**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**?

Se pone “#include <stdio.h>” al principio de cualquier programa escrito en C, debido a que, si se excluye esta librería, acciones básicas como la entrada y salida serán imposibles de hacer.

* 1. ¿Cuál es el objetivo principal de ese programa?

El programa se ha construido en base a que, dados 2 números pedidos al usuario, se hará una serie seguida de cuatro operaciones posibles que incluyen: sumar, restar, multiplicar y dividir los dos valores que se dieron por el usuario.

* 1. ¿Cuántas variables tiene en total?

Están: result de “funcionCuatro”, y a, b, c y d de la función “main”, dando un total de 5 variables.

* 1. ¿Cuáles de esas variables son variables locales y cuáles son variables globales? ¿Por qué?

Todas las 5 variables mencionadas son variables locales, pues cada una de ellas se encuentra dentro de sus respectivas funciones. Si, por ejemplo, una de estas fuera una global, se definiría por fuera de la función, como, por ejemplo: “float result = 0.0” estaría en la misma indentación del #include.

* 1. ¿Cuántas funciones definidas por el usuario tiene ese programa?

Se puede decir que las primeras 4 funciones que van antes del “main” son definidas por el usuario, debido a que estas tienen un contenido específico definido para cada una de ellas, mientras que no importa cuánto contenido único que se le añada a esta por el usuario,

* 1. ¿Cuál es el nombre de cada una de esas funciones?

funcionUno, funcionDos, funcionTres, y funcionCuatro.

* 1. ¿Para qué cree usted que se convierten los números enteros en flotantes en la función cuatro?

Para tener una exactitud de datos antes, durante y después de la división, además del manejo propio de datos que se dan como producto de la operación matemática, es necesario utilizar datos float en vez de datos int.

* 1. ¿Explique qué es la función main y por qué es importante?

Como se mencionó anteriormente, la función “main” es la parte vital de cualquier código de C, debido a que las operaciones especificadas en este son las que se van a correr principalmente cada vez que se corra el código compilado.

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?

En sí, no es posible ejecutar el programa con los cambios pedidos, debido a que se está tratando de compilar el programa de forma en la que no se conocen las funciones que se llaman en “main”, lo cual resulta en una variedad de errores.

* 1. 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?

Como la variable x es requerida como parámetro para hacer la operación hecha por la “funcionDos”, borrarla manda un mensaje de error de “x no está declarada” al momento de compilar el código, además de los otros errores que aparecen en el resto del código que aparecen en las filas respectivas donde se trata de llamar a x.

1. **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:

