國立台灣海洋大學資訊工程系 2A　C++ 程式設計 期中考試題

姓名：_____

系級：_____

學號：_____

104/04/21 (二)

考試時間：**09:30 – 12:00**

請儘量回答，總分有 **120**，看清楚每一題所佔的分數再回答

考試規則：1. 不可以翻閱參考書、作業及程式
2. 不可以使用任何形式的電腦 (包含手機、計算機、相機以及其它可運算或是連線的電子器材)
3. 請勿左顧右盼、請勿交談、請勿交換任何資料、試卷題目有任何疑問請舉手發問 (看不懂題目不見得是你的問題，有可能是中英文名詞的問題)、最重要的是隔壁的答案可能比你的還差，白卷通常比錯得和隔壁一模一樣要好
4. 提早繳卷同學請直接離開教室，請勿逗留喧嘩
5. 違反上述任何一點之同學一律依照學校規定處理
6. 繳卷時請繳交***簽名過之試題卷及答案卷***

在一個應用軟體中，需要處理二維的*下三角*矩陣以及其運算，因此將矩陣設計成一個 TriAngMatrix 類別，每一個這樣子類別的物件代表一個 n 列(row), n 行(column)的*下三角*矩陣，這樣子的物件提供一些基本的資料設定與運算功能，如下測試範例所示：

```
1. ifstream infile("sample.dat"); // 開啓檔案串流
2. TriAngMatrix matA(infile);   // 由檔案串流中初始化矩陣 A
3. TriAngMatrix matB(infile);   // 由檔案串流中初始化矩陣 B
4.
6. TriAngMatrix *matC=0;        // 將矩陣 A 乘矩陣 B, 存放於
7. matC = matA.multiply(matB); // 動態配置的 TriAngMatrix 物
8.                              // 件中並且將該矩陣指標傳回
9. cout << "A =\n" << matA << endl; // 列印矩陣 A 的內容
```

```
10. cout << "B =\n" << matB << endl;
11. cout << "A*B =\n" << *matC << endl;
12. TriAngMatrix matD;
13. matD = matA.add(*matC); // 傳回結果物件
14. cout << "A+A*B =\n" << matD << endl;
15.
16. delete matC;     // 刪除矩陣 C 物件
```

檔案 sample.dat 內容如下：

```
3
1.0
2.1 3.0
7.0 8.3 9.5
3
1.1
2.0 3.4
3.2 4.0 5.0
```

輸出結果如下：

```
A =
    1.0     0.0     0.0
    2.1     3.0     0.0
    7.0     8.3     9.5
B =
    1.1     0.0     0.0
    2.0     3.4     0.0
    3.2     4.0     5.0
A*B=
    1.1     0.0     0.0
    8.3    10.2     0.0
   54.7    66.2    47.5
A+A*B =
    2.1     0.0     0.0
   10.4    13.2     0.0
   61.7    74.5    57.0
```

請回答下列問題，依序完成 TriAngMatrix 類別的設計，請分別註明程式應該放在哪一個檔案中，以及應該引入的標頭檔案

1. [5] 假設在這個應用程式內所有的矩陣最大是 50 列, 50 行，我們希望先用最簡單的方法來實作，所以用固定大小的二維 double 陣列來存放資料，應該怎麼定義 TriAngMatrix 類別內存放矩陣內容的資料成員變數？由於這個類別所描述的矩陣的列數與行數相等，但卻不是固定的，所以你也需要設計記錄列數(行數)的資料成員變數，請注意存取權限

Sol:

--- TriAngMatrix.h ---

```
#pragma once
class TriAngMatrix
{
private:
      double m_data[50][50];
      int m_dim;
};
```

2. [15] 接上題，根據上面這一段程式的要求，這個 TriAngMatrix 類別**至少**應該有兩個建構元 (constructor) 函式；一個 multiply() 成員函式負責將兩個矩陣相乘起來，請不要修改原本兩個矩陣的內容，新配置一個矩陣物件並且回傳其指標；一個 add()成員函式負責將兩個矩陣加起來，請不要修改原本兩個矩陣的內容，請回傳一個 TriAngMatrix 物件；一個 print() 成員函式負責列印矩陣的資料到傳入的 ostream&型態的輸出串流；以及一個全域的 operator<<() 函式用來擴充 iostream；請以 class 語法定義這個類別，以及這幾個成員函式的原型 (prototype)，請注意參數的型態、回傳值的型態、以及存取權限 (public or private)，可以使用 const 的地方請盡量使用 const

Sol:

```
--- TriAngMatrix.h ---
#pragma once
#include <iostream>
using std::istream;
using std::ostream;
class TriAngMatrix
{
public:
    TriAngMatrix();
    TriAngMatrix(istream & is);
    TriAngMatrix* multiply(const TriAngMatrix& rhs) const;
    TriAngMatrix add(const TriAngMatrix& rhs) const;
    void print(ostream& os);
private:
    double m_data[50][50];
    int m_dim;
};

ostream& operator<<(ostream& os, TriAngMatrix& matrix);
```

3. [5] 請撰寫預設建構元 (default constructor) 成員函式，運用初始化串列 (initialization list) 將矩陣維度設為 0x0，請問在上面程式中什麼地方需要使用

Sol:

```
--- TriAngMatrix.cpp ---
TriAngMatrix::TriAngMatrix():m_dim(0)
{
}
第 12 列  TriAngMatrix matD;  時使用
```

4. [10] 請撰寫由檔案串流中讀取資料的建構元函式，資料的格式如上圖所示，首先有一個整數代表列數(行數) (例如 3 代表 3x3 的矩陣)，接下來每一列資料是這個下三角矩陣中一列資料的內容

Sol:

```
--- TriAngMatrix.cpp ---
TriAngMatrix::TriAngMatrix(istream& is)
{
    int i, j;
    is >> m_dim;
    for (i=0; i<m_dim; i++)
        for (j=0; j<=i; j++)
            is >> m_data[i][j];
}
```

5. [10] 請撰寫矩陣相乘的成員函式，回傳值的型態為此物件的指標 (TriAngMatrix*)，基本的矩陣相乘範例如右 $\begin{bmatrix} a & 0 \\ b & c \end{bmatrix} \times \begin{bmatrix} d & 0 \\ e & f \end{bmatrix} = \begin{bmatrix} ad & 0 \\ bd+ce & cf \end{bmatrix}$，請注意函式的參數運用 call-by- reference 方

式傳入，函式裡應該先檢查兩個矩陣是否可以相乘，如果可以相乘的話，應該要運用 new 配置一個新的 TriAngMatrix 的物件，將相乘後的矩陣資料設定好以後並且將指標傳回，不能相乘的話則傳回 0，請特別注意上三角的部份是 0，請避免不需要的乘法

Sol:

```
--- TriAngMatrix.cpp ---
TriAngMatrix* TriAngMatrix::multiply(const TriAngMatrix& rhs) const
{
    int i, j, k;
    TriAngMatrix *ptr=0;
    if (m_dim == rhs.m_dim)
    {
        ptr = new TriAngMatrix;
        ptr->m_dim = m_dim;
        for (i=0; i<m_dim; i++)
            for (j=0; j<=i; j++)
                for (ptr->m_data[i][j]=0, k=j; k<=i; k++)
                    ptr->m_data[i][j] += m_data[i][k] * rhs.m_data[k][j];
    }
    return ptr;
}
```

6. [10] 請撰寫矩陣相加的成員函式，相加的運算為行與列對應的元素相加，回傳值的型態為此 TriAngMatrix 類別的物件，如果矩陣維度不同，無法相加的話，傳回矩陣的維度設為 0

Sol:

```
--- TriAngMatrix.cpp ---
TriAngMatrix TriAngMatrix::add(const TriAngMatrix& rhs) const
{
    int i, j;
    TriAngMatrix tmp;
    if (m_dim != rhs.m_dim)
        tmp.m_dim = 0;
    else
        tmp.m_dim = m_dim;
    for (i=0; i<tmp.m_dim; i++)
        for (j=0; j<=i; j++)
            tmp.m_data[i][j] = m_data[i][j] + rhs.m_data[i][j];
    return tmp;
}
```

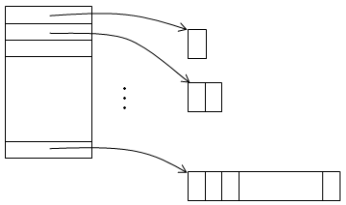
7. [10] 請製作一個 print(ostream& os) 成員函式列印矩陣的內容到輸出串流，並且製作 operator<<() 函式，運用 std::setprecision(int), std::fixed, std::setw(int)調整輸出格式

Sol:

```
--- TriAngMatrix.cpp ---
void TriAngMatrix::print(ostream& os)
{
    int i, j;
    for (i=0; i<m_dim; i++)
    {
        for (j=0; j<=i; j++)
            os << setw(6) << fixed << setprecision(1) << m_data[i][j] << " ";
        for (j=i+1; j<m_dim; j++)
            os << setw(6) << fixed << setprecision(1) << 0.0 << " ";
        os << endl;
    }
}
ostream& operator<<(ostream& os, TriAngMatrix& matrix)
{
```

```
    matrix.print(os);
    return os;
}
```

8. [15] 由於程式有可能需要處理更大的*下三角*矩陣，例如 1000x1000 的矩陣，如果希望能夠節省存放右上矩陣中那些內容為 0 的記憶體，請重做題 1, 3 與 4，動態地運用 new 來配置記憶體? (注意：如果你配置方法適當，同時前面沒有作不必要的動作的話，你在 2, 4, 5, 6, 7 題裡所寫的函式裡陣列存取的部份應該不需要更改，只有在產生新陣列時需要運用新的建構元，反之你可能需要修改上面加法或是乘法的程式；另外建構元請傳入一個整數的陣列維度，並且使用預設參數的語法使其成為 default ctor)，提示：請修改資料成員以及建構元，配置如右上圖的三角形二維陣列



Sol:

```
--- TriAngMatrix.h ---
class TriAngMatrix
{
public:
    TriAngMatrix(int dim=0);
    TriAngMatrix(istream & is);
    virtual ~TriAngMatrix();
    TriAngMatrix(const TriAngMatrix& src);
    TriAngMatrix& operator=(const TriAngMatrix& rhs);
    ...
private:
    double **m_data;
    int m_dim;
};
--- TriAngMatrix.cpp ---
TriAngMatrix::TriAngMatrix(int dim):m_dim(dim),m_data(0)
{
    int i;
    if (m_dim>0)
    {
        m_data = new double*[m_dim];
        for (i=0; i<m_dim; i++)
            m_data[i] = new double[i+1];
    }
}
TriAngMatrix::TriAngMatrix(istream& is)
{
    int i, j;
    is >> m_dim;
    m_data = new double*[m_dim];
    for (i=0; i<m_dim; i++)
        m_data[i] = new double[i+1];
    for (i=0; i<m_dim; i++)
        for (j=0; j<=i; j++)
            is >> m_data[i][j];
}
```

9. [10] 為配合上題中動態配置記憶體的機制，你應該如何撰寫解構元來釋放所配置的記憶體

Sol:

```
--- TriAngMatrix.h ---
class TriAngMatrix
{
public:
```

```
    …
    virtual ~TriAngMatrix();
    ...
};
--- TriAngMatrix.cpp ---
TriAngMatrix::~TriAngMatrix(void)
{
    int i;
    for (i=0; i<m_dim; i++)
        delete[] m_data[i];
    delete[] m_data;
}
```

10. [10] 為了配合使用動態配置的記憶體，請製作一個拷貝建構元 TriAngMatrix(const TriAngMatrix &src)，請問上面測試範例程式在哪幾列會使用到這個函式

Sol:

```
--- TriAngMatrix.h ---
class TriAngMatrix
{
public:
    …
    TriAngMatrix(const TriAngMatrix& src);
    ...
};
--- TriAngMatrix.cpp ---
TriAngMatrix::TriAngMatrix(const TriAngMatrix& src):m_dim(src.m_dim)
{
    int i, j;
    m_data = new double*[m_dim];
    for (i=0; i<m_dim; i++)
        m_data[i] = new double[i+1];
    for (i=0; i<m_dim; i++)
        for (j=0; j<=i; j++)
            m_data[i][j] = src.m_data[i][j];
}
第 13 列  matA.add(*matC)  傳回矩陣物件時使用
```

11. [10] 為了配合動態配置的記憶體，請製作一個設定運算子 operator=(const TriAngMatrix &rhs)，請問上面測試範例程式在什麼地方會使用到這個函式

Sol:

```
--- TriAngMatrix.h ---
class TriAngMatrix
{
public:
    …
    TriAngMatrix& operator=(const TriAngMatrix& rhs);
    ...
};
--- TriAngMatrix.cpp ---
TriAngMatrix& TriAngMatrix::operator=(const TriAngMatrix& rhs)
{
    int i, j;
    if (&rhs == this)
        return *this;
    for (i=0; i<m_dim; i++)
        delete[] m_data[i];
    delete[] m_data;
    m_dim = rhs.m_dim;
    m_data = new double*[m_dim];
    for (i=0; i<m_dim; i++)
```

```
            m_data[i] = new double[i+1];
        for (i=0; i<m_dim; i++)
            for (j=0; j<=i; j++)
                m_data[i][j] = rhs.m_data[i][j];
        return *this;
    }
```

第 13 列 matD = matA.add(*matC) 設定給 matD 時使用

12. [10] 假設右圖中由 infile 檔案資料串流中讀入 10 個 5x5 的矩陣，請寫一小段程式運用 vector 類別的 iterator 將 matrixArray 裡所有的矩陣乘起來，請注意由於在 5 中所定義的 multiply 成員函式會動態配置新的矩陣物件以儲存結果，這些暫時性的矩陣需要用 delete 敘述刪除，才不會造成記憶體遺失(memory leakage)，另外請問上面這段程式有使用拷貝建構元嗎?

```
1.  int i;
2.  vector<TriAngMatrix> matrixArray;
2.  for (i=0; i<10; i++)
3.      matrixArray.push_back(TriAngMatrix(infile));
```

Sol:

```
--- main.cpp ---
#include <vector>
using std::vector;
#include "TriAngMatrix.h"
#include <iostream>
using std::cout;
    …
    vector<TriAngMatrix> matrixArray;
    for (int i=0; i<10; i++)
        matrixArray.push_back(TriAngMatrix(infile));
    vector<TriAngMatrix>::iterator iter;
    TriAngMatrix prod(*matrixArray.begin());
    for (iter=matrixArray.begin()+1; iter!=matrixArray.end(); iter++)
    {
        matC = prod.multiply(*iter);
        cout << *matC << endl;
        prod = *matC;
        delete matC;
    }
    cout << "Product is\n" << prod << endl;
```

push_back(TriAngMatrix(infile)); 時會使用拷貝建構元