先说一下为啥 Unit 没写
因为...
C++ 没有反射!!!
Vhat the(我一句脏话没说)
PS.Gtest 会生成一个新类继承自原来的 TestSuite, 所以无需反射, 但这招我不会

好了, 我们先说模板.
我们知道, 现今 100% 的程序 Or 函数是与一个图灵机等价的 (我只说等价, 你也可以用了 LISP 再把它变成图灵机.)
So, 它的行为很固定. 你把参数, 文件, 网络, 环境, 时钟..... 都固定, 结果就都一样.
那莫不是太无趣了??? 我们得加点可变的参数.
加什么呢??? 加一个图灵机做参数吧. 改数据参数太无趣了.
不过这个图灵机直接加入参数也不行.
算了, 让它成为这个图灵机的一部分吧.
然后??? 这个图灵机就算运用了" 模板方法"
所谓模板方法, 就是执行某些行为不确定的函数
等等, 所以我每调用一个 this 的非 Final 函数都算模板方法???
Sorry, 某种意义上, 算.
(C++ 自带 Final,233)

PS. 如果你调用的非 Final 函数是参数, 不算模板, 不过这算组合, 比继承好 233
PS2. 为啥我觉得模板后面应直接接个策略呢???

再说迭代器提到迭代器, 不得不说它的原形–指针
OOPS, 人都被吓跑了.
我们来看看, 指针能干啥?
见 Pointer.cpp
指针就是迭代器中最厉害的的, 速度最快, 支持一切操作.
反正 C++ 泛型支持 DuckTyping, 指针不是类也不要紧 (233)

(所以 C++ 迭代器没有 HasNext 和 Remove, 却有些支持向后遍历)
你可以看看 C++STL, 需要一个集合, 就要求给一个开头一个超尾 (结尾后一个), 还是静态方法
C++ 为了兼容 C 指针, 费了多少苦心啊.
不过这下就不用操心数组了. 反正数组指针 C 里是一个东西 (不懂的去补大学课)
PS. 迭代器类型我一般写 auto, 自动类型推导, 毕竟中规中举写实在太长
OOPS, 写好长了呢!!! 结束吧.