**软件工程期末总结**

**题目总结：  
1.** （第12章：代码工程质量）：

结构化程序设计是什么，？

给一段代码，是结构化的吗，不是的话把它改成结构化的。

答：

结构化程序设计：所有的程序都可以由以下3张语句块构成：（1）赋值语句块（2）if-then-else语句块（3）while循环块。这3中语句块是不再分隔的语句块。此时便是结构化程序。如果出现如goto语句，则会发生从一个模块转移到另一个模块或另一个模块内部，破坏了程序的结构化，此时就不再是结构化程序。

【注：可以有限制的使用goto语句而不破坏程序的结构化】

**2. （**第13章：软件测试理论与技术）：

1根据代码，画program flow

2填表，熟练理解 语句覆盖，分支覆盖，MC/DC概念

**Table 6-1:**

x = 3.0;

y = 3.0;

if (AA && BB)

{

z = x + y;

if (CC)

{ z = y + x/y;}

else

{ y = x + y}

}

z = z + x/y;

Table 6-2: Test-Cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| The need of coverage test  (测试要求) | Test-cases (测试用例) | | | | | |
| Inputs | | | Outputs expected | | |
| AA | BB | CC | x | y | z |
| Statement coverage = 100%  (语句覆盖达 100%) | 1  1 | 1  1 | 1  0 |  |  |  |
| Branch coverage or  Decision–Decision Path(DDP)  = 100%  (分支覆盖达 100%) | 1  1  0 | 1  1  0 | 1  0  1 |  |  |  |
| MC/DC = 100% | 1  1  0  1 | 1  1  1  0 | 1  0  1  0 |  |  |  |

**第二章 基于计算机的系统**

计算机系统故障种类：  
(1) 硬件故障

(2) 软件故障

系统软件故障通常可以分为以下几类：

1. 计算机的除零错误
2. 指向内存地址的指针超出规定的内存范围
3. 整形数上溢或下溢
4. 浮点数上溢出或下溢出
5. 其它类型的错误包括数据精度不够，输出或显示的数据位数不够等等。

以下为WEB服务器的典型故障：

1. 资源耗尽。例如，内存泄露，线程池耗尽导致的速率降低，资源密集处理导致服务请求超时，快速增长的日志文件消耗掉磁盘
2. 计算逻辑错误。例如，引用数据库中的表，使用未释放的你村，指针奔溃，程序死锁等造成的同步错误。
3. 系通过在，系统过载导致的资源竞争，以及与其它软件进程之间的频繁交互。
4. 可恢复的代码错误。系统的容错设计过于复杂，并且没得到充分的测试。
5. 系统的升级错误。例如，升级前的备份工作出现问题，升级时未检查升级前后的软件兼容问题。软件升级后自动进入到缺省状态。继承错误或第三方软件有错。

(3) 人为操作故障

因错误操作导致的刺痛故障主要分为配置错误，规程错误和综合错误，以下是详细介绍。

1. 配置错误。其发生原因可能是由于错误的配置导致系统复位缺省的配置或是配置文件崩溃，启动脚本忽略了必要的服务等等。
2. 规程错误。其发生原因可能是由于数据库备份出现故障，恢复了错误的备份，忘记启动WEB服务器，错误删除文件，不正确输入，以及忘记删除日志文件导致磁盘被占满。
3. 综合错误。其发生原因可能是有与设备滑落造成创伤，或是电源意外断开。

**第三章 软件开发过程**

软件开发过程使用模型：  
**1. 瀑布模型**

软件对安全和性能有极其严格的要求，容不得半点疏漏，比如航空航天软件。这样用瀑布模型的话能够有效地控制每一环节，所有流程都有文档可循。

**2. 快速原型模式**

快速体现用户需求，确保交付的产品满足用户的要求，非常适合界面和人机交互的系统的快速开发。

当时是规模不是很大的，商业型的应用软件，更倾向于快速得到可用的软件。而且软件需要能够适应市场的变化作出相应的调整。原型模型无疑是更适合的。

**3. 增量模型**

优点：

（1）采用增量模型的优点是人员分配灵活，刚开始不用投入大量人力资源；

（2）如果核心产品很受欢迎，则可增加人力实现下一个增量；

（3）可先发布部分功能给客户，对客户起到镇静剂的作用

应用：

（1）进行已有产品升级或新版本开发，增量模型是非常适合的；

（2）对完成期限严格要求的产品，可以使用增量模型；

**4. 渐进式模型**

强调目标与方案之间相互调试不是一步到位，注重反馈调节，不断微调，而且注重对现行政策的修改和补充

应用：

中小规模的交互式系统

大系统的一部分（例如，用户接口）

生命周期较短的系统

**5. 螺旋模型**

螺旋式模型采用周期性方法开发，每个项目阶段使用瀑布式模型法，每个周期都包括需求定义，风险分析，工程实现，评审共4个阶段，进行迭代。因此它强调了进行风险分析，适用于做庞大，复杂，高风险系统。

**6. 同步和稳定模型**

满足未来的用户需求，确保部件能够成功集成，适合小规模和大规模的产品开发。但所体现的是“足够好”而不是等到产品“完美”才发布，除微软之外没有得到普遍使用

**7. 建造和调试模型**

适合短程序，不需要维护的程序，几乎没有软件工程的概念，不适合任何稍大的程序。

**8. 形式化方法**

形式化方法是指通过数学的方法来解决软件工程领域的问题，主要包含建立精确的数学模型以及对模型的分析活动，。此种方法的出发点是数学逻辑方法，目的是开发可靠的软件产品。

**第四章 软件质量**

I**SO 9126模型**

从**产品质量**观点出发，产品质量模型将产品质量分解为许多质量因素。ISO 9126模型将软件质量属性划分为6个特性（**功能性，可靠性，易用性，效率，维护性和可移植性**）

**相关名词解释：**

**功能性：**功能性是指软件在指定条件下使用时，软件产品提供满足明确和隐含要求的功能的能力，包含密安性，互操作性，适合性，准确性等。

**可靠性：**可靠性是指在特定条件下使用时，软件产品维持规定的性能级别的能力。包括成熟性，容错性，易恢复性。

**易用性：**易用性是从用户观点来看，指在特定条件下使用时，软件产品被理解，学习，使用和吸引用户的能力。包括易理解性，易操作性，易学习性，吸引性。

**效率：**在规定条件下软件产品所占有的资源数量以及所提供的功能和性能的能力。包括时间特性，资源利用性.

**可维护性：**是指软件产品可被修改和维护的能力，包括易分析性，易修改性，稳定性，已测试性。

**可移植性：**软件产品从一个环境迁移到另一种环境的能力。

**第五章 系统的可信赖性**

相关名词解释：  
**可信赖性：**人们对基于计算机系统所提供的服务的信任程度，包括以下属性：**可靠性：**系统服务的连续程度

**安全性：**不会给环境和人员带来灾难性的后果，获得安全性的方法是设计容错机制，容错是从发生事故和错误组合的系统设计概率角度出发的，而没有考虑系统被攻击的情况。容错是指一个系统有多个独立的部件同时完成一个任务。

**可维护性：**修复和进入正常工作状态的能力

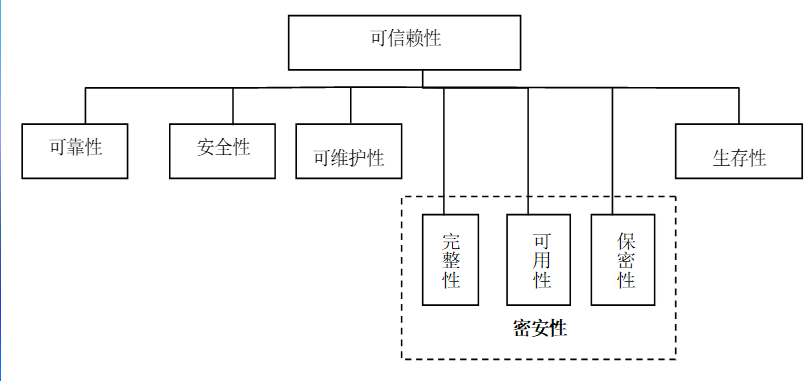
**密安性：**由可用性和具有授权的完整性和保密性所构成。包括：防止非授权用户存取或访问系统中的信息，以及防止意外事故等引起的不希望的信息泄露。

**完整性：**不会发生不适当的修改信息的情况

可用性:

密保性:

**生存性,鲁棒性：**在系统被攻击时，系统面对攻击的坚固和抵抗程度。



**第六章：**

**软件销售与采购**

采购过程：

1. 策划组织采购策略，评审采购者的目的
2. 建立一个适合组织需要的采购过程
3. 定义软件需求
4. 识别潜在供货商
5. 准备合同要求
6. 方案评审，选择供货商
7. 监督和管理供货商
8. 软件验收
9. 运行

**第八章 需求工程**

**相关概念：**

**软件开发周期的六个步骤:**

1. 制定计划（确定要开发软件系统的总目标，给出功能、 性能、可靠性以及接口等方面的要求，完成该软件任务的可行性研究，制定出完成开发任务的实施计划）

2. 需求分析（对待开发软件提出的需求进行分析并给出详细的定义）

3. 设计（把各项需求转换成软件的体系结构，对每个模块要完成的工作进行具体的描述，为源程序编写打下基础）

4. 程序编码（把软件设计转换成计算机可以接受的程序代码）

5. 测试（通过单元模块测试和组装测试，对各项需求进行有效性测试）

6. 运行维护（改正性维护 适应性维护 完善性维护  ）

**需求分析**：是软件开发周期最重要的阶段，目的是从非正式的思想中提炼出“规格说明或规范”，提出系统的功能和非功能需求，以及如何测量是否满足这些要求的准则或度量标准，将这些编写成为文档即需求规范，如果此规范描述的是系统级的，包括了软件和硬件则称为系统需求规范；如果只包括软件则称为软件需求规范。

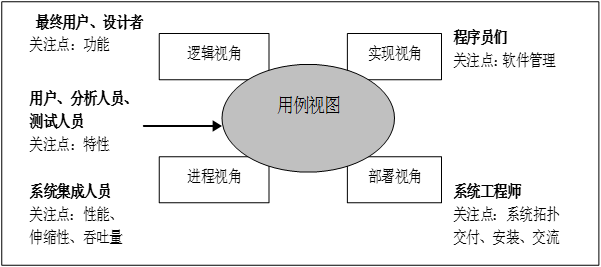
需求类型：用户需求，系统需求（建设者角度）

需求内涵：功能需求，非功能需求，领域和行业需求（可以归到非功能需求 中）

**第十章 体系结构设计**

Philippe Kruchten 建立了“4+1”视图模型。该模型从四个角度(逻辑、实现、进程和部署)指出不同的相关利益方关心的事情。

及时进步哩！



体系结构设计方法

“分割”法 7层设计

“抽象”法 java虚拟机

“压缩”法 提高性能

“组合”法 多个模块组合到一起

“复制”法 copy

“资源共享”法 数据库,黑板.

设计方法对质量的影响

**第十一章 体系结构模式与应用**

**相关概念：**

设计体系结构的目的是更容易的建造软件系统，体系结构分级可进一步划分为：

概念级体系结构

模块级体系结构

运行级体系结构

代码级体系结构

模块体系结构：

系统的模块化设计是系统进行复用的关键，模块级体系结构反映了对软件代码实现时的期望

体系结构描述语言（ADL）

体系结构分析和描述语言（AADL）

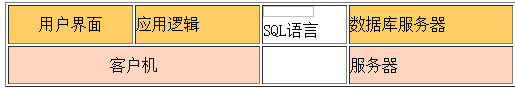
C/S模式与B/S模式

**C/S模式**

**两层C/S模式：**

服务器：只负责各种数据的处理和维护，为各个客户机应用程序管理数据

客户机：包含文档处理软件、决策支持工具、数据查询等应用逻辑程序，通过网络使用SQL语言发送、请求和分析从服务器接收的数据。

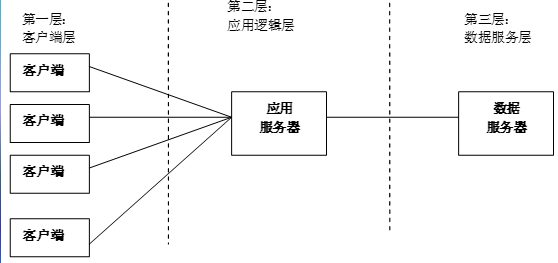
缺点：  
系统的可靠性有所降低、缺乏灵活性、资源浪费严重以及维护费用较高  


**三层C/S模式：**

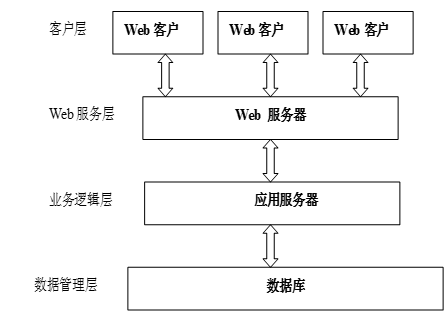
客户端层：负责数据的表现，接收用户请求，控制用户界面

应用逻辑层（应用服务层）：负责实现应用逻辑，既给客户端层提供服务，也会依据客户端请求向第三层服务器提出服务请求

数据服务层：负责向第二层提供服务，如：数据库中的数据存储。



**多层C/S模式：**



**C/S模式优点：**

1. C/S模式提供了更安全的存取模式
2. 可使用性：每层可部署多个服务器，互为备份，一个出现故障还可以使用其它的，虽然性能有所降低，但是仍可用。
3. 可修改性：分隔增加了可修改性，每层都有其内部逻辑，修改时不会影响其它层。
4. 可伸缩性：各层中的服务器都有备份，多个服务运行在同一个或多个不同的服务器上，体系结构模式可以得到很好地提升。

**B/S模式**

即浏览器/服务器模式，本质是三层C/S模式。

客户机：用户的计算机上安装浏览器，在客户端运行浏览器软件。

应用服务层：负责实现应用逻辑，既给客户端层提供服务，也会依据客户端请求向第三层服务器提出服务请求

数据服务层：负责向第二层提供服务，如：数据库中的数据存储

**C/S模式与B/S模式对比：**

C／S模式（这里的C/S模式应该是指两层模式）

优点：

1. 具有很强交互性
2. 具有安全的存取模式
3. 响应速度快、利于处理大量数据。

缺点：

1. 变更不够灵活
2. 维护和管理的难度较大
3. 分布功能弱且兼容性差，因此缺少通用性。

B／S结构（这里的B/S模式应该可以指三层C/S模式）

优点：

1. 分布性强
2. 可修改性较强，维护方便
3. 开发简单且共享性强、总体拥有成本低。

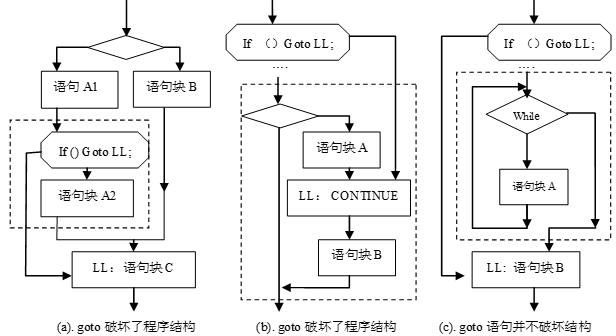
缺点：

1. 数据安全性问题
2. 对服务器要求过高
3. 数据传输速度慢

**第十二章 代码工程**

1. 结构化程序设计：所有的程序都可以由以下3种语句块构成：（1）赋值语句块（2）if-then-else语句块（3）while循环块。这3中语句块是不再分隔的语句块。此时便是结构化程序。如果出现如goto语句，则会发生从一个模块转移到另一个模块或另一个模块内部，破坏了程序的结构化，此时就不再是结构化程序。

【注：可以有限制的使用goto语句而不破坏程序的结构化，如下：】



1. McCabe的复杂度度量

公式：V(G)=e-n+2

其中，e为边的个数，n为结点个数，V(G)为圈复杂度。

如下图1：

该程序流程图是一个连通图，但不是强连通的，因为缺乏一个从出口点到入点的通路。鉴于此，McCabe扩展了上述定理----增加一个从终结点(f)向进入点(a)画一个虚有向边。

V(G)=e-n+2=9-6+2=5

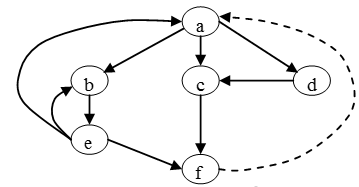


图1 程序控制流图

一个强连通图G的循环圈数V(G)等于该图中线性独立路径数目，所以可以这样理解：一个独立的不重复的循环就是一个独立路径，其它路径可由其组成，每个独立路径之间没有包含关系。如上图中的独立路径共有5个，分别为：(abefa),(beb),(abea),(acfa),(adcfa)。

例如：(abeabebebef)可以描述为：(abea)+2(beb)+(abefa)

语句块的结构化控制流图：

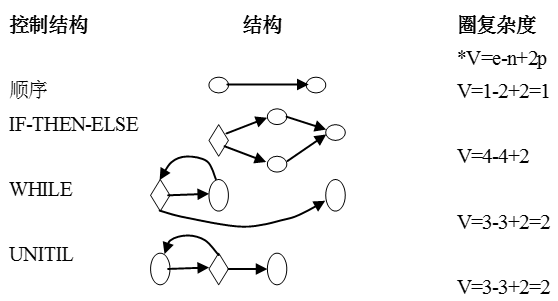


图2 典型控制结构的V(G)

McCabe定义的V(G)具有如下特性：

1. V(G)>=1
2. 在代码中插入和删除一个功能语句并不影响V(G)的值，因为边数和结点同时递增/减.
3. V(G)只取决于图中的判断结构，从图2可看到判断语句和循环语句的V(G)是一样的，只有判断语句才会增加边数和结点的差
4. 假定一个主程序M调用两个子程序A和B，将3个程序合并后会有：

V(MUAUA)=V(M)+V(A)+V(B)

现在V(G)的计算公式简化为：V= p+1，p是二叉判断谓词。P在控制流程图中表示只有两个边的判断结点。

如下 图3：

其中a,b,c是二叉结点，d是四叉结点

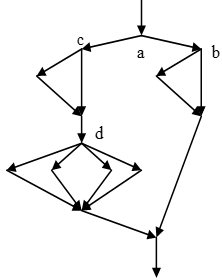


图3 V(G)简化计算的例子

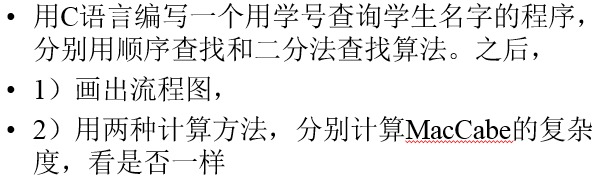
对于二叉结点p为1；对于四叉结点p为3，所以根据公式V(G)=P+1得

V(G)=P+1=3+3X1+1=7

同时，根据原始公式V(G)=e-n+1得

V(G)=e-n+1=20-14+1=7

**第12章作业：**



**优秀答案：  
见第十四章，MacCabe运用公式：V(G)=e-n+2p，其中e为边数（注：边数不包括那条补上的虚线），n为结点数，p为1**

**第十三章 软件测试理论与技术**

测试方法：  
**基于控制流（白箱测试）：**

主要测试程序控制路径的结构

1. 完全路径覆盖准则(Path Coverage Criterion)：执行路径的集合P满足路径覆盖准则，当且仅当，P包含流程图中从开始结点到最终结点的所有的执行路径.
2. 语句覆盖准则：可执行路径的集合P满足语句覆盖准则，当且仅当，对于流程图中所有结点n，P中至少有一个路径p，使得n在路径p上。

语句覆盖率 = （起码被执行过一次语句数）/ （总的语句数）\* 100%。

1. 分支覆盖准则(Branch Coverage Criterion)：执行路径的集合P满足分支覆盖准则，当且仅当对于流程图中所有的边，集合P中起码有一个路径p 包含了边e 。

分支覆盖率 = （起码被执行过一次分支数）/ （总的分支数）\* 100%

1. 圈复杂数覆盖准则(Cyclomatic-Number Criterion)：执行路径的集合P满足圈复杂度准则，当且仅当，P起码包含V个独立路径集合中的一个路径。这里：V= e-n +2，之后将V的计算简化为：V= p+1，p是二叉判断谓词。P在控制流程图中表示只有两个边的判断结点。

见12章

1. 多条件覆盖准则：条件组合中每个子条件的所有可能性都要试。
2. MC/DC（条件/判断）覆盖准则：多条件判断准则的简化版,条件组合包含所有可能结果，测试集能够表明每个条件独立的影响最终判断结果

**基于数据流：**

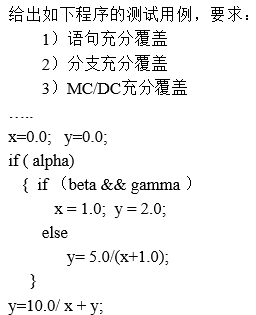
更关注对数据值的边界，变量的使用分析。

每个变量的使用分别出现在：定义时，计算时和判断时。将其分别表示为 def， c-use， p-use。

**基于需求规范（黑箱测试）：**

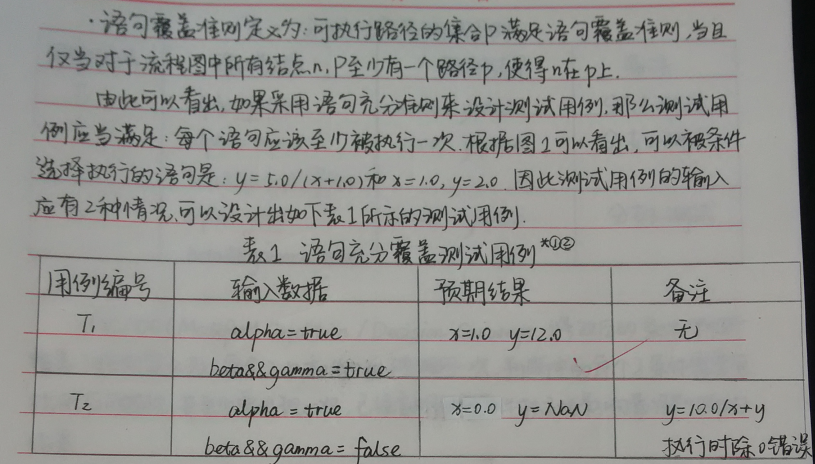
不关心代码的内部结构，只关注软件的输入输出以及环境

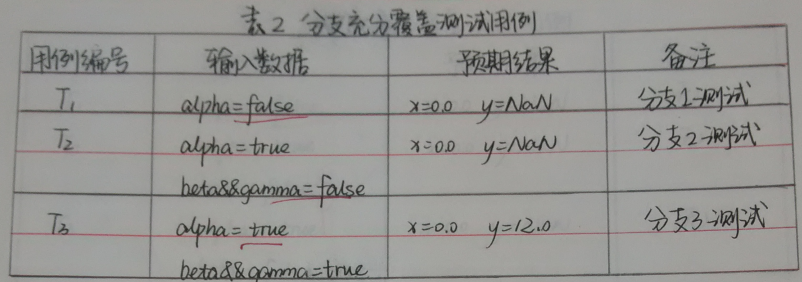
**第13章作业：**

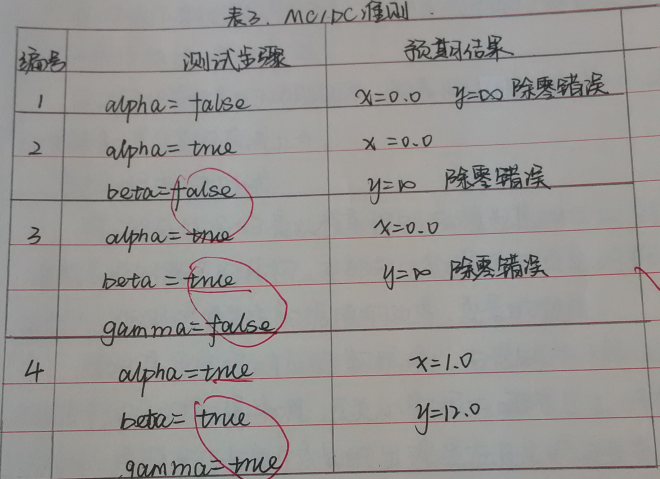


**优秀答案：**

语句充分覆盖准则：

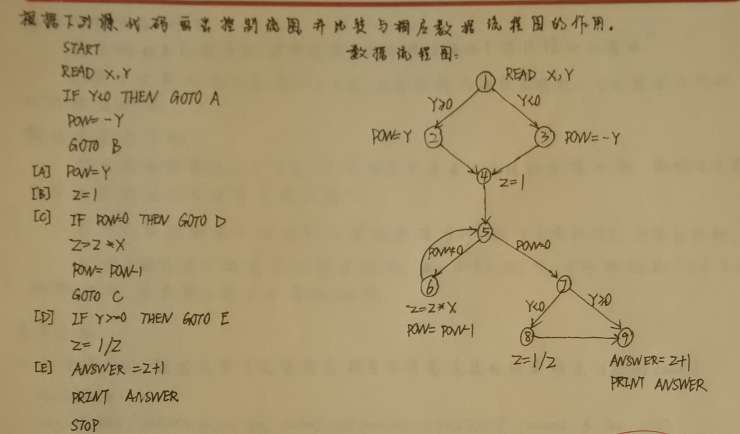


条件充分覆盖准则：  


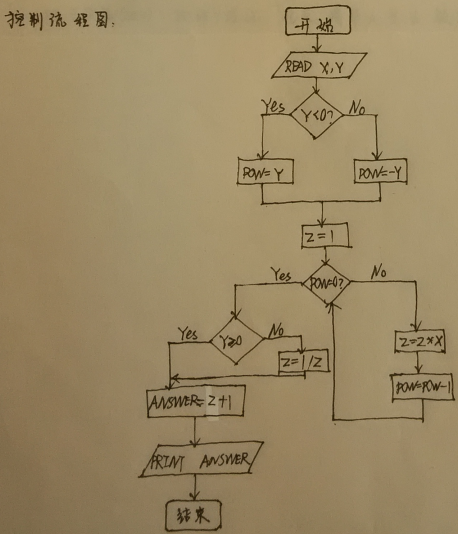
MC/DC覆盖准则：  


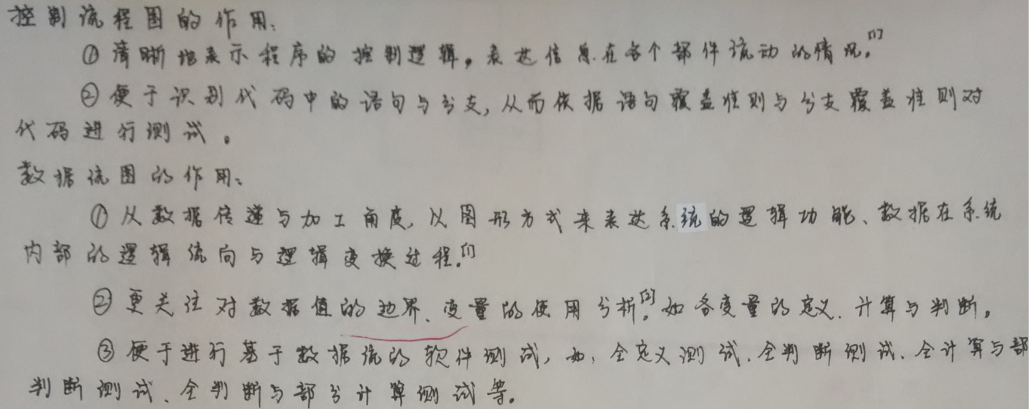
**附加作业：**

根据下列源代码画出控制流图，并比较与相应数据流图的作用。



**答案：**





**第十四章 软件测试工程**

软件测试目的：尽可能发现软件中的错误，提高软件质量和可信赖性，这里的质量泛指对软件的性能，可靠安全性，密安性等指标要求。

1. 测试方法

静态测试：

对文档和代码的人工质量评审

动态测试：

单元测试，集成测试，系统测试，验收测试

1. 测试过程

测试环境的准备->测试用例设计->测试的执行和记录->测试结果分析与评判

1. 测试用例的设计

测试用例：对被测试系统的输入数据，预测的中间和最终输出结果，测试的执行步骤，执行测试的前置条件，评价该测试用例是否测试成功的判断条件。

1. 随机方法

测试人员（或测试工具）随机选择和组合测试数据形成测试用例

1. 判定表方法

判定表用来描述应用程序的行为。

如下图1 为判定表，图2 为从判定表得到的测试用例

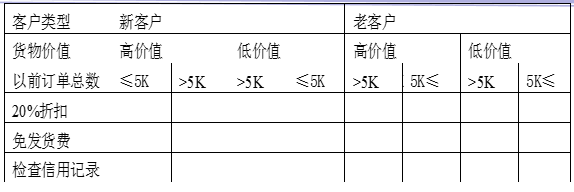


图1 网上购物的判定表

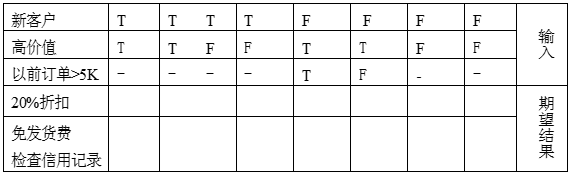


图2 网上购物的测试用例

特点：简单具有系统性，但是缺乏对约束条件的归纳，输入和期望结果之间的关联表现不够。

1. 等价类划分

降低测试工作量的主要方法是将被测系统划分区域，原则是找出被测软件系统的同类项或等价类。

等价类：一个等价类(equivalence partition)是输入集合的子集。用该子集中的每个输入元素对被测系统S进行测试时，所引起S发生同样的运行错误，或都不能引起S发生错误。

划分等价类的方法：

1. 依据整型或浮点变量值划分

例：在一个应用程序中，用整数age表示人的年龄。假设其范围是[1..120]，那么age的输入自然被划分为三类：

* + - (a)自然数的集合[0,∞]，
    - (b)计算机表达的整数，例如，[0..32767]，
    - (c) 实际的范围[1..120]。

例：在测试中，再考虑到：(d)工作年龄段为[1..60]和退休年令段[61..120]。由此，一个测试年龄的等价类为：

<1，[1..60]，[61..120]，[120, 32767]，>32767。

1. 依据字符串变量划分

例：定义：firstname: string，则其等价类为：

{空字符串}，{张三}，{最长长度的字符串}，{超过最长长度的字符串}

1. 数组变量

例：int [] aName： new int [3]; 则其等价类为：

{[ ]}, {[-10,20]}，{[-9,0,12,15]}

1. 枚举量划

例：例如定义：Autocolor: {red, blue, green}; 则其等价类为：

{{red}，{blue}, {green}}

1. 布尔量划分

例：Boolean; 则其等价类为: {{true}，{false}}

1. 边界值方法

有时如果简单地从等价类中取一组值是有风险的，因为软件的许多错误往往就出现在边界值上**。**

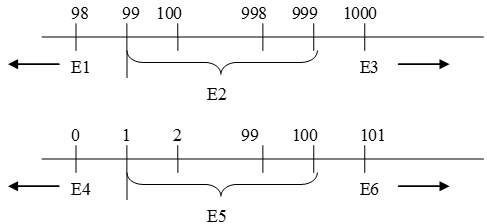
【注：】内部程序数据结构有预定义的边界(例如，100项的数组)，一定要测试其边界的值

例：假设code 的范围是99..999，quantity的范围时1..100

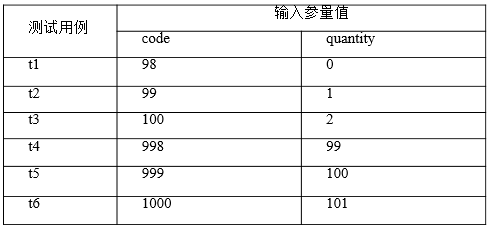
第一步：划分出等价类如下：

* + Code的等价类：E1: <99 ； E2: [99..999] ；E3: >999。
  + quantity的等价类：E4: < 1；E5: [1..100]； E6: >100。

第二步：标出各个变量的边界：



第三步：构造出测试用例

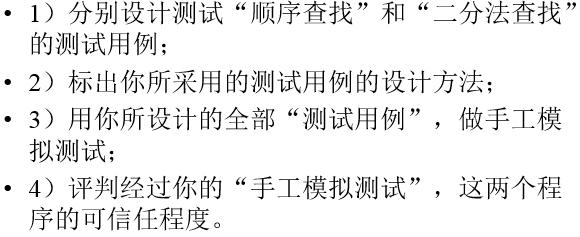


边界值方法

1. 因果图方法

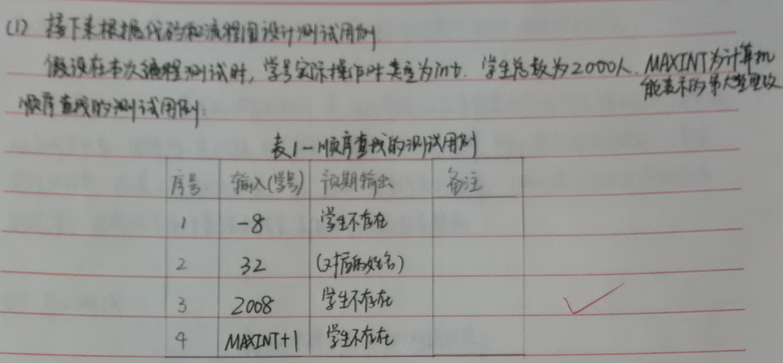
跟踪因果图转换为测试用例

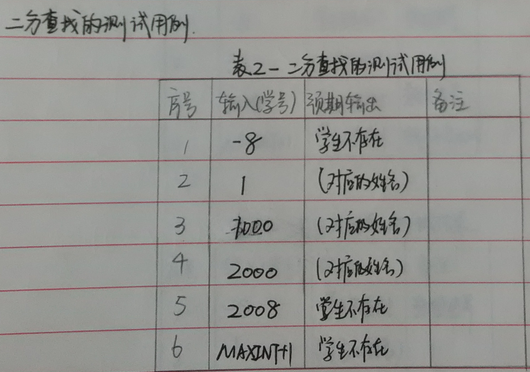
**第十四章作业：**

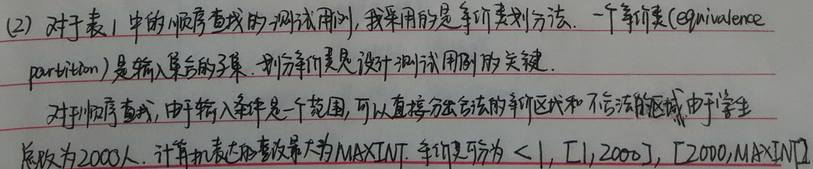


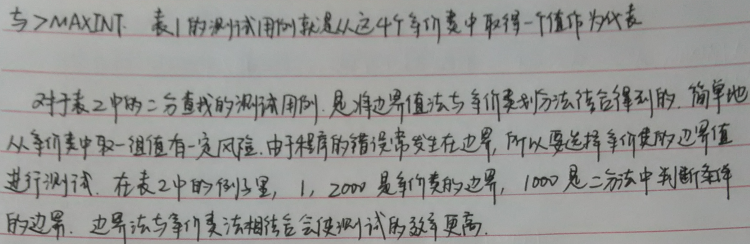
**优秀答案：**

**【注意】两种算法的测试用例个数最好复杂的算法用例数更多，且二分法要注意测试数组长度为奇数和偶数的时候。**









**第十五章 软件缺陷预测与预防**

软件缺陷预测：为了预防软件运行中风险的发生，根据项目的当前状态预测下一阶段应当排出的缺陷个数

1. 基于代码的缺陷预测：

**公式一（秋山）：**

D = 4.86 + 0.018L，其中D为缺陷个数，L为代码行数

**公式二（Gaffiney）：**

D=4.2 +0.0015(L)4/3，其中D为缺陷个数，L为代码行数

**公式三（UNISYS公司）：**

D= 0.069 +0.00156L+0.00000047(L)2

根据本公式，Compton 和Withrow认为对于Ada语言，每个模块的最优规模为83行源代码。称为“金发原则（Goldilocks Principle）”，即，模块不能太大也不能太小。

大模块比小模块更可靠。也就是说，模块的规模要适当，才能减少缺陷。

**公式四（Kitchenham）：**

C=0.25MCI – 0.53DI + 0.09VG，其中MCI是机器代码指令数，DI表示数据项。

1. 软件缺陷预防

软件缺陷预防的基础是分析缺陷发生的因果关系。

如下图1 为缺陷预防的流程图：

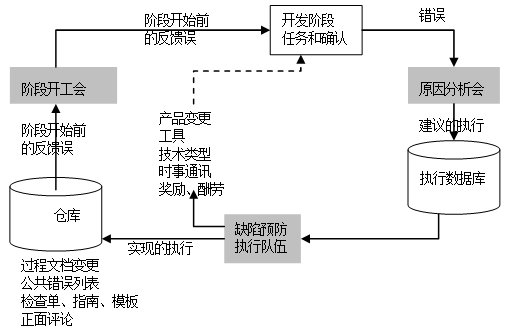
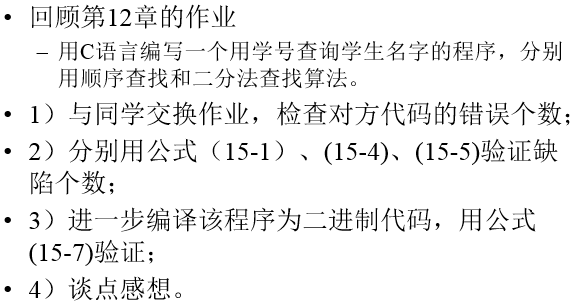


图1 缺陷预防流程图

开发阶段任务和确认->查看项目的错误统计数和分类->原因分析会议（讨论错误原因，提出避免措施，调整开发过程）->执行建议（执行数据库）->安排缺陷与反执行队伍（定期召开会议讨论）->实现执行（仓库）->反馈（是缺陷预防的重要步骤）->

**第十五章作业：**



**第十六章 质量控制与管理**

1. 软件系统质量来源于两方面：

系统运行时表现出的性能不足

影响因素：软件采用的体系结构，软件处理业务的模式软件的使用和管理方法以及软件使用环境

开发过程带来的缺陷或隐藏的错误

质量低下原因：

软件设计不足，开发过程遗留缺陷以及维护和使用不当

1. SQA：

建立独立的软件质量保证(Software Quality Assurance)队伍，简称为SQA。

SQA 队伍具有两个任务：

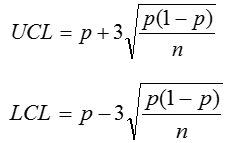
1. 组织或协调评审小组对中间的产品进行审查。目的是对具体产品的评价，

（2） 组织审查开发小组是否按计划的流程或活动在进行。目的是对开发流程的评价。

1. 质量统计方法

缺陷服从二值分布则：

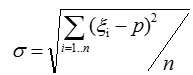
每编写一行代码出错的可能性p = (错误的代码行数) / (被抽检的总代码行数)



缺陷范围为[LCL.UCL]

当n足够大时，“二值分布”趋近于正态分布：

质量采样值ξi (i=1…n)，计算出平均值：  

上线UCL和下线LCL，p + nσ和p – nσ (n表示有几个σ)。

缺陷分布如图 1：

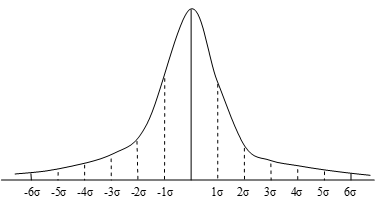


图1 缺陷分布

1. 质量宽泛的控制

质量宽泛：实际质量的均值与目标一致，但方差太大

如下图 2：

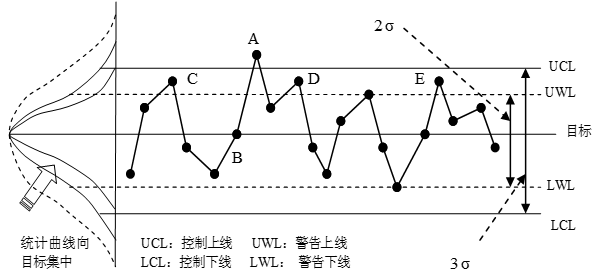


图2 质量控制示意图

其中：  
目标是设定的平均缺陷值

UCL是设定的质量上线值，一般UCL=目标值+3a

LCL是设定的质量下线值，一般LCL=目标值+2a，当LCL计算<0时将其设定为0

在图中：

B：完全符合质量目标要求

A：超出了质量偏差的要求，该点严重降低了整个系统的质量，称为特殊变异点

C,D,E：质量警告点

其余点：一般变异点，与要求有偏差，但未超出范围。

为了控制质量的宽泛性，首先要寻找A点发生的原因，然后消除变异点A，那么统计方差会降低，接着会出现新的特殊变异。随着特殊变异点的小时，偏差和期望目标俊辉发生变化，体现为统计曲线越来越“瘦”，质量指标（千行代码缺陷率）会降低。

建立质量控制上线的同时建立质量警告线可以为过程的改进和优化提高机会，例如，研究图2中的点C,D,E的质量变异情况，可以为后期的技术和过程改进提供参考。

**第十六章作业：**



**优秀答案：**