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

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Karina García Morales |
| *Asignatura:* | Fundamentos de Programación |
| *Grupo:* | 1121 |
| *No de Práctica(s):* | Práctica no. 10 |
| *Integrante(s):* | Espinosa Carrillo Luis Andrés |
|  |  |
| *No. de Equipo de cómputo empleado* | Equipo no. 20 |
| *Semestre:* | 2019-1 |
| *Fecha de entrega:* |  |
| *Obervaciones:* |  |
|  |  |

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.

**Desarrollo:**

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.

* **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.

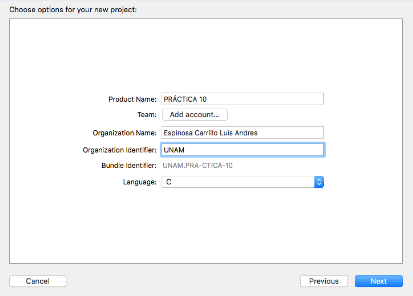
La depuración de un programa es útil cuando:

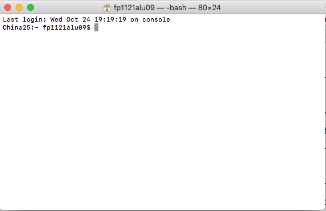
* **Se desea optimizar el programa:** no basta que el programa se pueda compilar y se someta a pruebas que demuestren que funciona correctamente.
* **El programa tiene algún fallo:**el programa no muestra los resultados que se esperan para cierta entrada de datos debido a que el programador cometió algún error durante el proceso de diseño.
* **El programa tiene un error de ejecución o defecto**

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

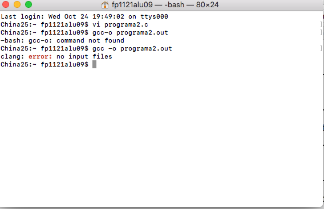
* **Ejecutar el programa:** se procede a ejecutar el programa en la herramienta de depuración ofreciendo diversas opciones para ello.
* **Mostrar el código fuente del programa:**muestra cuál fue el código fuente del programa con el número de línea con el fin de emular la ejecución del programa sobre 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 br*eakpoint,*sirve para detener la ejecución del programa en algún punto indicado previamente por medio del número de línea.
* **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: de**shace el efecto provocado por alguna de las funciones anteriores para repetir una sección del programa.
* **Visualizar el valor de las variables:** permite conocer el valor de alguna o varias variables.

Al copiar un ejercicio en siaefi y compilarlo nos marcó que había error:

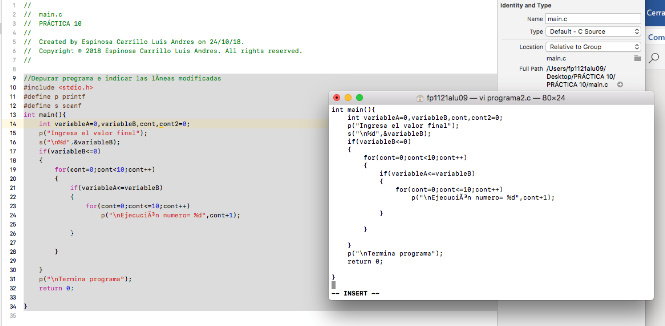




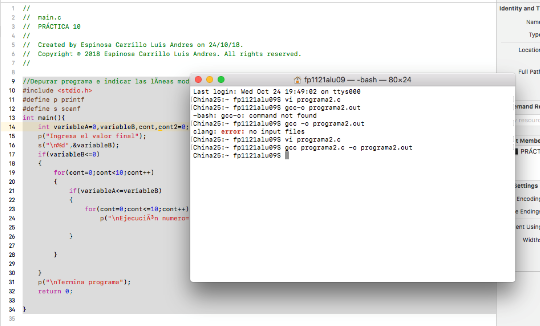




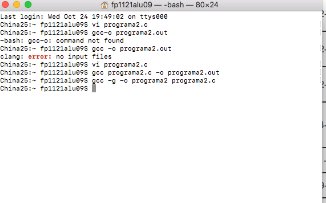
Después corregimos el programa:



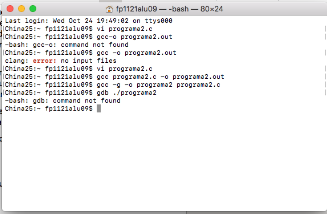
Compilamos el programa:



Depuramos el programa:

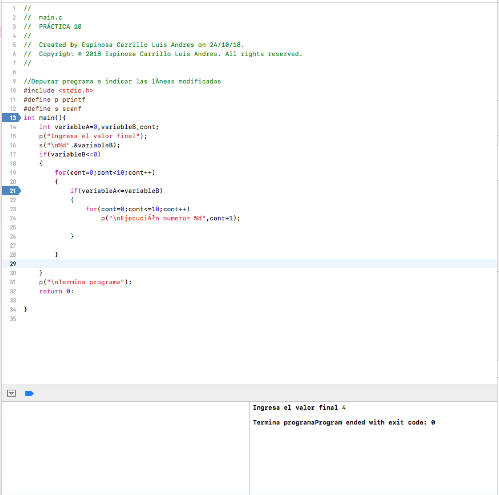


Iniciamos el depurador, pero no teníamos ese depurador:

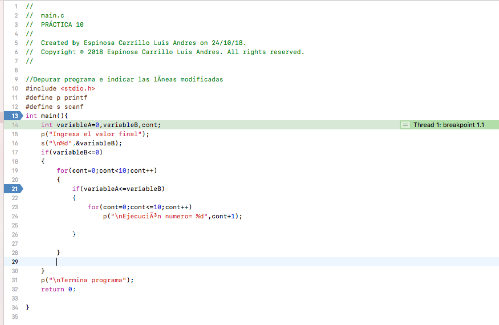
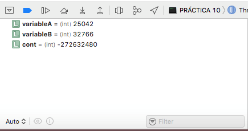


* ***list*o *l****:*Permite listar diez líneas del código fuente del programa, si se desea visualizar todo el código fuente debe invocarse varias veces este comando para mostrar de diez en diez líneas. Se puede optar por colocar un número separado por un espacio para indicar a partir de qué línea desea mostrarse el programa. También es posible mostrar un rango de líneas introduciendo el comando y de qué línea a qué línea separadas por una coma. Ejemplo: list
* ***b****:*
* ***d*o *delete****:*Elimina un punto de ruptura, indicando cuál es el que debe eliminarse usando el número de línea. Ejemplo: d 5
* ***clear****:*Elimina todos los puntos de ruptura. Ejemplo: *clear*
* *i****nfo line****:*Permite mostrar información relativa a la línea que se indique después del comando. Ejemplo: info*line 8*
* ***run*o *r:***Ejecuta el programa en cuestión. Si el programa tiene un punto de ruptura se ejecutará hasta dicho punto, de lo contrario se ejecutará todo el programa.
* ***c****:*Continúa con la ejecución del programa después de un punto de ruptura.
* ***s****:*Continúa con la siguiente instrucción después de un punto de ruptura.
* ***n****:*Salta hasta la siguiente línea de código después de un punto de ruptura.
* ***p*o *print****:*Muestra el valor de una variable, para ello debe escribirse el comando y el nombre de la variable separados por un espacio. Ejemplo: *p suma\_acumulada*
* ***ignore****:*Ignora un determinado punto de ruptura indicándolo con el número de línea de código. Ejemplo: igno*re 5*
* ***q*o *quit****:*Termina la ejecución de GDB.

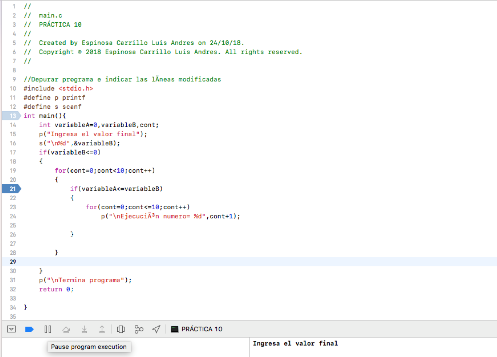
Colocamos puntos de ruptura:



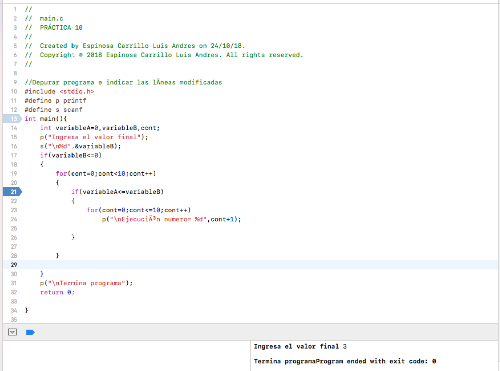
Nos mostró las variables con su tipo de dato y su valor:

Se sigue ejecutando el programa hasta llegar al siguiente breakpoint:

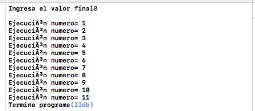


Hicimos un cambio de variable:

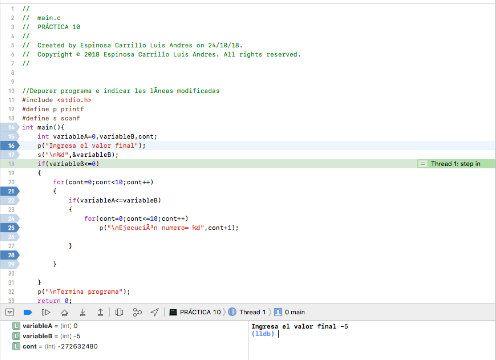


Continuamos ejecutando el programa hasta que el contador fuera igual a 10:





Quitamos el primer ciclo for ya que no importa si está o no en el programa para ejecutarse:



Usamos el depurador para acortar líneas de código, liberamos espacio en la memoria, puntos de ruptura, para hacer más eficiente el programa, e ir en el programa pasa por paso, variables extra que no utilizamos, los valores de las variables nos ayudan a darnos cuenta de que podemos depurar en el programa.

**Ejercicios de Tarea:**

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.



**Este programa nos muestra la suma de un número ingresado por el usuario y un aumento en el contador iniciado en 1 y con aumento de 2. Hubo un error de no declarar a la función principal.**

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.



**Muestra las tablas de multiplicar del 1 al 10. Había un error al colocar <=10 en cada ciclo for y una llave que había de más.**

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



**Había un error de no poner un & en los scanf y en no haber declarado la función principal.**



**Lo que se le puede mejorar a este programa es el ahorro de espacio en la impresión de la matriz para mostrar los valores dados por los usuarios.**

**Conclusiones:**

En conclusión puedo decir que la práctica no fue tan difícil, el problema fue que al principio no entendía muy bien como funcionaba bien el depurador y de cómo descubrir los errores del programa, de hecho me costó algo de trabajo al momento de realizar los ejercicios de tarea por lo que tuve que pedir un poco de ayuda.

**Bibliografía**

* 􏰀  Gutiérrez Rodríguez, Javier Jesús. Pri*meros pasos con GDB. C*onsulta: octubre de 2016. Disponible en: ﷟HY[p://www.lsi.us.es/~javierj/ssoo\_ficheros/GuiaGDB.htm](http://p/www.lsi.us.es/~javierj/ssoo_ficheros/GuiaGDB.htm%22ht)
* 􏰀  Ferreira, Amelia. *Depurador gdb*. Consulta: octubre de 2016. Disponible en: <http://learnassembler.com/gdbesp.html>
* 􏰀  Ferreira, Amelia. *Depurador gdb - uso de la opción -g de gcc.* Consulta: octubre de 2016. Disponible en: ﷟[ttp://learnassembler.com/opc.html](http://ttp/learnassembler.com/opc.html%22" \t "_blank)
* 􏰀  Gutiérrez, Erik Marín. De*puración de programas Dev C++. C*onsulta: octubre de 2016. Disponible en: ﷟HY[p://programacionymetodos.blogspot.mx/2012/05/depuracion- de](http://p/programacionymetodos.blogspot.mx/2012/05/depuracion-%22ht)-programas-dev-c.html
* 􏰀  González Cárdenas, Miguel Eduardo; Marín Lara, Claudia Lorena; Noguerón Pérez, Pedro. Apunt*es De Computadoras Y Programación. Univ*ersidad Nacional Autónoma de México.
* 􏰀  Pozo Coronado, Salvador. Primeros pasos con GDB. Consulta: octubre de 2016. Disponible en: <http://www.c.conclase.net/devcpp/?cap=depurar>