donde se guardará el documento.
- **:wq** guarda los cambios y sale de *vi*.

GNU nano

Es un editor de texto disponible para sistemas operativos basados en UNIX en línea de comandos. Se puede acceder en un entorno gráfico desde la aplicación de terminal. Este editor es mucho más intuitivo que *vi*, aunque menos potente. No es necesario saber cómo se utiliza ya que proporciona una interfaz que describe los comandos básicos. (Figura 3)


nano es un editor clon de otro editor llamado pico.

Para iniciar nano, debe teclearse desde la línea de comandos:

```
nano nombre_archivo[.ext]
```

Donde “*nombre_archivo*” es el nombre del archivo a editar o el nombre de un archivo nuevo.

Una vez en el editor, en la parte inferior se pueden observar los comandos básicos. Si se presiona la tecla **F1** es posible visualizar la ayuda con la lista de todos comandos que existen.

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	7/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Los atajos de teclado pueden corresponder a:

- ^ que es la tecla **Ctrl**.
- M- que es la tecla **Esc** o bien **Alt**.

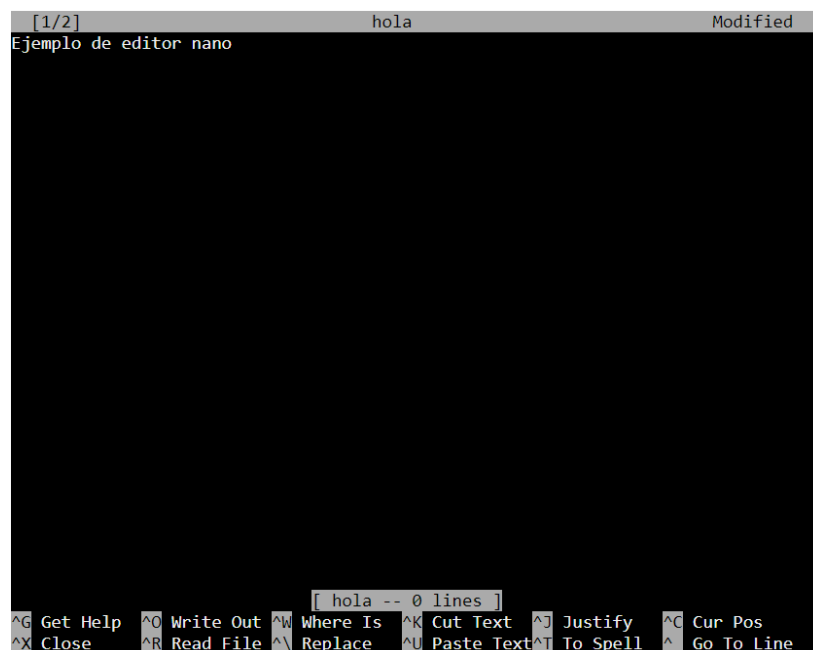


Figura 3: El editor de texto nano.


Otros

Existen otros editores actualmente como: Visual Studio Code, Atom, Sublime Text, Notepad, entre otros.

Compiladores

Una vez codificado un programa en C en algún editor de texto, éste debe ser leído por un programa que produzca un archivo ejecutable. A este programa se le conoce como compilador. Para el caso del lenguaje C el compilador traduce el código fuente del programa a código ejecutable.

Un programa en C tampoco puede ser escrito de manera arbitraria debe respetar una serie de reglas para que el compilador pueda entenderlas y realizar su función. Un

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	8/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

estándar muy común es **ANSI C** y existen diferentes extensiones como ISOC99 y GNU C que representan mejoras para el estándar original. Realizar un programa en dicho estándar garantiza que puede ejecutarse en cualquier máquina siempre y cuando exista un compilador hecho para ella. A veces, el programador no sigue un estándar o lo desconoce, usando características no estándar que, a la hora de usar el mismo programa para otra máquina no funciona, teniendo que realizar adaptaciones que se reflejan en costos. Por ejemplo, es muy común empezar a desarrollar programas en C en plataforma Windows en procesadores x86 usando características propias. Al trasladar, por alguna necesidad, dicho programa a plataforma GNU/Linux con procesador ARM, el programa no funcionará porque no se siguió el estándar que garantiza universalidad para el lenguaje C.


Es muy común cometer algún error al elaborar un programa en C como son faltas a la sintaxis que indica el estándar, usar elementos que no se habían declarado, utilizar funciones de una biblioteca sin haberla especificado, entre muchos otros que se irán conociendo en un futuro. La mayoría de estos errores provocan que el compilador no pueda generar el programa ejecutable y muestre en la línea de comandos de qué error se trata y en qué línea pudo haberse producido.

Es importante señalar que un solo error puede desencadenar muchos otros y al corregirlo los demás dejarán de ser errores. También, en muchas ocasiones el error no se encuentra en la línea que el compilador señala sino en líneas anteriores, lo que señala es la línea en la que el compilador ya no encontró estructura el programa codificado ya que éste no tiene criterio propio sino se trata de un programa de computadora.

Cuando el compilador señala un error no cabe más que invocar algún editor de texto, revisar cuidadosamente el programa y corregir. Se debe verificar la coherencia total del programa para evitar tener que repetir este paso de manera continua.

A veces el compilador arroja advertencias durante el proceso, se generará el archivo ejecutable, pero puede tener problemas al momento de ejecución por lo que es mejor investigar de qué tratan o porqué se generaron.

A continuación, se presentan dos compiladores que pueden ser de utilidad.

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	9/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

gcc (GNU Compiler Collection)

Es un conjunto de compiladores de uso libre para sistemas operativos basados en UNIX. Entre sus compiladores existe el que sirve para programas escritos en C. Se encuentra por defecto en diversas distribuciones de GNU/Linux. El compilador trabaja en línea de comandos.

Existe también una versión modificada que puede ejecutar y crear programas para plataformas Windows en un paquete llamado MinGW (Minimalist GNU for Windows).

Al compilar un programa en C el compilador genera diversos archivos intermedios que corresponden a las distintas fases que realiza. Éstas no son de interés por el momento y son eliminadas una vez obtenido el archivo ejecutable. gcc tiene diferentes opciones de ejecución para usuarios más avanzados.

Suponiendo que se tiene un programa escrito en C y se le llamó *calculadora.c*, la manera de compilarlo es localizándose mediante la línea de comandos en la ruta donde el archivo se encuentra y ejecutando el comando:

```
gcc calculadora.c
```


Esto creará un archivo *a.out* (en Windows *a.exe*) que es el programa ejecutable resultado de la compilación.

Si se desea que la salida tenga un nombre en particular, debe definirse por medio del parámetro *-o* de gcc. Por ejemplo, para que se llame *calculadora.out* (en Windows *calculadora.exe*), se escribe en la línea de comandos:

```
gcc calculadora.c -o calculadora.out
```

A veces, para realizar un programa más complejo, se necesitan bibliotecas que se instalaron en el equipo previamente y se definió su uso en el programa escrito en C pero al momento de compilar es necesario indicar a GCC que se están usando bibliotecas que no se encuentran en su repertorio de bibliotecas estándar. Para ello es necesario utilizar el parámetro *-l* seguido inmediatamente por el nombre de la biblioteca, sin dejar espacio alguno:

```
gcc calculadora.c -o calculadora -lnombre_biblioteca
```

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	10/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés)

Los IDE están diseñados para maximizar la productividad del programador proporcionando componentes muy unidos con interfaces de usuario similares. Los IDE presentan un único programa en el que se llevan a cabo las diversas etapas de desarrollo de software. Generalmente, este programa suele ofrecer muchas características para la creación, modificación, compilación, implementación y depuración de software. Esto contrasta con el desarrollo de software utilizando herramientas no relacionadas, como *vi*, GNU Compiler Collection (*gcc*) o *make*.

Dev-C++: Este emplea el compilador MinGW. Se trata de un software libre, sencillo, ligero y eficiente, para la plataforma Windows.


Code Blocks: Este es un software libre, multiplataforma. Code Blocks es una alternativa a Dev-C++ y desarrollada mediante el propio lenguaje C++. Sus capacidades son bastante buenas y es muy popular entre los nuevos programadores. Se puede encontrar separado del compilado o la versión “mingw” que incluye g++ (*gcc* para C++).

Xcode: Este es uno de los mejores IDE para programar en Mac con el compilador *gcc* e Interface Builder.

Ejecución

La ejecución es la etapa que sigue después de haber compilado el programa. Una vez compilado el programa, se puede distribuir para equipos que ejecuten el mismo sistema operativo y tengan la misma plataforma de hardware (tipo de procesador, set de instrucciones y arquitectura en general). Los pasos para realizar la ejecución dependen del sistema operativo y del entorno.

En Windows se puede ejecutar el programa haciendo doble clic sobre el programa ya compilado, pero se recomienda exhaustivamente que se haga desde *símbolo de sistema*.

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	11/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Considerando que se tiene un programa compilado en un sistema base Unix cuyo nombre es *calculadora.out*, para ejecutar debe teclearse en línea de comandos:

.calculadora.out

Si el programa realizado necesita tener una entrada de información por medio de argumentos, éstos se colocan así:

.calculadora.out argumento1 argumento2


En un inicio, cualquier programa escrito en C sólo funcionará en modo línea de comandos, ya que para que tenga una interfaz gráfica se requiere de conocimientos avanzados sobre el lenguaje y del sistema operativo para el que se diseña el programa.

Es difícil realizar un programa con interfaz gráfica universal y que respete ANSI C, ya que el entorno depende del sistema operativo y las herramientas que provee.

Muchos errores no se reflejarán en el compilador porque el programa está correctamente escrito de acuerdo con lo que ANSI C señala, pero lo que se programó puede ser erróneo y tener resultados distintos a los deseados. Por ello, en la fase de ejecución deben hacerse diversas pruebas para verificar que el programa hace lo que debería.

Aunque el programa aparentemente funciona, deben hacerse pruebas exhaustivas para verificar la integridad del programa. Por ejemplo, si el programa realiza una división, es necesario evaluar que el divisor no sea cero, o bien, que los números que se manejan no rebasen el valor máximo que el equipo soporta, entre muchos otros detalles que pueden presentarse.

Es muy común que se construya un programa conforme a lo que ANSI C determina y se pruebe en un equipo personal que, por lo general, es de altas prestaciones. A la hora de instalar ese programa en un equipo especializado que tiene prestaciones limitadas, el programa puede no funcionar ya que no se optimizó. Por ejemplo, se crea un programa que al ejecutarse ocupa 600 MB en memoria principal en un equipo con 4 GB y se ejecutará sin problemas. Sin embargo, si este programa tiene que ejecutarse en un equipo con 1GB de memoria principal su rendimiento será muy pobre debido a que tanto el sistema operativo como otros procesos están ocupando ya una parte de esta memoria.

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	12/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Es muy importante tratar de que un programa sea íntegro y optimizado para garantizar su correcto funcionamiento y en ocasiones, el prestigio de su autor.

Lenguaje de programación C

C es un lenguaje de propósito general basado en el paradigma estructurado. El teorema del programa estructurado, demostrado por Böhm-Jacopini, dicta que todo programa puede desarrollarse utilizando únicamente 3 instrucciones de control: Secuencia, Selección e Iteración.

Un programa en C consiste en una o más funciones, de las cuales una de ellas debe llamarse *main()* y es la principal.

Al momento de ejecutar un programa objeto (código binario), se procesarán las instrucciones que estén definidas dentro de la función principal. Dicha función puede contener sentencias, estructuras de control y comentarios. Dentro de las sentencias se encuentran la declaración y/o asignación de variables, la realización de operaciones básicas, y las llamadas a funciones.


Licencia GPL de GNU

El software presente en esta guía práctica es libre bajo la licencia GPL de GNU, es decir, se puede modificar y distribuir mientras se mantenga la licencia GPL.

```

/*
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 * Authors: Julio A. de León, Jorge A. Solano and Hugo Zuñiga
 */

```

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	13/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Comentarios

Es una buena práctica en cualquier lenguaje de programación realizar comentarios para documentar el programa. En C existen dos tipos de comentarios: el comentario por línea y el comentario por bloque.

El comentario por línea inicia cuando se insertan los símbolos // y termina con el salto de línea (hasta donde termine el renglón).


El comentario por bloque inicia cuando se insertan los símbolos /* y termina cuando se encuentran los símbolos */. Cabe resaltar que el comentario por bloque puede abarcar varios renglones.

Código con comentarios (se emplean los símbolos para indicar comentarios en C)

El siguiente programa muestra las sintaxis de comentarios en C:

Programa1.c

```
#include <stdio.h>
int main() {
// Comentario por línea
/* Comentario por bloque
que puede ocupar
varios renglones */
// Este código compila y ejecuta
/* pero no muestra salida alguna
debido a que un comentario
ya sea por línea o por bloque */
// no es tomado en cuenta al momento
// de compilar el programa,
/* sólo sirve como documentación en el */
/*
código fuente
*/
return 0;
}
```

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	14/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

NOTA. Al iniciar el programa se deben agregar todas las bibliotecas que se van a utilizar en el mismo, es decir, funciones externas necesarias para ejecutar el programa. En lenguaje C la biblioteca estándar de entrada y salida está definida en *stdio.h* (standard input(i) output(o)) y provee, entre otras, funciones para lectura y escritura de datos que se verán a continuación.

El conjunto de caracteres en C

Del mismo modo que en nuestro lenguaje habitual utilizamos un conjunto de caracteres para construir instrucciones que tengan significado, los programas que se realicen en C se escriben utilizando un conjunto de caracteres formado por lo siguiente:

- Las 26 letras minúsculas del alfabeto inglés (a b c d e f g h i j k l m n o p q r s t u v w x y z).
- Las 26 letras mayúsculas (A B C D E F G H I J K L M N O P Q R S T U V W X Y Z).
- Los 10 dígitos (1 2 3 4 5 6 7 8 9 0).
- Los símbolos especiales (@ ^ { } [] () & \$ % # ~ ' " / ? ; : _ . , = / * - +).
- El espacio en blanco o barra espaciadora.

Nota: Lenguaje C es sensible a minúsculas y mayúsculas.

Esquema de la Estructura General de un programa en C (Figura 4 y 5).

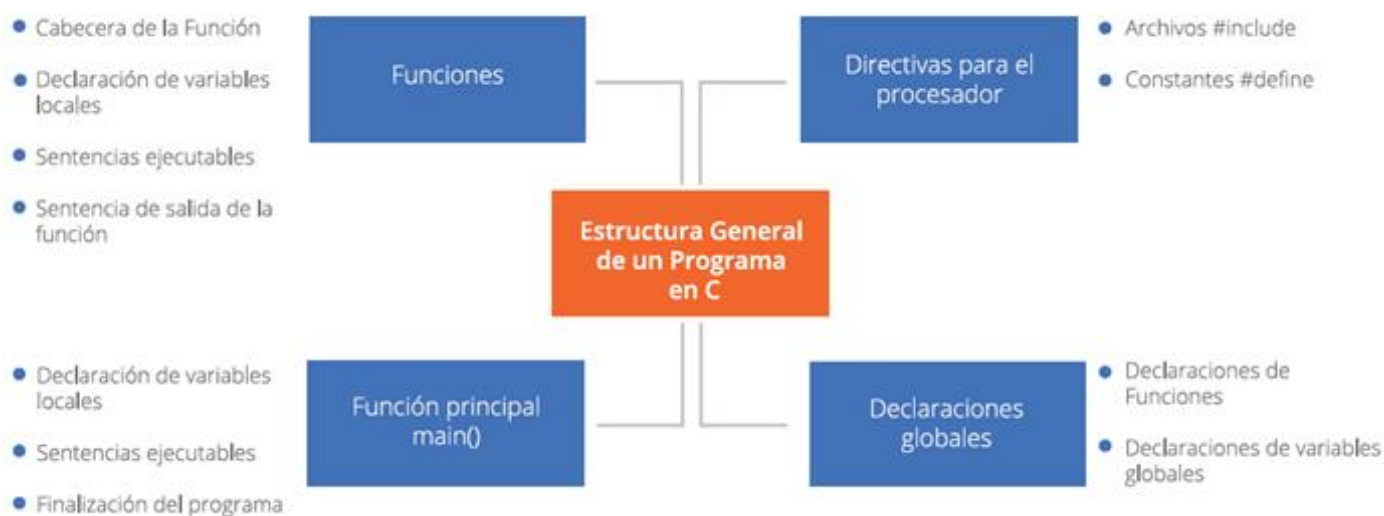



Figura 4: Estructura General de un programa en C

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	15/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

<code>#include <stdio.h></code>	Directivas o bibliotecas
<code>long cuadrado(long a); long v;</code>	Declaraciones globales
<pre>int main() { long t; printf("\nPrograma de ejemplo para obtener el cuadrado de un número"); printf("\nDa un número entero: "); scanf("%ld",&t); printf("\nEl cuadrado del número es: %ld\n",cuadrado(t)); return 0; }</pre>	Función principal main
<code>long cuadrado(long a) { return a*a; }</code>	Funciones secundarias o adicionales


Figura 5: Estructura general de un programa en C

Palabras reservadas

Las palabras reservadas (Tabla 1) son palabras que tienen un significado predefinido estándar y sólo se pueden utilizar para su propósito ya establecido; no se pueden utilizar como identificadores definidos por el programador. En lenguaje C son las siguientes:

Tabla 1: Palabras reservadas

<i>auto</i>	<i>double</i>	<i>int</i>	<i>struct</i>
<i>break</i>	<i>else</i>	<i>long</i>	<i>switch</i>
<i>case</i>	<i>enum</i>	<i>register</i>	<i>typedef</i>
<i>char</i>	<i>extern</i>	<i>return</i>	<i>union</i>
<i>const</i>	<i>float</i>	<i>short</i>	<i>unsigned</i>
<i>continue</i>	<i>for</i>	<i>signed</i>	<i>void</i>
<i>default</i>	<i>goto</i>	<i>sizeof</i>	<i>volatile</i>
<i>do</i>	<i>if</i>	<i>static</i>	<i>while</i>

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	16/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Tipos de datos


El lenguaje C ofrece distintos tipos de datos (Tabla 2), cada uno de los cuales se puede encontrar representado de forma diferente en la memoria de la computadora.

Tabla 2: Tipos de datos.

Tipo	Descripción	Espacio en Memoria	Rango
<i>int</i>	Cantidad entera	2 bytes o una palabra (varía de un compilador a otro)	-32 767 a 32 767
<i>char</i>	Carácter	1 byte	-128 a 127
<i>float</i>	Número en punto flotante (un número que incluye punto decimal y/o exponente)	1 palabra (4 bytes)	$3.4E^{-38}$ a $3.4 E^{38}$
<i>double</i>	Número en punto flotante de doble precisión (más cifras significativas y mayor valor posible del exponente)	2 palabras (8 bytes)	$1.7E^{-308}$ a $1.7E^{308}$

Existen algunos calificadores que se aplican a ciertos tipos de datos básicos. *short* (corto) y *long* (largo), califican a enteros, por ejemplo: *short int* y *long int*, en estos casos se puede omitir la palabra *int*. Los calificadores *signed* (con signo) y *unsigned* (sin signo) se usan para los tipos entero y carácter, por ejemplo: *unsigned char*, *unsigned long int*, *signed short*; si se omiten estos calificadores, por defecto se considera *signed*. Por último, el calificador *long* también se puede usar con *double*: *long double*.

Los datos de tipo flotante o doble siempre tienen signo, por lo que no se debe usar el calificador de signo en estos tipos de datos básicos.

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	17/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Código declaración de variables (se declaran variables y se asignan datos)

El siguiente programa muestra la manera en la que se declaran y asignan variables de diferentes tipos: numéricas (enteros y reales) y tipo carácter.

No compila por la línea `unsigned double puntoFlotanteNumero2 = 238.2236`, lo cual es correcto y es lo que se espera. La razón de que no compile es porque los valores reales siempre llevan signo y no es correcto usar algún calificador de signo, en este caso el *unsigned*.

Programa2.c

```
#include <stdio.h>


int main() {
    short enteroNumero1 = 115;
    signed int enteroNumero2 = 55;
    unsigned long enteroNumero3 = 789;
    char caracterA = 65;
    char caracterB = 'B';

    float puntoFlotanteNumero1 = 89.8;

    unsigned double puntoFlotanteNumero2 = 238.2236;
    return 0;
}
```

Entrada y salida de datos

Se puede acceder a una función de entrada/salida de datos desde cualquier parte de un programa con simplemente escribir el nombre de la función, seguido de una lista de argumentos entre paréntesis. Los argumentos representan los datos que le son enviados a la función. Algunas funciones de entrada/salida no requieren argumentos, pero deben aparecer los paréntesis vacíos.

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	18/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

La mayoría de las versiones de C incluyen una colección de archivos de cabecera que proporcionan la información necesaria para las distintas funciones de biblioteca. Cada archivo contiene generalmente la información necesaria para la utilización de un determinado grupo de funciones de biblioteca.

Estos archivos se incluyen en un programa mediante la instrucción *#include* al comienzo del programa. Como norma general, el archivo de cabecera requerido para la entrada/salida estándar se llama *stdio.h*

La biblioteca *stdio.h* contiene diversas funciones, tanto para imprimir en la salida estándar (monitor) como para leer de la entrada estándar (teclado).

Se pueden escribir datos en el dispositivo de salida estándar, utilizando la función de biblioteca *printf*.

En términos generales, la función *printf* se escribe:

printf(cadena de control, arg1, arg2, argn)

En donde cadena de control hace referencia a una cadena de caracteres que contiene información sobre el formato de la salida y *arg1, arg2, ... , argn* son argumentos que representan los datos de salida.

Los argumentos pueden ser constantes, variables simples o nombres de arreglos o expresiones más complicadas.

En la cadena de control contiene dos tipos de objetos: caracteres ordinarios que son copiados al flujo de salida y especificaciones de conversión; estos últimos causan la conversión y escritura de los argumentos que le siguen a la cadena de control, de acuerdo con el orden de aparición.

Cada especificación de conversión debe comenzar con el signo de porcentaje (%).

En su forma más sencilla, cada especificación de conversión estará formado por el signo de porcentaje, seguido de un carácter de conversión que indica el tipo de dato correspondiente.

Dentro de la cadena de control, las diferentes especificaciones de conversión pueden estar seguidos o pueden estar separados por caracteres de espaciado (espacios en blanco, tabuladores o caracteres de nueva línea) (Tabla 3).



	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	19/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Tabla 3: Caracteres de conversión de los datos de salida de uso común [tabla], basada en Gottfried, 1997, *Programación en C* (p. 102).

Carácter de conversión	Significado
<i>c</i>	El dato es visualizado como un carácter
<i>d</i>	El dato es visualizado como un entero decimal con signo
<i>e</i>	El dato es visualizado como un valor en coma flotante con exponente
<i>f</i>	El dato es visualizado como un valor en coma flotante sin exponente
<i>g</i>	El dato es visualizado como un valor en coma flotante utilizando la conversión tipo <i>e</i> o tipo <i>f</i> según sea el caso. No se visualizan los ceros finales ni el punto decimal cuando no es necesario
<i>i</i>	El dato es visualizado como un entero con signo
<i>o</i>	El dato es visualizado como un entero octal, sin el cero inicial
<i>s</i>	El dato es visualizado como una cadena de caracteres
<i>u</i>	El dato es visualizado como un entero decimal sin signo
<i>x</i>	El dato es visualizado como un entero hexadecimal sin el prefijo 0x

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	20/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Código que ejemplifica el uso de la función *printf*.

El siguiente programa imprime en pantalla diversos tipos de datos ulizando la función *printf*. Compila y ejecuta bien.

Programa3.c


```
#include <stdio.h>

int main() {
    //Declaración de variables
    int entero;
    float flotante;
    double doble;
    char caracter;
    //Asignación de variables
    entero = 14;
    flotante = 3.5f;
    doble = 6.8e10;
    caracter = 'A';

    //Funciones de salida de datos en pantalla
    printf("La variable entera tiene valor: %i \n", entero);
    printf("La variable flotante tiene valor: %f \n", flotante);
    printf("La variable doble tiene valor: %f \n", doble);
    printf("La variable caracter tiene valor: %c \n", caracter);

    printf("Entero como octal: %o \n Como Hexadecimal %X \n", entero, entero);
    printf("Flotante con precisión: %5.2f \n", flotante);
    printf("Doble con precisión: %5.2f \n", doble);
    printf("Carácter como entero: %d \n", caracter);

    return 0;
}
```

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	21/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Para imprimir con formato también se utilizan algunas secuencias de caracteres de escape. C maneja los siguientes (Tabla 4):

Tabla 4: Secuencias de caracteres

Especificador	Descripción
\a	Carácter de alarma
\b	Retroceso
\f	Avance de hoja
\n	Salto de línea
\r	Regreso de carro
\t	Tabulador horizontal
\v	Tabulador vertical
\0	Carácter nulo

Se pueden leer datos desde el dispositivo de entrada estándar, utilizando la función de biblioteca *scanf*.

La función *scanf* se puede utilizar para introducir cualquier combinación de valores numéricos, caracteres individuales y cadenas de caracteres. Es análoga a la función *printf*, con la diferencia de que su propósito es introducir datos en lugar de visualizarlos.


En términos generales, la función se escribe:

scanf(cadena de control, arg1, arg2, ... , argn)

Donde cadena de control hace referencia a una cadena de caracteres que contiene cierta información sobre el formato de los datos y *arg1, arg2, ..., argn* son argumentos que representan los datos (Tabla 5).

Tabla 5: Caracteres de conversión de los datos de entrada de uso común [tabla], basada en Gottfried, 1997, *Programación en C* (p. 92).

Carácter de conversión	Significado
<i>c</i>	El dato es un carácter
<i>d</i>	El dato es un entero decimal
<i>e</i>	El dato es un valor en coma flotante
<i>f</i>	El dato es un valor en coma flotante
<i>g</i>	El dato es un valor en coma flotante
<i>h</i>	El dato es un entero corto

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	22/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

<i>i</i>	El dato es un entero decimal, octal o hexadecimal
<i>o</i>	El dato es un entero octal
<i>s</i>	El dato es una cadena de caracteres seguida de un carácter de espaciado (se añade automáticamente el carácter nulo \0 al final)
<i>u</i>	El dato es un entero sin signo
<i>x</i>	El dato es un entero hexadecimal

Los argumentos pueden ser variables o arreglos y sus tipos deben coincidir con los indicados por caracteres de conversión correspondientes en la cadena de control. Cada nombre de variable debe ser precedido por un *ampersand* (&); sin embargo, los nombres de arreglos (se verán en lecciones posteriores) no deben ir precedidos por el *ampersand*.

Código que ejemplifica el uso de la función scanf

El siguiente programa muestra el almacenamiento de datos en variables. Compila y ejecuta bien.

Programa4.c


```
#include <stdio.h>

int main()
{
    int entero;
    float flotante;

    printf("Ingresa el valor entero: ");
    scanf("%i", &entero);
    printf("El valor ingresado es: %d\n", entero);

    printf("Ingresa el valor float: ");
    scanf("%f", &flotante);
    printf("El valor ingresado es: %f\n", flotante);

    return 0;
}
```

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	23/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Código que utiliza las funciones *printf* y *scanf*.

El siguiente código muestra cómo almacenar e imprimir variables. Compila y ejecuta bien.

Programa5.c

```
#include <stdio.h>


int main()
{
    int enteroNumero;
    char caracterA = 65;          // Convierte el entero a carácter ASCII.
    double puntoFlotanteNumero;

    // Asignar valor de teclado a una variable.
    printf("Escriba un valor entero: ");
    scanf("%i", &enteroNumero);
    printf("Escriba un valor real: ");
    scanf("%lf", &puntoFlotanteNumero);

    // Imprimir valores con formato.
    printf("\nImprimiendo las variables enteras \a\n");
    printf("\t Valor de enteroNumero = %i \a\n", enteroNumero);
    printf("\t Valor de caracterA = %c \a\n", caracterA);
    printf("\t Valor de puntoFlotanteNumero = %lf \a\n",
        puntoFlotanteNumero);

    printf("\t Valor de enteroNumero en base 16 = %x \a\n", enteroNumero);
    printf("\t Valor de caracterA en código hexadecimal = %x\n", caracterA);
    printf("\t Valor de puntoFlotanteNumero\n");
    printf("en notación científica = %e\n", puntoFlotanteNumero);

    return 0;
}
```

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	24/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Operadores

Los operadores aritméticos que maneja el lenguaje C se describen en Tabla 6:

Tabla 6: Operadores aritméticos

Operador	Operación	Uso	Resultado
+	Suma	125.78 + 62.5	188.28
-	Resta	65.3 - 32.33	32.97
*	Multiplicación	8.27 * 7	57.75
/	División	15 / 4	3.75
%	Módulo	4 % 2	0

Los operadores lógicos a nivel de bits que maneja el lenguaje C se describen en la Tabla 7:

Tabla 7: Operadores lógicos nivel de bits

Operador	Operación	Uso	Resultado
>>	Corrimiento a la derecha	8 >> 2	2
<<	Corrimiento a la izquierda	8 << 1	16
&	Operador AND	5 & 4	4
	Operador OR	3 2	3
~	Complemento ar-1	~2	1

Expresiones lógicas

Las expresiones lógicas están constituidas por números, caracteres, constantes o variables que están relacionados entre sí por operadores lógicos. Una expresión lógica puede tomar únicamente los valores verdadero o falso.

Los operadores de relación permiten comparar elementos numéricos, alfanuméricos, constantes o variables (Tabla 8).


	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	25/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Tabla 8: Operadores relacionales

Operador	Operación	Uso	Resultado
==	Igual que	'h' == 'H'	Falso
!=	Diferente a	'a' != 'b'	Verdadero
<	Menor que	7 < 15	Verdadero
>	Mayor que	11 > 22	Falso
<=	Menor o igual	15 <= 22	Verdadero
>=	Mayor o igual	20 >= 35	Falso

Los operadores lógicos permiten formular condiciones complejas a partir de condiciones simples (Tabla 9).

Tabla 9: Operadores lógicos


Operador	Operación	Uso
!	No	! p
&&	Y	a > 0 && a < 11
	O	opc == 1 salir != 0

Código que ejemplifica el uso de operadores a nivel de bits.

El siguiente programa muestra cómo manipular números a nivel de bits:

- Corrimiento de bits a la izquierda y a la derecha
- Operador AND a nivel de bits
- Operador OR a nivel de bits

Compila y ejecuta bien.

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	26/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Programa6.c

```
#include <stdio.h>
int main()
{
    short ocho, cinco, cuatro, tres, dos, uno;


    // 8 en binario: 0000 0000 0000 1000
    ocho = 8;
    // 5 en binario: 0000 0000 0000 0101
    cinco = 5;
    // 4 en binario: 0000 0000 0000 0100
    cuatro = 4;
    // 3 en binario: 0000 0000 0000 0011
    tres = 3;
    // 2 en binario: 0000 0000 0000 0010
    dos = 2;
    // 1 en binario: 0000 0000 0000 0001
    uno = 1;
    printf("Operadores aritméticos\n");
    printf("5 modulo 2 = %d\n",cinco%dos);
    printf("Operadores lógicos\n");
    printf("8 >> 2 = %d\n",ocho>>dos);
    printf("8 << 1 = %d\n",ocho<<1);
    printf("5 & 4 = %d\n",cinco&cuatro);
    printf("3 | 2 = %d\n",tres|dos);

    printf("\n");
    return 0;
}
```

Lenguaje C posee operadores para realizar incrementos y decrementos de un número.

El operador ++ agrega una unidad (1) a su operando. Es posible manejar pre-incrementos (++n) o pos-incrementos (n++).

El operador -- resta una unidad (1) a su operando. Se pueden manejar pre-decrementos (--n) o pos-decrementos (n--).

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	27/27
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Bibliografía

- El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, segunda edición, USA, Pearson Educación 1991.