new expression、operator new 和 placement new——三个妞（new）的故事（1）

之前虽然一直知道有 new expression、operator new 和 placement new，但对于这三个"new"，却不甚了了，这些天从《深度探索 C++对象模型》读到 new 和 delete，特意结合《C++ Primer》 写下这篇笔记，以作总结。三个虽然都是"妞"（new），但每个妞都不相同各有各的特点，各 有各的风味，本文重点在于总结比较这三个"妞"，但期间也不忘提一提推倒这三个"妞"的哥们 ——delete。

new expression 和 operator new

一个看起来很简单的 new expression 运算，其实暗含一些步骤，像这样的一次简单运用： `int *p=new int (5)`实际上包含着两个步骤：

1. 调用一个合适的 operator new 实体分配足够的未类型化的内存。
2. 调用合适的构造函数初始化这块内存，当然 `int` 没有构造函数，但是会进行赋值操作： `*p=5`。

由此可见：new expression 和 operator new 完全不是一回事，但关系不浅——operator new 为 new expression 分配内存。

摘录一下 《C++ primer》关于对比 new expression 和 operator new 的一小段话：

> *标准库函数 operator new 和 operator delete 的命名容易让人误解。与其他operator 函 数（如 `operator=`）不同，这些函数没有重载 new 或 delete expression，实际上，我们 不能重定义 new 或 delete expression 的行为。*

这段话有两个要点：

1. operator new 和 operator delete 不是 new expression 和 delete expression 的重载，它 们完全是另外的一个独立的东西，具有不同的语意，这与 operator + 是对+ expression 的重载不同。
2. new expression 和 delete expression 是不能被重载的，可以看出它们与普通的 expression 不同。

operator new 其实也是可以直接利用的，譬如当我们只想分配内存，而不愿意进行初始化的 时候，我们就可以直接用 operator new 来进行。用法如下：

```
T* newelements = static_cast<T*>(operator new ( sizeof(T) );
```

标准库重载有两个版本的 operator new，分别为单个对象和数组对象服务，单个对象版本的 提供给分配单个对象 new expression 调用，数组版的提供给分配数组的 new expression 调 用：

```
void *operator new(size_t);      // allocate an objectvoid *operator
new[](size_t);      // allocate an array
```

我们可以分别重载这两个版本，来定义我们自己的分配单个对象或对象数组的内存方式。当我们自己在重载 operator new 时，不一定要完全按照上面两个版本的原型重载，唯一的两个 要求是：返回一个 `void*`类型和第一个参数的类型必须为 `size_t`。

还要注意的是，在类中重载的 operator new 和 operator delete 是隐式静态的，因为前者运 行于对象构造之前，后者运行与对象析构之后，所以他们不能也不应该拥有一个 this 指针来 存取数据。另外，new expression 默认调用的是单参数的 operator new——上面声明的那种， 而其它不同形式的重载，则只能显式调用了。

delete expression 与 new expression 相对应，而 operator delete 则与 operator new 对应。 依上所述，则不难推断出关于 delete expression 和 operator delete 之间的关系以及一些特 性，此略。

当使用 new expression 来动态分配数组的时候，Lippman 在《深度探索 C++对象模型》中指出： 当分配的类型有一个默认构造函数的时候，new expression 将调用一个所谓的 `vec_new()`函 数来分配内存，而不是 operator new 内存。但我在 VC ++ 2010 上测试的结果却是，不论有没 有构造函数，new expression 都是调用 operator new 来分配内存，并在此之后，调用默认构 造函数逐个初始化它们，而不调用所谓的 `vec_new()`，也许 cfront 确实离我们有点遥远。

参考：Lippman 的两本书《深度探索 C++对象模型》和《C++ Primer》。