Total Cost Solver in CSharp

Introduction

Welcome to the lesson on creating a "Total Cost Solver" in C#! In this tutorial, we'll embark on a journey to develop a simple yet powerful C# console program that calculates the total cost of items you wish to purchase. Whether you're a budding programmer or someone looking to streamline everyday tasks, this lesson will demystify the process of building a program that crunches numbers for you.

The ability to calculate the total cost of items is more than just a programming exercise; it's a practical skill with applications in budgeting, shopping, and financial management. Imagine being able to swiftly determine the total cost of groceries, online orders, or any other purchases. With our C# "Total Cost Solver," you'll gain the skills to make these calculations with ease and accuracy.

In this lesson, we'll start from scratch and guide you through the creation of a C# console program that takes item names, prices, and quantities as input, computes the total cost, and presents it to the user. We'll break down the process into manageable steps, from setting up the project to validating user input and performing calculations. By the end, you'll not only have a working program but also the confidence to apply these principles to your programming endeavors.

Objectives

1. Learn the Fundamentals of C# Console Programming
    - Objective: By the end of this lesson, students will grasp the foundational concepts of C# console programming, including input/output operations, variable handling, and basic arithmetic calculations.
2. Develop Proficiency in User Input Handling
    - Objective: Upon completing this lesson, learners will be proficient in prompting users for input, validating that input for correctness, and providing clear error messages in case of invalid entries.
3. Create a Functional Total Cost Calculator
    - Objective: Students will be able to design and build a C# program capable of receiving item names, prices, and quantities as input, performing accurate cost calculations, and displaying the total cost, thus enhancing their practical programming skills.
4. Apply Problem-Solving Skills to Real-World Scenarios
    - Objective: Through the creation of the "Total Cost Solver" program, participants will develop problem-solving abilities applicable to various real-world situations, such as

shopping, budgeting, and financial management, promoting problem-solving skills beyond the coding realm.

These objectives are designed to guide learners through acquiring both fundamental programming knowledge and practical problem-solving skills in a real-world context.

Source code example

```
1.  using System;
2.
3.  namespace TotalCostSolver
4.  {
5.      class Program
6.      {
7.          static void Main(string[] args)
8.          {
9.              Console.WriteLine("iNetTutor.com");
10.             Console.WriteLine("Total Cost Solver");
11.             Console.WriteLine("-----------------");
12.
13.             // Ask the user for the name of the item
14.             Console.Write("Enter the name of the item: ");
15.             string itemName = Console.ReadLine();
16.
17.             // Ask the user for the price per item
18.             Console.Write($"Enter the price of {itemName} per item: ");
19.             double pricePerItem;
20.             if (!double.TryParse(Console.ReadLine(), out pricePerItem) || pricePerItem
    < 0)
21.             {
22.                 Console.WriteLine("Invalid input. Please enter a valid non-negative
    number for the price per item.");
23.                 return;
24.             }
25.
26.             // Ask the user for the quantity of items purchased
27.             Console.Write($"Enter the quantity of {itemName} purchased: ");
28.             int quantity;
29.             if (!int.TryParse(Console.ReadLine(), out quantity) || quantity < 0)
30.             {
31.                 Console.WriteLine("Invalid input. Please enter a valid non-negative
    integer for the quantity purchased.");
32.                 return;
33.             }
34.
35.             // Calculate the total cost
36.             double totalCost = pricePerItem * quantity;
37.
38.             // Display the calculated total cost
39.             Console.WriteLine($"Total cost for {quantity} {itemName}(s):
    {totalCost:C}");
40.
41.             Console.WriteLine("Press any key to exit.");
42.             Console.ReadKey();
43.         }
```

```
    }
}
```

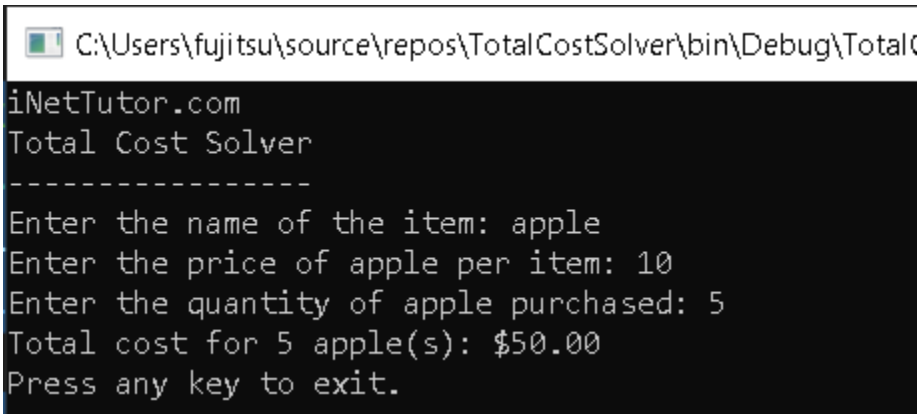
Explanation

Explanation of Key Sections:

1. Namespace and Class Declaration: The program begins with the declaration of the namespace and class.
2. Main Method: This is the entry point of the program. It displays introductory messages and prompts the user for input.
3. User Input for Item Name: The program asks the user to input the name of the item and stores it in the itemName variable.
4. User Input for Item Price: The user is prompted to enter the price per item. It uses double.TryParse to validate that the input is a valid positive number.
5. User Input for Item Quantity: Similarly, the program asks the user for the quantity of items, using int.TryParse for validation.
6. Total Cost Calculation: The total cost is calculated by multiplying the item price and quantity, and the result is stored in the totalCost variable.
7. Display Result: Finally, the program displays the calculated total cost with appropriate formatting, and waits for any key to be pressed before exiting.

Key code segments include user input handling, validation, and the arithmetic calculation to determine the total cost based on item price and quantity. These segments are essential for the program to accurately calculate and display the total cost.

Output



Summary

The "Total Cost Solver" program is a simple yet practical tool that calculates the total cost of items based on their price and quantity. It provides a straightforward solution for individuals and businesses to quickly determine their expenses when purchasing items in bulk. Understanding how to create such programs is valuable not only for personal use but also in various professional settings. Whether you're managing finances, running a small business, or working on a budgeting project, the ability to compute total costs efficiently is a fundamental skill in the world of finance and economics.

This program serves as an excellent starting point for those learning C# programming. It covers essential concepts such as user input, data validation, arithmetic calculations, and output formatting. To fully grasp these concepts and become proficient in programming, practice is key. I encourage you to experiment with this program, modify it, and explore variations. Try adding features like handling different currencies or computing discounts. By doing so, you'll solidify your programming skills and be better prepared to tackle more complex projects. Remember that programming is a journey of continuous learning, and each project you embark on brings you one step closer to mastery.

So, don't stop here! Keep exploring, keep coding, and keep expanding your knowledge. The world of programming offers endless possibilities, and your skills can unlock exciting opportunities in various fields. Happy coding!

Exercises and Challenges

1. Multiple Item Types: Modify the program to handle multiple types of items with different prices and quantities. Allow users to input the name, price, and quantity for each item, and calculate the total cost for the entire shopping list.

2. Tax Calculation: Integrate a tax rate into the program. Prompt the user to enter a tax rate, and then calculate the total cost including taxes. Ensure proper formatting for the final amount.

3. Discounts and Coupons: Implement a discount system. Allow users to enter discount percentages or coupon codes and adjust the total cost accordingly. Make sure to validate and apply the discounts accurately.

4. Currency Conversion: Extend the program's functionality to handle different currencies. Ask the user to specify the currency for each item or the entire purchase, and perform currency conversion if necessary.

Quiz

1. **What is the purpose of the "Total Cost Solver" program?**
   a) To calculate the area of a circle
   b) To compute the total cost of items based on price and quantity
   c) To convert currency from one type to another
   d) To calculate the average of a set of numbers

2. **In the program, what information does the user need to input first?**
   a) The tax rate
   b) The name of the item
   c) The quantity of items purchased
   d) The currency type

3. **If a user enters a negative price per item, what will the program display?**
   a) "Invalid input. Please enter a valid non-negative number for the price per item."
   b) "Total cost for 0 items: $0.00"
   c) "Enter the quantity of items purchased:"
   d) "Press any key to exit."

4. **What does the program do if the user enters an invalid quantity (e.g., a negative number)?**
   a) It calculates the total cost without considering the quantity.
   b) It displays the total cost as $0.00.
   c) It prompts the user to enter a valid non-negative integer for the quantity.
   d) It exits the program without any message.

5. **What is the final output of the program?**
   a) The name of the item
   b) The price per item
   c) The calculated total cost for the specified quantity of items
   d) The tax amount to be added to the total cost

```
1. using System;
2.
3. namespace TotalCostSolver
4. {
5.     class Program
6.     {
7.         static void Main(string[] args)
8.         {
9.             Console.WriteLine("iNetTutor.com");
10.            Console.WriteLine("Total Cost Solver");
11.            Console.WriteLine("-----------------");
12.
13.            // Ask the user for the name of the item
14.            Console.Write("Enter the name of the item: ");
15.            string itemName = Console.ReadLine();
16.
17.            // Ask the user for the price per item
18.            Console.Write($"Enter the price of {itemName} per item: ");
19.            double pricePerItem;
20.            if (!double.TryParse(Console.ReadLine(), out pricePerItem) || pricePerItem < 0)
21.            {
22.                Console.WriteLine("Invalid input. Please enter a valid non-negative number for the price per item.");
23.                return;
24.            }
25.
26.            // Ask the user for the quantity of items purchased
27.            Console.Write($"Enter the quantity of {itemName} purchased: ");
28.            int quantity;
29.            if (!int.TryParse(Console.ReadLine(), out quantity) || quantity < 0)
30.            {
31.                Console.WriteLine("Invalid input. Please enter a valid non-negative integer for the quantity purchased.");
32.                return;
33.            }
34.
35.            // Calculate the total cost
36.            double totalCost = pricePerItem * quantity;
37.
38.            // Display the calculated total cost
39.            Console.WriteLine($"Total cost for {quantity} {itemName}(s): {totalCost:C}");
40.
41.            Console.WriteLine("Press any key to exit.");
42.            Console.ReadKey();
43.        }
44.    }
45. }
```