**[c++类的实例化，有没有new的区别](https://www.cnblogs.com/chris-cp/p/3578976.html)**

A a;

A\* a=new a();

以上两种方式皆可实现类的实例化，有new的区别在于：

1.前者在堆栈中分配内存，后者为动态内存分配，在一般应用中是没有什么区别的，但动态内存分配会使对象的可控性增强。

2.不加new在栈中分配内存

3.大程序用new，小程序直接申请

4.只是把对象分配在堆栈内存中

5.new必须delete删除，不用new系统会自动回收内存

new创建类对象，使用完后需使用delete删除，跟申请内存类似。所以，new有时候又不太适合，比如在频繁调用场合，使用局部new类对象就不是个好选择，使用全局类对象或一个经过初始化的全局类指针似乎更加高效。

栈区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。堆区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。堆和栈的区别可以用如下的比喻来看出：使用栈就象我们去饭馆里吃饭，只管点菜（发出申请）、付钱、和吃（使用），吃饱了就走，不必理会切菜、洗菜等准备工作和洗碗、刷锅等扫尾工作，他的好处是快捷，但是自由度小。使用堆就象是自己动手做喜欢吃的菜肴，比较麻烦，但是比较符合自己的口味，而且自由度大。

**一、new创建类对象与不new区别**

* new创建类对象需要指针接收，一处初始化，多处使用
* new创建类对象使用完需delete销毁
* new创建对象直接使用堆区，而局部不用new定义类对象则使用栈区
* new对象指针用途广泛，比如作为函数返回值、函数参数等
* 频繁调用场合并不适合new，就像new申请和释放内存一样

**二、new创建类对象实例**

**1、new创建类对象例子：**

CTest\* pTest=new CTest();

CTest -> Public;

Delete pTest;

pTest用来接收类对象指针。

不用new，直接使用类定义申明：

CTest mTest;

此种创建方式，使用完后不需要手动释放，该类析构函数会自动执行。而new申请的对象，则只有调用到delete时再会执行析构函数，如果程序退出而没有执行delete则会造成内存泄漏。

**2、只定义类指针**

这跟不用new申明对象有很大区别，类指针可以先行定义，但类指针只是个通用指针，需要在new之前为该类对象分配任何大小的内存空间。比如：

CTest\* pTest=NULL;

使用普通方式创建的类对象，在创建之初就已经分配了内存空间。而类指针，如果未经过对象初始化，则不需要delete释放。

**3、new对象指针作为函数参数和返回值**

class Friend

{

public:

Friend(int a);

~Friend();

public:

int PrivateTest();

int IsPublic;

private:

int IsPrivate;

};

Friend::Friend(int a)

{

IsPublic = a;

}

Friend::~Friend()

{

}

int Friend::PrivateTest()

{

return IsPrivate;

}

int main()

{

Friend MyFriend(1);

cout << MyFriend.IsPublic << endl;

Friend\* NewFriend = new Friend(123);

cout<<NewFriend->IsPublic << endl;

delete NewFriend;

return 0;

}