北京交通大学考试试题（A或B卷）

课程名称：面向对象程序设计与C++ 学年学期：2019—2020学年第一学期

课程编号：80L342Q 开课学院：计算机与信息技术学院 出题教师：王涛 等

学生姓名：_____ 学号：_____ 任课教师：_____

学生学院：_____ 班级：_____

题号	一	二	三	总分
得分				
阅卷人				

一 填空选择题（单选，20分，每空1分）

1. 在面向对象的程序设计中，基类可以为其派生类提供一组公共的行为，而各个派生类可以为这组公共行为提供不同的实现，从而使得派生类的对象在执行这些公共行为时会有不同的表现，这种特性称为面向对象的【 】

A. 封装性 B. 消息性 C. 继承性 D. 多态性

2. 关于域的描述中，错误的是【 】

A. 一个类是一个独立的域
B. 派生类域被自动嵌套在基类类域中
C. 域操作符::可以被重载
D. 不同域中可以定义同名变量和函数

3. 下列对变量的引用中错误的是【 】

A. int a; int &p=a;　　B. char a; char *p=&a;
C. int a; int &p; p=a;　　D. float a; float p=a;

4. 如果一个函数的声明是 int fun(char *)，则下列函数声明中，错误地重载了该函数的是【 】

A. int fun(char *, int)　　B. void fun(char *)
C. int fun(int)　　D. void fun(char *, int)

5. 下面是关于派生类构造函数的执行顺序的叙述：

① 执行派生类的构造函数函数体中的内容。
② 按照基类被继承时声明的顺序（从左向右）执行基类的构造函数。
③ 按照内嵌成员对象在类中声明的顺序执行它们的构造函数。

上述执行步骤的正确排序是【　　】

A. ① → ② → ③　　B. ② → ① → ③

C. ③ → ② → ①　　D. ② → ③ → ①

6. 设 A 是一个有不带参数的构造函数的类，fun 是一个如下定义的函数：

A fun(A & a){ return a; }

则在执行程序段 { A a; fun(a); } 时，对 A 类的：

① 构造函数，② 析构函数，③ 拷贝构造函数的调用顺序是【　　】

A. ① → ③ → ③ → ② → ② → ②

B. ① → ③ → ② → ②

C. ① → ② → ③ → ②

D. ① → ② → ③ → ③ → ② → ②

7. 下列关于静态数据成员的特性的叙述中，错误的是【　　】

A. 说明静态数据成员时前边要加修饰符 static。

B. 静态数据成员要在类体外进行初始化。

C. 静态数据成员不是所有对象所共有的。

D. 初始化静态数据成员时，要在其名称前加类名和作用域运算符。

8. 下面关于友元函数和友元类的叙述中，错误的是【　　】

A. 如果在类 A 中把类 B 的某个函数声明为自己的友元函数，那么在类 A 的定义之前一定要有类 B 的完整定义。

B. 如果在类 A 中把类 B 声明为自己的友元类，那么在类 A 的定义之前可以仅有类 B 的前向声明，而不必有类 B 的完整定义。

C. 如果将类 B 声明为类 A 的友元，那么类 A 也就成为类 B 的友元。

D. 如果将类 B 声明为类 A 的友元，那么类 B 中的函数可以访问类 A 的全部成员。

9. 下列关于常成员和常对象的叙述中，正确的是【　　】

A. 通过常对象只能调用其常成员函数。

B. 常成员函数只能由常对象调用。

C. 通过常成员函数可以修改常成员数据的值。

D. 常成员函数可以通过类名来调用。

```
        C(int a, int b, int c, int d):B1(a),memberB2(d),memberB1(c),B2(b) {
                cout<<"C 的构造函数被调用。"<<endl;
         }
        ~C() { cout<<"C 的析构函数被调用。"<<endl;}
private:
        B1 memberB1;        B2 memberB2;
};
void main(){
  C obj(1,2,3,4);
}
```

6 写出程序运行结果(8 分).

```
#include<iostream>
using namespace std;
class Shape{
public:
        Shape() { cout<<"Shape 的构造函数被调用。"<<endl; }
        void Display() { cout<<"Shape 的显示函数被调用。"<<endl; }
        virtual ~Shape() { cout<<"Shape 的析构函数被调用。"<<endl; }
};
class Rectangle: public Shape {
public: Rectangle () {
                Side=Hight=0;
                 cout<<"Rectangle 的默认构造函数被调用。"<<endl;
          }
        Rectangle(int xx,int yy)  {
                Side=xx;  Hight=yy;
                cout<< "Rectangle 的构造函数被调用。"<<endl;
        }
        void Display() {
              cout<<" Rectangle 的边长和高为："<<Side<<","<<Hight<<endl;
        }
        ~Rectangle()  {  cout<<"Rectangle 的析构函数被调用。"<<endl;  }
      int GetSide() {return Side;}
      int GetHight() {return Hight;}
 private:
     int  Side, Hight;
};
void main()
{
   Shape *ptr1 = new Rectangle[2];
   (*(ptr1+1)).Display();
        delete [] ptr1;
}
```

2 写出程序运行结果(6 分).

```
#include <iostream>
using namespace std;
class point {
protected:
        int x, y;
public:
        point(int xx, int yy) {
                x = xx, y = yy;
                cout << "construct point: " << x << "," << y << endl;
        }
        ~point() {      cout << "destroy point" << endl;}
};
class circle {
protected:
        point center;
        int  radius;
public:
        circle(int x, int y, int r):center(x, y),radius(r) {
                cout << "construct circle: " << radius << endl;
         }
        ~circle(){      cout << "destroy circle " << endl;}
};
int main() {
        circle c(1,2,3);
        return 0;
}
```

3 写出程序运行结果(6 分).

```
#include <iostream>
using namespace std;
class counter {
protected:
        static int count;
        int id;
public:
        counter() {
                id = ++ count;
                cout << "id = " << id << ", count = " << count << endl;
        }
};
int counter::count = 0;
static counter c[3];
```

```
int main() {
        cout << "initialize program" << endl;
        return 0;
}
```

4 写出程序运行结果(10 分).

```
#include <iostream>
#include <string.h>
using namespace std;
class text {
protected:
        char *content;
        int len;
public:
        text(int size)          { content = new char[len = size]; }
        ~text()                 { if(content != NULL) delete []content; }
        void show()             { cout << content << endl; }
        char * GetContent()     { return content;}
        void SetContent(char *str) { strcpy(content, str); }
};
int main() {
        text s1(20);    s1.SetContent("hello");
        text s2(s1);    strcat(s2.GetContent(), " world" );
        s1.show();      s2.show();
        return 0;
}
```

(1) 请写出程序的输出结果(4 分);

(2) 程序运行时是否会出现异常？如要避免, 应该对 text 类如何修改？(6 分)

5 写出程序运行结果(8 分).

```
#include <iostream>
using namespace std;
class B1 {
public:  B1(int i) {cout<<"B1 的构造函数被调用:"<<i<<endl;}
        ~B1() {cout<<"B1 的析构函数被调用。"<<endl;}
};
class B2 {
public: B2(int j) {cout<<"B2 的构造函数被调用:"<<j<<endl;}
        ~B2() {cout<<"B2 的析构函数被调用。"<<endl;}
};
class C: public B2, public B1 {
public:
```

三 程序设计题（30 分）

IP 地址本中包含两类 IP 地址:

- 普通地址, 包含如下信息:
  地址值 (唯一), 地址名, 对应的 MAC 地址
  例: 192.168.1.1　PC-01　　00-0a-eb-15-f0-ff
- 组地址, 包含如下信息:
  地址值 (唯一), 地址名, 该组中包含的地址列表(列表中每个地址是一个普通地址).
  例: 239.1.1.1　　Group-01　　{192.1681.1; 192.168.1.2; 192.168.1.3}

采用 C++面向对象程序设计方法，编写 IP 地址本管理程序:

- 从一个文本文件“IP_book.txt”中读取 IP 地址本信息。文件格式及示例数据如下：

  0　192.168.1.1　PC-01　　00-0a-eb-15-f0-ff
  0　192.168.1.2　PC-02　　00-bc-ad-45-0f-f0
  0　192.168.1.3　PC-03　　00-ba-cc-e0-38-20
  1　239.1.1.1　　Group-01　{192.1681.1; 192.168.1.2; 192.168.1.3}
  0　192.168.1.4　PC-04　　10-ac-00-05-1f-a0
  0　192.168.1.5　PC-05　　10-aa-00-00-28-10
  1　239.1.1.2　　Group-02　{192.1681.4; 192.168.1.5}

  注：每一行首部的 0/1 数字用于标识是否为组地址；以上仅为示例数据，实际数据按此格式存储，但数据内容不定。
- 给定一个 IP 地址值, 查找对应的信息. 如果所给是一个组地址, 应该列出该组中所有地址的信息.
  以上述示例数据为例，若输入“192.168.1.1”则输出
  192.168.1.1　PC-01　　00-0a-eb-15-f0-ff
  若输入“239.1.1.2”则输出
  239.1.1.2　　Group-02　{192.1681.4; 192.168.1.5}
  192.168.1.4　PC-04　　10-ac-00-05-1f-a0
  192.168.1.5　PC-05　　10-aa-00-00-28-10
- 添加/删除一个 IP 地址.
- 将 IP 地址本信息写入文件“IP_book.txt”中，写入格式与原格式相同.

10. 关于成员函数特征的下列叙述中，错误的是【　　】

A. 成员函数一定是内联函数；　　B. 成员函数可以重载；

C. 成员函数可以设置缺省参数值；　D. 成员函数可以是静态的。

11. 设 A 是一个类，且语句"A * p = new A[10];"是能够正确执行的，则下列关于该语句的执行的叙述最恰当的是【　　】

A. 类 A 有不带参数的构造函数，且构造函数被调用 10 次。

B. 类 A 有拷贝构造函数，且拷贝构造函数被调用 10 次。

C. 类 A 有这样的构造函数，其每个参数都有默认的形参值，且该构造函数被调用 10 次。

D. A 和 C 皆有可能。

12. 如果派生类中有指针类型的数据成员，该数据成员指向的内存空间需要动态申请，并且需要在构造函数中初始化该数据成员，在析构函数中回收该成员可能指向的内存空间。此时，如果存在通过基类的指针删除派生类的对象的情况，则需要【　　】

A. 将基类的构造函数声明为虚函数。

B. 将基类的析构函数声明为虚函数。

C. 将派生类的构造函数声明为虚函数。

D. 将派生类的析构函数声明为纯虚函数。

13. 运算符">"被友元重载，则表达式"obj1 > obj2"被编译器解释为【　　】

A. operator >(obj1, obj2)　B. >(obj1, obj2)

C. obj2.operator >(obj1)　D. obj1.operator >(obj2)

14. 程序中重载了 C++流运算符<<，它是【　　】

A. 用于输出操作的成员函数。　　B. 用于输入操作的成员函数。

C. 用于输入操作的非成员函数。　D. 用于输出操作的非成员函数。

15. 有如下动态数组模板类 Array 的声明，请在空格处填入最合适的内容。

```
__(1)__ <class T>
class Array
{protected:
  int size;          //动态数组的长度
  __(2)__ elements;    //动态数组的首地址
public:
```

7 写出程序运行结果(8 分)。

```
#include <iostream>
using namespace std;
class Expt{};
class Expt1: public Expt{
public:
        Expt1() { cout<<"Expt1 的构造函数被调用。"<<endl; }
        Expt1(Expt1 & e) {cout<<"Expt1 的拷贝构造函数被调用。"<<endl; }
        ~Expt1() { cout<<"Expt1 的析构函数被调用。"<<endl; }
};
class Expt2: public Expt1 {
public:
        Expt2() { cout<<"Expt2 的构造函数被调用。"<<endl; }
        Expt2(Expt2 & e) { cout<<"Expt2 的拷贝构造函数被调用。"<<endl; }
        ~Expt2() { cout<<"Expt2 的析构函数被调用。"<<endl; }
};
void MyFunc() {   Expt1 e;  throw e; }
void main() {
  try {
     cout << "在 try 块中，调用 MyFunc()。" << endl;
     MyFunc();
     cout << "在 try 块中，MyFunc()执行结束。" << endl;
  }
  catch( Expt E ) { cout << "捕获到 Expt 类型异常。"<<endl; }
  catch( Expt2 E ) {  cout << "捕获到 Expt2 类型异常。"<<endl; }
  cout << "回到 main 函数。" << endl;
}
```

```
    Array(int i);
    (3) ~Array();        //析构函数
    Array(  (4)  a);     //拷贝构造函数
    (5) operator[](int index);  //重载下标运算符
    (6) ostream & operator <<(ostream &o, Array & a);
};
```

(1) 空格(1)处应该填写【 】

A. virtual B. template C. void D. class

(2) 空格(2)处应该填写【 】

A. T * B. T C. int * D. int

(3) 空格(3)处应该填写【 】

A. void B. int C. virtual D. template

(4) 空格(4)处应该填写【 】

A. Array B. Array * C. Array & D. const Array &

(5) 空格(5)处应该填写【 】

A. T B. T * C. T & D. void

(6) 空格(6)处应该填写【 】

A. friend B. const C. virtual D. void

二 程序分析（50分）

1 写出程序运行结果(4分).

```
#include <iostream>
using namespace std;
char* inc(char* ch) {return ++ch;}
char& inc(char& ch) {return ++ch;}
int main(){
    char buf[16] = "cplusplus";
    char* p = buf;
    for (int i = 0; i < 6; i++) {
        p = inc(p);
        inc(*p);
    }
    cout << buf << endl;
    return 0;
}
```