new/delete 和malloc/free 的区别一般汇总

一、基本概念

malloc/free：

1、函数原型及说明：

void \*malloc(long NumBytes)：该函数分配了NumBytes个字节，并返回了指向这块内存的指针。如果分配失败，则返回一个空指针（NULL）。

void free(void \*FirstByte)： 该函数是将之前用malloc分配的空间还给程序或者是操作系统，也就是释放了这块内存，让它重新得到自由。

2.内存操作：

malloc函数的参数是接受需要分配的内存字节数，如果内存能够满足请求量，那么将会返回：指向被分配的内存块起始位置

free函数释放的是指针指向的内存(不是释放的指针本身，不会删除指针本身),其中指针必须指向所释放内存空间的首地址

new/free:

1.操作时发生事件：

new的时候会有两个事件发生：1).内存被分配(通过operator new 函数) 2).为被分配的内存调用一个或多个构造函数构建对象

delete的时候，也有两件事发生：1).为将被释放的内存调用一个或多个析构函数 2).释放内存(通过operator delete 函数)

2.特殊应用：

使用delete是未加括号，delete便假设删除对象是单一对象。否则便假设删除对象是个数组

因此，如果在调用new时使用了[]，则在调用delete时也使用[]，如果你在调用new的时候没有[]，那么也不应该在调用时使用[]。

二、malloc/free 和new/delete 的本质区别：

1.malloc/free是C/C++语言的标准库函数，new/delete是C++的运算符

2.new能够自动分配空间大小

3.对于用户自定义的对象而言，用maloc/free无法满足动态管理对象的要求。对象在创建的同时要自动执行构造函数，对象在消亡之前要自动执行析构函数。由于malloc/free是库函数而不是运算符，不在编译器控制权限之内，不能够把执行构造函数和析构函数的任务强加于malloc/free。因此C++需要一个能对对象完成动态内存分配和初始化工作的运算符new，以及一个能对对象完成清理与释放内存工作的运算符delete---简而言之 new/delete能进行对对象进行构造和析构函数的调用进而对内存进行更加详细的工作，而malloc/free不能。

三、联系

既然new/delete的功能完全覆盖了malloc/free，为什么C++还保留malloc/free呢？因为C++程序经常要调用C函数，而C程序只能用malloc/free管理动态内存。如果用free释放“new创建的动态对象”，那么该对象因无法执行析构函数而可能导致程序出错。如果用delete释放“malloc申请的动态内存”，理论上讲程序不会出错，但是该程序的可读性很差。所以new/delete，malloc/free必须配对使用。

四、使用范例

void \* malloc(size\_t size);

用malloc 申请一块长度为length 的整数类型的内存，程序如下：

int \*p = (int \*) malloc(sizeof(int) \* length);

我们应当把注意力集中在两个要素上：“类型转换”和“sizeof”。

1. malloc 返回值的类型是void \*，所以在调用malloc 时要显式地进行类型转换，将void \* 转换成所需要的指针类型。 􀂋

2.malloc 函数本身并不识别要申请的内存是什么类型，它只关心内存的总字节数。

void free( void \* memblock );

为什么free 函数不象malloc 函数那样复杂呢？这是因为指针p 的类型以及它所指的内存的容量事先都是知道的，语句free(p)能正确地释放内存。如果p 是NULL 指针，那么free

对p 无论操作多少次都不会出问题。如果p 不是NULL 指针，那么free 对p连续操作两次就会导致程序运行错误。

new/delete 的使用要点

运算符new 使用起来要比函数malloc 简单得多，例如：

int \*p1 = (int \*)malloc(sizeof(int) \* length);

int \*p2 = new int[length];

这是因为new 内置了sizeof、类型转换和类型安全检查功能。对于非内部数据类型的对象而言，new 在创建动态对象的同时完成了初始化工作。如果对象有多个构造函数，那么new 的语句也可以有多种形式。

如果用new 创建对象数组，那么只能使用对象的无参数构造函数。例如

Obj \*objects = new Obj[100]; // 创建100 个动态对象 不能写成

Obj \*objects = new Obj[100](1);// 创建100 个动态对象的同时赋初值1

在用delete 释放对象数组时，留意不要丢了符号‘[]’。例如

delete []objects; // 正确的用法

delete objects; // 错误的用法 后者相当于delete objects[0]，漏掉了另外99 个对象。

new出来的指针是直接带类型信息的，而malloc返回的都是void指针，需要进行类型转换