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

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | RODRIGUES ESPINO CLAUDIA |
| *Asignatura:* | FUNDAMENTOS DE PROGRAMACION |
| *Grupo:* | 4 |
| *No de Práctica(s):* | PRACTICA 10: Depuración de programas. |
| *Integrante(s):* | FLORES MENDOZA OLGA |
|  |  |
| *No. de Equipo de cómputo empleado* | 44 |
| *Semestre:* | 2019-2 |
| *Fecha de entrega:* | 27 DE ABRIL DE 2019 |
| *Obervaciones:* |  |
|  |  |

COMPILACION

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

**PRACTICA 10**

**OBJETIVOS:**

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.

**INTRODUCCION:**

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.

La depuración de un programa es útil cuando:

* Se desea optimizar el programa: Debe realizarse un análisis exhaustivo del programa en ejecución para averiguar cuál es su flujo de operación y encontrar formas de mejorarlo 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 debido a que el programador cometió algún error durante el proceso de diseño. Muchas veces encontrar este tipo de fallos suele ser difícil, 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: Algunas veces el programa se detiene inesperadamente, esto suele ocurrir por error en el diseño. Como es inesperado no se conoce la parte del programa donde se provoca el defecto, teniendo que recurrir a la depuración para encontrarlo.

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: se indica qué parte del código fuente se está ejecutando a la hora de correr el programa.
* Punto de ruptura: 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. 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.

**DESARROLLO DE ACTIVIDADES:**

Dos de los programas fueron creados en programas escritos en C con Xcode de Mac Xcode es otro Entorno de Desarrollo Interactivo (IDE) que, contiene herramientas de depuración, cuenta con un editor y un compilador entre otros. Este es un nuevo editor ya que no lo habíamos trabajado antes, de modo que se siguieron las siguientes instrucciones para realizar la edición del programa.

1. Abrir la aplicación Xcode y crear un nuevo proyecto seleccionando Create a new Xcode project; se deberá seleccionar macOS y Command Line Tool, y dar Next.
2. Nombrar al proyecto y definir sus características como el nombre de la organización, organización identificación, tipo de lenguaje, etc.
3. Se abrió una ventana donde hay que indicar dónde se grabará el proyecto. Por ejemplo en Documents. Después dar Create.
4. Aparecerá el área de trabajo para editar y depurar programas.

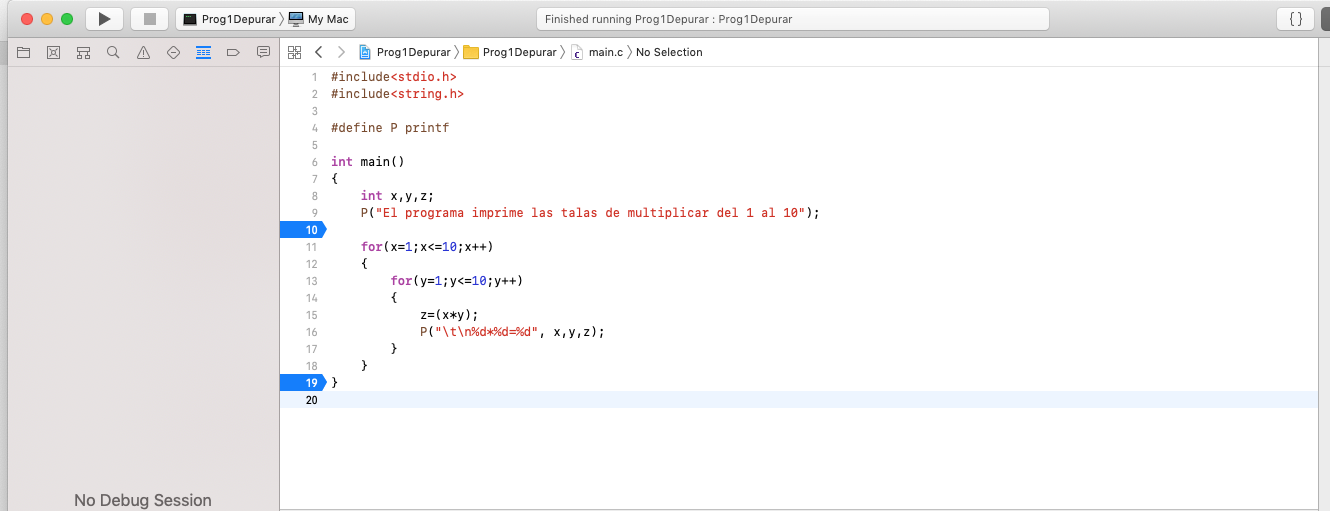
A partir de este punto trabajamos en la edición como se ha venido haciendo. Se recalca que los programas 1 y 2 fueron desarrollados en Xcode, mientras que el 3 fue en Dev-c++.

**PROGRAMA 1:** Este programa imprime las tablas de multiplicar del 1 al 10.

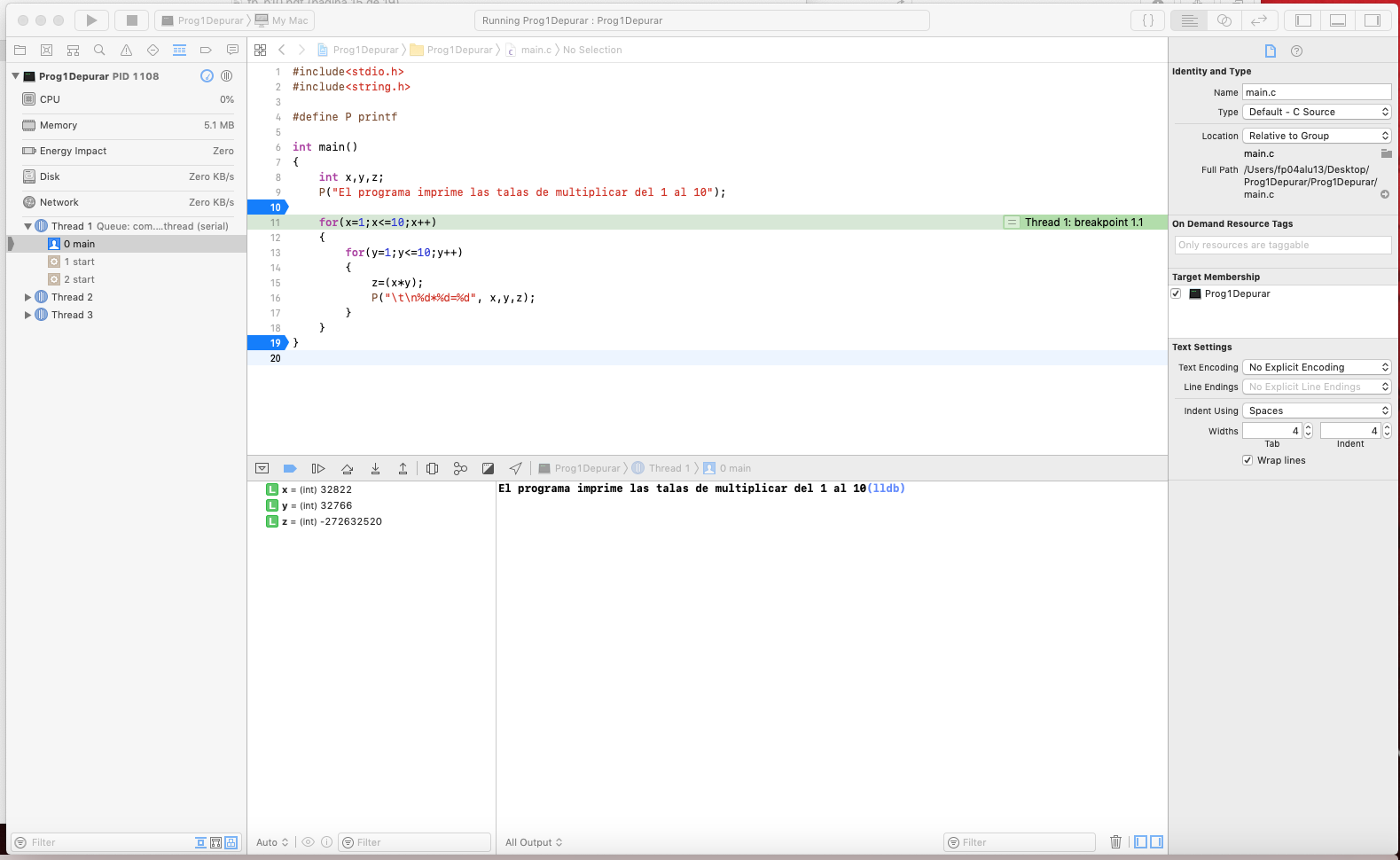
Usando el lenguaje en C primero editamos el programa empleando comandos como: #define, for, operadores básicos. Lo compilamos para verificar que funcionara de acuerdo a lo escrito. **Es muy importante compilar y ejecutar correctamente en programa antes de la depuración.**

En Xcode la manera de compilar fue: usando las teclas Cmd + B

En Xcode la manera de ejecutar fue: oprimiendo la opción **PLAY** ubicada en la parte superior izquierda del editor.

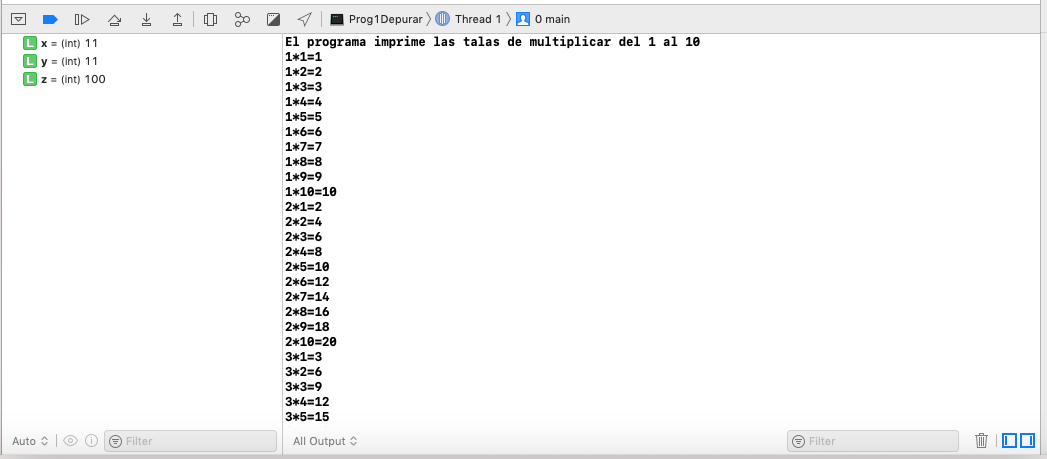
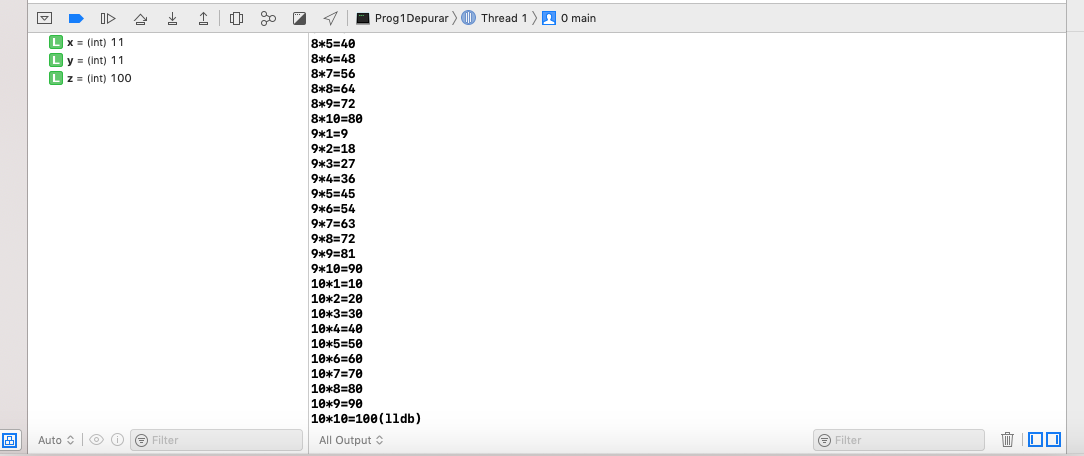
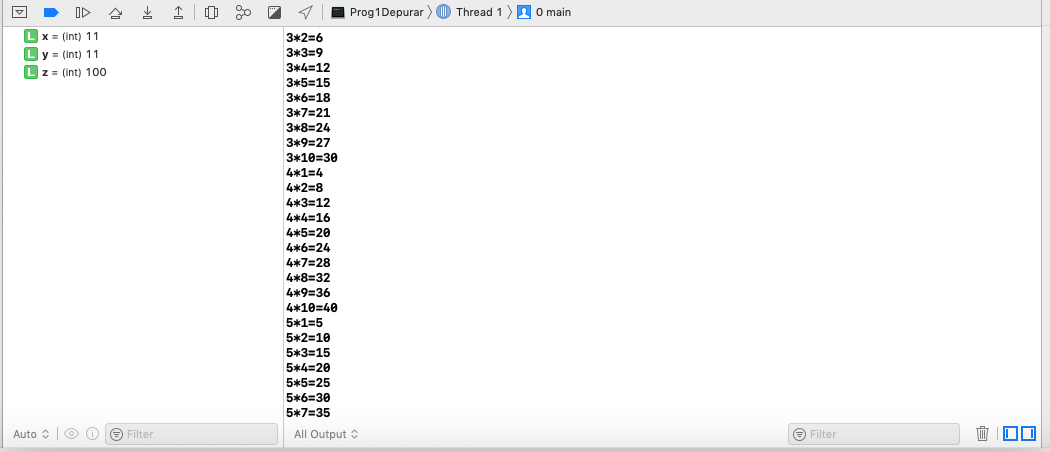
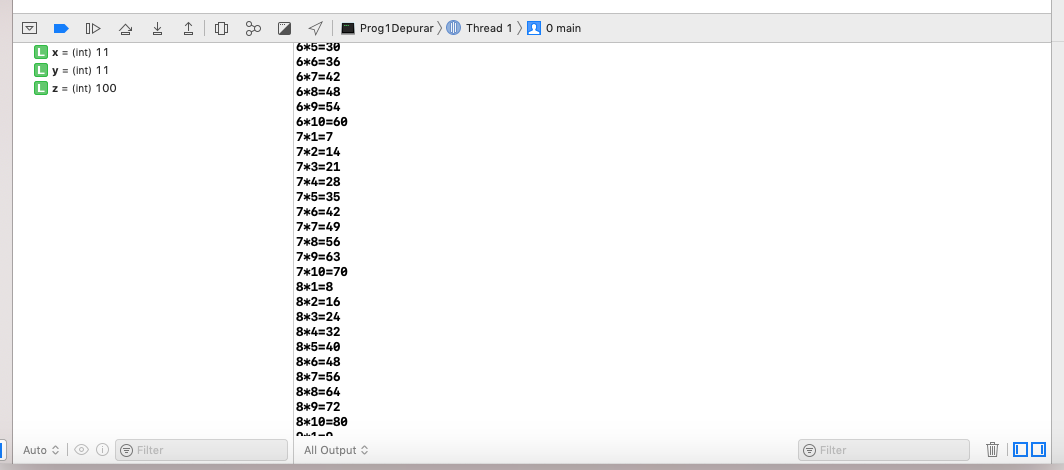
****

Puntos de ruptura

Una vez validada la ejecución se indicaron unos **puntos de ruptura**, estos se realizan 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. Posteriormente 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.

Añadir variables

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.

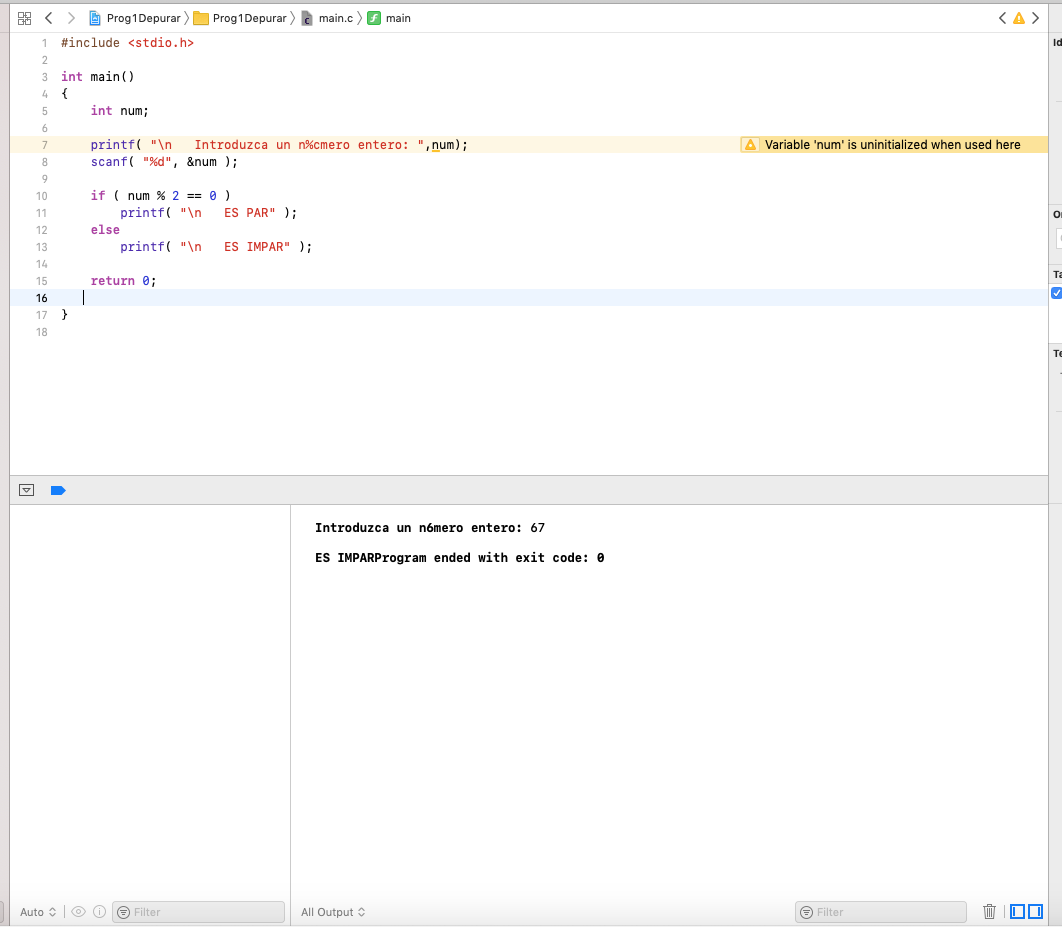
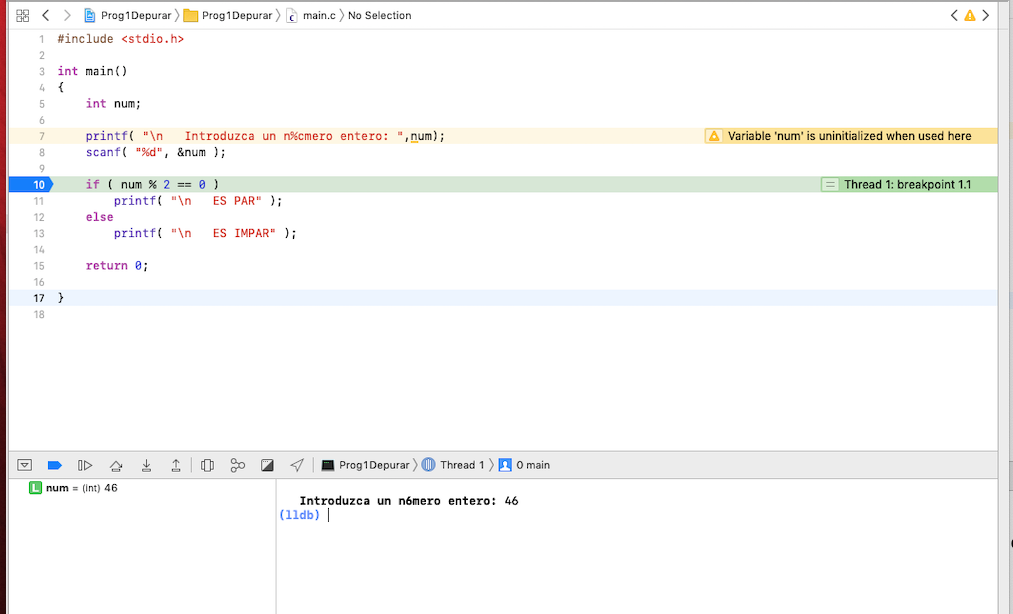
** **

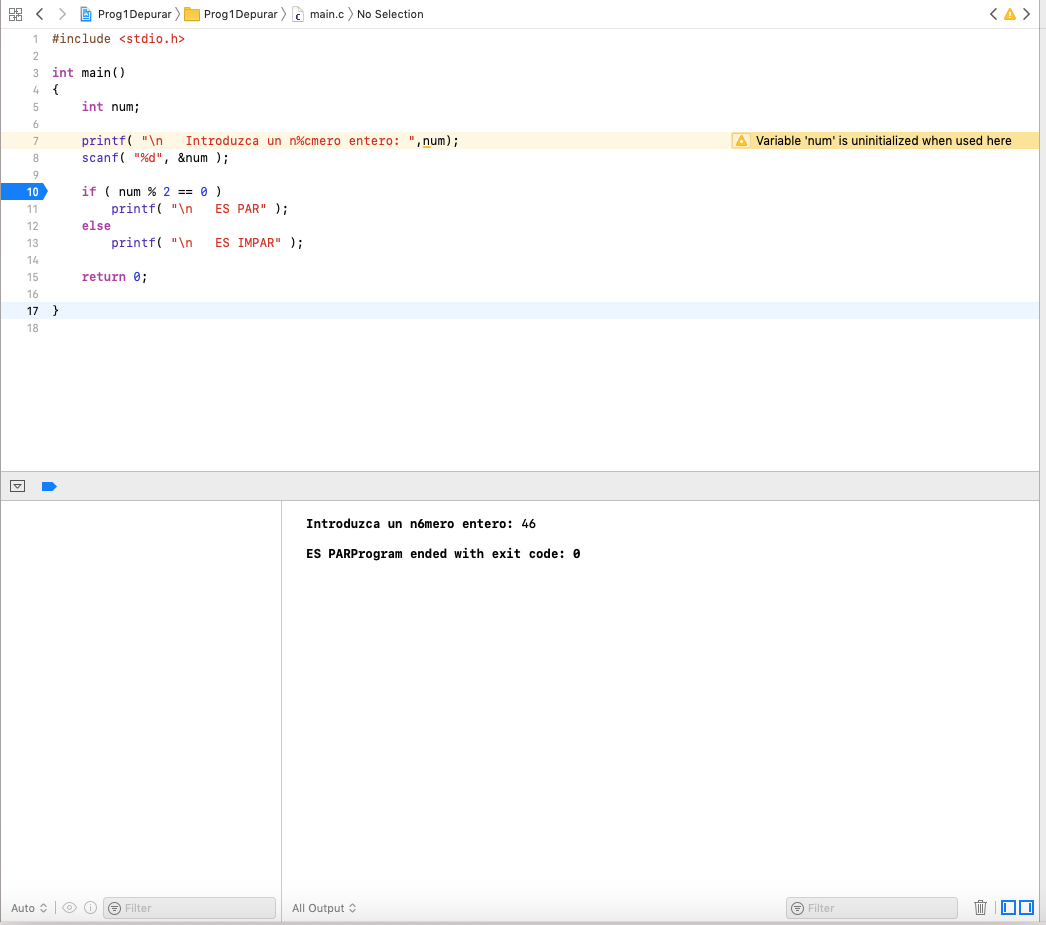
Ejecución del programa

**PROGRAMA 2:** El siguiente programa deja entrar un número para determinar si es par o impar.

Se emplearon las estructuras printt, scanf, if-else y return. Después de editar el programa siguió la compilación y ejecución en las cuales no se presentaros errores.

Comienza depuración. Se marcaron los puntos de ruptura

****

El programa realiza la siguiente instrucción. El programa sigue su transcurso correctamente.

Ejecución del programa

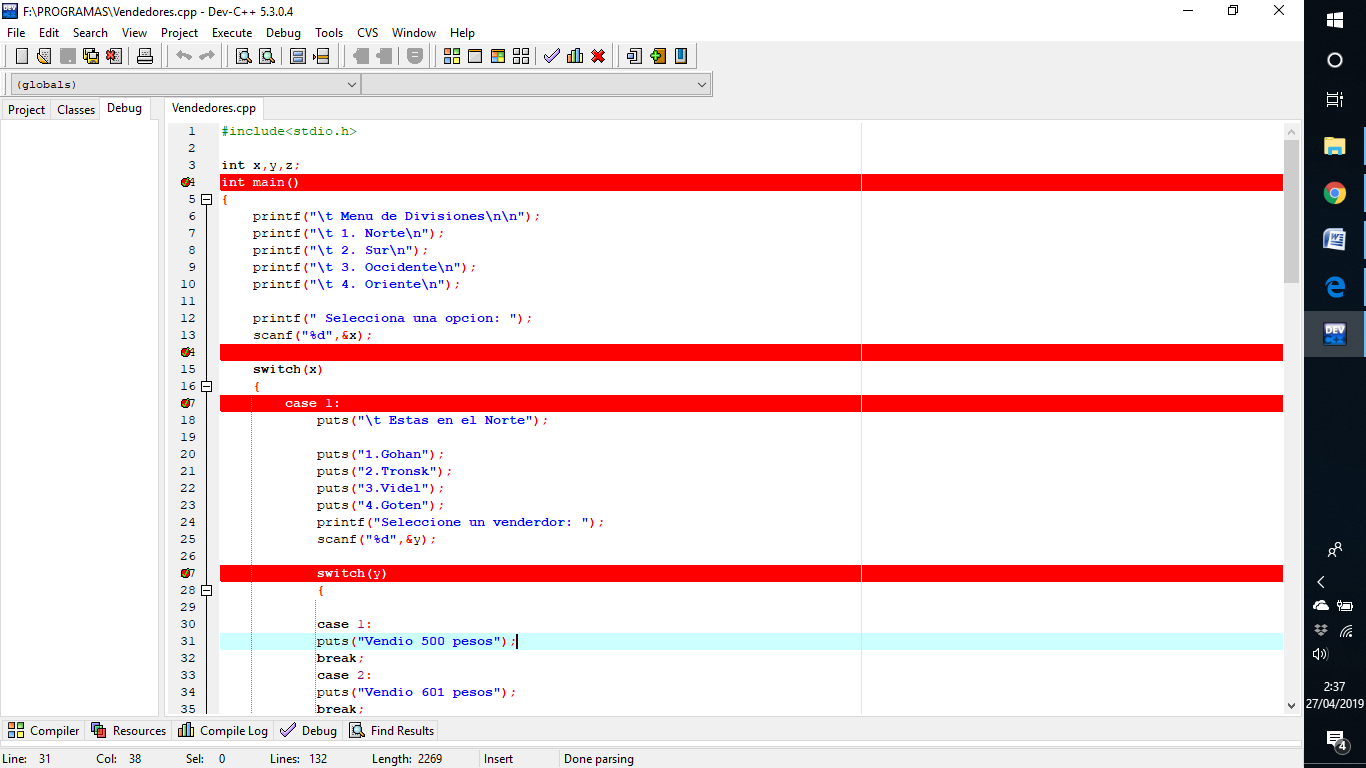
**PROGRAMA 3:** Este programa muestra la cantidad de dinero vendida de queda vendedor.

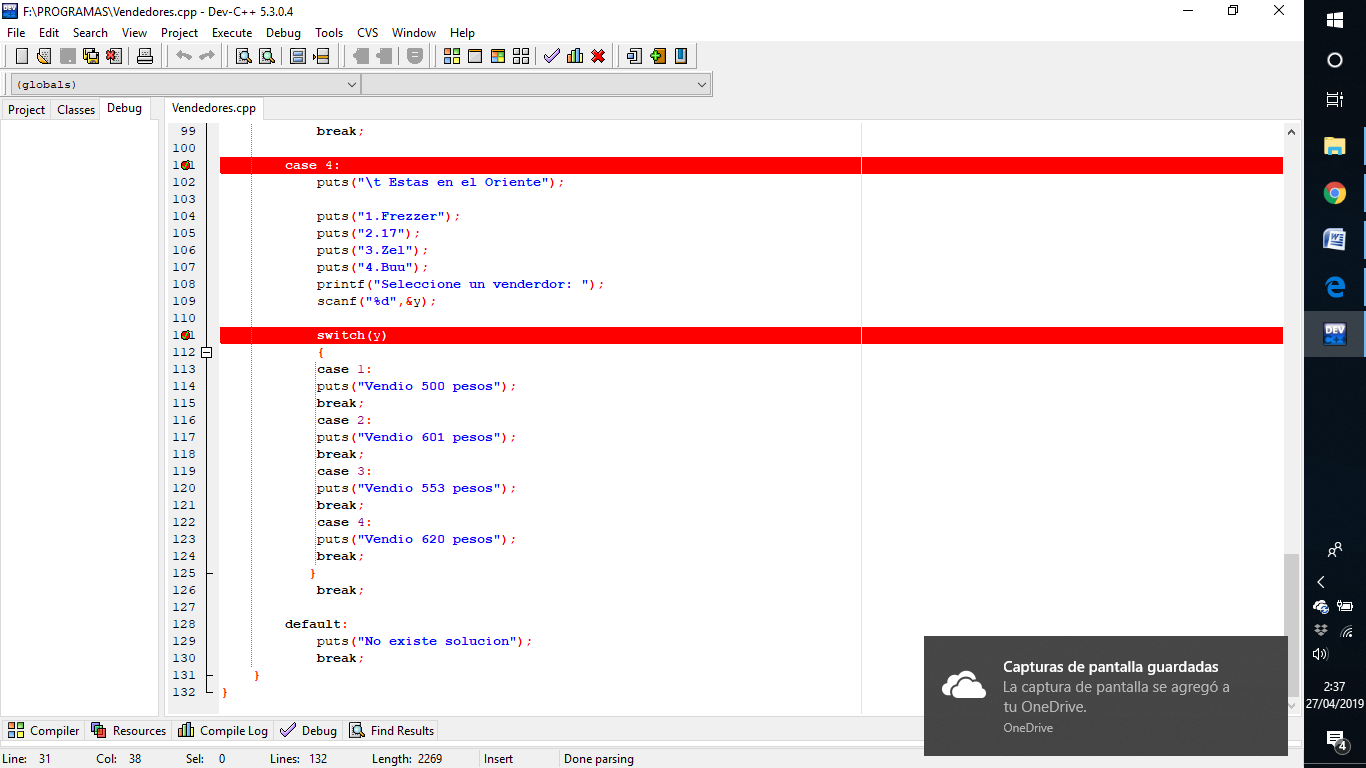
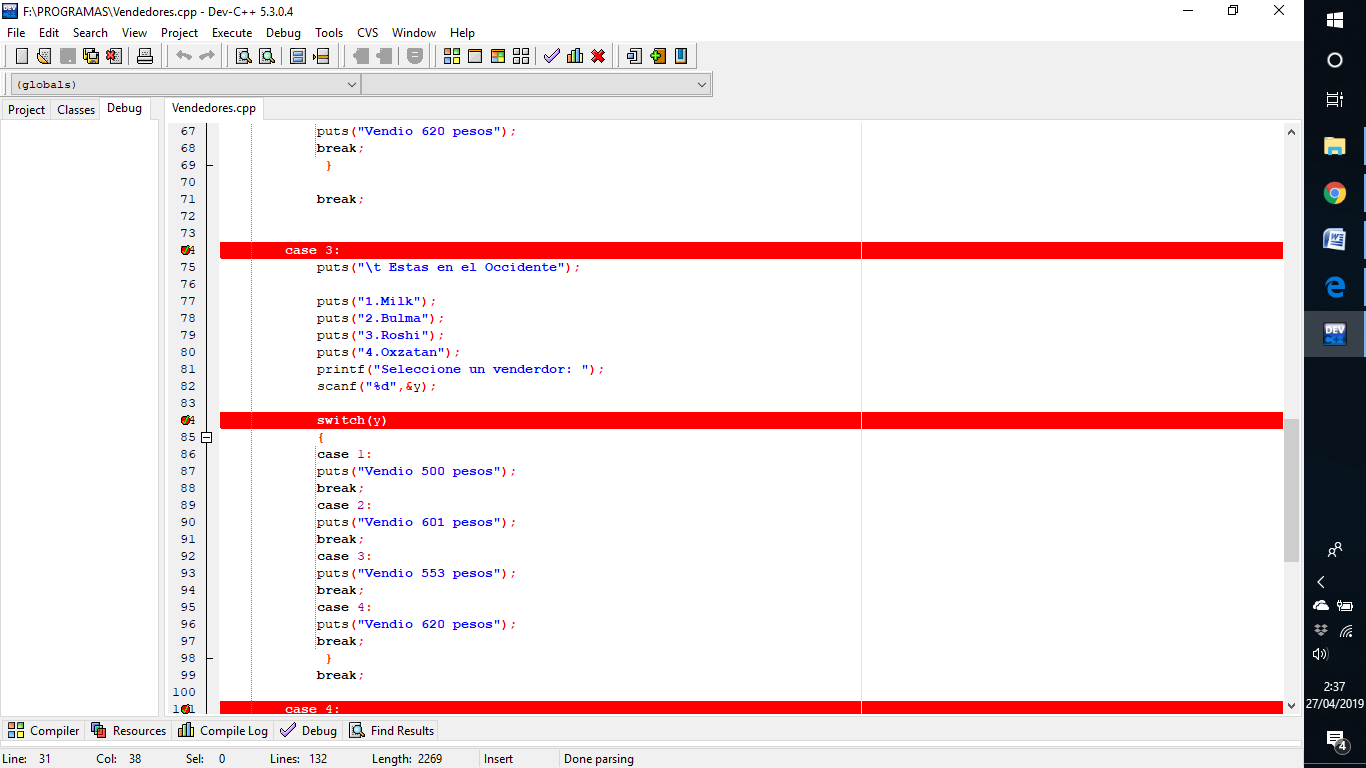
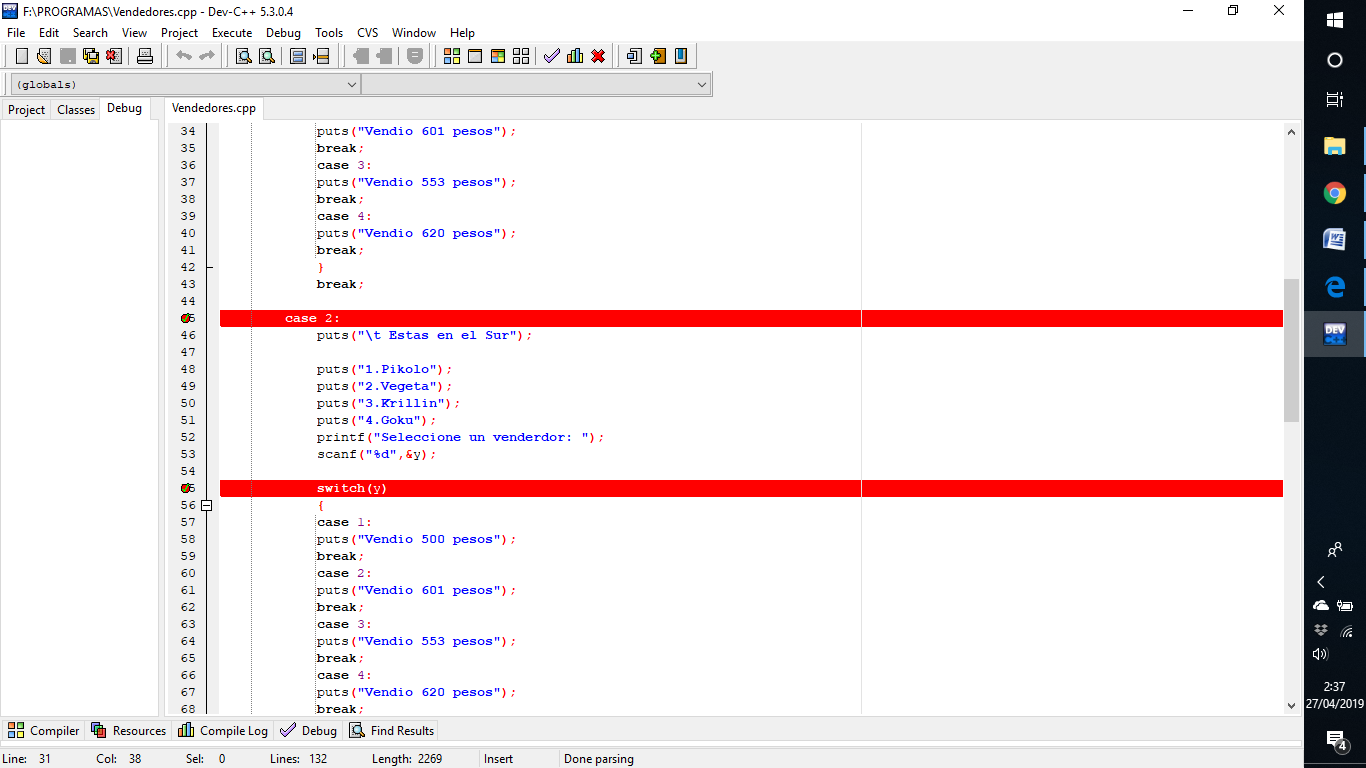
Se realizo en Dev-C++. En este caso se empleo switch-case, break, entre otros operadores.

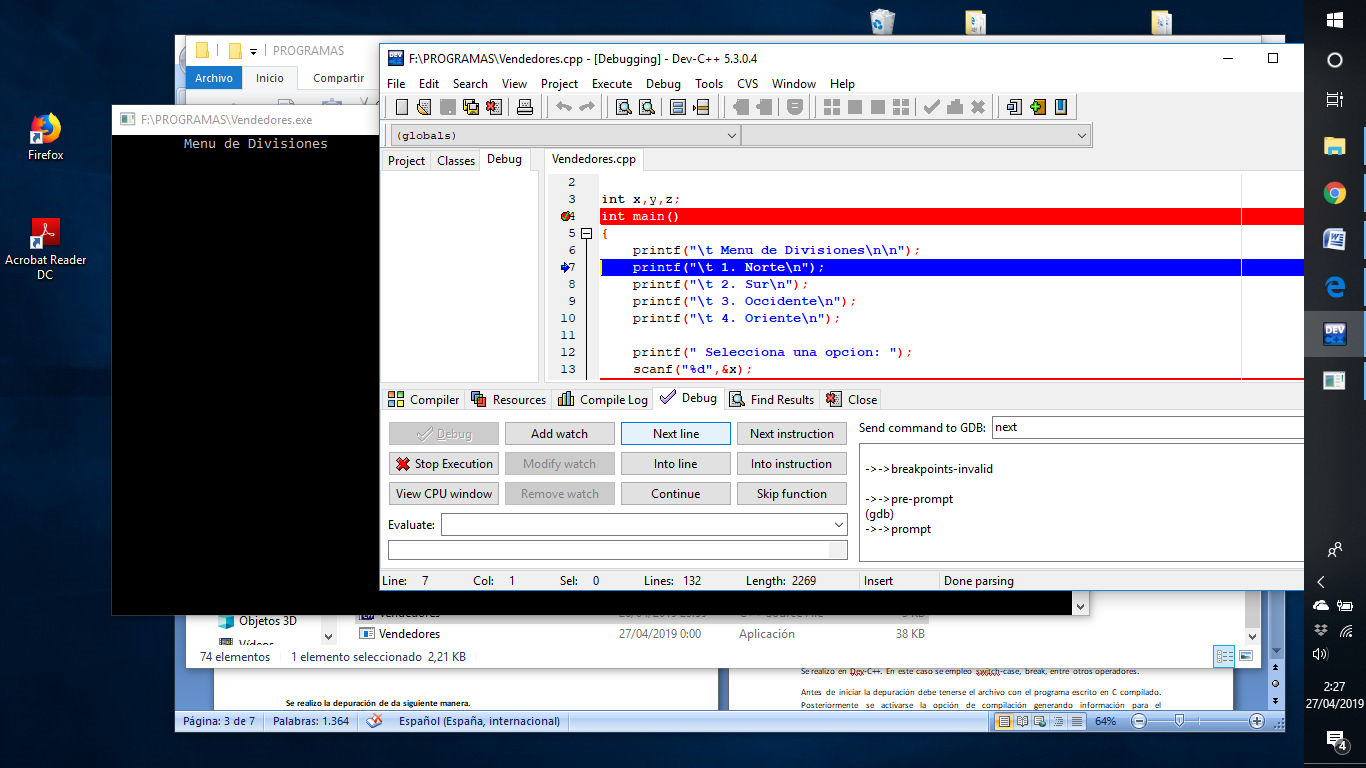
**Antes de iniciar la depuración debe tenerse el archivo con el programa escrito en C compilado.** Posteriormente se activarse la opción de compilación generando información para el depurador. Para activar esta opción debe abrirse el menú Herramientas → Opciones del Compilador y acceder a la pestaña Generación/Optimización de Código y finalmente, en la subpestaña Enlazador (linker), activar la opción Generar Información de Depuración. Después de realizar lo anterior, el programa puede compilarse y ejecutarse con lo que ofrece el menú Ejecutar.

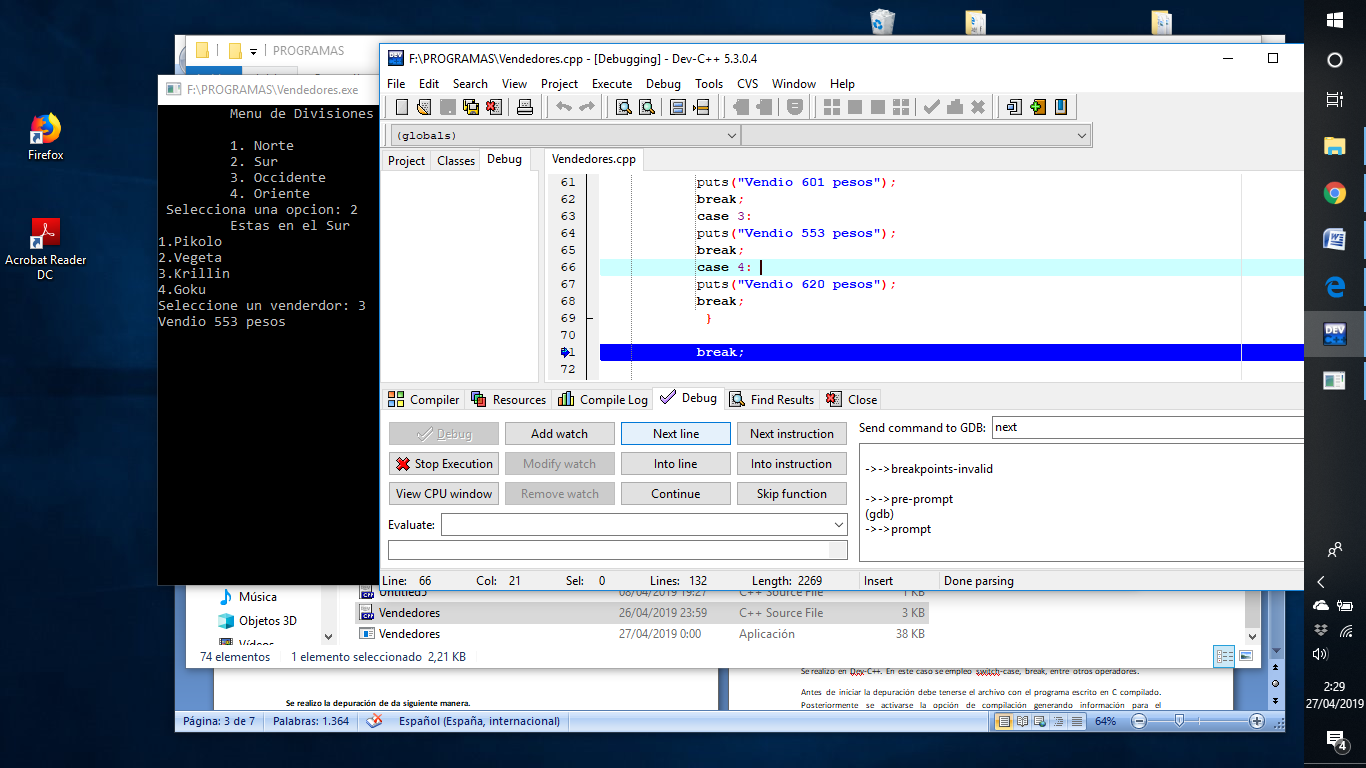
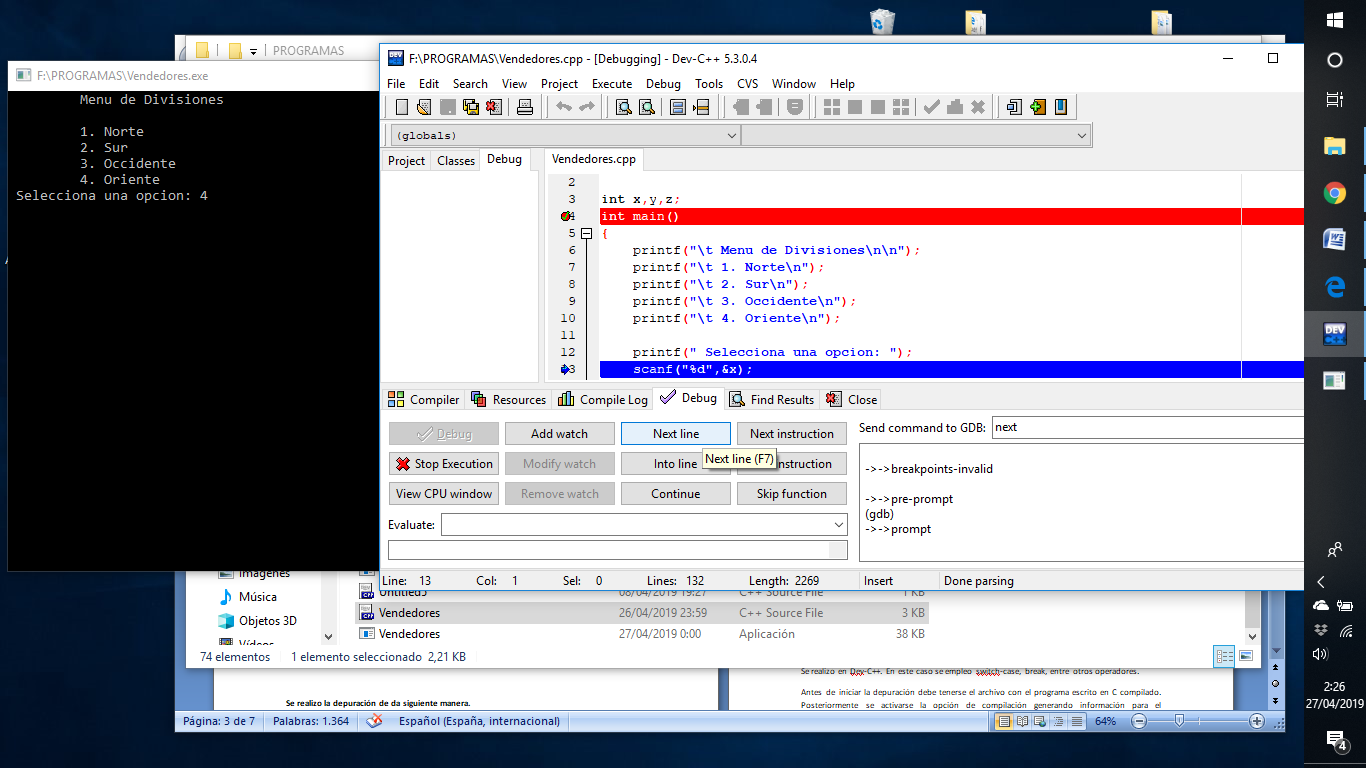
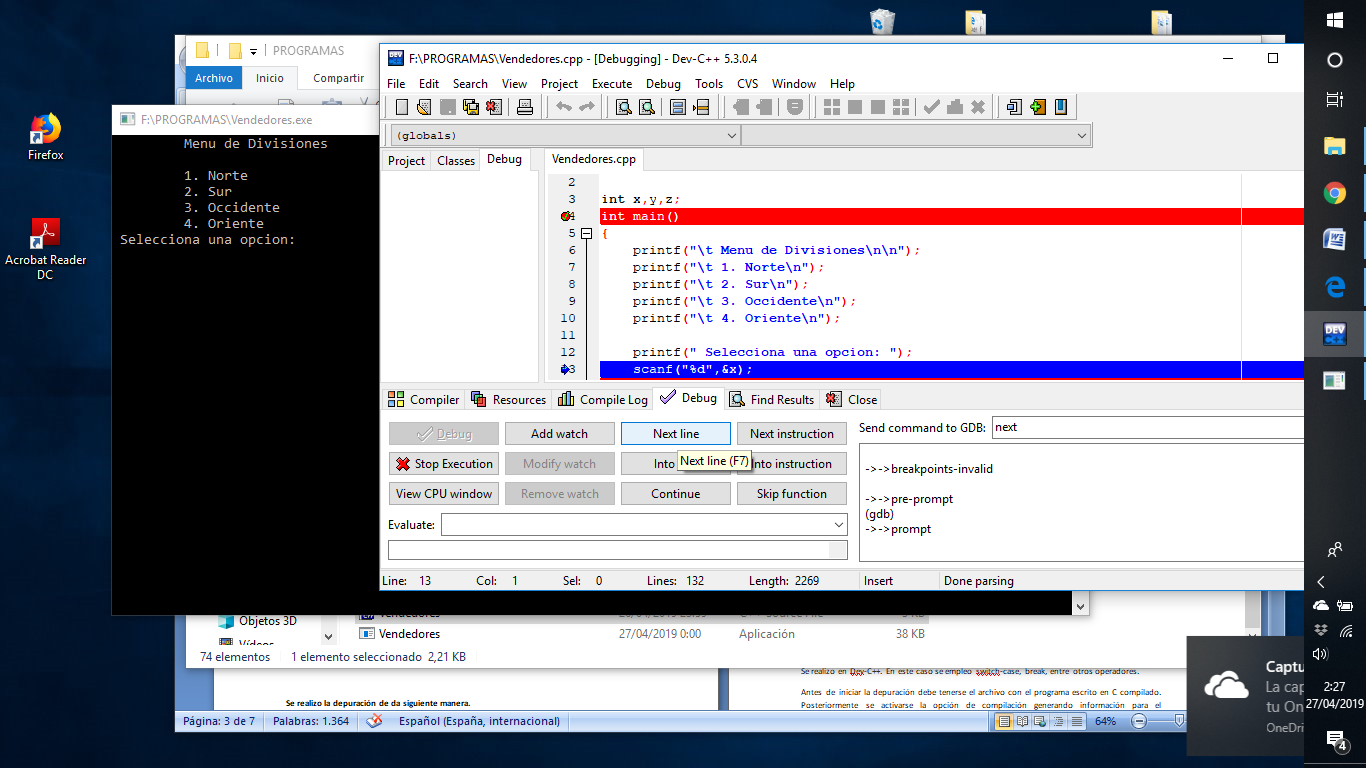
Para agregar puntos de ruptura, debe hacerse clic en la línea de código donde se desea colocar y ésta se volverá en **color rojo**. Para retirarlo se hace clic de nuevo en la línea.

Coloque los puntos en los renglones que creí convenientes.





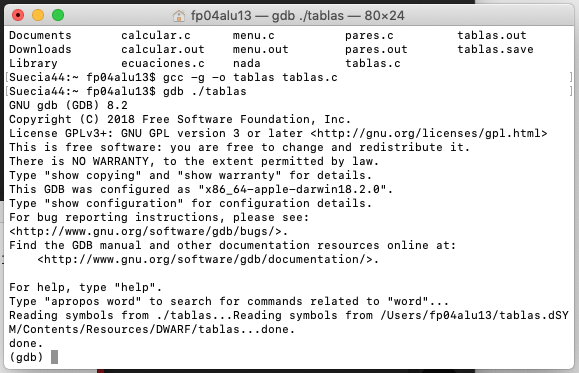
Después de marcar los puntos de ruptura utilice la tecla Debug para comenzar la depuración, cada vez que quería realizar la instrucción oprimía el botón Nex Line, así el programa fue depurando las paso a paso. Como se muestra en las imágenes.



Linea que se ejecuta

**COMENTARIO:**

Intente realizar la depuración en la terminal GNU como se indicaba en la práctica, pero el resultado no fue el esperado. Aquí muestro lo que sucedió, lo que me lleva a pensar que se necesitan hacer mas pasos antes de continuar con la depuración.

****

**CONCLUSIONES:**

El uso de la depuración me llevo a pensar en que existen casos en los que no alcanzamos a reconocer si existe un error en el programa, pensamos que el programa está bien pero la manera de ejecutarse es errónea o simplemente no corre, es decir, el programa compila pero la ejecución no es correcta entonces nos hacemos la pregunta ¿Qué está fallando? Y es precisamente cuando hacemos uso de la depuración. Este nos ayuda a reconocer en donde se presenta la falla o el error, al mismo tiempo nos brinda una visualización de cómo se puede reducir y mejorar el programa. La depuración es un concepto nuevo pero no complicado. En mi experiencia fue más fácil depurar en Xcode y Dev-C++ porque como ya mencione anteriormente trate de usar la terminal pero el resultado no me favoreció.

Es conveniente no marcar MUCHOS puntos de ruptura porque eso imprecaría muchos play o Nex Line, por eso se recomienda hacerlo en puntos estratégicos.