	Carátula para entrega de prácticas	
Facultad de Ingeniería	Laboratorio de docencia	



Laboratorios de computación salas A y B

<i>Profesor:</i>	Ing. Karina García Morales
<i>Asignatura:</i>	Fundamentos de Programación
<i>Grupo:</i>	1121
<i>No de Práctica(s):</i>	10
<i>Integrante(s):</i>	Jacinto Rodríguez Moisés Rodrigo
<i>No. de Equipo de cómputo empleado</i>	06
<i>Semestre:</i>	2019-1
<i>Fecha de entrega:</i>	30/10/2018
<i>Obervaciones:</i>	

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.

Desarrollo

Error: Es provocado por un error humano

Defecto: Es provocado por una falla en la ejecución del programa y se detiene

Falla: Se debe a cuando una instrucción no llega a un fin

Punto de ruptura: Define qué es lo que deseamos ejecutar

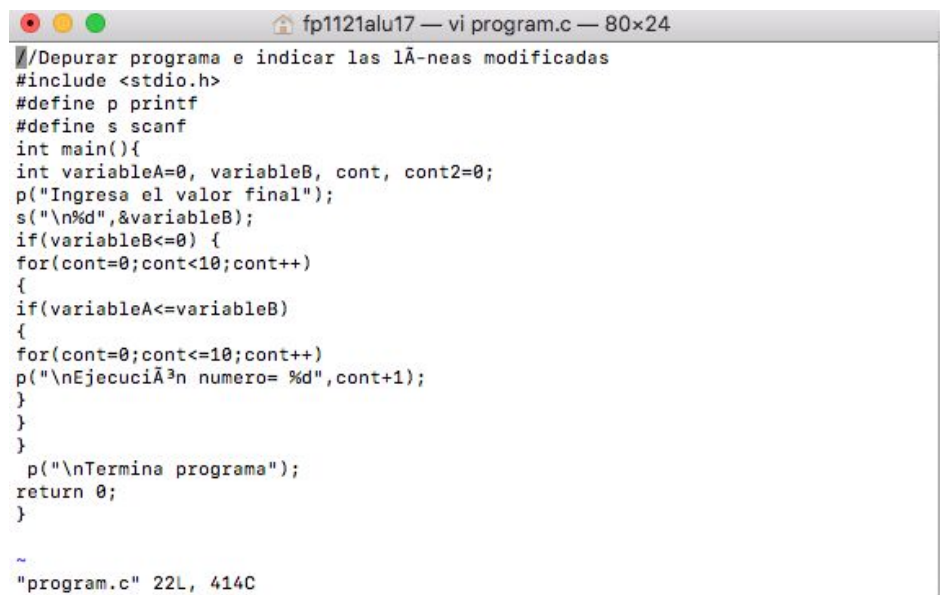
GCC: Es el compilador

GDB: Es el depurador para cualquier programa ejecutable

información de depuración: Sirve para ver el valor de las variables, avanzar en las instrucciones, colocar y quitar puntos de ruptura

```
//Depurar programa e indicar las líneas modificadas
#include <stdio.h>
#define p printf
#define s scanf
int main(){
int variableA=0, variableB, cont, cont2=0;
p("Ingresa el valor final");
s("\n%d",&variableB);
if(variableB<=0) {
for(cont=0;cont<10;cont++)
{
if(variableA<=variableB)
{
for(cont=0;cont<10;cont++)
p("\nEjecución numero= %d",cont+1);
}
}
}
p("\nTermina programa");
return 0; }
```

Programa en el terminal



```
fp1121alu17 — vi program.c — 80x24
//Depurar programa e indicar las líneas modificadas
#include <stdio.h>
#define p printf
#define s scanf
int main(){
int variableA=0, variableB, cont, cont2=0;
p("Ingresa el valor final");
s("\n%d",&variableB);
if(variableB<=0) {
for(cont=0;cont<10;cont++)
{
if(variableA<=variableB)
{
for(cont=0;cont<=10;cont++)
p("\nEjecución numero= %d",cont+1);
}
}
}
p("\nTermina programa");
return 0;
}

"program.c" 22L, 414C
```

Comienzo de la compilación del programa.



```
fp1121alu17 — -bash — 80x24
Arabia06:~ fp1121alu17$ gcc -g -o program program.c
Arabia06:~ fp1121alu17$
```

agregar captura de gcc -g -o program program.c
captura 20:10

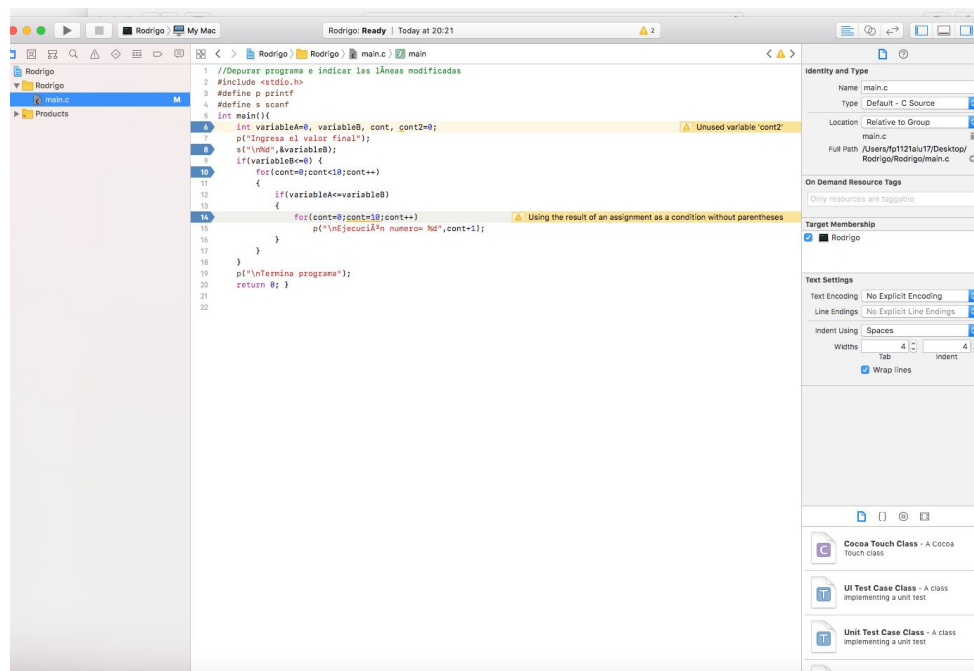
Compilación del programa

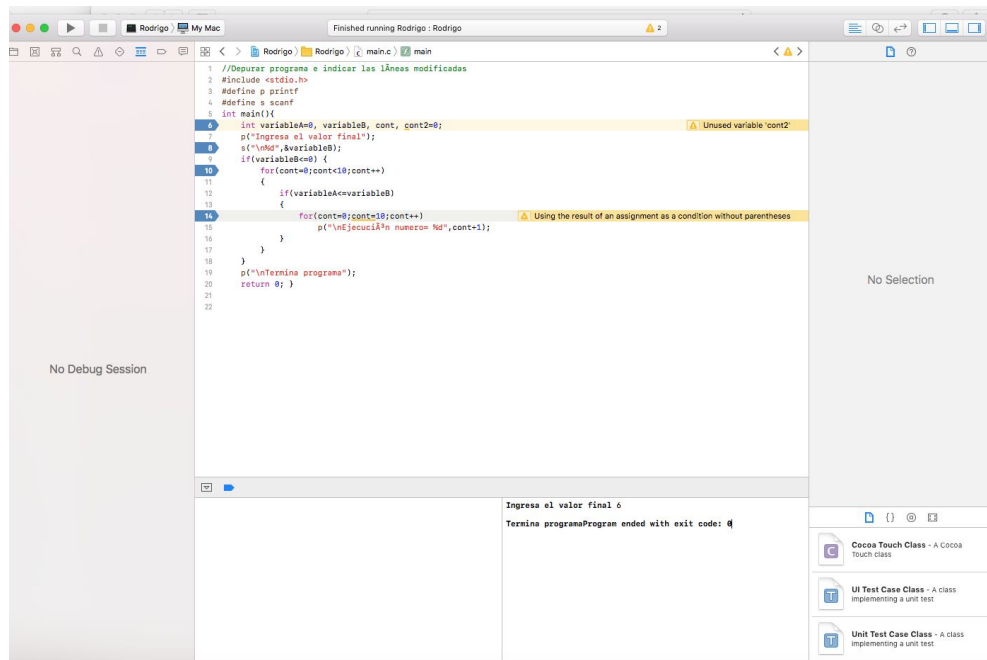
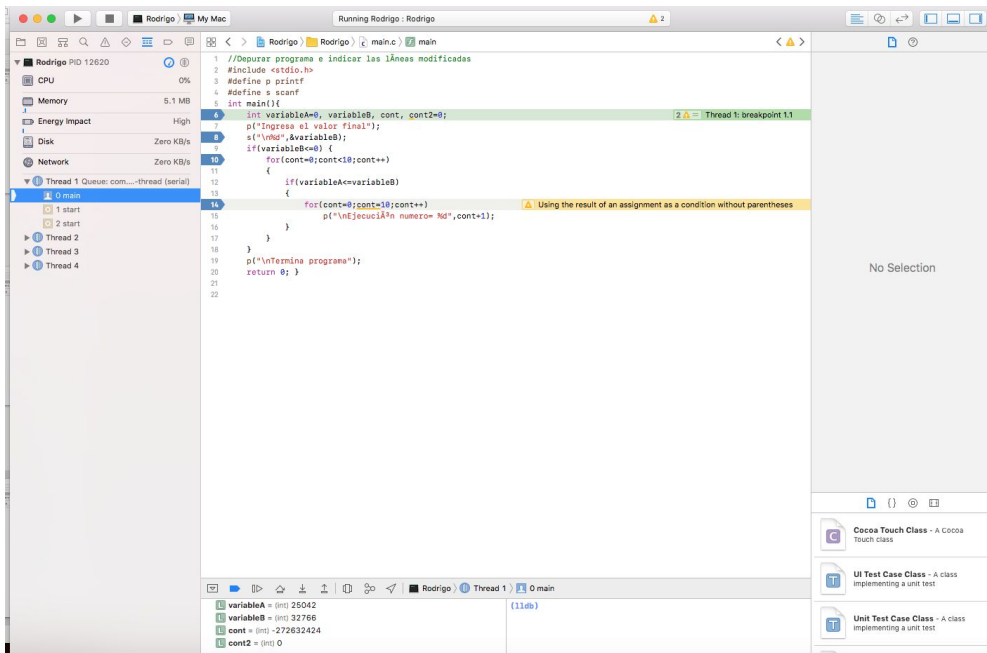


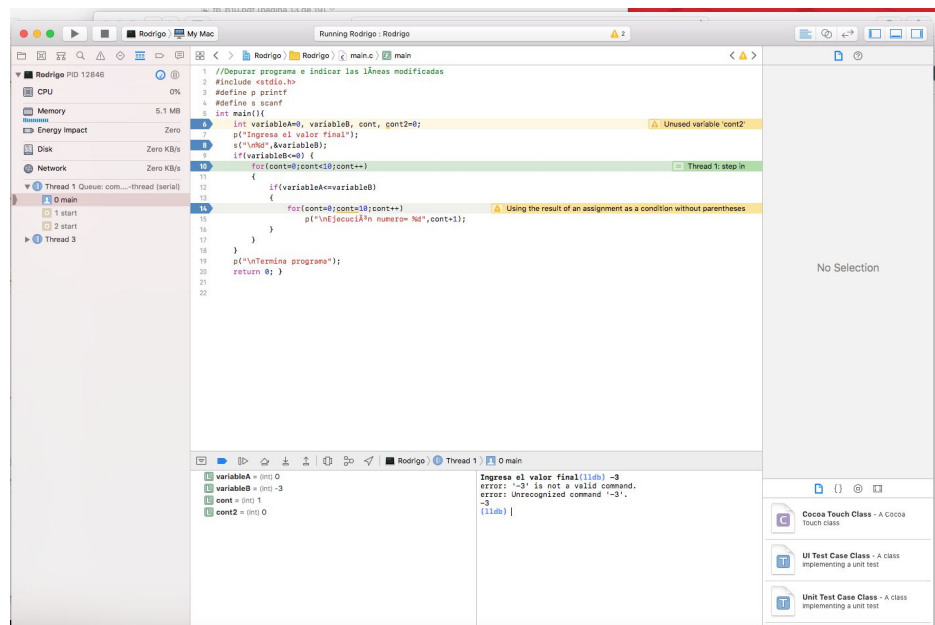
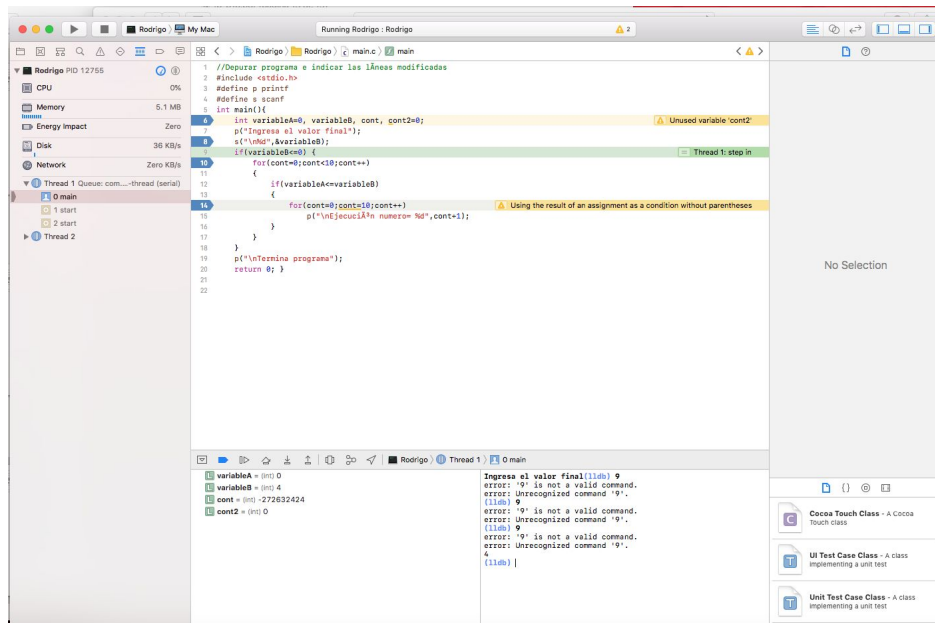
```
fp1121alu17 — -bash — 80x24
[Arabia06:~ fp1121alu17$ gcc -g -o program program.c
[Arabia06:~ fp1121alu17$ gcc -g -o program program.c
[Arabia06:~ fp1121alu17$
```

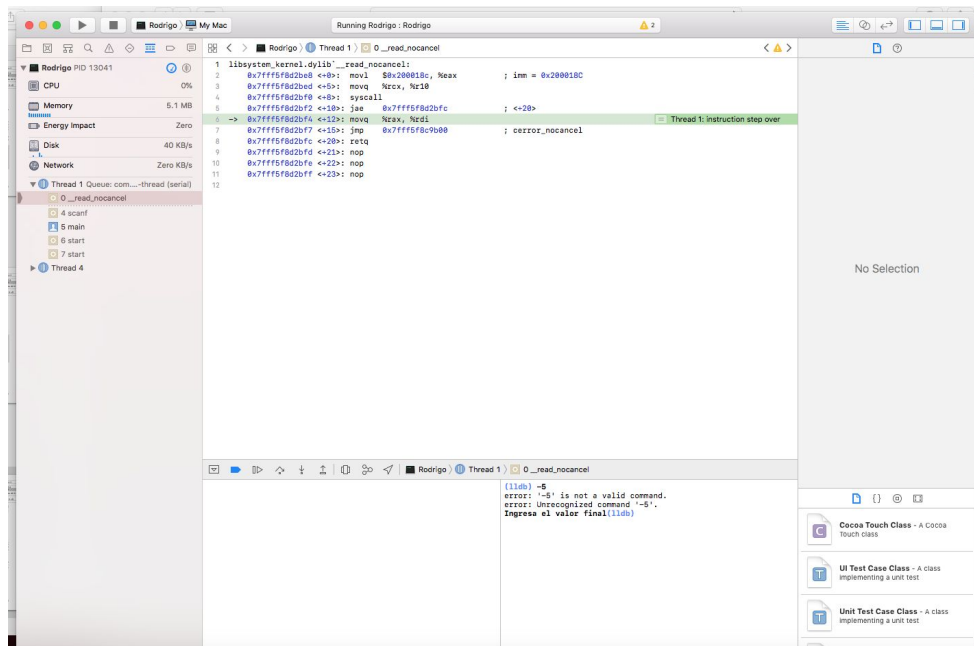
Los puntos de ruptura indican en qué parte del código se detienen los valores.

Puntos de ruptura









¿Para qué sirve la depuración?

La depuración es de gran utilidad para encontrar errores dentro del programa hecho, en este caso nuestro programa es óptimo y eficiente, liberamos espacio en memoria y se manifestaba paso por paso lo que está ocurriendo en los ciclos nos ayudó a identificar en el programa el uso de variables extra que no eran utilizadas, logramos identificar los errores humanos como de sintaxis, nos ayudó en la depuración a identificar el valor de las variables

encontrar la falla con el apuntador

comentar el programa de violación de segmento

TAREA

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.

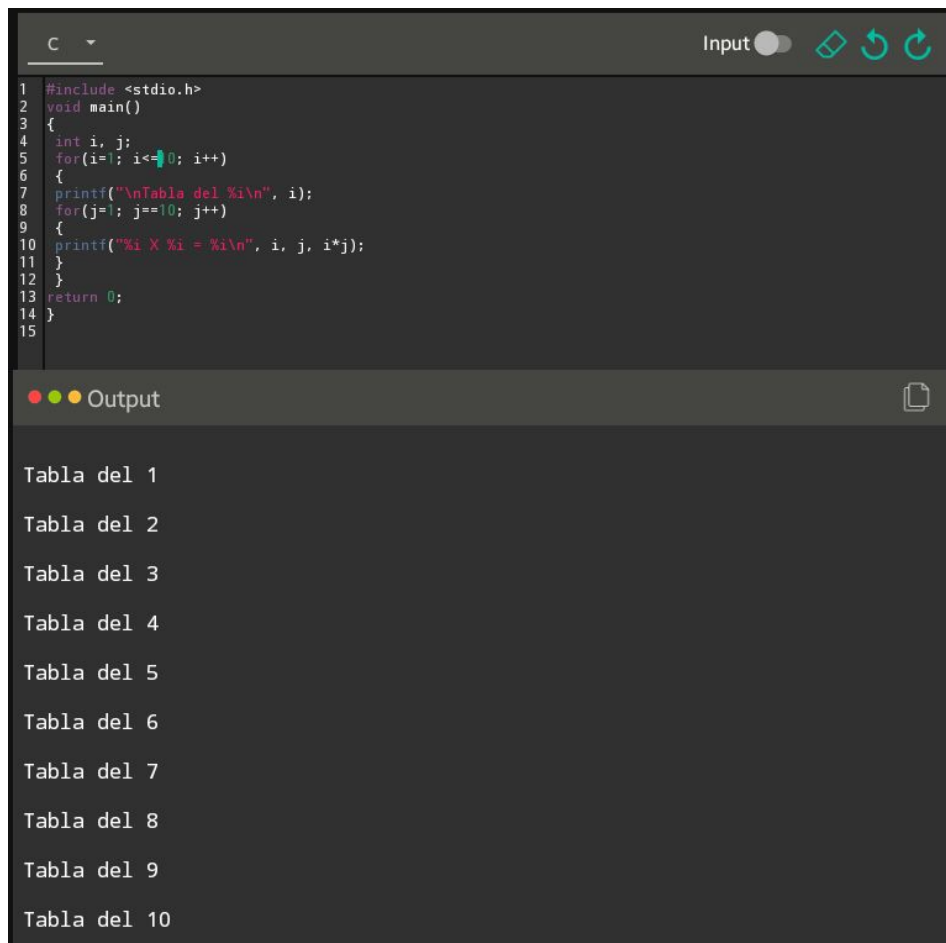


The image shows a terminal window with a dark background. At the top, there is a tab labeled 'C' and a button labeled 'Entrada' with a green circle icon. Below the tab, the C code is displayed with line numbers 1 through 16. The code includes `<stdio.h>`, defines `main()`, and declares variables `N`, `CONT`, and `AS`. It prompts the user to enter a number, reads it into `N`, and enters a `while` loop where `AS` is incremented by `CONT` and `CONT` is incremented by 2, until `CONT` is greater than `N`. Finally, it prints the value of `AS` and returns 0. Below the code, there is an 'Output' section with a document icon, showing the program's execution: it prompts 'TECLEA UN NUMERO:', the user enters '9', and it outputs 'EL RESULTADO ES 9'.

```
1 #include <stdio.h>
2 void main()
3 {
4     int N, CONT, AS;
5     AS=0;
6     CONT=1;
7     printf("TECLEA UN NUMERO: ");
8     scanf("%i",&N);
9     while(CONT<=N)
10    {
11        AS=(AS+CONT);
12        CONT=(CONT+2);
13    }
14    printf("\nEL RESULTADO ES %i\n", AS);
15    return 0;
16 }
```

TECLEA UN NUMERO:
EL RESULTADO ES 9

El programa anterior lee el número introducido por parte del usuario para que después entre en los ciclos siguientes; y con ello se hace una suma +1 de un ciclo y otra de +2, siendo una impresión total de +3 en el resultado mostrado.



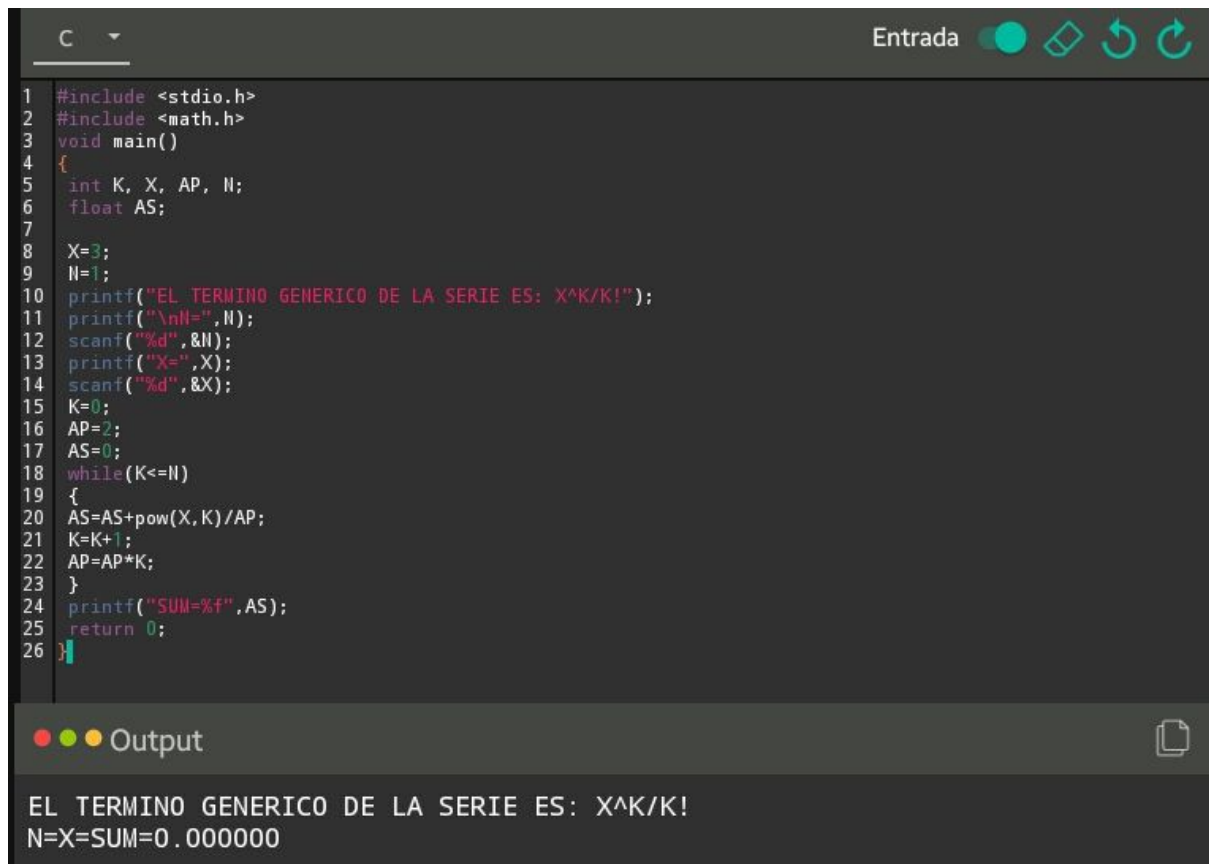
The image shows a code editor window with a dark theme. At the top, there's a tab labeled 'C' and a toolbar with an 'Input' toggle, a copy icon, and a refresh icon. The code is as follows:

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

Below the code editor is an 'Output' window. It contains the following text:

```
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
Tabla del 10
```

En el programa anteriormente manifestado; el error del porqué no presentaba la opción de la tabla del 10 era por la razón de que en el primer ciclo for se manifestaba que “i<10” por lo que solo se veía hasta la tabla del 9 y al usar “i<=10” incluye la tabla del 10 porque se especifica que “i” puede tomar el valor de 10.



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

Entrada

Output

EL TERMINO GENERICO DE LA SERIE ES: X^K/K!
N=X=SUM=0.000000

Ocurrió una violación de segmento porque no se hizo el uso de “%” para leer las variables correspondientes (en scanf), así como para “printf”; en ambos casos no se pudo obtener la dirección de memoria correspondiente de cada caso.

Conclusión

La depuración es un proceso fundamental para la programación básica; la ayuda que nos brinda a encontrar los errores correspondientes de nuestro programa es de gran avance para agilizar los procesos que podemos llevar a cabo y por ende, saber con mayor exactitud la causa de esos fallos.

El hecho de practicar y tener conocimiento de aspectos tan básicos pero fundamentales hace que podamos ser más competitivos y logremos desarrollar determinadas aptitudes para organizar un proceso de manera más productiva y eficiente a la hora de elaborar un programa en esta plataforma.