面试技巧

这里告诉大家面试中的几个技巧:

1、简历上做一个引导:

在词汇上做好区分，比如熟悉Java，了解python，精通c语言

这样的话对自己的掌握程度有个区分，也好让面试官有个着重去问，python本来写的也只是了解，自然就不会多问你深入的一些东西了。

2、在面试过程中做一个引导:

面试过程中尽量引导到自己熟知的一个领域，比如问到你说一下DNS寻址，然后你简单回答（甚至这步也可以省略）之后，可以说一句，自己对这块可能不是特别熟悉，对计算机网络中的运输层比较熟悉，如果有具体的，甚至可以再加一句，比如TCP和UDP

这样的话你可以把整个面试过程往你熟知的地方引导，也能更倾向于体现出你的优势而不是劣势，但是此方法仅限于掌握合适的度，比如有的知识点是必会的而你想往别处引就有点说不过去了，比如让你说几个c++的关键字，你一个也说不上来，那可能就真的没辙了。

3、在自我介绍中做一个引导:

一般面试的开头都会有一个自我介绍，在这个位置你也可以尽情的为自己的优势方面去引导。

4、面试过程中展示出自信:

面试过程中的态度也要掌握好，不要自卑，也不要傲娇，自信的回答出每个问题，尤其遇到不会的问题，要么做一些引导，实在不能引导也可以先打打擦边球，和面试官交流一下问题，看起来像是没听懂

题意，这个过程也可以再自己思考一下，如果觉得这个过程可以免了的话也直接表明一下这个地方不太熟悉或者还没有掌握好，千万不要强行回答。

请问C++11有哪些新特性?

参考回答：
C++11 最常用的新特性如下：
auto关键字：编译器可以根据初始值自动推导出类型。但是不能用于函数传参以及数组类型的推导

nullptr关键字：nullptr是一种特殊类型的字面值，它可以被转换成任意其它的指针类型；而NULL一般被宏定义为0，在遇到重载时可能会出现问题。

智能指针：C++11新增了std::shared_ptr、std::weak_ptr等类型的智能指针，用于解决内存管理的问题。

初始化列表：使用初始化列表来对类进行初始化

右值引用：基于右值引用可以实现移动语义和完美转发，消除两个对象交互时不必要的对象拷贝，节省运算存储资源，提高效率

atomic原子操作用于多线程资源互斥操作

新增STL容器array以及tuple

请你回答一下野指针是什么?

参考回答：
野指针就是指向一个已删除的对象或者未申请访问受限内存区域的指针

说一下C++和C的区别

参考回答：
设计思想上：
C++是面向对象的语言，而C是面向过程的结构化编程语言

语法上：

C++具有封装、继承和多态三种特性

C++相比C，增加多许多类型安全的功能，比如强制类型转换、

C++支持范式编程，比如模板类、函数模板等

用户态和核心态

请你说一说用户态和内核态区别

参考答案

参考回答:
用户态和内核态是操作系统的两种运行级别，两者最大的区别就是特权级不同。用户态拥有最低的特权级，内核态拥有较高的特权级。运行在用户态的程序不能直接访问操作系统内核数据结构和程序。内核态和用户态之间的转换方式主要包括：系统调用，异常和中断。

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

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

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。

静态多态：函数重载和运算符重载属于静态多态
动态多态：派生类和虚函数实现运行时多态

实现：用父类的一个引用去指向子类，如 `Animal & animal = cat;`

谈一谈static的作用

1）static变量的内存只被分配一次，因此其值在下一次调用时仍维持上次的值（因为它是在程序执行前就被定义了）
2）作用周期是程序开始到该程序结束
2）在模块内的static全局变量可以被模块内所有函数访问，但不能被模块外其他函数访问
3）static类成员函数不能接受this指针，所以不能访问普通类成员变量

请谈一谈引用和指针的区别

引用是给已经存在的变量起的别名，它本身不是一个变量，在函数参数中使用引用可以直接使用原始数据

指针是一个变量，指向一块内存，它的值是内存的地址

区别：

- 初始化：引用必须在声明时初始化，指针则都可以
- 赋值：引用不能重复赋值，指针可以重复赋值
- 内存分配上：指针是一个变量需要占据内存，引用不需要
- 大小：sizeof(引用)得到的是被引用对象的大小，sizeof(指针)得到的是4个字节
- 多级：引用只有一级，指针有多级指针
- const：有const指针，没有const引用
- 自增：指针加的是一个单位（如二维数组），引用加的是值

浅拷贝和深拷贝

核心：如果属性有在堆区开辟内存的，一定要自己提供拷贝构造函数，来防止默认浅拷贝带来的问题

浅拷贝：简单的赋值拷贝操作
深拷贝：在堆区重新申请空间，进行拷贝操作

自己定义拷贝构造函数，解决编译器默认浅拷贝带来的问题

官方答案：浅拷贝是增加了一个指针，指向原来已经存在的内存。而深拷贝是增加了一个指针，并新开辟了一块空间让指针指向这块新开辟的空间。浅拷贝在多个对象指向一块空间的时候，释放一个空间会导致其他对象所使用的空间也被释放了，再次释放便会出现错误。

浅拷贝：同一个保险柜，多做一把钥匙（pointer），任何一把钥匙拿走里面的东西，都会导致别的钥匙失去意义

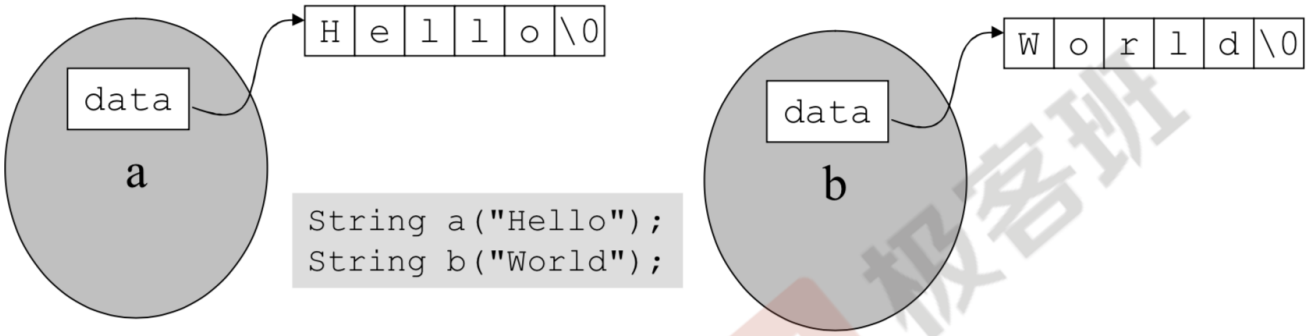
深拷贝：仿造原来的保险柜，造一个新的，再配一把钥匙（pointer），这样，只有自己能打开柜子，十分保险

浅拷贝问题出现在类里有指针，调用拷贝函数会出现浅拷贝问题。
浅拷贝问题就是指针申请空间，浅拷贝是拷贝指针地址。
两个指针指向同一块堆区空间。释放的时候会发生重复释放。造成段错误。

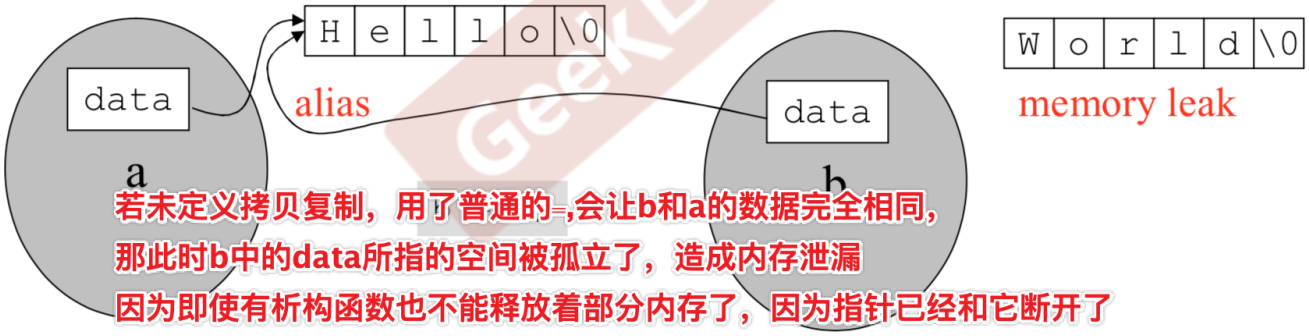
如果没定义拷贝复制，编译器默认是浅拷贝（逐个成员依次拷贝）

浅拷贝只复制指向某个对象的指针（地址），而不复制对象本身，新旧对象还是共享同一块内存。但**深拷贝**会另外创造一个一模一样的对象，新对象跟原对象不共享内存，修改新对象不会改到原对象。

class with pointer members 必須有 copy ctor 和 copy op=



使用 default copy ctor 或 default op= 就會形成以下局面



若未定义拷贝复制，用了普通的=,会让b和a的数据完全相同，
那此时b中的data所指的空间被孤立了，造成内存泄漏
因为即使有析构函数也不能释放着部分内存了，因为指针已经和它断开了

上图不仅会造成内存泄漏，而且当b的值发生改变时，a的值也会发生改变，因为他们指向同一个东西