Memory Management

Use the same form in corresponding uses of new and delete

eryar@163.com

对应的 ***new*** 和 ***delete*** 要采用相同的形式

用 ***new operator*** 时会发生两件事：首先，内存通过 ***operator new*** 被分配；然后，为被分配的内存调用一个或多个构造函数。

用 ***delete operator*** 时也会发生两件事：首先，为将释放的内存调用一个或多个析构函数；然后，通过 ***operator delete*** 释放内存。

对于 ***delete operator*** 来说会有这样一个重要的问题：内存中有多少个对象需要被删除呢？删除多了，程序也就崩溃了；删除少了，就有内存未被正确释放。这个问题简单来说就是：要被删除的指针指向是单个对象，还是对象数组？这只有你来告诉 ***delete operator***。如果你在用 ***delete operator*** 时没有括号，***delete*** 就会认为指向的是单个对象；否则它就会认为指向的是一个数组。

Prefer new and delete to malloc and free

malloc 和 ***free***（及其变体）会产生问题的原因在于它们太简单：他们不知道构造函数和析构函数。（有时越简单越好。）

假设用两种方法给一个包含 ***10*** 个 ***string*** 对象的数组分配空间，一个用 ***malloc***，另一个用 ***new***：

```
string *stringarray1 =
static_cast<string*>(malloc(10 * sizeof(string)));

string *stringarray2 = new string[10];
```

其结果是，***stringarray1*** 确实指向的是可以容纳 ***10*** 个 ***string*** 对象的足够空间，但内存里并没有创建这些对象。当释放这些内存时，你一定会这么做：

```
free(stringarray1);
delete [] stringarray2;
```

调用 ***free*** 将会释放 ***stringarray1*** 指向的内存，但内存里的 ***string*** 对象不会调用析构函数。如果 ***string*** 对象象一般情况那样，自己已经分配了内存，那这些内存将会全部丢失。相反，当对 ***stringarray2*** 使用 ***delete*** 时，数组里的每个 ***string*** 对象都会在内存释放前调用析构函数。即然 ***new*** 和 ***delete*** 可以这么有效地与构造函数和析构函数交互，选用它们是显然的。

把 ***new***/***delete*** 与 ***malloc***/***free*** 混用也是个坏想法。对一个用 ***new*** 获取来的指针调用 ***free***，或者对一个用 ***malloc*** 获取来的指针调用 ***delete***，其后果是不可预测的。

示例程序：

```
#include <iostream>
using namespace std;

class CTest
{
public:
    CTest() { cout<<"Default constructor."<<endl; }
    ~CTest() { cout<<"Default destrcutor."<<endl; }

};

int main(int argc, char *argv[])
{
    cout<<"=== Test new/delete begin ==="<<endl;
    CTest* pTestNewDelete = new CTest;
    delete pTestNewDelete;
    cout<<"=== Test new/delete end   ==="<<endl;

    cout<<"~~~ Test malloc/free begin ~~~"<<endl;
    CTest* pTestMallocFree = static_cast<CTest*>
(malloc(sizeof(CTest)));
    free(pTestMallocFree);
    cout<<"~~~ Test malloc/free end   ~~~"<<endl;

    return 0;
}
```

输出：

=== Test new/delete begin ===
Default constructor.
Default destrcutor.
=== Test new/delete end ===
~~~ Test malloc/free begin ~~~
~~~ Test malloc/free end ~~~

来源：

1. Scott Meyers Effective C++