



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

### Laboratorios de computación Salas A y B

---

*Profesor:* Alejandro Pimentel

*Asignatura:* Laboratorio de protraction

*Grupo:* 135

*No de Práctica(s):* Práctica 9

Areli González Segura

*Integrante(s):*

*No. de Equipo de  
cómputo empleado:* 23

*No. de Lista o Brigada:* 5319 no. Lista: 19

*Semestre:* Primer semestre

*Fecha de entrega:* 14/octubre/2019

*Observaciones:*

CALIFICACIÓN: \_\_\_\_\_

Objetivo:

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

Introducción:

## WHILE

- Un ciclo es cualquier construcción de programa que repite una sentencia o secuencia de sentencias un número de veces.
- La sentencia o grupo de sentencias que se repite en un bloque se denomina cuerpo del ciclo y cada repetición del cuerpo del ciclo se llama iteración del ciclo.
- Un ciclo while tiene una condición del ciclo, una expresión lógica que controla la secuencia de repetición.
- La posición de esta condición del ciclo es delante del cuerpo del ciclo y significa que un ciclo while es un ciclo de preverificación de modo que cuando se ejecuta el mismo, se evalúa la condición antes de que se ejecute el cuerpo del ciclo.

•

Sintaxis

```
while ( condición )  
    Sentencia;
```

- Si se requiere realizar más de una sentencia se deben utilizar llaves.

```
while ( condición )  
{  
    Bloque de sentencias;  
}
```

Aquí se ejecuta el (los) sentencia (s) mientras la condición es verdadera; al momento de ser falsa termina el ciclo. Si la condición es falsa la primera vez nunca se ejecuta(n) el (las) sentencia(s).

## DO WHILE

El ciclo do-while (Instrucción hacer – repetir mientras) es un tipo de estructura repetitiva eficiente. Lo que lo diferencia con el while es que en la estructura do-while la condición se evalúa al finalizar el ciclo, esto hace que las instrucciones se ejecuten cuando menos una vez.

La ejecución de esta estructura se realiza de la siguiente manera.

1.- Se ejecutan las instrucciones que se encuentran dentro del do, para esto es necesario ponerlas entre llaves.

2.- Después evalúa la expresión dentro de while. Si la expresión es falsa, el ciclo do-while finaliza y pasa a la siguiente instrucción del programa. Si la expresión es verdadera, el ciclo se repite.

## FOR

Un **for en programación** se usa cuando queremos repetir un conjunto de instrucciones un número finito de veces.

Vamos a ver qué significa esto... En programación existen los bucles, Como puede ser escribir con **while** o **for**.

¿Cuándo usar **for** en programación? ¿Qué diferencia hay entre **for** y **while**?

Con **while** se va repitiendo el código en base a una condición, es decir, mientras esa condición sea verdadera.

Con **for**, las instrucciones se repiten el número de veces que le decimos, normalmente le ponemos un número (o el valor de una variable o una constante).

## DEFINE

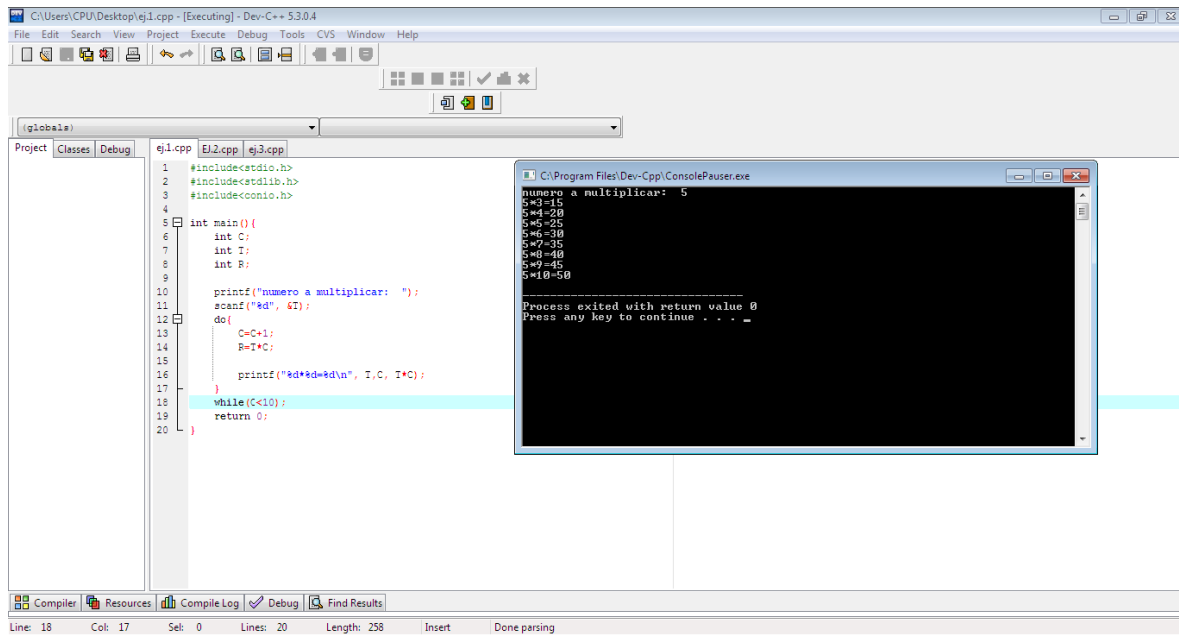
El **define** es una palabra clave que se utiliza para declarar un nombre especial con un significado. Es muy parecido a una variable, con la diferencia de que no se puede cambiar a lo largo del programa.

Actividades a realizar:

Para cada uno de los siguientes problemas, elegir un tipo de ciclo y resolverlo. Al final, deben usar los tres tipos de ciclos y usar **define** por lo menos una vez.

## Actividad 1

Hacer un programa que pida un número y muestre su tabla de multiplicar (hasta el 10).



The screenshot shows a C++ IDE with a project named 'ej1.cpp'. The code in the editor is as follows:

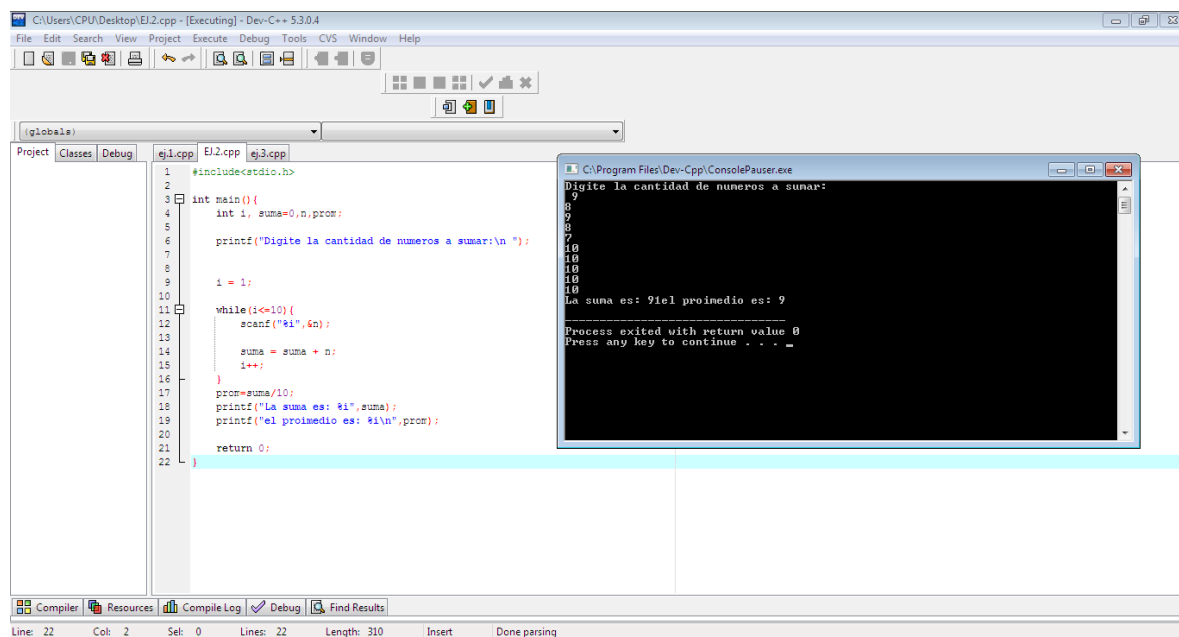
```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<conio.h>
4
5 int main() {
6     int C;
7     int I;
8     int R;
9
10    printf("numero a multiplicar: ");
11    scanf("%d", &I);
12    do {
13        C=C+1;
14        R=I*C;
15        printf("%d*d=%d\n", I,C, I*C);
16    }
17    while(C<=10);
18    return 0;
19 }
```

The output window shows the execution of the program with the input '5':

```
numero a multiplicar: 5
5*3=15
5*4=20
5*5=25
5*6=30
5*7=35
5*8=40
5*9=45
5*10=50
Process exited with return value 0
Press any key to continue . . .
```

## Actividad 2

Hacer un programa que pida y lea 10 números y muestre su suma y su prome



The screenshot shows a C++ IDE with a project named 'ej2.cpp'. The code in the editor is as follows:

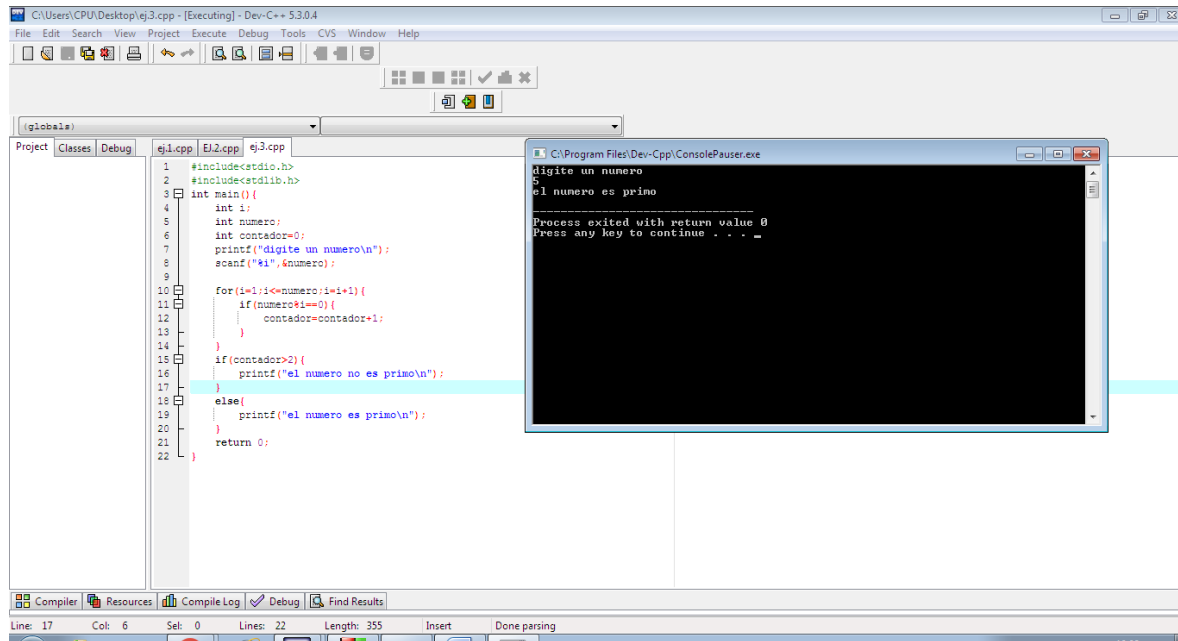
```
1 #include<stdio.h>
2
3 int main() {
4     int i, suma=0, n, prom;
5
6     printf("Digite la cantidad de numeros a sumar:\n ");
7
8     i = 1;
9
10    while(i<=10) {
11        scanf("%i", &n);
12        suma = suma + n;
13        i++;
14    }
15    prom=suma/10;
16    printf("La suma es: %i", suma);
17    printf("\nel promedio es: %i\n", prom);
18    return 0;
19 }
```

The output window shows the execution of the program with the input '9' for the number of numbers to sum:

```
Digite la cantidad de numeros a sumar:
9
9
9
9
9
9
9
9
9
9
La suma es: 91el promedio es: 9
Process exited with return value 0
Press any key to continue . . .
```

### Actividad 3

Hacer un programa que pida un número e indique si es primo o no.



The screenshot shows the Dev-C++ IDE with a C++ program for checking prime numbers. The code is as follows:

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 int main() {
4     int i;
5     int numero;
6     int contador=0;
7     printf("digite un numero\n");
8     scanf("%i",&numero);
9
10    for(i=1;i<=numero;i=i+1){
11        if(numero%i==0){
12            contador=contador+1;
13        }
14    }
15    if(contador>2){
16        printf("el numero no es primo\n");
17    }
18    else{
19        printf("el numero es primo\n");
20    }
21    return 0;
22 }
```

The execution output window shows the following text:

```
digite un numero
5
el numero es primo

Process exited with return value 0
Press any key to continue . . .
```

### Conclusiones:

En esta práctica me quedó más clara la forma en la que funciona cada uno de los diferentes comandos de repetición en lenguaje c (unos, mejor conocidos como bucles) y en qué casos nos pueden hacer más fácil la elaboración un código, pero si es muy importante saber bien que hace cada uno y la manera en la que se ejecutan puesto que si no será contraproducente. Por otra parte si entendí la función `#define` pero no supe cómo aplicarlo correctamente en las actividades. Finalmente la práctica me sirvió bastante para practicar estas funciones puesto que si me costó trabajo entenderles bien.