Unach
UNIVERSIDAD NACIONAL DE CHIMBORAZO

FACULTAD DE INGENIERÍA

ESTRUCTURAS DE CONTROL JAVA/C

Mgs. Diego Reina Haro

COMPUTACIÓN MÓVIL

ESTRUCTURAS DE CONTROL

COMPUTACIÓN MÓVIL fppt.com

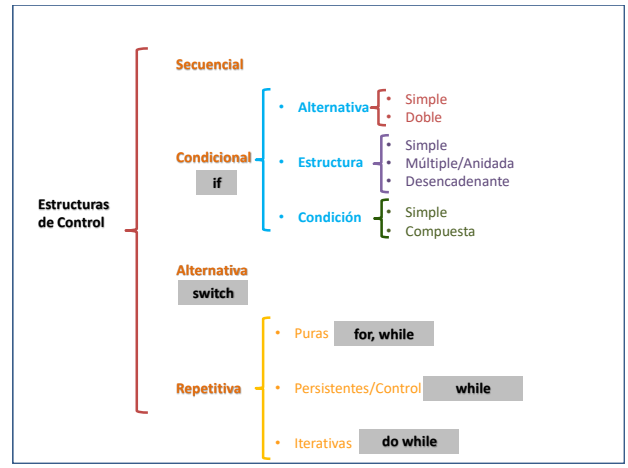
ESTRUCTURAS DE CONTROL

Las estructuras de control determinan la secuencia en la que se ejecutarán las instrucciones de un programa.

Las estructuras de control se dividen en tres categorías en función del flujo de ejecución:

- ***Estructura secuencial.***
- ***Estructuras condicional.***
- ***Estructura alternativas.***
- ***Estructuras repetitiva.***

COMPUTACIÓN MÓVIL fppt.com



ESTRUCTURA SECUENCIAL

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURA SECUENCIAL

La estructura secuencial está formada por una secuencia de instrucciones que se ejecutan en orden una a continuación de la otra.

Cada una de las instrucciones están separadas por el carácter punto y coma (;). No obstante, en algunos casos nos interesará agrupar en un bloque una serie de instrucciones, como veremos al explicar las estructuras de selección y de iteración.

El bloque de sentencias se define por el carácter llave de apertura ({) para marcar el inicio del mismo, y el carácter llave de cierre (}) para marcar el final.

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURAS DE CONTROL

ESTRUCTURA SECUENCIAL

Ejemplo:

```
{
instrucción 1;
instrucción 2;
instrucción 3;
.....
instrucción N;
}
```

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURAS DE CONTROL

ESTRUCTURA SECUENCIAL

EJERCICIOS

Diseñar un programa en lenguaje C, que permita sumar 2 números cualesquiera.

Diseñar un programa en lenguaje C, que permita en contratar el area de un cuadrado.

Diseñar un programa en lenguaje C, que permita encontrar el area de un triangulo

Diseñar un programa en lenguaje C, que permita encontrar el area de un circulo.

COMPUTACIÓN MÓVIL fppt.com

Sol: Diseñar un programa en lenguaje C/JAVA, que permita sumar 2 números cualesquiera.

```
# include <iostream.h>
# include <stdio.h>
# include <conio.h>

int main ()
{
    double a, b ,c ;
    cin>>a;
    cin>>b;
    c= a+b;
    cout<<c;
    getch();
    return 0;
}
```

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURAS DE CONTROL

ESTRUCTURA CONDICIONAL

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURAS DE CONTROL

ESTRUCTURA CONDICIONAL

Las estructuras condicionales controlan si una sentencia o bloque de sentencias se ejecutan, en función del cumplimiento o no de una condición o expresión lógica.

En el lenguaje de programación , la estructura de control

- **if** (condiconal)

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURAS DE CONTROL

ESTRUCTURA CONDICIONAL "IF"

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción IF

Esta instrucción hace que, se ejecuten unas sentencias u otras dependiendo del valor que toma una condición. La instrucción **IF** se puede clasificar de 3 formas:

Alternativa, Estructura y Condición.

Por su **ALTERNATIVA** se dividen en: simple o doble

Alternativa simple:

Cuando existe una única solución ya sea por verdadero o falso.

```
if (condicion)
{
  instrucción 1;
  instrucción 2;
  instrucción 3;
}
```

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción IF

Alternativa doble:

Cuando existe una solución por verdadero y otra solución por falso

```
if (condicion)
{
   Instrucción 1;
   instrucción 2;
}
else
{
   instrucción 3;
   instrucción 4;
}
```

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción IF

También existe una clasificación según la **ESTRUCTURA** es decir su conformación o armando.

- ***Estructura Simple***
- ***Estructura Múltiple***
- ***Estructura Desencadenante.***

Estructura Simple.- Se caracteriza porque su estructura maneja tan solo 2 instrucciones posibles ante la condición dada, así:

```
if (condicion1)
    instrucción1;
else
   instrucción2;
```

Cuando existe una solución por verdadero y otra solución por falso

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción IF

Estructura Múltiple.- Se caracteriza porque su estructura es capaz de resolver varias condiciones, para ellos es necesario **anidar** tantas instrucciones **IF** como sea necesario, así:

```
if (condicion1)
  instrucción1;
else if(condicion2)
     instrucción2;
   else if(condicion3)
        instrucción3;
      else if(condicion4)
           instruccion4;
         else
           instrucción5;
```

Cuando existen varias soluciones, que dependen de varias condiciones.

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción IF

Estructura Desencadenante.- Se caracteriza porque su estructura es capaz de resolver varias instrucciones, pero las instrucciones **IF** se colocan una después de la otra. (no anidadas).

```
if (condicion1)
  instrucción1;
else
  instrucción1;
-----------------------------
if (condicion2)
    instrucción2;
else
instrucción2;
-----------------------------
if (condicion3)
    instrucción3;
else
instrucción3;
```

Cuando existen varias soluciones, que dependen de varias condiciones; esto obliga a resolver cada condición por separado una debajo de la otra. (es más didáctico para un programador)

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción IF

La parte principal de una instrucción **IF** es la condición, existe 2 tipos de condicione con las que se puede trabajar.

- ***Condición SIMPLE***
- ***Condición COMPUESTA***

Toda condición tiene 3 parámetros: **VARIABLE**, **OPERADOR** y **VALOR**

```
if ( suma == 100 )
```

```
<=
>=
!=
```

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción IF

IF con Condición SIMPLE

Una instrucción **IF** con condición simples, evalúa una posibilidad para determinar un resultado ya sea por verdadero o por falso.

```
if (sueldo >= 500 )
{
……
}
else
{
…….
}
```

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción IF

IF con Condición COMPUESTA

Una instrucción **IF** con condición compuesta, evalúa dos o mas posibilidades para determinar un resultado. Las condiciones compuestas se forman con los operadores lógicos **AND** y **OR**.

```
if ( (sueldo >= 500) and (edad>= 35 ))
{
   ……
}
else
{
   …….
}
```

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción IF

EJERCICIOS

Diseñar un programa que lea la **edad de una persona** y muestre como resultado si es **MAYOR DE EDAD** o **MENOR DE EDAD**.

Diseñar un programa que lea la temperatura de una persona y muestra si esta **EMFERMA O SANA**

Diseñar un programa que lea un **número entero** y muestra si es **PAR** o **IMPAR**.

Diseñar un programa que me permita validar números solo de 3 cifras, si bien ingresado mostrar **CORRECTO,** si esta mal ingresado que muestre un **INCORRECTO**.

Diseñar un programa que lea un **números enteros (1-7)** y muestre como resultado el día de la semana que le corresponda.

Diseñar un programa que ingrese una letra y muestre como resultado si es **vocal** o **consonante**.

fppt.com

Sol: Diseñar un programa que lea la **edad de una persona** y muestre como resultado si es **MAYOR DE EDAD** o **MENOR DE EDAD**.

```
# include <iostream.h>
# include <conio.h>
# include <stdlib.h>

int main ()
{
  int edad;
  cout<<"Ingrese su Edad: ";
  cin>>edad;

  if (edad >= 18)
  {
       cout<<"ES MAYOR DE EDAD.!!!";
  }
  else
  {
    cout<<"ES MENOR DE EDAD.!!!";
  }
  getch();
  return 0;
}
```

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción IF

LECCION

1 Diseñar un programa que lee un **número entero** que corresponde a una hora del día **(0-23)** y muestra un mensaje según la hora que se haya leído. Considere:

Entre 0 y 11 -> Buenos Días.
Entre 12 y 18 -> Buenas Tardes.
Entre 19 y 23 -> Buenas Noches.

2 Diseñar un programa que lee la calificación obtenida por un alumno **(0-10)** en un examen y muestra la nota equivalente en texto.

10 -> Sobresaliente
9 ->Muy Buena
8-> Buena
7-> Deficiente
6-> Muy deficiente
Para el resto = Pésimo

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción IF

LECCION

1 Diseñar un programa que lee un **número entero** que corresponde al signo del zodiaco **(1-12)** y muestra un mensaje con el nombre del signo y la fecha:

1 = Es Acuario y su fecha es 21/01 a 19/02….
2 = Piscis y su fecha es 20/02 a 20/03…….. (VER ANEXO AL FINAL)

2 Diseñar un programa que determine cuantas cifras tiene un numero ingresado: ejemplo

1 -> el numero tiene 1 cifra
68 -> el numero tiene 2 cifras
115 -> el numero tiene 3 cifras
.
.
Hasta un máximo de 7 cifras

COMPUTACIÓN MÓVIL fppt.com

ESTRUCTURAS DE CONTROL

ESTRUCTURA ALTERNATIVA "SWITCH"

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción SWITCH

La sentencia **switch** selecciona una de entre múltiples alternativas.

La forma general de esta expresión es la siguiente:

```
switch (expresión)
{
      case constante1:
               instrucciones;
               break;
      case constante 2:
               instrucciones;
               break;
      . . .
      default:
               instrucciones;
}
```

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción SWITCH

En una instrucción **switch**, ***expresión*** debe ser una expresión con un valor entero, y ***(constante1, constante2, …,)*** deben ser constantes enteras, constantes de tipo carácter o una expresión constante de valor entero. ***Expresión*** también puede ser de tipo char, ya que los caracteres individuales tienen valores enteros.

```
switch (expresión)
{
      case 1:
               instrucciones;
               break;
      case 'A':
               instrucciones;
               break;
      . . .
      default:
               instrucciones;
}
```

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción SWITCH

La instrucción **switch** evalúa la ***expresión*** entre paréntesis y compara su valor con las constantes de cada **CASE**. Se ejecutarán las instrucciones de aquel **CASE** cuya constante coincida con el valor de la expresión, y continúa hasta el final del bloque o hasta una instrucción que transfiera el control fuera del bloque del **switch** *(una instrucción break, o return)*.

Si no existe una constante igual al valor de la expresión, entonces se ejecutan las sentencias que están a continuación de **DEFAULT** si existe *(no es obligatorio que exista, y no tiene porqué ponerse siempre al final)*.

```
switch (expresión)
{
      case 1:
               instrucciones;
               break;
      case 'A':
               instrucciones;
               break;
      . . .
      default:
               instrucciones;
}
```

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción SWITCH

PRACTICA

Diseñar un programa que lea como entrada un **numero entero** comprendido entre **1 – 7** y muestre el **día de la semana** correspondiente.

```
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>

int main()
{
   int num;

   cout << "Ingrese un número comprendido entre [1- 7]: ";
   cin >> num;
```

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción SWITCH

PRACTICA

Diseñar un programa que lea como entrada un **numero entero** comprendido entre **1 – 7** y muestre el **día de la semana** correspondiente.

```
switch (num)
 {
      case 1 : {
            cout<<"corresponde a: LUNES";
            break;
            }
      case 2 : {
            cout<<"corresponde a: MARTES";
            break;
            }
      case 3 : {
            cout<<"corresponde a: MIERCOLES";
            break;
            }
      case 4 : {
            cout<<"corresponde a: JUEVES";
            break;
            }
      case 5 : {
            cout<<"corresponde a: VIERNES";
            break;
            }
      case 6 : {
            cout<<"corresponde a: SABADO";
            break;
            }
      case 7 : {
            cout<<"corresponde a: DOMINGO";
            break;
            }
      default : {
             cout << "DATO MAL INGRESADO"<< endl;
            }
 }
 getch();
 return 0;
}
```

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción SWITCH

EJERCICIOS

Programa que al ingresar un **numero entero** entre **(1-12),** muestra el nombre correspondiente al **mes**.

Programa que lee **dos números** y **una operación (- , +, *, /)** y realiza la operación entre esos números según la operación escogida.

Programa que determina si **un carácter** digitado **es una vocal o una consonante**. (mayúsculas o minúsculas)

Menú de Operaciones (**Suma, Resta, Multiplicación y división**)

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción SWITCH

LECCION

Programa que pida *su altura en metros* , y muestre una lista de opciones para cambiarlo de unidad a: centímetros, pulgadas, yardas, kilómetros y muestre el resultado convertido en dicha unidad escogida.

Digite su altura(mt.): ???

1. **Centimetros**
2. **Pulgadas.**
3. **Yardas.**
4. **Kilometros**

Escoja la conversión: ??

Su altura en ___ es de: ???

ESTRUCTURAS DE CONTROL
ESTRUCTURA CONDICIONAL O ALTERNATIVA

Instrucción SWITCH

LECCION

1

Menú de Operaciones
1. **Calculo de la Potencia**
2. **Determinar si un numero es par o impar**
3. **Salir**

¿Escoja su opción?

2

Menú de Operaciones
1. **Invertir un numero de 2 cifras**
2. **Determinar el mayor de 2 números ingresados**
3. **Salir**

¿Escoja su opción?

ESTRUCTURAS DE CONTROL

ESTRUCTURAS REPETITIVAS

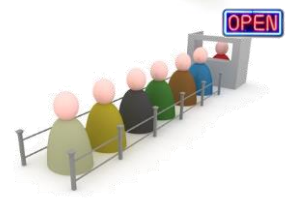
ESTRUCTURAS DE CONTROL

ESTRUCTURAS REPETITIVAS

Las estructuras repetitivas, permiten diseñar programas que pueden contener ciclos ***repetitivos automáticos*** o ***ciclos repetiditos que pueden ser detenidos por el usuario*** (iterativo).

Existen 3 tipos de estructuras repetitivas o iteractivas:

- ***while*** (mientras……)
- ***do-while*** (hacer …….mientras; repetir……hasta)
- ***for*** (desde/para……)



ESTRUCTURAS REPETITIVAS " WHILE "

ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERATIVAS

Instrucción WHILE

Ejecuta una instrucción o un bloque de instrucciones cero o más veces, dependiendo del valor de la condición.
Se evalúa la ***condición***, y si es ***cierta***, se ejecuta la ***instrucción*** o bloque de instrucciones y se vuelve a evaluar la ***condición***; pero si la condición es ***falsa***, se pasa a ejecutar la siguiente instrucción después del ***while.***

```
while (condicion)
{
instrucción 1;
.............
instrucción N;
}
```


ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERATIVAS

Instrucción WHILE

BANDERA "cambia de estado"
(VARIABLE, OPERADOR, VALOR)
(CONDICIÓN SIMPLE O COMPUESTA)

```
while (condicion)
{
instrucción 1;
.............
instrucción N;
}
```

Generalmente en las estructuras repetitivas o iteractivas, suele usarse un componente llamado ***acumulador.***

Es una ***variable*** inicializada en ***0*** que permite acumular valores; muy útil para contar, incrementar u otra operación matemática en la cual sea necesario hacer un control de repetición.

ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERATIVAS

Instrucción WHILE

Acumulador.- Es una ***variable*** inicializada en ***0(+), 1(*)*** que permite acumular valores; muy útil para contar, u otra operación matemática en la cual sea necesario hacer un control de repetición.

Incrementador/Contador.- llamado también contador es una variable que suele ir aumentando de valor con el fin de permitir repetir procesos varias veces. ***cont=cont+1***

Bandera.- también llamada switch es una variable que suele cambiar de estado (cambia de valor), con la finalidad de indicar que ha cambiado la condición. ***band=1; sw=1***

ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERATIVAS

Instrucción WHILE

Existen 3 formas de usar una estructura while:

- Repetitiva *(condiciones: contadores, acumuladores, limites)*
- Persistencia o de Control *(condiciones inversas)*
- Iteractivos *(banderas, switch)*

WHILE
REPETITIVAS PURAS

ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERATIVAS

Instrucción WHILE

EJERCICIOS -REPETITIVOS

Imprimir una serie numérica del 1 al 10

Imprimir una serie numérica del 1 hasta el valor que el usuario decida

Mostrar todos los números pares existente entre [1 - 100].

Generar la Tabla de Multiplicación que solicite el usuario.

Programa que suma los n primeros número enteros. (n es ingresado por el usuario).

Programa que suma los n número enteros, ***ingresados por el usuario***; (n es ingresado por el usuario).

LECCIÓN

ESTRUCTURAS REPETITIVAS WHILE

1. Realizar un algoritmo que determine el factorial de un numero ingresado por el usuario. (*while)

$4! = 1 \times 2 \times 3 \times 4 = 24$

$6! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 = 720$

2. Realizar un algoritmo que determine la formula de la potencia (*while)

$2^3 = 2 \times 2 \times 2 = 8$

$3^5 = 3 \times 3 \times 3 \times 3 \times 3 = 243$

WHILE PERSISTENTES

ESTRUCTURAS REPETITIVAS WHILE

EJERCICIOS - ITERATIVOS

(condición inversa)

Control Persistente para números positivos

Control Persistente para Números de 2 cifras

Control Persistente para Números Pares

Programa que admita ingresar varios números positivos, cuando se inserte un negativo finalice el programa.

ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERATIVAS

Instrucción WHILE

PRACTICA

Programa en C; que lee números enteros hasta que se lee un número negativo. Se muestra la suma de todos los números leídos excepto el número negativo.

```
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>

int main()
{
  int acum, num;
  acum = 0;
  cout << "Introduzca un numero: ";
  cin >> num;
  while (num >= 0)
    {
      acum = acum + num;
      cout << "Introduzca otro numero: ";
      cin >> num;
    }
  cout << endl << "La suma es: " << acum <<
endl;
  getch();
  return 0;
}
```

ESTRUCTURAS REPETITIVAS WHILE

EJERCICIOS - PERSISTENTES O DE CONTROL

(condición inversa)

EJERCICIOS APLICATIVOS

Realizar las suma entre 2 números que tengan 2 cifras (controlar 2 cifras)

Ingresar la edad de una persona "mayor de edad" (controlar mayor de edad) y continuación ingresar los años de trabajo; determinar a los cuanto años comenzó a trabajar

Ingresar la fecha de nacimiento de una persona (controlar dia "1 - 31" , mes "1-12" y año "1940 - 2017", y a continuación imprimir su signo del zodiaco (ver anexo signos) *

WHILE ITERATIVOS

ESTRUCTURAS REPETITIVAS WHILE

EJERCICIOS - ITERATIVOS

(bandera)

Programa que me permita sumar 2 números y repita el proceso hasta que el usuario decida.

Programa que permita comprobar si un numero ingresado es par o impar, y ofrezca al usuario ingresar otro numero.

Programa Menu, que me permita realizar las 4 operaciones básicas, y que al finalizar me permita repetir el menú.

ESTRUCTURAS DE CONTROL — LECCIÓN

1 **Ingresar un número entero de 4 cifras *(controlar que sea de cuatro cifras* persistente*)* y mostrar la suma total de cada una de sus cifras indicando además si el resultado de la suma es par o impar, al finalizar ofrecer al usuario la oportunidad de ingresar otro numero de cuatro cifras S / N, para volver a realizar el calculo.**

Calcular el promedio de 3 notas ingresadas por el usuario *(controlar que las notas sean sobre 20* persistente*)*, Si el promedio es superior a 16 acompañar a la nota la palabra APROBADO caso contrario SUSPENSO, al finalizar ofrecer al usuario la oportunidad de ingresar otras 3 notas S / N, para volver a realizar el calculo. **2**

ESTRUCTURAS REPETITIVAS O ITERATIVAS

" DO....WHILE "

ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERATIVAS

Instrucción DO ...WHILE

Se ejecuta la *instrucción* o bloque de *instrucciones* y a continuación se evalúa la *condición*. Si la condición es ***cierta***, se vuelve a ejecutar la instrucción o bloque de instrucciones, y si es ***falsa***, pasa a ejecutarse la siguiente instrucción después del **do-while**.

Cuando se utiliza una instrucción **"do-while"** el bloque de instrucciones se ejecuta al menos una vez, ya que la condición se evalúa al final.

En cambio, con una instrucción **while**, puede suceder que el bloque de instrucciones no llegue a ejecutarse nunca si la condición inicialmente es falsa.

ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERACTIVA

Instrucción DO ...WHILE

```
do
{
   instrucción 1;
   ..............
   instrucción N;
} while (condicion);
```

BANDERA "cambia de estado"
(VARIABLE, OPERADOR, VALOR)
(CONDICIÓN SIMPLE O COMPUESTA)

Termina el ciclo cuando la condición es **falsa**

ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERACTIVA

Instrucción WHILE

PRACTICA

Programa que lea cualquier número entero diferente de 0.

```
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>

int main()
{
   int numero;
   do
   {
      cout << "Introduzca un numero entero distinto de 0: ";
      cin >> numero;
   }
   while (numero !=0);

   getch();
   return 0;

}
```

ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERACTIVA

Instrucción WHILE

EJERCICIOS

Algoritmo que calcule el factorial de un numero.

Algoritmo que calcule la formula de la potencia: x^y; $3^{2=9}$

Algoritmo que calcule la formula de la potencia: x^y y le permita al usuario volver a ejecutar el programa, hasta que el usuario decida cerrar el programa.

ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERACTIVA

Instrucción WHILE

EJERCICIOS

Programa que ingrese 2 números y a continuación se muestre un menú con las cuatro operaciones básicas. El programa mostrara el resultado de la operación escogida; el programa no debe finalizar hasta que el usuario lo decida ¿Desea Salir del Programa S/N ?

ESTRUCTURAS REPETITIVAS
" FOR "

ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERACTIVA

Instrucción FOR

Un bucle ***FOR*** hace que una instrucción o bloque de instrucciones se repitan un número determinado de veces mientras se cumpla la ***condición***.

```
for(inicialización; limite; incremento/decremento)
{
 instrucción 1;
 ..........
 instrucción N;
}
```

a continuación de la palabra ***FOR*** y entre paréntesis debe haber siempre **tres zonas ó parámetros, separadas por punto y coma**:

- *zona de inicialización*
- *zona de limite (condición)*
- *zona de incremento ó decremento.*

ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERACTIVA

Instrucción FOR

Esta instrucción es especialmente indicada para bucles donde se conozca el ***número de repeticiones*** que se van a hacer.

Como regla práctica podríamos decir que las instrucciones while y do-while se utilizan generalmente cuando no se conoce a priori el número de pasadas, y la instrucción ***for*** se utiliza generalmente cuando sí se conoce el número de pasadas.

ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERACTIVA

Instrucción FOR

EXPLICACIÓN

```
int n;

for (n = 1; n <= 10; n++)
{
   instrucción 1;
   instrucción 2;
   instrucción 3;
   ..................;
   instrucción N;
}
```

Comenzará en 1
Repetir 10 veces
Incrementar de 1 en 1

```
for (n = 10; n >=1; n--) {}
for (n = 1; n <=100; n=n+2) {}
for (n = 100; n >=1; n=n-2) {}
```

ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERACTIVA

Instrucción FOR

PRACTICA

Programa que imprima la serie del 1 al 10.

Programa que imprima la serie del 20 al 1.

Programa que imprima los 25 primeros números, múltiplos del 4.

Programa que permita imprimir una Tabla de Multiplicar, el usuario deberá indicar que tabla desea imprimir y cuantos elementos de la tabla imprimir

ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERACTIVA

Instrucción FOR

Programa que imprima los 25 primeros números, múltiplos del 4.

```
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>

int main()
{
   int n,res;
   for (n = 1; n <=25; n=n+1)
   {
      res= 4 * n;
      cout <<res << endl;
   }
   getch ();
   return 0;
}
```


ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERACTIVA

Instrucción FOR

EJERCICIOS

Programa que imprima la tabla de multiplicar que el usuario requiera.

Programa que imprima el factorial de un número entero ingresado por el usuario.

0!= 1 *(REGLA)*
1!= 1
2!=1x2=2
3!=1x2x3=6
.

ESTRUCTURAS DE CONTROL
ESTRUCTURAS REPETITIVAS O ITERACTIVA

Instrucción FOR

LECCION

Programa que imprima los N primeros números de la serie de FIBONACCI

0,1, 1, 2, 3, 5, 8……

ESTRUCTURAS DE CONTROL

GRACIAS

Mgs. Diego Reina Haro