INFORMATORIO- UNIDAD III: ESTRUCTURAS DE CONTROL

Apunte de Teoría

Elaborado por Noelia Pinto | Blas Cabas Geat

Tabla de contenido

Unidad III: Estructuras de Control

Contenido Temático

a. Estructuras Secuenciales.
b. Estructuras Condicionales.
c. Estructuras Repetitivas.

Estructuras de Control

Introducción

Las estructuras de control tienen un fin en concreto: señalar orden en que tienen que sucederse los pasos de un programa. Dichas estructuras poseen las siguientes características:

- Una estructura de control tiene un único punto de entrada y un único punto de salida.
- Una estructura de control se compone de sentencias o de otras estructuras de control.

Tales características permiten desarrollar de forma muy flexible todo tipo de algoritmos aún cuando sólo existen tres tipos fundamentales de estructuras de control:

- Secuenciales
- Condicionales
- Repetitivas

Estructuras Secuenciales

La *estructura secuencial* es aquella en la que una acción (instrucción) sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente, hasta el fin del proceso.

Las estructuras secuenciales involucran al conjunto de acciones simples que hemos visto en la unidad anterior. Un ejemplo en Python de un algoritmo usando estructuras secuenciales sería el siguiente:

Ejemplo 1
Dados dos números enteros ingresados por el usuario, calcule y muestre la suma obtenida.

Solución en Python

```
x = input('Ingrese primer número')
y = input('Ingrese segundo número')
a = int(x) + int(y)
print('El resultado de la suma entre los números %s y %s es:%s' %(x, y, a))
```

Estructuras Condicionales

No todos los problemas pueden resolverse empleando estructuras secuenciales.

Cuando hay que tomar una decisión aparecen las *estructuras condicionales*. En nuestra vida diaria se nos presentan situaciones donde debemos decidir.

- ¿Elijo la carrera A o la carrera B?
- ¿Me pongo este pantalón azul o aquél rojo?
- Para ir al trabajo, ¿elijo el camino A o el camino B?

Por supuesto que en un problema se combinan estructuras secuenciales y condicionales.

Es habitual que, llegado a un punto de la resolución de un problema, existan diferentes acciones que pueden llegar a ejecutarse. Pero la elección de las acciones a ejecutar no se realiza al azar, sino que existe una condición según la cual ejecutaremos unas u otras acciones.

Estructura Condicional Simple

Este es el tipo más sencillo de estructura condicional. Sirve para implementar acciones condicionales del tipo siguiente:

- Si se verifica que una determinada condición se cumple, se puede ejecutar una serie de instrucciones y luego seguir adelante.
- Si la condición NO se cumple, NO se ejecutan dichas instrucciones y se sigue adelante.

La condición al evaluarse debe arrojar un resultado VERDADERO para que el conjunto de acciones pueda ejecutarse

Una vez evaluada la expresión condicional, el flujo del programa recupera su caracter secuencial y se continúa ejecutando las siguientes instrucciones.

Ejemplo 2
Solicitar al usuario que ingrese una edad, su programa debe emitir un mensaje si es Mayor de Edad

Solución en Python

```
edad = int(input('ingrese edad'))
if (edad >= 18):
            print('Es mayor de edad')
```

Estructura Condicional Alternativa

Este tipo de estructura permite implementar condicionales en los que hay dos acciones alternativas:

- Si se verifica una determinada condición, ejecutar una serie de instrucciones (bloque 1).
- Si no, esto es, si la condición NO se verifica, ejecutar otra serie de instrucciones (bloque 2).

Ejemplo 3
Dado dos números ingresados por el usuario, indique cuál de los dos es mayor

Solución en Python

```
numero1=int(input('Ingrese primer número'))
numero2=int(input('Ingrese segundo número'))
if (numero1 > numero2):
    print('El número %s es mayor al número %s' %(numero1, numero2))
else:
    print('El número %s es mayor al número %s' %(numero2, numero1))
```

Bloque 1
Si se cumple la condición se ejecutan las acciones del primer bloque

Bloque 2
Si no se cumple la condición se ejecutan las acciones del segundo bloque

Cuando se utiliza Python, en caso de no necesitarse la ejecución de acciones al cumplirse o una condición se puede utilizar la palabra pass. Por ejemplo,

```
edad=int(input('Ingresa tu edad'))
if edad>15:
    pass
else:
    print('No mientas, no tenés', edad)
```

En este caso si el usuario ingresara una edad mayor a 15, no se muestra ningún mensaje. Si fuera al revés, el programa se burla de la edad.

Estructuras de Selección Múltiple

Con frecuencia es necesario que existan más de dos elecciones posibles. Este problema se podría resolver por estructuras selectivas simples o alternativas, anidadas o en cascada.

Pero si el número de alternativas es grande puede plantear serios problemas de escritura y de legibilidad. Por este motivo, en estos casos, se recomienda el uso de la estructura de *selección múltiple* que permite evaluar una variable que puede tomar de 1 a n valores y según ocurra uno de esos valores, se realizará una de las **n** acciones; es decir, que el programa seguirá sólo un determinado camino entre varios.

Ejemplo 4
Dado dos números ingresados por el usuario, indique si la suma entre ellos es positiva, negativa o igual a 0.
Solución en Python

```
n1 = int(input('Ingrese primer número'))
n2 = int(input('Ingrese segundo número'))
suma = n1 + n2
if suma < 0:
    print('La suma es negativa')
elif suma == 0:
    print('La suma es cero')
else:
    print('La suma es positiva')
```

Estructuras Repetitivas

Las estructuras repetitivas o bucles permiten repetir una o varias acciones un número determinado de veces. Cada una de las repeticiones se conoce como *iteración*.

El conjunto de instrucciones contenidas en un programa que se repite un número determinado de veces se llama *ciclo*.

Los ciclos pueden ser *definidos* (cuando se conoce las veces que se debe iterar) e *indefinidos* (cuando no se conoce cuántas veces se repite el ciclo).

Estructura Pre-Test

- Ciclo Indefinido.
- El final de bucle está controlado con una condición (expresión lógica o booleana). El conjunto de acciones se ejecuta mientras la evaluación de la condición devuelva un resultado *verdadero*.
- El ciclo se puede ejecutar 0 o más veces.

Ejemplo 5
Dado un número entero N calcular la suma de todos los números entre 1 y N.

Solución en Python

```
cont = 0
suma = 0
N = int(input('Ingrese tope màximo: '))
while cont <= N:
    suma = suma + cont
    cont = cont + 1
print('La suma total es: ', suma)
```

Estructura Post-Test

- Ciclo Indefinido.
- El final de bucle está controlado con una condición (expresión lógica o booleana). El conjunto de acciones se ejecuta mientras la evaluación de la condición devuelva un resultado *falso*.
- El ciclo se ejecuta al menos una vez.
- No existe la implementación en Python, por este motivo no nos vamos a explayar aquí.

Estructura Manejado por Contador

- Ciclo Definido.
- El final de bucle está controlado por un contador que define la cantidad de veces a iterar.
- Se conoce de antemano cuántas veces se debe repetir el conjunto de acciones.
- La variable contador se va incrementando automáticamente de acuerdo al incremento indicado. No hace falta inicializarla.

Ejemplo 7
Diseñe un algoritmo que imprima los primeros 8 números naturales.

Solución en Python

```
for x in range(1,9):
    print(x)
```

- Se puede observar la utilización de la función **range(),** en el anterior ejemplo. Esta función trabaja con listas y puede tener uno, dos o tres parámetros:

```
range (4): [0,1,2,3]
range(3,7): [3,4,5,6]
range(4,8,2): [4,6]
range(4,1,-1): [4,3,2]
```

Control de Bucles

- `Break` se puede usar en bucles `for` y `while` y simplemente termina el bucle actual y sigue con la ejecución de la próxima instrucción, por ejemplo:

a) Controlar fin de bucle

```
while True:
    op = input('Ingrese cualquier palabra, termina con FIN--> ')
    if op == 'FIN':
        break
    else:
        print(op)
print('Terminó la ejecución con FIN')
```

b) La sentencia `break` es usada también para terminar un ciclo aun cuando la evaluación de la condición no devuelva `False`, por ejemplo:

```
# Primer ejemplo
for letra in "Python":
    if letra == "h":
        break
    print("Letra actual :", letra)

# Segundo ejemplo
var = 10
while var > 0:
    var = var -1
    if var == 5:
        break
    print("Valor actual de la variable :", var)
```

Si se fijan en el primer ejemplo al llegar a la letra "h" simplemente se termina (rompe) el ciclo (bucle) y se sigue con el segundo ejemplo. En el segundo ejemplo la variable va disminuyendo su valor hasta que llega a 5, en donde se termina (rompe) el ciclo (bucle)

- Al aparecer un `continue` en Python, este regresa al comienzo del bucle, ignorando todos los estamentos que quedan en la iteración actual del bucle e inicia la siguiente iteración. Queda más claro con un ejemplo:

```
# Primer ejemplo
for letra in "Python":
    if letra == "h":
        continue
    print("Letra actual :", letra)

# Segundo ejemplo
var = 10
while var > 0:
    var = var -1
    if var == 5:
        continue
    print("Valor actual de la variable :", var)
```

En el primer ejemplo al llegar a la letra "h" simplemente termina esa iteración (ignorando al print que sigue en la linea 5) y continua con la siguientes iteraciones (letras o y n) hasta que se termina el ciclo (bucle).

En el segundo ejemplo la variable va disminuyendo su valor hasta que llega a 5, en donde se termina esa iteración del ciclo (bucle) y se continúa con las iteraciones que siguen hasta que se termina el bucle.