



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: **ALEJANDRO PIMENTEL**

Asignatura: **FUNDAMENTOS DE PROGRAMACIÓN**

Grupo: **BLOQUE 135**

No de Práctica(s): **PRACTICA 7 “Fundamentos de Lenguaje C”**

Integrante(s): **ALITZEL ANAID GUTIÉRREZ RAMOS**

*No. de Equipo de
cómputo empleado:*

No. de Lista o Brigada: **9370**

Semestre: **1er SEMESTRE**

Fecha de entrega: **03-10-2019**

Observaciones:

CALIFICACIÓN: _____

INTRODUCCIÓN

Todos los programas necesitan, en algún momento, almacenar números o datos ingresado por el usuario. Estos datos son almacenados en variables, y en C++ como en otros lenguajes estas variables deben tener un tipo y es lo que se veremos en esta práctica.

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.

SECCIONES EQUIVALENTES

TIPOS DE VARIABLES

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, mas espacio en memoria ocupará.



1.1 bool

Por lo general utiliza **1** byte de memoria, valores: **true** o **false**.

1.2 char

Utiliza generalmente **1** byte de memoria, permite almacenar un carácter, valores; **256** caracteres.

1.3 unsigned short int

Utiliza generalmente **2** bytes de memoria, valores: de **0** a **65 535**

1.4 short int

Utiliza generalmente **2** bytes de memoria, valores: de **-32768** a **32767**.

1.5 unsigned long int

Utiliza generalmente **4** bytes de memoria, valores: de **0** a **4 294 967 295**.

1.6 long int

Utiliza generalmente **4** bytes de memoria, valores: de **-2 147 483 648** a **2 147 483 647**.

1.7 int (16 bits)

Utiliza generalmente **2** bytes de memoria, valores: de **-32 768** a **32 767**.

1.8 int (32 bits)

Utiliza generalmente **4** bytes de memoria, valores: de **-2 147 483 648** a **2 147 483 647**.

1.9 unsigned int (16 bits)

Utiliza generalmente 2 bytes de memoria, valores: de 0 a 65 535.

1.10 unsigned int (32 bits)

Utiliza generalmente 2 bytes de memoria, valores: de 0 a 4 294 967 295.

1.11 double

Utiliza generalmente 8 bytes de memoria, valores: de 2.2e-308 a 3.4e-38.

1.12 float

Utiliza generalmente 4 bytes de memoria, valores: de 1.2e-308 a 3.4e-38.

Atención!

El tamaño de las variables en memoria puede variara de un PC a otro.

DECLARACIÓN

Para declarar una variable, basta con indicar su tipo y su nombre. Existen ciertas convenciones en cuanto al nombre de las variables. Algunos prefieren separar las partes de un nombre con '_', otros prefieren escribir una mayúscula para separarlas. Ejemplo:

```
int recetaDelMes;
```

O

```
int receta_del_mes;
```

Lo importante es que utilices siempre la misma convención para tus programas.

ASIGNAR UN VALOR

Es posible asignar un valor a una variable al momento de declararla:

```
int recetaDelMes = 12301;
```

También es posible declarar varias variables en una misma línea, pero en este caso, todas las variables de la línea tendrán el mismo tipo.

```
int recetaDelMes = 12301, recetaDelAño = 45644545;
```

ERROR AL DEFINIR UN TIPO DE VARIABLE (ENTEROS CON SIGNO).

¿Qué pasa si el tipo que hemos elegido es muy pequeño? Si el número es mayor al máximo admitido por el tipo, entonces el valor de la variable será el valor mínimo admitido por este tipo.

```
unsigned short int numero = 65535;
cout << numero << endl;
numero++;
cout << numero << endl;
```

Si ejecutamos este código, la segunda línea no escribirá 65536, sino 0.
Esto es idéntico para todos los tipos.

ERROR AL DEFINIR UN TIPO DE VARIABLE (ENTEROS SIN SIGNO).

Para enteros sin signo, sucede lo mismo, una vez que el tipo alcanza su tamaño máximo, pasa a su valor mínimo.

```
short int numero = 32767;
cout << numero << endl;
numero++;
cout << numero << endl;
```

Si ejecutamos este código, la segunda línea no escribirá 32768, sino -32768.

<i>Tipo</i>	<i>Descripción</i>	<i>Bits</i>	<i>Rango</i>		
unsigned char	carácter sin signo	8	0	a	255
char	carácter	8	-128	a	127
short int	entero corto	16	-32768	a	32767
unsigned int	entero sin signo	32	0	a	4 294 967 295
int	entero	32	-2 147 483 648	a	2 147 483 647
unsigned long	entero largo sin signo	32	0	a	4 294 967 295
enum	enum	16	-2 147 483 648	a	2 147 483 647
long	entero largo	32	-2 147 483 648	a	2 147 483 647
float	real (con punto decimal)	32	3.4×10^{-38}	a	3.10×10^{38}
double	real doble	64	1.7×10^{-308}	a	1.7×10^{308}
long double	real doble largo	80	3.4×10^{-4932}	a	1.1×10^{4932}

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

<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

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

COMPARACIONES.

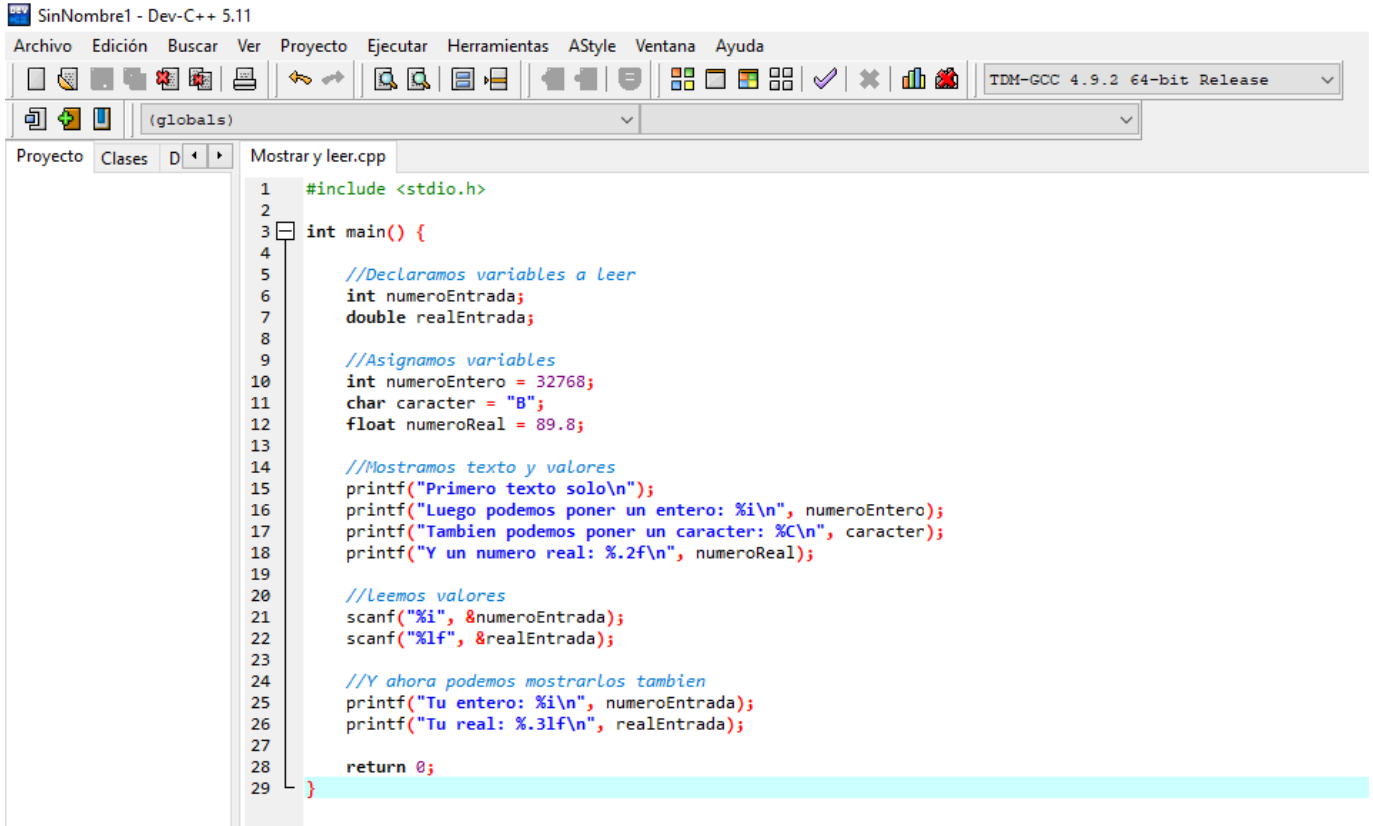
<i>Operador</i>	<i>Operación</i>	<i>Uso</i>	<i>Resultado</i>
==	Igual que	'h' == 'H'	Falso
!=	Diferente a	'a' != 'b'	Verdadero
<	Menor que	7 < 15	Verdadero
>	Mayor que	11 > 22	Falso
<=	Menor o igual	15 <= 22	Verdadero
>=	Mayor o igual	20 >= 35	Falso

OPERADORES LÓGICOS.

<i>Operador</i>	<i>Operación</i>
!	No
&&	Y
	O

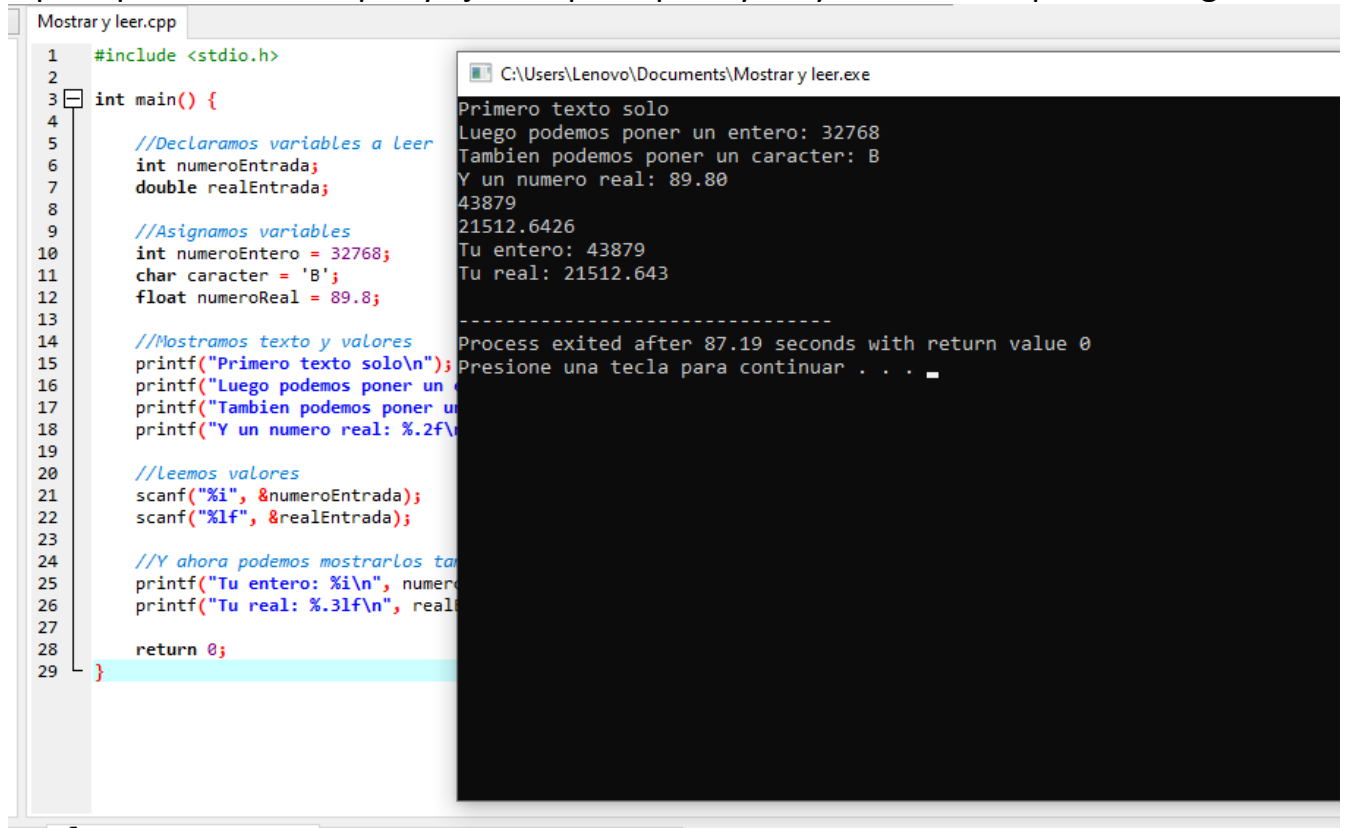
ACTIVIDADES DESARROLLADAS

1. Mostramos y leímos;



```
1 #include <stdio.h>
2
3 int main() {
4
5     //Declaramos variables a leer
6     int numeroEntrada;
7     double realEntrada;
8
9     //Asignamos variables
10    int numeroEntero = 32768;
11    char caracter = "B";
12    float numeroReal = 89.8;
13
14    //Mostramos texto y valores
15    printf("Primero texto solo\n");
16    printf("Luego podemos poner un entero: %i\n", numeroEntero);
17    printf("Tambien podemos poner un caracter: %C\n", caracter);
18    printf("Y un numero real: %.2f\n", numeroReal);
19
20    //leemos valores
21    scanf("%i", &numeroEntrada);
22    scanf("%lf", &realEntrada);
23
24    //Y ahora podemos mostrarlos tambien
25    printf("Tu entero: %i\n", numeroEntrada);
26    printf("Tu real: %.3lf\n", realEntrada);
27
28    return 0;
29 }
```

Después para mostrar copile y ejecute para que leyera y mostrara en pantalla negra:

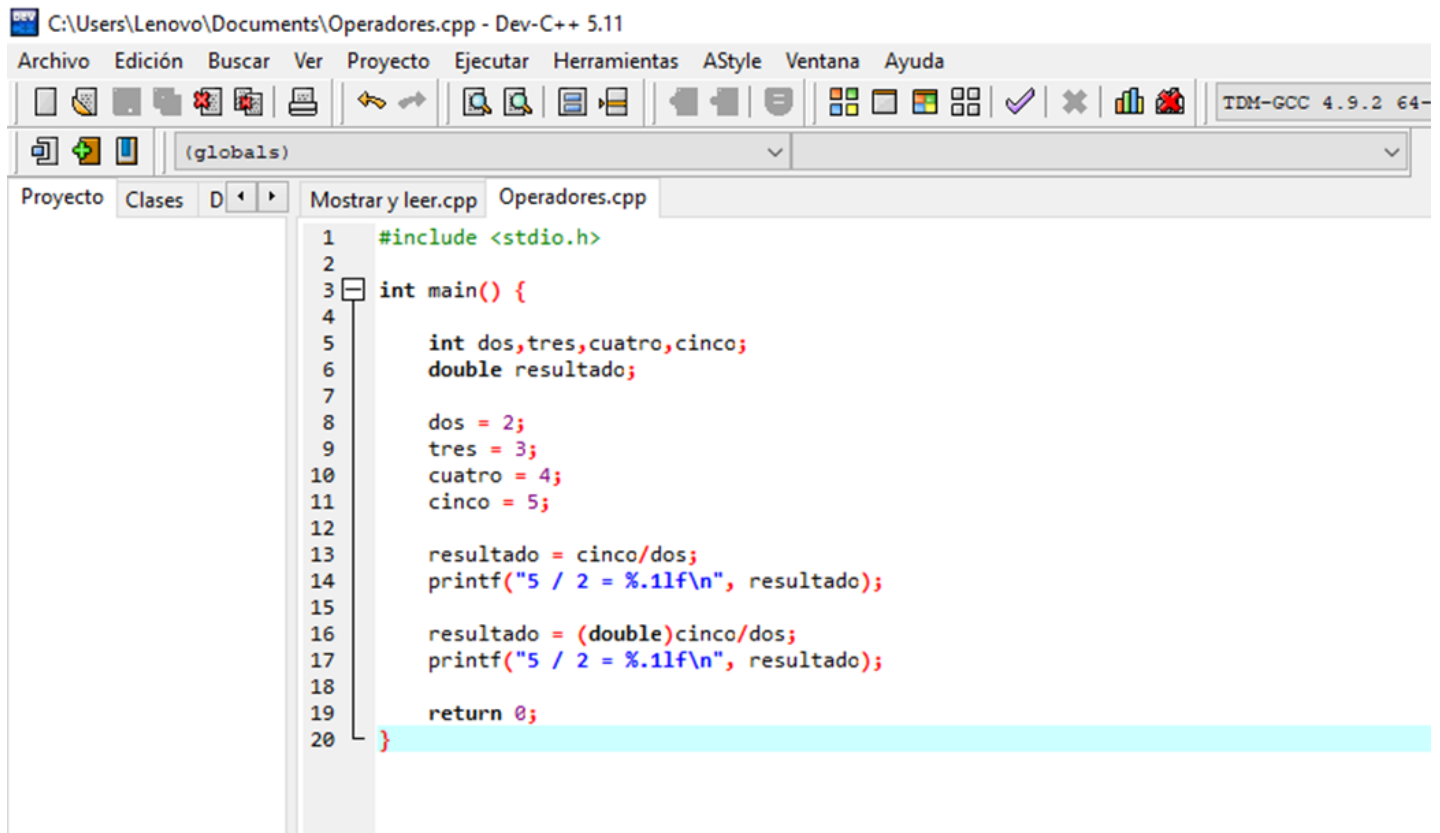


```
1 #include <stdio.h>
2
3 int main() {
4
5     //Declaramos variables a leer
6     int numeroEntrada;
7     double realEntrada;
8
9     //Asignamos variables
10    int numeroEntero = 32768;
11    char caracter = 'B';
12    float numeroReal = 89.8;
13
14    //Mostramos texto y valores
15    printf("Primero texto solo\n");
16    printf("Luego podemos poner un e
17    printf("Tambien podemos poner u
18    printf("Y un numero real: %.2f\
19
20    //leemos valores
21    scanf("%i", &numeroEntrada);
22    scanf("%lf", &realEntrada);
23
24    //Y ahora podemos mostrarlos ta
25    printf("Tu entero: %i\n", numero
26    printf("Tu real: %.3lf\n", real
27
28    return 0;
29 }
```

```
C:\Users\Lenovo\Documents\Mostrar y leer.exe
Primero texto solo
Luego podemos poner un entero: 32768
Tambien podemos poner un caracter: B
Y un numero real: 89.80
43879
21512.6426
Tu entero: 43879
Tu real: 21512.643
-----
Process exited after 87.19 seconds with return value 0
Presione una tecla para continuar . . .
```

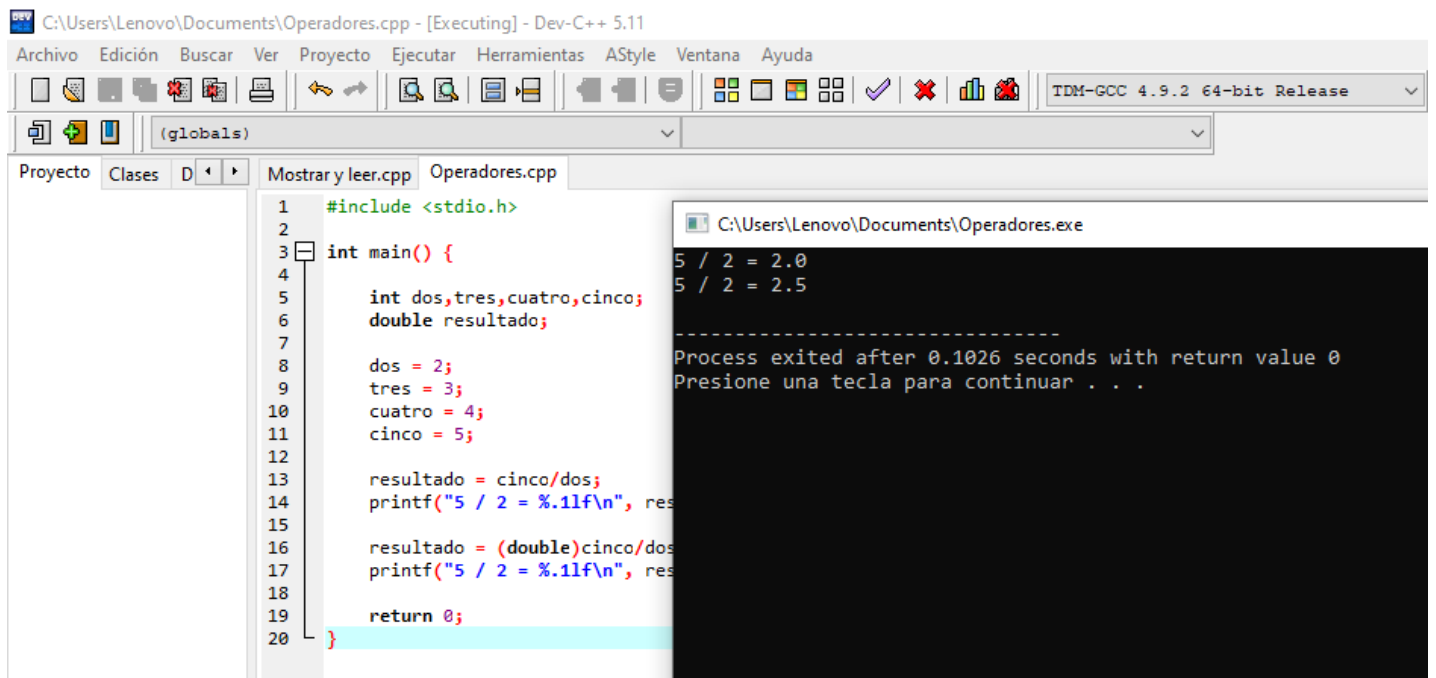
Y efectivamente solo muestra 3 decimales como lo ordenamos.

2. Operadores:



```
1  #include <stdio.h>
2
3  int main() {
4
5      int dos,tres,cuatro,cinco;
6      double resultado;
7
8      dos = 2;
9      tres = 3;
10     cuatro = 4;
11     cinco = 5;
12
13     resultado = cinco/dos;
14     printf("5 / 2 = %.11f\n", resultado);
15
16     resultado = (double)cinco/dos;
17     printf("5 / 2 = %.11f\n", resultado);
18
19     return 0;
20 }
```

La división de enteros dará otro entero por eso pusimos “double” así nos da un número real.

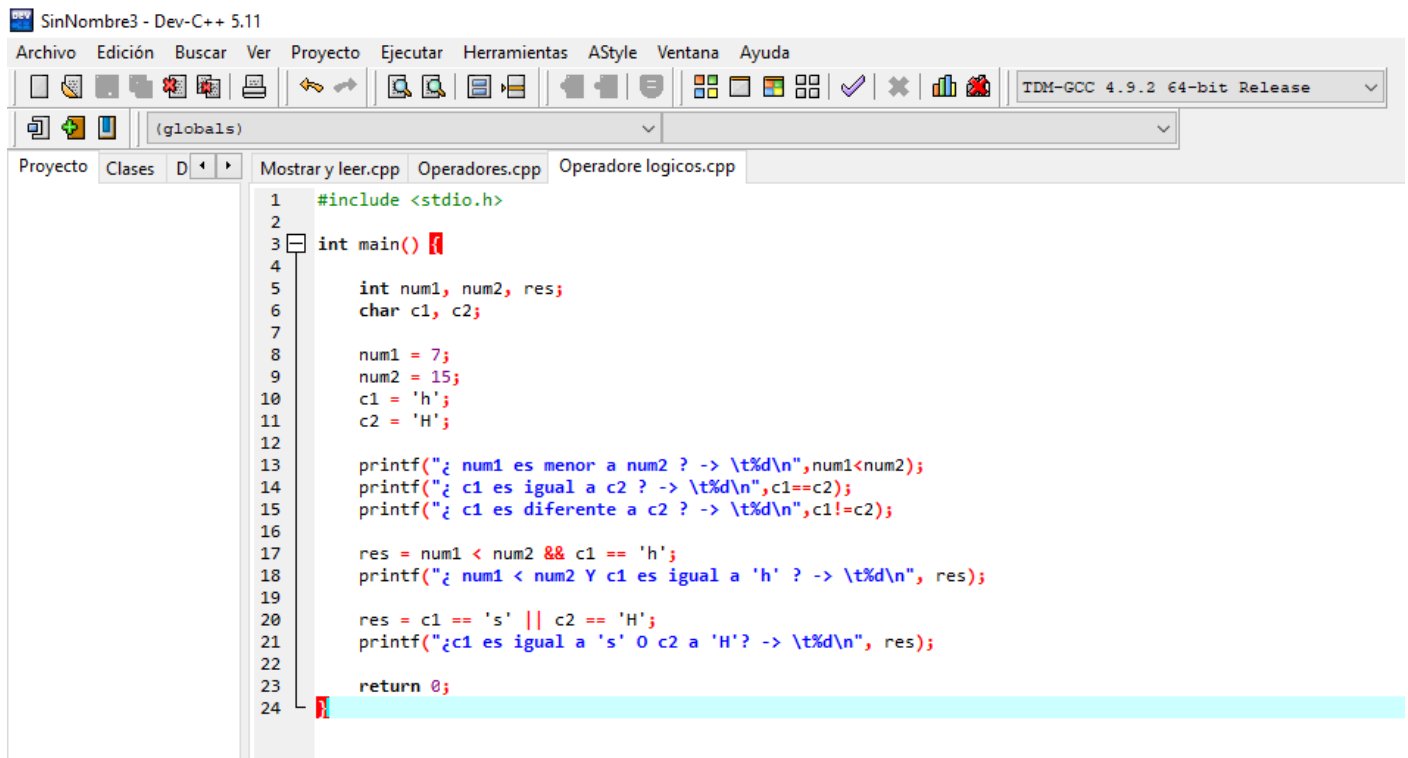


```
1  #include <stdio.h>
2
3  int main() {
4
5      int dos,tres,cuatro,cinco;
6      double resultado;
7
8      dos = 2;
9      tres = 3;
10     cuatro = 4;
11     cinco = 5;
12
13     resultado = cinco/dos;
14     printf("5 / 2 = %.11f\n", resultado);
15
16     resultado = (double)cinco/dos;
17     printf("5 / 2 = %.11f\n", resultado);
18
19     return 0;
20 }
```

```
C:\Users\Lenovo\Documents\Operadores.exe
5 / 2 = 2.0
5 / 2 = 2.5
-----
Process exited after 0.1026 seconds with return value 0
Presione una tecla para continuar . . .
```

Así se muestra el ejecutar y copiar.

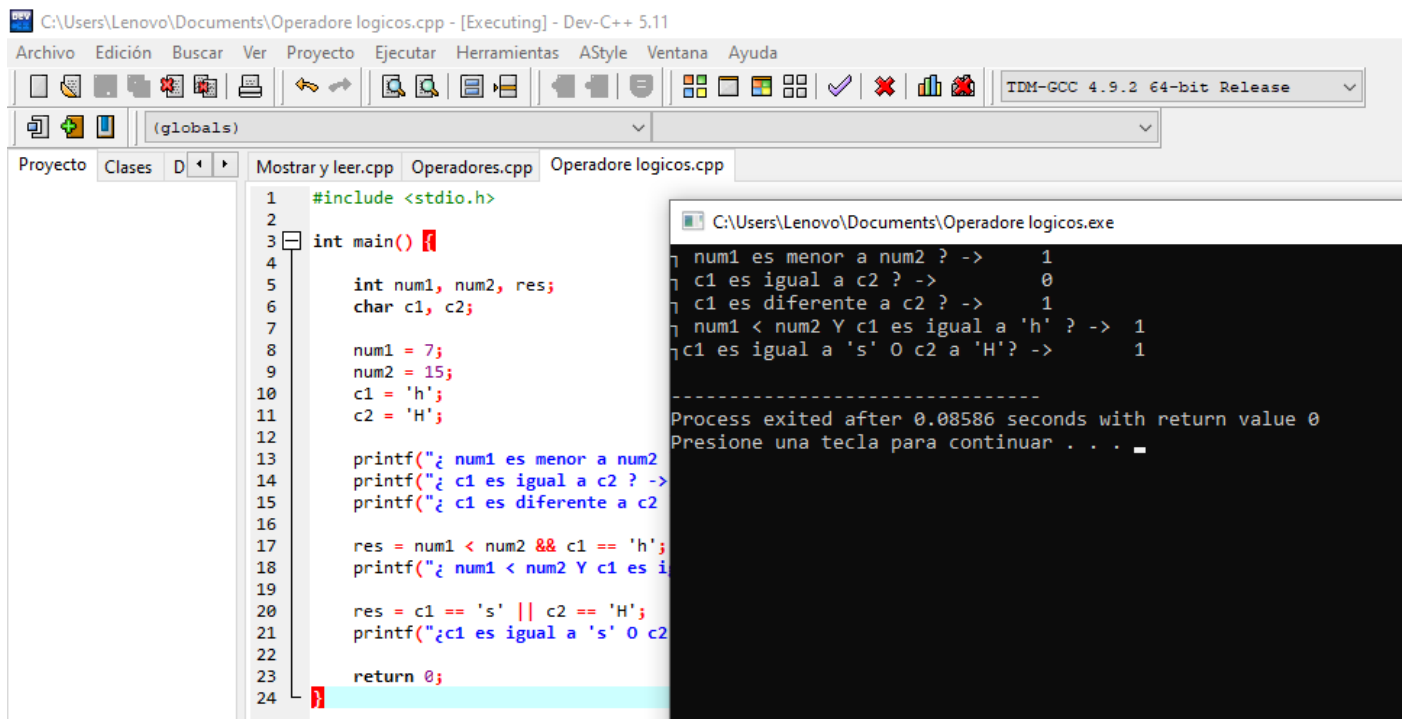
3. Operadores lógicos:



The screenshot shows the Dev-C++ 5.11 IDE with the file 'Operadores.cpp' open. The code defines a main function that tests various logical operators. It declares variables num1, num2, res, c1, and c2. It assigns values to num1 (7), num2 (15), c1 ('h'), and c2 ('H'). It then uses printf to display the results of several logical expressions: num1 < num2, c1 == c2, c1 != c2, num1 < num2 && c1 == 'h', num1 < num2 || c1 == 'h', c1 == 's' || c2 == 'H', and c1 == 's' || c2 == 'H'.

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int num1, num2, res;
6     char c1, c2;
7
8     num1 = 7;
9     num2 = 15;
10    c1 = 'h';
11    c2 = 'H';
12
13    printf("num1 es menor a num2 ? -> %d\n", num1 < num2);
14    printf("c1 es igual a c2 ? -> %d\n", c1 == c2);
15    printf("c1 es diferente a c2 ? -> %d\n", c1 != c2);
16
17    res = num1 < num2 && c1 == 'h';
18    printf("num1 < num2 Y c1 es igual a 'h' ? -> %d\n", res);
19
20    res = c1 == 's' || c2 == 'H';
21    printf("c1 es igual a 's' O c2 a 'H' ? -> %d\n", res);
22
23    return 0;
24 }
```

Solo pueden tener valores lógicos, 1 para Verdadero y 0 para Falso.



The screenshot shows the Dev-C++ 5.11 IDE with the file 'Operadore logicos.cpp' open. The code is the same as in the previous screenshot. A terminal window is open, showing the output of the program. The output displays the results of the logical expressions: num1 es menor a num2 ? -> 1, c1 es igual a c2 ? -> 0, c1 es diferente a c2 ? -> 1, num1 < num2 Y c1 es igual a 'h' ? -> 1, and c1 es igual a 's' O c2 a 'H' ? -> 1. The process exited after 0.08586 seconds with return value 0.

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int num1, num2, res;
6     char c1, c2;
7
8     num1 = 7;
9     num2 = 15;
10    c1 = 'h';
11    c2 = 'H';
12
13    printf("num1 es menor a num2
14    printf("c1 es igual a c2 ? ->
15    printf("c1 es diferente a c2
16
17    res = num1 < num2 && c1 == 'h';
18    printf("num1 < num2 Y c1 es i
19
20    res = c1 == 's' || c2 == 'H';
21    printf("c1 es igual a 's' O c2
22
23    return 0;
24 }
```

```
C:\Users\Lenovo\Documents\Operadore logicos.exe
num1 es menor a num2 ? -> 1
c1 es igual a c2 ? -> 0
c1 es diferente a c2 ? -> 1
num1 < num2 Y c1 es igual a 'h' ? -> 1
c1 es igual a 's' O c2 a 'H' ? -> 1
-----
Process exited after 0.08586 seconds with return value 0
Presione una tecla para continuar . . .
```


CONCLUSIÓN

En conclusión, una variable en C es un espacio que reservamos en memoria para poder guardar información, las variables se utilizan a menudo para guardar números caracteres, entre otros tipos de datos, no solo en C si no en cualquier lenguaje de programación. En C cada variable (espacio reservado en memoria) puede utilizarse para guardar información, y dicha información se representa por medio de un tipo de dato específico, que determina como la interpretará el lenguaje en sí.