4.1 程式基本結構

程式的基本結構可概分為循序式結構（sequence structure）、選擇式結構（selection structure），與重複式結構（repetition structure）等三種，幾乎所有程式都是在循序式結構的基礎上，加上選擇式結構或重複式結構。

4.1.1 循序式結構

循序式結構（sequence structure）的程式是指程式依序從第一個敘述執行至最後一個敘述，如下面範例所示。循序式結構在前幾章已經介紹了很多，因此讀者也都很熟悉循序式結構的程式。

```
//循序式結構
#include <iostream>
using namespace std;

int main(int argc, char** argv)
{
   int Var = 5;                                    //第 1 步
   cout  << "Var 起始值 = " << Var;                 //第 2 步
   Var = 10;                                       //第 3 步
   cout  << "\nVar 變更值 = " << Var;               //第 4 步
   return 0;                                       //第 5 步
}
```

4.1.2 選擇式結構

選擇式結構（selection structure）的程式是指程式含有條件敘述，當條件敘述的條件成立（也就是條件運算式的值為 1）時，執行條件成立區的敘述。反之，當條件敘述的條件不成立（也就是條件運算式的值為 0）時，則結束條件選擇（是非結構）如下面範例一，或是執行條件不成立區的敘述（二選一結構）如下面範例二。

```
//選擇式結構一（是非結構）
#include <iostream>
using namespace std;

int main(int argc, char** argv)
{
   char inkey;                                  //第 1 步
```

```
    cout << "請按 Y 鍵，再按 Enter・・・"; //第 2 步
    cin >> inkey;                          //第 3 步
    if(inkey == 'Y' || inkey == 'y') {     //第 4 步
       cout << "祝您一路順風！\n";          //if 運算值為 1 時的第 5 步
    }                                      //if 運算值為 0 省略第 5 步
    return 0;                              //第 6 步
}
```

```
//選擇式結構二（二選一結構）
#include <iostream>
using namespace std;

int main(int argc, char** argv)
{
    char inkey;                            //第 1 步
    cout << "請按 Y 鍵，再按 Enter・・・"; //第 2 步
    cin >> inkey;                          //第 3 步
    if(inkey == 'Y' || inkey == 'y')       //第 4 步
       cout << "祝您一路順風！\n";          //if 運算值為 1 時的第 5 步
    else                                   //否則
       cout << "God Bless！\n";            //if 運算值為 0 時的第 5 步
    return 0;                              //第 6 步
}
```

選擇式結構除了是非結構（if）與二選一結構（if-else），另外還有多選一結構（if-elseif），將在 4.3 節詳細介紹。

4.1.3 重複式結構

重複式結構（repetition structure）的程式是指程式含有重複敘述，當重複敘述的條件成立（也就是條件運算式的值為 1）時，重複執行重複區的敘述。反之，當重複敘述的條件不成立（也就是條件運算式的值為 0）時，則結束重複結構，如下面範例。

重複式結構除了下面範例的後測試型迴圈結構（do-while loop）外，還有前測試型迴圈結構（while loop）與計數型迴圈結構（for loop），將在下一章詳細介紹。

```
//重複式結構
#include <iostream>
using namespace std;
int main(int argc, char** argv)
{
    int count = 1, sum = 0;                    //第 1 步
    do {                                       //第 2 步
       sum += count;                           //第 3 步 ◄──┐
       count ++;                               //第 4 步    │
    } while (count <= 10);                     //count<=10 返回第 3 步 ──┘
    return 0;
}
```

4.2 條件選擇

在日常生活中，通常處理某些事情時都事先分析判斷再作決定。例如，今天要不要帶電腦課本，此時您會先想想今天有沒有電腦課，有電腦課則要帶電腦課本，沒有則不帶電腦課本。同理，在程式設計中，也經常使用條件敘述，協助判斷並決定程式流程。例如，是否要結束程式時，通常先判斷使用者的輸入，若使用者輸入 “y” 或 “Y” 則結束程式，若使用者輸入 “n” 或 “N” 則不結束程式。

4.2.1 關係運算符號

微處理器（micro processor）中的算術邏輯單元（arithmetic logic unit）除了可以處理算術運算之外，還可以處理關係運算與邏輯運算。關係運算就是比較二個運算元的大小，包括大於、小於、大於等於、小於等於、等於、不等於。

關係運算符號（relational operators）包括 >、>=、<、<=、==、!= 等。關係運算的值為真或假，若為真則傳回值為 1（true），若為假則傳回值為 0（false）。表 4.1 列出各個關係運算符號與它們的功能、範例與說明。

表 4.1 關係運算符號

符號	功能	範例	說明
>	大於	a>b	若 a>b 則結果為真
<	小於	a<b	若 a<b 則結果為真
>=	大於等於	a>=b	若 a>=b 則結果為真
<=	小於等於	a<=b	若 a<=b 則結果為真
==	等於	a==b	若 a==b 則結果為真
!=	不等於	a!=b	若 a!=b 則結果為真

表 4.2 列出許多簡單的關係運算式與運算值，這些都只是數值形式的關係運算式，所以讀者很容易從關係運算式與它的運算值了解各個關係運算符號的功能。

表 4.2 關係運算式值

關係運算式	關係運算式值
4.3 > 3.7	1 (true)
2.5 < 0.25	0 (false)
6.125 >= 6.25	0 (false)
3.75 <= 3.75	1 (true)
1 != 1	0 (false)

表 4.3 列出算術運算符號與關係運算符號的優先運算順序，從表中可看出小括號第一優先，第二是算術運算中的乘除與餘數運算符號，第三是算術運算中的加減運算符號，第四才是關係運算符號。

表 4.3 運算符號的優先順序

運算符號（operators）	優先順序（priority）
( )	1
*, /, %	2
+, -	3
==, <, >, <=, >=, !=	4

下面範例提供一個算術與關係運算的混合運算式，第一式是原始的混合運算式，從此式可能很難看出運算的結果。第二式則是將優先運算的部分放入小括號中，因此可以很清楚的表示出混合運算式的優先運算順序，進而推算出運算式的值。所以筆者建議在混合運算式中盡量使用小括號來表示運算的順序。

```
18 % 4 * 5 - 7         <= 12 + 25 / 4 - 18            //第 1 式
(((18 % 4) * 5) - 7)   <= ((12 + (25 / 4)) - 18)      //第 2 式
((    2    * 5) - 7)   <= ((12 +     6   ) - 18)
(        10     - 7)   <= (       18       - 18)
         3              <=         0
                     false
```

下面三個範例提供變數的關係運算，實際上比較變數的關係就是比較變數值的關係，例如第一個範例首先宣告 a = 3 然後比較是否 a == 3，比較時先以 3 代入 a，然後比較是否 3 == 3。別忘了關係運算值是布林值（0 或 1），所以存放關係運算值的變數 x 必須宣告為 bool 型態。

```
int a = 3;
bool x = (a == 3);                                  //∵a 等於 3 ∴x=true
```

第二與第三個範例中，都使用 cout << (關係運算式);，因此直接輸出關係運算式值，而不儲存到變數中。

```
int a = 3, b = 6, c = 3;
bool x = (b >= a);                                  //∵b>a ∴x=1
bool y = (c >= a);                                  //∵c==a ∴y=1
cout << (a >= b);                                   //∵a<b ∴顯示 0
```

```
int a = 5, b = 5, c = 3;
bool x = (a == b);                                  //∵a==b ∴x=1
bool y = (a != b);                                  //∵a==b ∴y=0
cout << (a != c);                                   //∵a!=c ∴輸出 1
```

程式 4-01：關係運算符號練習

```
1.   //儲存檔名：d:\C++04\C0401.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   int main(int argc, char** argv)
6.   {
7.      int a = 1, b = 4, c = 4;                    //宣告並啟始 a, b, c 值
```

```
8.      cout << "a=" << a << endl;              //輸出字串、a 值、跳行
9.      cout << "b=" << b << endl;              //輸出字串、b 值、跳行
10.     cout << "c=" << c << endl;              //輸出字串、c 值、跳行
11.     bool x = a<b, y = c>=a;
12.     cout << "a<b 為 " << x << endl;         //輸出字串、a<b 值
13.     cout << "c>=a 為 " << y << endl;        //輸出字串、c>=a 值
14.     cout << "a!=b 為 " << (a!=b) << endl;   //輸出字串、a!=b 值
15.     cout << "a==c 為 " << (a==c) << endl;   //輸出字串、a==c 值
16.     system("PAUSE");
17.     return 0;
18. }
```

程式輸出

```
a=1
b=4
c=4
a<b 為 1
c>=a 為 1
a!=b 為 1
a==c 為 0
```

4.2.2 邏輯運算符號

邏輯運算符號（logical operators）包括 !、&&、||。邏輯運算的值為真或假，若為真則傳回值為 1（true），若為假則傳回值為 0（false）。

表 4.4 邏輯運算符號

符號	功能	範例	說明
!	邏輯 NOT	!(a==1)	若 a≠1 則結果為真
&&	邏輯 AND	a>1 && a<9	若 1<a<9 則結果為真
||	邏輯 OR	a<1 || a>9	若 a<1 或 a>9 則為真

表 4.5 列出 AND 與 OR 運算的真值表。AND 邏輯運算（&&）是二個運算式皆為 1（true）則運算值為 1（true），其餘的運算值為 0（false）。OR 邏輯運算（||）是有一個運算式為 1（true）則運算值為 1（true），其餘的運算值為 0（false）。口訣如下：

AND 運算：有一個運算式為 0（false）則 A && B 為 0（false）。
OR 運算：有一個運算式為 1（true）則 A || B 為 1（true）。

表 4.5 邏輯 AND 與 OR 運算真值表

運算式 A	運算式 B	A && B	A || B
0 (false)	0 (false)	0 (false)	0 (false)
0 (false)	1 (true)	0 (false)	1 (true)
1 (true)	0 (false)	0 (false)	1 (true)
1 (true)	1 (true)	1 (true)	1 (true)

表 4.6 列出 NOT 運算的真值表，NOT 輯運算（!）是符號右邊的運算式為 0（false）時其運算值為 1（true），而符號右邊的運算式為 1（true）時其運算值為 0（false）。

表 4.6 邏輯 NOT 運算真值表

運算式 A	!A
0 (false)	1 (true)
1 (true)	0 (false)

表 4.7 列出許多簡單的邏輯運算式與運算值，這些都只是數值形式的邏輯運算式，所以讀者很容易從邏輯運算式與它的運算值了解各個邏輯運算符號的功能。

表 4.7 邏輯運算式值

邏輯運算式	邏輯運算式值
(4.3 > 3.7) && (7 == 5 + 2)	1 (true)
(2.5 < 0.25) || (7 == 15 / 2)	1 (true)
!(15 >= 10 + 5)	0 (false)

表 4.8 列出算術運算、關係運算、與邏輯運算符號的優先運算順序。若運算式中包含多個小括號則內層先運算再向外層運算，其餘的同等級的運算符號都是由左而右運算。

表 4.8 運算符號的優先順序

運算符號（operators）	優先順序（priority）
( )	由內而外運算
!	由左而右運算
*, /, %	由左而右運算
+, -	由左而右運算
==, <, >, <=, >=, !=	由左而右運算
&&	由左而右運算
||	由左而右運算

下面範例提供一個算術與邏輯運算的混合運算式，它是將優先運算的部分放入小括號中，因此可以很清楚的表示出混合運算式的優先運算順序，進而推算出運算式的值。所以筆者建議在混合運算式中盡量使用小括號來表示運算的順序。

```
!(4.3 > 3.7)  && (7 == 5 + 2) || (15 / 3 < 5)
!(   true    ) && (   true    ) || (  5  <  5  )
 (  false    ) && (   true    ) || (   false   )
(    false   ) && (   true    ) || (   false   )
(            false             ) || (   false   )
                    false
```

程式 **4-02**：邏輯與關係運算符號練習

```
1.   //儲存檔名：d:\C++04\C0402.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   int main(int argc, char** argv)
6.   {
7.      int a = 1, b = 4, c = 4;                //宣告並啟始 a, b, c 值
8.      bool x = !(a < b && b == c || c <=a);   //宣告並指定 x 值
9.      cout << "a=" << a << endl;              //輸出字串、a 值、跳行
10.     cout << "b=" << b << endl;              //輸出字串、b 值、跳行
11.     cout << "c=" << c << endl;              //輸出字串、c 值、跳行
12.     cout << "!(a<b && b==c || c<=a) = " << x << endl; //輸出字串、x 值
13.     system("PAUSE");
14.     return 0;
15.  }
```

程式輸出

```
a=1
b=4
c=4
!(a<b && b==c || c<=a) = 0
```

4.2.3 if 敘述

```
if (條件運算式)
    C++ 敘述;
```

- **if 敘述（if statement）**是是非結構。若條件運算式的結果為 1（true）則執行 if 的下一個敘述後結束 if，若條件運算式的結果為 0（false）則跳過 if 的下一個敘述而結束 if。

下面範例是判斷輸入字元是否為 'Y' 或 'y'。利用 if 敘述判斷輸入值 letter 是否等於字元 'Y' 或 'y'，若等於則顯示 "Yes" 訊息，若不等於則不顯示 "Yes" 訊息而結束。

```
cin >> letter;                                  //輸入 letter 字元
if (letter == 'Y' || letter == 'y')             //若 letter = 'Y'或'y'
   cout << "Yes";                                //則顯示"Yes"
```

下面範例是判斷輸入值是否為 3 的倍數。利用 if 敘述判斷輸入值 number 除 3 的餘數是否等於 0，若餘數為 0 表示 number 是 3 的倍數則顯示一訊息，若餘數不為 0 表示 number 不是 3 的倍數則不顯示訊息而結束。

```
cin >> number;                                  //輸入 number 資料
if (number % 3 == 0)                            //若 number/3 餘數為 0
   cout << number << " 為 3 的倍數";             //則顯示訊息
```

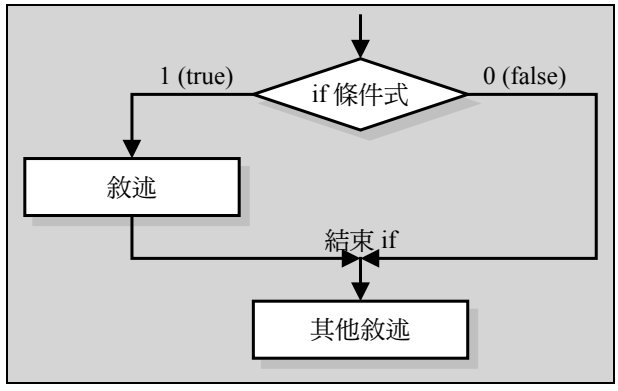
下面範例是判斷輸入值是否為大於 0。利用 if 敘述判斷輸入值 num 是否大於 0，若大於 0 則執行 sum+=num 敘述，若小於 0 則不執行 sum+=num 敘述而結束。

```
sum = 0
cin >> num;                                     //輸入 num 資料
if (num > 0)                                    //若 num 大於 0
   sum += num;                                  //sum=sum+num
```

上面範例的 if 敘述與 sum+=num; 敘述可以合併成一行敘述如下。

```
sum = 0
cin >> num;                                    //輸入 num 資料
if (num > 0) sum += num;                       //若 num>0 則 sum+=num
```

下圖是 if 敘述的結構圖，它顯示條件運算式的結果為 1（true）則執行敘述 1，條件運算式的結果為 0（false）則跳過敘述 1。




```
if (條件運算式)
{
      敘述 1;
      敘述 2;
      .
      敘述 n;
}
```

- **if 區塊（if block）**的工作方式與 if 敘述的工作方式相似。若條件運算式的結果為 1（true）則執行區塊中所有的敘述，若條件運算式的結果為 0（false）則不執行區塊中任何敘述而結束 if 區塊。
- **{ }** 大括號指示 if 區塊的起始與結束位置，若省略大括號則變成 if 的單行敘述。

下面範例是從 if 單行敘述修改而來，當輸入值 num 是否大於 0 則執行 sum+=num 敘述與 cout << sum; 敘述，若小於 0 則不執行區塊中的任何敘述而結束。

```
sum = 0
cin >> num;                                    //輸入 num 資料
if (num > 0)                                   //若 num 大於 0
{                                              //if 區塊起始
   sum += num;                                 //sum=sum+num
   cout << sum;                                //顯示 sum 值
}                                              //if 區塊結束
```

程式 **4-03**：判斷正數

```
1.   //儲存檔名：d:\C++04\C0403.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   int main(int argc, char** argv)
6.   {
7.      int input;
8.
9.      cout << "請輸入一個整數：";
10.     cin >> input;                            //輸入字元並存入 input
11.     if (input >= 0) {                        //若 input >= 0
12.        cout << input << " 是正數。\n";       //  顯示訊息
13.     }                                        //若 input < 0 則結束
14.     system("PAUSE");
15.     return 0;
16.  }
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入一個整數：25 Enter
25 是正數。
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入一個整數：-25 Enter
```

上面程式的 if 區塊只包含一個敘述，所以可以省略 if 起始與結束的大括號。

4.2.4 if-else 敘述

```
if (條件運算式)
    敘述 1;
else
    敘述 2;
```

- **if-else 敘述（if-else statement）**是二選一的結構。若條件運算式的結果為 1（true）則執行 if 區塊的敘述後結束 if，若條件運算式的結果為 0（false）則執行 else 區塊的敘述後結束 if。

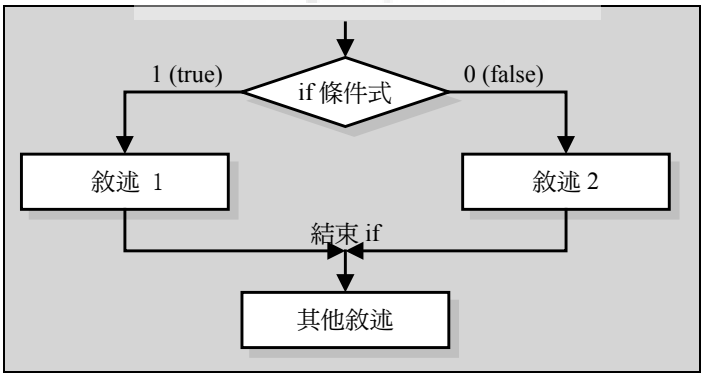
下面範例是判斷輸入字元是否為 'Y' 或 'y'。利用 if 敘述判斷輸入值 letter 是否等於字元 'Y' 或 'y'，若等於則顯示 "Yes" 訊息，若不等於則顯示 "No" 訊息而結束。

```
cin >> letter;                                  //輸入 letter 字元
if (letter == 'Y' || letter == 'y')             //若 letter == 'Y'或'y'
   cout << "Yes";                               //則顯示"Yes"
else                                            //否則
   cout << "No";                                //則顯示"No"
```

下面範例是判斷輸入值是否為 3 的倍數。利用 if 敘述判斷輸入值 number 除 3 的餘數是否等於 0，若餘數為 0 表示 number 是 3 的倍數則顯示 "是 3 的倍數"，若餘數不為 0 表示 number 不是 3 的倍數則顯示 "不是 3 的倍數" 後結束。

```
cin >> number;                                  //輸入 number 資料
if (number % 3 == 0)                            //若 number/3 餘數為 0
   cout << number << " 是 3 的倍數";             //則顯示訊息 1
else                                            //否則
   cout << number << " 不是 3 的倍數";           //則顯示訊息 2
```

下圖是 if-else 敘述的結構圖，它顯示條件運算式的結果為 1（true）則執行敘述 1 後結束 if，條件運算式的結果為 0（false）則執行敘述 2 後結束 if。




```
if (條件運算式)
{
    敘述區 1;
} else {
    敘述區 2;
}
```

- **if-else 區塊（if-else block）**的工作方式與 if-else 敘述的工作方式相似。若條件運算式的結果為 1（true）則執行區塊 1 中所有的敘述後結束 if，若條件運算式的結果為 0（false）則執行區塊 2 中所有的敘述後結束 if。
- **{ }** 大括號指示 if 區塊的起始位置與結束位置，若省略大括號則變成 if 的單行敘述。
- **敘述區**中可包含一個或多個 C++ 敘述。

程式 **4-04**：**判斷正、負數**

```
1.  //儲存檔名：d:\C++04\C0404.cpp
2.  #include <iostream>
3.  using namespace std;
4.
5.  int main(int argc, char** argv)
6.  {
7.     int number;
8.
9.     cout << "請輸入一個整數：";
```

```
10      cin >> number;                          //輸入字元並存入 number
11.     if (number >= 0) {                      //若 number >= 0
12.        cout << number << " 是正整數。\n";    //  顯示訊息
13.     }                                       //if 區塊結束點
14.     else {                                  //若 number < 0
15.        cout << number << " 是負整數。\n";    //  顯示訊息
16.     }                                       //else 區塊結束點
17.     system("PAUSE");
18.     return 0;
19.  }
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入一個整數：25 Enter
25 是正整數。
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入一個整數：-25 Enter
-25 是負整數。
```

上面程式的 if 區塊與 else 區塊都只包含一個敘述，所以可以省略 if 或 else 的起始與結束大括號。

程式 **4-05**：判斷奇、偶數

```
1.   //儲存檔名：d:\C++04\C0405.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   int main(int argc, char** argv)
6.   {
7.      cout << "請輸入一整數：";
8.      int number;                             //宣告整數變數
9.      cin >> number;                          //輸入 number 資料
10.     if (number % 2 == 0)                    //若 number/2 餘數為 0
11.        cout << number << " 是偶數\n";       //則顯示訊息 1
12.     else                                    //否則
13.        cout << number << " 是奇數\n";       //則顯示訊息 2
14.     system("PAUSE");
15.     return 0;
16.  }
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入一整數：125 Enter
125 是奇數
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入一整數：250 Enter
250 是偶數
```

程式 **4-06**：判斷上限溢位

```
1.   //儲存檔名：d:\C++04\C0406.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   int main(int argc, char** argv)
6.   {
7.      short num1, num2, sum;
8.
9.      cout << "請輸入短整數 1：";
10.     cin >> num1;                                  //輸入短整數並存入 num1
11.     cout << "請輸入短整數 2：";
12.     cin >> num2;                                  //輸入短整數並存入 num2
13.     if ((num1 + num2) > 32767) {                  //若 num1+num2 > 32767
14.        cout << num1 << " + " << num2 << " = 上限溢位\n"; //顯示錯誤訊息
15.     } else {                                      //若 num1+num2<=32767
16.        sum = num1 + num2;
17.        cout << num1 << " + " << num2 << " = " << sum << endl;//顯示運算值
18.     }
19.     system("PAUSE");
20.     return 0;
21.  }
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入短整數 1：32700 Enter
請輸入短整數 2：68 Enter
32700 + 68 = 上限溢位
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入短整數 1：32700 Enter
請輸入短整數 2：67 Enter
32700 + 67 = 32767
```

4.2.5 if-else if 敘述

```
if(條件運算式 1)
    敘述 1;
else if(條件運算式 2)
    敘述 2;
else
    敘述 n;
```

- **if-else if 敘述（if-else if statement）**是多選一的結構。若 if 條件運算式的值為 1（true）則執行 if 區塊的敘述後結束 if，若 if 條件運算式的值為 0（false）則再比較 else if 的條件運算式，若 else if 條件運算式的值為 1（true）則執行該 else if 區塊的敘述後結束 if，若所有的條件運算式的值皆為 0（false）則執行 else 區塊的敘述後結束 if。

- **if-else if 結構**就類似單選的選擇題。假設有 4 個選擇的單選題，若答案 1 正確則選 1，若答案 2 正確則選 2，若答案 3 正確則選 3，若 1、2、3 皆錯則選 4 以上皆非。

- **else 敘述**就等於單選題的以上皆非。可是有些題目並沒有以上皆非的答案，而且答案 1、2、3、4 皆錯怎麼辦？這時候就應該填 0。所以在 if-else if 結構中，若省略 else 敘述而所有條件運算式值皆為 0（false），則不執行任何敘述而結束 if。

下面範例是利用 if-else if 判斷變數值的正、負、或零，若變數 num>0 則 plus 加 1，若變數 num<0 則 minus 加 1，以上皆非則 zero 加 1。

```
if (num > 0)                                    //若 num > 0
   plus += 1;                                   //則 plus = plus + 1
else if (num < 0)                               //若 num < 0
   minus += 1;                                  //則 minus = minus + 1
else                                            //以上皆非
   zero += 1;                                   //則 zero = zero + 1
```

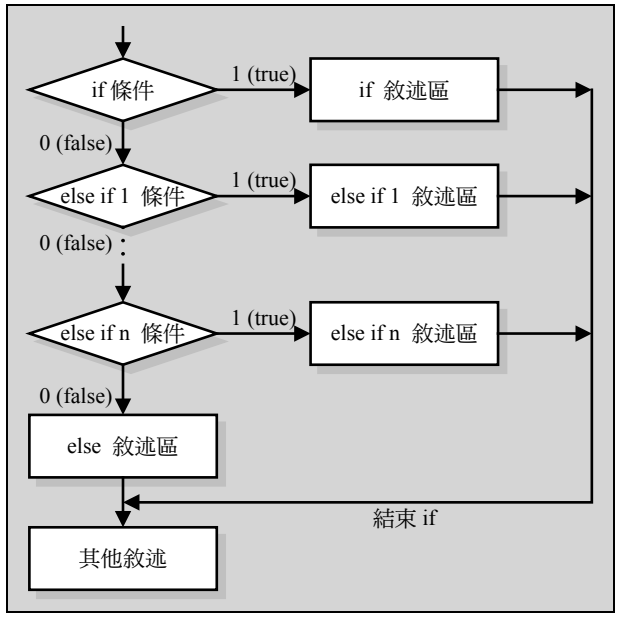
```
if(條件運算式 1)
{
      敘述區 1;
} else if(條件運算式 2) {
      敘述區 2;
} else {
      敘述區 n;
}
```

- **if-else if 區塊（if-else if block）**的工作方式與 if-else if 敘述的工作方式相似。若 if 條件運算式的值為 1（true）則執行敘述區 1 所有的敘述後結束 if，若 if 條件運算式的值為 0（false）則繼續比較其餘的 else if 的條件運算式，若其中一個 else if 的條件運算式值為 1（true）則執行該 else if 敘述區 2 所有的敘述後結束 if，若所有的條件運算式的值皆為 0（false）則執行 else 敘述區 n 的敘述後結束 if。
- 若省略 else 敘述而所有條件運算式值皆為 0（false），則不執行任何敘述而結束 if。
- **{ } 大括號**指示 if 區塊的起始與結束位置，若省略大括號則變成 if 的單行敘述。
- **敘述區**中可包含一個或多個 C++ 敘述。

下面範例是在 short 的加法運算前，利用 if-else if 判斷加法運算是否會產生 short 的上限或下限溢位，若溢位則顯示上限或下限溢位，若沒有溢位則進行 short 的加法運算。注意：若輸入值大於 32767，則會被當成負數，可能不會造成溢位。

```
short num1, num2;
if ((num1 + num2) > 32767) {                    //若 num1+num2>32767
    cout << num1 << " + " << num2;
    cout << " = 上限溢位";                       //  顯示錯誤訊息
} else if ((num1 + num2) < -32768) {            //若 num1+num2<-32768
    cout << num1 << " + (" << num2;
    cout << ") = 下限溢位";                      //  顯示錯誤訊息
```

```
} else {                                          //以上皆非
   sum = num1 + num2;
   cout << num1 << " + " << num2;
   cout << " = " << sum;                          //顯示運算值
}
```



程式 **4-07**：判斷大於、小於、或等於零

```
1.   //儲存檔名：d:\C++04\C0407.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   int main(int argc, char** argv)
6.   {
7.      int number;
8.
9.      cout << "請輸入一個整數：";
10.     cin >> number;                             //輸入字元並存入 number
11.     if(number > 0) {                           //若 number >= 0
12.        cout << number << " 大於 0\n";          //  顯示訊息
13.     }                                          //if 區塊結束點
14.     else if(number < 0) {                      //若 number < 0
15.        cout << number << " 小於 0\n";          //  顯示訊息
16.     }                                          //else if 區塊結束點
17.     else {                                     //以上皆非
18.        cout << number << " 等於 0\n";          //  顯示訊息
```

```
19.     }                                          //else 區塊結束點
20.     system("PAUSE");
21.     return 0;
22.  }
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入一個整數：25 Enter
25 大於 0
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入一個整數：-25 Enter
-25 小於 0
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入一個整數：0 Enter
0 等於 0
```

上面程式的 if 區塊、else if 區塊與 else 區塊都只包含一個敘述，所以可以省略 if、else if 或 else 的起始與結束大括號。

程式 **4-08**：判斷大寫、小寫、數字、與符號鍵

```
1.   //儲存檔名：d:\C++04\C0408.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   int main(int argc, char** argv)
6.   {
7.      char letter;
8.      cout << "請按打字鍵，再按 Enter：";
9.      cin >> letter;                              //輸入字元並存入 letter
10.     if (letter >= 'A' && letter <= 'Z')         //若'A'<=letter<='Y'則
11.        cout << letter << " 為大寫鍵\n";         //  輸出字串並結束 if
12.     else if (letter >= 'a' && letter <= 'z') //若'a'<=letter<='y'則
13.        cout << letter << " 為小寫鍵\n";         //  輸出字串並結束 if
14.     else if (letter >= '0' && letter <= '9') //若'0'<=letter<='9'則
15.        cout << letter << " 為數字鍵\n";         //  輸出字串並結束 if
16.     else                                        //以上皆非則
17.        cout << letter << " 為符號鍵\n";         //  輸出字串並結束 if
18.     system("PAUSE");
19.     return 0;
20.  }
```

程式輸出：粗體字表示鍵盤輸入

```
請按打字鍵，再按 Enter：C Enter
C 為大寫鍵
```

程式輸出：粗體字表示鍵盤輸入

```
請按打字鍵，再按 Enter：+ Enter
+ 為符號鍵
```

程式輸出：粗體字表示鍵盤輸入

```
請按打字鍵，再按 Enter：7 Enter
7 為數字鍵
```

程式輸出：粗體字表示鍵盤輸入

```
請按打字鍵，再按 Enter：a Enter
a 為小寫鍵
```

程式 **4-09**：區分分數等級

```
1.  //儲存檔名：d:\C++04\C0409.cpp
2.  #include <iostream>
3.  using namespace std;
4.
5.  int main(int argc, char** argv)
6.  {
7.     short number;
8.     cout << "請輸入成績 (0 - 100)：";
9.     cin >> number;                              //輸入數值並存入 number
10.    if (number >= 90 && number <= 100)          //若 90<=number<=100 則
11.       cout << "成績甲等\n";                    //  輸出字串並結束 if
12.    else if (number >= 80 && number <= 89)      //若 80<=number<=89 則
13.       cout << "成績乙等\n";                    //  輸出字串並結束 if
14.    else if (number >= 70 && number <= 79)      //若 70<=number<=79 則
15.       cout << "成績丙等\n";                    //  輸出字串並結束 if
16.    else if (number >= 60 && number <= 69)      //若 60<=number<=69 則
17.       cout << "成績丁等\n";                    //  輸出字串並結束 if
18.    else if (number <= 59)                      //若 number<=59 則
19.       cout << "成績戊等\n";                    //  輸出字串並結束 if
20.    system("PAUSE");
21.    return 0;
22. }
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入分數 (0 - 100)：95 Enter
成績甲等
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入分數 (0 - 100)：89 Enter
成績乙等
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入分數 (0 - 100)：78 Enter
成績丙等
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入分數 (0 - 100)：57 Enter
成績戊等
```

程式 **4-10**：判斷上限與下限溢位

```
1.   //儲存檔名：d:\C++04\C0410.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   int main(int argc, char** argv)
6.   {
7.      short num1, num2, sum;
8.
9.      cout << "請輸入短整數 1：";
10.     cin >> num1;                                  //輸入字元並存入 num1
11.     cout << "請輸入短整數 2：";
12.     cin >> num2;                                  //輸入字元並存入 num2
13.     if ((num1 + num2) > 32767) {                  //若 num1+num2>32767
14.        cout << num1 << " + " << num2;
15.        cout << " = 上限溢位\n";                    //  顯示錯誤訊息
16.     } else if ((num1 + num2) < -32768) {          //若 num1+num2<-32768
17.        cout << num1 << " + (" << num2;
18.        cout << ") = 下限溢位\n";                   //  顯示錯誤訊息
19.     } else {                                      //以上皆非
20.        sum = num1 + num2;
21.        cout << num1 << " + " << num2;
22.        cout << " = " << sum << endl;              //  顯示運算值
23.     }
24.     system("PAUSE");
```

```
25.     return 0;
26.  }
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入短整數 1：-32767 Enter
請輸入短整數 2：-2 Enter
-32767 + (-2) = 下限溢位
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入短整數 1：32767 Enter
請輸入短整數 2：2 Enter
32767 + 2 = 上限溢位
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入短整數 1：30000 Enter
請輸入短整數 2：2000 Enter
30000 + 2000 = 32000
```

4.2.6 巢狀 if 敘述

巢狀 if 敘述（nested if statements）是一個 if 敘述（或區塊）包含於另一個 if 敘述（或區塊）之中，簡單的說就是大 if 包小 if。

下面範例是巢狀 if 結構，首先外層 if 先判斷 number 是否大於 0，若大於 0 再進行內層 if 判斷 number 是否小於 9，二個條件皆成立時顯示 “YES” 字串。若外層 if 判斷 number 小於或等於 0，則不再執行內層 if 而結束外層 if 敘述。實際上，它相當於 if(number > 0 && number < 9) 的關係運算式。

```
if (number > 0)                                    //若 number>0
{
   if (number < 9)                                 //若 number<9
      cout << "Yes";                                //則顯示"Yes"
}
```

下面範例先判斷 num 是否被 3 整除，若被 3 整除再判斷 num 是否被 5 整除，二個條件皆成立時顯示 “是 3 和 5 的倍數” 字串。可是當不能被 3

整除時，則執行 else 下的敘述顯示 “不是 3 的倍數” 字串。例如當 num=35 時，num 除以 3 不等於 0，所以執行 else 敘述，顯示 “35 不是 3 的倍數”。

```
if (num % 3 == 0)                                    //num 是否為 3 的倍數
{                                                    //num 是 3 的倍數
   if (num % 5 == 0)                                 //num 是否為 5 的倍數
      cout << num << "是 3 和 5 的倍數";             //num 是 3 和 5 的倍數
}
else                                                 //num 不是 3 的倍數
   cout << num << "不是 3 的倍數";                   //顯示字串 2
```

若將上面巢狀 if-else 區塊範例中的大括號省略如下：

```
if (num % 3 == 0)                                    //若 num 除以 3 等於 0
   if (num % 5 == 0)                                 //若 num 除以 5 等於 0
      cout << num << "是 3 和 5 的倍數";             //則輸出數值與字串一
else                                                 //若 num 除以 3 不等於 0
   cout << num << "不是 3 的倍數";                   //則輸出數值與字串二
```

則 C++ 編譯器會以下面區塊型式來編譯上面的巢狀 if，雖然程式語法沒有錯誤，編譯過程也沒有錯誤，但程式邏輯錯誤，所以執行結果當然有錯。例如當 num=36 時，num 除以 3 等於 0，但 num 除以 5 不等於 0，所以執行內層的 else 敘述，顯示 “36 不是 3 的倍數”，而造成錯誤。

```
if (num % 3 == 0) {                                  //若 num 除以 3 等於 0
   if (num % 5 == 0)                                 //若 num 除以 5 等於 0
      cout << num << "是 3 和 5 的倍數";             //則輸出數值與字串一
   else                                              //若 num 除以 5 不等於 0
      cout << num << "不是 3 的倍數";                //則輸出數值與字串二
}
```

下面範例是巢狀 if 結構來判斷 year 是否為閏年年份，判斷順序如下：

1. 不是 4 的倍數，則不是閏年，例如 2003 年。
2. 是 100 和 400 的倍數，則是閏年，例如 2000 年。
3. 是 100 但不是 400 的倍數，則不是閏年，例如 1000 年。
4. 其他是 4 的倍數，則都是閏年，例如 2004 年。

```
int year;
if (year % 4 != 0)                                   //若 year 不是 4 的倍數
   cout << year << "不是閏年";                       //則顯示 year 不是閏年
```

```
else if (year % 100 == 0)                          //若 year 是 100 的倍數
{
   if (year % 400 == 0)                            //且 year 不是 400 的倍數
      cout << year << "是閏年";                    //則顯示 year 是閏年
   else                                            //否
      cout << year << "不是閏年";                  //則顯示 year 不是閏年
}
else                                               //否
   cout << year << "是閏年";                       //則顯示 year 是閏年
```

上面範例的巢狀 if 結構也可簡化成非巢狀 if 結構如下，但是判斷條件由小範圍到大範圍，例如先判斷 400 倍數、再判斷 100 倍數、再判斷 4 的倍數。

```
      int year;
      if(year %400 == 0) {                         //若 year 是 400 的倍數
         cout << year << "是閏年";                 //則顯示 year 是閏年
      } else if (year %100 ==0){                   //若 year 是 100 的倍數
         cout << year << "不是閏年";               //則顯示 year 不是閏年
      } else if (year %4 == 0) {                   //若 year 是 4 的倍數
         cout << year << "是閏年";                 //則顯示 year 是閏年
      } else {                                     //若 year 不是 4 的倍數
         cout << year << "不是閏年";               //則顯示 year 不是閏年
      }
```

如果在搭配關係運算符號與邏輯運算符號，還可將上面的範例簡化如程式 4-11。

程式 **4-11**：判斷閏年年份

```
1.   //儲存檔名：d:\C++04\C0411.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   int main(int argc, char** argv)
6.   {
7.      int year;
8.      cout << "請輸入西元年份：";
9.      cin >> year;
10.     if((year%400 == 0) || ((year%4 == 0) && (year%100 != 0)))
11.        cout << year << "是閏年\n";              //則顯示 year 是閏年
12.     else                                        //否
13.        cout << year << "不是閏年\n";            //則顯示 year 不是閏年
14.     system("PAUSE");
15.     return 0;
16.  }
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入西元年份：2018 Enter
2018 不是閏年
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入西元年份：2020 Enter
2020 是閏年
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入西元年份：2000 Enter
2000 是閏年
```

程式輸出：粗體字表示鍵盤輸入

```
請輸入西元年份：1000 Enter
1000 不是閏年
```

4.2.7 switch 敘述

```
switch (條件運算式)
{
    case 數值 1:
        敘述區 1;
        break;   //中斷 switch
    case 數值 2:
        敘述區 2;
        break;   //中斷 switch
    default:
        敘述區 n;
}
```

- **switch** 是多重分支的條件判斷敘述。switch 敘述的功能類似 if-else if 敘述，不同的是 switch 只有一個判斷條件，而 if-else if 可以有許多不同的判斷條件。

- **條件運算式**相當於 switch 敘述的條件，此運算式的值必須是整數（short、int、或 long）或字元（char），而不是布林值（bool）。
- **數值 1、數值 2、...、數值 n** 是與運算式的比較值。當運算式的值等於數值 1 則執行敘述區 1 的敘述，當運算式的值等於數值 2 則執行敘述區 2 的敘述，當運算式的值皆不等於數值 1、數值 2... 時則執行 default 敘述區 n 的敘述。
- **敘述區**中可包含一個或多個 C++ 敘述。
- **break** 用來中斷 switch 敘述。因為執行完任何 case 敘述區後並不自動結束 switch 區塊，所以必須使用 break 敘述結束 switch 區塊，否則他會繼續向下執行其他 case 敘述區與 default 敘述區。
- **{ }** 大括號表示 switch 區塊的起始與結束位置，若只有一個敘述則可簡化如下。

```
switch(條件運算式)
{
    case 數值 1: 敘述 1;
       break;   // 中斷 switch
    case 數值 2: 敘述 2;
       break;   // 中斷 switch
    default: 敘述 n;
}
```

下面範例是以 switch 敘述來比較字元變數 letter 與字元資料 'Y' 或 'N'。當 letter 的值等於 'Y' 則執行 case 'Y' 的敘述，直到執行 break 敘述才結束 switch 區塊。當 letter 的值等於 'N' 則執行 case 'N' 下的敘述，直到執行 break 敘述才結束 switch 區塊。若 letter 的值不等於 'Y' 或 'N' 則執行 default 敘述後結束 switch。

```
switch(letter)                          //條件 = letter
{
   case 'Y':                            //若 letter = 'Y'
      cout << "Yes";                    //顯示 "Yes"
```

```
      break;                              //中斷 switch
   case 'N':                              //若 letter = 'N'
      cout << "No";                       //顯示 "No"
      break;                              //中斷 switch
   default:                               //若 letter != 'Y'或'N'
      cout << "Unexpected";               //顯示 "Unexpected"
}
```

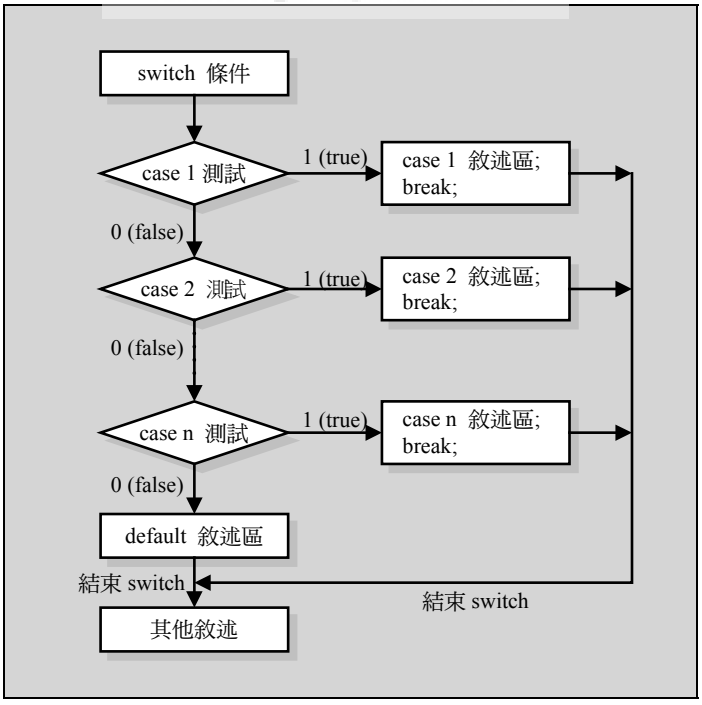
上面範例可被簡化如下：

```
switch(letter) {                          //條件 = letter
   case 'Y': cout << "Yes";               //letter='Y'顯示 "Yes"
      break;                              //中斷 switch
   case 'N': cout << "No";                //letter='N'則顯示 "No"
      break;                              //中斷 switch
   default:                               //若 letter != 'Y'或'N'
      cout << "Unexpected";               //顯示 "Unexpected"
}
```

或許讀者會問，若寫一個類似 if (letter==‘Y’ || letter==‘y’) 的 case 的敘述，是不是使用 case ‘Y’ || ‘y’: ？當然不是，而是使用連續的二個 case ‘Y’: case ‘y’: 表示若 letter 等於大寫 ‘Y’ 或小寫 ‘y’ 都執行相同的敘述。同理，使用連續的二個 case ‘N’: case ‘n’: 表示若 letter 等於大寫 ‘N’ 或小寫 ‘n’ 都執行相同的敘述。

下面範例的原理很簡單，當 letter = ‘Y’: 則符合 case ‘Y’: 並執行 case ‘y’: 與 cout << “Yes”; 直到 break; 敘述才中斷 switch，所以不論 letter 是否等於 ‘y’: cout << “Yes”; 敘述都將被執行。而當 letter = ‘y’: 則符合 case ‘y’: 並執行 cout << “Yes”; 直到 break; 敘述才中斷 switch。

```
switch(letter) {                          //條件 = letter
   case 'Y':                              //若 letter = 'Y'
   case 'y':                              //若 letter = 'y'
      cout << "Yes";                       //顯示 "Yes"
      break;                              //中斷 switch
   case 'N':                              //若 letter = 'N'
   case 'n':                              //若 letter = 'n'
      cout << "No";                        //顯示 "No"
      break;                              //中斷 switch
   default:                               //若 letter != 'Y'或'N'
      cout << "Unexpected";               //顯示 "Unexpected"
}
```



程式 **4-12**：按 + - * / 鍵執行 + - * / 運算

```
1.   //儲存檔名：d:\C++04\C0412.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   int main(int argc, char** argv)
6.   {
7.      char letter;
8.      int num1 = 75, num2 = 15;
9.      cout << "num1 = 75, num2 = 15 \n";
10.     cout << "請選擇 +,-,*,/ :";
11.     cin >> letter;                          //輸入字元並存入 letter
12.     switch (letter)
13.     {
14.        case '+':                            //若 letter = '+'
15.           cout << "num1 + num2 = " << num1 + num2;
16.           cout << endl; break;              //跳行、並跳出 switch
17.        case '-':                            //若 letter = '-'
18.           cout << "num1 - num2 = " << num1 - num2;
19.           cout << endl; break;              //跳行、並跳出 switch
20.        case '*':                            //若 letter = '*'
21.           cout << "num1 * num2 = " << num1 * num2;
22.           cout << endl; break;              //跳行、並跳出 switch
```

```
23.        case '/':                                  //若 letter = '/'
24.           cout << "num1 / num2 = " << num1 / num2;
25.           cout << endl; break;                    //跳行、並跳出 switch
26.     }
27.     system("PAUSE");
28.     return 0;
29.  }
```

程式輸出：粗體字表示鍵盤輸入

```
num1 = 75, num2 = 15
請選擇 +,-,*,/ ：+ Enter
num1 + num2 = 90
```

程式輸出：粗體字表示鍵盤輸入

```
num1 = 75, num2 = 15
請選擇 +,-,*,/ ：- Enter
num1 - num2 = 60
```

程式輸出：粗體字表示鍵盤輸入

```
num1 = 75, num2 = 15
請選擇 +,-,*,/ ：* Enter
num1 * num2 = 1125
```

程式輸出：粗體字表示鍵盤輸入

```
num1 = 75, num2 = 15
請選擇 +,-,*,/ ：/ Enter
num1 / num2 = 5
```

程式 **4-13**：建立功能表

```
1.   //儲存檔名：d:\C++04\C0413.cpp
2.   #include <iostream>
3.
4.   using namespace std;
5.
6.   int main(int argc, char** argv)
7.   {
8.      char inChar;
9.      cout << "a. 新增資料\tb. 插入資料\tc. 刪除資料\t 其他. 結束程式：";
10.     cin >> inChar;                                //inChar=輸入字元
11.     switch (inChar)
12.     {
13.        case 'A':                                  //若 inChar 為 A 字元
14.        case 'a':                                  //或 inChar 為 a 字元
```

```
15            cout << "新增資料\n";
16            break;
17         case 'B':                                  //若 inChar 為 B 字元
18         case 'b':                                  //或 inChar 為 b 字元
19            cout << "插入資料\n";
20            break;
21         case 'C':                                  //若 inChar 為 C 字元
22         case 'c':                                  //或 inChar 為 c 字元
23            cout << "刪除資料\n";
24            break;
25         default:                                   //inChar 為其他字元
26            cout << "結束程式\n";
27      }
28      system("PAUSE");
29      return 0;
30   }
```

程式輸出：粗體字表示鍵盤輸入

```
a. 新增資料    b. 插入資料    c. 刪除資料    其他. 結束程式：a Enter
新增資料
```

程式輸出：粗體字表示鍵盤輸入

```
a. 新增資料    b. 插入資料    c. 刪除資料    其他. 結束程式：b Enter
插入資料
```

程式輸出：粗體字表示鍵盤輸入

```
a. 新增資料    b. 插入資料    c. 刪除資料    其他. 結束程式：c Enter
刪除資料
```

程式輸出：粗體字表示鍵盤輸入

```
a. 新增資料    b. 插入資料    c. 刪除資料    其他. 結束程式：z Enter
結束程式
```