Arrays

-Arrays

Un array es un conjunto de elementos de un mismo tipo,agrupados bajo un único nombre,se accede a cada valor indicando su posición en el conjunto,empezando por el 0.

Declaración de array

```
int [] numeros = new int [tamaño del array];
```

Acceso a posiciones del array

```
int [] numeros = new int [tamaño del array];
numeros[0]=valor;
numeros[1]=valor;
numeros[2]=valor;
…
Console.WriteLine(numeros[5]);
```

Dar valores iniciales a un array

Opción A

```
int [] numeros ={1, 2, 3, 4};
```

Opción B

```
Int [] numeros = new int [4] { 1, 2, 3, 4};
```

Recorrer los elementos de un array

Opción A

Bucle hasta tamaño puesto a mano,su inconveniente es que si cambias el tamaño te toca cambiarlo en todas las partes del código donde estuviera.

```
Int [] numeros = new int [10];

for (int i = 0; i < 10; i++)
{
      Console.WriteLine(numeros[i]);
}
```

Opción B

Definir una constante con el tamaño.

```
Int TAM = 10;
Int [] numeros = new int [TAM];

for (int i = 0; i < TAM; i++)
{
      Console.WriteLine(numeros[i]);
}
```

Opción C

Utilizar la propiedad **Length** del array

```
Int [] numeros = new int [10];

for (int i = 0; i < numeros.Length; i++)
{
      Console.WriteLine(numeros[i]);
}
```

Buscar en un array

Para ver si un dato existe,debemos comprobar cada posición del array,una por una,si solo nos interesa saber si aparece o no,una vez lo encuentre acabará el programa,sin embargo,si queremos saber cuantas veces aparece debemos recorrer todo el array.

```
bool encontrado = false;
int i = 0;
while (!encontrado && i < numeros.Length)
{
    if (numeros[i] == 15)
        encontrado = true;
    else
        i++;
}
```

Buscar máximos y mínimos

Ponemos el primer valor del array como máximo/mínimo para evitar darle un valor arbitrario inicial,posteriormente vamos evaluando cada valor del valor en busca de un valor que lo sobrepase,en cuyo caso ocupará su lugar.

```
int maximo = numeros[0];
```

```
for (int i = 1; i < numeros.Length; i++)
    if (numeros[i] > maximo)
    {
        maximo = numeros[i];
    }
```

Arrays sobredimensionados

Son arrays definidos con más capacidad de la que realmente necesitan,permiten ir añadiendo o eliminando elementos del array de forma dinámica,usaremos una variable auxiliar para almacenar el número de valores almacenados en el array.

```
string[] textos = new string[100];
int cantidad = 0;
```

ARRAYS SOBREDIMENSIONADOS: AÑADIR ELEMENTOS AL FINAL

Colocamos el valor en la posición final (la misma que la auxiliar cantidad).

```
if (cantidad < textos.Length)
```

```
{
    textos[cantidad] = "Nuevo texto";
    cantidad++;
}
```

ARRAYS SOBREDIMENSIONADOS: AÑADIR ELEMENTOS EN UNA POSICIÓN

Buscamos la posición donde queremos añadir el valor,si hay huecos libres,desplazamos todos los valores posteriores a ese un hueco a la derecha y añadimos el valor en la posición deseada.

```
if (cantidad < textos.Length)
{
    for (int i = cantidad; i >= posicion; i--)
        textos[i] = textos[i-1];
    textos[posicion] = "Hola";
    cantidad++;
}
```

ARRAYS SOBREDIMENSIONADOS: BORRAR UN ELEMENTO DE UNA POSICIÓN

Buscamos la posición donde queremos borrar(si hay elementos en dicha posición),desplazaremos desde esa posición hasta el final todos un valor hacia atrás para tapar el hueco y decrementamos el contador.

```
if (cantidad > 0 && posicion >= 0 && posicion < cantidad)
{
   for (int i = posicion; i < cantidad - 1; i++)
      numeros[i] = numeros[i+1];
   cantidad--;
```

ARRAYS BIDIMENSIONALES

Permite estructurar la información en filas(primera dimensión) y columnas(segunda dimensión) para cada fila.

Alternativas:

Arrays rectangulares (matrices o tablas).

Arrays de arrays.

ARRAYS RECTANGULARES

Definición

Definimos cuántas filas y columnas tiene el array, separadas por comas.

```
Int [,] tabla = new int [filas, columnas];
```

Acceso

Accedemos a cada posición indicando su fila y columna, separadas por comas.

```
tabla[1, 3] = 12;
Console.WriteLine(tabla[0, 4]);
```

Recorrido

Necesitamos un doble bucle anidado (el primero para las filas y el segundo para las columnas, o viceversa).

Usamos la instrucción **GetLength(n)** del array para obtener el tamaño de la dimensión n, y así saber cuándo parar.

array.GetLength(0) para las filas.

array.GetLength(1) para las columnas.

```
for (int i = 0; i < tabla.GetLength(0); i++)
{
    for (int j = 0; j < tabla.GetLength(1); j++)
        Console.WriteLine(tabla[i, j] + " ");
    Console.WriteLine();
}
```

ARRAYS DE ARRAYS

Definición

Definimos cuántas filas vamos a crear en una pareja de corchetes.

Para cada fila, indicamos qué tamaño va a tener en la segunda pareja de corchetes.

```
int[][] datos = new int[cantidad arrays][];
datos[0] = new int[tamaño array];
datos[1] = new int[tamaño array];
datos[2] = new int[tamaño array];
```

Acceso

Colocamos en los primeros corchetes la fila que nos interesa, y en los segundos corchetes la columna dentro de esa fila.

```
datos[fila][columna] = valor;
Console.WriteLine(datos[fila][columna]);
```

Recorrido

Usamos un doble bucle anidado, y la propiedad **Length** para saber el tamaño de cada dimensión.

```
// datos.Length = número de filas
for (int i = 0; i < datos.Length; i++)
{
   // datos[i].Length = columnas de la fila i
   for (int j = 0; j < datos[i].Length; j++)
      {
            Console.Write(datos[i][j] + " ");
      }
   Console.WriteLine();
}
```