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

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Ing. Claudia Rodríguez Espino |
| *Asignatura:* | Fundamentos de Programación |
| *Grupo:* | 3 |
| *No de Práctica(s):* | 10 |
| *Integrante(s):* | Villegas Garzón Behram Yair |
| *Semestre:* | 2018-2 |
| *Fecha de entrega:* |  |
| *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.

**Introducción**

Depurar un programa significa someterlo a un ambiente de ejecución controlado por medio de herramientas dedicadas a ello. Este ambiente permite conocer exactamente el flujo de ejecución del programa, el valor que las variables adquieren, la pila de llamadas a funciones, entre otros aspectos. Es importante poder compilar el programa sin errores antes de depurarlo.

Antes de continuar, es necesario conocer las siguientes definiciones (extraídas del Glosario IEEE610) ya que son parte latente del proceso de Desarrollo de Software:

**Error**. Se refiere a una acción humana que produce o genera un resultado incorrecto.

**Defecto (Fault).** Es la manifestación de un error en el software. Un defecto es encontrado porque causa una Falla (failure).

**Falla (failure**). Es una desviación del servicio o resultado esperado.

La depuración de un programa es útil cuando:

- Se desea optimizar el programa: no basta que el programa se pueda compilar y se someta a pruebas que demuestren que funciona correctamente. Debe realizarse un análisis exhaustivo del mismo en ejecución para averiguar cuál es su flujo de operación y encontrar formas de mejorarlo (reducir el código, utilizar menos recursos llegando a los mismos resultados, hacer menos rebuscado al algoritmo), o bien, encontrar puntos donde puede fallar con ciertos tipos de entrada de datos.

- El programa tiene algún fallo: el programa no muestra los resultados que se esperan para cierta entrada de datos debido a que el programador cometió algún error durante el proceso de diseño. Muchas veces encontrar este tipo de fallos suele ser difícil, ya sea porque la percepción del programador no permite encontrar la falla en su diseño o porque la errata es muy pequeña, pero crucial. En este caso es de mucha utilidad conocer paso a paso cómo se ejecutan las estructuras de control, qué valor adquieren las variables, etc.

- El programa tiene un error de ejecución o defecto: cuando el programa está ejecutándose, éste se detiene inesperadamente. Suele ocurrir por error en el diseño o implementación del programa en las que no se contemplan las limitaciones del lenguaje de programación o el equipo donde el programa se ejecuta. Como el programa se detiene inesperadamente, no se conoce la parte del programa donde se provoca el defecto, teniendo que recurrir a la depuración para encontrarlo. El más común de este tipo de defecto es la “violación de segmento”.

Algunas funciones básicas que tienen en común la mayoría de los depuradores son las siguientes:

- Ejecutar el programa: se procede a ejecutar el programa en la herramienta de depuración ofreciendo diversas opciones para ello.

- Mostrar el código fuente del programa: muestra cuál fue el código fuente del programa con el número de línea con el fin de emular la ejecución del programa sobre éste, es decir, se indica qué parte del código fuente se está ejecutando a la hora de correr el programa.

- Punto de ruptura: también conocido por su traducción al inglés breakpoint, sirve para detener la ejecución del programa en algún punto indicado previamente por medio del número de línea. Como la ejecución del programa es más rápida de lo que podemos visualizar y entender, se suelen poner puntos de ruptura para conocer ciertos parámetros de la ejecución como el valor de las variables en determinados puntos del programa. También sirve para verificar hasta qué punto el programa se ejecuta sin problemas y en qué parte podría existir el error, esto es especialmente útil cuando existe un error de ejecución.

- Continuar: continúa con la ejecución del programa después del punto de ruptura.

- Ejecutar la siguiente instrucción: cuando la ejecución del programa se ha detenido por medio del depurador, esta función permite ejecutar una instrucción más y detener el programa de nuevo. Esto es útil cuando se desea estudiar detalladamente una pequeña sección del programa. Si en la ejecución existe una llamada a función se ingresará a ella.

- Ejecutar la siguiente línea: es muy similar a la función anterior, pero realizará todas las instrucciones necesarias hasta llegar a la siguiente línea de código. Si en la ejecución existe una llamada a función se ignorará.

- Ejecutar la instrucción o línea anterior: deshace el efecto provocado por alguna de las funciones anteriores para volver a repetir una sección del programa.

- Visualizar el valor de las variables: permite conocer el valor de alguna o varias variables.

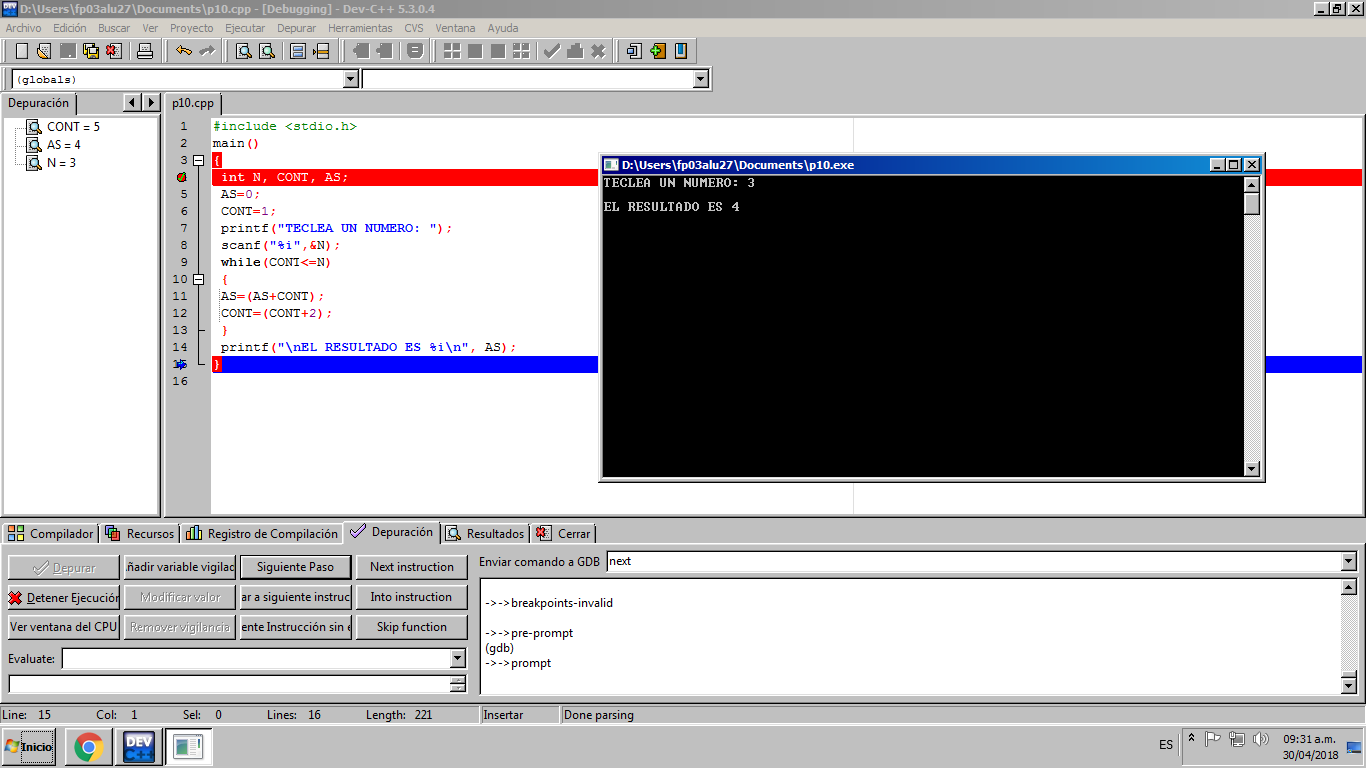
Dependiendo de la herramienta usada para compilar el programa, si es de consola o de terminal, su uso y las funciones disponibles variarán.

En las IDE (Entornos de Desarrollo Interactivo), suelen existir herramientas de depuración integradas de manera gráfica. Es muy común que existan dos modos de desarrollar un programa y producir el archivo ejecutable que son “Debug” y “Release”. El primer modo se recomienda exclusivamente durante el desarrollo del programa para poder depurarlo continuamente durante cualquier prueba de ejecución. El segundo modo se establece cuando el programa ha sido terminado y totalmente probado.

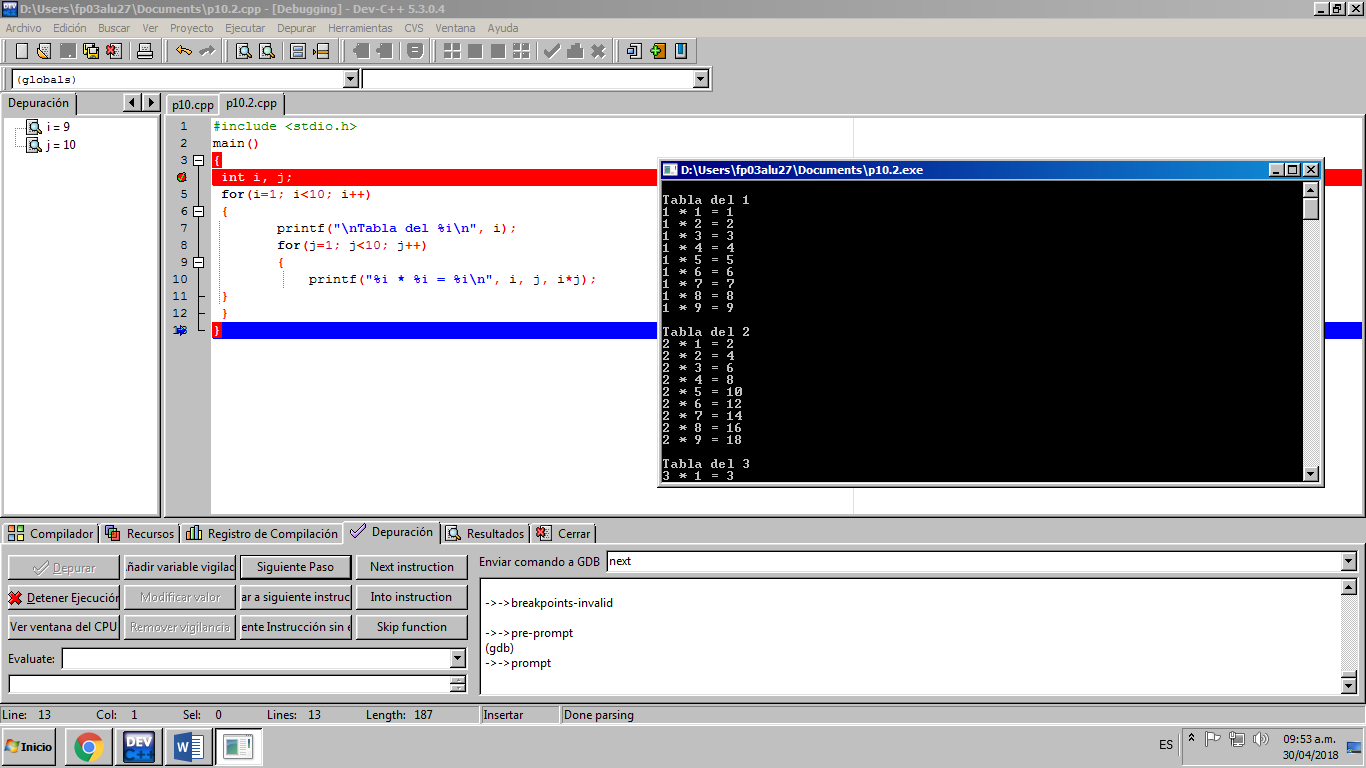
**Desarrollo**

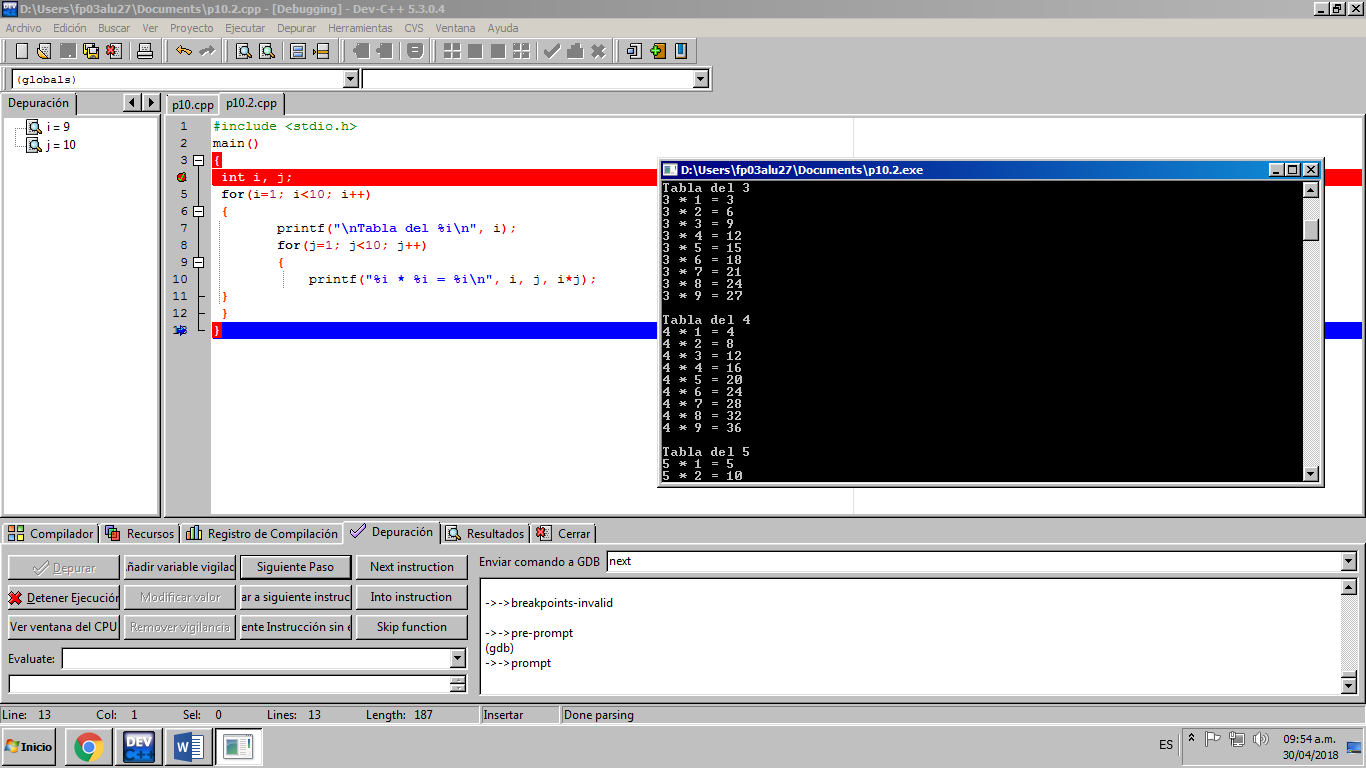
Utilizando un depurador revisamos diferentes programas y determinamos donde está el error y lo corregimos

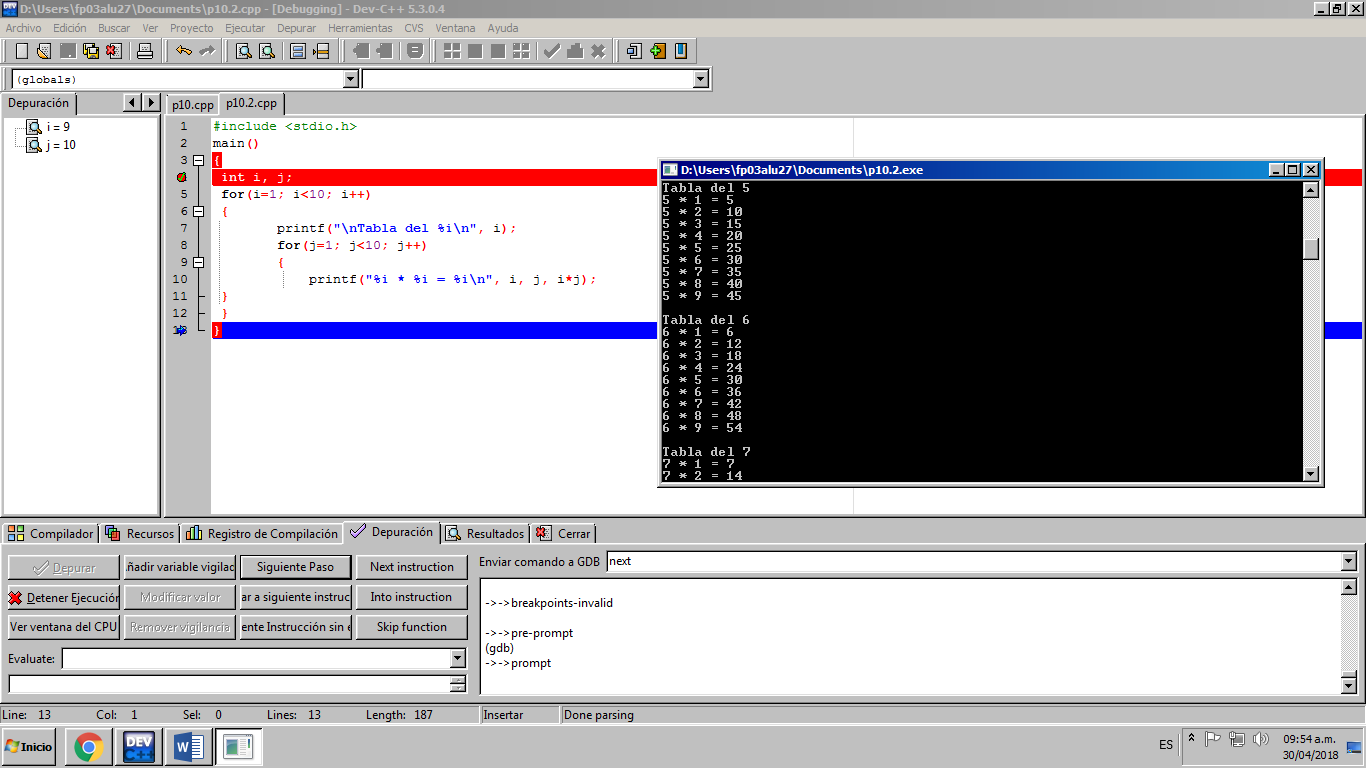
1. Contador

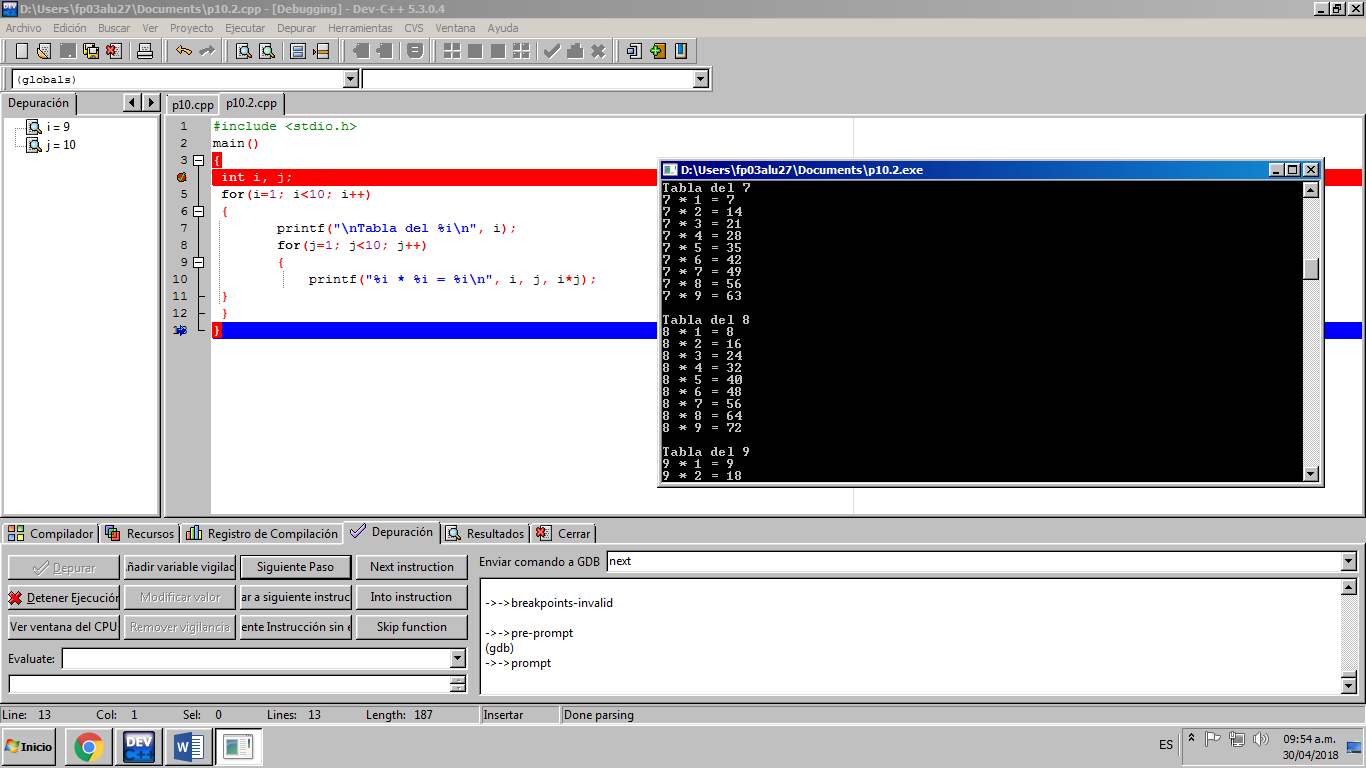


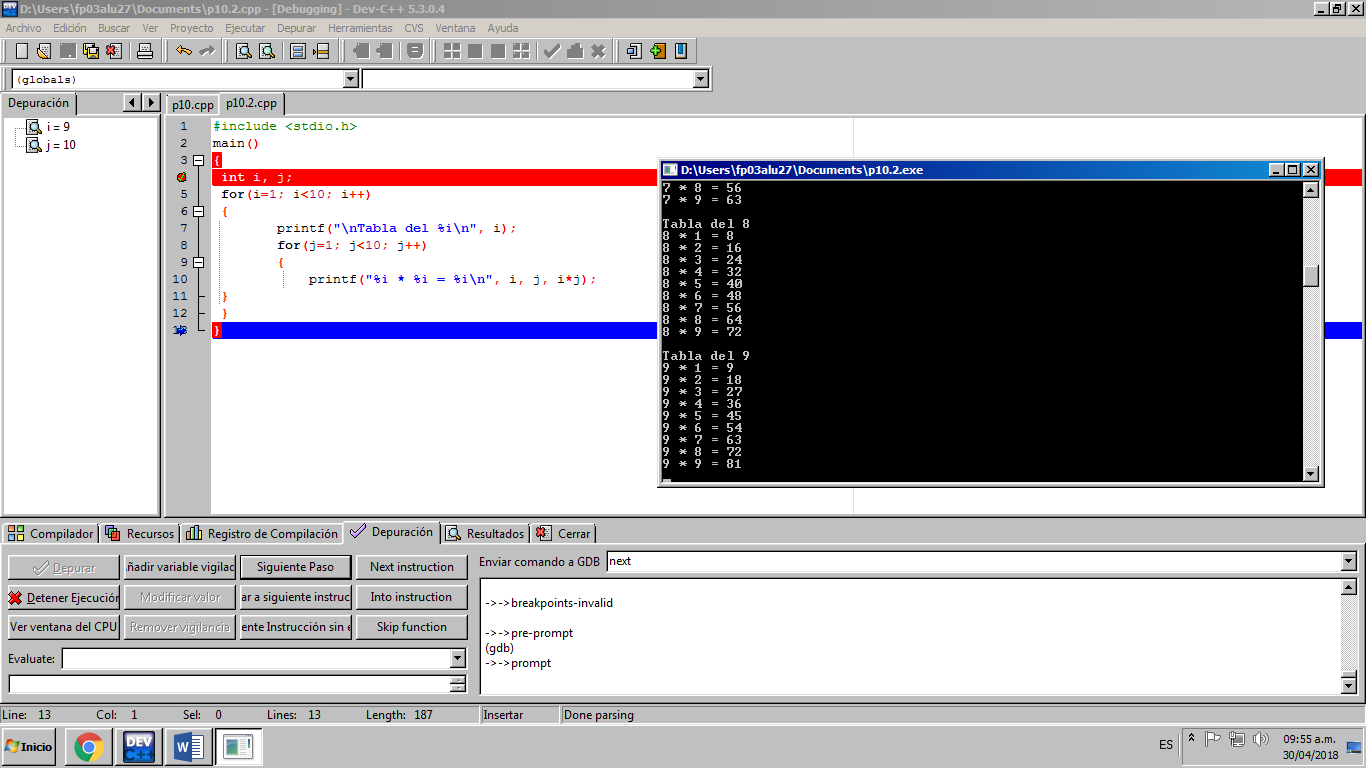
2. Tablas de multiplicación



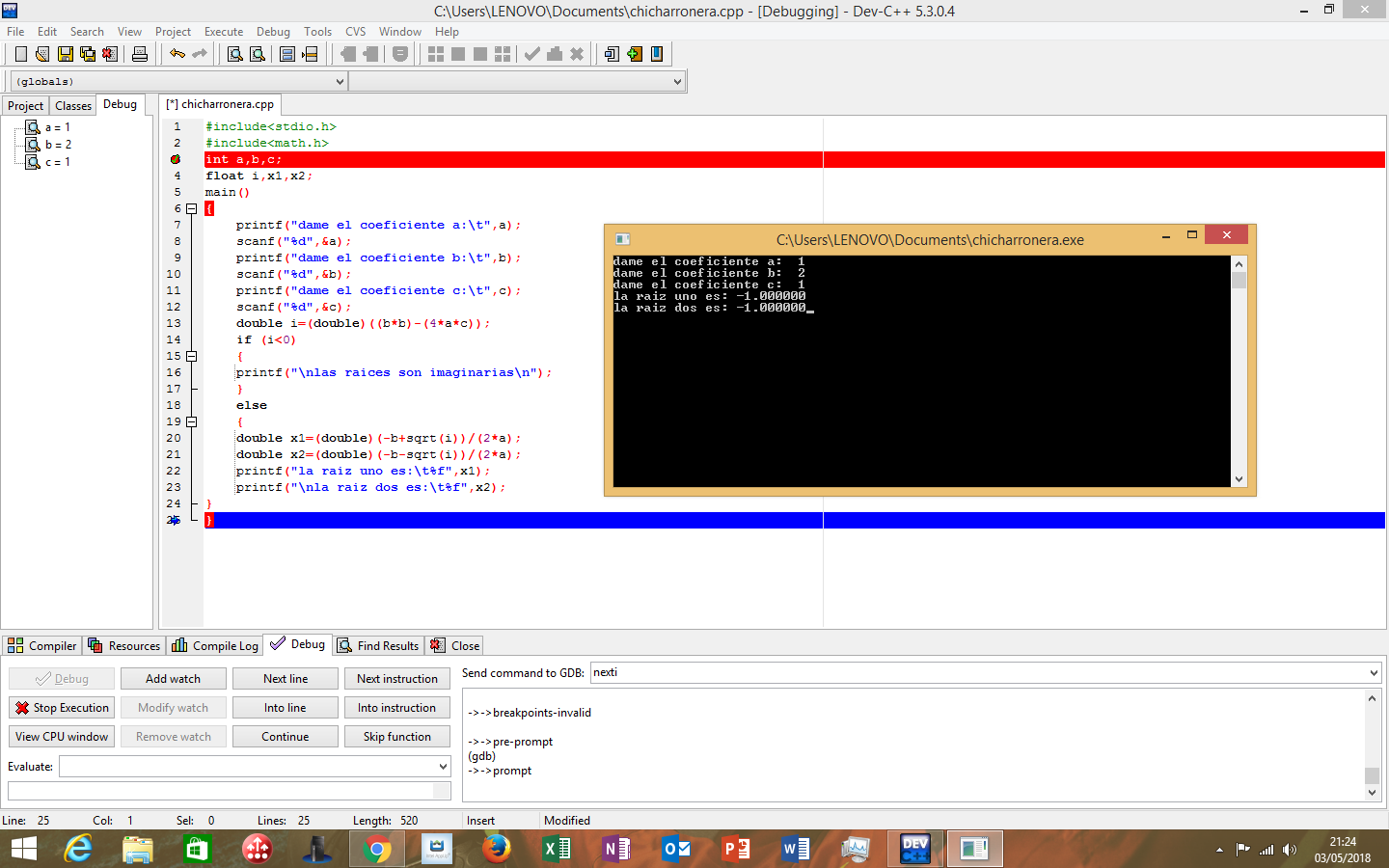


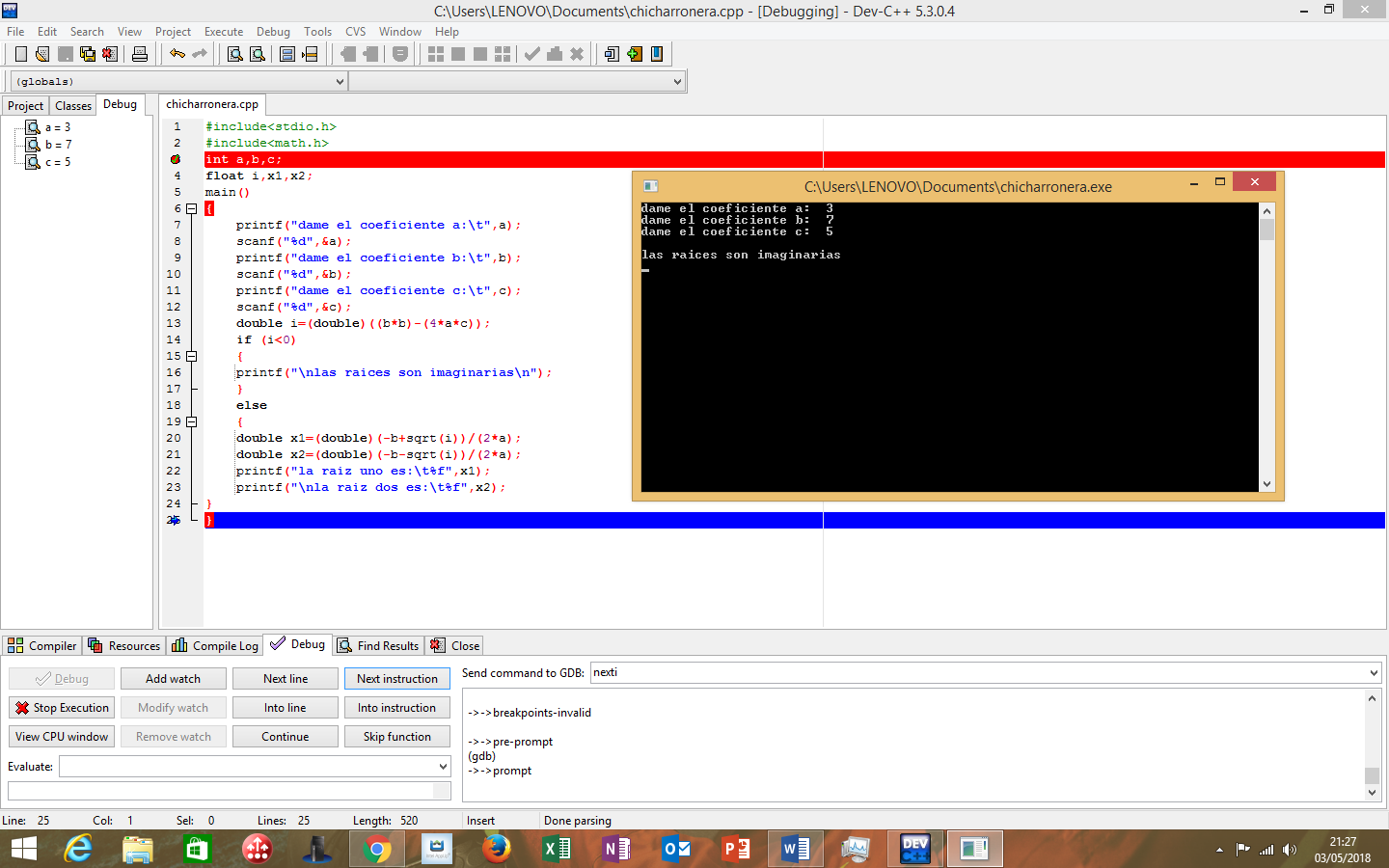




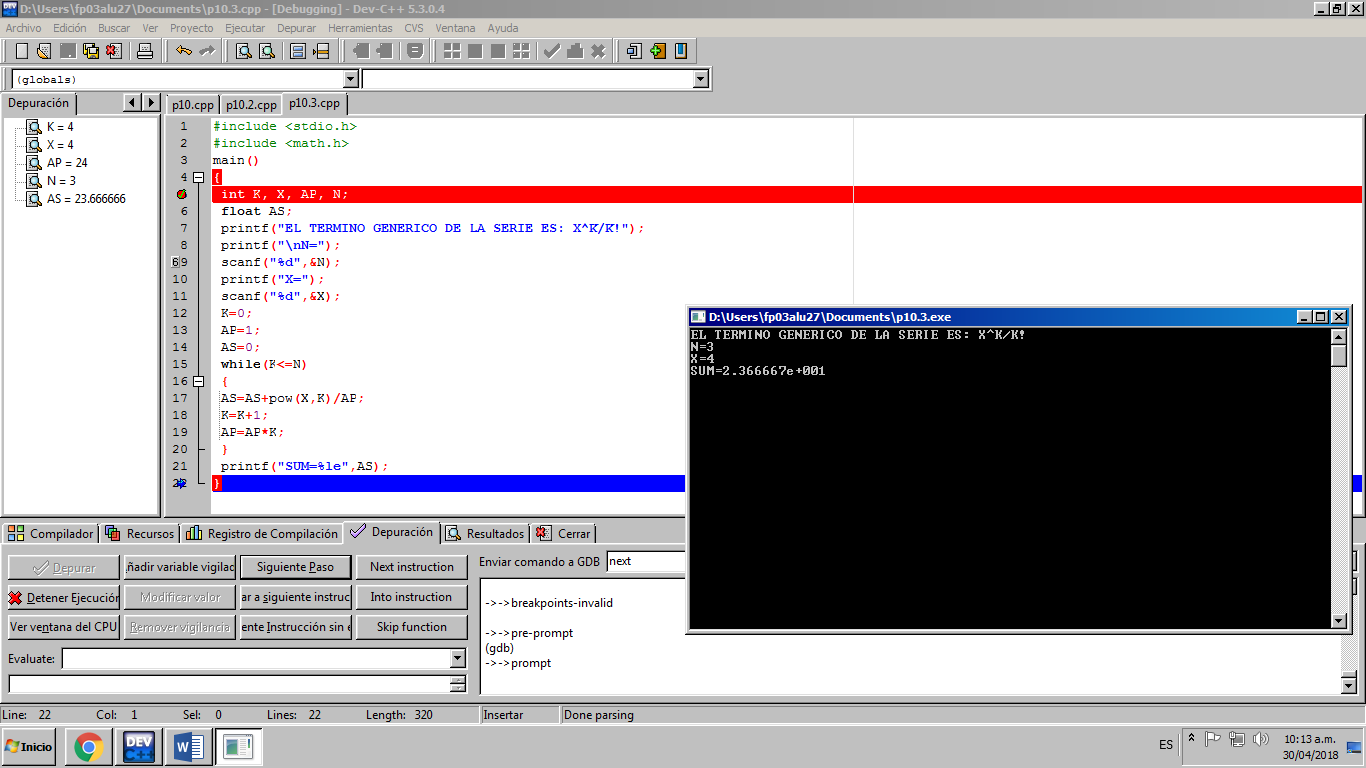


3. Chicharronera





4. Término genérico de la serie



**Conclusión**

El desarrollo de esta práctica me ayudo a comprender de mejor manera la función de un depurador, el cual resulta muy útil al momento de revisar nuestros programas, ya que hay veces que no funcionan adecuadamente y no sabemos en qué parte de nuestro programa esta nuestro error, así que con el depurador podemos analizar nuestro programa paso por paso para poder visualizar donde está el error.