



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): Nava Corona Nadia Erandeni

*No. de Equipo de cómputo
empleado:*

No. de Lista o Brigada: 6948

Semestre: 2020-1

Fecha de entrega: Miércoles 02 de octubre

Observaciones: Bien, pero tienes una omisión, recuerda que lo ideal es que entiendas el código, no solo copiarlo.

CALIFICACIÓN: 10

Fundamentos de lenguaje en C

Introducción.

C es un lenguaje de programación de propósito general que ofrece economía sintáctica, control de flujo y estructuras sencillas y un buen conjunto de operadores. No es un lenguaje de muy alto nivel y más bien un lenguaje pequeño, sencillo y no está especializado en ningún tipo de aplicación. Esto lo hace un lenguaje potente, con un campo de aplicación ilimitado y, sobre todo, se aprende rápidamente.

Objetivo.

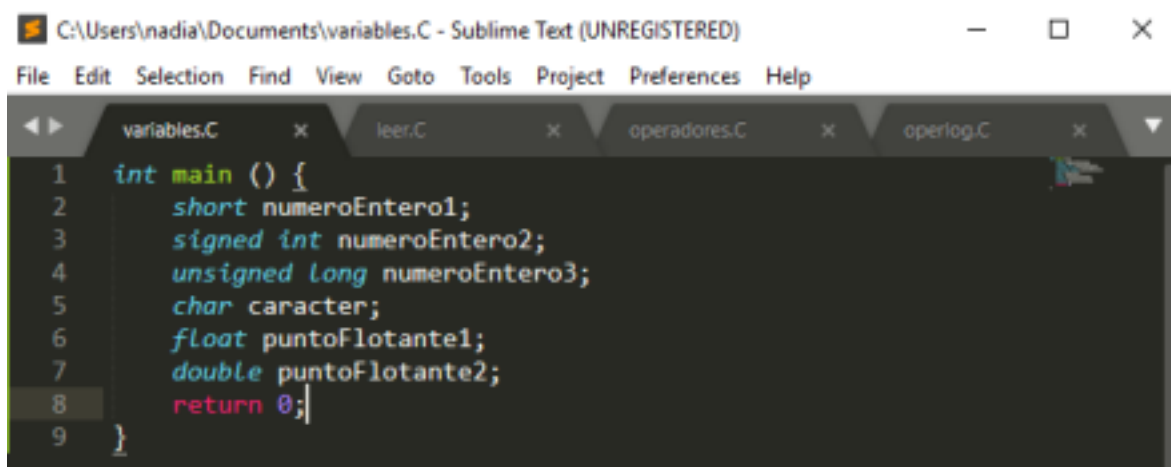
Elaborar programas en lenguaje C utilizando las instrucciones de control de tipo secuencia, para realizar la declaración de variables de diferentes tipos de datos, así como efectuar llamadas a funciones externas de entrada y salida para asignar y mostrar valores de variables y expresiones.

Tipos de variables

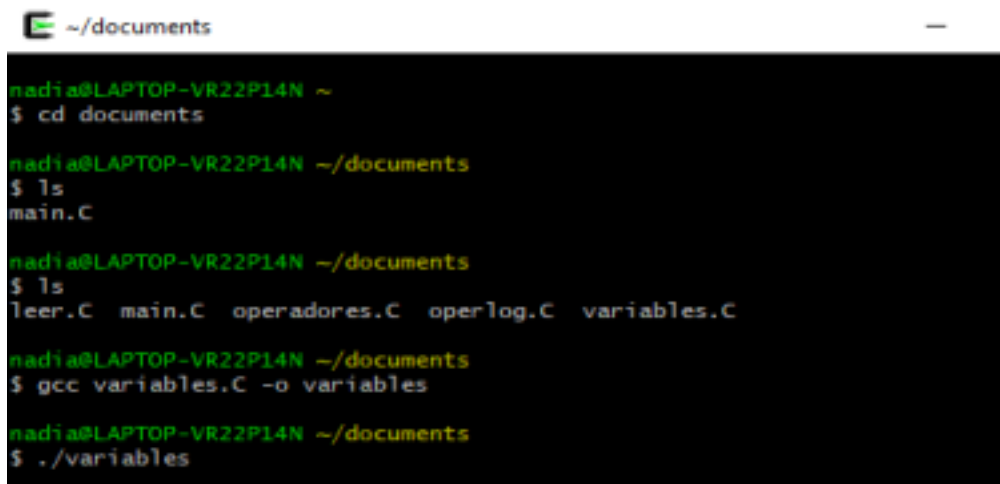
Para declarar una variable, basta con indicar su tipo y su nombre. **Short** utiliza generalmente 2 bytes de memoria, valores: de -32768 a 32767, **signed int** utiliza generalmente 2 bytes de memoria, valores: de 0 a 65 535. **Unsigned long** utiliza generalmente 2 bytes de memoria, valores: de 0 a 65 535. **Char** utiliza generalmente 1 byte de memoria, permite almacenar un carácter, valores; 256 caracteres. **Float** utiliza generalmente 4 bytes de memoria, valores: de 1.2e-308 a 3.4e-38. **Double** utiliza generalmente 4 bytes de memoria, valores: de 1.2e-308 a 3.4e-38.

Esto sirve para ser más específicos con los tipos de datos que vamos a meter.

En este programa no utilizamos print (mostrar) por lo que al momento de correrlo no ocurre nada.



```
C:\Users\nadia\Documents\variables.C - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
variables.C x leer.C x operadores.C x operlog.C x
1 int main () {
2     short numeroEntero1;
3     signed int numeroEntero2;
4     unsigned long numeroEntero3;
5     char caracter;
6     float puntoFlotante1;
7     double puntoFlotante2;
8     return 0;
9 }
```



```
~/documents
nadia@LAPTOP-VR22P14N ~
$ cd documents
nadia@LAPTOP-VR22P14N ~/documents
$ ls
main.C
nadia@LAPTOP-VR22P14N ~/documents
$ ls
leer.C main.C operadores.C operlog.C variables.C
nadia@LAPTOP-VR22P14N ~/documents
$ gcc variables.C -o variables
nadia@LAPTOP-VR22P14N ~/documents
$ ./variables
```

Mostrar y leer.

Al momento de hacer tu programa debes especificar las variables que vas a utilizar, por ende, al momento de querer mostrar y leer debes usar las **especificaciones de formato**. En este ejemplo, primero se declararon las variables y se les asignó un valor y letra. En la parte de `printf` donde queríamos que se mostrara el **número entero** ya declarado usamos `%i`, para poner el **carácter** usamos `%c` y para imprimir el **flotante** usamos `%2.f` (el dos señala los números que se mostrarán después del punto). Así que al momento de correr el programa se mostrará el texto junto con los valores o caracteres asignados.

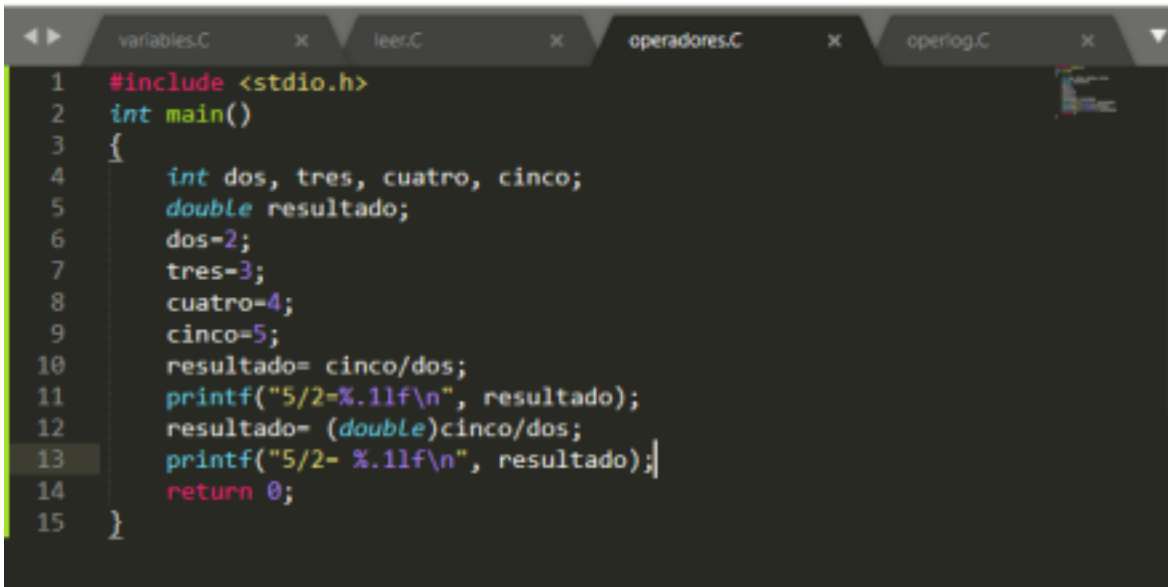
```
variables.C x leer.C x operadores.C x operlog.C x
1  #include <stdio.h>
2  int main()
3  {
4      int numeroEntrada;
5      double realEntrada;
6      int numeroEntero = 32768;
7      char caracter = 'B';
8      float numeroReal = 89.8;
9      printf("Primero texto solo\n");
10     printf("Luego podemos poner un entero: %i\n", numeroEntero);
11     printf("También podemos poner un caracter: %c\n", caracter);
12     printf("Y un numero real: %.2f\n", numeroReal);
13     scanf ("%i", &numeroEntrada);
14     scanf ("%lf", &realEntrada);
15     printf("Tu entero: %i\n", numeroEntrada);
16     printf("Tu real: %.3lf\n", realEntrada );
17     return 0;
18 }
```

```
~/documents
nadia@LAPTOP-VR22P14N ~
$ cd documents
nadia@LAPTOP-VR22P14N ~/documents
$ ls
leer.C main.C operadores.C operlog.C variables.C variables.exe
nadia@LAPTOP-VR22P14N ~/documents
$ gcc leer.C -o leer
nadia@LAPTOP-VR22P14N ~/documents
$ ./leer
Primero texto solo
Luego podemos poner un entero: 32768
También podemos poner un caracter: B
Y un numero real: 89.80
```

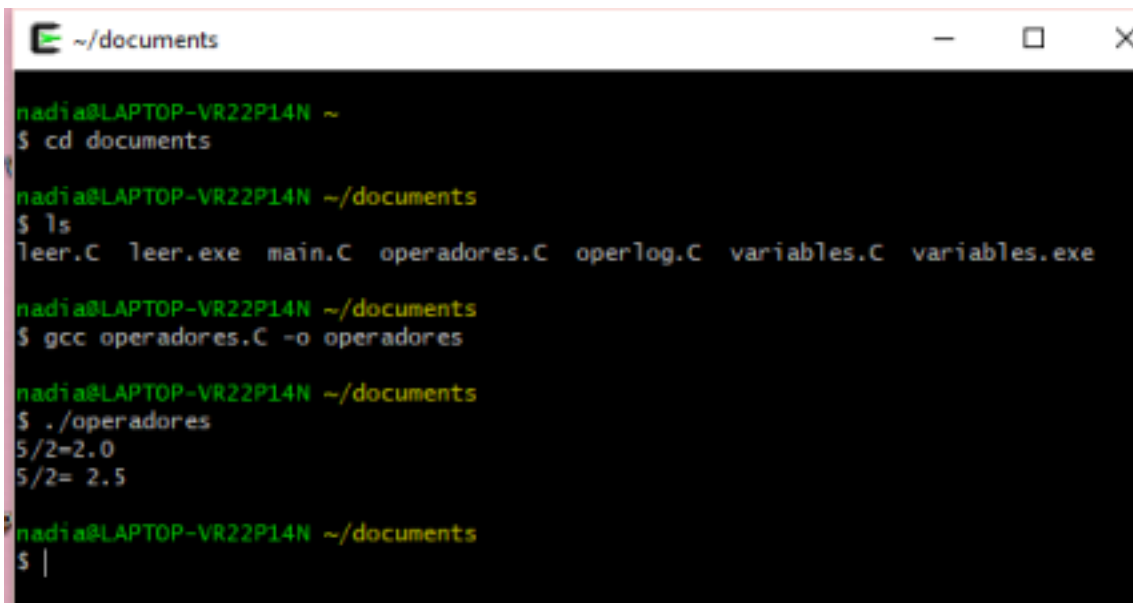
Aquí el programa se quedó esperando números para leer, revisa el código.

Operadores

Los operadores sirven para que se realice una operación dentro del programa, como lo puede ser una división, resta, suma, multiplicación y módulo. En el ejemplo siguiente, usamos división representada con `/`. Aunque es la misma división, el resultado se muestra diferente debido a que la segunda tiene la especificación *double*.



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



```
~/documents
nadia@LAPTOP-VR22P14N ~
$ cd documents
nadia@LAPTOP-VR22P14N ~/documents
$ ls
leer.C leer.exe main.C operadores.C operlog.C variables.C variables.exe
nadia@LAPTOP-VR22P14N ~/documents
$ gcc operadores.C -o operadores
nadia@LAPTOP-VR22P14N ~/documents
$ ./operadores
5/2=2.0
5/2= 2.5
nadia@LAPTOP-VR22P14N ~/documents
$ |
```

Operadores lógicos

Los operadores lógicos son los que ya conocíamos como AND, NOT y OR. En lenguaje en C estos se representan de la siguiente manera: AND (`&&`), NOT (`!`) y OR (`||`). En este ejemplo usamos solamente el operador lógico `&&` para mostrar una condición donde: `num1 < num2 && c1=='h'` (num1 es menor que num2 y c1 es igual a h). Además utilizamos también operadores de comparación, como lo es `==` (igual), `!=` (diferente de), `<` menor que, `>` mayor que, etc. Al momento de correrlo, las preguntas fueron contestadas con números, esto indica si es verdadero o falso en lenguaje de programación.

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

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

nadia@LAPTOP-VR22P14N ~/documents
$ ls
leer.C      main.C      operadores.exe  operadoreslog.exe  variables.C
leer.exe    operadores.C  operadoreslog.C  operlog.C          variables.exe

nadia@LAPTOP-VR22P14N ~/documents
$ gcc operadoreslog.C -o operadoreslog

nadia@LAPTOP-VR22P14N ~/documents
$ ./operadoreslog
¿ 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
```

Conclusiones.

Los tipos variables son espacios reservados en la memoria que, como su nombre indica, pueden cambiar de contenido a lo largo de la ejecución de un programa.

Al momento de leer y mostrar tu programa, es importante especificar que variable quieres que se imprima (con los especificadores de variable), ya que si no especificas la variable se imprimirá solo el texto puesto entre comillas.

Para realizar operaciones dentro de tu programa, es importante utilizar los operadores adecuados, ya que el lenguaje de programación es distinto al que usamos normalmente para realizar una operación a mano. Y por lo último los operadores lógicos que sirven para combinar condiciones, ya sea que se cumplan dos condiciones (&&), o que se deba de cumplir alguna de ellas (||).

Es importante utilizar adecuadamente el lenguaje en C, ya que esto nos permite que el programa se realice correctamente, aunque al momento de correrlo en la terminal nos muestre que error tenemos, es importante estar atentos, ya que puede que en realidad en el lenguaje no tengamos algún error pero en una operación si.