|  |  |  |
| --- | --- | --- |
|  | **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):* | 12 |
| *Integrante(s):* | Alejandro Meneses Mercado |
|  |  |
|  |  |
| *Semestre:* | 2018-1 |
| *Fecha de entrega:* | 10 de noviembre del 2017 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

Práctica 12: Funciones

Objetivo:

Elaborar programas en C donde la solución del problema se divida en funciones. Distinguir lo que es el prototipo o firma de una función y la implementación de ella, así como manipular parámetros tanto en la función principal como en otras.

Desarrollo:

Primero vimos que un programa en C consiste en una o mas funciones, esto con el objetivo de dividir las tareas y que sea más fácil la depuración, la mejora y el entendimiento del código.

Después vimos su sintaxis la cual es:

valorRetorno nombre (parámetros){

// bloque de código de la función

}

Donde los parámetros son datos de entrada con los que trabajará la función, se definen dentro del paréntesis, separados por comas e indicando el tipo de dato.

Y vimos que el valor de retorno indica el tipo de dato que va a regresar la función al terminar el bloque de código de la misma.

Tambien vimos que las funciones deben ser declaradas antes de ser invocadas. Por lo que una buena práctica es declarar todas las funciones al inicio del programa.

Y su sintaxis es:

valorRetorno nombre (parámetros);

Con los mismos elementos que se vieron anteriormente solo que sin el bloque de código, porque solo estamos declarando a la función.

Después vimos que las variables declaradas dentro de un programa tienen un tiempo de vida que depende de la posición donde se declaren.

Aparte vimos que en C hay dos tipos de variables: variables locales y variables globales.

Revisamos cada una de ellas y vimos que las variables que se declaren dentro de cada función se conocen como variables locales ya que existen al momento de que la función es llamada y desaparecen cuando la función llega a su fin.

Y las variables que se declaran fuera de cualquier función se llaman variables globales porque existen durante la ejecución de todo el programa y pueden ser utilizadas por cualquier función.

También vimos que la función main puede recibir parámetros.

Vimos su sintaxis la cual es:

int main (int argc, char \*\* argv);

Aquí main puede recibir como parámetro de entrada un arreglo de cadenas al ejecutar el programa. La longitud del arreglo se guarda en el primer parámetro (argument counter) y el arreglo de cadenas se guarda en el segundo parámetro (argument vector).

Y vimos que para ejecutarlos se usa nombrePrograma.exe arg1 arg2 arg3 ...

Por último vimos que en C se pueden definir elementos estáticos.

Revisamo su sintaxis la cual es:

static tipoDato nombre;

static valorRetorno nombre(parámetros);

El atributo static en una variable hace que ésta permanezca en memoria desde su creación y durante toda la ejecución del programa, es decir, su valor se mantendrá hasta que el programa llegue a su fin.

Como actividad tuvimos que elaborar 3 programas con funciones.

1. Programa que calcula el cubo de un número

Código:

#include<stdio.h>

#include<conio.h>

long cubo();

long res,in,out;

main()

{

cubo();

getch();

}

long cubo()

{

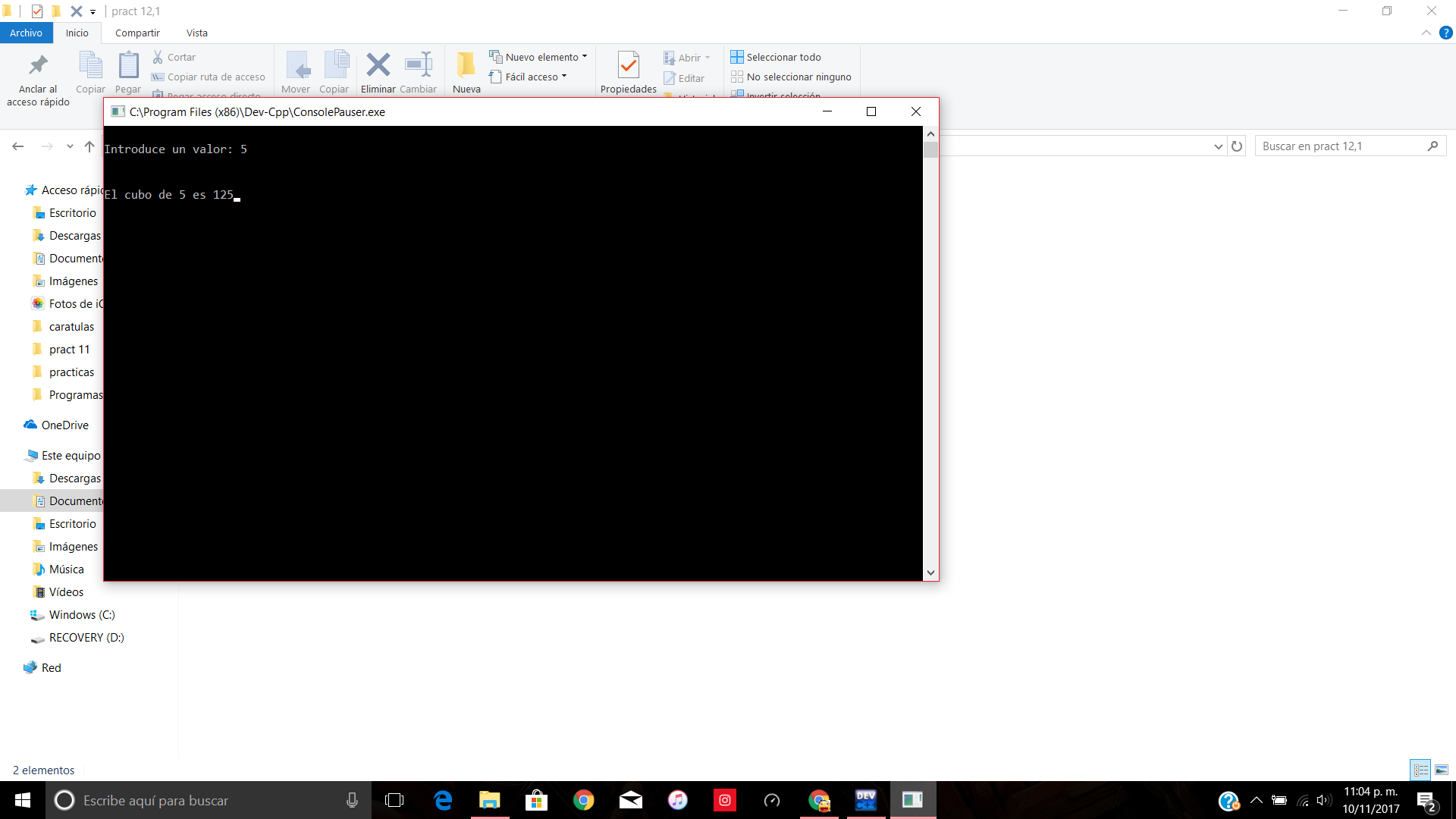
printf("\nIntroduce un valor: ");

scanf("%ld",&in);

res=(in\*in\*in);

printf("\n\nEl cubo de %ld es %ld",in,res);

}



1. Programa que utiliza funciones para crear una calculadora básica

Código:

#include <stdio.h>

int x,reg;

float a,b,c;

int menu();

float suma(), resta(), multiplicacion(), division();

main()

{

do

{

menu();

switch(x)

{

case 1:

suma();

break;

case 2:

resta();

break;

case 3:

multiplicacion();

break;

case 4:

division();

break;

default:

printf("Opcion invalida\n");

break;

}

printf("\nDeseas regresar al menu principal: (1) Si (0) No\n");

scanf("%d",&reg);

}

while(reg==1);

}

int menu()

{

system("cls");

printf("\nDame el valor de tu primer numero: ");

scanf("%f",&a);

printf("\nDame el valor de tu segundo numero: ");

scanf("%f",&b);

printf("\n");

printf("1).- Suma\n");

printf("2).- Resta\n");

printf("3).- Multiplicacion\n");

printf("4).- Division\n");

printf("\nSelecciona una Opcion: ");

scanf("%d",&x);

return x;

}

float suma()

{

system("cls");

c=a+b;

printf("\nEl resultado de la operacion es: %.2f\n", c);

return c;

}

float resta()

{

system("cls");

c=a-b;

printf("\nEl resultado de la operacion es: %.2f\n", c);

return c;

}

float multiplicacion()

{

system("cls");

c=a\*b;

printf("\nEl resultado de la operacion es: %.2f\n", c);

return c;

}

float division()

{

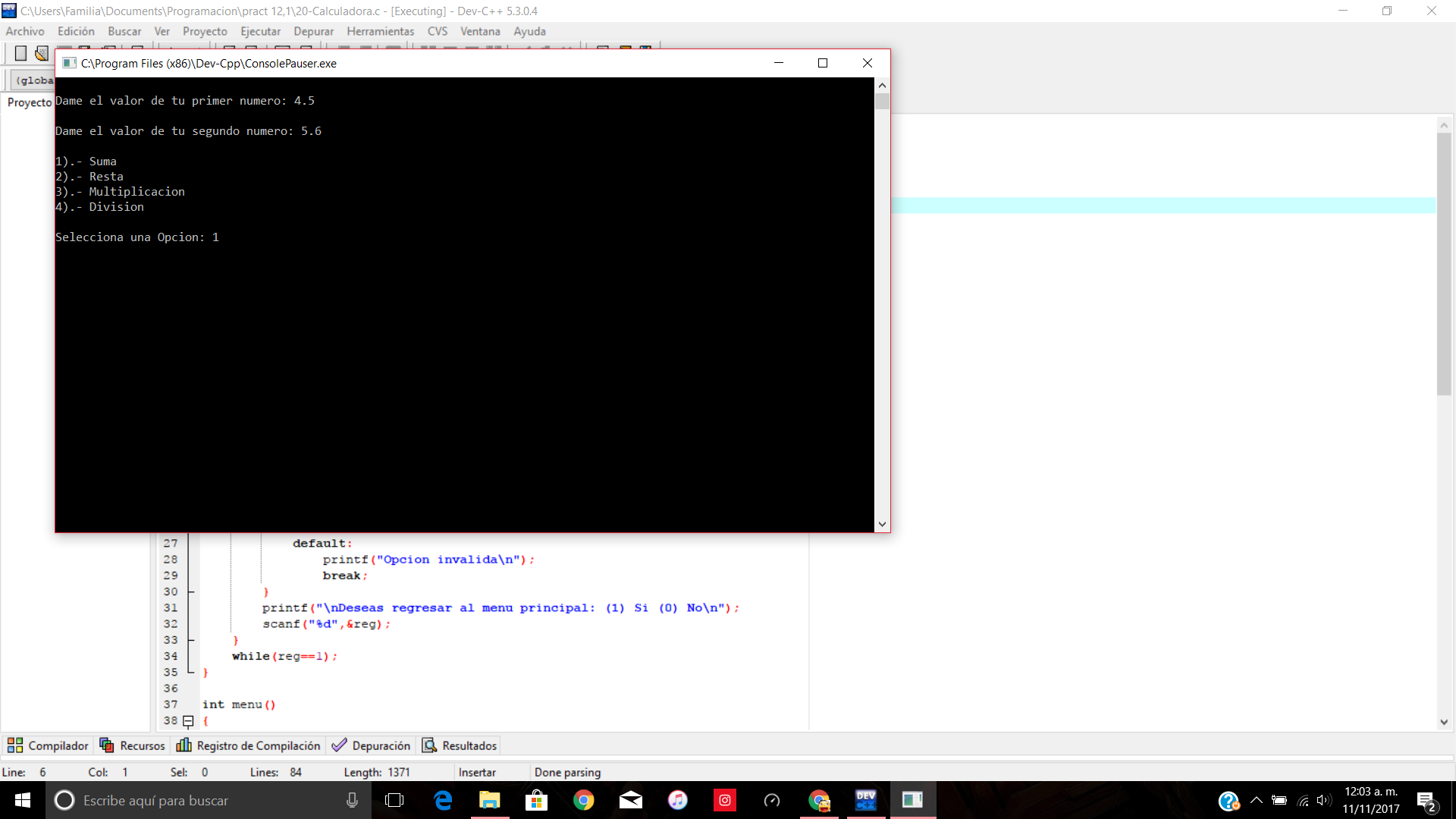
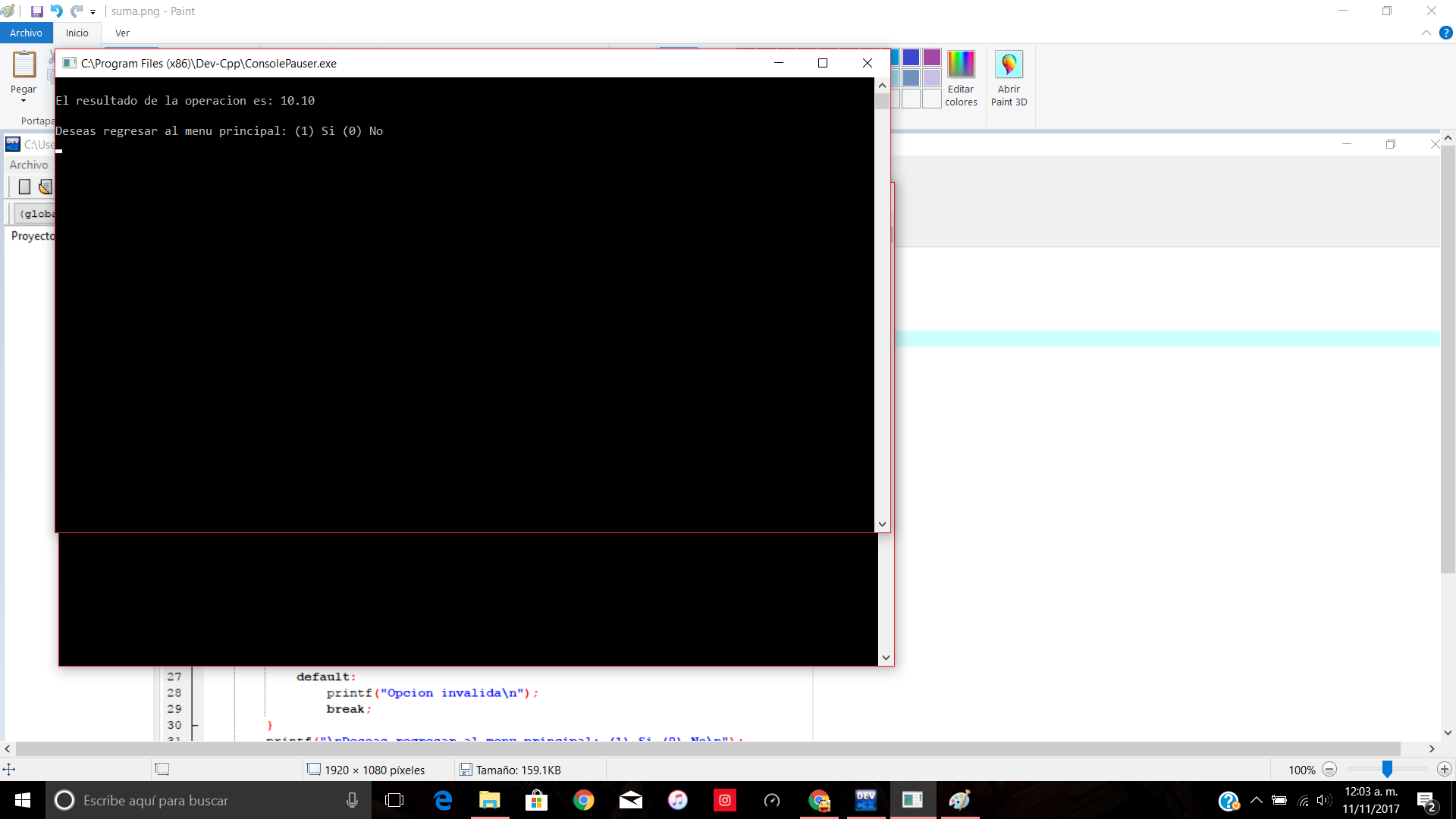
system("cls");

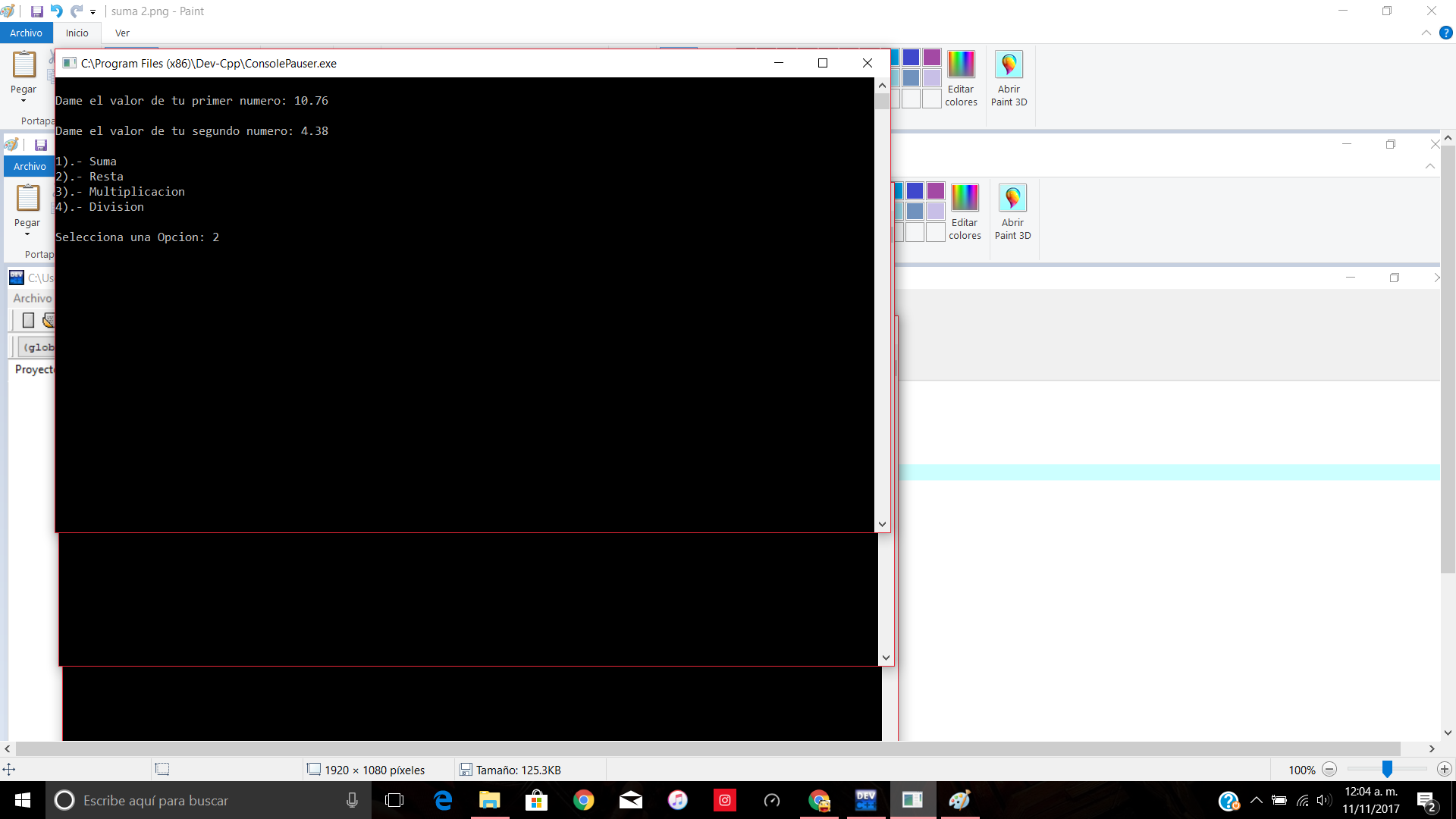
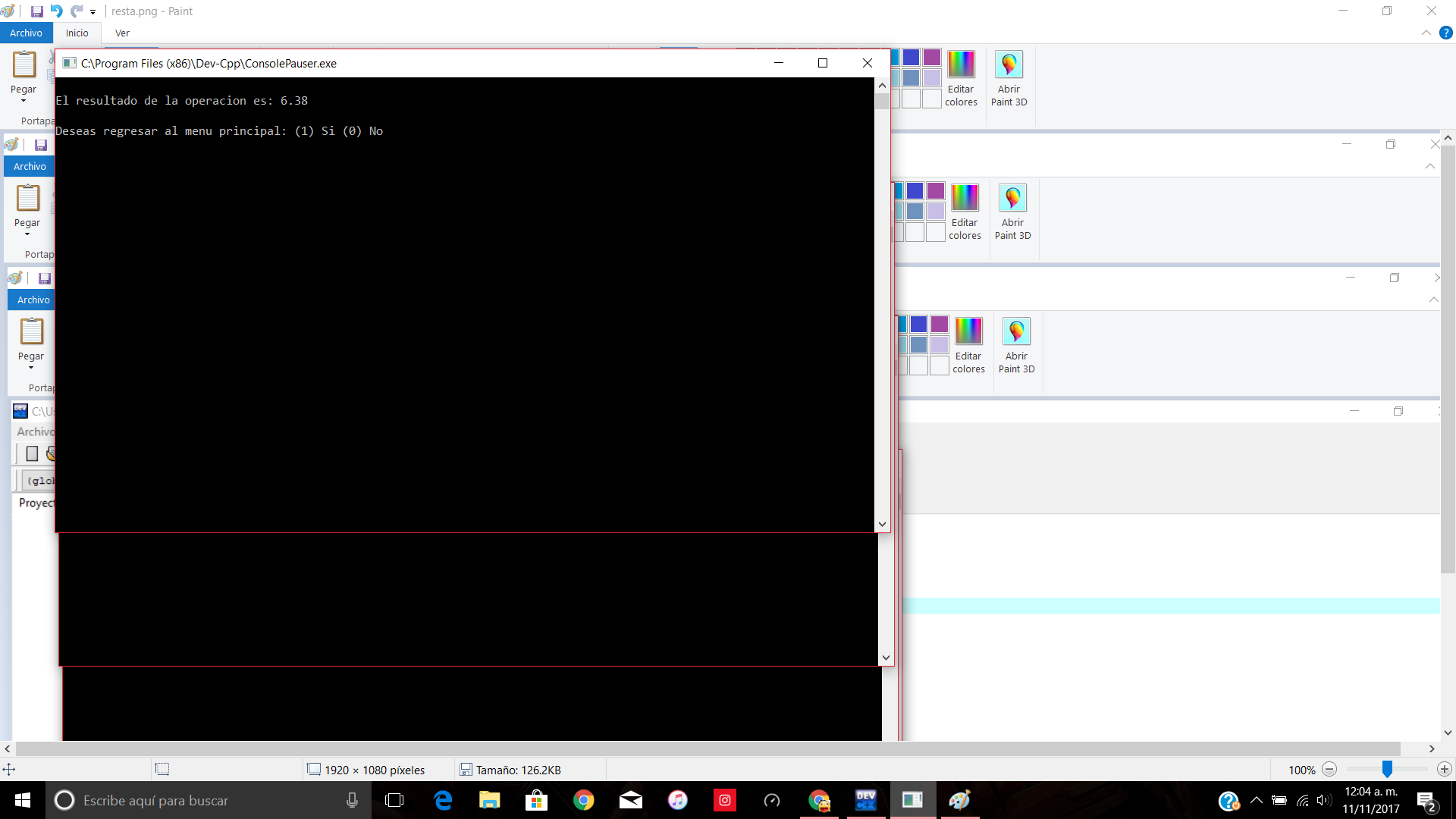
c=a/b;

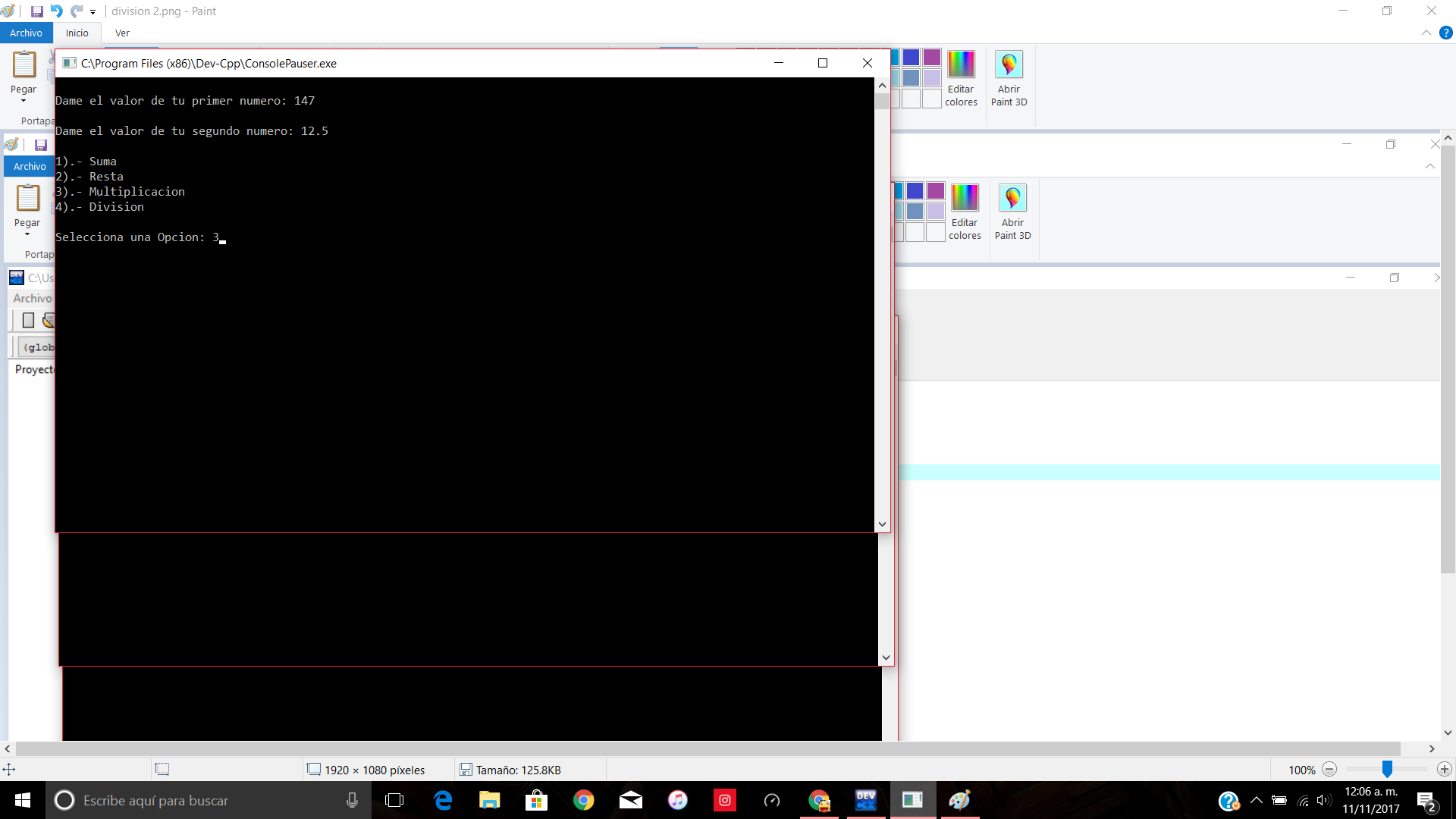
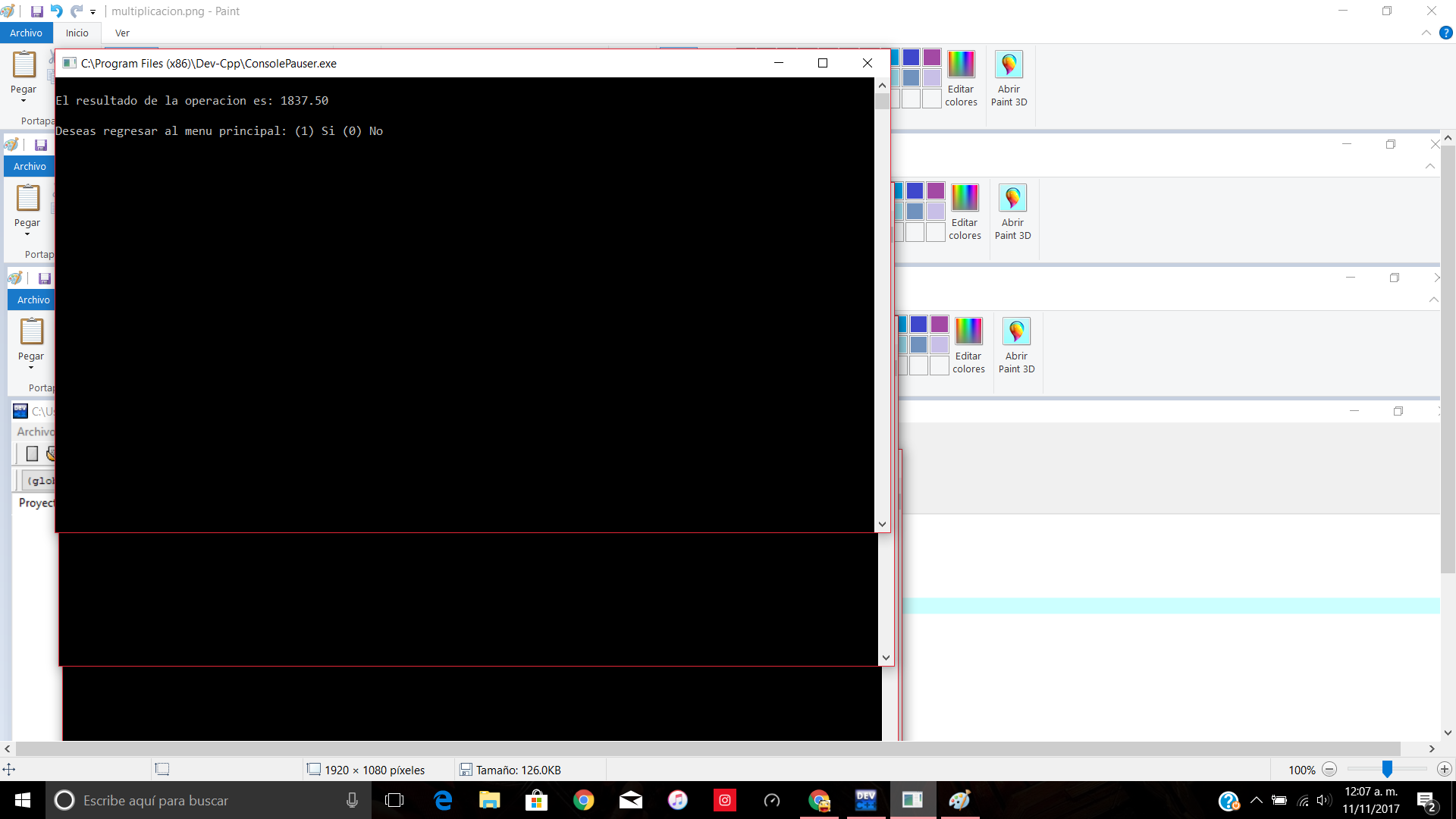
printf("\nEl resultado de la operacion es: %.2f\n", c);

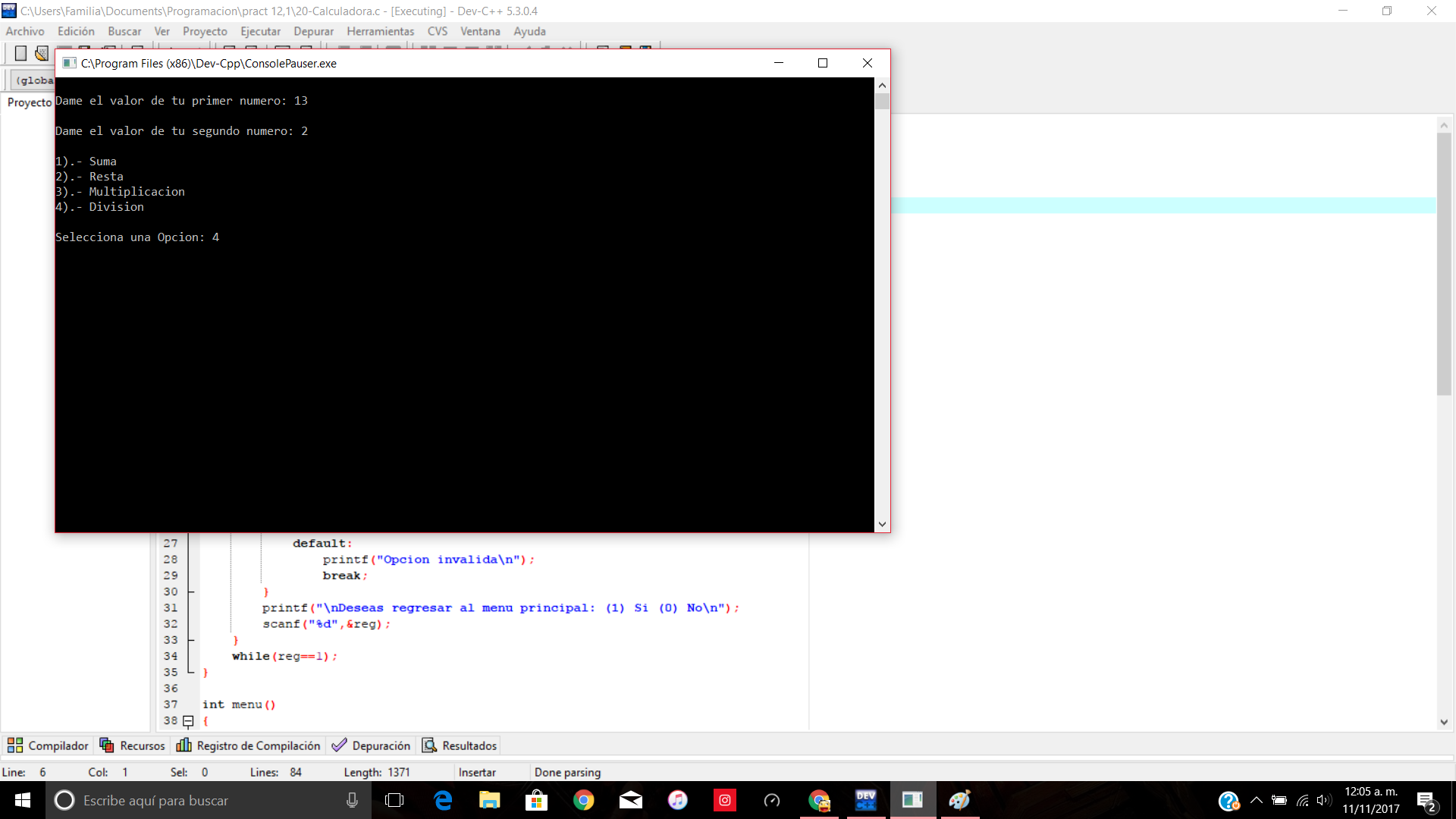
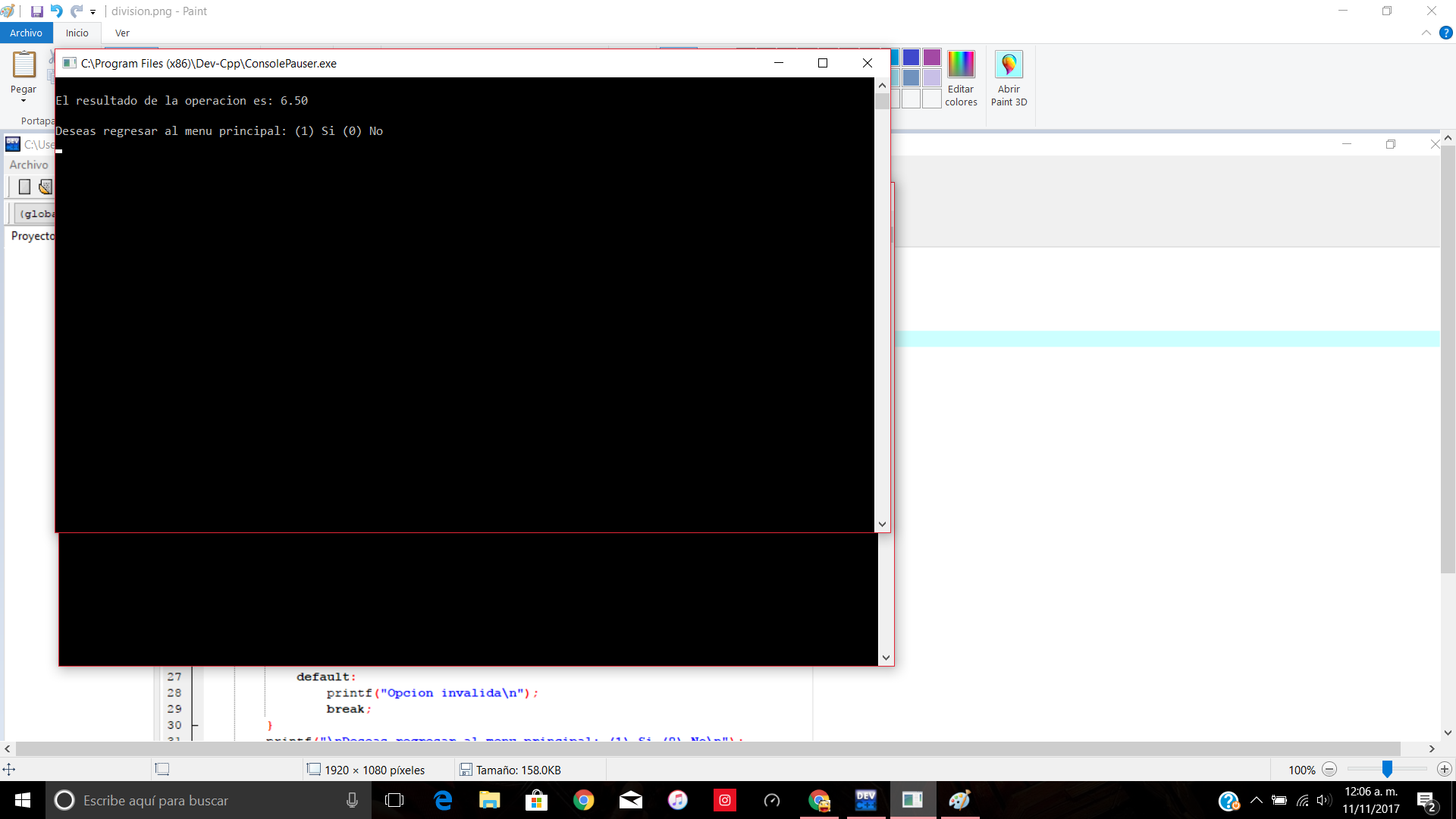
return c;

}









1. Programa que utiliza 2 funciones para pedir un valor, imprimirlo y multiplicarlo por 100

Código:

#include <stdio.h>

int funcion1();

int funcion2();

int x;

main()

{

funcion2();

x=x\*100;

printf("\nEl resultado es %d", x);

}

int funcion1()

{

printf("Dame un numero: ");

scanf("%d",&x);

return x;

}

int funcion2()

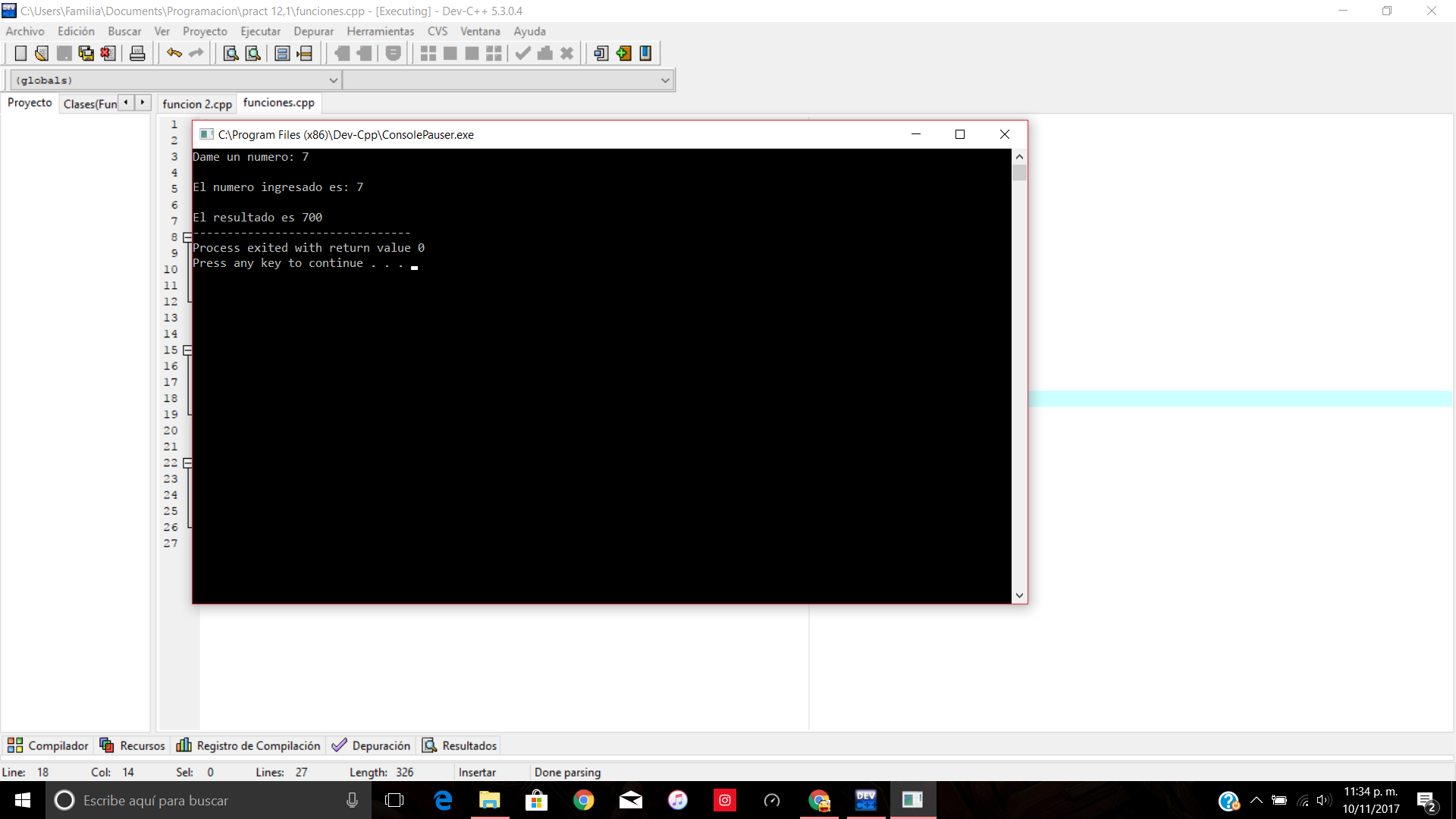
{

funcion1();

printf("\nEl numero ingresado es: %d\n",x);

return x;

}



Conclusiones:

Lo visto en esta práctica nos servira mucho en futuros programas que realicemos ya que los podremos acortar y tener mejor control gracias a las funciones, aparte entendimos el funcionamiento de estas y vimos su eficiencia.