	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): Práctica #10

Integrante(s): Hernández González Braulio

No. de Equipo de cómputo empleado: No Aplica

No. de Lista o Brigada: 24

Semestre: 2021-1

Fecha de entrega: 14 / 12 / 20

Observaciones:

CALIFICACIÓN: _____

PRÁCTICA 10 – Depuración de programas.

Objetivos:

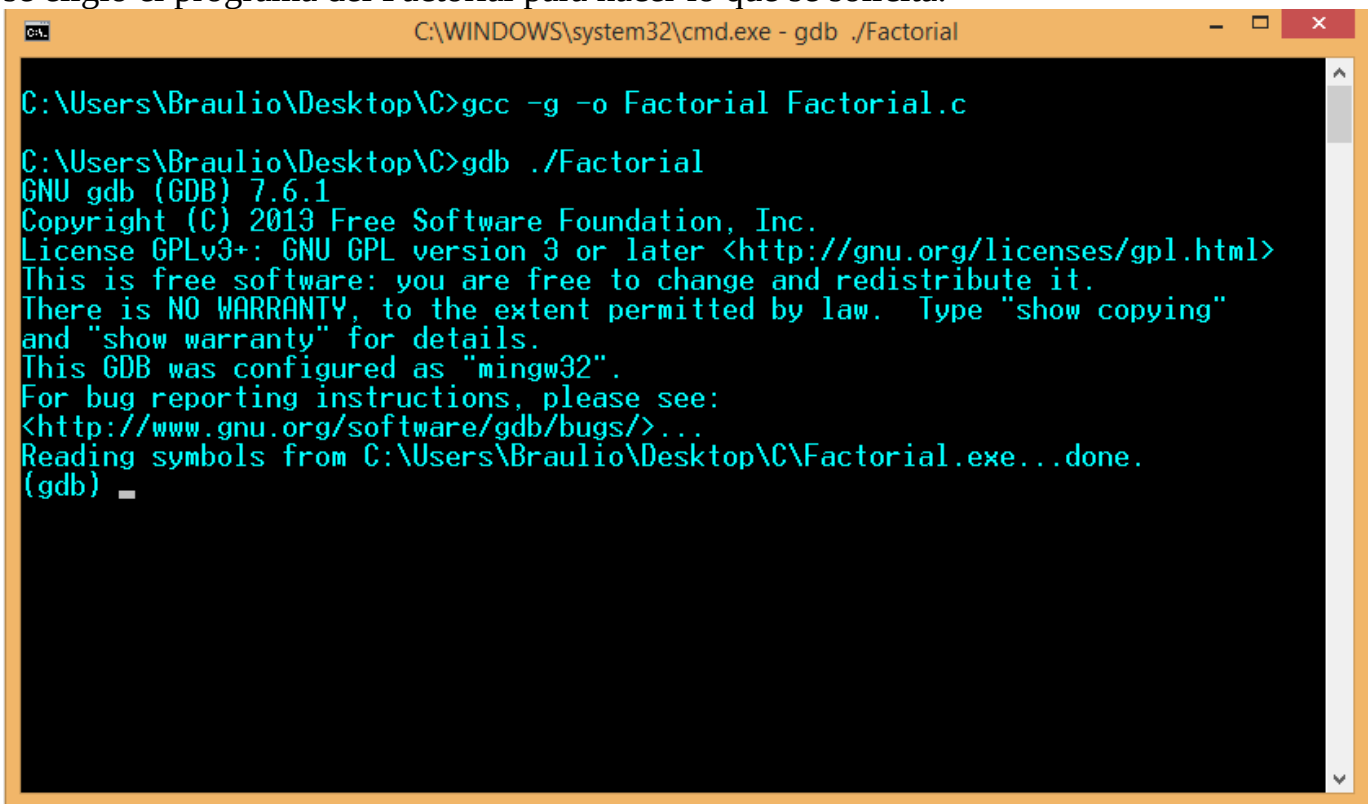
Aprender las técnicas básicas de depuración de programas en C para revisar de manera precisa el flujo de ejecución de un programa y el valor de las variables; en su caso, corregir posibles errores.

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.

Desarrollo:

Usando las estructuras e instrucciones de un depurador en C mediante el comando gcc y gdb, se eligió el programa del Factorial para hacer lo que se solicita.



Comando l o list:

```
(gdb) l
1       #include<stdio.h>
2       int main()
3       {
4               //Variables a usar
5               char au=163;
6               signed long n,res=1;
7
8               //Inicio de la calculadora
9               printf("\n\t\tCalculadora de n%cmeros factoriales\n\n",au);
10              printf("Ingresa el n%cmero inicial: ",au);
(gdb) list
11              scanf("%li",&n);
12
13              //Calcular el factorial
14              for (signed long i=1;i<=n;i++)
15              {
16                      res=res*i;
17              }
18              printf("El factorial del n%cmero %li es: %li",au,n,res);
19              return 0;
20      }(gdb)
```

Comando b:

```
20      }(gdb) b 10
Breakpoint 1 at 0x401440: file Factorial.c, line 10.
(gdb)
```

Comando d o delete:

```
20      }(gdb) b 10
Breakpoint 1 at 0x401440: file Factorial.c, line 10.
(gdb) d 10
No breakpoint number 10.
(gdb)
```

Comando clear:

```
(gdb) b 10
Breakpoint 1 at 0x401440: file Factorial.c, line 10.
(gdb) b 5
Breakpoint 2 at 0x40141e: file Factorial.c, line 5.
(gdb) clear 5
Deleted breakpoint 2
(gdb) clear 10
Deleted breakpoint 1
(gdb)
```

Comando info line:

```
(gdb) l
1       #include<stdio.h>
2       int main()
3       {
4               //Variables a usar
5               char au=163;
6               signed long n,res=1;
7
8               //Inicio de la calculadora
9               printf("\n\t\tCalculadora de n%cmeros factoriales\n\n",au);
10              printf("Ingresa el n%cmero inicial: ",au);
(gdb) info line 5
Line 5 of "Factorial.c" starts at address 0x40141e <main+14>
   and ends at 0x401423 <main+19>.
(gdb) info line 1
Line 1 of "Factorial.c" is at address 0x401410 <main> but contains no code.
(gdb) info line 2
Line 2 of "Factorial.c" is at address 0x401410 <main> but contains no code.
(gdb) info line 6
Line 6 of "Factorial.c" starts at address 0x401423 <main+19>
   and ends at 0x40142b <main+27>.
(gdb)
```

Comando run o r:

```
(gdb) b 11
Breakpoint 2 at 0x401455: file Factorial.c, line 11.
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) y

Starting program: C:\Users\Braulio\Desktop\C/./Factorial.exe
[New Thread 5628.0x1930]

                Calculadora de números factoriales

Ingresa el número inicial:
Breakpoint 2, main () at Factorial.c:11
11              scanf("%li",&n);
(gdb)
```

Comando c:

```
(gdb) b 9
Breakpoint 2 at 0x40142b: file Factorial.c, line 9.
(gdb) run
Starting program: C:\Users\Braulio\Desktop\C/./Factorial.exe
[New Thread 7952.0x4b8]

Breakpoint 2, main () at Factorial.c:9
9               printf("\n\t\tCalculadora de n%cmeros factoriales\n\n",au);
(gdb) c
Continuing.

                Calculadora de números factoriales

Ingresa el número inicial: 5
El factorial del número 5 es: 120[Inferior 1 (process 7952) exited normally]
(gdb)
```

Comando s:

```
Breakpoint 2, main () at Factorial.c:9
9               printf("\n\t\tCalculadora de n%cmeros factoriales\n\n",au);
(gdb) s

                Calculadora de números factoriales

10              printf("Ingresa el n%cmero inicial: ",au);
(gdb)
```

Comando n:

```
Breakpoint 2, main () at Factorial.c:9
9               printf("\n\t\tCalculadora de n%cmeros factoriales\n\n",au);
(gdb) n

                Calculadora de números factoriales

10              printf("Ingresa el n%cmero inicial: ",au);
(gdb)
```

Comando p o print:

```
(gdb) p res
$3 = 1
(gdb)
```

Comando ignore:

```
(gdb) b 10
Breakpoint 2 at 0x401440: file Factorial.c, line 10.
(gdb) ignore 10
Second argument (specified ignore-count) is missing.
(gdb)
```

Comando q o quit:

```
(gdb) q
A debugging session is active.

        Inferior 1 [process 6244] will be killed.

Quit anyway? (y or n) y

C:\Users\Braulio\Desktop\C>
```

Ejercicios propuestos:

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   #include<stdio.h>
2   void main()
3   {
4       char au=163;
5       int N, CONT, AS;
6       AS=0;
7       CONT=1;
8
9       printf("Teclea un n%cmero: \n",au);
10      scanf("%i",&N);
11      while(CONT<=N)
12      {
13          AS=(AS+CONT);
14          CONT=(CONT+2);
15      }
16      printf("\nEl resultado es: %i",AS);
17  }
```

FUNCIONALIDAD DEL PROGRAMA:

Este programa lo que hace es una suma de números impares, mas el valor fijo de la variable CONT por cada dos valores consecutivos, donde uno es impar y otro par, es decir, forma a parejas de par e impar con el mismo resultado dado por AS.

La variable CONT siempre va a ser un valor fijo de 3, pues inicia en 1 y se le suman 2, sin que exista alguna alteración, dando como valor fijo siempre 3.

Ahora bien, la variable AS depende del valor N, ya que, si N es mayor o igual que el valor de 1, se ejecuta el ciclo while, y puede hacer una suma, de lo contrario siempre dará 0; por esa razón N no puede ser un valor menor a 0.

AS junta a los dos primeros números pares e impares que son el 1 y 2 e inicia el proceso donde les resta en su base de inicio 0, -2 para que pueda iniciar en 1, y le suma el valor fijo de CONT para obtener el mismo resultado en esos dos números.

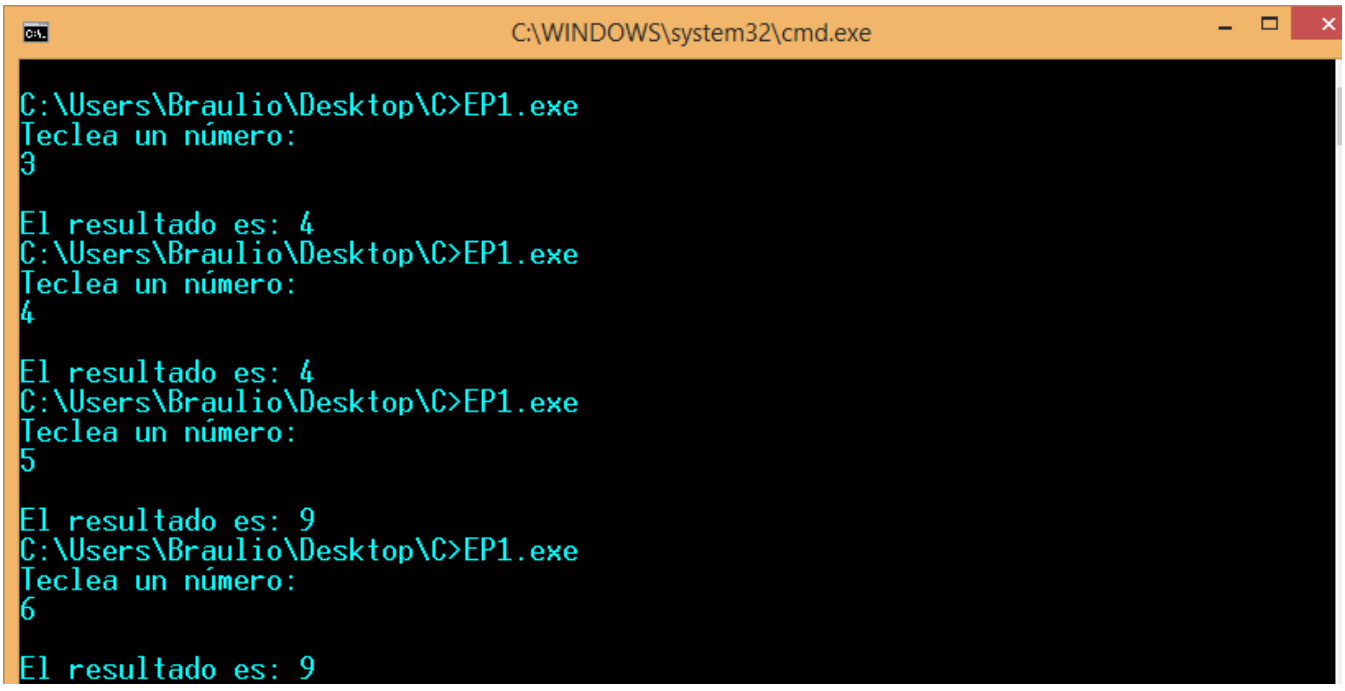
Posteriormente, AS vuelve a juntar a la siguiente pareja de números, un impar y par que vendría siendo el 3 y 4, vuelve a hacer la suma de su base fija (0), mas el valor de CONT (3) mas el primer número impar diferente a 1, es decir 3, y aplicando el mismo resultado a 3 y 4.

Posteriormente, AS vuelve a juntar a la siguiente pareja de impar y par, que vendría siendo el 5 y 6, vuelve a hacer la suma de su base fija (0), mas el valor de CONT (3) t el valor final del primer impar anterior, más el segundo número par, es decir, 5, y aplicar el mismo resultado a 5 y 6.
Así sucesivamente.

Observación:
No use el depurador, puesto que al ejecutar el código y observar los resultados me pude dar cuenta de una serie de patrones que le dan forma al programa.

FORMA SIMPLIFICADA:

N= 1 ; cont=3 ; as=0+3-**2** = 1 -2 base
N=2 ; cont=3 ; as=0+3-**2** = 1
N=3 ; cont=3 ; as=0+3+**1** = 4 **-2**+3 = 1
N=4 ; cont=3 ; as=0+3+**1**=4
N=5 , cont=3 ; as=0+3+**6**=9 **1**+5=6
N=6 ; cont=3 ; as=0+3+**6**=9
N=7 ; cont=3 ; as=0+3+**13**=16 **6**+7=13
N=8 ; cont=3 ; as=0+3+**13**=16
N=9 ; cont=3 ; as=0+3+**22**=25 **13**+9=22
N=10 ; cont=3 ; as=0+3+**22**=25
N=11 ; cont=3 ; as=0+3+**33**=36 **22**+11=33
N=12 ; cont=3 ; as=0+3+**33**=36



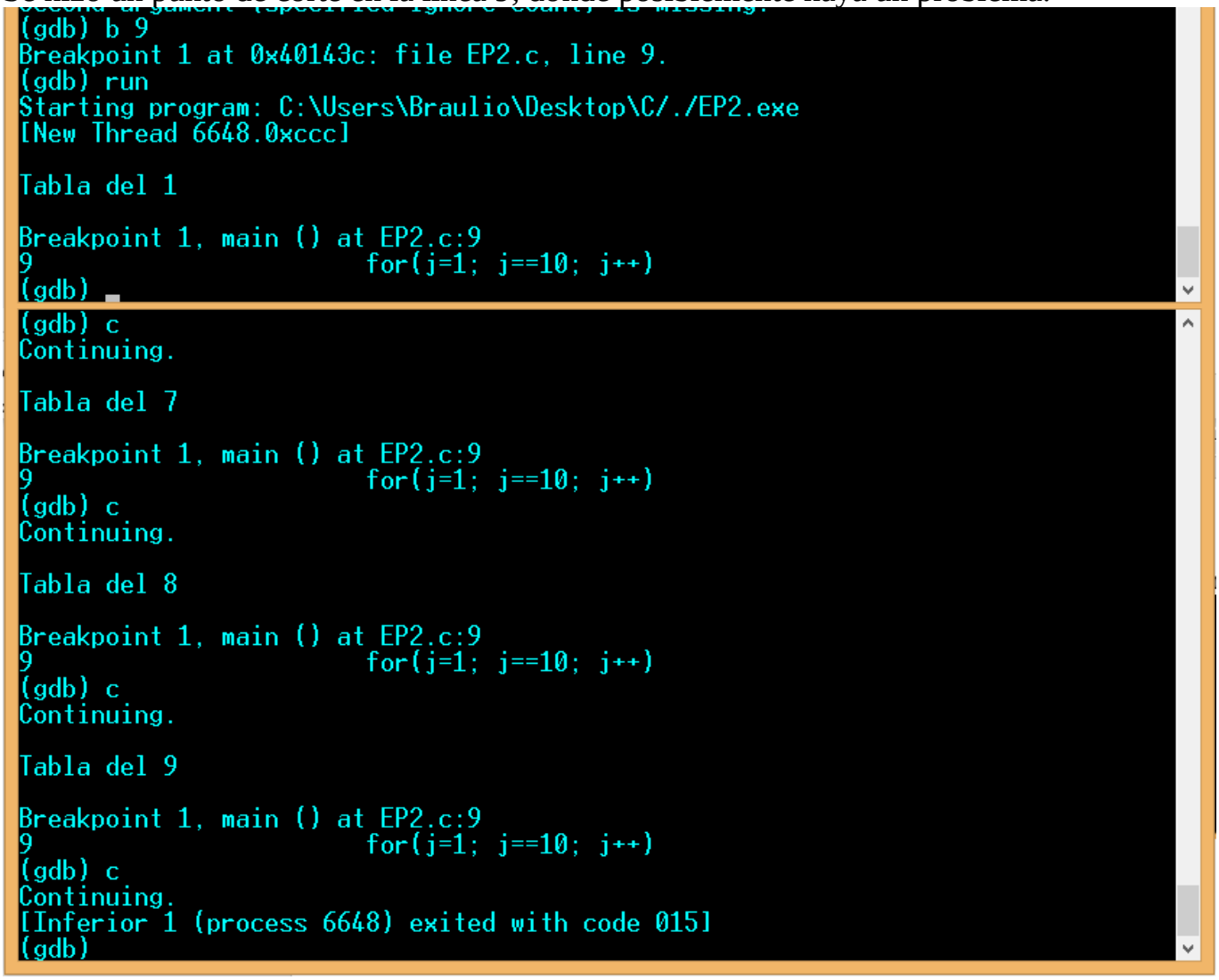
El siguiente programa debe mostrar las tablas de multiplicar desde la 1 hasta la 10. En un principio no se mostraba la tabla del 10, después de intentar corregirse sin un depurador dejaron de mostrarse el resto de las tablas. Usar un depurador de C para averiguar el funcionamiento del programa y corregir ambos problemas.

```c
#include<stdio.h>
void main()
{
    int i,j;

    for(i=1; i<10; i++)
    {
        printf("\nTabla del %i\n",i);
        for(j=1; j==10; j++)
        {
            printf("%i x %i = %i\n",i,j,i*j);
        }
    }
}
```

Al ejecutar el código ya dado:

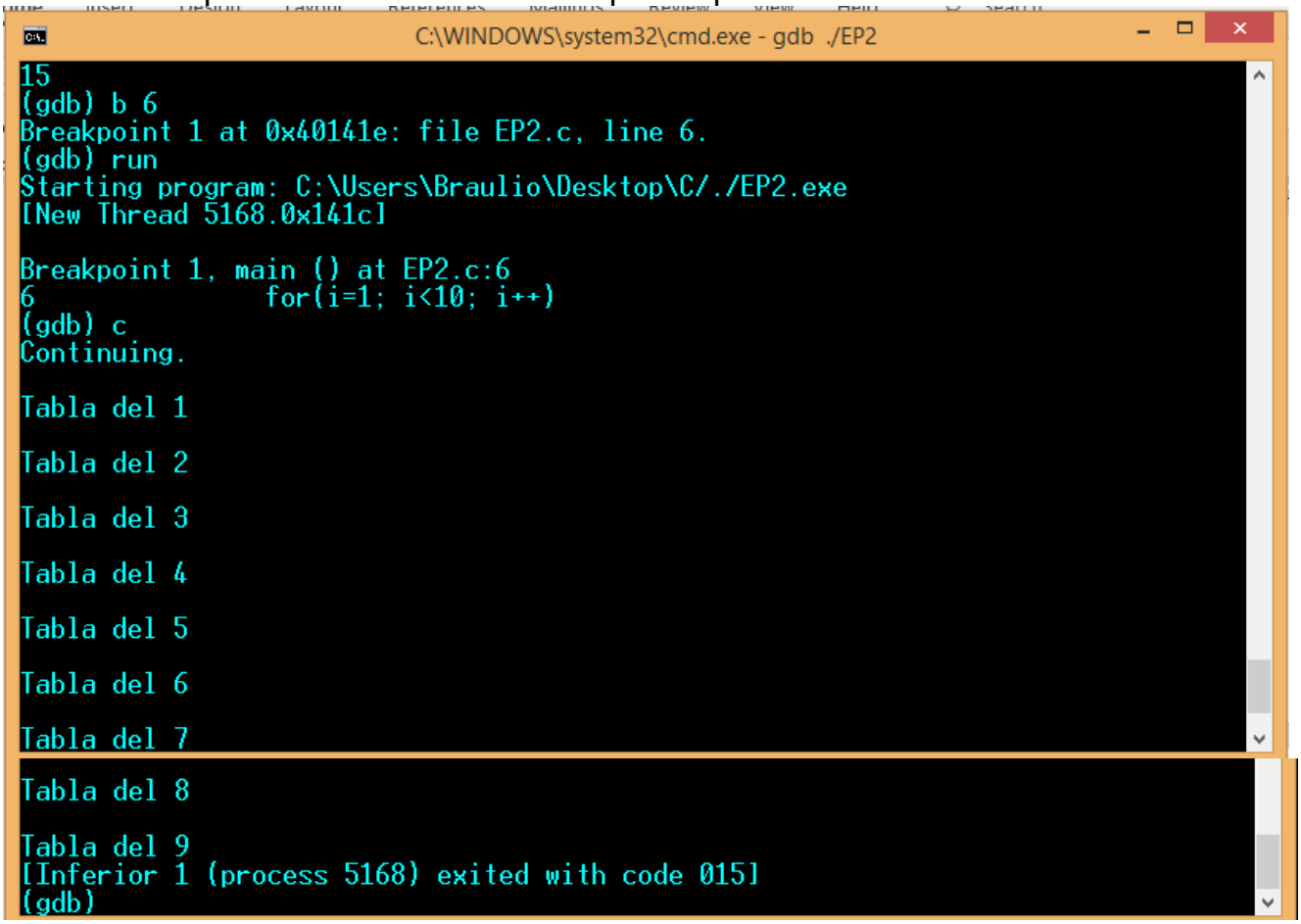
```
C:\Users\Braulio\Desktop\C>EP2.exe

Tabla del 1

Tabla del 2

Tabla del 3

Tabla del 4

Tabla del 5

Tabla del 6

Tabla del 7

Tabla del 8

Tabla del 9

C:\Users\Braulio\Desktop\C>
```

Se hizo un punto de corte en la linea 9, donde posiblemente haya un problema:



Posible error al definir la variable j en el segundo for, puesto que no enseña las operaciones necesarias que estamos solicitando, que es enseñar detalladamente cada operación de cada tabla hasta el valor de i por 10.

Se realizó un punto de corte en la línea 6 por un posible error:



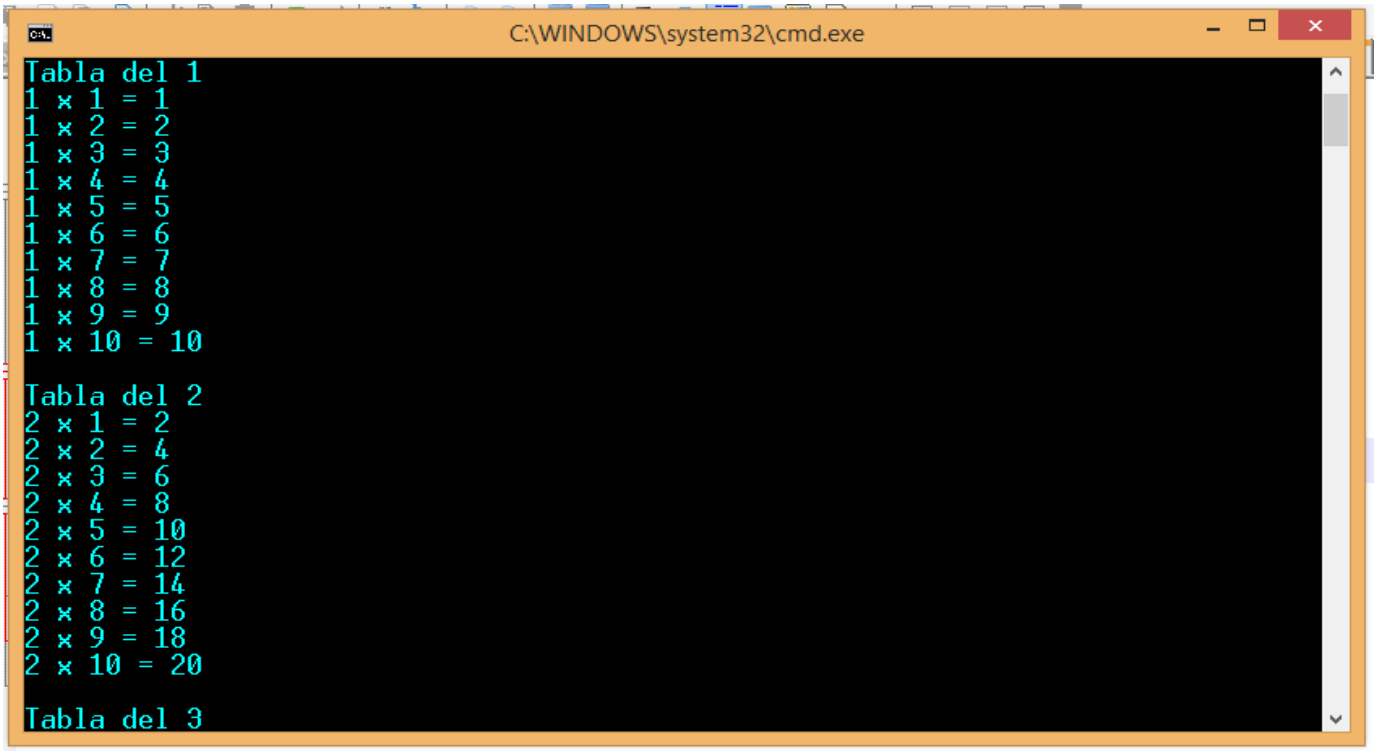
Posible error al definir a la variable i, puesto que llega nada mas hasta la tabla del 9 cuando estamos solicitando la del 10.

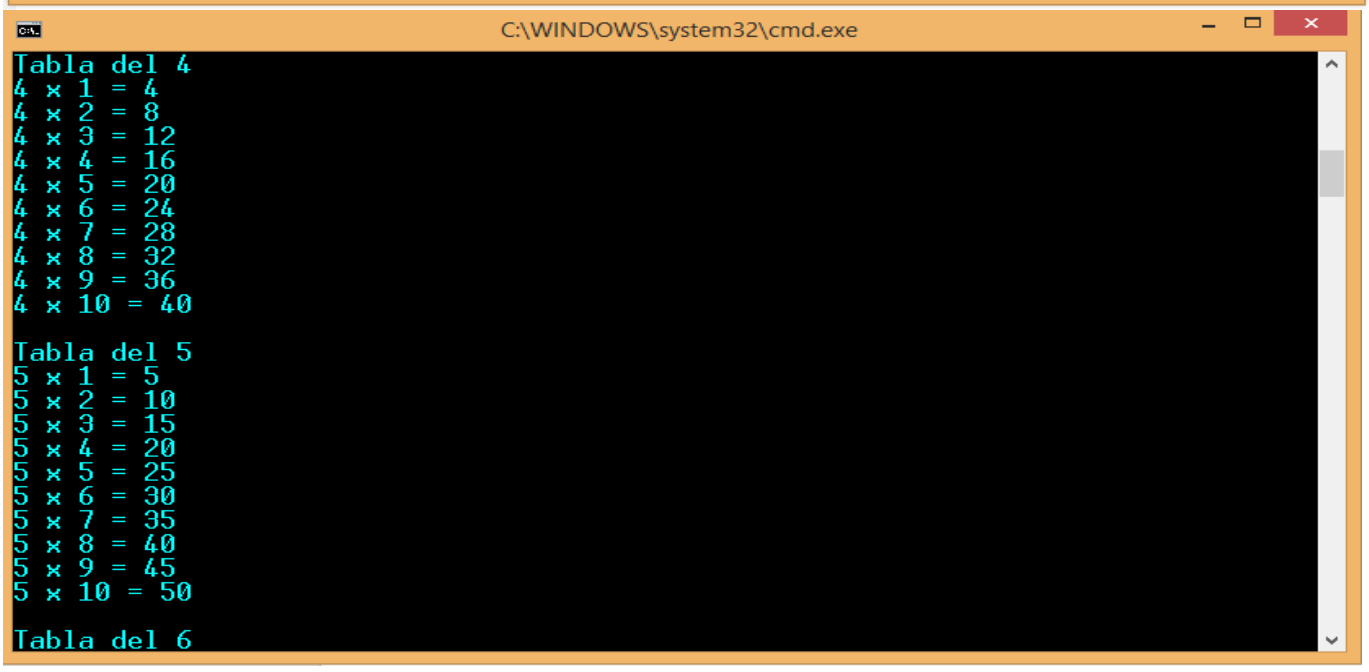
Después de hacer uso del depurador y ver como corre el programa, se puede observar dos errores notorios en esas líneas que tienen el for en su mayoría y se trata de definir a la variable:

```
1  #include<stdio.h>
2  void main()
3  {
4      int i,j;
5
6      for(i=1; i<10; i++)
7      {
8          printf("\nTabla del %i\n",i);
9          for(j=1; j==10; j++)
10         {
11             printf("%i x %i = %i\n",i,j,i*j);
12         }
13     }
14 }
```

Primer error: i está mal definida en for, ya que está "diciendo" que solo van a tomar valores menores a 10, produciendo que solamente se llegue hasta la tabla del 9.
Solución: i < 11.

Segundo error: j está mal definida en for, ya que está definida dos veces, una cuando va a valer 1 y otra cuando tiene que valer 10, dando a entender que o agarra el valor a 1 o agarra el valor cuando tiene que valer 10.
Además, no está definida para la tabla del 10, ya que hasta ahí llegaría.
Solución: cambiar el signo de igual a, por menor a y definir 11 en vez de 10.





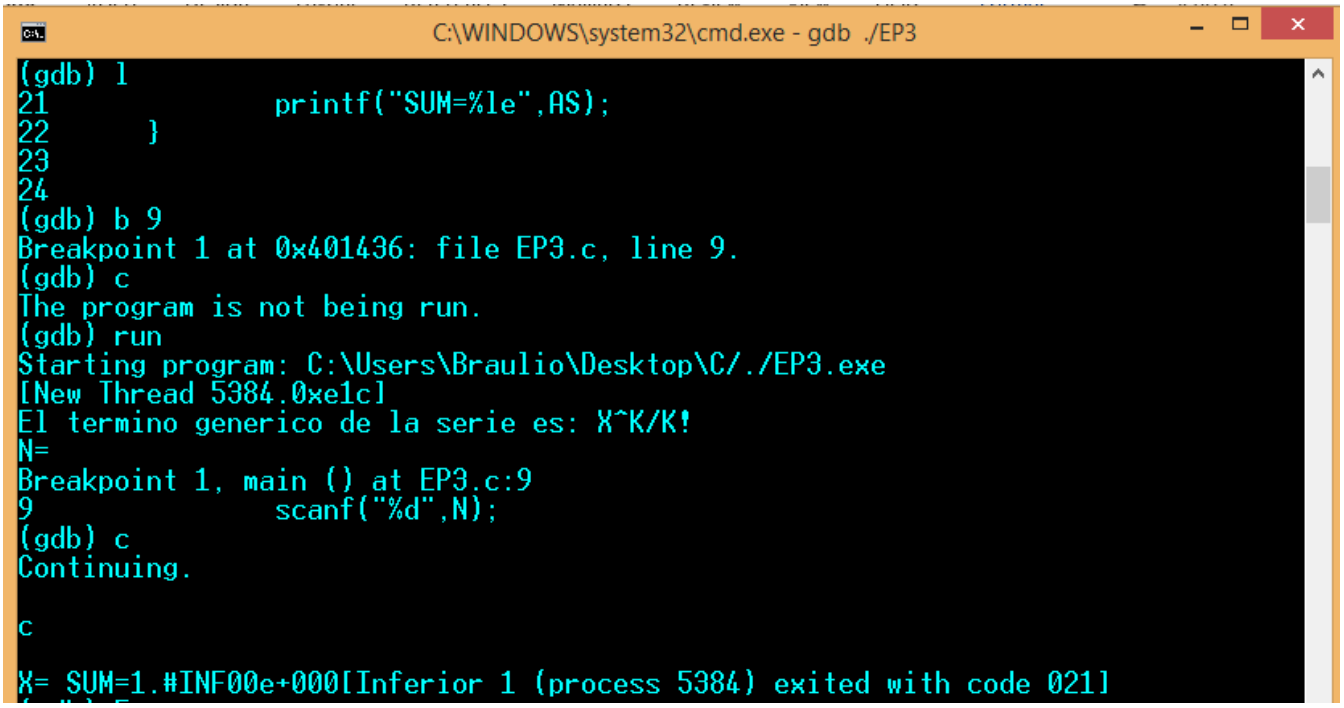
El siguiente programa muestra una violación de segmento durante su ejecución y se detiene; usar un depurador para detectar y corregir la falta.

```c
#include<stdio.h>
#include<math.h>
void main()
{
    int K, X, AP, N;
    float AS;
    printf("El termino generico de la serie es: X^K/K!");
    printf("\nN= ");
    scanf("%d",N);
    printf("\nX= ");
    scanf("%d",X);
    K=0;
    AP=1;
    AS=0;
    while(K<=N)
    {
        AS=AS+pow(X,K)/AP;
        K=K+1;
        AP=AP*K;
    }
    printf("SUM=%le",AS);
}
```

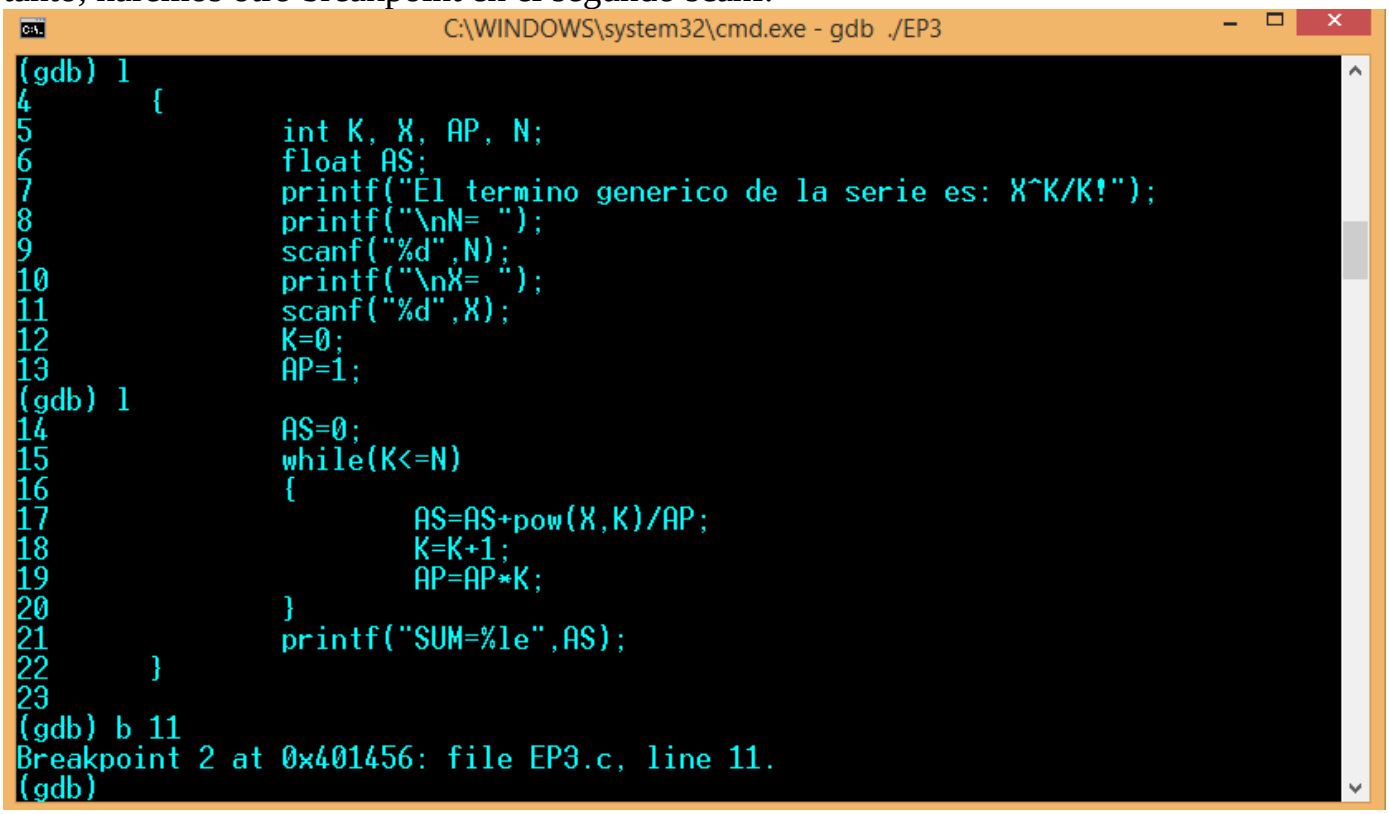
El problema al ejecutar el programa marca un error al momento de ingresar el valor de N, aunque si escribe la leyenda “\Nn= “; quiere decir, que hay un problema en el primer scanf del valor de N.

Hacemos un breakpoint en la línea 9 y ejecutamos para ver que sucede.

```
22      }
23
24
(gdb) b 9
Breakpoint 1 at 0x401436: file EP3.c, line 9.
(gdb)
```



Al seguir corriendo el programa una vez ignorando el valor de N, podemos observar que pide el valor de X y posteriormente el programa se detiene y lanza un valor sin sentido matemático alguno.
Quiere decir que hay también un problema al momento de solicitar el valor de X, por lo tanto, haremos otro breakpoint en el segundo scanf.



```
(gdb) run
Starting program: C:\Users\Braulio\Desktop\C/./EP3.exe
[New Thread 2804.0x1650]
El termino generico de la serie es: X^K/K!
N=
Breakpoint 1, main () at EP3.c:9
9               scanf("%d",N);
(gdb) c
Continuing.

c

X=
Breakpoint 2, main () at EP3.c:11
11              scanf("%d",X);
(gdb) c
Continuing.
SUM=1.#INF00e+000[Inferior 1 (process 2804) exited with code 021]
```

El programa al no tener el valor de X ni de N no puede seguir haciendo el proceso correcto. Quiere decir que mediante el depurador pudimos observar que tenemos un error en la linea 9 y 11 del programa que tenemos que corregir.

Solución al problema:

La instrucción scanf tiene como sintaxis: $scanf("\%variable", \&ValorVariable);$

La instrucción scanf en este programa está escrito como:

$scanf("\%i", N);$ y $scanf("\%i", X);$ donde olvidaron el ampersand.

Corregido el código en esa sección:

```
 7      printf("El termino generico de la serie es: X^K/K!");
 8      printf("\nN= ");
 9      scanf("%d",&N);
10      printf("\nX= ");
11      scanf("%d",&X);
```

Corriendo el programa modificado:

```
C:\Users\Braulio\Desktop\C>EP3.exe
El termino generico de la serie es: X^K/K!
N= 1

X= 5
SUM=6.000000e+000
C:\Users\Braulio\Desktop\C>EP3.exe
El termino generico de la serie es: X^K/K!
N= 0

X= 9
SUM=1.000000e+000
C:\Users\Braulio\Desktop\C>EP3.exe
El termino generico de la serie es: X^K/K!
N= 6

X= 20
SUM=1.237766e+005
C:\Users\Braulio\Desktop\C>
```

CONCLUSIÓN

En lo personal, al momento de estar leyendo y ejecutando el código no requiero necesariamente de un depurador, puesto que ejecutando el código o leyéndolo me doy cuenta cuando una operación, una sentencia, una variable o un valor está mal definido y/o mal escrito, que puede llegar a ser el causante de un problema o de un difícil entendimiento del programa.
Un depurador es una herramienta útil cuando estás adentrándote al mundo de la programación por si algún programa no te funciona a la primera y no encuentras el error, usando las sentencias de breakpoint, ignore, run y continue para ir evaluando que sucede con el programa e ir buscando fallas.
Pero una vez que ya estás muy adentrado ya no es muy necesario a menos que sea para el entendimiento de un programa complejo, con muchas variables o mucho código.

REFERENCIAS:

(s.f). Salas A y B. [Manual de prácticas]. Recuperado de: http://lcp02.fi-b.unam.mx/