重载，覆盖，隐藏

重载（overload）：
同一个类中，函数名相同，函数签名不同。
覆盖（override）：
子类中存在与基类虚函数相同的函数签名。
隐藏（hide）：
子类中存在与基类中函数相同的函数名。
拷贝构造函数(如字符串类String)：

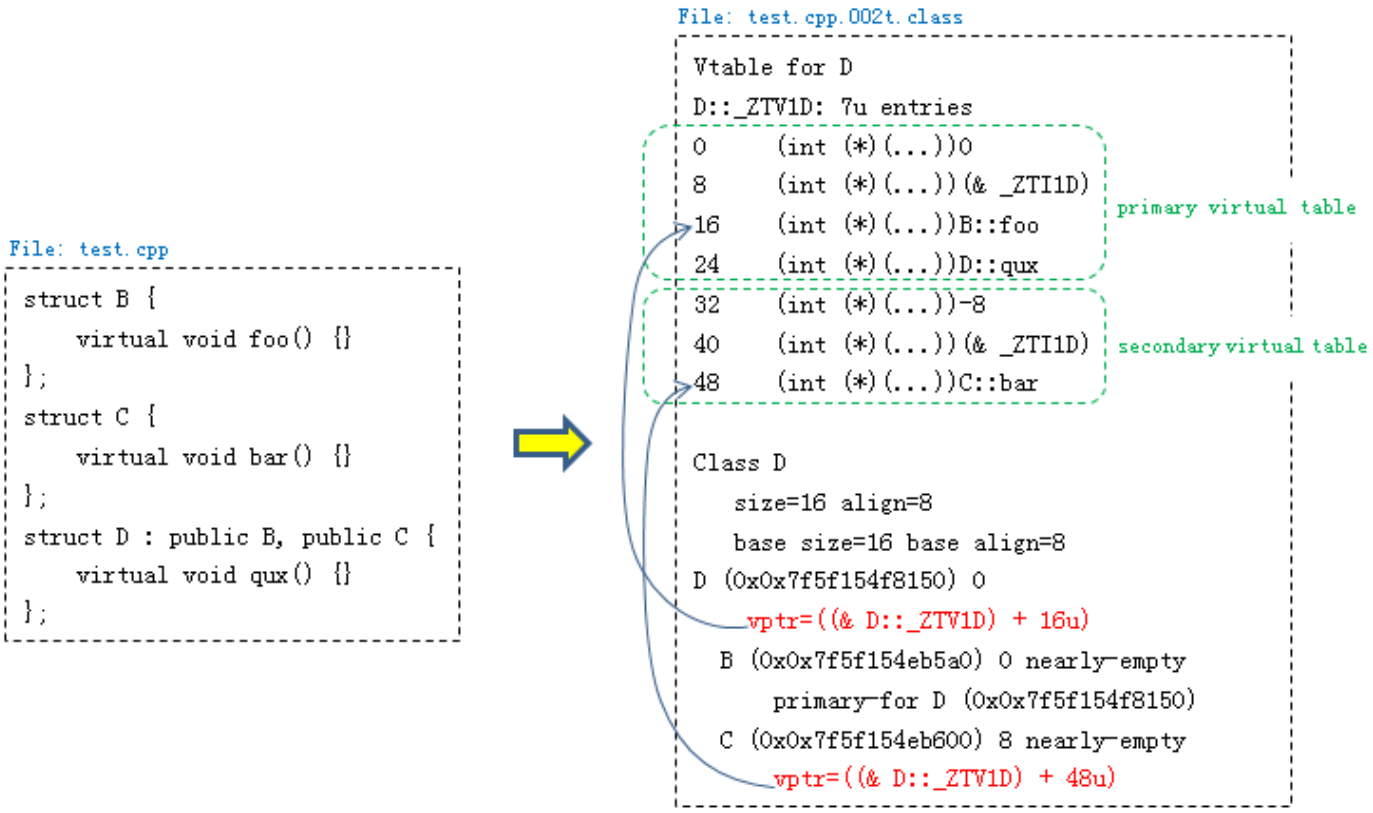
```
String::String(const String& src)
{
   str = malloc(strlen(src.str)+1);
   strcpy(str, src.str);
}
```

赋值函数：

```
String& String::operator=(const
String &src)
{
   if (this != &src) {
      if (str != NULL) free(str);
      str = malloc(strlen(src.str)+1);
      strcpy(str, src.str);
   }
   return *this;
}
```

多态、虚函数表

C++类中的virtual函数指针都被放到vtbl的数组中，称为虚函数表。
如果子类中函数覆盖基类虚函数，则基类中该虚函数指针的位置在子类中会被替换为子类的覆盖函数。虚析构函数也一样的逻辑。
多重继承的时候，子类会有多个虚函数表，每个虚函数表对应一个基类。根据声明顺序，第一个基类的虚函数表称为主虚表，后续的基类虚函数表称为副虚表。
可以通过gcc -fdump-class-hierarchy test.cpp 查看类结构层次。



多态、虚函数表

虚析构函数：把析构函数声明为虚函数，会在每个类的虚函数表中添加一条虚函数指针记录，而子类中该条目会替换成子类的析构函数地址。这样就类似正常的多态机制，delete基类的指针的时候就可以调用子类的析构函数，层层销毁。

```
Class B {
public:
   virtual void foo(){}
};

Class C {
public:
   virtual void bar(){}
   virtual ~C(){}
};

Class D:public B, public C {
public:
   virtual void derive(){}
};
```

```
Vtable for D
D::_ZTV1D: 11u entries
0    (int (*)(...))0
4    (int (*)(...))(& _ZTI1D)
8    (int (*)(...))D::foo
12   (int (*)(...))D::derive
16   (int (*)(...))D::~D
20   (int (*)(...))D::~D
24   (int (*)(...))-8
28   (int (*)(...))(& _ZTI1D)
32   (int (*)(...))C::bar
36   (int (*)(...))D::_ZThn8_N1DD1Ev
40   (int (*)(...))D::_ZThn8_N1DD0Ev

Class D
  size=12 align=4
  base size=12 base align=4
D (0x0xb6e3b4c0) 0
   vptr=((& D::_ZTV1D) + 8u)
 B (0x0xb6e43070) 0
     primary-for D (0x0xb6e3b4c0)
 C (0x0xb6e430a8) 8 nearly-empty
     vptr=((& D::_ZTV1D) + 32u)
```