Universidad Nacional Autónoma de México

Facultad de Ingeniería

**Asignatura:**

Fundamentos de la Programación

Actividad Asíncrona 14 | Estructuras de Repetición

**Alumno:**

Sánchez Estrada Angel Isaac

**Profesor:**

Ing. Guadalupe Cruz Mendoza

Ing. Marco Antonio Martínez Quintana

**Fecha:**

2 de diciembre del 2020



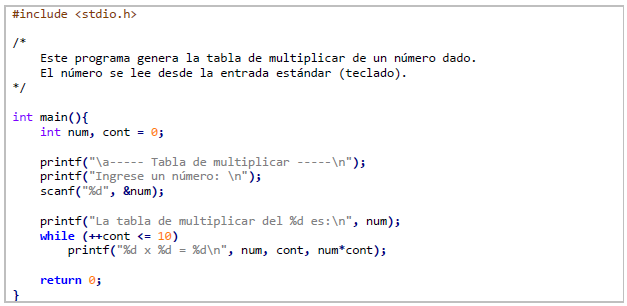
**Estructura de control repetitiva while**

La estructura repetitiva (o iterativa) while primero valida la expresión lógica y si ésta se cumple (es verdadera) procede a ejecutar el bloque de instrucciones de la estructura, el cual está delimitado por las llaves {}. Si la condición no se cumple se continúa el flujo normal del programa sin ejecutar el bloque de la estructura, es decir, el bloque se puede ejecutar de cero a ene veces. Su sintaxis es la siguiente:

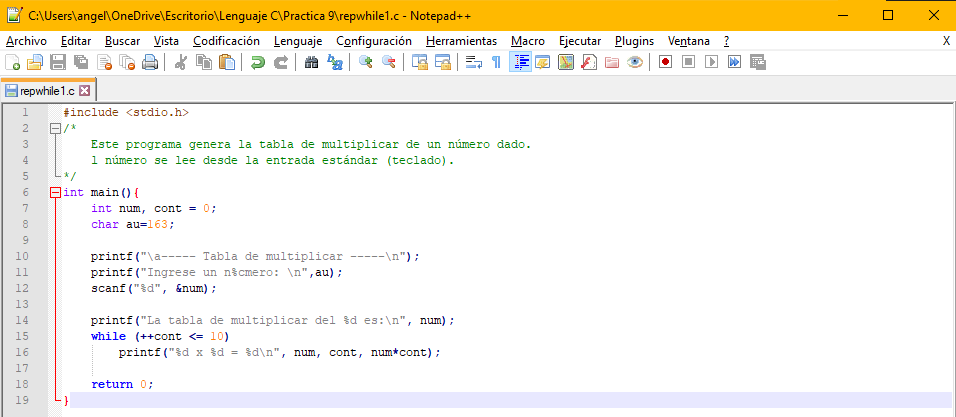


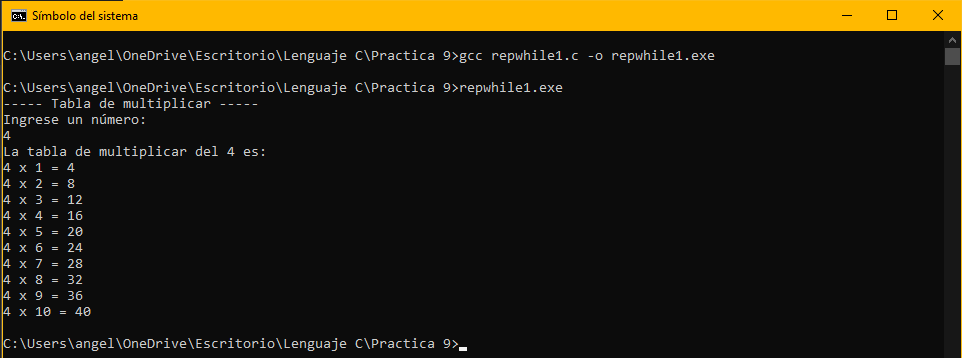
Si el bloque de código a repetir consta de una sola sentencia, entonces se pueden omitir las llaves.

* Codigo (estructura de repetición while)

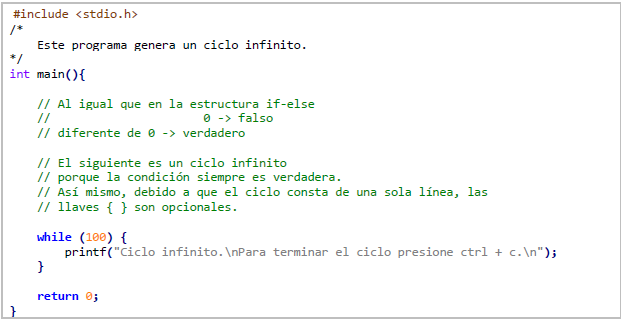


Resultado:

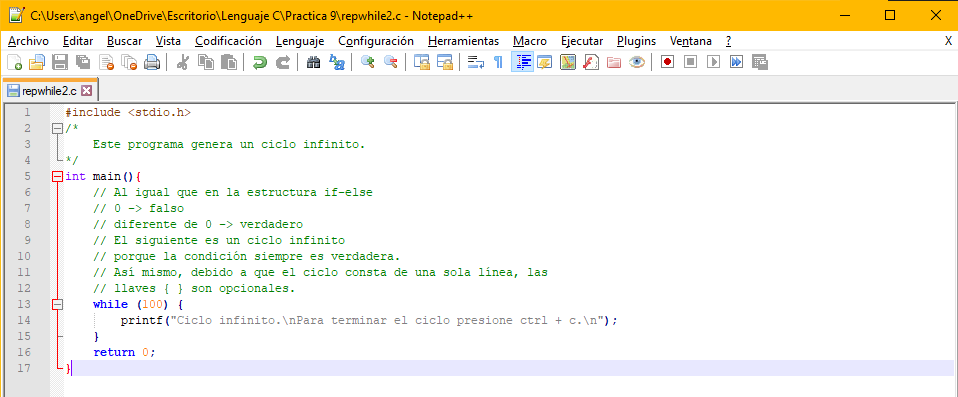


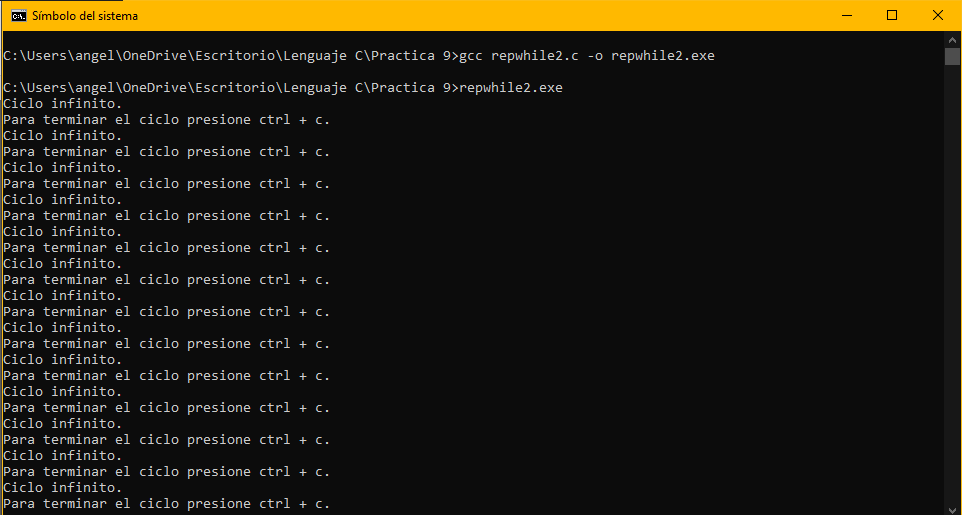


* Codigo (estructura de repetición while)



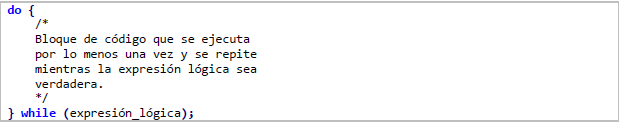
Resultado:





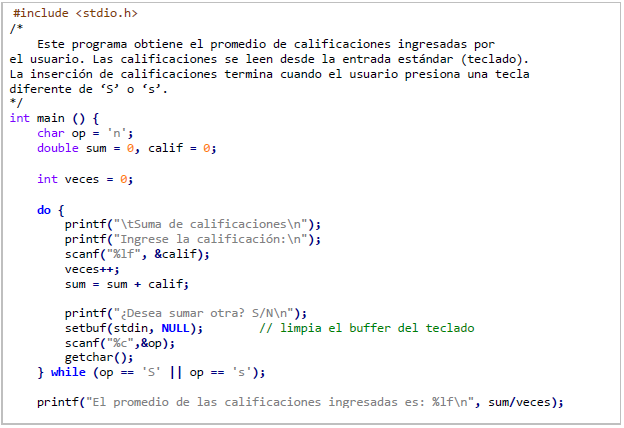
**Estructura de control repetitiva do-while**

do-while es una estructura cíclica que ejecuta el bloque de código que se encuentra dentro de las llaves y después valida la condición, es decir, el bloque de código se ejecuta de una a ene veces. Su sintaxis es la siguiente:



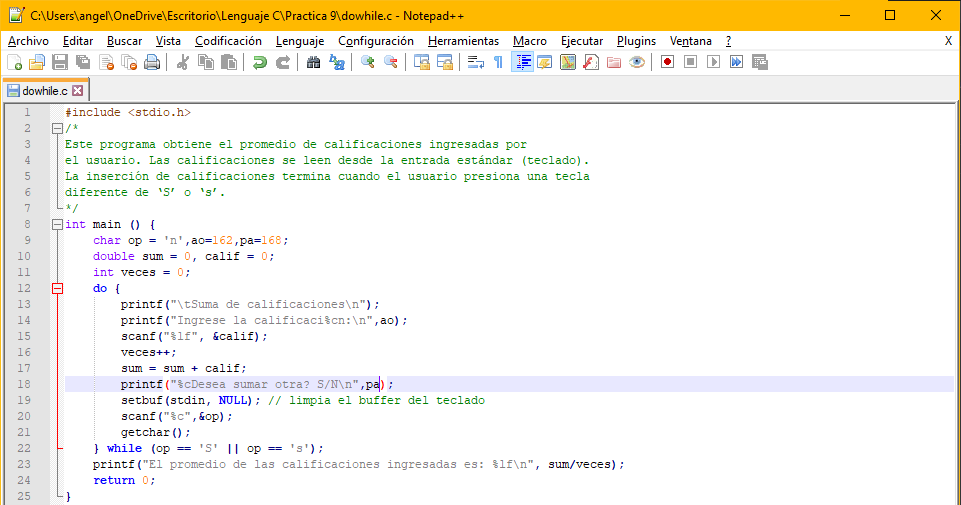
Si el bloque de código a repetir consta de una sola sentencia, entonces se pueden omitir las llaves. Esta estructura de control siempre termina con el signo de puntuación ';'.

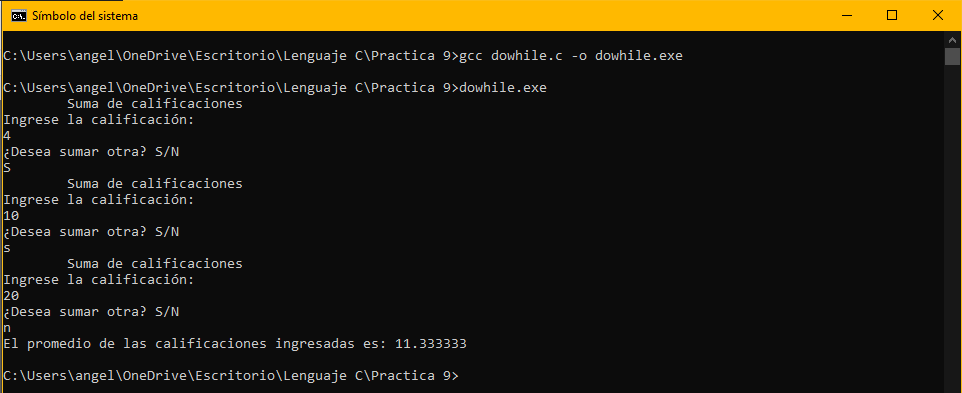
* Codigo (estructura de repetición do -while)



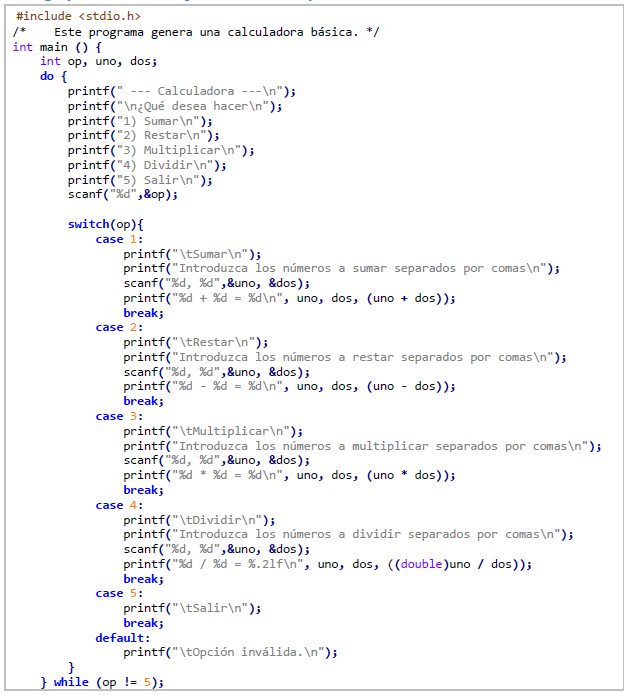


Resultado:



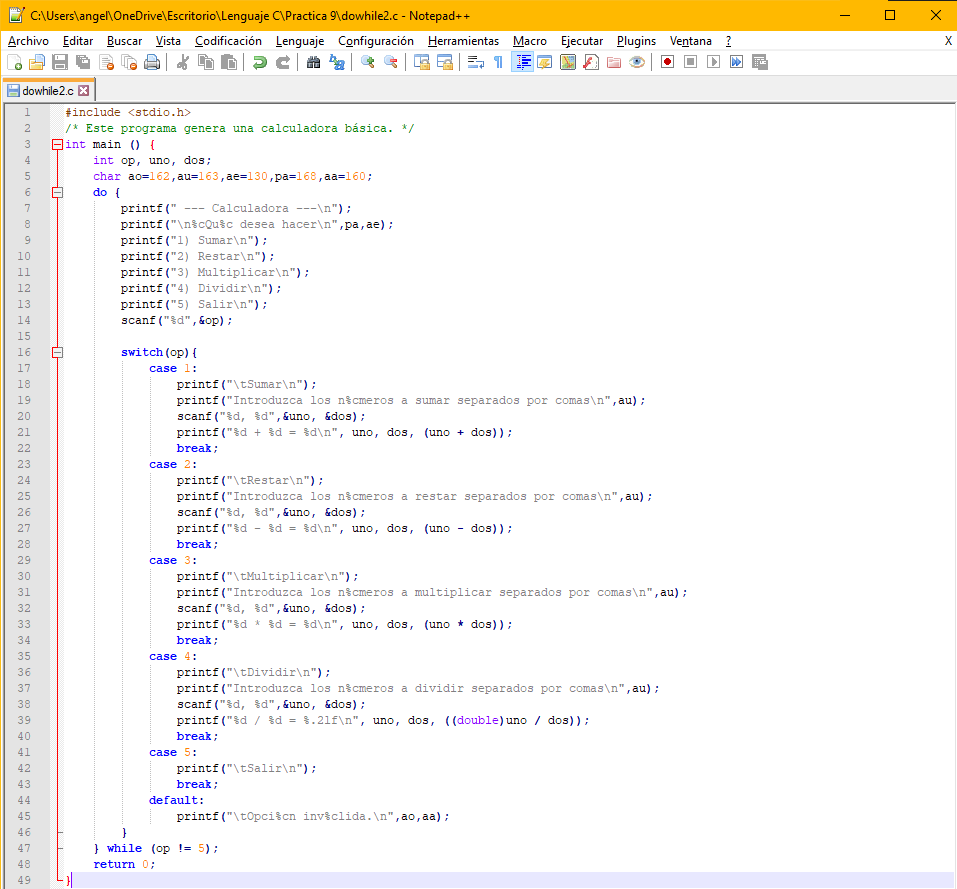


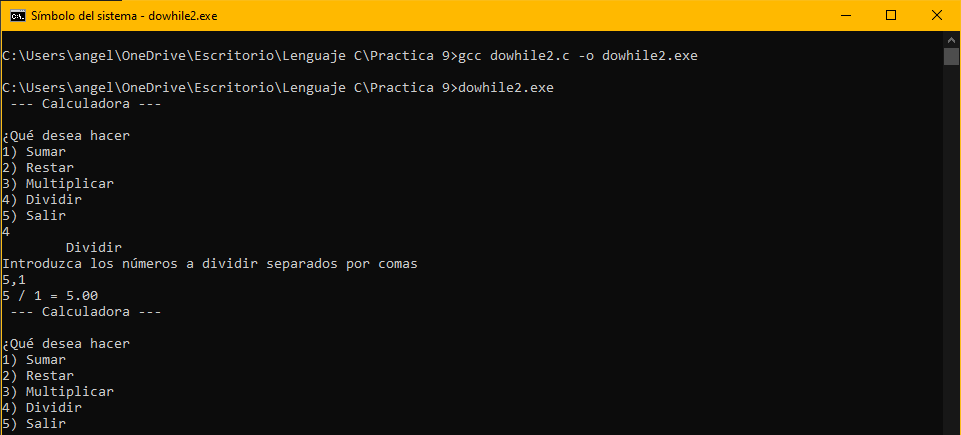
* Código (estructura de repetición do -while)





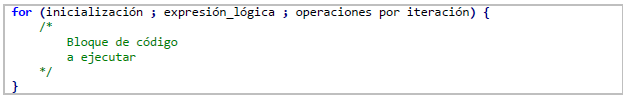
Resultado:





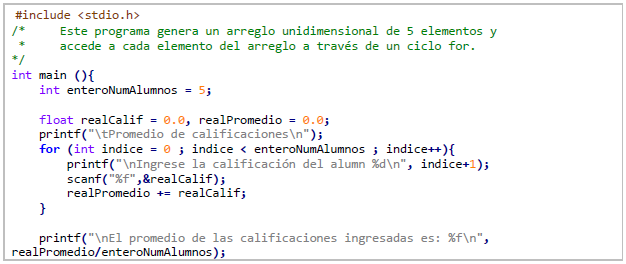
**Estructura de control de repetición for**

Lenguaje C posee la estructura de repetición for la cual permite realizar repeticiones cuando se conoce el número de elementos que se quiere recorrer. La sintaxis que generalmente se usa es la siguiente:



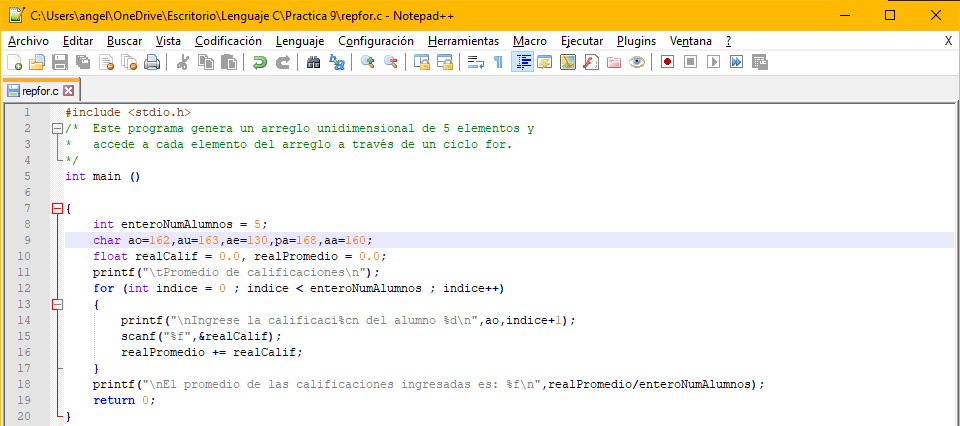
La estructura for ejecuta 3 acciones básicas antes o después de ejecutar el bloque de código. La primera acción es la inicialización, en la cual se pueden definir variables e inicializar sus valores; esta parte solo se ejecuta una vez cuando se ingresa al ciclo y es opcional. La segunda acción consta 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; esta parte es opcional. 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; esta parte también es opcional.

* Código (estructura de repetición for)





Resultado:





**Define**

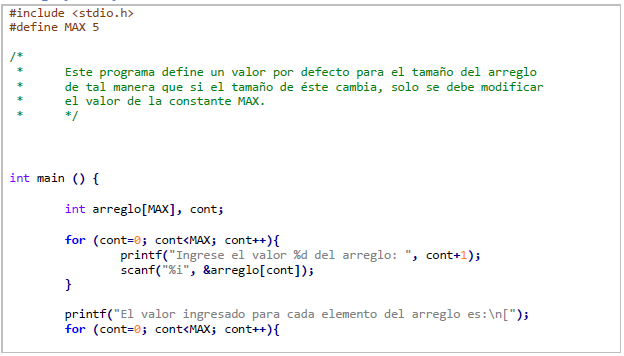
Las líneas de código que empiezan con # son directivas del preprocesador, el cual se encarga de realizar modificaciones en el texto del código fuente, como reemplazar un símbolo definido con #define por un parámetro o texto, o incluir un archivo en otro archivo con #include.

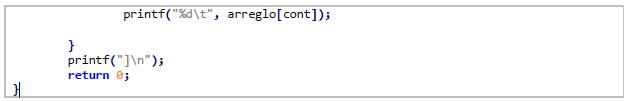
define permite definir constantes o literales; se les nombra también como constantes simbólicas. Su sintaxis es la siguiente:

#define <nombre> <valor>

Al definir la constante simbólica con #define, se emplea un nombre y un 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.

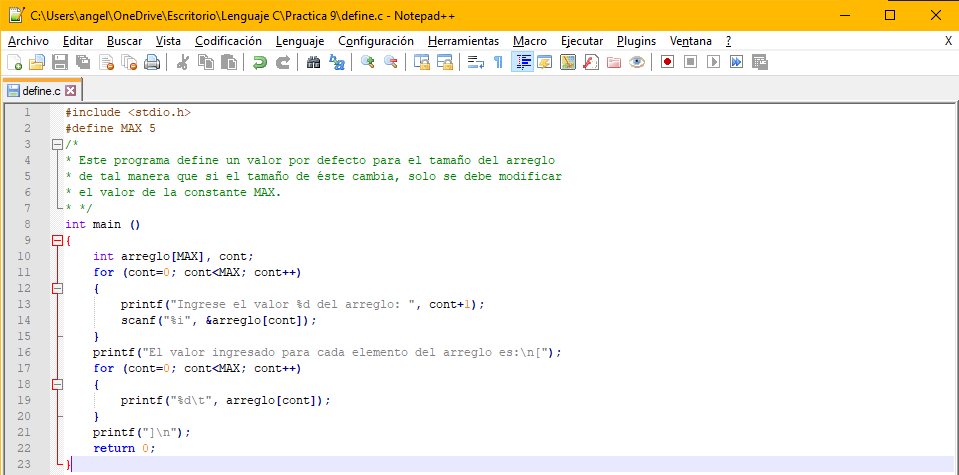
* Código (define)

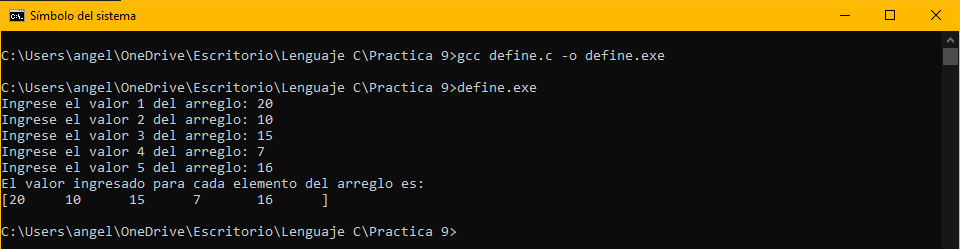




Cuando se compila el programa, se reemplazan la palabra MAX por el valor definido para la misma. Esto permite que, si el tamaño del arreglo cambia, solo se tiene que modificar el valor definido para MAX y en automático todos los arreglos y el recorrido de los mismos adquieren el nuevo valor (Mientras se use MAX para definir el o los arreglos y para realizar los recorridos).

Resultado:

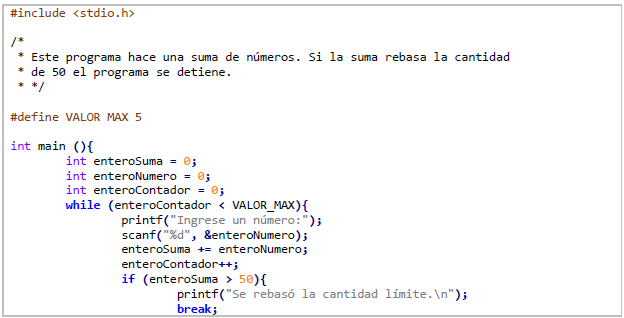


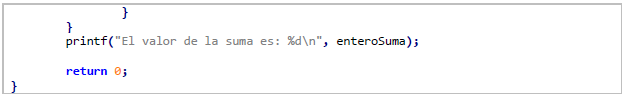


**Break**

Algunas veces es conveniente tener la posibilidad de abandonar un ciclo. La proposición **break** proporciona una salida anticipada dentro de una estructura de repetición, tal como lo hace en un switch. Un *break* provoca que el ciclo que lo encierra termine inmediatamente.

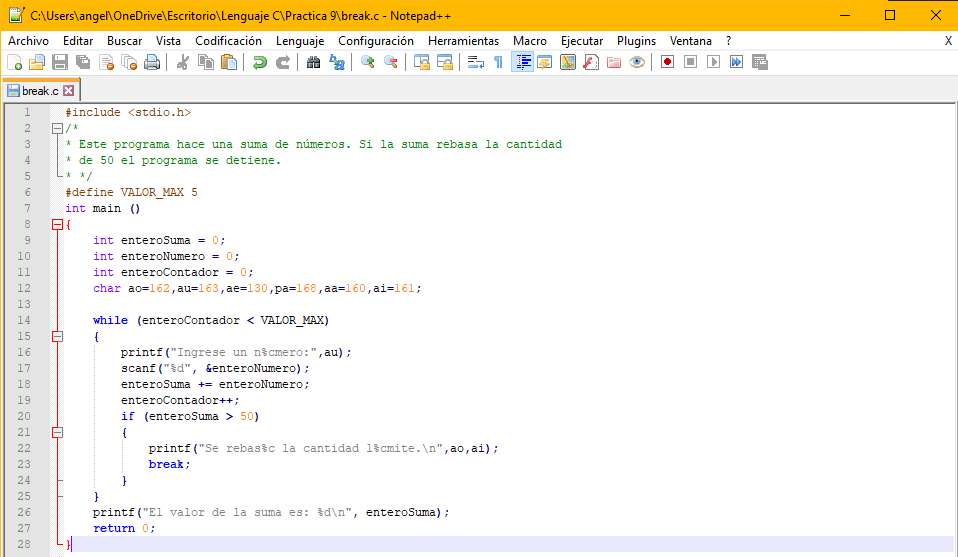
* Código (Break)

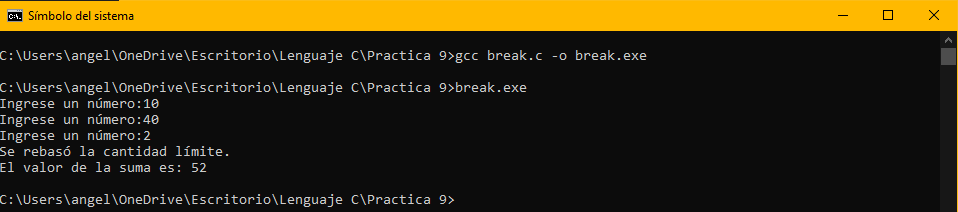




Cuando se compila el programa, MAX se sustituye por 5.

Resultado:



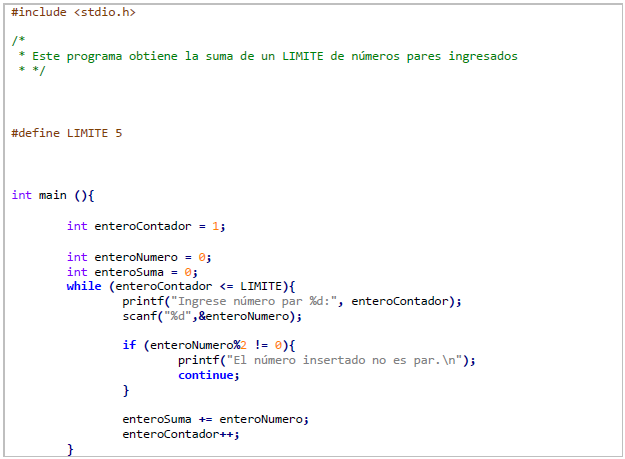


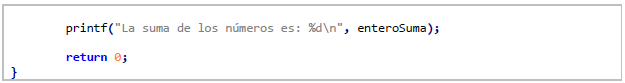
**Continue**

La proposición **continue** provoca que inicie la siguiente iteración del ciclo de repetición

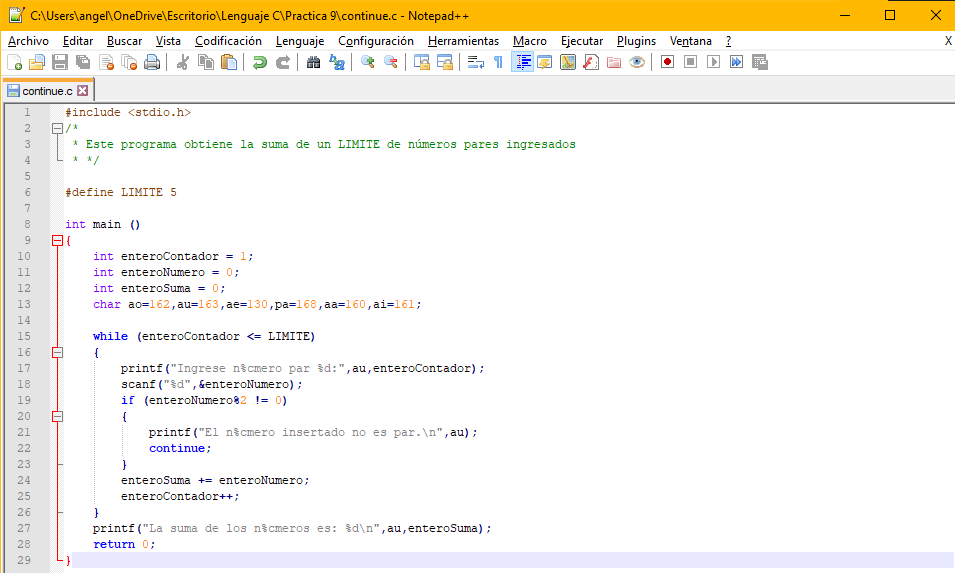
que la contiene.

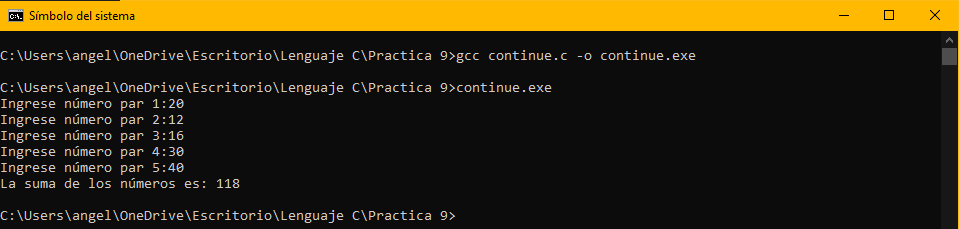
* Código (continue)





Resultado:





**Estructuras de repetición**

Las estructuras de repetición nos permiten como su nombre lo indica, repetir una o más instrucciones, ya sea un número determinado de veces o mientras se cumpla una condición.

* for
* while
* do-while

**Ciclo for-gauss.c**

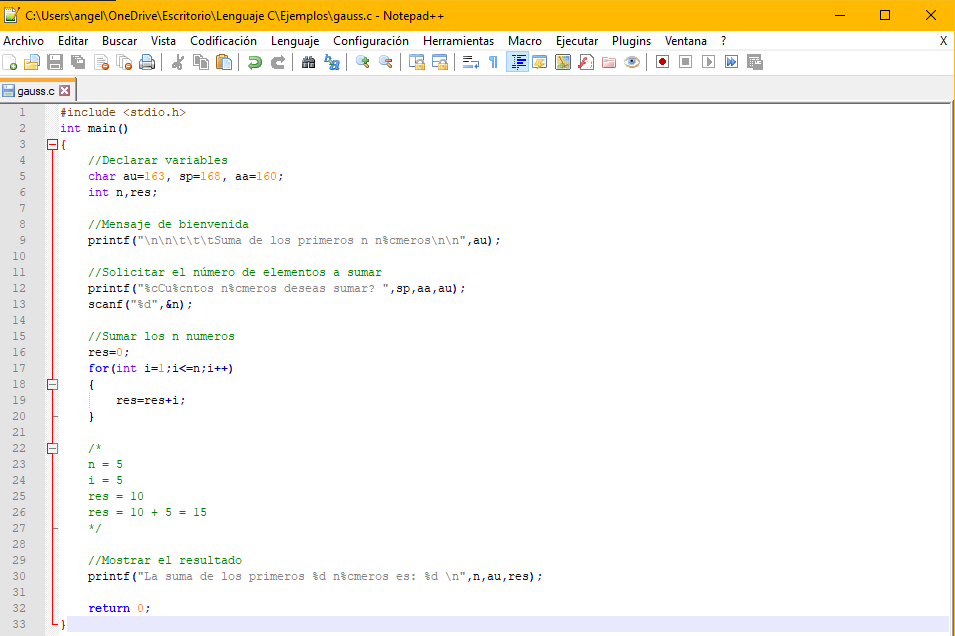
Nos permite repetir una o más instrucciones un determinado número de veces.

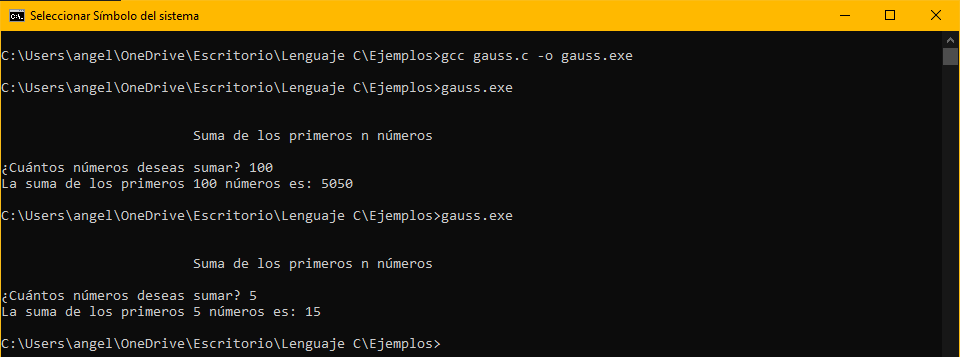
for(inicialización; expresión\_lógica; operaciones\_por\_iteración)

{

// Instrucciones a repetir

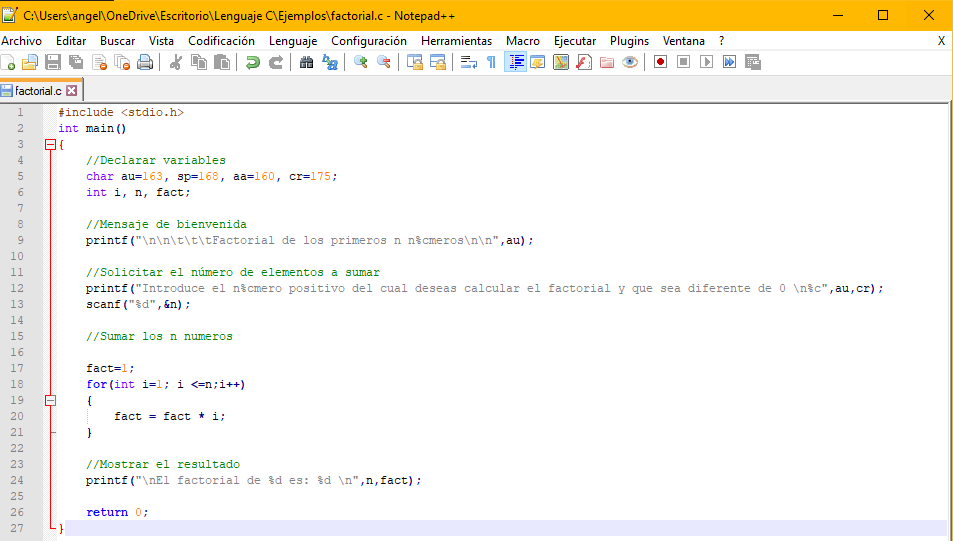
}

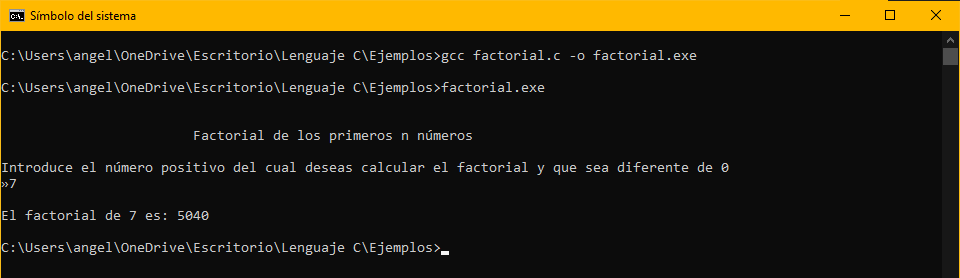




**Ejercicio 4**

**Programa que calcula la factorial de un numero con for**





**Ciclo while**

Esta estructura de repetición nos permite validar una condición antes de realizar el ciclo.

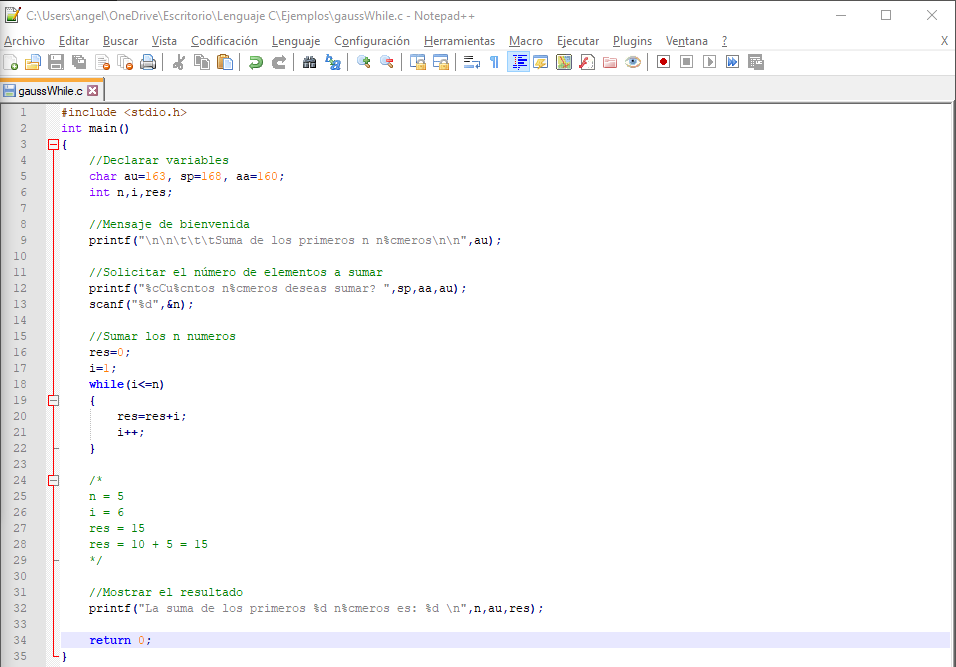
while(condición o condiciones)

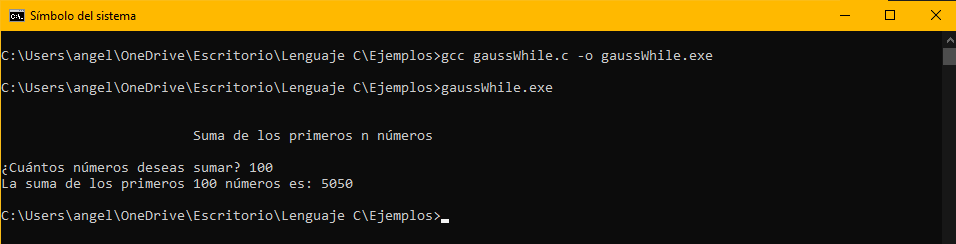
{

//Instrucciones a repetir

}

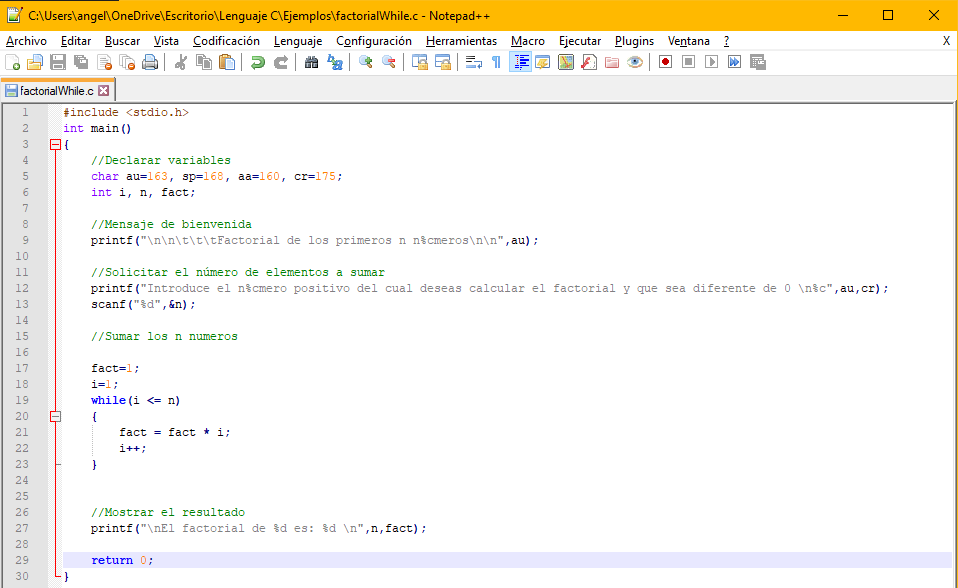
**Ciclo while-gaussWhile.c**

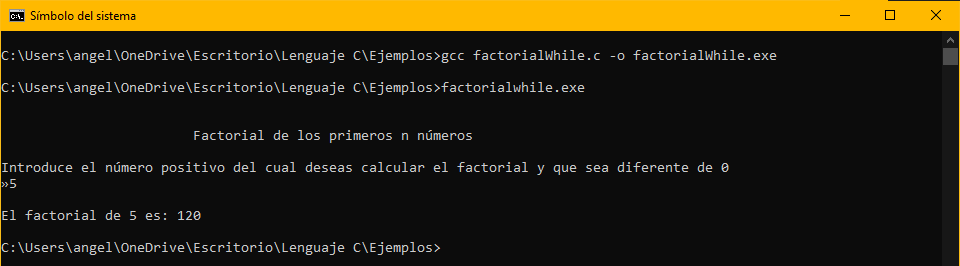




**Ejercicio 5**

**Programa que calcula la factorial de un numero con While**





**Ciclo do-while**

Este ciclo es muy utilizado para la generación de menús, ya que nos permite repetirlo tantas veces queramos mientras no se dé la opción salir. Y algo peculiar es que siempre se ejecuta al menos una vez.

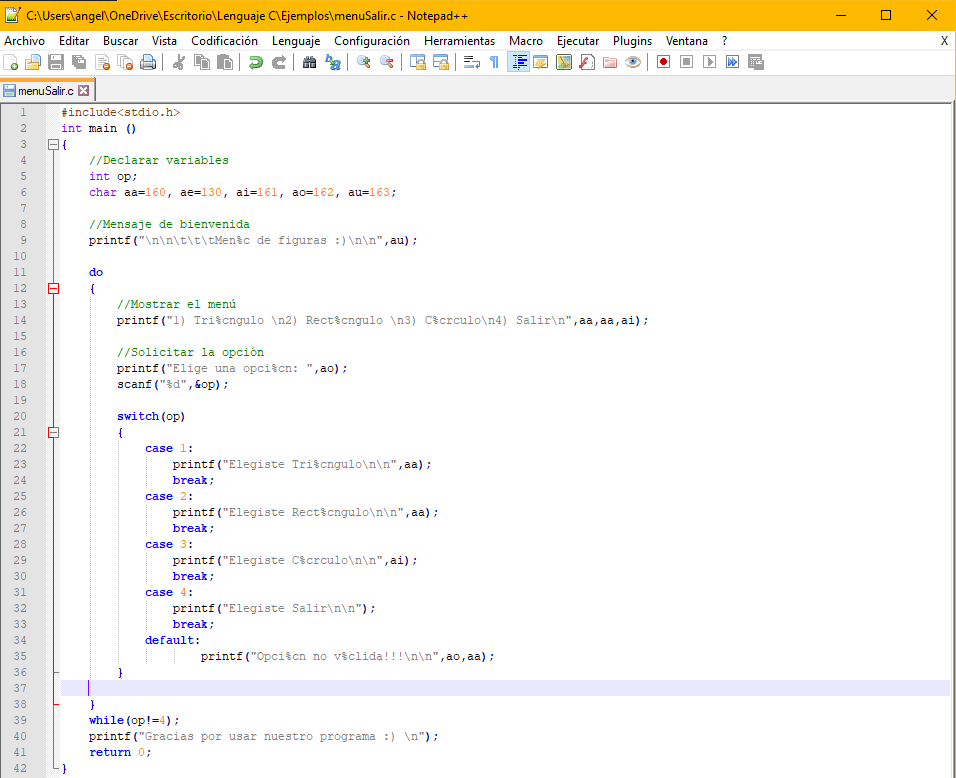
do

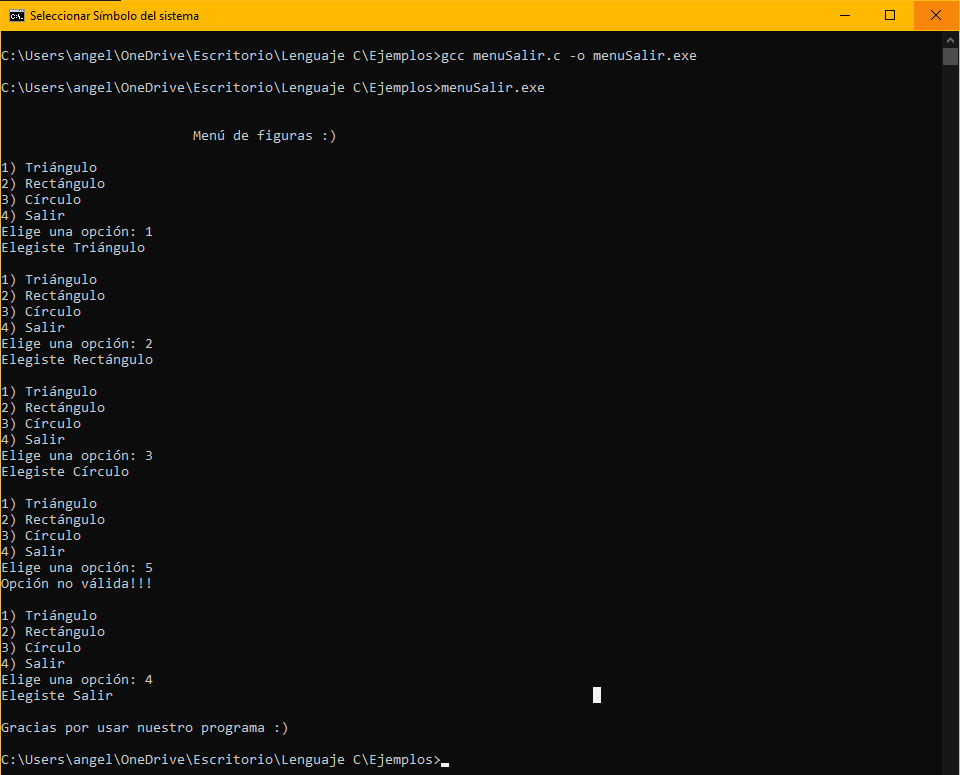
{

//Instrucciones a repetir

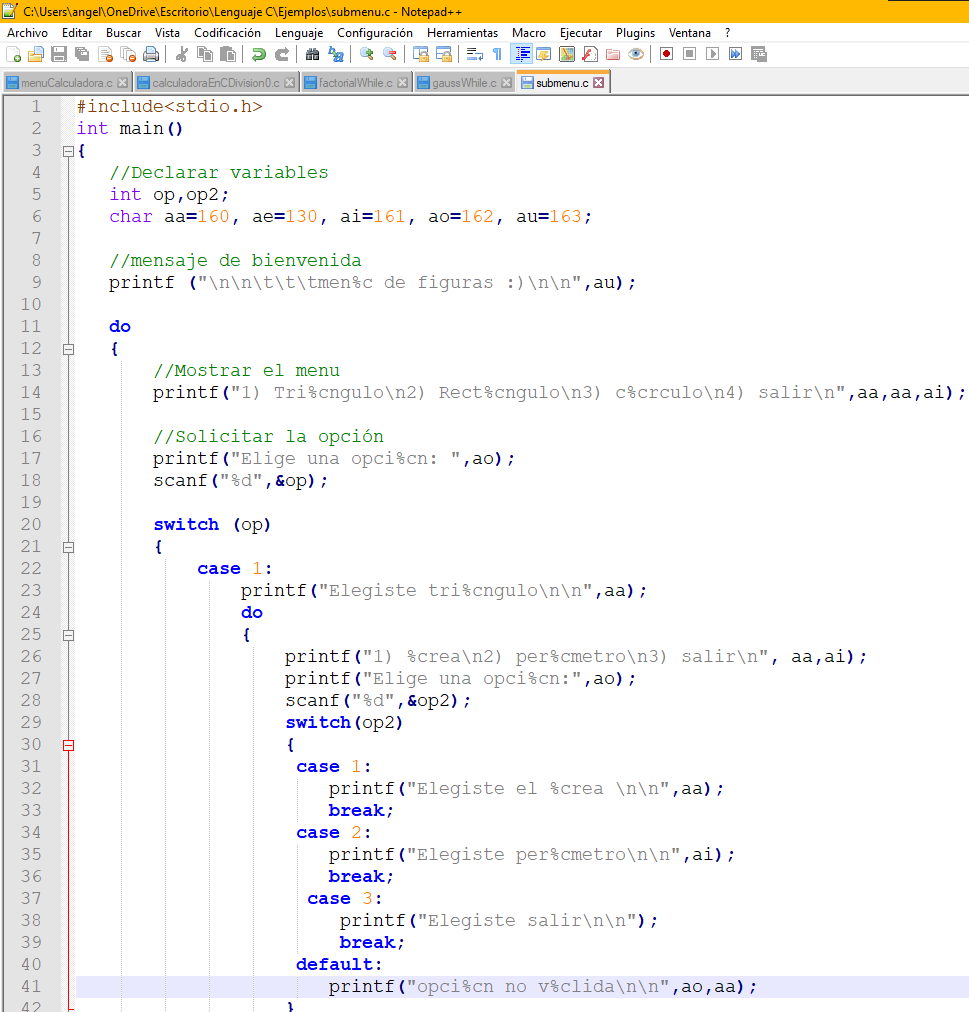
} while (condición o condiciones);

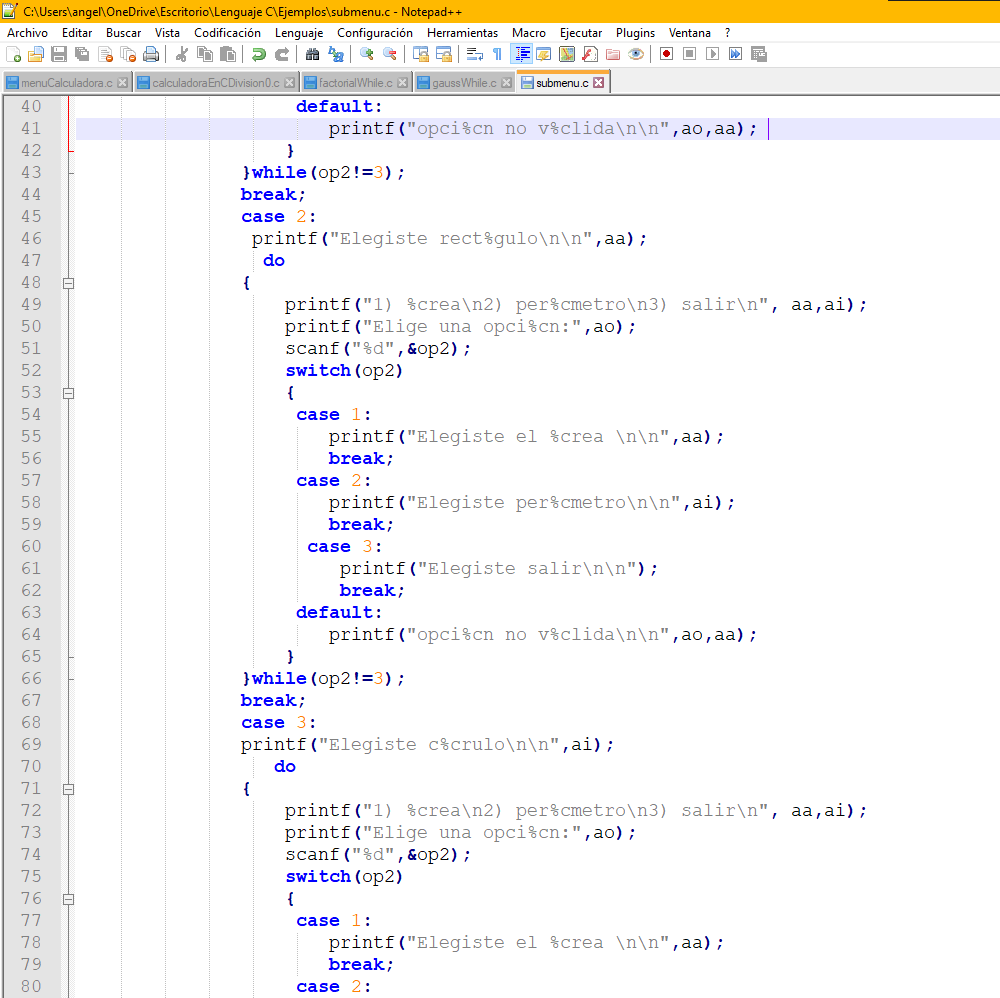
**Ciclo do-while-menuSalir.c**

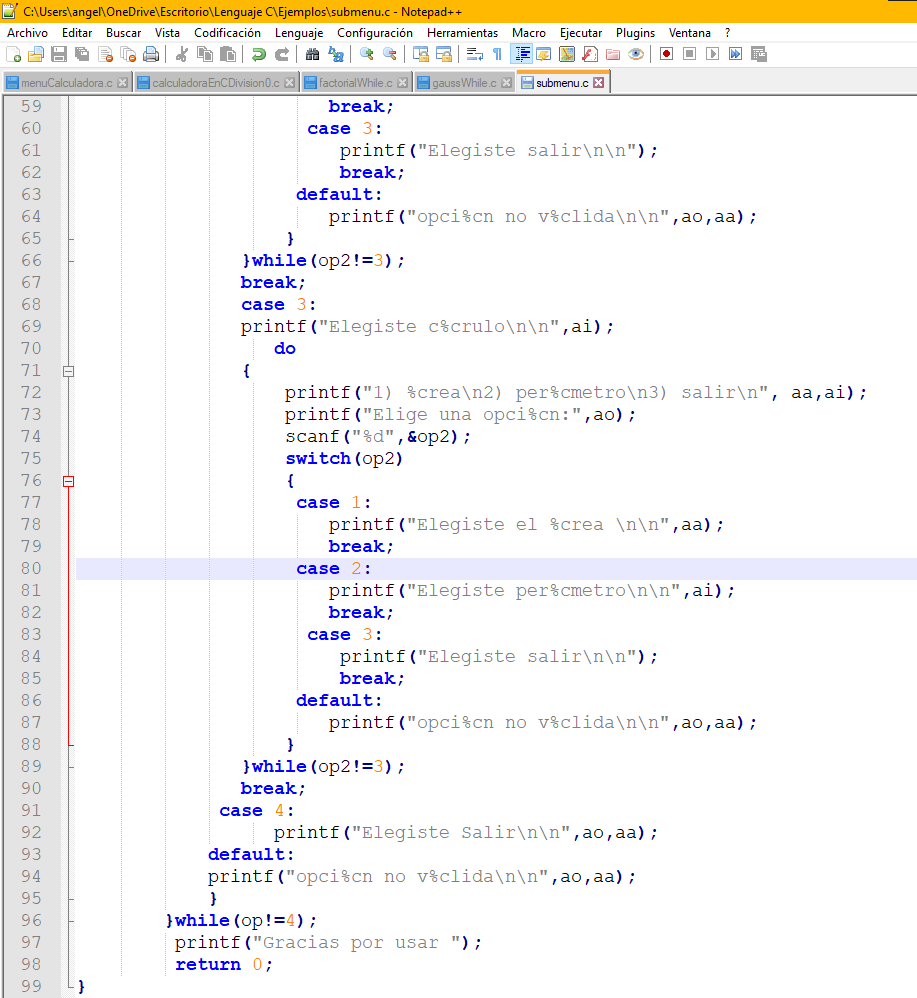


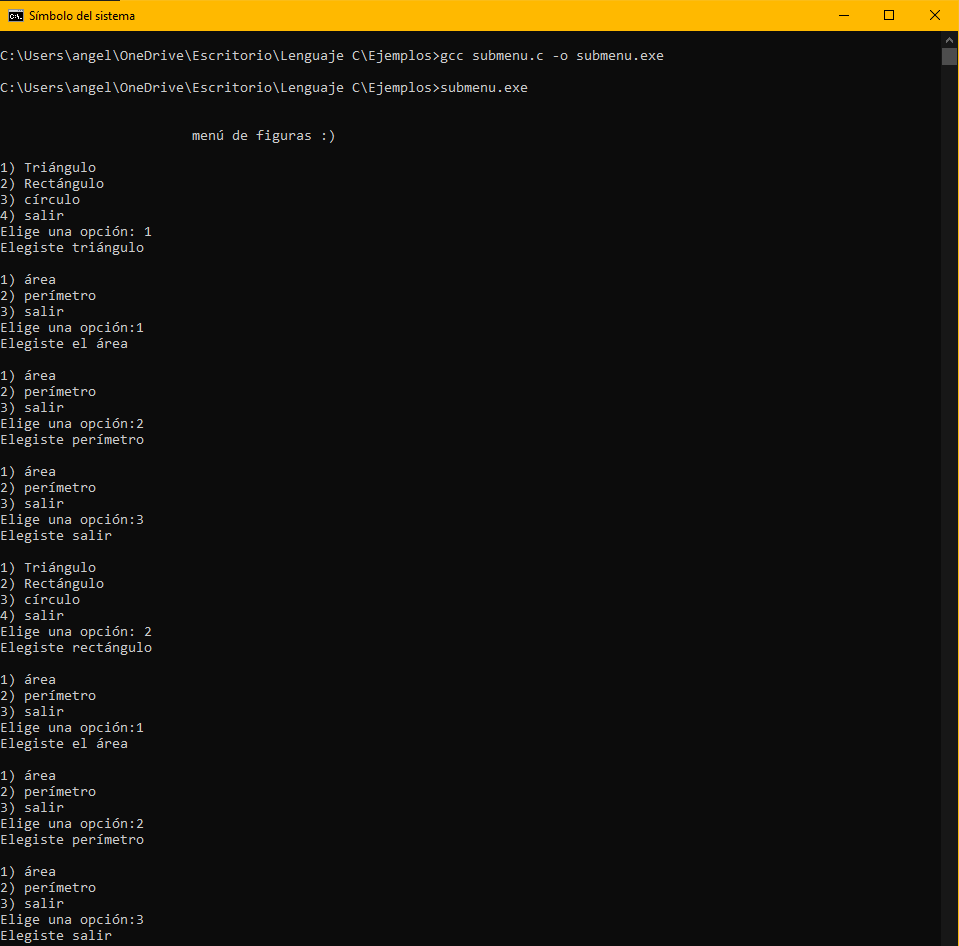


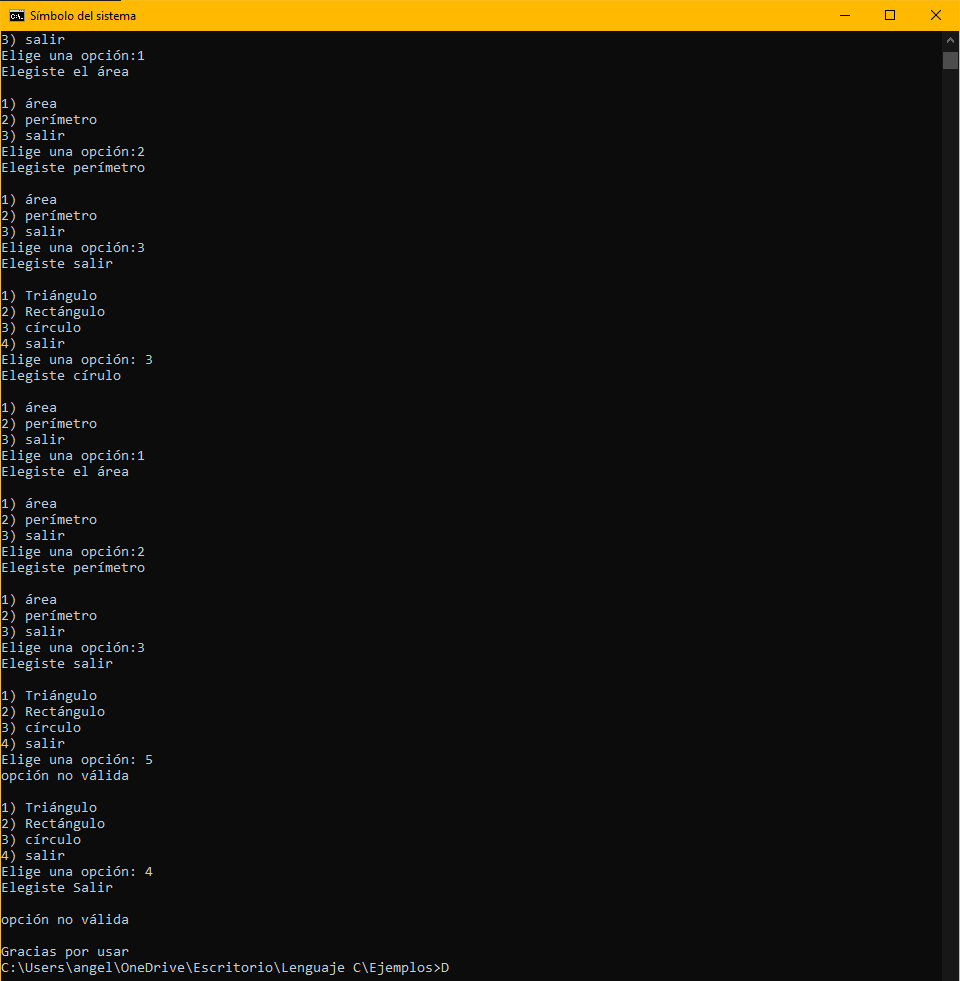
**Ciclo do-while-subMenu.c**







****

****

**Calculadora con opción de salir**

#include<stdio.h>

int main ()

{

//Declarar variables

int op,res,n1,n2,div,mod,fact,n,i;

char aa=160, ae=130, ai=161, ao=162, au=163, sp=168, cr=175;

//Mensaje de bienvenida

printf("\n\n\t\t\tBienvenidos a la Calculadora Maestra :)\n\n",au);

do

{

//Mostrar el menú

printf("\n1) Suma \n2) Resta \n3) Multiplicaci%cn \n4) Divisi%cn \n5) M%cdulo \n6) Factorial \n7) Suma de los primeros n n%cmeros \n8) Salir\n",ao,ao,ao,au);

//Solicitar la opción

printf("Elige una opci%cn: ",ao);

scanf("%d",&op);

switch(op)

{

case 1:

printf("Dame 2 n%cmeros separados por coma: ",au);

scanf("%i,%i",&n1,&n2);

res=n1+n2;

printf("La suma de %d y %d es: %d \n",n1,n2,res);

break;

case 2:

printf("Dame 2 n%cmeros separados por coma: ",au);

scanf("%i,%i",&n1,&n2);

res=n1-n2;

printf("La resta de %d y %d es: %d \n",n1,n2,res);

break;

case 3:

printf("Dame 2 n%cmeros separados por coma: ",au);

scanf("%i,%i",&n1,&n2);

res=n1\*n2;

printf("La multiplicaci%cn de %d y %d es: %d \n",ao,n1,n2,res);

break;

case 4:

printf("Dame 2 n%cmeros separados por coma: ",au);

scanf("%i,%i",&n1,&n2);

if(n2==0)

{

printf("La divisi%cn de %d y %d es: indefinida\n",ao,n1,n2);

}

else

{

div=n1/n2;

printf("La divisi%cn de %d y %d es: %d \n",ao,n1,n2,div);

}

break;

case 5:

printf("Dame 2 n%cmeros separados por coma: ",au);

scanf("%i,%i",&n1,&n2);

if(n2==0)

{

printf("El m%cdulo de %d y %d es: indefinida \n",ao,n1,n2);

}

else

{

mod=n1%n2;

printf("El m%cdulo de %d y %d es: %d \n",ao,n1,n2,mod);

}

break;

case 6:

printf("Introduce el n%cmero positivo del cual deseas calcular el factorial y que sea diferente de 0 \n%c",au,cr);

scanf("%d",&n);

fact=1;

i=1;

while(i <= n)

{

fact = fact \* i;

i++;

}

printf("\nEl factorial de %d es: %d \n",n,fact);

break;

case 7:

printf("%cCu%cntos n%cmeros deseas sumar? ",sp,aa,au);

scanf("%d",&n);

res=0;

i=1;

while(i<=n)

{

res=res+i;

i++;

}

printf("La suma de los primeros %d n%cmeros es: %d \n",n,au,res);

break;

case 8:

printf("Elegiste Salir\n\n");

break;

default:

printf("Opci%cn no v%clida!!!\n\n",ao,aa);

}

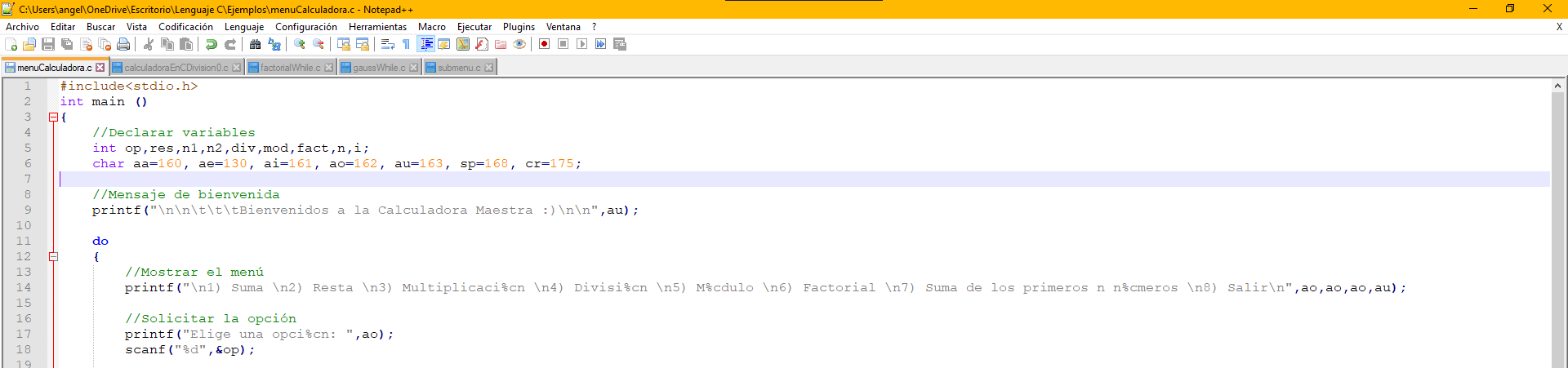
}

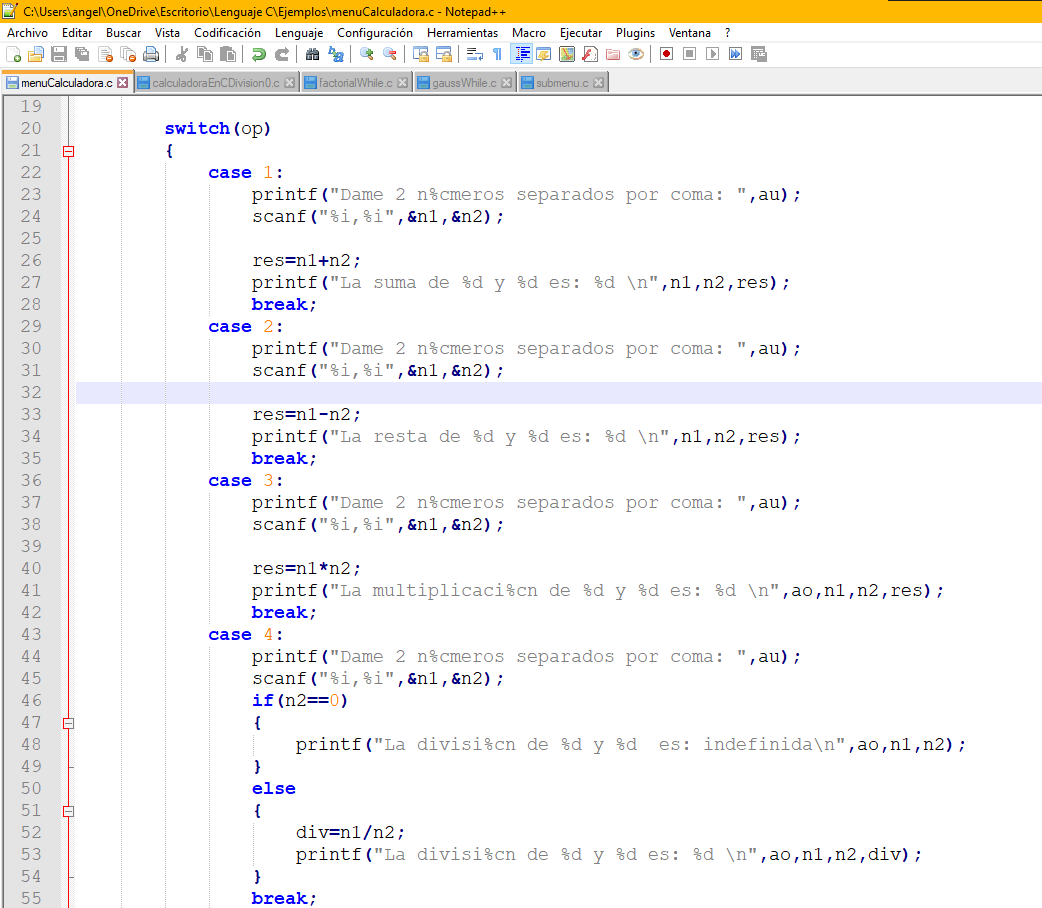
while(op!=8);

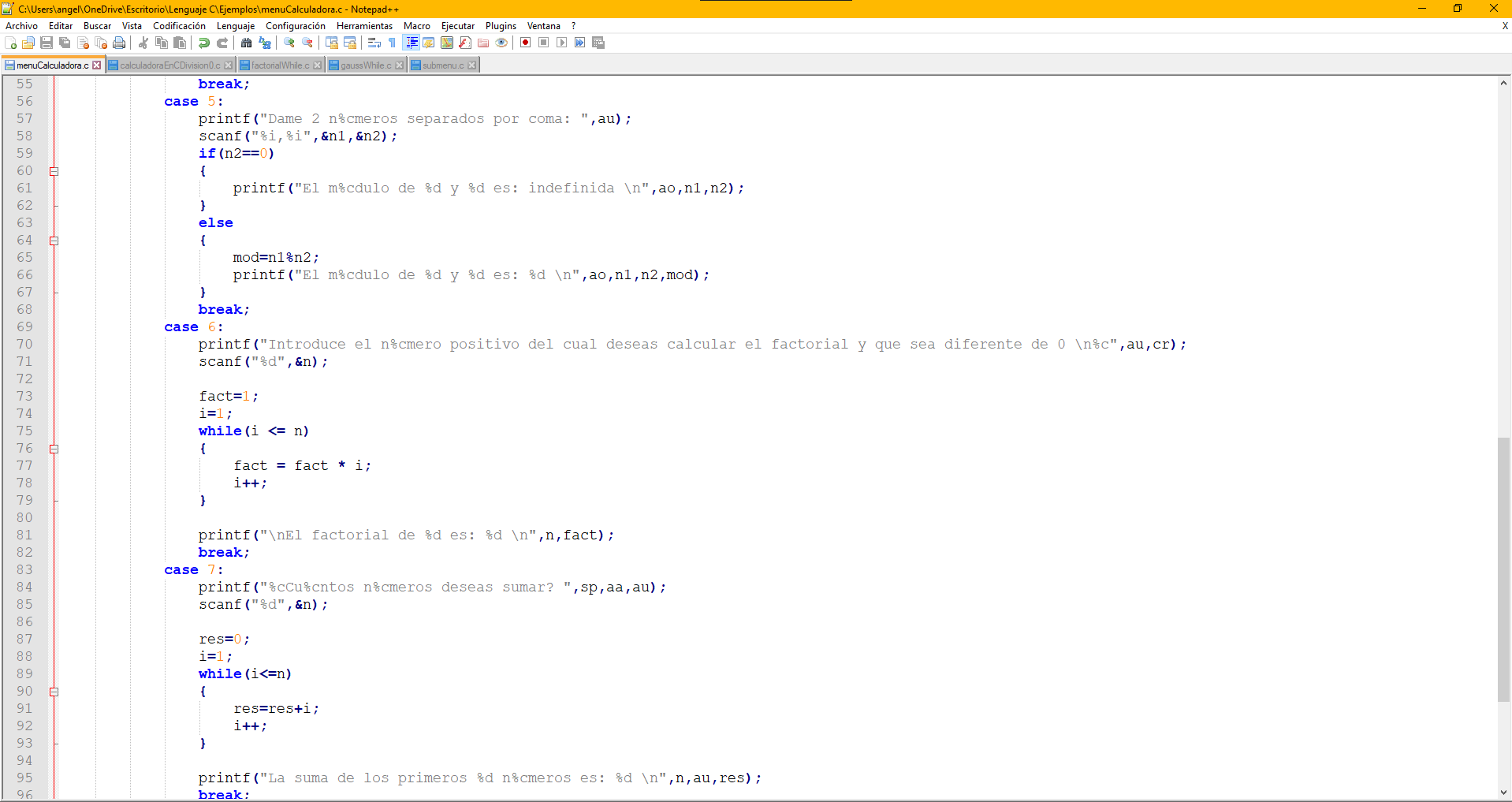
printf("Gracias por usar nuestro programa :) \n");

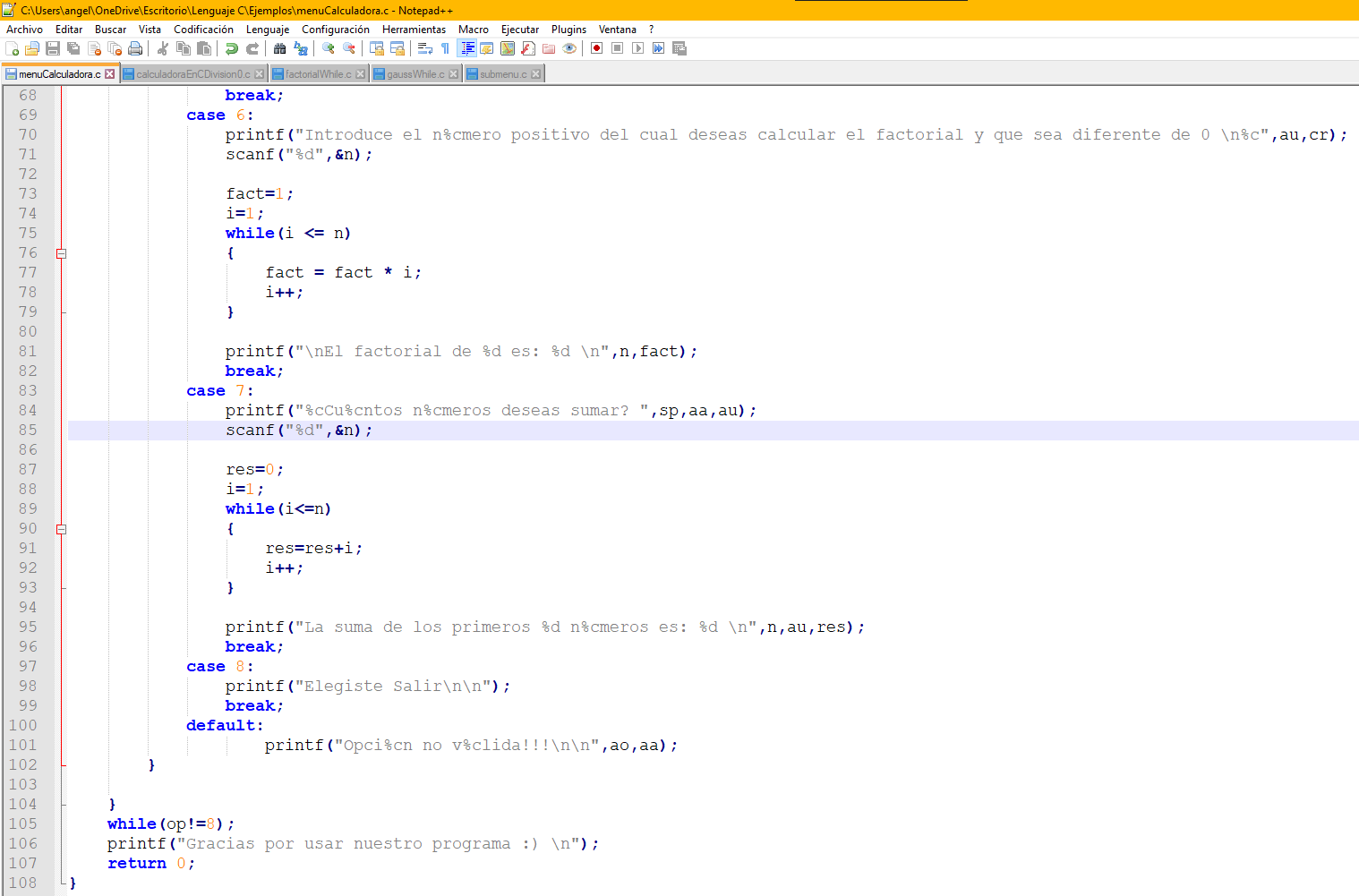
return 0;

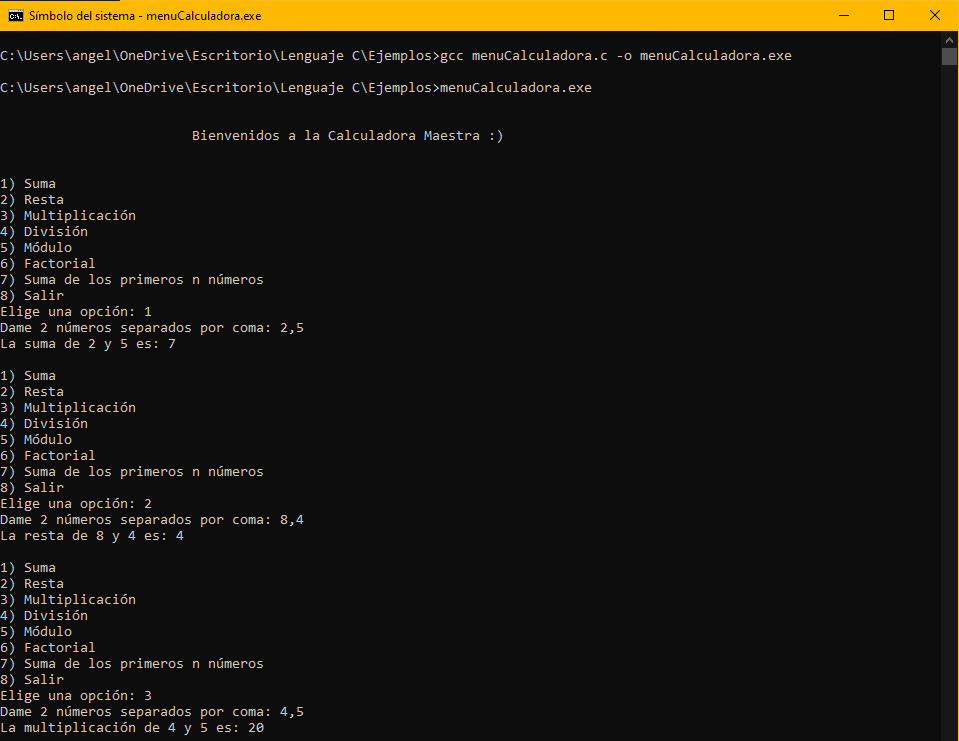
}

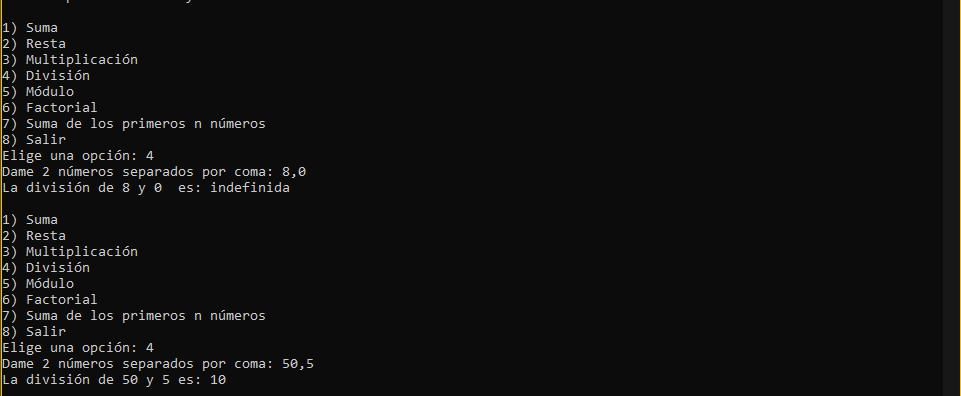




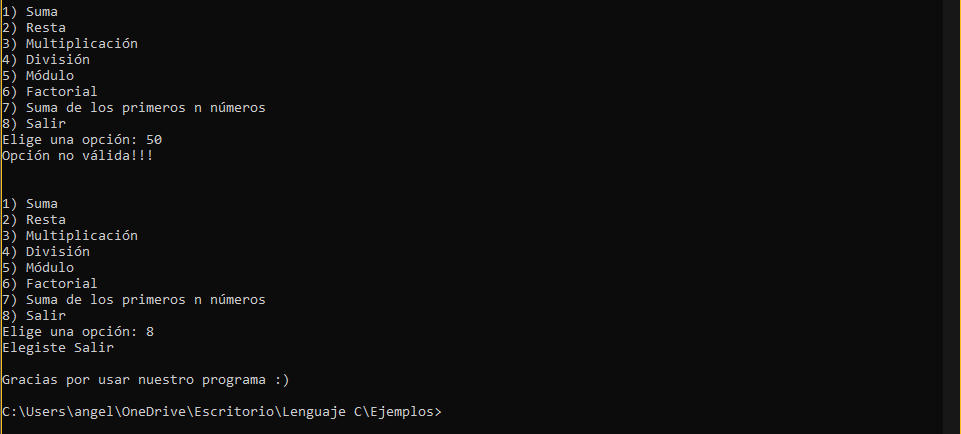












**Cibergrafía o Referencias Electrónicas:**

* Cursos en Línea. (s. f.). Soluciones MyL. Recuperado 03 de diciembre de 2020, de <http://solucionesmyl.com/cursos/lenguaje_c/scanf.html>
* Lenguaje C. (s. f.). Pai Torneiros (Programador de Aplicaciones Informáticas). Recuperado 03 de diciembre de 2020, de http://pai-torneiros.blogspot.com/2011/08/tabla-ascii-en-lenguaje-c.html