

Tema 1

Introducción a la programación

Programación (Desarrollo de aplicaciones web) Carlos Alberto Cortijo Bon



Esta obra está bajo una [Licencia Creative Commons Atribución-NoComercial-SinDerivadas 4.0 Internacional](https://creativecommons.org/licenses/by-nc-nd/4.0/)

Índice

1. Programas y algoritmos	1
2. Notaciones para programas	1
3. Diagramas de flujo	2
3.1. Operaciones de entrada y salida	2
3.2. Sentencias de asignación	3
3.3. Estructuras de control	4
3.4. Secuencias de valores (<i>arrays</i> o <i>arreglos</i>)	8

1. Programas y algoritmos

Un **programa** es una secuencia de instrucciones para un ordenador que se almacena en una memoria. Un programa es un objeto estático.

Para ejecutar un programa, se crea un **proceso**. Un proceso es un objeto dinámico que incluye el estado de ejecución del programa para el que se crea. Este estado va cambiando a lo largo del tiempo. Para un mismo programa podría crearse más de un proceso, y el estado de cada uno evolucionaría de manera independiente a lo largo del tiempo.

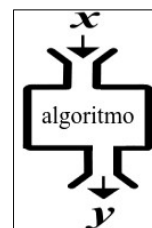
En los actuales ordenadores, los procesos los ejecuta un microprocesador, que puede ejecutar más de un proceso a la vez.

Los programas tienen dos características importantes:

1. Su ejecución puede no terminar nunca. Un programa puede, por ejemplo, controlar la entrada a un aparcamiento. Este programa estaría continuamente ejecutando un bucle sin fin en el que primero espera la llegada de un vehículo, después abre la puerta para que pase, y después cierra la puerta una vez que el vehículo ha pasado. Un servidor web es un programa que se ejecuta indefinidamente, proporcionando el contenido correspondiente para cada petición que recibe.
2. Pueden interactuar con el mundo exterior para obtener información y para mostrar información, o bien para actuar sobre objetos externos, enviándoles órdenes para que realicen acciones. Un programa puede mostrar un mensaje en pantalla pidiendo que se introduzca un número, y acto seguido leer un número que se introduce por teclado. Un programa que controla la entrada a un aparcamiento puede obtener una imagen de la matrícula mediante una cámara, enviar las órdenes y la información apropiadas a una impresora para que imprima un *ticket* de aparcamiento, y mostrar un mensaje en una pantalla pidiendo que se recoja el *ticket*. Por último, puede enviar una orden a la barrera para que suba y permita el paso de un coche. Un sistema operativo está formado por un conjunto de programas, muchos de los cuales se ejecutan indefinidamente porque su propósito es proporcionar servicios a otros programas. El programa que muestra este documento se puede ejecutar indefinidamente sin hacer nada más que esperar a que el usuario interactúe con su interfaz de usuario, pulsando determinadas teclas del teclado, o pulsando con el ratón sobre determinadas zonas de su interfaz de usuario.

Un **algoritmo** es un conjunto de instrucciones para realizar un cálculo o realizar una tarea. Se puede considerar un tipo particular de programa que tiene las siguientes características:

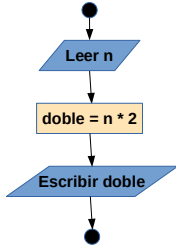
1. Su ejecución termina en un número finito de pasos. Este puede ser todo lo grande que sea, pero siempre finito.
2. No interactúa con el mundo exterior. Puede recibir uno o más valores de entrada, y en base a ellos calcular uno o más valores de salida. En este aspecto, se asemeja bastante a una función matemática. Se puede decir que un algoritmo es un procedimiento de cálculo que permite obtener el valor de una función para valores dados.



Un programa puede incluir diversos algoritmos. Por ejemplo, el programa para el control de la entrada a un aparcamiento incluye un algoritmo que, a partir de una imagen de la matrícula (que se puede representar como una secuencia de números que indican el color para cada uno de los píxeles), proporciona la matrícula del coche, que se puede representar como una secuencia de caracteres que representa cada uno una cifra o una letra.

2. Notaciones para programas

Un programa se puede expresar utilizando diversas notaciones. Se muestra un ejemplo para un programa que lee un número entero y escribe el doble del número leído, expresado con distintas notaciones.

Diagrama de flujo	Pseudocódigo	Lenguaje de programación (C++)
 <pre> graph TD Start(()) --> LeerN[/Leer n/] LeerN --> Doble[n * 2] Doble --> EscribirDoble[/Escribir doble/] EscribirDoble --> End(()) </pre>	<pre> Leer n doble = n * 2 Escribir doble </pre>	<pre> #include <iostream> using namespace std; int main() { int n, doble; cin >> n; doble = n * 2; cout << doble; } </pre>

- Diagramas de flujo. Es una notación gráfica. Existen distintos tipos de instrucciones, para cada uno de los cuales se emplea una figura distinta. En este ejemplo se puede ver que para las operaciones de entrada y salida se emplean trapezoides, y para las de asignación rectángulos.
- Pseudocódigo. Es una notación textual. Cada diagrama de flujo tiene su equivalente en forma de pseudocódigo. El pseudocódigo no es una notación estándar ni formal, al contrario que los lenguajes de programación. El pseudocódigo es algo intermedio entre el lenguaje natural y los lenguajes de programación. Con el pseudocódigo se evitan las ambigüedades e imprecisiones del lenguaje natural, y a la vez se evitan muchos detalles que es necesario especificar con un lenguaje de programación.
- Lenguajes de programación. Son notaciones formales. Hay muchos lenguajes de programación. Cada uno tiene su propia sintaxis, que hay que utilizar para escribir programas válidos. Para este ejemplo se ha utilizado el lenguaje de programación C++. Una cosa que salta a la vista y que cabe reseñar es que, con este lenguaje, es necesario declarar el tipo de cada variable (en este caso, `n` y `doble` de tipo `int` (número entero)). También es el caso en el lenguaje de programación Java, pero no es así en otros lenguajes de programación.

3. Diagramas de flujo

Son una notación gráfica para programas. Incluyen operaciones básicas de entrada y salida que permiten leer y escribir datos, así como realizar operaciones con los datos de los que se dispone.

Son útiles para el aprendizaje de la programación. La creación de un programa correspondiente en un lenguaje de programación determinado es más o menos directa a partir de un diagrama de flujo.

3.1. Operaciones de entrada y salida

Las instrucciones de entrada y de salida se representan con un paralelogramo.

Estas instrucciones son las que permiten que un programa se comunique con el mundo exterior.

- La operación de entrada (o lectura) permite leer un dato y almacenarlo en una variable.
- La operación de salida (o escritura) permite escribir información.

Normalmente, una operación de entrada lee datos del teclado, y una operación de salida escribe datos en la pantalla.



	Operación de entrada: lectura de un valor hacia la variable var
	Operación de salida: escritura del valor de la variable var

Tabla 1.1: Operaciones de entrada y salida

3.2. Sentencias de asignación

Permiten asignar un valor a una variable, identificada por un nombre. Tienen la siguiente forma.

```
var = valor
```

Donde *var* es el nombre de una variable, y *valor* es el valor que se le asigna.

El valor puede ser:

- Un valor literal o atómico, que puede ser de distintos tipos:
 - N Numérico, con o sin decimales. Ejemplos: `plnic = 2`, `peso = 2.27183`, `temp = -1.5`.
 - Alfabético. Cadenas de caracteres, que se representan entre comillas dobles. Ejemplos: `saludo = "Hola"`, `nombre = "Manuel"`, `rom = "X"`.
- Otra variable. Por ejemplo: `a = b`.
- El resultado de evaluar una expresión, en la que se pueden utilizar valores literales o atómicos, variables, y diversos tipos de operadores, entre ellos:
 - Operadores aritméticos. Por ejemplo: `precio = 5 + 6.5*horas`

Se pueden utilizar los siguientes operadores. Esta no pretende ser una lista exhaustiva. En cualquier caso, se trata de pseudocódigo, no de un lenguaje de programación determinado. Si se echa en falta algún operador, se puede describir verbalmente la operación a realizar. Pero debe hacerse de manera clara y precisa. Por ejemplo, se podría escribir raíz(a) para indicar \sqrt{a} o raíz_cúbica(a) para indicar $\sqrt[3]{a}$.

Operador	Significado	Ejemplo
+	Suma	a + b
-	Resta	a - b
*	Multiplicación	a * b
/	División	a / b
div	División entera	a div b
%	Resto de división entera	a % b
^	Potencia	a ^ b

- Operador de concatenación de cadenas. Por ejemplo: `saludoPers = saludo + ", " + nombre`.

Con lo poco que se ha visto hasta ahora ya se pueden plantear programas sencillos pero interesantes. Por ejemplo: no existe ningún tipo de sentencia que permita intercambiar directamente los valores de dos variables. Para ello debe utilizarse una variable auxiliar, como se muestra en el siguiente programa de ejemplo.

Diagrama de flujo	Pseudocódigo
<pre> graph TD Start(()) --> Read[/Leer a, b/] Read --> Aux[aux = a] Aux --> A[a = b] A --> B[b = aux] B --> Write[/Escribir a, b/] Write --> End(()) </pre>	<pre> Leer a, b aux = a a = b b = aux Escribir a, b </pre>

3.3. Estructuras de control

En la siguiente ilustración se muestra la representación en los diagramas de flujo de las distintas estructuras de control disponibles en los lenguajes estructurados. En ellos, cada operación se representa mediante un rectángulo, y cada decisión mediante un rombo. Se muestra también cómo se podría expresar lo mismo en pseudocódigo.

Los rectángulos representan sentencias, y los rombos representan condiciones que se comprueban para, dependiendo de si se cumplen o no, seguir el curso de ejecución por una rama o por otra.

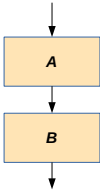
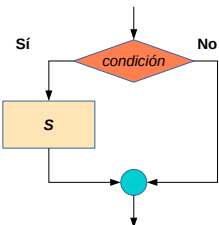
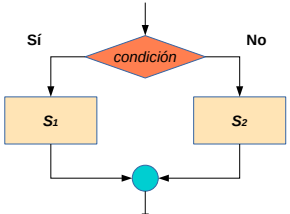
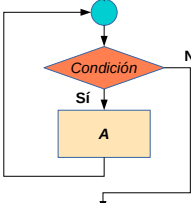
	Diagrama de flujo	Explicación	Pseudocódigo
Secuencia		Se ejecutan las sentencias una tras otra: primero A y después B.	A B (Nota: Se utilizan empezar y terminar)
Condición (variante 1)		Si se cumple la condición, se ejecuta la sentencia S.	Si condición Entonces S FinSi
Condición (variante 2)		Si se cumple la condición, se ejecuta la sentencia S ₁ . Si no se cumple, se ejecuta la sentencia S ₂ .	Si condición Entonces S₁ SiNo S₂ FinSi
Iteración		Mientras se cumpla la condición, se ejecuta la sentencia A.	Mientras condición Hacer A FinMientras

Tabla 1.2: Estructuras de control básicas

Para expresar las condiciones, se suelen utilizar operadores relacionales, que permiten expresar relaciones entre cantidades.

Operador	Significado	Ejemplo
<	Menor	i < n
<=	Menor o igual	iMol <= 10
>	Mayor	c > 1
>=	Mayor o igual	cRes >= 0
=	Igual	a = b
<>	Distinto	a <> b

Para combinar distintas condiciones, se pueden utilizar operadores lógicos.

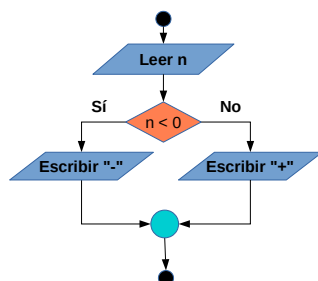
Operador	Significado	Ejemplo
o	Se cumple alguna de las condiciones	car = "a" o car = "i"
y	Se cumplen todas las condiciones	nPer >= 0 y nPer < maxPer
no	No se cumple la condición	no (x > min y x < max)

Dentro de un grupo de sentencias contenidas en una de las ramas de una estructura de control (por ejemplo: las ramas de estructuras Si ... SiNo ... FinSi, Mientras ... FinMientras), debe dejarse un espacio al principio de cada sentencia. Esto se conoce como indentación, y es esencial para la legibilidad. En algunos lenguajes de programación, como por ejemplo Python, la indentación es necesaria no ya para la legibilidad de un programa, sino también para su corrección.

A
B
Si condición Entonces A FinSi
Si condición Entonces A SiNo B FinSi
Mientras condición Hacer A FinMientras

A continuación se dan algunos ejemplos sencillos de diagramas de flujo que utilizan las estructuras anteriores.

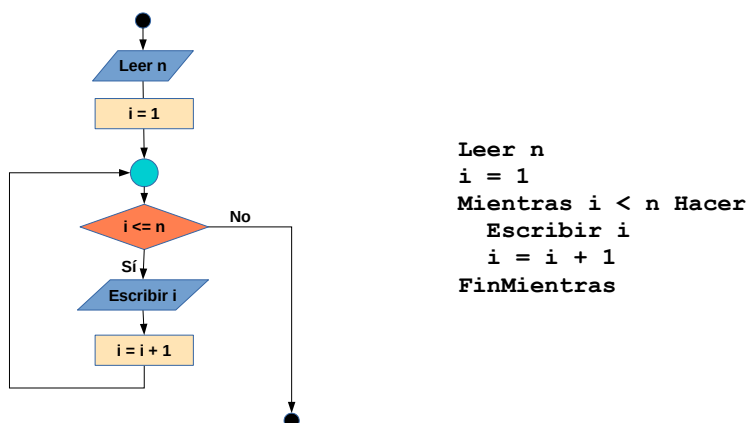
El siguiente ejemplo es el diagrama de flujo y el pseudocódigo para un programa que lee un número y escribe "-" si es negativo y "+" en caso contrario.



```

Leer n
Si n < 0 Entonces
    Escribir "-"
SiNo
    Escribir "+"
FinSi
  
```

El siguiente ejemplo es el diagrama de flujo y el pseudocódigo para un programa que lee un número, que se asume que es positivo, y escribe todos los números entre 1 y ese número.

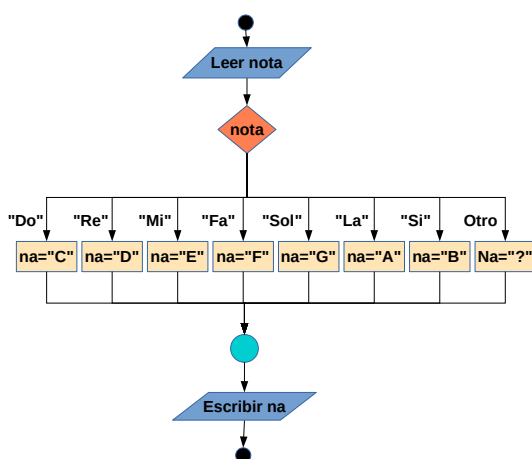


Se puede representar también la selección múltiple dependiendo de los diferentes valores de una variable.

	Diagrama de flujo	Explicación	Pseudocódigo
Selección múltiple según valores de una variable.	<p>(Nota: en una rama puede haber más de un valor, separados por comas).</p>	<p>Dependiendo de los diferentes valores que pueda tener <i>var</i>, se ejecuta un bloque u otro. Si no tiene ninguno de los valores anteriores, se ejecuta un bloque por defecto. Esto último es opcional. Es decir, se puede especificar un bloque para valores particulares pero no un bloque por defecto.</p>	<pre> Según var Caso valor1 S1 Caso valor2 S2 ... Otro caso Sn Fin Según </pre>

Tabla 1.3: Estructura de selección múltiple

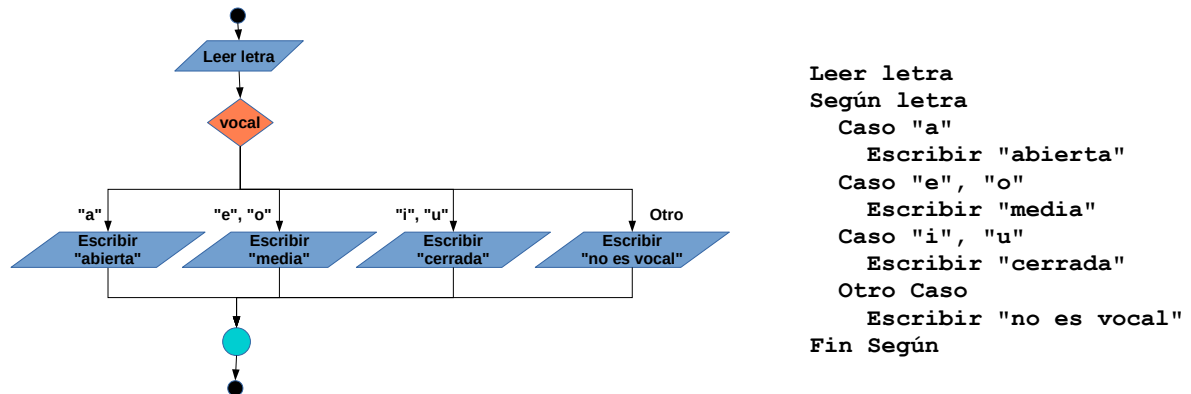
El siguiente ejemplo muestra el diagrama de flujo para un programa que utiliza la estructura anterior. El programa lee el nombre de una nota musical en notación latina y escribe el nombre correspondiente en notación anglosajona. También el pseudocódigo correspondiente.



```

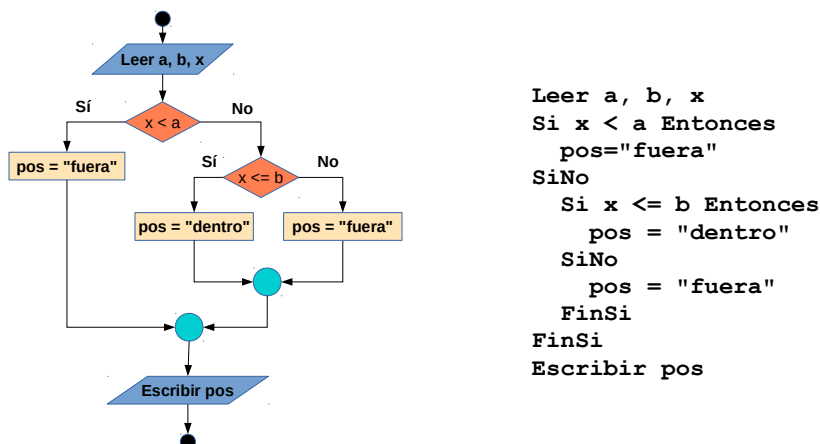
Leer nota
Según nota
  Caso "Do"
    na = "C"
  Caso "Re"
    na = "D"
  Caso "Mi"
    na = "E"
  Caso "Fa"
    na = "F"
  Caso "Sol"
    na = "G"
  Caso "La"
    na = "A"
  Caso "Si"
    na = "B"
  Otro caso
    na = "?"
Fin Según
Escribir na
  
```


Como ya se ha comentado, en alguna rama puede haber más de un valor, separados por comas. En el siguiente ejemplo, se lee una letra y, si es una vocal, se escribe un texto indicando el tipo de vocal que es (abierta, media o cerrada). Si no es una vocal, se escribe un texto indicando que no es una vocal.

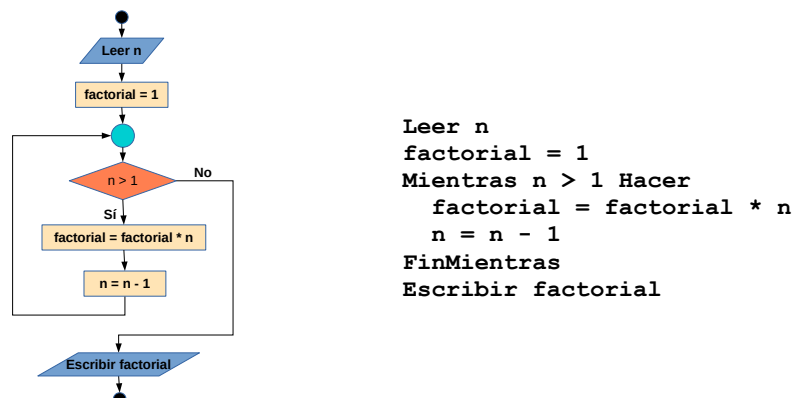


A continuación se incluyen algunos ejemplos más.

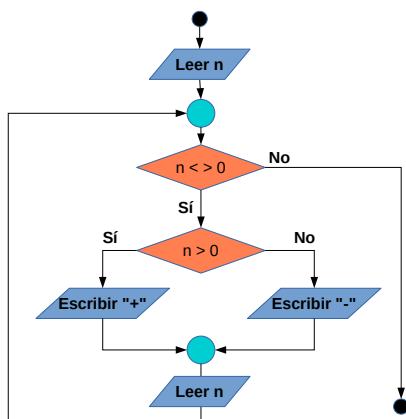
El siguiente ejemplo es el diagrama de flujo y el pseudocódigo para un programa que lee tres números a, b y x. Se asume que entre los dos primeros el primero es menor o igual que el segundo, es decir, que $a \leq b$. El programa escribe "dentro" si x está entre a y b (es decir, si $a \leq x \leq b$), o no, y "fuera" en caso contrario.



El siguiente ejemplo es el diagrama de flujo para un algoritmo que calcula el factorial de un número, y el pseudocódigo correspondiente.



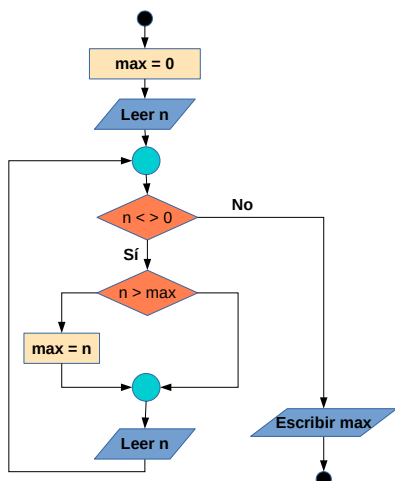
A continuación se muestra el diagrama de flujo y pseudocódigo para un programa que lee números, uno a uno, hasta que se introduce un 0. Para cada número, escribe "+" o "-" según sea positivo o negativo.



```

Leer n
Mientras n <> 0 Hacer
  Si n > 0 Entonces
    Escribir "+"
  SiNo
    Escribir "-"
  FinSi
  Leer n
FinMientras
  
```

A continuación se muestra el diagrama de flujo y pseudocódigo para un programa que lee números, uno a uno, hasta que se introduce un 0. Se asume que todos los números introducidos son no negativos. Cuando se introduce un 0, el programa escribe el máximo de entre todos los números introducidos.



```

max = 0
Leer n
Mientras n <> 0 Hacer
  Si n > max Entonces
    max = n
  FinSi
  Leer n
FinMientras
Escribir max
  
```

3.4. Secuencias de valores (*arrays* o arreglos)

Una secuencia de valores, habitualmente denominada *array* en inglés o arreglo en español, almacena varios valores. Por ejemplo, se podría tener una secuencia para los nombres de los días de la semana.

```
diasSemana = { "Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo" }
```

Cada elemento de una secuencia está en una posición o índice. El índice de la primera posición es 0. Para el ejemplo anterior, se haría referencia a los elementos de la secuencia con `diasSemana[0]`, `diasSemana[1]`, hasta `diasSemana[6]`.

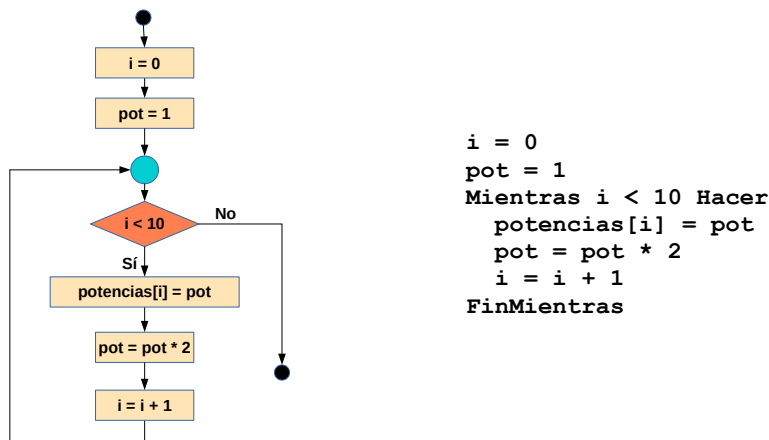
Se puede hacer referencia a la longitud de la secuencia con `longitud(secuencia)`, donde *secuencia* es el nombre de la secuencia. Para el ejemplo anterior, `longitud(diasSemana)` es 7. La longitud de la secuencia viene dada por el mayor de los índices correspondiente a una posición en la que hay un elemento.

También se puede asignar un valor a un elemento de una secuencia. Por ejemplo: `fib[0] = 1`. El valor asignado se puede especificar también con una expresión.

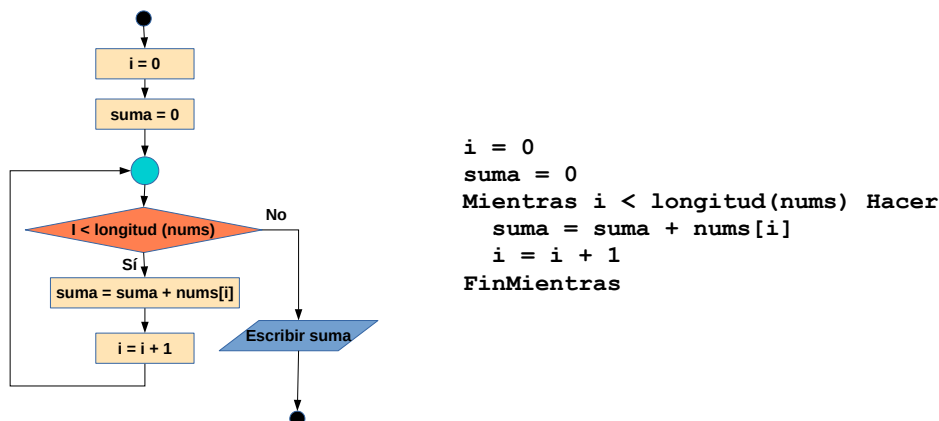
Se pueden utilizar los elementos de una secuencia como si fueran una variable, en lo que respecta a obtener y modificar su valor. Por ejemplo: se podría asignar a una variable el primer día de la semana con `primerDiaSem = diasSemana[0]`. También se pueden utilizar en expresiones de

Se pueden utilizar bucles para obtener todos los elementos de una secuencia, o para asignarles un valor.

El siguiente ejemplo almacena en una secuencia las 10 primeras potencias de 2, empezando por 1.



El siguiente ejemplo obtiene la suma de los elementos de una secuencia.



Actividad 1.1

Escribir un diagrama de flujo para un programa que pida un número y escriba la suma de los números de 1 al número introducido.

Actividad 1.2

Escribir un diagrama de flujo para un programa que pida un número tras otro. Cuando se introduzca un 0, el programa escribirá la suma de los números positivos que se han introducido. Es decir, los números negativos se ignoran.

Actividad 1.3

Escribir un diagrama de flujo que pida una cantidad numérica. Este es el importe de la compra de un cliente. Entre 100€ (inclusive) hasta 1000€ (no inclusive), se aplica un 5% de descuento. A partir de 1000€ (inclusive) se aplica un 10% de descuento. El programa debe escribir la cantidad final que debe pagar el cliente, una vez aplicado el descuento que, en su caso, corresponda.

Actividad 1.4

Escribir un diagrama de flujo que pida una cantidad numérica y un tipo de cliente. Si el tipo de cliente es A, tiene un descuento de 8%. Si es B, tiene un descuento de un 16%, si es C de un 18%, y si es D de un 20%. El programa debe escribir la cantidad final que debe pagar el cliente, una vez aplicado el descuento que, en su caso, corresponda.