FUNDAMENTOS DE PROGRAMACIÓN

TEORÍA Nº 4

UNIDAD 2: ESTRUCTURAS REPETITIVAS

Facultad de Ingeniería
Universidad Nacional de Jujuy

ÍNDICE

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

ESTRUCTURAS REPETITIVAS (1)

- Las soluciones basadas en 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 (2)

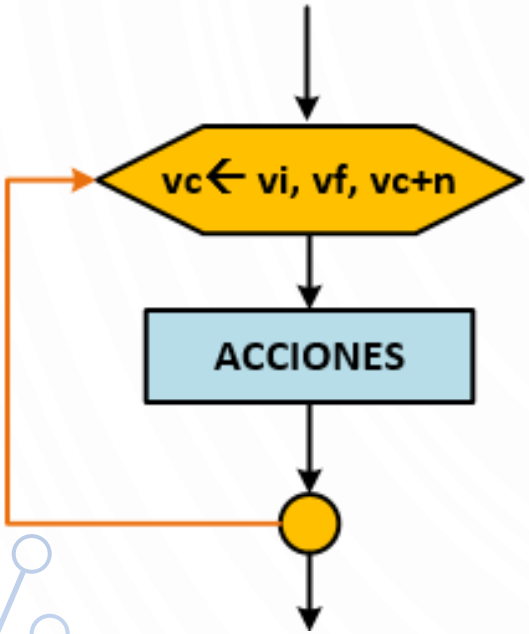
- Un **bucle** o *loop* es un conjunto de acciones que deben repetirse.
- El número de repeticiones (iteraciones) puede ser conocido a priori o no.
- En programación, las estructuras **For**, **While** y **Do While** permiten especificar el conjunto de acciones que deben ejecutarse en forma repetida.

ESTRUCTURA FOR(1)

- La estructura **For** 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/decremento** de la variable de control puede especificarse (por defecto es 1).

ESTRUCTURA FOR (2)

- For




```
for(vc=vi;vc<=vf;vc=vc+n)
   accion_simple;
```

```
for(vc=vi;vc<=vf;vc=vc+n)
{
   bloque_accion;
}
```

- ✓ **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

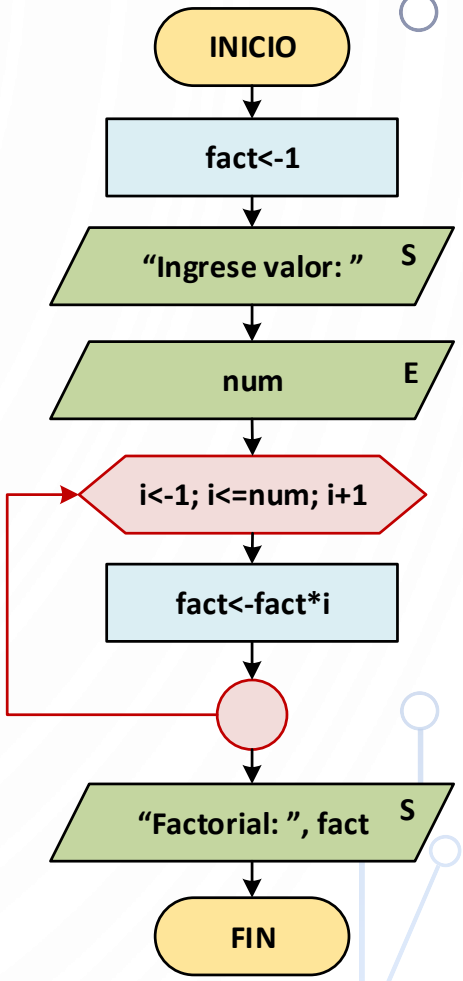
ESTRUCTURA FOR (3)

- Diseñe un algoritmo que calcule el factorial de un número ingresado por el usuario.

```
#include <iostream>
#include <stdlib.h>

using namespace std;

main()
{ int num,i, fact=1;
  cout << "Ingrese valor: ";
  cin >> num;
  for(i=1;i<=num;i++)
     fact=fact*i;
  cout << "Factorial: " << fact << endl;
  system("pause");
}
```



ESTRUCTURA WHILE (1)

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

ESTRUCTURA WHILE (2)

- While


```
while (condición)
  accion_simple;
```

```
while (condición)
{
  bloque_accion;
}
```

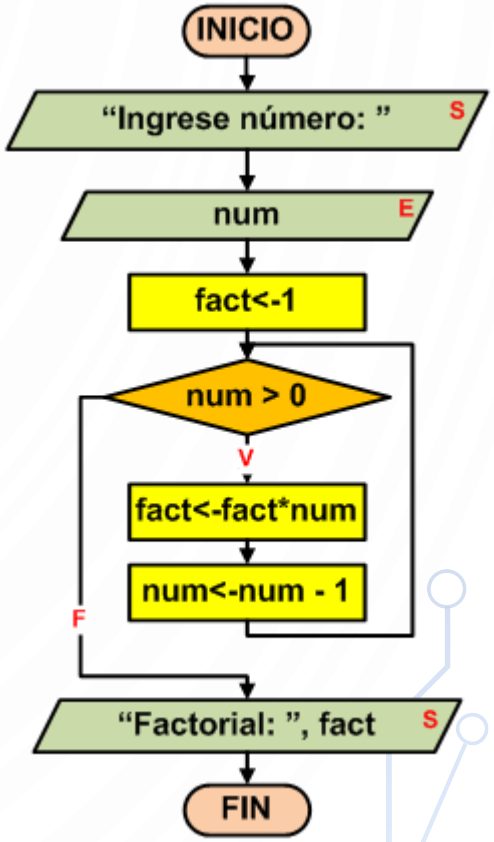
ESTRUCTURA WHILE (3)

- Diseñe un algoritmo que calcule el factorial de un número ingresado por el usuario.

```
#include <iostream>
#include <stdlib.h>

using namespace std;

main()
{ int num, fact=1;
  cout << "Ingrese numero: ";
  cin >> num;
  while (num>0)
   { fact=fact*num;
     num=num-1;
   }
  cout << "Factorial: " << fact << endl;
  system("pause");
}
```



ESTRUCTURA DO WHILE (1)

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

ESTRUCTURA DO WHILE (2)

- Do-While


```
do
{
    acciones
} while (condición);
```

true

false

ESTRUCTURA DO WHILE(3)

- Diseñe un algoritmo que calcula la suma de valores ingresados por el usuario hasta que se introduce un CERO.

```
#include <iostream>
#include <stdlib.h>

using namespace std;

main()
{  int num,suma=0;
   do
   {  cout << "Ingrese un valor: ";
      cin >> num;
      suma=suma+num;
   } while (num!=0);
   cout << "La suma de valores es: " << suma << endl;
   system("pause");
}
```

ESTRUCTURA DO WHILE (4)

- Modifique el algoritmo anterior de modo que utilice el concepto de bandera para finalizar el bucle.

```
#include <iostream>
#include <stdlib.h>
using namespace std;
main()
{ float num, suma=0;
  bool seguir;
  do
  { cout << "Ingrese valor: ";
    cin >> num;
    suma=suma+num;
    if (num==0)
      seguir=false;
    else
      seguir=true;
  } while(seguir==true);
  cout << "La suma es: " << suma << endl;
  system("pause");
}
```

- La variable lógica ***seguir*** permite detectar en qué momento se ingresa un dato cero.
- ***seguir*** es VERDADERA si el dato ingresado es distinto de cero.
- ***seguir*** es FALSA cuando el valor ingresado es igual a cero.
- el bucle finaliza cuando ***seguir*** es FALSA (num=0), ya que la condición de repetición no se cumple.

FINALIZACIÓN DE BUCLES (1)

- ¿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*

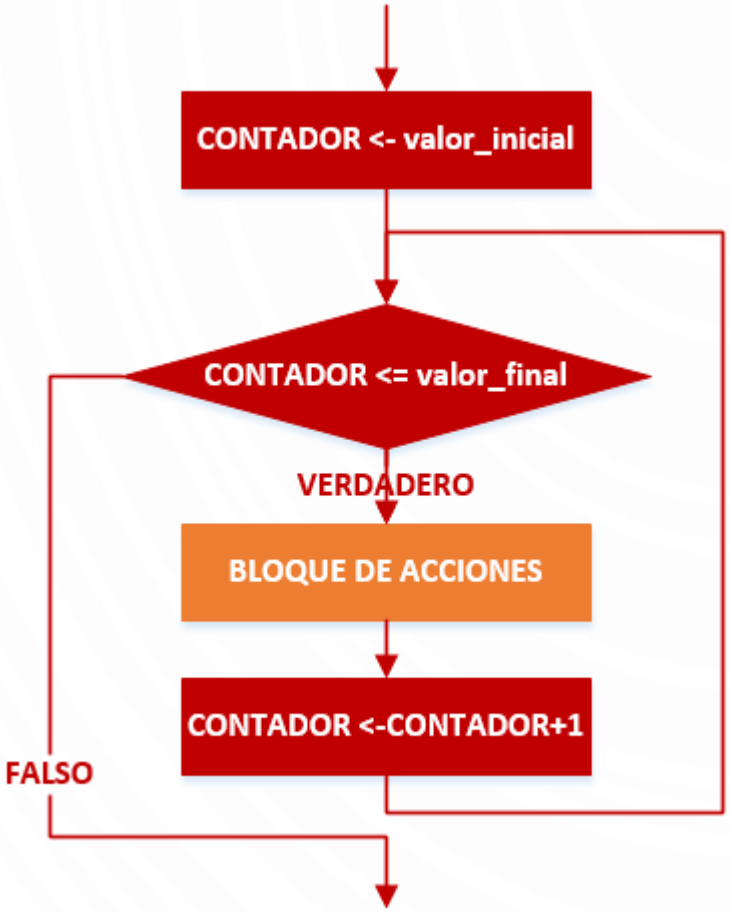
FINALIZACIÓN DE BUCLES (2)

- 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
```

WHILE - MIENTRAS: CONTROLADO POR CONTADOR (1)



- **Inicialización**: se asigna el valor inicial al contador
- **Condición de repetición**: se verifica que el contador se encuentre entre el valor_inicial y valor_final
  - Cond. Verdadera repite
  - Cond. Falsa finaliza
- **Modificación del contador**: se incrementa o decrementa el valor del contador.

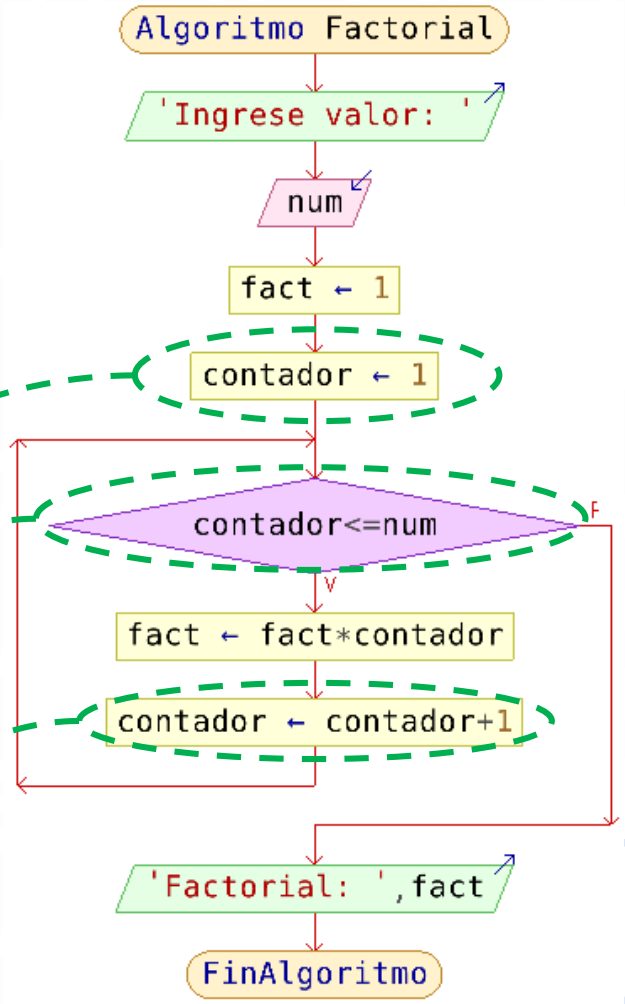
WHILE - MIENTRAS: CONTROLADO POR CONTADOR (2)

- Diseñe un algoritmo que calcule el factorial de un número ingresado por el usuario. Implemente el bucle de cálculo con estructuras MIENTRAS y utilice la finalización por contador.
- Inicialización:

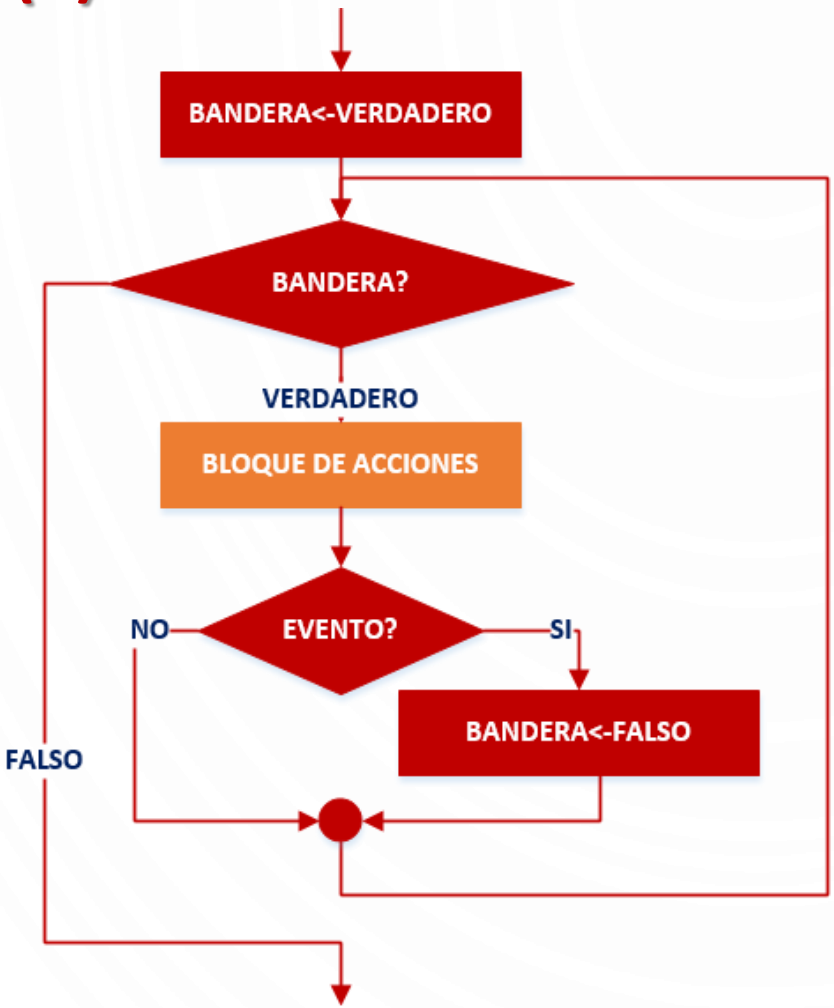
  contador ← 1
- Condición de repetición:

  contador <= num
- Modificación:

  contador ← contador + 1



WHILE - MIENTRAS: CONTROLADO POR BANDERA (1)



- **Inicialización**: se asigna el valor inicial a la bandera
- **Condición de repetición**: se analiza el valor de la bandera
  - Cond. Verdadera repite
  - Cond. Falsa finaliza
- **Detección de un evento**: se verifica si determinado evento ocurrió o no en el programa. La ocurrencia del evento implica modificar la bandera.

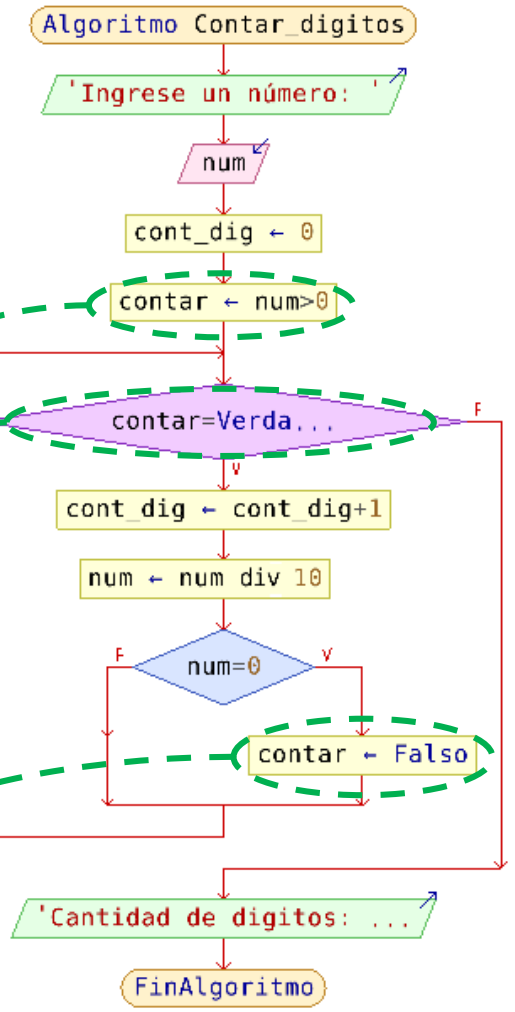
WHILE - MIENTRAS: CONTROLADO POR BANDERA (2)

- Diseñe un algoritmo que calcule cuántos dígitos tiene un número ingresado por el usuario. Implemente el bucle de cálculo con estructuras MIENTRAS y utilice la finalización por bandera.
- Inicialización:

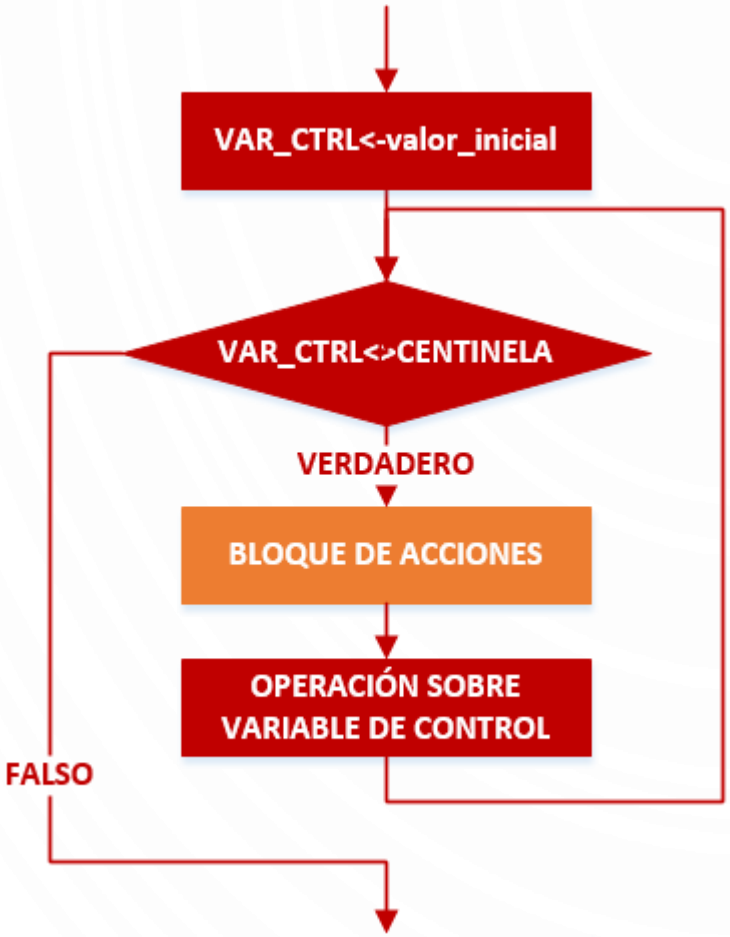
  contar ← num > 0
- Condición de repetición:

  contar = Verdadero
- Modificación:

  contar ← Falso



WHILE-MIENTRAS: CONTROLADO POR CENTINELA (1)



- **Inicialización**: se asigna el valor inicial a la variable de control del bucle
- **Condición de repetición**: se verifica que la variable de control sea distinta al valor centinela.
  - Cond. Verdadera repite
  - Cond. Falsa finaliza
- **Modificación de la variable de control**: se realiza alguna operación que modifica la variable de control.

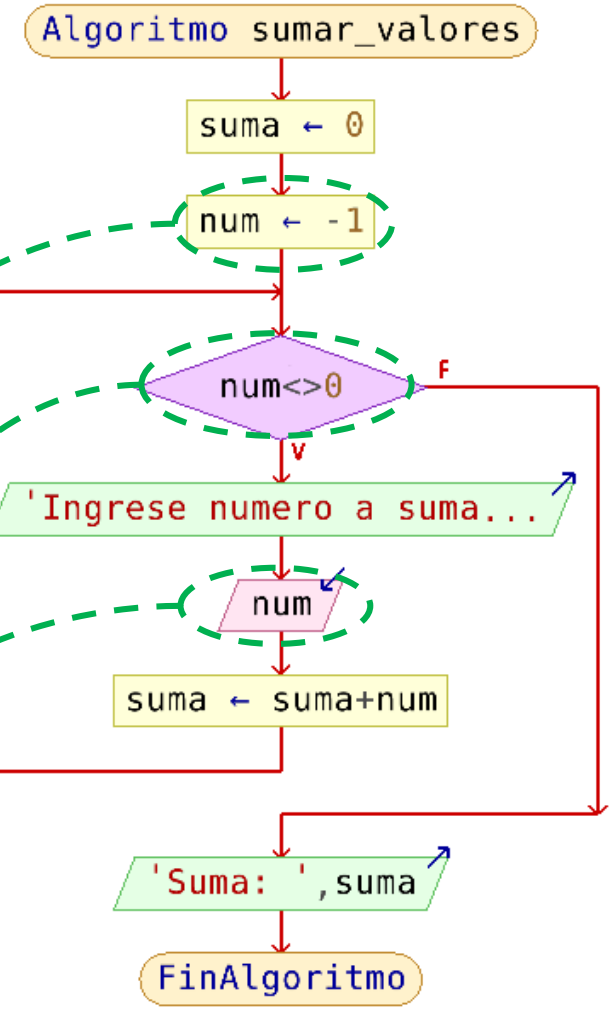
WHILE-MIENTRAS: CONTROLADO POR CENTINELA (2)

- Diseñe un algoritmo que sume valores ingresados por el usuario, hasta que se introduzca un 0. Implemente el bucle de cálculo con estructuras MIENTRAS y utilice la finalización por centinela.
- Inicialización:

  num ← -1 (arbitrario)
- Condición de repetición:

  num <> 0
- Modificación:

  LEER num



RESUMEN (1)

- Estructura **FOR**
  - 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 **WHILE**
  - Pre-condicional: 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 **DO-WHILE**
  - Pos-condicional: la condición de repetición se evalúa luego de ejecutar 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).

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.

EJERCICIOS A RESOLVER

1. 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
2. Diseñe un algoritmo que sume 50 valores ingresados por el usuario.
3. Diseñe un algoritmo que calcule la potencia $\mathbf{a^b}$ mediante productos sucesivos.
4. Diseñe un algoritmo que muestre ***n*** **veces** un mensaje, siendo ***n y el mensaje*** especificados por el usuario.
5. Diseñe un algoritmo que calcule el máximo común divisor de 2 números ingresados por el usuario.
6. Diseñe un algoritmo que cuente valores ingresados por el usuario hasta que se presente **un CERO**.
7. Diseñe un algoritmo que determine el mínimo de una serie de datos ingresados por el usuario. Finaliza con 0.
8. Diseñe un algoritmo que sume valores hasta que el usuario ingrese un valor par. Utilice el concepto de centinela para controlar el bucle.
9. Modifique el algoritmo anterior para utilizar el concepto de bandera para control del bucle.