

選擇敘述

C 是一種程序性語言（procedure language），表示它是一行接一行執行敘述的，不過有些敘述在某些條件成立時，才會被執行的，因此，需要選擇敘述（selection statement）來輔助之。

C 的選擇敘述計有 `if`、`if...else`、`else if`、`switch...case`。以下我們將一一的探討之。在未進入主題前，先來談幾個在選擇敘述與下一章的迴圈敘述皆會用到的 `bool` 資料型態和關係運算子。

3-1 bool 型態

`bool` 型態（或稱布林型態）是 `Boolean`（布林）的縮寫。它表示不是 `true` 就是 `false`。在 C 語言，除了 0 以外，其餘的數字皆為真，亦即只有 0 的數值是假，若是 1 或 -1，則是真。C 的真，是以 `true` 表示，而假，是使用 `false` 表示，注意，都是小寫英文字母喔！

 範例程式：p[3-1].c

```
01  #include <stdio.h>
02  #include <stdbool.h>
03  int main(void)
04  {
05      bool yesOrNo = true;
06      printf("#1: %d\n", yesOrNo);
07
08      yesOrNo = false;
09      printf("#2: %d\n", yesOrNo);
10  }
```

```
11     yesOrNo = 11;
12     printf("#3: %d\n", yesOrNo);
13
14     yesOrNo = 0;
15     printf("#4: %d\n", yesOrNo);
16
17     yesOrNo = -1;
18     printf("#5: %d\n", yesOrNo);
19
20     yesOrNo = 21;
21     printf("#6: %d\n", yesOrNo);
22
23     return 0;
24 }
```

```
#1: 1
#2: 0
#3: 1
#4: 0
#5: 1
#6: 1
```

從輸出結果發現，若是 `true` 值，則是以 1 印出，而 `false` 則以 0 印出。此程式記得要載入 `stdbool.h` 標頭檔。

程式中的

```
yesOrNo = 11;
```

敘述表示將 11 指定給 `yesOrNo`，因為它不是 0，此時因為 `yesOrNo` 型態是 `bool`，所以它的值為 `true`，輸出的答案是 1。其他的敘述就不再贅述。

3-2 關係運算子

選擇敘述中的條件式，其答案不是 `true`，就是 `false`。而這些條件式可是由關係運算子（`relational operator`）和運算元所組成的。

關係運算子又稱比較運算子（`comparative operator`）。表 3-1 是 C 語言的關係運算子。運算子是一種符號，用以表示某種功能罷了。

表 3-1 C 語言的關係運算子

關係運算子	功能	範例 (<code>int x=100;</code>)	結果
<code><</code>	小於	<code>x < 100</code>	false
<code><=</code>	小於等於	<code>x <= 100</code>	true
<code>></code>	大於	<code>x > 200</code>	false
<code>>=</code>	大於等於	<code>x >= 100</code>	true
<code>==</code>	等於	<code>x == 100</code>	true
<code>!=</code>	不等於	<code>x != 100</code>	false

表中要注意的是，等於運算子（`==`），是由兩個 `=` 所組成，若只有一個 `=`，則是指定或設定的運算子。表 3-1 中的範例可以輔助你了解其功能。

練習題

3.1 假設 `x` 是 1，試問以下的運算式何者為 `true`。

- | | |
|---------------------------|----------------------------|
| (a) <code>x > 0</code> | (d) <code>x >= 0</code> |
| (b) <code>x < 0</code> | (e) <code>x != 1</code> |
| (c) <code>x != 0</code> | (f) <code>x <= 1</code> |

3.2 試問以下片段程式的輸出結果。

```
#include <stdio.h>
#include <stdbool.h>
int main(void)
{
    bool b = true;
    int i = b;
    printf("b: %d\n", b);
    printf("i: %d\n", i);

    return 0;
}
```

3-3 if 敘述

當你只想在某一條件為真時，執行其對應的敘述時，而條件為假時，不予以理會，此時將會以 if 敘述為之，其語法如下：

```
if (條件判斷式) {
    條件判斷式為真時，執行的敘述
}
```

上述語法對應的流程圖，如圖 3-1 所示：

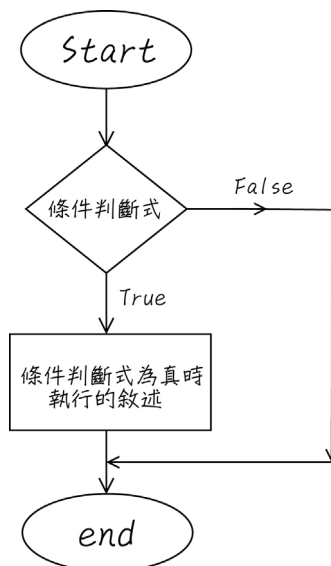


圖 3-1 if 敘述流程圖

我們來看一些範例與其說明。



範例程式：p[3-3]-1.c

```

01 | #include <stdio.h>
02 | int main(void)
03 | {
04 |     int num;
05 |     printf("Enter an integer: ");
06 |     scanf("%d", &num);
07 |     if (num > 0) {
08 |         printf("%d 是正整數\n", num);
09 |     }
10 |     printf("Over\n");
11 |     return 0;
12 | }
```

```
Enter an integer: 50
50 是正整數
Over
```

```
Enter an integer: -3
Over
```

此程式的 `if` 判斷式，若為真時，所要執行的敘述只有一個時，是可以省略左、右大括號，但若執行的敘述有二個或以上時，則必須加上左、右大括號，將其括起來，否則，它只會執行一個敘述而已，如以下的範例程式。



範例程式：p[3-3]-2.c

```
01  #include <stdio.h>
02  int main(void)
03  {
04      int num;
05      printf("Enter an integer: ");
06      scanf("%d", &num);
07      if (num > 0) {
08          printf("%d", num);
09          printf(" 是正整數\n");
10      }
11      printf("Over\n");
12      return 0;
13  }
```

```
Enter an integer: 50
50 是正整數
Over
```

```
Enter an integer: -3
Over
```

此程式的 `if` 條件式若為真時，所執行的敘述以二個敘述來表示，因此要加上左、右大括號。我的處理方法是，不管是執行一個或多個敘述，都加上左、右大括號，這樣子在下次再維護時，可能會加上條件式為真時的執行敘述，就不會出現錯誤的訊息。

再來看一範例，請你輸入一半徑，若半徑大於 0，才計算它的面積並加以輸出，若小於等於 0，就不予以理會。



範例程式：p[3-3]-3.c

```

01  #include <stdio.h>
02  #include <math.h>
03  int main(void)
04  {
05      int radius;
06      double area;
07      printf("輸入一半徑: ");
08      scanf("%d", &radius);
09      if (radius > 0) {
10          area = M_PI * radius * radius;
11          printf("area = %.2f\n", area);
12      }
13      printf("Over\n");
14      return 0;
15  }

```

```

輸入一半徑: 10
area = 314.159
Over

```

```

輸入一半徑: -10
Over

```

**練習題**

3.3 試問下一程式，若輸入 100 和 -100 時，其分別輸出結果為何？

```

#include <stdio.h>
int main(void)
{
    int i;
    printf("Enter an integer: ");
    scanf("%d", &i);
    if (i > 0)
        printf("%d", i);
        printf(" 是正整數\n");
    printf("Over\n");
    return 0;
}

```

3.4 請撰寫一個 if 敘述，使其在 x 大於 0 時，指定數值 1 給 y。

3.5 請撰寫一個 if 敘述，使其在考績（appraisal）大於 85 時，將薪資（salary）提高 2%。薪資初始值為 36000，考績由主管輸入。

3.6 試修正以下的程式碼。

```
#include <stdio.h>
int main(void)
{
    int radius = 2;
    if radius >= 0
        area = radius * radius * PI;
    printf("Area of circle is %.2f\n", salary);
    return 0;
}
```

3-4 if...else

以上的 if 選擇敘述只關心條件式為 true 時，要處理的事項，而條件式為假時，不予以理會。但有時候，我們要執行條件式為 true 或 false 時都有要處理事項，那該怎麼做呢？這就是本節要討論的 if...else 選擇敘述，其語法如下：

```
if (條件判斷式) {
    條件判斷式為真時，執行的敘述
}
else {
    條件判斷式為假時，執行的敘述
}
```

一樣的道理，若要執行的敘述有兩個或兩個以上時，則必須要有左、右大括號。切記，切記。此語法對應的流程圖，如圖 3-2 所示：

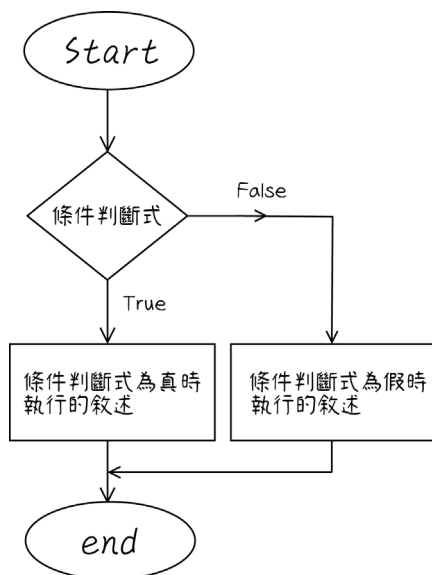


圖 3-2 if...else 敘述流程圖

從 C 邁向 C++

學完了 C 語言，下一步就是 C++ 程式語言。C 是傳統的程序性語言（*procedure language*），而 C++ 可說是物件導向程式語言（*object-oriented programming language*），C 可以說是 C++ 的部分集合。物件導向程式語言有三大特性，包括封裝（*encapsulation*），繼承（*inheritance*），以及多型（*polymorphism*）。在未論述這三大特性之前，先來談談輸出與輸入。

14-1 C++ 的輸出與輸入

C++ 的輸出和輸入皆是利用串流運算子的概念來完成的。其實它比 C 的輸出與輸入來得更簡單直覺。記得要載入 `iostream` 來檢查輸出與輸入的語法是否正確，並且設定命名空間（*namespace*），如下所示：

```
#include <iostream>
using namespace std;
```

還有 C++ 程式的延伸檔名是 `.cpp` 喔！表示（*c plus plus*）

14-1-1 輸出串流運算子：<<

C++ 的輸出是利用 `cout` 加上 `<<` 輸出串流運算子（兩個向左的雙箭頭）來實現，將資料輸出於螢幕。如下一範例程式所示：

 範例程式：p[14-1-1]-1.cpp

```
01 | #include <iostream>
02 | using namespace std;
```

```

03 |
04 | int main()
05 | {
06 |     cout << "Hello, world";
07 |     return 0;
08 | }

```

```
Hello, world
```

此程式表示將 `Hello, world` 輸出於螢幕。若要跳行，則可利用 `endl`，或是直接加 `\n` 於輸出字中，如下所示：

 範例程式：p[14-1-1]-2.cpp

```

01 | #include <iostream>
02 | using namespace std;
03 |
04 | int main()
05 | {
06 |     cout << "Hello, world" << endl;
07 |     cout << "C++ programming is fun\n";
08 |     cout << "Learning C++ now!";
09 |     return 0;
10 | }

```

```
Hello, world
C++ programming is fun
Learning C++ now!
```

14-1-2 輸入串流運算子：>>

C++ 的輸入是利用 `cin` 加上 `>>` 輸入串流運算子（兩個向右的雙箭頭）來實現，從鍵盤輸入資料存於給予的變數。如下範例程式

 範例程式：p[14-1-2]-1.cpp

```

01 | #include <iostream>
02 | using namespace std;
03 |
04 | int main()
05 | {
06 |     int number;
07 |     cout << "Enter an integer: ";
08 |     cin >> number; //從鍵盤輸入資料，並儲存於 number 變數
09 |     cout << "number = " << number;

```

```

10 |         return 0;
11 |     }

```

```

Enter an integer: 100
number = 100

```

14-1-3 格式化的輸出

資料輸出時有時要加以格式化才會整齊美觀，以下是幾個常用的方法，如 14-1 所示：

表 14-1 C++ 常用的輸出格式化方法

方法	功能
setw(n)	設定 n 個欄位寬
left	向左靠齊
right	向右靠齊
setprecision(n)	設定有效位數
setfill('*')	以 * 填滿空白處。你也可以用其他字元
fixed	印出小數點後六位數的浮點數

以上的方法，要載入 `iomanip` 標頭檔才能執行。

 範例程式：p]14-1-3]-1.cpp

```

01 | #include <iostream>
02 | #include <iomanip>
03 | using namespace std;
04 |
05 | int main()
06 | {
07 |     double d = 123.456;
08 |     cout << setfill('*');
09 |     cout << setw(10);
10 |     cout << d << endl;
11 |     cout << setw(7) << setprecision(4) << d << endl;
12 |     return 0;
13 | }

```

```

***123.456
**123.5

```

此程式 `setw(10)` 表示給予十個欄位寬，利用 `setfill('*')` 將空白處以 `*` 充填之。而 `setprecision(4)` 表示有效位數是 4 位，請注意，`setfill('*')` 是永久性的設定，而 `setw(10)` 和 `setw(7)` 皆為暫時性，表示用完一次輸出即無效。

`fixed` 是針對浮點數而來，若沒有加此方法，則只會印出六位有效位數，不包含小數點。

 範例程式：p]14-1-3]-2.cpp

```
01 | #include <iostream>
02 | #include <iomanip>
03 | using namespace std;
04 |
05 | int main()
06 | {
07 |     double d = 123.4567;
08 |     cout << d << endl;
09 |     cout << fixed;
10 |     cout << d << endl;
11 |     return 0;
12 | }
```

```
123.457
123.456700
```

此程式的 `d` 是浮點數變數，但第一次印出來是 123.457，只會印出六位有效位數，有四捨五入喔。加上 `fixed` 後，就會變為 123.456700，是指印出小數點後六位數字，由於 `d` 的浮點數後面才 4 位數字，所以小數點後面會補二個 0。

若你指定印出浮點數的小數點後面 2 位數，則必須以 `fixed` 和 `setprecision(2)` 來完成。如以下範例程式所示：

 範例程式：p]14-1-3]-3.cpp

```
01 | #include <iostream>
02 | #include <iomanip>
03 | using namespace std;
04 |
05 | int main()
06 | {
07 |     double d = 123.4567;
08 |     cout << setfill('*');
09 |     cout << setw(10);
10 |     cout << fixed << setprecision(2) << d << endl;
```

```

11     return 0;
12 }

```

```

****123.46

```

此程式利用 `fixed` 與 `setprecision(2)` 兩者一起作用，印出小數點後面 2 位數。

表 14-1 所列的輸出格式化方法，對輸出多個資料時，可以達到美觀的視覺感受。

 範例程式：p]14-1-3]-4.cpp

```

01 // using C++
02 #include <iostream>
03 #include <iomanip>
04 using namespace std;
05
06 int main()
07 {
08     int x=123, y=1245678, z=12345;
09     cout << setw(10) << x << setw(10) << y << setw(10) << z << endl;
10     cout << setw(10) << y << setw(10) << z << setw(10) << x << endl;
11     cout << setw(10) << z << setw(10) << x << setw(10) << y << endl;
12     return 0;
13 }

```

123	1245678	12345
1245678	12345	123
12345	123	1245678

輸出的資料預設是向右靠齊。每一個資料皆給予 10 個欄位寬。若要向左靠齊，C++ 是加上 `left` 方法，如下所示：

 範例程式：p]14-1-3]-5.cpp

```

01 #include <iostream>
02 #include <iomanip>
03 using namespace std;
04
05 int main()
06 {
07     int x=123, y=1245678, z=12345;
08     cout << left;
09     cout << setw(10) << x << setw(10) << y << setw(10) << z << endl;
10     cout << setw(10) << y << setw(10) << z << setw(10) << x << endl;
11     cout << setw(10) << z << setw(10) << x << setw(10) << y << endl;
12     return 0;
13 }

```

123	1245678	12345
1245678	12345	123
12345	123	1245678

其他有關浮點數與字串的格式化，由於篇幅有限的關係，就不再多談。

是談完了輸出與輸入之後，接下來，談談物件導向程式設計的第一個特性：封裝。

14-2 類別與物件

從封裝的字面上大致可以得知，它將某些事項包裝起來，其實是將資料成員（`data member`）或成員函式（`member function`）包裝在一個所謂類別（`class`）的名稱上。上述的資料成員簡稱資料，而成員函式為了與傳統程式設計所稱的函式有所區別，以方法（`method`）來稱呼與識別也是不錯的選擇。

在傳統的程式設計，主角是函式（`function`），寫好了一切就搞定了，而資料是次等公民，但如果資料不正確，函式再怎麼優，程式執行後產生的結果還是錯的，這就是所謂垃圾進垃圾出（`garbage in garbage out`，`GIGO`）。因此，資料和函式是同等重要的，應該一視同仁。

封裝的主要功能是將資料成員和成員函式裝在一起，視為同等的重要，而且也將資料成員和成員函式分區存放。其實類別的重要用意是要將資料成員加以保護或隱藏，這樣子會減少觸及資料成員，從而也會減少 `GIGO` 的發生，進而減少軟體的維護成本。

物件（`object`）為類別的實例（`instance`）。我們可對類別建立多個實例。物件與實例這兩個名詞常常交換使用。類別與物件之間的關係，就等同於草莓派食譜與草莓派之間的關係：我們可透過單一的草莓派食譜，製作多個草莓派。

也可以這樣說，物件是用來表示真實世界裡可被明確辨識的實體。比方說，大學生、研究生、椅子、圓形，香蕉，都可以被視為物件。類別定義了一個大原則，然後以此大原則定義一個或多個物件。例如在水果類別下，你可以定義香蕉、橘子、藍莓等物件。在程式的撰寫上，類別像是資料型態，物件像是變數。

利用 ChatGPT 有效地學習 C 程式設計

撰寫程式時，利用 ChatGPT 可以在許多階段發揮輔助作用，無論是初學者還是資深開發者都能受益。以下列出具體應用範疇與優點：

A-1 程式碼產生

- 根據需求描述，自動生成完整或部分程式碼
- 範例：
 - 「用 C 語言撰寫一個 Bubble Sort 片段程式」
 - 「寫一個 Hanoi tower 的程式」

以下是利用 ChatGPT 的問與答：

以C語言撰寫bubble sort的片段程式

當然可以，以下是用 C 語言撰寫的 Bubble Sort（氣泡排序）範例程式（片段），並附上中文註解：

✓ Bubble Sort 核心程式碼片段：

```
#include <stdio.h>

void bubbleSort(int arr[], int n) {
    int i, j, temp;
    for (i = 0; i < n - 1; i++) {           // 外層控制回合數，共需 n-1 輪比較
        for (j = 0; j < n - 1 - i; j++) {    // 每輪比較次數會遞減
            if (arr[j] > arr[j + 1]) {       // 如果前一項比後一項大 → 交換
                temp = arr[j];               // 交換兩數
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
```

✦ 使用範例（主程式）：

```
int main() {
    int data[] = {8, 3, 1, 6, 2, 5};
    int n = sizeof(data) / sizeof(data[0]);

    bubbleSort(data, n);    // 呼叫排序函式

    printf("排序後的結果：\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", data[i]);
    }

    return 0;
}
```

🗨 說明：

- 時間複雜度：最差與平均情況為 $O(n^2)$ ，適合小筆資料排序。
- 排序邏輯：每回合將「最大值冒泡」到右端。

如果你想要優化版本（提早結束排序）、或支援由小到大 / 由大到小排序，我也可以幫你擴充。

A-2 除錯與錯誤說明

- 分析錯誤訊息（`traceback` / `log`）
- 解釋語法錯誤、型別錯誤、執行時錯誤
- 提供修正建議或可能原因

以下是利用 ChatGPT 的問與答之範例

請修正以下的程式

```
#include <stdio.h>
int main()
{
    double d;
    scanf("%f", d);
    printf("%f", d);
    return 0;
}
```

你的程式目前有兩個錯誤會導致執行失敗或行為異常，以下是問題與修正：

✗ 原始錯誤說明：

```
double d;
scanf("%f", d);    // ✗ 應該傳地址 &d，且格式碼錯誤
printf("%f", d);   // ✔ 可接受 double，但建議用 %lf 更明確
```

錯誤類型	說明
%f 格式碼錯誤	scanf 中 %f 是用來輸入 float，輸入 double 時應該用 %lf
d 應加 &	scanf 需要變數的「記憶體位址」，所以要寫成 &d

✔ 修正後程式：

```
#include <stdio.h>

int main() {
    double d;
    printf("請輸入一個小數：");
    scanf("%lf", &d);           // 修正：%lf 並加上 &d
}
```