الفصل العاشر
المعاملات ودوال السلاسل الحرفية والتاريخ

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

- *المعاملات الحسابية*
- *معاملات ربط (وصل) العبارات*
- *المعاملات العلائقية*
- *المعاملات المنطقية*
- *أولويات تنفيذ المعاملات*
- *استخدام دوال السلاسل الحرفية*
- *التعامل مع التاريخ والوقت*

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 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 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 .Write Line(3.12 + 12)
```

وعند تنفيذ هذا الأمر ستحصل على الناتج 15.12 داخل النافذة Immediate Window بالجزء السفلى من نافذة بيئة التطوير (لإظهار النافذة الفورية Immediate Window إن لم تكن ظاهرةً بالفعل، اضغط الاختصار Ctrl+ G بلوحة المفاتيح أو افتح قائمة Debug ثم اختر Immediate من القائمة الفرعية Windows).

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

```
Dim A = 5: Dim B = 6.5
Debug .Write Line (A +B)
Debug .Write Line (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).

```
Dim A = 10: Dim B = 8.5
Debug .Write Line (A- B)
Debug .Write Line (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).

عمليتاالضرب والأسMultiplication / 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 .Write Line (2*3)
Debug .Write Line (2 ^ 3)
Debug .Write Line (1.5 * 7)
Debug .Write Line (3.4 ^ 1.29)
```

ستحصل على الناتج التالى (على الترتيب):

```
6
8
10.5
4.84850449955586
```

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 Basicعمليتين 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 2012** استخدام الصيغ المختصرة فى العمليات الحسابية. يوضح جدول ١٠-١ التالى الصيغة المختصرة للعمليات الحسابية المختلفة.

جدول ١٠-١الصيغ 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 ١٠-٢التالى المعاملات العلائقية وأمثلة على استخدامها.

جدول ١٠-٢المعاملات 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 ١٠-٣التالى يبين وظيفة كل عامل من العوامل المنطقية.

جدول ١٠-٣ 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 ١٠-٤ 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 ١٠-٤ 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

المعامل	معناه	الترتيب
And	Conjunction "و"	١٠
OR	Inclusive OR "أو"	١١
XOR	Exclusive OR "أو المقتصرة"	١٢
EqV	Equivalence "تكافؤ"	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 التالى:

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 الرقميةوالسلاسل الحرفية. قد لا تشعر بأهمية السلاسل الحرفية في الوقت الحاضر لو كنت من المبرمجين المبتدئينولكن مع الوقت ستدرك أهميتهاوتحتاج 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 الوظائف (الدوال)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 الوظائف (الدوال) التي تتعامل مع السلاسل الحرفية إلى عدة فئات:

- منها ما يقوم بحذف أو استقطاع جزء من السلسلة الأصليةويكون بها سلسلة جديدة، يشمل الوظائف: **()Left** و**()Right** و**()Mid**و**()SubString**.
- ومنها ما يقوم باستخراج سلسلة حرفية من أخرى بعد حذف المسافات الخالية **Spaces** وتشتمل على: **()Trim**و**()TrimEnd** و**()PadLeft**.
- ومنها الذي يقوم بالتحويل بين الأحرف اللاتينية الكبيرةوالصغيرةوتضم **()ToUpper** و**()ToLower** و**()UCase**و**()LCase**.
- وهناك الوظيفة**()InStr**والوظيفة**()IndexOf**التي تبحث في سلسلة حرفية عن سلسلة أخربوتعطي مكانها.والوظيفة**()Len**أو الخاصية **Length**التي تعطي طول السلسلة الحرفية (عدد حروفها).
- وأخيراً الوظيفة**()Replace** المستخدمة فى استبدال جزء داخل نص بجزء آخر والوظيفة**()strReverse** المستخدمة لعكس النص وغيرها من دوال تفكيك النص وتجميعه.

ونوضح في الفقرات التالية كيفية استخدام هذه الدوال.

تغيير حالة الأحرف

هناك أربع وظائفتستخدم في تغيير حالة الأحرف اللاتينية وهى**()ToUpper**و**()UCase**و**()ToLower**و**()LCase**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 الوظيفة، 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 الوظيفة**()ToUpper** بدلاً من الوظيفة**()UCase** ليصبح الكود السابق كما يلى:

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

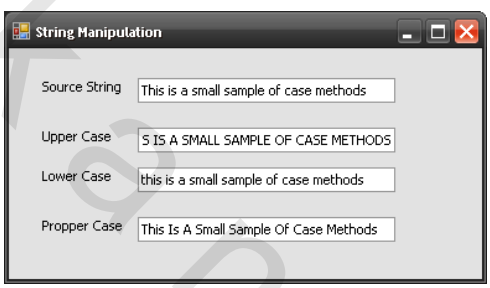
وبنفس الطريقة يمكنك استخدام الوظيفتين**()LCase** و**()ToLower**.

من الوظائف الأخرى المستخدمة في تغيير حالة الأحرف، الوظيفة**()StrConv** وهي

تستخدم في تحويلات متعددة منها التحويل بين الأحرف الكبيرةوالصغيرة، ومنها تحويلات لا تهمنا مثل تحويلات بين حروف يابانية.ولكن أحد التحويلات الممكن القيام بها يستحق الذكروهو تحويل الأحرف إلى "الحالة المناسبة" **Proper Case** ، أي تحويل الحرف الأول من الكلمة إلى الشكل الكبيروتحويل بقية الأحرف إلى الشكل الصغير في كافة الكلمات في النص. مثال لذلك في السطر التالي :

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

يوضح شكل ١٠-١ التالى التحويلات المختلفة للأحرف.



شكل ١٠-١ تأثير التحويلات المختلفة على حالة الأحرف.

حذفوإضافة 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 تقطيعهوالحصول على كل معلومة على انفراد. أول الوظائفالمستخدمة في البحث عن نص داخل نص آخر هي الوظيفة**InStr()**.

صيغة استخدام هذه الوظيفة كالتالي:

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

والشكل البسيط لاستخدامها أن نحدد النص الذي سنبحث فيه SourceStr والذي سنبحث عنه SearchStr وسترجع لنا الوظيفة 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 الوظيفة 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 الوظيفة عند البحث عن m القيمة ٦ إذا بدأت البحث من أول حرف. أما عند البدء من الحرف السابع فلن ترى الحرف السادسولذا سترجع القيمة 14وهو أول حرف m تالي.

هناك معامل اختياري آخر لهذه الوظيفةوهو 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 الوظيفة في عملية "تقطيع" نص إلى كلماته منفردة. يتضمن ذلك بحث متكرر عن الحروفواستقطاع للكلمات:

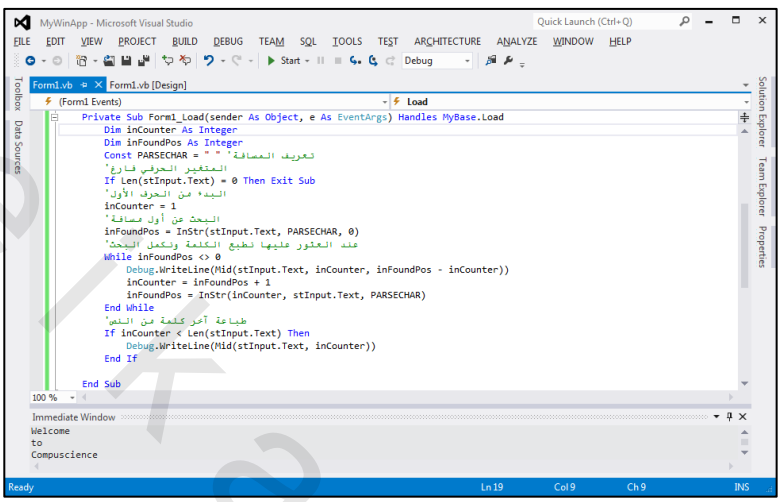
```
Dim inCounter As Integer
Dim inFoundPos As Integer
Const PARSECHAR = " " 'تعريفالمسافة
'المتغيرالحرفيفارغ
If Len(stInput.Text) = 0 Then Exit Sub
'البدءمنالحرفالأول
inCounter = 1
'البحثعنأولمسافة
inFoundPos = InStr(stInput.Text, PARSECHAR, 0)
'عندالعثورعليهانطبعالكلمةونكملالبحث
While inFoundPos <> 0
Debug.WriteLine(Mid(stInput.Text, inCounter, inFoundPos -
inCounter))
inCounter = inFoundPos + 1
inFoundPos = InStr(inCounter, stInput.Text, PARSECHAR)
End While
'طباعةآخركلمةمنالنص
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 **ImmediateWindow** (انظر شكل

.(٢-١٠



شكل ١٠-٢ استخدام الوظيفة**InStr()** لتقطيع النص إلى كلمات منفصلة.

على الرغم من شيوع استخدام الوظيفة()Instr فى 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 الوظيفة()IndexOf بدلاً منها.

استخدام الوظيفة()IndexOf

تعتبر الوظيفة()IndexOf إحدى الوظائف التى ظهرت بالإصدار السابق من Visual Basic والمستخدمة فى البحث داخل النصوص. وتحتوى هذه الوظيفة 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 الوظيفة**()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 الوظيفة**()IndexOf**. فإذا قامت الوظيفة 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 الوظيفة**()Len** لمعرفة طول نص (عدد حروفه)ويستخدم ذلك عادة للتأكد من عدم تجاوز النص لطول معين.وصيغة هذه الوظيفة كالتالي:

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

لاحظ أن هذه الوظيفة تعطي عدد الحروفوليس 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 الوظيفة**()TextWidth**وإذا أردت معرفة الارتفاع الفعلي استخدمالوظيفة**()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 المقارناتوما إلى ذلك.

يتم التخلص من المسافات الزائدة بالدوال التالية:

- الوظيفةTrimStart(): تمحو المسافات من بداية النص
- الوظيفةTrimEnd(): تمحو المسافات من نهاية النص
- الوظيفةTrim(): تمحو المسافات من بدايةونهاية النص

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

```
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 الوظائفTrim() 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.

اجتثاث جزء من النص

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

الوظيفة()Left : ترجع عدد معين من الأحرف من بداية النص

الوظيفة()Right : ترجع عدد معين من الأحرف من نهاية النص

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

الوظيفة()SubString: ترجع عدد معين من الأحرف من أى مكان فى النص ويمكنك استخدام هذه الوظيفة كبديل للوظائف الثلاثة السابقة.

الصيغة العامة لهذه الوظائف 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، startposهو ترتيب الحرف الذي يبدأ عنده الاقتباس.

الأمثلة الآتية توضح استخدام الوظيفةوالقيمة التي ترجعها:

الصيغة	النص الذي تنشئه الوظيفة
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 الوظيفتين الأوليينولابد من أن يكون عدد أكبر من صفر،ولو ساوى صفر سنحصلعلى نص خالي (طوله صفر)،ولو زاد عن عدد الحروف في النص الأصلي سينتج النص الأصلي بأكمله.

أيضا startpos إجباري في الوظيفتين 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 الوظيفة 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 الوظيفةReplace()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 الوظيفة 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 للوظيفةReplace() فهي:

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

حيث المعاملات للوظيفة 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 الوظيفة**()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 ١٠-٥التالي:

جدول ١٠-٥ 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 الوظيفة على البحث فقط، وإنما يمكنك استخدامها في إزالة نصوص أو حروف معينة، بأن يتم تحديد نص الوظيفة**()Replace** خالي("") في معامل **strReplace**.

ونظرا لأن عملية الاستبدال عملية ليست بسيطة، فإن الوظيفة**()Replace** قد تعطي في سلسلة البيانات الناتجة رسالة خطأ أو دليل عليها وليس سلسلة حرفية صحيحة،يوضح جدول ١٠-٦التالي هذه الحالات:

جدول ١٠-٦ الحالات الخاصة للوظيفة**()Replace**

الحالة	القيمة المرجعة **strResult**
StrSource نص خالي	نص خالي ""

الحالة	القيمة المرجعة strResult
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 الوظيفة بها الكثير من المعاملات، لذا ستجد في المشروع Stringsالمرفق نموذج دسم بالعناصر لتوضيح كافة معاملات هذه الوظيفة، كما يبدو عند استخدامه في شكل ١٠-٣.قم بتجربة كافة البدائل لكل اختيار، لكي تدرك تأثير المعاملات المختلفة على الوظيفة.

شكل ١٠-٣ برنامج Strings، عند استخدامه لاستبدال نص في سلسلة حرفية.

تستخدم الوظيفةMid()أيضاً لحذف جزء من وسط نص، نفس الاسم يستخدم لاستبدال جزء من نص بآخر، ولكن في هذه الحالة لا نستخدم الوظيفةMid() ولكن العبارة Mid وصيغة الاستخدام مختلفة قليلاً، فالعبارة لن ترجع نص آخروإنما 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 سبقوأوضحناها.

مثال على العبارة السابقة:

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

عكس ترتيب الحروف في سلسلة بيانات

تتيح لك الوظيفة الجديدة **StrReverse()** أن تعكس ترتيب الحروف في سلسلة حرفية بالكامل، مثلا السلسلة "**This is a Computer**" تصبح "**retupmoC a si sihT**" بعد استخدام هذه الوظيفة، صيغة استخدام الوظيفة السابقة هي:

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

حيث:

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

لكي ترى تأثير هذه الوظيفة قم باستخدام النموذج **frmReverse** داخل المشروع**Strings**، وأدخل جملة في المكان المخصص ثم اضغط زر التنفيذ، لترى النص الناتج بعد تنفيذ الوظيفة، كما ترى في شكل ١٠-٤.

شكل ١٠-٤ دالة ()strReverse عند تطبيقها على سلسلة حرفية في برنامجStrings.

تجزئة سلسلة بيانات إلى مصفوفة من سلاسل البيانات

تستقبل الوظيفة()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 الوظيفة بالطبع هذه الفواصل مع الكلمات الناتجة بعد التجزئة.

الصيغة العامة للوظيفة()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 يحددعملية 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:

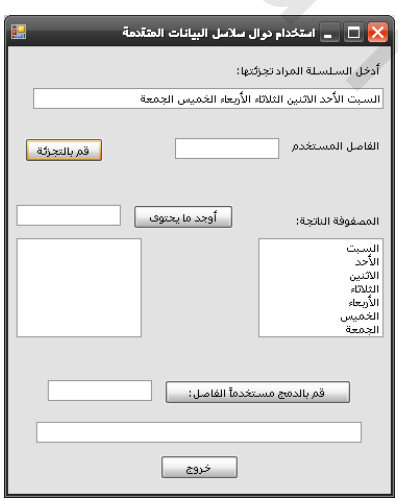
الثابت	القيمة	التوضيح
CompareMethod.Binary	0	يتم أخذ حالة الـ delimiter في الاعتبار
CompareMethod.Text	1	لا يتم أخذ حالة الـ delimiter في الاعتبار

إن هذه الوظيفة القوية Split() والتي ترجع مصفوفة من السلاسل الحرفية الجزئية، تفتح

تعمل المصفوفة Array بطريقة مشابهة لطريقة عمل المتغير إلا أنها تحتوى على عدد من العناصر المتماثلة بدلاً من عنصر واحد فى حالة المتغير، حيث يتم استخدام الرمزين ( ) لتمييز المتغير عن المصفوفة. وسوف نتناول المصفوفات بالتفصيل بالفصل الثانى عشر إن شاء الله.

الطريق إلى اثنين من الوظائف القوية أيضا وهما: Filter() وJoin() الآتيتين.

يوضح شكل ١٠-٥النموذج frmConversion داخل المشروع Strings، تجد مربع نص، قم بكتابة ما تشاء فيه، وحدد الفاصل، ثم اضغط زر "قم بالتجزئة"، يظهرالناتج أمامك في مربع السرد الأيمن.



شكل ١٠-٥النموذج 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، تقوم الوظيفةFilter() بإرجاع مصفوفة جديدة تحتوي على العناصر التي يتحقق فيها الشرط الذي تحدده فقط.

الصيغة العامة لهذه الوظيفة كالتالي:

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

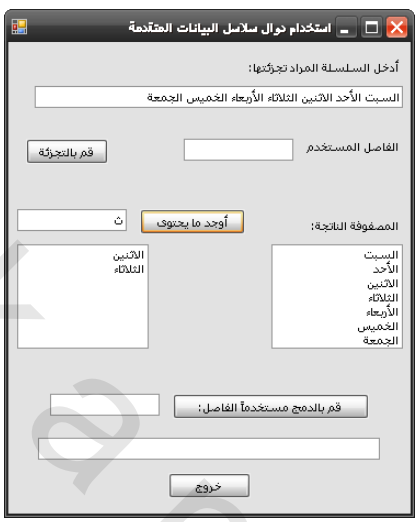
حيث:

- **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 الوظيفةSplit() لمعرفة القيم المختلفة لهذا المعامل).

لو لم تجد الوظيفة 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 الوظيفة 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 ١٠-٦).



شكل ١٠-٦ استخدام الوظيفة**Filter()** لاختيار عناصر معينة من مصفوفة سلاسل حرفية.

دمج عناصر مصفوفة حرفية

الوظيفة**Join()** هي المقابلة في وظيفتها للوظيفة **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.

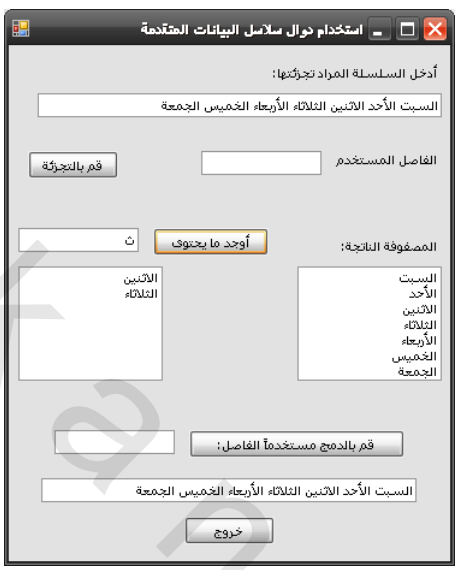
الصيغة العامة لهذه الوظيفة كالتالي:

```
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 ١٠-٧.



شكل ١٠-٧ استخدام الوظيفة**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 كتابتهاوإن 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 الوظيفة**Chr(X)** حيث **X** كود يعبر عن الحرف المراد كتابته بالنظام **ASCII** وتتراوح قيمته من ٠ إلى ٢٥٥ والتى تتضمن الأحرف والرموز الأخرى المستخدمة مثل > ،< وغيرها،و أيضاً تتضمن كود الأحرف الخاصة مثل **Backspace** والسطر الجديد وخلافه.

مثلا السطر التالي يستخدم هذه الوظيفة 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 ١٠-٧التالي بعض الأحرف الخاصة التي لا يمكن إدخالها في النص من خلال لوحة المفاتيح:

جدول ١٠-٧ 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 وظيفة تقوم بعكس ما تقوم به الوظيفة Chr() وهى الوظيفة(“”) Asc ويتم إعطاء هذه الوظيفة الحرف فتعطي الوظيفة 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 الوظائف الآتية:

الوظيفة	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 الوظائف:

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

التعامل مع التاريخ والوقت

التواريخ والأوقات ليست أرقام، وليست سلاسل نصية. ولكي تتعامل مع التواريخ والأوقات في Visual Basic، فأنت بحاجة لاستخدام دوال خاصة بالتاريخ. وباستخدام دوال التاريخ الخاصة، يمكنك جمع فترة زمنية (أيام، أسابيع، ساعات... الخ)، وتحديد فترة زمنية تقع بين تاريخين أو وقتين، وكذلك معرفة الوقت أو التاريخ في أى لحظة حسب ساعة حاسبك.

استرجاع تاريخ النظام الحالي ووقته

يوجد في Visual Basicوظيفتين لإرجاع معلومات عن الوقت والتاريخ الحاليين للنظام وهما الوظيفة()Now التى تقوم بإرجاع التاريخ والوقت الحاليين والوظيفة()Today التى تقوم بإرجاع التاريخ الحالى ووقت منتصف الليل كما فى الكود التالى:

```
Dim dt As Date = Now()
Debug.WriteLine(Date.Now)
Debug.WriteLine(Date.Today)
```

```
Debug.WriteLine(dt)
```

وخرج هذا الكود يكون كما فى شكل ١٠-٨ التالى.

شكل ١٠-٨ الحصول على التاريخ والوقت الحاليين.

وكما ترى فإننا نحصل من الوظيفتين على التاريخ والوقت كاملين، لكن ماذا لو أردنا الحصول على جزء من التاريخ أو جزء من الوقت مثل اليوم فقط أو الساعة فقط أو الشهر فقط ؟ فى هذه الحالة يمكنك استخدام الخصائص المصاحبة لكل دالة للحصول على الجزء المطلوب كما فى العبارة التالية:

```
Debug.WriteLine(Now.DayOfWeek)
```

التى تقوم بإرجاع اسم اليوم الموجود بالتاريخ الحالى. يوضح الجدول ١٠-٨ التالى الخصائص التى يمكنك استخدامها للحصول على أجزاء من التاريخ والوقت.

جدول ١٠-٨ الخصائص المستخدمة للحصول على أجزاء معينة من التاريخ والوقت

اسم الخاصية	القيمة المرجعة	نوع البيانات الناتج
Date	التاريخ فقط	Date
TimeofDay	الوقت فقط	TimeSpan
Month	الشهر من ١ إلى ١٢	Integer
Day	اليوم من ١ إلى ٣١	Integer
Hour	الساعة من ٠ إلى ٢٣	Integer
Minute	الدقيقة من ٠ إلى ٥٩	Integer
Second	الثانية من ٠ إلى ٥٩	Integer
Millisecond	الملى ثانية من ٠ إلى ٩٩٩	Integer
DayOfWeek	اليوم فى الأسبوع من ٠ إلى ٦	Integer

اسم الخاصية	القيمة المرجعة	نوع البيانات الناتج
DayOfYear	اليوم فى السنة من ١ إلى ٣٦٦	Integer

زيادة التاريخ أو الوقت

لإضافة فترة زمنية لتاريخ أو وقت محدد، استخدم الوظيفة**DateAdd()**.وهذه الوظيفة تأخذ الشكل التالي:

DateAdd (Inteval, Number,DateValue)

حيث:

- **Interval**: هي وسيطة الفترة الزمنية (**Argument**) المطلوب إضافتها
- **Number**: عدد السنين أو الشهور أو الأيام...الخ المطلوب إضافتها للتاريخ المحدد
- **DateValue**: التاريخ المطلوب إضافة فترة زمنية إليه.

وسيطة الفترة الزمنية (**Argument**) عبارة عن ثابت أو متغير أو تعبير يمرر إلى الإجراء.

يعرض الجدول ١٠-٩التالي القيم الممكنة للفترة الزمنية. ويجب أن تضع وسيطة الفترة الزمنية بين علامتي اقتباس أو بدلاً من ذلك يمكنك اختيار الثوابت التى تظهر أثناء كتابتك لكود الوظيفة.

جدول ١٠-٩ ضبط الفترة الزمنية للوظيفة**DateAdd()** والوظيفة**DateDiff()**

الضبط	التوصيف	الثابت
YYYY	سنة	DateInterval.Year
Q	ربع سنة	DateInterval.Quarter
M	شهر	DateInterval.Month
Y	يوم من السنة	DateInterval.DayofYear
D	يوم	DateInterval.Day

الضبط	التوصيف	الثابت
W	أيام أسبوع	DateInterval.DayofWeek
WW	أسبوع	DateInterval.Week
H	ساعة	DateInterval.Hour
N	دقيقة	DateInterval.Minite
S	ثانية	DateInterval.Second

لتوضيح استخدام هذه الوظيفة، دعنا نرى الكود التالى:

```
Dim dt As Date = Now()
dt = DateAdd(DateInterval.Day, 2, Now)
Debug.WriteLine(dt)
dt = DateAdd(DateInterval.Month, 2, Now)
Debug.WriteLine(dt)
dt = DateAdd(DateInterval.Year, 2, Now)
Debug.WriteLine(dt)
```

إذا قمت بتنفيذ هذا الكود، تحصل على نتيجة مشابهة لشكل ١٠–٩ التالى.

شكل ١٠–٩ نافذة استخدام الوظيفة**DateAdd()**.

تحديد الفترة الزمنية بين تاريخين أو وقتين

يمكنك بسهولة حساب الفترة الزمنية بين تاريخين أو وقتين بواسطة الوظيفة**DateDiff()** . والشكل العام للوظيفة**DateDiff()** كالتالي:

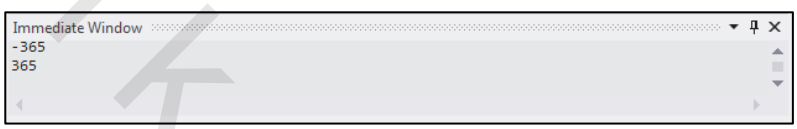
```
DateDiff(interval, Date1, Date2)
```

تقبل **Interval** (وسيطة الفترة الزمنية) نفس القيم بالضبط كالوظيفة السابقة **DateAdd()** (انظر الجدول ١٠–٨ السابق) و**Date1** هو التاريخ المطروح بينما **Date2** هو التاريخ المطروح منه. الوظيفة**DateDiff()** ترجع رقما من النوع **Long**يشير إلى عدد الفترات الزمنية المحددة بين تاريخين (بالسنة ، أو الشهر ، أو اليوم ...الخ).

لتوضيح وشرح الوظيفة**DateDiff()**دعنا نرى الكود التالى:

```
Dim Lvar As Long
Dim dt1 As Date = "12 - 12 - 2003"
Dim dt2 As Date = "12 - 12 - 2002"
Lvar = DateDiff(DateInterval.Day, dt1, dt2)
Debug.WriteLine(Lvar)
Lvar = DateDiff(DateInterval.Day, dt2, dt1)
Debug.WriteLine(Lvar)
```

إذا قمت بتنفيذ هذا الكود، تحصل على نتيجة مشابهة لشكل ١٠-١٠ التالى.



شكل ١٠-١٠استخدام الوظيفة() Date Diff

لاحظ في الأمثلة السابقة أن عبارة الطباعة الأولى ترجع رقما سالبا، وذلك لأن التاريخ الثانى أحدث من التاريخ الأول، أى أن الفترة المحددة لم تمر بعد. وهذا الأمر سيكون مفيدا في تحديد ترتيب تاريخين لأنك بهذه الطريقة تستطيع مقارنتهما باستخدام الوظيفة**Date Diff()** وتحديد التاريخ الأحدث (الأقرب) من خلال ما ترجعه الوظيفة**Date Diff()** فإذا كان سالبا فان هذا يعني أن التاريخ الأول أقدم، وإذا كان موجبا فان هذا يعني أن التاريخ الأول أحدث.