河北师范大学软件学院
Software College of Hebei Normal University

C++

第八讲 类和对象（四）

C++备课组 丁盟

自我介绍

丁盟

qq：2622885094

上一讲教学目标

- 掌握C++中静态成员的使用
- 掌握C++中对象成员的使用
- 掌握C++中const与类的结合使用

本讲教学目标

- 了解友元的概念
- 掌握友元函数的使用
- 掌握友元类的使用

1 关于友元

- 问题的产生
- 友元的分类

2 友元函数

- 类的提前声明
- 友元非成员函数
- 友元成员函数

3 友元类

问题的产生

- 为何引入友元？

通常一个类中的非公有成员是无法被该类的非成员函数直接访问的！

如果我们将函数声明为这个类的友元，该函数将能访问该类的非公有成员！

1 关于友元

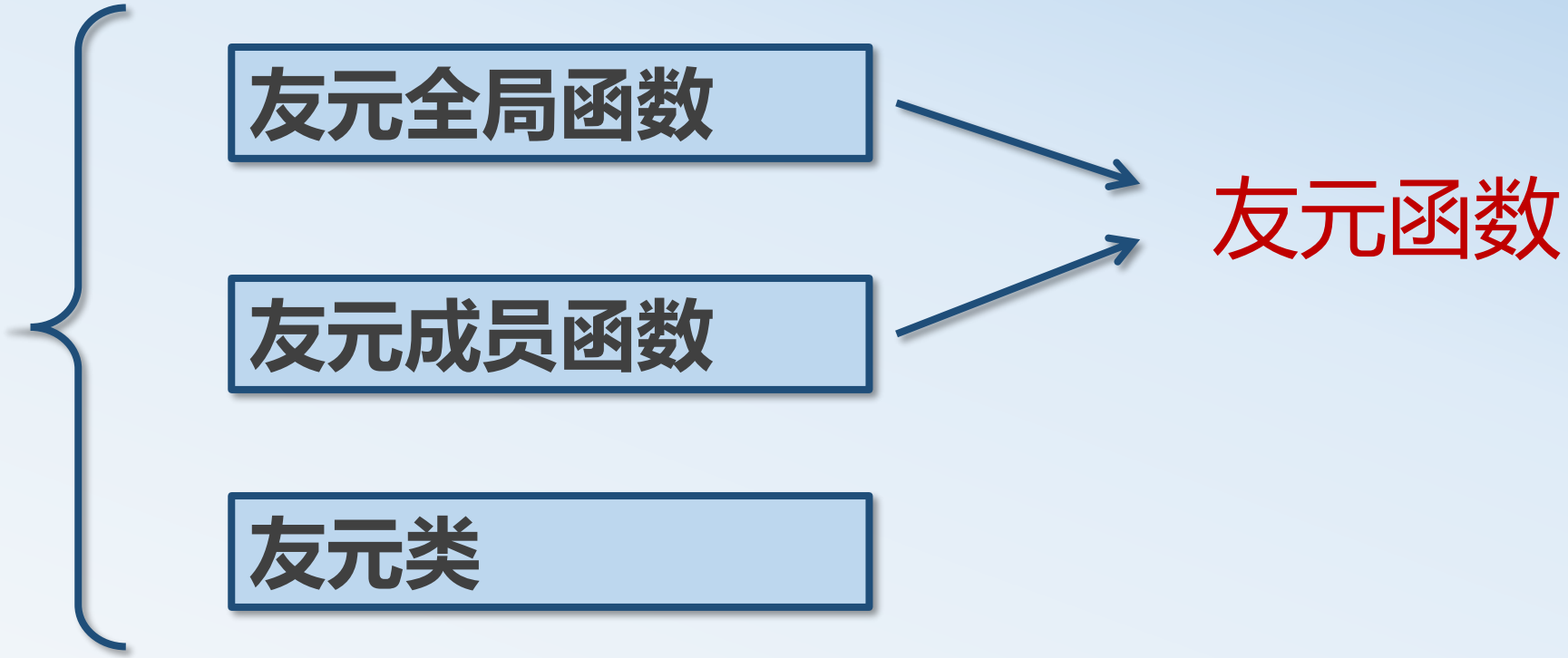
- 问题的产生
- 友元的分类

2 友元函数

- 类的提前声明
- 友元非成员函数
- 友元成员函数

3 友元类

友元的分类



1 关于友元

- 问题的产生
- 友元的分类

2 友元函数

- 类的提前声明
- 友元非成员函数
- 友元成员函数

3 友元类

类的提前声明

- 回忆函数的原型声明

```
void test();  // 函数提前声明
int main(void)
{
    test();
    return 0;
}
void test()
{
    cout << "函数的提前声明！" << endl;
}
```

类的提前声明

- 类和函数也是类似的,有时也需要提前声明

```
class B;                    // 类提前声明
void ff(B &){}              // 可以声明引用
int main(void)
{
    B * p = NULL; // 可以定义指针
    B b;                    // ERROR 不能定义对象
    return 0;
}
class B {};
```

1 关于友元

- 问题的产生
- 友元的分类

2 友元函数

- 类的提前声明
- 友元非成员函数
- 友元成员函数

3 友元类

友元非成员函数

- 概念：

  将类外的函数（顶层函数）声明为友元函数

- 声明方法：

  在类中用friend声明函数！

```
friend 类型 函数名(形参表);
```

友元非成员函数

```
class Time {
public:
    Time(int aH = 0,
         int aM=0, int aS=0);
    friend void display(Time &);
private:
    int m_iHour;
    int m_iMinute;
    int m_iSecond;
};
Time::Time(int aH, int aM,
           int aS) {
    m_iHour = aH;
    m_iMinute = aM;
    m_iSecond = aS;
}
```

```
void display(Time &t)
{
    cout<< t.m_iHour
        << ":"
        << t.m_iMinute
        << ":"
        << t.m_iSecond
        << endl;
}

int main(void)
{
    Time t(12, 30, 55);
    display(t);

    return 0;
}
```

友元非成员函数

[注意]

1. 友元函数应该声明在类体内，且friend仅在声明时使用，只出现一次。
2. 调用方式和普通全局函数一样。
3. 友元全局函数的声明语句就是函数的声明语句。

友元非成员函数

4. 友元全局函数也可以在类内定义，此时编译环境仍将其视为全局函数，且定义就相当于声明而不需要提前声明。
5. 一个全局函数可以被声明为多个类的友元。
6. 多个全局函数可以被声明为一个类的友元。

1 关于友元

- 问题的产生
- 友元的分类

2 友元函数

- 类的提前声明
- 友元非成员函数
- 友元成员函数

3 友元类

友元成员函数

- 概念：

  一个类的成员函数可以被声明为另一个类的友元。

- 声明方法：

  在类中用friend声明函数！

```
friend 类型 类名::函数名(形参表);
```

必不可少

友元成员函数

```
class Atest;  // 提前声明
class Btest {
public:
    Btest(int aX = 4):m_dVal(aX){}
    void display(Atest &aVal);
private:
    double m_dVal;
};
class Atest {
public:
    Atest(int aX = 3):ival(aX){}
    friend void Btest
        ::display(Atest &aVal);
private:
    int m_iVal;
};
```

```
void Btest::display(Atest &aVal)
{
    cout << "Atest.m_iVal = "
        << aVal.m_iVal << endl;
    cout << "Btest.m_dVal = "
        << m_dVal << endl;
}

int main(void)
{
    Atest testa;
    Btest testb;
    testb.display(testa);

    return 0;
}
```

友元成员函数

[思考]

类A放在类B的前面是否就不用提前引用了呢？

[结论]

一定先定义友元成员函数所在的类。

由于C++没有提供提前声明类成员的方法，而类的提前声明只能用于定义指针或声明引用形参！

因此如果用到类成员，则必须给出类的定义。

友元成员函数

[注意]

定义友元成员函数时，一定要注意友元函数的位置同时也要注意两个类的位置关系！

1 关于友元

- 问题的产生
- 友元的分类

2 友元函数

- 类的提前声明
- 友元非成员函数
- 友元成员函数

3 友元类

友元类

- 为什么引入友元类？

如果一个类A中的成员函数都希望访问类B的非公有成员，我们可以将A声明为B类的友元类

友元类

❖ 声明方法：

```
friend 类名；
```

```
class Date; // 切记提前引用
class Person
{
    friend Date;
};
```

```
class Date
{
    ...
};
```

在Person类中的位置任意

友元类

[注意]

1. 友元不具备传递性！
2. 友元关系单向的！
3. 友元关系破坏了信息隐蔽的原则，但友元关系也提供了数据共享的一种方法。
4. 友元不受访问限定符限制。

内容小节

1. 如果一个函数希望访问某个类的私有成员则需将这个函数声明为类的友元。
2. friend只能在声明友元时出现一次！
3. 声明友元函数或友元类一定注意提前声明。
   - 若用到成员，必须在此前给出类的定义。
   - 函数中只是用到类名，则只需给出提前引用。
   - 提前引用后只能定义类的指针或声明引用形参。

本讲教学目标

- 了解友元的概念
- 掌握友元函数的使用
- 掌握友元类的使用


THANKS