

# Guía práctica de estudio 10: Depuración de programas

A partir de los 3 ejercicios propuestos de la práctica oficial utilizar algún entorno de depuración (gdb, Dev-C++ y Code::Blocks) para encontrar la utilidad del programa y funcionalidad de los principales comandos de depuración como puntos de ruptura, ejecución paso por paso y seguimiento de una variable.

## EJERCICIO 1

Establecer un punto de ruptura en la línea correspondiente a la declaración de variables, realizar seguimiento para las variables AS y CONT, ejecuta paso por paso para observar su comportamiento y anótalo en la tabla.

### CÓDIGO ORIGINAL

```
1
2  #include <stdio.h>
3
4  int main(int argc, char *argv[])
5  {
6      int N, CONT, AS;
7      AS=0;
8      CONT=1;
9      printf("TECLEA UN NUMERO: ");
10     scanf("%i",&N);
11     while(CONT<=N)
12     {
13         AS=(AS+CONT);
14         CONT=(CONT+2);
15     }
16     printf("\n EL RESULTADO ES %d\n", AS);
17     getchar();
18     getchar();
19     return 0;
20 }
```

### TABLA DE COMPORTAMIENTO

ITERACIÓN	AS	CONT
INICIALES	0	1
1	1	1
2	4	3
3	4	5

### EXPLICACIÓN

Mediante el debug pudimos observar cómo iban obteniendo distintos valores las variables **AS** y **CONT** a través de cómo se iba ejecutando el programa línea por línea.

## EJERCICIO 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.

Establece un punto de ruptura en la línea correspondiente a la declaración de variables, realizar seguimiento para las variables *i* y *j* y ejecuta paso por paso para observar el comportamiento de las variables anotando sus valores en cada iteración en una tabla como la del ejercicio anterior.

### CÓDIGO ORIGINAL

```
1
2  #include <stdio.h>
3
4  int main(int argc, char *argv[])
5  {
6      int i, j;
7      for(i=1; i<10; i++)
8      {
9          printf("\nTabla del %i\n", i);
10         for(j=1; j==10; j++)
11         {
12             printf("%i X %i = %i\n", i, j, i*j);
13         }
14     }
15     getchar();
16     return 0;
17 }
```

### TABLA DE COMPORTAMIENTO

ITERACIÓN	i	j
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	6	1
7	7	1
8	8	1
9	9	1
10	-	-

NOTA 2: La tabla de comportamiento demuestra que una variable siempre obtiene en alguna parte del código el mismo valor, además de que no existe la tabla del número 10

## CÓDIGO MODIFICADO

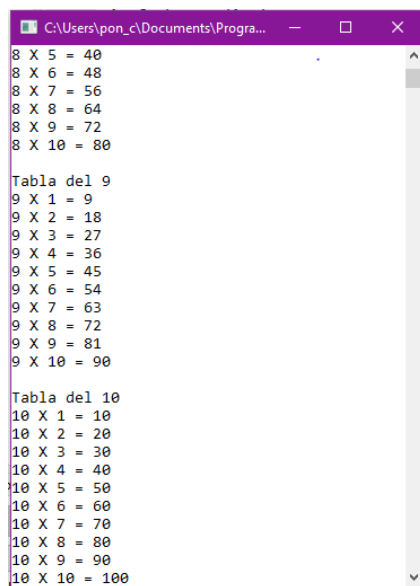
```
1  #include <stdio.h>
2
3  int main(int argc, char *argv[] )
4  {
5      int i, j;
6
7      for(i=1; i<11; i++)
8      {
9          printf("\nTabla del %i\n", i);
10         for(j=1; j<11; j++)
11             //Cambiamos el == por un < ya que es el condicional indicado para lo que se pide
12             {
13                 printf("%i X %i = %i\n", i, j, i*j);
14             }
15     }
16     getchar();
17     getchar();
18     return 0;
19 }
```

## EXPLICACIÓN

El problema de la impresión de las tablas de multiplicar se encontraba en la parte del segundo **ciclos for**, al tener como “j == 10” nunca podíamos entrar al segundo ciclo for y además podemos comprobar este error mediante la tabla de iteraciones, por ello es que solamente imprimíamos el primer printf (El que solo menciona tabla de multiplicar).

Por otro lado, incrementé los valores de los ciclos for tanto para la variable **i** como para la variable **j** ya que en la primera corrección solamente corregía el problema asignado con la impresión con las tablas, sin embargo, solamente se imprimía hasta la tabla 9 y el ejercicio pedía hasta la tabla 10, por ello los ciclos for los tuve que pasar de 10 a 11.

## PANTALLA DE EJECUCIÓN



```
C:\Users\pon_c\Documents\Progra...
8 X 5 = 40
8 X 6 = 48
8 X 7 = 56
8 X 8 = 64
8 X 9 = 72
8 X 10 = 80

Tabla del 9
9 X 1 = 9
9 X 2 = 18
9 X 3 = 27
9 X 4 = 36
9 X 5 = 45
9 X 6 = 54
9 X 7 = 63
9 X 8 = 72
9 X 9 = 81
9 X 10 = 90

Tabla del 10
10 X 1 = 10
10 X 2 = 20
10 X 3 = 30
10 X 4 = 40
10 X 5 = 50
10 X 6 = 60
10 X 7 = 70
10 X 8 = 80
10 X 9 = 90
10 X 10 = 100
```

Parte final del código que demuestra hasta la tabla 10

### EJERCICIO 3

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

#### CÓDIGO ORIGINAL

```
1
2  #include <stdio.h>
3  #include <math.h>
4
5  int main(int argc, char *argv[])
6  {
7      int K, X, AP, N;
8      float AS;
9      printf("EL TERMINO GENERICO DE LA SERIE ES: X^K/K!");
10     printf("\nN=");
11     scanf("%d", N);
12     printf("X=");
13     scanf("%d", X);
14     K=0;
15     AP=1;
16     AS=0;
17     while(K<=N)
18     {
19         AS=AS+pow(X,K)/AP;
20         K=K+1;
21         AP=AP*K;
22     }
23     printf("SUM=%le", AS);
24     getchar();
25     getchar();
26     return 0;
27 }
```

NOTA: Mediante el depurador lo primero que se aprecia es que al momento de ingresar el primer dato nos manda un mensaje de interrupción

#### CÓDIGO MODIFICADO

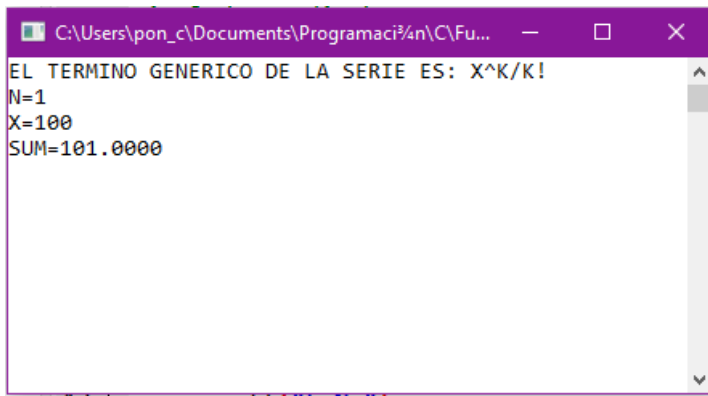
```
1  #include <stdio.h>
2  #include <math.h>
3
4  int main(int argc, char *argv[] )
5  {
6      int K, X, AP, N;
7      float AS;
8      //Prefiero tener los valores de las variables hasta la parte de arriba del código
9      K=0;
10     AP=1;
11     AS=0;
12     printf("EL TERMINO GENERICO DE LA SERIE ES: X^K/K!");
13     printf("\nN=");
14     scanf("%d",&N); // Faltaba el & en la lectura de la variable
15     printf("X=");
16     scanf("%d",&X); //Faltaba el & en la lectura de la variable
17
18     while(K<=N)
19     {
20         AS=AS+pow(X,K)/AP;
21         K=K+1; AP=AP*K;
22     }
23     printf("SUM=%.4f", AS);
24
25     getchar();
26     getchar();
27     return 0;
28 }
```

## EXPLICACIÓN

El principal error en el código inicial era que al momento de guardar los valores de las variables éstas no se guardaban realmente puesto el código original no incluía los "&" antes de las variables por ello es que se interrumpía el programa.

Por último, en el mensaje final decidí cambiar la manera en que se escribiera la variable flotante debido a que la manera original estaba expresada en manera exponencial, sin embargo, en gran cantidad de casos los valores que poseen son reales solamente tienen entre 0 a 4 decimales los cuales pueden ser más amistosos para el usuario en su forma decimal. (Esto es realmente opcional ya que es a gusto del usuario o público al que va dirigido).

## PANTALLA DE EJECUCIÓN



A screenshot of a Windows command prompt window. The title bar shows the file path "C:\Users\pon\_c\Documents\Programaci3n\C\Fu...". The window contains the following text:

```
EL TERMINO GENERICO DE LA SERIE ES: X^K/K!  
N=1  
X=100  
SUM=101.0000
```

## CONCLUSIONES

Alfonso Murrieta Villegas

En la presente práctica mediante el uso de un IDE (Dev C++) aprendí las ventajas y usos del proceso conocido como depuración o en inglés "debug".

La depuración es un proceso de ejecución donde un programa se somete a un determinado ambiente de ejecución donde podemos comprobar línea tras línea de código como este se va ejecutando.

La depuración como se vio en esta práctica tiene como principal ventaja el comprobar rigurosa y minuciosamente nuestros programas.