



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor:* M.I. Marco Antonio Martínez Quintana

*Asignatura:* Fundamentos de Programación

*Grupo:* 3

*No de Práctica(s):* 10

*Integrante(s):* Ramirez Garcia Diego Andres

*No. de Equipo de  
cómputo empleado:* No Aplica

*No. de Lista o Brigada:* 35

*Semestre:* Primero

*Fecha de entrega:* 11/Diciembre/2020

*Observaciones:* Material de gran importancia para el desarrollo de software dentro  
de los Entornos de Desarrollo Integrado (IDE).

**CALIFICACIÓN:** \_\_\_\_\_

# 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.

## 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.

## 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.

Definiciones extraídas del Glosario IEEE610:

**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:

- a) **La depuración de un programa es útil cuando:** Debe realizarse un análisis exhaustivo del mismo en ejecución para averiguar cuál es su flujo de operación y encontrar formas de mejorarlo.
- b) **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.
- c) **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.

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.
- **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.

- **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 los 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 se establece cuando el programa ha sido terminado y totalmente probado.

## Depuración de programas escritos en C con GCC y GDB

Para depurar un programa usando las herramientas desarrolladas por GNU, éste debe compilarse con información para depuración por medio del compilador GCC.

```
gcc -g -o calculadora calculadora.c
```

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

debe usarse la herramienta GDB, la cual, es el depurador para cualquier programa ejecutable realizado por GCC.

Para depurar un ejecutable debe invocarse a GDB en la terminal indicando cuál es el programa ejecutable a depurar, por ejemplo, para depurar calculadora:

```
gdb ./calculadora
```

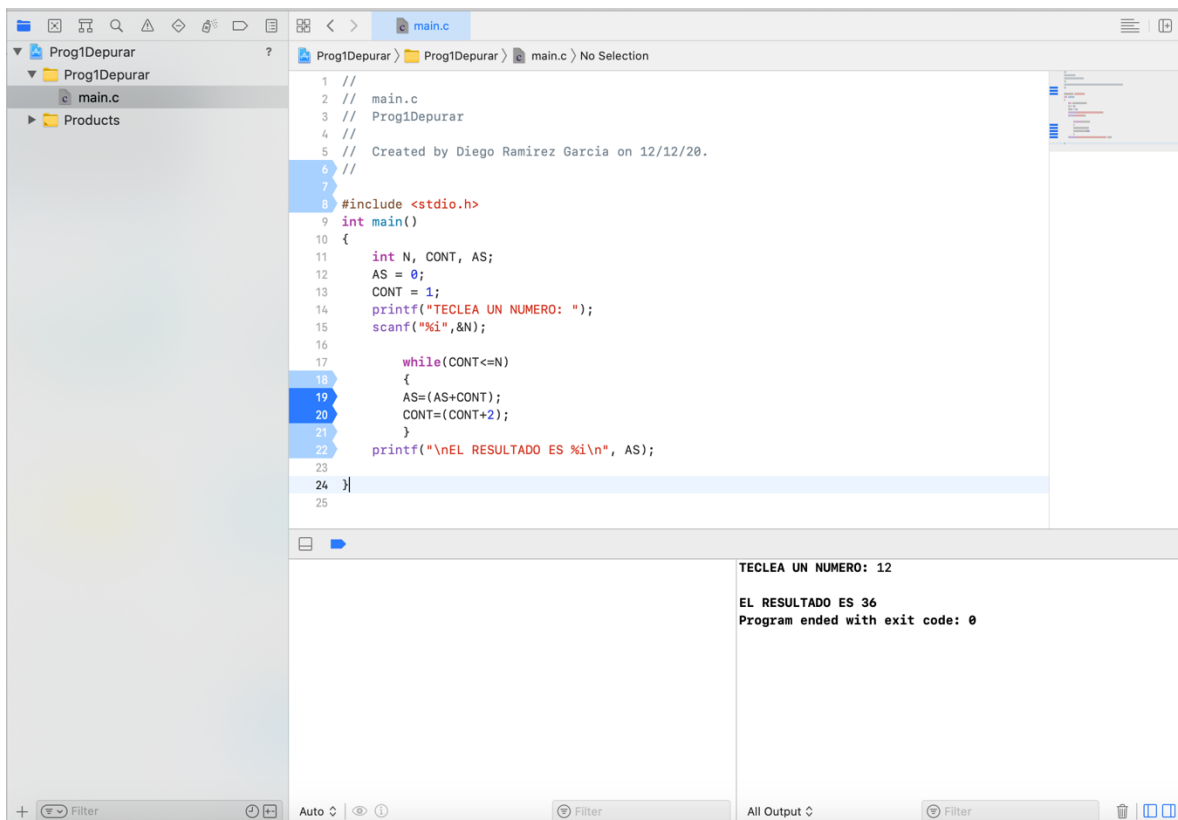
Al correr GDB se entra a una línea de comandos. De acuerdo al comando es posible realizar distintas funciones de depuración:

- ***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.
- ***b:*** Establece un punto de ruptura para lo cual debe indicarse en qué línea se desea establecer o bien también acepta el nombre de la función donde se desea realizar dicho paso. Ejemplo: *b 5*
- ***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*
- ***info 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: *ignore*
- ***q o quit:*** Termina la ejecución de GDB.

## Desarrollo de actividades

### Ejercicios propuestos

1. Para el siguiente código fuente, 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.



```
1 //
2 // main.c
3 // Prog1Depurar
4 //
5 // Created by Diego Ramirez Garcia on 12/12/20.
6 //
7 //
8 //
9 #include <stdio.h>
10 int main()
11 {
12     int N, CONT, AS;
13     AS = 0;
14     CONT = 1;
15     printf("TECLEA UN NUMERO: ");
16     scanf("%i",&N);
17
18     while(CONT<=N)
19     {
20         AS=(AS+CONT);
21         CONT=(CONT+2);
22     }
23     printf("\nEL RESULTADO ES %i\n", AS);
24 }
25
```

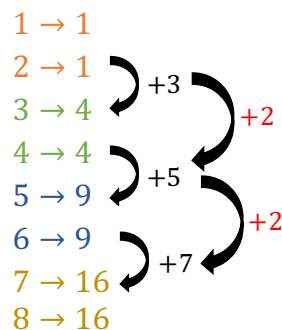
TECLEA UN NUMERO: 12

EL RESULTADO ES 36

Program ended with exit code: 0

### Funcionalidad del Programa

El presente programa tiene como funcionalidad cada dos números, es decir, después de cada múltiplo de dos se genera un incremento que al igual va de dos en dos unidades, es decir, el incremento varía en función de dos unidades con respecto al anterior incremento.



2. 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.

```
1 //
2 // main.c
3 // Prog1Depurar
4 //
5 // Created by Diego Ramirez Garcia on 12/12/20.
6 //
7
8 #include <stdio.h>
9 int main()
10 {
11     int i,j;
12
13     for(i = 1; i<=10; i++)
14     {
15         printf("\nTabla del %i\n", i);
16
17         for(j=10; j==10; j++)
18         {
19             printf("%i X %i = %i\n", i, j, i*j);
20         }
21     }
22 }
23
```

Tabla del 1  
1 X 10 = 10

Tabla del 2  
2 X 10 = 20

Tabla del 3  
3 X 10 = 30

Tabla del 4  
4 X 10 = 40

Tabla del 5  
5 X 10 = 50

Tabla del 6  
6 X 10 = 60

Tabla del 7  
7 X 10 = 70

Tabla del 8  
8 X 10 = 80

Tabla del 9  
9 X 10 = 90

Tabla del 10  
10 X 10 = 100  
(lldb)

## Funcionamiento del programa

El programa calcula la table del valor indicado en la variable “j”, en este caso, la table del numeral 10.

## Correcciones

El problema por el cual no se mostraba ninguna tabla era porque “j” estaba incivilizada en uno, así que tuve que inicializarla en 10 para que se mostraran las tablas; en cuanto al problema del porque no aparecia la tabla del diez, era porque “i” estaba limitada a 9 por el signo “< 10”, tenía dos opciones o agregar un igual “<= 10” o en su caso elegir el numeral 11 en el primer caso (“< 11”) para que de esta manera tomara al 10.

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

The screenshot shows a C program in a debugger. The code is as follows:

```
1 //
2 // main.c
3 // Prog1Depurar
4 //
5 // Created by Diego Ramirez Garcia on 12/12/20.
6 //
7
8 #include <stdio.h>
9 #include <math.h>
10 int main()
11 {
12     int K, X, AP, N;
13     float AS;
14     printf("EL TERMINO GENERICO DE LA SERIE ES: X^K/K!");
15
16     printf("\nN=");
17     scanf("%d", &N); //faltaba el "&"
18     printf("X=");
19     scanf("%d", &X); //faltaba el "&"
20     K=0;
21     AP=1;
22     AS=0;
23
24     while(K<=N)
25     {
26         AS=AS+pow(X, K)/AP;
27         K=K+1;
28         AP=AP*K; //factorial de K
29     }
30     printf("SUM=%le\n", AS);
31 }
32
```

The output window shows the following text:

```
EL TERMINO GENERICO DE LA SERIE ES: X^K/K!
N=2
X=2
SUM=5.000000e+00
Program ended with exit code: 0
```

The program ended with exit code 0, which is unexpected given the problem statement. The error message in the original image was "Program ended with exit code: 139", which is a common signal for a segmentation fault. The error was caused by missing ampersands in the scanf calls on lines 17 and 19.

### Funcionamiento del programa

Básicamente el programa nos solicita dos valores, el primero “X” será el termino a elevar, y el segundo “N” es el numeral que determina el exponente “K” y a su vez a la par el programa obtiene el factorial de K; una vez teniendo esos tres valores se efectúa la operación “X^K / K” y se suma a el valor de AS que es básicamente el valor de la función anterior con valores anteriores al solicitado.

### Correcciones

El programa carecía de ampersand (“&”) a la hora de solicitar un valor, este error se repetía en dos líneas de código, específicamente en la línea 17 y 19.



## Referencias

*Gutiérrez Rodríguez, Javier Jesús. Primeros pasos con GDB. Consulta: octubre de 2016. Disponible en: [http://www.lsi.us.es/~javierj/ssoo\\_ficheros/GuiaGDB.htm](http://www.lsi.us.es/~javierj/ssoo_ficheros/GuiaGDB.htm)*

*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: <http://learnassembler.com/opc.html>*

*Gutiérrez, Erik Marín. Depuración de programas Dev C++. Consulta: octubre de 2016. Disponible en: <http://programacionymetodos.blogspot.mx/2012/05/depuracion-de-programas-dev-c.html>*

*González Cárdenas, Miguel Eduardo; Marín Lara, Claudia Lorena; Noguerón Pérez, Pedro. Apuntes De Computadoras Y Programación. Universidad 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>*