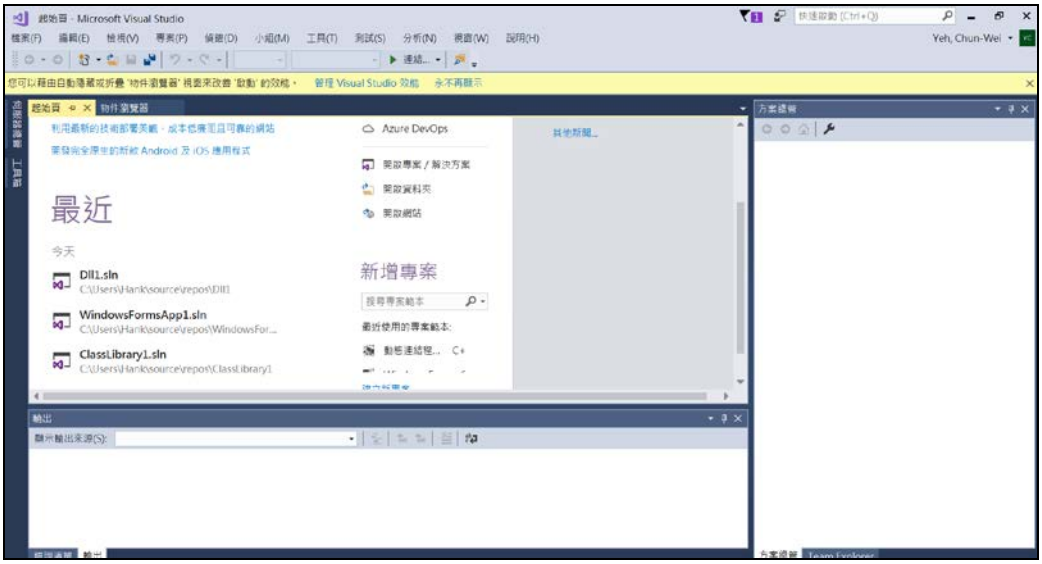
國立臺北科技大學自動化所 嵌入式工業機器視覺

Creating Dynamic Libraries (.dll) on Windows*

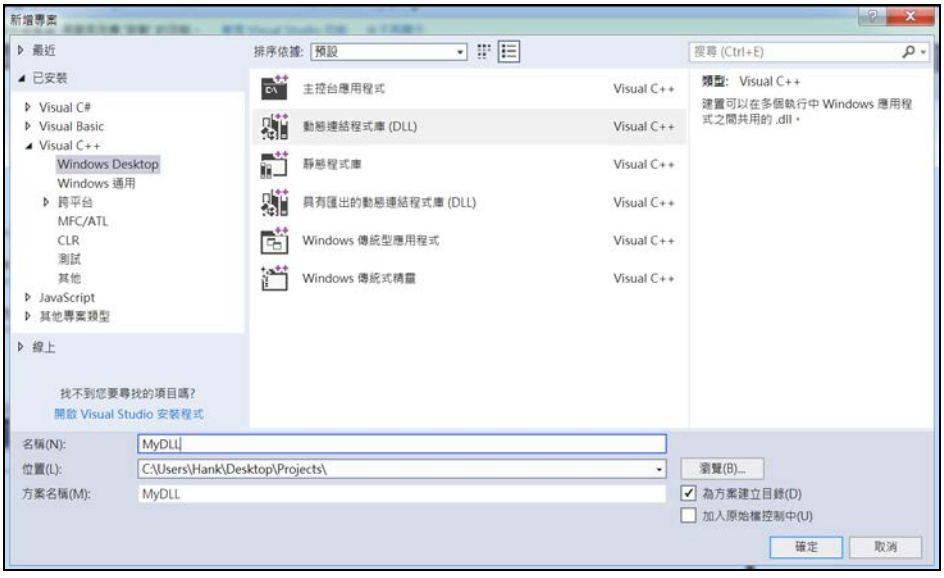
[Using P/Invoke for C++]

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.

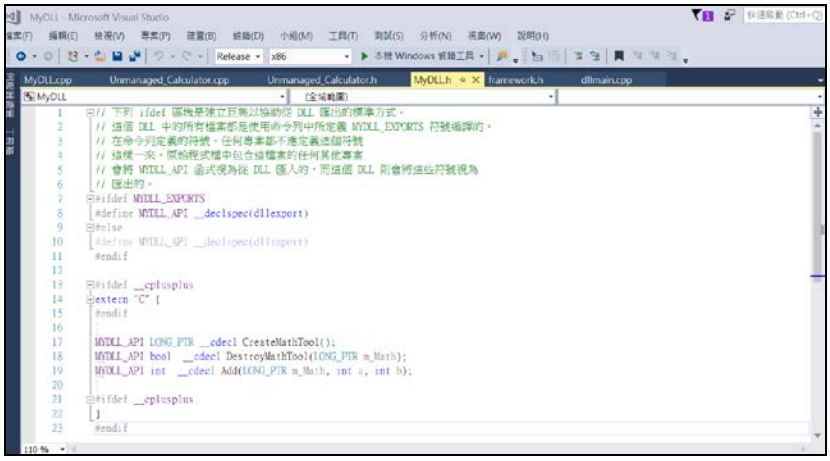
1. Select the menu File > New > Project



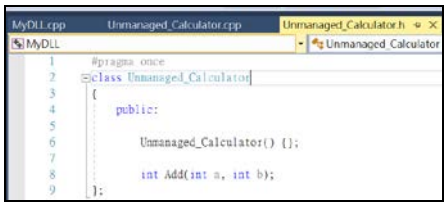
2. Select the Visual C++ -> Windows Desktop and the DLL option, and then put the project name and path.



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 and MyDLL.h) provides a simple demonstration of this.



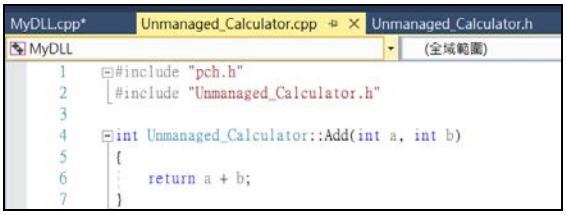
```
// 下列 ifdef 區塊是建立匯出 DLL 的標準方式。
// 這個 DLL 中的所有檔案都是使用命令列中所定義 MYDLL_EXPORTS 符號編譯的。
// 在命令列定義的符號，任何專案都不應定義這個符號
// 這樣一來，原始程式檔中包含這檔案的任何其他專案
// 會將 MYDLL_API 函式視為從 DLL 匯入的，而這個 DLL 則會將這些符號視為
// 匯出的。
#ifdef MYDLL_EXPORTS
#define MYDLL_API __declspec(dllexport)
#else
#define MYDLL_API __declspec(dllimport)
#endif

#ifdef __cplusplus
extern "C" {
#endif

MYDLL_API LONG_PTR __cdecl CreateMathTool();
MYDLL_API bool  __cdecl DestroyMathTool(LONG_PTR m_Math);
MYDLL_API int  __cdecl Add(LONG_PTR m_Math, int a, int b);

#ifdef __cplusplus
}
#endif
```



```
#pragma once
class Unmanaged_Calculator
{
    public:

        Unmanaged_Calculator() {};

        int Add(int a, int b);
};
```

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

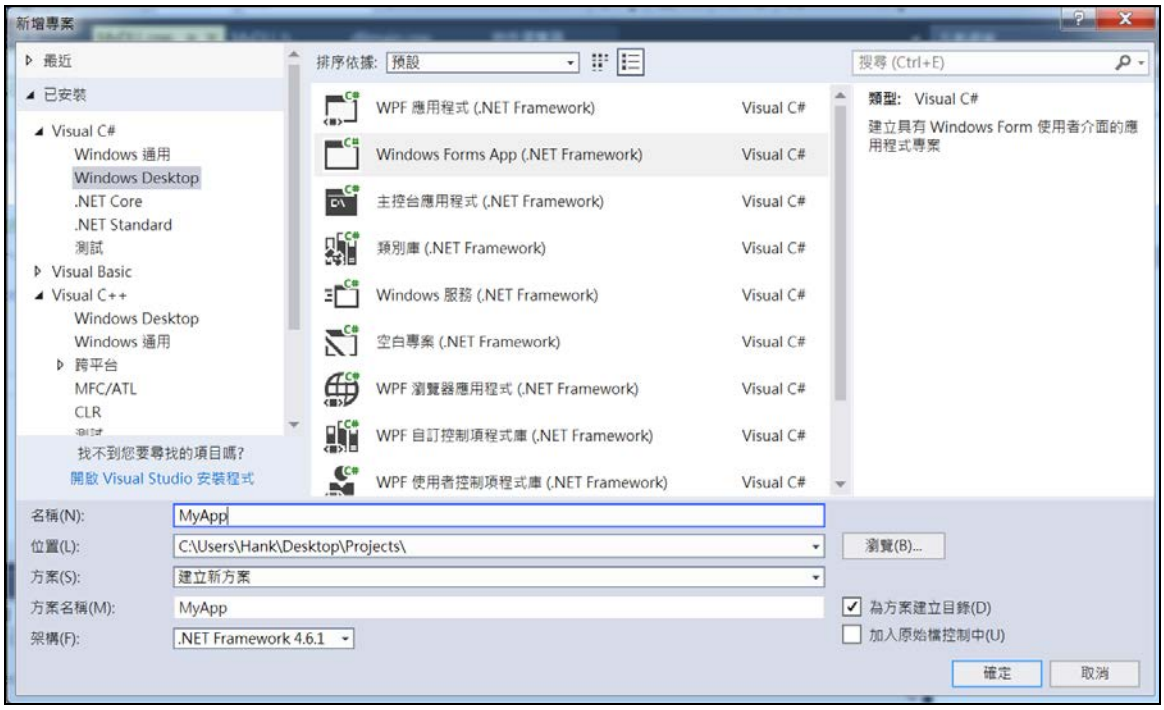
```
// MyDLL.cpp : 定義 DLL 應用程式的匯出函式。
//
#include "pch.h"
#include "Unmanaged_Calculator.h"
#include "MyDLL.h"

// 這是匯出函式的範例。
MYDLL_API LONG_PTR __cdecl CreateMathTool()
{
    return (LONG_PTR) new Unmanaged_Calculator();
}
MYDLL_API bool  __cdecl DestroyMathTool(LONG_PTR m_Math)
{
    Unmanaged_Calculator *Calculator = (Unmanaged_Calculator*)m_Math;

    if (Calculator) delete Calculator;

    return true;
}

MYDLL_API int  __cdecl Add(LONG_PTR m_Math, int a, int b)
{
    Unmanaged_Calculator *Calculator = (Unmanaged_Calculator*)m_Math;

    return (Calculator->Add(a, b));

}
```

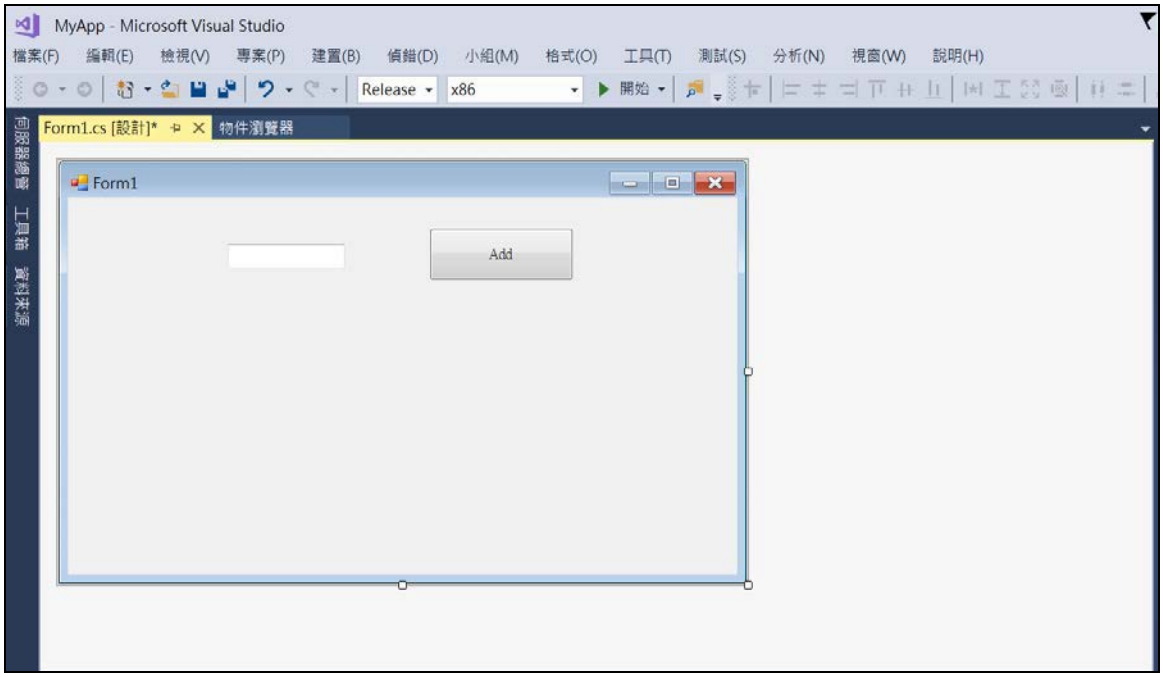


```
#include "pch.h"
#include "Unmanaged_Calculator.h"

int Unmanaged_Calculator::Add(int a, int b)
{
    return a + b;
}
```

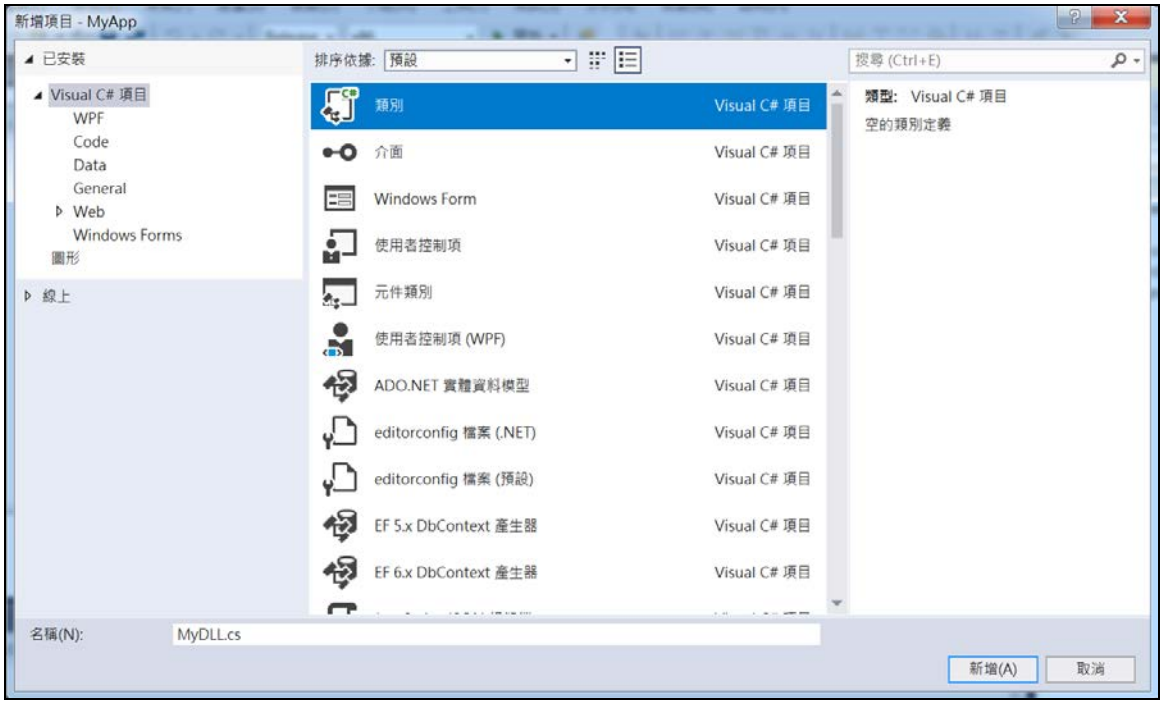
5. Build → Build Project (or Build MyDLL), then Visual Studio will generate a .dll file and an associated .lib import file.

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

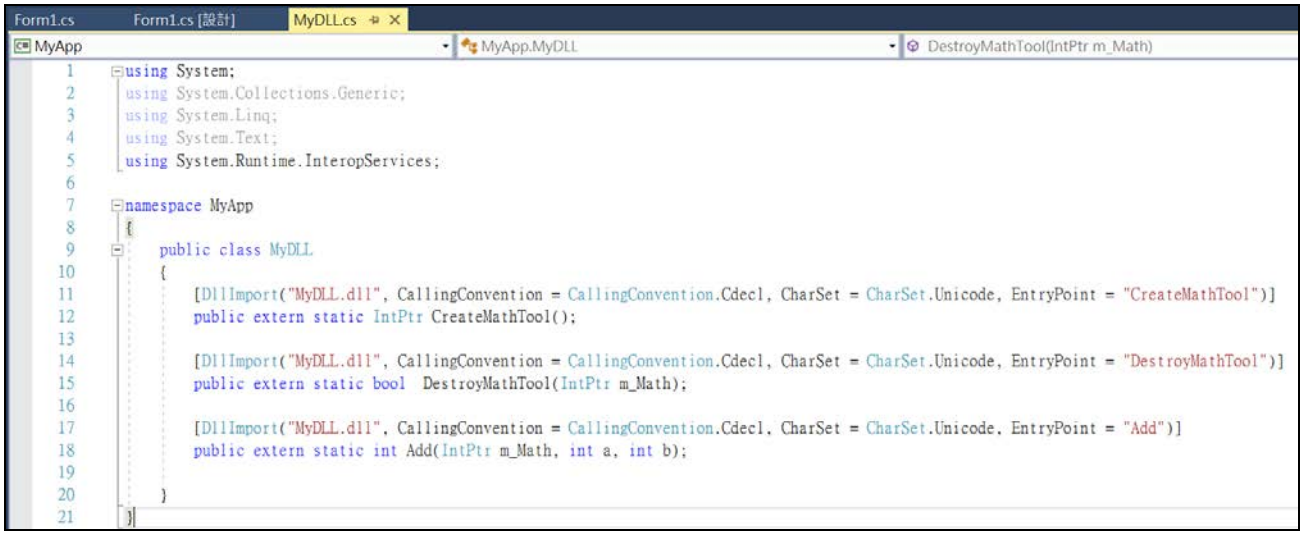




7. Add MyDLL.cs to your C# project.



8. The details of MyDLL.cs are as follows.

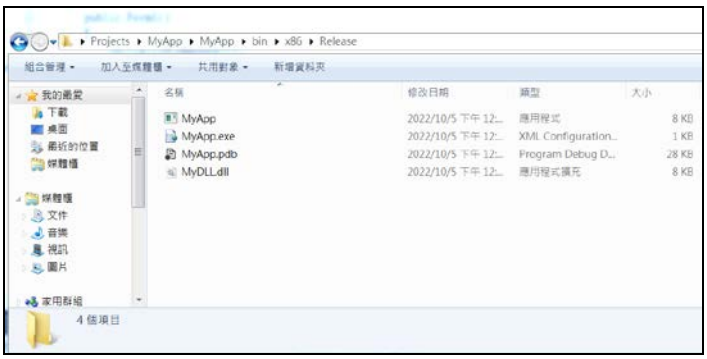


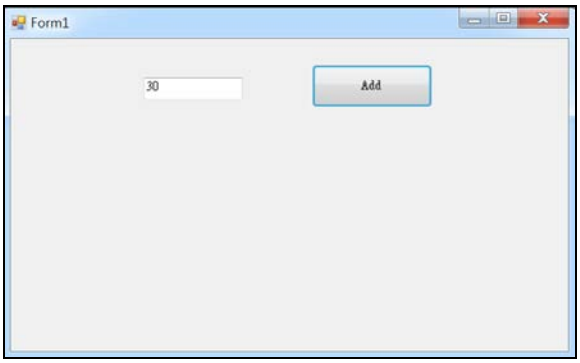
9. Call the function in the events of Form1_Load, Form1_FormClosing and buttion1_Click:

Form1.cs

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MyApp
{
    public partial class Form1 : Form
    {
        IntPtr m_Math;

        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            int a = 10;
            int b = 20;

            int c = MyDLL.Add(m_Math, a, b);

            textBox1.Text = c.ToString();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            m_Math = MyDLL.CreateMathTool();
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            MyDLL.DestroyMathTool(m_Math);
        }
    }
}
```

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





Exercises:

1. Please add the functions of subtract, multiply and divide to MyDLL.dll.
2. Please add the entry point of the subtract, multiply and divide functions to MyDLL.cs.
3. Please add the button and textbox to the GUI.
4. Run the demo to verify your work.