**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**?

Sirve para importar serie de librerías que se especifican después del #include.

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

Realizar una serie de cálculos usando los operadores aritméticos básicos (suma, resta, multiplicación y división) mediante dos números insertados.

* 1. ¿Cuántas variables tiene en total?

Hay 4 variables principales a, b, c y d, pero si contamos las variables individuales de las funciones serian 8 mas y la variable del result que en total serian 13 variables.

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

Todas las variables son locales ya que están dentro de la función main y las otras están adentro de sus funciones.

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

4 sin contar la función main ya que sin la función main no habría programa y pues si la contamos 5.

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

main(), funcionUno, funcionDos, funcionTres y funcionCuatro

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

Nos llegan dos enteros, si yo divido dos enteros y no los convierto en flotantes, el resultado en la funcionCuatro si divido dos números que den un numero decimal no me va a dar el resultado correcto me da 0.0000000 ya que coge solo el numero entero sin aproximar nada lo cual nos da 0, por ejemplo, si dividimos 3/4 nos da 0.75 pero como no recibe decimales coge el 0 como entero y le agrega los ceros, si lo hago con dos números divisibles si me da un resultado correcto de forma entera.

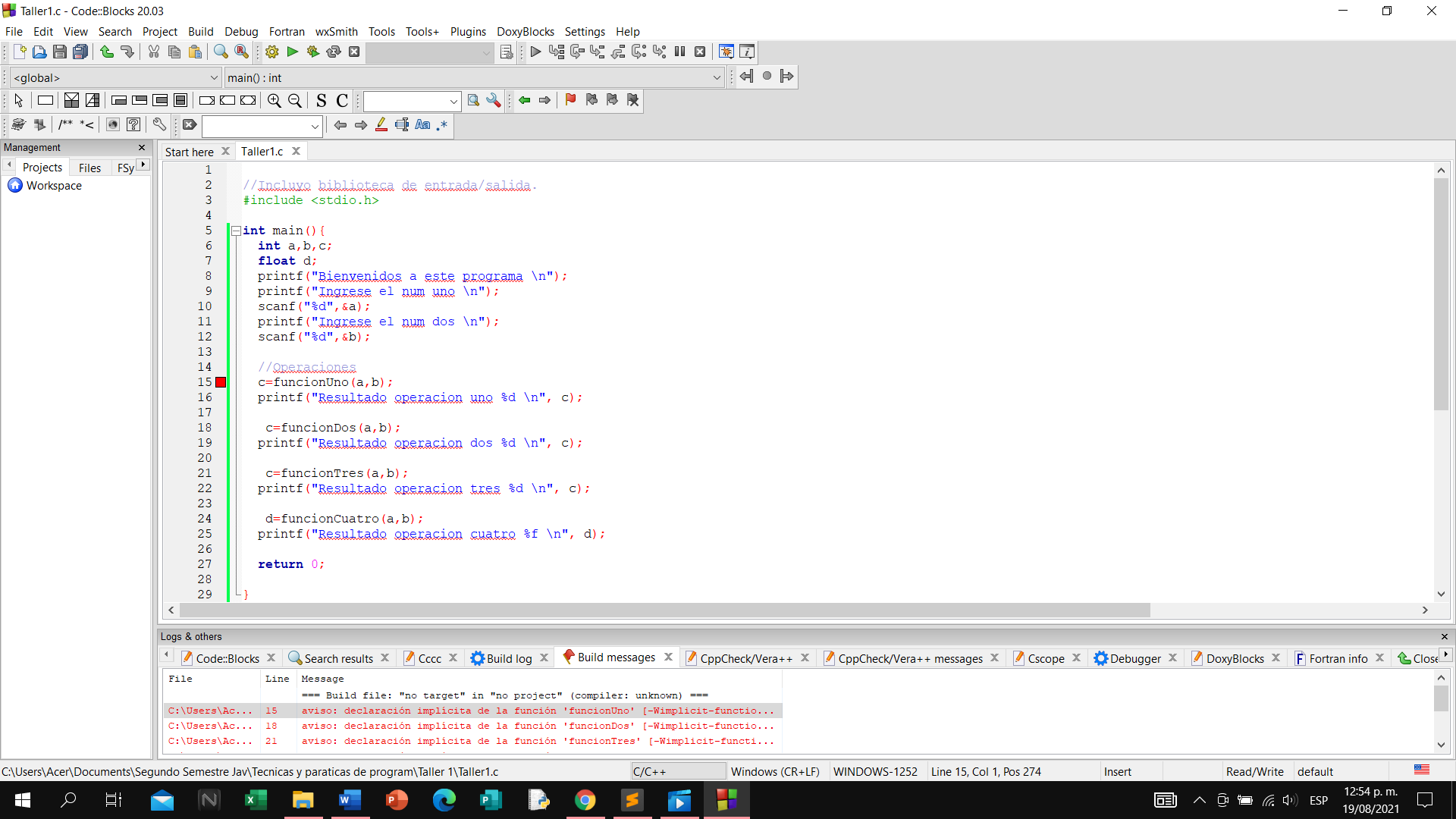
En conclusión, se convierte para evitar que se dividan dos enteros y se pierda la parte decimal.

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

Es la función que se ejecuta por default, es el punto de entrada de cualquier programa en C, sin la función main no hay programa.

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?

Cuando se pone la función main al principio del código después de las librerías, el problema que se genera es el siguiente, normalmente el programa va leyendo y tiene que saber con qué va a trabajar desde el principio para que cuando se busque la funciónUno pueda entrar o encontrarla para hacer los procesos, por eso se declaran las funciones primero, si las funciones no están declaradas en la primera parte y primero se lee la función main, el programa no tiene con que trabajar y nos marca el error, porque no encuentra las funciones que se hallan definido.



* 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?

Si quitamos la variable x de la funcionDos cuando la vaya a llamar no la va encontrar ósea que no está declarada entonces el error es que estamos usando una variable sin declarar retornando a nada y también solo estaría recibiendo un valor.

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:

