新觀念的 Visual Basic.NET 教本

第 5 章 使用資料

5-1 資料的表示法

VB.NET 資料的分類

- 數值資料
- 字串資料
- 日期時間資料
- 布林資料

數值資料－整數

- VB.NET 的整數與數學的整數並沒有什麼不同，如2005、0、+512、-204等均爲正確的寫法，但逗號是不能使用的，如10,000便是不正確的表示法。
- 十六進位數要在前面加上&H，八進位數要在前面加上&O(字母O)，以十進位的100爲例，表示成十六進位是64(註: $(64)_{16} = \mathbf{6}\times16^1 + \mathbf{4}\times16^0 = 100$)，表示成八進位是144(註: $(144)_8 = \mathbf{1}\times8^2 + \mathbf{4}\times8^1 + \mathbf{4}\times8^0 = 100$) 所以：
  - `100`　100的十進位表示法
    `&H64`　100的十六進位表示法
    `&O144` 100的八進位表示法

上機：顯示十六及八進位數

- 功能：寫一程式讓使用者輸入十進位數，然後顯示其十六進位數及八進位數。

1. 建立一Windows應用程式專案，專案名稱定為ch0501。
2. 在表單上佈置五個Label、一個TextBox及一個Button，如圖-1。其中所需設定之屬性如下：

物件	屬性	屬性值
Label1	Text	十進位數:
Label2	Text	十六進位數:
Label3	Text	八進位數:
TextBox1	Text	(清成空白)
Label4	Text	(清成空白)
Label5	Text	(清成空白)
Button1	Text	轉換

上機：顯示十六及八進位數

3. 在 Button1_Click之 中輸入以下程式：

```
Private Sub Button1_Click(...) Handles Button1.Click
    Label4.Text = Hex(Val(TextBox1.Text))
    Label5.Text = Oct(Val(TextBox1.Text))
End Sub
```

4. 執行程式，接著在TextBox之中輸入100，然後按下「轉換」鈕，結果如圖-2。

數值資料－實數

- 數學的實數可分成小數與分數兩種，其中小數方面，VB.NET的表示法也跟數學完全相同，如3.14159、0.0016、-8000.5等均爲正確的寫法。
- 但VB.NET並沒有分數的表示法，不過我們可以用除號‘/’把分數的意義表達出來，例如把寫成1/3，只是1/3經運算後得到的值爲0.33333333333333，並不完全等於1/3，但這是電腦的限制，也是沒辦法的事情。

數值資料－科學記號

一般寫法	科學記號表示法	說 明
9.75E+09	9.75×10^{9}	將小數點左移9位,得到9.75,再乘以10^{9}
5.6E-07	5.6×10^{-7}	將小數點右移7位,得到5.6,再乘以10^{-7}

- 科學記號的表示法，在VB.NET裡面，如果使用算術運算符號，也可以輕易地將它的意義表達出來，例如：

  9.75×10^{9} 表達成 9.75*10^9

  5.6×10^{-7} 表達成 5.6*10^-7

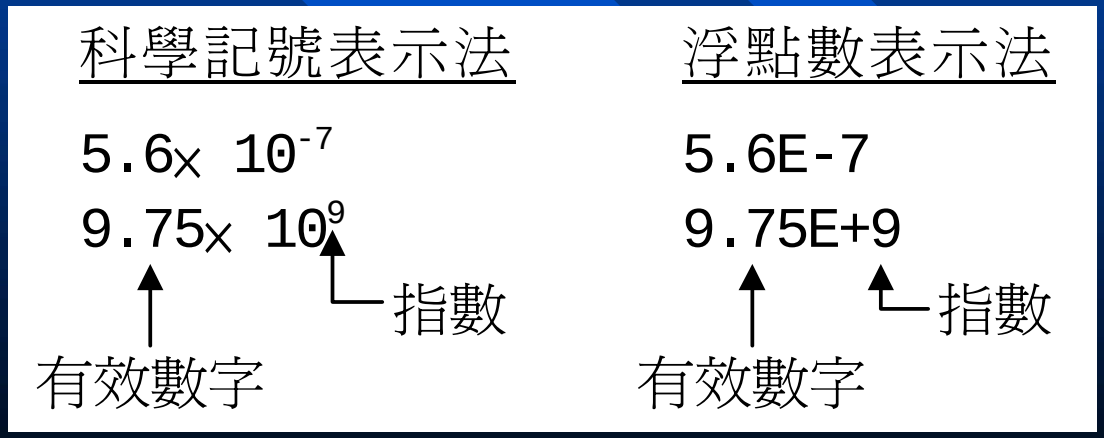
  其中＊代表乘號,^代表次方符號。

數值資料－浮點數

- 除了利用算術運算式來表示科學記號之外，VB.NET還提供了一種更有效率的表示法，叫做「浮點數表示法」，首先是抓出科學記號中的「有效數字」及「指數」，然後寫成以下格式：

  有效數字 E 指數

- 實例：



字串資料

- 字串是由一連串的字元組成的串列，在VB.NET中，一個字串的前後必須以“” 括起來。
- 由於雙引號被拿來當作界定字串的符號，所以如果字串裡面又含有雙引號，會引起VB.NET的混淆，例如 “ABC”xyz“ 就不是正確的字串，爲了避免這樣的混淆，VB.NET規定字串中的雙引號要用兩個雙引號來表示，例如：

```
Console.WriteLine( "ABC""xyz" )
' C與x之間的兩個雙引號只代表一個雙引號, 所以輸出:
ABC”xyz
```

日期時間資料

- 日期表示法：
  #月/日/西元年#
  - 實例：
    #7/1/1997# 表示 西元 1997 年 7 月 1 日
    #12/31/2003# 表示 西元 2003 年 12 月 31 日
- 時間表示法：
  #時:分:秒#、#時:分:秒AM#、#時:分:秒PM#
  - 實例：
    #17:05:10# 表示 下午 5 點 5 分 10 秒
    #9:30:00 AM# 表示 上午 9 點 30 分
    #7:55:30 PM# 表示 下午 7 點 55 分 30 秒

布林資料

- 布林資料指的是「真」與「假」的集合，又稱「真假資料」。
  - 「真」：以 True 表示。
  - 「假」：以 False 表示。
- 布林資料常用來代表一個條件式的成立與否，例如：

```
Dim X
X = 2 > 1      ‘ 把 2 > 1 的比較結果指定給 X
Console.WriteLine( X ) ' 結果輸出: True
X = 2 < 1      ' 把 2 < 1 的比較結果指定給 X
Console.WriteLine( X ) ' 結果輸出: False
```

5-2 串接運算

字串與字串的串接

- 字串與字串的串接就是把兩個字串連接成爲一個新的字串。
- 符號：+。
- 實例：

```
Dim First, Last, Name
First = "Peter"
Last = "Wang"
Name = First + " " + Last

Console.WriteLine( Name )   ‘ 輸出: Peter Wang
```

串接運算的注意事項

- '+' 號用在字串與字串之間叫做「串接運算」，用在數值與數值之間叫做「算術運算」，不要將兩者混爲一談，例如：

```
Console.WriteLine( 50 + 20 )       ' 算術運算, 輸出: 7
Console.WriteLine( "50" + "20" ) ' 串接運算, 輸出: 5
```

各種類型資料的串接

- ‘+’ 號只能用在 ‘字串與字串’ 的串接。
- ‘&’ 也是串接用的運算符號，但可用在 ‘各種類型資料’ 的串接，實例：

```
Dim S1 = "Do you love me? "
Dim Re = True
Dim S2 = S1 & Re   ' 字串與布林資料的串接
Console.WriteLine( S2 ) ‘ 輸出: Do you love me? True

Console.WriteLine( "12345." & 678 )
' 字串與數值串接後, 直接輸出, 結果輸出: 12345.678

Console.WriteLine( 678 & 0.12345 )
' 數值與數值的串接, 結果輸出: 6780.12345

Dim X = "VB " + 7.0        ' 錯誤！'+' 號只能用來串接字串與字串
```

字串與數值的相‘+’

- 字串就其內容上可分成數值型字串及文字型字串，所謂數值型字串必須是內容符合數值表示法的字串，反之，則爲文字型字串，例如：

```
"3600"
"73.59"
"1.5E-13" ' 以上三個字串皆爲數值型字串
"123 6"   ' 含有非數字的空白字元，屬於文字型字串
"#1005"   ' 含有非數字的 '#' 字元，屬於文字型字串
```

- 其中數值型字串十分特殊，當它與字串在一起做運算時，它維持字串的特性；但是當它跟數值資料一起做運算時，卻被當作數值資料。

字串與數值的相‘+’ 實例

- Console.WriteLine("123" + "ABC")
  - 輸出: 123ABC，因爲當數值型字串與文字型字串在一起做運算時，數值型字串仍維持字串特性，所以運算符號‘+’被視爲字串的串接。
- Console.WriteLine(“123” + 100)
  - 輸出: 223，因爲當數值型字串與數值在一起做運算時，數值型字串被當作數值，所以運算符號‘+’被視爲數值的相加。
- Console.WriteLine("123" + "100")
  - 輸出: 123100，因爲當兩個數值型字串在一起運算時，兩個數值字串都維持字串特性，所以運算符號'+' 還是被視爲字串的串接。

5-3 資料型別與變數宣告敘述

VB.NET的資料型別

- VB.NET依性質將資料分成數值、字串、日期時間、布林等四類。
- 資料類別不同，運算的方式也會不同。
- 數值資料比較特殊，在考慮到運算效率、所佔空間、及精確程度等因素下，VB.NET把數值資料的儲存方式又分成位元組整數型別、短整數型別、整數型別、長整數型別、十進位數型別、倍精準度型別、單精準度型別等。

資料型別分類表

資料類別	資料型別	型別英文名稱
字串	字串	**String**
日期時間	日期時間	**Date**
布林	布林	**Boolean**
數值	位元組整數	**Byte**
	短整數	**Short**
	整數	**Integer**
	長整數	**Long**
	十進位數	**Decimal**
	倍精準度	**Double**
	單精準度	**Single**

檢驗資料的型別

```
Console.WriteLine( TypeName(“Computer”) )
' 輸出: String

Console.WriteLine( TypeName(#11/25/2003#) )
' 日期資料, 輸出: Date

Console.WriteLine( TypeName(#12:35:50#) )
' 時間資料, 輸出: Date。 註：日期與時間統稱「Date」型別。

Console.WriteLine( TypeName(True) )
' 布林資料, 輸出: Boolean

Console.WriteLine( TypeName(1.5) )
' 小數, 輸出: Double(倍精準度型別)

Console.WriteLine( TypeName(4.9E+12) )
' 浮點數, 輸出: Double(倍精準度型別)

Console.WriteLine( TypeName(100) )
' 整數, 輸出:Integer(整數型別)
```

變數宣告敘述

- 格式：
  Dim 變數名稱 As 型別名稱
  - 型別名稱可以是String、Date、Boolean、Byte、Short、Integer、Long、Decimal、Double或Single
  - 實例：

```
Dim A As String          ‘將變數A宣告成String型別
Dim B As Date            ‘將變數B宣告成Date型別
Dim C As Boolean         ‘將變數C宣告成Boolean型別
Dim D As Integer         ‘將變數D宣告成Integer型別
Dim E As Decimal         ‘將變數E宣告成Decimal型別
```

注意事項

```
Dim A ‘ 變數A沒有明確指定資料型別
Dim B As Integer  ‘ 變數B指定成整數型別
A = 123
B = 123
A = “ABC”        ‘沒問題, 變數A的型別由整數變成字串
B = “ABC”        ‘錯誤, 變數B的型別永遠都是整數, 不接受字串資料
Dim C As Decimal = 98.7
B = C   ' 將98.7指定給整數型別的B
Console.WriteLine( B )          ‘ 輸出: 99, 四捨五入
```

- 明確指定變數的資料型別之後，變數所能接受的資料也會受到限制。未明確指定資料型別的變數(如以上例子中的變數A)可接受任意型別的資料；明確指定資料型別的變數(如以上例子中的變數B)只能接受相同型別或相容型別的資料。

注意事項

- 除了相容型別之外，若不同型別之間的資料是可以互相轉換的，也將爲VB.NET所接受，例如：

```
Dim B As Integer
Dim S1 As String = “123”
Dim S2 As String = “ABC”

B = S1          ‘ 正確, S1的內容爲 “123”，可以轉換成整數
Console.WriteLine( B )  ‘ 輸出: 123

B = S2          ‘ 錯誤, S2的內容爲 “ABC”, 無法轉換爲整數
S2 = 99         ‘ 正確, 99可以轉成爲字串的 “99”
B = S2  ‘ 正確, S2的內容變成 “99”, 可以轉換爲整數
Console.WriteLine( B )   ‘ 輸出: 99
```

5-4　數值資料型別

分類	型別名稱	中文譯名
整數	Byte	位元組整數
	Short	短整數
	Integer	整數
	Long	長整數
實數	Decimal	十進位數
	Double	倍精準度
	Single	單精準度

整數類資料型別

- 整數類型別有Byte(位元組)、Short(短整數)、Integer(整數)、Long (長整數)四種，它們最大的差異在於所佔空間(位元組數, bytes)，而所佔空間將連帶影響數值範圍，如下表：

型別名稱	所佔空間	數值範圍
Byte(位元組數)	1 bytes	0～255 (2^0-1～2^8-1)
Short(短整數)	2 bytes	-32768～32767 (-2^{15}～2^{15}-1)
Integer(整數)	4 bytes	-2147483648～2147483647 (-2^{31}～2^{31}-1)
Long(長整數)	8 bytes	-9223372036854775808～ 9223372036854775807 (-2^{63}～2^{63}-1)

使用整數的觀念一：數值範圍

- 數值範圍限定了各整數型別的最大值及最小值.
  - 舉例來說，Short型別的最小值是 -32768、最大值是32767，運算結果絕不可以超出此一範圍，否則會產生錯誤，例如：

```
Dim X As Short = 32700
X = X + 300  ‘ 超出Short的數值範圍，產生錯誤!
```

在以上敘述中，變數X被宣告成Short整數，一開始其值等於32700，但接下來加上300後其結果33000超出Short型別的數值範圍，所以無法指定給Short整數，會產生錯誤。

使用整數的觀念二：執行效能

- Long型別的乘除速度比其他整數型別(Integer、Short、Byte)慢。
- Integer、Short、Byte三種型別的乘除法速度並無太大的差異。
- 建議：如果範圍介於 –2147483648與2147483647之間，則可以選擇執行效能較佳的Integer型別，否則才選擇Long型別。

實數類資料型別

- 實數類型別有Decimal(十進位數)、Single(單精準度)、Double(倍精準度)三種，其所佔空間、有效位數、數值範圍的差異，如下表：

型　別	所佔空間	有效位數	數值範圍
Decimal	12 bytes	29 位	-7.92282E+28 ～792281625142643375935439503335
Double	8 bytes	15 位	-1.79769313486232E+308 ～1.79769313486232E+308
Single	4 bytes	7 位	-3.40E+38 ～3.402823E+38

使用實數的觀念一: 有效位數

- 有效位數指的是不包含小數點所能表示的位數，在這幾個實數型別中，Single型別的有效位數是7位、Double型別15位、Decimal型別29位。
- 在指定敘述中，如果資料的位數超過其所屬型別的有效位數，一律四捨五入，例如：

```
Dim X1 As Single  ' 宣告一個Single變數，其有效位數是7位

X1 = 1.23456789   ' 指定的數1.23456789是9位數,
      ‘ 四捨五入後成爲7位(等於1.234568)
Console.WriteLine(X1)   ' 輸出: 1.234568
```

使用實數的觀念一: 有效位數

```
Dim X2 As Double   ' 宣告一個Double變數，其有效位數是15位

‘ 指定敘述(1), 小數超過15位
X2 = 1.234567890123456 7
Console.WriteLine(X2)     ' 輸出: 1.23456789012346

‘ 指定敘述(2), 沒有超過15位
X2 = 12345678901234
Console.WriteLine(X2)     ' 輸出: 12345678901234

‘ 指定敘述(3), 整數超過15位
X2 = 1234567890123456 7
Console.WriteLine(X2)     ' 輸出: 1.23456789012346E+16
```

- 1.234567890123456 7超過15位，四捨五入成爲1.23456789012346。
- 12345678901234沒有超過15位，不必改變依然等於12345678901234。
- 1234567890123456 7超過15位，四捨五入成爲1234567890123460，但會輸出成:1.23456789012346**E+16**。

使用實數的觀念一: 有效位數

- 使用 Decimal 的注意事項：

```
Dim X3 As Decimal
X3 = 123456789012345678901234 5      ‘ 錯誤訊息: 溢位
```

  - 原敘述應改成：

```
X3 = 123456789012345678901234 5D
```

- 錯誤例子二：

```
Dim X4 As Decimal
X4 = 1.23456789012345678901234 5
Console.WriteLine(X4)          ‘ 輸出: 1.23456789012346
```

  應改成：

```
X4 = 1.23456789012345678901234 5D
```

使用實數的觀念二：數值範圍

- 所佔空間較少的Double及Single型別(分別佔用8 bytes及4 bytes)其有效位數雖然少於Decimal型別，但數值範圍卻大於Decimal型別，其中Double的最大值爲1.7976931348623$2E+308$，是Decimal型別最大值的10^{279}倍，即使是較小的Single型別其最大值爲3.402823$E+38$，也是Decimal型別最大數的10^{9}倍。

使用實數的觀念二：數值範圍

- Double及Single型別除了能表示較大的數之外，也能夠表示較接近0的數，對Decimal型別來說，可表達最接近0的數等於：

  $\pm 0.0000000000000000000000000001 = \pm 10^{-28}$

  而Double及Single型別可表示最接近0的數卻分別等於：

  Double: ±4.9406564841247E-324

  Single: ±1.401298E-45

  其細微程度分別是Decimal型別的2×10^{295}及7×10^{16}倍。

使用實數的觀念三: 數值的誤差

- 從數值範圍來看，Double及Single型別似乎優於Decimal型別，但是當Double及Single型別使用於微小資料時要特別注意，因爲Double及Single型別計算微小資料時會產生誤差：

```
Dim D1 As Double
Dim D2 As Decimal
Dim I As Integer

‘ D1及D2每次加1.0E-20, 共加1000次
For I = 1 To 1000
    D1 = D1 + 1.0E-20
    D2 = D2 + 1.0E-20
Next

Console.WriteLine(D1)    ‘ 輸出: 1.00000000000002E-17
Console.WriteLine(D2)    ‘ 輸出: 1.0E-17
```

數值常數的型別

- 表示數值常數時，VB.NET會根據以下規則來設定其型別：
  1. 小數或浮點數，將該數值常數設定成Double型別。
  2. 數值介於 -2147483648～2147483647，將該數值常數設定成Integer型別。
  3. 數值介於 -9223372036854775808～9223372036854775807，將該數值常數設定成Long型別。

```
1.0     小數,     Double型別
5.6E-7          浮點數, Double型別
12345 整數,     Integer型別
12345678901234      整數,     Long型別
```

自定數值常數的型別

型別標示字元	型別	型別中文名稱
S	Short	短整數
I	Integer	整數
L	Long	長整數
F	Single	單精準度數
R	Double	倍精準度數
D	Decimal	十進位數

- 如果不想採用VB.NET預設的規則來設定數值常數的型別，可以在常數之後加上S、I、L、F、R、D等字元自定常數的型別。

自定數值常數的型別

- 實例：

  1.0F　　在數值之後加上F, 這是Single型別常數
  5.6E-7D 在數值之後加上D, 這是Decimal型別常數
  12345S　在數值之後加上S, 這是Short型別常數
  123456789012D 在數值之後加上D, 這是Decimal型別常數

小心！0.5的四捨五入

- 當含有小數的數值被轉成整數時，VB.NET處理四捨五入的方式，有一個地方與我們過去的習慣不同，例：

```
Dim I As Integer          ‘ I 爲整數型別
I = 1.5    ’四捨五入之後指定給I
Console.WriteLine(I)  ‘ 預期: 2, 結果: 2, 相符
I = 4.5
Console.WriteLine(I)  ’預期: 5, 結果: 4, 不相符
```

- 當被捨棄的小數值剛好等於0.5時，並不像我們過去的習慣一樣馬上進位，而是必須先判斷前一位是偶數還是奇數，如果是奇數則進位，偶數則捨棄，所以：

```
I = 1.5    ‘ 被捨棄的 0.5 前一位是奇數，所以進位
I = 4.5    ’被捨棄的 0.5 前一位是偶數，所以捨棄
I = 4.5001          ' 被捨棄的 5001 不是剛好等於 5，
    ' 所以與我們過去的習慣一樣，直接進位
Console.WriteLine(I)  ' 結果: 5
```