**Reading**

# 暗时间

* 为什么我们会在小摊和超时前为几块钱的交易踌躇半天，却在生活中的重大决策前浑浑噩噩、人云亦云？糟糕的判断与决策令我们的生活变得糟糕。而最关键的是我们很难学会质疑自己的判断，而总是能找到其他为自己辩解的理由。
* 人类的思维充满着各种各样的捷径，每一条捷径都是一把双刃剑。一方面，它降低了大脑的认知复杂度，有助于迅速做出在绝大部分时候都正确的判断；但另一方面，它也导致人们把大部分情况下成立的法则当成放之四海而皆准的。可以说，有多少条捷径，就有多少条谬误。
* 善于利用思维时间的人能够在重要的事情上时时主动提醒自己，将临时的记忆变成硬编码的行为习惯。
* 能够迅速进入专注状态，以及能够长期保持专注状态，是高效学习的两个最重要的习惯。
* 善于利用思维时间的人，可以无形中比别人多出很多时间，从而实际意义上比别人多活很多年。.
* 看书并记住书中的东西只是记忆，要深入理解一个事物需要推理，看到别人看不到的地方，推理的过程就是你的思维时间，即暗时间。走路、吃饭、逛街、睡觉、坐公交、出游，所有这些时间都是暗时间，通通利用起来，反刍、消化、深入理解平时看的读的东西。
* 将你的目标分隔成一个个小的里程碑，设定你自己的进度条。
* 不要过早的退出循环，要坚持！过早退出是一切失败的根因！前进的道路处处有困难，但这并不是你一个人，简单的拿起搜索引擎，借鉴前人在遇到同样的困难是怎么处理的。
* 区别他们的不是兴趣的有无，而是他们性格里有没有维持兴趣的火种一直燃烧下去的燃料。

# 自控力

## 意志力挑战

* 我要做意志力挑战：你想多做一些的事，这样做能提高你的生活质量！
* 我不要意志力挑战：你生活中最“顽固”的习惯是什么？你想要放弃的改掉的，它妨碍了你的健康、幸福、成功！
* 我想要意志力挑战：最想集中精力完成哪一项重要的长远目标？哪种当下的渴望最有可能分散你的注意力、诱导你远离自己的目标？

# 探索式测试

漫游测试法、地标测试法、极限测试法

测试人员的工作：包括在不同的环境中运行软件，使用合理的测试数据，并在较短的测试周期内尽可能多的尝试不同的输入值。

局部探索式测试法：

全局探索式测试法：

必须要把所有要做的事情按照优先级排序，然后从重要的事情做起。

根据软件的各种属性，可以将决策分为5部分：输入、状态、代码路径、用户数据、和执行环境。

## 用户输入

输入者的是由环境产生的一种刺激，该刺激导致被测试的应用程序有所响应。输入必须导致软件执行某些代码，并以某种方式作出反应。

影响因素：

* 各种输入之间的相互影响（单个输入、组合输入）
* 输入顺序的影响，枚举所有可能的恰当的顺序
* 软件接收输入后会执行的四个基本任务：接收输入、产生输出、存储数据和进行运算。

### 合法、非法输入

* 合法输入、非法输入：开发人员会忽视、不喜欢编写错误处理代码，因此测试应该重视对非法输入的地方的测试。
* 错误处理的三种方式（开发人员）：输入筛选器、输入检查、使用异常。
* 输入筛选器：1 开发人员可能对合法、非法输入值判断错误，漏掉一个非法值或把一个合法输入值归为非法都是非常严重的问题。2 是否可以绕过屏蔽器【使用某种方法可以让输入值进入系统，或者输入值进入系统后还可以修改】。
* 输入检查：if……else…… 测试时应重点检查显示的错误信息【信息会显示当前输入值被认定非法的根本原因、如何修改让他变成合法输入值，如此可提供更多的测试数据选择】
* 异常处理：错误信息提示的一般比较笼统，测试时对于异常提示可以继续使用刚才引发的异常输入数据，或稍作修改看是否会导致出错，或尝试其他一些会调用该函数的测试用例—查看结果【接连不断的引发异常很可能会让程序彻底失效】

### 常规输入、非常规输入

常规输入：正常合法的开发人员能够考虑到的输入；非常规输入：特殊字符【比如Windows的特殊键esc、shift、alt、ctrl、enter等，系统保留词，可针对不同操作系统、浏览器、编程语言、运行环境考虑】

### 默认输入、用户提供的输入

默认输入可为空、null、或默认非空值【测试可做删除默认值操作、将默认值作为临界点考虑边界值做修改，如+1 -1，字符串头部、尾部修改替换……】

### 输出指导输入选择

输入值的属性【类型、长度、大小、值……】

* 先明确自己希望软件会产生什么样的反应，然后确定哪些输入会引发相应输出，然后再测试中使用这些输入值。
* 输出可分为合法输出、非法输出。测试人员必须把精力主要放在生成各种合法输出上，这样可以保证新增功能和场景都被测试到。
* 可积极主动的从输出结果来思考整个问题，首先确定希望程序产生什么样的输出结果，然后考虑所有用户场景，看如何去生成期望的结果；或者先观察输出结果然后再选择新的输入，并保证新的输入是重新计算后的结果.
* 软件在第一次输入测试时在未被初始化的状态下产生输出，第二次输入测试是在软件已被初始化的状态下产生输出；因此测试中应至少进行二次操作，确保软件功能的正确性。
* 对软件中保存数据重复使用的情况，要做修改值再次验证，确保重新生成了新的数据。
* 考虑输入时的状态，可能同一输入值在不同的状态下会产生不同的结果。

### 软件状态

* 软件的一个状态就是状态空间中的一个点，它由所有内部数据结构的取值来唯一确定。
* 使用状态信息来帮助寻找相关的输入：测试输入的各种组合，如果两个或更多个输入在某种程度上是相关联的，那么他们应该放在一起测。

### 代码路径

代码逻辑的分支if……else…… case……switch 循环语句

### 用户数据

### 运行环境

包括操作系统、驱动程序、文件、配置、网络】带宽

## 全局探索式测试

类比游客旅游—漫游测试法。

商业区：代码的启动关闭接口，主要实现的功能（包装盒上的那些特性）

* 指南测试法：产品说明书
* 卖点测试，质疑测试：想象你是软件的销售人员，有一个可能随意打断你的演讲的客户提问“如果怎样操作会怎样？想要怎样的结果应该怎样操作？
* 地标测试法：列出软件的关键特性，确定他们之间的顺序，然后从一个地标执行到另一个地标来探索应用程序，直到访问了列表中的所有地标，在测试过程中，需要记录已经使用过哪些地标，并创建一个地标覆盖图来标识工作的进度。
* 极限测试法：以某种输入方式、数据使得软件达到最大的运算能力，所做的操作不一定有实际意义，但都是在软件允许的范围内。
* 快递测试法：跟随某一测试数据的周转流程，在各个阶段的展示作用，专注于数据
* 深夜测试：软件的启动、停止、最后的数据备份归档、维护
* 遍历测试法：选定某一目标，使用可以发现的最短路径来访问包含的所有对象。

历史区：历史遗留的代码，特性功能。缺陷修复代码

* 恶邻测试法：把缺陷数目同产品特性联系起来，可以跟踪究竟在哪些地方会出现产品缺陷，产品缺陷的地方应反复测试，还应该对临近功能使用遍历测试法进行测试。
* 古董测试：旧代码放入新的功能后可能导致问题。
* 上一版测试：对新旧版本功能特性的更新或删除，着重测试确保产品没有遗漏必需的功能。

娱乐区测试

* 页面布局、文本格式化、背景或模板修改等辅助特性，美化样式方面。
* 配角测试法：临近主要功能的地方

旅游区：某些对新用户有吸引力的功能，然而老用户不再使用。

* 特性之间的相互作用：1.有关输入的问题，这两个特性会不会处理同一个输入；2.有关输出的问题，这两个特性是否在可见的用户界面上操作同一块区域，他们会产生同一个输出吗？；3.有关数据的问题，这两个特性会操作其共享的一些内部数据？是读取共享数据还是修改？；4.有相互作用的两个特性，两个功能就会有交互，要放在一起测试。
* 收藏家测试法：手机软件的输出，收集的越多越好，测试人员要到达所有可到达的地方并把观察到的输出结果记录下来。
* 长路径测试法：到达目的地之前尽量多的在应用程序中穿行，选择长路径，把那个埋在应用程序最深处的界面作为测试目标。
* 超模测试法：只看表面，图形界面的按钮控件与期望是否相符合。
* 测一送一：同时运行同一应用程序的多个拷贝的情况。
* 取消测试法：寻找软件中的耗时操作，在程序正在加载过程中，点击取消退出、或者强制停止操作。或者在软件处理某一请求时，开始另一个同样的操作。
* 懒汉测试法：测试软件中，接收所有默认值，保持输入字段继续为空，表单中尽可能少填数据，尽可能的选择容易的操作走完整个流程。
* 破坏者：强迫软件做一些操作；掌握软件完成操作必须使用的资源；在不同程度上移除那些资源或限制那些资源。
* 反叛测试：逆向测试法—输入一些不可能的值，测试软件的错误处理能力；歹徒测试法—使软件报错，记录错误的信息；错误测试法—以错误的顺序操作流程（购物车为空时区结算）
* 强迫症测试法：对软件的一个功能进行重复操作

## 基于场景的探索式测试

### 场景

* 讲述用户故事
* 描述需求
* 演示产品功能
* 演示集成场景
* 描述设置和安装
* 描述警告和出错情形