

第六章作业

6.3

(1)

上机代码及运行结果：

```
1 #include <iostream>
2
3 using namespace std;
4
5 class Point {
6 public:
7     Point(float a, float b) : x(a), y(b) {}
8     ~Point() { cout << "executing Point destructor" << endl; }
9 private:
10     float x, y;
11 };
12
13 class Circle : public Point {
14 public:
15     Circle(float a, float b, float r) : Point(a, b), radius(r) {}
16     ~Circle() { cout << "executing Circle destructor" << endl; }
17 private:
18     float radius;
19 };
20
21 int main() {
22     Point *p = new Circle(2.5, 1.8, 4.5);
23     delete p;
24
25     return 0;
26 }
```

Execution Summary
Output Summary Data
class Point {
public:
 Point(float a, float b) : x(a), y(b) {}
 ~Point() { cout << "executing Point destructor" << endl; }
private:
 float x, y;
};
class Circle : public Point {
public:
 Circle(float a, float b, float r) : Point(a, b), radius(r) {}
 ~Circle() { cout << "executing Circle destructor" << endl; }
private:
 float radius;
};
int main() {
 Point *p = new Circle(2.5, 1.8, 4.5);
 delete p;
 return 0;
}

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL POLYGLOT NOTEBOOK

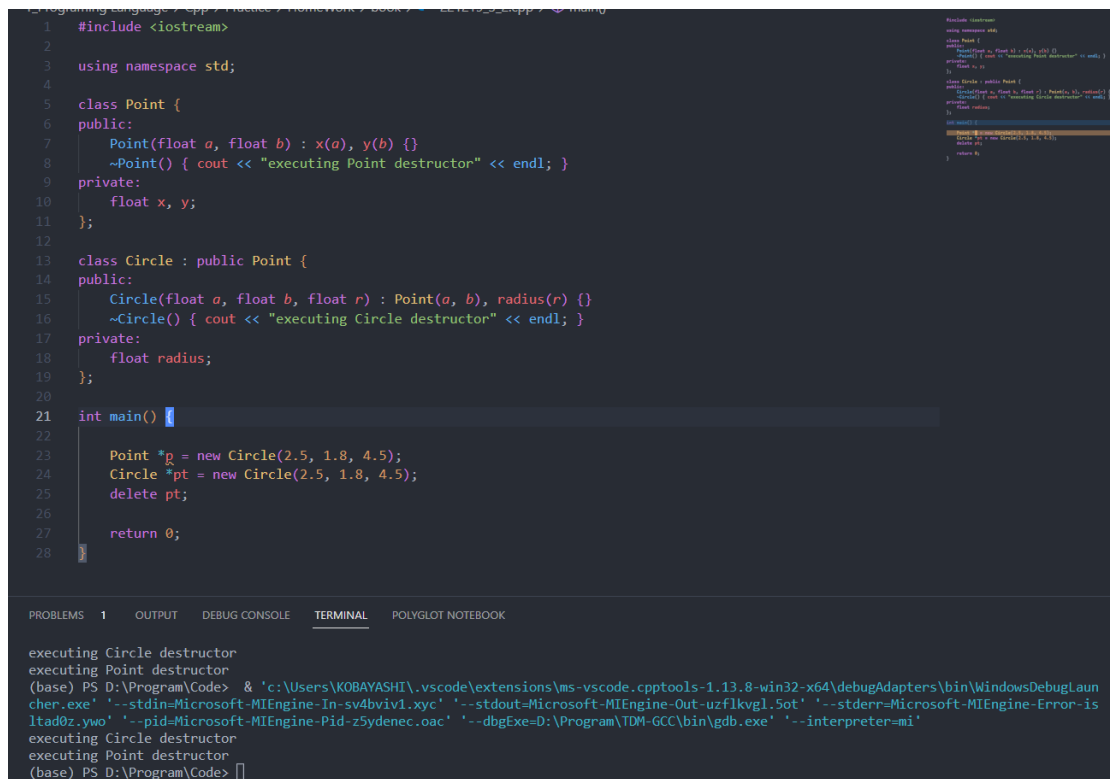
(base) PS D:\Program\Code> & 'c:\Users\KOBAYASHI\.vscode\extensions\ms-vscode.cpptools-1.13.8-win32-x64\debugAdapters\bin\WindowsDebugLa
ncher.exe' '--stdin=Microsoft-MIEngine-In-sp0215ka.ntr' '--stdout=Microsoft-MIEngine-Out-gz001zbv.qdm' '--stderr=Microsoft-MIEngine-Error-
5rye0aql.30p' '--pid=Microsoft-MIEngine-Pid-zui2dfdy.sdh' '--dbgExe=D:\Program\TDM-GCC\bin\gdb.exe' '--interpreter=mi'
executing Point destructor
(base) PS D:\Program\Code>

程序分析：

运行结果显示，使用 delete 运算符通过该 Point 指针尝试将其指向 Circle 类对象进行释放时，只会调用 Point 类中定义的析构函数。这是因为在 Circle 类继承 Point 类时，没有将 Point 类中的析构函数用 virtual 声明虚函数，使得类中没有生成对应的虚函数表。因此在通过 Point 类指针释放 Circle 类对象时，程序无法通过虚函数列表调用 Circle 类的析构函数。

(2)

上机代码及运行结果：



```
1 #include <iostream>
2
3 using namespace std;
4
5 class Point {
6 public:
7     Point(float a, float b) : x(a), y(b) {}
8     ~Point() { cout << "executing Point destructor" << endl; }
9 private:
10     float x, y;
11 };
12
13 class Circle : public Point {
14 public:
15     Circle(float a, float b, float r) : Point(a, b), radius(r) {}
16     ~Circle() { cout << "executing Circle destructor" << endl; }
17 private:
18     float radius;
19 };
20
21 int main() {
22
23     Point *p = new Circle(2.5, 1.8, 4.5);
24     Circle *pt = new Circle(2.5, 1.8, 4.5);
25     delete pt;
26
27     return 0;
28 }
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL POLYGLOT NOTEBOOK

```
executing Circle destructor
executing Point destructor
(base) PS D:\Program\Code> & 'c:\Users\KOBAYASHI\.vscode\extensions\ms-vscode.cpptools-1.13.8-win32-x64\debugAdapters\bin\WindowsDebugLaun
cher.exe' '--stdin=Microsoft-MIEngine-In-sv4bviv1.xyc' '--stdout=Microsoft-MIEngine-Out-uzflkvgl.5ot' '--stderr=Microsoft-MIEngine-Error-is
ltad0z.ywo' '--pid=Microsoft-MIEngine-Pid-zSydenec.oac' '--dbgExe=D:\Program\TDM-GCC\bin\gdb.exe' '--interpreter=mi'
executing Circle destructor
executing Point destructor
(base) PS D:\Program\Code>
```

程序分析：

从程序的运行结果中可以看出，在 Circle 类对象的生命周期结束时，程序先调用了派生类的构造函数，然后再调用了基类的构造函数。

虽然用这种方法也能够正常地调用派生类和基类的析构函数，但如果有多多个派生类的时候，程序调用虚构函数时将有可能导致二义性最终产生预想不到的后果。同时，为了操作的便利性，最好的做法还是使用虚函数。

(3)

上机代码及运行结果：

```
1_Programing Language > Cpp > Practice > HomeWork > book > 221219_3_3.cpp > Point
1 #include <iostream>
2
3 using namespace std;
4
5 class Point {
6 public:
7     Point(float a, float b) : x(a), y(b) {}
8     virtual ~Point() { cout << "executing Point destructor" << endl; }
9 private:
10     float x, y;
11 };
12
13 class Circle : public Point {
14 public:
15     Circle(float a, float b, float r) : Point(a, b), radius(r) {}
16     virtual ~Circle() { cout << "executing Circle destructor" << endl; }
17 private:
18     float radius;
19 };
20
21 int main() {
22
23     Point *p = new Circle(2.5, 1.8, 4.5);
24     delete p;
25
26     return 0;
27 }
```

StackFrame: stackFrame
using namespace std;
class Point {
public:
 Point(float a, float b) : x(a), y(b) {}
 virtual ~Point() { cout << "executing Point destructor" << endl; }
private:
 float x, y;
};
class Circle : public Point {
public:
 Circle(float a, float b, float r) : Point(a, b), radius(r) {}
 virtual ~Circle() { cout << "executing Circle destructor" << endl; }
private:
 float radius;
};
int main() {
 Point *p = new Circle(2.5, 1.8, 4.5);
 delete p;
 return 0;
}

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL POLYGLOT NOTEBOOK

加载个人及系统配置文件用了 1131 毫秒。
(base) PS D:\Program\Code> conda activate base
(base) PS D:\Program\Code> & 'c:\Users\KOBAYASHI\.vscode\extensions\ms-vscode.cpptools-1.13.8-win32-x64\debugAdapters\bin\WindowsDebugLa
ncher.exe' '--stdin=Microsoft-MIEngine-In-w35zk1hk.b3t' '--stdout=Microsoft-MIEngine-Out-qv0yyu44.fnb' '--stderr=Microsoft-MIEngine-Error-
0um2bb23.p32' '--pid=Microsoft-MIEngine-Pid-4d0y123s.qrr' '--dbgExe=D:\Program\TDM-GCC\bin\gdb.exe' '--interpreter=mi'
executing Circle destructor
executing Point destructor
(base) PS D:\Program\Code>

程序分析：

在基类和派生类中将析构函数声明为虚函数。其实，在派生类中的 virtual 声明是可选的，在将基类的虚函数声明为虚函数后，其派生类中的析构函数将自动地成为虚函数。

同时，可以看出，将基类中的析构函数声明为虚函数之后，就能够在通过基类指针释放派生类对象时能够正常地根据派生链顺序调用析构函数，正常地释放内存空间，防止了（1）中那样的内存泄露。

6.5

上机代码与运行结果：

```

5  class Shape {
6  public:
7      virtual double area() const = 0;
8  };
9
10 class Circle : public Shape {
11 public:
12     Circle(double r) : radius(r) {}
13     virtual double area() const { return 3.14159 * radius * radius; }
14 protected:
15     double radius;
16 };
17
18 class Square : public Shape {
19 public:
20     Square(double s) : side(s) {}
21     virtual double area() const { return side * side; }
22 protected:
23     double side;
24 };
25
26 class Rectangle : public Shape {
27 public:
28     Rectangle(double w, double h) : width(w), height(h) {}
29     virtual double area() const { return width * height; }
30 protected:
31     double width, height;
32 };
33
34
35 class Trapezoid : public Shape {
36 public:
37     Trapezoid(double t, double b, double h) : top(t), bottom(b), height(h) {}
38     virtual double area() const { return (top + bottom) * height / 2; }
39 protected:
40     double top, bottom, height;
41 };
42
43 class Triangle : public Shape {
44 public:
45     Triangle(double w, double h) : width(w), height(h) {}
46     virtual double area() const { return width * height / 2; }
47 protected:
48     double width, height;
49 };
50
51 int main() {
52     Circle circle(12.6);
53     Square square(3.5);
54     Rectangle rectangle(4.5, 8.4);
55     Trapezoid trapezoid(2.0, 4.5, 3.2);
56     Triangle triangle(4.5, 8.4);
57     Shape *pt[5] = {&circle, &square, &rectangle, &trapezoid, &triangle};
58     double areas = .0;
59     for (int i=0; i<5; ++i) {
60         areas += pt[i]->area();
61     }
62     cout << "total of all areas = " << areas << endl;
63     return 0;
64 }

```

```

total of all areas = 578.109
(base) PS D:\Program\Code>

```

程序分析：

在 main 函数中分别建立了 5 个类的对象，并定义了一个基类指针数组 pt，使其每一个元素指向一个派生类对象；for 循环的作用是将 5 个类对象的面积累加。pt[i]->area()是调用指针数组 pt 中第 i 个元素（是一个指向 Shape 类的指针）所指向的派生类对象的虚函数 area。