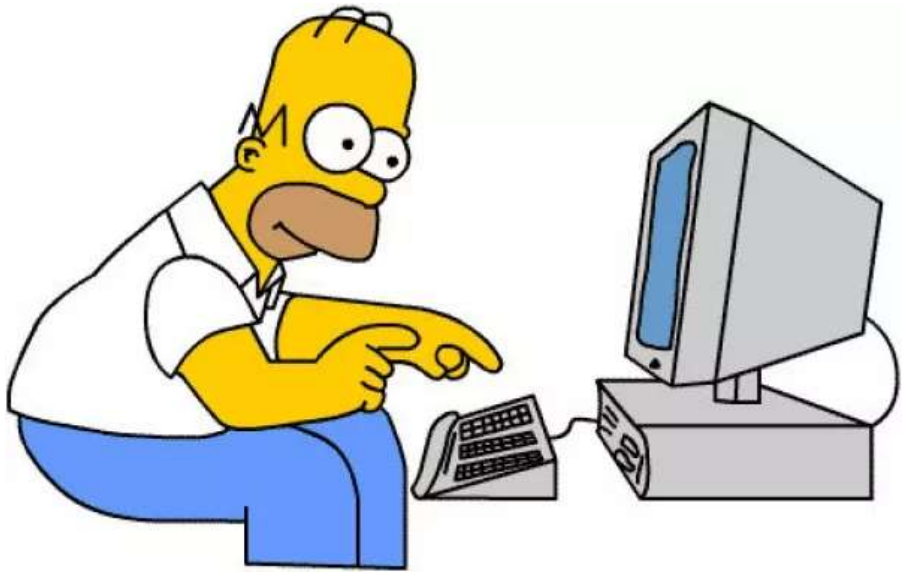
INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.



INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada.

La **programación estructurada** es un estilo con el cual él se busca que el programador elabore programas sencillos y fáciles de entender, la programación estructurada hace uso de tres **estructuras básicas de control** que son: Estructura **Secuencial**, Estructura **Selectiva** (Condicional) y la Estructura **Repetitiva** (Iterativa, de Control).

Mediante la combinación de estas tres estructuras junto con la **programación modular** es posible crear cualquier tarea, obteniendo programas que pueden ser leídos desde su inicio hasta su fin en una forma continua, sin tener que estar saltando de un lugar a otro del programa, tratando de seguir el rastro de la lógica establecida por el programador.

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Las **estructuras de control** de un lenguaje de programación son métodos de especificar el orden en que las instrucciones de un algoritmo se ejecutarán.

Estas nos dan control absoluto sobre el orden de ejecución y nos permiten saltar un bloque de instrucciones o repetir otras.

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

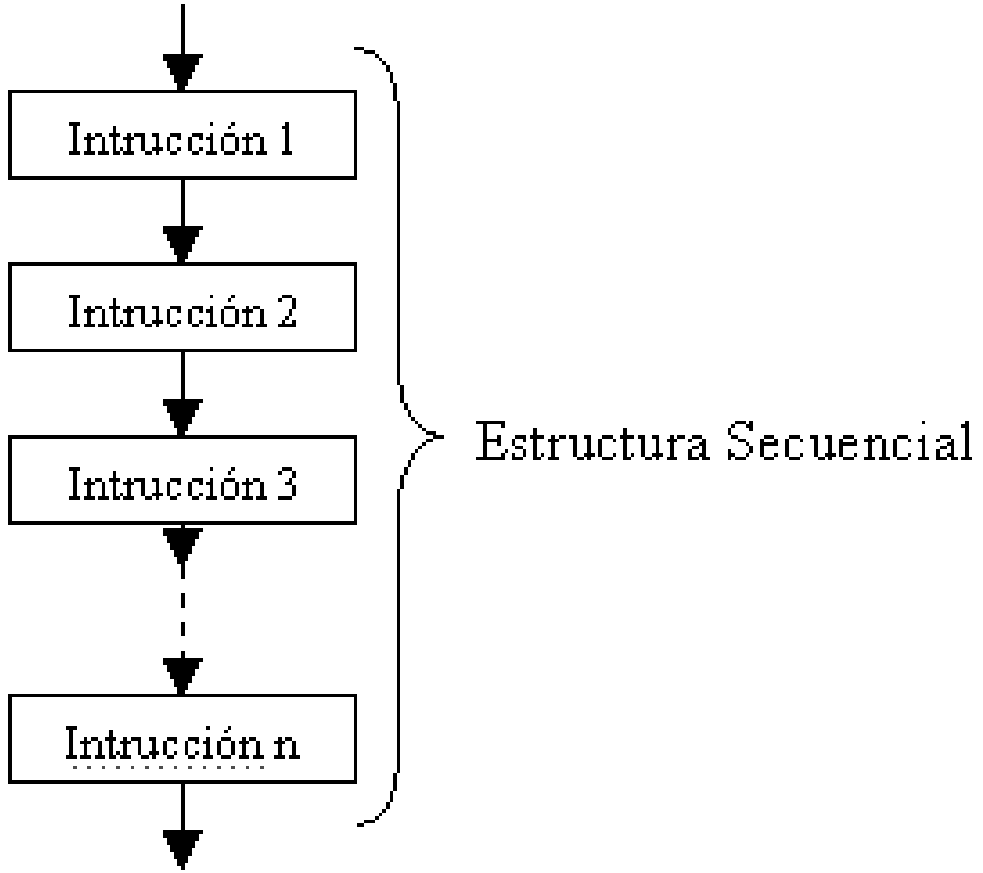
INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Estructura Secuencial

Las sentencias son ejecutadas en el orden en que aparecen escritas en el programa.

Una instrucción no se ejecuta hasta que finaliza la anterior.



INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Estructura Secuencial

- **Declaración de variables y constantes:** Indicar al principio del algoritmo las variables que se van a utilizar a lo largo del programa.

Int Puntos

- **Asignación:** Dar a una variable un valor.

Puntos = 100

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Estructura Secuencial

- **Lectura:** Asignar a una variable un valor recibido por un dispositivo de entrada.

```
edad = int(input('Teclear edad: '))  # entrada de entero
peso = float(input('Teclear peso: '))  # entrada de flotante
nombre = input('Teclear nombre: ')  # entrada de cadena
```

- **Escritura:** Enviar un resultado o un mensaje a un dispositivo de salida.

```
print(nombre, edad, 'años', peso, 'kg')  # muestra datos
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Estructuras Selectivas o Condicionales

En estas se evalúa una condición y en función al resultado se ejecutan una determinada secuencia de instrucciones.

Estas estructuras a su vez se encuentran clasificadas en tres:

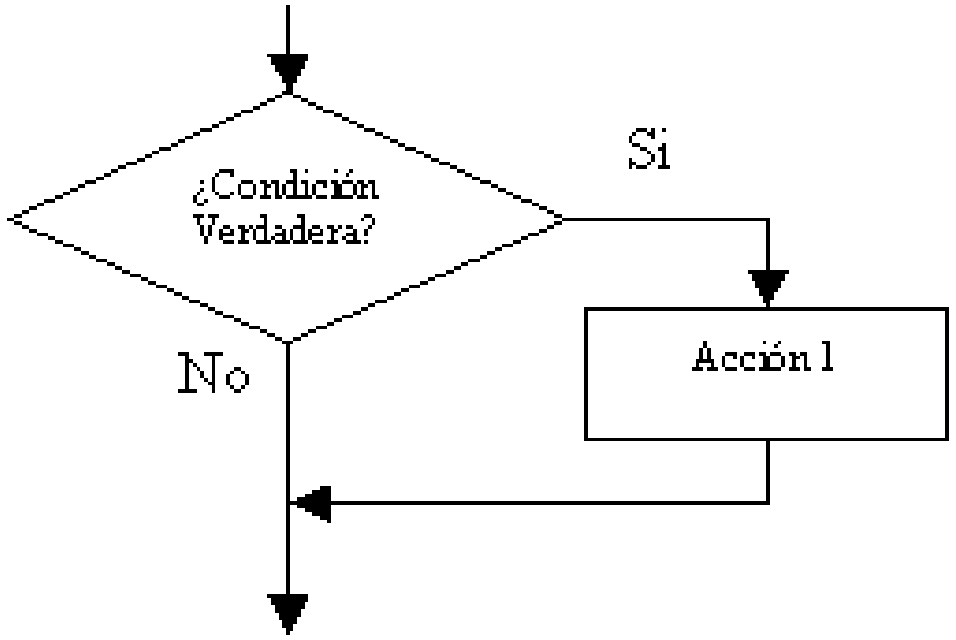
- Selectiva simple.
- Selectiva doble.
- Selectivas múltiples.

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Selectiva simple

Ejecuta una determinada condición y si el resultado es verdadero se ejecuta solo una determinada acción. Si la condición es falsa el programa sigue con su secuencia normal.



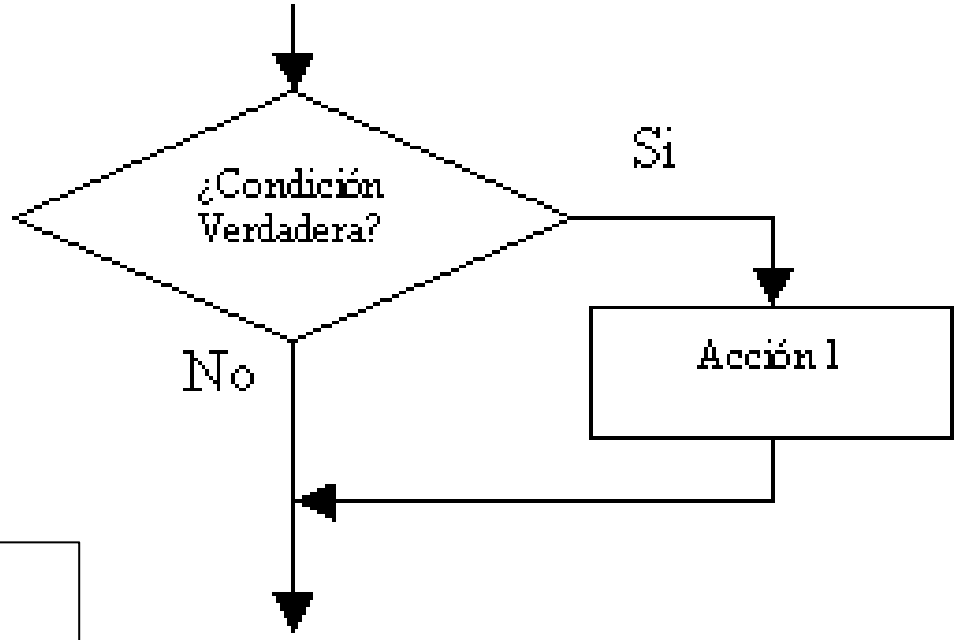

```
SI condición ENTONCES
    Bloque de código
FIN_SI
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Selectiva simple

```
if condición:
        aquí van las órdenes que se ejecutan si
        la condición es cierta y que pueden
        ocupar varias líneas
```




```
numero = int(input("Escriba un número positivo: "))
if numero < 0:
    print("¡Le he dicho que escriba un número positivo!")
    print(f"Ha escrito el número {numero}")
```

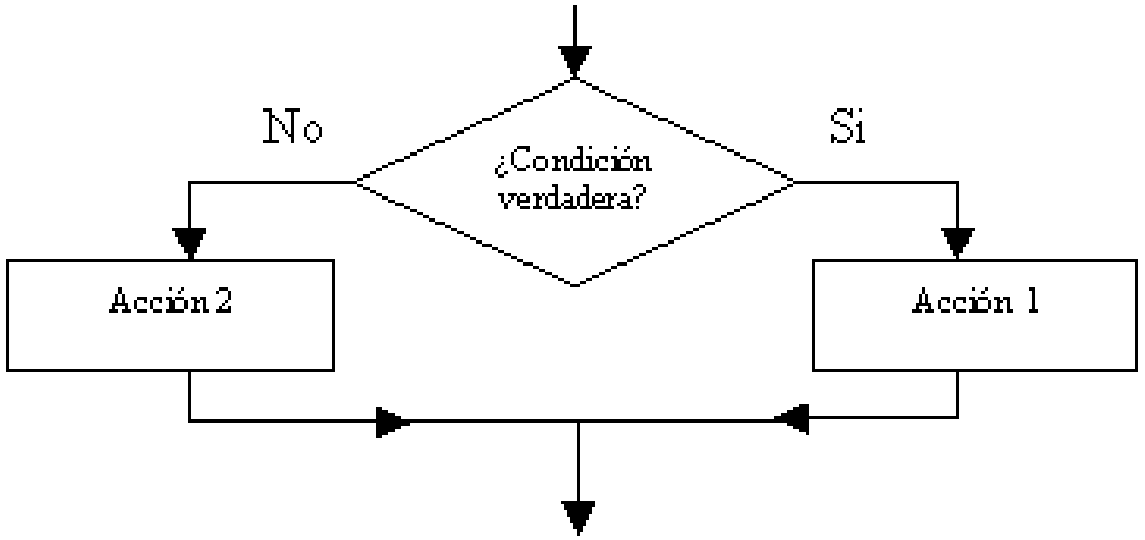
```
SI condición ENTONCES
    Bloque de código
FIN_SI
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Selectiva doble

Permite elegir entre dos opciones posibles en función al resultado de una determinada condición. Si la condición es verdadera, se ejecuta la acción 1 y si es falsa se ejecuta la acción 2.




```
SI condición ENTONCES
        Bloque de código 1
SI_NO ENTONCES
        Bloque de código 2
FIN_SI
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Selectiva doble

```
if condición:
    aquí van las órdenes que se ejecutan si la condición es cierta
    y que pueden ocupar varias líneas
else:
    y aquí van las órdenes que se ejecutan si la condición es
    falsa y que también pueden ocupar varias líneas
```

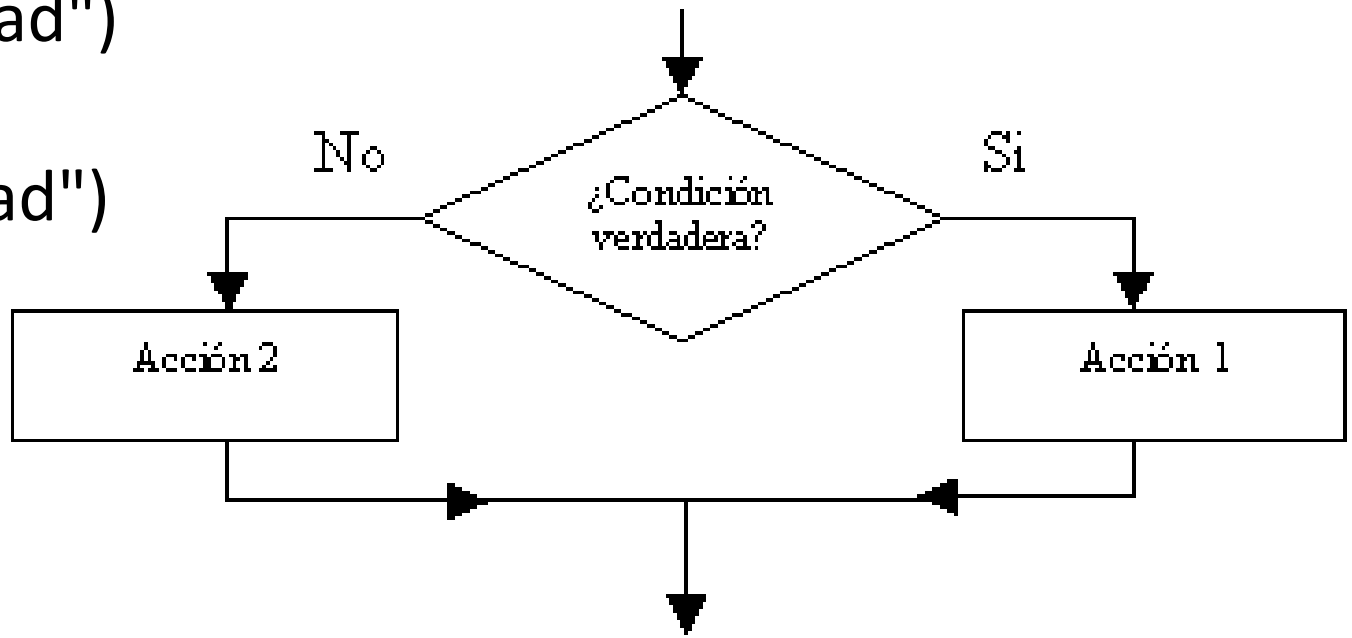
```
SI condición ENTONCES
        Bloque de código 1
SI_NO ENTONCES
        Bloque de código 2
FIN_SI
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Selectiva doble

```
edad = int(input("¿Cuántos años tiene? "))

if edad < 18:
    print("Es usted menor de edad")
else:
    print("Es usted mayor de edad")

print("¡Hasta la próxima!")
```

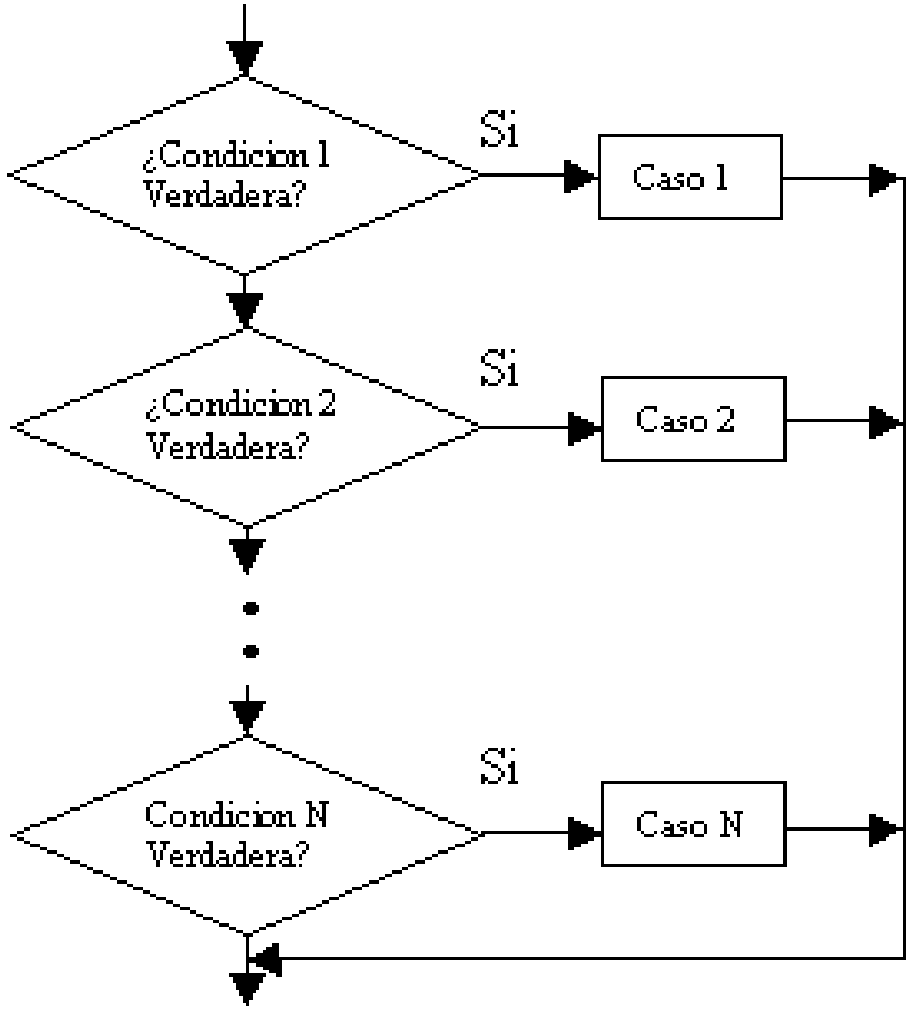


INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Selectivas múltiples

La estructura de selección múltiple evaluará una expresión que podrá tomar n valores distintos 1, 2, 3, 4, …..n. Según se elija uno de estos valores en la condición, se realizará una de las **N** acciones.

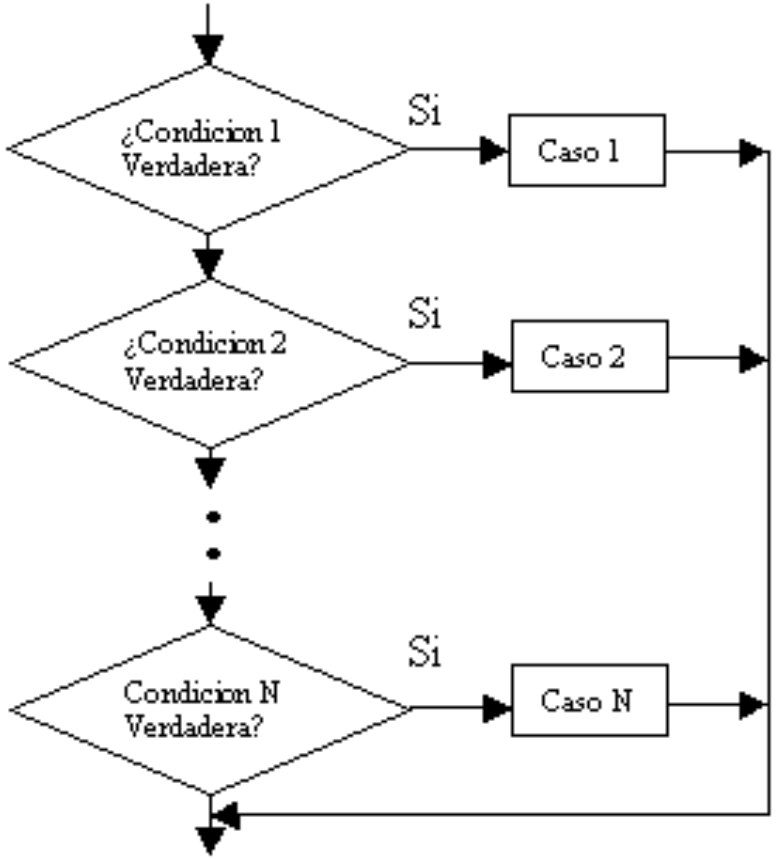


INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Selectivas múltiples

```
SI condición 1 ENTONCES
        Bloque de código 1
SI_NO
    SI condición 2 ENTONCES
        Bloque de código 2
    SI_NO
        Bloque de código 3
    FIN_SI
FIN_SI
```

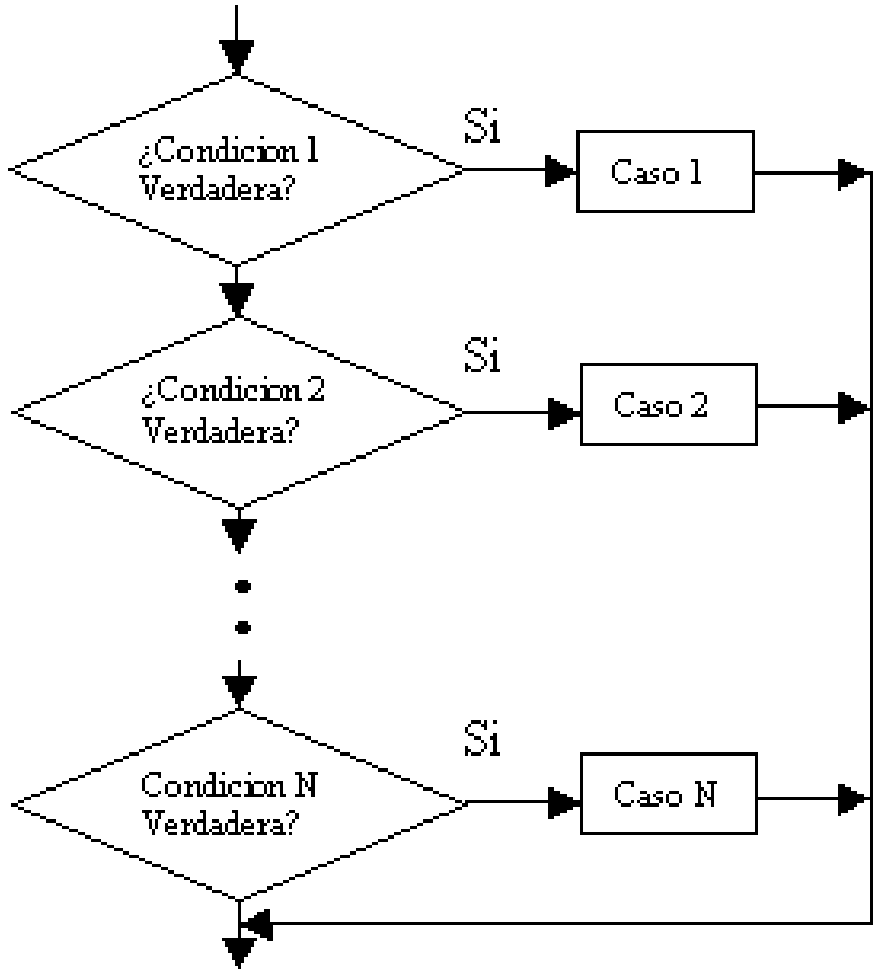


INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Selectivas múltiples

```
if condición_1:
    bloque 1
elif condición_2:
    bloque 2
else:
    bloque 3
```



INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Selectivas múltiples

```
edad = int(input("¿Cuántos años tiene? "))

if edad < 0:
    print("No se puede tener una edad negativa")
elif edad < 18:
    print("Es usted menor de edad")
else:
    print("Es usted mayor de edad")
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Estructuras Repetitivas (Iterativas, de Control).

Son utilizadas para que repitan la ejecución de un conjunto de instrucciones mientras se cumple una cierta condición.

Estas estructuras a su vez se encuentran clasificadas en tres tipos:

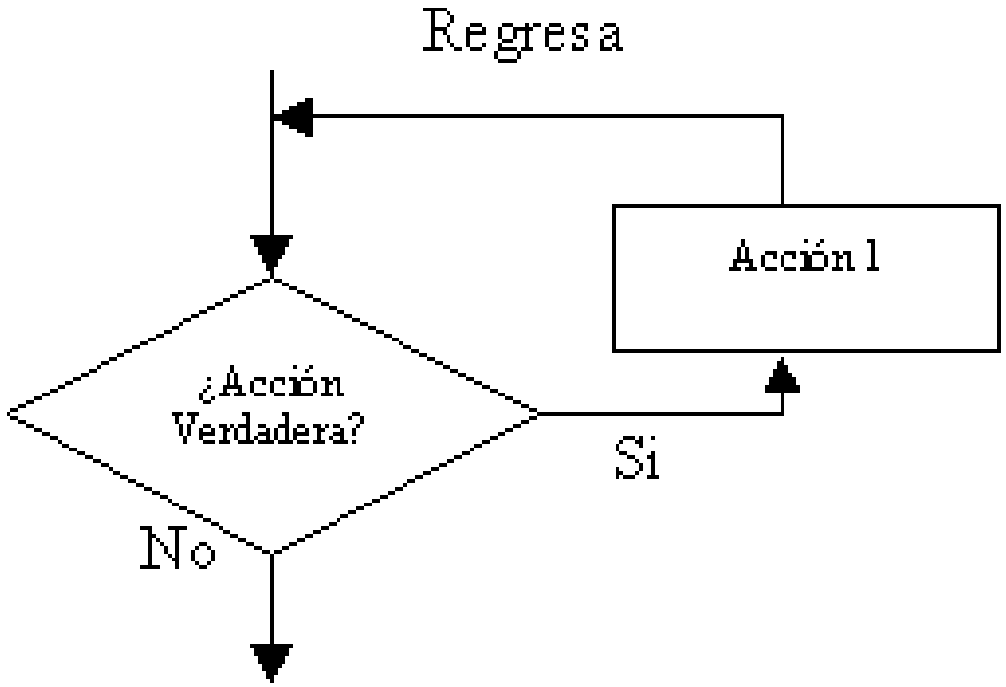
- Mientras
- Repetir
- Desde/Para

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Estructura mientras

La estructura repetitiva “mientras’ es aquella en que el cuerpo del bucle se repite mientras se cumple una determinada condición. Cuando se ejecuta esta instrucción, la primera cosa que sucede es que se evalúa la condición y si la expresión es verdadera, entonces se ejecuta el cuerpo del bucle. Este proceso se repite una y otra vez mientras la condición es verdadera.

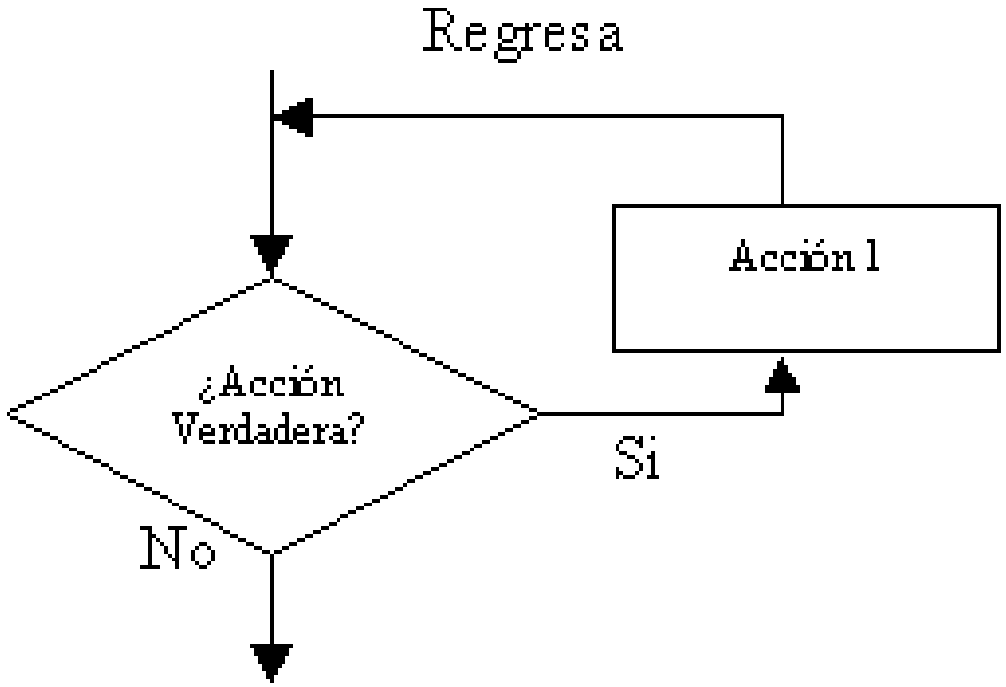


INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Estructura mientras

```
MIENTRA condición HACER:
    Bloque de código
FIN_MIENTRAS
```



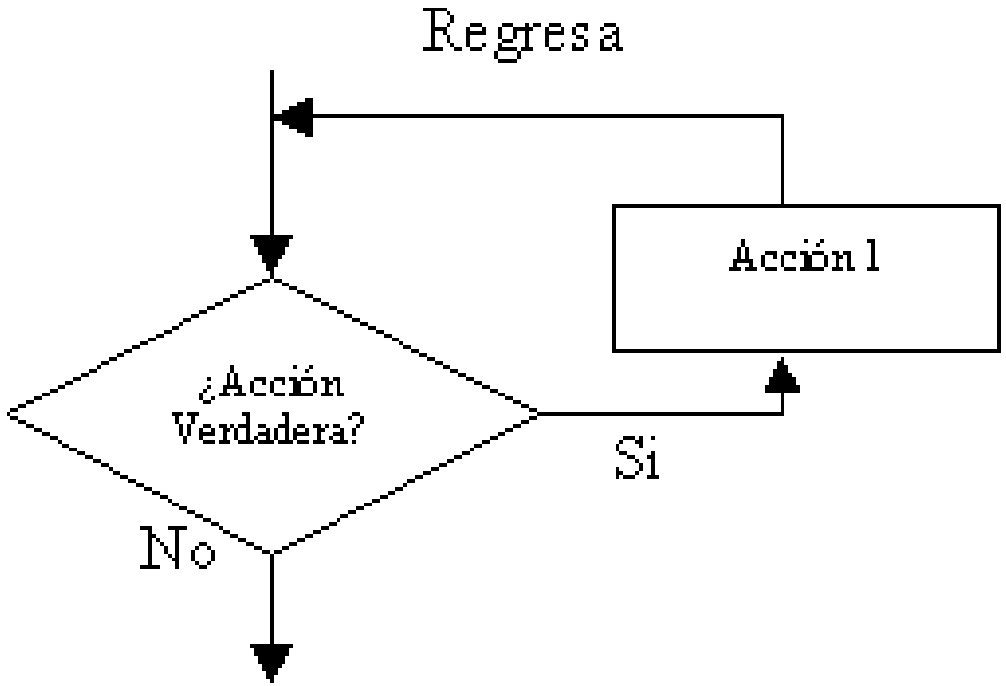
PROGRAMACIÓN ORIENTADA A OBJETOS.

Programación estructurada - Estructuras de control.

Estructura mientras

```
while condicion:
    cuerpo del bucle
```

```
i = 1

while i <= 3:
    print(i)
    i += 1

print("Programa terminado")
```



MIENTRA condición HACER:
    Bloque de código
FIN_MIENTRAS

PROGRAMACIÓN ORIENTADA A OBJETOS.

Programación estructurada - Estructuras de control.

Estructura mientras

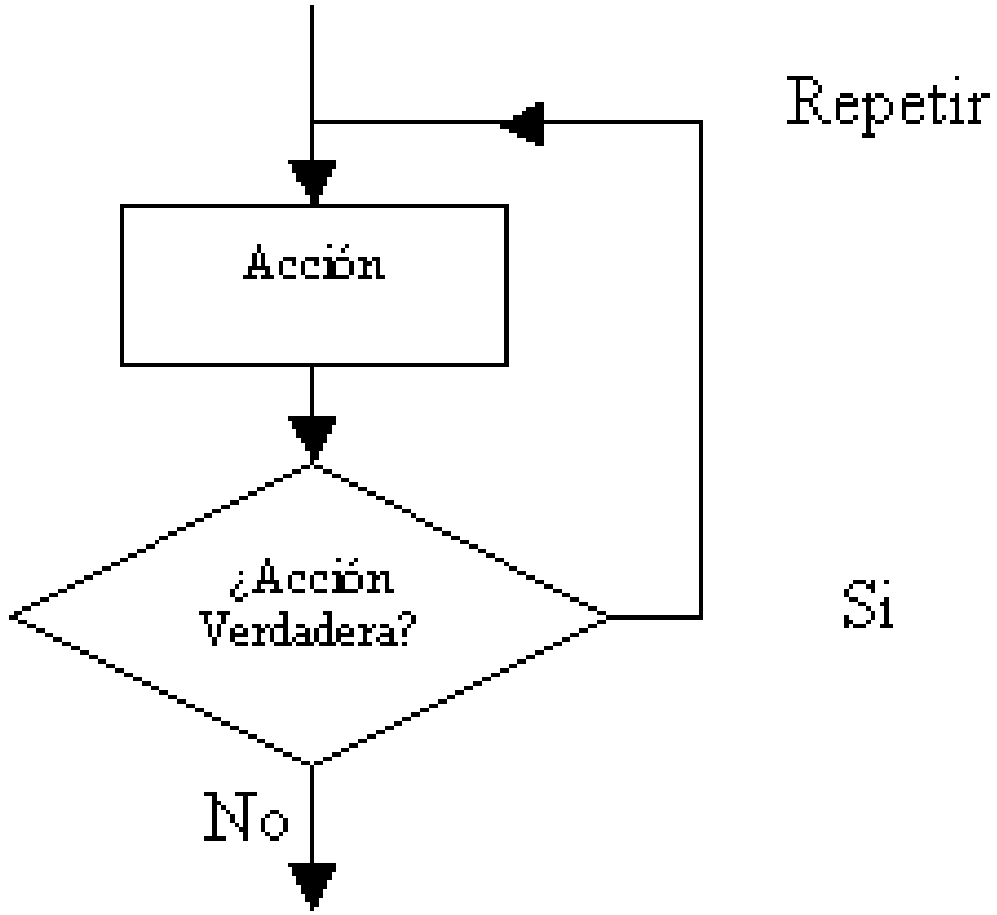
```
numero = int(input("Escriba un número positivo: "))

while numero < 0:
    print("¡Ha escrito un número negativo! Inténtelo de nuevo")
    numero = int(input("Escriba un número positivo: "))

print("Gracias por su colaboración")
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Estructura repetir

Se ejecuta hasta que se cumpla una condición determinada que se comprueba al final del bucle, esto permite que la iteración se ejecute al menos una vez antes de que la condición sea evaluada.

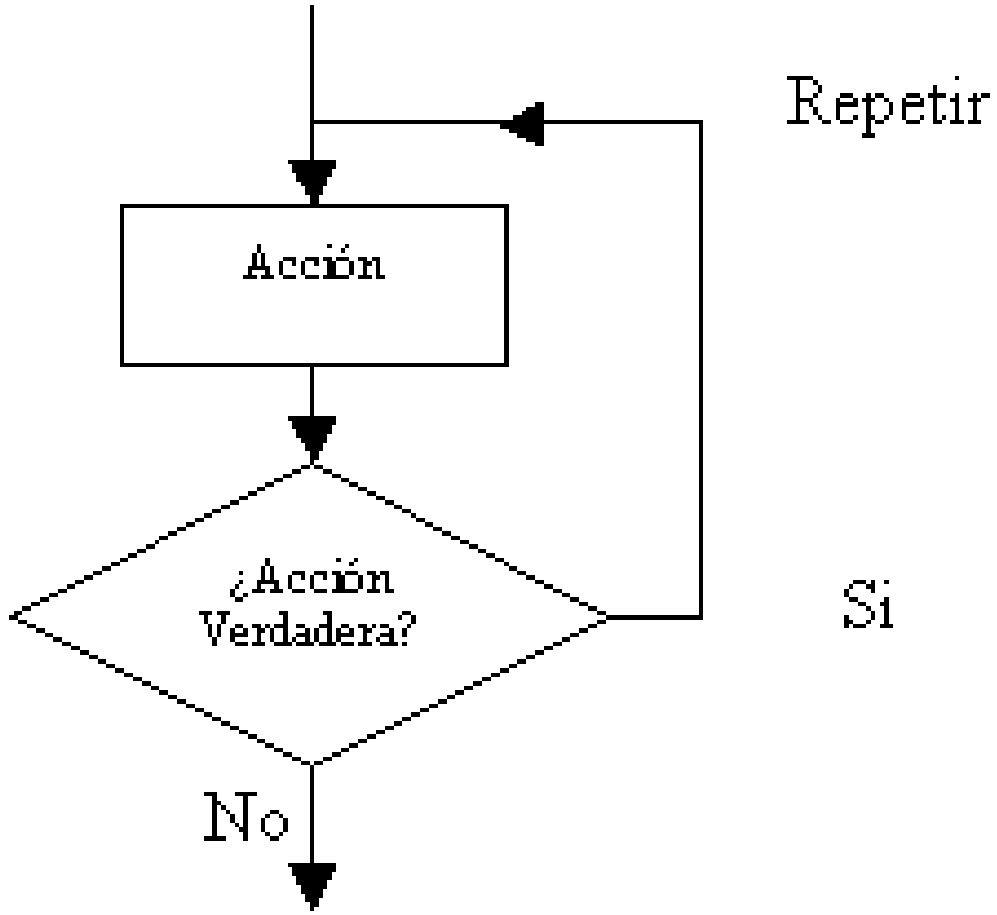


INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Estructura repetir

```
REPETIR:
    Bloque de código
HASTA condición
```

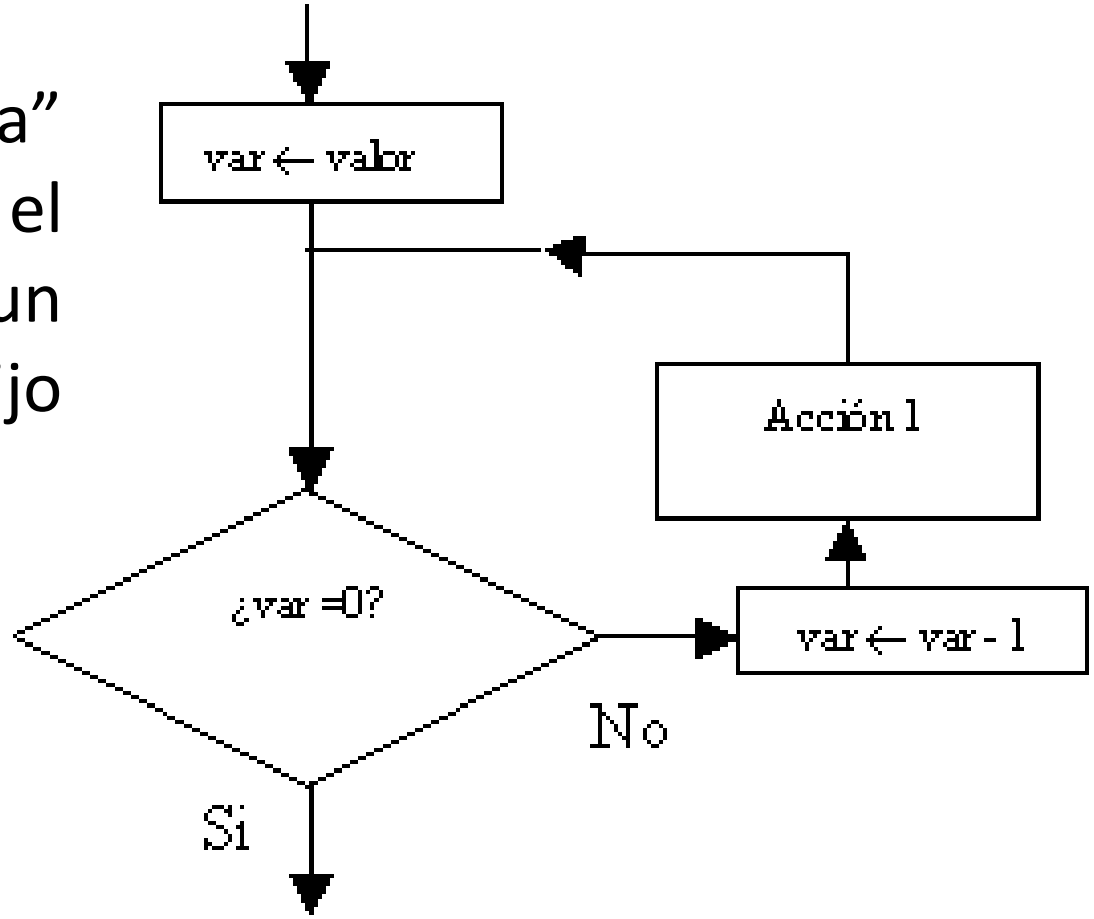


INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Estructuras desde/para

Se utilizan las estructuras “desde/para” cuando se conocen con certeza el número de veces que desea repetir un bucle, es decir, cuando es un número fijo de veces.

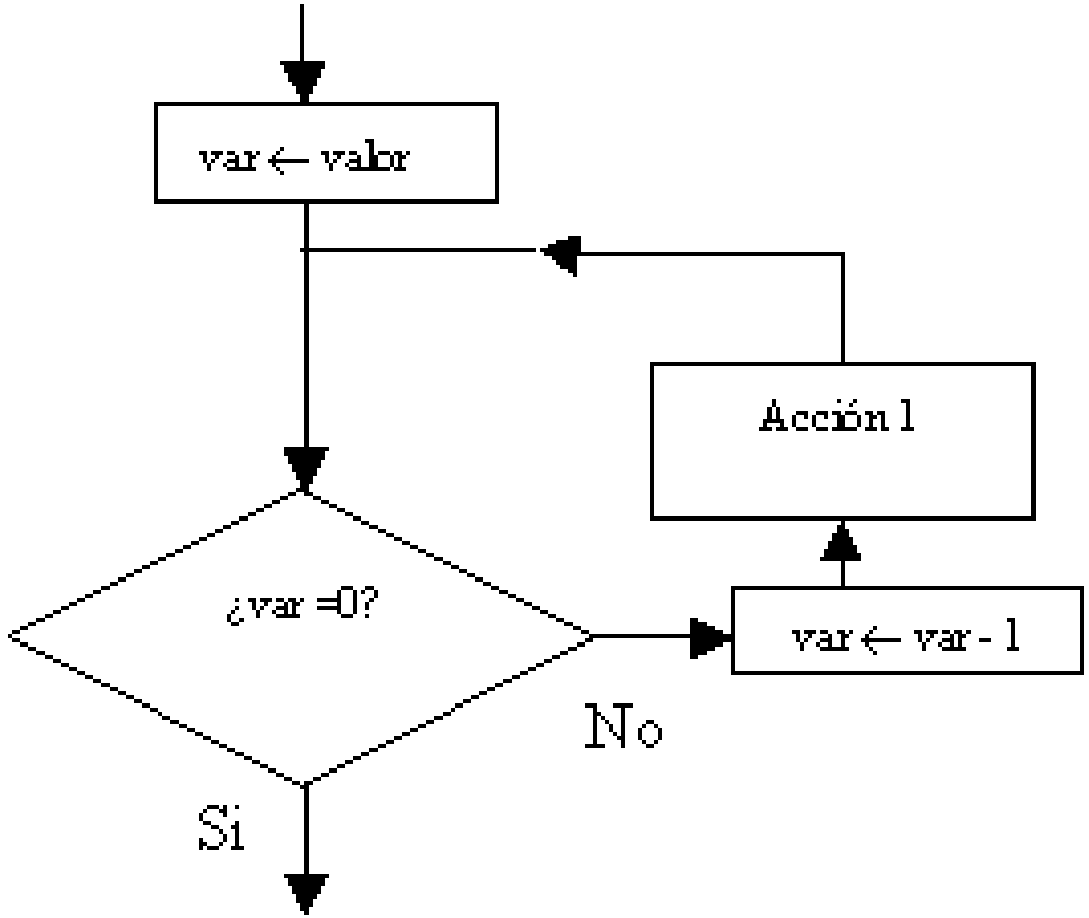


INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Estructuras desde/para

```
PARA i = 10 MIENTRAS i != 0 CON i = i-1
HACER:
    Bloque de código
FIN_PARA
```



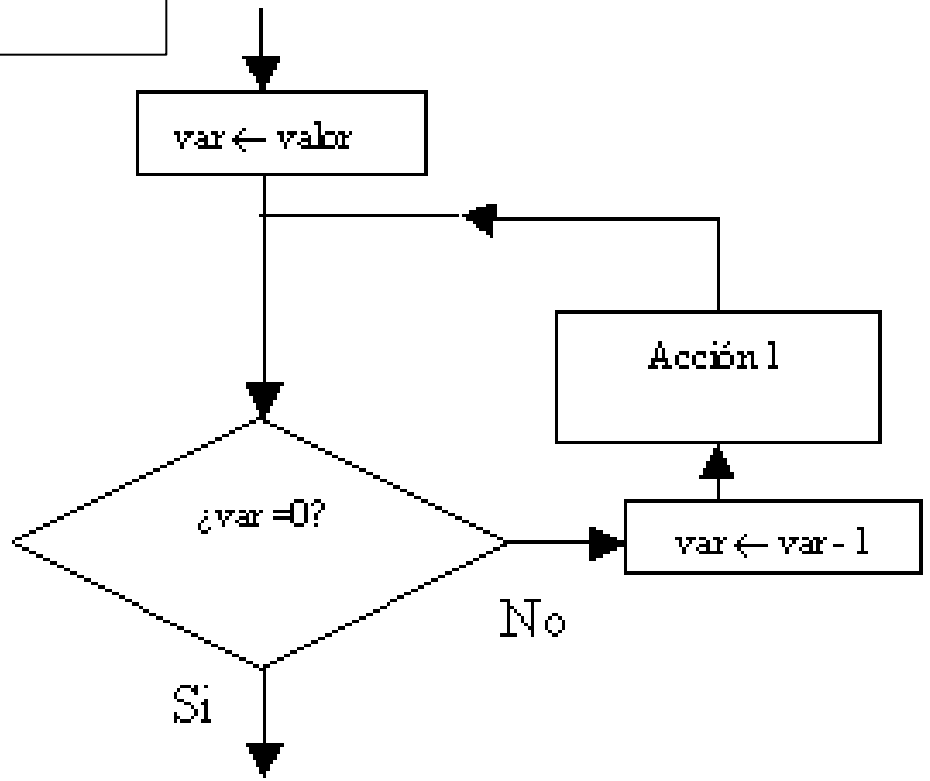
INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Estructuras desde/para

```
for variable in elemento iterable (lista, cadena, range, etc.):
    cuerpo del bucle
```

```
print("Comienzo")

for i in ["Alba", "Benito", 27]:
    print(f"Hola. Ahora i vale {i}")

print("Final")
```



INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

Estructuras desde/para

```
veces = int(input("¿Cuántas veces quiere que le salude? "))

for i in range(veces):
    print("Hola ", end="")

print()
print("Adiós")
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada.

Bucles anidados

Se habla de bucles anidados cuando un bucle se encuentra en el bloque de instrucciones de otro bloque.

Al bucle que se encuentra dentro del otro se le puede denominar bucle interior o bucle interno. El otro bucle sería el bucle exterior o bucle externo.

Los bucles pueden tener cualquier nivel de anidamiento (un bucle dentro de otro bucle dentro de un tercero, etc.).

Al escribir bucles anidados, hay que prestar atención al sangrado de las instrucciones, ya que ese sangrado indica a Python si una instrucción forma parte de un bloque u otro.

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada.

Bucles anidados

En el ejemplo, el bucle externo (el controlado por i) se ejecuta 3 veces y el bucle interno (el controlado por j) se ejecuta dos veces por cada valor de i. Por ello la instrucción print() se ejecuta en total 6 veces (3 veces que se ejecuta el bucle externo x 2 veces que se ejecuta cada vez el bucle interno = 6 veces).

En general, el número de veces que se ejecuta el bloque de instrucciones del bucle interno es el producto de las veces que se ejecuta cada bucle.

```
for i in range(3):
    for j in range(2):
        print(f"i vale {i} y j vale {j}")
```

La costumbre más extendida es utilizar la letra **"i"** como nombre de la variable de control del bucle externo y la letra **"j"** como nombre de la variable de control del bucle interno (o **"k"** si hay un tercer nivel de anidamiento), pero se puede utilizar cualquier otro nombre válido.

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada.

range()

range() genera una secuencia de números que van desde 0 por defecto hasta el número que se pasa como parámetro menos 1. Se pueden pasar hasta tres parámetros separados por coma, donde el primer es el **inicio** de la secuencia, el segundo el **final** y el tercero el **salto** que se desea entre números. Por defecto se empieza en 0 y el salto es de 1.

range(inicio, fin, salto)

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada.

range()

range(5,20,2) #se generarán números de 5 a 20 de dos en dos.

```
for i in range(5, 20, 2):
    print(i) #5,7,9,11,13,15,17,19
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada.

range()

Se pueden generar también secuencias inversas, empezando por un número mayor y terminando en uno menor, pero para ello el salto deberá ser negativo.

```
for i in range (5, 0, -1):
    print(i) #5,4,3,2,1
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada.

Sentencia break

La sentencia break nos permite alterar el comportamiento de los bucles **while** y **for**. Concretamente, permite **terminar con la ejecución del bucle**.

Esto significa que una vez se encuentra la palabra **break**, el bucle se habrá terminado. La sentencia **break** rompe el flujo dentro del cuerpo de instrucciones del bucle para abandonarlo.

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada.

Sentencia break

En el ejemplo se busca una letra en una palabra. Se itera toda la palabra y en el momento en el que se encuentra la letra que buscábamos, se rompe el bucle y se sale.

```
cadena = 'Python’

for letra in cadena:

    if letra == 'h':
        print("Se encontró la h")
        break

    print(letra)
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada.

Sentencia break

La condición ***while True*** haría que la sección de código se ejecutara indefinidamente, pero al hacer uso del **break**, el bucle se romperá cuando x valga cero.

Por norma general, y salvo casos muy concretos, si ves un **while True**, es probable que haya un **break** dentro del bucle.

```
x = 5

while True:
    x -= 1
    print(x)

    if x == 0:
        break

    print("Fin del bucle")
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada.

Sentencia break

Como hemos dicho, el uso de **break** rompe el bucle, pero sólo aquel en el que está dentro.

Es decir, si tenemos dos bucles anidados, el **break romperá el bucle anidado**, pero no el exterior.

```
for i in range(0, 4):
    # El break no afecta a este for

    for j in range(0, 4):
        break
      #Nunca se realiza más de una iteración

    print(i, j)
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada.

Sentencia continue

El uso de **continue** al igual que el ya visto **break**, nos permite modificar el comportamiento de los bucles **while** y **for**.

Concretamente, **continue** se salta todo el código restante en la iteración actual y **vuelve al principio** en el caso de que aún queden iteraciones por completar.

La diferencia entre el **break** y **continue** es que el **continue no rompe el bucle**, si no que **pasa a la siguiente iteración saltando el código pendiente.**

Gracias a **continue** podemos forzar la conclusión de una iteración para continuar con la siguiente.

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada.

Sentencia continue

En el siguiente ejemplo podemos ver como cuando la x vale 3, se llama al **continue**, lo que hace que se salte el resto de código de la iteración (el ***print()***). Por ello, vemos como el número 3 no se imprime en pantalla.

```
x = 5

while x > 0:
    x -= 1

    if x == 3:
        continue

    print(x)
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada.

Programación Estructurada

Conjunto de técnicas para desarrollar algoritmos fáciles de escribir, verificar, leer y modificar.

INTRODUCCIÓN A LA PROGRAMACIÓN

Programación estructurada - Estructuras de control.

"Fin del tema"