**2014年选择题复习**

**1、有以下代码，**

**string s;while(cin>>s);**

**cout<<s;如从键盘上输入：This is C++ program!则正确的显示结果为：**

1. **This is C++ program!**
2. **This**
3. **program!**
4. **显示结果不确定**

**2、如果某C++函数的返回值类型为引用形式，下面描述中正确的是：**

1. **C++函数的返回值类型不允许是引用形式**
2. **不允许返回该函数的局部自动变量**
3. **需要通过临时变量的过渡来实现返回**
4. **在主调函数中建立了对函数返回值的引用**

**3、设va是vector<int>类型的向量，现有如下程序段：**

**for ( int i=0; i<va.size(); i++ ){ cout<<::abs(va[i])<<endl;}其中，::abs()函数调用的含义是\_\_\_\_\_\_\_\_。**

1. **调用std名空间中的abs函数。**
2. **调用不属于任何名空间的全局函数abs。**
3. **此表述方法有语法错误。**
4. **调用用户自定义名空间中的abs函数。**

**4、在标准输出流控制显示格式的操作中，以下不属于状态机制的为：**

**A、showpos**

**B、left**

**C、hex**

**D、setfill**

**5、如下C++程序的正确运行结果为：#include <iostream>#include <vector>using namespace std;void fun(vector<int> a){ int i; for(i=0;i<5;i++) {**

**a[i] = (i+1)\*10;}**

**}**

**void main(){ int i; vector<int> t(6,1); fun(t); for(i=0;i<t.size();i++) { cout<<setw(10)<<t[i]; } cout<<endl;**

**}A、10 20 30 40 50 60**

**B、0 10 20 30 40 1**

**C、1 1 1 1 1 1**

**D、5 5 5 5 5 5**

**6、在C++中，针对如下程序片段，下面描述中正确的是：**

**void num(const int &a){**

**…}void main( ){ int x=9; … num(x); …}A、实参x将值传递给了形参a**

**B、在被调用的num函数中为实参x建立了名为a的const引用**

**C、实参x将其地址传递给了形参a**

**D、形参a在函数num调用开始后即在内存中占用相应空间，并在函数执行过程中此空间的地址和其中存储的数值都不能发生变化**

**7、以下代码片段错误的是：**

**A、char \*s; s=”OK”;**

**B、char s[20]=”OK”;**

**C、string s; s=”OK”;**

**D、char \*s; strcpy(s,”OK”);**

**E、char s[20]; s=”OK”;**

**8、下列关于静态成员函数的描述中正确的是\_\_\_\_\_\_\_\_\_\_\_。**

**A、在静态成员函数中可以使用this指针**

**B、在建立对象前，就可以为静态数据成员赋值**

**C、静态成员函数在类外定义时，要加static前缀**

**D、静态成员函数只能在类外定义**

**9、假定AB为一个类，则执行**

**AB\*p=new AB(1，2); AB arry[2];**

**语句时共调用该类构造函数的次数为\_\_\_\_\_\_\_\_\_\_\_\_\_。**

**A、0 B、1 C、2 D、3**

**10、**

**10. 关于类和对象不正确的说法是（ ）**

**A) 类是一种类型，它封装了数据和操作**

**B) 对象是类的实例**

**C) 一个类的对象只有一个**

**D) 一个对象必属于某个类**

**11. 在类定义的外部，可以被访问的成员有（ ）**

**A) 所有类成员 B) private或protected的类成员**

**C) public的类成员 D) public或private的类成员**

**12. 关于this指针的说法错误的是（ ）**

**A) this指针必须显示说明 B) 当创建一个对象后，this指针就指向该对象**

**C) 成员函数拥有this指针 D) 静态成员函数不拥有this指针**

**13. 声明一个类的对象时，系统自动调用（ ）函数，撤消对象时，系统自动调用（C ）函数**

**A) 成员函数 B) 构造函数 C) 析构函数 D) 普通函数**

**14. 下面对构造函数的不正确描述是（ ）**

**A) 系统可以提供默认的构造函数 B) 构造函数可以有参数，所以可以有返回值**

**C) 构造函数可以重载 D) 构造函数可以设置默认参数**

**15.下面对析构函数的正确描述是（ ）**

**A) 系统不能提供默认的析构函数 B) 析构函数必须由用户定义**

**C) 析构函数没有参数 D) 析构函数可以设置默认参数**

**16. 对静态成员的不正确描述是（ ）**

**A) 静态成员不属于对象，是类的共享成员 B) 静态数据成员要在类外定义和初始化**

**C) 调用静态成员函数时要通过类或对象激活，所以静态成员函数拥有this指针**

**D) 非静态成员函数也可以操作静态数据成员**

**17. 下面选项中，不是类的成员函数为（ ）**

**A) 构造函数 B) 析构函数 C）友元函数 D) 缺省参数的构造函数方言**

**18.下面对友元的错误描述是（）**

**A) 关键字friend用于声明友元**

**B) 一个类的成员函数可以是另一个类的友元**

**C) 友元函数访问对象的成员不受访问特性影响**

**D) 友元函数通过this指针访问对象成员**

**19.在C++中，类与类之间的继承关系具有（ ）**

**A) 自反性 B) 对称性 C) 传递性 D) 反对称性**

**20. 下列关于类的继承描述中，（ ）是错误的**

**A) 派生类可以访问基类的所有数据成员，也能调用基类的所有成员函数**

**B) 派生类也是基类，但基类不一定具有派生类的全部属性和方法**

**C) 继承描述类的层次关系，派生类可以具有与基类相同的属性和方法**

**D) 一个基类可以有多个派生类，一个派生类可以有多个基类**

**21. 当一个派生类保护继承一个基类时，基类中的所有公有成员成为派生类的（ A）**

**A) public成员 B) private成员 C) protected成员 D) 友元**

**22.当一个派生类私有继承一个基类时，基类中的所有公有成员和保护成员成为派生类的（ ）**

**A) public成员 B) private成员 C) protected成员 D) 友元**

**23. 不论派生类以何种方法继承基类，都不能使用基类的（ ）**

**A) public成员 B) private成员 C) protected成员 D) private成员和protected成员**

**24. 下面描述中，错误的是（ ）**

**A) 在基类定义的public成员在公有继承的派生类中可见，也能在类外被访问**

**B) 在基类定义的public和protected成员在私有继承的派生类中可见，在类外可以被访问**

**C) 在基类定义的public和protected成员在保护继承的派生类中不可见**

**D) 在派生类中不可见的成员要变成可访问的，需要进行访问声明。如何声明？**

**25. 在C++中，不能被派生类继承的函数是（ ）**

**A) 成员函数 B) 构造函数 C) 析构函数 D) 静态成员函数**

**26. 在创建派生类对象时，构造函数的执行顺序是（ D ）**

**A) 对象成员构造函数、基类构造函数、派生类本身的构造函数**

**B) 派生类本身的构造函数、基类构造函数、对象成员构造函数**

**C) 基类构造函数、派生类本身的构造函数、对象成员构造函数**

**D) 基类构造函数、对象成员构造函数、派生类本身的构造函数**

**27. 在C++中，要实现动态联编，必须使用（ ）调用虚函数**

**A) 类名 B) 派生类指针 C) 对象名 D) 基类指针**

**28. 下列函数中，可以作为虚函数的是（ ）**

**A) 普通函数 B) 构造函数 C)成员函数 D) 析构函数**

**29. 在派生类中，重载一个虚函数时，要求函数名、参数的个数、参数的类型、参数的顺序和函数的返回值（ ）**

**A) 不同 B) 相同 C) 相容 D) 部分相同**

**30. 下面函数原型声明中，（ ）声明了fun为纯虚函数**

**A) void fun()=0; B) virtual void fun()=0;**

**C) virtual void fun(); D) virtual void fun(){};**

**31. 若一个类中含有纯虚函数，则该类称为（ ）**

**A) 基类 B) 纯基类 C) 派生类 D) 抽象类**

**32. 假设A为抽象类，下列声明（ ）是正确的**

**A) A fun(int); B) A \*p; C) int fun(A); D) A Obj;**

**33. 下面描述中，正确的是（）**

**A) 虚函数是没有实现的函数 B) 纯虚函数的实现在派生类中**

**C) 抽象类是没有纯虚函数的类 D) 抽象类指针可以指向不同的派生类**

**34. 关于函数模板，描述错误的是（ ）**

**A) 函数模板必须由程序员实例化为可执行的函数模板**

**B) 函数模板的实例化由编译器实现**

**C) 一个类定义中，只要有一个函数模板，则这个类是类模板**

**D) 类模板的成员函数都是函数模板，类模板实例化后，成员函数也随之实例化**

**35. 下列的模板说明中，正确的（ ）**

**A) template <typename T1, typename T2>**

**B) template <class T1, T2>**

**C) template <class T1, class T2>**

**D) template <typename T1; typename T2>**

**36. 假设有函数模板定义如下：**

**Template <typename T>**

**Max(T a, T b, T &c)**

**{ c=a+b; }**

**下列选项正确的是（ ）**

**A) float x,y; float z; Max(x,y,z);**

**B) int x,y,z; Max(x,y,z);**

**C) int x,y; float z; Max(x,y,z);**

**D) float x; int y, z; Max(x,y,z);**

**37. 关于类模板，描述错误的是（ ）**

**A) 一个普通基类不能派生类模板**

**B) 类模板从普通类派生，也可以从类模板派生**

**C) 根据建立对象时的实际数据类型，编译器把类模板实例化为模板类**

**D) 函数的类模板参数须通过构造函数实例化**

**38. 建立类模板对象的实例化过程为（ ）**

**A) 基类->派生类 B) 构造函数->对象**

**C) 模板类->对象 D) 模板类->模板函数**

**39. 在C++中，容器是一种（ ）**

**A) 标准类 B) 标准对象 C) 标准函数 D) 标准类模板**

**40. read函数的功能是从输入流中读取（ ）**

**A) 一个字符 B) 当前字符 C) 一行字符 D) 指定若干个字符**

**41. 下列选项中，用于清除基数格式位置以十六制输出的语句是（ ）**

**A) cout<<setf(ios::dec, ios::basefield);**

**B) cout<<setf(ios::hex, ios::basefield);**

**C) cout<<setf(ios::oct, ios::basefield);**

**D) cin>>setf(ios::hex, ios::basefield);**

**42. 包含类fstream定义的头文件是（ ）**

**A) fstream.h B) ofstream.h C) ifstream.h D) iostream.h**

**43. 要求打开文件 d:\file.dat，可写入数据，正确的语句是（ ）**

**A) ifstream infile(“d:\file.dat”, ios::in); //单斜杠**

**B) ifstream infile(“d:\\file.dat”, ios::in);**

**C) ofstream infile(“d:\file.dat”, ios::out); //单斜杠**

**D) fstream infile(“d:\\file.dat”, ios::in| ios::out); //输入流**

**44. 假定已定义浮点型变量data，以二进制方式把data的值写入输出文件流对象outfile中去，正确的语句是（）**

**A) outfile.write((float \*) &data, sizeof(float));**

**B) outfile.write((float \*) &data, data);**

**C) outfile.write((char \*) &data, sizeof(float));**

**D) outfile.write((char \*) &data, data);**

**1、友元函数有一下特点：**

* **友元函数不是类的成员函数——除非确因成员函数频繁，一般不要定义成友元函数（与面向对象思想矛盾）；**
* **友元函数主要用于运算符的重载**
* **友元函数可以访问对象的私有成员，只是在类中声明时加上friend，由于不是成员函数，定义时不必在函数名前加上“类名::”**
* **友元函数无this指针，因此，一般应带有一个入口参数——通过入口参数传递的对象名来引用该对象的成员。**
* **当一个函数需要访问二个或以上的类时，友元函数非常有用**
* **友元函数是相对与类的成员函数而言的。**

**2、未经初始化的静态全局变量会被程序自动初始化为0**

**3、静态成员函数不拥有this指针 非静态成员函数也可以操作静态数据成员 静态数据成员要在类外定义和初始化 静态成员不属于对象，是类的共享成员**