面向对象方法程序实例
– 静态成员和多文件结构

【例2-21】设计一个立方体类，该类具有边长，能够设置立方体的边长，求立方体的体积。该类还能够记录和显示当前立方体的数量。要求：

①合理地设计属性和方法。

②考虑如何通过构造函数和析构函数维护对象数量的情况。

③合理地设计类成员的访问控制方式和静态特性。

④用主函数测试类。

⑤要求用多文件结构实现程序。

```
// Cube.h
class Cube
{
private:
    double m_x;                         // 边长
    static int numOfObject;             // 数量
public:
    Cube();                 // 无参构造函数
    ~Cube();                // 析构函数
    void set(double x); // 设置边长
    double getVolume();             // 求体积
    static void displayNumOfObject();       // 显示对象数量
};
```

```cpp
// Cube.cpp
#include "Cube.h"
#include <iostream>
using namespace std;
// 静态成员变量定义和初始化
int Cube::numOfObject = 0;
Cube::Cube()
{
    m_x = 0;
    numOfObject++;  // 创建了一个新对象
}
Cube::~Cube()
{
    numOfObject--;    // 销毁了一个对象
}
```

```cpp
void Cube::set(double x)
{
    m_x = x;
}
double Cube::getVolume()
{
    return m_x*m_x*m_x;
}
void Cube::displayNumOfObject()
{
    cout<<"对象数量为："<<numOfObject<<endl;
}
```

```cpp
// testCube.cpp
#include "Cube.h"
#include <iostream>
using namespace std;
int main()
{
    Cube cubeA, *pCubeB = NULL;
    pCubeB->displayNumOfObject();
    cout<<"cubeA的体积为："
        <<cubeA.getVolume()<<endl;
    cubeA.set(3);
    cout<<"cubeA的体积为："
        <<cubeA.getVolume()<<endl;
    pCubeB = new Cube();
    Cube::displayNumOfObject();
    delete pCubeB;
    cubeA.displayNumOfObject();
    return 0;
}
```

运行结果：

```
对象数量为：1
cubeA的体积为：0
cubeA的体积为：27
对象数量为：2
对象数量为：1
```