

Tomado de: <https://www.tiobe.com/tiobe-index/>  
 Fecha: 2020-08-18

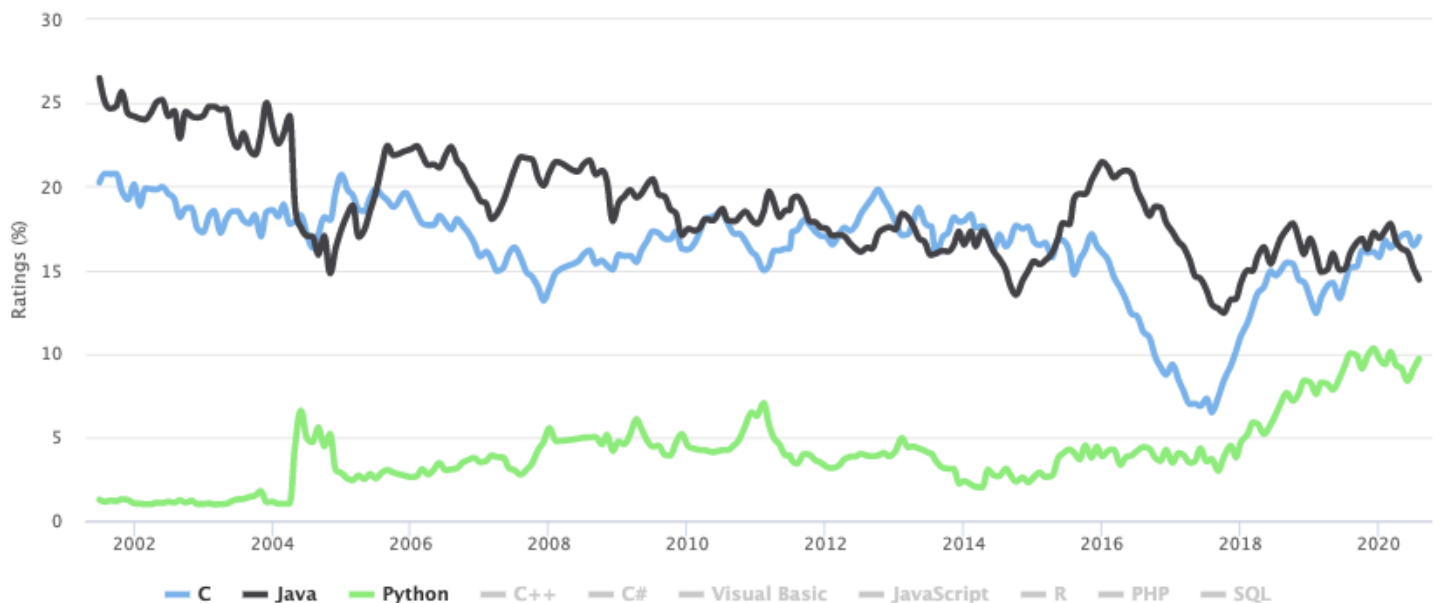
A nivel mundial los primeros diez(10) lenguajes de programación de computadoras más utilizados son, según el índice TIOBE para Agosto de 2020 ( TIOBE Index for Aug 2020)

" ...

tiobe.com/tiobe-index/						
	Aug 2020	Aug 2019	Change	Programming Language	Ratings	Change
1	2		▲	C	16.98%	+1.83%
2	1		▼	Java	14.43%	-1.60%
3	3			Python	9.69%	-0.33%
4	4			C++	6.84%	+0.78%
5	5			C#	4.68%	+0.83%
6	6			Visual Basic	4.66%	+0.97%
7	7			JavaScript	2.87%	+0.62%
8	20		▲▲	R	2.79%	+1.97%
9	8		▼	PHP	2.24%	+0.17%
10	10			SQL	1.46%	-0.17%

### TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)



Tomado de:

<https://www.xataka.com/aplicaciones/venerable-lenguaje-programacion-c-cumple-48-anos-dandole-repaso-al-cada-vez-popular-python>

Fecha: 2020-06-20

"Publicado el 2020-01-09

El venerable lenguaje de programación C cumple 48 años dándole un repaso al cada vez más popular Python

.... Su utilización a lo largo de los años se ha consolidado de tal forma que casi cinco décadas después el lenguaje C sigue siendo uno de los más populares del mundo. El último índice de TIOBE así lo confirma... Los analistas de TIOBE indicaban que una de las razones es el auge que **C** ha tenido en dispositivos inteligentes y en la Internet de las Cosas (Internet of Things -IoT).... De hecho, hay otra gran ventaja de este lenguaje sobre el resto de opciones de mercado: **está virtualmente soportado en todas las plataformas existentes hoy en día** en el mundo de la computación. Como explicaban estos analistas, "C sobresale cuando se aplica a dispositivos pequeños que son críticos para el rendimiento. Es fácil de aprender, y hay un compilador de C disponible para cada procesador".

Tomado de:

<https://www.universidadviu.com/deberias-aprender-programacion-c/>

Fecha: 2020-06-20

"Publicado 2018-03-21

¿Por qué deberías aprender programación en C?

La **programación en C** se utiliza, entre otras cosas, para el desarrollo de sistemas operativos. El primer sistema operativo escrito en C fue Unix. Más tarde, otros sistemas operativos como Linux también fueron escritos en C. Pero C no es sólo el lenguaje de programación de los sistemas operativos. C es el precursor e inspirador para casi todos los lenguajes de alto nivel más populares disponibles en la actualidad. De hecho, **Perl, PHP, Python y Ruby** están escritos mediante programación en C.... C es **simple, elegante** y endiabladamente **rápido**. Es también **compacto y eficiente**. C tiene punteros puros, operadores bit a bit, y las keywords extern, volatile, static, y register lo que significa que vas a entender más acerca de cómo escribir el código eficiente que se puede obtener de cualquier lenguaje de alto nivel. La única cosa que te enseñará estas cosas mejor es un lenguaje de más bajo nivel, pero con los complejos procesadores modernos no sería recomendable utilizar ensamblador o similar....

Los siguientes son algunos de los sistemas que son utilizados por millones de personas y están programados en el lenguaje C.

### Microsoft Windows

El kernel de Microsoft Windows se desarrolla principalmente en C, con algunas partes en lenguaje ensamblador. Durante décadas, el sistema operativo más utilizado del mundo, con aproximadamente el 90 por ciento de la cuota de mercado, ha sido impulsado por un núcleo escrito en C.

### Linux

Como ya hemos dicho Linux también está escrito principalmente en C, con algunas partes, al igual que windows, en ensamblador. Alrededor del 97 por ciento de los 500 superordenadores más potentes del mundo corren el núcleo de Linux. También se utiliza en muchos ordenadores personales.

### Mac

Los ordenadores Mac también son soportados por una **programación en C**. El núcleo OS X está escrito principalmente en C y cada programa y driver en un Mac, igual que en Windows y Linux, se ejecuta con un kernel C.

### Móvil

Los kernels de **iOS, Android y Windows Phone** también están escritos en C. En realidad son sólo adaptaciones móviles de los kernels existentes en Mac OS, Linux y Windows. Así que los teléfonos inteligentes que utilizamos todos los días se están ejecutando en un núcleo con programación en C.

### Bases de datos

Las **bases de datos más populares del mundo, incluyendo bases de datos Oracle, MySQL, MS SQL Server y PostgreSQL, están codificados en C** (los primeros tres de ellos en realidad tanto en C como C++). Las bases de datos se utilizan en todo tipo de sistemas: financieros, gubernamentales, en medios de

comunicación, entretenimiento, telecomunicaciones, salud, educación, comercio minorista, redes sociales, web, y similares.

### **Películas 3D**

Las películas en 3D son creadas con aplicaciones que generalmente están escritas en C y C ++. Esas aplicaciones tienen que ser muy eficientes y rápidas, ya que manejan una gran cantidad de datos y hacen muchos cálculos por segundo. Cuanto más eficientes son, menos tiempo toma a los responsables de crearlas para generar los fotogramas de las películas. Y eso no solo es tiempo sino también mucho dinero ahorrado.

### **Sistemas embebidos, un ejemplo del uso diario**

Como ya dijimos, muchos sistemas embebidos utilizan programación en C en la actualidad. Pero vamos a verlo con un **ejemplo de una rutina diaria**.

Imagina que te despiertas un día para irte de compras. El despertador que te despierta probablemente tiene programación en C. A continuación utilizamos el microondas y la cafetera express para hacer el desayuno. También tienen sistemas integrados y por lo tanto probablemente están programados en C. Si enciendes el televisor o la radio mientras desayunas, estarás utilizando también sistemas integrados que funcionan con C. Al abrir la puerta del garaje con el mando a distancia también estás usando un sistema embebido que es más que probable que tenga programación en C.

Después entras en tu coche y te encuentras las siguientes características, también **programadas en C**:

- Transmisión automática
- Sistemas de detección de presión de los neumáticos
- Sensores ( temperatura, nivel de aceite, etc.)
- La configuración de los espejos retrovisores
- La pantalla del salpicadero
- Los frenos ABS
- El control automático de la estabilidad
- El control de velocidad de crucero
- El climatizador
- Las cerraduras a prueba de niños
- La apertura sin llave
- Los asientos con calefacción
- El control del airbag

Llegas a la tienda, aparcas el coche y vas a una máquina expendedora para obtener un refresco. ¿Qué lenguaje utilizan para programar esta máquina expendedora? Probablemente C. Luego compras algo en la tienda. La caja registradora está programada también en C. ¿Y cuando pagas con tu tarjeta de crédito? Lo has adivinado: el lector de tarjetas de crédito está, otra vez, probablemente programado en C.

Todos estos dispositivos son sistemas embebidos. Son como pequeños equipos que tienen un microcontrolador / microprocesador interno en el que se está ejecutando un programa, también llamado firmware. Ese programa debe detectar pulsaciones de teclas y actuar en consecuencia, y también mostrar información al usuario. Por ejemplo, el reloj de alarma debe interactuar con el usuario, detecta la pulsación del botón, y a veces incluso, el tiempo que está siendo presionado. Y después programa el dispositivo en consecuencia, a la vez que presenta al usuario la información relevante. El sistema de frenos antibloqueo del coche, por ejemplo, debe ser capaz de detectar de bloqueo repentino de los neumáticos y actuar para liberar la presión en los frenos por un pequeño periodo de tiempo para desbloquearlos, evitando de este modo el arrastre no controlado del vehículo. Todos los cálculos se realizan mediante un sistema integrado programado. Aunque el lenguaje de programación utilizado en sistemas embebidos puede variar de una marca a otra, con mayor frecuencia utilizan **programación en C**, debido a las características de este lenguaje como flexibilidad, eficiencia, rendimiento y cercanía con el hardware...."

*Tomado de:*

*<https://sites.google.com/site/lenguajecprogramacion/dscargas>*

*Fecha: 2020-06-20*

*"..Características el lenguaje C*

Su principal característica es que es portable, quiere decir que puedes adaptar los programas escritos para un tipo de computadora en otra. También es estructurado, por que se divide en módulos que son independientes entre si.

El lenguaje C inicialmente fue creado para la programación de:

Sistemas Operativos

Interpretes

Editores

Ensambladores

Compiladores

Administradores de bases de datos

..

Programación de sistemas

Estructuras de datos y sistemas de bases de datos

Aplicaciones científicas

Software gráfico

Análisis numérico

... Gracias a este lenguaje, la tecnología ha podido alcanzar un gran avance en todos los ámbitos ..es por eso que es necesario que este ... lenguaje se aprenda para poder realizar utilerías capaces de trabajar y apoyar proyectos..”

Para facilidad en el aprendizaje, lo más recomendable es utilizar un software denominado IDE.

### ¿Qué es un IDE y para qué sirve?

Un entorno de desarrollo integrado o entorno de desarrollo interactivo, en inglés Integrated Development Environment (IDE), es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.

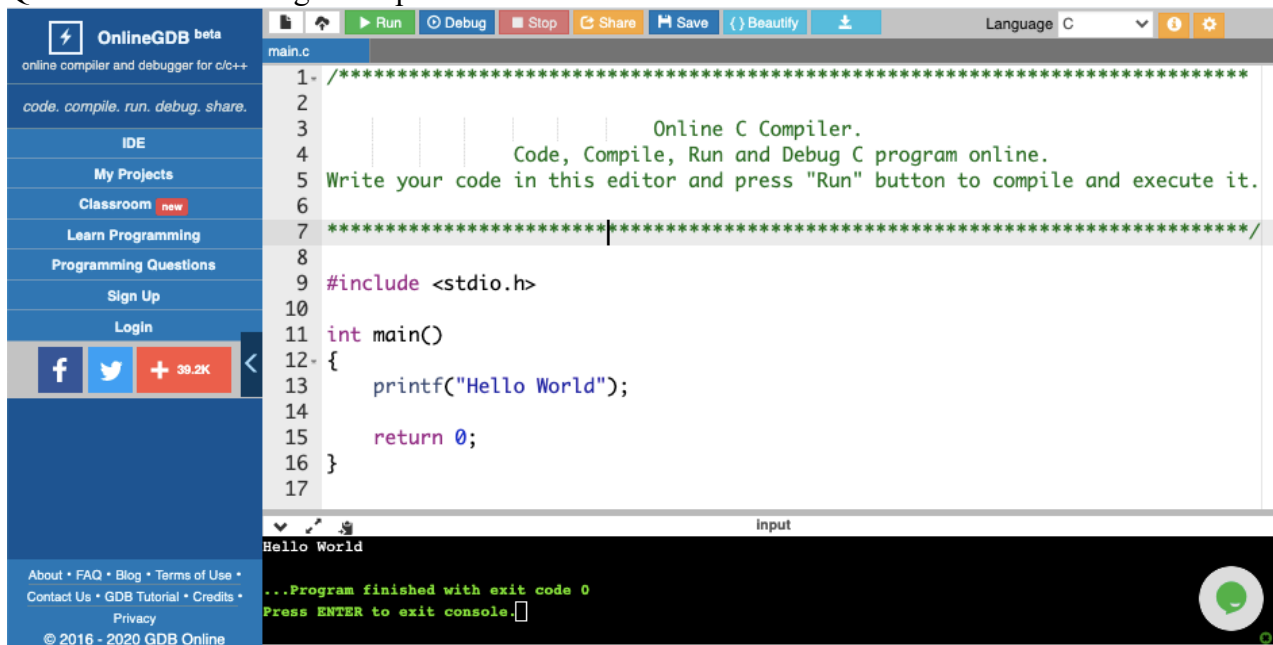
Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, consiste en un editor de código, un depurador y un constructor de interfaz gráfica.

Existen muchos programas IDE, sin embargo, utilizaremos uno que se nos ofrece en la web, que además nos presta el servicio de ejecutar nuestros programas, sin necesidad de tener en nuestra computadora el lenguaje C. Esto es que desde un smartphone con acceso a internet o una tablet, podríamos ejecutar practicamente todos los programas de la asignatura.

El link es:

<https://www.onlinegdb.com/>

Que nos mostrara la siguiente pantalla:



Si se registra con el email institucional "@utp.edu.co" o con uno de "@gmail.com", podrá registrarse gratuitamente por "Sign UP" y grabar(Save) sus programas.

En la opción "Language", que se nos ofrece en la esquina superior derecha, podemos elegir varios lenguajes de programación, en nuestro caso elegiremos "C".

### ¿Qué es un programa para computadora?

Un programa para computadora es una secuencia de instrucciones que especifican cómo ejecutar una computación. La computación puede ser algo matemático, como solucionar un sistema de ecuaciones, hacer cálculos estadísticos, hacer simulación de mercados financieros, etc.

Las instrucciones (comandos, órdenes) tienen una apariencia diferente en lenguajes de programación diferentes, pero existen algunas funciones básicas que se presentan en casi todo lenguaje:

**Salida:** Mostrar datos en la pantalla del ordenador, enviar datos a una impresora, a un archivo, u otro dispositivo de salida (motor, alarma.)

**Entrada:** Recibir datos del teclado, un archivo u otro dispositivo (Sensor de temperatura, alarmas, etc.).

**Matemáticas:** Ejecutar operaciones básicas de matemáticas como sumas (+), restas (-), multiplicaciones (\*), divisiones (/), residuo(resto de)(remainder) (%),etc.

**Operación condicional:** Probar la veracidad de alguna condición y ejecutar una secuencia de instrucciones apropiada.

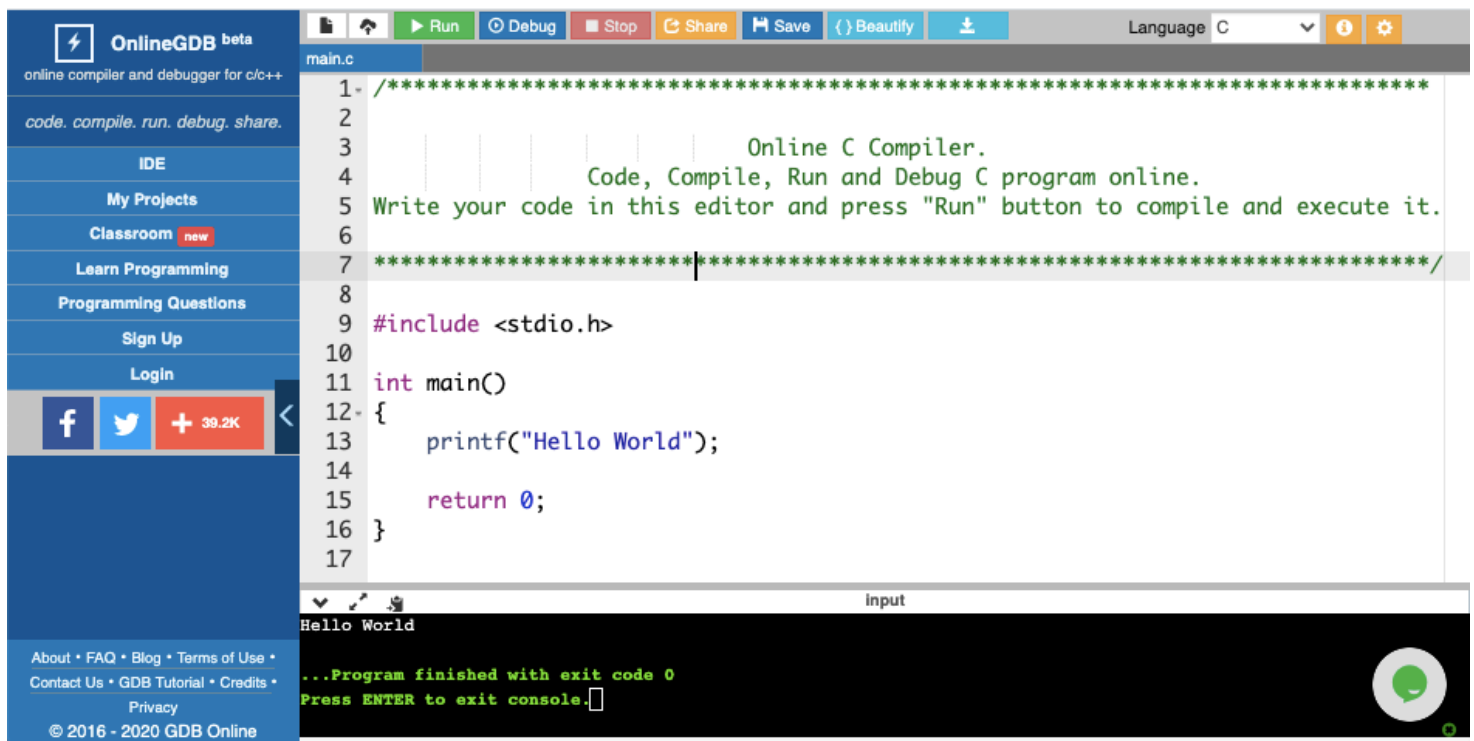
**Repetición:** Ejecutar alguna acción repetidas veces, normalmente con alguna variación.

**Lo crea o no, eso es todo.** Todos los programas que existen, por complicados que sean, están formulados exclusivamente con tales instrucciones. Así, una manera de describir la programación de computadoras es: El proceso de romper una tarea en tareas cada vez más pequeñas hasta que estas tareas sean suficientemente simples para ser ejecutadas con una de estas instrucciones simples.

Comencemos con la función de salida (El marco de referencia es salida desde el ordenador).

La instrucción que ofrece C es: **printf();**

Pruebe los ejemplos de esta instrucción en el IDE propuesto, escríbalos y luego presione el botón 



The screenshot shows the OnlineGDB IDE interface. On the left is a sidebar with navigation links like 'IDE', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Sign Up', and 'Login'. The main editor area displays a C program in 'main.c' with the following code:

```
1- /*****  
2  
3  
4  
5  
6  
7- *****/  
8  
9 #include <stdio.h>  
10  
11 int main()  
12 {  
13     printf("Hello World");  
14  
15     return 0;  
16 }  
17
```

Below the code editor is an 'input' field and a console output area. The console shows the output 'Hello World' and the message '...Program finished with exit code 0'. At the bottom right, there is a green circular button with a white play icon.

Al correr(Run) el programa, en la parte "negra", aparece lo que en la pantalla de su computador mostraría este programa; así:

Hello World

Tomado de:

<https://www.aulafacil.com/cursos/programacion/lenguaje-de-programacion-c/primer-programa-en-c-l16525>

Fecha: 2020-04-20

# MI PRIMER PROGRAMA EN C

No te preocupes si al principio te cuesta captar los conceptos básicos de la programación o si hay líneas de código que no entiendes, es normal, al empezar vas a ver ciertas cosas que no se pueden explicar en este momento que no sabes nada, porque son conceptos que se aprenderán posteriormente en el curso. Así que por eso no te preocupes.

Como ya hemos aclarado, C es un lenguaje estructurado que se escribe por instrucciones de programación, y cada instrucción acabará con un ";" al final, que indica el final de la instrucción y que pasará a ejecutarse la siguiente.

Veamos como queda nuestro primer programa:

```
#include <stdio.h>

int main() {
    // printf nos muestra datos por pantalla
    printf("Hola Mundo. \n");
    return 0;
}
```

Lo único que hace este programa es mostrar la frase "Hola Mundo" en la PANTALLA de nuestro ordenador.

Primero quiero dejar claro que las frases que llevan antes "//" son cosas que el compilador no ejecuta. Esto se llama comentarios, y nos sirven para que nosotros mismos, los programadores, sepamos que hace cada instrucción. En este tipo de programas no, ahora imagínese un programa de más de 3000 líneas de código; éste tendría que tener comentarios aclaratorios para que no se convierta en un caos, para otros programadores y nosotros mismos cuando después de algún tiempo necesitemos volver a revisar este código.

Bien, empezamos a analizar el código.

**#include <stdio.h>**

Estos "#include" hacen referencia a librerías. Las librerías las veremos más adelante, por lo que ya entenderas por qué están ahí. De momento ponédalas sin más, porque son necesarias para que funcione el programa.

**int main(){**

...

**}**

Esto es la función principal "main", es decir, la función principal del programa. Todo lo que esté dentro de { ... } del main es lo que se va a ejecutar cuando ejecutemos el programa.

Por eso, todo programa debe llevar su "main", aquí estamos diciendo:

```
int main()
```

Que la función devuelve un entero (int), y que la función se llama main

Nota: Por definición del lenguaje C, toda función, por defecto retorna ó devuelve un entero(int).

```
printf("Hola Mundo.");
```

**printf** es una función(que esta en la librería de funciones <stdio.h>) que nos permite imprimir ó escribir en pantalla. Escribiremos entre paréntesis y comillas como veis, la frase a mostrar, y el programa la mostrará; es decir en pantalla mostrara lo siguiente:

Hola Mundo.

```
return 0;
```

Ya que por definición del lenguaje C, toda función por defecto retorna ó devuelve un entero, por ahora retornemos el entero cero(0), luego daremos una más amplia explicación.

El formato básico de un programa en lenguaje C, tendría:

#include <stdio.h>	Indicamos al lenguaje, que incluya la librería " <b>standar input output</b> ". Esta librería contiene funciones para entrada de datos al computador y salida de datos desde el computador (impresión de datos), como por ejemplo la función de salida de datos a pantalla (impresión) <b>printf</b>
int main(){  return 0; }	Indicamos, que la función devuelve un entero (int), y que la función se llama main  Aquí con <b>return 0;</b> estamos indicando que retornamos ó devolvemos un cero(0), cumpliendo el hecho que toda función por defecto devuelve un entero

...."

**EL LENGUAJE C ES  
SENSIBLE AL CONTEXTO**

**LAS MAYÚSCULAS SON  
DIFERENTES A LAS  
minúsculas**



# Bit's y Byte's

Las computadoras actuales, operan utilizando circuitos electrónicos, donde la información de datos alfabéticos y numéricos, son conformados por cadenas de ceros(0) y unos(1) que desde los circuitos electrónicos un uno(1) equivale a que halla voltaje y un cero(0) a que no halla voltaje. Se dice entonces, que la computación es binaria(dos dígitos) y es posible gracias al **álgebra booleana**, que fue inventada en el año 1854 por el matemático inglés George **Boole**. El **álgebra de Boole** es un método para simplificar los circuitos lógicos (o a veces llamados circuitos de conmutación lógica) en electrónica digital.

"Las interpretaciones respectivas de los símbolos 0 y 1 en el sistema de lógica son Nada y Universo."

*George Boole*

Los procesadores, son los dispositivos electrónicos, basados en el sistema binario, que permiten manejar los datos de los usuarios que requerimos usarlos para extraer de ellos información.

Un pequeño resumen de la historia de los microprocesadores, nos dará pie a la explicación de los bits y los bytes.

*Tomado a adaptado de:*

*<https://procesadoresarmenia.wordpress.com/historia-del-procesador-intel/>*

*Fecha: 2020-06-19*

"...HISTORIA DEL PROCESADOR INTEL

El primer procesador comercial, el Intel 4004, fue presentado el 15 de noviembre de 1971. Los diseñadores fueron Ted Hoff y Federico Faggin de Intel, y Masatoshi Shima de Busicom (más tarde ZiLOG).

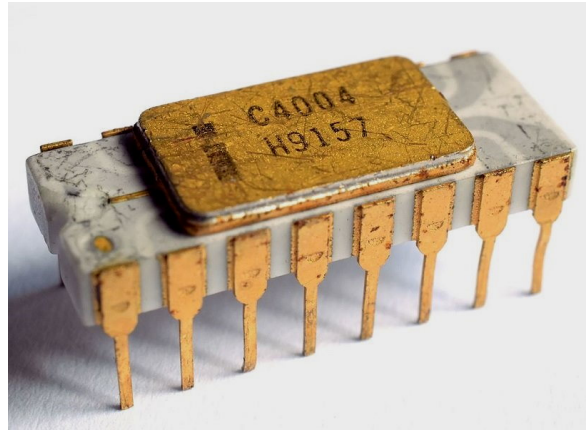
....

Existen una serie de fabricantes de microprocesadores, como IBM, Intel, Zilog, Motorola, Cyrix y AMD. A lo largo de la historia y desde su desarrollo inicial, los microprocesadores han mejorado enormemente su capacidad, desde los viejos Intel 8080, Zilog Z80 o Motorola 6809, hasta los recientes Intel Core 2 Duo, Intel Core 2 Quad, Intel Xeon, Intel Itanium II, Transmeta Efficeon o Cell.

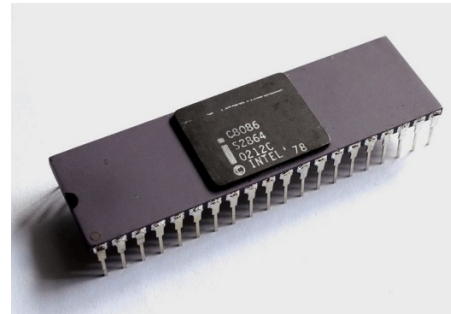
Ahora los nuevos microprocesadores pueden tratar instrucciones de hasta **256 bits**, habiendo pasado por los de 128, 64, 32, 16, 8 y **4 bits**. Desde la aparición de los primeros computadores en los años cuarenta del siglo XX, muchas fueron las evoluciones que tuvieron los procesadores antes de que el microprocesador surgiera por simple disminución del procesador.

## Procesadores INTEL:

- 4004, año 1971, cuatro(4) bits



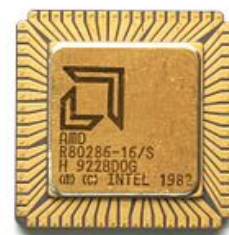
- 8086, año 1978, ocho(8) bits



- 8088, año 1979, dieciseis(16) bits. Este procesador no dispone de ningún modo de hacer operaciones en punto flotante. Si queremos dotarle de esta habilidad, habría que añadirle un coprocesador matemático. El coprocesador matemático más común era el Intel 8087. Fue el microprocesador usado para el primer computador personal de IBM, el **IBM PC**, que salió al mercado en agosto de 1981. Hay que tener en cuenta que la mayoría del hardware, de principios de los 80, era de 8 bits, y más barato. El hardware de **16 bits** era casi inexistente en 1981 y carísimo



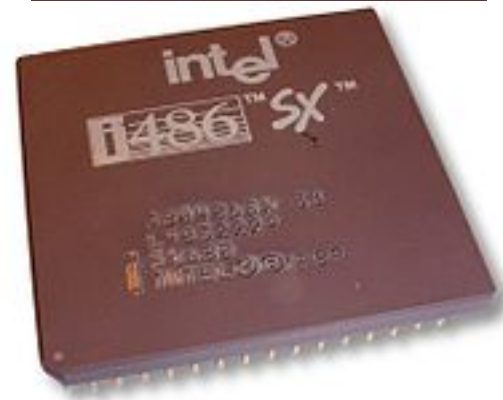
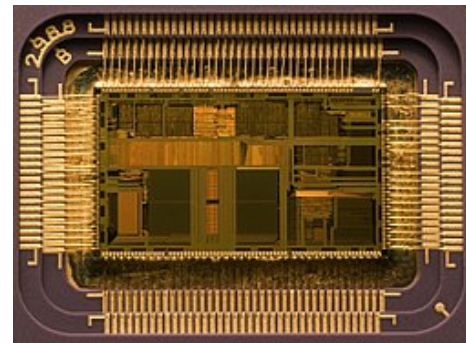
- 80286, año 1982, dieciseis(16) bits + Multitarea. Fue el microprocesador elegido para equipar al **IBM Personal Computer/AT**, introducido en 1984, lo que causó que fuera el más empleado en los compatibles AT hasta principios de los 1990.



- 80386, año 1986, treinta y dos(32) bits + Multitarea



- 80486, año 1989, treinta y dos(32) bits + Multitarea + una unidad de punto flotante y un caché de memoria de 8 KBytes



- Pentium, año 1993, treinta y dos(32) bits



Familia Pentium

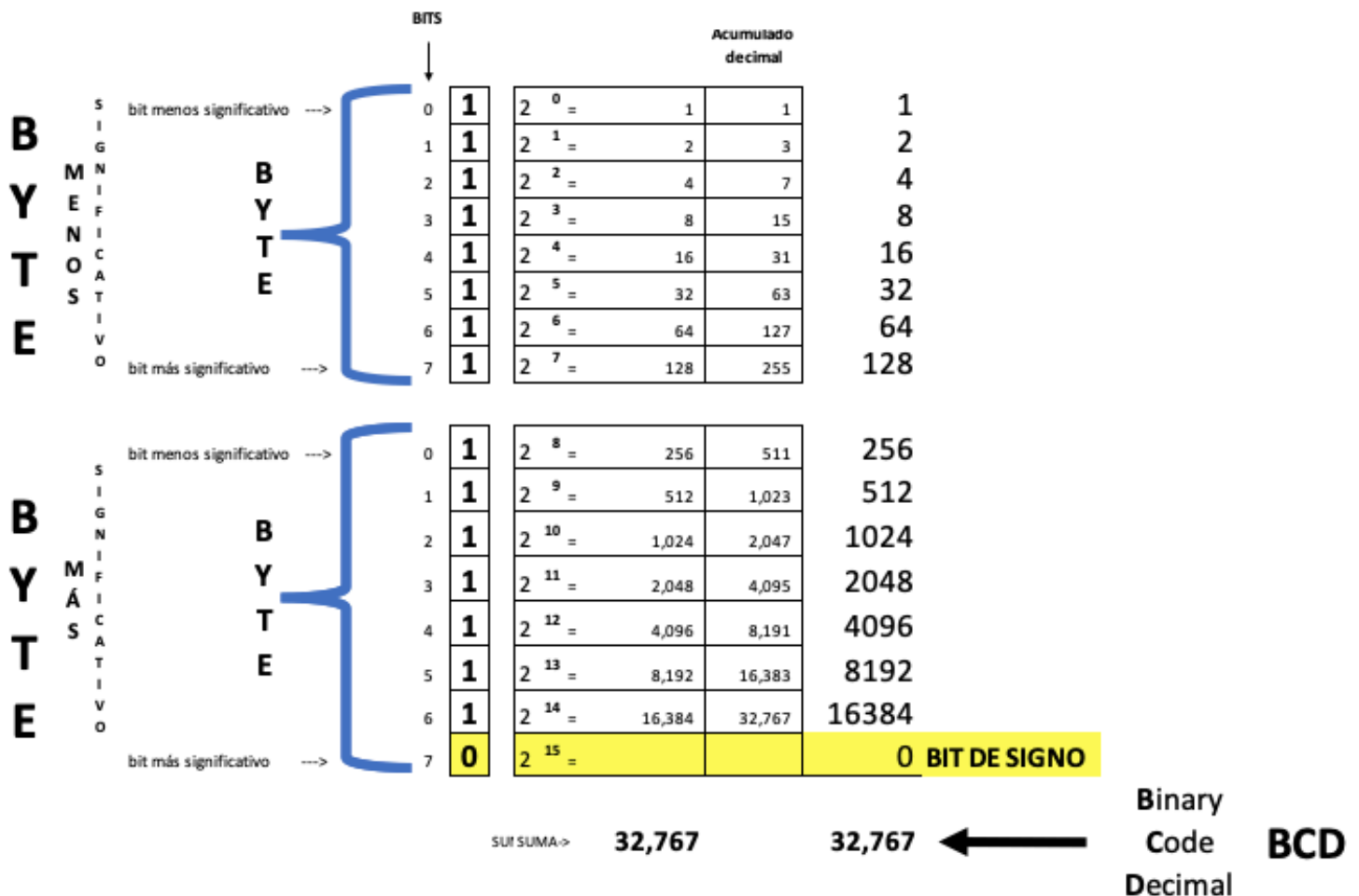
<p>- Pentium Pro, año 1995, treinta y dos(32) bits,</p>	 <p>The image shows the top of an Intel Pentium Pro processor. It has a gold-colored central area with the 'intel' logo and 'PENTIUM PRO' text. Above this, small text reads 'K608521EX200 SL22V 256K ICOMP 2 #228'. Below the gold area, more small text reads 'L7892304-8315 INTEL © 1995'.</p>
<p>- Pentium II, año 1997, treinta y dos(32) bits</p>	 <p>The image shows an Intel Pentium II processor in its black plastic packaging. The packaging has 'pentium II' and the 'intel' logo printed on it. The processor is shown at an angle, revealing the pins on the bottom.</p>
<p>- Pentium II Xeon, año 1998, sesenta y cuatro (64) bits. Dual Core</p>	
<p>- Pentium III, Año 1999, sesenta y cuatro (64) bit</p>	 <p>The image shows the top of an Intel Pentium III processor. It has a black plastic package with a central window showing the silicon die. The text 'pentium III' is printed on the package. The 'intel' logo is visible in the top left corner. Gold pins are visible on the bottom edges.</p>
<p>- Pentium 4, Año 2000, sesenta y cuatro(64) bits</p>	 <p>The image shows the top of an Intel Pentium 4 processor. It has a square, silver-colored metal cap. Text on the cap includes 'INTEL © 04', 'PENTIUM 4', '2.40GHZ/1M/533', 'SL88F PHILIPPINES', and '75458865'. There is also a QR code and the numbers '6549B136' and '8125'.</p>

<p>- Pentium M, Año 2003, sesenta y cuatro(64) bits</p>	 <p>The image shows the underside of an Intel Pentium M processor. It features a green printed circuit board (PCB) with a central black square heat spreader. Two black labels are visible: one at the top right with '1.4/2M/533' and 'INTEL®' and another at the bottom with 'RH88536 730' and '7514A764 SL840'.</p>
<p>- Pentium D, Año 2005, sesenta y cuatro(64) bits</p> <p>- Core 2 Duo y Core 2 Quad, Año 2006, sesenta y cuatro(64) bits</p>	 <p>The image shows the top of an Intel Core 2 Duo processor. It has a square, silver-colored metal heat spreader with a green PCB visible around the edges. The heat spreader has text including 'INTEL® CORE™2 Duo', 'SL941', and 'Intel®'.</p>
<p>Intel® Xeon® Processor E7-8890 v4, año 2020, veinticuatro(24) núcleos. sesenta y cuatro(64) bits</p>	 <p>The image shows the top of an Intel Core i9 X-series processor. It features a large, square, silver-colored metal heat spreader with the Intel logo and 'CORE i9 X-series' printed on it. The green PCB is visible at the bottom.</p>

Como puede apreciarse, los procesadores para las computadoras nacen con ocho(8) bits y a la fecha se producen hasta 128 bits de manera comercial y algunos de 1024 bits como prototipos de investigación. La mayoría de los computadores actuales trabajan con sesenta y cuatro(64) y treinta y dos(32) bits, porque de hecho los sistemas operativos comerciales, trabajan a sesenta y cuatro(64) bits.

Los bits de un procesador se refieren al ancho de su Unidad Aritmética Lógica (ALU). Este es el ancho del número binario que puede manejar de forma nativa en una sola operación. Por lo tanto, un procesador de "32 bits" puede operar directamente en valores de hasta 32 bits de ancho en instrucciones individuales.

Los lenguajes de programación de computadoras como el lenguaje C, han ido evolucionando acorde al hardware. Por muchos años, para una variable de tipo **int**, se reservó dos(2) bytes; así:



Dejando el bit más significativo del byte más significativo para el signo, por lo que para versiones de procesadores de dieciséis(16) bits ó lo que es lo mismo, para procesadores de dos(2) bytes, se podrían manejar enteros entre +32767 y -32767.

Sin embargo, muchos lenguajes no se han adaptado a los sesenta y cuatro(64) bits y se han quedado en treinta y dos(32) bits, por lo que el tipo de dato **int**, que se utiliza en estos computadores son de cuatro(4) bytes, dejando el bit más significativo del byte más significativo para el signo, por lo que para versiones de procesadores de treinta y dos(32) bits ó lo que es lo mismo, para procesadores de cuatro(4) bytes, se podrían manejar enteros entre [-2,147,483,647 y +2,147,483,647]. Esto es que para una variable de tipo **int**, si reservo cuatro(4) bytes; tendremos:



**BYTE**

SIGNIFICATIVO

bit menos significativo

**BYTE**

bit más significativo

BITS

Acumulado decimal

0	1	$2^0 =$	1	1
1	1	$2^1 =$	2	3
2	1	$2^2 =$	4	7
3	1	$2^3 =$	8	15
4	1	$2^4 =$	16	31
5	1	$2^5 =$	32	63
6	1	$2^6 =$	64	127
7	1	$2^7 =$	128	255

1  
2  
4  
8  
16  
32  
64  
128

bit menos significativo

**BYTE**

bit más significativo

0	1	$2^8 =$	256	511	256
1	1	$2^9 =$	512	1,023	512
2	1	$2^{10} =$	1,024	2,047	1,024
3	1	$2^{11} =$	2,048	4,095	2,048
4	1	$2^{12} =$	4,096	8,191	4,096
5	1	$2^{13} =$	8,192	16,383	8,192
6	1	$2^{14} =$	16,384	32,767	16,384
7	1	$2^{15} =$	32,768	65,535	32,768

256  
512  
1,024  
2,048  
4,096  
8,192  
16,384  
32,768

bit menos significativo

**BYTE**

bit más significativo

0	1	$2^{16} =$	65,536	131,071	65,536
1	1	$2^{17} =$	131,072	262,143	131,072
2	1	$2^{18} =$	262,144	524,287	262,144
3	1	$2^{19} =$	524,288	1,048,575	524,288
4	1	$2^{20} =$	1,048,576	2,097,151	1,048,576
5	1	$2^{21} =$	2,097,152	4,194,303	2,097,152
6	1	$2^{22} =$	4,194,304	8,388,607	4,194,304
7	1	$2^{23} =$	8,388,608	16,777,215	8,388,608

65,536  
131,072  
262,144  
524,288  
1,048,576  
2,097,152  
4,194,304  
8,388,608

bit menos significativo

**BYTE**

bit más significativo

0	1	$2^{24} =$	16,777,216	33,554,431	16,777,216
1	1	$2^{25} =$	33,554,432	67,108,863	33,554,432
2	1	$2^{26} =$	67,108,864	134,217,727	67,108,864
3	1	$2^{27} =$	134,217,728	268,435,455	134,217,728
4	1	$2^{28} =$	268,435,456	536,870,911	268,435,456
5	1	$2^{29} =$	536,870,912	1,073,741,823	536,870,912
6	1	$2^{30} =$	1,073,741,824	2,147,483,647	1,073,741,824
7	0	$2^{31} =$			0 BIT DE SIGNO

0 BIT DE SIGNO

**BYTE**

SIGNIFICATIVO

SUMA-> 2,147,483,647

2,147,483,647

Binary Code

**BCD**

**TIPOS DE DATOS EN C Y RANGO DE LOS TIPOS DE DATOS**(Si utilizamos un procesador de 32 bits o de 64 bits, pero operando con 32 bits)

Tipo	Explicación	Especificador de formato
char	La unidad direccionable más pequeña de la máquina que puede contener un juego de caracteres básico. Es un tipo entero ( <b>int</b> ). Este tipo puede ser con signo o sin signo. Contiene bits CHAR_BIT. Un(1) byte.	%c
signed char	Del mismo tamaño que <b>char</b> , pero garantiza que es con signo(+/-). Capaz de contener al menos el rango [-127, +127]. Un(1) byte.	%c (o %i para salida numérica)
unsigned char	Del mismo tamaño que <b>char</b> , pero garantizado que no tiene signo(+/-). Contiene al menos el rango [0, 255]. Un(1) byte.	%c (o %i para salida numérica)
short short int signed short signed short int	Tipo entero ( <b>int</b> ) con signo <i>corto</i> . Capaz de contener al menos el rango [-32,767, +32,767]; tiene 16 bits de tamaño(dos bytes).	%i
unsigned short unsigned short int	Tipo entero <i>corto</i> sin signo. Contiene al menos el rango [0, 65,535] . tiene 16 bits de tamaño(dos bytes).	%i ó %u
int signed signed int	Tipo entero con signo básico. Capaz de contener al menos el rango [- 2,147,483,647 a + 2,147,483,647]; tiene 32 bits de tamaño. (cuatro bytes).	%i ó %d
unsigned unsigned int	Tipo entero básico sin signo. Contiene al menos el rango [0 a + 4,294,967,295], tiene 32 bits de tamaño. (cuatro bytes).	%u ó %i



Tipo	Explicación	Especificador de formato
long long int signed long signed long int	Tipo entero <b>con signo largo</b> . Capaz de contener al menos el rango [-4.61169E+18 a +4.61169E+18]; tiene 64 bits de tamaño.(ocho bytes)	%li
unsigned long unsigned long int	Tipo entero <b>largo sin signo</b> . Capaz de contener al menos el rango [0 a +9.22337E+18]; tiene 64 bits de tamaño.(ocho bytes)	%lu ó %li
long long long long int signed long long signed long long int	Tipo entero <b>largo largo con signo</b> (+/-). Capaz de contener al menos el rango [-8.50706E+37 a +8.50706E+37]. Especificado desde la versión C99 del estándar. tiene 128 bits de tamaño.(dieciseis (16) bytes). Depende de su S.O, versión de C y su procesador.	%lli
unsigned long long unsigned long long int	Tipo entero <b>largo largo sin signo</b> . Contiene al menos el rango [0, +1.70141E+38]; Especificado desde la versión C99 del estándar. tiene 128 bits de tamaño.(dieciseis(16) bytes). Depende de su S.O, versión de C y su procesador.	%llu
float	Tipo de punto flotante real, generalmente denominado tipo de punto flotante de precisión simple. Propiedades reales no especificadas (excepto límites mínimos), sin embargo, en la mayoría de los sistemas, este es <b>el formato de punto flotante binario de precisión simple IEEE 754 (32 bits)</b> . Este formato es requerido por el anexo F opcional "IEC 60559 aritmética de punto flotante". Tiene 32 bits de tamaño. (cuatro bytes). Contiene al menos el rango [3,402823466 E-38 a 3,402823466 E+38];	Conversión de texto: %f %F %g %G %e %E %a %A
double	Tipo de punto flotante real, generalmente denominado tipo de punto flotante de doble precisión. Propiedades reales no especificadas (excepto los límites mínimos), sin embargo, en la mayoría de los sistemas, este es <b>el formato de punto flotante binario de doble precisión IEEE 754 (64 bits)</b> . Este formato es requerido por el anexo F opcional "IEC 60559 aritmética de punto flotante". Tiene 64 bits de tamaño. (ocho bytes). Contiene al menos el rango [1,7976931348623158E-308 a 1,7976931348623158 E+308]	%lf %lF %lg %lG %le %lE %la %lA

Tipo	Explicación	Especificador de formato
long double	Tipo real de punto flotante, generalmente asignado a un formato de número de punto flotante de precisión extendida. Propiedades reales no especificadas. Puede ser un formato de punto flotante de precisión extendida x86 (128 bits en memoria = ocho(8) bytes), el "doble-doble" (128 bits), flotante de precisión cuádruple IEEE 754 formato de punto (128 bits), o lo mismo que doble. Rango con signo entre: $[-1.7E-308(-1.7 \times 10^{-308})$ a $1.7E+308(1.7 \times 10^{+308})$ ó sin signo $[3.4E-4932(3.4 \times 10^{-4932})$ a $3.4E+4932(3.4 \times 10^{+4932})$	<div>%Lf</div> <div>%LF</div> <div>%Lg</div> <div>%LG</div> <div>%Le</div> <div>%LE</div> <div>%La</div> <div>%LA</div>

## MOSTRANDO DATOS POR PANTALLA

Tomado y adaptado de: <https://es.wikipedia.org/wiki/Printf>

Fecha: 2020-04-20

".....

### **printf**

Numerosos lenguajes de programación implementan una función printf (print formatted), para mostrar una cadena(char\*) con formato. Esta, originaria del lenguaje de programación C, tiene la siguiente firma ó prototipo:

```
int printf(const char* formato,...)
```

La cadena formato provee una descripción de la salida, con "placeholders" marcados por caracteres de escape (\n \t...) para especificar la localización relativa y caracteres de control("%") para especificar el tipo de salida que la función debe producir.

Por ejemplo en C:

```
printf("Color %s, numero1 %d, numero2 %05d, real %.2f ", "rojo", 12345, 89, 3.14);
```



Si utilizamos el programa IDE, que se nos ofrece en sitio <https://www.onlinegdb.com> de internet, tendríamos:

```

1 #include <stdio.h>
2
3 int main(){
4     printf("Color %s, numero1 %d, numero2 %05d, real %.2f ", "rojo", 12345, 89, 3.14);
5
6     return 0;
7 }
8

```

input

Color rojo, numero1 12345, numero2 00089, real 3.14

...Program finished with exit code 0

Press ENTER to exit console.

**La función printf retorna el número de caracteres impresos, o un valor negativo si ocurre un error.**

## Operadores aritméticos de C

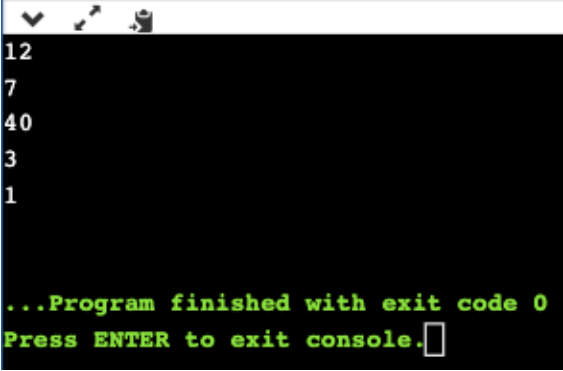
Los operadores aritméticos realizan operaciones aritméticas (matemáticas) sobre valores como la suma, la resta, la multiplicación, etc.

Considere la siguiente tabla en la que se describen los operadores aritméticos de C junto con ejemplos:

Operador	Descripción	Ejemplo
+	utilizado para sumar valores.	5 + 7
-	usado para restar un valor de otro.	10 - 3
*	utilizado para multiplicar dos números	8 * 5
/	utilizado para dividir un valor por otro. La división siempre devolverá el tipo de dato de punto flotante(float)	9 / 3
%	utilizado para devolver el resto(residuo)(remainder) de la división en dos operandos.	9 % 2

Si utilizamos el programa IDE, que se nos ofrece en sitio <https://www.onlinegdb.com> de internet, tendríamos:

```
1 #include <stdio.h>
2 int main(){
3     printf("%i\n",5+7);
4     printf("%i\n",10-3);
5     printf("%i\n",8*5);
6     printf("%i\n",9/3);
7     printf("%i\n",9%2);
8     return 0;
9 }
```



Recuerde que "%i\n" indica que la función printf, imprimirá en formato de tipo **int** un valor y después saltará al próximo renglón y se ubicará el cursor en la primera columna de ese renglón

## Variables

Una de las características más potentes de los lenguajes de programación es la capacidad de manipular variables.

Una variable es una ubicación de memoria que tiene un nombre y un valor (algunos datos).

Cada variable debe tener un nombre único llamado identificador. Es de mucha ayuda pensar en las variables como contenedores que almacenan datos (data) los cuales puede ser cambiados después a lo largo del programa.

<p>Una variable es:</p> <ul style="list-style-type: none"><li>- Un(1) contenedor para almacenar datos</li><li>- Una(1) posición de memoria a la que se le asigna un único nombre y sirve para almacenar datos.</li></ul>	<p>La memoria de una computadora tiene muchos contenedores</p>
	

En el lenguaje C, antes de utilizar una variable, debe declararse, definiendo el tipo de dato que va a contener.

El operador de asignación `=` se usa para asignar un valor a una variable.

El operador igual a, `(=)`, es el más simple de todos los operadores y asigna a la variable del lado izquierdo cualquier variable o resultado del lado derecho siempre y cuando sea del mismo tipo de dato. Ejemplos:

```
int edad = 25;  
float pi = 3.1415926;
```

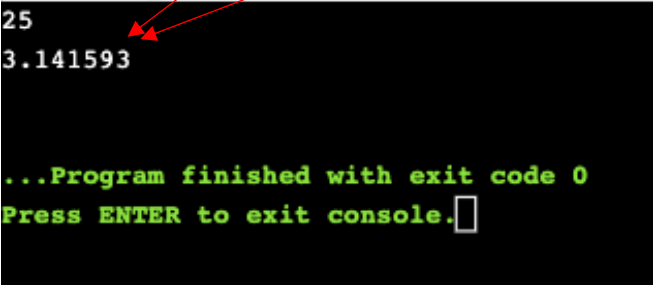


Estos ejemplos muestran dos(2) declaraciones de variables y dos(2) asignaciones.

La primera declara variable una de tipo int, la llama "edad" y le asigna el valor entero(int) 25

La segunda declara variable una de tipo "punto flotante"(float), la llama "pi" y le asigna el valor float 3.1415926

Probemos el siguiente programa:

<pre>1 #include &lt;stdio.h&gt; 2 int main(){ 3     int edad = 25; 4     float pi = 3.1415926; 5     printf("%i\n", edad); 6     printf("%f\n", pi); 7 8     return 0; 9 }</pre> 	<p>Dado que el float es de precisión simple, solo almacena seis(6) dígitos decimales, por eso al imprimir este valor redondea el valor en el sexto dígito decimal.</p>
---	--

Otra forma, de representar una asignación de un valor a una variable es:

edad ← 25  
pi ← 3.1415926

Con la flecha(←) se quiere indicar que el valor, que esta a la derecha se le asigna a la variable que esta a la izquierda.

Otro ejemplo:

```
1 #include <stdio.h>
2 int main(){
3     int ValorPresente;
4     float Interes, ValorFuturo;
5     ValorPresente = 100;
6     Interes = 2.35/100;
7     ValorFuturo = ValorPresente * (1+Interes);
8     printf ("Valor a pagar = %f ", ValorFuturo);
9     return 0;
10 }
```

input

Valor a pagar = 102.349998

...Program finished with exit code 0  
Press ENTER to exit console.

Lo anterior, se puede representar en un diagrama flujo de datos (DFD's), para lo cual los programadores utilizan varias herramientas que tienen símbolos estandarizados por normas internacionales. Ejemplo:

Código fuente (Libreto ó guion)	Diagrama de Flujo de Datos(DFD)
<p><b>ValorPresente <math>\leftarrow</math> 100</b></p> <p><b>Interes <math>\leftarrow</math> 2.35/100</b></p> <p><b>ValorFuturo <math>\leftarrow</math> ValorPresente*(1+Interes)</b></p> <p><b>printf ("Valor a pagar= %f ", ValorFuturo)</b></p>	<pre>graph TD; INICIO([INICIO]) --&gt; P1[ValorPresente &lt;-- 100]; P1 --&gt; P2[Interes &lt;-- 2.35/100]; P2 --&gt; P3[ValorFuturo &lt;-- ValorPresente*(1+Interes)]; P3 --&gt; O1{"Valor a pagar =", ValorFuturo}; O1 --&gt; FIN([FIN]);</pre>

Tomado a adaptado de:  
[https://es.wikipedia.org/wiki/Diagrama\\_de\\_flujo](https://es.wikipedia.org/wiki/Diagrama_de_flujo)  
Fecha: 2020-06-23

“...**Diagrama de flujo**

El diagrama de flujo o flujograma o diagrama de actividades es la representación gráfica de un algoritmo o proceso. Se utiliza en disciplinas como programación, economía, procesos industriales y psicología cognitiva.

... **Historia**

La paternidad del diagrama de flujo es en principio algo difusa. El método estructurado para documentar gráficamente un proceso como un flujo de pasos sucesivos y alternativos, el "proceso de diagrama de flujo", fue expuesto por Frank Gilbreth, en la Sociedad Americana de Ingenieros Mecánicos (ASME), en 1921, bajo el enunciado de "Proceso de Gráficas-Primeros pasos para encontrar el mejor modo". Estas herramientas de Gilbreth rápidamente encontraron sitio en los programas de ingeniería industrial.

Al principio de los 30, un ingeniero industrial, Allan H. Mogensen comenzó la formación de personas de negocios en Lake Placid, Nueva York, incluyendo el uso del diagrama de flujo. Art Spinanger, asistente a las clases de Mogesen, utilizó las herramientas en su trabajo en Procter & Gamble, donde desarrolló su “Programa Metódico de Cambios por Etapas”. Otro asistente al grupo de graduados en 1944, Ben S. Graham, director de ingeniería de Formcraft Standard Register Corporation, adaptó la gráfica de flujo de procesos al tratamiento de la información en su empresa. Y desarrolló la gráfica del proceso de múltiples flujos en múltiples pantallas, documentos, y sus relaciones. En 1947, ASME adoptó un conjunto de símbolos derivados de la obra original de Gilbreth como Norma ASME para los gráficos de procesos (preparada Mishad, Ramsan y Raiaan).

**Sin embargo, según explica Douglas Hartree fueron originalmente Herman Goldstine y John von Neumann quienes desarrollaron el diagrama de flujo (inicialmente llamado "diagrama") para planificar los programas de ordenador.**



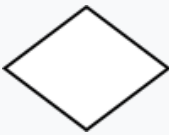



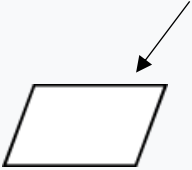




En la **década de 1970** la popularidad de los diagramas de flujo como método propio de la informática disminuyó, con el nuevo hardware y los nuevos lenguajes de programación de tercera generación. Y por otra parte se convirtieron en instrumentos comunes en el mundo empresarial. Son una expresión concisa, legible y práctica de algoritmos. Actualmente se aplican en muchos campos del conocimiento, especialmente como simplificación y expresión lógica de procesos, etc.

...El Instituto Nacional Estadounidense de Estándares (ANSI, por su siglas en inglés) estableció estándares para los diagramas de flujo y sus símbolos en los años 1960s.<sup>3</sup> La Organización Internacional de Normalización (ISO, por sus siglas en inglés) adoptó los símbolos ANSI en 1970. <sup>4</sup> El estándar actual, ISO 5807, fue revisado en....” **2019**

Tomado y adaptado de:  
<https://www.iso.org/standard/11955.html>  
Fecha:2020-06-23

“...ISO 5807:1985 Procesamiento de la información: símbolos de la documentación y convenciones para datos, diagramas de flujo para programación de computadoras y del sistema, diagramas de red del programa y diagramas de recursos del sistema ... ESTA NORMA FUE REVISADA Y CONFIRMADA POR ÚLTIMA EN <b>2019</b> . POR LO TANTO, ESTA VERSIÓN SIGUE ACTUAL ... ICS: 35.080 Software 01.080.50 Símbolos gráficos para uso en tecnología de la información y telecomunicaciones y en la documentación técnica relevante ..”	“...ISO 5807:1985 Information processing — Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts....THIS STANDARD WAS LAST REVIEWED AND CONFIRMED IN <b>2019</b> . THEREFORE THIS VERSION REMAINS CURRENT... ICS : 35.080 Software 01.080.50 Graphical symbols for use on information technology and telecommunications technical drawings and in relevant technical documentation..”
---	--



Forma ANSI/ISO	Otras formas	Descripción
		Terminal: Indica el inicio o fin de un programa o subprocesos. Se representa como un "Estadio", ovalo o rectángulo redondeado. Usualmente contienen la palabra "Inicio" o "Fin", o alguna otra frase señalando el inicio o fin de un proceso, como "presentar consulta" o "recibir producto".
		Proceso: Representa un conjunto de operaciones que cambiar el valor, forma o ubicación de datos. Representado como un rectángulo.
		Decisión : Muestra una operación condicional que determina cuál de los dos caminos tomará el programa. La operación es comunmente una pregunta de sí/no o una prueba de verdadero/falso. Representada como un rombo.
	  	Entrada: Indica el proceso de hacer entrar datos(Normalmente desde el teclado del ordenador). Representado como un paralelogramo y/o un teclado de computador visto lateralmente
	  	Salida : Indica el proceso de hacer sacar o mostrar datos, desde el ordenador normalmente hacia la pantalla del ordenador. Representado como un paralelogramo.y/o una pantalla de ordenador(modelo viejo no plana) vista lateralmente.

Recomendación: Utilicen la herramienta gratuita, ofrecida en la página web:  
<https://app.lucidchart.com/es/users/registerLevel#/pricing>

Por favor registrarse en la opción "Gratis"



Tomado de:

<https://www.aulafacil.com/cursos/programacion/lenguaje-de-programacion-c/primer-programa-en-c-l16525>

Fecha: 2020-04-20

"...

## EJEMPLOS DE DECLARACIÓN DE VARIABLES SEGÚN SU TIPO

DECLARACION CORRECTA	ERRORES FRECUENTES
<pre>int num; int x=7; float r; float b=3.198; char q; char r= 'A';</pre>	<pre>int pp = 10.32;</pre> <p>Nos dará error, porque el valor es un número con decimales, por lo que tendremos que utilizar un double o float.</p>

..Vamos a aprender ahora a crear variables, darles valor y mostrar ese valor por pantalla. De esta forma sabemos qué valor contiene cada variable.

Haremos un programa muy simple en el que crearemos dos(2) variables de diferente tipo, le asignaremos un valor y lo mostraremos por pantalla.

El código es el siguiente:

```
#include <stdio.h>  
int main(){  
    int num;           //Creamos la variable tipo entero (int)  
    char character;    //Creamos la variable tipo char  
    //Inicializamos las variables  
    num = 3;  
    character = 'h';  
    //Mostramos el valor de esas variables por pantalla  
    printf("El numero es: %i\n", num);  
    printf("El caracter es: %c\n", character);  
    return 0;  
}
```

Recordad los comentarios que voy poniendo en el código, son muy útiles para que sepáis qué está realizando cada instrucción del programa.

Cabe decir que la creación y asignación de las variables se pueden hacer en una sola línea también, eso lo podéis hacer como queráis. En este caso, lo hice aparte para que se vea un poco más claro, pero también se puede hacer lo siguiente:

```
int num = 10;
```

Es lo mismo. El programa funcionará igualmente.

```
printf("<texto_a_mostrar>", <variables_a_mostrar>)
```

Entre comillas se escribe el texto que se mostrará, y cuando se quiera mostrar una variable se hace con el "%". La letra que lleva detrás cambiará dependiendo del tipo de dato que utilicemos. En la siguiente tabla veréis las letras para cada tipo:

int -> %i      char -> %c      float -> %f      double -> %f

Para los dos últimos se utiliza el mismo.

Bueno, y retomando la muestra de los valores, una vez hayamos escrito el texto entre las comillas luego vendrá una "," para separar, y a continuación el nombre de la variable a mostrar... como podéis apreciar en el código.

## EL % DE LOS ESPECIFICADORES DE FORMATO

Los % especificadores que puede usar en el lenguaje C, estandar ANSI, son:

%c	<b>char</b> carácter
%i	<b>int</b> entero con signo
%d	<b>int</b> entero con signo
%e	<b>float</b> de punto flotante ó real con decimales ó exponencial
%f	<b>float</b> de punto flotante ó real con decimales (con signo)
%o	<b>int</b> entero octal (base 8)
%p	direccion de puntero
%s	<b>char[]</b> string ó cadena de caracteres
%u	<b>int</b> entero sin signo
%x	<b>int</b> entero hexadecimal (base 16)
%lu	<b>unsigned long</b> entero largo sin signo
%ld	<b>long</b> entero largo con signo

Ejemplo:

Codigo -Programa	Pantalla	
<pre>#include&lt;stdio.h&gt; int main() {     int a,b;     float c,d;      a = 15;     b = a / 2;     printf("%d \n ", b);     printf("%3d \n ",b);     printf("%03d \n ",b);      c = 15.3;     d = c / 3;     printf("%3.2f \n ",d);     return 0; }</pre>	<pre>7  7 007  5.10</pre>	<p><i>Formato <b>d</b>ecimal</i></p> <p><i>Formato <b>d</b>ecimal de tres(3) posiciones</i></p> <p><i>Formato <b>d</b>ecimal de tres(3) posiciones con llenado de ceros</i></p> <p><i>Formato <b>f</b>loat 3 enteros punto 2 decimales</i></p>

Tomado de:

<https://www.codingunit.com/printf-format-specifiers-format-conversions-and-formatted-output>

Fecha: 25.Sep.2018

" ...

## FORMATO DE CADENAS DE CARACTERES **string**

Codigo -Programa	Pantalla
<pre>#include&lt;stdio.h&gt; int main() {     printf(":%s:\n", "Hola, mundo !");     printf(":%15s:\n", "Hola, mundo !");     printf(":%.10s:\n", "Hola, mundo !");     printf(":%-10s:\n", "Hola, mundo !");     printf(":%-15s:\n", "Hola, mundo !");     printf(":%.15s:\n", "Hola, mundo !");     printf(":%15.10s:\n", "Hola, mundo !");     printf(":%-15.10s:\n", "Hola, mundo !");     return 0; }</pre>	<pre> :Hola, mundo !: :  Hola, mundo !: :Hola, mund: :Hola, mundo !: :Hola, mundo ! : :Hola, mundo !: :  Hola, mund: :Hola, mund  :</pre>

Esto es:

**`printf (":% s: \n", "Hola, mundo!");`** La sentencia imprime la cadena (no ocurre nada especial).

**`printf (":% 15s: \n", "Hola, mundo!");`** La sentencia imprime la cadena, pero imprime 15 caracteres. Si la cadena es más pequeña, las posiciones "vacías" se llenarán con "espacios en blanco".

**`printf (":% .10s: \n", "Hola, mundo!");`** La sentencia imprime la cadena, pero imprime solo 10 caracteres de la cadena.

**`printf (":% - 10s: \n", "Hola, mundo!");`** La sentencia imprime la cadena, pero imprime al menos 10 caracteres. Si la cadena es más pequeña, se agrega "espacio en blanco" al final. (Ver el siguiente ejemplo)

**`printf (":% - 15s: \n", "Hola, mundo!");`** La sentencia imprime la cadena, pero imprime al menos 15 caracteres. La cadena en este caso es más corta que los 15 caracteres definidos, por lo tanto, se agrega "espacio en blanco" al final (definido por el signo menos).

**`printf (":% 15s: \n", "Hola, mundo!");`** La sentencia imprime la cadena, pero imprime solo 15 caracteres de la cadena. En este caso, la cadena es más corta que 15, por lo que se imprime toda la cadena.

**`printf (":% 15.10s: \n", "Hola, mundo!");`** La sentencia imprime la cadena, pero imprime 15 caracteres. Si la secuencia es más pequeña, las posiciones "vacías" se rellenarán con "espacios en blanco". Pero solo imprimirá un máximo de 10 caracteres, por lo que solo se imprimirá una parte de la cadena nueva (cadena anterior más las posiciones en blanco).

**`printf (":% - 15.10s: \n", "Hola, mundo!");`** la sentencia imprime la cadena, pero hace exactamente lo mismo que la declaración anterior, acepta el "espacio en blanco" se agrega al final.

Tomado de:

<https://en.wikipedia.org/wiki/Sizeof>

Fecha: Jueves 27.Sep.2018

## **OPERADOR UNARIO sizeof**

Muchos programas deben conocer el tamaño de almacenamiento de un tipo de datos en particular. Aunque para cualquier implementación dada de C o C ++ el tamaño de un tipo de datos particular es constante, los tamaños de incluso los tipos primitivos en C y C ++ pueden definirse de manera diferente para las diferentes plataformas de implementación.

En los lenguajes de programación C y C ++ , el **operador unario sizeof** genera el tamaño de una variable o de tipo de datos, medida en el número de bytes ú octetos( de 8-bits) como unidades de almacenamiento necesarios para el tipo de dato. En consecuencia, se garantiza que el tamaño de construcción de (char) sea de un(1) byte ú octeto( de 8-bits). sizeof no es una función, es un **operador unario**.

El **operador unario sizeof** tiene un solo operando, que es una variable o una conversión de tipo de datos, que iría entre paréntesis. Los tipos de datos pueden no ser solo tipos primitivos , como tipos enteros(int) y de coma flotante(float), sino también tipos de punteros y tipos de datos compuestos ( uniones , estructuras, etc).

En general, no es seguro asumir el tamaño de ningún tipo de datos. Por ejemplo, aunque la mayoría de las implementaciones de C y C ++ en sistemas de **32 bits** definen tipo int como cuatro bytes ú octetos (de 8 bits), este tamaño puede cambiar cuando el código se transporta a un sistema diferente, rompiendo el código. Vgr: 8 bits, 16 bits, 64 bits, etc. La excepción a esto es el tipo de datos es el tipo **char**, que siempre tiene el tamaño un(1) byte ú octeto(de 8 bits) en cualquier implementación de C que cumpla con los estándares.

El **operador unario sizeof** calcula el espacio de almacenamiento de memoria requerido de su operando. El operando se escribe siguiendo la palabra reservada **sizeof** y puede ser el símbolo de un espacio de almacenamiento, por ejemplo, una variable, un nombre de tipo o una expresión. Si es un nombre de tipo, debe estar entre paréntesis. El resultado de la operación es el tamaño del operando en bytes o el tamaño de la representación de la memoria. Para las expresiones, evalúa el tamaño de representación para el tipo que resultaría de la evaluación de la expresión, que no se realiza.

Ejemplo:

Código -Programa
<pre>#include&lt;stdio.h&gt; int main () {     printf("%i",sizeof(int));     return 0; }</pre>

Evalúa el tamaño de un tipo de dato **int**

Pantalla

4
---

Cuatro bytes ú octetos(de 8 bits)

Ejemplo:

Código -Programa
<pre>#include&lt;stdio.h&gt; int main () {     printf("%i",sizeof(char));     return 0; }</pre>

Evalúa el tamaño de un tipo de dato **char**

Pantalla

1
---

Un byte ú octeto(de 8 bits)

Código -Programa
<pre>#include&lt;stdio.h&gt; int main () {     char cad='A';     printf("%i",sizeof(cad));     return 0; }</pre>

Evalúa el tamaño de una variable de tipo de dato **char**

Pantalla

1

Un byte ú octeto(de 8 bits)

---

Código -Programa
<pre>#include&lt;stdio.h&gt; int main () {     float num=3.1415926;     printf("%i",sizeof(num));     return 0; }</pre>

Evalúa el tamaño de una variable de tipo de dato **float**

Pantalla

4

Cuatro(4) bytes ú octetos(de 8 bits)

---

Código -Programa
<pre>#include&lt;stdio.h&gt;  int main () {     int a=32000;     printf("%ld",sizeof(a*10000000000));     return 0; }</pre>

Evalúa el tamaño en bytes del resultado de una expresión. En este caso la expresión necesitaría ocho(8) bytes, lo que sería un tipo de dato **long int**

Pantalla

8

Ocho(8) bytes ú octetos(de 8 bits)



# Bit's y Byte's

Hablemos nuevamente sobre los bit's y los byte's


Cuando declaramos variables, estamos pensando en los valores que podrían llegar a contener, pero que pasa si tratamos de almacenar un valor mayor al que pueden llegar a contener; ejemplo:

Código lenguaje C	Pantalla del ordenador
<pre>#include &lt;stdio.h&gt; int main(){     int dato = 2147483647;     printf("%i",dato);      return 0; }</pre>	2147483647

En el anterior código en lenguaje C, hemos creado la variable **dato** de tipo **int** y le hemos asignado el máximo valor que puede almacenar una variable de tipo **int** y luego imprimimos en la pantalla del ordenador en formato de tipo entero( "%i" ) ese valor.


Ahora observemos ¿qué pasa si almaceno un valor con una(1) unidad más grande?; así:

`int dato = 2147483648;`

Código lenguaje C	Pantalla del ordenador
<pre>#include &lt;stdio.h&gt; int main(){     int dato = 2147483648;     printf("%i",dato);      return 0; }</pre>	 -2147483648

Se presenta un desbordamiento(**OVERFLOW**) de la capacidad de la variable y toma un valor **negativo**

Probemos este otro programa:

Código lenguaje C	Pantalla del ordenador
<pre>#include &lt;stdio.h&gt; int main(){     int dato = 21474836479;     printf("%i",dato);     return 0; }</pre>	 -2147483647

Lo que observamos es que empieza a sumar sobre el primer número negativo, pero almacena igualmente un número **negativo**  
 Probemos este otro programa:

Código lenguaje C	Pantalla del ordenador
<pre>#include &lt;stdio.h&gt; int main(){     int dato = 2147483647+ 2147483649;     printf("%i",dato);      return 0; }</pre>	warning: <b>overflow</b> in implicit constant conversion [-Woverflow]  <b>0</b>

Dado de 2147483647+ 2147483649 = **4294967296** = **2<sup>32</sup>**, podríamos escribir:

Código lenguaje C	Pantalla del ordenador
<pre>#include &lt;stdio.h&gt; int main(){     int dato = <b>4294967296</b>;     printf("%i",dato);      return 0; }</pre>	warning: <b>overflow</b> in implicit constant conversion [-Woverflow]  <b>0</b>

Esto es que el lenguaje C, nos advierte que se presento un desbordamiento de la capacidad de la variable **dato**. Que se necesito de otro byte para almacenar este dato(todos los bits de estos cuatro bytes se colocaron en cero(0)).

Byte menos significativo

Primer byte

0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
2	2	2	2	2	2	2	2

Segundo byte

0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
2	2	2	2	2	2	2	2

Tercer byte

0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
2	2	2	2	2	2	2	2

Byte más significativo

Cuarto byte

0	0	0	0	0	0	0	0
31	30	29	28	27	26	25	24
2	2	2	2	2	2	2	2

bit de signo

QUINTO BYTE

0	0	0	0	0	0	0	1
39	38	37	36	35	34	33	32
2	2	2	2	2	2	2	2

4,294,967,296

Binary

Code

Decimal

BCD



La solución entonces es declarar a la variable **dato**, de tipo **long int** (ocho(8) bytes), de tal forma que pueda contener un número más grande e imprimirlo en formato long int ("%li" ); así:

Código lenguaje C	Pantalla del ordenador
<pre>#include &lt;stdio.h&gt; int main(){     long int dato = 4294967296;     printf("%li ",dato);      return 0; }</pre>	4294967296

Recordemos que el tipo de dato "long int" es de ocho(8) bytes y puede almacenar datos en un rango de:  
[- 4.61169E+18 a +4.61169E+18 ].

**Tarea:** Leer la información asociada a los link´s:

<https://www.rtve.es/noticias/20140604/error-software-convirtio-lanzamiento-espacial-carisimos-fuegos-artificiales/948262.shtml>

<http://wiki.us.es/juanan/wakka.php?wakka=AccidenteAriane5>

Hay que explicar que paso y como podría haberse resuelto el problema.

-----  
*Tomado y adaptado de:*

<https://docs.microsoft.com/en-us/cpp/c-language/cpp-integer-limits?view=vs-2019>

*Fecha: 2020-06-20*

## "...Límites de enteros C

Límites en las constantes enteras

CONSTANTE	SENTIDO	VALOR
CHAR_BIT	Número de bits en la variable más pequeña que no es un campo de bits.	8
SCHAR_MIN	Valor mínimo para una variable de tipo con signo char .	-128
SCHAR_MAX	Valor máximo para una variable de tipo firmado char .	127
UCHAR_MAX	Valor máximo para una variable de tipo unsigned char .	255 (0xff)
CHAR_MIN	Valor mínimo para una variable de tipo char .	-128
CHAR_MAX	Valor máximo para una variable de tipo char .	127
MB_LEN_MAX	Número máximo de bytes en una constante de varios caracteres.	5 5
SHRT_MIN	Valor mínimo para una variable de tipo short .	-32768
SHRT_MAX	Valor máximo para una variable de tipo short .	32767

CONSTANTE	SENTIDO	VALOR
USHRT_MAX	Valor máximo para una variable de tipo sin signo corto .	65535
INT_MIN	Valor mínimo para una variable de tipo int .	-2147483647 – 1
INT_MAX	Valor máximo para una variable de tipo int .	2147483647
UINT_MAX	Valor máximo para una variable de tipo unsigned int .	4294967295
LONG_MIN	Valor mínimo para una variable de tipo largo .	-2147483647 – 1
LONG_MAX	Valor máximo para una variable de tipo largo .	2147483647
ULONG_MAX	Valor máximo para una variable de tipo sin signo largo .	4294967295
LLONG_MIN	Valor mínimo para una variable de tipo largo largo .	-9,223,372,036,854,775,807 – 1
LLONG_MAX	Valor máximo para una variable de tipo largo largo .	9.223.372.036.854.775.807
ULLONG_MAX	Valor máximo para una variable de tipo unsigned long long .	18,446,744,073,709,551,615

Si un valor excede la representación entera más grande, el compilador de Microsoft genera un error.....”

Estas constantes, son extraidas si se incluye la librería <limits.h>. Ejemplo:

Código en lenguaje C	Pantalla del ordenador
<pre>#include &lt;stdio.h&gt; #include&lt;limits.h&gt; int main(){     printf("%i \n",CHAR_BIT);     printf("%i \n",SCHAR_MIN);     printf("%i \n",SCHAR_MAX);     printf("%li \n",LONG_MIN);     return 0; }</pre>	<pre>8 -128 127 -9223372036854775808</pre>

**Tarea:** Construir un programa en lenguaje C, que imprima todas las constantes de los limites de los enteros en el lenguaje C que su computadora y/o su versión del lenguaje instalado en su computadora.

Su código y los resultados visualizados en pantalla, debe mostrarlos a dos columnas. La primera con el “Código en lenguaje C” y la segunda columna con lo que mostró en la “Pantalla del ordenador” su código al ser ejecutado.

Código en lenguaje C	Pantalla del ordenador