|  |
| --- |
| Laboratorio de Computación  Salas A y B |
|  |

|  |  |
| --- | --- |
| Profesor: | García Morales Karina |
| Asignatura: | Fundamentos de Programación |
| Grupo: | 1121 |
| No de Práctica(s): | 09 |
| Integrante(s): | Valle Olivas Guillermo |
|  |  |
|  |  |
|  |  |
| No. de Equipo de cómputo empleado: |  |
| Semestre: | 2019-1 |
| Fecha de entrega: |  |
| Observaciones: |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Estructuras de repetición**

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

**Actividades**:

x Elaborar un programa que utilice la estructura while en la solución de un problema

x Elaborar un programa que requiera el uso de la estructura do-while para resolver un problema. Hacer la comparación con el programa anterior para distinguir las diferencias de operación entre while y do-while.

x Resolver un problema dado por el profesor que utilice la estructura for en lugar de la estructura while.

x Usar la directiva define para elaboración de código versátil.

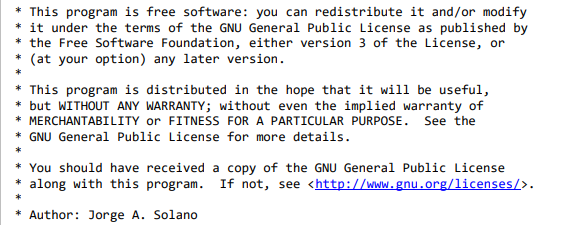
**Introducción**

Las estructuras de repetición son las 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.

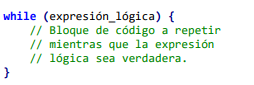
**Licencia GPL de GNU**

El software presente en esta práctica es libre bajo la licencia GPL de GNU, es decir, se puede modificar y distribuir mientras se mantenga la licencia GPL.



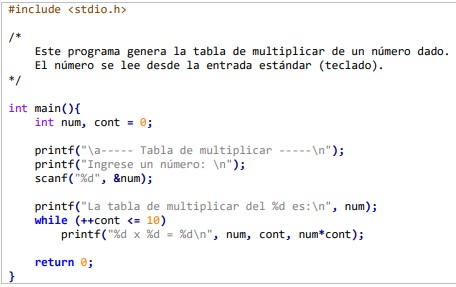
**Estructura de control repetitiva while**

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:

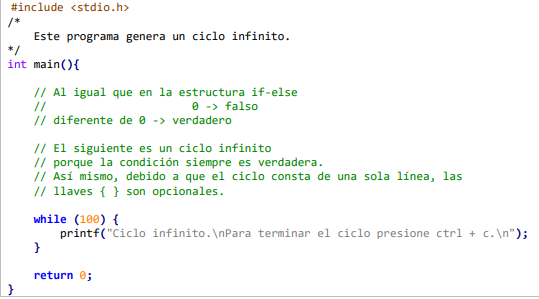


Si el bloque de código a repetir consta de una sola sentencia, entonces se pueden omitir las llaves.

**Código (estructura de repetición while)**

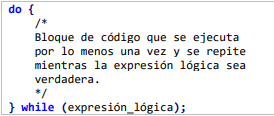


**Código (estructura de repetición while)**



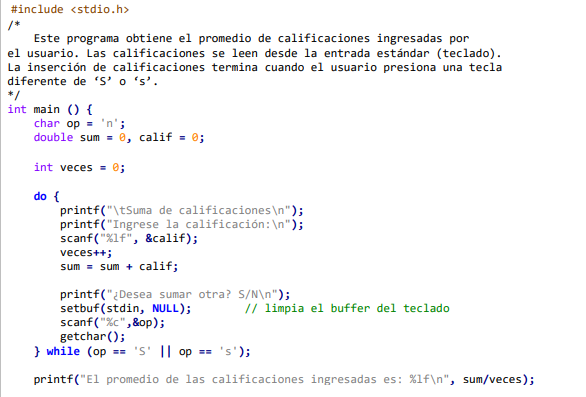
**Estructura de control repetitiva do-while**

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:

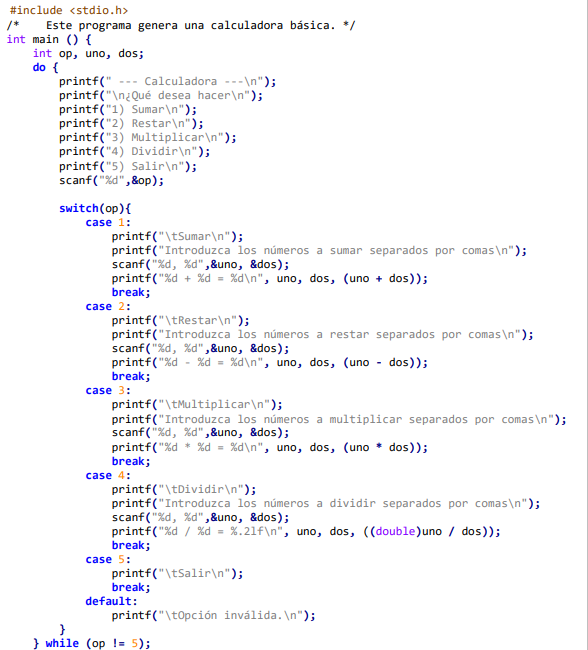


Si el bloque de código a repetir consta de una sola sentencia, entonces se pueden omitir las llaves. Esta estructura de control siempre termina con el signo de puntuación ';'.

**Código (estructura de repetición do-while)**

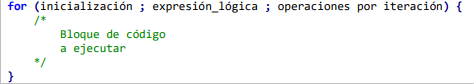


**Código (estructura de repetición do-while)**



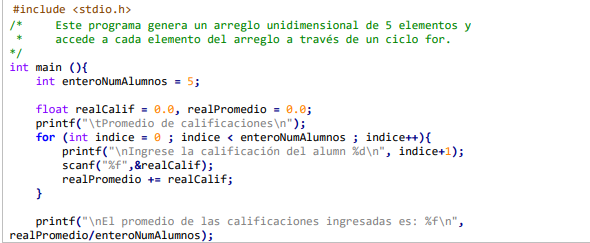
**Estructura de control de repetición for**

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)**



**Define**

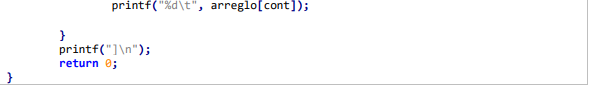
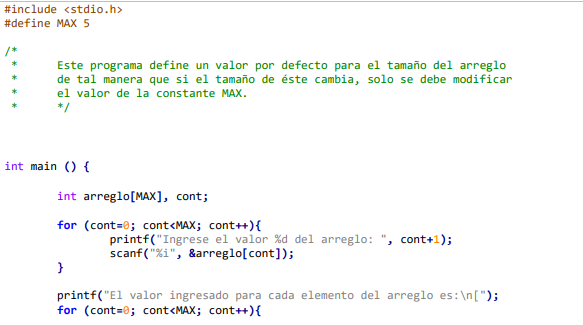
Las líneas de código que empiezan con # son directivas del preprocesador, el cual se encarga de realizar modificaciones en el texto del código fuente, como reemplazar un símbolo definido con #define por un parámetro o texto, o incluir un archivo en otro archivo con #include.

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



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)**

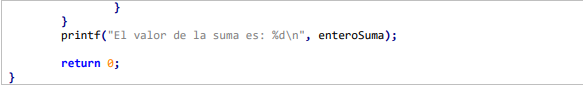
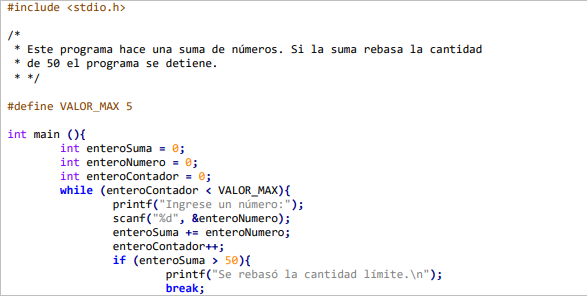


Cuando se compila el programa, se reemplazan la palabra MAX por el valor definido para la misma. Esto permite que, si el tamaño del arreglo cambia, solo se tiene que modificar el valor definido para MAX y en automático todos los arreglos y el recorrido de los mismos adquieren el nuevo valor (Mientras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

**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)**

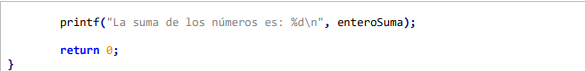
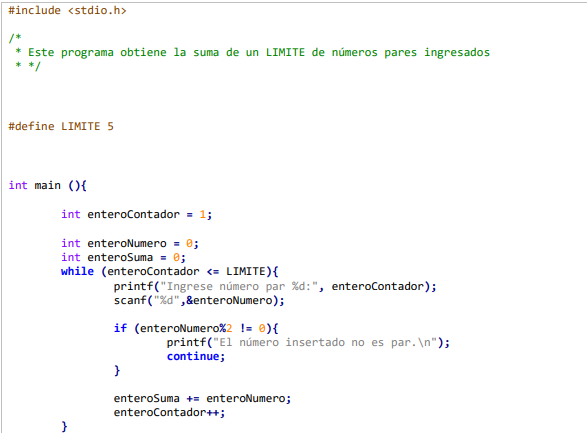


Cuando se compila el programa, MAX se sustituye por 5.

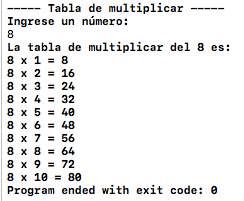
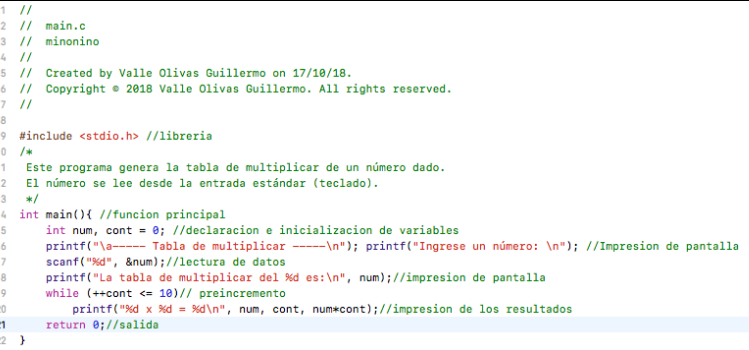
**Continue**

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

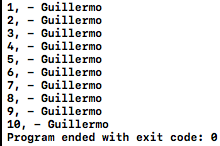
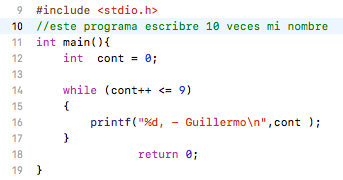
**Código (continue)**



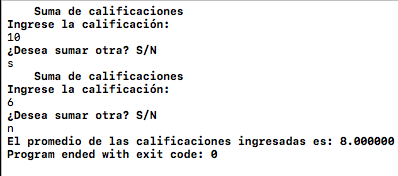
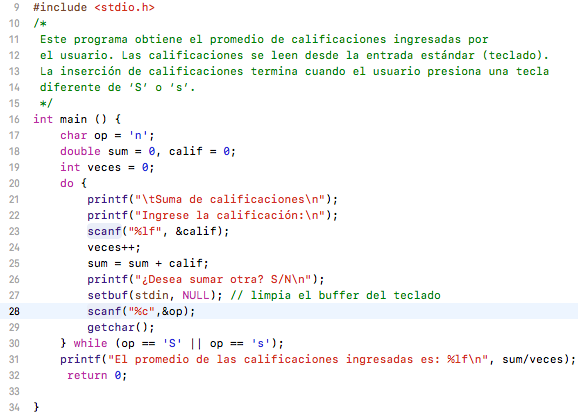
**Capturas de otros ejercicios realizados en el laboratorio**



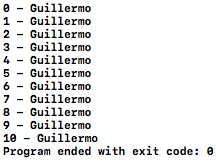
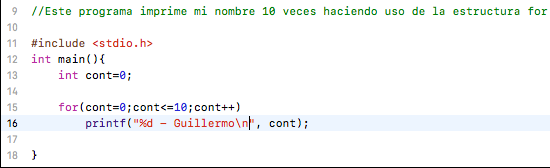
**/////////**



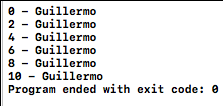
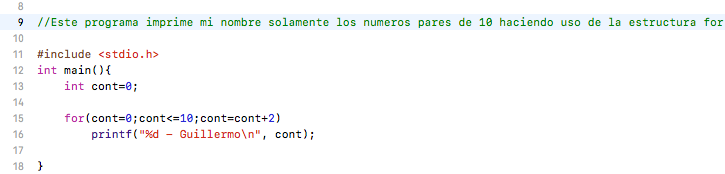
**///////**



**///////**

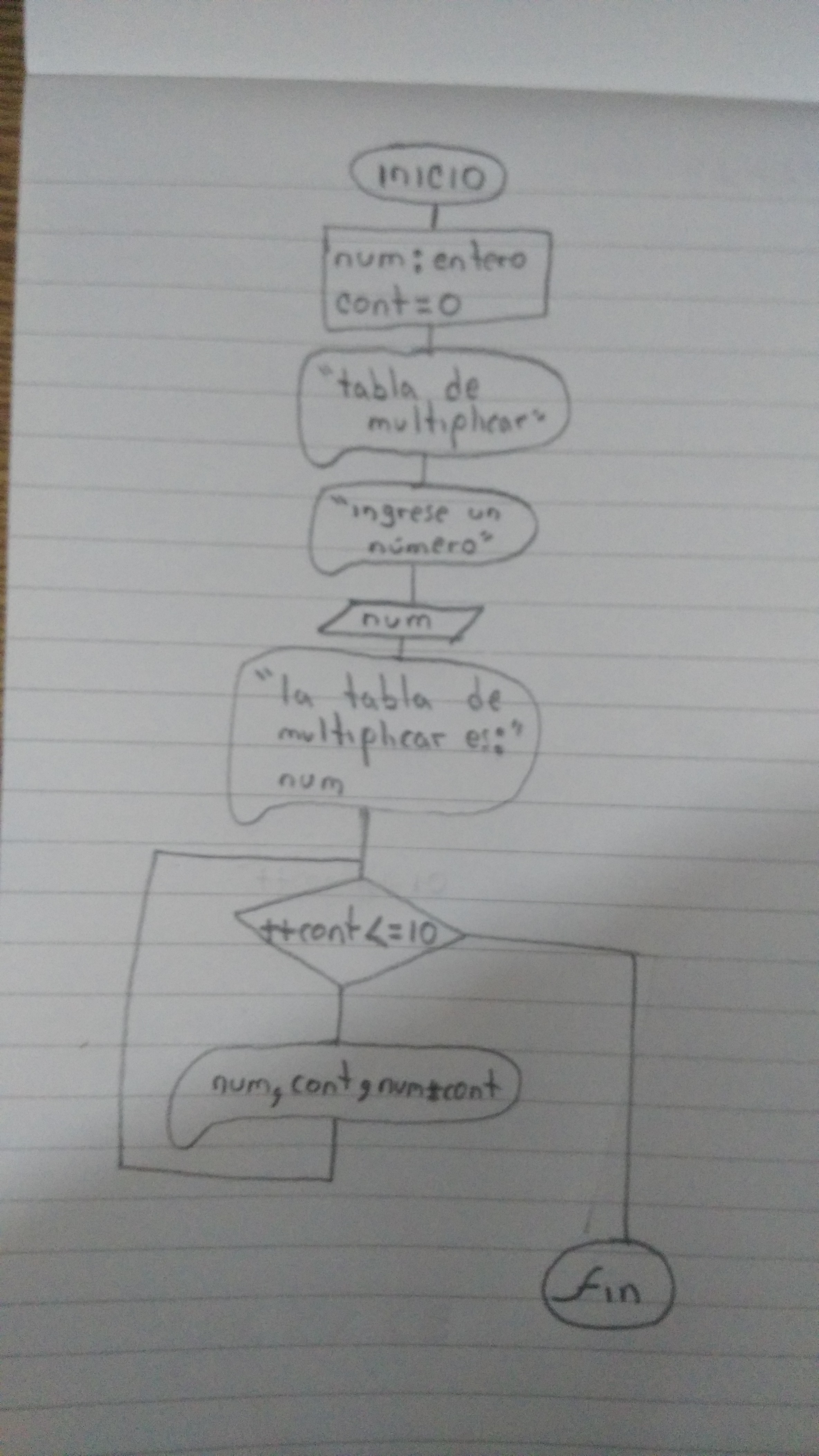


**//////**

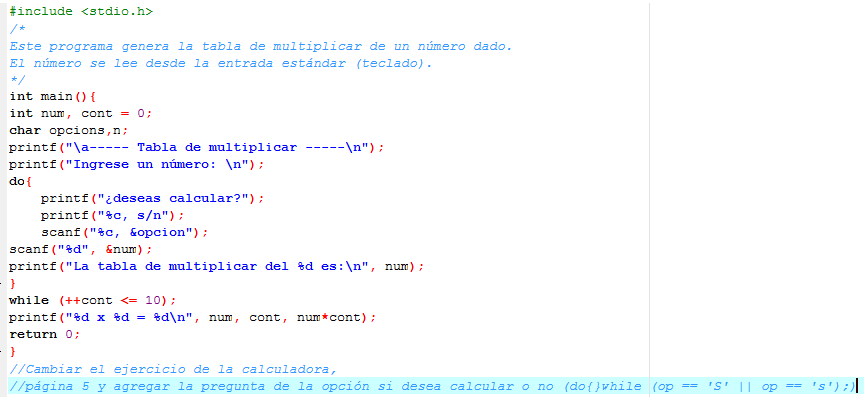


**Ejercicios de tarea**

1.- Dibujar el diagrama de flujo del programa para obtener la tabla de multiplicar con ciclo While.



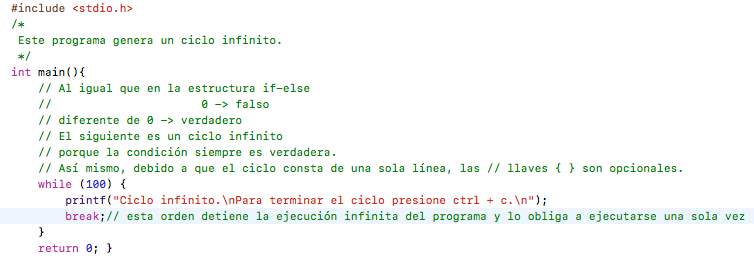
2.- Cambiar el ejercicio de la calculadora, página 5 y agregar la pregunta de la opción si desea calcular o no (do{}while (op == 'S' || op == 's');)



3.- Explica que sucede con el ciclo while al colocarle un valor positivo, negativo y cero y porque razón se obtiene ese resultado.

Lo que sucede tiene su origen en el código binario, si se coloca un valor de **0,** toma en cuenta que es un bit apagado y por lo tanto lo considera como falso, por el contrario, si se coloca un valor **diferente a 0, sea el valor que sea**, lo considera como una opción verdadera y el ciclo continuará su curso.

4.- Qué sucede con el ejercicio que ejecuta el break, si se omite y que sucede si se coloca?



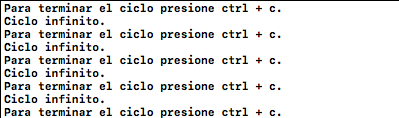
**Con break:**

Esta orden detiene la ejecución infinita del programa y lo ejecuta una sola vez

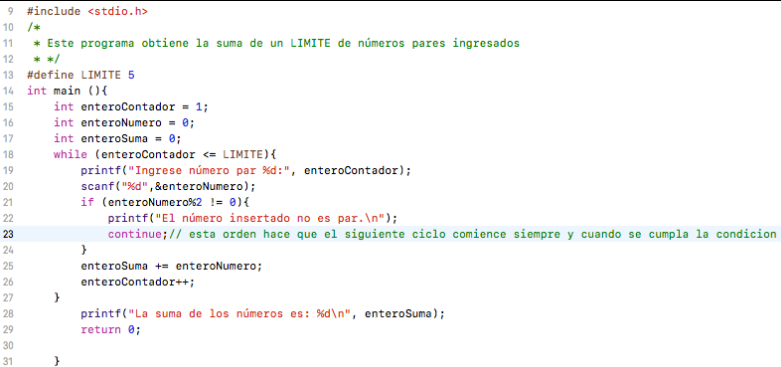


**Sin break:**

Sin esta orden el programa se continúa ejecutando de una forma infinita



5.- Qué sucede con el ejercicio que ejecuta continue, si se omite y que sucede si se coloca?



**Con continue:** La orden permite que el ciclo que sigue pueda ejecutarse siempre y cuando se cumpla la condición marcada.

**Sin continue:** Sin esta orden el ciclo continúa su curso sin importar la condición.

**Conclusión**

Con ayuda de esta práctica, me pude ayudar a repasar la sintaxis de los ciclos, además pude aprender nuevas órdenes que complementan las iteraciones; lo que si me costó trabajo fue entender en qué parte del código se colocan las nuevas órdenes y qué es lo que hacían, por ejemplo en la de break es muy notorio lo que hace, pero la de continue me costó trabajo ver la tarea que realizaba.

**Bibliografía**



El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, segunda edición, USA, Pearson Educación 1991.