一．1.D 2.A 3.C 4.B 5.D 6.D 7.B 8.B 9.D 10.B 11.B 12.C 13.D 14.A 15.D

1. （1）-> （2）this指针 （3）友元 （4）构造函数 (5)delete []pa (6)不是 （7）纯虚函数 （8）纯虚函数 （9）抽象类 （10）先调用基类后调用派生类 （11）参数多态 （12）包含多态 （13）过载多态 （14）不可以 （15）可以 （16）虚基类 （17）作用域分辨符 （18）static （19）-> (20)用来强行抛出异常 (21)捕捉try语句块产生的或有throw抛出的异常，然后进行处理
2. 当用类的一个对象去初始化该类的另一个对象（或引用）时系统自动调用拷贝构造函数实现拷贝赋值。若函数的形参为类对象，调用函数时，实参赋值给形参，系统自动调用拷贝构造函数。当函数的返回值是类对象时，系统自动调用拷贝构造函数。
3. 编度译时的多态是通过静态连编来实现的；运行时的多问态是用动态连编来实现答的。编译时的多态性主要是通过函数重载和运算符重载来实现的回；运行时的多态性主要是通过虚函数来实现答的。
4. 优点：通过继承，我们可以大幅减少多余的代码，并扩展现有代码的用途。我们可以在标准的模块上构建我们的程序，而不必一切从头开始。这可以减少软件开发时间并提高生产率。由于继承、封装、多态的特性，使系统更灵活、更容易扩展，而且成本较低。

缺点：类和继承等特点使得程序会多很多指针操作来定位函数入口和自身要维护虚拟方法表等额外的工作，程序的处理效率相对要低。

1. 32.结果：20

20

30

20

33.结果：Instance 1 is constructed.

Instance 1 is constructed.

Instance 2 is constructed.

34.结果：i:10

j:12

35.结果：B's constructor called.

B's default constructor called.

C's constructor called.

5

6

C's destructor called.

B's destructor called.

B's destructor called.

1. 十二行构造函数缺少成员名或基类名，类A不存在默认构造函数。

十二行改成B(int i, int j, int k, int l) :A(i, j) { x = k; y = l; }

1. （1）virtual （2）3.14\*r\*r
2. Complex.h头文件部分:

#include<iostream>

using namespace std;

class Complex

{

private:

double a, b;

public:

Complex():a(0), b(0) {}

Complex(double x, double y)

{

a = x;

b = y;

}

~Complex()

{

}

Complex operator+(Complex A)

{

return Complex(a + A.a, b + A.b);

}

Complex operator-(Complex A)

{

return Complex(a - A.a, b - A.b);

}

Complex operator\*(Complex A)

{

return Complex(a \* A.a - b \* A.b, b \* A.a + a \* A.b);

}

friend ostream& operator<<(ostream& os, const Complex& c);

};

ostream& operator<<(ostream& os, const Complex& c)

{

os << c.a;

if (c.b > 0)

os << "+" << c.b << "i" ;

else

os << c.b << "i" ;

return os;

}

Complex.cpp部分：

#include "Complex.h"

#include <iostream>

using namespace std;

void main(){

Complex c1(2.0, 3.0);

Complex c2(1.0, 2.0);

cout << c1+c2 << endl;

cout << c1-c2 << endl;

cout << c1\*c2 << endl;

}

39.