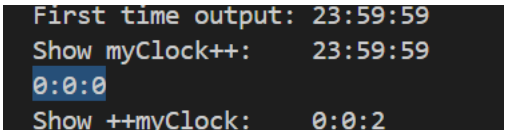


计算机学院 高级语言程序设计 课程实验报告

实验题目：多态 编程		学号：202200400053
日期：2024-05-09	班级：2202	姓名：王宇涵
Email： 1941497679@qq.com		
<p>实验目的：</p> <ol style="list-style-type: none">1. 掌握双目、单目运算符重载（编译时多态_静态绑定）2. 练习运算符重载的两种方式：类成员和非成员函数实现3. 掌握虚函数、纯虚函数、抽象类的应用（动态多态）4. 练习运行时类型识别技术在编程中的应用。		
<p>实验软件和硬件环境：</p> <p>软件环境：VSCODE + DEV-C++</p> <p>硬件环境：Legion Y7000P</p>		
<p>实验步骤与内容：</p> <ol style="list-style-type: none">1. 练习第 8 章 PPT 例 8-1，双目运算符重载（复数类加减法用成员函数实现）。 试在 main()函数中添加以下表达式，分析出错原因？ (1) <code>c3=1+c2;</code> 答：编译错误，因为 1 是 int 型，不能直接与 Complex 型对象相加。 (2) 将 <code>operator + (const Complex &c2)</code> 形参中的 <code>const</code> 去掉，main()中用表达式 <code>c3 = c1+1;</code> 可以吗？ 答：编译错误，因为 1 是 int 型，不能直接与 Complex 型对象相加。2. 练习第 8 章 PPT 例 8-2，单目运算符重载。 在 main()中试用 <code>++++myClock</code> 和后 <code>++++</code>，看能否实现连续加 1？如下： <pre>int main(){ Clock myClock(23, 59, 59); cout << "First time output: "; myClock.showTime(); cout << "Show myClock++: "; (myClock++++).showTime(); (myClock).showTime(); cout << "Show ++myClock: "; (++++myClock).showTime(); return 0; }</pre><pre>First time output: 23:59:59 Show myClock++: 23:59:59 0:0:0 Show ++myClock: 0:0:2</pre> 答：输出结果表明 <code>myClock++++</code> 只能实现+1，不能显示连续+1，而 <code>++++myClock</code> 可以实现连续+1。		

3. 实践第 8 章 PPT 例 8-3, 以非成员函数形式重载 Complex 的加减法运算和 “<<” 运算符。

(1) 试试将 main() 中 c3 运算式中的 c1 或 c2 变为数字, 看能执行吗? 分析原因。

```
c1 = (5, 4)
c2 = (2, 10)
c3 = c1 - 1 = (4, 4)
```

答: 可以执行, 原因为将 1 作为实部传入初始化为一个无名对象, 从而进行运算。

(2) 尝试类外运算符重载不用 friend 实现。如果不写 friend, 会不会当成员函数声明? (不再在类中声明)

```
ss Complex 此运算符函数的参数太多 C/C++(344)
lic: //外
Complex( 查看问题 (Alt+F8) 快速修复... (Ctrl+.) real(r), imag(i) { } //构造函数
Complex operator + (const Complex &cc1, const Complex &cc2); //运算符+重载
```

答: 如果不写 friend 会被当成员函数声明。

对比例 8-1 和 8-3, 运算符重载为成员函数与非成员函数, 分别使用以下运算表达式测试。

如何解决其中的出错? 二义性? 第一个操作数不是该类对象的混合运算? 交换律?

1.0+c2, c2+1.0, 2+c2 与 c2+2, 更多类型参与怎么办?

```
c3 = 1.0 + c2;
cout << c3 << endl;
c3 = c2 + 1.0;
cout << c3 << endl; (3, 10)
c3 = 2 + c2;
cout << c3 << endl; (3, 10)
c3 = c2 + 2;
cout << c3 << endl; (4, 10)
c3 = c2 + 2;
cout << c3 << endl; (4, 10) 没有与这些操作数匹配的 "+" 运算符 C/C++(349)
```

答: 测试输入如图, 非成员函数正常显示, 成员函数提示无匹配的+运算符。

解决出错的方法: 是只使用非成员函数定义运算符重载, 这样两个操作数都可以不是该类对象。更多类型参与也没有关系, 只要能够保证将其他类隐含转化为 Complex 类型即可。

4. 学生用书第 8 章实验 8, 实验任务 (1) lab8_1, 参见习题 8-7。(单目操作符重载)

试试 Point& operator++(); 中的&可以省略吗? 有什么区别?

答: 不能省略, 不使用引用无法实现对原对象的数值操作, 只会返回临时变量, 从而无法实现自增的效果。

5. 学生用书实现第 8 章实验 8, 实验任务 (2) lab8_2, 虚函数应用对比。

```
vehicle run!
vehicle stop!
bicycle run!
bicycle stop!
motocar run!
motocar stop!
motocycle run!
motocycle stop!
vehicle run!
vehicle stop!
vehicle run!
vehicle stop!
vehicle run!
vehicle stop!
vehicle run!
vehicle stop!
vehicle run!
vehicle stop!
```

```
vehicle run!
vehicle stop!
bicycle run!
bicycle stop!
motocar run!
motocar stop!
motocycle run!
motocycle stop!
vehicle run!
vehicle stop!
bicycle run!
bicycle stop!
motocar run!
motocar stop!
motocycle run!
motocycle stop!
```

答: 上图分别是使用前和使用后。可见虚函数可以是运行时多态的基础。

6. 分析第 8 章 PPT 例 8-5, 非多态类型, 无虚析构函数, 造成内存泄漏的例子, 分析出错原因?

答: 出错原因: 无虚析构函数, 删除 b 指针时只删除了 base* 的指针而没有删除 derived* 的指针。

(a) 在 main()中增加语句 `Base b1=Derived();` 请分析输出结果。

```
Base destructor
```

答：在 main 函数中创建了一个 Base 对象 b1，它是通过 Derived 的隐式类型转换得到的，因此只会调用基类的析构函数，因为 b1 本身是一个基类对象。

(b) 在 main()中增加语句 `Derived d1; fun(&d1);` 程序会出问题吗？

```
Base destructor
Derived destructor
Base destructor
Base destructor
```

答：会出问题，在创建 Derived 对象 d1 时，会调用 Derived 的构造函数来初始化成员变量 p，而 fun 函数只删除了 base*的指针而没有删除 derived*的指针，成员变量 p 指向的内存空间没有被释放，产生内存泄漏

7. 第 8 章 PPT，P68 例 8-9 dynamic_cast 例，做如下修改，分析运行结果。

(1) 给 void fun(Base *b) 函数中的 if 语言增加如下语句，

```
else
    cout<<b<<" "<<d<<endl;
```

```
Base::fun1()
0x62fe08 0
Derived1::fun1()
Derived1::fun2()
Derived2::fun1()
Derived2::fun2()
```

答：第一个调用 fun 函数时，传入的是一个 Base 指针，不含有 derived1*的类型，因此无法转化。

(2) 试将其中的 dynamic_cast 改为 static_cast

```
Base::fun1()
```

答：static_cast 不能用于指针之间，整型和指针之间，不同类型的引用之间互相转换，因此程序产生运行错误

8. 实现习题 8-6，抽象类。（Shape 类的析构函数之前应该加 virtual）

```
The area of the Circle is 78.5
The perimeter of the Circle is 31.4
The area of the Rectangle is 24
The perimeter of the Rectangle is 20
```

答：运行结果如上。

9. 实现习题 8-8，父指子（父类指针指子类对象）实现动态多态。

```
调用派生类的函数fn1()
调用基类的非虚函数fn2()
调用派生类的函数fn1()
调用派生类的函数fn2()
PS: D:\Baidu\Synedisk\CLASSES\
```

答：只有当函数为虚函数的时候，才有动态多态。

Fn2 不为虚函数，因此 pBaseClass->fn2()生成结果为调用基类的函数 fn2

10. 课本 8.6 综合实例，个人银行账户管理程序的改进。

```
2008-11-1      #S3755217 created
2008-11-1      #02342342 created
2008-11-1      #C5392394 created
(d)deposit (w)withdraw (s)show (c)change day (n)next month (e)exit
```

答：进行如下改进

(1)将 show 函数声明为虚函数,因此通过指向 CreditAccount 类实例的 Account 类型的指针来调用 show 函数时,被实际调用的将是为 CreditAccount 类定义的 show 函数,这样,如果创建一个 Account 指针类型的数组,使各个元素分别指向各个账户对象,就可以通过一个循环来调用它们的 show 函数;
(2)在 Account 类中添加 deposit、withdraw、settle 这 3 个函数的声明,且将它们都声明为纯虚函数,这使得通过基类的指针可以调用派生类的相应函数,而且无需给出它们在基类中的实现。经过这一改动之后,Account 类就变成了抽象类。

结论分析与体会:

本次实验我通过验证,调试,设计实验,成功掌握了运算符重载,虚函数,抽象类,运行时类型识别知识点,深刻理解了多态的特性,巩固了理论课知识的同时,通过实际的案例深化了自己的理解,提高了自己的编程能力,为我未来的学习打下了良好的基础。

就实验过程中遇到的问题及解决处理方法,自拟 1—3 道问答题:

1. 运行时如何进行类型识别?

答:用 dynamic_cast 做类型转换的尝试,或用 typeid 直接获取类型信息.

2. 父指针转化为子指针需要注意什么?

答:执行基类向派生类的转换时,一定要确保被转换的指针和引用所指向或引用的对象符合转换的目的类型.

3. 虚函数有什么意义?

答:虚函数经过派生之后,就可以实现运行过程中的多态.