Arrays

Definition

An array is a group of consecutive memory locations (contiguous) all consisting of the same type.

Individual memory elements within the array can be accessed via a common identifier (the name of the array) and an integer-valued expression. We will see how this is done as we develop and use arrays.

Reference Type

Arrays in C# are objects, and they are of reference type.

Defining an Array in C#

There are two steps in establishing an array.

First: Declare an identifier as an array type:

Second: Allocate the actual array object.

Example:

```
int[ ] Grades;  //Grades will reference an
                //array of integers:
Grades = new int[50];  //Allocate space for
                       //50 integers
```

Same Declaration

```
int[ ] Grades = new int[50];
```

In general,

in C#, given a *type* , the presence of [ ] following the type, represents an array type whose entries are of the specified type:

```
int[ ]            //array of integers
char[ ]           //array of characters
double[ ]         //an array of doubles.
```

Yet another way to declare an array.

```
int[ ] List = {1,2,10,20,30,11,12};
```

This declares and initializes the array.

Here, we are not specifically calling the new to allocate memory at runtime. List will consist of 7 integer elements containing the numbers 1, 2, 10, 20, 30, 11, & 12 respectively.

Accessing Individual Elements

If an array has N elements (N>0), then the elements in that array can be accessed by the array’s name and an index that ranges from 0 to N-1 inclusive.

Consider the example on the next slide.

int[ ] List = {1,2,10,20,30,11,12};

List[0] has value 1.

List[1] has value 2.

List[2] has value 10.

List[3] has value 20.

List[4] has value 30.

List[5] has value 11.

List[6] has value 12.

Terminology

In the example on the previous page:

List is the “name” of the array. Like a variable identifier it can be any legal user-defined identifier.

List[i] Placing [ ] after the name “dereferences” the array and gives the individual element. The quantity inside the [ ] is called the index. It must be an integer valued expression whose value is from 0 to N-1 where N is the number of elements in the array.

In general,

the name of an array is a reference type. When it is dereferenced as indicated before (name[expression]) the result might be a value type or a reference type, depending on the type of the array.

Example: List[i] from our previous slides is a value type, since List is an array of integers.

Length Property

Objects can have properties defined for them. Arrays are objects and Length is a property associated with any array object. A property is accessed using the object name, the dot operator, and the property name. For example:

int[ ] X = {8,3,4,2,3,4}; X.Length has value 6.

There are six items in the array. Note that this is analogous to the Length property associated with strings.

Look at a problem:

Declare an array of size 1000 and fill it with random integers in the range 1 to 20.

Problem (cont.)

Now add the code that will count the number of times each number in the range 1 to 20 was written into the array.

Problem (cont.)

Now do the same thing as the previous two steps, but generate and count the 1000 numbers in the range 1 to 20, without storing all the numbers.

Problem 2.

Write a program that will ask the user to input the number of random birthdays (numbers in the range 1 to 365) that she would like to generate. Then generate that many random birthdays, and report the ones that were hit more than once.

This simulates the experiment we ran in class in which we looked at 40 random birthdays.

Problem 3

An experiment consists of choosing N people at random and seeing if two have the same birthday.

Write a program that will ask the user to input N, and then simulate 10000 experiments. Report the number of experiments in which two people had the same birthday.

Consider the following method:

```
static int FindMax(int[ ] X)
{
   int Max = X[0];
   for(int i = 1; i<X.Length; i++)
      if(X[i]>Max)
      Max = X[i];
   return Max;
}
```

Questions about the method:

- What is its return type?
- How many parameters are there and what are their types?
- What is the method name?
- What is the method doing?
- Given an array B of integers, how would you call this method and pass B to it?

For Example:

```
int[] B = {4,4,2,3,7,3,2,4,3,8,7,5};

Console.WriteLine(FindMax(B));
```

We will discuss what is going on with this call to the method, and describe how B is being passed to the method.