第8讲 程序设计

8－2 循环结构

过程式循环 Do, While, For

1. Do 语句

Do[循环体的表达式, 循环范围]

一重循环

Do[expr, {i, m, n, d}]

在循环范围内按步长d对expr求值。

i 循环变量，m 初始值，n中止值，d 步长. m, m + d, m + 2 d, ...

循环变量可为整数/小数分数复数枚举. expr 复合表达式.

Do[expr, {i, m, n}] 步长 为1

Do[expr, {i, n}] 初值和步长都为1

Do[expr, {n}] 计算n 次表达式expr

Do[expr, {i, list}] 按循环列表list的元素, 对expr求值

例1：输出：n 从E 到10 按步长π递增.

```
Do[Print[n], {n, E, 10, Pi}]
```

例2：循环变量 n 是个隐含的局部变量，它的变化不会影响语句外部的变量n的值.

```
n = 100; Do[Print[n], {n, E, 10, Pi}]; n
```

例3：循环变量是枚举类型.

```
Do[Print[t], {t, {Red, Green, Blue, Yellow}}]
```

例4：留意变量是字符的循环.

```
t = "t"; Do[t = 1 / (3 k + t); Print[t], {k, 3}]
```

$$\cfrac{1}{3+t}$$

$$\cfrac{1}{6+\cfrac{1}{3+t}}$$

$$\cfrac{1}{9+\cfrac{1}{6+\cfrac{1}{3+t}}}$$

例5： 循环范围 {k, 1, 3} 与 {k, {1, 3}} 有区别 ？

```
t = "t"; Do[t = 1 / (3 k + t); Print[t], {k, {1, 3}}]
```

$$\cfrac{1}{3+t}$$

$$\cfrac{1}{9+\cfrac{1}{3+t}}$$

二重循环

```
Do[expr, {i, i0, i1, is}, {j, j0, j1, js}]
```

i 从i0 到i1 按步长is递增；对每个i，
j 从j0 到j1 按步长js递增，计算表达式expr，
在前的是外循环，在后的是内循环.

例6 ：输出：i 从1 到3 ，j从1到3的 {i, j}.

```
Do[Print[{i, j}], {i, 3}, {j, 3}]
```

```
Do[ Do[Print[{i, j}], {j, 3}], {i, 3}]
```

例7 ：输出：i 从1 到3 ，对每个i，j从1到i的 {i, j}.

```
Do[Print[{i, j}], {i, 3}, {j, i}]
```

2. While 语句

```
While[cond, expr]
```

若逻辑条件 cond = True，则对循环体表达式求值，
重复对条件判断和对循环体求值过程直到cond ≠ True时退出循环.

例8：计算 1 + 2 + 3 + 4 + 5

```
s = 0; k = 1; While[k ≤ 5, s = s + k; Print[s]; k++]
```

```
s = 0; Do[s = s + k; Print[s], {k, 1, 5}]
```

例9：用辗转相除法计算两个数的最大公约数.

```
{a, b} = {49, 21 }
While[b ≠ 0, {a, b} = {b, Mod[a, b]}; Print[{a, b}]]; a
```

例10：用 Newton 迭代法计算3的平方根.

$$x_{k+1} = x_k - \frac{x_k^2 - 3}{2x_k}$$

```
xb = 1.5; xa = xb + 1;
While[Abs[xb - xa] > 0.00001,
    xa = xb;
    xb = xa - (xa^2 - 3) / 2 / xa;
    Print[xb]]
```

3 For 循环

For[初始值, 条件, 修正循环变量, 循环体]

For[init, cond, incr, expr]

init 最先求值，接下来按照cond、expr、incr的顺序依次对表达式求值，直到cond ≠ True.

init, cond, incr, expr均为复合表达式，逗号是4个部分的分隔符.

例11 ：计算 1 + 2 + 3 + 4 + 5

For[init, cond, incr, expr]　For[init, cond, incr]

```
For[s = 0; n = 1, n ≤ 5, n++, s += n; Print[s]] (*正确的程序*)
```

```
For[s = 0; n = 1, n ≤ 5, ++n; s += n; Print[s]](*错误的程序*)
```

例12 ：与 Do 循环 不同，在 While、For 循环 中没有隐含的局部变量.

```
n = 100; For[n = E, n ≤ 10, n = n + Pi, Print[n]]; Print["n=", n]  (* n+=Pi *)
```

```
e
```

```
e + π
```

```
e + 2 π
```

```
n=e + 3 π
```

例 13：生成多项式序列 f (n) = $1 + \left(1 + \left(1 + x^2\right)^2 + \ldots\right)^2$

```
For[i = 1; f = x, i < 4, i++, f = 1 + f^2; Print["f(", i, ")=", f]]
```

```
f(1)=1 + x^2
```

```
f(2)=1 + (1 + x^2)^2
```

```
f(3)=1 + (1 + (1 + x^2)^2)^2
```

4. ++ 和 --

函数	说明
x++、x--	在使用x后将x增 (减) 1
++x、--x	在使用x前将x增 (减) 1
x += y、x -= y	x = x + y、x = x - y
x *= y、x /= y	x = x * y、x = x / y

5. 迭代函数（函数式循环）

Nest[f, x, n] 迭代n次，f (f (f (... f (x) ...)))

NestList[f, x, n] 迭代n次并给出每步迭代值

NestWhile[*f*, *expr*, *test***]** 从*expr*开始，重复应用*f*直到*test*不再是True为止.

NestWhileList[f, expr, test] 同上，列出迭代列表

FixedPoint[*f*, *expr***]** 从*expr*开始，重复应用*f*直到结果不再改变

FixedPointList[f, x0] 同上，列出迭代列表

TakeWhile[{a1, ..., an}, f] 列出 f (a1) =. . . = f (ak) = True 的ai

LengthWhile[{a1, ..., an}, f] 给出满足 (a1) =. . . = f (ak) = True的元素个数

例14：以1 .0 为初值，用Newton迭代法计算3的平方根，做5次迭代并输出计算结果.

$$x_{k+1} = x_k - \frac{x_k^2 - 3}{2x_k}$$

```
f[x_] := (x + 3 / x) / 2; NestList[f, 1.0, 5]
FixedPointList[f, 1., 5]
```

例15：取出列表中的偶数

```
TakeWhile[{2, 4, 6, 1, 2, 3}, EvenQ]
```

例16：判断列表中偶数的个数

```
LengthWhile[{2, 4, 6, 1, 2, 3}, EvenQ]
```