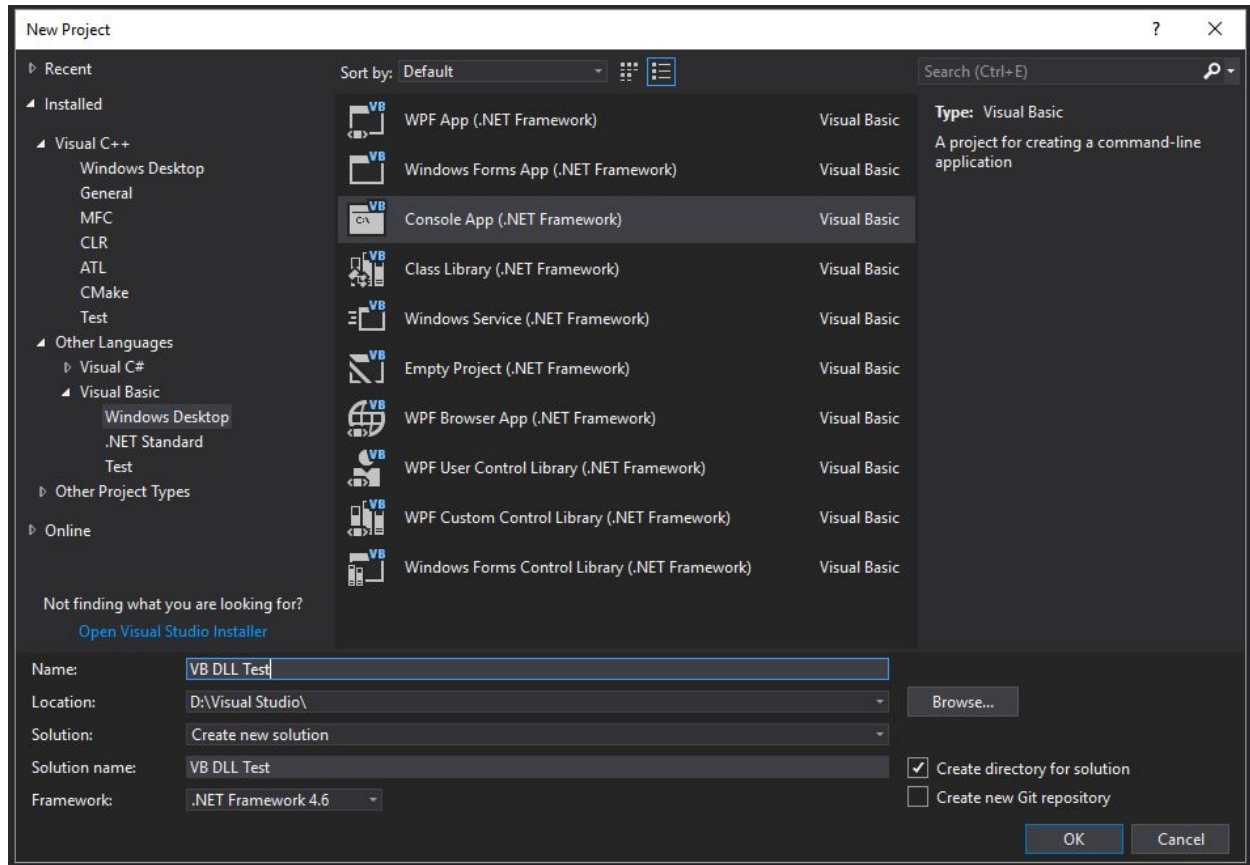


This tutorial was made using Microsoft Visual Studio 2017 in Visual Basic, C++ and C#. For each I will use one kind of project. Being that the kind of project may limit or give more features to the language, therefore these are also the currently tested and confirmed working with the DLL:

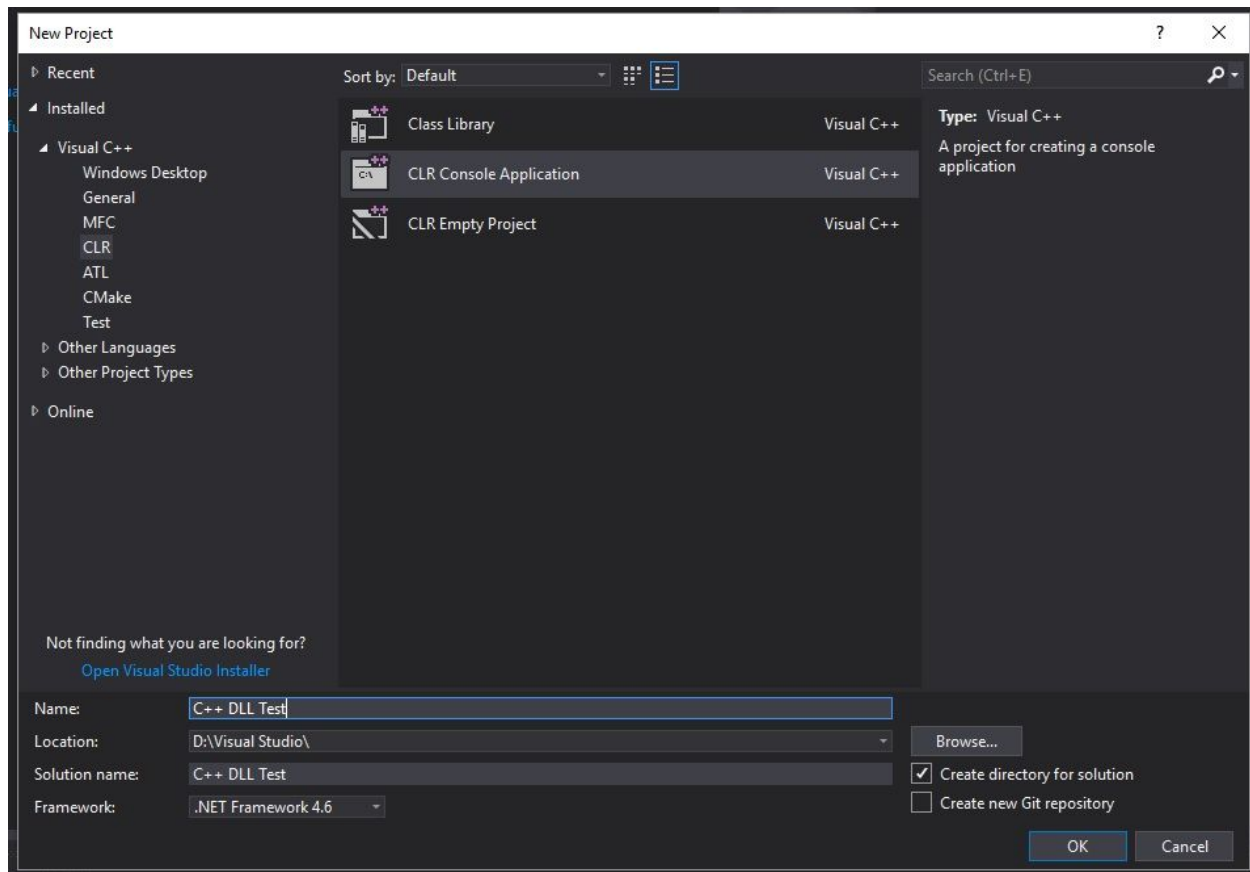
Step 1 - Creating Project:

From the top menu go to File -> New -> Project...

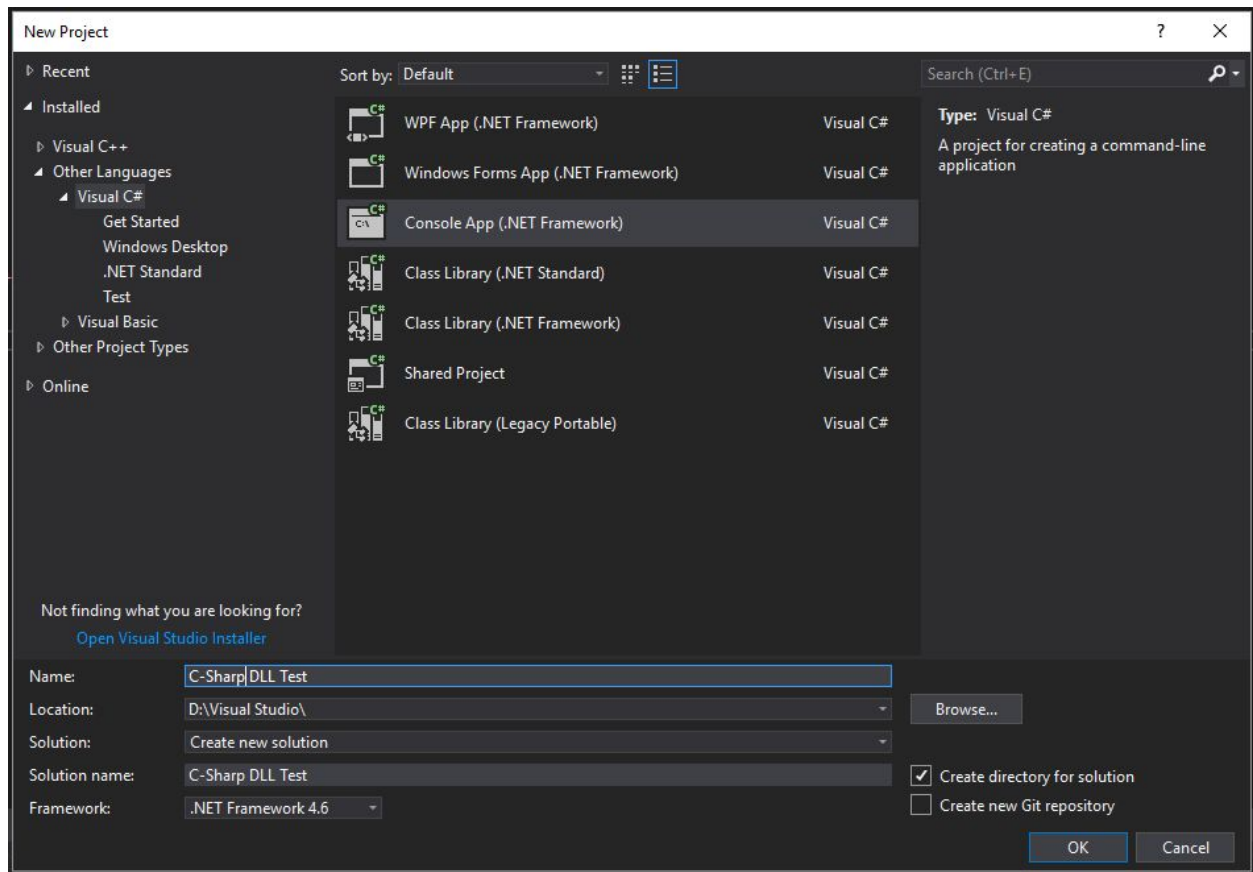
Visual Basic Console App (.NET Framework):



C++ CLR Console Application:

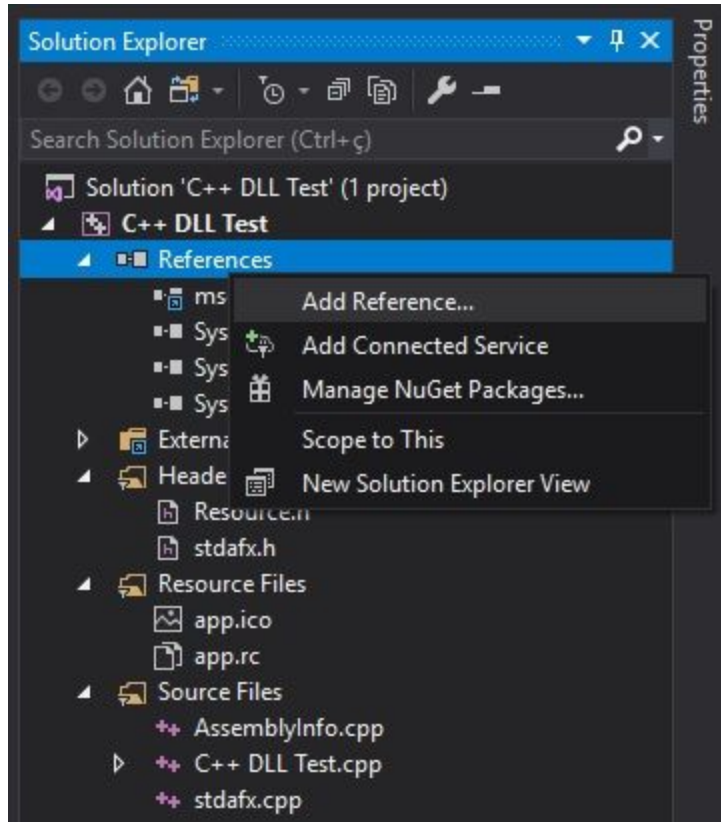


C# Console App (.NET Framework):

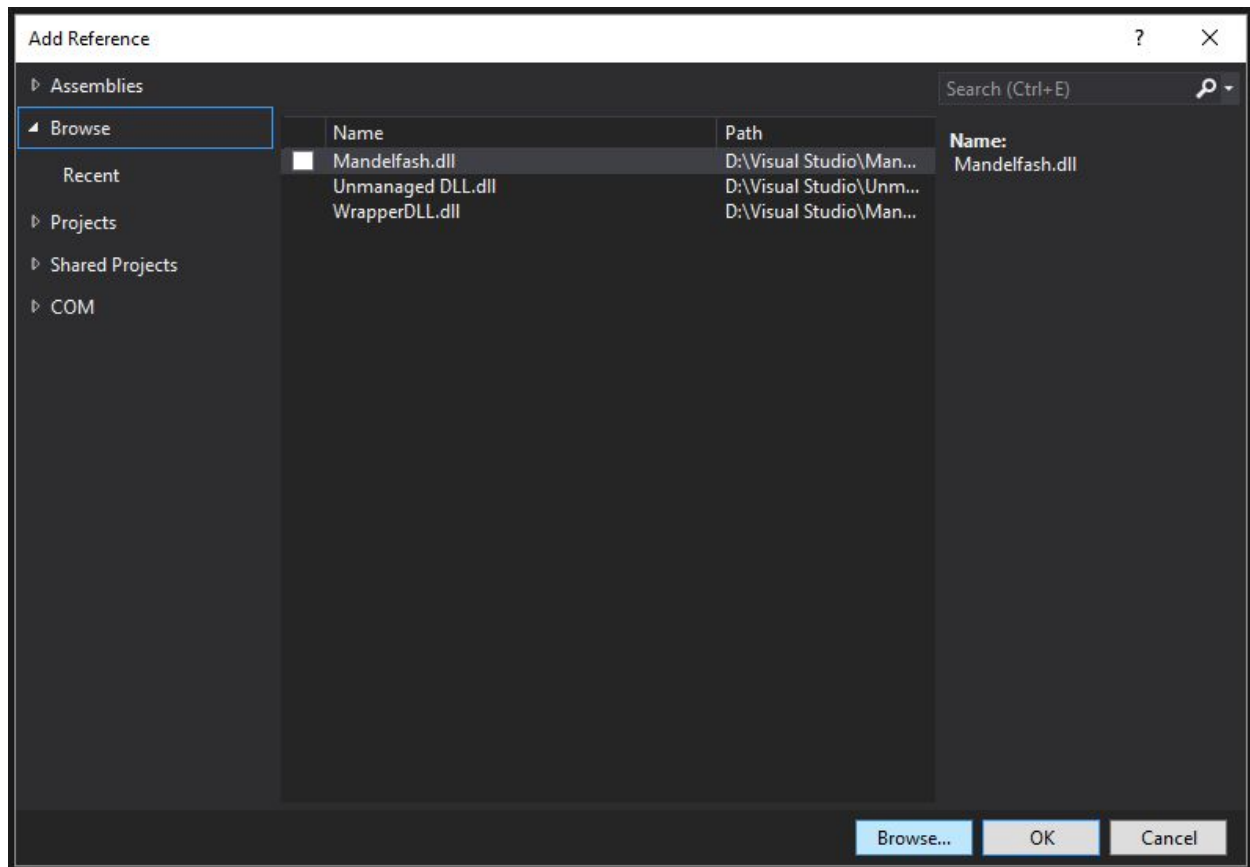


Step 2 - Adding a reference to the DLL:

This is the same for every language. Open the solution explorer and go to the References and right-click. Click on Add Reference...

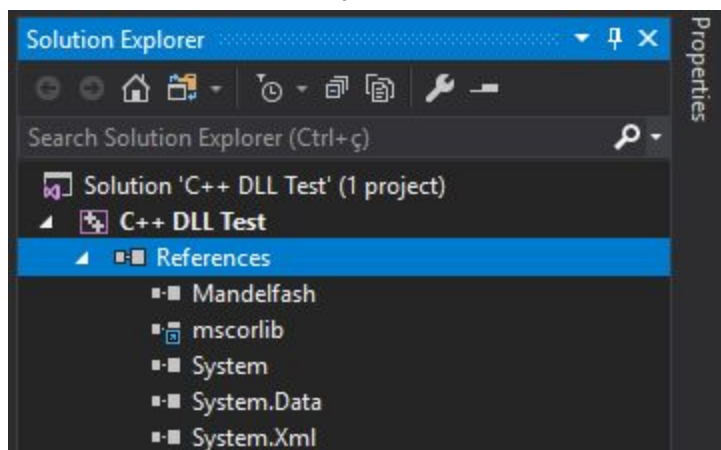


On the opened menu select Brose and click on the Browse... button on the corner. This will open a file select standard window. Find the folder and select the Mandelfash.dll file.



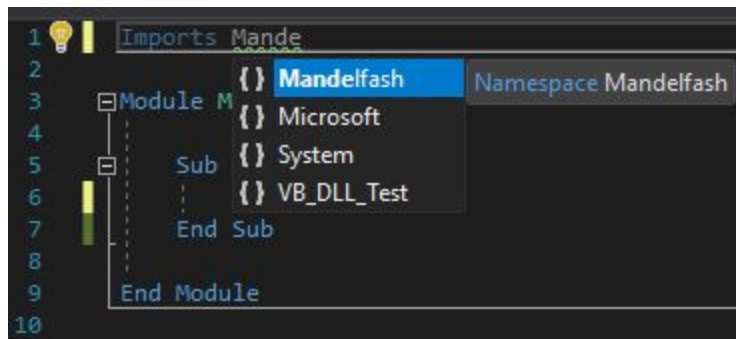
After added the files should appear on the above screen. Just make sure it is checked and click Ok to add it.

The file should appear only as Mandelfash in the list of references.



If the inclusion has been done correctly the file should now be a possible inclusion suggestion for your code. The inclusion should look like this for each language:

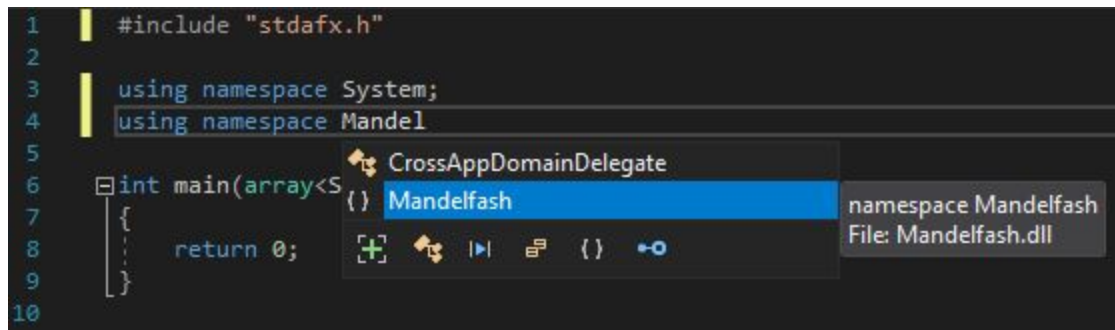
VB:



```
1 Imports Mandelfash
2
3 Module M
4     {} Microsoft
5     Sub
6         {} System
7         {} VB_DLL_Test
8     End Sub
9 End Module
10
```

The screenshot shows a Visual Basic code editor with a dropdown menu open for the 'Imports' statement. The menu lists 'Mandelfash' (highlighted), 'Microsoft', 'System', and 'VB_DLL_Test'. The code in the background is a module named 'M' with a sub 'Sub' and an 'End Module' statement.

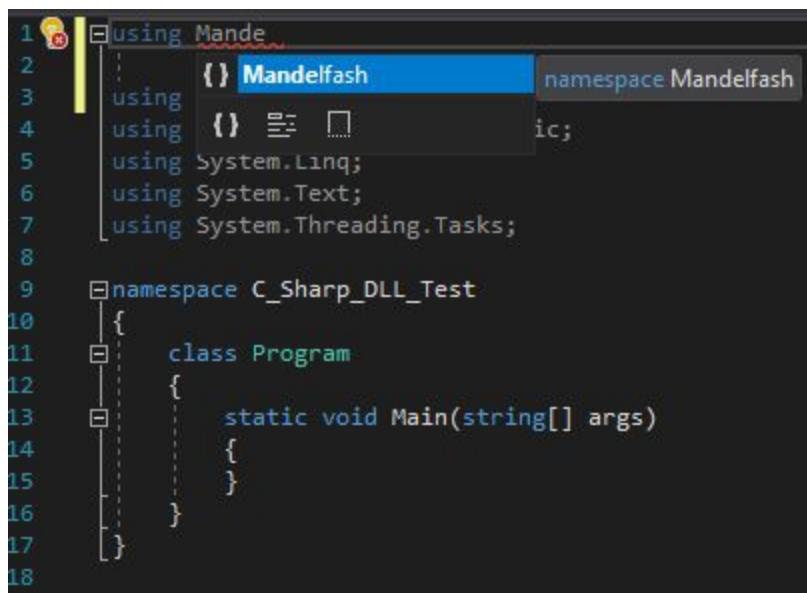
C++:



```
1 #include "stdafx.h"
2
3 using namespace System;
4 using namespace Mandelfash
5
6 int main(array<S
7 {
8     return 0;
9 }
10
```

The screenshot shows a C++ code editor with a dropdown menu open for the 'using namespace' statement. The menu lists 'CrossAppDomainDelegate' and 'Mandelfash' (highlighted). The code in the background is a C++ program with a 'main' function and a 'return 0;' statement.

C#:

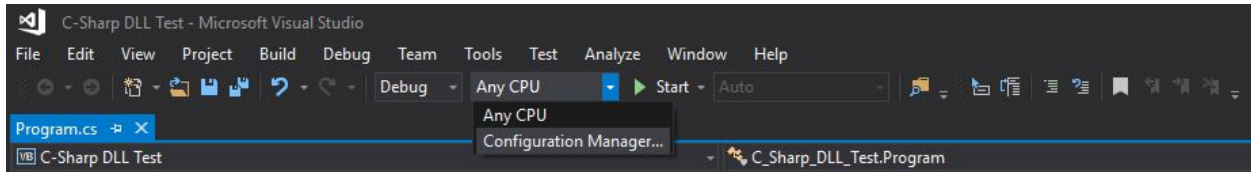


```
1 using Mandelfash
2
3 using
4 using
5 using System.Linq;
6 using System.Text;
7 using System.Threading.Tasks;
8
9 namespace C_Sharp_DLL_Test
10 {
11     class Program
12     {
13         static void Main(string[] args)
14         {
15         }
16     }
17 }
18
```

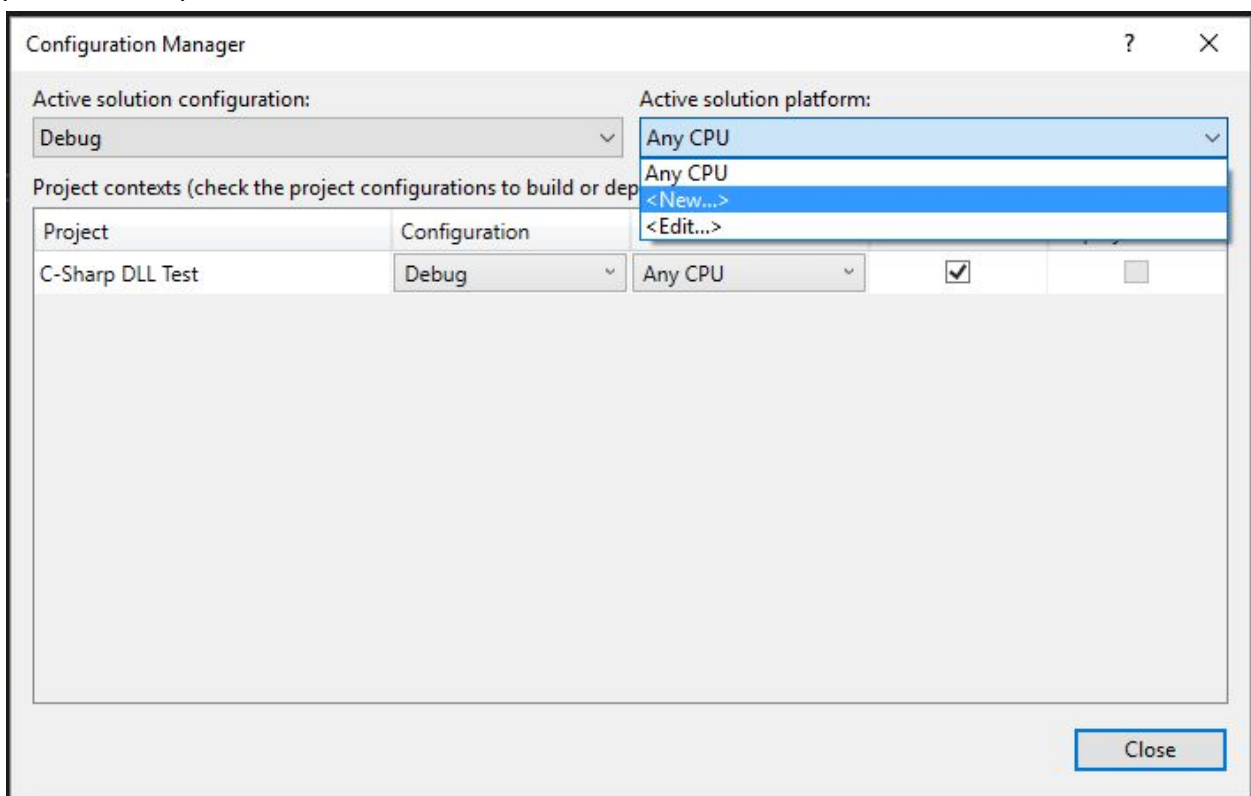
The screenshot shows a C# code editor with a dropdown menu open for the 'using' statement. The menu lists 'Mandelfash' (highlighted). The code in the background is a C# program with a 'Main' method and a 'return 0;' statement.

Step 3 - Build Configuration:

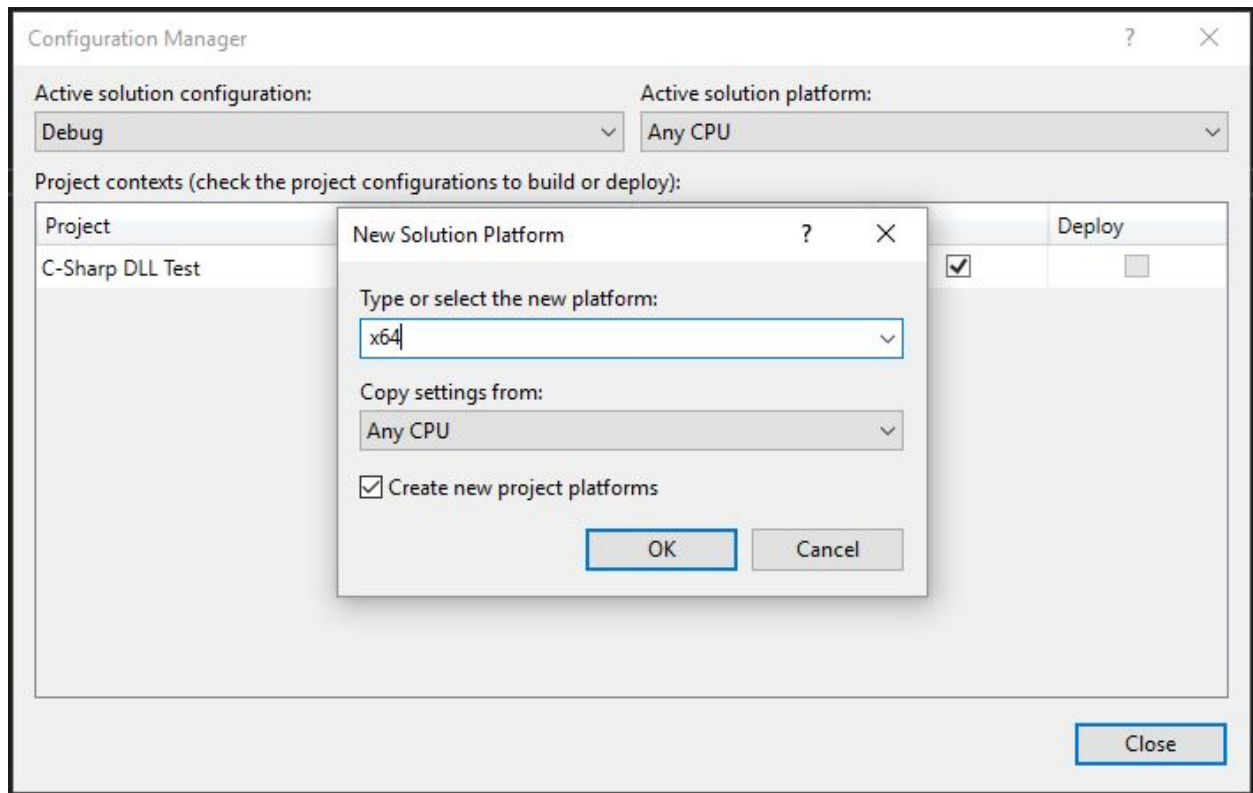
The DLL need the code to be built as x64. While in C++ x64 will most likely be a default option, in both VB and C# you will probably need to add it. On the menu below select where it says “Any CPU” or “x86” and click on “Configuration Manager...”.



If the option is unavailable by default you will need to click on <New...> on the Active solution platform's drop down menu.



Just select x64, click Ok and we are good to go.



After closing this menu make sure the build option is set to x64:



Step 4 - First Execution:

Here it is three code samples. One for each language, feel free to play around with the methods explained in the documentation to see what works best for you.

VB:

```
1 Imports Mandelfash
2
3 Module Module1
4
5     Sub Main()
6         Dim calc As New ColorCalculator()
7         calc.startCalculator("1", "1", "1")
8
9         Dim results As List(Of List(Of Integer))
10
11         results = calc.getResults()
12
13         Console.WriteLine("The RGB color values is Red: " + results.Item(0).Item(0).ToString() + ", Green: " + results.Item(0).Item(1).ToString() + ", Blue: " + results.Item(0).Item(2).ToString())
14         Console.WriteLine("Press any key to close...")
15         Console.ReadKey()
16     End Sub
17 End Module
18
19
20
```

C++:

```
1 #include "stdafx.h"
2
3 using namespace System;
4 using namespace System::Collections::Generic;
5 using namespace Mandelfash;
6
7 int main(array<System::String ^> ^args)
8 {
9     ColorCalculator^ calc = gcnew ColorCalculator();
10
11     calc->startCalculator("1", "1", "1");
12
13     List<List<int>>^ results = calc->getResults();
14
15     List<int>^ firstEntry = results[0];
16
17     Console::WriteLine("The RGB color values is Red: " + firstEntry[0] + ", Green: " + firstEntry[1] + ", Blue: " + firstEntry[2]);
18     Console::WriteLine("Press any key to close...");
19     Console::ReadKey();
20     return 0;
21 }
22
```

C#:

```
using Mandelfash;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace C_Sharp_DLL_Test
{
    class Program
    {
        static void Main(string[] args)
        {
            ColorCalculator calc = new ColorCalculator();

            calc.startCalculator("1", "1", "1");

            List<List<int>> results = calc.getResults();

            Console.WriteLine("The RGB color values is Red: " + results[0][0] + ", Green: " + results[0][1] + ", Blue: " + results[0][2]);
            Console.WriteLine("Press any key to close...");
            Console.ReadKey();
        }
    }
}
```

Step 5 - First Execution:

Whichever language you used, if you did exactly as above here will be the resulting console screen when you click “Start”, “Local Windows Debugger” or any similar:

```
The RGB color values is Red: 0, Green: 0, Blue: 0  
Press any key to close...
```