实 验 报 告

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 班级学号 | 202015116 | 班级排序号 | 200124 | 姓名 | 梁祺若 | 实验日期 | 2021-5-24 |
| 实验名称 | | | | 实验九: 继承 | | | |

**第1题：**阅读文件夹“Inheritance”里所有的代码（共计18个）。针对每一段代码，回答下列问题：

（1）此段代码编译时是否会出现编译错误？若是请分析原因。

（2）若此段代码不出现编译错误，则请解释其运行结果。

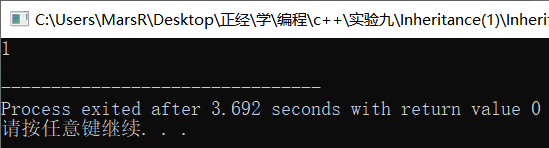
代码1：此段代码编译时不会出现编译错误。

此段代码的预期运行结果如下：

1

原因为：Arry类中定义了public的函数output用来输出值。函数判断指针位数和大小的值，相等输出1。

此段代码的实际运行结果如下：



代码2：此段代码编译时会出现编译错误，出错的位置位于第57行，出错的原因是无输出内容，new的数组没有释放，内存过大。应创建析构，在析构中delete。

代码3：此段代码编译时会出现编译错误，出错的位置位于第56 行，出错的原因是此处拷贝为浅拷贝，arry1和arry2为一个存储空间，改为深拷贝程序才能运行。Arry类里默认存在拷贝构造函数Arry（Arry &L）。

代码4：此段代码编译时不会出现编译错误。

此段代码的预期运行结果如下：

教师姓名:赵老师

教师年龄:35

教师专业:计算机科学与技术

教师职称:副教授

学生姓名:李同学

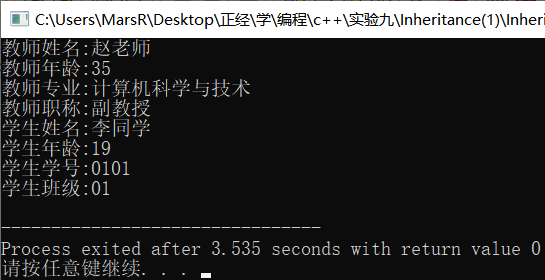
学生年龄:19

学生学号:0101

学生班级:01

原因为：继承，依次按输出要求输出。

此段代码的实际运行结果如下：



代码5：此段代码编译时不会出现编译错误。

此段代码的预期运行结果如下：

教师姓名:赵老师

教师年龄:35

教师专业:计算机科学与技术

教师职称:副教授

学生姓名:李同学

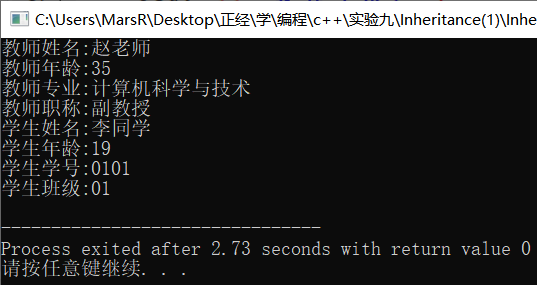
学生年龄:19

学生学号:0101

学生班级:01

原因为：依次按输出要求输出。但代码总体较代码4更加优化。

此段代码的实际运行结果如下：



代码6：此段代码编译时不会出现编译错误。

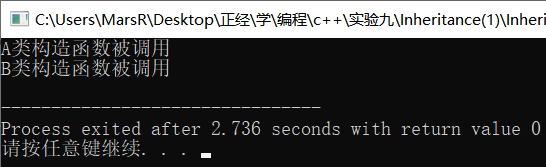
此段代码的预期运行结果如下：

A类构造函数被调用

B类构造函数被调用

原因为：将基类在派生类中调用，故两个均被输出。

此段代码的实际运行结果如下：



代码7：此段代码编译时不会出现编译错误。

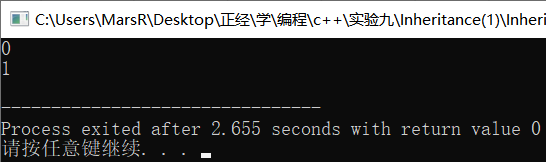
此段代码的预期运行结果如下：

0

1

原因为：将基类在派生类中调用，故两个均被输出。基类输出0，派生类输出1。

此段代码的实际运行结果如下：



代码8：此段代码编译时会出现编译错误，出错的位置位于第30行，出错的原因是在基类中没有定义相关联的A：：A。

代码9：此段代码编译时不会出现编译错误。

此段代码的预期运行结果如下：

A类构造函数被调用

B类构造函数被调用

B类析构函数被调用

A类析构函数被调用

原因为：按调用顺序输出。先输出构造函数，再输出析构函数。构造按顺序输出，析构按相反顺序。

此段代码的实际运行结果如下：



代码10：此段代码编译时不会出现编译错误。

此段代码的预期运行结果如下：

Shape的构造函数被调用!

Point的构造函数被调用!

Shape的构造函数被调用!

Point的构造函数被调用!

Shape的构造函数被调用!

Shape的构造函数被调用!

Point的构造函数被调用!

Shape的构造函数被调用!

Point的构造函数被调用!

Line的构造函数被调用!

Point的析构函数被调用!

Shape的析构函数被调用!

Point的析构函数被调用!

Shape的析构函数被调用!

Line的析构函数被调用!

Point的析构函数被调用!

Shape的析构函数被调用!

Point的析构函数被调用!

Shape的析构函数被调用!

Shape的析构函数被调用!

Point的析构函数被调用!

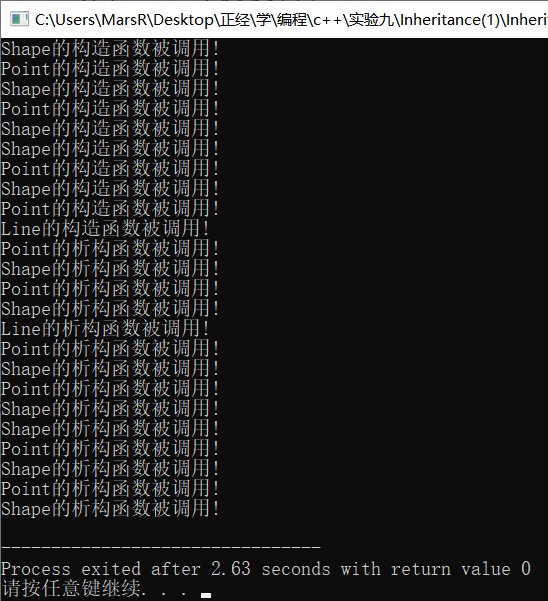
Shape的析构函数被调用!

Point的析构函数被调用!

Shape的析构函数被调用!

原因为：拷贝构造的函数不输出，但在析构时会输出。构造输出顺序为输出所示，析构时相反。

此段代码的实际运行结果如下：



代码11：此段代码编译时不会出现编译错误。

此段代码的预期运行结果如下：

A类构造函数被调用

B类构造函数被调用

A类构造函数被调用

B类拷贝构造函数被调用

1

2

3

4

1

0

3

4

B类析构函数被调用

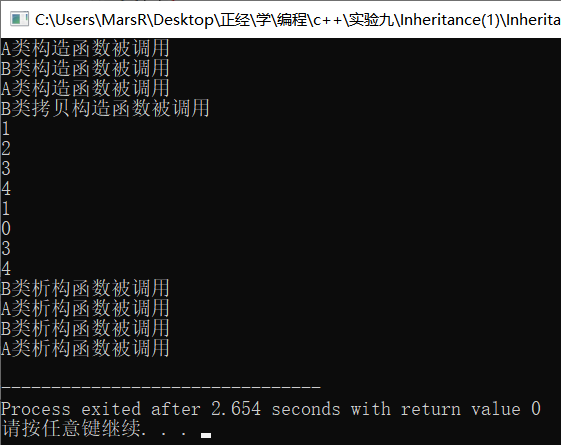
A类析构函数被调用

B类析构函数被调用

A类析构函数被调用

原因为：先输出构造，再析构

此段代码的实际运行结果如下：



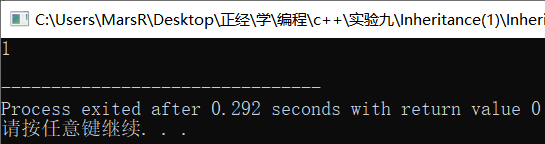
代码12：此段代码编译时不会出现编译错误。

此段代码的预期运行结果如下：

1

原因为：访问新增，不是访问本身。访问本身要objb.A：a 。

此段代码的实际运行结果如下：



代码13：此段代码编译时不会出现编译错误。

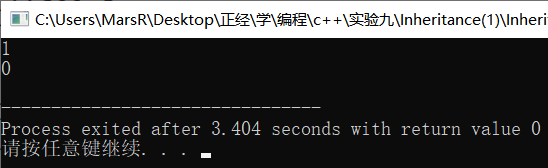
此段代码的预期运行结果如下：

1

0

原因为：访问新增和本身。

此段代码的实际运行结果如下：



代码14：此段代码编译时不会出现编译错误。

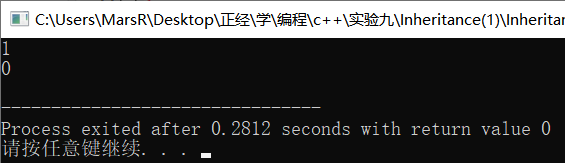
此段代码的预期运行结果如下：

1

0

原因为：return的是继承的。继承的访问不到新增的。

此段代码的实际运行结果如下：



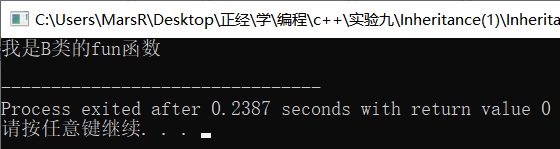
代码15：此段代码编译时不会出现编译错误。

此段代码的预期运行结果如下：

我是B类的fun函数

原因为：重写函数。调用自己的函数结果。

此段代码的实际运行结果如下：



代码16：此段代码编译时不会出现编译错误。

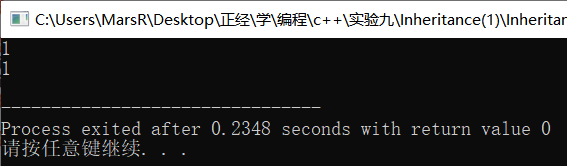
此段代码的预期运行结果如下：

1

1

原因为：在子类被调用时，先调用父类再调用子类，故两次都是1.

此段代码的实际运行结果如下：



代码17：此段代码编译时不会出现编译错误。

此段代码的预期运行结果如下：

A类构造函数被调用

B类构造函数被调用

A类拷贝构造函数被调用

B类拷贝构造函数被调用

1

2

3

4

1

2

3

4

B类析构函数被调用

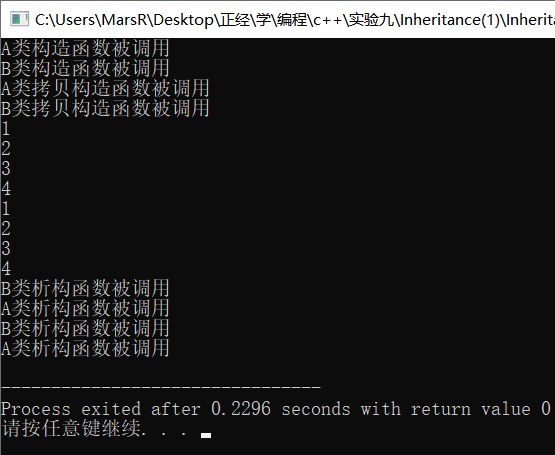
A类析构函数被调用

B类析构函数被调用

A类析构函数被调用

原因为：先调用构造，再调用析构

此段代码的实际运行结果如下：



代码18：此段代码编译时不会出现编译错误。

此段代码的预期运行结果如下：

Shape的构造函数被调用!

Point的构造函数被调用!

Shape的构造函数被调用!

Point的构造函数被调用!

Shape的构造函数被调用!

Shape的构造函数被调用!

Point的构造函数被调用!

Shape的构造函数被调用!

Point的构造函数被调用!

Line的构造函数被调用!

Point的析构函数被调用!

Shape的析构函数被调用!

Point的析构函数被调用!

Shape的析构函数被调用!

Point的析构函数被调用!

Shape的析构函数被调用!

Point的析构函数被调用!

Shape的析构函数被调用!

Line的析构函数被调用!

Point的析构函数被调用!

Shape的析构函数被调用!

Point的析构函数被调用!

Shape的析构函数被调用!

Shape的析构函数被调用!

Point的析构函数被调用!

Shape的析构函数被调用!

Point的析构函数被调用!

Shape的析构函数被调用!

原因为：先调用构造，再调用析构，拷贝的在构造时不输出

此段代码的实际运行结果如下：

