Contents

The foreach Loop

1

The foreach Loop

 When writing a for loop that accesses each element of an array once, you will end up writing code like this:

```
for(int i = 0; i < myArray.Length; i++)
{
      <do something with myArray[i]>;
}
```

- In some cases, this code has unnecessary repetition: If you are not using the counter i for anything other than an array index, you still need to declare it, increment it, and write the condition with myArray. Length
- The **foreach loop** is a shortcut that allows you to get rid of the counter variable and the loop condition. It has this syntax:

```
foreach(<type> <variableName> in <arrayName>)
{
      <do something with variable>
}
```

- The loop will repeat exactly as many times as there are elements in the array
- On each iteration of the loop, the variable will be assigned the next value from the array, in order
- The variable must be the same type as the array
- For example, this loop accesses each element of homeworkGrades and computes their sum:

```
int sum = 0;
foreach(int grade in homeworkGrades)
{
    sum += grade;
}
```

- The variable grade is declared with type int since homeworkGrades is an array of int
- grade has a scope limited to the body of the loop, just like the counter variable i
- In successive iterations of the loop grade will have the value homeworkGrades[0], then homeworkGrades[1], and so on,

through homeworkGrades[homeworkGrades.Length - 1]

• A **foreach** loop is **read-only** with respect to the array: The loop's variable cannot be used to *change* any elements of the array. This code will result in an error:

```
foreach(int grade in homeworkGrades)
{
    grade = int.Parse(Console.ReadLine());
}
```