	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorios de docencia

Laboratorio de Computación Salas A y B

Profesor: GARCIA MORALES KARINA ING.

Asignatura: FUNDAMENTOS DE PROGRAMACION

Grupo: 1121

No de Práctica(s): 9

Integrante(s): JOSE DANIEL CALLEJAS SANDOVAL

No. de Equipo de cómputo empleado: 25

Semestre: 1

Fecha de entrega: 23-10-2018

Observaciones:

CALIFICACIÓN: _____

Objetivo:

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

Desarrollo:

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.

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.

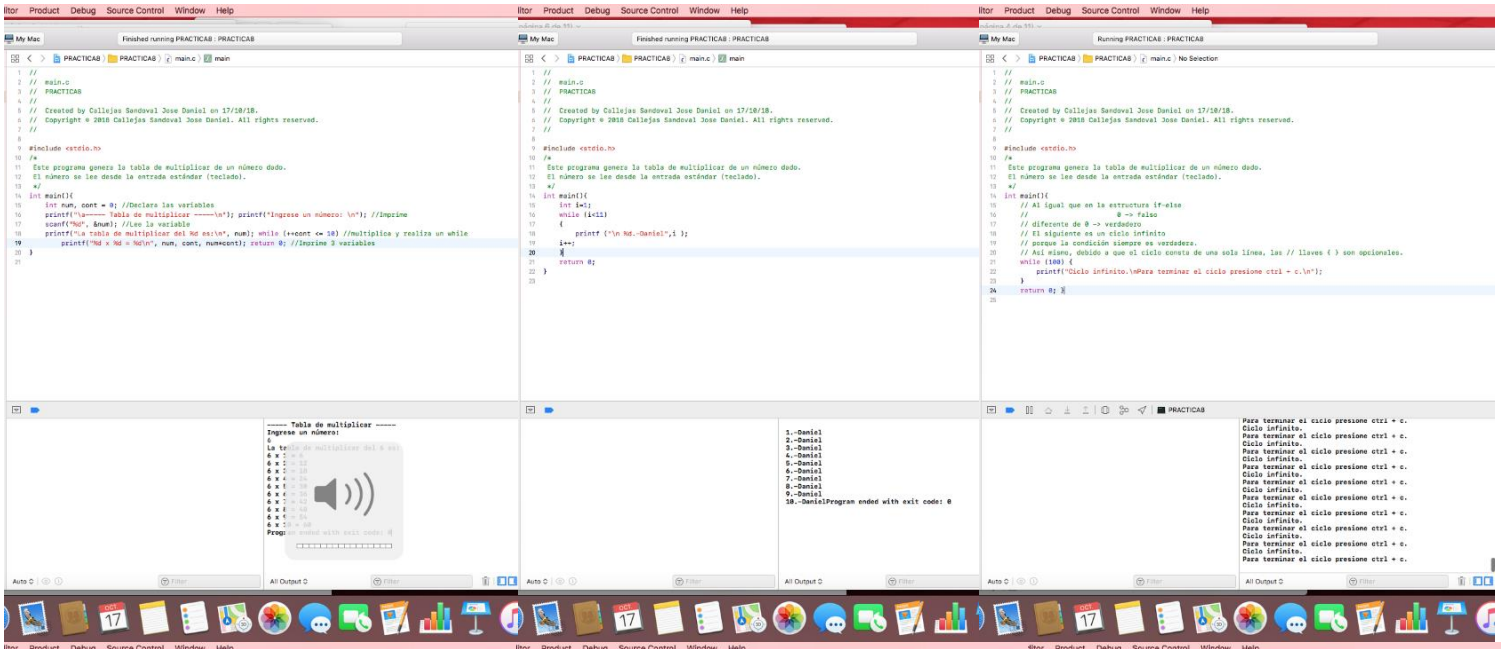
Estructura de control repetitiva do-while: do-while es una estructura cíclica que ejecuta el bloque de código que se encuentra dentro de las llaves y después valida la condición, es decir, el bloque de código se ejecuta de una a ene veces.

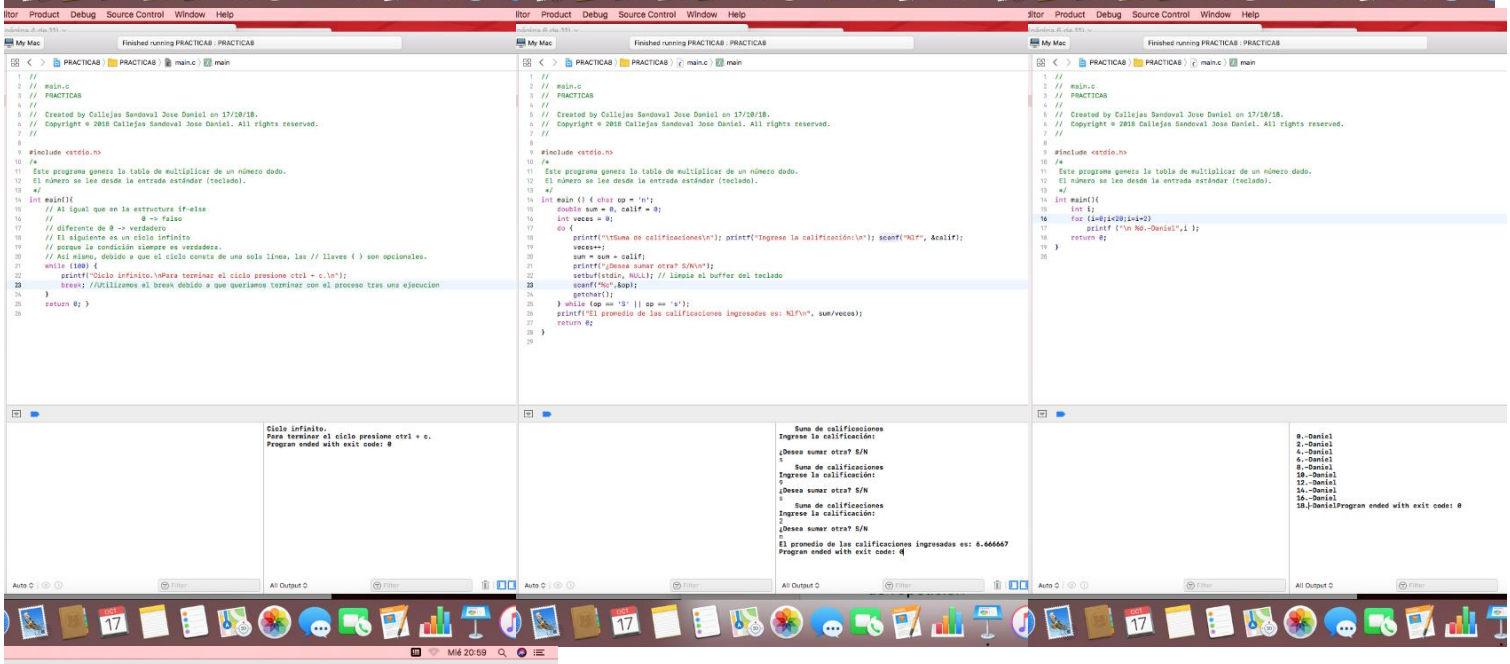
Estructura de control de repetición for: Lenguaje C posee la estructura de repetición for la cual permite realizar repeticiones cuando se conoce el número de elementos que se quiere recorrer. La estructura for ejecuta 3 acciones básicas antes o después de ejecutar el bloque de código. La primera acción 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.

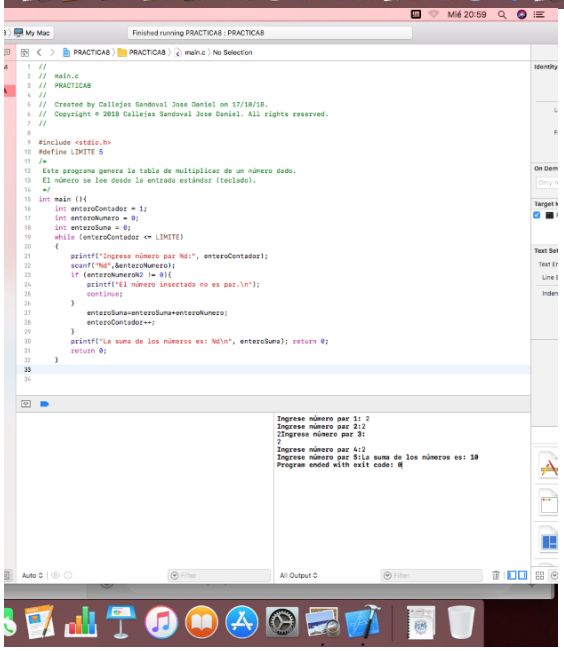
Define: Las líneas de código que empiezan con # son directivas del preprocesador, el cual se encarga de realizar modificaciones en el texto del código fuente, como reemplazar un símbolo definido con #define por un parámetro o texto, o incluir un archivo en otro archivo con #include. define permite definir constantes o literales; se les nombra también como constantes simbólicas. Su sintaxis es la siguiente: #define 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.

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.

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

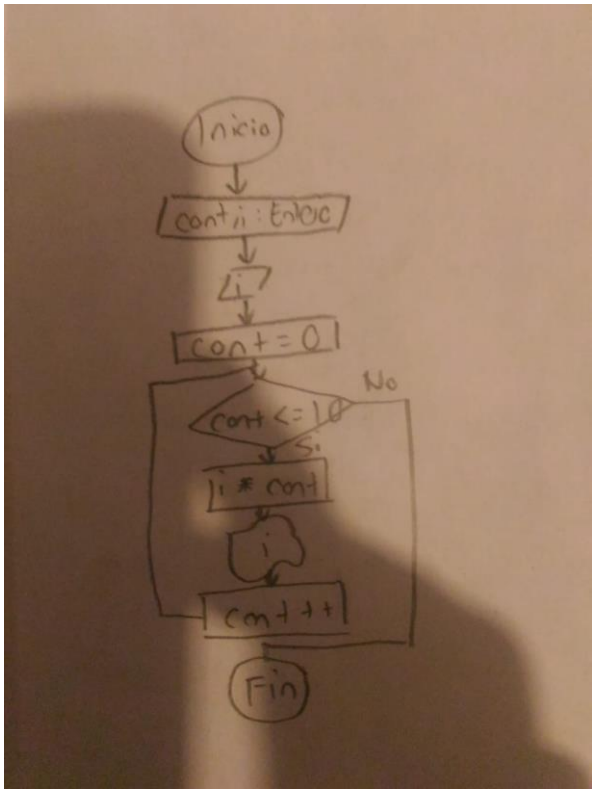




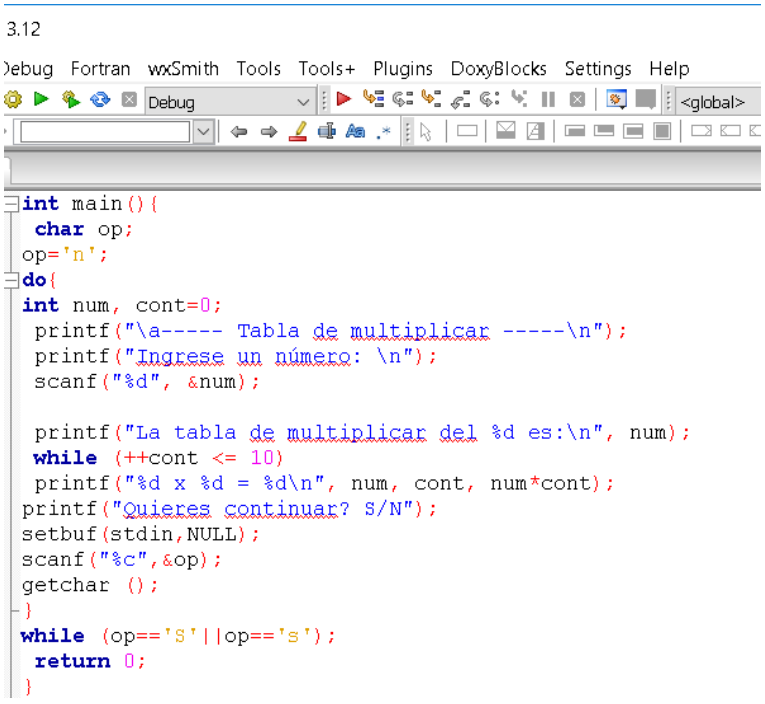


Tareas:

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



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




```
int main(){
 char op;
op='n';
do{
int num, cont=0;
 printf("\a----- Tabla de multiplicar -----\n");
 printf("Ingrese un número: \n");
 scanf("%d", &num);

 printf("La tabla de multiplicar del %d es:\n", num);
 while (++cont <= 10)
 printf("%d x %d = %d\n", num, cont, num*cont);
printf("Quieres continuar? S/N");
setbuf(stdin,NULL);
scanf("%c",&op);
getchar ();
}
while (op=='S'||op=='s');
 return 0;
}
```

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

Con valor positivo y negativo, toma la proposición como verdadera y se ejecuta infinitas veces.

Con valor cero, toma la proposición como falsa y no se ejecuta ni una sola vez.

Debido a que el while se basa en los booleanos (siendo 0, falso).

3.- ¿Que sucede con el ejercicio que ejecuta el break, si se omite y que sucede si se coloca?

El break permite que se “rompa” el ciclo para así lograr que solo se ejecute una sola vez en lugar de varias.

4.- ¿Que sucede con el ejercicio que ejecuta continue, si se omite y que sucede si se coloca?

El continue sirve para poder proceder a la siguiente iteracion

Conclusiones: Al finalizar la práctica, comprendí las principales diferencias entre los ciclos while, do-while y for y como cada uno es utilizado, según lo que necesitamos obtener.

Los ejercicios de tarea fueron un buen repaso para todo lo que vimos durante la sesión.

La práctica tuvo un ritmo adecuado para poder ver todos los ejemplos/casos con detenimiento y analizarlos.

Bibliografía:

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