
静态测试与动态测试



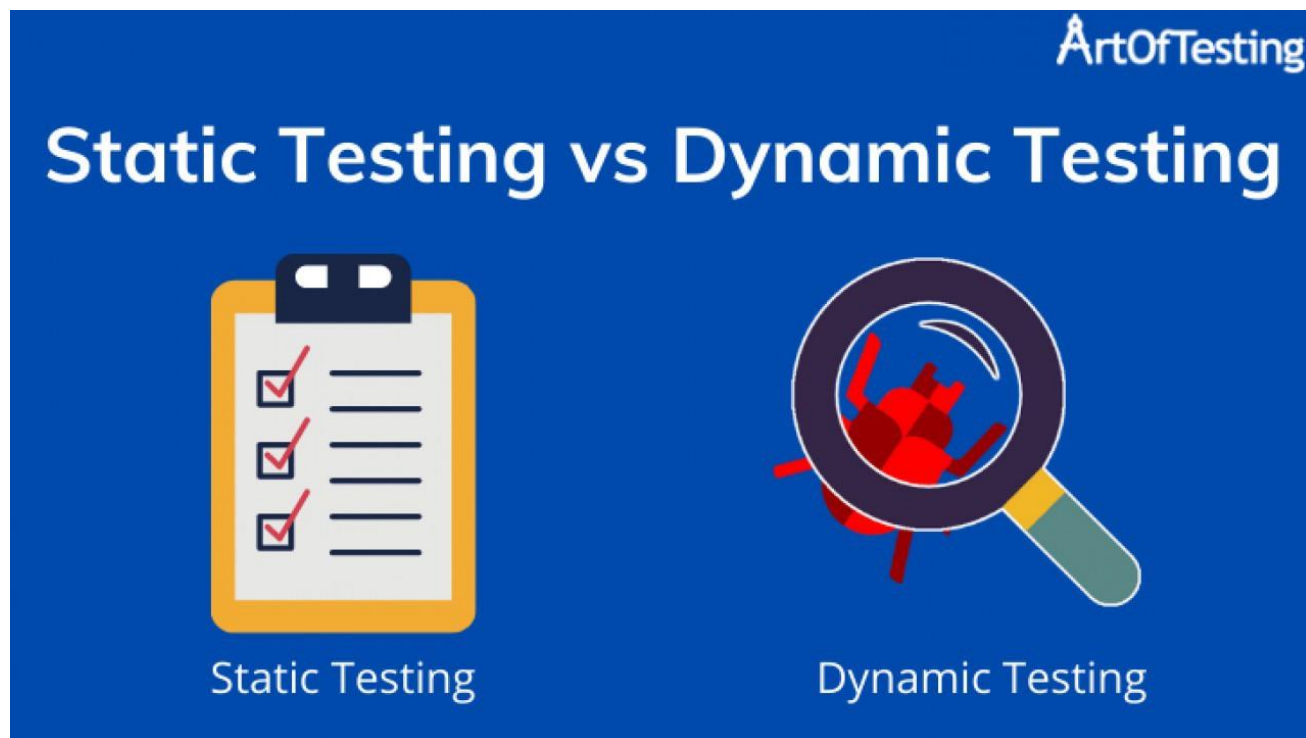


问题。

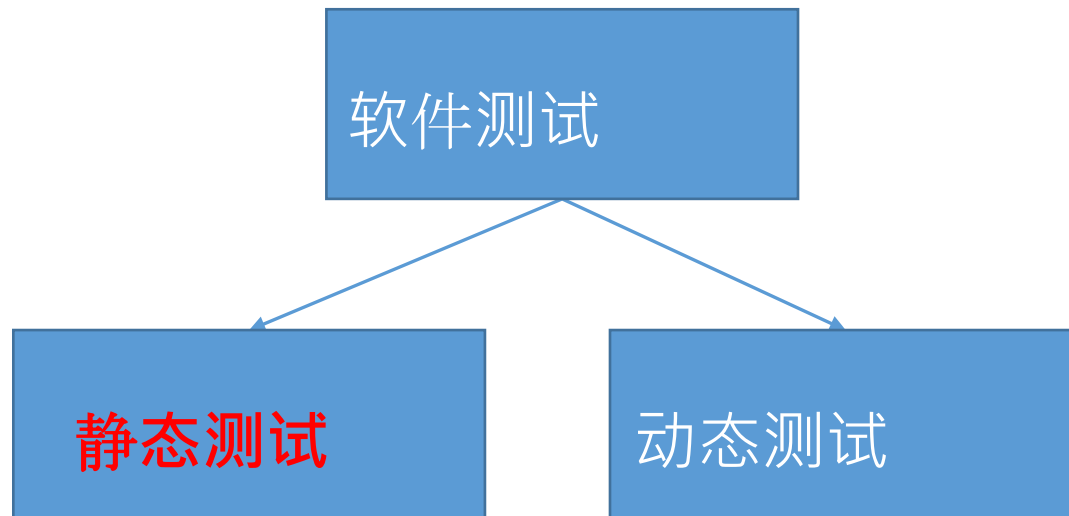
如何进行软件测试？



我们今天要讨论的内容



静态测试与动态测试



什么是静态测试



- ✓ **静态测试**是软件测试的一种类型，在这种测试中，软件应用程序**在不执行代码的情况下**被测试。
- ✓ 静态测试是一种测试技术，不涉及到**执行或运行**该程序或软件产品。
- ✓ 相反，它涉及到在软件开发生命周期（SDLC）的每个阶段**对软件进行检查**，以符合《**软件手册**》中提到的**要求或标准**。



软件产品的**需求规格**（SRS）。

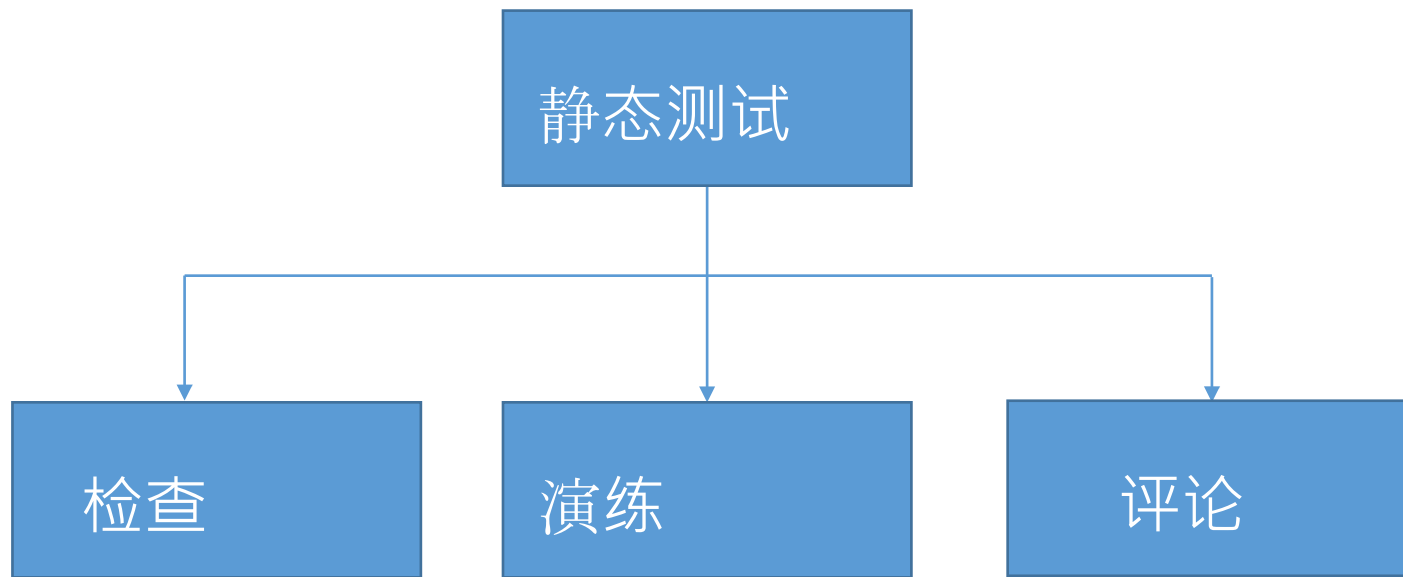


什么是静态测试

- ✓ 静态测试的主要目的是为了**提高质量**。
通过**在**软件开发过程的**早期阶段发现错误**，**来提高**软件应用的质量。
- ✓ 对代码、需求文件和文件设计进行人工或**自动审查**，
以发现错误。



静态测试的类型



检查



✓ 检查。

- ▶ 检查过程是在SLDC的早期阶段进行的。
并应用于产品的一个特定部分，如SRS、代码、产品设计等。
- ▶ 它涉及到在早期阶段对产品的各个组成部分进行人工检查。
- ▶ 检查过程的一个重要部分是使用检查表来检查程序中的常见错误。



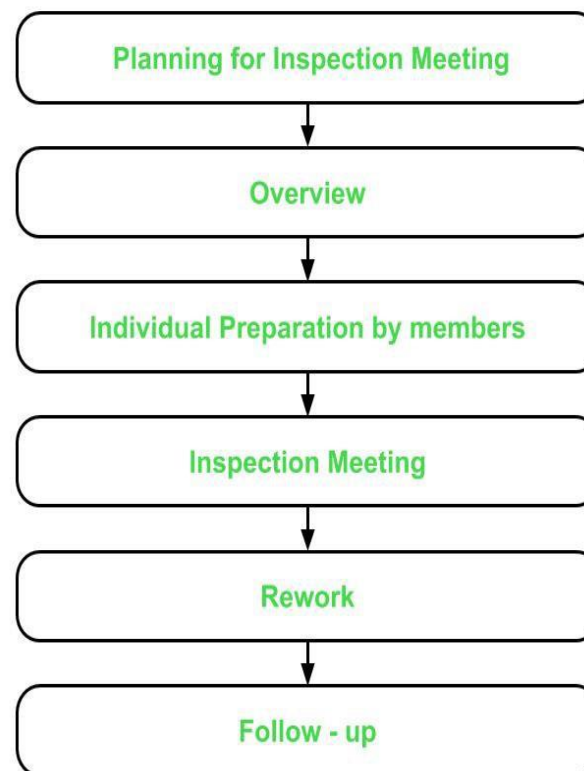
Table 3.1
Inspection Error Checklist Summary, Part I

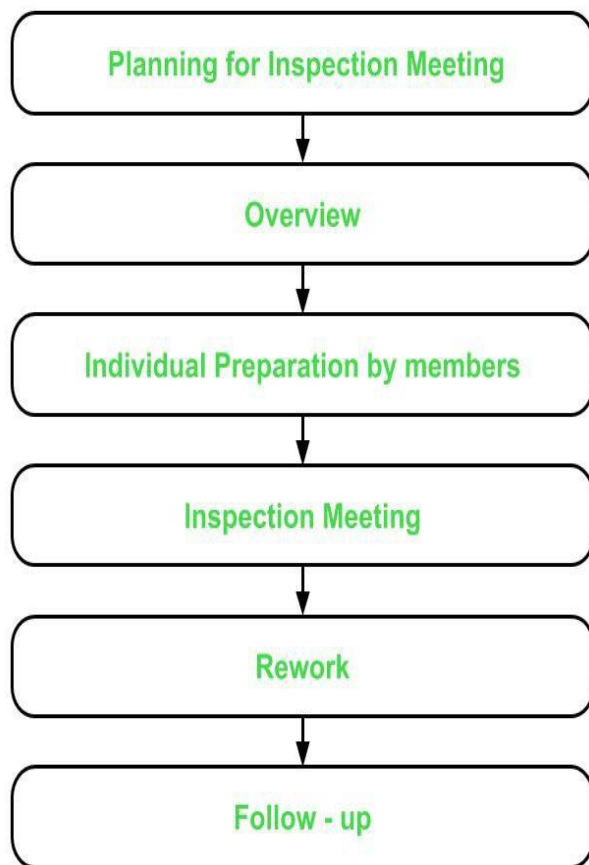
<i>Data Reference</i>	<i>Computation</i>
1. Unset variable used?	1. Computations on nonarithmetic variables?
2. Subscripts within bounds?	2. Mixed-mode computations?
3. Non integer subscripts?	3. Computations on variables of different lengths?
4. Dangling references?	4. Target size less than size of assigned value?
5. Correct attributes when aliasing?	5. Intermediate result overflow or underflow?
6. Record and structure attributes match?	6. Division by zero?
7. Computing addresses of bit strings?	7. Base-2 inaccuracies?
Passing bit-string arguments?	
8. Based storage attributes correct?	8. Variable's value outside of meaningful range?
9. Structure definitions match across procedures?	9. Operator precedence understood?
10. Off-by-one errors in indexing or subscripting operations?	10. Integer divisions correct?
11. Are inheritance requirements met?	

检查

✓ 检查。

- ▶ 软件检查过程可能包括六个阶段，即如下所示：





检查

1 检查会议的规划

- ▶ 这个阶段的重点是确定要检查的产品和这次检查的目标。
- ▶ 一位主持人--负责管理整个检查工作进程，在这个阶段被分配。
- ▶ 被指定的主持人会检查产品是否准备好接受检查。主持人还选择检查小组并分配他们的角色。
- ▶ 主持人还安排了检查会议并将所需材料分发给检查小组。

检查

一个检查小组可以包括。

- ▶ **主持人**：- 主持人领导检查过程。他的作用是决定检查的类型、方法和审查小组的组成。主持人还负责安排会议，在会议前分发文件，指导其他小组成员，为会议打气，引导可能的讨论，并储存收集的数据。



检查

一个检查小组可以包括。

- ▶ **作者**：--作为 "被检查文件 "的作者，作者的基本目标应该是尽可能多地了解有关提高文件质量的情况。作者的任务是阐明不清楚的地方，理解发现的缺陷。
- ▶ **抄写员/记录员**：- 抄写员（或记录员）必须记录每一个发现的缺陷以及会议上提出的任何建议或反馈，以便进行流程改进。



检查



一个检查小组可以包括。

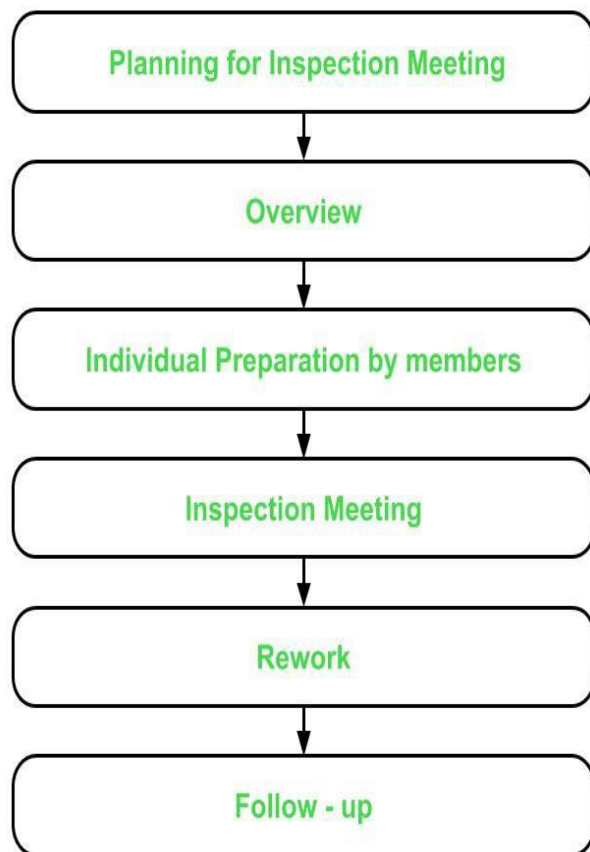
- ▶ **评审员。** 评审员的作用是根据规范、标准和领域知识检查缺陷和进一步改进。
- ▶ **经理:-** 经理参与审查，因为他或她决定检查的执行，在项目时间表中分配时间，并确定审查过程的目标是否已经达到。



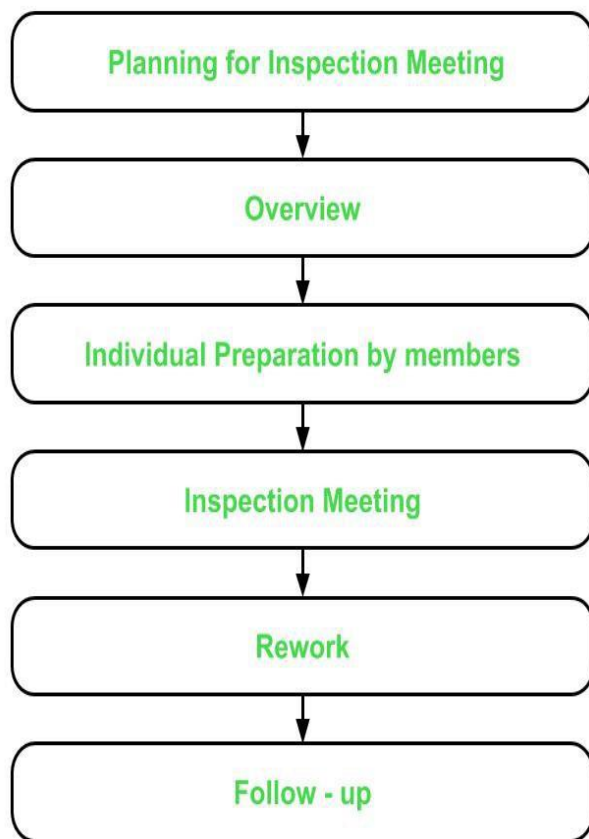


检查

2 概述



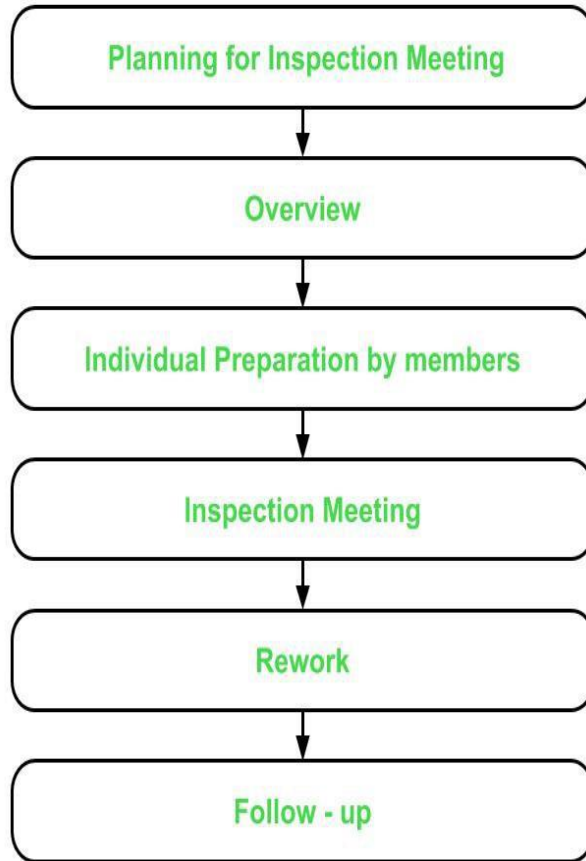
- ▶ 在这个阶段，检查小组会得到检查会议的所有背景资料。
- ▶ 作者--是负责开发产品的编码员或设计师，介绍他对产品的逻辑和推理，包括产品的功能、预期目的和开发产品时使用的方法或概念。
- ▶ 要确保检查小组的每个成员都了解并熟悉将要举行的检查会议的目标和目的。



检查

3 成员的个人准备

- ▶ 在这一阶段，检查小组的成员通过研究前面几个阶段提供的材料，单独为检查会议做准备。
- ▶ 团队成员识别产品中的潜在错误或bug，并将其记录在日志中。
- ▶ 该日志最后提交给主持人。
- ▶ 然后，版主将所有从成员那里收到的日志进行汇编，并将其副本发送给作者。



检查

4 检查会议

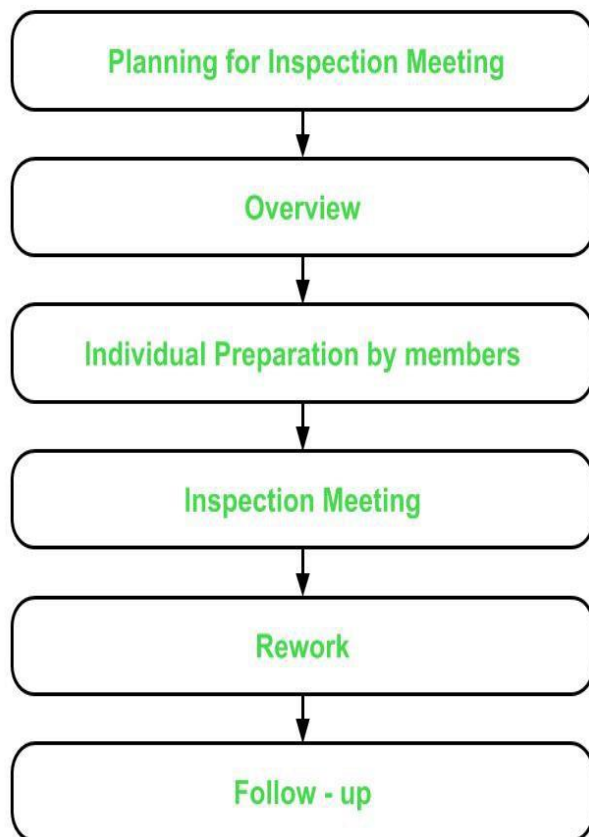
- ▶ 这一阶段涉及作者对团队成员在汇编的日志中提出的问题进行讨论。
- ▶ 委员们就提出的问题是是否是错误作出决定。
- ▶ 主持人总结会议并提供会议摘要--这是一份在产品中发现的错误清单，将由作者解决。

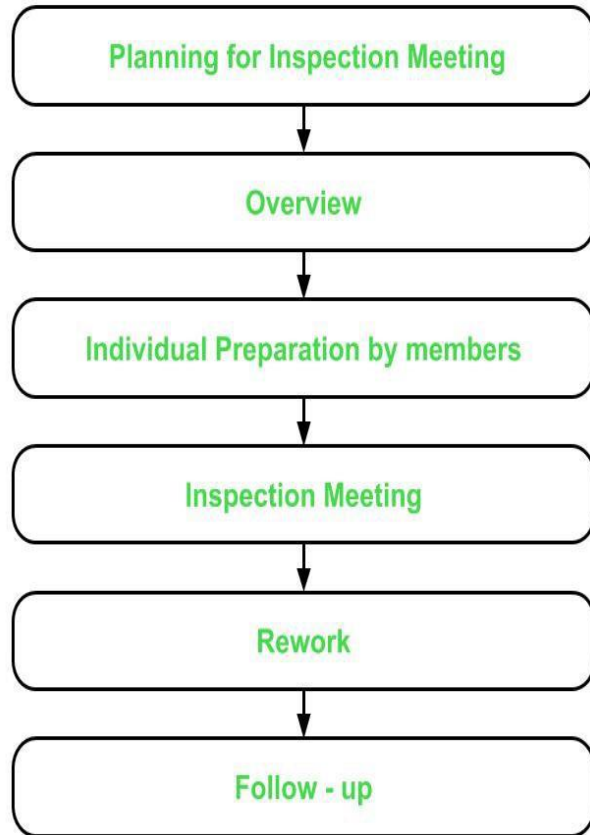


检查

5 重新制作

- ▶ 作者根据主持人在上一阶段提出的摘要清单进行返工。
- ▶ 作者修复了所有的错误并向版主报告。



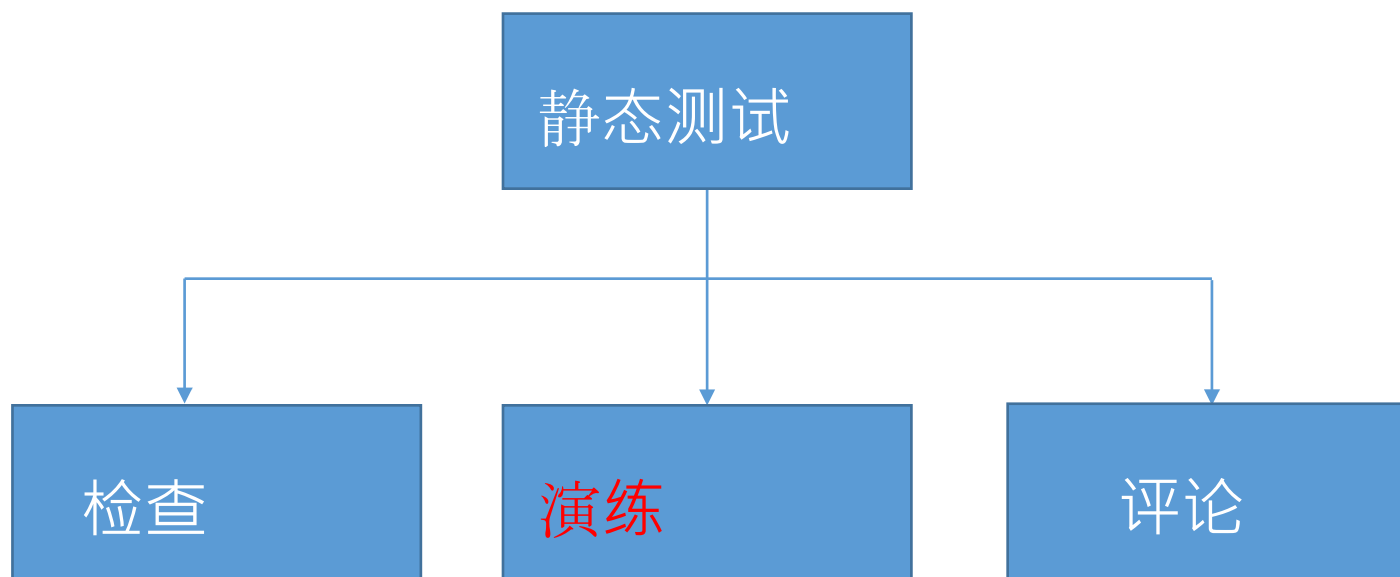


检查

6 后续行动

- ▶ 主持人检查是否所有的错误都已解决。
- ▶ 然后，主持人准备了一份报告。
- ▶ 如果所有的错误都得到了修复和解决，主持人就会发布该文件。
- ▶ 否则，未解决的问题将被添加到报告中，并安排另一次检查会议。

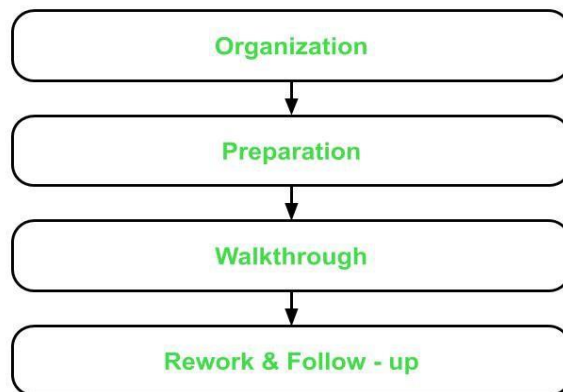
静态测试的类型



静态测试的类型

✓ 走过。

- ▶ 这种类型的静态测试不那么正式，也不那么严格。
- ▶ 与检查过程相比，它有一个更简单的过程。它包括以下四个步骤。





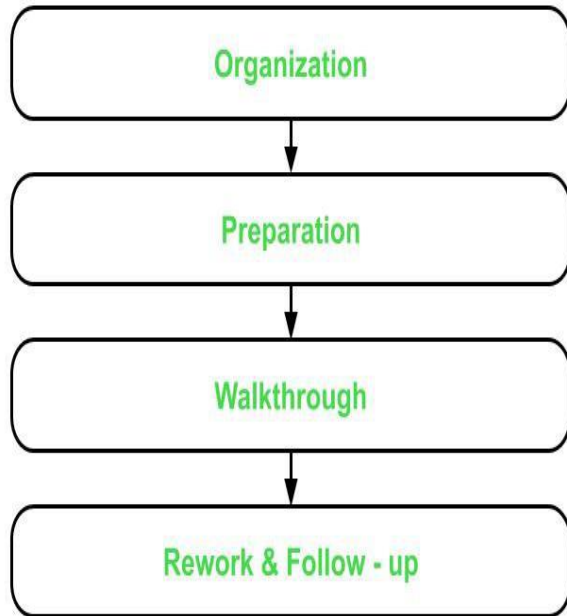
演练

1个组织

这一步涉及为选定的演练团队分配角色和责任。

► 该小组可由以下成员组成。

- a) 协调员--组织和协调所有成员进行与演练有关的活动。
- b) 介绍者--介绍要检查的项目。
- c) 抄写员--记下成员提出的所有问题和建议。
- d) 测试员--发现要检查的项目中的缺陷或错误。
- e) 作者。



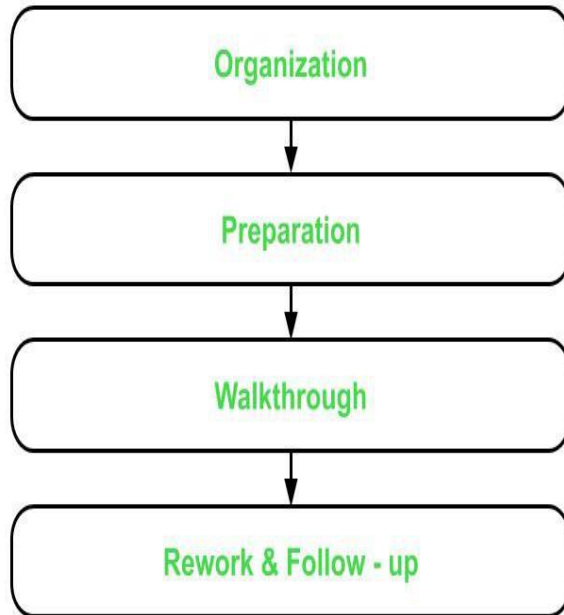


演练

1个组织

对其他参与者的建议包括

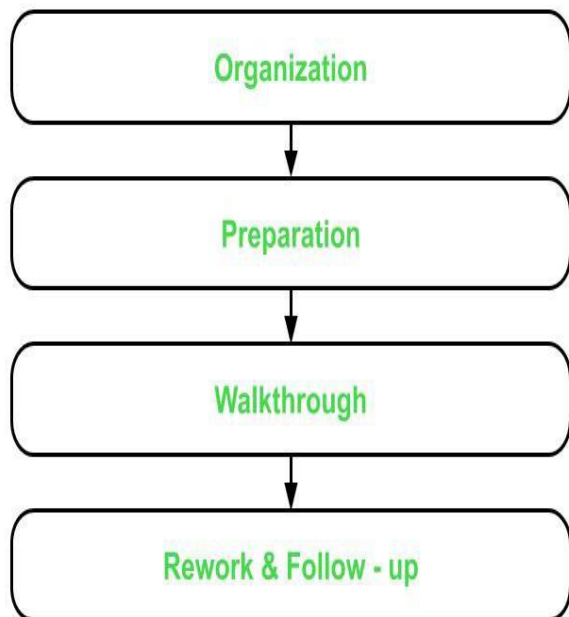
- (1) 一个经验丰富的程序员。
- (2) 一个编程语言专家。
- (3) 一个新的程序员（以提供一个新鲜的、不偏不倚的观点）。
- (4) 最终将维持该计划的人。
- (5) 一个来自不同项目的人。
- (6) 与程序员来自同一个编程小组的人。



演练

2 准备

在这一步，重点在于为演练做准备，这可能包括思考测试产品所需的基本测试案例。

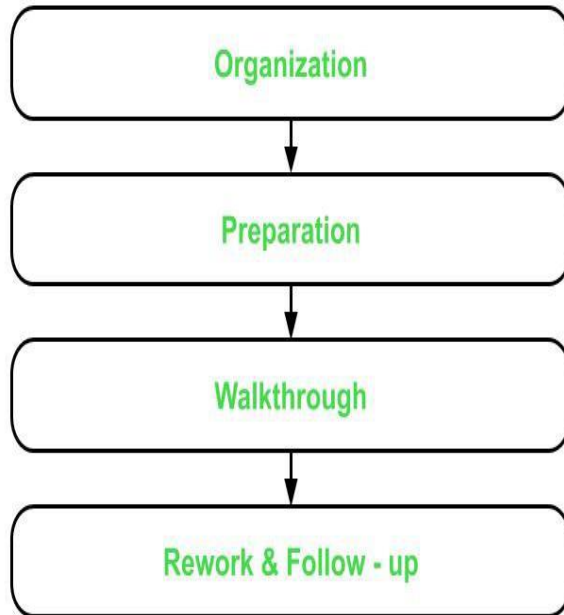




演练

3 演练

- ▶ 演练是由一个测试人员进行的，他带着一小套测试用例来参加会议。
- ▶ 测试人员在精神上执行测试用例，并将结果记在纸上或演示媒体上。

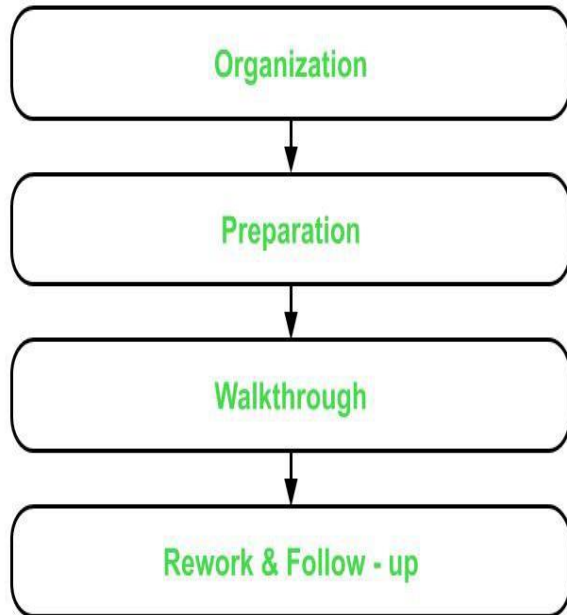




演练

4 返工和跟进

- ▶ 这一步与检查过程的最后两个阶段相似。
- ▶ 如果没有发现错误，那么产品就被批准发布。
- ▶ 否则，错误就会被解决，并再次进行结构化演练。
 -





问题。

检查和穿行的区别是什么？





检查与演练

Inspection

Walkthrough

It is formal.

It is informal.

Initiated by project team.

Initiated by author.

A group of relevant persons from different departments participate in the inspection.

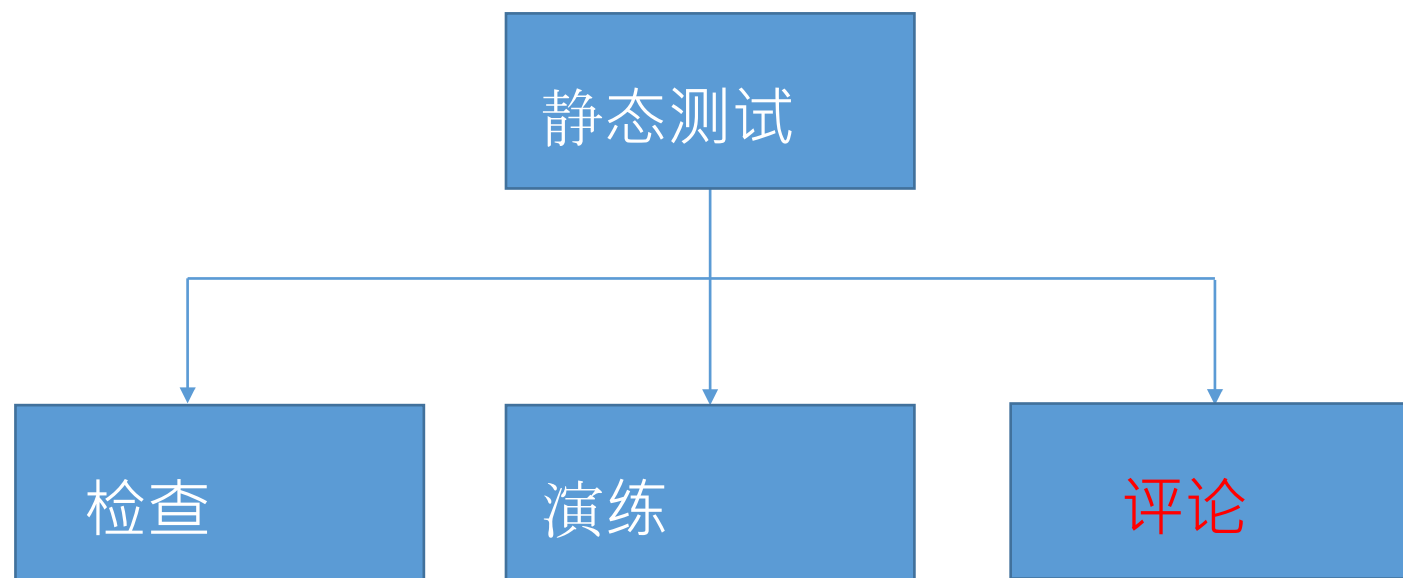
Usually team members of the same project take participation in the walkthrough. Author himself acts walkthrough leader.

Checklist is used to find faults.

No checklist is used in the walkthrough.



静态测试的类型



评论




- ✓ **审查**是用来评估和评价产品的，通过检查其符合开发**标准、准则和规范**。
- ✓ 它没有一个确定的过程，大部分的工作是由主持人进行的，如下文所述。
 - 1 主持人收集并向所有团队成员分发材料和文件。





评论

2 主持人还准备了一套指标来评估产品
关于规格和已经建立的标准和准则，

- 一致性
 - 文件
 - 遵守标准
 - 完整性
 - 问题定义和要求
- 



评论

3 结果被记录在一份文件中，其中包括两个缺陷作为以及建议。

4 最后，缺陷得到解决，建议被考虑用于改进产品。



检查 VS 演练 VS 评论

正式的

非正式什么是正式？

你是否了解测试员的工作是什么？
静态测试？

- ☐ A 了解
- ☐ B 不懂
- ☐ C 明白吗？



提交

常见的编程错误

- 1、内存泄漏的故障（内存泄漏）。
- 2、数组越界故障（**Out of Bounds Array Access**）
- 。3、使用未初始化变量故障（**Uninitialized Variable**）。
- 4.空指针使用故障（**NULL**指针解除引用）。
- 5.非法计算类故障（非法操作）。
- 6.死循环结构（死循环）。
- 7 资源泄漏（资源泄漏）。
- 8 并发故障（Concurrency）
9. 安全漏洞（**Security vulnerability**）
- 10， 疑问代码故障（**Confused code**）



- 高能预警
 - 前方进入程序猿模式
- 高能量警告！！！！。
 - 进入编程器模式

A problem has been detected and windows has been shut down to prevent damage to your computer.

The problem seems to be caused by the following file: SPCMDCON.SYS

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:

*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)

*** SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, DateStamp 3d6dd67c

1 记忆泄漏

- 请求的内存没有被释放。
 - listrec *add_list_entry(listrec *entry,int value){
 - listrec *new_entry=(listrec *)malloc(sizeof(listrec))。
 - if(!new_entry)
 - 返回NULL
 - 做点什么。。。。
 - 返回NULL;
 -}
- 请求函数和释放函数不匹配
 - str=malloc(10); ... ; delete(str); ...
 - str=new(10); ... ; free(str); ...

- malloc matchs free ; new matchs delete

- 指向请求的内存的指针已经改变了

- `char *p=malloc(10)。`
- `++p;`
- `free(p);`//没有原始地址

- 反复释放

- `char *str=new char[100];`
- `char *p;`
- `str="abc"。`
- `p=str;`
- 删除字符串。
- 删除`p;`//将一个地址释放两次

1 记忆泄漏

在计算机科学中，当计算机程序错误地管理内存分配，使不再需要的内存没有被释放时，就会发生内存泄漏。

当一个对象存储在内存中但不能被运行的代码访问时，也可能发生内存泄漏。

因为它们可以在应用程序运行时耗尽可用的系统内存，所以内存泄漏往往是软件老化的原因或促成因素。

2 界外数组访问

- `int data[10];`
- `for(i=0; i<=10; i++){data[i]=...}。`
- `#define MAXPATTERN 3`
- `#define MAXFILELEN 12`
- `char l_Pat[MAXPATTERN+1][MAXFILELEN+1]。`
- `.....`
- `memset(l_Pat, 0, 20*(MAXFILELEN+1))。`

2 超出界限的阵列访问

- 程序员不小心访问了数组中任何一个超出边界的索引，导致未定义行为。
- 程序可以访问一些不属于自己的内存，这可能会导致程序崩溃，如分段故障。

3 未初始化的变量

- char c;
 - while(c!='\n' && c=getchar() && c!=EOF) ...。
-
- listrec *new_entry=NULL。
 - new_entry->value=value。

3 未初始化的变量

- 未初始化的变量是指一个已经声明的变量，但在使用之前没有被设置为一个明确的已知值。
- 它将有一些价值，但不是一个可预测的价值。因此，它是一个编程错误，也是软件中常见的错误来源。

4 解除对NULL指针的引用

- DirSettingLink *l_Link ;
while(l_Link)
{
 g_DirRoot = l_Link->next; l_Link->next = NULL;
 删除l_Link->m_node.m_dirname; 删除l_Link。
 l_Link = g_DirRoot;
}

4 解除对NULL指针的引用

NULL指针解除引用发生在应用程序解除引用一个它期望有效的指针，但却是NULL，通常导致崩溃或退出。

因为空指针并不指向一个有意义的对象，试图解除引用（即访问存储在该内存位置的数据）一个空指针通常（但并不总是）会导致运行时错误或程序立即崩溃。

5 非法操作

`int x=0;`

`int y=1;`

`int m= -1;`

`int z。`

`z=y/x。`

`z=logx (Y) ;`

`z=sqrt (m) 。`

5 非法操作

非法操作是指计算机不能执行或不理解的命令。

计算机将向用户返回一个错误，提醒她执行命令失败的事实，而用户正在运行的程序可能被操作系统终止。

6 死循环

```
for(i=1; i<=100; j++ )
```

```
for(i=1;i++)
```

```
for(i=1;i!=100;i=i+2)
```

- 使用单一的等号来检查平等性
- `char x='Y'。`
- `while(x='Y') {`
- `//...`
- `cout<<"继续？(Y/N)"。`
- `cin>>x。`
- `}`
- `//"为什么我的循环永远不会结束？"`

6 死循环

死循环/无限循环/无尽循环是计算机程序中的一连串指令，这些指令**无休止地循环**，要么是由于循环没有终止条件，要么是有一个永远无法满足的条件，要么是导致循环重新开始。

7 资源泄漏

在计算机科学中，资源泄漏是计算机程序消耗资源的一种特殊类型，即程序不释放其获得的资源。

这种情况通常是由程序中的一个错误造成的。典型的资源泄漏包括内存泄漏和句柄泄漏，特别是文件句柄泄漏，尽管内存通常与其他资源分开考虑

问题：你的键盘上有多少个键可以同时按下去？

8 并发

在多用户环境中不受控制地执行并发事务会导致各种问题。

当两个访问相同数据项的事务以这样的方式交错操作时，这个问题就会发生，一个事务会在另一个应用任何更新之前访问数据。

同步问题

- 全球账户
- 1 : 增加=100。
- 2: $\text{count} = \text{增量} + \text{ACCOUNT}$;
- 3: $\text{ACCOUNT} = \text{count}$;

同步化的解决方案。 锁定

- 全球账户
- 1: 锁定ACCOUNT。
- 2 : 增加=100。
- 3: $\text{count} = \text{increasement} + \text{ACCOUNT}$;
- 4: $\text{ACCOUNT} = \text{count}$;
- 5: 释放ACCOUNT

僵局

1. 全球ACCOUNT1, ACCOUNT2
2. 锁定ACCOUNT1。
3. $ACCOUNT1 = \text{getchar()} + ACCOUNT1;$
4. 锁定ACCOUNT2。
5. 账户2=账户1+账户2。
6. 释放ACCOUNT2。
7. 释放ACCOUNT1。

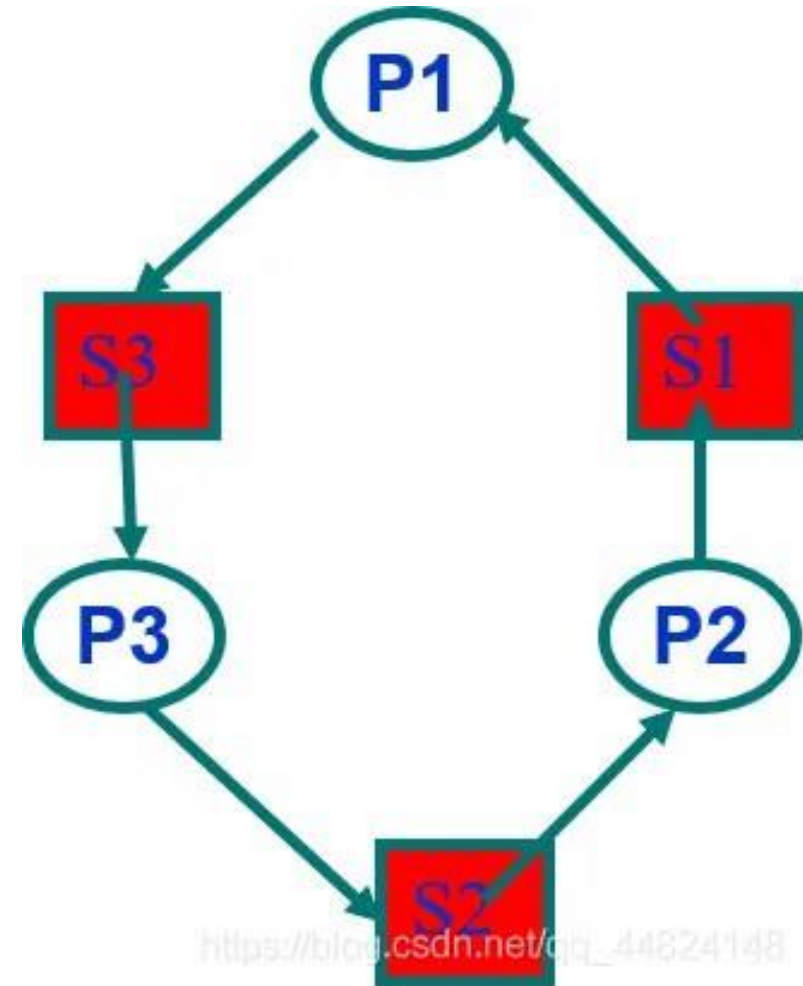
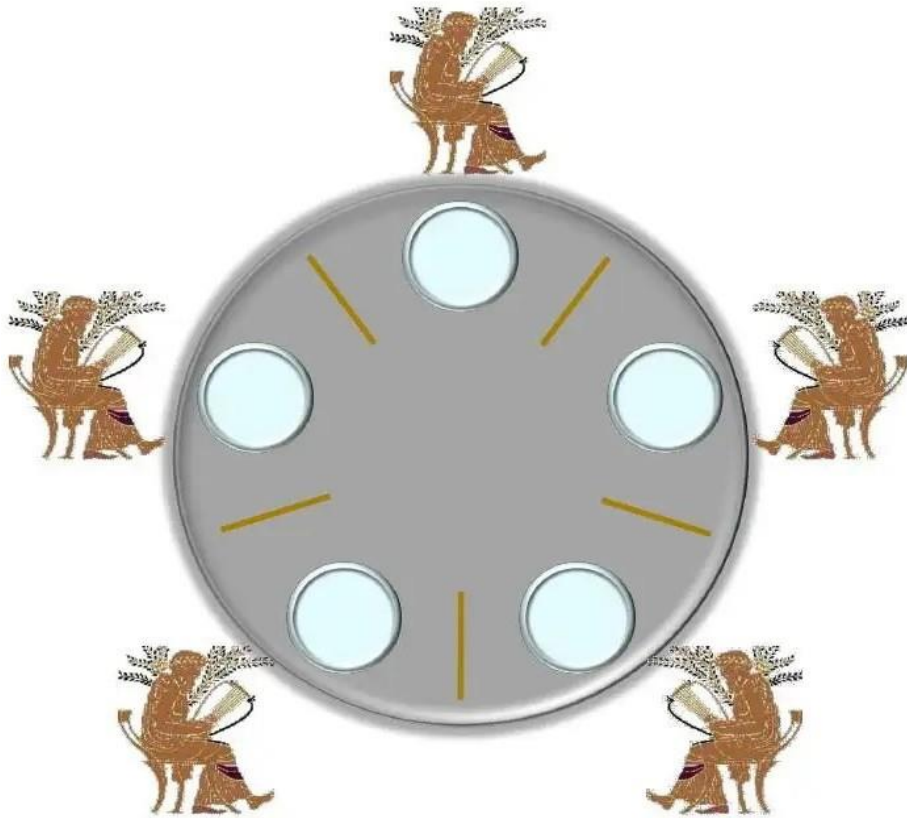
1. 全球ACCOUNT1, ACCOUNT2
2. 锁定ACCOUNT2。
3. $ACCOUNT2 = \text{getchar()} + ACCOUNT2;$
4. 锁定ACCOUNT1。
5. 账户1=账户2+账户1。
6. 释放ACCOUNT1。
7. 释放ACCOUNT2。

僵局

在并发计算中，死锁是指一些实体组的任何成员都不能继续进行的任何情况，因为每个成员都在等待另一个成员，包括它自己，采取行动，如发送消息或更常见的是释放锁。

死锁是多处理系统、并行计算和分布式系统中常见的问题，因为在这些情况下，系统经常使用软件或硬件锁来仲裁共享资源和实现进程同步。

DeadLock



9 安全漏洞

安全漏洞被定义为计算机组件或系统配置的非预期特性，它使不利事件或损失发生的风险成倍增加，其原因是意外暴露、蓄意攻击或与新系统组件的冲突。



SANGFOR

登录

用户名:

密 码:

登 录

- 例子：SQL注入
- `SELECT * FROM admin WHERE username='$name' AND password = '$passwd'.`

你认为SQL代码是正确的吗？

- ☐ A 有一些问题，没有问题
- ☐ C 我不知道

提交

永远不要相信用户

- SQL注入可以在没有密码的情况下以超级管理员的身份访问系统
- `SELECT * FROM admin WHERE username='admin'or 1='1'AND password='*****'`



用户名

admin' or 1='1



密码

●●●●●●●●



验证码

MXEK

验证码输入错误!

登录

9 安全漏洞

有风险的操作 例子：

```
random random()
```

```
srand((unsigned) time(NULL))
```

10 混乱的代码

有些代码是很难理解的。

这类问题不一定会导致系统错误，但可能会潜在地导致错误或导致低性能。

- 虚数f(int b){
- Char a = b; //?
- }

- `Void f(double b){`
- `如果(b==0){`
- `做点什么。`
- `}`

- Void f(double b){
- 如果(b==0){
- 做一些事情 ; //可能永远不会被执行
- }

小测

验

```
for (i=0; i<N; i++)  
{  
    如果 (条件)  
        DoSomething()。  
    否则  
        DoOtherthing()。  
}
```

```
如果(条件)  
{  
    for (i=0; i<N; i++)  
        DoSomething()。  
}  
否则  
{  
    for (i=0; i<N; i++)  
        DoOtherthing()。  
}
```

这两个代码做同样的事情？哪一个运行得更快？

这两个代码做同样的事情？哪一个运行得更快？

- ☐ A 一样的 不一样
- ☐ B 的
- ☐ C 左边的更快 右边的更快
- ☐ D

提交

- 一个不言而喻的规则

- 软件编程中的 "666" 规则

- 前6行代码通常用于检查错误（function、reference、I/O、循环、memory操作等）。
 - 60%的代码用于容错
 - 每天有6小时用于检查错误
 - 6行，而不是1行

静态测试

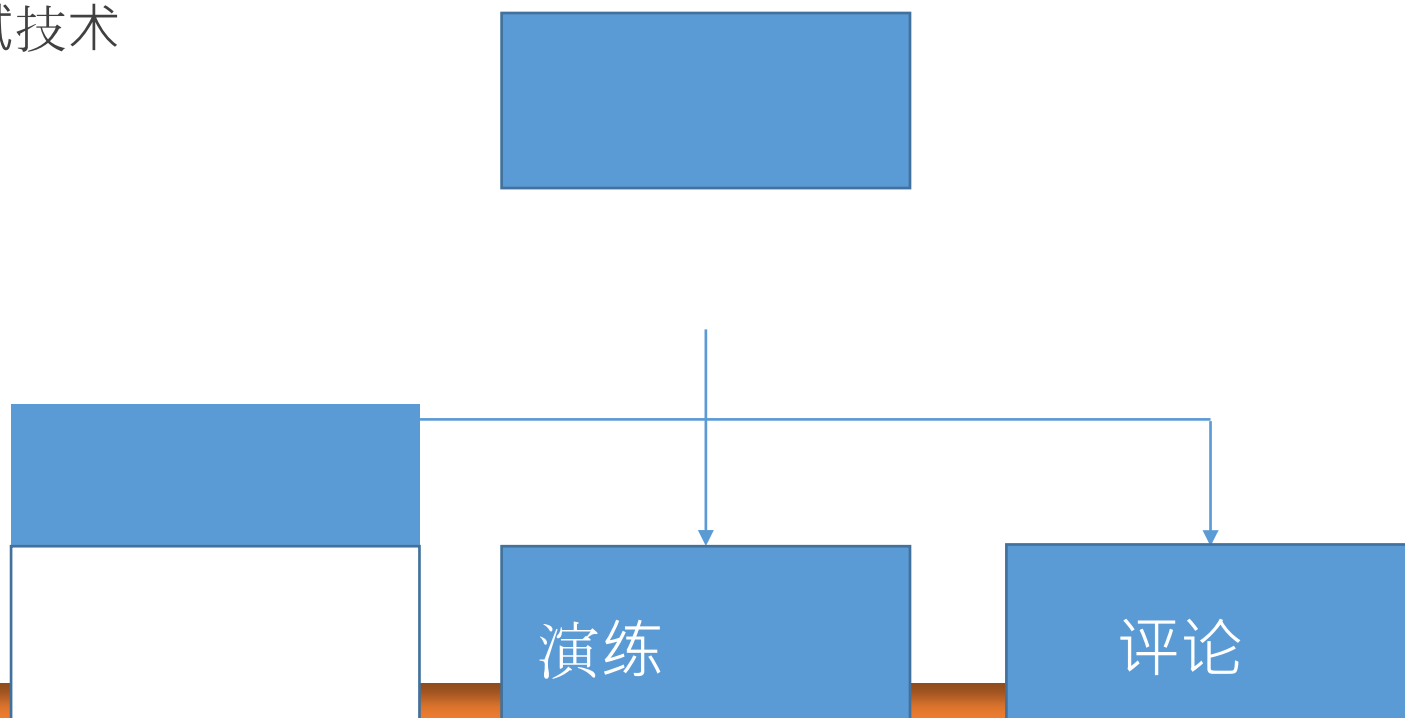
查看代码/文件并在检查列表中找到错误

总结 y

✓ 什么是静态测试？

静态测试是软件测试的一种类型，在这种测试中，软件应用是
不执行代码。

✓ 静态测试技术



总结 y

常见的编程错误

- 1、内存泄漏的故障（内存泄漏）。
- 2、数组越界故障（Out of Bounds Array Access）。
- 3、使用未初始化变量故障（Uninitialized Variable）。
- 4.空指针使用故障（NULL指针解除引用）。
- 5.非法计算类故障（非法操作）。
- 6.死循环结构（死循环）。
7. 资源泄漏（资源泄漏）。
8. 并发故障（Concurrency）
9. 安全漏洞（Security vulnerability）
10. 疑问代码故障（Confused code）