第三部分 软件测试应用

3.1 单元测试

内容回顾

- 对变量的测试
  - 基本概念：定义变量，使用变量，定义/使用节点对，定义/使用路径，定义/清除路径
- 对变量测试的步骤
  - 确定重点测试的变量V
  - 确定变量V的所有定义/使用节点对
  - 找到高风险路径
  - 针对找到的高风险路径进行测试

内容回顾

- 静态白盒测试
  - 什么是静态白盒测试
    - 对系统静态检查，这种检查通常不需要运行被测软件，而是直接对软件形式和结构进行分析
  - 为什么进行静态白盒测试
  - 怎样进行静态白盒测试
    - 代码检查、静态结构分析、代码质量度量

内容回顾

- 代码检查
  - 审查、团队评审、走查、结对编程、同行桌查、轮查、特别检查
  - 评审流程
- 静态结构分析
  - 函数间调用关系
  - 函数调用层次（确定重点和优先测试项）
  - 分析：是否存在多出口，孤立语句，环复杂度，非结构化设计
- 代码质量度量
  - 不同的因素*权重，求和的思路

本节教学目标

- 理解单元测试的基本概念
- 理解单元测试的过程

目录

1 单元测试的基本概念

2 单元测试的内容

3 单元测试的总结

单元测试的基本概念

- 什么是单元测试(Unit Testing)
  - 是指对软件中的最小可测试单元或基本组成单元进行检查和验证
- 单元选取的原则
  - 对于面向过程的开发语言来说，单元常指一个函数或子过程
  - 对于面向对象的开发语言来说，单元一般指一个类
  - 图形化软件中，单元常指一个窗口或一个菜单

目录

1 单元测试的基本概念

2 单元测试的内容

3 单元测试的总结

单元测试的内容

- 静态测试
  - 主要是通过走查、审查等会议方式，依据模块的详细设计，将代码与缺陷检查表进行对照，查看代码是否符合标准和规范
- 动态测试
  - 主要包括对模块接口、模块边界条件、模块独立路径和错误处理进行测试

单元测试内容

- 模块接口测试：考虑数据能否正确地输入和输出
  - 输入的实参与形参在个数、属性和顺序上是否匹配；
  - 被测模块调用其他模块时，传递的实参在个数、属性和顺序上与被调用模块的形参是否匹配；
  - 是否存在与当前入口点无关的参数引用；
  - 是否修改了只作输入用的只读形参；
  - 全局变量在各模块中的定义是否一致；
  - 是否将某些约束条件作为形参来传递

单元测试内容

- 模块边界条件测试：在被测模块的输入/输出域边界或其附近设计测试用例

单元测试内容

- 对模块中每条独立执行路径进行测试，以发现如下问题
  - 是否正确理解了操作符的优先次序；
  - 是否存在被零除的风险；
  - 是否不满足运算精度要求；
  - 变量初值是否正确；
  - 是否存在错误的逻辑运算符或优先次序；
  - 关系表达式中是否存在错误的变量和比较符；
  - 是否存在不可能的循环终止条件，导致死循环；
  - 是否存在迭代发散，导致不能退出；
  - 是否错误修改了循环变量，导致循环次数多1次或少1次

单元测试的内容

- 模块的所有错误处理路径测试
  - 输出的错误提示是否难以理解；
  - 错误提示是否信息不足，导致无法定位发现的缺陷；
  - 显示的错误是否与实际遇到的缺陷不符合；
  - 是否存在不当的异常处理；
  - 是否存在无法按预先自定义的出错处理方式来处理的情况

单元测试内容举例

```
// FuncRevenueAccount是一个账单优惠计算的函数
#include "stdio.h"
#include "math.h"
double FuncRevenueAccount( double amount )
{
        double rate = 1.0; // 设置折扣率
        if( amount <= 800 ) // 若账单不高于800元，则无折扣
                rate = 1.0;
        else if( amount > 800 && amount <= 1800 )
                rate = 0.9; // 9折
         else if( amount > 1800 && amount <= 4800 )
                rate = 0.8; // 8折
```

单元测试举例

```
        else if( amount > 4800 )
            rate = 0.7; // 7折
        else if( amount <= 0 ){ // 否则，赋予一个负数，表示无效
            return -1.0;
        }
        return amount * rate; // 返回经优惠计算之后的账单
}
```

单元测试举例

- 第一步：做静态和动态检查
- 第二步：编写测试用例做相应测试(借鉴黑盒测试用例设计方法如：等价类、边界值)
- 第三步：使用判定覆盖或独立路径覆盖进行测试（有时会与黑盒测试用例重合，则选其一即可）

单元测试举例

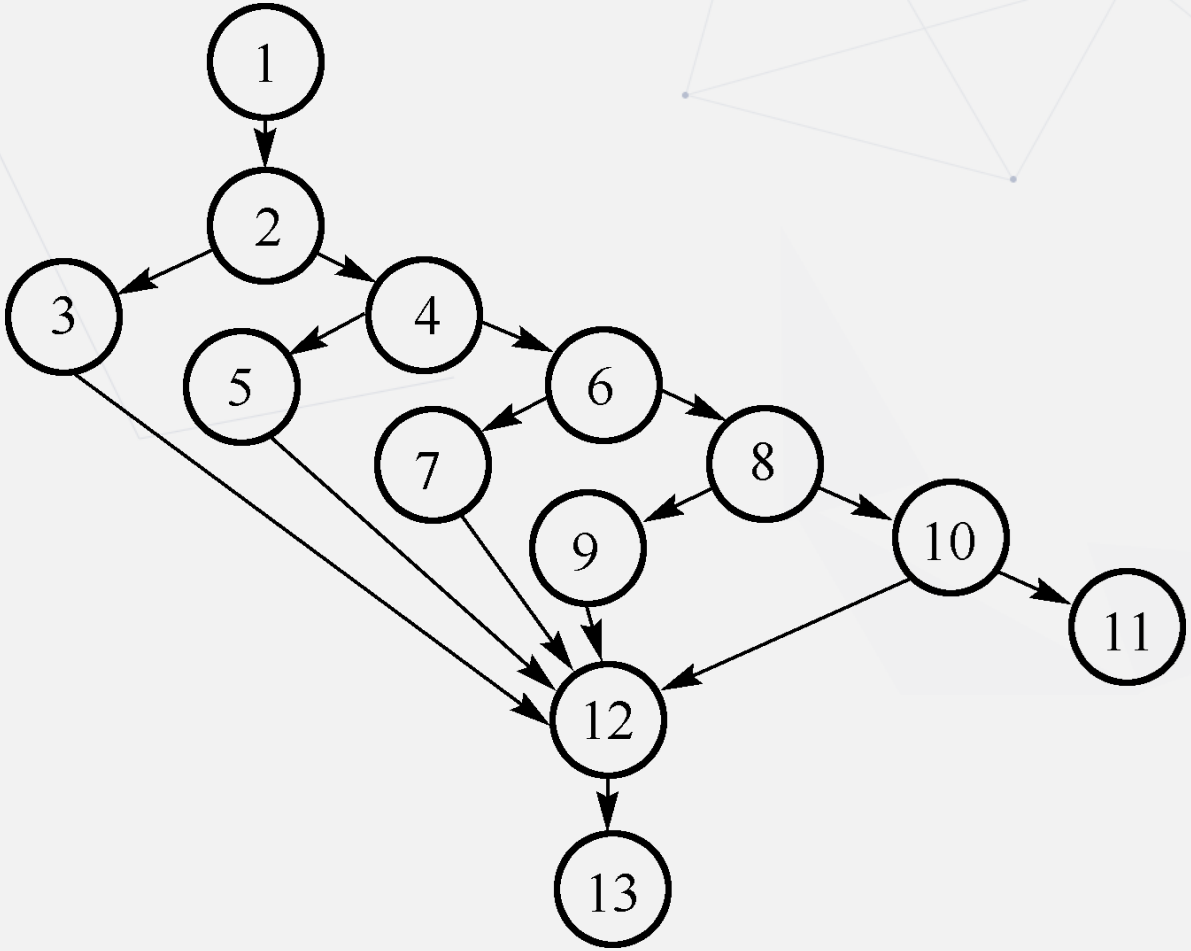
测试用例 ID	输入 (amount)	预期输出 (amount)	备注
RA-UT-001	400.0	400.0	amount 在等价类 (0,800]中，无折扣
RA-UT-002	1300.0	1170.0	amount 在等价类 (800,1800]中，折扣 0.90
RA-UT-003	3300.0	2640.0	amount 在等价类 (1800,4800]中，折扣 0.80
RA-UT-004	8000.0	5600.0	amount 在等价类[4800,+)中，折扣 0.70
RA-UT-005	0.0	-1.0	边界点 0 附近的测试数据，实际输出为 0.0
RA-UT-006	1.0	1.0	边界点 0 附近的测试数据
RA-UT-007	799.0	799.0	边界点 800 附近的测试数据，无折扣
RA-UT-008	800.0	800.0	边界点 800 附近的测试数据，无折扣
RA-UT-009	801.0	720.9	边界点 800 附近的测试数据，折扣 0.90
RA-UT-010	1799.0	1619.1	边界点 1800 附近的测试数据，折扣 0.90

单元测试举例

测试用例 ID	输入 (amount)	预期输出 (amount)	备注
RA-UT-011	1800.0	1620.0	边界点 1800 附近的测试数据，折扣 0.90
RA-UT-012	1801.0	1440.8	边界点 1800 附近的测试数据，折扣 0.80
RA-UT-013	4799.0	3839.20	边界点 4800 附近的测试数据，折扣 0.80
RA-UT-014	4800.0	3840.0	边界点 4800 附近的测试数据，折扣 0.80
RA-UT-015	4801.0	3360.7	边界点 4800 附近的测试数据，折扣 0.70
RA-UT-016	-1.0	-1.0	无效数据的处理，实际输出为-10.0
RA-UT-017	-10.0	-1.0	无效数据的处理

单元测试举例

- 使用独立路径/逻辑覆盖进行测试



单元测试工具

- 单元测试可以借助工具完成，如编译环境，自动审查代码工具
- 单元测试框架Junit，subunit等等

目录

1 单元测试的基本概念

2 单元测试的内容

3 单元测试的总结

单元测试总结

- 分析被测单元
- 分析被测单元中包含的逻辑关系
- 使用静态检查和动态检查的方法（可以借助工具）
- 被测单元功能检查等等

内容总结

- 单元测试概念
- 单元测试内容
- 单元测试步骤
- 单元测试工具

Question