PROGRAMACIÓN Y MÉTODOS NUMÉRICOS
2503506

ESTRUCTURAS DE REPETICIÓN

Andrés Agudelo
Departamento de Ingeniería Mecánica
andres.agudelos@udea.edu.co

UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

Contenido

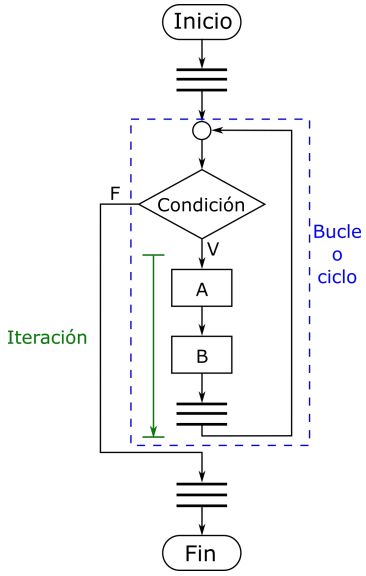
Estructuras de repetición

Estructuras de repetición

Se usan cuando se debe repetir un grupo de instrucciones mientras se cumpla alguna condición.

- Son frecuentes → Los computadores están diseñados especialmente para ejecutar tareas que se deben repetir muchas veces.
- La condición se establece en términos de una variable de control, mediante operadores de relación y operadores lógicos. → Rompimiento del ciclo.
- Se conocen como bucles o ciclos → Una vez se termina de ejecutar la secuencia de instrucciones, se regresa al inicio de la estructura para verificar la condición y repetir el proceso: ⟳
- Una ejecución completa (de principio a fin) de la secuencia de instrucciones se denomina iteración.

Estructuras de repetición



Estructuras de repetición

Variables de control

Variables numéricas que permiten controlar la repetición.

- **Conteo:**
  Sirven para contar la ocurrencia de un evento mediante incrementos o disminuciones constantes:

  $$\texttt{contador} \leftarrow \texttt{contador} \pm \texttt{constante}$$

- **Acumulación:**
  Contienen un valor que cambia en cada iteración, acumulando cantidades (suma/resta) que pueden ser variables:

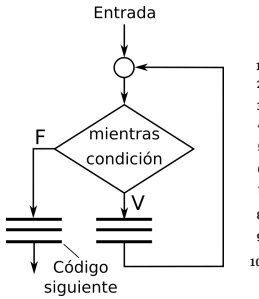
  $$\texttt{suma} \leftarrow \texttt{suma} \pm \texttt{valor}$$

Estructura **mientras** (**while**)

Estructura mientras (*while*)

En ésta el cuerpo del ciclo se ejecuta siempre que la condición del bucle sea verdadera. Por lo tanto, es posible que no haya iteraciones si la condición es falsa.

El número de veces que se repetirá el ciclo no está definido previamente.
⇒ Riesgo de que el bucle se ejecute indefinidamente.



algoritmo Estructura Mientras

var Definición de las variables del problema

```
1  inicio
2  |  ...
3  |  mientras Condición haga
4  |  |  Instrucción 1
5  |  |  Instrucción 2
6  |  |  ...
7  |  |  Instrucción n
8  |  fin mientras
9  |  Código siguiente
10 fin
```

```
while Condición:
    Instrucción 1
    Instrucción 2
    ...
    Instrucción n

Código siguiente
```


Estructura **mientras**

`mientras` condición **`haga`**
  Bloque de instrucciones
`fin mientras`

Funcionamiento

1. Al comienzo de cada iteración se evalúa la expresión lógica o condición.
2. Si el resultado es verdadero, se ejecuta el conjunto de instrucciones y se vuelve a iterar → Regreso automático al paso 1.
3. Si el resultado es falso, se omite la ejecución del ciclo mientras, y el programa se sigue ejecutando por la instrucción siguiente al fin mientras.

Estructura **mientras**

Ejemplo 1

Escribir de forma ascendente los 100 primeros números naturales.

algoritmo Números naturales hasta 100

var entero: n

```
1  inicio
2      n ← 1
3      mientras n <= 100 haga
4          escribir n
5          n ← n + 1
6      fin mientras
7  fin
```

```
n = 1
while n <= 10:
    print(n)
    n = n + 1
```

```
n = 1
while n <= 10:
    print(n)
    n += 1
```

`+=` `-=` `*=` `/=`

Estructura **mientras**

Ejemplo 2

Diseñar un algoritmo que calcule y muestre la suma de una determinada cantidad de números que se ingresan de forma secuencial.
Realice la prueba de escritorio con los siguientes números:

30, 10, 2, 7, 15, 100, 50

Análisis:
Se requiere una variable que acumule la suma de los números y otra que determine el número de valores sumados, para que cuando sea igual a la cantidad de valores deseados, termine el proceso de lectura.

Cantidad de números a sumar → `num_val`
Cada número leído → `Num`
Acumulador → `sum_Num`
Contador → `cont_Num`

Estructura **mientras**

Ejemplo 2

Algoritmo para sumar una cantidad determinada de números.

algoritmo Suma de *n* números naturales

Entradas: real: Num entero: num_val
Intermedias: entero: cont_Num %Contador
Salidas: entero: num_val
real: sum_Num %Acumulador

```
inicio
1      leer num_val
2      Cont_Num ← 1                    %Inic. contador
3      sum_Num ← 0                     %Inic. acumulador
4      mientras Cont_Num <= num_val haga
5          leer Num
6          sum_Num ← sum_Num + Num     %Acum. Num
7          Cont_Num ← Cont_Num + 1     %Incr. contador
8      fin mientras
9      escribir 'La suma de los',num_val,'números es:',sum_Num
10 fin
```

Prueba de escritorio

⇓

Implementación en Python

Estructura **mientras**

Ejemplo 2

Prueba de escritorio: 30, 10, 2, 7, 15, 100, 50 (7 números)

num_val	Cont_Num	Num	sum_Num	Pantalla
7				
7	1		0	
7	1	30	30	
7	2	10	40	
7	3	2	42	
7	4	7	49	
7	5	15	64	
7	6	100	164	
7	7	50	214	
7	8			La suma de los 7 números es: 214

Estructura **mientras**

Ejemplo 3

La velocidad de un paracaidista de masa m que parte del reposo, el cual cae con un coeficiente de arrastre C, se determina según la siguiente relación:

$$V(t) = \frac{mg}{C}\left[1 - e^{-(\frac{C}{m})t}\right] \quad (1)$$

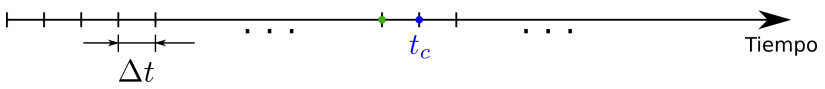
Una aproximación numérica a esta solución se puede obtener mediante la siguiente expresión:

$$V_a(t + \Delta t) = V_a(t) + \left(g - \frac{C\ V_a(t)}{m}\right)\Delta t \quad (2)$$

Diseñe un algoritmo que determine la velocidad mediante la ecuación 2 para un tiempo final y un paso temporal determinados, así como los errores absoluto y relativo de la aproximación.

Estructura **mientras** – Ejemplo 3

$$V(t) = \frac{mg}{C}\left[1 - e^{-(\frac{C}{m})t}\right]$$



$$V_a(t + \Delta t) = V_a(t) + \left(g - \frac{C\ V_a(t)}{m}\right)\Delta t$$

Estructura **mientras**

Ejemplo 3 – Análisis

Se requiere usar las ecuaciones (1) y (2) para calcular las velocidades exacta (`V`) y aproximada (`V_a`), respectivamente.

$$V(t) = \frac{mg}{C}\left[1 - e^{-(\frac{C}{m})t}\right] \quad (1)$$

En la ecuación (1) hace falta conocer la masa del paracaidista (`m`), el coeficiente de arrastre (`C`), y el instante para el cuál se debe realizar el cálculo: tiempo de caída (`t_c`).

Adicionalmente, se usa una variable para representar la aceleración gravitacional (`g`).

Estructura **mientras**

Ejemplo 3 – Análisis

$$V_a(t + \Delta t) = V_a(t) + \left(g - \frac{C\ V_a(t)}{m}\right)\Delta t \quad (2)$$

En la ecuación (2) se debe conocer además el paso temporal (`Dt`) que se usará para calcular la velocidad aproximada (`V_a`).

Para calcular la velocidad aproximada se debe comenzar en un valor inicial (conocido), y avanzar en el tiempo (desde cero), usando una variable (`t`) que permita incrementar la cantidad Δt (`Dt`) hasta llegar al instante de interés (`t_c`).

Estructura **mientras**

Ejemplo 3 – Análisis

Los errores absoluto y relativo se definen de la siguiente forma:

$$\texttt{error\_abs} = \texttt{V} - \texttt{V\_a} \quad [m/s]$$

$$\texttt{error\_rel} = 100 \left| \frac{\texttt{V} - \texttt{V\_a}}{\texttt{V}} \right| \quad [\%]$$

Variables:

- Entradas: `m`, `C`, `t_c`, `Dt`
- Salidas: `V`, `V_a`, `error_abs`, `error_rel`
- Intermedias: `t`, `g`

Estructura **mientras**

Ejemplo 3 – Pasos

1. Inicializar el tiempo de cálculo: `t` $\leftarrow 0$
2. Inicializar la velocidad aproximada: `V_a` $\leftarrow 0$
3. Asignar el valor de la aceleración gravitacional: `g` $\leftarrow 9.81$
4. Leer las entradas: `m`, `C`, `t_c`, `Dt`
5. Calcular la velocidad exacta en `t_c` usando la ecuación (1)
6. Calcular la velocidad aproximada usando la ecuación (2):
   $\Rightarrow$ Proceso iterativo que comienza en `t` $= 0$ y va hasta `t` $=$ `t_c`, avanzando `Dt` en cada paso o iteración.
7. Una vez termina la iteración, el valor final de `V_a` será la aproximación de la velocidad de caída en el instante `t_c`, por lo tanto se procede a calcular los errores absoluto y relativo.
8. Se escriben mensajes que muestren los resultados: velocidades exacta y aproximada en el instante `t_c`, así como los errores.

Estructura **mientras**

Ejemplo 3 – Caída de un paracaidista

algoritmo Aproximación a la velocidad de un paracaidista

Entradas: real: `m, C, t_c, Dt`
Intermedias: real: `t, g`
Salidas: real: `V, V_a, error_abs, error_rel`

```
1  inicio
2      t ← 0                                          %Inicialización del tiempo de cálculo
3      V_a ← 0                                        %Valor inicial de la velocidad
4      g ← 9.81                                       %Aceleración gravitacional [m/s2]
5      leer m, C, t_c, Dt
6      V ← ((m*g)/C)*(1 - exp(-(C/m)*t_c))            %Velocidad exacta al cabo del tiempo t_c [m/s]
7      mientras t < t_c haga
8          t ← t + Dt                                 %Incremento del contador de tiempo
9          V_a ← V_a + (g - ((C*V_a)/m))*Dt           %Aprox. al valor de la velocidad [m/s]
10     fin mientras
11     error_abs ← V - V_a
12     error_rel ← 100 * |error_abs / V|
13     escribir 'La velocidad exacta a los ',t_c,' s es: ',V,' m/s'
14     escribir 'La velocidad aproximada a los ',t_c,' s es :',V_a,' m/s'
15     escribir 'El error absoluto es : ',error_abs,' m/s'
16     escribir 'El error relativo es :',error_rel,' %'
17 fin
```


Estructura **mientras**

Ejemplo 3

1. Realice la prueba de escritorio con los siguientes valores:

$$m = 68.1\ kg$$
$$C = 12.5\ Ns/m$$
$$g = 9.81\ m/s^2$$
$$t_c = 10\ s$$
$$\Delta t = 2\ s$$

2. Realice el programa en Python, usando los mismos valores de m, C, g, t_c, y los siguientes valores de Δt:
   - 0.5 s
   - 2 s
   - 5 s

   ¿Cómo se comportan los errores al cambiar el valor del incremento temporal para la solución numérica?

Estructura **mientras**

Interrupción de bucles

En ocasiones es necesario interrumpir la ejecución de un ciclo de repetición en algún punto interno del bloque de instrucciones que se repiten.

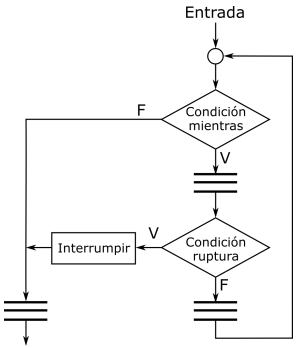
$$\Downarrow$$

Dependerá de que se cumpla o no alguna condición.

La interrupción puede hacerse de dos formas:

1. Abandonando el ciclo de repetición definitivamente.
   → `interrumpir`
2. Abandonando la iteración en curso, pero comenzando la siguiente.
   → `continuar`

Estructura **mientras**



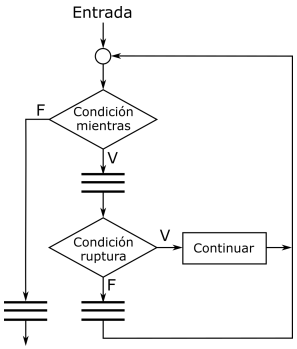
algoritmo Interrupción del flujo

Intermedias: Declaración de variables

```
1  inicio
2      ...
3      mientras Condición haga
4          Instrucciones V
5          si Condición ruptura entonces
6              interrumpir
7          fin si
8          Instrucciones F
9      fin mientras
10     ...
11 fin
```

```
while Condición:
    Instrucciones V
    if Condición ruptura:
        break
    Instrucciones F
```


Estructura **mientras**



algoritmo Continuación del flujo

Intermedias: Declaración de variables

```
1  inicio
2      ...
3      mientras Condición haga
4          Instrucciones V
5          si Condición ruptura entonces
6              continuar
7          fin si
8          Instrucciones F
9      fin mientras
10     ...
11 fin
```

```
while Condición:
    Instrucciones V
    if Condición ruptura:
        continue
    Instrucciones F
```


Ejemplo 4

Ejemplo 4 – Estructura mientras con interrupción

Diseñar un algoritmo, a partir del ejemplo 2, el cual calcule y muestre la suma de una determinada cantidad de números que se ingresan de forma secuencial.

En este caso, la suma de los números no debe superar el valor de 150. Si esto sucede, se debe detener el proceso y reportar la suma parcial de los números cuya suma es menor a 150, informando cuántos números se alcanzaron a sumar.

Realice la prueba de escritorio con los siguientes números:

30, 10, 2, 7, 15, 100, 50

Ejemplo 4

Ejemplo 4 – Estructura mientras con interrupción

Análisis:

Cada que se ingrese un número, éste se acumulará en la suma, y en este punto se deberá verificar que ésta no supere el valor establecido en cada caso. Cuando esto suceda, se debe interrumpir la estructura mientras.

Con respecto al problema original, se debe crear una variable nueva:
Velocidad límite → `V_lim`

Se debe crear una condición de ruptura (estructura de decisión simple), que permita interrumpir el ciclo, en caso de que sea necesario:

$$\texttt{sum\_Num} \geq \texttt{V\_lim}$$

Ejemplo 4

algoritmo Suma de *n* números con valor límite

```
Entradas:      real:   Num, V_lim          entero:   num_val
Intermedias:   entero: cont_Num
Salidas:       entero: num_val
               real:   sum_Num                                  %Acumulador
inicio
1   Cont_Num ← 1                                                %Inic. contador
2   sum_Num ← 0                                                 %Inic. acumulador
3   leer num_val, V_lim
4   mientras Cont_Num <= num_val haga
5       leer Num
6       sum_Num ← sum_Num + Num                                 %Acum. Num
7       si sum_Num >= V_lim entonces
8           escribir 'Se sumaron', cont_Num − 1, 'números sin superar el valor de', V_lim
9           escribir 'El resultado de la suma parcial es:', sum_Num - Num
10          interrumpir
11      fin si
12      Cont_Num ← Cont_Num + 1                                 %Incr. contador
13  fin mientras
14  si Cont_Num == num_val + 1 entonces
15      escribir 'La suma de los ',num_val,' números es: ',sum_Num
16  fin si
17 fin
```

Prueba de escritorio ⇒ Implementación en Python

Estructura **mientras**

Bucles infinitos

Se dan cuando, sin intención, se define mal la condición de parada.

Se trata de un error frecuente con la estructura mientras. Es indeseable, ya que el programa se estanca.

Ejemplo

Algoritmo para calcular el beneficio de ahorrar un capital determinado, con intereses desde el 10 % y por debajo del 20 % anual.

algoritmo Interés

```
var     real:   tasa, capital, interes
1 inicio
2     tasa ← 0.1
3     mientras tasa ∼= 0.2 haga
4         interes ← tasa * capital
5         escribir 'Interés producido:', interes
6         tasa ← tasa + 0.02
7     fin mientras
8 fin
```

`tasa` ⇒ [0.1, 0.12, 0.14, 0.16, 0.18, 0.20]

`tasa` ← `tasa` + 0.03

`tasa` ⇒ [0.1, 0.13, 0.16, 0.19, 0.22, 0.25 . . .]

¡Bucle infinito!

$\texttt{tasa} < 0.2$

Estructura **mientras**

Bucles infinitos

Regla práctica:

Es conveniente que los operadores de comparación usados en las condiciones de las estructuras de repetición sean de mayor o menor que (o también mayor/menor o igual que), y **no** de igualdad o desigualdad.

En la codificación en un lenguaje de programación, esta regla debe seguirse estrictamente en el caso de comparación de números reales, ya que como esos valores se almacenan en cantidades aproximadas, las comparaciones de igualdad de valores reales normalmente plantean problemas.

Comparación de números reales

⇓

$<, >$ ó $<=, >=$

Estructura **mientras**

Posibilidad

Usar una estructura de decisión para interrumpir el bucle mientras.

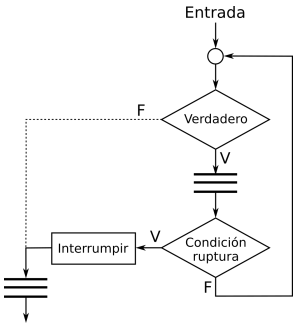
$$\Downarrow$$

Se garantiza por lo menos la primera ejecución del bucle.

Consiste en crear una condición inicial que siempre sea verdadera, y preguntarse durante la ejecución del bucle si se cumple una condición específica para interrumpir su ejecución.

Esta es una práctica que da mayor control, reduciendo mucho la posibilidad de definir mal la condición inicial y caer en un bucle infinito.

Estructura **mientras**




```
algoritmo Estructura Mientras
var    Definición de las variables del problema
inicio
    ...
    mientras Verdadero haga
        Instrucciones
        si Condición de ruptura entonces
            interrumpir
        fin si
    fin mientras
    ...
fin
```

```
while True:
    Instrucciones
    if Condición ruptura:
        break
    ...
```


Estructura **para** (*for*)

Estructura **para**

Permite implementar la repetición de un cierto conjunto de instrucciones un número predeterminado de veces.

Se utiliza una variable de control del bucle, llamada también índice, que va recorriendo un conjunto prefijado de valores en un orden determinado.

Para cada valor del índice en dicho conjunto, se ejecuta una vez el mismo conjunto de instrucciones.

Al finalizar el bloque de instrucciones (iteración) se actualiza el valor del índice y se regresa al inicio del bucle:

- Si no se especifica un incremento específico para el índice, se aumentará en uno (1) su valor al final de cada iteración.
- Se puede establecer un incremento diferente a la unidad, con lo cual se aumentará el índice en este valor, al final de cada iteración.

Estructura **para** (*for*)

Estructura **para**

Al implementar una estructura repetitiva de este tipo, se debe iniciar con la palabra **para**, seguida de los parámetros de la estructura, y se debe terminar con la instrucción **fin para**.

Los parámetros de la estructura determinan cuántas veces se repetirá el conjunto de instrucciones que la componen, y consisten en la definición del nombre y los valores que tomará la variable de control durante las iteraciones.

Variable de control o índice (`var_cont`):

Su nombre suele ser una letra que identifica el índice cuyo valor cambia después de cada iteración. Los nombre más comunes son `i`, `j`, `k`, `m`, `n`. El valor de la variable de control se suele usar en las instrucciones de la estructura **para**.

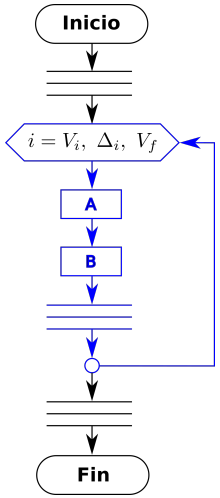
Estructura **para** (*for*)

Estructura **para**

Al implementar una estructura repetitiva **para**, se debe especificar lo siguiente:

- **Valor inicial de la variable de control** (`val_ini`, V_i)**:**
  Indica el valor de partida del índice, el cual se usa en la primera iteración.
- **Valor del incremento del índice** (`incr`, Δ_i)**:**
  Este valor indica en cuánto aumentará el valor del índice o variable de control entre dos iteraciones sucesivas. Si no se especifica su valor, se toma de forma predeterminada como la unidad.
- **Valor final de la variable de control** (`val_fin`, V_f)**:**
  Indica el valor final que tomará la variable de control. Cuando ésta alcance este valor, se terminará la repetición de las instrucciones y se continuará con el algoritmo.

Estructura **para**



algoritmo Estructura Para

Entradas: Definición variables de entrada
Intermedias: Definición de variables intermedias
Salidas: Definición de variables de salida

```
1 inicio
2     ...
3     para var_cont = val_ini incremento incr hasta val_fin haga
4         Instrucción A
5         Instrucción B
6         ...
7     fin para
8     ...
9 fin
```

Incremento automático de una unidad en cada iteración:

para `var_cont` = `val_ini` **hasta** `val_fin` **haga**

Python → `range`

Estructura **para**

Ejemplo 5

Un curso está compuesto por *n* alumnos, a los cuales se les hicieron tres evaluaciones.

Se debe realizar un algoritmo que escriba el promedio de las 3 notas para cada uno de ellos, con sus datos básicos (código y nombre), así como el promedio de las notas de todos los alumnos del curso.

Realice la prueba de escritorio con los siguientes valores:

Código	Nombre	Notas
A20190201	Eric Clapton	4.5, 4.8, 5.0
A20190202	Nicky Jam	1.5, 1.8, 2.0
A20190203	Carlos Vives	3.5, 4.0, 4.0

Estructura **para**

Ejemplo 5 – Análisis:

Los datos básicos son el código (`cod`) y el nombre (`nombre`).
Se deben leer estos datos para cada estudiante, además de las 3 notas (`nota_1`, `nota_2`, `nota_3`).

También se necesita conocer el número de estudiantes (`n`), lo mismo que calcular el promedio de las 3 notas por cada estudiante (`prom`).

Finalmente, se deben acumular los promedios de cada alumno, con el fin de obtener la sumatoria (`suma`) y el promedio del curso (`prom_T`).

Adicionalmente, al recorrer la lista de estudiantes mediante la estructura **para**, se requiere definir el índice. ⇒ Se usará la variable `i` como contador de alumnos.

Estructura **para**

Ejemplo 5 – Análisis:

Variables de entrada:

- Número de estudiantes → `n` (entero)
- Datos básicos de cada estudiante → `cod`, `nombre` (string)
- Notas de cada estudiante → `nota_1`, `nota_2`, `nota_3` (real)

Variables intermedias:

- Contador de alumnos (variable de control) → `i` (entero)
- Suma de las notas del curso → `suma` (real)

Variables de salida:

- Datos básicos de cada estudiante → `cod`, `nombre`
- Promedio de cada estudiante → `prom` (real)
- Promedio de las notas del curso → `prom_T` (real)

Estructura **para** – Ejemplo 5

algoritmo Promedio de las notas de los estudiantes y del curso

```
Entradas:     entero:  n ;                                  %Número de alumnos del curso
                real:  nota_1, nota_2, nota_3 ;             %Notas de cada alumno
              string:  cod, nombre ;                        %Datos de cada alumno
Intermedias:  entero:  i ;                                  %Contador de alumnos
                real:  suma ;                               %Suma de todos los promedios del curso
Salidas:      string:  cod, nombre
                real:  prom ;                               %Promedio individual
                       prom_T ;                             %Promedio total del curso
inicio
1   suma ← 0;                                               %Inicializa el acumulador de promedios
2   leer n
3   para i = 1 hasta n haga
4       leer cod, nombre
5       leer nota_1, nota_2, nota_3
6       prom ← (nota_1 + nota_2 + nota_3)/3
7       suma ← suma + prom ;                                %Acumula el valor del promedio del estudiante
8       escribir 'Código: ', cod,', Nombre: ', nombre,', Promedio: ', prom
9   fin para
10  prom_T ← suma/n
11  escribir 'El promedio del curso es: ', prom_T
12 fin
```


Estructura **para**

Ejemplo 5

Prueba de escritorio:

Código	Nombre	Notas
A20190201	Eric Clapton	4.5, 4.8, 5.0
A20190202	Nicky Jam	1.5, 1.8, 2.0
A20190203	Carlos Vives	3.5, 4.0, 4.0

Paso	suma	n	i	cod	nombre	nota_1	nota_2	nota_3	prom	prom_T	Pantalla
1	0										
2	0	3									
3	0	3	1								
4	0	3	1	A20190201	Eric Clapton						
5	0	3	1	A20190201	Eric Clapton	4.5	4.8	5.0			
6	0	3	1	A20190201	Eric Clapton	4.5	4.8	5.0	4.8		
7	4.8	3	1	A20190201	Eric Clapton	4.5	4.8	5.0	4.8		
8	4.8	3	1	A20190201	Eric Clapton	4.5	4.8	5.0	4.8		Código: A20190201, Nombre: Eric Clapton, Promedio: 4.8

Estructura **para**

Paso	suma	n	i	cod	nombre	nota_1	nota_2	nota_3	prom	prom_T	Pantalla
9	4.8	3	2	A20190201	Eric Clapton	4.5	4.8	5.0	4.8		
10	4.8	3	2	A20190202	Nicky Jam	4.5	4.8	5.0	4.8		
11	4.8	3	2	A20190202	Nicky Jam	1.5	1.8	2.0	4.8		
12	4.8	3	2	A20190202	Nicky Jam	1.5	1.8	2.0	1.8		
13	6.6	3	2	A20190202	Nicky Jam	1.5	1.8	2.0	1.8		
14	6.6	3	2	A20190202	Nicky Jam	1.5	1.8	2.0	1.8		Código: A20190202, Nombre: Nicky Jam, Promedio: 1.8
15	6.6	3	3	A20190202	Nicky Jam	1.5	1.8	2.0	1.8		
16	6.6	3	3	A20190203	Carlos Vives	1.5	1.8	2.0	1.8		
17	6.6	3	3	A20190203	Carlos Vives	3.5	4.0	4.0	1.8		
18	6.6	3	3	A20190203	Carlos Vives	3.5	4.0	4.0	3.8		
19	10.4	3	3	A20190203	Carlos Vives	3.5	4.0	4.0	3.8		
20	10.4	3	3	A20190203	Carlos Vives	3.5	4.0	4.0	3.8		Código: A20190203, Nombre: Carlos Vives, Promedio: 3.8
21	10.4	3	4	A20190203	Carlos Vives	3.5	4.0	4.0	3.8		
22	10.4	3	4	A20190203	Carlos Vives	3.5	4.0	4.0	3.8	3.4	
23	10.4	3	4	A20190203	Carlos Vives	3.5	4.0	4.0	3.8	3.4	El promedio del curso es: 3.4

Anidamiento de estructuras de repetición

Anidamiento de estructuras de repetición

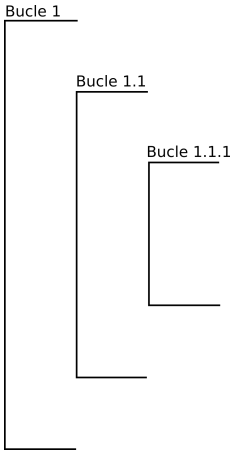
Las estructuras de repetición se pueden anidar, tal como sucede con las estructuras de selección.

$\Downarrow$

Insertar un bucle (interno) completamente dentro de otro bucle (externo).

- Para cada valor del índice o variable de control del bucle externo, los bucles más internos se ejecutan completamente.
- Se pueden anidar los diferentes tipos de estructuras repetitivas entre sí sin restricciones, siempre que cada bucle esté bien definido y no haya solapamiento entre bucles externos e internos.

Anidamiento de estructuras de repetición

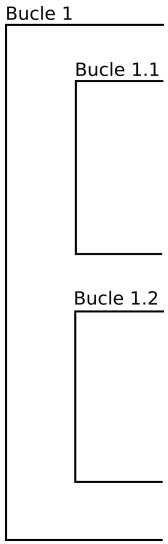


algoritmo Anidamiento de estructuras de repetición

Intermedias: Definición de las variables a usar.

```
1  inicio
2      ...
3      inicio bucle 1 condición 1 haga
4          ...
5          inicio bucle 1.1 condición 1.1 haga
6              ...
7              inicio bucle 1.1.1 condición 1.1.1 haga
8                  ...
9              fin bucle 1.1.1
10             ...
11         fin bucle 1.1
12         ...
13     fin bucle 1
14     ...
15 fin
```

Anidamiento de estructuras de repetición



algoritmo Anidamiento de estructuras de repetición

Intermedias: Definición de las variables a usar.

```
1  inicio
2      ...
3      inicio bucle 1 condición 1 haga
4          ...
5          inicio bucle 1.1 condición 1.1 haga
6              ...
7          fin bucle 1.1
8          ...
9          inicio bucle 1.2 condición 1.2 haga
10             ...
11         fin bucle 1.2
12         ...
13     fin bucle 1
14     ...
15 fin
```


Estructura **para**

Ejemplo 6

Realice un algoritmo que implemente las tablas de multiplicar del 1 al 10, usando la estructura **para**.
El algoritmo de mostrar la tabla correspondiente a cada número, identificándola con un mensaje ('Tabla del 5:')

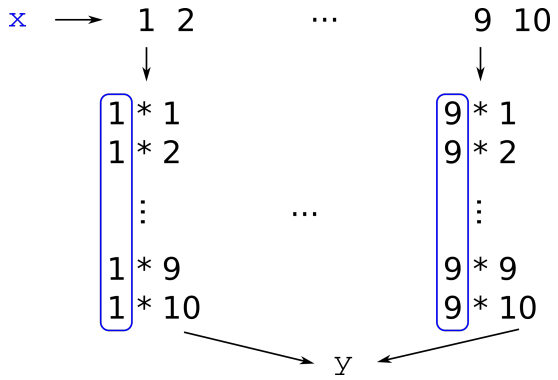
Análisis:
Problema: multiplicación de dos números. $\Rightarrow$ dos variables numéricas (enteros: `x`, `y`).

Cada número debe variar entre 1 y 10:

`x`, `y`: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 $\Rightarrow$ Dos ciclos **para**.

`x` y `y` sirven como variable de control de cada ciclo, ya que sus valores son enteros sucesivos.

Estructura **para**



Para cada valor de `x`, `y` debe recorrer todos sus valores (de 1 a 10)

$\Downarrow$

Bucle anidado.

Estructura **para** – Ejemplo 6

algoritmo Tablas de multiplicar del 1 al 10

Entradas: entero: `x, y`

Salidas: entero: `x, y`

```
1  inicio
2      para x = 1 hasta 10 haga
3          escribir 'Tabla del ', x, ':'
4          para y = 1 hasta 10 haga
5              escribir x, ' * ', y, ' = ', x * y
6          fin para
7      fin para
8  fin
```

Prueba de escritorio

$\Downarrow$

Implementación en Python

Ejercicios

Propuestos en la página web del curso

1. Estructura mientras: tiempo de actividad por rango de edad.
2. Velocidad de caída de un paracaidista, usando la estructura `para`.
3. Anidamiento de estructuras de repetición: nota final de un curso.

A continuación

Próxima clase

Estructuras de datos:

- Datos simples y estructurados.
- Cadenas de caracteres.
- Arreglos de datos: unidimensionales, multidimensionales.
- Registros (Listas y diccionarios)