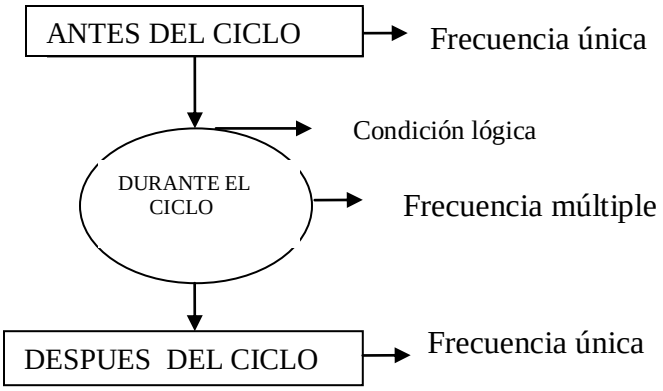
Unidad V
Estructuras Iterativas

Las estructuras iterativas (repetitivas o ciclos) son aquellas que nos permiten repetir varias veces un proceso de acuerdo a una condición lógica.

Estructura iterativa. Esquema:



Entre las estructuras iterativas se encuentran dos tipos de sentencias para controlar la iteración: la sentencia for y la sentencia while.

Sentencia for

Este tipo de sentencia se utiliza si se conoce con exactitud la cantidad de elementos a procesar. Tiene el siguiente formato:

```
for  (int i=vInicial; i < vFinal ;i=i+1)
 {
    //inicio y fin si existe más de una sentencia
 }
```

La instrucción for consta de tres partes:
- Creación e inicialización de la variable de control de la iteración. (int i=VInicial)
- Comprobación de la variable de control de la iteración (i< VFinal)
- Incremento del valor de la variable de control de la iteración (i=i+1)

Por ejemplo si se desean procesar los sueldos de 50 empleados. La instrucción for sería la siguiente:

```
for (int i =0;i < 50;i=i+1)
{
  //Las instrucciones entre el inicio y fin se realizaran 50 veces (0-49)
}
```

Sentencia while-Interacción con el usuario

Se utiliza cuando no se conoce la cantidad de veces que se van a procesar los elementos, pero se infiere que dicha cantidad no es significativa. La sentencia permite que los datos se lean mientras la respuesta que suministre el usuario sea la de continuar el procesamiento La estructura es la siguiente:

```
char respuesta;
respuesta='S';
   while (respuesta = ='S')
      {
        //Sentencias
        //
      cout <<" Desea procesar otros datos S/N " <<endl;
      cin >> respuesta;

      }
```

Sentencia while-Dato centinela

Un centinela es un valor especial utilizado para indicar el final del procesamiento de los datos. En este tipo de estructuras se necesita una variable que al finalizar los datos lea el centinela, el primer valor que este tome debe leerse antes de la estructura iterativa luego, en la condición de la estructura se verifica si el valor de la variable coincide con el centinela, si coincide, ha llegado a su fin. Es importante señalar que si existen varios atributos se selecciona el más representativo como dato centinela. El valor del centinela debe ser del mismo tipo que la variable que la lee y además debe ser un dato que no ocurra en la realidad. Suponga que se van a leer placas de automóviles y no se conoce la cantidad de automóviles. Como las placas son de tipo string, puede leerse * como centinela , estando en presencia de un dato que no ocurre en la realidad pero sigue siendo de tipo string.

La estructura general es la siguiente:

INVESTIGAR LA ESTRUCTURA

Metodología orientada a objetos

Para implementar el uso de estructuras iterativas simples en la metodología orientada a objetos se definen los siguientes pasos:

Análisis del problema:

1. Asignación de responsabilidades
2. Relaciones entre clases
3. Frecuencia de interfaces y métodos
4. Esquema de la estructura iterativa

Observaciones:

- ✓ La técnica del subrayado así como la escritura de los métodos set y get en la asignación de responsabilidades no están presentes en el análisis para este nivel de estructuras iterativas.
- ✓ En el paso relaciones entre clases se indica la relación y cardinalidad presente entre las clases Involucradas.
- ✓ El paso equivalente a la descripción de los métodos con estructura secuencial /selectiva en este nivel se denomina frecuencia de interfaces y métodos.
- ✓ El esquema de la estructura iterativa forma parte del análisis y guarda relación con la estructura de cuerpo principal, en el se indica la condición lógica que controla la estructura (for o while)

Diseño:

El diseño se expresará de acuerdo a los tipos de datos en C++

Implementación:

Se desarrolla la definición de las estructuras de las clases involucradas, el desarrollo de todos los métodos de la clase que contenga cálculos aritméticos así como también los métodos acumular, contar, mayor o menor y se desarrolla el cuerpo principal reflejando la estructura iterativa correspondiente.

Contadores, Acumuladores, Mayor y Menor

Dentro de las estructuras iterativas o repetitivas se encuentran elementos como: contar, acumular, mayor y menor de un conjunto de datos, así como también la lectura y escritura múltiple.

Contador:

El término está asociado a la actividad de contar. Es un tipo de variable que se incrementa según un valor constante. En la vida cotidiana, existen procesos que involucran contar, cada uno de ellos se realiza mediante el incremento de un valor constante, en el caso de las horas y los días ese incremento o valor constante es 1.

Formato de un contador:

```
contador = contador + 1
```

Ejemplos:

```
contVisitas =contVisitas + 1
contMenores =contMenores + 1
```

Todo proceso de contar involucra un valor inicial y se inicializa en el constructor de su clase, antes de iniciar el conteo (frecuencia única). Las veces que se realice un conteo lo determina la cantidad de elementos a procesar, una vez terminado el conteo en algunos casos es necesario conocer ese valor, el cual se muestra una única vez. El valor inicial para un contador puede ser cero (0) ó uno (1), se utilizará cero (0) en la mayoría de los casos. Un contador es una variable de tipo entera. Se utilizará contadores en caso de requerir cantidades, por ejemplo: cantidad de alumnos, vuelos, entradas vendidas etc.

Operaciones:
Inicializar: se realiza en el constructor de la clase mayor (Frecuencia única)

```
cont= 0;
```

Aplicar formula del contador (Frecuencia múltiple)

```
cont = cont+1;
```

Imprimir el contador (Frecuencia única)

```
cout<< "La cantidad es: ";
cout<< elObjeto.getCont();
```

Acumulador:

Se utiliza este tipo de variable cuando es necesario totalizar ó sumar valores sucesivamente. Ejemplo de acumuladores lo son: totalizar sueldos, monto final a pagar ó totalizar montos con los cuales podemos posteriormente calcular un promedio.

Formato:

acumulador = acumulador +variable

Ejemplos:

```
acumEdad   = acumEdad  + edad
acumSueldo  = acumSueldo + sueldo
```

El valor inicial para un acumulador deber ser cero (0). Un acumulador es una variable cuyo tipo de dato depende del tipo de variable que acumula. Ejemplo: Si la variable edad es de tipo int entonces acumEdad es de tipo int; si consideramos la variable sueldo, sueldo es de tipo float, entonces acumSueldo debe ser considerado de tipo float y su valor inicial seria 0.00

Operaciones:
Inicializar, se realiza en el constructor de la clase mayor (Frecuencia única)

```
acum= 0;
```

Aplicar formula del acumulador (Frecuencia múltiple)

```
acum=  acum+variable;
```

Imprimir ó mostrar el acumulador (Frecuencia única)

```
cout << "El total acumulado es: " <<obj.getAcum();
```

Mayor y Menor de un conjunto de elementos.

Cálculo del mayor valor de un grupo de datos:

El cálculo del mayor valor se realiza mediante la siguiente instrucción:

```
if (variable > mayor)
    mayor=variable
```

* La variable mayor se inicializa en cero.

Cálculo del menor valor de un grupo de datos:

El cálculo del menor valor se realiza mediante la siguiente instrucción:

```
if (variable < menor)
    menor =variable
```

- La variable menor se inicializa en un valor inalcanzable para la variable que se selecciona

Notas:

- Las variables de tipo contador, acumulador, menor y mayor no tienen métodos set ya que en el constructor se le asigna un valor inicial por defecto.

- Los métodos contar, acumular, menor y mayor son métodos que no retornan; ellos están dentro de la estructura iterativa si se desea conocer el valor final de un contador, total acumulado, mayor o menor debe invocarse el método get de cada uno de los anteriormente citados fuera de la estructura iterativa.