	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: *Rodríguez Espino Claudia Ing.*

Asignatura: *Fundamentos de Programación*

Grupo: *1104*

No de Práctica(s): *10*

Integrante(s): *Santa Rosa Ortiz Thelma Jazmín.*

No. de Equipo de cómputo empleado *50*

Semestre: *1°*

Fecha de entrega: *20 de octubre del 2018*

Obervaciones:

CALIFICACIÓN: _____

Practica 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.

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.

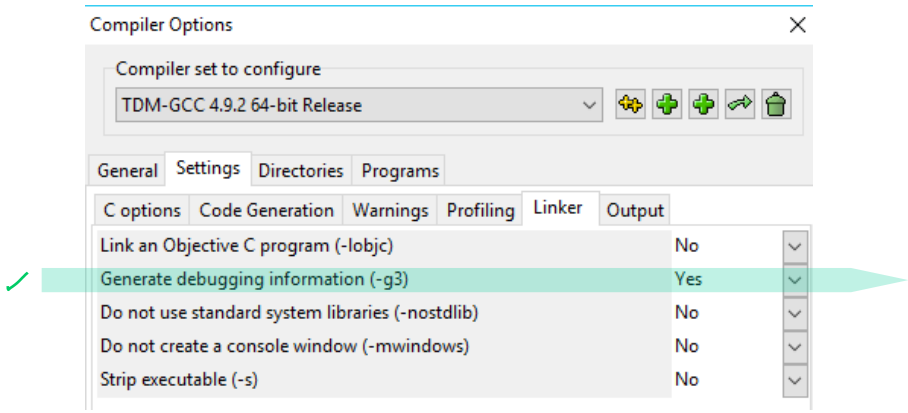
En las IDE (Entornos de Desarrollo Interactivo), suelen existir herramientas de depuración integradas de manera gráfica. Es muy común que existan dos modos de desarrollar un programa y producir el archivo ejecutable que son "Debug" y "Release". El primer modo se recomienda exclusivamente durante el desarrollo 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.

Depuración de programas escritos en C con Dev-C++ 5.0.3.4

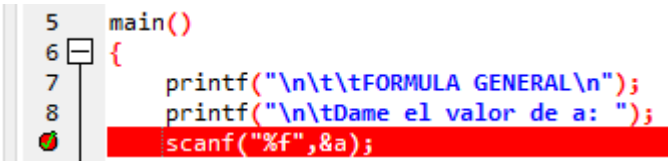
Dev-C++, es una IDE especializada para desarrollar programas escritos en C o C++. Si bien incorpora un editor de textos y un compilador integrados, también posee un depurador. Cabe destacar que por defecto Dev-C++ se basa en el compilador GCC y el depurador en GDB, aunque de manera gráfica ello es transparente para el usuario ya que todo simula una sola herramienta.

Antes de iniciar la depuración debe tenerse a la mano el archivo con el programa escrito en C o proceder a escribirlo en la misma IDE. Para ello debe usarse el menú *Archivo → Nuevo → Código Fuente* si se piensa usar la IDE para escribirlo o en su lugar *Archivo → Abrir Proyecto o Archivo* si ya existía el código fuente.

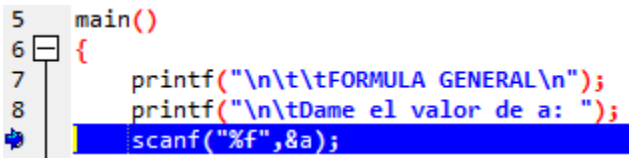
Una vez que se tiene el programa, debe activarse la opción de compilación generando información para el depurador. Para activar esta opción debe abrirse el menú Herramientas → Opciones del Compilador y acceder a la pestaña *Generación/Optimización de Código* y finalmente, en la subpestaña Enlazador (linker), activar la opción Generar Información de Depuración:



Después de realizar lo anterior, el programa realizado 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 y volverá a su color normal.



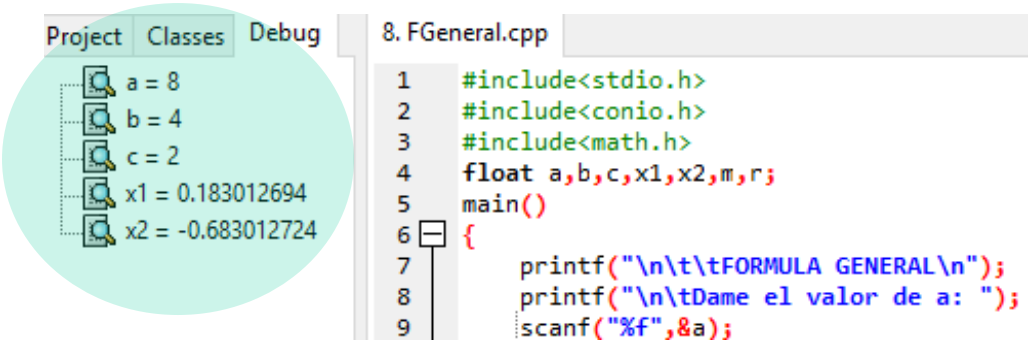
Para depurar el programa, primero se debe compilar con el menú *Ejecutar → Compilar* y luego depurar con *Depurar → Depurar.* El programa se abrirá y se ejecutará hasta el primer punto de ruptura seleccionado. También se abrirá un cuadro de herramientas en la parte inferior del programa que tiene las principales herramientas de depuración en la parte derecha. Cabe destacar que la línea que se ejecuta actualmente es la que se resalta en color azul:



Cuando se llega al punto de ruptura, se tienen diversas opciones, Siguiente Paso ejecuta la siguiente línea (si existe una iteración o función, la saltará), Avanzar Paso a Paso ejecuta instrucción por instrucción (una función o iteración serán ejecutadas instrucción por instrucción), Saltar Paso ejecuta hasta el siguiente punto de ruptura.

Para detener la depuración puede seleccionarse la opción Parar ejecución. La opción Ver ventana del CPU permite ver a detalle las instrucciones enviadas al procesador, registros de memoria involucrados y valor de cada una de las banderas en el procesador.

Finamente, para estudiar el valor de cada variable, se puede recurrir a la función Añadir Watch y escribir el nombre de la variable. En un cuadro a la izquierda, se verá el nombre de la variable y su valor hasta el punto donde se está ejecutando el programa:



Cuando se utiliza esta IDE, se recomienda usar los accesos directos mencionados en los propios menús, lo cual permite usarla de manera óptima. El nombre de las funciones y menús pueden variar según el idioma en el que se haya instalado la IDE.

-Al leer toda la teoría sobre esta practica y en especial de la depuración en Dev-C++ depuramos 3 de los programas que ya habíamos hecho en la clase de fundamentos de programación, unos compañeros lo tenían sin terminar o no compilaba entonces la depuración les sirvió para que ejecutaran sus programas, pero, en mi caso ya estaban bien hechos solo vi que funcionara bien y observe los valores que tomaban mis variables.

FORMULA GENERAL

----------Cuando la raíz es imaginaria----------

Project Classes Debug

- a = 8
- b = 4
- c = 2
- x1 = 0.183012694
- x2 = -0.683012724

8. FGeneral.cpp

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
float a,b,c,x1,x2,m,r;
main()
{
    printf("\n\t\tFORMULA GENERAL\n");
    printf("\n\tDame el valor de a: ");
    scanf("%f",&a);
    printf("\tDame el valor de b: ");
    scanf("%f",&b);
    printf("\tDame el valor de c: ");
    scanf("%f",&c);
    if (a>0)
    {
        m=(-b/(2*a));
        r=((b*b)-(4*a*c));
        if(r>=0)
        {
            r=sqrt(r);
            r=(r/(2*a));
            x1=(m+r);
            printf("\tx1 es igual a: %.2f", x1);
            x2=(m-r);
            printf("\n\tx2 es igual a: %.2f", x2);
        }
        else
        {
            puts("\n\t-La raiz es imaginaria-\a");
            r=(r*-1);
            r=sqrt(r);
            r=(r/(2*a));
            x1=(m+r);
            printf("\tx1 es igual a: %.2f", x1);
            x2=(m-r);
            printf("\n\tx2 es igual a: %.2f", x2);
        }
    }
    getch();
}
```

E:\Programacion\8. FGeneral.exe

```
                FORMULA GENERAL

        Dame el valor de a: 8
        Dame el valor de b: 4
        Dame el valor de c: 2

        -La raiz es imaginaria-
        x1 es igual a: 0.18
        x2 es igual a: -0.68
```

---------- Cuando la raíz existe----------

Project Classes Debug

- a = 2
- b = 8
- c = 4
- x1 = -0.585786462
- x2 = -3.41421366

8. FGeneral.cpp

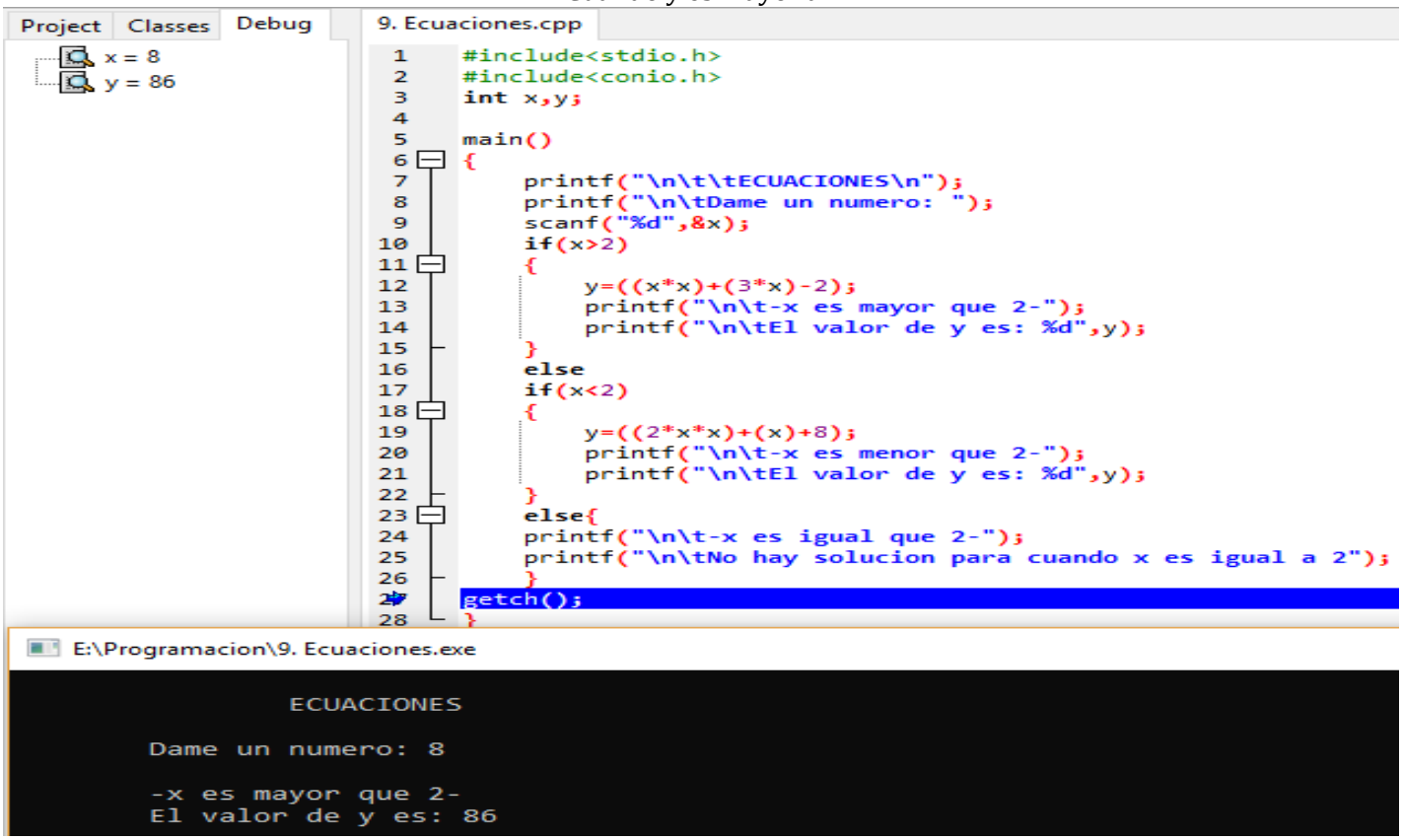
```
#include<stdio.h>
#include<conio.h>
#include<math.h>
float a,b,c,x1,x2,m,r;
main()
{
    printf("\n\t\tFORMULA GENERAL\n");
    printf("\n\tDame el valor de a: ");
    scanf("%f",&a);
    printf("\tDame el valor de b: ");
    scanf("%f",&b);
    printf("\tDame el valor de c: ");
    scanf("%f",&c);
    if (a>0)
    {
        m=(-b/(2*a));
        r=((b*b)-(4*a*c));
        if(r>=0)
        {
            r=sqrt(r);
            r=(r/(2*a));
            x1=(m+r);
            printf("\tx1 es igual a: %.2f", x1);
            x2=(m-r);
            printf("\n\tx2 es igual a: %.2f", x2);
        }
        else
        {
            puts("\n\t-La raiz es imaginaria-\a");
            r=(r*-1);
            r=sqrt(r);
            r=(r/(2*a));
            x1=(m+r);
            printf("\tx1 es igual a: %.2f", x1);
            x2=(m-r);
            printf("\n\tx2 es igual a: %.2f", x2);
        }
    }
    getch();
}
```

E:\Programacion\8. FGeneral.exe

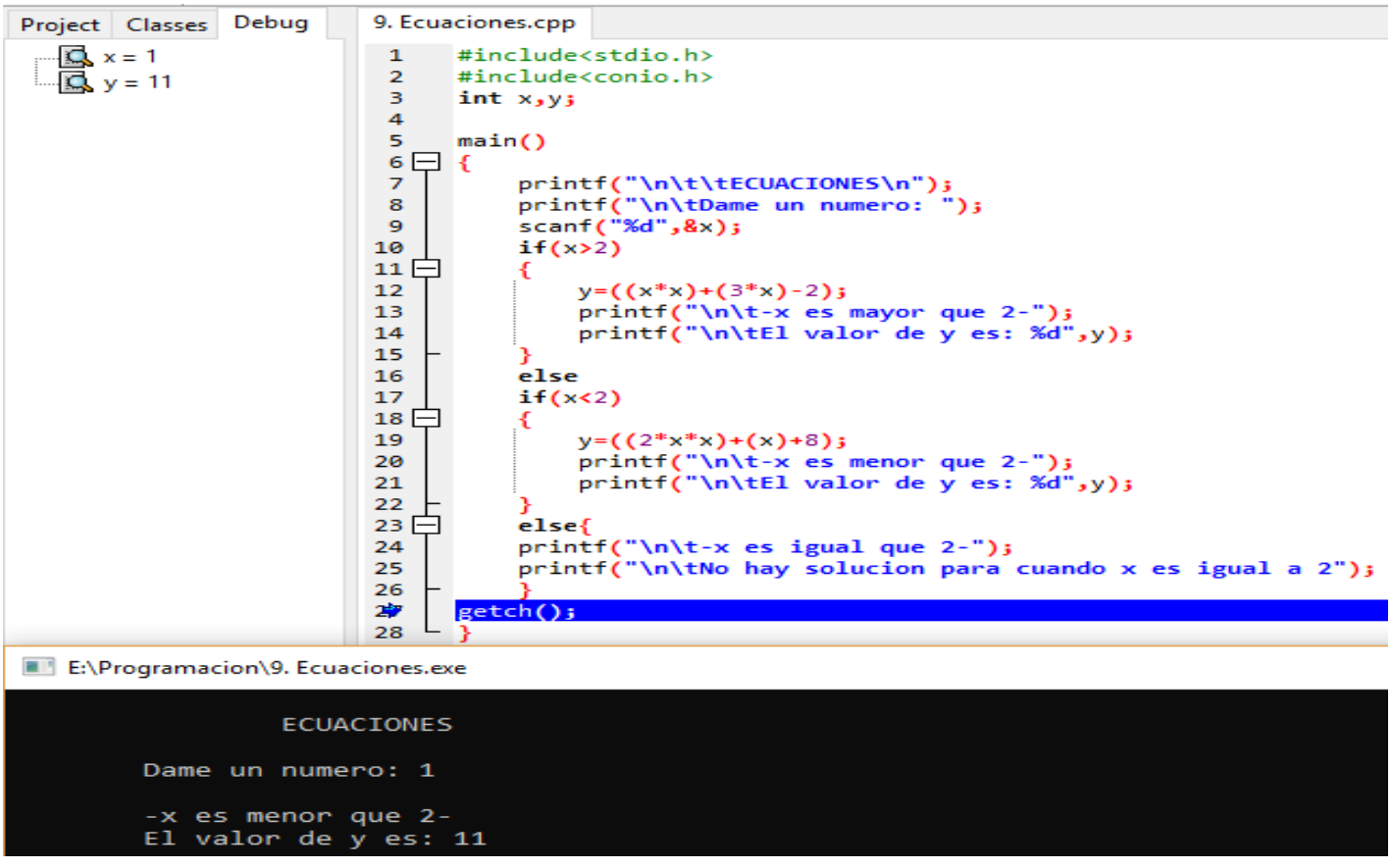
```
                FORMULA GENERAL

        Dame el valor de a: 2
        Dame el valor de b: 8
        Dame el valor de c: 4
        x1 es igual a: -0.59
        x2 es igual a: -3.41
```

ECUACIONES

----------Cuando y es mayor a 2----------



----------Cuando y es menor a 2----------



----------Cuando y es igual a 2----------

Project | Classes | Debug

x = 2
y = 0

9. Ecuaciones.cpp

```
#include<stdio.h>
#include<conio.h>
int x,y;

main()
{
    printf("\n\t\tECUACIONES\n");
    printf("\n\tDame un numero: ");
    scanf("%d",&x);
    if(x>2)
    {
        y=((x*x)+(3*x)-2);
        printf("\n\t-x es mayor que 2-");
        printf("\n\tEl valor de y es: %d",y);
    }
    else
    if(x<2)
    {
        y=((2*x*x)+(x)+8);
        printf("\n\t-x es menor que 2-");
        printf("\n\tEl valor de y es: %d",y);
    }
    else
    printf("\n\tNo hay solucion para cuando x es igual a 2");
getch();
}
```

E:\Programacion\9. Ecuaciones.exe

```
            ECUACIONES

      Dame un numero: 2

      No hay solucion para cuando x es igual a 2
```

GASTOS MENSUALES

Project | Classes | Debug

a = 12
s = 18666
prom = 1555.5

18. Control de gastos mensuales.cpp

```
#include<stdio.h>
#include<conio.h>
int a;
float arr[12], prom=0, s=0;
main()
{
    printf("\n\t\tCONTROL DE GASTOS MENSUALES\n\n");
    for(a=1; a<13; a++)
    {
        printf("\tDime el gasto del mes %d: $", a);
        scanf("%f", &arr[a]);
        s=(s+arr[a]);
    }
        a--;
        prom=(s/a);
        printf("\n\t-El promedio anual es: $%f", prom);
    getch();
}
```

E:\Programacion\18. Control de gastos mensuales.exe

```
            CONTROL DE GASTOS MENSUALES

      Dime el gasto del mes 1: $600
      Dime el gasto del mes 2: $980.5
      Dime el gasto del mes 3: $963
      Dime el gasto del mes 4: $2069.5
      Dime el gasto del mes 5: $9654
      Dime el gasto del mes 6: $785
      Dime el gasto del mes 7: $692
      Dime el gasto del mes 8: $450
      Dime el gasto del mes 9: $600
      Dime el gasto del mes 10: $841
      Dime el gasto del mes 11: $246
      Dime el gasto del mes 12: $785

      -El promedio anual es: $1555.500000
```

Conclusión:

En esta práctica de laboratorio aprendimos las distintas técnicas básicas de depuración de programas en C, para, poder revisar de manera precisa el flujo de ejecución de un programa y el valor de las variables. Todo esto para corregir errores o posibles errores, hasta encontrar o ver como mejorar el programa.