|  |  |
| --- | --- |
| 1. 定义union 2. 使用union类型 3. 匿名union 4. 含有类类型成员的union 5. 使用类管理union成员 6. 管理判别式并销毁string 7. 管理需要拷贝控制的联合成员 | Union token{}  Token first\_token = {‘a’}  Token last\_token;  Token \*pt = new token;  Last\_token.cval = ‘z’;  Pt->ival = 42  判别式 |
| Union联合是一种特殊的类，一个union可以有多个数据成员，但是在任意时刻只有一个数据成员可以有值。一个成员赋值之后，其它成员就变成未定义的状态了  不含有引用成员  可以有类类型  可以指定访问限定符，默认public  可以定义包括析构函数和析构函数在内的成员函数。  不能继承它类，不能做基类，不能有虚函数   1. 定义union 2. 使用union类型 3. 匿名union 4. 未命名的union 5. 同一个作用域可直接访问成员 6. 不能包含受保护的成员或私有成员，也不能定义成员函数 7. 含有类类型成员的union 8. 若类成员类型定义了自己的构造函数或拷贝控制成员，union对应的合成版本被声明为删除的。   为什么是删除的？因为必须要给这个类成员类型初始化，但是需要调用构造函数不可吗？   1. 当将union的值改为类类型成员对应的值时，必须使用该类型的构造函数   当将类类型成员的值改为一个其他值时，必须运行该类型的析构函数   1. 但是我们不知道当前union存储的是什么值（只能有一个值），所以无法确定是否调用析构类成员或者构造类成员 2. 某个类中含有的union成员，若union成员对应的拷贝控制成员为删除的，则类与之对应的拷贝控制也是删除的，所以需要显示的定义 3. 使用类管理union成员 4. 用一个enum来追踪union存储的值是什么，称：**判别式** 5. 构造函数初始为int值 6. 一组赋值运算符处理一个union值给union值 7. 需要类的析构函数处理string析构，不会自动执行 8. 管理判别式并销毁string 9. 在每个一组赋值运算符（除stirng外）都有叛别是否为string进行析构 10. 在管理string的赋值运算符有些差别   若两个union对象都是string，则可以直接赋值  若左侧不是string对象，则需要new构造一个，   1. 管理需要拷贝控制的联合成员 2. 拷贝赋值运算符 3. 需要一个通用函数处理所有可能的枚举成员，进行赋值，特殊是当右侧是str，左侧不是str则需要new一个   2）处理string成员的三种可能情况  a. 左侧和右侧都是string  b. 两个都不是string  c. 有一个是string  需要先处理左侧对象是string右侧不是string， 就析构，就能让b与c共同在一个else里调用通用函数  A情况  左侧右侧都是string，直接赋值  B情况  左侧不是string，右侧不是string调用通用函数  C情况  左侧是string，右侧不是string，调用通用函数(因为左侧的一开始就析构了)  左侧不是string，右侧是string，调用通用函数(将右侧的string给左侧)  /\*  思想很重要：  管理需要拷贝控制的联合成员：  1.类内有匿名union，有一个enum判别式  2.用拷贝控制给union赋值，通过enum是否构造还是析构  3.比如，给token = 32，判断当前是否为string，是就析构，再给ival赋值32  4.比如，给token赋值string，判断当前是否为string，若不是就构造，若是就直接赋值  \*/ | |