软考

2019年5月6日　　下午8:41

数据流图

> 数据流图（Data Flow Diagram）：简称 DFD，它从数据传递和加工角度，以图形方式来表达系统的逻辑功能、数据在系统内部的逻辑流向和逻辑变换过程，是结构化系统分析方法的主要表达工具及用于表示软件模型的一种图示方法

数字字典（DD，与 DFD 配套。用于描述数据流图中的数据流、文件的数据构成、 加工处理及外部实体的一种工具）

符 号	含 义	例及说明
=	被定义为	
+	与	X=a + b 表示X由a 和 b 组成
[•••|•••]	或	X=[a | b] 表示X由 a或 b组成
{•••}	重复	X={a} 表示X由 0个或多个 a 组成
m{•••}n或{•••}N	重复	X=2{a}6 或 x={a}$^{6}_{2}$ 表示重复2~5次 a
(•••) m	可选	X=(a) 表示 a 可在X中出现，也可不出现
"•••"	基本数据元素	X="a" 表示X是取值为字符a 的数据元素
••	连接符	X=1••8 表示X可取1到8中的任意一个值

数据流条目

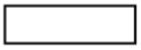
管理要求=【登记客户信息|日统计要求】
查询要求=借贷部统计
客户信息管理=【客户借款表|客户还款表】
日统计表=【日还款统计表|日借款统计表】
借款单=客户号+账号+项目（购房贷款|助学贷款|购物贷款）+金额+
借款日期
+期限+月付金额
还款单=客户号+账号+项目（购房贷款|助学贷款|购物贷款）+金额+
还款日期
+期限+月付金额
项目营业情况表=项目名+月利润

文件条目

文件名：客户借贷信息文件
组成：{客户号+账号+金额+项目+借款日期|还款日期+期限}
组织：按客户名递增顺序排列

文件名：银行查询文件
组成：{项目+月利润+借款日期+期限}
组织：按项目名递增排序

基本图形符号

符号	名称	说明
	加工	在圆中注明加工的名字与编号
	数据流	在箭头边给出数据流的名称与编号，注意不是控制流
	数据存储文件	文件名称为名词或名词性短语
	数据源点或汇点	在方框中注明数据源或汇点的名称

分层数据流图

分层的数据流图主要分为：顶层图和0层图

顶层图是确定与外部实体之间的输入和输出数据流

0层图是将顶层图中的加工分解成若干个加工，并用数据流连接这些加工。是顶层图的细化过程。满足结构化方法原则中的自顶向下，逐层分局的原则

数据平衡原则

1. 分层数据流图中的数据平衡原则
   父类和子类之间的数据流必须保持一致，包括数量和内容上一致，或者上（下）层输出等于上（下）层的输出
2. 每张数据流图的数据平衡原则
   加工的输入数据流和输出数据流要平衡，保证加工的输出数据流都有对应的输入和输出数据流

黑洞：只进不出
奇迹：只出不进
灰洞：加工不出输出流

画法步骤

1. 确定系统的输入输出
   由于系统究竟包括哪些功能可能一时难于弄清楚，可使范围尽量大一些，把可能有的内容全部都包括进去。此时，应该向用户了解“系统从外界接受什么数据”、“系统向外界送出什么数据”等信息，然后，根据用户的答复画出数据流图的外围
2. 由外向里画系统的顶层数据流图
   首先，将系统的输人数据和输出数据用一连串的加工连接起来。在数据流的值发生变化的地方就是一个加工。接着，给各个加工命名。然后，给加工之间的数据命

名。最后，给文件命名

顶层流图只包含一个加工，用以表示被开发的系统，然后考虑该系统有哪些输入数据、输出数据流。顶层图的作用在于表明被开发系统的范围以及它和周围环境的数据交换关系

3. 自顶向下逐层分解，绘出分层数据流图

对于大型的系统，为了控制复杂性，便于理解，需要采用自顶向下逐层分解的方法进行，即用分层的方法将一个数据流图分解成几个数据流图来分别表示。

分层

一般将层号从0开始编号，采用自顶向下，由外向内的原则。画0层数据流图时，分解顶层流图的系统为若干子系统，决定每个子系统间的数据接口和活动关系

编号

如果一张数据流图中的某个加工分解成另一张数据流图时，则上层图为父图，直接下层图为子图。子图及其所有的加工都应编号

父图与子图的平衡

子图的输入输出数据流同父图相应加工的输入输出数据流必须一致，此即父图与子图的平衡。

局部数据存储

当某层数据流图中的数据存储不是父图中相应加工的外部接口，而只是本图中某些加工之间的数据接口，则称这些数据存储为局部数据存储

提高数据流图的易懂性

注意合理分解，要把一个加工分解成几个功能相对独立的子加工，这样可以减少加工之间输入、输出数据流的数目，增加数据流图的可理解性

4. 实现

图3-3 飞机机票预定系统顶层图

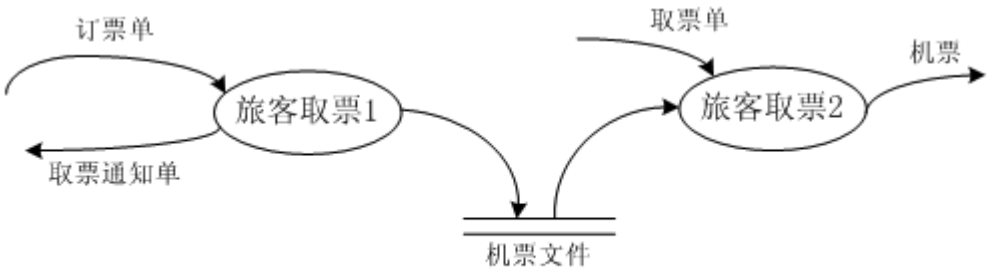


图3-4 飞机机票预定系统0层图

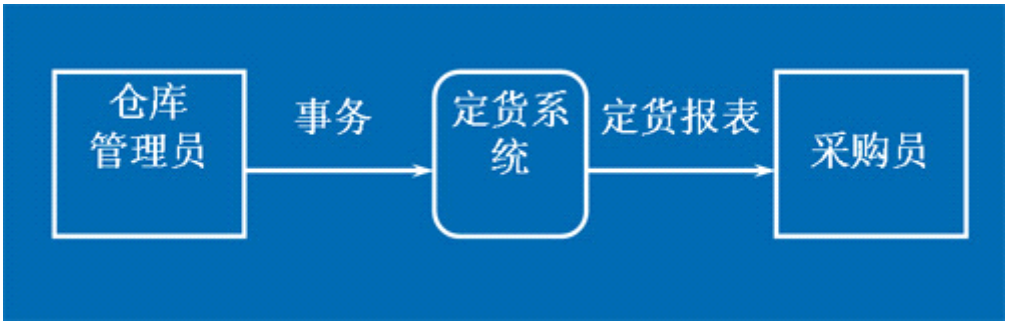
例题

假设一家工厂的采购部门每天需要一张定货报表。报表按零件编号排序，表中列出所有需要再次定货的零件。对于每个需要再次定货的零件应该列出下述数据：零件编号、零件名称、定货数量、目前价格、主要供应商、次要供应商。零件入库或出席称为事务，通过放在仓库中的CRT终端把事务报告给定货系统。当某种零件的库存数量少于库存临界值时就应该再次定货

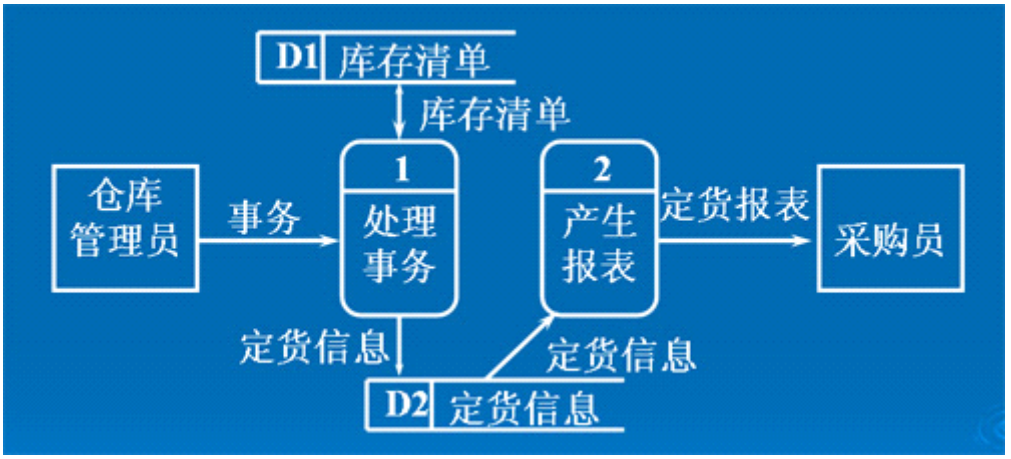
源点/终点 （外部实体）	• 采购员 • 仓库管理员
数据处理	• 产生报表 • 处理事务
数据流	• 定货报表 零件编号 零件名称 定货数量 目前价格 主要供应商 次要供应商 • 事务 零件编号 事务类型 数量
数据存储	• 定货信息（见定货报表） • 库存清单 零件编号 库存量 库存量临界值

步骤

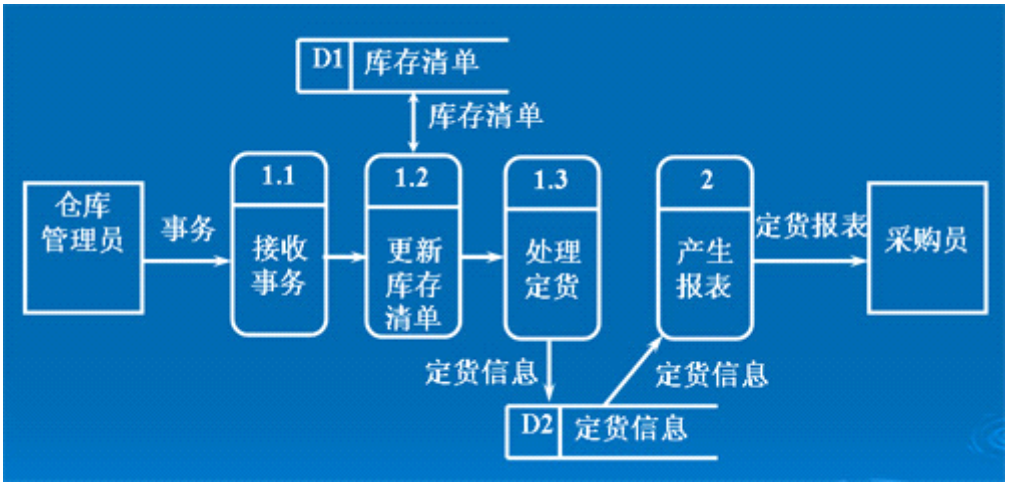
第一步，画出最概括的系统模型。因为任何系统实质上都是由若干个数据源点/终点以及一个处理组成。这个处理就代表了系统对数据加工变换的基本功能



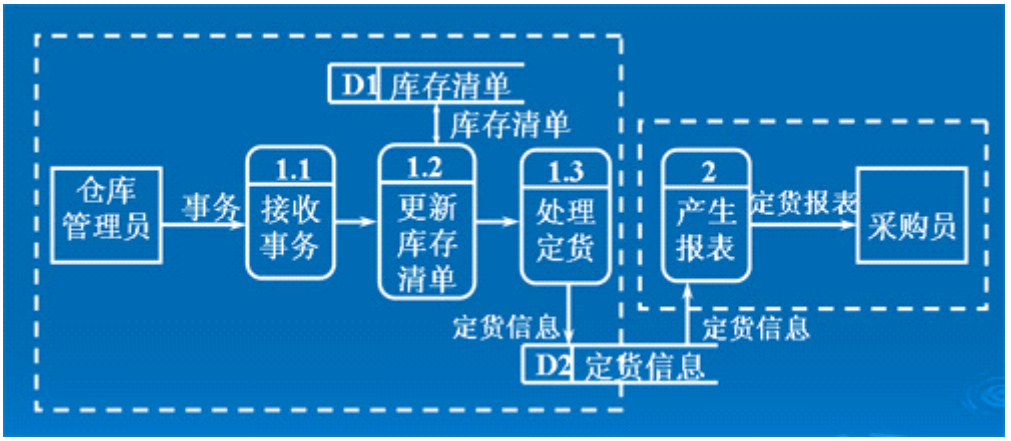
对上图进行细化



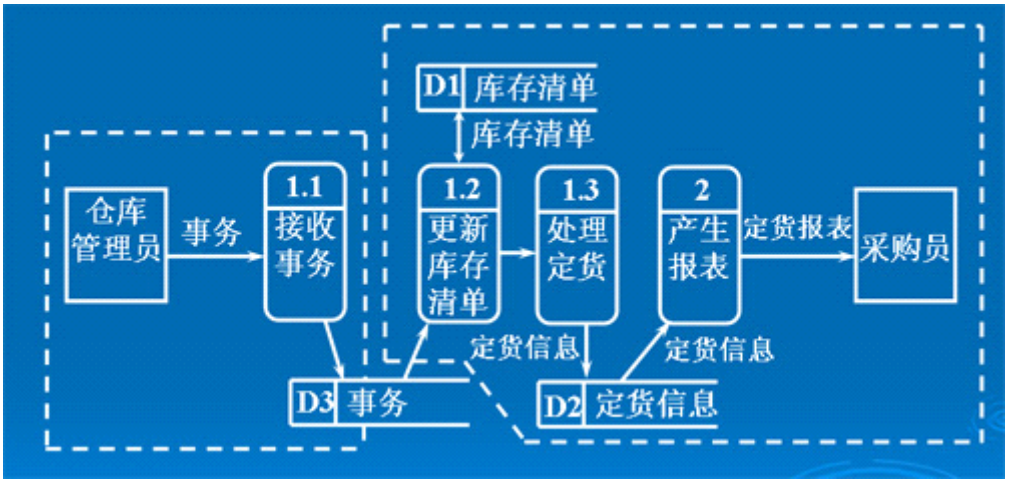
再细化一点



勾画出边界

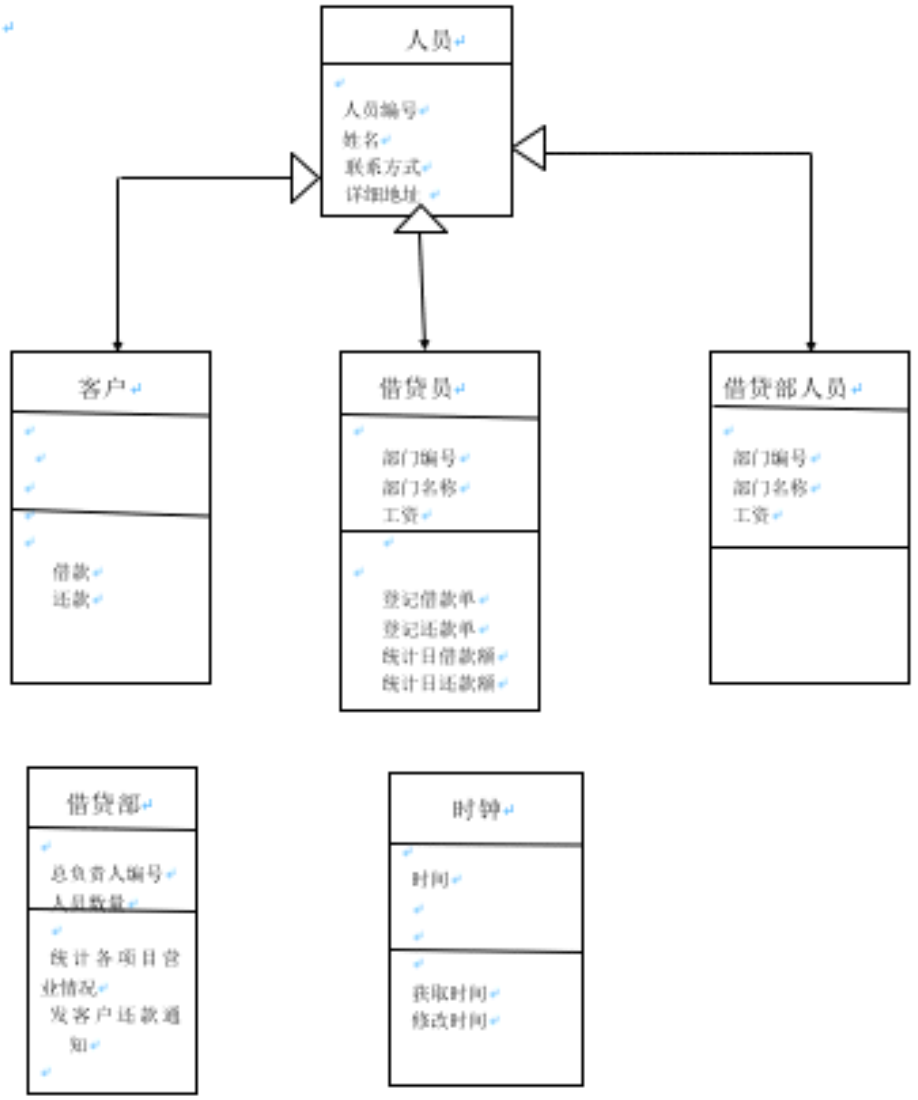

D1 库存清单
库存清单
仓库管理员
事务
1.1 接收事务
1.2 更新库存清单
1.3 处理定货
2 产生报表
定货报表
采购员
定货信息
定货信息
D2 定货信息

最后细化一下


D1 库存清单
库存清单
仓库管理员
事务
1.1 接收事务
1.2 更新库存清单
1.3 处理定货
2 产生报表
定货报表
采购员
定货信息
定货信息
D3 事务
D2 定货信息

类

该系统主要的类：人员，客户，借贷员，借贷部人员，



PDL（过程设计语言）

PAD 图

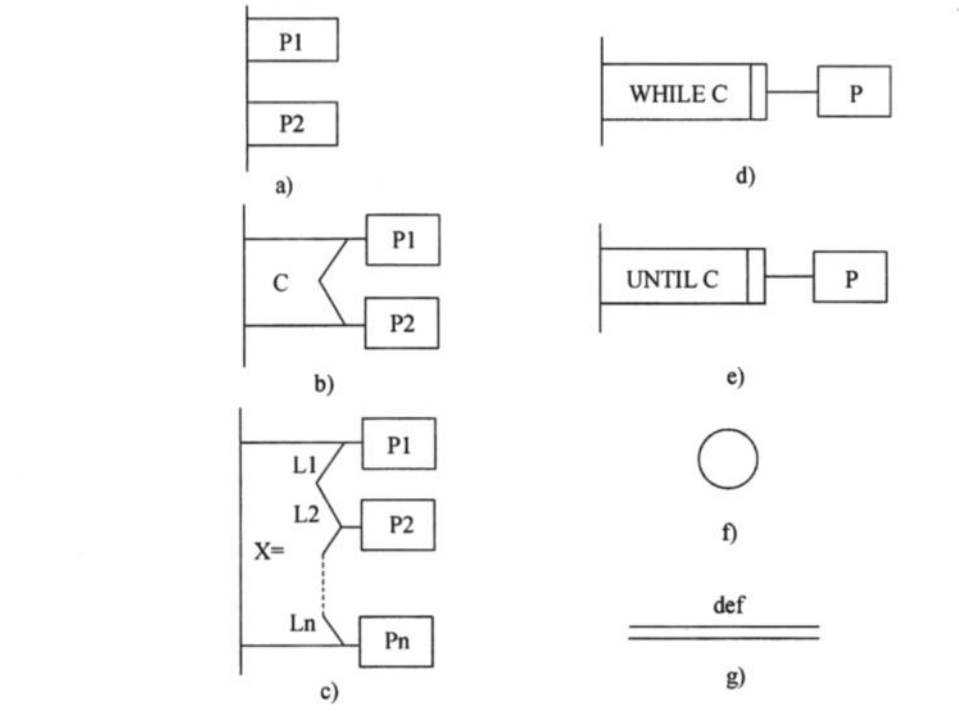


图 3-52　PAD 图的基本符号

a）顺序（先执行 P1 后执行 P2）　b）选择（IF C THEN P1 ELSE P2）　c）CASE 型多分支　d）WHILE 型循环（WHILE C DO P）　e）UNTIL 型循环（REPEAT P UNTIL C）　f）语句标号　g）定义

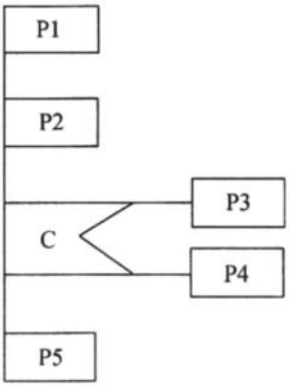


图 3-53　初始的 PAD 图

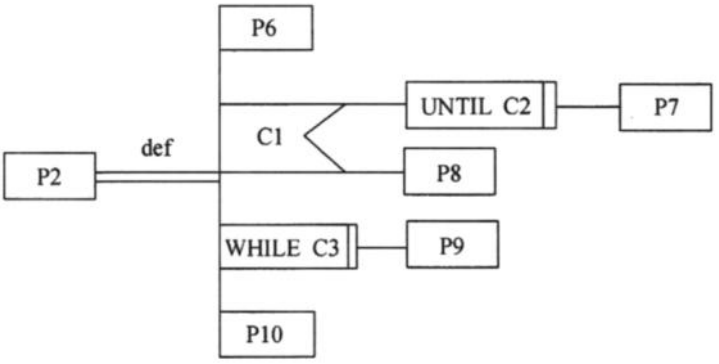


图 3-54　使用“def”符号细化处理 P2

NS 图

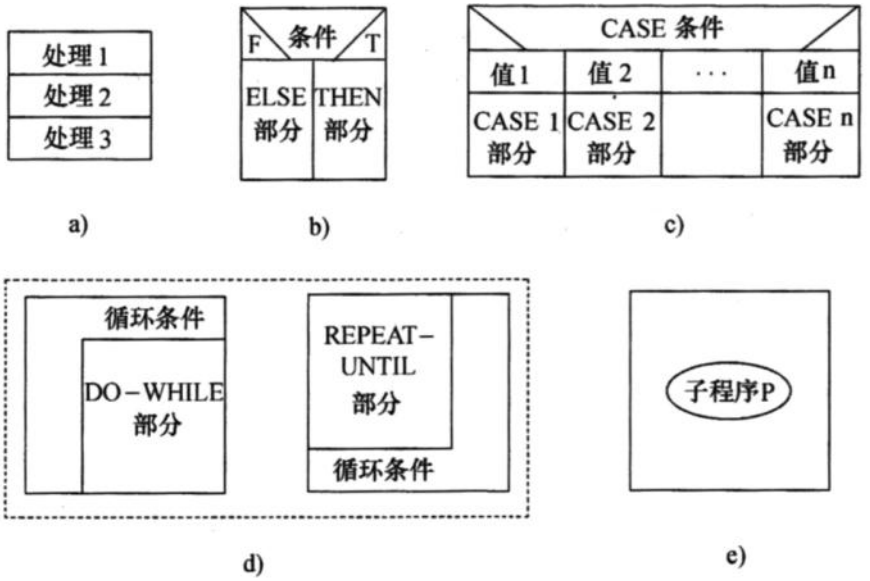
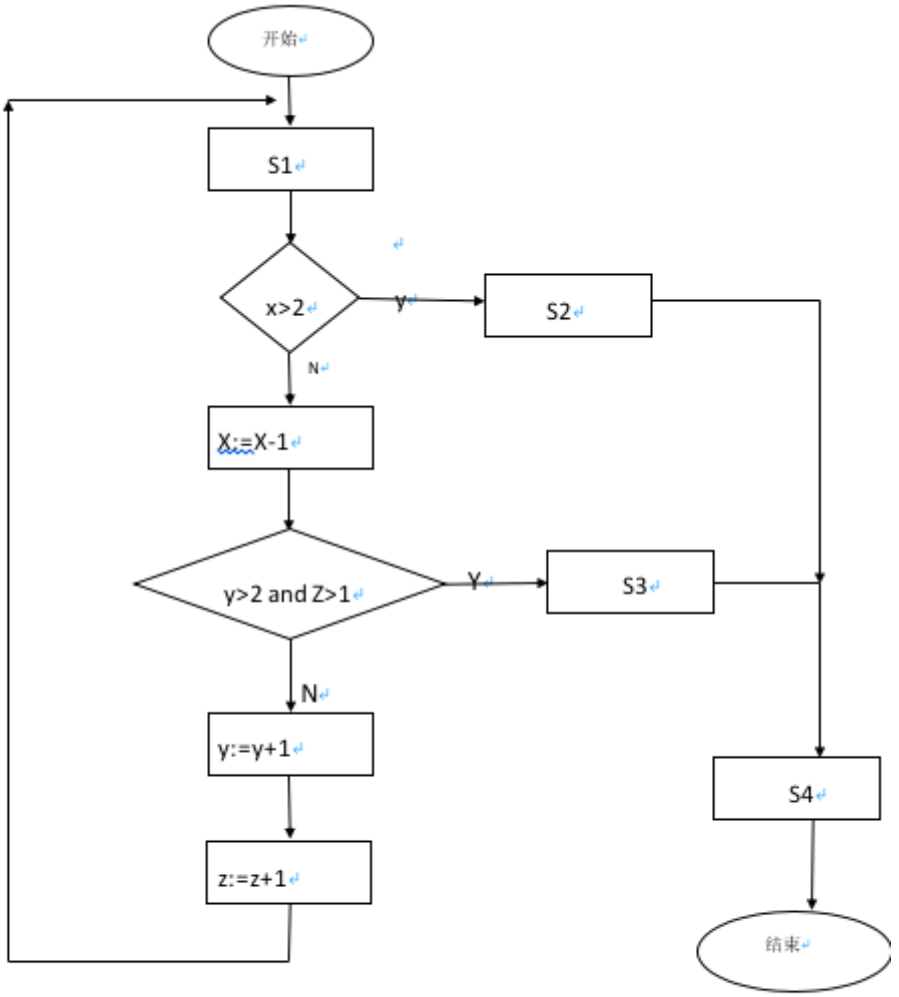


图 3-51　盒图的基本符号

a）顺序　b）IF-THEN-ELSE 型分型　c）CASE 型多分支　d）循环　e）调用子程序 P

程序流程图



分支覆盖：x=3,y=3,z=2　　x=1,y=1,z=1　　x=2,y=3,z=2

条件组合覆盖：　x=3,y=3,z=2　　x=2,y=2,z=1　　x=2,y=3,z=2　　x=2,y=3,z=1

x=2,y=2,z=2

示例

if A and B then Action1
if C or D then Action2

①语句覆盖最弱，只需要让程序中的语句都执行一遍即可 。上例中只需设计测试用例使得 A=true B=true C=true 即可

②分支覆盖又称判定覆盖：使得程序中每个判断的取真分支和取假分支至少经历一次，即判断的真假均曾被满足。上例需要设计测试用例使其分别满足下列条件即可

（1）A=true，B=true，C = true，D=false

（2）A=true，B=false，C = false，D=false

③条件覆盖：要使得每个判断中的每个条件的可能取值至少满足一次。上例中第一个判断应考虑到A=true，A=false，B=true，B=false第二个判断应考虑到C = true，C = false，D=true，D=false，所以上例中可以设计测试用例满足下列条件

（1）A=true，B=true，C = true，D=true

（2）A=false，B=false，C = false，D=false

④路径覆盖：要求覆盖程序中所有可能的路径。所以可以设计测试用例满足下列条件

（1）A=true，B=true，C = true，D=true

（2）A=false，B=false，C = false，D=false

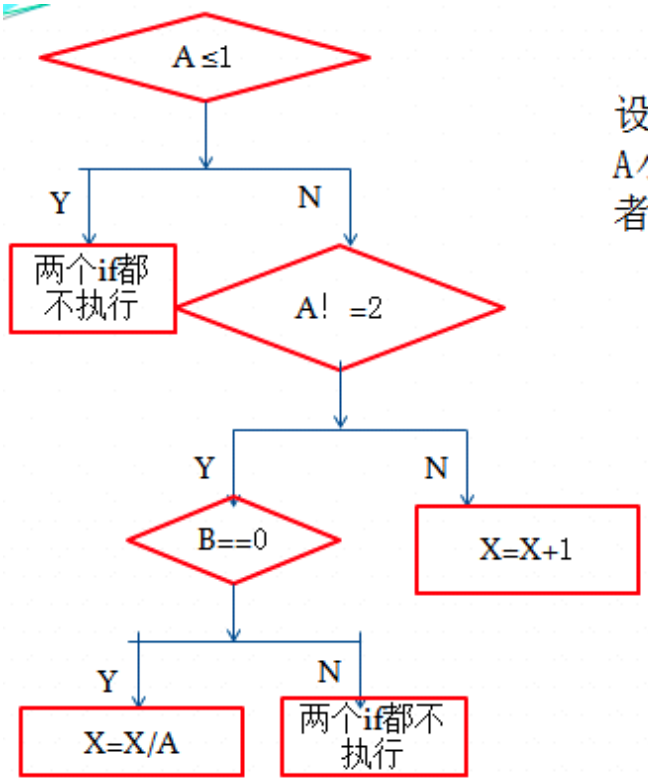
（3）A=true，B=true，C = false，D=false

（4）A=false，B=false，C = true，D=true

对于以下程序段：

```
1  void f(int X,int A,int B)
2  {
3    if((A>1)&&(B==0))
4      X=X/A;
5    if((A=2)||(x>1))
6      X=X+1;
7  }
```

为了达到100%的路径覆盖率至少需要设计几个测试用例



设计3个测试用例，分别对应A小于等于1，A大于1小于2或者A大于2，A是否等于2