静态成员

静态成员是指被static修饰的成员变量或成员函数，在程序运行过程中只占一份内存，类似于全局变量，且也存储在全局区。

静态成员变量逻辑上属于类，可以通过类的权限控制静态成员的访问权限。

静态成员函数内部只能访问静态成员变量或函数，因为静态成员不依赖于对象的创建，所以也不可以通过this指针访问。如果未创建对象，调用静态成员函数里面访问了非静态函数或变量，逻辑上是行不通的。构造函数和析构函数也不可能是静态的。

对象计数器

静态成员变量的一个重要应用是统计一个类创建了多少对象。

计数器可以定义为静态成员变量，每创建一个对象，在构造函数中计算器+1，销毁一个对象，将计数器-1。

```cpp
#include <iostream>
using namespace std;

class Student {
private:
    int m_id;
    static int ms_count;
public:
    static int get_count() {
        return ms_count;
    }

    Student(int id = 0) : m_id(id) {
        ms_count++;
    }

    ~Student() {
        ms_count--;
    }
};

int Student::ms_count = 0;

int main() {

    Student* stu1 = new Student(101);

    cout << Student::get_count() << " " << stu1->get_count() << endl;

    Student* stu2 = new Student(102);
    cout << Student::get_count() << " " << stu1->get_count() << endl;

    delete stu2;

    cout << Student::get_count() << " " << stu1->get_count() << endl;
    return 0;
}
```

单例设计模式

在程序设计过程中，经常会有只能创建一个实例的需求。比如，一个系统中可以存在多个打印任务，但是只能有一个正在工作的任务。

单例设计模式可以借助static静态成员实现。为了防止随意创建或删除对象，私有化构造和析构函数，并使用类的私有静态指针变量指向类的唯一实例，使用一个共有的静态方法获取该实例。

```cpp
#include <iostream>
using namespace std;

class Student {
private:
    static int ms_id;
    static Student* ms_stu;
    Student(){}
    ~Student(){}
public:
    static Student* createStudent(int id) {
        if (ms_stu == NULL) {
            ms_stu = new Student();
            ms_id = id;
        }

        return ms_stu;
    }

    static void deleteStudent() {
        if (ms_stu != NULL) {
            delete ms_stu;
            ms_id = -1;
        }
    }

    static int getStudentId() {
        return ms_id;
    }
};

int Student::ms_id = -1;
Student* Student::ms_stu = NULL;

int main() {

    Student* stu = Student::createStudent(101);
    cout << stu->getStudentId() << endl;

    stu->deleteStudent();
    cout << stu->getStudentId() << endl;

    return 0;
}
```