# **Tutorial: Create a simple C# console app in Visual Studio (part 1 of 2)**

In this tutorial, you use Visual Studio to create and run a C# console app, and explore some features of the Visual Studio integrated development environment (IDE). This tutorial is part 1 of a two-part tutorial series.

In this tutorial, you:

* Create a Visual Studio project.
* Create a C# console app.
* Debug your app.
* Close your app.
* Inspect your complete code.

[In part 2](https://docs.microsoft.com/en-us/visualstudio/get-started/csharp/tutorial-console-part-2?view=vs-2019), you extend this app to add more projects, learn debugging tricks, and reference third-party packages.

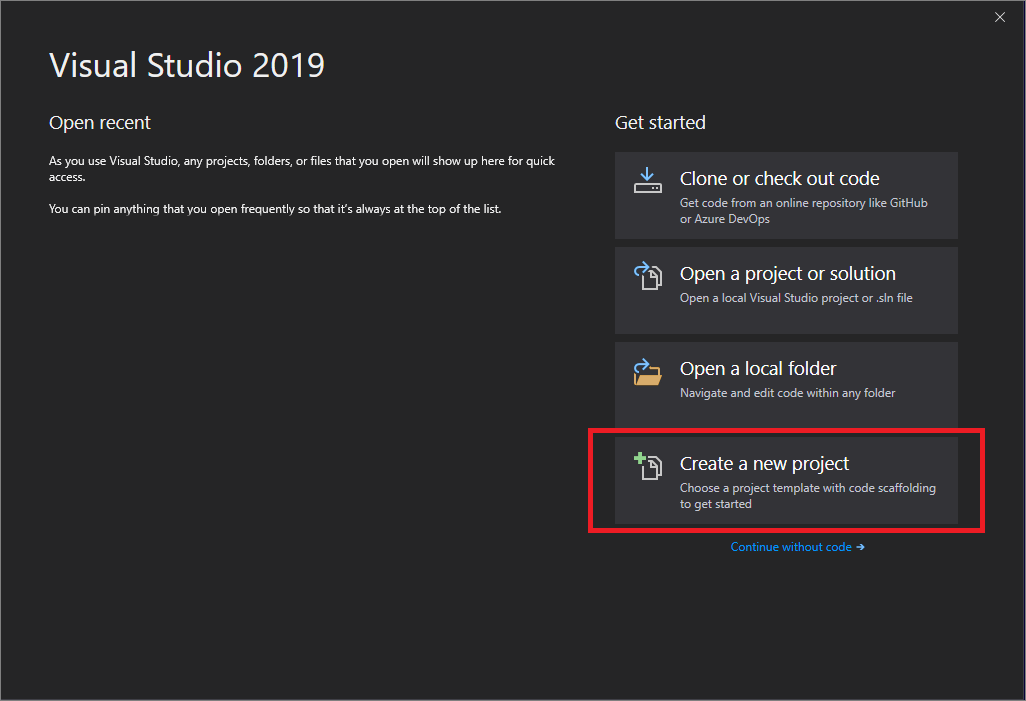
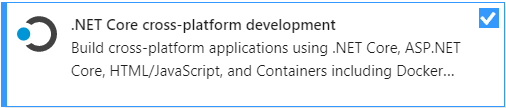
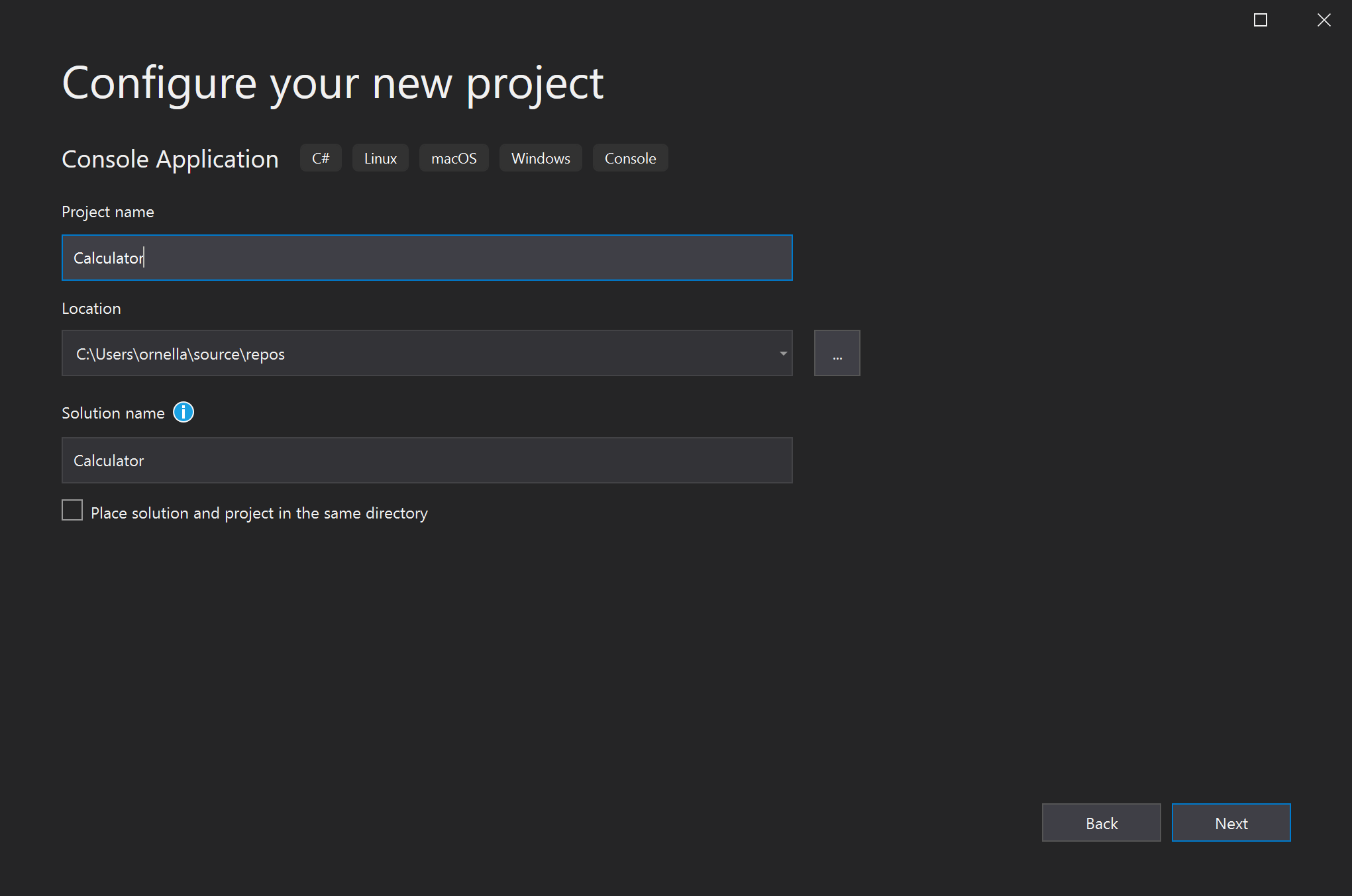
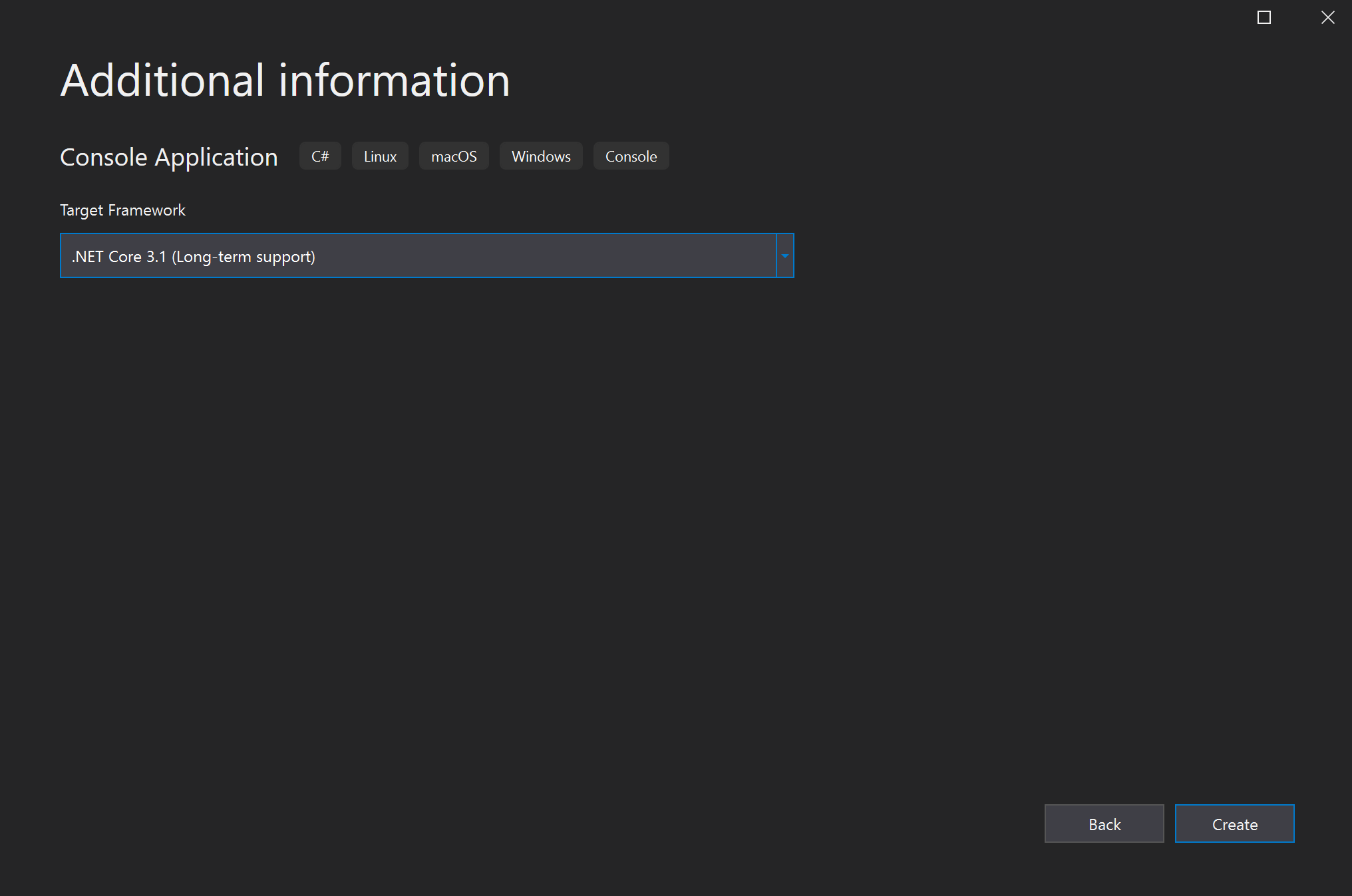
## **Prerequisites**

You must have Visual Studio installed.

If you haven't already installed Visual Studio, go to the [Visual Studio downloads](https://visualstudio.microsoft.com/downloads) page to install it for free.

## **Create a project**

To start, create a C# application project. The project type comes with all the template files you need.

1. Open Visual Studio, and choose Create a new project in the Start window.  
   
2. In the Create a new project window, choose C# from the Language list. Next, choose Windows from the Platform list and Console from the project types list.  
   After you apply the language, platform, and project type filters, choose the Console Application template, and then select Next.  
    Note  
   If you don't see the Console Application template, select Install more tools and features.  
   Screenshot that shows the Install more tools and features link.  
   Then, in the Visual Studio Installer, choose the .NET Core cross-platform development workload.  
     
   After that, choose the Modify button in the Visual Studio Installer. You might be prompted to save your work; if so, do so. Next, choose Continue to install the workload. Then, return to step 2 in this "[Create a project](https://docs.microsoft.com/en-us/visualstudio/get-started/csharp/tutorial-console?view=vs-2019#create-a-project)" procedure.
3. In the Configure your new project window, type or enter *Calculator* in the Project name box. Then, choose Next.  
   
4. In the Additional information window, .NET Core 3.1 should already be selected for your target framework. If not, select .NET Core 3.1. Then, choose Create.  
     
   Visual Studio opens your new project, which includes default "Hello World" code.

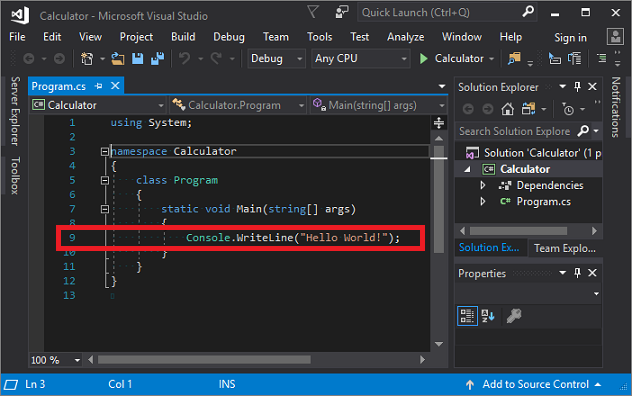
## **Create the app**

In this section, you:

* Explore some basic integer math in C#.
* Add code to create a basic calculator app.
* Debug the app to find and fix errors.
* Refine the code to make it more efficient.

### **Explore integer math**

Start with some basic integer math in C#.

1. In the code editor, delete the default "Hello World" code.  
     
   Specifically, delete the line that says, Console.WriteLine("Hello World!");.
2. In its place, type the following code:
3. C#
4. Copy

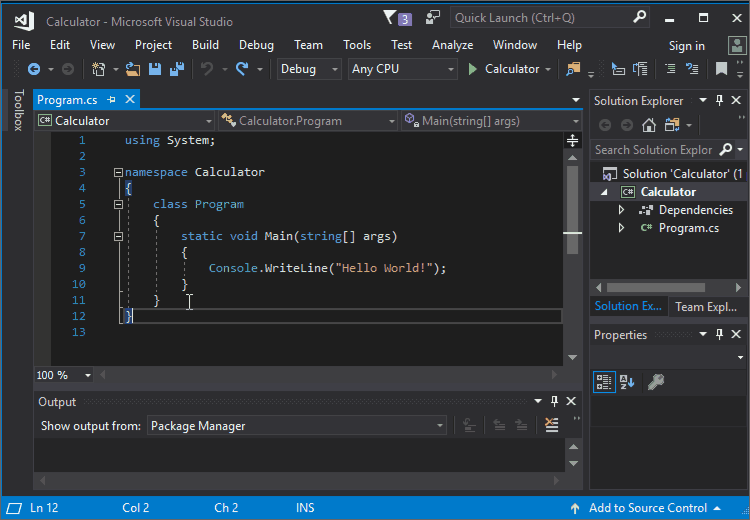
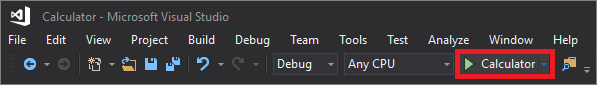
int a = 42;

int b = 119;

int c = a + b;

Console.WriteLine(c);

Console.ReadKey();

1. Notice that when you do so, the IntelliSense feature in Visual Studio offers you the option to autocomplete the entry.  
    Note  
   The following animation isn't intended to duplicate the preceding code. It's intended only to show how the autocomplete feature works.  
   
2. Choose the green Start button next to Calculator to build and run your program, or press F5.  
     
   A console window opens that reveals the sum of 42 + 119, which is 161.  
   
3. (Optional) You can change the operator to change the result. For example, you can change the + operator in the int c = a + b; line of code to - for subtraction, \* for multiplication, or / for division. Then, when you run the program, the result changes, too.
4. Close the console window.

### **Add code to create a calculator**

Continue by adding a more complex set of calculator code to your project.

1. In the code editor, replace all the code in *program.cs* with the following new code:
2. C#
3. Copy

using System;

namespace Calculator

{

class Program

{

static void Main(string[] args)

{

// Declare variables and then initialize to zero.

int num1 = 0; int num2 = 0;

// Display title as the C# console calculator app.

Console.WriteLine("Console Calculator in C#\r");

Console.WriteLine("------------------------\n");

// Ask the user to type the first number.

Console.WriteLine("Type a number, and then press Enter");

num1 = Convert.ToInt32(Console.ReadLine());

// Ask the user to type the second number.

Console.WriteLine("Type another number, and then press Enter");

num2 = Convert.ToInt32(Console.ReadLine());

// Ask the user to choose an option.

Console.WriteLine("Choose an option from the following list:");

Console.WriteLine("\ta - Add");

Console.WriteLine("\ts - Subtract");

Console.WriteLine("\tm - Multiply");

Console.WriteLine("\td - Divide");

Console.Write("Your option? ");

// Use a switch statement to do the math.

switch (Console.ReadLine())

{

case "a":

Console.WriteLine($"Your result: {num1} + {num2} = " + (num1 + num2));

break;

case "s":

Console.WriteLine($"Your result: {num1} - {num2} = " + (num1 - num2));

break;

case "m":

Console.WriteLine($"Your result: {num1} \* {num2} = " + (num1 \* num2));

break;

case "d":

Console.WriteLine($"Your result: {num1} / {num2} = " + (num1 / num2));

break;

}

// Wait for the user to respond before closing.

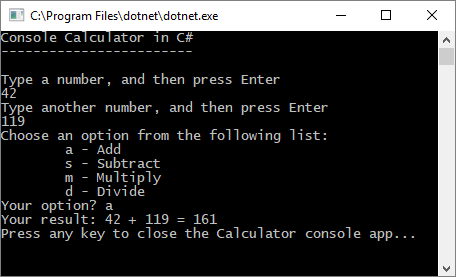
Console.Write("Press any key to close the Calculator console app...");

Console.ReadKey();

}

}

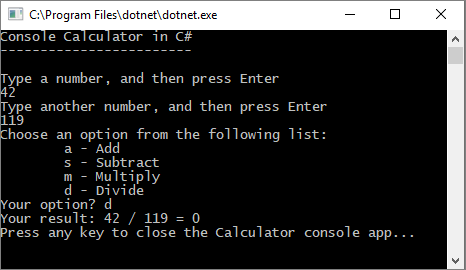
}

1. Select the Calculator button or press F5 to run your app.  
   A console window opens.
2. In the console window, follow the prompts to add the numbers 42 and 119 together.  
   Your app should look similar to the following screenshot:  
   

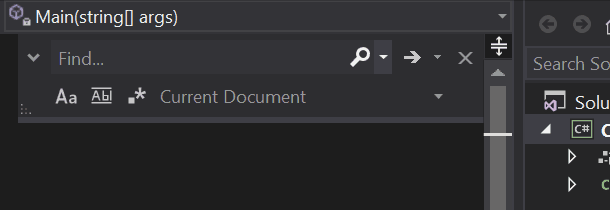
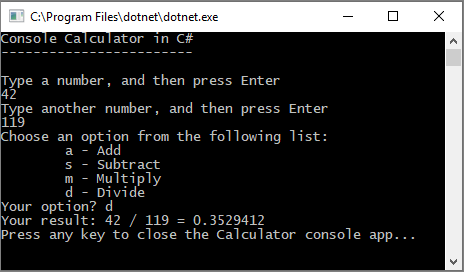
### **Add decimal functionality**

Now, tweak the code to add more functionality.

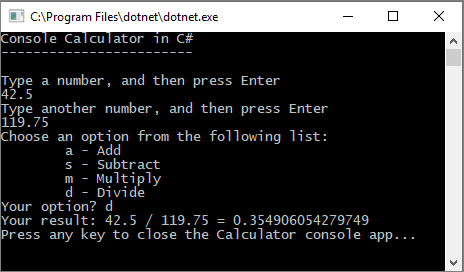
The current calculator app only accepts and returns whole numbers. For example, if you run the app and divide the number 42 by the number 119, your result is zero, which isn't exact.



To fix the code to improve precision by handling decimals:

1. From *program.cs* in the Visual Studio editor, press Ctrl+H to open the Find and Replace control.
2. Type *int* in the control, and type *float* in the Replace field.
3. Select the icons for Match case and Match whole word in the control, or press Alt+C and Alt+W.
4. Select the Replace all icon or press Alt+A to run the search and replace.  
   
5. Run your calculator app again, and divide the number 42 by the number 119.  
   The app now returns a decimal number instead of zero.  
   

Now the app can produce decimal results. Make a few more tweaks to the code so the app can calculate decimals too.

1. Use the Find and Replace control to change each instance of the float variable to double, and to change each instance of the Convert.ToInt32 method to Convert.ToDouble.
2. Run your calculator app, and divide the number 42.5 by the number 119.75.  
   The app now accepts decimal values, and returns a longer decimal numeral as its result.  
     
   In the [Revise the code](https://docs.microsoft.com/en-us/visualstudio/get-started/csharp/tutorial-console?view=vs-2019#revise-the-code) section, you reduce the number of decimal places in the results.

## **Debug the app**

You've improved your basic calculator app, but your app doesn't yet handle exceptions, such as user input errors. For example, if users try to divide by zero, or enter an unexpected character, the app might stop working, return an error, or return an unexpected nonnumeric result.

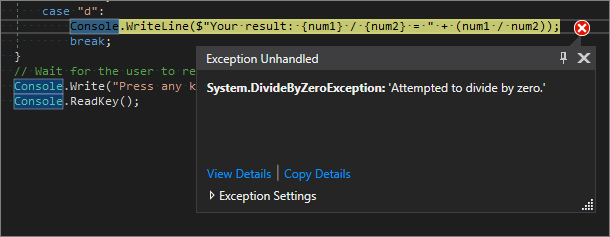
Let's walk through a few common user input errors, locate them in the debugger if they appear there, and fix them in the code.

Tip

For more information about the debugger and how it works, see [First look at the Visual Studio debugger](https://docs.microsoft.com/en-us/visualstudio/debugger/debugger-feature-tour?view=vs-2019).

### **Fix the "divide by zero" error**

If you try to divide a number by zero, the console app might freeze, and then shows you what's wrong in the code editor.



Note

Sometimes the app doesn't freeze, and the debugger doesn't show a divide-by-zero error. Instead, the app might return an unexpected nonnumeric result, such as an infinity symbol. The following code fix still applies.

To change the code to handle this error:

1. In *program.cs*, replace the code between case "d": and the comment that says // Wait for the user to respond before closing with the following code:
2. C#
3. Copy

// Ask the user to enter a non-zero divisor until they do so.

while (num2 == 0)

{

Console.WriteLine("Enter a non-zero divisor: ");

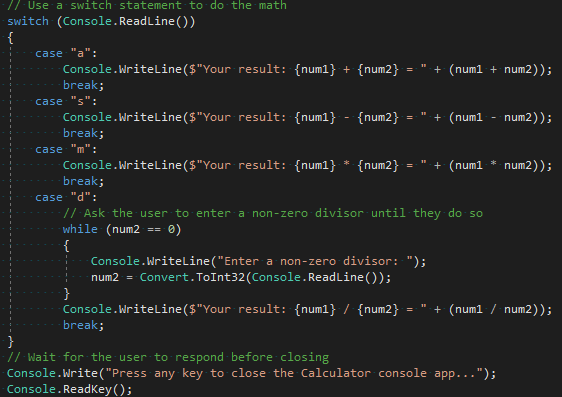
num2 = Convert.ToInt32(Console.ReadLine());

}

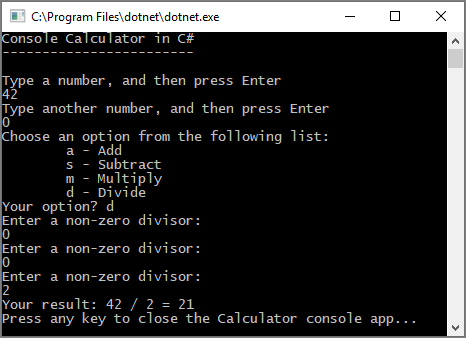
Console.WriteLine($"Your result: {num1} / {num2} = " + (num1 / num2));

break;

}

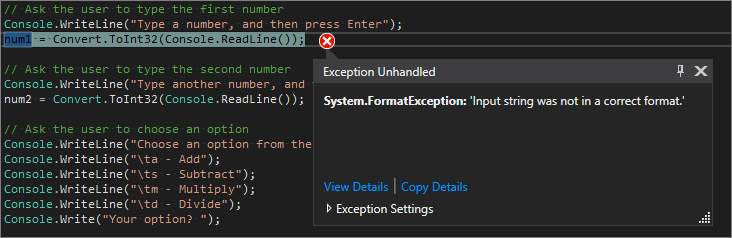
1. After you replace the code, the section with the switch statement should look similar to the following screenshot:  
   

Now, when you divide any number by zero, the app asks for another number, and keeps asking until you provide a nonzero number.



### **Fix the "format" error**

If you enter an alphabetic character when the app expects a numeric character, the app freezes. Visual Studio shows you what's wrong in the code editor.



To prevent this exception, you can refactor the code you've previously entered.

#### **Revise the code**

Rather than rely on the program class to handle all the code, you can divide your app into two classes: Calculator and Program.

The Calculator class handles the bulk of the calculation work, and the Program class handles the user interface and error-handling work.

Let's get started.

1. In *program.cs*, delete everything in the Calculator namespace between its opening and closing braces:
2. C#
3. Copy

using System;

namespace Calculator

{

}

1. Between the braces, add the following new Calculator class:
2. C#
3. Copy

class Calculator

{

public static double DoOperation(double num1, double num2, string op)

{

double result = double.NaN; // Default value is "not-a-number" if an operation, such as division, could result in an error.

// Use a switch statement to do the math.

switch (op)

{

case "a":

result = num1 + num2;

break;

case "s":

result = num1 - num2;

break;

case "m":

result = num1 \* num2;

break;

case "d":

// Ask the user to enter a non-zero divisor.

if (num2 != 0)

{

result = num1 / num2;

}

break;

// Return text for an incorrect option entry.

default:

break;

}

return result;

}

}

1. Also add a new Program class, as follows:
2. C#
3. Copy

class Program

{

static void Main(string[] args)

{

bool endApp = false;

// Display title as the C# console calculator app.

Console.WriteLine("Console Calculator in C#\r");

Console.WriteLine("------------------------\n");

while (!endApp)

{

// Declare variables and set to empty.

string numInput1 = "";

string numInput2 = "";

double result = 0;

// Ask the user to type the first number.

Console.Write("Type a number, and then press Enter: ");

numInput1 = Console.ReadLine();

double cleanNum1 = 0;

while (!double.TryParse(numInput1, out cleanNum1))

{

Console.Write("This is not valid input. Please enter an integer value: ");

numInput1 = Console.ReadLine();

}

// Ask the user to type the second number.

Console.Write("Type another number, and then press Enter: ");

numInput2 = Console.ReadLine();

double cleanNum2 = 0;

while (!double.TryParse(numInput2, out cleanNum2))

{

Console.Write("This is not valid input. Please enter an integer value: ");

numInput2 = Console.ReadLine();

}

// Ask the user to choose an operator.

Console.WriteLine("Choose an operator from the following list:");

Console.WriteLine("\ta - Add");

Console.WriteLine("\ts - Subtract");

Console.WriteLine("\tm - Multiply");

Console.WriteLine("\td - Divide");

Console.Write("Your option? ");

string op = Console.ReadLine();

try

{

result = Calculator.DoOperation(cleanNum1, cleanNum2, op);

if (double.IsNaN(result))

{

Console.WriteLine("This operation will result in a mathematical error.\n");

}

else Console.WriteLine("Your result: {0:0.##}\n", result);

}

catch (Exception e)

{

Console.WriteLine("Oh no! An exception occurred trying to do the math.\n - Details: " + e.Message);

}

Console.WriteLine("------------------------\n");

// Wait for the user to respond before closing.

Console.Write("Press 'n' and Enter to close the app, or press any other key and Enter to continue: ");

if (Console.ReadLine() == "n") endApp = true;

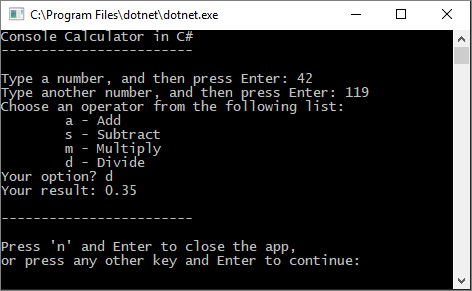
Console.WriteLine("\n"); // Friendly linespacing.

}

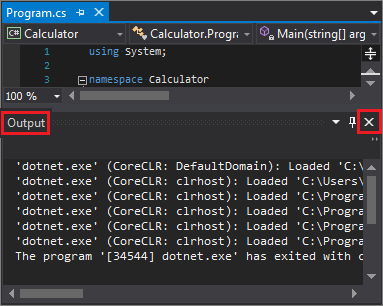
return;

}

}

1. Select the Calculator button or press F5 to run your app.
2. Follow the prompts and divide the number 42 by the number 119. Your results should look similar to the following screenshot:  
     
   You can now enter more equations until you choose to close the console app. There are also fewer decimal places in the results. And if you enter an incorrect character, you get an appropriate error response.

## **Close the app**

1. If you haven't already done so, close the Calculator app.
2. Close the Output pane in Visual Studio.  
   
3. In Visual Studio, press Ctrl+S to save your app.

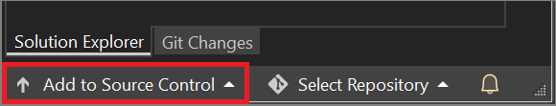
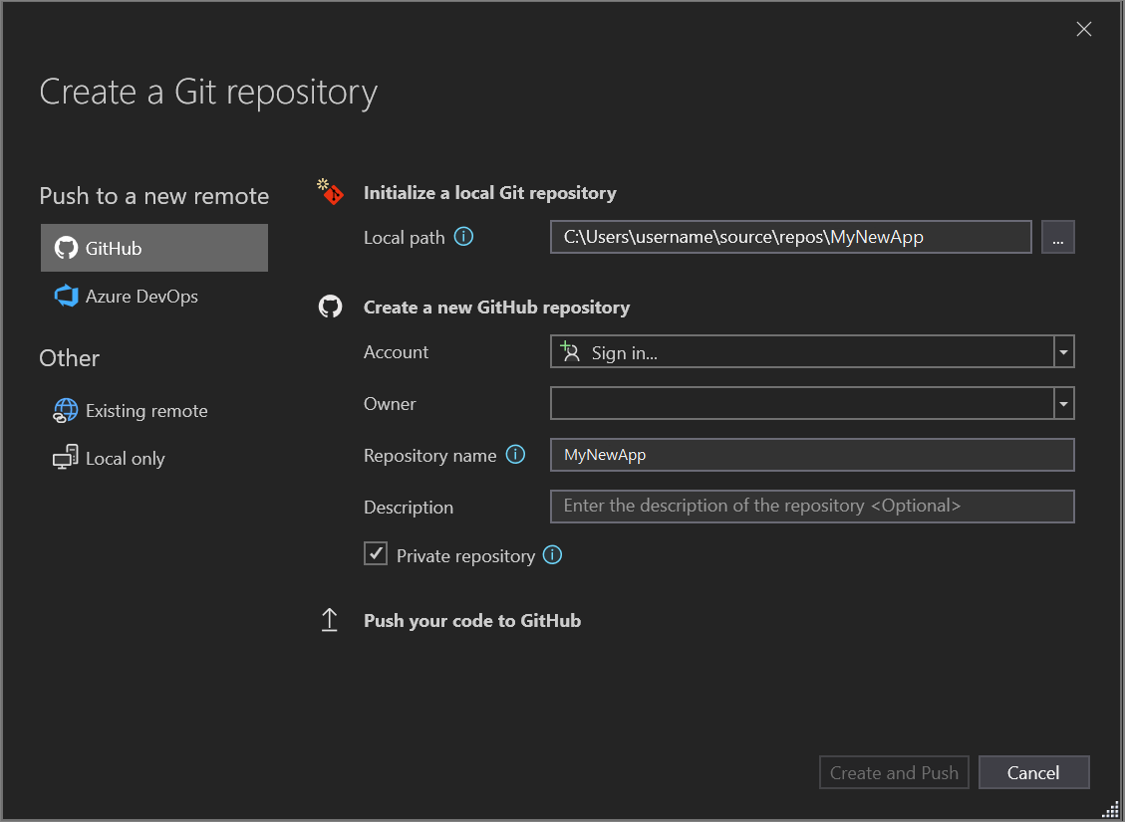
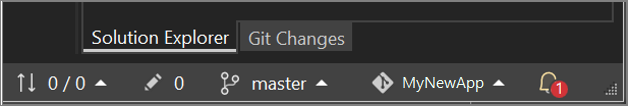
## **Add Git source control**

Now that you've created an app, you might want to add it to a Git repository. We've got you covered. Visual Studio makes that process easy with Git tools you can use directly from the IDE.

Tip

Git is the most widely used modern version control system, so whether you're a professional developer or you're learning how to code, Git can be very useful. If you're new to Git, the <https://git-scm.com/> website is a good place to start. There, you can find cheat sheets, a popular online book, and Git Basics videos.

To associate your code with Git, you start by creating a new Git repository where your code is located. Here's how:

1. In the status bar at the bottom-right corner of Visual Studio, select Add to Source Control, and then select Git.  
   
2. In the Create a Git repository dialog box, sign in to GitHub.  
     
   The repository name auto-populates based on your folder location. By default, your new repository is private, which means you're the only one who can access it.  
    Tip  
   Whether your repository is public or private, it's best to have a remote backup of your code stored securely on GitHub. Even if you aren't working with a team, a remote repository makes your code available to you from any computer.
3. Select Create and Push.  
   After you create your repository, you see status details in the status bar.  
     
   The first icon with the arrows shows how many outgoing/incoming commits are in your current branch. You can use this icon to pull any incoming commits or push any outgoing commits. You can also choose to view these commits first. To do so, select the icon, and then select View Outgoing/Incoming.  
   The second icon with the pencil shows the number of uncommitted changes to your code. You can select this icon to view those changes in the Git Changes window.

To learn more about how to use Git with your app, see the [Visual Studio version control documentation](https://docs.microsoft.com/en-us/visualstudio/version-control/?view=vs-2019).

## **Review: Code complete**

In this tutorial, you made many changes to the calculator app. The app now handles computing resources more efficiently, and handles most user input errors.

Here's the complete code, all in one place:

C#

Copy

using System;

namespace Calculator

{

class Calculator

{

public static double DoOperation(double num1, double num2, string op)

{

double result = double.NaN; // Default value is "not-a-number" which we use if an operation, such as division, could result in an error.

// Use a switch statement to do the math.

switch (op)

{

case "a":

result = num1 + num2;

break;

case "s":

result = num1 - num2;

break;

case "m":

result = num1 \* num2;

break;

case "d":

// Ask the user to enter a non-zero divisor.

if (num2 != 0)

{

result = num1 / num2;

}

break;

// Return text for an incorrect option entry.

default:

break;

}

return result;

}

}

class Program

{

static void Main(string[] args)

{

bool endApp = false;

// Display title as the C# console calculator app.

Console.WriteLine("Console Calculator in C#\r");

Console.WriteLine("------------------------\n");

while (!endApp)

{

// Declare variables and set to empty.

string numInput1 = "";

string numInput2 = "";

double result = 0;

// Ask the user to type the first number.

Console.Write("Type a number, and then press Enter: ");

numInput1 = Console.ReadLine();

double cleanNum1 = 0;

while (!double.TryParse(numInput1, out cleanNum1))

{

Console.Write("This is not valid input. Please enter an integer value: ");

numInput1 = Console.ReadLine();

}

// Ask the user to type the second number.

Console.Write("Type another number, and then press Enter: ");

numInput2 = Console.ReadLine();

double cleanNum2 = 0;

while (!double.TryParse(numInput2, out cleanNum2))

{

Console.Write("This is not valid input. Please enter an integer value: ");

numInput2 = Console.ReadLine();

}

// Ask the user to choose an operator.

Console.WriteLine("Choose an operator from the following list:");

Console.WriteLine("\ta - Add");

Console.WriteLine("\ts - Subtract");

Console.WriteLine("\tm - Multiply");

Console.WriteLine("\td - Divide");

Console.Write("Your option? ");

string op = Console.ReadLine();

try

{

result = Calculator.DoOperation(cleanNum1, cleanNum2, op);

if (double.IsNaN(result))

{

Console.WriteLine("This operation will result in a mathematical error.\n");

}

else Console.WriteLine("Your result: {0:0.##}\n", result);

}

catch (Exception e)

{

Console.WriteLine("Oh no! An exception occurred trying to do the math.\n - Details: " + e.Message);

}

Console.WriteLine("------------------------\n");

// Wait for the user to respond before closing.

Console.Write("Press 'n' and Enter to close the app, or press any other key and Enter to continue: ");

if (Console.ReadLine() == "n") endApp = true;

Console.WriteLine("\n"); // Friendly linespacing.

}

return;

}

}

}