程序示例

从第 5 章至第 9 章将用 3 个程序实例来描述各种单元测试方法，这 3 个程序分别是三角形问题（软件测试界的经典问题），复杂逻辑函数——NextDate 问题（给出第二天的日期），以及一个典型的管理信息系统应用问题——佣金问题。将这 3 个程序结合在一起，可以呈现出单元层次上软件测试所能遇到的绝大多数问题。在第 11 章至第 17 章讨论较高层次测试时要用到另外 4 个程序：简化的自动柜员机（ATM）系统 SATM；货币转换器，一个典型基于图形用户界面（GUI）的事件驱动式应用；Saturn 汽车风挡雨刷控制器；以及车库门控制器，可以用来阐述“系统的系统”（复杂系统）中的一些问题。

为了研究基于代码的测试，本章给出了 3 个单元层次程序实例的伪代码。第 11 章至第 17 章将在系统层面上给出 SATM 系统、货币转换器、雨刷控制器以及车库门控制器的描述。这些应用将采用有限状态自动机、事件驱动 Petri 网、状态图以及统一建模语言（UML）来进行建模。

2.1 通用伪代码

伪代码是表现程序源代码的一种“独立于语言”的形式。这里采用的伪代码均借用了一些 Visual Basic 语言的要素，设计成了两个层次的结构：单元层次结构和程序组件层次结构。这里的单元既可以理解为传统的组件（过程、函数等），也可以理解为面向对象的组件（类、对象等）。这种处理方法并不是很正规，表达式、变量表和字段描述等很多概念都没有正式定义就使用了。尖括号（<>）中的项表示的是在此位置上可以使用的语言元素。利用伪代码的另外一部分好处在于可以忽略很多无关紧要的细节，本书中我们就用自然语言来表述那些既正式又复杂的条件（见表 2-1）。

表 2-1 通用伪代码

语言单元	通用伪代码结构
注释	'< 说明文字 >
数据结构声明	Type < 类型名称 > < 字段描述列表 > End < 类型名称 >
数据声明	Dim< 变量 >As< 类型 >
赋值语句	< 变量 > = < 表达式 >
输入	Input(< 变量列表 >)
输出	Output(< 变量列表 >)
简单条件	< 表达式 > < 关系操作符 > < 表达式 >
复合条件	< 条件 > < 逻辑连接符 > < 条件 >
序列	语句按串行顺序排列

简单选择	If < 条件 > Then
	<then 语句 >
	EndIf

（续）

语言单元	通用伪代码结构
选择	If < 条件 > 　Then<then 语句 > 　Else<else 语句 > EndIf
多重选择	Case < 变量 > Of 　Case 1：< 谓词 > 　　<case 子句 > … Case n：< 谓词 > 　<case 子句 > EndCase
计数器控制的循环	For < 计数器 > = < 开始 > To < 结束 > 　　< 循环体 > EndFor
预测试循环	While < 条件 > 　　< 循环体 > EndWhile
后测试循环	Do 　< 循环体 > Until < 条件 >
过程定义（对函数和面向对象方法均可）	< 过程名称 >（ Input:< 变量列表 >; Output:< 变量列表 >) 　< 主体 > End< 过程名称 >
过程间通信	Call < 过程名称 >（ < 变量列表 >; < 变量列表 >)
类 / 对象的定义	< 名称 >（ < 属性列表 >; < 方法列表 >，< 主体 >) End < 名称 >
单元间通信	msg < 目标对象名 >.< 方法名 > (< 变量列表 >)
对象创建	Instantiate < 类名 >.< 对象名 >(< 属性值列表 >)
对象析构	Delete < 类名 >.< 对象名 >
程序	Program < 程序名称 > 　< 单元列表 > End< 程序名称 >

2.2　三角形问题

三角形问题是软件测试文献中最常使用的程序例子。在软件测试 30 年的历程中，一些有重要影响的文献主要有：Gruenberger（1973）、Brown and Lipov（1975）、Myers（1979）、Pressman（1982）及后续版本、Clarke（1983，1984）、Chellappa（1987）和 Hetzel（1988）等。当然还有很多其他文献，但上述这些已足以说明问题了。

2.2.1　问题描述

1. 初级版本

三角形程序将接受三个整数输入 a、b 和 c，分别代表三角形的三条边。程序输出为这三条边所构成的三角形的类型，即等边三角形（Equilateral）、等腰三角形（Isosceles）、一般

三角形（Scalene）或非三角形四类（NotATriangle），有时也把直角三角形作为第五类，在某些习题中我们也会用到它。

2. 升级版本

三角形程序将接受三个整数输入 *a*、*b* 和 *c*，分别代表三角形的三条边。整数 *a*、*b* 和 *c* 应满足以下条件：

c1. $1 \le a \le 200$	c4. $a < b + c$
c2. $1 \le b \le 200$	c5. $b < a + c$
c3. $1 \le c \le 200$	c6. $c < a + b$

程序的输出是根据三条边所确定的三角形类型：等边三角形（Equilateral）、等腰三角形（Isosceles）、一般三角形（Scalene）或非三角形四类（NotATriangle）。如果任何一个输入数值不能满足 c1、c2 或 c3 这三个条件中的任何一个，程序将会输出一条消息来提示这种情况，例如：“Value of b is not in the range of permitted values”（*b* 的取值不在允许范围之内）。如果 *a*、*b* 和 *c* 的取值均能满足条件 c1、c2 和 c3，程序给出以下 4 种结论之一：

（1）如果三条边全部相同，程序输出结果为等边三角形。

（2）如果恰好有一对边相同，程序输出结果为等腰三角形。

（3）如果不存在相等的取值，程序输出结果为一般三角形。

（4）如果条件 c4、c5 和 c6 中存在不能满足的情况，程序输出结果为非三角形。

2.2.2 三角形问题的讨论

三角形问题在软件测试界能够经久不衰的原因之一在于它包含了既明确而又复杂的逻辑关系。它也十分典型地表现出定义的不完整会如何严重影响客户、开发人员和测试人员之间的有效沟通。第一个版本的规格说明就假设开发人员事先是了解三角形的基本知识的，特别是三角不等式：两边之和必须大于第三边才能构成三角形。选择 200 作为边长的上限完全是为了方便随意取的，在第 5 章构造边界值测试用例时我们还会用到这个限制。

2.2.3 三角形问题的经典实现

作为本书程序示例的第一个，这个三角形程序的“经典”实现很类似于 FORTRAN 语言。图 2-1 给出了这种实现的流程图。图 2-2 给出了升级版的流程图。流程图中各处理框的编号对应于接下来给出的程序伪代码中注释号（类似 FORTRAN）。（此处的编号同 Pressman（1982）所采用的一致。）这种实现形式展示了它的历史传承，在 2-2-4 节中还会给出一种更好的实现。

这里采用变量 `match` 来记录每对边相等的情况。FORTRAN 风格典型的复杂性被变量 `match` 表现出来了：可以看出，三个关于三角不等式测试一个都没用到。如果有两条边相等，如 `a` 和 `c`，那么只需要测试 `a+c` 与 `b` 关系即可。（因为 `b` 一定大于 0，`a+b` 一定大于 `c`，因为 `c` 与 `a` 相等。）显然这种方法可以有效减少所需的比较次数。这种实现方法效率高，但也损失了程序的易懂性，测试的难度也加大了。在后面研究程序执行的不可行路径时，我们会发现这还是非常有用。这是本书保留它的最大原因。在此，有 6 条路径可以到达“非三角形”框（12-1 至 12-6），有 3 条达到“等腰三角形”框（15-1 至 15-3）。

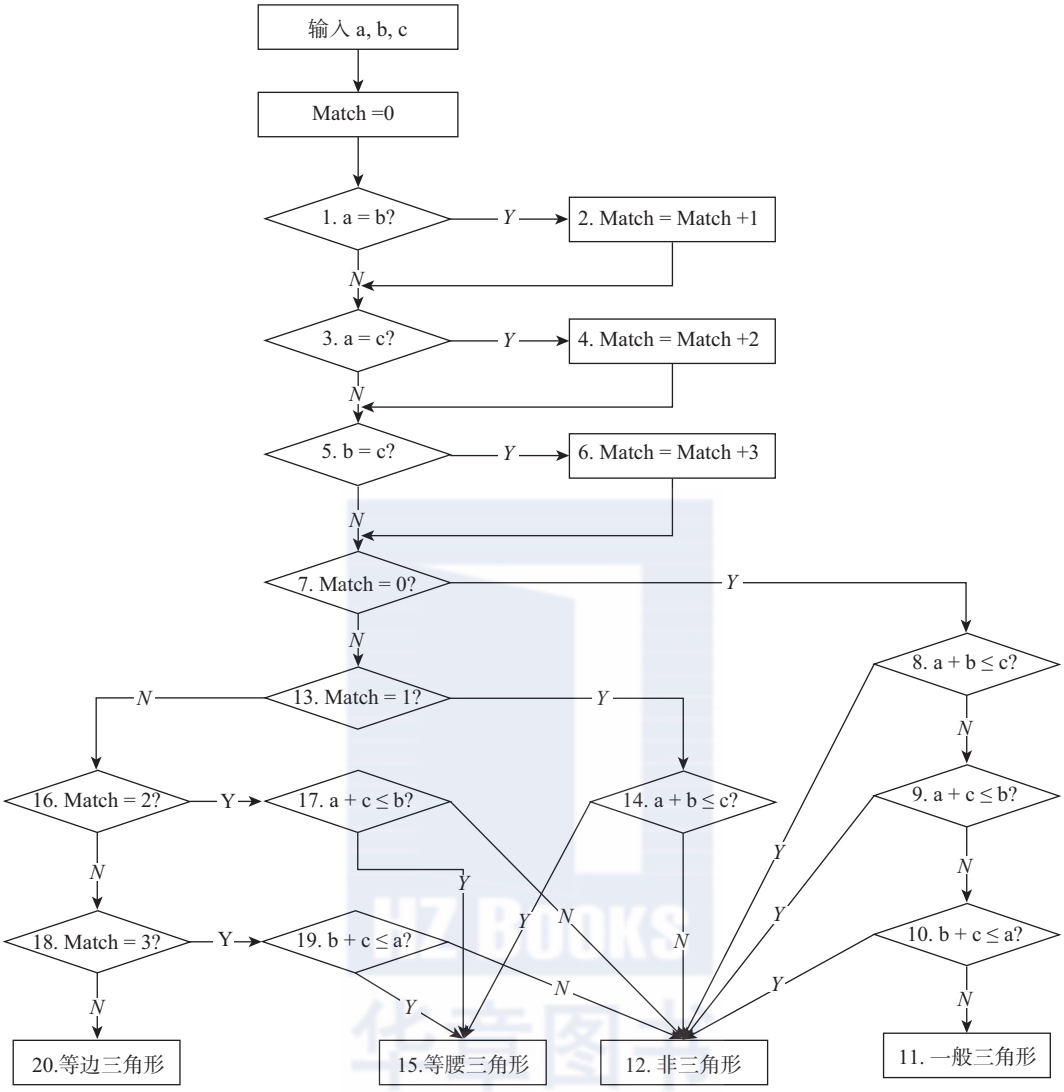

图 2-1　三角形问题的经典实现流程图

下面给出伪代码。

```
Program triangle1 'Fortran-like version
'
Dim a, b, c, match As INTEGER
'
Output("Enter 3 integers which are sides of a triangle")
Input(a, b, c)

Output("Side A is",a)
Output("Side B is",b)
Output("Side C is",c)
match = 0
If a = b                                                    '(1)
   Then match = match + 1                                   '(2)
EndIf
If a = c                                                    '(3)
   Then match = match + 2                                   '(4)
```

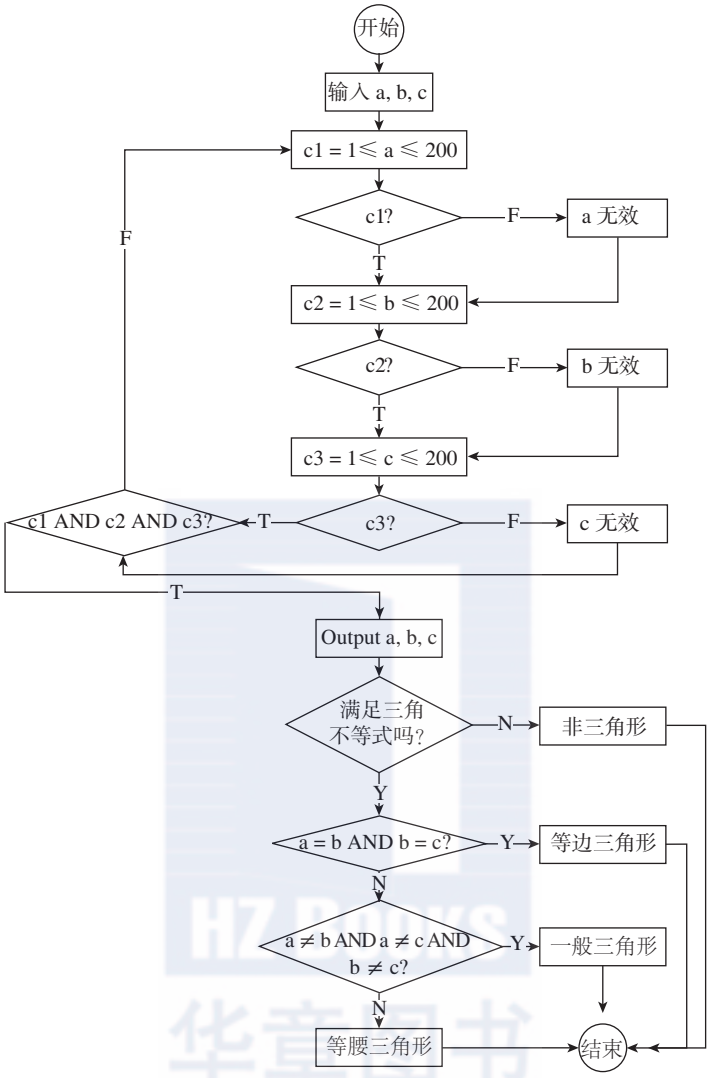

图 2-2　升级版三角形问题的实现流程图

```
EndIf
If b = c                                                    '(5)
   Then match = match + 3                                   '(6)
EndIf
If match = 0                                                '(7)
   Then If (a + b)≤ c                                       '(8)
           Then Output("NotATriangle")                      '(12.1)
           Else If (b + c) ≤ a                              '(9)
                   Then Output("NotATriangle")              '(12.2)
                   Else If (a + c) ≤ b                      '(10)
                           Then Output("NotATriangle")      '(12.3)
                           Else Output ("Scalene")          '(11)
                        EndIf
                EndIf
        EndIf
```

```
    Else If match = 1                                          '(13)
            Then If (a + c) ≤ b                                '(14)
                    Then Output("NotATriangle")                '(12.4)
                    Else Output ("Isosceles")                  '(15.1)
                 EndIf
            Else If match=2                                    '(16)
                    Then If (a + c) ≤ b
                            Then Output("NotATriangle")        (12.5)
                            Else Output ("Isosceles")          '(15.2)
                         EndIf
                    Else If match = 3                          '(18)
                         Then If (b + c) ≤ a  '(19)
                            Then Output("NotATriangle")        '(12.6)
                            Else Output ("Isosceles")          '(15.3)
                         EndIf
                    Else Output ("Equilateral")                '(20)
                 EndIf
             EndIf
      EndIf
EndIf
'
End Triangle1
```

2.2.4 三角形问题的结构化实现

```
Program triangle2 'Structured programming version of simpler specification

Dim a,b,c As Integer
Dim IsATriangle As Boolean

'Step 1: Get Input
Output("Enter 3 integers which are sides of a triangle")
Input(a,b,c)
Output("Side A is",a)
Output("Side B is",b)
Output("Side C is",c)
'Step 2: Is A Triangle?'
If (a < b + c) AND (b < a + c) AND (c < a + b)
   Then IsATriangle = True
   Else IsATriangle = False
EndIf
'
'Step 3: Determine Triangle Type
If IsATriangle
   Then If (a = b) AND (b = c)
        Then Output ("Equilateral")
        Else If (a ≠ b) AND (a ≠ c) AND (b ≠ c)
                Then Output ("Scalene")
                Else Output ("Isosceles")
             EndIf
        EndIf
   Else Output("Not a Triangle")
EndIf
End triangle2
```

第三个版本

```
Program triangle3'
Dim a, b, c As Integer
Dim c1, c2, c3, IsATriangle As Boolean
```

```
'Step 1: Get Input
Do
    Output("Enter 3 integers which are sides of a triangle")
    Input(a, b, c)
    c1 = (1 ≤ a) AND (a ≤ 300)
    c2 = (1 ≤ b) AND (b ≤ 300)
    c3 = (1 ≤ c) AND (c ≤ 300)
    If NOT(c1)
         Then Output("Value of a is not in the range of permitted values")
    EndIf
    If NOT(c2)
         Then Output("Value of b is not in the range of permitted values")
    EndIf
    If NOT(c3)
         ThenOutput("Value of c is not in the range of permitted values")
    EndIf
Until c1 AND c2 AND c3
Output("Side A is",a)
Output("Side B is",b)
Output("Side C is",c)
'Step 2: Is A Triangle?
If (a < b + c) AND (b < a + c) AND (c < a + b)
    Then IsATriangle = True
    Else IsATriangle = False

EndIf
'Step 3: Determine Triangle Type
If IsATriangle
    Then If (a = b) AND (b = c)
            Then Output ("Equilateral")
            Else If (a ≠ b) AND (a ≠ c) AND (b ≠ c)
                    Then Output ("Scalene")
                    Else Output ("Isosceles")
                 EndIf
         EndIf
    Else Output("Not a Triangle")
EndIf
End triangle3
```

2.3 NextDate 日期函数

三角形程序的复杂性源于输入与正确输出结果之间的关系。下面我们通过 NextDate 函数来讲解另一个复杂问题——输入变量之间的逻辑关系。

2.3.1 问题描述

NextDate 是一个拥有 month、date 和 year 三个输入变量的函数，给出输入日期后面一天的日期。显然变量 month、date 和 year 均取整数值，并且应满足如下条件（年份范围结束在 2012 年是任意选取的，并且是来自第一版）：

c1. 1 ≤ month ≤ 12
c2. 1 ≤ day ≤ 31
c3. 1812 ≤ year ≤ 2012

同处理三角形程序一样，还可以使我们的问题陈述更加明确，要求定义出程序对输入变量 month、day 和 year 无效取值的响应，定义程序对输入变量无效逻辑组合的响应。例如，对每年 6 月 31 日的响应方法。如果条件 c1、c2 或 c3 中有任意一条不满足，则 NextDate 函

数应该给出一条输出来提示相应变量的取值不在允许范围内。例如，“Value of month not in the range 1...12”。由于存在许多无效的日 – 月 – 年的组合，NextDate 函数根据所有此类情况合并为一个提示信息：“Invalid Input Date”。

2.3.2 NextDate 函数的讨论

NextDate 函数之所以复杂有两个原因：一是前面讨论过的输入域复杂性，二是判断某一年是否为闰年的判别规则的复杂性。每一年实际上有 365.2422 天，因此要设立闰年来解决这个“多余天数”的问题。如果每四年一个闰年，还会有一点点的误差。罗马历法（尤其是格里高利教皇之后）采用每一百年调整闰年设置的办法来解决这个问题。方法是，如果年份数值可以被 4 整除，并且不是整世纪年，则该年为闰年；世纪年只有是 400 的倍数时才为闰年（Inglis，1961）。所以，1992 年、1996 年和 2000 年都是闰年，而 1900 年不是。NextDate 函数还能说明软件测试领域的一个常见情况，我们常常能看到 Zipf 定律的具体实例，Zipf 定律说明 80% 的活动发生在 20% 的空间中。留心观察一下这里处理闰年问题需要源代码的量以及在第二种实现形式中检查输入值的有效性所使用的源代码的量。

2.3.3 NextDate 函数的实现

```
Program NextDate1 'Simple version
Dim tomorrowDay,tomorrowMonth,tomorrowYear As Integer
Dim day,month,year As Integer
Output ("Enter today's date in the form MM DD YYYY")
Input (month, day, year)
Case month Of
Case 1: month Is 1,3,5,7,8, Or 10: '31 day months (except Dec.)
 If day < 31
   Then tomorrowDay = day + 1
   Else
    tomorrowDay = 1
    tomorrowMonth = month + 1
 EndIf
Case 2: month Is 4,6,9, Or 11 '30 day months
 If day < 30
   Then tomorrowDay = day + 1
   Else
    tomorrowDay = 1
    tomorrowMonth = month + 1
 EndIf
Case 3: month Is 12: 'December
 If day < 31
   Then tomorrowDay = day + 1
   Else
    tomorrowDay = 1
    tomorrowMonth = 1
    If year = 2012
      Then Output ("2012 is over")
      Else tomorrow.year = year + 1
 EndIf
Case 4: month is 2: 'February
 If day < 28
   Then tomorrowDay = day + 1
   Else
    If day = 28
      Then If ((year is a leap year)
```

```
                Then tomorrowDay = 29 'leap year
                Else 'not a leap year
                  tomorrowDay = 1
                  tomorrowMonth = 3
              EndIf
    Else If day = 29
              Then If ((year is a leap year)
                     Then tomorrowDay = 1

                          tomorrowMonth = 3
                     Else 'not a leap year
                       Output("Cannot have Feb.", day)
                   EndIf
          EndIf
     EndIf
 EndIf
EndCase
Output ("Tomorrow's date is", tomorrowMonth, tomorrowDay, tomorrowYear)
End NextDate

Program NextDate2   Improved version
'
Dim tomorrowDay,tomorrowMonth,tomorrowYear As Integer
Dim day,month,year As Integer
Dim c1, c2, c3 As Boolean
'
Do
 Output ("Enter today's date in the form MM DD YYYY")
 Input (month, day, year)
 c1 = (1 ≤ day) AND (day ≤ 31)
 c2 = (1 ≤ month) AND (month ≤ 12)
 c3 = (1812 ≤ year) AND (year ≤ 2012)
 If NOT(c1)
  Then Output("Value of day not in the range 1..31")
 EndIf
 If NOT(c2)
  Then Output("Value of month not in the range 1..12")
 EndIf
 If NOT(c3)
  Then Output("Value of year not in the range 1812..2012")
 EndIf
Until c1 AND c2 AND c2

Case month Of
Case 1: month Is 1,3,5,7,8, Or 10: '31 day months (except Dec.)
 If day < 31
    Then tomorrowDay = day + 1
    Else
     tomorrowDay = 1
     tomorrowMonth = month + 1
 EndIf
Case 2: month Is 4,6,9, Or 11 '30 day months
 If day < 30
    Then tomorrowDay = day + 1
    Else
     If day = 30
        Then tomorrowDay = 1
             tomorrowMonth = month + 1
        Else Output("Invalid Input Date")
     EndIf
 EndIf
Case 3: month Is 12: 'December
```

```
  If day < 31
     Then tomorrowDay = day + 1
     Else
      tomorrowDay = 1
      tomorrowMonth = 1
      If year = 2012
         Then Output ("Invalid Input Date")
         Else tomorrow.year = year + 1
      EndIf
  EndIf
 Case 4: month is 2: 'February
  If day < 28
     Then tomorrowDay = day + 1
     Else
      If day = 28
         Then
          If (year is a leap year)
             Then tomorrowDay = 29 'leap day
             Else 'not a leap year
              tomorrowDay = 1
              tomorrowMonth = 3
          EndIf
         Else
          If day = 29
             Then
              If (year is a leap year)
                 Then tomorrowDay = 1
                      tomorrowMonth = 3
                 Else
                  If day > 29
                     Then Output("Invalid Input Date")
                  EndIf
              EndIf
          EndIf
      EndIf
  EndIf
 EndCase
 Output ("Tomorrow's date is", tomorrowMonth, tomorrowDay, tomorrowYear)
 '
 End NextDate2
```

2.4 佣金问题

第三个实例佣金问题是典型的商务计算问题，其中包含了计算和决策等步骤，可以引申出许多重要的测试问题。这个例子的主要用途是讨论数据流和基于切片的测试。

2.4.1 问题描述

从前有一位销售人员在亚利桑那州代销密苏里军械制造厂生产的步枪配件，包括枪机（lock）、枪托（stock）和枪管（barrel）。枪机售价 45 美元，枪托售价 30 美元，枪管售价 25 美元。销售人员每个月至少要卖出一个枪机，一个枪托和一个枪管（但是没有必要是一支完整的步枪），而制造厂的生产能力限制销售人员一个月最多只能卖出 70 个枪机、80 个枪托和 90 个枪管。每走访过一个城镇之后，销售人员都要给密苏里军械厂发一封电报，汇报在这一城镇中销售枪机、枪托和枪管的数量。销售人员月末会再发一封很短的电报，通知“-1 个枪机售出”。这样军械厂就知道当月的销售活动已经结束了，计算销售人员应得的佣金了。

佣金计算方法如下：销售总额1000美元以下（含1000美元）部分的佣金为10%，1000至1800美元之间部分的佣金为15%，超过1800美元的部分的佣金为20%。

2.4.2 佣金问题的讨论

在这个佣金问题示例中，我们一下子就能看明白佣金的计算方法。在现实生活中会遇到其他一些有多个变量的累加函数，例如在填写us1040收入报税表时遇到的各种计算就是这样。（所以我们还是继续讨论步枪吧。）这个佣金程序可分为三个部分：输入数据处理部分，验证输入数据的有效性（同三角形问题和NextDate函数一样），销售额统计计算部分，以及佣金计算部分。此处我们省略了对输入数据有效性的验证，借用了典型的管理信息系统（MIS）的数据采集功能中所常用的条件循环语句While来模拟对电报的处理。

2.4.3 佣金问题的实现

```
Program Commission (INPUT,OUTPUT)
'
Dim locks, stocks, barrels As Integer
Dim lockPrice, stockPrice, barrelPrice As Real
Dim totalLocks,totalStocks,totalBarrels As Integer
Dim lockSales, stockSales, barrelSales As Real
Dim sales,commission : REAL
'
lockPrice = 45.0
stockPrice = 30.0
barrelPrice = 25.0
totalLocks = 0
totalStocks = 0
totalBarrels = 0
'
Input(locks)
While NOT(locks = -1)      'Input device uses -1 to indicate end of data
    Input(stocks, barrels)
    totalLocks = totalLocks + locks
    totalStocks = totalStocks + stocks
    totalBarrels = totalBarrels + barrels
    Input(locks)
EndWhile
'
Output("Locks sold:", totalLocks)
Output("Stocks sold:", totalStocks)
Output("Barrels sold:", totalBarrels)
'
lockSales = lockPrice * totalLocks
stockSales = stockPrice * totalStocks

barrelSales = barrelPrice * totalBarrels
sales = lockSales + stockSales + barrelSales
Output("Total sales:", sales)
'
If (sales > 1800.0)
    Then
       commission = 0.10 * 1000.0
       commission = commission + 0.15 * 800.0
       commission = commission + 0.20 * (sales-1800.0)
    Else If (sales > 1000.0)
            Then
```

```
                commission = 0.10 * 1000.0
                commission = commission + 0.15*(sales-1000.0)
            Else commission = 0.10 * sales
        EndIf
EndIf
Output("Commission is $",commission)
End Commission
```

2.5　SATM 系统

我们需要一个涉及面更广的实例来更好地讨论集成测试和系统测试的有关问题（见图 2-3 ）。

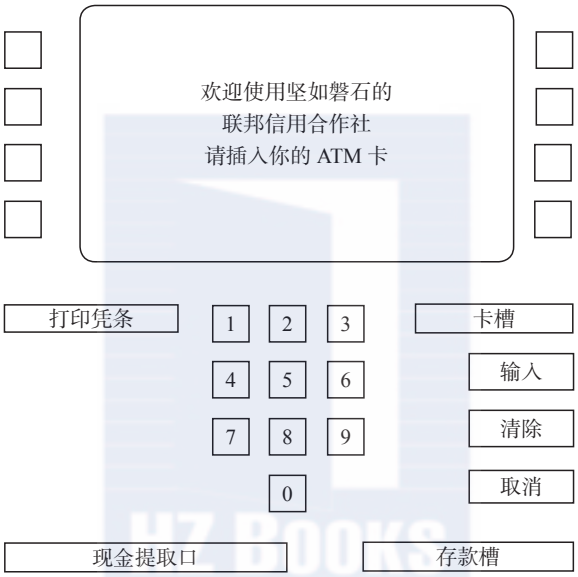

图 2-3　SATM 终端

这里给出的自动柜员机（ATM）实例是极小的，但是它却包含了典型的 CS（客户端 – 服务器）系统中在客户端会涉及的大量功能和交互操作。

2.5.1　问题描述

SATM 系统同银行客户之间的信息沟通采用图 2-4 所示的 15 种界面来实现，系统界面的主要特征如图 2-3 所示。客户可以在 SATM 上选用 3 种银行业务：存款、取款和余额查询。为简单起见，这些业务仅仅用于支票账户。

客户来到 SATM 机前，系统显示界面 1。每个客户使用标有个人账户编号（PAN 码）的银行卡来使用 SATM 机，PAN 码是打开系统内客户账户文件的关键，账户文件中包含了客户姓名和账户信息等内容。如果客户 PAN 码与某个账户文件一致，系统就向客户显示界面 2。如果没有找到相同 PAN 码的账户文件，系统就显示界面 4，并收走该银行卡。

在界面 2 中，系统提示客户输入个人身份编号（PIN 码）。如果所输入的 PIN 码正确（即同账户文件中的信息一致），则系统显示界面 5，否则显示界面 3。这里客户一共有 3 次机会来输入正确的 PIN 码，3 次均失败后，系统将显示界面 4，并收走该银行卡。

处于界面 5 时，客户在界面 5 所显示的选项中选择所需业务。如果要查询余额，系统显

示界面 14。如果要存款，系统首先要检查终端控制文件中的一个字段，借此来确定存款信封槽的状态。如果存款信封槽没有问题，系统显示界面 7，获取存款金额。如果有问题，系统显示界面 12。成功输入存款金额后，系统显示界面 13，接收存款信封，处理存款。之后系统显示界面 14。

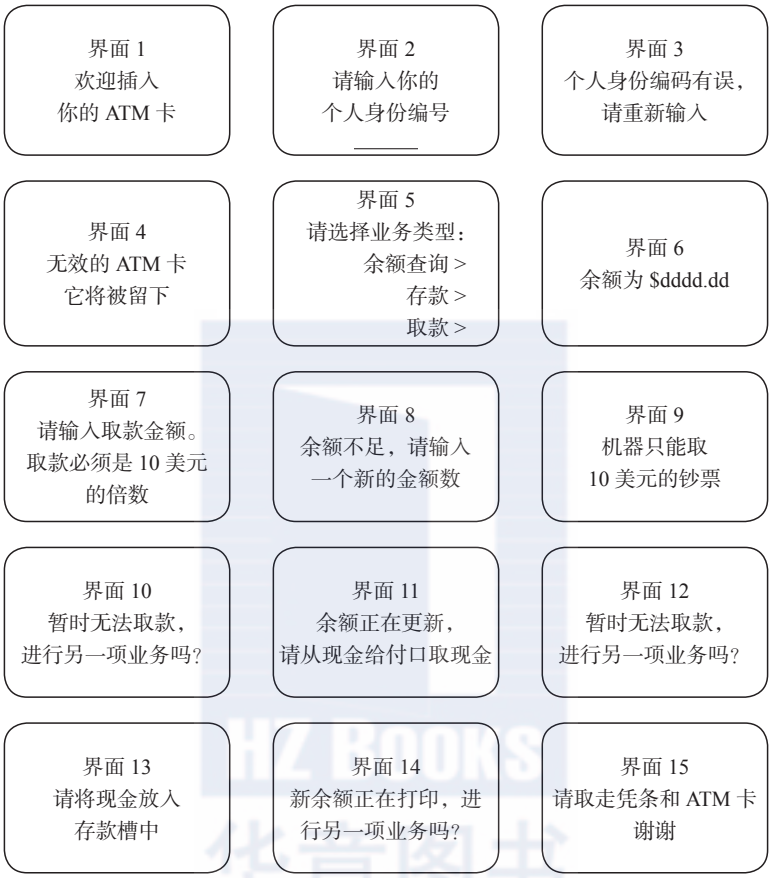

图 2-4　SATM 的各个界面

如果要取款，系统首先要检查终端控制文件中的取款通道状态字段（判断通道是否可用）。如果通道堵塞，系统显示界面 10；否则显示界面 7，等待客户输入取款金额。成功输入取款金额后，系统还要检查终端文件状态，核实是否有足够的现金。如果现金不足，则显示界面 9，否则进一步处理取款业务。系统检查客户的余额（与余额查询业务的处理过程相同）。如果账户余额不足，则显示界面 8；如果资金充足，则显示界面 11 并付出现金。账户余额被打印在业务凭条上，操作同余额查询业务的处理。客户把钱取走后，系统显示界面 14。

在界面 10、12 或 14 中，如果客户选择“否”，系统会显示界面 15，并退出客户银行卡。从卡槽中取走卡后，系统显示界面 1。在界面 10、12 或 14 中，如果客户选择“是”，系统将显示界面 5，这样客户就可以选择其他业务。

2.5.2　SATM 系统的讨论

在上面给出的系统描述中，有大量的信息都被“隐藏”起来了。比如，细心的读者会发

现该 ATM 终端里只有 10 美元面值的钞票（见界面 7）。这个定义可能比我们通常见到的更精确。这是对实际的 ATM 机有意进行的简化（因此取名 SATM）。

其他问题可以通过一系列假设来解决。例如，是否需要定义取款的上限？如果客户能使用多个 ATM 终端，怎样才能防止取款金额超过其账户的实际余额？还会有一些初始化方面的问题，如最初要在机器中放多少现金？如何在系统中添加新客户？为了简单起见，这里对这些实际应用中的具体问题就不再进一步讨论了。

2.6　货币兑换计算器

货币兑换计算程序也是一个事件驱动程序，但它更侧重与图形用户界面（GUI）相关的代码。图 2-5 给出了一个简单的 GUI。

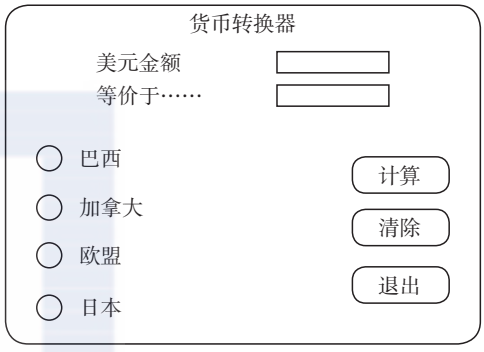

图 2-5　货币转换器图形用户界面

这个应用程序可以将美元转换为下面 4 种货币中的任意一种：巴西雷亚尔、加拿大元、欧元或日元。用“单选按钮”（选项按钮）来选择国别，这些按钮是互斥的。选择了一个国家后，系统的响应是补全的相应提示语。例如，按下了“加拿大”按钮后，“可兑换……”的提示就会变成“可兑换加拿大元”。同时程序还会在可转换金额的输出位置旁边显示一面加拿大国旗。选择币种前后，用户都可以输入美元金额。两项操作完成后，用户可以点击“计算”“清除”或“退出”按钮。点击“计算”会将美元金额转换为所选货币的兑换金额。点击“清除”可以重新设置币种、美元金额、转换金额以及相应的提示。点击“退出”会结束该应用程序。这个货币转换器的例子在第 15 章中很好地说明 UML 描述和面向对象实现。

2.7　雨刷控制器

土星牌汽车风挡雨刷是由一个带刻度盘的控制杆来操控的，其中控制杆有 OFF（关）、INT（间歇）、LOW（低速）和 HIGH（高速）4 个位置，刻度盘有 3 个位置，分别用数字 1、2 和 3 表示。刻度盘位置指示 3 种间歇速度，并且只有控制杆处于 INT 位置时刻度盘位置才有意义。如表 2-2 所示的决策表给出了控制杆位置和刻度盘位置所对应的风挡雨刷的实际工作速度（每分钟摆动的次数）。

表 2-2　控制杆和刻度盘位置所对应的雨刷速度

c1. 控制杆位置	停止	间歇	间歇	间歇	低速	高速
c2. 刻度盘位置	—	1	2	3	—	—
a1. 雨刷速度	0	4	6	12	30	60

2.8　车库门遥控开关

打开车库门的系统有以下几个部分组成：一个驱动马达、一个驱动链、车库门轮距、灯，以及一个电子控制器。系统的大部分是由 110V 的工业电源驱动。几个设备与车库门控制器相连通，这些设备是：一个无线小键盘（通常安装在汽车里），一个安装在车库门外面的数字键盘，以及一个固定在墙上的按钮。此外，还有两个安全设备：一个靠近地板的激光

束和一个障碍传感器。仅仅当车库门正在关闭时，后面两个设备才会运转。如果光束被打断（可能被一个宠物打断），这个门会立即停止，然后反向，直到这个门完全打开。如果当门正在关闭时，遇到了一个障碍（假如一个孩子的三轮车落在了门口），这个门停止，并且反向，直到它完全打开。当门正在关闭或者正在打开时，还有一种方法阻止它运转。任何一种设备（无线键盘，数字键盘和固定在墙上的控制按钮）都会发出一种信号。对任何一种信号的反应都是不同的，即：这个门会停在适当的位置上。任何一种设备发出来的随后的信号会在门停止的相同方向上启动它。有一些传感器可以检测门什么时候移动到一个极限位置，即：完全开着或者完全关闭的状态。当门运转时，灯是亮着的，并且当这个门到下一个极限状态时，灯会持续亮大概 30 秒。

除这个基本的车库门遥控开关之外，这三个指示装置和安全设备是可选的。在第 17 章，这个例子将会被用在系统的系统中的探讨。现在，一个车库门遥控开关的系统建模语言（SysML）环境图如图 2-6 所示。

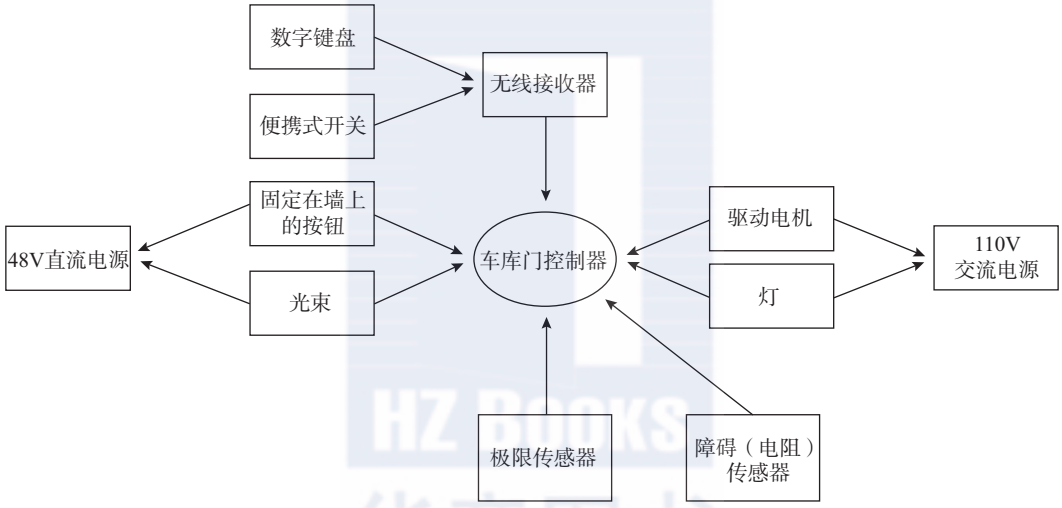

图 2-6　车库门控制器的 SysML 图

2.9　习题

1. 重新考察图 2-1 所给出的三角形程序的经典流程图。变量 `match` 的取值可以是 4 或 5 吗？有可能依次“执行”方框 1、2、5 和 6 吗？
2. 回顾第 1 章中对程序的规格说明的规定行为和程序实现的实际行为之间关系的讨论。如果仔细研究 NextDate 函数的程序实现，你就会发现一个问题：处理 30 天月份（4 月、6 月、9 月和 11 月）的 CASE 子句中，没有对 `day=31` 情况的处理。请讨论这个实现是否正确。处理 2 月时，CASE 子句没有对 `day=29` 情况的处理，请讨论这个实现是否正确。
3. 在第 1 章中曾提到测试用例的一部分是期望输出。请给出 NextDate 函数对 1812 年 6 月 31 日的测试用例的期望输出是什么？为什么？
4. 对三角形问题的一种常见的扩展是检查是否为直角三角形。如果三条边满足勾股定理（即 $c^2=a^2+b^2$），则为直角三角形。此时要求按递增的顺序给出各条边，即应有 $a \leqslant b \leqslant c$。请扩展 Triangle3 程序来处理直角三角形。在后面的习题中还要再次进行这种扩展。
5. 边长分别为 −3、−3 和 5 时，Triangle2 程序会怎么处理呢？请采用第 1 章中给出的思想方法来讨论这个问题。

6. 计算前一天日期的函数 YesterDate 函数是 NextDate 函数的逆函数。对给定的 `year`、`month`、`day`，YesterDate 函数应返回这一天的前一天的日期。把这个问题作为本章中实例的一种扩展，请采用你喜欢的语言（或伪代码）编写一个 YesterDate 程序。
7. 在 GUI 设计中，一部分技巧在于防止用户输入错误。事件驱动式的应用程序特别容易受到输入错误的影响，原因在于各种事件可能以任何顺序出现。在本章给出的 SATM 系统中，用户可能会输入美元金额后就马上点击“计算”按钮，而忘记选择国家。类似地，用户也可能选择国家就马上点击计算按钮，而没忘记输入美元金额。GUI 设计者可以利用一种称为“强制导航”的做法来避免此类问题的发生。在 Visual Basic 语言中可以利用控件的可见性，来实现这一点。请讨论如何实现之。
8. CRC 出版公司的网站 (http://www.crcpress.com/product/isbn/9781466560680) 上提供了本书的一些补充软件，这一系列软件大都是我在为研究生开设的“软件测试”课上使用过的。其中的第一部分是使用 naive.xls 程序（在绝大多数 Microsoft Excel 下都可运行）来测试三角形问题、NextDate 函数和佣金问题。你可以在表格中设定一些测试用例，然后只要点击“ Run Test Case”（执行测试用例）按钮就可以运行这些测试用例。利用 naive.xls 程序来以一种直觉（朴素）的方法好好测试一下这三个实例，这是成为测试专家的起点。在这些实例中，每个程序都被有意设置了一些故障。每当你发现了失效，请尽量去推想其中潜在的故障到底是什么。请把你的结果都记录下来，在第 5 章、第 6 章和第 9 章中比较各种测试思想方法时还会用得到。

2.10 参考文献

Brown, J.R. and Lipov, M., Testing for software reliability, *Proceedings of the International Symposium on Reliable Software,* Los Angeles, April 1975, pp. 518–527.

Chellappa, M., Nontraversible paths in a program, *IEEE Transactions on Software Engineering,* Vol. SE-13, No. 6, June 1987, pp. 751–756.

Clarke, L.A. and Richardson, D.J., The application of error sensitive strategies to debugging, *ACM SIGSOFT Software Engineering Notes,* Vol. 8, No. 4, August 1983.

Clarke, L.A. and Richardson, D.J., A reply to Foster's comment on "The Application of Error Sensitive Strategies to Debugging," *ACM SIGSOFT Software Engineering Notes,* Vol. 9, No. 1, January 1984.

Gruenberger, F., Program testing, the historical perspective, in *Program Test Methods,* William C. Hetzel, Ed., Prentice-Hall, New York, 1973, pp. 11–14.

Hetzel, Bill, *The Complete Guide to Software Testing,* 2nd ed., QED Information Sciences, Inc., Wellesley, MA, 1988.

Inglis, Stuart J., *Planets, Stars, and Galaxies*, 4th Ed., John Wiley & Sons, New York, 1961.

Myers, G.J., *The Art of Software Testing,* Wiley Interscience, New York, 1979.

Pressman, R.S., *Software Engineering: A Practitioner's Approach*, McGraw-Hill, New York, 1982.