الفصل الرابع عشر
أنواع العمليات والتعبيرات ومعالجة السلاسل الحرفية

كما أن العمليات الحسابية أمر هام في حياتنا اليومية، فإنها أمر تحتوي بالنسبة للبرامج، وفي هذا الفصل ستتعرف على أساسيات إنجاز العمليات الحسابية والمنطقية، وستعرف أيضا كيفية معالجة سلاسل البيانات النصية والتاريخ والوقت.

بالانتهاء من هذا الفصل ستكتسب المعارف وتتدرب على المهارات التى تجعلك قادرا على:

- المعاملات الإسنادية
- المعاملات الحسابية
- معاملات ربط (وصل) العبارات
- المعاملات العلائقية
- المعاملات المنطقية
- أولويات تنفيذ المعاملات
- استخدام دوال السلاسل الحرفية

استخدام المعاملات Operators

يتيح **Visual Basic** لمبرمجيه إنجاز العديد من العمليات الحسابية والمنطقية والتعامل مع العبارات النصية والتى يصعب أن تجدها مجتمعه فى لغات أخرى بنفس الإمكانيات والسهولة المصاحبة لكل عملية ومن أشهرها:

- العمليات الإسنادية
- العمليات الحسابية **Arithmetic Operations**
- عمليات ربط العبارات **String Operations**
- العمليات العلائقية **Relational Operations**
- العمليات المنطقية **Logical Operations**

وتستخدم كل عملية من العمليات السابقة عدة عوامل أو معاملات (Operators). فمثلا يلزم لإنجاز العمليات الحسابية عدة معاملات مثل معاملات الجمع "+" أو الطرح "-" أو الضرب "*" والقسمة المشهورة "/". ستتعلم في هذا الفصل المعاملات التى تلزم لإنجاز العمليات الأخرى التى ذكرناها.

كلمة معامل أو عامل تقابل كلمة Operator التى يستخدمها Visual Basic وقد تجدها هنا في الشرح مرة عامل ومرة معامل وكلها بمعنى Operator.

وفيما يلي نوضح كيفية إنجاز كل عملية من هذه العمليات الأساسية وسنشرح أيضا استخدام معاملات كل مجموعة من هذه المجموعات.

العمليات الإسنادية

نحتاج الآن بعد معرفة حافظات البيانات (المتغيرات والثوابت) إلى معرفة كيف يمكننا قراءتهما واستخدامهما أو حفظ القيمة في المتغيرات و التعامل معها. في الحقيقة لاستخدام القيمة المخزنة في أحد المتغيرات يكفي ذكر اسم المتغير. مثلا لطباعة قيمة المتغير Salary على شريط عنوان النموذج Form1 نستخدم العبارة التالية:

```
Form1.Text = Salary
```

وبنفس الطريقة يمكن استخدام هذا الاسم في تعبيرات حسابية Arithmetic Expressions. وبالنسبة لتغيير قيمة متغير فإننا نستخدم عبارة التخصيص Assignment Statement وهي من أشهر و أهم عبارات البرمجة. أما إذا أعلنا عن المتغير فقط دون تخصيص قيمة له فإن Visual Basic يعطيه قيمة تلقائية بمجرد الإعلان عنه هذه القيمة تكون كالتالي:

نمط المتغير	القيمة التلقائية
الأنماط الرقمية مثل Integer, Single,...	0
نمط التاريخ Date	30/12/1898
نمط سلسلة البيانات String	""
النمط المنطقي Boolean	False

و تحتاج عادة إلى عملية تخصيص قبل استخدام المتغير تسمى عملية الشحن بالقيم الابتدائية أو الاستهلال **Initialization** كي تضع قيمة مناسبة للمتغير خلاف هذه القيمة التلقائية.

عبارات التخصيص

لحفظ قيمة معينة في أحد المتغيرات (بعد الإعلان عنه) يتم استخدام عبارة التخصيص **(Assignment Statement)** والتي تأخذ الصورة:

```
varname = value
```

Value في العبارة السابقة يمكن أن تكون قيمة مباشرة (أرقام أو سلاسل حرفية) أو ثابت أو متغير أو وظيفة **Method** أو تعبير حسابي **Expression** يتضمن كل ما سبق بالإضافة إلى العوامل التي تضمها. الشرط أن تكون قيمة من نفس نمط المتغير أو من نمط مكافئ.

من أمثلة عبارات التخصيص:

```
inAge = 25
dbCritical = 2.1^1.5+Sin(3.5)
strName= "Waleed" & "Mohammed"
dtBirthDate = # 1/8/1974 #
sgAvg = sgTotal / sgCount
```

كما ترى تشبه العبارات السابقة العبارات التي نغير من خلالها قيمة إحدى خصائص كائن مثل النموذج، وفي الحقيقة يتم التعامل مع الخصائص كالمتغيرات تماما لأنها في الحقيقة متغيرات. و يمكن نقل قيمة خاصية مثلا في متغير للتعامل معها بعبارة التخصيص، مثل:

```
Stname = Txtname.Text
```

و العكس أيضا صحيح مثل:

```
Innumber = Txtnumber.Text
```

كما ترى في المثال السابق يتم تخصيص قيمة حرفية **string** لمتغير عددي **inNumber**، كيف ذلك ؟ ألم نقل أن الأنماط يجب أن تتكافأ؟ في الحقيقة يقوم **Visual Basic** بالتحويل بين الأنماط متى أمكن. إلا أنك يجب ألا تعتمد على ذلك لأسباب عديدة. أحدها أنه لو كان مربع النص **txtName** لا يحتوي على عدد و إنما على حروف أبجدية فإنك ستحصل على خطأ يسمى بخطأ "عدم توافق الأنماط" **Type**

Mismatch Error لأنك أردت أن تضع قيمة حرفية في متغير رقمي.و يستحسن دائما أن تقوم بهذا التحويل بنفسك إن أردت كالتالي:

```
inNumber = Convert.ToInt32(txtNumber.Text)
```

استخدمنا هنا دالة التحويل ToInt32() الموجودة داخل التصنيف Convert ومعناها حول نمط المتغير إلى النمط Integer "عدد"، وهناك العديد من دوال التحويل فيVisual Basic وكلها موجودة داخل التصنيف Convert. تبدأ كلها بحرفى To ويعقبها النمط الذي تريد التحويل إليه كما فى جدول ١٤-١.

جدول ١٤-١ دوال التحويل داخل Visual Basic 2008

النوع	دالة التحويل
Byte	To Byte
Date	To Date Time
Double	To Double
Decimal	To Decimal
Integer	ToInt32
Long	ToInt64
Single	To Single
Char	To Char
String	To String
Short	ToInt16
Boolean	To Boolean
SByte	To SByte

هناك حل آخر للمشكلة السابقة وهو أن نتأكد من وجود قيمة رقمية في مربع النص قبل إجراء عملية التخصيص حتى لا تصدر رسالة خطأ، وذلك بإضافة كود كالتالي:

```
If IsNumeric (txtNumber.Text) Then
        InNumber=txtNumber.Text
End if
```

العمليات الحسابية Arithmetic Operations

كما أن العمليات الحسابية ضرورية في حياتنا اليومية فإن العمليات الحسابية أمر حيوى بالنسبة لتطوير البرامج. تستخدم لغة Visual Basic العمليات الحسابية الآتية:

عملية الجمع Addition

تستخدم المعامل (الرمز) "+" (كلمة معامل أو رمز هنا تقابل كلمة Operator) وتكون

بين قيمتين ثابتتين كما فى هذا المثال:

```
Debug.WriteLine(3.12 + 12)
```

وعند تنفيذ هذا الأمر ستحصل على الناتج 15.12 داخل النافذة Immediate Window بالجزء السفلى من نافذة بيئة التطوير.

وقد تكون إحدى القيمتين متغير أو أحدهما ثابت والأخر متغير كما فى المثال التالى (اكتب الأوامر التالية داخل أى إجراء):

```
A = 5 : B = 6.5
Debug.WriteLine(A +B)
Debug.WriteLine(A + B + 2)
```

وعند التنفيذ سيكون الناتج كما يلى

```
11.5
13.5
```

وكما تلاحظ فإن عملية الجمع تتعامل مع قيم من جميع الأنواع.

عملية الطرح Subtraction

تستخدم المعامل (الرمز) (-) وبنفس الطريقة السابقة تتعامل مع جميع القيم بجميع أنواعها، انظر المثال التالى والنتيجة التى ستحصل عليها بعد التنفيذ (اكتب الأوامر فى أحد الإجراءات).

```
A = 10 : B = 8.5
Debug.WriteLine(A - B)
Debug.WriteLine(B - A)
```

وعند التنفيذ سيكون الناتج كما يلى:

```
1.5
-1.5
```

وتدل العلامة السالبة (-) قبل القيمة من اليسار على أن القيمة الأولى (8.5) كانت أقل من القيمة الثانية (10).

عمليتا الضرب والأس Multiplication / Exponentiation

يستخدم في عملية الضرب (Multiplication) المعامل "*" ويستخدم في عملية الأس (Exponentiation) المعامل "^" وتستخدم العمليتان بنفس الطريقة السابقة.

مثال (اكتب التعليمات التالية فى أحد الإجراءات):

```
Debug.WriteLine( 2*3 )
Debug.WriteLine(2 ^ 3)
```

```
Debug.WriteLine(1.5 * 7)
Debug.WriteLine(3.4 ^ 1.29)
```

ستحصل على الناتج التالى (على الترتيب):

```
6
8
10.5
4.848145
```

عمليات القسمة والناتج الصحيح من القسمة وباقى القسمة

يستخدم مع عملية القسمة **(division)** الرمز "/" . انظر المثال التالى والنتيجة التى تحصل عليها.

مثال:

```
Debug.WriteLine( 8/2 )
Debug.WriteLine( 10/3)
```

الناتج هو:

```
4
3.33
```

كما يتيح لك **Visual Basic** عمليتين أخرتين الأولى هى عملية الناتج الصحيح من القسمة **(Integer division)** ويرمز لها بالرمز "\" بدلاً من الرمز السابق "/" ويكون ناتج هذه العملية الجديدة هو الجزء الصحيح (دون تقريب) فقط.

مثال:

```
Debug.WriteLine( 14\5 )
Debug.WriteLine( 10\3)
```

الناتج هو:

```
2
3
```

أما العملية الثالثة فهى عملية باقى القسمة **(Reminder)** ويرمز لها بالرمز **(mod)** ويكون ناتج هذه العملية هو الجزء المتبقى من القسمة فقط.

مثال:

```
Debug.WriteLine( 14\5 )
Debug.WriteLine( 14 Mod 5)
```

الناتج هو:

```
2
4
```

حيث أن ناتج قسمة الرقم (14) على الرقم (5) هو العدد الصحيح (2) ويتبقى القيمة (4).

ومن السمات الجديدة فى Visual Basic 2008 استخدام الصيغ المختصرة فى العمليات الحسابية. يوضح جدول ١٤-٢ التالى الصيغة المختصرة للعمليات الحسابية المختلفة.

جدول ١٤-٢الصيغ الحسابية المختصرة

العملية	الصيغة الطويلة	الصيغة المختصرة
الجمع	X=X+2	X+=2
الطرح	X=X-2	X-=2
الضرب	X=X*2	X*=2
القسمة	X=X/2	X/=2
قسمة الرقم الصحيح	X=X\2	X\=2
الرفع للأس	X=X^2	X^=2
ربط السلاسل	X=X& "Text"	X&="Text"

وصل (ربط) العبارات

إذا أردت إدماج متغيرين (كل متغير له قيمة حرفية معينة) فى متغير آخر جديد فعليك باستخدام المعامل "&" ويسمى Ampersand ويوضع بين القيمتين كما فى المثال التالى:

```
First = "Compuscience "
Second = "Company"
Result  = First  & Second
```

الناتج هو:

```
Compuscience Company
```

كما يمكنك استخدام المعامل "+" بدلاً من المعامل "&" للحصول على نفس النتيجة السابقة.

```
First  = "Compuscience "
Second = "Company "
```

```
Result = First + Second
```

الناتج هو أيضاً:

```
Compuscience Company
```

عمليات المقارنة

نقصد بها عمليات مقارنة بين قيمتين ويكون الناتج بالقيمة True وتعنى نتيجة صحيحة أو القيمة False وتعنى نتيجة خاطئة. وتعرف القيمة True أو القيمة False على أنها قيمة منطقية. ويوضح الجدول ١٤-٣ التالى المعاملات العلائقية وأمثلة على استخدامها.

جدول ١٤-٣ المعاملات العلائقية وأمثلة على استخدامها.

المعامل	معناه	أمثله	
=	Equals يساوى	3=3	"2" = "2"
< >	Not equal لا يساوى	4 <>2	"A" <> "B"
>	Greater than اكبر من	8 > 5.4	"B" > "A"
<	Less than اصغر من	4 < 6.5	"C" < "B"
> =	Greater than or equal اكبر من أو يساوى	5 > = 5 6 > = 5	"A" >= "A" "D" >="C"
< =	Less than or equal اصغر من أو يساوى	7 < = 7 6 < = 7	"B" < = "B" "A" < = "B"

العمليات المنطقية

يستخدم Visual Basic ستة أنواع من العوامل المنطقية تتعامل جميعها مع قيمتين منطقيتين (ما عدا العملية Not فتتعامل مع قيمة واحدة منطقية فقط) والجدول ١٤-٤ التالى يبين وظيفة كل عامل من العوامل المنطقية.

جدول ١٤-٤ العوامل المنطقية.

المعامل	اسمه	وظيفته
Not لا	Complement	في حالة Not (True) يكون الناتج False وفي حالة Not (False) يكون الناتج True.
And و	Conjunction	لابد أن يكون التعبيران الشرطيان صحيحان لتكون النتيجة True وفى الأحوال الأخرى تكون النتيجة False.
OR أو	Disjunction	تكون النتيجة True إذا كان أحد التعبيرين الشرطيين صحيحا وتكون النتيجة False فى الأحوال الأخرى.
XOR أو المقتصرة	Exclusive OR	تكون النتيجة True إذا كان واحد فقط من التعبيرين الشرطيين صحيحا، وتكون النتيجة False إذا كان كلاهما صح أو خطأ
Eqv تكافؤ	Equivalence	تكون النتيجة True إذا كان كلا التعبيرين الشرطيين صحيحا أو خطأ.

أولويات تنفيذ المعاملات

إذا اشتمل الأمر على أكثر من نوع من العوامل السابقة فان تنفيذ هذه العوامل يتم طبقاً لأولويات محددة والجدول ١٤-٥ التالى يوضح ترتيب تنفيذ هذه الأولويات، فمثلا الترتيب رقم ١ فى الجدول يسبق الترتيب رقم ٢ فى التنفيذ ...وهكذا بمعنى أن عمليات فك الأقواس تأتى قبل عمليات الأس يليها عملية الإشارة السالبة ثم عمليات الضرب والقسمة ...وهكذا.

جدول ١٤-٥ أولويات المعاملات

المعامل	معناه	الترتيب
( )	الأقواس Parantheses	١
^	الأس Exponentiation	٢
-	الإشارة السالبة Negation	٣
* , /	الضرب والقسمة Multiplication, division	٤
\	الناتج الصحيح من القسمة Integer division	٥
Mod	باقى القسمة Modulo Arithmetic	٦
+ , -	الجمع والطرح Addition, Subtraction	٧
=, <, >, <>, <=, >=	عوامل علائقية Relational Operators	٨
Not	عوامل منطقية Logical Negation	٩
And	"و" Conjunction	١٠
OR	"أو" Inclusive OR	١١
XOR	"أو المقتصرة" Exclusive OR	١٢
EqV	"تكافؤ" Equivalence	١٣

أمثلة:

مثال ١ : العملية 2 * 3 + 4

يتم تنفيذها طبقا للترتيب التالى:

١. يتم حساب عملية الضرب أولاً (الضرب يسبق الجمع) 2 * 3 = 6
٢. يتم حساب عملية الجمع ثانيا 6 + 4 = 10

مثال ٢: العملية 2 * (3+4)

يتم تنفيذها طبقا للترتيب التالي:

١. يتم حساب ما بين القوسين أولاً (حيث أن الأقواس تسبق عملية الضرب) 3+4=7
٢. يتم حساب عملية الضرب ثانيا 2 * 7 = 14 ويكون الناتج هو 14.

التعامل مع السلاسل الحرفية

تعرضنا فيما سبق للسلاسل الحرفية Strings وقلنا أنها الصورة التي يتم بها تخزين النصوص في ذاكرة الحاسب. حيث يشغل كل حرف أو رمز بايت واحد من الذاكرة.

والسلاسل الحرفية نوع هام جدا من البيانات، حتى أن البيانات يمكن تقسيمها إلى نوعين أساسين هما البيانات الرقمية والسلاسل الحرفية. قد لا تشعر بأهمية السلاسل الحرفية في الوقت الحاضر لو كنت من المبرمجين المبتدئين ولكن مع الوقت ستدرك أهميتها وتحتاج إلى هذا الشرح. فيما بقي من هذا الفصل ستعرف كيف تتعامل مع السلاسل الحرفية.

لاحظ أن أي نص، مثل اسم الكائن Name وعنوانه Text الذي تكتبه في مربع هو في النهاية سلسلة حرفية.

بالرغم من سهولة تعديل النصوص داخل مربعات النصوص Text Boxes إلا أنه في كثير من الأحيان تحتاج إلى التعديل في النص بواسطة البرنامج وهناك العديد من الدوال (الوظائف) التي يمكن أن تستخدم لذلك. ولغة Visual Basic من اللغات الغنية في هذا المجال، حيث يوجد بها عدد كبير من الدوال المتخصصة في التعامل مع السلاسل الحرفية.

جميع هذه الدوال يتم تمرير نص إليها في صورة متغير حرفي String وترجع متغيراً من النوع String.

تنقسم الدوال التي تتعامل مع السلاسل الحرفية إلى عدة فئات:

- منها ما يقوم بحذف أو استقطاع جزء من السلسلة الأصلية ويكون بها سلسلة جديدة ، يشمل الدوال: Left و Right وMid وSubString.

- ومنها ما يقوم باستخراج سلسلة حرفية من أخرى بعد حذف المسافات الخالية Spaces وتشتمل على: Trim وTrimEnd وPadLeft.
- ومنها الذي يقوم بالتحويل بين الأحرف اللاتينية الكبيرة والصغيرة وتضم ToUpper وToLower وUCase وLCase.
- وهناك الدالة InStr والدالة IndexOf التي تبحث في سلسلة حرفية عن سلسلة أخرى وتعطي مكانها. والدالة Len أو الخاصية Length التي تعطي طول السلسلة الحرفية (عدد حروفها).
- وأخيراً الدالة Replace المستخدمة في استبدال جزء داخل نص بجزء آخر والدالة strReverse المستخدمة لعكس النص وغيرها من دوال تفكيك النص وتجميعه.

ونوضح في الفقرات التالية كيفية استخدام هذه الدوال.

تغيير حالة الأحرف

هناك أربعة دوال تستخدم في تغيير حالة الأحرف اللاتينية وهما ToUpper وUCase وToLower وLCase وكما يتضح من الأسماء الأولى والثانية ترجع نص حروفه كلها تم تحويلها إلى حروف كبيرة Capitals والثالثة والرابعة تفعل العكس أي ترجع نص كل حروفه الهجائية Small. من الاستخدامات الجيدة لهذه الدالة، مقارنة قيمتين حرفيتين لا نهتم فيها بحالة الحروف.

المثال التالي يوضح كيف يمكن مقارنة مدخلات بقيمة حرفية داخل البرنامج (مكتوبة بحروف كبيرة):

```
Select Case Ucase(txtInput.Text)
Case "TEACHER"
        'Login as Teacher
Case "STUDENT"
        'Login as Student
Case "OFFICER "
        'Login as Officer
End Select
```

في المثال السابق لا يهم أن يدخل المستخدم النص بحروف كلها كبيرة بل يكفي أن تتطابق الأحرف سواء كبيرة أو صغيرة.

يمكنك استخدام الدالة **ToUpper** بدلاً من الدالة **UCase** ليصبح الكود السابق كما يلي:

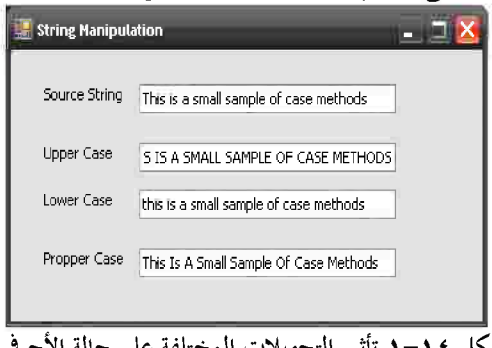
```
Select Case txtInput.Text.ToUpper
Case "TEACHER"
        'Login as Teacher
Case "STUDENT"
        'Login as Student
Case "OFFICER "
        'Login as Officer
End Select
```

وبنفس الطريقة يمكنك استخدام الدالتين **LCase** و**ToLower**.

من الدوال الأخرى المستخدمة في تغيير حالة الأحرف ، الدالة **StrConv** وهي تستخدم في تحويلات متعددة منها التحويل بين الأحرف الكبيرة والصغيرة، ومنها تحويلات لا تهمنا مثل تحويلات بين حروف يابانية. ولكن أحد التحويلات الممكن القيام بها يستحق الذكر وهو تحويل الأحرف إلى "الحالة المناسبة" **Proper Case** ، أي تحويل الحرف الأول من الكلمة إلى الشكل الكبير وتحويل بقية الأحرف إلى الشكل الصغير في كافة الكلمات في النص. مثال لذلك في السطر التالي :

```
StrConv( TextBox1.Text, VbStrConv.ProperCase)
```

يوضح شكل ١٤-١ التالي التحويلات المختلفة للأحرف.



شكل ١٤-١ تأثير التحويلات المختلفة على حالة الأحرف.

حذف وإضافة أجزاء إلى النص

من الوظائف الهامة المطلوبة لمعالجة المتغيرات الحرفية إمكانية إضافة أو حذف جزء من النص، وكذلك البحث عن نص داخل نص كبير. أو استبدال جزء من نص بآخر. وهذا هو ما سنتناوله فيما يلي.

البحث عن نص داخل آخر

نحتاج إلى البحث عن نص داخل نص آخر للعديد من الأسباب منها أن يكون النص مكون من العديد من المقاطع كل منها يحمل معلومة فنحتاج إلى تقطيعه والحصول على كل معلومة على انفراد. أول الدوال المستخدمة في البحث عن نص داخل نص آخر هي الدالة **InStr**.

صيغة استخدام هذه الدالة كالتالي:

```
InStr(sourceStr,searchStr)
```

والشكل البسيط لاستخدامها أن نحدد النص الذي سنبحث فيه **SourceStr** والذي سنبحث عنه **SearchStr** وسترجع لنا الدالة ترتيب الحرف الذي وجدت عنده النص المراد ، حيث دليل الحرف الأول من النص يكون ١. أما إن لم تجد النص الذي تبحث عنه فسترجع القيمة صفر.

مثال:

```
chpos=InStr("My Name Is Ahmed","Is") ' Returns 9
chpos=InStr("My Name Is Ahmed","he") ' Returns 0
```

لو أردت البحث عن القيمة الحرفية بدءاً من حرف معين في النص المصدر (الذي سنبحث فيه) وليس من أول حرف يمكن أن تحدد هذا الحرف كالتالي:

```
chpos=InStr("My Name Is Ahmed","Is",4) 'Returns 9
```

لاحظ أن القيمة المرجعة من الدالة لا تتأثر، كل ما هنالك أنك تحاول اختصار زمن البحث.

توضيح أكثر لتأثير هذا المعامل في المثال التالي:

```
chpos=InStr("My Name Is Ahmed","m")  'Returns 6
chpos=InStr("My Name Is Ahmed","m",7)  'Returns 14
```

في المثال السابق ترجع الدالة عند البحث عن m القيمة ٦ إذا بدأت البحث من أول حرف. أما عند البدء من الحرف السابع فلن ترى الحرف السادس ولذا سترجع القيمة 14 وهو أول حرف m تالي.

هناك معامل اختياري آخر لهذه الدالة وهو يحدد إن كان البحث سيأخذ في اعتباره حالة الأحرف Case-Sensitive أم لا Case-Insensitive. القيمة الافتراضية هي CompareMethod.Binary (أو 0) ويتم فيها أخذ حالة الأحرف في الاعتبار، بينما القيمة CompareMethod.Text (أو 1) تهمل حالة الأحرف عند البحث.

مثال:

```
chpos=InStr("My name Is Ahmed","N")  'Returns 0
chpos=InStr("My name Is Ahmed","N",CompareMethod.Text)
'Returns 4
```

في المثال الأول اهتم البحث الافتراضي (القيمة صفر) بحالة الأحرف، ولذلك اعتبر حرف "n" غير موجود عند البحث عن حرف "N". أما في المثال الثاني فاعتبر الحرف موجوداً.

المثال التالي يوضح استخدام هذه الدالة في عملية "تقطيع" نص إلى كلماته منفردة. يتضمن ذلك بحث متكرر عن الحروف واستقطاع للكلمات:

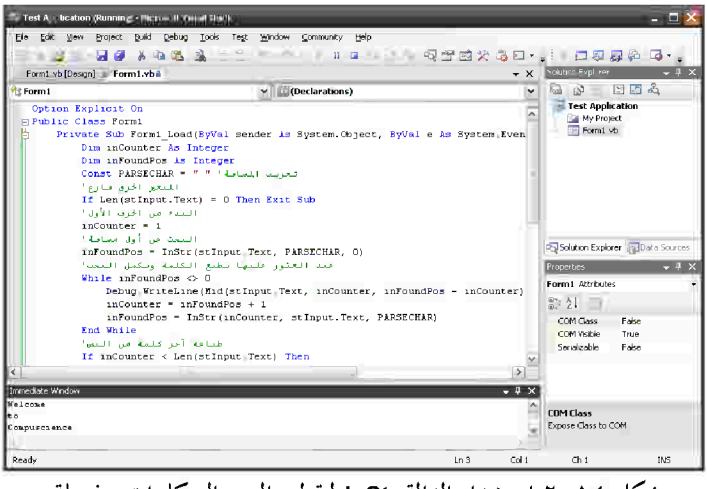
```
Dim inCounter As Integer
Dim inFoundPos As Integer
Const PARSECHAR = " " 'تعريف المسافة
'المتغير الحرفي فارغ
If Len(stInput.Text) = 0 Then Exit Sub
'البدء من الحرف الأول
inCounter = 1
'البحث عن أول مسافة
inFoundPos = InStr(stInput.Text, PARSECHAR, 0)
'عند العثور عليها نطبع الكلمة ونكمل البحث
While inFoundPos <> 0
Debug.WriteLine(Mid(stInput.Text, inCounter, inFoundPos -
inCounter))
```

```
inCounter = inFoundPos + 1
inFoundPos = InStr(inCounter, stInput.Text, PARSECHAR)
End While
'طباعة آخر كلمة من النص
If inCounter < Len(stInput.Text) Then
Debug.WriteLine(Mid(stInput.Text, inCounter))
End If
```

إذا قمت على سبيل المثال يوضع العبارة التالية داخل مربع النص **stInput**:

Welcome To Compuscience

فستحصل على النتيجة التالية داخل النافذة الفورية **Immediate Window** (انظر شكل ١٤-٢).



شكل ١٤-٢ استخدام الدالة **InStr** لتقطيع النص إلى كلمات منفصلة.

على الرغم من شيوع استخدام الدالة **Instr** في عمليات البحث داخل النصوص، إلا أنه يعاب عليها بدء ترقيم الحروف بالرقم **1** وليس **0** وهذا على عكس غالبية الدوال إن لم يكن كلها، لذا يفضل استخدام الدالة **IndexOf** بدلاً منها.

استخدام الدالة IndexOf

تعتبر الدالة **IndexOf** إحدى الدوال الجديدة في **Visual Basic 2008** والمستخدمة

فى البحث داخل النصوص. وتحتوى هذه الدالة على العديد من المعاملات إلا أنها تبدو فى أبسط صورها على الصيغة التالية:

```
variable = text1.IndexOf("text2")
```

حيث:

- يعبر **Variable** عن المتغير الذى يحتوى على مكان وجود النص المراد البحث عنه.
- يعبر **text1** عن النص الذى نرغب فى البحث فيه.
- يعبر **text2** عن النص الذى نرغب فى البحث عنه.

للتعرف على استخدام الدالة **IndexOf**، دعنا نرى الكود التالى:

```
Dim strSentence As String
Dim intFoundPosition As Integer
strSentence = "I will see you next friday"
intFoundPosition = strSentence.IndexOf("you")
If intFoundPosition < 0 Then
    Debug.WriteLine("I could not find you")
Else
    Debug.WriteLine("I found you at position " &
intFoundPosition)
End If
```

فى هذا الكود يتم البحث عن كلمة **"you"** داخل العبارة من خلال الدالة **IndexOf**. فإذا قامت الدالة بإرجاع قيمة أقل من الصفر وهو ما يعنى عدم وجود الكلمة بالعبارة يتم تنفيذ أحد العبارات وإلا يتم تنفيذ عبارة أخرى مع بيان مكان وجود النص. فنحصل على العبارة التالية:

```
I found you at position 11
```

معرفة طول نص

تستخدم الدالة **Len** لمعرفة طول نص (عدد حروفه) ويستخدم ذلك عادة للتأكد من عدم تجاوز النص لطول معين. وصيغة هذه الدالة كالتالي:

```
inlength=Len(stName)
```

لاحظ أن هذه الدالة تعطي عدد الحروف وليس الطول الفعلي الذي سيأخذه النص عند عرضه على الشاشة بأحد الخطوط مقاسا بوحدات قياس الأطوال على الشاشة مثل البوصة

أو البكسل Pixel، إذا أردت معرفة هذا الطول الفعلي استخدم الدالة TextWidthوإذا أردت معرفة الارتفاع الفعلي استخدم الدالة TextHeight.

يمكنك أيضاً استخدام الخاصية Length للحصول على طول سلسلة البيانات كما فى الكود التالى:

```
intValue = txtName.Length
```

التخلص من المسافات الزائدة

المسافات Spaces الزائدة هي المسافات التي توجد في أول أو آخر النص. وهي تؤدي إلى أن عمليات المقارنة بين نصين متشابهين تنتج عدم تشابه. فمثلاً العبارات الثلاثة التالية عند تنفيذها تعطي نتائج مختلفة رغم أن النص الفعلي واحد:

```
Debug.WriteLine (Len("    Computer"))
Debug.WriteLine (Len("  Computer   "))
Debug.WriteLine (Len("Computer    "))
```

لذلك يفضل التخلص من هذه المسافات الزائدة عند إجراء المقارنات وما إلى ذلك.

يتم التخلص من المسافات الزائدة بالدوال التالية:

- الدالة TrimStart: تمحو المسافات من بداية النص
- الدالة TrimEnd: تمحو المسافات من نهاية النص
- الدالة Trim: تمحو المسافات من بداية ونهاية النص

لهذه الدوال نفس الصيغة في الاستخدام، قم بتنفيذ الأمثلة التالية لتتبين تأثير هذه الدوال:

```
Debug.WriteLine (Trim("  Waleed   ") & " " & Trim("  Mohamed   "))
Dim st As String = ("   12 Salah Eddin st.   ")
st.Trim 'Remove spaces from left and right
st.TrimEnd 'Remove spaces from right only
st.TrimStart 'Remove spaces from left only
```

لا يقتصر استخدام الدوال Trim وTrimEnd على حذف المسافات فقط وإنما يمكنك استخدامها فى حذف الحروف الأخرى شريطة أن تكون فى بداية النص أو نهايته كما فى

الكود التالى:

```
MessageBox.Show("racecar".Trim("rac".ToCharArray))
```

وفيه يتم حذف الحروف "rac" من بداية النص ونهايته وبالتالى يتبقى الحرف "e" فقط.

تعتبر الدوال TrimStart وTrimEnd من الدوال الجديدة بالإصدار السابق من Visual Basic.

اجتثاث جزء من النص

الدوال الآتية تقوم بتكوين نص من نص آخر، وهو ما نعنيه بالاقتباس:

الدالة **Left** : ترجع عدد معين من الأحرف من بداية النص

الدالة **Right** : ترجع عدد معين من الأحرف من نهاية النص

الدالة **Mid** : ترجع عدد معين من الأحرف من وسط النص (بداية من حرف محدد)

الدالة **SubString**: ترجع عدد معين من الأحرف من أى مكان فى النص ويمكنك استخدام هذه الدالة كبديل للدوال الثلاثة السابقة.

الصيغة العامة لهذه الدوال كالآتي:

```
OutStr=Microsoft.VisualBasic.Left (InpStr, NumChars)
OutStr= Microsoft.VisualBasic.Right (InpStr, NumChars)
OutStr=Mid (InpStr, startpos [,NumChars])
OutStr=InpStr.SubString(startpos[,NumChar])
```

حيث OutStr هو النص الناتج ، InpStr هو النص الأصلي ، NumChars عدد الأحرف المراد اقتباسها، startpos هو ترتيب الحرف الذي يبدأ عنده الاقتباس.

الأمثلة الآتية توضح استخدام الدالة والقيمة التي ترجعها :

الصيغة	النص الذي تنشئه الدالة
Microsoft.VisualBasic.Left("This is My Name",4)	"This"
Microsoft.VisualBasic.Right("This is My Name",4)	"Name"
Mid("This is My Name",6,2)	"is"
"This is My Name".SubString(2,5)	"is is"
"This is My Name".SubString(2)	"is is My Name"

لاحظ أن NumChars إجباري في الدالتين الأوليين ولابد من أن يكون عدد أكبر من صفر، ولو ساوى صفر سنحصل على نص خالي (طوله صفر)، ولو زاد عن عدد الحروف في النص الأصلي سينتج النص الأصلي بأكمله.

أيضا startpos إجباري في الدالتين الآخرتين، ولابد أن يكون عدد أكبر من صفر، وإن كان بعدد أكبر من طول النص فسينتج النص الخالي (zero length string)، وNumChars في الدالتين اختيارى، وعند عدم تحديده سترجع الدالة الحروف بدءاً من startpos وحتى نهاية النص.

استبدال الأحرف داخل النص

هل استخدمت خاصية البحث والاستبدال Find and Replace من قبل في أي برنامج لمعالجة النصوص؟ لو قمت بذلك – وهذا شبه أكيد– فإن الدالة Replace() تصبح سهلة التناول للغاية، فهي المكافئ البرمجي لهذه الخاصية في برامج تحرير النصوص.

كمثال مبدأي لنفرض أن لديك سلسلة بيانات طويلة كالآتي:

```
strDocument= "This is the first line in the document..."
```

لاحظ أن هذه السلسلة قد تكون محتويات مربع نصوص.

لو أنك تريد في برنامجك أن تستبدل أي نص جزئي مثل "first" بنص جزئي آخر، وليكن "second" في كافة السلسلة يمكنك أن تستخدم الدالة الجديدة Replace() كالتالي:

```
strResult=Replace(strDocument, "first" , "second")
```

السلسلة الناتجة بعد التغيير هي strResult فنحصل على العبارة التالية:

```
This is the second line in the document...
```

الصورة السابقة في الاستخدام هي الصورة البسيطة، أما الصيغة الكاملة للدالة Replace فهي:

```
strResult=Replace(strSource, strFind, strReplace, intStart,
intCnt, intCompare)
```

حيث المعاملات للدالة كالتالي:

strSource: النص المطلوب البحث فيه

strFind: النص المطلوب البحث عنه

strReplace: النص المطلوب الاستبدال به

intStart: ترتيب الحرف المراد بدء البحث منه (في النص **strSource**)

intCnt : أقصى عدد لمرات الاستبدال

intCompare : قيمة رقمية (**0** أو **1**) تحدد هل يتم الالتزام بحجم الحروف **Capital** أو **small** عند المقارنة (في الحروف اللاتينية فقط).

وترجع الدالة **strResult** وهي سلسلة حرفية تحتوي على النص بعد الاستبدال، لو استخدمت المعامل **intStart** فإن السلسلة الناتجة تبدأ من هذا الحرف المحدد داخل **intStart**.

بالنسبة للمعامل **intCompare**، فهو يحدد أسلوب المقارنة كما فى جدول ١٤-٦ التالي:

جدول ١٤-٦ قيم المعامل **intCompare**.

الثابت	القيمة	التوضيح
CompareMethod.Binary	0	يتم أخذ حالة الحرف Case (Capital or small) في الاعتبار ،أي أن 'a' مثلا لا تكافئ 'A' في هذه الحالة
CompareMethod.Text	1	يتم تجاهل حالة الحرف Case (Capital or small) عند المقارنة، أي أن 'a' تكافئ 'A' في هذه الحالة

لا يقتصر استخدام الدالة على البحث فقط، وإنما يمكنك استخدامها في إزالة نصوص أو حروف معينة، بأن يتم تحديد نص الدالة **Replace** خالي ("") في معامل **strReplace**. ونظرا لأن عملية الاستبدال عملية ليست بسيطة، فإن الدالة **Replace** قد تعطي في سلسلة البيانات الناتجة رسالة خطأ أو دليل عليها وليس سلسلة حرفية صحيحة، يوضح جدول ١٤-٧ التالي هذه الحالات:

جدول ١٤-٧ الحالات الخاصة للدالة Replace

الحالة	القيمة المرجعة strResult
StrSource نص خالي	نص خالي ""
StrSource=Null	Error (يتوقف البرنامج)
strFind نص خالي	نسخة من strSource
strReplace نص خالي	نسخة من strSource مع محو كل strFind يتم إيجاده
IntStart>len(strSource)	نص خالي ""
intCnt=0	نسخة من strSource

في الحقيقة هذه الدالة بها الكثير من المعاملات، لذا ستجد في المشروع NewString نموذج دسم بالعناصر لتوضيح كافة معاملات هذه الدالة، كما يبدو عند استخدامه في شكل ١٤-٣.قم بتجربة كافة البدائل لكل اختيار، لكي تدرك تأثير المعاملات المختلفة على الدالة.

شكل ١٤-٣ برنامج NewString، عند استخدامه لاستبدال نص في سلسلة حرفية.

تستخدم الدالة Mid أيضاً لحذف جزء من وسط نص، نفس الاسم يستخدم لاستبدال جزء من نص بآخر، ولكن في هذه الحالة لا نستخدم الدالة Mid ولكن العبارة Mid وصيغة

الاستخدام مختلفة قليلاً، فالعبارة لن ترجع نص آخر وإنما ستقوم بتغيير في نص موجود.

الصيغة العامة لهذه العبارة كالتالي:

```
Mid (sourcestr, startpos [, numchars]) = replstr
```

حيث **replstr** هو النص المراد وضعه بدل آخر موجود في النص الأصلي. وبقية المعاملات سبق وأوضحناها.

مثال على العبارة السابقة:

```
strN= "My Name Is Hassan"
Mid (strN,12) = "Salem"
Debug.WriteLine(strN)   ' --- gives the string "My Name Is Salem"
```

عكس ترتيب الحروف في سلسلة بيانات

تتيح لك الدالة الجديدة **StrReverse()** أن تعكس ترتيب الحروف في سلسلة حرفية بالكامل، مثلا السلسلة **"This is a Computer"** تصبح **"retupmoC a si sihT"**

بعد استخدام هذه الدالة، صيغة استخدام الدالة السابقة هي:

```
strResult = StrReverse(strsource)
```

حيث:

- **strSource** هي السلسلة الأصلية.
- **strResult** هي السلسلة الناتجة بعد العكس.

لكي ترى تأثير هذه الدالة قم باستخدام النموذج **frmReverse** داخل المشروع **Strings**، وأدخل جملة في المكان المخصص ثم اضغط زر التنفيذ، لترى النص الناتج بعد تنفيذ الدالة، كما ترى في شكل ١٤-٤.

شكل ١٤-٤ دالة **strReverse()** عند تطبيقها على سلسلة حرفية في برنامج **Strings**.

تجزئة سلسلة بيانات إلى مصفوفة من سلاسل البيانات

تستقبل الدالة **Split()** سلسلة بيانات ثم تقوم بتقطيعها إلى كلمات منفصلة، يعتمد هذا بالطبع على معرفة الفواصل التي تفصل بين الكلمات في السلسلة الأصلية. فلو كانت هذه السلسلة مثلا عبارة عن جملة، فإن الفاصل عادةً ما يكون المسافة **Space**، ولو كانت سجل من قاعدة بيانات فإن الفاصل عادةً يكون الفاصلة العادية **Comma** ','.تتيح **Split()** تعريف فاصل **Separator** آخر، يتم التجزئة عندها إذا وجدت، ولن ترجع الدالة بالطبع هذه الفواصل مع الكلمات الناتجة بعد التجزئة.

الصيغة العامة للدالة **Split** كالتالي:

strResult=Split(strList,strDelimeter,intElemCnt, intCompare)

حيث:

- **strResult**: هي المصفوفة الناتجة.
- **strList**: هو سلسلة البيانات المراد تجزئتها
- **strDelimiter**: هو رمز يؤخذ كفاصلة تستخدم في تجزيء السلسلة، القيمة الافتراضية له هو الرمز **Space** .
- **intElemCnt** : هو العدد الأقصى للعناصر في المصفوفة الناتجة
- **intCompare** : هو عدد يحدد عملية مقارنة الفاصل **delimiter** هل تتم مع أخذ حالة الحرف (**Capital or Small**) في الاعتبار أم لا(إن كان من الحروف الأبجدية اللاتينية) وهي تأخذ أحد القيم التالية:

الثابت	القيمة	التوضيح
CompareMethod.Binary	0	يتم أخذ حالة الـ **delimiter** في الاعتبار
CompareMethod.Text	1	لا يتم أخذ حالة الـ **delimiter** في الاعتبار

إن هذه الدالة القوية **Split()** والتي ترجع مصفوفة من السلاسل الحرفية الجزئية، تفتح الطريق إلى اثنين من الدوال القوية الجديدة أيضا وهما: **Filter()** و**Join()** الآتيتين.

يوضح شكل ١٤-٥ النموذج **frmConversion** داخل المشروع **Strings**، تجد مربع

نص، قم بكتابة ما تشاء فيه، وحدد الفاصل، ثم اضغط زر "قم بالتجزئة"، يظهر الناتج أمامك في مربع السرد الأيمن.

شكل ١٤-٥ النموذج frmConversion عند استخدامه في تجزيء سلسلة حرفية تحوي أيام الأسبوع.

تصفية مصفوفة من السلاسل الحرفية

بعد تجزئة السلسلة الحرفية إلى عناصر في مصفوفة حرفية String Array، قد تحتاج إلى اختيار عناصر معينة من هذه المصفوفة بناءاً على وجود أحرف أو نص معين فيها، الدالة الجديدة Filter تقوم بإرجاع مصفوفة جديدة تحتوي على العناصر التي يتحقق فيها الشرط الذي تحدده فقط.

الصيغة العامة لهذه الدالة كالتالي:

```
strResult = Filter( strList, strFind, bolInclude, intCompare)
```

حيث:

- **strResult**: مصفوفة تحتوي على العناصر التي يوجد بها النص المحدد strFind من المصفوفة strList.
- **strList**: المصفوفة التي يتم البحث في عناصرها.
- **strFind**: النص المراد البحث عنه في مصفوفة السلاسل الحرفية strList.
- **bolInclude**: متغير منطقي (نعم/ لا) يحدد هل يتم تكوين المصفوفة الجديدة

من العناصر التي وجد بها النص **strFind** (نعم)، أم من العناصر التي لا تحتوي على هذا النص (لا).

- **intCompare**: عدد يبين طريقة البحث عن **strFind** في **strList**، هل يهمل حالة الحرف (**Case**) أم يأخذها في الاعتبار (راجع الدالة **Split** لمعرفة القيم المختلفة لهذا المعامل).

لو لم تجد الدالة أي عنصر يحتوي على النص **strFind** فإنها ترجع مصفوفة خالية (لو كان **bolInclude=True**)، أما لو كانت المصفوفة **strList** نفسها خالية أو مصفوفة متعددة الأبعاد **multi-dimensional**، فإن الدالة تصدر خطأ.

لو اتخذنا المصفوفة التي أنشأناها في المثال السابق كمثال والتي تحتوي على أيام الأسبوع، يمكننا أن نبحث مثلا عن أي عنصر يحتوي على حرف "ث" مثلا، كما ترى في شكل ١٤-٦).

شكل ١٤-٦ استخدام الدالة **Filter** لاختيار عناصر معينة من مصفوفة سلاسل حرفية.

دمج عناصر مصفوفة حرفية

الدالة **Join** هي المقابلة في الوظيفة للدالة **()Split**، فإن كانت **()Split** تقوم بتجزئه سلسلة حرفية إلى سلاسل صغيرة وترجع مصفوفة تحتوي على هذه السلاسل، فإن **()Join**

تقوم بإنشاء سلسلة حرفية من عناصر مصفوفة.

الصيغة العامة لهذه الدالة كالتالي:

strResult = Join (strList, strDelimiter)

حيث:

- **strList**: هو المصفوفة المحتوية على العناصر المراد دمجها.
- **strDelimiter**: هو الرمز المستخدم كفاصل يوضع بين العناصر عند تكوين السلسلة الحرفية.

strResult: هي السلسلة الحرفية الناتجة بعد دمج عناصر المصفوفة.

كامتداد للمثال السابق، يمكننا أن ندمج عناصر المصفوفة ثانية، ولكن مع استخدام فاصل مختلف ، وليكن "--" ، كما ترى في شكل ١٤-٧.

شكل ١٤-٧ استخدام **Join** لدمج المصفوفة من جديد مع تغيير الفاصل.

يحتوى البرنامج **Strings** على أكثر من نموذج. فإذا أردت تجربة عمل أحد النماذج، اجعل هذا النموذج هو نموذج بدء تشغيل البرنامج من خلال نافذة الخصائص الخاصة بالمشروع.

التعامل مع الأحرف الخاصة

هناك بعض الأحرف التي لا يمكن كتابتها وإن كان لها أثر واضح في النصوص عموماً،مثل حرف البدء بسطر جديد.

وللتعبير عن هذه الأحرف التي لا تظهر ضمن الحروف الأبجدية والرموز المعروفة نستخدم الدالة Chr(X) حيث X كود يعبر عن الحرف المراد كتابته بالنظام ASCII وتتراوح قيمته من ٠ إلى ٢٥٥ والتي تتضمن الأحرف والرموز الأخرى المستخدمة مثل > ،< وغيرها،و أيضاً تتضمن كود الأحرف الخاصة مثل Backspace والسطر الجديد وخلافه.
مثلا السطر التالي يستخدم هذه الدالة لإدراج رموز البدء بسطر جديد في أحد المتغيرات الحرفية:

```
MessageBox.Show(Chr(13) & Chr(10))
```

حيث الرمزان 10 (NewLine) و13 (Carriage Return) يسببا الفصل بين السطور.
يوضح الجدول ١٤-٨ التالي بعض الأحرف الخاصة التي لا يمكن إدخالها في النص من خلال لوحة المفاتيح:

جدول ١٤-٨ أكواد بعض الأحرف الخاصة

الكود	ما يماثله
٨	مفتاح التراجع Backspace
٩	مفتاح Tab
١٠	مفتاح الانتقال إلى سطر جديد
١٣	مفتاح بدء من أول السطر
٣٢	مسافة
٣٤	علامة الحذف المزدوجة (")
٤٨	رمز الرقم (0)
٦٥	رمز حرف (A)
٩٧	رمز حرف (a)

وهناك دالة تقوم بعكس ما تقوم به الدالة Chr() وهي الدالة Asc (" ") ويتم إعطاء هذه الدالة الحرف فتعطي الدالة الكود الخاص بهذا الحرف فمثلا عند كتابة Asc("A") تعطينا القيمة ٦٥.

التعامل مع الحروف والأرقام

كثيراً ما نجد أرقام يتم التعامل معها على أنها مدخلات نصية وليست أعداد. فمثلاً الرقم البريدي لا يتم التعامل معه كقيمة قابلة للجمع والطرح ولكن كود يعبر عن منطقة معينة. وكثيراً ما نحتاج إلى تحويل مدخل ما من الصورة النصية إلى الصورة العددية أو العكس. لهذا الغرض نستخدم الدوال الآتية:

الدالة	الاستخدام
Str(x)	لتحويل المتغير الرقمي "x " إلى الصيغة النصية
Val(x)	لتحويل المتغير النصي(المكون من أرقام) "x" إلى الصيغة الرقمية

أمثلة لاستخدام هذه الدوال:

الصيغة	الناتج
Str(123)	"123" (ثابت حرفي)
Val("123")	123 (عدد صحيح)
Val("123 Kg")	123 (يتم إهمال الحروف غير الرقمية)
Val("Street")	0