小码哥教育 SEEMYGO

友元

- 友元包括友元函数和友元类
- 如果将函数A（非成员函数）声明为类C的友元函数，那么函数A就能直接访问类C对象的所有成员
- 如果将类A声明为类C的友元类，那么类A的所有成员函数都能直接访问类C对象的所有成员
- 友元破坏了面向对象的封装性，但在某些频繁访问成员变量的地方可以提高性能

```
class Point {
    friend Point add(const Point &, const Point &);
    friend class Math;
private:
    int m_x;
    int m_y;
public:
    Point() { }
    Point(int x, int y) :m_x(x), m_y(y) { }
};
```

```
Point add(const Point &p1, const Point &p2) {
    return Point(p1.m_x + p2.m_x, p1.m_y + p2.m_y);
}

class Math {
    void test() {
        Point point;
        point.m_x = 10;
        point.m_y = 20;
    }

    static void test2() {
        Point point;
        point.m_x = 10;
        point.m_y = 20;
    }
};
```

内部类

- 如果将类A定义在类C的内部，那么类A就是一个内部类（嵌套类）
- 内部类的特点
  - 支持public、protected、private权限
  - 成员函数可以直接访问其外部类对象的所有成员（反过来则不行）
  - 成员函数可以直接不带类名、对象名访问其外部类的static成员
  - 不会影响外部类的内存布局
  - 可以在外部类内部声明，在外部类外面进行定义

```
class Point {
    static void test1() {
        cout << "Point::test1()" << endl;
    }
    static int ms_test2;
    int m_x;
    int m_y;
public:
    class Math {
    public:
        void test3() {
            cout << "Point::Math::test3()" << endl;
            test1();
            ms_test2 = 10;

            Point point;
            point.m_x = 10;
            point.m_y = 20;
        }
    };
};
```

小码哥教育 SEEMYGO

内部类 – 声明和实现分离

```
class Point {
    class Math {
        void test();
    };
};

void Point::Math::test() {

}
```

```
class Point {
    class Math;
};

class Point::Math {
    void test() {

    }
};
```

```
class Point {
    class Math;
};

class Point::Math {
    void test();
};

void Point::Math::test() {

}
```

局部类

- 在一个函数内部定义的类，称为局部类
- 局部类的特点
  - 作用域仅限于所在的函数内部
  - 其所有的成员必须定义在类内部，不允许定义static成员变量
  - 成员函数不能直接访问函数的局部变量（static变量除外）

```
int m_age1 = 0;

void test() {
    static int s_age2 = 0;
    int age3 = 0;

    class Point {
        int m_x;
        int m_y;
    public:
        static void display() {
            m_age1 = 10;
            s_age2 = 20;
            age3 = 30;
        }
    };

    Point::display();
}
```