Unidad 6: Sentencias repetitivas

Cátedra Introducción a la Programación
Facultad de Informática
Buenos Aires 1400 (8300) Neuquén
Universidad Nacional del Comahue, Argentina

---

1. ESTRUCTURAS REPETITIVAS: CICLOS

Un problema analizado en la unidad anterior decía:

> Leer un número. Si el número es positivo escribir un mensaje "Numero positivo", si el número es igual a 0 un mensaje "Igual a 0", y si el número es negativo escribir un mensaje "Numero negativo".

Se nos plantea a continuación un nuevo problema, similar al anterior:

Problema 6.1.. El usuario debe poder ingresar muchos números y cada vez que se ingresa uno debemos informar si es positivo, cero o negativo.

Una opción para implementar la solución es utilizando ***ciclos definidos***, preguntando al usuario al inicio del programa, cuántos números va a ingresar para consultar.

```
MODULO analizandoSec( Entero i ) RETORNO ∅
    Entero j, x
    PARA j← 1 HASTA i+1 PASO 1
        LEER (x)
        SI (x >0) ENTONCES
                ESCRIBIR(x, "es positivo")
        OTRO-SI (x = 0 ) ENTONCES
                ESCRIBIR(x, "es igual a cero")
        SINO
                ESCRIBIR(x, "es No positivo")
        FIN SI
    FIN PARA
FIN MODULO


ALGORITMO principal
    Entero cantidad
    ESCRIBIR(Çuántos nros desea procesar?")
    LEER(cantidad)
    analizandoSec(cantidad)
FIN ALGORITMO principal
```

A continuación se implementa en PHP la solución con una estructura repetitiva **for** :

```php
function  analizandoSec_for($i){
        for ($j = 1; $j <= $i; $j++) {
                echo "Ingrese un valor";
                $numero = trim(fgets(STDIN));
                if ($numero > 0)
                        echo "Numero positivo";
                elseif ($numero == 0)
                        echo  "Igual a 0";
```

```
9                   else
10                              echo  "Numero negativo";
11          }
12  }
```

A continuación se implementa en PHP la solución con una estructura repetitiva **while** :

```
1   function  analizandoSec_while($i){
2           $j=1;
3           while ($j <= $i) {
4                   echo "Ingrese un valor";
5                   $numero = trim(fgets(STDIN));
6                   if ($numero > 0)
7                           echo "Numero positivo";
8                   elseif ($numero == 0)
9                           echo  "Igual a 0";
10                  else
11                          echo  "Numero negativo";
12                  $j ++;
13          }
14  }
```

Sin embargo el usuario puede considerar que esta solución no es muy práctica, porque lo obliga a contar de antemano cuántos números va a querer procesar, sin equivocarse, en lugar de ingresar uno a uno los números hasta procesarlos a todos.

1.1. Ciclos indefinidos

Para poder resolver este problema sin averiguar primero la cantidad de números a procesar, debemos introducir una instrucción que nos permita construir ciclos que no requieran que se informe de antemano la cantidad de veces que se repetirá el cálculo del cuerpo. Se trata de ***ciclos indefinidos*** en los cuales se repite el cálculo del cuerpo mientras una cierta condición es verdadera.

Un ciclo indefinido es de la forma

```
1   while (condición){
2           <hacer algo>
3   }
```

Donde **while** es una palabra reservada, la condición es una expresión booleana, igual que en las instrucciones alternativas **"if"**. Y el cuerpo es, como siempre, una o más instrucciones de PHP.

El diseño de la solución propuesta es el siguiente:

```
MODULO pcn_loop() RETORNO ∅
    String hayDatos, Entero x
    hayDatos <– "si"
    MIENTRAS ( hayDatos = "si" ) ENTONCES
        LEER (x)
        SI (x >0) ENTONCES
                ESCRIBIR(x, "es positivo")
        OTRO-SI (x = 0 ) ENTONCES
                ESCRIBIR(x, "es igual a cero")
        SINO
                ESCRIBIR(x, "es No positivo")
        FIN SI
        ESCRIBIR ("Hay más datos? si/no"
        LEER(hayDatos)
    FIN MIENTRAS
FIN MODULO
```

ATENCIÓN:

Observación: *Si bien en la materia recomendamos implementar funciones con retorno, pues nos permiten reusar el resultado en otras expresiones, aún no contamos con las herramientas (por ejemplo 'arreglos' que estudiaremos en la siguiente unidad) para retornar un conjunto de datos. Por lo tanto para el caso de repetitivas que NO produzcan un único cálculo (por ejemplo, una sumatoria de números), deberemos escribir los resultados en pantalla.*

1.2. Ciclos interactivos

¿Cuál es la condición y cuál es el cuerpo del ciclo en nuestro problema? Claramente, el cuerpo del ciclo es el ingreso de datos y la verificación de si es positivo, negativo o cero. En cuanto a la condición, es que haya más datos para seguir calculando.

Definimos una variable *hayMasDatos*, que valdrá "Si" mientras haya datos. Se le debe preguntar al usuario, después de cada cálculo, si hay o no más datos. Cuando el usuario deje de responder "Si", dejaremos de ejecutar el cuerpo del ciclo.

Una aproximación al código necesario para resolver este problema podría ser:

```
1  function  loop_interactivo(){
2          $hayMasDatos = "Si";
3          while ($hayMasDatos == "Si"){
4                  echo "Ingrese un valor";
5                  $numero = trim(fgets(STDIN));
6                  if ($numero > 0)
7                          echo "Numero positivo";
8                  elseif ($numero == 0)
9                          echo  "Igual a 0";
10                 else
11                         echo  "Numero negativo";
12                 echo "¿Quiere seguir? <Si-No>: ";
13                 $hayMasDatos = trim(fgets(STDIN));
14         }
15 }
```

En la implementación se evalua la condición *($hayMasDatos == "Si")*, esta opción supone que si llamó a este programa es porque tiene algún dato para calcular, y darle el valor inicial "Si" a hayMasDatos.

El esquema del ciclo interactivo es el siguiente:

—*hayMasDatos* hace referencia a "Si".

—Mientras hayMasDatos haga referencia a "Si":
  —Pedir datos.
  —Realizar cálculos.
  —Preguntar al usuario si hay más datos ("Si" cuando los hay). *hayMasDatos* hace referencia al valor ingresado.

1.3. Ciclo con centinela

Un problema que tiene nuestra primera solución es que resulta poco amigable preguntarle al usuario después de cada cálculo si desea continuar. Se puede usar el método del *centinela*: un valor que se define y que, si se lee, le indica al programa que el usuario desea salir del ciclo. En este caso, podemos suponer que si ingresa el caracter * indica que desea terminar.

El esquema del ciclo con centinela es el siguiente:

1. Pedir datos.
2. Mientras el dato pedido no coincida con el centinela:
   —Realizar cálculos.
   —Pedir datos.

En nuestro caso, pedir datos corresponde a :

—Pedir número.

El programa resultante es el siguiente:

```
1  function  loop_centinela(){
2          echo "Ingrese un numero ('*' para terminar):";
3          $numero = trim(fgets(STDIN));
4          while ($numero <>"*"){
5                  if ($numero > 0)
6                          echo "Numero positivo";
7                  elseif ($numero == 0)
8                          echo  "Igual a 0";
9                  else
10                         echo  "Numero negativo";
11                 echo "Ingrese un numero ('*' para terminar): ";
12                 $numero = trim(fgets(STDIN));
13         }
14 }
```

El ciclo con centinela es muy claro pero tiene un problema: hay dos lugares (la primera línea del cuerpo y la última línea del ciclo) donde se ingresa el mismo dato. Si en la etapa de mantenimiento tuviéramos que realizar un cambio en el ingreso del dato (cambio de mensaje, por ejemplo) deberíamos estar atentos y hacer dos correcciones iguales.

Sería preferible poder leer el dato x en un único punto del programa.

A continuación, tratamos de diseñar una solución con esa restricción.

Es claro que en ese caso la lectura tiene que estar dentro del ciclo para poder leer más de un número, por lo tanto le tendremos que dar un valor inicial a la variable, de forma que nos permita entrar el ciclo y cortarlo en el momento requerido.

Codificamos en PHP la solución al problema de los números usando ese esquema:

```
1  function  loop_centinela2(){
2          $x = -1;
3          while ($x <>"*"){
4                  echo "Ingrese un numero ('*' para terminar): ";
5                  $x = trim(fgets(STDIN));
6                  if( $x == "*")
7                          echo  "Saliendo....";
8                  elseif ($x > 0)
9                          echo  "Numero positivo";
10                 elseif ($x == 0)
11                         echo  "Igual a 0";
12                 else
13                         echo  "Numero negativo";
14         }
15 }
```

ATENCIÓN:

Desde hace mucho tiempo los ciclos infinitos vienen trayéndoles dolores de cabeza a los programadores. Cuando un programa deja de responder y se queda utilizando todo el procesador de la computadora, suele deberse a que el programa entró en un ciclo del que no puede salir. Estos ciclos pueden aparecer por una gran variedad de causas. A continuación algunos ejemplos de ciclos de los que no se puede salir, siempre o para ciertos parámetros. Queda como ejercicio encontrar el error en cada uno.

```
function menor_factor_primo($x){
        #Devuelve el menor factor primo del número x.
        $n = 2
        while ($n <= $x)
                if ($x %$n == 0)
                        return $n;
}
```

```
function buscar_impar($x){
        #Divide el número recibido por 2 hasta que sea impar.
        while ($x %2 == 0)
                $x = $x / 2 ;
        return $x ;
}
```

1.4. Ejercicios

Ejercicio 5.1.. Nuevamente, se desea facturar el uso de un telefono. Para ello se informa la tarifa por segundo y la duracion de cada comunicacion expresada en horas, minutos y segundos. Como resultado se informa la duracion en segundos de cada comunicacion y su costo. Resolver este problema usando

1. Ciclo definido.
2. Ciclo interactivo.
3. Ciclo con centinela.

Ejercicio 5.2.. Mantenimiento del tarifador: al final del día se debe informar cuántas llamadas hubo y el total facturado. Hacerlo con todos los esquemas anteriores.

Ejercicio 5.3.. Nos piden que escribamos una función que le pida al usuario que ingrese un número positivo. Si el usuario ingresa cualquier cosa que no sea lo pedido se le debe informar de su error mediante un mensaje y volverle a pedir el número. Resolver este problema usando

1. Ciclo interactivo.
2. Ciclo con centinela.

1.5. Resumen

—Además de los ciclos definidos, en los que se sabe cuáles son los posibles valores que tomará una determinada variable, existen los ciclos indefinidos, que se terminan cuando no se cumple una determinada condición.

—La condición que termina el ciclo puede estar relacionada con una entrada de usuario (ciclo interactivo) o depender del procesamiento de los datos (ciclo centinela).

REFERENCIA DEL LENGUAJE PHP

for ($j = 1; $j <= $i; $j++) {
<bloque>
}

Introduce una estructura repetitiva **for** donde <bloque> se repite desde $j=1 hasta que $j llega al valor $i

$j=1;
while ($j <= $i) {
<bloque>
$j++;
}

Introduce una estructura repetitiva **while** donde <bloque> se repite desde $j=1 hasta que $j llega al valor $i