C++ Basic Course

6/16/2018

Homework Assignment No. 12

(Total 100%)

1. **(60%)**: A maze class hierarchy includes a base class `MapSite` and three derived classes `Room`, `Wall`, and `Door`. The interfaces for each are:

```
enum Direction {North, South, East, West};
```

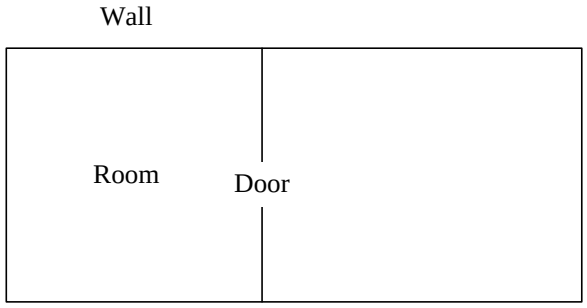
```
class MapSite {
   public:
      virtual void enter() = 0;
};
```

```
class Room : public MapSite {
   public:
      Room(int roomNo);
      MapSite* getSide(Direction) const;
      void setSide(Direction, MapSite*);
      void enter();
  // add your own data members
};
```

```
class Wall : public MapSite {
   public:
      Wall();
      void enter();
};
```

```
class Door : public MapSite {
   public:
      Door(Room* r1= nullptr, Room* r2= nullptr);
      void enter();
  // add your own data members
};
```

Implement this class hierarchy and allow their objects to support creating a maze consisting of two rooms with a door between them and three walls for each room:



And support the following `enter` behavior:

- If you try to enter a wall, you hurt your nose.
- If you try to enter a door, you go into another room.

Note that you will need to add proper data members in addition to the public interfaces for `Room` and `Door` class definition.

Create the two-room maze:

```
Room* r1 = new Room(1);
Room* r2 = new Room(2);
Door* theDoor = new Door(r1, r2);
Wall* w1r1 = new Wall;
Wall* w2r1 = new Wall;
Wall* w3r1 = new Wall;
Wall* w1r2 = new Wall;
Wall* w2r2 = new Wall;
Wall* w3r2 = new Wall;

r1->setSide(North, w1r1);
r1->setSide(East, theDoor);
r1->setSide(South, w2r1);
r1->setSide(West, w3r1);

r2->setSide(North, w1r2);
r2->setSide(East, w2r2);
r2->setSide(South, w3r2);
r2->setSide(West, theDoor);
```

The sample `enter` behavior:

```
r1->getSide(North)->enter();
// cout "Hit the wall and hurt your nose!"

r1->getSide(East)->enter(); // cout "Entering another room"
r1->enter();  // cout "You are in the room No.1."
```

A sample run looks like:

```
Hit the wall and hurt your nose!
Entering another room.
You are in the room No.1.
請按任意鍵繼續 . . .
```

2. **(40%)** Design an abstract base class `GeometricObject` and three derived classes named `Circle`, `Square` and `Triangle` that extends `GeometricObject`. Create a few `Circle`, `Square` and `Triangle` objects and put them into a `std::multiset` container with a sorting criterion based on their area. Use the following client code and sample runs

to test your program.

```
#include <iostream>
#include <set>
#include "GeometricObject.h"
#include "Square.h"
#include "Circle.h"
#include "Triangle.h"
#include "LessGeom.h"
using namespace std;
int main()
{
    GeometricObject* square1 = new Square(10);
    GeometricObject* circle1 = new Circle(5);
    GeometricObject* circle2 = new Circle(4);
    GeometricObject* triangle = new Triangle(3, 4, 5);
    GeometricObject* square2 = new Square(5);

    multiset< GeometricObject*, LessGeom > geos = {square1, circle1,
circle2, triangle, square2 };
    for (const auto& e : geos){
        cout << *e << " with area equals " << e->getArea() << endl;
    }
    delete square1;
    delete circle1;
    delete circle2;
    delete triangle;
    delete square2;
    return 0;
}
```

```
Triangle with area equals 6
Square with area equals 25
Circle with area equals 50.2655
Circle with area equals 78.5398
Square with area equals 100
請按任意鍵繼續 . . .
```

<u>HW Grading Policy:</u>

1. You should consider about exception handling, e.g. error input, file opening fail, etc.
   請注意所有例外狀況的處理，例如：錯誤的符號字串輸入、檔案開啟失敗等。
2. The coding style includes your output format.
   輸出資料的格式將納入格式評分。
3. If your code is not compilable, your score in this problem is zero (including coding style).
   若程式無法編譯，則該題以零分計算。(包含格式分數)
4. Your program will be tested with other data which is not the same as provided samples.
   除了題目所提供的範例測試資料以外，作業程式碼將以額外的測試資料進行測試。

- Coding Style (20%): 編碼格式分數
  1. format

整體形式與輸出資料的格式

2. comments
   註解
3. readability
   可讀性
4. variables naming
   變數命名方式
5. typesetting
   型別設定

- Functionality (80%): 功能性分數
  1. run-time performance:
     執行時的表現
     1) samples not passed -> x
        範例測資錯誤 => 此部分零分
     2) samples passed but some tests failed -> partial
        範例測資通過但是部分測資失敗 => 部份給分
     3) samples and tests all passed
        範例測資與所有測資通過 => 此部分滿分
  3. excellent method++
     綜合以上，又以能展現解決問題的巧思尤佳。