



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

<i>Profesor:</i>	M.I. Marco Antonio Martínez Quintana
<i>Asignatura:</i>	Fundamentos de Programación
<i>Grupo:</i>	03
<i>No de Práctica(s):</i>	09
<i>Integrante(s):</i>	Teran García Rodolfo Mario
<i>No. de Equipo de cómputo empleado:</i>	No aplica
<i>No. de Lista o Brigada:</i>	
<i>Semestre:</i>	2021-1
<i>Fecha de entrega:</i>	30/11/2020
<i>Observaciones:</i>	M.I. Marco Antonio Martínez Quintana

Calificación: _____

Práctica #09: Estructuras de repetición

Objetivo:

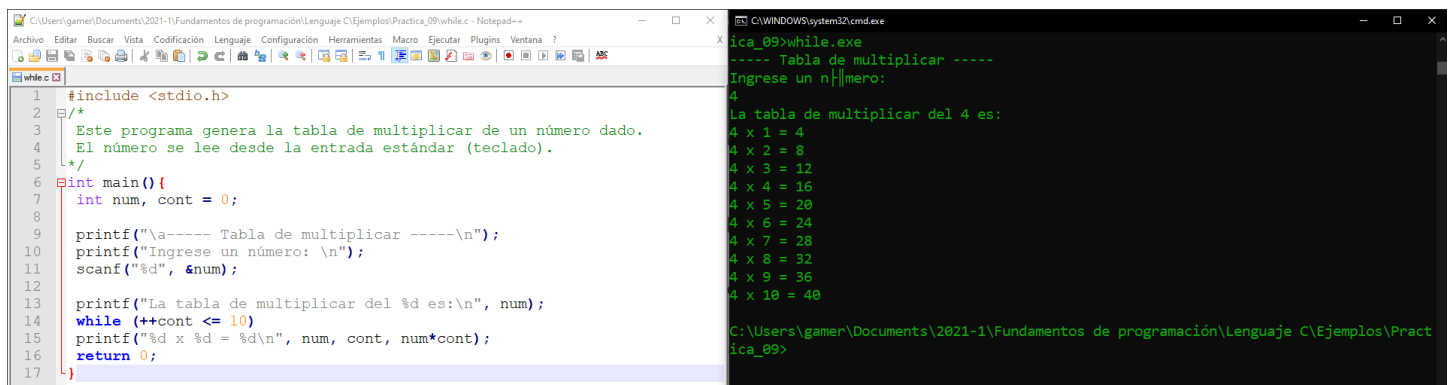
Elaborar programas en C para la resolución de problemas básicos que incluyan las estructuras de repetición y la directiva define.

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.

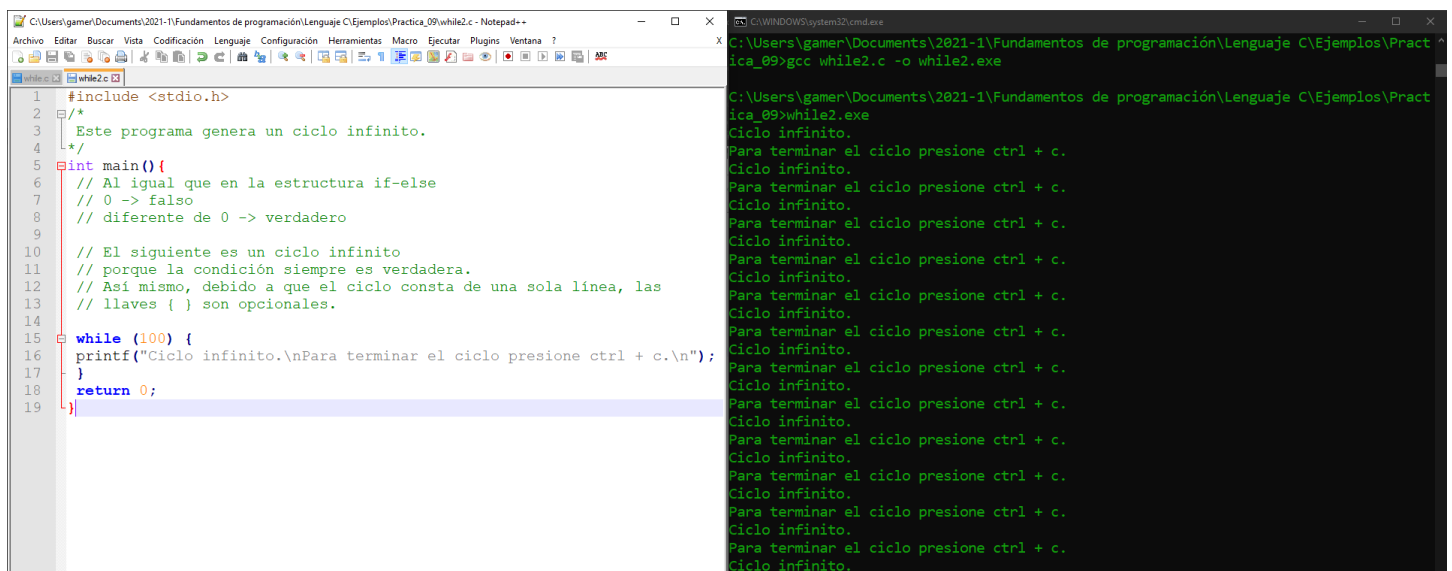
Desarrollo:

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 nueve veces.



```
1 #include <stdio.h>
2
3 /*
4  Este programa genera la tabla de multiplicar de un número dado.
5  El número se lee desde la entrada estándar (teclado).
6 */
7
8 int main(){
9     int num, cont = 0;
10
11     printf("\a----- Tabla de multiplicar ----- \n");
12     printf("Ingrese un número: \n");
13     scanf("%d", &num);
14
15     printf("La tabla de multiplicar del %d es:\n", num);
16     while (++cont <= 10)
17         printf("%d x %d = %d\n", num, cont, num*cont);
18     return 0;
19 }
```

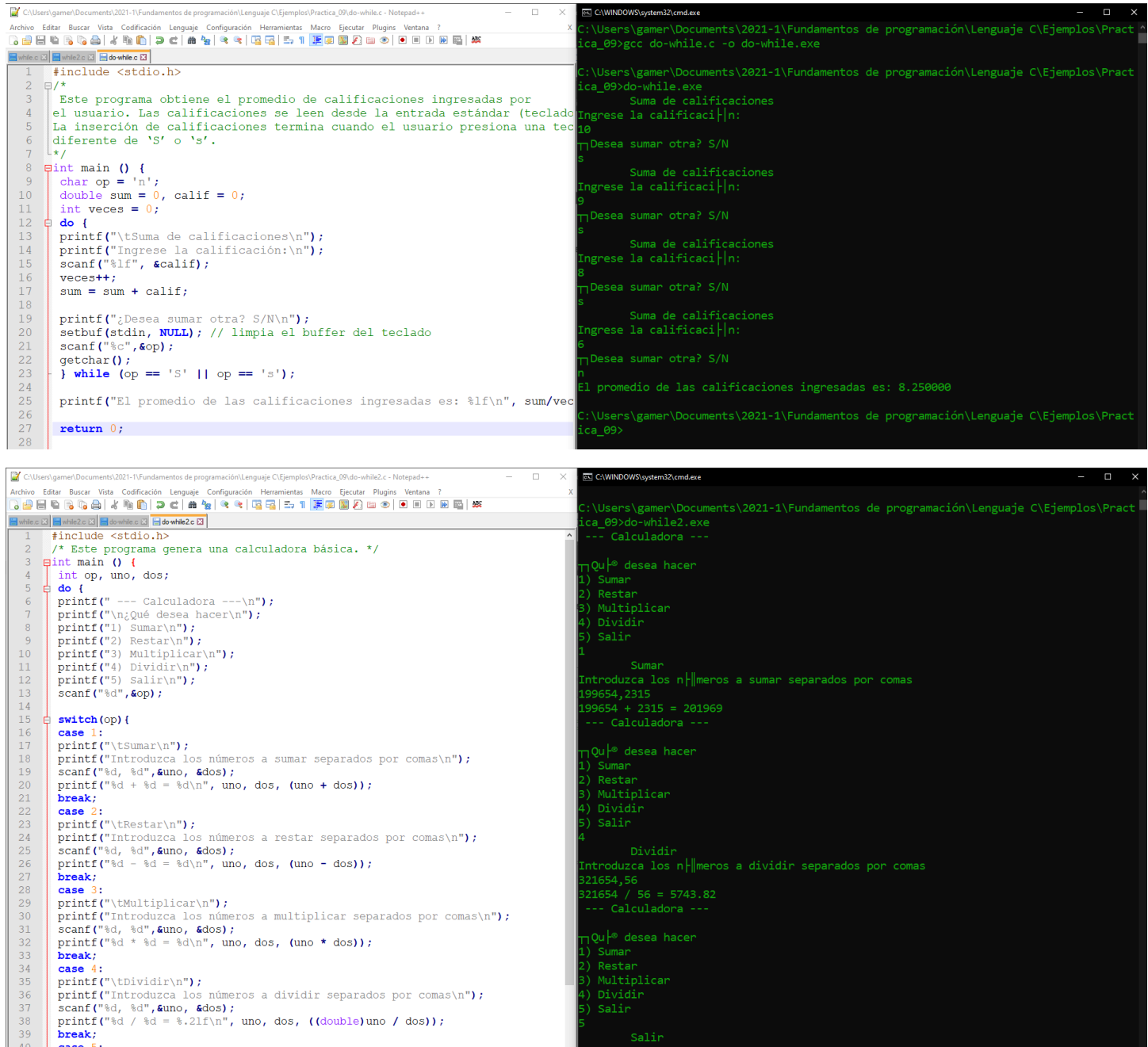
The terminal output shows the program execution: it prompts for a number, reads '4', and displays the multiplication table for 4 from 1 to 10.



```
1 #include <stdio.h>
2
3 /*
4  Este programa genera un ciclo infinito.
5 */
6
7 int main(){
8     // Al igual que en la estructura if-else
9     // 0 -> falso
10    // diferente de 0 -> verdadero
11
12    // El siguiente es un ciclo infinito
13    // porque la condición siempre es verdadera.
14    // Así mismo, debido a que el ciclo consta de una sola línea, las
15    // llaves { } son opcionales.
16
17    while (100) {
18        printf("Ciclo infinito.\nPara terminar el ciclo presione ctrl + c.\n");
19    }
20    return 0;
21 }
```

The terminal output shows the program running an infinite loop, printing "Ciclo infinito." repeatedly. It includes instructions on how to terminate the loop by pressing Ctrl + C.

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 nueve veces. 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 ';'.



The image displays four screenshots of a C program and its execution. The first two screenshots show the source code in Notepad++, and the last two show the program's output in a command prompt.

Top Left: Source Code (do-while.c)

```
1 #include <stdio.h>
2 /*
3  Este programa obtiene el promedio de calificaciones ingresadas por
4  el usuario. Las calificaciones se leen desde la entrada estándar (teclado)
5  La inserción de calificaciones termina cuando el usuario presiona una tecla
6  diferente de 'S' o 's'.
7  */
8 int main () {
9     char op = 'n';
10    double sum = 0, calif = 0;
11    int veces = 0;
12    do {
13        printf("\tSuma de calificaciones\n");
14        printf("Ingrese la calificación:\n");
15        scanf("%lf", &calif);
16        veces++;
17        sum = sum + calif;
18
19        printf("\tDesea sumar otra? S/N\n");
20        setbuf(stdin, NULL); // limpia el buffer del teclado
21        scanf("%c", &op);
22        getchar();
23    } while (op == 'S' || op == 's');
24
25    printf("El promedio de las calificaciones ingresadas es: %lf\n", sum/veces);
26
27    return 0;
28 }
```

Top Right: Execution Output

```
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_09>gcc do-while.c -o do-while.exe
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_09>do-while.exe
Suma de calificaciones
Ingrese la calificación:
10
Desea sumar otra? S/N
s
Suma de calificaciones
Ingrese la calificación:
9
Desea sumar otra? S/N
s
Suma de calificaciones
Ingrese la calificación:
8
Desea sumar otra? S/N
s
Suma de calificaciones
Ingrese la calificación:
6
Desea sumar otra? S/N
n
El promedio de las calificaciones ingresadas es: 8.250000
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_09>
```

Bottom Left: Source Code (do-while2.c)

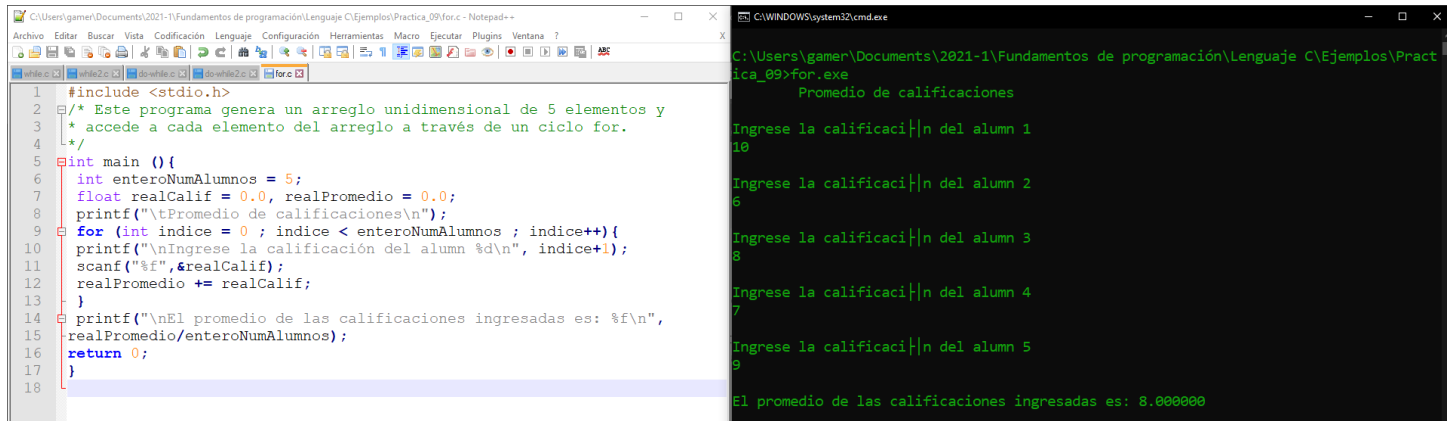
```
1 #include <stdio.h>
2 /* Este programa genera una calculadora básica. */
3 int main () {
4     int op, uno, dos;
5     do {
6         printf(" --- Calculadora ---\n");
7         printf("\n¿Qué desea hacer?\n");
8         printf("1) Sumar\n");
9         printf("2) Restar\n");
10        printf("3) Multiplicar\n");
11        printf("4) Dividir\n");
12        printf("5) Salir\n");
13        scanf("%d", &op);
14
15        switch(op) {
16            case 1:
17                printf("\tSumar\n");
18                printf("Introduzca los números a sumar separados por comas\n");
19                scanf("%d", &uno, &dos);
20                printf("%d + %d = %d\n", uno, dos, (uno + dos));
21                break;
22            case 2:
23                printf("\tRestar\n");
24                printf("Introduzca los números a restar separados por comas\n");
25                scanf("%d", &uno, &dos);
26                printf("%d - %d = %d\n", uno, dos, (uno - dos));
27                break;
28            case 3:
29                printf("\tMultiplicar\n");
30                printf("Introduzca los números a multiplicar separados por comas\n");
31                scanf("%d", &uno, &dos);
32                printf("%d * %d = %d\n", uno, dos, (uno * dos));
33                break;
34            case 4:
35                printf("\tDividir\n");
36                printf("Introduzca los números a dividir separados por comas\n");
37                scanf("%d", &uno, &dos);
38                printf("%d / %d = %.2lf\n", uno, dos, ((double)uno / dos));
39                break;
40            case 5:
41                printf("Salir\n");
42                return 0;
43        }
44    } while (op != 5);
45 }
```

Bottom Right: Execution Output

```
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_09>do-while2.exe
 --- Calculadora ---
¿Qué desea hacer?
1) Sumar
2) Restar
3) Multiplicar
4) Dividir
5) Salir
1
Sumar
Introduzca los números a sumar separados por comas
199654,2315
199654 + 2315 = 201969
 --- Calculadora ---
¿Qué desea hacer?
1) Sumar
2) Restar
3) Multiplicar
4) Dividir
5) Salir
4
Dividir
Introduzca los números a dividir separados por comas
321654,56
321654 / 56 = 5743.82
 --- Calculadora ---
¿Qué desea hacer?
1) Sumar
2) Restar
3) Multiplicar
4) Dividir
5) Salir
5
Salir
```

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

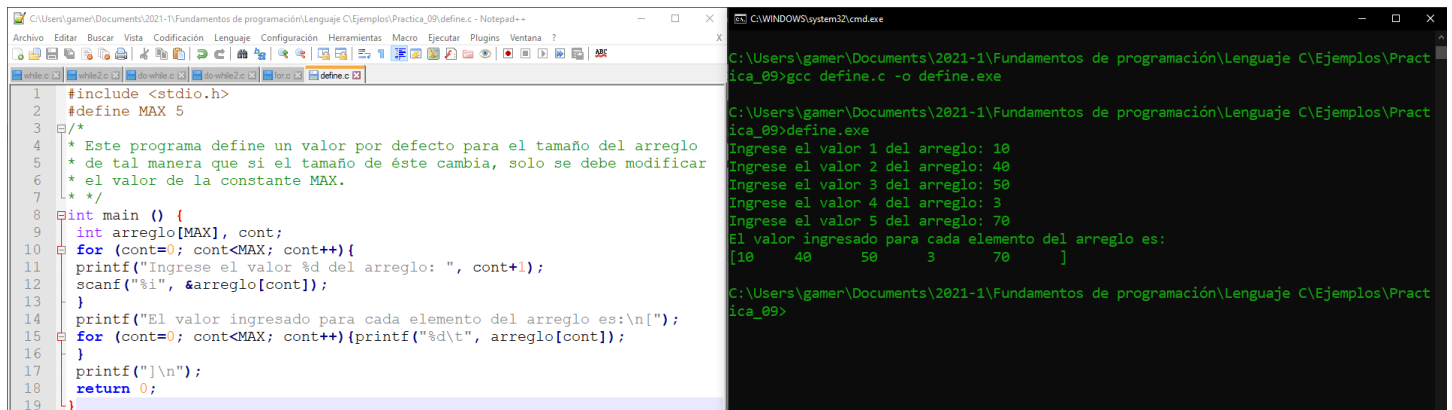
verdadera, ejecuta el bloque de código, si no se cumple se continúa la ejecución del programa; esta parte es opcional. La tercera parte consta de un conjunto de operaciones que se realizan cada vez que termina de ejecutarse el bloque de código y antes de volver a validar la expresión lógica; esta parte también es opcional.



```
#include <stdio.h>
/* Este programa genera un arreglo unidimensional de 5 elementos y
 * accede a cada elemento del arreglo a través de un ciclo for.
 */
int main () {
    int enteroNumAlumnos = 5;
    float realCalif = 0.0, realPromedio = 0.0;
    printf("\tPromedio de calificaciones\n");
    for (int indice = 0; indice < enteroNumAlumnos; indice++){
        printf("\nIngrese la calificación del alumn %d\n", indice+1);
        scanf("%f", &realCalif);
        realPromedio += realCalif;
    }
    printf("\nEl promedio de las calificaciones ingresadas es: %f\n",
        realPromedio/enteroNumAlumnos);
    return 0;
}
```

```
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_09>for.exe
Promedio de calificaciones
Ingrese la calificación del alumn 1
10
Ingrese la calificación del alumn 2
6
Ingrese la calificación del alumn 3
8
Ingrese la calificación del alumn 4
7
Ingrese la calificación del alumn 5
9
El promedio de las calificaciones ingresadas es: 8.000000
```

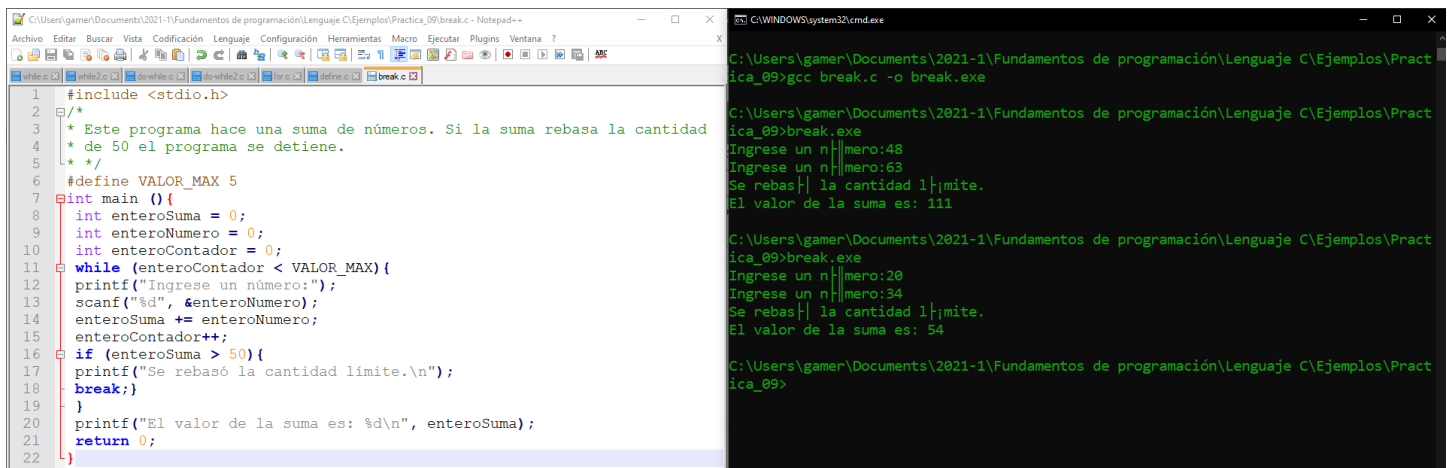
Define: Las líneas de código que empiezan con # son directivas del preprocesador, el cual se encarga de realizar modificaciones en el texto del código fuente, como reemplazar un símbolo definido con #define por un parámetro o texto, o incluir un archivo en otro archivo con #include, define permite definir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



```
#include <stdio.h>
#define MAX 5
/*
 * Este programa define un valor por defecto para el tamaño del arreglo
 * de tal manera que si el tamaño de éste cambia, solo se debe modificar
 * el valor de la constante MAX.
 */
int main () {
    int arreglo[MAX], cont;
    for (cont=0; cont<MAX; cont++){
        printf("Ingrese el valor %d del arreglo: ", cont+1);
        scanf("%i", &arreglo[cont]);
    }
    printf("El valor ingresado para cada elemento del arreglo es:\n");
    for (cont=0; cont<MAX; cont++){printf("%d\t", arreglo[cont]);
    }
    printf("\n");
    return 0;
}
```

```
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_09>gcc define.c -o define.exe
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_09>define.exe
Ingrese el valor 1 del arreglo: 10
Ingrese el valor 2 del arreglo: 40
Ingrese el valor 3 del arreglo: 50
Ingrese el valor 4 del arreglo: 3
Ingrese el valor 5 del arreglo: 70
El valor ingresado para cada elemento del arreglo es:
[10 40 50 3 70 ]
```

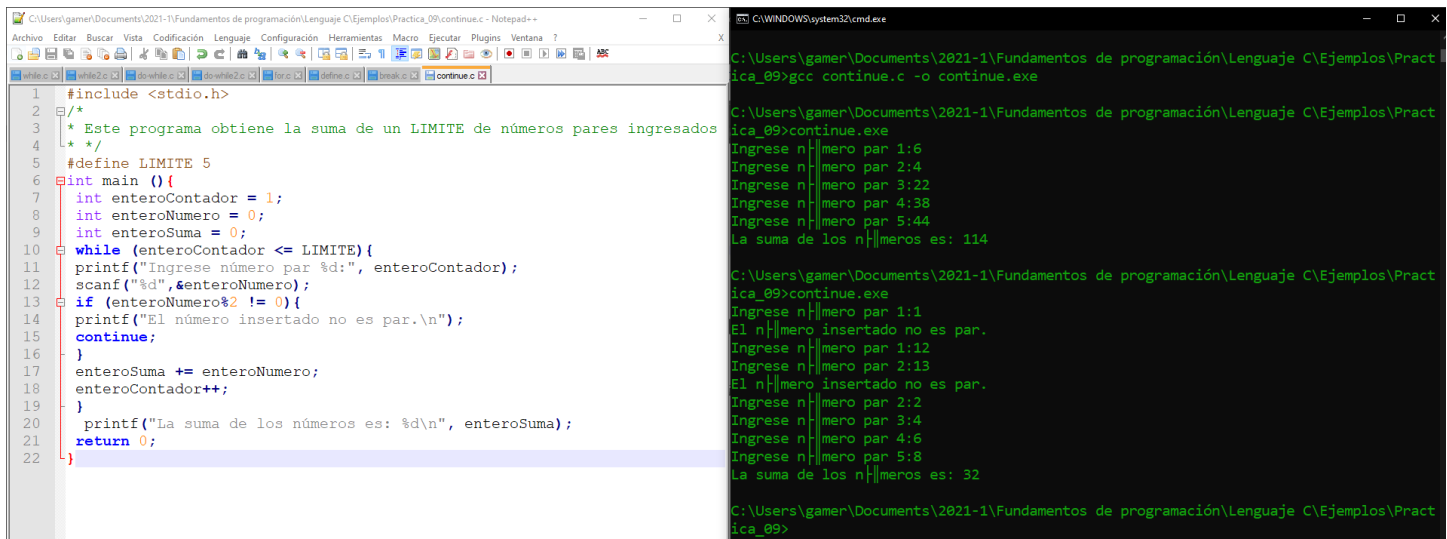
Break: Algunas veces es conveniente tener la posibilidad de abandonar un ciclo. La proposición break proporciona una salida anticipada dentro de una estructura de repetición, tal como lo hace en un switch. Un break provoca que el ciclo que lo encierra termine inmediatamente.



```
#include <stdio.h>
/*
 * Este programa hace una suma de números. Si la suma rebasa la cantidad
 * de 50 el programa se detiene.
 */
#define VALOR_MAX 5
int main () {
    int enteroSuma = 0;
    int enteroNumero = 0;
    int enteroContador = 0;
    while (enteroContador < VALOR_MAX){
        printf("Ingrese un número:");
        scanf("%d", &enteroNumero);
        enteroSuma += enteroNumero;
        enteroContador++;
        if (enteroSuma > 50){
            printf("Se rebasó la cantidad limite.\n");
            break;
        }
        printf("El valor de la suma es: %d\n", enteroSuma);
        return 0;
    }
```

```
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_09>gcc break.c -o break.exe
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_09>break.exe
Ingrese un n|mero:48
Ingrese un n|mero:63
Se rebas| la cantidad l|mite.
El valor de la suma es: 111
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_09>break.exe
Ingrese un n|mero:20
Ingrese un n|mero:34
Se rebas| la cantidad l|mite.
El valor de la suma es: 54
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_09>
```

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



```
#include <stdio.h>
/*
 * Este programa obtiene la suma de un LIMITE de números pares ingresados
 */
#define LIMITE 5
int main () {
    int enteroContador = 1;
    int enteroNumero = 0;
    int enteroSuma = 0;
    while (enteroContador <= LIMITE){
        printf("Ingrese número par %d:", enteroContador);
        scanf("%d",&enteroNumero);
        if (enteroNumero%2 != 0){
            printf("El número insertado no es par.\n");
            continue;
        }
        enteroSuma += enteroNumero;
        enteroContador++;
    }
    printf("La suma de los números es: %d\n", enteroSuma);
    return 0;
}
```

```
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_09>gcc continue.c -o continue.exe
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_09>continue.exe
Ingrese n|mero par 1:6
Ingrese n|mero par 2:4
Ingrese n|mero par 3:22
Ingrese n|mero par 4:38
Ingrese n|mero par 5:44
La suma de los n|meros es: 114
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_09>continue.exe
Ingrese n|mero par 1:1
El n|mero insertado no es par.
Ingrese n|mero par 1:12
Ingrese n|mero par 2:13
El n|mero insertado no es par.
Ingrese n|mero par 2:2
Ingrese n|mero par 3:4
Ingrese n|mero par 4:6
Ingrese n|mero par 5:8
La suma de los n|meros es: 32
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_09>
```

Conclusión:

En esta practica como en la anterior, pudimos darle continuidad a nuestro proceso de aprendizaje del lenguaje en C mediante el uso de estructuras de control, mismas que son la base de todo programa y el alma de todo código, en esta oportunidad con estructuras un poco más complejas de uso más común en la vida cotidiana, en sistemas que utilizan las grandes empresas, se ven aun las estructuras de control como las que aplicamos hoy en día, tal vez en un conjunto más elaborado pero con base similar, misma base que será la que nos va a acompañar durante toda nuestra formación, siendo las estructuras de control un tema imprescindible, pues al ser el alma de nuestro código fuente nos brindan una o varias alternativas para desenmarañar los secretos de la programación como una herramienta que por medio de la codificación de instrucciones comprensibles por una computadora nos guía a nosotros mismo a llevar a cabo tareas que incluso hace cincuenta años parecían magia o cosa imposible, hoy en día no son solo una realidad, sino el día a día de todo mundo.