Técnico en

DESARROLLO DE SOFTWARE

Lógica de la Programación II

Lógica de la Programación II

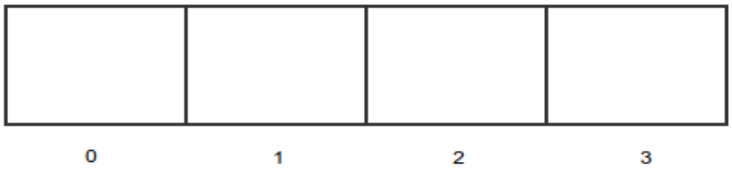
Unidad II

Arreglos

1. Arreglos

Se denomina arreglo a una colección ordenada de valores. Cada valor es llamado elemento y cada elemento tiene una posición numérica dentro del arreglo conocido como índice.

Dependiendo del lenguaje de programación el índice puede comenzar en cero o en uno. Los arreglos se pueden representar de la siguiente forma:



En JavaScript el índice comienza en 0 y no necesita tener índices contiguos.

Según el lenguaje de programación los valores que se almacenan en un arreglo pueden ser del mismo tipo de datos o de múltiples tipos de datos.

Al hablar del tamaño o dimensión de un arreglo, se refiere al número de elementos que el arreglo contiene. En algunos lenguajes de programación los arreglos pueden ser estáticos o dinámicos.

- *Arreglos estáticos:* se debe declarar la dimensión del arreglo antes de utilizarlo y no puede crecer o decrecer durante la ejecución del programa.
- *Arreglos dinámicos:* no hay necesidad de declarar una dimensión fija, pueden crecer o decrecer según sea necesario.

1.1. Creando arreglos

Pseudocódigo

Para definir un arreglo denominado números lo podemos hacer de la siguiente forma:

```
definir numeros:arreglo

definir numeros ← arreglo[7]

definir arregloVacio ← arreglo[ ]
```

En la segunda definición teníamos lo siguiente: arreglo[7], donde el número 7 indica que se ocuparán 7 casillas:

```
definir numeros ← arreglo[7]
```

Si dejamos vacío en lugar del número, podemos indicar que desconocemos la cantidad de elementos que tendrá nuestro arreglo, por lo que podemos inicializarlo como vacío de la siguiente forma:

```
definir arregloVacio ← arreglo[ ]
```

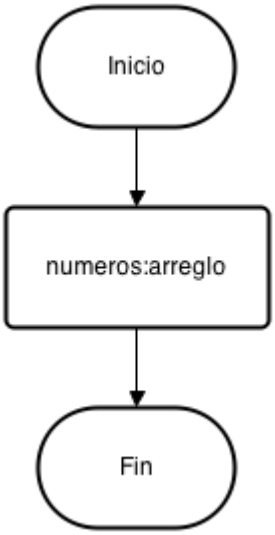
Ejemplos:

```
//Numeros

definir numeros ← arreglo[3]

numeros ← arreglo[1,2,3]

definir arregloVacio ← arreglo[ ]

//Literales

definir coloresPrimarios ← arreglo["rojo","amarillo","azul"]

definir numeros ← arreglo[4,8,15,16,23,42]

//Variables

definir email ← "ejemplo@galileo.edu"

definir puntos ← 5

definir multiplesTiposDeDatos = arreglo[email, puntos, verdadero, 0.5]

//Si queremos omitir una posición del arreglo, podemos hacerlo de la siguiente forma:

definir numeros ← arreglo [1,2, ,4,5]
```

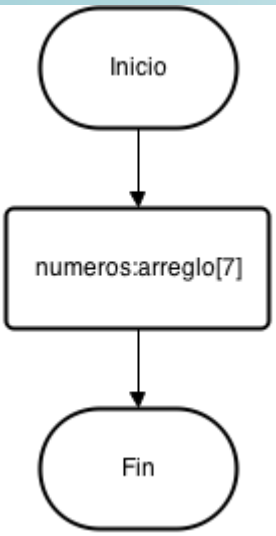
```
//Si se dan cuenta, quedó un espacio vacío entre el elemento con valor 2 y el elemento con valor 4, por lo que solo deben de dejar el espacio en vacío, separándolo con una coma del siguiente elemento.
```

Diagrama de flujo

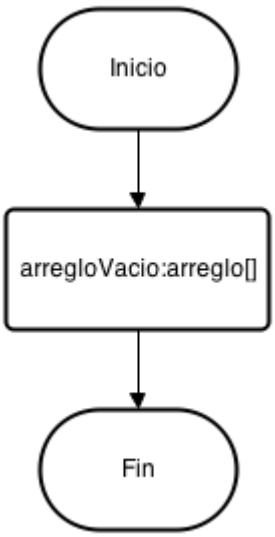
Para declarar una variable del tipo de dato arreglo lo podemos hacer de la siguiente forma:



Para declarar un arreglo de un tamaño específico, por ejemplo un arreglo de tamaño siete:

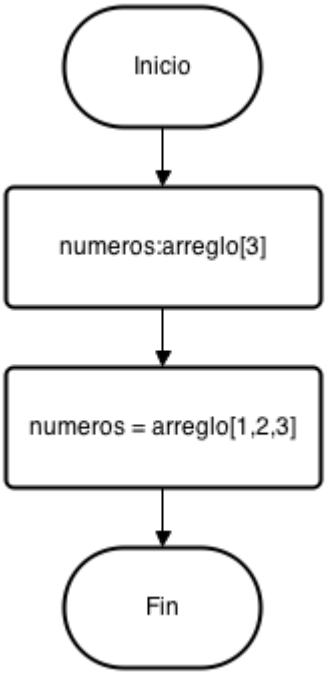


Para declarar un arreglo vacío, se declara la variable del tipo arreglo y se inicializa con corchetes []. Por ejemplo:

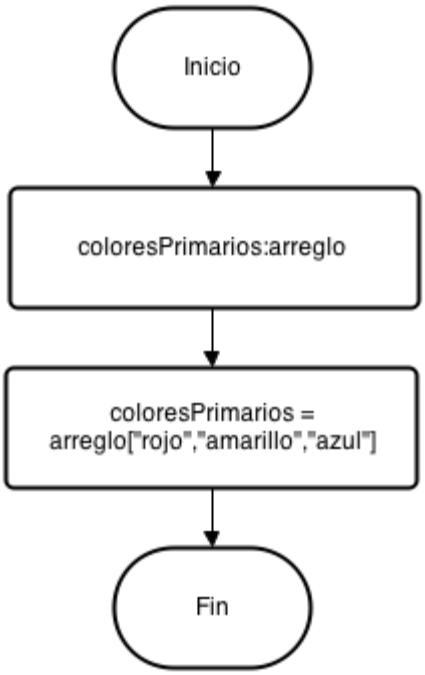


Ejemplos:

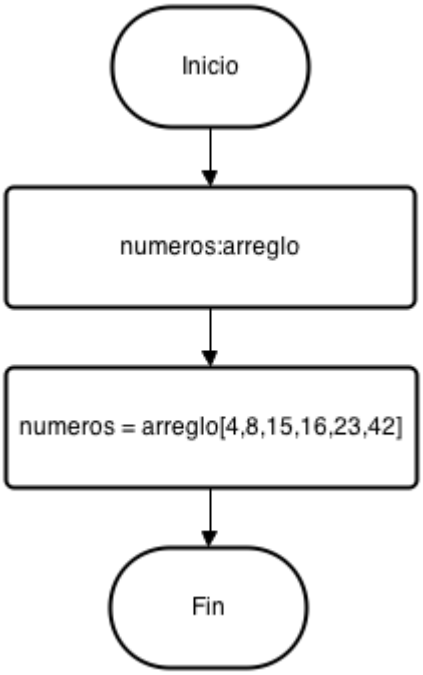
Almacenar literales en un arreglo de tamaño fijo



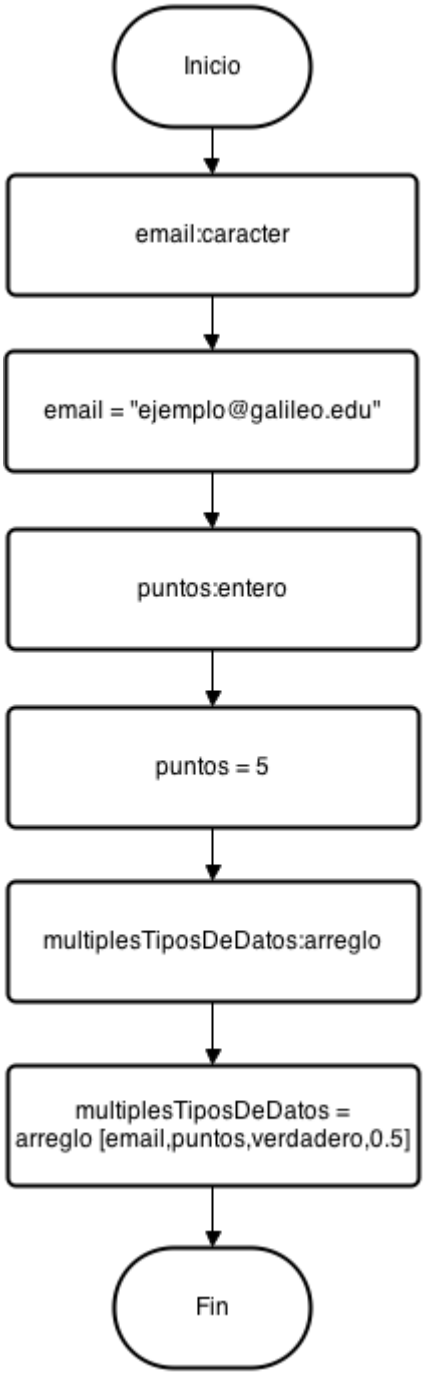
Almacenar literales en un arreglo de tamaño no definido



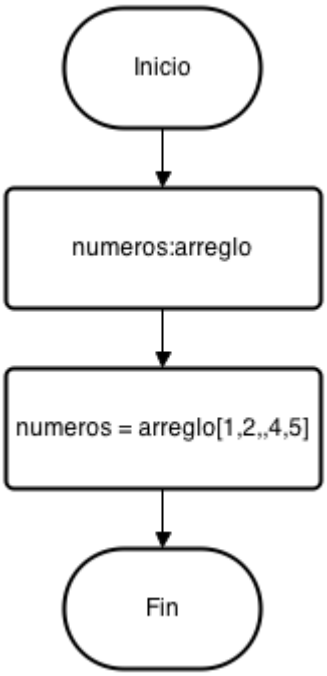
Almacenar números en un arreglo de tamaño no definido



Almacenar variables en un arreglo



Omitir una posición en un arreglo



Código en JavaScript

Para declarar un arreglo se puede hacer de dos formas:

1. Se debe escribir entre corchetes la lista de elementos separada por comas. Los elementos pueden ser literales, variables o constantes. Por ejemplo:

```
//Numeros

var numeros;

numeros = [1,2,3];

//Arreglo vacío
```

```
var arregloVacio = [];

//Literales

var coloresPrimarios = ["rojo","amarillo","azul"];

var numeros = [4,8,15,16,23,42];

//Variables

var email = "ejemplo@galileo.edu";

var puntos = 5;

var multiplesTiposDeDatos = [email,puntos,true,0.5];
```

Si se omite un elemento en el listado, el elemento omitido tendrá valor indefinido:

```
var numeros = [1,2,,4,5];
```

2. Utilizando el constructor Array();

```
        // Especificando el tamaño del arreglo

var numeros = Array(4);

// Listando el contenido del arreglo

var numeros = Array(1,2,4,5);
```

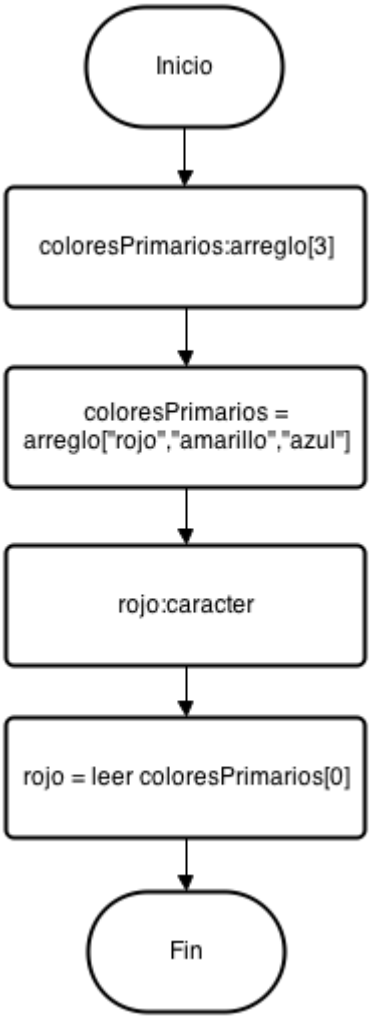
1.2. Leyendo los valores de un arreglo

Pseudocódigo

```
definir coloresPrimarios ← arreglo[3]

coloresPrimarios ← arreglo [“rojo”, “amarillo”, “azul”]

definir rojo ← Leer coloresPrimarios[0]
```

La instrucción “Leer coloresPrimarios[0]” va a obtener el valor que se encuentra en el arreglo “coloresPrimarios” en la posición 0, el cual es “rojo”, por lo que le asigna el valor “rojo” a la variable “rojo”.

Diagrama de flujo



JavaScript

Para leer un elemento de un arreglo se debe escribir el nombre del arreglo y el índice del elemento entre corchetes []. Para el siguiente ejemplo la variable “red” obtendrá el valor “rojo”, ya que el valor que tiene el arreglo en la posición 0 es “rojo”.

```
var coloresPrimarios = ["rojo","amarillo","azul"];

var red = coloresPrimarios[0];
```

El índice también puede estar almacenado en una variable o en un arreglo.

```
var posicion = 2;

var numerosPrimos = [1,3,5,7,11,13];

var cinco = numerosPrimos[posicion]; //valor de numerosPrimos[2] es 5.

var once = numerosPrimos[posicion + 2]; //valor de numerosPrimos[4] es 11.

var trece = numerosPrimos[numerosPrimos[posicion]];
//numerosPrimos[numerosPrimos[2]] es equivalente a numerosPrimos[5] lo cual nos devuelve el valor 13 que corresponde al elemento que se encuentra en la posicion 5 del arreglo numerosPrimos.
```

1.3. Escribiendo los valores de un arreglo

Pseudocódigo

Si queremos asignarle un valor a cada casilla del arreglo, entonces lo vamos a llevar a cabo de la siguiente forma:

```
definir vocales ← arreglo[5]

vocales[0] ← “a”

vocales[1] ← “e”
```

definir i ← “i”

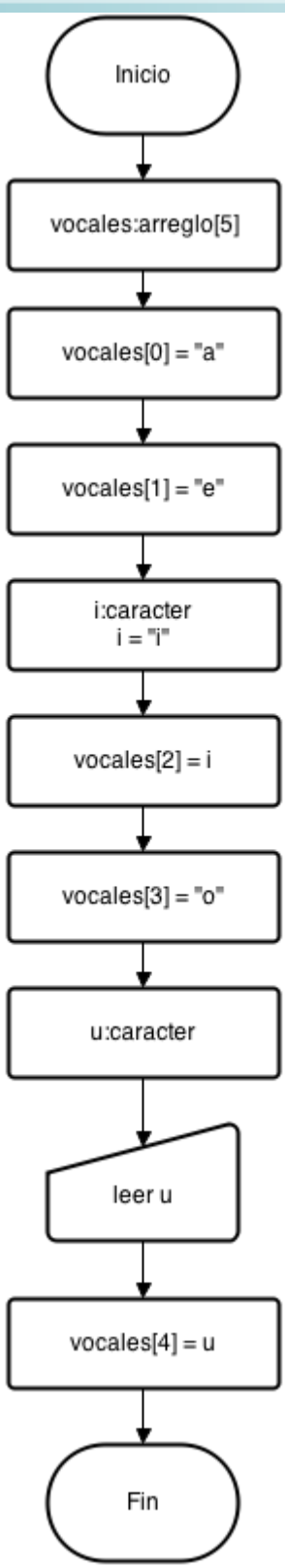
vocales[2] ← i

vocales[3] ← “o”

vocales[4] ← leer cadena de caracteres

La última opción permite que el usuario lea del teclado una cadena de caracteres y le asigna a la posición 4 del arreglo “vocales” el valor ingresado.

Diagrama de flujo


Inicio
vocales:arreglo[5]
vocales[0] = "a"
vocales[1] = "e"
i:caracter
i = "i"
vocales[2] = i
vocales[3] = "o"
u:caracter
leer u
vocales[4] = u
Fin

JavaScript

Para escribir un elemento de un arreglo se debe escribir el nombre del arreglo, el índice del elemento entre corchetes [] y el valor a guardar en el elemento. Por ejemplo:

```
var vocales = [];  //definimos el arreglo

vocales[0] = "a";  //a la posicion 0 del arreglo vocales le asignamos “a”

vocales[1] = "e";  //a la posicion 1 del arreglo vocales le asignamos “e”

var i = "i";  //declaramos la variable i con el valor “i”

vocales[2] = i;  //a la posicion 2 del arreglo vocales le asignamos el valor de i el cual corresponde a “i”.

vocales[3] = "o"; //a la posicion 3 del arreglo vocales le asignamos “o”

var u = prompt("Ingrese la vocal u");

vocales[4] = u; //a la posicion 4 del arreglo vocales le asignamos lo que ingrese el usuario.
```

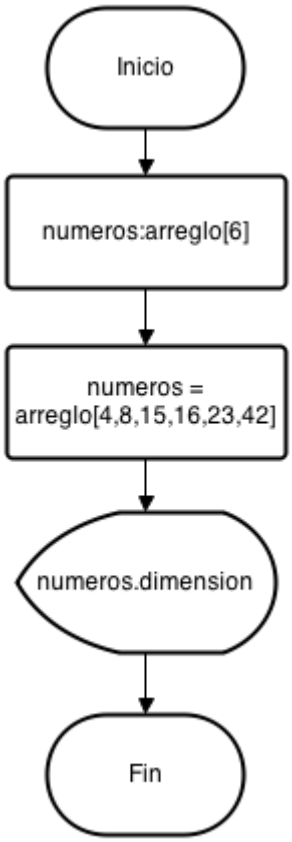
1.4. Dimensión de un arreglo

Pseudocódigo

```
definir numeros ← arreglo[6]

numeros ← arreglo [4,8,15,16,23,42]
```

```
imprimir numeros.dimension    //lo cual debería imprimir 6 que es el tamaño del arreglo.
```

Diagrama de flujo



JavaScript

Para obtener la dimensión o el tamaño de un arreglo se utiliza el comando .length();

```
var numeros = [4,8,15,16,23,42];

alert(numeros.length);  //Dimension 6 ya que el arreglo cuenta con 6 posiciones.
```

Si se han omitido valores de algunos elementos de un arreglo, la dimensión será igual al índice mayor más uno.

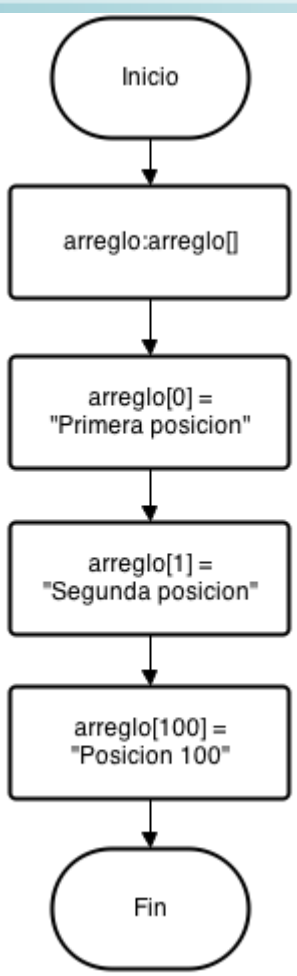
```
var cosas = ["bombilla","globo","tornillo"];

cosas[20] = "clavo";

var tamanio = cosas.length; //Dimension 21, ya que se tiene ocupada la posición 20 del arreglo “tamanio”.
```

1.5. Agregando elementos a un arreglo

Pseudocódigo

```
definir arreglo ← arreglo[ ]

arreglo[0] ← “Primera posicion”

arreglo[1] ← “Segunda posicion”

arreglo[100] ← “Posicion 100”
```

Diagrama de flujo



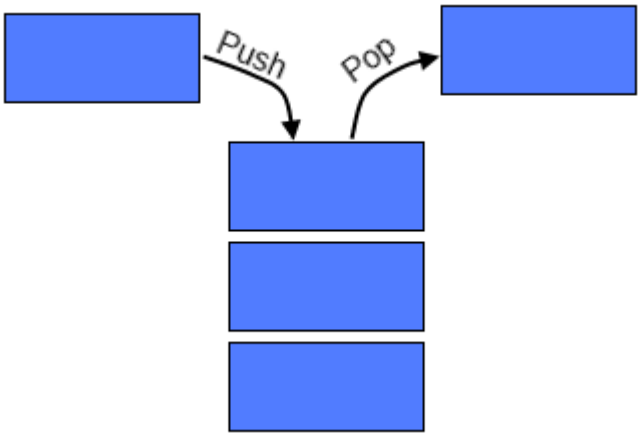
JavaScript

Existen dos maneras de agregar elementos a un arreglo. Esto puede ser de la misma manera que hemos visto con anterioridad, simplemente agregar los elementos al índice. Ejemplo:

```
var arreglo = [];

arreglo[0] = "Primera posicion";
```

```
arreglo[1] = "Segunda posicion";

arreglo[100] = "Posicion 100";
```

También puede utilizarse un método nuevo que se denomina push() y que se hace que el arreglo se comporte como una pila o stack. En una pila todos los elementos entran a la pila, salen de la misma en orden inverso.



Por ejemplo:

```
var arreglo = [];

arreglo.push("Primera posicion");

arreglo.push("Segunda posición", "Tercera posición","Cuarta Posición" );

/* El contenido de arreglo es ["Primera posicion","Segunda posición", "Tercera posición","Cuarta Posición"] */
```

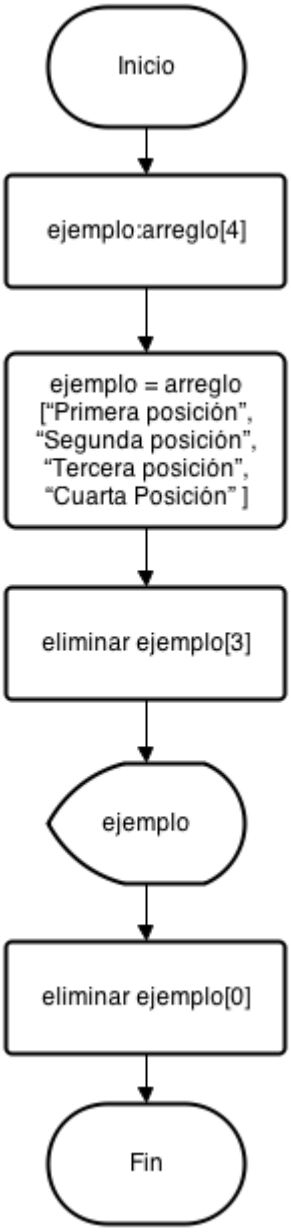
```
alert(arreglo);
```

1.6. Eliminando elementos de un arreglo

Pseudocódigo

```
definir ejemplo ← arreglo[4]

ejemplo ← arreglo[“Primera posición”, “Segunda posición”, “Tercera posición”, “Cuarta Posición” ]

eliminar ejemplo[3] //elimina la última posición del arreglo.

//al imprimir el arreglo, este devolverá:

imprimir ejemplo; // “Primera posición”, “Segunda posición”, “Tercera posición”,

//si eliminamos la primera posición del arreglo:

eliminar ejemplo[0]

//al imprimir el arreglo, este devolverá:

imprimir ejemplo; // , “Segunda posición”, “Tercera posición”,
```

Diagrama de flujo



JavaScript

La función inversa de push() para agregar valores a un arreglo es pop() la cual reduce la dimensión del arreglo en uno y devuelve el valor que se eliminó del arreglo.

Ejemplo:

```
var ejemplo = ["Primera posición","Segunda posición", "Tercera posición","Cuarta Posición"]

var ultimoValor = ejemplo.pop(); // Valor "Cuarta Posición"

/* El contenido de arreglo es ["Primera posicion","Segunda posición", "Tercera posición"]; */

alert(ejemplo);
```

Otra manera de eliminar el valor de un arreglo es de la siguiente manera:

```
var ejemplo = [1,2,3,4,5];

delete ejemplo[2];

// Entonces el contenido del ejemplo sería el siguiente [1,2,NULL,4,5]

alert(ejemplo);
```

Como se puede apreciar, utilizando la función delete no afecta las dimensiones del arreglo sino únicamente asigna valores nulos al índice especificado.

1.7. Recorriendo un arreglo

A veces es necesario recorrer un arreglo para lo cual se utilizan estructuras iterativas.

Pseudocódigo

Utilizando for

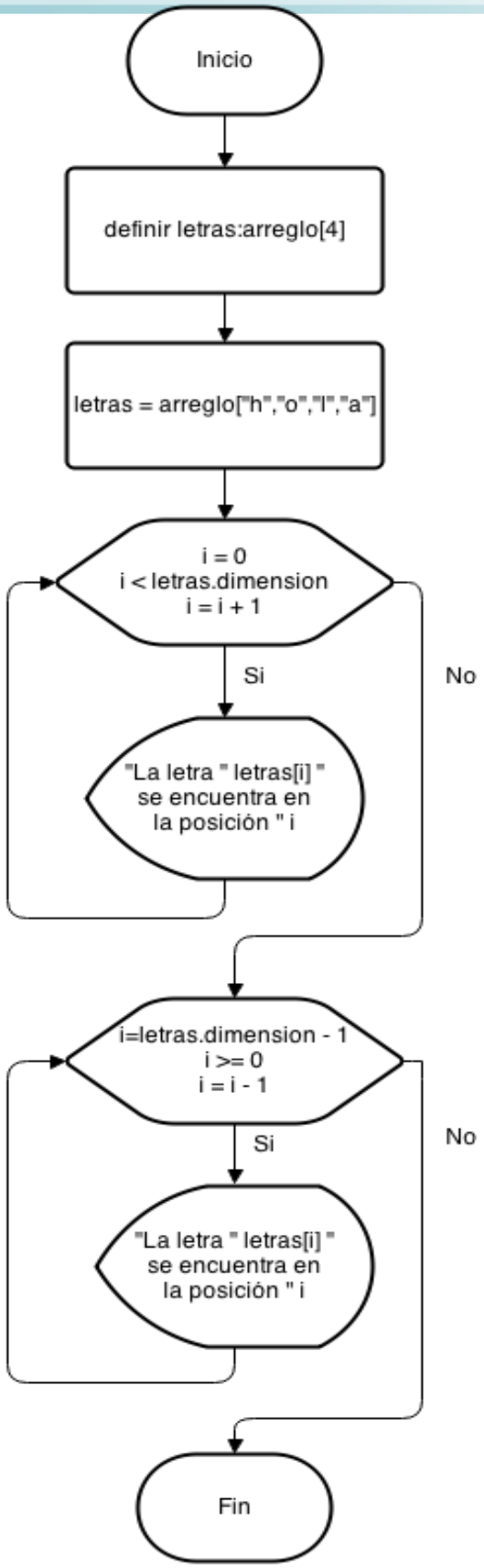
```
definir letras ← arreglo["h","o","l","a"]

// for va a recorrer desde el índice más pequeño que es cero hasta el índice más grande

PARA (i ← 0; MIENTRAS i < letras.dimension; i++)

    imprimir "La letra "  letras[i]  " se encuentra en la posición " i

FIN PARA

// Ahora lo va a recorrer a la inversa desde el índice más grande hasta el índice 0

PARA (i ← letras.dimension - 1; MIENTRAS i >= 0; i--)

    imprimir "La letra " letras[i] " se encuentra en la posición " i

FIN PARA
```

Utilizando while

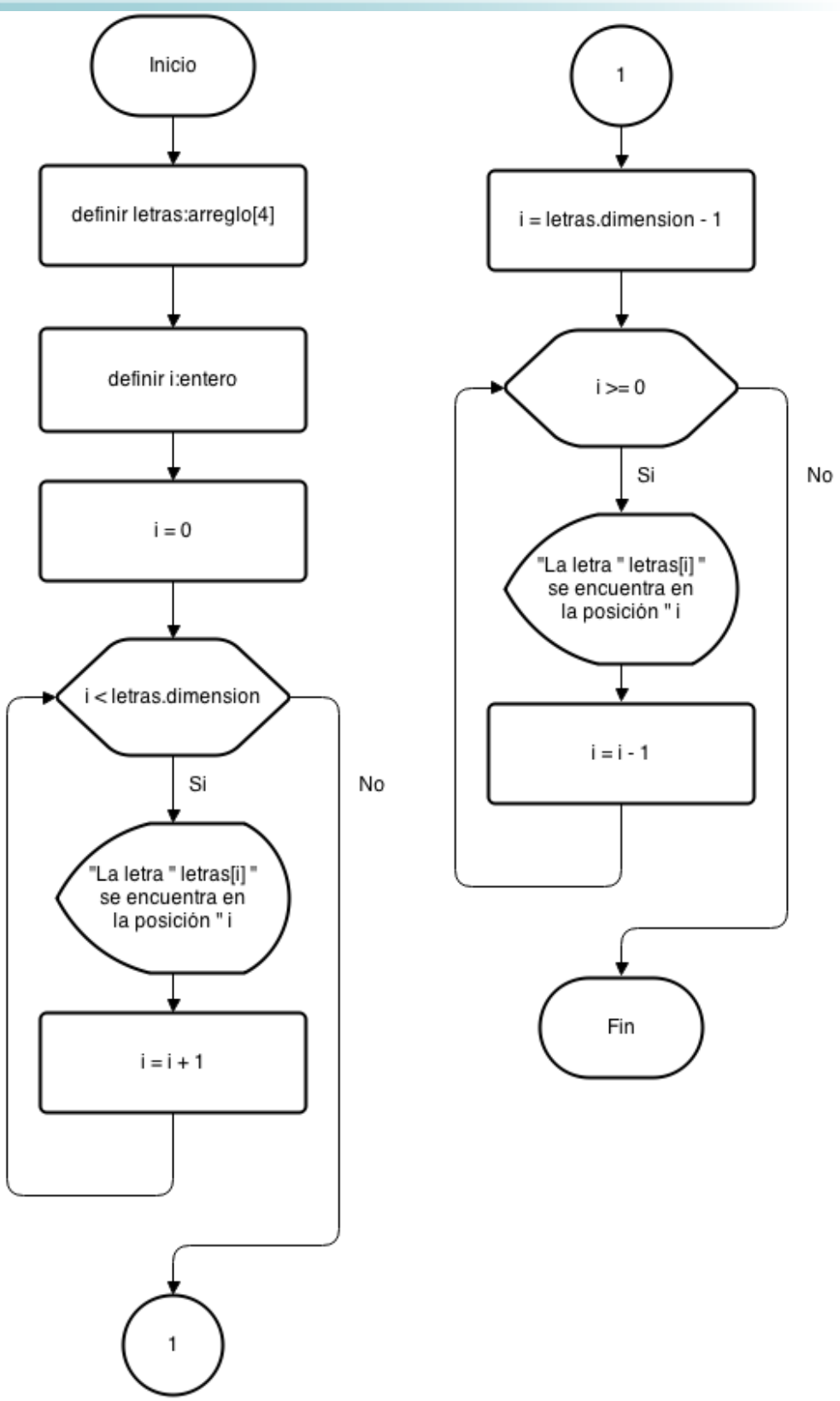
```
definir letras ← arreglo["h","o","l","a"]

// while va a recorrer desde el índice más pequeño que es cero hasta el índice más grande

definir i ← 0;

MIENTRAS (i < letras.dimension)

    imprimir "La letra " letras[i] " se encuentra en la posición " i

    i ← i + 1;

FIN MIENTRAS

// Ahora lo va a recorrer a la inversa desde el índice más grande hasta el índice 0

i ← letras.dimension - 1;

MIENTRAS (i >= 0)

    imprimir "La letra " letras[i] " se encuentra en la posición " i

    i ← i - 1;

FIN MIENTRAS
```

Diagrama de flujo

Utilizando for



Utilizando while



JavaScript

Utilizando for

```
var letras = ["h","o","l","a"];

/* for va a recorrer desde el índice más pequeño que es cero hasta el índice más grande
*/

for (var i = 0; i < letras.length;  i++) {

    alert("La letra " + letras[i] + " se encuentra  en la posición " + i);

}

/* Ahora lo va a recorrer a la inversa desde el índice más grande hasta el índice 0 */

for (var i = letras.length  - 1; i >= 0; i--) {

    alert("La letra " + letras[i] + " se encuentra  en la posición " + i);

}
```

Utilizando while

```
var letras = ["h","o","l","a"];
```

```
/* while va a recorrer desde el índice más pequeño que es cero hasta el índice más grande */

var i = 0;

while (i < letras.length) {

    alert("La letra " + letras[i] + " se encuentra en la posición " + i);

    i = i + 1;

}

/* Ahora lo va a recorrer a la inversa desde el índice más grande hasta el índice 0 */

i = letras.length - 1;

while (i >= 0) {

    alert("La letra " + letras[i] + " se encuentra en la posición " + i);

    i = i - 1;

}
```

1.8. Arreglos y cadenas de caracteres

Hay ocasiones en las que para resolver un problema es más fácil manejar las cadenas de caracteres como arreglos o mostrar el contenido de un arreglo como una cadena de caracteres. Para esto cada lenguaje de programación tiene sus propios métodos.

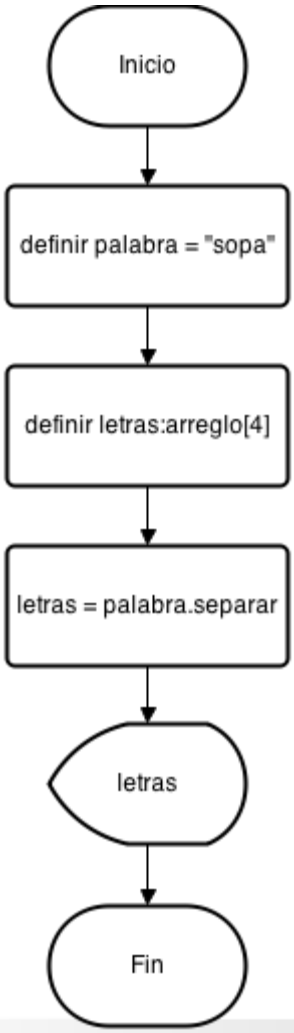
Convertir una cadena de caracteres en un arreglo

Pseudocódigo

En pseudocódigo utilizaremos el método .separar, por ejemplo:

```
definir palabra ← "sopa"

definir letras ← arreglo[4]

letras ← palabra.separar

imprimir letras //Imprimirá el arreglo
```

Diagrama de flujo



JavaScript

En JavaScript se utiliza el método .split(caracter_de_division), el carácter de división indica que se va a dividir la cadena de caracteres cada vez que este carácter se encuentre. Por ejemplo:

```
var palabra = "sopa";

//Utilizamos "" para poder separar cada carácter

var letras = palabra.split("");

alert(letras); //Mostrará lo siguiente: s,o,p,a ya que esta dividiendo la palabra sopa por cada caracter.
```

```
var listado = "Jose,Juan,Maria,Ruth";

//Utilizamos "," para poder separar cada nombre en el arreglo

var nombres = listado.split(",");

alert(nombres[3]); //Mostrará en pantalla Ruth
```

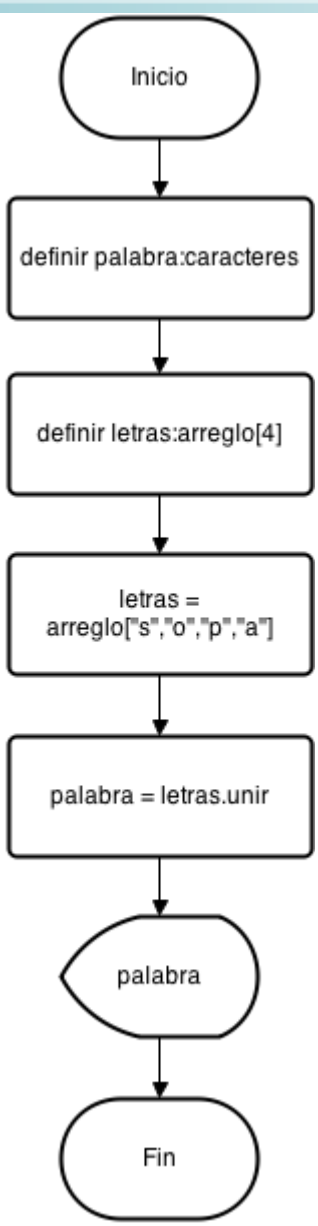
Convertir un arreglo en una cadena de caracteres

Pseudocódigo

En pseudocódigo utilizaremos el método .unir, por ejemplo:

```
definir palabra ← caracteres

definir letras ← arreglo[4]

letras ← arreglo["s","o","p","a"]

palabra ← letras.unir

imprimir palabra // Imprimirá "sopa"
```

Diagrama de flujo



JavaScript

En Javascript se utiliza el método .join (caracter_de_division) el cual concatena todos los elementos de un arreglo en una cadena de caracteres. Opcionalmente se puede especificar un carácter de división para separar cada elemento en la cadena de caracteres resultante. Si no se especifica por defecto se utiliza una coma. Por ejemplo:

```
var letras = ["s","o","p","a"];

var palabra = letras.join(""); //Utilizamos “” como carácter de division

alert(palabra); //Devolvera sopa
```

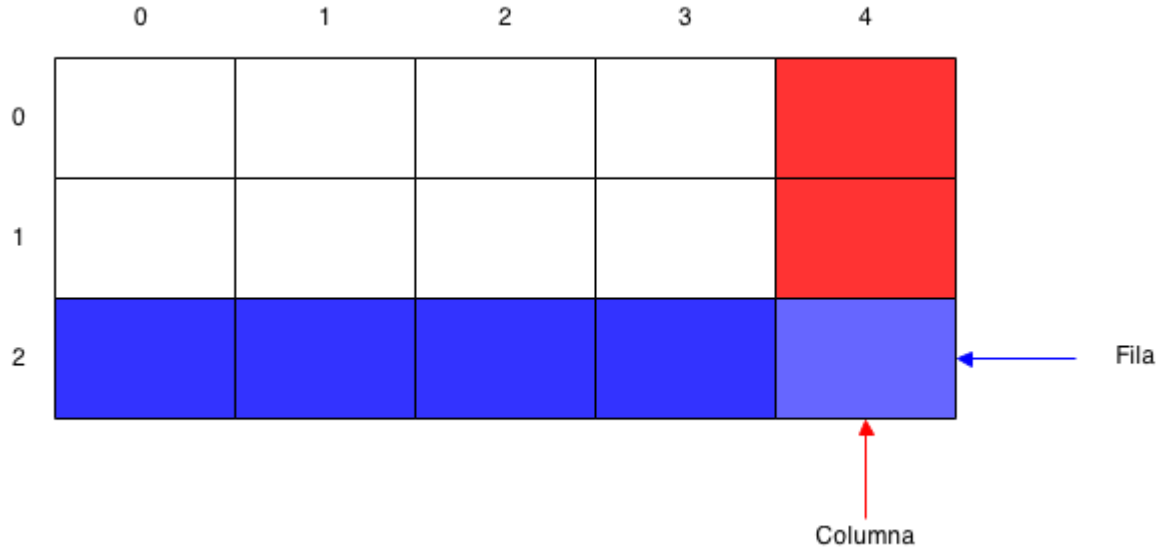
```
var nombres = ["Jose","Juan","Maria","Ruth"];

//Utilizamos "," para poder concatenar cada elemento de la lista en el string

var listado = nombres.join(",");

alert(listado);
```

2. Arreglos multidimensionales

Un arreglo puede contener otros elementos y no exclusivamente literales. En un arreglo también pueden guardarse otros arreglos y de esta manera se pueden crear arreglos de varias dimensiones.

Matrices

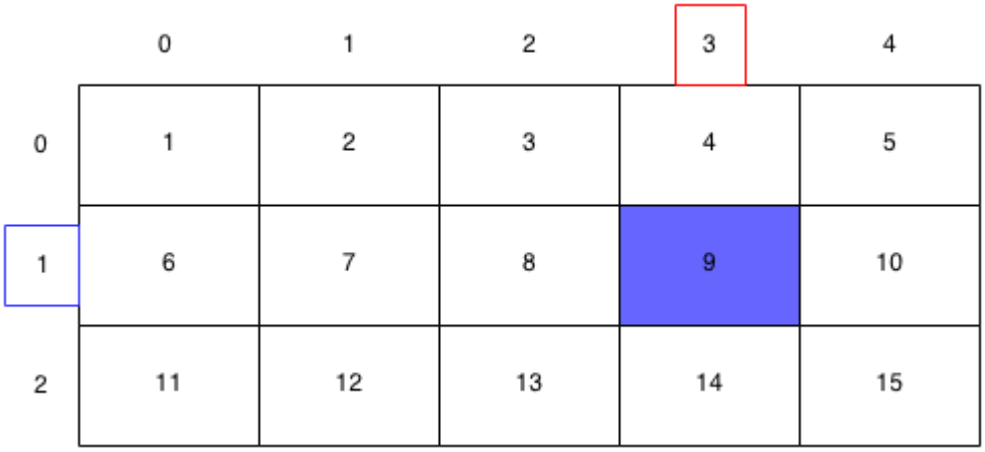
Son arreglos de dos dimensiones con una cantidad de filas y una cantidad de columnas. Cada elemento tiene una posición dentro de la matriz que se identifica mediante dos índices el de su fila y el de su columna.



Si tenemos la siguiente matriz:

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15

Podemos determinar que tiene 3 filas y 5 columnas. Ahora si queremos encontrar la posición en la matriz del literal numérico 9, primero ubicamos el índice de la fila que es 1 y luego el índice de la columna que es 3.

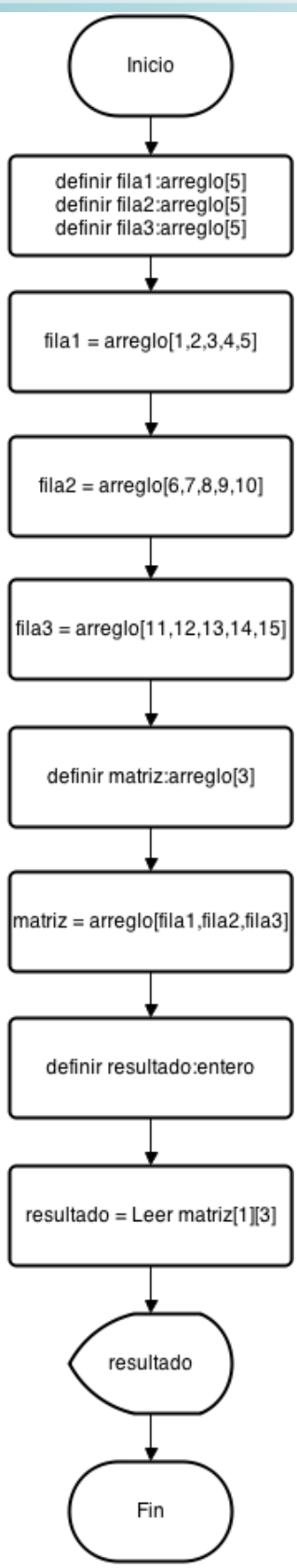


Pseudocódigo

```
definir fila1 ← arreglo[5]

definir fila2 ← arreglo[5]

definir fila3 ← arreglo[5]

fila1 ← arreglo[1,2,3,4,5]

fila2 ← arreglo[6,7,8,9,10]

fila3 ← arreglo[11,12,13,14,15]

definir matriz ← arreglo[3]

matriz ← arreglo[fila1,fila2,fila3]
```

```
// El contenido de matriz es lo siguiente

//              [[ 1 , 2 , 3 , 4 , 5 ],

//               [ 6 ,7 , 8 , 9 ,10],

//               [11,12,13,14,15]

//              ]



//Si deseamos obtener el valor 9, entonces sería de la siguiente manera:



definir resultado ← Leer matriz[1][3]

imprimir resultado
```

Diagrama de flujo


Inicio
definir fila1:arreglo[5]
definir fila2:arreglo[5]
definir fila3:arreglo[5]
fila1 = arreglo[1,2,3,4,5]
fila2 = arreglo[6,7,8,9,10]
fila3 = arreglo[11,12,13,14,15]
definir matriz:arreglo[3]
matriz = arreglo[fila1,fila2,fila3]
definir resultado:entero
resultado = Leer matriz[1][3]
resultado
Fin

Javascript

```
var fila1 = [1,2,3,4,5];

var fila2 = [6,7,8,9,10];

var fila3 = [11,12,13,14,15];



var matriz = [fila1,fila2,fila3];



/* El contenido de matriz es lo siguiente

            [[ 1 , 2 , 3 , 4 , 5 ],

             [ 6 ,7 , 8 , 9 ,10],

             [11,12,13,14,15]

            ]

*/



//Si deseamos obtener el valor 9, entonces sería de la siguiente manera:
```

```
var resultado = matriz[1][3];

alert(resultado);

/* Esto es porque accedemos a la posición 1 del arreglo matriz que contiene el valor del arreglo fila2. Con el segundo índice accedemos a la posición 3 del arreglo que nos devuelve el primer índice, dando como resultado el número nueve.*/
```

Recorriendo una matriz

Para recorrer una matriz también se utilizan estructuras iterativas. Solo que ahora se debe recorrer arreglos dentro de un arreglo.

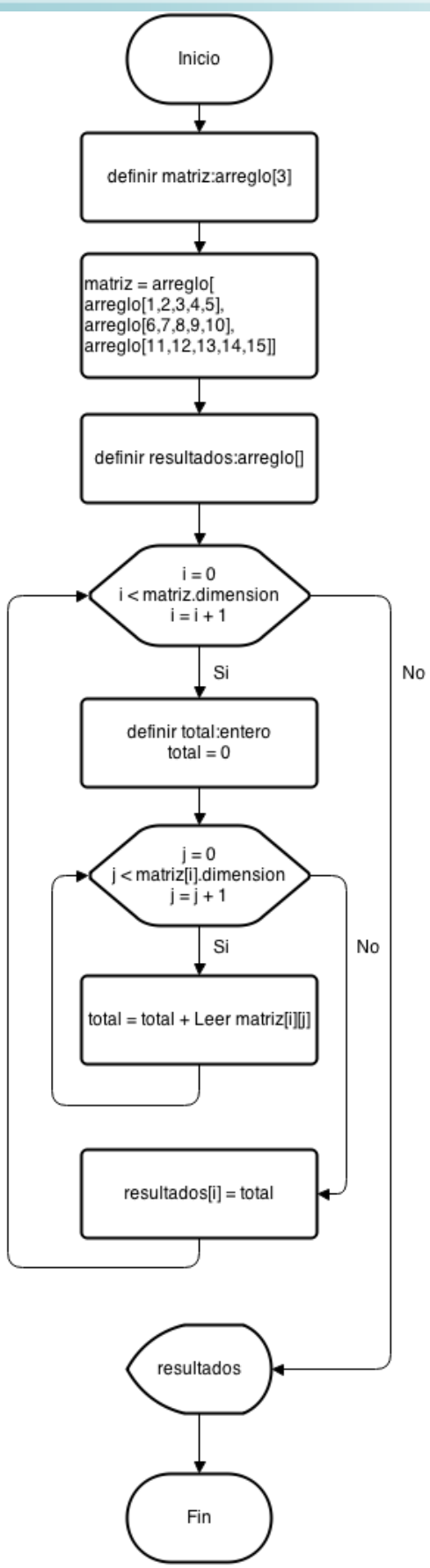
Ejemplo: Calcular la suma de cada fila de la matriz y guardar los resultados en un arreglo.

Pseudocódigo

```
definir matriz ← arreglo[3]

matriz ← arreglo[arreglo[1,2,3,4,5],arreglo[6,7,8,9,10],arreglo[11,12,13,14,15]]

definir resultados ← arreglo[]

// Con el siguiente for vamos a recorrer las filas

PARA (i ← 0; MIENTRAS i < matriz.dimension; i++)

  definir total ← 0
```

```
// Con el siguiente for vamos a recorrer las columnas

PARA (j ← 0; MIENTRAS  j < matriz[i].dimension;  j++)

  total ← total + (Leer matriz[i][j])

FIN PARA

resultados[i] ← total

FIN PARA

imprimir resultados
```

Diagrama de flujo


Inicio
definir matriz:arreglo[3]
matriz = arreglo[
arreglo[1,2,3,4,5],
arreglo[6,7,8,9,10],
arreglo[11,12,13,14,15]]
definir resultados:arreglo[]
i = 0
i < matriz.dimension
i = i + 1
Si
No
definir total:entero
total = 0
j = 0
j < matriz[i].dimension
j = j + 1
Si
No
total = total + Leer matriz[i][j]
resultados[i] = total
resultados
Fin

Javascript

```
var matriz = [[1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15]];

var resultados = [];

// Con el siguiente for vamos a recorrer las filas

for (var i = 0; i < matriz.length;  i++) {

    var total = 0;



    // Con el siguiente for vamos a recorrer las columnas

    for (var j = 0; j < matriz[i].length;  j++) {

        total = total + matriz[i][j];

    }

    resultados[i] = total;

}

// El contenido del arreglo resultados es [15,40,65]

alert(resultados);
```

Descargo de responsabilidad

La información contenida en este documento descargable en formato PDF o PPT es un reflejo del material virtual presentado en la versión online del curso. Por lo tanto, su contenido, gráficos, links de consulta, acotaciones y comentarios son responsabilidad exclusiva de su(s) respectivo(s) autor(es) por lo que su contenido no compromete al área de e-Learning del Departamento GES o al programa académico al que pertenece.

El área de e-Learning no asume ninguna responsabilidad por la actualidad, exactitud, obligaciones de derechos de autor, integridad o calidad de los contenidos proporcionados y se aclara que la utilización de este descargable se encuentra limitada de manera expresa para los propósitos educacionales del curso.