
	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	1/9
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Guía práctica de estudio 08: Estructuras de repetición



Elaborado por	Actualizado por:	Revisado por:
M.C. Edgar E. García Cano Ing. Jorge A. Solano Gálvez	M.C. Cintia Quezada Reyes	M.C. Laura Sandoval Montaño

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	2/9
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Guía de práctica de estudio 08: Estructuras de repetición

Objetivo:

El alumno elaborará programas en C para la resolución de problemas básicos que incluyan las estructuras de repetición.


Actividades:

- Elaborar un programa que utilice la estructura *while* en la solución de un problema
- Elaborar un programa que requiera el uso de la estructura *do-while* para resolver un problema. Hacer la comparación con el programa anterior para distinguir las diferencias de operación entre *while* y *do-while*.
- Resolver un problema dado por el profesor que utilice la estructura *for* en lugar de la estructura *while*.

Introducción

Las estructuras de repetición son las llamadas también 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.

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	3/9
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Licencia GPL de GNU

El software presente en esta práctica es libre bajo la licencia GPL de GNU, es decir, se puede modificar y distribuir mientras se mantenga la licencia GPL.

```

/*
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 *
 * Author: Jorge A. Solano
 *
 */

```


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 { } y regresa a validar la condición nuevamente, esto lo realiza mientras la condición sea verdadera. Cuando 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 un determinado número de veces. Su sintaxis es la siguiente:

```

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

```

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	4/9
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

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

Códigos (estructura de repetición *while*)

Este programa genera la tabla de multiplicar de un número dado. El número se lee desde la entrada estándar (teclado).

Programa1.c

```
#include <stdio.h>

int main()
{
    int num, cont = 0;
    printf("\a----- Tabla de multiplicar ----- \n");
    printf("Ingrese un número: \n");
    scanf("%d", &num);
    printf("La tabla de multiplicar del %d es:\n", num);
    while (++cont <= 10)
        printf("%d x %d = %d\n", num, cont, num*cont);
    return 0;
}
```


Este programa genera un ciclo infinito.

Programa2.c

```
#include <stdio.h>

int main()
{
    /* Al igual que en la estructura if-else, 0 -> falso y diferente de 0 -> verdadero.
       El siguiente es un ciclo infinito porque la condición siempre es verdadera.
       Así mismo, debido a que el ciclo consta de una sola línea, las llaves { } son
       opcionales.*/

    while (100)
    {
        printf("Ciclo infinito.\nPara terminar el ciclo presione ctrl + c.\n");
    }
    return 0;
}
```

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	5/9
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

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 un determinado número de 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.
    */
} while (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 punto y coma (;).

Código (estructura de repetición *do-while*)

Este programa obtiene el promedio de calificaciones ingresadas por el usuario. Las calificaciones se leen desde la entrada estándar (teclado). La inserción de calificaciones termina cuando el usuario presiona una tecla diferente de 'S' o 's'.

Programa3.c

```
#include <stdio.h>
int main ()
{
    char op = 'n';
    double sum = 0, calif = 0;
    int veces = 0;
    do
    {
        printf("\tSuma de calificaciones\n");
        printf("Ingrese la calificación:\n");
        scanf("%lf", &calif);
        veces++;
        sum = sum + calif;
        printf("¿Desea sumar otra? S/N\n");
        setbuf(stdin, NULL); // limpia el buffer del teclado
        scanf("%c", &op);
        getchar();
    }
    while (op == 'S' || op == 's');
    printf("El promedio de las calificaciones ingresadas es: %lf\n", sum/veces);
    return 0;
}
```


	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	6/9
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Código (estructura de repetición *do-while*)

Este programa genera una calculadora básica

Programa4.c

```
#include <stdio.h>
int main ()
{
    int op, uno, dos;
    do
    {
        printf(" --- Calculadora ---\n");
        printf("\n¿Qué desea hacer\n");
        printf("1) Sumar\n");
        printf("2) Restar\n");
        printf("3) Multiplicar\n");
        printf("4) Dividir\n");
        printf("5) Salir\n");
        scanf("%d",&op);
        switch(op)
        {
            case 1:
                printf("\tSumar\n");
                printf("Introduzca los números a sumar separados por comas\n");
                scanf("%d, %d",&uno, &dos);
                printf("%d + %d = %d\n", uno, dos, (uno + dos));
                break;
            case 2:
                printf("\tRestar\n");
                printf("Introduzca los números a restar separados por comas\n");
                scanf("%d, %d",&uno, &dos);
                printf("%d - %d = %d\n", uno, dos, (uno - dos));
                break;
            case 3:
                printf("\tMultiplicar\n");
                printf("Introduzca los números a multiplicar separados por comas\n");
                scanf("%d, %d",&uno, &dos);
                printf("%d * %d = %d\n", uno, dos, (uno * dos));
                break;
            case 4:
                printf("\tDividir\n");
                printf("Introduzca los números a dividir separados por comas\n");
                scanf("%d, %d",&uno, &dos);
                printf("%d / %d = %.21f\n", uno, dos, ((double)uno / dos));
                break;
            case 5:
                printf("\tSalir\n");
                break;
            default:
                printf("\tOpción inválida.\n");
        }
    }
    while (op != 5);
}
```

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	7/9
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Estructura de control de repetición *for*

El lenguaje C posee la estructura de repetición *for* la cual permite realizar repeticiones cuando generalmente el control de la repetición está definido sobre una variable contador. 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, dos antes y una 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 acción 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 acción es opcional. La tercera acción 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 acción también es opcional.


Código (estructura de repetición *for*).

Este programa genera un arreglo unidimensional de 5 elementos y accede a cada elemento del arreglo a través de un ciclo *for*.

Programa5.c

```
#include <stdio.h>
int main ()
{
    int enteroNumAlumnos = 5;
    float realCalif = 0.0, realPromedio = 0.0;
    printf("\tPromedio de calificaciones\n");
    for (int indice = 0 ; indice < enteroNumAlumnos ; indice++)
    {
        printf("\nIngrese la calificación del alumn %d\n", indice+1);
        scanf("%f",&realCalif);
        realPromedio += realCalif;
    }

    printf("\nEl promedio de las calificaciones ingresadas es: %f\n",
    realPromedio/enteroNumAlumnos);
    return 0;
}
```

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	8/9
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

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.

Código (*define*)


Este programa define un valor por defecto para el tamaño del arreglo de tal manera que si el tamaño de éste cambia, solo se debe modificar el valor de la constante MAX.

Programa5.c

```
#include <stdio.h>
#define MAX 5
int main ()
{
    int arreglo[MAX], cont;
    for (cont=0; cont<MAX; cont++)
    {
        printf("Ingrese el valor %d del arreglo: ", cont+1);
        scanf("%i", &arreglo[cont]);
    }

    printf("El valor ingresado para cada elemento del arreglo es:\n[");
    for (cont=0; cont<MAX; cont++)
    {
        printf("%d\t", arreglo[cont]);
    }
    printf("]\n");
    return 0;
}
```

Cuando se compila el programa, se reemplazan la palabra MAX por el valor definido para la misma. Esto permite que, si el tamaño del arreglo cambia, solo se tiene que

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	9/9
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

modificar el valor definido para MAX y en automático todos los arreglos y el recorrido de los mismos adquieren el nuevo valor (mientras se use MAX para definir el o los arreglos y para realizar los recorridos).

Bibliografía



El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, segunda edición, USA, Pearson Educación 1991.