1. malloc和new的区别

1 [malloc](http://zhidao.baidu.com/search?word=malloc&fr=qb_search_exp&ie=utf8" \t "_blank)与free是C++/C语言的标准[库函数](http://zhidao.baidu.com/search?word=%E5%BA%93%E5%87%BD%E6%95%B0&fr=qb_search_exp&ie=utf8" \t "_blank)，new/delete是C++的[运算符](http://zhidao.baidu.com/search?word=%E8%BF%90%E7%AE%97%E7%AC%A6&fr=qb_search_exp&ie=utf8" \t "_blank)。它们都可用于申请[动态内存](http://zhidao.baidu.com/search?word=%E5%8A%A8%E6%80%81%E5%86%85%E5%AD%98&fr=qb_search_exp&ie=utf8" \t "_blank)和释放内存。  
2 对于非内部[数据类型](http://zhidao.baidu.com/search?word=%E6%95%B0%E6%8D%AE%E7%B1%BB%E5%9E%8B&fr=qb_search_exp&ie=utf8" \t "_blank)的对象而言，光用maloc/free无法满足动态对象的要求。对象在创建的同时要自动执行[构造函数](http://zhidao.baidu.com/search?word=%E6%9E%84%E9%80%A0%E5%87%BD%E6%95%B0&fr=qb_search_exp&ie=utf8" \t "_blank)，对象在消亡之前要自动执行[析构函数](http://zhidao.baidu.com/search?word=%E6%9E%90%E6%9E%84%E5%87%BD%E6%95%B0&fr=qb_search_exp&ie=utf8" \t "_blank)。由于[malloc](http://zhidao.baidu.com/search?word=malloc&fr=qb_search_exp&ie=utf8" \t "_blank)/free是[库函数](http://zhidao.baidu.com/search?word=%E5%BA%93%E5%87%BD%E6%95%B0&fr=qb_search_exp&ie=utf8" \t "_blank)而不是[运算符](http://zhidao.baidu.com/search?word=%E8%BF%90%E7%AE%97%E7%AC%A6&fr=qb_search_exp&ie=utf8" \t "_blank)，不在编译器控制权限之内，不能够把执行[构造函数](http://zhidao.baidu.com/search?word=%E6%9E%84%E9%80%A0%E5%87%BD%E6%95%B0&fr=qb_search_exp&ie=utf8" \t "_blank)和[析构函数](http://zhidao.baidu.com/search?word=%E6%9E%90%E6%9E%84%E5%87%BD%E6%95%B0&fr=qb_search_exp&ie=utf8" \t "_blank)的任务强加于[malloc](http://zhidao.baidu.com/search?word=malloc&fr=qb_search_exp&ie=utf8" \t "_blank)/free。   
3 因此C++语言需要一个能完成[动态内存](http://zhidao.baidu.com/search?word=%E5%8A%A8%E6%80%81%E5%86%85%E5%AD%98&fr=qb_search_exp&ie=utf8" \t "_blank)分配和初始化工作的[运算符](http://zhidao.baidu.com/search?word=%E8%BF%90%E7%AE%97%E7%AC%A6&fr=qb_search_exp&ie=utf8" \t "_blank)new，以一个能完成清理与释放内存工作的运算符delete。注意new/delete不是[库函数](http://zhidao.baidu.com/search?word=%E5%BA%93%E5%87%BD%E6%95%B0&fr=qb_search_exp&ie=utf8" \t "_blank)。  
4 C++程序经常要调用C函数，而C程序只能用malloc/free管理[动态内存](http://zhidao.baidu.com/search?word=%E5%8A%A8%E6%80%81%E5%86%85%E5%AD%98&fr=qb_search_exp&ie=utf8" \t "_blank)。  
5 new可以认为是malloc加[构造函数](http://zhidao.baidu.com/search?word=%E6%9E%84%E9%80%A0%E5%87%BD%E6%95%B0&fr=qb_search_exp&ie=utf8" \t "_blank)的执行。new出来的指针是直接带类型信息的。而malloc返回的都是void指针。

1. Java和C、C++的区别

1. Java没有预处理指令。（如C中的#define , #include , #ifdef等）。C中的常量定义在Java中用static final来取代。Java中没有C中的全局变量。

2. Java[跨平台](http://zhidao.baidu.com/search?word=%E8%B7%A8%E5%B9%B3%E5%8F%B0&fr=qb_search_exp&ie=utf8" \t "_blank)，既不同的操作系统都可以通过JVM来解释Java程序，而C、C++、C#则与平台相关的，有些指令只在某些操作系统中才能执行。具体原因是，Java是解释型语言，所有代码都会翻译成统一的、与系统无关的bytecode，然后放到JVM上运行；而C是编译式语言，代码要先通过编译器转换成与系统相关的中间代码，然后才能运行。

3. Java中没有了指针，它使用了类似的句柄来取代指针，但是Java中不允许对句柄进行加减，没有取地址操作符之类的东东。

4. Java无法对内存进行控制，而C则可以通过指针对内存进行人为分配。  
还有一点，Java有Gabbage Collection机制，可以自动回收不再使用的空间，而C则要用free()函数释放空间。

5. Java没有goto语句。Java在C提供的控制语句基础上增加了异常处理和标签break和continue语句。这些可以替代goto的作用。

6. C要求一个方法或块中使用的所有局部变量的定义在该方法或块的最开始处定义，而Java允许这些定义在方法或块的任意地方出现。Java不要求在调用一个函数以前已经定义了该函数，可以在调用点后面定义。而C有这个要求。

7. Java不支持C中的strut 和 union类型。Java支持方法重载。c++支持多重继承，这是c++的一个特征，它允许多父类派生一个类。尽管多重继承功能很强，但使用复杂，而且会引起许多麻烦，编译程序实现它也很不容易。Java不支持多重继承，但允许一个类继承多个接口(界面)，实现了c++多重继承的功能，又避免了c++的许多缺陷。

8. Java不支持C中的enum关键字。Java不支持C中的bitfields能力。Java不支持C的typedef。Java不支持C的方法指针。. Java不支持C的可变参数表。

9. Java不支持[系统调用](http://zhidao.baidu.com/search?word=%E7%B3%BB%E7%BB%9F%E8%B0%83%E7%94%A8&fr=qb_search_exp&ie=utf8" \t "_blank)，既无法与操作系统进行交互，这是由它[跨平台](http://zhidao.baidu.com/search?word=%E8%B7%A8%E5%B9%B3%E5%8F%B0&fr=qb_search_exp&ie=utf8" \t "_blank)的特点决定的，而C则可以调用系统中的指令，如fork()。这也就是为什么Java中只有线程（Thread）概念而没有进程（Process）概念，而C两者均有。

10. 类型转换,在c和c十十中有时出现数据类型的隐含转换，这就涉及了自动强制类型转换问题。例如，在c十十中可将一浮点值赋予整型变量，并去掉其尾数。Java不支持c十十中的自动强制类型转换，如果需要，必须由程序显式进行强制类型转换。

1. 链表怎么判断有没有环路

两个指针，步长分别为1和2，然后看追没追到。

1. C++中inline和define的区别

内联函数inline是指那些定义在类体内的成员函数，即该函数的函数体放在类体内。引入内联函数的主要目的是：解决程序中函数调用的效率问题。inline 定义的类的内联函数，函数的代码被放入符号表中，在使用时直接进行替换，（像宏一样展开），没有了调用的开销，效率也很高。 很明显，类的内联函数也是一个真正的函数，编译器在调用一个内联函数时，会首先检查它的参数的类型，保证调用正确。然后进行一系列的相关检查，就像对待任何一个真正的函数一样。这样就消除了它的隐患和局限性。inline 可以作为某个类的成员函数，当然就可以在其中使用所在类的保护成员及私有成员。  
 内联函数和宏的区别在于，宏是由预处理器对宏进行替代，而内联函数是通过编译器  
控制来实现的。而且内联函数是真正的函数，只是在需要用到的时候，内联函数像宏一样  
的展开，所以取消了函数的参数压栈，减少了调用的开销。你可以象调用函数一样来调用  
内联函数，而不必担心会产生于处理宏的一些问题。  
 内联函数与带参数的宏定义进行下比较，它们的代码效率是一样，但是内联欢函数要优于宏定义，因为内联函数遵循的类型和作用域规则，它与一般函数更相近，在一些编译器中，一旦关上内联扩展，将与一般函数一样进行调用，比较方便。

什么时候用内联函数：  
内联函数在C++类中，应用最广的，应该是用来定义存取函数。我们定义的类中一般会把数据成员定义成私有的或者保护的，这样，外界就不能直接读写我们类成员的数据了。对于私有或者保护成员的读写就必须使用成员接口函数来进行。如果我们把这些读写成  
员函数定义成内联函数的话，将会获得比较好的效率。

内联函数的优缺点：  
我们可以把它作为一般的函数一样调用，但是由于内联函数在需要的时候，会像宏一  
样展开，所以执行速度确比一般函数的执行速度要快。当然，内联函数也有一定的局限性。就是函数中的执行代码不能太多了，如果，内联函数的函数体过大，一般的编译器会放弃内联方式，而采用普通的方式调用函数。(换句话说就是，你使用内联函数，只不过是向编译器提出一个申请，编译器可以拒绝你的申请）这样，内联函数就和普通函数执行效率一样了。

如何禁止函数进行内联：  
如果使用VC++，可以使用/Ob命令行参数。当然，也可以在程序中使用 #pragma auto\_inline达到相同的目的。

注意事项：  
1.在内联函数内不允许用循环语句和开关语句。  
2.内联函数的定义必须出现在内联函数第一次被调用之前。