循环结构

Ref As Pseudocode

FOR 循环与 STEP

迭代（iteration）就是重复执行一组语句。负责重复的结构叫作**循环**（loop）。伪代码有三种循环，每一种适合不同的任务。

■ 计数控制循环

如果在循环开始之前就知道**要重复多少次**，就用 `FOR`。它叫作**计数控制循环**（count-controlled loop）。

```
FOR <identifier> ← <value1> TO <value2>
   <statement(s)>
NEXT <identifier>
```

```
DECLARE Counter : INTEGER
FOR Counter ← 1 TO 5
   OUTPUT Counter, " x 3 = ", Counter * 3
NEXT Counter
```

- **循环计数器**（loop counter，这里是 `Counter`）必须是 `INTEGER`。
- 它依次取 `value1` 到 `value2` 之间的**每一个**值，**两端都包括**。所以 `1 TO 5` 执行五次。
- 如果 `value1 = value2`，循环体执行一次；如果 `value1 > value2`，循环体一次也不执行。
- 在 `NEXT` 后面写上计数器的名字是好习惯，循环套循环时尤其如此。
- 不要在循环里自己修改计数器，`FOR` 循环会负责这件事。

■ STEP

加上 `STEP` 可以按 1 以外的步长计数。步长可以是**负数**，用来倒数。

```
FOR <identifier> ← <value1> TO <value2> STEP <increment>
```

```
DECLARE i : INTEGER
FOR i ← 0 TO 20 STEP 5
   OUTPUT i
NEXT i
FOR i ← 3 TO 1 STEP -1
   OUTPUT "Countdown ", i
NEXT i
```

一旦下一个值会**超过** `value2`，循环就停止。所以 `FOR i ← 1 TO 10 STEP 4` 只给出

1、5 和 9。

■ 循环中的循环

一个循环可以放在另一个循环里面。外层循环每执行一遍，内层循环就**完整地**执行一次。

```
DECLARE Row : INTEGER
DECLARE Column : INTEGER
DECLARE Line : STRING
FOR Row ← 1 TO 3
   Line ← ""
   FOR Column ← 1 TO 4
      Line ← Line & NUM_TO_STR(Row * Column) & " "
   NEXT Column
   OUTPUT Line
NEXT Row
```

（NUM_TO_STR 把数字变成字符串，这样才能用 & 连接；见第 7 章。）外层循环执行 3 次，每次内层循环执行 4 次，一共 12 遍。

常见错误

- FOR Counter = 1 TO 5。FOR 这一行要用箭头 ←。
- 倒数时没写 STEP -1。FOR i ← 10 TO 1 一次也不会执行。
- 内外层循环结束的顺序写反。**最后**打开的循环必须**最先**结束。

WHILE 前置条件循环

如果**不知道**要重复多少次，而且循环可能需要执行**零**次，就用 WHILE。

```
WHILE <condition>
   <statement(s)>
ENDWHILE
```

```
DECLARE Number : INTEGER
Number ← 40
WHILE Number > 9
   Number ← Number - 9
ENDWHILE
OUTPUT Number
```

- 每一遍开始**之前**先检查条件。所以它叫作**前置条件循环**（pre-condition loop）。
- 条件为 TRUE **时**就继续重复，条件变为 FALSE 时停止。
- 如果第一次检查时条件就是 FALSE，循环体**一次也不执行**。

■ 循环里必须有东西在变化

循环体必须改变条件所检查的内容。否则条件永远是 TRUE，就会形成**无限循环**(infinite loop)。

```
DECLARE Balance : REAL
DECLARE Years : INTEGER
Balance ← 1000.0
Years ← 0
WHILE Balance < 2000.0
   Balance ← Balance * 1.1
   Years ← Years + 1
ENDWHILE
OUTPUT "Doubled after ", Years, " years"
```

这里 Balance 每一遍都在增长，所以循环一定会结束。

常见错误

- 忘记写 ENDWHILE。
- 循环之前没有给变量赋值。条件是先检查的，所以变量必须已经有值。
- 无限循环：循环体里没有任何东西改变条件。

REPEAT 后置条件循环

如果循环体必须**至少执行一次**，就用 REPEAT，例如先让用户输入，再检查输入是否正确。

```
REPEAT
   <statement(s)>
UNTIL <condition>
```

```
DECLARE Mark : INTEGER
REPEAT
   OUTPUT "Enter a mark from 0 to 100"
   INPUT Mark
UNTIL Mark >= 0 AND Mark <= 100
OUTPUT "Mark saved: ", Mark
```

- 每一遍执行**之后**才检查条件：这是**后置条件循环**（post-condition loop）。
- 循环一直重复，**直到**条件为 TRUE。条件一变成 TRUE，循环就停止。
- 没有 ENDREPEAT：UNTIL 这一行就是循环的结束。

注意两种条件的意思正好相反：WHILE 在条件为 TRUE 时继续；REPEAT 在条件为 TRUE 时停止。

```
DECLARE Total : INTEGER
DECLARE Count : INTEGER
Total ← 0
Count ← 0
REPEAT
   Count ← Count + 1
   Total ← Total + Count
UNTIL Total > 20
OUTPUT "Added 1 to ", Count, " to pass 20: total ", Total
```

常见错误

- 条件写反。”年龄不对就一直问”要写成 UNTIL 年龄**正确**。
- 循环体可能需要执行零次时却用了 REPEAT。这时应该用 WHILE。

如何选择循环

循环	何时使用	何时检查	至少执行
FOR	开始之前就知道要执行几遍	（按计数）	0 次
WHILE	不知道次数，而且可能需要执行**零**次	每遍**之前**	0 次
REPEAT	不知道次数，而且至少需要执行**一**次	每遍**之后**	1 次

考试中的典型情况：

- 遍历数组的每个元素：FOR。
- 读文件直到结束：WHILE NOT EOF(...)，因为文件可能是空的。
- 反复要求输入，直到输入有效：REPEAT ... UNTIL。
- 在数组中查找，找到后提前停止：带标志的 WHILE（或 REPEAT）。

■ 练习

1. 用一个 FOR 循环输出 2 到 20 之间的所有偶数。

2. 从 1 开始，不断把一个数翻倍。用 WHILE 循环输出要翻倍多少次才能超过 1000。

3. 把下面的 WHILE 循环改写成输出相同的 REPEAT 循环：

```
Count ← 1
WHILE Count <= 3
   OUTPUT Count
   Count ← Count + 1
ENDWHILE
```

<details> <summary> 参考答案 </summary>

```
DECLARE Number : INTEGER
FOR Number ← 2 TO 20 STEP 2
   OUTPUT Number
NEXT Number
```

```
DECLARE Value : INTEGER
DECLARE Doublings : INTEGER
Value ← 1
Doublings ← 0
WHILE Value <= 1000
   Value ← Value * 2
   Doublings ← Doublings + 1
ENDWHILE
OUTPUT Doublings
```

输出 10，因为 2 的 10 次方是 1024。

```
DECLARE Count : INTEGER
Count ← 1
REPEAT
   OUTPUT Count
   Count ← Count + 1
UNTIL Count > 3
```

`WHILE` 的条件 `Count <= 3` 变成了它的反面 `Count > 3`。两个循环之所以输出相同，只是因为这里的 `WHILE` 循环体至少会执行一次。

</details>