國立屏東大學 資訊工程學系 C程式設計入門教材 ====== - #7 選擇 ====== 早在上一世紀60年代，學者們就已經證明了從簡單到複雜的各式應用問題都可以由順序(Sequence)、選擇(Selection)與迴圈(Loop)三種基本結構所組成的程式加以解決[(Corrado Böhm and Giuseppe Jacopini,"Flow diagrams, Turing machines and languages with only two formation rules." Communications of the ACM, Vol.9, pp. 366–371. 1966.)]。截至目前為止，本書所有範例程式的執行都是從main()函式開始，一行、一行的執行，直到main()結束為止。這種"一條腸子通到底"的線性執行動線就是順序結構的意思---簡單有效、但還不夠，我們還需要學習其它兩種結構才能夠解決所有的問題，這就是本章與下一章的目的 --- 分別為讀者介紹選擇與迴圈結構。 C++語言支援讓程式依據特定條件執行不同動線的選擇敘述(Selection Statement)。我們將可以為程式的執行定義特定條件，讓程式視不同情況執行不同的程式碼。本章將先介紹與定義條件相關的邏輯運算式(Logical Expression)、以及兩個常用的條件敘述 — if及switch。 隨著我們開始使用條件敘述，程式設計的複雜性也隨之提升，原本所使用的IPO模型已經不敷所需；因此，本章也將介紹第二種程式設計的思維模型工具 — 流程圖(flowchart)。透過此一工具，我們將能更容易地設計出結構與動線都更為複雜的程式。 ---- ===== - 邏輯運算式(Logical Expression) ===== 所謂的邏輯運算式(Logical Expression)亦稱為布林運算式(Boolean Expression)、由著名數學家[[wp>George Boole]]所提出，是當代電腦科學的重要基礎。邏輯運算式的運算結果稱為布林值(Boolean Value)、只能是true與false兩種可能，分別代表某種情況、情境或是狀態、條件的「正確」與「錯誤」、「成立」與「不成立」、「真」與「偽」等「正面的」或「負面的」兩種可能。我們在[[cppbook:ch-varconsttype#布林型態 4.3.4 布林型態]]已經介紹過、C語言所支援的布林型態為bool、其數值true與false亦可以非0(預設為1)以及0的整數值表達[1)]，並且在6.4.4 布林型態的數值介紹過其相關的輸入與輸出格式設定相關函式與操控子，請讀者自行加以回顧。

邏輯運算式的運算元(Operand)可以是數值、常數、變數、函式呼叫、甚至是其它的運算式；至於在運算子(Operator)方面可再區分為關係運算子(Relational Operator)、相等運算子(Equality Operator)、不相等運算子(Inequality Operator)與邏輯運算子(Logical Operator)等多項分類，我們將在本節中分別加以介紹。

0.0.1 關係運算子

關係運算子(Relational Operator)是一個左關聯的二元運算子，用以判斷兩個運算元(數值、函式、變數或運算式)之間的關係，其可能的關係有：大於、小於、等於、或不等於、C語言提供以下的關係運算子，如table 1、

符號	範例	意義
>	a > b	a 是否大於 b
<	a < b	a 是否小於 b
>=	a >= b	a 是否大於或等於 b
<=	a <= b	a 是否小於或等於 b

Tab. 1: Relational Operators

雖然我們已經提過、C語言將數值0視為false、並將其它所有非0的數值視為true、但關係運算子會以數值0代表運算結果為false、以數值1代表true、還要注意關係運算子較算術運算子的優先順序低，所以像是 x + y < i - j 等同於 (x + y) < (i - j)、在C語言中、x < y < z等同於 (x < y) < z、因為關係運算子為左關聯。假設x=1, y=3, z=5、 x<y<z ⇒ (x < y) <z ⇒ 1 < z ⇒ 1 、

0.0.2 相等運算子/Equality Operator

相等運算子是一個binary運算子，用以判斷兩個運算元(數值、函式、變數或運算式)之值是否相等、C語言提供以下的關係運算子，如table 2、

符號	範例	意義
==	a == b	a 是否等於 b
!=	a != b	a 是否不等於 b

Tab. 2: Equality Operators

同樣地，相等運算子會以數值0代表運算結果為false□以數值1代表true□還要注意相等運算子與關係運算子一樣，其優先順序都較算術運算子來得低。

<note important>是 == ， 不是 = 。 千萬不要將比較兩數是否相等的==寫成=，這實在是一個常常會遇到的錯誤!建議您以後如果遇到程式執行結果錯誤，但找不出任何問題時，試試檢查一下所有的 = 與 ==，有很高的機會可以改正您的程式□</note>

0.0.3 邏輯運算子/Logical Operator

邏輯運算子共有以下三種，如table 3□

符號	意義	unary or binary
!	NOT	unary
&&	AND	binary
||	OR	binary

Tab. 3: Logical Operators

其運算結果請參考table 4的真值表：

X	Y	NOT X	X AND Y	X OR Y
0	0	1	0	0
0	1		0	1
1	0	0	0	1
1	1		1	1

Tab. 4: Truth Table

假設變數score代表c語言的修課成績，以下的邏輯運算即為檢查成績是否介於0~100：

```
((score >= 0) && (score <=100))
```

0.0.4 優先順序(Precedence of Operators)

我們將目前為止介紹過的運算子之優先順序整理如table 5(表中是以優先權高至低依序列示)：

運算子	符號
一元運算子	+(正)、-(負)□++□--□!(NOT)
算術運算子(乘除)	*□/□%
算術運算子(加減)	+□-
關係運算子	>=□<=□>□<
相等運算子	==□!=

運算子	符號
邏輯運算子	&&、||
條件運算子	?:
指定運算子	=、*=、/=、%=、+=、-=

Tab. 5: 各運算子的優先順序(由高至低)

0.1 IF敘述

當我們在程式寫作時，某些功能可能是要視情況來決定是否要加以執行的。在C語言中，提供一個if敘述，可以做到依特定條件成立與否，來決定該執行哪些程式碼。if的語法如下：

```
if ( expression ) statement
```

當expression成立時，也就是其布林值為true、或是其數值不為0時，才會執行後面所接的statement。例如：

```
if (score >= 60) printf("You are pass!\n");
```

上面這行程式的意思是，如果score>=60則印出“You are pass!”。當然，若條件不成立時，後面所接的printf()是不會被執行的。如果條件成立時，想要進行的處理需要一行以上的程式碼該怎麼辦呢？請參考下面的語法：

```
if ( expression ) { statements }
```

我們可以在if敘述後，以一組大括號來將要執行的程式碼包裹起來。這種被包裹起來的程式碼又稱為複合敘述(compound statment)。請參考下面的例子：

```
if (score >= 60)
{
   printf("Your score is %d\n", score);
   printf("You are pass!\n");
}
```

如果我們想判斷的條件不只一個，那又該怎麼辦呢？其實if敘述也是敘述，所以在compound statement中當然可以含有if敘述，請參考下面的程式：

```
if (score >= 60)
{
   printf("Your score is %d\n", score);
   printf("You are pass!\n");
```

```
    if(score >= 90)
    {
        printf("You are very excellent!\n");
    }
}
```

<note important>良好的縮排(indent)習慣，可以為您的程式碼帶來容易閱讀、維護與除錯等好處</note>

下面是另一個例子：

```
if (score >= 0)
{
    if(score <= 100)
    {
        printf("This score %d is valid!\n", score);
    }
}
```

不過這個例子，還可以改寫成：

```
if ((score >= 0)&&(score <=100))
{
    printf("This score %d is valid!\n", score);
}
```

<note important> 我曾經看過有人把程式這樣寫，雖然我可以瞭解其用意，但這個程式是錯誤的！因為relational operator是左關聯0<=score<=100 → (0<=score)<=100假設score為50 → 1 <=100 → 1 → true但如果score是110，這個運算式的結果還是true

```
if ( 0 <= score <= 100)
{
    printf("This score %d is valid!\n", score);
}
```

</note>
延續上面的例子，若是想要在score超出範圍時，印出錯誤訊息，那又該如何設計呢？請參考下面的程式：

```
if ((score >= 0)&&(score <=100))
{
    printf("This score %d is valid!\n", score);
}

if((score<0) || (score>100))
{
    printf("Error! The score %d is out of range!\n", score);
```

```
}
```

在這段程式碼中的兩個if敘述，其實是互斥的，也就是當第一個if的條件成立時，第二個if的條件絕不會成立，反之亦然。這種情況可以利用下面的語法，把兩個if敘述整合成一個：

```
if ( expression ) statement else statement


if ( expression ) { statements } else { statements }
```

else保留字可以再指定一個敘述或是複合敘述，來表明當if條件不成立時，所欲進行的處理。請參考下面的例子：

```
if ((score < 0) || (score >100))
{
   printf("Error! The score %d is out of range!\n", score);
}
else
{
   printf("This score %d is valid!\n", score);
}
```

再一次考慮到if敘述也是一種敘述，在else的後面，我們也可以再接一個if敘述，例如下面的例子：

```
if ((score < 0) || (score >100))
{
   printf("Error! The score %d is out of range!\n", score);
}
else
{
   if(score>=60)
   {
      printf("You are pass!\n");
   }
}
```

類似的結構延伸，下面的程式碼也是正確的：

```
if ((score < 0) || (score >100))
{
   printf("Error! The score %d is out of range!\n", score);
}
else
```

```
{
   if(score>=60)
   {
      printf("You are pass!\n");
   }
   else
   {
      printf("You are fail!\n");
   }
}
```

上述的程式碼，也可以利用if敘述及else保留字後面可以接一個敘述(只有一個敘述時，大括號可以省略)，我們可以將部份的大括號去掉，請參考下面的程式碼:

```
if ((score < 0) || (score >100))
{
   printf("Error! The score %d is out of range!\n", score);
}
else if(score>=60)
{
   printf("You are pass!\n");
}
else
{
   printf("You are fail!\n");
}
```

0.1.1 IPO程式設計範例

我們以下面這一個例子做為本小節的結尾。

假設在一個售貨系統中，有一個代表銷售總金額的變數total當total大於5000時，我們要為客戶打九五折。

現在我們來說明如何完成這個動作：首先是我們要如何判斷total是否大於5000? 這可以用下面這個敘述來完成：

```
if( total > 5000 )
```

這一個邏輯運算式會傳回true或是false的值。接下來的問題是，如果傳回的值是true則我們應該將total的值乘上0.95：

```
total = total * 0.95;
```

讓我們以IPO分析如下：

- Input Section:
  - 取得total
- Process Section:
  - 判斷total是否大於5000
  - 若是則total*=0.95
- Output Section:
  - 輸出total的值

考慮到打95折是以*0.95方式進行，則必須注意其結果為浮點數，現在讓我們把Declaration Section也加入IPO分析，並增加更多細節：

- Declaration Section:
  - float total;
- Input Section:
  - printf(“Please input the total:”);
  - scanf(“%f”, &total);
- Process Section:
  - if(total > 5000)
    - total*=0.95;
- Output Section:
  - printf(“Total=%f”, total);

完整的程式碼，很容易就可以依上面的IPO分析來完成：

h total.c

```
#include <stdio.h>

int main()
{
   // Declaration Section:
   float total;

   // Input Section:
   printf("Please input the total:");
   scanf("%f", &total);

   // Process Section:
   if(total > 5000 )
   {
      total*=0.95;
   }

   // Output Section:
   printf("Total=%f", total);
}
```

0.2 switch 敘述

請參考下面的資料表：

deptID	系所名稱	分機號碼
1	資訊工程系	21201
2	電腦與通訊系	21303
3	電腦與多媒體系	21701

以下的程式讓使用者輸入系所代碼後，依系所印出其分機號碼:

```
int deptID;

scanf("%1d", &deptID);

if( deptID == 1 )
{
   printf("Computer Science and Information Engineering\n");
   printf("Phone: (08)7238700 ext.21201.\n");
}
else if( deptID == 2 )
{
   printf("Computer and Communications\n");
   printf("Phone: (08)7238700 ext.21303.\n");
}
else if( dept == 3 )
{
   printf("Computer and Multimedia\n");
   printf("Phone: (08)7238700 ext.21701.\n");
}
else
{
   printf("The value of deptID %d is invalid!\n", deptID);
}
```

上面這個程式具備「給定一個數值(或運算式)，依其值決定該執行的程式碼」的程式結構。同樣的結構□C語言提供另一個選擇□switch敘述，其語法如下：

```
switch (expression)
{
    case constant-expression□ statements
    …
    case constant-expression□ statements
        default: statements
}
```

其執行流程是：先判斷接在switch後面的運算式的值，依照其值去執行對應的case後的敘述群。我們將其

語法各個部份分別說明如下：

- switch(expression)
  - 以switch開頭，緊接一個括號與其內的expression。這裡的expression其運算結果必須為整數或字元(預設為int型態，若為char型態也會被自動轉換為int)。其它的浮點數或字串等都不被接受。
- {…} → 最後以一組大括號與其內的敘述定義不同情況下的處理方法。可以接受的敘述包含以下兩種:
  - case constant-expression : statements
    - case →以case開頭
    - constant-expression → 所謂的constant expression即為運算式，但其運算元僅接受constant值，例如數值1, 2, 3等整數值、或'A', 'B', 'C'等字元型態(會被自動轉換成整數)或運算式 1+1, 1+2, 2+3等皆可；但不允許含有變數，例如x+y是不被接受的。
    - statements → 一行或一行以上的敘述，此部份為選擇性。當前述的expression的運算結果與這裡的constantexpression運算結果相同時，則從此處開始往下繼續執行程式碼。
  - default : statements → 若前面的各個case的constant expression都不等同於switch的expression運算結果，那麼程式就跳過前述的各個case直接到default這裡執行剩下的敘述。注意default是選擇性的，也可以不寫。

現在，讓我們使用switch敘述將前述的例子重寫如下：

```
int deptID;

scanf("%1d", &deptID);

switch (deptID )
{
   case 1:
            printf("Computer Science and Information Engineering\n");
            printf("Phone: (08)7238700 ext.21201.\n");
            break;
   case 2:
            printf("Computer and Communications\n");
            printf("Phone: (08)7238700 ext.21303.\n");
            break;
   case 3:
            printf("Computer and Multimedia\n");
            printf("Phone: (08)7238700 ext.21701.\n");
            break;
   default:
            printf("The value of deptID %d is invalid!\n", deptID);
}
```

請將上述程式碼編輯、編譯並加以執行，看看結果為何？注意，在這個程式中，每個case的敘述後都加了一個break敘述，其作用是讓程式的執行跳離其所屬的程式區塊中，一個程式區塊(block)是一組左右對稱的大括號與其內的敘述所組成。假設deptID的輸入值為2，請參考figure1，紅色粗線即為程式執行的動線，當deptID等於2時，程式的執行會跳過一部份，直接到case 2:的地方再加以執行，直到遇到break時，則跳出所屬的程式區塊，意即結束了這個switch敘述的執行。

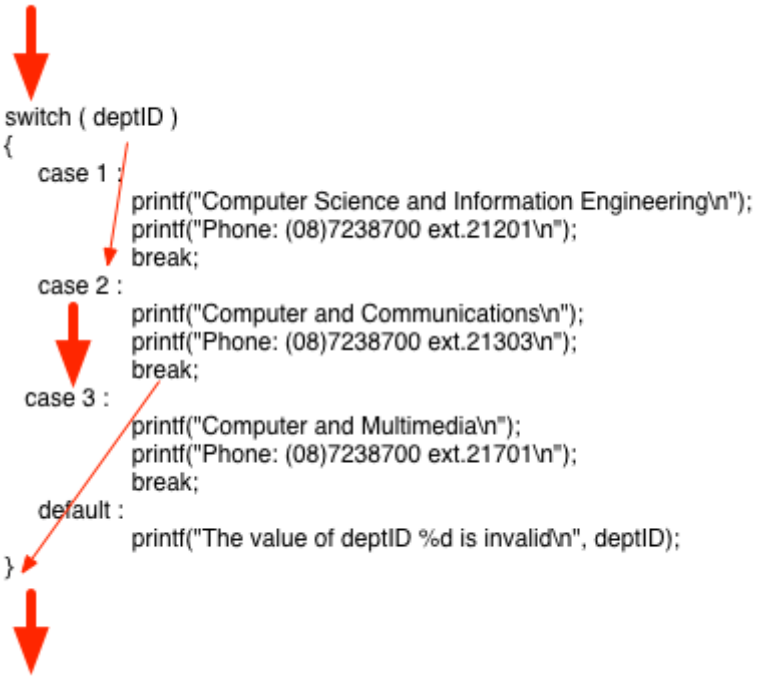

Fig. 1

figure 2則顯示了不使用break的情況下，其程式執行的動線。因為沒有break□所以程式從case 2開始執行後就會一直執行到底，其輸出結果為:

```
2
Computer and Communications
Phone: (08) 7238700 ext.21303.
Computer and Multimedia
Phone: (08) 7238700 ext.21701.
The value of deptID 2 is invalid!
```

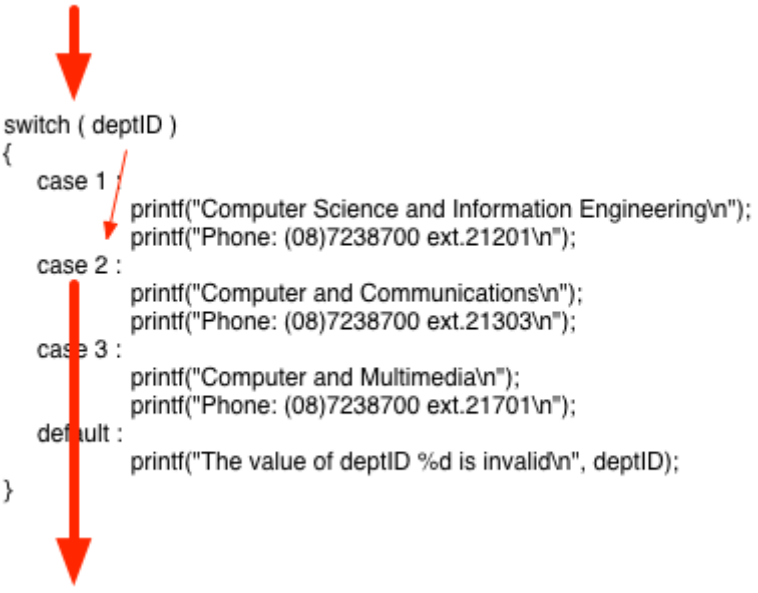

Fig. 2

當程式具有類似處理需求時，使用switch敘述將可以讓程式的架構更為簡單易讀。請再思考看看，還有哪些應用適合使用switch敘述呢？

- 程式提供操作選項，而各選項負責執行不同的功能:

```
switch (choice)
{
   case 'i':
             insert_data();
             break;
   case 'x':
             execute();
             break;
   case 'q':
             exit(0);
             break;
}
```

- 國小學生週一至週五，每天下課時間不同，有時半天、有時整天：

```
switch (weekday)
{
   case 1:
   case 2:
   case 4:
   case 5:
            printf("After school at 4:00pm\n");
            break;
   case 3:
            printf("After school at 12:00am\n");
}
```

- 設計一程式，輸入一整數N□計算並印出1+2+...+N的結果：

```
int n=0, sum=0;
scanf("%d",&n);

switch (n)
{
   case 10: sum+=10;
   case  9: sum+=9;
   case  8: sum+=8;
   case  7: sum+=7;
   case  6: sum+=6;
   case  5: sum+=5;
   case  4: sum+=4;
```

```
    case  3: sum+=3;
    case  2: sum+=2;
    case  1: sum+=1;
}
printf("Sum=%d\n", sum);
```

0.3 條件運算式(Conditional Expression)

C語言還有提供一種特別的運算式，稱為條件運算式(conditional expression)，可依條件決定運算式的傳回值，其語法如下：

```
expression1 ? expression2 : expression3
```

運算式的運算結果expression1的值為true(非0的數值)或false(數值0)而定，當expression1為true時，傳回expression2的值；否則當expression1為false時，傳回expression3的值。事實上，這等同於下面的if敘述：

```
if (expression1)
   result = expression2;
else
   result = expression3;
```

條件式敘述有可能是因為像「如果expression為真則…. 否則….」這樣的述敘，在程式中出現的機會很高的緣故吧！請參考以下的應用：

```
int x=1, y=2, z;
if(x>y)
   z=x;
else
   z=y;
```

上面這段程式碼是令z為x與y兩者中較大的值，如果以條件運算式改寫，則只要寫成

```
z= x>y ? x: y;
```

即可，是不是簡化很多？下面這行程式，假設score為學生成績，則可以簡單地檢查score是否大於100，若超過100則以100分計。

```
score = score > 100 ? 100 : score ;
```

現在讓我們想想下面的運算式在做些什麼呢?

```
x = (x%10)!=0 ? (x-x%10+10) : x;
```

[1)] 使用非0整數與0代表布林數值是承襲自C語言的做法。