虚函数

语法

- 允许子类拥有自己的函数，父类拥有所有子类共同的函数。当使用父类指针或引用调用共同函数时，会根据具体的对象调用属于其自己的函数
- 虚函数使用关键字`virtual`
- 多态：同一个操作，作用于不同的对象，有不同的效果
- 虚函数父类和子类的函数的返回值、参数必须完全相同

```
class Testclass {
    ...
    virtual void foo(...)
    {
        ...
    }
    ...
};
```

模拟

不使用`virtual`关键字，达到虚函数的效果

```
class People {
public:
    // 利用成员函数指针模拟虚函数
    typedef void(People::*sayf_ptr)();
    typedef void(People::*watchf_ptr)();
    typedef void(People::*writef_ptr)();

    People() {}

    void say()
    {
        cout << "People::say()" << endl;
    }

    void watch()
    {
        cout << "People::watch()" << endl;
    }

    void write()
    {
        cout << "People::write()" << endl;
    }
```

```
public:
    sayf_ptr Say = &People::say;
    watchf_ptr Watch = &People::watch;
    writef_ptr Write = &People::write;
};

class Student : public People {
public:
    Student()
    {
        // 修正成员函数指针的指向
        Say = (sayf_ptr)(&Student::say);
        Watch = (watchf_ptr)(&Student::watch);
        Write = (writef_ptr)(&Student::write);
    }

    void say()
    {
        cout << "Student::say()" << endl;
    }

    void watch()
    {
        cout << "Student::watch()" << endl;
    }

    void write()
    {
        cout << "Student::write()" << endl;
    }
};

class Teacher : public People {
public:
    Teacher()
    {
        // 修正成员函数指针的指向
        Say = (sayf_ptr)(&Teacher::say);
        Watch = (watchf_ptr)(&Teacher::watch);
        Write = (writef_ptr)(&Teacher::write);
    }

    void say()
    {
        cout << "Teacher::say()" << endl;
    }

    void watch()
    {
        cout << "Teacher::watch()" << endl;
    }

    void write()
```

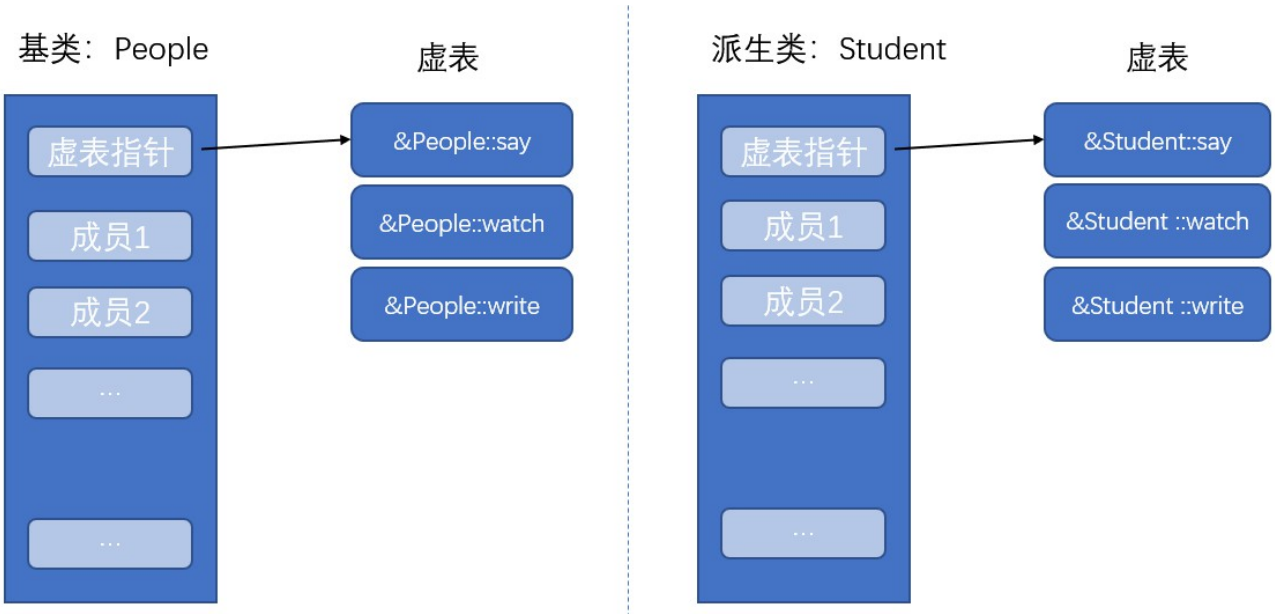
```
    {
        cout << "Teacher::write()" << endl;
    }
};

// 模拟虚函数
int main(void)
{
    People *objs[3] { &People(), &Student(), &Teacher() };

    for(int i = 0; i < 3; i++) {
        cout << i + 1 << " ------------ " << endl;
        (objs[i]->*(objs[i]->Say))();
        (objs[i]->*(objs[i]->Watch))();
        (objs[i]->*(objs[i]->Write))();
        cout << endl;
    }

    system("pause");
    return 0;
}
```

实现



- 存放虚函数地址的数组称之为**虚表**，对象中指向虚表的地址称之为**虚表指针**
- 一般虚表指针存放在对象首地址
- 一般虚表存放在全局数据区
- 含有虚表指针的类比没有虚表指针的类大4字节（32位下）
- 虚函数调用过程：
    1. 拿到对象首地址
    2. 取出虚表指针
    3. 查虚表找对应函数地址

4. 调用虚函数

- 每个类都有自己的虚表，每个类会在自己的构造函数中填入自己的虚表指针，这个时机比构造函数体执行要早
- 只要类含有虚函数就会有虚表

语法细节

- 在子类的一般成员函数中调用虚函数**有**多态效果
- 在父类的一般成员函数中调用虚函数**有**多态效果
- 在构造函数中调用虚函数**没有**多态效果
- 在析构函数中调用虚函数**没有**多态效果
- 构造**不能**是虚函数
- 析构**必须**是虚函数