## **● 请你回答一下为什么析构函数必须是虚函数？为什么C++默认的析构函数不是虚函数 考点:虚函数 析构函数**

将可能会被继承的父类的析构函数设置为虚函数，可以保证当我们new一个子类，然后使用基类指针指向该子类对象，释放基类指针时可以释放掉子类的空间，防止内存泄漏。

C++默认的析构函数不是虚函数是因为虚函数需要额外的虚函数表和虚表指针，占用额外的内存。而对于不会被继承的类来说，其析构函数如果是虚函数，就会浪费内存。因此C++默认的析构函数不是虚函数，而是只有当需要当作父类时，设置为虚函数。

## **● 请你来说一下函数指针**

1、定义

函数指针是指向函数的指针变量。

函数指针本身首先是一个指针变量，该指针变量指向一个具体的函数。这正如用指针变量可指向整型变量、字符型、数组一样，这里是指向函数。

C在编译时，每一个函数都有一个入口地址，该入口地址就是函数指针所指向的地址。有了指向函数的指针变量后，可用该指针变量调用函数，就如同用指针变量可引用其他类型变量一样，在这些概念上是大体一致的。

2、用途：

调用函数和做函数的参数，比如回调函数。

3、示例：

char \* fun(char \* p)  {…}       // 函数fun

char \* (\*pf)(char \* p);             // 函数指针pf

pf = fun;                        // 函数指针pf指向函数fun

pf(p);                        // 通过函数指针pf调用函数fun

## **● 请你来说一下fork函数**

Fork：创建一个和当前进程映像一样的进程可以通过fork( )系统调用：

成功调用fork( )会创建一个新的进程，它几乎与调用fork( )的进程一模一样，这两个进程都会继续运行。在子进程中，成功的fork( )调用会返回0。在父进程中fork( )返回子进程的pid。如果出现错误，fork( )返回一个负值。

## **● 请你来说一下C++中析构函数的作用**

析构函数与构造函数对应，当对象结束其生命周期，如对象所在的函数已调用完毕时，系统会自动执行析构函数。

析构函数名也应与类名相同，只是在函数名前面加一个位取反符~，例如~stud()，以区别于构造函数。它不能带任何参数，也没有返回值（包括void类型）。只能有一个析构函数，不能重载。

如果用户没有编写析构函数，编译系统会自动生成一个缺省的析构函数（即使自定义了析构函数，编译器也总是会为我们合成一个析构函数，并且如果自定义了析构函数，编译器在执行时会先调用自定义的析构函数再调用合成的析构函数），它也不进行任何操作。所以许多简单的类中没有用显式的析构函数。

如果一个类中有指针，且在使用的过程中动态的申请了内存，那么最好显示构造析构函数在销毁类之前，释放掉申请的内存空间，避免内存泄漏。

类析构顺序：1）派生类本身的析构函数；2）对象成员析构函数；3）基类析构函数。

**d、析构函数**

1、概念：一个特殊的成员函数,在对象销毁时,由编译器自动进行调用，完成对象中资源的清理工作

注意：只是清理对象中管理的资源，并不会回收对象的空间

对象如果是堆上的，对象的空间有delete或者delete[]来进行回收

如果是栈上的对象

2、特性：

>>析构函数的名字：类名前添加~

>>用户如果没有显式定义，将由编译器自动生成

编译器生成的析构函数是不会自动释放对象中资源，因此如果类中涉及到资源管理，需要用户显式给出析构函数的定义

>>析构函数一定是无参，没有返回值：不能重载，一个类最多只能有一个析构函数

3、编译器生成的默认的析构函数

4、调用场景：在对象销毁时---->函数结束时调用delete和delete[]

## **● 请你来说一下静态函数和虚函数的区别**

静态函数在编译的时候就已经确定运行时机，虚函数在运行的时候动态绑定。虚函数因为用了虚函数表机制，调用的时候会增加一次内存开销

## **● 请你来说一说重载和覆盖**

重载：两个函数名相同，但是参数列表不同（个数，类型），返回值类型没有要求，在同一作用域中  
重写：子类继承了父类，父类中的函数是虚函数，在子类中重新定义了这个虚函数，这种情况是重写

## **● 请你说一说strcpy和strlen**

strcpy是字符串拷贝函数，原型：

char \*strcpy(char\* dest, const char \*src);

从src逐字节拷贝到dest，直到遇到'\0'结束，因为没有指定长度，可能会导致拷贝越界，造成缓冲区溢出漏洞,安全版本是strncpy函数。  
strlen函数是计算字符串长度的函数，返回从开始到'\0'之间的字符个数。

## **● 请你说一说你理解的虚函数和多态**

多态的实现主要分为静态多态和动态多态，静态多态主要是重载，在编译的时候就已经确定；动态多态是用虚函数机制实现的，在运行期间动态绑定。举个例子：一个父类类型的指针指向一个子类对象时候，使用父类的指针去调用子类中重写了的父类中的虚函数的时候，会调用子类重写过后的函数，在父类中声明为加了virtual关键字的函数，在子类中重写时候不需要加virtual也是虚函数。  
虚函数的实现：在有虚函数的类中，类的最开始部分是一个虚函数表的指针，这个指针指向一个虚函数表，表中放了虚函数的地址，实际的虚函数在代码段(.text)中。当子类继承了父类的时候也会继承其虚函数表，当子类重写父类中虚函数时候，会将其继承到的虚函数表中的地址替换为重新写的函数地址。使用了虚函数，会增加访问内存开销，降低效率。

**多态**

1.多态的概念

多态:同一个事物，在不同场景下表现出的不同的状态

举例子来说明:见人说人话，见鬼说鬼话 学生买票的例子

2.多态的分类

静态多态(早绑定，静态联编): 在编译期间，根据所传递的实参类型或者实例化的类型,来确定到底应该调用那个函数即:在编译期间确定了函数的行为---函数重载、模板

动态多态(晚绑定,动态联编):在程序运行时，确定具体应该调用那个函数

3.动态多态的实现条件-- -在继承的体系中

>>虚函数&重写:基类中必须包含有虚函数(被virtual修饰的成员函数)，派生类必须要对基类的虚函数进行重写

>>关于虚函数调用:必须通过基类的指针或引用调用虚函数

体现:在程序运行时，基类的指针或引用指向那个子类的对象，就会调用那个子类的虚函数

4.重写

>> 1.基类中的函数一定是虚函数

>> 2.派生类虚函数必须与基类虚函数的原型一致：返回值类型 函数名字(参数列表)

例外:a、协变--基类虚函数返回值基类的指针或引用

派生类虚函数返回派生类的指针或引用基类虚函数和派生类虚函数的返回值类型可以不同

b、析构函数:如果将基类中析构函数设置成虚函数,派生类的析构函数提供，两个析构函数就可以构成重写；两个析构函数名字不同

>>3.基类虚函数可以和派生类虚函数的访问权限不一-样

1>C++中构成重写的条件非常严格:虚函数 并且 原型一致

基类虚函数: virtual void TestFunc()

派生类虚函数: virtual void TetsFunc();//细节性的不同不容易发现，而导致重写失败

2>为了让编译器在编译期间帮助用户检测是否重写成功，C++11提供非常有用的关键字

A、override:专门]让编译帮助用户检测派生类是否成功重写了基类的虚函数

如果重写成功:编译通过

如果重写失败:编译失败

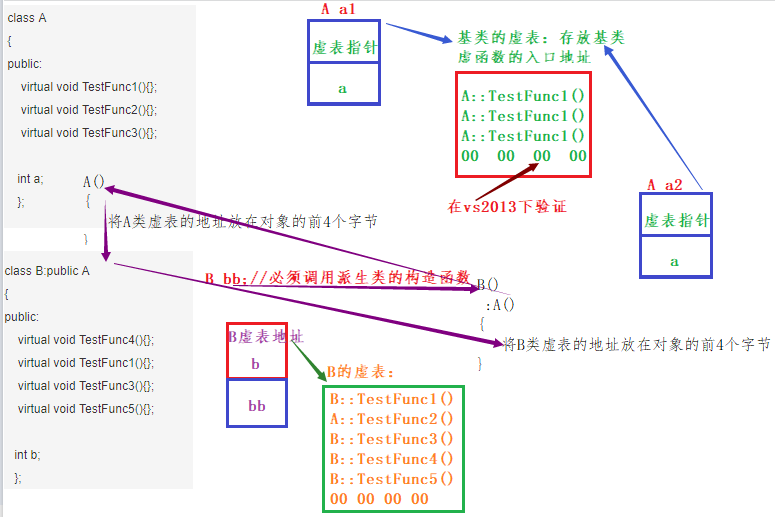
1. final:如果用户不想要子类重写基类的虚函数,可以使用final修饰该关键字

5、多态的实现原理

编译器在编译时会将类中的虚函数按照一定的规则存储在虚表中，在创建对象时，只需要将虚表的地址存储在对象的前四个字节

>>虚表构建过程

1、基类：编译器按照各个虚函数在类中声明的先后次序依次将虚函数保存在虚表中



2.子类虚表的构建过程

a.将基类虚表中内容拷贝一份到子类虚表中

b.如果子类重写了基类中那个虚函数，编译器会用子类自己虛函数的地址覆盖相同偏移量位置的基类虚函数地址

c.如果派生类新增加自己的虚函數，编译器会将派生类新增加的虚函数按照其在派生类中的声明次序依次放在虚表的最后

## **● 请你来说一下C++里是怎么定义常量的？常量存放在内存的哪个位置？**

常量在C++里的定义就是一个top-level const加上对象类型，常量定义必须初始化。对于局部对象，常量存放在栈区，对于全局对象，常量存放在全局/静态存储区。对于字面值常量，常量存放在常量存储区。

## **● 请你来回答一下const修饰成员函数的目的是什么？**

const修饰的成员函数表明函数调用不会对对象做出任何更改，事实上，如果确认不会对对象做更改，就应该为函数加上const限定，这样无论const对象还是普通对象都可以调用该函数。