	<b>Caratula para entrega de prácticas</b>
Facultad de ingeniería	Laboratorio de docencia

## Laboratorios de computación salas A y B

*Profesor:* Alejandro Pimentel

*Asignatura:* Fundamentos de Programación

*Grupo:* 3

*No de Práctica(s):* 10

*Integrante(s):* Arteaga Munguía Erick Alejandro

*No. de Equipo de  
cómputo empleado:* Rumania

*No. de Lista o Brigada:* 6294

*Semestre:* 2020-1

*Fecha de entrega:* 28/10/19

*Observaciones:* Tarde entrega.  
No se cumple con el objetivo de la primera actividad, y en hacen falta descripciones de los procedimientos para las otras.

**CALIFICACIÓN:** 5

## 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.

¿Que es gdb y para que funciona?

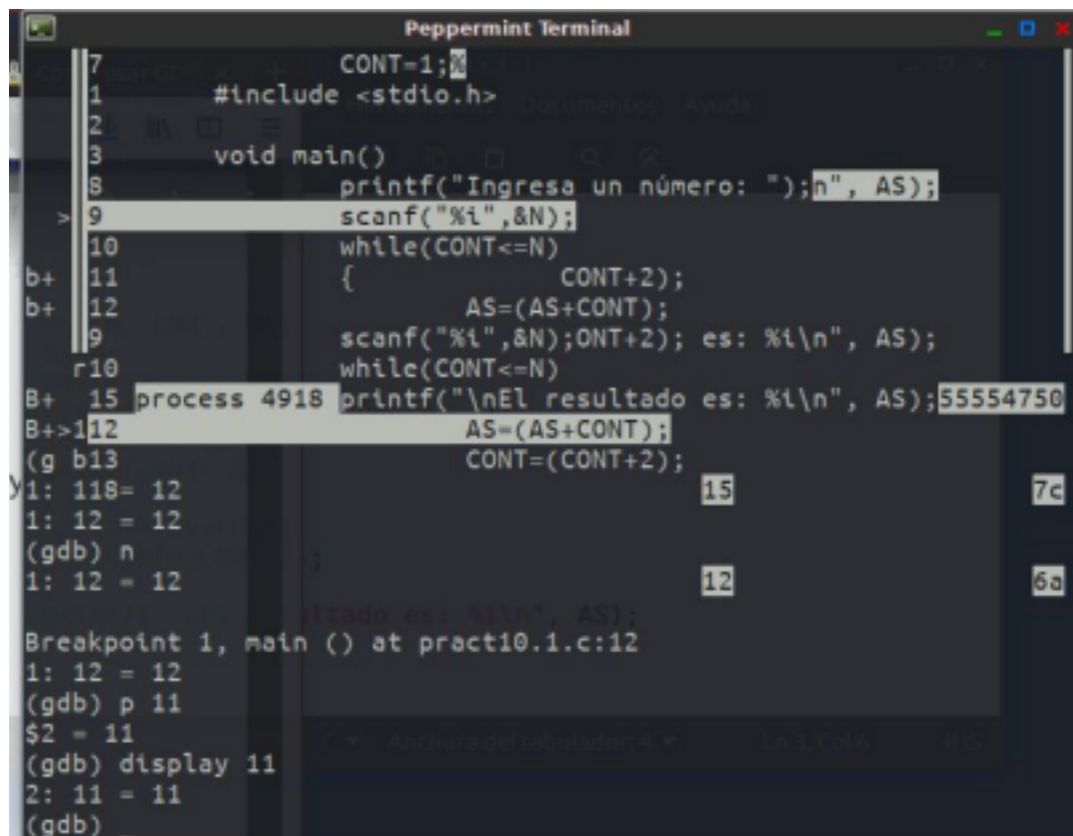
**GDB o GNU Debugger** es el depurador estándar para el compilador GNU.

Es un depurador portable que se puede utilizar en varias plataformas Unix y funciona para varios lenguajes de programación como C, C++ y Fortran. GDB fue escrito por Richard Stallman en 1986. GDB es software libre distribuido bajo la licencia GPL.

GDB ofrece la posibilidad de trazar y modificar la ejecución de un programa. El usuario puede controlar y alterar los valores de las variables internas del programa.

Al depurar un programa en C, nos referimos a analizarlo en programas dedicados a la depuración, los cuales nos brindan un ambiente controlado. Este nos permite visualizar con mayor detalle el proceso del programa. Con esto se busca encontrar cualquier error en las líneas de código u optimizar el mismo.

## ACTIVIDADES.



```
Peppermint Terminal
7      CONT=1;
1      #include <stdio.h>
2
3      void main()
8      {
9          printf("Ingresa un número: ");
10         scanf("%i",&N);
11         while(CONT<=N)
12         {
13             CONT=CONT+2;
14             AS=(AS+CONT);
15             scanf("%i",&N);
16             printf("es: %i\n", AS);
17         }
18         printf("\nEl resultado es: %i\n", AS);
19     }
20
21     process 4918
22     B+>112 AS=(AS+CONT);
23     (gdb) b 13
24     Breakpoint 1, main () at pract10.1.c:12
25     1: 12 = 12
26     (gdb) n
27     1: 12 = 12
28     Breakpoint 1, main () at pract10.1.c:12
29     1: 12 = 12
30     (gdb) p 11
31     $2 = 11
32     (gdb) display 11
33     2: 11 = 11
34     (gdb)
```

GDB es una herramienta muy importante ya que nos indica en dónde está el error, ya que da un salto de línea. Facilita identificar los errores de los programas.

En la primer actividad no había errores, tenían que describir el funcionamiento del programa con ayuda de GDB.

Se violó el código.

Se utiliza gdb para ver dónde está el error.

```
jorge@ubuntu-vm:~/ejemplos$ ./ejemplo1
GNU gdb (Ubuntu 8.1-0ubuntu3.1) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
Para las instrucciones de informe de errores, vea:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Leyendo símbolos desde ./ejemplo1...hecho.
(gdb) _
```

```
1 //ejemplo1.c
2 #include <math.h>
3
4 void main()
5 {
6     int K, AP, N;
7     double X, AS;
8     printf("Ingrese cuántos términos calcular de la serie: X^K/K!");
9     printf("\nN=");
10    //falta la & en los scanf
11    scanf("%i",&N);
12    printf("X=");
13    scanf("%lf",&X);
14    K=0;
15    AP=1;
16    AS=0;
17    //se cambió el k<=N por K<N para que el proceso se haga
18    //N veces
19    while(K<N)
20    {
21        AS=AS+pow(X,K)/AP;
22        K=K+1;
23        AP=AP*K;
24    }
25    printf("Resultado=%le",AS);
26 }
```

```

1  #include <stdio.h>
2
3  int main()
4  { //se agrego una variable porque al final se imprime numero pero
5    //el resultado final de numero es 0 asi que se agrego numero2
6    int numero,numero2;
7
8    printf("Ingrese un número:\n");
9    scanf("%i",&numero);
10   //numero2 es igual a numero para mantener su funcion
11   numero2=numero;
12   long int resultado = 1;
13   //se cambio el "numero2>=0" a "numero2>0" para evitar la multiplicacion por 0
14   while(numero2>0){
15       resultado *= numero2;
16       //se cambio la posicion de numero2— para que se restara hasta el final
17       //y no al principio del proceso iterativo
18       numero2--;
19   }
20
21   printf("El factorial de %i es %li.\n", numero, resultado);
22
23   return 0;
24 }

```

Conclusión:

La depuración es muy útil para poder encontrar errores cometidos en el código, ya que nos permite analizarlo con mayor detenimiento y así saber en dónde está la falla. Es importante poder compilar el programa sin errores antes de depurarlo.