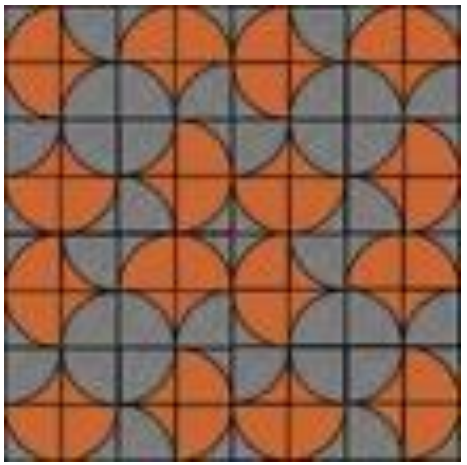
Estructuras de Repetición Algoritmos

- Mientras
- Hacer-Mientras
- Para



Repetición

- Las estructuras de repetición permiten la ejecución repetida de una lista o secuencia de instrucciones (también llamada bloque de instrucciones). El número de veces que el bloque de instrucciones se ejecutará se puede especificar de manera explícita o a través de una condición que indica cuando se ejecuta de nuevo o cuando no. A cada ejecución el bloque de instrucciones se le conoce como una **iteración.**

Tipos de estructuras de repetición

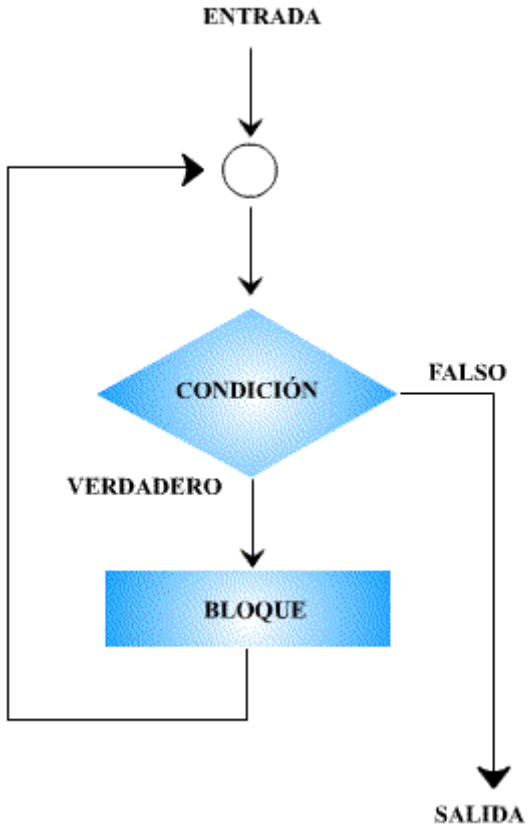
Existen tres tipos principales de sentencias de repetición

- **Ciclo MIENTRAS**
- **Ciclo HACER-MIENTRAS**
- **Ciclo PARA**

CICLO MIENTRAS

El **CICLO MIENTRAS** ejecuta un bloque de acciones 'mientras' que una condición dada se cumpla, es decir, cuando la condición se evalúa verdadera. La condición es evaluada antes de ejecutar el bloque de acciones y si la condición no se cumple, el bloque no se ejecuta. De esta manera es que el número de repeticiones del bloque de acciones sea cero, pues, si la condición de entrada se evalúa falsa, el bloque no será ejecutado alguna vez. La forma general del ciclo mientras es la siguiente:

Diagrama de Flujo y Pseudo código



Mientras(condición)

Bloque de instrucciones

Fin_mientras

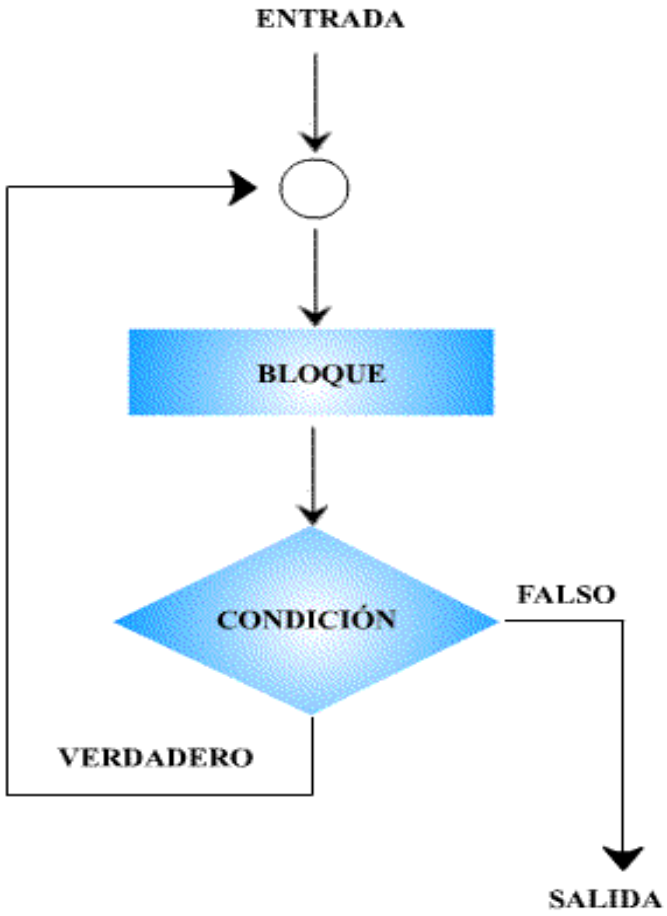
Ejemplo

- El problema es calcular la suma de los números naturales desde 1 hasta **n.**
- **Inicio**
- Entero i,n,Suma
- Leer(n)
- i=1
- Suma =0
- **MIENTRAS** (i<=n)
- Suma=Suma+i
- i=i+1
- **Fin_mientras**
- Escribir (Suma)
- **Fin**

Ciclo Hacer-Mientras

- El **CICLO HACER-MIENTRAS** es similar al ciclo mientras, la diferencia radica en el momento de evaluación de la condición. En el ciclo hacer-mientras la condición se evalúa antes de la ejecución del bloque, en el ciclo HACER-MIENTRAS la condición se evalúa después de ejecutar el bloque de acciones, por lo tanto, el bloque se ejecuta por lo menos ***una vez***. El bloque se ejecuta nuevamente si la condición se evalúa a verdadero y no se ejecuta más si se evalúa falso. La forma general del ciclo HACER-MIENTRAS es la siguiente:

Diagrama de Flujo y Pseudo código



- HACER

  Bloque de instrucciones

- MIENTRA (Condición)

Ejemplo:

- El problema es calcular la suma de los números naturales desde 1 hasta **n.**
- **Inicio**
- Entero i, n, Suma
- Leer(n)
- i=1
- Suma =0
- **HACER**
- Suma=Suma+i
- i=i+1
- **MIENTRAS** (i<=n)

- Escribir (Suma)
- **Fin**

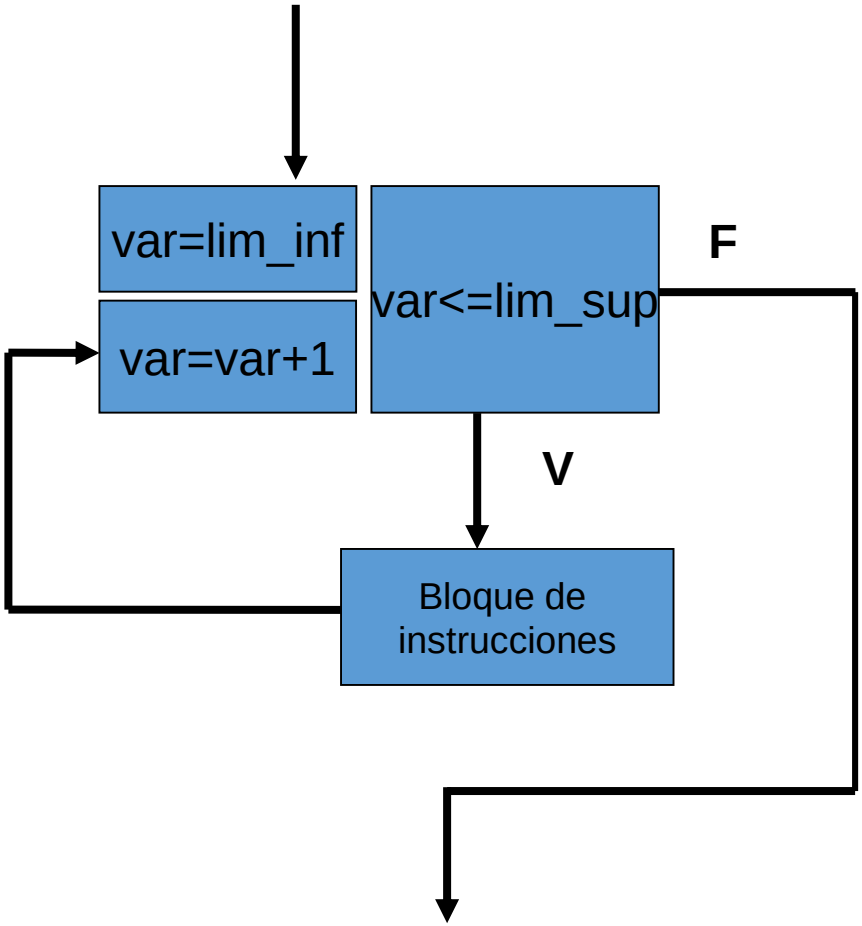
Ciclo PARA

- El **CICLO PARA** permite la ejecución repetida de un conjunto de acciones. El número de veces que el bloque es ejecutado está determinado por los valores que puede tomar una variable contadora (de tipo entero), en un rango definido por un límite inferior (inclusive) y un límite superior (inclusive). Después de ejecutar el bloque de acciones en cada iteración la variable contadora es incrementada en uno (1) automáticamente y en el momento en que la variable sobrepasa el límite superior el ciclo termina.

Ciclo PARA Continuación

- El valor final de la variable contadora depende mucho del lenguaje de programación utilizado, por lo tanto, no es recomendable diseñar algoritmos que utilicen el valor de la variable contadora de un ciclo **para**, después de ejecutar el mismo. De la definición del ciclo **para** se puede inferir que el bloque de acciones no se ejecuta alguna vez si el límite inferior es mayor al límite superior y que si el límite superior es mayor o igual al límite superior, el número de veces que el conjunto de acciones se ejecutará es igual a uno más el límite superior menos el límite inferior. La forma general del ciclo para es la siguiente:

Diagrama de Flujo y Pseudo código



Para var= **lim_inf** hasta **lím_super** hacer

< bloque de instrucciones >

Fin_para

Ejemplo. Ciclo Para

- Leer 20 números e imprimir cuantos son positivos, cuantos negativos y cuantos neutros.

```
Inicio
  cn = 0
  cp = 0
  cneg = 0
  Para x = 1 hasta 20 hacer
        Leer num
        Sin num = 0 entonces
              cn = cn + 1
          si no
              Si num > 0 entonces
                    cp = cp + 1
                 si no
                    cneg = cneg + 1
              Fin-si
        Fin-si
  Fin-para
  Imprimir cn, cp, cneg
Fin.
```

Ejemplo 2. Ciclo Para

- El problema es calcular la suma de los números naturales desde 1 hasta **n.**
- **Inicio**
- Entero i, n, Suma
- Leer(n)
- i=1
- Suma =0
- **PARA i=1 hasta n hacer**
- Suma=Suma+i
- **Fin_para**

- Escribir (Suma)
- **Fin**

TIPOS DE VARIABLES UTILES EN REPETICIÓN

- **Variables contadoras**

  Como su nombre lo indica estas variables se usan fundamentalmente para contar, por lo tanto deben ser de tipo entero. Un ejemplo de este tipo de variables es la variable de control en un **ciclo para**. Una variable contadora se incrementa (o decrementa) en un valor constante en cada iteración del ciclo.

Ejemplo. Desarrollar un algoritmo que imprima los números impares en orden descendente que hay entre 1 y 100.

Variables acumuladoras

- La función de una variable acumuladora es almacenar valores numéricos que generalmente se suman (o multiplican) en cada iteración, por lo tanto la variable debe ser de tipo entero o real. Por ejemplo, en los diferentes algoritmos presentados para solucionar el problema de calcular la suma de los números naturales desde 1 hasta **n**, la variable **suma** es una variable **acumuladora**.
- **Ejemplo.** Calcular las suma de los cuadrados de los números entre 1 y 100.

Variables bandera

- Una variable bandera es utilizada dentro de la condición del ciclo, ya sea sin negar, negada o conectada con una expresión booleana, para determinar cuándo un ciclo se sigue iterando o cuándo no. De esta manera una variable bandera debe ser de tipo booleano. (El lenguaje C estándar no tiene tipos booleanos, C++ si)

Variables bandera. Cont.

- **Ejemplo.** Realizar un programa que lea una serie de números reales y los sume. El programa debe preguntar al usuario cuándo desea ingresar un siguiente dato y si el usuario responde que no desea ingresar más datos el programa debe confirmar la respuesta. Si el usuario desea continuar ingresando datos se debe seguir solicitando datos y si el usuario confirma su deseo de salir, el programa debe mostrar la suma de los datos leídos y terminar.

Traducción de Estructuras Repetitivas

	Pseudo Código	C++

mientras	**Mientras (condición) hacer**
	
	bloque_instrucciones
	
	fin_mientras
		while (condición) {
		
		bloque_instrucciones
		
		};

hacer-mientras	**Haga** bloque_instrucciones **mientras (condicion)**	do { bloque_instrucciones }**while** (condición);

para	**para variable (lim_inf hasta lim_sup) hacer**
	
	bloque_instrucciones
	
	lim_inf : valor inicial para la variable
	lim_sup:valor final para la variable
		for (var:=**lim_inf**;var <= **lim_sup; var** ++){
		
		bloque_instrucciones
		
		}