	**UNIVERSIDAD DON BOSCO** **FACULTAD DE INGENIERIA** **ESCUELA DE COMPUTACIÓN**
Ciclo I	**Programación de Algoritmos** **Guía de Laboratorio No. 7** **Estructuras repetitivas**

I. RESULTADOS DE APRENDIZAJE

Que el estudiante:

- Desarrolle soluciones informáticas a problemas que requieran estructuras de repetición.
- Implemente los parámetros de control de cada estructura repetitiva disponible en C++
- Aplique los conceptos de variable contador y acumulador
- Diseñe estructuras repetitivas anidadas

II. INTRODUCCIÓN

Métodos para repetir instrucciones en Lenguaje C

Muchas veces dentro de algún programa, es necesario repetir varias veces un procedimiento representado por un conjunto de instrucciones bien definidas.

Para este tipo de situaciones los lenguajes de programación brindan un conjunto de instrucciones que se conocen con sentencias repetitivas, llamadas también **ciclos**, **lazos** o **bucles**.

Tipos de estructuras repetitivas

El lenguaje C/C++ proporciona tres tipos de sentencias repetitivas que son conocidas como ciclos: `for`, `while` y `do-while`.

Las características y la sintaxis de la primera estructura (for) se estudiara a continuación.

Instrucción for

Un lazo for es una estructura de control que permite repetir un grupo de sentencias un número especificado de veces. Las sintaxis de un lazo for se muestran a continuación:

Método 1: Para una sola instrucción a repetir	**Método 2**: Para repetir un bloque de instrucciones

`for(expresion1; expresion2; expresion3)`
`  instrucciónUnica;`
	`for(expresion1; expresion2; expresion3){`
	`//instrucción1 ;`
	`//instrucción2 ;`
	`.`
	`.`
	`//instruccionN;`
	
	`}`

Las expresiones utilizadas en un f o r , se definen a continuación:

- **e x p r e s i ó n 1** , es conocida como *inicialización*. Es una asignación de una o más variables, en la que se establece el valor inicial de la variable de control (contador) del ciclo.
- **e x p r e s i ó n 2** , es una *condición* de algún tipo que, mientras sea evaluada como t r u e , permite la ejecución de las instrucciones del ciclo.
- **e x p r e s i ó n 3** , conocida como *incremento*. Es otra asignación, en donde se realiza el incremento o decremento de la variable de control o contador del lazo.

Y todas estas expresiones contenidas en el paréntesis del for deben estar separadas por (**;**) PUNTO Y COMA.

Si va a utilizar más de una variable de control (dos contadores), debe separar las instrucciones de inicialización y/o de incremento con comas.

Sin embargo, la condición siempre será solamente expresión booleana (que puede incluir a todas las variables de control utilizadas.

No es imprescindible que existan todas las expresiones dentro del paréntesis del for.

Pueden aparecer vacías algunas de ellas; o incluso, todas.

Sintaxis de un ciclo for

```
for(i=0,j<10;exp2;i++,j--)

for ( ; exp2 ; exp3 )
 ó también

for ( exp1 ; ; )
 ó también

for ( ; ; )
```

El programador en C deberá haber previsto alguna manera alternativa de salir del lazo, ya que si no, la ejecución del mismo se volverá infinita (o tan larga como se mantenga encendida la computadora).

A este tipo de lazo o ciclo erróneo se le denomina **lazo o ciclo infinito**.

Instrucción while

La sentencia repetitiva while es el segundo de tipo de iteraciones posibles en C.

Su sintaxis podría expresarse de las siguientes formas:

Método 1: Para una sola instrucción a repetir	**Método 2**: Para repetir un bloque de instrucciones

`while(expresion)`
`  instruccionUnica;`
	`while(expresión){`
	`//bloque de instrucciones a repetir`
	`instruccion1;`
	`Instruccion2;`
	`.`
	`.`
	`.`
	`instruccionN;`
	`}`

La sintaxis del lazo **while** expresada en palabras significa lo siguiente: "**mientras expresión de un resultado true, se ejecutará el bloque de instrucciones hasta la última instrucción N del bloque**".

La expresión es **evaluada antes de iniciar el primer ciclo**, por lo que, en caso de ser falsa esta condición, nunca entrara al ciclo y C continuara con el resto del programa, ignorando todo el cuerpo del lazo while.

Por lo general, dentro del bloque de instrucciones, **se modifican variables de la expresión condicional del while, para controlar así la duración de la iteración**.

Instrucción do – while

Método 1: Para una sola instrucción a repetir	**Método 2**: Para repetir un bloque de instrucciones

`do`
` instrucción Única a repetir;`
`while(condition);`
	`do {`
	`//bloque de instrucciones a repetir`
	`instruccion1;`
	`Instruccion2;`
	`.`
	`.`
	`instruccionN;`
	`} while(expresión);`

Su bloque de instrucciones **siempre se ejecuta la primera vez**, independientemente del resultado de la condición.

Y luego, dependiendo del resultado de la expresión condicional colocada al final dentro del while, si está es `true`, se repetirá la ejecución del bloque de instrucciones, de lo contrario se terminará el ciclo o lazo.

Alterando la ejecución de un ciclo

Lenguaje C++ presenta 2 instrucciones para modificar la ejecución de un ciclo/lazo, para así alterar la ejecución "normal" en cada ciclo.

Estas dos instrucciones son: **break** y **continue**.

La instrucción break (Salir del bucle)

Produce una salida inmediata de cualquier estructura de control antes de que llegue a su finalización, ya sea selectiva múltiple (como el `switch`), o para producir la salida anticipada de un ciclo (`while`, `do…while` o en un `for`).

La ejecución del programa continúa con la primera instrucción fuera de la estructura de control suspendida.

```
#include <iostream>
using namespace std;
main() {
int n=6 ,i=1;
while(i<=10){
cout<<n<<"x"<<i<<"="<<n*i
 <<endl;
  if(i==5) break;
  i++;
}
}
```

```
6x1=6
6x2=12
6x3=18
6x4=24
6x5=30

--------------------------------
Process exited after 0.01173 seconds wi
Presione una tecla para continuar . . .
```

Cuando la variable i toma el valor de 5, la condición del while se cumple e ingresa al ciclo para ejecutarlo.

Evaluá en su estructura if si variable i es igual a 5.

Como esta condición es verdadera, ejecuta sentencia break, saliendo del ciclo

La variable `i` termina con el dato 5 pues jamás se ejecuta la multiplicacion de n*i

La instrucción continue (Saltar una repetición)

Ya no continúa con el resto de las instrucciones del ciclo o lazo en ejecución, pero a diferencia de la instrucción `break`, C++ continúa con la siguiente iteración de ese ciclo.

```
#include <iostream>
using namespace std;
main() {
int n=6 ,i=1;
while(i<=10){
 i++;
 if(i==5) continue;
 cout<<n<<"x"<<i<<"="<<n*i
   <<endl;
}
}
```

El código anterior imprime en pantalla:

```
6x2=12
6x3=18
6x4=24
6x6=36
6x7=42
6x8=48
6x9=54
6x10=60
6x11=66

--------------------
```

Como se puede observar en la salida de datos, la multiplicación cuando i es 5 es el único que no se imprime.

La condición de continuidad del ciclo se evalúa nuevamente después de ejecutar la instrucción `continue`, y si el resultado es `true` el ciclo continúa, de lo contrario (si es FALSO) se termina.

Puede utilizar la instrucción **`continue`** con los ciclos `while`, `do…while` o `for`.

III. MATERIALES Y EQUIPO

No.	Requerimiento	Cantidad
1	Memoria USB	1
2	Computadora con compilador de C++	1

IV. PROCEDIMIENTO

1. Cree una carpeta llamada **PAL_Guia7**, en el cual guardará sus archivos de ejemplos y del análisis de resultados.

2. Prepare la siguiente pareja de códigos fuentes, en un archivo de código fuente diferente para cada uno.

Nombres de programas PAL_Guia7_E1A, PAL_Guia7_E1B , respectivamente

Desarrolle un programa que despliegue en pantalla la tabla de multiplicar de un número entero dado por usuario.

PAL_Guia7_E1A.cpp

```
//Programa para ver tabla de multiplicar
#include<iostream>
using namespace std;
#include<conio.h>

main(){
/*num para generar tabla multiplicacion (TM)*/
 int N;
 //contador para generar la TM
 int i;
 cout<<"\n\tVER TABLA DE MULTIPLICAR\n\n";
 cout<<"Ingrese numero para ver su tabla de multiplicar: ";
 cin>>N;

 cout<<"\n\nTabla de multiplicar del "<<N<<endl;

 for(i=1;i<=10;i=i+1){
  cout<<"\n "<<N<<" por "<<i<<" es "<<N*i;
 } //fin for i

 getch();
}
```

PAL_Guia7_E1B.cpp

```
//Programa para ver tabla de multiplicar
#include<iostream>
using namespace std;
#include<conio.h>

main(){
 //num. para generar tabla multiplicacion (TM)
 int N;
 //contador para generar la TM
 int i;
 cout<<"\n\tVER TABLA DE MULTIPLICAR\n\n";
 cout<<"Ingrese numero para ver su tabla de multiplicar: ";
 cin>>N;

 cout<<"\n\nTabla de multiplicar del "<<N<<endl;
 i=1;
 while(i<=10){
  cout<<"\n "<<N<<" por "<<i<<" es "<<N*i;
  i=i+1;
 } //fin while i
 getch();
}
```

3. Realice la compilación de cada uno de los códigos anteriores.

Haga diferentes pruebas de ejecución. Para un mismo dato de entrada, debera obtener los mismos resultados en ambas aplicaciones.

4. Cree un nuevo archivo de código fuente de C++ y proceda a digitar el código del segundo ejemplo a continuación:

Nombre de programa PAL_Guia7_E2:

Desarrolle una aplicación que le permita obtener el factorial de un número entero positivo ingresado por el usuario.

En caso que el usuario se equivoque en el valor solicitado, pida el valor nuevamente hasta un máximo de 3 veces, sino, finalizara el programa.

Debe utilizar solamente estructuras `while` y también a `do-while` para elaborar la solución.

```
#include<iostream>
using namespace std;
#include<conio.h>

main() {
   int i; //contador
   int numero, factorial = 1;

   cout<<"\t\tPrograma para calculo del factorial de un numero\n\n";

/* Solicita un numero positivo al usuario.
Si se equivoca, lo pedira de nuevo solo 2 veces mas, sino finaliza
el programa */
   cout<<"Digita un numero entero mayor que cero: ";
   i=0; //cuenta las veces que usuario se equivoca al ingresar un numero

   do{
      cin>>numero;
      if(numero<=0){
        i++; //incrementa conteo-error de ingreso de numero solicitado

        if(i==3)
          return(0); //finaliza ejecucion de main
        else
          cout<<"\nTe equivocaste de valor, intentalo de nuevo.."<<
```

```
            "\ndigita a un numero positivo: ";
      }//fin if
   } while(numero <= 0);
   //En este ciclo se obtiene el factorial de un número
   i=numero; //inicia conteo con valor positivo dado por usuario
   while(i > 0){ //Genera n enteros, entre numero hasta 1
       factorial = factorial * i; //calcula factorial solicitado
       i=i-1; //decrementa conteo
   }//fin while i

   //Muestra el resultado obtenido
   cout<<"\n\n Factorial de "<<numero<<" es "<<factorial;

   getch();
}//fin de main
```

3. Realice la compilación del código anterior y ejecútelo.

Compruebe equivocándose a propositivo más de 2 veces en el valor positivo solicitado.

Luego, compruebe que el factorial de 5 es 120.

Observe que la variable **i** se usa como contador en 2 momentos diferentes del programa.

Nombre del programa PAL_Guia7_E3:

Permita a un docente ingresar un total de notas que desea que la PC procese.

Luego podrá ingresarlas una a una.

Una nota es válida solamente en el rango (0.0-10.0), de lo contrario, debe llamar la atención al docente e indicarle que debe ingresar nuevamente la nota.

Al finalizar el ingreso correcto de todas las notas, el programa le muestra al usuario estos resultados:

a) Cantidad de notas aprobadas
b) Cantidad de notas reprobados
c) nota promedio de las notas reprobadas

```
#include <iostream>
using namespace std;

#include <iomanip> //manipuladores de formato para cout
#include<windows.h> //acceso a posicion de Cursor
#include<conio.h> //funcion getch
```

```
main() {

 //para acceso al cursor de la ventana de aplicacion
 HANDLE cur= GetStdHandle(STD_OUTPUT_HANDLE);
 COORD p; //coordenadas x,y de posicion del cursor

 //separacion horizontal y vertical entre resultados de tabla
 const int ancho=19, altura=1;
 int fil=0,col=0; //num. fila y columna para ubicar notas validas

 int nalum; //total de notas a ingresar
 int cont=0; //contador de notas validas recibidas
 float nota; //nota individual
 float reprobados=0,aprobados=0,acumreprob=0,acumaprob=0;

 bool salir=false; //bandera usada por ciclo do-while
 cout<<"*  Programa Academico de notas aprobados y reprobados  *\n\n";
 cout<<"\nIngrese cantidad de alumnos a procesar: "; cin >>nalum;
 cout<<"\nIngrese la nota del alumno # "<<cont+1<<" :";
 while(cont<nalum){
  p.X= 27; p.Y=5;
  SetConsoleCursorPosition(cur,p);
  cout<<"# "<<cont+1<<": ";
  //inicia ciclo infinito para capturar solo a una nota valida
  salir=false;
  do{
   p.X= 32; p.Y=5;
   SetConsoleCursorPosition(cur,p);

   cin>>nota; //captura nota ingresada por usuario

   if(nota<0 || nota>10) {//determina si nota es incorrecta
    p.X= 32;   p.Y=6;
    SetConsoleCursorPosition(cur,p);
    cout<<" >> ERROR, nota debe estar entre 0 a 10.0 ";
    getch();
    //hace efecto de borrado de mensaje de error
    p.X= 32; p.Y=6;
    SetConsoleCursorPosition(cur,p);
    cout.width(50); cout.fill(' ');
    cout<<" ";
   }
   else salir=true;//permite finalizar ciclo do-while
   //borra nota ingresada y asi preparar espacio para prox. nota
   p.X= 32; p.Y=5; SetConsoleCursorPosition(cur,p);
   cout.width(50); cout.fill(' ');
   cout<<" ";
  }while(!salir); //crea un ciclo infinito
```

```
 cont++; //cuenta una nota valida
 //clasifica nota y actualiza resultados
 if(nota < 6){
  reprobados = reprobados+1;
  acumreprob += nota;
 }else{
  aprobados = aprobados + 1;
  acumaprob += nota;
 }
 //imprime la nota en pantalla
 p.X= 2+ancho*col; p.Y=8+altura*fil;
 SetConsoleCursorPosition(cur,p);
 cout<<"Nota # "<<cont<<" : "<<nota;

 //determina coordenadas de prox. impresion de nota
 fil++;
 if(fil>9){
  col++; //cambia a prox. columna
  fil=0; //comienza de nuevo a filas
 }
}//fin while-cont
cout.precision(2); p.X= 60; p.Y=9; SetConsoleCursorPosition(cur,p);
cout<<"\tResultados finales:";

p.X= 60; p.Y=10; SetConsoleCursorPosition(cur,p);
cout<<setw(35)<<setfill('_')<<".";

cout.setf(ios::left);

p.X= 60; p.Y=12; SetConsoleCursorPosition(cur,p);
cout<<setw(32)<<"Numero de alumnos aprobados :"<<aprobados;

p.X= 60; p.Y=14; SetConsoleCursorPosition(cur,p);
cout<<setw(32)<<"Numero de alumnos reprobados:"<<reprobados;

p.X= 60; p.Y=16; SetConsoleCursorPosition(cur,p);
if(reprobados>0){
 acumreprob /= reprobados;
 cout<<setw(32)<<"Nota Promedio de los reprobados:"<<acumreprob;
}else cout<<"No hay estudiantes reprobados";
getch();
}//fin main
```

4. Compile el programa anterior.

Haga una prueba de ingreso de 5 notas, equivocándose a propósito en el ingreso de varias de ellas. Ejecute otra prueba, esta vez con 24 notas.

5. Ahora responda estas preguntas:

- *¿Cuál es la mayor cantidad de notas que permite ingresar esta aplicacion?*
- *¿Cuáles variables de este código fuente se usan como "acumuladores"*
- *¿Cuales variables como "contadores"? Justifique su respuesta…*

V. ANÁLISIS DE RESULTADOS

Elabore una aplicación de solución en C++ para cada uno de los siguientes problemas.
Si el problema indica restricciones, deben ser respetadas e implementadas en la solución.

PROBLEMA 1

La conjetura de Collatz dice que, a partir de cualquier número inicial, la siguiente sucesión obtenida siempre termina en 1:

- Si el numero de la sucesión es par, el siguiente numero se obtiene por dividirlo entre 2
- Si es un numero impar, el siguiente valor se obtiene al multiplicarlo por 3 y sumarle 1.
- Si el numero es 1, la sucesion finaliza

Por ej: si se inicia con 6, obtendremos la sucesión de números 6, 3, 10, 5, 16, 8, 4, 2, 1.

Escriba un programa que permita verificar la conjetura de Collatz para cualquier entero dado, y que imprima la secuencia correspondiente.

PROBLEMA 2

Determine si un número entero positivo es "Perfecto". Un número perfecto es aquel que es igual a la suma de todos sus divisores exactos (excepto el mismo).

Cuando el programa determine que un número no es perfecto, deberá indicarlo al usuario con la justificación apropiada. Ejemplos:

28 es perfecto, porque es igual a la suma de sus divisores, excepto el mismo, es 28 (1+2+4+7+14)

Otro número perfecto es 496.

En cambio 36, no es perfecto, porque la suma de sus divisores es 55 (1+2+3+4+6+9+12+18)

- En la solución, solamente puede usar ciclos `do-while`.

PROBLEMA 3

Solicite al usuario la base y la exponente de una potencia cuyo resultado desea saber.

- La base puede ser un número con punto flotante y el exponente puede ser un número entero positivo o negativo, e incluso cero.
- El programa no debe utilizar la librería de funciones matemáticas (`math.h`)
- Además, si se requiere usar estructuras repetitivas, solo podrá usar únicamente a `while` o sino ciclos `do-while`.

PROBLEMA 4

El valor de épsilon `e=2.718281828...` (base de los logaritmos naturales) se determina gracias a la sumatoria de N términos de la siguiente formula:

$$e = \sum_{x=0}^{N} \frac{1}{x!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} \ldots + \frac{1}{N!}$$

Solicite a usuario el total de términos N con el cual desea que se genere y vea en pantalla al valor aproximado de esta constante.

Si requiere varios ciclos en la solución, cada tipo de ciclo utilizado no debe repetirse en la solución.

Sugerencias:

- Utilice las banderas `fixed` y `precisión` del operador `cout`, para determinar resultado con hasta 12 cifras decimales.
- Recuerde que el factorial de un numero `n` se calcula así: $n! = n * (n-1)!$
- Y el factorial de cero es 1, es decir: **`0! = 1`**

PROBLEMA 5

La función seno de un angulo X (dado en radianes) se puede calcular gracias a la sumatoria de términos de la serie siguiente:

$$sen(X) = X - \frac{X^3}{3!} + \frac{X^5}{5!} - \frac{X^7}{7!} + \frac{X^9}{9!} - \ \ldots.$$

Solicite a usuario el valor del angulo X (en grados sexagesimales) y el total N de términos para calcular el valor aproximado de la función Seno(X).

- No olvidar convertir el angulo X recibido a radianes.

  Por ejemplo, si usuario brinda un angulo X de 30 grados sexagesimales y desea 3 terminos de la serie, la función seno (30) dariaa un aproximado de 0.5000021326. El valor exacto es de 0.5
- Si la solución requiere usar más de un ciclo, todos deberán ser de tipos diferentes.
- Fije la precisión a 10 cifras decimales, con las sentencias:

```
cout.setf(ios::fixed); cout.precision(10);
```

- No debe utilizar a las funciones de trigonometría disponibles en la librería math.h

PROBLEMA 6

Solicite 2 números enteros positivos al usuario, para luego, indicarle si ambos números ingresados son "amigos", además, debe justificar su afirmación de manera exacta.

Dos números son "amigos", cuando la suma de todos los divisores del primer número (excepto el mismo número) es igual al segundo número y viceversa.

Por ejemplo:

Los números 220 y 284 son amigos, porque:

La suma de los divisores de 220 (1+2+4+5+10+11+20+22+44+55+110) es 284 y luego, la suma de los divisores de 284 (1+2+4+71+142) es 220.

Cada tipo de ciclo requerido en la solucion lo puede usar solo una vez en la solución.

PROBLEMA 7

Haga un programa que genere un menú para calcular el area de diferentes cuerpos geométricos (cuyas medidas son brindadas en milimetros).

- El usuario puede elegir calcular el area de: 1. Esfera, 2. Cubo, 3. Cilindro, o también 4. Salir.
- Si el usuario selecciona las opciones del 1 al 3, el programa le pide los datos necesarios, hace el cálculo del área del cuerpo geométrico seleccionado y le muestra la respuesta. Después, el programa tendrá que mostrar nuevamente el menú al usuario para que pueda elegir otra opción.
- El programa continúa indefinidamente mientras usuario no seleccione 4, Salir.

Guía de Laboratorio No. 7
RÚBRICA DE EVALUACIÓN

Actividad a evaluar: ANÁLISIS DE RESULTADOS

Formar grupos entre 3 a 5 estudiantes, llenar esta hoja de evaluación y entregarla a su docente.

Luego, su instructor seleccionará 3 problemas del análisis de resultados, para ser resueltos apropiadamente por el grupo.

Lista de Integrantes:

CARNET 1	CARNET 2	CARNET 3	CARNET 4	CARNET 5

Problemas a resolver:

Criterio a evaluar	¿Prob 1?	¿Prob 2?	¿Prob 3?	PROMEDIO
(15%) Define las variables de entrada y salida esperadas				
(30%) Implementa cada estructura de bucle esperadas en el problema. Se cumplen las restricciones dadas en la redacción				
(25%) Código fuente compila y se obtiene a c/u de los resultados solicitados				
(15%) Diálogo con usuario es el apropiado				
(15%) Documenta apropiadamente a cada ejercicio solicitado				
			Nota:	