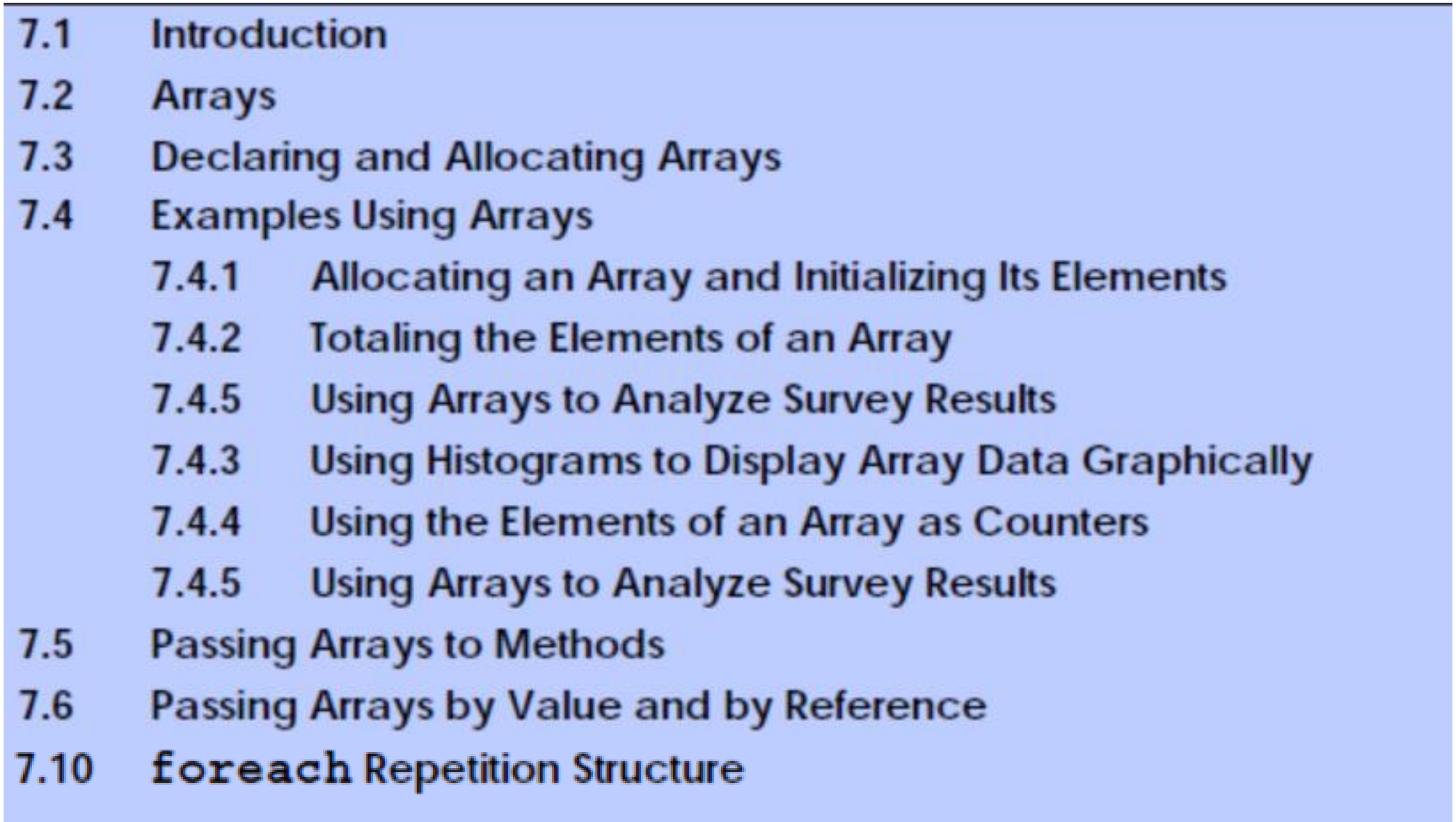
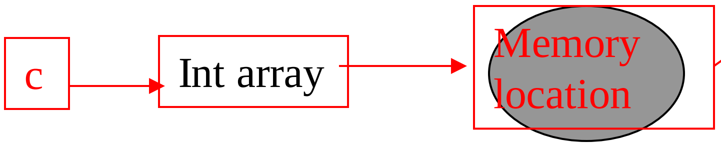
Chapter 7 - Arrays



7.1 Introduction
7.2 Arrays
7.3 Declaring and Allocating Arrays
7.4 Examples Using Arrays
    7.4.1 Allocating an Array and Initializing Its Elements
    7.4.2 Totaling the Elements of an Array
    7.4.5 Using Arrays to Analyze Survey Results
    7.4.3 Using Histograms to Display Array Data Graphically
    7.4.4 Using the Elements of an Array as Counters
    7.4.5 Using Arrays to Analyze Survey Results
7.5 Passing Arrays to Methods
7.6 Passing Arrays by Value and by Reference
7.10 `foreach` Repetition Structure

Arrays

- int c[12];
  - c is the array *name*
  - c has 12 *elements* (c[0], c[1], … c[11])
  - c.length is a variable storing array c's *length* (here, its value is 12)



c[0]	-45
c[1]	6
c[2]	0
c[3]	72
c[4]	1543
c[5]	-89
c[6]	0
c[7]	62
c[8]	-3
c[9]	1
c[10]	6453
c[11]	78

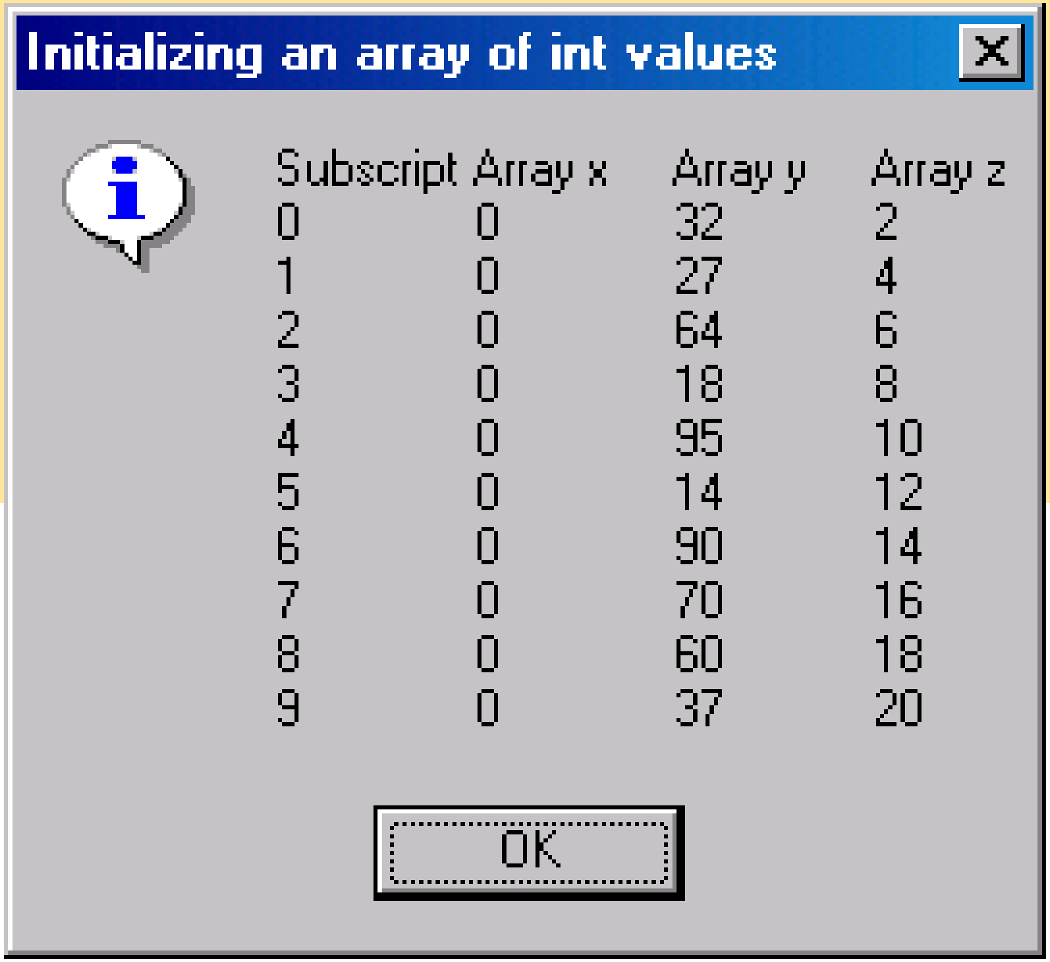
7.3 Declaring and Allocating Arrays

- Array declarations and initializations need not be in the same statement
- **new** operator to allocate dynamically the number of elements in the array
  - “**new**” is to specify how many elements OS will give to the array
- Arrays can be initialized in declaration
  - with *initializer list*
    - The list contains the number of elements and their values

```
using System;
using System.Windows.Forms;
class InitArray
{
  static void Main( string[] args )
   {
     string output = "";
     int[] x;
     x = new int[ 10 ];
     int[] y={32,27,64,18, 95, 14, 90, 70, 60, 37};
     const int ARRAY_SIZE = 10;
     int[] z;
     z = new int[ ARRAY_SIZE ];
     for ( int i = 0; i < z.Length; i++ )
        z[ i ] = 2 + 2 * i;
     output += "Subscript\tArray x\tArray y\tArray z\n";

```

```
      for ( int i = 0; i < ARRAY_SIZE; i++ )
         output += i + "\t" + x[ i ] + "\t" + y[ i ] +
            "\t" + z[ i ] + "\n";
      MessageBox.Show( output,
      "Initializing an array of int values",
      MessageBoxButtons.OK,
      MessageBoxIcon.Information );
   }
}
```



Initializing an array of int values

```
Subscript Array x     Array y     Array z
0         0           32          2
1         0           27          4
2         0           64          6
3         0           18          8
4         0           95          10
5         0           14          12
6         0           90          14
7         0           70          16
8         0           60          18
9         0           37          20
```

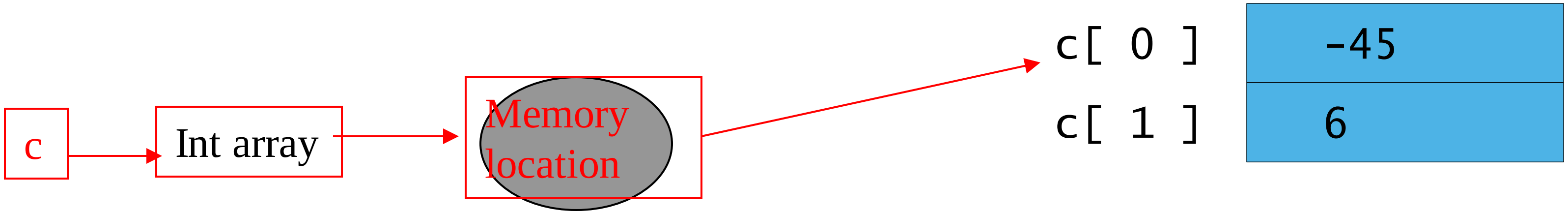
Declaring and Allocating Arrays

- Declaring and Allocating arrays
  - Arrays are objects (i.e., reference variable) that occupy memory
  - Allocated dynamically with operator **new**

```
int[] c = new int[ 2 ];
```

  - The result is equivalent to

```
int[] c;                // declare array
 c = new int[ 2 ]; // allocate array
```



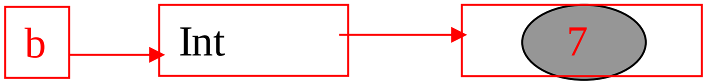
String and general Arrays

- In C, Java, when declare a string variable and assign content to it
  - **One more char** (string stop sign '\0') is appended to the end of the string
    - As the indicator to denote the end of the string
  - In Delphi, no such a char is appended
    - They use the first element of the array to store the length of the array.
      - i.e., array[0] stores the length of the array
- If you want to give the content to a string variable, one char by one char, in the program,
  - you should add '\0' by yourself
  - Otherwise, the string operation will error

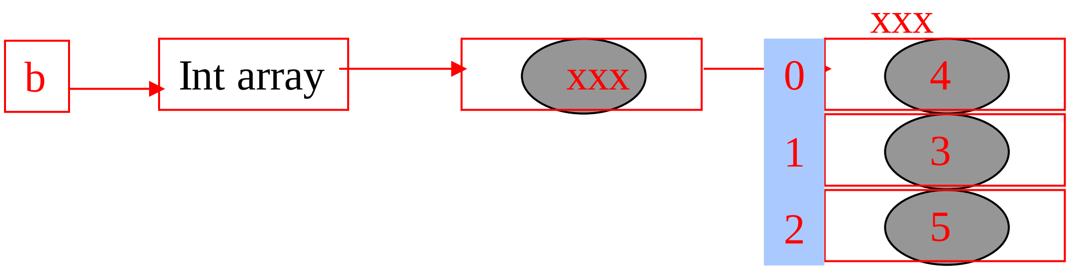
```
string st;
st[1] = 'o';
st[2] = 'k';
st[3] = '\0';
```

Compare String array and general Arrays

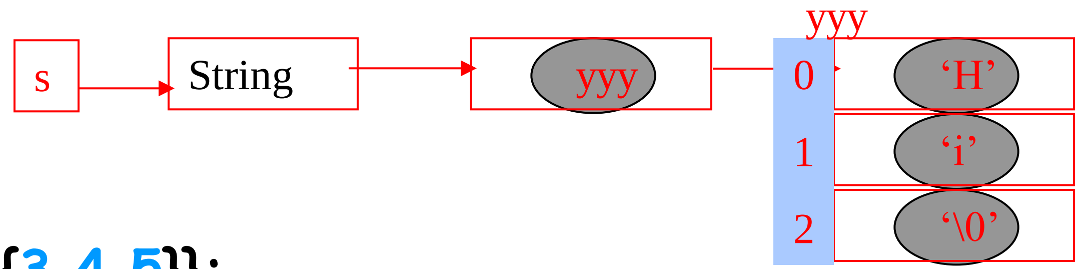
- int b = 7;



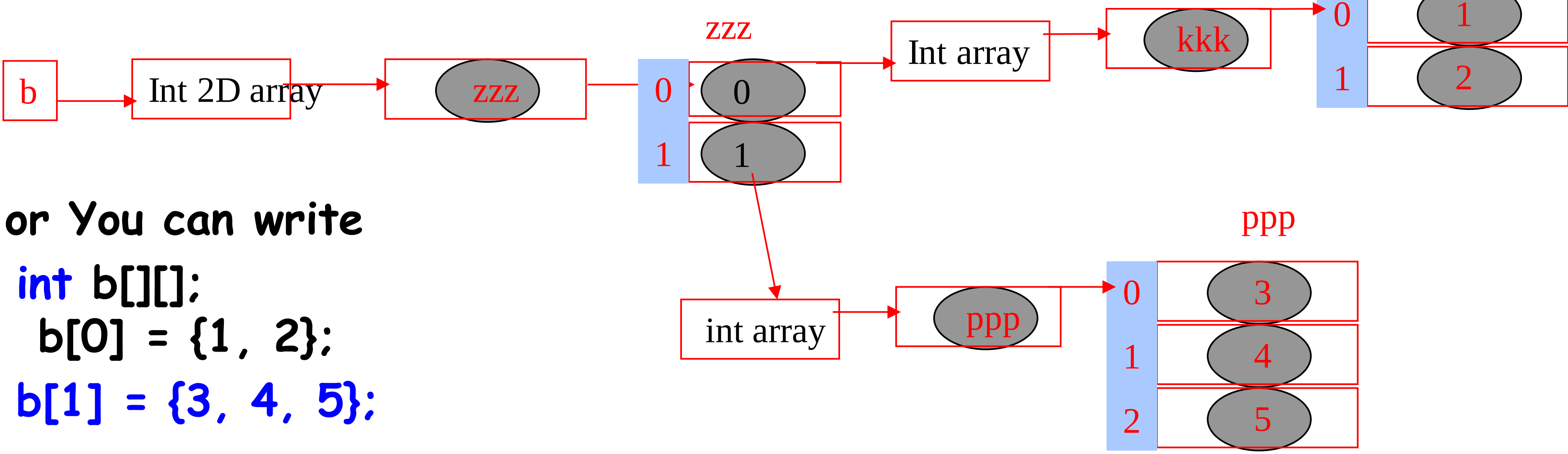
- int[] b = {4, 3, 5};



- string s = “Hi”;



- int[][] b ={{1,2},{3,4,5}};

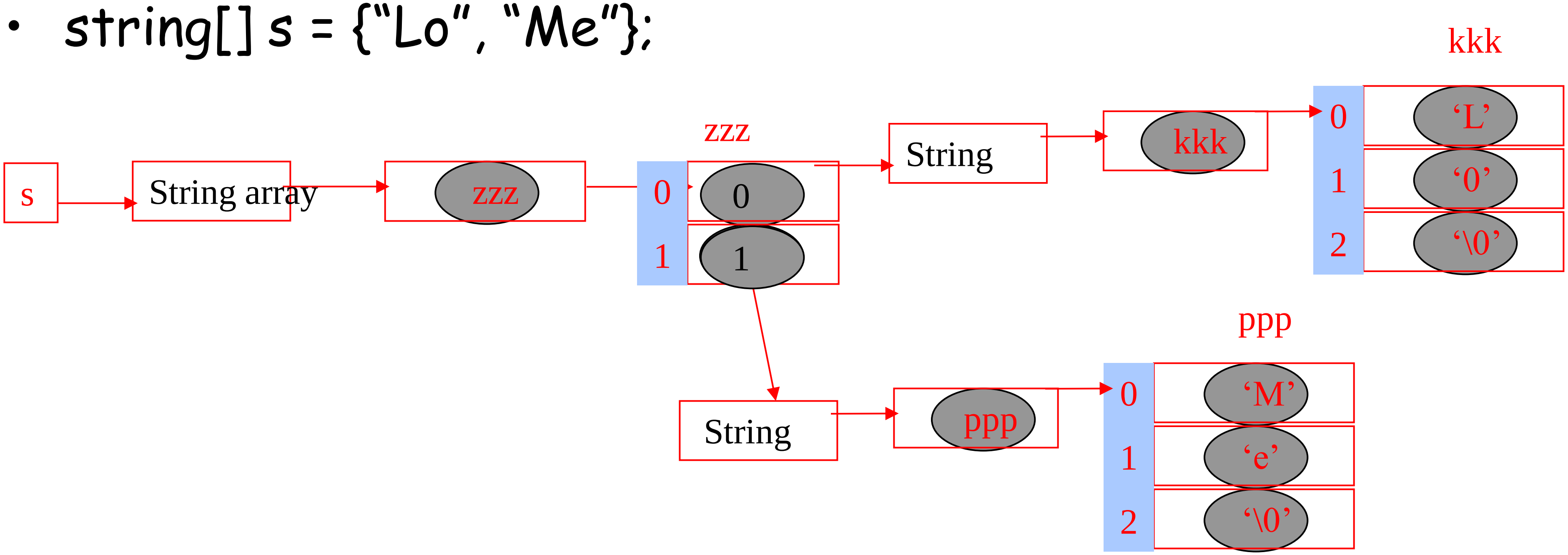


-or You can write

```
int b[][];
 b[0] = {1, 2};
b[1] = {3, 4, 5};
```

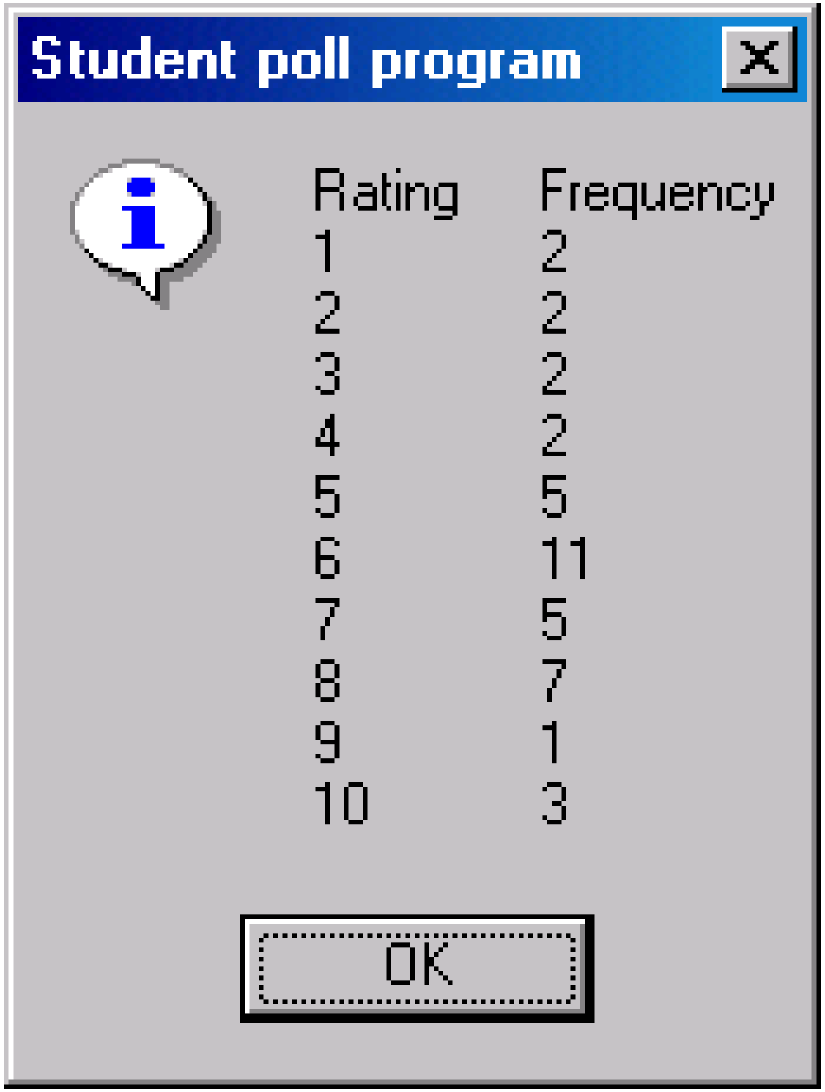
Compare String array and general Arrays

- string[] s = {"Lo", "Me"};



```
using System;
using System.Windows.Forms;
class StudentPoll
{
    static void Main( string[] args )
    {
     int[] responses= {1, 2, 6, 4, 8, 5, 9, 7, 8, 10, 1,
      6, 3, 8, 6, 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7,
      5, 6, 6, 5, 6, 7, 5, 6, 4, 8, 6, 8, 10 };
     int[] frequency = new int[ 11 ];
     string output = "";
     for (int answer=0; answer<responses.Length;
      answer++ )
         ++frequency[ responses[ answer ] ];
     output += "Rating\tFrequency\n";
     for ( int rating = 1; rating < frequency.Length;
      rating++)
         Output +=rating+"\t"+frequency[rating] + "\n";
      MessageBox.Show( output, "Student poll program",
         MessageBoxButtons.OK,
        MessageBoxIcon.Information );
    } }
```

StudentPoll.cs
Program Output



```
Rating   Frequency
1        2
2        2
3        2
4        2
5        5
6        11
7        5
8        7
9        1
10       3
```

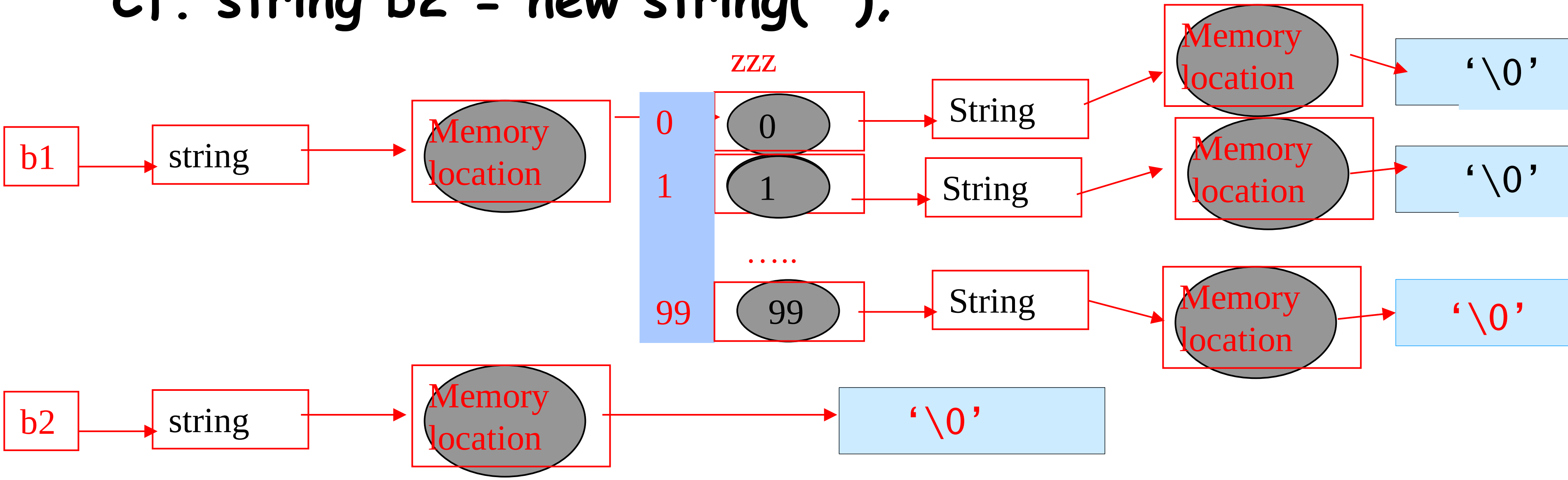
Declaring and Allocating Arrays

- We can allocate arrays of objects, too

  String [] b1 = new String[100];

  // to allocate 100 strings for b

  Cf. string b2 = new string("");



- String a=new String("aaaaa"); 和 String a; a="aaaaa"; 有區別嗎？

Passing Arrays to Methods

- In Java, C#, every object is pass-by-reference
  - The called and caller share the same data space (i.e., the fourth box shared by both)
- In Java, C#, arrays are objects
  - Therefore, arrays are passed to methods by reference
  - But if passing an element of array, the element is passed by value
- To pass array argument to a method
  - Specify array name without brackets [ ], like
    - Array **hourlyTemperatures** is declared as

      ```
      int[] hourlyTemperatures = new int[24];
      ```

    - The method calling is written as

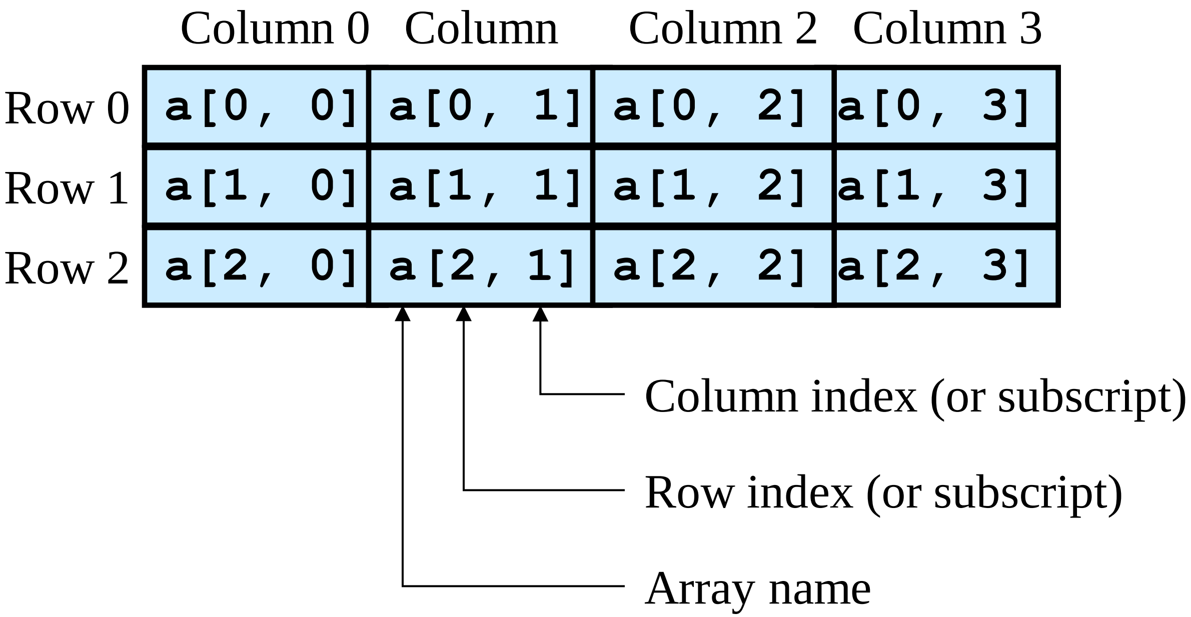
      ```
      modifyArray(hourlyTemperatures);
      ```

7.9 Multiple-Subscripted Arrays

- Rectangular arrays
  - Often represent tables in which each row is the same size and each column is the same size
  - By convention, first subscript identifies the element's row and the second subscript the element's column
- Jagged Arrays
  - Arrays of arrays
  - Arrays that compose jagged arrays can be of different lengths

◀ ▶

7.9 Multiple-Subscripted Arrays



	Column 0	Column	Column 2	Column 3
Row 0	`a[0, 0]`	`a[0, 1]`	`a[0, 2]`	`a[0, 3]`
Row 1	`a[1, 0]`	`a[1, 1]`	`a[1, 2]`	`a[1, 3]`
Row 2	`a[2, 0]`	`a[2, 1]`	`a[2, 2]`	`a[2, 3]`

Column index (or subscript)

Row index (or subscript)

Array name

Fig. 7.13 Double-subscripted array with three rows and four columns.

7.10 foreach Repetition Structure

- foreach
  - repetition structure
  - is used to iterate through values in data structures such as arrays
  - A variable is used as a counter to represent the value of each element

```
using System;
class ForEach
{
   static void Main( string[] args )
   {
      int[,] gradeArray = { { 77, 68, 86, 73 },
         { 98, 87, 89, 81 }, { 70, 90, 86, 81 } };
      int lowGrade = 100;
      foreach ( int grade in gradeArray )
      {
         if ( grade < lowGrade )
            lowGrade = grade;
      }
      Console.WriteLine( "The minimum grade is: " + lowGrade );
   }
}
```

```
The minimum grade is: 68
```