**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:



**Juan Francesco García Vargas**

Desarrollo Taller 1 Técnicas y prácticas de programación:

1. Preguntas generales
   1. La instrucción #include en lenguaje C, tiene como función incluir en un código bibliotecas, las cuales permiten realizar determinados tipos de operaciones, como ejemplo más básico de esto existe la función #incluide <stdio.h>, una biblioteca que permite a un código imprimir datos y recibir datos.
   2. El programa con nombre y extensión punto c “Taller1.c” encontrado en la plataforma brigthspace tiene como objetivo principal pedir al usuario dos números en pantalla para realizar cálculos básicos con estos, es decir, sumarlos, restarlos, multiplicarlos y dividirlos. Para esto, el programa usa funciones diferentes para cada operación anteriormente mencionada, además de usar una ultima función para la entrada y lectura de los dos números ingresados por el usuario.
   3. En el programa podemos encontrar un total de 12 variables, 4 dentro de la función *main()*,de estas dos representaran los números que entraran a cada función que hará una determinada operación, otra de las variables representa el resultado de haber ejecutado los números ingresados por el usuario en las funciones para sumar, restar y multiplicar, y finalmente la última variable será para el resultado la función que divide los dos números.

Para cada función de una operación básica existe un *Int x* y un *Int y* que son los números que entran en esa determinada función.

* 1. Las variables que se encuentran en el programa están todas dentro de funciones, unas dentro del *main ()* y otras dentro de las demás funciones, por lo que no encontramos variables fuera de funciones, es decir globales.
  2. El programa contiene 5 funciones en total, 4 de ellas harán las operaciones y una función main hará la lectura de los datos ingresados por el usuario.
  3. La función que suma tiene como nombre funcionUno, la función que resta funcionDos, la función que multiplica funcionTres y la función que divide duncionCuatro.
  4. Al expresar que el resultado de la variable *cociente* es un numero flotante, en c, al ser altamente tipado, se necesita que estos valores para ejecutarse sean del mismo tipo de dato, razón por la cual se convierten los números en flotantes.
  5. La función main es la función principal de un código en c, su traducción al español lo dice. La función main sirve como punto de partida para un código, es decir su aparición es fundamental a la hora de programar en c. Normalmente en la función main se encuentran las invocaciones a las demás funciones que componen un código.

**Modificar y contestar**

1. ¿Qué resultado obtuvo?

El resultado obtenido al compilar el programa modificado es de un error el cual se debe a una declaración explicita. Puntualmente, se ve de esta manera el error…

\modificar.c: En la función 'main':

.\modificar.c:14:5: aviso: declaración implícita de la función 'funcionUno' [-Wimplicit-function-declaration]

c=funcionUno(a,b);

^~~~~~~~~~

.\modificar.c:17:6: aviso: declaración implícita de la función 'funcionDos' [-Wimplicit-function-declaration]

c=funcionDos(a,b);

^~~~~~~~~~

.\modificar.c:20:6: aviso: declaración implícita de la función 'funcionTres' [-Wimplicit-function-declaration]

c=funcionTres(a,b);

^~~~~~~~~~~

.\modificar.c:23:6: aviso: declaración implícita de la función 'funcionCuatro' [-Wimplicit-function-declaration]

d=funcionCuatro(a,b);

^~~~~~~~~~~~~

.\modificar.c: En el nivel principal:

.\modificar.c:40:7: error: tipos en conflicto para 'funcionCuatro'

float funcionCuatro(int x, int y){

^~~~~~~~~~~~~

.\modificar.c:23:6: nota: la declaración implícita previa de 'funcionCuatro' estaba aquí

d=funcionCuatro(a,b);

^~~~~~~~~~~~~

¿Por qué cree que se genera este error?

Este error se genera al declarar en la función principal *main ()* la invocación de otras funciones, que para este caso son, funcionUno, funcionDos, funcionTres y funcionCuatro. Ya que al invocar una función o una operación antes de escribirla generara un error en el que se le estará pidiendo al programa que haga algo que todavía no conoce.

1. ¿Por qué cree que obtuvo ese resultado?

Si borramos de la función *funcionDos* el parámetro que recibe como *int x* el programa generaría un error donde al retornar en la función la expresión *x + y*, no reconocería la variable ya que no fue definida, el error se presenta de la siguiente manera…

.\modificar.c: En la función 'funcionDos':

.\modificar.c:9:10: error: 'x' no se declaró aquí (primer uso en esta función)

return x-y;

^

.\modificar.c:9:10: nota: cada identificador sin declarar se reporta sólo una vez para cada función en el que aparece

.\modificar.c: En la función 'main':

.\modificar.c:37:6: error: demasiados argumentos para la función 'funcionDos'

c=funcionDos(a,b);

^~~~~~~~~~

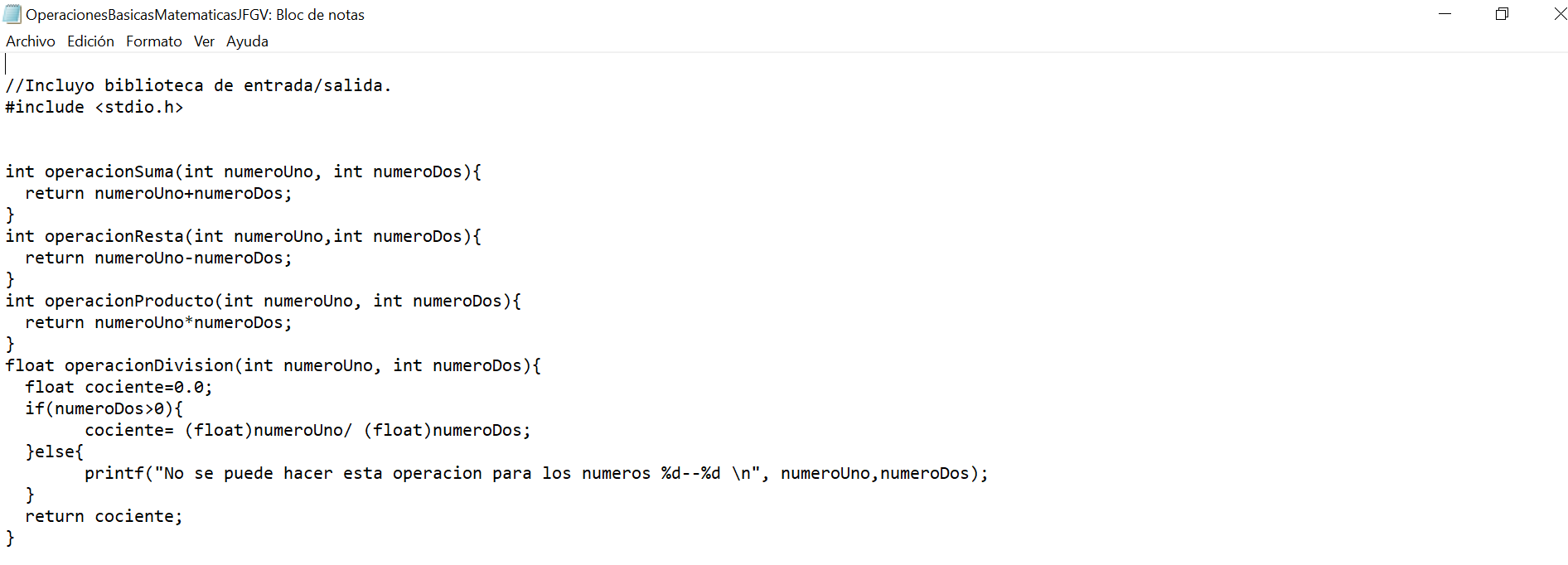
.\modificar.c:8:5: nota: se declara aquí

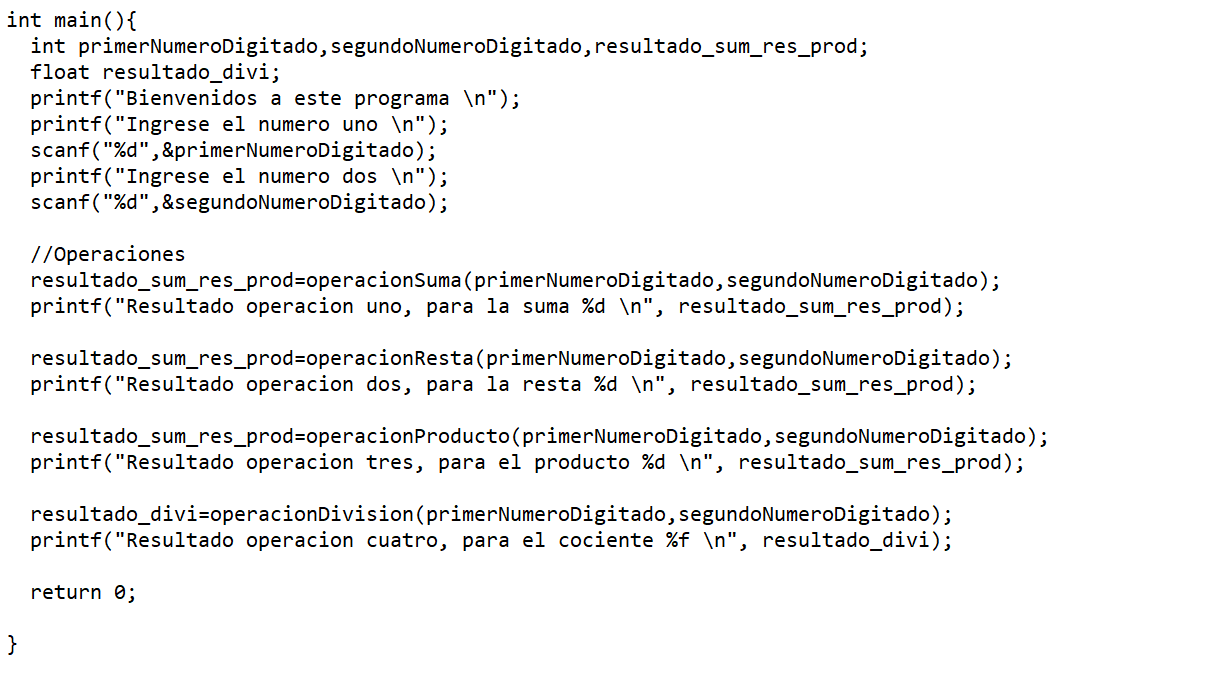
int funcionDos(int y){

^~~~~~~~~~

Ahora bien, si elimináramos de la función *funcionDos* la x que se encuentra en el retorno, pues análogamente obtendremos que la *funcionDos* solo mostraría el mismo número digitado por el usuario guardado en la variable y.

**Mejorar**





Compilación:

