流程控制

Horazon

C#程式設計

程式學習地圖 (初步、核心邏輯)

基本結構

變數宣告

循序

選擇 (if, switch)

迴圈 (for, while, do-while)

程式邏輯：循序

循序概念由於太直覺、基礎了，所以沒有特別說明
即為程式「由上而下、一行接一行」的執行過程

```
int a = 5;
int b = 3;
int sum = 5 + 3;
Console.WriteLine(sum);      //8

a = 10;
b = 20
sum = a + b;
Console.WriteLine(sum);      //30
```

程式邏輯：選擇

但是如果只有循序的概念，電腦能夠完成的工作非常侷限
如同在現實世界，我們需要「選擇」的功能來下指令

> 如果店家有賣蘋果的話，買三個蘋果，否則改買五個橘子

所以我們也需要告訴程式，如何「選擇」

程式邏輯：if

這邊先簡化問題

如果店家有賣蘋果的話，買三個蘋果

在這狀況下，就能使用if來完成

```
if(店家賣蘋果)
{
    購買三個蘋果
}
```

注意 大括號 { } 的使用，必須成對出現

程式邏輯：if

實際的程式描述為

```
if(布林邏輯運算式)
{
    邏輯通過時執行之陳述式
}
```

注意小括號 () 與 大括號 {} 的運用

布林邏輯運算式指一段 **會獲得真(true)或假(false)**的運算式

程式邏輯：if

//TODO:流程圖

程式邏輯：if

```
int myApple = 0;
bool sellApple = true;
if(sellApple)
{
    myApple += 3;
}
```

有些人會調整 大括號 { } 位置

```
if(sellApple) {
    myApple += 3;
}
```

當只有一行的時候，可以省略 大括號 { }

```
if(sellApple)
    myApple += 3;
```

程式邏輯：if/else

我們最初的需求是，但後面被省略了

如果店家有賣蘋果的話，買三個蘋果，否則改買五個橘子

利用else，就能做出「否則」的邏輯

```
if(店家賣蘋果)
{
    購買三個蘋果
}
else
{
    購買五個橘子
}
```

程式邏輯：if/else

完整規則如下

```
if (邏輯表達式)
{
    邏輯通過時執行之陳述式
}
else
{
    邏輯不通過時執行之陳述式
}
```

程式邏輯：if/else

//TODO 流程圖

程式邏輯：if/else if/else

有時候需要更多的判斷選擇，在if與else之間可以使用else if

```
if (邏輯表達式 A)
{
    邏輯通過時執行之陳述式
}
else if (邏輯表達式 B)
{
    邏輯不通過A但通過B時執行之陳述式
}
else if (邏輯表達式 C)
{
    邏輯不通過AB但通過C時執行之陳述式
}
else
{
    邏輯不通過ABC時執行之陳述式
}
```

程式邏輯：if/else if/else

//TODO:流程圖

程式邏輯：if/else if/else

視情況可省略最後的 else

此時如同if，若無符合任何邏輯，則沒有陳述式會被執行

```
if (邏輯表達式 A)
{
    邏輯通過時執行之陳述式
}
else if (邏輯表達式 B)
{
    邏輯不通過A但通過B時執行之陳述式
}
else if (邏輯表達式 C)
{
    邏輯不通過AB但通過C時執行之陳述式
}
```

運用與思考

學會了if/else以後，做以下的思考

在[周末]且[沒有下雨]的日子，我會出門玩

使用前面的內容是可以完成的，但 ...

```
if(周末)
{
    if(下雨天)
    {
    }
    else
    {
        出門玩
    }
}
```

邏輯運算子

運用與思考

```
if(周末)
{
    if(下雨天)
    {
    }
    else
    {
        出門玩
    }
}
```

就可以簡化為

```
if(周末 && !下雨天)
{
    出門玩
}
```