使用new操作符来分配动态内存时，如果是动态数组，如：

int\* p = new int[7];

delete[] p;

p = nullptr;

new将内存分配和对象构造组合在一起。当分配单个对象时，没有问题，但在分配动态数组时，内存分配和对象构造分离开来，会更加有效，可以在真正需要时才执行对象创建操作。

new分配动态数组的存在一个大的缺陷：没有默认构造函数的对象无法使用。

例：程序C++11\_test9

class A

{

public:

A(int k) : number\_(k)

{

}

~A() = default;

private:

int number\_;

};

// A\* ptr = new A[3]; // error, A不存在默认构造函数

使用allocator类可以将内存分配和对象构造分离开来。它分配的内存时原始的未构造的内存。

例：程序C++11\_test9

*allocator*<A> alloc;

A\* ptr = alloc.*allocate*(3); // 为3个A分配了内存

// 构造动态数组ptr中的对象

*cout* << "start construct" << *endl*;

alloc.*construct*(&ptr[0], 1);

alloc.*construct*(&ptr[1], 2);

alloc.*construct*(&ptr[2], 3);

ptr[0].Print();

ptr[1].Print();

ptr[2].Print();

// 删除之前分配的对象

*cout* << "start destructor" << *endl*;

alloc.*destroy*(&ptr[0]);

alloc.*destroy*(&ptr[1]);

alloc.*destroy*(&ptr[2]);

// 释放内存

*cout* << "delete memory" << *endl*;

alloc.*deallocate*(ptr, 3);

输出为：

start construct

Constructor A

Constructor A

Constructor A

number is 1

number is 2

number is 3

start destructor

Destructor A

Destructor A

Destructor A