1 Slide 11.3

Create a base class A, B1/B2 inheriting A, and C inheriting B1 and B2 **without public keyword**. Add a public function to A, try to use it with a C pointer.

2 Slide 11.4

Demonstrate (while changing inheritance from private to protected to public):

- protected function use

```
  class A
{
protected:
    void a() {}
};

class B : /* protected */ A {
public:
    void b() { a(); }
};

class C : B
{
public:
    void c() { a(); }
};
```

- private function use

3 11.27

```
 class A
{
public:
    virtual void f() { std::cout << v << std::endl; }
    int v;
};

class B1 : public A {
public:
    B1() { v = 1; }
};

class B2 : public A {
public:
    B2() { v = 2; }
};
```

```
class C : public B1, public B2
{
};

int main()
{
    C c;
    c.f()
    // pause
    c.B1::f();
    // pause
    A* a = &c;
    // pause
    A* a = (B1*)&c;
    a->f();
    return 0;
}
```

Add *virtual* inheritance to A for B1 and B2. Add *virtual* function:

```
virtual void B1::f() { std::cout << "B1 " << v << std::endl; }
virtual void B2::f() { std::cout << "B2 " << v << std::endl; }
```

4 11.28

```
 class A
{
public:
    void f() { std::cout << "a" << std::endl; }
    int v;
};

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

Use override keyword. Show the effect of non-virtual destructor Show the effect of calling virtual functions from constructor.