CGT 456

Arrays

Declaring – Single Dimension

- private int[] x;
- private int[] numbers; //declare numbers as an int array of any size
- private string[] words; //declare words as a string array of any size
- private dog[] myDog; //declare myDog as a dog array of any size

Creating a new instance

- After you declare the array, you can specify the size:

- numbers = new int[7]; //numbers is a 7-element array
- numbers = new int[15]; //now it's a 15-element array

- words = new string[5]; //words is a 5-element array
- words = new string[20]; //now it’s a 20-element array

- myDog = new dog[3]; //myDog is an array of 3 dogs
- myDog = new dog[30]; //now it’s a 30-element array

Initializing

- int[] numbers = new int[5] {1, 2, 3, 4, 5};
- string[] words = new string[3] {"Bottle", "Cup", "Art"};
- // dog is a little more involved
  - private dog doggie1, doggie2;

    …
  - dog doggie1 = new dog();
  - dog doggie2 = new dog();
  - dog[] myDog = new dog[2] {doggie1, doggie2};

Retrieving values from array

- numbers[2] //accesses the 3rd element of the array
- words[0] //accesses the 1st element of the array
- myDog[5] //accesses the 6th element of the array

- numbers[3] = 5;
  - //sets the 4th element equal to the number 5
- words[1] = “aardvark”;
  - //sets the 2nd element equal to “aardvark”
- myDog[2] = doggie1;
  - //sets the 3rd element equal to the dog object: doggie1

Length of an array

- int lengthOfNums, lengthOfWords, lengthOfDog;

- lengthOfNums = numbers.Length;
- lengthOfWords = words.Length;
- lengthOfDog = myDog.Length;

Length of an array

-
  ```
  for(int i=0; i < words.Length; i++)
  {
      Response.Write(words[i].ToString());
  }
  ```

Alternately – using foreach

-
  ```
  foreach(int i in words)
  {
      Response.Write(i);
  }
  ```

Declaring – Two Dimensional

- private int[,] x;
  - private int[,] counters;
    - //declare counters as a 2-dimensional int array of any size
  - private string[,] names;
    - //declare names as a 2-dimensional string array of any size
  - private cat[,] kittens;
    - //declare kittens as a 2-dimensional cat array of any size

Creating a new instance

- After you declare the array, you can specify the size:
  - counters = new int[7,7]; //counters has 7 rows and 7 cols
  - counters = new int[3,7]; //now it has 3 rows and 7 cols

  - names = new string[5,4]; //names has 5 rows and 4 cols
  - names = new string[2,2]; //now it has 2 rows and 2 cols

  - kittens = new cat[3,3]; //kittens has 3 rows and 3 cols
  - kittens = new cat[9,9]; //now it has 9 rows and 9 cols

Initializing (3 ways to do the same thing)

-
  ```
  int[,] counters = new int[2,3] {{1, 2, 3},
                                  {4, 5, 6}
                                 };
  ```
- **OR**

-
  ```
  int[,] counters  =  new int[,] {{1, 2, 3},
                                  {4, 5, 6}
                                 };
  ```
- **OR**

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

Initializing (3 ways to do the same thing)

```
string[,] names = new string[3,2]{{“Sam”, “Tom”},
                                  {“Pat”, “Jim”},
                                  {“Scott”, “Craig”}
                                 };
```

- **OR**

```
string[,] names =  new string[,] {{“Sam”, “Tom”},
                                  {“Pat”, “Jim”},
                                  {“Scott”, “Craig”}
                                 };
```

- **OR**

```
string[,] names =  {{“Sam”, “Tom”},
                    {“Pat”, “Jim”},
                    {“Scott”, “Craig”}
                   };
```

Initializing (3 ways to do the same thing)

- //cat is a little more involved
  - private cat kitten1, kitten2, kitten3, kitten4;
    …
  - cat kitten1 = new cat();
  - cat kitten2 = new cat();
  - cat kitten3 = new cat();
  - cat kitten4 = new cat();
- //continued on next slide…

Initializing (3 ways to do the same thing)

- //continued from previous slide…

```
cat[,] litter = new cat[2,2] {{kitten1, kitten2},
                              {kitten3, kitten4}
                             };
```

- **OR**

```
cat[,] litter =  new cat[,]  {{kitten1, kitten2},
                              {kitten3, kitten4}
                             };
```

- **OR**

```
cat[,] litter =  {{kitten1, kitten2},
                  {kitten3, kitten4}
                 };
```

Declare & Initialize a 9x9 int array

```
private int[,] solution1 = { {7,9,2,3,5,1,8,4,6},
                             {4,6,8,9,2,7,5,1,3},
                             {1,3,5,6,8,4,7,9,2},
                             {6,2,1,5,7,9,4,3,8},
                             {5,8,3,2,4,6,1,7,9},
                             {9,7,4,8,1,3,2,6,5},
                             {8,1,6,4,9,2,3,5,7},
                             {3,5,7,1,6,8,9,2,4},
                             {2,4,9,7,3,5,6,8,1}
                           };
```

Retrieving values from array

- counters[0,2]
  - //accesses the integer in the 1st row, 3rd column of the array
- names[1,0]
  - //accesses the string in the 2nd row, 1st column of the array
- cat[5,4]
  - //accesses the cat object in the 6th row, 5th column of the array

- counters[3,1] = 5;
  - //sets the integer in the 4th row, 2nd column of the array equal to the number 5
- names[1,3] = “Harry”;
  - //sets the string in the 2nd row, 4th column of the array equal to “Harry”
- cat[0,1] = kitten1;
  - //sets the cat object in the 1st row, 2nd column equal to the cat object: kitten1

Length of a 2-dimensional array

-
  ```
  int[,] solution  =  { {1,2,3,4},
                        {5,6,7,8},
                        {9,10,11,12}
                      };
  ```

- Response.Write(solution.Length);
  - //writes out: 12
  - //there are 12 values in the array

for loop for a 2-dimensional array

```
//rows
for (int i = 0; i < 3; i++)
{
    //cols
    for (int k = 0; k < 4; k++)
    {
        //check for last array item-don't put comma after last one
        if( ((i+1) * (k+1)) == solution.Length)
                Response.Write(solution[i,k].ToString());
        else
                Response.Write(solution[i,k].ToString() + ", ");

    } //end inner for loop
} //end outer for loop

//writes out:   1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
```

More Advanced…

- 3-dimensional array:

  -
    ```
    int[,,] items = new int[3,4,5];
    ```
- Jagged array:

  -
    ```
    int[][] numbers = {new int[]{1,2,3},
                       new int[]{4,5,6,7,8,9} };
    ```

- There are others…