malloc(free)与new(delete)的区别

malloc与free是C++/C语言的标准库函数，new/delete是C++中运算符。

使用new/free的适应面向对象编程，malloc/free函数无法满足动态对象的要求。类的对象创建时，需要自动执行构造函数，当类销毁时，需要自动执行析构函数。由于malloc/free函数是库函数，不是运算符，不在编译器的控制范围内，不能够把执行构造函数和析构函数的任务强加给malloc/free函数，需要不是库函数的运算符new和delete。

```
class Obj
{
public :
    Obj(void){ cout << “Initialization” << endl; }
    ~Obj(void){ cout << “Destroy” << endl; }
    void Initialize(void){ cout << “Initialization” << endl; }
    void Destroy(void){ cout << “Destroy” << endl; }
};

void UseMallocFree(void)
{
    Obj  *a = (obj *)malloc(sizeof(obj));   // 申请动态内存
    a->Initialize();                        // 初始化
    //…
    a->Destroy();   // 清除工作
    free(a);        // 释放内存
}

void UseNewDelete(void)
{
    Obj  *a = new Obj;  // 申请动态内存并且初始化
    //…
    delete a;           // 清除并且释放内存
}
```

使用malloc只能做到分配内存的工作，所以还需要一个初始化函数Initilize来初始化函数。使用delete只能做到释放内存，不能做到数据的清除作用。

所以我们不要企图使用malloc/free来完成对象的内存管理，应该使用new/delete。由于内部数据类型对象没有构造与析构的过程，所以malloc/free 和 new/delete是等价的。

既然new/delete的功能完全覆盖了malloc/free，为什么C++不把malloc/free淘汰出局呢？这是因为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。