```
class MyClass {
    friend class FriendClass;  // FriendClass 可以访问 MyClass 的所有成员
    friend void friendFunction(MyClass& obj);  // friendFunction 可以访问 MyClass
的所有成员
    //在类的内部定义
private:
    int value;

public:
    MyClass(int val) : value(val) {}
};

class FriendClass {
public:
    void modify(MyClass& obj) {
        obj.value = 10;  // 可以直接访问 MyClass 的私有成员
    }
};

void friendFunction(MyClass& obj) {
    obj.value = 20;  // 同样可以直接访问 MyClass 的私有成员
}
```

这个friend关键字一般定义在类中，可以看作自己，在自己内部定义哪些类和外部函数是朋友，加在定义的开始

那么一旦一个类或者函数有friend属性，则代表其可以随便访问这个类的所有变量，包括私有成员变量，不需要get函数之类的操作来传递值

如果出现以下的情况：

其实很多类中都有同名函数，但是如果你只想要一个类中的同名函数有权限的话，就把两个表达组合起来

```
//Friend member functions 这个类似于其实很多类中都有同名函数，但是如果你只想要一个类中的同名
函数有权限的话，就把两个表达组合起来
//Declaration : friend <return type> <class name>::<func name> (parameters);
friend void Friendclass::friendfunction(MyClass& obj);
```

声明在类内部的函数，定义在类外部的函数

本质上来说还是属于这个类的成员函数，只是避免类内部过于陇拶

```
class A{
    private:
    int a;
    public:
    void externalfunction();
};

void A::externalfunction(){
    this->a =10;
}
```

一定要注意了，这个重载函数实际上是属于这个类的

要和上面的友元函数分开来

友元函数本质上来说不是属于这个类，需要传入类对象的引用，要注意