学号 E21714049 专业 计算机科学与技术 姓名 梅世祺
实验日期 2019.04.11 教师签字 成绩

实验报告

【实验名称】 数据的共享与保护

【实验目的】

1.学习类的友元函数的定义与调用，理解友元函数的意义。

2.学习友元类的声明和意义，在友元函数的基础之上理解友元类的使用。

3.学习静态数据成员的使用。

【实验原理】

1. 类的友元函数可以直接访问该类对象的私有成员；

2. 类 A 的友元类 B 的对象可以直接访问类 A 的私有成员。

【实验内容】

实验一 两圆圆心距离计算及位置关系判断

题目：定义一个表示圆的类 Circle，包含 x,y,r 三个私有变量，分别为圆心 x 坐标，圆心 y 坐标和圆半径。声明 Circle 类的两个友元函数 distance 和 relation，分别计算两圆圆心位置和判断两圆位置关系。

要求：1.两圆的参数 x,y,r 从命令行读入；

2. 程序具有健壮性；

3. 测试五种位置关系。

原理：利用圆心距与两圆半径之间的关系来判断两圆的位置关系。假设 dc 为圆心距,R 与 r 分别是两圆的半径，则：

（1）dc ＞ R + r，两圆外离；

（2）dc = R + r，两圆外切；

（3）|R - r| ＜ dc ＜ R + r，两圆相交；

（4）dc = |R - r|，两圆内切；

（5）0 <= dc ＜ |R - r|，两圆内含。

注意：浮点数之间做比较考虑精度问题。

实验结果（含源码）：

实验一 两圆圆心距离计算及位置关系判断

```
class Circle
{
private:
    double x, y, r;
public:
    Circle(double x, double y, double r) : x(x), y(y), r(r) {
        if (r<=0) {
            cout<<"\033[31m[Error]\033[0m illegal radius "<<r<<"! Please input a positive radius."<<endl;
            exit(1);
        }
    };
    // calculate distance between two points
    friend double distance(Circle c1, Circle c2);
    // determine the position relationship between two circle.
    friend int relation(Circle c1, Circle c2);
};
```

Circle 类的实现，包括 3 个私有变量 x, y, r 和两个友元函数 distance 和 relation。
Note: 在构造函数中，增加了对半径是否大于 0 的判断，以增强程序的容错性。

```cpp
// return true if the two numbers are approximately equal
bool aequal(double x, double y, double precision=0.0001) {
    if(abs(x - y) <= precision) {
        return true;
    } else {
        return false;
    }
}
```

考虑到浮点数之间的相等很难做到绝对的相等，若两个浮点数在一定精度内的相等，我们则认为两个数字相等。这里单独设计了一个 aequal 函数，用于判断两个浮点数是否近似相等。

```cpp
// calculate distance between two points
double distance(Circle c1, Circle c2) {
    double x1 = c1.x, y1 = c1.y;
    double x2 = c2.x, y2 = c2.y;
    double result = (x1-x2)*(x1-x2) + (y1-y2)*(y1-y2);

    if (result > 0) {
        return sqrt(result);
    } else {
        return 0;
    }
}
```

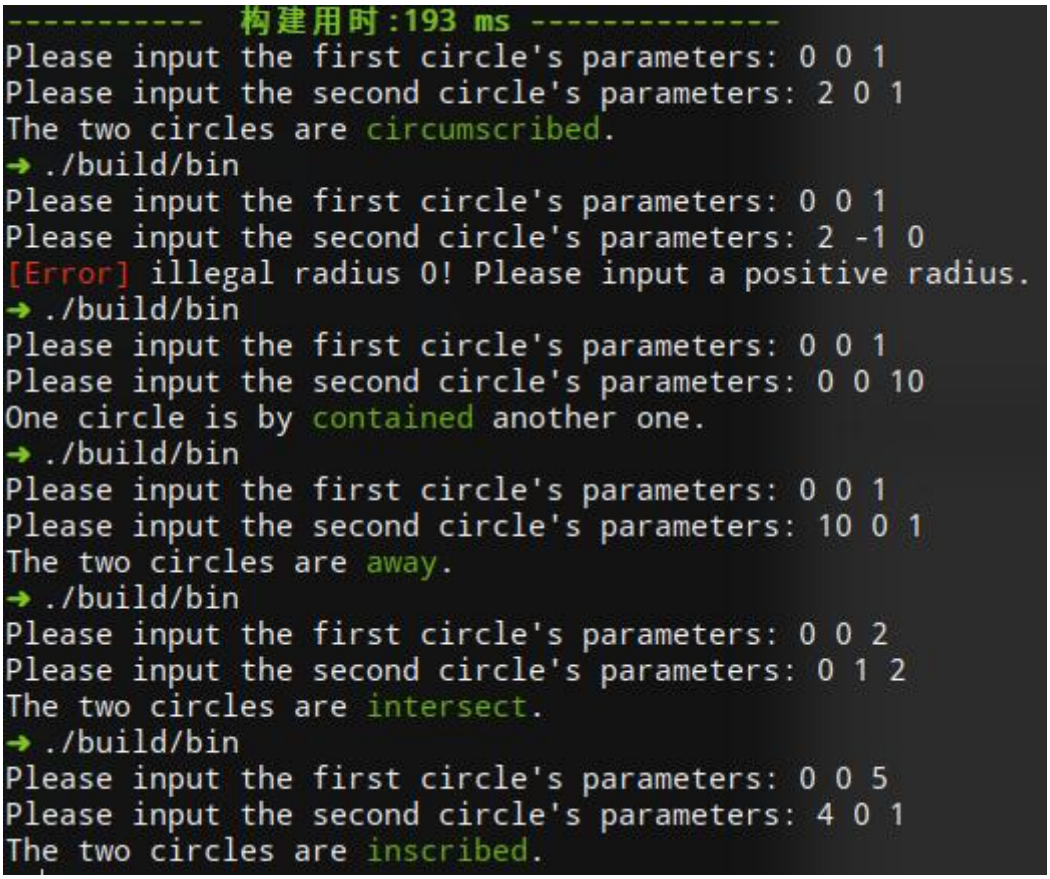
distance 函数，用于计算两个点之间的距离。

```
// determine the position relationship between two circle.
int relation(Circle c1, Circle c2) {
    double dc = distance(c1, c2);
    double R = c1.r;
    double r = c2.r;

    if (dc > R+r) {
        cout<<"The two circles are \033[32maway\033[0m."<<endl;
        return 4; // two circles away
    } else if (aequal(dc, R+r)) {
        cout<<"The two circles are \033[32mcircumscribed\033[0m."<<endl;
        return 3; // the two circles are circumscribed
    } else if (dc > abs(R-r) && dc < R+r) {
        cout<<"The two circles are \033[32mintersect\033[0m."<<endl;
        return 2; // the two circles intersect
    } else if (aequal(dc, abs(R-r))) {
        cout<<"The two circles are \033[32minscribed\033[0m."<<endl;
        return 1; // two circles inscribed
    } else {
        cout<<"One circle is by \033[32mcontained\033[0m another one."<<endl;
        return 0; // one circle is contained by another one
    }
}
```

releation 函数，用于判断两个圆之间的关系。

```
int main(int argc, char const *argv[])
{
    double x1, x2, y1, y2, r1, r2;

    cout<<"Please input the first circle's parameters: ";
    cin>>x1>>y1>>r1;
    Circle c1(x1, y1, r1);
    cout<<"Please input the second circle's parameters: ";
    cin>>x2>>y2>>r2;
    Circle c2(x2, y2, r2);
    relation(c1, c2);
    return 0;
}
```

主函数实现



程序运行结果，分别展示了内含、内切、相交、外切和外离 5 种两圆之间的关系。其次还演示了当输入圆的半径小于等于 0 时程序的行为，体现了程序的健壮性。

实验二 图书借阅

题目：定义一个图书类 Book 和一个读者类 Reader，实现读者借阅图书，图书能记录当前总馆藏记录数，剩余馆藏数和总借阅次数。

要求： 1. 程序具有健壮性，例如剩余馆藏数不足的时候不能借阅；

2. 多次添加（创建新的图书类）、借阅图书（调用读者类的 borrow 方法），并依次打印出当前总馆藏记录数，剩余馆藏数和总借阅次数等结果。

原理：图书类使用静态数据成员（私有）保存当前总馆藏记录数，剩余馆藏数和总借阅次数；读者类为图书类的友元类。

实验结果（含源码）：

实验一图书借阅

```
class Book {
    public:
        Book(string name) : name(name), borrowed(false) {
            total++;
            remaining++;
        };
        // show total collection records, remaining collections and total borrowings
        void display();
        // borrow books
        friend class Reader;
    private:
        static int total, remaining, borrowings;
        string name; // the book name
        bool borrowed;
};
```

Book 类定义，包括 1 个带参数的构造函数、一个公开方法 display 和 3 个静态类成员 total, remaining 和 borrowings（分别表示图书馆的总藏书量、剩余藏书量和被借走的书籍的数目），还有两个私有成员 name 和 borrowed（分别表示该书籍的名称和当前的状态）。

其次，还声明了一个友元类 Reader。

```
class Reader {
    public:
        // constructor
        Reader(string name) : name(name) {};
        // borrow books from the library
        int borrow(Book* book);
    private:
        string name;
};
```

Reader 类定义，包括 1 个带参数的构造函数、一个公开方法 borrow 和一个私有成员 name（表示读者的姓名）。

下面分别介绍 Book 类和 Reader 类的相关成员函数的实现细节：

```
void Book::display() {
    cout<<"total: "<<total<<" remaining: "<<remaining<<" borrowings: "<<borrowings<<endl;
}
```

Book::display 的实现，直接使用 cout 打印相关变量的值即可。

```
int Reader::borrow(Book* book) {
    if (book->borrowed) {
        cout<<"Sorry, this book is borrowed!"<<endl;
        return 1;
    }

    if (Book::remaining <= 0) {
        cout<<"Sorry, the library is empty."<<endl;
        return 2;
    }

    cout<<name<<" borrowed the book \""<<book->name<<"\"."<<endl;

    Book::remaining--;
    Book::borrowings++;
    book->borrowed = true;

    return 0;
}
```

Reader::borrow 的实现，在外借之前判断该书是否已被借走或者图书馆藏书数量小于等于 0，如果是则拒绝请求；否则外借图书，调整 Book::remaining、Book::borrowings 和 book->borrowed 的值。

```cpp
int Book::total = 0;
int Book::remaining = 0;
int Book::borrowings = 0;

using namespace std;

int main()
{
    Book* b1 = new Book("Understanding ECMAScript 6");
    b1->display();
    Book* b2 = new Book("You don't know JavaScript");
    b2->display();
    Reader r = Reader("Mike");
    r.borrow(b1);
    b1->display();
    r.borrow(b1);
    b1->display();
    r.borrow(b2);
    b1->display();
}
```

静态类成员变量的初始化（表示图书管的初始状态）和主函数的代码实现。

```
---------- 构建用时:2100 ms --------------
total: 1 remaining: 1 borrowings: 0
total: 2 remaining: 2 borrowings: 0
Mike borrowed the book "Understanding ECMAScript 6"
total: 2 remaining: 1 borrowings: 1
Sorry, this book is borrowed!
total: 2 remaining: 1 borrowings: 1
Mike borrowed the book "You don't know JavaScript".
total: 2 remaining: 0 borrowings: 2
```

程序运行结果：测试了对已借走书再借的情形；测试了图书馆书籍数目为空后再借的情形。