PROGRAMACIÓN I

Unidad 3

Composición Secuencial de Acciones. Estados iniciales y finales. Estados intermedios y Refinamiento sucesivos. La estructura de decisión. Estructura de decisión múltiple. Tipos de estructura de Control: Sentencia si-sino. Sentencia Switch. Sentencias anidadas. Principio de Inducción. La estructura de Iteración. Tipos de estructura de iteración: Sentencia While. Sentencia For. Sentencia Do-While. Programación Esquemática.

Sentencias de Control

Las sentencias de control permiten controlar el flujo del programa, tomando decisiones a partir de comparaciones.

- **Se usan instrucciones condicionales y de ciclos.**
- **Un condicional es un conjunto de sentencias que pueden o no ejecutarse, dependiendo del resultado de una condición.**
- **Un ciclo es un conjunto de sentencias que son ejecutadas varias veces, hasta que una condición de término es satisfecha.**
- **Tanto los condicionales como los ciclos contienen a otras sentencias. Para indicar esta relación ,las sentencias contenidas no se escriben en la misma columna que la sentencia de control, sino un poco más a la derecha**

Sentencias de Control

- Las instrucciones **condicionales** son:
  - si -SI
  - SWITCH o SEGÚN
- Las instrucciones de **ciclo** son:
  - WHILE - MIENTRAS
  - FOR - PARA
  - REPEAT - REPETIR

La ESTRUCTURA REPETITIVA se utiliza cuando es necesario que un conjunto de instrucciones se ejecuten un cierto número finito de veces .

Existen dos tipos de implementar las estructuras repetitivas;

- **la primera es aquella en donde se tiene perfectamente establecido el número de veces que un grupo de acciones se van a ejecutar (20, 5, 2 veces). En este caso se utiliza un contador. En este caso el contador se puede incrementar como decrementar.**

- **y la segunda en la que el número de repeticiones es desconocido y se hará hasta que se cumpla o no cierta condición. Por ejemplo que un número sea mayor que cero.**

- **La condición puede ser simple o estar combinada mediante operadores lógicos.**

- **Las estructuras que repiten una secuencia de instrucciones un número determinado de veces se denominan bucles**

- **Se denomina iteración al hecho de repetir la ejecución de una secuencia de acciones.**

- **Las dos principales preguntas a realizarse en el diseño de un bucle son :**
    - **¿qué contiene el bucle? Y**
    - **¿cuántas veces se debe repetir?**

- **Un bucle que nunca se termina se denomina bucle infinito o sin fin. Los bucles sin fin no intencionados son perjudiciales para la programación y se deben evitar siempre.**

- *Contador:* Es una variable que se incrementa, cuando se ejecuta, en una unidad o en una cantidad constante.

  Ejemplo: **contador := contador + 1**

  multiplo := multiplo + 3

- *Acumulador:* Es una variable que se incrementa en una cantidad variable.

  Ejemplo: **suma := suma + numero**

SENTENCIA WHILE O MIENTRAS

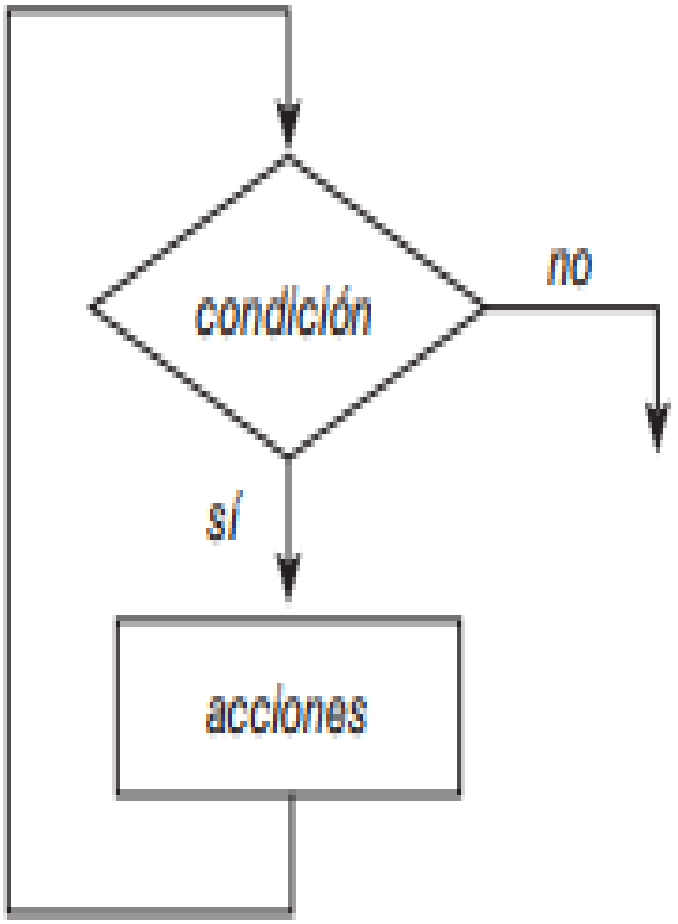
La estructura repetitiva **MIENTRAS** es aquella en que el cuerpo del bucle se repite mientras se cumple una determinada condición.

Cuando se ejecuta la instrucción **mientras**, lo primero que sucede es que se evalúa la condición.

Si la condición es *falsa*, no se toma ninguna acción y el programa prosigue en la siguiente instrucción del bucle.

Si la condición es *verdadera*, entonces se ejecuta el cuerpo del bucle, después de lo cual se evalúa de nuevo la condición. Este proceso se repite una y otra vez **mientras** la condición sea verdadera.

MIENTRAS O WHILE primero evalúa la condición y si se cumple entra en el ciclo hasta que la condición no se cumpla.




```
mientras (condición) hacer
    sentencia1;
    sentencia2;
    sentencia1;
        .
        .
        .
        .
    sentencia n;

fin mientras
```

```
Accion Contador es

Ambiente
   cont,acum,a: entero;

   cont=0; ( inicializa contador)

   acum=0; (inicializa acumulador)

Algoritmo
 mientras (cont <3) hacer
      escribir (“Ingrese un valor”);
      leer (a);
      escribir (“ Numero ingresado“,a);
      cont=cont+1;      (incrementa      contador)
      acum=acum+a; (actualiza el acumulador)

   fin mientras

escribir (“ El valor acumulado es :”,acum);

fin acción
```

En este problema se tiene perfectamente establecido el número de veces que un grupo de acciones se van a ejecutar. En este caso tres veces.

La condición será evaluada antes de cada iteración

El cuerpo del bucle mientras se ejecuta hasta que la condición sea falsa.

El contador y el acumulador se actualizan en cada iteración

EJERCICIO:

Realizar un programa que lea una serie de números reales y los sume. El programa debe preguntar al usuario cuando 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.

```
Acción bandera es
Ambiente
bandera: entero;
 dato,suma: entero;
 c: caracter ;

algoritmo
 bandera := 1;
 suma := 0 ;

mientras (bandera = 1) hacer
   escribir (“Introduzca un dato:”) ;
   leer (dato);
   suma := suma + dato ;

   escribir (“Desea continuar ingresando datos (S/N):”) ;
   leer (c) ;
   si (c = ‘N’ OR c = ‘n’) entonces bandera = 0 ;
   fin_si
 fin_mientras

escribir( “La suma es:”, suma)
```

EJERCICIO:

¿Qué hace la siguiente acción estructurada?

```
Acción quehace es
Ambiente
  a,b,c: entero;
Algoritmo
    escribir(“ Introduce un número: ”);
    leer(a);
     b=1;
    mientras (b <= 10) hacer
          c=a*b;
          escribir (c);
          b=b+1;
fin mientras

fin acción
```

Para responder a la pregunta realice la prueba de escritorio

Comprobación o Prueba de Escritorio

a	b	c
4	1	4
4	2	8
4	3	12
4	4	16
4	5	20
4	6	24
..	..	..
4	10	40

EJERCICIO:

Calcular el perímetro de un cuadrado solamente cuando el valor ingresado sea correcto. Usar sentencia MIENTRAS para realizar el control del valor ingresado.

```
Acción perímetro es
Ambiente
  p,a: entero;
  p:=0;
  a:=0;
Algortimo
        mientras (a <= 0) hacer
                escribir ("Ingrese un valor positivo");
                leer(a);
        fin mientras

        p:= a*4;
        escribir("Perimetro=",p);

Fin acción
```

¿Qué problema presenta la sentencia MIENTRAS?

```
Acción perímetro es

Ambiente
  p,a,b,c: entero;
  p:=0;
  c:=0;

Algoritmo

escribir ("Ingrese la cantidad de veces que verificará el valor del lado\n");
leer (b);

        mientras (a <= 0 ∧ c < b ) hacer
                escribir ("Ingrese un valor positivo");
                leer(a);
                c:=c+1;
        fin mientras

        si (c= b) entonces escribir (“Superó la cantidad de intentos permitidos”);
          sino
                p:= a*4;
                escribir("Perimetro=",p);
       fin si
Fin acción
```

EJERCICIO:

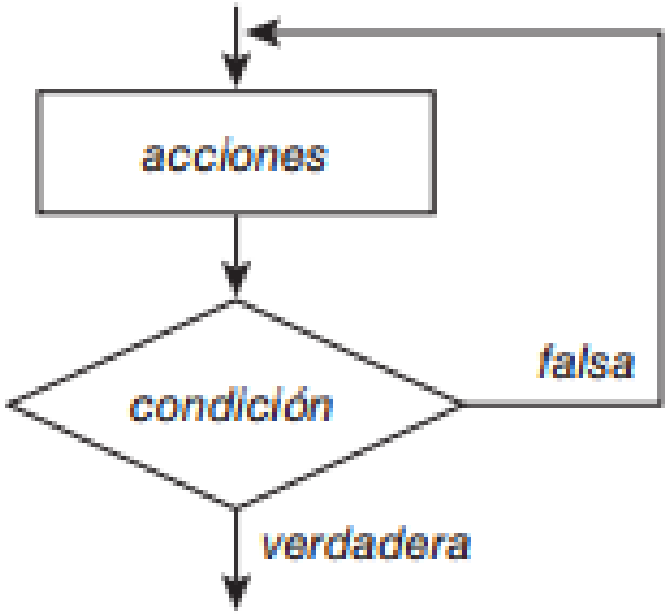
Realizar un programa que permita contar la cantidad de números pares ingresados por teclado.

```
Acción perímetro es

Ambiente
un,num,d,c: entero;
d:= 0;
c:=0;

Algoritmo
        escribir("Ingrese cantidad de numeros a ingresar");
        leer (nu);

        mientras (c<=nu) hacer
                escribir("Ingrese un numero");
                leer (num);
                si (num MOD 2 =0) entonces
                        escribir ("Numero",num);
                        d=d+1;
                fin si
                c:=c+1;
        fin mientras

        si (d=0) escribir ("No se ingresaron números pares");
            sino escribir("Números pares ingresdos:",d);
fin acción
```

SENTENCIA DO-WHILE O REPETIR

En un bucle Repetir, primero se ejecuta el bloque de instrucciones y, después, se evalúa la condición (<*expresión_lógica*>). En el caso de que esta sea falsa, se vuelve a ejecutar el bloque de instrucciones. Y así sucesivamente, hasta que, la condición sea verdadera.

Diagrama de flujo




```
repetir
    sentencia1;
    sentencia2;
    sentencia1;
        .
        .
        .
        .
    sentencia n;

hasta (condición)
```

```
Acción contar es
Ambiente
num : entero;
num:=1;

Algoritmo
        repetir
                escribir(num);
                num:=mun+1;
          Hasta (num>10);

    Escribir ("condicion de salida:",num);

Fin acción
```

La condición será evaluada después de cada iteración

Calcular el perímetro de un cuadrado. Verificar que el lado ingresado sea correcto.

```
Acción perímetro es

Ambiente
  p,a,b,c: entero;
  p:=0;
  c:=0;

Algoritmo

  repetir
       {
          escribir (“Ingrese un valor positivo”);
          leer (a);
       }
hasta (a >0);

p= a*4;
escribir (“Perimetro=”p);

Fin acción
```

En este problema el número de repeticiones es desconocido y se hará hasta que se cumpla o no cierta condición. En este caso se repite el bucle hasta que se ingrese un valor de lado positivo.

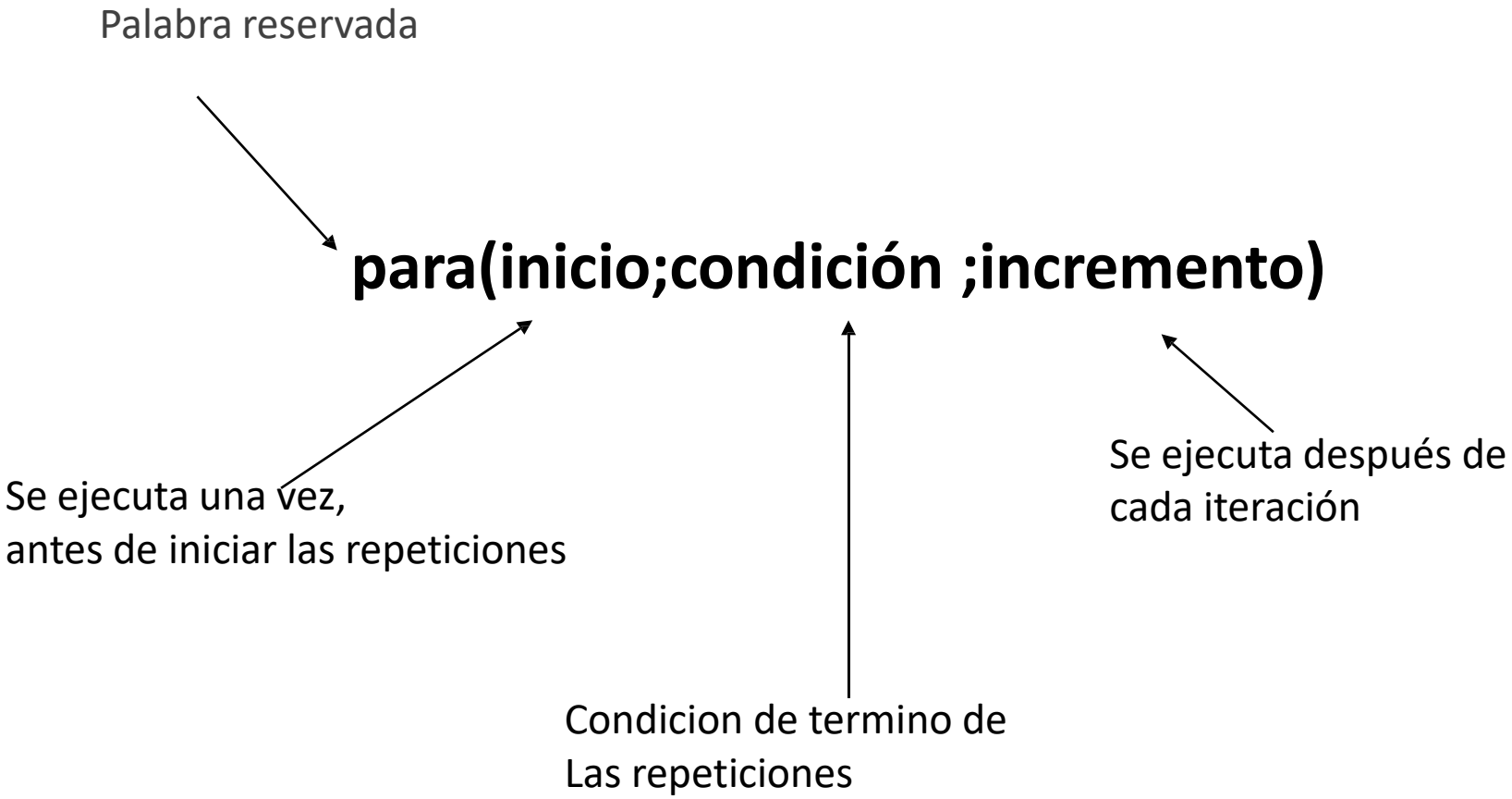
```
Acción perímetro es

Ambiente
  p,a,b,c: entero;
  p:=0;
  c:=0;

Algoritmo

  repetir
       {
          escribir (“Ingrese un valor positivo”);
          leer (a);
       }
hasta (a <=0);

p= a*4;
escribir (“Perimetro=”p);

fin acción
```

¿Qué problema presenta la sentencia REPETIR?

```
Acción perímetro es

Ambiente
  p,a,b,c: entero;
  p:=0;
  c:=0;

Algoritmo

escribir ("Ingrese la cantidad de veces que verificará el valor del lado\n");
leer (b);

        repetir
                escribir ("Ingrese un valor positivo");
                leer(a);
                c:=c+1;
        hasta (a > 0  ∧  c > b )

        si (c= b) entonces escribir (“Superó la cantidad de intentos permitidos”);
          sino
                p:= a*4;
                escribir("Perimetro=",p);
        fin si
Fin acción
```

SENTENCIA FOR O PARA

- En muchas ocasiones, se conoce de antemano el número de veces que se desean ejecutar las acciones de un bucle, en estos casos número de iteraciones es fija, se debe usar la estructura **desde** o **para** (en inglés **FOR**). Esta estructura **desde**, ejecuta las acciones del cuerpo del bucle un número específico de veces, y de forma automática controla el número de iteraciones o pasos a través del cuerpo del bucle.


Palabra reservada
para(inicio;condición ;incremento)
Se ejecuta una vez,
antes de iniciar las repeticiones
Condicion de termino de
Las repeticiones
Se ejecuta después de
cada iteración

Ejemplo: Ingresar tres números y mostrarlos por pantalla usando PARA

```
Acción perímetro es
Ambiente
Cont,a : entero;
  cont:=0;

algoritmo
para (cont = 0; cont<=2;cont:=cont+1)
     escribir ("Ingrese un valor");
     leer (a);
     escribir (" Numero ingresado:",a);
fin para
fin acción
```

En este problema se tiene perfectamente establecido el número de veces que un grupo de acciones se van a ejecutar. En este caso tres veces.

```
Cont=0;
mientras cont<2
    escribir ("Ingrese un valor");
    leer (a);
    escribir (" Numero ingresado:",a);
    cont:=cont+1;
fin mientras
```

Comparación de Bucles

WHILE	- Adecuada para búsquedas - El cuerpo puede no ser ejecutado - La verificación de la condición precede a la ejecución del cuerpo
FOR	- Adecuado para recorridos - La verificación de la condición precede a la ejecución del cuerpo
REPETIR	- Adecuado en el caso de que debamos garantizar que el cuerpo del bucle se ejecuta por lo menos en una ocasión.

Estructuras repetitivas anidadas

- De igual forma que se pueden anidar o encajar estructuras de selección, es posible insertar un bucle dentro de otro.

- Las reglas para construir estructuras repetitivas anidadas son iguales en ambos casos: la estructura interna debe estar incluida totalmente dentro de la externa y no puede existir solapamiento.

- Las variables índices o de control de los bucles toman valores de modo tal que por cada valor de la variable índice del ciclo externo se debe ejecutar totalmente el bucle interno.

```
Acción multiplicación es es
Ambiente
  a,b,c,d,cont: entero;
Algoritmo
escribir (“Ingrese la cantidad de tablas de multiplicar a calcular”);
leer (d);
cont:=0;
mientras (cont<d) hacer
      escribir(“ Ingresar tabla a calcular”);
      leer(a);
       b=1;
      mientras (b <= 10) hacer
              c:=a*b;
              escribir (c);
              b:=b+1;
        fin mientras
     cont:=cont+1;
fin mientras
fin acción
```