



西北工业大学

NORTHWESTERN POLYTECHNICAL UNIVERSITY

C++程序设计

Programming in C++



1011018

主讲：魏英，计算机学院

对象的定义和使用

- ◆ 3、对象成员的引用.....●
- ◆ 4、对象的赋值.....●
- ◆ 5、对象、对象指针或对象引用作为函数的参数和返回值.....●

- ▶ 访问对象中的成员可以有3种方法：
 - ▶ ①通过对象名和对象成员引用运算符（.）访问对象中的成员；
 - ▶ ②通过指向对象的指针和指针成员引用运算符（->）访问对象中的成员；
 - ▶ ③通过对象的引用变量和对象成员引用运算符（.）访问对象中的成员；

- ▶ 1. 通过对象名访问对象中的成员

- ▶ 访问对象中数据成员的一般形式为：

对象名.成员名

- ▶ 调用对象中成员函数的一般形式为：

对象名.成员函数（实参列表）

- ▶ 需要注意，从类外部只能访问类公有的成员。

- ▶ 例如已定义一个类：

```
class Data { //Data类
public:
    int data;
    void fun(int a,int b,int d);
private:
    void add(int m) { data+=m; }
    int x,y;
    .....
};
```

▶ 则函数

```
void caller1()
{
    Data A, B; //定义对象
    A.fun(1,2,3); //正确, 类外部可以访问类的public成员函数
    A.data=100; //正确, 类外部可以访问类的public数据成员
    A.add(5); //错误, 类外部不能访问类任何private的成员
    A.x=A.y=1; //错误, 类外部不能访问类任何private的成员
    B.data=101; //正确, A.data和B.data是两个对象的数据成员,
    为不同的存储空间
    B.fun(4,5,6); //正确, A.fun和B.fun函数调用相同的代码, 但
    作用不同的数据成员
}
```

▶ 2. 通过对象指针访问对象中的成员

▶ 访问对象中数据成员的一般形式为：

对象指针—>成员名

▶ 调用对象中成员函数的一般形式为：

对象指针—>成员函数（实参列表）

▶ 则函数

```
void caller1()
{
    Data A, *p, *p1; //定义对象指针变量
    p1=&A; //p1指向对象A
    p1->data=100; //正确，类外部可以访问类的public数据成员
    p1->fun(1,2,3); //正确，类外部可以访问类的public成员函数
    p = new Data; //动态分配Data对象
    p->data=100; //正确，类外部可以访问类的public数据成员
    p->fun(1,2,3); //正确，类外部可以访问类的public成员函数
    delete p; //撤销p所指向的Data对象
}
```


▶ 3. 通过引用变量访问对象中的成员

▶ 访问对象中数据成员的一般形式为：

对象引用变量名.成员名

▶ 调用对象中成员函数的一般形式为：

对象引用变量名.成员函数(实参列表)

▶ 则函数

```
void caller1()
{
    Data A, &r=A; //定义对象引用变量
    r.data=100; //正确，类外部可以访问类的public数据成员
    r.fun(1,2,3); //正确，类外部可以访问类的public成员函数
}
```

- ▶ 如果一个类定义了两个或多个对象，则这些同类的对象之间可以互相赋值。这里所指的对象的“值”是指对象中所有数据成员的值。

- ▶ 对象赋值的一般形式为：

对象名1=对象名2

- ▶ 注意对象名1和对象名2必须属于同一个类。

- ▶ 关于对象赋值的说明：
- ▶ （1）对象的赋值只对其中的非静态数据成员赋值，而不对成员函数赋值。
- ▶ （2）如果对象的数据成员中包括动态分配资源的指针，按上述赋值的原理，赋值时只复制了指针值而没有复制指针所指向的内容。

26.5 对象、对象指针或对象引用作为函数的参数和返回值

- ▶ 函数的参数可以是对象、对象指针或对象引用。
- ▶ (1) **当形参是对象时，实参要求是相同类的对象名**，C++不能对类对象进行任何隐式类型转换。此时形参是实参对象的副本。
- ▶ 采用这样的值传递方式会增加函数调用在空间、时间上的开销，特别是当数据成员的长度很大时，开销会急剧增加。
- ▶ 实际编程中，传递对象时需要考虑类的规模带来的调用开销，如果开销很大时建议不用对象作为函数参数。

26.5 对象、对象指针或对象引用作为函数的参数和返回值

- ▶ (2) 当形参是对象指针时，实参要求是同类对象的指针，C++不能对对象指针进行任何隐式类型转换。
- ▶ 函数调用时，无论类多大规模，传递的参数是一个地址值，其开销非常小。
- ▶ 采用地址传递方式，在函数中若按间接引用方式修改了形参对象本质上就是修改实参对象。因此，使用对象指针作为函数参数可以向主调函数传回变化后的对象。

- ▶ (3) 当形参是对象引用时，实参要求是同类的对象，其功能与对象指针相似。
- ▶ 此时函数形参对象实际上是实参对象的别名，为同一个对象。在函数中若修改了形参对象本质上就是修改实参对象。因此，使用对象引用作为函数参数可以向主调函数传回变化后的对象。

26.5 对象、对象指针或对象引用作为函数的参数和返回值

【例26.2】

```
1  #include <iostream>
2  using namespace std;
3  void func1(Data a, Data *p, Data &r)
4  {
5      a.data=100;   p->data=200;   r.data=300;
6  }
7  int main()
8  {
9      Data A, B, C;
10     A.fun(1,2,3); B.fun(4,5,6); C.fun(7,8,9);
11     func1(A,&B,C); //将对象A、B的地址、对象C的引用传递到函数func1
12     return 0;
13 }
```


26.5 对象、对象指针或对象引用作为函数的参数和返回值

- ▶ 如果不希望在函数中修改实参对象的值，函数形参可以作const限定，例如：

```
void func2(Data a, const Data *p, const Data &r)
{
    a.data=100;
    p->data=200; //错误，左值是const对象
    r.data=300; //错误，左值是const对象
}
```

- ▶ 不必对对象形参作const限定，如上述代码中的形参a，因为即使在函数中修改了a，也不会影响实参对象。

- ▶ 函数返回值可以是对象、对象指针或对象引用。
- ▶ 函数返回对象时，将其内存单元的所有内容复制到一个临时对象中。因此函数返回对象时会增加调用开销。
- ▶ 函数返回对象指针或引用，本质上返回的是对象的地址而不是它的存储内容，因此**不要返回局部对象的指针或引用**，因为它在函数返回后是无效的。

26.5 对象、对象指针或对象引用作为函数的参数和返回值

▶ 例:

```
Data func1()  
{  
    Data a; a.fun(1,2,3);  
    return a; //可以返回局部对象，它被复制返回  
}  
  
Data* func2(Data *p1, Data *p2)  
{  
    if (p1->data > p2->data) return p1;  
    return p2;  
}  
  
Data& func3(Data &r1, Data &r2)  
{  
    if (r1.data > r2.data) return r1;  
    return r2;  
}
```

26.5 对象、对象指针或对象引用作为函数的参数和返回值

```
void caller()  
{  
    Data A, B, C;  
    A.fun(1,2,3); B.fun(4,5,6);  
    C=func1();  
    func2(&A,&B)->data=100; //等价于 (&B)->data=100;  
    func3(A,B).data=100; //等价于 B.data=100;  
}
```

26.5 对象、对象指针或对象引用作为函数的参数和返回值

▶ 例:

```
Data* func()  
{  
    Data a;  
    Data *p=&a;  
    return p;    //返回局部对象a的指针p  
}  
void caller()  
{  
    Data *p1;  
    p1=func();  
    p1->data=100; //a已不存在, 引用错误  
}
```

CP 程序设计