

Preguntas detonadoras



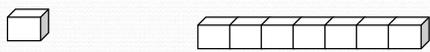
- ❑ ¿Qué es un arreglo? ¿Para qué sirve?
- ❑ ¿Cuáles son las características que distinguen a los arreglos?
- ❑ Un arreglo, ¿puede almacenar objetos?
- ❑ Un arreglo, ¿puede ser un atributo de una clase?
- ❑ ¿Cómo se acceden a los datos de un arreglo?
- ❑ ¿Se pueden acceder a los datos de un arreglo mediante una propiedad?
- ❑ ¿Cómo se recorren las celdas de un arreglo?
- ❑ ¿Cuáles son las operaciones básicas con arreglos?

3

Tipos de datos

TIPOS DE DATOS

- Simples – Almacenan un valor
- Compuestos – Almacenan un conjunto de valores



Simple Compuesto

Fig. 3.1. Tipos de datos simples y compuestos.

4

Arreglos

Conjunto finito,
homogéneo
y estático
de datos relacionados
e indexados

Homogéneo significa que todas sus celdas son del mismo tipo de dato

Estático se refiere a que, una vez declarado, no cambia su tamaño

A	
0	43
1	23
2	12
3	68
4	97

Índices

Celdas

5

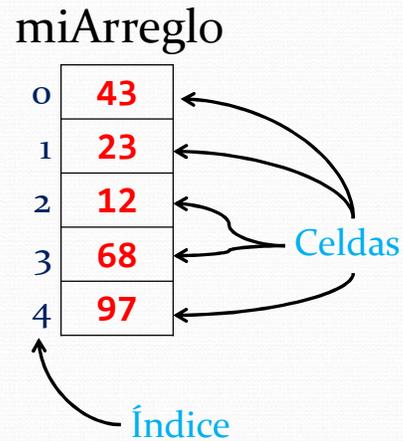
Dimensiones de arreglos

- **Unidimensionales (Vectores)**
 - 1 índice
- **Bidimensionales (Matrices)**
 - 2 índices (renglón y columna)
- **Tridimensionales (Cubos)**
 - 3 índices (renglón, columna y plano)
- **Multidimensionales**

6

Arreglos unidimensionales

- Solamente tienen **1** índice
- También se conocen con el nombre de "vectores"



7

Operaciones básicas con arreglos

Operaciones básicas con arreglos

- Declaración y creación
- Inicialización
- Inserción
- Eliminación
- Búsqueda
- Ordenamiento
- Recorrido

Recuerde que **NO** se pueden agregar o eliminar celdas del arreglo; solamente su contenido

8

Declaración de arreglos

- No sólo basta con declararlo, sino también debe crearse con el operador **new**

```
int [ ] miArreglo; // declara el arreglo  
miArreglo = new int[12]; // crea el arreglo  
  
double [ ] otroArreglo = new double[10];
```

9

Inicialización de arreglos

```
int [ ] miArreglo = { 1, 2, 3, 4, 5, 6 };  
string [ ] nombres = { "Emilio", "Pedro" };  
double [ ] otroArreglo = { 3.4, -4.5, 7.0 };
```

10

Manejo del tamaño del arreglo

- Declarar una constante con el tamaño del arreglo

```
const int intTamano = 15;

// declara el arreglo
int [ ] miArreglo;

// reserva memoria
miArreglo = new int[intTamano];
```

11

Creación de arreglos

- Se pueden crear arreglos cuyo tamaño se pueda establecer a partir de una expresión que produzca un valor entero

```
// variable
int intTamano = 15;

// declara el arreglo
int [ ] miArreglo;

// reserva memoria
miArreglo = new int[intTamano];
```

12

Capturar el tamaño del arreglo

- Se pueden crear arreglos capturando el tamaño del arreglo

```
// variable
int intTamano;

// captura el tamaño del arreglo
Console.Write("Teclee el tamaño del arreglo: ");
intTamano=int.Parse(Console.ReadLine());

// declara y crea el arreglo
int [ ] miArreglo = new int[intTamano];
```

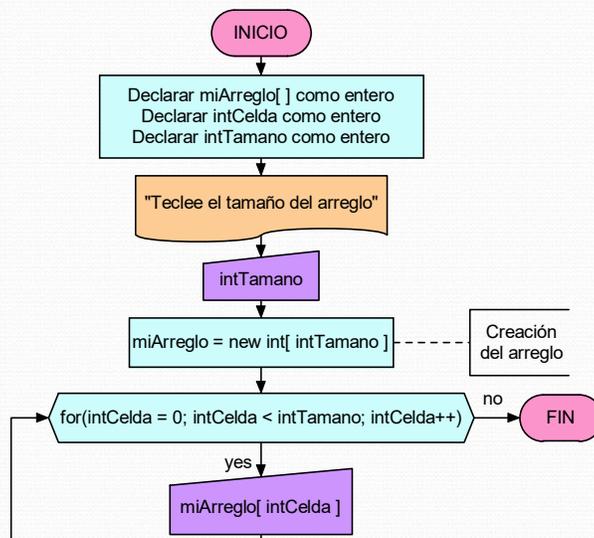
Capturar los datos de las celdas del arreglo

- Suponga que declara un arreglo que almacena datos numéricos enteros
- El usuario captura la cantidad de celdas
- Se crea el arreglo de acuerdo a la cantidad de celdas
- Se implementa un ciclo para capturar el dato de cada una de las celdas

miArreglo

0	?
1	?
2	?
3	?
4	?

Diagrama de flujo



15

Recorrido de arreglos

- ❖ La longitud de un arreglo se obtiene con la propiedad **Length**

```

int [ ] miArreglo = { 1, 2, 3, 4, 5, 6 };

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

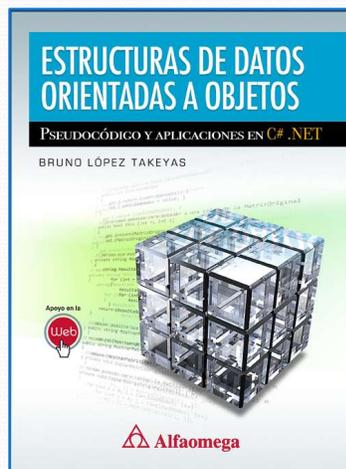
16

Recorrido de un arreglo con un ciclo *foreach*

```
string[] miArreglo = {"Pepe", "Paola"};  
  
foreach(string strNombre in miArreglo)  
{  
    Console.WriteLine(strNombre);  
}
```

17

LECTURA



Para consultar este tema con mayor profundidad se recomienda la lectura de:

Capítulo 3.- Arreglos

18

Ordenamiento de los datos de un arreglo

- Los algoritmos que ordenan datos de los arreglos se conocen como “**sorteadores**” o “**métodos de ordenamiento**”
- **Existen muchos métodos de ordenamiento**
- Cada método de ordenamiento tiene una estrategia para reacomodar sus datos
- **No se “mueven” las celdas del arreglo, sino su contenido**

19

Método de la burbuja

- Es el método de ordenamiento de datos más sencillo de implementar
- **Su nombre se debe a que durante el ordenamiento de los datos el elemento a ordenar “burbujea” hacia arriba en el arreglo**
- Existen varias versiones:
 - *Burbuja izquierda*
 - *Burbuja derecha*
 - *Burbuja con señal*

Principio del funcionamiento de la burbuja izquierda

- Recorre el arreglo de derecha a izquierda
- Compara los datos adyacentes en el arreglo
- Hace los intercambios pertinentes
- Implementa un ciclo que controla la posición que ocupará el dato del arreglo (i)
- Dentro de este ciclo hay otro ciclo que recorre el arreglo de derecha a izquierda para hacer las comparaciones (j)
- Los desplazamientos de los datos se hacen hacia la izquierda del arreglo

Intercambio de datos

- *Algunos métodos de ordenamiento requieren intercambiar datos*
- *Para lograrlo se apoyan en una variable auxiliar (temporal)*

Arreglo		
0	6	<--- i
1	4	
2	7	
3	2	
4	3	
5	5	
6	1	<--- j

temporal

22

Ejemplo de intercambio de datos

- 1) Copiar el valor del primer dato en la variable temporal.
- 2) Copiar el valor del segundo dato en la variable del primer dato.
- 3) Copiar el valor de la variable temporal en la variable del segundo dato.

Arreglo			
0	6	<--- primer dato	
1	4		
2	7		
3	2		temporal
4	3		
5	5		
6	1	<--- segundo dato	

23

Intercambio de datos

```

Intercambiar(Arreglo[], p, i): nulo
1.-Temporal = Arreglo[p]
2.-Arreglo[p] = Arreglo[i]
3.-Arreglo[i] = Temporal
{Fin del método}

```

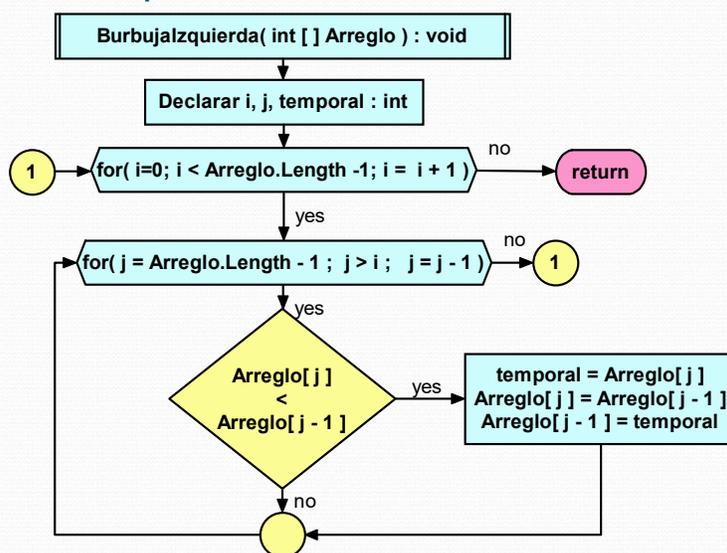
24

Pseudocódigo de la burbuja izquierda

BurbujaIzquierda(Arreglo[]): nulo

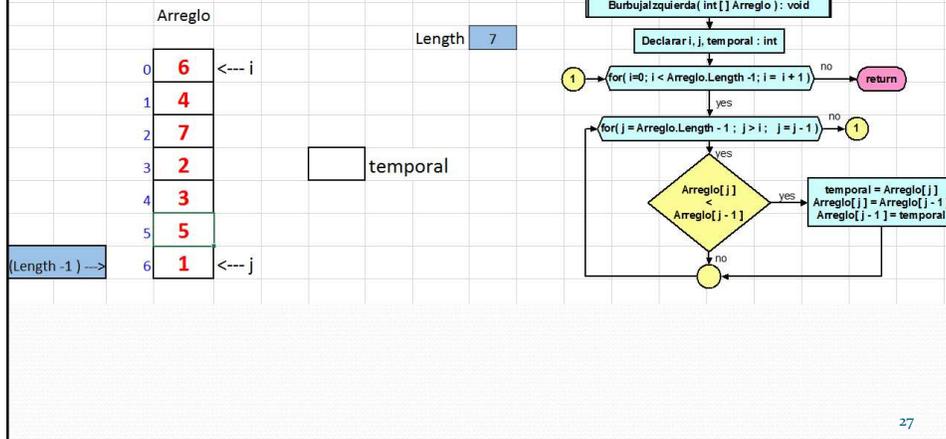
- 1.- REPETIR CON i DESDE 0 HASTA $\text{Arreglo.Length}-1$ CON PASO 1
 - 1.1. REPETIR CON j DESDE $\text{Arreglo.Length}-1$ HASTA i CON PASO -1
 - 1.1.1 SI $\text{Arreglo}[j] < \text{Arreglo}[j-1]$ ENTONCES
 - 1.1.1.1 Intercambia(Arreglo , j , $j-1$)
 - 1.1.2. {FIN DE LA CONDICIONAL DEL PASO 1.1.1}
 - 1.2. {FIN DEL CICLO DEL PASO 1.1}
- 2.- {FIN DEL CICLO DEL PASO 1}
- 3.- RETURN

Diagrama de flujo del método de la burbuja izquierda que ordena números enteros

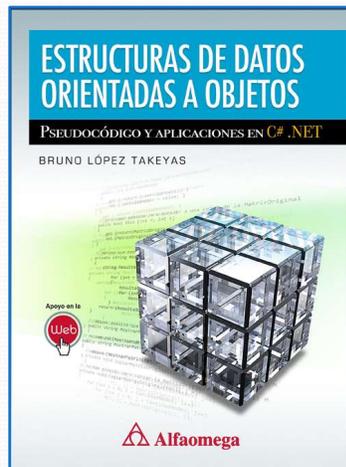


26

Funcionamiento del método de la burbuja



LECTURA



Para consultar este tema con mayor profundidad y revisar otros métodos de ordenamiento se recomienda la lectura de:

Capítulo 10.- Métodos de ordenamiento

LECTURA

Para generar datos aleatorios, se recomienda la lectura de los apuntes



¿Cómo generar datos aleatorios en C#?

Incluye generar:

- Nombres aleatorios
- Sexo
- Fecha
- Datos booleanos
- etc.

<https://nlaredo.tecnm.mx/takeyas/Apuntes/Fundamentos%20de%20Programacion/Apuntes/07.-Aleatorios.pdf>

29

¿Cómo llenar un arreglo con números enteros aleatorios?

```
// Declaración del arreglo
int [] miArreglo;

// Declaración del número aleatorio
Random aleatorio = new Random();

Console.WriteLine("\nTeclee la cantidad de celdas del arreglo: ");
int intTamañoArreglo = int.Parse(Console.ReadLine());

// Creación del arreglo
miArreglo = new int[intTamañoArreglo];

// Llena el arreglo con números aleatorios
for (int intCelda = 0; intCelda < intTamañoArreglo; intCelda++)
    miArreglo[intCelda] = aleatorio.Next();
```

30

¿Cómo calcular el tiempo de ejecución del algoritmo?

```
Console.WriteLine("Inicia cronómetro ...");
DateTime dtmHoraInicio = DateTime.Now;
.
.
.
.

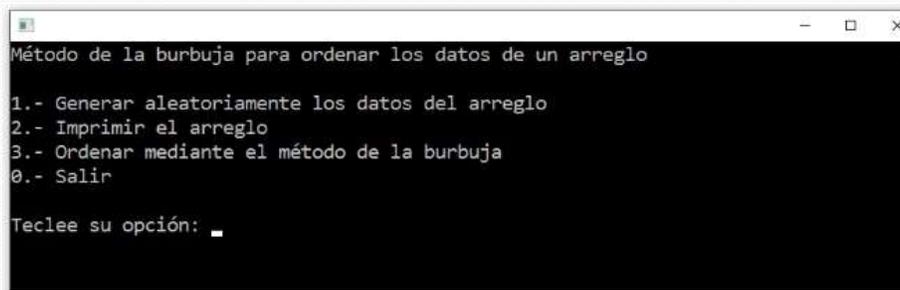
DateTime dtmHoraFin = DateTime.Now;
Console.WriteLine("Finaliza cronómetro ..");

TimeSpan tiempo = dtmHoraFin - dtmHoraInicio;
Console.WriteLine("Tiempo:"+tiempo.TotalMilliseconds+" ms.");
```

31

Ejercicio

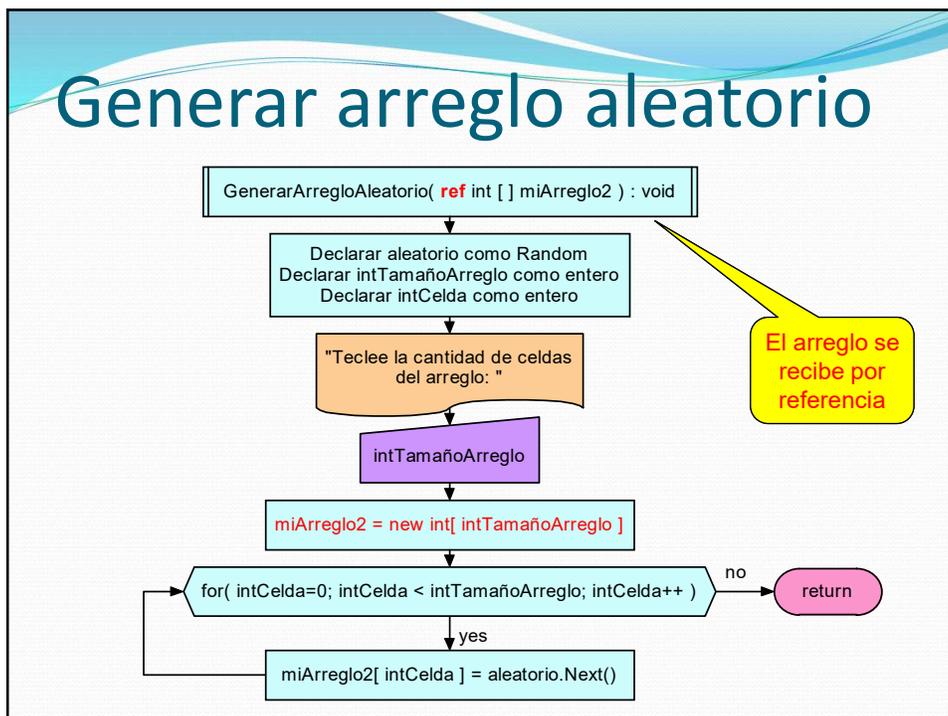
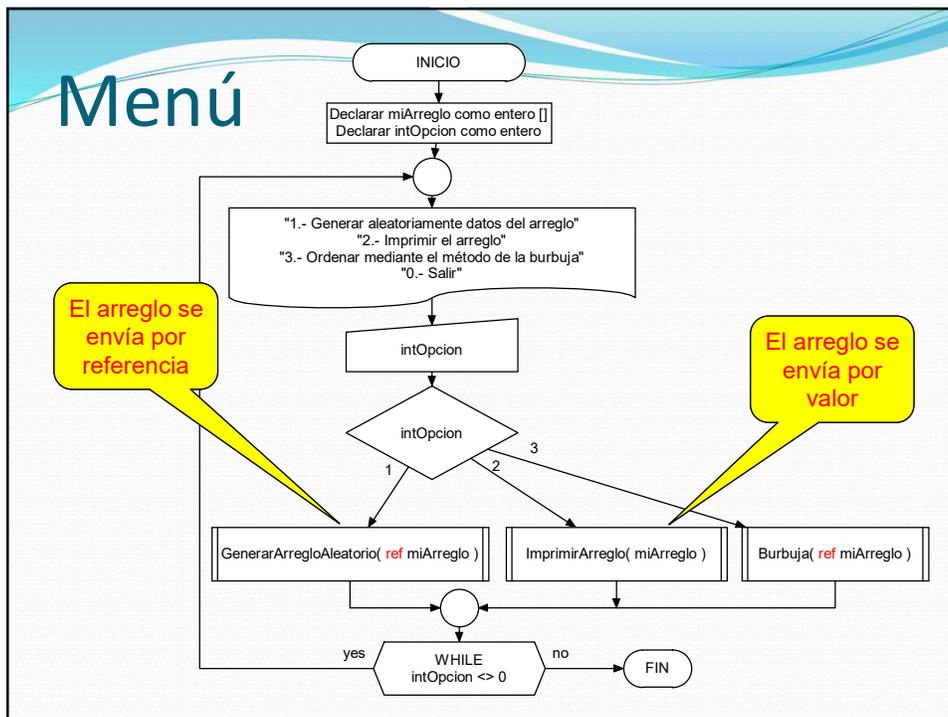
- Llenar un arreglo con números aleatorios
- Enviar el arreglo por referencia a un método para ordenar sus datos
- Enviar el arreglo por valor a un método para imprimir sus datos



```
Método de la burbuja para ordenar los datos de un arreglo

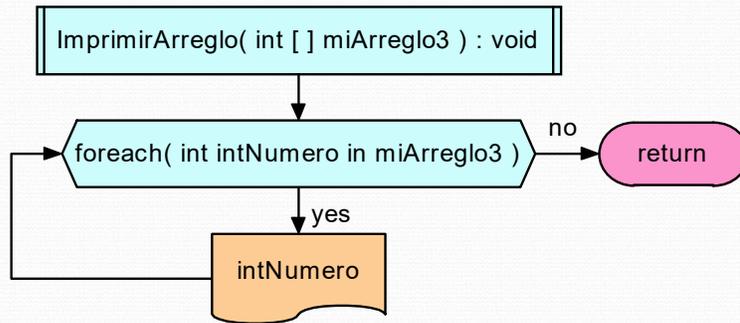
1.- Generar aleatoriamente los datos del arreglo
2.- Imprimir el arreglo
3.- Ordenar mediante el método de la burbuja
0.- Salir

Teclee su opción: █
```



Imprimir el arreglo

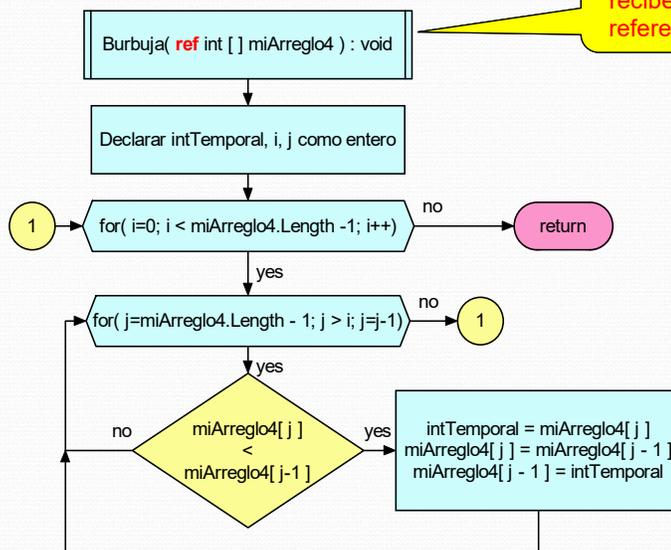
El arreglo se recibe por valor



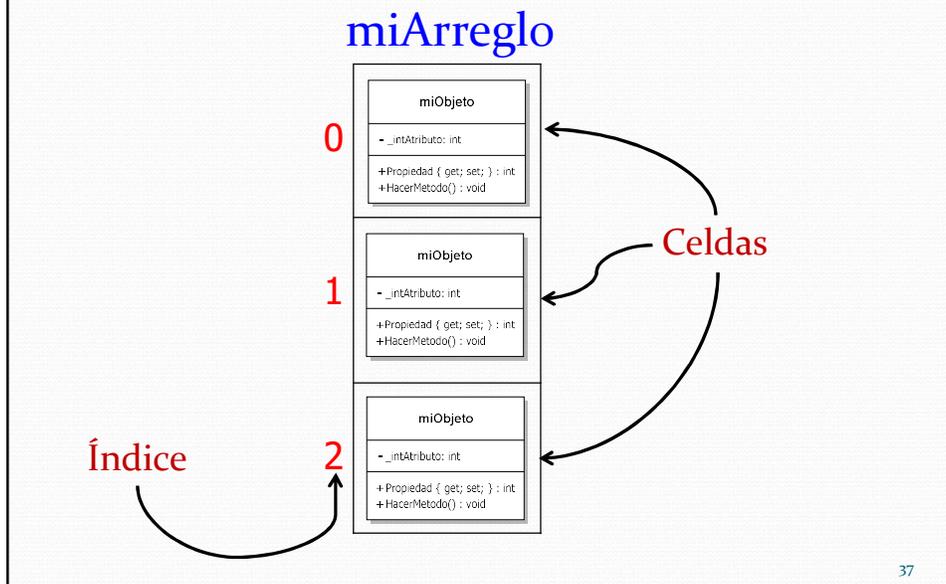
Este método recibe el arreglo por valor porque NO cambia sus datos, solamente los consulta para imprimirlos

Ordenar el arreglo

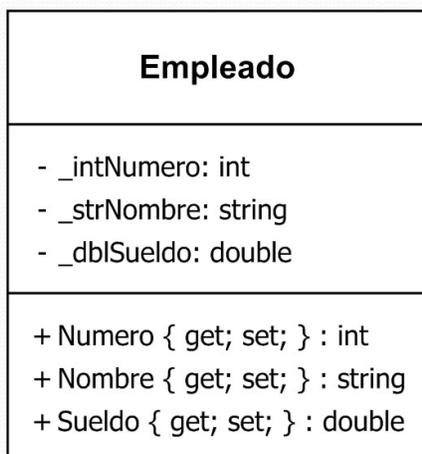
El arreglo se recibe por referencia



Arreglo de objetos



Declaración y creación de un arreglo de objetos



- Suponga que desea un arreglo que almacene los empleados de una empresa
- Datos
 - **Número**
 - **Nombre**
 - **Sueldo**

38

Declaración y creación de un arreglo de objetos (*cont.*)

```
// Declaración del arreglo
Empleado [] miArreglo;

// Creación del arreglo
miArreglo = new Empleado[intTamaño];
```

39

Menú de opciones

```
ARREGLO DE OBJETOS DE EMPLEADOS

1.- Capturar el arreglo y sus objetos
2.- Imprimir los datos del arreglo

Seleccione su opción:
```

40

Codificación del menú de opciones

```

static void Main(string[] args)
{
    // Declaraciones de variables locales
    Empleado[] miArreglo=null;
    int intOpcion = 0;

    do {
        Console.Clear();
        Console.WriteLine("\nARREGLO DE OBJETOS DE EMPLEADOS");
        Console.WriteLine("\n\n1.- Capturar el arreglo y sus objetos");
        Console.WriteLine("2.- Imprimir los datos del arreglo");
        Console.Write("\n\nSeleccione su opción: ");
        intOpcion = int.Parse(Console.ReadLine());
        switch(intOpcion) {
            case 1: CapturarArreglo(ref miArreglo); break; // Por referencia
            case 2: ImprimirArreglo(miArreglo); break; // Por valor
        }
    } while (intOpcion!=0);
}

```

Capturar el arreglo: Opciones...

a) Crear el objeto, capturar sus datos y luego almacenarlo en la celda del arreglo

- Crear el arreglo especificando el tamaño
- Implementar un ciclo
 - Dentro del ciclo, crear un objeto de tipo Empleado
 - Capturar los datos del objeto del empleado
 - Almacenar el objeto en alguna celda del arreglo
- Iterar el ciclo

b) Capturar los datos del objeto directamente en la celda

- Crear el arreglo especificando el tamaño
- Implementar un ciclo
 - Dentro del ciclo, crear un objeto de tipo Empleado directamente en la celda del arreglo
 - Capturar los datos de un empleado directamente en la celda del arreglo
- Iterar el ciclo

42

Captura del arreglo (opción a)

```

static void CapturarArreglo(ref Empleado [] nuevoArreglo)
{
    // Declaraciones locales
    int intTamaño = 0;
    Empleado nuevoEmpleado;
    Console.Clear();
    Console.WriteLine("CAPTURAR EL ARREGLO Y SUS OBJETOS\n\n");
    do {
        Console.Write("\nTeclee el tamaño del arreglo: ");
        intTamaño = int.Parse(Console.ReadLine());
    } while (intTamaño < 2);

    // Crear el arreglo según el tamaño capturado por el usuario
    nuevoArreglo = new Empleado[intTamaño];

    for(int intCelda=0; intCelda<intTamaño; intCelda++)
    {
        Console.WriteLine("\n\nCAPTURE LOS DATOS DEL EMPLEADO "+intCelda.ToString());
        Console.WriteLine();

        // Creación de un nuevo objeto para el Empleado
        nuevoEmpleado = new Empleado();

        Console.Write("Número? ");
        nuevoEmpleado.Numero = int.Parse(Console.ReadLine());

        Console.Write("Nombre? ");
        nuevoEmpleado.Nombre = Console.ReadLine();

        Console.Write("Sueldo? ");
        nuevoEmpleado.Sueldo = double.Parse(Console.ReadLine());

        // Almacenar el nuevoEmpleado en la intCelda del nuevoArreglo
        nuevoArreglo[intCelda] = nuevoEmpleado;
    }
}

```



43

Captura del arreglo (opción b)

```

static void CapturarArreglo2(ref Empleado[] nuevoArreglo)
{
    // Declaraciones locales
    int intTamaño = 0;

    Console.Clear();
    Console.WriteLine("CAPTURAR EL ARREGLO Y SUS OBJETOS\n\n");
    do {
        Console.Write("\nTeclee el tamaño del arreglo: ");
        intTamaño = int.Parse(Console.ReadLine());
    } while (intTamaño < 2);

    // Crear el arreglo según el tamaño capturado por el usuario
    nuevoArreglo = new Empleado[intTamaño];

    for (int intCelda = 0; intCelda < intTamaño; intCelda++) {
        Console.WriteLine("\n\nCAPTURE LOS DATOS DEL EMPLEADO " + intCelda.ToString());
        Console.WriteLine();

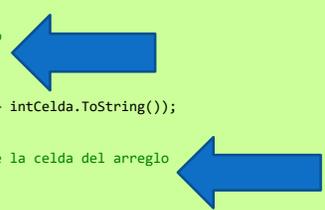
        // Creación de un nuevo objeto para el Empleado dentro de la celda del arreglo
        nuevoArreglo[intCelda] = new Empleado();

        Console.Write("Número? ");
        nuevoArreglo[intCelda].Numero = int.Parse(Console.ReadLine());

        Console.Write("Nombre? ");
        nuevoArreglo[intCelda].Nombre = Console.ReadLine();

        Console.Write("Sueldo? ");
        nuevoArreglo[intCelda].Sueldo = double.Parse(Console.ReadLine());
    }
}

```



44

Imprimir el arreglo: Opciones ...

- Recorrer automáticamente las celdas del arreglo mediante el ciclo *foreach*
- Recorrer las celdas del arreglo de forma "tradicional" mediante el ciclo *for*

```
DATOS DEL ARREGLO DE OBJETOS DE EMPLEADOS
Número      Nombre                               Sueldo
-----
5           Bruno López Takeyas                 $55.55
4           Juan Carlos Pereyra Estrada         $44.56

Oprima cualquier tecla para continuar ...
```

45

Imprimir el arreglo (opción a)

```
static void ImprimirArreglo(Empleado [] unArreglo)
{
    Console.Clear();
    Console.WriteLine("DATOS DEL ARREGLO DE OBJETOS DE EMPLEADOS\n\n");
    Console.WriteLine("Número      Nombre                               Sueldo");
    Console.WriteLine("-----");

    foreach(Empleado unEmpleado in unArreglo)
        Console.WriteLine(unEmpleado.Numero.ToString().PadRight(6, ' ') +
            "      "+unEmpleado.Nombre.PadRight(30, ' ')+
            "      "+unEmpleado.Sueldo.ToString("C"));

    Console.WriteLine("\n\nOprima cualquier tecla para continuar ...");
    Console.ReadKey();
}
```

46

Imprimir el arreglo (opción b)

```
static void ImprimirArreglo2(Empleado[] unArreglo)
{
    Console.Clear();
    Console.WriteLine("DATOS DEL ARREGLO DE OBJETOS DE EMPLEADOS\n\n");
    Console.WriteLine("Número    Nombre                               Sueldo");
    Console.WriteLine("-----");
}

for (int i = 0; i < unArreglo.Length; i++)
    Console.WriteLine(unArreglo[i].Numero.ToString().PadRight(6, ' ') +
        " " + unArreglo[i].Nombre.PadRight(30, ' ') +
        " " + unArreglo[i].Sueldo.ToString("C"));

Console.WriteLine("\n\nOprima cualquier tecla para continuar ...");
Console.ReadKey();
}
```

47

Ejemplo de uso de la referencia *this*

Persona
- nombre: string - edad: int
+ ModificarDatos(nombre: string, edad: int) : void



El uso de la palabra ***this*** para referirse a los miembros internos de una clase es opcional, pero es necesaria cuando un parámetro y un atributo comparten el mismo nombre

40

Codificación de la referencia *this*

```
class Persona
{
    // Declaración de los atributos privados
    private string nombre;
    private int edad;

    // Mutator
    public void ModificarDatos(string nombre, int edad)
    {
        this.nombre = nombre;
        this.edad = edad;
    }
}
```

La referencia *this*

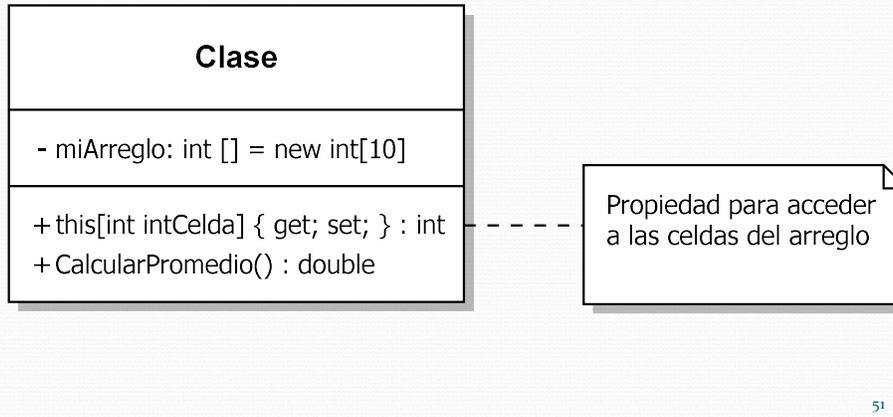
- Para hacer referencia (explícita) a un elemento que se encuentra dentro de la misma clase (ésta) se utiliza “*this*”.

```
class Articulo
{
    private double precio = 0;
    public void PonerPrecio ( double precio )
    {
        this.precio = precio;
    }
    public double ConsultarPrecio()
    {
        return this.precio;
    }
}
```

Muy útil cuando existen distintos elementos con el mismo nombre, pero con distinto significado dentro de un ámbito determinado.

50

Diseño de una clase con un arreglo como atributo privado



Codificación de la clase

```
class Clase
{
    // Atributo privado
    private int [] miArreglo = new int[10];

    // Propiedad (referencia this)
    public int this[int intCelda]
    {
        get { return miArreglo[ intCelda ]; }
        set { miArreglo[ intCelda ] = value; }
    }

    // Método
    public double CalcularPromedio()
    {
        . . .
    }
}
```

Propiedad para acceder al valor de una celda de un arreglo de una clase

```
// Declaración y creación del objeto
Clase miObjeto = new Clase();

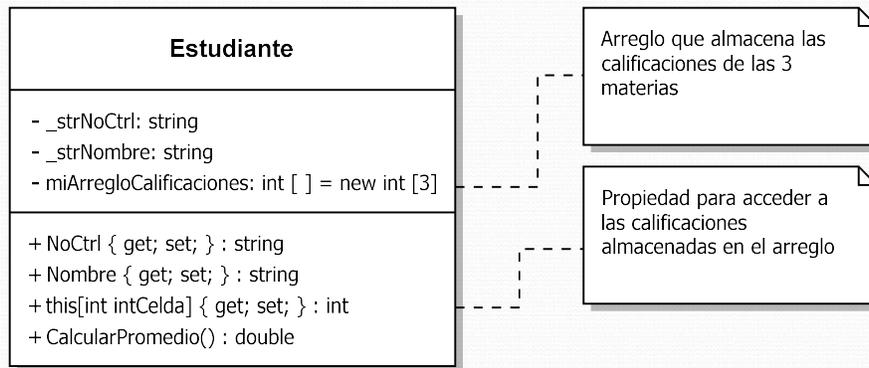
for (int intCelda = 0; intCelda < 10; intCelda++)
{
    Console.WriteLine("Teclee el dato {0}: ", intCelda + 1);
    miObjeto[intCelda] = int.Parse(Console.ReadLine());
}
```

53

Ejercicio resuelto

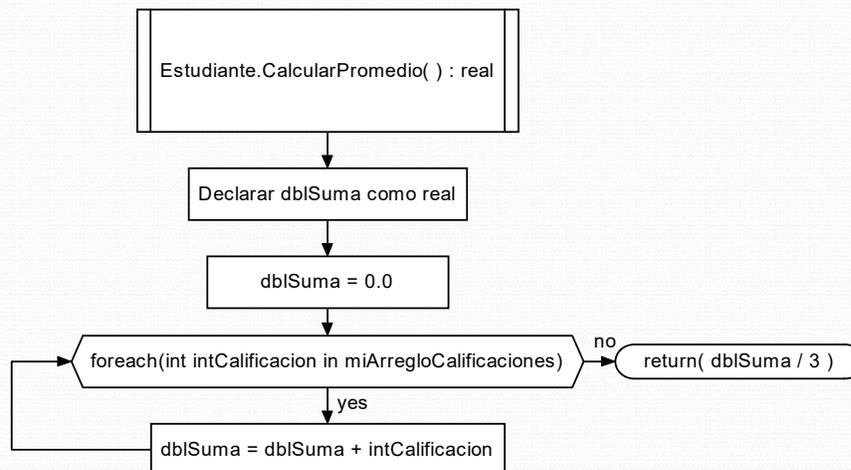
- Un profesor requiere una aplicación para administrar los datos de sus estudiantes:
 - *Número de control (string)*
 - *Nombre (string)*
 - *Calificaciones de sus 3 materias (int)*
- Para ello, requiere un arreglo donde cada celda almacene un objeto con los datos de un estudiante

Diseño de la clase Estudiante



55

Diagrama de flujo del método para calcular el promedio de un estudiante

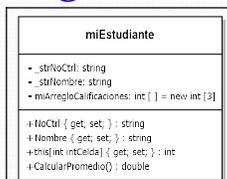


56

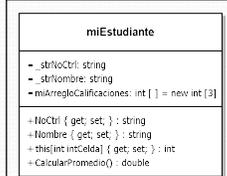
Arreglo de objetos de tipo Estudiante

miArregloEstudiantes

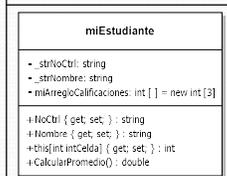
0



1



2



57

Operación de la aplicación

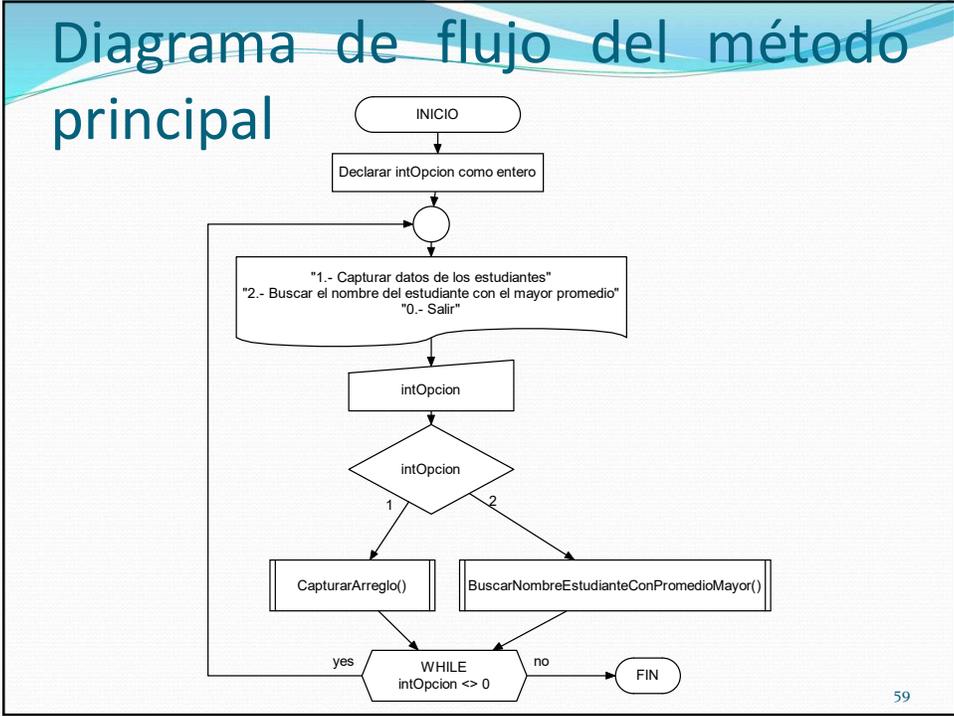
- Diseñe un menú con las opciones

MENÚ DE OPCIONES

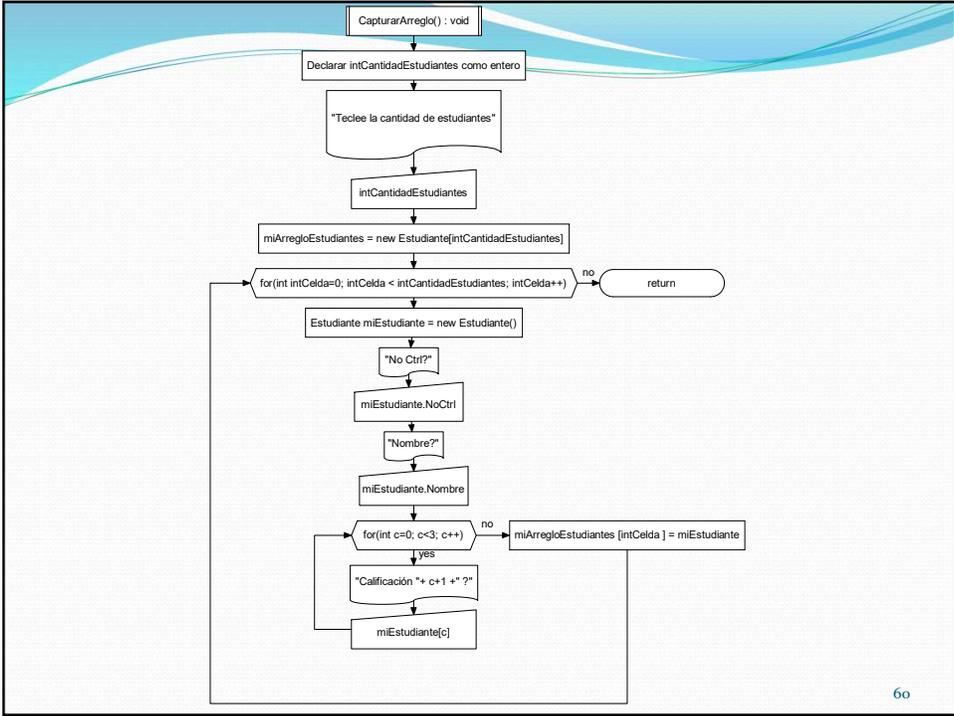
```

1.- Capturar los datos de los estudiantes
2.- Nombre del estudiante con el mayor promedio
0.- Salir
  
```

Teclee su opción:

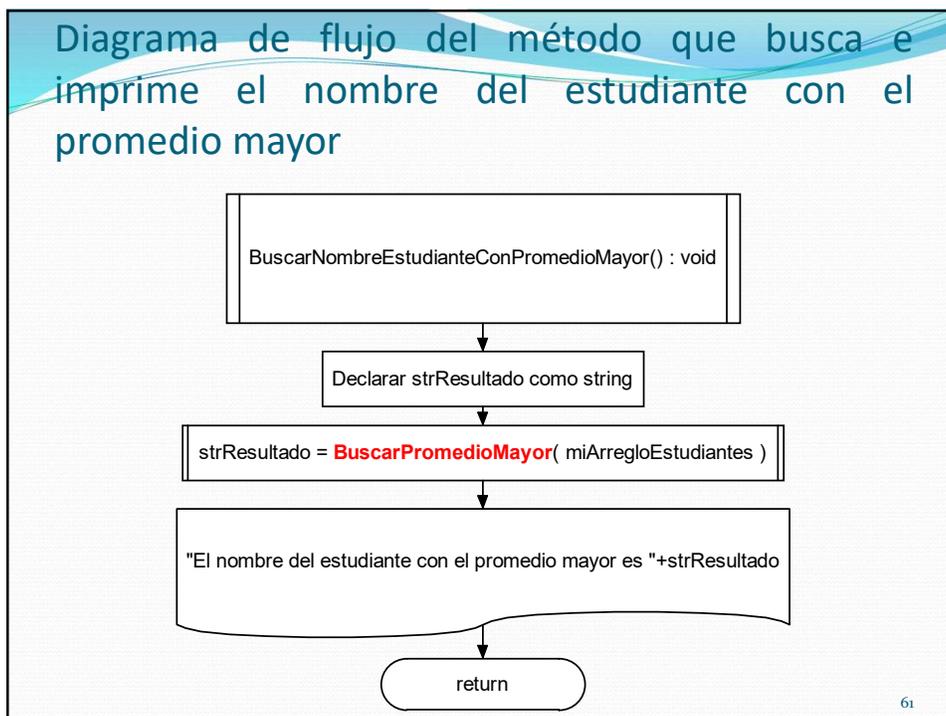


59



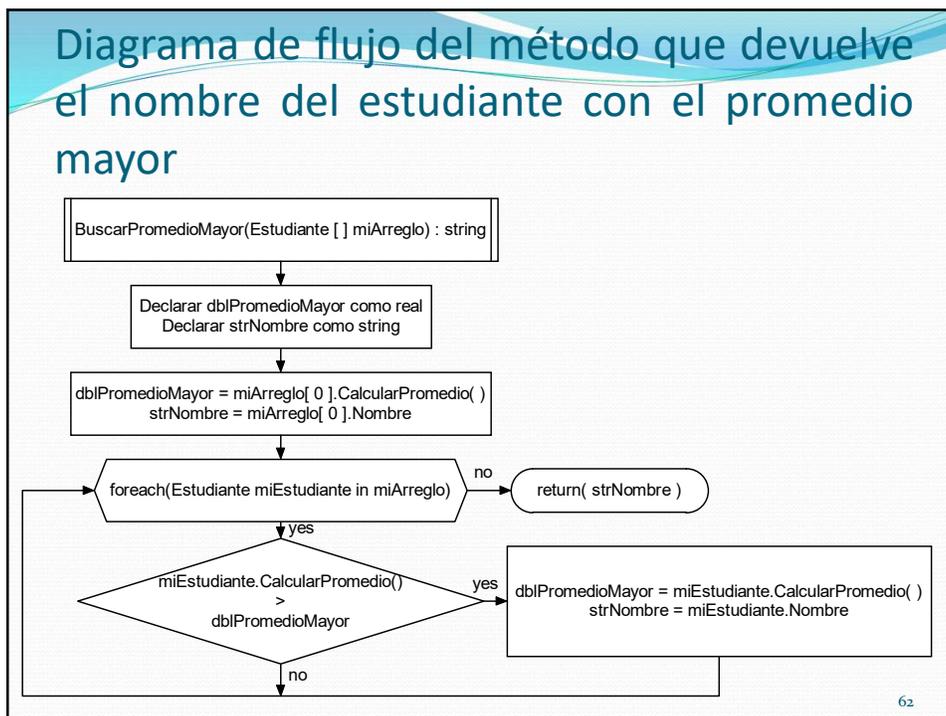
60

Diagrama de flujo del método que busca e imprime el nombre del estudiante con el promedio mayor



61

Diagrama de flujo del método que devuelve el nombre del estudiante con el promedio mayor



62

Arreglos bidimensionales

- Tienen 2 índices (renglón, columna)
- También se conocen con el nombre de "matrices"

<code>miArreglo</code>	Columna 0	Columna 1
Renglón 0	<code>miArreglo[0,0]</code>	<code>miArreglo[0,1]</code>
Renglón 1	<code>miArreglo[1,0]</code>	<code>miArreglo[1,1]</code>
Renglón 2	<code>miArreglo[2,0]</code>	<code>miArreglo[2,1]</code>

63

Declaración de matrices

- No sólo basta con declararlo, sino también debe crearse con el operador **new**
- Se declara primero la cantidad de renglones seguido de la cantidad de columnas

```
int [ , ] matriz; // declara la matriz
matriz = new int[2,3]; // reserva memoria

double [ , ] matriz2 = new double[3,4];
```

64

Inicialización de matrices

- Se declara e inicializa una matriz de 3×2

3 renglones

2 columnas

```
int [ , ] matriz = { { 1, 2 },  
                    { 3, 4 },  
                    { 5, 6 } };
```

65

Recorrido de matrices

```
int [ , ] matriz = { { 1, 2 },  
                    { 3, 4 },  
                    { 5, 6 } };
```

```
for(int r=0; r< matriz.GetLength(0); r++)  
{  
    Console.WriteLine("\n");  
    for(int c=0; c< matriz[r].GetLength(1); c++)  
        Console.WriteLine("\t" + matriz[r,c]);  
}
```

66

Recorrido de una matriz mediante el ciclo *foreach*

```
int [ , ] matriz = { { 1, 2 },
                    { 3, 4 },
                    { 5, 6 } };

foreach(int numero in matriz)
{
    Console.WriteLine("\t" + numero);
}
// Imprime 1 2 3 4 5 6
```

67

Ejemplo: Menú

```
static void Main(string[] args)
{
    // Declaración de una variable local para la matriz
    int[,] miMatriz = null;

    int intOpcion = 0;
    do {
        Console.Clear();
        Console.WriteLine("M A T R I C E S\n");
        Console.WriteLine("1.- Generar datos aleatorios de la matriz");
        Console.WriteLine("2.- Imprimir matriz");
        Console.WriteLine("0.- Salir");
        Console.Write("\nTeclee su opción: ");
        intOpcion = int.Parse(Console.ReadLine());

        switch(intOpcion) {
            // Se envía por referencia
            case 1: GenerarDatosMatriz(ref miMatriz); break;
            // Se envía por valor
            case 2: ImprimirMatriz(miMatriz); break;
        }
    } while (intOpcion!=0);
}
```

Método para llenar la matriz

```
// Método que recibe la matriz por referencia para llenarla con datos aleatorios
static void GenerarDatosMatriz(ref int [,] matriz)
{
    int intReglones = 0, intColumnas = 0;

    Console.Clear();
    Console.WriteLine("Capture los datos de la matriz");
    Console.Write("\n\n¿Cantidad de renglones? ");
    intReglones = int.Parse(Console.ReadLine());

    Console.Write("\n\n¿Cantidad de columnas? ");
    intColumnas = int.Parse(Console.ReadLine());

    // Creación de la matriz
    matriz = new int[intReglones, intColumnas];

    Random aleatorio = new Random();

    // Generar datos aleatorios para llenar la matriz
    for (int r = 0; r < intReglones; r++)
        for (int c = 0; c < intColumnas; c++)
            matriz[r, c] = aleatorio.Next(100);

    Console.WriteLine("\n\nSe ha generado la matriz con datos aleatorios");
    Console.WriteLine("\nOprima cualquier tecla para continuar...");
    Console.ReadKey();
}
```

Método para imprimir la matriz

```
// Método que recibe la matriz por valor para imprimir sus
datos
static void ImprimirMatriz(int [,] matriz)
{
    Console.Clear();
    Console.WriteLine("Datos de la matriz");

    for(int r=0; r<matriz.GetLength(0); r++)
    {
        Console.Write("\n");
        for (int c = 0; c < matriz.GetLength(1); c++)
            Console.Write(matriz[r, c].ToString() + "\t"); ;
    }
}
```

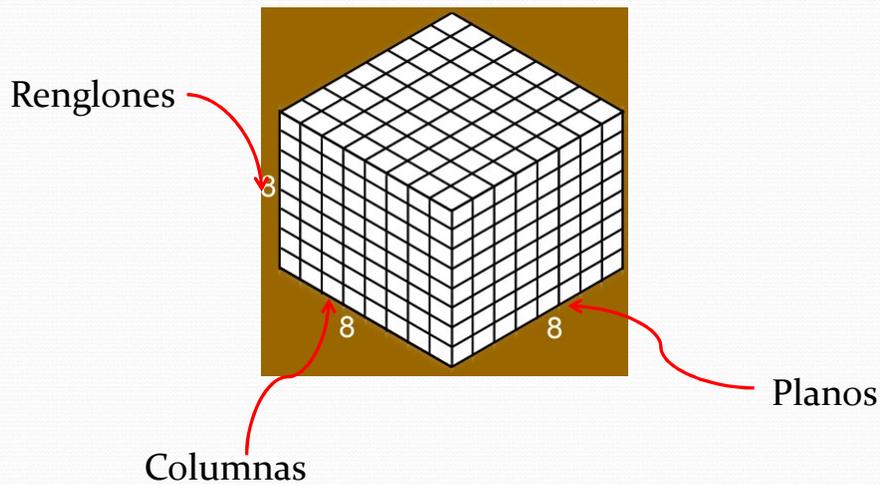
70

Ejemplo de salida

```
Datos de la matriz
15      74      8      15
57      97      2      99
30      2       60     48
Oprima cualquier tecla para continuar...
```

71

Arreglos tridimensionales (cubos)



72

Declaración de cubos

- No sólo basta con declararlo, sino también debe crearse con el operador **new**
- Se declara primero la cantidad de renglones seguido de la cantidad de columnas y luego los planos

```
int [ , , ] miCubo; // declara el cubo
miCubo = new int[2,3,4]; // reserva memoria

double [ , , ] miCubo2 = new double[3,4,5];
```

73

Manejo de cubos

```
int[, ,] miCubo = new int[2, 3, 4];

for(int r=0; r < miCubo.GetLength(0); r++)
  for(int c=0; c < miCubo.GetLength(1); c++)
    for (int p=0; p < miCubo.GetLength(2); p++)
    {
      Console.WriteLine("\nCubo[{0},{1},{2}] ---> ", r, c, p);
      miCubo[r, c, p] = int.Parse(Console.ReadLine());
    }
```

74

Recorrido de un cubo mediante el ciclo *foreach*

```
int [ , , ] miCubo = new int [2,3,4];  
  
foreach(int numero in miCubo)  
{  
    Console.WriteLine("\t" + numero);  
}
```

75

Recorrido de un cubo mediante ciclos *for*

```
int [ , , ] miCubo = new int [2,3,4];  
  
for(int r=0; r < miCubo.GetLength(0); r++)  
    for(int c=0; c < miCubo.GetLength(1); c++)  
        for(int p=0; p < miCubo.GetLength(2); p++)  
            Console.WriteLine( miCubo[r, c, p] );
```

76

CUESTIONARIO

Contestar el
Cuestionario 06.- Arreglos
en la plataforma EaD IT Nuevo
Laredo - Synesi

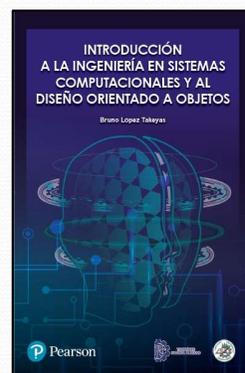
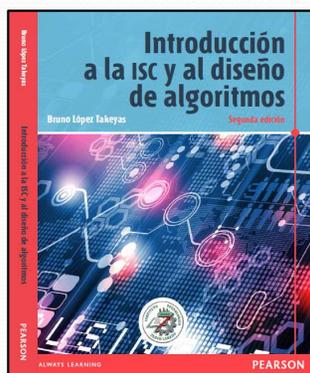
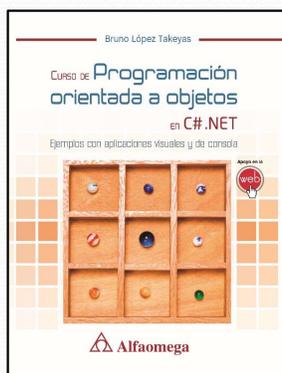


Plataforma EaD IT
Nuevo Laredo

77

Otros libros del autor

<https://nlaredo.tecnm.mx/takeyas/Libro>



✉ bruno.lt@nlaredo.tecnm.mx

f Bruno López Takeyas