Carátula para entrega de prácticas	
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación Salas A y B

Profesor:	Alejandro Esteban Pimentel Alarcon
Asignatura:	Fundamentos de Programacion
Grupo:	135
No de Práctica(s):	9
Integrante(s):	Godínez Juárez Alondra Itati
No. de Equipo de cómputo empleado:	
No. de Lista o Brigada:	316146153
Semestre:	2020-1
Fecha de entrega:	Lunes 14 de octubre del 2019
Observaciones:	

CALIFICACIÓN: _____

Estructuras de repetición

Introducción. En la presente práctica, explicaremos el funcionamiento de las estructuras de repetición, con el fin de adentrarnos aún más en la tarea de hacer códigos en lenguaje C eficientemente empleando nuevas funciones y ahorrando líneas.
Objetivo. Elaborar programas en C para la resolución de problemas básicos que incluyan las estructuras de repetición y la directiva define.
WHILE

```
while ( <expresión_lógica> )
{
  <bloque_de_instrucciones>
}
```

Para que se ejecute el <bloque_de_instrucciones>, la condición tiene que ser verdadera. Por el contrario, si la condición es falsa, el <bloque_de_instrucciones> no se ejecuta.
El <bloque_de_instrucciones> de un bucle while puede ejecutarse cero o más veces (iteraciones). Si el <bloque_de_instrucciones> se ejecuta al menos una vez, seguirá ejecutándose repetidamente, mientras que, la condición sea verdadera. Pero, hay que tener cuidado de que el bucle no sea infinito.

Cuando el flujo de un programa llega a un bucle while, existen dos posibilidades:

1. Si la condición se evalúa a falsa, el bloque de instrucciones no se ejecuta, y el bucle while finaliza sin realizar ninguna iteración.
2. Si la condición se evalúa a verdadera, el bloque de instrucciones sí que se ejecuta y, después, se vuelve a evaluar la condición, para decidir, de nuevo, si el bloque de instrucciones se vuelve a ejecutar o no. Y así sucesivamente, hasta que, la condición sea falsa.

DO-WHILE

```
do {
        /* CÓDIGO */
} while (/* Condición de ejecución del bucle */)
```

Se utiliza para especificar un ciclo condicional que se ejecuta al menos una vez. En algunas circunstancias en las que la determinada acción se ejecutará de una o más veces, pero al menos una vez.

FOR

```
for ( <expresión_1> ; <expresión_2> ; <expresión_3> )
{
  <bloque_de_instrucciones>

}
```

El bucle for es una variante del bucle while y, al igual que éste, puede iterar cero o más veces. Sin embargo, el bucle for sólo se suele usar cuando se conoce el número exacto de veces que tiene que iterar el bucle. Éste es el caso del problema planteado en el ejemplo 1 del apartado

Instrucción while en C, en el cual, se sabe de antemano que el bucle tiene que iterar, exactamente, diez veces.

DEFINE

En C las constantes se declaran con la directiva #define, esto significa que esa constante tendrá el mismo valor a lo largo de todo el programa.
El identificador de una constante así definida será una cadena de caracteres que deberá cumplir los mismos requisitos que el de una variable (sin espacios en blanco, no empezar por un dígito numérico, etc).

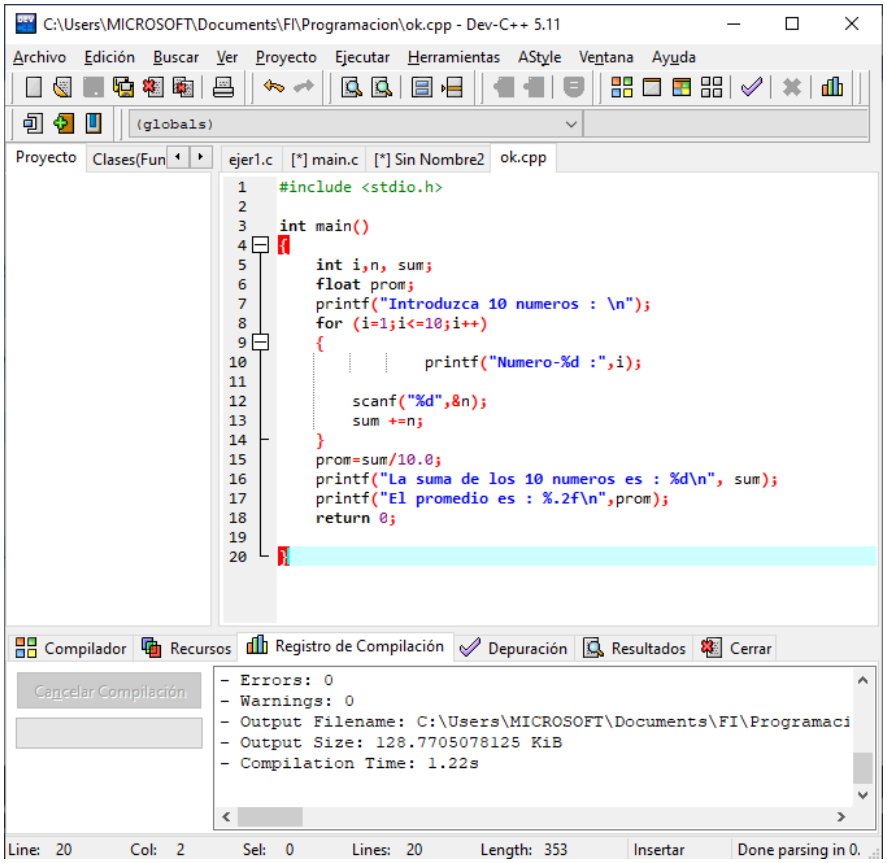
#define <constante> <secuencia_de_caracteres>

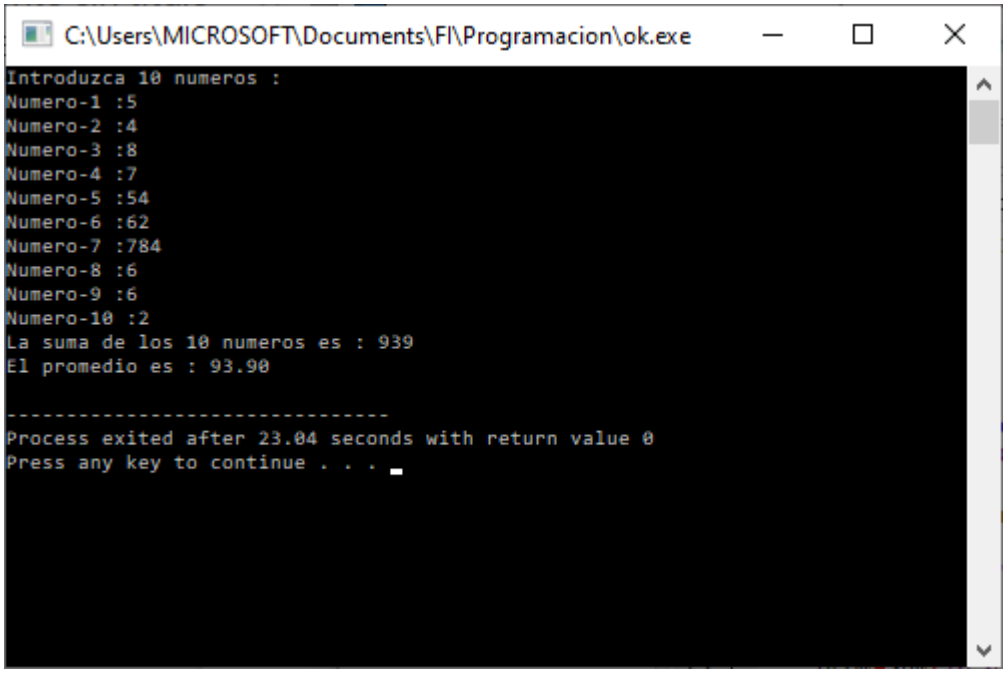
Actividades. Para cada uno de los siguientes problemas, elegir un tipo de ciclo y resolverlo. Al final, deben usar los tres tipos de ciclos y usar define por lo menos una vez.

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

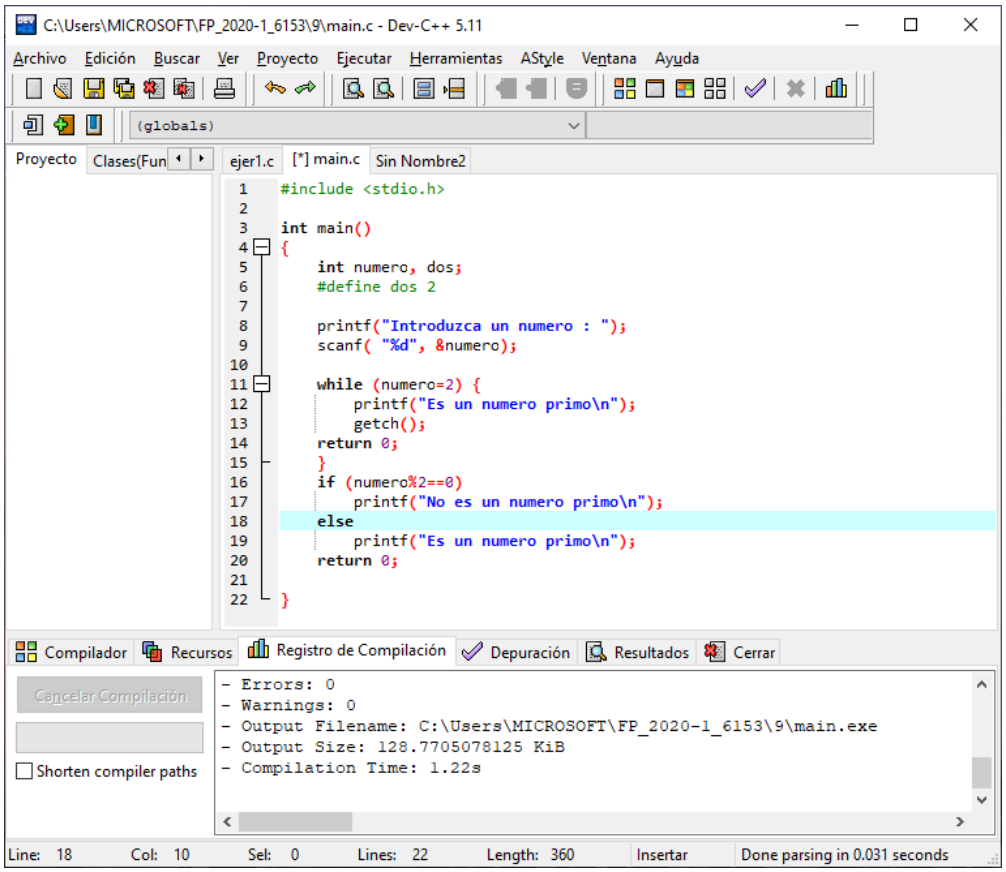
```
Documentos — -bash — 74×33
[Tunez52:Documents fp03alu16$ gcc act1.c -o act1
[Tunez52:Documents fp03alu16$ ./act1
Escriba un numero
14

   La tabla de multiplicar del numero 14 es:
   14 * 1 = 14
   14 * 2 = 28
   14 * 3 = 42
   14 * 4 = 56
   14 * 5 = 70
   14 * 6 = 84
   14 * 7 = 98
   14 * 8 = 112
   14 * 9 = 126
   14 * 10 = 140
[ Tunez52:Documents fp03alu16$ ./act1
Escriba un numero
8

   La tabla de multiplicar del numero 8 es:
   8 * 1 = 8
   8 * 2 = 16
   8 * 3 = 24
   8 * 4 = 32
   8 * 5 = 40
   8 * 6 = 48
   8 * 7 = 56
   8 * 8 = 64
   8 * 9 = 72
   8 * 10 = 80
Tunez52:Documents fp03alu16$
```

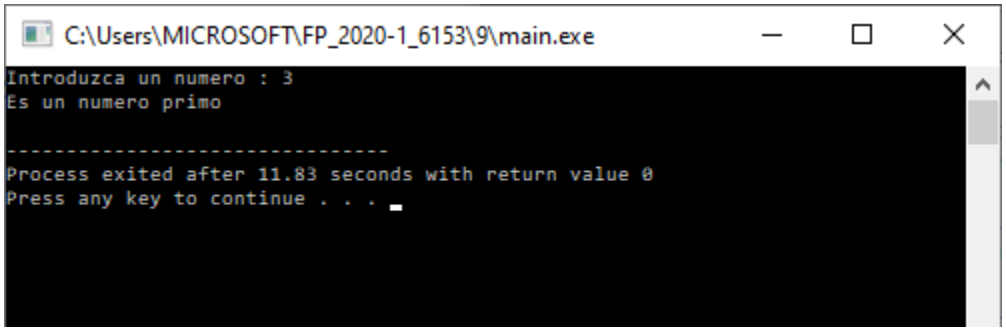
★ Hacer un programa que pida y lea 10 números y muestre su suma y su promedio.

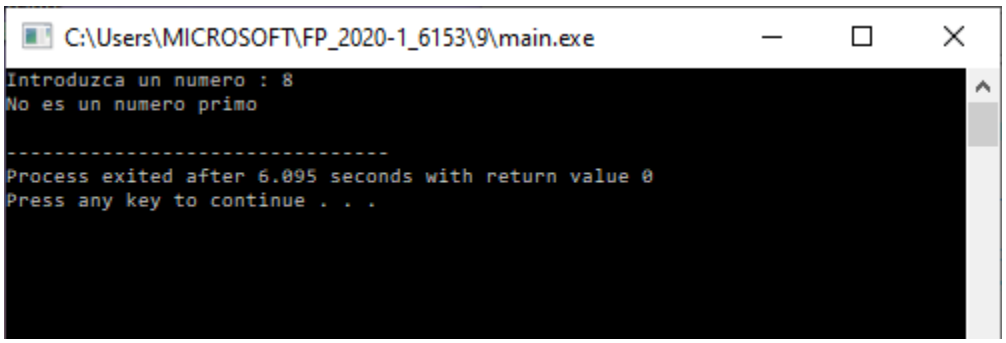




★ Hacer un programa que pida un número e indique si es primo o no.







Conclusión. Después de ver todas estas funcionalidades, vemos que sirven para fines específicos de repetición de acciones de acuerdo al fin de nuestro programas, por lo que es cuestión de nosotros, los programadores, ver si la función nos funciona o no.