

Carátula para entrega de prácticas	
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

<i>Profesor:</i>	Alejandro Pimentel
<i>Asignatura:</i>	Fundamentos de programación
<i>Grupo:</i>	3
<i>No de Práctica(s):</i>	9
<i>Integrante(s):</i>	Uno Karin Natalia
<i>No. de Equipo de cómputo empleado:</i>	16
<i>No. de Lista o Brigada:</i>	8723 #50
<i>Semestre:</i>	1
<i>Fecha de entrega:</i>	14 de octubre del 2019
<i>Observaciones:</i>	

CALIFICACIÓN: _____

PRÁCTICA 9:

ESTRUCTURAS DE REPETICIÓN.

OBJETIVO

Se elaborarán programas en lenguaje C con estructuras de repetición (WHILE, DO-WHILE y FOR) y declaraciones con define.

INTRODUCCIÓN

Bucle WHILE

Estos bucles se utilizan cuando queremos repetir la ejecución de unas sentencias un número indefinido de veces, siempre que se cumpla una condición. Se más sencillo de comprender que el bucle FOR, pues no incorpora en la misma línea la inicialización de las variables su condición para seguir ejecutándose y su actualización. Sólo se indica, como veremos a continuación, la condición que se tiene que cumplir para que se realice una iteración.

```
while (condición){  
    //sentencias a ejecutar  
}
```

Bucle DO...WHILE

El bucle do...while es una variación del bucle while visto anteriormente. Se utiliza generalmente cuando no sabemos cuantas veces se habrá de ejecutar el bucle, igual que el bucle WHILE, con la diferencia de que sabemos seguro que el bucle por lo menos se ejecutará una vez.

```
do {  
    //sentencias del bucle  
} while (condición)
```

Recopilado en: <https://desarrolloweb.com/articulos/567.php>

Bucle FOR

El bucle se ejecuta siempre una vez y al final se evalúa la condición para decir si se ejecuta otra vez el bucle o se termina su ejecución.

Esta sentencia permite realizar un bucle repetidamente en base a una condición, la cual suele estar basada en el valor de un contador que se actualiza después de cada ejecución del bucle.

```
for ( [<inicio>] ; [<condicion>] ; [<incremento>] )  
<sentencia>
```

De otros lenguajes. <inicio> e <incremento> pueden ser llamadas a funciones o sentencias de asignación. <sentencia> es del cuerpo del bucle, y puede ser cualquier bloque de código. La utilización más común suele adoptar la forma:

```
for ( <inicio> ; <condición> ; <incremento> ) {  
<sentencia> }
```

Recopilado en:

[https://www.ecured.cu/Sentencias_de_iteraci%C3%B3n_\(programaci%C3%B3n\)#Bucle_for](https://www.ecured.cu/Sentencias_de_iteraci%C3%B3n_(programaci%C3%B3n)#Bucle_for)

#define

Para expresar una constante con un nombre, la constante debe ser declarada previamente. Las constantes que se declaran en un programa escrito en lenguaje C reciben un tratamiento diferente al de la mayoría de los lenguajes de programación. En C, para representar a las constantes, se utilizan constantes simbólicas. Una constante simbólica representa (sustituye) a una secuencia de caracteres, en vez de representar a un valor (dato almacenado en memoria).

Para declarar una constante simbólica, en lenguaje C, se utiliza una nueva directiva del preprocesador:

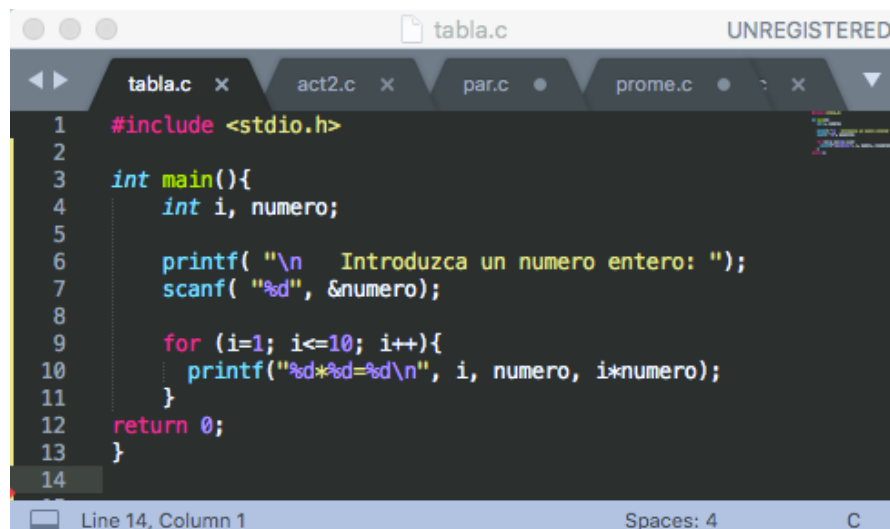
```
#define <constante> <secuencia_de_caracteres>
```

La directiva #define indica al preprocesador que debe sustituir, en el código fuente del programa, todas las ocurrencias del <nombre_de_la_constante> por la <secuencia_de_caracteres>, antes de la compilación.

Recopilado en: http://www.carlospes.com/curso_de_lenguaje_c/01_07_constantes.php

DESARROLLO/RESULTADOS

Actividad 1. Tabla de multiplicar

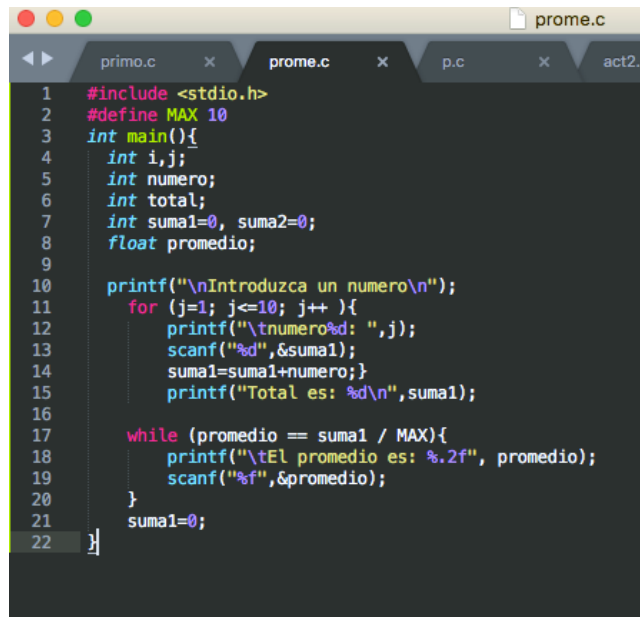


The screenshot shows a code editor window titled 'tabla.c' with a tab bar containing 'tabla.c', 'act2.c', 'par.c', and 'prome.c'. The code is as follows:

```
1  #include <stdio.h>  
2  
3  int main(){  
4      int i, numero;  
5  
6      printf( "\n  Introduzca un numero entero: ");  
7      scanf( "%d", &numero);  
8  
9      for (i=1; i<=10; i++){  
10         printf("%d*%d=%d\n", i, numero, i*numero);  
11     }  
12     return 0;  
13 }  
14
```

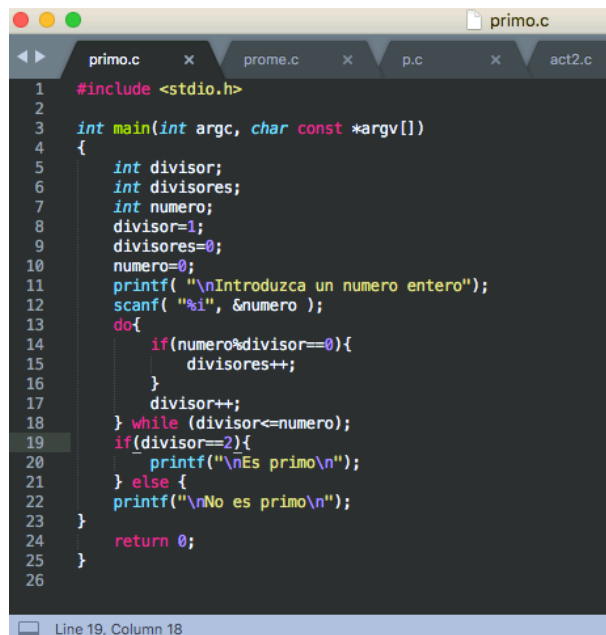
The status bar at the bottom indicates 'Line 14, Column 1', 'Spaces: 4', and 'C'.

Actividad 2. Promedio



```
1 #include <stdio.h>
2 #define MAX 10
3 int main(){
4     int i,j;
5     int numero;
6     int total;
7     int suma1=0, suma2=0;
8     float promedio;
9
10    printf("\nIntroduzca un numero\n");
11    for (j=1; j<=10; j++){
12        printf("\tnumero%d: ",j);
13        scanf("%d",&suma1);
14        suma1=suma1+numero;
15        printf("Total es: %d\n",suma1);
16
17    while (promedio == suma1 / MAX){
18        printf("\tEl promedio es: %.2f", promedio);
19        scanf("%f",&promedio);
20    }
21    suma1=0;
22 }
```

Actividad 3. Primo



```
1 #include <stdio.h>
2
3 int main(int argc, char const *argv[])
4 {
5     int divisor;
6     int divisores;
7     int numero;
8     divisor=1;
9     divisores=0;
10    numero=0;
11    printf( "\nIntroduzca un numero entero");
12    scanf( "%i", &numero );
13    do{
14        if(numero%divisor==0){
15            divisores++;
16        }
17        divisor++;
18    } while (divisor<=numero);
19    if(divisor==2){
20        printf("\nEs primo\n");
21    } else {
22        printf("\nNo es primo\n");
23    }
24    return 0;
25 }
26
```

Line 19, Column 18

CONCLUSIÓN

Podemos concluir que los programas en lenguaje C, para su repetición o iteración se requieren del uso de una de las 3 formas (WHILE, DO-WHILE y FOR) y para declaraciones con define son para dar un valor a una variable definida, escritos como fue mostrado.