



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:

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  
Luego podemos poner un entero: 10  
También podemos poner un caracter: B  
Un numero real: 89.80  
Violación 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 <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) _
```

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.
0x00005555555554806 in main (argc=19, argv=0x1100000010) at ejemplo1.c:21
21             lista[i] = i;
(gdb) _
```

Con el comando list se muestran las líneas del código

```
(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) _
```

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: ??
(gdb) _
```

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

```
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 caracter = 'B';
9          float numeroReal = 89.8;
10         long int suma = 0;
11         double promedio;
12
13         // Mostramos texto y valores
14         printf("Primero texto solo\n");
15         printf("Luego podemos poner un entero: %i\n", numero);

native process 3770 In: main                        L3    PC: 0x55555555470a
(gdb) start
Punto de interrupci3n 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 lnea se usa el comando n

```
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 caracter = 'B';
9      > float numeroReal = 89.8;
10         long int suma = 0;
11         double promedio;
12
13         // Mostramos texto y valores
14         printf("Primero texto solo\n");
15         printf("Luego podemos poner un entero: %i\n", numero);
16
17         // Podemos llenar la lista con valores
18         for(int i = numero ; i >= numero ; i++){
19             lista[i] = i;
20         }
21
22         // Y ahora podemos hacer calculos con la lista
23
24     }
```

native process 3770 In: main L9 PC: 0x55555555478d
(gdb) start
Punto de interrupción temporal 1 at 0x70a: file ejemplo1.c, line 3.
Starting program: /home/jon/Descargas/ejemplo1
Temporary breakpoint 1, main (argc=1, argv=0x7fffffff7b8) at ejemplo1.c:3
(gdb) n
(gdb) n
(gdb)

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 está corriendo.  
(gdb) break 20  
Punto de interrupción 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
10         long int suma = 0;
11         double promedio;
12
13         // Mostramos texto y valores
14         printf("Primero texto solo\n");
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     }
```

native process 4464 In: main L21 PC: 0x5555555547f9
Punto de interrupción temporal 1 at 0x70a: file ejemplo1.c, line 3.
Starting program: /home/jon/Descargas/ejemplo1
Temporary breakpoint 1, main (argc=1, argv=0x7fffffff7b8) at ejemplo1.c:3
(gdb) n
(gdb) p i
\$1 = 10
(gdb) _


```
ejemplo1.c
10      long int suma = 0;
11      double promedio;
12
13      // Mostramos texto y valores
14      printf("Primero texto solo\n");
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

native process 4464 In: main L21 PC: 0x5555555547f9

Temporary breakpoint 1, main (argc=1, argv=0x7fffffff7b8) at ejemplo1.c:3
(gdb) n
(gdb) p i
$1 = 10
(gdb) print lista
$2 = {-1, 0, 0, 0, -134241688, 32767, -134224112, 32767, 0, 0}
(gdb)
```

Se puede dar seguimiento a las variables con el comando display así se visualiza siempre

```
ejemplo1.c
10      long int suma = 0;
11      double promedio;
12
13      // Mostramos texto y valores
14      printf("Primero texto solo\n");
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

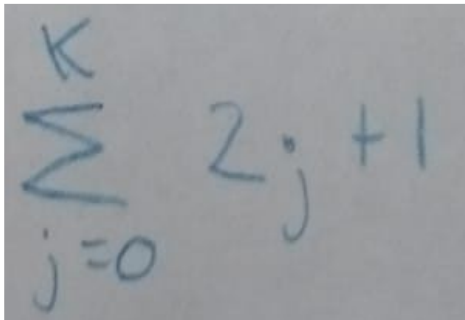
native process 4464 In: main L21 PC: 0x5555555547f9
$1 = 10
(gdb) print lista
$2 = {-1, 0, 0, 0, -134241688, 32767, -134224112, 32767, 0, 0}
(gdb) display i
1: i = 10
(gdb) display lista
2: lista = {-1, 0, 0, 0, -134241688, 32767, -134224112, 32767, 0, 0}
(gdb)
```

ACTIVIDAD 1

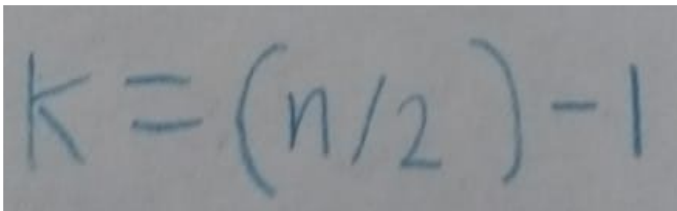
Encontrar la funcionalidad del siguiente programa

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

El resultado es de la siguiente sumatoria


$$\sum_{j=0}^k 2j+1$$

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


$$k = (n/2) - 1$$

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

```

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     //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 cambio 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 }

```

ACTIVIDAD

3

```

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 así 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.