多态性的概念

多态性的概念

通过类型兼容，对于基类及其公有派生类的对象，可以使用相同的函数进行统一处理。比如，函数参数是基类类型，而实际调用该函数时既可以传入基类对象，也可以传入派生类对象。

【例3-5】类型兼容示例。

```
// Person.h
#include <iostream>
using namespace std;
class Person
{
public:
    Person(char *name, bool sex)
    {
        strcpy(m_name, name);
        m_sex = sex;
    }
    void DisplayInfo()
    {
        cout<<"个人信息："<<endl
            <<"姓名："<<m_name<<endl
            <<"性别：";
        if (m_sex==true) cout<<"男"<<endl;
        else cout<<"女"<<endl;
    }
protected:
    char m_name[20];        // 姓名
    bool m_sex;    // 性别（true：男，false：女）
};
```

```
// Student.h
#include "Person.h"
class Student : public Person
{
public:
    Student(char *sno, char *name, bool sex, char *major)
        : Person(name, sex)
    {
        strcpy(m_sno, sno);
        strcpy(m_major, major);
    }
    void DisplayInfo()
    {
        cout<<"学生信息："<<endl
            <<“学号：” <<m_sno<<endl
            <<"姓名："<<m_name<<endl
            <<"性别：";
        if (m_sex==true) cout<<"男"<<endl;
        else cout<<"女"<<endl;
        cout<<"专业："<<m_major<<endl;
    }

private:
    char m_sno[8];          // 学号
    char m_major[20];       // 专业
};
```

```cpp
// main.cpp
#include "Student.h"
void Print(Person &rp)
{
    rp.DisplayInfo();
}

int main()
{
    Student student("1210101", "张三", true, "计算机应用");
    Person person("李四", false);
    Print(student);              // 以Student类对象作为实参
    Print(person);               // 以Person类对象作为实参
    return 0;
}
```

输出结果：

```
个人信息：
姓名：张三
性别：男
个人信息：
姓名：李四
性别：女
```

- 在主函数中两次调用Print()函数，分别将基类对象person和派生类对象student作为实参传递给基类引用rp。但是，在使用基类引用rp调用DisplayInfo()函数时都是调用基类定义的函数。
- 显然，我们并不希望看到上面的运行结果。我们希望当以派生类对象作为实参传递给基类引用rp时，使用rp调用DisplayInfo()函数能够调用派生类定义的函数。这种能够根据指针或引用所表示的对象的实际类型来调用该对象所属类的函数、而不是每次都调用基类中函数的特性，就是所谓的多态性。