


| | | |
|--|---|--|
|  | Carátula para entrega de prácticas | |
| Facultad de Ingeniería | Laboratorio de docencia | |

Laboratorios de computación salas A y B

Profesor: Alejandro Esteban Pimentel Alarcón

Asignatura: Fundamentos de programación

Grupo:

3

No de Práctica(s): 9

Integrante(s):

Crail Ávila Regina 8973

*No. de Equipo de
cómputo empleado:*

29

No. de Lista o Brigada:

9

Semestre:

2020-1

Fecha de entrega:

14/10/19

Observaciones:

En ninguna de tus actividad usas el #define.
Además, tus últimas dos actividades no son correctas.
~~En la segunda no muestras evidencias de que esté~~
funcionando el programa correctamente (no das entradas)
Y en la segunda tu algoritmo no está bien hecho.

CALIFICACIÓN: 6

ESTRUCTURAS DE REPETICIÓN

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

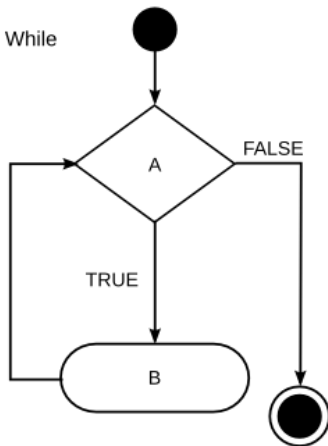
While: Es un bucle de ejecución continua mientras se cumpla la expresión colocada entre paréntesis en la cabecera del bucle. La variable de prueba tendrá que cambiar para salir del bucle. La situación podrá cambiar a expensas de una expresión dentro el código del bucle o también por el cambio de un valor en una entrada de un sensor.

A continuación se presentan ejemplos:

```
while (expresión_lógica) {  
    // Bloque de código a repetir  
    // mientras que la expresión  
    // lógica sea verdadera.  
}
```

Do while: Ésta estructura se encarga de repetir de forma cíclica un conjunto de instrucciones que se encuentren dentro del bucle, esta repetición se realizará hasta que se cumpla la condición de parada que definamos y que es evaluada con la palabra reservada while.

While (A = TRUE) Do
B
End While



```
do  
{  
    // Instrucciones del ciclo.  
} while (condicion de parada)
```

For: Se usa para repetir un bloque de sentencias encerradas entre llaves un número determinado de veces. Cada vez que se ejecutan las instrucciones del bucle se vuelve a testear la condición. La declaración for tiene tres partes separadas por (;). La inicialización de la variable local se produce una sola vez y la condición se testea cada vez que se termina la ejecución de las instrucciones dentro del bucle. Si la condición sigue cumpliéndose, las instrucciones del bucle se vuelven a ejecutar. Cuando la condición no se cumple, el bucle termina.

parenthesis

declare variable (optional)

initialize test increment or decrement

```
for(int x = 0; x < 100; x++){  
    println(x); // prints 0 to 99  
}
```

The diagram highlights the structure of a 'for' loop. A large orange arrow labeled 'parenthesis' points to the entire 'for' statement. Three smaller arrows point to the individual parts: a blue arrow for 'declare variable (optional)' pointing to 'int x = 0', a pink arrow for 'initialize' pointing to 'x = 0', a pink arrow for 'test' pointing to 'x < 100', and a pink arrow for 'increment or decrement' pointing to 'x++'.

Define: La instrucción #define nos permite declarar constantes (y algunas cosas más) de una manera rápida y sencilla. Hay que tener en cuenta que al declarar constantes con #define debemos hacerlo después de los #include para importar librerías pero antes de declarar nuestras funciones y demás. A continuación un ejemplo:

#define MAX 5

ACTIVIDAD

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.

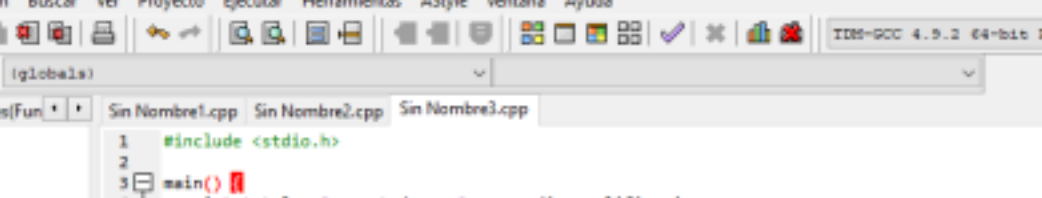
- Hacer un programa que pida un número y muestre su tabla de multiplicar (hasta el 10).

```
Sin Nombre1.cpp
1  #include <stdio.h>
2
3  int main()
4  {
5      char seguir;
6      int i, numero;
7
8      do
9      {
10         printf( "\n Ingrese un numero no decimal: ", 163);
11         scanf( "%d", &numero );
12
13         printf( "\n Tabla de multiplicar del %d es:\n", numero );
14         /* Inicio de la accion */
15
16         for ( i = 1 ; i <= 10 ; i++ )
17             printf( " \n %d * %d = %d; i; numero; i * numero ");
18         /* Fin de la accion */
19
20         printf( " \n\n %cMostrar otra tabla, (s/n)? ", 168);
21         fflush( stdin );
22         scanf( "%c", &seguir );
23     }while ( seguir != 'n' );
24
25     return 0;
26 }
```

C:\Users\Usuario\Desktop\REGINA\Sin Nombre1.exe

```
Ingrese un numero no decimal: 8
Tabla de multiplicar del 8 es:
1879308848 * 1879333184 = 6480464; i; numero; i * numero
1879308848 * 1879333184 = 6480464; i; numero; i * numero
1879308848 * 1879333184 = 6480464; i; numero; i * numero
1879308848 * 1879333184 = 6480464; i; numero; i * numero
1879308848 * 1879333184 = 6480464; i; numero; i * numero
1879308848 * 1879333184 = 6480464; i; numero; i * numero
1879308848 * 1879333184 = 6480464; i; numero; i * numero
1879308848 * 1879333184 = 6480464; i; numero; i * numero
1879308848 * 1879333184 = 6480464; i; numero; i * numero
1879308848 * 1879333184 = 6480464; i; numero; i * numero
¿Mostrar otra tabla, (s/n)? : 5
Ingrese un numero no decimal: 5
Tabla de multiplicar del 5 es:
1879308848 * 1879333184 = 6480464; i; numero; i * numero
1879308848 * 1879333184 = 6480464; i; numero; i * numero
1879308848 * 1879333184 = 6480464; i; numero; i * numero
1879308848 * 1879333184 = 6480464; i; numero; i * numero
1879308848 * 1879333184 = 6480464; i; numero; i * numero
```

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



The screenshot shows the Dev-C++ IDE with the file 'Sin Nombre3.cpp' open. The code is as follows:

```
1  #include <stdio.h>
2
3  main()
4  {
5      int total = 0, contador = 0, promedio, calificacion;
6
7      while (contador != 10) {
8          printf("Insertar los numeros, \n");
9          scanf("%d", &calificacion);
10         total += calificacion;
11         contador += 1;
12     }
13     if(contador != 0){
14         promedio = total / 10;
15         printf("Promedio total: %d\n", promedio);
16     }
17     return 0;
18 }
```



Esta muestra
de funcionamiento
está incompleta

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

```
Sin Nombre1.cpp Sin Nombre2.cpp
1  #include <stdio.h>
2  #include <iostream>
3
4  using namespace std;
5  int main() {
6      int divisor = 1, divisores = 0, num = 0;
7      cout<<"Ponga un numero: ";
8      cin>> num;
9      do{
10         if(num % divisor == 0){
11             divisores++;
12         }
13         divisor++;
14     }while(divisor <= num);
15     if(divisores == 2){
16         cout<<"n-> El numero "<<num<<" Primo ";
17         cout<<"n-> El numero "<<num<<" No es primo ";
18     }
19     return 0;
20 }
```

Tu programa no distingue entre los dos tipos de salida. Siempre que da una respuesta muestra ambos letreros, tanto que es primo como que no

```
C:\Users\Usuario\Desktop\REGINA\Sin Nombre2.exe
Ponga un numero: 2
n-> El numero 2 Primo n-> El numero 2 No es primo
-----
Process exited after 4.746 seconds with return value 0
Presione una tecla para continuar . . .
```

CONCLUSIÓN:

Para terminar considero que ésta práctica ha servido mucho ya que hemos aprendido como hacer programas que se repiten así como conocer su estructura, analizarla, etc...Ahora sabemos las diferencias que existen entre do, while, do-while y también hemos aprendido a usar el define que es muy importante ya que su función es declarar variables y esto como ingenieros industriales nos sirve mucho para programar, etc...