	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Adrián Ulises Mercado Martínez

Asignatura: Fundamentos de Programación

Grupo: 15

No de Práctica(s): 9

Integrante(s): Walls Chávez Luis Fernando y Gutiérrez Santiago Diego.

No. de Equipo de cómputo empleado:

No. de Lista o Brigada:

Semestre: 2020-I

Fecha de entrega: 17/10/19

Observaciones: No listamos el equipo de cómputo empleado ya que ambos tuvimos un retardo y no ingresamos al laboratorio.

CALIFICACIÓN: _____

Introducción:

Las estructuras de repetición son las llamadas estructuras cíclicas. Permiten ejecutar un conjunto de instrucciones de manera repetida, mientras que la expresión lógica a evaluar se cumpla. En lenguaje C existen tres estructuras de repetición:

- While
- Do-while
- For

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) {
Instrucción a cumplir que es verdadera}

Do-while:

Es una estructura cíclica que permite realizar repeticiones cuando se conoce el número de elementos que se quiere recorrer. La sintaxis que generalmente se usa es la siguiente 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 {
Instrucción que se cumple por lo menos una vez}
while (expresión lógica);

For:

Ejecuta 3 acciones básicas antes o después de ejecutar el bloque de código.

1. 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.
2. 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.
3. 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.

for (inicialización; expresión lógica; operaciones por iteración) {Código a ejecutar}

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

#define<nombre><Valor>

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.

Desarrollo:

ENTERO SUMA)

```c
#include <stdio.h>
#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;
}
```

SUMA PAR/IMPAR)

```c
#include <stdio.h>
/*
* Este programa obtiene la suma de un LIMITE de números pa
*  */
#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;
}
```

ARREGLO)

```c

#include <stdio.h>

int main (){
int enteroNumAlumnos = 5;
float realCalif = 0, realPromedio = 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;
}
```

ARREGLO CON DEFINE)

```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;
}
```

CICLO INFINITO)

```c
#include <stdio.h>

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

CALCULADORA)

```c
#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;
}
```

PROMEDIO)

```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);
     scanf("%c",&op);
     getchar();
  }
  while (op == 'S' || op == 's');
      printf("El promedio de las calificaciones ingresadas es: %lf\n", sum/veces);
      return 0;
}
```

TABLA DE MULTIPLICAR)

```c
#include <stdio.h>
#include <stdlib.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;
}
```

Conclusiones:

Diego Santiago Gutiérrez:

Los ciclos son una parte fundamental de la programación; ya que, gracias a ellos podemos realizar procesos de una forma sencilla, ordenada y según a los criterios establecidos. Como se vio en ésta práctica, recurrimos a los 3 ciclos repetitivos para poder realizar tareas las cuales era necesario que el proceso tuviera un bucle. Sin estas estructuras, la programación sería difícil y eso haría que los códigos tal vez se alarguen a procesos demasiados largos. Siendo así, que tienen sentido solo cuando tengan una condición a la cual considerar verdadera o falsa para poder continuar con el resto de las operaciones.

Luis Fernando Walls Chávez

Con cada práctica cursada podemos observar la retroalimentación entre estas. En esta ocasión observamos de nuevo las estructuras cíclicas, pero con la sintaxis y estándares del lenguaje C. Sólo existen 3 tipos en este lenguaje: **while, do-while y for.** Y aunado a esto utilizamos las funciones **break y continue** que, respectivamente, nos permite abandonar un ciclo si establecemos una condición, y nos permite regresar al inicio de una iteración si se cumple una condición; estas últimas dos herramientas funcionan muy bien, aparte de comprobar la evaluación de un ciclo, para depurar un programa y poder observar con detenimiento las instrucciones de cada ciclo.