utel UNIVERSIDAD®

Dirección Académica

Programación estructurada

Ciclo Mayo 2022

CONTENIDO

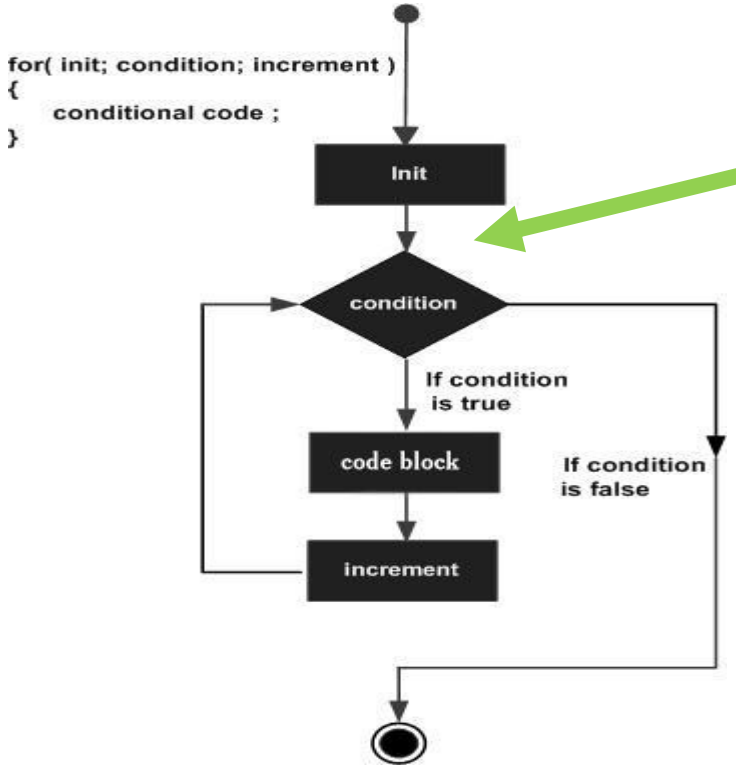
Orden del día

- Repaso clase anterior
- Ejemplo de repaso
- Estructuras de datos en C#
- Ejemplo en C#
- Ejercicio de repaso

REPASO CLASE ANTERIOR

Repaso clase anterior

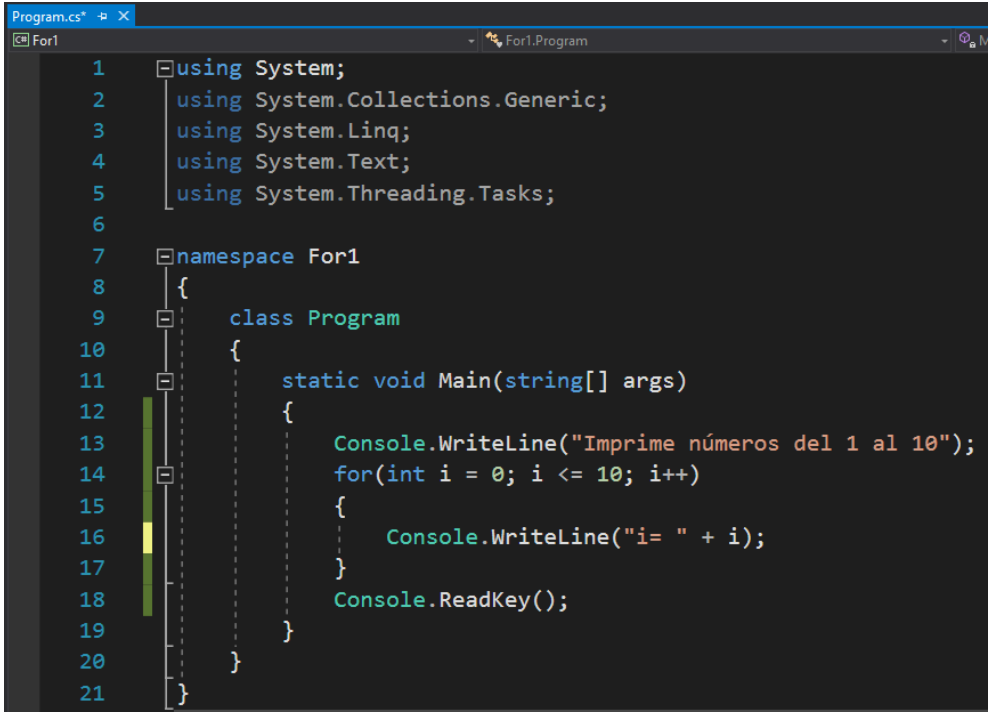
Estructuras de repetición For




```
Console.WriteLine(«Buenos dias»);

for(i = 0 ; i < 10 ; i ++)

{

Console.WriteLine(i);

if(i == 5)

{

break;

}

}

Console.ReadKey();
```

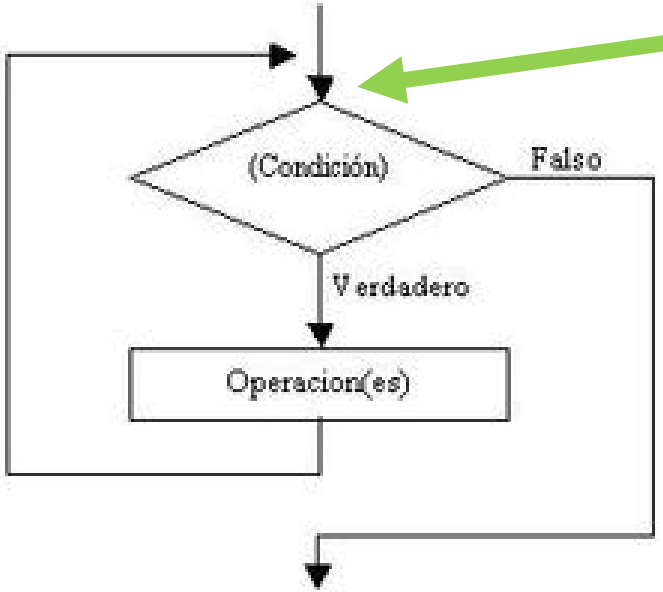
Ejemplos

Programa ejemplo, que imprime los números comprendidos del 1 al 10:



Estructuras de repetición

Estructuras de repetición While



Sintaxis:

```
while (condición)
    CONJUNTO DE INSTRUCCION(ES)
```

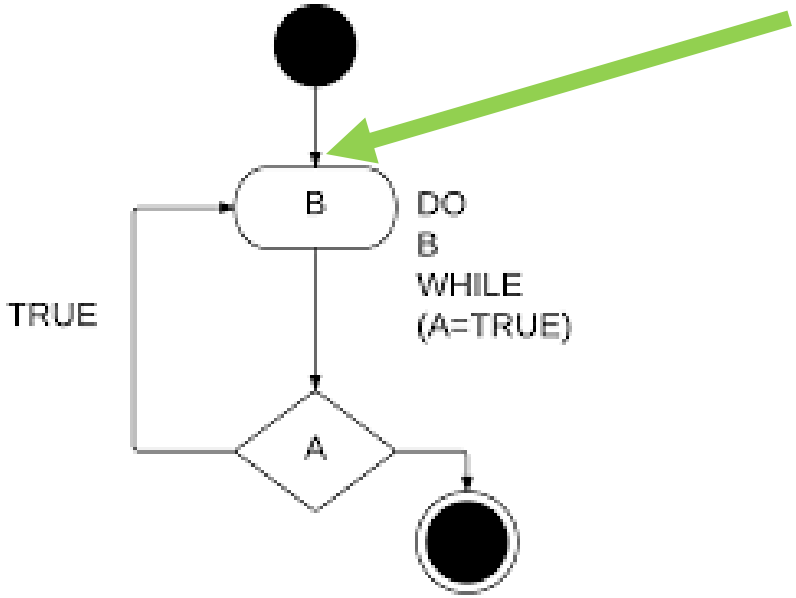
Ejemplos

Un ejemplo que nos diga si cada número que tecleemos es positivo o negativo, y que termine cuando tecleemos el número 0

```
using System;
public class Ejemplo {
            public static void Main() {

            int numero;
            Console.Write("Teclea un número (0 para salir): ");
            numero = Convert.ToInt32(Console.ReadLine());
            while (numero != 0) {
                        if (numero > 0)
                                    Console.WriteLine("Es positivo");
                        else
                                    Console.WriteLine("Es negativo");

            Console.WriteLine("Teclea otro número (0 para salir): ");
            numero = Convert.ToInt32(Console.ReadLine());
            }
      }
}
```

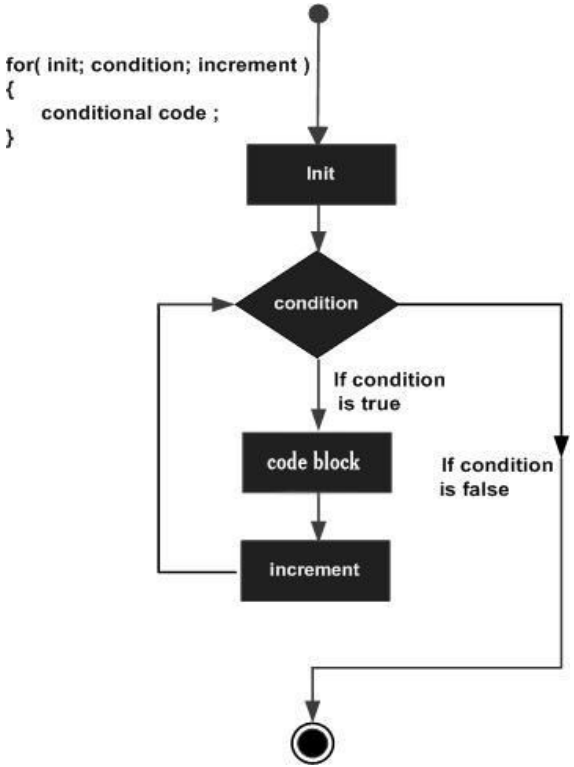
Estructuras de repetición

Estructuras de repetición Do While

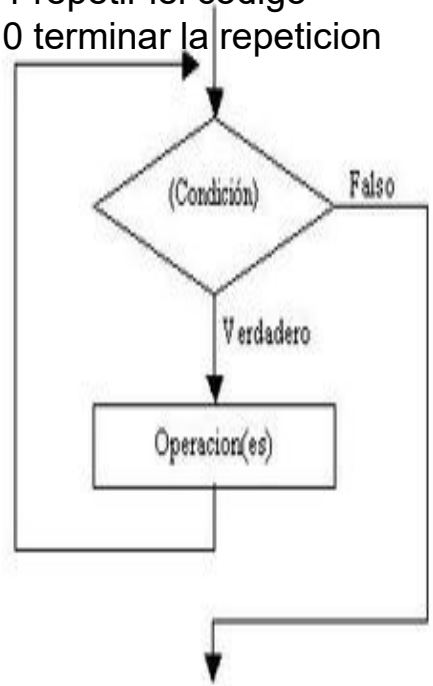


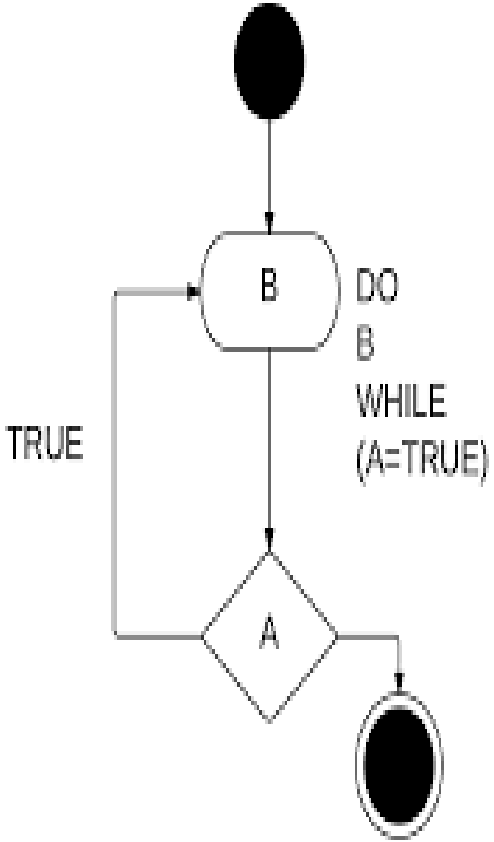

```
do

{

//Codigo a ejecutar

}while(i == 10)
```

Variables
Contador



Variable
Banderas
Int opción
1 repetir lel codigo
0 terminar la repeticion





Ejemplos

Programa ejemplo, que imprime los primeros números comprendidos del 1 al 10:

```
/// <summary>
///  //Do While loop checks the condition at the end of the loop.
//Thats why Do while loop guarantees to execute at least once.
/// </summary>

1 reference
public static void DoWhileloop()
{
    int[] array = new int[3];
    array[0] = 4;
    array[1] = 5;
    array[2] = 6;
    //Print above numbers from array using Do While loop.
    int whileInt = 0;
    do
    {
        Console.WriteLine(array[whileInt]);
        whileInt++;

    } while (whileInt < array.Length);
}
```

PRESENTACION

Presentación

La pregunta sobre la que reflexionarás en esta unidad es:

¿¿De qué forma los arreglos permiten el almacenamiento masivo de datos?

Cuando las variables atómicas son insuficientes para almacenar datos, es necesario utilizar otro tipo de variables que permitan resguardar más de un valor bajo el mismo identificador. En esta unidad aprenderás la definición y manipulación de arreglos, así como las diferentes operaciones que pueden realizarse sobre ellos.

Objetivo

- Introducirse en el manejo de las estructuras conociendo mecanismos de recorrido.
- Identificar la sintaxis para declarar arreglos.

Resultados de aprendizaje

- Identificar la sintaxis para declarar arreglos.
- Elaborar programas que requieran el manejo de grandes cantidades de datos.
- Distinguir el uso de las variables atómicas y de los arreglos.

Competencias

- Generar soluciones mediante la utilización de arreglos de una y dos dimensiones, para aplicar el almacenamiento de múltiples valores de un mismo tipo de dato.

DESARROLLO

Introducción

Los algoritmos operan sobre datos de distinta naturaleza, por lo tanto los programas que implementan dichos algoritmos necesitan una forma de representarlos.

Tipo de dato es una clase de objeto ligado a un conjunto de operaciones para crearlos y manipularlos, un tipo de dato se caracteriza por

1.Un rango de valores posibles.

2.Un conjunto de operaciones realizadas sobre ese tipo.

3.Su representación interna.

Introducción

Al definir un identificador de un determinado tipo el nombre del identificador indica la localización en memoria, el tipo los valores y operaciones permitidas, y como cada tipo se representa de forma distinta en la computadora los lenguajes de alto nivel hacen abstracción de la representación interna e ignoran los detalles pero interpretan la representación según el tipo

Ordinales: Enteros, Lógico, Booleano y Carácter.

No ordinales: Reales y Cadenas.

Arreglo (Array)

Colección ordenada e indexada de elementos con las siguientes características:

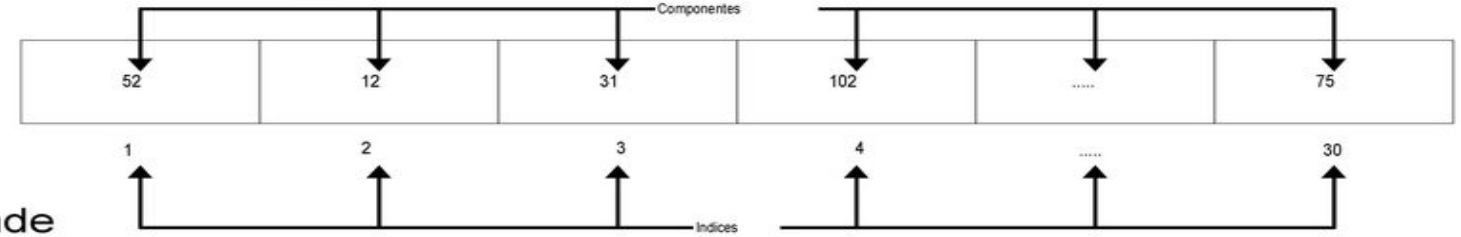
- Todos los elementos son del mismo tipo, un arreglo es una estructura homogénea.
- Los elementos pueden recuperarse en cualquier orden, simplemente indicando la posición que ocupa dentro de la estructura, esto indica que el arreglo es una estructura indexada.
- El operador de acceso es el operador [ ] .

  La memoria ocupada a lo largo de la ejecución del programa es fija, por esto es una estructura estática.
- El nombre del arreglo se socia a un área de memoria fija y consecutiva del tamaño especificado en la declaración.
- El índice debe ser de tipo ordinal. El valor del índice puede verse como el desplazamiento respecto de la posición inicial del arreglo.

Arreglo (Array)

Ejemplo
Sea **V** un arreglo de 30 elementos enteros con índices enteros.
V = (52,12,31,102,....75)
V (50)= v(1), v(2), v(3),...., v(30),

Su representación grafica serà



Donde

NTC = (30 – 1 + 1) = 30

Cada componente del arreglo **V** será un número entero, y podrá accederse por medio de un índice que será un valor comprendido entre 1 y 30.

De esta manera, por ejemplo:
V(1) hace referencia al elemento de la posición 1 cuyo valor es 52
V(2) hace referencia al elemento de la posición 2 cuyo valor es 12

Ejemplos

Programa ejemplo, que imprime los primeros números comprendidos del 1 al 10:

```
using System;
using System.Linq;
using System.Text;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace VarianteCicloForeachRecorridoArrays
{
    public class Program
    {
        public static void Main(string[] args)
        {
            Random rnd = new Random();
            int [] valores = new int[10];

            for( int i = 0; i < 10; i++ )
                valores[i] = rnd.Next(1, 100);

            foreach (int recorrido in valores)
                Console.Write( $"{recorrido}, " );


            Console.ReadKey();
        }
    }
}
```

ACTIVIDADES DE REFORZAMIENTO

Actividades de reforzamiento

Ejercicio 1. El siguiente ejemplo genera todos los elementos en la matriz de automóviles:

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

for (int i = 0; i < cars.Length; i++) {

Console.WriteLine(cars[i]);

}
```

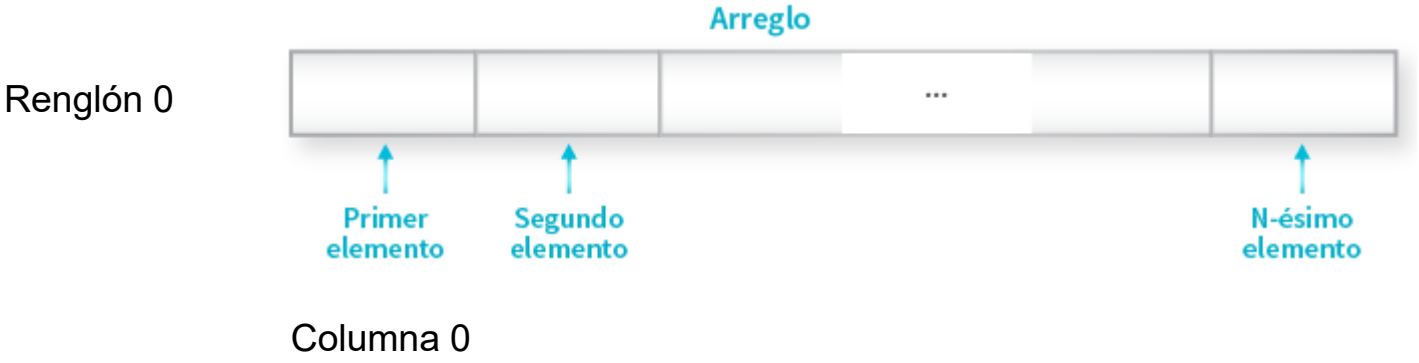
Actividades de reforzamiento

Ejercicio 1. El siguiente ejemplo genera todos los elementos en la matriz de autos, usando un bucle **foreach**:

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

foreach (string i in cars) {

Console.WriteLine(i);

}
```

Conclusión

Un arreglo puede representarse gráficamente como se muestra en la figura:



Conclusión

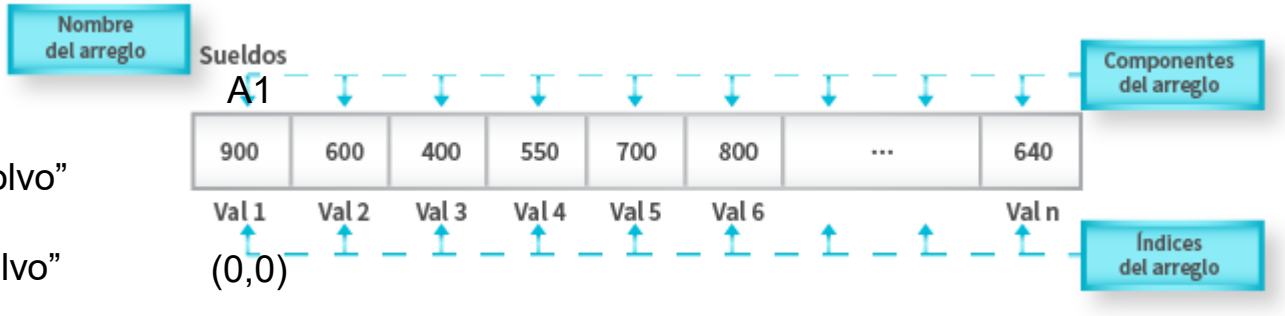
- **Los componentes:** son los valores que se almacenan en cada una de sus casillas.

- **Los índices*:*** la posición dentro del arreglo.

Para hacer referencia a un componente de un arreglo se necesita:

- El nombre del arreglo o nombre del vector
- El índice del elemento

Variable i = 0
Excel A1
Valores A1 =“Volvo”
Programación (
Cars (0,0) = “Volvo”



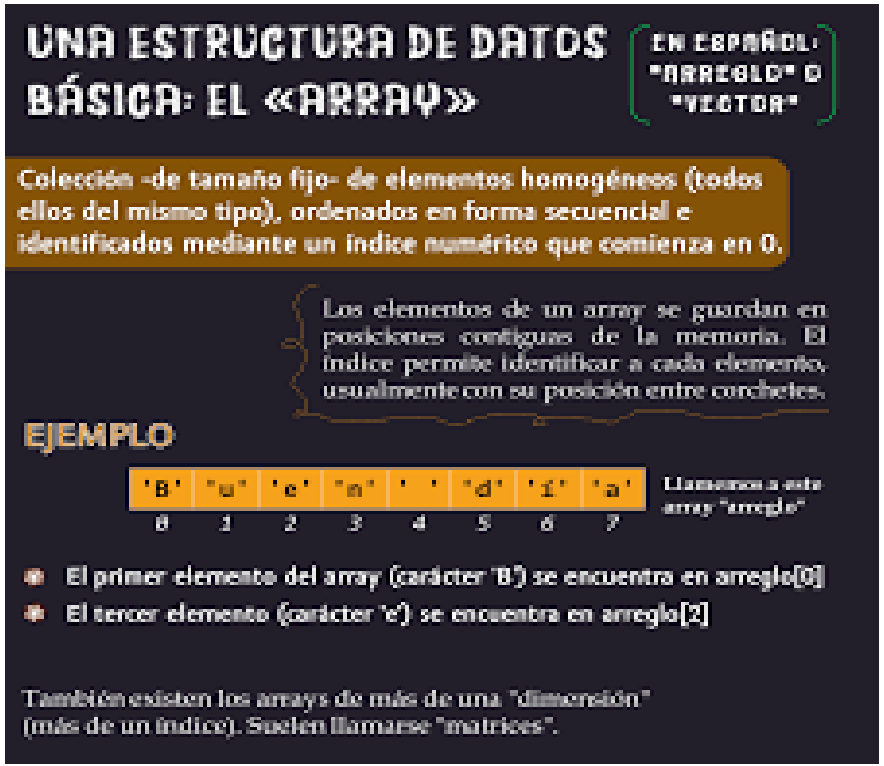
PREGUNTA DE INVESTIGACION

Pregunta de investigación

¿Los arreglos permiten el almacenamiento de diferentes tipos de datos?

CIERRE

Cierre


UNA ESTRUCTURA DE DATOS BÁSICA: EL «ARRAY»
EN ESPAÑOL: "ARREGLO" O "VECTOR"
Colección -de tamaño fijo- de elementos homogéneos (todos ellos del mismo tipo), ordenados en forma secuencial e identificados mediante un índice numérico que comienza en 0.
Los elementos de un array se guardan en posiciones contiguas de la memoria. El índice permite identificar a cada elemento, usualmente con su posición entre corchetes.
EJEMPLO
'B' 'u' 'e' 'n' ' ' 'd' 'í' 'a'
0 1 2 3 4 5 6 7
Llamamos a este array "arreglo"
El primer elemento del array (carácter 'B') se encuentra en arreglo[0]
El tercer elemento (carácter 'e') se encuentra en arreglo[2]
También existen los arrays de más de una "dimensión" (más de un índice). Suelen llamarse "matrices".

BIBLIOGRAFÍA COMPLEMENTARIA

http://www.msdn.microsoft.com/net/ecma.

“A programmer’s introduction to C#” escrito por Eric Gunnerson y publicado por Apress en 2000.

C# and the .NET Framework”, escrito por Andrew Troelsen y publicado por Apress en 2001

“C# Essentials”, escrito por Beb Albahari, Peter Drayton y Brand Merril y publicado por O’Reilly en 2000.

“C# Programming with the Public Beta”, escrito por Burton Harvey, Simon Robinson, Julian Templeman y Karli Watson y publicado por Wrox Press en 2000.

“Inside C#”, escrito por Tom Archer y publicado por Microsoft en 2000 • “Presenting C#”, escrito por Christoph Wille y publicado por Sams Publishing en 2000.

ÍNDICE

DUDAS