GUÍA PARA SOBREVIVIR A PROGRAMACIÓN DE PRIMERO BIOTECNOLOGÍA

TEORÍA

Para comenzar a programar es importante tener clara la teoría de la algoritmia, y por ello, te ofrecemos esta breve, pero completa guía en la que podrás entender los conceptos más básicos. Así, más adelante, podrás poner en práctica el marco teórico. En la guía encontrarás el siguiente contenido:

Después, puedes poner a prueba tus conocimientos en los ejercicios y cuestionarios que hemos preparado para ti.

1. Introducción

Pero antes de empezar, ¿qué es un algoritmo?

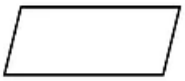
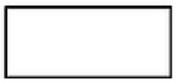
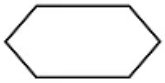
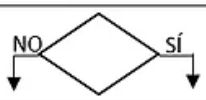
Un algoritmo (del griego y latín, *dixit algorithmus* y este a su vez del matemático persa Al-Juarismi) es un conjunto de instrucciones bien definidas, ordenadas y finitas que permite realizar una actividad.

En la vida cotidiana, empleamos algoritmos frecuentemente, por ejemplo, los manuales de usuario, que muestran algoritmos para usar un dispositivo, o las instrucciones que le da un jefe a sus empleados.

Para dar dichas instrucciones utilizamos **organigramas**, esquemas gráficos en los que se representa la sucesión de tareas como si se tratase de un flujo continuo. También se utiliza el **pseudo-código**, una descripción escueta y sintetizada de los organigramas.

En cuanto a los organigramas, es importante tener clara la **simbología**.

- Inicio organigrama: círculo con C en el interior
- Fin organigrama: círculo con F en el interior
- Flujo de información: flecha hacia abajo
- Entrada/salida de datos: romboide
- Operaciones: rectángulo
- Bucles: hexágono
- Condiciones: rombo
- Conectores: círculo

	conectores
	entrada/salida de datos
	operaciones
	bucles (estructuras repetitivas)
	condiciones
	flujo de información

2. Bucles

¿Qué es un bucle ?

Un cálculo que se repite varias veces se programa mediante una estructura denominada.

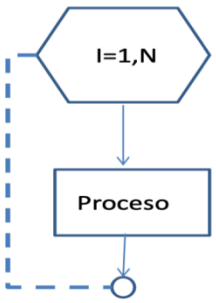
Los bucles más comunes son el ***while*** (ejecutar un trozo de código mientras la condición sea verdadera), ***for*** (en el que podemos marcar el número de veces que queremos que nuestro código se repita)

Elementos del bucle

- Variable de control (normalmente se utiliza la letra “i”)
- Si tenemos bucles anidados (bucles dentro de bucles) se utilizan las siguientes letras posteriores a la i
- Inicialización de la variable de control (valores iniciales que toma esa variable de control)
- La condición de control (el valor que puede tomar esa variable como valor final), es decir, hasta dónde puede llegar esa variable de control
- Una vez que llega a este valor, el bucle deja de repetirse
- Incremento (Δ): valor en el que se va a incrementar esa variable de control en cada interacción del bucle
- Cuerpo: código que se va a ejecutar en cada interacción del bucle

Programación en R

for (I in 1:N) { Proceso }



I sale valiendo N+1

Bucles anidados (nested loops)

Ejemplo: queremos obtener el valor de un sumatorio.

$$T = \sum_{r=1}^{M} (r^2 + 1)$$

$T = 0;$

$T = 0 + 2 = 2$

$T = 2 + 5 = 7$

$T = 7 + 10 = 17$

...

$T = T + (M^2 + 1)$

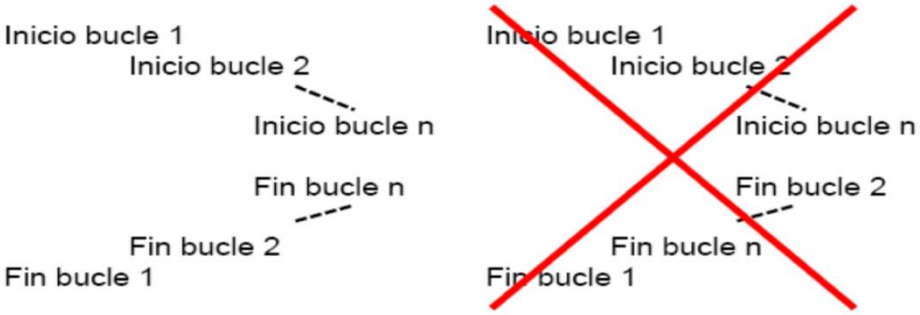
Inicio
Leer M
Hacer T←0
Para r desde 1 hasta M
T←T+(r²+1)
Fin del bucle en r
Escribir T
Fin

Un bucle anidado es un bucle que se encuentra incluido en el bloque de sentencias de otro bloque. Los bucles pueden tener cualquier nivel de anidamiento (un bucle dentro de otro bucle dentro de un tercero, etc.).

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.

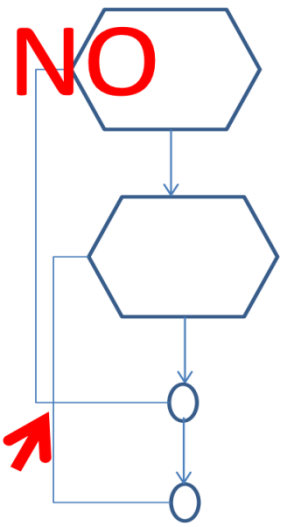
En los bucles anidados es importante utilizar variables de control distintas, para no obtener resultados inesperados.

Los bucles pueden anidarse siempre que se cumpla:

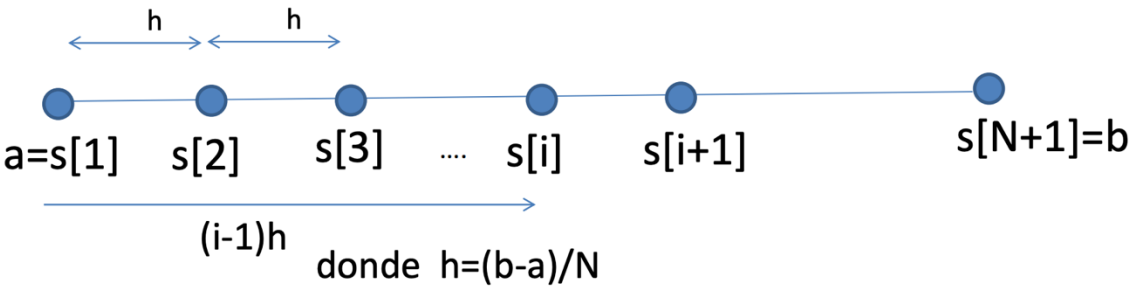


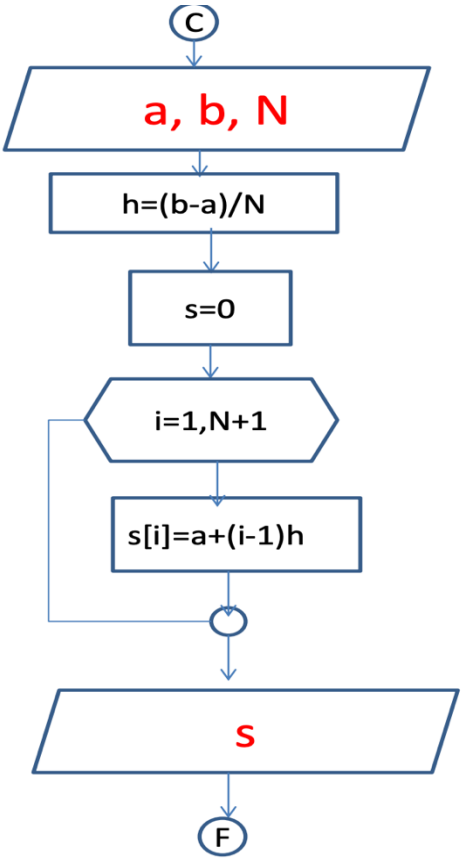
¡ATENCIÓN!

En los bucles anidados, tenemos que cerrar primero el último, luego el penúltimo, y así sucesivamente



Ejemplo propuesto en clase: queremos dividir un segmento [a, b] en N subintervalos iguales generando un vector s que contenga las abscisas de los extremos de los subintervalos (haciendo coincidir s [1]=a.





s[i]=s[i-1]+h

i=1	s[1]=a
i=2	s[2]=a+h
i=3	s[3]=a+2h
i=4	s[4]=a+3h
i=5	s[5]=a+4h=b

s[i]=a+(i-1)h

NOTA:
Se podría no poner s=0 sin embargo al programar Sí es necesario ponerlo: s=c(0)

Bucles condicionales:

Estructuras condicionales de ramificación (tipo if) y bucles condicionales (tipo while)

Los **condicionales, o las sentencias condicionales**, son un tipo de instrucciones de control. Junto con los bucles, nos sirven para controlar nuestro programa sin necesidad de intervenir, es decir, gracias a las instrucciones de control, nuestro programa podrá actuar de una forma más o menos autónoma. En lo que respecta a los condicionales, el programa actuará en función de qué condición se cumpla en cada caso determinado.

Existen varios tipos de condicionales, que dependerán del número de condiciones que necesitemos cubrir. Los condicionales más utilizados son if, if-else y switchcase.

El símbolo del organigrama para la condición es un rombo, dentro del cual se introduce la condición, y por sus vértices salen los dos flujos de información posibles, según si la condición se cumple (es verdadera) o no (es falsa).

Empezamos estudiando variables y operadores lógicos y operaciones que las relacionan:

Variables lógicas

T: True (Verdadero)

F: False (Falso)

Expresiones de relación

== Igual > mayor que < menor que != Distinto
>= mayor o igual <= menor o igual

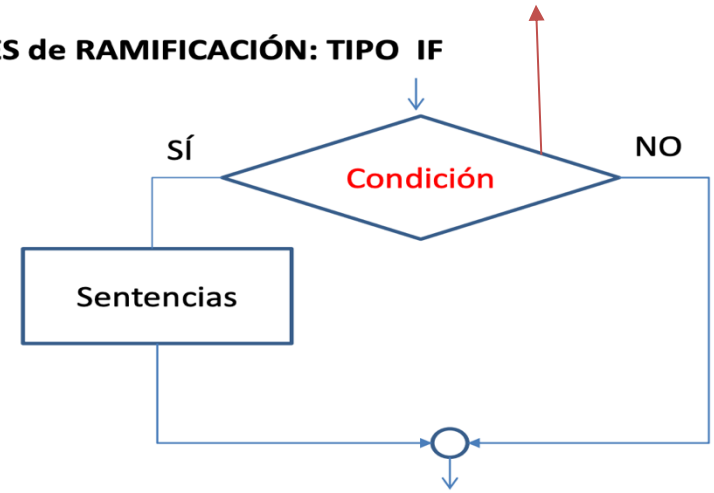
Operadores lógicos

& And (Y) | Or (O)

Las ramas son concluyentes ; o voy por un camino o voy por otro

ESTRUCTURAS CONDICIONALES de RAMIFICACIÓN: TIPO IF

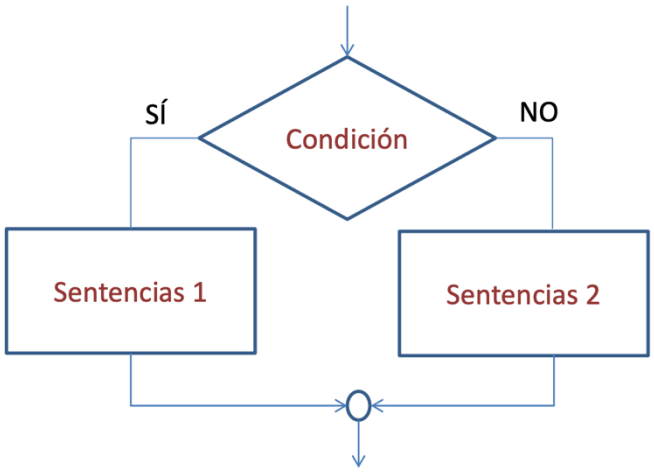
```
Si   (Condición)
          Sentencias
Fin Condición
```



En R

```
if (Condicion){
          Sentencias
}
```

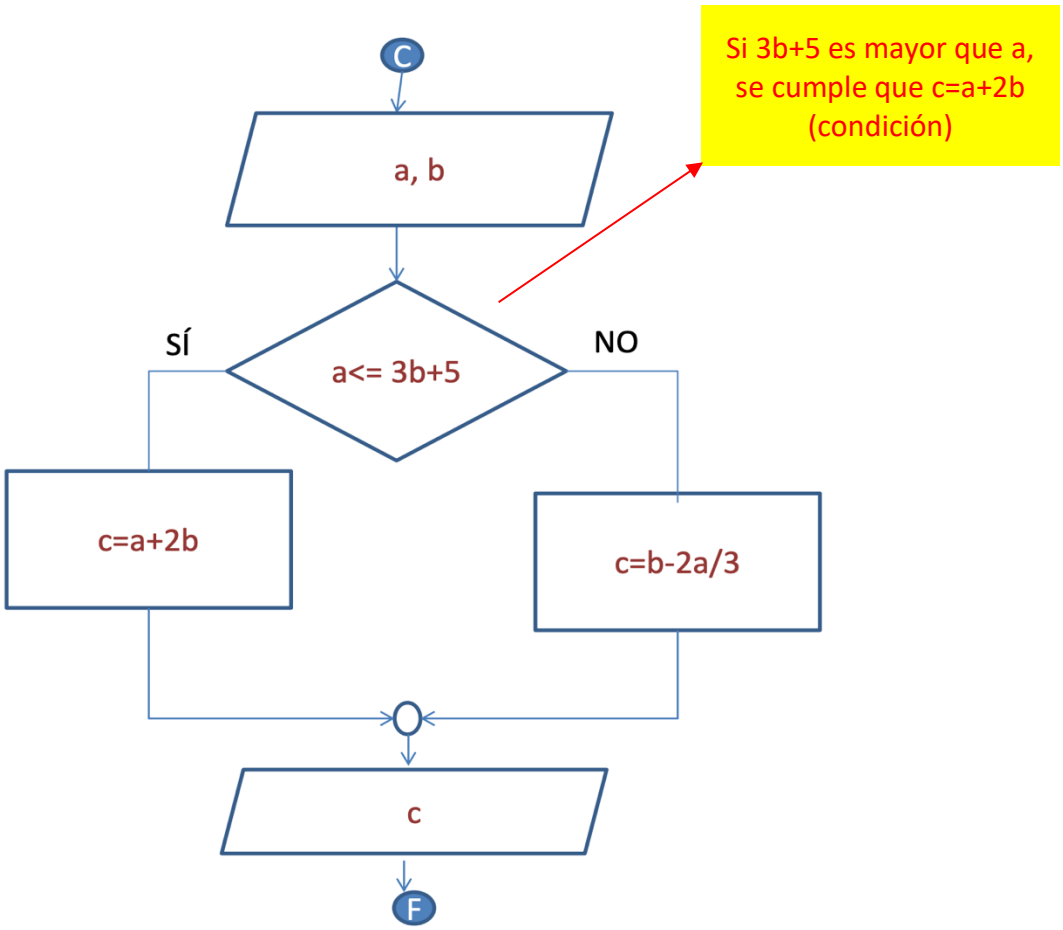
```
Si  (Condición)
        Sentencias 1
Si no
        Sentencias 2
Fin condición
```



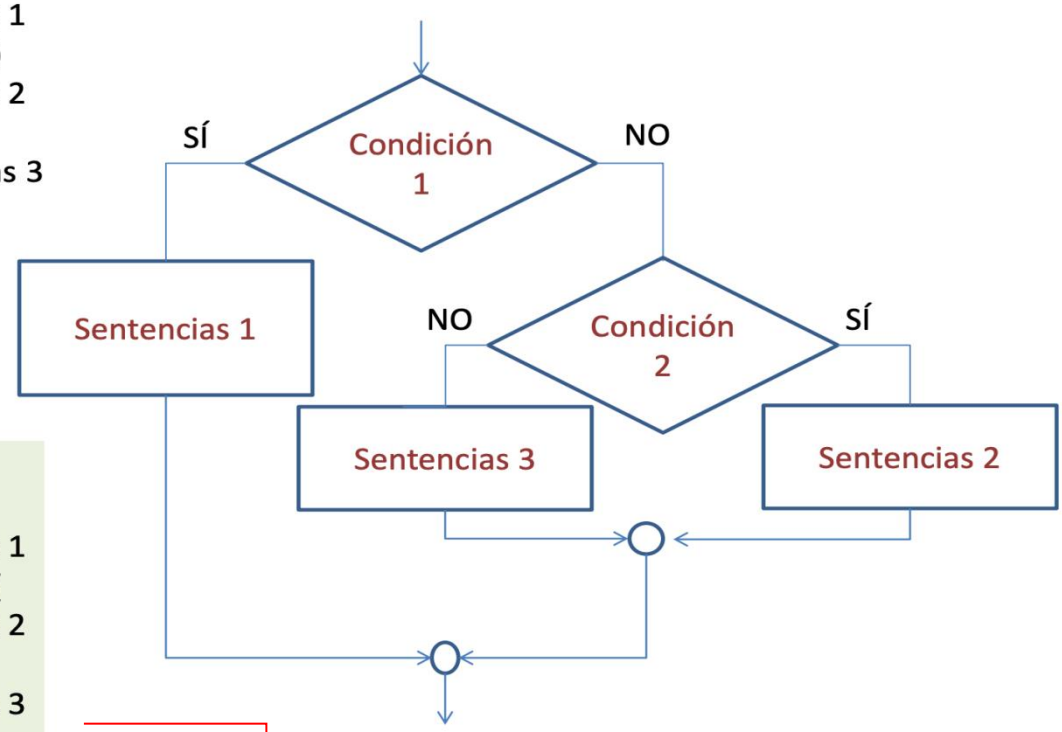

```
En R:
if (Condición){
        Sentencias 1
}else{
        Sentencias 2
}
```

else significa “en cualquier otro caso ”
¡¡¡ NUNCA LLEVA CONDICIÓN!!!

Ejercicio propuesto en clase: dadas dos variables a, b obtener una variable c que valga c=a+2b si se cumple a<= 3b+5 y en caso contrario c = b-2a/3



```
Si  (Condición1)
        Sentencias 1
Si no, si (Condición2)
        Sentencias 2
Si no
        Seentencias 3
Fin condición
```




```
En R:
if (Condición1){
        Sentencias 1
}else if (Condición2){
        Sentencias 2
}else{
        Sentencias 3
}
```

else if ¡¡¡ SIEMPRE LLEVA CONDICIÓN!!!

Ejercicios propuestos: Hacer los siguientes organigramas y pseudo-códigos:

1. Dado un vector v de N componentes queremos encontrar el elemento de mayor valor absoluto y su posición. Resultado: elemento con su signo y posición.

2. Encontrar también el de menor valor absoluto.

3. En una matriz A (m,n) encontrar los elemntos de mayor y menor valor absoluto y su fila y columna.

amax (elemento mayor) ; amin (elemento menor);
imax (fila), jmax (columna) ; imin (fila) , jmin (columna)

3. Sumatorio

Un **sumatorio** es un operador matemático que permite representar sumas muy grandes y se simboliza con la letra griega sigma (Σ). Después del sumatorio, por lo general aparece una variable con un subíndice representado con la letra *i*, dicho subíndice indica qué valores de la variable se suman y se le conoce como índice de suma.

$$\sum_{i=m}^{n} X_i = X_m + X_{m+1} + X_{m+2} + \ldots + X_n$$

El sumatorio no es más que una operación de suma repetida desde «n» veces.

Ejemplo sencillo:

- Realizar un algoritmo para resolver el sumatorio:

$$P \leftarrow \sum_{i=1}^{N} i^3(1+i^2)$$

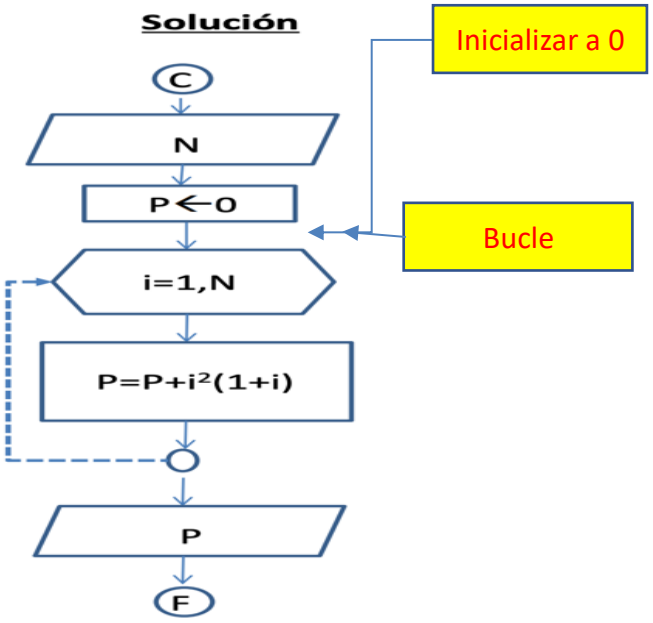
P← 0
Para i=1, P← $0+1^3$ $(1+1^3)=2$
Para i=2, P← $2+2^3$ $(1+2^3)=42$
Para i=3, P← $42+3^3$ $(1+3^3)=312$

…

Para i=i, P← P+ i^3 $(1+i^2)$

…

Hasta i=N



¡ATENCIÓN!

Es importante sumar el valor de la suma anterior cada vez que introducimos un nuevo valor en el sumatorio

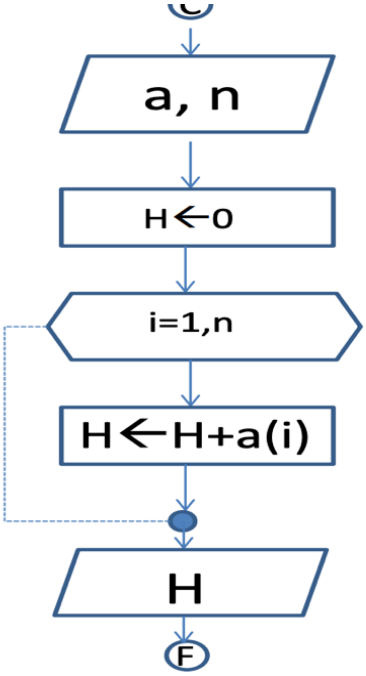
Ejemplo sencillo:

- Realizar un organigrama, y el correspondiente pseudo-código para sumar las componentes de un vector a de n componentes

$$H = \sum_{i=1}^{n} a_i$$

❖ Pseudo-código:

Inicio
Leer a, n
Hacer H ← 0
Para i desde 1 hasta n hacer
H= H + a(i)
Fin del bucle
Escribir H
Fin



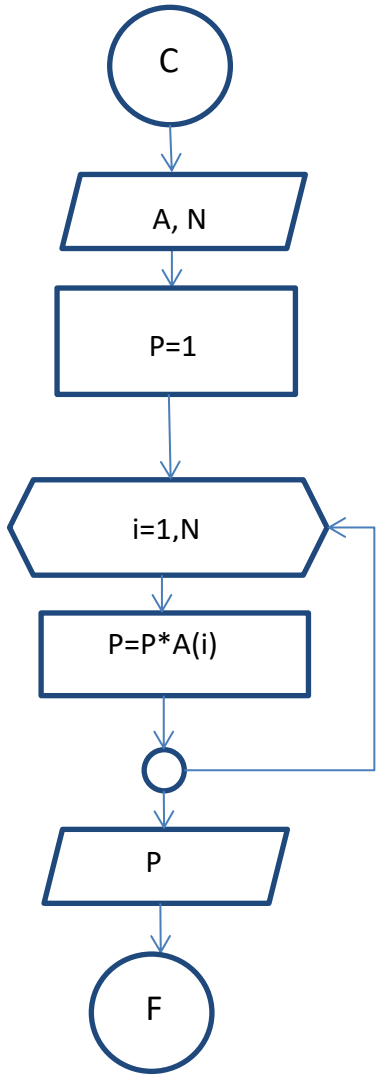
4. Productorio

Los **productorios** se realizan de la misma forma que los sumatorios con algunas diferencias que veremos a continuación.

Matemáticamente se escribe:

$$P = \prod_{i=1}^{N} A(i)$$

Ejemplo sencillo: el factorial de un número es un productorio. Por ejemplo,
5!=1x2x3x4x5=120
El organigrama para resolver un factorial:



¡Cuidado! Los productorios se inicializan a 1, ya que, si lo hiciésemos a 0 (como en los sumatorios), al multiplicar por 0, el resultado sería siempre

Es **importante** multiplicar el valor del producto anterior cada vez que introducimos un nuevo valor

Ejercicios propuestos. Realizar los siguientes organigramas y pseudo-códigos.

1. Halla el resultado de $\sum_{i=1}^{m}(a(i)) + \prod_{j=3}^{t+2}(b(j))$

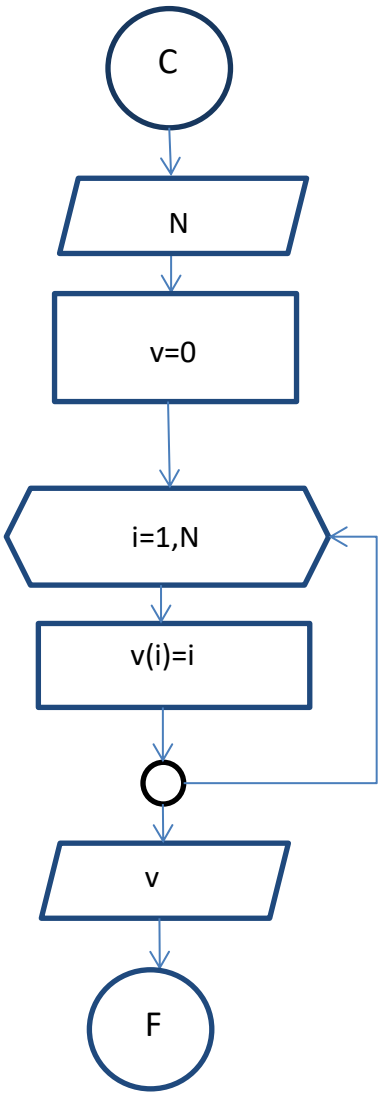
5. Vectores

Un **vector** es una estructura de datos que permite almacenar un conjunto de datos del mismo tipo. Se define un vector con un nombre y gracias a un subíndice (normalmente i), hacemos referencia a cada elemento/ componente del mismo.

Ejemplo sencillo: realizar un organigrama que permita obtener un vector v cuyas componentes sean los N primeros números naturales.

Pseudo-código:

```
Inicio
        Leer N
        Hacer v=0
        Para i desde 1 hasta N
                Hacer v(i)=i
        Fin bucle
        Escribir v
Fin
```



Ejercicios propuestos. Realizar los siguientes organigramas y pseudo-códigos.

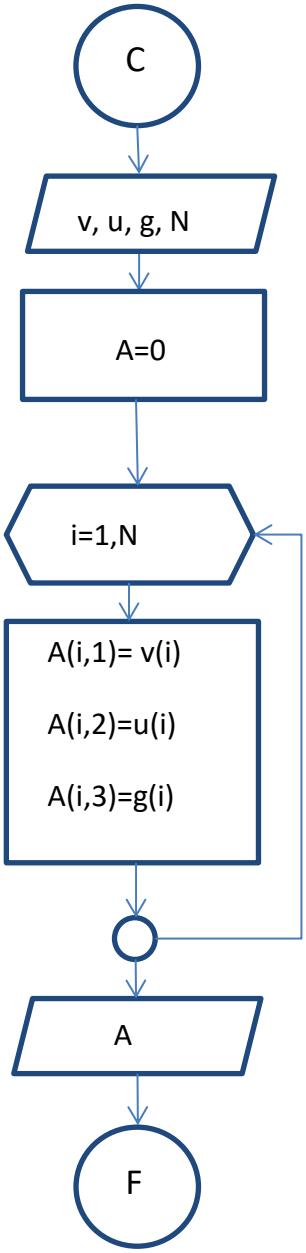
1. Sumar las componentes de un vector a de n componentes. Nota: debes usar un sumatorio.
2. Realizar el producto escalar de dos vectores x, y de N componentes.

6. Matrices

Una **matriz** es una estructura (un poco más compleja que los vectores) que permite almacenar un conjunto de datos ordenados en filas y columnas.

Se define una matriz con un nombre y gracias a un subíndice (normalmente i,j), hacemos referencia a cada elemento/componente.

Ejemplo sencillo: realizar un organigrama que permita obtener una matriz A con N filas y 3 columnas, cuya primera columna contenga los N primeros números naturales (vector v, del anterior apartado), la segunda contenga el vector u y la tercera contenga el vector g.



Ejercicios propuestos. Realizar los siguientes organigramas y pseudo-códigos.

1. Realizar la suma de dos matrices
2. Realizar el producto de dos matrices