



Estructuras Repetitivas

Introducción a la Programación
Facultad de Informática
Univ.Nac. del Comahue



Estructuras de Control

Controlan la secuencia o flujo de ejecución de las instrucciones de un programa

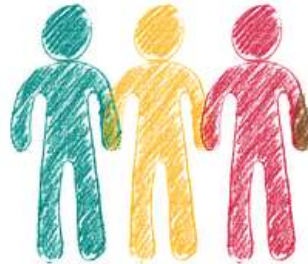
- Podemos dividir las en tres categorías:
 - Secuencial
 - Alternativa o de Condición
 - Iterativa o Repetitiva ← **HOY**



Estructuras de Control Repetitiva

Motivación

Problema: Solicitar las notas de 3 alumnos y obtener el promedio de las notas



La **estructura de control secuencial** es suficiente para resolver el **problema**: 3 variables para almacenar las notas, luego calcular la suma y por ultimo obtener el promedio

Ejemplo: Si las notas son 7.5 , 9.5 y 8, entonces para obtener el promedio debemos Sumar las 3 notas: $7.5 + 9.5 + 8 = 25$ y luego dividir la suma por la cantidad de notas: $25 / 3 = 8.333$



Estructuras de Control Repetitiva

Motivación

Problema: Solicitar las notas de 3 alumnos y obtener el promedio de las notas



PROGRAMA PRINCIPAL

FLOAT Nota1, Nota2, Nota3,
FLOAT Suma, Promedio

Traza Programa Principal

Nota1	Nota2	Nota3	Suma	Promedio	Pantalla
7.5	9.5	8.0	0	8.333	El promedio es: 8.333
			7.5		
			17		
			25		


➡ Suma \leftarrow 0

➡ LEER(Nota1) 

➡ Suma \leftarrow Suma + Nota1

➡ LEER(Nota2) 

➡ Suma \leftarrow Suma + Nota2

➡ LEER(Nota3) 

➡ Suma \leftarrow Suma + Nota3

➡ Promedio \leftarrow Suma / 3

➡ ESCRIBIR("El promedio es: ", promedio)

FIN PROGRAMA

Por simplicidad quitamos las instrucciones ESCRIBIR para pedir las notas



Estructuras de Control Repetitiva

Motivación

Problema: Solicitar las notas de 60 alumnos y obtener el promedio de las notas



¿hay que crear 60 variables? 🤖

¿Y si quisieramos utilizar el mismo programa para calcular el promedio de 100 alumnos?

Para este tipo de problemas vamos a recurrir a la estructura de control repetitiva.



Estructuras de Control Repetitiva

Motivación

Problema: Solicitar las notas de 60 alumnos y obtener el promedio de las notas



PROGRAMA PRINCIPAL

FLOAT Nota1, Nota2, Nota3,
FLOAT Suma, Promedio

Suma \leftarrow 0

LEER(Nota1)

Suma \leftarrow Suma + Nota1

LEER(Nota2)

Suma \leftarrow Suma + Nota2

LEER(Nota3)

Suma \leftarrow Suma + Nota3

Promedio \leftarrow Suma / 3

ESCRIBIR("El promedio es: ", promedio)

FIN PROGRAMA

PROGRAMA PRINCIPAL

FLOAT Nota

FLOAT Suma, Promedio

Suma \leftarrow 0

LEER(Nota)

Suma \leftarrow Suma + Nota

*Repetir
60 veces*

Promedio \leftarrow Suma / 60

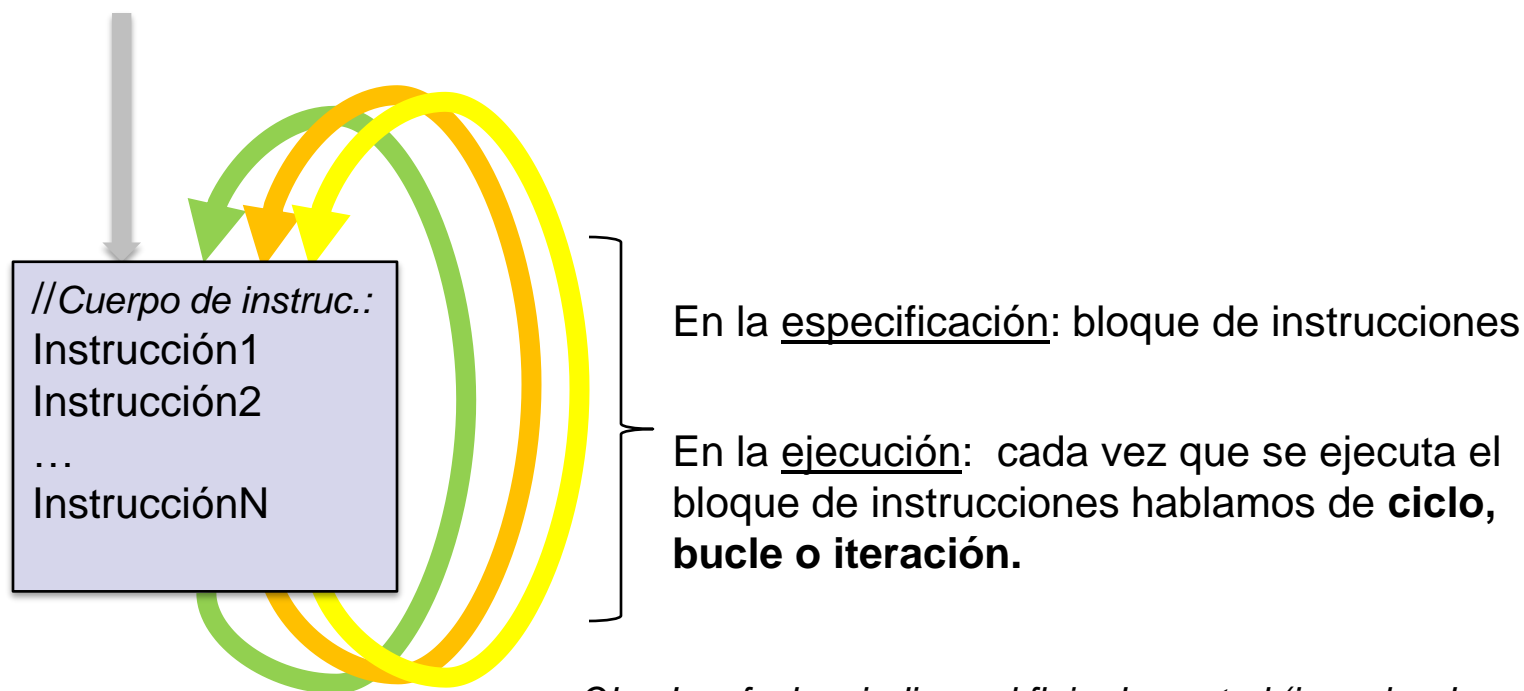
ESCRIBIR("El promedio es: ", promedio)

FIN PROGRAMA



Estructuras Repetitivas

- **Instrucciones** que permiten **repetir la ejecución de un bloque de instrucciones** tantas veces como sea necesario.
- Cada ejecución del bloque de instrucciones se conoce como **ciclo, bucle o iteración**.



Obs. Las flechas indican el flujo de control (i.e., el orden de ejecución)

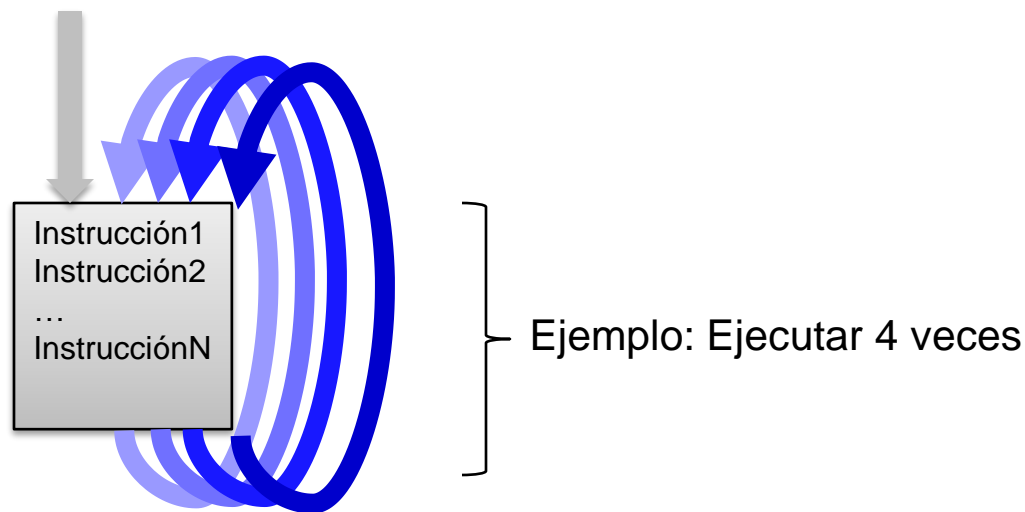


Estructuras Repetitivas

El número de veces que se ejecutará el bloque puede definirse:

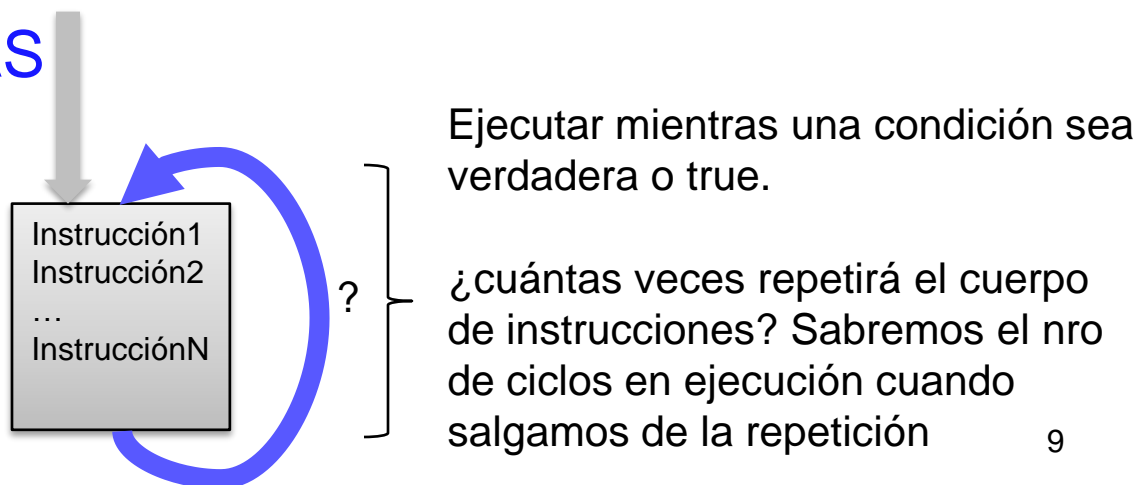
a) Explícitamente:

Instrucción Repetitiva **PARA**
(ciclo definido)



b) mediante una expresión lógica o condición que indica si el bloque debe ejecutarse nuevamente:

Instrucción Repetitiva **MIENTRAS**
(ciclo indefinido)





INSTRUCCIONES DE LA ESTRUCTURA DE CONTROL REPETITIVA



Instrucción Mientras (while)

- Permite ejecutar un bloque de instrucciones **mientras** que una **expresión lógica o condición sea verdadera**.

Sintáxis Pseudocódigo:

MIENTRAS (condición) **HACER**

Instrucción1

Instrucción2

....

instrucciónN

FIN MIENTRAS

Sintáxis PHP:

```
while(condicion) {  
    instruccion1;  
    instruccion2;  
    ...  
    instruccion N;  
}
```

*Bloque de
instrucciones
de la
estructura
repetitiva*



Instrucción Mientras (While)

¿Cómo ejecuta?

1° - Resuelvo y evaluó la condición.

. Si la condición es Verdadera ejecuto el bloque de instrucciones (instrucción1 a instrucciónN) y vuelvo a 1°- resuevlo y evalúo la condición

. Si la condición es Falsa: Fin Mientras. Termina la instrucción repetitiva

➡ **MIENTRAS** (condición) **HACER**

➡ Instrucción1

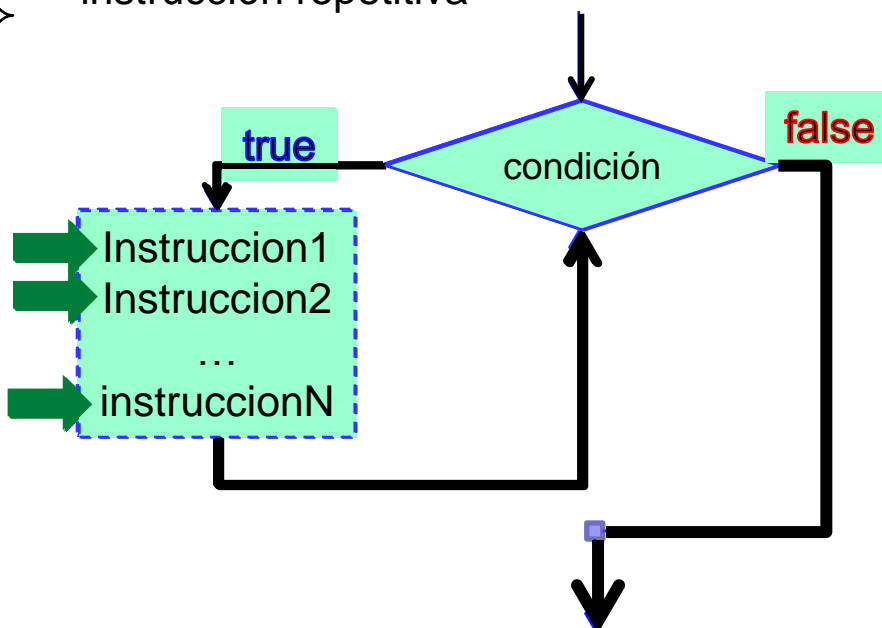
➡ Instrucción2

....

➡ instrucciónN

➡ **FIN MIENTRAS**

```
while(condicion) {  
    instruccion1;  
    instruccion2;  
    ...  
    instruccion N;  
}
```



CICLOS: 3



Instrucción Mientras (while)

CONSIDERACIONES

- Al comenzar la ejecución de la instrucción MIENTRAS puede pasar que la condición se evalúe como falsa, por lo tanto el bloque de instrucciones nunca será ejecutado, es decir, la instrucción MIENTRAS producirá CERO ciclos. Esto puede ser correcto según el problema que estemos resolviendo.
- Si al ejecutar la instrucción repetitiva, resulta que la condición siempre es verdadera, el bloque de instrucciones se ejecutará infinitas veces. Esto es **incorrecto** por que no cumplimos con la definición de algoritmo: “Es una secuencia finita de pasos”.



Ojo con Ciclos infinitos!

Asegurarse que alguna de las variables de la condición sea modificada en el ciclo y produzca que la condición sea falsa

Pseudocódigo:

```
MIENTRAS (condición) HACER
    instruccion1
    instruccion2
    ...
    instruccionN
FIN MIENTRAS
```

PHP:

```
while(condicion) {
    instruccion1;
    instruccion2;
    ...
    instruccion N;
}
```

*BLOQUE
De
Instrucciones*



Instrucción While

**Ejercicio 1 TP6: Escribir en
pantalla los números
Naturales pares menores
iguales a N**



Instrucción Repetir...Mientras (do...while)

- Similar al bucle Mientras, con la diferencia que el bloque de instrucciones se ejecuta **al menos una vez**. La expresión lógica se evalúa luego de ejecutar el bloque de instrucciones.
- Si la expresión lógica se evalúa verdadera se realiza una nueva iteración. Si es falsa, el bloque de instrucciones no se vuelve a ejecutar

Sintáxis Pseudocódigo:

REPETIR

Instrucción1
Instrucción2
....
instrucciónN

MIENTRAS (condicion)

Sintáxis PHP:

```
do {  
    instruccion1;  
    instruccion2;  
    ...  
    instruccion N;  
} while(condicion);
```

Bloque de instrucciones de la estructura repetitiva



Instrucción Repetir...Mientras (do...while)

¿Cómo ejecuta?

1° - Ejecuto el bloque de instrucciones (instrucción1 a instrucciónN).

2° - Resuelvo y evalúo la condición

. Si la condición es Verdadera ejecuto el bloque de instrucciones (instrucción1 a instrucciónN) y vuelvo a 2°- resuelvo y valúo la condición

. Si la condición es Falsa: **Fin**. Termina la instrucción repetitiva

→ **REPETIR**

→ Instrucción1

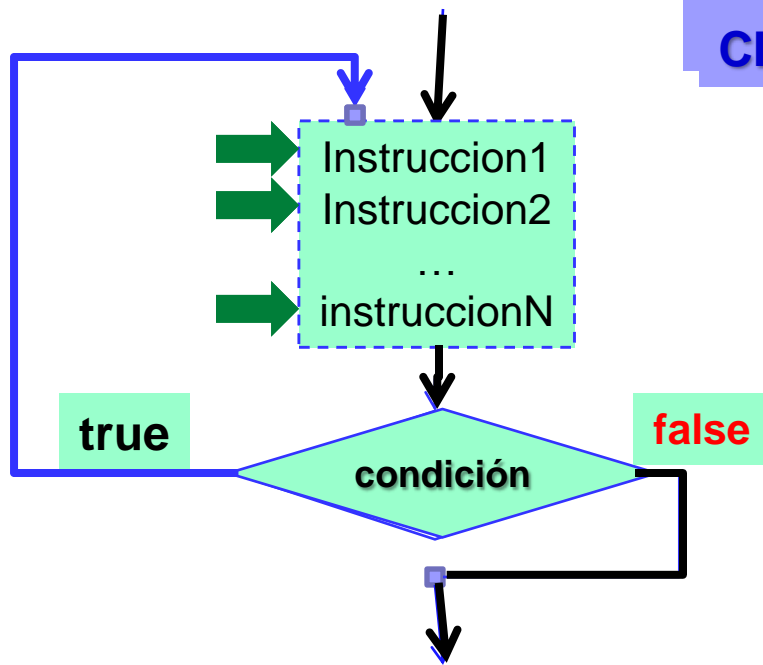
→ Instrucción2

....

→ instrucciónN

→ **MIENTRAS**(condicion)

```
do {  
    instruccion1;  
    instruccion2;  
    ...  
    instruccion N;  
} while(condicion);
```





Instrucción Repetir..Mientras (do..while)

CONSIDERACIONES

- Destaquemos nuevamente que esta instrucción siempre ejecutará al menos una vez el bloque de instrucciones, es decir, por lo menos tendrá 1 CICLO.
- Al igual que la instrucción MIENTRAS, si al ejecutar la instrucción repetitiva, resulta que la condición siempre es verdadera, el bloque de instrucciones se ejecutará infinitas veces.



Ojo con Ciclos infinitos!

Asegurarse que alguna de las variables de la condición sea modificada en el ciclo y produzca que la condición sea falsa

Pseudocódigo:

REPETIR

Instrucción1
Instrucción2
....
instrucciónN

MIENTRAS (condicion)

PHP:

```
do {  
    instruccion1;  
    instruccion2;  
    ...  
    instruccion N;  
} while(condicion);
```

*BLOQUE
De
Instrucciones*



Instrucción Repetir...Mientras (do...while)

Ejercicio 2 TP6:

Escribir un Programa Principal que le solicite a un usuario números hasta que ingrese un Cero. El resultado es la suma de todos los números leídos.



Instrucción Para (for)

- Ejecuta un bloque de instrucciones un número determinado de veces.
- Es el más apropiado cuando conocemos la cantidad de ciclos(N)

Pseudocódigo:

```
PARA  $i \leftarrow valorInicial$  HASTA  $valorUmbral$  PASO  $1$  HACER  
    Instrucción1  
    Instrucción2  
    ....  
    instrucciónN  
FIN PARA
```

Decimos que la variable entera i es un CONTADOR.

PHP:

```
for( $(\$i = valorInicial;$   $\$i < valorUmbral;$   $\$i=\$i+1)$ ) {  
    instruccion1;  
    instruccion2;  
    ...  
    instruccionN;  
}
```

Sólo cuando el paso es 1
En lugar de $\$i = \$i + 1$
Podemos especificar
 $\$i++$



Instrucción Para (for)

Pseudocódigo:

```
ESCRIBIR("ingrese un nro:")
LEER(n)
PARA  $i \leftarrow 0$  HASTA  $n$  PASO 1 HACER
    instruccion1
    instruccion2
    ...
    instruccionN
FIN PARA
```

PHP:

```
echo "ingrese un nro:";
$n = trim(fgets(STDIN));
for ($i = 0; $i < $n; $i++) {
    instruccion1;
    instruccion2;
    ...
    instruccionN;
}
```

- Recordemos que la variable entera i es un contador.
- En este caso, el valor inicial fue reemplazado por el valor cero, y el umbral fue reemplazado por una variable n que contiene un valor entero
- Se reemplaza la operación $\$i = \$i + 1$ por la operación de PHP equivalente $\$i++$





Instrucción Para (for)

¿Cómo ejecuta?

```
PARA  $i \leftarrow 0$  HASTA  $n$  PASO 1 HACER  
    instruccion1  
    instruccion2  
    ...  
    instruccionN  
FIN PARA
```

```
for ( $\$i = 0$ ;  $\$i < \$n$ ;  $\$i++$ ) {  
    instruccion1;  
    instruccion2;  
    ...  
    instruccionN;  
}
```

1°- Inicializo la variable contador i ($i \leftarrow 0$. puedo inicializar en cualquier valor dependiendo del problema: 1, 10,..)

2°- evalúo si $0 \leq i < N$ (entre el valor inicial y el umbral)

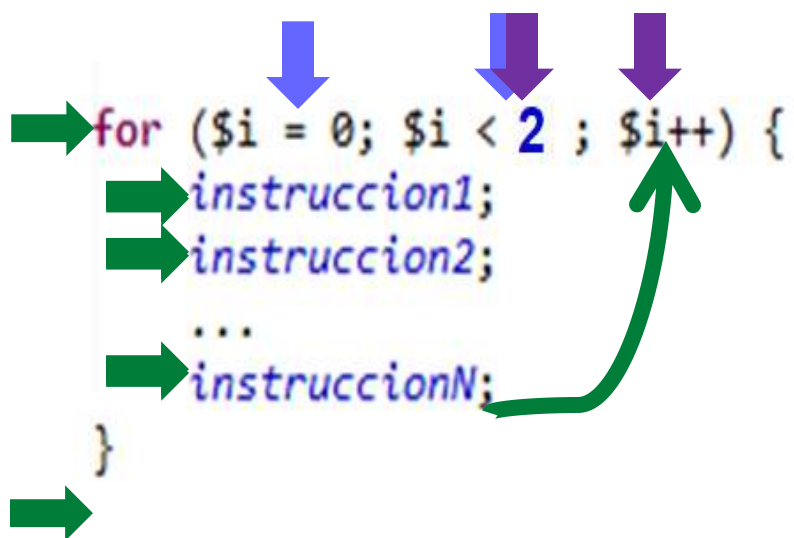
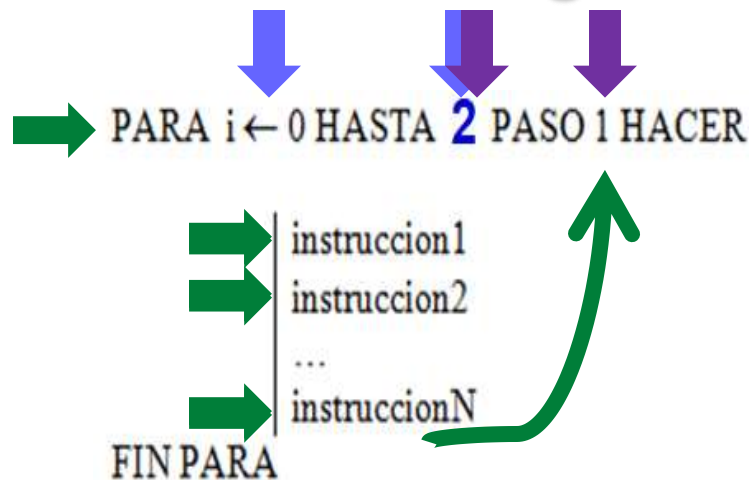
3°- si 2° es Verdadero entonces
a) ejecutar el bloque de instrucciones
b) incrementar, según indica el paso, la variable contador i
c) volver a 2°

4°- si 2° es Falso, terminar la instrucción.

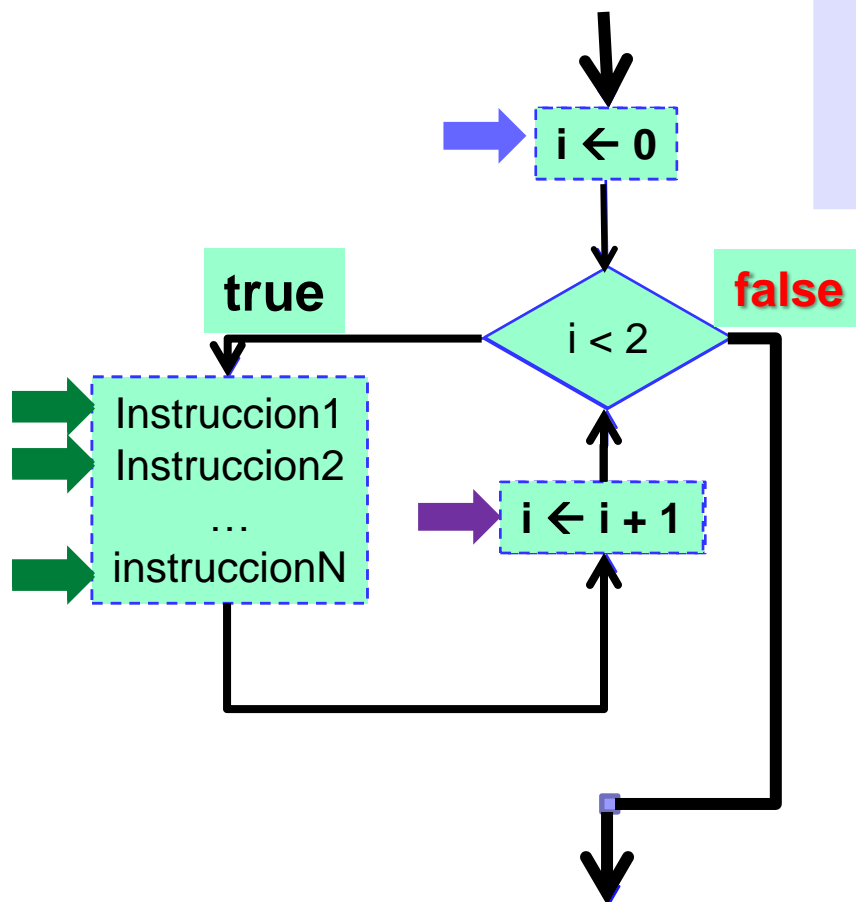


Instrucción Para (for)

¿Cómo ejecuta?



CICLOS: 2



traza

i

0

1

2



Instrucción PARA (for)

CONSIDERACIONES

- Al comenzar la ejecución de la instrucción PARA puede pasar que inicializamos la variable contador i en un valor, y el umbral sea menor al valor de i, por lo tanto al comenzar a ejecutar la instrucción PARA, i superó el valor umbral y el bloque de instrucciones nunca será ejecutado, es decir, la instrucción producirá CERO ciclos. Esto puede ser correcto según el problema que estemos resolviendo.
- La instrucción for puede ejecutar ciclos infinitos, si hacemos mal la traducción de pseudocódigo a PHP.

PARA $i \leftarrow \text{valorInicial}$ HASTA valorUmbral PASO 1 HACER

instruccion1
instruccion2
...
instruccionN

FIN PARA

```
for ($i = valorInicial; $i < valorUmbral; $i=$i+1) {  
    instruccion1;  
    instruccion2;  
    ...  
    instruccionN;  
}
```



Ojo con Ciclos infinitos!

*Al traducir a php incluimos una condición:
 $\$i < \text{umbral}$*

Si nos equivocamos en la traducción puede producirse un ciclo infinito



Instrucción For

**Ejercicio 3 TP6: escribir los
números positivos
MENORES a N**



Repasemos ...

¿Cuándo es **más adecuada** una instrucción?

¿Si conocemos
explícitamente la
cantidad de Iteraciones?

para
(for)

¿Si definimos una
condición para continuar
ejecutando el bloque?

mientras
(while)

repite 0 o más

repetir..mientras
(do..while)

repite 1 o más



Repasemos ...

Bucle **MIENTRAS**:

Repite un conjunto de instrucciones mientras la condición sea verdadera. Es decir termina cuando la condición sea falsa. Ejecuta 0 o más ciclos.

```
while (condición){  
    //acciones del bucle  
}
```

Bucle **REPETIR MIENTRAS**:

Repite un conjunto de instrucciones mientras la condición sea verdadera, es decir, termina de ciclar cuando la condición es falsa. A diferencia de la anterior, se ejecuta al menos 1 ciclo.

```
do {  
    //acciones del bucle  
} while (condición);
```

Bucle **PARA**:

Introduce un ciclo definido, en el cual se conoce el valor inicial y final de la variable iteradora, y cuál es incremento de la variable en cada iteración.

```
for (inicio;condicion;incremento) {  
    //acciones del bucle  
}
```



Repasemos ...

MIENTRAS: Repite mientras la condición sea verdadera. Ejecuta 0 o más ciclos.

PROGRAMA PRINCIPAL

secuencia

MIENTRAS (expresion booleana) **HACER**

Bloque de

instrucciones a

Repetir

FIN MIENTRAS

secuencia

FIN PROGRAMA....

REPETIR MIENTRAS: Repite hasta que la condición sea verdadera. Ejecuta 1 o más ciclos.

PROGRAMA PRINCIPAL

secuencia

REPETIR

Bloque de

Instrucciones a

Repetir

MIENTRAS (expresion booleana)

secuencia

FIN PROGRAMA....



Repasemos ...

PARA: Repite desde un limite inferior a uno superior incrementando la variable **contador** según el incremento (varIncr).

PROGRAMA PRINCIPAL

secuencia

PARA cont \leftarrow limInferior **HASTA** limSuperior **PASO** varIncr **HACER**

Cuerpo de la

Estructura

Repetitiva

FIN PARA

secuencia

FIN PROGRAMA



Variables “especiales”

En las estructuras repetitivas llamaremos a algunas variables con nombres “especiales” según la función que cumplen en la repetición:

Contadores: son variables de **tipo de dato Entero**. **Antes de la instrucción repetitiva debe ser inicializada (por lo general en 0, pero puede ser cualquier valor entero)**. Luego, se incrementan en un valor constante en cada repetición (por lo general en 1).

Ejemplo instrucción dentro del cuerpo de instrucciones que se repite:

$i \leftarrow i + 2$

...
 $i \leftarrow 0$

...
INICIO INSTRUCCIÓN REPETITIVA

...
 $i \leftarrow i + 1$

...
FIN INSTRUCCIÓN REPETITIVA

Inicialización del contador con un valor entero. Puede ser cero u otro valor entero

Incremento del contador en un valor constante. Puede ser 1 o cualquier otro valor distinto de cero.



Variables “especiales”

En las estructuras repetitivas llamaremos a algunas variables con nombres “especiales” según el rol que cumplen en la repetición:

Acumuladores (suma): se utiliza para efectuar sumas sucesivas. **Antes de la instrucción repetitiva, en general, inicializamos la variable en 0,** y dentro de la repetición acumulan valores variables. Ejemplo instrucción dentro del cuerpo de instrucciones que se repite: **`$suma = $suma + $numero`**

Acumuladores (Producto): se utiliza para efectuar productos sucesivos. **Antes de la instrucción repetitiva, en general, inicializamos la variable en 1,** y dentro de la repetición acumula el producto de valores variables. Ejemplo instrucción dentro del cuerpo de instrucciones que se repite:
`$prod = $prod * $numero`

Acumuladores (string): se utiliza para efectuar productos sucesivos **Antes de la instrucción repetitiva, en general, inicializamos la variable en “”,** y dentro de la repetición al acumulador se concatena valores string variables. Ejemplo instrucción dentro del cuerpo de instrucciones que se repite:
`$oracion = $oracion . $palabra`



Variables “especiales”

En las estructuras repetitivas llamaremos a algunas variables con nombres “especiales” según el rol que cumplen en la repetición:

Bandera (flag): Variable que asume valor true o false. **Antes de la instrucción repetitiva se debe inicializar**, y se utiliza dentro de la condición de un bucle, para determinar si el bucle sigue o no iterando.

Ejemplo instrucción dentro del cuerpo de instrucciones que se repite:
\$bandera = \$numero <> -1



Variables “especiales”

```
/*  
 * Sumatoria de los Primeros N Nros Naturales  
 * @param integer $N  
 * @return integer  
 */
```

```
function sumatoria($N){  
    $acum = 0;  
    for ($i = 1; $i <= $N; $i++) {  
        $acum = $acum + $i;  
    }  
    return $acum;  
}
```

**Inicialización de las variables
\$acum y \$i**

\$i es un Contador

\$acum es un Acumulador



Variables “especiales”

```
/*
 * Productoria de los Primeros N Nros Naturales
 * @param integer $N
 * @return integer
 */
function productoria($N){
    $i = 1;
    $prod = 1;
    while ( $i <= $N ) {
        $prod = $prod * $i;
        $i++;
    }

    return $prod;
}
```

Inicialización de las variables
\$i y \$prod

\$prod es un Acumulador

\$i es un Contador

Ejercicio: Hacer la traza para N=4



Variables “especiales”

```
/**  
 * Permite ingresar usuario y contraseña.  
 * Verifica si es posible o no usar el sistema.  
 * El usuario puede seguir intentando mientras no ingrese "salir"  
 */
```

```
function login(){  
    $verifica = false;  
    $salir = false;
```

**Inicialización de las variables
\$verifica y \$salir**

```
while(!$verifica && !$salir){  
    echo "Ingrese Nombre usuario o Salir:";  
    $usuario = trim(fgets(STDIN));  
    if($usuario=="Salir" || $usuario=="salir"){  
        $verifica = false;  
        $salir = true;  
    }else{  
        echo "Ingrese contraseña:";  
        $contrasenia = trim(fgets(STDIN));  
        if($usuario == 'majo' && $contrasenia='123'){  
            $verifica = true;  
        }elseif($usuario == 'gigi' && $contrasenia='123'){  
            $verifica = true;  
        }else{  
            $verifica = false;  
            echo "Vuelva a intentarlo, el usuario/contraseña no se pudieron verificar.\n";  
        }  
    }  
}  
return $verifica;  
}
```

**\$verifica y \$salir son
banderas**

Ejercicio: Hacer trazas para distintos valores de entrada



Tipos de Ciclos

Según el tipo de problema que necesitamos resolver y la condición de corte de la repetición:

- ciclos definidos
- ciclos indefinidos
 - ciclos interactivos
 - ciclos con centinelas

Leer apunte en Pedco!



Tipos de Ciclos

ciclos definidos:

Antes de iniciar la instrucción repetitiva se conoce cuantos ciclos debe ejecutar la instrucción repetitiva.

Se recomienda utilizar la instrucción repetitiva **PARA (for)**.

observación: para ciclos definidos también se pueden utilizar otras instrucciones, pero en la materia evaluaremos que puedan identificar cuándo es un ciclo definido y que utilicen la instrucción recomendada

Ejemplo: sumar las notas de los N alumnos de la materia Introducción a la Programación.



Tipos de Ciclos

ciclos indefinidos:

No se conoce a priori cuántos ciclos debe ejecutar la instrucción repetitiva. Se repite mientras una condición sea verdadera.

Se recomienda utilizar la instrucción repetitiva
MIENTRAS(while) o
REPETIR..MIENTRAS(do..while)



Tipos de Ciclos

Algunos ciclos indefinidos:

ciclos interactivos: en cada iteración se le pregunta al usuario si continuar.

Ejemplo: sumar la cantidad de notas de alumnos mientras el usuario desee seguir ingresando valores. En cada ciclo debemos preguntar al usuario si desea ingresar un nuevo valor.

ciclos con centinelas: se repite mientras no coincide con el centinela

Ejemplo: sumar la cantidad de notas de alumnos mientras la nota sea distinta de -1.



```
function main( )  
{  
  for (count = 1; count <= 500; count++)  
    echo "NO TIRARE AVIONES DE PAPEL EN CLASE" ;  
}  
  
main( ) ;
```

BUEN INTENTO





Un Informático con insomnio

