|  |  |
| --- | --- |
| 1. 定位new表达式 2. 显式的析构函数调用 | New (place\_address) type[] {initializers}  Sp->~string(); |
| 1. 定位new表达式   就是定位到我们创建的内存的地址来构造对象。  这个地址不一定要指向operator new分配的内存。   1. Allocator的allcate与deallocate成员与operator new 与operator delete成员相似,只负责分配或释放内存空间，不会操作对象 2. 但是allocator多了一个construct成员来操作对象，若我们重载了operator new与operator delete成员自定义分配内存，那么我们需要定位new来进行构造对象 3. Place address是可以传入operator new 分配内存的指针(就是malloc(size)分配的内存返回指向内存的地址)，进行对initiallizers对象初始化，同construct作用 4. 定位new也可以只传入一个地址值，但是type要，初始对象可以不用，则编译器调用operator new(size\_t,void\*)来分配它的内存，然后返回指针再给定位new构造对象，得到的就是未定义的对象 5. 与construct的区别是给定位new表达式的指针无需一定指向operator new分配的内存（P753介绍） 6. 显示的析构函数调用 7. 和destroy类似，只是操作删除对象，不回收释放内存，可以重新使用该空间   代码:  void \*mem = malloc(1);// 分配内存，返回指向这个内存的地址  int \*i3 = new (mem) int(2);// 指向地址 构造初始值  cout << \*i3 << endl;// 为2  通过malloc分配内存返回这个内存的指针，再给定位new，指定的内存地址构造对象。  void \*mem2 = malloc(2);  int \*i4 = new(mem2) int;// 调用operator new(size\_t,void\*)  可以省去构造的对象，只有内存的地址和type，代表需要调用operator new(size\_t,void\*)  由new表达式负责在指定的地址初始化对象。 | |