VECTORES

ARREGLOS UNIDIMENSIONALES

Parte 1

TIPOS DE DATOS PRIMITIVOS

Variables de tipo primitivas (byte, short, int, long, float, double…) las cuales como característica principal solo permiten almacenar un único valor cada vez (int x=9: x almacena solo el valor 9)…

ARREGLOS

Un array (arreglo) es una estructura de datos que contiene una colección de datos del mismo tipo, en lugar de declarar cada dato de manera independiente.

Por medio de los arreglos veremos cómo crear un elemento que nos permite definir una “variable” que contenga diferentes datos del mismo tipo asociados al mismo identificador.

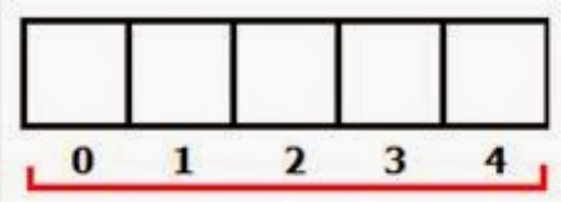
Básicamente se tienen 3 categorías de arreglos, los arreglos unidimensionales (vectores), los arreglos bidimensionales (matrices) y los arreglos multidimensionales (varias dimensiones),

VECTORES

Los arreglos unidimensionales también son conocidos como vectores, es muy común que estos sean conocidos solo como arreglos simplemente refiriéndose a ellos como los de una sola dimensión, se componen de una fila y una o más columnas, las cuales representan espacios de memoria donde se pueden almacenar datos.

Para poder trabajar con arreglos, se debe crear el arreglo definiendo cual es el tipo de datos que va a contener y cuál es el tamaño del mismo (cantidad de datos que puede almacenar), de ese modo si definimos que el arreglo va a ser de tipo byte, solo podrá almacenar datos de tipo byte, no se puede mezclar en dicho arreglo datos int, short o long por ejemplo.

La creación de un arreglo se realiza en 3 pasos básicos: Declaración, construcción e inicialización.



DECLARACION DE VECTORES

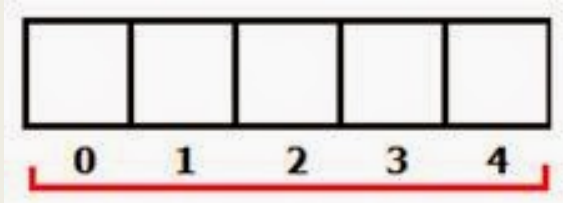
Los arreglos se identifican porque al momento de su creación se utilizan corchetes [ ] , al usarlos java automáticamente identifica que se va a trabajar con arrays, teniéndose dos formas generales para su creación.

<tipoDato> identificador[ ]; Ej: int arregloDeEnteros[ ];

O también…

<tipoDato> [ ] identificador; Ej: int[ ] arregloDeEnteros;

Donde tipoDato define el tipo de dato de cada uno de los valores que puede contener el arreglo.



CONSTRUCCION DE VECTORES

Después de haber declarado el array se puede construir e inicializar de dos maneras. **Forma1.**

la primera se usa cuando inicialmente no sabemos cuáles son los valores que va a contener el arreglo, ya que luego serán ingresados, se crea con la siguiente estructura:

Identificador = new <tipoDato> [tamaño]; Ej. arregloDeEnteros = new int[5];

En el ejemplo anterior se puede observar como el arregloDeEnteros declarado previamente, ahora es creado con un tamaño de 5 posiciones de memoria, estas posiciones corresponden a un espacio donde se pueden almacenar 5 datos de tipo int, iniciando desde la posición 0 a la posición tamaño-1, para este caso siendo el tamaño 5, entonces se inicia de la posición 0 hasta 4, siendo cada posición una columna del arreglo.

INICIALIZACION DE VECTORES

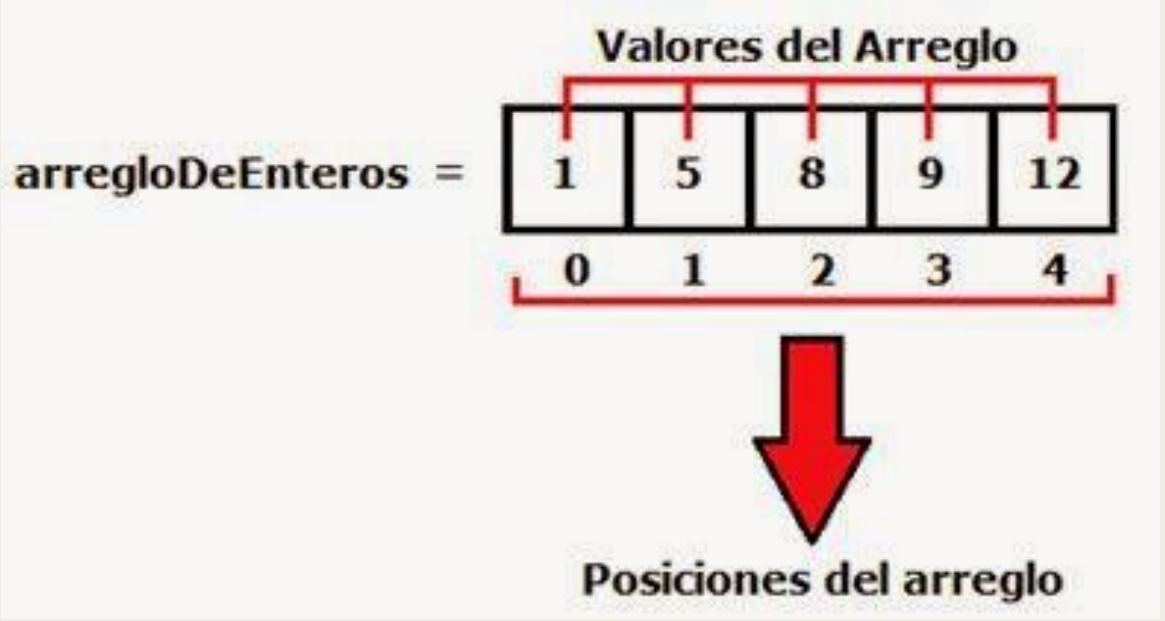
Para llenar el arreglo se debe tener presente el tamaño asignado pues cada posición almacenará un valor del tipo de dato del arreglo, el llenado se realiza de la siguiente manera:

identificador[posición]=dato;

Sabemos que el identificador corresponde al nombre del arreglo, posición a alguna de las casillas del arreglo y dato el valor a asignar, que debe ser del tipo de dato del arreglo.

Ej:

```
arregloDeEnteros[0]=1;
arregloDeEnteros[1]=5;
arregloDeEnteros[2]=8;
arregloDeEnteros[3]=9;
arregloDeEnteros[4]=12;
```



INICIALIZACION DE VECTORES

Como se mencionó, es muy importante tener presente tanto el tipo de dato como el tamaño del arreglo, ya que es muy común que se presenten los siguientes errores:

- Almacenar otros tipos de datos diferentes al tipo de dato del arreglo

arregloDeEnteros[0]=”1”; //Error, si es numérico no puede estar como String

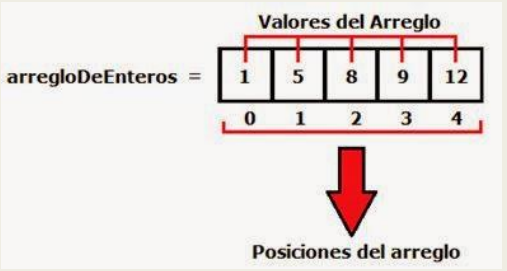
arregloDeEnteros[1]=”hola”; //Error, el arreglo es de tipo int, y se quiere almacenar un String

- Almacenar datos en una posición diferente al tamaño del arreglo, esto indicaría que hay error en el índice del arreglo.

arregloDeEnteros[-1]=3; //Error, el tamaño del arreglo es de 5 pero las posiciones van de 0 a 4

arregloDeEnteros[5]=8; //Error, el tamaño del arreglo es de 5 pero las posiciones van hasta 4

arregloDeEnteros[8]=8; //Error, el tamaño del arreglo es de 5 pero las posiciones van hasta 4



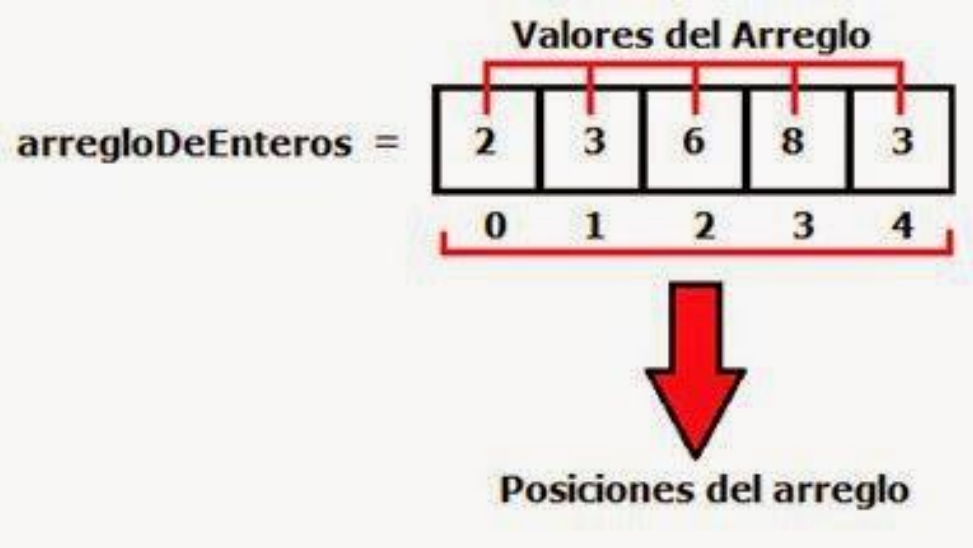
INICIALIZACION DE VECTORES

Forma 2.
Esta forma se usa cuando sabemos con exactitud cuáles son los valores que va a contener el arreglo, aquí el proceso de construcción e inicialización se hace directo y se realiza de la siguiente manera:

tipoDato[ ] Identificador = {valor1, valor2, valor3, valor4};

El identificador corresponde al nombre del arreglo, las llaves corresponden a lo que va a contener el arreglo, cada posición se identifica separada por comas (,) y los valores corresponden a los datos que se van a almacenar.

Ej: int[ ] arregloDeEnteros= {2, 3, 6, 8,3};



Como se puede observar no fue necesario indicar cuál es el tamaño del arreglo, ya que java identifica el tamaño gracias a las posiciones y cantidad de valores separados por coma.

ACCESO DE LOS DATOS DE VECTORES

Para acceder a la información almacenada dentro de un arreglo se debe tener presente el nombre del arreglo, tamaño y el tipo de datos que contiene.

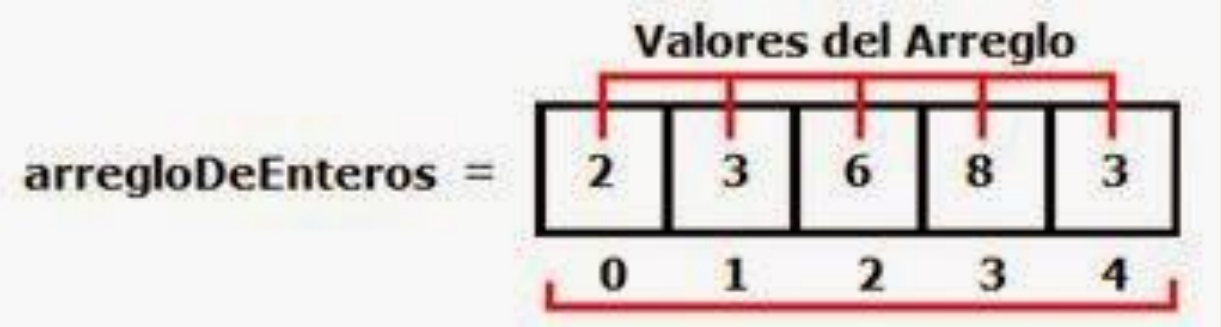
Por ejemplo, si queremos almacenar un dato de un arreglo en una variable, la forma de acceder es por medio de un índice que corresponde a la posición del valor a obtener:

variable = Identificador[posición];

donde la variable corresponde a una variable del tipo de dato que se quiere almacenar, el identificador corresponde al nombre del arreglo y la posición a alguno de los valores entre 0 y tamaño-1

Tomando el arreglo anterior como ejemplo, queremos obtener el valor en la posición 3 del arreglo.

Entonces:
int dato=arregloDeEnteros[3];
Por lo tanto dato, almacenará el valor 8.



Ejemplo Forma 1:

```
1  String arregloA[];//Declaración del arreglo
2    arregloA=new String[4];//Creación o construcción del arreglo
3
4    //Llenado del arreglo
5    arregloA[0]="hola";//inicialización arreglo en la posición 0
6    arregloA[1]="casa";//inicialización arreglo en la posición 1
7    arregloA[2]="perro";//inicialización arreglo en la posición 2
8    arregloA[3]="oso";//inicialización arreglo en la posición 3
9
10   //Obteniendo información del arreglo
11   System.out.println("Valor arreglo en la posición 0: "+arregloA[0]);
12   System.out.println("Valor arreglo en la posición 1: "+arregloA[1]);
13   System.out.println("Valor arreglo en la posición 2: "+arregloA[2]);
14   System.out.println("Valor arreglo en la posición 3: "+arregloA[3]);
```

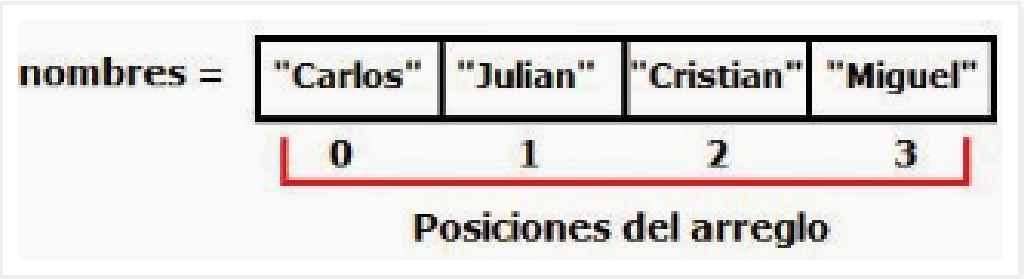
El ejemplo anterior crea un arreglo de tipo String y tamaño 4 con posiciones de 0 a 3

Ejemplo Forma 2:

```java
String arregloA[];//Declaración del arreglo
  //Declaración, Inicialización y Creación del arreglo
  String nombres[]={"Carlos","Julian","Cristian","Miguel"};

  //Obteniendo información del arreglo
  System.out.println("Valor arreglo en la posición 0: "+nombres[0]);
  System.out.println("Valor arreglo en la posición 1: "+nombres[1]);
  System.out.println("Valor arreglo en la posición 2: "+nombres[2]);
  System.out.println("Valor arreglo en la posición 3: "+nombres[3]);
```

El ejemplo anterior crea un arreglo de tipo String y tamaño 4 con posiciones de 0 a 3



LLENADO Y CONSULTA POR MEDIO DE CICLOS

Cuando se desea llenar un arreglo de muchas posiciones, los ciclos juegan un papel muy importante ya que nos permitirán hacer este proceso más dinámico, pues podemos recorrer cada posición usando la variable de incremento, tanto para asignar como para obtener.

Teniendo en cuenta que siempre cuando asignamos o consultamos datos del arreglo, debemos indicar cuál es la posición o índice que vamos a usar, la posición podemos trabajarla como una variable que toma cada uno de los valores posibles que a tomar.

Ej: *arreglo[posición]=valor //*asignación valor en el arreglo
*Variable= arreglo[posición] //*asignación valor del arreglo en la variable

```java
public class Principal {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Scanner entrada=new Scanner(System.in);
		int[] a=new int[0];
		int op,T=0;
		do{   System.out.println("0.Salir");
			System.out.println("1. Generar vector");
			System.out.println("2. Mostrar vector");
			System.out.println("opcion");
			op=entrada.nextInt();
			switch(op)
			{     case 1: System.out.println("Tamaño del vector: ");
					T=entrada.nextInt();
					a=new int[T];
					leerElementos(a,T);
					 break;
				case 2: System.out.println("Vector:");
					 mostrar(a,T);
					 break;
			}
		}while(op!=0);
	}
	 public static void leerElementos(int M[],int N)
	{     Scanner entrada=new Scanner(System.in);
		for(int f=0;f<N;f++)
		 {   System.out.println("Elemento" +f+" : ");
			 M[f]= entrada.nextInt();
		 }
	}
	 public static void mostrar(int M[],int N)
	{     for(int f=0;f<N;f++)
		 {
			System.out.print("  "+M[f]);
		 }
		System.out.println(" ");
	}
}
```

a	op	T

M	N	f

M	N	f	Pantalla