

資料結構與C++進階班 樣版

講師：黃銀鵬

E-mail: yinpenghuang@gmail.com

重要，template具有代表C 變到 C++的意義其中之一(另一個為class)

樣版(想成資料型態猶如變數)

- ▶ 過去的變數都是固定型別，根據數值的變化來達到程式的多樣性。
- ▶ 樣板是為了將多樣型別當作變數，讓程式可以根據型別而產生變化。
- ▶ 樣版類別全部程式碼都要寫在h檔。

```
int Max(int a, int b)
{
    if(a>b) return a;
    else return b;
}
```

```
float Max(float a, float b)
{
    if(a>b) return a;
    else return b;
}
```

```
T Max(T a, T b)
{
    if(a>b) return a;
    else return b;
}
```

stl(standard
template library)
標準函式庫，學
template，也學會標
準函式庫要怎麼用。

無限多份的code，
重複3次以上的
code就要合在一起，
T是型別的變數

樣板函式

```
Microsoft Visual Studio Debug Console

Integer Data Array:
10097 18479 32194 29846 27805 17777 13550 635 26620 9841
Max Integer Number: 32194
Float Data Array:
0.64 0.5 0.91 0.31 0.98 0.73 0.52 0.38 0.53 0.19
Max Float Number: 0.98
```

```
#include <iostream>
#include <iomanip>

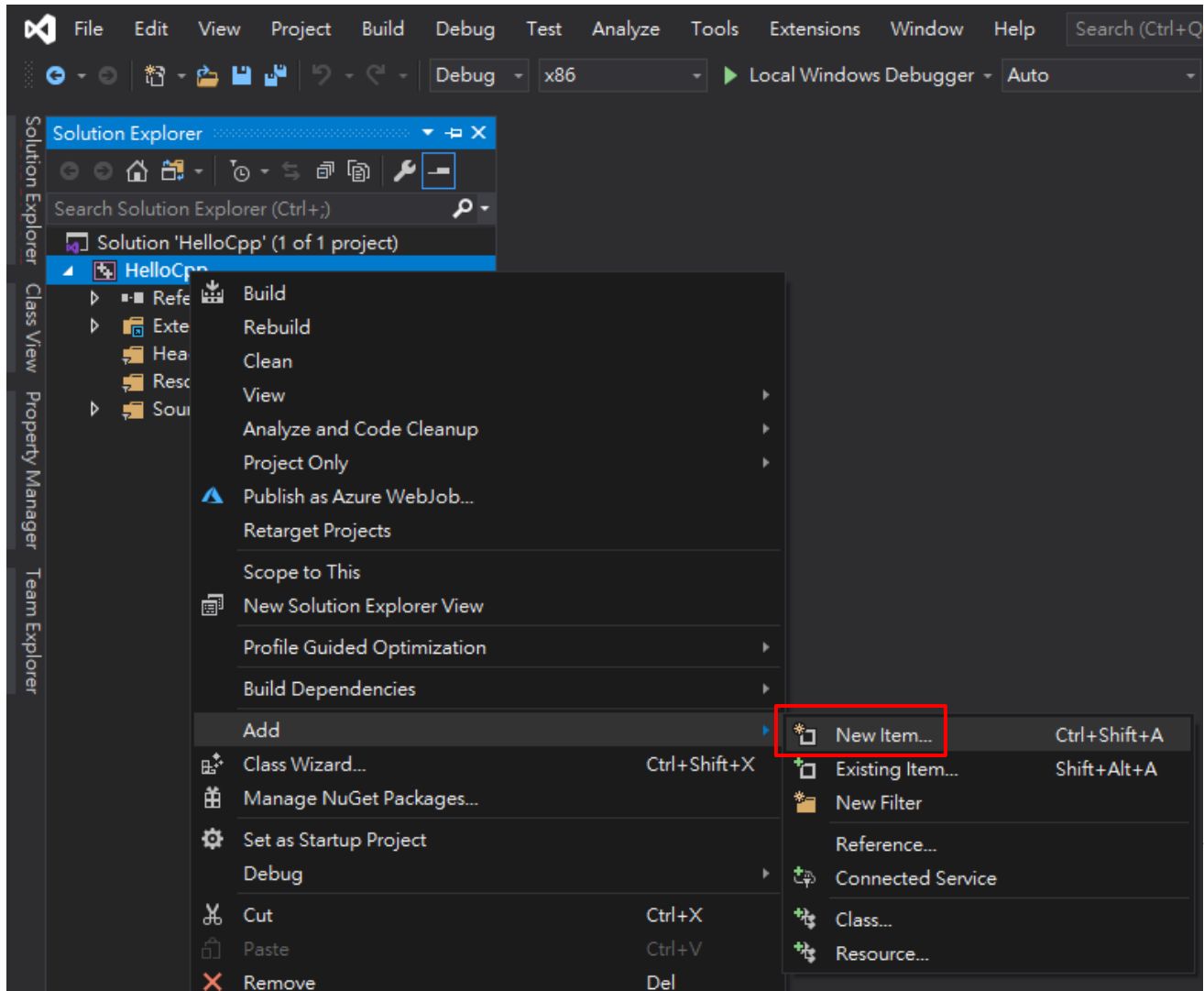
template<class T> T Max(T a, T b);

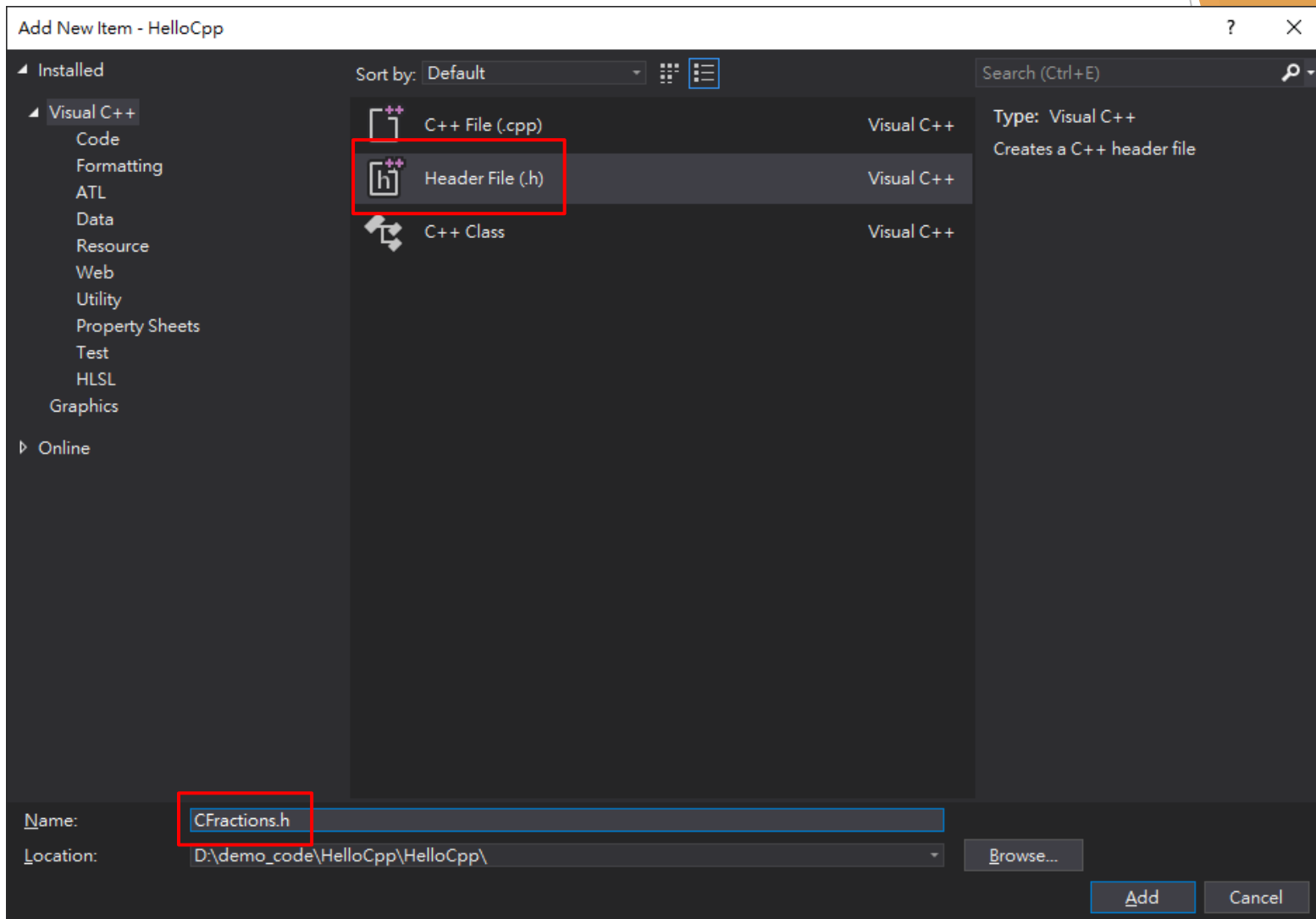
int main()
{
    int arraySize = 10;
    int* piArray = new int[arraySize];
    int i;
    srand((unsigned int)time(NULL));
    std::cout << "Integer Data Array: " << std::endl;
    for (i = 0; i < arraySize; ++i)
    {
        piArray[i] = rand();
        std::cout << std::setw(6) << piArray[i] << "\t";
    }
    std::cout << std::endl;
    int imax = piArray[0];
    for (int i = 1; i < arraySize; ++i)
    {
        用的T是甚麼型別
        imax = Max<int>(imax, piArray[i]);
    }
    std::cout << "Max Integer Number: " << imax << std::endl;
```

```
float* pfArray = new float[arraySize];
std::cout << "Float Data Array: " << std::endl;
for (i = 0; i < arraySize; ++i)
{
    pfArray[i] = (float)rand() / RAND_MAX;
    std::cout << std::setprecision(2) << std::setw(6) << pfArray[i] << "\t";
}
std::cout << std::endl;
float fmax = pfArray[0];
for (int i = 1; i < arraySize; ++i)
{
    fmax = Max<float>(fmax, pfArray[i]);
}
std::cout << "Max Float Number: " << fmax << std::endl;
}

template<class T>
T Max(T a, T b)
{
    if (a > b) return a;
    else return b;
}
```

新增標頭檔



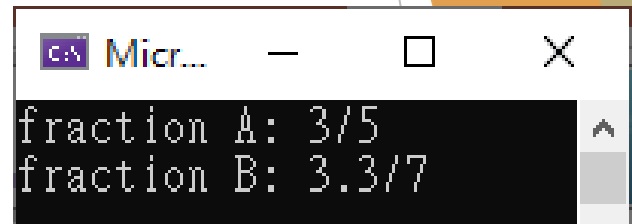


樣板類別

```
#include <iostream>
#include "CFractions.h"

int main()
{
    CFractions<int, int> fractionA;
    fractionA.SetNumerator(3);
    fractionA.SetDenominator(5);
    std::cout << "fraction A: ";
    fractionA.Print();
    std::cout << std::endl;

    CFractions<float, int> fractionB(3.3, 7);
    std::cout << "fraction B: ";
    fractionB.Print();
    std::cout << std::endl;
}
```



Micr...

fraction A: 3/5
fraction B: 3.3/7

.h內

```
#pragma once
#include <iostream>

template <class T1, class T2>
class CFractions
{
public:
    CFractions();
    CFractions(T1 num, T2 den);
    ~CFractions();
    void SetNumerator(T1 num);
    void SetDenominator(T2 den);
    void Print();
private:
    T1 m_Numerator; //分子
    T2 m_Denominator; //分母
};

template<class T1, class T2> constuctor
inline CFractions<T1, T2>::CFractions()
    : m_Numerator()
    , m_Denominator()
{
}

template<class T1, class T2>
inline CFractions<T1, T2>::CFractions(T1 num, T2 den)
{
    m_Numerator = num;
    m_Denominator = den;
}
```

有可能是自訂一型別，沒辦法確定型別，就不要給初始值，初始化為該型別的預設值，若為自定義型別，則呼叫那個類別的 standard default

```
template<class T1, class T2>
inline CFractions<T1, T2>::~~CFractions()
{
}

template<class T1, class T2>
inline void CFractions<T1, T2>::SetNumerator(T1 num)
{
    m_Numerator = num;
}

template<class T1, class T2>
inline void CFractions<T1, T2>::SetDenominator(T2 den)
{
    m_Denominator = den;
}

template<class T1, class T2>
inline void CFractions<T1, T2>::Print()
{
    std::cout << m_Numerator << "/" << m_Denominator;
}
```


參考書

- ▶ C++ Templates 全覽, 2/e (C++ Templates: The Complete Guide, 2/e)

- ▶ 譯者：劉家宏
- ▶ 出版社：碁峰
- ▶ ISBN：9865022303



難嗑，工具書，
遇到再查

想想看

- ▶ 使用**templates**技術時，可否用保留字**struct**取代**class**？(可)
- ▶ 能使用哪些類別當作參數來指定給**templates**？

自定義型可以，隨便取代不行，
確定支援**template**內的功能。因為沒有**operator overloading**所以，
有些比較不能用