|  |  |
| --- | --- |
| 1. 动态数组 2. New和数组 3. 分配一个数组会得到一个元素类型的指针 4. 初始化状态分配对象的数组 5. 动态分配一个空数组是合法的 6. 释放动态数组 7. 只能指针和动态数组 | Int \*pia = new int[get\_size()];  Typedef int art[42];  Int \*p = new art;  Int \*Pia = new int[10];  Int \*pia2=new int[10]();  String \*psa = new string[10];  String \*psa2 = new string[10]();  Int \*pia3 = new int[10]{0,1,2,3,4,5,6,7,8,9};  String \*psa3 = new string[10]{“a”,”an”};  Int \*p = new int[0];  Delete[] p;  Unique\_ptr<int[]> up(new int[10];  Up.release();//delete []  Up[i] = I;  Shared\_ptr<int> sp(new int[10],[](int \*p){delete[] p;});  Sp.reset();  \*(sp.get()+i\_=I; |
| 1. 动态数组 2. 一次为很多对象分配内存功能 3. 两种分配一个对象数组的方法，new和allocator 4. new将分配和初始化绑定在一起，没用到的内存，但是也分配了内存并且有初始值 就浪费了。allocator可将分配和初始化分离，这样更灵活 5. 最好使用vector标准库容器，而不是动态分配数组，容器的类可以使用默认版本的拷贝、赋值和析构操作。分配动态数组的类必须定义自己版本的操作，在拷贝、复制以及销毁对象时管理所关联的内存。 6. 分配动态数组的类必须定义自己版本的操作，在拷贝 复制以及销毁对象时管理所关联的内存 7. New和数组 8. 在类型名之后跟一对方括号，在其中指明要分配的对象的数目。分配成功返回指向第一个对象的指针 9. 大小必须是整形不必是常量，也可以用一个别名。 10. 分配一个数组会得到一个元素类型的指针 11. 叫 动态数组，并不是真的是数组类型的对象，而是得到一个数组元素类型的指针。即使使用类型别名，也不会得到数组类型的对象 12. 因为不是数组类型，所以不能使用begin()，也不能使用for处理动态数组中的元素 13. 初始化状态分配对象的数组 14. new分配的对象，都是默认初始化，可以对数组中的元素进行值初始化，跟上空括号。 15. 可以提供一个元素初始化器的花括号列表 16. 初始化可以混合，前4个初始化器初始，剩下的值初始化 17. 若初始化数目小于元素数目，剩余元素将进行值初始化。   若 大于， 报错，不会分配任何内存，抛出bad\_array\_length异常   1. 不能在值初始化的括号中给出初始化器，意味着不能用auto分配数组 2. 动态分配一个空数组是合法的 3. 动态分配一个空数组是合法的并且可以使用下标为0的元素，但不能解引用 4. 就像尾后指针一样，可以进行比较操作，可加减0； 5. 不能定义长度为0的数组，但可以一个动态分配一个空数组 6. 释放动态数组 7. 只删除p对象是错误的 未定义的 报错，猜想：只删除第一个元素吧？ 或者销毁第一个元素并且不指向数组了所以会访问不到，但是数组依然存在除了第一个位置的 8. 需要加上方括号对，按逆序销毁 9. 空括号是必须的。意味p只是指向一个对象数组的首元素，而不是一个类型为arrT的的单一对象   Note:单delete p可能不发出警告，但在运行时会异常   1. 只能指针和动态数组 2. Unique可以管理new分配的数组的unique\_ptr版本 3. 并且可以用下标访问数组中的元素 4. Unique销毁自动调用delete[] 5. 不直接支持管理动态数组，若想需要自定义删除器，使用lambda作为删除器，使用delete[]释放数组。   若没有删除器，则默认使用delete删除对象，会未定义的   1. 不提供下标访问数组的元素，而且只能指针类型不支持算术运算。需用get先获取内置指针再操作 | |