物件導向程式設計及應用第三次作業

Due: 2019/12/14 13:00

※注意事項：請依照課程網站內所公告之“作業檔案命名規則與規定”進行作業檔案命名以及繳交作業，未依照規定將斟酌扣分。

- 第一題：(50%)

介面(Interface)在物件導向的觀念中是一種僅提供功能的簽章。在 JAVA、C#等新式程式語言中，已經提供關鍵字 interface 實作，傳統的 C++語言則可以利用 abstract base class (ABC) 的方式來實現介面。所謂的介面是僅提供方法的特殊類別，如人類、貓、狗…等都具有行走的介面，因此，我們可以建立一個 IWalk 的 abstract base class，該類別提供 Walk 的 pure virtual function（注意：介面僅能有方法，不能有變數）。而人類、貓、狗…等實際走路的方法(如用兩條腿站立走路、或是四肢著地行走)則由繼承自 IWalk 的衍生類別 Human、Cat、Dog 來實作 Walk 的函數。在習慣上，介面類別的名稱開頭皆以大寫的 I 字眼以作為識別。

請撰寫一個描述空間中的座標點的 Point 類別（可利用第二次作業中的 Point 類別修改）完成下列程式。

建立一個提供基本幾何運算的介面 IGeometry，具有下列方法：
1.計算表面積 double Area()
2.計算周長 double Perimeter()
3.計算體積 double Volume()

請撰寫三角錐(Pyramid)、長方體(Cuboid)、圓柱(Cylinder)三種幾何圖形繼承自 IGeometry，並各自提供下列擴充方法：

- 三角錐 Pyramid
  1.具有四個頂點座標的私有成員 Point* vertices
  2.建構子(constructor)、除構子(destructor)、拷貝建構子(copy constructor)及指定運算子(assignment operator)
  3.計算形心的公有方法 Point Center()

- 長方體 Cuboid
  1.具有八個頂點座標的私有成員 Point* vertices
  2.建構子、除構子、拷貝建構子及指定運算子
  3.計算三組邊長的公有方法 double* SideLength()
  4.計算三組面積的公有方法 double* SideArea()

- 直圓柱 Cylinder
  1.具有上下圓的圓心、半徑的私有成員 Point Top, Point Bottom, double r
  2.建構子、除構子、拷貝建構子及指定運算子
  3.計算高的公有方法 double Height()
  4.計算底圓面積的公有方法 double BottomArea()
  5.計算側面面積的公有方法 double SideArea()

請撰寫主程式，在主程式中，請以 IGeometry 建立動態陣列，並分別實體化三種圖形（可能如下程式碼所示）。各圖形建立時所需要的私有成員數據以亂數產生。在主程式中，請測試各類別的所有方法。

```
IGeometry** arr = new IGeometry*[3];
arr[0] = new Pyramid();
arr[1] = new Cuboid();
arr[2] = new Cylinder();
```

- 請將上述宣告撰寫於 HW03_1.h 標頭檔案內。
- 請將上述宣告的定義撰寫於 HW03_1.cpp 程式碼檔案內。
- 請撰寫一個主程式 useHW03_1.cpp，以測試上述功能。

※除輸入與輸出導向運算子外，不可宣告任何函式與類別為**夥伴(friend)**※

※請勿使用**標準樣板函式庫(Standard Template Library)**或**巨集指令**※

- 第二題：(50%)

佇列 (Queue) 是一個表示排隊的資料結構，它具備有先進先出 (first-in-first-out) 的特性。最簡單的實作方式，是以一維陣列和兩個變數 front 和 rear 來實作佇列。其中 front 表示佇列的第一個元素，而 rear 則是佇列的最後一個元素的位置。為了使得佇列的應用更靈活，我們已樣板類別來實作。這樣完成的佇列的類別宣告如下：

```
template <class KeyType>
class Queue
{
private:
    int front, rear;
    KeyType* queue;
    int MaxSize;
public:
    Queue(int MaxQueueSize = DefaultQueueSize);
    // Create an empty queu whose maximum size is MaxQueueSize
    bool IsFull();
    // if number of elements in the queue is equal to the maximum size of
    // the queue, return TRUE; otherwise, return FALSE
    bool IsEmpty();
    // if number of elements in the queue is equal to 0, return TRUE
    void Add(const KeyType& item);
    // if IsFull(), then QueueFull(); else insert item at rear of the queue
    KeyType* Delete();
    // if IsEmpty(), then QueueEmpty() and return 0;
    // else remove the item at the front of the queue and return a pointer to it
};
```

關於 Queue 的說明，可以參考 C++ Primer Plus 6th Ed. 第 12 章的介紹與程式碼實作。在本題中，不使用課本所介紹的 Linked List 方式實作，而以簡單的一維陣列實作。當然，仍然需要實作拷貝建構子、指定運算子、除構子…等必須實作的函數，以及需要的任何函數或運算子重載。

- 請將上述宣告與定義撰寫於 HW03_2.h 標頭檔案內。
- 請撰寫一個主程式 useHW03_2.cpp，以測試上述功能。

※除輸入與輸出導向運算子外，不可宣告任何函式與類別為**夥伴(friend)**※

※請勿使用**標準樣板函式庫(Standard Template Library)**或**巨集指令**※