	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: *Rodríguez Espino Claudia Ing.*

Asignatura: *Fundamentos de Programación*

Grupo: *1104*

No de Práctica(s): *9*

Integrante(s): *Santa Rosa Ortiz Thelma Jazmín.*

No. de Equipo de cómputo empleado *52*

Semestre: *1°*

Fecha de entrega: *29 de septiembre del 2018*

Obervaciones:

CALIFICACIÓN: _____

Practica 9. Estructuras de Repetición.

Objetivo:

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

Desarrollo:

Las estructuras de repetición son las llamadas estructuras cíclicas, iterativas o de bucles. Permiten ejecutar un conjunto de instrucciones de manera repetida (o cíclica) mientras que la expresión lógica a evaluar se cumpla (sea verdadera).

En lenguaje C existen tres estructuras de repetición: while, do-while y for. Las estructuras while y do-while son estructuras repetitivas de propósito general.

Estructura de control repetitiva while

La estructura repetitiva (o iterativa) while primero valida la expresión lógica y si ésta se cumple (es verdadera) procede a ejecutar el bloque de instrucciones de la estructura, el cual está delimitado por las llaves {}. Si la condición no se cumple se continúa el flujo normal del programa sin ejecutar el bloque de la estructura, es decir, el bloque se puede ejecutar de cero a ene veces. Su sintaxis es la siguiente:

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

Si el bloque de código a repetir consta de una sola sentencia, entonces se pueden omitir las llaves.

Estructura de control repetitiva do-while

do-while es una estructura cíclica que ejecuta el bloque de código que se encuentra dentro de las llaves y después valida la condición, es decir, el bloque de código se ejecuta de una a ene veces. Su sintaxis es la siguiente:

```
do {
        /*
        Bloque de código que se ejecuta por lo
        menos una vez y se repite mientras la
        expresión lógica sea verdadera.
        */
} do (expresión_lógica);
```

Si el bloque de código a repetir consta de una sola sentencia, entonces se pueden omitir las llaves. Esta estructura de control siempre termina con el signo de puntuación ';'.

Estructura de control de repetición for

Lenguaje C posee la estructura de repetición for la cual permite realizar repeticiones cuando se conoce el número de elementos que se quiere recorrer. La sintaxis que generalmente se usa es la siguiente:

```
For (inicialización;expresión_lógica;operaciones por iteración) {
        /*
        Bloque de código a ejecutar
        */
}
```

La estructura for ejecuta 3 acciones básicas antes o después de ejecutar el bloque de código. La primera acción es la inicialización, en la cual se pueden definir variables e inicializar sus valores; esta parte solo se ejecuta una vez cuando se ingresa al ciclo y es opcional. La segunda acción consta de una expresión lógica, la cual se evalúa y, si ésta es verdadera, ejecuta el bloque de código, si no se cumple se continúa la ejecución del programa; esta parte es opcional. La tercera parte consta de un conjunto de operaciones que se realizan cada vez que termina de ejecutarse el bloque de código y antes de volver a validar la expresión lógica; esta parte también es opcional.

Define

Las líneas de código que empiezan con # son directivas del preprocesador, el cual se encarga de realizar modificaciones en el texto del código fuente, como reemplazar un símbolo definido con #define por un parámetro o texto, o incluir un archivo en otro archivo con #include.

define permite definir constantes o literales; se les nombra también como constantes simbólicas. Su sintaxis es la siguiente:

#define <nombre><valor>

Al definir la constante simbólica con #define, se emplea un nombre y un valor. Cada vez que aparezca el nombre en el programa se cambiará por el valor definido. El valor puede ser numérico o puede ser texto.

Break

Algunas veces es conveniente tener la posibilidad de abandonar un ciclo. La proposición break proporciona una salida anticipada dentro de una estructura de repetición, tal como lo hace en un switch. Un *break* provoca que el ciclo que lo encierra termine inmediatamente.

Continue

La proposición *continue* provoca que inicie la siguiente iteración del ciclo de repetición que la contiene.

Realizamos 6 programas en la clase, utilizando las estructuras de control: while, do while y for. También utilizamos la directiva *define* la hace mucho más fácil definir un printf o scanf.

1 A 100 CON WHILE Y DEFINE

13.1.2 1a100 while define.cpp | 13.2.2 1a100 do while define.cpp | 13.3 1a100 for define.cpp | 7.1 Exponencial define

```
1   #include<stdio.h>
2   #include<conio.h>
3   #define p printf
4   int a;
5
6   main()
7   {
8       a=1;
9       while(a<=100)
10          {
11          p("%d\t", a);
12          a++;
13          }
14      getch();
15
16  }
```

C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe

```
1       2       3       4       5       6       7       8       9       10
11      12      13      14      15      16      17      18      19      20
21      22      23      24      25      26      27      28      29      30
31      32      33      34      35      36      37      38      39      40
41      42      43      44      45      46      47      48      49      50
51      52      53      54      55      56      57      58      59      60
61      62      63      64      65      66      67      68      69      70
71      72      73      74      75      76      77      78      79      80
81      82      83      84      85      86      87      88      89      90
91      92      93      94      95      96      97      98      99      100
```

1 A 100 CON DO-WHILE Y DEFINE

13.2.2 1a100 do while define.cpp | 13.3 1a100 for define.cpp | 7.1 Exponencial define.cpp | 5.1 Factorial define.cpp

```
1   #include<stdio.h>
2   #include<conio.h>
3   #define p printf
4   int a;
5
6   main()
7   {
8       a=1;
9       do
10          {
11          p("%d\t", a);
12          a=a+1;
13          }
14      while(a<=100);
15      getch();
```

C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe

```
1       2       3       4       5       6       7       8       9       10
11      12      13      14      15      16      17      18      19      20
21      22      23      24      25      26      27      28      29      30
31      32      33      34      35      36      37      38      39      40
41      42      43      44      45      46      47      48      49      50
51      52      53      54      55      56      57      58      59      60
61      62      63      64      65      66      67      68      69      70
71      72      73      74      75      76      77      78      79      80
81      82      83      84      85      86      87      88      89      90
91      92      93      94      95      96      97      98      99      100
```

1 A 100 CON FOR Y DEFINE

13.3 1a100 for define.cpp | 7.1 Exponencial define.cpp | 5.1 Factorial define.cpp | Tabla de multiplicar for.cpp

```
1   #include<stdio.h>
2   #include<conio.h>
3   #define p printf
4   int a;
5
6   main()
7   {
8       a=1;
9       for (a=1;a<=100;a++)
10          {
11          p("%d\t", a);
12          }
13      getch();
```

```
C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe
1       2       3       4       5       6       7       8       9       10
11      12      13      14      15      16      17      18      19      20
21      22      23      24      25      26      27      28      29      30
31      32      33      34      35      36      37      38      39      40
41      42      43      44      45      46      47      48      49      50
51      52      53      54      55      56      57      58      59      60
61      62      63      64      65      66      67      68      69      70
71      72      73      74      75      76      77      78      79      80
81      82      83      84      85      86      87      88      89      90
91      92      93      94      95      96      97      98      99      100
```

EXPONENCIAL CON FOR Y DEFINE.

7.1 Exponencial define.cpp | 5.1 Factorial define.cpp | Tabla de multiplicar for.cpp

```
1   #include<stdio.h>
2   #include<math.h>
3   #include<conio.h>
4   #define p printf
5   #define s scanf
6       int x,y,a;
7   main()
8   {
9       p("Dame el numero a elevar\n");
10      s("%d",&y);
11      for(a=0;a<=10;a++)
12      {
13          x=pow(y,a);
14          p("El numero elevado %d elevado a la potencia %d es: %d \n",y,a,x);
15      }
16      getch();
```

```
C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe
Dame el numero a elevar
5
El numero elevado 5 elevado a la potencia 0 es: 1
El numero elevado 5 elevado a la potencia 1 es: 5
El numero elevado 5 elevado a la potencia 2 es: 25
El numero elevado 5 elevado a la potencia 3 es: 125
El numero elevado 5 elevado a la potencia 4 es: 625
El numero elevado 5 elevado a la potencia 5 es: 3125
El numero elevado 5 elevado a la potencia 6 es: 15625
El numero elevado 5 elevado a la potencia 7 es: 78125
El numero elevado 5 elevado a la potencia 8 es: 390625
El numero elevado 5 elevado a la potencia 9 es: 1953125
El numero elevado 5 elevado a la potencia 10 es: 9765625
```

FACTORIAL CON DO-WHILE Y DEFINE.

[*] 5.1 Factorial define.cpp | Tabla de multiplicar for.cpp

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define p printf
#define s scanf
int x,m=1,f,y;
main()
{
    do
    {
        p("Este programa calcula el factorial de un numero.\n\n\tDame un numero: ");
        s("%d",&x);
        if(x>0 && x<=5)
        {
            for(f=1;f<=x;f++);
        {
            m=(f*m);
        }
        p("\tEl factorial del numero %d es %d.",x,m);
        }
        else
        p("\tNo puedo sacar el factorial.");
        p("\n\tQuieres repetir el ejercicio? Si=1, No=0\n\t-");
        s("%d",&y);
        system("cls");
    }
    while(y==1);
    p("\n\t\t\t--- Adios ---");
    getch();
```

C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe

```
Este programa calcula el factorial de un numero.

        Dame un numero: 5
        El factorial del numero 5 es 6.
        Quieres repetir el ejercicio? Si=1, No=0
        -
```

TABLA DE MULTIPLLICAR CON FOR Y DEFINE

Tabla de multiplicar for.cpp

```
#include <stdio.h>
#include <conio.h>
#define p printf
int a,b,c;
main()
{
 p("\a----- Tabla de multiplicar -----\n");

 for(a=0;a<=10;a++)
 {
    p("La tabla de multiplicar %d", a);
    for (b=1;b<=10;b++)
    {
        c=a*b;

         p("%d x %d = %d\n",a,b,c);
    }
 }
 getch();
```

H:\Programacion\Tabla de multiplicar for.exe

```
La tabla de multiplicar 88 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
8 x 6 = 48
8 x 7 = 56
8 x 8 = 64
8 x 9 = 72
8 x 10 = 80
La tabla de multiplicar 99 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90
La tabla de multiplicar 1010 x 1 = 10
10 x 2 = 20
```

Conclusión.

Esta práctica se me hizo muy importante ya que aprendimos a elaborar programas en C para la resolución de problemas usando estructuras de repetición como while, do while y for, a demás de incluir la directiva define.

Me di cuenta de que esta mejor utilizar la directiva define si se llegara a hacer un programa muy extenso porque escribimos mucho menos, mientras cuidemos que no se repita la letra con nuestras variables.