第 5 章　循环结构程序设计

循环结构和顺序结构、选择结构是结构化程序设计的 3 种基本控制结构。在编程中常常遇到一些操作并不复杂，但需要反复执行多次的问题，比如对全年级 1500 名学生都进行例 4-5 出现的百分制转换为五分制操作，这必须用到循环结构。

循环结构是指在程序中有规律地反复执行某一程序块的结构。被重复执行的程序块称为循环体，循环体的执行与否及次数多少视循环类型与条件而定。当然，无论何种类型的循环结构，其共同的特点是，必须确保循环体的重复执行能被终止。除了循环体以外，循环结构还包括循环控制部分，即用于规定循环的重复次数或重复条件。一般地，在循环结构前要有进行初始化的操作，主要用来对循环结构涉及的变量赋初值。

Visual Basic 常用的循环语句有 For…Next，While…Wend 和 Do…Loop 语句。For…Next 用于已知循环次数的情况，而 While…Wend 和 Do…Loop 一般用于不知道循环次数、在给定条件成立时执行循环体的情况。

5.1　For 循环语句

For 循环是计数型循环结构，用于控制循环次数预知的循环结构。

格式：

```
For <循环变量>=<初值> To <终值> [Step <步长>]
    <语句块>
    [Exit For]
    <语句块>
Next [<循环变量>]
```

说明：

(1) 循环变量：必须为数值型，取值范围在初值和终值之间。

(2) 步长：是数值型，一般为正，初值应小于等于终值；若为负，初值应大于等于终值；默认为 1。

(3) 语句块：可以是一条或多条语句，构成循环体。

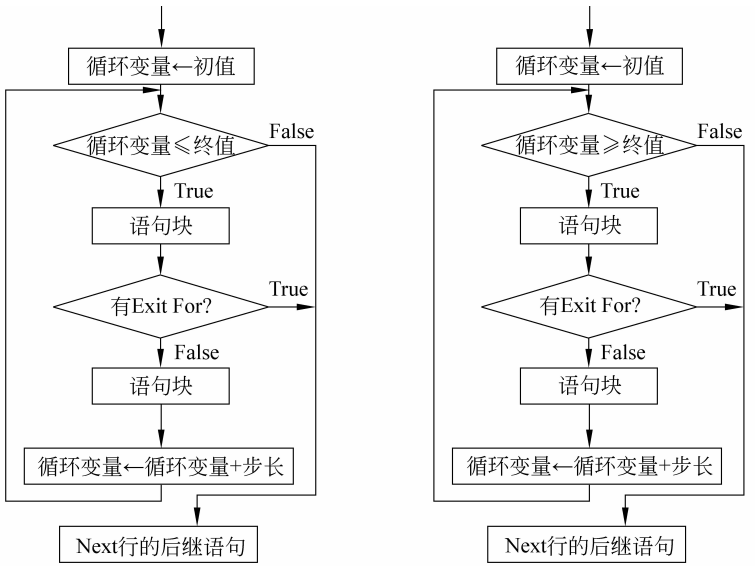
(4) Exit For：表示当遇到该语句时退出循环，执行 Next 行的后继语句。

(5) 循环次数 m 由初值、终值和步长决定，m＝Int((终值－初值)/步长＋1)。

For 循环的执行过程是，进入 For 循环，首先把初值赋给循环变量，然后将循环变量的值与终值进行比较，如果循环变量值超过终值就退出循环，否则执行循环体，然后将循环变量按步长增值，继续与终值比较，如果循环变量还是没有超过终值，则继续执行循环体，这样依次进行，直到循环变量值超过终值，循环结束。

这里“超过终值”有两种含义：当步长为正数，“超过终值”表示当前循环变量的值大于

终值；当步长为负数，“超过终值”表示当前循环变量的值小于终值。For 循环的执行流程如图 5-1所示。



(a) 步长为正数的For循环执行流程图　　(b) 步长为负数的For循环执行流程图

图 5-1　步长为正数、负数时 For 循环的执行流程图

【例 5-1】 求 1＋2＋3＋…＋100 的值。

分析：求上述 100 个数的和采用逐次累加的方法。这里用变量 sum 存放累加和。编程时，在循环结构前面，将其初值设为 0；用变量 i 存放加数，同时 i 还可以作为累加的次数，也称循环计数器，从 1 开始计到 100 为止。界面设计中用标签 Label1 提示计算的表达式，文本框 Text1 输出运算结果，单击命令按钮 Command1 进行计算，单击命令按钮 Command2 结束程序。

程序代码：

```
Private Sub Command1_Click()
    Dim i As Integer, sum As Integer
    sum=0
    For i=1 To 100 Step 1
        sum=sum+i
    Next i
    Text1.Text=sum
End Sub
Private Sub Command2_Click()
    End
End Sub
```

程序运行结果如图 5-2 所示。

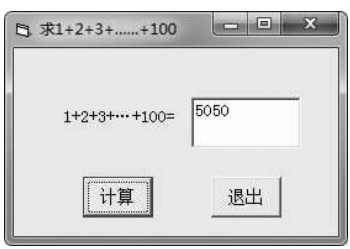


图 5-2　例 5-1 程序运行结果

注意：在求累加和的循环结构中，sum 的初值要设置为 0，且放在循环结构前。

【例 5-2】 求 $n!$，n 的值由用户输入。

分析：根据阶乘的定义可知：$n!=1\times2\times3\times\cdots\times(n-1)\times n$，所以应采用连乘的方法。下面代码中用变量 fact 存放连乘积，其初值设置为 1，放置于循环结构前；用变量 i 存放要连乘的乘数，同时 i 作为循环计数器，从 1 开始计到 n 为止。界面设计与上题类似。

程序代码：

```
Private Sub Command1_Click()
    Dim fact As Double, n As Integer, i As Integer
    n=Val(Text1.Text)
    fact=1
    For i=1 To n                    '步长为 1 时，可以省略
        fact=fact * i
    Next i
    Text2.Text=fact
End Sub
```

程序运行结果如图 5-3 所示。

图 5-3　例 5-2 程序运行结果

注意：在求连乘积的循环结构中，fact 的初值要设置为 1，也放在循环结构前。

【例 5-3】 有一袋球（总数在 100～200 个之间），若一次拿出来 4 个，则拿到最后还剩 2 个；若一次拿 5 个，则最后剩 3 个；若一次拿 6 个则正好拿完。编程计算该袋球原来有多少个？

分析：设计循环结构，对 100～200 之间的每一个数进行依次测试，满足条件的数就是所求的结果。设总共有 i 个球，且直接将 i 作为循环控制变量。满足条件“一次拿出来 4 个，则拿到最后还剩 2 个”的表达式为 i Mod 4＝2，满足其余两个条件的表达式分别为 i Mod 5＝3 和 i Mod 6＝0。它们三者需要同时成立，因而是逻辑与的关系。

程序代码：

```
Private Sub Form_Click()
    Dim i As Integer
    For i=100 To 200
        If i Mod 4=2 And i Mod 5=3 And i Mod 6=0 Then Print "袋中球的个数为", i
    Next i
End Sub
```

程序运行结果如图 5-4 所示。

图 5-4　例 5-3 程序运行结果

思考：如果得到第一个满足条件的数就结束，程序将如何修改？

【例 5-4】 如果一个三位数等于其各位数字的立方和，则称这个数为水仙花数，如 $153=1^3+5^3+3^3$，$370=3^3+7^3+0^3$。试编程找出所有的水仙花数。

分析：从 100，101，102，…，999 中，对每个数均要检测，看它是否符合水仙花数的条件。设每次要测试的数为 i，针对判断条件，我们首先求出测试数据 i 的个位、十位和百位上的数字。

程序代码：

```
Private Sub Form_Click()
    Dim a%, b%, c%, i%
    For i=100 To 999
        a=Int(i/100)                    '求百位数字
        b=Int(i/10)-a*10                '求十位数字
        c=i-a*100-b*10                  '求个位数字
        If i=a*a*a+b*b*b+c*c*c Then Print i,
    Next i
End Sub
```

程序运行结果如图 5-5 所示。

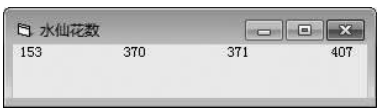


图 5-5　例 5-4 程序运行结果

例 5-3、例 5-4 均是对指定范围内的数逐一进行测试。在某个范围内列举出全部数据，对其中满足给定条件者进行相应处理，这里用到了程序设计中被称为“穷举”的算法。

穷举法的基本思想是根据提出的问题，列举所有可能的情况，并用问题中提出的条件检验哪些是需要的，哪些是不需要的。因此，穷举法常用于解决“是否存在”或“有多少种可能”等类型的问题。例如，求解不定方程的问题可以采用穷举法。

穷举法的特点是算法比较简单，但当列举的可能情况较多时，执行穷举算法的工作量将会很大。因此，在用穷举法设计算法时，应该重点注意使方案优化，尽量减少运算工作量。通常，只要对实际问题做详细的分析，将与问题有关的知识条理化、完备化、系统化，从中找出规律，或对所有可能的情况进行分类，引出一些有用的信息，列举量是可以减少的。

5.2　While 循环语句

虽然在 For…Next 循环结构中通过增加带条件的 Exit For 语句，可以在循环次数不确定、只给出循环结束条件时退出循环体，但是它与 While…Wend 循环结构相比显得复杂。While 循环又称为“当循环”，属于条件型循环。它根据某一条件进行判断，决定是否执行循环。

格式：

```
While <条件表达式>
    <语句块>
Wend
```

功能：如果＜条件表达式＞为 True(非 0 值)，则执行＜语句块＞(即循环体)；如果＜条件表达式＞为 False(值为 0)，则退出循环。其执行流程如图 5-6 所示。

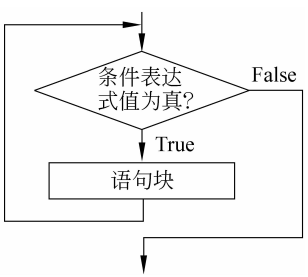


图 5-6　While 循环执行流程图

说明：

(1) ＜条件表达式＞的组成与 If 语句中＜条件表达式＞的组成要求相同。

(2) 实际进行程序设计时，循环体中语句的执行应能使条件发生改变，以使＜条件表达式＞的值最终可以出现 False。否则，会出现死循环。

(3) While 循环与 For 循环的区别是：For 循环对循环体执行指定次数；While 循环则是当条件表达式为 True 时重复执行循环体，循环次数预先无法确定。

例如，有如下一段程序：

```
While b>0
    c=c+a
    b=b-1
Wend
```

该程序通过重复做加法来计算 c＝c＋a，重复的条件是 b＞0。每次执行循环体以前，都要按 While 语句指定的条件表达式(b＞0)求一次值。如果它的结果为 True，则执行组成循环体的语句。直到条件表达式为 False(b＜＝0) 时才结束循环，控制转移到 Wend 的后继语句。

While 循环需要指定一个循环终止的条件，而 For 循环只能进行指定次数的重复。因此，当需要由某个条件是否出现来控制是否循环时，应当使用 While…Wend 一类的循环。

【例 5-5】 用 While 循环完成例 5-1 的要求。

分析：应用 While 循环求累加和时，累加到 100 是结束循环的条件。若设每次累加的加数为 i，循环的条件即为 i＜＝ 100。每次累加后要为下次累加做准备，因此求和后要改变 i 的值。运行结果与例 5-1 结果相同。

程序代码：

```
Private Sub Command1_Click()
    Dim i%, sum%
    sum=0: i=1
    While i<=100
        sum=sum+i
        i=i+1
    Wend
    Text1.Text=sum
End Sub
Private Sub Command2_Click()
    End
End Sub
```

对比两种不同的循环结构可以看出，使用 For 循环时，通过预知的循环次数来控制循环体的执行；使用 While 循环时，依靠设定条件表达式来控制循环体执行与否。

【例 5-6】 设 sum＝$1^2+2^2+3^2+\cdots+n^2$，求 sum 小于 1000 时 n 的最大值。

分析：该问题的累加次数不能确定，但是可设定是否执行累加操作的条件，即 sum＜1000，因此采用 While 循环。

程序代码：

```
Private Sub Form_Click()
    Dim n%, sum#
```

```
    sum=0: n=0
    While sum<1000
        n=n+1
        sum=sum+n^2
    Wend
    Text1.Text=n-1
End Sub
```

当累加操作终止后，sum 的值必然大于等于 1000，故此时 n－1 才是本题所求的答案。

程序运行结果如图 5-7 所示。

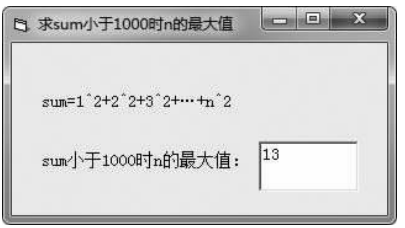


图 5-7　例 5-6 程序运行结果

思考：如果采用先累加再改变加数 n 的值，程序如何修改才能得到正确答案？

【例 5-7】 我国现有人口 13 亿，按照年增长率 1.2%计算，多少年后我国人口达到 20 亿？

分析：我国现有人口 13 亿，设 p 是 n 年后我国人口达到的数量(20 亿)，r 是年增长率(1.2%)。求解此问题，可根据公式 $p=13\times(1+r)^n$，根据终止条件 $p<20$，利用 While 循环求解 n。

程序代码：

```
Private Sub Form_Click()
    Dim p As Single, r As Single, i As Integer
    p=13
    r=0.012
    i=0
    While p<20
        p=p * (1+r)
        i=i+1
    Wend
    Print i; "年后，我国人口将达到"; p; "亿"
End Sub
```

图 5-8　例 5-7 程序运行结果

程序运行结果如图 5-8 所示。

5.3　Do…Loop 循环语句

Do…Loop 循环也属于条件型循环，即根据条件来决定是否执行循环。相比 While 循环，Do…Loop 循环的应用具有更强的灵活性，主要体现在：它既能指定循环开始条件，又能指定循环结束条件；既能构成前测型 Do…Loop 循环语句，又能构成后测型 Do…Loop 循环语句。

5.3.1　前测型 Do…Loop 循环语句

前测型 Do…Loop 循环的特点是先判断循环条件，根据条件决定是否执行循环体，执行

循环体的最少次数为 0。

格式：

```
Do [While | Until <条件表达式>]
    <语句块>
    [Exit Do]
    <语句块>
Loop
```

说明：

(1) Do While…Loop 是当型循环结构。当＜条件表达式＞的值为 True 时执行循环体；当 ＜条件表达式＞的值为 False 时退出循环。

(2) Do Until…Loop 是直到型循环结构。当＜条件表达式＞的值为 False 时执行循环体；直到＜条件表达式＞的值为 True 时退出循环。

(3) ＜条件表达式＞的组成与 If 语句中＜条件表达式＞的组成要求相同。

(4) Exit Do 语句表示当遇到该语句时，退出循环，执行 Loop 行的后继语句。

(5) 当省略[While | Until＜条件表达式＞]子句时，即循环结构仅由 Do…Loop 关键字构成，表示无条件循环，这时在循环体内应该有 Exit Do 语句，否则为死循环。

前测型 Do…Loop 循环的执行流程如图 5-9 所示。

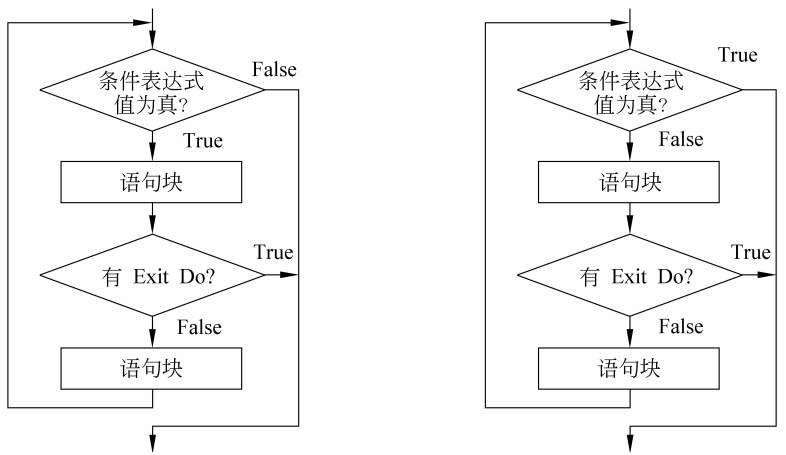


图 5-9　Do While…Loop 和 Do Until…Loop 循环执行流程图

【例 5-8】 用 Do 循环结构实现例 5-2 求 $n!$ 的程序。

分析：一般 For 循环结构可以改用 While 循环和 Do 循环实现，反之则不然。例 5-2中的程序代码若用 Do While…Loop 循环语句改写，则循环控制条件是 i＜＝n。

程序代码：

```
Private Sub Command1_Click()
    Dim fact As Double, n As Integer, i As Integer
    n=Val(Text1.Text)
```

```
    fact=1
    i=1
    Do While i<=n
        fact=fact * i
        i=i+1
    Loop
    Text2.Text=fact
End Sub
```

思考：如果用 Do Until…Loop 循环语句实现，应怎样编写代码？

【例 5-9】 求满足 1＋2＋3＋…＋n≥1000 的 n 的最小值。

分析：本例也是一个累加问题，但累加次数不知道，仅仅知道累加结束的条件：累加和大于或等于 1000 就中止了，因此用直到型循环结构 Do Until…Loop 实现。

程序代码：

```
Private Sub Form_Click()
    Dim n%, s%
    n=0
    sum=0
    Do Until sum>1000 Or sum=1000
        n=n+1
        sum=sum+n
    Loop
    Print "sum="; sum, "n="; n
End Sub
```

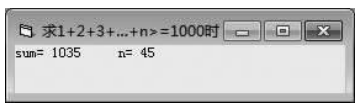


图 5-10　例 5-9 运行结果

程序运行结果如图 5-10 所示。

思考：本例如果用 Do While …Loop 循环语句实现，对以上的代码应如何改动？

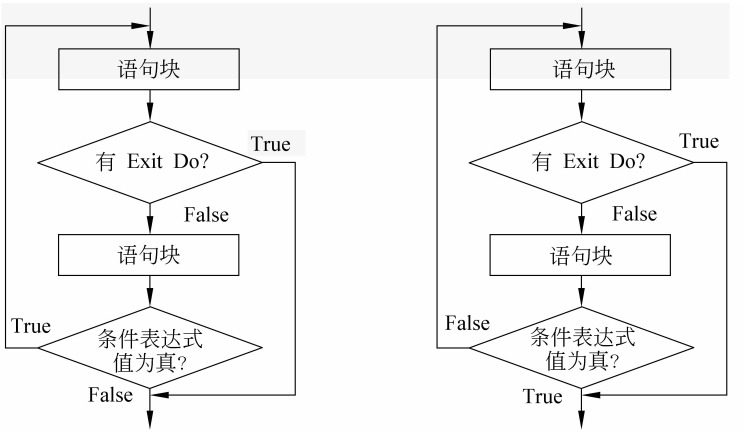
5.3.2　后测型 Do…Loop 循环语句

格式：

```
Do
    <语句块>
    [Exit Do]
    <语句块>
Loop [While | Until<条件表达式>]
```

说明：后测型 Do…Loop 循环语句的说明同前测型 Do…Loop 循环语句的说明(1)～(5)。二者的主要区别在于：后测型要先执行一次循环体，再判断条件；而前测型要先判断条件，然后根据判断结果决定是否执行循环体。因此，对于后测型循环语句，不管条件是否满足，循环体至少有一次执行机会。

后测型 Do…Loop 循环的执行流程如图 5-11 所示。



(a) Do …Loop While循环执行流程图　　(b) Do …Loop Until循环执行流程图

图 5-11　Do…Loop While 与 Do…Loop Until 循环执行流程图

【例 5-10】 将 400～600 之间能够被 3 整除的数输出。

分析：对 400～600 之间的每个数进行测试，循环次数是已知的，完全可采用 For 循环。若采用后测型 Do…Loop Until 循环，则需要先对被测数 i 赋初值，并在循环体内修改它：i=i+1。循环结束的条件是 i>600。

程序代码：

```
Private Sub Form_Click()
    Dim i%, count%                          'count 作为计数器,统计输出个数
    i=400
    count=0
    Do
        If i Mod 3=0 Then
            Print i;
            count=count+1
            If count Mod 8=0 Then Print   '控制一行输出 8 个数后,换行
        End If
        i=i+1
    Loop Until i>600
End Sub
```

程序运行结果如图 5-12 所示。

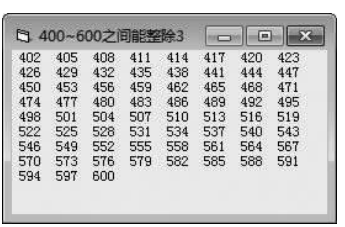


图 5-12　例 5-10 程序运行结果

思考：对于循环次数已知的情况，分别采用 For 循环和 Do…Loop 循环，哪个更方便？

【例 5-11】 已知 π 的近似值可用如下公式表示：

$$\frac{\pi}{4}\approx 1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots+(-1)^{n-1}\frac{1}{2n-1}$$

利用该公式求 π，要求最后一项的绝对值小于 0.000001。

分析：该题仍然需要累加求解，累加的各项不考虑符号位可表示为 1/(2 * i－1)。由于累加次数事前未知，但累加计算结束的条件已知：最后一项的绝对值小于 0.000001。因此

选用 Do…Loop While 循环结构完成。

程序代码：

```
Private Sub Form_Click()
    Dim i&, r# , sum#          'i 作为计数器,r 作为当前项,sum 作为累加和
    i=1
    r=1
    sum=0
    Do
        r=1/(2 * i-1)
        r=r * (-1)^(i+1)
        sum=sum+r
        i=i+1
    Loop While Abs(r)>=0.000001
    sum=sum * 4
    Print "圆周率的近似值为"; sum
End Sub
```

图 5-13　圆周率的近似值

程序运行结果如图 5-13 所示。

思考：如果本例用 Do…Loop Until 循环语句实现，对以上的代码应如何改动？

5.4　循环嵌套

前述的循环程序循环体内都不包括其他循环结构，即都属于单层循环。对于复杂的问题，循环体内常常又涉及其他循环操作。一个循环结构中又包含一个或多个循环结构被称为循环嵌套，或称多重循环。

多重循环对嵌套的层数没有限制。有几层嵌套，就称为几重循环，如二重循环、三重循环、四重循环等。一般地，把嵌套在一个循环体内部的另一个循环结构称为内循环；这样，嵌套了其他内循环部分的循环结构就称为外循环。

为了使多重循环结构具有较好的可读性，通常用缩进方式书写相应的源代码。

使用多重循环时要注意以下几点：

(1) 多重循环的执行过程是，外循环每执行一次，内循环要从头到尾执行一遍；

(2) 外循环必须完全包含内循环，不能交叉，如表 5-1 和表 5-2 所示；

(3) 内循环变量与外循环变量不能同名；

(4) 在多重循环的任何一层中，都可以使用 Exit Do 或 Exit For 退出循环。但要注意，只能退出 Exit Do 或 Exit For 语句所对应的最内层循环，而不是一次退出多层循环。

例如，有如下的程序(为叙述方便起见，将代码行标注了序号)：

```
1  f=1
2  For i=1 to 10
3    For j=1 to 10
4        f=f * i * j
```