



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: 135

No de Práctica(s): 10

Integrante(s): ALITZEL ANAID GUTIÉRREZ RAMOS
LAURA MUÑOZ REYES

*No. de Equipo de
cómputo empleado:* TONGA 51

No. de Lista o 9370
2823

Semestre: 2020 - 1

Fecha de entrega: 28/10/2019

Observaciones: Muy bien, pero recuerda poner el espacio para
la calificación

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.

La depuración de un programa es útil cuando:

- ✓ Se desea optimizar el programa: Debe realizarse un análisis exhaustivo en ejecución para averiguar cuál es su flujo de operación y encontrar formas de mejorarlo o encontrar puntos donde puede fallar con ciertos tipos de entrada de datos.
- ✓ El programa tiene algún fallo: El programa no muestra los resultados que se esperan para cierta entrada de datos debido a que el programador cometió algún error durante el proceso de diseño. Muchas veces encontrar este tipo de fallos suele ser difícil, ya sea porque la percepción del programador no permite encontrar la falla en su diseño.
- ✓ El programa tiene un error de ejecución o defecto: Cuando el programa está ejecutándose, éste se detiene inesperadamente. Suele ocurrir por un error en el

diseño o implementación del programa en las que no se contemplan las limitaciones del lenguaje de programación o el equipo donde el programa se ejecuta.

Para depurar un programa usando las herramientas desarrolladas por GNU, éste debe compilarse con información para depuración por medio del compilador GCC.

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



- ✓ *list* o *l*: Permite listar diez líneas del código fuente del programa, si se desea visualizar todo el código fuente debe invocarse varias veces este comando para mostrar de diez en diez líneas. Se puede optar por colocar un número separado por un espacio para indicar a partir de qué línea desea mostrarse el programa. También es posible mostrar un rango de líneas introduciendo el comando y de qué línea a qué línea separadas por una coma.
- ✓ *b*: Establece un punto de ruptura para lo cual debe indicarse en qué línea se desea establecer o bien también acepta el nombre de la función donde se desea realizar dicho paso.
- ✓ *do delete*: Elimina un punto de ruptura, indicando cuál es el que debe eliminarse usando el número de línea.

- ✓ *clear*: Elimina todos los puntos de ruptura.
- ✓ *info line*: Permite mostrar información relativa a la línea que se indique después del comando.
- ✓ *run* o *r*: Ejecuta el programa en cuestión. Si el programa tiene un punto de ruptura se ejecutará hasta dicho punto, de lo contrario se ejecutará todo el programa.
- ✓ *c*: Continúa con la ejecución del programa después de un punto de ruptura.
- ✓ *s*: Continúa con la siguiente instrucción después de un punto de ruptura.
- ✓ *n*: Salta hasta la siguiente línea de código después de un punto de ruptura.
- ✓ *p* o *print*: Muestra el valor de una variable, para ello debe escribirse el comando y el nombre de la variable separados por un espacio.
- ✓ *ignore*: Ignora un determinado punto de ruptura indicándolo con el número de línea de código.
- ✓ *q* o *quit*: Termina la ejecución de GDB.

EJEMPLO

```
#include <stdio.h>

int main(int argc, char * argv[]) {

    // Asignamos variables
    int numero = 10;
    int lista[numero];
    char caracter = 'B';
    float numeroReal = 89.8;
    long int suma = 0;
    double promedio;

    // Mostramos texto y valores
    printf("Primero texto solo\n");
    printf("Luego podemos poner un entero: %i\n", numero);
    printf("También podemos poner un caracter: %c\n", caracter);
    printf("Un numero real: %.2f\n", numeroReal);

    // Podemos llenar la lista con valores
    for(int i = numero ; i >= numero ; i++){
        lista[i] = i;
    }

    // Y ahora podemos hacer calculos con la lista
    for(int i = numero ; i >= numero ; i++){
        suma += lista[i];
    }
    promedio = suma / numero;
    printf("La suma es: %li\n", suma);
    printf("El promedio es: %lf\n", promedio);

    return 0;
}
```

gcc -g

```
Tonga51:Desktop fp03alu48$ gcc ejemplo1.c -o ejemplo1_
Tonga51:Desktop fp03alu48$ ./ejemplo1_
Primero texto solo
Luego podemos poner un entero: 10
También podemos poner un caracter: B
Un numero real: 89.80
Segmentation fault: 11
Tonga51:Desktop fp03alu48$ gcc -g ejemplo1.c -o ejemplo1
Tonga51:Desktop fp03alu48$ gdb ./ejemplo1_
GNU gdb (GDB) 8.2
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-apple-darwin18.2.0".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
```

```
Last login: Mon Oct 14 09:29:16 on ttys001
[Tonga51:~ fp03alu48$ servidor
[Tonga51:~ fp03alu48$ ssh fp03alu48@192.168.2.200
The authenticity of host '192.168.2.200 (192.168.2.200)' can't be established
RSA key fingerprint is SHA256:jTgFsbvP7IaIpwchV27DaUa9i2pvAVVZwZzbIneOF8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.200' (RSA) to the list of known hosts.
[fp03alu48@192.168.2.200's password:
```

Escritorio

```
-bash: aviso: setlocale: LC_CTYPE: no se puede cambiar el local (UTF-8)
[[fp03alu48@samba ~]$ ls
Escritorio
[[fp03alu48@samba ~]$ cd Escritorio
[[fp03alu48@samba Escritorio]$ ls
Proyector.desktop ejemplo1.c
[[fp03alu48@samba Escritorio]$ gcc -std=c99 -g ejemplo1.c -o ejemplo1
[[fp03alu48@samba Escritorio]$
```

gdb

```
[[fp03alu48@samba Escritorio]$ gcc -std=c99 -g ejemplo1.c -o ejemplo1
[[fp03alu48@samba Escritorio]$ gdb ./ejemplo1
GNU gdb (GDB) Fedora (7.4.50.20120120-42.fc17)
Copyright (C) 2012 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-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /users/fp03/fp03alu48/Escritorio/ejemplo1...done.
```

run

```
[(gdb) run
Starting program: /users/fp03/fp03alu48/Escritorio/ejemplo1
Primero texto solo
Luego podemos poner un entero: 10
También podemos poner un caracter: B
Un numero real: 89.80

Program received signal SIGSEGV, Segmentation fault.
0x0000000040060c in main (argc=19, argv=0x1100000010) at ejemplo1.c:21
21             lista[i] = i;
Missing separate debuginfos, use: debuginfo-install glibc-2.15-37.fc17.x86_64
```

list (l)

```
[(gdb) list
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++){
(gdb) █
```

quit (q)

```
Inferior 1 [process 21650] will be killed.

Quit anyway? (y or n) y
```

CTRL + X + A

```
[ No Source Available ]

exec No process In:                               Line: ??   PC: ??
(gdb) █
```

start

```
ejemplo1.c
1      #include <stdio.h>
2
3      int main(int argc, char * argv[]) {
4
5          // Asignamos variables
6      int numero = 10;
7      int lista[numero];
8      char character = 'B';
9      float numeroReal = 89.8;
10     long int suma = 0;
11     double promedio;
12
13     // Mostramos texto y valores

child process 22198 In: main                        Line: 6   PC: 0x400542
(gdb) start
Temporary breakpoint 1 at 0x400542: file ejemplo1.c, line 6.
Starting program: /users/fp03/fp03alu48/Escritorio/ejemplo1

Temporary breakpoint 1, main (argc=1, argv=0x7fffffff378) at ejemplo1.c:6
Missing separate debuginfos, use: debuginfo-install glibc-2.15-37.fc17.x86_64
(gdb) █
```


next (n)

```
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
9  11      // 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 22198 In: main                                Line: 20    PC: 0x40060f
(gdb) start
Temporary breakpoint 1 at 0x400542: file ejemplo1.c, line 6.
Starting program: /users/fp03/fp03alu48/Escritorio/ejemplo1

Temporary breakpoint 1, main (argc=1, argv=0x7fffffff378) at ejemplo1.c:6
Missing separate debuginfos, use: debuginfo-install glibc-2.15-37.fc17.x86_64
(gdb) █
```

print (p)

```
12
13      // Mostramos texto y valores
3  14      printf("Primero texto solo\n");
15      printf("Luego podemos poner un entero: %i\n", numero);
5  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      }
28      promedio = suma / numero;
29      printf("La suma es: %li\n", suma);
19 30      printf("El promedio es: %lf\n", promedio);
Un numero real: 89.80 un caracter: B
child process 22945 In: main                                Line: 21    PC: 0x4005ff
The program is not being run.
Starting program: /users/fp03/fp03alu48/Escritorio/ejemplo1

Temporary breakpoint 1, main (argc=1, argv=0x7fffffff378) at ejemplo1.c:6
Missing separate debuginfos, use: debuginfo-install glibc-2.15-37.fc17.x86_64
(gdb) p i
$1 = 10 i
(gdb) print lista
print lista
$2 = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
(gdb) █
```

display

```
3 14      printf("Primero texto solo\n");
15      printf("Luego podemos poner un entero: %i\n", numero);
5 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
>      // 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;
29      printf("La suma es: %li\n", suma);
19 30      printf("El promedio es: %lf\n", promedio);
Un numero real: 89.80 un caracter: B
child process 22945 In: main                               Line: 21   PC: 0x4005ff
The program is not being run.
Starting program: /users/fp03/fp03alu48/Escritorio/ejemplo1
Missing separate debuginfos, use: debuginfo-install glibc-2.15-37.fc17.x86_64
(gdb) print lista
print lista
$2 = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
(gdb) display i
display i
1: i = 10
(gdb) display lista
display lista
2: lista = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
(gdb) █
```

```
> 20      for(int i = numero ; i >= numero ; i++){
> 21          lista[i] = i;
> 22      }
> 23
>      // Y ahora podemos hacer calculos con la lista
25      for(int i = numero ; i >= numero ; i++){
26          suma += lista[i];
> 20      for(int i = numero ; i >= numero ; i++){
20      for(int i = numero ; i >= numero ; i++){
> 20      for(int i = numero ; i >= numero ; i++){
21          lista[i] = i;s: %lf\n", promedio);
Un numero real: 89.80 un caracter: B
child process 22945 In: main                               Line: 21   PC: 0x4005ff
The program is not being run.
Starting program: /users/fp03/fp03alu48/Escritorio/ejemplo1
Missing separate debuginfos, use: debuginfo-install glibc-2.15-37.fc17.x86_64
(gdb) print lista
print lista
$2 = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
(gdb) display i
display lista
(gdb)
1: i = 10
(gdb) n
n
2: lista = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
1: i = 11
(gdb) n
n
2: lista = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
1: i = 11
(gdb) █
```

| | |
|---|----|
| 0 | 60 |
| 1 | 5f |
| 0 | 50 |

```

20         for(int i = numero ; i >= numero ; i++){
> 20         for(int i = numero ; i >= numero ; i++){
20         for(int i = numero ; i >= numero ; i++){;
>n20         for(int i = numero ; i >= numero ; i++){
120         for(int i = numero ; i >= numero ; i++){ Line: 21 PC: 0x4005ff
> 20         for(int i = numero ; i >= numero ; i++){
r21         lista[i] = i; escritorio/ejemplo1
Missing separate debuginfos, use: debuginfo-install glibc-2.15-37.fc17.x86_64
(gdb) print lista
print lista
$2 = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
(gdb) display i
display lista
(gdb)
1: i = 10
1: i = 11
(gdb) n
2: lista = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
(gdb) n
2: lista = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
1: i = 12
(gdb) n
2: lista = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
1: i = 13
(gdb) n
2: lista = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
1: i = 13
(gdb)

```

```

0
1
0
1
0
1
0
0
60
5f
60
5f
60
5f
60

```

```

A
O
[
^

```

break (b)

```

[(gdb) break 20
Breakpoint 1 at 0x4005f7: file ejemplo1.c, line 20.
[(gdb) run
Starting program: /users/fp03/fp03alu48/Escritorio/ejemplo1
Primero texto solo
Luego podemos poner un entero: 10
También podemos poner un caracter: B
Un numero real: 89.80

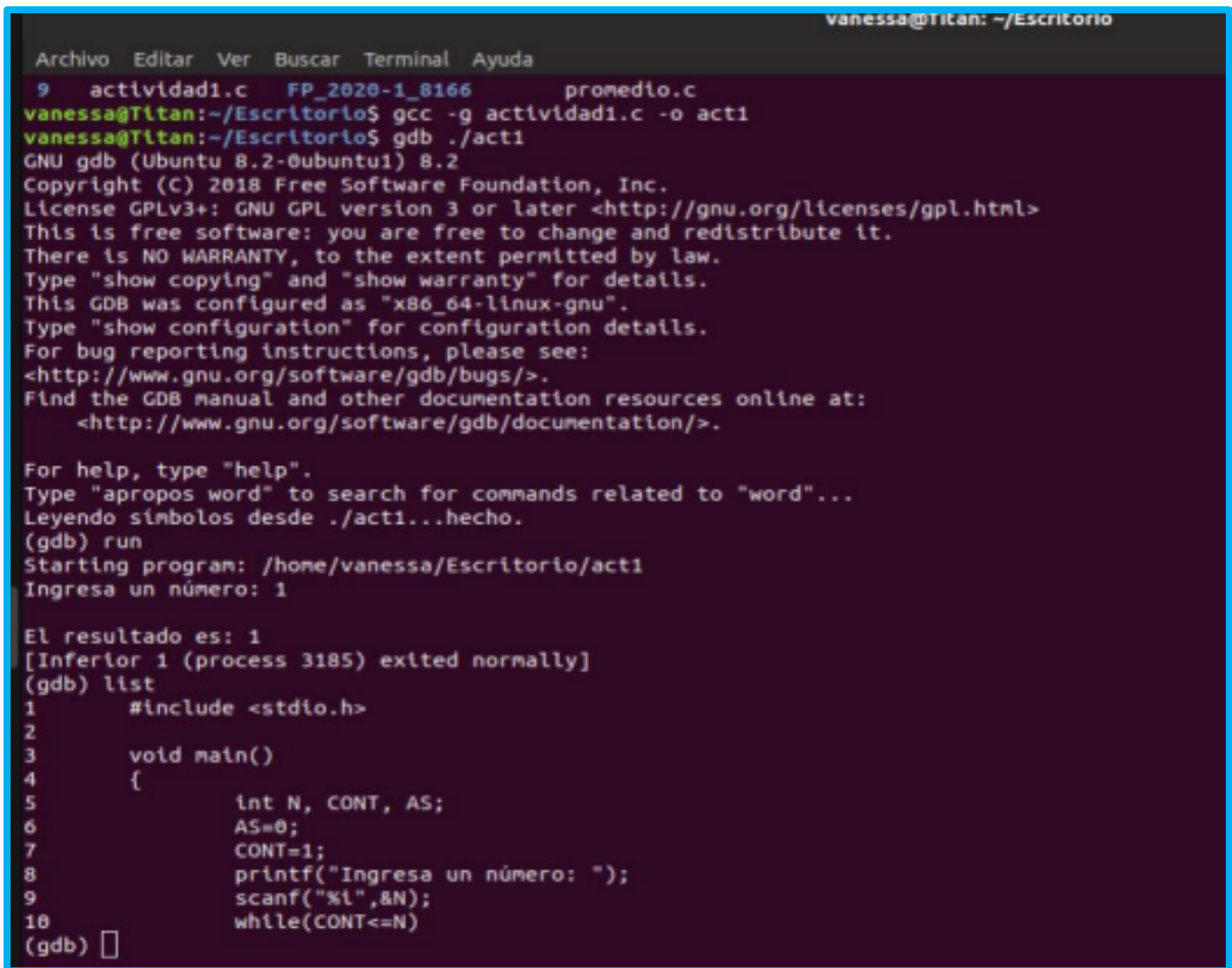
Breakpoint 1, main (argc=1, argv=0x7fffffff378) at ejemplo1.c:20
20         for(int i = numero ; i >= numero ; i++){
Missing separate debuginfos, use: debuginfo-install glibc-2.15-37.fc17.x86_64
(gdb)

```

Actividad 1

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

Para empezar a ver cómo funcionaba, lo compilamos de manera que pudiéramos utilizar gdb, como lo hicimos anteriormente y así ver con detalle su ejecución.



```
vanessa@Titan: ~/Escritorio
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
9  actividad1.c  FP_2020-1_8166  promedio.c
vanessa@Titan:~/Escritorio$ gcc -g actividad1.c -o act1
vanessa@Titan:~/Escritorio$ gdb ./act1
GNU gdb (Ubuntu 8.2-0ubuntu1) 8.2
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.
For bug reporting instructions, please see:
<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 ./act1...hecho.
(gdb) run
Starting program: /home/vanessa/Escritorio/act1
Ingresa un número: 1

El resultado es: 1
[Inferior 1 (process 3185) exited normally]
(gdb) list
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)
(gdb) □
```

Y después corrimos el programa igual con GDB para poder analizar con mayor precisión.

```
vanessa@Titan: ~/Escritorio
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

actividad1.c
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+2);
14         }
15         printf("\nEl resultado es: %i\n", AS);
16     }
```

```
native process 3204 In: main          L9      PC: 0x5555555518b
(gdb) start
Punto de interrupción temporal 1 at 0x5555555515d: file actividad1.c, line 4.
Starting program: /home/vanessa/Escritorio/act1

Temporary breakpoint 1, main () at actividad1.c:4
(gdb) n
Ingresa un número: 3
```

Después de que analizamos con gdb, al ir saltando renglón por renglón, se ve que la funcionalidad del programa es recibir un 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.

Esto se repite por un ciclo que tiene como condición que, *contador* aumentará 2 unidades siempre y cuando sea **menor o igual** al valor que se introduce al inicio, por tanto, hasta que *contador* alcance un valor que no sobrepase o sea igual al número que se introduce, seguirá haciendo los acumulados de la suma tanto en *AS* como en *contador*. Cuando lo alcance, entonces ahí se detiene el ciclo y muestra el resultado total que se fue acumulando en *AS*.

Este fue el resultado después de correrlo en la terminal.


```
dades Terminal ▾ mar, 22 de oct, 00:18 ●
vanessa@Titan: ~/Escritorio

Archivo Editar Ver Buscar Terminal Ayuda
vanessa@Titan:~$ cd Escritorio
vanessa@Titan:~/Escritorio$ ls
1 a2 actividad1.c FP_2020-1_8166 promedio.c
2 a3 actividad2 main 'tips para el ensayo.pptx'
3 ac2 actividad2.c primo
8 act1 actividad3.c primo.c
9 act3 ensayorúbrica.docx promedio
vanessa@Titan:~/Escritorio$ gcc actividad1.c -o ac1
vanessa@Titan:~/Escritorio$ ./ac1
Ingresa un número: 4

El resultado es: 4
vanessa@Titan:~/Escritorio$ ./ac1
Ingresa un número: 5

El resultado es: 9
vanessa@Titan:~/Escritorio$ ./ac1
Ingresa un número: 6

El resultado es: 9
vanessa@Titan:~/Escritorio$ ./ac1
Ingresa un número: 7

El resultado es: 16
vanessa@Titan:~/Escritorio$ ./ac1
Ingresa un número: 2

El resultado es: 1
vanessa@Titan:~/Escritorio$ ./ac1
Ingresa un número: 9

El resultado es: 25
vanessa@Titan:~/Escritorio$
```

Actividad 2

- ✓ Utilizar GDB para corregir el programa.
- ✓ Nota: Para compilar el código de la actividad, ejecutar:

```
$ gcc -w actividad2.c -o actividad2 -lm
```

En esta actividad vamos a seguir los mismos pasos que en la primera, solo que con diferente compilación ya que con la primera opción, este programa no se puede compilar.

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
1876      ln vfscanf.c
vanessa@Titan:~/Escritorio$ gcc -g -w actividad2.c -o a2 -lm
vanessa@Titan:~/Escritorio$ gdb ./a2
GNU gdb (Ubuntu 8.2-0ubuntu1) 8.2
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.
For bug reporting instructions, please see:
<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 ./a2...hecho.
(gdb) run
Starting program: /home/vanessa/Escritorio/a2
Ingrese cuántos términos calcular de la serie: X^K/K!
N=4
X=5
Resultado=6.537500e+01[Inferior 1 (process 2427) exited normally]
(gdb) list
1      #include <stdio.h>
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         scanf("%i",&N);
(gdb) □
```

Y cuando lo corregimos y lo corrimos en la terminal, estos fueron nuestros resultados:

```

numero.c  tablas.c  promedio.c  actividad2.c
1  #include <stdio.h>
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     scanf("%i",&N);
11     printf("X=");
12     scanf("%lf",&X);
13     K=0;
14     AP=1;
15     AS=0;
16     while(K<=N)
17     {
18         AS=AS+pow(X,K)/AP;
19         K=K+1;
20         AP=AP*K;
21     }
22     printf("Resultado=%le\n",AS);
23 }
24

```

```

Jades  Terminal
vanes

Archivo  Editar  Ver  Buscar  Terminal  Ayuda
vanessa@Titan:~$ cd Escritorio
vanessa@Titan:~/Escritorio$ gcc -w actividad2.c -o acti2 -lm
vanessa@Titan:~/Escritorio$ ./acti2
Ingrese cuántos términos calcular de la serie: X^K/K!
N=3
X=4
Resultado=2.366667e+01
vanessa@Titan:~/Escritorio$ ./acti2
Ingrese cuántos términos calcular de la serie: X^K/K!
N=32
X=25
Resultado=-2.224746e+35
vanessa@Titan:~/Escritorio$ ./acti2
Ingrese cuántos términos calcular de la serie: X^K/K!
N=-8
X=-9
Resultado=0.000000e+00
vanessa@Titan:~/Escritorio$

```


Actividad 3

- ✓ Utilizar GDB para corregir el programa.

```
ades  Terminal  jue, 17 de oct, 15:32 ●
vanessa@Titan: ~/Escritorio

Archivo  Editar  Ver  Buscar  Terminal  Ayuda

vanessa@Titan:~/Escritorio$ gcc actividad3.c -o a3
vanessa@Titan:~/Escritorio$ ./a3
Ingrese un número:
8
El factorial de -1 es 0.
vanessa@Titan:~/Escritorio$ gcc -g actividad3.c -o act3
vanessa@Titan:~/Escritorio$ gdb ./act3
GNU gdb (Ubuntu 8.2-0ubuntu1) 8.2
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.
For bug reporting instructions, please see:
<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 ./act3...hecho.
(gdb) run
Starting program: /home/vanessa/Escritorio/act3
Ingrese un número:
8
El factorial de -1 es 0.
[Inferior 1 (process 2237) exited normally]
(gdb) list
1      #include <stdio.h>
2
3      int main()
4      {
5          int numero;
6
7          printf("Ingrese un número:\n");
8          scanf("%d", &numero);
```

Archivo Editar Ver Buscar Terminal Ayuda

```
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;
```

vanessa@Titan:~/Escritorio\$ gdb ./act3

GNU gdb (Ubuntu 8.2-0ubuntu1) 8.2

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.

For bug reporting instructions, please see:

<<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 ./act3...hecho.

(gdb) break 11

Punto de interrupción 1 at 0x11a8: file actividad3.c, line 11.

(gdb) run

Starting program: /home/vanessa/Escritorio/act3

Ingrese un número:

6

Breakpoint 1, main () at actividad3.c:11

```
11     while(numero>=0){
```

(gdb)

```
actividad3.c
B+> 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 %li.\n", numero, resultado);
17
18    return 0;
19 }
20
21
22
23
24
25

native process 2261 In: main
(gdb) start
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Punto de interrupción temporal 2 at 0x5555555516d: file actividad3.c, line 4.
Starting program: /home/vanessa/Escritorio/act3

Temporary breakpoint 2, main () at actividad3.c:4
(gdb) print numero
$1 = 21845
(gdb) display numero
1: numero = 21845
(gdb) 
```

Después de corregir algunos errores, estos fueron nuestros resultados:

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
vanessa@Titan:~$ cd Escritorio
vanessa@Titan:~/Escritorio$ gcc actividad3.c -o a3
vanessa@Titan:~/Escritorio$ ./a3
Ingrese un número:
3
El factorial de 272380032 es 6
vanessa@Titan:~/Escritorio$ gcc actividad3.c -o ac3
vanessa@Titan:~/Escritorio$ ./ac3
Ingrese un número:
3
El factorial de 3 es 6
vanessa@Titan:~/Escritorio$ ./ac3
Ingrese un número:
5
El factorial de 5 es 120
vanessa@Titan:~/Escritorio$ ./ac3
Ingrese un número:
26
El factorial de 26 es -1569523520172457984
vanessa@Titan:~/Escritorio$ ./ac3
Ingrese un número:
16
El factorial de 16 es 20922789888000
vanessa@Titan:~/Escritorio$ 
```

Conclusión

En conclusión, siempre hay que tener en cuenta que los ordenadores hacen sólo aquello que nosotros les decimos que hagan: si los programas no funcionan es porque no los hemos diseñado bien o porque hemos cometido algún error, y no porque el compilador o el ordenador no funcionen correctamente.

Debemos tener siempre esto en mente, y una vez que se nos ha pasado el mal humor después de analizar y pensar durante mucho tiempo sobre nuestros algoritmos y programas, sin obtener resultados, debemos reflexionar y volver al problema.

Los programas no pueden ser depurados siempre, antes deben cumplir determinadas condiciones. La primera es que se pueda obtener un ejecutable. Si no llegamos a eso los errores serán de compilación o de enlazado, y un depurador no tiene sentido hasta que se solucionen todos esos problemas.