

# Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

Profesor: Alejandro Esteban Pimentel Alarcon

Asignatura: Fundamentos de Programación

*Grupo:* 3

No. de practica: 10

Integrantes:

Jonan Gómez Mendoza 5641 Franco Inglés Carolina 2836

Lucia Nicole Rosette Hernández 2768

Semestre: 1

Fecha de entrega: octubre 28, 2019

Observaciones: La práctica está incompleta, tanto en la

actividad 2 como en la actividad 3, no muestran que el programa haya sido ejecutado con éxito. No les penalizaré mucho por los comentarios

que pusieron.

CALIFICACIÓN:

## PRÁCTICA #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:

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.

Al intentar ejecutar un programa sale el letrero violación de segmento

```
jon@jon-Parallels-Virtual-Platform ~ $ cd Descargas/
jon@jon-Parallels-Virtual-Platform ~/Descargas $ gcc ejemplo1.c -o main
jon@jon-Parallels-Virtual-Platform ~/Descargas $ ./main
Primero texto solo
.uego podemos poner un entero: 10
También podemos poner un caracter: B
Jn numero real: 89.80
/iolación de segmento
```

Se inicia el depurador gdb para poder ver el error

```
jon@jon-Parallels-Virtual-Platform ~/Descargas $ gdb ./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 <a href="http://gnu.org/licenses/gpl.html">http://gnu.org/licenses/gpl.html</a>
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:
<a href="http://www.gnu.org/software/gdb/bugs/">http://www.gnu.org/software/gdb/bugs/>.</a>
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)
```

Se inicia con run

```
(gdb) run
Starting program: /home/jon/Descargas/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.
0x0000555555554806 in main (argc=19, argv=0x1100000010) at ejemplo1.c:21
21 lista[i] = i;
(gdb)
```

Con el comando list se muestran las lineas del codigo

```
(gdb) list
                   printf("También podemos poner un caracter: %c\n", caracter); 
 <math>printf("Un numero real: %.2f\n", numeroReal);
16
17
18
19
                   // Podemos llenar la lista con valores
26
                   for(int i = numero ; i >= numero ; i++){
21
                             lista[i] = i;
23
24
                    // Y ahora podemos hacer calculos con la lista
25
                    for(int i = numero ; i >= numero ; i++){
(ddb)
```

Con el comando quit te sales del depurador

```
(gdb) quit
Una sesión de depuración está activa.
Inferior 1 [process 3688] will be killed.
¿Salir de cualquier modo? (y or n) y
```

Con ctrl+x+a se abre una interfaz gráfica

```
[ No Source Available ]

exec No process In:

L?? PC: ??
```

Con el comando start se empieza a correr el programa linea por linea

```
ejemplo1.c
              #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 = θ;
double promedio;
                       printf("Primero texto solo\n");
printf("Luego podemos poner un entero: %i\n", numero);
native process 3770 In: main
                                                                         L3
                                                                                PC: 0x55555555470a
(gdb) start
Punto de interrupci∎n temporal 1 at 0x70a: file ejemplo1.c, line 3.
Starting program: /home/jon/Descargas/ejemplo1
Temporary breakpoint 1, main (argc=1, argv=0x7fffffffe7b8) at ejemplo1.c:3
(gdb)
```

Para pasar a la siguiente línea se usa el comando n

```
ejemplo1.c
             #include <stdio.h>
             int main(int argc, char * argv[]) {
                      // Asignamos variables
                      int numero = 10;
                      int huncro
int lista[numero];
int lista[numero];
                      char caracter = 'B';
float numeroReal = 89.8;
                      long int suma = 0;
                      double promedio;
    11
                      // Mostramos texto y valores
     14
                      printf("Luego podemos poner un entero: %i\n",
                                                                           PC: 0x55555555478d
                                                                    19
native process 3770 In: main
(gdb) start
Punto de interrupcian temporal 1 at 0x70a: file ejemplo1.c, line 3.
Starting program: /home/jon/Descargas/ejemplo1
Temporary breakpoint 1, main (argc=1, argv=0x7fffffffe7b8) at ejemplo1.c:3
(gdb) n
(gdb) n
(adb)
```

Con el comando break podemos poner un break en la línea especificado en este ejemplo en la línea 20

```
Program terminated with signal SIGSEGV, Segmentation fault.
The program no longer exists.
Este programa no est corriendo.
(gdb) break 20
Punto de interrupcion 2 at 0x5555555547f1: file ejemplo1.c, line 20.
(gdb)
```

Con el comando p podemos imprimir una variable en específico en este caso i

```
ejemplo1.c
                          long int suma = 0;
                         double promedio;
     11
     12
13
                          // 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);
     14
15
     18
     19
                          // Podemos llenar la lista con valores
     20
                          for(int i = numero ; i >= numero ; i++){
     21
                                    lista[i] = i;
     10
     23
     24
                          // Y ahora podemos hacer calculos con la lista
native process 4464 In: main
                                                                                      PC: 0x555555547f9
Punto de interrupciun temporal 1 at 0x70a: file ejemplo1.c, line 3.
Starting program: /home/jon/Descargas/ejemplo1
Temporary breakpoint 1, main (argc=1, argv=0x7fffffffe7b8) at ejemplo1.c:3
(gdb) n
(gdb) p i
(gdb)
```

```
ejemplo1.c
                        long int suma = \theta;
     11
12
13
14
15
16
17
                        double promedio;
                        // Mostramos texto y valores
                       printf("Primero texto solo\n");
                       printf("Luego podemos poner un entero: %1\n", numero);
                       printf("También podemos poner un caracter: %c\n", caracter);
printf("Un numero real: %.2f\n", numeroReal);
     19
                        // Podemos llenar la lista con valores
     20
                        for(int i = numero ; i >= numero ; i++){
                                 lista[1] = 1:
     21
     23
                        // Y ahora podemos hacer calculos con la lista
                                                                       L21
native process 4464 In: main
                                                                              PC: 0x555555547f9
Temporary breakpoint 1, main (argc=1, argv=0x7fffffffe7b8) at ejemplo1.c:3
(gdb) n
(gdb) p i
(gdb) print lista
$2 = {-1, θ, θ, θ, -134241688, 32767, -134224112, 32767, θ, θ}
 (adb)
Se puede dar seguimiento a las variables con el comando display así se visualiza siempre
       ejemplo1.c
```

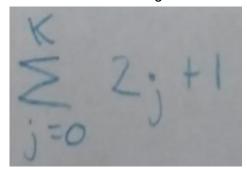
```
long int suma = \theta;
                         double promedio;
     12
13
14
                         // Mostramos texto y valores
                         printf("Primero texto solo\n");
printf("Luego podemos poner un entero: %i\n", numero);
printf("También podemos poner un caracter: %c\n", caracter);
                         printf("Un numero real: %.2f\n", numeroReal);
     18
                         // Podemos llenar la lista con valores
                         for(int i = numero ; i >= numero ; i++){
    lista[i] = i;
     2θ
     23
                         // Y ahora podemos hacer calculos con la lista
native process 4464 In: main
                                                                             L21
                                                                                     PC: 0x555555547f9
$1 = 10
(gdb) print lista
$2 = {-1, θ, θ, θ, -134241688, 32767, -134224112, 32767, θ, θ}
(gdb) display i
(gdb) display lista
2: lista = {-1, 0, 0, 0, -134241688, 32767, -134224112, 32767, 0, 0}
```

### **ACTIVIDAD 1**

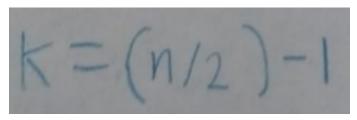
Encontrar la funcionalidad del siguiente programa

```
#include <stdio.h>
2
   void main()
        int N, CONT, AS;
        AS=0;
        CONT=1:
          intf("Ingresa un número: ");
             ("%i",&N);
        while(CONT<=N)
11 -
            AS=(AS+CONT);
12
            CONT=(CONT+2);
13
14
        printf("\nEl resultado es: %i\n", AS);
15
16
```

El resultado es de la siguiente sumatoria



Donde así se calcula k con la n que metemos en el programa



Se debe de redondear la división de n/2 al siguiente número por ejemplo 5/2=2.5 se redondea a 3 o 13/2=6.5 se redondea a 7 actividad 2

```
#include <stdio.h>
#include <math.h>
void main()
{
    int K, AP, N;
    double X, AS;
    printf("Ingrese cuántos términos calcular de la serie: X^K/K!");
    printf("\nN=");
    scanf("%i",&N);
    printf("X=");
    scanf("%lf",&X);
    K=0;
    AP=1;
    //se cambio el k<=N por K<N para que el proceso se haga
    while(K<N)
    {
        AS=AS+pow(X,K)/AP;
        K=K+1;
        AP=AP*K;
    printf("Resultado=%le",AS);
```

ACTIVIDAD 3

```
#include <stdio.h>

int main()

//se agrego una variable porque al final se imprime numero pero
//el resultado final de numero es 0 asi que se agrego numero2
int numero,numero2;

printf("Ingrese un número:\n");
scanf("%i",&numero);
//numero2 es igual a numero para mantener su funcion
numero2=numero;
long int resultado = 1;
//se cambio el "numero2>=0" a "numero2>0" para evitar la multiplicacion por 0
while(numero2>0){
    resultado *= numero2;
    //se cambio la posicion de numero2— para que se restara hasta el final
    //y no al principio del proceso iterativo
    numero2—;
}

printf("El factorial de %i es %li.\n", numero, resultado);
return 0;
```

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.