C++期末检测卷

姓名 李受渊

班级 11J192

学号20191001576

1. 单选题

1.B 2.D 3.C 4.B 5.B 6.C 7.D 8.B 9.D 10.D

11.B 12.C 13.D 14.A 15.D

二．填空题

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 抽象类

23.先调用派生类的析构函数，再调用基类的析构函数

24.包含多态、过载多态、参数多态

25.不可以 可以

26.作用域分辨符 定义虚基类

27.private ：：

28.抛出异常 捕获异常

29.

1．当使用类的一个对象去初始化该类的另一个新对象时。

2．如果函数的形参是类的对象，那么当调用该函数时拷贝构造函数也会被调用。

3．如果函数的返回值是类的对象，那么函数执行完成返回调用者时会被调用。

30.

函数重载和运算符重载是编译时的多态，虚函数是运行时的多态，

编译时的多态是通过静态联编实现，运行时多态是动态联编实现

31.

优点：1、易维护

2、质量高

3、效率高

4、易扩展

不足：

我认为并不是所有的都需要面向对象，无法做到全都应用。

三．

32.

答：

20

20

30

20

33.

答：

Instance 1 is constructed.

Instance 1 is constructed.

Instance 2 is constructed.

34.

答：

i:10

j:12

35.

答：

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.

36.

答：

错误1 在12行，构造函数错误，改成B(int i,int j,int k,int l):A(i,j){x=k;y=l;}

错误2 在15行，要显示ab，得改成void f1（）{A::Show();}

37.

1 virtual

2 3.14\*r\*r

38.

答：

#include <iostream>

using namespace std;

class Complex {

public:

Complex(double x,double y):m\_r(x),m\_i(y){}//构造函数

~Complex(){}//析构函数

Complex operator+(const Complex& t);//‘+’重载

Complex operator-(const Complex& t);//‘-’重载

Complex operator\*(const Complex& t);//‘\*’重载

friend ostream& operator<<(ostream& cout, const Complex& t);//“<<"重载

private:

double m\_r;

double m\_i;

};

Complex Complex:: operator+(const Complex& t){

Complex m(\*this);

m.m\_r = m.m\_r + t.m\_r;

m.m\_i = m.m\_i + t.m\_i;

return m;

}

Complex Complex:: operator-(const Complex& t) {

Complex m(\*this);

m.m\_r = m.m\_r - t.m\_r;

m.m\_i = m.m\_i - t.m\_i;

return m;

}

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

Complex m(\*this);

m.m\_r = m.m\_r \* t.m\_r;

m.m\_i = m.m\_i \* t.m\_i;

return m;

}

ostream& operator<<(ostream& c, const Complex& t) {

c << t.m\_r << "+" << t.m\_i << "j" << endl;

return c;

}

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.

答：

#include <iostream>

using namespace std;

class Point {

public:

double x, y;

friend class LineString;

};

class LineString { //由多个点组成的线串类，由 n 个点组成的折线包含 n-1 个线段

public:

LineString(Point\* pnts, int num); //构造函数,通过传入点数组来构造

LineString (const LineString &another); // 复制构造函数

~LineString(); // 析构函数

LineString& operator =(const LineString& rhs) {// 赋值函数

LineString m(\*this);

m.m\_data = rhs.m\_data;

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

return m;

}

Point& operator[](int index){//返回线串中第 index 个点的引用

return m\_data[index - 1];

}

private:

Point\* m\_data; // 用于保存组成线串的点

int m\_num;// 线串中点的数量

};

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

\*m\_data = \*pnts;

m\_num = num;

}

LineString::LineString(const LineString& another) {

m\_data = another.m\_data;

m\_num = another.m\_num;

}