|  |  |
| --- | --- |
| 1. 命名空间 2. 命名空间的定义 3. 每个命名空间都是一个作用域 4. 命名空间可以是不连续的 5. 定义本书的命名空间 6. 定义命名空间成员 7. 模板特例化 8. 全局命名空间 9. 嵌套的命名空间 10. 内联命名空间 11. 未命名的命名空间 | 命名空间污染  命名空间  Namespace name{}// 无分号  Name::Query q = name::Query(“hello”);  Namespacename::sales\_data  Namespacename::operator+()  Namespace std{  Template<> struct hash<Sales\_data>;  }  全局命名空间 ::member\_name  Namespacename:namespacename::meme\_ber;  Inline namespace name;  Namespace{} |
| 1. 命名空间 2. 多个库将名字放置在全局命名空间中将引发**命名空间污染** 3. 命名空间为防止名字冲突提供更加可控的机制 4. 命名空间的定义 5. 命名空间作用域后面无需分号 6. 每个命名空间都是一个作用域 7. 不同命名空间内可以有相同名字的成员 8. 定义在某个命名空间中的名字可以被该命名空间内的其它成员直接访问，也可以被这些成员内嵌作用域中的任何单位访问。位于命名空间之外的代码则必须明确指出所用的名字属于哪个命名空间 9. 命名空间可以是不连续的 10. 一个命名空间的定义，若之前有了这个命名空间的定义是为这个命名空间添加成员，或创建一个新的命名空间 11. 这样可以让一个命名空间声明函数，另一个命名空间定义函数 12. 接口和实现分离机制 13. 定义多个类型不相干的命名空间应该使用单独的文件分别表示每个类型 14. 定义本书的命名空间 15. 可以分离，模块性 16. 使用时需要 using namespacename::类名; 17. 不把#Include头文件放在命名空间内部，不然报错，视图将命名空间std嵌套在命名空间cs中。（是会报错，好像重新定义string的命名空间了） 18. 定义命名空间成员 19. 定义命名空间成员时，可以一个大的命名空间包围，如果声明了这个成员，直接使用，而不需要再使用前缀。在重新声明命名空间时 也不用再次加上命名空间 20. 也可以在外部，需要明确指出其所属的命名空间 21. 模板特例化 22. 模板特例化必须定义在原始模板所属的命名空间中 23. 声明了模板特例化就能在外部定义 24. 全局命名空间 25. 在所有类、函数、命名空间之外即是全局命名空间 26. 以隐式的方式声明 27. 全局作用域中定义的名字被隐式地添加到全局命名空间中 28. ::member\_name 29. 嵌套的命名空间 30. Namespace stds{namespace stdss{}} 31. 不能实现在外层命名空间中声明函数，在内层命名空间中定义函数 32. 但是内层空间可以访问外层定义的成员 33. 内联命名空间 34. Inline,可以被外层命名空间直接使用 35. 就是在嵌套的命名空间中，若内部的命名空间是inline的，可以直接使用，而不需要两个命名空间。 36. 对于2)书上的一个命名空间引入两个头文件，头文件有命名空间，因为fifthEd命名空间是内联的，所以可以直接命名空间：：fifthEdmember,而另外一个Fourth访问里面的成员需要：命名空间：：fourthnamespace::fourthmember; 37. 可以在一个命名空间中引入我们自定义的文件，文件里有命名空间，但是不知道为什么不能讲#include <string.h>放入我们自定义的命名空间中 38. 未命名的命名空间 39. 未命名空间中定义的变量拥有**静态**声明周期，直到程序结束才销毁 40. 在某个给定的文件内可以不连续，但不能跨多个文件 41. 两个文件未命名命名空间中有相同的名字是互不相干的 42. 一个头文件定义了未命名的命名空间，则该命名空间中定义的名字将在每个包含了该头文件的文件中对应不同实体 43. 可以直接使用，不能使用作用域运算符 44. 若定义在最外层作用域中，需与全局作用域中的名字有所区别。可以使用全局作用域指明是全局作用域的。但是未命名空间内的成员也可以用::访问，但是只是多加一步，是会被全局作用域同名变量覆盖的。 45. 一个未命名空间也能嵌套在其它命名空间当中。 46. Note：未命名空间的成员是静态的，可以取代static命名的成员。C+的static是从c继承来的 | |