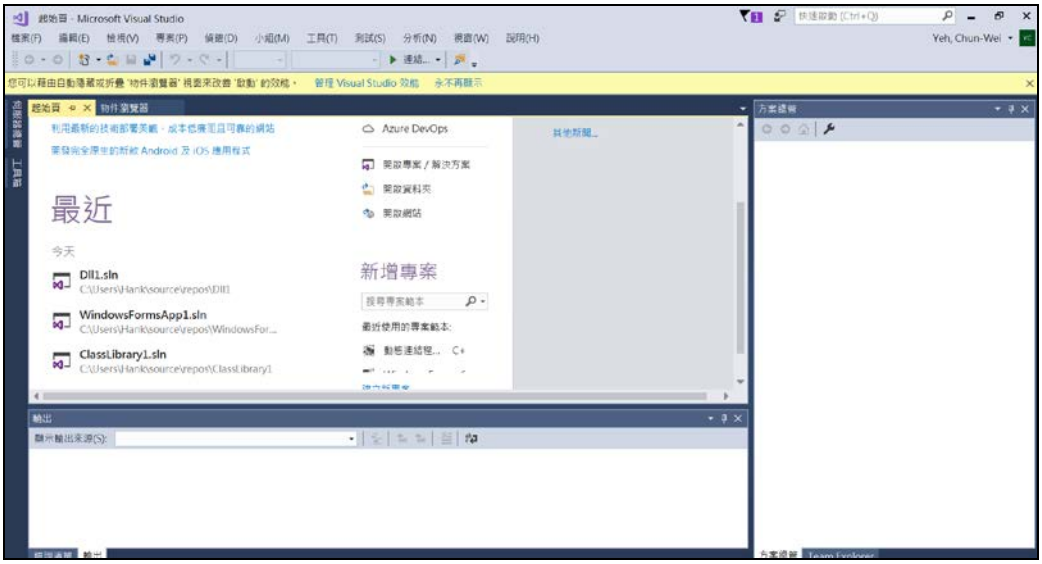
國立臺北科技大學自動化所 嵌入式工業機器視覺

Creating Dynamic Libraries (.dll) on Windows*

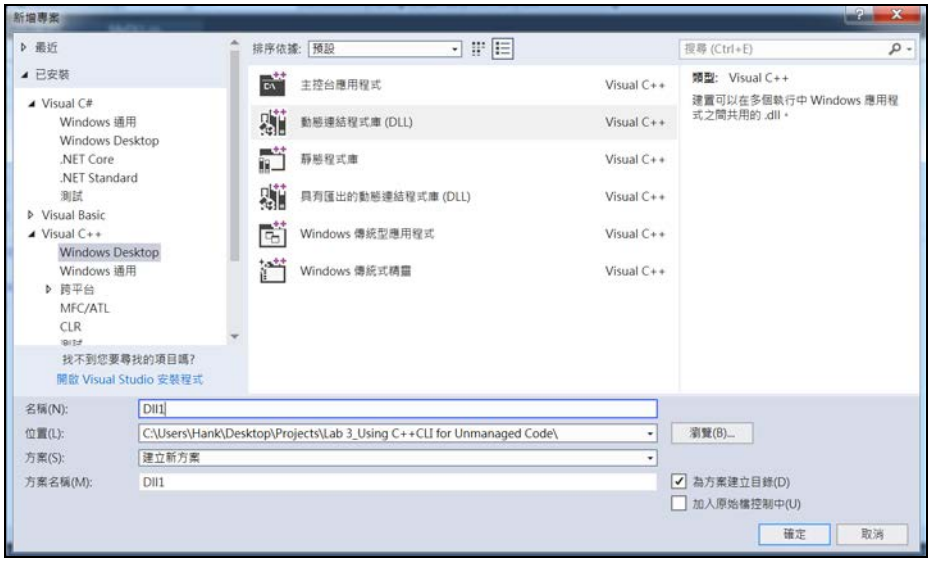
[Using C++/CLI for Unmanaged Code]

The following steps describe how to create a dynamic library on Windows. These steps are for Microsoft Visual Studio 2017, although the steps are similar for other versions of Visual Studio.

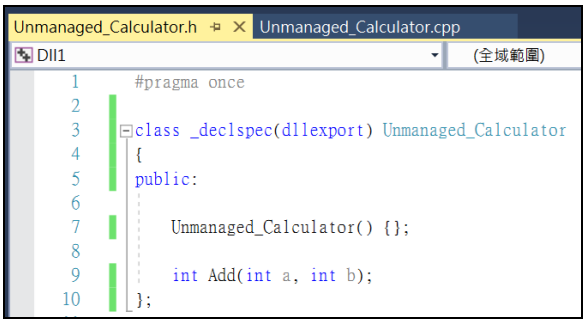
1. Select the menu File > New > Project



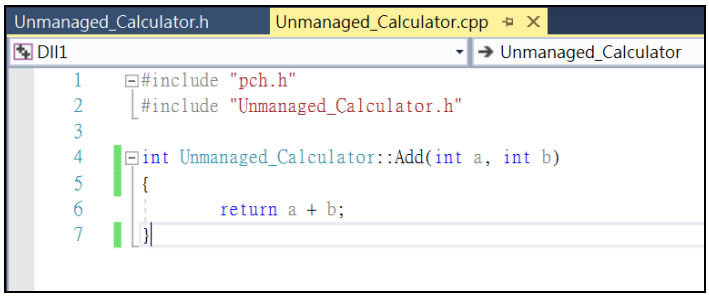
2. Select the Visual C++ -> Windows Desktop and the DLL option, and then put the project name and path.



3. If you want a function to be callable from a DLL on Windows, you must explicitly mark its declaration. The following code (Unmanaged_Calculator.h) provides a simple demonstration of this.

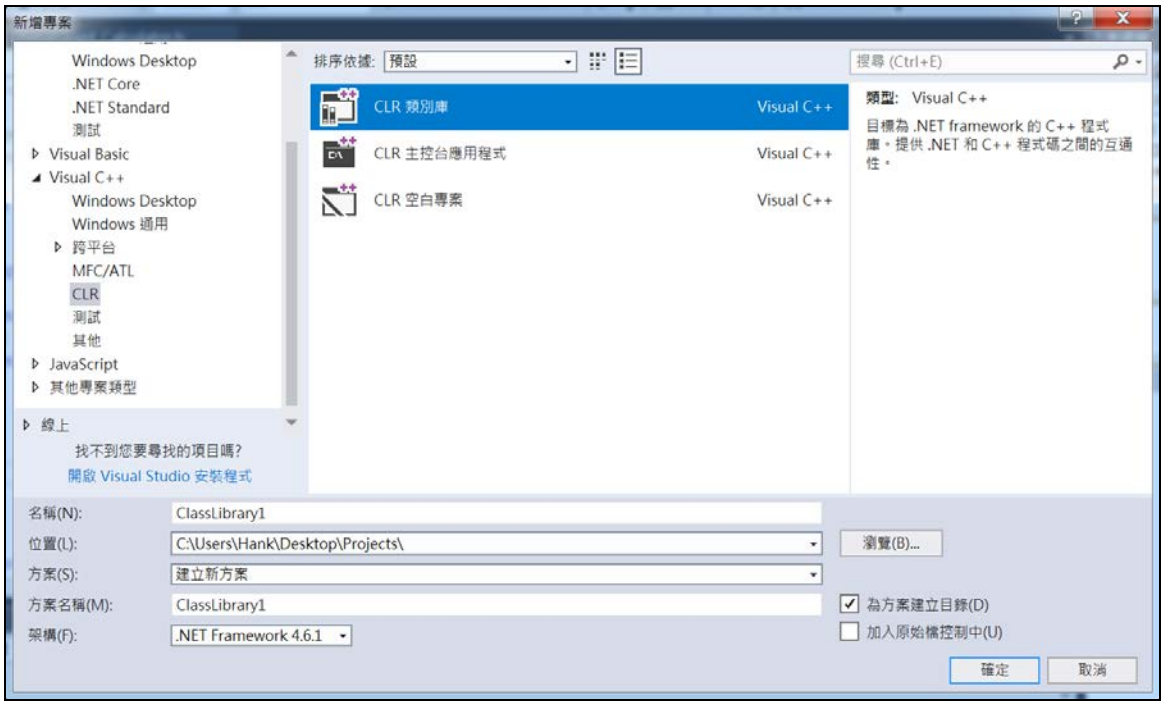


4. You can then add new or existing source files (Unmanaged_Calculator.cpp) to your project under the Source Files folder in the left-hand pane.



5. Build → Build Project (or Build Dll1), then Visual Studio will generate a .dll file and an associated .lib import file.

6. Select the Visual C++ -> CLR and the CLR Classes option, and then put the project name and path.

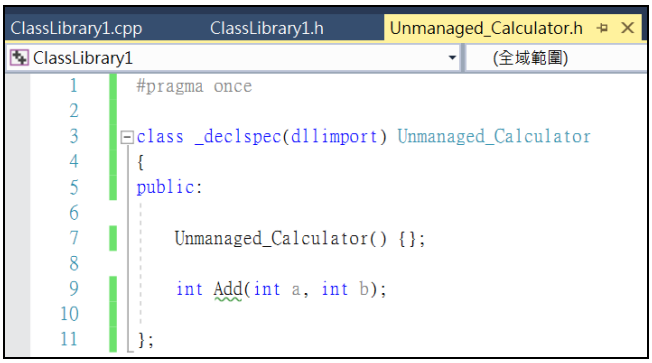


7. If you want a function to be callable from a DLL on managed code (.NET), you must explicitly mark its declaration. The following code (ClassLibrary1.h) provides a simple demonstration of this.

ClassLibrary1.h

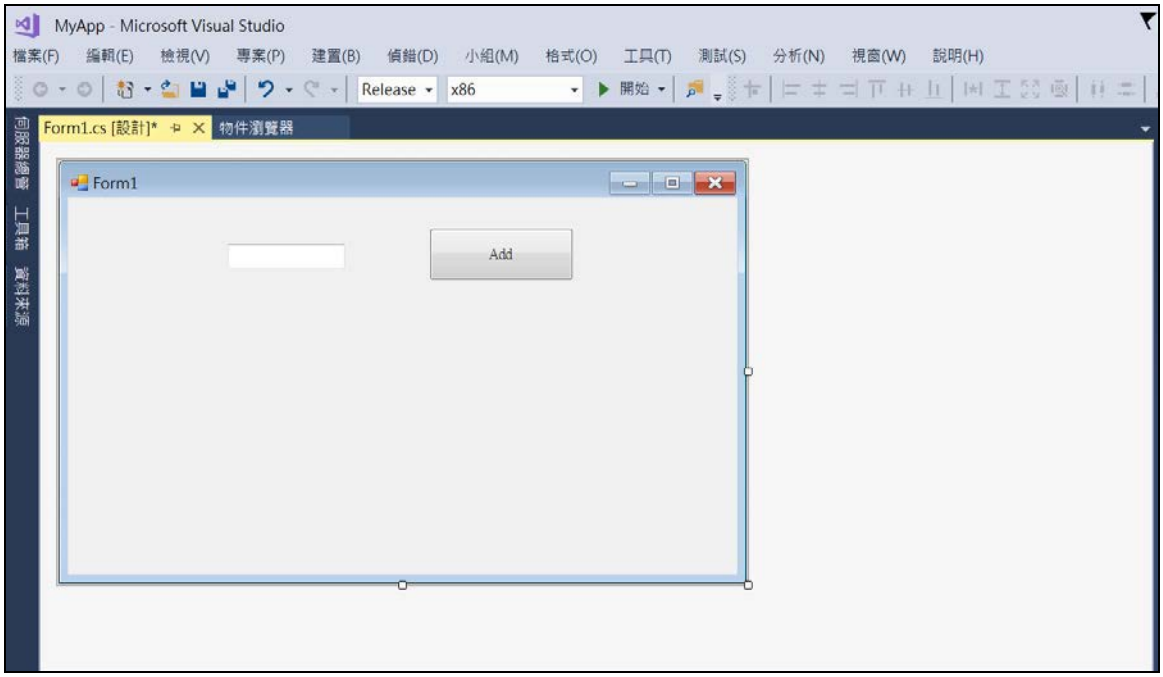
```
#include "Unmanaged_Calculator.h"

#pragma once
#pragma comment(lib, "Dll1.lib")

using namespace System;
namespace ClassLibrary1
{
    public ref class ManagedClass
    {
    public:
        // Allocate the native object on the C++ Heap via a constructor
        ManagedClass() : m_Impl(new Unmanaged_Calculator) {}
        // Deallocate the native object on a destructor
        ~ManagedClass()
        {
            delete m_Impl;
        }
    protected:
        // Deallocate the native object on the finalizer just in case no destructor is called
        !ManagedClass()
        {
            delete m_Impl;
        }
    public:
        int Add(int a, int b)
        {
            return (m_Impl->Add (a, b));
        }

    private:
        Unmanaged_Calculator    *m_Impl;
    };
}
```

8. You can then add new or existing header files (Unmanaged_ Calculator.h) to your project.

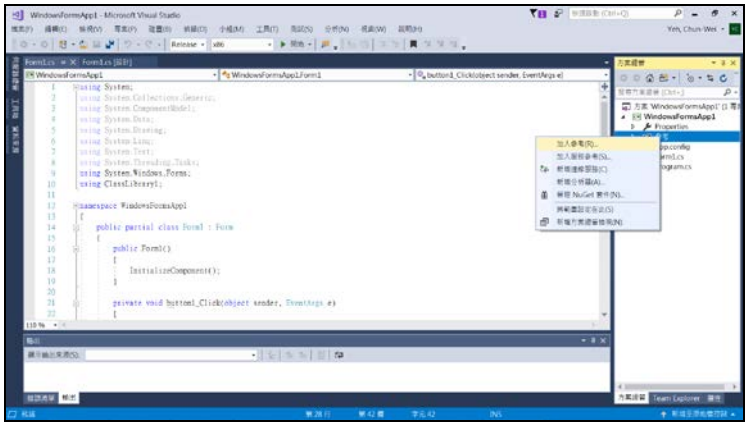


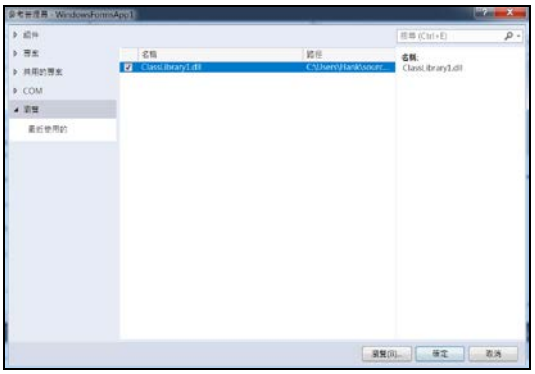
9. Build Project (or Build ClassLibrary1), then Visual Studio will generate a managed .dll file (ClassLibrary1.dll).

10. The man-machine interface, named WindowsFormsApp1, will be created by using C# to new a button and a textbox as follows.



11. Add ClassLibrary1.dll to your C# project.



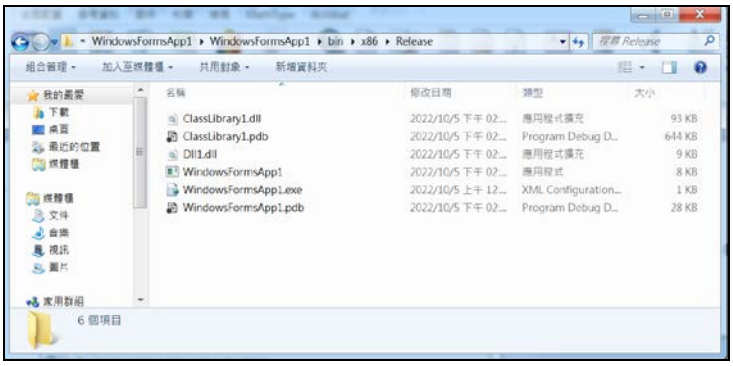


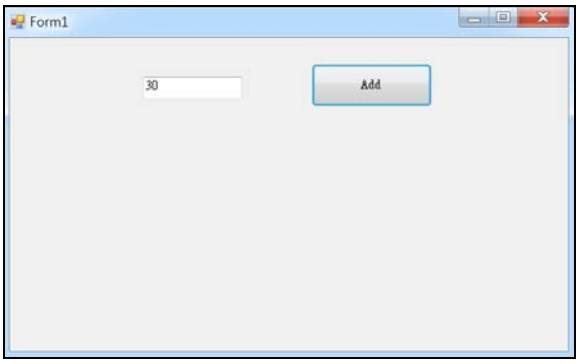
12. Call the function in the event of buttion1_Click:

ClassLibrary1.h | Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using ClassLibrary1;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            ManagedClass calculator = new ManagedClass();

            int a = 10;
            int b = 20;

            int c = calculator.Add(a, b);

            textBox1.Text = c.ToString();
        }
    }
}
```

10. Build → Build WindowsFormsApp1, and copy the Dll1.dll file built in step 5 to the folder of WindowsFormsApp1.exe. Now your WindowsFormsApp1.exe is executable.





Exercises:

1. Please add the functions of subtract, multiply and divide to Dll1.dll.
2. Please add the corresponding functions of the subtract, multiply and divide to ClassLibrary1.h.
3. Please add the button and textbox to the GUI.
4. Run the demo to verify your work.