循环结构

Ref Pseudocode

FOR 计数循环

迭代 (iteration) 重复执行语句。当你在**开始之前**就知道要重复多少次时, 使用 `FOR` 循环 —— 计数控制 (count-controlled) 循环。

```
FOR <identifier> ← <value1> TO <value2>
    <statements>
NEXT <identifier>
```

```
DECLARE Index : INTEGER
FOR Index ← 1 TO 5
    OUTPUT "Line ", Index
NEXT Index
```

计数器 (counter) 从第一个值开始, 循环对直到第二个值 (**包含**它) 的每个值各运行一次。`1 TO 5` 运行五次。

STEP

`STEP` 改变每次跳跃的大小。负的步长会倒着数。

```
DECLARE Index : INTEGER
FOR Index ← 2 TO 10 STEP 2
    OUTPUT Index
NEXT Index
FOR Index ← 3 TO 1 STEP -1
    OUTPUT "Countdown ", Index
NEXT Index
```


累加总和

最常见的用法: 循环之外有一个变量, 循环不断往里加。

```
DECLARE Index, Total : INTEGER
Total ← 0
FOR Index ← 1 TO 10
    Total ← Total + Index
NEXT Index
OUTPUT "Sum of 1 to 10 is ", Total
```

`Total ← 0` 必须写在循环**之前**。写在里面, 它每一轮都会被清零。

常见错误

- `NEXT` 写错了计数器。嵌套循环必须按打开的相反顺序闭合: 先写内层的 `NEXT`。
- 以为 `1 TO 5` 运行四次。两端都包含。
- 在循环内部改动计数器。让 `FOR` 自己掌管它。

WHILE 前条件循环

`WHILE` 循环在每一轮**之前**检验, 因此它可能运行**零**次。当重复次数取决于循环运行中发生的事情时, 用它。

```
WHILE <condition> DO
    <statements>
ENDWHILE
```

```
DECLARE Total : INTEGER
Total ← 1
WHILE Total < 100 DO
    Total ← Total * 2
ENDWHILE
OUTPUT "First power of two past 100: ", Total
```

循环内部必须有东西最终使条件变为假, 否则循环永不结束 —— 这就是**无限循环** (infinite loop)。

```
DECLARE Count : INTEGER
Count ← 5
WHILE Count > 0 DO
    OUTPUT Count
    Count ← Count - 1
ENDWHILE
OUTPUT "Lift off"
```

常见错误

- 忘记改动条件所检验的变量, 造成无限循环。
- 漏掉 `DO` 或 `ENDWHILE`。
- 在次数已知时用 `WHILE`。`FOR` 一行就把意思说清楚了。

REPEAT 后条件循环

`REPEAT` 循环在每一轮**之后**检验, 因此它总是**至少运行一次**。

```
REPEAT
    <statements>
UNTIL <condition>
```

“ 至少一次 ” 正是选择它的理由: 验证 (validating) 输入时, 你必须先问, 才能判断答案。

```
DECLARE Number : INTEGER
Number ← 0
REPEAT
    Number ← Number + 25
    OUTPUT "Trying ", Number
UNTIL Number >= 100
OUTPUT "Reached ", Number
```

△ `UNTIL` 陈述的是**停止**循环的条件;`WHILE` 陈述的是**继续**循环的条件。两者相反, 弄反了是经典错误。

	何时检验	至少运行一次?	条件的含义
`FOR`	看计数器	否 (空区间运行零次)	—
`WHILE`	之前	否	为真时继续
`REPEAT`	之后	**是**	变为真时停止

常见错误

- 想写 `UNTIL Number >= 100` 却写成 `UNTIL Number < 100`。
- 在循环体必须能运行零次的场合用 `REPEAT`。
- 忘记 `UNTIL` 就是结构的结尾 —— 没有 `ENDREPEAT` 这种写法。