

Unidad 3

ESTRUCTURAS DE PROGRAMACIÓN

Objetivos

- Conocer y usar las estructuras básicas de programación.
- Interpretar los esquemas del flujo de control de las distintas estructuras.
- Correcta composición y escritura de las expresiones lógicas que sirven como condiciones.
- Diferenciar entre procesos que se excluyen y procesos dependientes.
- Aprender a construir bucles correctamente.
- Conocer y usar adecuadamente las variables de control de bucle (VCB).
- Conocer, distinguir, y saber usar, contadores, acumuladores, interruptores y datos centinelas.
- Realización de procesos de lectura en número determinado e indeterminado.
- Aprender otras estructuras de control.
- Construir algoritmos usando la estructura más adecuada en cada caso.
- Realizar las pruebas necesarias para detectar errores.
- Saber interpretar algoritmos sencillos ya resueltos.

Contenidos

- Estructura de control Secuencial
- Estructura de control Alternativa
 - SI_Si no
 - SI Anidados
 - SI_Si no_SI
- **Estructura de control Repetitiva**
 - **Bucles Mientras**
- **Contadores y Acumuladores**
- **Dato Centinela**
- **Indicadores**
- **Tipos de bucles**
- **Subtareas Iterativas**
- **Como diseñar bucles**
- Estructuras de Selección múltiple SEGÚN
 - Sintaxis
 - Aplicación a procesos controlados por MENÚS
- Otras estructuras repetitivas
 - Bucles REPETIR
 - Bucles PARA
- Criterios para la elección de una estructura repetitiva.
- Anexo. Números Aleatorios

Introducción

- Las estructuras de control
 - Son usadas por algunas metodologías de programación para la construcción de algoritmos.
- Con estas estructuras
 - se podrán construir desde los programas más sencillos hasta los más complicados,
- Por esta razón
 - es importante que se comprenda perfectamente como funcionan y aprender a utilizarlas correctamente.

Estructura de control secuencial

- **Orden físico**

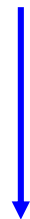
Disposición en que están escritas las instrucciones.

- **Orden lógico o flujo de control**

Orden de ejecución de las instrucciones en un programa.

Una **estructura secuencial** está formada por un conjunto de instrucciones que se ejecutan una tras otra, es decir coincide su orden físico y lógico.

Inst.....1
Inst.....2
.....
Inst.....n



Flujo de control

Ver ejercicio final Unidad 2

Ejercicio: Dado un capital, un rédito determinado y un tiempo de imposición de dicho capital, calcular el interés simple que se produce.

Estructura de control alternativa, condicional, de selección o de bifurcación

- Permite elegir entre distintas acciones
- El orden físico no coincide con el orden lógico.
- Formato:

SI (expresión booleana)

[ENTONCES]

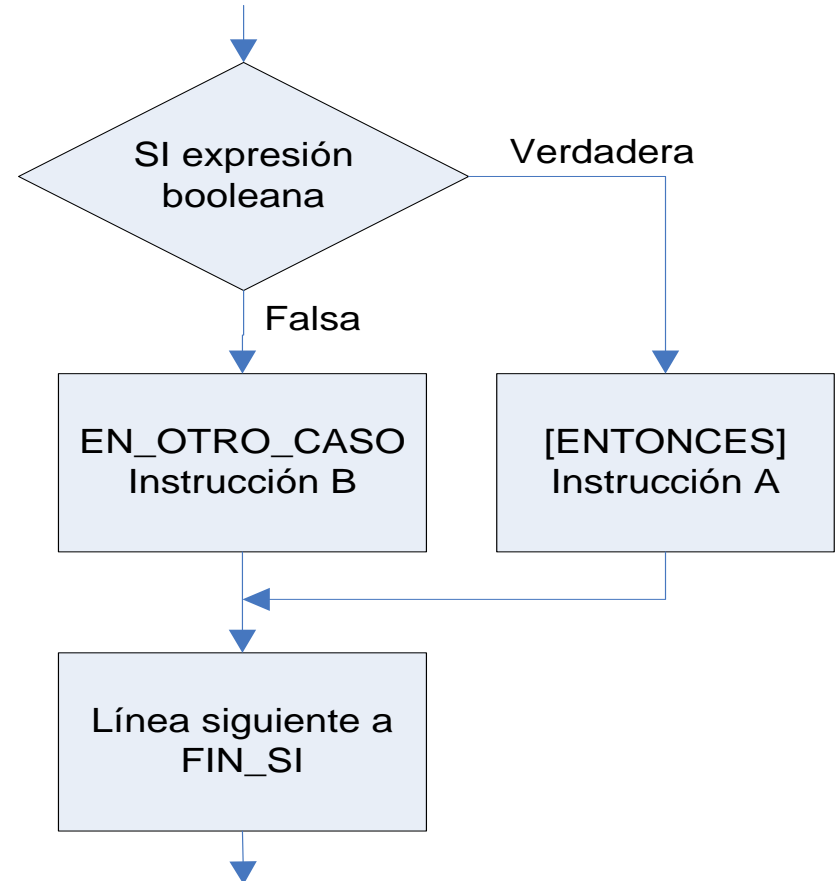
Instrucción A

[SINO/EN_OTRO_CASO]

Instrucción B

Fin_SI

//Línea siguiente a Fin_SI



Ejercicio: Programa para leer por teclado tres números si el primero es negativo calcular el producto y si no lo es calcular la suma.

Estructura de control alternativa

- **Si anidados:** Anidar estructuras es ejecutar una dentro de otra. Adecuada cuando una condición debe ser comprobada antes de que se evalúe otra condición.

SI (expresión 1)

SI (expresión 2)

SI (expresión 3)

Instrucciones A

SINO

Instrucciones B

FIN_SI

SINO

Instrucciones C

FIN_SI

SINO

Instrucciones D

FIN_SI

Deberá controlarse la legibilidad.

Mirar ejemplo pag. 7

- **Si SiNo Si:** Adecuado para comparaciones por rangos consecutivos

SI (expresión 1)

Instrucción A

[SINO]

SI (expresión 2)

Instrucción B

[SINO]

Instrucción C

FIN_SI

FIN_SI

Realizar el ejercicio de la página 10

Modificar el ejercicio de la página 8 usando una variable cadena para almacenar el mensaje

Estructuras repetitivas o iterativas

- Existen procesos en los que **se deben ejecutar** acciones **mientras** se cumpla una condición determinada.
- Ej.: “caminar **mientras** no esté cansado”, “dormir **mientras** sea de noche”.
- También llamada **Bucle**, es instrucción formada por varias sentencias conocidas como **Cuerpo del bucle**, que se repite **n** veces (**n iteraciones**).
- **Salida del bucle**: se produce cuando llega la **Condición de terminación**.

Formato:

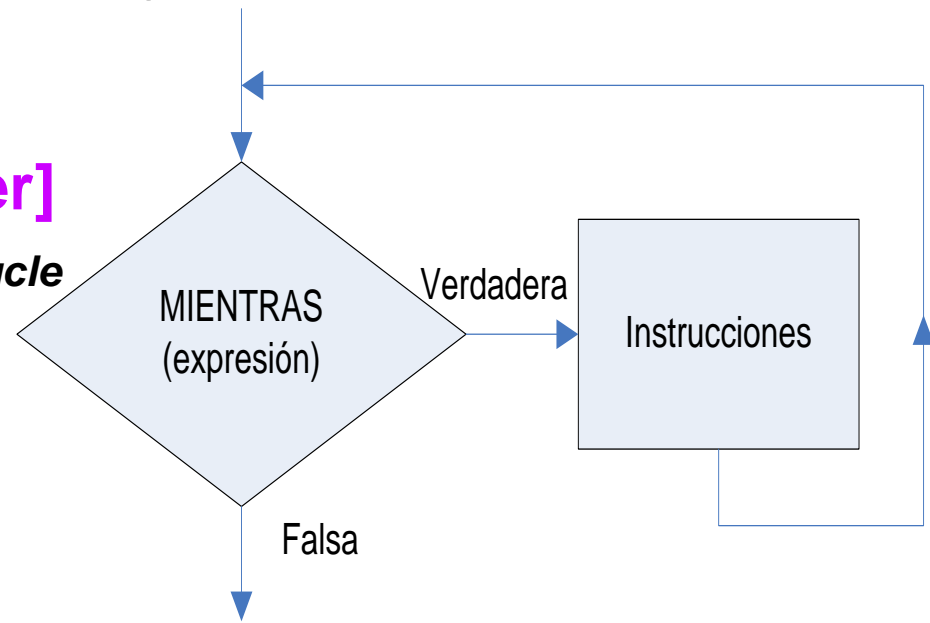
Mientras (*Expresión*) [Hacer]

Instrucciones *//cuerpo del bucle*

Fin_mientras

//Línea siguiente a Fin_mientras

■ **Mirar ejemplo pag. 13**



Estructuras repetitivas

- La estructura de control *mientras* también puede utilizarse *anidada*

Formato:

mientras (expresión 1) [hacer]

mientras(expresión 2) [hacer]

mientras(expresión 3) [hacer]

Instrucciones

fin_mientras

fin_mientras

fin_mientras

Variables típicas especializadas

■ CONTADORES

- ❑ Definición: Variable que se incrementa o decrementa de forma **constante**.
- ❑ Deben ser inicializados siempre
- ❑ Deben ser actualizados según el siguiente formato:

VariableContador = VariableContador + (o -) Constante

- ❑ Ej: marcador de un partido de fútbol

Ejercicio página 16

INSTANTE	LOCAL	VISITANTE
0 (Antes de comenzar el partido)	0	0
A (Un momento cualquiera)	2	1
B (Después de marcar otro gol el Local)	3	1

Variables típicas especializadas

ACUMULADORES

- ❑ Definición: Variable que se incrementa o decrementa de forma **variable**.
- ❑ Deben inicializarse, tanto los sumativos (a 0) como los multiplicativos (a 1).
- ❑ Deben ser actualizados según el siguiente formato:

VarAcum = VarAcum + (-, *, :) Variable

- ❑ Tipos
 - Sumativos
 - Multiplicativos

Ejercicios página 19 y 21

Variables típicas especializadas

■ DATO CENTINELA

- ❑ **Definición:** Es un **dato** que, siendo del mismo tipo, no pertenece al rango de valores permitidos para una variable, por lo que se puede usar para finalizar un proceso de entrada de datos.
- ❑ **Ej:**
Usar -1 para finalizar una entrada de números positivos
Usar “FIN” para acabar la lectura de nombres de alumnos.

Ejercicio: Diseñar un algoritmo que sume números impares negativos. La entrada de datos acaba cuando se lea el 0.

Ver ejercicio página 23

Variables típicas especializadas

■ SWITCHES, FLAGS O INDICADORES

- Son variables booleanas que se utilizan para controlar el flujo lógico de un programa.
- Al valor que posee en un momento dado se le suele llamar también **estado**.
- Se usan para:
 - Saber si el programa ha pasado por un determinado punto preguntando por su estado.
 - Salir de un ciclo cuando toma un determinado valor.
 - Ejecutar una u otra acción dependiendo de su estado.

Ejercicio: Modificar el algoritmo de la diapositiva anterior usando un indicador para controlar la entrada de datos.

Ver ejercicio página 25

Tipos de bucles

VCB: Variable de Control del Bucle

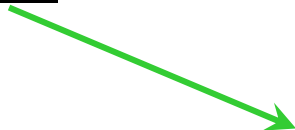
Objeto sobre el que recae el control para la ejecución del bucle.

■ **CONTROLADOS POR CONTADOR:**



“batir la mezcla 100 veces”

■ **CONTROLADOS POR SUCESOS**



“batir la mezcla hasta que tenga textura de almíbar”

Bucles controlados por Contador

El número de veces que se repite es conocido con anterioridad.

- **Características:**

- El cuerpo se ejecuta siempre un número determinado de veces
- Usa un contador como VCB

- **Operaciones**

- Inicializar contador (VCB)
- Evaluar el contador
- Variar o actualizar el contador

- **Cuidado con:**

- Inicializar correctamente la VCB antes de entrar en el bucle.
- Variar correctamente la VCB, dentro del bucle, antes de cada siguiente iteración, para propiciar la condición de salida.
- Estudiar las veces que se repite el bucle.

Bucles controlados por Contador

Formato	Ejemplo: contar diez números
<i>//Se repite un n° conocido de veces</i>	CONSTANTE N = 10
Inicializar la VCB	Contador = 0 <i>/*Inicializar VCB*/</i>
.....
Mientras (Expresión para Evaluar VCB)	Mientras (contador < N) <i>/* Comparar VCB*/</i>
.....
.....
Expresión para Variar la VCB	Contador = Contador + 1 <i>/*Incrementar o decrementar VCB*/</i>
Fin_mientras	Fin_mientras

Ejemplo: Sumar 20 números leídos de teclado y presentar en pantalla el resultado.

■ CONTROLADO POR CONTADOR

Pseudocódigo generalizado

Inicio

Inicializar el contador

mientras no haya 20 n^os leídos // *Evaluar contador*

 Obtener numero

 Realizar suma

actualización del contador

finmientras

Presentar resultados

fin

Pseudocódigo detallado

Entorno

Constantes

MAX = 20

Variables

entero **cont**, numero, suma = 0 // *Acumulador*

Programa principal

Inicio

cont = 0 // *Inicialización de VCB*

mientras (**cont < MAX**) // *Evaluación de VCB*

 //*Obtener numero*

 Escribir ("Teclee un número")

 Leer (Numero)

 //*Realizar suma*

 suma = suma + Numero //*Acumulador*

cont = cont + 1 //*actualización de VCB*

finmientras

escribir ("Total:", suma)

fin

Ejercicio pag 29

Bucles controlados por Sucesos

El bucle se repite hasta que algún acontecimiento dentro del cuerpo del bucle provoca la terminación.

■ Tipos:

- Controlados por centinela.
- Controlados por indicador
- Controlados por fin de fichero

■ Características: Se usarán en el tratamiento de ficheros

- No se sabe la cantidad de veces que se repetirá el bucle

■ Operaciones

- Dependen del tipo de bucle

Bucles controlados por centinela

■ Características:

- El valor de la VCB que provoca la salida, no es válido para la entrada de datos a procesar.

■ Operaciones

- Lectura anticipada.
- Evaluación por comparación con el dato centinela.
- Lectura final.

Formato:

Lectura de datos *//lectura anticipada*

Mientras (*Evaluación de VCB*) /*¿El dato leído es el centinela? */

// Cuerpo del bucle

.....

<proceso>

.....

Lectura de datos *//lectura final*

Fin_mientras

Bucles controlados por indicador

Formato:

■ Características:

- El cuerpo del bucle registra si se ha producido o no el suceso que controla el proceso.

■ Operaciones

- Inicializar indicador.
- Evaluar indicador.
- Actualizar indicador.

Inicializar indicador */*Para que entre en el bucle la primera vez, o bien realizar proceso que hace tomar estado al indicador*/!* **¡Ojo!**

Mientras (Evaluación del indicador)

// Cuerpo del bucle

.....

<proceso>

.....

Actualización del indicador

Fin_mientras

Ejemplo: Sumar 20 números leídos de teclado y presentar en pantalla

■ CONTROLADO POR SUCESO

■ indicador

■ CONTROLADO POR CONTADOR

Entorno

Constantes

MAX = 20

Variables

entero cont = 0, numero, suma = 0 **//Acumulador**

Inicio

cont = 0 **// Inicialización de VCB**

mientras (cont < MAX) **// Evaluación de VCB**

Escribir ("Teclee un número")

Leer (Numero)

suma = suma + Numero **// A. Acumulador**

cont = cont + 1 **//actualización de VCB**

finmientras

escribir ("Total:", suma)

fin

Ejercicio pag 33

Entorno

Constantes

MAX = 20

Variables

entero cont, numero, suma = 0

booleana salir

Inicio

cont = 0

salir = "falso" **// Inicialización de VCB**

mientras (salir == "falso") **// Evaluación de VCB**

Escribir ("Teclee un número")

Leer (Numero)

suma = suma + Numero

cont = cont + 1

// Registro del suceso que provoca la actualización de la VCB

si (cont >= MAX)

salir = "verdadero" **//actualización de VCB**

finsi

finmientras

escribir ("Total:", suma)

fin

¡Ojo! discutir

Subtareas iterativas

■ Bucles contadores:

Contienen alguna variable contadora, pero no son bucles controlados por contador.

■ Tipos:

- **Contador de iteración:** termina valiendo igual que el número de iteraciones del bucle.
Ej.: proceso de números positivos usando el centinela -1 como final de proceso y deseamos contar cuantos números han sido procesados.
- **Contador de sucesos:** Cuenta las veces que ocurre un evento durante el proceso que realiza el bucle.
Ej.: Proceso con números positivos y contar cuantos de ellos son pares.

Ejercicio pag 35. Contar además cuántos caracteres son vocales.

■ Bucles sumadores:

Contienen alguna variable acumuladora, pero son bucles controlados por cualquier tipo de VCB.

Ejercicio pag 37

Diseñar bucles correctamente

■ Invariante del bucle:

- ❑ Definición: Está formado por las condiciones que deben cumplirse al comienzo de cada iteración para ejecutar el bucle.
- Debe cumplirse la condición de entrada, esto es parte del invariante, pero no lo es todo,
- Además:
 - ❑ Debe conocerse el rango de la VCB
 - ❑ inicialización de contadores y sumadores,
 - ❑ actualizaciones de contadores y sumadores, etc.

Diseñar bucles correctamente

- ❑ Diseño: Cómo determinar el invariante.
Dar respuesta a las siguientes preguntas
 - ¿Cuál es la condición de terminación del bucle? ¿y la de entrada?
 - ¿Cómo debe inicializarse y cómo se actualiza la VCB?
 - ¿Cuál es el proceso que se repite?
 - ¿Cómo debe actualizarse ese proceso?

Escritas las sentencias esenciales pueden añadirse las de refuerzo de código.

Se ilustrará con el siguiente ejemplo.

Ejemplo: Sumar N enteros impares leídos de teclado

■ **Bucle CONTROLADO POR INDICADOR**

El algoritmo debe tener al menos los siguientes pasos

.....

.....

Inicializar Indicador de parar a falso

Inicializar Contador y Acumulador

Mientras (Parar sea falso)

Obtener Numero

Si (Numero es impar)

ProcesarImpar

Si (Se han contado N impares)

Cambiar Parar a Cierto //actualizar la VCB

Finsi

Sino

Fin_Si

Fin_Mientras

.....

.....

Ejemplo: Sumar N enteros impares leídos de teclado

Detallamos un poco el código anterior

El algoritmo debe tener al menos los siguientes pasos

.....

Inicializar Indicador_Parar = falso */*o indicador_seguir inicializado a verdad*/*

Inicializar Cont_Imp a 0

Inicializar la suma a 0

Mientras (Indicador_Parar == falso)

 Leer (Numero)

 Si (Numero MOD 2 <> 0)

 Cont_Imp = Cont_Imp+1

 Suma = Suma +Numero

 Si (Cont_Imp == N)

 Indicador_Parar = Verdad

 Finsi

Sino

.....

.....

Fin_Si

Fin_Mientras

.....

Ver ejemplo pag. 40

Actividades de Diseño. Resumen

Hay que estudiar dos aspectos en el diseño de un bucle:

■ ***Diseño del flujo de control***

- ❑ Tipo de bucle: ¿VCB?
- ❑ Cómo y dónde se inicializa la VCB y cómo y dónde se actualiza

■ ***Diseño del proceso que se repite***

- ❑Cuál es el proceso que se repite
- ❑Cómo se inicializa y actualiza ese proceso

Actividades de Diseño. Resumen

■ Diseño del flujo de control

1. Tipo de bucle, ¿Cuál es la condición de terminación? ¿y la de entrada?. Para ello estudiar el texto.
2. ¿Cómo debe inicializarse y cómo se actualiza la VCB?
 - a) Bucle controlado por contador.
Iniciación : Antes de la primera iteración, normalmente a 0
Actualización : al final de cada iteración
 - a) Bucle controlado por centinela.
Iniciación : Lectura anticipada, antes de la primera iteración
Actualización : Lectura final, físicamente al final del bucle.
 - c) Bucle controlado por indicador.
Iniciación : a verdadero o falso, antes de la primera iteración
Actualización : Cambio de estado cuando ocurra el proceso que se debe evaluar.

Actividades de Diseño. Resumen

- Diseño del proceso que se repite
 1. ¿Cuál es el proceso que se repite?: leer, escribir, contar, sumar, ordenar,...
 2. ¿Cómo debe inicializarse y actualizarse ese proceso?: Depende de sus características.

Por ejemplo, si el proceso es contar iteraciones o sucesos:

Inicialización : contador de iteraciones y el contador de sucesos antes de la primera iteración.

Actualización:

 - el contador de iteraciones al final de cada iteración.
 - el contador de sucesos se actualiza cuando ocurra el suceso

Ejercicio pag. 43

Estructura de selección múltiple

- Permiten seleccionar varias acciones alternativas y elegir una de ellas en tiempo de ejecución.
- Aplicable a procesos controlados por menú.

```
según (expresión) [hacer]
    para expresión == constante 1
        instrucciones A
    para expresión == constante 2
        instrucciones B
    ....
[en otro caso]
    instrucciones X
finsegún
```

//Línea siguiente a Fin_según

Ejercicio pag. 49

PROGRAMA PRINCIPAL GENERALIZADO

Inicio

Presentar MENU

Elegir y *validar* Opción

Mientras no quiera finalizar // (Opcion <> 'F')

Obtener números /*Inicializa y actualiza el proceso*/

Según (Opcion)

Para Opcion == 'S'

Realizar SUMAR

Para Opcion == 'R'

Realizar RESTAR

Para Opcion == 'M'

Realizar MULTIPLICAR

Para Opcion == 'D'

Validar Numero2 y DIVIDIR

Fin Según

Presentar resultados

Presentar MENU

Elegir y *validar* Opcion

Fin_mientras

FinPP

Detallado pag. 51

Otras estructuras repetitivas

REPETIR

1

Repetir

Instrucciones

Mientras (Expresión) //mientras exp. cierta

2.

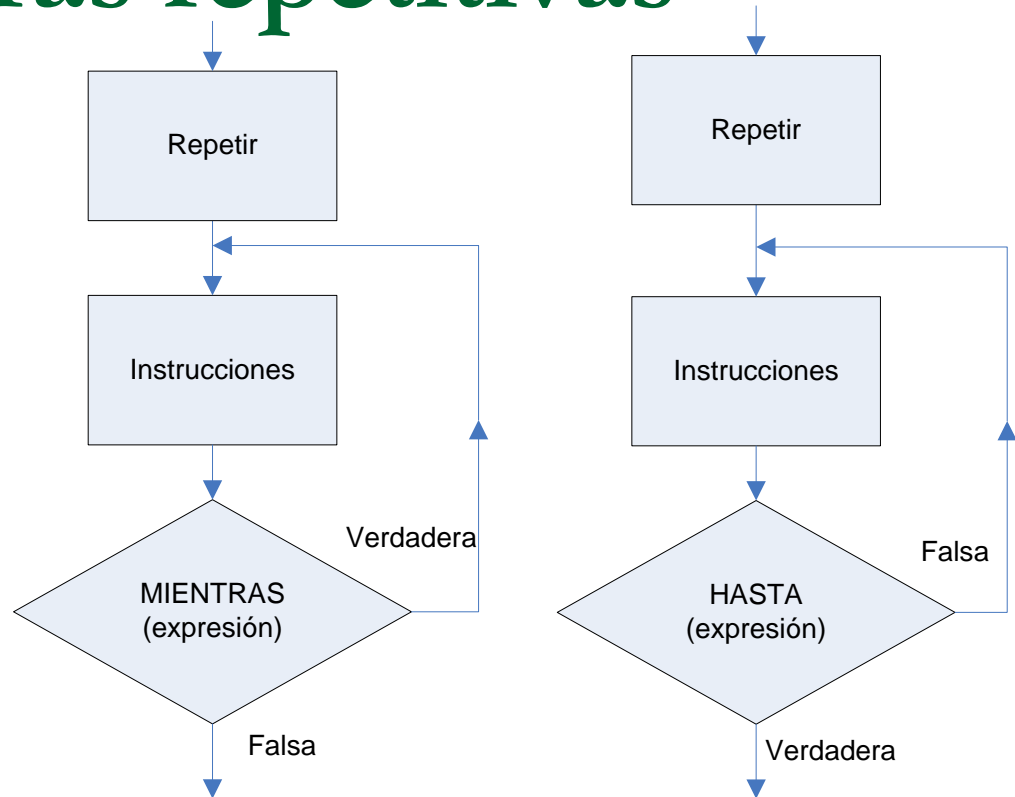
Repetir

Instrucciones

Hasta (Expresión)

//hasta exp. Cierta //mientras exp. falsa

Se conocen como estructuras **posttest**, a diferencia de **mientras** que se conoce como **pretest**



- Características:
 - ❑ Se ejecutan al menos una vez.
 - ❑ Útiles para validar datos de entrada.
 - ❑ Su implementación depende del lenguaje.

Ejemplo: Procesar calificaciones de exámenes que se leerán desde teclado. La puntuación debe estar entre 1 y 10, ambos inclusive. Validar la entrada.

Solución Repetir...Mientras

.....
Repetir
 Escribir (“Introduzca una nota ([1,10])”)
 Leer (nota)
Mientras (nota <1 o nota >10)

Solución Repetir...Hasta

.....
Repetir
 Escribir (“Introduzca una nota ([1,10])”)
 Leer (nota)
Hasta (nota >=1 y nota <=10)

Solución Mientras

.....
Escribir (“Introduzca una nota ([1,10])”)
Leer (nota)
Mientras (nota <1 o nota >10)
 Escribir (“Introduzca una nota ([1,10])”)
 Leer (nota)
Fin Mientras

¡Ojo ! : Es válido todo lo estudiado para diseño de bucles.

Otras estructuras repetitivas

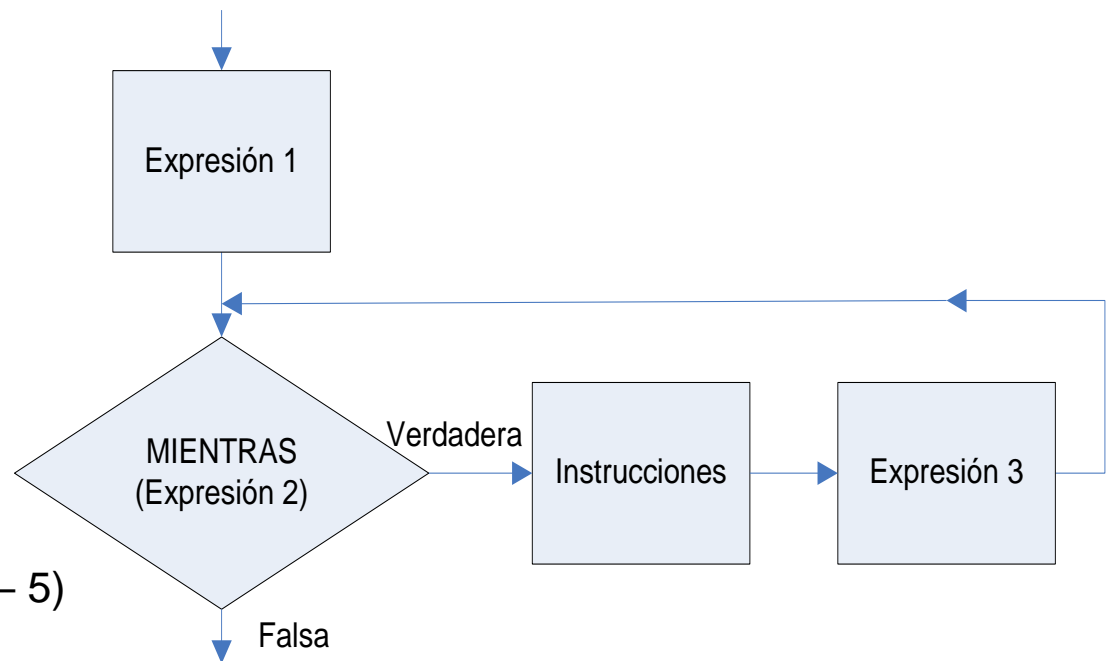
Bucle controlado por contador **PARA**

Para (*Expresión1*, Mientras *Expresión2*, *Expresión3*)

Instrucciones

Fin_Para

//Línea siguiente a Fin_Para



para (i =100, mientras (i > 50), i = i – 5)

escribir (i)

finpara

▪NOTA: Es válido todo lo estudiado para diseño de bucles.

Criterios para elegir tipo de bucle

- Si es controlado por contador usar PARA.
- Si es controlado por contador y suceso usar PARA o MIENTRAS. Si el nº de iteraciones es grande usar PARA.
- Si es controlado por suceso y debe realizarse al menos una vez, usar REPETIR.
- Si es controlado por suceso y no se sabe nada sobre la primera iteración, usar MIENTRAS.
- Si es apropiado MIENTRAS o REPETIR, usar el que mejor refleje el contexto.
- Siempre que exista duda usar MIENTRAS.

Números aleatorios

- Son generados al azar.
- La mayoría de los lenguajes tienen funcionalidades para generar números aleatorios.
- La forma de generarlos depende de cada lenguaje.
- En Java se puede usar la clase **Random** de **java.util**, o la función estática **random** de la clase **Math**.
- En pseudocódigo se seguirá la siguiente sintaxis:

Numero_Generado = GNA ()

- **GNA** (Generar Número Aleatorio), representa una función que genera números reales entre 0.0 y 1.0.
 - Si necesitamos un n° entre 0 y 10 basta con multiplicar por 10 y quedarse la parte entera. Si queremos un n° entre 0 y 10 incluido, multiplicamos por 11

" Si algo se consigue
¡bieeeen!;

Si no,
hay que volver a empezar. //Todo lo demás son
//fantasías."

Edouard Manet, pintor,
(París, 1832-1883)