|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Ing. Claudia Rodríguez Espino |
| *Asignatura:* | Fundamentos de Programación |
| *Grupo:* | 1102 |
| *No de Práctica(s):* | 09 |
| *Integrante(s):* | Alejandro Meneses Mercado |
|  |  |
|  |  |
| *Semestre:* | 2018-1 |
| *Fecha de entrega:* | 20 de octubre del 2017 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

Práctica 09: Estructuras de repetición

Objetivo:

Elaborar programas en C para la resolución de problemas básicos que incluyan las estructuras de repetición y la directiva define.

Desarrollo:

Primero vimos un repaso acerca de lo visto en clase y practicas anteriores de las estructuras de repetición, las cuales tambien son llamadas cíclicas o iterativas y permiten ejecutar un conjunto de instrucciones de manera repetida mientras que la expresión lógica a evaluar se cumpla.

Vimos que en C hay 3 estruturas de repetición las cuales son: while, do-while y for.

Empezamos revisando la estructra while que primero valida la expresión lógica y si ésta se cumple procede a ejecutar el bloque de instrucciones de la estructura. Si la condición no se cumple se continúa el flujo normal del programa sin ejecutar el bloque de la estructura. El bloque de código se puede ejecutar de una a ene veces.

La sintaxis de esta estructua es:

while(expresión logica)

{

Bloque de instrucciones

}

Después vimos la estructura do-while la cual ejecuta el bloque de código que se encuentra dentro de las llaves y después valida la condición. El bloque de código se ejecuta de una a ene veces.

Su sintaxis es:

do

{

Bloque de instrucciones

}

while(expresión lógica);

Después vimos la última estructura llamada for la cual permite realizar repeticiones cuando se conoce el número de elementos que se quiere recorrer.

Esta estructura ejecuta 3 acciones básicas antes o después de ejecutar el bloque de código. La primera acción es la inicialización, en la cual se pueden definir variables e inicializar sus valores, la segunda acción consta de una expresión lógica, la cual se evalúa y, si ésta es verdadera, ejecuta el bloque de código, si no se cumple se continúa la ejecución del programa, la tercera parte consta de un conjunto de operaciones que se realizan cada vez que termina de ejecutarse el bloque de código y antes de volver a validar la expresión lógica.

Su sintaxis es:

for( inicialización; expresión lógica; operaciones)

{

Bloque de instrucciones

}

Otra cosa que vimos fue el Define el cual permite definir constantes o literales; se les nombra también como constantes simbólicas. Vimos que su sintaxis es:

#define <nombre> <valor>

Cada vez que aparezca el nombre en el programa se cambiará por el valor definido. El valor puede ser numérico o puede ser texto.

Un ejemplo que hicimos en clase fue definir printf como p de la sig. manera: #define printf p

Por ultimo vimos el break que proporciona una salida anticipada dentro de una estructura de repetición, tal como lo hace en un switch, el break provoca que el ciclo que lo encierra termine inmediatamente.

Como actividad tuvimos que elaborar 4 programas utilizando las estructuras de repetición vistas:

1. Programa que muestra las tablas de multiplicar del 1 al 10 con la estructura while.

Código:

#include<stdio.h>

int a=1,b=1,r;

main()

{

printf("\n\t\tTablas de multiplicar del 1 al 10");

printf("\n");

while (a<=10)

{

printf("\n\t\tTabla de multiplicar del %d",a);

printf("\n");

while (b<=10)

{

r=a\*b;

printf("\n %d \* %d = %d",a,b,r);

b= b+1;

}

b=1;

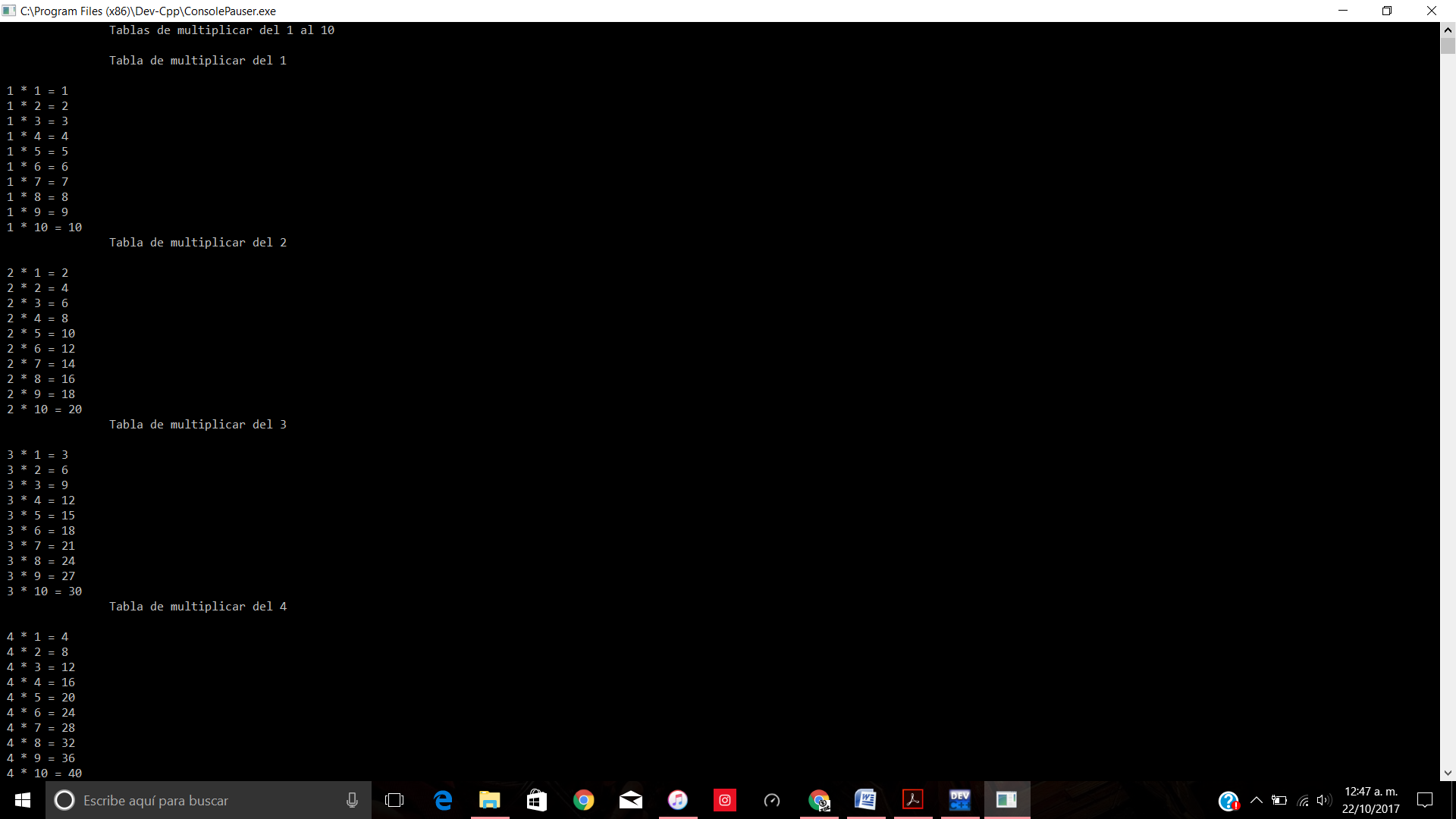
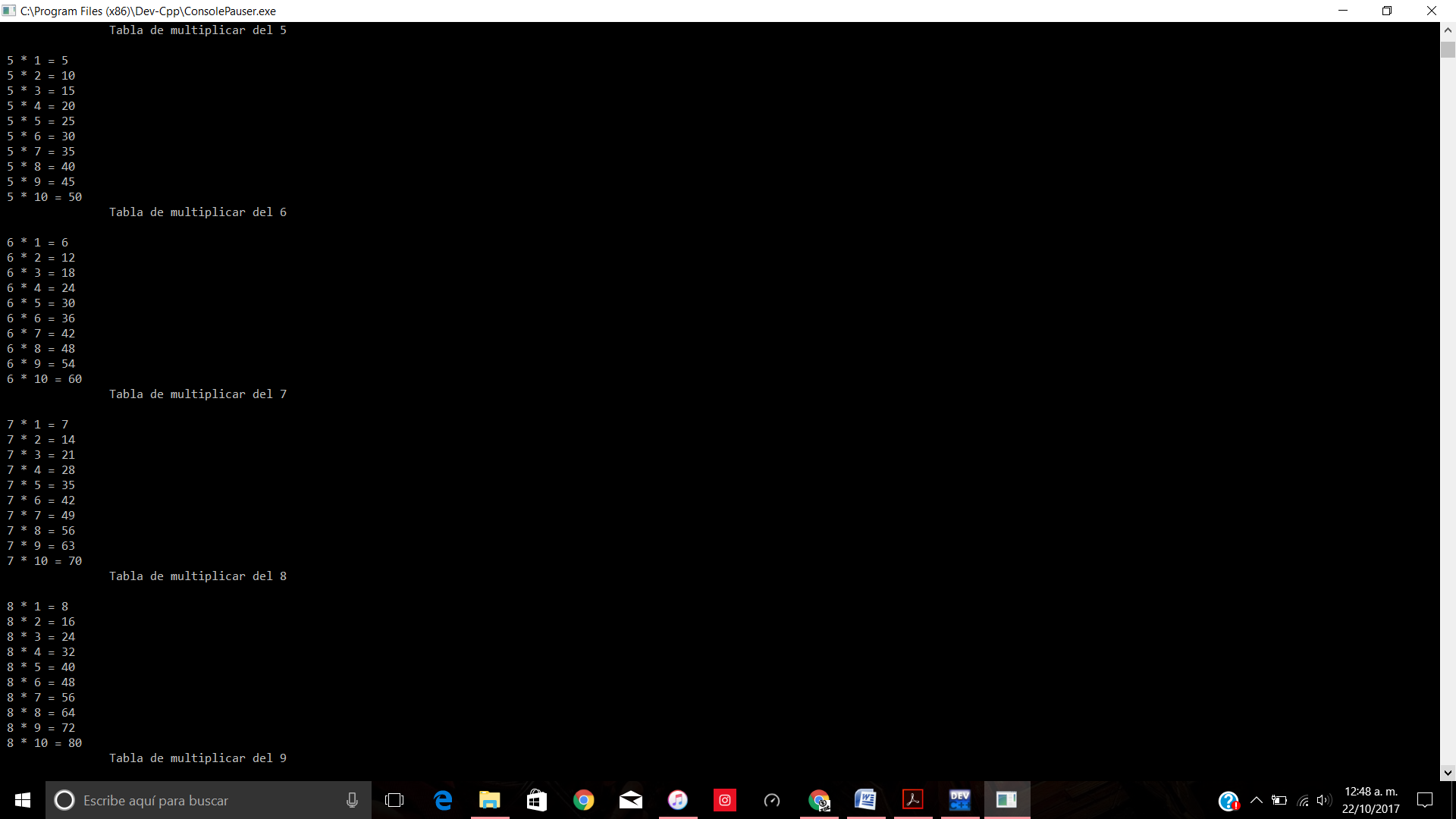
a = a + 1 ;

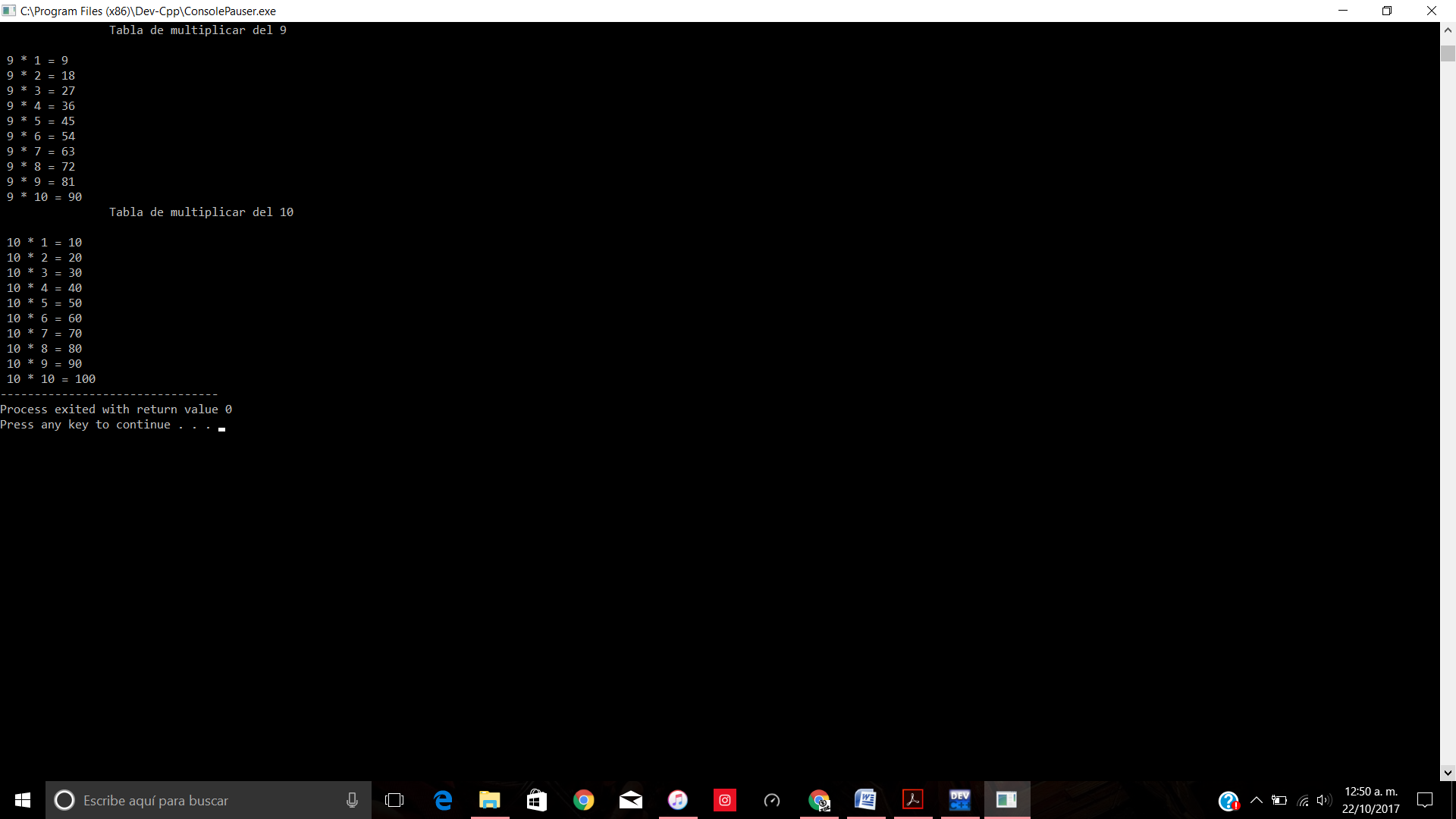
}

return 0;

}

Ejecutable:





1. Programa que muestra las tablas de multiplicar del 1 al 10 con la estructura do-while.

Código:

#include<stdio.h>

int a=1,b=1,r;

main()

{

printf("\n\tPrograma que muestra las tablas de multiplicar del 1 al 10");

printf("\n");

do

{

printf("\n\t\tTabla de multiplicar del %d",a);

printf("\n");

do

{

r=a\*b;

printf("\n %d \* %d = %d",a,b,r);

b= b+1;

}

while(b<=10);

b=1;

a = a + 1 ;

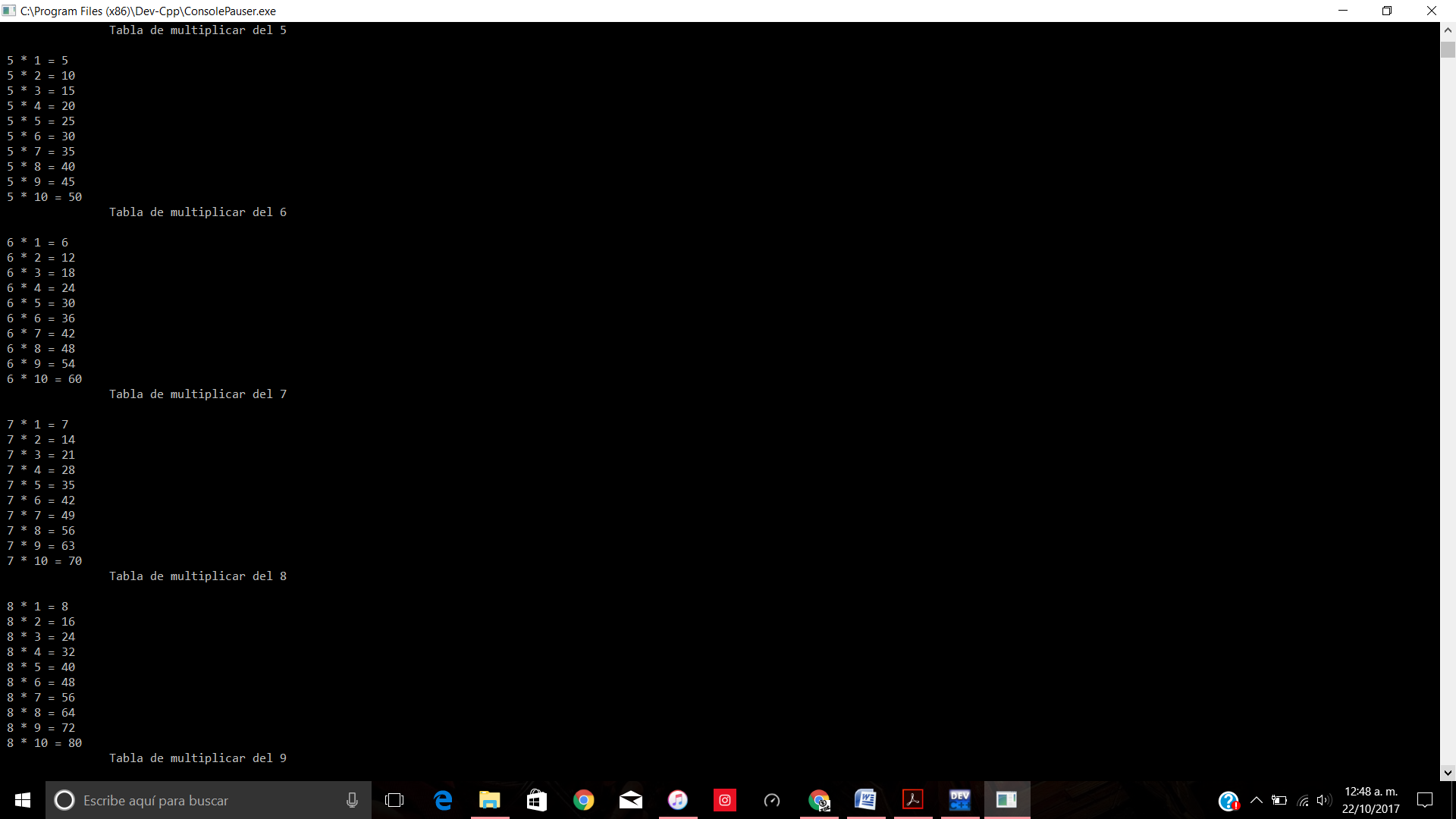
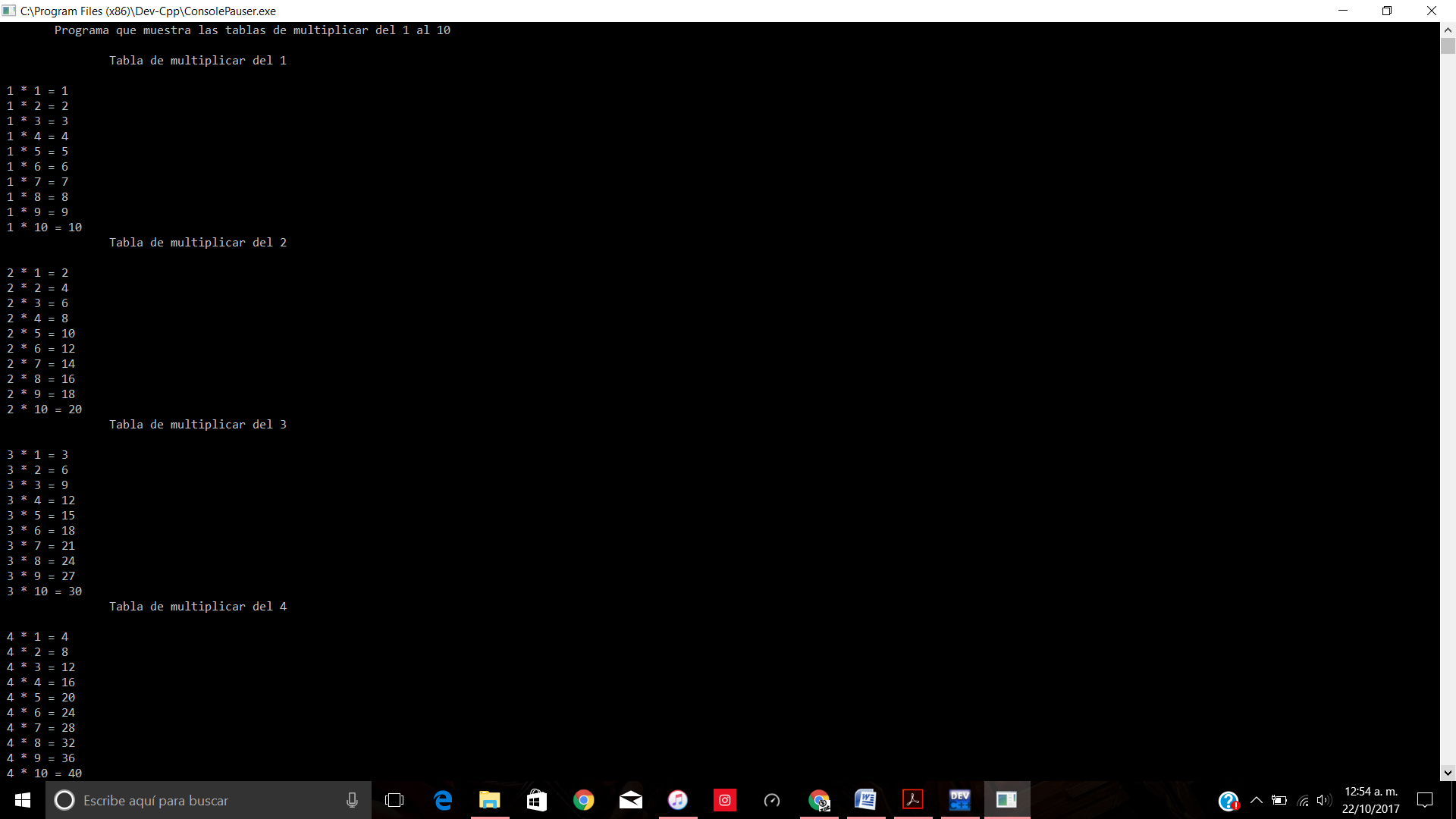
}

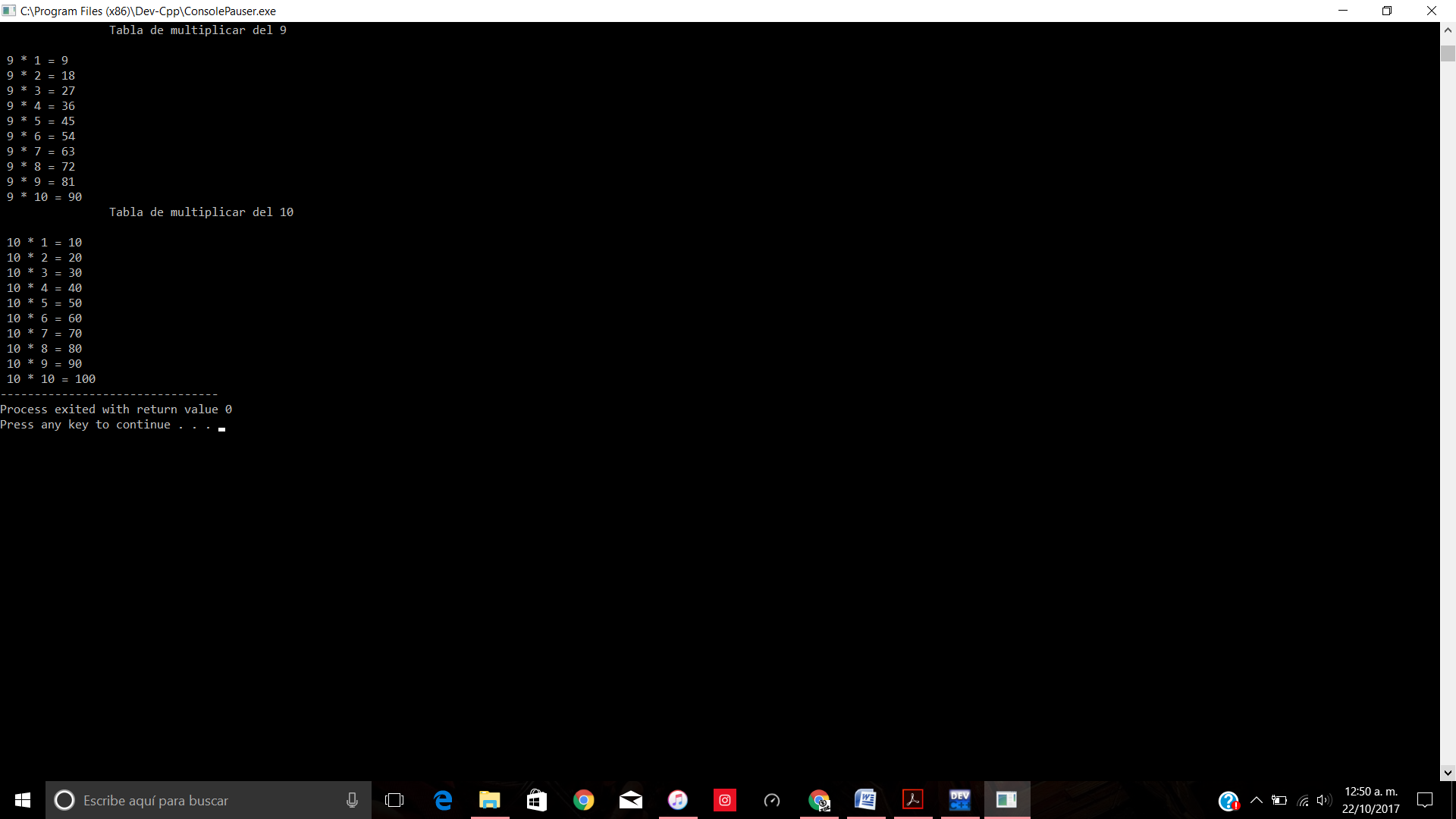
while(a<=10);

return 0;

}

Ejecutable:





1. Programa que muestra las tablas de multiplicar del 1 al 10 con la estructura for.

Código:

#include<stdio.h>

int a,b,r;

main()

{

for(a=1;a<11;a++)

{

printf("\n\t\tTabla de multiplicar del %d",a);

printf("\n");

for(b=1;b<11;b++)

{

r=a\*b;

printf("\n %d\*%d=%d",a,b,r);

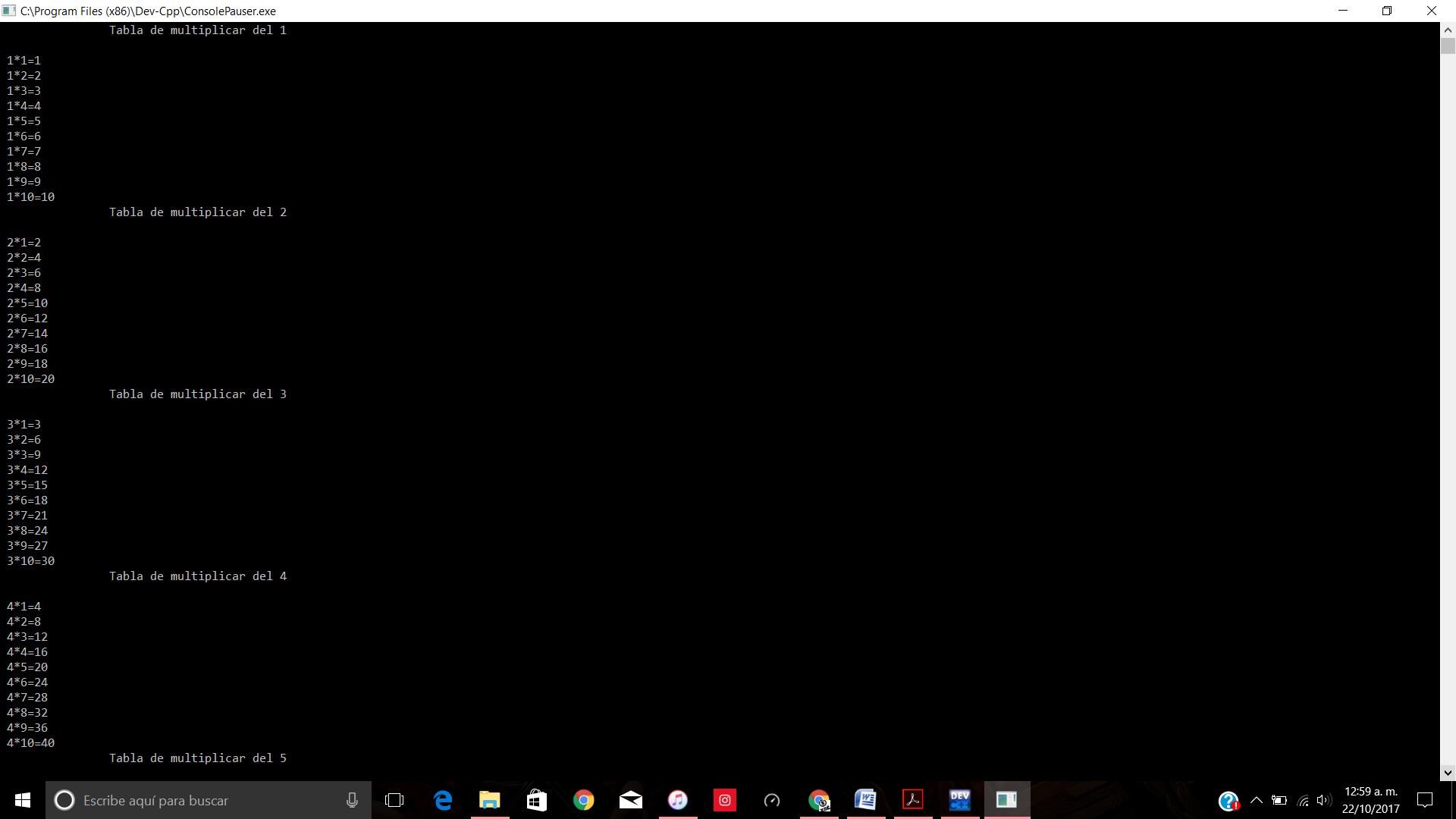
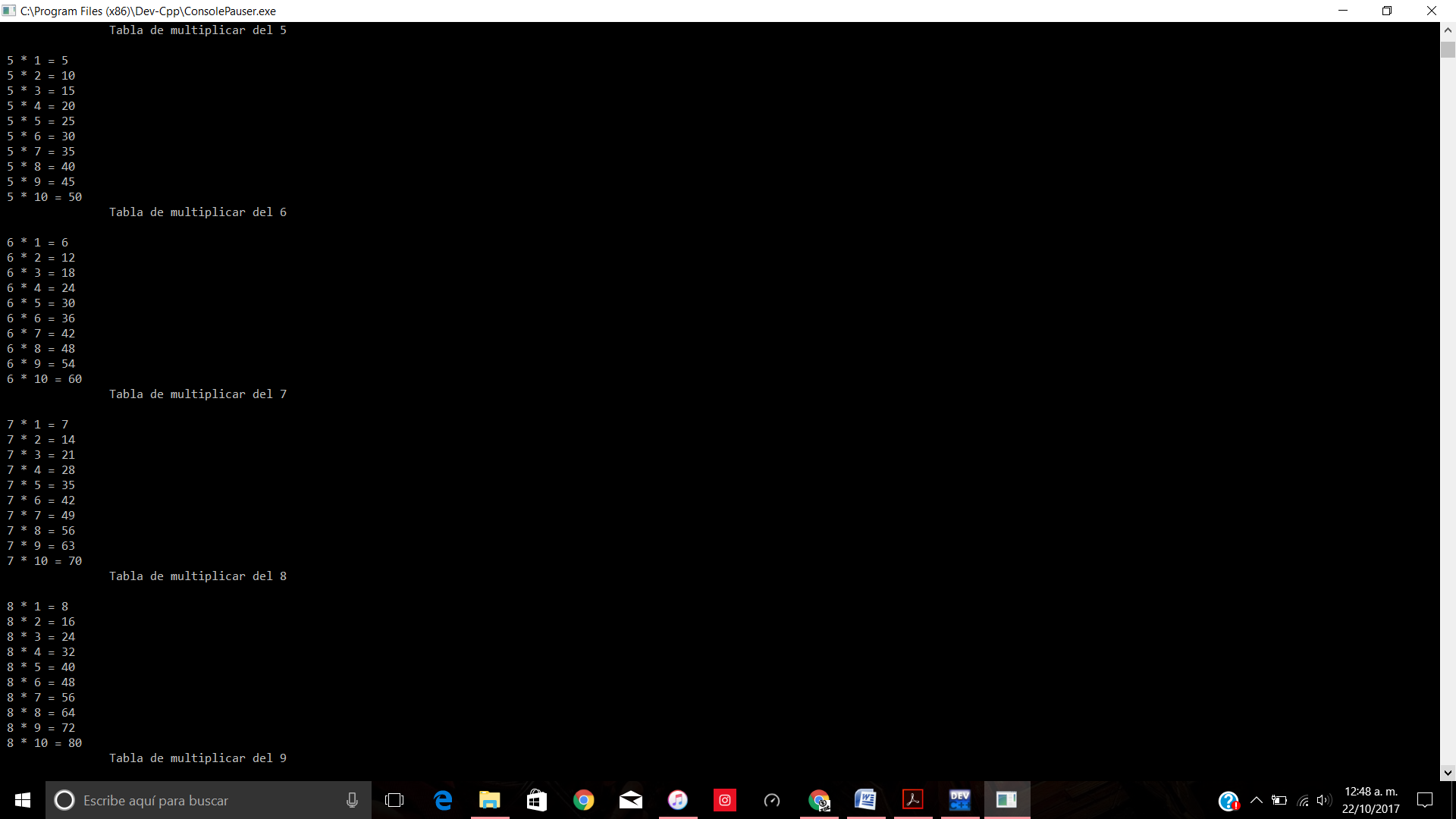
}

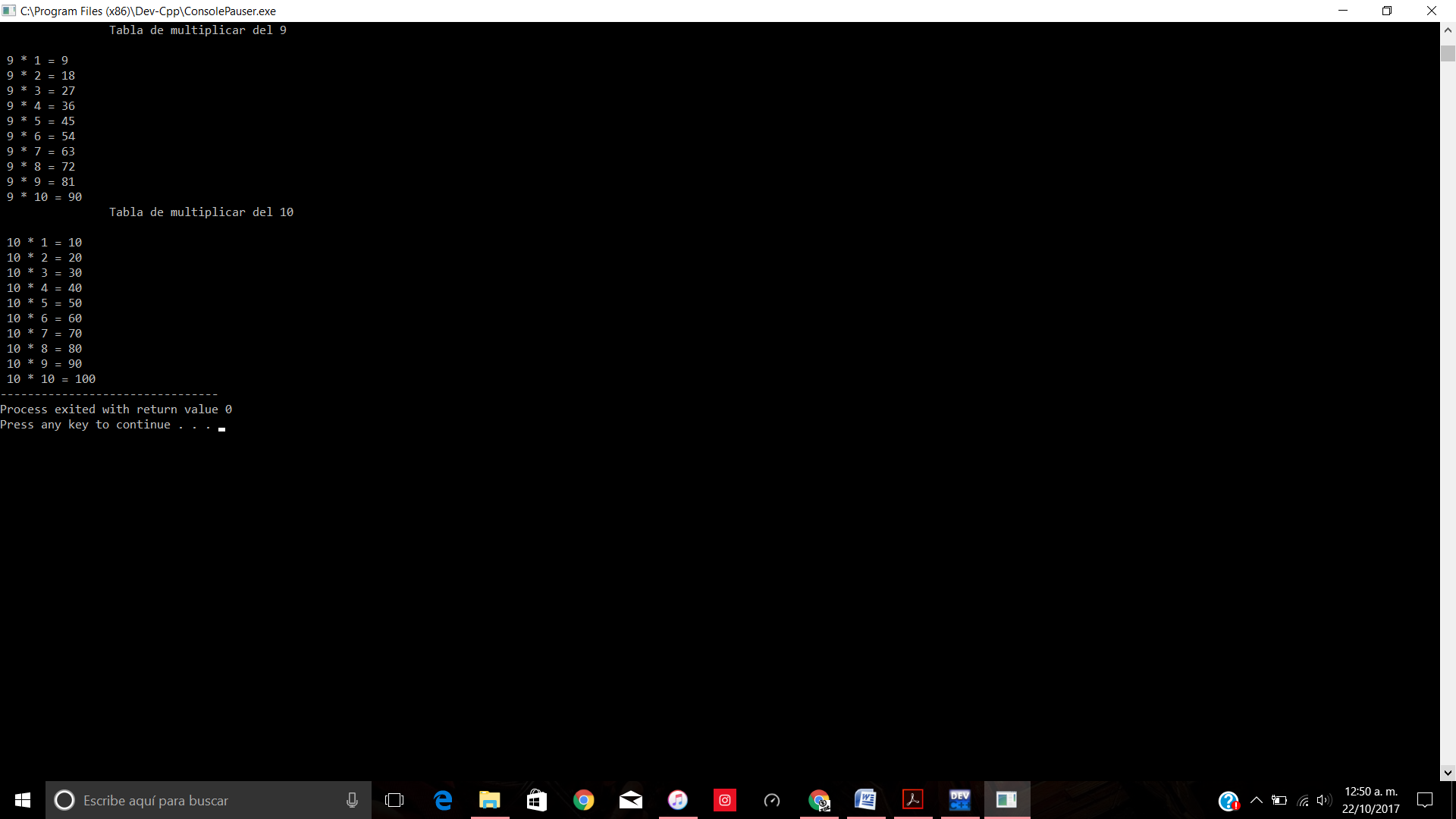
}

return 0;

}

Ejecutable:





1. Programa que calcula la potencia de un número

Código:

#include<stdio.h>

#include<math.h>

float a,n,r;

main()

{

printf("\t\t\tPrograma que calcula la potencia de un numero\n");

printf("\nIntroduce el valor de tu base: \n");

scanf("%f",&a);

printf("\nIntroduce el valor de tu exponente: \n");

scanf("%f",&n);

while(n<11)

{

r=pow(a,n);

printf("El numero %.0f elevado a la %.0f es igual a %.0f",a,n,r);

break;

}

while(n>10)

{

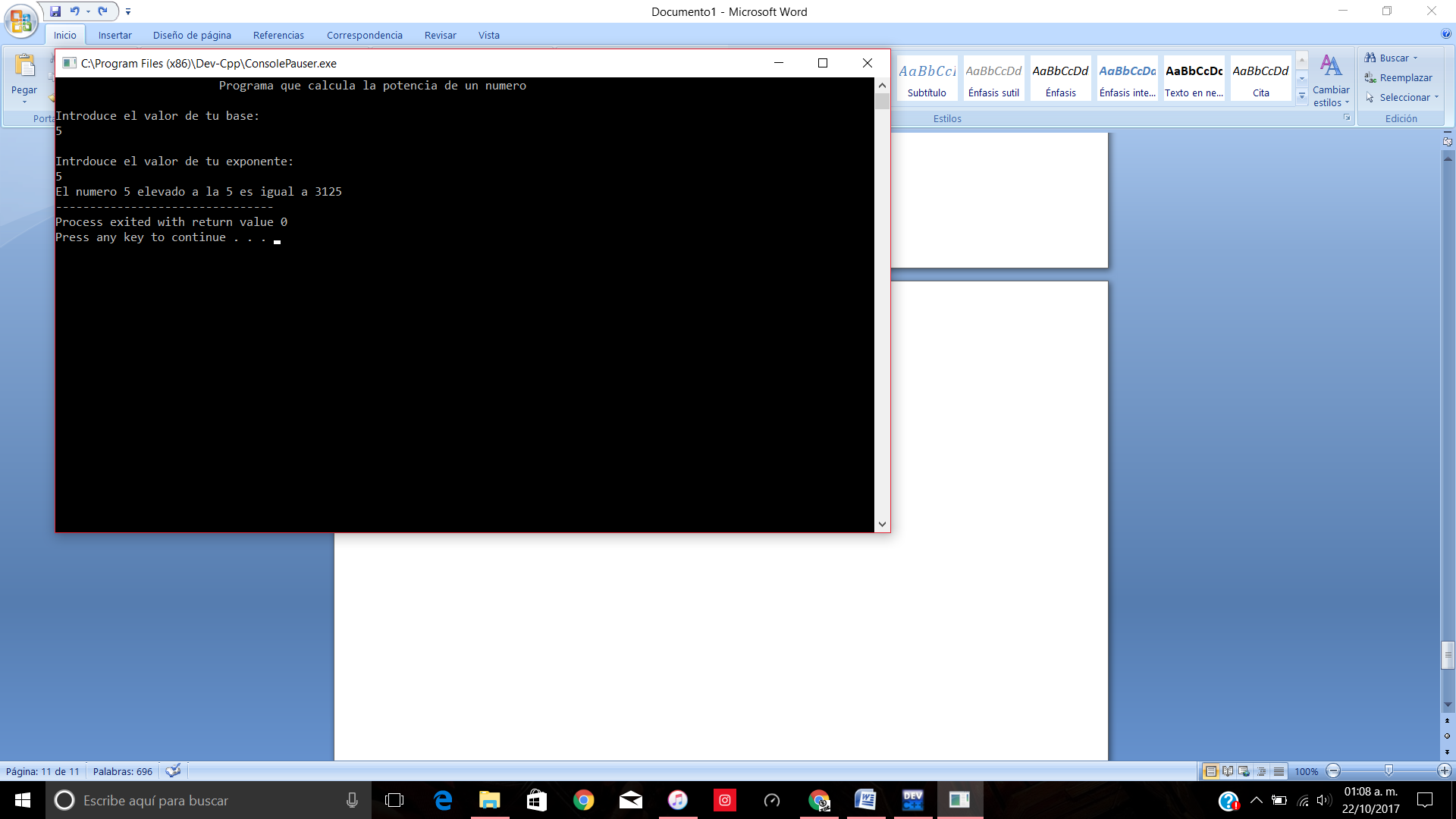
printf("El exponente debe ser menor a 10");

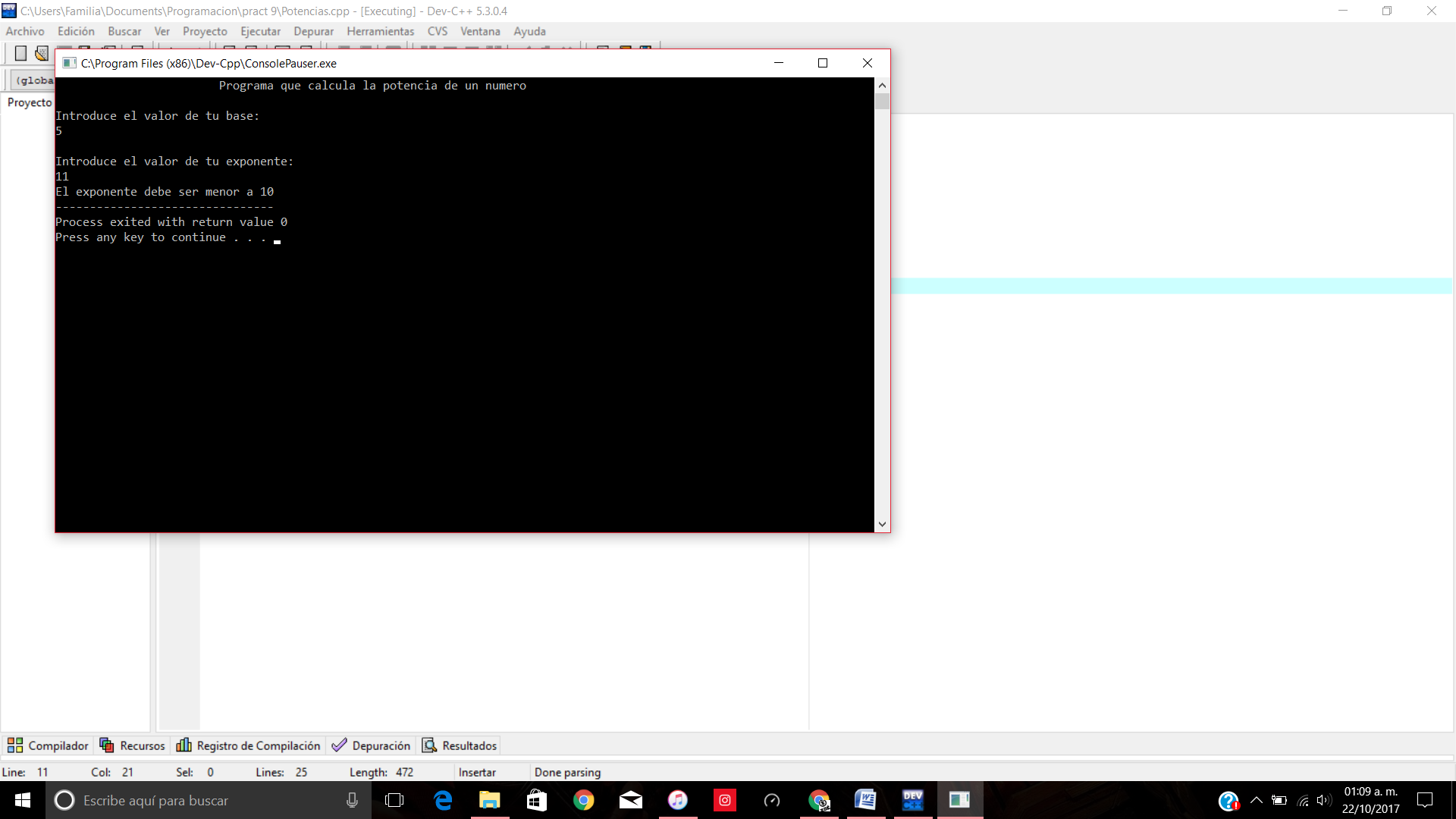
break;

}

}

Ejecutable:





Conclusiones:

Con esta práctica pudimos reforzar y entender mejor acerca de estas estructuras de repetición ya vistas previamente en clase y que nos serviran mucho para algunos programas que hagamos en clase.