一、判断题

1、get()函数不能从流中提取终止字符，终止字符仍留在流中。getline()函数可以从流中提取终止字符，但终止字符被丢弃。 【正确】

2、记录流的当前格式化状态标志字中的每一位用于记录一种格式，这种格式是不能被设置或清除的。【错】

**3、对每个可重载的运算符来讲，它既可以重载为友元函数，又可以重载为成员函数，还可以重载为非成员函数  【错】**

     注：并非对“每个”可重载的运算符来说,它既可以重载为友元函数,又可以重载为成员函数,还可以重载为非成员函数；一些运算符（如下标运算符[]，函数调用运算符()，赋值运算符=等）只能重载为成员函数，而左操作数为std::cout或std::cin的输入输出运算符只能重载为非成员函数。

4、动态绑定是在运行时选定调用的成员函数的。【正确】  
5、操纵符本身是一个对象，它可以直接被提取符或插入符操作。【正确】  
6、使用提取符(<<)可以输出各种基本数据类型的变量的值，也可以输出指针值。 【正确】  
7、作用域运算符(::)只能用来限定成员函数所属的类。 【错】  
8、可以在类的构造函数中对静态数据成员进行初始化。 【错】  
9、在私有继承中,基类中所有成员对派生类的对象都是不可见的。 【对】  
10、派生类中至少包含了它地所有基类地成员，在这些成员中可能有的是不可访问。【对】  
11、只要是类M继承了类N,就可以说类M是类N的子类型。【错】  
12、多继承情况下,派生类的构造函数中基类构造函数的执行顺序取决于定义派生类时所指定的各基类的顺序。【对】  
13、预定义的插入符从键盘上接收数据是不带缓冲区的。【错】

14、不可以定义抽象类的对象。 【正确】

15、内联函数的定义必须出现在第一次调用内联函数之前。【正确】

**16、模板函数与函数模板的意义完全相同。【错】**

17、只有常成员函数才可以操作常对象。【正确】

**18、引用可以不初始化。 【错】**

19、构造函数可以设置默认参数。 【正确】

20、类的析构函数的作用是对象的初始化。 【错】

21、cout的默认输出对象是键盘, cin的默认输入对象是屏幕。【错】

22、**抽象类可以用来直接创建对象。 【错】**

**23、常对象可以调用任意成员函数。 【错】**

24、C++语言支持封装性和继承性，不支持多态性。【错】

25、关系运算符可以用来比较两个字符的大小，也可以用来比较两个字符串的大小。【错】

26、字符串”sdfgh”的长度是5。【正确】

**27、引用和指针都可作函数参数。【正确】**

28、友元函数说明在类体内，它是一种成员函数。【错】

29、友元类必须被说明为公有成员。【错】

30、纯虚函数是在基类中说明的虚函数，它在该基类中没有定义的操作内容。【正确】

31、析构函数不能被继承。【正确】

32、运算符重载以后，其优先级和结核性都不能改变。【正确】

33、在C++中，函数名实际上是一个指针，它指向函数的入口。【正确】

**34、在程序能正确支持所设计的正常情况之后，再添加语句来处理异常情况，这种思想就是异常处理。**【正确】

**35、派生类是基类的子类型。【错误】**

**注：3中继承方式，公有派生类才是基类真正的子类，它完整地继承了基类地功能。**

二、 选择题

1. 每个类（C）构造函数

A. 只能有一个  B. 只可有公有的 C. 可以有多个 D. 只可有缺省的

2. 下列特性中C与C++共有的是( D)

A. 继承 B. 封装 C. 多态性 D. 函数定义不能嵌套

3. 下列关于析构函数的描述中，不正确的是（B）

A. 类中有且仅有一个析构函数 B. 析构函数可以有形参 C. 析构函数没有函数类型

D. 析构函数在对象消失时被自动执行

4. 下面关于友元函数的描述中，正确的是（C）

A. 友元函数的实现必须在类的内部定义 B. 友元函数是类的成员函数

C. 友元函数破坏了类的封装性和隐藏性 D. 友元函数不能访问类的私有成员

5.实现动态联编，可以通过（A）调用虚函数

A. 对象指针 B. 成员名限制 C. 对象名 D. 派生类名

6.下列对派生类的描述中，错误的是（D）

A.一个派生类可以作为另一个派生类的基类

B.派生类至少有一个基类

C.派生类的缺省继承方式是private

D.派生类只含有基类的公有成员和保护成员

7.C++类体系中，能被派生类继承的是（B）

A.构造函数

B.虚函数

C.析构函数

D.友元函数

8.C++是从早期的C语言逐渐发展演变而来的，与C语言相比，它在求解问题上进行的最大改进是（B）

A.面向过程

B.面向对象

C.安全性

D.复用性

9.下列关于构造函数的描述中，错误的是（D）

A.构造函数可以设置默认参数

B.构造函数在定义类对象时自动执行

C.构造函数可以是内联函数

D.构造函数不可以重载

10.下面的描述中，表达错误的是（B）

A.公有继承时基类中的public成员在派生类中仍是public的

B.公有继承时基类中的private成员在派生类中仍是private的

C.公有继承时基类中的protected成员在派生类中仍是protected的

D.私有继承时基类中的public成员在派生类中是private的

11.下面叙述不正确的是（D）

A.派生类一般都用公有派生

B.对基类成员的访问必须是无二义性的

C.赋值兼容规则也适用于多重继承的组合

D.基类的公有成员在派生类中仍然是公有的

12.在公有派生类的成员函数不能直接访问基类中的继承来的某个成员，则该成员一定是基类中的（A）

A.私有成员

B.公有成员

C.保护成员

D.保护成员或私有成员

13.对基类和派生类的关系描述中，错误的是（B）

A.派生类是基类的具体化

B.基类继承了派生类的属性

C.派生类是基类定义的延续

D.派生类是基类的特殊化

14.关于this指针使用的说法正确的是（A）

A.保证每个对象拥有自己的数据成员，但共享处理这些数据的代码

B.保证基类私有成员在子类中可以被访问

C.保证基类保护成员在子类中可以被访问

D.保证基类公有成员在子类中可以被访问

**15.一个函数功能不太复杂，但要求被频繁调用，则应把它定义为（A）**

**A.内联函数**

**B.重载函数**

**C.递归函数**

**D.嵌套函数**

16.假设class Y:public X ，即类Y是类X的派生类，则定义一个Y类的对象时和删除Y类对象时，调用构造函数和析构函数的顺序是（A）

A. X,Y; Y,X

B. X,Y; X,Y

C. Y,X; X,Y

D. Y,X; Y,X

17.假定一个类的构造函数A（int aa,int bb）{a=aa;b=a\*bb;}，则执行A x(4,5);语句后，x.a和x.b的值分别为（C）

A.3和5 B.5和4 C.4和20 D.20和5

18．下列不能作为类的成员的是（B）

A.自身类对象的指针

B.自身类对象

C.自身类对象的引用

D.另一个类的对象

19.假定AA为一个类，a()为公有的函数成员，x为该类的一个对象，则访问x对象中函数成员a()的格式为（B）

A.x.a B.x.a() C.x->a D.(\*a).a()

20.关于对象概念的描述中，说法错误的是（A）

A.对象就是C语言中的结构变量

B.对象代表着正在创建的系统中的一个实体

C.对象是类的一个变量

D.对象之间的信息传递是通过消息进行的

21.编写C++程序一般要经过的几个步骤依次是（B）

1. 编辑、调试、编译、连接
2. 编辑、编译、连接、运行
3. 编译、调试、编辑、连接
4. 编译、编辑、连接、运行

22.下面叙述不正确的是（D）

1. 派生类一般都是公有派生
2. 对基类成员的访问必须是无二义性的
3. 赋值兼容规则也适用于多重继承的组合
4. 基类的公有成员在派生类中仍然是公有的

23.所谓数据封装就是将一组数据和与这组数据有关的操作组装在一起，形成一个实体，这个实体也就是（A）

1. 类
2. 对象
3. 函数体
4. 数据块

24.在公有派生类的成员函数不能直接访问基类中继承下来的某个成员，则该成员一定是基类中（A）

A．私有成员

B．公有成员

C．保护成员

D．保护成员或私有成员

25.对基类和派生类的关系描述中，错误的是（B）

A．派生类是基类的具体化

B．基类继承了派生类的属性

C．派生类是基类定义的延续

D．派生类是基类的特殊化

26.关于this指针使用说法正确的是（A）

A．保证每个对象拥有自己的数据成员，但共享处理这些数据的代码

B．保证基类私有数据成员在子类中可以被访问

C．保证基类保护数据成员在子类中可以被访问

D．保证基类公有数据成员在子类中可以被访问

27.所谓多态是指（B）

A．不同的对象调用不同名称的函数

B．不同的对象调用相同名称的函数

C．一个对象调用不同名称的函数

D．一个对象调用不同名称的对象

28.一个函数功能不太复杂，但要求被频繁调用，则应把它定义为（A）

A．内联函数

B．重载函数

C．递归函数

D．嵌套函数

29.在类中说明的成员可以使用关键字的是（A）

A．public

B．extern

C．cpu

D．register

30.下列不能作为类的成员的是（B）

A．自身类对象的指针

B．自身类对象

C．自身类对象的引用

D．另一个类的对象

31.使用地址作为实参传给形参下列说法正确的是（D）

A．实参是形参的备份

B．实参与形参无联系

C．形参是实参的备份

D．实参与形参是同一对象

32.所谓多态是指（B）

A．不同的对象调用不同名称的函数

B．不同的对象调用相同名称的函数

C．一个对象调用不同名称的函数

D．一个对象调用不同名称的对象

33.在C++中，使用流进行输入输出，其中用于屏幕输入的是（A）

A．cin

B．cerr

C．cout

D．clog

34.假定AA为一个类.a()为该类公有的函数成员，x为该类的一个对象，则访问x对象中函数成员a()的格式为（B）

A．x.a

B．x.a()

C．x->a

D．(\*x).a()

35.关于对象概念的描述中，说法错误的是（A）

A．对象是C语言中的结构变量

B．对象代表着正在创建的系统中的一个实体

C. 对象是类的一个变量

D. 对象之间的信息传递是通过消息进行的

36.静态成员函数没有（B）

A. 返回值

B. this指针

C. 指针参数

D. 返回类型

37.假定AB为一个类，则执行“AB a(2),b[3],\*p[4];”语句时调用构造函数的次数为（B）

A．3

B．4

C．5

D．9

**38.有关多态性说法不正确的是（C）**

**A.C++语言的多态性分为编译时的多态性和运行时的多态性**

**B.编译时的多态性可以通过函数重载实现**

**C.运行时的多态性可以通过模板和虚函数实现（基类指针或基类引用和虚函数实现）**

**D.实现运行时的多态性的机制称为动态多态性**

39.假定一个类的构造函数为“A(int i=4,int j=0){a=i,b=j;}”，则执行“A x(1);”语句后，x.a和x.b的值分别为（A）

A.1和0

B.1和4

C.4和0

D.4和1

40.类MyA的拷贝初始化构造函数是（C）

A. MyA（）

B.MyA（MyA\*）

C.MyA（MyA&）

D.MyA(MyA)

41.C++中函数原型不能标识（D）

A.函数的返回类型

B.函数参数的个数

C.函数参数类型

D.函数的功能

42.友元关系不能是（D）

A.提高程序的运行效率

B.是类与类的关系

C.是一个类的成员函数与另一个类的关系

D.继承

43.实现两个相同类型数加法的函数模板的声明是（D）

A.add(T x,T y)

B.T add(x,y)

C.T add(T x,y)

D.T add(T x,T y)

**44.下列不是描述类的成员函数的是（C）**

**A.构造函数**

**B.析构函数**

**C.友元函数**

**D.拷贝构造函数**

45.下面有关类的说法不正确的是（C）

A.一个类可以有多个构造函数

B.一个类只有一个析构函数

C.析构函数需要指定参数

D.在一个类中可以说明具有类类型的数据成员

46.下列有关重载函数的说法中正确的是（C）

A.重载函数必须具有不同的返回值类型

B.重载函数参数个数必须相同

C.重载函数必须有不同的形参列表

D.重载函数名可以不同

47.this指针存在的目的是（C）

A.保证基类私有成员在子类中可以被访问

B.保证基类保护成员在子类中可以被访问

C.保证每个对象拥有自己的数据成员，但共享处理这些数据成员的代码

D.保证基类公有成员在子类中可以被访问

13.关于new运算符的下列描述中，错误的是（D）

A.它可以用来动态创建对象和对象数组

B.使用它创建的对象或对象数组可以使用运算符delete删除

C.使用它创键对象时要调用构造函数

D.使用它创键对象数组时必须指定初始值

48．已知：p是一个指向类A数据成员m的指针，A1是类A的一个对象。如果要给m赋值为5，正确的是（C）

A.A1.p=5;

B.A1->p=5;

C.A1.\*p=5;

D.\*A1.p=5;

分析：A中p是指针即地址，错误；B选项中A1不是指针不能使用指向运算符->,错误；

“\*”比“.”级别要高，所以D选项\*A1.p=5相当于（\*A1）.p=5;错误。

49.以下基类中的成员函数表示纯虚函数的是（A）

A.virtual void tt()=0;B.void tt(int)=0;

C.virtual void tt(int);

D.virtual void tt(int) {}

50.C++体系中，不能被派生类继承的有（B）

A.常成员函数

B.构造函数

C.虚函数

D.静态成员函数

51.静态成员函数不能说明为（C）

A.整型函数

B.浮点函数

C.虚函数

D.字符型函数

52.说明内联函数的关键字是（ A ）。

A. inline B. virtual C. define D. static

53.假定CAb为一个类，则执行CAb oX；语句时将自动调用该类的（B ）

A. 有参构造函数 B. 无参构造函数

C. 拷贝构造函数 D. 赋值重载函数

54.cin是某个类的标准对象的引用，该类是（ B ）。

A. ostream B. istream C. stdout D. stdin

55.下面的哪个保留字不能作为函数的返回类型？（ C ）

A. void B. int C. new D. long

56.不能参与重载的运算符是（ A ）。

A. 类 B. 函数 C. 函数模板 D. 运算符

57.由于数据隐藏的需要，静态数据成员通常被说明为（ A ）。

A. 私有的 B. 公有的 C. 保护的 D. 不可访问的

58.编译时多态性使用什么获得？（ A ）

A. 重载函数 B. 继承 C. 虚函数 D. B和C

59.拷贝构造函数的参数通常是（ C ）。

A. 无特殊要求 B. 指向对象的指针

C. 自已类对象的常引用 D. 对象

60.C++有几种联编？（ B ）

A. 1种 B. 2种 C. 3种 D. 4种

61.基类和派生类可以分别称为（ B ）。

A. “大类”和“小类” B. “父类”和“子类”

C. “小类”和“大类” D. “子类”和“父类”

**62.下列对类的构造函数和析构函数描述正确的是（ A ）。**

**A)构造函数可以重载，析构函数不能重载**

**B)构造函数不能重载，析构函数可以重载**

**C)构造函数可以重载，析构函数可以重载**

**D)构造函数不能重载，析构函数不能重载**

63.在函数定义前加上关键字“inline”，表示该函数被定义为（ B ）。

A）重载函数 B）内联函数

C）成员函数 D）普通函数

64.下面有关重载函数的说明中， （ C ）是正确的。

A) 重载函数必须具有不同的返回值类型

B) 重载函数形参个数必须不同

C) 重载函数必须具有不同的形参列表

D) 重载函数名可以不同

65.已知: Print( )函数是一个类的常成员函数，它无返回值，下列表示中，正确的是 ( A )。

A）void Print( ) const； B) const void Print( )；

C) void const Print( )； D) void Print(const)

66.假定Myclass为一个类,那么下列的函数说明中( D )为该类的析构函数。

A) void ～Myclass( ); B) ～Myclass( int n);

C) Myclass( ); D) ～Myclass( )

67.下面类的定义中有( C ) 处错误。

class myclass{int i=0; public: void myclass( );

~myclass(value); }

A) 1 B)2 C)3 D)4

68.说明虚函数的关键字是（ B ）。

A. inline B. virtual C. define D. static

69.cout是某个类的标准对象的引用，该类是（ A ）。

A. ostream B. istream C. stdout D. stdin

70.一个类的所有对象共享的是（ D ）。

A. 私有数据成员 B. 公有数据成员

C. 保护数据成员 D. 静态数据成员

71.静态成员函数没有（ B ）。

A. 返回值 B. this指针 C. 指针参数 D. 返回类型

72.关于C++与C语言关系的描述中，（ D）是错误的。

A.C语言是C++语言的一个子集

B.C语言与C++语言是兼容的

C.C++语言对C语言进行了一些改进

D.C++语言和C语言都是面向对象的

73.按照标识符的要求，（A ）符号不能组成标识符。

A.连接符 B.下划线

C.大小写字母 D.数字字符

74.为了避免嵌套的if-else语句的二义性，C语言规定else总是与（ C）组成配对关系。

A.缩排位置相同的if B.在其之前未配对的if

C.在其之前未配对的最近的if D.同一行上的if

75.在"int a[ ][3]={{1},{3,2},{4,5,6},{0}};"中，a[2][2]的值是（ C）。

A.1 B.0 C.6 D.2

76.设"char \*\*s；"，以下正确的表达式是（B）。

A.s="computer"; B.\*s="computer";

C.\*\*s="computer"; D.\*s='c';

77.对于"int \*pa[5]；"的描述中，（D ）是正确的。

A.pa是一个指向数组的指针，所指向的数组是5个int型元素

B.pa是一个指向某数组中第5个元素的指针，该元素是int型变量

C.pa [5]表示某个元素的第5个元素的值

D.pa是一个具有5个元素的指针数组，每个元素是一个int型指针

78.在下列表示引用的方法中，（A ）是正确的。

　　已知：int m=10;

A.int &x=m; B.int &y=10;

C.int &z; D.float &t=&m;

79.下列for循环的次数为（ B）。

　　for（i=0, x=0; !x&&i<=5; i++）

A.5 B.6 C.1 D.无限

80.对于C/C++语言的函数，下列叙述中正确的是（A）。

A.函数的定义不能嵌套，但函数调用可以嵌套

B.函数的定义可以嵌套，但函数调用不能嵌套

C.函数的定义和调用都不能嵌套

D.函数的定义和调用都可以嵌套

81.在一个被调用函数中，关于return语句使用的描述，（ D）是错误的。

A.被调用函数中可以不用return语句

B.被调用函数中可以使用多个return语句

C.被调用函数中，如果有返回值，就一定要有return语句

D.被调用函数中，一个return语句可以返回多个值给调用函数

82.在一个函数中，要求通过函数来实现一种不太复杂的功能，并且要求加快执行速度，选用（ A）。

A内联函数 B.重载函数

C.递归调用 D.嵌套调用

83.使用fseek函数可以实现的操作是（ A）。

A.改变文件指针的当前位置

B.文件的顺序读写

C.文件的随机读写

D.以上都不对

84.下列存储标识符中，（ C）的可见性与存在性不一致。

A.外部类 B.自动类

C.内部静态类 D.寄存器类

85.在如下结构定义中，不正确的是（ B）。

A.struct student

　　　{int no;

　　　　char name[10];

　　　　float score;

　　　};

B.struct stud[20]

　　　{

　　　　int no;

　　　　char name[10];

　　　　float score;

　　　};

C.struct student

　　　{

　　　　int no;

　　　　char name[10];

　　　　float score;

　　　} stud[20];

D.struct

　　　{

　　　　int no;

　　　　char name[10];

　　　　float score;

　　　}stud[100] ;

86.将两个字符串连接起来组成一个字符串时，选用（C ）函数。

A.strlen( ) B.strcpy( )

C.strcat( ) D.strcmp( )

87. ( D)不是构造函数的特征

A.构造函数的函数名与类名相同

B.构造函数可以重载

C.构造函数可以设置缺省参数

D.构造函数必须指定类型说明

88.已知：类A中一个成员函数说明如下：

　　void Set(A& a);

　　其中，A&的含义是(C )。

A.指向类A的指针为a

B.将a的地址值赋给变量Set

C.a是类A对象的引用，用来作函数Set（）的参数

D.变量A与a按位与作为函数Set( )的参数

89.已知：print( )函数是一个类的常成员函数，它无返回值，下列表示中，（ A）是正确的。

A.void print( ) const; B.const void print( );

C.void const print( ); D.void print(const);

**90.关于虚函数的描述中，（C ）是正确的。**

**A.虚函数是一个static类型的成员函数**

**B.虚函数是一个非成员函数**

**C.基类中说明了虚函数后，派生类中将其对应的函数可不必说明为虚函数**

**D.派生类的虚函数与基类的虚函数具有不同的参数个数和类型**

91.关于new运算符的下列描述中，（ D）是错的。

A.它可以用来动态创建对象和对象数组

B使用它创建的对象和对象数组可以使用运算符delete删除

C.使用它创建对象时要调用构造函数

D.使用它创建对象数组时必须指定初始值

92.‘A’的ASCII码为65，n为int型，执行 n =‘A’+‘6’-‘3’；后，n的值为（B）。

A）‘D’ B) 68 C) 不确定的值 D) 编译出错

93.下列变量名中， （ A ）是合法的。

A）CHINA B) byte-size C) double D) A+a

94.在static int a[3][3]={{1}，{3，2}，{4，5，6}}；中, a[2][2]的值是 （ C ）。

A）0 B) 5 C）6 D）2

95.若有定义 int a=3, \*p=&a ；则 \*p的值是 （ B ）。

A）常量a的地址值 B）3

C）变量p的地址值 D）无意义

96.下列关于指针运算的各叙述中，不正确的叙述是 （ D ）。

A）指向同一数组的两个指针，可以进行相等或不等的比较运算；

B）可以用一个空指针赋值给某个指针；

C）指向数组的指针，可以和整数进行加减运算；

D）指向同一数组的两个指针，可以进行有意义的相加运算。

97.已知x、y、z是int型变量，且x=3，y=4，z=5；则下面各表达式中，值为0的是（ D ）。

A）‘x’&&‘y’ B） x<=y C）x || y+z && y-z D) !((x<y) && !z || 1)

98.下列各选项中，正确的选项是（ A ）。

A) 0和‘\0’值相等。 B) 0和‘0’ 值相等 。

C)‘\0’和 ‘0’ 值相等 。 D) 0、‘\0’和‘0’ 值都不相等。

99.若有说明语句：char ch=‘\72’；则变量 ch （A）。

A）包含1个字符 B) 包含2个字符

C) 包含3个字符 D) 说明不合法

100.在下列关于C++与C语言关系的各描述中， D 是错误的。

A）C语言是C++的一个子集； B）C语言与C++是兼容的；

C）C++对C语言进行了一些改进； D） C++和C语言都是面向对象的。

101.在下列关于类概念的各描述中， A 是错误的。

A）类就是C语言中的结构体类型；

B）类是具有共同行为的若干对象的统一描述体；

C）类是创建对象的样板；

D）类是抽象数据类型的实现。

102.在下列各选项中，符合C++语法的赋值表达式是 C 。

A） d=3+=d+6 B） d=6+e=d+8

C） d=e+=f+6 D） d=2+e, d+5=1

103.对 for（ 表达式1 ； ；表达式3 ） 可理解为 B 。

A）for（ 表达式1 ；0 ；表达式3 ）

B）for（ 表达式1 ；1 ；表达式3 ）

C）for（ 表达式1 ；表达式1 ；表达式3 ）

D）for（ 表达式1 ；表达式3 ；表达式3 ）

104.若有定义：int a[3][4] ； 则对a数组某一元素的正确表示是 C 。

A）a[2][4] B) a[1,3] C）a[1+1][0] D) a(2)(1)

105.若有初始化 static char str[]=“NDJSZX” ；则其中的str[]相当于 B 。

A）str[8] B) str[7] C) str[6] D) str[2]

106.一维整型数组a的正确定义是 D 。

A) int a(10)；

B) int n=10； int a[n]；

C) int n； cin>>n； int a[n]；

D) const int SIZE=10； int a[SIZE]；

107.采用函数重载的目的在于 D ；

A) 实现共享 B）减少空间

C) 提高速度 D）使用方便，提高可读性

108.下列的各类函数中， C 不是类的成员函数。

A）构造函数 B）析构函数

C）友元函数 D）拷贝初始化构造函数

109.若有定义：int x , \*p ； 则正确的赋值表达式是 D 。

A) \*p=\*x B) \*p=&x C) p=x D) p=&x

**110.下列静态数据成员特性中， D 是错误的。**

**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) 引用静态数据成员时，要在静态数据成员名前加<类名>和作用域分辨符；**

**D) 静态数据成员不是所有对象共有的。**

111.以下各选项中，正确的函数声明是 C 。

A）double func( int x ， int y )

B）int func( int x ； int y ) ；

C）float func( int x ， int y ) ；

D）char func( int x ， y ) ；

**112.在C++中，函数原型不能标识(D )**

**A.函数的返回类型 B.函数参数的个数**

**C.函数参数类型 D.函数的功能**

113.在C++程序中，对象之间的相互通信通过(B )

A.继承实现 B.调用成员函数实现

C.封装实现 D.函数重载实现

114.对于任意一个类，析构函数的个数最多为( B)

A.0 B.1 C.2 D.3

115.下面函数模板定义中不正确的是（A）

A.template<class Q> B.template<class Q>

Q F(Q x){ Q F(Q x){

return Q+x; return x+x;

} }

C.template<class T> D.template<class T>

T F(T x){ T F(T x){

return x\*x; return x>1;

} }

116.友元关系不能(A )

A.继承

B.是类与类的关系

C.是一个类的成员函数与另一个类的关系

D.提高程序的运行效率

**117.语句ofstream f(″SALARY.DAT″,ios::app｜ios::binary);的功能是建立流对象f，试图打开文件SALARY.DAT 并与之连接，并且(A )**

**A.若文件存在，将文件写指针定位于文件尾；若文件不存在，建立一个新文件**

**B.若文件存在，将其置为空文件；若文件不存在，打开失败**

**C.若文件存在，将文件写指针定位于文件首；若文件不存在，建立一个新文件**

**D.若文件存在，打开失败；若文件不存在，建立一个新文件**

118.下面说法正确的是(B )

A.内联函数在运行时是将该函数的目标代码插入每个调用该函数的地方

B.内联函数在编译时是将该函数的目标代码插入每个调用该函数的地方

C.类的内联函数必须在类体内定义

D.类的内联函数必须在类体外通过加关键字inline 定义

119.可以用p.a 的形式访问派生类对象p 的基类成员a，其中a 是(D )

A.私有继承的公有成员 B.公有继承的私有成员

C.公有继承的保护成员 D.公有继承的公有成员

120.在公有派生情况下，有关派生类对象和基类对象的关系，不正确的叙述是(C )

A.派生类的对象可以赋给基类的对象

B.派生类的对象可以初始化基类的引用

C.派生类的对象可以直接访问基类中的成员

D.派生类的对象的地址可以赋给指向基类的指针

121.对于类定义

class A{

public:virtual void func1( ){ }

void func2( ){ }

};

class B:public A{

public:void func1( ){cout<<″class B func 1″<<end1;}

virtual void func2( ){cout<<″class B func 2″<<end1;}

};

下面正确的叙述是(C )

A. A::func2( )和B::func1( )都是虚函数

B. A::func2( )和B::func1( )都不是虚函数

C. B::func1( )是虚函数，而A::func2( )不是虚函数

D. B::func1( )不是虚函数，而A::func2( )是虚函数

122.派生类对象可访问基类中的什么成员( A )

A.公有继承的公有成员 B.公有继承的私有成员

C.公有继承的保护成员 D.私有继承的公有成员

123.定义析构函数时，应该注意( C )。

A.其名与类名完全相同 B.返回类型是void类型

C.无形参，也不可重载 D.函数体中必须有delete语句

124.如果类A被说明成类B的友元，则( D )。

A.类A的成员即类B的成员

B.类B的成员即类A的成员

C.类A的成员函数不得访问类B的成员 D.类B不一定是类A的友元

125.应在下列程序划线处填入的正确语句是（ C ）

#include

class Base

{ 　public:

void fun(){cout<<"Base::fun"<<endl;}

};

class Derived:public Base

{ 　void fun()

{\_\_\_\_\_\_\_\_\_\_\_\_\_ // 显示调用基类的函数fun()

cout<<"Derived::fun"<<ENDL;

}

};

A．fun(); B. Base.fun(); C. Base::fun(); D. Base->fun();

126.面向对象程序设计将数据与( A )放在一起，做为一个相互依存、不可分割的整体来处理。

A. 对数据的操作 B. 信息 C. 数据隐藏 D. 数据抽象

127.在类中声明转换函数时不能指定( A )。

A. 参数 B. 访问权限

C. 操作 D. 标识符

128.在派生类中重新定义虚函数时必须在( A )方面与基类保持一致。

A. 参数类型 B. 参数名字

C. 操作内容 D. 赋值

129.下面关于C++中类的继承与派生的说法错误的是（ C ）。

A．基类的protected成员在公有派生类的成员函数中可以直接使用

B．基类的protected成员在私有派生类的成员函数中可以直接使用

C．公有派生时，基类的所有成员访问权限在派生类中保持不变

D．基类的protected成员在保护派生类的成员函数中可以直接使用

130.重载赋值操作符时，应声明为( C )函数。

A. 友元 B. 虚 C. 成员 D. 多态

131.在下面选项中，对类的拷贝构造函数的声明形式是（ B ）

A. A::A(&) B. A::A(const A &) C. A::A(A) D. void A::A(A &a)

132.下面关于成员函数特征的描述中，错误的是（ A ）

A．成员函数不可以设置参数的默认值 B．成员函数可以重载

C．成员函数可以是内联函数 D．成员函数可以是静态的

133.下面关于类模板叙述不正确的是（ B ）

A．可以有多个构造函数 B. 类模板是模板类的实例

C. 一个类模板可以用来实例化多个模板类 D. 类模板可以有多个参数

**134.要使一个类成为抽象类，这个类应具有的条件是（ A ）**

**A．至少有一个纯虚函数 B. 至少有一个虚函数**

**C. 只含有一个纯虚函数 D. 只含有一个虚函数**

135.有关运算符重载正确的描述是（ D ）

A. C++语言允许在重载运算符时改变运算符的操作个数

B. C++语言允许在重载运算符时改变运算符的优先级

C. C++语言允许在重载运算符时改变运算符的结合性

D. C++语言允许在重载运算符时改变运算符的原来的功能

136.下列说法正确的是（ D ）。

A. 一个类的对象可以是另一个类的成员

B. 一个类可以作为另一个类的派生类

C. 一个类可以在另一个类定义体进行定义

D. 以上说法均正确

**137.有如下程序：**

**#include <iostream.h>**

**class BASE{**

**char c;**

**public:**

**BASE(char n):c(n){}**

**virtual ~BASE(){cout<<c;}**

**};**

**class DERIVED:public BASE**

**{**

**char c;**

**public:**

**DERIVED(char n):BASE(n+1),c(n){}**

**~DERIVED(){cout<<c;}**

**};**

**int main()**

**{**

**DERIVED('X');**

**return 0;**

**}**

执行上面的程序将输出（ C ）。

A)X B)YX C)XY D)Y

138.类MyClass的定义如下：

class MyClass

{ public:

MyClass(){value=0;}

SetValue(int i){value=i;}

private:

int value;

};

则对语句：MyClass \*p,my;p=&my;正确的描述是（ D ）。

A. 语句p=&my;是把对象my赋值给指针变量p；

B. 语句MyClass \*p,my;会调用两次类MyClass的构造函数

C. 对语句\*p.SetValue(5)的调用是正确的

D. 语句p->SetValue(5)与my.SetValue(5)等价

139.下列各类函数中，（C）不是类的成员函数。

A. 构造函数 B. 析构函数 C. 友元函数 D. 拷贝构造函数

140.派生类的对象对它的基类成员中（A）是可以访问的。

A. 公有继承的公有成员 B. 公有继承的私有成员

C. 公有继承的保护成员 D. 私有继承的公有成员

141.下面有关构造函数和new运算符关系正确的说法是（ D ）

A．new运算符不调用构造函数 B. 构造函数一定调用new运算符

C．当生成新类的实例时，先调用new运算符，然后调用构造函数进行初始化

D. 当new运算符动态产生类的对象时，new运算符也自动调用构造函数

142.下面有关类性质的说法错误的是（ D ）

A.一个类可以有多个构造函数，但只有一个析构函数

B.析构函数和构造函数都不能有返回类型

C.不能给析构函数指定参数

D.一个类中不可以声明具有类类型的数据成员

143.面关于友元函数描述正确的是（ C ）

A．友元函数是类的成员

B．友元函数在类的外部声明

C．友元函数在类中声明

D. 友元函数能够派生

144.下面叙述正确的是（ D ）

A. 派生类不可以使用私用派生

B. 保护派生时，基类的保护成员在派生类中是公有的

C. 对基类成员的访问允许二义性

D. 赋值兼容规则也适用于多重继承的组合

**145.有关运算符重载正确的描述是（ D ）**

**A. C++语言允许在重载运算符时改变运算符的操作个数**

**B. C++语言允许在重载运算符时改变运算符的优先级**

**C. C++语言允许在重载运算符时改变运算符的结合性**

**D. C++语言允许在重载运算符时改变运算符的原来的功能**

146.下面描述中，表达错误的是（ B ）

A. 公有继承时基类中的public成员在派生类中仍是public的

B. 公有继承时基类中的private成员在派生类中仍是private的

C. 公有继承时基类中的protected成员在派生类中仍是protected的

D. 私有继承时基类中的public成员在派生类中是private的

147.拷贝构造函数的参数是（ C ）

A. 某个对象名 B. 某个对象的成员名

C. 某个对象的引用名 D. 某个对象的指针名

148.下列关于构造函数的描述中，错误的是（ D ）

A．构造函数可以设置默认参数　 B.构造函数在定义类对象时自动执行

C．构造函数可以是内联函数　 D.构造函数不可以重载

149.数组作为函数的形参时，把数组名作为实参，传递给函数的是（A ）

A．该数组的首地址 B．该数组的元素个数

C．该数组中的各元素值 D．该数组的大小

150.通过一个析构函数调用虚函数时，C++系统对该调用采用(B　)。

A. 动态联编 B. 静态联编 C. 不确定是哪种联编 D.函数重载

151.下面有关重载函数的说法中正确的是（ C ）

A．重载函数必须具有不同的返回值类型；

B．重载函数形参个数必须不同；

C．重载函数必须有不同的形参列表

D．重载函数名可以不同；

152.使用string类建立对象的不正确方式是（D）。

A. string str(“OK”)； B. string str=”OK”；

C. string str； D. string str=’OK’;

**153.面关于C++中类的继承与派生的说法错误的是( C )**

**A．基类的protected成员在公有派生类的成员函数中可以直接使用**

**B．基类的protected成员在私有派生类的成员函数中可以直接使用**

**C．私有派生时，基类的所有成员访问权限在派生类中保持不变**

**D．继承可以分为单一继承与多重继承**

154.面向对象程序设计将数据与( A )放在一起，做为一个相互依存、不可分割的整体来处理。

A. 对数据的操作 B. 信息 C. 数据隐藏 D.数据抽象

155.一个类的析构函数 （A）。

A.唯一的 B.允许重载 C.至多可有两个 D.只能是缺省的

156.下列各类函数中，( C )不是类的成员函数。

A. 构造函数 B. 析构函数 C. 友元函数 D. 拷贝构造函数

157.在下面选项中，对类的拷贝构造函数的声明形式是（ B ）

A．A::A(&) B. A::A(constA&) C. A::A(A) D. void A::A(A&a)

158.下面关于静态成员描述正确的是（ B ）

A. 静态成员是对象的数据成员

B. 静态成员是对象的成员

C. 静态成员是对象的成员函数

D. 静态成员不是对象的成员

159.下面关于关于成员函数特征的描述中，错误的是（ A ）

A．成员函数不可以设置参数的默认值

B．成员函数可以重载

C．成员函数可以是内联函数

D．成员函数可以是静态的

160.下面叙述正确的是（ D ）

A．基类的保护成员在派生类中仍然是保护的

B．基类的公有成员在派生类中仍然是公有的

C．基类的私有成员在派生类中是私有的

D．从基类派生时，可以有3种派生方法

161.关于常成员的描述中，错误的是（B）

A.常成员包含常数据成员和常成员函数两种

B.常数据成员必须是公有的

C.常数据成员要使用构造函数成员初始化列表进行初始化

D.常对象只能调用它的常成员函数，不能调用其它的成员函数

**162.对于动态分配内存空间描述正确的是（D）**

**A.使用new运算符分配的内存空间的长度必须是常量**

**B.delete运算符可以释放动态的存储空间和静态的存储空间**

**C.有new分配的内存空间是不连续的**

**D.delete运算符只能释放由new分配的动态存储空间**

163.C++中串流类是在头文件strstream.h中定义的，下列不属于串流类的是（C）

A.strstream B.ostrstream C.ofstream D.isstrstream

164.下列关于基类和派生类的描述中，错误的是（ C ）。

A．一个基类可以生成多个派生类 B．基类中所有成员都是它的派生类的成员

C．基类中成员访问权限继承到派生类中不变 D．派生类中除了继承的基类成员还有自己成员

165．下列关于派生类的描述中，错误的是（ D ）。

A．派生类至少有一个基类 B．一个派生类可以作另一个派生类的基类

C．派生类的构造函数中应包含直接基类的构造函数 D．派生类默认的继承方式是public

166.下列关于子类型的描述中，错误的是（ B ）。

A．在公有继承下，派生类是基类的子类型

B．如果类A是类B的子类型，则类B也是类A的子类型

C．如果类A是类B的子类型，则类A的对象就是类B的对象

D．在公有继承下，派生类对象可以初始化基类的对象引用

167.派生类构造函数的成员初始化列表中，不能包含的初始化项是（ B ）。

A．基类的构造函数 B．基类的子对象 C．派生类的子对象 D．派生类自身的数据成员

168.执行以下程序： char \*str; cin>>str; cout<<str; 若输入abcd 1234↙则输出（D）。

A．abcd B．abcd 1234 C．1234 D．输出乱码或出错

169.执行下列程序：char a[200]; cin.getline(a,200,’‘); cout<<a; 若输入abcd 1234↙则输出（A）。

A．abcd B．abcd 1234 C．1234 D．输出乱码或出错

170.以下程序执行结果（A）。

cout.fill(‘#’);

cout.width(10);

cout<<setiosflags(ios::left)<<123.456;

A．123.456### B．123.4560000 C．####123.456 D．123.456

171.当使用ifstream定义一个文件流，并将一个打开文件的文件与之连接，文件默认的打开方式为（A）。

A．ios::in B．ios::out C．ios::trunc D．ios::binary

172.read函数的功能是从输入流中读取(D )。

1. 一个字符 B.当前字符 C.一行字符 D.指定若干字节

三、填空题

**1.类和对象的关系可表述为：类是对象的抽象，而对象则是类的实例。**

2.如果只想保留公共基类的一个复制，就必须使用关键字virtual把这个公共基类声明为虚基类。

3.类的静态成员分为静态数据成员和静态成员函数。

4.若要把void fun()定义为类A的友元函数，则应在类A的定义中加入语句friend void fun(A &a)。

5.静态成员函数没有隐含的this指针。所以，静态成员函数只能访问类的静态数据成员，非静态成员函数访问类的非静态数据成员。

6.运算符重载要求保持其原来的操作数个数、结合性、优先级和语法结构。

7.通过关键字Template 可以声明模板，通过关键字class(或typename)指定函数模板的类型参数，有几个类型参数就有几个类型关键字。

8.列出C++中两种用户自定义的数据类型：类、结构体。

9.C++体系中，不能被派生类继承的有构造函数。构造函数的作用是在创建对象时初始化对象的数据成员。析构函数的作用是在对象被系统释放前，清理内存。

10.后置自增运算符“++”重载为类的成员函数（设类名为A）的形式为A operator++(int)。

11.从现实的角度来讲，多态性可以划分为两类：静态多态性和动态多态性。

12.列出C++中的两种代码复用方式：继承和组合或模板。

13.假定AB是一个类，则该类的复制构造函数的函数头为：AB(AB&)。

14.后置自增运算符“++”重载为类的成员函数（设类名为A）的形式为：A operator++(int)。AA operator+(AA x)表示为运算符+重载。

15.对象将其大部分实现细节隐藏起来，这种机制称为封闭。

16.建立类和类的对象时，系统对类不分配内存，系统对对象分配内存。

17.基类和派生类的关系称为继承和派生。

18.当保护继承时，基类的公有或保护成员在派生类中称为保护成员，派生类对象不能直接访问基类的任何成员。

19.C++支持两种多态性：编译时的多态性（重载）和运行时的多态性（虚函数）。

20.在C++中定义类时，通常是利用变量描述对象的“特性”；利用函数描述对象的“行为”。

21.在定义对象的同时初始化其成员变量，需要利用构造函数，用于初始化变量的值可以是固定的常量，也可以通过参数获得。

22.在私有继承关系下，基类的公有成员和保护成员将成为类中的私有成员，它们只能由派生类的成员函数访问；基类的私有成员将成为派生类中的不可访问的成员。

23.面向对象方法构造的系统，其基本单位是对象。

24.若采用x.abc(y)表达式调用一个成员函数，在成员函数中使用的\*this就代表了类外的x对象。

25.在一个类中定义复制构造函数的目的，是为了当利用该类的一个对象初始化另一个对象时，能够避免这两个对象的同一指针同时指向同一块地址。

26.若多个基类及其派生类中都定义了同名的函数成员，要访问相应函数时，就需要在函数名前加上基类名和类区分符。

**27.在每个成员函数中，隐含的第一个参数的参数名为this指针。**

28.C++封装性、继承性和多态性是面向对象思想的主要特征。

29.C++语言提供了处理异常的有效机制，通过throw抛出异常，通过try检测异常，通过catch捕捉并处理异常。

30.类的成员有3中访问属性，其默认的访问属性为private；访问属性为protected的成员既可被派生类继承又保持了封装和信息隐藏性。

31.对于已存在的定义“int \*ptr=new int[10];”，应该用delete []ptr;语句释放动态内存空间。

32.通过在虚函数参数表后加virtual，可以定义为纯虚函数。含有纯虚函数的类称为抽象类。

33.在继承中，如果只有一个基类。则这种继承方式称为单一继承，如果基类名有多个，则这种继承方式称为多重继承。

34.基类中的私有成员不能被派生类的成员访问。

35.带有纯虚函数的类称为抽象类，它只能作为基类来使用。

36.在一个派生类中，对基类成员、类对象成员和非类对象成员的初始化次序是基类、后组合类、最后为自身类。

37.虚基类的作用是消除派生类中来自基类成员而产生的二义性。

38.不能重载的运算符有.(类成员运算符)、.\*(类成员指针运算符)、::(类作用域运算符)、?:(条件运算符)和sizeof(求字节数运算符)。

39.不能重载为友元函数的运算符有=、()、[]、->。

40.当程序与外界环境进行信息交换时，存在着两个对象，一个是程序中对象，另一个是文件对象。

41.根据对字节内容的解释方式，字节流分为字符流和二进制流。

42.系统在主存中开辟的、用来临时存放输入输出数据的区域，称为缓冲区。

43.可以使用命名空间来解决潜在的名称冲突问题。

44.可以使用的标准输入输出流有cin、cout、cerr、clog、其中cerr为非缓冲区。

45.模板可以用一个代码段指定一组相关函数，称为函数模板；或者一组相关类，称为类模板。

**46.从一个函数模板产生的相关函数都是同名的，编译器用函数参数类型的解决方法调用相应的函数。**

47.所有的函数模板都是以关键字template开始的，关键字之后用尖括号<>括起来的是形式参数表。

48.常量对象必须初始化，并且不能在生成后修改。

49.new运算符优越于malloc的地方在于：以new分配空间后，还会调用类的构造函数，所以new可以实现与使用局部对象一样的效果。

50.每个对象都是所属类的一个（实例）。

**51.C++的流库预定义了4个流，它们是cin、cout、clog、（cerr）。**

52.定义类的动态对象数组时，系统只能够自动调用该类的（无参）构造函数对其进行初始化。

53.在C++中访问一个指针所指向的对象的成员所用的指向运算符是（-））。

54.假如一个类的名称为MyClass,使用这个类的一个对象初始化该类的另一个对象时，可以调用（复制或拷贝）构造函数来完成此功能。

55.对赋值运算符重载时，应声明为（类成员）函数。

**56.如果要把A类成员函数f()且返回值为void声明为类B的友元函数，则应在类B的定义中加入的语句（friend void A::f();）。**

57.C++语言支持两种多态性分别是编译时的多态性和（运行时）的多态性。

58.在面向对象的程序设计中，将一组对象的共同特性抽象出来形参（类）。

59.定义类动态数组时，元素只能靠自动调用该类的（无参构造函数）来进行初始化。

60.抽象类中至少要有一个（纯虚函数）。

61.一个抽象类的派生类可以实例化的必要条件是实现了所有的（纯虚函数的定义）。

62.静态成员函数、友元函数、构造函数和析构函数中，不属于成员函数的是（友元函数）。

63.由const修饰的对象称为（常对象）。

64.C++程序设计中，建立继承关系倒挂的树应使用（单继承）。

65.基类的公有成员在派生类中的访问权限由（访问控制方式或继承方式）决定。

66.不同对象可以调用相同名称的函数，但执行完全不同行为的现象称为（多态性）。

67.this 指针始终指向调用成员函数的（对象）。

68.类模板用来表达具有（相同处理方法）的模板类对象集。

69.C++语言中如果调用函数时，需要改变实参或者返回多个值，应该采取（传地址或引用）的方式。

**70.如果要把类B的成员函数void fun()说明为类A的友元函数，则应在类A中加入语句（friend void B::fun();）**

**71.设函数max是由函数模板实现的，并且max(3.5, 5)和max(3, 5)都是正确的函数调用，则此函数模板具有（2）个类型参数。**

72.在C++中，函数重载与虚函数帮助实现了类的（多态 ）性。

73.由static修饰的数据成员为该类的所有对象（共享）。

74.重载函数在参数类型或参数个数上不同，但（函数名）相同。

**75.使用new建立的动态对象在不用时必须用（delete ）释放所占用的空间。**

76.定义类的动态对象数组时，系统只能够自动调用该类的无参构造函数对其进行初始

化。

77.在C++程序设计中，建立继承关系倒挂的树应使用\_\_\_单一(或单)\_\_\_\_\_\_继承。

78.表达式cout<<end1 还可表示为\_\_\_ cout<<′\n′(或cout<<′\12′,或cout<<′\xA′,或其它等价形式)\_\_\_\_\_\_。

79.基类的公有成员在派生类中的访问权限由\_\_\_访问控制(或其它等价形式)\_\_\_\_\_\_决定。

80.C++支持的两种多态性分别是编译时的(或静态，或操作重载)多态性和运行时的(或动态，或虚函数)多态性。

81.C++中语句const char \* const p=″hello″；所定义的指针p 和它所指的内容都不能被\_\_\_\_改变(或重新赋值)\_\_\_\_\_。

82.假定AB 为一个类，则语句AB(AB&x)；为该类\_\_\_\_复制初始化(或拷贝)\_\_\_\_\_构造函数的原型说明。

83.将关键字const 写在成员函数的\_\_\_函数头(或参数表) \_和\_\_\_\_\_\_函数体\_\_\_之间时，所修饰的是this 指针。

84.在C++中，访问一个对象的成员所用的运算符是\_\_\_成员选择运算符(或“.”)\_\_\_\_\_\_，访问一个指针所指向的对象的成员所用的运算符是\_\_\_成员访问运算符(或“>”)\_\_\_\_。

85.派生类从一个或多个以前定义的该类的\_基类\_\_\_。

86.若希望用一个已有对象来构造另一个同类型的对象，可以使用\_\_拷贝构造函数\_\_来实现。

87. C++的I/O是以字节流 的形式实现的，每个C++编译系统都带有一个面向对象的输入/输出软件包，这就是 I/O流类库 。

88.当基类中的某个成员函数被声明为虚函数后，此虚函数就可以在一个或多个派生类中被重新定义，在派生类中重新定义时，其函数原型，包括返回类型、函数名、参数个数、参数类型以及参数的顺序都必须与基类中的原型完全相同。

89.静态成员属于类，而不属于任何一个对象，它被同一个类的所有对象共享。

90.重载运算符的含义必须清楚，不能有二义性。

91.类的继承方式有私有继承private、保护继承protected、公有继承public。

**92.从一个函数模板产生的相关函数都是同名的，编译器用\_\_\_\_\_\_函数参数类型\_\_\_的解决方法调用相应的函数。**

**四、阅读程序题**

1.#include<iostream>

using namespace std;

int &f(int &i)

{

i += 10;

return i;

}

int main()

{

int k = 0;

int &m = f(k);

cout << k << endl;

m = 20;

cout << k << endl;

return 0;

}

该程序的运行结果是\_\_\_\_\_\_\_\_\_。

10

20

2、有如下程序：

#include<iostream>

using namespace std;

class Base {

public:

void setx(int i)

{

x = i;

}

int getx()

{

return x;

}

public:

int x;

};

class Derived :public Base {

public:

void sety(int i)

{

y = i;

}

int gety()

{

return y;

}

void show()

{

cout << "Base::x=" << x << endl;

}

public:

int y;

};

int main()

{

Derived bb;

bb.setx(16);

bb.sety(25);

bb.show();

cout << "Base::x=" << bb.x << endl;

cout << "Derived::y=" << bb.y << endl;

cout << "Base::x=" << bb.getx() << endl;

cout << "Derived::y=" << bb.gety() << endl;

return 0;

}

该程序的运行结果是\_\_\_\_\_\_\_\_\_。

Base::x=16

Base::x=16

Derived::y=25

Base::x=16

Derived::y=25

**3.** #include<iostream>

using namespace std;

class N

{

private:

int A;

static int B;

public:

N(int a)

{

A = a;

B += a;

}

static void f1(N m);

};

void N::f1(N m)

{

cout << "A=" << m.A << endl;

cout << "B=" << B << endl;

}

int N::B = 0;

int main()

{

N P(5), Q(9);

N::f1(P);

N::f1(Q);

return 0;

}

该程序的运行结果是\_\_\_\_\_\_\_\_\_。

A=5

B=14

A=9

B=14

4. 有如下程序：

#include<iostream>

using namespace std;

class M {

int x, y;

public:

M()

{

x = y = 0;

}

M(int i, int j)

{

x = i; y = j;

}

void copy(M \*m);

void setxy(int i, int j)

{

x = i; y = j;

}

void print()

{

cout << x << "," << y << endl;

}

};

void M::copy(M \*m)

{

x = m->x;

y = m->y;

}

void fun(M m1, M \*m2)

{

m1.setxy(12, 15);

m2->setxy(22, 25);

}

int main()

{

M p(5, 7), q;

q.copy(&p);

fun(p, &q);

p.print();

q.print();

return 0;

}

该程序的运行结果是\_\_\_\_\_\_\_\_\_。

5,7

22,25

5. #include<iostream>

using namespace std;

class A {

int a, b;

public:

A(int i, int j)

{

a = i; b = j;

}

void move(int x, int y)

{

a += x; b += y;

}

void show()

{

cout << "(" << a << "," <<b<< ")" << endl;

}

};

class B :private A {

int x, y;

public:

B(int i, int j, int k, int l) :A(i, j)

{

x = k; y = l;

}

void show()

{

cout << x << "," << y << endl;

}

void fun()

{

move(3, 5);

}

void f1()

{

A::show();

}

};

int main()

{

A e(1, 2);

e.show();

B d(3, 4, 5, 6);

d.fun();

d.show();

d.f1();

return 0;

}

该程序的运行结果是\_\_\_\_\_\_\_\_\_。

(1,2)

5,6

(6,9)

6.有如下程序：

#include <iostream>

using namespace std;

void swap(int &m, int &n)

{

int temp;

temp = m; m = n; n = temp;

}

int main()

{

int a = 5, b = 10;

cout << "a="<< a <<" b="<< b << endl;

swap(a, b);

cout <<"a="<< a <<" b="<< b << endl;

return 0;

}

该程序的运行结果是\_\_\_\_\_\_\_\_\_。

a=5 b=10

a=10 b=5

7．有如下程序：

#include<iostream>

using namespace std;

class BASE{

char c;

public:

BASE(char n):c(n){}

virtual ~BASE(){cout<<c;}

};

class DERIVED:public BASE

{

char c;

public:

DERIVED(char n):BASE(n+1),c(n){}

~DERIVED(){cout<<c;}

};

int main()

{

DERIVED('X');

return 0;

}

该程序的运行结果是\_\_\_\_\_\_\_\_\_。

XY

8.若有以下程序:

#include <iostream.h>

class Base

{

public:

void Fun() { cout << "1" << endl;}

};

class Derived:public Base

{

public:

void Fun() { cout << "2" << endl; }

};

int main()

{

Derived a;

Base \*p;

p=&a;

p->Fun();

a.Fun();

return 0;

}

上面程序的输出结果为：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1

2

**9.**阅读下面程序，写出输出结果。

#include <iostream>

using namespace std;

class Point

{

public:

Point (int x = 0, int y = 0): m\_x(x), m\_y(y){ }

int GetX() const { return m\_x; }

int GetY() const { return m\_y; }

void SetX(int x) { m\_x = x; }

void SetY(int y) { m\_y = y; }

private:

int m\_x; // X坐标

int m\_y; // X坐标

};

int main(void)

{

Point oPoint1;

const Point oPoint2(3, 4);

cout << oPoint1.GetX() << endl;

oPoint1.SetX(1);

cout << oPoint1.GetX() << endl;

oPoint1.SetY(2);

cout << oPoint1.GetY() << endl;

cout << oPoint2.GetX() << endl;

cout << oPoint2.GetY() << endl;

return 0;

}

上面程序的输出结果为：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0

1

2

3

4

**10.** 阅读下面程序，写出输出结果。

#include<iostream>

using namespace std;

class A

{

int a,b;

public:

A() { a = b = 0; }

A(int aa, int bb)

{

a = aa;

b = bb;

cout << a << ' ' << b << endl;

}

};

int main()

{

A x, y(2,3);

return 0;

}

上面程序的输出结果为：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2 3

**11**．阅读下面程序，写出输出结果。

#include < iostream.h >

void f1( int I ) { I += 10 ; }

void f2( int \*I ) { \*I += 10 ; }

void f3( int& n ) { n += 10 ; }

void main( )

{

int I = 0 ;

f1(I) ;

cout <<"I is "<< I << endl ;

f2( &I ) ;

cout <<"I is "<< I << endl ;

f3(I) ;

cout <<"I is "<< I << endl ;

}上面程序的输出结果为：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

I is 0

I is 10

I is 20

12. 分析程序，写出程序的输出结果（7分）。

#include<iostream.h>

class point{

private:

int x,y;

public:

point(int xx=0,int yy=0){x=xx;y=yy;cout<<”构造函数被调用”<<endl;}

point(point &p);

int GetX(){return x;}

int GetY(){return y;}

};

point::point(point &p)

{x=p.x;

y=p.y;

cout<<”拷贝构造函数被调用”<<endl;

}

void main()

{point A(4,5);

point B(A);

cout<<B.GetX()<<endl;

}

程序的运行结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

构造函数被调用

拷贝构造函数被调用

4

**13.** **#include<iostream.h>**

**int a;**

**class CA {**

**private :**

**int a;**

**public:**

**void SetA() { a = 10; }**

**CA( int i) { a = i; }**

**int GetA() { return ::a; }**

**};**

**int main ()**

**{**

**int a;**

**a = 15;**

**::a = 2;**

**cout <<::a << ","<<a << endl;**

**CA objCA(100);**

**objCA.SetA ();**

**cout <<::a << ","<<a << "," << objCA.GetA() << endl;**

**}**

程序的运行结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2,15

2,15,2

**14.** **#include<iostream.h>**

**class A {**

**public:**

**A( ) { }**

**virtual void func()**

**{ cout << "A::func" << endl; }**

**~A( ) { }**

**virtual void fund( )**

**{ cout << "A::fund" << endl; }**

**};**

**class B:public A {**

**public:**

**B ( ) { func( ) ; }**

**void fun( ) { func( ) ; }**

**~B ( ) { fund( ); }**

**};**

**class C : public B {**

**public :**

**C( ) { }**

**void func( )**

**{cout << "C::func" << endl; }**

**~C() { fund( ); }**

**void fund()**

**{ cout << "C::fund" << endl;}**

**};**

**void main()**

**{ C c; }**

程序的运行结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**A::func**

**C::fund**

**A::fund**

15. #include <iostream.h>

class A {

public :

virtual ~A() {cout<<"DestructA" <<endl; }

};

class B: public A {

public:

virtual ~B() {cout<<"DestructB" << endl; }

};

class C: public B {

public:

~C() { cout << "DestructC" << endl; }

};

void main() {

A \* pa = new C;

delete pa;

A a;

}

程序的运行结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DestructC**

**DestructB**

**DestructA**

**DestructA**

16. #include <iostream.h>

class classa{

public:

classa(){cout<<"基类\n";}

};

class classb:public classa{

public:

classb(){cout<<"派生类\n";}

};

void main(){

classb x;

cout<<"结束\n";

}

程序的运行结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**基类**

**派生类**

**结束**

**17.** **#include<iostream>**

**using namespace std;**

**class name{**

**int i;**

**static int j;**

**public:**

**name(int x=0){i=x;j++;}**

**void display(){**

**cout<<"i="<<i<<",j="<<j<<endl;**

**}**

**};**

**int name::j=0;**

**void main(){**

**name a,b;**

**a.display();**

**b.display();**

**}**

程序的运行结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**i=0,j=2**

**i=0,j=2**

**18.** #include<iostream>

using namespace std;

class A{

public:

A(int i){cout<<"A:"<<i<<endl;}

};

class B{

public:

B(int i){cout<<"B:"<<i<<endl;}

};

class C{

public:

C(){cout<<"C:-"<<endl;}

};

class D:public B,public A,public C{

B member;

public:

D(int a,int b,int c):A(a),member(c),B(b){}

};

void main(){

D obj(1,2,3);

}

程序的运行结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**B:2**

**A:1**

**C:-**

**B:3**

**五、程序设计题**

1、建立类cylinder,cylinder的构造函数被传递了两个double值，分别表示圆柱体的半径和高。用类cylinder计算圆柱体的体积，并存储在一个doubel变量中。在类cylinder中包含了一个成员函数vol，用来显示每个cylinder对象的体积。请用C++编写此程序, 并编写测试程序进行测试。

参考代码如下：

#include<iostream>

using namespace std;

class cylinder {

public:

cylinder(double a, double b);

void vol();

private:

double r, h;

double volume;

};

cylinder::cylinder(double a, double b)

{

r = a; h = b;

volume = 3.1415926\*r\*r\*h;

}

void cylinder::vol()

{

cout << "volume is:" << volume << endl;

}

int main()

{

cylinder x(2.2, 8.09);

x.vol();

return 0;

}

2、给出下面的抽象基类container:

class container {

protected:

double radius;

public:

container(double radiusl);

virtual double surface\_area() = 0;

virtual double volume() = 0;

};

要求建立3个继承container的派生类cube、sphere与cylinder，让每一个派生类都包含虚函数surface\_area()和volume()，分别用来计算正方体、球体和圆柱体的表面积及体积。要求写出主程序，应用C++的多态性，分别计算边长为6.0的正方体、半径为5.0的球体，以及半径为5.0和高为6.0的圆柱体的表面积和体积。

参考代码如下：

#include<iostream>

const double PI = 3.14159;

using namespace std;

class container {

protected:

double radius;

public:

container(double radius1);//抽象类container的构造函数

virtual double surface\_area() = 0;//纯虚函数suface\_area()

virtual double volume() = 0;//纯虚函数volume()

};

container::container(double radiusl)//定义抽象类container的构造函数

{

radius = radiusl;

}

class cube :public container //声明一个正方体派生类cube

{

public:

cube(double radius1):container(radius1)

{}

double surface\_area()//定义纯虚函数surface\_area

{

return 6 \* radius\*radius;

}

double volume()//定义纯虚函数volume

{

return radius\*radius\*radius;

}

};

class sphere :public container //声明一个球体派生类sphere

{public:

sphere(double radius1) :container(radius1) {}

double surface\_area()

{

return 4 \* PI\*radius\*radius;

}

double volume()

{

return PI\*radius\*radius\*radius \* 4 / 3;

}

};

class cylinder :public container {

double height;

public:

cylinder(double radius1, double height1) :container(radius1)

{

height = height1;

}

double surface\_area()

{

return 2 \* PI\*radius\*(radius + height);

}

double volume()

{

return PI\*radius\*radius\*height;

}

};

int main()

{

container \*ptr;

cube obj1(2.0);

sphere obj2(2.0);

cylinder obj3(2.0, 3.0);

ptr = &obj1;

cout << "这个正方体的表面积是：" << ptr->surface\_area() << endl;

cout << "这个正方体的体积是：" << ptr->volume() << endl;

ptr = &obj2;

cout << "这个球体的表面积是：" << ptr->surface\_area() << endl;

cout << "这个球体的体积是：" << ptr->volume() << endl;

ptr = &obj3;

cout << "这个圆柱体的表面积是：" << ptr->surface\_area() << endl;

cout << "这个圆柱体的体积是：" << ptr->volume() << endl;

return 0;

}

3、定义一个复数类Complex, 二个数据成员为double型r, i 为private属性。定义带二个参数的构造函数和一个Show( ) 函数用以输出r, i的值, 另外作为成员函数重载的运算符”+”的功能是将此类二个对象的数据成员r和i对应相加。这些成员函数的属性均为public。 请用C++编写此程序, 并编写测试程序进行测试。

参考代码如下：

#include <iostream >

using namespace std;

class Complex

{

private:

double r, i;

public:

Complex(double a, double b) : r(a), i(b) {}

void Show() { cout << r << " " << i << endl; }

Complex operator +(Complex obj)

{

return Complex(r + obj.r, i + obj.i);

}

};

int main()

{

Complex c1(3.5, 4.5), c2(2.5, 5.5), c3(0.0, 0.0);

c3 = c1 + c2;

c3.Show();

return 0;

}

4、声明一个Student类，在该类中包括一个数据成员score(分数)、两个静态数据成员total\_score(总分)和count(学生人数):还包括一个成员函数account()用于设置分数、累计学生的成绩之和、累计学生人数，一个静态成员函数sum()用于返回学生的成绩之和，另一个静态成员函数average()用于求全班成绩的平均值。在main函数中，输入某班同学的成绩，并调用上述函数求出全班学生的成绩之和和平均分。

参考代码如下：

#include<iostream>

using namespace std;

class Student{

public:

void account(double s);

static double sum();

static double average();

private:

double score;

static double total\_score;

static int count;

};

int Student::count=0;

double Student::total\_score=0;

void Student::account(double s)

{

score=s;

total\_score+=score;

count++;

}

double Student::sum()

{

return total\_score;

}

double Student::average()

{

return total\_score/count;

}

int main()

{

Student stu[10];

int n;

double s;

cin>>n;

for(int i=0;i<n;i++)

{cout<<"请输入第"<<i+1<<"号学生的成绩：";

cin>>s;

stu[i].account(s);

}

cout<<Student::sum()<<endl;

cout<<Student::average()<<endl;

return 0;

}

5、编写一个求任意类型数组中最大元素和最小元素的程序，要求将求最大元素和最小元素的函数设计成函数模板。

参考代码如下：

#include<iostream>

using namespace std;

template<typename T>

T MAX(T a[], int n)

{

T max=a[0];

for (int i = 1; i<n; i++)

if (a[i]>max)

max = a[i];

return max;

}

template<typename T>

T MIN(T a[], int n)

{

T min = a[0];

for (int i = 1; i<n; i++)

if (a[i]<min)

min = a[i];

return min;

}

int main()

{

int a[10] = { 3,5,6,8,9,1,2,7,0,10 };

double b[10] = { 2.1,3.5,6.8,7.9,3.2,5.9,9.0,8.8,9.8,9.2 };

char c[10] = { 'a','v','e','y','q','a','f','h','m','b' };

cout << "数组a中的最大元素是：" << MAX(a, 10) << endl;

cout << "数组a中的最小元素是：" << MIN(a, 10) << endl;

cout << "数组b中的最大元素是：" << MAX(b,10) << endl;

cout << "数组b中的最小元素是：" << MIN(b, 10) << endl;

cout << "数组c中的最大元素是：" << MAX(c, 10) << endl;

cout << "数组c中的最小元素是：" << MIN(c, 10) << endl;

return 0;

}

6、编写一个函数模板，用于求数组中各元素之和，并编写测试程序进行测试。

函数模板声明如下：

template <class Type>

Type Sum(Type tArray[], int iSize)

参考代码如下：

#include <iostream>

using namespace std;

template <class Type>

Type Sum(Type tArray[], int iSize)

{

Type tSum = 0;

for (int i = 0; i < iSize; i++)

{

tSum = tSum + tArray[i];

}

return tSum;

}

int main( )

{

int a[] = {2, 3,6};

double b[] = {1.8, 2.9, 9.9, 8.6};

cout << Sum(a, 3) << endl;

cout << Sum(b, 4) << endl;

return 0;

}

**7、**定义一个抽象类CShape，它有一个纯虚函数GetLength()；派生出四边型类CSquare和圆类CCircle，在派生类中重载函数GetLength()，用于求图形的周长，编写测试程序进行测试。

参考代码如下：

**#include <iostream>**

**using namespace std;**

**class CShape**

**{**

**public:**

**virtual double GetLength() const = 0;**

**};**

**class CSquare:public CShape**

**{**

**public:**

**CSquare(double dWidth, double dHeight)**

**{**

**m\_dWidth = dWidth;**

**m\_dHeight = dHeight;**

**}**

**double GetLength() const**

**{**

**return 2 \* (m\_dWidth + m\_dHeight);**

**}**

**private:**

**double m\_dWidth, m\_dHeight;**

**};**

**class CCircle:public CShape**

**{**

**public:**

**CCircle(double dRadius)**

**{**

**m\_dRadius = dRadius;**

**}**

**double GetLength() const**

**{**

**return 3.1415926 \* m\_dRadius \* m\_dRadius;**

**}**

**private:**

**double m\_dRadius;**

**};**

**int main(void)**

**{**

**CSquare oSquare(2, 3);**

**cout << oSquare.GetLength() << endl;**

**CCircle oCircle(10);**

**cout << oCircle.GetLength() << endl;**

**return 0;**

**}**