引子：

在 C++ 中，你也许经常使用 new 和 delete 来动态申请和释放内存，但你可曾想过以下问题呢？

• new 和 delete 是函数吗？

• new [] 和 delete [] 又是什么？什么时候用它们？

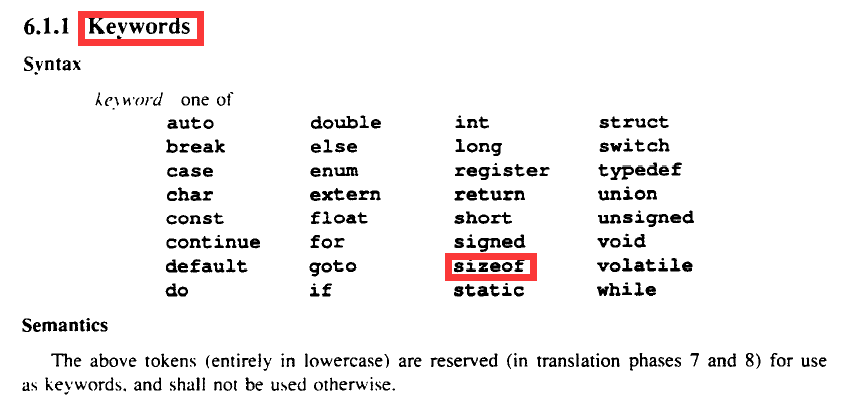
• 你知道 operator new 和 operator delete 吗？

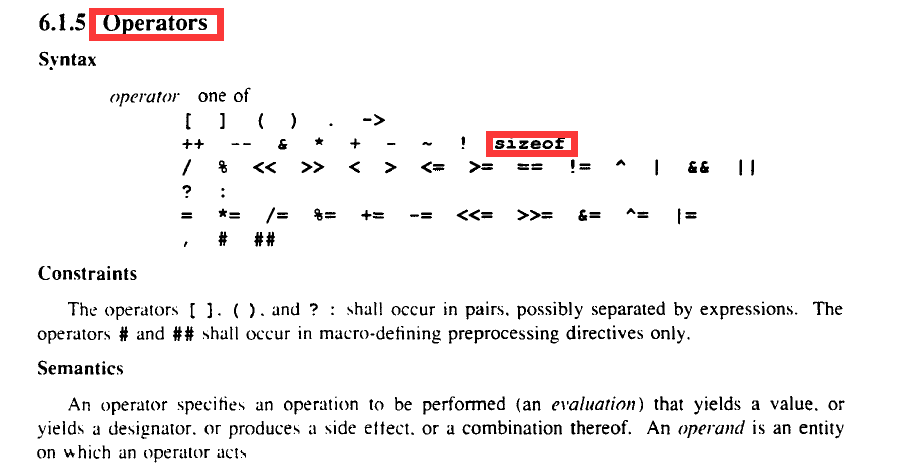
• 为什么 new/delete ，new[]/delete[] 配对使用？

一．new 和 delete 到底是什么？

sizeof 是函数吗？

相信很多资深的程序员也没注意过这个问题，其实在c的官方文档中指明sizeof不是函数，是c的关键字和操作符!





和sizeof类似，new 和 delete也不是函数，它们都是C++ 定义的关键字，通过特定的语法可以组成表达式。和 sizeof 不同的是，sizeof 在编译时候就可以确定其返回值，但new和delete背后的机制则比较复杂。在我用实例来解释new背后的机制之前，我们需要知道 operator new 和 operator delete 是什么玩意

二．operator new 和 operator delete

这两个其实是 C++ 语言标准库的库函数，原型分别如下：

void \*operator new(size\_t); //allocate an object

void \*operator delete(void \*); //free an object

void \*operator new[](size\_t); //allocate an array

void \*operator delete[](void \*); //free an array

后面两个你可以先不看，前面两个均是 C++ 标准库函数，这两个函数和 C 语言中的 malloc 和 free 函数有点像了，都是用来申请和释放内存的，最大的区别是这两个函数叜分配失败时可能会抛出异常，并且 operator new 申请内存之后不对内存进行初始化，直接返回申请内存的指针。我们可以直接在我们的程序中使用这几个函数

三．new/delete背后的机制

知道上面两个函数之后，我们用一个实例来解释 new 和 delete 背后的机制：

我们不用简单的 C++ 内置类型来举例，使用复杂一点的类类型，定义一个类 A：

class A

{

public:

A(int v) : var(v)

{

fopen\_s(&file, "test", "r");

}

~A()

{

fclose(file);

}

private:

int var;

FILE \*file;

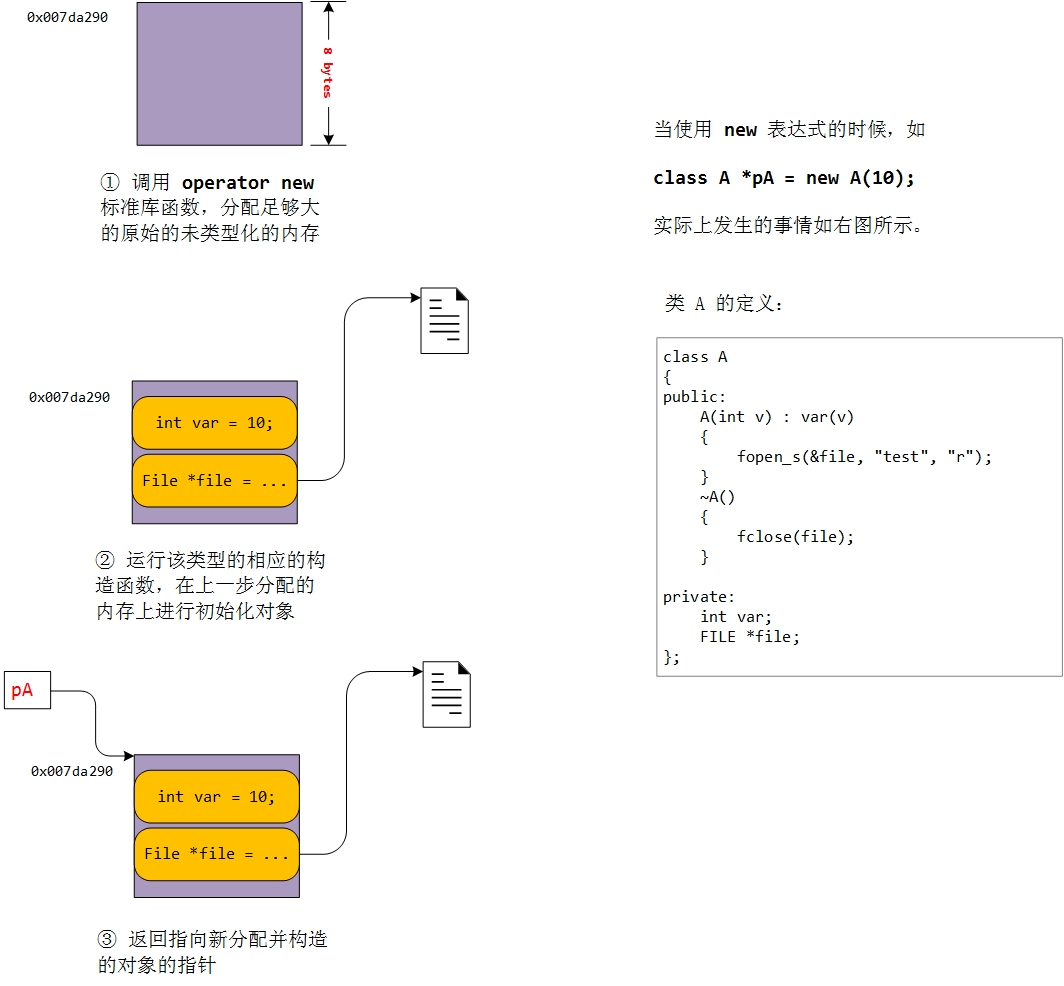
};

很简单，类 A 中有两个私有成员，有一个构造函数和一个析构函数，构造函数中初始化私有变量 var 以及打开一个文件，析构函数关闭打开的文件。

我们使用

class A \*pA = new A(10);

来创建一个类的对象，返回其指针 pA,如下图所示 new 背后完成的工作：

[](https://github-camo.global.ssl.fastly.net/48182649501319d27121ede912b19be491b1b0a2/687474703a2f2f692e696d6775722e636f6d2f454a736e6b4a342e6a7067)

简单总结一下：

1)首先需要调用上面提到的 operator new 标准库函数，传入的参数为 class A 的大小，这里为 8 个字节(两个变量：var,file的大小)，这样函数返回的是分配内存的起始地址，这里假设是 0x007da290；

2)上面分配的内存是未初始化的，也是未类型化的，第二步就在这一块原始的内存上对类对象进行初始化，调用的是相应的构造函数，这里是调用 A:A(10); 这个函数，从图中也可以看到对这块申请的内存进行了初始化，var=10, file 指向打开的文件；

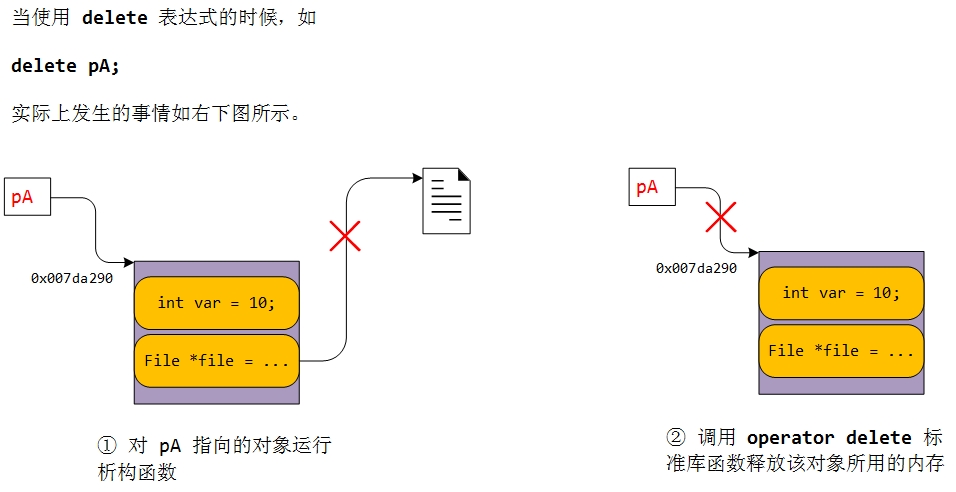
3)最后一步就是返回新分配并构造好的对象的指针，这里 pA 就指向 0x007da290 这块内存，pA 的类型为类 A 对象的指针。

所有这三步，你都可以通过反汇编找到相应的汇编代码，在这里我就不列出了。

好了，那么 delete 都干了什么呢？还是接着上面的例子，如果这时想释放掉申请的类的对象怎么办？当然我们可以使用下面的语句来完成：

delete pA;

delete 所做的事情如下图所示：

[](https://github-camo.global.ssl.fastly.net/8bfb4701b5c270dc3b27c86bce57f2d31562d82f/687474703a2f2f692e696d6775722e636f6d2f3156767239467a2e6a7067)

delete 就做了两件事情：

1)调用 pA 指向对象的析构函数，对打开的文件进行关闭。

2)通过上面提到的标准库函数 operator delete 来释放该对象的内存，传入函数的参数为 pA 的值，也就是 0x007d290。

总结：

1. new后面接的是c++的基本类型时，如int,本身和malloc没有本质的区别
2. new后面接的是类时，会多了构造函数和析构函数的调用；

四．new[]/delete[]背后的机制

我们经常要用到动态分配一个数组，也许是这样的：

string \*psa = new string[10];

int \*pia = new int[10];

上面在申请一个数组时都用到了 new [] 这个表达式来完成，按照我们上面讲到的 new 和 delete 知识，第一个数组是 string 类型，分配了保存对象的内存空间之后，将调用 string 类型的默认构造函数依次初始化数组中每个元素；第二个是申请具有内置类型的数组，分配了存储 10 个 int 对象的内存空间，但并没有初始化。

如果我们想释放空间了，可以用下面两条语句：

delete [] psa;

delete [] pia;

都用到 delete [] 表达式，注意这地方的 [] 一般情况下不能漏掉！我们也可以想象这两个语句分别干了什么：第一个对 10 个 string 对象分别调用析构函数，然后再释放掉为对象分配的所有内存空间；第二个因为是内置类型不存在析构函数，直接释放为 10 个 int 型分配的所有内存空间。

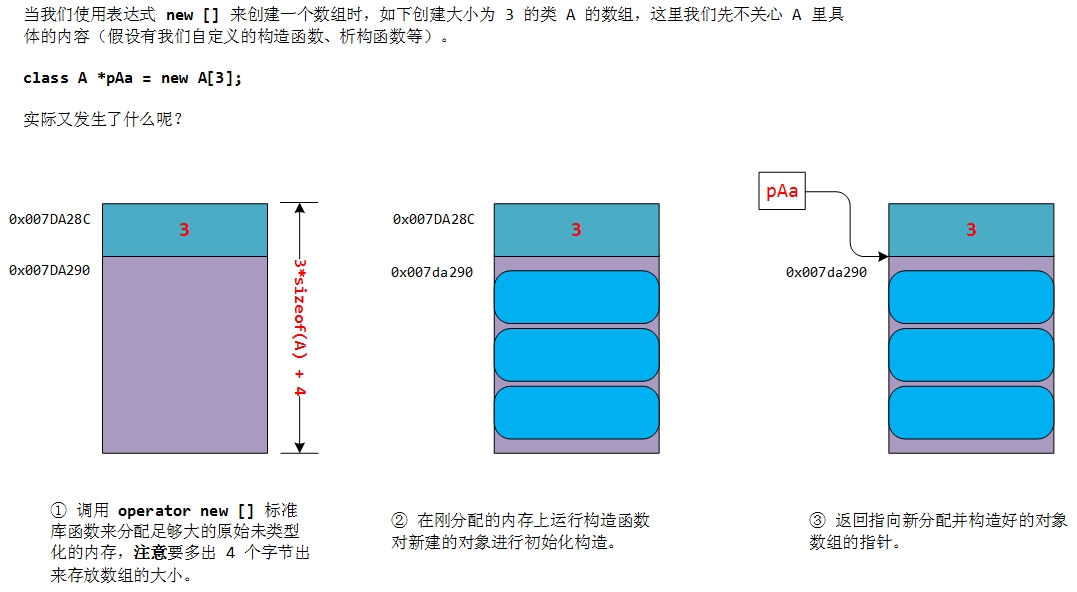
这里对于第一种情况就有一个问题了：我们如何知道 psa 指向对象的数组的大小？怎么知道调用几次析构函数？

这个问题直接导致我们需要在 new [] 一个对象数组时，需要保存数组的维度，C++ 的做法是在分配数组空间时多分配了 4 个字节的大小，专门保存数组的大小，在 delete [] 时就可以取出这个保存的数，就知道了需要调用析构函数多少次了。

还是用图来说明比较清楚，我们定义了一个类 A，但不具体描述类的内容，这个类中有显示的构造函数、析构函数等。那么 当我们调用

class A \*pAa = new A[3];

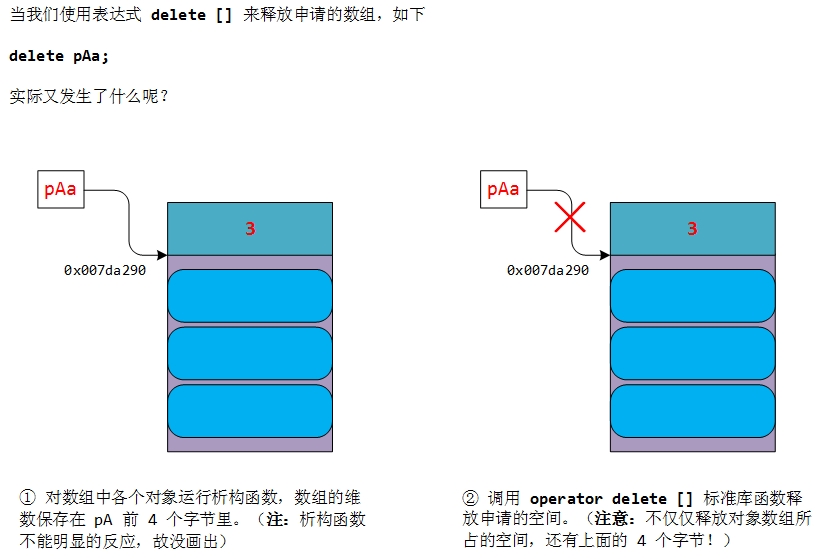
时需要做的事情如下：

[](https://github-camo.global.ssl.fastly.net/1a2da83d54ac0110f8cdb13a6645dd5f9240e760/687474703a2f2f692e696d6775722e636f6d2f366857304431702e6a7067)

从这个图中我们可以看到申请时在数组对象的上面还多分配了 4 个字节用来保存数组的大小，但是最终返回的是对象数组的指针，而不是所有分配空间的起始地址。

这样的话，释放就很简单了：

delete []pAa;

[](https://github-camo.global.ssl.fastly.net/db76cce60aaa6f0ad3ab65cd00b0ca26a5271547/687474703a2f2f692e696d6775722e636f6d2f3155425444316c2e6a7067)

这里要注意的两点是：

• 调用析构函数的次数是从数组对象指针前面的 4 个字节中取出；

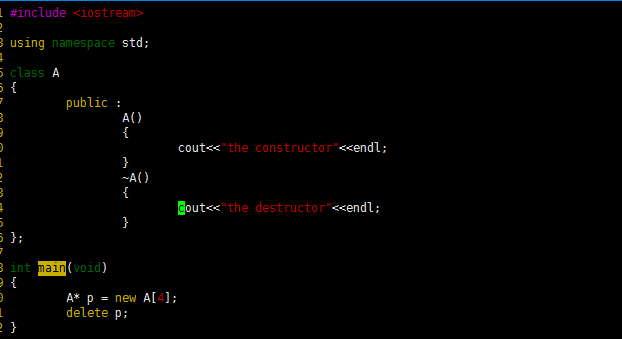
• 传入 operator delete[] 函数的参数不是数组对象的指针 pAa，而是 pAa 的值减 4。

五．为什么 new/delete 、new []/delete[] 要配对使用？

用上面的例子来说明：

class A \*p = new class A[4];

delete p;





那么 delete pAa; 做了两件事：

• 调用一次 pAa 指向的对象的析构函数；

• 调用 operator delete(pAa); 释放内存。

这里存在两个问题：

1. 这里只对数组的第一个类对象调用了析构函数，后面的两个对象均没调用析构函数，如果类对象中申请了大量的内存需要在析构函数中释放，而你却在销毁数组对象时少调用了析构函数，这会造成内存泄漏;

2)直接释放 pAa 指向的内存空间，这个总是会造成严重的段错误，程序必然会奔溃！因为分配的空间的起始地址是 pAa 指向的地方减去 4 个字节的地方。你应该传入参数设为那个地址！

同理，你可以分析如果使用 new 来分配，用 delete [] 来释放会出现什么问题？是不是总会导致程序错误？

总的来说，记住一点即可：new/delete、new[]/delete[] 要配套使用总是没错的！