



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Alejandro Esteban Pimentel

Asignatura: Fundamentos de programación

Grupo: 3

No de Práctica(s): Practica 7

Integrante(s):

Garcés Gallardo Julio César
Velazco Gómez Noé

Si son más de uno, deben
poner sus números de
cuenta como siempre

*No. de Equipo de
cómputo empleado:*

No. de Lista o Brigada: 14

Semestre: Primer semestre

Fecha de entrega: 03 de Octubre del 2019

Observaciones:

Deben de tener cuidado de utilizar
los caracteres correctos, la comilla simple
es la vertical, no la que ustedes están usando
es por eso que el programa no corre como debe

CALIFICACIÓN: 8

Tema: Fundamentos de lenguaje C.

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.

Introducción: Se dará a conocer los diferentes tipos de variables que pueden ingresar en el lenguaje c, tanto el nombre de la variable como su valor:

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

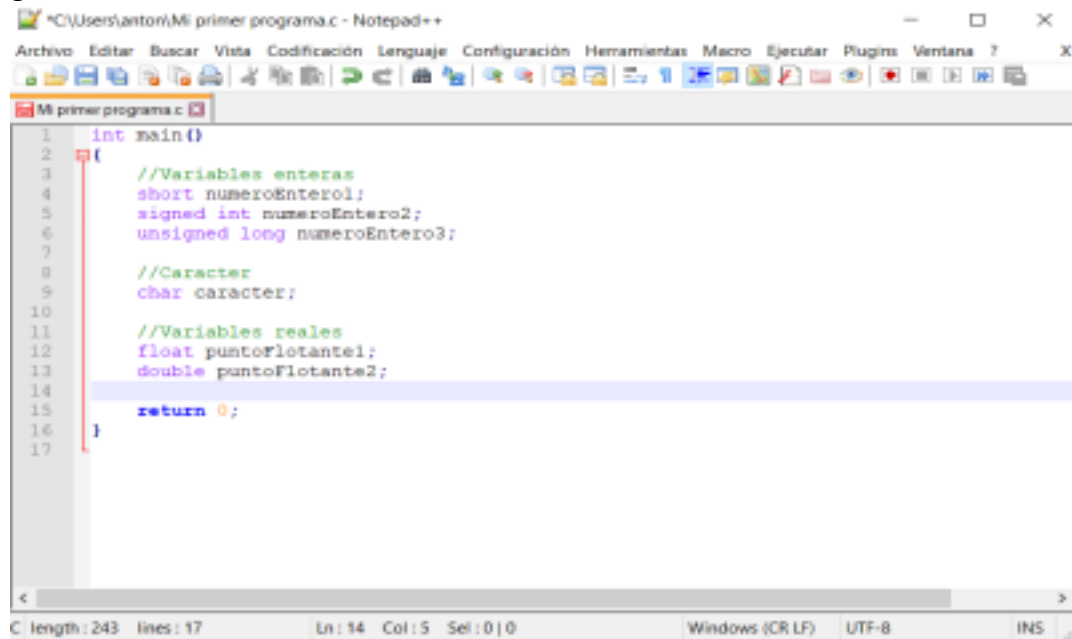
Asi como también el signo que se usa para leer o escanear la variable y algunos signos de operadores:

<i>Operador</i>	<i>Operación</i>	<i>Tipo de dato</i>	<i>Especificador de formato</i>
!	No	Entero	%d, %i, %ld, %li, %o, %x
&&	Y	Flotante	%f, %lf, %e, %g
	O	Carácter	%c, %d, %i, %o, %x
		Cadena de caracteres	%s

Desarrollo:

<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

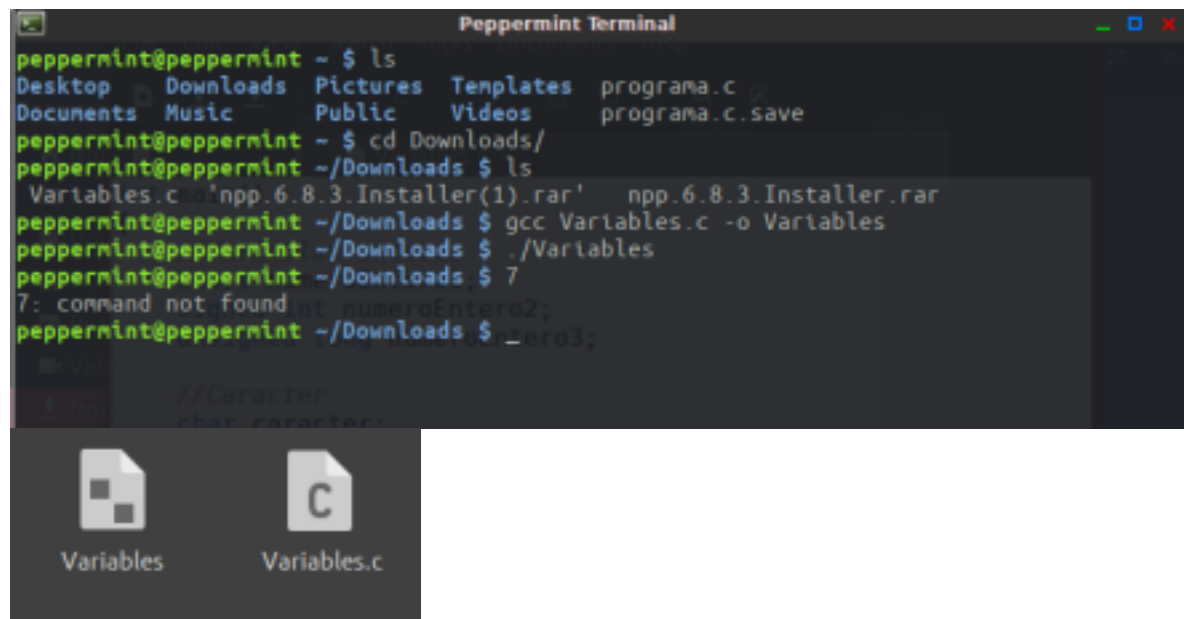
Para ver los diferentes tipos de variables en lenguaje c, creamos un programa que lea números enteros y reales y caracteres, como se muestra en esta imagen, este programa lo guardamos como Variables.c



```
1 int main()
2 {
3     //Variables enteras
4     short numeroEntero1;
5     signed int numeroEntero2;
6     unsigned long numeroEntero3;
7
8     //Caracter
9     char caracter;
10
11     //Variables reales
12     float puntoFlotante1;
13     double puntoFlotante2;
14
15     return 0;
16 }
17
```

C: length: 243 lines: 17 Ln: 14 Col: 5 Sel: 0|0 Windows (CR LF) UTF-8 INS

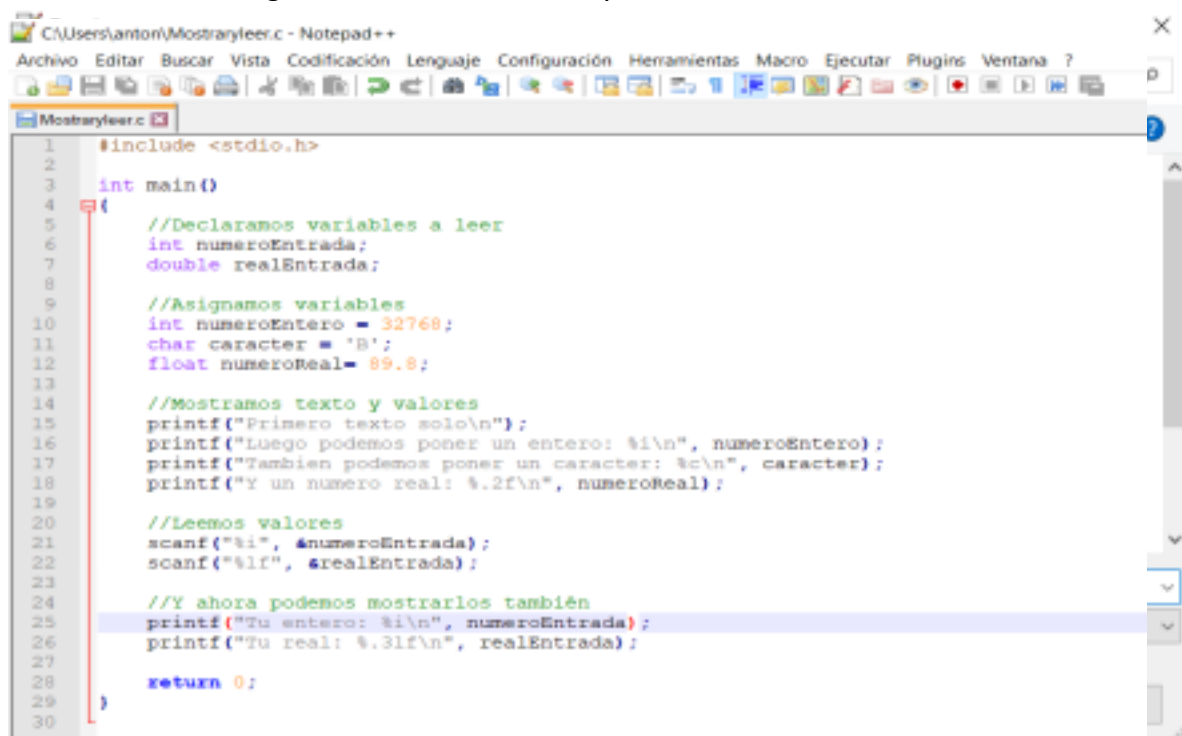
Cuando trate de compilar y correr este programa, si se compilo, pero no realizó ninguna acción, ya que por lo que veo, este programa solo recibe la información.



```
peppermint@peppermint ~ $ ls
Desktop Downloads Pictures Templates programa.c
Documents Music Public Videos programa.c.save
peppermint@peppermint ~ $ cd Downloads/
peppermint@peppermint ~/Downloads $ ls
Variables.c 'npp.6.8.3.Installer(1).rar' npp.6.8.3.Installer.rar
peppermint@peppermint ~/Downloads $ gcc Variables.c -o Variables
peppermint@peppermint ~/Downloads $ ./Variables
peppermint@peppermint ~/Downloads $ 7
7: command not found
peppermint@peppermint ~/Downloads $
```

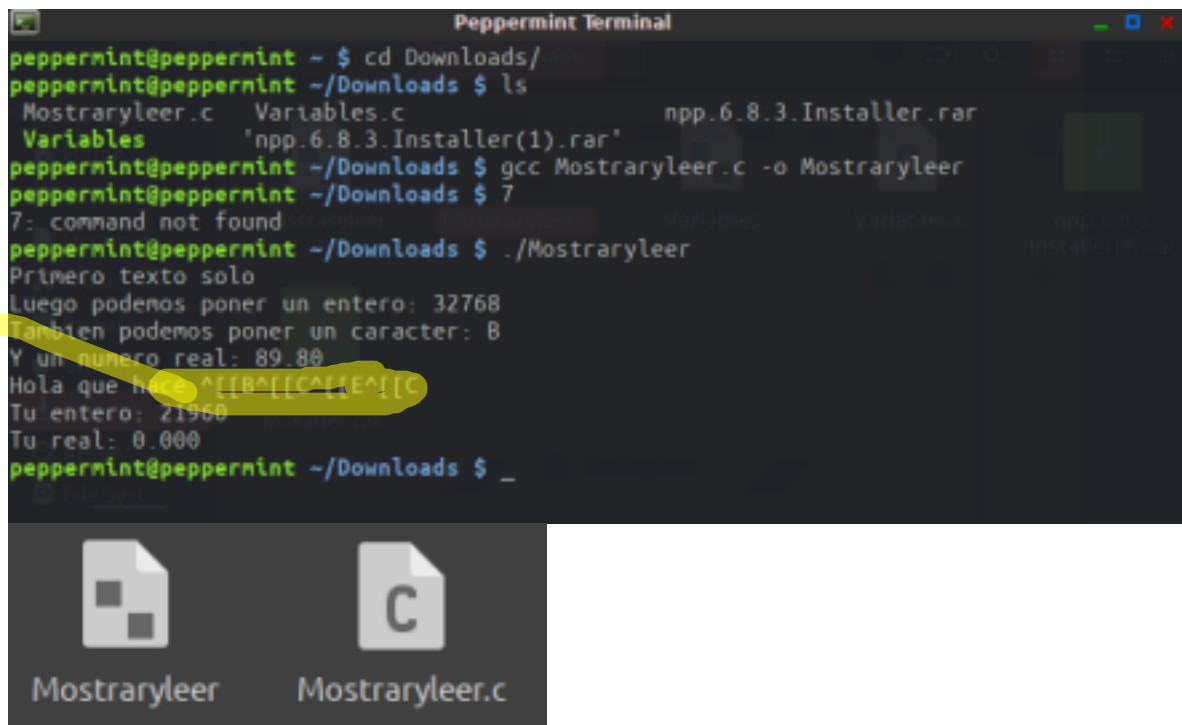
Variables Variables.c

Luego creamos otro programa donde incluimos lo visto anteriormente y aparte empezamos a usar especificadores de formato, esto justamente para especificar si usamos enteros, caracteres, etc. Y lo guardamos como Mostraryleer.c



```
1 #include <stdio.h>
2
3 int main()
4 {
5     //Declaramos variables a leer
6     int numeroEntero;
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", &numeroEntero);
22    scanf("%lf", &realEntrada);
23
24    //Y ahora podemos mostrarlos también
25    printf("Tu entero: %i\n", numeroEntero);
26    printf("Tu real: %.3lf\n", realEntrada);
27
28    return 0;
29 }
30
```

Despues lo compile y corré, y este fue diferente, ya que al correrlo me permitía ingresar un numero entero o real, texto o hasta caracteres, y a partir de lo que metía es lo que me mostraba:

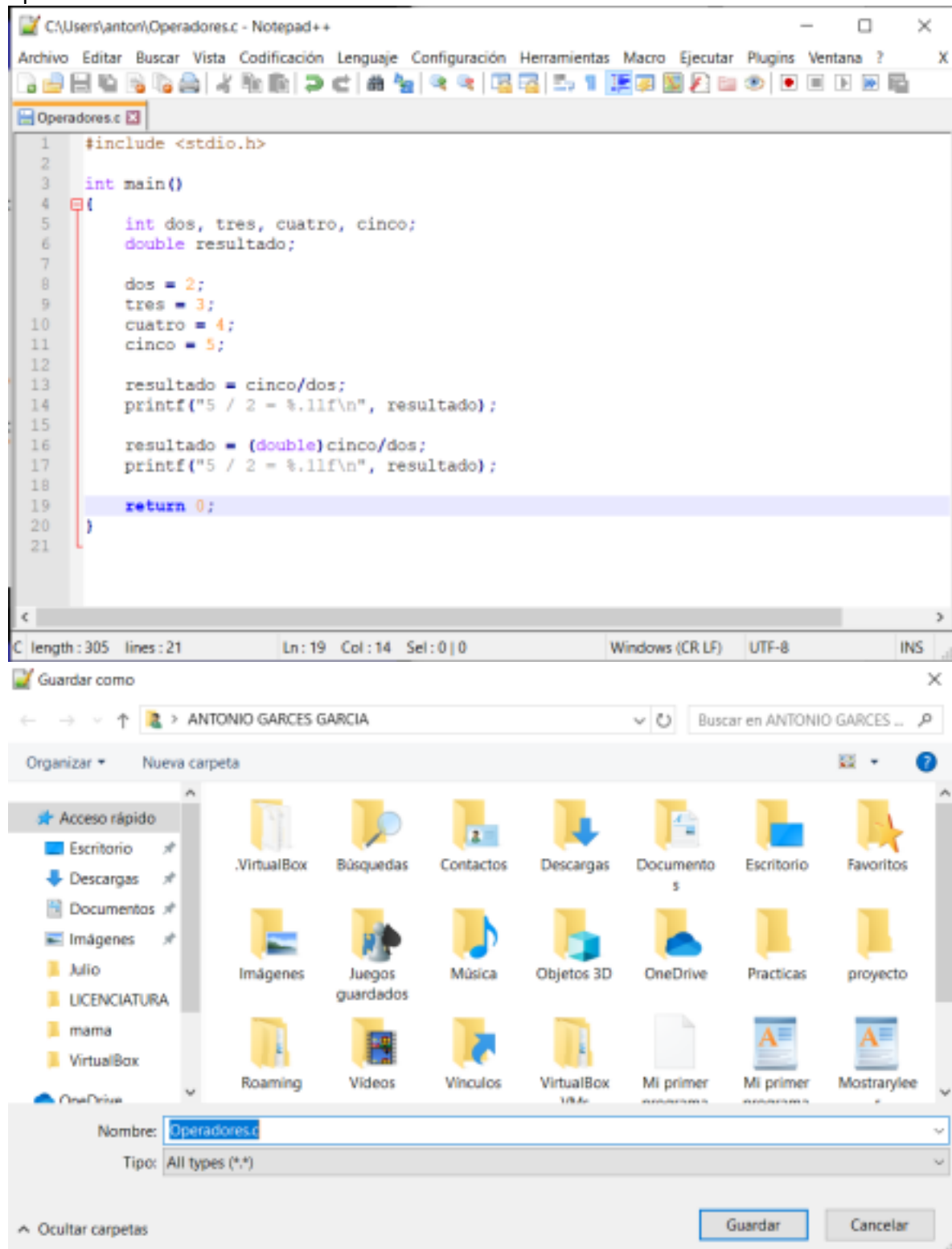


```
peppermint@peppermint ~ $ cd Downloads/
peppermint@peppermint ~/Downloads $ ls
Mostraryleer.c  Variables.c  npp.6.8.3.Installer.rar
Variables      'npp.6.8.3.Installer(1).rar'
peppermint@peppermint ~/Downloads $ gcc Mostraryleer.c -o Mostraryleer
peppermint@peppermint ~/Downloads $ 7
7: command not found
peppermint@peppermint ~/Downloads $ ./Mostraryleer
Primero texto solo
Luego podemos poner un entero: 32768
Tambien podemos poner un caracter: B
Y un numero real: 89.80
Hola que hace ^[[B^[[C^[[E^[[C
Tu entero: 21960
Tu real: 0.000
peppermint@peppermint ~/Downloads $ _
```

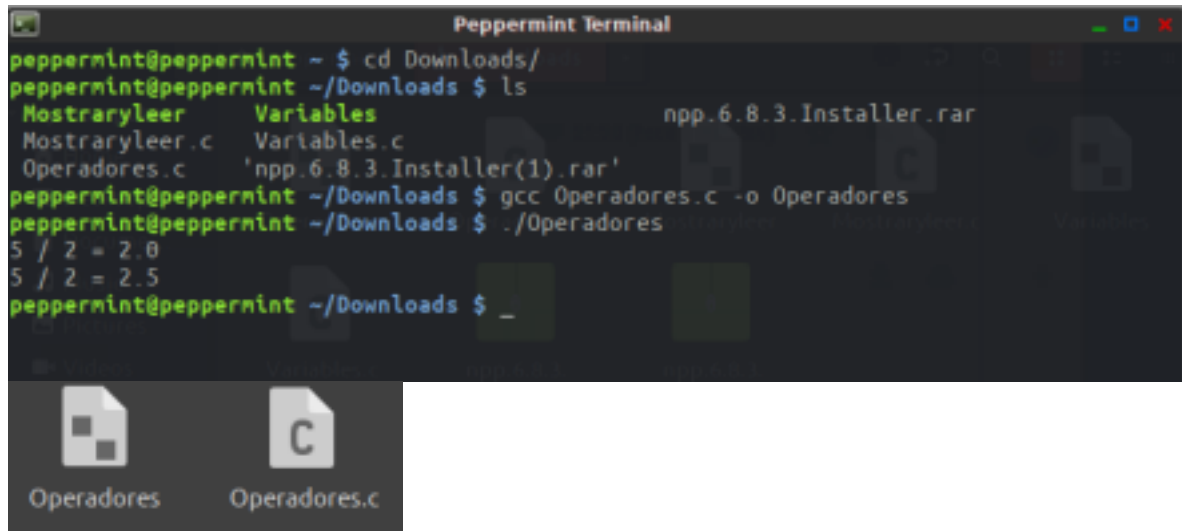
Mostraryleer Mostraryleer.c

El programa esperaba entradas numéricas, revisa el código

Luego creamos un programa donde empezamos a utilizar operadores como *, +, -, /, entre otros, tal como se muestra a continuación, y este programa lo guardamos con el nombre operadores.c



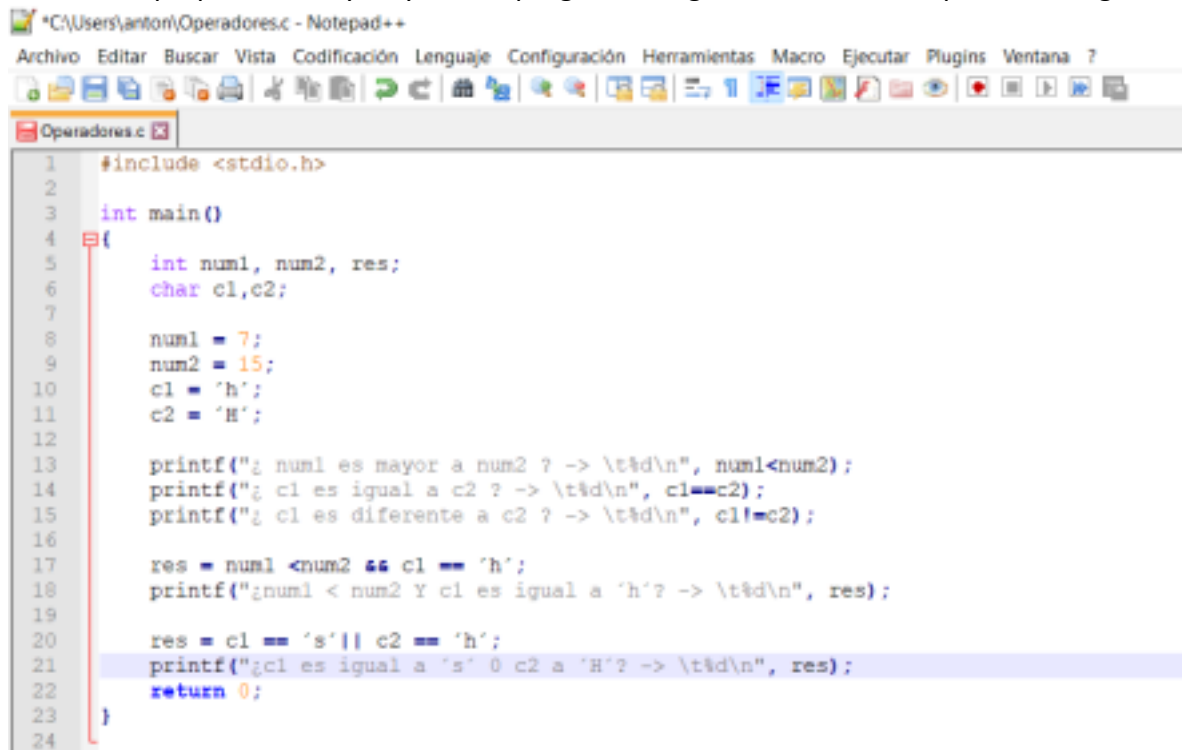
Cuando compilamos este programa, tuvo algo chistoso que no me había percatado antes, y es que el programa ya tiene definido ciertos números y ciertas acciones, entonces solo muestra el resultado de esta división.



```
peppermint@peppermint ~ $ cd Downloads/
peppermint@peppermint ~/Downloads $ ls
Mostraryleer      Variables      npp.6.8.3.Installer.rar
Mostraryleer.c    Variables.c
Operadores.c      'npp.6.8.3.Installer(1).rar'
peppermint@peppermint ~/Downloads $ gcc Operadores.c -o Operadores
peppermint@peppermint ~/Downloads $ ./Operadores
5 / 2 = 2.8
5 / 2 = 2.5
peppermint@peppermint ~/Downloads $ _
```

Operadores Operadores.c

Por ultimo creamos un programa que utilizara los operadores como !, && y ||, esto para indicar las preposiciones y, o y no. El programa lo guardamos como operadoreslogicos.c



```
*C:\Users\anton\Operadores.c - Notepad++
Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Plugins Ventana ?

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 mayor a num2 ? -> \t%d\n", num1 < num2);
14     printf("¿ c1 es igual a c2 ? -> \t%d\n", c1 == c2);
15     printf("¿ c1 es diferente a c2 ? -> \t%d\n", c1 != c2);
16
17     res = num1 < num2 && c1 == 'h';
18     printf("¿ num1 < num2 Y c1 es igual a 'h'? -> \t%d\n", res);
19
20     res = c1 == 's' || c2 == 'h';
21     printf("¿ c1 es igual a 's' O c2 a 'H'? -> \t%d\n", res);
22     return 0;
23 }
24
```

En este programa se trataba de encontrar equivalencias o quien era el mayor, pero tuvimos un problema, cuando tratamos de compilarlo nos salieron varios errores en la parte de los caracteres y no pudimos correr el programa, nos salió lo siguiente:

```
Peppermint Terminal
Operadoreslogicos.c:10:9: error: missing terminating ' character
    c1 = 'h';
           ^
Operadoreslogicos.c:10:8: error: 'h' undeclared (first use in this function)
    c1 = 'h';
           ^
Operadoreslogicos.c:10:8: note: each undeclared identifier is reported only once for each function it appears in
Operadoreslogicos.c:11:2: error: expected ';' before 'c2'
    c2 = 'H';
    ^
Operadoreslogicos.c:11:6: error: stray '\302' in program
    c2 = 'H';
           ^
Operadoreslogicos.c:11:7: error: stray '\264' in program
    c2 = 'H';
           ^
Operadoreslogicos.c:11:9: error: stray '\302' in program
    c2 = 'H';
           ^
Operadoreslogicos.c:11:10: error: stray '\264' in program
    c2 = 'H';
           ^
Operadoreslogicos.c:17:27: error: stray '\302' in program
    res = num1 < num2 && c1 == 'h';
                               ^

Peppermint Terminal
(sin ^sunto)
Operadoreslogicos.c:20:14: error: stray '\264' in program
    res = c1 == 's' || c2 == 'h';
               ^
Operadoreslogicos.c:20:16: error: stray '\302' in program
    res = c1 == 's' || c2 == 'h';
               ^
Operadoreslogicos.c:20:17: error: stray '\264' in program
    res = c1 == 's' || c2 == 'h';
               ^
Operadoreslogicos.c:20:15: error: 's' undeclared (first use in this function)
    res = c1 == 's' || c2 == 'h';
               ^
Operadoreslogicos.c:20:26: error: stray '\302' in program
    res = c1 == 's' || c2 == 'h';
                       ^
Operadoreslogicos.c:20:27: error: stray '\264' in program
    res = c1 == 's' || c2 == 'h';
                       ^
Operadoreslogicos.c:20:29: error: stray '\302' in program
    res = c1 == 's' || c2 == 'h';
                           ^
Operadoreslogicos.c:20:30: error: stray '\264' in program
    res = c1 == 's' || c2 == 'h';
                           ^
peppermint@peppermint ~/Downloads $
```

Tratamos de realiza cambios en los espacios, revisamos si algo estaba mal escrito, pero no pudimos encontrar cual es el error.

El error era la comilla, usaron la comilla incorrecta. La correcta es la que sale con la tecla de la derecha del 0(cero)

Conclusión: Concluimos que todo esto es muy parecido al pseudocódigo, pero tiene algunos detalles que lo hacen diferente, y justamente estos detalles lo vuelven un poco complicado, esto lo pudimos ver con el programa de operadores lógicos, incluso un espacio hace la diferencia en el lenguaje c, pero además de eso no es tan complicado, son los mismos signos y estructura, por lo cual lo consideramos útil, pero con un nivel de dificultad mas alto.