软件质量保证与测试

Software Quality Assurance and Testing

第 4 章 白盒测试

4.6 程序插桩和调试

金陵科技學院

程序插桩技术

在软件的动态测试中，程序插桩是一种基本的测试手段，有着广泛的应用。

程序插桩是借助于往被测程序中插入操作，来实现测试目的一种方法，即向源程序中添加一些语句，来实现对程序语句的执行、变量的变化等情况进行监测和检查。

最简单的插桩是在程序中插入输出语句，以监测变量的取值或者状态是否符合预期。断言是一种特殊的插桩，是在程序的特定部位插入语句用来检查变量的特性。

程序插桩技术

1. 设计插桩

在程序中插桩是需要付出成本的，包括插入代码的成本和用完之后去掉这些代码的成本，所以程序插桩并不是随意进行的。对程序进行插桩时，应当考虑以下问题：

① 需要通过插桩探测哪些信息？

② 在代码的什么部位设置探测点？

典型的探测点如：每个程序块的第1个可执行语句之前；for,do-while,do until等循环语句处；if-else等条件语句各分支处；输入语句之后；函数、过程、子程序调用语句之后；return语句之后；goto语句之后等。

程序插桩技术

③需要设置多少个探测点？

应当优选插桩方案，使得需要设置的探测点尽可能少。

④需要插入哪些语句？

在进行程序插桩时，除了插入输出语句之外，还可以在程序中特定部位插入某些用以判断变量特性的语句，程序执行时这些语句会自动判断变量取值或状态是否符合预期要求。

程序插桩技术

2. 插桩的作用

想要了解一个程序在某次运行中所有可执行语句被覆盖的情况，或是每个语句的实际执行次数等，都可以利用插桩技术来实现。

插桩的作用可以是：

① 语句覆盖统计

② 分支覆盖统计

③ 判断变量的动态特性

程序插桩技术

3. 程序插桩实例

有一段程序，功能是求两个数的最大公约数，程序代码如下：

```
int  gsd (int X, int Y)
{     int Q=X;
      int R=Y;
      while(Q!=R)
      {     if(Q>R)
                  Q=Q-R;
            else R=R-Q;
      }
      return Q;
}
```

程序插桩技术

在对这段程序进行测试时，可以进行插桩，以检测程序中各个节点的执行次数。插桩采用的语句可以如下：

C(i)=C(i)+1，i=1，2，…n

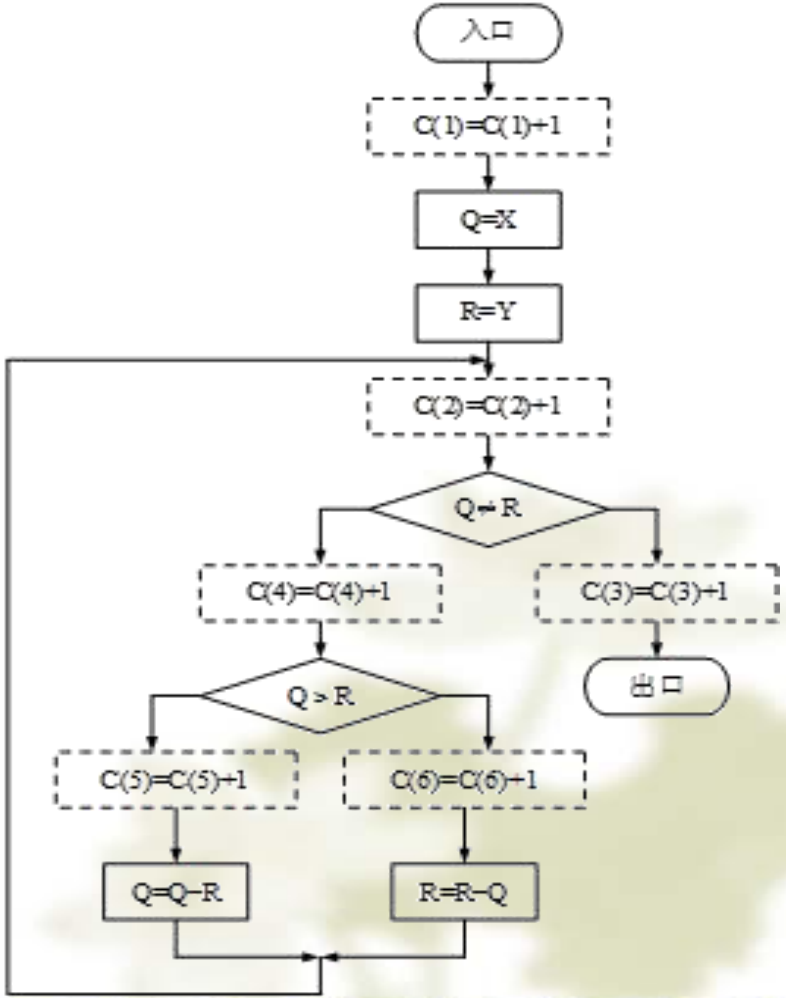
即程序每经过该位置节点的时候，计数器就会加1，最终计数器等于几，就意味着程序执行时经过了该节点几次。插桩后的程序流程图如下图所示。

程序插桩技术

图中虚线框中的内容并不是源程序的内容，而是为了记录该位置的执行次数而插入的。虚线框中的代码就是为了完成计数，形式为：

C（i）=C（i）+1 ；

i = 1，2，3，……，6 ；



程序插桩技术

当然还需要在程序的入口处插入对计数器C(i)初始化的语句，并在出口处插入输出这些计数器C(i)结果的语句，才能构成完整的插桩，这样就能记录并输出在程序中插桩的各个位置节点的实际执行次数。

从插桩后的程序流程图不难看出，如果测试完成后所有的C(i)均大于0，则测试实现了语句覆盖、判定覆盖和条件覆盖等。

程序调试

1. 测试和调试

软件开发过程中，需要对程序进行测试和调试，测试和调试的含义完全不同。

测试是去发现软件中的问题的过程，是一个可以系统进行的有计划的过程，可以事先确定测试策略，设计测试用例，可以把测试结果和预期的结果进行比较。测试发现的不一定是错误本身，而可能只是错误的外部征兆或表现。

程序调试

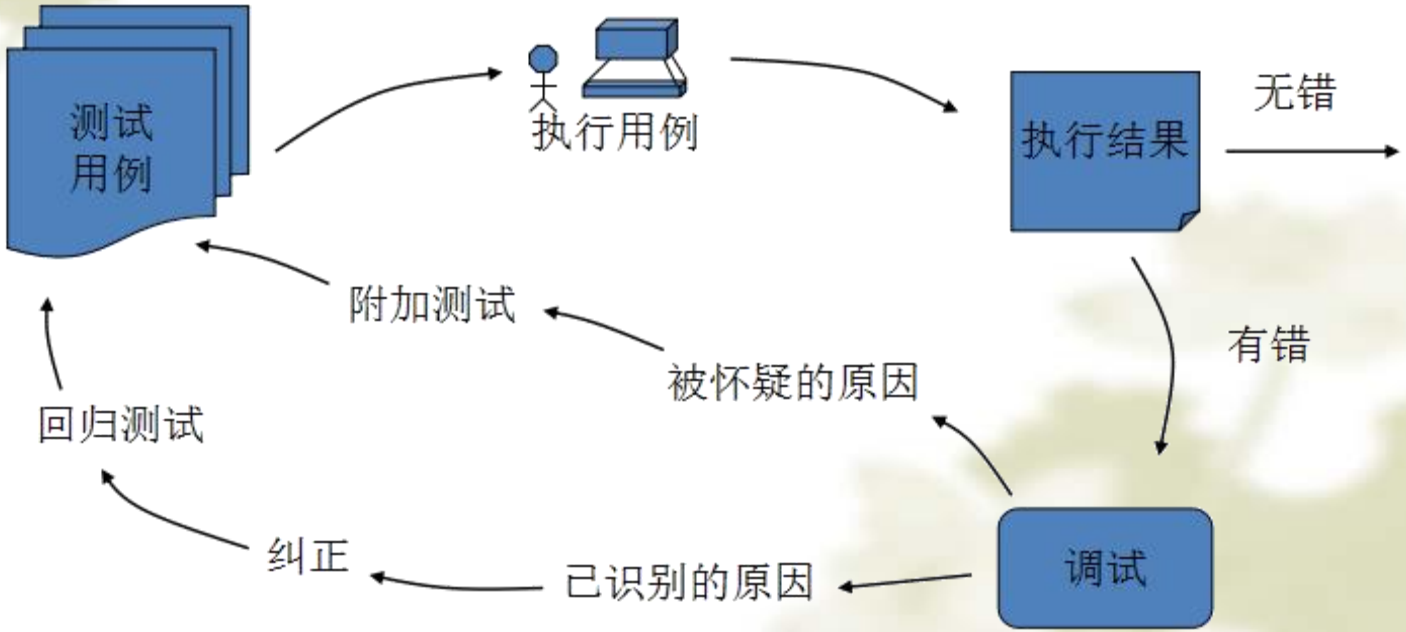
调试是在发现错误之后消除错误的过程。调试应充分利用测试结果和测试提供的信息，全面分析，先找出错误的根源和具体位置，再进行修正，将错误消除。

从职责上说，测试工作只需要发现错误即可，并不需要修正错误，而调试的职责就是要修正错误。软件开发者有时需要同时肩负这两种职责，对自己开发的程序进行测试，发现问题，并对其进行调试，修正错误。

程序调试

2. 调试的过程

调试的过程如下图所示。



程序调试

调试时如果已经识别或者说找到测试中所发现错误的原因，就可以直接予以修正，然后进行回归测试。如果没有找到问题的原因，那么可以先假设一个最有可能的错误原因，然后通过附加测试来验证这样的假设是否成立，直到找出错误原因为止。

程序调试

有时调试工作难度很大，原因很多，例如：

- 症状和原因可能相隔很远，高度耦合的程序结构加深了这种情况
- 症状可能是由误差引起的，程序本身并没有错误
- 症状可能和时间有关
- 症状可能在另一错误改正后消失或暂时性消失
- 症状由不太容易跟踪的人工错误引起
- 很难重新完全产生相同输入条件(如输入顺序不确定的实时应用系统)
- 症状可能是时有时无(耦合硬件的嵌入式系统常见)
- 症状可能时由分布在许多不同任务中的原因引起的

程序调试

有时程序员会因为在调试程序时，找不到问题出在哪，而让软件开发工作陷入困境。正是由于调试工作很难，具有一定规模和复杂度程序，有些问题发现后，不太容易找到出错的具体位置，所以调试工作是程序员能力和水平的一个重要体现。

程序调试

程序调试能力因人而异，在某种程度上可以说是跟人的个性和天赋有关，有的程序员非常善于调试程序，有的则不具备这样的能力，即使是具有相同教育背景和工作背景的程序员，他们的程序调试能力也可能有很大差别。

在某种角度上来说，调试是一种很容易让人感到沮丧的编程工作。尤其让人烦恼的是，你认识到自己犯下了错误，因为程序出错了，但你却找不到错误出在哪？自我怀疑、项目进度要求等引起的高度焦虑，会增加调试工作的难度。

程序调试

3. 调试的方法

调试的方法主要有：回溯法(Backtracking)、原因排除法(Cause elimination)、归纳法（Induction)、演绎法(Deduction)。

这些方法的具体实施可以借助调试工具来辅助完成，例如带调试功能的编译器、动态调试辅助工具“跟踪器”、内存映像工具等。

程序调试

3. 调试的方法

回溯法是指，从程序出现不正确结果的地方开始，沿着程序的执行路径，往上游寻找错误的源头，直到找出程序错误的实际位置。

例如，程序有5000行，测试发现最后输出的结果是错误的，采用回溯法，可以先在第4500行插桩，检查中间结果是否正确，若正确，则错误很可能发生在 第4500—5000行之间。若不正确，则在第4000行插桩，依此类推，直到找出程序错误的具体位置。

程序调试

4. 重现缺陷

软件缺陷是存在于软件中的那些不希望或不可接受的偏差，只要缺陷客观存在，那么任何缺陷都是可重现的。软件缺陷并不是间歇发生的，即使发生的条件很多，出现的概率很小，但一旦满足了确切的条件，缺陷还是会再次重现。

不管是软件测试还是调试，都需要让因软件缺陷重现。在软件测试时，这样做是为了确认缺陷确实存在，并确切的描述缺陷，而在调试时，这样做是为了根据重现的缺陷，找到出错的原因和具体的位置，以便修正。

程序调试

有的缺陷很隐蔽，要重现有一定难度，需要符合特定的条件，在试图重现缺陷时，需要考虑以下一些情况：

- 竞争条件：处理延时、处理加快
- 被遗忘的细节
- 缺陷造成的影响会导致其无法重现
- 例如破坏了原始数据
- 缺陷是依赖于内存的
- 仅会在初次运行时出现的缺陷

程序调试

- 因数据错误导致的缺陷
- 由一些其它问题附带引起的缺陷
- 间断性硬件故障
- 缺陷依赖于时间
- 缺陷依赖于资源
- 缺陷是由于环境变量长期积累造成的
- 代码中的特殊分支

本节内容就讲到这里，谢谢，再见！

金陵科技學院