|  |
| --- |
| Laboratorio de Computación  Salas A y B |
|  |

|  |  |
| --- | --- |
| Profesor: | García Morales Karina |
| Asignatura: | Fundamentos de Programación |
| Grupo: | 1121 |
| No de Práctica(s): | 07 |
| Integrante(s): | Valle Olivas Guillermo |
|  |  |
|  |  |
|  |  |
| No. de Equipo de cómputo empleado: |  |
| Semestre: | 2019-1 |
| Fecha de entrega: |  |
| Observaciones: |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**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.

**Actividades:** 

1.- Crear un programa en lenguaje C que tenga definidas variables de varios tipos, se les asigne valores adecuados (por lectura o asignación directa) y muestre su valor en la salida estándar. 

2.- En un programa en C, asignar valores a variables utilizando expresiones aritméticas; algunas con uso de cambio de tipo (cast) 

3.- Elaborar expresiones relacionales/lógicas en un programa en C y mostrar el resultado de su evaluación.

**Introducción**

Una vez que un problema dado ha sido analizado (se identifican los datos de entrada y la salida deseada), que se ha diseñado un algoritmo que lo resuelva de manera eficiente (procesamiento de datos), y que se ha representado el algoritmo de manera gráfica o escrita (diagrama de flujo o pseudocódigo) se puede proceder a la etapa de codificación.

La codificación se puede realizar en cualquier lenguaje de programación estructurada, como lo son Pascal, Python, Fortran o PHP. En este curso se aprenderá el uso del lenguaje de programación C.

Dentro del ciclo de vida del software, la implementación de un algoritmo se encuentra dentro de la etapa de codificación del problema. Esta etapa va muy unida a la etapa de pruebas

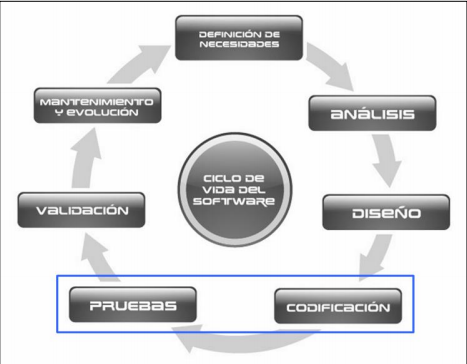


Figura 1: Ciclo de vida del software, resaltando las etapas de códificación y pruebas, las cuales se cubrirán en esta práctica.

**Lenguaje de programación C**

El proceso de desarrollo del lenguaje C se origina con la creación de un lenguaje llamado BCPL, que fue desarrollado por Martin Richards.

BCPL tuvo influencia en un lenguaje llamado B, el cual se usó en 1970 y fue inventado por Ken Thompson, esto permitió el desarrollo de C en 1971, el cual lo inventó e implementó Dennis Ritchie.

C es un lenguaje de propósito general basado en el paradigma estructurado. El teorema del programa estructurado, demostrado por Böhm-Jacopini, dicta que todo programa puede desarrollarse utilizando únicamente 3 instrucciones de control: 

Secuencia 

Selección 

Iteración

Por otro lado, C es un lenguaje compilado, es decir, existe un programa (llamado compilador) que, a partir de un código en lenguaje C, genera un código objeto (ejecutable).

Para crear un programa en C se siguen tres etapas principales: edición, compilación y ejecución. 

1. Edición: Se escribe el código fuente en lenguaje C desde algún editor de textos. 

2.- Compilación: A partir del código fuente (lenguaje C) se genera el archivo en lenguaje máquina (se crea el programa objeto o ejecutable). 

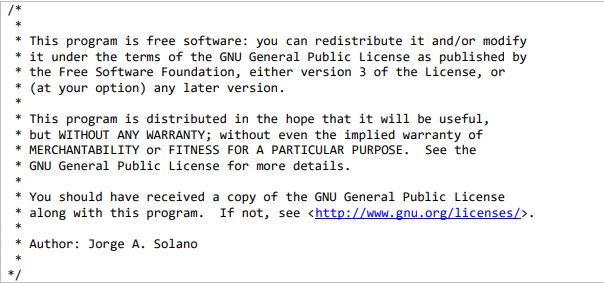
3.- Ejecución: El archivo en lenguaje máquina se puede ejecutar en la arquitectura correspondiente.

Un programa en C consiste en una o más funciones, de las cuales una de ellas debe llamarse main( ) y es la principal.

Al momento de ejecutar un programa objeto (código binario), se ejecutarán únicamente las instrucciones que estén definidas dentro de la función principal. La función principal puede contener sentencias, estructuras de control y comentarios. Dentro de las sentencias se encuentran la declaración y/o asignación de variables, la realización de operaciones básicas, y las llamadas a funciones.

**Licencia GPL de GNU**

El software presente en esta guía práctica es libre bajo la licencia GPL de GNU, es decir, se puede modificar y distribuir mientras se mantenga la licencia GPL.

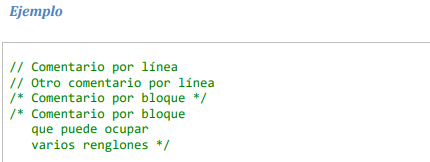


**Comentarios**

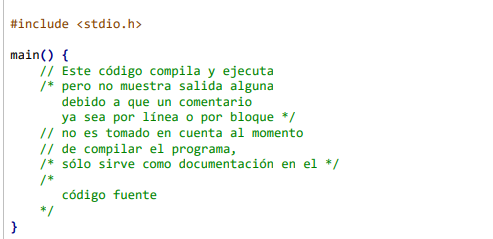
Es una buena práctica en cualquier lenguaje de programación realizar comentarios para documentar el programa. En C existen dos tipos de comentarios: el comentario por línea y el comentario por bloque.

El comentario por línea inicia cuando se insertan los símbolos '//' y termina con el salto de línea (hasta donde termine el renglón).

El comentario por bloque inicia cuando se insertan los símbolos '/\*' y termina cuando se encuentran los símbolos '\*/'. Cabe resaltar que el comentario por bloque puede abarcar varios renglones.



**Código con comentarios**



NOTA. Al iniciar el programa se deben agregar todas las bibliotecas que se van a utilizar en el mismo, es decir, funciones externas necesarias para ejecutar el programa. En lenguaje C la biblioteca estándar de entrada y salida está definida en 'stdio.h' (standard in out) y provee, entre otras, funciones para lectura y escritura de datos que se verán a continuación.

**Declaración de variables**

Para declarar variables en C se sigue la siguiente sintaxis:

[modificadores] tipoDeDato identificador [= valor];

Por lo tanto, una variable puede tener modificadores (éstos se analizarán más adelante y son opcionales), debe declarar el tipo de dato que puede contener la variable, debe declarar el identificador (nombre o etiqueta) con el que se va a manejar el valor y se puede asignar un valor inicial a la variable (opcional).

También es posible declarar varios identificadores de un mismo tipo de dato e inicializarlos en el mismo renglón, lo único que se tiene que hacer es separar cada identificador por comas.

tipoDeDato identificador1[= valor], identificador2[= valor];

Tipos de datos

Los tipos de datos básicos en C son: 

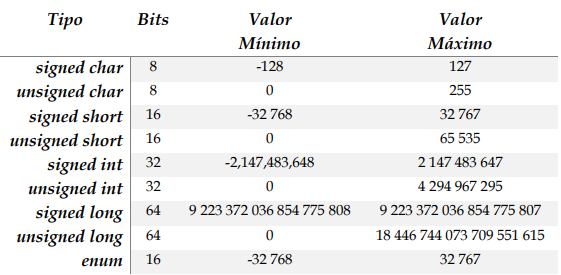
Caracteres: codificación definida por la máquina. 

Enteros: números sin punto decimal. 

Flotantes: números reales de precisión normal. 

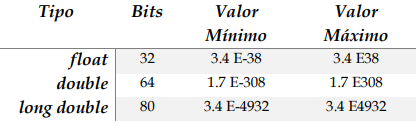
Dobles: números reales de doble precisión.

Las variables enteras que existen en lenguaje C son:



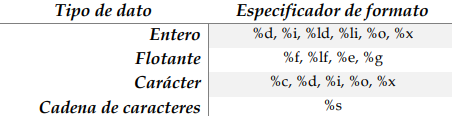
Si se omite el clasificador por defecto se considera 'signed'.

Las variables reales que existen en lenguaje C son:



Las variables reales siempre poseen signo.

Para poder acceder al valor de una variable se requiere especificar el tipo de dato. Los especificadores que tiene lenguaje C para los diferentes tipos de datos son:



El especificador de dato se usa para guardar o imprimir el valor de una variable.

Para imprimir un valor entero en base 10 se pueden usar los especificadores %d o %i (%ld ó %li para enteros largos), para imprimir un valor entero en base 8 se utiliza el especificador %o y para imprimir un valor entero en base 16 se utiliza el especificador %x.

Un valor real se puede imprimir con el especificador %f para reales de precisión simple, %lf para reales de doble precisión, %e (notación científica) y %g (redondea la parte fraccionaria a 3 dígitos significativos).

Una variable de tipo carácter se puede imprimir con el especificador %c, con los especificadores %i o %d para imprimir el valor del código ASCII del carácter en base 10, con el especificador %o para imprimir el valor del código ASCII del carácter en base 8 o con el especificador %x para imprimir el valor del código ASCII del carácter en base 16.

El lenguaje C también permite manejar cadenas de caracteres y éstas se pueden imprimir con el especificador %s.

NOTA: Las cadenas de caracteres se manejarán cuando se aborde el tema de arreglos.

**Identificador**

Un identificador es el nombre con el que se va a almacenar en memoria un tipo de dato. Los identificadores siguen las siguientes reglas:

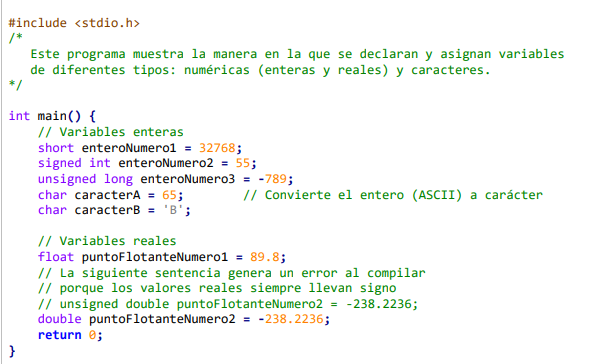
Debe iniciar con una letra [a-z]. 

Puede contener letras [A-Z, a-z], números [0-9] y el carácter guión bajo (\_).

NOTA: A pesar de que variables como 'areadeltriangulo' o 'perimetro\_del\_cuadrado' son declaraciones válidas como identificadores, es una buena práctica utilizar la notación de camello para nombrar las variables como convención.

En la notación de camello los nombres de cada palabra empiezan con mayúscula y el resto se escribe con minúsculas (a excepción de la primera palabra, la cual inicia también con minúscula). No se usan puntos ni guiones para separar las palabras. Además, las palabras de las constantes se escriben con mayúsculas y se separan con guion bajo.

**Código declaración de variables**

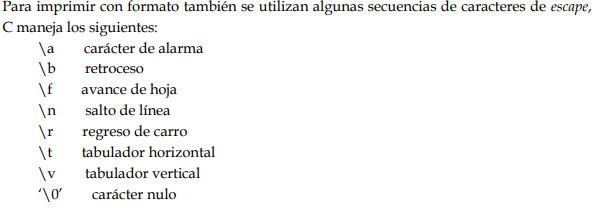


La biblioteca 'stdio.h' contiene diversas funciones tanto para imprimir en la salida estándar (monitor) como para leer de la entrada estándar (teclado).

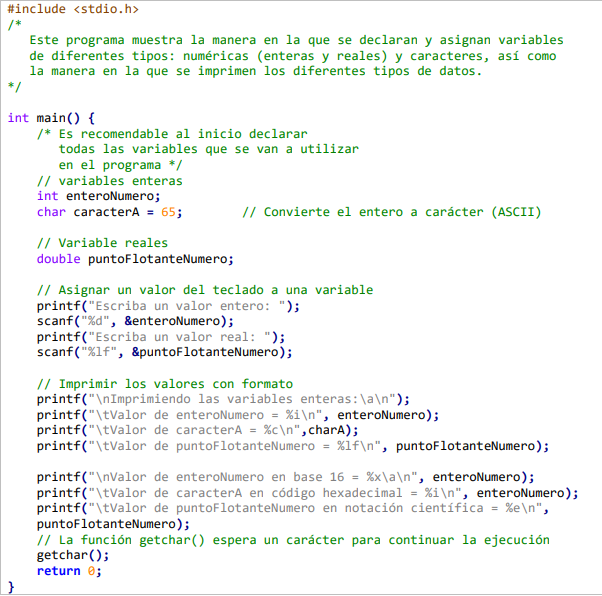
printf es una función para imprimir con formato, es decir, se tiene que especificar entre comillas el tipo de dato que se desea imprimir, también se puede combinar la impresión de un texto predeterminado:



scanf es una función que sirve para leer datos de la entrada estándar (teclado), para ello únicamente se especifica el tipo de dato que se desea leer entre comillas y en qué variable se quiere almacenar. Al nombre de la variable le antecede un ampersand (&), esto indica que el dato recibido se guardará en la localidad de memoria asignada a esa variable.



**Código almacenar e imprimir variables**



**Modificadores de alcance**

Los modificadores que se pueden agregar al inicio de la declaración de variables son const y static.

El modificador const impide que una variable cambie su valor durante la ejecución del programa, es decir, permite para crear constantes. Por convención, las constantes se escriben con mayúsculas y se deben inicializar al momento de declararse.

Ejemplo:



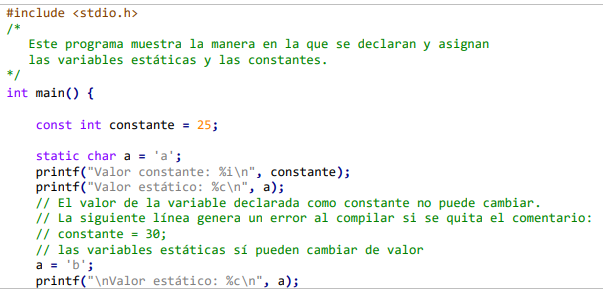
El modificador static indica que la variable permanece en memoria desde su creación y durante toda la ejecución del programa, es decir, permanece estática en la memoria.

Ejemplo



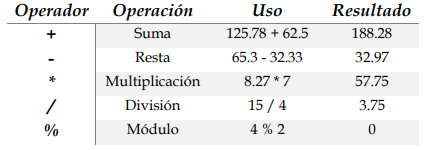
NOTA: Cuando se llegue al tema de funciones se verá la utilidad de las variables estáticas.

**Código variables estáticas y constantes**

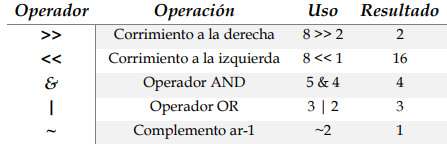


**Operadores**

Los operadores aritméticos que maneja lenguaje C se describen en la siguiente tabla:

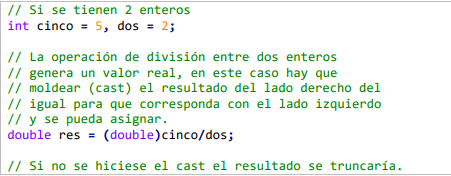


Los operadores lógicos a nivel de bits que maneja el lenguaje C se describen en la siguiente tabla:

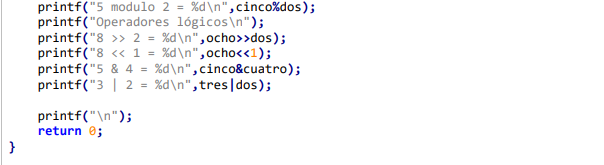
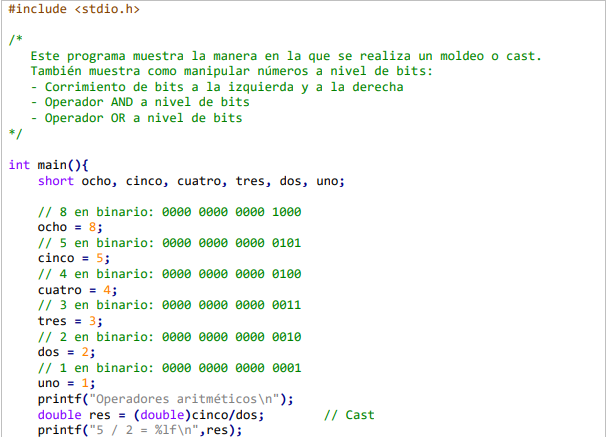


**Moldeo o cast**

El resultado de una operación entre dos tipos de datos iguales puede dar como resultado un tipo de dato diferente, en esos casos es necesario moldear el resultado. A este proceso se le conoce como cast.



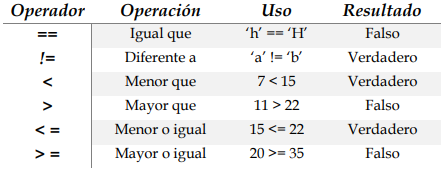
**Código operadores**



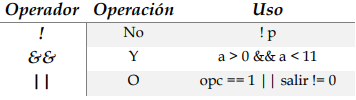
**Expresiones lógicas**

Las expresiones lógicas están constituidas por números, caracteres, constantes o variables que están relacionados entre sí por operadores lógicos. Una expresión lógica puede tomar únicamente los valores verdadero o falso.

Los operadores de relación permiten comparar elementos numéricos, alfanuméricos, constantes o variables.



Los operadores lógicos permiten formular condiciones complejas a partir de condiciones simples.



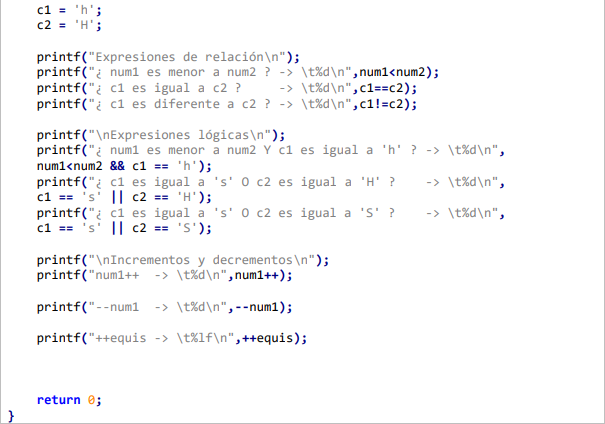
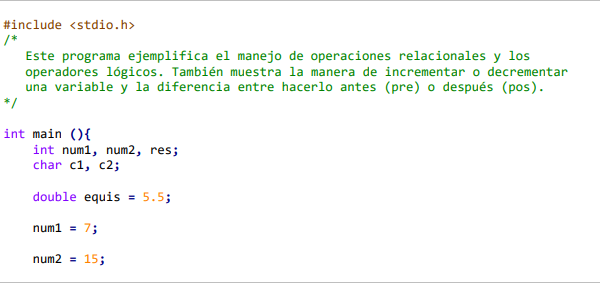
Lenguaje C posee operadores para realizar incrementos y decrementos de un número.

El operador ++ agrega una unidad (1) a su operando. Es posible manejar preincrementos (++n) o posincrementos (n++).

El operador -- resta una unidad (1) a su operando. Se pueden manejar predecrementos (-- n) o posdecrementos (n--).

NOTA: Lenguaje C maneja los resultados booleanos (Verdadero o falso) con números enteros, cuando el resultado de una comparación es falso el valor regresado es cero, cuando la comparación es verdadera el valor regresado es 1.

**Código expresiones lógicas**



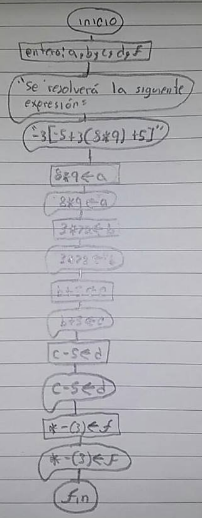
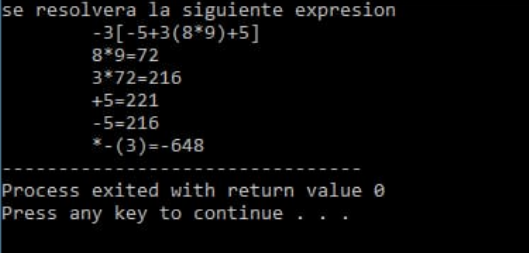
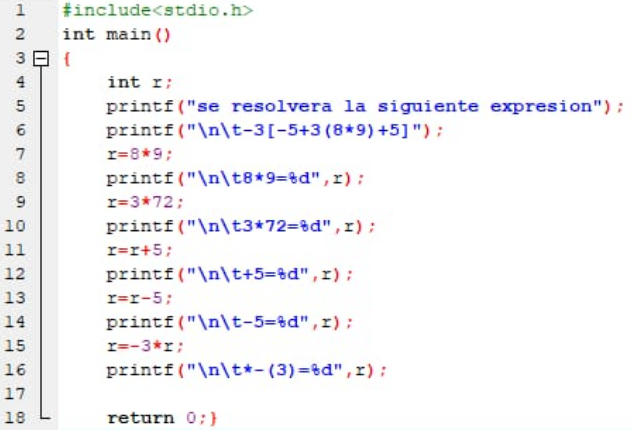
**Depuración de programas**

Cuando un programa falla (no termina su ejecución de manera correcta) y la información enviada por el compilador es muy general, se puede ejecutar el programa en un contexto controlado para saber, exactamente, dónde está fallando. Se revisará este tema en la guía práctica de estudio “Depuración de programas” para conocer las diferentes herramientas que nos ayudan a encontrar los errores de un programa.

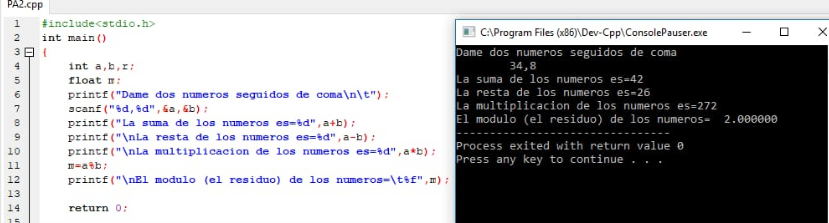
**Ejercicios de tarea**

Realizar el pseudocódigo, diagrama de flujo y programa de los siguientes ejercicios:

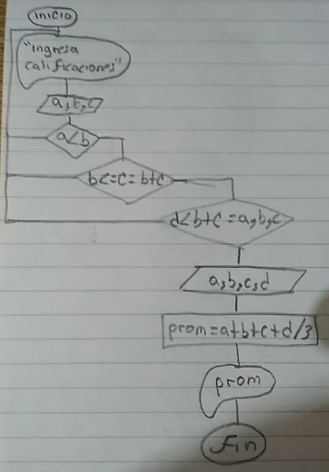
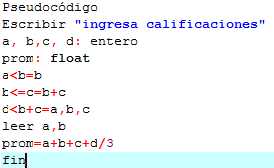
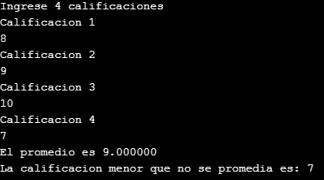
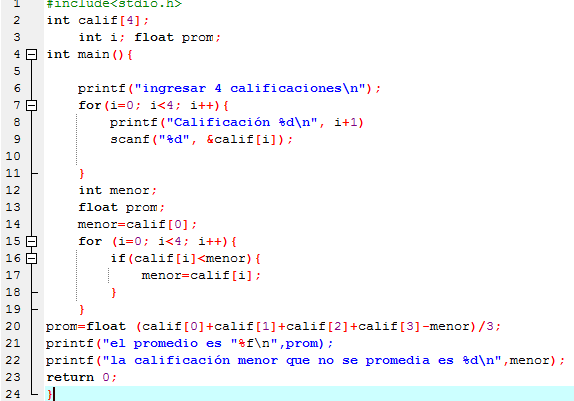
1. Desarrolle un programa que resuelva por pasos la siguiente expresión -3[-5+3(8\*9)+5]



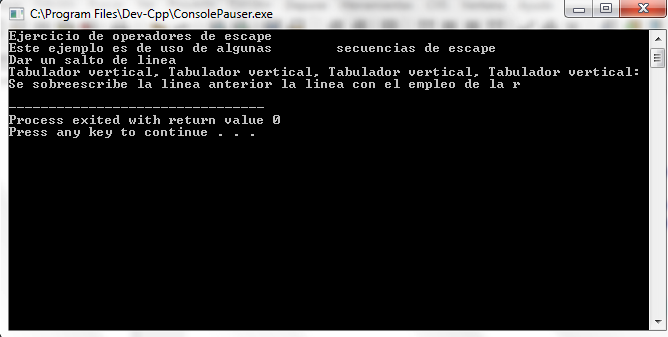
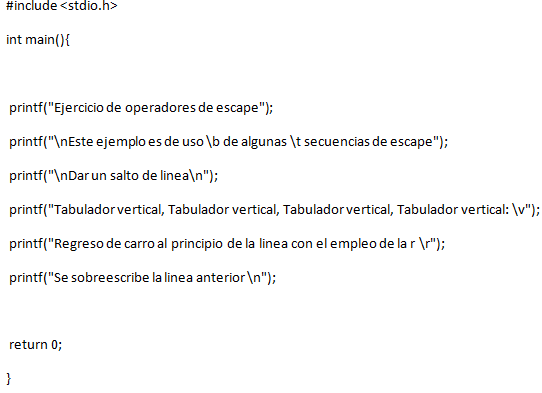
1. Usando las tabla de verdad de la función lógica AND (&) y OR ( | ) indique la respuesta de AB!=CD&&A|D/&&B|D
2. Programar una calculadora de dos números que permita realizar la suma, resta, Multiplicación, Modulo y que se muestren en pantalla los resultados. (Se requiere entregar el DFD, Pruebas, Pseudocódigo, pruebas y condigo fuente en C y sus pruebas de salida).



1. Solicitar las calificaciones de sus prácticas, el promedio de prácticas de su curso se calcula en base a cuatro prácticas calificadas de las cuales se elimina la nota menor y se promedian las tres notas más altas.



1. Ejecutar el siguiente código y explicar que hace cada secuencia de caracteres de escape(Los caracteres de escape son \n, \t, etc.)



\n: **aplica un salto de línea**

\t: **tabula**

\b: **retrocede**

\v: **tabula de forma vertical**

\r: **regresa al inicio de la línea**

**Conclusiones**

Me sigue resultando demasiado difícil lograr que el código compile, de hecho hay unos programas que no pude ejecutar, no se si mi aplicación está mal o algo así, porque con el mismo código intenté compilarlo 3 veces y no pude, me decía que no tenía declaradas ciertas variables, pero al revisar si estaban declaradas. De igual manera, me cuesta mucho trabajo ver que hacen los programas cuando veo el código, me falta repasar muchísimo para lograr atenuar tan siquiera un poco este gran problema que vengo arrastrando desde la práctica pasada.

**Referencias**

1.- El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, segunda edición, USA, Pearson Educación 1991.

2.- Carlos Guadalupe (2013). Aseguramiento de la calidad del software (SQA). [Figura 1]. Consulta: Junio de 2015. Disponible en: <https://www.mindmeister.com/es/273953719/aseguramiento-de-la-calidad-delsoftware-sqa>