El futuro digital es de todos | MinTIC

UNIVERSIDAD EL BOSQUE | MisiónTIC 2022

Ciclo 1

Semana 4

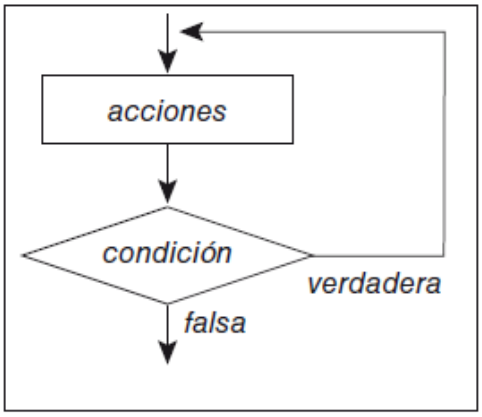
Estructuras de control: repetición o iteración

Lectura 4 – *Estructura haga-mientras (do-while)*

Convenio de asociación no. 777 de 2021 suscrito entre el Fondo Único de Tecnologías de la Información y las Comunicaciones y la Universidad el Bosque.

Estructura haga-mientras (do-while)

El ciclo while y el ciclo for se denominan bucles pre-test porque evaluan la condición al comienzo, pero en algunas ocasiones se requiere que el bucle se ejecute al menos una vez sin importar la condición, bucles post-test, como es el caso de la estructura do-while. El cuerpo del bucle se ejecutará al menos una vez y luego dependiendo de la condición, el ciclo se repetirá, condición es verdadera, o no, condición es falsa. Como se puede ver en la siguiente figura:




```
hacer
   <acciones>
mientras  (<expresión>)
```

Sentencia hacer-mientras::=

```
hacer
  <cuerpo del bucle>
mientras (<condición_del_bucle>)
```

donde

```
<cuerpo del bucle>::= <sentencia>
                   ::= <sentencia_compuesta>

<condición del bucle>::= <expresión booleana>
```

Nota: el cuerpo del bucle se repite mientras <condición del bucle> sea verdadero.

Figura 1: tomado de Joyanes

Suponga que se debe hacer un algoritmo que lea un número entero (por ejemplo, 198) y se obtenga el número inverso (por ejemplo, 891). Una solución sería:

```
algoritmo invertirnummero
var
  entero: num, digitoSig
inicio
  num ← 198
  escribir ('Número: ← ', num)
  escribir ('Número en orden inverso: ')
  hacer
      digitoSig = num MOD 10

      escribir(digitoSig)
      num = num DIV 10
  mientras num > 0
fin
```

La salida de este programa se muestra a continuación:

```
Número:   198
Número en orden inverso:     891
```

Figura 2: Tomado de Joyanes

Con cada iteración se obtiene el dígito más a la derecha, ya que es el residuo de la división enteradel valor del número por 10. Así, en la primera iteración digitoSig vale 8, que es el residuo de la división entera de 198 en 10, luego se divide 198 entre 10 y se toma el cociente entero 19 y se asigna a la variable num. Se verifica la condición que es verdadera puesto que num es 19 y se comienza otra iteración. En esta iteración digitoSig vale 9, que es el residuo de la división entera de 19 en 10, luego se divide 19 entre 10 y se toma el cociente entero 1 y se asigna a la variable num. Se verifica la condición que es verdadera puesto que num es 1 y se comienza otra

iteración. En esta iteración digitoSig vale 1, que es el residuo de la división entera de 1 en 10, luego se divide 1 entre 10 y se toma el cociente entero 0 y se asigna a la variable num. Al evaluar la condición el ciclo se detiene porque num ya no es mayor que cero.

Es oportuno aclarar que esta estructura tal como se presentó no tiene implementación directa en Python, se puede simular por supuesto, como se verá en semanas posteriores.