



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 Práctica(s): 10

Integrante(s): Crail Ávila Regina
Ortiz Garcia Cesar Alan

*No. de Equipo de cómputo
empleado:* 20,21

No. de Lista o Brigada: 8973
9070

Semestre: 20-21

Fecha de entrega: 27/oct/19

Observaciones: Sus programas están bien corregidos.
Lamentablemente, los objetivos de la práctica son
utilizar depuradores, y no muestran ninguna
evidencia de su uso.

CALIFICACIÓN: **6**

Primeros pasos con GDB

jigrodriguez@gmail.com

Este documento pretende ser una breve y sencilla guía práctica al depurador GDB (<http://gnu.org>). A lo largo de esta guía, usaremos como ejemplo un sencillo programa cuyo código se muestra a continuación:

```
#include <stdio.h>
int main() {
    int v = 0;
    int i;
    for (i = 0; i < 5; i++) {
        v += i;
    }
    printf("Resultado: %i\n", v);
    return 0;
}
```

Lo primero que hemos de hacer es compilar dicho programa con la opción "-g" para que se incluya la información necesaria para la depuración. Si el programa está guardado en un archivo

```
$ gcc -g -o test test.c
```

Una vez compilado, podemos comenzar la depuración invocando a gdb con el nombre del archivo compilado.

```
$ gdb test
```

```
GNU gdb 5.3-debian
```

```
Copyright 2002 Free Software Foundation, Inc.
```

```
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
```

```
Type "show copying" to see the conditions.
```

```
There is absolutely no warranty for GDB. Type "show warranty" for details.
```

```
This GDB was configured as "i386-linux"...
```

```
(gdb)
```

La línea "(gdb)" nos indica que el depurador está listo para empezar a procesar nuestras órdenes. El primer comando que veremos es (l)ist. Este comando muestra el código fuente del programa que estoy depurando.

```
(gdb) l
```

```
1 #include <stdio.h>
2
3 int main() {
4
5 int v = 0;
6 int i;
7
8 for (i = 0; i < 5; i++) {
9 v += i;
10 }
```

```
(gdb)
```

Para ejecutar el programa, utilizamos el comando (r)un. Como aún no hemos puesto ningún punto de ruptura, el programa se ejecutará entero.

```
(gdb) r
```

```
Starting program: /ramdisk/home/knoppix/test
```

```
Resultado: 10
```

```
Program exited normally.
```

```
(gdb)
```

Vamos a colocar ahora un punto de ruptura. Como su nombre indica, un punto de ruptura es un punto donde la ejecución de nuestro programa se detiene. Para colocar un punto de ruptura utilizamos el comando (b)reakpoint seguido de la línea donde queremos ponerlo.

(gdb) b 9

Breakpoint 1 at 0x804834e: file test.c, line 9.

(gdb)

Ya tenemos un punto de ruptura en la línea 9. Si ahora ejecutamos el programa se detendrá cuando llegue a esa línea.

(gdb) r

Starting program: /ramdisk/home/knoppix/test

Breakpoint 1, main () at test.c:9

9 v += i;

(gdb)

Para continuar la ejecución del programa utilizamos el comando (c)ontinue. En nuestro ejemplo, el programa se volverá a detener en el mismo punto de ruptura.

(gdb) c

Continuing.

Breakpoint 1, main () at test.c:9

9 v += i;

(gdb) c

Continuing.

Breakpoint 1, main () at test.c:9

9 v += i;

(gdb) c

Continuing.

Breakpoint 1, main () at test.c:9

9 v += i;

(gdb) c

Continuing.

Breakpoint 1, main () at test.c:9

9 v += i;

(gdb) c

Continuing.

Resultado: 10

Program exited normally.

(gdb)

Puesto que tenemos un punto de ruptura en el bucle y este da 5 vueltas, hemos tenido que continuar 5 veces para ejecutar el programa completo. También es posible ejecutar el programa línea a línea con el comando (n)ext.

(gdb) r

Starting program: /ramdisk/home/knoppix/test

Breakpoint 2, main () at test.c:9

9 v += i;

(gdb) n

8 for (i = 0; i < 5; i++) {

(gdb) n

Breakpoint 2, main () at test.c:9

9 v += i;

(gdb)

En cualquier momento podemos ver el valor de una variable con el comando (p)rint seguido del nombre de la variable.

(gdb) p v

\$1 = 1

(gdb) c

Continuing.

Breakpoint 2, main () at test.c:9

9 v += i;

(gdb) c

Continuing.

Breakpoint 2, main () at test.c:9

9 v += i;

(gdb) p v

\$2 = 6

(gdb)

Para eliminar un punto de ruptura utilizamos el comando delete y el número del punto a eliminar. En nuestro ejemplo

(gdb) delete 1

(gdb)

Por último, para salir del depurador utilizamos el comando (q)uit.

Hay mucha más información disponible tanto en la página del manual como en los archivos info:

\$ man gdb

\$ info gdb

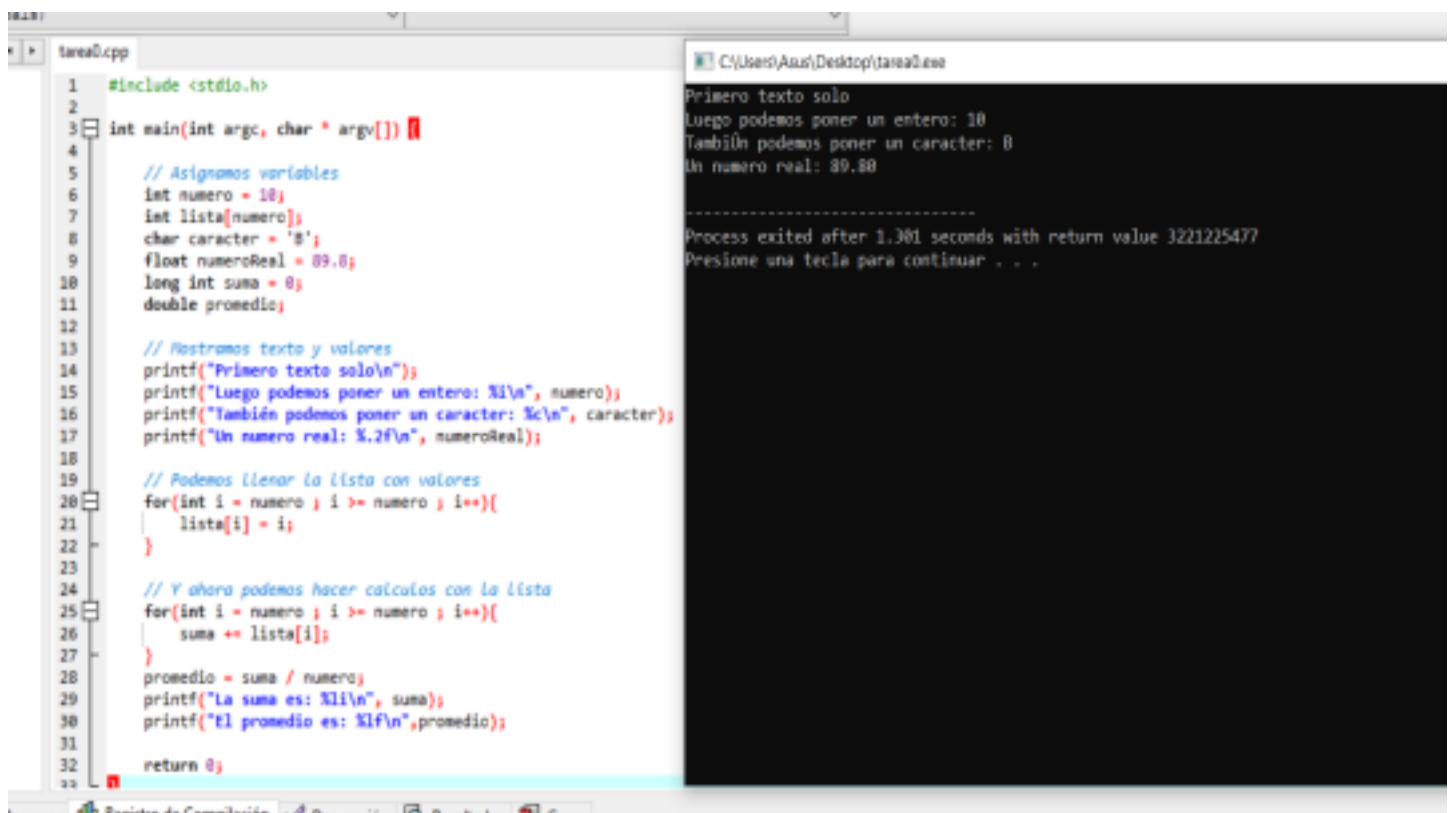
También puede utilizarse interfaces gráficas para trabajar con GB de manera más cómoda. Uno de esos interfaces es DD y puede descargarse en <http://sourceforge.net/projects/ddd/>

1._

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

Se ejecutó y probó el programa adecuadamente, usando comandos, samba, se pudo llegar a que el primer programa no tenía ningún error y se podía correr bien.

Lo que hace es que recibe un número, después otro, se guarda el resultado de la suma más el contador así que aumenta de dos en dos y inicia desde 1.



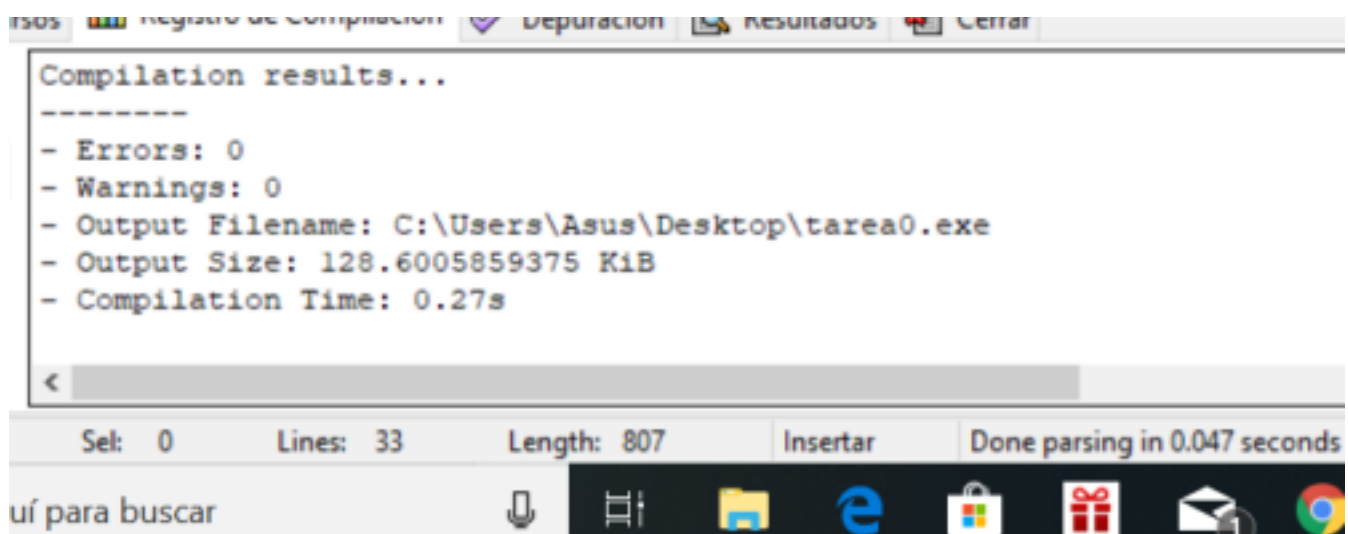
The image shows a code editor window on the left and a terminal window on the right. The code editor displays a C++ program named `tarea0.cpp`. The program includes `<stdio.h>` and defines a `main` function. It declares variables: `numero` (int), `lista` (array of ints), `caracter` (char), `numeroReal` (float), `suma` (long int), and `promedio` (double). The program prints a message, prompts for an integer, a character, and a real number. It then fills an array `lista` with values from `numero` to `numero + 1`. Next, it calculates the sum of the array elements and prints the sum and the average. Finally, it returns 0.

```
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 = '0';
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    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);
30    printf("El promedio es: %lf\n", promedio);
31
32    return 0;
33 }
```

The terminal window shows the output of the program. It displays the prompts and the user's input for the integer, character, and real number. After the program execution, it shows the calculated sum and average. The process exited after 1.301 seconds with a return value of 3221225477.

```
C:\Users\Ausu\Desktop\tarea0.exe
Primero texto solo
Luego podemos poner un entero: 10
También podemos poner un caracter: 0
Un numero real: 89.80

-----
Process exited after 1.301 seconds with return value 3221225477
Presione una tecla para continuar . . .
```



```

21             lista[i] = i;
Missing separate debuginfos, use: debuginfo-install glibc-2.15-37.fc17.x86_64
(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
[A debugging session is active.

```

Inferior 1 [process 22276] will be killed.

```

Quit anyway? (y or n) y
[[fp03alu37@samba ~]$ 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/fp03alu37/ejemplo1...done.
(gdb) break 20
Breakpoint 1 at 0x4005f7: file ejemplo1.c, line 20.
(gdb) run
Starting program: /users/fp03/fp03alu37/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=0x7fffffff398) 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) n
21             lista[i] = i;
(gdb) print lista
$1 = {-163754450, 0, 4195102, 0, -1, 0, -7536, 32767, -7520, 32767}
(gdb) display i
1: i = 10
(gdb) dsplay lista
Undefined command: "dsplay". Try "help".
(gdb) n
20         for(int i = numero ; i >= numero ; i++){
1: i = 10
(gdb) n
21             lista[i] = i;
1: i = 11
(gdb) n
20         for(int i = numero ; i >= numero ; i++){
1: i = 11
(gdb) n
21             lista[i] = i;
1: i = 12
(gdb) n
20         for(int i = numero ; i >= numero ; i++){
1: i = 12
(gdb) n
21             lista[i] = i;
1: i = 13
(gdb) n
20         for(int i = numero ; i >= numero ; i++){
1: i = 13
(gdb) n
21             lista[i] = i;
1: i = 14
(gdb) n
20         for(int i = numero ; i >= numero ; i++){

```

2._CORREGIR

```
#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;
    AS=0;
    while(K<=N)
    {
        AS=AS+pow(X,K)/AP;
        K=K+1;
        AP=AP*K;
    }
    printf("Resultado=%le",AS);
}
```

Ya que para nosotros se nos hace más práctico y accesible usar C++ tanto para la corrección de programas como la ejecución lo decidimos usar puesto que es MÁS CLARO y sin tantas tantas cosas que para nosotros es innecesario, analizando y haciendo correcciones que nos marcaba c++ las cambiamos, y este fue el resultado al que llegamos.

La ejecución hace un EXPONENCIAL, dos números ingresados, X, Y.

CORREGIDO.

```
ejemplo2.cpp
1  #include <stdio.h>
2  #include <math.h>
3
4  int 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("\nX=");
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 }
```



```
os Registro de Compilación Depuración Resultados Cerrar
Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Asus\Desktop\Programas\ejemplo2.exe
- Output Size: 151.7294921875 KiB
- Compilation Time: 0.41s
<
Sel: 0 Lines: 23 Length: 352 Insertar Done parsing in 0.016 seconds
```

los términos calcular de la serie: $X^K/K!$ ");

,n",AS));

```
C:\Users\Asus\Desktop\Programas\ejemplo2.exe
Ingrese cuantos términos calcular de la serie:  $X^K/K!$ 
N=43
X=56
Resultado=1.#INF00e+000
-----
Process exited after 4.4 seconds with return value 0
Presione una tecla para continuar . . .
```

);

,n",AS));

```
C:\Users\Asus\Desktop\Programas\ejemplo2.exe
Ingrese cuantos términos calcular de la serie:  $X^K/K!$ 
N=9
X=4
Resultado=5.415414e+001
-----
Process exited after 3.403 seconds with return value 0
Presione una tecla para continuar . . .
```

3._ CORREGIR.

```
#include <stdio.h>

int main()
{
    int numero;

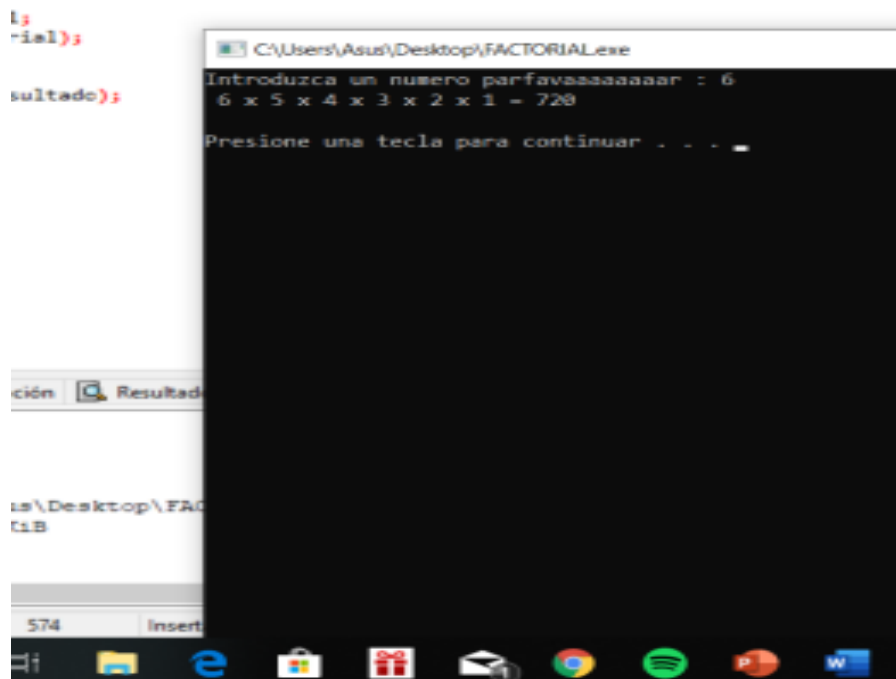
    printf("Ingrese un número:\n");
    scanf("%i", &numero);

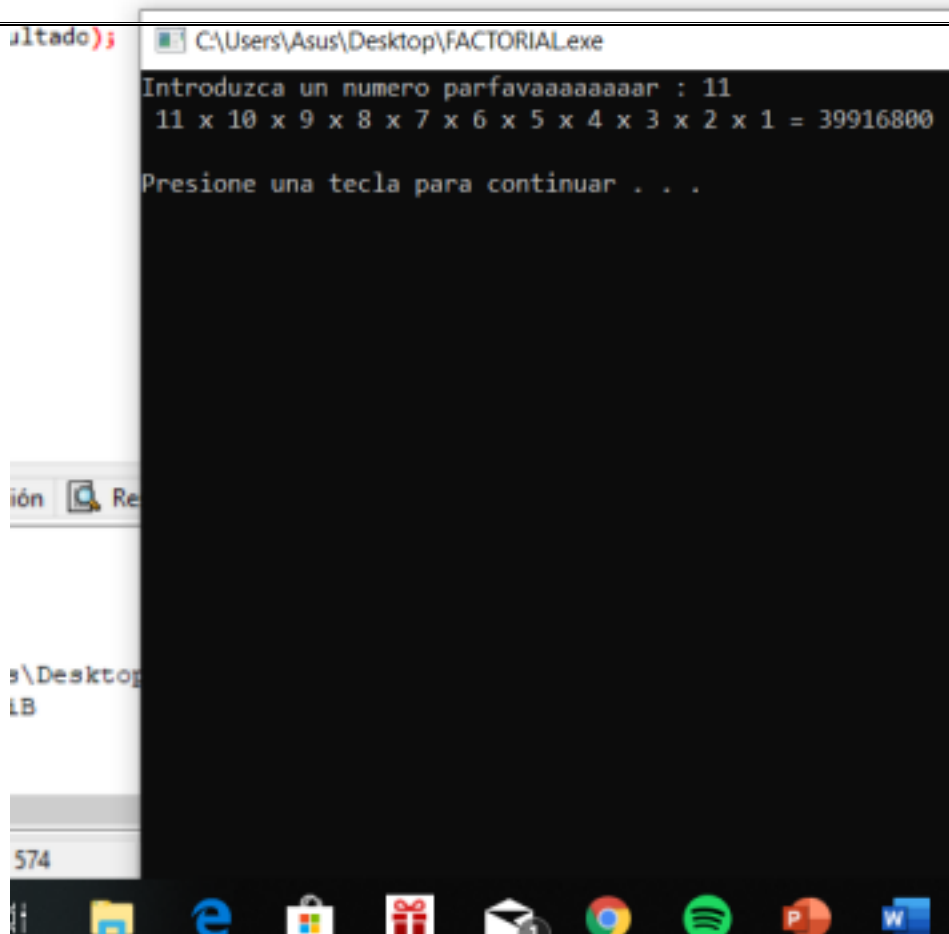
    long int resultado = 1;
    while(numero >= 0) {
        numero--;
        resultado *= numero;
    }

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

    return 0;
}
```

Haciendo uso de c++ de igual forma a las pasadas del misto puesto en la documentación de la misma introducida con anterioridad actual, se corrigió el programa, ya que tuvimos que hacer y cambiar varias que nos marcaba c++, el programa es para calcular la factorial de un número





A screenshot of a Windows desktop environment. A black command prompt window is open, displaying the execution of a program titled 'C:\Users\Asus\Desktop\FACTORIAL.exe'. The program prompts the user to 'Introduzca un numero parfavaaaaaaaaar : 11'. It then displays the calculation '11 x 10 x 9 x 8 x 7 x 6 x 5 x 4 x 3 x 2 x 1 = 39916800' and asks the user to 'Presione una tecla para continuar . . .'. The desktop background is white, and the taskbar at the bottom shows various application icons including File Explorer, Edge, Store, Mail, Chrome, Spotify, PowerPoint, and Word. A file explorer window is partially visible on the left side of the screen.

```
Altado); C:\Users\Asus\Desktop\FACTORIAL.exe
Introduzca un numero parfavaaaaaaaaar : 11
11 x 10 x 9 x 8 x 7 x 6 x 5 x 4 x 3 x 2 x 1 = 39916800
Presione una tecla para continuar . . .
```

CONCLUSIONES:

Es mucho más fácil y práctico usar c++ ya que no da tantos rodeos y te lo dice claro y conciso, cada programa es diferente pero tiene fallas, las cuales igual con habilidad se pudieron haber corregido, aún así amamos c++ a comparación de otro editores ya que los demás están pésimos.