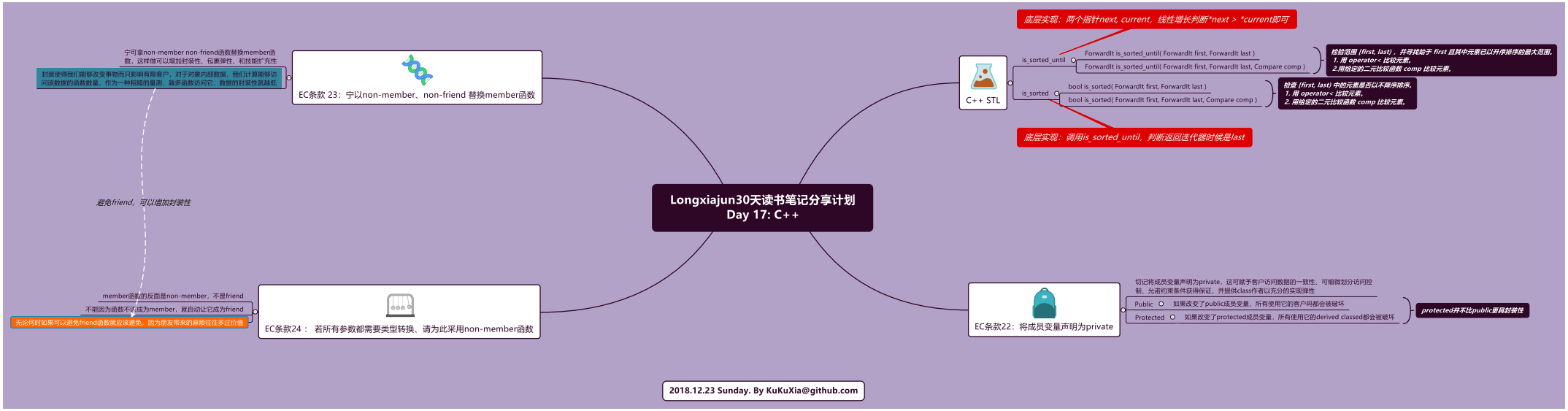
Longxiajun30天读书笔记分享计划
Day 17: C++
EC条款 23：宁以non-member、non-friend 替换member函数
宁可拿non-member non-friend函数替换member函数，这样做可以增加封装性、包裹弹性、和技能扩充性
封装使得我们能够改变事物而只影响有限客户，对于对象内部数据。我们计算能够访问该数据的函数数量，作为一种粗糙的量测，越多函数访问它，数据的封装性就越低
避免friend，可以增加封装性
EC条款24： 若所有参数都需要类型转换、请为此采用non-member函数
member函数的反面是non-member，不是friend
不能因为函数不可成为member，就自动让它成为friend
无论何时如果可以避免friend函数就应该避免。因为朋友带来的麻烦往往多过价值
C++ STL
底层实现：两个指针next, current，线性增长判断*next > *current即可
is_sorted_until
ForwardIt is_sorted_until(ForwardIt first, ForwardIt last)
ForwardIt is_sorted_until(ForwardIt first, ForwardIt last, Compare comp)
检验范围 [first, last) ，并寻找始于 first 且其中元素已以升序排序的最大范围。
1. 用 operator< 比较元素。
2.用给定的二元比较函数 comp 比较元素。
is_sorted
bool is_sorted(ForwardIt first, ForwardIt last)
bool is_sorted(ForwardIt first, ForwardIt last, Compare comp)
检查 [first, last) 中的元素是否以不降序排序。
1. 用 operator< 比较元素。
2. 用给定的二元比较函数 comp 比较元素。
底层实现：调用is_sorted_until，判断返回迭代器时候是last
EC条款22：将成员变量声明为private
切记将成员变量声明为private。这可赋予客户访问数据的一致性、可细微划分访问控制、允诺约束条件获得保证，并提供class作者以充分的实现弹性
Public
如果改变了public成员变量，所有使用它的客户吗都会被破坏
Protected
如果改变了protected成员变量，所有使用它的derived classed都会被破坏
protected并不比public更具封装性
2018.12.23 Sunday. By KuKuXia@github.com