1. malloc和new有啥区别

就3点 1成对使用 2new和malloc都是在堆空间申请内存 3 new=malloc+对象的构造函数 delete=对象的析构函数+free

1. c++多态的实现

c++的多态性用一句话概括就是:在基类的函数前加上virtual关键字，在派生类中重写该函数，运行时将会根据对象的实际类型来调用相应的函数，如果对象类型是派生类，就调用派生类的函数，如果对象类型是基类，就调用基类的函数。

通过静态连编和动态连编，静态连编包括普通函数重载、类成员函数重载、运算符重载、模板重载

一种是"动态重载"，指的是运行时确定调用哪个，包括：覆盖。"静态联编"和"动态联编"共同组成了c++的"多态"功能。

关于虚函数表：

        a. 什么是虚函数表：对含有"virtual"修饰符的类，编译器在编译时，会给该类型制造一个该类型所属的虚函数表；

        b. 虚函数表本质上是：一个函数指针表，类的函数名-----地址，这个函数必须用virtual修饰过；

        c. 虚函数表在哪：在应用程序的常量区，；

        d. 虚函数表怎么作用：运行时，根据指针所对应的对象是哪个类型的对象，从虚函数表中找到对应函数名的地址进而执行；

        对析构函数用virtual修饰是典型的覆盖的运用，当(父类指针指向的)子类对象发生析构时：

        a. 如果父类的析构函数用virtual修饰，那么析构时，通过"动态联编"会调用子类的析构函数；然后自然析构父类；

        b. 如果父类的析构函数没有用virtual修饰，那么析构时，仅仅自然析构父类；这个过程实际就是所谓的隐藏；

c++11另外还规定了修饰符final，对于不想再被子类覆盖的虚函数，加入final修饰符，它的子类即便想覆盖也无妨再覆盖了，或者说，覆盖行为到它这里截止；

c++多态的总结：

        1、"静态联编"：

      1、同一作用域(相同名字空间，类型下相同作用域(public/protected/private))的普通/成员函数，函数名相同，参数不相同；

            2、运算符重载；

            3、编译时确定模板实现类(模板重载)；

            如：template<class T> class A{ T data}; A<int> a；

            4、隐藏；隐藏是程序开发的bug应该加以避免；

        2、"动态联编"：

            指的就是覆盖，运行时由虚函数表根据所处对象的类型，确定执行哪一个函数；

            良好的程序应该使用覆盖；

        3、如何避免覆盖变成隐藏：

            a. 不要忘了用virtual修饰；

            b. 同时使用override修饰，忘了用virtual时编译器可以报错；

3.STL中Vector和List的区别

vector拥有一段连续的内存空间，能很好的支持随机存取，  
因此vector<int>::iterator支持“+”，“+=”，“<”等操作符。

list的内存空间可以是不连续，它不支持随机访问，  
因此list<int>::iterator则不支持“+”、“+=”、“<”等

vector<int>::iterator和list<int>::iterator都重载了“++”运算符。

总之，如果需要高效的随机存取，而不在乎插入和删除的效率，使用vector;  
如果需要大量的插入和删除，而不关心随机存取，则应使用list。

list是一个双向链表结构，和vector完全不同，vector可以下标操作，但是list是不可以的。

常用数据结构：

一、数组

　　1. 静态数组  
　　　　array //在编译期确定数组大小，在运行期无法改变数组大小，所以称之为静态数组。

　　2. 动态数组  
　　　　2.1. vector  
　　　　2.2. priority\_queue  
　　　　2.3. deque  
　　　　2.4. stack  
　　　　2.5. queue  
二、单向链表  
　　forward\_list

三、双向链表  
　　list

四、树  
　　1. set  
　　2. multiset  
　　3. map  
　　4. multimap

五、映射  
　　1. unordered\_set  
　　2. unordered\_multiset  
　　3. unordered\_map  
　　4. unordered\_multimap

[IMG_256](https://www.cnblogs.com/woxinfeixiang2015/p/javascript:void(0);)

代码运行的4个阶段

预处理-编译-汇编-链接。

Const 和#define的区别

是否分配内存，define 是替换 运行时替换

一个由C/C++编译的程序占用的内存分为以下几个部分 ：

    1、栈区（stack）— 由编译器自动分配释放 ，存放函数的参数值，局部变量的值等。其

操作方式类似于数据结构中的栈。

    2、堆区（heap） — 一般由程序员分配释放， 若程序员不释放，程序结束时可能由OS回

收 。注意它与数据结构中的堆是两回事，分配方式倒是类似于链表，呵呵。

    3、全局区（静态区）（static）—，全局变量和静态变量的存储是放在一块的，初始化的

全局变量和静态变量在一块区域， 未初始化的全局变量和未初始化的静态变量在相邻的另

一块区域。 - 程序结束后由系统释放。

    4、文字（只读）常量区 —常量字符串就是放在这里的。 程序结束后由系统释放

    5、代码段—存放函数体的二进制代码，直接的操作数也是存储在这个位置的。如int a=4;。

首先，讨论的堆和栈指的是内存中的“堆区”和“栈区”，OC语言是C语言的超集，所以先了解C语言的内存模型的内存管理会有很大的帮助。C语言的内存模型分为5个区：栈区、堆区、静态区、常量区、代码区。每个区存储的内容如下：

1、栈区：存放函数的参数值、局部变量等，由编译器自动分配和释放，通常在函数执行完后就释放了，其操作方式类似于数据结构中的栈。栈内存分配运算内置于CPU的指令集，效率很高，但是分配的内存量有限，比如iOS中栈区的大小是2M。

2、堆区：就是通过new、malloc、realloc分配的内存块，编译器不会负责它们的释放工作，需要用程序区释放。分配方式类似于数据结构中的链表。在iOS开发中所说的“内存泄漏”说的就是堆区的内存。

3、静态区：全局变量和静态变量（在iOS中就是用static修饰的局部变量或者是全局全局变量）的存储是放在一块的，初始化的全局变量和静态变量在一块区域，未初始化的全局变量和未初始化的静态变量在相邻的另一块区域。程序结束后，由系统释放。

4、常量区：常量存储在这里，不允许修改。

5、代码区：存放函数体的二进制代码。

堆和栈的区别：

1、堆空间的内存是动态分配的，一般存放对象，并且需要手动释放内存。当然，iOS引入了ARC（自动引用计数管理技术）之后，程序员就不需要用代码管理对象的内存了，之前MRC（手动管理内存）的时候，程序员需要手动release对象。另外，ARC只是一种中间层的技术，虽然在ARC模式下，程序员不需要像之前那么麻烦管理内存，但是需要遵循ARC技术的规范操作，比如使用属性限定符weak、strong、assigen等。因此，如果程序员没有按ARC的规则并合理的使用这些属性限定符的话，同样是会造成内存泄漏的。

2、栈空间的内存是由系统自动分配，一般存放局部变量，比如对象的地址等值，不需要程序员对这块内存进行管理，比如，函数中的局部变量的作用范围（生命周期）就是在调完这个函数之后就结束了。这些在系统层面都已经限定住了，程序员只需要在这种约束下进行程序编程就好，根本就没有把这块内存的管理权给到程序员，肯定也就不存在让程序员管理一说。

从申请的大小方面讲：

栈空间比较小；

堆空间比较大。

从数据存储方面来说：

栈空间中一般存储基本数据类型，对象的地址；

堆空间一般存放对象本身，block的copy等。

## **new和delete运算符**

我们得出: new/delete会调用类的构造函数和析构函数

注意: a. new/delete,malloc/free最好匹配使用;

b. new和malloc最大区别: new会调用类的构造函数,malloc不会;

c. delete和free同理;new/delete是运算符,malloc/free函数。所以new/delete效率应该会高点。