	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	Marco Antonio Martinez Quintana
Asignatura:	Fundamentos de Programación
Grupo:	03
No de Práctica(s):	Practica N°11
Integrante(s):	Torres Gracian Christian Ivan
No. de Equipo de cómputo empleado:	No Aplica
No. de Lista o Brigada:	55
Semestre:	Semestre N°1
Fecha de entrega:	
Observaciones:	

CALIFICACIÓN: _____

Guía de práctica de estudio 09: Estructuras de repetición

Objetivo:

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

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*.
- Usar la directiva define para elaboración de código versátil.

Introducción

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.

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 {}. 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.

Código (estructura de repetición while)

```c
#include <stdio.h>

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

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;
}
```

Código (estructura de repetición while)

```c
#include <stdio.h>
/*
    Este programa genera un ciclo infinito.
*/
int main(){

    // Al igual que en la estructura if-else
    //                          0 -> falso
    // 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;
}
```

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.
    */
} 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 ';'.

Código (estructura de repetición do-while)

```
#include <stdio.h>
/*
    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'.
*/
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;
}
```

Código (estructura de repetición do-while)

```
 #include <stdio.h>
/*      Este programa genera una calculadora básica. */
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 = %.2lf\n", uno, dos, ((double)uno / dos));
                break;
            case 5:
                printf("\tSalir\n");
                break;
            default:
                printf("\tOpción inválida.\n");
        }
    } while (op != 5);
```

```
    return 0;
}
```

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.

Código (estructura de repetición for)

```
#include <stdio.h>
/*      Este programa genera un arreglo unidimensional de 5 elementos y
 *      accede a cada elemento del arreglo a través de un ciclo for.
*/
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;
}
```

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)

```
#include <stdio.h>
#define MAX 5

/*
 *      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.
 *      */



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 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).

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.

Código (break)

```
#include <stdio.h>

/*
 * Este programa hace una suma de números. Si la suma rebasa la cantidad
 * de 50 el programa se detiene.
 * */

#define VALOR_MAX 5

int main (){
        int enteroSuma =  0;
        int enteroNumero = 0;
        int enteroContador = 0;
        while (enteroContador < VALOR_MAX){
                printf("Ingrese un número:");
                scanf("%d", &enteroNumero);
                enteroSuma += enteroNumero;
                enteroContador++;
                if (enteroSuma > 50){
                        printf("Se rebasó la cantidad límite.\n");
                        break;
```

```
            }
        }
        printf("El valor de la suma es: %d\n", enteroSuma);

        return 0;
}
```

Cuando se compila el programa, MAX se sustituye por 5.

Continue

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

```
#include <stdio.h>

/*
 * Este programa obtiene la suma de un LIMITE de números pares ingresados
 * */



#define LIMITE 5



int main (){

        int enteroContador = 1;

        int enteroNumero = 0;
        int enteroSuma = 0;
        while (enteroContador <= LIMITE){
                printf("Ingrese número par %d:", enteroContador);
                scanf("%d",&enteroNumero);

                if (enteroNumero%2 != 0){
                        printf("El número insertado no es par.\n");
                        continue;
                }

                enteroSuma += enteroNumero;
                enteroContador++;
        }

        printf("La suma de los números es: %d\n", enteroSuma);

        return 0;
}
```

Código (continue)

Resultados

```c
#include<stdio.h>
int main()
{
    // variables
    char u=163, sp=168, e=130;
    int n,res;

    //Bienvendia
    printf("\n\n\t\t\t\t Factorial de un n%cmero\n\n ",u);

    //Solicitar numero
    printf("\n\t %cDe qu%c n%cmero desaeas saber su factorial:  ",sp,e,u);
    scanf("%d",&n);

    //Operacion
    res=1;

    for(int i=1;i<=n;i++)
    {
        res=res*i;
    }



    //Mostrar resultado
    printf("\n\t %cl factorial del n%cmero dado es: %i",e,u,res);

    return 0;
}
```

```c
#include <stdio.h>
int main()
{
    //Variables
    char a=160;
    char u=163;
    char sp=168;
    int n,i,res;

    //Bienvenida
    printf("\n\n\t\t\t\t Suma de los primeros n n%cmeros \n\n",u);

    // Solisitar numeros
    printf("\t%cCu%cntos n%cmeros deseas  sumar?   ",sp,a,u);
    scanf("%i",&n);

    //Suma de los n numeros
    res=0;
    i=1;
    while(i<=n)
    {
        res=res+i;
        i++;
    }

    //Resultado
    printf("\n\t La suma de los primeros %d n%cmeros es: %i",n,u,res);

    return 0;
}
```

```c
#include<stdio.h>
int main()
{
    // variables
    char u=163, sp=168, e=130;
    int n,i,res;

    //Bienvendia
    printf("\n\n\t\t\t\t Factorial de un n%cmero\n\n ",u);

    //Solicitar numero
    printf("\n\t %cDe qu%c n%cmero desaeas saber su factorial:  ",sp,e,u);
    scanf("%d",&n);

    //Operacion
    res=1;
    i=1;

    while(i<=n)
    {
        res=res*i;
        i++;
    }


    //Mostrar resultado
    printf("\n\t %cl factorial del n%cmero dado es: %i",e,u,res);

    return 0;
}
```

Bibliografía

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