static variable

```
//静态成员变量实际上是属于整个类的，并不属于obj
//所以说所有的obj都可以共享这个成员变量
//关于这个变量的定义一定是在类的外部，用 类型名 类名::静态成员变量 = 赋值 来实现
class Student {
public:
    char name;
    int score;
    static int total_number;
    static int average_score;
    Student() {
        //init
    }
    ~Student();

};
int Student::total_number = 1000;
int Student::average_score = 90;
```

想要使用这个静态成员变量，首先在类内部定义的时候就说好，在前面加上static关键字

之后定义一定是在类的外部，也就是赋值操作一定在类的外部

`int Class::variable_name = ...`

类型名 类名::静态成员变量 = 赋值

static function

```
//静态成员函数只能操作静态成员变量
//非静态成员函数可以操作两种成员变量
class MyClass {
private:
    static int privateStaticVar; // 私有静态成员变量

public:
    static int publicStaticVar; // 公开静态成员变量

    static int getPrivateStaticVar() {
        return privateStaticVar; // 提供公开接口访问私有静态成员变量
    }
};

// 在类外初始化静态成员变量
int MyClass::privateStaticVar = 1;
int MyClass::publicStaticVar = 2;

int main() {
    MyClass obj1;
    // 正确：通过类名访问公开的静态成员变量
    std::cout << obj1.publicStaticVar << std::endl;//两种方法都可以访问到
```

```
    std::cout << MyClass::publicStaticVar << std::endl;

    // 错误：试图直接访问私有的静态成员变量
    // std::cout << MyClass::privateStaticVar << std::endl; // 编译错误

    // 正确：通过类提供的公开接口访问私有的静态成员变量，与正常思路一样
    std::cout << MyClass::getPrivateStaticVar() << std::endl;

    return 0;
}
//这里一定要注意静态成员变量不仅可以通过对象来访问，也可以直接用类直接访问
```

静态成员函数只能访问、操作静态成员变量，不能访问对象非静态成员变量，因为后者需要实例化

非静态成员函数可以操作两种成员变量

要注意静态成员函数和静态成员变量可以通过类直接访问到，不用通过实例化对象来访问到，如果用类来访问的话，要记得在类的后面加上::域操作