数组

Ref As Pseudocode

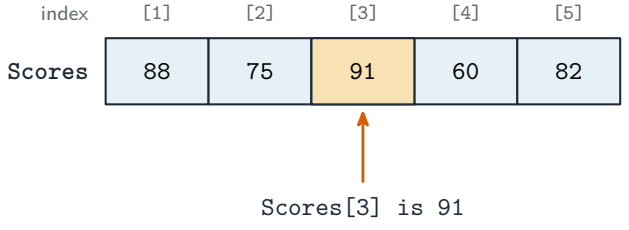
一维数组

数组(array)用一个名字存放许多个**同一数据类型**的值。每个值是一个**元素**(element)，通过它的 **索引**（index），也就是它的位置编号，来访问。数组的大小是固定的，不会改变。

```
DECLARE <identifier> : ARRAY[<lower>:<upper>] OF <data type>
```

```
DECLARE Scores : ARRAY[1:5] OF INTEGER
Scores[1] ← 88
Scores[2] ← 75
Scores[3] ← 91
Scores[4] ← 60
Scores[5] ← 82
OUTPUT "The third score is ", Scores[3]
```

- `[1:5]` 给出**下界**（lower bound）1 和**上界**（upper bound）5，所以这个数组有 5 个元素。
- 剑桥通常从 **1** 开始。一定要写出下界，因为有些语言从 0 开始。
- 索引写在**方括号**里：`Scores[3]`。
- 索引可以是任何结果为整数的表达式：`Scores[i]`、`Scores[n + 1]`。



Scores[1:5]：五个格子，一个名字，每个格子通过索引访问

■ 用循环访问每个元素

用 FOR 循环的计数器作索引，就能依次访问每个元素。这是 Paper 2 中最常见的写法。

```
DECLARE Names : ARRAY[1:4] OF STRING
DECLARE i : INTEGER
Names[1] ← "Ali"
Names[2] ← "Bo"
Names[3] ← "Chen"
Names[4] ← "Dara"
FOR i ← 1 TO 4
   OUTPUT i, ": ", Names[i]
NEXT i
```

你**不能**用一条语句给一组元素赋值。`Names[1 TO 4] ← ""` 是不允许的，要用循环：

```
DECLARE Names : ARRAY[1:30] OF STRING
DECLARE Index : INTEGER
FOR Index ← 1 TO 30
   Names[Index] ← ""
NEXT Index
OUTPUT "All 30 names cleared"
```

但你**可以**用一行把整个数组复制给另一个大小和类型都相同的数组：`SavedGame ← Board`。

常见错误

- 索引超出范围，例如对 `ARRAY[1:5]` 使用 `Scores[6]` 或 `Scores[0]`。这是运行时错误。
- 用圆括号：`Scores(3)`。数组要用方括号。
- 忘记写数据类型：每个数组都需要 `OF <data type>`。

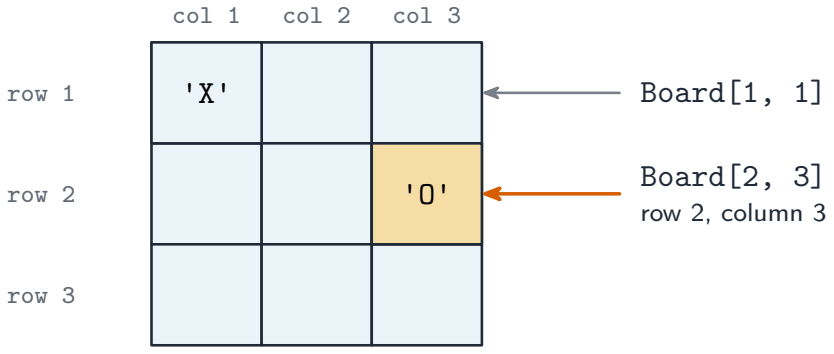
二维数组

二维（two-dimensional）数组就像一张表格，有**行**和**列**。访问一个元素需要两个索引。

```
DECLARE <identifier> : ARRAY[<lower1>:<upper1>, <lower2>:<upper2>] OF
↪  <data type>
```

```
DECLARE Board : ARRAY[1:3, 1:3] OF CHAR
Board[1, 1] ← 'X'
Board[2, 3] ← 'O'
OUTPUT "Row 2, column 3 holds ", Board[2, 3]
```

- 按惯例，第一个索引是**行**，第二个索引是**列**。
- 两个索引写在**同一对**方括号里，用逗号隔开：`Board[2, 3]`，而不是 `Board[2][3]`。



Board[1:3, 1:3]：第一个索引选行，第二个索引选列

■ 用嵌套循环处理二维数组

要访问每个元素，用一个循环处理行，在它里面再用一个循环处理列。

```
DECLARE Amount : ARRAY[1:2, 1:3] OF INTEGER
DECLARE Row : INTEGER
DECLARE Column : INTEGER
DECLARE RowTotal : INTEGER
Amount[1, 1] ← 4
Amount[1, 2] ← 7
Amount[1, 3] ← 1
Amount[2, 1] ← 9
Amount[2, 2] ← 2
Amount[2, 3] ← 5
FOR Row ← 1 TO 2
   RowTotal ← 0
   FOR Column ← 1 TO 3
      RowTotal ← RowTotal + Amount[Row, Column]
   NEXT Column
   OUTPUT "Total for row ", Row, " is ", RowTotal
NEXT Row
```

`RowTotal ← 0` 写在外层循环**里面**，所以每一行都从零开始重新累加。

常见错误

- 把两个索引写反：数组按”先行后列”建立，却写成 `Board[Column, Row]`。
- 把 `RowTotal ← 0` 写在外层循环**之前**。这样第二行的总和里也包含了第一行。

求和、计数、最大值与最小值

大多数数组题目考的都是下面几种固定写法。把它们当作“ 模板 ”来记。

■ 求和与平均值

```
DECLARE Marks : ARRAY[1:5] OF INTEGER
DECLARE Total : INTEGER
DECLARE i : INTEGER
Marks[1] ← 56
Marks[2] ← 71
Marks[3] ← 48
Marks[4] ← 90
Marks[5] ← 65
Total ← 0
FOR i ← 1 TO 5
   Total ← Total + Marks[i]
NEXT i
OUTPUT "Total: ", Total
OUTPUT "Average: ", Total / 5
```

循环**之前**把**总和**（total）设为 0，在循环里累加，循环**之后**再做除法。

■ 统计满足条件的元素个数

```
DECLARE Marks : ARRAY[1:5] OF INTEGER
DECLARE PassCount : INTEGER
DECLARE i : INTEGER
Marks[1] ← 56
Marks[2] ← 71
Marks[3] ← 48
Marks[4] ← 90
Marks[5] ← 65
PassCount ← 0
FOR i ← 1 TO 5
   IF Marks[i] >= 60 THEN
      PassCount ← PassCount + 1
   ENDIF
NEXT i
OUTPUT PassCount, " marks are 60 or more"
```

■ 最大值与最小值

从**第一个元素**开始，而不是从 0 开始。然后把其余每个元素都与它比较。

```
DECLARE Marks : ARRAY[1:5] OF INTEGER
DECLARE Highest : INTEGER
DECLARE Lowest : INTEGER
DECLARE i : INTEGER
Marks[1] ← 56
Marks[2] ← 71
Marks[3] ← 48
Marks[4] ← 90
Marks[5] ← 65
Highest ← Marks[1]
Lowest ← Marks[1]
FOR i ← 2 TO 5
   IF Marks[i] > Highest THEN
      Highest ← Marks[i]
   ENDIF
   IF Marks[i] < Lowest THEN
      Lowest ← Marks[i]
   ENDIF
NEXT i
OUTPUT "Highest ", Highest, ", lowest ", Lowest
```

为什么 `Lowest` 不从 0 开始？因为没有分数低于 0，`Lowest` 就会一直是 0，结果是错的。从 `Marks[1]` 开始永远正确，即使数组里有负数也一样。

■ 练习

1. 数组 `Temps[1:7]` 存放一周的气温。输出气温最高的那一天的编号（不是气温本身）。

<details> <summary> 参考答案 </summary>

```
DECLARE Temps : ARRAY[1:7] OF INTEGER
DECLARE HottestDay : INTEGER
DECLARE Day : INTEGER
Temps[1] ← 21
Temps[2] ← 25
Temps[3] ← 19
Temps[4] ← 28
Temps[5] ← 24
Temps[6] ← 28
Temps[7] ← 22
HottestDay ← 1
FOR Day ← 2 TO 7
   IF Temps[Day] > Temps[HottestDay] THEN
      HottestDay ← Day
   ENDIF
NEXT Day
OUTPUT "Hottest day: ", HottestDay
```

输出 4。要记录目前最佳元素的**索引**，而不是它的值。第 6 天也是 28 度，但 `>` 不会用相等的值去替换，所以保留的是第一个最热的日子。

</details>

常见错误

- 把 `Total ← 0` 写在循环里面。这样每一遍都会重新清零。
- 求平均值时除错了数。要除以实际累加的值的个数。
- 题目要的是**第一个**最大值时却用了 `>=`。遇到相等的值时 `>=` 会再次替换，结果得到的是**最后一个**最大值。