选择结构

Ref As Pseudocode

IF、THEN、ELSE、ENDIF

选择（selection）是指程序根据一个**条件**（condition）来决定下一步做什么。条件是一个结果为 TRUE 或 FALSE 的表达式。

■ 不带 ELSE 的 IF

```
IF <condition> THEN
   <statement(s)>
ENDIF
```

```
DECLARE Temperature : INTEGER
Temperature ← 31
IF Temperature > 30 THEN
   OUTPUT "Hot day"
   OUTPUT "Drink water"
ENDIF
OUTPUT "Have a good day"
```

如果条件为 TRUE，就执行 THEN 和 ENDIF 之间的语句；如果为 FALSE，就跳过它们。无论哪种情况，程序之后都会继续执行 ENDIF 后面的内容。

■ 带 ELSE 的 IF

```
IF <condition> THEN
   <statement(s)>
ELSE
   <statement(s)>
ENDIF
```

```
DECLARE Mark : INTEGER
Mark ← 41
IF Mark >= 50 THEN
   OUTPUT "Pass"
ELSE
   OUTPUT "Fail"
ENDIF
```

两个**分支**（branches）中恰好执行**一个**：不会两个都执行，也不会两个都不执行。

■ 9618 的排版

- `THEN` 写在 `IF` **这一行的末尾**。
- `ELSE` 和 `ENDIF` 与 `IF` 对齐。
- 每个分支里的语句都要缩进。

常见错误

- 忘记写 `ENDIF`。每个 `IF` 都需要一个。
- `IF Mark ← 50 THEN`。箭头用来存值；条件中的比较要用 `=`。
- 把 `THEN` 单独写在 `IF` 的下一行。那是 **IGCSE 0478** 的排版；在 9618 中它写在 `IF` 这一行上。

嵌套 IF

放在另一个 `IF` 里面的 `IF` 叫作**嵌套**（nested）`IF`。当结果多于两种，或者第二个问题只有在第一个问题之后才有意义时，就使用它。

```
DECLARE Mark : INTEGER
Mark ← 64
IF Mark >= 70 THEN
   OUTPUT "Distinction"
ELSE
   IF Mark >= 50 THEN
      OUTPUT "Merit"
   ELSE
      OUTPUT "Fail"
   ENDIF
ENDIF
```

- 内层 `IF` 位于外层的 `ELSE` 里面，并且有它**自己**的 `ENDIF`。
- 数一数：两个 `IF` 就需要两个 `ENDIF`。
- **先判断最高的分界线。**内层的 `Mark >= 50` 只有在分数已经低于 70 时才会执行，所以不需要再写 `AND Mark < 70`。

剑桥伪代码中**没有** `ELSEIF` 或 `ELIF` 关键字。要先写 `ELSE`，再在里面写一个新的 `IF`，就像上面那样。

■ 在第一个问题里再问第二个问题

有时内层 IF 位于 THEN 分支里：

```
DECLARE ChallengerScore : INTEGER
DECLARE ChampionScore : INTEGER
DECLARE HighestScore : INTEGER
ChallengerScore ← 95
ChampionScore ← 80
HighestScore ← 90
IF ChallengerScore > ChampionScore THEN
   IF ChallengerScore > HighestScore THEN
      OUTPUT "New champion and new highest score"
   ELSE
      OUTPUT "New champion"
   ENDIF
ELSE
   OUTPUT "The champion stays"
ENDIF
```

常见错误

- 漏掉内层的 ENDIF。这样后面的每一行看起来都属于错误的代码块。
- 先判断最低的分界线：如果先写 IF Mark >= 50，85 分会输出 Merit，永远到不了 Distinction 的判断。
- 写了 ELSE IF，却只用一个 ENDIF 结束。每个 IF 都要分别结束。

CASE OF 多分支

CASE OF 根据**一个变量的值**选择一个分支。当你反复判断同一个变量时，它比许多层嵌套的 IF 更清楚。

```
CASE OF <identifier>
   <value 1> : <statement(s)>
   <value 2> : <statement(s)>
   OTHERWISE : <statement(s)>
ENDCASE
```

```
DECLARE Move : CHAR
DECLARE Position : INTEGER
Position ← 50
Move ← 'A'
CASE OF Move
   'W' : Position ← Position - 10
   'S' : Position ← Position + 10
   'A' : Position ← Position - 1
   'D' : Position ← Position + 1
   OTHERWISE : OUTPUT "Unknown key"
ENDCASE
OUTPUT Position
```

- 每一行是：一个**值**、一个冒号 :、然后是要做的事。
- 各个情况**按顺序**判断。第一个匹配的情况被执行，然后整个 CASE 就结束了，其余情况不再判断。
- OTHERWISE 可以不写。如果写了，它必须放在**最后**，在没有任何值匹配时执行。在 9618 中它要带冒号：OTHERWISE :。

■ 用 TO 表示范围

在 9618 中，一个情况可以是一个**范围**（range）：<value1> TO <value2>，两端都包括在内。

```
DECLARE Mark : INTEGER
Mark ← 64
CASE OF Mark
   70 TO 100 : OUTPUT "A"
   60 TO 69 : OUTPUT "B"
   50 TO 59 : OUTPUT "C"
   OTHERWISE : OUTPUT "U"
ENDCASE
```

■ 一个情况里有多条语句

一个情况可以执行不止一条语句。把它们写在接下来的几行，并缩进：

```
DECLARE Choice : INTEGER
DECLARE Balance : REAL
Balance ← 100.0
Choice ← 2
CASE OF Choice
   1 : OUTPUT "Balance: ", Balance
   2 :
      Balance ← Balance - 20.0
      OUTPUT "Took out 20. Balance: ", Balance
   OTHERWISE : OUTPUT "Not a menu choice"
ENDCASE
```

■ 用 IF 还是 CASE?

- 一个变量与**固定的值或范围**比较：CASE OF 往往更清楚。
- 条件涉及**多个变量**，或用到 AND / OR：必须用 IF。

■ 练习

1. Day 存放 1 到 7 之间的一个数。用 CASE OF 实现：1 到 5 输出 "Weekday"，6 或 7 输出 "Weekend"，其他值输出 "Error"。

2. 用嵌套 IF 实现：0–12 岁输出 "Child"，13–19 岁输出 "Teen"，20 岁及以上输出 "Adult"。

<details> <summary> 参考答案 </summary>

```
DECLARE Day : INTEGER
Day ← 6
CASE OF Day
   1 TO 5 : OUTPUT "Weekday"
   6 TO 7 : OUTPUT "Weekend"
   OTHERWISE : OUTPUT "Error"
ENDCASE
```

```
DECLARE Age : INTEGER
Age ← 15
IF Age <= 12 THEN
   OUTPUT "Child"
ELSE
   IF Age <= 19 THEN
      OUTPUT "Teen"
   ELSE
      OUTPUT "Adult"
   ENDIF
ENDIF
```

</details>

常见错误

- 在 `CASE OF` 中写条件，例如 `Mark >= 50 :`。一个情况只能是一个值或一个范围，绝不能是比较式。
- 把 `OTHERWISE` 放在最后以外的位置。
- 忘记写 `ENDCASE`。
- `OTHERWISE` 后面不加冒号。那是 IGCSE 的写法；9618 写作 `OTHERWISE :`。