类的静态成员函数

- 如果类的成员函数声明时被static修饰，它就是静态成员函数。类的公有静态成员函数的一般访问形式是：
- <类名>::<静态成员函数名>([实参])
- 也可以是：
- <对象名>.<静态成员函数名>([实参])
- 或
- <对象指针>-><静态成员函数名>

- 【例2-13】静态成员示例。将例2-12中的静态数据成员m_averageAge声明为私有成员，相应地声明一个静态成员函数float GetAverageAge()，为类外提供一个访问私有静态数据成员m_averageAge的接口。编写主函数测试Student类。

```
// Student.h
class Student
{
public:
    Student(char *pname, double age); //构造函数
    static double getAverageAge(); //静态成员函数
    static int getTotalNumber(); //静态成员函数
    void printStudentInfo (); //非静态成员函数
    ~Student();
private:
    static double m_averageAge;  //静态数据成员
    static int m_totalNumber; //学生总数
    char m_name[20];
    double m_age;
};
```

```cpp
// Student.cpp
#include "Student.h"
#include <string>
#include <iostream>
using namespace std;
double Student::m_averageAge=0;                 //定义静态成员
int Student::m_totalNumber=0;                   //定义静态成员
Student::Student(char *pname, double age)       //定义构造函数
{
    strcpy(m_name,pname);
    m_age=age;
    m_averageAge= m_totalNumber* m_averageAge+m_age;
    m_totalNumber++; // 创建一个对象，学生总数增
    m_averageAge=m_averageAge/m_totalNumber;
}
double Student::getAverageAge() //静态成员函数
{
    return m_averageAge;
}
```

```
int Student::getTotalNumber()        //定义静态成员函数
{
    return m_totalNumber;
}
void Student::printStudentInfo()     //定义非静态成员函数
{
    cout<<"学生的姓名为:"<<m_name<<endl;
    cout<<"学生的年龄为"<<m_age<<endl;
}
Student::~Student()
{
    m_averageAge= m_totalNumber* m_averageAge-m_age;
    m_totalNumber--;
    m_averageAge=m_averageAge/m_totalNumber;
}
```

```cpp
// testStudent.cpp
#include <iostream>
#include "Student.h"
using namespace std;
int main()
{
    Student student1("刘丹",19);
    Student student2("陈刚",17);
    Student *ps=&student2;
    student1.printStudentInfo();
    ps->printStudentInfo();
    cout<<"当前对象的数量为："<<Student::getTotalNumber()<<endl;
    cout<<"当前学生的平均年龄为："
        <<student1.getAverageAge()<<endl;
    return 0;
}
```

- 提示：
- 非静态成员函数即可以访问静态数据成员，也可以访问非静态数据成员。

  但静态成员函数只能访问静态数据成员，而不能访问非静态数据成员。