1. 编写C++程序一般需经过的几个步骤依次是（）

A. 编辑、调试、编译、连接B. 编辑、编译、连接、运行

C. 编译、调试、编辑、连接D. 编译、编辑、连接、运行

2. 决定C++语言中函数的返回值类型的是（）

A. return语句中的表达式类型

B. 调用该函数时系统随机产生的类型

C. 调用该函数时的主调用函数类型

D. 在定义该函数时所指定的数据类型

3. 下面叙述不正确的是（）

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

4. 所谓数据封装就是将一组数据和与这组数据有关操作组装在一起，形成一个实体，这实体

也就是（）

A. 类

B. 对象

C. 函数体

D. 数据块

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

的（）

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

6. 对基类和派生类的关系描述中，错误的是（）

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

7. 关于this指针使用说法正确的是（）

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。

8. 所谓多态性是指 （）

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

9. 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. 内联函数

B. 重载函数

C. 递归函数

D. 嵌套函数

10. 下面函数模板定义中不正确的是（）

A. A

B. B

C. C

D. D

11. 假设ClassY:publicX，即类Y是类X的派生类，则说明一个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次序分别为（）

A. X,Y；Y,X

B. X,Y；X,Y

C. Y,X；X,Y

D. Y,X；Y,X

12. 适宜采用inline定义函数情况是（）

A. 函数体含有循环语句

B. 函数体含有递归语句

C. 函数代码少、频繁调用

D. 函数代码多、不常调用

13. 假定一个类的构造函数为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（）

A. 3和15

B. 5和4

C. 4和20

D. 20和5

14. 在类中说明的成员可以使用关键字的是（）

A. public

B. extern

C. cpu

D. register

15. 下列不能作为类的成员的是（）

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

16. 使用地址作为实参传给形参，下列说法正确的是（）

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

17. 下列程序的输出结果是（）

#include <iostream.h>

void main()

{int n［］［3］={10,20,30,40,50,60};

int (\*p)［3］;

p=n;

cout<<p［0］［0］<<"，"<<\*(p［0］+1)<<"，"<<(\*p)［2］<<endl;}

A. 10，30，50

B. 10，20，30

C. 20，40，60

D. 10，30，60

18. 在C++中，使用流进行输入输出，其中用于屏幕输入（）

A. cin

B. cerr

C. cout

D. clog

19. 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（）

A. x.a

B. x.a()

C. x->a

D. （\*x）.a()

20. 关于对象概念的描述中，说法错误的是（）

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

1. C++的流库预定义了4个流，它们是cin、cout、clog和\_\_\_。

2. 每个对象都是所属类的一个\_\_\_。

3. 在已经定义了整型指针ip后，为了得到一个包括10个整数的数组并由ip所指向，应使用语

句\_\_\_。

4. 函数模板中紧随template之后尖括号内的类型参数都要冠以保留字\_\_\_。

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

6. 表达式cout<<end1 还可表示为\_\_\_。

7. 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\_\_\_。

8. 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\_\_\_构造函数来完成此功能。

9. 对赋值运算符进行重载时，应声明为\_\_\_函数。

10. 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\_\_\_。

11. 下列程序段的输出结果是\_\_\_。

for(i=0,j=10,k=0;i<=j;i++,j-=3,k=i+j);cout<<k;

12. String 类的\_\_\_方法返回查找到的字符串在主串的位置。

13. int n=0;

while（n=1）n++;

while循环执行次数是\_\_\_。

14. 控制格式输入输出的操作中，函数\_\_\_是用来设置填充字符。要求给出函数名和参数类型

15. C++语言支持的两种多态性分别是编译时的多态性和\_\_\_的多态性。

16. 设函数sum是由函数模板实现的，并且sum(3,6)和sum(4.6,8)都是正确的函数调用，则函

数模板具有\_\_\_个类型参数。

17. 执行下列代码

string str("HelloC++");

cout<<str.substr(5，3);

程序的输出结果是\_\_\_。

18. 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成\_\_\_。

19. 定义类动态对象数组时，元素只能靠自动调用该类的\_\_\_来进行初始化。

20. 已知有20个元素int类型向量V1，若用V1初始化为V2向量，语句是\_\_\_。

1. #include <iostream.h>

class Test

{private:

int x,y=20;

public:

Test(int i,int j){x=i,y=j;}

int getx(){return x;}

int gety(){return y;}

};

void main()

{Test mt(10,20);

cout<<mt.getx()<<endl;

cout<<mt.gety()<<endl;

}

2. #include <iostream.h>

class Test

{int x,y;

public:

fun(int i,int j)

{x=i;y=j;}

show()

{cout<<"x="<<x;

if(y)

cout<<",y="<<y<<endl;

cout<<endl;}

};

void main()

{Test a;

a.fun(1);

a.show();

a.fun(2,4);

a.show();

}

3. #include <iostream.h>

class A

{int i;

public:

virtual void fun()=0;

A(int a)

{i=a;}

};

class B:public A

{int j;

public:

void fun()

{cout<<"B::fun()＼n"; }

B(int m,int n=0):A(m),j(n){}

};

void main()

{A \*pa;

B b(7);

pa=&b;

}

4. #include <iostream.h>

class X

{public:

int x;

public:

X(int x)

{cout<<this->x=x<<endl;}

X(X&t)

{x=t.x;

cout<<t.x<<endl;

}

void fun(X);

};

void fun(X t)

{cout<<t.x<<endl;}

void main()

{fun(X(10));}

5. #include <iostream.h>

#include <string.h>

class Bas

{public:

Bas(char \*s="＼0"){strcpy(name,s);}

void show();

protected:

char name［20］;

};

Bas b;

void show()

{cout<<"name:"<<b.name<<endl;}

void main()

{Bas d2("hello");

show();

}

1. 在下面程序横线处填上适当字句，以使该程序执行结果为：

50 4 34 21 10

0 7.1 8.1 9.1 10.1 11.1

#include <iostream.h>

template <class T>

void f (\_\_\_\_\_\_\_\_\_\_)

{\_\_\_\_\_\_\_\_\_\_;

for (int i=0;i<n/2;i++)

t=a［i］, a［i］=a［n-1-i］, a［n-1-i］=t;

}

void main ()

{int a［5］={10,21,34,4,50};

double d［6］={11.1,10.1,9.1,8.1,7.1};

f(a,5);f(d,6);

for (int i=0;i<5;i++)

cout <<a［i］<< "";

cout <<endl;

for (i=0;i<6;i++)

cout << d［i］ << "";

cout << endl;

}

2. 在下面程序的底画线处填上适当的字句，使该程序执行结果为40。

#include <iostream.h>

class Test

{ public:

\_\_\_\_\_\_;

Test (int i=0)

{x=i+x;}

int Getnum()

{return Test::x+7;}

};

\_\_\_\_\_\_\_;

void main()

{Test test;

cout<<test.Getnum()<<endl;;

}

3. 在下列程序的空格处填上适当的字句，使输出为：0，2，10。

#include <iostream.h>

#include <math.h>

class Magic

{double x;

public:

Magic(double d=0.00):x(fabs(d))

{}

Magic operator+(\_\_\_\_\_\_)

{

return Magic(sqrt(x\*x+c.x\*c.x));

}

\_\_\_\_\_\_\_operator<<(ostream & stream,Magic & c)

{ stream<<c.x;

return stream;

}

};

void main()

{Magic ma;

cout<<ma<<", "<<Magic(2)<<", "<<ma+Magic(-6)+

Magic(-8)<<endl;

}

4. 下面是一个输入半径，输出其面积和周长的C++程序，在下划线处填上正确的语句。

#include <iostream>

\_\_\_\_\_\_\_\_\_;

\_\_\_\_\_\_\_\_\_;

void main()

{double rad;

cout<<"rad=";

cin>>rad;

double l=2.0\*pi\*rad;

double s=pi\*rad\*rad;

cout<<"＼n The long is："<<l<<endl;

cout<<"The area is："<<s<<endl;}

5. 程序实现大写字母转换成小写字母。

#include <iostream.h>

void main()

{char a;

\_\_\_\_\_\_\_;

cin>>a;

if(\_\_\_\_\_\_\_)

a=a+i;

cout<<a<<endl;

}

1. 给出下面程序输出结果。

#include<iostream.h>

class a

{public:

virtual void print()

{cout<< "a prog..."<< endl;};

};

class b:public a

{};

class c:public b

{public:

void print(){cout<<"c prog..."<<endl;}

};

void show(a \*p)

{(\*p).print();

}

void main()

{a a;

b b;

c c;

show(&a);

show(&b);

show(&c);

}

2. 给出下面程序输出结果。

#include <math.h>

#include <iostream.h>

#include <iomanip.h>

bool fun(long n);

void main()

{long a=10,b=30,l=0;

if(a%2==0) a++;

for(long m=a;m<=b;m+=2)

if(fun(m))

{if(l++%10==0)

cout <<endl;

cout <<setw(5) <<m;

}

}

bool fun(long n)

{int sqrtm=(int)sqrt(n);

for(int i=2;i<=sqrtm;i++)

if(n%i==0)

return false;

return true;

}

3. 给出下面程序输出结果。

#include <iostream.h>

class Test

{int x,y;

public:

Test(int i,int j=0)

{x=i;y=j;}

int get(int i,int j)

{return i+j;}

};

void main()

{Test t1(2),t2(4,6);

int (Test::\*p)(int,int=10);

p=Test::get;

cout<<(t1.\*p)(5)<<endl;

Test \*p1=&t2;

cout<<(p1->\*p)(7,20)<<endl;

}

4. #include <iostream.h>

#include <string.h>

#include <iomanip.h>

class student

{char name［8］;

int deg;

char level［7］;

friend class process; // 说明友元类

public:

student(char na［］,int d)

{ strcpy(name,na);

deg=d;

}

};

class process

{ public:

void trans(student &s)

{int i=s.deg/10;

switch(i)

{case 9:

strcpy(s.level, "优");break;

case 8:

strcpy(s.level,"良");break;

case 7:

strcpy(s.level,"中");break;

case 6:

strcpy(s.level,"及格");break;

default:

strcpy(s.level,"不及格");

}

}

void show(student &s)

{cout<<setw(10)<<s.name<<setw(4)<<s.deg<<setw(8)<<s.level<<endl;}

};

void main()

{ student st［］={student("张三",78),student("李四",92),student("王五

",62),student("孙六",88)};

process p;

cout<<"结 果:"<<"姓名"<<setw(6)<<"成绩"<<setw(8)<<"等级"<<endl;

for(int i=0;i<4;i++)

{ p.trans(st［i］);

p.show(st［i］);}

}

1. 已定义一个Shape抽象类，在此基础上派生出矩形Rectangle和圆形Circle类，二者都有

GetPerim（）函数计算对象的周长，并编写测试main（）函数。

class Shape

{public:

Shape(){}

~Shape(){}

virtual float GetPerim()=0;

}

1. 静态成员函数没有（）

A. 返回值

B. this指针

C. 指针参数

D. 返回类型

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

为（）

A. 3

B. 4

C. 5

D. 9

3. 有关多态性说法不正确的是（）

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. 编译时的多态性可通过函数重载实现

C. 运行时的多态性可通过模板和虚函数实现

D. 实现运行时多态性的机制称为动态多态性

4. 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. 1和0

B. 1和4

C. 4和0

D. 4和1

5. 类MyA的拷贝初始化构造函数是 （）

A. MyA()

B. MyA(MyA\*)

C. MyA(MyA&)

D. MyA(MyA)

6. 在C++中，函数原型不能标识（）

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

7. 友元关系不能（）

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

8. 实现两个相同类型数加法的函数模板的声明是（）

A. add(T x,T y)

B. T add(x,y)

C. T add(T x,y)

D. T add(T x,T y)

9. 在int a=3,int \*p=&a；中，\*p的值是（）

A. 变量a的地址值

B. 无意义

C. 变量p的地址值

D. 3

10. 下列不是描述类的成员函数的是（）

A. 构造函数

B. 析构函数

C. 友元函数

D. 拷贝构造函数

11. 如果从原有类定义新类可以实现的是（）

A. 信息隐藏

B. 数据封装

C. 继承机制

D. 数据抽象

12. 下面有关类说法不正确的是（）

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

13. 在函数定义中的形参属于（）

A. 全局变量

B. 局部变量

C. 静态变量

D. 寄存器变量

14. 下列有关重载函数的说法中正确的是（）

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

15. this指针存在的目的是（）

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

16. 关于new运算符的下列描述中，错误的是（）

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

17. 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（）

A. A1.p=5;

B. A1->p=5;

C. A1.\*p=5;

D. \*A1.p=5;

18. 以下基类中的成员函数表示纯虚函数的是（）

A. virtual void tt()=0

B. void tt(int)=0

C. virtual void tt(int)

D. virtual void tt(int){}

19. C++类体系中，不能被派生类继承的有（）

A. 常成员函数

B. 构造函数

C. 虚函数

D. 静态成员函数

20. 静态成员函数不能说明为（）

A. 整型函数

B. 浮点函数

C. 虚函数

D. 字符型函数

1. 假设int a=1,b=2;则表达式(++a/b)\*b--的值为\_\_\_。

2. 抽象类中至少要有一个\_\_\_函数。

3. 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\_\_\_。

4. 下面程序的输出结果为\_\_\_。

#include <iostream.h>

void main()

{int num=2,i=6;

do

{i--;

num++;

}while(--i);

cout<<num<<endl;

}

5. 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\_\_\_。

6. 在用C＋＋进行程序设计时，最好用\_\_\_代替malloc。

7. 由const修饰的对象称为\_\_\_。

8. 在C++程序设计中，建立继承关系倒挂的树应使用\_\_\_继承。

9. 基类的公有成员在派生类中的访问权限由\_\_\_决定。

10. 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\_\_\_。

11. this指针始终指向调用成员函数的\_\_\_。

12. 预处理命令以\_\_\_符号开头。

13. 类模板用来表达具有\_\_\_的模板类对象集。

14. C++程序的源文件扩展名为\_\_\_。

15. 在#include命令中所包含的头文件，可以是系统定义的头文件，也可以是\_\_\_的头文件。

16. vector类中向向量尾部插入一个对象的方法是\_\_\_。

17. 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\_\_\_方式。

18. 语句序列

ifstream infile;

infile.open(“data.dat”);

的功能可用一个语句实现，这个语句是\_\_\_。

19. 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\_\_\_。

20. 在编译指令中，宏定义使用\_\_\_指令。

1. #include <iostream>

#include <fstream>

#include <string>

using namespace std;

class A

{public:

A(const char \*na){strcpy(name,na);}

private:

char name［80］;

};

class B:public A

{ public:

B(const char \*nm):A(nm){}

void show();

};

void B::show()

{ cout<<"name:"<<name<<endl;

}

void main()

{ B b1("B");

b1.show();

}

2. #include <iostream.h>

void f(int \*a,int n)

{int i=0,j=0;

int k=0;

for(;i<n/2;i++)

{k=a［i］;

a［i］=a［n-i-1］;

a［n-i-1］=k;

}

}

void show(int a［］,int n)

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

cout<<a［i］<<" ";

cout<<endl;

}

void main()

{int p［5］;

int i=0,n=5;

for(;i<5;i++)

{p［i］=i;}

f(\*p,n);

show(p,n);

3. #include <iostream.h>

void main()

{int i(3),j(8);

int \* const p=&i;

cout<<\*p<<endl;

p=&j;

cout<<\*p<<endl;

}

4. #include <iostream.h>

　　void main()

　　{int i,\*p;

　　i=10;

　　\*p=i;

　　cout<<\*p<<endl;

　　}

5. #include <iostream.h>

　class A

{private:

int x,y;

public:

void fun(int i,int j)

{x=i;y=j;}

void show()

{cout<<x<<" "<<y<<endl;}

};

void main()

{A a1;

a1.fun(2);

a1.show();

}

1. 完成下面类中成员函数的定义。

#include <iostream>

#include <string>

using namespace std;

class str

{private:

char \*st;

public:

str(char \*a)

{set(a);

}

str & operator=(\_\_\_\_)

{delete st;

set(a.st);

return \*this;

}

void show(){cout<<st<<endl;}

~str(){delete st;}

void set(char \*s)//初始化st

{\_\_\_\_\_

strcpy(st,s);

}

};

void main()

{str s1("he"),s2("she");

s1.show(),s2.show();

s2=s1;

s1.show(),s2.show();}

2. 一个类的头文件如下所示，num初始化值为5，程序产生对象T，且修改num为10，并使用

show()函数输出num的值10。

#include <iostream.h>

class Test

{private:

static int num;

public:

Test(int);

void show();

};

\_\_\_\_\_\_\_\_\_

Test::Test(int n)

{num=n;}

void Test::show()

{cout<<num<<endl;}

void main()

{Test t(10);

\_\_\_\_\_\_\_\_\_

}

3. 下面是一个三角形三边，输出其面积C++程序，在下划线处填上正确的语句。

#include <iostream.h>

#include <math.h>

void area()

{double a,b,c;

cout<<"Input a b c:";

\_\_\_\_\_\_\_\_

if(a+b>c&&a+c>b&&c+b>a)

{double l=(a+b+c)/2;

\_\_\_\_\_\_\_

cout<<"The area is:"<<s<<endl;

}

else

cout<<"Error"<<endl;

}

void main()

{area();}

4. 下面程序中Base是抽象类。请在下面程序的横线处填上适当内容，以使程序完整,并使程序

的输出为:

Der1 called!

Der2 called!

#include <iostream.h>

class Base

{public:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

};

class Der1:public Base

{public:

void display(){cout<<"Der1 called!"<<endl;}

};

class Der2:public Base

{public:

void display(){cout<<"Der2 called!"<<endl;}

};

void fun(\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

{p->display();}

void main()

{Der1 b1;

Der2 b2;

Base \* p=&b1;

fun(p);

p=&b2;

fun(p);

}

5. 下面程序中用来求数组和。请在下面程序的横线处填上适当内容，以使程序完整,并使程序

的输出为:s＝150。

#include <iostream.h>

class Arr

{int \*a,n;

public:

Arr():a(0),n(0){}

Arr(int \*aa, int nn)

{n=nn;

a=new int［n］;

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

\*(a+i)=\*(aa+i);

}

~Arr(){delete a;}

\_\_\_\_\_\_\_\_\_\_\_\_\_；

{return \*(a+i);}

};

void main()

{int b［5］={10,20,30,40,50};

Arr a1(b,5);

int i=0,s=0;

\_\_\_\_\_\_\_\_\_\_\_\_\_

s+=a1.GetValue(i);

cout<<"s="<<s<<endl;

}

1. 给出下面程序输出结果。

#include <iostream.h>

class example

{int a;

public:

example(int b=5){a=b++;}

void print(){a=a+1;cout <<a<<"";}

void print()const

{cout<<a<<endl;}

};

void main()

{example x;

const example y(2);

x.print();

y.print();

}

2. 给出下面程序输出结果。

#include <iostream.h>

void main()

{ int \*p1;

int \*\*p2=&p1;

int b=20;

p1=&b;

cout<<\*\*p2<<endl;

}

3. 给出下面程序输出结果。

#include <iostream.h>

class Base

{private:

int Y;

public:

Base(int y=0) {Y=y;cout<<"Base("<<y<<")＼n";}

~Base() {cout<<"~Base()＼n";}

void print() {cout <<Y<< "";}

};

class Derived:public Base

{private:

int Z;

public:

Derived (int y, int z):Base(y)

{Z=z;

cout<<"Derived("<<y<<","<<z<<")＼n";

}

~Derived() {cout<<"～Derived()＼n";}

void print()

{Base::print();

cout<<Z<<endl;

}

};

void main()

{Derived d(10,20);

d.print();

}

10 20

～Derived()

~Base()

［解析］派生类对象，先调用基类构造函数输出Base(10)，后调用派生类构造函数输出

Derived(10,20)，后执行d.print(),调用派生类的print，再调用Base::print()输出10，后返回

输出z的值20。后派生类析构，再基类析构。

4. 给出下面程序输出结果。

#include <iostream.h>

class A

{public:

A()

{cout<<"A 构造函数＼n";fun();}

virtual void fun()

{cout<<"A::fun() 函数＼n";}

};

class B:public A

{public:

B()

{cout<<"B构造函数＼n";fun();}

void fun() {cout<<"B::fun() calle函数＼n";}

};

void main()

{B d;}

1. 编写类String的构造函数、析构函数和赋值函数和测试程序。

　　已知类String的原型为：

#include <iostream.h>

#include <string.h>

class String

{public:

String(const char \*str=NULL); // 普通构造函数

String(const String &other); // 拷贝构造函数

~String(); // 析构函数

String & operator=(const String &other); // 赋值函数

void show()

{cout<<m\_data<<endl;

}

private:

char \*m\_data; // 用于保存字符串

};

1. 设有定义int i;double j＝5;，则10+i+j值的数据类型是（）

A. int

B. double

C. float

D. 不确定

2. 要禁止修改指针p本身，又要禁止修改p所指向的数据，这样的指针应定义为（）

A. const char \*p=“ABCD”；

B. char \*const p=“ABCD”；

C. char const \*p=“ABCD”；

D. const char \* const p=“ABCD”；

3. 类的构造函数被自动调用执行的情况是在定义该类的（）

A. 成员函数时

B. 数据成员时

C. 对象时

D. 友元函数时

4. 已知类A是类B的友元，类B是类C的友元，则（）

A. 类A一定是类C的友元

B. 类C一定是类A的友元

C. 类C的成员函数可以访问类B的对象的任何成员

D. 类A的成员函数可以访问类B的对象的任何成员

5. 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. 1和0

B. 1和4

C. 4和0

D. 4和1

6. 关于this指针使用说法正确的是（）

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。

7. 所谓多态性是指 （）

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

8. 友元关系不能（）

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

9. 语句ofstream f(“TEMP.DAT”,ios::app｜ios::binary)?的功能是建立流对象f，试图打

开文件TEMP.DAT 并与之连接，并且（）

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

10. 构造函数不具备的特征是（）

A. 构造函数的函数名与类名相同

B. 构造函数可以重载

C. 构造函数可以设置默认参数

D. 构造函数必须指定类型说明

11. 在公有继承的情况下，基类的公有或保护成员在派生类中的访问权限（）

A. 受限制

B. 保持不变

C. 受保护

D. 不受保护

12. 假定一个类的构造函数为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（）

A. 3和15

B. 5和4

C. 4和20

D. 20和5

13. C++对C语言做了很多改进，即从面向过程变成为面向对象的主要原因是（）

A. 增加了一些新的运算符

B. 允许函数重载，并允许设置缺省参数

C. 规定函数说明符必须用原型

D. 引进了类和对象的概念

14. 在类中说明的成员可以使用关键字的是（）

A. public

B. extern

C. cpu

D. register

15. C++语言中所有在函数中定义的变量，连同形式参数，都属于（）

A. 全局变量

B. 局部变量

C. 静态变量

D. 函数

16. 在私有继承的情况下，基类成员在派生类中的访问权限（）

A. 受限制

B. 保持不变

C. 受保护

D. 不受保护

17. 使用地址作为实参传给形参，下列说法正确的是（）

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

18. C++的继承性允许派生类继承基类的（）

A. 部分特性，并允许增加新的特性或重定义基类的特性

B. 部分特性，但不允许增加新的特性或重定义基类的特性

C. 所有特性，并允许增加新的特性或重定义基类的特性

D. 所有特性，但不允许增加新的特性或重定义基类的特性

19. 对于int \*pa［5］;的描述，正确的是（）

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是一个指向某个数组中第5个元素的指针，该元素是int型变量

C. pa［5］表示某个数组的第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

20. 以下基类中的成员函数表示纯虚函数的是（）

A. virtual void tt()=0

B. void tt(int)=0

C. virtual void tt(int)

D. virtual void tt(int){}

1. 单目运算符作为类成员函数重载时，形参个数为\_\_\_个。

2. 抽象类中至少要有一个\_\_\_函数。

3. 设类A有成员函数void f(void)；若要定义一个指向类成员函数的指针变量pf来指向f，该

指针变量的声明语句是：\_\_\_。

4. 执行下列程序

double a=3.1415926,b=3.14;

cout<<setprecision(5)<<a<<", "<<setprecision(5)<<b<<endl;

程序的输出结果是\_\_\_。

5. vector类中用于删除向量中的所有对象的方法是\_\_\_。

6. 重载的运算符保持其原有的\_\_\_、优先级和结合性不变。

7. 编译时的多态性通过\_\_\_函数实现。

8. 基类的公有成员在派生类中的访问权限由\_\_\_决定。

9. 假设类X的对象x是类Y的成员对象，则“Y Obj”语句执行时，先调用类\_\_\_的构造函数。

10. 下列程序段的输出结果是\_\_\_。

cout.setf(ios::showpos);

cout<<509.3<<endl;

11. 下列程序段的输出结果是\_\_\_。

for(i=0,j=10,k=0;i<=j;i++,j-=3,k=i+j);cout<<k;

12. C＋＋中ostream的直接基类\_\_\_。

13. int n=0;

while（n=1）n++;

while循环执行次数是\_\_\_。

14. C++中有两种继承：单继承和\_\_\_。

15. 在C++中，利用向量类模板定义一个具有10个int的向量A，其元素均被置为1，实现此操作

的语句是\_\_\_。

16. vector类中向向量尾部插入一个对象的方法是\_\_\_。

17. 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\_\_\_方式。

18. 若函数的定义处于调用它的函数之前，则在程序开始可以省去该函数的\_\_\_语句。

19. 在C++中有两种参数传递方式：传值和\_\_\_。

20. 将指向对象的引用作为函数的形参，形参是对象的引用，实参是 \_\_\_。

1. class ABC

{int a;

public:

ABC(int aa)a(aa){}

};

2. #include <iostream.h>

class Aton

{int X,Y;

protected:

int zx,zy;

public:

void init(int i,int j){zx=i;zy=j;}

Aton(int i,int j,int n=0,int m=0)

{X=i,Y=j,zx=m,zy=n;

}

};

void main()

{Aton A(25,20,3,5);

A.init(5,9);

cout<<A.X()<<endl;

3. #include <iostream.h>

class Bas

{public:

~Bas(){cout<<"Bas construct"<<endl;}

virtual void f()=0;

};

class Dev:public Bas

{public:

~Dev(){cout<<"Bas construct"<<endl;}

virtual void f(){cout<<"Dev::f"<<endl;}

};

void main()

{Bas \*a=new Bas();

Dev p;

a=&p;

a->f();

}

4. 以下程序实现交换a,b变量的值，请用下横线标出错误所在行并给出修改意见。

#include <iostream.h>

void swap(int &a,int &b)

{a=a+b;

b=a-b;

a=a-b;

}

void main()

{int a=19,b=15;

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

swap(&a,&b);

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

}

5. #include <iostream.h>

void main()

{int i(3),j(8);

int \* const p=&i;

cout<<\*p<<endl;

p=&j;

cout<<\*p<<endl;

}

1. 在下面程序横线处填上适当内容，使程序执行结果为："hello, andylin"。

#include <iostream>

#include <string.h>

using namespace std;

class mystring

{public:

char \* pdata;

mystring(int len)

{pdata=new char［len+1］;

}

~mystring()

{delete pdata;}

void show(){cout<<pdata<<endl;}

};

void fun(mystring\*\* array,int len)

{mystring\*old=\*array;

\_\_\_\_\_\_\_;

memcpy(\*array, old, len);

}

void main()

{mystring str(20);

mystring\*pstr=&str;

mystring\*\*ppstr=&pstr;

strcpy(str.pdata,"hello,andylin");

fun(ppstr, 20);

\_\_\_\_\_\_\_

}

2. 在下面程序横线处填上适当字句，完成类的定义。

class line;

class box

{ private:

int color;

int upx, upy;

int lowx, lowy;

public:

\_\_\_\_\_\_\_\_\_

void set\_color (int c){color=c;}

void define\_box (int x1, int y1, int x2, int y2)

{upx=x1;upy=y1;lowx=x2;lowy=y2;}

};

class line

{ private:

int color;

int startx, starty;

int endx, endy;

public:

friend int same\_color(line l,box b);

void set\_color (int c) {color=c;}

void define\_line (\_\_\_\_\_\_\_\_\_\_\_)

{startx=x1;starty=y1;endx=x2;endy=y2;}

};

int same\_color(line l, box b)

{if (l.color==b.color) return 1;

return 0;

}

3. 下面程序用来求直角三角形斜边长度。

#include <iostream.h>

#include <math.h>

class Point

{private:

double x,y;

\_\_\_\_\_\_\_\_\_\_

public:

Point(double i=0,double j=0)

{x=i;y=j;}

Point(Point &p)

{x=p.x;y=p.y;}

};

class Line

{private:

Point p1,p2;

public:

Line(Point &xp1,Point &xp2):\_\_\_\_\_\_\_\_{}

double GetLength();

};

double Line::GetLength()

{double dx=p2.x-p1.x;

double dy=p2.y-p1.y;

return sqrt(dx\*dx+dy\*dy);

}

void main()

{ Point p1,p2(6,8);

Line L1(p1,p2);

cout<<L1.GetLength()<<endl;

}

4. 在下面程序的底画线处填上适当的字句，使该程序执行结果为40。

#include <iostream.h>

class Test

{ public:

\_\_\_\_\_\_;

Test (int i=0)

{x=i+x;}

int Getnum()

{return Test::x+7;}

};

\_\_\_\_\_\_\_;

void main()

{Test test;

cout<<test.Getnum()<<endl;;

}

5. 在下列程序的空格处填上适当的字句，使输出为：0，2，10。

#include <iostream.h>

#include <math.h>

class Magic

{double x;

public:

Magic(double d=0.00):x(fabs(d))

{}

Magic operator+(\_\_\_\_\_\_)

{

return Magic(sqrt(x\*x+c.x\*c.x));

}

\_\_\_\_\_\_\_operator<<(ostream & stream,Magic & c)

{ stream<<c.x;

return stream;

}

};

void main()

{Magic ma;

cout<<ma<<", "<<Magic(2)<<", "<<ma+Magic(-6)+

Magic(-8)<<endl;

}

1. 运行程序，写出程序执行的结果。

#include <iostream.h>

void main()

{int a,b,c;

char ch;

cin>>a>>ch>>b>>c;//从键盘上输入1.5×c×10×20,×表示一个空格

cout<<a<<endl<<ch<<endl<<b<<endl<<c<<endl;

}

5

0

［解析］使用cin进行输入字符的输入的问题。1-->a,.-->ch,5-->b,空格转换为零给了c。

2. 给出下面程序输出结果。

#include <iostream.h>

class A

{public:

A()

{cout<<"As cons."<<endl;}

virtual ~A()

{cout<<"As des."<<endl;}

virtual void f()

{cout<<"As f()."<<endl;}

void g()

{f();}

};

class B:public A

{public:

B()

{f();cout<<"Bs cons."<<endl;}

~B()

{cout<<"Bs des."<<endl;}

};

class C:public B

{public:

C()

{cout<<"Cs cons."<<endl;}

~C()

{cout<<"Cs des."<<endl;}

void f()

{cout<<"Cs f()."<<endl;}

};

void main()

{A \*a=new C;

a->g();

delete a;

}

1. 已知交通工具类定义如下。

要求：（1)实现这个类；（2）定义并实现一个小车类car，是它的公有派生类，小车本身的私有

属性有载人数，小车的函数有init(设置车轮数，重量和载人数），getpassenger(获取载人数

)，print（打印车轮数，重量和载人数）。

class vehicle

{protected:

int wheels;//车轮数

float weight;//重量

public:

void init(int wheels,float weight);

int get\_wheels();

float get\_weight();

void print();

};

void vehicle::init(int wheels,float weight)

{this->wheels=wheels;

this->weight=weight;

cout<<wheels<<endl;

}

int vehicle::get\_wheels()

{return wheels;

}

float vehicle::get\_weight()

{return weight;}

void vehicle::print()

{cout<<"车轮数："<<wheels<<","<<"重量："<<weight<<endl;}

1. 当一个类的某个函数被说明为virtual时，该函数在该类的所有派生类中（）

A. 都是虚函数

B. 只有被重新说明时才是虚函数

C. 只有被重新说明为virtual时才是虚函数

D. 都不是虚函数

2. 要禁止修改指针p本身，又要禁止修改p所指向的数据，这样的指针应定义为（）

A. const char \*p=“ABCD”；

B. char \*const p=“ABCD”；

C. char const \*p=“ABCD”；

D. const char \* const p=“ABCD”；

3. 函数调用func((exp1,exp2),(exp3,exp4,exp5))中所含实参的个数为（）

A. 1

B. 2

C. 4

D. 5

4. 设有函数模板

template <class Q>

Q Sum(Q x,Q y)

{return (x)+(y);}

则下列语句中对该函数模板错误的使用是（）

A. Sum(10,2);

B. Sum(5.0,6.7)；

C. Sum(15.2f,16.0f);

D. Sum(“AB”,”CD”);

5. 类B是类A的公有派生类，类A和类B中都定义了虚函数func(),p是一个指向类A对象的指针

，则p->A::func()将（）

A. 调用类A中的函数func()

B. 调用类B中的函数func()

C. 根据p所指的对象类型而确定调用类A中或类B中的函数func()

D. 既调用类A中函数，也调用类B中的函数

6. 在面向对象的程序设计中，首先在问题域中识别出若干个 （）

A. 函数

B. 类

C. 文件

D. 过程

7. 已知f1和f2是同一类的两个成员函数，但f1不能直接调用f2，这说明（）

A. f1和f2都是静态函数

B. f1不是静态函数，f2是静态函数

C. f1是静态函数，f2不是静态函数

D. f1和f2都不是静态函数

8. 下列有关模板和继承的叙述正确的是 （）

A. 模板和继承都可以派生出一个类系

B. 从类系的成员看，模板类系的成员比继承类系的成员较为稳定

C. 从动态性能看， 继承类系比模板类系具有更多的动态特性

D. 相同类模板的不同实例一般没有联系，而派生类各种类之间有兄弟父子等关系

9. 有关C++编译指令，以下叙述正确的是（）

A. C++每行可以写多条编译指令

B. #include指令中的文件名可含有路径信息

C. C++的编译指令可以以#或//开始

D. C++中不管＃if后的常量表达式是否为真，该部分都需要编译

10. 在C++中不返回任何类型的函数应该说明为（）

A. int

B. char

C. void

D. double

11. 若Sample类中的一个成员函数说明如下：

void set(Sample& a)，则Sample& a的含义是（）

A. 指向类Sample的名为a的指针

B. a是类Sample的对象引用，用来作函数Set（）的形参

C. 将a的地址赋给变量Set

D. 变量Sample与a按位与的结果作为函数Set的参数

12. 下列关于静态数据成员的描述中正确的是（）

A. 静态数据成员是类的所有对象所共有的

B. 静态数据成员要在构造函数内初始化

C. 类的每个对象有自己的静态数据成员

D. 静态数据成员不能通过类的对象调用

13. 在编译指令中，宏定义使用哪个指令（）

A. #if

B. #include

C. #define

D. #error

14. 类的析构函数是对一个对象进行以下哪种操作时自动调用的是（）

A. 建立

B. 撤销

C. 赋值

D. 引用

15. 关于new运算符的下列描述中，错误的是（）

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

16. 如果类A被说明成类B的友元，则（）

A. 类B不一定是类A的友元

B. 类B的成员即类A的成员

C. 类A的成员即类B的成员

D. 类A的成员函数不得访问类B的成员

17. 假定一个类的构造函数为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（）

A. 4和5

B. 4和20

C. 4和24

D. 20和5

18. 下列运算符中，在C++语言中不能重载的是（）

A. \*

B. >=

C. ：：

D. /

19. C++语言建立类族是通过（）

A. 类的嵌套

B. 类的继承

C. 虚函数

D. 抽象类

20. 在C++语言中，数据封装要解决的问题是（）

A. 数据的规范化

B. 便于数据转换

C. 避免数据丢失

D. 防止不同模块之间数据的非法访问

1. 若要使用string类，需要引入的头文件是\_\_\_。

2. 在函数前面用\_\_\_保留字修饰时，则表示该函数表为内联函数。

3. 向量操作方法中\_\_\_方法返回向量中的第一个对象。

4. C++派生类使用两种基本的面向对象技术：第一种称为性质约束，即对基类的性质加以限制

；第二种称为\_\_\_，即增加派生类的性质。

5. 重载的运算符保持其原有的\_\_\_、优先级和结合性不变。

6. 编译时的多态性通过\_\_\_函数实现。

7. 预处理语句有三种，分别是宏定义、文件包含和\_\_\_。

8. 构造函数、析构函数和友元函数中，不是该类成员的是\_\_\_。

9. 控制格式输入输出的操作中，函数\_\_\_是设置域宽的。要求给出函数名和参数类型）。

10. 派生类的成员一般分为两部分，一部分是\_\_\_，另一部分是自己定义的新成员。

11. C＋＋中ostream的直接基类\_\_\_。

12. vector的\_\_\_方法返回向量中的最后一个对象。

13. 执行下列代码

int i=230;

cout <<"i="<<hex <<i<<endl;

程序的输出结果为\_\_\_。

14. 在C++中有两种参数传递方式即值传递和\_\_\_传递。

15. 使用new为int数组动态分配10个存储空间是\_\_\_。

16. 面向对象的四个基本特性是多态性、继承性、和封装性\_\_\_。

17. 定义虚函数所用的关键字是\_\_\_。

18. 执行下列代码

cout<<“oct:”<<oct<<34;

程序的输出结果是\_\_\_。

19. 在C++中要创建一个文件输入流对象fin，同时该对象打开文件“Test.txt”用于输入，则

正确的声明语句是\_\_\_。

20. 如果一个派生类只有一个唯一的基类，则这样的继承关系称为\_\_\_。

1. class ABC

{int a;

public:

ABC(int aa)a(aa){}

};

2. #include <iostream.h>

class T

{protected:

int p;

public:

T(int m){p=m;}

};

void main()

{ T a(10);

cout<<a.p<<endl;

}

3. #include <iostream>

using namespace std;

class Date;

class Time

{public:

Time(int h,int m,int s)

{hour=h,minute=m,sec=s;}

void show(Date & d);

private:

int hour,minute,sec;

};

class Date

{public:

Date(int m,int d,int y)

{month=m,day=d,year=y;}

void Time::show(Date &);

private:

int month,day,year;

};

void Time::show(Date & d)

{cout<<d.month <<"-"<<d.day<<"-"<<d.year<<endl;

cout<<hour<<":"<<minute<<":"<<sec<<endl;

}

void main()

{Time t1(9,23,50);

Date d1(12,20,2008);

t1.show(d1);

4. 输出最小值，有一处错误。

#include <iostream.h>

class Test

{int a,b;

int getmin()

{return (a<b?a:b);}

public:

int c;

void setValue(int x1,int x2,int x3)

{a=x1;b=x2;c=x3;}

int GetMin();

};

int Test::GetMin()

{int d=getmin();

return (d=d<c?d:c);

}

void main()

{Test t1;

t1.setValue(34,6,2);

cout<<t1.getmin ()<<endl;

}

5. 实现数值、字符串的交换。

#include <iostream>

#include <string>

using namespace std;

template<class T>

void Swap(T& a,T& b)

{T temp;

　temp=a,a=b,b=temp;

}

void main()

{int a=5,b=9;

char s1［］="Hello",s2［］="hi";

Swap(a,b);

Swap(s1,s2);

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

cout<<"s1="<<s1<<",s2="<<s2<<endl;

}

1. 在下划线处填上缺少的部分。

#include <iostream.h>

class A

{int a,b;

public:

\_\_\_\_\_；//定义构造函数，使参数i和j的默认值为0

{a=i;b=j;}//在函数体中用i初始化a，用j初始化b

};

main()

{A \*p;

\_\_\_\_\_;//调用带参构造函数生成由p指向的动态对象

//使a和b成员分别被初始化为4和5

}

4，5。

2. 在下面程序横线处填上适当内容，使程序执行结果为：

S=2

S=5

S=9

#include <iostream.h>

void sum(int i)

{static int s;

\_\_\_\_\_\_\_\_\_;

cout<<"s="<<s<<endl;

}

void main (void)

{int i;

for (i=0;\_\_\_\_\_\_\_\_)

sum(i);

}

3. 下面程序运行的结果是：5+10=15。

#include <iostream.h>

class Test

{ private:

int x,y;

public:

Test() {x=y=0;}

void Setxy(int x,int y) {\_\_\_\_\_\_}

void show(){\_\_\_\_\_\_}

};

void main()

{Test ptr;

ptr.Setxy(5,10);

ptr.show();

}

4. 完成下面类中成员函数的定义。

#include <iostream.h>

#include <iomanip.h>

class Arr

{protected:

float \*p;

int n;//数组大小（元素个数）

public:

Arr(int sz=10)

{ n=sz;

p=new float［n］;

}

~Arr(void)

{

\_\_\_\_\_\_\_\_\_

}

int Getn(void) const

{

return n;

}

float & operator［］(int i)

{

\_\_\_\_\_\_\_\_

}

void Print();

};

void Arr::Print()

{int i;

for(i=0;i< this->Getn();i++)

{if (i%10==0)

cout << endl;

cout<<setw(6)<<p［i］;

}

cout<<endl;

}

void main()

{Arr a(20);

for (int i=0;i<a.Getn();i++)

a［i］=i\* 2;

a.Print();

}

5. 请在下面程序的横线处填上适当内容，以使程序完整,并使程序的输出为:

11,10

13,12

#include <iostream.h>

class A

{int a;

public:

A(int i=0){a=i;}

int Geta(){return a;}

void show(){cout<<a<<endl;}

};

class B

{A a;

int b;

public:

B(int i,int j)\_\_\_\_\_\_\_\_\_

{}

void show(){cout<<a.Geta()<<","<<b<<endl;}

};

void main()

{B b［2］={B(10,11),B(12,13)};

for(int i=0;i<2;i++)

\_\_\_\_\_\_\_\_\_\_

}

1. 给出下面程序输出结果。

#include<iostream.h>

class a

{public:

a(int i=10){x=i;cout<<"a:"<<x<<endl;}

int x;

};

class b:public a

{public:

b(int i):A(i){x=i;cout<<"b:"<<x<<", "<<a::x<<endl;}

private:

a A;

int x;

};

void main()

{b B(5);

}

2. 运行程序，写出程序执行的结果。

#include<iostream.h>

class Location

{public:

int X,Y;

void init(int initX,int initY);

int GetX();

int GetY();

};

void Location::init (int initX,int initY)

{X=initX;

Y=initY;

}

int Location::GetX()

{return X;

}

int Location::GetY()

{return Y;

}

void display(Location& rL)

{cout<<rL.GetX()<<" "<<rL.GetY()<<＼n;

}

void main()

{Location A［5］={{5,5},{3,3},{1,1},{2,2},{4,4}};

Location \*rA=A;

A［3］.init(7,3);

rA->init(7,8);

for (int i=0;i<5;i++)

display(\*(rA++));

}

3 3

1 1

7 3

4 4

［解析］对象数组的使用。使用数组对象修改了A［3］元素的值，又使用指针修改指针所指向的

第一个元素的值，因此修改了A［0］和A［3］元素的值。

3. 给出下面程序输出结果。

#include <iostream.h>

int a［8］={1,2,3,4,5,6,7};

void fun(int \*pa,int n);

void main()

{int m=8;

fun(a,m);

cout<<a［7］<<endl;

}

void fun(int \*pa,int n)

{for (int i=0;i<n-1;i++)

\*(pa+7)+=\*(pa+i);

}

4. 给出下面程序输出结果。

#include <iostream.h>

class A

{int \*a;

public:

A(int x=0):a(new int(x)){}

~A() {delete a;}

int getA() {return \*a;}

void setA(int x) {\*a=x;}

};

void main()

{A x1,x2(3);

A \*p=&x2;

(\*p).setA(x2.getA()+5);

x1.setA(10+x1.getA());

cout<<x1.getA()<<""<<x2.getA()<<endl;

}

1. 已知交通工具类定义如下。

要求：（1)实现这个类；（2）定义并实现一个小车类car，是它的公有派生类，小车本身的私有

属性有载人数，小车的函数有init(设置车轮数，重量和载人数），getpassenger(获取载人数

)，print（打印车轮数，重量和载人数）。

class vehicle

{protected:

int wheels;//车轮数

float weight;//重量

public:

void init(int wheels,float weight);

int get\_wheels();

float get\_weight();

void print();

};

void vehicle::init(int wheels,float weight)

{this->wheels=wheels;

this->weight=weight;

cout<<wheels<<endl;

}

int vehicle::get\_wheels()

{return wheels;

}

float vehicle::get\_weight()

{return weight;}

void vehicle::print()

{cout<<"车轮数："<<wheels<<","<<"重量："<<weight<<endl;}

1. 静态成员函数没有（）

A. 返回值

B. this指针

C. 指针参数

D. 返回类型

2. 在类的定义中，用于为对象分配内存空间，对类的数据成员进行初始化并执行其他内部管

理操作的函数是（）

A. 友元函数

B. 虚函数

C. 构造函数

D. 析构函数

3. 所有在函数中定义的变量，都是（）

A. 全局变量

B. 局部变量

C. 静态变量

D. 寄存器变量

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

为（）

A. 3

B. 4

C. 5

D. 9

5. 如果表达式++a中的“++”是作为成员函数重载的运算符，若采用运算符函数调用格式，则

可表示为（）

A. a.operator++(1)

B. operator++(a)

C. operator++(a,1)

D. a.operator++()

6. 已知f1和f2是同一类的两个成员函数，但f1不能直接调用f2，这说明（）

A. f1和f2都是静态函数

B. f1不是静态函数，f2是静态函数

C. f1是静态函数，f2不是静态函数

D. f1和f2都不是静态函数

7. 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. 内联函数

B. 重载函数

C. 递归函数

D. 嵌套函数

8. 解决定义二义性问题的方法有（）

A. 只能使用作用域分辨运算符

B. 使用作用域分辨运算符或成员名限定

C. 使用作用域分辨运算符或虚基类

D. 使用成员名限定或赋值兼容规则

9. 在main函数中可以用p.a的形式访问派生类对象p的基类成员a，其中a是（）

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

10. 在C++中不返回任何类型的函数应该说明为（）

A. int

B. char

C. void

D. double

11. 若Sample类中的一个成员函数说明如下：

void set(Sample& a)，则Sample& a的含义是（）

A. 指向类Sample的名为a的指针

B. a是类Sample的对象引用，用来作函数Set（）的形参

C. 将a的地址赋给变量Set

D. 变量Sample与a按位与的结果作为函数Set的参数

12. 要实现动态联编必须（）

A. 通过成员名限定来调用虚函数

B. 通过对象名来调用虚函数

C. 通过派生类对象来调用虚函数

D. 通过对象指针或引用来调用虚函数

13. 在派生类中定义虚函数时，可以与基类中相应的虚函数不同的是（）

A. 参数类型

B. 参数个数

C. 函数名称

D. 函数体

14. 实现两个相同类型数加法的函数模板的声明是（）

A. add(T x,T y)

B. T add(x,y)

C. T add(T x,y)

D. T add(T x,T y)

15. 下列不是描述类的成员函数的是（）

A. 构造函数

B. 析构函数

C. 友元函数

D. 拷贝构造函数

16. 继承机制的作用是（）

A. 信息隐藏

B. 数据封装

C. 定义新类

D. 数据抽象

17. 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（）

A. A1.p=5;

B. A1->p=5;

C. A1.\*p=5;

D. \*A1.p=5;

18. 如果采用动态多态性，要调用虚函数的是（）

A. 基类对象指针

B. 对象名

C. 基类对象

D. 派生类名

19. 若有以下定义，则说法错误的是（）

int a=100,\*p=&a;

A. 声明变量p，其中\*表示p是一个指针变量

B. 变量p经初始化，获得变量a的地址

C. 变量p只可以指向一个整型变量

D. 变量p的值为100

20. C++语言建立类族是通过（）

A. 类的嵌套

B. 类的继承

C. 虚函数

D. 抽象类

1. 假设int a=1,b=2;则表达式(++a/b)\*b--的值为\_\_\_。

2. 复制构造函数使用\_\_\_作为形式参数。

3. 通过C++语言中的\_\_\_机制，可以从现存类中构建其子类。

4. 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\_\_\_。

5. 在下面的类定义中，私有成员有\_\_\_。

class Location

{int X,Y;

protected:

int zeroX,zerxY;

int SetZero(intzeroX, intzeroY);

private:

int length,height;

public:

void init(int initX,int initY);

int GetX();

int GetY();

};

6. 在C++程序设计中，建立继承关系倒挂的树应使用\_\_\_继承。

7. C++支持的两种多态性分别是\_\_\_多态性和运行多态性。

8. 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

\_\_\_。

9. 在C++中，定义虚函数的关键字是\_\_\_。

10. 采用私有派生方式，基类的public成员在私有派生类中是\_\_\_成员。

11. 对赋值运算符进行重载时，应声明为\_\_\_函数。

12. 在C++中有两种参数传递方式即值传递和\_\_\_传递。

13. 预处理命令以\_\_\_符号开头。

14. 在构造函数和析构函数中调用虚函数时采用\_\_\_。

15. C++是通过引用运算符\_\_\_来定义一个引用的。

16. 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\_\_\_。

17. 如果要把PI声明为值为3.14159类型为双精度实数的符号常量，该声明语句是\_\_\_。

18. 在C＋＋四个流对象中，\_\_\_用于标准屏幕输出。

19. 执行下列代码

int a=32;

double c=32;

cout.setf(ios::hex);

cout<<"hex:a="<<a<<",c="<<c<<endl;

cout.unsetf(ios::hex);

程序的输出结果为\_\_\_。

20. 已知有20个元素int类型向量V1，若用V1初始化为V2向量，语句是\_\_\_。

1. #include <iostream.h>

class A

{ private:

int x;

public:

A(int i){x=i;}

A(){x=0;}

friend int min(A&,A&);

};

int min(A & a,A &b)

{ return (a.x>b.x)?a.x:b.x;

}

void main()

{ A a(3),b(5);

cout<<a.min(a,b)<<endl;

}

2. #include <iostream.h>

class shape

{public:

virtual int area(){return 0;}

};

class rectangle:public shape

{public:

int a, b;

void setLength (int x, int y) {a=x;b=y;}

int area() {return a\*b;}

};

void main()

{rectangle r;

r.setLength(3,5);

shape s1,\*s2=&r;

cout <<r.area() <<endl;

s2=s1;

cout <<s2.area()<<endl;

}

3. 下面的类定义中有一处错误，请用下横线标出错误所在行并给出修改意见。

#include <iostream.h>

template <class T>

class A

{private:

T x,y,s;

public:

A(T a,T b)

{x=a,y=b;s=x+y;}

void show()

{cout<<"x+y="<<s<<endl;

}

};

void main()

{ A <int>add(10,100);

add.show();

}

4. 生成具有n个元素的动态数组。

#include <iostream.h>

void main()

{int n;

cin>>n;

int a［n］;

a［0］=2;

cout<<a［0］<<endl;

}

5. #include <iostream.h>

class A

{int i;

public:

virtual void fun()=0;

A(int a)

{i=a;}

};

class B:public A

{int j;

public:

void fun()

{cout<<"B::fun()＼n"; }

B(int m,int n=0):A(m),j(n){}

};

void main()

{A \*pa;

B b(7);

pa=&b;

}

1. 在下面程序横线处填上适当字句，以使该程序执行结果为：

50 4 34 21 10

0 7.1 8.1 9.1 10.1 11.1

#include <iostream.h>

template <class T>

void f (\_\_\_\_\_\_\_\_\_\_)

{\_\_\_\_\_\_\_\_\_\_;

for (int i=0;i<n/2;i++)

t=a［i］, a［i］=a［n-1-i］, a［n-1-i］=t;

}

void main ()

{int a［5］={10,21,34,4,50};

double d［6］={11.1,10.1,9.1,8.1,7.1};

f(a,5);f(d,6);

for (int i=0;i<5;i++)

cout <<a［i］<< "";

cout <<endl;

for (i=0;i<6;i++)

cout << d［i］ << "";

cout << endl;

}

2. 完成下面类中成员函数的定义。

#include <iostream.h>

#include <iomanip.h>

class Arr

{protected:

float \*p;

int n;//数组大小（元素个数）

public:

Arr(int sz=10)

{ n=sz;

p=new float［n］;

}

~Arr(void)

{

\_\_\_\_\_\_\_\_\_

}

int Getn(void) const

{

return n;

}

float & operator［］(int i)

{

\_\_\_\_\_\_\_\_

}

void Print();

};

void Arr::Print()

{int i;

for(i=0;i< this->Getn();i++)

{if (i%10==0)

cout << endl;

cout<<setw(6)<<p［i］;

}

cout<<endl;

}

void main()

{Arr a(20);

for (int i=0;i<a.Getn();i++)

a［i］=i\* 2;

a.Print();

}

3. 下面是一个输入半径，输出其面积和周长的C++程序，在下划线处填上正确的语句。

#include <iostream>

\_\_\_\_\_\_\_\_\_;

\_\_\_\_\_\_\_\_\_;

void main()

{double rad;

cout<<"rad=";

cin>>rad;

double l=2.0\*pi\*rad;

double s=pi\*rad\*rad;

cout<<"＼n The long is："<<l<<endl;

cout<<"The area is："<<s<<endl;}

4. 在下划线处填上缺少的部分。

#include <iostream.h>

class Samp

{public:

void Setij(int a,int b){i=a,j=b;}

~Samp()

{cout<<"Destroying.."<<i<<endl;}

int GetMuti(){return i\*j;}

protected:

int i;

int j;

};

int main()

{Samp \*p;

p=new Samp［5］;

if(!p)

{cout<<"Allocation error＼n";

return 1;

}

for(int j=0;j<5;j++)

p［j］.Setij(j,j);

for(int k=0;k<5;k++)

cout<<"Muti［"<<k<<"］ is:"<<p［k］.\_\_\_\_\_\_\_\_\_\_<<endl;

\_\_\_\_\_\_\_\_\_\_

return 0;

}

5. 请在下面程序的横线处填上适当内容，以使程序完整,并使程序的输出为:

11,10

13,12

#include <iostream.h>

class A

{int a;

public:

A(int i=0){a=i;}

int Geta(){return a;}

void show(){cout<<a<<endl;}

};

class B

{A a;

int b;

public:

B(int i,int j)\_\_\_\_\_\_\_\_\_

{}

void show(){cout<<a.Geta()<<","<<b<<endl;}

};

void main()

{B b［2］={B(10,11),B(12,13)};

for(int i=0;i<2;i++)

\_\_\_\_\_\_\_\_\_\_

}

1. 给出下面程序输出结果。

#include <iostream.h>

class Base

{private:

int Y;

public:

Base(int y=0) {Y=y;cout<<"Base("<<y<<")＼n";}

~Base() {cout<<"~Base()＼n";}

void print() {cout <<Y<< "";}

};

class Derived:public Base

{private:

int Z;

public:

Derived (int y, int z):Base(y)

{Z=z;

cout<<"Derived("<<y<<","<<z<<")＼n";

}

~Derived() {cout<<"～Derived()＼n";}

void print()

{Base::print();

cout<<Z<<endl;

}

};

void main()

{Derived d(10,20);

d.print();

}

10 20

～Derived()

~Base()

［解析］派生类对象，先调用基类构造函数输出Base(10)，后调用派生类构造函数输出

Derived(10,20)，后执行d.print(),调用派生类的print，再调用Base::print()输出10，后返回

输出z的值20。后派生类析构，再基类析构。

2. 给出下面程序输出结果。

#include <iostream.h>

class test

{int x;

public:

test(int i=0):x(i){}

virtual void fun1()

{cout << "test::x"<<x<<endl;}

};

class ft:public test

{int y;

public:

void fun1(){cout <<"ft::y="<<y<<endl;}

ft(int i=2):test(i),y(i){}

};

void main()

{ ft ft1(3);

void (test::\*p)();

p=test::fun1;

(ft1.\*p)();

}

1. 求n(n=3)个学生的最高分和最低分及姓名，已有student类声明和main函数，完成

student类的实现部分。

#include <iostream.h>

#include <string.h>

class student

{char name［10］;

int deg;

public:

student(char na［］="",int d=0);

char \* getname();

friend int compare(student &s1,student &s2);

int getdeg();

};

void main()

{student st［］={student("王强",74),student("李刚",68),student("张雪",84)};

int i=0,min=0,max=0;

for(i=1;i<3;i++)

{if(compare(st［max］,st［i］)==-1)

max=i;

if(compare(st［min］,st［i］)==1)

min=i;

}

cout<<"最高分："<<st［max］.getdeg()<<"姓名:"<<st［max］.getname()<<endl;

cout<<"最低分:"<<(\*(st+min)).getdeg()<<"姓名:"<<st［max］.getname()<<endl;

}

1. 当一个类的某个函数被说明为virtual时，该函数在该类的所有派生类中（）

A. 都是虚函数

B. 只有被重新说明时才是虚函数

C. 只有被重新说明为virtual时才是虚函数

D. 都不是虚函数

2. 以下基类中的成员函数表示纯虚函数的是 （）

A. virtual void vf(int)

B. void vf(int)=0

C. virtual void vf()=0

D. virtual void yf(int){}

3. 下面对静态数据成员的描述中，正确的是（）

A. 静态数据成员可以在类体内进行初始化

B. 静态数据成员可以直接用类名或者对象名来调用

C. 静态数据成员不能用private控制符修饰

D. 静态数据成员不可以被类的对象调用

4. 所谓数据封装就是将一组数据和与这组数据有关操作组装在一起，形成一个实体，这实体

也就是（）

A. 类

B. 对象

C. 函数体

D. 数据块

5. 类B是类A的公有派生类，类A和类B中都定义了虚函数func(),p是一个指向类A对象的指针

，则p->A::func()将（）

A. 调用类A中的函数func()

B. 调用类B中的函数func()

C. 根据p所指的对象类型而确定调用类A中或类B中的函数func()

D. 既调用类A中函数，也调用类B中的函数

6. 在面向对象的程序设计中，首先在问题域中识别出若干个 （）

A. 函数

B. 类

C. 文件

D. 过程

7. 在下列成对的表达式中，运算结果类型相同的一对是（）

A. 7.0／2.0和7.0／2

B. 5／2.0和5／2

C. 7.0／2和7／2

D. 8／2和6.0／2.0

8. 下列不具有访问权限属性的是（）

A. 非类成员

B. 类成员

C. 数据成员

D. 函数成员

9. 以下有关继承的叙述正确的是 （）

A. 构造函数和析构函数都能被继承

B. 派生类是基类的组合

C. 派生类对象除了能访问自己的成员以外，不能访问基类中的所有成员

D. 基类的公有成员一定能被派生类的对象访问

10. 下列有关模板和继承的叙述正确的是 （）

A. 模板和继承都可以派生出一个类系

B. 从类系的成员看，模板类系的成员比继承类系的成员较为稳定

C. 从动态性能看， 继承类系比模板类系具有更多的动态特性

D. 相同类模板的不同实例一般没有联系，而派生类各种类之间有兄弟父子等关系

11. 适宜采用inline定义函数情况是（）

A. 函数体含有循环语句

B. 函数体含有递归语句

C. 函数代码少、频繁调用

D. 函数代码多、不常调用

12. 要采用动态多态性，说法正确的是（）

A. 基类指针调用虚函数

B. 派生类对象调用虚函数

C. 基类对象调用虚函数

D. 派生类指针调用虚函数

13. C++类体系中，不能被派生类继承的有（）

A. 转换函数

B. 构造函数

C. 虚函数

D. 静态成员函数

14. 下列不是描述类的成员函数的是（）

A. 构造函数

B. 析构函数

C. 友元函数

D. 拷贝构造函数

15. 下列不能作为类的成员的是（）

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

16. 下列不是描述类的成员函数的是（）

A. 构造函数

B. 析构函数

C. 友元函数

D. 拷贝构造函数

17. 关于对象概念的描述中，说法错误的是（）

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

18. 派生类的构造函数的成员初始化列表中，不能包含（）

A. 基类的构造函数

B. 基类的对象初始化

C. 派生类对象的初始化

D. 派生类中一般数据成员的初始化

19. 关于new运算符的下列描述中，错误的是（）

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

20. 假定一个类的构造函数为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（）

A. 4和5

B. 4和20

C. 4和24

D. 20和5

1. 在C++中，编译指令都是以\_\_\_（符号）开始。

2. 在函数前面用\_\_\_保留字修饰时，则表示该函数表为内联函数。

3. 执行下列程序

int a=21,j=16;

cout.setf(ios::hex);

cout<<a<<"";

cout.unsetf(ios::hex);

cout<<j<<endl;

程序的输出结果是\_\_\_。

4. 在单继承和多继承方式中，面向对象的程序设计应尽量使用\_\_\_继承。

5. 函数模板中紧随template之后尖括号内的类型参数都要冠以保留字\_\_\_。

6. 在C＋＋中，访问指针所指向的对象的成员使用\_\_\_运算符。

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

8. 局部对象和全局对象中，在同一程序中\_\_\_生存期最长。

9. this指针始终指向调用成员函数的\_\_\_。

10. 派生类的主要用途是可以定义其基类中\_\_\_。

11. 在用class定义一个类时，数据成员和成员函数的默认访问权限是\_\_\_。

12. 使用new为int数组动态分配10个存储空间是\_\_\_。

13. 类模板用来表达具有\_\_\_的模板类对象集。

14. 如果通过同一个基类派生一系列的类，则将这些类总称为\_\_\_。

15. 面向对象的四个基本特性是多态性、继承性、和封装性\_\_\_。

16. 所有模板都是以\_\_\_关键字和一个形参表开头的。

17. 在C++语言中，访问一个对象的成员所用的成员运算符是\_\_\_。

18. 开发一个C++语言程序的步骤通常包括编辑、\_\_\_、连接、运行和调试。

19. 执行下列代码

string str("HelloC++");

cout<<str.substr(5，3);

程序的输出结果是\_\_\_。

20. 定义\_\_\_函数时，应在参数个数或参数类型上有所不同。

1. #include <iostream>

#include <fstream>

#include <string>

using namespace std;

class A

{public:

A(const char \*na){strcpy(name,na);}

private:

char name［80］;

};

class B:public A

{ public:

B(const char \*nm):A(nm){}

void show();

};

void B::show()

{ cout<<"name:"<<name<<endl;

}

void main()

{ B b1("B");

b1.show();

}

2. 下面的程序有错误，请修改。

#include <iostream.h>

class A

{private:

int a;

public:

void func(B &);

A(int i){a=i;}

};

class B

{private:

int b;

friend void A::func(B &);

public:

B(int i){b=i;}

};

void A::func(B& r)

{a=r.b;

cout<<a<<endl;

}

void main()

{ B bt(3);

A at(10);

at.func(bt);

}

3. #include <iostream.h>

class Test

{private:

int x,y=20;

public:

Test(int i,int j){x=i,y=j;}

int getx(){return x;}

int gety(){return y;}

};

void main()

{Test mt(10,20);

cout<<mt.getx()<<endl;

cout<<mt.gety()<<endl;

}

4. #include <iostream.h>

class A

{private:

int x,y;

public:

void fun(int i,int j)

{x=i;y=j;}

void show()

{cout<<x<<" "<<y<<endl;}

};

void main()

{A a1;

a1.fun(2);

a1.show();

}

5. #include <iostream.h>

class A

{private:

int x;

protected:

int y;

public:

A(int i,int j){x=i;y=j;}

};

class B:public A

{public:

　　　B(int a,int b):A(a,b){}

　　　void show(){cout<<x<<,<<y<<endl;}

};

void main()

{B b(8,9);

b.show();

}

1. 完成下面类中成员函数的定义。

#include <iostream.h>

class vehicle

{protected:

int size;

int speed;

public:

void set(int s){speed=s;}

\_\_\_\_\_get(){return speed/10;}

};

class car:public vehicle

{ public:

int get(){return speed;}

};

class truck:public vehicle

{ public:

int get(){return speed/2;}

};

int max(\_\_\_\_\_\_)

{ if(v1.get()>v2.get())

return 1;

else

return 2;

}

void main()

{ truck t;

car c;

t.set(160);

c.set(80);

cout<<max(t,c)<<endl;//此结果输出为2

}

2. 完成下面类中成员函数的定义。

#include <iostream>

#include <string>

using namespace std;

class str

{private:

char \*st;

public:

str(char \*a)

{set(a);

}

str & operator=(\_\_\_\_)

{delete st;

set(a.st);

return \*this;

}

void show(){cout<<st<<endl;}

~str(){delete st;}

void set(char \*s)//初始化st

{\_\_\_\_\_

strcpy(st,s);

}

};

void main()

{str s1("he"),s2("she");

s1.show(),s2.show();

s2=s1;

s1.show(),s2.show();}

3. 下面程序段用来求三角形的面积，首先判断三边不符合组成三角形时，返回-1，符合时输

出三角形面积。

#include <iostream.h>

#include <math.h>

double area(double a,double b,double c)

{if(\_\_\_\_)

return -1;

else

{

double ar,l;

l=(a+b+c)/2;

ar=sqrt(l\*(l-a)\*(l-b)\*(l-c));

return ar;

}

}

void main()

{double i=0,j=0,k=0;

cout<<"输入三角形三边：";

cin>>i>>j>>k;

double s=area(i,j,k);

if(s<0)

cout<<"不是三角形"<<endl;

else

\_\_\_\_\_\_

}

4. 在下面程序横线处填上适当内容，使程序执行结果为：

S=2

S=5

S=9

#include <iostream.h>

void sum(int i)

{static int s;

\_\_\_\_\_\_\_\_\_;

cout<<"s="<<s<<endl;

}

void main (void)

{int i;

for (i=0;\_\_\_\_\_\_\_\_)

sum(i);

}

5. 下面是一个三角形三边，输出其面积C++程序，在下划线处填上正确的语句。

#include <iostream.h>

#include <math.h>

void area()

{double a,b,c;

cout<<"Input a b c:";

\_\_\_\_\_\_\_\_

if(a+b>c&&a+c>b&&c+b>a)

{double l=(a+b+c)/2;

\_\_\_\_\_\_\_

cout<<"The area is:"<<s<<endl;

}

else

cout<<"Error"<<endl;

}

void main()

{area();}

1. 给出下面程序输出结果。

#include <iostream.h>

class example

{int a;

public:

example(int b=5){a=b++;}

void print(){a=a+1;cout <<a<<"";}

void print()const

{cout<<a<<endl;}

};

void main()

{example x;

const example y(2);

x.print();

y.print();

}

2. 给出下面程序输出结果。

#include <iostream.h>

class A

{public:

A()

{cout<<"A 构造函数＼n";fun();}

virtual void fun()

{cout<<"A::fun() 函数＼n";}

};

class B:public A

{public:

B()

{cout<<"B构造函数＼n";fun();}

void fun() {cout<<"B::fun() calle函数＼n";}

};

void main()

{B d;}

1. 写一个程序，定义一个抽象类Shape,由它派生3个类：Square(正方形)、Trapezoid（梯形

）和Triangle三角形。用虚函数分别计算几种图形面积、并求它们的和。要求用基类指针数组

，使它每一个元素指向一个派生类对象。

　　#include <iostream.h>

　　class Shape

　　{public:

virtual double area()const=0;

　　};