# 选择结构

Ref Pseudocode

## IF, THEN, ELSE, ENDIF

**选择**结构根据一个条件执行一组语句或另一组语句。

```
IF <condition>
  THEN
    <statements>
ENDIF
```

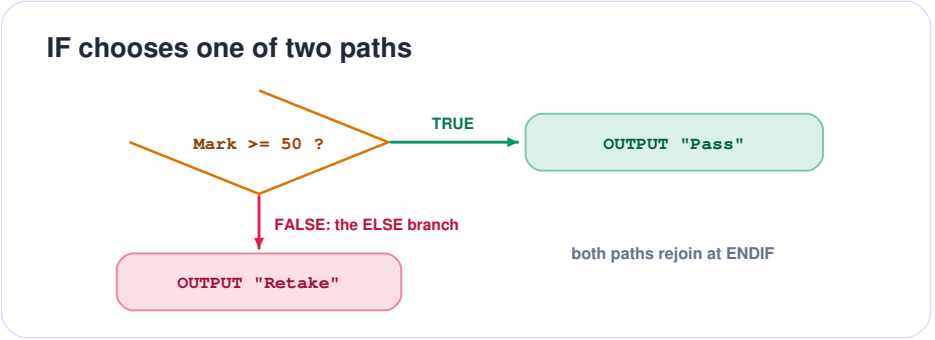
```
DECLARE Mark : INTEGER
Mark ← 72
IF Mark >= 50
  THEN
    OUTPUT "Pass"
ENDIF
```

对另一种情况添加 ELSE。两个分支中恰好只有一个被执行 — 不会同时执行，也不会都不执行。都不会。

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

排版是学生容易扣分的地方，因此请将其视为一种固定格式来记忆：

- IF 及其条件写在同一行，**条件后不留任何内容**。
- THEN 和 ELSE 各占一行，缩进 **两个**空格。
- 语句缩进 **四个**空格。
- ENDIF 回到其对应 IF 所在的列位置。




*IF* 在两条路径中选择一条；两者在 *ENDIF* 处汇合

## 嵌套

嵌套在 IF 中的 IF 可处理两个以上结果。每个结果都需要自己的 ENDIF。

```
DECLARE Mark : INTEGER
Mark ← 85
IF Mark >= 80
  THEN
    OUTPUT "Distinction"
  ELSE
    IF Mark >= 50
      THEN
        OUTPUT "Pass"
      ELSE
        OUTPUT "Fail"
    ENDIF
ENDIF
```

### 常见错误

- 遗漏 ENDIF。每一个 IF 都必须闭合，缺失任何一个都是语法错误，标记会一目了然地显示出来。标记会一目了然地显示出来。
- IF Mark >= 50 THEN OUTPUT "Pass" 写在一行上。**这种形式不存在于任何**课程大纲中；请写成三行。
- 将 IF Mark = 50 THEN 写成 IF Mark ← 50。箭头表示赋值；= 表示比较。

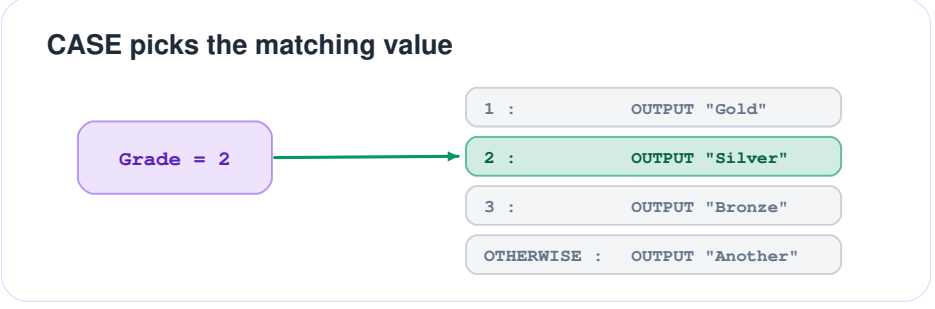
## CASE OF

当一个变量与多个**单个值**进行比较时，使用 CASE OF 表述要清晰得多比一叠嵌套的 IFs 更清楚。

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

```
DECLARE Choice : INTEGER
Choice ← 2
CASE OF Choice
    1 : OUTPUT "You chose north"
    2 : OUTPUT "You chose south"
    3 : OUTPUT "You chose east"
    4 : OUTPUT "You chose west"
    OTHERWISE OUTPUT "That is not a direction"
ENDCASE
```

- 每个分支都是一个**值**，后跟冒号，再跟要执行的操作。
- OTHERWISE 捕获所有被列出的值遗漏的内容。它是可选的，写一个是良好的编程习惯 — 这表示当用户输入意外内容时会发生什么。
- 第一个匹配的分支会运行，其余分支将被跳过。




*CASE* 将值与每个选项逐一比较；*Grade = 2* 匹配第二个

### ■ 范围和一个字符

```
DECLARE Grade : CHAR
Grade ← 'B'
CASE OF Grade
    'A' : OUTPUT "Excellent"
    'B' : OUTPUT "Good"
    'C' : OUTPUT "Satisfactory"
    OTHERWISE OUTPUT "Unclassified"
ENDCASE
```

### 常见错误

- 使用 CASE OF 作为**条件**。它对一个变量进行值测试；而 Mark >= 50 需要一个 IF。
- 忘记 ENDCASE。
- 省略 OTHERWISE 然后疑惑为什么意外的值完全没有任何反应。