DEFINICIONES BÁSICAS DE LAS ESTRUCTURAS DE DATOS

En programación, una estructura de datos es una forma de organizar un conjunto de datos elementales con el objetivo de facilitar su manipulación. Un dato elemental es la mínima información que se tiene en un sistema.

Una estructura de datos define la organización e interrelación de éstos y un conjunto de operaciones que se pueden realizar sobre ellos. Las operaciones básicas son:

- Adicionar un nuevo valor a la estructura.
- Borrar un valor de la estructura.
- Búsqueda, encontrar un determinado valor en la estructura para realizar una operación con este valor, en forma SECUENCIAL o BINARIO (siempre y cuando los datos estén ordenados)

Otras operaciones que se pueden realizar son:

- Ordenamiento, de los elementos pertenecientes a la estructura.
- Apareo, dadas dos estructuras originar una nueva ordenada y que contenga a las apareadas.

Existen dos clases de tipos de datos: Simples (sin estructuras) y Compuestos (estructurados).

- **Datos simples**
  - estándar
    - Entero (integer, int)
    - Real (float, double)
    - Carácter (char)
    - Lógico (boolean)
  - definido por el programador
    - Subrango (subrange)
    - Enumerativo
- **Datos estructurados**
  - estáticos
    - Array (vector/matriz)
    - Registro
    - Archivo (fichero)
    - Conjunto
    - Cadena (string)
  - dinámicos
    - Lista (pila/cola)
    - Lista enlazada
    - Árbol
    - Grafo

Los tipos de datos *simples o primitivos* significa que no están compuestos de otras estructuras de datos; lo más frecuentes y utilizados por casi todos los lenguajes de programación son: **enteros, reales, carácter y lógico**. Los tipos de datos compuestos están construidos basados en tipos de datos primitivos; el ejemplo más significativo es la cadena (string) de caracteres.

Los tipos de datos simples pueden ser organizados en diferentes estructuras de datos: *estáticas y dinámicas*.

Las **estructuras de datos estáticas** son aquéllas en las que el tamaño ocupado en memoria se define antes de que el programa se ejecute y no puede modificarse ese tamaño durante la ejecución del programa. Estas estructuras están en casi todos los lenguajes de programación: *array* (vectores y matrices), *registros*, *etc*.

Las **estructuras de datos dinámicas** no tienen las limitaciones o restricciones en el tamaño de memoria ocupada, como si lo tiene las estructuras estáticas.

Una característica importante que diferencia a los tipos de datos es la siguiente: los tipos de datos simples tienen como característica común que cada variable representa un elemento; los tipos de datos estructurados, tienen como característica común que un **identificador** (nombre) puede representar múltiples datos individuales, pudiendo cada uno de éstos ser referenciado independientemente.

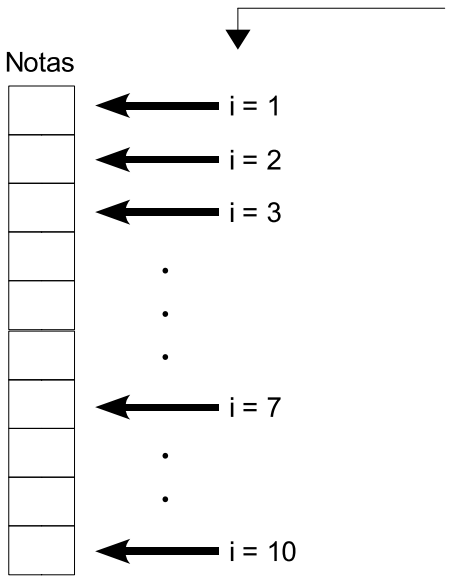
La elección del tipo de estructura de datos idonea a cada aplicación dependerá esencialmente del tipo de aplicación.

ARRAYS UNIDIMENSIONALES O VECTORES

Un arreglo unidimensional(vector) es un tipo de datos estructurado que está formado de una colección finita y ordenada de datos del mismo tipo. Un array puede estar compuesto de todos sus elementos de tipo de dato entero, otro puede tener todos sus elementos de tipo de dato reales, etc.

A los arrays unidimensionales se le conoce también como arreglos unidimensionales. Un array unidimensional es una estructura que se le asigna un nombre y a cada elemento que tiene ese array se asocia con un único **índice**. El tipo de acceso a los arreglos unidimensionales es el acceso directo, es decir, podemos acceder a cualquier elemento del arreglo sin tener que consultar a elementos anteriores o posteriores, esto mediante el uso de un **índice** para cada elemento del arreglo que nos da su posición relativa

Ejemplo 1:
Se tiene un array unidimensional llamado **Notas** de 10 elementos; este vector va almacenar elementos de tipo de dato entero.

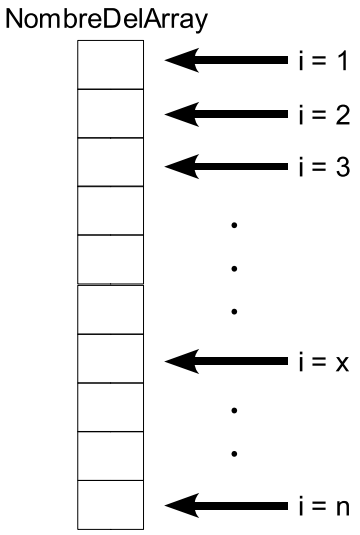


El índice del array unidimensional es la variable **i**, que va a tomar valores desde el 1 hasta la cantidad máxima de elementos, que para este ejemplo es 10.

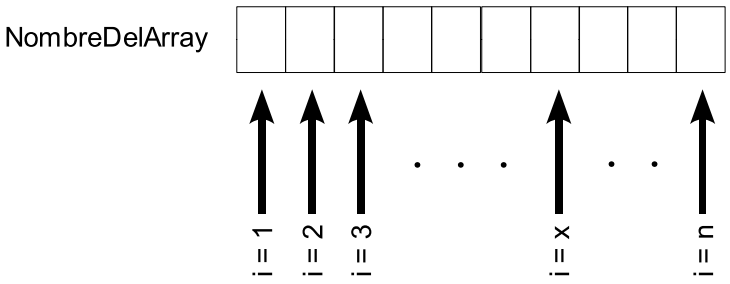
Cada valor del índice representa a un elemento del arreglo, es decir,
el índice i=1 representa al 1er elemento,
el índice i=2 representa al 2do elemento,
el índice i=3 representa al 3er elemento y asi sucesivamente hasta el índice i=10 que representa al ultimo elemento del array.

En base al ejemplo anterior podemos definir la *sintaxis* de un array unidimensional:

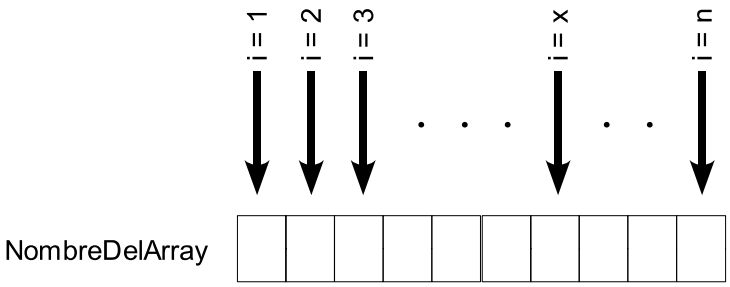
Sintaxis: Representación gráfica de un array unidimensional



También se puede representar gráficamente de la siguiente forma:



y de esta ultima también:



INGRESO DE DATOS

El ingreso de datos al arreglo se hace de la siguiente forma:

Sintaxis:

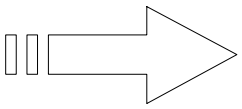
```
NombreDelArray [ índice ] = ValorDelElemento

Leer ( NombreDelArray [ índice ] )
```

Asignación de valores al arreglo unidimensional

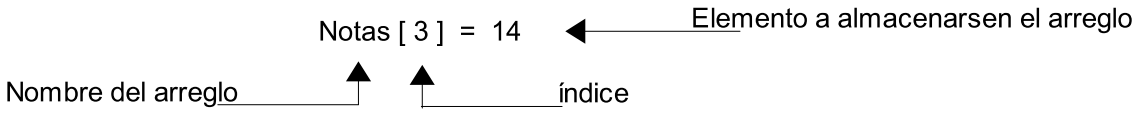
Del ejemplo 1, haremos el ingreso de los elementos al arreglo **Notas**:

```
Notas[ 1 ] = 15
Notas[ 2 ] = 18
Notas[ 3 ] = 14
      .
      .
      .
Notas[ 7 ] = 17
      .
      .
Notas[ 10 ] = 20
```



Notas
15
18
14
.
.
.
17
.
.
20

Podemos apreciar que para ingresar un elemento al arreglo se debe determinar en que lugar se almacenará, para ello utilizamos el índice. Por ejemplo:



Para este ejemplo la nota 14 se almacenará en el índice 3, pero se puede almacenar en otro índice: Notas[5] = 14, o en otro: Notas[9] = 14. En la programación podemos definir en que índice queremos almacenar un elemento.

Lectura de datos al arreglo unidimensional

Si queremos ingresar valores al arreglo por intermedio del teclado, haremos uso de la función Leer.

Ejemplos:

```
Leer (Notas[1])     En esta instrucción ingresamos una nota en el índice 1.
Leer (Notas[7])     En esta otra instrucción ingresamos una nota en el índice 7.
```

y asi para todos los índices del arreglo.

Para el **llenado** de un arreglo unidimensional se hace uso de las estructuras repetitivas, la más utilizada es la estructura FOR. Por ejemplo tenemos un arreglo llamado arrayPrecios de 10, 20, 100 ó 1000 elementos donde se almacenarán los precios de diferentes productos. Para este tipo de casos se debe utilizar las estructuras repetitivas para su llenado de datos.

A continuación se muestra en Pseudocódigo la forma de ingresar los 1000 datos de precios al arreglo arrayPrecios.

```
desde i = 1 hasta 1000 hacer
    Leer (arrayPrecios[i])
fin desde
```

MOSTRAR DATOS

Mostrar los datos de un arreglo unidimensional se hace de la siguiente forma:

Sintaxis:

```
NombreDelArray [ índice ]
```

Del ejemplo 1, mostraremos los elementos del arreglo **Notas**:

```
Mostrar (Notas[ 1 ])
```

Esta instrucción muestra el valor 15

```
Mostrar (Notas[ 2 ])
```

Esta instrucción muestra el valor 18

```
Mostrar (Notas[ 7 ])
```

Esta instrucción muestra el valor 17, y asi sucesivamente cada instrucción mostrará los elementos del arreglo.

De la misma forma como se ha hecho el llenado de datos del arreglo arrayPrecios, también serán necesarias las estructuras repetitivas para mostrar sus elementos.

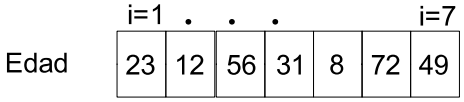
```
desde i = 1 hasta 1000 hacer
    Mostrar (arrayPrecios[i])
fin desde
```

Nota:

Y esta misma estructura repetitiva nos servirá para poder hacer los cálculos que deseamos realizar con los elementos del arreglo.

OPERACIONES CON VECTORES

Para llevar acabo estas operaciones necesitaremos de un arreglo. A continuación se muestra un arreglo llamado Edad de 7 elementos que almacena datos de tipo entero.



OPERACION	RESULTADO
Leer (Edad [1])	Lee un elemento por teclado (23) y lo almacena en el Arreglo Edad dentro del índice 1. Y así sucesivamente ingresamos los demas valores al arreglo.
Mostrar (Edad [4])	Muestra el elemento que existe dentro del índice 4: el valor a mostrar es 31. Y de la misma forma podemos mostrar todos los elementos del arreglo.
P = Edad [4] * 2	Se multiplica el elemento que existe dentro del índice 4 con el número 2 y el resultado se asigna a la variable P. **P = 31 * 2** **P = 62**
S = Edad [2] + Edad [5]	En la variable S se almacena la suma de los elementos que se encuentran en el índice 2 y 5. **S = 12 + 8** **S = 20**
Edad [5] = Edad [3] - Edad [1]	En el índice 5 del arreglo Notas se almacena la resta de los elementos que se encuentran en el índice 3 y 1. **Edad [5] = 56 - 23** **Edad [5] = 33**
Edad [6] = (Edad [4] * 4 + 8) / Edad [2]	En el índice 6 del arreglo Notas se almacena el resultado de la expresión. **Edad [6] = (31*4 + 8)/12** **Edad [6] = 11**
Edad [7] = Edad [5] + Edad [6]	En el índice 7 del arreglo Notas se almacena la suma de los elementos que se encuentran en el índice 5 (Observar que en la 5ta operación Nota[5] toman un nuevo valor) y el índice 6. **Edad [7] = 33 + 11** **Edad [7] = 44**
Edad [2] = Edad [2] / Edad [2]	En el índice 2 del arreglo Notas se almacena el resultado de la división del elemento que se encuentra en el índice 2 entre su mismo valor. **Edad [2] = 12 - 12** **Edad [2] = 1**
Edad [4] = Edad [7] + Edad [6] - Edad [5] + Edad [4] - Edad [3] + Edad [2] - Edad [1]	En el índice 4 del arreglo Notas se almacena la suma y resta de los elementos que se encuentran en los índices 7, 6, 5 y 4. **Edad [7] = 44 + 11 – 33 + 31 – 56 + 1 - 23** **Edad [7] = - 25**