C++程序设计入门（上）

第五周编程作业

1.题目内容：（难度：中）

增强 Screen 类，使之在程序中只能生成一个实例

基于第四单元作业【2】或者作业【3】，只修改其中的 Screen 类
如果基于第四单元作业【3】修改，请将所有 MyRectangle 类的相关代码删掉，避免无意间的副作用

1）在 Screen 类中，添加两个 string 类型的数据成员 enter 和 leave，并在构造函数中将他们分别初始化为字符串“enter screen”和“leave screen”（每个字符串中只有一个空格分隔两个单词）
2）在 Screen 类中，增加一个 Screen* 类型的静态的私有数据成员 instance；
3）在 Screen 类中，增加一个静态公有的 getInstance(int width, int height) 函数，该函数返回 instance 的值。两个参数均带有默认值，分别为 640 和 480
4）在 getInstance 函数中，判断 instance 的值
A. 若 instance 值为 0（即“instance 是一个空指针”）
   a. 以 width 和 height 作为构造函数的参数，使用 new 运算符创建一个 Screen 对象
   b. 将新的 Screen 对象的地址保存在 instance 中；
B. 若 instance 的值不为 0（即 instance 指向一个 Screen 对象），则返回 instance 的值
5）在 getInstance 函数中，不检查宽和高是否合理。合理性检测仍然由 Screen 类的构造函数负责
6）修改 Screen 类的构造函数
a. 删除 Screen 类的默认构造函数，只保留带参构造函数，并修改之使其能初始化数据成员
b. 删除第 4 单元作业的 Screen 类的所有构造函数中的【cout << "screen" << endl;】语句
c. Screen 类的所有构造函数【均应输出】数据成员 enter 中的字符串内容并换行（使用 std::endl），但是【不再输出其它信息】
d. Screen 类的构造函数仍然使用第四单元作业中所要求的 exitWhenInvalidScreen 函数检查屏幕宽和高的有效性。（可以直接复用第四单元作业的相关代码）；该部分代码必须放在输出数据成员 enter 的代码之后
7）如有必要，则增加或者修改其他数据成员及函数成员
8）不要忘记在类外对 Screen 类的所有静态成员进行初始化，否则编译器会报告链接出错。

- **补充说明：现在的 Screen 类使用了一种【设计模式】，叫做“单例模式”，可以保证在这个程序中只会有一个 Screen 的实例。**
- **在本作业中，我们完成了单例模式的 70%。在下一个作业中将全部完成它**

程序所用主函数如下（不可做任何修改！）

```
int main() {
  int width, height;
```

```
  Screen *screen1, *screen2;

  std::cin >> width >> height;

  screen1 = Screen::getInstance(width, height);
  screen2 = Screen::getInstance();

  std::cout << screen1->getWidth() << ' ' << screen1->getHeight() << std::endl;
  std::cout << screen2->getWidth() << ' ' << screen2->getHeight();

#ifdef DEBUG
  std::cin.get();
#endif
  return 0;
}
```

输入格式:
空格分隔的整数，代表屏幕的宽和高

输出格式：
字符串或者空格分隔的整数

输入样例：
800 600

输出样例：
enter screen
800 600
800 600

注：上述输出样例第三行之后没有换行符

2.题目内容：（难度：易）
基于单元 5 的作业【1】，修改 Screen 类，完成单例模式练习

修改本单元作业【1】中的 Screen 类

1）为 Screen 类添加析构函数

a. 析构函数应首先输出数据成员 leave 的内容并换行（使用 std::endl）

b. 然后再执行其他必要的操作（如果有的话）

2）在 Screen 类中，添加一个 deleteInstance()函数

a. 函数类型自行根据 main()中的代码确定

b. 功能：将 getInstance()函数中申请的内存归还给操作系统。

c. 将数据成员 instance 设置为空指针

3）将 Screen 类中的所有构造函数都变成 **private** 成员

4）删除 Screen 类中的所有 setter 函数

5）保留屏幕宽高的合法性检测

程序的主函数如下

```
int main() {
  int width, height;
  Screen *screen1, *screen2;

  std::cin >> width >> height;

  screen1 = Screen::getInstance(width, height);
  screen2 = Screen::getInstance();
  if (screen1 != screen2 ) {
      std::cout << "two instances" << std::endl;
  }

  std::cout << screen2->getWidth() << ' ' << screen2->getHeight() << std::endl;
  screen2->deleteInstance();

  screen1 = Screen::getInstance(2*width, 2*height);
  std::cout << screen1->getWidth() << ' ' << screen1->getHeight() << std::endl;
  screen1->deleteInstance();
```

```
#ifdef DEBUG
  std::cin.get();
#endif
  return 0;
}
```

注意：要小心使用 delete 这个运算符，不要用错地方，否则你的程序可能会崩溃

输入格式:

空格分隔的两个整数，代表屏幕的宽和高

输出格式：

依据输入的数据不同，输出的内容、行数均有变化

输入样例：

```
480 320
```

输出样例：

```
enter screen
480 320
leave screen
enter screen
960 640
leave screen
```

注：一共有 7 行输出，第 7 行是空行

3.题目内容：（难度：易）
增强 Screen 类，使之在程序中只能生成一个实例
增强 MyRectangle 类，添加颜色信息
创建 MyCircle 类
在 main 函数中创建类的实例。

本作业的代码基于本单元作业【2】的代码改进，并且将第 4 单元作业【3】中的 MyRectangle 类代码添加到本单元作业中，同时修改 MyRectangle 类：

1）在 MyRectangle 类中，添加与颜色相关的属性：增加表示三种颜色分量（红色，绿色，蓝色）的三个数据域成员；

2）在 MyRectangle 类中，增加函数 setColor(int R, int G, int B)；该函数接收三个参数，代表颜色中的 Red、Green、Blue 分量的大小，该函数将颜色保存在类的数据域成员中。函数返回值自行定义

3）MyRectangle 类的**默认构造函数**将矩形左上角顶点的坐标均设置为**(10,10)**，将右下角顶点坐标设置为**(100,100)**

4）保留 MyRectangle 类的**带参构造函数**和**拷贝构造函数，**并对他们进行必要的修改

5）MyRectangle 类的**默认构造函数和带参构造函数**需要将表示颜色的数据成员初始化为白色，也就是 RGB 三个颜色分量的值均为 255；拷贝构造函数则不必

6）MyRectangle 类的**所有构造函数**的末尾均应使用 cout 输出**字符串“myrectangle”并换行**（使用 std::endl）

7）**删除** MyRectangle 类中**检查坐标有效性的代码，并删除调用这些代码的代码**（注意检查 Draw()函数中的代码）

8）Draw()函数中，**仅仅只包含如下代码：**

a. 用 **cout 输出**矩形的**左上顶点的 x、y 坐标**以及**矩形的宽度和高度**（坐标值以及宽高等 4 个数值间以 1 个空格分隔）然后换行（使用 std::endl）；

b. 然后**用 cout 输出**矩形颜色的 Red、Green、Blue 分量的整数值（用十进制输出），用单个空格分隔开这三个整数，然后换行（使用 std::endl）

9）如有必要，则增加其他数据成员及函数成员

10）不要输出任何未要求输出的信息，尤其是**不必要的“空格”**，不然会导致测试例无法通过。

main() 函数：

需使用如下 main()函数（不得更改）

```
int main() {
  int width, height;
  int leftX, leftY, rightX, rightY;
  Screen *screen;

  cin >> width >> height;
  cin >> leftX >> leftY >> rightX >> rightY;

  screen = Screen::getInstance(width, height);
  MyRectangle myRectangle(leftX, leftY, rightX, rightY, screen);
  myRectangle.setColor(0, 0, 0xff);
  myRectangle.Draw();

  screen->deleteInstance();

#ifdef DEBUG
  std::cin.get();
#endif
  return 0;
}
```

输入格式:

空格分隔的整数

输出格式:

字符串或者空格分隔的整数

输入样例:

640 320

100 101 200 301

输出样例：

enter screen

myrectangle

100 101 100 200

0 0 255

leave screen

注：输出样例公有 6 行，其中最后一行是空行

第 4 单元作业【3】中的 MyRectangle 类的参考代码

这部分代码【仅供参考】。若你没有完成/通过第 4 单元作业【3】，则可以参考该代码

```
class MyRectangle {
private:
    int x1_, y1_, x2_, y2_;
    Screen* screen_;

    int getWidth() {
        return x2_ - x1_;
    }

    int getHeight() {
        return y2_ - y1_;
    }

    bool isValid() {
        bool isPositive, isInScreen, isLeadingDiagonal;
        isPositive = (x1_ > 0) && (y1_ > 0) && (x2_ > 0) && (y2_ > 0);
        isInScreen = (x1_ < screen_->getWidth()) && (x2_ < screen_->getWidth()
```

```
) &&
                (y1_ < screen_->getHeight()) && (y2_ < screen_->getHeight());
    isLeadingDiagonal = (getWidth() > 0) && (getHeight() > 0);

    return (isPositive && isInScreen && isLeadingDiagonal);
  }

public:
  MyRectangle (int x1, int y1, int x2, int y2, Screen* screen) {
    x1_ = x1;
    y1_ = y1;
    x2_ = x2;
    y2_ = y2;
    screen_ = screen;

    cout << "myrectangle" << endl;
  }
   MyRectangle () {
    x1_ = y1_ = x2_ = y2_ = 0;
    screen_ = 0;

    cout << "myrectangle" << endl;
  }

  void setCoordinations(int x1, int y1, int x2, int y2) {
    x1_ = x1;
    y1_ = y1;
    x2_ = x2;
    y2_ = y2;
  }
```

```
    void setScreen(Screen& screen) {
        screen_ = &screen;
    }

    void Draw () {
        if (!isValid()) {
            cout << "invalid myrectangle" << endl;
        } else {
            cout << x1_ << " " << y1_ << " " <<
                    this->getWidth() << " " <<
                    this->getHeight() << endl;
        }
    }
};
```

4.题目内容：（难度：易）

在本单元作业【3】的基础上，修改 MyRectangle 类，并且增加新的 MyCircle 类表示圆形

练习编写拷贝构造函数

修改 MyRectangle 类：

1)增加公有 showScreen() 成员函数

a. 该函数通过使用类内的 screen_私有数据成员访问 MyRectangle 对象所保存的屏幕的信息

b. 输出屏幕的宽和高，以空格分隔，然后换行

新增 MyCircle 类：

1）在 MyCircle 类中，增加表示三种颜色分量（Red、Green、Blue）的数据域成员；

2）在 MyCircle 类中，增加函数 setColor(int R, int G, int B)；该函数接收三个参数，代表颜色中的 Red、Green、Blue 分量的大小，该函数将颜色保存在类的数据域成员中。

3）MyCircle 类中增加一个私有的指向 Screen 类型的指针成员 screen_

4）MyCircle 类的**构造函数 1 接受 3 个整型参数**和 **1 个 Screen*类型的参数**

a. 按照顺序，整型参数分别为圆心的 x、y 坐标，以及圆的半径。

b. 此处不检查坐标及半径的有效性

c. Screen*类型的参数指向一个已经存在的 Screen 对象，本构造函数将该 Screen 对象的地址存入私有数据成员 screen_ 中

5）MyCircle 类的**默认构造函数**将圆心的坐标设置为**(200,200)，半径设置为 100**

6）MyCircle 类的"**构造函数 1**"与默认构造函数均将表示颜色的数据成员初始化为白色，也就是 RGB 三个颜色分量的值均为 255

7）为 MyCircle 类添加拷贝构造函数

a. 在拷贝构造函数的**尾部添加输出语句，输出字符串"copy mycircle"并换行**

b. **思考：**该拷贝构造函数的参数，应该是 MyCircle& 类型，还是 const MyCircle& 类型？**这决定着本测试所给的主函数能否通过编译。**

c. **思考：**该拷贝构造函数是否需要"深拷贝"？

8）MyCircle 类的**所有非拷贝构造函数**均应**输出字符串"mycircle"并换行**

9）MyCircle 类中应提供 setCenter(int x, int y)用于设置圆心坐标，提供 setRadius(int r)用于设置圆的半径。函数的返回值自行确定。

10）在 Draw()中用输出:

a. 圆心的 x、y 坐标以及半径（坐标值以及半径等 3 个数值间以 1 个空格分隔）然后换行；

b. 圆的颜色的 RGB 分量的值，用空格分隔开的三个整数，然后换行

11）增加公有 showScreen() 成员函数 （与 MyRectangle 类的函数相同）

a. 该函数通过使用类内的 screen_私有数据成员访问 MyCircle 对象所保存的屏幕的信息

b. **输出屏幕的宽和高，以空格分隔，然后换行**

12）MyCircle 类中应提供 setScreen(Screen& screen)用于设置该类的实例所对应的 Screen 对象（同 MyRectangle 类中的代码）1）即，setScreen 函数会将引用参数 screen 这个对象的地址赋给 MyCircle 类中的私有成员 screen_2）要注意：私有成员 screen_ 是对象指针类型，而 setScreen()的形式参数 screen 是对象引用类型 3）所以，在 setScreen()函数体内， 要取 screen 这个参数的地址，再将该地址赋值给私有成员 screen_4）函数返回值类型由你自己决定

13）以上所有换行均请使用 std::endl

14）如有必要，则增加其他数据成员及函数成员

15）不要输出任何未要求输出的信息，尤其是**不必要的"空格"**，不然会导致测试例无法通过。

主函数如下（不可修改）：

```
int main() {
  int width, height;
  cin >> width >> height;

  int leftX, leftY, rightX, rightY;
  cin >> leftX >> leftY >> rightX >> rightY;

  int centerX, centerY, radius;
  cin >> centerX >> centerY >> radius;

  Screen *screen = Screen::getInstance(width, height);

  MyRectangle myRectangle(leftX, leftY, rightX, rightY, screen);
  myRectangle.setColor(0, 0, 0xff);
  myRectangle.showScreen();
  myRectangle.Draw();

  // 构造圆形对象数组
  //// 第一个元素使用匿名对象（调用带参构造函数）初始化
  //// 第二个元素使用匿名对象（调用默认构造函数）初始化
  MyCircle myCircles[2] = { MyCircle(centerX, centerY, radius, screen) };

  // 设置对象数组中第二个元素的属性。注意访问成员函数的不同方法
  (myCircles + 1)->setCenter(centerX+10, centerY+20);
  myCircles[1].setRadius(radius+30);
  (*(myCircles+1)).setColor(0x00, 0x00, 0x00);
  myCircles[1].setScreen(*screen);
  for(int i=0; i<=1; i++) {
      myCircles[i].showScreen();
      (myCircles+i)->Draw();
```

```
  }

  // 调用拷贝构造函数以 myCircles 数组中的第二个元素为模板创建新对象
  MyCircle yourCircle(myCircles[1]);

  yourCircle.showScreen();
  (&yourCircle)->Draw();

  screen->deleteInstance();

#ifdef DEBUG
  std::cin.get();
#endif
  return 0;
}
```

输入格式:

首先输入空格分隔的屏幕宽度和高度

然后输入空格分隔的矩形的坐标（四个整数），代表【左上--右下】对角线上的两个坐标点

最后输入圆形的圆心坐标和半径

输出格式：

见输出样例。

不同的输入会导致不同的输出信息

输入样例：

```
800 600
50 50 400 300
```

500 200 100

输出样例：

enter screen

myrectangle

800 600

50 50 350 250

0 0 255

mycircle

mycircle

800 600

500 200 100

255 255 255

800 600

510 220 130

0 0 0

copy mycircle

800 600

510 220 130

0 0 0

leave screen

注意：输出样例的最后一行为空行，不可遗漏

4.题目内容：（难度：难）

1）本次作业在第 5 单元作业【1】的基础之上，修改而来；

2）在 Screen 类的构造函数中调用图形库的 initgraph()；

3）在 Screen 类的析构函数中调用图形库的 closegraph() ；

4）在 MyRectangle 类的 Draw()函数中调用图形库的 rectangle()函数绘图；

5）在 MyCircle 类的 Draw()函数中调用图形库的 Circle()函数或者 fillellipse()函数绘图；

6)必要的地方，将原程序中的 cout 语句以图形库中的 outtextxy()或者 xyprintf()等函数代替；

7)必要的地方，将原程序中的 cin 语句以图形库中的 getInteger()、getCoords()等函数代替 （**注意：这两个函数只有使用 ege-13.04.02-full.zip 这个 ege 发行版才有！ 你从其它任何地方下载的图形库中都没有这些函数！**）

8)你可以继续增加其他绘图类，比如描述点的 MyPoint 类，描述线的 MyLine 类，描述弧形的 MyArc 类、描述颜色的 MyColor 类等

9)对原来的代码进行其他修改，使得程序可以正确运行，并根据输入的坐标等信息绘图。