C++ Programming

Chapter 8 Objects and Classes

Part 3/4

Zheng Guibin
（郑贵滨）

哈爾濱工業大學
Harbin Institute of Technology

Objects and Classes

对象和类

- 什么是对象、类？
- 类的定义
- 类的使用
- 构造函数
- 类的接口、类的实现
- 析构函数（补充）
- const成员函数（补充）
- **友元(补充)**

友元

- 特点:
  - 友元可以访问与其有好友关系的类中的所有成员包括私有成员。
- 意义:
  - 提高访问效率增加灵活性，程序员可以在封装和快速性方面做合理选择。
  - 为了确保数据的完整性，及数据封装与隐藏的原则，建议尽量不使用或少使用友元。
- 友元分类
  - 友元函数
  - 友元类

友元函数

- 友元函数是定义在类外部的一个函数，但可访问类的私有成员变量
- 使用规则：
  - 友元函数在类内声明，并在函数的类型说明符前加 friend。

friend <类型> <友元函数名> (<参数表>);

  - 定义在类体外，定义格式和普通函数相同。
  - 它是非成员函数，调用与普通函数相同。
  - 它可以直接访问该类中的私有成员
- 注意：
  - 访问对象中的成员必须通过对象名。
  - 虽然类中有友元函数的原型，但是友元仍然不是成员函数。

友元函数

```
class  A
   { private:
          int  num ;
     public:
        void MemberFun(int) ;
        friend void FriendFun(A * , int) ;
   } ;
void FriendFun( A * ptr , int x  )
{  ptr -> num = x ;   }
void A:: MemberFun( int x )
 {  num = x ;          }
```

```
void A:: MemberFun(  int x )
 {  this-> num  =  x ; }
```

说明语句位置
与访问描述无关

友员函数通过对象参数访问私有数据成员

成员函数通过this指针在对象上操作

友元函数

```
class  A
   { private:
         int  num ;
     public:
        void MemberFun(int) ;
        friend void FriendFun(A & , int) ;
   } ;
void FriendFun( A & a ,    int x  )
{  a. num = x ;          }
void A:: MemberFun( int x )
 {  num = x ;            }
```

友员函数通过对象参数访问私有数据成员

```
void A:: MemberFun(  int x )
{  (*this). num = x ;}
```

成员函数通过this指针在对象上操作

友元函数

```
class Time
{public:
  Time(int,int,int);

 private:
          int hour, minute, sec;
friend void display(const Time &);
};

Time::Time(int h, int m, int s)
{  hour=h;      minute=m;   sec=s; }

void display(  const Time& t )
{   cout<<t.hour<<":"<<t.minute<<":"<<t.sec<<endl; }
```

```
#include <iostream.h>
int main( )
{
  Time t1(10,13,56);
  display(t1);
  return 0;
}
```

友元成员函数

```
// 友元成员函数
#include <iostream.h>
class Date;
class Time
{ public:
            Time(int,int,int);
            void display(Date &);
  private:
            int hour,minute, sec;
};
class Date
{  public:
            Date(int,int,int);
            friend void Time::display(Date &);
  private:
      int month, day, year;
};
```

对Date类的提前引用声明

声明Time中的display函数为友元成员函数

友元成员函数

```
Time::Time(int h,int m,int s)    //类Time的构造函数
{   hour=h; minute=m; sec=s;  }

void Time::display(Date &d)
{   cout<<d.month<<"/"<<d.day<<"/"<<d.year<<endl;
    cout<<hour<<":"<<minute<<":"<<sec<<endl;    }

Date::Date(int m,int d,int y)          //类Date
{   month=m;  day=d;  year=y;  }

int main( )
{   Time t1(10,13,56);          //定义Time类对象t1
    Date d1(12,25,2004);        //定义Date类对象d1
    t1.display(d1);
    return 0;}
```

引用Date类对象中的私有数据

引用本类对象中的私有数据

调用t1中的display函数，实参是Date类对象d1

友元类

- 当一个类A作为另一类B的友元类时，类A的所有成员函数都是类B的友元函数。
- 友元类定义格式

  在类体内声明

  friend class 类名;

```
class Time
{
public:
  Time(int,int,int);
  void display (Date &);
private:
  int hour, minute, sec;
};
```

```
class Date
{ public:
    Date(int,int,int);
    void show( );
    friend   class Time;
  private:
   int month; int day;
   int year;
};
```

友元类

- 友元关系不具有传递性。
  - 类A是类B的友元，类B是类C的友元，但并不表示A是C的友元。

- 友元关系不具有交换性。
  - 类A是类B的友元，但并不表示类B是类A的友元。

练习

设计描述位置的一个点类Point，其属性为其x,y坐标，包括设置坐标，显示位置等方法

要求:

定义构造函数，复制构造函数;

定义静态成员变量total,存放当前点的数目。

定义求两点距离的友元函数distance。

```
void main()
{    Point  p1 ,  p2( 4.0, 6.0 ) ;
      Point  p3 = p2;
     double  d = distance ( p1, p2 ) ;
     cout << "This distance is " << d << endl ;
}
```

```
// 普通函数作友元函数:计算两点之间的距离
class Point
{ public:
     Point(double xi = 0, double yi = 0) ;
     setXY(double xx, double yy) ;
     double getX( ) ;
     double getY( ) ;
     void show( );
     friend double distance ( Point & a, Point & b ) ;
  private:
    double X, Y ;
    static int    total;
} ;
```

```
// 普通函数作友元函数:计算两点之间的距离
  double distance(Point & a, Point & b )
{   double dx = a.X - b.X ;
    double dy = a.Y - b.Y ;
    return sqrt ( dx * dx + dy * dy ) ;
 }

void main()
{
  Point  p1 ,  p2( 4.0, 6.0 ) ;
  double  d = distance ( p1, p2 ) ;
  cout << "This distance is " << d << endl ;
}
```

练习

1.若需要把一个函数“void F( );”定义为一个类AB的友元函数，则应在类AB的定义
中加入一条语句：_____。

2.以下关于友元函数的叙述不正确的是（ ）

A. 友元函数提供了类之间数据共享的一个途径

B. 一个友元函数可以访问任何类的任何成员

C. 友元函数破坏了数据的封装

D. 友元函数增加了程序设计的灵活性

3.友元函数的作用是（ ）

A.提高程序的效率 B.加强类的封装性

C.实现数据的隐蔽性 D.增加成员函数的种类

4.友元类中的所有成员函数可以对该类的私有成员进行存取操作.

小结

- 类通常用class定义。类是数据成员和成员函数的封装。
- 类成员由private, protected, public决定访问特性。
- 类的实例称为对象, 对象调用成员函数。
- 构造函数在创建和初始化对象时自动调用。析构函数则在对 象作用域结束时自动调用。
- 重载构造函数和拷贝构造函数提供了创建对象的不同初始化方式。
- 静态成员属于整个类，提供一种同类对象的共享机制。
- 一个类的友元可以访问该类各种性质的成员。