1. 类 CBase 的定义如下： 在构造函数 CDerive 的下列定义中，正确的是（）

```
class CBase{
  int x;
  public:
  CBase(int n){x=n;}
};
class CDerive:public CBase{
  CBase y;
  int z;
  public:
  CDerive(int a,int b,intc);
}
```

A CDerive::CDerive(int a，int b，int c)：x(a),y(b),z(c){}

B CDerive::CDerive(int a，int b，int c)：CBase(a),y(b),z(c){}

C CDerive::CDerive(int a，int b，int c)：CBase(a),CDerive(b),z(c){}

D CDerive::CDerive(int a，int b，int c)：x(a),CBase(b),z(c){}

2. 317 在公有派生情况下，有关派生类对象和基类对象的关系，下列叙述不正确的是(　　)

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

3. 对于类的继承方式和可见性，下面哪一种说法是错误的？

A 继承方式有 3 种选择：私有、公有和保护，继承方式决定了基类成员在派生类中的可见性

B 默认为私有继承，但常用的却是公有继承

C 无论哪一种继承方式，派生类中都不能访问其基类中的私有成员，但能访问基类中的保护成员和公有成员

D 在公有继承方式中，派生类没有继承基类的私有成员

4. 在32位系统中，有如下类定义：

```
class A{
  public:
          int fun1();
          virtual  void fun2();
  private:
          int _a1;
          static int _a2;
};
class B:  public A{
  public:
  virtual  void fun2();
}
```

请问sizeof(B)的值为：

A 20　　B 16　　C 12　　D 8

5. 下列程的运行结果是 B1 1&B2&B1 2&B2&C 3 ，请为横线处选择合适的程序（）

```
#include<iosream>
using namespace std;
class B1{
  public:
    B1(int i){
    cout<<”B1”<<””<<i<<”&”;
    }
};
class B2{
  public:
  B2(){
    cout<<”B2”<<”&”;
```

```
    }
  };
  classC:public B1,public B2{
    Public:
    C(int a,int b,int c){
        Cout<<”C”<<”  ”<<indiv<<”&”;
    }
    Private:
      B1 m1;
      B2 m2;
      Int inndiv;
  }
  Int main(){
    C c(1,2,3);
  }
```

A B1(a),m1(b)　　B B2(a),m2(b)　　C B1(a),B2(b)　　D m1(a),m2(b)

6. 下面这段代码运行时会出现什么问题？

```
class A{
  public:
    void f(){  printf("A\n");  }
};
class B: public A{
  public:
    virtual void f(){  printf("B\n");  }
};
int main(){
    A *a = new B;
    a->f();
    delete a;
```

```
    return 0;
}
```

A 没有问题，输出 B　　B 不符合预期的输出 A

C 程序不正确　　D 以上答案都不正确

7. 有如下类定义：

```
class A{
  public:
      int fun1();
      virtual void fun2();
  private:
      int _a1;
      static int _a2;
};
class B: public A{
  public:
      virtual void fun2();
};
```

请问 sizeof(B)的值为：

A 20　　B 16　　C 12　　D 8

8. 下面这段代码运行时会出现什么问题？

```
class A{
  public:
    void f(){
        printf("A\n");
    }
};
class B: public A{
  public:
```

```
    virtual void f(){
        printf("B\n");
    }
};
int main(){
    A *a = new B;
    a->f();
    delete a;
    return 0;
}
```

A 没有问题，输出 B　　B 不符合预期的输出 A

C 程序不正确　　D 以上答案都不正确

9. 在 64 位系统中，有如下类：

```
class A{
  public:
      void *p1;
  private:
      void *p2;
  protected:
      void *p3;
};
class B: public A {};
```

那么 sizeof(B)的数值是（）

A 8　　B 12　　C 16　　D 20　　E 24

10. 有如下程序段：

```
#include <iostream>
using namespace std;
class A {
```

```
    public:
    ~A() {  cout << "~A()";  }
};
class B{
    public:
    virtual ~B() {  cout << "~B()";  }
};
class C: public A, public B {
    public:
    ~C() {  cout << "~C()";  }
};
int main() {
    C * c = new C;
    B * b1 = dynamic_cast<B *>(c);
    A * a2 = dynamic_cast<A *>(b1);
    delete a2;
}
```

则程序输出：

A ~C()~B()~A()　　B ~C()~A()~B()　　C A)B)都有可能　　D 以上都不对

11. 下列程序的运行结果是（）

```
class A{
 public:
 void f() {  cout<<"A::f()";  }
};
class B: public A{
  public:
      void f(){ cout<<"B::f()";  }
};
int main(){
```

```
    B b;
    b.f();
    return 0;
}
```

A A::f()　　B A::f()B::f()　　C B::f()　　D B::f()A::f()

12. 下面有关继承、多态、组合的描述，说法错误的是?

A 封装，把客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏

B 继承可以使用现有类的所有功能，并在无需重新编写原来的类的情况下对这些功能进行扩展

C 隐藏是指派生类中的函数把基类中相同名字的函数屏蔽掉了

D 覆盖是指不同的函数使用相同的函数名，但是函数的参数个数或类型不同

13. 以下程序输出结果是____。

```
class A{
  public:
      virtual void func(int val = 1){  std::cout<<"A->"<<val <<std::endl;  }
      virtual void test() { func();}
};
class B : public A
{
  public:
      void func(int val=0) {std::cout<<"B->"<<val <<std::endl;}
};
int main(int argc ,char* argv[]){
    B*p = new B;
    p->test();
    return 0;
}
```

A. A>0　　B. A>1　　C. B>1　　D. B>0　　E. 编译出错

14. 使用派生类的主要原因是()

A. 提高代码的可重用性　　B. 提高程序的运行效率

C. 加强类的封装性　　D. 实现数据的隐藏

15. 下列关于继承的描述中，正确的是（　　）

A.继承不是类之间的一种关系

B.C++语言仅支持单一继承

C.继承会增加程序的冗余性

D.继承是面向对象方法中一个很重要的特性

16. 在C++, 下列哪一个可以做为对象继承之间的转换

A.static_cast　　B.dynamic_cast

C.const_cast　　D.reinterpret_cast

17.

```
using namespace std;
class Base{
        int x;
        public:
                Base(int b): x(b) {}
                virtual  void display(){ cout << x << endl; }
};
class Derived:  public Base
{
        int y;
        public:
                Derived(int d): Base(d), y(d) {}
                void display(){ cout << y << endl; }
```

```
};
int main(){
        Base b(2);
        Derived d(3);
        b.display();
        d.display();
        Base *p = &d;
        p->display();
        system("pause");
         return 0;
}
```

上面程序的输出结果是（　　）

A.2 2 3　　B.3 2 2　　C.2 3 3　　D.2 3 2

18. 看以下代码：

```
class A{
        public:
        ~A();
};
A::~A(){ printf("delete A ");  }
class B :  public A  {
        public:
        ~B();
};
B::~B(){
        printf("delete B ");
}
```

请问执行以下代码

```
A *pa =  new B();
delete pa;
```

输出的串是（）

A.delete A　　B.delete B

C.delete B delete A　　D.delete A delete B

19. 上述例程调用的Fun函数顺序为（　　）

A.B::Fun, D::Fun　　B.B::Fun, B::Fun

C.D::Fun, D::Fun　　D.D::Fun, B::Fun

20.阅读下面代码：

```
class B{
  public:
        virtual void Fun(){}
};
class D:  public B{
  public:
        void Fun(){}
};
A D dd;          B B* pb = &dd;      C D* pd = &dd;      D pb->Fun();
E pd->Fun();
```

21.

```
#include <iostream>
using namespace std;
class parent{
        int i;
    protected:
        int x;
    public:
      parent(){x=0;i=0;}
      void change(){x++;i++;}
```

```
      void display();
};
class son:public parent{
        public:
                void modify();
};
void parent::display(){cout<<"x="<<x<<endl;}
void son:: modify() {x++;}
int main(){
    son A; parent B;
    A.display();
    A.change();
    A.modify();
    A.display();
    B.change();
    B.display();
}
```

上面程序的输出是（ ）

A. x=1 x=0 x=2　　B. x=2 x=0 x=1

C. x=0 x=2 x=1　　D. x=0 x=1 x=2

22. 32 位机器上，有三个类 A B C 定义如下，请确定 sizeof(A) sizeof(B) sizeof(C)的大小顺序.

```
struct A{
        A() {}
        ~A() {}
        int m1;
        int m2;
};
struct B:A{
```

```
        B() {}
        ~B() {}
        int m1;
        char m2;
        static char m3;
};
struct C{
        C() {}
        virtual~C() {}
        int m1;
        short m2;
};
```

A.A=B=C　　B.A<B<C　　C.A=C<B　　D.A<C<B

23. 下列程序编译时会出现错误，请根据行号选择错误位置(　　)

```
#include <iostream>
using namespace std;
class A{
      int a1;
  protected:
      int a2;
  public:
      int a3;
};
class B:  public A{
      int b1;
  protected:
      int b2;
  public:
      int b3;
```

```
};
class C:private B{
      int c1;
  protected:
      int c2;
  public:
      int c3;
};
int main(){
    B obb;
    C obc;
    cout<<obb.a1;//1
    cout<<obb.a2;//2
    cout<<obb.a3;//3
    cout<<obc.b1;//4
    cout<<obc.b2;//5
    cout<<obc.b3;//6
    cout<<obc.c3;//7
    return 0;
}
```

A.1,2　　B.2,5,7　　C.3,4,7　　D.4,5,6

24.What's the output of the following code?

```
class A{
        public:
        virtual void f(){ cout<<"A::f()"<<endl;  }
        void f()  const {  cout<<"A::f() const"<<endl;  }
};
class B :  public   A{
        public:
```

```
        void f(){  cout<<"B::f()"<<endl;  }
};
g(const A* a){  a->f(); }
int main(){
        A* a =  new B();
        a->f();
        g(a);
        delete a ;
}
```

A.B::f() B::f()const　　B.B::f() A::f()const

C.A::f() B::f()const　　D.A::f() A::f()const

25.　对下面的程序段

```
#include <iostream>
using namespace std;
class CA{
  public:
    virtual void f1(){
        cout<<"CA::f1"<<endl;
        f2();
      }
};
class CB:public:CA{
  public:
    void f1(){
      cout<<"CB::f1()"<<endl;
      f2();
    }
  f2(){ cout<<"CB::f2()"<<endl; }
};
```

```
class CC:publie CB{
  public:
    virtual void f2(){ cout<<"CC::f2()"<<endl; }
};
int main(){
  CC c;
  CA *pa = &c;
  pa->f1();
  return 0;
}
```

编译运行后，程序输出结果是（　）

A.CB::f1() CC::f2()　　B.CB::f1() CB::f2()

C.CB::f1() CA::f2()　　D.CA::f1() CC::f2()

26. 关于子类型的描述中，（ ）是错误的?

A. 在公有继承下,派生类是基类的子类型

B. 子类型关系是不可逆的

C.子类型就是指派生类是基类的子类型

D.一种类型当它至少提供了另一种类型的行为,则这种类型是另一种类型的子类型

27. 请指出下列代表有(　)处错误。

```
#include<stdio.h>
class A{
        public:
            virtual void f(){}
        public :
              A(){ f(); }
};
class B1:public A(){
```

```
        private:
                char *_s;
        public:
                B1()  {  _s=new char[1024];  }
        private:
                void f() {
                  delete _s[];
                  _s=NULL;
                }
}
class B2:public A{
        private:
        int * _m;
        public:
                B2(){ _m=new int(2016);  }
        private:
                virtual void f(){
                        delete _m;
                        _m=NULL;
                }
}
int main(){
        A*a1=new B();
        A*a2=new C;
        delete a1;
        delete a2;
        return 0;
}
```

A.3　　B.4　　C.5　　D.6

28. 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 B 中函数 func()

B.即调用类 A 中函数，也调用类 B 中的函数

C.调用类 A 中函数 func()

D.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()

29.有以下一段代码：

```
#include <iostream>
using namespace std;
class A {
  public :
      void run(void) { cout << "run()" << endl; }
      void run(int a){ cout << "run(A)" << endl; }
};
class B : public A {
  public :
      void run(int a) { cout << "run(B)" << endl;}
};
int main(void){
  B b;
  b.run(0);//语句 1
  b.A::run(1); //语句 2
  b.run(); //语句 3
  b.A::run(); //语句 4
  return 0;
}
```

编译时会产生错误的语句有：</iostream>

A 语句 1　　B 语句 2　　C 语句 3　　D 语句 4

30. 有如下程序：执行后的输出结果应该是：

```
#include <iostream>
class BASE{
  public:
  ~BASE(){std::cout<<"BASE";}
};
class DERIVED: public BASE{
  public:
  ~DERIVED(){std::cout<<"DERIVED";}
};
int main(){ DERIVED x; }
```

A BASE　　B DERIVED　　C BASEDERIVED　　D DERIVEDBASE

31. 在C++, 下列哪一个可以做为对象继承之间的转换

A static_cast　　B dynamic_cast

C const_cast　　D reinterpret_cast

32. 下面代码的输出结果是什么 (　　)

```
#include <iostream>
using namespace std;
class A{
    public: void virtual f() {      cout << "A" << " ";}
};
class B : public A{
    public: void virtual f(){      cout << "B" << " ";}
};
int main(){
    A *pa = new A();
    pa->f();
```

```
    B *pb=(B *)pa;
    pb->f();
    delete pa, pb;
    pa=new B();
    pa->f();
    pb=(B *)pa;
    pb->f();
    return 0;
}
```

A. A A B A　　B. A B B A　　C. A A B B　　D. A A A B

33.在哪种派生方式中，派生类可以访问基类中的 protected 成员（　　）

A public 和 private　　B public 和 protected

C protected 和 private　　D public、protected、private 均可

34.关于派生类，以下说法正确的有

A 从基类接收成员。

B 屏蔽基类成员

C 在声明派生类时增加成员，它体现了派生类对基类功能的扩充。

D 在声明派生类时，还要系统或程序员自定义派生类的构造函数。

35.对于继承方式和可见性，下面哪一种说法是错误的？

A 继承方式有 3 种选择：私有、公有和保护，继承方式决定了基类成员在派生类中的可见性

B 默认为私有继承，但常用的却是公有继承

C 无论哪一种继承方式，派生类中都不能访问其基类中的私有成员，但能访问基类中的保护成员和公有成员

D 在公有继承方式中，派生类没有继承基类的私有成员

36.有一个类 B 继承自类 A，他们数据成员如下：

```
class A {
private：
    int &a;
};
class B : public A {
private:
    int a;
public:
    const int b;
    A c;
    static const char* d;
    A* e;
};
```

则构造函数中，成员变量一定要通过初始化列表来初始化的是____。

A b c　　B b c e　　C b c d e　　D c e

37. 以下关于抽象类的说法正确的有

A.抽象类只能用作其他类的基类

B.不能使用抽象类定义对象

C.抽象类不能用作参数类型、函数返回类型或显式转换的类型

D.抽象类不能有构造函数和析构函数

38. 在C++中，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

39.建立派生类对象时,3 种构造函数分别是 a(基类的构造函数)、b(成员对象的构造函数)、c(派生类的构造函数)这 3 种构造函数的调用顺序为:

A.abc　　B.acb　　C.cab　　D.cba

40. C++类体系中，不能被派生类继承的有?

A 构造函数　　B 静态成员函数　　C 非静态成员函数　　D 赋值操作函数

41.

```
class Base{
    public:
        virtual    ~Base(){std::out<<"Base Destructor"<<std::endl;}
}
class Derived: public Base{
    public :
        ~Derived(){std::out<<"Derived    Destructor" <<std::endl;}
}
Base* p=new Derived();
delete p;
```

上面代码在执行 delete p 时，控制台会输出什么内容（）

A.Base Destructor　　B.Derived Destructor

C.Base Destructor　　D.Derived Destructor

E.Derived Destructor　　F.Base Destructor

42. 一个类如果有一个以上的基类就叫做（）。

A. 循环继承　　B. 单继承　　C. 非法继承　　D. 多继承

43. 下列一段 C++代码的输出是?

```
#include "stdio.h"
class Base{
 public:
```

```
    int Bar(char x){
        return (int)(x);
    }
    virtual int Bar(int x){
        return (2 * x);
    }
};
class Derived : public Base{
public:
    int Bar(char x){
        return (int)(-x);
    }
    int Bar(int x){
        return (x / 2);
    }
};
int main(void){
    Derived Obj;
    Base *pObj = &Obj;
    printf("%d,", pObj->Bar((char)(100)));
    printf("%d,", pObj->Bar(100));
}
```

A.100，-100　　B.100，50　　C.200，-100　　D.200，50

44. 派生类对象可以访问基类成员中的（）？

A. 公有继承的私有成员　　B. 私有继承的公有成员

C. 公有继承的保护成员　　D. 以上都错

45. 193、请选择下列程序的运行结果

```
#include<iostream>
```

```
using namespace std;
class B0//基类 BO 声明{
    public://外部接口
        virtual void display()//虚成员函数 {  cout<<"B0::display0"<<endl;}
};
class B1:public B0//公有派生 {
  public:
    void display() { cout<<"B1::display0"<<endl; }
};
class D1: public B1//公有派生 {
  public:
    void display(){ cout<<"D1::display0"<<endl; }
};
void fun(B0 ptr)//普通函数 {   ptr.display(); }
int main( )//主函数 {
  B0 b0;//声明基类对象和指针
  B1 b1;//声明派生类对象
  D1 d1;//声明派生类对象
  fun(b0);//调用基类 B0 函数成员
  fun(b1);//调用派生类 B1 函数成员
  fun(d1);//调用派生类 D1 函数成员
}
```

A 、B0::display( ) B0::display( ) B0::display( )

B 、B0::display( ) B0::display( ) D1::display( )

C 、B0::display( ) B1::display( ) D1::display( )

D 、B0::display( ) B1::display( ) B1::display( )

46. 当一个派生类公有继承一个基类时，基类中的所有公有成员成为派生类的（）。

A. public 成员　　B. private 成员　　C. protected 成员　　D. 友元

47.
```
class foo {
        public:
        foo(){};
    };
    class boo : public foo {
        boo() : foo() { };
    };
```

以上代码编译时需要什么标准？

A. c++03　　B. c++0x　　C. c++11　　D. c++98

E. 都不正确

48. 请将 B 类的构造函数补充完整，要求用 x 初始化 a. 请为横线处选择合适的程序（　　）

```
class A {
  int a;
    public:
    A(int x=0) { a=x; }
};
class B: public A {
    int b;
    public:
    B(int x): _____________
    { b=x+1; }
};
```

A a(x)　　B A(x)　　C B(x)　　D a=x

49. 当不同的类具有相同的间接基类时，（）。

A 各派生类无法按继承路线产生自己的基类版本

B 为了建立唯一的间接基类版本，应该声明间接基类为虚基类

C 为了建立唯一的间接基类版本，应该声明派生类虚继承基类

D 一旦声明虚继承，基类的性质就改变了，不能再定义新的派生类

50. 下面代码的输出是什么？

```
class A    {
   public:
        A()   {        }
        ~A() {       cout<<"~A"<<endl;      }
};
class B:public A    {
       public:
            B(A &a):_a(a)    {    }
            ~B()    {    cout<<"~B"<<endl;    }
       private:
            A _a;
};
int main(void)    {
            A a;            //很简单，定义 a 的时候调用了一次构造函数
            B b(a);
}
```

A. ~B

B. ~B
   ~A

C. ~B
   ~A
   ~A

D. ~B
   ~A
   ~A

~A

51. 有一个类 B 继承自类 A，他们数据成员如下：

```
class A {
    private：
        int &a;
};
class B : public A {
    private:
        int a;
    public:
        const int b;
        A c;
        static const char* d;
        A* e;
};
```

则构造函数中，成员变量一定要通过初始化列表来初始化的是____。

A b c　　B b c e　　C b c d e　　D c e

E b d　　F. b e

52. 在哪种派生方式中，派生类可以访问基类中的 protected 成员（　　）

A.public 和 private　　B.public 和 protected

C.protected 和 private　　D.public、protected、private 均可

53. 对于继承方式和可见性，下面哪一种说法是错误的？

A 继承方式有 3 种选择：私有、公有和保护，继承方式决定了基类成员在派生类中的可见性

B 默认为私有继承，但常用的却是公有继承

C 无论哪一种继承方式，派生类中都不能访问其基类中的私有成员，但能访问基类中的保护成员和公有成员

D 在公有继承方式中，派生类没有继承基类的私有成员

54. 下面描述中,表达正确的有()

A 公有继承是基类中的 public 成员在派生类中仍是 public 的

B 公有继承是基类中的 private 成员在派生类中仍是 private 的

C 公有继承是基类中的 protected 成员在派生类中仍是 protected 的

D 私有继承是基类中的 public 成员在派生类中仍是 private 的

55. 对于 protected 成员，下面说法错误的是:（）

A 基类可以访问从所有派生类造型（cast）成基类对象的 protected 成员

B 从公有和保护继承的派生类继承的子类里可以访问基类的 protected 成员

C 派生类可以定义一个同名的非 protected 成员

D 派生类可以访问基类对象的公有 protected 成员

56. 从内存实现或者反射的角度来看，关于继承的说法正确的是（）。

注：此处的继承不代表能调用

A. 子类将继承父类的所有的数据域和方法

B. 子类将继承父类的其可见的数据域和方法

C. 子类只继承父类 public 方法和数据域

D. 子类只继承父类的方法，而不继承数据域

57. 现有一个程序如下：

```
#include
class A{
   public:
        void f( ) { cout << “A::f()”; }
};
```

```
class B{
    public:
        void f( ) { cout << “B: f()”; }
        void g( ) { cout << “B: g()”; }
};
class C : public A, public B{
    public:
        void g( ) { cout << ”C::g()”; }
    void h( ) {
        cout << ”C::h()”;
        f(); //语句 1
    }
};
void main(){
    C obj;
    obj.f();      //语句 2
    obj.A::f(); //语句 3
    obj.B::f(); //语句 4
    obj.g(); //语句 5
}
```

则编译时会产生错误的语句有

A. 语句 1　　B. 语句 2　　C. 语句 3　　D. 语句 4　　E. 语句 5

58. 继承是面向对象的三个特点之一，下列关于继承特点的说法正确的有

A. 公有继承的特点是基类的公有成员和保护成员作为派生类的成员时，它们都保持原有的状态，而基类的私有成员仍然是私有的，不能被这个派生类的子类所访问。

B. 私有继承的特点是基类的公有成员和保护成员都作为派生类的私有成员，并且不能被这个派生类的子类所访问。

C. 公有继承的特点是基类的公有成员和保护成员都成为派生类的公有成员，而基类的私有成员仍然是私有的，不能被这个派生类的子类所访问。

D. 保护继承的特点是基类的所有公有成员和保护成员都成为派生类的保护成员，并且只能被它的派生类成员函数或友元访问，基类的私有成员仍然是私有的。

59. 有以下一段代码：

```
class A{
    public:
     A() {}
     ~A() {}
};
class B : public A{
    public:
    B() {}
    ~B() {}
    public:
    int a;
```

};

若：x=sizeof(A),y=sizeof(B)，请问 x，y 的值分别是多少？

A. x=0，y=4　　B. x=1，y=4　　C. x=1，y=5　　D. x=4，y=8

60. 下面程序

```
class A{
   public:
        ~A() { fprintf(stderr, "A"); }
};
class B: public A{
   public:
        ~B() { fprintf(stderr, "B"); }
};
int main(int argc, char *argv[ ]){
      B b;
      return 0;
}
```

的输出是（）

A. BA　　B. AB　　C. A　　D. B

61. 设 A 为基类，AX 为派生类，

```
class A {
```

```
    public:
        A();
        ~A();
        Data* data;
};
class AX : public A {
    public:
        AX();
        ~AX();
        AXData* ax_data;
};
```

以下不会内存泄漏的写法吗？

A. AX* p = new AX(); delete p;

B. A* p = new AX(); delete p;

C. std::shared_ptr<AX> p{new AX()};

D. std::shared_ptr<A> p{new AX()};

62. 看以下代码：

```
class A {
        public:
        ~A();
};
```

```
A::~A() {    printf("delete A ");    }

class B : public A {

      public:

      ~B();

};

B::~B() { printf("delete B "); }
```

请问执行以下代码

```
A *pa = new B();

delete pa;
```

输出的串是（）

A. delete A　　B. delete B　　C. delete B delete A　　D. delete A delete B

63. 通过基类对象名、指针只能使用从基类继承的成员。

A.T　　B.F

64. 以下哪个方法是类 A 的合法继承方法：

```
class A {

      protected int method1(int a, int b) {    return 0;    }

}
```

A. public int method1(int a, int b) {return 0; }

B. private int method1(int a, int b) { return 0; }

C. public short method1(int a, int b) { return 0; }

D. static protected int method1(int a, int b) { return 0; }

65. 继承是面向对象的三个特点之一，下列关于继承特点的说法正确的有

A. 公有继承的特点是基类的公有成员和保护成员作为派生类的成员时，它们都保持原有的状态，而基类的私有成员仍然是私有的，不能被这个派生类的子类所访问。

B. 私有继承的特点是基类的公有成员和保护成员都作为派生类的私有成员，并且不能被这个派生类的子类所访问。

C. 公有继承的特点是基类的公有成员和保护成员都成为派生类的公有成员，而基类的私有成员仍然是私有的，不能被这个派生类的子类所访问。

D. 保护继承的特点是基类的所有公有成员和保护成员都成为派生类的保护成员，并且只能被它的派生类成员函数或友元访问，基类的私有成员仍然是私有的。