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

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| Profesor: | García Morales Karina |
| Asignatura: | Fundamentos de Programación |
| Grupo: | 1121 |
| No de Práctica(s): | 10 |
| Integrante(s): | Tinoco Ramos Isaac Ricardo |
|  |  |
|  |  |
|  |  |
| No. de Equipo de cómputo empleado: |  |
| Semestre: | Primero |
| Fecha de entrega: | 30/10/2018 |
| Observaciones: |  |
|  |  |

Calificación\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

**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:**

El proceso de prueba es la fase en la cual se ejecuta el programa con algunos datos, especialmente seleccionados, para encontrar los errores que pueda haber El proceso de depuración es la fase en la cual se hacen modificaciones al programa para eliminar los errores. Hay dos tipos de depuración: difícil y profunda. Una depuración difícil significa que hay problemas con la lógica del programa, mientras que una depuración profunda sólo mejora el programa, lo libera de errores sencillos o busca código optimizable. Además de buscar errores, la depuración permite detectar algunos detalles importantes para el mejoramiento del programa; por ejemplo:1. Expresiones booleanas que requieren simplificación. 2. Segmentos de código que nunca se ejecutan, 3. Instrucciones en las que se ocupa el 90% del tiempo, que pueden ser optimizables, e inclusive 4. Necesidad de volver a empezar desde el principio. Desafortunadamente, no existe ningún método que permita probar por completo, en un tiempo razonable, un programa de gran complejidad. Esto se debe a que mientras mayor cantidad de errores se detectan (y corrigen), más difícil resulta detectar los siguientes. Ante la imposibilidad de hacer pruebas exhaustivas se recurre a una selección cuidadosa de datos de prueba. Si el programa funciona correctamente con esos datos, se puede tener mucha confianza sobre su futuro comportamiento. El momento de dejar de buscar errores es cuando el esfuerzo de hacerlo es mayor que el costo de los fallos aún no detectados. Para reducir las posibilidades de error se debe tener en cuenta que la mayoría de los errores ocurre en las etapas de especificación y diseño (por prestar poca atención a la naturaleza del problema), y se reducen siguiendo algún método estructurado. También se generan errores en la traducción del diseño al código; y finalmente suele haber errores en la comprobación y depuración, la misma corrección de errores puede dar pauta a otros.

Existe una gran variedad de errores ocurridos mientras la depuración:

Existen diferentes situaciones en las cuales se suelen introducir errores en un programa: 1. Violación (no cumplimiento) de las reglas gramaticales del lenguaje de programación 2. Los errores en el diseño del algoritmo en el que está basado el programa.

**Errores de sintaxis**

Los errores de sintaxis son aquellos que se producen cuando el programa viola las reglas de gramática del lenguaje. Errores de sintaxis típicos son: ⎥ Escritura incorrecta de palabras reservadas , Omisión de signos de puntuación (comillas, punto y coma…). Los errores de sintaxis son los más fáciles de ubicar, ya que son detectados y aislados por el compilador.

Puntos y comas después de la cabecera “main()”

♣ Olvido del carácter llave de apertura o cierre ({,})

♣ Omisión de puntos y coma al final de una sentencia

♣ Olvido de la doble barra inclinada antes de un comentario

♣ Olvido de las dobles comillas al cerrar una cadena.

♣ Etc.

**Errores lógicos**

Estos son errores del programador en el diseño del algoritmo y posterior programa. Los errores lógicos son más difíciles de encontrar y aislar, ya que no suelen ser detectados por el compilador

Ejemplo total\_grados\_celsius=fahrenheit\_a\_celsius \* temperatura\_cel; Es una sentencia perfectamente legal en C++ pero la ecuación no responde a ningún cálculo válido para obtener el total de grados Celsius en una sala.

**Errores lógicos**

Se pueden detectar errores lógicos comprobando el programa en su totalidad, comprobando su salida con los resultados previstos. Se pueden prevenir errores con un estudio detallado del algoritmo antes de que el programa se ejecute. Una vez que se ha determinado que un programa contiene un error lógico, encontrar el error es una de las tareas más difíciles de la programación. El depurador (debugger), es un programa diseñado para la detección, verificación y corrección de errores.

**Errores de regresión**

Estos errores se crean accidentalmente cuando se intenta corregir un error lógico. Siempre que se corrige un error se debe comprobar totalmente la exactitud (corrección) para asegurarse que se fija el error que se está tratado y no produce otro error. Los errores de regresión son comunes, pero son fáciles de leer y corregir.

**Mensajes de error**

Los compiladores emiten mensajes de error durante las fases de compilación, de enlace o de ejecución del programa. Se agrupan en tres bloques:

1. Errores fatales. Pueden indicar un error interno del compilador, la compilación se detiene inmediatamente

2. Errores de sintaxis. Errores típicos de sintaxis, errores de línea de órdenes y errores de acceso a memoria o disco.

3. Advertencias. No impiden la compilación, indican condiciones sospechosas pero legítimas como parte del lenguaje.

**Errores en tiempo de ejecución**

Los errores de ejecución se deben, normalmente, a un error en el algoritmo que resuelve el problema. Un ejemplo típico es una división de enteros cuyo código está bien escrito, pero si el divisor es cero, se produce un error de ejecución “fatal” que detiene el programa. Tipos de errores de ejecución, Aquellos que son detectados por el sistema en tiempo de ejecución del programa C++ y aquellos que permiten la terminación del programa pero producen resultados incorrectos.

Un error en tiempo de ejecución puede tener como resultado que el programa obligue a la computadora a realizar una operación ilegal tal como dividir un número por cero o manipular datos no válidos o no definidos. Cuando ocurre este tipo de error, la computadora detendrá la ejecución de su programa y emitirá (visualizará) un mensaje de error.

Para compilar, por ejemplo, un programa llamado calculadora.c con GCC con información de depuración, debe realizarse en una terminal con el siguiente comando:

gcc -g -o calculadora calculadora.c

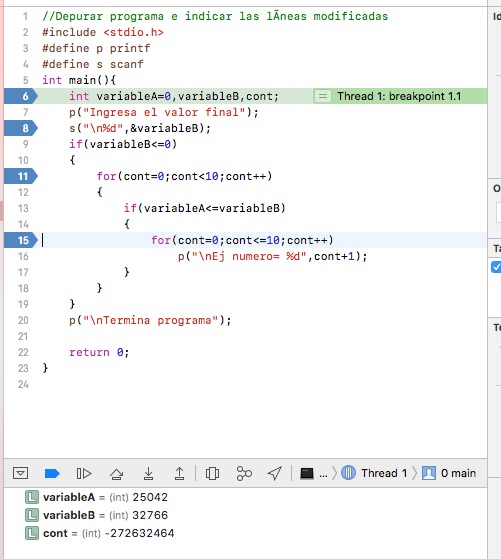
El parámetro -g es quien indica que el ejecutable debe producirse con información de depuración.

Para la depuración en Xcode de Mac se hace de la siguiente manera:

Para iniciar con algunas actividades de depuración, vamos a indicar puntos de ruptura. Esto se realiza haciendo clic, con el botón derecho del mouse, sobre el número que indica la línea de código donde se desea detener y aparecerá una flecha azul:

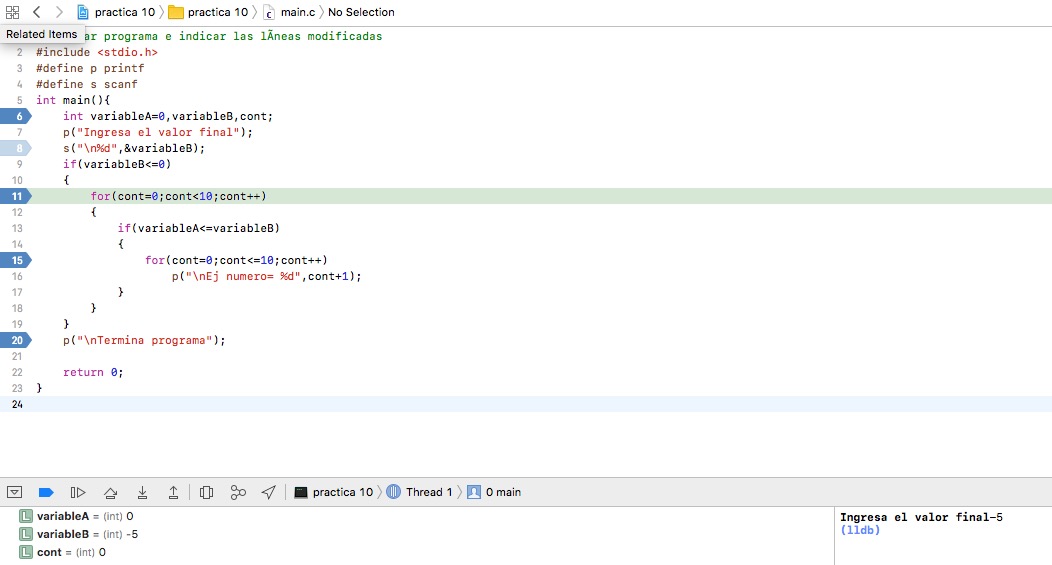


Una vez definidos los puntos de ruptura, al momento de ejecutar el programa se detendrá al encontrar el primer punto de ruptura, indicando automáticamente el valor de todas las variables definidas en el programa hasta ese punto. Por ejemplo:



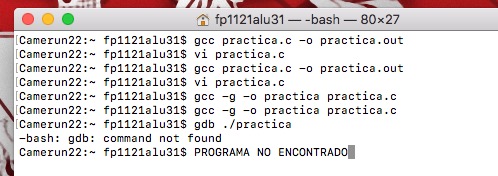
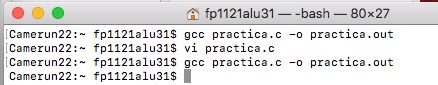
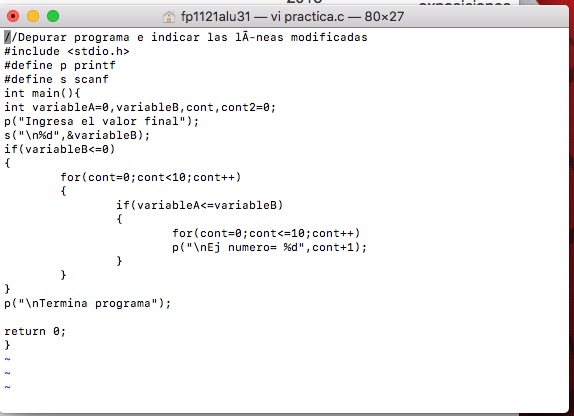
Como se puede observar, en la parte inferior se muestran las variables con su valor y tipo de dato declarado. Si a una variable aún no se le asigna un valor, éste será indeterminado.

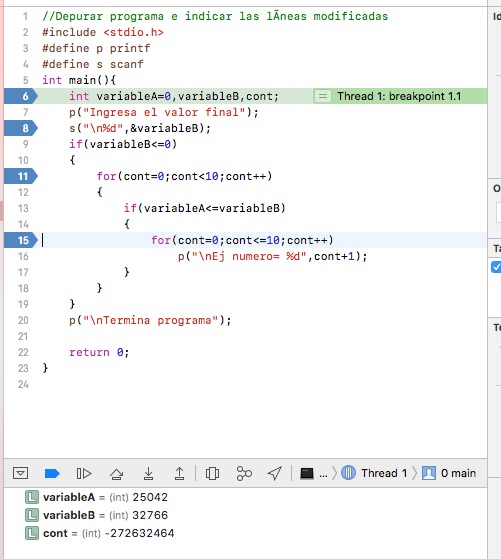
La continuidad y control de la ejecución del programa, se realiza a través de los íconos de la barra de resultados:

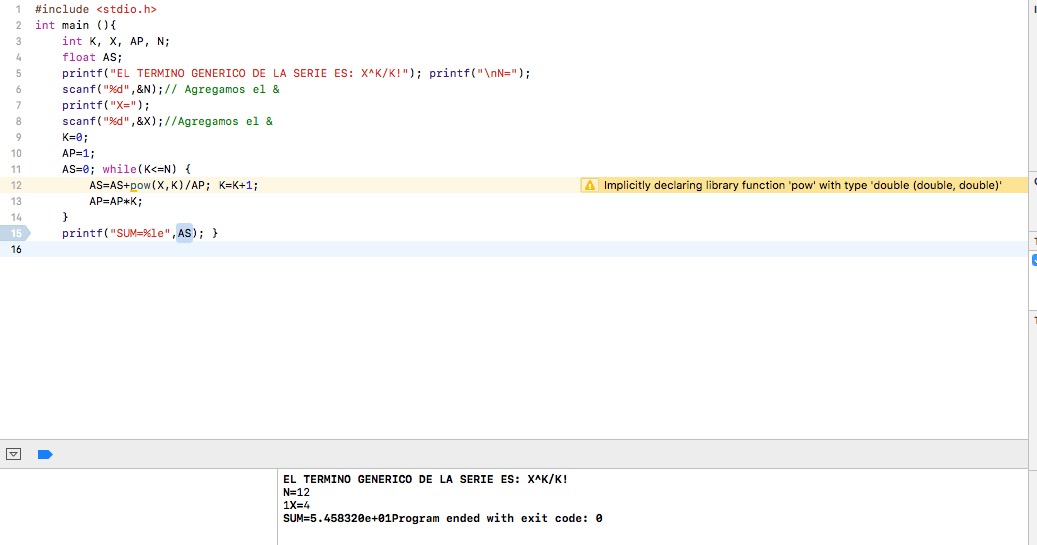


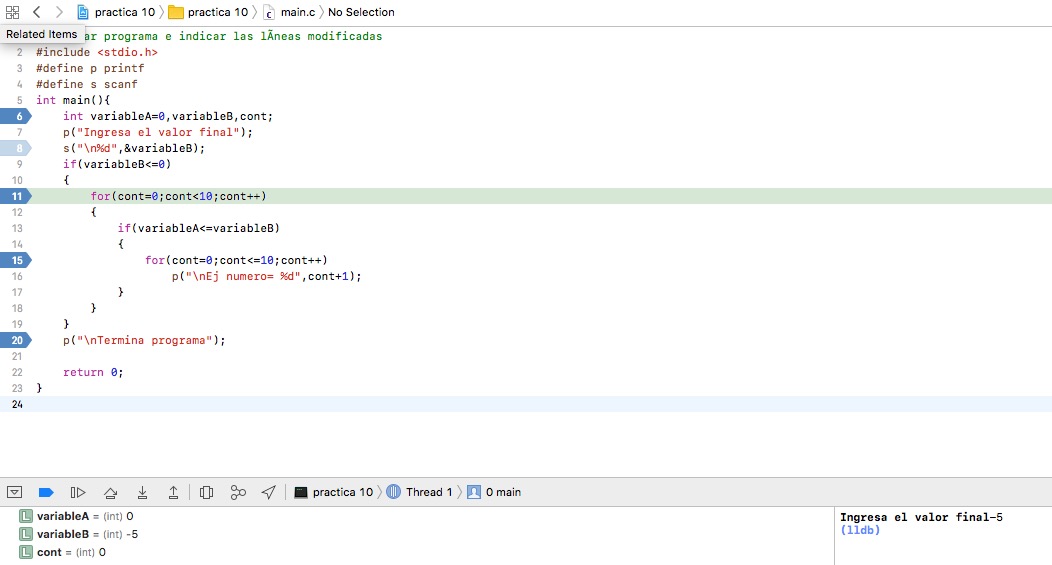
**Desarrollo:**

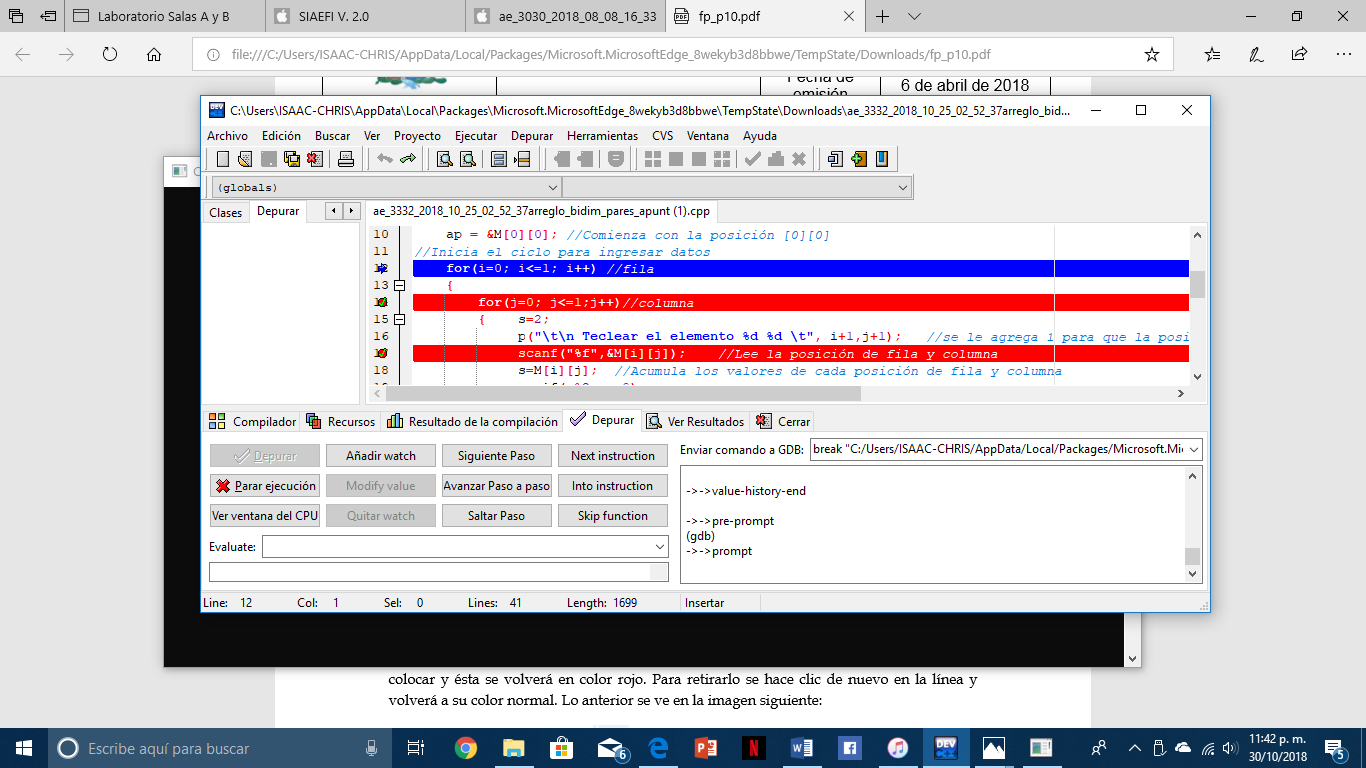
**Creación del programa, más corrección de errores de compilación**

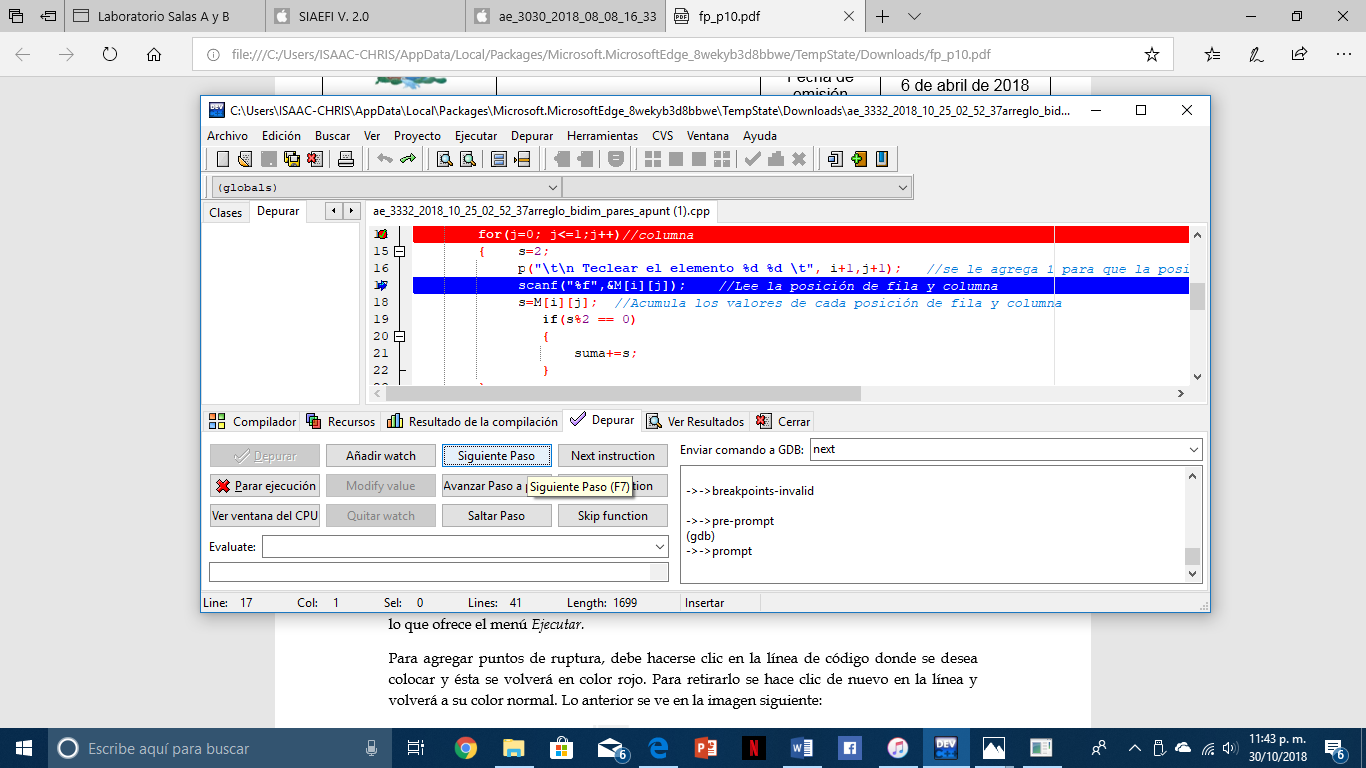
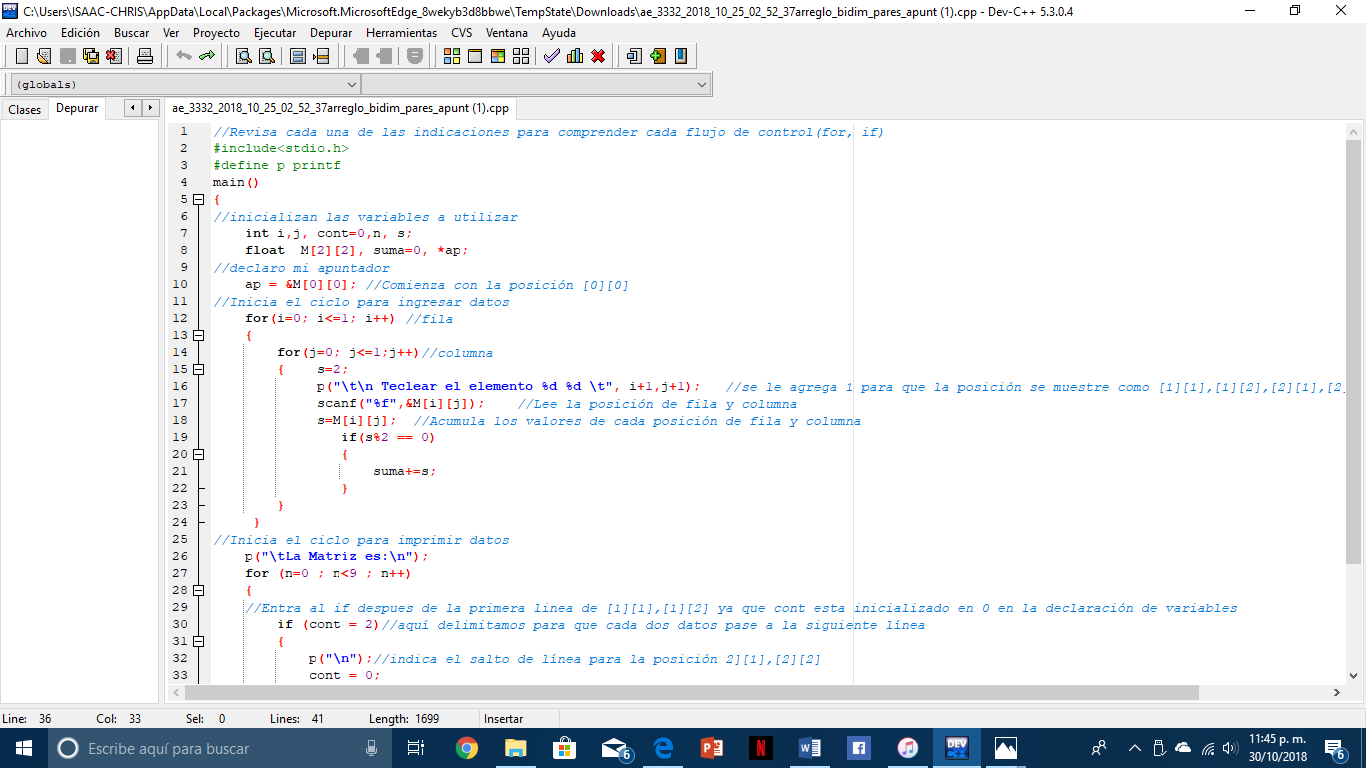
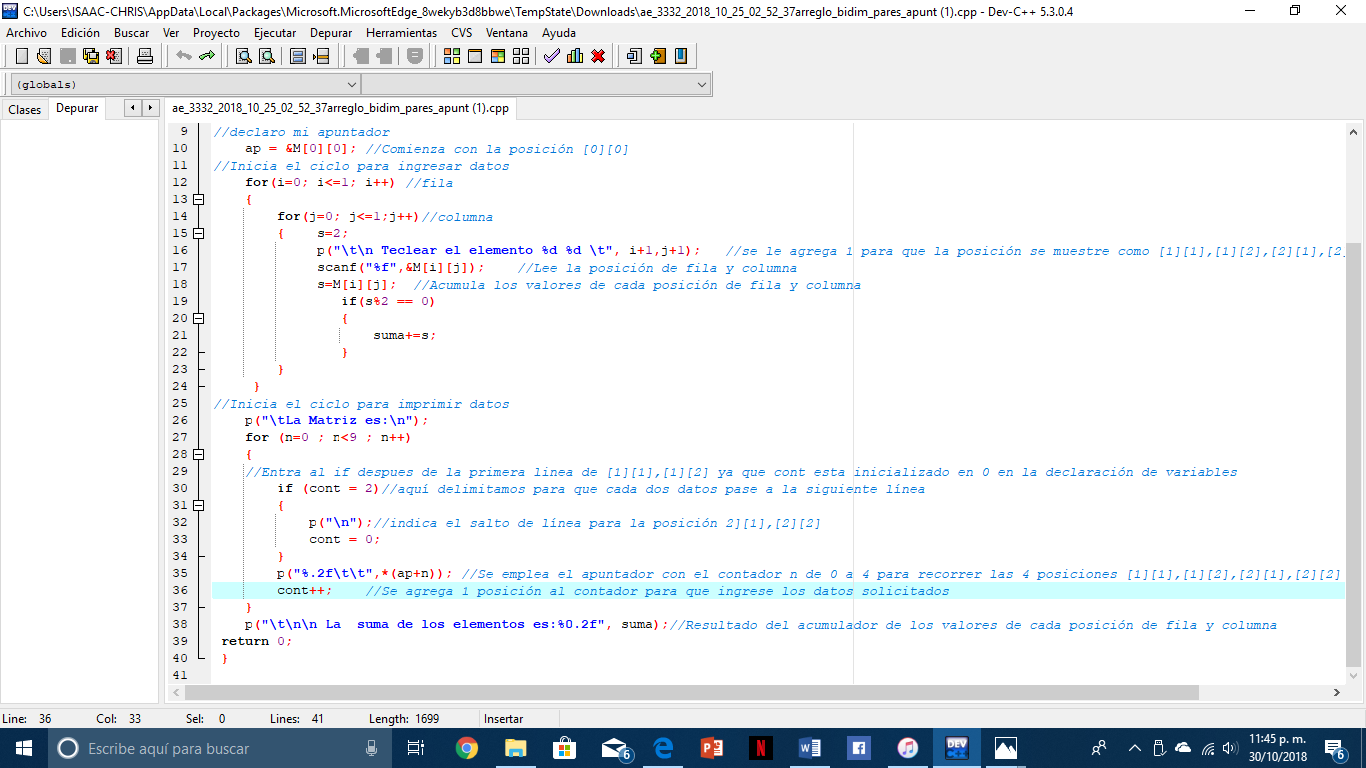










**Conclusión:**

para resumir la depuración de un programa es la forma de saber si un programa contiene errores o no, así mismo también nos ayuda a corregir dichos errores, se dice que un programa esta depurado cuando está libre de errores. Cuando se depura un programa se hace un seguimiento del funcionamiento de dicho programa y se van estudiando los valores de las distintas variables, así como los resultados obtenidos en las operaciones.  
Una vez depurado el programa se solucionan los posibles errores encontrados y se procede a depurar otra vez y estas acciones se repiten hasta que el programa no contiene ningún tipo de errores, tanto en tiempo de programación como en tiempo de ejecución.

Existe una gran variedad de errores ocurridos mientras la depuración:

Errores de sintaxis, Errores lógicos, Errores lógicos, Errores de regresión,

Mensajes de error y Errores en tiempo de ejecución

Link practica 6

https://github.com/Richicardo/practica1\_fdp/blob/master/Práctica%206.docx