|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Ing. Rodríguez Espino Claudia |
| *Asignatura:* | Fundamentos de programación |
| *Grupo:* | 1122, Bloque: 139 |
| *No de Práctica(s):* | Décima práctica |
| *Integrante(s):* | Fonseca Ramírez Gadiel Moisés |
|  |  |
|  |  |
| *Semestre:* | 2018-1 |
| *Fecha de entrega:* | 27/octubre/2017 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

**Guía práctica de estudio 10: Depuración de programas**

**Objetivo:**

Aprender las técnicas básicas de depuración de programas en C para revisar de manera precisa el flujo de ejecución de un programa y el valor de las variables; en su caso, corregir posibles errores.

**Actividades:**

* Revisar, a través de un depurador, los valores que va tomando una variable en un programa escrito en C, al momento de ejecutarse.
* Utilizando un depurador, revisar el flujo de instrucciones que se están ejecutando en un programa en C, cuando el flujo depende de los datos de entrada.

**Actividades realizadas en clase:**

En esta práctica, tras leerla; la profesora nos explicó cómo utilizar el depurador de programas en Dev C++, además de cómo añadir variables para observar que cambios tienen cuando se avanza con el programa, de este modo podemos revisar si las variables tienen algún error en el procedimiento que deben de seguir.

Para evaluar esta práctica la profesora nos pidió que reparáramos los programas que vienen en el manual de prácticas y también el de la chicharronera, usando el depurador para encontrar fácilmente los errores que estos tuvieran.

* Para el siguiente código fuente, utilizar algún entorno de depuración para encontrar la utilidad del programa y la funcionalidad de los principales comandos de depuración, como puntos de ruptura, ejecución de siguiente línea o instrucción.

#include <stdio.h>

int main()

{

int N, CONT, AS;

AS=0;

CONT=1;

printf("TECLEA UN NUMERO: ");

scanf("%i",&N);

while(CONT<=N)

{

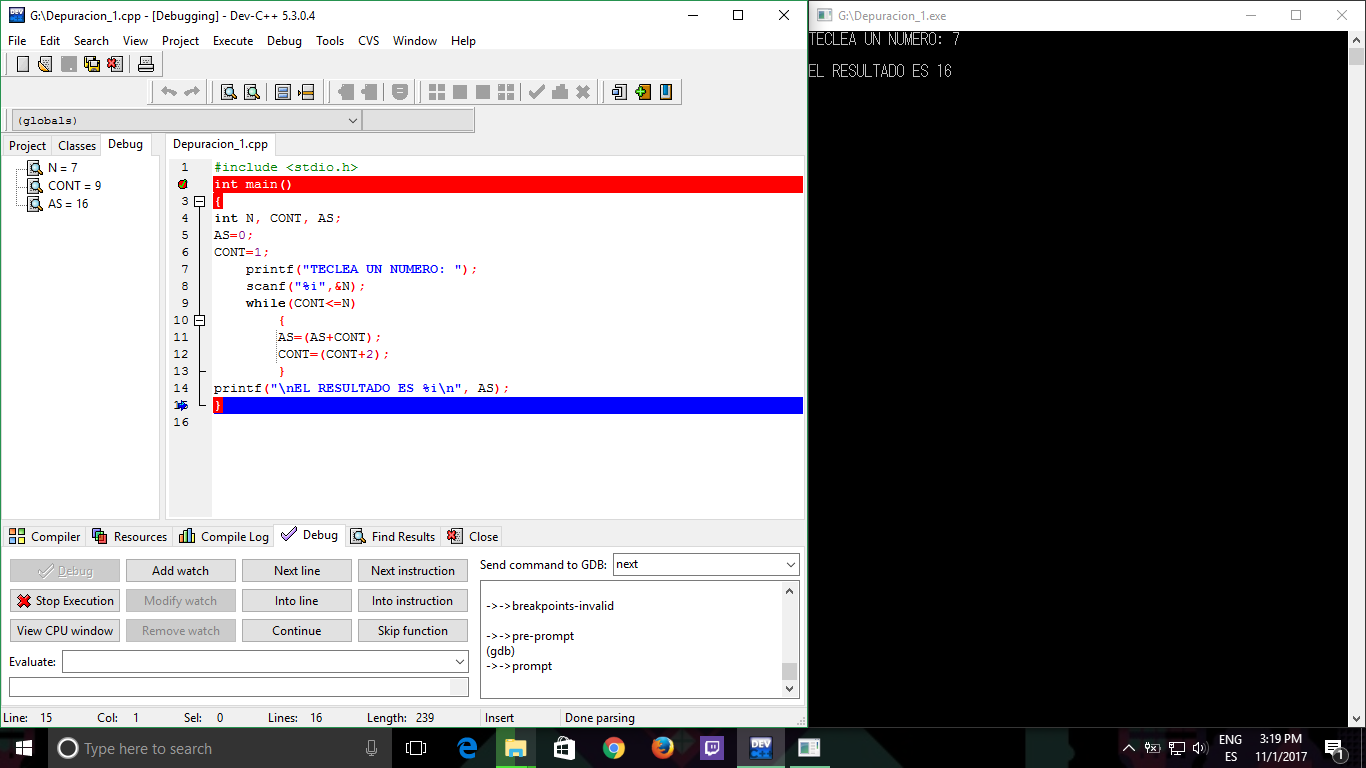
AS=(AS+CONT);

CONT=(CONT+2);

}

printf("\nEL RESULTADO ES %i\n", AS);

}



Lo que este programa hace es que partiendo desde el 1, cada dos números va a sumar un número primo más que en el par anterior. Es decir que si se le ingresa un valor:

N=3, imprimirá AS=4. Esto sumando los primeros dos números primos de CONT=1, 3, 5 durante el ciclo.

Ejemplo

N=1, AS=1, CONT=1, 3.

N=2, AS=1, CONT=1, 3.

N=4, AS=4, CONT=1, 3, 5.

N=8, AS=16, CONT=1, 3, 5, 7, 9.

N=7, AS=16, CONT=1, 3, 5, 7, 9.

Cuando el ciclo termina, no se suma el último número primo, por lo que no lo suma a AS.

* El siguiente programa debe mostrar las tablas de multiplicar desde la del 1 hasta la del 10. En un principio no se mostraba la tabla del 10, luego después de intentar corregirse sin un depurador dejaron de mostrarse el resto de las tablas. Usar un depurador de C para averiguar el funcionamiento del programa y corregir ambos problemas.

#include <stdio.h>

//Tablas de multiplicar del uno al diez

int main()

{

int i, j;

for(i=1; i<11; i++)

{

printf("\nTabla del %i\n", i);

for(j=1; j<11; j++)

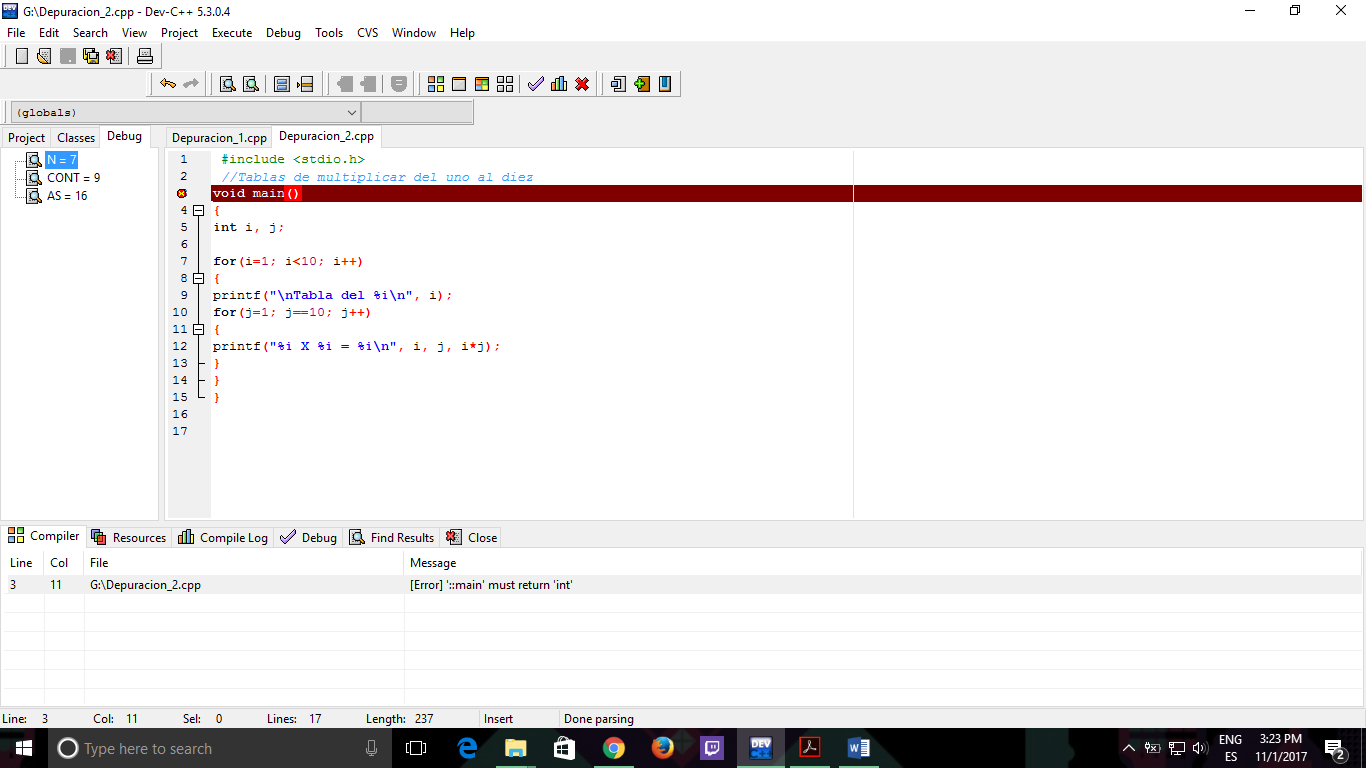
{

printf("%i X %i = %i\n", i, j, i\*j);

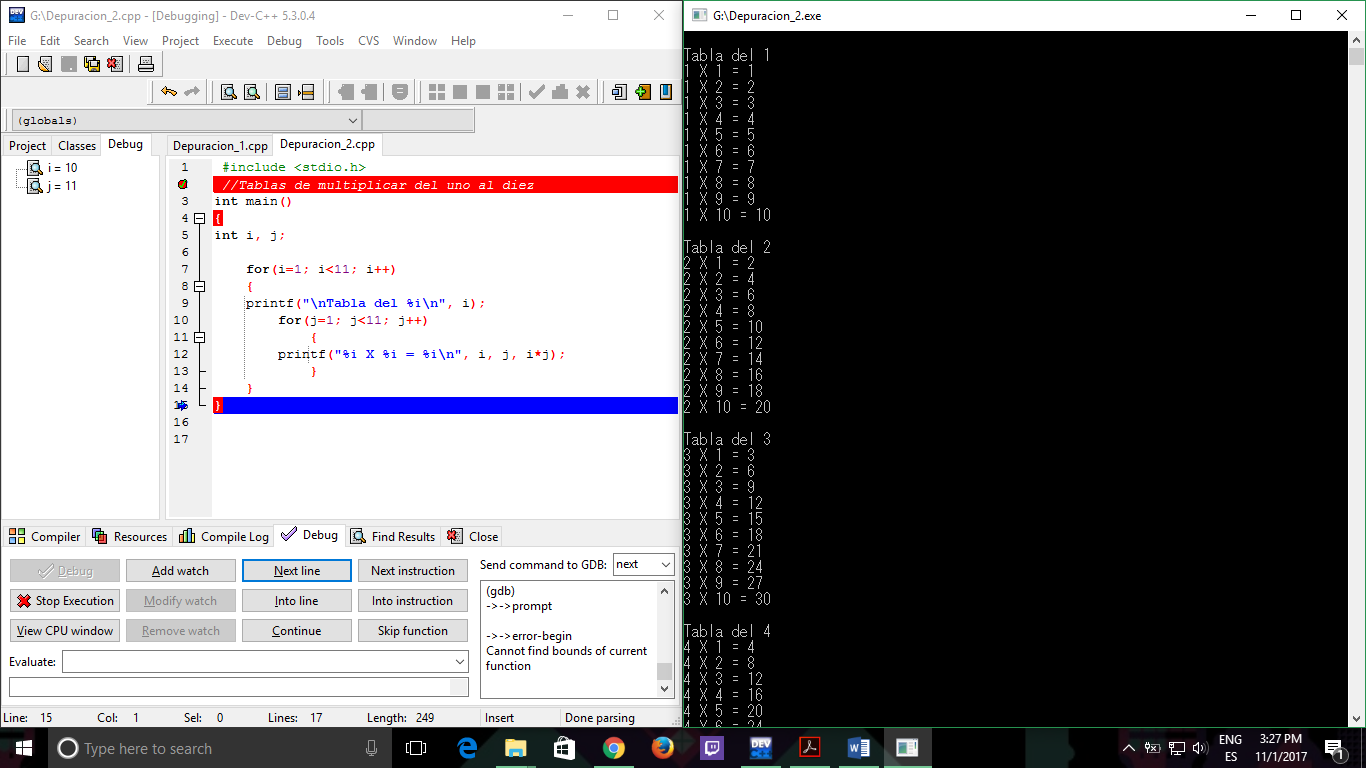
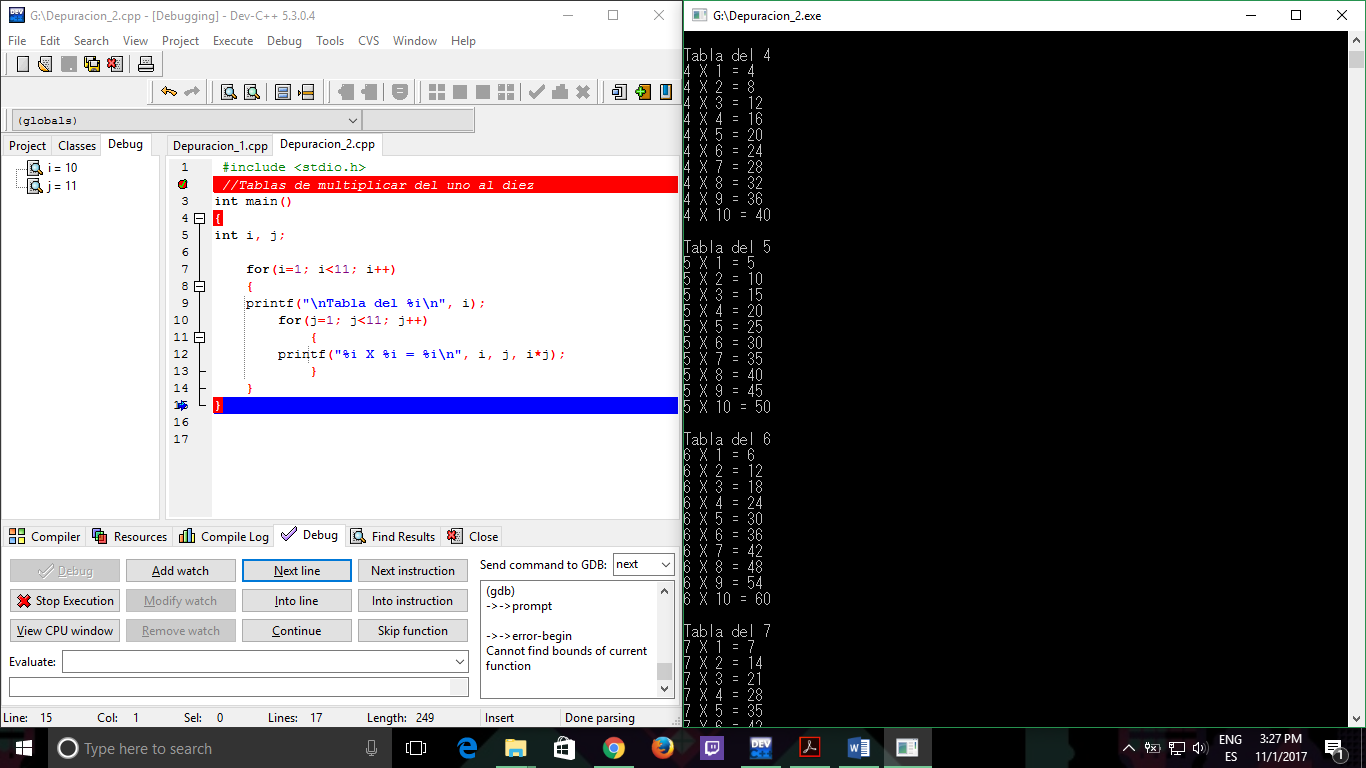
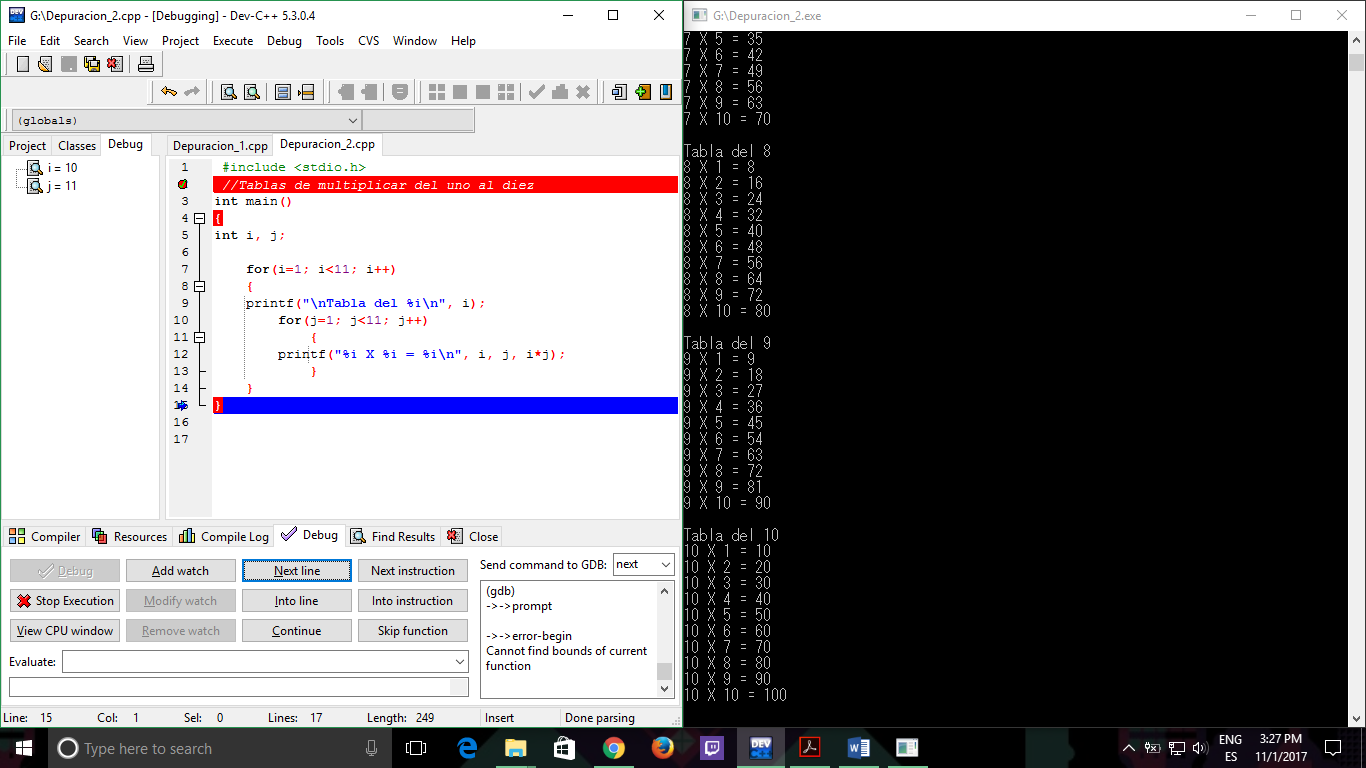
}

}

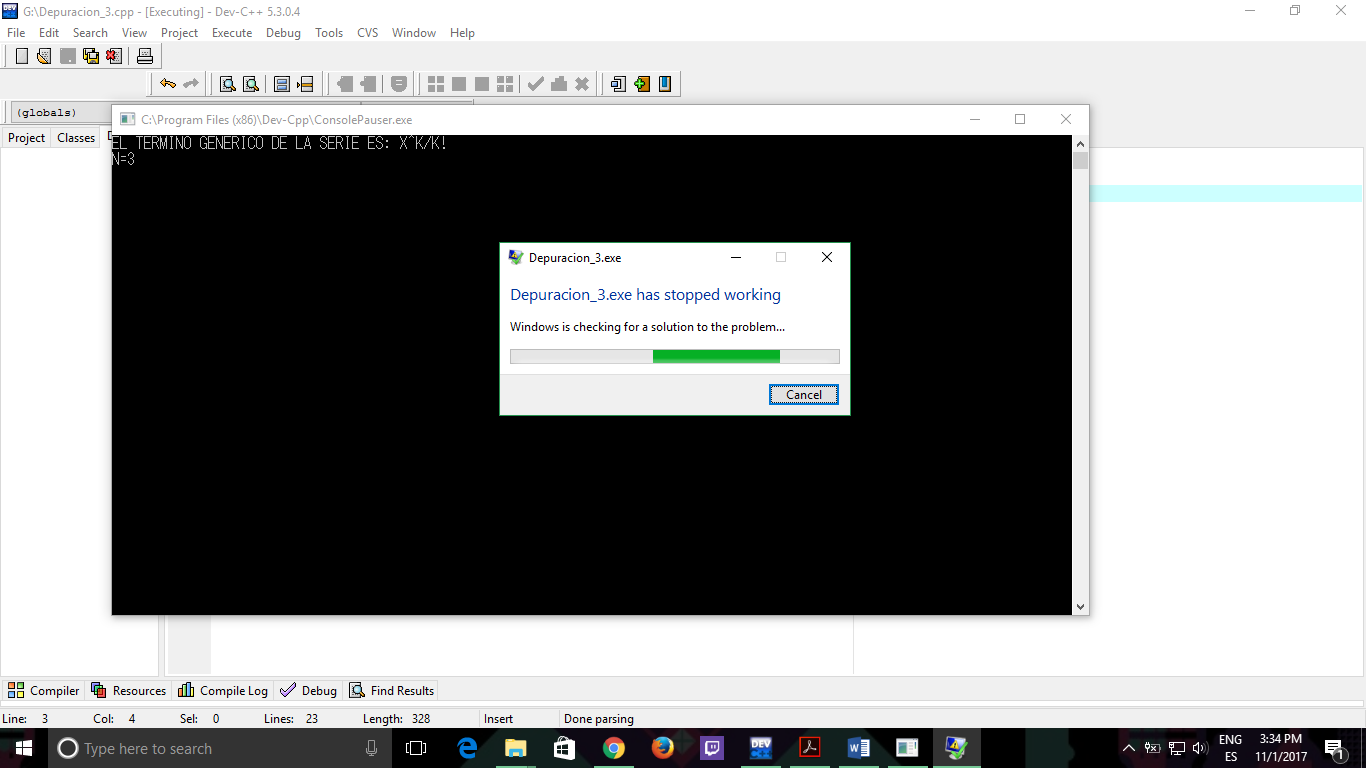
}



Una vez reparado imprime correctamente las tablas de multiplicar del 1 al 10.

* El siguiente programa muestra una *violación de segmento* durante su ejecución y se interrumpe; usar un depurador para detectar y corregir la falla.



#include <stdio.h>

#include <math.h>

int main()

{

int K, X, AP, N;

float AS;

printf("EL TERMINO GENERICO DE LA SERIE ES: X^K/K!");

printf("\nN=");

scanf("%d",&N);

printf("X=");

scanf("%d",&X);

K=0;

AP=1;

AS=0;

while(K<=N)

{

AS=AS+pow(X,K)/AP;

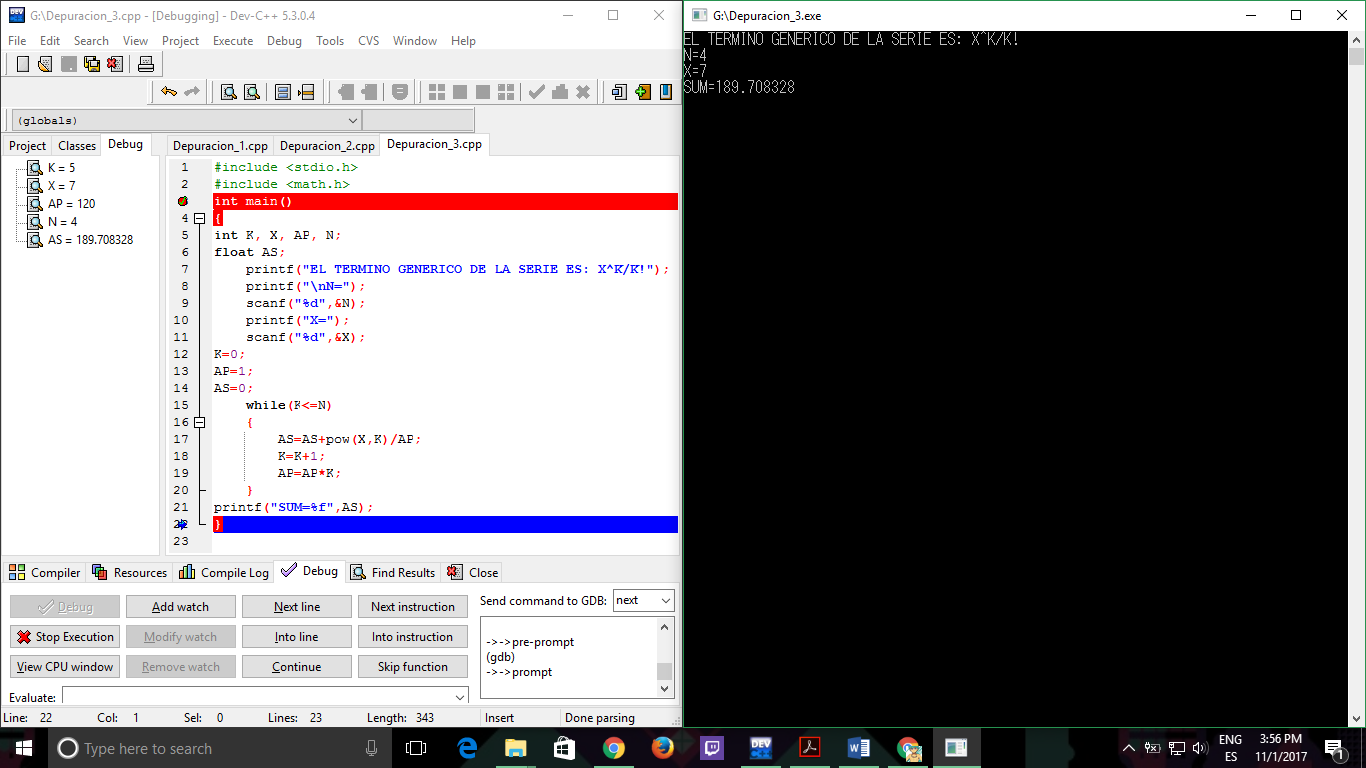
K=K+1;

AP=AP\*K;

}

printf("SUM=%f",AS);

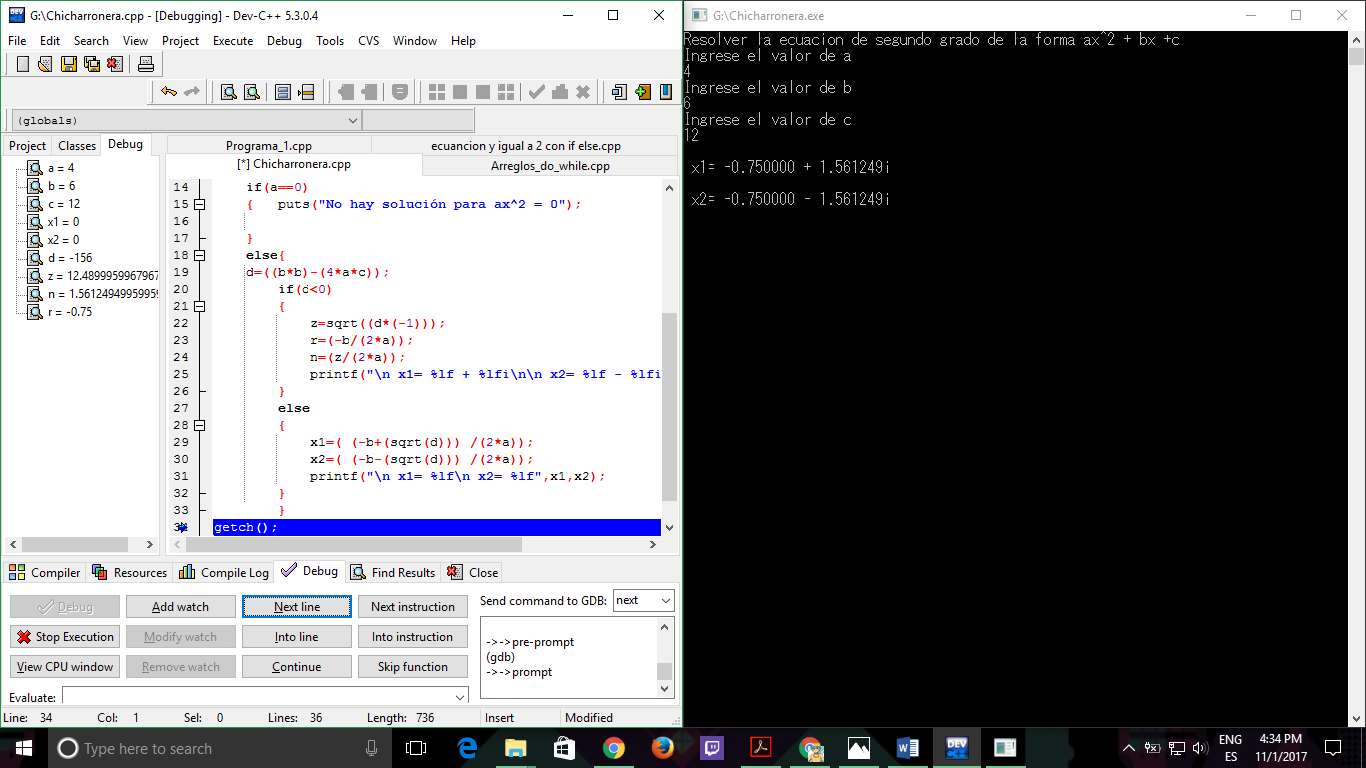
}



El problema que causa que el programa no sea ejecutable, es que en el código erróneo, no hay ‘&’ en los ‘scanf’.

* Ejecutar el programa de la chicharronera usando un depurador.

Este programa no contenía ningún erro, pero al usar el depurador podemos observar como cambia el valor de las variables que se están evaluando para así saber si es que las operaciones se están realizando correctamente.



**Conclusión:**

El depurador nos permite observar cómo se está ejecutando un programa, además de ver los cambios en el valor de las variables durante la ejecución paso a paso del mismo. Al controlar la ejecución línea por línea del programa, también podemos ver el flujo que este está siguiendo, además de que al usar el depurador, si hay un error que haga que el programa se cierre, en el depurador es más controlable.