100 XP

Exercise - Construct a nested loop structure for student grade calculations

25 minutes

In this exercise, you'll add a string array to hold the student names, and then implement a nested `foreach` structure that iterates through the student names in an outer loop and student scores in the inner loop. You'll begin by constructing the `studentNames` array and a `foreach` loop that iterates through the array elements. Next, you'll move the code that's used to calculate Sophia's grades into the code block of the "names" loop. Finally, you'll implement the code logic that uses the student's name to access their scores array, calculate their average score, and write their grade to the console. The detailed tasks that you'll complete during this exercise are:

1. Create names array: Create a student names array.
2. Create outer loop: Create a `foreach` loop that iterates through the student names.
3. Develop outer loop code block: Relocate the code that calculates and reports Sophia's score, placing it in the code block of the "names" loop.
4. Update calculations and reporting: Update the code that performs student score calculations using a new scores array.

> **Important**
>
> You need to have completed this module's previous Exercise, "Create arrays and foreach loops", before you begin this Exercise.

Create a student names array and outer foreach loop

In this task, you'll create a student names array and a `foreach` loop that iterates through the student names.

1. Ensure that you have the Program.cs file open in the Visual Studio Code Editor.

2. Scroll to the top of your code file, and then locate the code lines that are used to declare the scores arrays.

3. Create a blank code line below the declaration of the scores arrays.

   Your blank code line should be located between the lines used to declare the scores arrays and the line used to declare `sophiaSum`.

4. To create a string array named `studentNames` that holds the names of the students, enter the following code:

   ```
   C#

   // Student names
   string[] studentNames = new string[] { "Sophia", "Andrew", "Emma", "Logan" };
   ```

   Notice that you've specified the student names as part of the declaration.

5. To create a `foreach` statement that you can use to iterate through the student names, enter the following code:

   ```
   C#

   foreach (string name in studentNames)
   {
   }
   ```

6. To verify that your `foreach` loop is iterating through the `studentNames` array as intended, update the code block of the `foreach` statement as follows:

   ```
   C#

   foreach (string name in studentNames)
   {
       Console.WriteLine($"{name}");

   }
   ```

7. Take a minute to review the code that you've created.

   ```
   C#
   ```

```
// Student names
string[] studentNames = new string[] { "Sophia", "Andrew", "Emma", "Logan" };

foreach (string name in studentNames)
{
    Console.WriteLine($"{name}");

}
```

Your code will use this `foreach` loop as the outer loop of your application. During this exercise, you'll implement the following logic in your application:

For each of the students in the `studentNames` array, your application will:

- determine the current student.
- access the current student's scores.
- calculate the current student's grade (sum and average).
- write the current student's grade to the console.

For now, however, you'll just write the names of the students to the console.

8. On the Visual Studio Code **File** menu, click **Save**.

9. In the Visual Studio Code EXPLORER panel, right-click **Starter**, and then select **Open in Integrated Terminal**.

> **ⓘ Important**
>
> The Terminal command prompt must be displaying the folder path for your Program.cs file.

10. At the Terminal command prompt, type **dotnet build** and then press Enter.

    The `dotnet build` command instructs the compiler to build the application. If any errors are detected, they will be reported.

11. If you see Error or Warning messages, you need to fix them before continuing.

12. At the Terminal command prompt, type **dotnet run** and then press Enter.

13. Verify that your code produced the following output:

Output

```
Sophia
Andrew
Emma
Logan
Student         Grade

Sophia:         92.2    A-
Press the Enter key to continue
```

> **Note**
>
> If you don't see the list of student names above Sophia's score report, go back and verify that you entered your code correctly.

14. In the TERMINAL panel, to stop your running application, press the Enter key.

15. Close the Terminal panel.

Calculate Sophia's score inside the outer names loop

In this task, you'll relocate the code that calculates and reports Sophia's score, placing it in the code block of the "names" loop.

1. In the Visual Studio Code Editor, locate the code lines that are used to calculate and report Sophia's grade.

C#

```
int sophiaSum = 0;

decimal sophiaScore;

foreach (int score in sophiaScores)
{
    // add the exam score to the sum
    sophiaSum += score;

}

sophiaScore = (decimal)sophiaSum / currentAssignments;
```

```
Console.WriteLine("Student\t\tGrade\n");
Console.WriteLine("Sophia:\t\t" + sophiaScore + "\tA-");
```

> **Note**
>
> Your next step will be to move this code from its current location to the code block of the "names" `foreach` loop.

2. Use a cut-and-paste operation to move the code that calculates and reports Sophia's grade to the code block of the "names" `foreach` loop.

   If you're unsure how to cut-and-paste in Visual Studio Code, try the approach described in the following steps:

   a. Select the code that's used to calculate and report Sophia's grade.

      You can click-and-drag to select code lines.

   b. On the Visual Studio Code **Edit** menu, select **Cut**.

   c. In the Visual Studio Code Editor, position the cursor on the blank code line below the following code: `Console.WriteLine($"{name}");`

   d. On the Visual Studio Code **Edit** menu, select **Paste**.

3. Update your code to display proper code line indentation.

> **Tip**
>
> Visual Studio Code provides a `Format Document` command that can be used to keep your code formatting updated. Right-click inside the Visual Studio Code Editor, and then select **Format Document** from the popup menu. On Windows computers, the keyboard shortcut for this command is: `Shift + Alt + F`. Linux and macOS computers use alternative shortcuts when necessary to avoid conflicts with OS supplied shortcuts.

4. Ensure that your updates match the following code:

```
C#
```

```
// Student names
string[] studentNames = new string[] { "Sophia", "Andrew", "Emma", "Logan" };

foreach (string name in studentNames)
{
    Console.WriteLine($"{name}");
    int sophiaSum = 0;

    decimal sophiaScore;

    foreach (int score in sophiaScores)
    {
        // add the exam score to the sum
        sophiaSum += score;

    }

    sophiaScore = (decimal)sophiaSum / currentAssignments;

    Console.WriteLine("Student\t\tGrade\n");
    Console.WriteLine("Sophia:\t\t" + sophiaScore + "\tA-");
}

Console.WriteLine("Press the Enter key to continue");
Console.ReadLine();
```

Notice that at this point, your code will calculate and report Sophia's score regardless of the `name` of the current student. You will address that shortly.

5. Delete the following code:

```
C#

Console.WriteLine($"{name}");
```

6. On the blank code line that you just created, enter the following code:

```
C#

if (name == "Sophia")
{
```

7. Create a blank code line after the code that's used to write Sophia's grade to the console.

8. To close the code block of the `if` statement, enter the following code:

C#

```
}
```

9. Update your code to display proper code line indentation.

> **Tip**
>
> Use the `Format Document` command to keep your code formatting updated. Right-click inside the Visual Studio Code Editor panel, and then select **Format Document** from the popup menu.

10. Take a minute to review your code.

    Your code should match the following code:

C#

```
// initialize variables - graded assignments
int currentAssignments = 5;

int[] sophiaScores = new int[] { 90, 86, 87, 98, 100 };
int[] andrewScores = new int[] { 92, 89, 81, 96, 90 };
int[] emmaScores = new int[] { 90, 85, 87, 98, 68 };
int[] loganScores = new int[] { 90, 95, 87, 88, 96 };

// Student names
string[] studentNames = new string[] {"Sophia", "Andrew", "Emma", "Logan"};

foreach (string name in studentNames)
{
    if (name == "Sophia")
    {
        int sophiaSum = 0;
        decimal sophiaScore;

        foreach (int score in sophiaScores)
        {
            // add the exam score to the sum
            sophiaSum += score;
        }

        sophiaScore = (decimal)(sophiaSum) / currentAssignments;

        Console.WriteLine("Student\t\tGrade\n");
        Console.WriteLine("Sophia:\t\t" + sophiaScore + "\tA-");
```

```
    }
}

Console.WriteLine("Press the Enter key to continue");
Console.ReadLine();
```

Notice that the `if` statement inside your outer `foreach` code block limits which student's grade is calculated and reported. This isn't exactly what you need, but it's a step in the right direction.

11. On the Visual Studio Code **File** menu, click **Save**.
12. In the Visual Studio Code EXPLORER panel, right-click **Starter**, and then select **Open in Integrated Terminal**.

> **ⓘ Important**
>
> The Terminal command prompt must be displaying the folder path for your Program.cs file.

13. At the Terminal command prompt, type **dotnet build** and then press Enter.

    The `dotnet build` command instructs the compiler to build the application. If any errors are detected, they will be reported.

14. If you see Error or Warning messages, you need to fix them before continuing.
15. At the Terminal command prompt, type **dotnet run** and then press Enter.
16. Verify that your code produced the following output:

Output

```
Student         Grade

Sophia:         92.2    A-
```

> **ⓘ Note**
>
> If you still see the list of student names displayed above Sophia's score report, make sure that you saved your updates.

17. In the TERMINAL panel, to stop your running application, press the Enter key.

18. Close the Terminal panel.

Update the nested loop to calculate all student scores

In this task, you'll update the code that performs student score calculations using a new scores array. You'll begin by creating an array named `studentScores` that can be used to hold the scores of any student. Next, you'll create an `if .. elseif` construct that uses the current student's name to assign their scores array to `studentScores`. Finally, you'll update the code that calculates and reports the student's grades. When you're done, the report should include the name and numeric score for all students.

1. Create a blank code line below the declaration of the `studentNames` array.

   The blank line should be above the outer `foreach` statement.

2. To create an integer array that you can use to hold the scores of the current student, enter the following code:

   ```
   C#

   int[] studentScores = new int[10];
   ```

   Notice that this code doesn't assign any values to the array at this point. You simply specify that the array can contain 10 elements.

3. Create a blank code line at the top of the outer `foreach` code block.

   The blank line should be inside the `foreach` code block and above the `if` statement that evaluates whether `name` is equal to Sophia.

4. To create a string variable that will be used to hold the name of the current student, enter the following code:

   ```
   C#

   string currentStudent = name;
   ```

> **Note**
>
> You could continue to use `name` to track the name of the current student as you iterate through the names array, but using `currentName` will make it easier to understand your code logic as you build out your application in the upcoming steps.

5. To substitute `currentStudent` for `name` in the `if` statement that evaluates whether `name` is equal to Sophia, update your code as follows:

```
C#

if (currentStudent == "Sophia")
```

6. Move the code that calculates and reports Sophia's score to a location below the code block.

   You're moving all of the code that's in the code block to a location below the code block. The reason for doing this will become apparent during the next few steps.

7. Verify that the code in your outer `foreach` code block matches the following code:

```
C#

{
    string currentStudent = name;

    if (currentStudent == "Sophia")
    {
    }

    int sophiaSum = 0;
    decimal sophiaScore;

    foreach (int score in sophiaScores)
    {
        // add the exam score to the sum
        sophiaSum += score;

    }

    sophiaScore = (decimal)sophiaSum / currentAssignments;
```

```
    Console.WriteLine("Student\t\tGrade\n");
    Console.WriteLine("Sophia:\t\t" + sophiaScore + "\tA-");

}
```

8. To assign the `sophiaScores` array to `studentScores` when `currentStudent == "Sophia"`, update your `if` statement code as follows:

```
C#

if (currentStudent == "Sophia")
    studentScores = sophiaScores;
```

   Notice that you've removed the curly braces from the `if` statement code block during this code update.

9. To add an `else if` statement that assigns the `andrewScores` array to `studentScores` when `currentStudent == "Andrew"`, enter the following code:

```
C#

else if (currentStudent == "Andrew")
    studentScores = andrewScores;
```

10. Create another `else if` statement to assign the `emmaScores` array to `studentScores` when `currentStudent == "Emma"`.

11. Create an `else if` statement to assign the `loganScores` array to `studentScores` when `currentStudent == "Logan"`.

12. Ensure that your `foreach` code block matches the following code:

```
C#

foreach (string name in studentNames)
{
    string currentStudent = name;

    if (currentStudent == "Sophia")
        studentScores = sophiaScores;
```

```
    else if (currentStudent == "Andrew")
        studentScores = andrewScores;

    else if (currentStudent == "Emma")
        studentScores = emmaScores;

    else if (currentStudent == "Logan")
        studentScores = loganScores;

    int sophiaSum = 0;

    decimal sophiaScore;

    foreach (int score in sophiaScores)
    {
        // add the exam score to the sum
        sophiaSum += score;

    }

    sophiaScore = (decimal)sophiaSum / currentAssignments;

    Console.WriteLine("Student\t\tGrade\n");
    Console.WriteLine("Sophia:\t\t" + sophiaScore + "\tA-");

}
```

Next, you need to update the inner `foreach` loop to use `studentScores` and "depersonalize" the variables that you use in your calculations.

13. To substitute `studentScores` for `sophiaScores` in the `foreach` loop that iterates through the scores array, update your code as follows:

```
C#

foreach (int score in studentScores)
```

14. To replace the Sophia-specific variable declarations with more generic names, update your code as follows:

```
C#

int sumAssignmentScores = 0;

decimal currentStudentGrade = 0;
```

These two variable declarations are intended to replace the following Sophia-specific variable declarations:

```
C#

int sophiaSum = 0;

decimal sophiaScore;
```

15. To apply the new variable name to the equation used to sum student scores, update your inner `foreach` code block as follows:

```
C#

foreach (int score in studentScores)
{
    // add the exam score to the sum
    sumAssignmentScores += score;
}
```

16. To apply the new variable name to the equation used to calculate the average score, update your code as follows:

```
C#

currentStudentGrade = (decimal)(sumAssignmentScores) / currentAssignments;
```

17. Take a minute to review your code.

```
C#

int[] studentScores = new int[10];

foreach (string name in studentNames)
{
    string currentStudent = name;

    if (currentStudent == "Sophia")
        studentScores = sophiaScores;

    else if (currentStudent == "Andrew")
        studentScores = andrewScores;

    else if (currentStudent == "Emma")
        studentScores = emmaScores;
```

```
    else if (currentStudent == "Logan")
        studentScores = loganScores;

    // initialize/reset the sum of scored assignments
    int sumAssignmentScores = 0;

    // initialize/reset the calculated average of exam + extra credit scores
    decimal currentStudentGrade = 0;

    foreach (int score in studentScores)
    {
        // add the exam score to the sum
        sumAssignmentScores += score;
    }

    currentStudentGrade = (decimal)(sumAssignmentScores) / currentAssignments;

    Console.WriteLine("Student\t\tGrade\n");
    Console.WriteLine("Sophia:\t\t" + sophiaScore + "\tA-");

}
```

Your nested `foreach` loops will now iterate through the student names and use the student's scores to calculate their grades, but you still need to update the code used to generate the score report.

18. To print the student name and calculated score to the console, update the second `Console.WriteLine` statement as follows:

    C#

    ```
    Console.WriteLine($"{currentStudent}\t\t{currentStudentGrade}\t?");
    ```

    Notice that this code has replaced the letter grade assignment with a "?". You will work on automating the assignment of letter grades in the next exercise.

19. Move the `Console.WriteLine` statement that's used to write the column labels of your score report to the location just above the outer `foreach` loop.

    You don't want to repeat the column headers for each student score, so you move this code to a point above the outer `foreach` loop.

20. On the Visual Studio Code **File** menu, click **Save**.

21. Take a minute to review your application code.

    Your full application should now match the following code:

C#

```
// initialize variables - graded assignments
int currentAssignments = 5;

int[] sophiaScores = new int[] { 90, 86, 87, 98, 100 };
int[] andrewScores = new int[] { 92, 89, 81, 96, 90 };
int[] emmaScores = new int[] { 90, 85, 87, 98, 68 };
int[] loganScores = new int[] { 90, 95, 87, 88, 96 };

// Student names
string[] studentNames = new string[] { "Sophia", "Andrew", "Emma", "Logan" };

int[] studentScores = new int[10];

// Write the Report Header to the console
Console.WriteLine("Student\t\tGrade\n");

foreach (string name in studentNames)
{
    string currentStudent = name;

    if (currentStudent == "Sophia")
        studentScores = sophiaScores;

    else if (currentStudent == "Andrew")
        studentScores = andrewScores;

    else if (currentStudent == "Emma")
        studentScores = emmaScores;

    else if (currentStudent == "Logan")
        studentScores = loganScores;

    // initialize/reset the sum of scored assignments
    int sumAssignmentScores = 0;

    // initialize/reset the calculated average of exam + extra credit scores
    decimal currentStudentGrade = 0;

    foreach (int score in studentScores)
    {
        // add the exam score to the sum
        sumAssignmentScores += score;
    }
```

```
    currentStudentGrade = (decimal)(sumAssignmentScores) / currentAssignments;

    Console.WriteLine($"{currentStudent}\t\t{currentStudentGrade}\t?");
}
```

With the code that generates the student's score report updated; it appears that you're ready to check your work.

Check your work

In this task, you'll run the application to verify that your code logic is working as expected.

1. Ensure that you've saved your changes to the Program.cs file.
2. In the Visual Studio Code EXPLORER panel, right-click **Starter**, and then select **Open in Integrated Terminal**.
3. At the Terminal command prompt, type **dotnet build** and then press Enter.
4. If you see Error or Warning messages, you need to fix them before continuing.
5. At the Terminal command prompt, type **dotnet run** and then press Enter.
6. Verify that your code produced the following output:

Output

```
Student         Grade

Sophia          92.2    ?
Andrew          89.6    ?
Emma            85.6    ?
Logan           91.2    ?
Press the Enter key to continue
```

7. In the TERMINAL panel, to stop your running application, press the Enter key.
8. Close the Terminal panel.

Congratulations, your application has come a long way from where you started out. You are making efficient use of arrays and `foreach` iterations, and you've integrated an `if` statement that enables your code to select the correct scores array.

Next unit: Exercise - Implement code branches using selection statements

Continue >