软件工程与计算II总结

传统软件工程方法

- 需求分析
  - 需求工程基础
    - 需求的层次性、需求、问题域和规格说明
    - 需求谱系与分类
  - 需求分析的目标与任务
  - 结构化分析方法
    - 数据流图（建模，分层，精化）
    - 实体关系图
  - 信息结构表示方法
    - 分层框图
    - Warnier图

软件→用户的事

(功能性/...)

{ model
  用户 ok

Jackon { model
         princleple

软件工程与计算II总结
传统软件工程方法

- 软件设计
  - 软件设计基础
    - 软件设计分层
    - 设计过程的主要活动
    - 设计方法与模型
    - 软件设计描述
      - 设计视图和设计图
      - 设计视角和设计关注

软件工程与计算II总结
传统软件工程方法

- 概要设计
  - 软件概要设计的目标与基本任务
    - 软件设计的基本任务
  - 软件设计方法
  - 软件设计的基本原理
    - 模块化、抽象、信息隐蔽、模块独立性、内聚性、耦合性
    - 内聚性、耦合性的划分
    - 模块独立性原则对指导设计的意义
  - 软件结构优化准则
    - 软件结构、模块的影响范围、模块的控制范围
    - 软件结构设计的优化准则

软件工程与计算II总结

传统软件工程方法

- 详细设计
  - 详细设计的目标基本任务
  - 图形设计工具，会画
    - 流程图
    - 方块图
    - PAD图
- 代码设计
  - 设计可靠代码
    - 契约式设计、防御式编程
  - 模型辅助设计复杂代码
  - 单元测试用例开发

软件工程与计算II总结
传统软件工程方法

- 软件测试
  - 测试完备性的含义
  - 动态测试方法
    - 白盒法
    - 黑盒法
  - 测试用例的设计
    - 逻辑覆盖、等价类划分、边界值分析、错误推测
    - 逻辑覆盖中各种覆盖之间的区别
    - 用白盒法、黑盒法设计测试用例
- 软件维护
  - 软件维护的类型
  - 软件维护技术

软件工程与计算II总结
面向对象软件工程方法

- 面向对象的需求分析
  - 面向对象的分析方法
    - 从问题领域到需求，结构视角出发自底向上的分析
  - UML需求建模
    - 用例与用例描述
    - 用例图
    - 概念类图（领域模型）
    - 顺序图
    - 状态图

软件工程与计算II总结
面向对象软件工程方法

- 面向对象的设计
  - 软件体系结构
    - 体系结构抽象与实现、部件、连接件和配置
    - 体系结构风格
    - 体系结构设计过程与原型构建
  - 详细设计方法
    - 面向对象的设计方法
      - 职责与静态模型、协作与动态模型
    - 面向对象方法下的模块化
      - 模块、耦合和内聚
    - 面向对象方法下的信息隐藏
      - 职责封装、为变更而设计
    - 设计模式

软件工程与计算II总结
软件过程

- 软件开发过程模型
  - 软件生命周期模型
  - 软件过程模型
    - 构建-修复模型
    - 瀑布模型
    - 增量迭代模型
    - 演化模型
    - 原型模型
    - 螺旋模型
    - Rational 统一过程
    - 敏捷过程

解答示例

试论述为什么应用演化式的开发方法可以提高测试的效率并降低开发的成本？

改善测试效果：

功能是分阶段开发的，测试工作量可以得到有效的控制，测试工作针对增量，有针对性

节约成本：

分批投入

边开发边回收成本

Modified Condition/Decision Coverage （MC/DC）

It is a form of exhaustive testing, in that during testing all of the below must be true at least once:

- Each decision tries every possible outcome
- Each condition in a decision takes on every possible outcome
- Each entry and exit point is invoked
- Each condition in a decision is shown to independently affect the outcome of the decision

Independence of a condition is shown by proving that only one condition changes at a time. A condition is shown to affect a decision's outcome independently by varying just that decision while holding fixed all other possible conditions.

Modified Condition/Decision Coverage （MC/DC）

```
#include<stdio.h>
 main()
{
     int iX, iY, iZ;
     scanf("%d ,%d, %d", &iX, &iY, &iZ);
     if (iX>5)||(iY==0)
          iX=iX/iZ;
     if (iX==10)&&(iZ>1)
          iX=iX+1;
     printf("%d", iX);
}
```

解答示例

MC/DC覆盖： 需要明确指出每一条测试用例对每个**判定**和**判定中条件组合**的覆盖结果

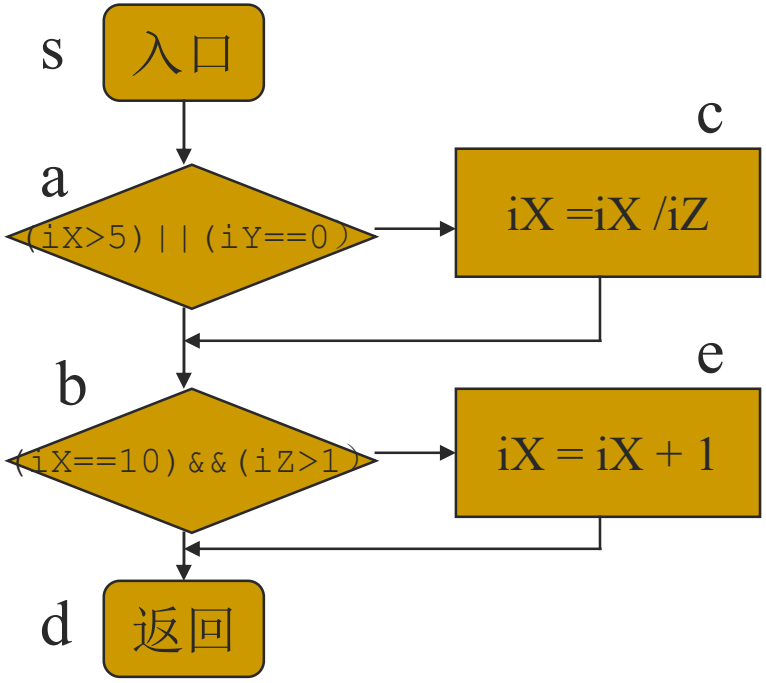
针对判定1需要覆盖三种可能的条件组合:

FF, TF, FT

针对判定2需要覆盖三种可能的条件组合:

TT,TF,FT

用例(iX,iY,iZ)	判定1条件组合	判定1	判定2条件组合	判定2
0,1,2	FF	F	FT	F
20,1,2	TF	T	TT	T
-10,0,-1	FT	T	TF	F



解答示例

路径覆盖： 注意路径是通过语句和判定的序列来表示的；需要明确指出不可行路径

测试数据：

（1）iX=20, iY=0, iZ=2 覆盖 sacbed

（2） 不可行 覆盖 sabed

（3）iX=10, iY=0, iZ=4 覆盖 sacbd

（4）iX=0, iY=1, iZ=0 覆盖 sabd

因判定1为假时，必然有iX<=5, 参考判定2为真的条件iZ>1，判定2中iX==10这一条件无法得到满足，因此路径（2）是**不可行路径**。