**Taller #1.**

**Comprensión de código, lenguajes compilados, lenguaje C, compilador Gcc**

**Tiempo estimado: 2 a 3 horas**

1. **Preguntas generales**

Compile como le fue presentado en clase el programa *Taller1.c.* Luego conteste las siguientes preguntas:

* 1. ¿Para qué sirve la instrucción **#include**?
  2. ¿Cuál es el objetivo principal de ese programa?
  3. ¿Cuántas variables tiene en total?
  4. ¿Cuáles de esas variables son variables locales y cuáles son variables globales? ¿Por qué?
  5. ¿Cuántas funciones definidas por el usuario tiene ese programa?
  6. ¿Cuál es el nombre de cada una de esas funciones?
  7. ¿Para qué cree usted que se convierten los números enteros en flotantes en la función cuatro? Explique
  8. ¿Explique qué es la función main y por qué es importante?

1. **Modificar y contestar**
   1. Cambie la función **main** de posición. Córtela y péguela luego de la declaración de las librerías. Compile y ejecute nuevamente el programa. ¿Qué resultado obtuvo? ¿Por qué cree usted que se obtuvo ese resultado?
   2. Modifique la función *funcionDos.* Borre la declaración de la variable *x*. Compile y ejecute nuevamente el programa. ¿Qué resultado obtuvo? ¿Por qué cree usted que se obtuvo ese resultado?
2. **Mejorar**

**Nota**: Para desarrollar este punto ubique la función **main** nuevamente al final del archivo y declare nuevamente la variable *x* en la *funciónDos* para que el programa compile y se ejecute sin errores.

El programa *taller1.c* no se encuentra correctamente documentado. El nombre del programa no se relaciona con la funcionalidad principal, los nombres de las funciones no tienen relación con su objetivo, los nombres de las variables tampoco tienen relación con los elementos que almacenan, ni tiene documentación en las funciones que facilite la comprensión del código.

Modifique el programa actual y:

* 1. Renombre el archivo fuente de acuerdo con el objetivo principal del programa que fue especificado en el numeral 1.2.
  2. Renombre cada función del programa, de acuerdo con la ocupación principal que realiza cada función.
  3. Renombre cada variable para que el nombre tenga relación con su utilidad dentro del programa.
  4. Modifique los printf después de invocar a cada función para que indique a qué operación corresponde cada resultado.
  5. Pruebe que el nuevo programa compile y se ejecute correctamente y guárdelo con el nombre que usted crea que deba tener, más las iniciales de su nombre completo. Por ejemplo, si el programa sirviera para obtener las tablas de multiplicar, el programa se llamaría: tablasMultiplicarLGNA.c . La parte en verde es el nombre del programa, la parte en roja corresponde a sus iniciales, que en mi caso es LGNA. (por si las moscas, no le tiene que quedar el nombre con colores)

Para la documentación del programa tenga en cuenta el siguiente estándar de nombramiento:

Use nombres en lo posibles cortos y con un significado claro. La primera letra debe ser minúscula, si son más de 2 palabras se pone la primera letra de la primera palabra en minúscula y las iniciales de las demás palabras en mayúsculas. Además, para las funciones, el nombre debe comenzar por un verbo en infinitivo. Esta notación se llama ***lowerCamelCase.***

***Ejemplos de funciones:*** *quitarBoton, calcularCredito, sumarNumeros*

***Ejemplos de variables****: sumaGeneral, promedio, nroHabitantes.*

1. **Crear**

Para esta parte del taller debe consultar:

Como leer y escribir números enteros y como escribir cadenas en C.

Uso de condicionales en C.

Cómo calcular el módulo entre dos números para saber si una división es exacta.

* 1. Adicione al programa que modificó en el numeral 3, un procedimiento que le ingresa por parámetro un número entero de nueve (9) dígitos y debe indicar por pantalla si este número es o no palíndromo, es decir, que el número se puede leer igual de izquierda a derecha que de derecha a izquierda. (pista: se podría ayudar de uso de la división y el módulo)

Restricciones:

* No puede hacer uso de arreglos (cadenas, vectores, listas, matrices, etc.)
* Si desea, puede hacer uso de ciclos, aunque no es obligatorio.
* En la función main debe pedir el número de 9 dígitos al usuario, y hacer el llamado al procedimiento. El procedimiento debe estar creado afuera del main.
* Recuerde el uso de lowerCamelCase y la documentación de su código.
  1. Adicione al programa que modificó anteriormente, una función que calcule si un año es o no bisiesto y retorne 1 si el año es bisiesto y 0 si el año no es bisiesto.
  2. En la función main adicione: una línea que solicite el ingreso del año a analizar y la invocación a la función que calcula si el año es bisiesto.
  3. Si la función que calcula si el año es bisiesto retorna un 1, entonces imprima en pantalla “El año AAAA es bisiesto” y “mi nombre es XXXXX”. Donde, AAAA corresponde al año ingresado por el usuario y XXXXX corresponderá a su nombre. Si la función que calcula si el año es bisiesto retorna 0, entonces imprima en pantalla: “El año AAAA no es bisiesto y tengo YY hermano(s)”. Donde YY corresponde a la cantidad de hermanos que tenga. Si no tiene hermanos ponga 0. (También por si las moscas, tenga en cuenta que su nombre y su cantidad de hermanos es un valor arbitrario que usted digita, no necesita hacer cálculos ni nada parecido)

La lógica de este punto la podrá incluir en la función **main**, o en otra función creada por usted que sea llamada en el método **main**.

Tenga en cuenta que un año es bisiesto en dos casos posibles:

* Es divisible por 4 y no divisible por 100.
* Es divisible por 400. Por ejemplo, los años 1800 y 1900 no fueron bisiestos, pero sí lo fueron el 2000 y el 2012.

Un ejemplo del resultado de la ejecución del programa sería:



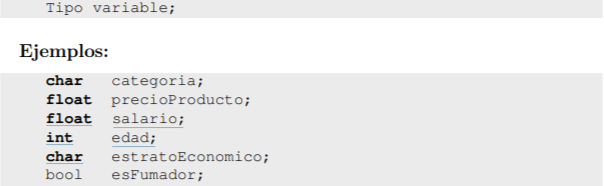
**Desarrollo**

1. **Preguntas generales**
   1. La instrucción **#include** sirve para utilizar funciones que no estén integradas dentro del programa C. Por dicha razón, utilizamos la directiva **#include**, que nos permite añadir librerías o funciones que se encuentran en otros ficheros a nuestro programa.
   2. El objetivo principal del Lenguaje C, es ser un lenguaje de sistema, considerado de propósito general y lo suficientemente cercano al procesador (sistema) para ser eficiente y poderoso, como para crear, desde sistemas operativos hasta aplicaciones generales de un usuario promedio.
   3. Es Importante recordar que Lenguaje C no tiene un tipo especial para las cadenas y los datos lógicos. Las cadenas deben ser trabajadas como un conjunto de **char** y los datos lógicos como elementos de tipo **int** o **char**, o emplear el tipo **bool** declarado en la biblioteca **stdbool.h**. El nombre o identificador de la variable, da el nombre o identifica la

posición de la memoria a la que se hace referencia.

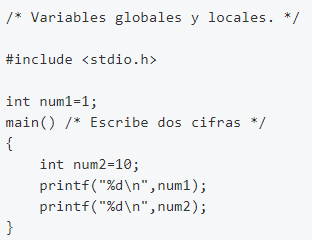
Contenido, es el valor que almacena, por ejemplo, 20, “Gabriela”, 45579.35, etc.

Los programas en Lenguaje C requieren de la declaración de variables para almacenar los datos. Declarar una variable significa separar un espacio de memoria, usando un nombre (identificador) y un tipo de dato.



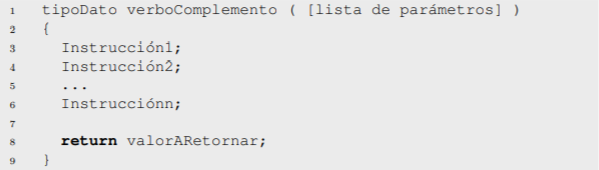
* 1. Las **variables globales** son aquellas que permanecen activas durante todo el programa. Se crean al iniciarse éste y se destruyen de la memoria al finalizar. Pueden ser utilizadas en cualquier función. Mientras que las **variables locales** son creadas cuando el programa llega a la función en la que están definidas. Al finalizar la función desaparecen de la memoria.

Si dos variables, una global y una local, tienen el mismo nombre, la local prevalecerá sobre la global dentro de la función en que ha sido declarada. Dos variables locales pueden tener el mismo nombre siempre que estén declaradas en funciones diferentes.



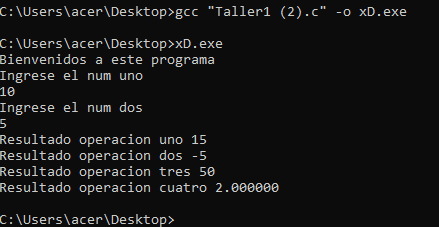
* 1. Una función es un tipo especial de procedimiento que entrega (retorna) un valor como resultado. Son funciones que no retornan ningún tipo de valor, aunque en principio esa definición es correcta, en realidad es imprecisa desde el punto de vista del concepto matemático de función del cual se deriva este término. La declaración de una función es semejante a la de un procedimiento, con la diferencia que no se escribe **void** antes del nombre de la función, sino el tipo del dato que la función retorna.

La forma general de esta estructura es presentada en el segmento del Programa:



* 1. Las funciones propias del Lenguaje C son: scanf, fgets, printf, sin, cos, tan, sqrt, main, entre otras.
  2. Porque hay la necesidad de dar un resultado que n necesariamente de un número entero, sino también un número decimal, es como si intentáramos dividir x= 7 / y=2, si lo ponemos en entero, con **int** nos da 4 redondeado, pero como se quiere tener certeza, se pone **float** para poner el resultado en decimal, ósea en 3.5.
  3. Todos los programas en lenguaje C deben tener una main o un principal() función. Es el núcleo de cada programa. Es necesario. los principal() función en realidad no tiene que hacer aparte de estar presente dentro de su código fuente en C nada. Con el tiempo, contiene instrucciones que indican a la computadora para llevar a cabo cualquier tarea que su programa está diseñado para hacer. Sin embargo, no se requiere oficialmente que hacer nada.

1. **Modificar y contestar**
   1. Se traba el programa, porque ponemos el main en las definiciones de las variables, y pues se confunde el programa, porque ejecuta variables y no operaciones.
   2. Pone el valor del num dos en negativo, en este caso “-5”.



Bibliografía

1. Delgado, H. (2021). Inclusión de archivos de cabecera - Directiva #include. Publicado el 8 de junio de 2021. Recuperado de <https://disenowebakus.net/inclusion-ficheros.php>
2. Pulgarín, R. Herrera, O & Gutiérrez, J. (2018). Introducción a la Programación en C. Publicado en Armenia - Quindío – Colombia. Recuperado de <http://sara.uniquindio.edu.co/wp-content/uploads/2020/07/Programaci%C3%B3n-I-Lenguaje-C.pdf>