



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor:* Alejandro Esteban Pimentel Alarcón

*Asignatura:* Fundamentos de Programación

*Grupo:* 135

*No de Práctica(s):* 7

*Integrante(s):* Torres Alcántara Alan Eliezer

*No. de Equipo de  
cómputo empleado:* Somalia 42

*No. de Lista o* 9032

*Semestre:* 2020 - 1

*Fecha de entrega:* 03/10/2019

*Observaciones:* Bien, pero ten cuidado de no utilizar variables  
a las que no les hayas asignado un valor antes.

**CALIFICACIÓN:** 9

# Fundamentos de Lenguaje C

Alejandro Pimentel  
[profpimentel9@gmail.com](mailto:profpimentel9@gmail.com)

Facultad de Ingeniería  
UNAM  
<https://qr.go.page.link/aeaVL>



## Objetivo

Elaborar 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.

# Programas en lenguaje C

C es un lenguaje de programación originalmente desarrollado por Dennis Ritchie entre 1969 y 1972 en los Laboratorios Bell como evolución del anterior lenguaje B. Al igual que B, es un lenguaje orientado a la implementación de sistemas operativos, en específico para Unix, C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistema, aunque también se utiliza para crear aplicaciones.

En 1978, Ritchie y Brian Kernighan publicaron la primera edición de “El lenguaje de programación C”, también conocido como la biblia de C. Este libro fue durante años la especificación informal del lenguaje.

El C de Kernighan y Ritchie es el subconjunto más básico del lenguaje que un compilador puede soportar. Durante muchos años, fue considerado "el mínimo común denominador" en el que los programadores debían programar cuando deseaban que sus programas fueran transportables, pues no todos los compiladores soportaban completamente ANSI.

## Tipos de variables

DATA TYPE	MEMORY (BYTES)	RANGE
short int	2	-32,768 to 32,767
unsigned short int	2	0 to 65,535
unsigned int	4	0 to 4,294,967,295
int	4	-2,147,483,648 to 2,147,483,647
long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295
long long int	8	$-(2^{63})$ to $(2^{63})-1$
unsigned long long int	8	0 to 18,446,744,073,709,551,615

Todos los programas necesitan, en algún momento, almacenar números o datos ingresados por el usuario. Estos datos son almacenados en variables, y en C++ como en otros lenguajes estas variables deben tener un tipo. Existen varios tipos de variables, y cada uno corresponde a un tamaño máximo de un número, un carácter o incluso una verdad, cuanto mayor sea el número que pueda admitir, más espacio en memoria ocupará.

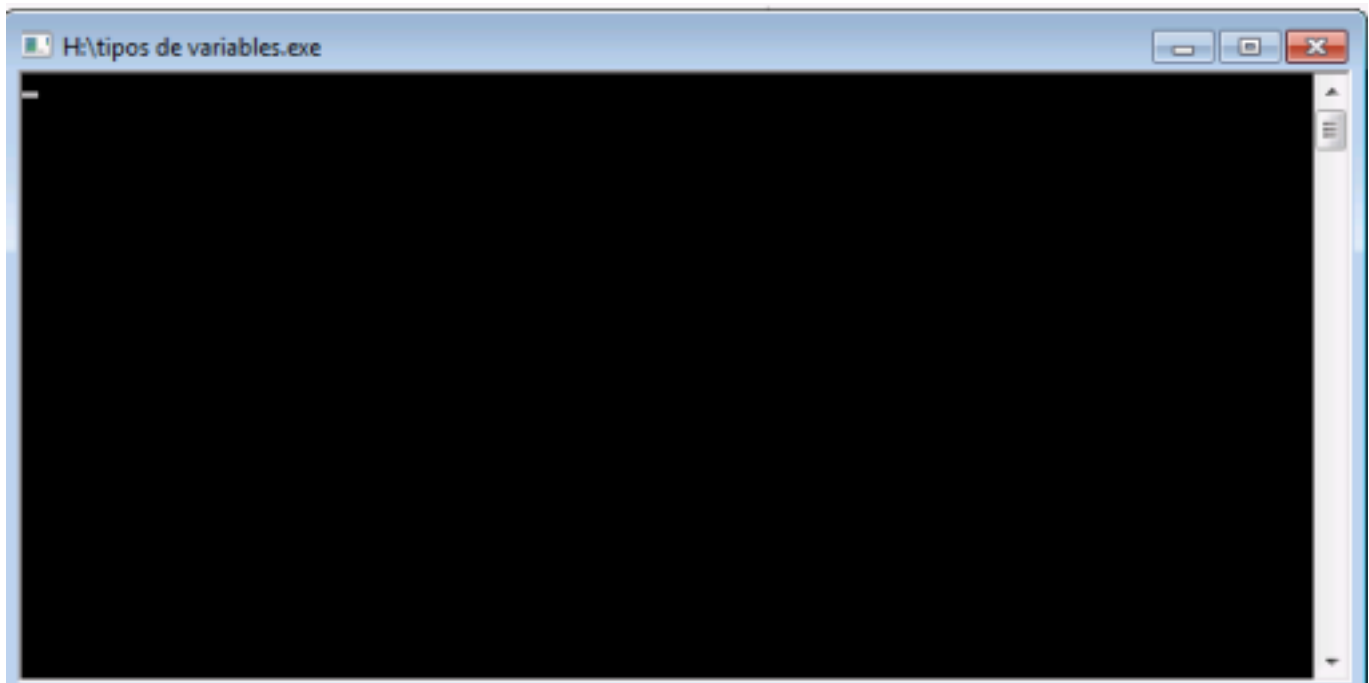
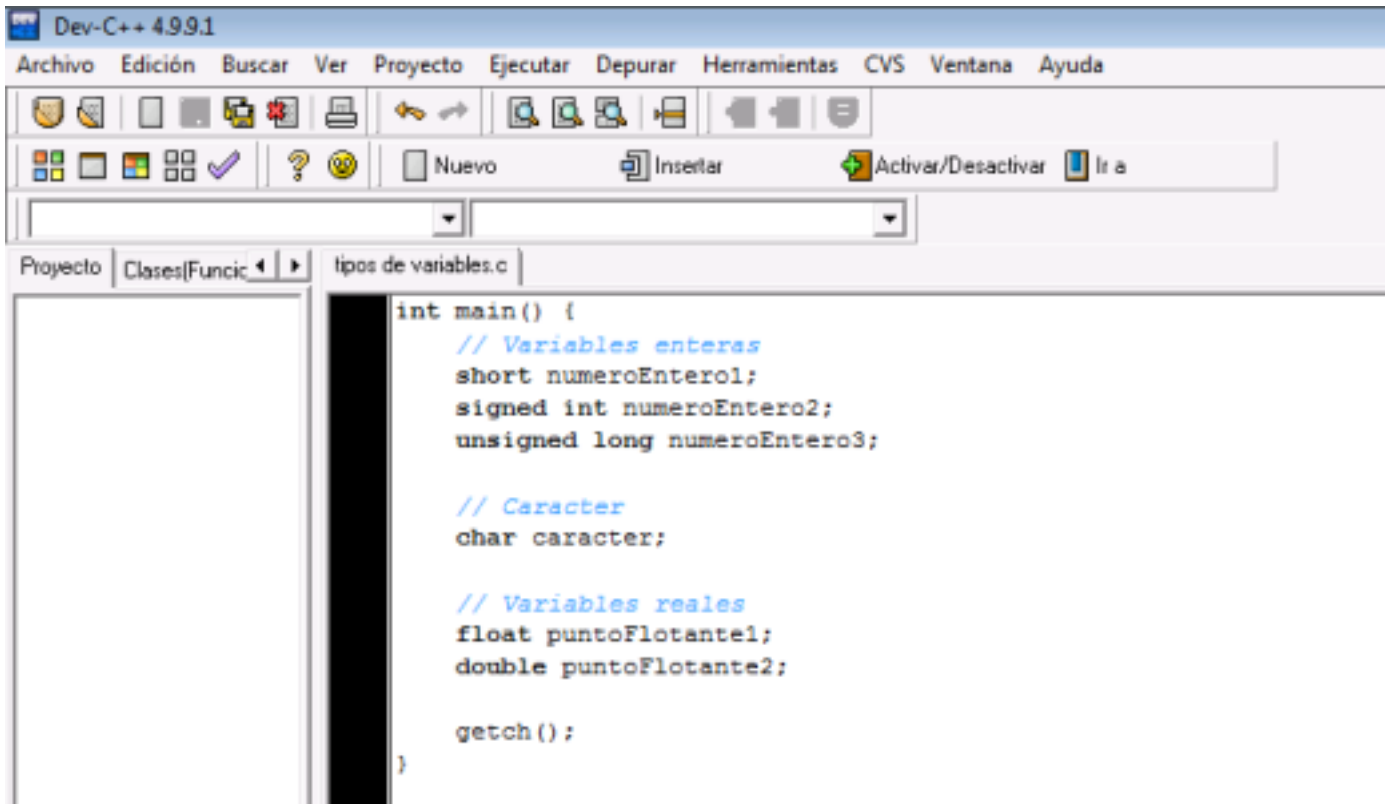
## Tipos de variables

Para los reales, se tienen también diferentes tipos de variables que asignan más bits para tener mayor rango y mayor precisión. Las variables reales siempre poseen signo.

<i>Tipo</i>	<i>Bits</i>	<i>Valor Mínimo</i>	<i>Valor Máximo</i>
<i>float</i>	32	3.4 E-38	3.4 E38
<i>double</i>	64	1.7 E-308	1.7 E308
<i>long double</i>	80	3.4 E-4932	3.4 E4932

## Tipos de variables

```
int main() {  
  
    // Variables enteras  
    short numeroEntero1;  
    signed int numeroEntero2;  
    unsigned long numeroEntero3;  
  
    // Caracter  
    char caracter;  
  
    // Variables reales  
    float puntoFlotante1;  
    double puntoFlotante2;  
  
    return 0;  
}
```



Al abrir el programa, la pantalla aparece en negro, ya que no establecimos al programa que imprimiera en pantalla o que leyera datos. Utilicé getch en vez de return 0 por mi PC que utiliza sistema operativo de Windows.

## Mostrar y Leer

<i>Tipo de dato</i>	<i>Especificador de formato</i>
<i>Entero</i>	%d, %i, %ld, %li, %o, %x
<i>Flotante</i>	%f, %lf, %e, %g
<i>Carácter</i>	%c, %d, %i, %o, %x
<i>Cadena de caracteres</i>	%s

## Mostrar y Leer

```
#include <stdio.h>

int main() {

    //Declaramos variables a leer
    int numeroEntrada;
    double realEntrada;

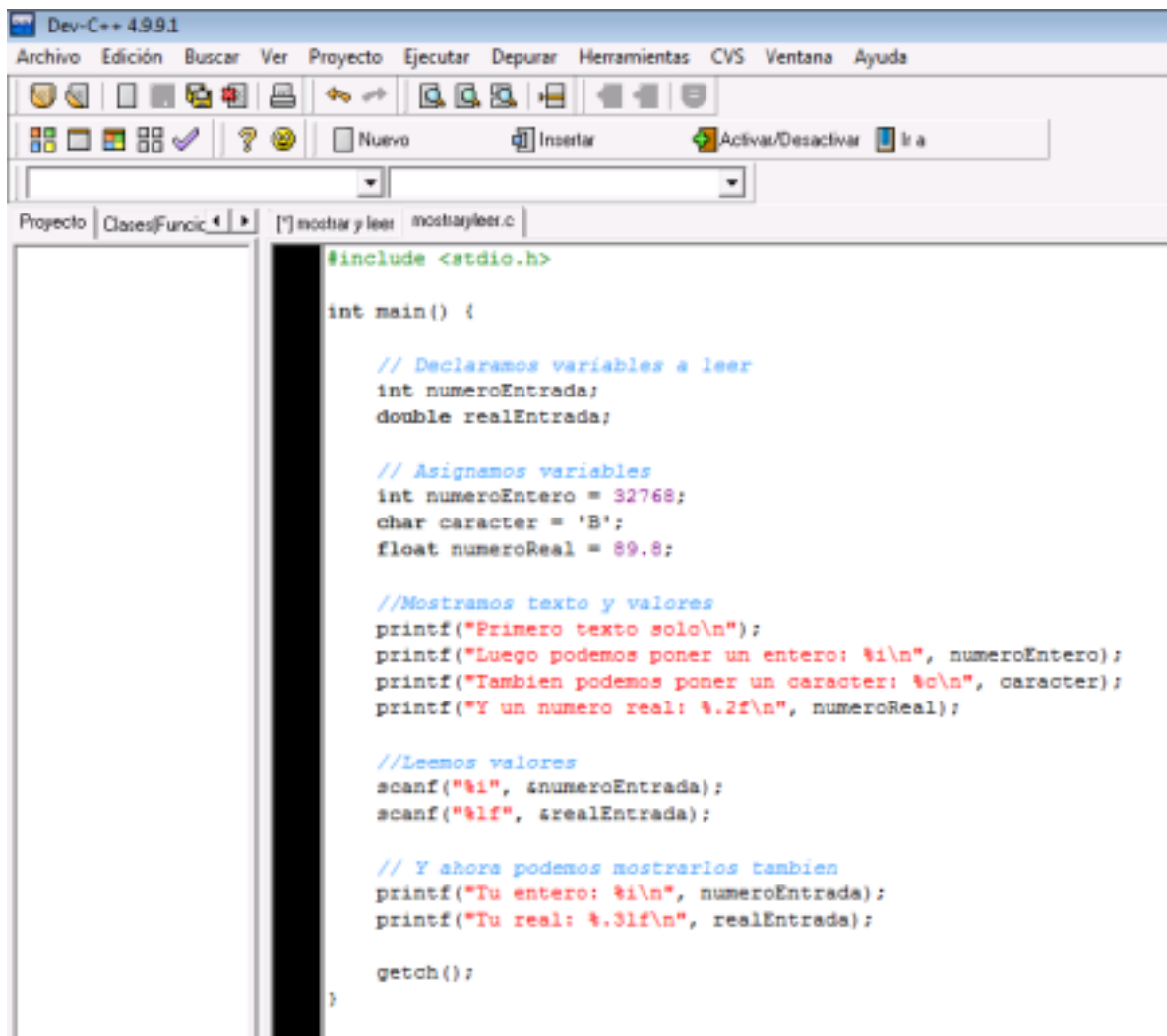
    // Asignamos variables
    int numeroEntero = 32768;
    char caracter = 'B';
    float numeroReal = 89.8;

    // Mostramos texto y valores
    printf("Primero texto solo\n");
    printf("Luego podemos poner un entero: %i\n", numeroEntero);
    printf("También podemos poner un caracter: %c\n", caracter);
    printf("Y un numero real: %.2f\n", numeroReal);

    // Leemos valores
    scanf("%i", &numeroEntrada);
    scanf("%lf", &realEntrada);

    // Y ahora podemos mostrarlos también
    printf("Tu entero: %i\n", numeroEntrada);
    printf("Tu real: %.3lf\n", realEntrada);

    return 0;
}
```



The image shows the Dev-C++ 4.9.9.1 IDE interface. The menu bar includes Archivo, Edición, Buscar, Ver, Proyecto, Ejecutar, Depurar, Herramientas, CVS, Ventana, and Ayuda. The toolbar contains icons for file operations and development tools. The project explorer on the left shows a project named 'mostrar y leer' with a file 'mostrar y leer.c'. The main editor window displays the following C code:

```
#include <stdio.h>

int main() {

    // Declaramos variables a leer
    int numeroEntrada;
    double realEntrada;

    // Asignamos variables
    int numeroEntero = 32768;
    char caracter = 'B';
    float numeroReal = 89.8;

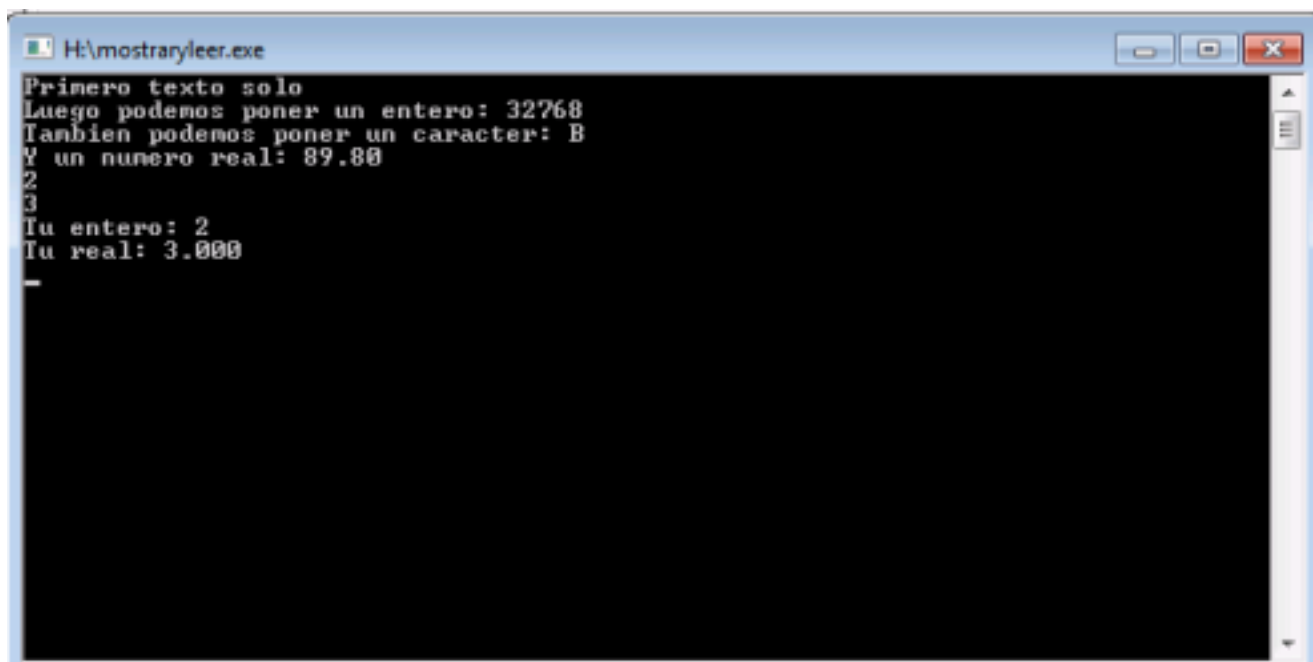
    //Mostramos texto y valores
    printf("Primero texto solo\n");
    printf("Luego podemos poner un entero: %i\n", numeroEntero);
    printf("Tambien podemos poner un caracter: %c\n", caracter);
    printf("Y un numero real: %.2f\n", numeroReal);

    //Leemos valores
    scanf("%i", &numeroEntrada);
    scanf("%lf", &realEntrada);

    // Y ahora podemos mostrarlos tambien
    printf("Tu entero: %i\n", numeroEntrada);
    printf("Tu real: %.3lf\n", realEntrada);

    getch();
}
```

Al correr el programa nos pidió dos números de entrada y nos entregó dos números de salida



The image shows the execution window of the program, titled 'H:\mostraryleer.exe'. The output is as follows:

```
Primero texto solo
Luego podemos poner un entero: 32768
Tambien podemos poner un caracter: B
Y un numero real: 89.80
2
3
Tu entero: 2
Tu real: 3.000
-
```

## Operadores

<i>Operador</i>	<i>Operación</i>	<i>Uso</i>	<i>Resultado</i>
+	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

## Operadores

```
#include <stdio.h>

int main() {
    int dos, tres, cuatro, cinco;
    double resultado;

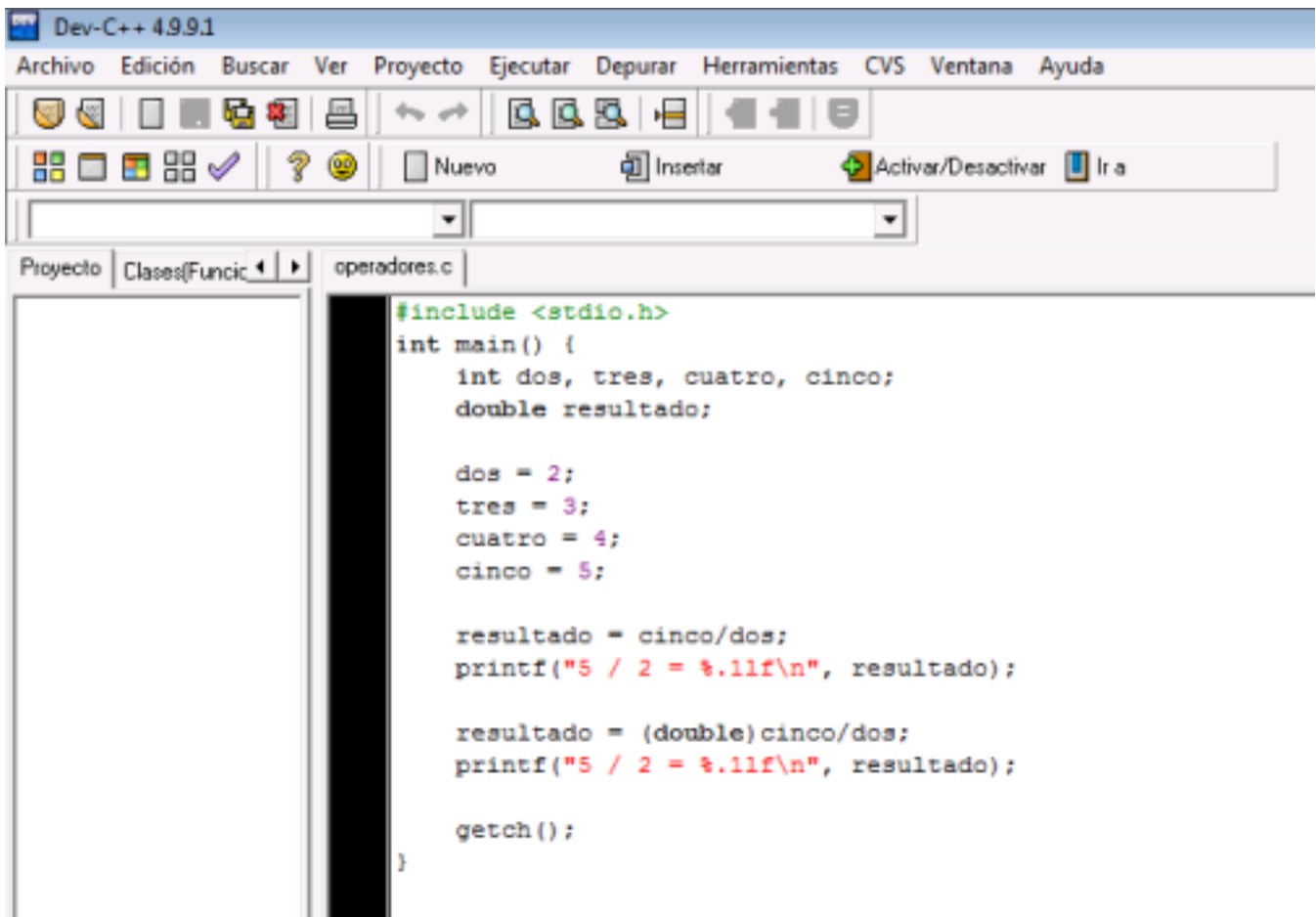
    dos = 2;
    tres = 3;
    cuatro = 4;
    cinco = 5;

    resultado = cinco/dos;
    printf("5 / 2 = %.1lf\n", resultado);

    resultado = (double)cinco/dos;
    printf("5 / 2 = %.1lf\n", resultado);

    return 0;
}
```





```
#include <stdio.h>

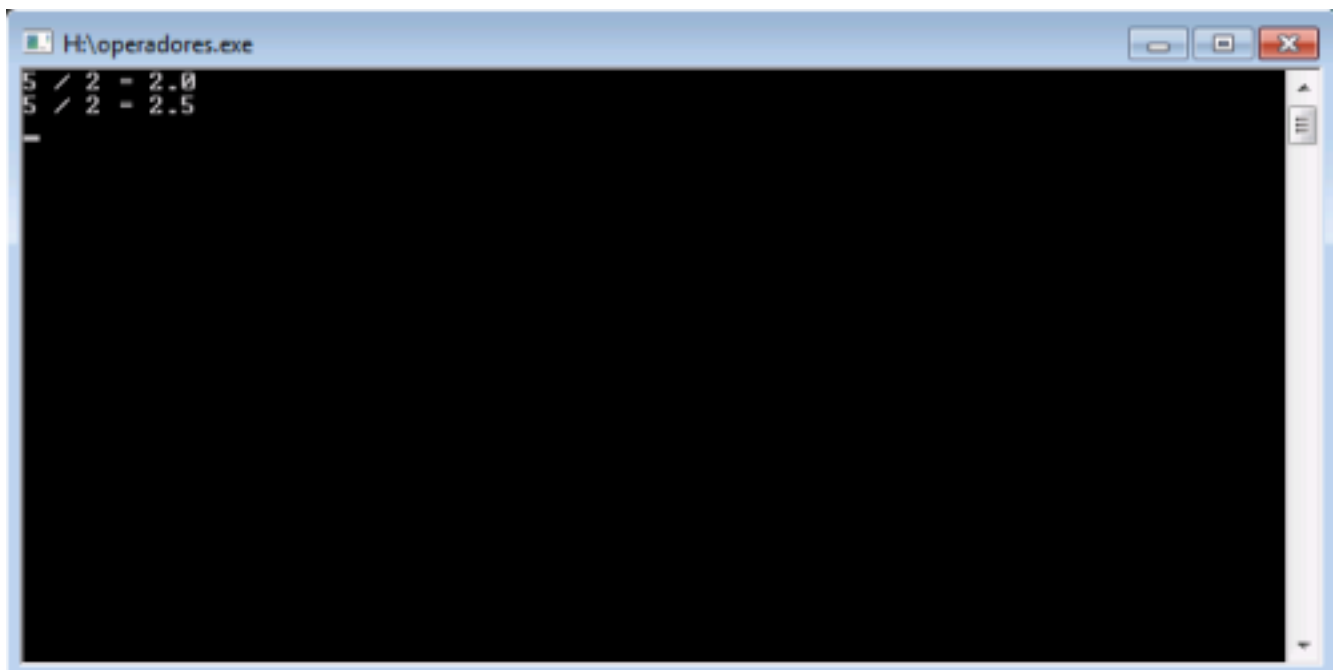
int main() {
    int dos, tres, cuatro, cinco;
    double resultado;

    dos = 2;
    tres = 3;
    cuatro = 4;
    cinco = 5;

    resultado = cinco/dos;
    printf("5 / 2 = %.11f\n", resultado);

    resultado = (double)cinco/dos;
    printf("5 / 2 = %.11f\n", resultado);

    getch();
}
```



```
H:\operadores.exe
5 / 2 = 2.0
5 / 2 = 2.5
```

En este programa se utilizan números en específico, para casos particulares, lo cual no es muy recomendable ya que se debe generalizar para cualquier caso.

## Comparaciones

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

## Operadores lógicos

### *Operador Operación*

<code>!</code>	No
<code>&amp;&amp;</code>	Y
<code>  </code>	O

## Operadores lógicos

```
#include <stdio.h>

int main() {
    int num1, num2, res;
    char c1, c2;

    num1 = 7;
    num2 = 15;
    c1 = 'h';
    c2 = 'H';

    printf("%i num1 es menor a num2 ? -> %d\n", num1 < num2);
    printf("%i c1 es igual a c2 ? -> %d\n", c1 == c2);
    printf("%i c1 es diferente a c2 ? -> %d\n", c1 != c2);

    res = num1 < num2 && c1 == 'h';
    printf("%i num1 < num2 Y c1 es igual a 'h' ? -> %d\n", res);

    res = c1 == 's' || c2 == 'H';
    printf("%i c1 es igual a 's' O c2 a 'H' ? -> %d\n", res);

    return 0;
}
```

```
#include <stdio.h>

int main() {
    int num1, num2, res;
    char c1, c2;

    num1 = 7;
    num2 = 15;
    c1 = 'h';
    c2 = 'H';

    printf("%i num1 es menor a num2 ? -> %d\n", num1 < num2);
    printf("%i c1 es igual a c2 ? -> %d\n", c1 == c2);
    printf("%i c1 es diferente a c2 ? -> %d\n", res);

    res = num1 > num2 && c1 == 'h';
    printf("%i num1 < num2 Y c1 es igual a 'h' ? -> %d\n", res);

    res = c1 == 's' || c2 == 'H';
    printf("%i c1 es igual a 's' 0 c2 a 'H'? -> %d\n", res);

    getch();
}
```

Estas usando esta variable sin valor

Este no es el resultado esperado

```
H:\operadoreslogicos.exe
num1 es menor a num2 ? -> 1
c1 es igual a c2 ? -> 0
c1 es diferente a c2 ? -> 2
num1 < num2 Y c1 es igual a 'h' ? -> 0
c1 es igual a 's' 0 c2 a 'H'? -> 1
```

Este programa imprime en pantalla las variables tal como se aclaran en el programa, sin embargo para el usuario que utiliza el programa puede ser confuso.

# Conclusión

Necesitamos resaltar la importancia de la escritura en un código para cualquier programa que se utilice, un error es la diferencia entre compilar y no compilar un programa, se debe ser muy cuidadoso al momento de escribir las variables, operadores, funciones y los elementos de entrada y salida.

C es un lenguaje importante para el desarrollo de sistemas operativos y aplicaciones, útil para introducirnos en el mundo de la programación ya que C es el lenguaje compatible para todos los compiladores, o su gran mayoría, por lo que este tipo de programas se pueden desarrollar en cualquier PC disponible.