



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor: Alejandro Esteban Pimentel Alarcon*

*Asignatura: Fundamentos de programación*

*Grupo: 3*

*No de Práctica(s): 7*

*Integrante(s): Rivera Sosa Arlethe*

*No. de Equipo de cómputo empleado:*

*No. de Lista o Brigada: 317083033*

*Semestre: 2020-1*

*Fecha de entrega: 4/octubre/19*

*Observaciones:* Bien

**CALIFICACIÓN: 10**

## Lenguaje en C

El lenguaje C es un lenguaje estructurado, en el mismo sentido que lo son otros lenguajes de programación tales como el lenguaje Pascal, el Ada o el Modula-2, pero no es estructurado por bloques, o sea, no es posible declarar subrutinas (pequeños trozos de programa) dentro de otras subrutinas, a diferencia de como sucede con otros lenguajes estructurados tales como el Pascal. Además, el lenguaje C no es rígido en la comprobación de tipos de datos, permitiendo fácilmente la conversión entre diferentes tipos de datos y la asignación entre tipos de datos diferentes. Todo programa de C consta, básicamente, de un conjunto de funciones, y una función llamada main, la cual es la primera que se ejecuta al comenzar el programa, llamándose desde ella al resto de funciones que compongan nuestro programa.

Cada bloque puede contener:

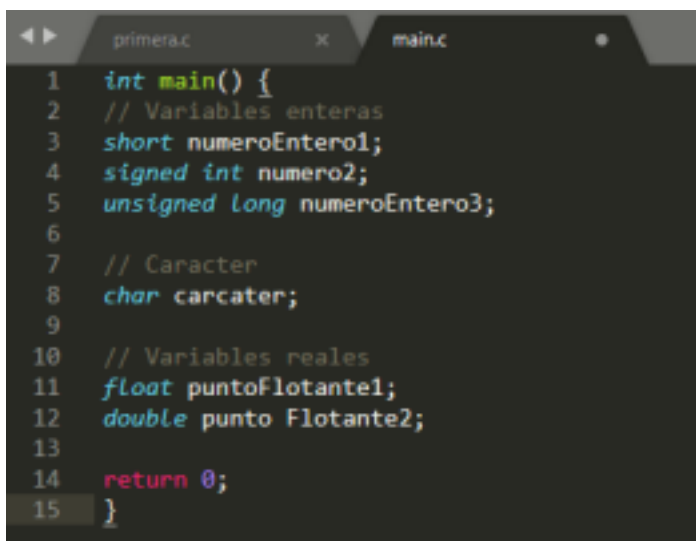
- Directivas del precompilador
- Declaraciones
- Una o más sentencias
- Comentarios
- Sistemas Operativos El Lenguaje de programación C

Cada sentencia debe estar terminada por:

- Cada bloque de sentencias se encierra entre llaves { . . }
- La función denominada main es la que primero se ejecuta
- Los comentarios pueden aparecer en cualquier lugar del código y se insertan entre /\* y \*/ así/\* esto es un comentario\*/  
o entre // y final de línea // esto es otro comentario

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.

Esta es una breve manera de cómo se ocupa C. Utilizando comandos para que haga alguna actividad en específico.



```
1  int main() {
2  // Variables enteras
3  short numeroEntero1;
4  signed int numero2;
5  unsigned long numeroEntero3;
6
7  // Caracter
8  char carcater;
9
10 // Variables reales
11 float puntoFlotante1;
12 double punto Flotante2;
13
14 return 0;
15 }
```

Declaramos variables, ponemos texto después de cada `//` y el programa no lo va a leer, simplemente es para saber que estamos haciendo nosotros.

Primero ponemos el tipo y después la variable:

`Double` realEntrada;

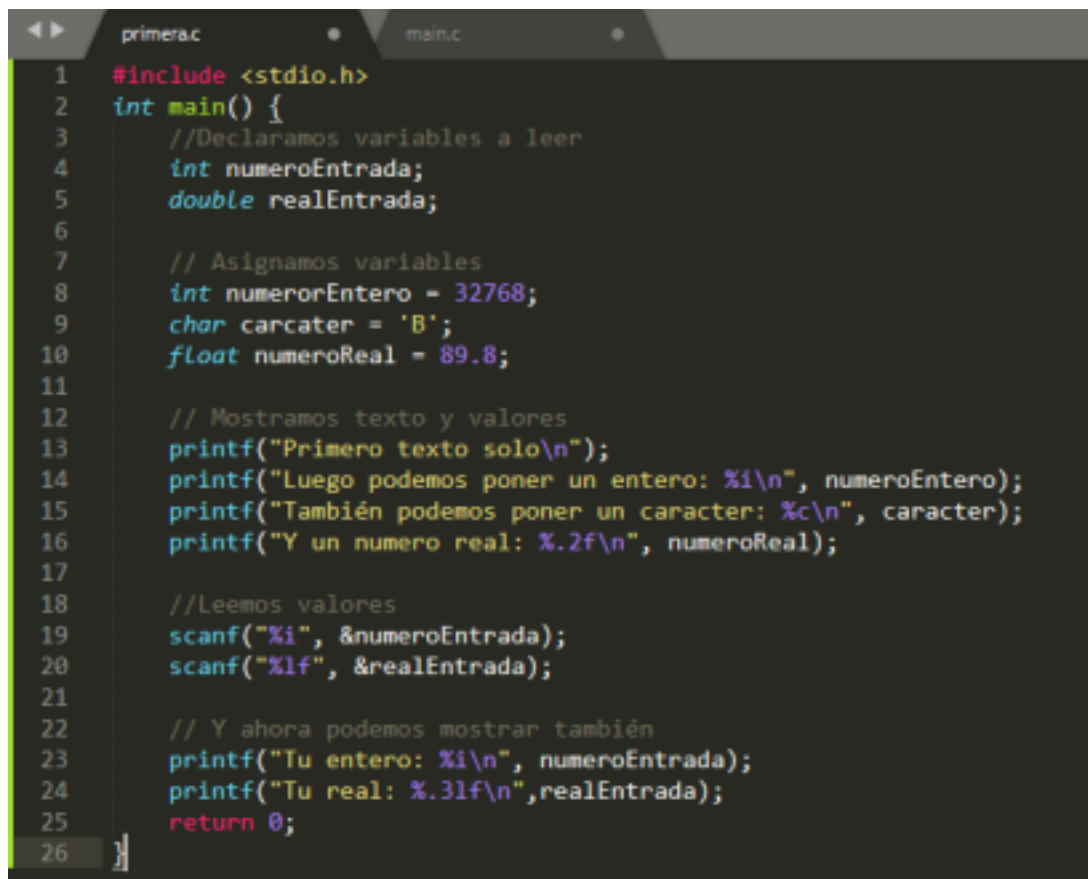
Si queremos mostrar el número tenemos que poner adentro del texto la variable. Es muy importante que después de cada línea poner un punto y coma, para que el programa lo pueda correr sin ninguna falla.

Para mostrar dígitos decimales, se coloca `%` seguido por `.número` de decimales deseados:

`%.2` f\n este comando solo muestra dos dígitos decimales.

Actividad1:

Muestra los números que recibe con algunos decimales extra.



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

Al momento de verificar nuestro programa en la terminal, si nos aparece algún mensaje de error, probablemente sea porque nuestro programa esta mal escrito o mal planteado.

De lo contrario hemos hecho un buen programa y tiene las características que deseamos.

```
nadia@LAPTOP-VR22P14N ~  
$ cd documents  
  
nadia@LAPTOP-VR22P14N ~/documents  
$ gcc primera.c -o primera  
  
nadia@LAPTOP-VR22P14N ~/documents  
$ ./primera  
Primero texto solo  
Luego podemos poner un entero: 32768  
También podemos poner un caracter: B  
Y un numero real: 89.80
```

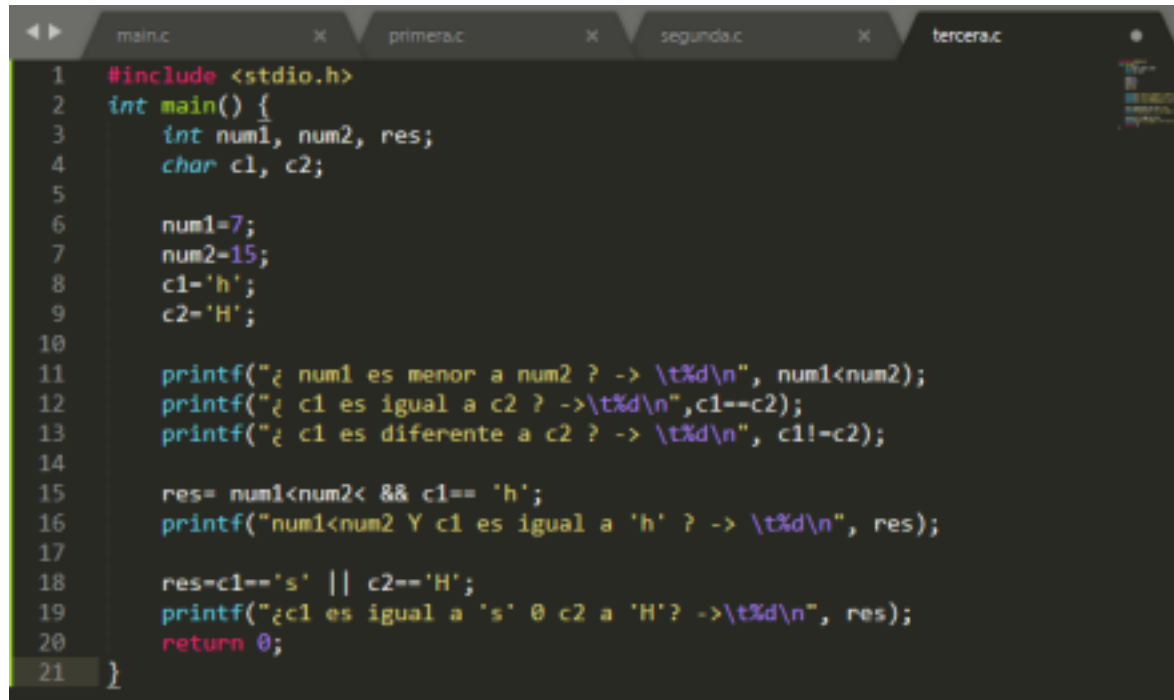
Actividad2: Aquí el programa se quedó esperando entrada, revisa el código.  
Muestra la respuesta con los valores obtenidos, de hacer una división.

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

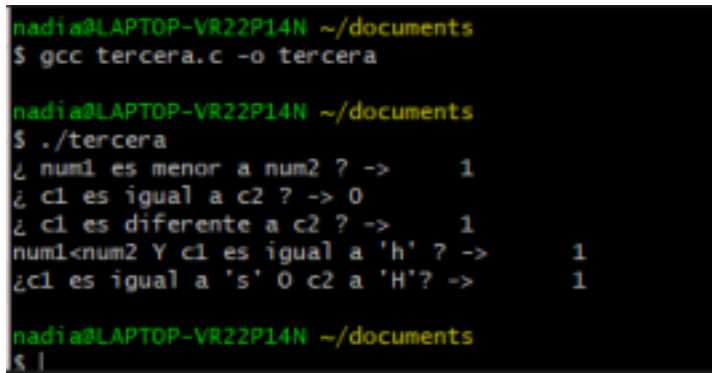
```
nadia@LAPTOP-VR22P14N ~/documents  
$ gcc segunda.C -o segunda  
  
nadia@LAPTOP-VR22P14N ~/documents  
$ ./segunda  
5/2=2.0  
5/2=2.5
```

### Actividad 3:

Muestra que número es mayor y cual es menor.



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



```
nadia@LAPTOP-VR22P14N ~/documents
$ gcc tercera.c -o tercera

nadia@LAPTOP-VR22P14N ~/documents
$ ./tercera
¿ 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' 0 c2 a 'H'? ->      1

nadia@LAPTOP-VR22P14N ~/documents
$ |
```

Conclusión: para poder compilar un programa, es muy necesario darse cuenta de que nuestro lenguaje este bien escrito y que pueda llegar a lo esperado. En esta practica nos dimos cuenta de que tan solo un error puede marcar la diferencia y que siempre hay que probar nuestros programas para verificar que cumplan con lo esperado.

Bibliografía:

- <https://www.dc.fi.udc.es/~so-grado/2019-20/Varios/CursoC.pdf>