|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre de la práctica** | **Prácticas en C++** | | | **No.** |  |
| **Asignatura:** | **Métodos Numéricos** | **Carrera:** | **ISIC** | **Duración de la práctica (Hrs)** |  |

**Emiliano Arce de Jesús Gpo.3402**

1. **Competencias especificas**

**II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):**

1. **Material empleado:**

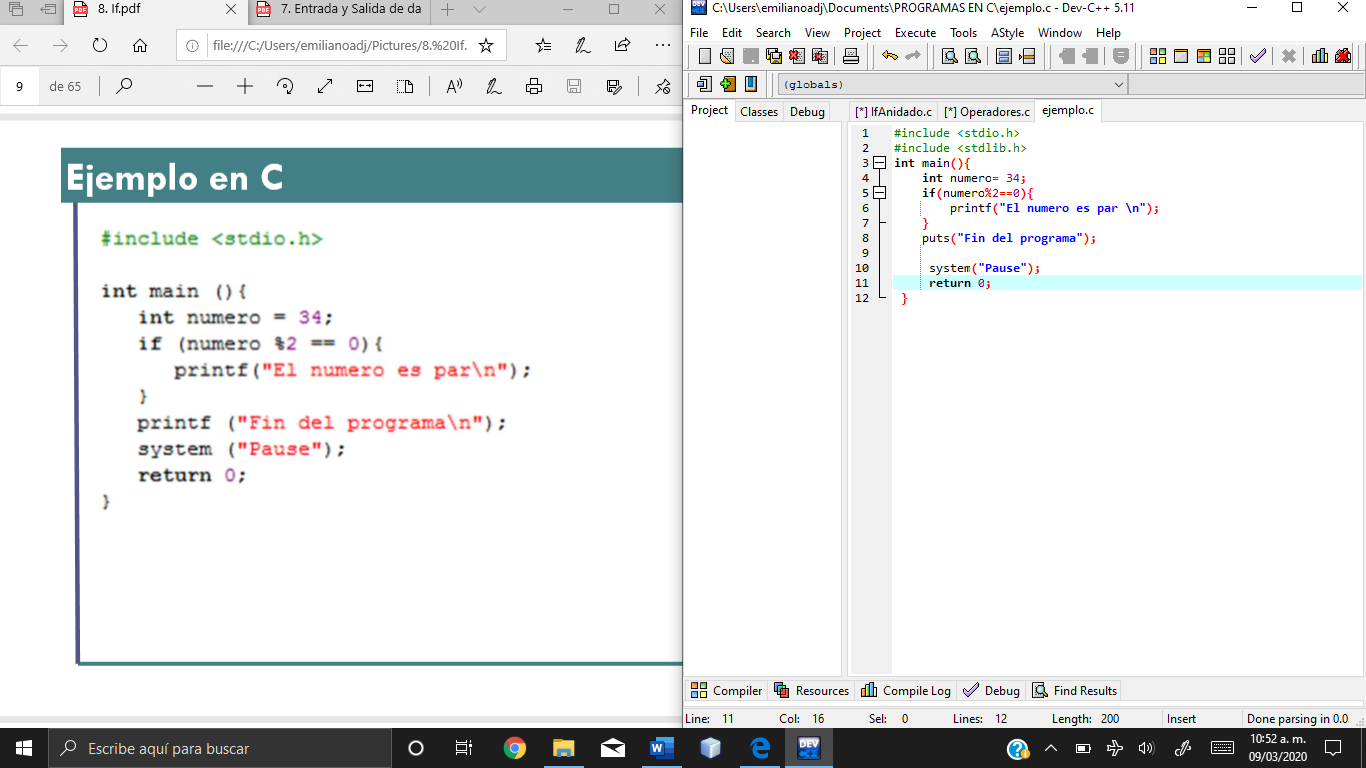
* **Lenguaje de programación estructurada Dev-C++**
* **Manuel de prácticas FO-aca**

1. **Desarrollo de la práctica:**

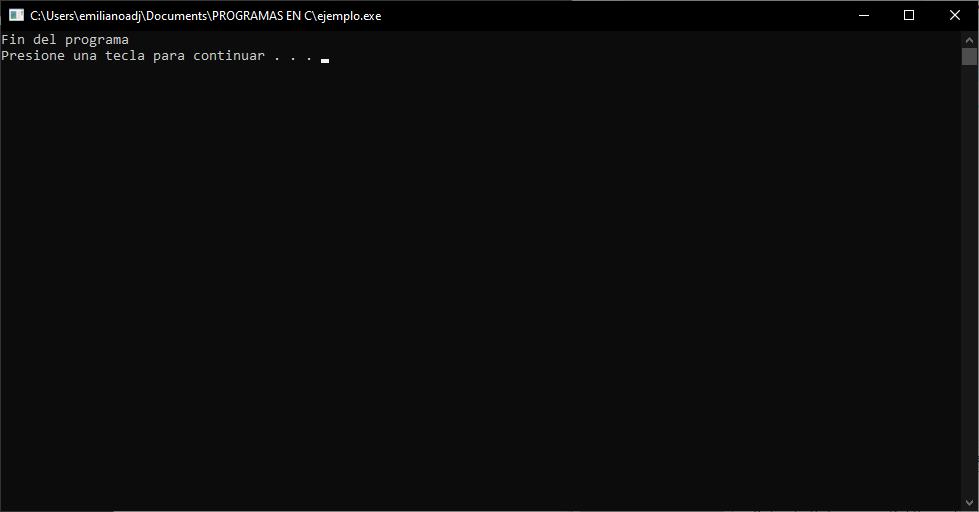
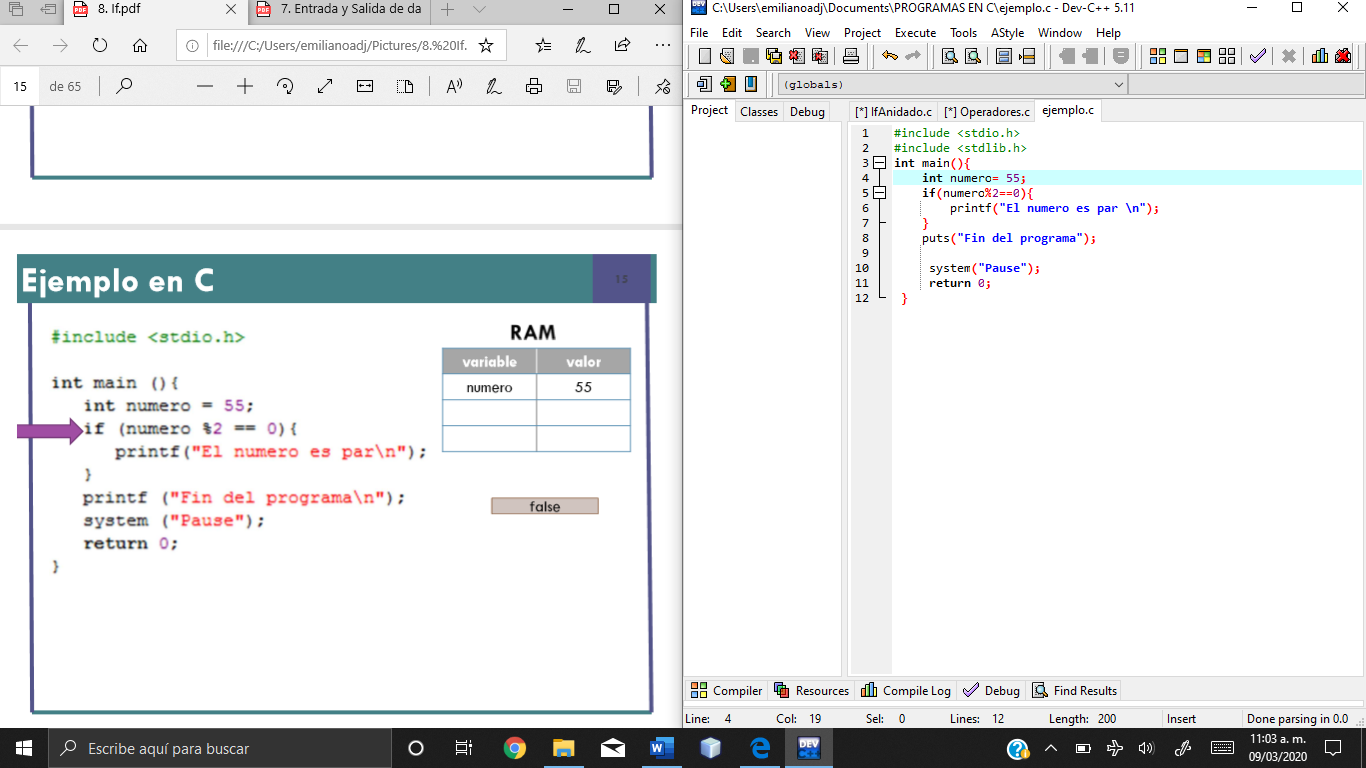
**Las estructuras condicionales como lo son if, for, while, do while, nos sirven de forma que podemos evaluar condiciones y partiendo de que si la condición es verdadera o falsa se deberá hacer un bloque de instrucciones, vamos a ver algunos ejemplos con la estructura condicional if.**

**Una estructura condicional simple, consiste en que si la condición es verdadera entonces ejecuta una o varias opciones, de lo contrario seguirá la ejecución normal del programa.**

**IF**



**En el programa anterior se puede apreciar como de un número ya establecido se le obtiene el residuo (módulo) y que este se evalúa de tal forma que si el resultado es igual a cero se deberá enviar un mensaje en pantalla que el numero es par, de lo contrario manda un mensaje que muestra el fin de dicho programa.**

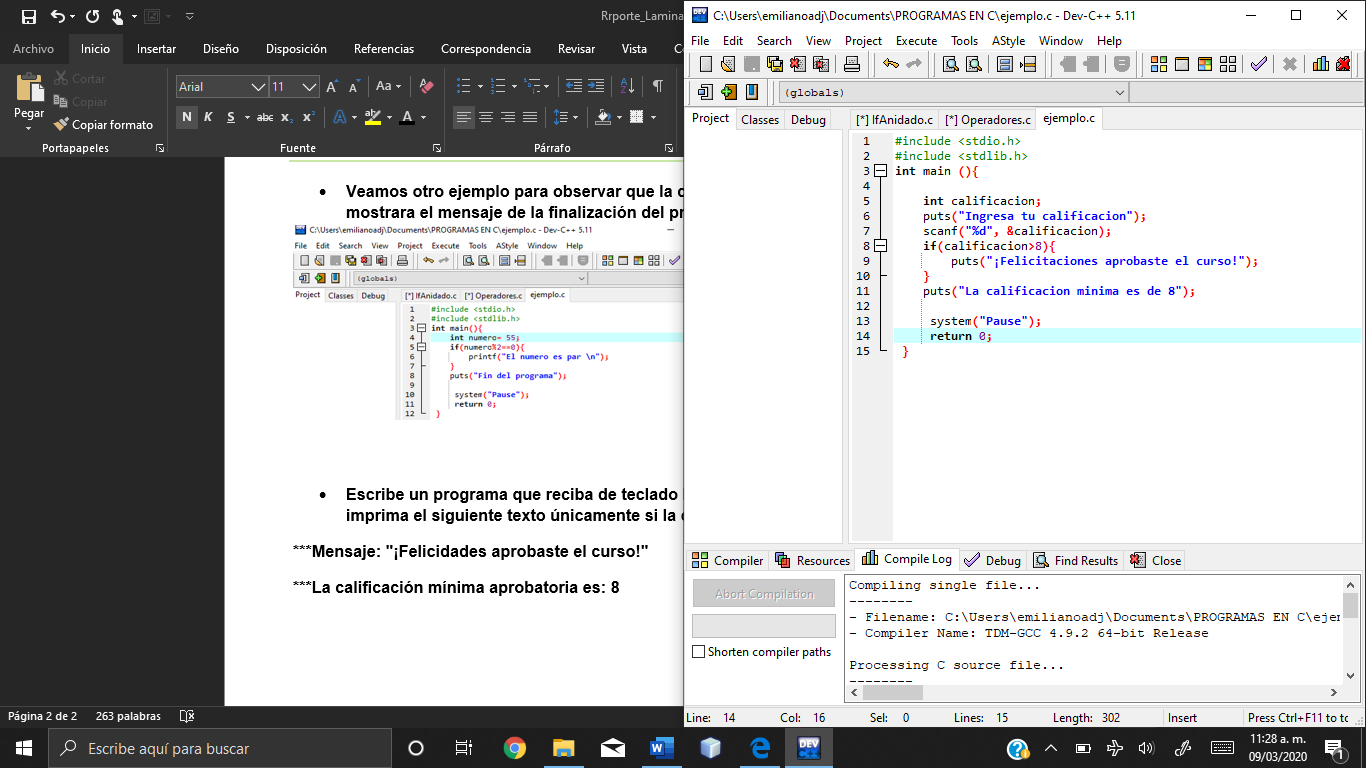
* **Veamos otro ejemplo para observar que la condición no es verdadera y en este caso mostrara el mensaje de la finalización del programa.**

**En este caso la conficion fue negativa y este es el mensaje que muestra.**

* **Escribe un programa que reciba de teclado la calificación del examen de un alumno e imprima el siguiente texto únicamente si la calificación fue aprobatoria.**

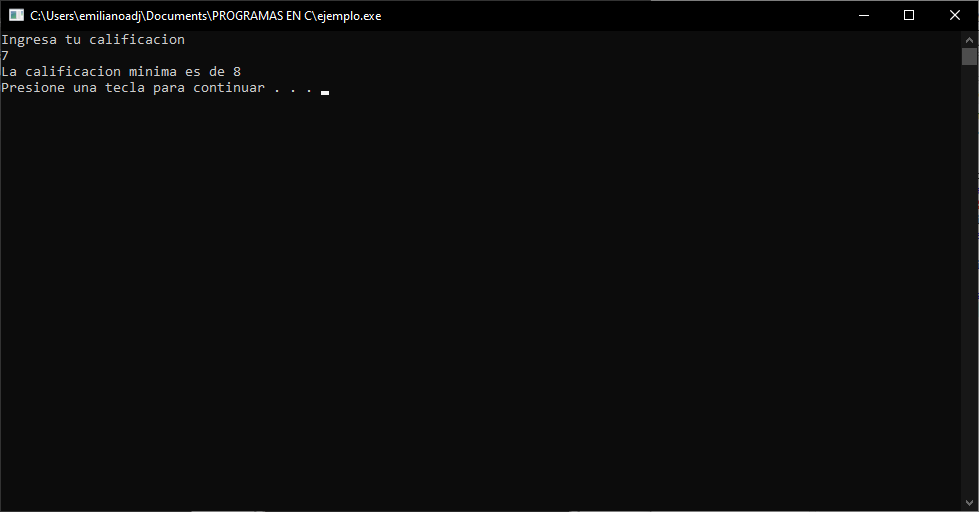
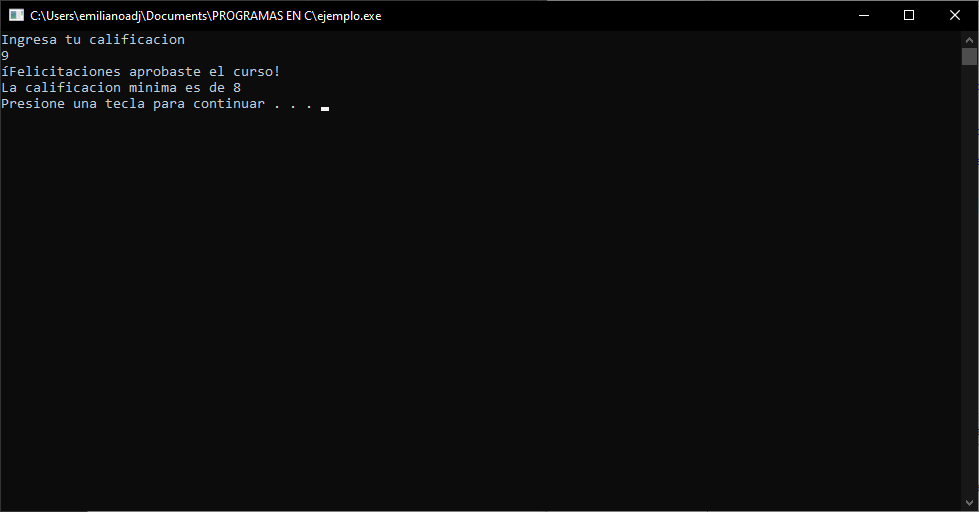
**\*\*\*Mensaje: "¡Felicidades aprobaste el curso!"**

**\*\*\*La calificación mínima aprobatoria es: 8**



**Este programa verifica si la calificación que el usuario ingresa es mayor a 8, ya que el promedio mínimo es de 8.**

**Entonces veamos una prueba de su funcionamiento,**



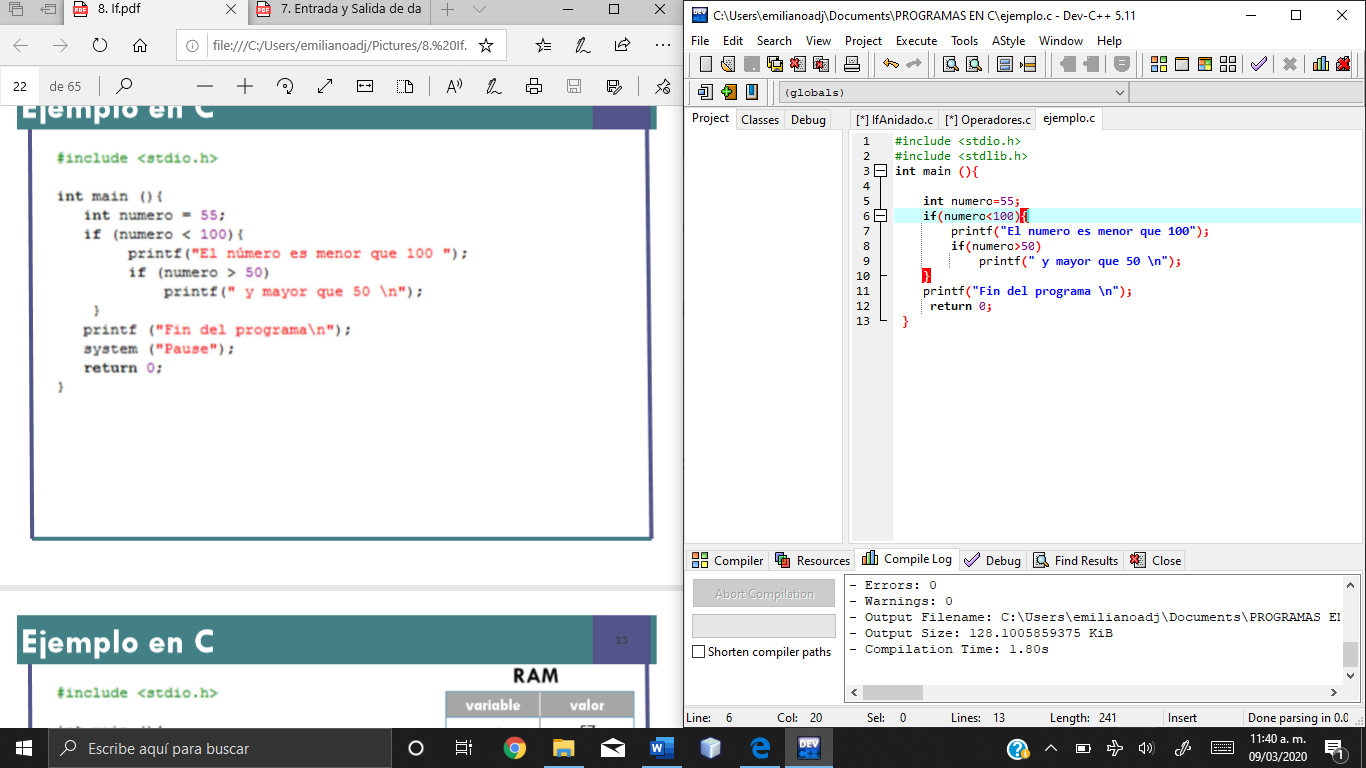
Este fue el caso contrario, la calificación que se ingreso fue de 7, por lo cual deberá mostrar:

La calificación mínima es de 8.

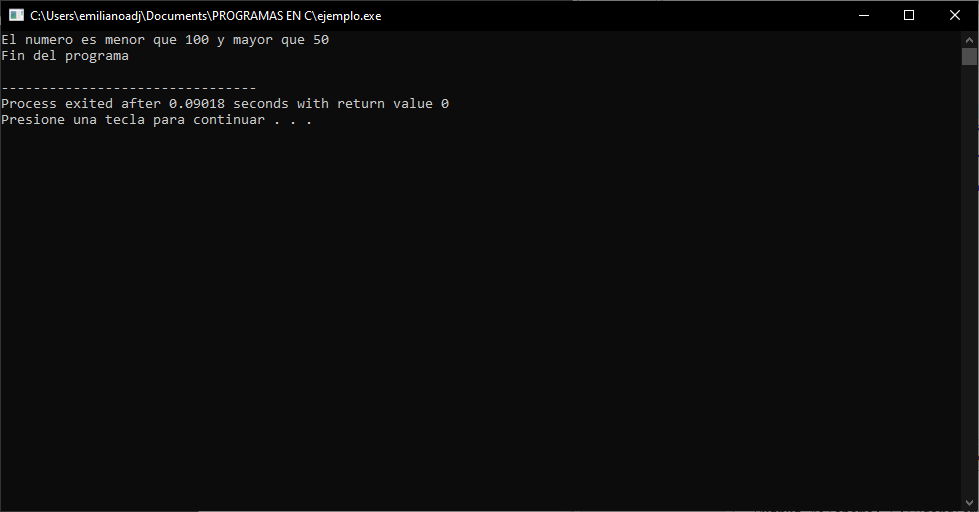
La calificación que se ingreso fue de 9, por lo cual deberá mostrar:

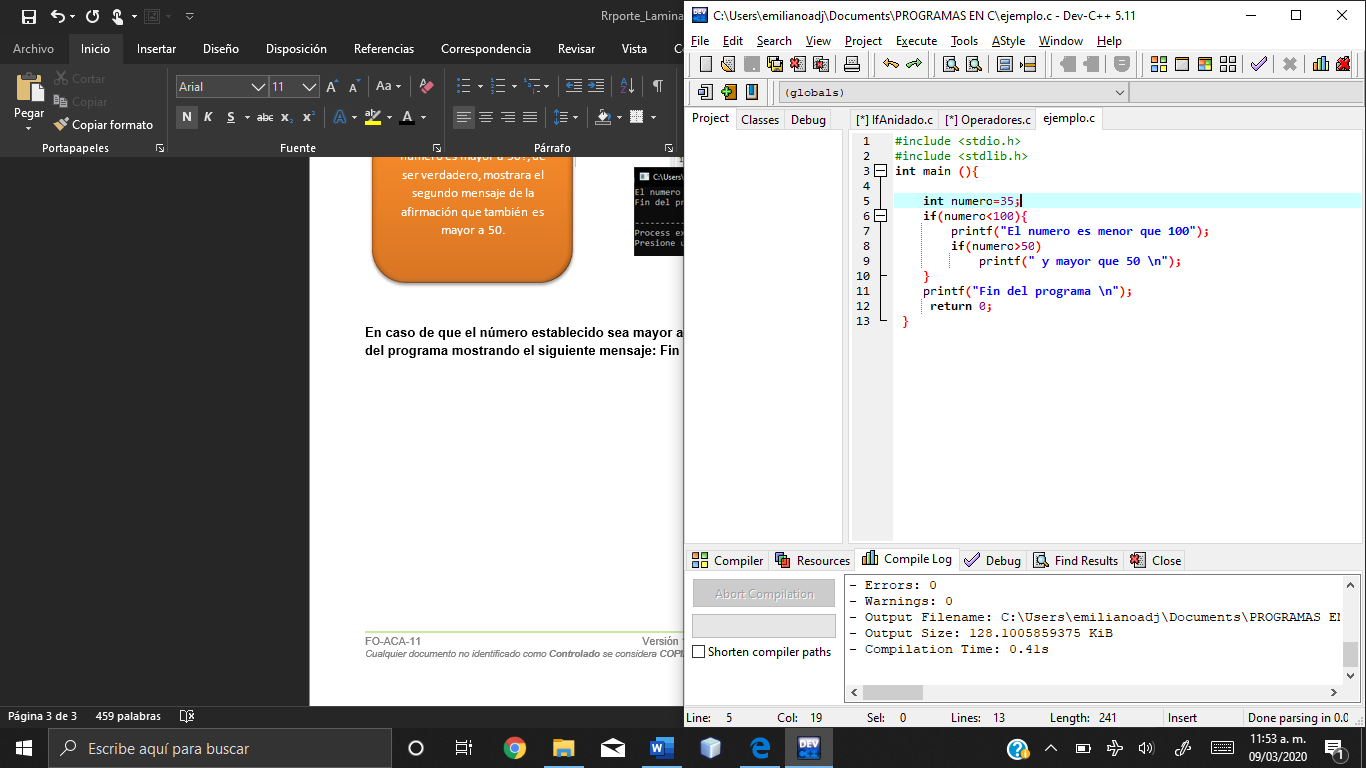
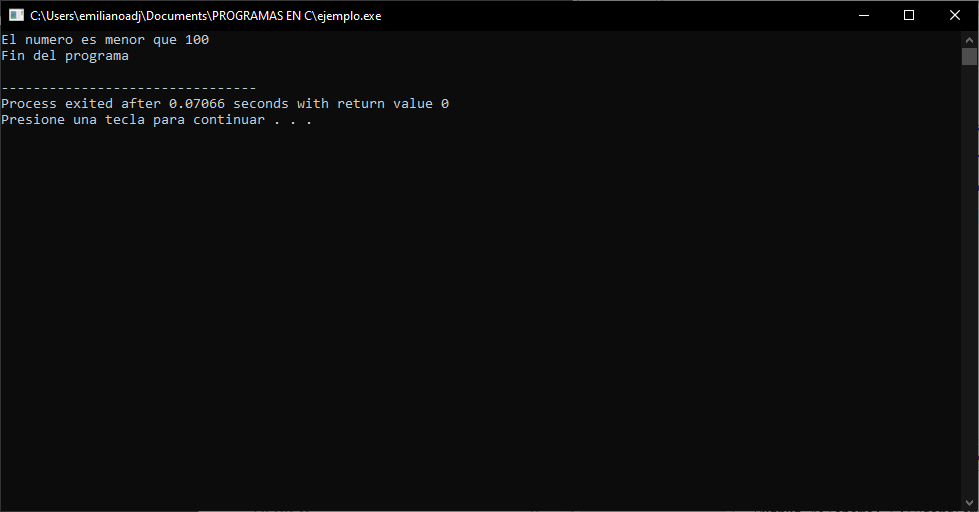
¡Felicitaciones aprobaste el curso!

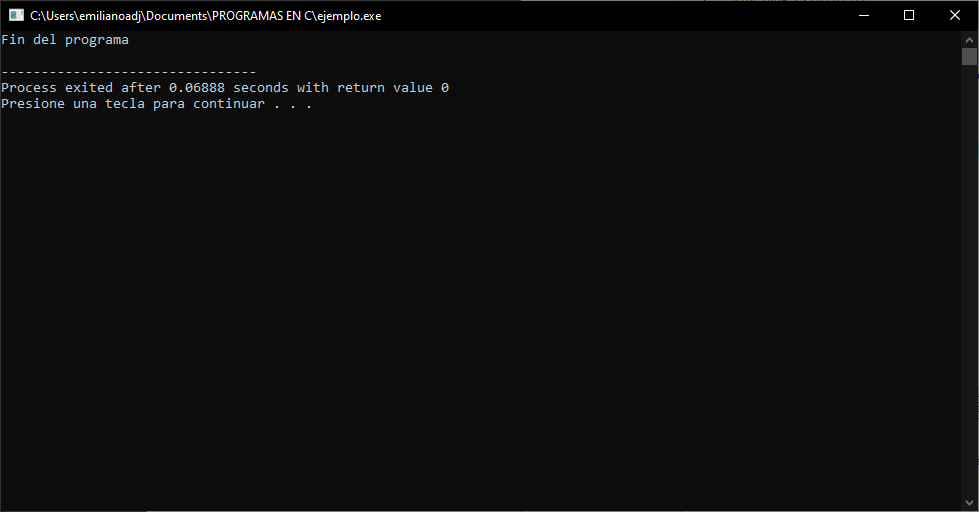
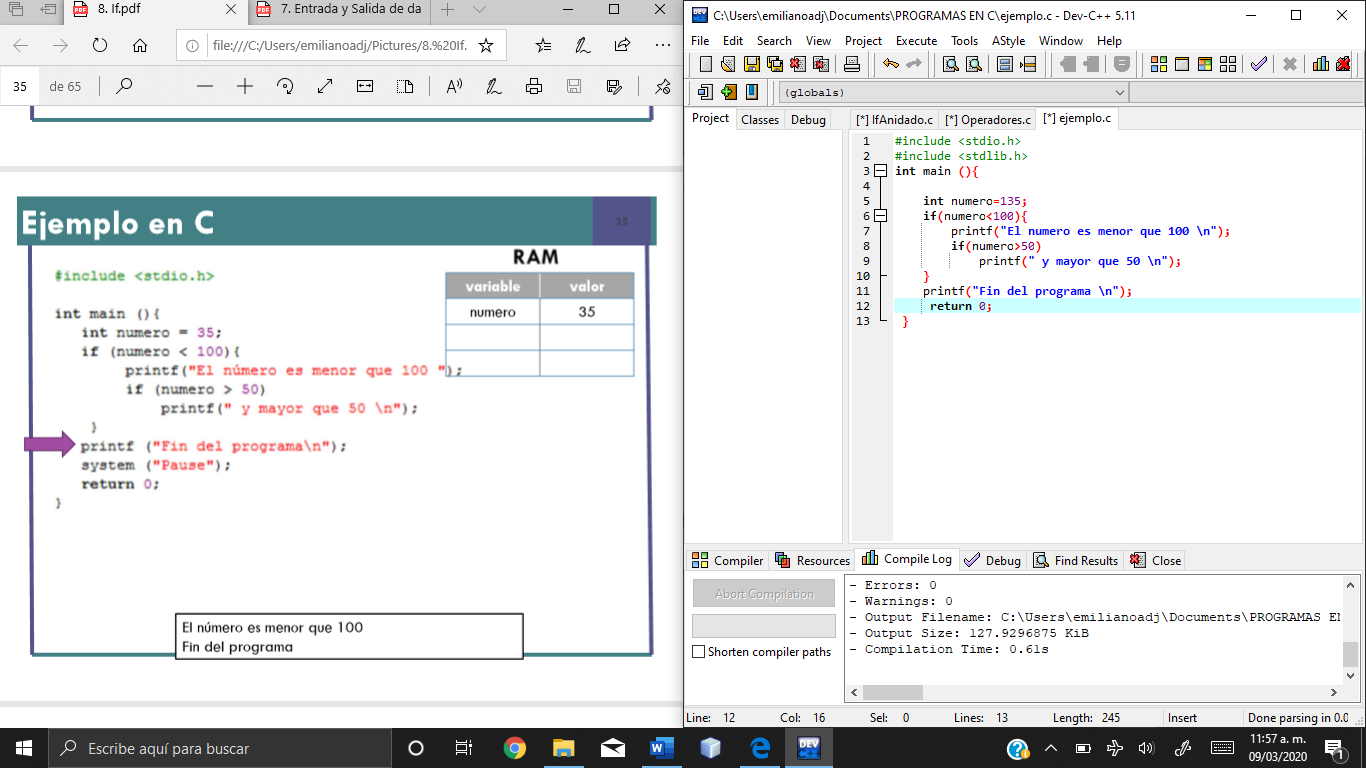
**Ahora vamos a analizar algunas estructuras de decisión simple anidadas, es decir, que existen estructuras dentro de otras estructuras.**

* **En el siguiente programa vamos a observar que dentro de una estructura condicional if, se encuentra otra estructura de este tipo.**

Para comenzar, tenemos establecida una variable con un valor de 55, entra en verificación: si el número es mayor a 100 mandara un mensaje que el número es mayor a 100, posteriormente entra en otra condición: ¿el número es mayor a 50?, de ser verdadero, mostrara el segundo mensaje de la afirmación que también es mayor a 50.



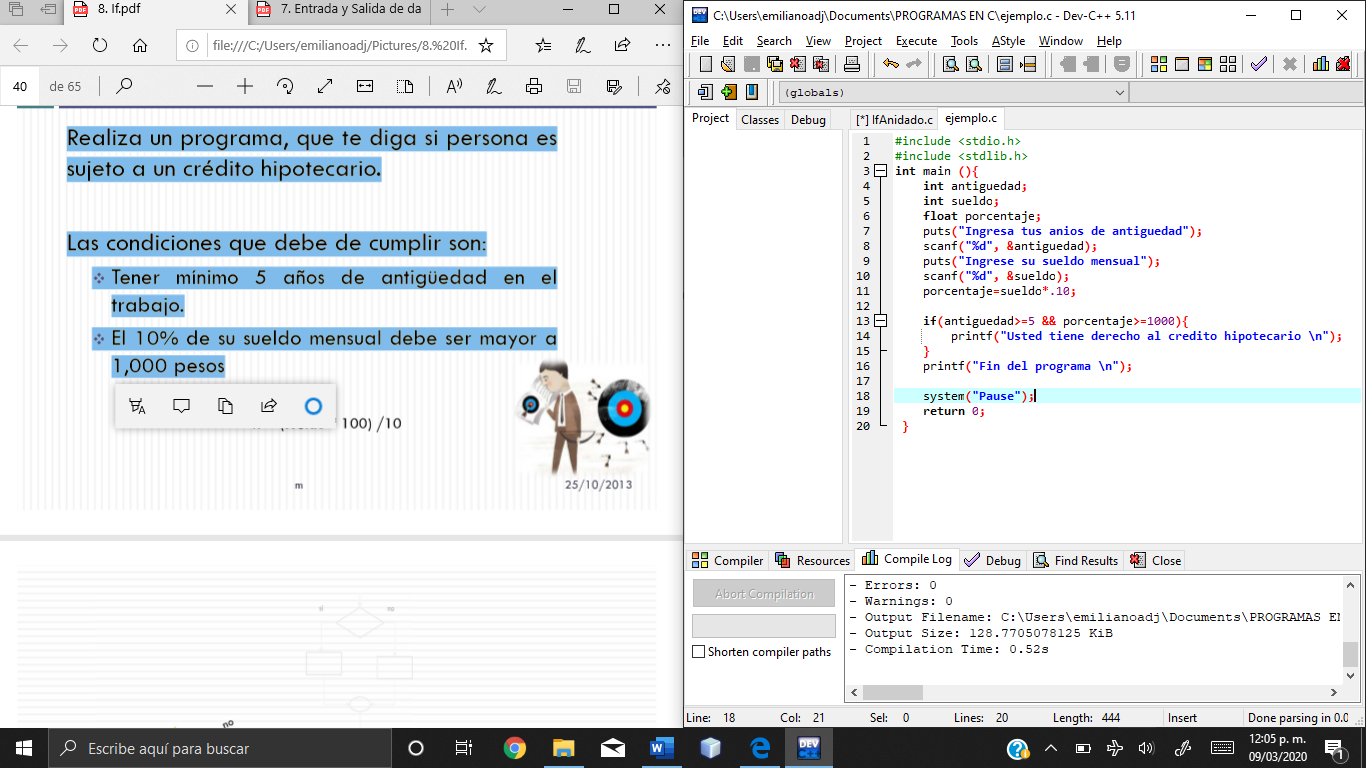
**Veamos que pasaría si el numero establecido es menor a 100, démosle un valor de 35.**

**¿Qué sucedería si el número es mayor a 100?**

* **Realiza un programa, que te diga si persona es sujeto a un crédito hipotecario.**

**Las condiciones que debe de cumplir son:**

**Tener mínimo 5 años de antigüedad en el trabajo.**

 **El 10% de su sueldo mensual debe ser mayor a 1,000pesos.**

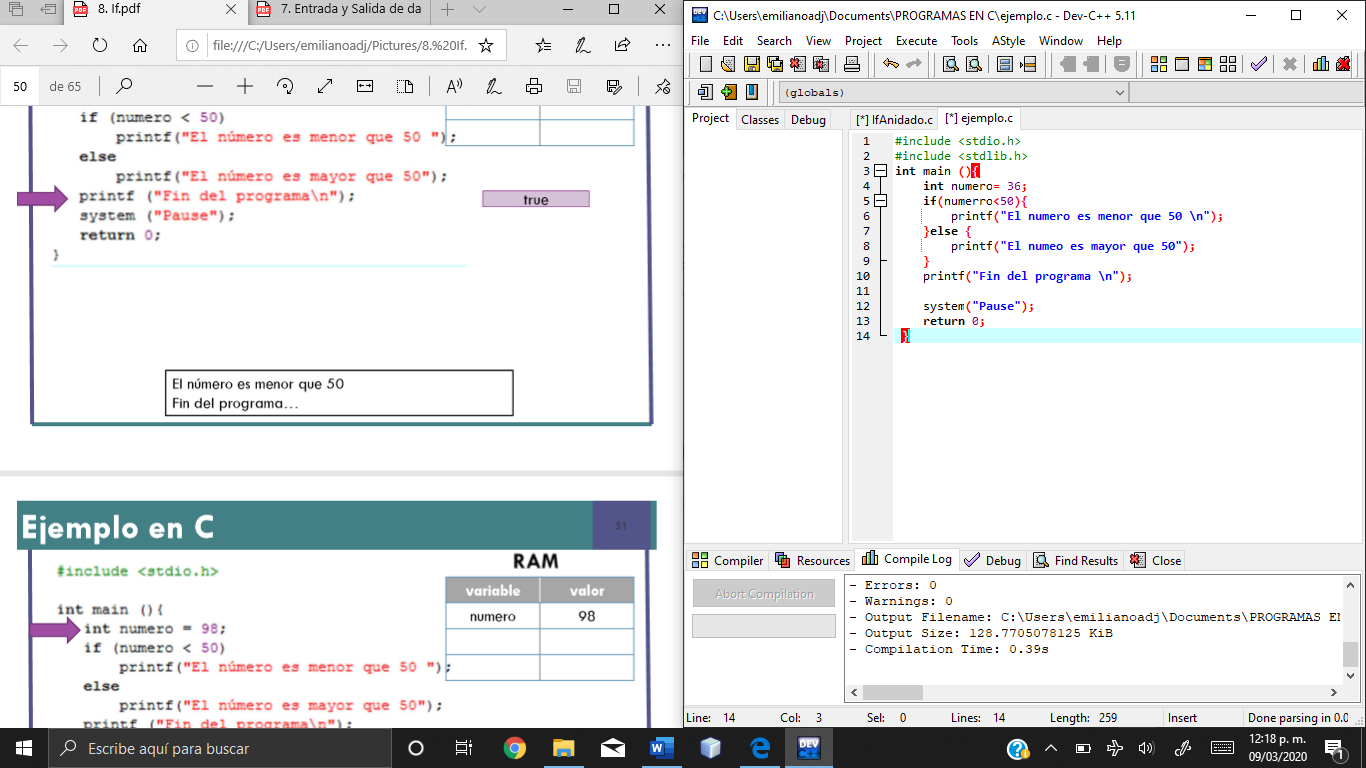
**Este programa lee datos desde el teclado, los datos requeridos son años de antigüedad y sueldo mensual, después se realiza una condición la cual va a contener dos comparaciones, la primera consiste en verificar si los años de antigüedad son mayores a 5 (línea 13), en caso de que esta primera parte de la condición sea falsa, el programa terminara, ya que el ejercicio pide que se cumplan las dos condiciones. La segunda comparación (línea 13) verifica si el 10% del salario mensual es mayor a 1000 (esto se obtiene multiplicando el salario mensual por .10 (línea 11)).**

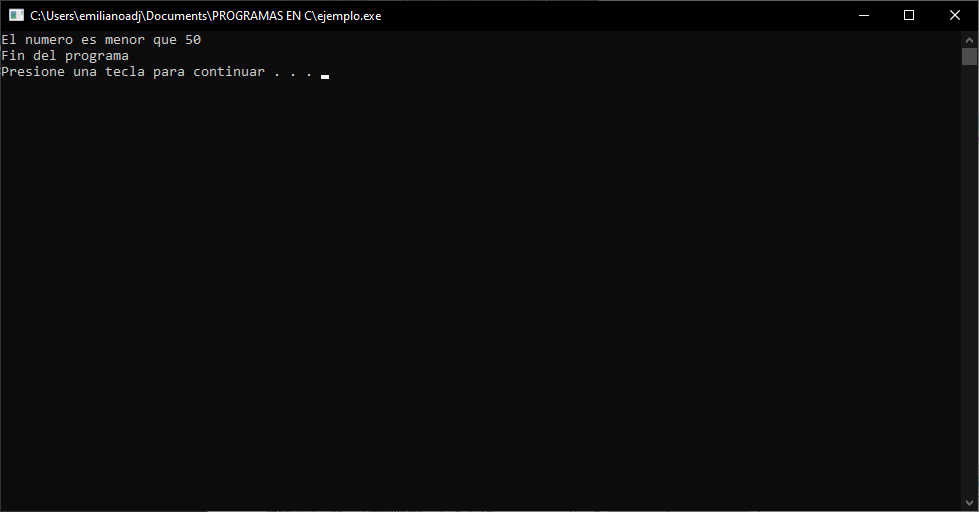
**En caso de que las dos condiciones se cumplan mostrara el siguiente mensaje:**

**“Usted tiene derecho al crédito hipotecario”. Si no, “Fin del programa”.**

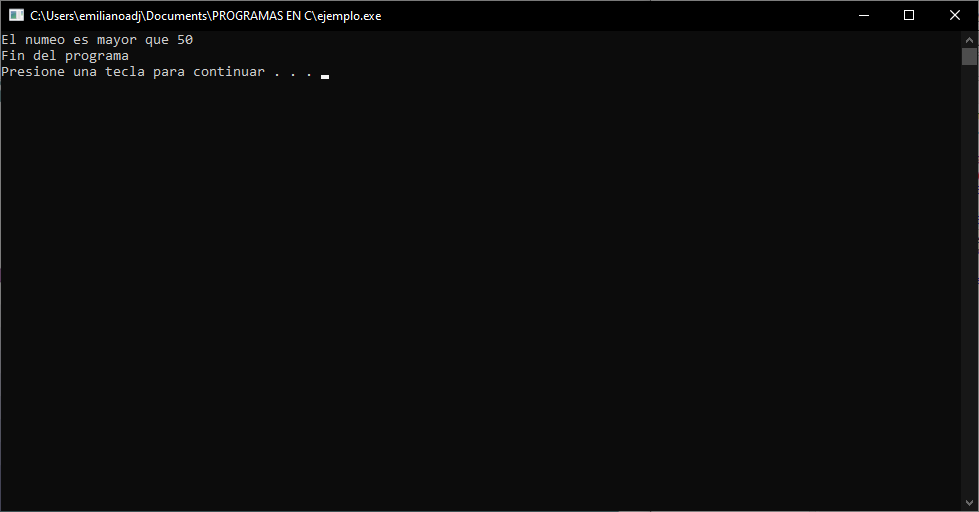
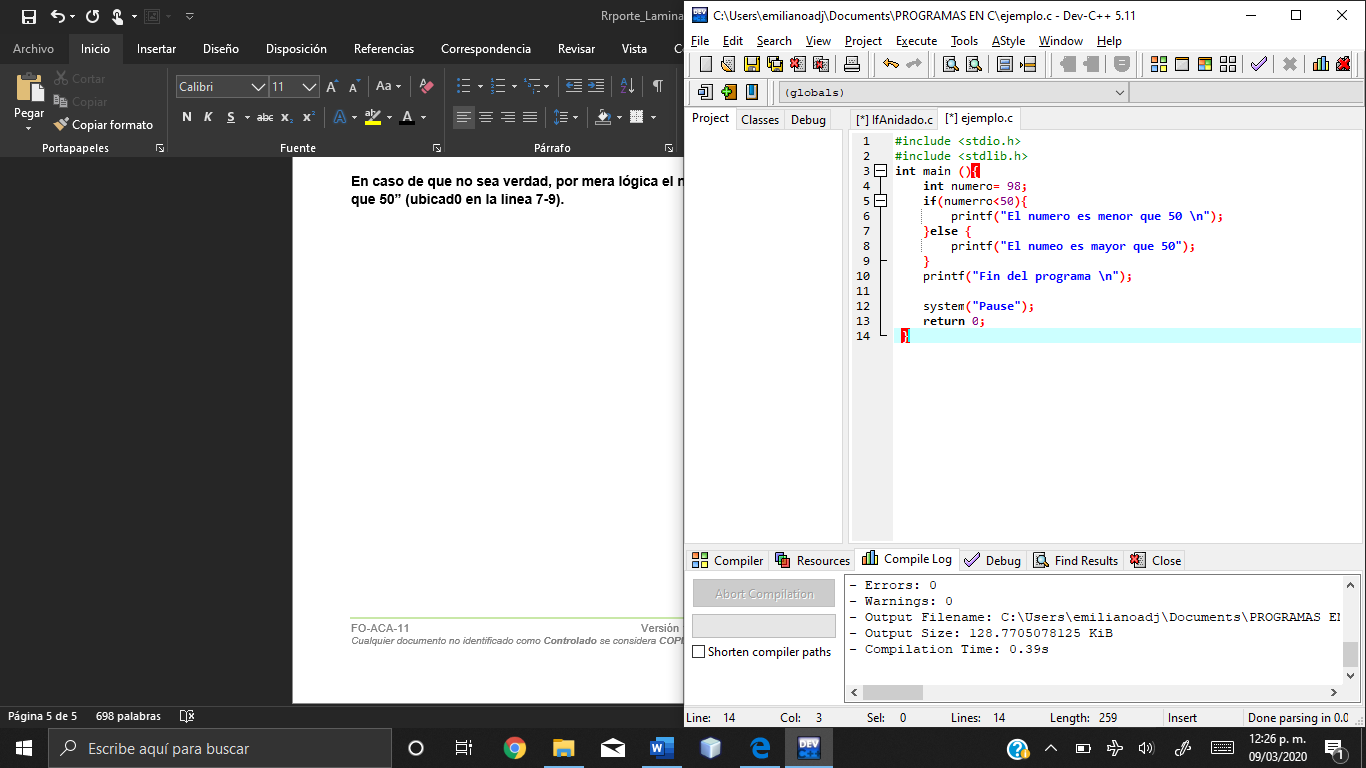
**if-else.**

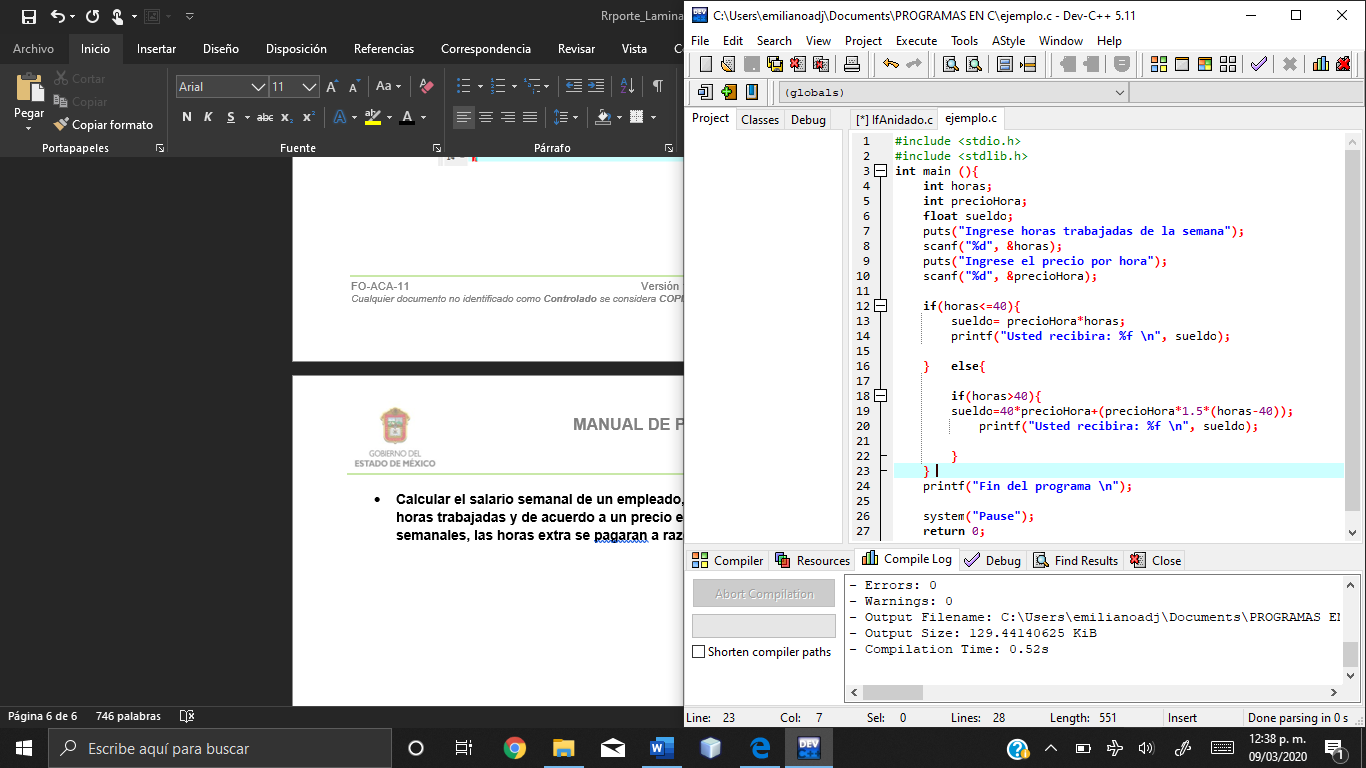
**Consiste en que, si la primera instrucción es falsa, pasara a ejecutar otro bloque de código encontrado en un else.**

* **Este codigo verifica si un numero establecido (linea 4) es menor a 50, de ser verdad mostrara el siguiente mensaje: “El número es menor que 50”.**



* **En caso de que no sea verdad, por mera lógica el numero debe ser mayor: “El número es mayor que 50” (ubicad0 en la linea 7-9).**



* **Calcular el salario semanal de un empleado, sabiendo que éste se calcula en base de las horas trabajadas y de acuerdo a un precio especificado por hora. Si se pasa de las 40 horas semanales, las horas extra se pagarán a razón de 1.5 veces la hora ordinaria.**

Se piden al usuario los siguientes datos desde teclado: horas trabajadas por semana, precio por hora trabajada.

Con los datos anteriores, se realiza primeramente una condición: las horas trabajadas son menores o iguales a 40.

De ser así, multiplicara las horas por el precio de cada una de esta y ese seria su sueldo semanal (línea 12-15).

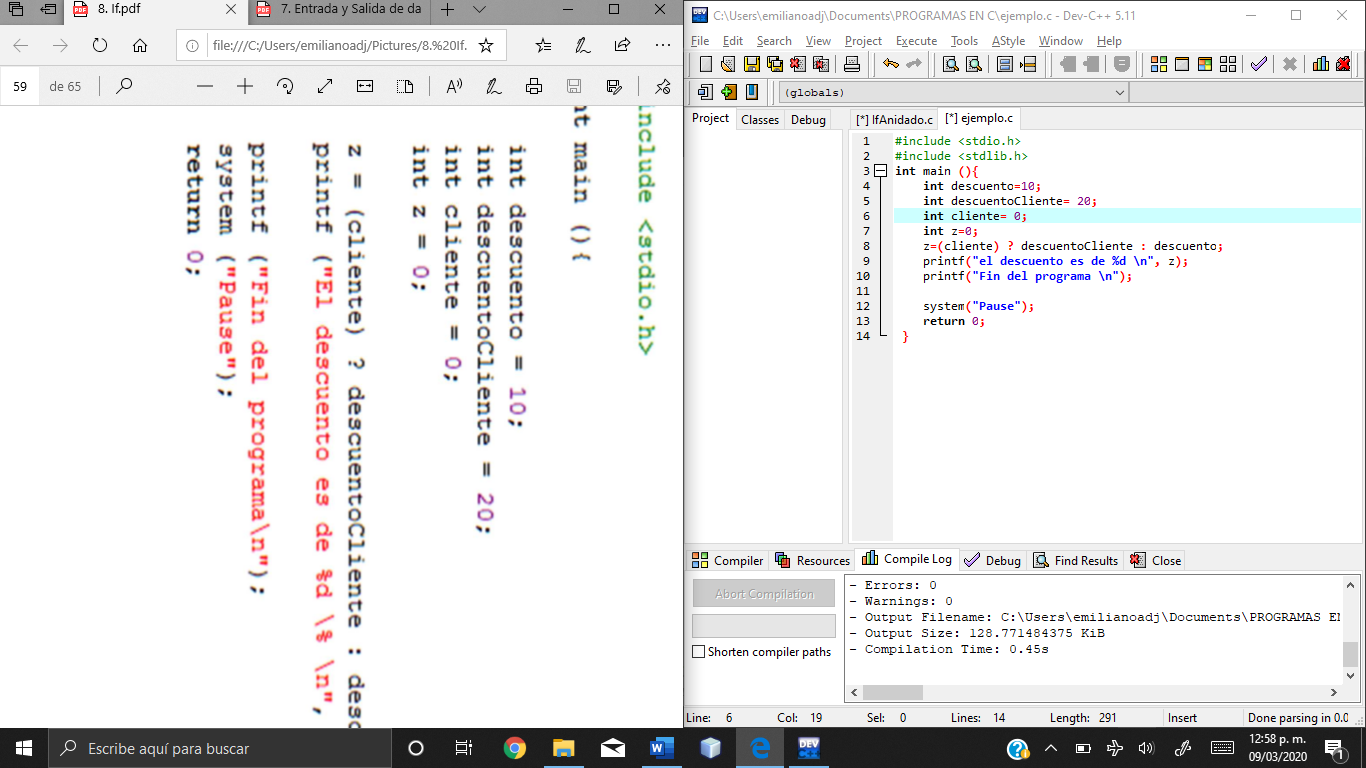
Si las horas trabajadas exceden las 40, entonces su sueldo se obtendrá de la siguiente manera:

40\*precioHora+(precioHora\*1.5\*(horas-40))

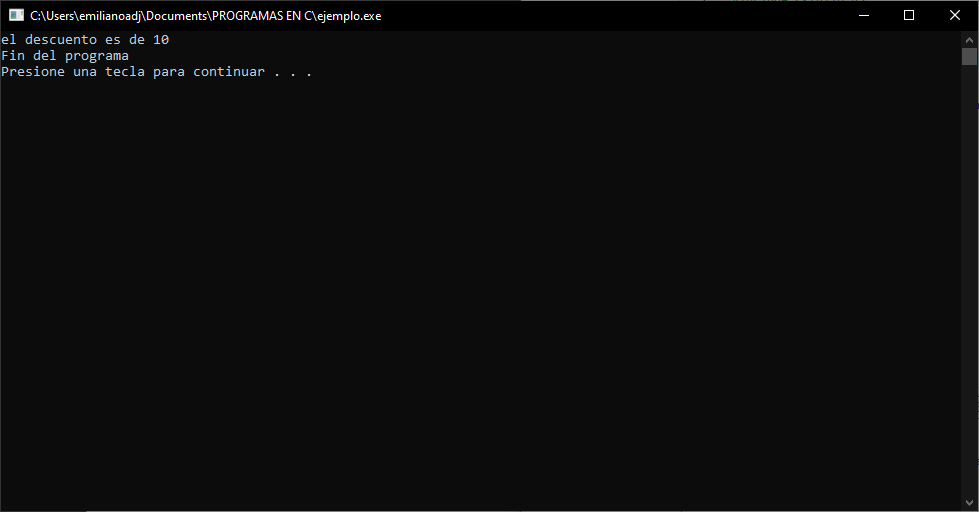
(línea 16-20).

Y ese sería su sueldo semanal.

* **Operador condicional.**

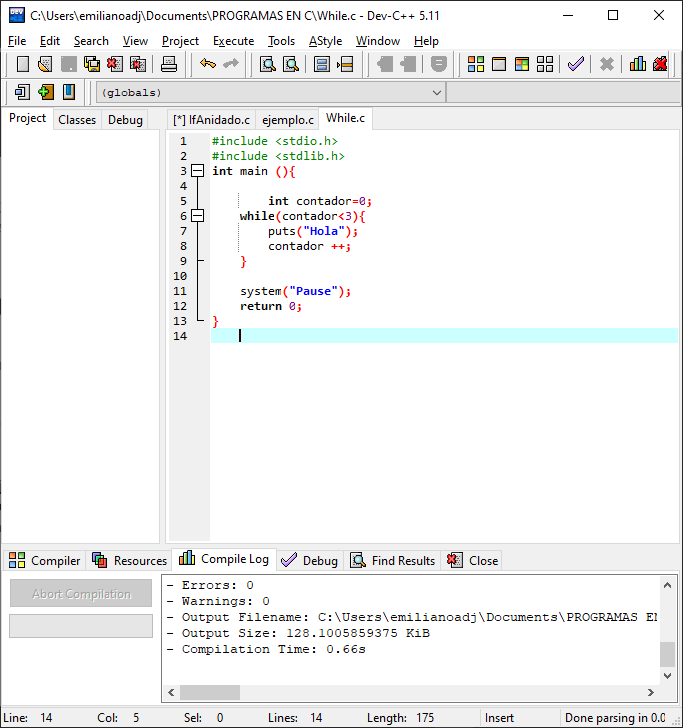
 **Sintaxis:exp1 ? exp2 : exp3**

**Si la exp1 evalúa a cierto, la expresión completa evalúa al valor de exp2.**

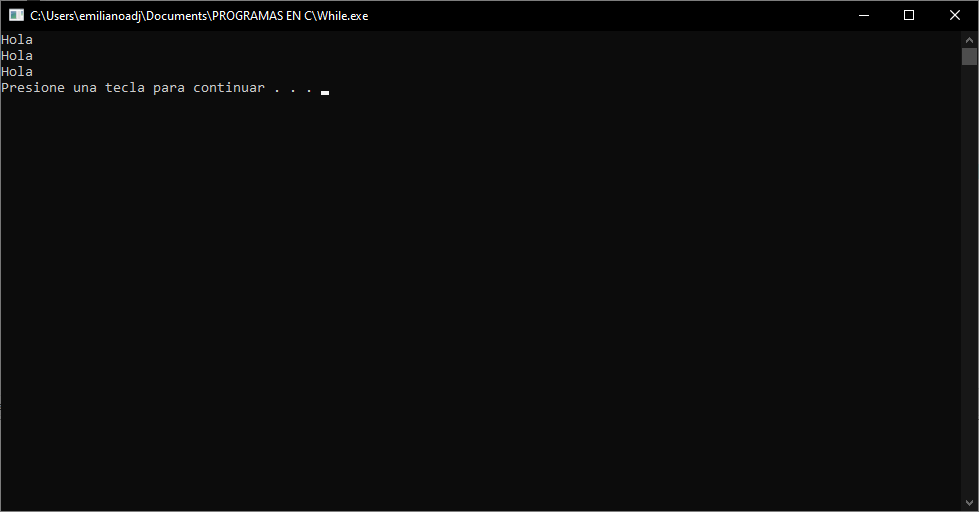
**Si la exp1 evalúa a falso, la expresión completa evalúa al valor de exp3**

**Estructura repetitiva while.**

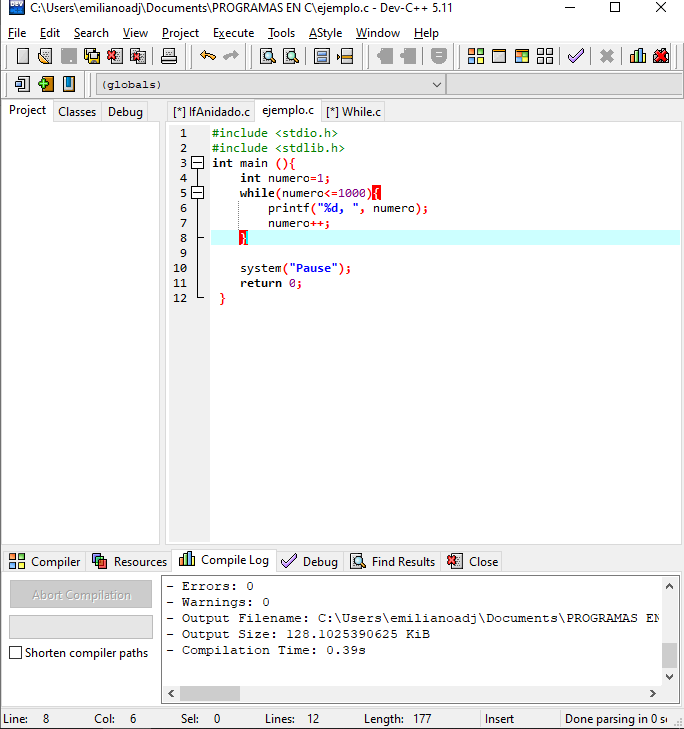
**Una estructura repetitiva while Permite que un conjunto de sentencias pueda ser ejecutadas repetidamente según el estado de una expresión lógica (condición), es decir, si es verdadero ejecuta una o varias veces la instrucción y nuevamente verifica la condición, si es falsa, entonces el ciclo termina.**

* **Ejemplo**

**Mientras contador que fue inicializado en 0 sea menor a 3, va a imprimir “hola”.**

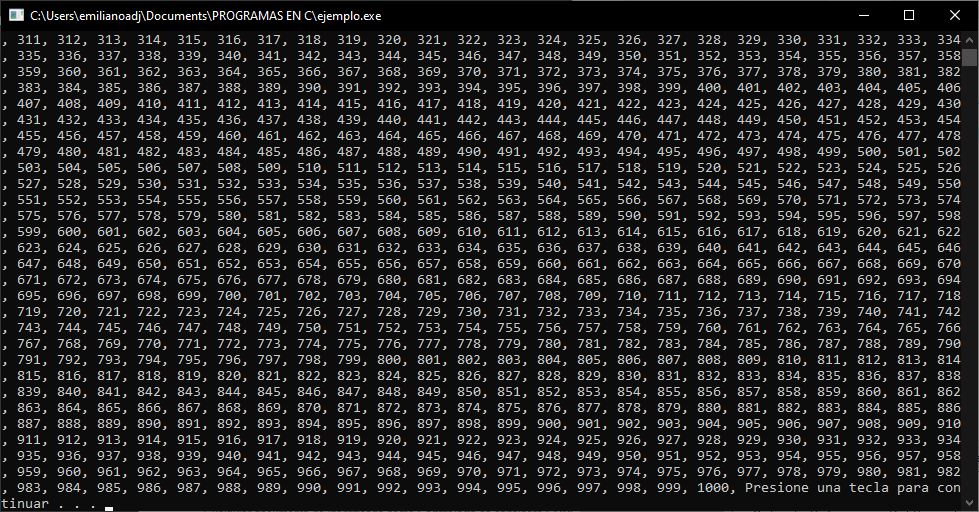
**Después de la primera impresión, evaluará al contador hasta que este sea menor a 3.**

* **El siguiente código muestra un ejemplo de como funciona in ciclo9 while.**

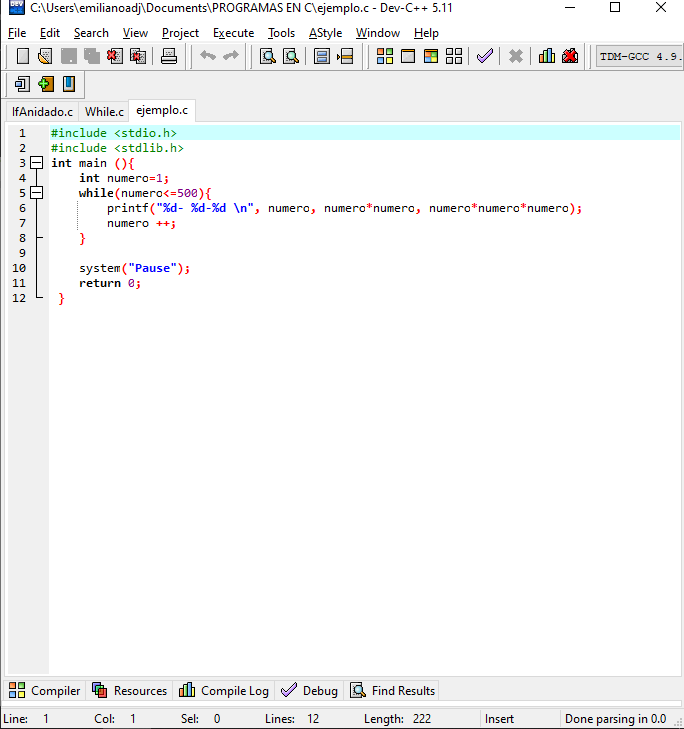
**Quizá sea muy parecido al for, pero a diferencia que este es un tanto más útil en ciertas ocasiones.**

**El codigo indica que mientras numero (que es un valor ya establecido) sea menor o igual a 1000, es decir, que aunque sea menor a mil o igual a mil, va a estar imprimiendo los numeros en serie incrementando en uno cada vez que imprima un número.**

**1, 2, 3,…,1000.**



* **Imprimir los cuadrados y los cubos de los primeros quinientos números naturales.**



**Un numero elevado al cuadrado es la multiplicación de si mismo dos veces:**

**32 =3\*3=9**

**Un numero elevado al cubo es multiplicado por si mismo 3 veces:**

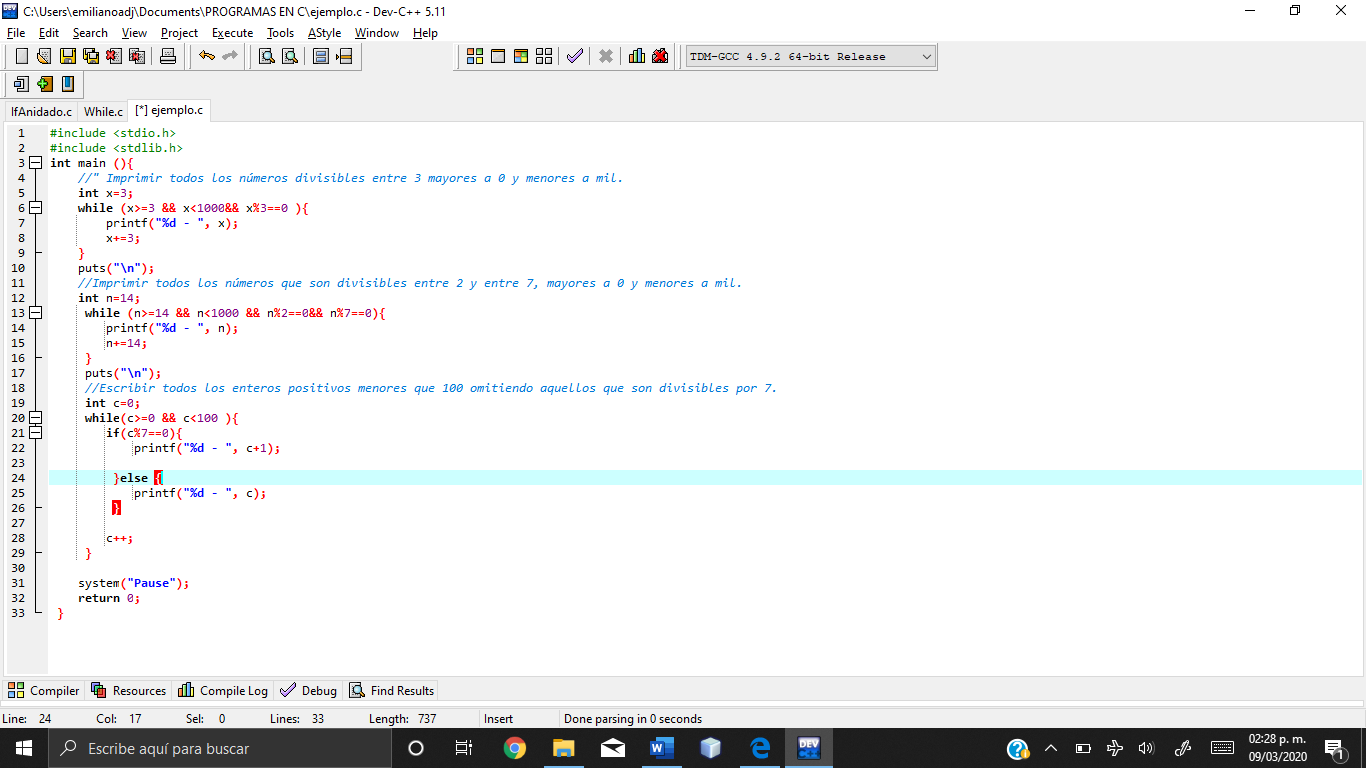
**33 =3\*3\*3=27**

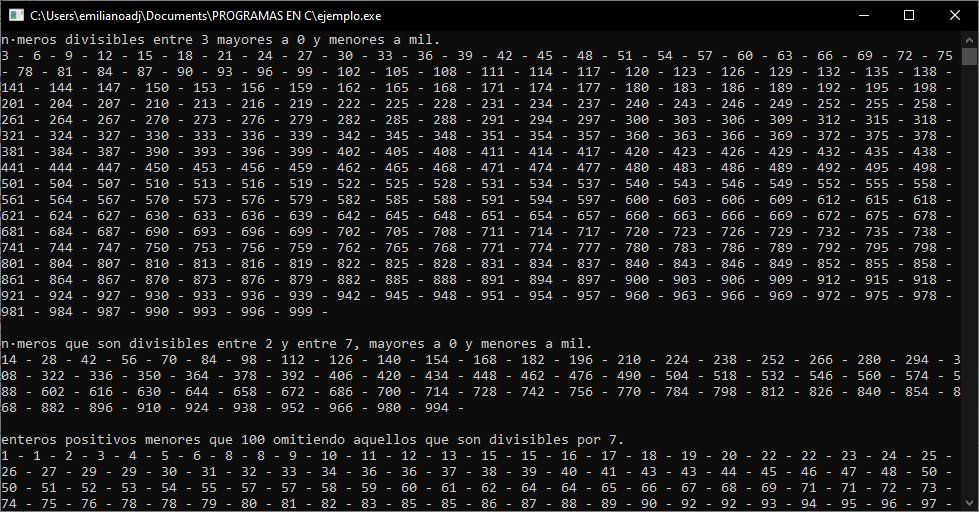


* **Imprimir todos los números divisibles entre 3 mayores a 0 y menores a mil.**

**Imprimir todos los números que son divisibles entre 2 y entre 7, mayores a 0 y menores a mil.**

**Escribir todos los enteros positivos menores que 100 omitiendo aquellos que son divisibles por 7.**

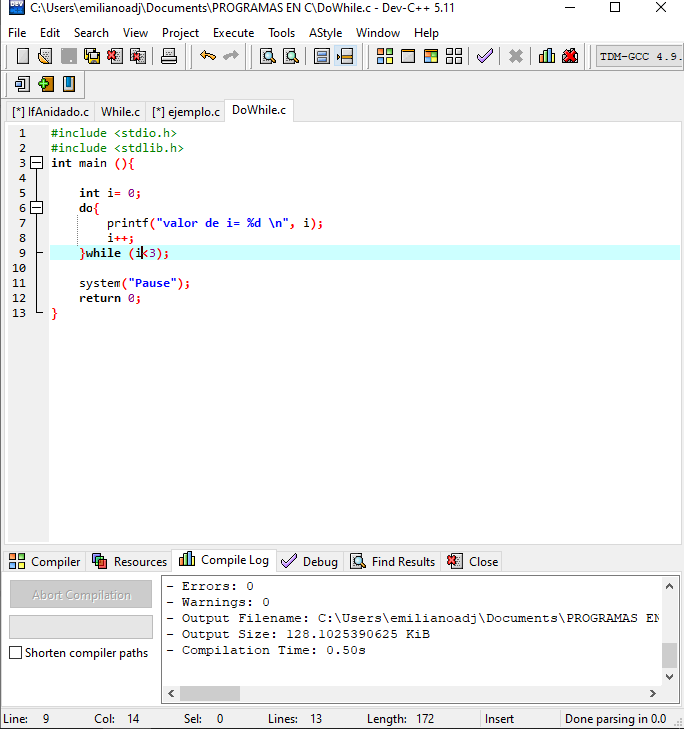




**El programa tiene dentro un algoritmo que realiza ciertas instrucciones para determinar que números debe imprimir y que números no cumplen las condiciones especificadas.**

**Estructura do-while.**

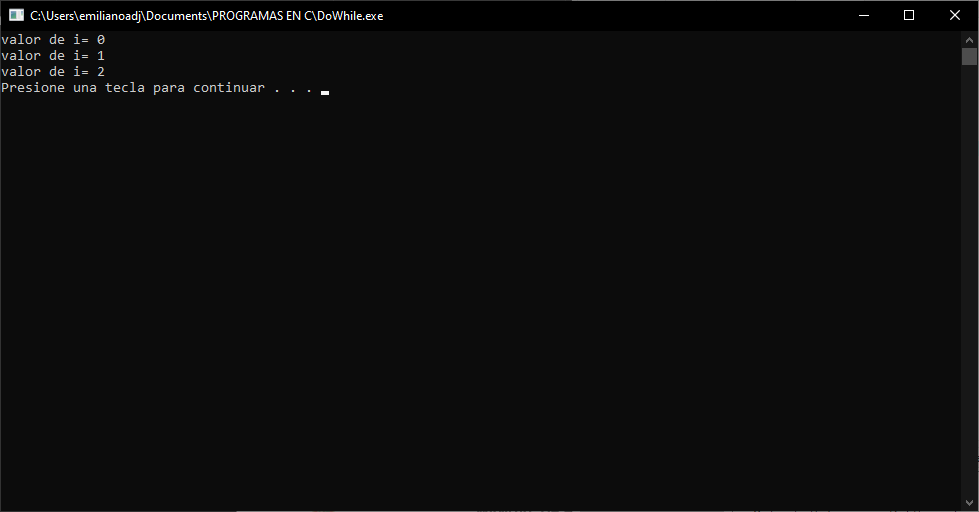
**A diferencia del while¸ esta estructura primero ejecuta el conjunto de instrucciones y después verifica que la condición se cumpla.**

* **Ejemplo.**

**Hacer (do):**

**imprimir “valor de i= x”. incrementa i en uno. Mientras (while)=**

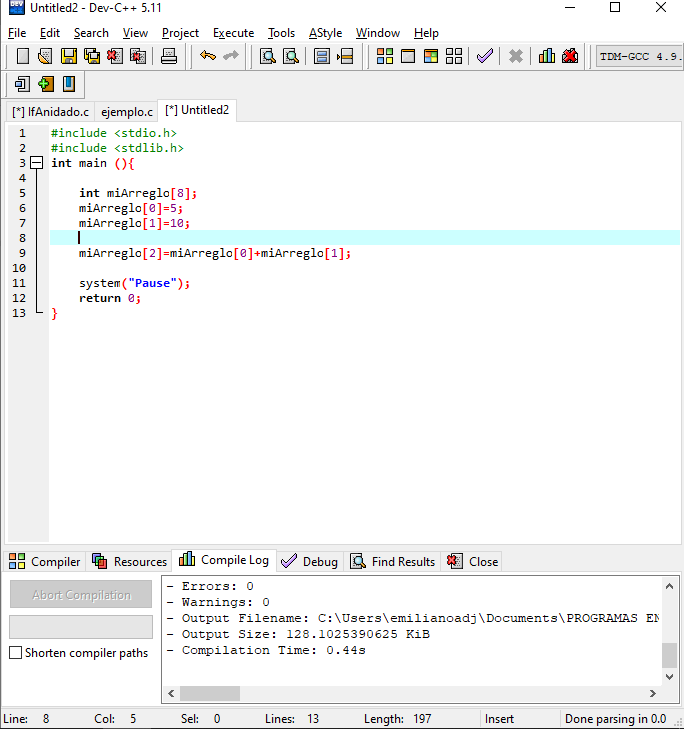
**i sea menor a 3.**

**Este sería el resultado del programa do-while anterior.**

**ARREGLOS**

**¿Qué es un arreglo? Un arreglo es una variable que hace referencia a varias posiciones de memoria, cada una de estas posiciones tendrá un índice y el primer índice debe comenzar en 0 hasta n posiciones.  
la forma de declarar un arreglo es la siguiente:**

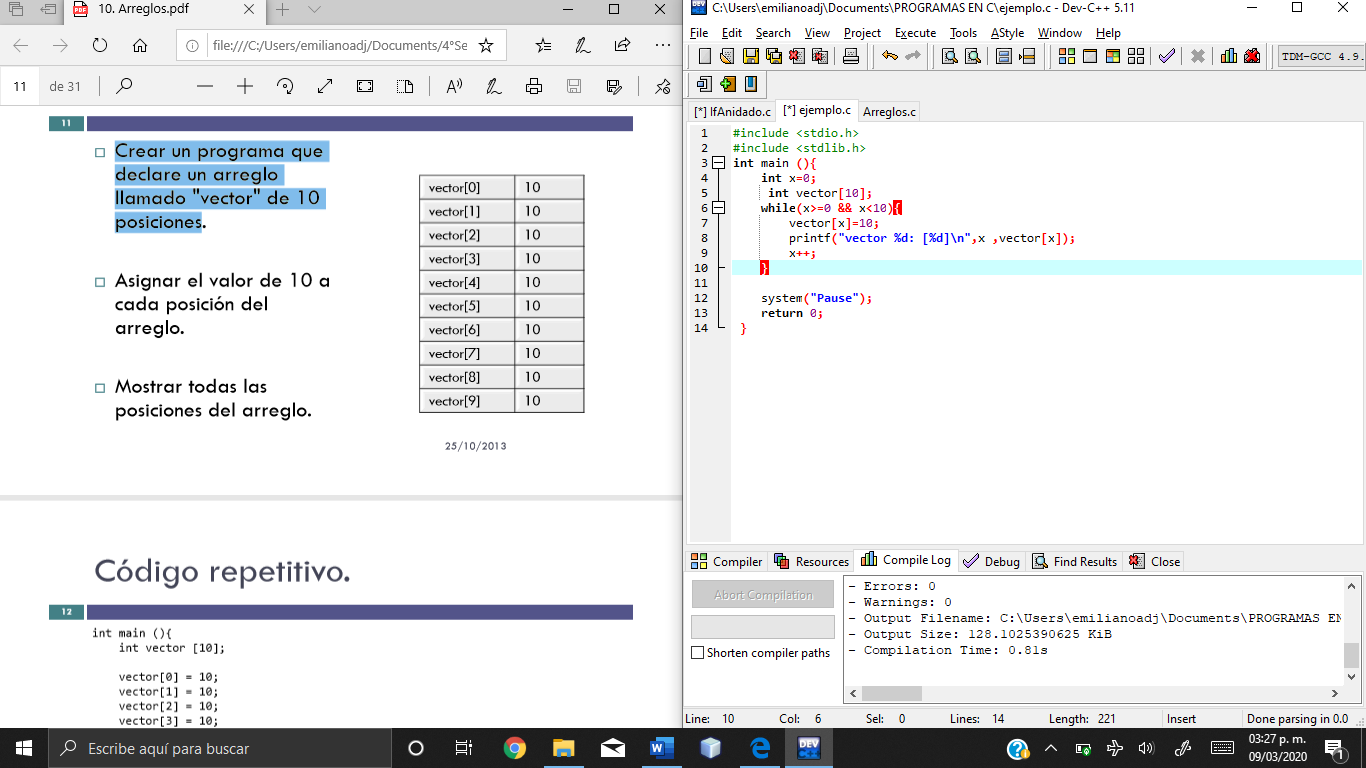
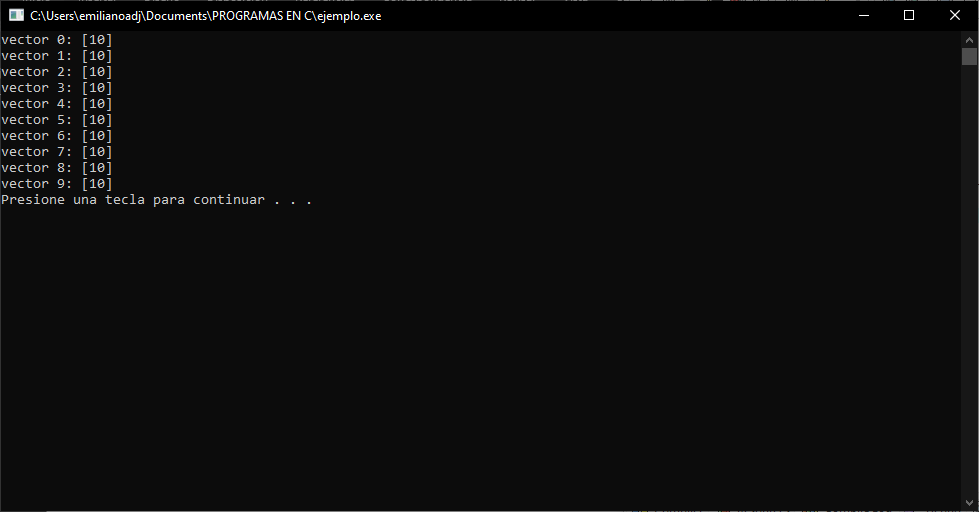
**int miArreglo [ 8];**

**Tipo de dato, nombre del arreglo, corchetes para indicar que es un arreglo, tamaño del arreglo (el número 8 se asigna a la memoria en 8 posiciones).**

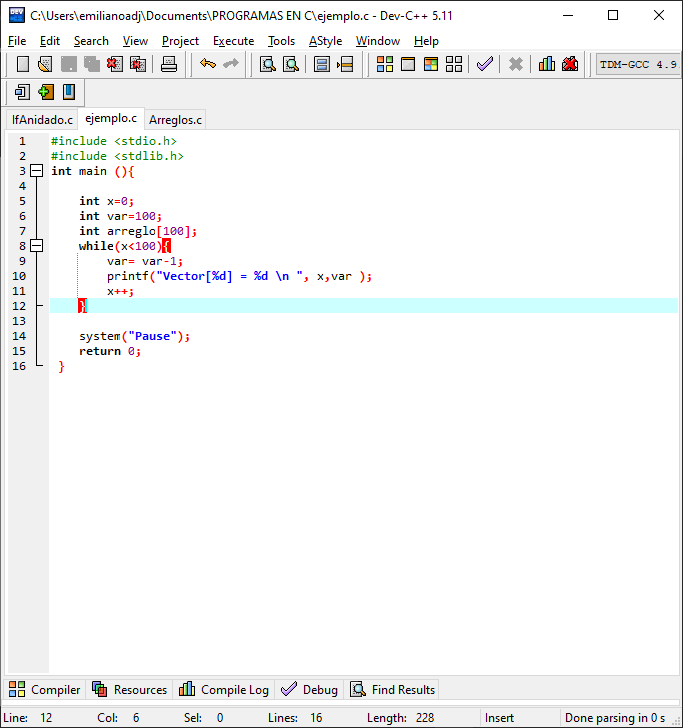
El código anterior muestra como a cada posición del arreglo (tamaño 8 (línea 5)) se le asigna un valor numérico entero (línea 6-7).

También se pueden realizar operaciones con arreglos, esto haciendo las mismas con el elemento que hay en cada posición del arreglo (línea 9).

* **Crear un programa que declare un arreglo llamado "vector" de 10 posiciones, asignar el valor de 10 a cada posición del arreglo y mostrar todas las posiciones del arreglo.**

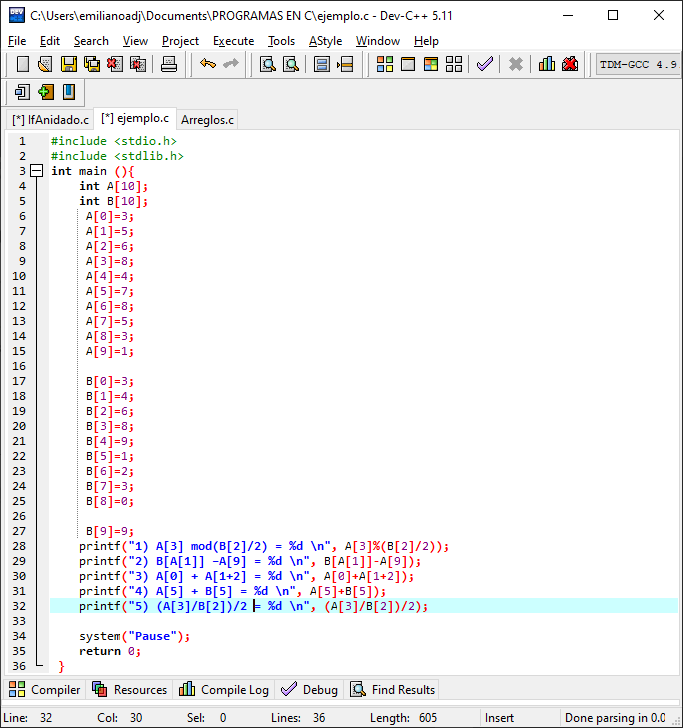


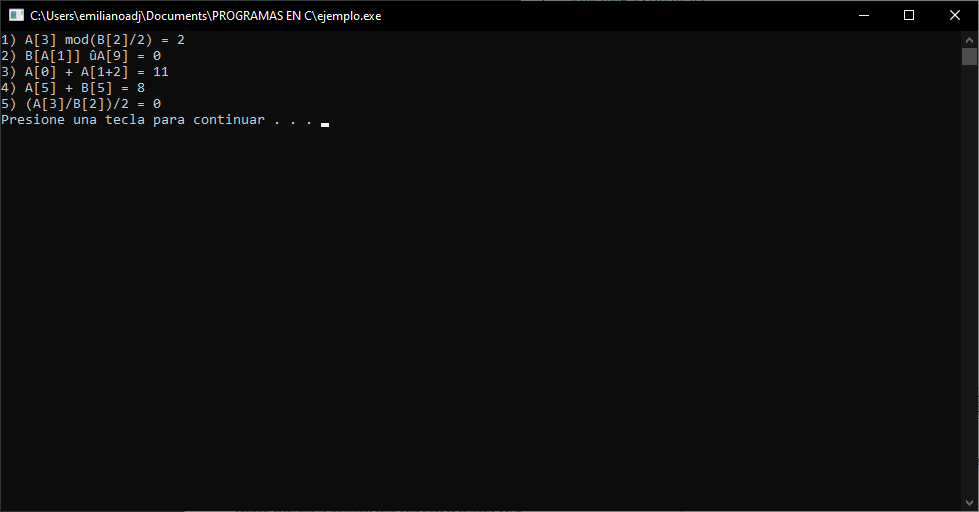
* **Crear un arreglo de 100 posiciones, llenar el orden inverso al índice y mostrar el arreglo en pantalla.**



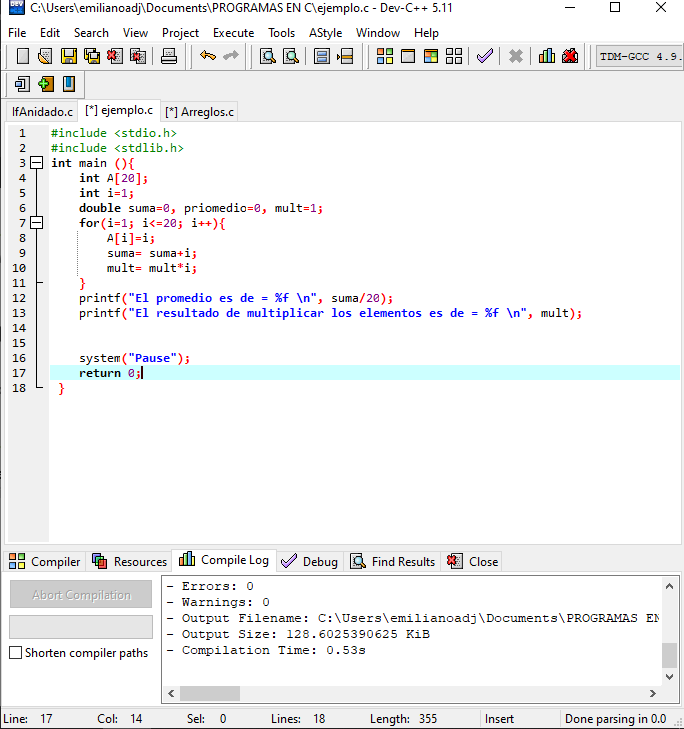
**Este ejercicio es similar al anterior con la diferencia que los llenados no se hacen como normalmente lo conocemos, si no que este lo realiza de forma inversa.**

* Dados A = [3, 5, 6, 8, 4, 7, 8, 5, 3, 1] y B = [3, 4, 6, 8, 9, 1, 2, 3, 0, 9], realiza las siguientes operaciones:

1. A[3] mod(B[2]/2)
2. B[A[1]] –A[9]
3. A[0] + A[1+2]
4. A[5] + B[5]
5. (A[3]/B[2])/2

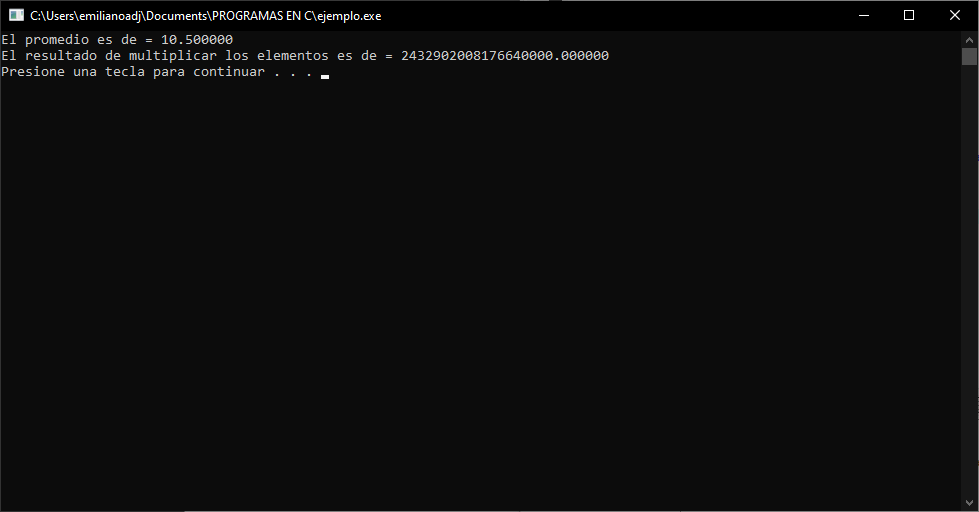


* **Creaunarreglode20posiciones, asígnale a cada elemento un valor, calcula el promedio de todos los elementos y calcula la multiplicación de todos los elementos.**



**Declaramos dos contadores, uno que se llama suma (es el que va a ir sumando cada número del uno al 20) y otro de nombre mult (ira multiplicando cada elemento: 1\*2\*3…).**

**Al final solo dividimos el contador suma entre 20 que es el tamaño del arreglo e imprimimos este resultado junto con el de la multiplicación.**



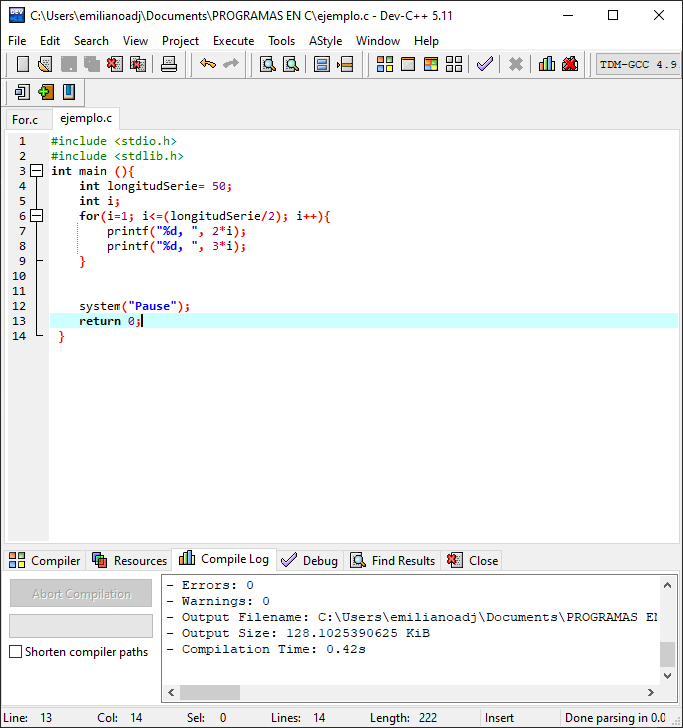
**ESTRUCTURA REPRTITIVA FOR**

**La estructura repetitiva for es similar a la del ciclo while.**

**Esta estructura es uy útil para recorrer arreglos, tiene 3 “tiempos”; la inicialización, la condición y el incremento.**

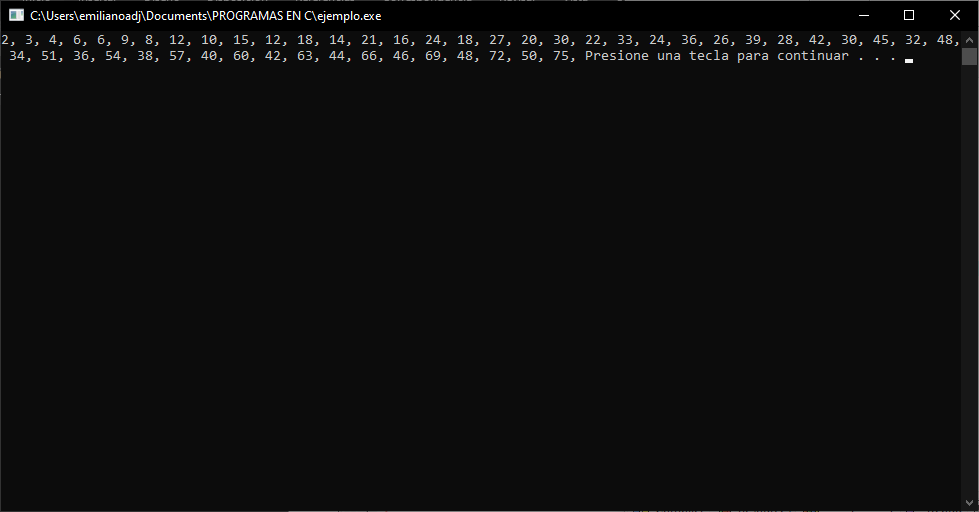
**Dentro de las llaves del ciclo for se encuentra el código que se repite.**

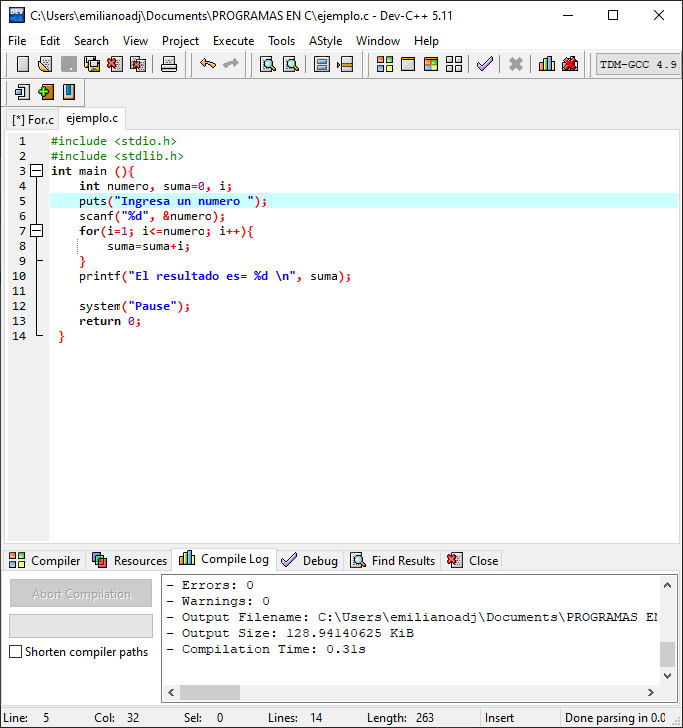
* **Ejemplo**

**se debe mostrar la serie: 2, 3, 4, 6, 6, 9, 8, 12, 10…**

**cada numero de la serie obtenida es una multiplicacion de las dos que se muestran en el código (linea 7-8).**

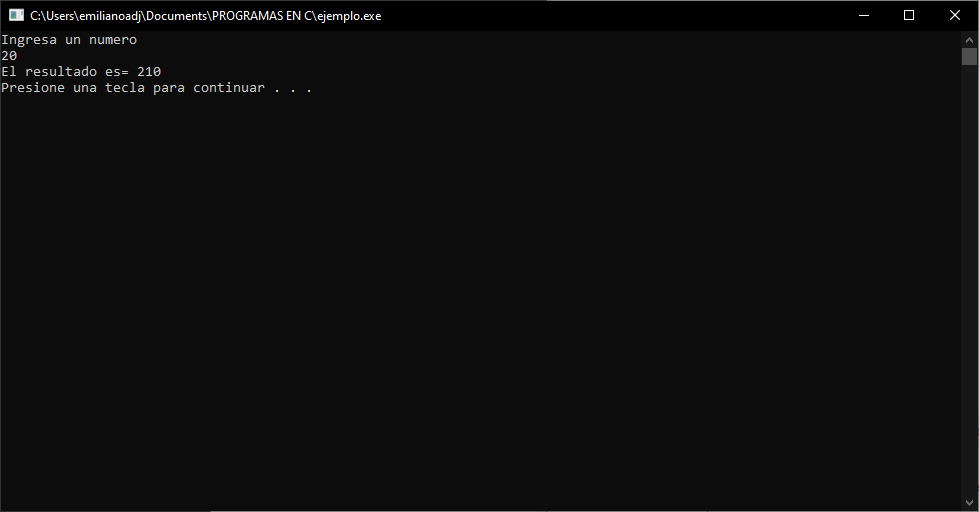
**Es decir, 2\*i=2 - 3\*i=3 – 2\*i=4 – 3\*i=6 - … etc.**

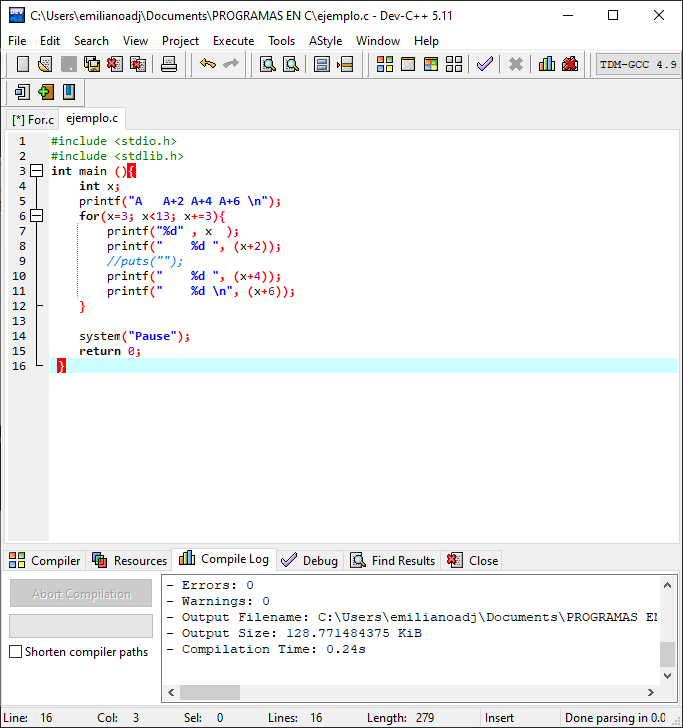


*  **Escribe un programa que reciba un número N del usuario y haga la suma de todos los números desde 1 hasta N.**

Lo que hace el programa, es básicamente ir sumando desde 0+1+2+ 3+…+n, que es establecido por el usuario.

Finalmente muestra el resultado en pantalla.



** Escriba un programa en C que utilice un ciclo para producir la siguiente tabla de valores**

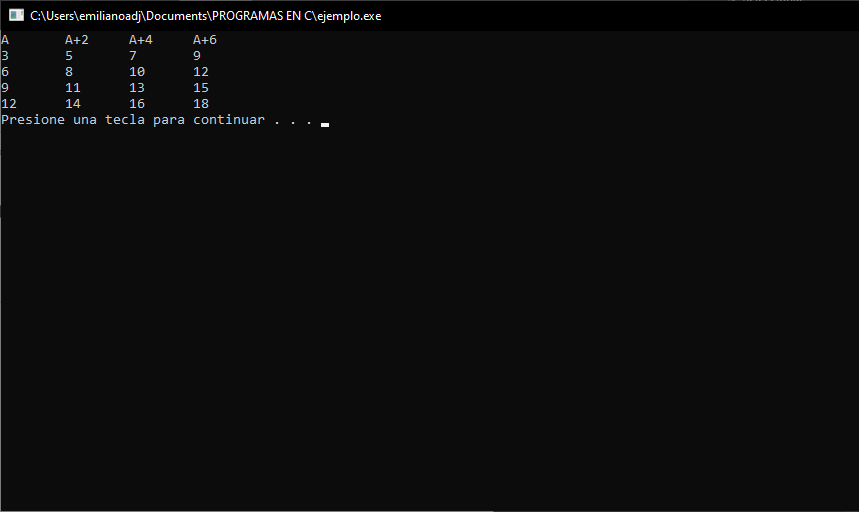
A A+2 A+4 A+6

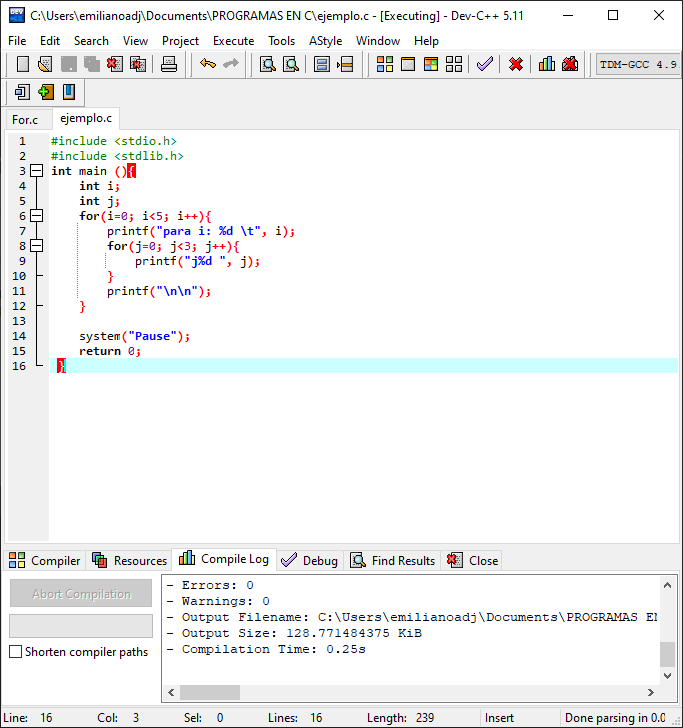
3 5 7 9

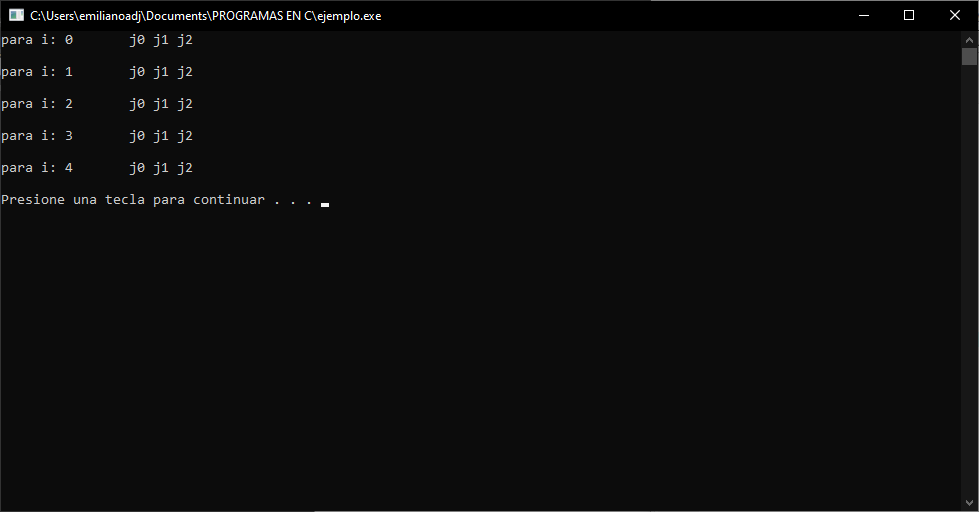
6 8 10 12

9 11 13 15

12 14 16 18

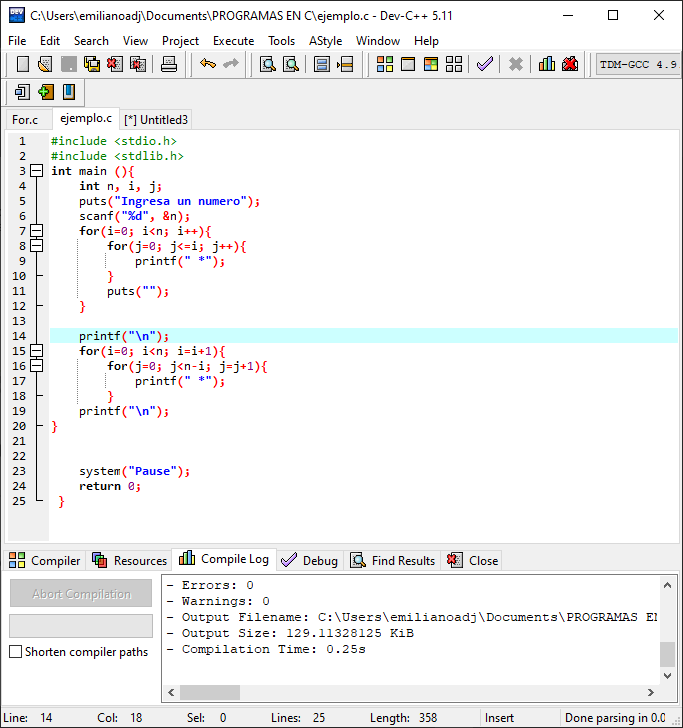
**utilizamos un ciclo for para la impresión de los resultados e hicmos unas operaciones para complementar las columnas siguientes…**

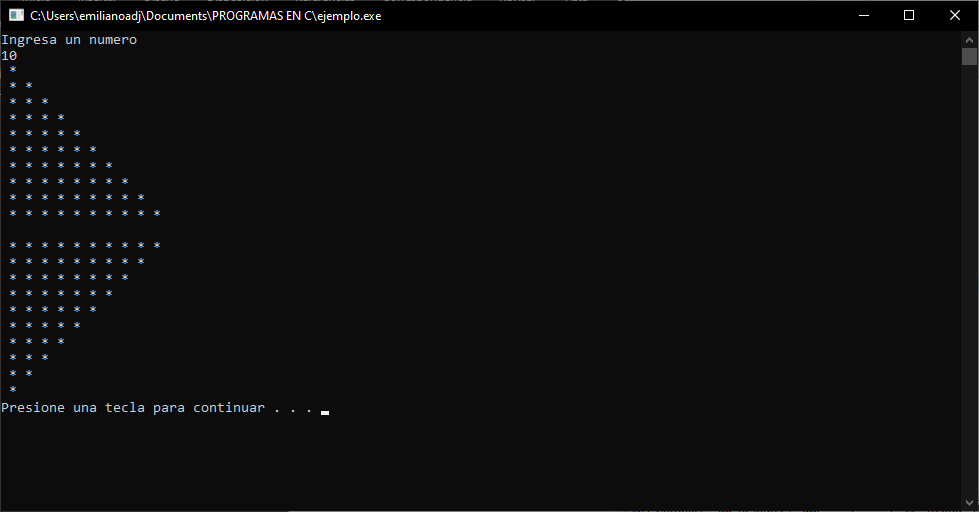
* **Ejemplo**



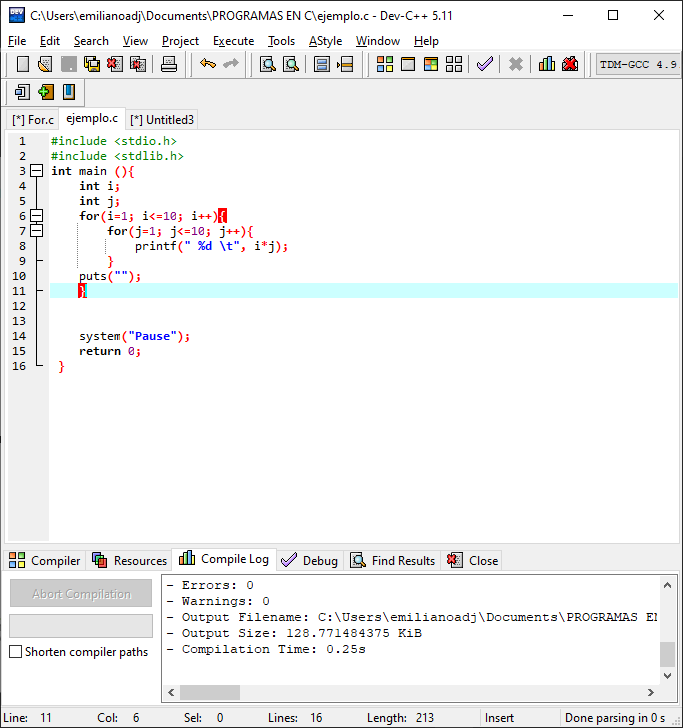
**Cada ciclo for contiene un bloque de instrucciones distinto lo cual está estructurado para su correcto funcionamiento.**

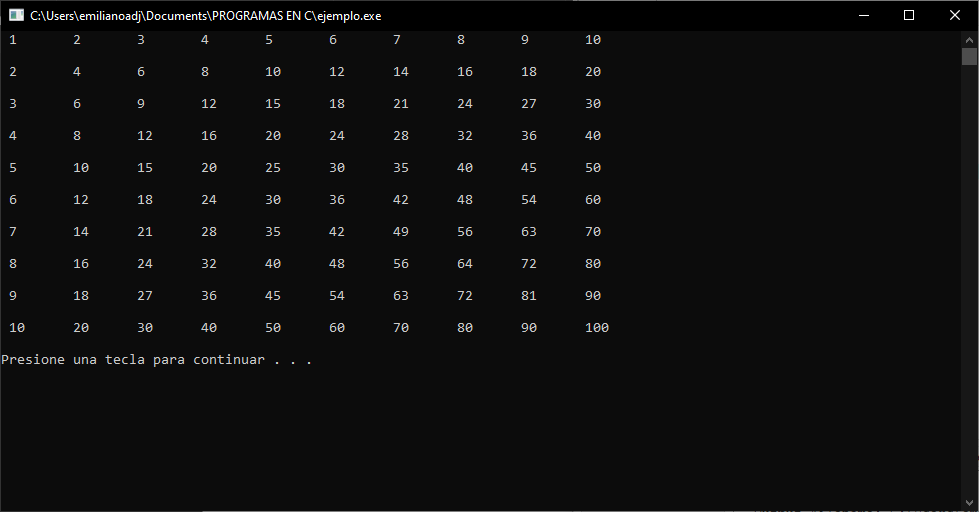
* **Escribe un programa que reciba un número entero N del usuario e imprima una escalinata de N pisos de asteriscos.**



**Este ejercicio utiliza las estructuras repetitivas for, en este caso están anidadas para la correcta impresión de un triangulo al derecho y al revés.**

* **Haz un programa que muestre las tablas de multiplicar.**

**Este programa muestra todas las tablas de multiplicar en forma de una tabla como se muestra a continuación.**



1. **CONCLUSIONES:**

**Las estructuras repetitivas y simples nos sirven de manera que podemos tener un bloque de código diferente dentro de cada una.**

**Nos ayudan a separar instrucciones, pero pueden en ocasiones ser algo confusas.**

**Cada tipo de estructura tiene diferente sentencia por eso debemos manipularlas bien para no cometer errores graves.**