# CICLOS CUANTITATIVOS

Los ciclos son estructuras repetitivas y tiene dos esquemas: Cuantitativos y Cualitativos. De manera general, todo ciclo debe tener tres aspectos fundamentales:

1. Dónde Inicia (Límite Inicial).
2. Dónde termina (Límite Final).
3. Una instrucción que modifica el inicio y hace que llegue a su terminación (Incremento o Decremento, dependiendo de los límites).

En esta clase, se tratará el esquema de Ciclos Cuantitativos y de estos existen tres tipos:

- Ciclo Mientras.
- Ciclo Haga Mientras.
- Ciclo Para.

## CICLO MIENTRAS

**ESTRUCTURA:**

```
//Variable inicializada (Dónde inicia)
Mq  (expresión lógica) haga
    //secuencia de instrucciones
    //instrucción que modifica la expresión lógica
FMq
```

La expresión lógica se evalúa cada vez que se repite el proceso. Esta controla **Dónde termina**.

- Si la expresión lógica es falsa, la secuencia de instrucciones no se ejecuta.
- Si es verdadera, se ejecuta una vez y automáticamente regresa a evaluar la expresión y así sucesivamente hasta que sea falsa.

En el manejo de ciclos, se manejan dos tipos de variables fundamentales: los contadores y los acumuladores.

**CONTADOR:**

- Es una variable que se utiliza para el conteo de acciones internas del ciclo.
- Cada vez que el ciclo se repite, esta variable aumenta o disminuye en un valor constante (Ej: contador ← contador + 1).
- Antes de usarse, se debe inicializar (contador ← 0).

**EJEMPLO 1:**

El siguiente ciclo, se repite 10 veces:

```
contador ← 1 // Límite Inicial

Mq (contador < = 10) haga   // Límite Final
        //secuencia de instrucciones

        contador ← contador + 1  // Incremento

FMq
```

**ACUMULADOR:**

- Es una variable que almacena cantidades variables resultado de procesos sucesivos.
- Se diferencia del contador, en que el incremento es variable, en lugar de constante (Ej: acumulador ← acumulador + valor).
- Antes de usarse, se debe inicializar (acumulador ← 0).

**EJEMPLO 2:**

El siguiente ciclo, acumula o suma, los números del 1 al 9:

```
contador ← 1            //Inicialización de contador y acumulador
acumulador ← 0

Mq (contador < 10) haga

        acumulador ← acumulador + contador
        contador ← contador + 1

FMq
```

**EJEMPLO 3:**

Hacer un algoritmo que encuentre la suma de n números digitados por el usuario.

```
Inicio
    Entero: n
    Real: suma

    Escriba: "Digite la cantidad de números a sumar"
    Lea: n

    suma ← OperacionSuma (n)

    Escriba: "La suma de los", n, "números que digitó es igual a", suma
Fin

Funcion OperacionSuma (Entero: N)

    Entero: con←0
    Real: num, acum←0

    Mq (con < N) haga

        Escriba: "Digite el número"
        Lea: num

        acum ← acum + num

        con ← con + 1
    FMq

    Retorne acum
FinFuncion
```

## CICLO HAGA MIENTRAS

**ESTRUCTURA:**

**//Variable inicializada (Dónde inicia)**

**Haga**
        //secuencia de instrucciones

        **//instrucción que modifica la expresión lógica**
**Mq (expresión lógica)**

La expresión lógica se evalúa cada vez que se repite el proceso. Esta controla **Dónde termina**.

- Como se observa en la representación, es similar al ciclo mientras, con la diferencia, que se utiliza en situaciones en las que se desea que se repita al menos una vez, una secuencia de instrucciones, antes de comprobar la expresión lógica del ciclo o condición de repetición.

- Las instrucciones sólo se ejecutan una vez, si la condición o expresión lógica es falsa.

**EJEMPLO 4: se hará el mismo ejemplo anterior, pero con Ciclo Haga Mq:**

En este ejemplo, solo se modificará la función, dado que el programa principal es el mismo. Se debe recordar que a diferencia del ciclo Mq, éste se ejecutaría al menos una vez.

```
Funcion OperacionSuma (Entero: N)

        Entero: con←0  // Límite Inicial
        Real: num, acum←0

        Haga
                Escriba: "Digite el número"
                Lea: num

                acum ← acum + num

                con ← con + 1  //Incremento
        Mq (con < N)  // Límite Final

    Retorne acum
FinFuncion
```

## CICLO PARA

**ESTRUCTURA:**

```
Para  (variable_control ← limite_inicial, limite_final, incremento) haga

        //secuencia de instrucciones

Fpara
```

El incremento o decremento se hace automático, no se necesita una instrucción que aumente el contador

**EJEMPLO 5: se hará el mismo ejemplo anterior, pero con Ciclo Para:**

Solo se modificará la función, dado que el programa principal es el mismo.

```
Funcion OperacionSuma (Entero: N)

        Entero: con
        Real: num, acum←0

        Para (con ← 0, N-1, 1) haga  //límite inicial = 0, límite final = N-1, incremento = 1
                Escriba: "Digite el número"
                Lea: num

                acum ← acum + num
        FPara

   Retorne acum
FinFuncion
```

**EJEMPLO 6:**

Para cada uno de los n estudiantes del POLI, donde cada uno cursa 3 materias, se tienen los siguientes datos:

- Código estudiante.
- Nota de cada una de las materias.
- Número de créditos de cada una de las materias.

Hacer un algoritmo que encuentre para cada estudiante, el número de créditos cursados y su promedio.

```
Inicio
    Entero: N

    Escriba: "Digite el número de estudiantes"
    Lea: N

    Promedio (N)
Fin
```

```
Procedimiento Promedio (Entero: n)
 Entero: con←1
 Real: nota1, ncreditos1, nota2, ncreditos2, nota3, ncreditos3,
 Real: codigo, totalcreditos, promedio

 Mq (con < = n) haga
        Escriba: “Digite su código”
        Lea: codigo
        Escriba: “Digite la nota de la materia 1 y el número de créditos de la misma”
        Lea: nota1, ncreditos1
        Escriba: “Digite la nota de la materia 2 y el número de créditos de la misma”
        Lea: nota2, ncreditos2
        Escriba: “Digite la nota de la materia 3 y el número de créditos de la misma”
        Lea: nota3, ncreditos3

        totalcreditos ← ncreditos1 + ncreditos2 + ncreditos3
        promedio ← (nota1* ncreditos1 +nota2*ncreditos2 +nota3*ncreditos3)/totalcreditos

        Escriba: “El estudiante con código”, codigo, “cursó”, totalcreditos, “créditos y tiene
        un promedio de”, promedio

        con ← con + 1
FMq

FinProcedimiento
```