



INetTutor.com

## Online Programming Lessons, Tutorials and Capstone Project guide

### Grade Computation in CSharp

In today's digital age, where education and evaluation have taken center stage, having a reliable tool to calculate grades efficiently is a game-changer. Whether you're a student looking to keep track of your scores or an educator seeking a convenient grading solution, you've come to the right place. In this blog post, we unveil a powerful Grade Computation Program in C# that simplifies the task of calculating final grades. Plus, we've got a special treat for you: a free download of the complete source code and a PDF tutorial to guide you every step of the way. Say goodbye to manual grade calculations and hello to precision and speed. Let's embark on this journey to demystify the art of grade computation with C#—your academic success is just a download away!

### Introduction

Grading systems are a fundamental aspect of education, and as a budding programmer, you will find that creating a Grade Computation Program is a valuable exercise in problem-solving and practical application development. Our program will not only compute the final grade but also determine whether the student has passed or failed, enhancing its real-world relevance.

Throughout this lesson, you will gain essential skills, including:

1. **Variable Handling:** Learn how to declare, initialize, and manipulate variables to store and process data effectively.
2. **Input Validation:** Explore techniques for validating user inputs, ensuring that only valid data is accepted.
3. **Arithmetic Operations:** Discover how to perform mathematical calculations, such as weighted averaging, within your C# program.
4. **Conditional Statements:** Understand how to use conditional statements (if-else) to make decisions and control the flow of your program.
5. **Real-World Relevance:** See how programming concepts are applied in practical scenarios, like educational grading systems.

By the end of this lesson, you will have developed a fully functional Grade Computation Program that can calculate final grades based on weighted components and provide immediate feedback on pass or fail status. This project will serve as a solid foundation for your future programming endeavors, demonstrating the power and versatility of the C# programming language in solving real-world problems.

### Objectives

Objectives of the lesson on creating a Grade Computation Program in C#, presented in an outcomes-based format:



INetTutor.com

## Online Programming Lessons, Tutorials and Capstone Project guide

1. Objective: Understand the importance of variables in programming.
  - Outcome: Students will recognize the significance of variables as containers for data and essential components in solving real-world problems through programming.
2. Objective: Create a C# program to calculate a student's final grade.
  - Outcome: Students will be able to design and implement a C# program that takes input for different grade components, applies weighted averaging, and calculates the final grade accurately.
3. Objective: Implement input validation to ensure data integrity.
  - Outcome: Students will develop the skill to validate user input effectively, preventing erroneous data from compromising program functionality.
4. Objective: Utilize arithmetic operations for weighted averaging.
  - Outcome: Students will demonstrate proficiency in performing arithmetic operations to calculate final grades based on given weightings.
5. Objective: Incorporate conditional statements to determine pass or fail status.
  - Outcome: Students will be able to use conditional statements to evaluate whether a student has passed or failed based on a defined passing threshold.
6. Objective: Enhance problem-solving skills through practical application.
  - Outcome: Students will gain experience in problem-solving by applying programming concepts to a practical scenario, enhancing their critical thinking and analytical abilities.
7. Objective: Develop a fully functional Grade Computation Program.
  - Outcome: Students will successfully create a Grade Computation Program that computes final grades and provides clear pass/fail feedback, demonstrating the application of programming skills to real-world situations.
8. Objective: Foster a foundation for further programming exploration.
  - Outcome: Students will establish a strong foundation in C# programming, which will empower them to tackle more complex programming challenges and pursue further learning and projects.

### Source code example

```
1. using System;
2.
3. namespace GradeComputation
4. {
5.     class Program
```



INetTutor.com

## Online Programming Lessons, Tutorials and Capstone Project guide

```
6.     {
7.         static void Main(string[] args)
8.         {
9.             // Initialize variables to store grades and weights
10.            double prelimGrade, midtermGrade, endTermGrade;
11.            double prelimWeight = 0.30, midtermWeight = 0.30, endTermWeight = 0.40;
12.
13.            Console.WriteLine("Grade Computation Program");
14.
15.            // Prompt and validate Prelim Grade
16.            prelimGrade = ValidateGrade("Prelim Grade");
17.
18.            // Prompt and validate Midterm Grade
19.            midtermGrade = ValidateGrade("Midterm Grade");
20.
21.            // Prompt and validate End Term Grade
22.            endTermGrade = ValidateGrade("End Term Grade");
23.
24.            // Calculate the final grade
25.            double finalGrade = (prelimGrade * prelimWeight) +
26.                                (midtermGrade * midtermWeight) +
27.                                (endTermGrade * endTermWeight);
28.
29.            // Display the final grade
30.            Console.WriteLine($"Your Final Grade is: {finalGrade:F2}");
31.
32.            // Check if the final grade is a pass or fail
33.            if (finalGrade >= 75)
34.            {
35.                Console.WriteLine("Congratulations! You passed.");
36.            }
37.            else
38.            {
39.                Console.WriteLine("Sorry, you failed.");
40.            }
41.
42.            Console.ReadLine(); // Keep the console window open
43.        }
44.
45.        // method to check (input validation) if the user input is numeric
46.        static double ValidateGrade(string gradeType)
47.        {
48.            double grade;
49.            bool validInput = false;
50.
51.            do
52.            {
53.                Console.Write($"Enter {gradeType}: ");
54.                if (double.TryParse(Console.ReadLine(), out grade))
55.                {
56.                    validInput = true;
57.                }
58.                else
59.                {
60.                    Console.WriteLine($"Invalid input for {gradeType}. Please enter a
valid number.");
61.                }

```



INetTutor.com

## Online Programming Lessons, Tutorials and Capstone Project guide

```
62.         } while (!validInput);
63.
64.         return grade;
65.     }
66. }
67. }
```

### Explanation

1. We begin by defining a C# program within the GradeComputation namespace.
2. Inside the Main method (the program's entry point), we declare variables to store the preliminary grade (prelimGrade), midterm grade (midtermGrade), and end-term grade (endTermGrade). We also define constants (prelimWeight, midtermWeight, and endTermWeight) to represent the weight of each grade.
3. We print a welcome message to the console to indicate that this is the "Grade Computation Program."
4. We use the ValidateGrade method to prompt the user for their preliminary, midterm, and end-term grades while validating the input. The ValidateGrade method is defined later in the code.
5. After obtaining the valid grades, we calculate the final grade by applying the respective weights to each grade component and summing them up.
6. We display the final grade using Console.WriteLine with formatting to show two decimal places.
7. Next, we check if the final grade is greater than or equal to 75 (the passing grade). If it is, we display "Congratulations! You passed." Otherwise, we display "Sorry, you failed."
8. Finally, we use Console.ReadLine() to keep the console window open until the user presses Enter, allowing them to view the results.

The ValidateGrade method is defined as a separate function to handle the input validation. It repeatedly prompts the user for a grade input until a valid numeric value is provided.


This program takes user inputs for grades, calculates a final grade based on predefined weights, and determines if the student has passed or failed based on a passing grade of 75. It's a simple example of input validation and conditional statements in C#.

### Output



INetTutor.com

## Online Programming Lessons, Tutorials and Capstone Project guide

 C:\Users\fujitsu\source\repos\GradeComputatic

```
Grade Computation Program
Enter Prelim Grade: 85
Enter Midterm Grade: 95
Enter End Term Grade: 86
Your Final Grade is: 88.40
Congratulations! You passed.
```

### Summary

In our recent blog post, we took a step into creating a Grade Computation Program in C#. We simplified variables, made naming conventions easier to understand, and broke down arithmetic operations for precise grade calculations. We ensured data accuracy through input validation and used conditional statements to determine pass/fail outcomes. By the end, readers became ready to use code to solve practical problems. We encouraged ongoing practice and exploration, emphasizing that this program is not just about code—it's a tool for academic success and programming expertise.

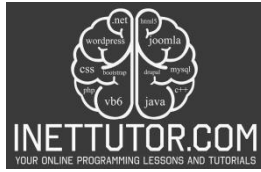
### Assessment

Assessment 1 (50 points):

*Enhancing Input Validation Assignment:* Modify the program to improve input validation. Currently, it validates that the input is a number, but it doesn't check if the entered grades are within the valid range (0-100). Enhance the program to ensure that the entered grades are valid, and if not, display an error message and prompt the user to re-enter the grade.

Assessment 2 (50 points):

*Implementing Letter Grades Assignment:* Expand the program to include letter grades based on the final numeric grade. Define a grading scale (e.g., A for 90-100, B for 80-89, etc.), and calculate and display both the numeric and letter grades for the user.



INetTutor.com

## Online Programming Lessons, Tutorials and Capstone Project guide

```
1. using System;
2.
3. namespace GradeComputation
4. {
5.     class Program
6.     {
7.         static void Main(string[] args)
8.         {
9.             // Initialize variables to store grades and weights
10.            double prelimGrade, midtermGrade, endTermGrade;
11.            double prelimWeight = 0.30, midtermWeight = 0.30, endTermWeight = 0.40;
12.
13.            Console.WriteLine("Grade Computation Program");
14.
15.            // Prompt and validate Prelim Grade
16.            prelimGrade = ValidateGrade("Prelim Grade");
17.
18.            // Prompt and validate Midterm Grade
19.            midtermGrade = ValidateGrade("Midterm Grade");
20.
21.            // Prompt and validate End Term Grade
22.            endTermGrade = ValidateGrade("End Term Grade");
23.
24.            // Calculate the final grade
25.            double finalGrade = (prelimGrade * prelimWeight) +
26.                (midtermGrade * midtermWeight) +
27.                (endTermGrade * endTermWeight);
28.
29.            // Display the final grade
30.            Console.WriteLine($"Your Final Grade is: {finalGrade:F2}");
31.
32.            // Check if the final grade is a pass or fail
33.            if (finalGrade >= 75)
34.            {
35.                Console.WriteLine("Congratulations! You passed.");
36.            }
37.            else
38.            {
39.                Console.WriteLine("Sorry, you failed.");
40.            }
41.
42.            Console.ReadLine(); // Keep the console window open
43.        }
44.
45.        // method to check (input validation) if the user input is numeric
46.        static double ValidateGrade(string gradeType)
47.        {
48.            double grade;
49.            bool validInput = false;
50.
51.            do
52.            {
53.                Console.Write($"Enter {gradeType}: ");
54.                if (double.TryParse(Console.ReadLine(), out grade))
55.                {
56.                    validInput = true;
57.                }
58.                else
59.                {
60.                    Console.WriteLine($"Invalid input for {gradeType}. Please enter a valid number.");
61.                }
62.            } while (!validInput);
63.
64.            return grade;
65.        }
66.    }
67. }
```