1. 虚函数、纯虚函数可以定义为 static 成员函数吗？为什么？

参考答案

不能。因为虚函数、纯虚函数只能用来说明类的实例成员函数（含有 this 指针，用于实现多态），而 static 成员函数不属于任何对象，不含 this 指针。

2. 构造函数、析构函数可以定义为虚函数和纯虚函数吗？为什么？

参考答案

构造函数的作用是构造一个确定的对象，没有多态性，而虚函数和纯虚函数用于实现动态多态，所以构造函数不能声明为虚函数和纯虚函数。
析构函数可以定义为虚函数，这样可以实现析构时的动态多态。析构函数可以定义为纯虚函数，这时派生类不可能来实现基类的析构函数，因此无法实例化。

3. 分析下面的程序，指出错误之处（解释错误原因），并写出 main()函数中每条正确的指令的屏幕输出结果。

```
struct A {
    int a = 0;
    int x = 1;
    void f() { cout << "A::f()"; }
    virtual void g() { cout << "A::g()"; }
    A(int x) { }
};
struct B: A {
    int x = 11;
    int y = 12;
    virtual void f() { cout << "B::f()"; }
    void g() { cout << "B::g()"; }
    void h() { cout << "B::h()"; }
    B(int x): A(x) { }
};
struct C: B {
    int x = 21;
    int y = 22;
    int z = 23;
    void f() { cout << "C::f()"; }
    void g() { cout << "C::g()"; }
    virtual void h() { cout << "C::h()"; }
    C(int x): B(x) { }
} c(1);
int main() {
```

```
    A *p = &c;
    p->f();
    p->g();
    p->h();
    cout << p->a;
    cout << p->x;
    cout << p->y;
    cout << p->z;
    /**/
    B *q = &c;
    q->f();
    q->g();
    q->h();
    cout << p->a;
    cout << q->x;
    cout << q->y;
    cout << q->z;
}
```

参考答案

只有 main()函数中存在错误的指令。

```
int main() {
    A *p = &c;
    p->f();             //A::f()
    p->g();             //C::g()
    p->h();             //错误, 类 A 中没有 h()
    cout << p->a;       //0 (A::a)
    cout << p->x;       //1 (A::x)
    cout << p->y;       //错误, 类 A 中没有数据成员 y
    cout << p->z;       //错误, 类 A 中没有数据成员 z
    /**/
    B *q = &c;
    q->f();             //C::f()
    q->g();             //C::g()
    q->h();             //B::h()
    cout << p->a;       //0 (A::a)
    cout << q->x;       //11 (B::x)
    cout << q->y;       //12 (B::y)
    cout << q->z;       //错误, 类 B 中没有数据成员 z
}
```

4. 指出如下各类可访问的成员及成员的访问权限。

```
class A {
    int  a;
```

```
    protected:
        int  b;
    public:
        int  c;
        ~A( );
    };
    class B: A {
        int  a;
    protected:
        int b;
        A::b;
    public:
        int  c, d;
    };
    class C: protected A {
        int  a;
    protected:
        int  b, e;
    public:
        int  g;
        A::c;
    };
    struct D: B, C {
        int a;
    protected:
        int b, f;
    public:
        int e, g;
};
```

参考答案

class A

private: a
protected: b
public: c, ~A()

class B

private: a, A::(c, ~A())
protected: b, A::b
public: c, d

class C

private: a
protected: b, e, A::(b, ~A())
public: g, A::c

class D

private:

protected: b, f, B::(~~b,~~A::b), C::(b, e, A::(b, ~A()))
public: a, e, g, B::(c, d), C::(g, A::c)

5. 指出如下程序的错误之处及其原因：

```
class A {
      int x;
      virtual int f( ) { return 0; }
      virtual int g( ) = 0;
protected:
      int  y;
public:
     virtual A( ) { }
} a;
struct B: A {
      A::x;
      using A::y;
      long f( )  { return 1L; };
      int g(int) { return 1; }
} b;
A *p = new A;
B *q = new B;
int f(A, B);
A g(B &);
int h(B *);
```

参考答案

(1) virtual A( )中去掉virtual，构造函数不能为虚函数。
(2) 不能定义对象a，因为类A是抽象类。
(3) B中，A::x 错，因为 A::x不能访问。
(4) B中，using A::y 错，因为不能将 A::x原来的 protected 属性扩大为 public。
(5) B中的long f()中的long应改为int，在基类和派生类中不能定义原型相同、只有返回值不同的成员函数。
(6) 不能定义对象b，因为类B是抽象类（含有纯虚函数g( )）。
(7) 不能 new A，因为类A是抽象类。
(8) 不能 new B，因为类B是抽象类。
(9) 不能为函数int f(A, B)产生实参对象，因为类A、B是抽象类。
(10) 函数A g(B &)不能返回A类对象，因为类A是抽象类。

6. 指出如下程序中main()中每行语句的输出结果。

```
struct A { A( ) { cout << 'A'; } };
struct B { B( ) { cout << 'B'; } };
```

```
struct C: A { C( ) { cout << 'C'; } };
struct D: A, virtual B { D( ) { cout << 'D'; } };
struct E: A, virtual B, virtual C {
        D d;
        E( ) { cout << 'E'; }
};
struct F: A, virtual B, virtual C, D, E {
        C  c;
        E  e;
        F( ) { cout << 'F'; }
};
void main(void)
{
     A  a;
     B  b;
     C  c;
     D  d;
     E  e;
     F  f;
}
```

参考答案

```
A  a;       //A
B  b;       //B
C  c;       //AC
D  d;       //BAD
E  e;       //BACABADE
F  f;       //BACAADABADEACBACABADEF
```

F 的构建过程见下图：