國立台灣海洋大學資訊工程系 C++ 程式設計 期中考試題

姓名：_____ 系級：_____ 學號：_____

95/04/19

考試時間：**13:20 – 15:30**

試題敘述蠻多的，看清楚題目問什麼，針對重點回答是很重要的，總分有 **115**，請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. 不得使用任何形式的電腦 (包含計算機)
3. 不得左顧右盼、不得交談、不得交換任何資料、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，不得逗留喧嘩
5. **違反上述任何一點之同學以作弊論，一律送請校方處理**
6. 繳卷時請繳交***簽名過之試題卷及答案卷***

1. a. [5] 請問將每一個類別的宣告和定義都分開來存放在 .h 和 .cpp 的檔案中有什麼好處?
   b. [5] 請問使用 C++ 語言時，在類別層次如何實作物件導向程式設計的 "封裝 (Encapsulation)" 特性?
   c. [5] 請問在撰寫程式時程式設計者在心裡面常常對資料內容有一些假設(例如: 指標內容不為 0, 經過 selectionSort 以後的序列應該是按照順序排列的), 我們在程式裡應該運用什麼敘述把這些假設寫出來? (請舉一個簡單的例子)
   d. [5] 請問下圖左程式片段中 inline 的意義為何? (為什麼要用 inline 這個語法?)

```
01 inline double square(double x)
02 {
03         return x * x;
04 }
```

```
01 class Quoter
02 {
03 public:
04         Quoter();
05         int lastQuote() const;
06 private:
07         int m_lastQuote;
08 };
```

   e. [10] 請問上圖右程式片段中 const 的意義為何? (為什麼要用 const 呢? 如此宣告的成員函式在撰寫時有什麼要注意的?)

Sol:

a. .h 檔案放類別的宣告 (介面)，.cpp 檔案中放類別的定義 (實作),
不同類別的程式分開來存放的目的是容易管理和修改，修改後也可以節省重新編譯的時間
.h 和 .cpp 檔案分開的目的是：只有 .h 檔案的內容是給其它模組引入的，.cpp 檔案的內容則不是，所以必須分開不同的檔案來存放

b. 第一是製作介面函式，所有的資料在使用時都透過這一組介面函式，常常這一組介面函式是以服務為導向的
第二是強制將資料放在 private 區段中，將介面函式放在 public 區段中，如此編譯器可以協助檢查出不透過介面函式的資料存取

c. 通常可以用 assert 敘述來將心裡面假設的事情明白地寫在程式裡，也可以用 assert 敘述將單元測試寫進程式裡，你寫一段程式一定會假設你的程式正確執行，這個假設其實和你使用的資料,測試的情境有關係,嘗試在各種情形下預測自己程式的表現,然後用 assert 敘述寫進程式裡。

d. inline 是告訴編譯器在翻譯這個程式中呼叫這個函式的敘述，例如 r = square(x+y);，時將這個敘述換成類似 z = x + y; r = z * z; 這樣子等效的敘述，以增進程式執行時的效率，注

意 inline 函式的宣告只對於同一個檔案內呼叫 square 的敘述有作用。

e. int lastquote() const; 為一個 const function 的宣告，這樣子的函式裡面不可以修改物件的狀態 (物件所有的資料成員)，宣告 const function 是對將來使用這個函式的程式的一種保障，呼叫這樣的函式之後物件內的狀態絕對不會有變化，不會有任何的 side effect

2. a. [10] 請問下圖左程式片段的設計中違背了物件導向程式設計的什麼原則? 該如何修改?

```
01 class Book
02 {
03 public:
04         Book(string title, string author);
05         string getTitle();
06         string getAuthor();
07 private:
08         string m_title;
09         string m_author;
10 };
11
12 ...
13 Book book("Harry Potter", "J. K. Rowling");
14 cout << "Title:" <<book.getTitle() << endl;
15 cout << "Author:" << book.getAuthor() << endl;
```

```
01 #include <iostream>
02 #include <vector>
03 #include <string>
04 using namespace std;
05
06 int main()
07 {
08         ifstream inf("inputData.txt");
09         string line;
10         vector<string> lines;
11
12         while (getline(inf, line))
13                 lines.push_back(line);
14
15         for (int i=0; i<lines.size(); i=i+2)
16                 cout << i << ":" << lines[i] << endl;
17
18         return 0;
19 }
```

inputData.txt

```
January
February
March
April
May
June
July
August
September
October
November
December
```

b. [10] 請運用初始化串列 (initialization list) 完成上圖左程式中的建構元函式的實作?

Sol:

a. 基本上還是間接地違背了封裝的原則，雖然沒有直接地違背：客戶端直接存取物件內的資料，但是透過 Accessor 函式來把所有資料送出這個 Book 類別的物件，還是不恰當的，如果能夠避免的話應該盡量避免，以這個例子來說，應該替 Book 類別增加一個 print() 的介面如下：

```
void Book::print()
{
        cout << "Title:" << m_title <<endl;
        cout << "Author:" << m_author << endl;
}
```

14 及 15 列換成 book.print()，如此封裝得比較完全

b. Book::Book(string title, string author): m_title(title), m_author(author)

```
{
}
```

3. a. [7] 請說明上圖中間程式最主要邏輯功能為何? 輸出為何?

b. [8] 請問在程式中哪些地方可能有呼叫建構元函式?

c. [10] 請使用 vector<string> 類別的 iterator 來存取容器物件的內容，撰寫一小段取代第 15 列和第 16 列程式功能的程式?

```
0: January
2: March
4: May
6: July
8: September
10: November
```

Sol:

a. 開啟 inputData.txt 檔案

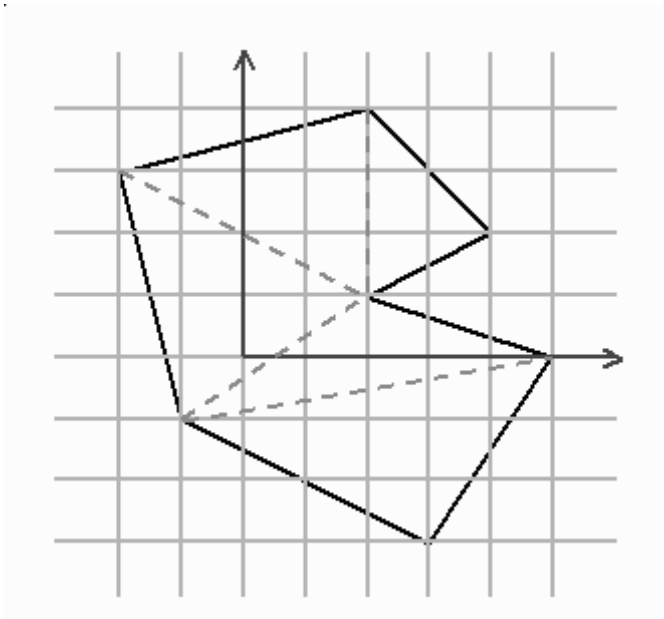
12 及 13 列由檔案中以列為單位依序將字串讀入程式中，存放在 lines 容器物件中

15 及 16 列將奇數列列印在螢幕上，印出資料如右

b. 第 8 列呼叫 ifstream(char *) 或是 ifstream(string) 建構元函式 1 次
第 9 列呼叫 string() 建構元函式 1 次
第 10 列呼叫 vector<string>() 建構元函式 1 次
第 13 列呼叫 string(string &) 建構元函式 12 次
第 18 列之前會反序解構 12 次 string 類別的物件 lines[i]，呼叫 ~string() 解構元
解構 1 次 vector<string> 類別的物件 lines
解構 1 次 string 類別的物件 line
解構 1 次 ifstream 類別的物件 inf

c. vector<string>::iterator iter;
for (iter=lines.begin(), i=0; iter<lines.end(); iter+=2, i+=2)
cout << i << ":" << *iter << endl;

4. [40] 下圖左為一個多邊形，在電腦繪圖的程式裡常常會出現，基本上我們可以把它表示為多個



```
ifstream inputFile("data.txt");                    主程式
Polygon graphicObject(inputFile);
graphicObject.draw();
```

data.txt

```
Polygon 1
5
Triangle 1
4 2
2 4
2 1
red
Triangle 2
2 1
2 4
-2 3
blue
Triangle 3
2 1
-2 3
-1 –1
green
Triangle 4
2 1
-1 –1
5 0
blue
Triangle 5
-1 –1
3 –3
5 0
yellow
```

三角形的組合，如此我們可以用三角形為單位來在螢幕上畫出來，也可以個別去指定它的顏色來著色，進一步也可以在各個三角形上面計算光的投射量，在各個三角形上貼上材質…現在我們來設計幾個基本的類別，並且利用建構元把這樣的多邊形的物件由檔案 data.txt 中讀取出來。下列說明為基本的評分要點，你在撰寫時只要把三個類別的內容寫完就可以了。

a. [3] 請宣告 Point, Triangle, 以及 Polygon 三個類別
b. [3] 多邊形 Polygon 類別中需要以 vector<Triangle *> 記錄裡面所有的三角形 Triangle 物件
c. [3] 三角形 Triangle 類別中需要設計三個 Point 類別的物件成員，同時 Triangle 類別也需要記錄三角形的顏色，
d. [3] 顏色請以 enum 宣告 red, green, blue, yellow, cyan, magenta
e. [3] 點的類別 Point 裡面只需要記錄兩個整數的座標值即可，視需要可能需要 accessor
f. [10] 三個類別都需要有一個接受輸入檔案串流 ifstream 參考的建構元，請注意讀取資料時要跳過 Polygon 1, Triangle 1, Triangle 2 …那些列的文字描述資料

g. [10] 請替 Polygon 及 Triangle 各撰寫一個 draw(() 的界面成員函式，Polygon::draw() 需要將 Polygon 裡面所有的三角形依序畫出, Triangle::draw() 請呼叫一個假設的全域繪圖函式 ::drawTriangle(int x1, int y1, int x2, int y2, int x3, int y3, int color); 來以指定的顏色畫出三角形

h. [5] 請替 Polygon 類別撰寫解構元函式

Sol:

```
// Point.h
#ifndef POINT_H
#define POINT_H
#include <fstream>
using namespace std;
class Point
{
public:
        Point (ifstream &);
        int getX();
        int getY();
private:
        int m_x, m_y;
};
#endif
```

```
// Point.cpp
#include "Point.h"
Point::Point(ifstream &infile)
{
        infile >> m_x >> m_y;
}
int Point::getX()
{
        return m_x;
}
int Point::getX()
{
        return m_x;
}
```

```
// Triangle.h
#ifndef TRIANGLE_H
#define TRIANGLE_H
enum Color {red, green, blue, yellow,
                cyan, magenta};
#include <fstream>
using namespace std;
class Triangle
{
public:
        Triangle(ifstream &);
        void draw();
private:
        Point m_point1, m_point2, m_point3;
        Color m_color;
};
#endif
```

```
// Triangle.cpp
#include "Triangle.h"
#include "Point.h"
#include <string>
using namespace std;
Triangle::Triangle(ifstream &infile):
        m_point1(infile), m_point2(infile),
        m_point3(infile)
{
        char cbuf[100]; string buf;
        infile.getline(cbuf, 99, '\n'); buf = cbuf;
        if (buf == "red") m_color = red;
        else if (buf == "green") m_color = green;
        else if (buf == "blue") m_color = blue;
        else if (buf == "yellow") m_color = yellow;
        else if (buf == "cyan") m_color = cyan;
        else if (buf == "magenta") m_color = magenta;
}
void Triangle::draw()
{
        drawTriangle(m_point1.getX(), m_point1.getY(),
                        m_point2.getX(), m_point2.getY(),
                        m_point3.getX(), m_point3.getY(),
                        m_color);
}
```

```
// Polygon.h
#ifndef POLYGON_H
#define POLYGON_H
#include <fstream>
#include <vector>
using namespace std;
class Triangle;
class Polygon
{
public:
        Polygon(ifstream &);
        Void draw();
        ~Polygon();
private:
        vector<Triangle *> m_triangles;
};
#endif
```

```
// Polygon.cpp
#include "Polygon.h"
#include "Triangle.h“
#include <fstream>
using namespace std;
Polygon::Polygon(ifstream &infile)
{
        Triangle *tmp;
        char buf[100];
        infile.getline(buf, 99, '\n');
        int nTriangles;
        infile >> nTriangle;
        infile.getline(buf, 99, '\n');
        for (int i=0; i<nTriangles; i++)
        {
                infile.getline(buf, 99, '\n');
                tmp = new Triangle(infile);
                m_triangles.push_back(tmp);
        }
}
```

```
void Polygon::draw()
{
        vector<Triangle *>::iterator iter;
        for (iter=m_triangles.begin();
                iter<m_triangles.end(); iter++)
            (*iter)->draw();
}
Polygon::~Polygon()
{
        vector<Triangle *>::iterator iter;
        for (iter=m_triangles.begin();
                iter<m_triangles.end(); iter++)
            delete (*iter);
}
```

人家都說一生中偶而會遇見困難，我倒是覺得一生中常常會遇見困難，所以遇見困難是正常的，重點在不要太快放棄，再堅持一下，也許就看到曙光了!

亡, 寫作業時最怕看到的不就是曙光...