网易互娱

2019年8月21日

9:05

面经

作者：starrring

链接：<https://www.nowcoder.com/discuss/183113?type=post&order=time&pos=&page=1>

来源：牛客网

上周真的过得非常的跌宕起伏，周一网易，周二腾讯ieg一面，周三腾讯IOS一面，等消息的日子每一天都很煎熬， 每天抱着一丝幻想又觉得幻想破灭，不过好在周五网易终于给发offer了！腾讯的IOS二面我就给回绝了，呜呜呜，其实腾讯IOS的招聘负责人对我好好来着，感觉对不起人家。哎，话不多说，写一篇来回馈前人的面经（面经真的超有用！超有用！超有用！重要的事情说三遍！），希望对后面的人有帮助，也赞赞RP！

首先说下我的准备过程和我面试的题目，最后再附上准备面试过程中收集的资料（其实就是很多前人的面经啦）。（准备过程废话比较多，因为想感慨一下上个星期的心酸的心路历程，看不下去就直接跳过吧）

**准备过程：**

我是前一周收到网易的一面通知，面完网易腾讯就约第二天一面，第三天大概知道自己腾讯ieg挂了以后就去联系了腾讯IOS问还愿意再给我一次机会面试吗，然后就预约了当天晚上一面IOS。所以整个过程中网易的准备最为充分吧。前一个星期我主要花了一天去网上收集了一波前人网易的面经，把所有问题都按类别列了出来，然后大概就了解网 准备过了，有少部分不是，但是都不是很难。整体面下来感觉还行，但当面试官说结束了，我一看才40分钟就问了一下不多问一点吗？面试官说最后结果不根据时间决定的，还说我的基础不错，我就以为我稳了，直到星期二的腾讯ieg，当时面试官也说我的基础不错，但是一来手撕代码我感觉表现不是很好，二来我感觉他们想招的可能是有一定方向（比如图形学、底层操作系统）并在这个方向有一定功底的人，所以周三知道自己挂了以后就开始反思两个面试官说我基础不错可能是没有其他方面的优势就只有说这个来安慰我了，然后从周二开始就没睡过好觉，每天都在担心自己网易也挂了。最后看来腾讯挂我的原因如下：1、楼主面试的太晚了（其实楼主3月就找人内推腾讯的日常实习，但到了4月都既没有人通知我笔试也没人通知我面试，还是收到网易的面试通知那一周楼主碰到一个同学已经被腾讯csig录取，楼主一问才知道原来要找相投的部门里面的员工内推才简历有可能被捞起，楼主当时就找了一个本科的同学问他能不能帮忙内推，但是找他的时候是周五，所以周一才收到面试通知，真的想感慨一下这个机制好鸡肋啊，只有找到了正确的内推被面试的效率才这么高），我的同学跟我说的原话是“别人Offer都拿完了，你才开始投”“我帮你问了好几个，基本都招满了，剩下的都是要求比较高的部门，剩下就看你运气了”2、就和我前面说的一样，他们想招的可能是有一定方向（比如图形学、底层操作系统）并在这个方向有一定功底的人3、手撕代码环节确实表现地差强人意。周三知道自己ieg挂了以后就联系了之前跟我有联系的腾讯IOS部门（这个负责人真的对我超好，本来3月-4月简历一直没人捞他给了我一个机会我拒绝了以后，发现腾讯提前批快结束了，不能白白浪费提前批的机会，我就去找他问还能不能面试，他说搜不到我简历了，后来我在后台刷新了下我的简历，他又联系问我要不要面试，我说下周有网易，下周再联系他，后来腾讯ieg联系我了，我就问他能不把我简历释放了推给ieg，他同意了，等我挂了ieg再找他，他都还愿意接受我，真的非常感谢这个小哥哥了）

**我面试的题目（有些面经里有就不写答案了）：**

**网易：**

前面的主要是聊项目（项目提及的技术问题有：1、图像的模糊原理是啥（楼主有个美图项目）2、如果路径中有障碍物怎么寻路（楼主有个导弹追击小游戏，但是没涉及障碍物，但楼主之前准备了A\*算法，所以就答了上去））

C++:

1、 虚函数的作用以及实现原理？

2、 熟练STL吗? STL中有什么类?

3、 Vector是怎么实现的？

4、 如果vector最开始是2^4，然后每次不够就翻倍，要经过几次的变换达到10000的内存（楼主慌忙之中算了一下大概是2^14接近于10000，就说了大概在13和14次之间吧，然后他就问你忘了最开始的4次方了吗，楼主连忙说“啊，对，应该是10次，不好意思啊”）

算法：

1、在之前回答完A\*算法后又让我简介下最小路径算法（楼主说了Dijkstra和floyd算法）

2、找出一个无序数组n中大小前K个数据（楼主说了O(n)的类似快排的计数算法）

3、这个算法时间复杂度是多少（O(n)）

4、为什么是O(n)（理想情况下每次都在一半里面寻找，1/2+1/4+1/8+…，最后就是O(n)）

5、这个算法时间复杂度的系数是多少（楼主慌忙之中给他算了一下，2）

6、如果k很小怎么办（楼主说了O(nlogk)的最大堆算法）

7、你说的最大堆算法时间复杂度是多少？（O(nlogk)）

8、你觉得比之前的O(n)好吗？（楼主心想好像确实不太对，但是该怎么编呢，就说如果k特别小的话logk是可能小于2的）

9、那k要多小呢？（4）

10、那假如说k=5呢（楼主此时已经冷静下来了，然后就说前面那个算法比较理想的情况系数才等于2，可能分的情况不好的时候就性能很差，而我后面这个算法就很确定是O(nlogk)，后来他就肯定了楼主说其实就是想考察前面这个算法不稳定）

11、如果说内存很小怎么办（楼主就说还是会采用第二个算法，因为只要满足k个大小的内存，后面的数据只要依次调入内存和最大堆堆顶元素进行比较就可以了）

12、如果k比内存还大怎么办（楼主就慌了，想到外部排序和归并，但具体怎么实现的就记不太清了，慌忙中就说了下外部排序和败者树（亚军一定是和冠军比输了的），其实也没怎么答好吧，后来他就没有继续问算法了）

网络：

1、 学过网络吗，都学了些啥（楼主也不可能详细展开啊，就说主要是学7个层嘛）

2、 了解TCP吗（学过，正想他会问TCP3次握手吧，他就又发问了）

3、 有实现过TCP吗，知道是怎么实现的吗（楼主内心一慌，就说以前实验有模拟过一次，但是太久远记不太到了，主要就是模拟客户端和服务端，好像调用了一个system库）

4、 是linux下的库吗（不是，windows下的）

5、 那具体干了什么（好久以前做的，不记得了，楼主就扯了下TCP三次握手）

有什么问题想问他吗？

End

**腾讯ieg：**

1、 虚函数的作用以及实现原理？

2、 一个空类占多少内存

3、 一个空类继承了有虚函数的基类占多少内存，为什么？

4、 如果基类里有多个虚函数这个空类占多少内存？

5、 构造函数可以是虚函数吗

6、 析构函数可以是虚函数吗

7、 Struct{int a;char b;int c;int d;char e;char f}占多少

8、 Void func(char \*ptr){ptr=(char\*)malloc(char);}

int main(){char \*ptr=null;func(ptr);strcpy(ptr,”hello world”);}

可以运行吗？错在哪？怎么改？改完还有一个错是什么？

9、 void func(void \*pSrc,void \*pdest,int length){}

实现一个底层函数，把pSrc内容拷到pDest里，pSrc和pDest分配好内存，底层函数不需再分配内存，length是需要拷贝的长度（问了他才告知分配的长度可能不够用，最后写完了他说可能有交叉，然后pdest在psrc前怎么复制，pdest在psrc后怎么复制）

有什么问题想问他吗？

**腾讯IOS:**

主要问的项目（楼主的美图软件项目是IOS的）

1、 生命周期时间（viewdidload等6个）

2、 如何把合成后的图片取出来（记不太到了，但是记得是一个函数直接拿出来的，后来看了下自己的代码是getImageFromView）

3、 除了ImageView还用了什么（Slider,fresh，手势）

4、 还记得手势的函数名吗

5、 为什么用到fresh(之前实验做学生管理系统用过，里面有tableview)

6、 讲一下tableview的从创建到释放的过程（我们创建直接用的MainStory，就没深究了）

7、 讲一下滤镜的实现（给他讲了一下我实现的滤镜）

8、 讲一下模糊的实现（直接掉的库，GPUImage）

9、 还记得CPUImage的过程吗（其实不太记得了，但是是很模板化的，讲了一下有首先选一个filter,然后有个process把fiter加进去）

算法：

1、 排序算法有哪些

2、 说一下快排的实现，时间复杂度，最坏时间复杂度

3、 说下堆排的实现，时间复杂度，最坏时间复杂度

操作系统：

1、 死锁

2、 死锁的四个条件

有什么想问他的

总结：其实全程就问了20-30min，说我Ios基础比较差，但是因为楼主可实习时间较长，只要愿意学就行，就给过了一面，让我准备下算法和操作系统等基础知识，以备二面

**我准备的资料（前方高能！！！1w+的字居然能传上去，本来还想用文档上传百度云的，牛客网NB）**

**网易：**

**C++方面的问题：**

1、虚函数的作用以及实现原理

2、overload以及overwrite的区别

1）覆盖override：派生类函数覆盖基类函数, 基类函数必须有virtual 关键字。

2）重载overload：可以将语义、功能相似的几个函数用同一个名字表示，但参数不同（包括类型、顺序不同），即函数重载。

3）重写overwrite：派生类的函数屏蔽了与其同名的基类函数

3、overload的话，只有函数返回值类型不同，会重载吗

不会

4、一个空的class类里有什么

1）构造函数

2）拷贝构造函数

3）析构函数

4）赋值运算符重载

5）取地址操作符重载

6）被const修饰的取地址操作符重载

5、struct S{char a; int b; static long c; }请问sizeof(S)是多少？为什么，有什么好处？

涉及到内存对齐机制

静态数据成员被编译器放在程序的一个global data members中，它是类的一个数据成员．但是它不影响类的大小，不管这个类实际产生了多少实例，还是派生了多少新的类，静态成员数据在类中永远只有一个实体存在。

而类的非静态数据成员只有被实例化的时候，他们才存在．但是类的静态数据成员一旦被声明，无论类是否被实例化，它都已存在．可以这么说，类的静态数据成员是一种特殊的全局变量．

所以该类的size为：32位系统上，是8个字节，字节对齐，方便寻址操作（当CPU试图读取的数值没有正确的对齐时，CPU可以执行两种操作之一：产生一个异常条件；执行多次对齐的内存访问，以便读取完整的未对齐数据，若多次执行内存访问，应用程序的运行速度就会慢）

6、用过动态指针吗

7、熟练STL吗

8、STL中有什么类（STL中有向量类）

1）vector: 内部实现是数组，一段连续的内存。

2）list， 内部实现是双链表

3）deque 内部实现是内存块的链表。

4）string： 连续的内存

5）set，map： 红黑树(平衡二叉树的一种)

6）hash\_map, hash\_set 用哈希表(散列表)来实现。

7）stack: 用vector或者是deque来实现

8）queue,用deque实现

9、由于用的是Pure C,把纯虚函数和虚函数弄混了，vtable答了上来，普通的继承多态啥的根本不问，问我有没有用过 C++和C交叉编译(没用过)、STL库(我没用过), 图形学的一些知识(不会)我当时真是尴尬万分…不过我说我用Lambda表达式，Java是1.8 的特性，C++里面是C++11的特性，面试官就问我Lambda的用法和好处，然后问我其他C++11的新特性有没有用过(没…)

10、const的作用，宏定义与const的区别

11、定义一个class，编译器的内存分配

1）类的大小为类的非静态成员数据的类型大小之和，也 就是说静态成员数据不作考虑。

2）普通成员函数与sizeof无关。

3）虚函数由于要维护在虚函数表，所以要占据一个指针大小，也就是4字节。

4）类的总大小也遵守类似class字节对齐的，调整规则。

12、函数重载，模板template，用法和区别

13、多态、虚函数、智能指针

14、指针与引用的区别

15、还有一些C++库函数的实现。

16、inline关键字是做什么用的？inline关键字在什么情况下会展开失败？

inline类似于宏替换，使用函数体替换调用处的函数名，省去了调用函数的开销，增快了代码的执行效率。但是又不是宏替换，inline函数是真正的函数，编译器会考虑语义。

函数体内代码长度过大，包含复杂的结构控制语句（while，switch）,包含内联函数本身，含有递归均会导致展开失败。

17、sizeof一个空类是多大？为什么？编译器为什么这么做？如果添加一个构造函数和析构函数呢？

1个字节，任何一个实例在内存中都占有一定的空间，也就有一个独一无二的地址，为了达到这个目的，编译器往往会给一个空类隐含的加一个字节，这样空类在实例化后在内存得到了独一无二的地址

调用构造函数和析构函数只需要知道函数的地址即可，而这些函数的地址只与类型相关，而与类型的实例无关，编译器也不会因为这两个函数而在实例内添加任何额外信息

18、在这个类中添加一个virtual函数后再sizeof，这时是多大？为什么？

C++编译器一旦发现一个类型有虚函数，就会为该类型生成虚函数表，并在该类型的每一个实例中添加一个指向虚函数表的指针，在32位系统中，一个指针占4个字节，因此sizeof得到4个字节

19、将这个类再virtual继承一个其它的空类，这是多大？为什么？

12个字节，这个类本身大小为4个字节，空类的大小为1个字节，加上虚基类偏移量表指针4个指针，又因为要指针对齐（4个字节），故一起12个字节（虚基类偏移量表不占用类的存储空间，表中记录了虚基类与本类的偏移地址；通过偏移地址，这样就找到了虚基类成员）

20、类有哪几种权限，分别说明？

private，protected，public

21、class A :class B{},A是私有继承还是？ 私有继承是做什么用的？

默认是私有继承，私有继承后，基类所有成员在派生类中为private成员。私有基类的public成员和protected成员在私有派生类中的访问属性相当于派生类中的私有成员，即派生类的成员函数能访问它们，而在派生类外不能访问它们。私有基类的私有成员在派生类中称为不可访问的成员，只有基类的成员函数可以引用它们。

22、子类的虚函数中能不能调用父类的虚函数，为什么？

23、有纯虚函数的类能不能实例化？

不能，有纯虚函数的类是抽象类，只能被继承，不能实例化。包含纯虚函是的类派生出来的类都必须重写这个纯虚函数

24、C++多态有哪几种？

静态多态（函数重载和运算符重载），是在编译的时候，就确定调用函数的类型；动态多态（虚函数实现），在运行的时候，才能确定调用的是哪个函数，动态绑定。运行基类指针指向派生类的对象，并调用派生类的函数。

a.应用形式上：静多态是发散式的，让相同的实现代码应用于不同的场合。动多态是收敛式的，让不同的实现代码应用于相同的场合。

b.思维方式上：静多态是泛型式编程风格，它看重的是算法的普适性；动多态是对象式编程风格，它看重的是接口和实现的分离度。

25、C++是怎么实现动态多态的？

虚函数表和指向虚函数表的vptr指针。这个需要注意vptr指针的分布初始化问题，是在构造函数之后，初始化列表和函数体之前完成的。

26、对象中的VPTR指针什么时候被初始化？

Vptr指针初始化的过程：

a.对象在创建的时,由编译器对VPTR指针进行初始化

b.只有当对象的构造完全结束后VPTR的指向才最终确定

c.父类对象的VPTR指向父类虚函数表

d.子类对象的VPTR指向子类虚函数表

当定义一个子类对象的时候比较麻烦，因为构造子类对象的时候会首先调用父类的构造函数然后再调用子类的构造函数。当调用父类的构造函数的时候，此时会创建Vptr指针（也可以认为Vptr指针是属于父类的成员，所以在子类中重写虚函数的时候virtual关键字可以省略，因为编译器会识别父类有虚函数，然后就会生成Vptr指针变量），该指针会指向父类的虚函数表；然后再调用子类的构造函数，此时Vptr又被赋值指向子类的虚函数表。

（执行父类的构造函数的时候Vptr指针指向的是父类的虚函数表，所以只能执行父类的虚函数）

上面的过程是Vptr指针初始化的过程。

这是因为这个原因，在构造函数中调用虚函数不能实现多态。

27、简要说说C++的静态多态?

函数重载和运算符重载，见上上题。

28、C++编译后的函数符号和C语言编译后的函数符号有哪些区别？为什么

C++语言支持函数重载，C语言不支持函数重载。函数被C++编译后在库中的名字与C语言的不同。假设某个函数的原型为void func(int x,int y)。该函数被C编译器编译后在库中的名字为\_foo，而C++编译器则会产生像\_foo\_int\_int之类的名字。 C++中提供了C连接交换指定符号 extern "C" 解决名字匹配问题。

29、C++智能指针有哪些？auto\_ptr和share\_ptr有什么区别？他们有什么作用？

STL一共给我们提供了四种智能指针：auto\_ptr , unique\_ptr , shared\_ptr和weak\_ptr

auto\_ptr的初衷是用来实现智能指针的，实现内存的自动回收。那么如何实现智能的呢？智能指针最基本的概念是引用计数，也就是智能指针内部有一个计数器，记录了当前内存资源到底有多少指针在引用（可以引用这个资源），当新增加一个可以访问这个资源的引用时，计数器会加1，反之会减去1，当计数器为0时，智能指针会自动释放它所管理的资源。手动申请，自动释放，就是智能的体现。

30、有序vector和list二分查找的时间复杂度分别是多少？

vector的二分相当于数组的二分，时间复杂度是O(logn)，list没办法二分，只能每次从头到尾找，时间复杂度为O(n)。

31、vector自动扩容是按什么大小进行的？

缺省的情况下vector的扩展机制是按2倍大小进行扩展的。在整个大小扩展的过程中，主要的步骤是：a.为需要的新容量分配足够的内存；b.将元素从原来的内存拷贝到新内存中去；c.销毁原来内存中的元素；d.归还原来的内存。

32、构造函数可以调用虚函数吗？语法上通过吗？语义上可以通过吗？

语法可以通过，但是语义不对。

总结来说：基类部分在派生类部分之前被构造，当基类构造函数执行时派生类中的数据成员还没被初始化。如果基类构造函数中的虚函数调用被解析成调用派生类的虚函数，而派生类的虚函数中又访问到未初始化的派生类数据，将导致程序出现一些未定义行为和bug，因此c++不让你走这条路。

33、析构函数可以抛出异常吗？为什么不能抛出异常？除了资源泄露，还有其他需考虑的因素吗？

1）如果析构函数抛出异常，则异常点之后的程序不会执行，如果析构函数在异常点之后执行了某些必要的动作比如释放某些资源，则这些动作不会执行，会造成诸如资源泄漏的问题。

2）通常异常发生时，c++的机制会调用已经构造对象的析构函数来释放资源，此时若析构函数本身也抛出异常，则前一个异常尚未处理，又有新的异常，会造成程序崩溃的问题。

34、c++中类型转换机制？各适用什么环境？dynamic\_cast转换失败时，会出现什么情况？（对指针，返回NULL.对引用，抛出bad\_cast异常）

对指针进行dynamic\_cast，失败返回null，成功返回正常cast后的对象指针；

对引用进行dynamic\_cast，失败抛出一个异常，成功返回正常cast后的对象引用。

35、拷贝构造函数作用及用途？什么时候需要自定义拷贝构造函数？

1) 一个对象以值传递的方式传入函数体； 2) 一个对象以值传递的方式从函数返回； 3) 一个对象需要通过另外一个对象进行初始化；

36、c++里面的虚函数的原理和实现

37、平时开发当中多态用的多么？

38、 多态的开销有多大？

39、菱形继承的虚函数的开销说一下

40、malloc是如何实现的？

malloc基本的实现原理就是维护一个内存空闲链表，当申请内存空间时，搜索内存空闲链表，找到适配的空闲内存空间，然后将空间分割成两个内存块，一个变成分配块，一个变成新的空闲块。如果没有搜索到，那么就会用sbrk()才推进brk指针来申请内存空间。

41、如果物理内存是2G 如果mallco 4G可以么？会有什么问题？

malloc的实现与物理内存自然是无关的，分配到的内存只是虚拟内存，而且只是虚拟内存的页号，代表这块空间进程可以用，实际上还没有分配到实际的物理页面。

42、能说一下STL几种容器么说一下这些容器的对比

43、vector和list的使用场景区别

1 如果你需要高效的随即存取，而不在乎插入和删除的效率，使用vector

2 如果你需要大量的插入和删除，而不关心随即存取，则应使用list

3 如果你需要随即存取，而且关心两端数据的插入和删除，则应使用deque

44、C++析构和构造的顺序，为什么析构函数最好是虚函数

1）构造函数顺序：先基类、再数据成员中是类对象的构造函数、最后派生类构造函数的函数体

2）析构函数顺序：与构造函数相反

3）析构函数最好是虚函数：若派生类有一个指向动态内存分配的数据成员，而又将基类的指针指向派生类对象，同时基类的析构函数又不是虚函数的话，编译器就实施静态绑定，释放基类指针所指对象的空间时候只执行基类的析构函数，不执行派生类的析构函数，那派生类动态分配的数据成员所申请的空间就不能被释放，这就造成了内存泄漏。

45、虚函数和虚函数表

1）虚函数：声明成员函数为虚函数以后，就可以实现动态绑定，也就是基类指针可以指向派生类对象，实现相同函数名的派生类的特定行为

2）虚函数表：就是用来运行时查询，帮助系统将某一函数名绑定到虚成员函数表中特定入口地址

46、智能指针了解么，全部都讲一下怎么用，会出现什么问题

47、知道c++11新特性吗，知道智能指针吗，知道智能指针实现吗，来你给我马上实现一个智能指针【要求写出成员和构造函数析构函数】

48、C 简单问了下static，然后就转战别的基础了。

**数据结构和算法：**

1.     向量和队列有什么区别

1）向量：能高效的进行随机存取，时间复杂度为o(1); 在进行插入和删除操作时，会造成内存块的拷贝，时间复杂度为o(n)。

2）队列：能高效的进行随机存取，时间复杂度为o(1);在内部方便的进行头尾部的插入和删除操作，时间复杂度为o(n)。

2.     向量和队列的实现原理是什么（当时我说可以用链表以及数组，面试官就问我链表如何实现向量的随机访问，然后我就意识到链表不能实现向量了，因为不能实现随机访问，这个问题我答地比较糟糕）

1）向量: vector和数组类似，拥有一段连续的内存空间，并且起始地址不变。因此能高效的进行随机存取，时间复杂度为o(1);但因为内存空间是连续的，所以在进行插入和删除操作时，会造成内存块的拷贝，时间复杂度为o(n)。另外，当数组中内存空间不够时，会重新申请一块内存空间并进行内存拷贝。

2）队列:deque的元素数据采用分块的线性结构进行存储，deque分成若干线性存储块，称为deque块。块的大小一般为512个字节，元素的数据类型所占用的字节数，决定了每个deque块可容纳的元素个数。

所有的deque块使用一个Map块进行管理，每个Map数据项记录各个deque块的首地址。Map是deque的中心部件，将先于deque块，依照deque元素的个数计算出deque块数，作为Map块的数据项数，创建出Map块。以后，每创建一个deque块，都将deque块的首地址存入Map的相应数据项中。

3. 什么是优先队列（当时我回答说优先队列可以用堆实现，面试官就问了第4个问题）

优先队列队首元素一定是当前队列中优先级最高的一个

优先队列可以用堆实现

4. 堆是用最大堆还是最小堆实现优先队列，为什么？

默认是用最大堆实现优先队列（priority\_queue<int, vector<int>, less<int> >默认衍生自vector,数字大的优先级越高）

5. 如果返回堆中最大的元素，要怎么做？

取堆顶元素，如果要取完后要删除，则把最后一个元素移至第一个元素，并将size自减1，然后从堆顶元素自上向下进行整堆

6. 如果堆中某元素的序号是5，那他两个自孩子的序号分别是多少？

10、11

7. 说一下快排的时间复杂度

平均复杂度O(nlogn),最坏情况下O(n^2),最好情况下O(nlogn)

8. 什么样的情况是快排的最坏情况，举个例子

元素均有序是快排的最坏情况

9. 如何解决快排的的最坏情况（我说的随机打乱）

选择划分元的时候随机进行选择

10. 说说随机打乱的具体实现

11. 给一系列整形数，其中除了一个数只有一个之外，其他数都有两个，请设计算法找到只有一个的那个数（我开始回答计数排序，后来他又问我有没有别的，我就说全放到set里，可以排除所有那些添加进set里让set元素个数不增加的所有元素，剩下的就是要找到那个数，不知道对不对，感觉应该还有更好的方法）

从头到尾将这n个数异或一遍，得到的数即为要找的那个数。

一个整型数组里除了两个数字之外，其他的数字都出现了两次。请写程序找出这两个只出现一次的数字。

思路:1）如果把数组中的所有数字都依次异或一遍，则可以消掉成对出现的数字，那么还有两个数字是单一的，肯定也不同，那么最终异或的结果肯定不是0。表示在二进制中肯定有一位是1，那么两个不同的数字，一定有一个在该位为1，另一个在该位为0。如果将整个数组按照该位是否为1分为两部分，那么这两部分各自包含一个单一数字。

12、一串数只出现一次，给定一个输入，让返回所有数对，数对中的两个数之和等于输入。我考虑了1min回答两种思路，第一种快排后两边向中间遍历，大了右边-1，小了左边+1, O(nlogn)的复杂度，然后就是说不排序就和冒泡一样遍历，O(n2)的复杂度。

13、问了游戏排行榜的数据结构应该怎么设计

我先说只取前几名的话堆排序，如果不是前几名的话要看查询多还是改动多，然后说目前只能想到BST。然后又聊会说可以先分组，再排序，就和Query 的select一样，最后join就可以了。后来上网上看了下，网上答案是桶排序和红黑树，虽然自己回答的不精确，不过看来思路还是差不多的。

14、要求我写快排，基本上就是说下思路就可以了，没什么难度。

15、数据库B+树（不了解B+树，扯了扯红黑树）

16、给你一个表（数据很大），有用户名和数据，如何快速检索某条数据

对索引排序+二分查找，对索引建表，在新表里可以用hash、分区等操作(B+树)

17、对数据压缩熟悉不（不熟悉，实习的时候直接调用导师接口，面试官就没接着问）

18、只有大小写英文字母的文本文档，数据量很大，如何压缩表示（因为我说了不懂数据压缩，就出了一个相关问题）

哈夫曼编码+详细实现过程，还可以对重复出现的字母加下标，对重复出现的子串编码（后面两种方法针对可能的具体问题，主要还是哈夫曼编码

19、游戏中，地图很大，英雄的技能释放半径和英雄的坐标已知，如何知道每个英雄的技能范围内的对手（维护一个以技能半径为阈值的大根堆，范围内的入堆，范围外的出堆，面试官不太满意，说不够全面，说只给出了一个环节的方法）

20、一个数组，把所有奇数排在偶数前面，且保证奇数和奇数相对位置不变，偶数和偶数相对位置不变

再用一个数组

21、检测npc之间的距离

22、Hash的知识。（hash的构建与冲突处理。）

1）构建：a直接定址法b除留余数法c数字分析法d平方取中法e折叠法

2）冲突处理：a.开放定址法（线性探测、平方探测、再散列）b拉链法

23、hash冲突，怎么解决（散链表，双重hash，等等）

24、图的搜索有哪几种方式？广搜要怎么做？需要什么额外空间吗

DFS和BFS，其中BFS需要开辟队列内存，DFS需要栈。

25、给定一个迷宫，部分坐标是无法通过的，求某两点间最短路径？（广搜+并查集）

从起点到终点的最短路径其实就是一个建立队列的过程：

1）从起点开始，先将其加入队列，设置距离为0；

2）从队列首端取出位置，将从这个位置能够到达的位置加入队列，并且让这些位置的距离为上一个位置的距离加上1；

3）循环2直到将终点添加到队列中，这说明我们已经找到了路径；

注意到在这个过程中，每次处理的位置所对应的距离是严格递增的，因此一旦找到终点，当时的距离就是最短距离；

26、简述Dijkstra算法的过程，描述一下A Star算法

A\*算法：

1）把起点加入 open list 。

2）重复如下过程：

a.遍历 open list ，查找 F 值最小的节点，把它作为当前要处理的节点。

b.把这个节点移到 close list 。

c.对当前方格的 8 个相邻方格的每一个方格？

◆     如果它是不可抵达的或者它在 close list 中，忽略它。否则，做如下操作。

◆     如果它不在 open list 中，把它加入 open list ，并且把当前方格设置为它的父亲，记录该方格的 F ， G 和 H 值。

◆     如果它已经在 open list 中，检查这条路径 ( 即经由当前方格到达它那里 ) 是否更好，用 G 值作参考。更小的 G 值表示这是更好的路径。如果是这样，把它的父亲设置为当前方格，并重新计算它的 G 和 F 值。如果你的 open list 是按 F 值排序的话，改变后你可能需要重新排序。

d.停止，当你

◆     把终点加入到了 open list 中，此时路径已经找到了，或者

◆     查找终点失败，并且 open list 是空的，此时没有路径。

3）保存路径。从终点开始，每个方格沿着父节点移动直至起点，这就是你的路径。opened\_table：采用优先队列实现的小顶堆，用于存放待扩展结点，同时利用F值作为排序指标；

closed\_table：采用set（红黑树）实现，便于快速查找格是否存在于closed\_table中；

27、找出一个无序数组中大小后K个数据？

类似于快速排序的思想，随机选取一个元素，把所有小于等于这个元素的数据移到左边，所有大于这个元素的数据移动到右边。

如果这个元素成了第K个数，直接返回这个数。如果左边的个数大于K，不管右边的数了，在左边重复上面的过程。如果左边的个数等于T<K,不管左边的数了，重复上面的过程，只是K=K-T-1。平均情况下，第一次划分的时间复杂度是O(N)，第二次就是O(N/2),总共是O(n+n/2+n/4+...)=O(n)

28、Set的底层实现是什么？红黑树是做什么用的？额外开销是多少？

set的底层实现是红黑树。红黑树是一种平衡二叉查找树。a结点是红色或黑色。b根节点是黑色，每个叶子结点都是黑色，c每个红色结点的两个孩子结点都是黑色。d从每个叶子到根的所有路径上不能有两个连续的红色结点。e从任一结点到其每个叶子的所有路径都包含相同数目的黑色结点。

红黑树和AVL树一样都对插入时间和删除时间以及查找时间提供了最好可能的最坏保障。时间复杂度是O(logn)，需要额外的空间也是O(logn)

29、给定1000亿个数据，要找出其中最大的一个值，内存只有1G？

大文件变小文件，然后每个文件里hash\_map统计最大的值，然后再归并排序。

30、给定1000亿个数据，里边有的数据有重复，要求设计一个算法删除重复数据？要求尽量快。

先取模分成小文件，然后每个文件使用hash\_map或者trie树。

31、洗牌算法，如何证明算法是随机的

需要随机置乱的n个元素的数组array：

for( i =n-1;i>=1;i--)

(0 =< j <= i)

交换array[i]和array[j]

end

32、100万个32位整数，如何最快找到中位数。能保证每个数是唯一的，如何实现O(N)算法？

这道题是编程之美或编程珠玑上的。

这道题使用位图，需要空间复杂度是512Ｍ。

33、hashmap和treemap

34、linux rwxrwxrwx文件拷贝

35、版本控制，github指令

38、n个球要分成m堆每个堆不能为空，有多少种分发

39、红黑树实现

每次先按BST方法插入数据，每次都将插入节点变红，然后再自下而上调整节点颜色，使其满足红黑树性质

40、堆了解么？怎么删除元素？怎么插入元素？

删除：删除顶元素，将最后一个数据移到第一个，size自减1，然后从第一个节点自上而下进行整堆

41、图的遍历介绍一下

DFS、BFS

42、你看，现在我百度一个ip地址可以查到那个IP的实际省市显地址，现在我给你每个县/村的ip段，要求你实现一下输入查找功能。【大概是问数据库，我完全不会，瞎答了】

43.你看看我们阴阳师手游里面，可以摇绳子【手机给我展示了一下，真的是像绳子不是单摆】，现在我要你来实现一个绳子，要可以摇那种。

44.你看我打开一个谷歌网页，输入一些单词，他下面给我提示了一些可能我需要的选项，比如我输入一个tail出现了balabala。你现在给我实现一下这个功能，会用什么数据结构什么算法呢。

1）Trie是一颗存储多个字符串的树。相邻节点间的边代表一个字符，这样树的每条分支代表一则子串，而树的叶节点则代表完整的字符串。和普通树不同的地方是，相同的字符串前缀共享同一条分支。

2）hashmap统计: 先对这批海量数据预处理。具体方法是：维护一个Key为Query字串，Value为该Query出现次数的HashTable，即hash\_map(Query，Value)，每次读取一个Query，如果该字串不在Table中，那么加入该字串，并且将Value值设为1；如果该字串在Table中，那么将该字串的计数加一即可，最终在O(N)的时间复杂度内用Hash表完成了统计。

3）堆排序：借助堆这个数据结构，找出Top K，时间复杂度为N‘logK。即借助堆结构，我们可以在log量级的时间内查找和调整/移动。因此，维护一个K(该题目中是10)大小的小根堆，然后遍历300万的Query，分别和根元素进行对比。所以，我们最终的时间复杂度是：O（N） + N’ \* O（logK），（N为1000万，N’为300万）。

45、.你说你打过acm比赛，acm比赛主要是用算法数据结构吧，你说说你觉得你遇到过的最巧妙的一个数据结构或者算法题是什么。【我回答了rope，正好面试官没听过，然后给他瞎bb了一番】

46、给你11位电话号码，让你通过电话找名字

47、 给你多个ipv4的区间，每个区间属于一个城市，如0.0.0.1-1.1.1.1属于北京，给你一个ipv4地址，你要输出所属的城市，如1.1.1.0输出北京。区间有可能会重叠，如果询问的ipv4属于多个城市，则输出所有所属城市。

48、平面最近点对。

**计算机网络：**

1、TCP和UDP的区别

2、tcp的可靠性怎么保证（三次握手、四次挥手、确认序列号）

1）检验和

典型的有CRC校验法。在要传输的k比特数据D后添加(n-k)比特冗余位。

2）序列号

通过序列号，可以去重、超时重传、数据有序到达。

3）确认应答

可以确认ACK之前的数据肯定到达了，保证了可靠性。

4）超时重传

发送的数据有可能因为网络拥堵，没有及时到达，发送端没有收到确认，超过计时器，就会进行重发。

发送端有可能收到许多重复确认，累计到一定次数，TCP认为网络或者接收端出现异常，重新发送丢失的数据包。

5）连接管理

通过三次握手四次挥手，也可提高可靠性。

6）流量控制

接收端处理数据的速度是有限的，如果发送端发的太快，接收端缓冲区容易满，会造成丢包以及引起丢包重传。

7）拥塞控制

发送数据的时候，不能刚开始就发送大量的数据，所以在不清楚网络状况的情况下，不能贸然发送大量的数据，有可能加重网络负担，TCP会使用慢启动机制，探探路，所以刚开始的时候，将拥塞窗口设为1，以后是指数增长，当达到阈值的时候，按照线性增长，到达拥塞窗口的最大值后，拥塞窗口重回1。

3、TCP和UDP的区别？分别举例它们的上层协议？

TCP是基于连接的，可靠的，偏向于传输大量数据，速度慢，http，ftp，smtp，telnet使用了tcp；UDP是无连接的，不可靠的，偏向于传输少量数据，速度快，dns,tftp,rip,snmp，rtp,nfs等使用了udp。

4、TCP4层网络层次、3次握手

5、TCP和UDP的知识点。TCP讲三次握手和四次挥手。

（1）三次握手

第一次握手：主机A发送同步报文段（SYN）请求建立连接。

第二次握手：主机B听到连接请求，就将该连接放入内核等待队列当中，并向主机A发送针对SYN的确认ACK，同时主机B也发送自己的请求建立连接（SYN）。

第三次握手：主机A针对\*\*主机B\*\*SYN的确认应答ACK。

（2）四次挥手

第一次挥手：当主机A发送数据完毕后，发送FIN结束报文段。

第二次挥手：主机B收到FIN报文段后，向主机A发送一个确认序号ACK（为了防止在这段时间内，对方重传FIN报文段）。

第三次挥手：主机B准备关闭连接，向主机A发送一个FIN结束报文段。

第四次挥手：主机A收到FIN结束报文段后，进入TIME\_WAIT状态。并向主机B发送一个ACK表示连接彻底释放。（如果客户端的确认应答丢失，算上这个丢失报文的时间，再加上服务端重传FIN的时间（重传后客户端重新启动2MSL计时器），2MSL的时间足够使客户端收到重传的FIN报文段。所以客户端不能立即进入CLOSED状态。）

6、网络，dns、https和http，非对称加密和加密比较一下

**操作系统：**

1.     进程和线程的区别

1）进程是系统分配资源（CPU以外）的基本单位

2）线程是被系统独立调度的基本单位（CPU的分配单位）

2. 线程间怎么共享资源

线程直接读/写进程数据段（如全局变量）来通信

3、线程使用共享资源会出现什么问题？需要怎么做？

操作系统具有异步性，如果对共享资源的访问不加以约制，一些具有相互制约、相互合作的线程得到的最终结果可能是错的

设置临界区，对临界区资源设置信号量，进行同步和互斥

4、多线程同步方式

1）软件实现方法（皮特森算法（双标志和单标志）：

Pi: flag[i]=true; turn=j;  while(flag[j]&&turn==j); ）

2）硬件实现方法（中断屏蔽方法（屏蔽中断/关中断）、硬件指令方法（原子操作））

3）信号量（PV操作）

4）管程（解决临界区分散所带来的管理和控制问题，包括共享结构数据说明、一组操作、设置共享数据初始值语句）

5、如何进行线程同步？在Windows下举例？分用户模式下同步和内核模式下同步 讨论？

用户模式下的方法有：原子操作（例如一个单一的全局变量），临界区。

内核模式下的方法有：事件，信号量，互斥量。

6、同步机制应遵循的准则

1）空闲让进

2）忙则等待

3）有限等待

4）让权等待

7. 进程间通信有哪些算法or多进程通信方式

1）共享存储（PV操作对共享空间读写进行同步互斥）

2）消息传递（1直接通信方式，挂在接收进程的消息缓冲队列上2间接通信方式，即信箱）

3）管道通信（管道是连接读进程和写进程通信的共享文件，限制管道大小，缓冲区允许一边写入另一边读出，管道通信是半双工通信）

4）客户机-服务器系统（包括：套接字（socket），远程过程调用和远程方法调用）

8、进程间通信有哪几种方式？在特定环境（比如两个程序需要共享一个文本）下哪种效率最高？Windows下如何进行内存共享？

无名管道，有名管道，信号量，信号，高级管道，消息队列，共享内存，sokect等，共享内存的效率最高，因为它可以直接读写内存，而不需要任何的数据拷贝。windows下主要通过映射机制实现的。共享内存的方式原理就是将一份物理内存映射到不同进程各自的虚拟地址空间中，这样每个进程都可以读取同一份数据，从而实现进程通信。因为是通过内存操作实现通信，因此是一种最高效的数据交换方法。

9、进程和线程的区别、进程如何调度（扯了进程维护线程池，临界区、事务、信号量、信号）

1）先来先服务调度算法

2）短作业优先调度算法

3）优先级调度算法

4）高响应比优先调度算法（响应比=（等待时间+要求服务时间）/要求服务时间）

5）时间片轮转调度算法

6）多级反馈队列调度算法

10、 简单说一下进程间切换发生的事情

1）保存处理机上下文，包括程序计数器和其他寄存器

2）更新PCB信息

3）把PCB移入相应的队列，如就绪、阻塞队列

4）选择另一个进程执行，并更新其PCB

5）更新内存管理的数据结构

6）恢复处理机上下文

7、进程安全如何保证（扯了进程的数据同步和锁的实现

安全状态：能找到一个分配资源的序列让所有进程都顺利完成

11、进程在什么情况下会互锁

多个进程同时占有对方需要的资源而同时请求对方的资源,而它们在得到请求之前不会释放所占有的资源

（1）     系统资源的竞争

（2）     进程推进顺序非法（信号量使用不当，A等B的消息，B等A的消息）

12、互锁怎么解决

1）预防死锁（设置某些限制条件，破坏死锁四个必要条件之一）

2）避免死锁（动态分配资源过程中，用某种方法防止系统进入不安全状态）

3）死锁检测及解除（剥夺资源、撤销进程、进程回退）

13、线程死锁的几个条件是什么？

（1）互斥条件：指线程对所分配的资源进行排他性使用，即在一段时间内某资源只由一个线程占用。

（2）请求和保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。

（3）不可剥夺条件：进程已获得的资源，在未使用之前，不能强行剥夺。

（4）循环等待条件：指在发生死锁时，必然存在一个进程资源循环等待链，链中每一个进程已获得的资源同时被链中下一个进程所请求

14、给定两个线程，A，B两个锁，举个造成死锁的例子？

程序中使用多个互斥量时，如果允许一个线程一直占有第一个互斥量，并且在试图锁住第二个互斥量时处于阻塞状态，但是拥有第二个互斥量的线程也在试图锁住第一个互斥量，这时就会发生死锁。因为两个线程都在相互请求另一个线程拥有的资源，所以这两个线程都无法向前运行，于是就产生死锁。如果所有线程总是在对互斥量B加锁之前锁住互斥量A，那么使用这两个互斥量不会产生死锁

15、可能面试官了解到我是系统方向，所以让我解释一个Hello World程序从C到最终运行起来的过程。

基本上就是先编译，得到Symbol, 链接器Resolve Symbol, Printf属于动态链接库里面的内容 所以涉及到GOT和PLT表，然后操作系统开新的进程，Load二进制文件，将控制流跳到程序入口Main函数执行等等。面试官听到动态链接库，又问了我一些动态链接库的内容和进程的地址空间和 内存的映射(基本上就是回答新进程的地址空间只是映射了Kernel代码，不用创建新的，动态链接库也是映射过来可以执行balabala

16、程序有哪几种链接方式？分别说明区别？哪种效率高？如果一个动态库没有.lib和头文件，要怎么使用里面的函数？

1）静态链接：在程序运行之前，先将各个目标模块及它们所需的库函数，链接成一个完整的装配模块，以后不再拆开。我们把这种事先进行链接的方式称为静态链接方式。

2）装入时动态链接：将用户源程序编译后所得到的一组目标模块，在装入内存时，采用边装入边链接的链接方式。

3）运行时动态链接：这是指对某些目标模块的链接，是在程序执行中需要该目标模块时，才对它进行的链接。（便于修改和更新，便于实现对目标模块的共享）

第三种方式效率较高。还可以节省大量的内存空间。

17、cache的作用和实现机制，讲了LRU、FIFO和LEU，详细介绍了LRU的三种实现

Cache作用：调节CPU与主存读取速度不一致的矛盾

Cache实现机制：将Cache和主存都分成若干大小相等的块，Cache中存储主存中最活跃的若干块副本

页面调度算法：

最佳置换算法OPT:选择以后不用的页面

最近最久未使用LRU:选择最近最久未使用的页面（堆栈类算法，需要寄存器和栈的硬件支持）

最不经常使用LFU:将一段时间内访问次数最少的页面换出

先进先出FIFO:选择最先装入内存的页面（基于队列）

时钟置换算法CLOCK(NRU):选择最近未用的页面

改进的时钟算法：考虑页面修改问题

18、问了操作系统的调度，页表之类的问题，然后讲了讲缓存算法。（LRU要求讲一下怎么实现）

LRU的实现：

1）用一个数组来存储数据，给每一个数据项标记一个访问时间戳，每次插入新数据项的时候，先把数组中存在的数据项的时间戳自增，并将新数据项的时间戳置为0并插入到数组中。每次访问数组中的数据项的时候，将被访问的数据项的时间戳置为0。当数组空间已满时，将时间戳最大的数据项淘汰。

2）利用一个链表来实现，每次新插入数据的时候将新数据插到链表的头部；每次缓存命中（即数据被访问），则将数据移到链表头部；那么当链表满的时候，就将链表尾部的数据丢弃。

3）利用链表和hashmap。当需要插入新的数据项的时候，如果新数据项在链表中存在（一般称为命中），则把该节点移到链表头部，如果不存在，则新建一个节点，放到链表头部，若缓存满了，则把链表最后一个节点删除即可。在访问数据的时候，如果数据项在链表中存在，则把该节点移到链表头部，否则返回-1。这样一来在链表尾部的节点就是最近最久未访问的数据项。

对于第一种方法，需要不停地维护数据项的访问时间戳，另外，在插入数据、删除数据以及访问数据时，时间复杂度都是O(n)。对于第二种方法，链表在定位数据的时候时间复杂度为O(n)。所以在一般使用第三种方式来是实现LRU算法。

19、基址变址寻址

1）相对寻址：PC的内容加上指令格式中的形式地址A而形成操作数的有效地址

2）基址寻址：将CPU中基址寄存器（BR）的内容加上指令格式中的形式地址A，而形成操作数的有效地址（基址寄存器内容由操作系统确定，内容不变，形式地址可变，用于分配存储空间）

3）变址寻址：变址寄存器（IX）的内容加上指令格式中的形式地址A，而形成操作数的有效地址（变址寄存器内容可由用户改变，形式地址A不变，用于数组）

20、堆和栈的区别？

（1）管理方式不同。栈由操作系统自动分配释放，无需我们手动控制；堆的申请和释放工作由程序员控制，容易产生内存泄漏；

（2）空间大小不同。每个进程拥有的栈的大小要远远小于堆的大小。理论上，程序员可申请的堆大小为虚拟内存的大小，进程栈的大小64bits的Windows默认1MB

（3）生长方向不同。堆的生长方向向上，内存地址由低到高；栈的生长方向向下，内存地址由高到低。

（4）分配方式不同。堆都是动态分配的，没有静态分配的堆。栈有2种分配方式：静态分配和动态分配。静态分配是由操作系统完成的，比如局部变量的分配。动态分配由alloca函数进行分配，但是栈的动态分配和堆是不同的，他的动态分配是由操作系统进行释放，无需我们手工实现。

（5）分配效率不同。栈由操作系统自动分配，会在硬件层级对栈提供支持：分配专门的寄存器存放栈的地址，压栈出栈都有专门的指令执行，这就决定了栈的效率比较高。堆则是由C/C++提供的库函数或运算符来完成申请与管理，实现机制较为复杂，频繁的内存申请容易产生内存碎片。显然，堆的效率比栈要低得多。

（6）存放内容不同。栈存放的内容，函数返回地址、相关参数、局部变量和寄存器内容等。堆，一般情况堆顶使用一个字节的空间来存放堆的大小，而堆中具体存放内容是由程序员来填充的。

21、静态变量是存放在数据段或者BSS段，是不入栈的

22、mmu虚拟内存映射（MMU负责虚拟地址映射为物理地址）

虚拟内存是一些系统页文件，存放在磁盘上，每个系统页文件大小为4K，物理内存也被分页框，每个页框大小也为4K，这样虚拟页文件和物理内存页就可以对应，实际上虚拟内存就是用于物理内存的临时存放的磁盘空间。页文件就是内存页， 物理内存中每页叫物理页，磁盘上的页文件叫虚拟页，物理页+虚拟页就是系统所有使用的页文件的总和。

23、Windows提供了3种方法来进行内存管理：

1）虚拟内存，最适合用来管理大型对象或者结构数组；

2）内存映射文件，最适合用来管理大型数据流（通常来自文件）以及在单个计算机上运行多个进程之间共享数据；

3）内存堆栈，最适合用来管理大量的小对象。

Windows操纵内存可以分两个层面：物理内存和虚拟内存。，

24、虚拟内存组成部分：

1） 页表机制

2） 中断机构

3） 地址变换机构

4） 内存和外存

25、 epoll的文件描述符是否有拷贝过程？

26、 让你设计一个方式实现内核和用户态都没有拷贝 如何实现？

**项目相关**

1、 后来面试官问了我研究生阶段做的事情，问我有没有接触过网络编程，因为了解到我最近用python，就问我有没有用过socket或者select这两个库

2、 听完自我介绍后面试官就开始问我各种简历上的项目经历，首先是研究生期间做的虚拟化部分，我做了简单介绍。之后是大四的软件测试和嵌入式开发，大三的LeapMotion和Android App开发， 之前接触过的Kinect开发等等。了解完后就开始问我熟悉的编程语言(我说的Java & Pure C)。

了解到LeapMotion是做的有关一个游戏的内容后就开始问我详细内容，由于里面用到了交互设备和图形学的简单知识，问我了不了解OpenGL。(我说的不了解，但是说了一些概念)。 之后聊到 游戏开发引擎，我只说了我听说过Unity和Unreal.. 然后说的是我不了解，专业是系统方向。

3、 如果以前做过3D建模的项目，还会着重考察3D建模的知识。比如openGL、DirectX等。如果有Unity3D的经验就更好了。

4、 Redis底层的数据结构、跳表怎么实现的？hash表怎么实现的？怎么rehash?是不是rehash一定要全部复制？什么时候会触发rehash?redis持久化介绍一下，epoll和select介绍一下

5、 系统设计题1：网络分区的时候，a的朋友圈被b点赞了，怎么同步这个消息

6、 系统设计题2：实现一个高精度的定时器（最后提示了linux内核实现了一个，有空可以看看）

7、 简历上的项目是python写的web，然后问了一些python的知识，例如Django里面的AbstractUser是怎么对密码进行加密的(项目里用了这个)，然后还有mysql一些注入的问题。

**Misc（杂）**

1、 问了我关于骨骼动画的原理，因为这个我简历有写我在项目中提供做美术素材，这个部分应该跟每个小伙伴简历上具体内容相关。

2、 其中还问过我一些图形学的知识，不过我不是很了解，所以只是介绍了一下景深、3D里面摄像头balabala，把当时大三LeapMotion项目的内容说了一点点…

3、 你用过哪些设计模式？

工厂方法和抽象工厂有哪些区别？

工厂方法模式属于对象创建型模式，它定义一个用户创建对象的接口，让子类决定实例化哪一个类。工厂方法模式使一个类的实例化延迟到其子类。具体来说就是一个一个抽象产品类，派生出很多个具体产品类；同时，一个抽象工厂类，派生出多个具体工厂类。而每个具体工厂类只能创建一个具体产品类的实例。

抽象工厂模式也属于对象创建型模式，它提供了一个创建一系列相关或相互依赖对象的接口，而无须制定它们具体的类。具体来说就是在多个抽象产品类中，每个抽象产品类可以派生出多个具体产品类。一个抽象工厂类，可以派生出多个具体工厂类。每个具体工厂类可以创建多个具体产品类的实例。

区别：工厂方法模式只有一个抽象产品类，而抽象工厂模式有多个。工厂方法模式的具体工厂类只能创建一个具体产品类的实例，而抽象工厂模式可以创建多个。

工厂方法：说白了就是一个方法，这个方法是创建具体的产品的，它要求所有的工厂都具有同一个签名的方法，必要时重写该方法；

抽象工厂：不能直接创建产品，只能创建工厂，即抽象工厂创建的产品是工厂。

**提问**

最后，面试官让我问问题，我大概问了两个：

1. 游戏行业有没有程序与美术的双栖人才？

2. 实习做什么样的工作？（要看分到哪两个组里）

3、网易研发和策划之间的协作和不同，研发是有自己的考核机制，也有与策划的反馈机制，所以可以影响游戏的设计，不过影响不大。

4、问我这种方向差异比较大的要怎么跟进，面试官也是介绍了那边的培训制度，不用担心

**腾讯：**

**C++方面的问题：**

1、 然后问了struct和class的区别

（当时说了默认public和private的区别）但是面试官不是很满意，可能应该有更深层次的说法，原谅我底层懂的真的不是很多。

C#中：class 是引用类型，structs是值类型，当你实例化一个class，它将创建在堆上。而你实例化一个struct，它将创建在栈上

2、 const的作用（只记得定义常量，定义常量指针，常量成员函数）

const定义时赋初值，以后只能使用，不能修改

3、指针和引用的区别

4、函数参数传递

1）值传递（passl-by-value）过程中，被调函数的形式参数作为被调函数的局部变量处理，即在堆栈中开辟了内存空间以存放由主调函数放进来的实参的值，从而成为了实参的一个副本。值传递的特点是被调函数对形式参数的任何操作都是作为局部变量进行，不会影响主调函数的实参变量的值。

2）引用传递(pass-by-reference)过程中，被调函数的形式参数虽然也作为局部变量在堆栈中开辟了内存空间，但是这时存放的是由主调函数放进来的实参变量的地址。被调函数对形参的任何操作都被处理成间接寻址，即通过堆栈中存放的地址访问主调函数中的实参变量。正因为如此，被调函数对形参做的任何操作都影响了主调函数中的实参变量

5、c++的多态性。

6、问了对inline的理解。（C++）

解决一些频繁调用的小函数对栈内存重复开辟所带来的消耗

7、能否将构造函数定义为虚函数？

8、问了STL中map是利用何种结构和方法实现的。

红黑树

8、 C++的三大特性？

封闭性、继承、多态

10、多态是什么实现的？

11、指针和引用的区别？

12、const有什么用法？

13、内存泄漏、数组越界问题、指针非法访问问题？智能指针

14、C++初始化列表使用

15、stl容器可以放入智能指针吗？auto\_ptr不可以，其余的可以

16、c++的多态，多态的实现，还有STL库的一些实现

17、C++的构造函数和析构函数的执行顺序？

18、结构体和类有什么区别？

19、堆、栈一般放哪些数据？

**数据结构和算法：**

1、有100盏灯，从1~100编号，开始灯的状态是亮的，然后按照1的倍数，2的倍数，3的倍数。。。一直到100的倍数翻转，问你最后熄灭的是哪几盏灯。当时应该仔细想好再写代码的，一开始思路略微麻烦了一些，其实类似素数筛那样走一遍就可以了，大概nlogn。

2、然后还问了一个第k大的问题

3、问了在2D的环境下，有极大数量的物体（假设为小球），如何优化使得系统开销和性能最好？

4、假如有100个小球有碰撞的检测（其中有一个为用户），如果需要统计小球的碰撞的次数总和，如果减少性能要求的前提下进行实现？

5、假如有100个小球有碰撞的检测（每一个都是用户），如果需要统计小球的碰撞的次数总和，如果减少性能要求的前提下进行实现？

6、问了一下我简历上的项目，看到我用了快排，于是问我这样的算法快排是稳定的吗？本来准备了快排的概念、实现方法，结果对方说：我不想问这种东西，因为太基础了，每个程序员都应该熟练掌握的。

7、你知道哪些设计模式？常用到的模式有哪些？单例模式+适配器模式

8、像红黑树、二叉树啥的。我是连STL库一块问的，因为STL库的实现本身就用了很多数据结构；

9、谈谈对于数据结构的认识（因为楼主都快把数据结构背下来了，洋洋洒洒说了半个小时）

10、b树b+树b\*树的区别（b+树对于叶子结点有一个双向链表，b\*树+非叶子结点）

11、动态规划（最短路径算法）

12、谈谈排序和堆排序的应用

13、AVL树和红黑树区别（一个通过旋转实现完全平衡，一个通过定义根结点的颜色）

**计算机网络：**

**操作系统：**

1、 问了线程和进程的区别

2、 问了对sleep的理解。（进程中的概念）

3、 进程和线程的区别？

4、 进程间通讯有几中方式？

5、 死锁是什么？

6、 进程间通讯方式

7、 进程线程协程的区别

8、进程和线程的区别？

**项目相关**

1、问了实现工程的代码量是越多越好还是越少越好？

2、 开发遇到的问题：那些比较简单，哪些比较需要花时间解决？

3、聊了聊做的项目的，谈了谈怎么设计的。

**Misc（杂）**

1、 问了shader。

2、 MVC架构有什么了解？

3、 前端有什么了解？

4、 有木有做过GUI的开发？

5、 对于游戏开发有什么兴趣？

6、 对于后台开发的认识？

7、 对于AI这一块有什么兴趣？

8、 对于信息安全计算有什么了解？

9、 关于游戏和Unity的内容

10、为什么ngui在VR里不能使用

11、NGUI Altas原理

12、用c#实现一个商城的业务逻辑（因为刚做过，还有印象）

13、在计算机图形学中如何进行平移、旋转、缩放？

14、有哪些光照模型？

15、如何判断空间中的一点在一个球体内？

16、Unity3D脚本的生命周期？

17、Unity3D的内存管理？

18、Unity3D的UI如何渲染？

19、面向组件的好处？

20、用Opengl如何画一个点，一条直线？

21、有哪些设计模式？

**提问**

网易面经

2019年8月21日

9:08

作者：greenpill

链接：<https://www.nowcoder.com/discuss/168189?type=post&order=time&pos=&page=1>

来源：牛客网

#3.15网易互娱面试

##自我介绍

##游戏

**1. 你觉得你参与游戏研发有什么优势？**

**基础知识**

**1. 讲一下C++的三大特性封装、继承和多态**

封装可以使得代码模块化，继承可以扩展已存在的代码，他们的目的都是为了代码重用。而多态的目的则是为了接口重用

* 封装：封装是在设计类的一个基本原理，是将抽象得到的数据和行为（或功能）相结合，形成一个有机的整体，也就是将数据与对数据进行的操作进行有机的结合，形成“类”，其中数据和函数都是类的成员。
* 继承：如果一个类别B“继承自”另一个类别A，就把这个B称为“A的子类”，而把A称为“B的父类别”也可以称“A是B的超类”。继承可以使得子类具有父类别的各种属性和方法，而不需要再次编写相同的代码。在令子类别继承父类别的同时，可以重新定义某些属性，并重写某些方法，即覆盖父类别的原有属性和方法，使其获得与父类别不同的功能。
  1. 访问权限
     + public： 父类对象内部、父类对象外部、子类对象内部、子类对象外部都可以访问。
     + protected:父类对象内部、子类对象内部可以访问，父类对象外部、子类对象外部都不可访问。
     + private：父类对象内部可以访问，其他都不可以访问。  
       |访问对象|public|protected|private|  
       |-|-|-|-|  
       |父类|可见|可见|可见|  
       |子类|可见|可见|不可见|  
       |父类外部|可见|不可见|不可见|  
       |子类外部|可见|不可见|不可见|
  2. 继承方式  
     ps.三种继承方式不影响子类对父类的访问权限，子类对父类只看父类的访问控制权。继承方式是为了控制子类(也称派生类)的调用方(也叫用户)对父类(也称基类)的访问权限。public、protected、private三种继承方式，相当于把父类的public访问权限在子类中变成了对应的权限。 如protected继承，把父类中的public成员在本类中变成了protected的访问控制权限；private继承，把父类的public成员和protected成员在本类中变成了private访问控制权。  
     ps.友元是类级别的，不存在继承的问题。
* 多态：多态性可以简单地概括为“一个接口，多种方法”，程序在运行时才决定调用的函数，它是面向对象编程领域的核心概念。多态(polymorphism)，字面意思多种形状。
  1. 静态多态：静态多态也称为静态绑定或早绑定。编译器在编译期间完成的，编译器根据函数实参的类型(可能会进行隐式类型转换)，可推断出要调用那个函数，如果有对应的函数就调用该函数，否则出现编译错误。
     + 函数重载  
       编译器根据函数不同的参数表，对同名函数的名称做修饰，然后这些同名函数就成了不同的函数（至少对于编译器来说是这样的）。函数的调用，在编译器间就已经确定了，是静态的。也就是说，它们的地址在编译期就绑定了（早绑定）。
     + 泛型编程  
       泛型编程就是指编写独立于特定类型的代码，泛型在C++中的主要实现为模板函数和模板类。  
       泛型的特性：  
        1. 函数模板并不是真正的函数，它只是C++编译生成具体函数的一个模子。  
        1. 函数模板本身并不生成函数，实际生成的函数是替换函数模板的那个函数，比如上例中的add(sum1,sum2)，这种替换是编译期就绑定的。  
        3. 函数模板不是只编译一份满足多重需要，而是为每一种替换它的函数编译一份。  
        4. 函数模板不允许自动类型转换。  
        5. 函数模板不可以设置默认模板实参。比如template <typename T=0>不可以。
  2. 动态多态  
     c++的动态多态是基于虚函数的。对于相关的对象类型，确定它们之间的一个共同功能集，然后在基类中，把这些共同的功能声明为多个公共的虚函数接口。各个子类重写这些虚函数，以完成具体的功能。客户端的代码（操作函数）通过指向基类的引用或指针来操作这些对象，对虚函数的调用会自动绑定到实际提供的子类对象上去。
  3. 宏多态（？）  
     带变量的宏可以实现一种初级形式的静态多态：  
     #include <iostream>  
     #include <string>  
       
     // 定义泛化记号：宏ADD  
     #define ADD(A, B) (A) + (B);  
       
     int main()  
     {  
      int i1(1), i2(2);  
      std::string s1("Hello, "), s2("world!");  
      int i = ADD(i1, i2); // 两个整数相加  
      std::string s = ADD(s1, s2); // 两个字符串“相加”  
      std::cout << "i = " << i << "/n";  
      std::cout << "s = " << s << "/n";  
     }
  4. 动态多态和静态多态的比较
     + 静态多态
       - 优点：
         1. 由于静多态是在编译期完成的，因此效率较高，编译器也可以进行优化；
         2. 有很强的适配性和松耦合性，比如可以通过偏特化、全特化来处理特殊类型；
         3. 最重要一点是静态多态通过模板编程为C++带来了泛型设计的概念，比如强大的STL库。
       - 缺点：
         1. 由于是模板来实现静态多态，因此模板的不足也就是静多态的劣势，比如调试困难、编译耗时、代码膨胀、编译器支持的兼容性不能够处理异质对象集合
     + 动态多态
       - 优点：
         1. OO设计，对是客观世界的直觉认识；
         2. 实现与接口分离，可复用
         3. 处理同一继承体系下异质对象集合的强大威力
       - 缺点：
         1. 运行期绑定，导致一定程度的运行时开销；
         2. 编译器无法对虚函数进行优化
         3. 笨重的类继承体系，对接口的修改影响整个类层次；
     + 不同点：
       - 本质不同，早晚绑定。静态多态在编译期决定，由模板具现完成，而动态多态在运行期决定，由继承、虚函数实现；
       - 动态多态中接口是显式的，以函数签名为中心，多态通过虚函数在运行期实现，静态多台中接口是隐式的，以有效表达式为中心，多态通过模板具现在编译期完成
     + 相同点：
       - 都能够实现多态性，静态多态/编译期多态、动态多态/运行期多态；
       - 都能够使接口和实现相分离，一个是模板定义接口，类型参数定义实现，一个是基类虚函数定义接口，继承类负责实现；

**2. 虚函数、纯虚函数**

- 概述

它虚就虚在所谓“推迟联编”或者“动态联编”上，一个类函数的调用并不是在编译时刻被确定的，而是在运行时刻被确定的。由于编写代码的时候并不能确定被调用的是基类的函数还是哪个派生类的函数，所以被成为“虚”函数。

虚函数只能借助于指针或者引用来达到多态的效果。常用的方式是父类指针指向子类对象。当有多个对于父类的继承时，可以统一用父类指针来表示各子类对象，但是事实上所指向的对象具体是哪一个，或者说所调用的函数是哪一个子类的对象的函数需要在运行时才知道。这就实现了多态。

```c  
 class A  
 {  
 public:  
 virtual void foo()  
 {  
 cout<<"A::foo() is called"<<endl;  
 }  
 };  
 class B:public A  
 {  
 public:  
 void foo()  
 {  
 cout<<"B::foo() is called"<<endl;  
 }  
 };  
 int main(void)  
 {  
 A \*a = new B();  
 a->foo(); // 在这里，a虽然是指向A的指针，但是被调用的函数(foo)却是B的!  
 return 0;  
 }  
 ```

- 语法

```c  
 virtual void fun1()=0 //纯虚函数  
 virtual void fun1() // 虚函数  
 ```

- 纯虚函数  
 纯虚函数是在基类中声明的虚函数，它在基类中没有定义，但要求任何派生类都要定义自己的实现方法。含有虚函数的类成为抽象类，不能生成对象。  
- 虚函数表  
 虚表是属于类的，而不是属于某个具体的对象，`一个类只需要一个虚表即可` 。同一个类的所有对象都使用同一个虚表。为了指定对象的虚表，对象内部包含一个虚表的指针，来指向自己所使用的虚表。为了让每个包含虚表的类的对象都拥有一个虚表指针，编译器在类中添加了一个指针，\*\_\_vptr，用来指向虚表。这样，当类的对象在创建时便拥有了这个指针，且这个指针的值会自动被设置为指向类的虚表。  
 C++的编译器应该是保证虚函数表的指针存在于对象实例中最前面的位置（这是为了保证取到虚函数表的有最高的性能——如果有多层继承或是多重继承的情况下）。

**3. 引用和指针的区别**

指针-对于一个类型T，T\* 就是指向T的指针类型，也即一个T\*类型的变量能够保存一个T对象的地址，而类型T是可以加一些限定词的，如const、volatile等等。

引用-引用是一个对象的别名，主要用于函数参数和返回值类型，符号X&表示X类型的引用。

* 不同点：  
  本质： 指针指向一块内存，它的内容是所指内存的地址；而引用则是某块内存的别名，引用不改变指向。
  1. 引用不可以为空，但指针可以为空。定义一个引用的时候，必须初始化。因此使用指针之前必须做判空操作，而引用就不必。
  2. 引用不可以改变指向，对一个对象"至死不渝"；但是指针可以改变指向，而指向其它对象。
  3. 引用的大小是所指向的变量的大小，因为引用只是一个别名而已；指针是指针本身的大小，4个字节（32位）。
  4. const int\* p-> 指向常量的指针，int \* const p->本身是常量的指针.后者需要在定义的时候初始化。对于引用来讲int const & p=i;和 const int &p=i;没什么区别，都是指指向的对象是常量。
  5. 引用和指针的++自增运算符意义不同，指针的++表示的地址的变化，一般是向下4个字节的大小（一个只针的大小），引用的++就是对应元素的++操作。
  6. 指针传递和引用传递
     + 指针传递参数本质上是值传递的方式，它所传递的是一个地址值。值传递过程中，被调函数的形式参数作为被调函数的局部变量处理，即在栈中开辟了内存空间以存放由主调函数放进来的实参的值，从而成为了实参的一个副本。值传递的特点是被调函数对形式参数的任何操作都是作为局部变量进行，不会影响主调函数的实参变量的值。
     + 引用传递过程中，被调函数的形式参数也作为局部变量在栈中开辟了内存空间，但是这时存放的是由主调函数放进来的实参变量的地址。被调函数对形参的任何操作都被处理成间接寻址，即通过栈中存放的地址访问主调函数中的实参变量。正因为如此，被调函数对形参做的任何操作都影响了主调函数中的实参变量。

**4. 深拷贝和浅拷贝（值拷贝和位拷贝）**

对于build-in类型来说，复制是简单的，都是开辟新的内存，将对应的值放入新开辟的位置即可，之后他们不再有关系（指针类型就是地址的复制，和指向的地方没关系）。

对于类而言，如果存在指针和引用类型，（const类型？），需要考虑自己来写拷贝构造函数和重载=操作符。

* 浅拷贝（位拷贝）  
  指将一个对象的内存映像按位原封不动的复制给另一个对象，所谓值拷贝就是指，将原对象的值复制一份给新对象。 在用"bitwise assignment"时会直接将对象的内存映像复制给另一个对象，这样两个对象会指向同一个内存区域，当一个对象被释放后，另一个对象的指针会成为野指针（悬垂指针）。这时，就应该编写operator=和copy constructor来实现值拷贝 。  
  默认的拷贝构造函数”和“缺省的赋值函数”均采用“位拷贝”而非“值拷贝”的方式来实现，倘若类中含有指针变量，这两个函数注定将出错。
* 深拷贝（值拷贝）  
  在“深拷贝”的情况下，对于对象中动态成员，就不能仅仅简单地赋值了，而应该重新动态分配空间。
* 产生复制的场景：
  1. 建立一个新对象，并用另一个同类的已有对象对新对象进行初始化
  2. 当函数的参数为类的对象时，这时调用此函数时使用的是值传递，也会产生对象的复制
  3. 函数的返回值是类的对象时，在函数调用结束时，需要将函数中的对象复制一个临时对象并传给改函数的调用处  
     这里通过查资料存疑， 默认拷贝构造函数基本工作原理：对所有的非POD类型成员变量执行拷贝构造，POD类型成员变量则进行位拷贝，当成员变量都是POD类型的时候，编译器也许不会产生拷贝构造函数，只是对对象进行位拷贝。 默认operator=基本工资原理：对所有的非POD类型成员变量执行operator=操作，POD类型成员变量则进行位拷贝，当成员变量都是POD类型的时候，编译器也许不会产生operator=，只是对对象进行位拷贝。 从上面两个工作原理看到： 1.当class成员变量是有const修饰、引用类型、不能够提供operator=操作(例如boost::nocopyable)时候就不会产生operator=，我们对这个class对象进行赋值操作的时候就是编译出错。 2.当class成员变量是不能够提供拷贝构造操作(例如boost::nocopyable)时候就不会产生拷贝构造函数，我们对这个class对象进行拷贝构造操作的时候就是编译出错。 以前(2005年前？)c++各种参考文献对构造、析构和拷贝关注较多的是“深拷贝”和“浅拷贝”问题，现在的c++可能考虑更多的是如何保证不被拷贝^\_^(个人见解)。

**5. python用过吗？**

**6. TCP和UDP的区别及应用场景**

* 区别
  1. 面向连接VS无连接  
     TCP建立一个连接需要3次握手IP数据包，断开连接需要4次握手。另外断开连接时发起方可能进入TIME\_WAIT状态长达数分钟（视系统设置，windows一般为120秒），在此状态下连接（端口）无法被释放。  
     UDP不需要建立连接，可以直接发起。
  2. 可靠VS不可靠  
     TCP利用握手、ACK和重传机制，udp没有。  
     1，校验和（校验数据是否损坏）；  
     2，定时器（分组丢失则重传）；  
     3，序列号（用于检测丢失的分组和重复的分组）；  
     4，确认应答ACK（接收方告知发送方正确接收分组以及期望的下一个分组）；  
     5，否定确认（接收方通知发送方未被正确接收的分组）；  
     6，窗口和流水线（用于增加信道的吞吐量）。（窗口大小：无需等待确认应答而可以继续发送数据的最大值）
  3. 有序性  
     TCP利用seq序列号对包进行排序，udp没有。
  4. 面向字节流vs面向报文
     + 面向报文  
       面向报文的传输方式是应用层交给UDP多长的报文，UDP就照样发送，即一次发送一个报文。因此，应用程序必须选择合适大小的报文。若报文太长，则IP层需要分片。UDP对应用层交下来的报文，既不合并，也不拆分，而是保留这些报文的边界。这也就是说，应用层交给UDP多长的报文，UDP就照样发送，即一次发送一个报文。（一个upd的最大报文长度2^16-1-20-8,20是ip报文头，8是udp报文头）
     + 面向字节流  
       面向字节流的话，虽然应用程序和TCP的交互是一次一个数据块（大小不等），但TCP把应用程序看成是一连串的无结构的字节流。TCP有一个缓冲，当应用程序传送的数据块太长，TCP就可以把它划分短一些再传送。如果应用程序一次只发送一个字节，TCP也可以等待积累有足够多的字节后再构成报文段发送出去。
  5. tcp有流量控制，udp没有
  6. tcp的头部比20bytes，udp8byres
* TCP应用场景：  
  效率要求相对低，但对准确性要求相对高的场景。因为传输中需要对数据确认、重发、排序等操作，相比之下效率没有UDP高。举几个例子：文件传输（准确高要求高、但是速度可以相对慢）、接受邮件、远程登录。
* UDP应用场景：  
  效率要求相对高，对准确性要求相对低的场景。举几个例子：QQ聊天、在线视频、网络语音电话（即时通讯，速度要求高，但是出现偶尔断续不是太大问题，并且此处完全不可以使用重发机制）、广播通信（广播、多播）。

**7. TCP连接的建立和断开**

**8. 死锁的条件及解决的办法**

* 死锁的条件  
  必须同时存在以下的四个条件才能发生死锁。
  1. 互斥条件  
     即某个资源在一段时间内只能由一个进程占有，不能同时被两个或两个以上的进程占有。这种独占资源如CD-ROM驱动器，打印机等等，必须在占有该资源的进程主动释放它之后，其它进程才能占有该资源。这是由资源本身的属性所决定的。
  2. 不可抢占条件。  
     进程所获得的资源在未使用完毕之前，资源申请者不能强行地从资源占有者手中夺取资源，而只能由该资源的占有者进程自行释放。
  3. 占有且申请条件。  
     进程至少已经占有一个资源，但又申请新的资源；由于该资源已被另外进程占有，此时该进程阻塞；但是，它在等待新资源之时，仍继续占用已占有的资源。
  4. 循环等待条件  
     存在一个进程等待序列{P1，P2，...，Pn}，其中P1等待P2所占有的某一资源，P2等待P3所占有的某一源，......，而Pn等待P1所占有的的某一资源，形成一个进程循环等待环。
* 死锁的预防  
  死锁的预防是保证系统不进入死锁状态的一种策略。

1. 破坏互斥条件  
   有些资源不能被共享。--没用  
   2. 破坏不可抢占条件。  
   可抢占式，即要求申请失败的进程释放自己占有的资源给别人用，降低系统性能。  
   3. 破坏占有且申请条件。  
   直接申请自己所需要的所有资源。--1.不可预知自己需要什么资源 2.资源利用率低，长期占有自己可能不用的资源。  
   4. 破坏循环等待条件  
   资源分类、编号，按序申请。 --·1.编号可能是困难的，维护相应的序列是困难的

* 死锁的避免  
  死锁的避免指的是不限制进程有关申请资源的命令，而是对进程所发出的每一个申请资源命令加以动态地检查，并根据检查结果决定是否进行资源分配。  
  银行家算法。当一个进程申请使用资源的时候，银行家算法通过先 试探 分配给该进程资源，然后通过安全性算法判断分配后的系统是否处于安全状态，若不安全则试探分配作废，让该进程继续等待。

判定安全状态需要已分配资源、还需要的资源、可用资源、finish判定符

**9. 复杂度为O(nlogn)的排序算法有哪些？是否稳定？选一个讲一下**

1. 快速排序
2. 归并排序
3. 堆排序

**10. 介绍一下哈希表？哈希函数中如果碰到冲突如何解决？**

哈希表（Hash Table，也叫散列表），是根据关键码值 (Key-Value) 而直接进行访问的数据结构。也就是说，它通过把关键码值映射到表中一个位置来访问记录，以加快查找的速度。哈希表的实现主要需要解决两个问题，哈希函数和冲突解决。

* 哈希函数  
  哈希函数也叫散列函数，它对不同的输出值得到一个固定长度的消息摘要。理想的哈希函数对于不同的输入应该产生不同的结构，同时散列结果应当具有同一性（输出值尽量均匀）和雪崩效应（微小的输入值变化使得输出值发生巨大的变化）。
* 冲突解决  
  链地址法的基本思想是，为每个 Hash 值建立一个单链表，当发生冲突时，将记录插入到链表中。

**11. 设计模式了解吗？**

SOLID

S（单一功能原则Single responsibility principle）规定每个类都应该有一个单一的功能，并且该功能应该由这个类完全封装起来。所有它的（这个类的）服务都应该严密的和该功能平行（功能平行，意味着没有依赖）。

O（开闭原则）软件中的对象（类，模块，函数等等）应该对于扩展是开放的，但是对于修改是封闭的。

L（里氏替换原则Liskov Substitution principle）派生类必须能完全取代其基类

I（接口隔离原则interface-segregation principle） 客户不依赖于他不使用的方法。

D(控制反转原则Inversion of Control)高层级的模块不应该依赖于低层级的模块，他们应该都依赖于抽象。细节依赖于抽象，而不是抽象依赖于细节。

依赖注入：Class A中用到了Class B的对象b，一般情况下，需要在A的代码中显式的new一个B的对象。

采用依赖注入技术之后，A的代码只需要定义一个私有的B对象，不需要直接new来获得这个对象，而是通过相关的容器控制程序来将B对象在外部new出来并注入到A类里的引用中。而具体获取的方法、对象被获取时的状态由配置文件（如XML）来指定。

**二叉树的先序遍历、中序遍历、后续遍历**

**题**

**1. 如何判断链表是否有环**

快慢指针：定义两个指针，同时从链表的头节点出发，一个指针一次走一步，另一个指针一次走两步。如果走得快的指针追上了走得慢的指针，那么链表就是环形链表；如果走得快的指针走到了链表的末尾（next指向 NULL）都没有追上第一个指针，那么链表就不是环形链表。

map：如果不考虑空间复杂度，可以使用一个map记录走过的节点，当遇到第一个在map中存在的节点时，就说明回到了出发点，即链表有环，同时也找到了环的入口。

所以寻找环入口的方法就是采用两个指针，一个从表头出发，一个从相遇点出发，一次都只移动一步，当二者相等时便是环入口的位置

**2. 如何判断链表是否相交**

我们可以从头遍历两个链表。创建两个栈，第一个栈存储第一个链表的节点，第二个栈存储第二个链表的节点。每遍历到一个节点时，就将该节点入栈。两个链表都入栈结束后。则通过top判断栈顶的节点是否相等即可判断两个单链表是否相交。因为我们知道，若两个链表相交，则从第一个相交节点开始，后面的节点都相交。

若两链表相交，则循环出栈，直到遇到两个出栈的节点不相同，则这个节点的后一个节点就是第一个相交的节点。

* 遍历链表记录长度。  
  同时遍历两个链表到尾部，同时记录两个链表的长度。若两个链表最后的一个节点相同，则两个链表相交。  
  有两个链表的长度后，我们就可以知道哪个链表长，设较长的链表长度为len1,短的链表长度为len2。  
  则先让较长的链表向后移动(len1-len2)个长度。然后开始从当前位置同时遍历两个链表，当遍历到的链表的节点相同时，则这个节点就是第一个相交的节点。

**如何用两个栈实现一个队列**

**二叉树的广度优先搜索**

**二叉树的深度优先搜索**

**一个整数数组，一个整数n，如何判断数组中是否存在两个数的和等于n**

**3. 有m个玩家，取游戏排行榜中排名前k的玩家进行决斗，如何选取**

快排：只排

容量为k最大堆

1. 游戏的帧率为60帧，每秒更新六十次，每次都要上传玩家数据，假设上传一次需要100ms，但是一次上传最多16ms，如何解决
2. 开服创建账号时，需要对玩家在数据库中的ID进行分配（数据库只有一个，但玩家会被分配到多个物理机中进行处理），每个新玩家创建都需要去数据库中查找，如何减少查找数据库的次数（提示，一台物理机向数据库请求一千条ID，下一台请求下一千条ID，当一千条分配完后再向数据库请求）
3. 玩家战斗结束后根据积分更新排行榜，在排行榜太大更新速度慢的情况下如何实时更新玩家排名（提示：可以区分粒度）
4. 阴阳师的进度条如何实现？当一个式神的回合结束后进度条如何变化？有没有影响进度条的情况？

有什么想问的

1. 实习进去之后会接触到什么技术？  
   主要是使用python语言，引擎可能会使用C++，没有python基础不影响，但是最好先自学
2. 如何知道自己是否通过面试？  
   等待通知，五到十个工作日之内，目前只面了几个，还不能确定

我的凉了的和没凉的面试记录地址（持续更新），https://github.com/darkmoon233/GreenPills-CSNotes