	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:	03
No de Práctica(s):	10
Integrante(s):	Teran García Rodolfo Mario
No. de Equipo de cómputo empleado:	No aplica
No. de Lista o Brigada:	
Semestre:	2021-1
Fecha de entrega:	07/12/2020
Observaciones:	M.I. Marco Antonio Martínez Quintana

Calificación:_____

Práctica #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.

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. Es importante poder compilar el programa sin errores antes de depurarlo.

Desarrollo:

En la práctica se expresa claramente como ocurre el proceso de depurar algún programa, así como las diferentes situaciones que se pueden presentar para el mismo, añadido a ello se presentan una serie de comandos los cuales son empleados para la depuración de l programa, las funcionalidades que tiene el depurar un programa y los requerimientos, así como las posibles rutas para la depuración de un programa.

Dentro de las instrucciones empleadas para la depuración de un archivo yo utilizo las presentadas par aun archivo que emplea el compilador gcc desde la terminal.

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.

Para compilar, por ejemplo, un programa llamado nombre.c con GCC con información de depuración, debe realizarse en una terminal con el siguiente comando:

gcc -g -o nombre nombre.c

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

Una vez hecho el paso anterior, 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 nombre:

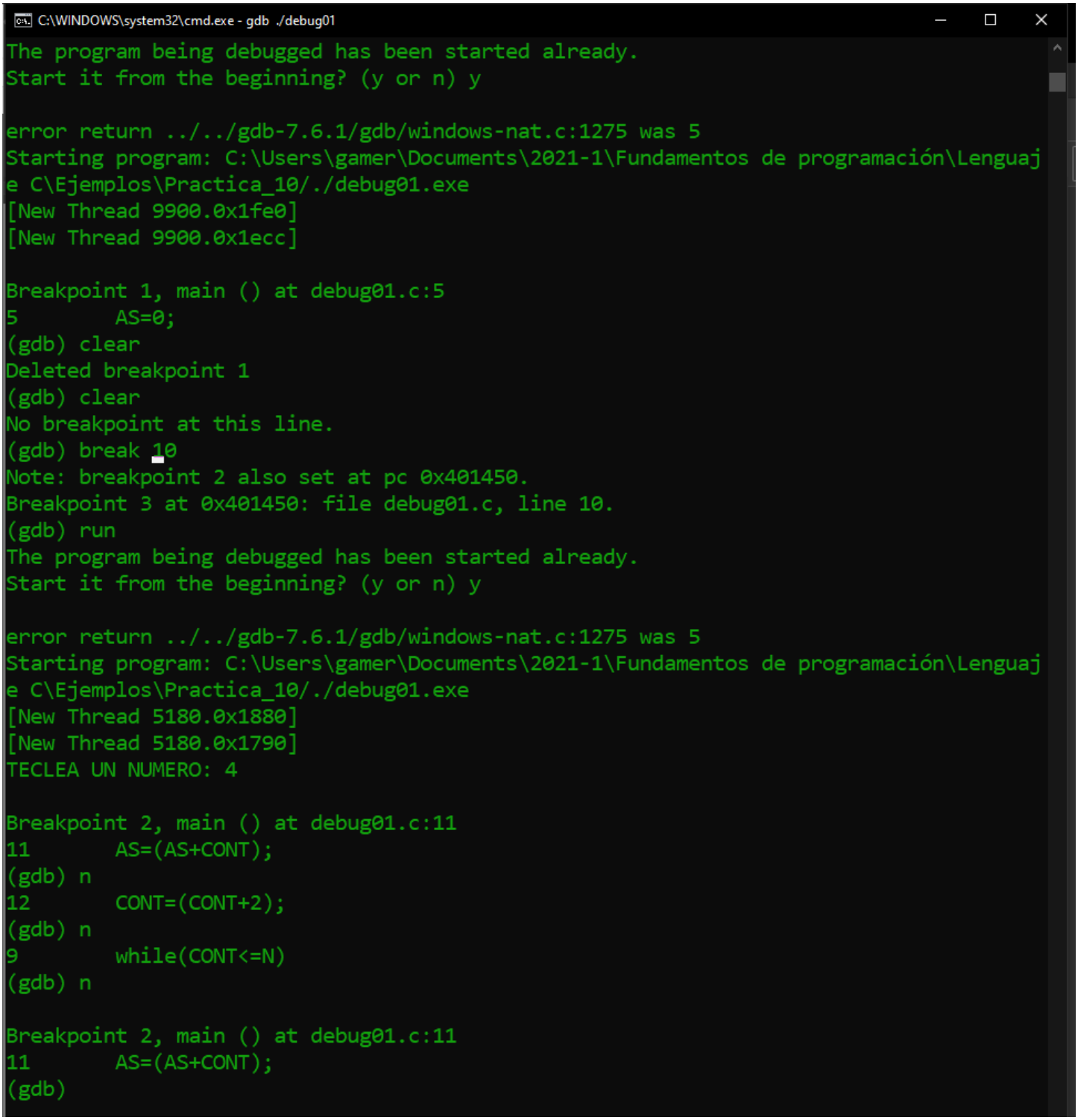
gdb ./nombre

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

En las actividades nos propone realizar la depuración de dos programas presentados en la práctica siguiendo los pasos o las líneas de comandos que puedan ayudarnos a ello, de modo que no solo puedan cumplir con su funcionamiento, sino que también lo hagan de la manera más optima posible sin la necesidad de sacrificar una gran cantidad de recursos en su ejecución.

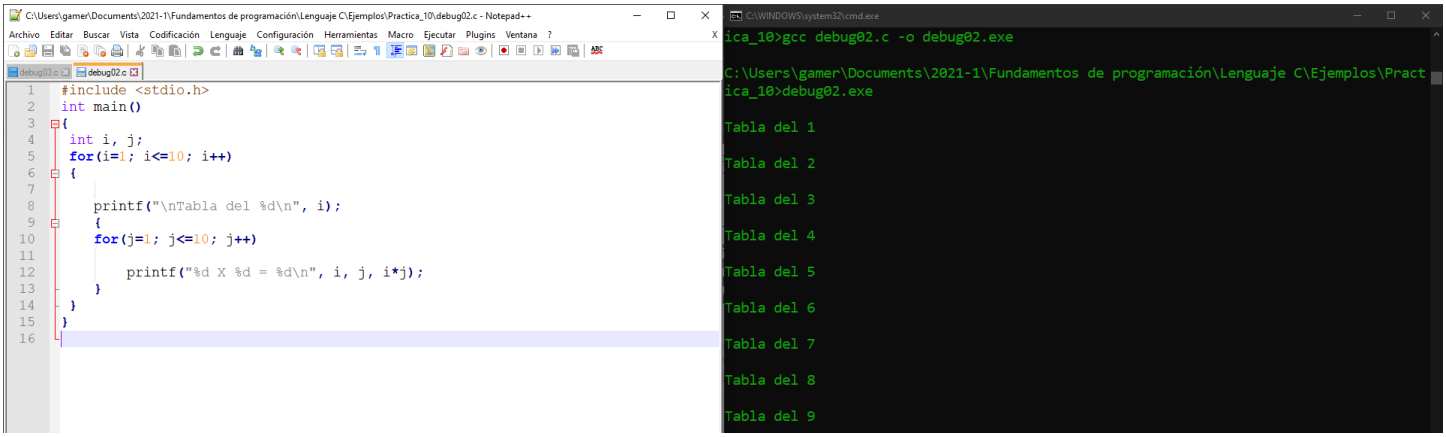
Primer programa:

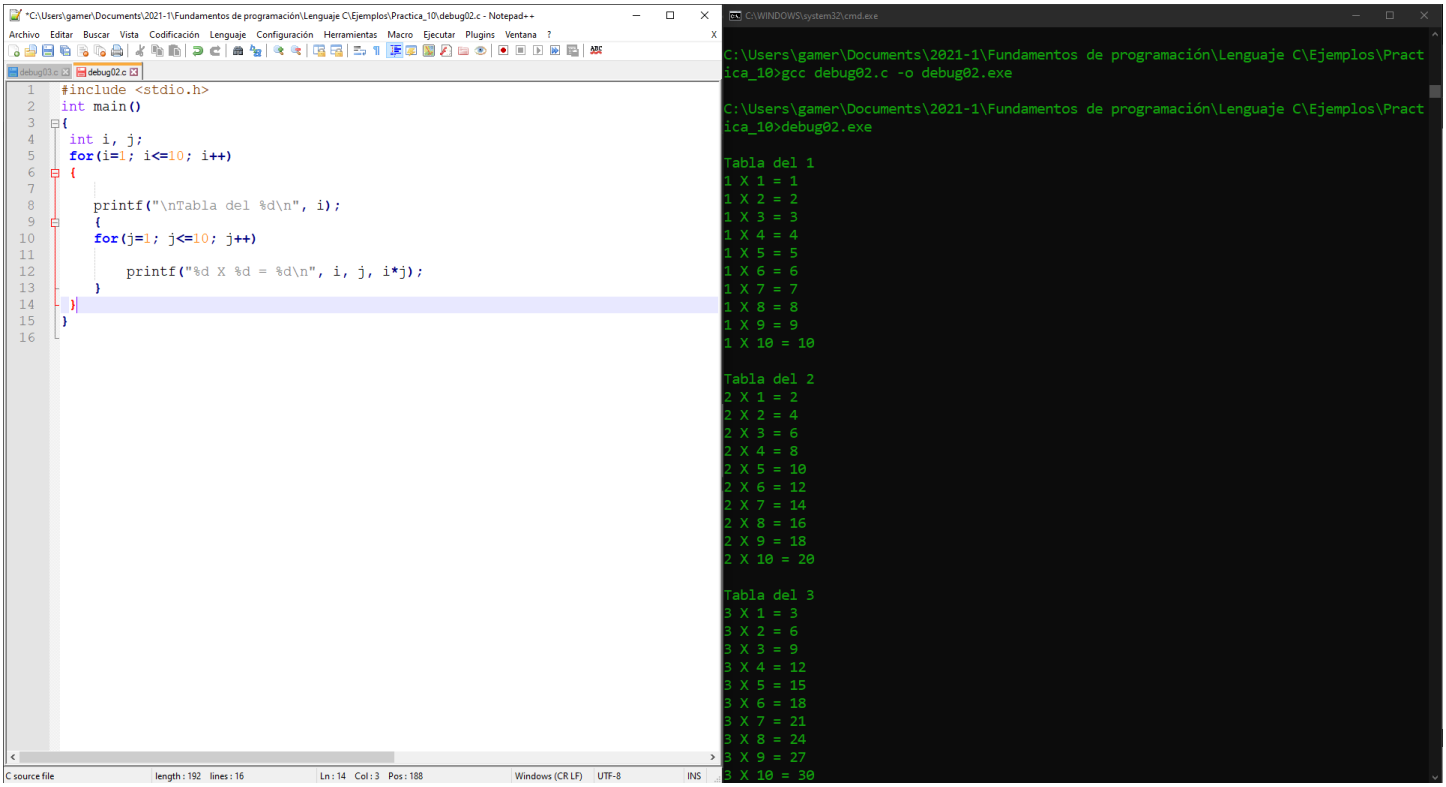
Para este programa podemos notar que al inicio en la compilación no hay ningún problema, parece compilarse en un ejecutable de manera normal, hasta el momento es un programa que nos muestra el número que nosotros tecleamos, pero al depurar el programa y ejecutarlo después de un breack establecido (el visto en primer proceso desde la ejecución en la línea de código), notamos algo peculiar, y resulta que la verdadera funcionalidad del programa es realizar una suma secundaria invisible la cual no es apreciable en los resultados del número expresado al final, bastante irregular si solo se ejecuta de manera general el programa sin depurarlo.



Segundo programa:

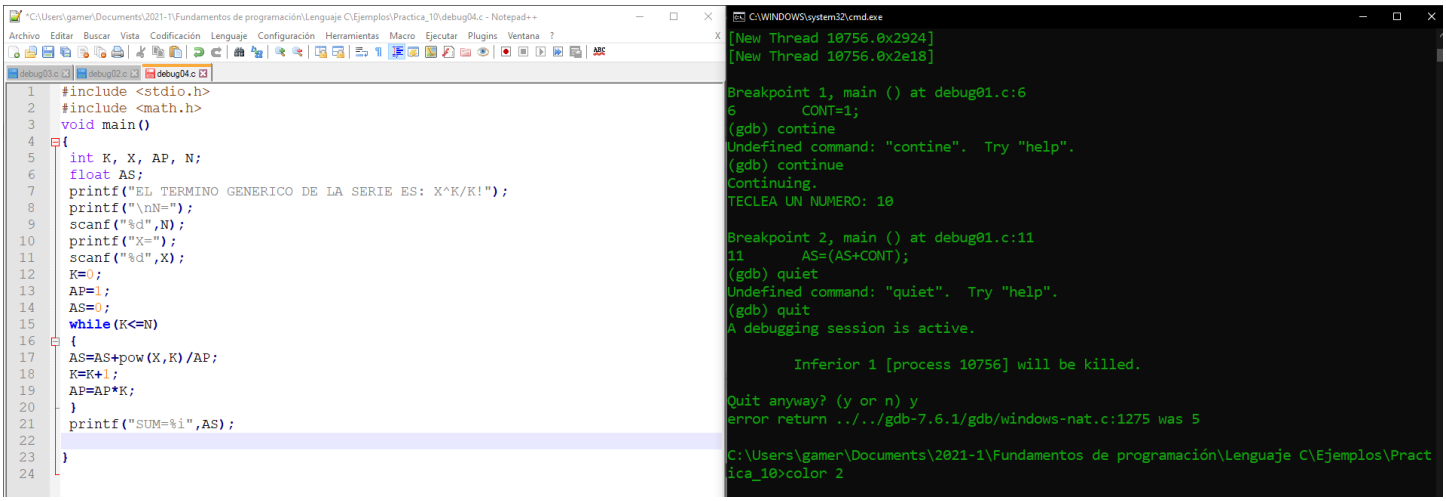
En esta ocasión ocurrió un problema con la anidación de las variables for y con las igualaciones que esta tenía en dicho ciclo, en el primer for, el cual iba haciendo crecer el valor del numero unitario por el cual se tenía que ir consiguiendo las tablas simplemente no existía porque el valor de for estaba restringido a la igual, suceso que nunca iba a pasar debido a que el valor de la variable "i" se encontraba dado previamente y nunca iba a igualar a diez, por ello que si se expresaba de manera continua la tabla de multiplicar pero no su respectivo resultado, tras depurar el programa y darme cuenta de llo la solución fue primeramente separar las anidaciones de cada for y posteriormente añadir el signo igual faltante en el primer for, mismo que sería empleado en el siguiente para indicar la tabla en cuestión y el resultado, en el segundo for se añadió el símbolo igual para que pudiera terminar con el ciclo y así imprimir en la pantalla la funcionalidad completa del programa desde un inicio, además del tipo de valor declarado para imprimir los primeros casos del programa es "%i", cuando al traterse de valores enteros debería ser "%d".





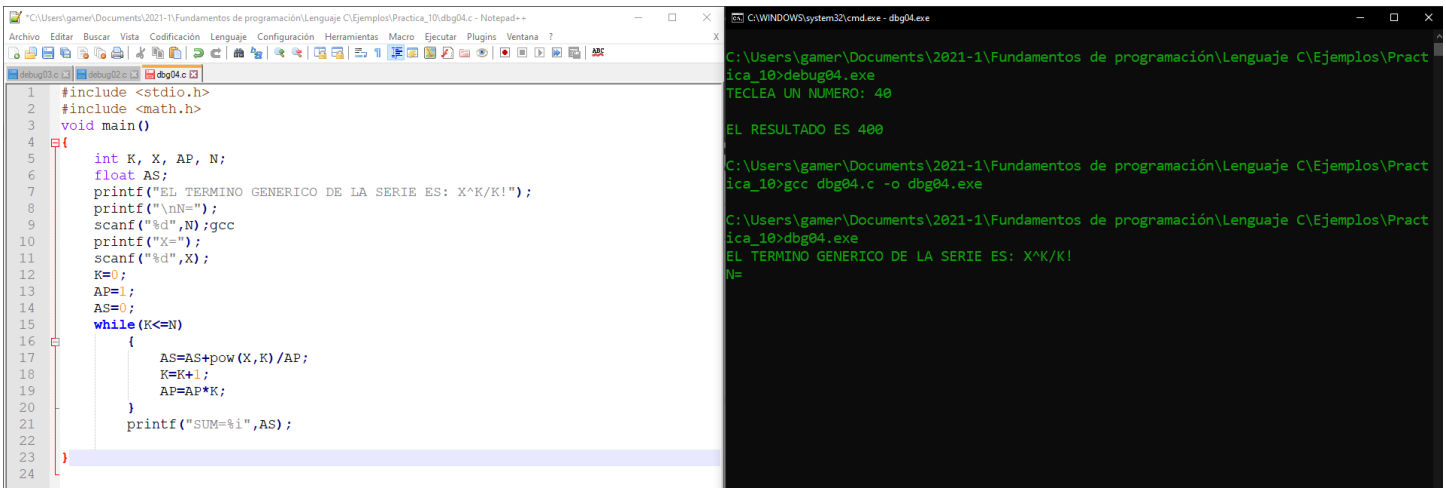
Tercer programa:

Este programa fue un poco más complicado que los anteriores debido a que para la correcta depuración tuve que ir realizando proceso a proceso varias veces para encontrar el verdadero problema donde yacía la inexistente funcionalidad del ejecutable, al parecer no presentaba ningún probable al momento de compilar el programa, sin embargo al momento de ejecutarlo la operación simplemente se detenía sin dar un solo indicio del por qué ocurría esto, para solución y como no había ni siquiera las instrucciones de qué tenía que ejecutar o hacer lel programa, fui desglosando parte por parte, en primer lugar citando algunas líneas de código, las cuales, me permitieron darme cuenta que tenía que realizar cierta operación y mostrar un resultado determinado, para este fin inicié por declara las funciones correspondientes y noté que una de las no estaba funcionando correctamente debido a que tenía operaciones de más en su ejecución las cuales provocaban un bucle, para solucionar este proceso restringí las operaciones en la función while, mismas que tenías una relación directa con los datos solicitas de manera inicial por ello no se mostraba una sola respuesta, porque al parecer, los datos que nosotros ingresamos al programa generan archivos basura por el tipo de dato y el tipo de salida que se desea para el final de la ejecución.



```
#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("X=");
 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=%i",AS);

}
```

```
[New Thread 10756.0x2924]
[New Thread 10756.0x2e18]

Breakpoint 1, main () at debug01.c:6
6       CONT=1;
(gdb) contine
Undefined command: "contine".  Try "help".
(gdb) continue
Continuing.
TECLEA UN NUMERO: 10

Breakpoint 2, main () at debug01.c:11
11      AS=(AS+CONT);
(gdb) quiet
Undefined command: "quiet".  Try "help".
(gdb) quit
A debugging session is active.

        Inferior 1 [process 10756] will be killed.

Quit anyway? (y or n) y
error return ../../gdb-7.6.1/gdb/windows-nat.c:1275 was 5

C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Pract
ica_10>color 2
```



```
#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);gcc
    printf("X=");
    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=%i",AS);

}
```

```
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Pract
ica_10>debug04.exe
TECLEA UN NUMERO: 40

EL RESULTADO ES 400

C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Pract
ica_10>gcc dbg04.c -o dbg04.exe

C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Pract
ica_10>dbg04.exe
EL TERMINO GENERICO DE LA SERIE ES: X^K/K!
N=
```

Conclusión:

Para mí ha sido hasta ahora la práctica más complicada del curso debido a que el contenido y los métodos de depuración requieren no únicamente de un gran conocimiento del entorno de programación en C, sino que requiera además una alta capacidad de raciocinio lógico para buscar la solución verdaderamente plantead a un problema real, visto de manera más clara en el ultimo programa presentado, donde no se nos decía ni siquiera el propósito del programa inicial que tenía que ser depurado para su correcto funcionamiento, sino que de entrada nos daba el programa no resuelto y nosotros teníamos que hacerlo funcionar de manera correcta en su ejecución por medio de la depuración del mismo, un problema de clase ahora, pero me imagina que en el futuro puede llegar a ser el día a día debido a que esos son los verdaderos problemas de un programador, supongamos en un caso hipotético en el que una persona especialista en este tema es contratada para laborar, normalmente los programas que tenga que depurar no son acompañados por una serie de instrucciones de su funcionamiento ni mucho menos constan de un código tan sencillo como el que nosotros tenemos que ver en la clase, para las situaciones reales hay que ser muy pacientes y estar muchas horas detrás de la consola para lograr comprender la manera en que la computadora va interpretando las señales propuestas por el código original, es ahí donde radica la complejidad d ellos sistemas avanzados en relación con a resolución de programas no depurados, pero sobre todo con el ambiente técnico e incluso el estrés que puede provocar el tener delante un problema como el que nos han planteado y que de ello dependa la vida laboral, por eso mismo resulta de vital importancia el estar cada vez más dentro de este mundo y poner en práctica cada concepto evaluado en las prácticas para no solamente conocerlo, sino dominarlo.