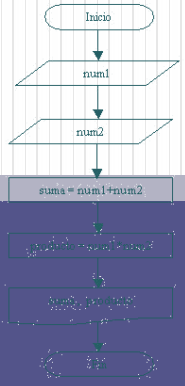
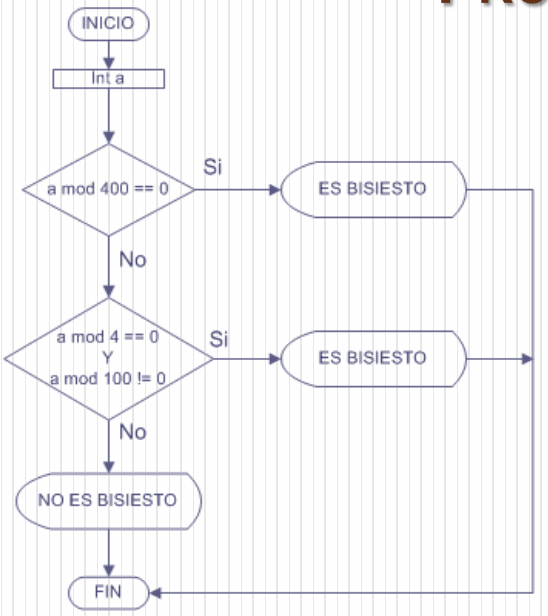
Analista Programador Universitario



Programación Estructurada

Programación Estructurada



Facultad de Ingeniería
Universidad Nacional de Jujuy

Índice

- Estructuras Repetitivas
  - PARA, MIENTRAS, REPETIR
  - Equivalencias entre estructuras repetitivas
  - Finalización de bucles
    - ✓ Por contador
    - ✓ Por valor centinela
    - ✓ Por bandera
- Anidamiento de Control
- Prueba de escritorio

Estructuras Repetitivas

- Soluciones que pueden expresarse mediante pasos secuenciales o la selección de 2 o más caminos de acción pueden construirse mediante estructuras secuenciales y/o selectivas.
- Los problemas cuya solución consiste en la repetición de conjuntos de acciones requieren estructuras especiales llamadas BUCLES.

Estructuras Repetitivas

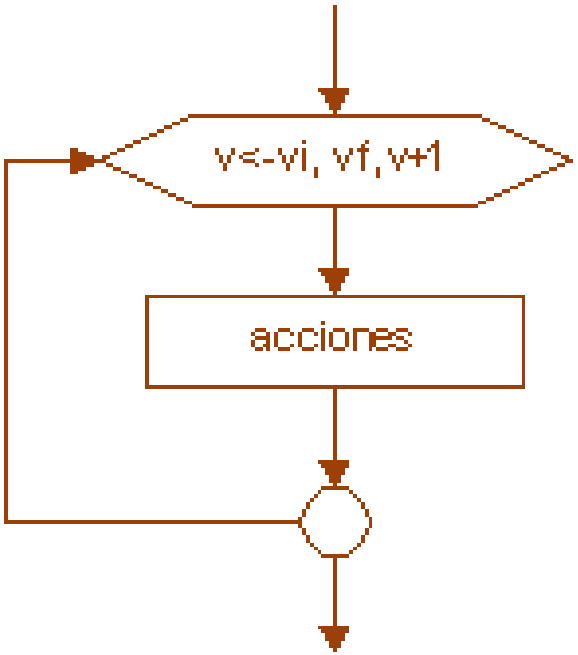
- Un bucle o *loop* es un conjunto de acciones que deben repetirse.
- El número de repeticiones (iteraciones) puede ser conocido a priori o no.
- En PE, las estructuras PARA, MIENTRAS y REPETIR permiten especificar el conjunto de acciones que deben ejecutarse en forma repetida.

Estructura PARA (1)

- La estructura PARA/FIN_PARA se aplica cuando el **número de repeticiones** a realizar es **conocido**.
- La estructura utiliza una variable de control que **cuenta** las repeticiones realizadas.
- La **variable de control** varía entre *valor_inicial* y *valor_final*.
- El **incremento** de la variable de control puede especificarse (por defecto es 1).

Estructura PARA (2)

```
PARA vc DESDE vi HASTA vf HACER CON PASO n
     acciones
FIN_PARA
```

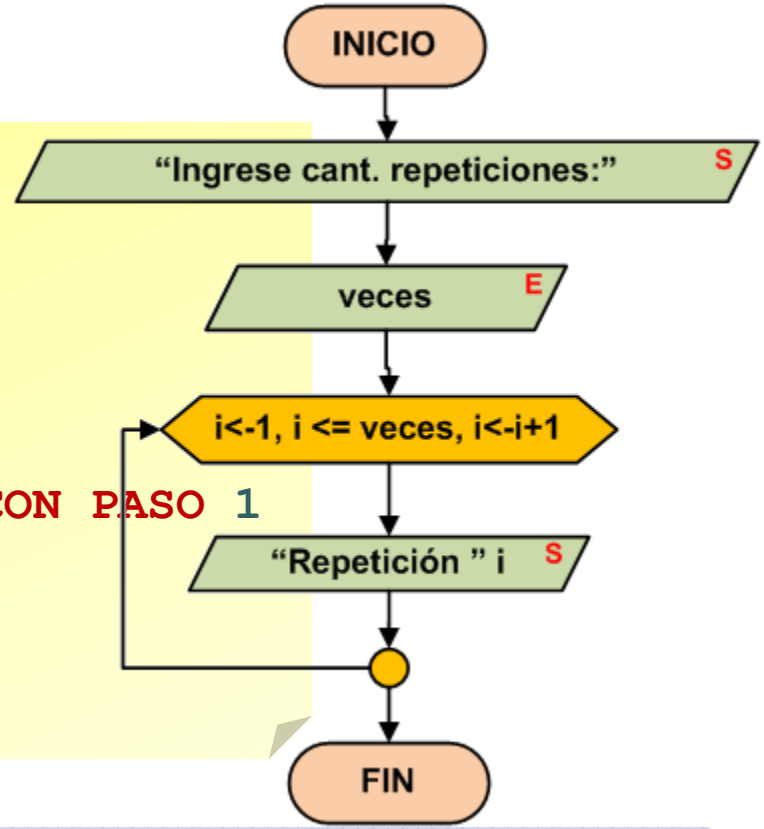


- ✓ **vc**: ***variable de control*** del bucle
- ✓ **vi**: ***valor inicial*** de la variable de control
- ✓ **vf**: ***valor final*** de la variable de control
- ✓ **n**: ***incremento*** de la variable de control

Ejemplo Repetitivas (1)

- Diseñe un algoritmo que muestre un mensaje ***n* veces**, *n* es ingresado por el usuario.

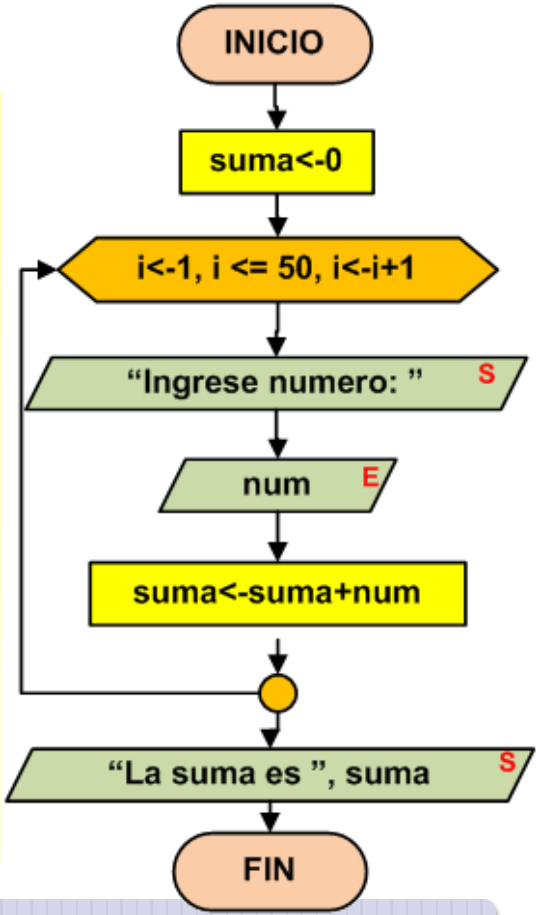
```
PROGRAMA ej_bucle_1
VARIABLES
   veces, i: ENTERO
INICIO
   ESCRIBIR "Ingrese cant. rep.: "
   LEER veces
   PARA i DESDE 1 HASTA veces HACER CON PASO 1
     ESCRIBIR "Repeticiones ", i
   FIN_PARA
FIN
```



Ejemplo Repetitivas (2)

- Diseñe un algoritmo que sume ***50*** **valores** ingresados por el usuario.

```
PROGRAMA ej_bucle_2
VARIABLES
    num, suma: REAL
    i: ENTERO
INICIO
   suma<-0 //inicialización de suma
   PARA i DESDE 1 HASTA 50 HACER CON PASO 1
     ESCRIBIR "Ingrese numero"
     LEER num
     suma<-suma+num
   FIN_PARA
   ESCRIBIR "La suma es ", suma
FIN
```

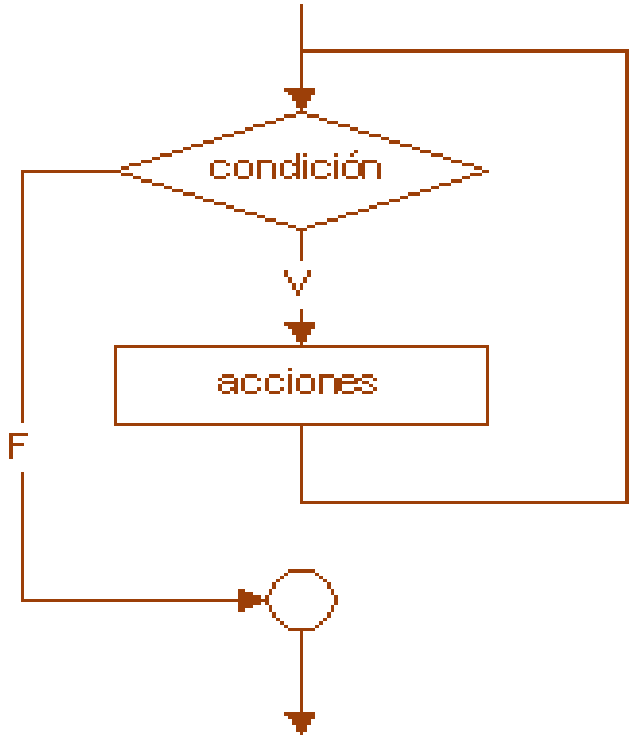


Estructura MIENTRAS (1)

- La estructura MIENTRAS repite un conjunto de acciones en tanto la condición de repetición sea VERDADERA.
- No necesita conocerse a priori el número de iteraciones a realizar.
- MIENTRAS es pre-condicional: la condición se evalúa antes de ejecutar el bloque de acciones (0 o más veces).
- Se aplica en cálculos aritméticos.

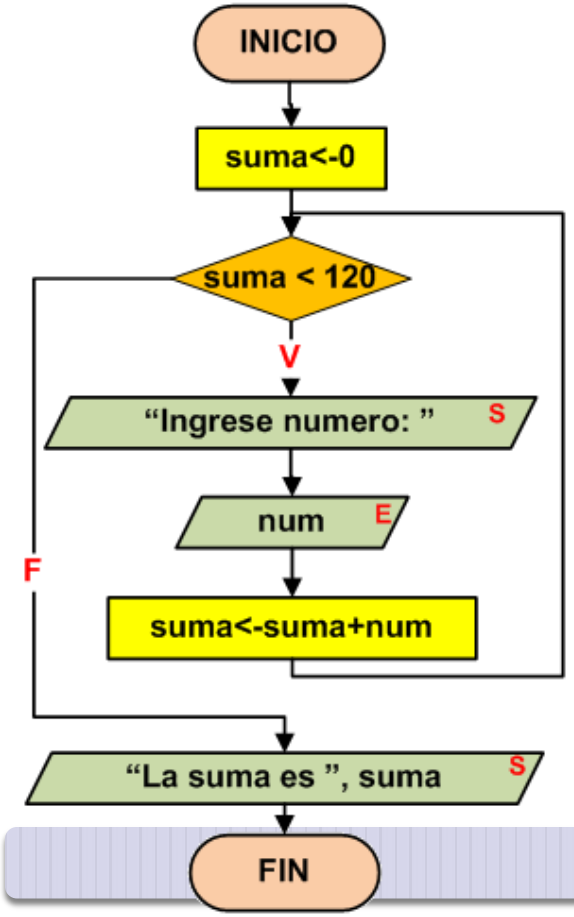
Estructura MIENTRAS (2)

```
MIENTRAS condición HACER
    acciones
FIN_MIENTRAS
```



Ejemplo Repetitivas (3)

- Diseñe un algoritmo que sume valores en tanto ésta **no supere el valor 120**.



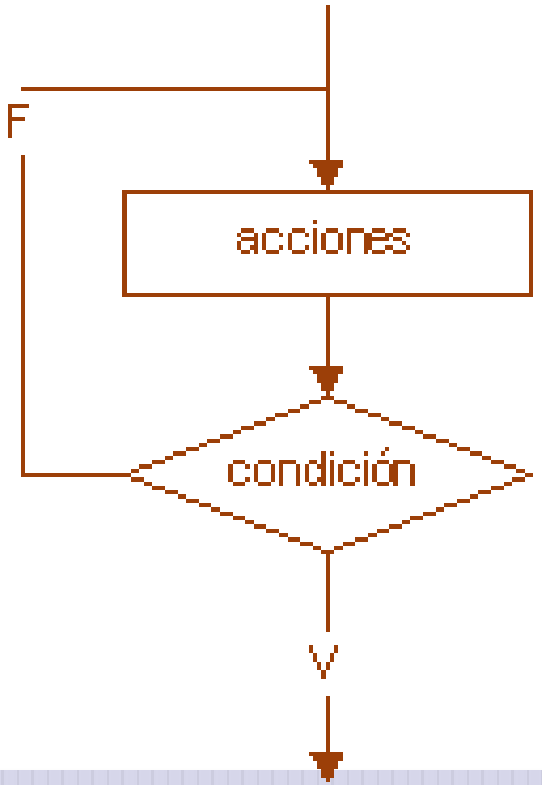

```
PROGRAMA ej_bucle_3
VARIABLES
    num, suma: ENTERO
INICIO
   suma<-0
   MIENTRAS suma < 120 HACER
     ESCRIBIR "Ingrese numero"
     LEER num
     suma<-suma+num
   FIN_MIENTRAS
   ESCRIBIR "La suma es ", suma
FIN
```

Estructura REPETIR (1)

- La estructura REPETIR repite un conjunto de acciones en tanto la condición de repetición sea FALSA.
- No necesita conocer a priori el número de iteraciones a realizar.
- REPETIR es pos-condicional: el bloque de acciones se ejecuta antes de evaluar la condición; si ésta es FALSA, el bloque de acciones se ejecuta nuevamente (1 o más veces).
- Se utiliza en el ingreso de datos.

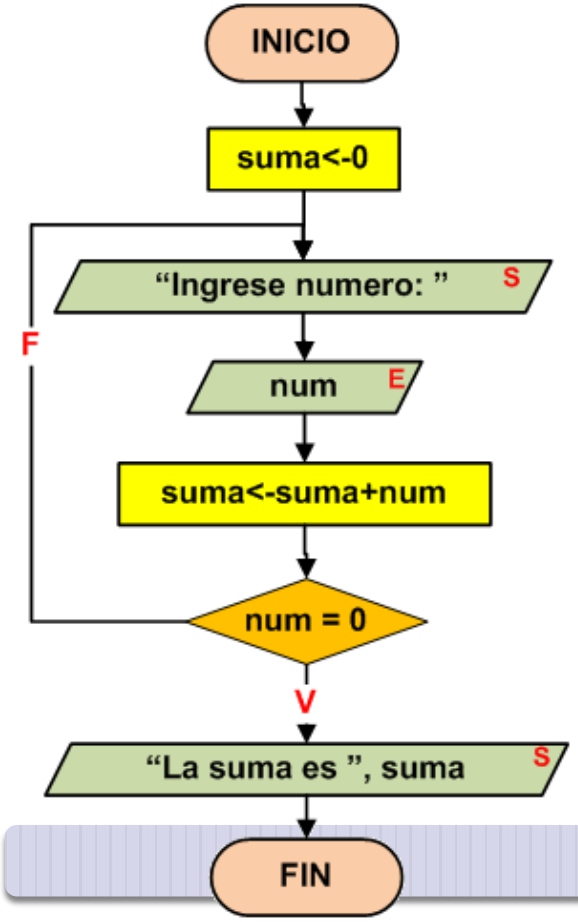
Estructura REPETIR (2)

```
REPETIR
      acciones
HASTA_QUE condición
```



Ejemplo Repetitivas (4)

- Diseñe un algoritmo que sume valores **hasta** que el usuario ingrese **un CERO**.




```
PROGRAMA ej_bucle_4
VARIABLES
    num, suma: ENTERO
INICIO
   suma<-0
   REPETIR
     ESCRIBIR "Ingrese numero"
     LEER num
     suma<-suma+num
   HASTA_QUE num = 0
   ESCRIBIR "La suma es ", suma
FIN
```

MIENTRAS y REPETIR

- Evaluación de la condición de repetición
  - MIENTRAS evalúa la condición antes de ejecutar el bloque de acciones (0 o más veces)
  - REPETIR evalúa la condición luego de ejecutar el bloque de acciones (1 o más veces)
- Finalización de bucle
  - MIENTRAS finaliza con condición FALSA
  - REPETIR finaliza con condición VERDADERA

PARA, MIENTRAS y REPETIR (1)

- Equivalencia entre PARA y MIENTRAS

```
PARA k DESDE 1 HASTA valor CON PASO n HACER
      Bloque de Acciones
FIN_PARA
```

```
k<-1 //inicialización de la var. de control
MIENTRAS k <= valor HACER //control valor final
      Bloque de Acciones
      k<-k+n //incremento
FIN_MIENTRAS
```

PARA, MIENTRAS y REPETIR (2)

- Equivalencia entre PARA y REPETIR

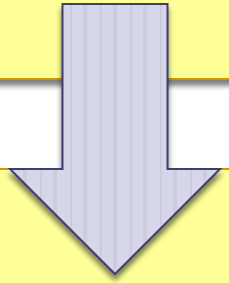
```
PARA k DESDE 1 HASTA valor CON PASO n HACER
      Bloque de Acciones
FIN_PARA
```

```
k<-1 //inicialización de la var. de control
REPETIR
      Bloque de Acciones
      k<-k+n //incremento
HASTA_QUE k > valor //control de valor final
```

PARA, MIENTRAS y REPETIR (3)

- Equivalencia entre MIENTRAS y REPETIR

```
MIENTRAS condición HACER
      Bloque de Acciones
      Acción que modifica la condición
FIN_MIENTRAS
```



```
REPETIR
      Bloque de Acciones
      Acción que modifica la condición
HASTA_QUE condición
```

Finalización de Bucles

- ¿Qué ocurre cuando en un programa un conjunto de acciones se repite sin control?
  - Bucles infinitos

- ¿Cuáles son los criterios para finalizar las iteraciones de un bucle?
  - Por *Valor Centinela*
  - Por *Bandera*
  - Por *Contador*

Finalización de Bucles (1)

- Los **bucles infinitos** no alcanzan la condición de finalización y por tanto se repiten indefinidamente.
- Se deben a errores de diseño: incorrecta formulación de la condición de finalización, omisión o errores en las instrucciones que modifican la condición de salida.

```
bandera<-VERDADERO
MIENTRAS bandera=VERDADERO HACER
        ESCRIBIR "BUCLE INFINITO"
FIN_MIENTRAS
```

```
contador<-1
MIENTRAS contador < 20 HACER
        ESCRIBIR "BUCLE INFINITO"
        contador<-contador-1
FIN_MIENTRAS
```

Finalización de Bucles (4)

- Finalización por **VALOR CENTINELA**

```
PROGRAMA producto_sumas
VARIABLES
      a,b:ENTERO // valor de entrada
      k:ENTERO // auxiliar
      prod:ENTERO // valor de salida
INICIO
     ESCRIBIR "Ingrese valores:"
     LEER a,b
     prod<-0
     k<-0
     REPETIR
       SI b <> 0 ENTONCES
          prod<-prod+a
          k<-k+1
        FIN_SI
     HASTA_QUE k=b
     ESCRIBIR "Producto: ", prod
FIN
```

Diseñe un algoritmo que calcule, mediante sumas sucesivas, el producto de 2 valores ingresados por el usuario. Utilice "finalización por centinela" para controlar el bucle de cálculo.

Finalización de Bucles (3)

- Finalización por **BANDERA**

```
PROGRAMA producto_sumas
VARIABLES
      a,b:ENTERO // valores de entrada
      contador:ENTERO // auxiliar
      bandera:LOGICO // auxiliar
      prod:ENTERO // valor de salida
INICIO
     ESCRIBIR "Ingrese valores:"
     LEER a,b
     prod<-0
     contador<-1
     bandera<-VERDADERO
     MIENTRAS bandera=VERDADERO HACER
         prod<-prod+a
         contador<-contador+1
         SI contador > b ENTONCES
            bandera<-FALSO
         FIN_SI
     FIN_MIENTRAS
     ESCRIBIR "Producto: ", prod
FIN
```

Diseñe un algoritmo que calcule, mediante sumas sucesivas, el producto de 2 valores ingresados por el usuario. Utilice "finalización por bandera" para controlar el bucle de cálculo.

Finalización de Bucles (2)

- Finalización por **CONTADOR**

```
PROGRAMA producto_sumas
VARIABLES
      a,b:ENTERO // valores de entrada
      contador:ENTERO // auxiliar
      prod:ENTERO // valor de salida
INICIO
     ESCRIBIR "Ingrese valores:"
     LEER a,b
     prod<-0
     contador<-0
     MIENTRAS contador < b HACER
         prod<-prod+a
         contador<-contador+1
     FIN_MIENTRAS
     ESCRIBIR "Producto: ", prod
FIN
```

Diseñe un algoritmo que calcule, mediante sumas sucesivas, el producto de 2 valores ingresados por el usuario. Utilice "finalización por contador" para controlar el bucle de cálculo.

Ejemplo Bucles (1)

- Diseñe un algoritmo que sume valores hasta que el usuario ingrese un valor par. Utilice el concepto de centinela para controlar el bucle.

```
PROGRAMA centinela
VARIABLES
   num,suma:REAL
INICIO
   suma<-0
   REPETIR
     ESCRIBIR "Ingrese numero"
     LEER num
     suma<-suma+num
   HASTA_QUE num div 2 = 0
   ESCRIBIR "La suma es ", suma
FIN
```

Ejemplo Bucles (2)

- Modifique el algoritmo anterior para utilizar el concepto de bandera para control del bucle.
- Modifique el algoritmo original de modo que el usuario indique cuántos valores serán sumados. Considere que sólo deberá sumar valores impares.

Anidamiento

- Consiste en combinar las estructuras de control básicas.
- Una estructura de control puede contener otra si es necesario (*anidamiento)*.
- Reglas para el anidamiento
  - la estructura interna debe quedar completamente incluida dentro de la externa, y
  - no puede existir solapamiento de estructuras.

Anidamiento Válido

- Ejemplos

```
SI condición_1 ENTONCES
    SI condición_2 ENTONCES
            acciones
    SINO
            acciones
    FIN_SI
SINO
    SI condición_3 ENTONCES
            acciones
    FIN_SI
FIN_SI
```

```
MIENTRAS condición_1 HACER
    PARA i DESDE vi HASTA vf HACER
            SI condición_2 ENTONCES
                    acciones_1
            SINO
                    acciones_2
            FIN_SI
            acciones_3
    FIN_PARA
FIN_MIENTRAS
```

Anidamiento Inválido

- Ejemplos

```
MIENTRAS condición_1 HACER
    SI condición_2 ENTONCES
        acciones
    FIN_MIENTRAS
FIN_SI
```

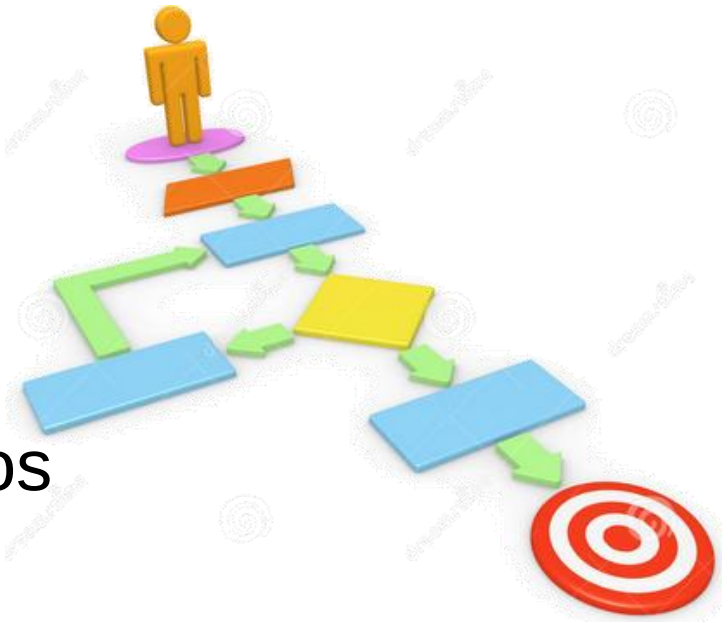
```
PARA i DESDE vi HASTA vf HACER
    MIENTRAS condición_1 HACER
        acciones
    FIN_PARA
FIN_MIENTRAS
```

```
REPETIR
    MIENTRAS condición_2 HACER
        SI condición_3 ENTONCES
            acciones
    FIN_MIENTRAS
        SINO
            SI condición_4 ENTONCES
                acciones
            FIN_SI
        FIN_SI
HASTA_QUE condición_1
```

```
SI condición_1 ENTONCES
    PARA i DESDE vi HASTA vf HACER
        acciones
    FIN_PARA
REPETIR
SINO
    acciones
FINS_SI
HASTA_QUE condición_2
```

Prueba de Escritorio (1)

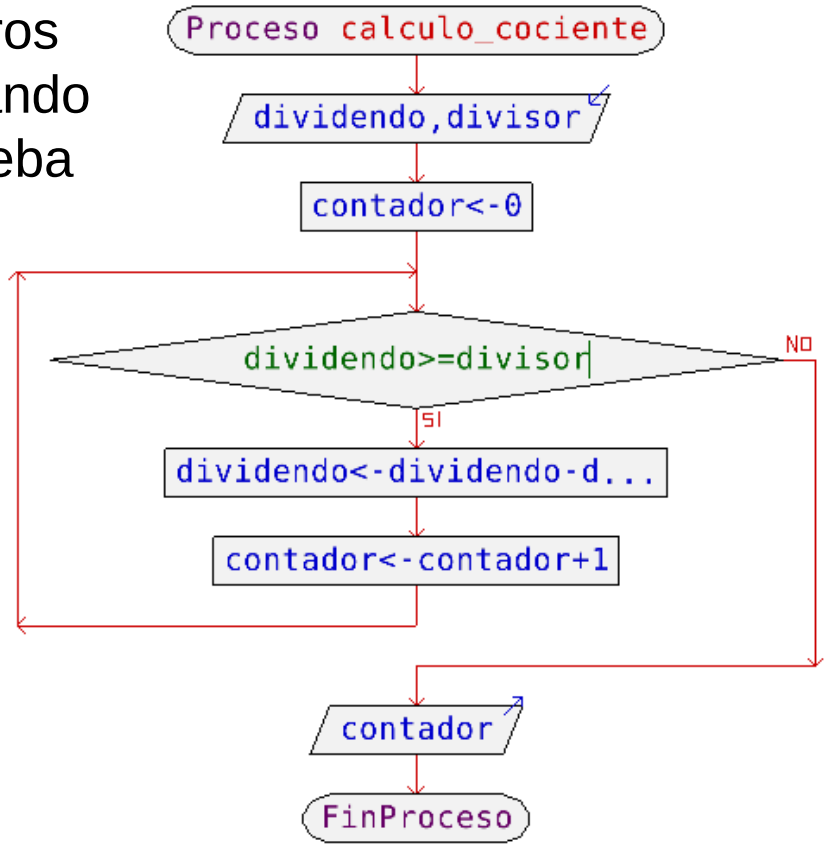
- Comprobación de un algoritmo en tiempo de diseño.
- Se analiza, paso a paso, el algoritmo y se indican los valores de las variables y condiciones.
- Se pueden probar tanto datos esperados como valores de excepción.



Prueba de Escritorio (2)

- Diseñe un algoritmo que calcule el cociente entero entre dos números ingresados por el usuario, aplicando restas sucesivas. Realice la prueba de escritorio para los valores: **dividendo=7 y divisor=2**

	VARIABLES			CONDICIONES
PASO	dividendo	divisor	contador	dividendo>=divisor
1	7			
2		2		
3			0	
4				VERDADERO
5	5			
6			1	
7				VERDADERO
8	3			
9			2	
10				VERDADERO
11	1			
12			3	
13				FALSO
RESULTADO: 3				



Resumen (1)

- Estructura **PARA**
  - Utiliza una variable de control de bucle (contador) que lleva cuenta del número de repeticiones.
  - Se aplica cuando se conoce el número de repeticiones a realizar.
  - El incremento de la variable de control puede ser configurado, por defecto, es 1.
  - Es una estructura pre-condicional

Resumen (2)

- Estructura **MIENTRAS**
  - Precondicional: la condición de repetición se evalúa antes de iniciar cada iteración del bucle.
  - Repite con condición VERDADERA, finaliza con condición FALSA.
  - Se aplica cuando NO se conoce el número de repeticiones a realizar.
  - Siempre debe incluir alguna instrucción que modifique la condición de repetición (finalización del bucle).

Resumen (3)

- Estructura **REPETIR**
  - Poscondicional: la condición de repetición se evalúa luego de ejecutar cada iteración del bucle.
  - Repite con condición FALSA, finaliza con condición VERDADERA.
  - Se aplica cuando NO se conoce el número de repeticiones a realizar.
  - Siempre debe incluir alguna instrucción que modifique la condición de repetición (finalización del bucle).

Bibliografía

- Sznajdleder, Pablo Augusto. Algoritmos a fondo. Alfaomega. 2012.
- López Román, Leobardo. Programación estructurada y orientada a objetos. Alfaomega. 2011.
- De Giusti, Armando *et al*. Algoritmos, datos y programas, conceptos básicos. Editorial Exacta, 1998.
- Joyanes Aguilar, Luis. Fundamentos de Programación. Mc Graw Hill. 1996.
- Joyanes Aguilar, Luis. Programación en Turbo Pascal. Mc Graw Hill. 1990.