面向对象程序设计检测卷

11j192 高章涵 20191001393

1. **选择题**

**1~5** BDCBB

**6~10**  ADBDD

**11~15** BCDAD

1. **填空与简答题**

**16.** —>

**17.** this 指针

**18.** 友元

**19.** 初始化列表

**20.** delete pa[ ]

**21.** 不是

**22.** 纯虚函数 纯虚函数 虚构类

**23.** 先调用子类析构函数，在调用父类析构函数

**24.** 过载多态 参数多态 包含多态

**25.** 可以 不可以

**26.** 虚基类 作用域分辨符

**27.** 对象名加点操作符 private

**28.** 抛出异常 捕获异常

**29.**

1.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。

2.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。

3.当函数的返回值是类对象时，系统自动调用拷贝构造函数。

**30.**

编译多态性通过函数重载和运算符重载实现。

运行多态性通过继承和虚函数来实现。

**31.**

优点：

(1)是一种全新的系统分析设计方法（对象、类、结构属性、方法）。

(2)适用于各类信息系统的开发。

(3)实现了对客观世界描述到软件结构的直接转换 ，大大减少后续软件开发量。

(4)开发工作的重用性、继承性高，降低重复工作量。

(5)缩短了开发周期。

缺点：

(1)需要一定的软件支持环境。

(2)不太适宜大型的MIS开发，若缺乏整体系统设计划分，易造成系统结构不合理、各部分关系失调等问题。

(3)只能在现有业务基础上进行分类整理，不能从科学管理角度进行理顺和优化。

1. **看程序写结果**

**32.**

20

20

30

20

**33.**

Instance 1 is constructed

Instance 1 is constructed

Instance 2 is constructed

**34.**

i:10

j:12

**35.**

B's constructor called.

B's default constructor called

C's constructor called.

1. **综合题**

**36.**

12： （i，j）前面缺少基类的名称。类A不存在相应的构造函数。

修改为： B(int i, int j,int k,int l): A(i, j) { x =k; y = l; }

15： B的show函数显示x，y的值

修改为： void f1( )A::{Show( );}

**37.**

1. virtual
2. 3.14\*r\*r;
3. **编程题**

**38.**

**#include<iostream>**

**using namespace std;**

**class Complex {**

**private:**

**double real;**

**double img;**

**public:**

**Complex(const double x, const double y);**

**Complex(const Complex\* d);**

**Complex operator+(const Complex& a);**

**Complex operator-(const Complex& b);**

**Complex operator\*(const Complex& c);**

**friend ostream& operator<<(ostream& out, Complex c1);**

**};**

**Complex::Complex(const double x, const double y) :**

**real(x), img(y)**

**{}**

**Complex::Complex(const Complex\* d) {**

**this->img = d->img;**

**this->real = d->real;**

**}**

**Complex Complex:: operator+(const Complex& a) {**

**this->img += a.img;**

**this->real += a.real;**

**Complex newa(this);**

**return newa;**

**}**

**Complex Complex::operator-(const Complex& b) {**

**this->img -= b.img;**

**this->real -= b.real;**

**Complex newb(this);**

**return newb;**

**}**

**Complex Complex::operator\*(const Complex& c) {**

**this->real = (this->real \* c.real) - (this->img \* c.img);**

**this->img = (this->real \* c.img) + (this->img \* c.real);**

**Complex newc(this);**

**return newc;**

**}**

**ostream& operator<<(ostream& out, Complex c1) {**

**out << "实部 " << c1.real << " " << "虚部" << c1.img << "i";**

**return out;**

**}**

**int main() {**

**Complex c1(1.0, 2.0);**

**Complex c2(1.5, 25.0);**

**cout << c1 + c2 << endl;**

**cout << c1 - c2 << endl;**

**cout << c1 \* c2 << endl;**

**return 0;**

**}**

**39.**

**#include<iostream>**

**using namespace std;**

**class Point {**

**public:**

**double x;**

**double y;**

**};**

**class LineString {**

**public:**

**LineString(Point\* pnts, int num);**

**LineString(const LineString& another);**

**~LineString();**

**LineString& operator =(const LineString& rhs);**

**Point& operator[](int index);**

**private:**

**Point\* m\_data;**

**int m\_num;**

**};**

**LineString::LineString(Point\* pnts, int num) {**

**m\_data = new Point[num];**

**for (int i = 0; i < num; i++) {**

**m\_data[i] = pnts[i];**

**}**

**m\_num = num;**

**}**

**LineString::LineString(const LineString& another) {**

**m\_num = another.m\_num;**

**m\_data = new Point[m\_num];**

**for (int i = 0; i < m\_num; i++) {**

**m\_data[i] = another.m\_data[i];**

**}**

**}**

**LineString::~LineString() {**

**delete[]m\_data;**

**m\_data = nullptr;**

**m\_num = 0;**

**}**

**LineString& LineString::operator=(const LineString& rhs) {**

**if (this == &rhs) {**

**return \*this;**

**}**

**delete[]m\_data;**

**m\_num = rhs.m\_num;**

**m\_data = new Point[m\_num];**

**delete[]rhs.m\_data;**

**}**

**Point& LineString::operator[](int index) {**

**return m\_data[index];**

**}**