Tutorial 1 – Fundamentals of C#

OOP 4200 – T. MacDonald

References

Chapters 1 - 7

Aim/Objectives

The aim of this tutorial is to re-familiarize you with the fundamentals of C#. Some of these topics were taught directly in C# in your Intro to Programming course. Others were covered in C++ in either OOP I or OOP II. The specific topics you will review are:

- Basic C# syntax, variables and expressions
- Console input and output, exception handling
- Boolean logic, branching and looping
- Single, multidimensional and jagged arrays
- Functions, parameters and arguments
- Strings, enums, and structs

Instructions

- ☐ Read chapters 1, 2, 3 and 4 from the textbook.
    - Complete the “Try It Out” exercises as is necessary for your understanding.
- ☐ Review the NumericInput, NumericFormatting, and NestedLoops example projects. Post any questions you might have in the discussion topic in DC Connect.
- ☐ Read chapters 5, 6, 7 from the textbook.
    - Complete Chapter 6: "Try It Out—Using a Delegate to Call a Function".
    - Complete Chapter 7: "Try It Out—Writing Text to the Output Window".
    - Complete any other “Try It Out” exercises as is necessary for your understanding.
- ☐ Complete the “Stucts, Methods, and Arrays” video exercise included in these instructions.
- ☐ Attach the complete solution folder for the ““Stucts, Methods, and Arrays” exercise, along with any of the “Try It Out” exercises you completed to the DC Connect drop box folder.

Tutorial 1 – Fundamentals of C#

OOP 4200 – T. MacDonald

Stucts, Methods, and Arrays Video Exercise

I recorded a video playlist here that steps you through this coding exercise. Follow along in the video, pausing as necessary. The solution you create will be part of your overall submission.

In this exercise we will create a C# console program that processes an array of percentage grades and displays a report using a number of methods.

Note the following:

Grade Range	Letter Grade	Description
90 to 100	A+	Outstanding
85 to <90	A	Exemplary
80 to <85	A-	Excellent
75 to <80	B+	Very Good
70 to <75	B	Good
65 to <70	C+	Satisfactory
60 to <65	C	Acceptable
55 to <60	D+	Conditional Pass
50 to <55	D	Conditional Pass
<50	F	Failure

Conversion Methods

We will write two methods: **PercentToLetterGrade( )** and **PercentToDescription( )**. Both should take a double argument representing a student's percentage grade and both should return a string. PercentToLetterGrade( ) returns the corresponding letter grade string. PercentToDescription( ) returns the corresponding description string. If the percentage grade is not between 0.0 and 100.0, the string returned should be **"INVALID"**.

We will define a delegate called **PercentToFeedback( )** which takes a double argument and returns a string. This will be used later as a reference to either PercentToGrade( ) or PercentToDescription( ) based on command line arguments.

Array Processing

We will create a data structure (struct) called **GradeStats**. It will include a public double to hold the average grade, and three public integers to hold the number of passing grades, the number of failures, and the number of invalid marks.

We will write one array processing method called **CalculateGradeStats( )** that will take an array of doubles representing a list of marks and a GradeStats struct to hold the average, pass count, fail count, and invalid count. We will not include invalid grades in the average. This method will return the total number of grades processed.

Tutorial 1 – Fundamentals of C#

OOP 4200 – T. MacDonald

Array Output

We will write a method called ShowGradeReport( ) that takes an array of doubles representing a list of marks and a variable of type PercentToFeedback (the delegate you created). We will call the CalculateGradeStats( ) method to do all the required processing. We will output each of the marks followed by the feedback string returned from PercentToFeedback( ). Once all the marks are displayed, we will show the overall count, average, number of passes, number of fails, and number of invalids. We will use the output examples as a guide.

Main

In Main( ), we will do the following:

- Declare an array of 5 doubles.
- Declare a delegate variable of type PercentToFeedback and initialize it with the PercentToLetter method.
- Use a loop to prompt the user for each of the 5 grades. Use exception handling to ensure only numeric input is allowed.
- In a try block, change the PercentToFeedback variable to PercentToDescription if the first command line argument is “description”.
- The following catch block is acceptable: `catch (Exception) { ;}`
- Call ShowGradeReport( ), passing the array and the PercentToFeedback variable.

Output Samples