



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).

A screenshot of a code editor window titled 'act1.c' with a 'UNREGISTERED' label in the top right corner. The editor shows a C program that prompts the user for a number and then displays its multiplication table from 1 to 10. The code is as follows:

```
1  #include <stdio.h>
2  int main () {
3
4      int num1, tabla;
5
6      printf("Escriba un numero\n");
7      scanf("%i", &num1);
8
9      printf( "\n  La tabla de multiplicar del numero %i es:\n", num1);
10
11     for ( tabla = 1 ; tabla <= 10 ; tabla++ )
12     {
13         printf( "  %i * %i = %i \n ", num1, tabla, num1*tabla);
14     }
15
16     return 0;
17 }
18
```

The status bar at the bottom indicates 'Line 18, Column 1', 'Tab Size: 4', and the language is 'C'.

```
Documents — -bash — 74x33
[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.

The screenshot shows the Dev-C++ 5.11 IDE with a C++ program named `ok.cpp` open. The program prompts the user to enter 10 numbers, calculates their sum and average, and displays the results. The compilation output at the bottom shows 0 errors and 0 warnings, with an output file size of 128.7705078125 KiB and a compilation time of 1.22s.

```
C:\Users\MICROSOFT\Documents\FI\Programacion\ok.cpp - Dev-C++ 5.11
Archivo Edición Buscar Ver Proyecto Ejecutar Herramientas AStyle Ventana Ayuda
(globals)
Proyecto Clases(Fun) ejer1.c [*] main.c [*] Sin Nombre2 ok.cpp
1 #include <stdio.h>
2
3 int main()
4 {
5     int i,n, sum;
6     float prom;
7     printf("Introduzca 10 numeros : \n");
8     for (i=1;i<=10;i++)
9     {
10         printf("Numero-%d :",i);
11
12         scanf("%d",&n);
13         sum +=n;
14     }
15     prom=sum/10.0;
16     printf("La suma de los 10 numeros es : %d\n", sum);
17     printf("El promedio es : %.2f\n",prom);
18     return 0;
19
20
Compilador Recursos Registro de Compilación Depuración Resultados Cerrar
Cancelar Compilación
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\MICROSOFT\Documents\FI\Programaci
- Output Size: 128.7705078125 KiB
- Compilation Time: 1.22s
Line: 20 Col: 2 Sel: 0 Lines: 20 Length: 353 Insertar Done parsing in 0. ...
```

```
C:\Users\MICROSOFT\Documents\FI\Programacion\ok.exe

Introduzca 10 numeros :
Numero-1 :5
Numero-2 :4
Numero-3 :8
Numero-4 :7
Numero-5 :54
Numero-6 :62
Numero-7 :784
Numero-8 :6
Numero-9 :6
Numero-10 :2
La suma de los 10 numeros es : 939
El promedio es : 93.90

-----
Process exited after 23.04 seconds with return value 0
Press any key to continue . . .
```

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

```
C:\Users\MICROSOFT\FP_2020-1_6153\9\main.c - Dev-C++ 5.11
Archivo Edición Buscar Ver Proyecto Ejecutar Herramientas AStyle Ventana Ayuda
(globals)
Proyecto Clases(Fun) ejer1.c [*] main.c Sin Nombre2
1 #include <stdio.h>
2
3 int main()
4 {
5     int numero, dos;
6     #define dos 2
7
8     printf("Introduzca un numero : ");
9     scanf("%d", &numero);
10
11     while (numero=2) {
12         printf("Es un numero primo\n");
13         getch();
14         return 0;
15     }
16     if (numero%2==0)
17         printf("No es un numero primo\n");
18     else
19         printf("Es un numero primo\n");
20     return 0;
21 }
22

Compilador Recursos Registro de Compilación Depuración Resultados Cerrar
Cancelar Compilación
Shorten compiler paths
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\MICROSOFT\FP_2020-1_6153\9\main.exe
- Output Size: 128.7705078125 KiB
- Compilation Time: 1.22s
Line: 18 Col: 10 Sel: 0 Lines: 22 Length: 360 Insertar Done parsing in 0.031 seconds
```

```
C:\Users\MICROSOFT\FP_2020-1_6153\9\main.exe
Introduzca un numero : 3
Es un numero primo

-----
Process exited after 11.83 seconds with return value 0
Press any key to continue . . .
```

```
C:\Users\MICROSOFT\FP_2020-1_6153\9\main.exe
Introduzca un numero : 8
No es un numero primo

-----
Process exited after 6.095 seconds with return value 0
Press any key to continue . . .
```

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.