循环结构

Ref Pseudocode

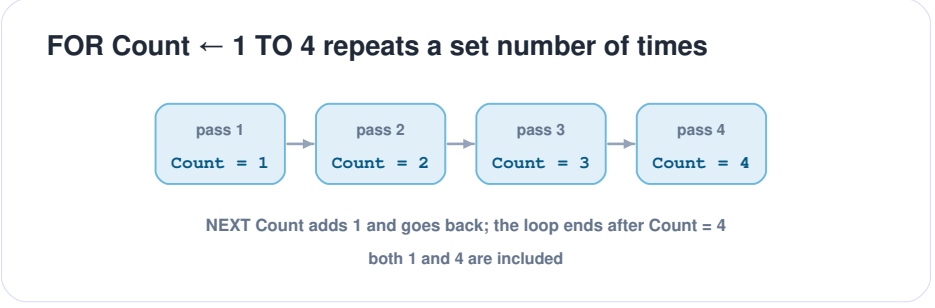
FOR, the counted loop

迭代重复执行语句。当你**在开始之前就知道**需要循环多少次时，应使用 FOR 循环即计数控制型计数控制循环。

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

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

计数器从起始值开始，对于直到并**包含**结束值的每一个数值都会运行一次。`1 TO 5` 运行五次。



FOR Count ← 1 TO 4 运行四次循环，且同时包含 *1* 和 *4*

■ STEP（步长）

STEP 改变每次跳跃的大小。负步长用于递减计数。

```
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
```

■ 累加求和

最常见的用法：定义一个位于循环外部的变量，由循环对其不断累加。

```
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` 必须放在循环**之前**。如果放在内部，每次迭代都会重置。

常见错误

- `NEXT` 命名了错误的计数器。嵌套循环的关闭顺序必须与打开顺序相反：先关闭内层的 `NEXT`。
- 假设 `1 TO 5` 运行四次。包含首尾两端。
- 在循环内部修改计数器。让 `FOR` 负责它。

WHILE, the pre-condition loop

`WHILE` 循环在每次迭代**之前**测试条件，因此可能一次都不执行。适用于重复次数取决于循环运行过程中发生的事件的情况。

```
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
```

循环内部的某项操作最终必须使条件变为假，否则循环将永远无法结束 — 即**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

- 忘记修改变量导致条件测试失败，从而产生无限循环。
- 遗漏了 `DO` 或 `ENDWHILE`。
- 在计数已知的情况下使用 `WHILE`。用 `FOR` 可以用一行代码清晰表达你的意图。

REPEAT, the post-condition loop

`REPEAT` 循环在每次迭代**之后**测试条件，因此至少会执行**一次**。

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

这正是选择它的理由：用于验证输入（validating input），此时你必须先询问才能判断答案是否有效。

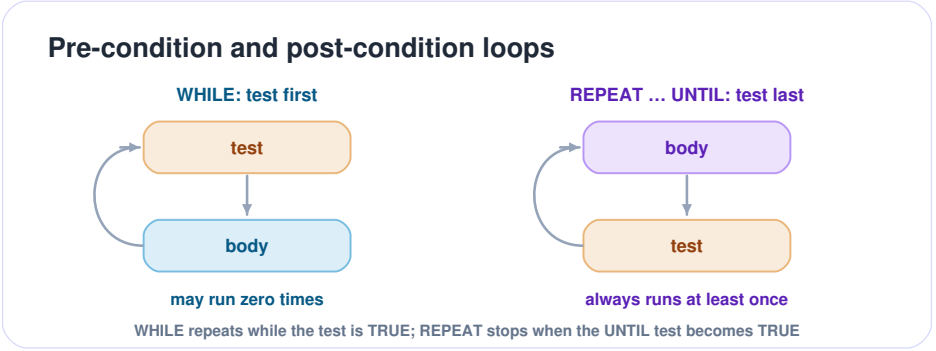
```
DECLARE Number : INTEGER
Number ← 0
REPEAT
    Number ← Number + 25
    OUTPUT "Trying ", Number
UNTIL Number >= 100
OUTPUT "Reached ", Number
```

△ `UNTIL` 表示**终止**循环的条件；`WHILE` 表示继续循环的条件 **使其持续运行**。两者互为反义，交换它们是常见的典型错误。

	测试时机	**至少执行一次?**	**条件含义**
`FOR`	计数器	否（空范围执行零次）	–
`WHILE`	之前	否	当此条件为真时继续执行
`REPEAT`	after	**是**	当此条件为真时停止

常见错误

- 将 `UNTIL Number < 100` 误写为 `UNTIL Number >= 100`。
- 在必须允许执行零次的循环体中使用 `REPEAT`。
- 忘记 `UNTIL` 会终止该结构 — 不存在 `ENDREPEAT`。



WHILE 先测试且可能执行零次；*REPEAT* 后测试且至少执行一次