|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre de la práctica** | **Lenguaje C** | | | **No.** | **1** |
| **Asignatura:** | **Métodos numéricos** | **Carrera:** | **INGENIERIA EN SISTEMAS COMPUTACIONALES** | **Duración de la práctica (Hrs)** |  |

**I. Competencia(s) específica(s):** \* \* \* \* \* \* \* \* \*

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

* Aula

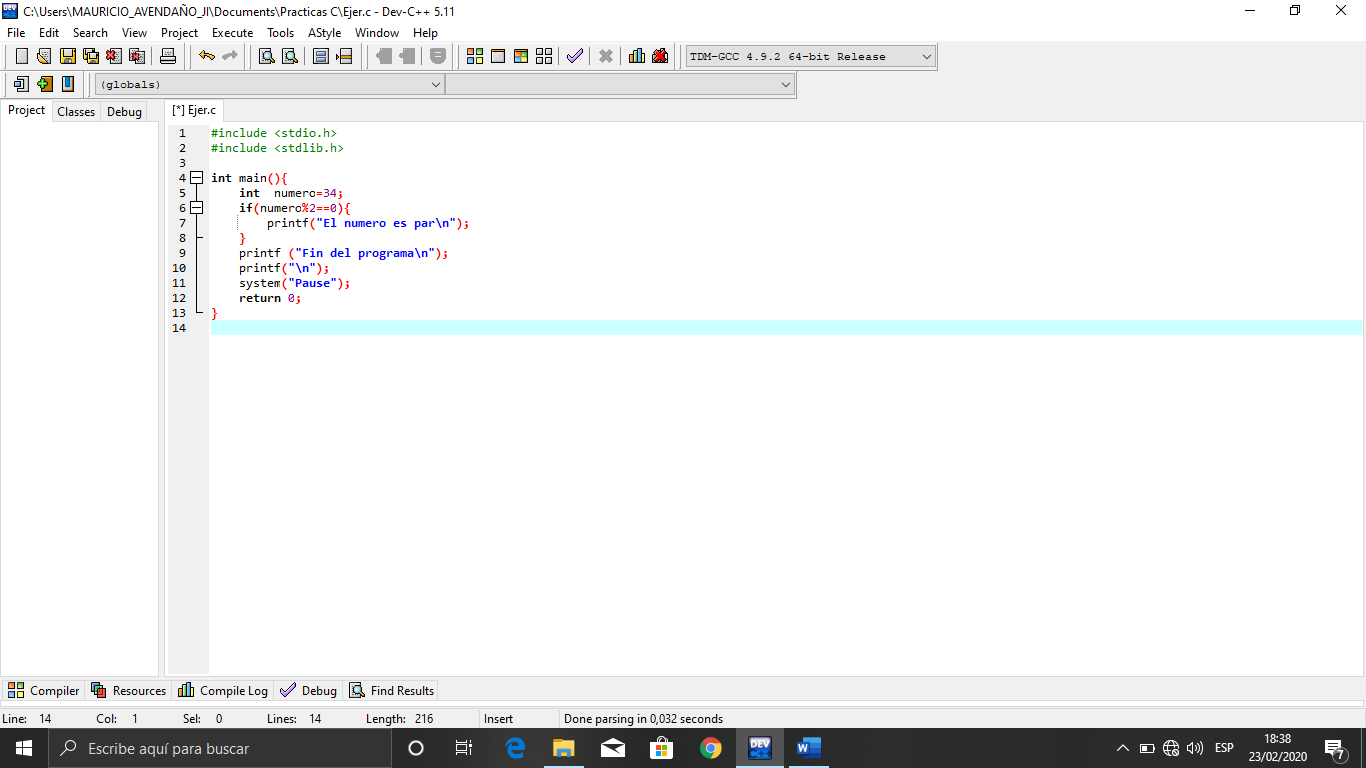
**III. Material empleado:**

* Dev-C++
* Editor de texto
* Documentos

**IV. Desarrollo de la práctica:**

**Documento 8**

Ejemplo. If



Declaración de variable, de tipo entero llamada número, inicializada en 34.

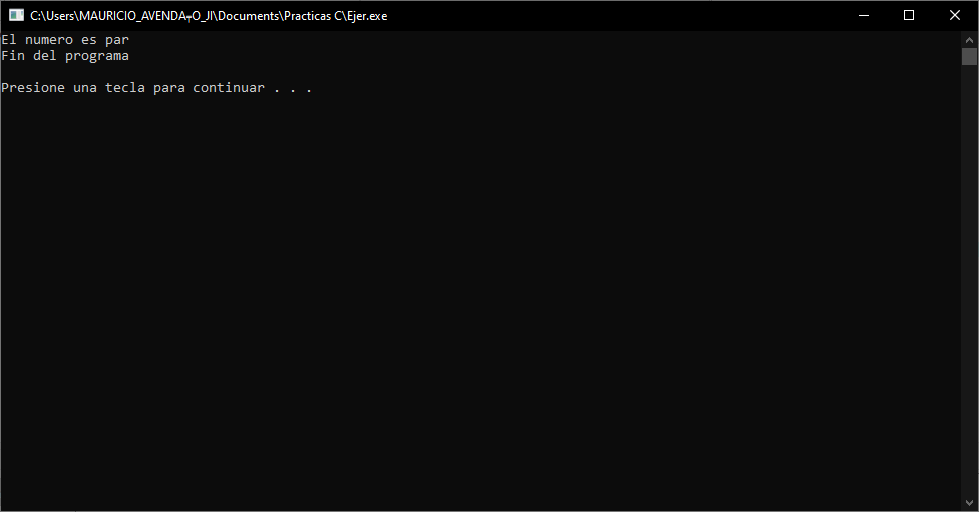
Condición que evalúa si el número es par o no, esto se hace obteniendo el residuo del número entre 2, si el residuo es cero quiere decir que es par, pero si es diferente de 0 es impar

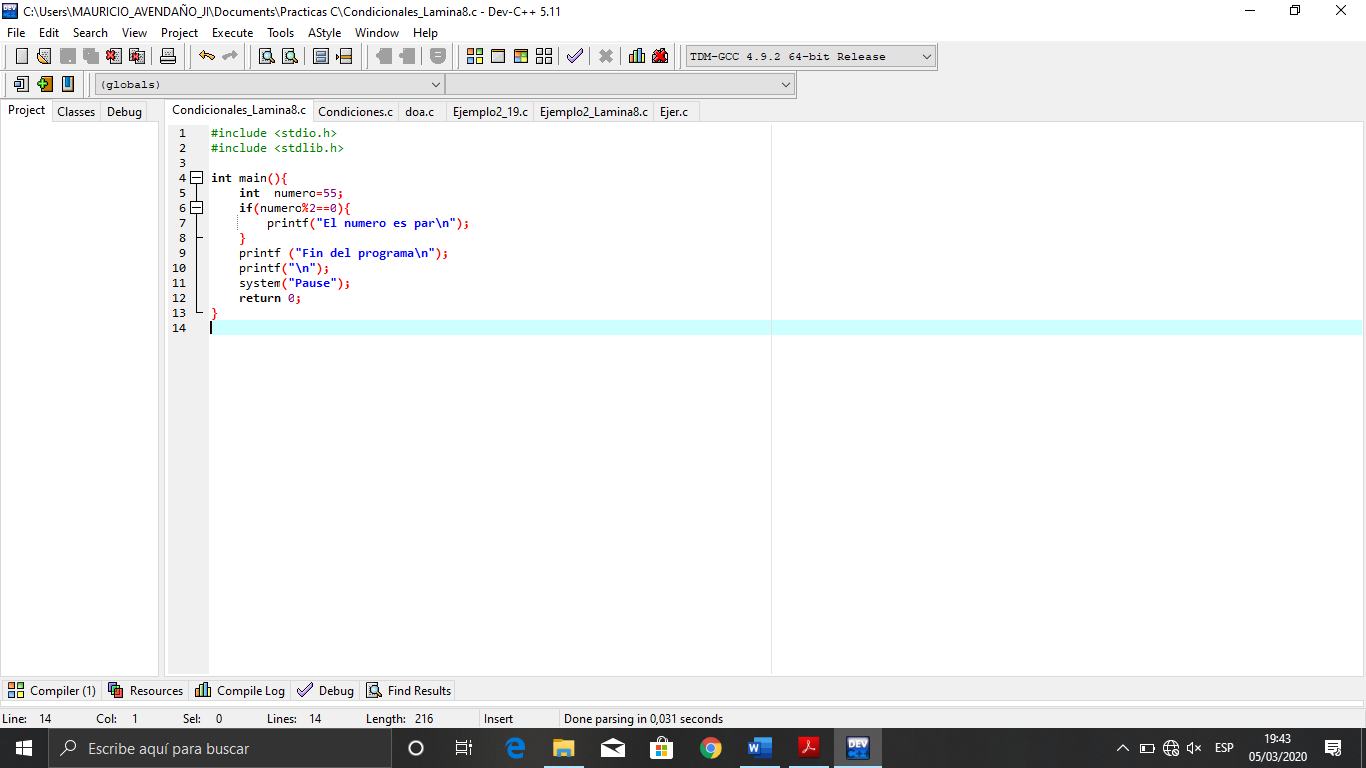
Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Valor que devuelve el método principal

Método principal

Bibliotecas





Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

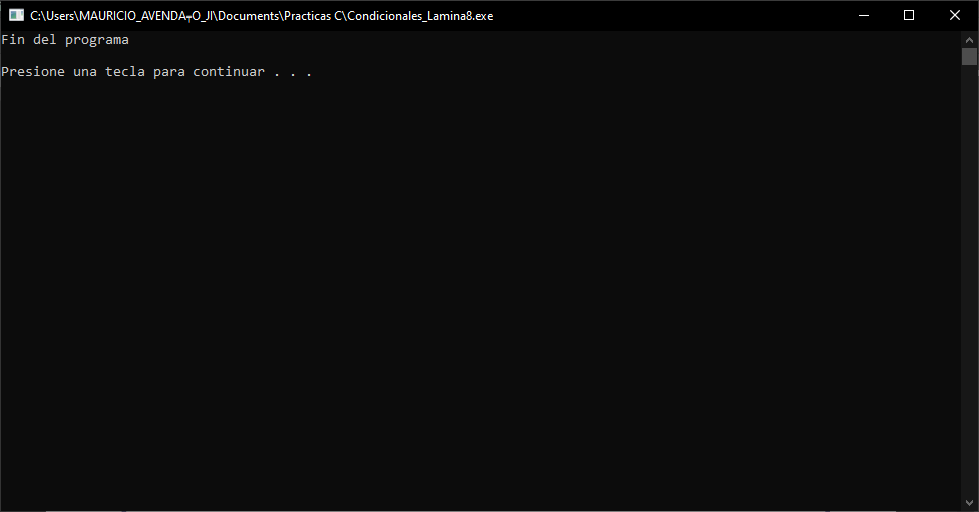
Declaración de variable, de tipo entero llamada número, inicializada en 55.

Condición que evalúa si el número es par o no, esto se hace obteniendo el residuo del número entre 2, si el residuo es cero quiere decir que es par, pero si es diferente de 0 es impar, en este caso la condición es falsa por lo que no ejecuta lo que está dentro del if.

Método principal

Valor que devuelve el método principal

Bibliotecas



Ejercicio. Escriba 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

Valor que devuelve el método principal

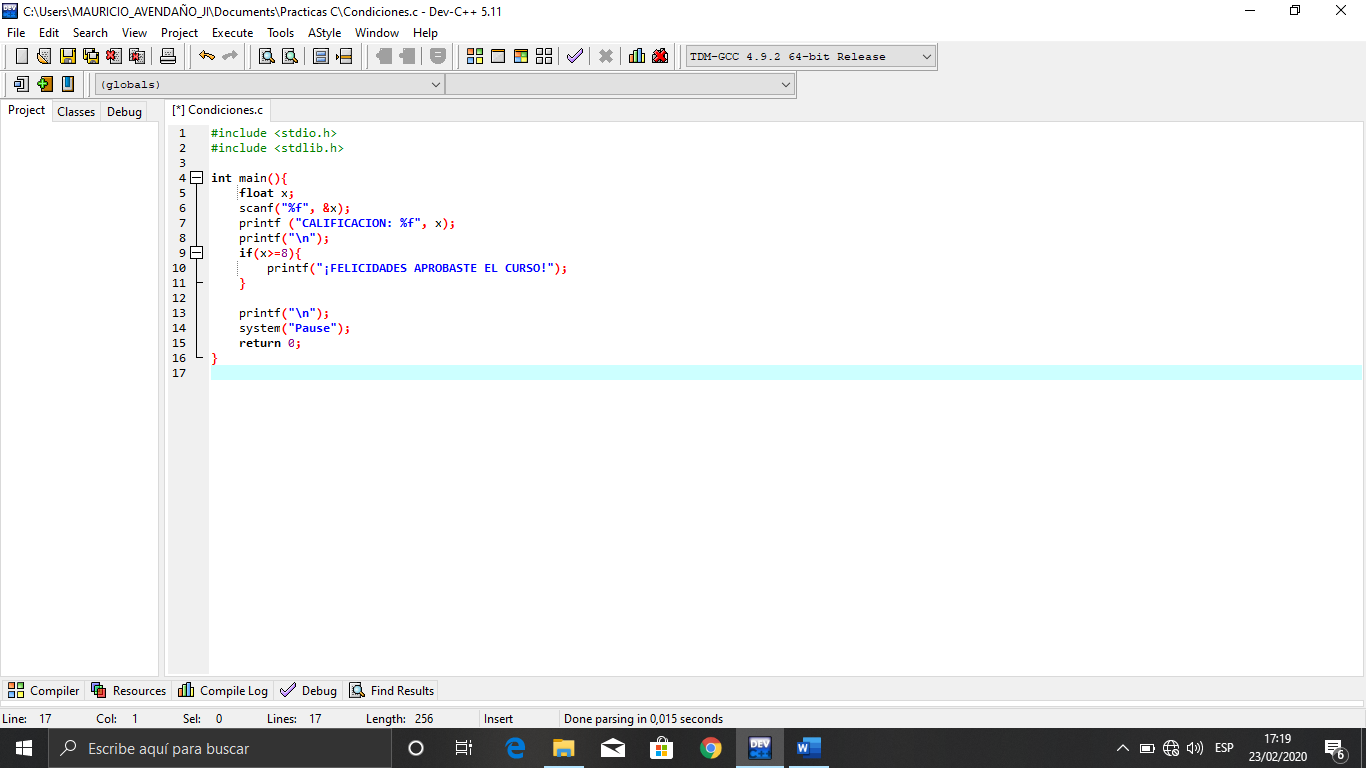
Declaración de variable, de tipo flotante llamada x.

Condición, si el valor de x es mayor o igual a 8 va a imprimir el mensaje de: felicidades, aprobaste el curso, pero si no se cumple la condición simplemente acaba la ejecución del programa.

Mensaje que manda la calificación del estudiante, la variable x ya tiene el valor de entrada, se asigna al especificador y se imprime en pantalla, después de esa línea se hace la comparación.

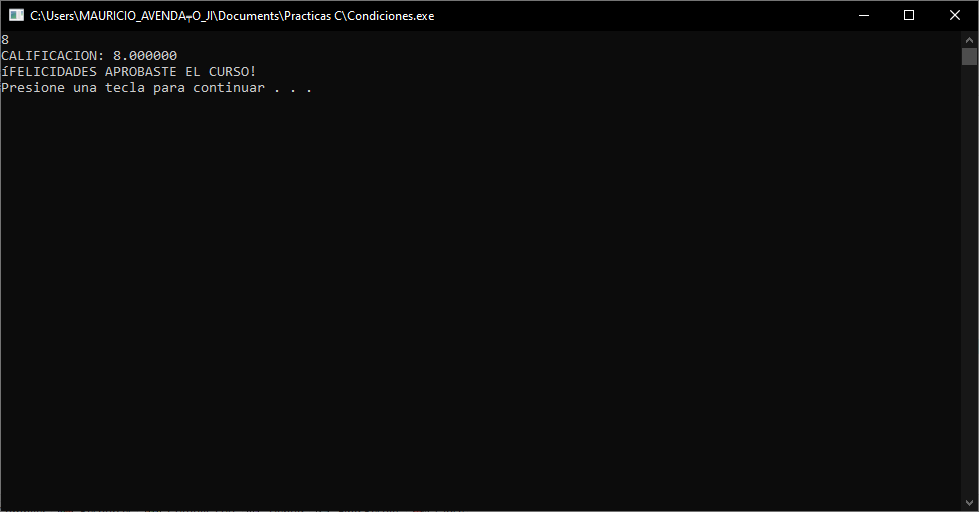
Bibliotecas

Esta función permite ingresar datos por teclado, consta del especificador de conversión, seguido del operador de dirección y el nombre de la variable en la que se alojara el dato ingresado.



Método principal

Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla

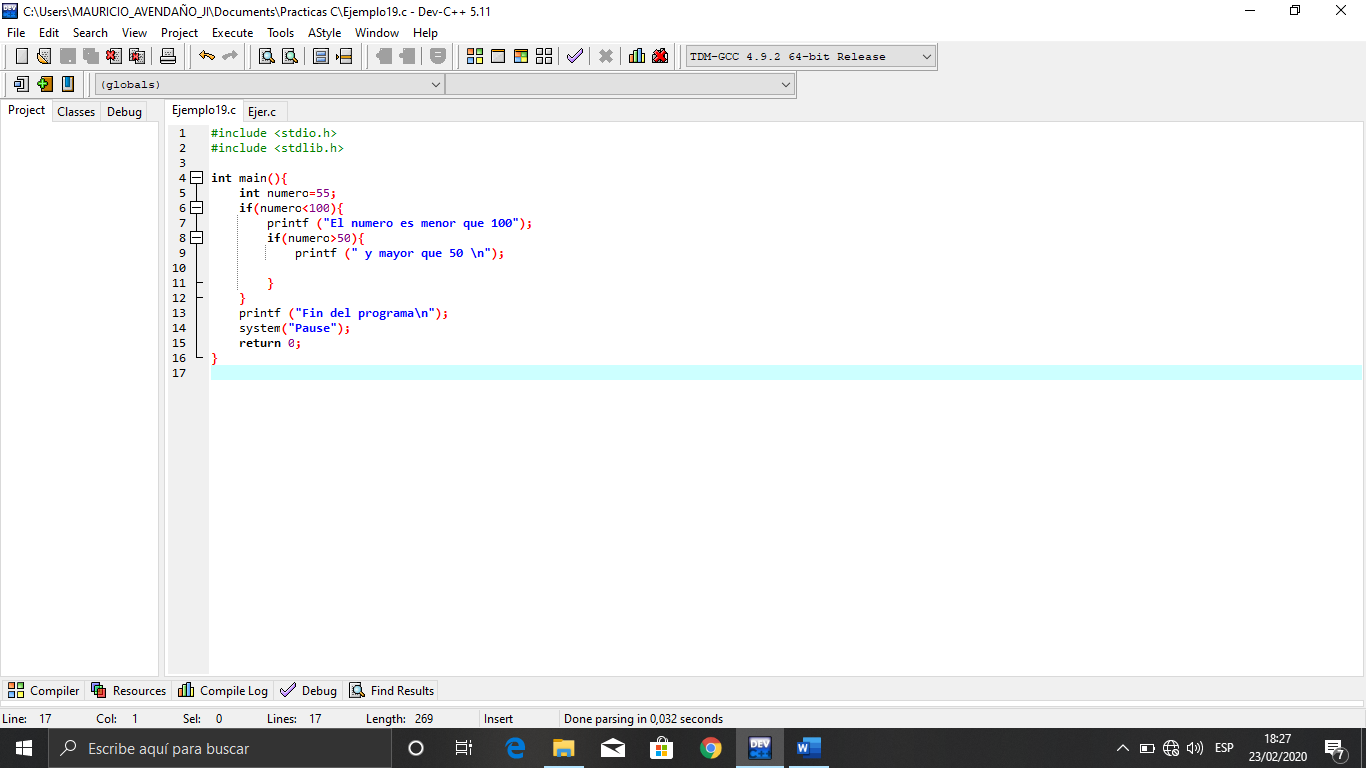


Dato ingresado por teclado, como se puede observar la condición si se cumple porque no es mayor a 8 pero si es igual, así que entra a la condición e imprime el mensaje de aprobado.

Mensaje que se encuentra dentro de la condición y en el caso de que la condición resulte verdadera se ejecutara, en caso contrario saltara esa línea y continuara. En este caso si fue verdadera por lo que si se ejecutó esa instrucción.

Mensaje de la calificación.

Ejemplo. If anidado



Condición que evalúa si número es menor a 100, si es así, se imprime el mensaje: el número es menor que 100, y continúa evaluando, si el número es mayor que 50, también imprimirá el mensaje: y mayor que 50, pero dado el caso que la primera condición fuera falsa, no imprimiría ni el primer mensaje y mucho menos evaluaría a la segunda condición.

Valor que devuelve el método principal

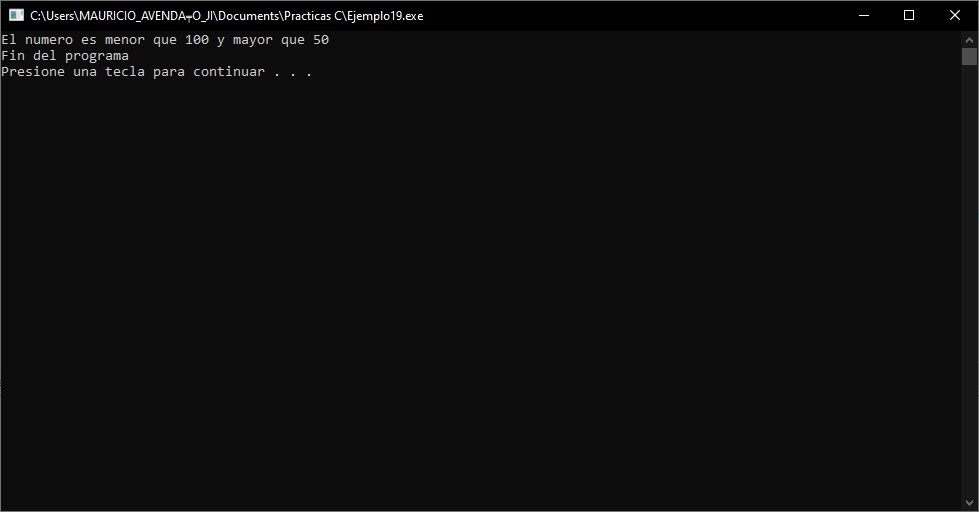
Declaración de variable, de tipo entero llamada número, inicializada en 55.

Método principal

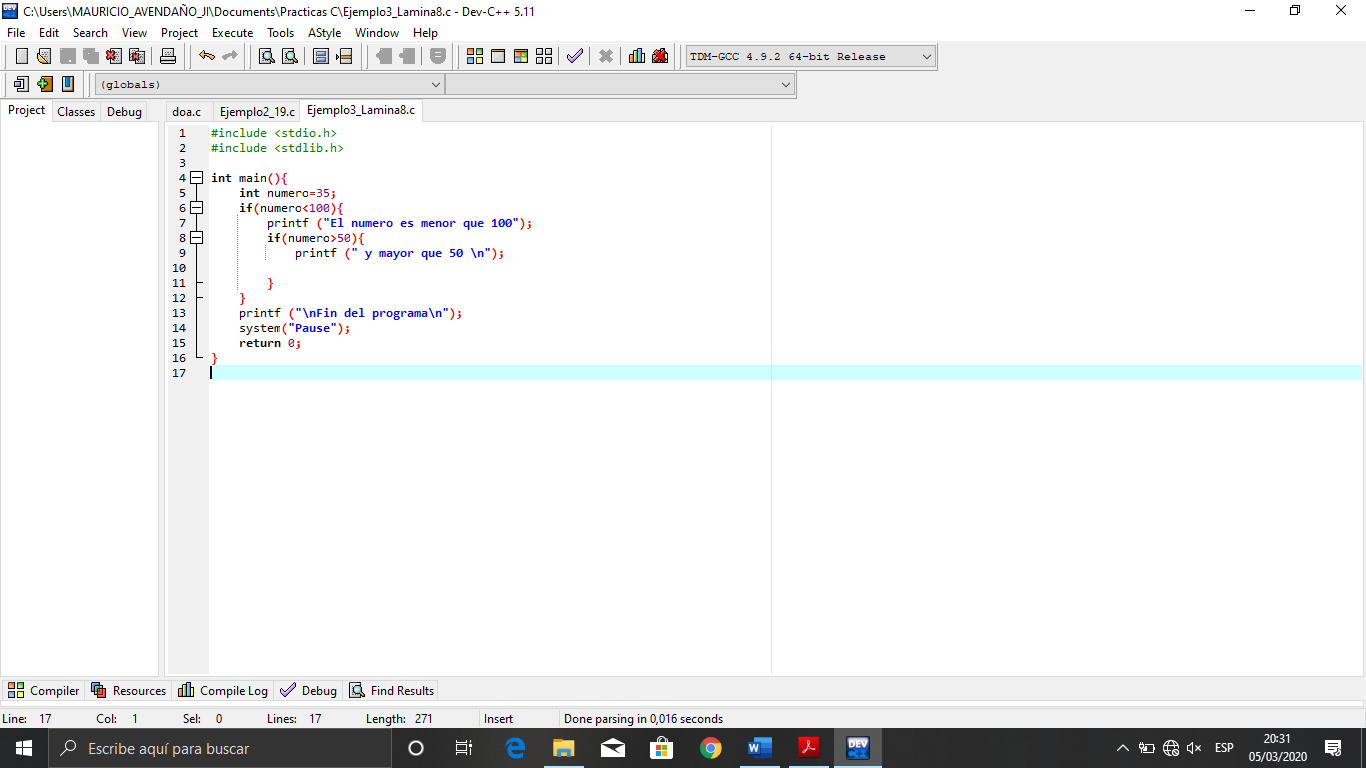
Mensaje y salto de línea al final

Bibliotecas

Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.



Ejemplo. If anidado



Valor que devuelve el método principal

Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

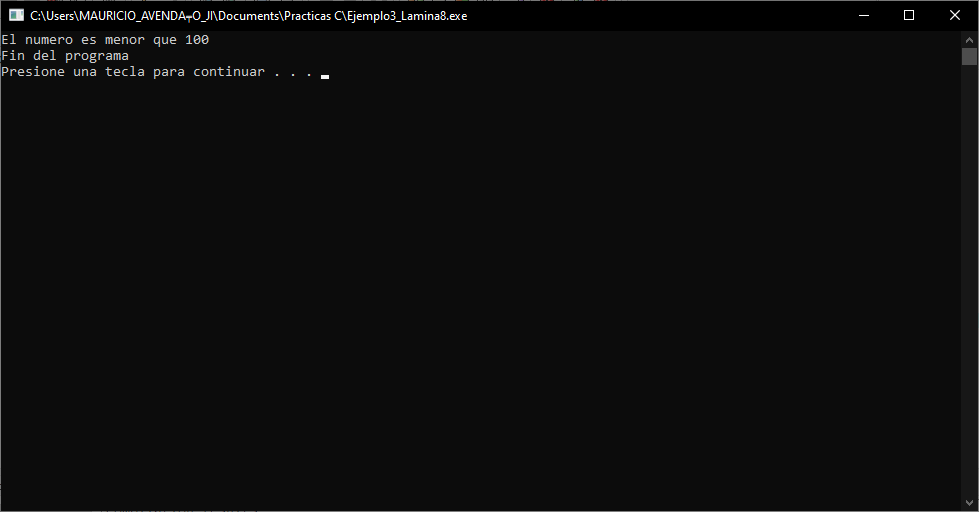
Mensaje y salto de línea al final

Condición que evalúa si número es menor a 100, si es así, se imprime el mensaje: el número es menor que 100, y continúa evaluando, si el número es mayor que 50, también imprimirá el mensaje: y mayor que 50, pero dado el caso que la primera condición fuera falsa, no imprimiría ni el primer mensaje y mucho menos evaluaría a la segunda condición.

Declaración de variable, de tipo entero llamada número, inicializada en 35.

Método principal

Bibliotecas



Ejemplo. If anidado



Declaración de variable, de tipo entero llamada número, inicializada en 35.

Condición que evalúa si número es menor a 100, si es así, se imprime el mensaje: el número es menor que 100, y continúa evaluando, si el número es mayor que 50, también imprimirá el mensaje: y mayor que 50, pero dado el caso que la primera condición fuera falsa, no imprimiría ni el primer mensaje y mucho menos evaluaría a la segunda condición.

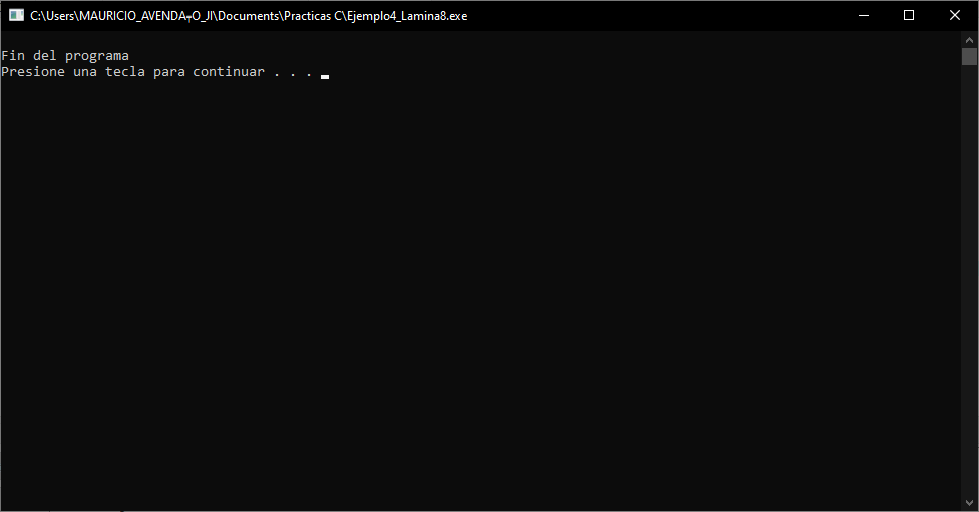
Mensaje y salto de línea al final

Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Valor que devuelve el método principal

Método principal

Bibliotecas

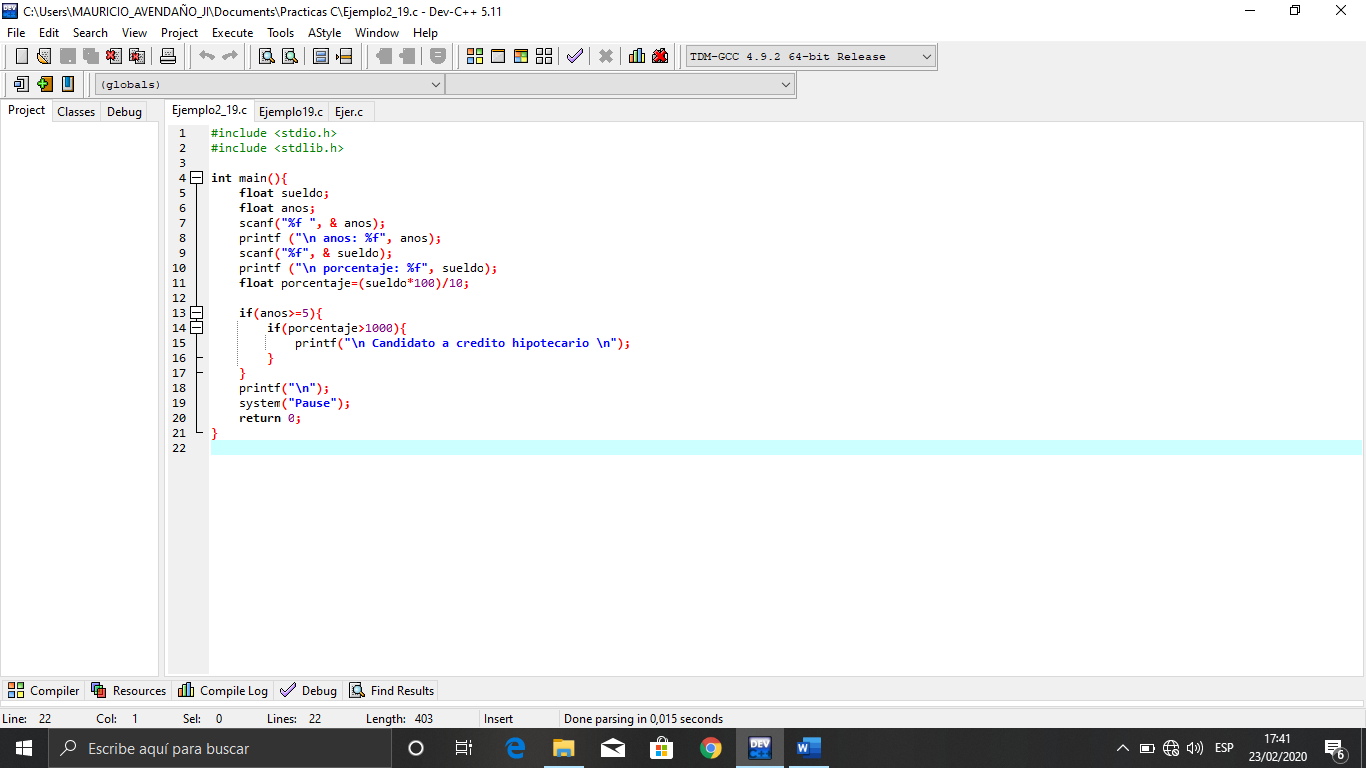


Ejercicio. Estructuras de decisión simple anidadas.

Realiza un programa, que te diga si una persona es sujeto a un crédito hipotecario. Las condiciones que debe cumplir son:

* Tener mínimo 5 años de antigüedad en el trabajo
* El 10% de su sueldo mensual debe ser mayor a 1,000 pesos. X=(sueldo\*100) /10

Esta función permite ingresar datos por teclado, consta del especificador de conversión, seguido del operador de dirección y el nombre de la variable en la que se alojara el dato ingresado.



Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Declaración de variables, de tipo flotante llamada sueldo y anos.

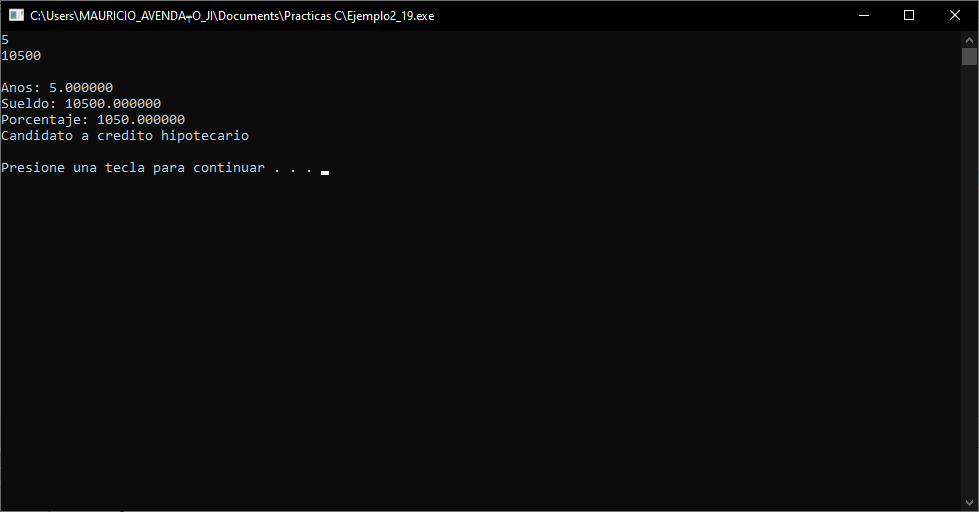
Método principal

Bibliotecas

Primero se hacen las operaciones para determinar el 10% a comparar, el resultado se guardará en la variable porcentaje, después pasa a la condición, si los anos ingresados son mayores o igual a 5 y el porcentaje más de 1000, es candidato a obtener el crédito, pero si no se cumplen ambas no puede.

Mensaje, consta del especificador de conversión, en el que se alojará el valor de la variable y se mostrará en pantalla, el especificador aquí estará esperando un valor de tipo float.

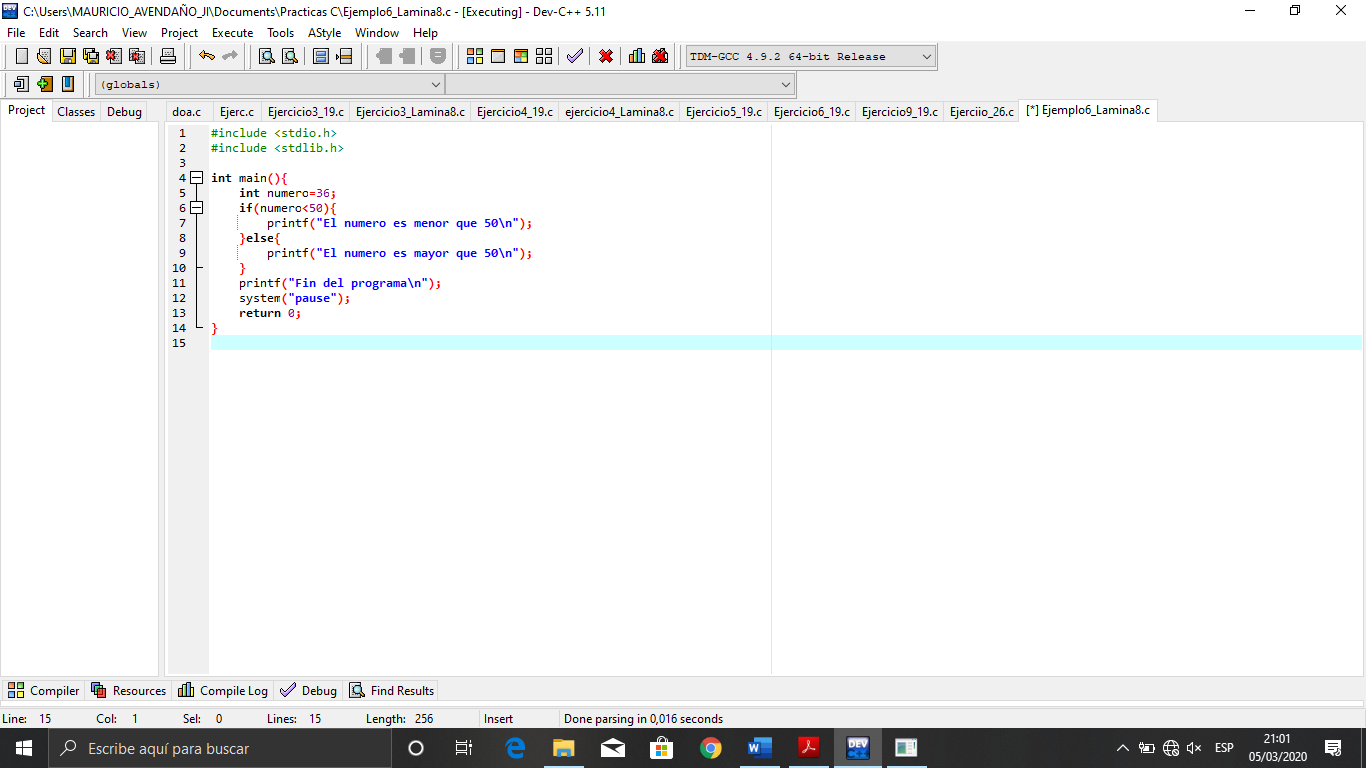
Valor que devuelve el método principal



Datos ingresados por teclado, como nos podemos dar cuenta ingreso 5 años y su sueldo de 10500, el 10% de su sueldo es de 1050 y como la condición pide que por lo menos tenga 5 años y que el 10% de su sueldo sea mayor de 1000, la persona si cumple con los requisitos para pedir el crédito hipotecario, por lo que, si se lo otorgaran, pero en caso que no cumpliera con las dos o ninguna, no se le otorgaría.

Mensaje que se encuentra dentro de la condición y en el caso de que la condición resulte verdadera se ejecutara, en caso contrario saltara esa línea y continuara. En este caso si fue verdadera por lo que si se ejecutó esa instrucción.

Ejemplo. Estructura de decision doble



El objetivo de la decisión doble es ejecutar unas sentencias, dependiendo si la decisión es falsa o verdadera, aquí, compara al número, si es menor a 50 imprime que es menor, pero si no, imprime que es mayor.

Al ultimo imprime un mensaje finalizando el programa.

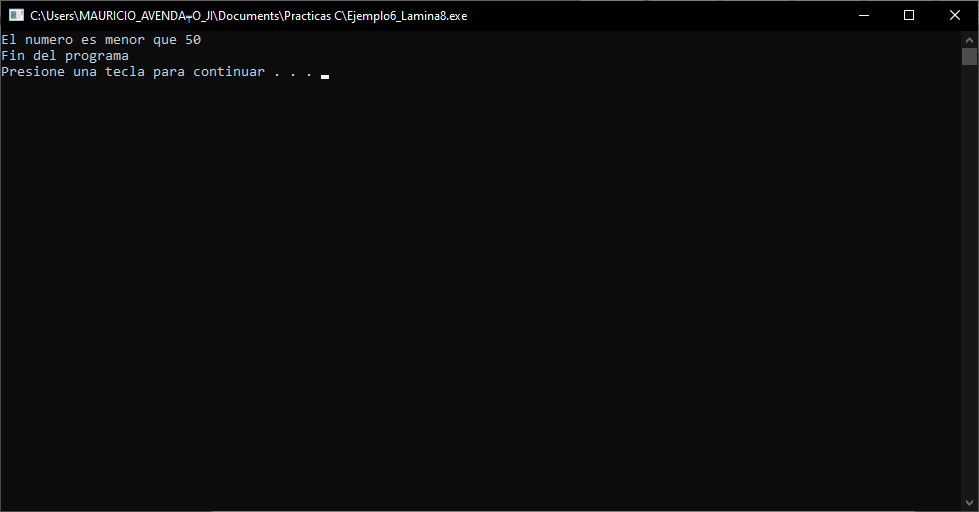
Declaración de la variable numero de tipo entero, con un valor inicial de 36

Método principal

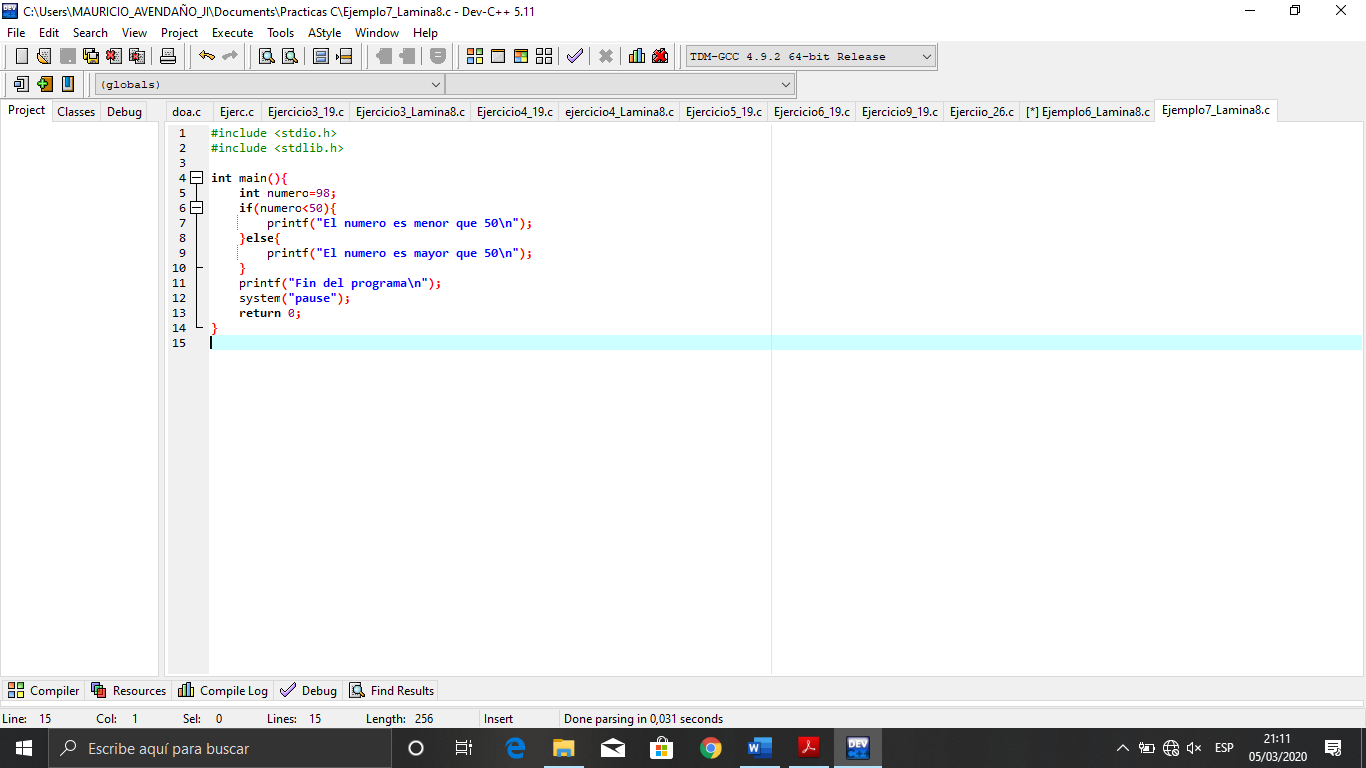
Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Valor que devuelve el método principal

Bibliotecas



Ejemplo. If doble



El objetivo de la decisión doble es ejecutar unas sentencias, dependiendo si la decisión es falsa o verdadera, aquí, compara al número, si es menor a 50 imprime que es menor, pero si no, imprime que es mayor.

Al ultimo imprime un mensaje finalizando el programa.

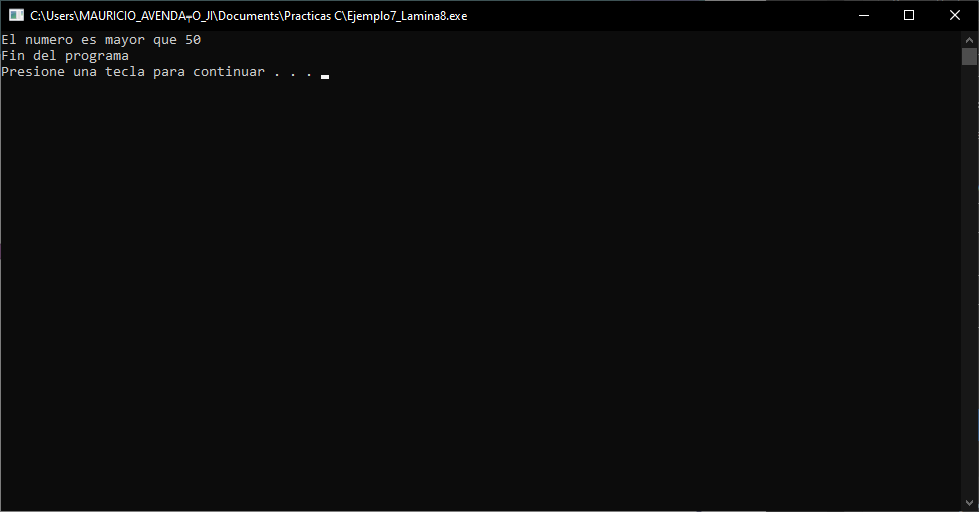
Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Valor que devuelve el método principal

Declaración de la variable número de tipo entero, con un valor inicial de 98

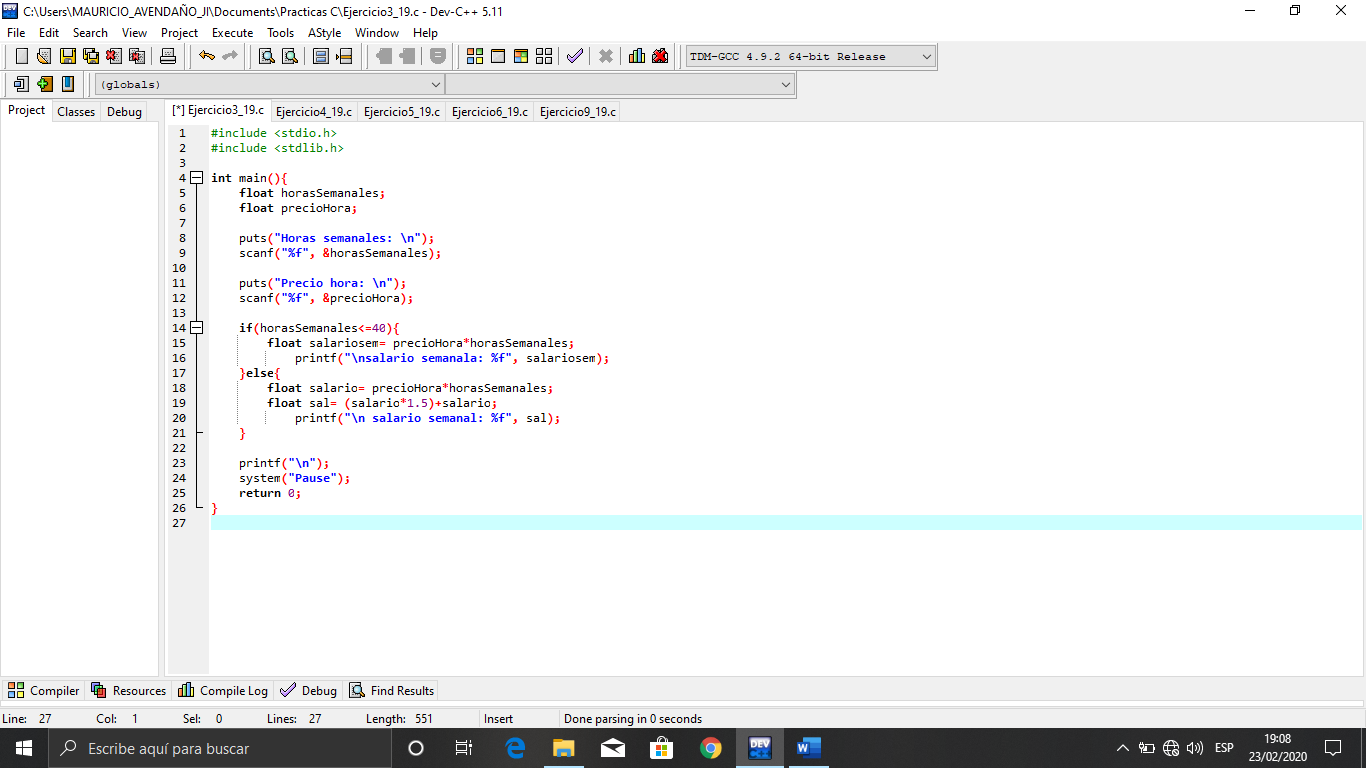
Método principal

Bibliotecas



Ejercicio. Estructuras de decision doble.

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



Función que permite la entrada de datos por teclado, consta de un operador de dirección para la variable y un especificador de conversión del tipo de dato que se espera recibir.

Valor que devuelve el método principal

Condición que evalúa las horas semanales, si son menor o igual a 40, entonces se le paga normal, pero si son más entonces se le paga a razón de 1.5 más del salario normal.

En las variables salariosem, salario y sal, se guardan los resultados de las operaciones. Printf() sirve para imprimir en pantalla un texto y valores de variables, consta de un especificador de conversión para asignar el valor de alguna variable que se desee mostrar.

Especificador de conversión

Operador de dirección

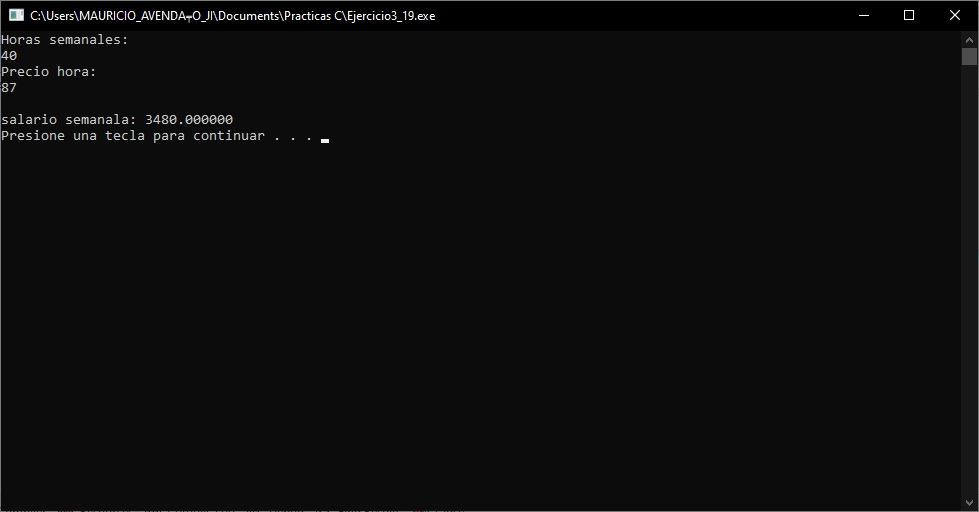
Bibliotecas

Método principal

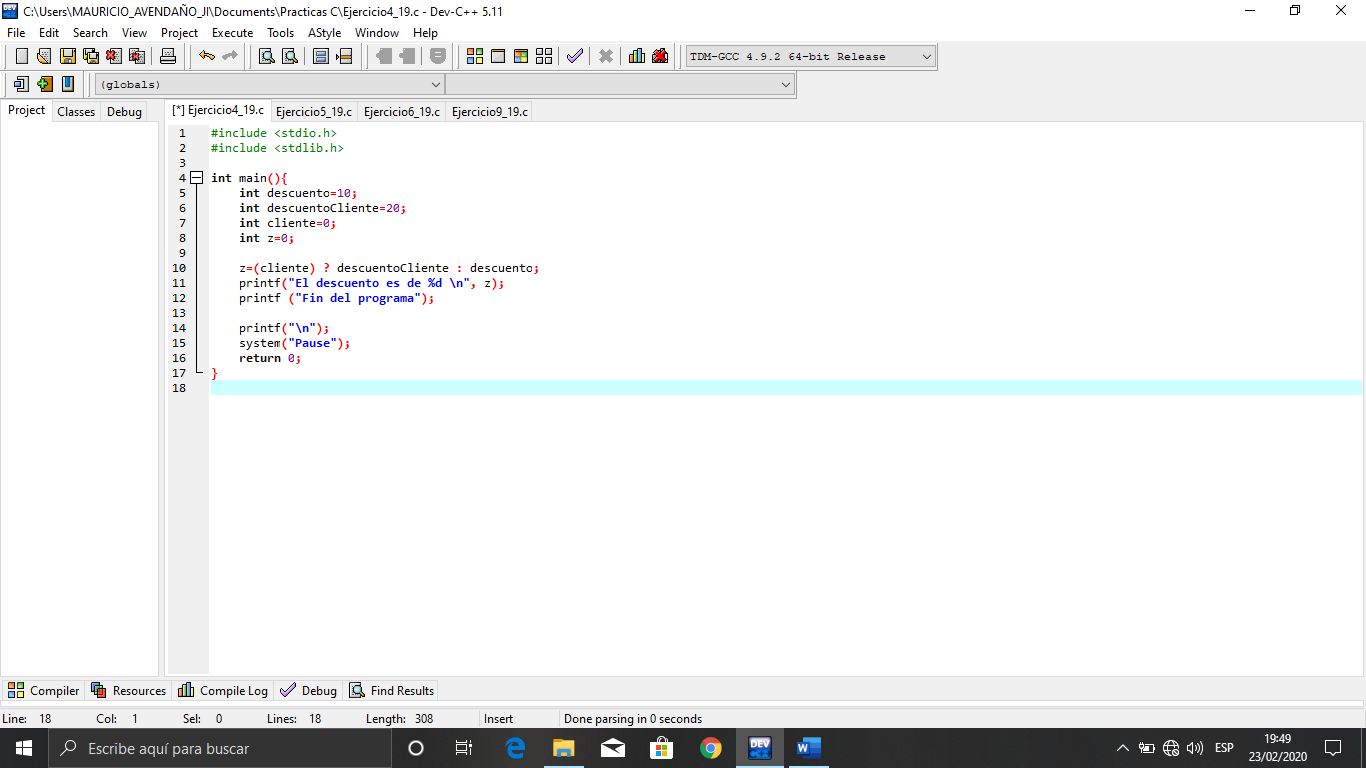
Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Declaración de variables, de tipo flotante llamadas horasSemanales y precioHora.

Puts() permite mandar un mensaje en pantalla.



Ejemplo. Operador condicional ternario.



Declaración de variables, de tipo entero llamadas descuento, descuentoCliente, cliente y z, cada una con un valor, cabe mencionar que el valor que adoptan cliente y z, aunque son de tipo int toman un valor booleano, en este caso 0 corresponde a false, si fuera 1 seria true.

Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

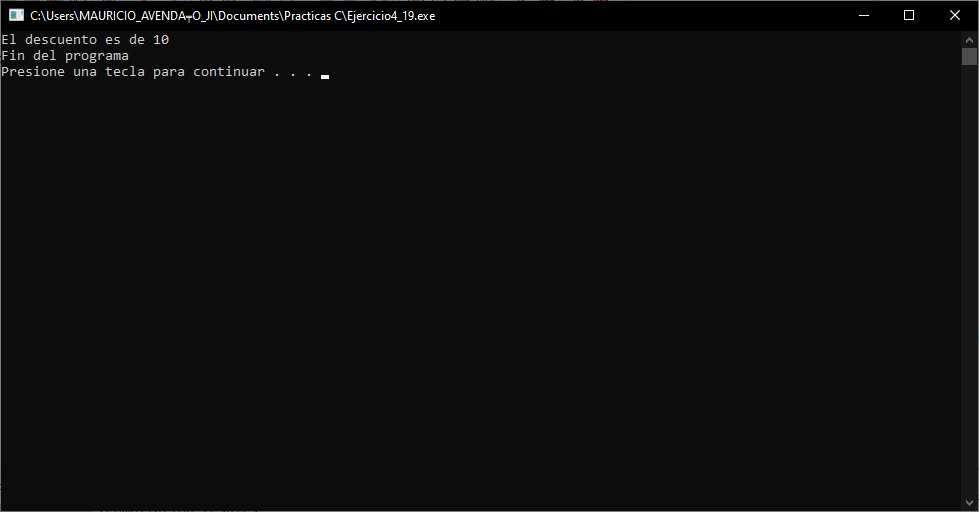
El operador ternario indica que si el primer enunciado es verdadero toma el segundo enunciado, pero si el primero es falso toma en cuenta el tercero, luego con ayuda del printf() mostramos en pantalla el valor que tomo z, y el mensaje de finalizado.

Método principal

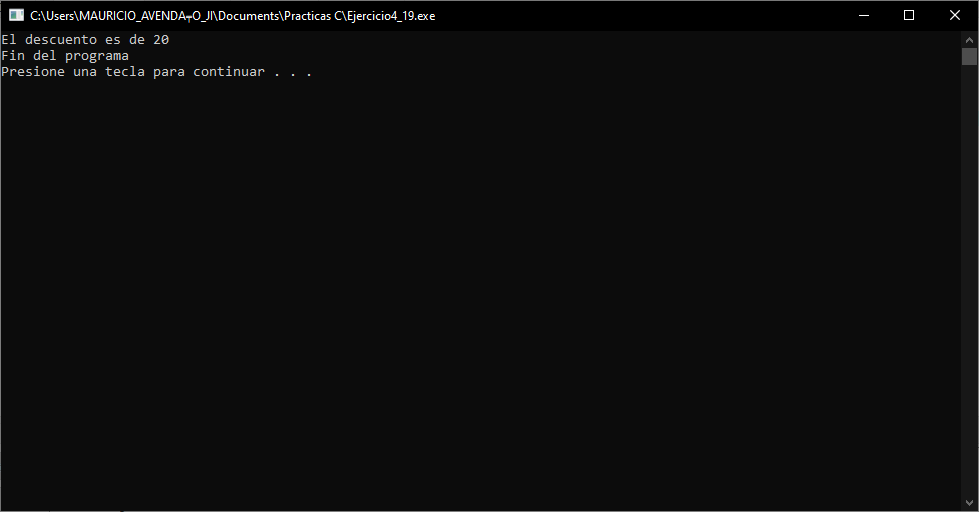
Bibliotecas

Valor que devuelve el método principal

Como nos podemos dar cuenta, el valor de cliente es 0, es decir, falso y como la indicación del operador ternario dice que si es falso toma el tercer enunciado, entonces z adopta el valor de 10, que es descuento.



En el caso que el valor de cliente fuera 1, es decir, true el valor que z adoptaría sería el de descuentoCliente que es 20, porque la indicación del operador ternario dice que si el enunciado 1 es verdadero toma el valor del 2.



**Documento 9**

Ejemplo. While. Imprimir 3 veces la palabra hola



Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Ciclo while, la función de este es ejecutar cierto número de instrucciones mientras la condición se siga cumpliendo, es decir, tenemos a contador que vale 0, entonces entra al ciclo y pregunta, 0 es menor que 3, como es cierto imprime Hola, e incrementa 1, ahora vale 1, entra de nuevo a la condición y pregunta 1 es menor que 3, es cierto, entonces imprime otra vez Hola, incrementa de nuevo, ahora vale 2, entra otra vez a la condición y pregunta, 2 es menor que 3, es cierto, imprime otra vez Hola, incrementa 1, ahora vale 3, vuelve a la condición y pregunta 3 es menor que 3, falso, como no es cierto sale del ciclo. Imprime el mensaje que esta fuera del ciclo y termina el programa.

Como nos dimos cuenta se deben imprimir 3 Hola, verifiquemos.

Incremento

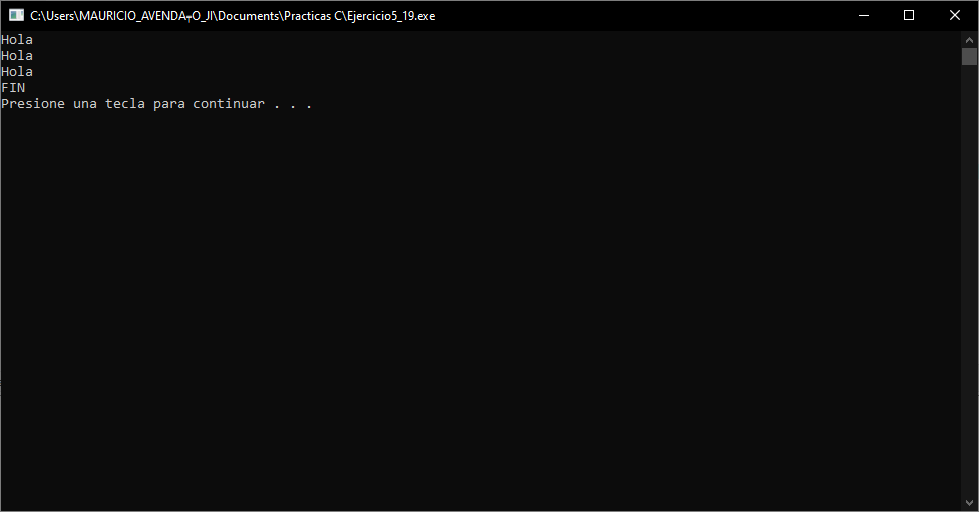
Condición

Declaración del contador de tipo entero, inicializado en 0.

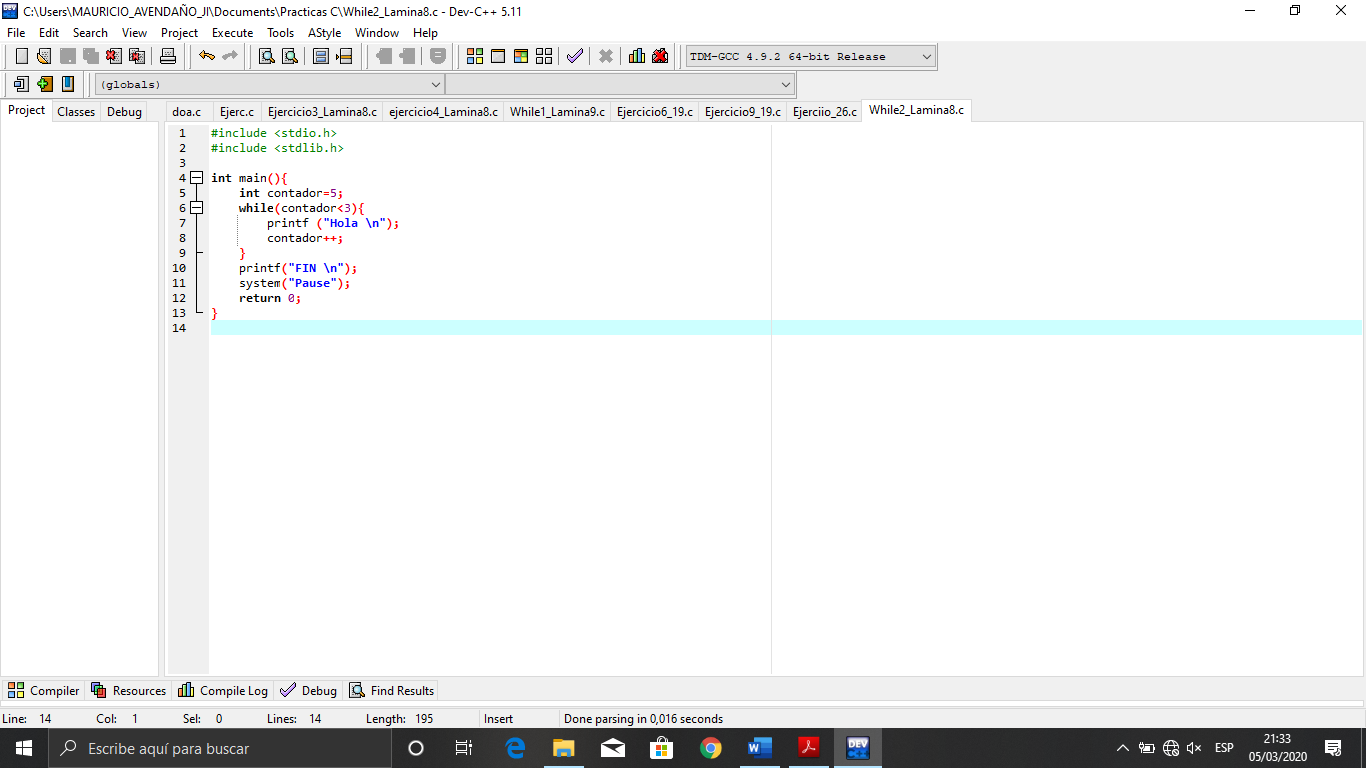
Método principal

Valor que devuelve el método principal

Bibliotecas



Ejemplo. While



Declaración del contador de tipo entero, inicializado en 5.

Ciclo while, la función de este es ejecutar cierto número de instrucciones mientras la condición se siga cumpliendo, es decir, tenemos a contador que vale 5, entonces entra al ciclo y pregunta, 5 es menor que 3, como es falso no ejecuta ninguna instrucción del ciclo, y continua con la ejecución normal, e imprime FIN.

Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

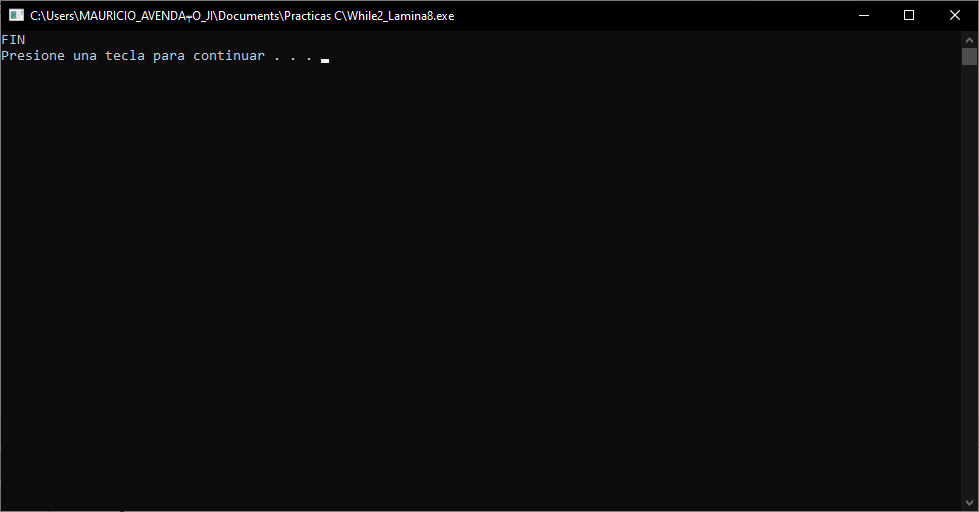
Bibliotecas

Incremento

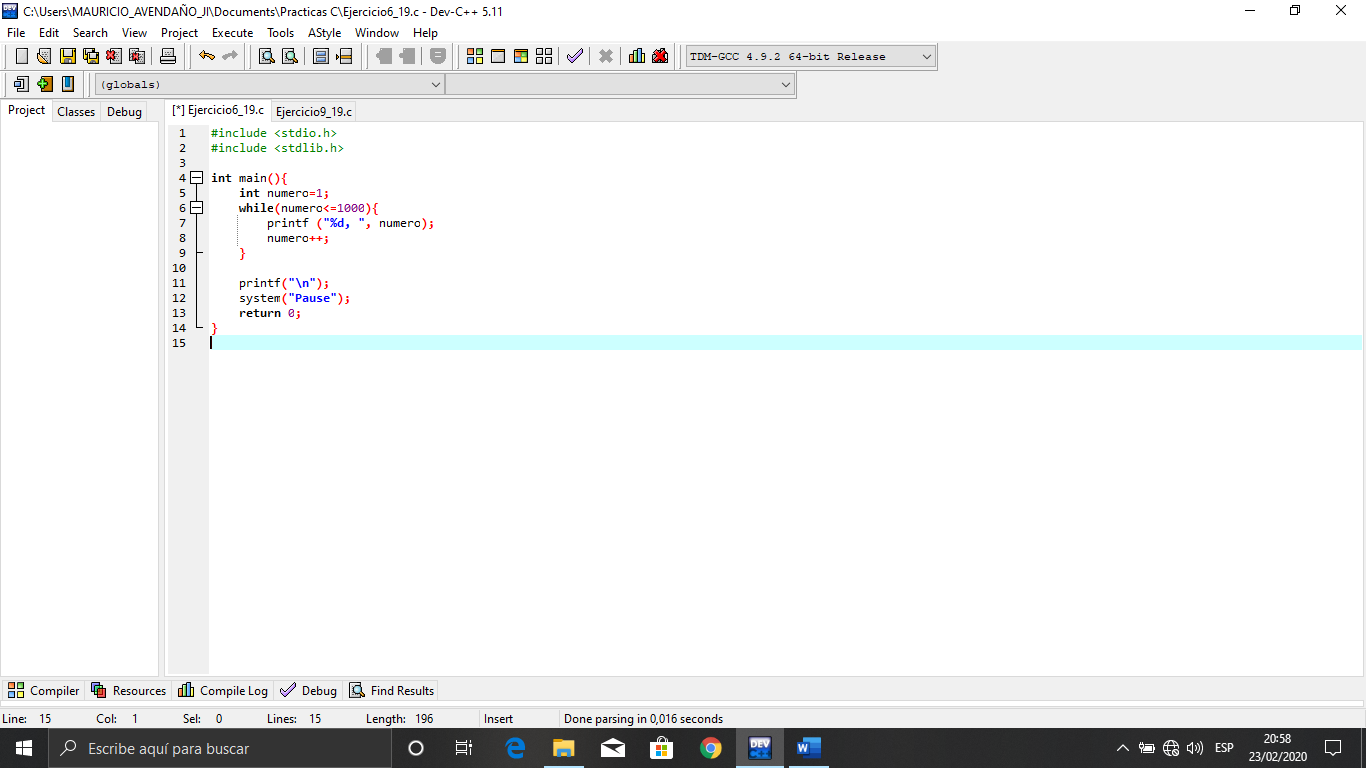
Condición

Valor que devuelve el método principal

Método principal



Ejemplo. While



Ciclo while, la función de este es ejecutar cierto número de instrucciones mientras la condición se siga cumpliendo, es decir, tenemos a número que vale 1, entonces entra al ciclo y pregunta, 1 es menor o igual a 1000, igual no pero menor si,, entonces imprime el valor del número (1), e incrementa 1, ahora vale 2, entra de nuevo a la condición y pregunta 2 es menor o igual que 1000, igual no pero menor si, entonces imprime el 2, incrementa de nuevo, ahora vale 3, entra otra vez a la condición y pregunta, 3 es menor o igual a 1000, igual no pero menor si, entonces imprime a 3, incrementa 1,y así sucesivamente, hasta que el valor del número no sea menor ni igual a 1000, sale del ciclo y termina el programa.

Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Valor que devuelve el método principal

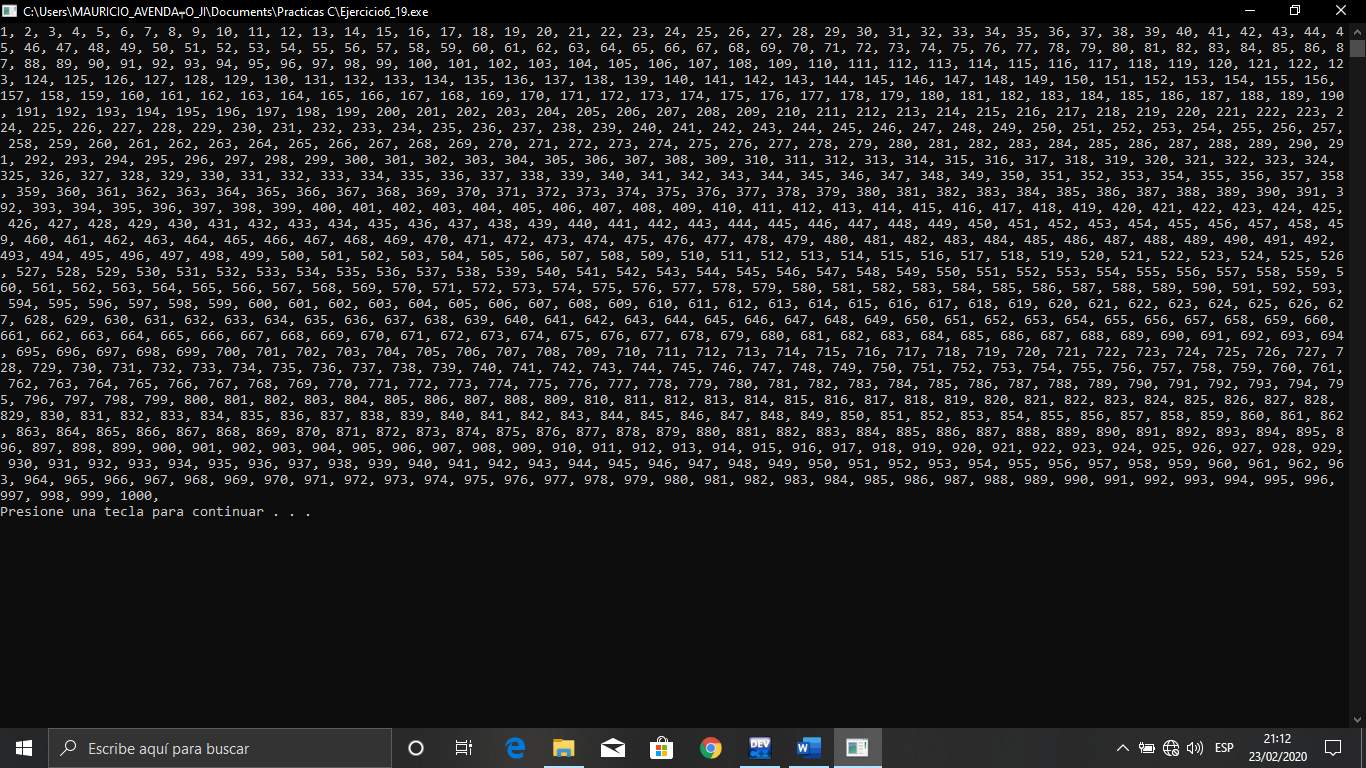
Incremento

Condición

Declaración de la variable número de tipo entero, inicializado en 1.

Método principal

Bibliotecas

continua **…**



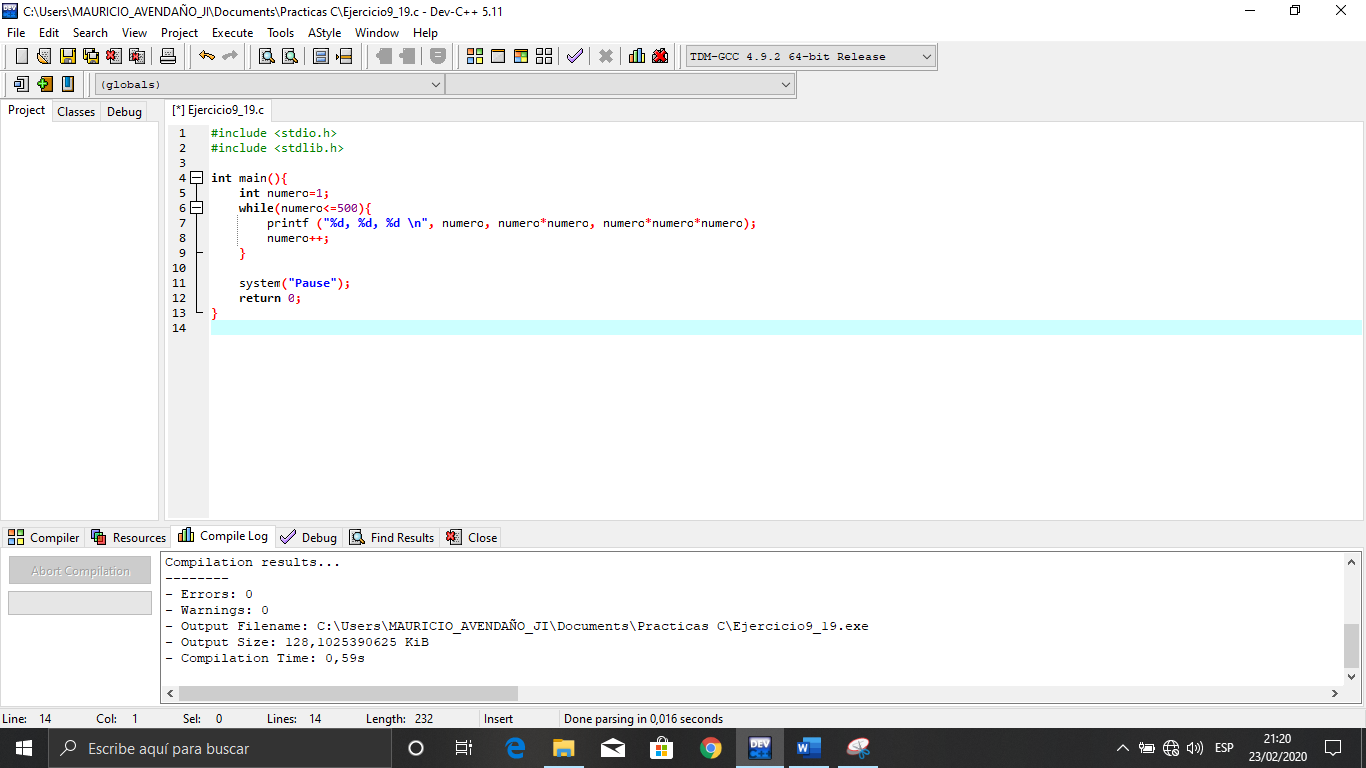
Ejercicio. While. Imprimir los cuadrados y los cubos de los primeros 500 números naturales.

La salida debe ser la siguiente: 1, 1, 1

2, 4, 8

3, 9, 27

4, 16, 64



Método principal

Declaración del contador de tipo entero, inicializado en 0.

Bibliotecas

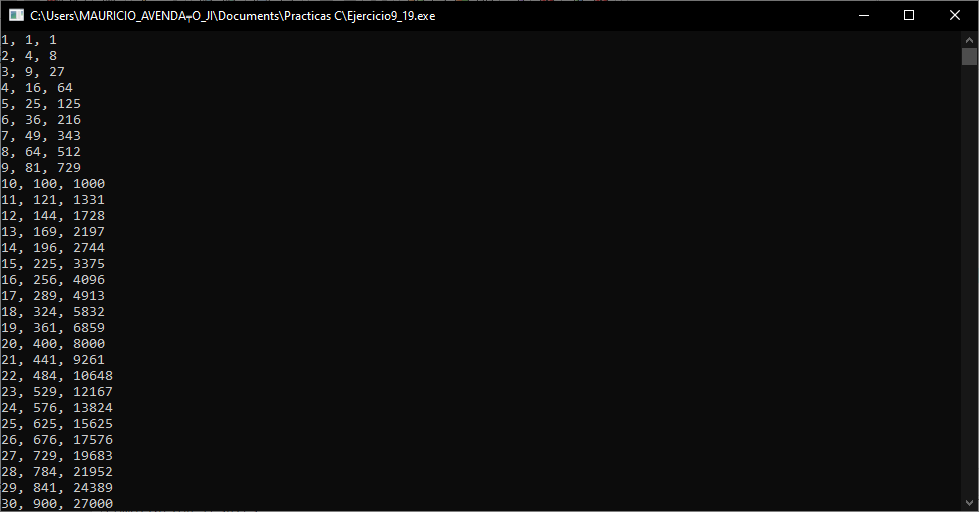
Condición

Incremento

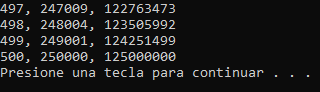
Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Valor que devuelve el método principal

Ciclo while, la función de este es ejecutar cierto número de instrucciones mientras la condición se siga cumpliendo, es decir, tenemos a número que vale 1, entonces entra al ciclo y pregunta, 1 es menor o igual a 500, igual no pero menor si, entonces imprime el número, el cuadrado y el cubo del mismo, resultado de las operaciones, e incrementa 1, ahora vale 2, entra de nuevo a la condición y pregunta 2 es menor o igual que 500, igual no pero menor si, entonces imprime otra vez, el número, su cuadrado y su cubo, incrementa de nuevo, ahora vale 3, entra otra vez a la condición y pregunta, 3 es menor o igual a 500, igual no pero menor si, entonces imprime al número, su cuadrado y su cubo, incrementa 1,y así sucesivamente, hasta que el valor del número no sea menor ni igual a 500, sale del ciclo y termina el programa.



continua**…**



Ejercicios. While

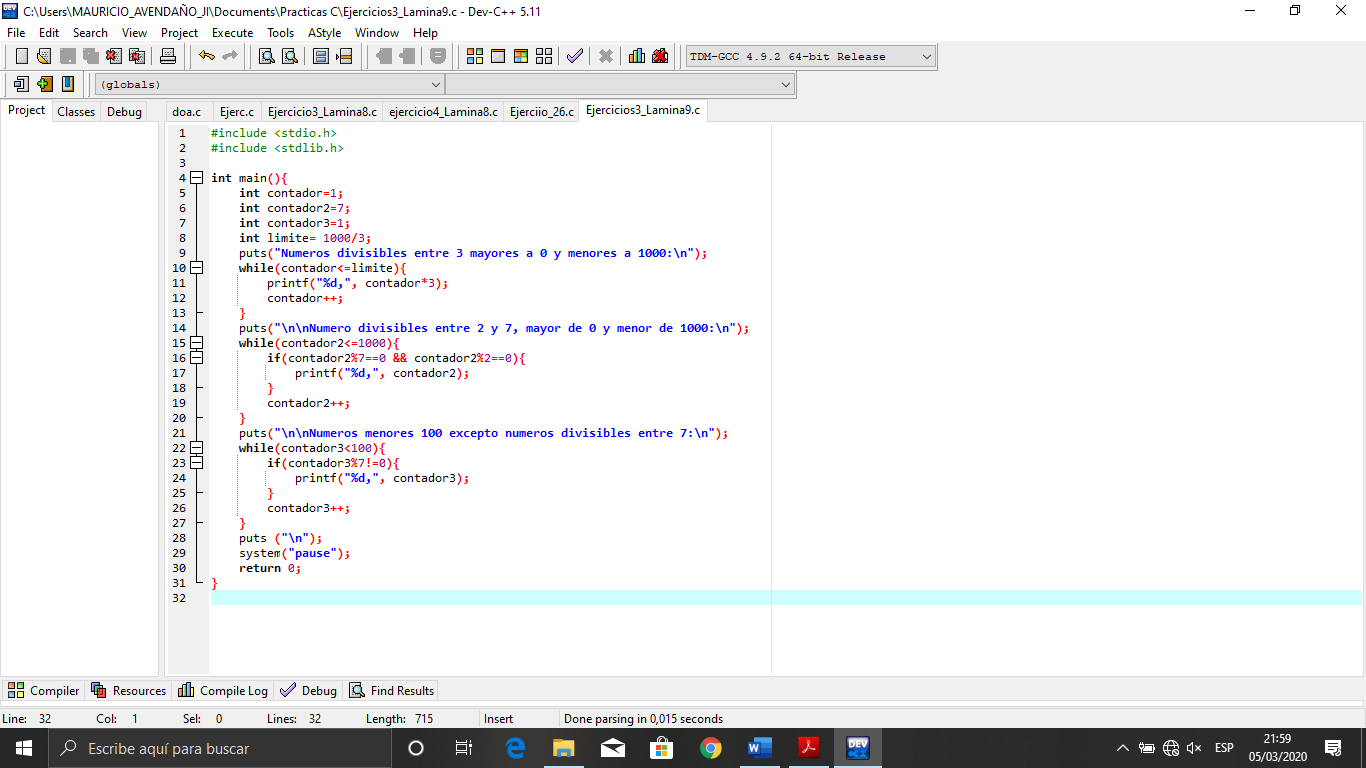
* Imprimir todos los números divisibles entre 3 mayores a 0 y menores a 1000.

Ejemplo: 3, 6, 9, 12, 15, 18, 21, …, 999

* Imprimir todos los números que son divisibles entre 2 y entre 7, mayores a 0 y menores a 1000

Ejemplo: 14, 28, 42, 56, ….

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



Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Valor que devuelve el método principal

Incremento

Condición del ciclo

Incremento

Condición del ciclo

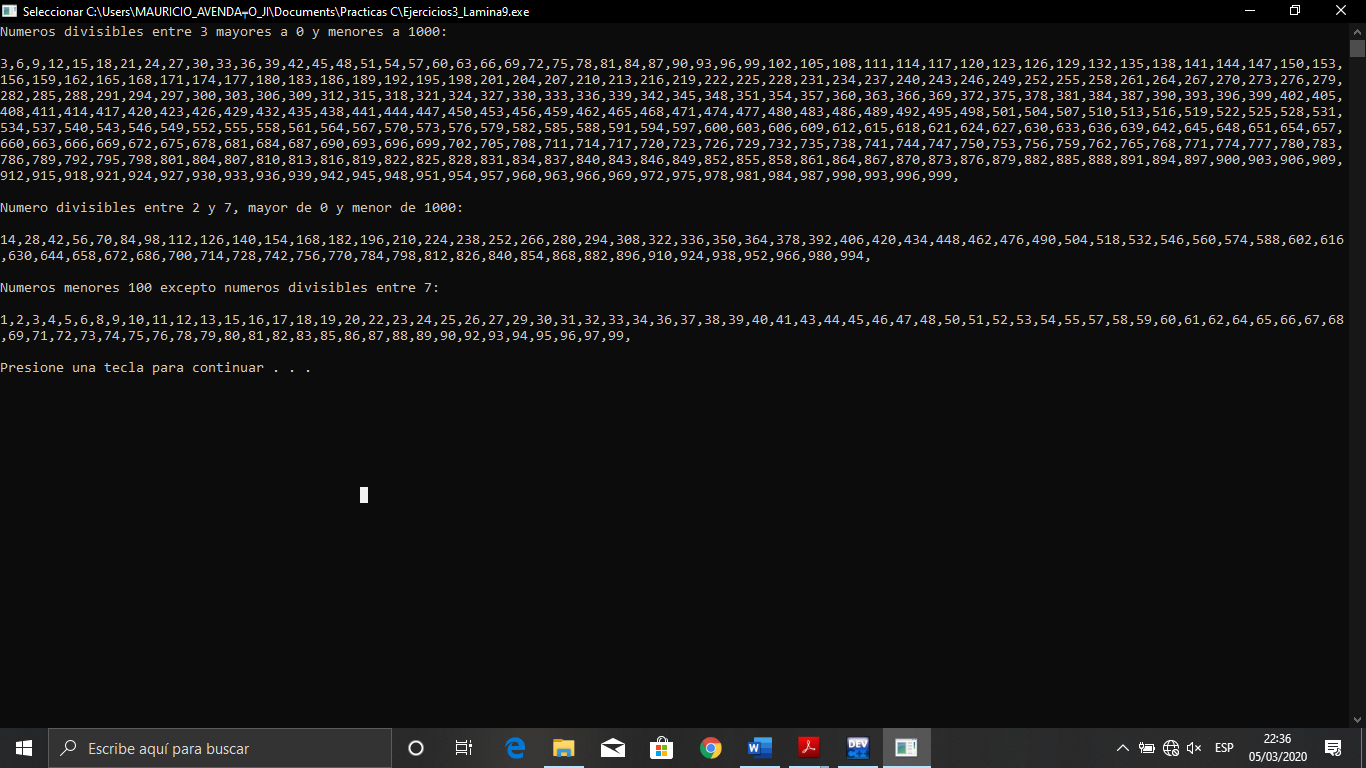
Incremento

Condición del ciclo

Declaración de variables y contadores, todos de tipo entero, pero iniciadas con diferente valor

Bibliotecas

* El primer ejercicio se resuleve con un ciclo, en este caso, se utilizo el while, inicializamos la variable en 1, porque se se declara en 0, cualquier numero multiplicado por 0 va a resultar 0, por ello se hizo esto. Luego por cada incremento, multiplicaremos el contador que va a adoptar el valor de 1, 2, 3, etc. por 3, e imprimimos el resultado, cuando el contador ya no este dentro de la limite de 100, se va a salir de este, y va continuar con las siguientes lineas. Esto es para imprimir los multiplos de 3.
* El siguiente ejercicio, inicializamos el contador en 7, porque no hay un numero antes de ese que sea multiplo de 7, de 2 si hay pero de 7 no, asi que iniciamos de 7, para saber si un numero es divisible por dos numeros, comparamos el residuo, en programacion el modulo, del contador entre 2 y 7, puesto que son los numeros que queremos imprimir. Si el modulo es igual a 0 imprimimos el contador, es decir, el numero, pero si los modulos no son iguales a 0, o por lo menos uno de ellos no es igual a 0, no se va a imprimir el numero, e incrementa en 1, para seguir comparando. Cuando el numero sea mayor al limite se sale del ciclo y termina la ejecucion de ese ciclo.
* Para el tercer ejercicio, esta mas sencillo de resolver, puesto que solo quitaremos los numeros que se dividen entre 7, para ello iniciamo igual el contador en 1, y empezamos a ecomparar si el modulo del numero entre 7 es diferente de 0 lo imprimimos, esto es porque queremos saber si es multiplo de 7, si el modulo no es 0, quiere decir que no es multiplo de 7 asi que se mostrara, si es igual a 0, entonces no se imprime, y asi el contador incrementa hasta llegr a 100.

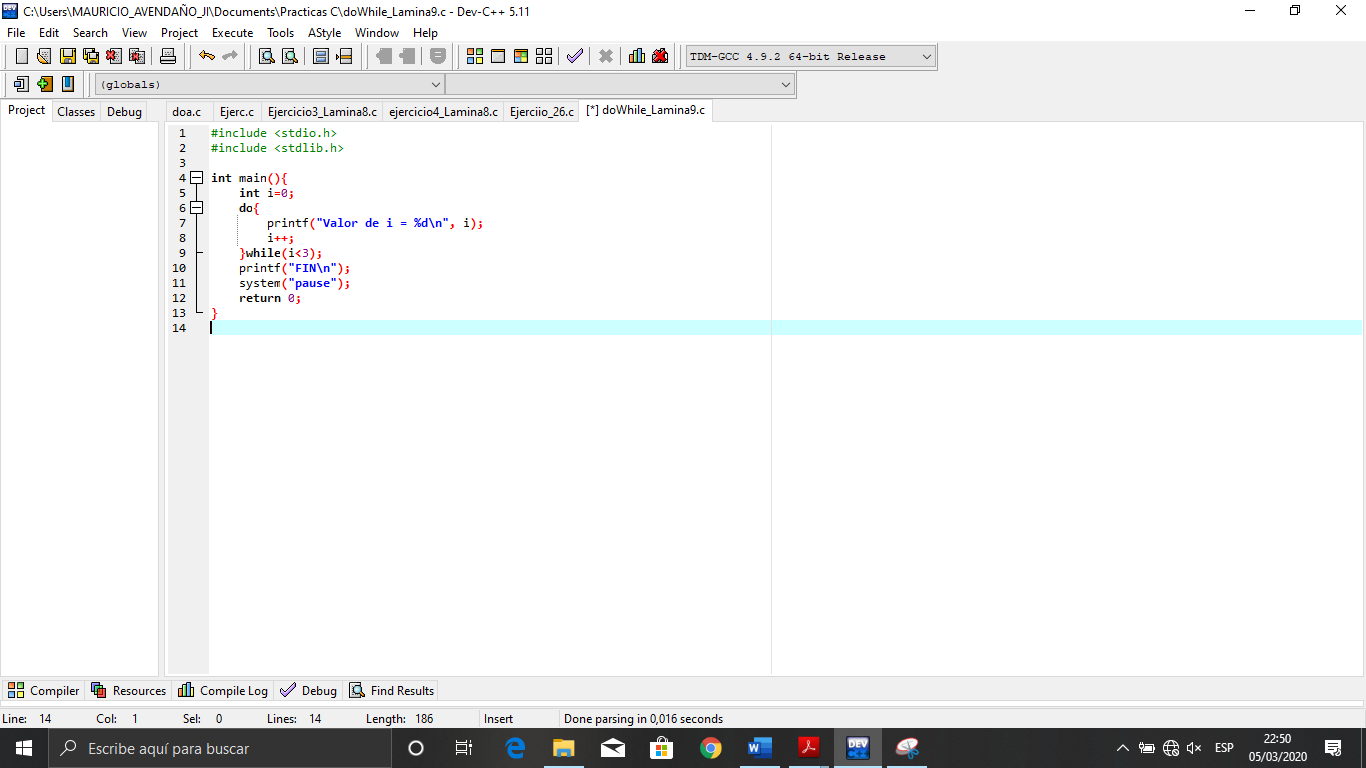
 **. . .**

 **. . .** 

**. . .** 

Ejemplo. Do\_while

Bibliotecas



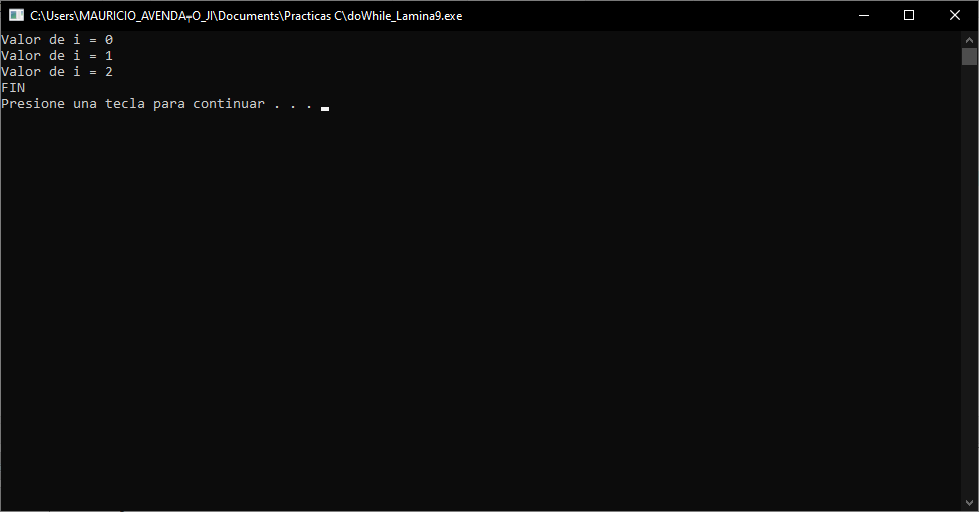
La función del ciclo do-while es ejecutar las instrucciones una primer vez, pero después de esa vez, entra a la condición si, la condición si cumple se sigue ejecutando pero si no, solo se ha ejecutado esa primera vez, en el código, i vale 0, entonces imprime el valor de i e incrementa, ahora i vale 1, y pregunta, 1 es menor a 3, como es verdadero vuelve a imprimir a i, incremente otra vez y ahora vale 2, como sigue siendo menor que 3, imprime otra vez, pero ya cuando incremente vale 3 y la condición ya no se cumple, porque 3 no es menor que 3, si no igual, por lo que sale del ciclo.

Valor que devuelve el método principal

Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Declaración del contador de tipo entero, inicializado en 0.

Método principal

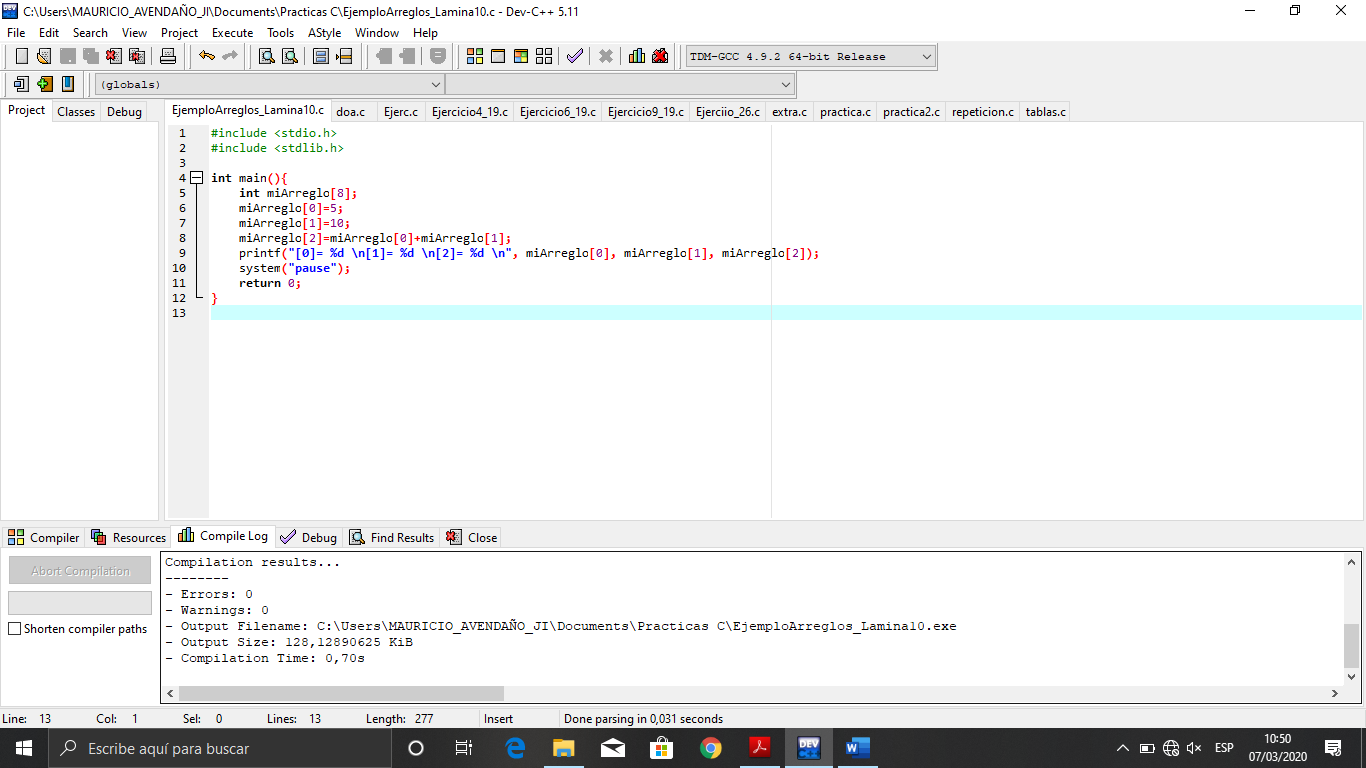


**Documento 10**

Ejemplo. Arreglos.

Método principal

Primero le asignamos el valor de 5, a la posición 0 del arreglo, cabe destacar que hablando de arreglos las posiciones comienzan desde 0, en posición 1, se le asigna el valor de 10, al final en la posición 2, se le asigna la suma de la posición 0 + la posición 1, y se imprime el valor de las 3 posiciones.



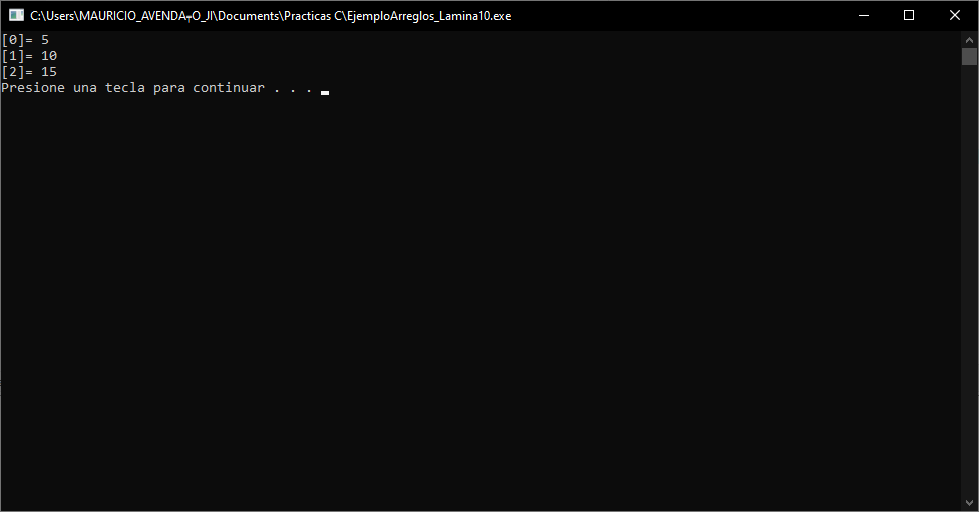
Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

En esta línea de código, le asignamos por medio del especificador de conversión el valor de cada posición del arreglo que deseamos mostrar en pantalla, pero cade mencionar, que cuando se quiera mostrar más de un valor se tienen que indicar un especificador de conversión para cada valor.

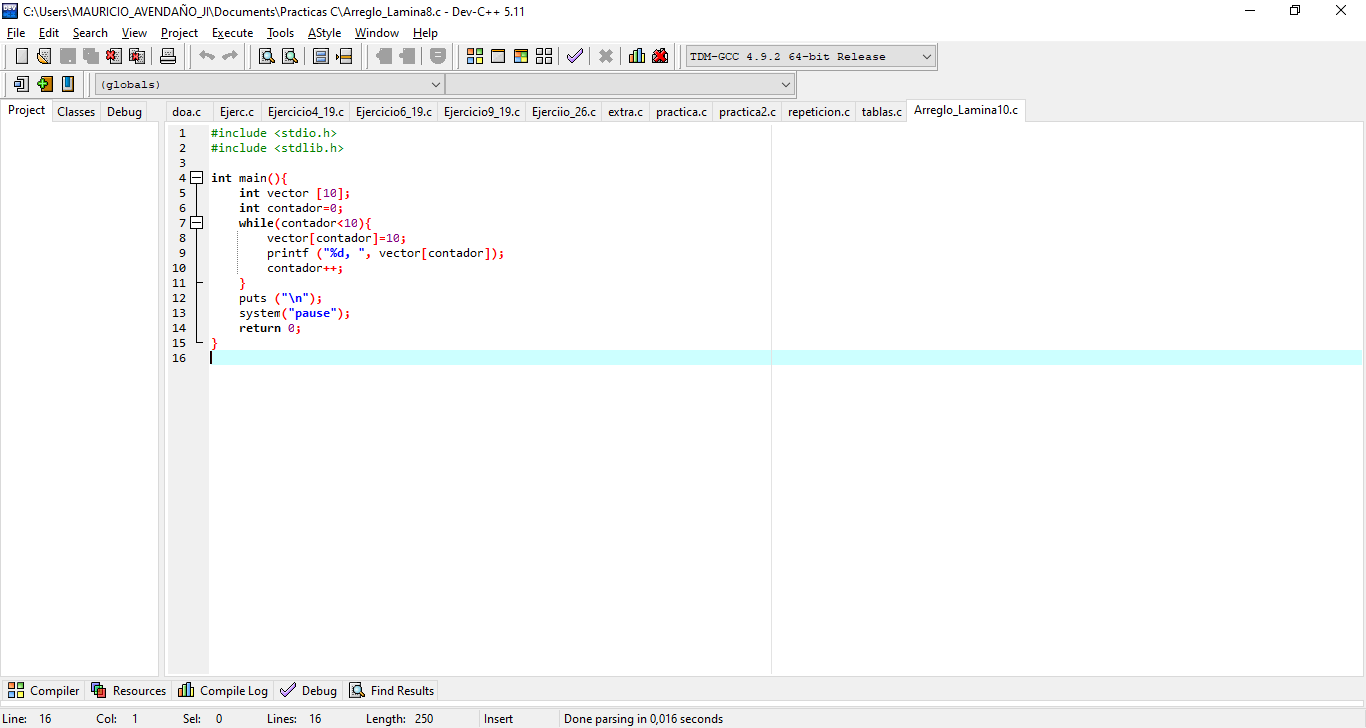
Declaración de un arreglo que tendrá valores de tipo entero, llamado miArreglo destinado a recibir 8 valores.

Valor que devuelve el método principal

Bibliotecas



Ejercicio. Arreglos. Crear un programa que declare un arreglo llamado “vector” de 10 posiciones, asignar el valor de 10 a cada posicion del arreglo y mostrar todas las posiciones del arreglo.



Para llenar el arreglo se hace uso del ciclo while, la condición es que el contador sea menor a 10, ya que recordemos que iniciamos en 0, luego a cada posición del arreglo respecto al valor del contador le asignamos un 10 como valor, y enseguida imprimimos esa posición, que se asigna al especificador de conversión, cade decir que no se pone 0, 1 o 2, etc., porque el contador hace referencia al valor de cada posición.

Valor que devuelve el método principal

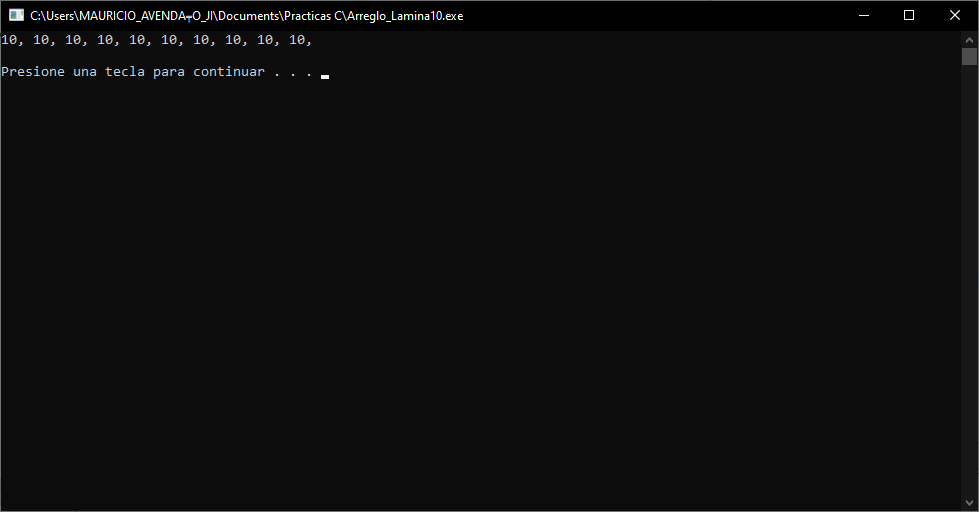
Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Incremento

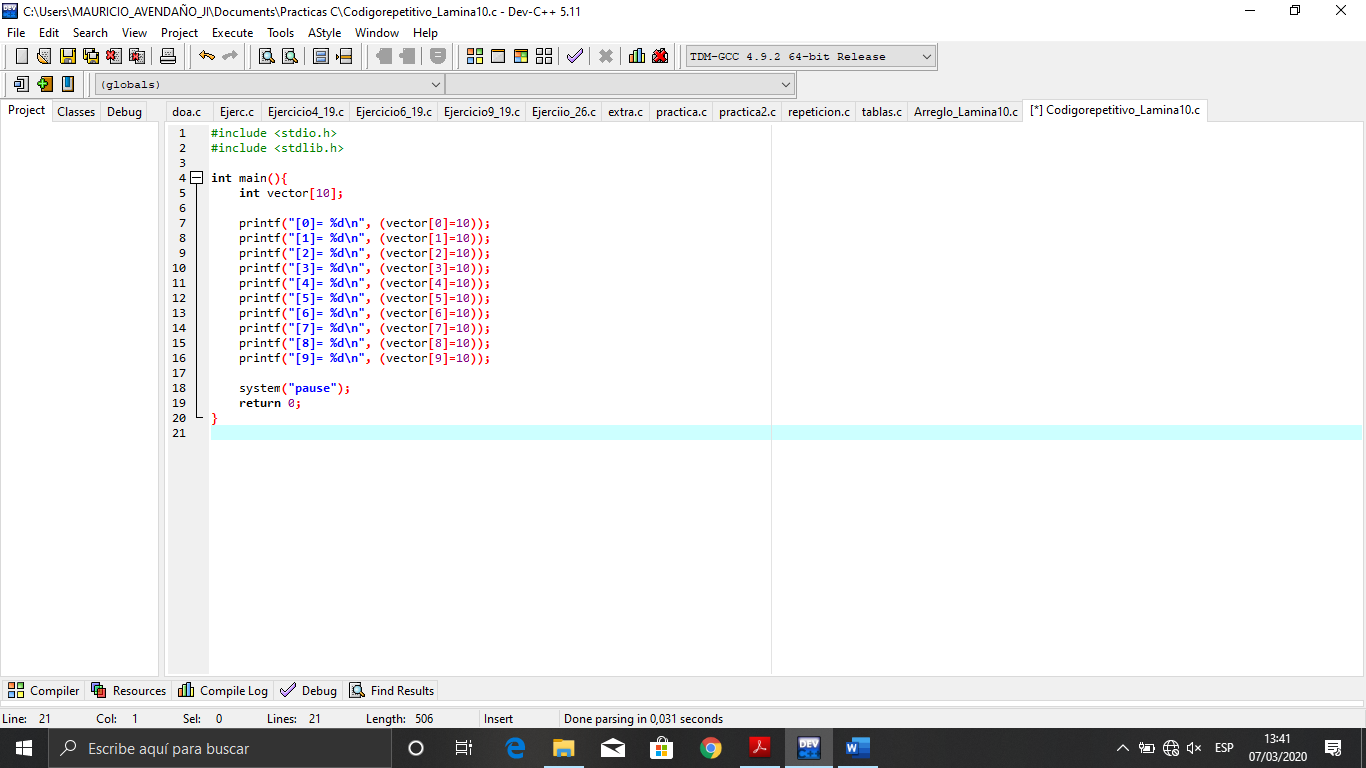
Bibliotecas

Declaración de un arreglo de tipo entero llamado vector con 10 posiciones, y una variable de tipo entero llamada contador inicializada en 0 para el ciclo while.

Método principal



Otra manera de hacer ejercicio es asignando a cada posición el valor indicado, pero esta forma no es recomendada, puesto que se lleva más tiempo, más líneas de código y además si planteamos que fuesen 100 posiciones, cuanto tiempo nos llevaría llenarlo



Declaración del arreglo llamado vector con 10 posiciones, este arreglo almacenará datos de tipo entero.

Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

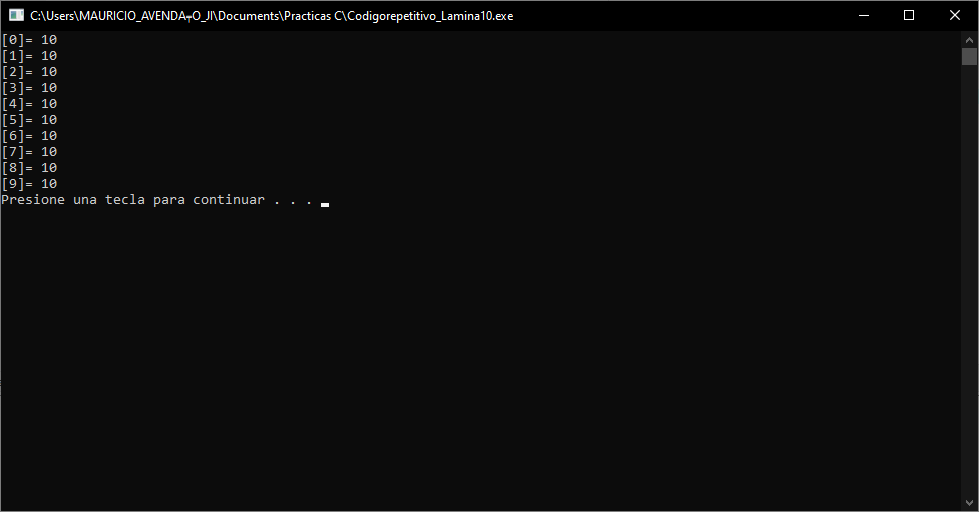
En esta manera, llenamos el arreglo por cada posición, y ahí mismo los mostramos, el valor de 10 se lo asignamos a la posición (0, 1, 2, 3, etc.), luego el valor de esa posición se asigna al especificador de conversión y se muestra en pantalla.

**%d** especificador de conversión de tipo entero

Valor que devuelve el método principal

Bibliotecas

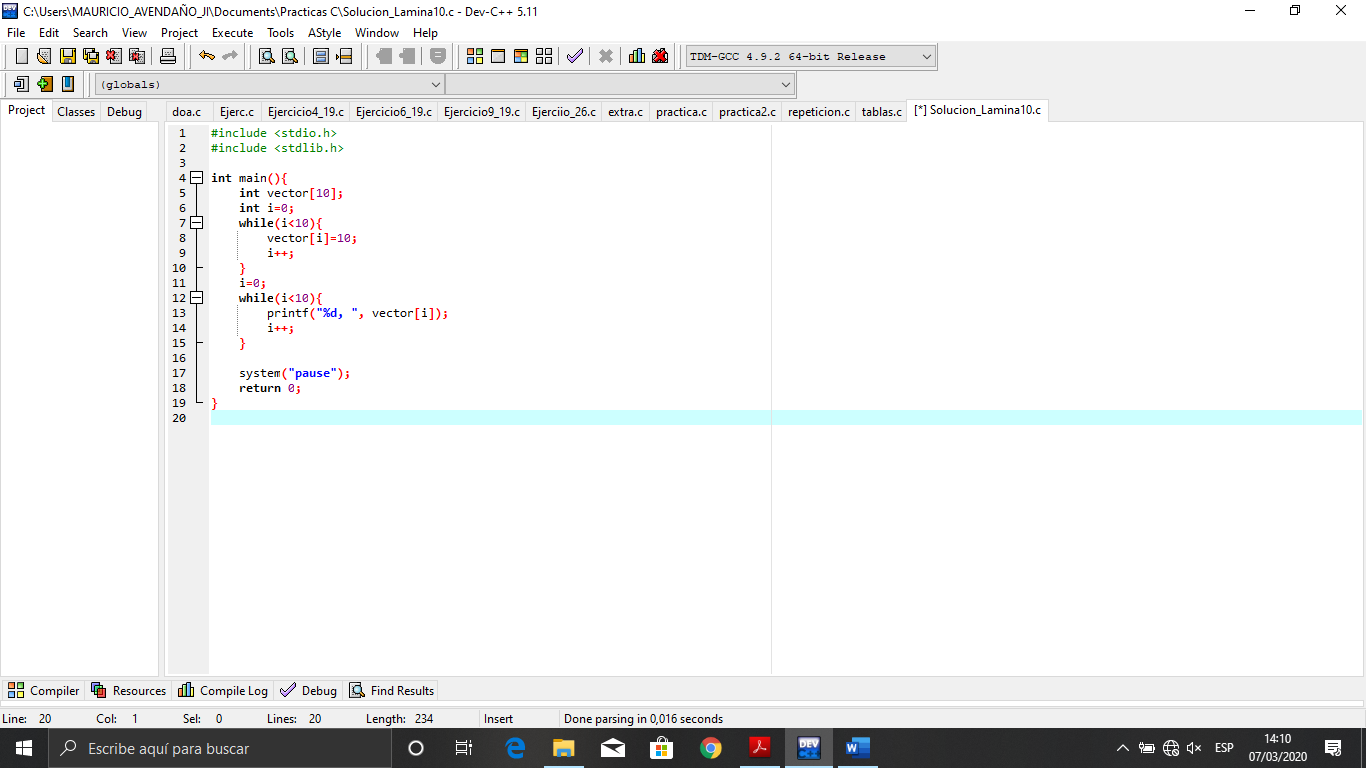
Método principal



Se hace referencia a la posición del arreglo

Se hace referencia al valor de la posición del arreglo

Se presenta otra solución, parecida a la primera, solo que en esta se utilizan dos ciclos, uno para el llenado del arreglo y otro para recorrer el arreglo e ir mostrando los valores de cada posición en pantalla.



Declaración del arreglo llamado vector con 10 posiciones para datos de tipo entero, y una variable i de tipo entero iniciada en 0 para los ciclos.

El primer ciclo es para el llenado del arreglo, inicia en 0, porque las posiciones de un arreglo inician así, y cada vez que va incrementando se le va asignando el valor de 10, en el segundo ciclo recorremos las mismas posiciones, pero ahora vamos mostrando en pantalla el valor de cada posición, y así va a ir incrementando i hasta salir del ciclo, el valor se asigna al especificador de acceso para que pueda imprimirse en pantalla. **%d** especificador de conversión de tipo entero

Valor que devuelve el método principal

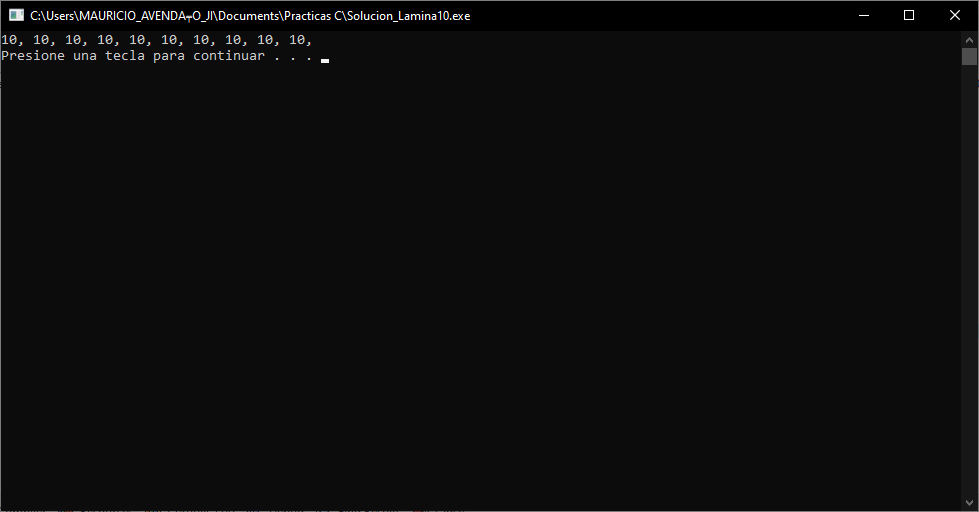
Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Incremento

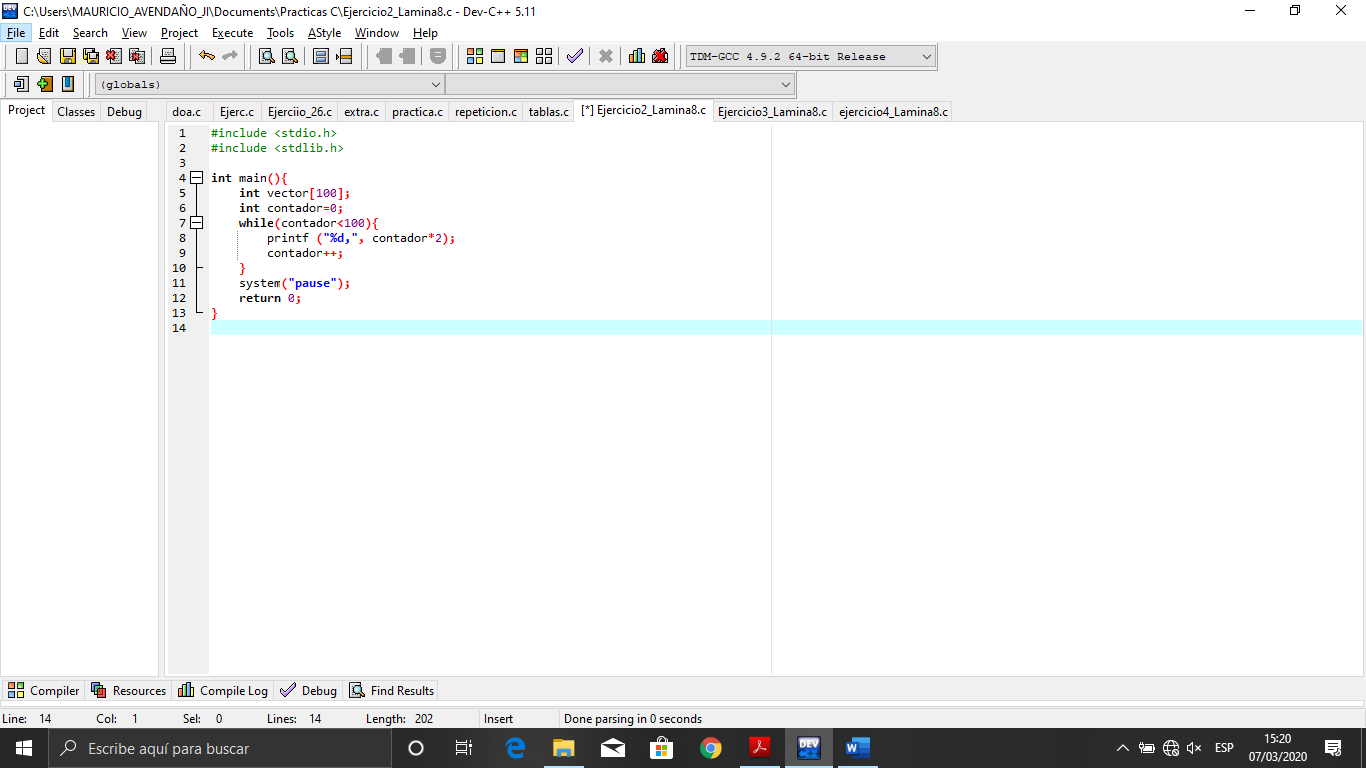
Incremento

Método principal

Bibliotecas



Ejercicio. Arreglos. Crear un arreglo de 100 posiciones, llenar el arreglo con la tabla del 2 y mostrar el arreglo en pantalla.



Este arreglo comienza en 0 porque recordemos que los arreglos inician desde esa posición, por cada vez que el ciclo repita, el valor dl contador se va a multiplicar x2, y se va a asignar al especificador de conversión para mostrarlo en pantalla, después el contador incrementa y hace lo mismo, de etsa manera se ejecuta hasta que la condición ya no se cumpla.

**%d** especificador de conversión de tipo entero

Declaración del arreglo llamado vector con 10o posiciones para datos de tipo entero, y una variable contador de tipo entero iniciada en 0 .para los

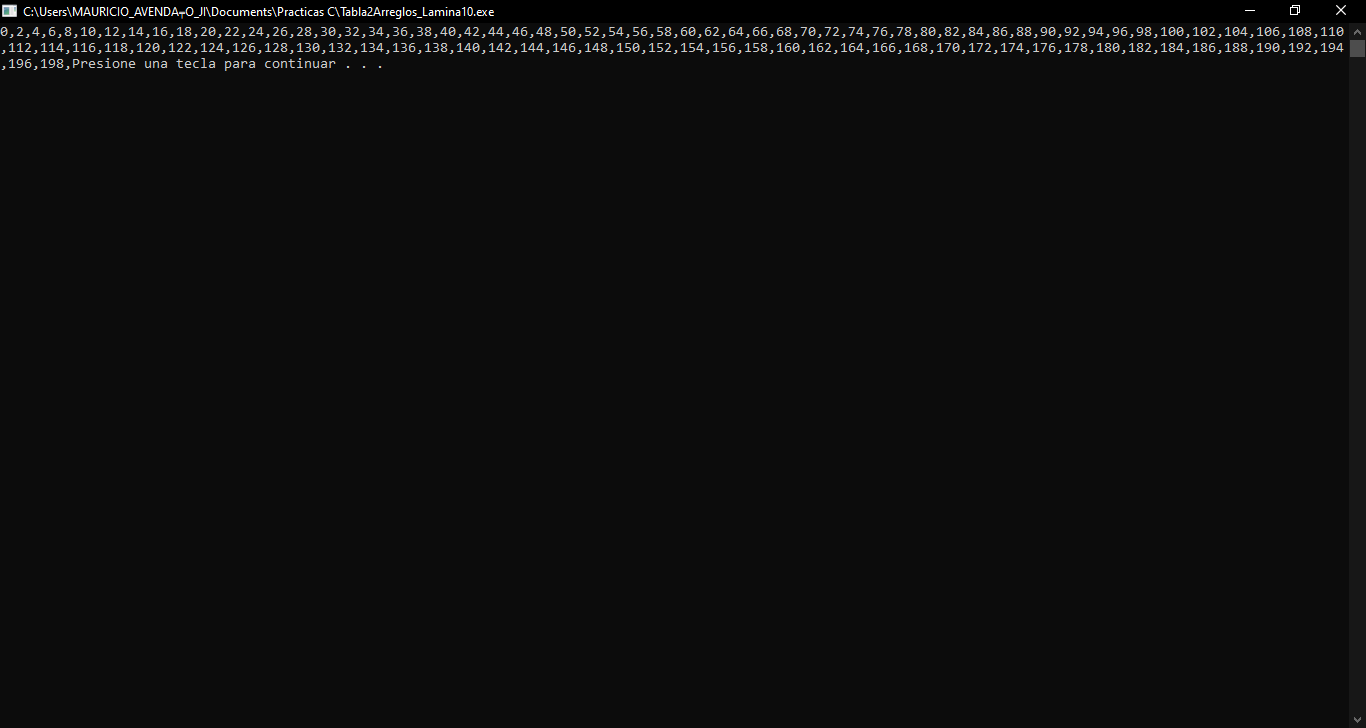
Método principal

Bibliotecas

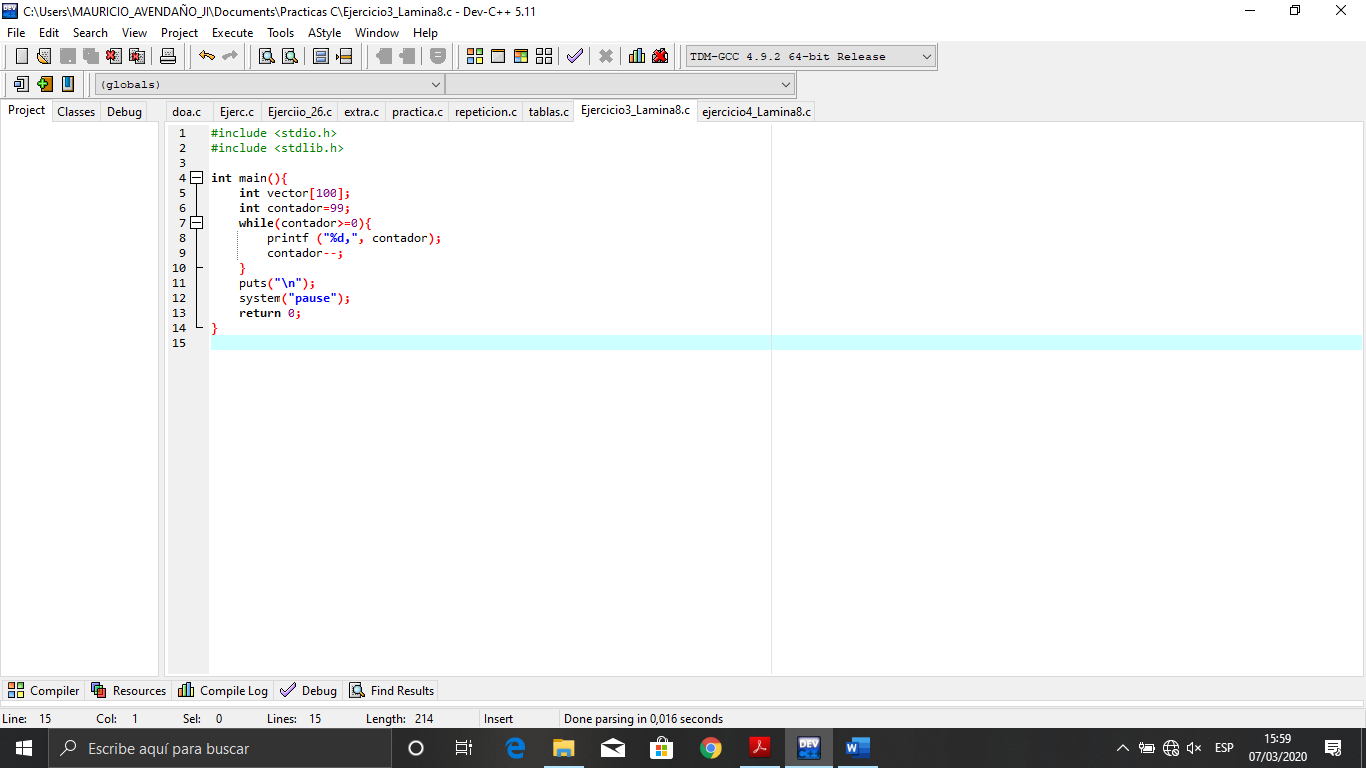
Valor que devuelve el método principal

Incremento

Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

 continua… 

Ejercicio. Arreglos. Crea un arreglo de 100 posiciones, llenar el orden inverso al índice y mostrar el arreglo en pantalla.



Declaración del vector de tipo entero con 100 posiciones, y un contador igual de tipo int con un valor de 99.

En el ciclo while la condición es que sea mayor o igual a 0, porque como el ejercicio nos pide el orden descendente tenemos que ir decrementando en vez de incrementar como en los otros ejercicios. Cada vez que la condición entre en el ciclo, el contador va a ir decrementando, es decir, 99, 98…, y el valor resultante se va a asignar al especificador de conversión y se mostrara en pantalla.

**%d** especificador de conversión de tipo entero

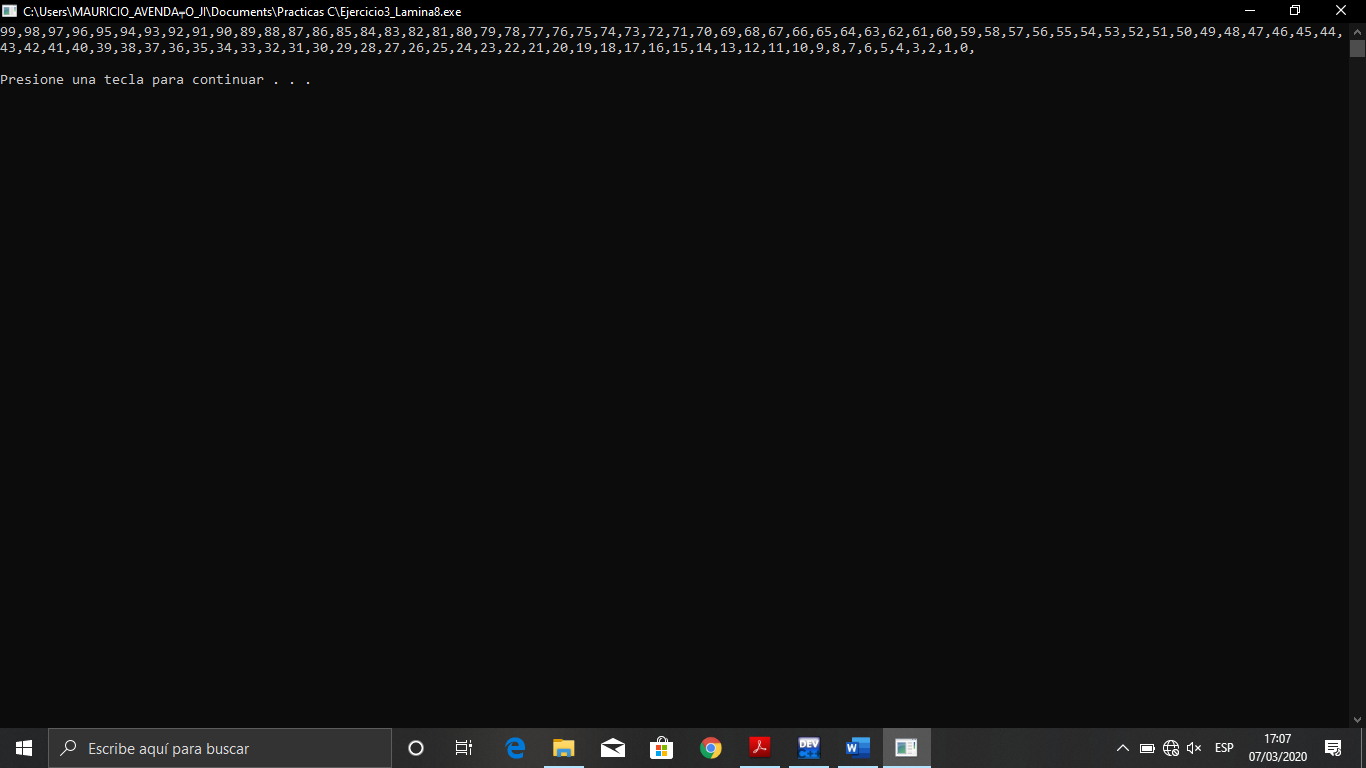
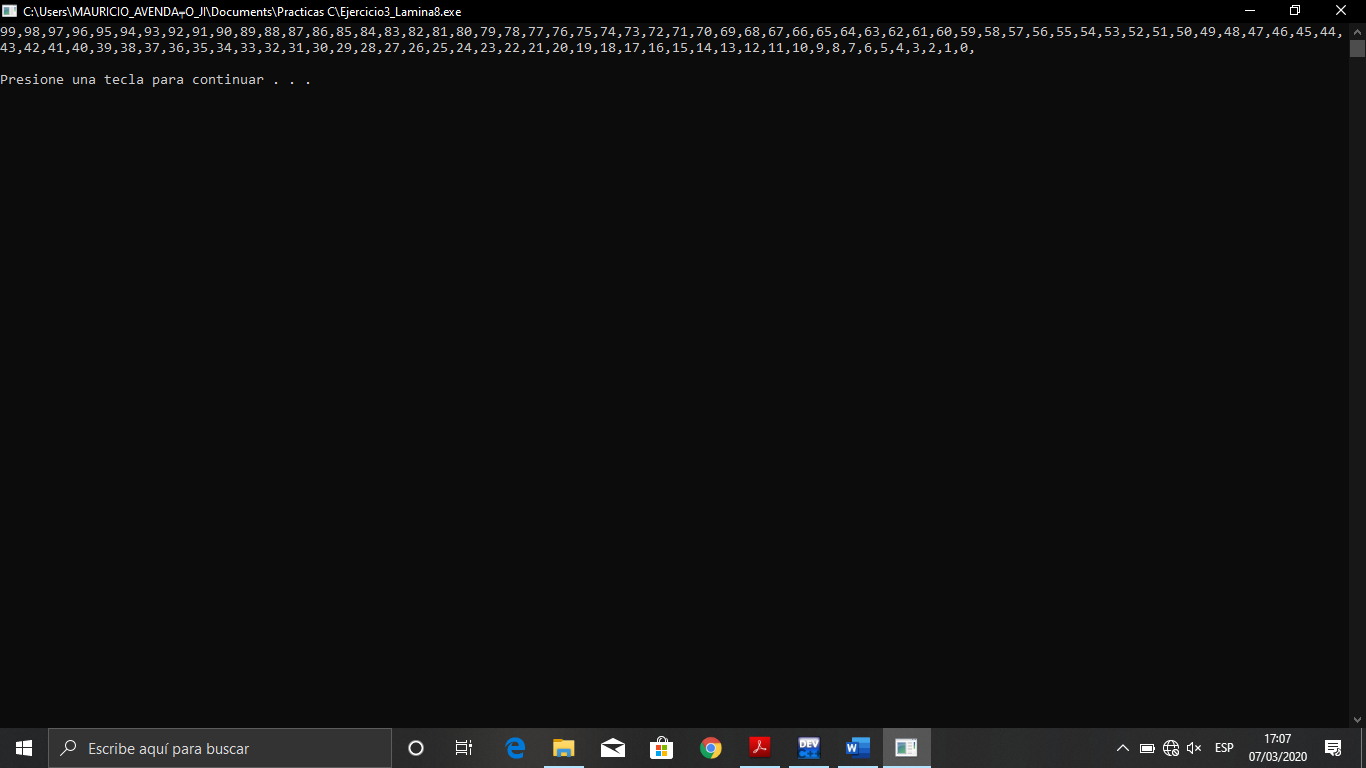
Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Valor que devuelve el método principal

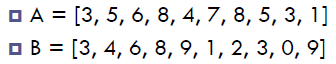
Bibliotecas

Método principal

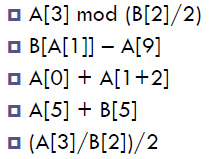
Incremento

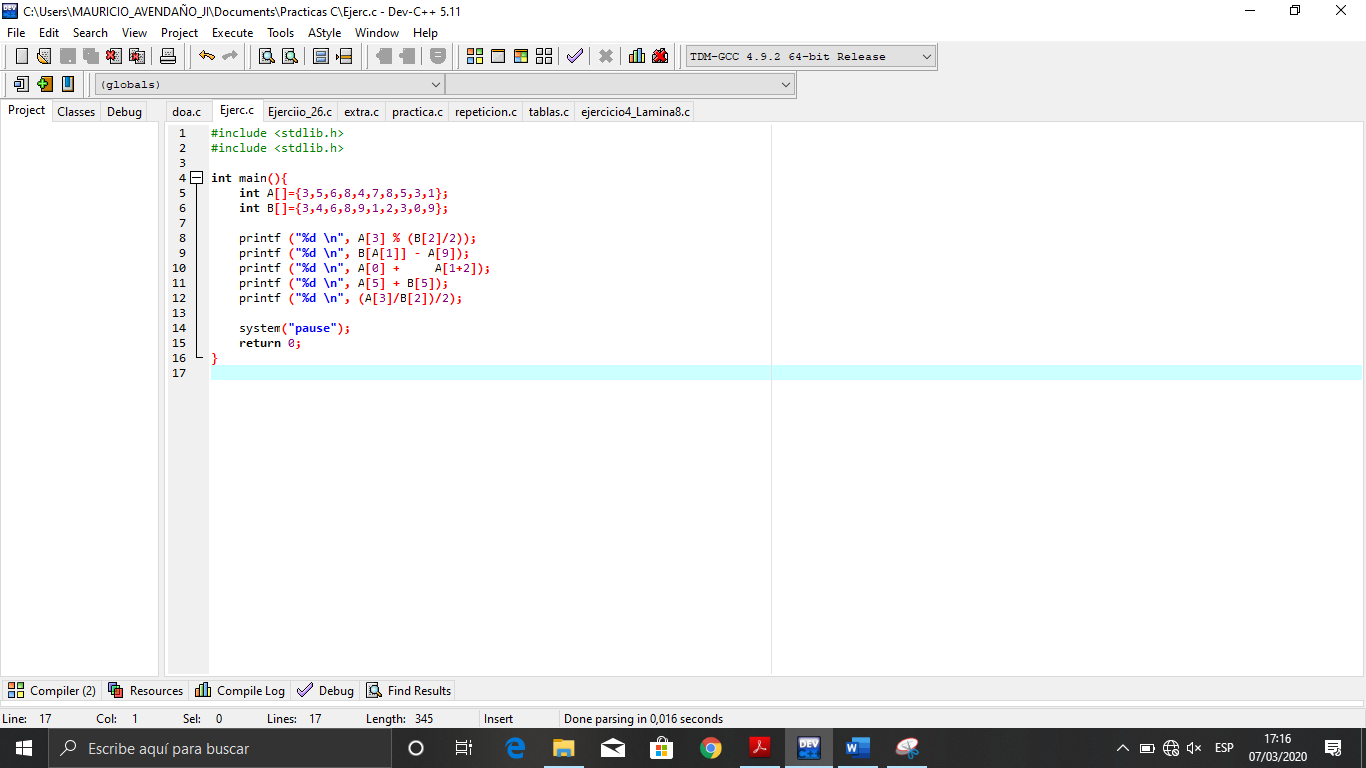
 continua… 

Ejercicio. Arreglos. Dados:



Realizar las siguientes operaciones:





Valor que devuelve el método principal

En este ejercicio, con ayuda de la función printf() mostramos en pantalla el resultado de las operaciones con las posiciones de los arreglos, el resultado, se alojara en el especificador de conversión y será impreso posteriormente, cabe mencionar que hay que tener en cuenta la jerarquía de operadores, puesto que el resultado será diferente si colocamos los signos o () en diferente orden.

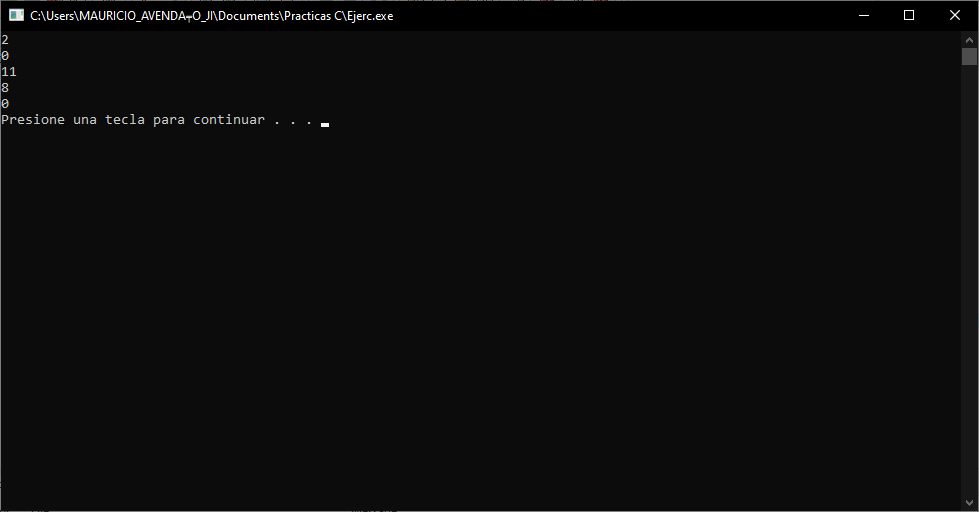
**%d** especificador de conversión de tipo entero

Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Bibliotecas

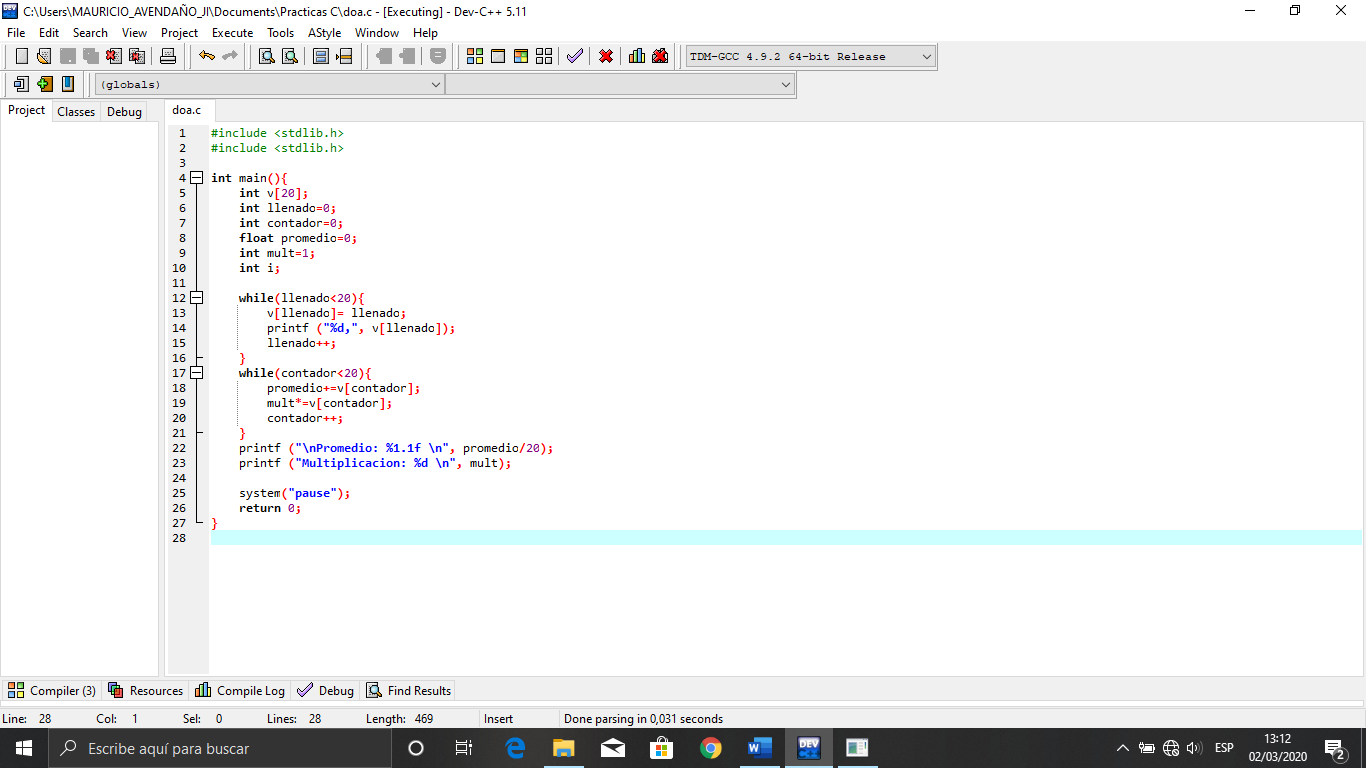
Método principal

Declaración de 2 arreglos, ambos de tipo entero, llamados A y B con 10 elementos cada uno.



Ejercicios. Arreglos. Crea un arreglo de 20 posiciones, asígnale a cada elemento un valor. Calcula el promedio de todos los elementos y calcula la multiplicación de todos los elementos.

En el primer ciclo hacemos el llenado del arreglo, con los valores que va a adoptando la misma variable, por ejemplo, en la posición 1 tiene el 0 y cuando incrementa en la posición 1 tiene el 1 y así sucesivamente hasta completarse. En el segundo ciclo recorremos el arreglo y vamos sumando y multiplicando el valor de cada posición, al último sacamos el promedio, e imprimimos los resultados.



Declaración de las variables a utilizar todas excepto una, son de tipo entero, la otra es de tipo flotante, casi todas con valor inicial 0 porque como estamos hablando de arreglos, empieza a contar desde l posición 0, menos la de mult, que tiene que multiplicarse con 1, si fuera 0, el resultado total fuese el mismo, y sería algo incorrecto,

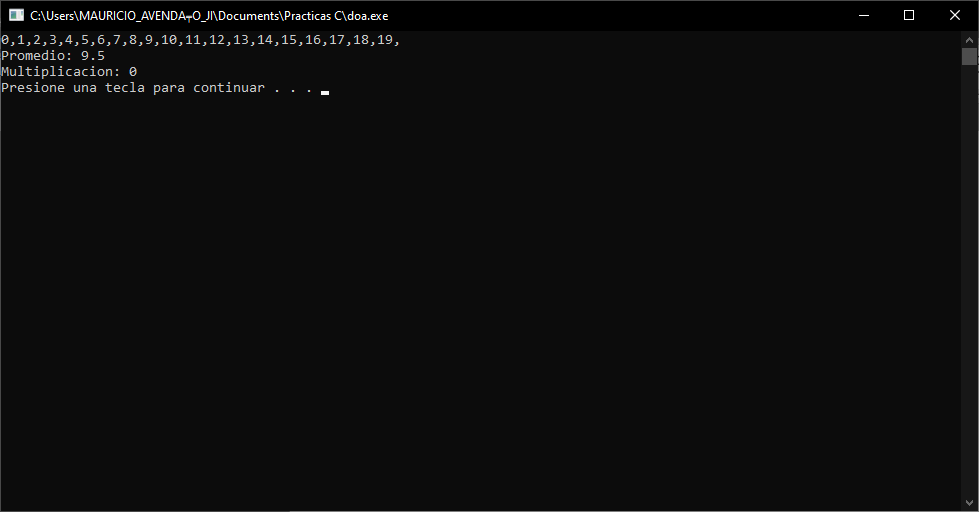
Valor que devuelve el método principal

Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Mandamos una impresión en pantalla con los resultados de la multiplicación de todos los valores, y el promedio de ellos, el valor de cada variable asigna al especificador de acceso, y el programa da por terminado

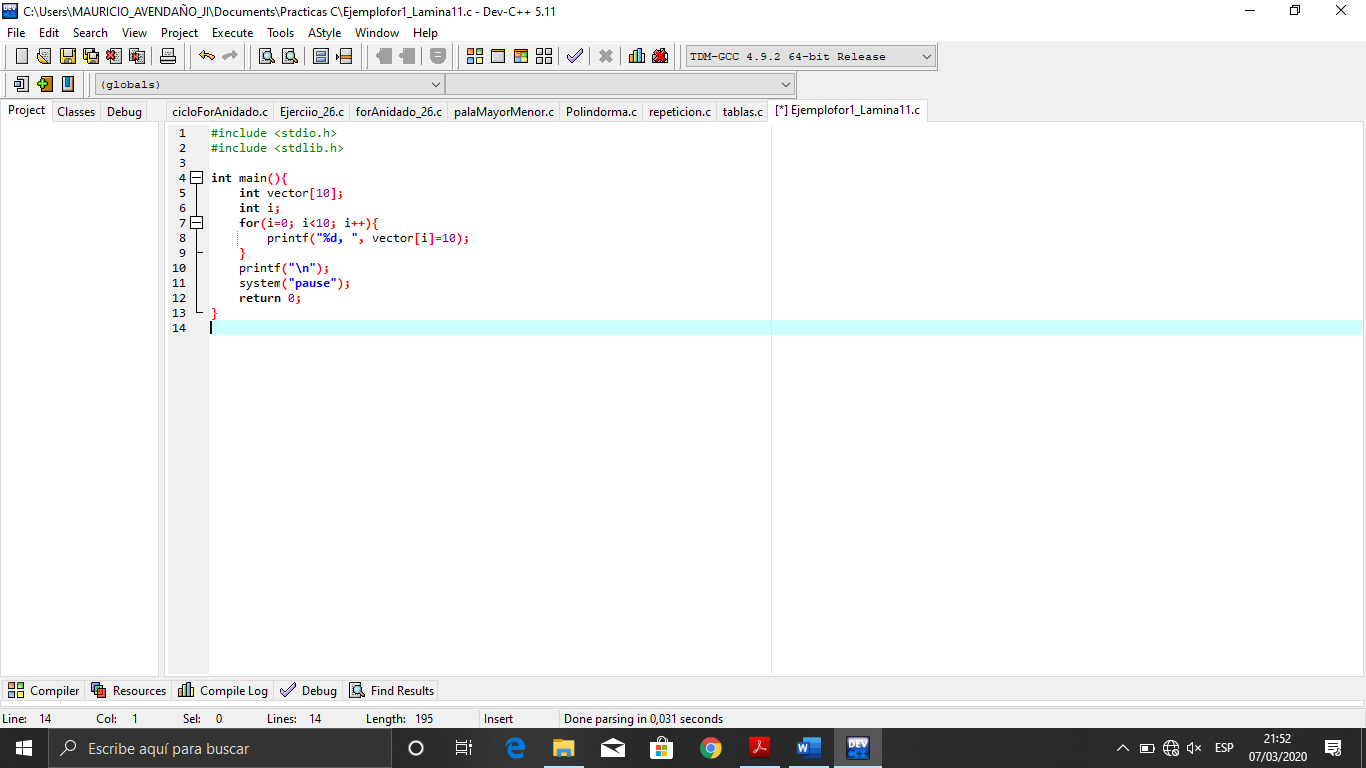
Bibliotecas

Método principal



**Documento 11**

Ejemplo. Estructura repetitiva for



Valor que devuelve el método principal

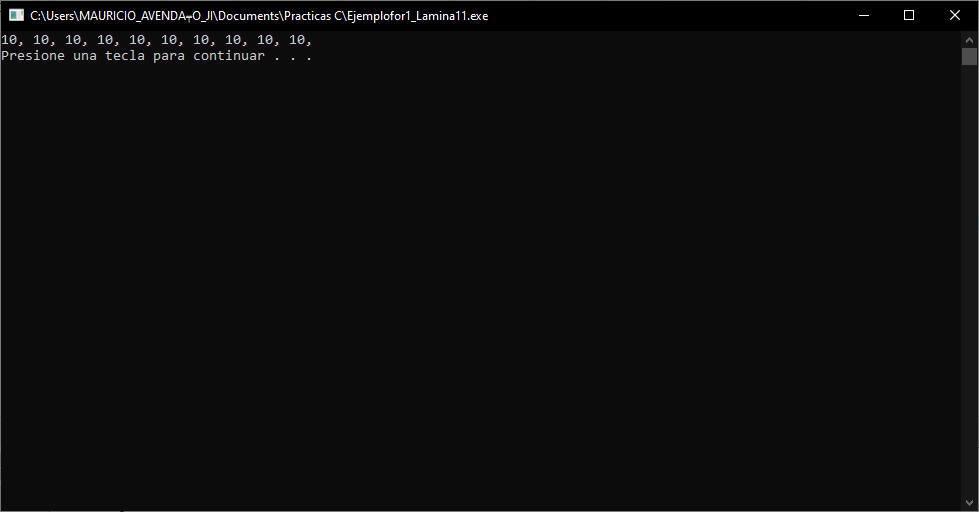
Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Declaración de 1 arreglo llamado vector de tipo entero con un tamaño de 10 y una variable i de tipo entero para el ciclo.

La función del ciclo for es similar al while, tiene una inicialización, una condición y un incremento, inicia en 0 por las características del arreglo, imprime la posición del arreglo con respecto a i, es decir, cuando i vale 0 hacemos referencia a la posición 0 del arreglo, cuando incrementa y vale 1, ahora la posición es 1, y así sucesivamente va a incrementar el valor de i hasta que la condición no se cumpla, primero i vale 0, cuestiona si 0 es menor a 10, como es verdadero imprime la posición con el valor de 10, primero se lo asigna a la posición y luego lo muestra, cuando incrementa ahora vale 1 y pregunta nuevamente, esto se repite hasta que i no sea menor a 10.

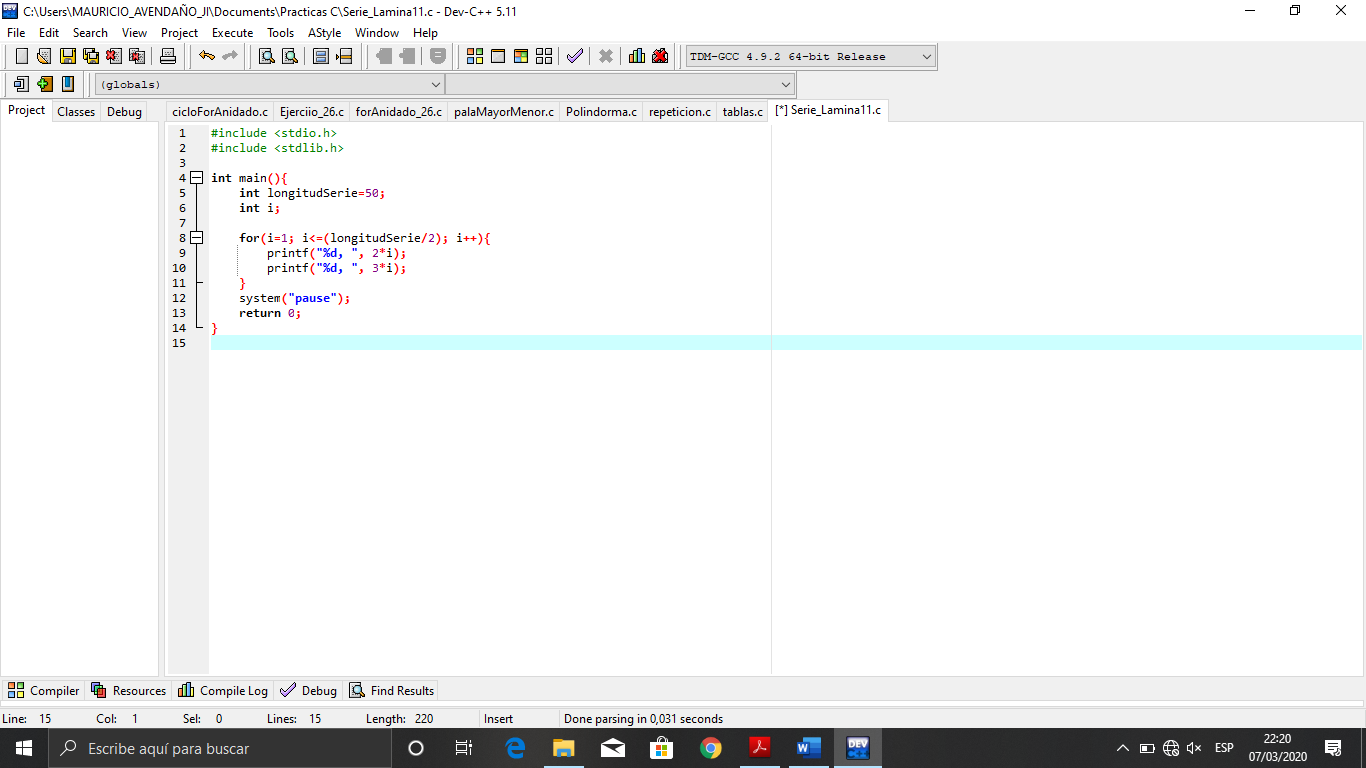
Bibliotecas

Método principal



Ejemplo. For. Imprima la serie 2, 3, 4, 6, 6, 9, 8, 12, 10, …

Método principal



Valor que devuelve el método principal

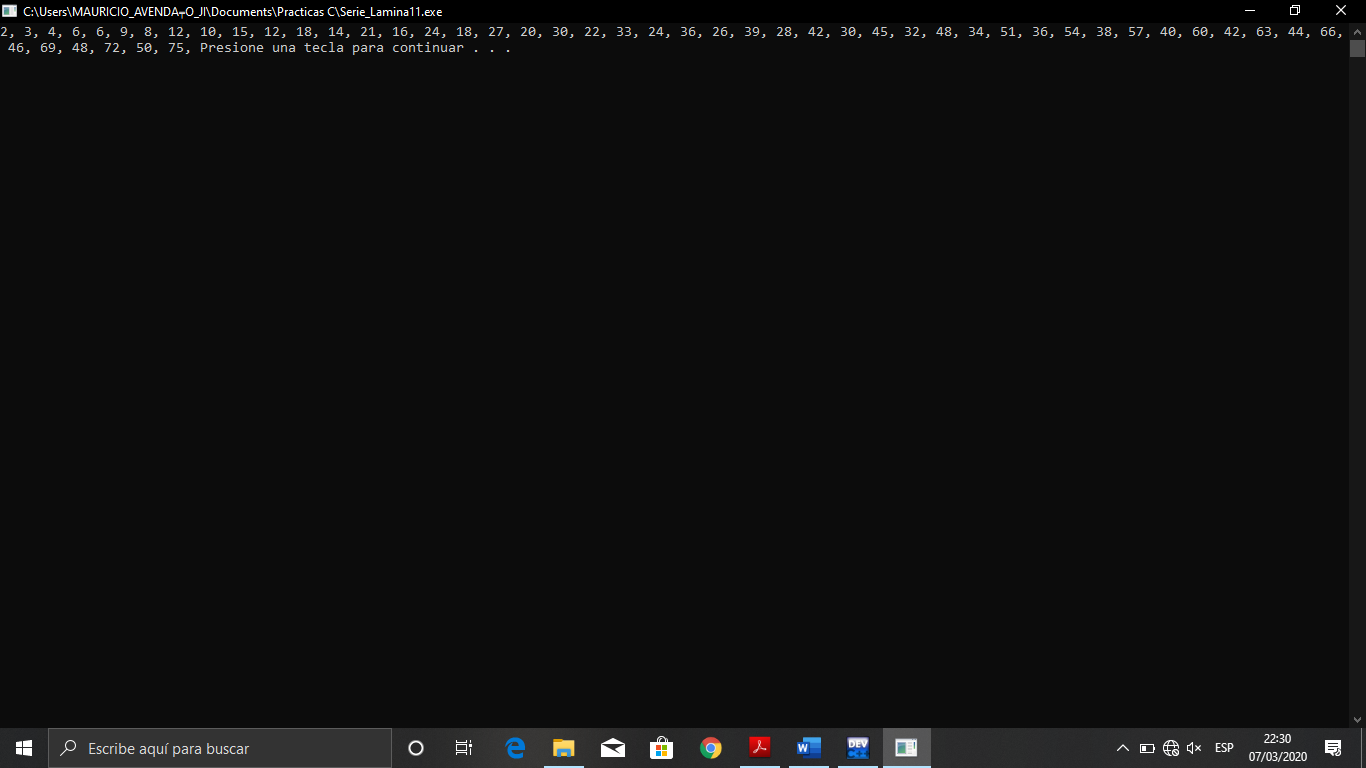
Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Este ciclo va a imprimir el valor de i multiplicado por 2 y por 3, cada uno por separado, el resultado de esas operaciones se asigna al especificador de conversión que espera un dato de tipo entero y se muestra en la pantalla.

%d Especificador de conversión

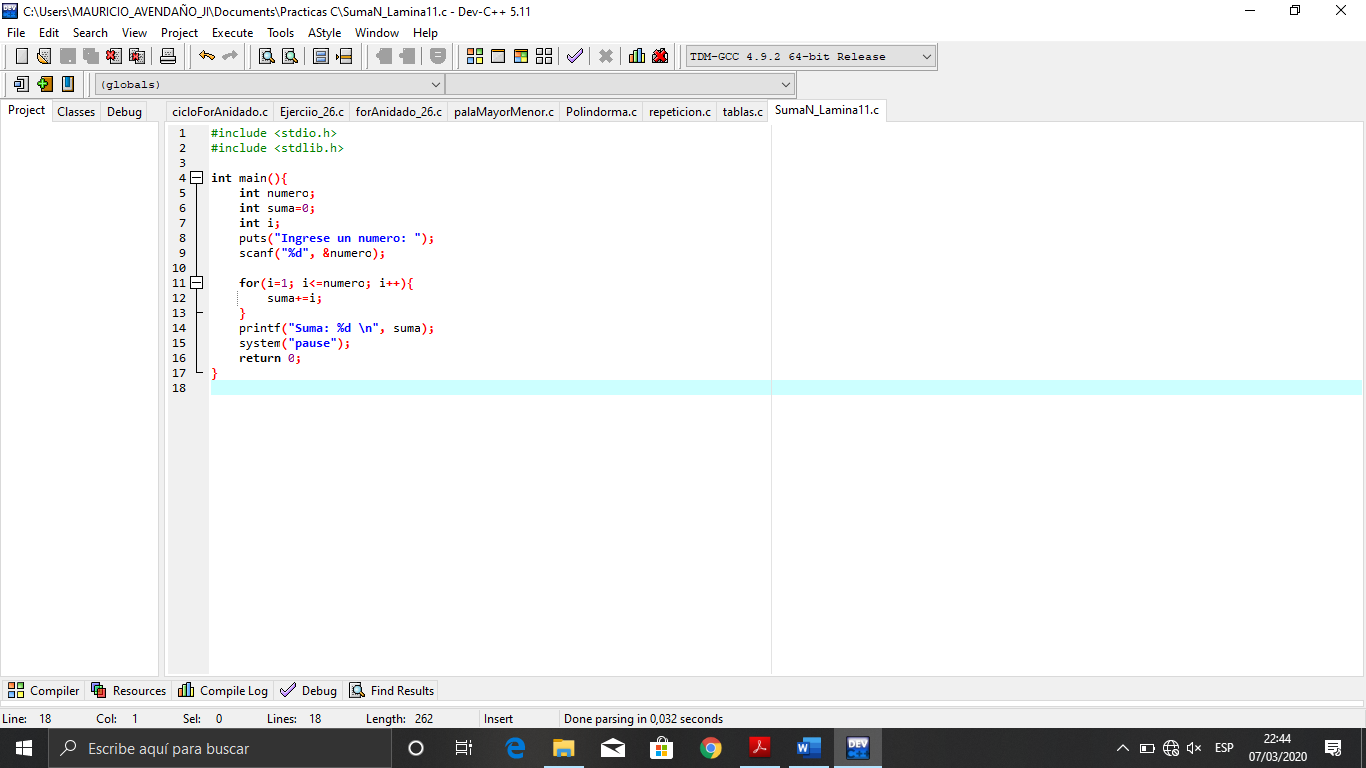
Declaración de 2 variables, ambas de tipo entero, la 1 llamada longitudSerie con un valor inicial de 50, y la otra llamada i.

Bibliotecas



Continua…

Ejercicio. For. Escribe un programa que recibe un número N del usuario y haga la suma de todos los números desde 1 hasta N. ejemplo: 5 ==> 1+2+3+4+5=15



Declaración de 3 variables una llamada número, la siguiente suma iniciada en 0 e i todas de tipo entero.

El ciclo inicia en 1 como el ejercicio lo indica, la suma está en 0 porque aún no se ha sumado nada, cuando el ciclo inicia, se suma el valor de i que es 1, incrementa y ahora vale 2, ese 2 se suma al valor que ya tenía, es decir, 1+2=3, incrementa i, ahora vale 3, suma de nuevo, 3+3=6 y así sucesivamente hasta que el ciclo termine. Cuando el ciclo finaliza imprimimos el valor de la suma resultante y el programa hará terminado.

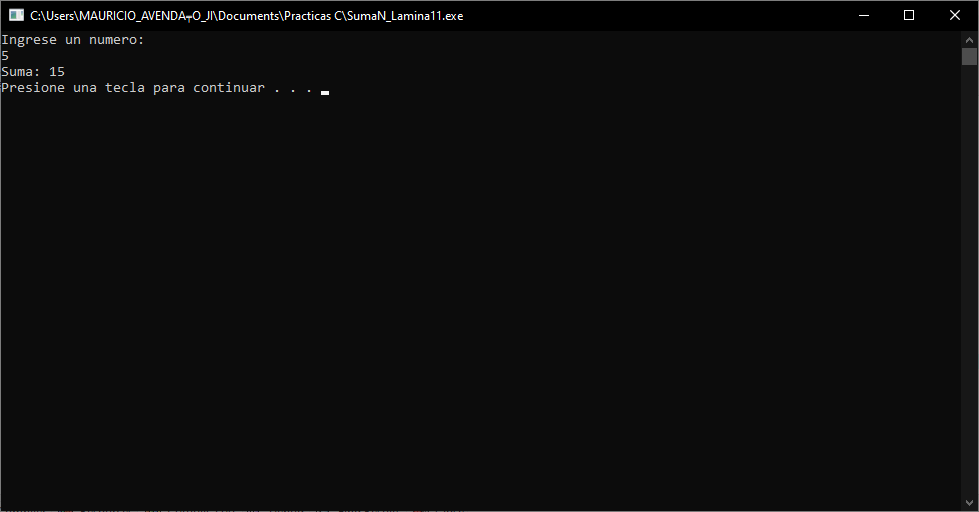
Bibliotecas

Método principal

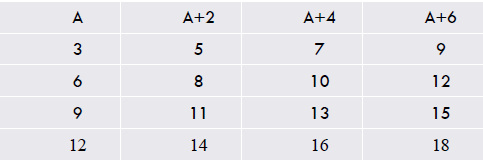
La función puts() nos sirve para mostrar un mensaje en pantalla con alguna instrucción que deseemos dar al usuario, con scanf() obtenemos por teclado el valor que vamos a ocupar, el valor se asigna por medio del especificador de acceso a la variable, en este caso, numero.

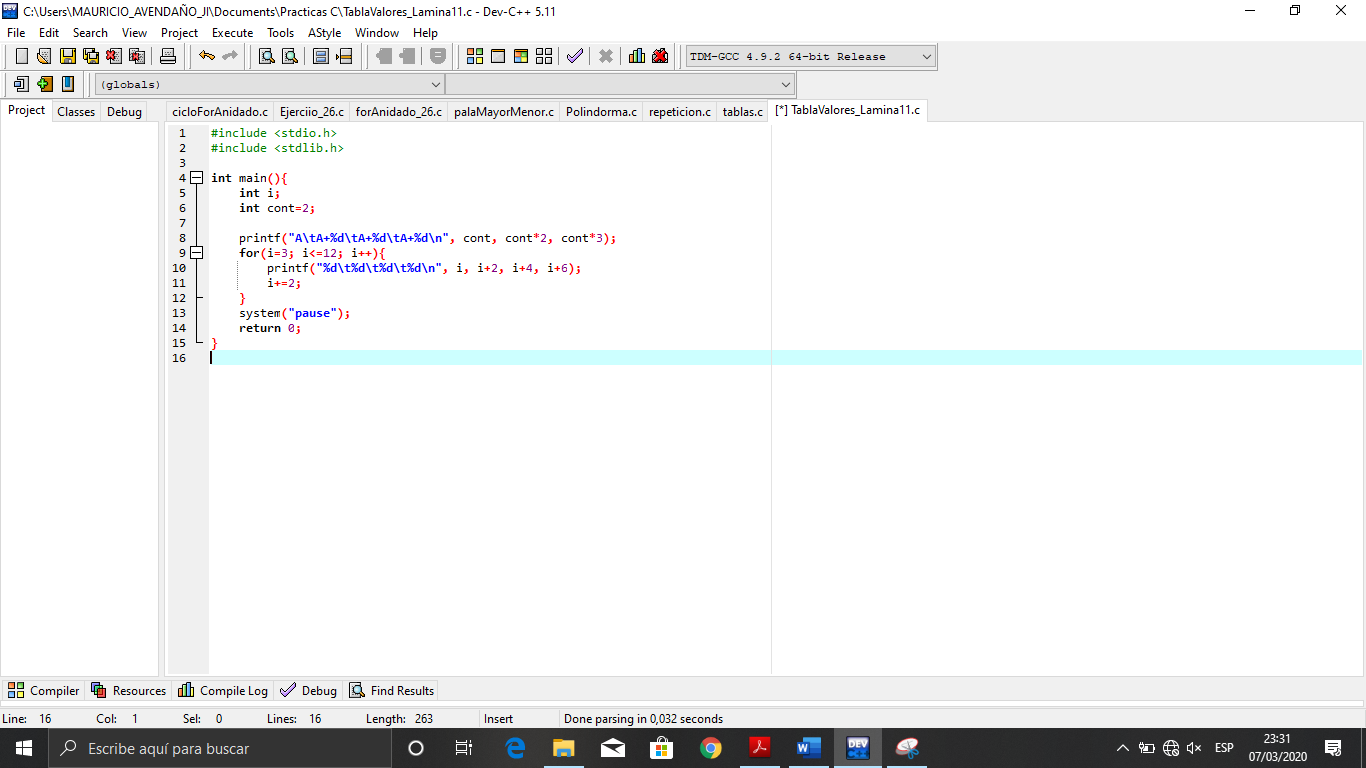
Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Valor que devuelve el método principal



Ejercicio. For. Escriba un programa en C que utilice un ciclo para producir la siguiente tabla de valores.





Declaración de 2 variables, ambas de tipo entero, una llamada i correspondiente a la inicialización del ciclo y la otra un contador con un valor inicial de 2.

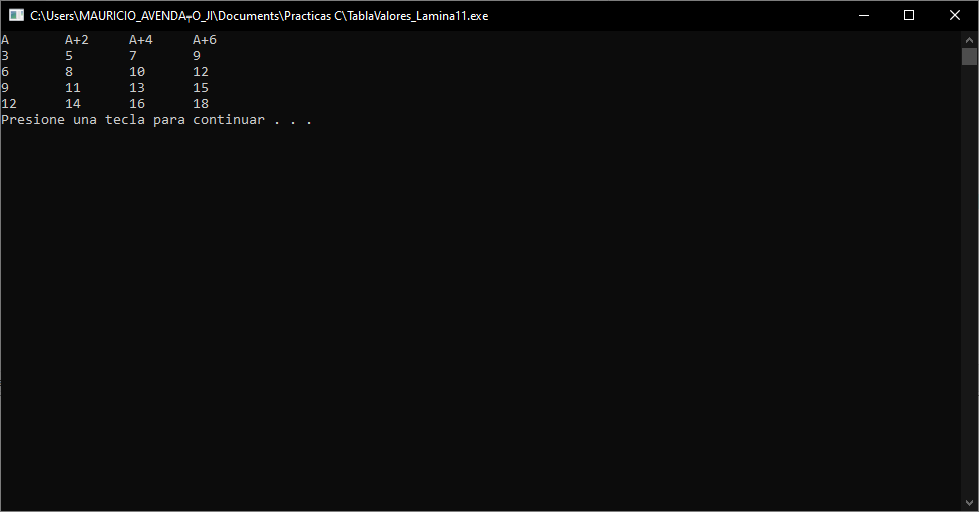
Bibliotecas

Lo primero que hacemos es imprimir el encabezado de la tabla, para ello creamos un contador que va a imprimir los números que acompañan a las letras, primero es A, después A+2, A+4, A+6, el contador es el que va a operar con el 2, 4 y 6, solo se multiplica por 2 y 3, para que salgan esos resultados. El ciclo corresponde a imprimir los números, este igual se multiplica por el número que dice el encabezado y se incrementa en 2, porque un incremento que ya tenía son 3, esto es para imprimir los valores de la primera columna, son múltiplos de 3, cada resultado se asigna a un especificador de conversión en este caso de tipo entero (%d)

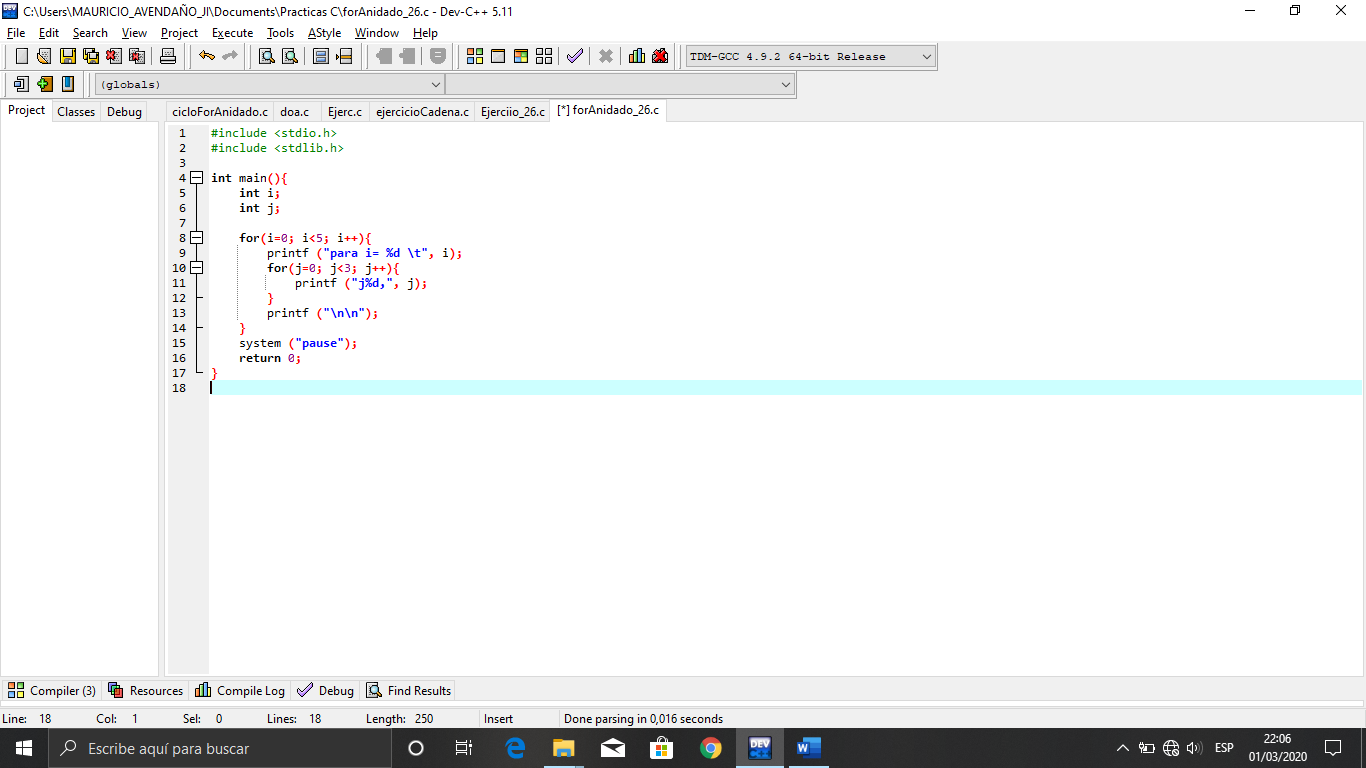
Valor que devuelve el método principal

Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Método principal



Ejemplo. For anidado.



Esto es un ciclo anidado, el ciclo externo inicia en 0 y el segundo igual, cuando entra al primer ciclo imprime el valor de i, este se lo asigna al especificador de acceso, cuando entra al segundo imprime el valor de j, pero este va a ir cambiando cuando vaya incrementando el ciclo interno, pero este va a ser igual que para el resto de las vueltas del ciclo externo. Esto va a finalizar hasta que ya no se cumpla la condición del ciclo externo.

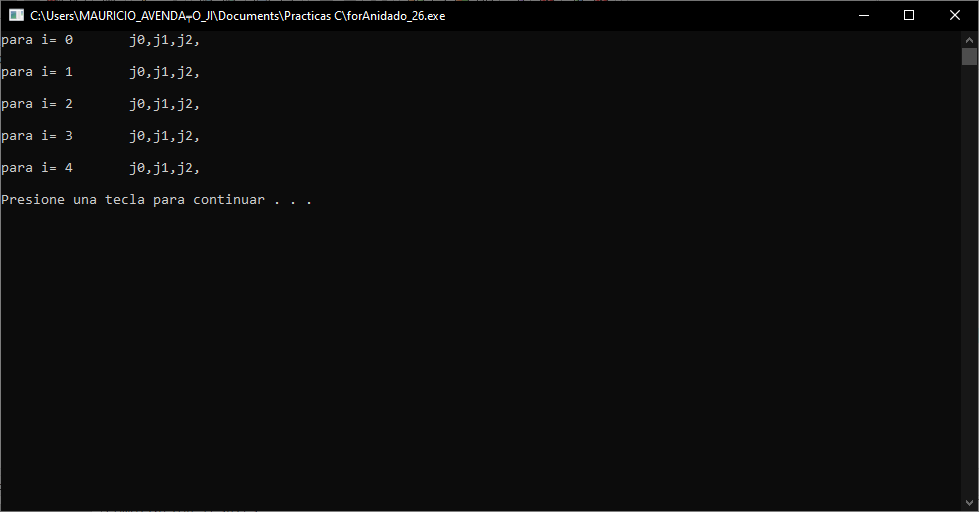
Valor que devuelve el método principal

Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

Declaración de las variables i y j ambas de tipo entero.

Bibliotecas

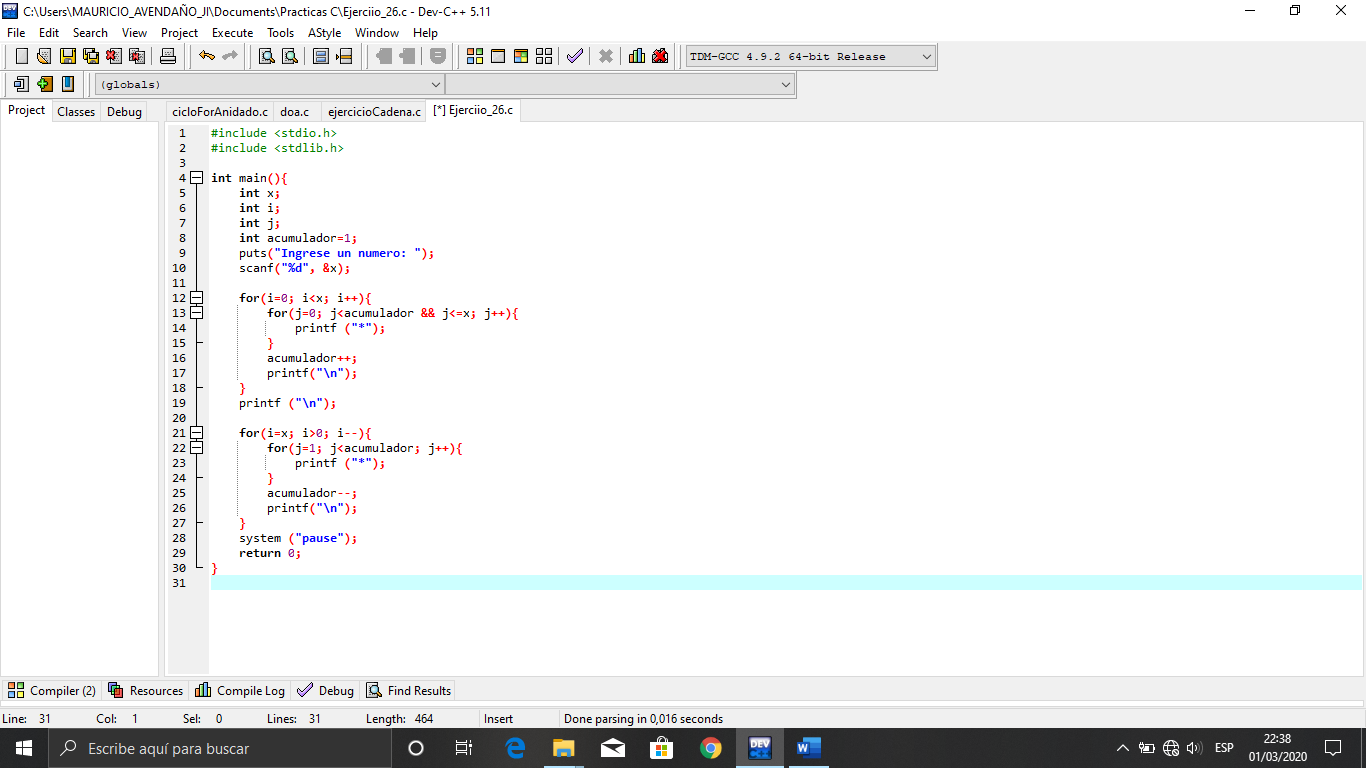
Método principal



Ejercicio. for anidado. Escriba un programa que re4ciba un numero entero N del usuario e imprima una escalinata de N pisos de asteriscos. Ejemplo: Si ingreso un 4.



Bibliotecas



Valor que devuelve el método principal

Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

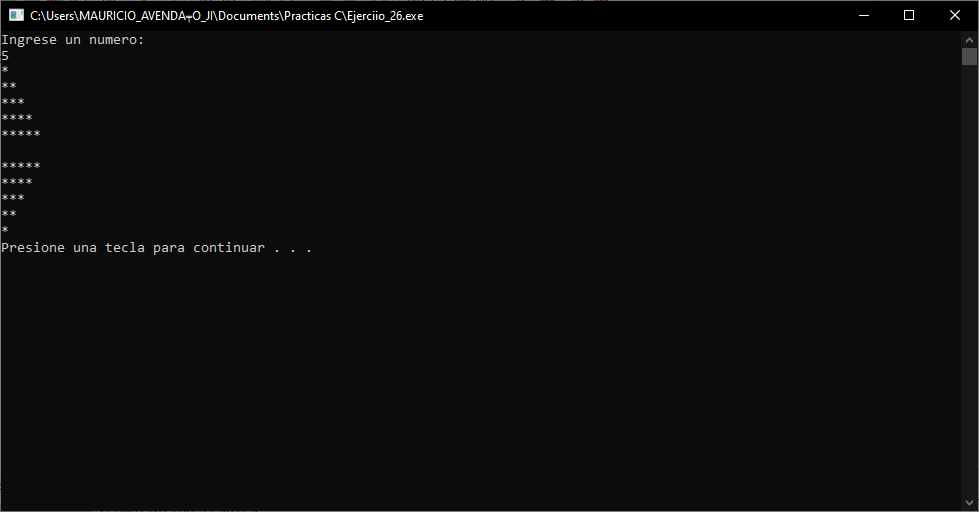
La función de estos ciclos es imprimir un triángulo de asteriscos, para eso contamos con un acumulador que va a ir aumentando en 1 el número de asteriscos para imprimir hasta su límite.

De la misma manera se efectúa el segundo ciclo anidado, pero este será en orden descendente, es decir, en vez de empezar a imprimir 1 asterisco, empezara a imprimir desde el no. Que se ingresó hasta llegar a 1.

Declaración de las variables x, i, j y acumulador, inicializado en 1, este adopta este valor para empezar a imprimir 1 asterisco.

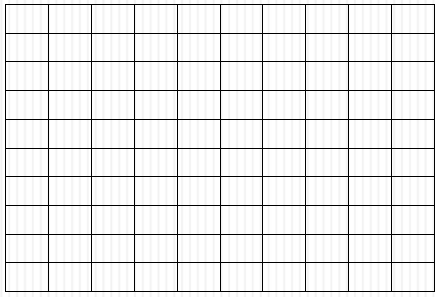
Método principal

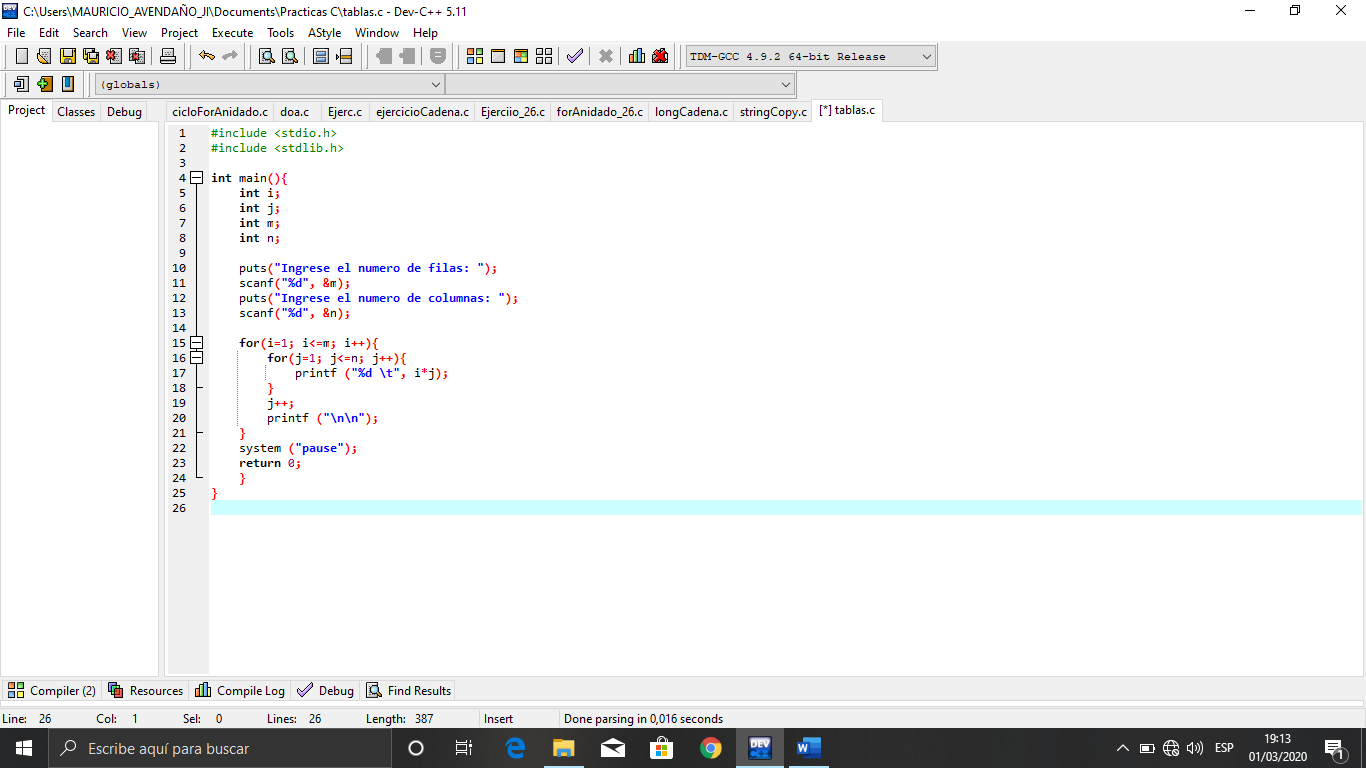
Hacemos uso de puts() para mandar un mensaje en pantalla, dando la instrucción al usuario, y scanf() para que pueda ingresar el dato que necesitamos, este datos se asigna l especificador de conversión y posteriormente lo utilizaremos.



Dato ingresado por teclado, este se va a tomar en cuenta para imprimir la altura de dicho triangulo, en el ejemplo se ingresó un 5, entonces primero se imprimirá 1 asterisco, luego 2, 3, y así hasta llegar al número 5, en el segundo, como ya se había mencionado anteriormente, imprimirá desde el número ingresado, en este caso 5, luego 4, 3 y de esta manera hasta el número 1.

Ejercicio. for anidado. Haz un programa que muestre las tablas de multiplicar como se muestra a continuación. M y N son ingresados por el usuario.





Este ciclo anidado sirve para imprimir las tablas en forma de matriz, primero se va a imprimir la primera línea con la tabla del 1, enseguida la del 2, después 3 y así sucesivamente. **\***

Valor que devuelve el método principal

Función que nos permite detener la ejecución del programa y visualizar los resultados en pantalla.

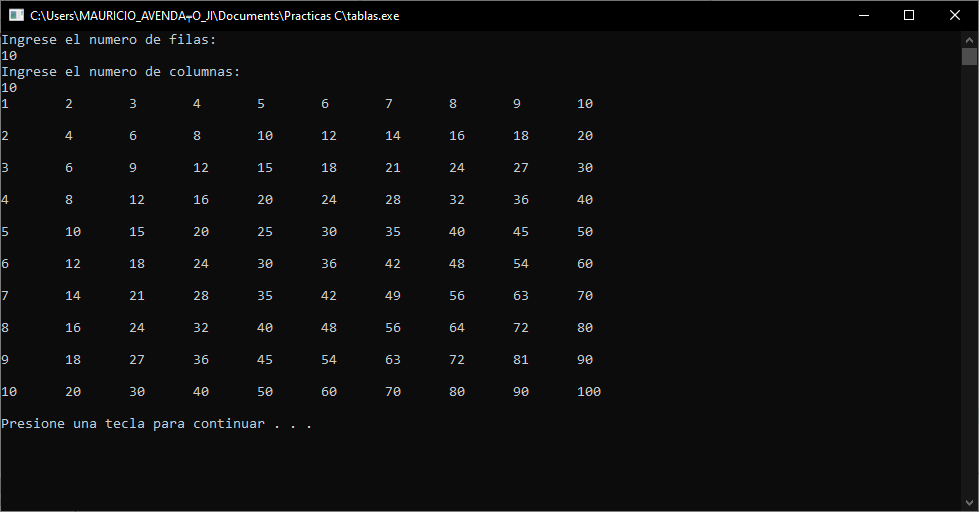
Puts() nos permite mandar un mensaje en pantalla, pero no permite asignar ningún valor de alguna variable, con scanf() podemos ingresar datos por teclado, el valor que se reciba se va a asignar al especificador de conversión dependiendo del tipo de dato a recibir. El primer dato es m, en esta variable se guardan el número de filas y en n el número de columnas.

Declaración de las variables i, j, m, n de tipo entero.

Bibliotecas

Método principal

**\***==> La función de este ciclo anidado es crear una matriz de n \* n, dependiendo del número de filas y columnas que el usuario ingrese, el ciclo externo inicia en 1, cuando ingresa al ciclo interno inicia igual en 1, entonces imprime en este caso la multiplicación de i\*j, es decir, 1\*1 cuando incrementa entra en la condición, y pregunta, 2 es menor o igual a x número de columnas, si es verdadero imprimirá a i\*j, pero ahora 1\*2, luego incrementa e imprime 1\*3 y así sucesivamente hasta completar las columnas para la primera fila, después incrementa el contador del ciclo externo y ahora imprime 2\*1, incrementa y pregunta de nuevo, 2 es menor o igual a x número de columnas, si es verdadero imprime 2\*2 y así sucesivamente, hasta completar las filas que se hayan pedido.



Datos ingresados por teclado, es decir, la cantidad de filas y columnas para la matriz, a partir de ahí se mostrarán los resultados de las tablas de multiplicar hasta los límites, es decir, las filas y las columnas deseadas.

Por ejemplo, se ingresó una matriz de 10\*10, por ende, se mostrarán las tablas de multiplicar del 1 al 10.

**V. Conclusiones:**

Para finalizar puedo decir que el lenguaje C es muy interesante, además te permite desarrollar cosas que posiblemente en otros lenguajes te lleve más tiempo realizar, puesto que este lenguaje te provee de funciones que te hacen más sencilla la programación.

En este manual se presentan las estructuras condicionales simples, así como las dobles, de igual manera las repetitivas como son el while, do\_while y el for y por ultimo los arreglos, que nos permiten almacenar una mayor cantidad de valores, a diferencia de las variables, pero solo de un tipo de dato.

Resulta más fácil trabajar con este tipo de sentencias, por lo que me parece un mejor lenguaje.