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

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Garcia Morales Karina |
| *Asignatura:* | Fundamentos de Programación |
| *Grupo:* | 22 |
| *No de Práctica(s):* | Practica 9 |
| *Integrante(s):* | Ramos Orozco Daniel |
|  |  |
|  |  |
| *Semestre:* | 2018-2 |
| *Fecha de entrega:* | 24-04-2018 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

**Objetivo:**

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

Las estructuras de repetición:

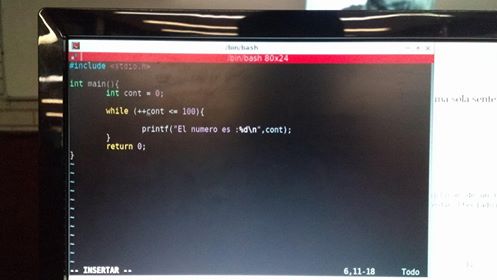
También llamadas estructuras cíclicas, iterativas o de bucles. Permiten ejecutar un conjunto de instrucciones de manera repetida (o cíclica) mientras que la expresión lógica a evaluar se cumpla (sea verdadera).

En lenguaje C existen tres estructuras de repetición: while, do-while y for. Las estructuras while y do-while son estructuras repetitivas de propósito general.

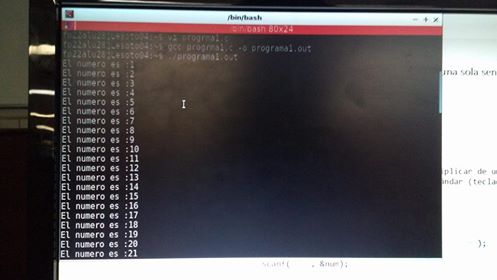
**Estructura de control repetitiva while**

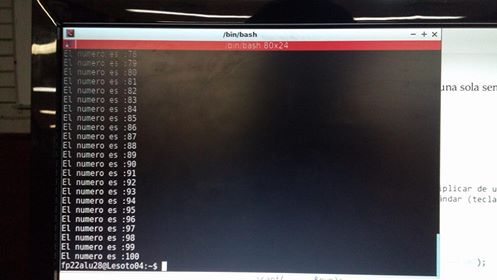
La estructura repetitiva (o iterativa) while primero valida la expresión lógica y si ésta se cumple (es verdadera) procede a ejecutar el bloque de instrucciones de la estructura, el cual está delimitado por las llaves {}. Si la condición no se cumple se continúa el flujo normal del programa sin ejecutar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

Para esta estructura utilizamos el siguiente código:



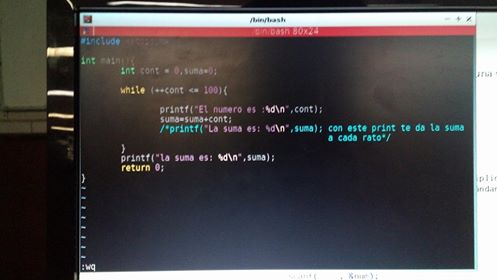
El cual lo compilamos y ejecutamos como siempre y nos dio el siguiente resultado:



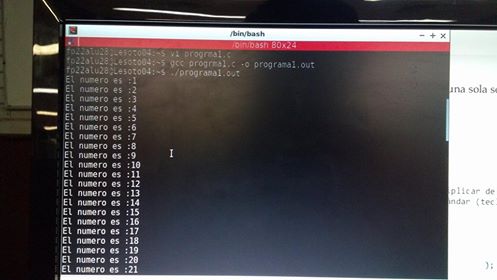


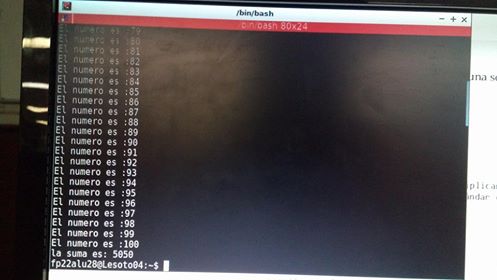
Lo que hace el ciclo es que te imprime los números del 1 al 100.

Después lo editamos para que ahora hiciera la suma de los números dados (1 al 100 que te los da el ciclo).

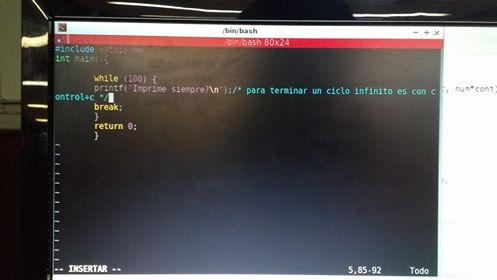


Lo compilamos y ejecutamos y nos dio casi lo mismo con el detalle que como lo editamos ahora nos dio la suma de todo.

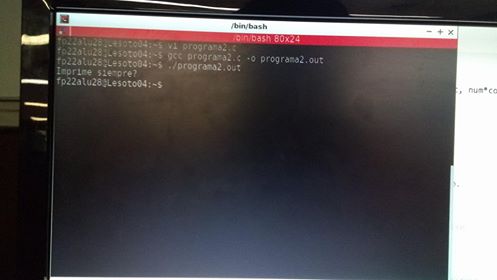




Para el siguiente ejemplo utilizamos el siguiente código:



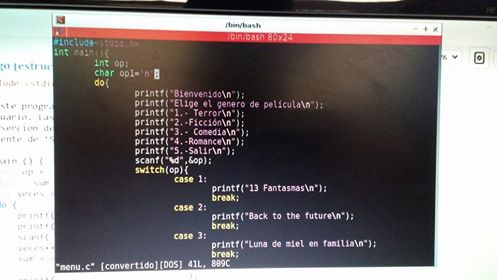
En este ejemplo nos damos cuenta que el ciclo se ejecuta siempre y si lo queremos parar debemos apretar “ctrl+c”, pero para que no pase le colocamos el “break” que ya habíamos visto que lo que hace es terminar el proceso que se desee.

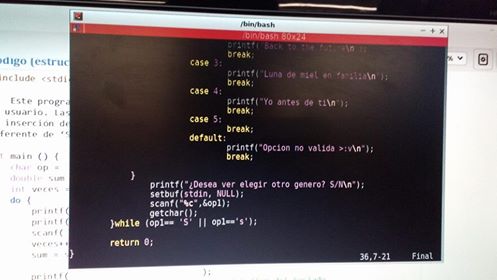


**Estructura de control repetitiva do-while**

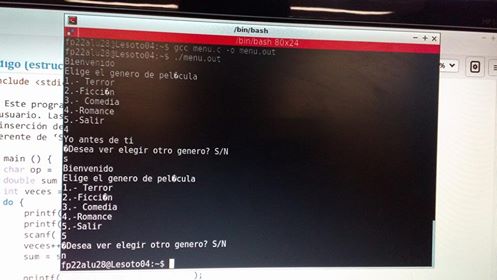
Do-while es una estructura cíclica que ejecuta el bloque de código que se encuentra dentro de las llaves y después valida la condición, es decir, el bloque de código se ejecuta de una a ene veces.

Para este ejemplo lo que hicimos es colocarle un ciclo do-while a nuestro código anteriormente hecho del menú y asi hacer que hasta el final nos pregunte si quiere ver otra opción o no y asi saber si se repite o si termina el ciclo.





Lo compilamos y ejecutamos como siempre:



Podemos apreciar que después de escoger un caso t de mostrarte el resultado, te pregunta si quieres ver otro genero y dependiendo de la respuesta repite o termina y todo es gracias principalmente a esta parte del ciclo:

do{

printf("¿Desea ver elegir otro genero? S/N\n");

setbuf(stdin, NULL);

scanf("%c",&op1);

getchar();

}while (op1== 'S' || op1=='s');

Ya que es el ciclo que repite hasta que tu decidas.

**Estructura de control de repetición for**

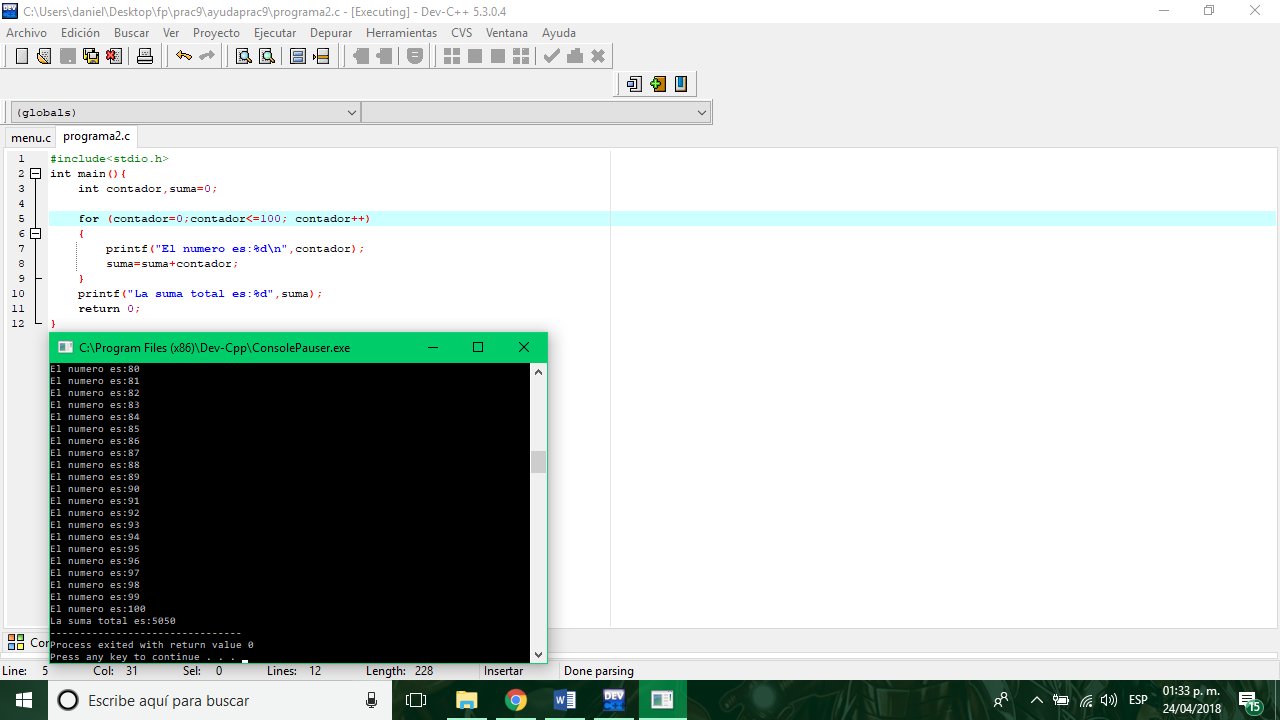
Lenguaje C posee la estructura de repetición for la cual permite realizar repeticiones cuando se conoce el número de elementos que se quiere recorrer.

for (inicialización ; expresión\_lógica ; operaciones por iteración) {

}

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.

Para este ejemplo que hicimos es programar el mismo ciclo que el que el ejemplo del “while” pero en ciclo “for” que lo que hace es contar del 1 al 100 y sumar todo los números de este rango.



Pero como no lo terminamos, nos comentó que lo acabáramos en casa.

**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.

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.

Para ejemplo de este puede ser:

**#define p printf**

**#define s scanf**

En estos define p y s se utilizarían para no estar escribiendo tanto

**#define Max 5** para darle un valor a algo y utilizarlo así en alguna condición

**Break**

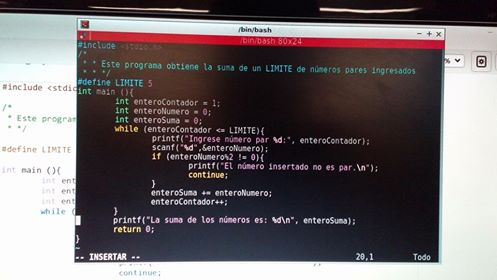
Algunas veces es conveniente tener la posibilidad de abandonar un ciclo. La proposición break proporciona una salida anticipada dentro de una estructura de repetición, tal como lo hace en un switch. Un break provoca que el ciclo que lo encierra termine inmediatamente.

El break ya lo habíamos visto para romper ciclos en los “swich” y tambien vimos que se podían hacer ciclos con los “if” que evaluaba una exprecion y si esta se iba a infinito con el break lo terminábamos.

**Continue**

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

Para este ejemplo vimos el código:

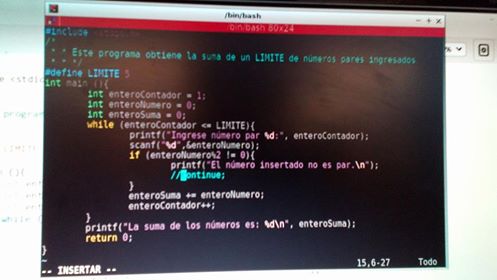


Lo compilamos y ejecutamos:

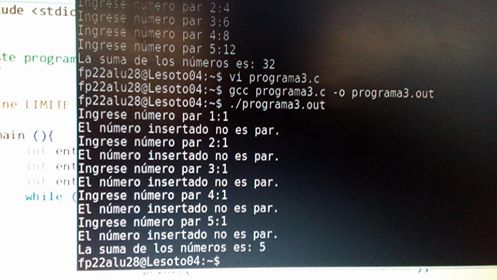


Y lo que hace es que con el “continue” de la condicion “if “es que si se cumple la condición con el “continue” se regresa al ciclo para volverlo hacer hasta que el ciclo no se cumpla y eso hace que el ciclo no continúe su camino.

Después lo editamos a continuación quitándole el “continue” para ver que efectivamente si hacia esto:



Lo compilamos y ejecutamos:

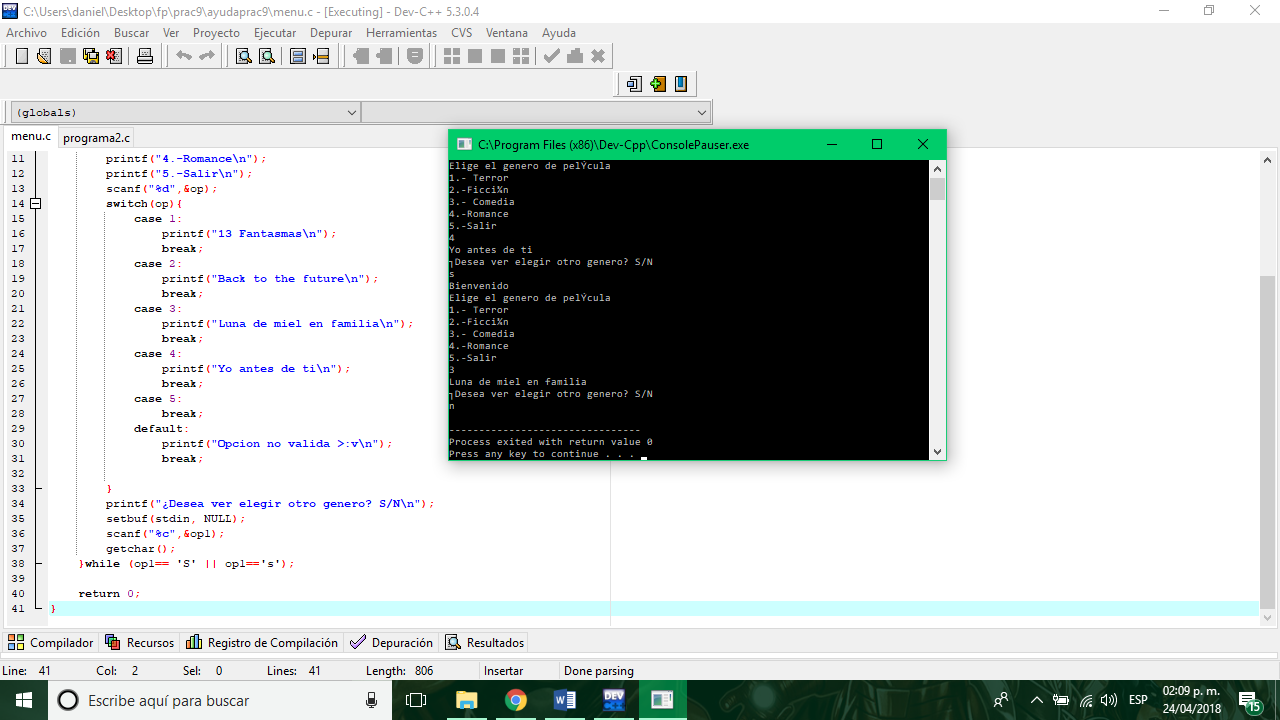


Y efectivamente como no tiene el “continue” que lo regrese continua con el cilo y eso hace que la condicion si no sea valida como se ve en la imagen.

Ejercicios propuestos

1.- En tu programa del switch para categorí­as de películas agrega el ciclo do-while

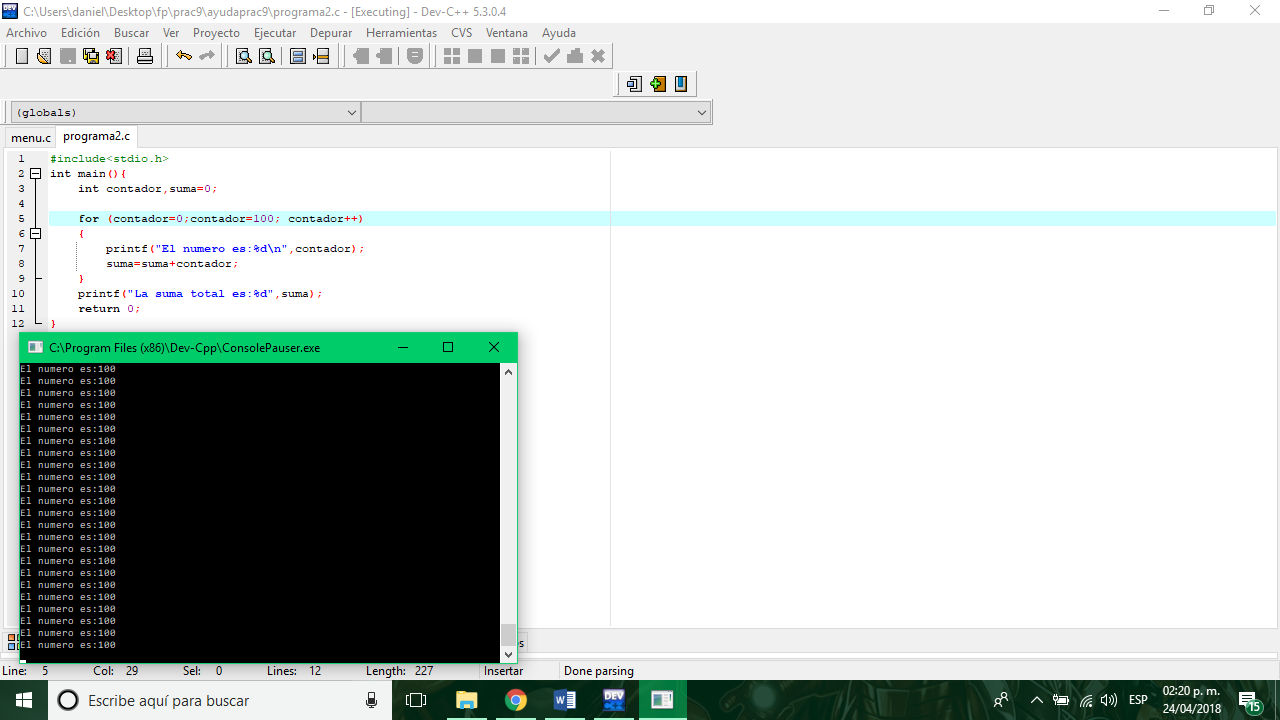
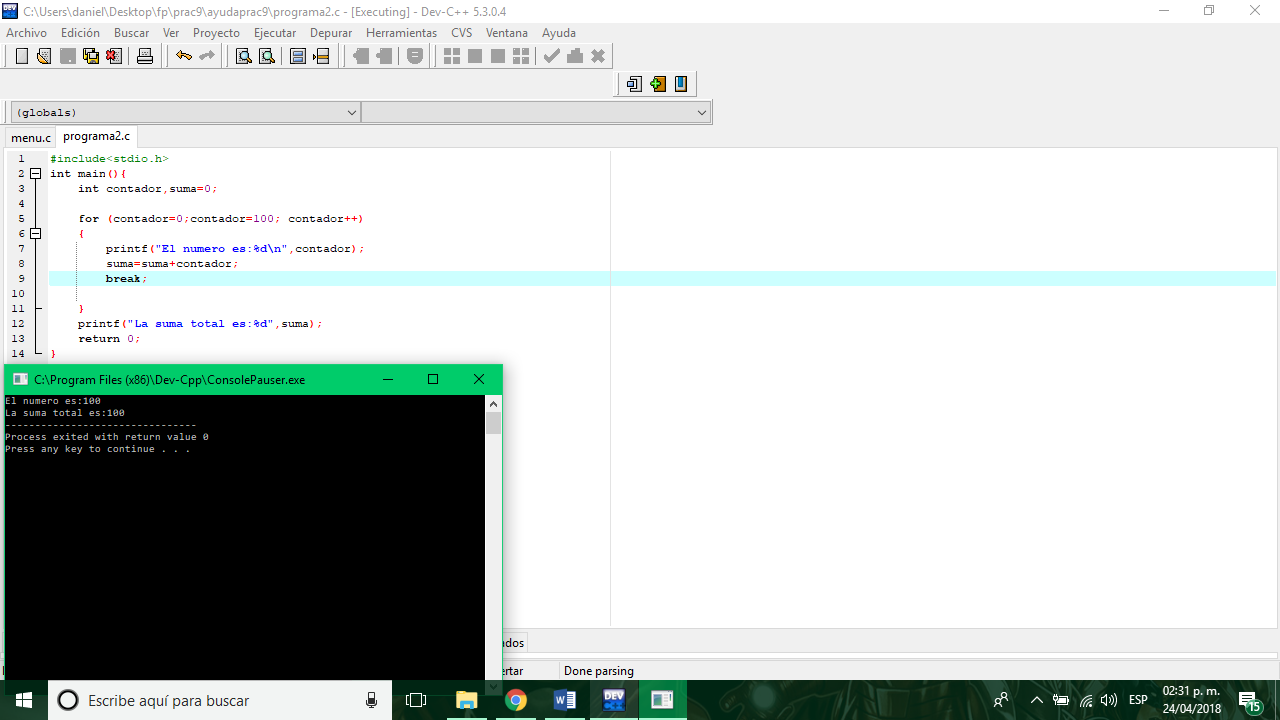
R: Este programa ya lo había acabado en la clase cuando lo dejo, de todos modos aquí se lo dejo de nuevo (se puede ver completo en el ejemplo del do-while).



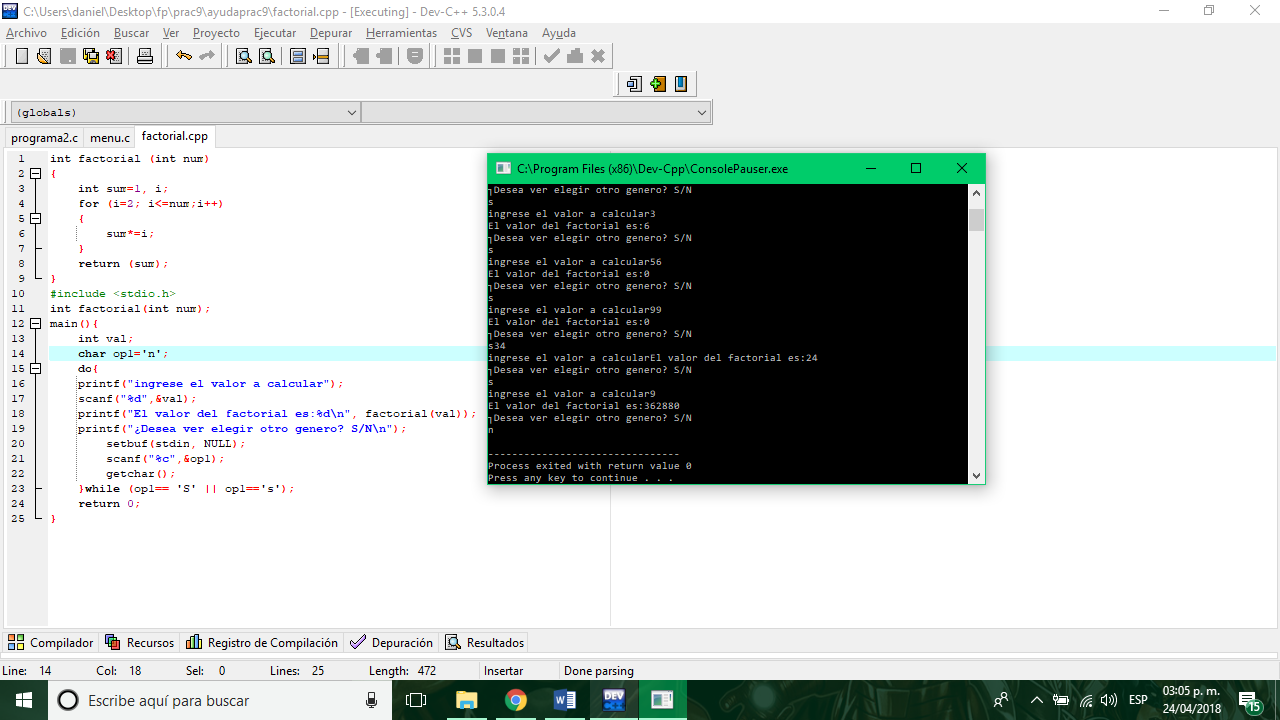
2.- Del ejercicio del ciclo for hecho en clase para ver la numeración del 1 al 100 y su sumatoria, modifica el ciclo for en la condición lógica de

contador <100 a contador = 100 y explica la salida.

R: Lo que curre qui es que como en ves de poner que haga hasta que sea menor que 100 osea hasta 99 (en mi caso yo puse en el ejemplo de este programa hasta <= para que hiciera hasta 100) y en ves de eso colocamos el =100 lo que hace es que solo se hace cuando contador vale 100 y como en este punto ya no hace mas coloca 100 indefinidamente ya que no hay algo que lo pare.

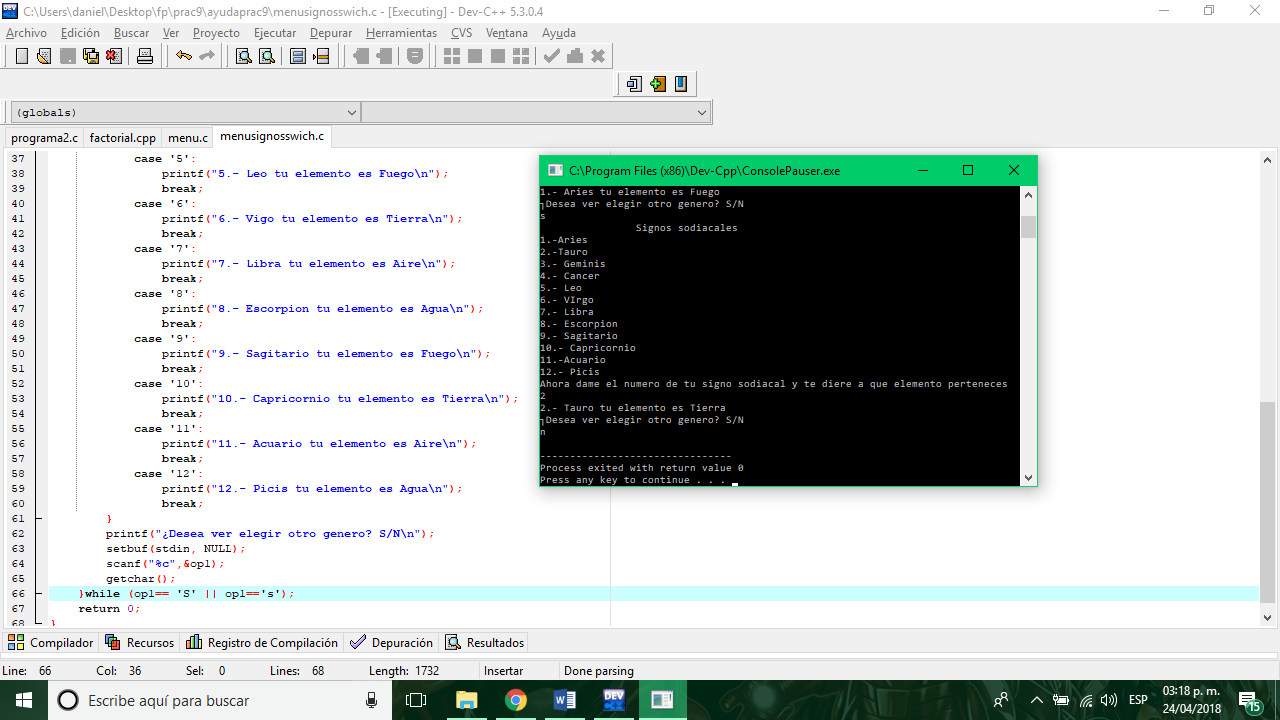
 

3.- Realiza el programa para calcular el factorial y preguntar al usuario si desea realizar otro cálculo antes de salir: R:



4.- Agregar un ciclo do-while al ejercicio propuesto número 4 de la práctica 8 y comentar cuál es el objetivo de éste.

R: El objetivo es el mismo que en el menú de películas y el del factorial y es darle al usuario la oportunidad de hacer infinitas consultas del programa, tanto el quiera.



5.- ¿Cuántas veces se ejecuta el siguiente fragmento de código?, observa y analiza

while (100)

Printf("Acierta");

break;

R: Se ejecuta una sola vez ya que tiene un break que rompe el ciclo y si no lo tuviera se iría al infinito ya que se cumple siempre y es como el que usted había dicho que respondiéramos -> for(i=0; i=valor ; i++) ya que valor empezaría en el 0.

**Conclusión:**

Podemos concluir que las estructuras de control repetitivas, son muy venas, ya que nos ayudan a hacer todavía más amplia nuestra gama de programa que podemos hacer y nos poyan haciendo el programa mucho mejor ya que nos permite hacer ciclos para repetir procesos y hacer un menú para cualquier cosa, como un menú de programas, que es de lo que está hecho prácticamente todas las aplicaciones que usamos en el día a día.

**Bibliografía:**

* El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, segunda edición, USA, Pearson Educación 1991.
* <http://lcp02.fi-b.unam.mx/>