	Carátula para entrega de prácticas	
Facultad de Ingeniería		Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Alejandro Esteban Pimentel Alarcón

Asignatura: Fundamentos de Programación

Grupo: 3

No de Práctica(s): #10

Integrante(s): Miranda Gutiérrez Celine, Torres Mendoza Alexa Erandy

*No. de Equipo de
cómputo empleado:*

No. de Lista o Brigada: 30, 49 Números de cuenta

Semestre: 2020-1

Fecha de entrega: 28/octubre/2019

Observaciones: Bien, pero recuerden usar sus números de cuenta

CALIFICACIÓN: 10

- **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**

La depuración de un programa es la forma de saber si un programa contiene errores o no, así mismo también nos ayuda a corregir dichos errores.

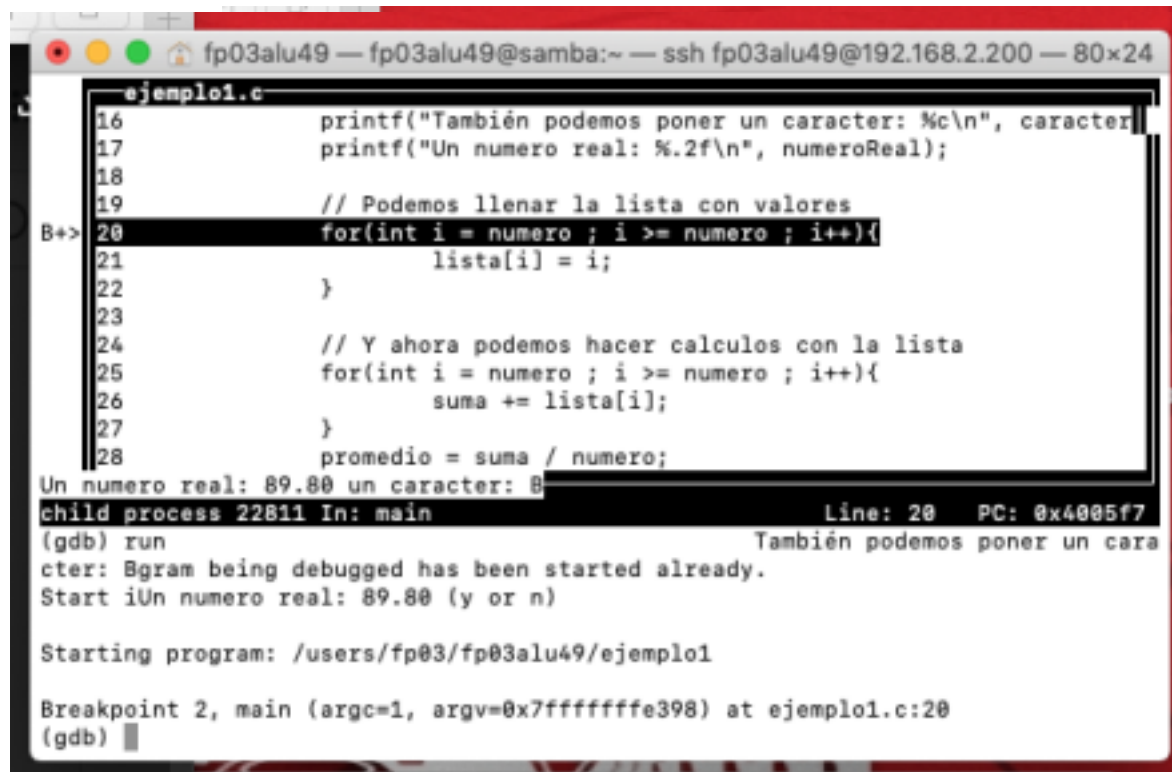
Se dice que un programa esta depurado cuando está libre de errores.

Cuando se depura un programa se hace un seguimiento del funcionamiento de dicho programa y se van estudiando los valores de las distintas variables, así como los resultados obtenidos en las operaciones.

Una vez depurado el programa se solucionan los posibles errores encontrados y se procede a depurar otra vez.

Estas acciones se repiten hasta que el programa no contiene ningún tipo de errores, tanto en tiempo de programación como en tiempo de ejecución.

Ejemplo:



```
fp03alu49 — fp03alu49@samba:~ — ssh fp03alu49@192.168.2.200 — 80x24
ejemplo1.c
16      printf("También podemos poner un caracter: %c\n", caracter);
17      printf("Un numero real: %.2f\n", numeroReal);
18
19      // Podemos llenar la lista con valores
B+> 20      for(int i = numero ; i >= numero ; i++){
21          lista[i] = i;
22      }
23
24      // Y ahora podemos hacer calculos con la lista
25      for(int i = numero ; i >= numero ; i++){
26          suma += lista[i];
27      }
28      promedio = suma / numero;
Un numero real: 89.80 un caracter: B
child process 22811 In: main                               Line: 20   PC: 0x4005f7
(gdb) run                                                  También podemos poner un cara
cter: Bgram being debugged has been started already.
Start iUn numero real: 89.80 (y or n)

Starting program: /users/fp03/fp03alu49/ejemplo1

Breakpoint 2, main (argc=1, argv=0x7ffffffe398) at ejemplo1.c:20
(gdb) █
```

```
fp03alu49 — fp03alu49@samba:~ — ssh fp03alu49@192.168.2.200 — 80x24
ejemplo1.c
15     printf("Luego podemos poner un entero: %i\n", numero);
16     printf("También podemos poner un caracter: %c\n", caracter
17     printf("Un numero real: %.2f\n", numeroReal);
18
19     // Podemos llenar la lista con valores
20     for(int i = numero ; i >= numero ; i++){
21         lista[i] = i;
22     }
23
24     // Y ahora podemos hacer calculos con la lista
25     for(int i = numero ; i >= numero ; i++){
26         suma += lista[i];
27     }
Un numero real: 89.80 un caracter: B
child process 22811 In: main Line: 21 PC: 0x4005ff
1: i = 11
1: i = 11
1: i = 12
1: i = 12
1: i = 13
1: i = 13
1: i = 14
(gdb)
```

ACTIVIDAD 1

Utilizar GDB para encontrar la utilidad del programa y describir su funcionalidad.

The screenshot shows the OnlineGDB interface with a C program being debugged. The program's logic is as follows:

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

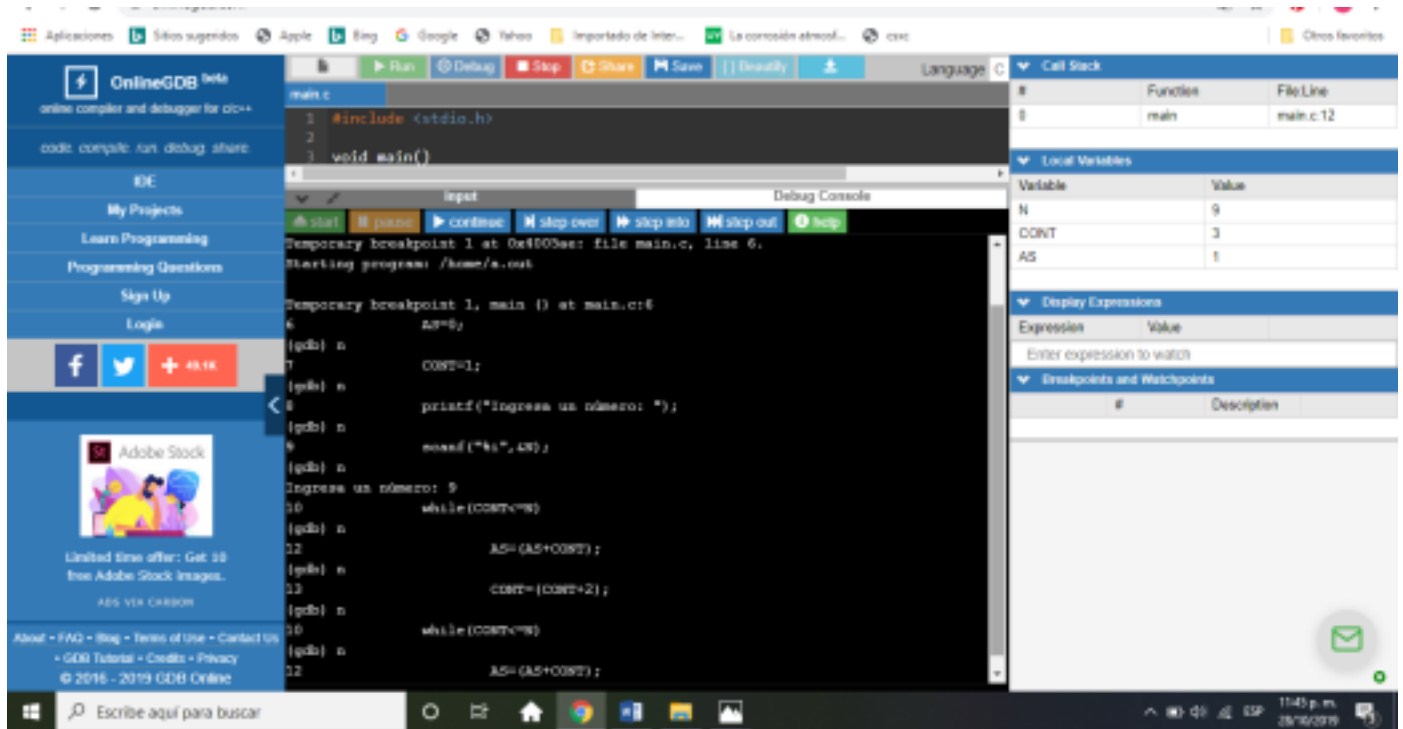
The current state of the program is shown in the 'Local Variables' table:

Variable	Value
N	9
CONT	3
AS	1

The 'Display Expressions' table shows the current expression being evaluated:

Expression	Value
AS=(AS+CONT);	10

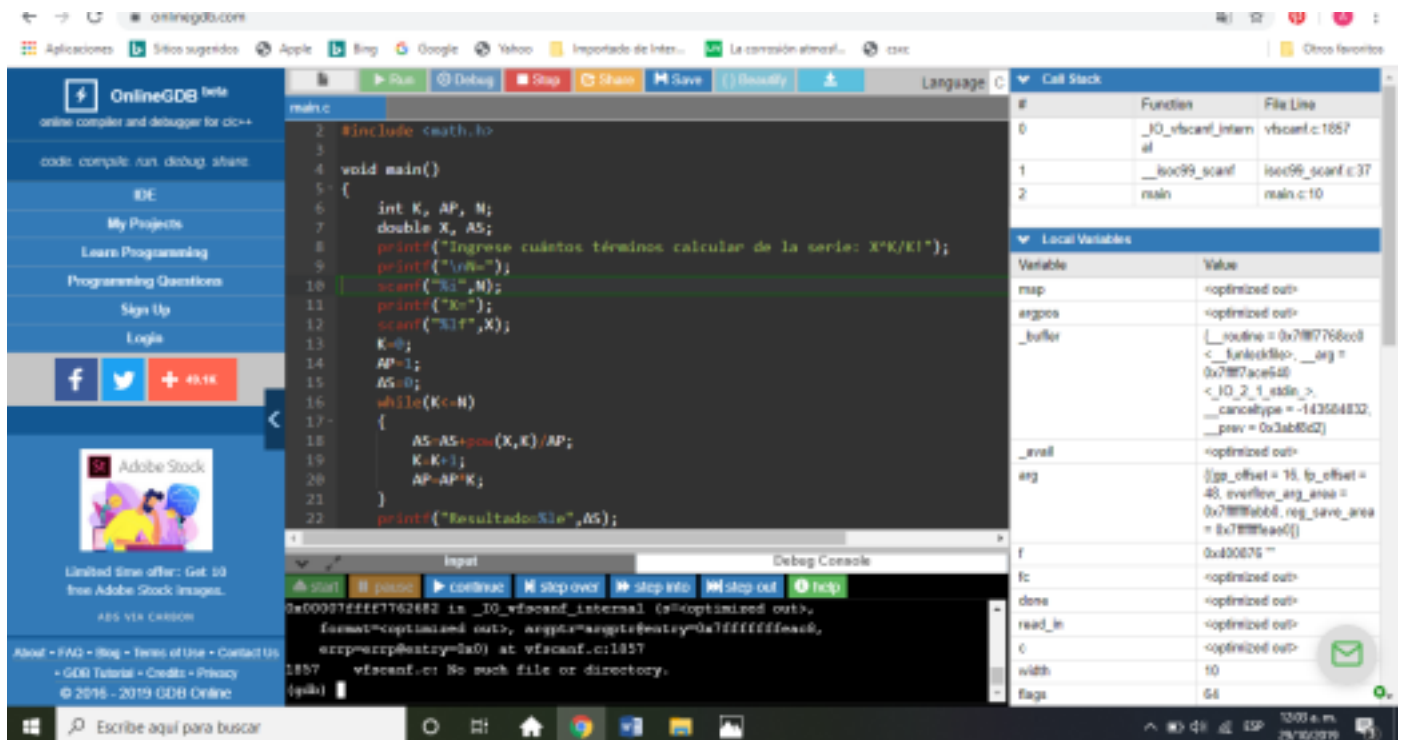
The 'Breakpoints and Watchpoints' table is empty.

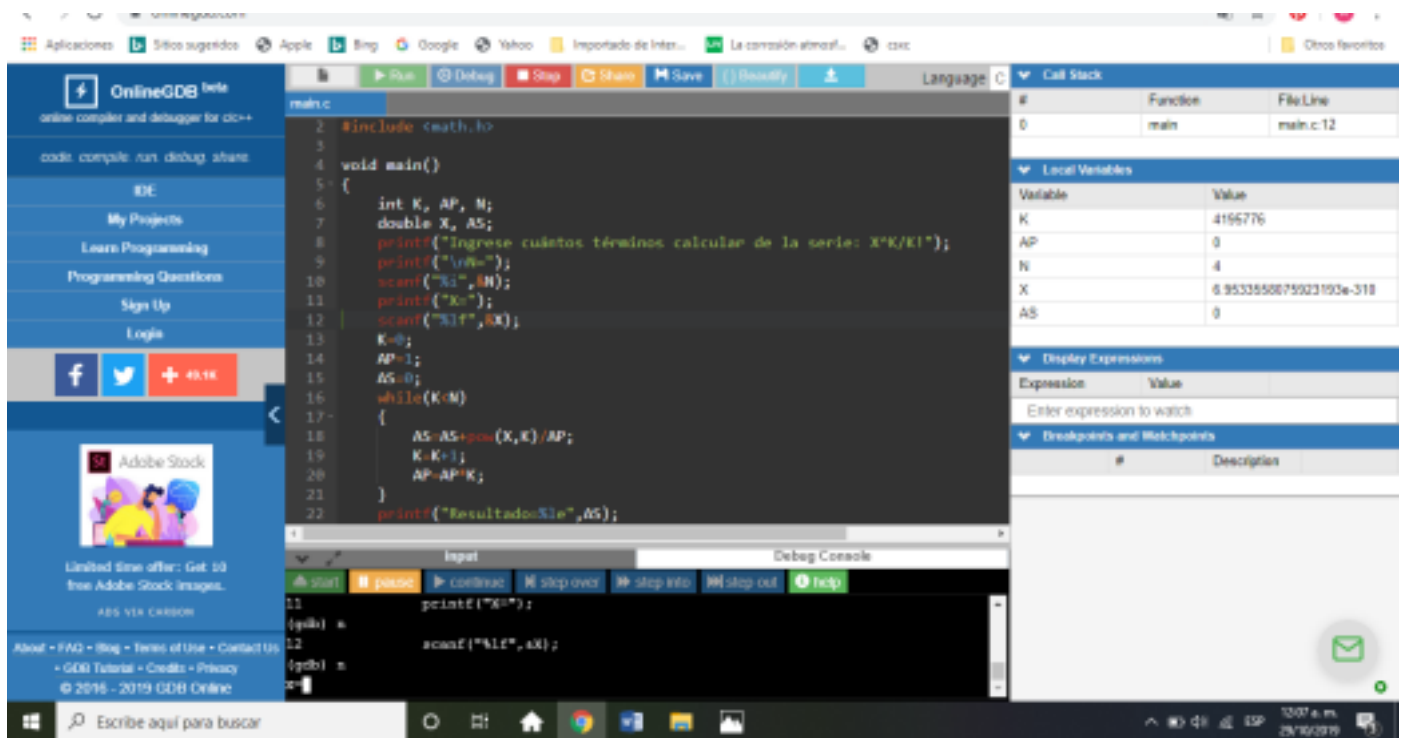


La funcionalidad del programa es recibir un número, número, después en otra variable llamada AS se va guardando el nuevo resultado de la suma de AS más el contador que igual es una variable que aumentará de dos en dos e inicia desde 1

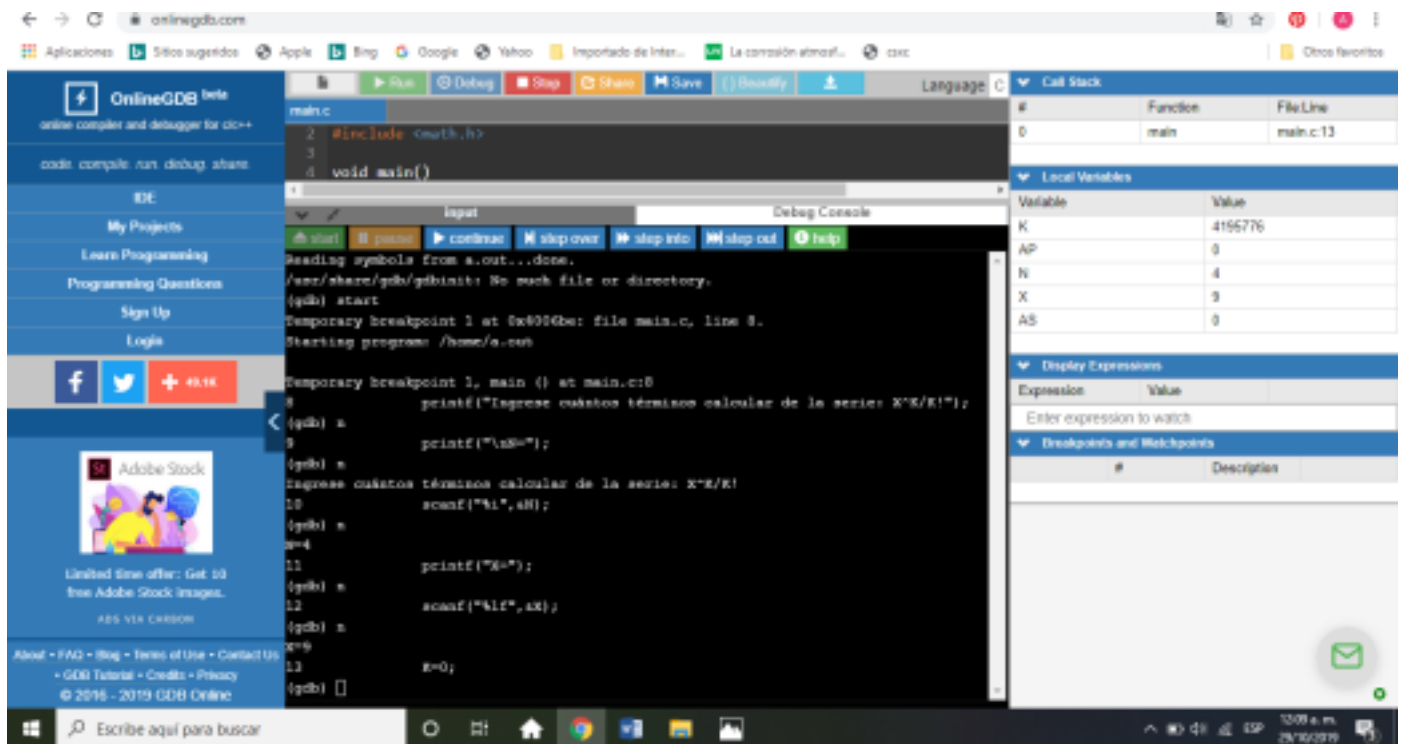
ACTIVIDAD 2

Utilizar GDB para corregir el programa. NOTA: para compilar el código de la actividad, ejecutar:
\$ gcc -w actividad2.c -o actividad2 -lm



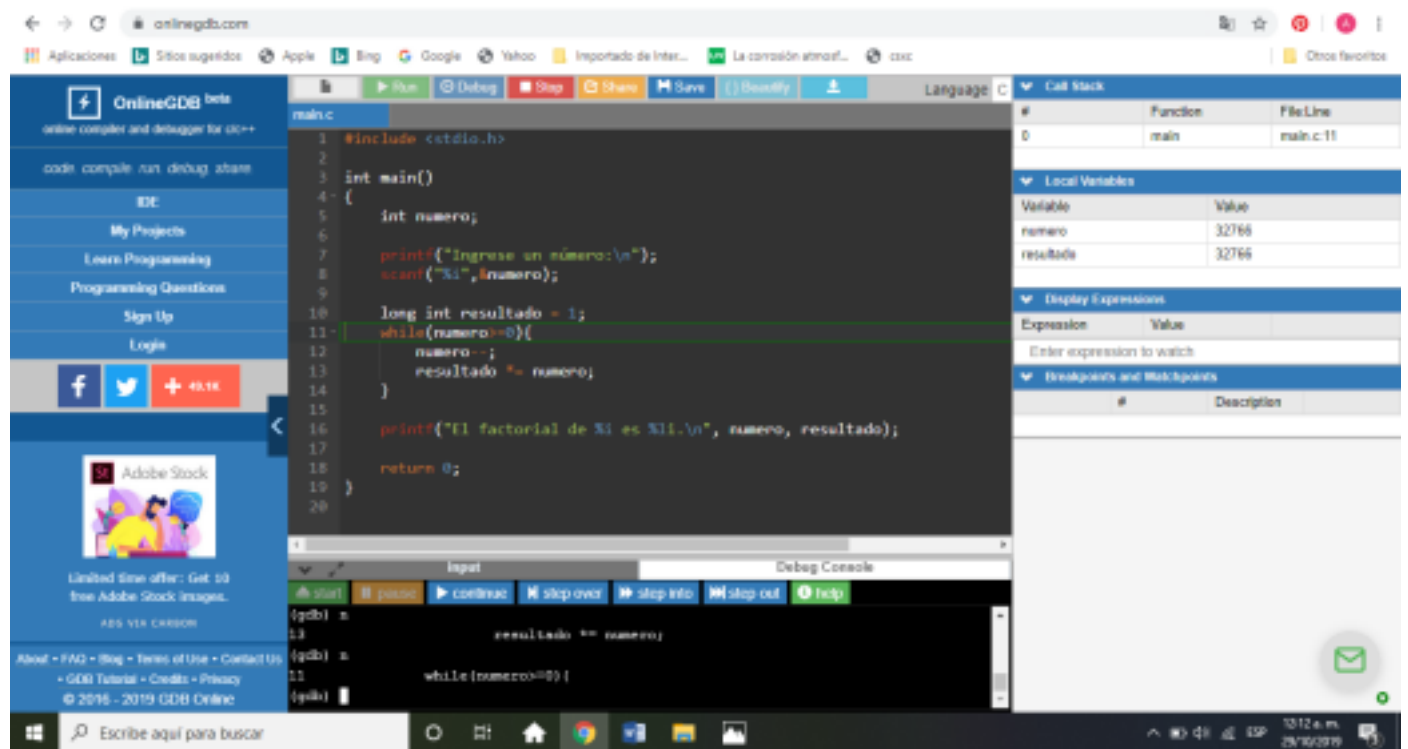


El error era que a los scanf les faltaba "&" para que el programa estuviera bien ejecutado, y quitar el igual a el while



ACTIVIDAD 3

Utilizar GDB para corregir el programa.

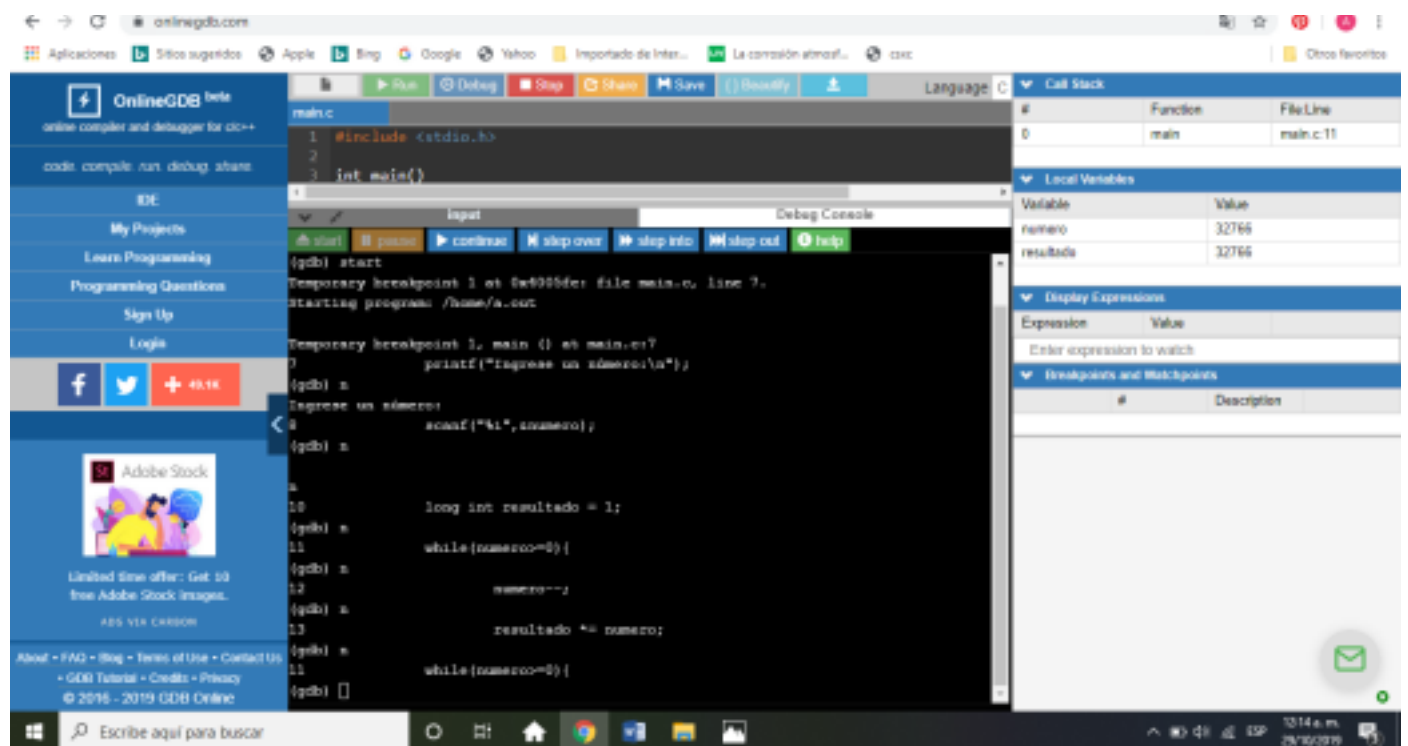


The screenshot shows the OnlineGDB website interface. The main editor displays a C program for calculating the factorial of a number. The program is as follows:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int numero;
6
7     printf("Ingrese un número:\n");
8     scanf("%i", &numero);
9
10    long int resultado = 1;
11    while(numero > 0){
12        numero--;
13        resultado *= numero;
14    }
15
16    printf("El factorial de %i es %i.\n", numero, resultado);
17
18    return 0;
19 }
20
```

The right sidebar shows the following information:

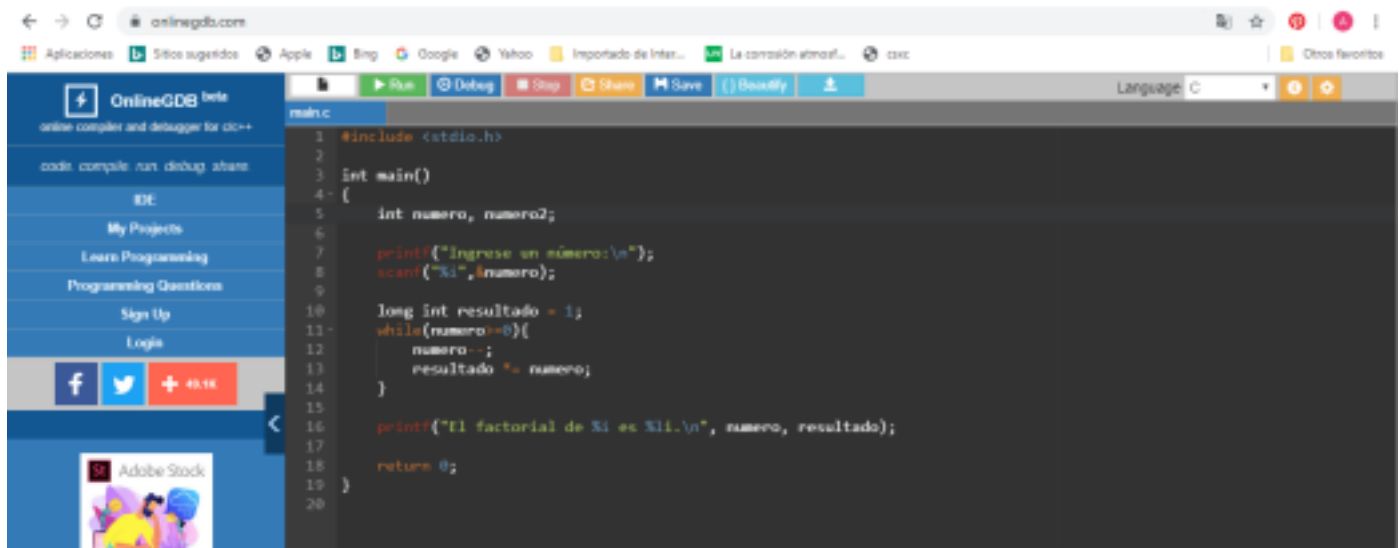
- Call Stack:** A table with columns #, Function, and File Line. It shows one entry: #0, main, main.c:11.
- Local Variables:** A table with columns Variable and Value. It shows two entries: numero (32766) and resultado (32766).
- Display Expressions:** A section for entering expressions to watch.
- Breakpoints and Watchpoints:** A section for setting breakpoints and watchpoints.



The screenshot shows the OnlineGDB website interface with the GDB console open. The console shows the following output:

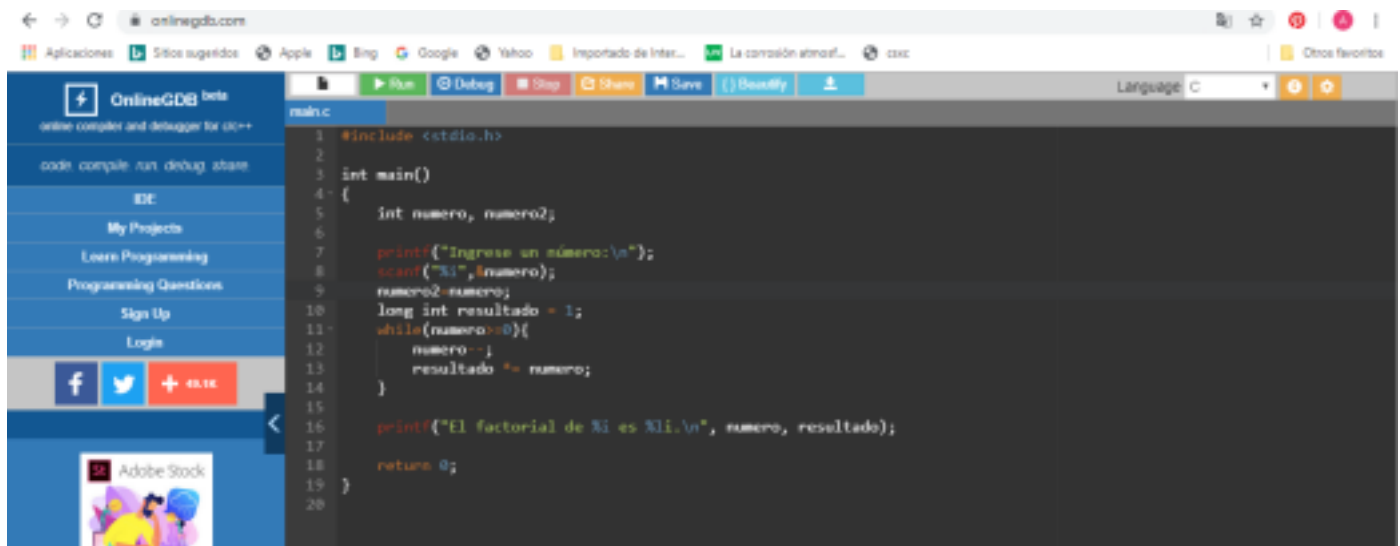
```
(gdb) start
Temporary breakpoint 1 at 0x4005df: file main.c, line 7.
Starting program: /home/a.out
Temporary breakpoint 1, main () at main.c:7
7     printf("Ingrese un número:\n");
(gdb) a
Ingrese un número:
8     scanf("%i", &numero);
(gdb) a
9
10    long int resultado = 1;
(gdb) a
11    while(numero > 0){
12        numero--;
(gdb) a
13        resultado *= numero;
(gdb) a
14    }
15
16    printf("El factorial de %i es %i.\n", numero, resultado);
17
18    return 0;
(gdb) a
19 }
```

Se le tiene que agregar una variable porque al final se imprime numero, pero el resultado final de numero es 0 así que se agrega numero2



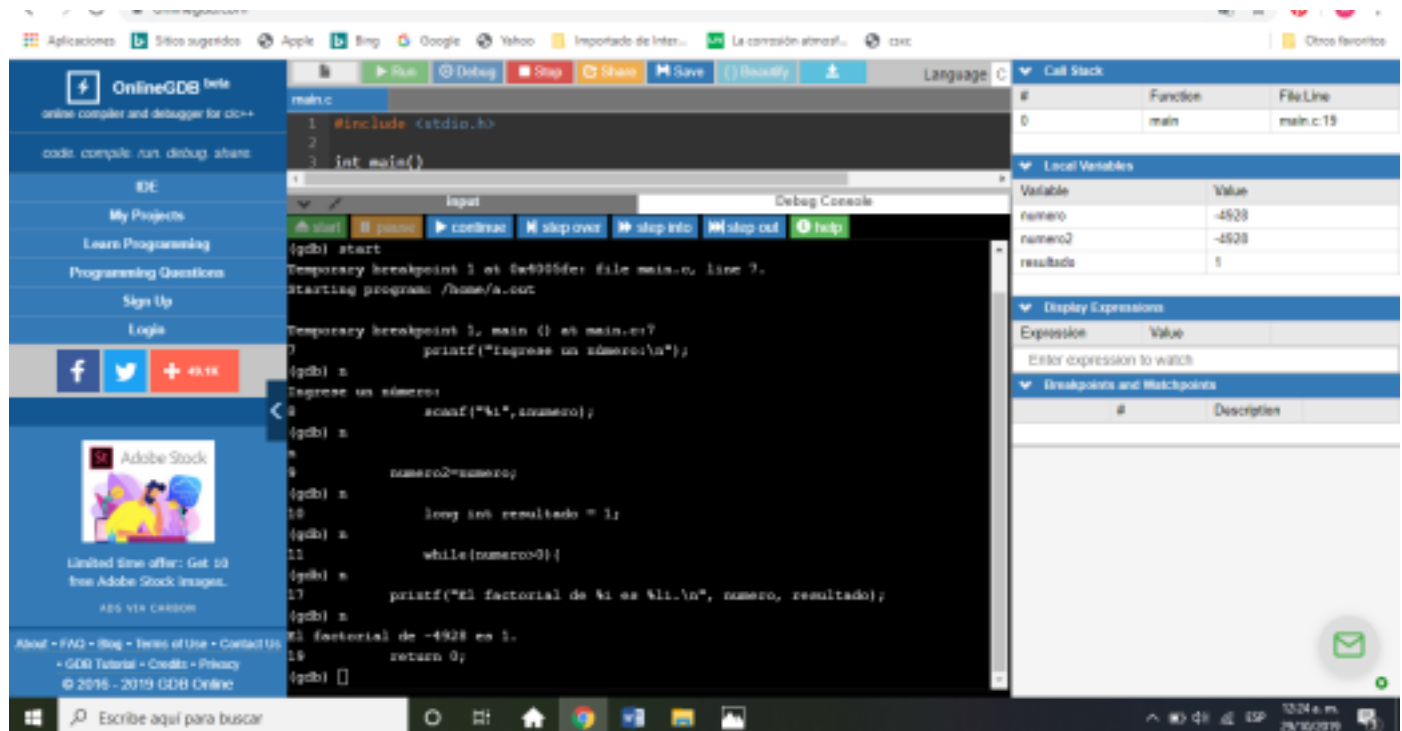
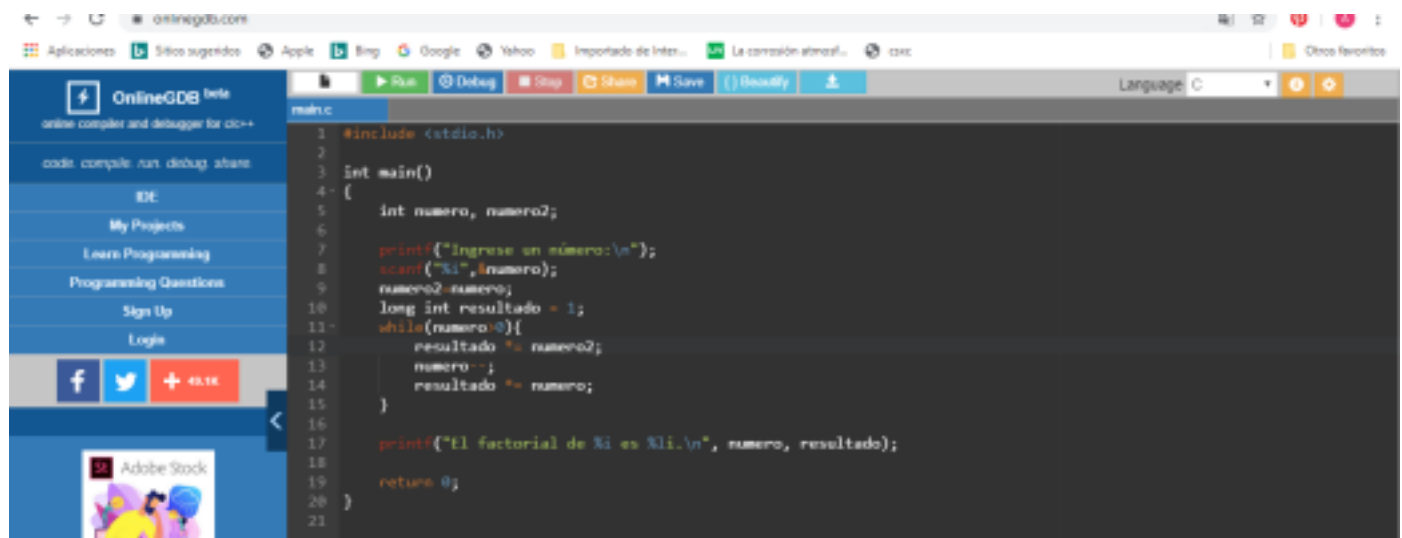
```
1 #include <stdio.h>
2
3 int main()
4 {
5     int numero, numero2;
6
7     printf("Ingrese un número:\n");
8     scanf("%i", &numero);
9
10    long int resultado = 1;
11    while(numero >= 0){
12        numero--;
13        resultado *= numero;
14    }
15
16    printf("El factorial de %i es %i.\n", numero, resultado);
17
18    return 0;
19 }
20
```

Numero 2 es igual a numero para mantener su función



```
1 #include <stdio.h>
2
3 int main()
4 {
5     int numero, numero2;
6
7     printf("Ingrese un número:\n");
8     scanf("%i", &numero);
9     numero2 = numero;
10    long int resultado = 1;
11    while(numero2 > 0){
12        numero--;
13        resultado *= numero;
14    }
15
16    printf("El factorial de %i es %i.\n", numero, resultado);
17
18    return 0;
19 }
20
```

Se cambia el numero2 >= 0 a numero2 > 0 esto para evitar la multiplicación por 0
Y se cambia la posición de numero2, para que se restara hasta el final y no al principio del proceso



CONCLUSIONES

La depuración de programas es el proceso de identificar y corregir errores de programación. En inglés se conoce como debugging, porque se asemeja a la eliminación de bichos (bugs), manera en que se conoce informalmente a los errores de programación.

Muchas veces se requiere incluir en el código fuente instrucciones auxiliares que permitan el seguimiento de la ejecución del programa, presentando los valores de variables y direcciones de memoria y ralentizando la salida de datos ("modo de depuración"). Dentro de un proceso formal de aseguramiento de la calidad, puede ser asimilado al concepto de "prueba unitaria".