

## الفصل العاشر عشر استخدام عبارات التحكم

سنشرح في هذا الفصل استخدام عبارات التحكم في الإجراءات ونعنى بها تلك العبارات التي تساعد في تحديد الشروط واتخاذ القرارات في حالة وجود أكثر من بديل، وتلك التي تستخدم في التكرارات والدورات بقصد تقليل تعليمات الإجراء. بانتهاء هذا الفصل ستتعرف على:

- ◆ عبارة الشرط **IF**
- ◆ التركيب **If...Then**
- ◆ التركيب **If..Then...Else**
- ◆ عبارة المتداخلة **If**
- ◆ عبارة المقارنة **Select Case**
- ◆ الدوارة **For...Next**
- ◆ الدوارة **Do...Loop**
- ◆ الدوارة **For...Each**
- ◆ مثال تطبيقي على العبارات الشرطية والدورات

يستخدم Visual Basic نوعان من عبارات الشرط هما:

- عبارة الشرط IF.
- عبارة المقارنة Select Case.

## معبارة الشرط IF

دائماً ترتبط بتحقيق شرط معين، فإذا وقع الشرط صحيحاً يتم تنفيذ سطر أو مجموعة من السطور، وتستخدم بتراكيب عديدة نوضحها فيما يلي:

### التركيب IF...Then

يستخدم لتنفيذ أمر واحد أو مجموعة أوامر في حالة تحقق شرط معين. إذا كان المطلوب تنفيذ أمر واحد في حالة وقوع الشرط صحيحاً فإن التركيب يأخذ الصورة العامة التالية:

**If <condition> Then <command>**

وفي هذا التركيب يتم تقييم الشرط (Condition) الوارد بالتعليمة، فإذا كان صحيحاً ينفذ Visual Basic الأمر <Command> الذي يلي كلمة Then، وإلا فإنه يتجاهله، ويصلح هذا التركيب عندما تريد تنفيذ أمر واحد في حالة تحقق الشرط. والمثال التالي يوضح هذه الفكرة:

**If In Age > 80 Then Message Box .Show (" You are too old")**

في هذا المثال الجملة الشرطية عبارة عن عبارة شرط هي IF. الشرط هو أن تكون قيمة Age أكبر من ٨٠، ويرتبط جواب الشرط في البداية بكلمة then ويتم إظهار الرسالة "You are too old" (جواب الشرط) في حالة وقوع الشرط صحيحاً (True).

لاحظ أن العبارة الشرطية تكتب بكاملها على نفس السطر لأن المطلوب تنفيذ أمر واحد في حالة تحقق الشرط. فإذا أردت تنفيذ مجموعة من الأوامر إذا تحقق الشرط أى وقع صحيحاً فإن تركيب IF...Then يأخذ الصورة العامة التالية:

**If <condition> Then**

.....  
**<commands>**

.....  
**End if**

وكما تلاحظ فى هذه الصورة العامة أن الأوامر تكتب فى السطر التالى لكلمة **Then** وأن التركيب ينتهى بعبارة **End if**

مثال

```
IF InAge>80 Then
    Beep()
    MessageBox.Show( "You are too old")
End if
```

التركيب **IF...Then...Else**

ويستخدم لتنفيذ مجموعة أوامر فى حالة تحقق شرط معين. أو مجموعة أخرى فى حالة عدم تحققه ويأخذ هذا التركيب الصورة العامة التالية:

```
If <condition> Then
    <commands>
Else
    <commands>
End if
```

وفى هذا التركيب يتم تقييم الشرط (**Condition**) الوارد به ، فإذا كان صحيحا ينفذ **Visual Basic** الأمر أو الأوامر (**Commands**) التى تلى كلمة **Then** ، والا فإنه ينفذ الأمر أو الأوامر التى تلى كلمة **Else**، ويصلح هذا التركيب عندما تريد تنفيذ مجموعة أوامر إذا وقع الشرط صحيحا ومجموعة أخرى إذا وقع الشرط خطأ.. والمثال التالى يوضح هذه الفكرة

```
IF InAge>= 80 Then
    Beep()
    Message Box .Show ("إنك لكبير السن")
Else
    Message Box .Show ("إن عمرك مناسب")
End If
```

فى المثال السابق تتم مقارنة محتويات المتغير **InAge** بالقيمة ٨٠ فإذا كانت **InAge** تساوى أو أكبر من ٨٠، تنفذ الأوامر التى تلى كلمة **Then**، أما إذا كان أقل من ٨٠، تنفذ الأوامر التى تلى كلمة **Else**.

### التعامل مع الشروط غير المتحققة

تعاملنا فيما سبق مع الشروط التي تتحقق بتنفيذ أوامر معينة، ولكن ماذا عند الرغبة في تنفيذ أوامر عند عدم تحقق شروط معينة؟ . يتيح العامل المنطقي **Not** ذلك بأن نكتبه قبل الشرط المراد فإذا لم يتحقق تم تنفيذ الأوامر، مثلاً:

#### If Not (st Passed =txt Passwd) Then End

المثال السابق يقارن محتويات مربع نص بكلمة سر مخزنة فإن لم يتطابقا تم إنهاء البرنامج، كما ترى العامل **Not** هو الذي قام بعكس المعنى للتعامل مع عدم التحقق. بصورة أعم يمكن التعامل مع كلا الحالتين أي التحقق وعدمه من خلال الصيغة الكاملة للعبارة الشرطية **If** و هي كالتالي:

```
If <condition> Then  
Statement1
```

```
Statement2 <condition=True
```

```
.....
```

```
Else
```

```
Statement1
```

```
Statement2 <condition=False
```

```
.....
```

```
End If
```

هذه الصيغة يزيد عليها فقرة كاملة هي **Else** إلى **End If** وهي تنفذ في حالة عدم تحقق الشرط (**condition=False**) ، أي أن جزءاً واحداً من الجزئين ينفذ، الذي بعد **Then** أو الذي بعد **Else** بناءً على قيمة **condition** هذا يوضح معنى التحكم الذي تقوم به العبارة.

من الصيغة السابقة ندرك قيمة هذه العبارة. لنفترض أننا لا نملك وسيلة لتغيير خط سير البرنامج، على ذلك كان علينا إنشاء عدة برامج لعدة حالات، وكان على المستخدم تحديد البرنامج المناسب الأمر الذي يصبح عسيراً إن لم تكن الحالات واضحة للمستخدم أو كانت كثيرة. و لكن باستخدام عبارة التفرع المشروط أصبح بالإمكان أن نغير من خط سير

البرنامج لكل حالة من المدخلات دون أن يضطر المستخدم لتشغيل عدة برامج.

### محاكاة المتداخلة NestedIf

العبارة الشرطية البسيطة سواء ذات السطر الواحد أو متعددة السطور تقوم باختبار شرط واحد ، ولكن إذا أردنا القيام بالعديد من الاختبارات كأن نقوم بمعرفة هل السن المدخل من المستخدم أكبر من ٢٥ مثلا ، ثم نختبر بعد ذلك هل هو أكبر من ٤٠ أو لا إن كان أكبر من ٢٥ وهكذا. بمعنى آخر هب أن لدينا برنامج يطلب من المستخدم إدخال عمره ثم يقوم بمعاملة المستخدم تبعا لفتته السنية أي يختبر كون العمر يقع في أي فئة ثم ينفذ الكود المناسب. أحد الصور لحل هذه المشكلة استخدام عبارات If المتداخلة كالتالي

```
If InAge>=25 Then
  If InAge>35 Then
    If InAge>65 Then
      MessageBox.Show("إنك كبير السن")
    Else
      MessageBox.Show("إن عمرك مناسب")
    End if
  Else
    MessageBox.Show("إنك لا تزال شاباً")
  End if
Else
  MessageBox.Show("لا زلت صغيرا جدا")
End if
```

المثال السابق يتعامل مع فئات سنية (أقل من ٢٥ ، من ٢٥ حتى ٣٥ ، من ٣٥ حتى ٦٥ ، أكبر من ٦٥) ثم يقوم بإظهار رسالة عن عمر المستخدم. ولكن في المثال السابق كثير من العبارات الزائدة عن الحد المرهقة في كتابتها. لذا يتيح Visual Basic صورة أخرى لكتابة العبارات المتداخلة باستخدام Elseif كالتالي:

```
If InAge<25 Then
  MessageBox.Show("لا زلت صغيرا جدا")
Elseif InAge<35 Then
```

```

MessageBox.Show("إنك لا تزال شاباً")
ElseifinAge<65 Then
    MessageBox.Show("إن عمرك مناسب")
Else
    MessageBox.Show("إنك كبير السن")
End If
    
```

- كما ترى سهلت العبارة السابقة وحسنت من شكل الكتابة ووضوح وظيفة الكود. عند تكوين العبارات المنطقية المركبة يجب أن نعرف أولوية تنفيذ العوامل المختلفة حتى نضمن أن يتم تنفيذ التعبير كما ينبغي. أولوية تنفيذ العوامل تتم بالترتيب التالي:
١. العوامل الحسابية بترتيبها المعروف.
  ٢. عامل الوصل (للمتغيرات الحرفية) **concatenation** وهو **&**.
  ٣. العوامل العلائقية (**Relational Operators**) مثل **<** و **>**.
  ٤. العوامل المنطقية (**Boolean operators**) مثل **And** و **Not**.

### استخدام عبارة **Select Case**

تصلح عبارة الشرط **IF** إذا كان جواب الشرط عبارة عن احتمالين أو ثلاثة. أما إذا كنت تتوقع عند تقييمك لشرط معين احتمالات كثيرة، فمن الأفضل أن تستخدم عبارة **Select Case** ، وتكون صيغتها العامة كما يلي:

```

Select Case <condVar><الشرط>
    Case <value1><الاحتمال الأول>
        Statement group 1
    Case <value2><الاحتمال الثاني>
        Statement group 2
    .....
    .....
    Case Else
        Statement group n
End Select
    
```

حيث تبدأ العبارة بـ **Select Case** يليها اسم المتغير أو التعبير (**expression**) الذي سيتم اختياره و من الممكن أن يكون هذا المتغير أو التعبير من أي نمط عددي حتى الأعداد الحقيقية أي التي بها كسور أو متغير حرفي **String**. يوضح جدول ١١-١ التالي أشهر أنماط المتغيرات المستخدمة مع عبارة **Select Case**.

جدول ١١-١ أنماط المتغيرات المستخدمة مع عبارة **Select Case**.

نمط المتغير	مثال
علائقى	Case IS <= 25
تساوى	Case IS = 15.5
تساوى	Case 15.5
مدى	Case -5 To 5
متعدد	Case IS < 100 , IS >0
حرفى	Case "كمبيوتر"

بعد ذلك تأتي الاحتمالات (عبارات **Case**) بعد كل منها إحدى قيم المتغير الذي سيتم مقارنته ثم يعقبها التعليمات التي ستنفذ إذا كان الشرط صحيحا أو كان المتغير بهذه القيمة. وأخيرا يأتي **Case Else** ومعناها إذا كان المتغير لا يساوي أي من القيم السابقة أو إذا لم يقع الشرط صحيحا، فإن التعليمات التي تلي **Else** هي التي تنفذ. مقطع واحد فقط من التعليمات في العبارة السابقة هو الذي سينفذ وهو الذي ينطبق على أول شرط صحيحا (أول **Case** تتحقق)، بعد ذلك ينتقل التنفيذ إلى آخر عبارة **End Select**، حتى ولو كان بعد هذه الحالة حالات أخرى متحققة فسوف يتجاهلها المترجم، علما بأنه ينبغي عليك التأكد من عدم وجود ذلك لأن هذا يعتبر أحد عيوب البرنامج الصورة السابقة للعبارة **Select Case** استخدمت قيمة واحدة بعد كل **Case**، و لكن يمكن أيضا التعامل بصورة أكفأ باستخدام عدة قيم أو مدى من القيم أو استخدام المقارنات و التي تعطي عددا لا نهائيا من القيم. المثال التالي يستخدم ذلك في تحديد نسبة الخصم على

المبيعات تبعا لعددتها:

### Select Case inQuantity

Case Is < 0 استخدام المقارنة ،

MessageBox.Show("الكمية يجب ألا تكون سالبة")

Exit Sub

Case 1 , 2 , 3 استخدام قائمة ،

SgDiscount=0

Case 4 To 9 استخدام مدى من القيم

SgDiscount=0.1 '10%

Case 10 To 49

SgDiscount=0.2 '20%

Case Is >15

SgDiscount=0.3 '30%

### End Select

لاحظ أننا استخدمنا في العبارة السابقة معامل علائقي في أول وآخر حالة (Case)، والقوائم في الحالة الثانية، ومدى أو نطاق من القيم في الحالة الثالثة والرابعة مما بسط من كتابة العبارة كما هو واضح. القوائم تتكون من قيم مفصولة بفاصلة (،) و المدى يعرف بقيمتين بينهما (To) .

يستخدم المعامل IS غالبا مع المعاملات العلائقية مثل (<> = > =) . بالرجوع إلى المثال السابق تجد أننا استخدمنا المعامل IS في أول حالة لتحديد هل المتغير أقل من القيمة صفر، وهل هو أكبر من القيمة 15 في آخر حالة. في حالة استخدام Case مع القوائم ومدى من القيم لم نستخدم العامل IS. علاوة على ما سبق، فإن القوائم لا يشترط أن تكون قيم منفصلة بل يمكن أن تكون مدى من القيم أو مقارنة مثل:

Case 1 To 4, 7 To 9, 11, 13, Is >23

## الدواريات Loops

لعل قوة الكمبيوتر تكمن أكثر ما تكمن في قدرته على تكرار المهام دون كلل أو خطأ،

والتكرار هو أساس معظم العمليات الهندسية و العلمية و كذلك المعلوماتية و الحاسوبية كأن تطبق وظيفة ما على كل سجلات قاعدة بيانات ، مثل طباعة مرتبات العاملين أو حساب الخصومات على كافة الموظفين و هكذا. و تعرف الكلمات الخاصة التي توضع في بداية و نهاية العبارات المطلوب تكرارها بالدورات.

يستخدم **Visual Basic** نوعين من الدورات، الأولى بعداد و تسمى **Counter loop** وهي التي تنفذ عدد محدد من المرات، الثانية لا يحدد فيها العدد و إنما تعتمد على أحد الشروط و تسمى الدورات المشروطة **Conditional loops** . فيما يلي شرح لهذين النوعين بالتفصيل.

#### الدورة For...Next

الدورات ذات العداد، هي الدورات المسماة **For** أو **For Next** و ذلك لأن الجزء المراد تكرار تنفيذه يكتب بين كلمتي **For** و **Next** . الصورة العامة للعبارة **For** كالتالي:

```
For counter = startval To endval [ step stepval ]
```

```
Statement1
```

```
Statement2
```

```
.....
```

```
Next [counter]
```

كما ترى يتم تحديد متغير يسمى العداد ثم نحدد القيمة المبدئية للعداد و القيمة النهائية ، و يمكننا أن نحدد الخطوة **step** التي يتم زيادة العداد بها في كل دورة ، إن لم نحدد هذه الخطوة فالقيمة الافتراضية أن العداد يزيد بمقدار ١ في كل مرة ، فإذا تعدى العداد القيمة **endvalue** فإن تنفيذ الدورة يتوقف و ينتقل التنفيذ إلى أول جملة بعد **Next** . إذا كانت القيمة النهائية أصغر من القيمة المبدئية فإن الدورة لن تنفذ على الإطلاق ، إلا إذا تم تحديد قيمة سالبة للخطوة ، مثلا **step -1** .

عند استخدام العداد يجب أن تتنبه إلى حدود قيم المتغير، فمثلا المتغير من النمط **Byte** أقصى قيمة يسعها هي ٢٥٥ ، و عليه لا يمكنك استخدامه كعداد يعد حتى ٤٠٠ مثلا ، و عليك حينئذ استخدام عداد من نمط آخر.

لا تقم بتغيير قيمة العداد داخل الدوارة أبداً، قد يؤدي ذلك إلى دوارة لا نهائية **infinite loop** وذلك بأن يظل البرنامج يكرر هذه الدوارة بلا نهاية مسبباً عطل للنظام.



أحيانا نحتاج إلى الخروج من الدوارة لسبب معين كان تنتهي وظيفة التكرار. يمكننا ذلك باستخدام **Exit For** ، التي تنقل التنفيذ إلى التعليمية التي تلي **Next** مباشرة.

مثال ١ :

المثال الآتي يستخدم الدوارة **For...Next** في أبسط صورها لطباعة الأرقام من ١ إلى ٥

```
Dim i As Integer
For i = 1 To 5
Message Box .Show (i)
Next i
```

وعن هذا المثال نوضح مايلي:

- أعلننا عن المتغير (i) للحصول على أرقام متغيرة من 1 إلى 5.
- تعمل الدوارة **Next .. For** على زيادة قيمة المتغير بمقدار (١) في كل مرة دون الحاجة إلى كتابة الأمر التالي في كل دورة.

**i = i + 1**

وذلك لأن الشرط المحدد في أمر **For** وهو **i=1 To 5** يشتمل على بداية عداد الدوارة ونهايته، حيث أن **i=1** معناها بداية العداد المستخدم داخل الدوارة ، **To 5** نهاية عداد الدوارة. ويجب الالتزام بهذه الصيغة دائماً مع أمر **For...Next** أي كتابة بداية العداد بعد علامة = ونهايته بعد كلمة **To**.

- يتم الخروج تلقائياً من الأمر **Next .. For** بمجرد الوصول إلى العدد (5) في المتغير (i) وهذا يعني أن عدد مرات التكرار التي تمت لمجموعة الأوامر المطلوب تكرارها هو (5).

### استخدام الوظيفة Step()

يتم استخدام الوظيفة Step() مع الأمر For .. Next لمعرفة مقدار الزيادة التى تتم على المتغير فى كل دورة. ففى الحالة السابقة لم نذكر الوظيفة Step() فى الأمر:

```
For i = 1 To 5
```

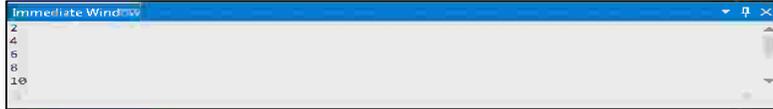
لأن الزيادة التلقائية فى كل دورة مقدارها ١ مالم تذكر خلاف ذلك بالوظيفة Step() ، أى أن الأمر السابق يساوى تماما هذا الأمر:

```
For i = 1 To 5 Step 1
```

فمثلا لطباعة الأرقام الزوجية فقط فى الأرقام ١ الى ١٠ ، استخدم الكود التالى:

```
For i = 2 To 10 Step 2  
Debug.WriteLine(i)  
next i
```

إذا قمت بتنفيذ هذا الكود، تحصل على النتيجة الموضحة بشكل ١-١١ التالى.



شكل ١-١١ طباعة الأرقام الزوجية فقط باستخدام الوظيفة Step().

### مثال ٢ :

المثال الآتى مثال هام، حيث يبين كيفية استخدام الدورات ، وكيفية الخروج الطارئ منها، هذا خلافا لكونه ذو فائدة عملية. المثال يبحث فى مصفوفة حرفية باسم Name Array عن اسم معين هو st search ، وبمجرد العثور على هذا الاسم، يتم الخروج من الدوارة وينتقل التنفيذ للسطر التالى لها. وبالطبع معظم عمليات البحث تتم هكذا، حيث لا حاجة أن نكمل البحث بعد العثور على ما نريده.

```
For cntr = 1 To 30  
    found= Instr (1, Name Array(Index), stsearch,1)  
    If found>0 then  
        txtResult.Text=NameArray(cntr)  
        Exit For  
    End If  
Next cntr
```

لاحظ في المثال السابق أن الدالة Instr تستخدم لاختبار وجود stsearch في أي عنصر من عناصر المصفوفة، وإن وجد فإن الوظيفة ترجع الحرف الذي يبدأ عنده النص المطلوب ( ١ أو أكثر ) وترجع الوظيفة صفراً إن لم تجد هذا النص. بعد ذلك يتم اختبار الناتج، فإن كان أكبر من صفر فإن العنصر الذي تم العثور عليه يتم عرضه في مربع نص يسمى txtResult، ثم يتم الخروج من الدوارة من خلال العبارة Exit For، ذلك لأننا لا نحتاج لتكملة البحث بعد أن وجدنا ما نريد. المثال السابق أيضاً يوضح العلاقة الوثيقة بين المصفوفات والدورات، حيث يمكن القيام بمهام عالية الكفاءة بالدمج بينهما.

نذكر أن Visual Basic وخلاف العديد من اللغات الأخرى تتيح استخدام متغير كسري Single أو Double كمتغير للدوارة كما تتيح استخدام خطوة Step كسرية ( أقل من ١ ) ولكن يجب ألا نلجأ إلى هذه الطريقة إلا عند الحاجة الشديدة لأن الأعداد الكسرية أصعب في التعامل واكتشاف الأخطاء من الأعداد الصحيحة.



### الدوارة Do... Loop

الفرق الرئيسي بين الدورات التكرارية والدورات المشروطة أن الأخيرة تعتمد في تنفيذها على شرط (Condition)، بينما الأولى تنفذ عدد معين من المرات. الشرط الذي يعتمد عليه التنفيذ هو أي تعبير يمكن أن ينتج القيمة True أو False. هذا التعبير من الممكن أن يكون وظيفة مثل Eof() (والتي تحدد هل وصلت إلى نهاية ملف أم لا) أو قيمة أحد خواص كائن مثل الخاصية Value الخاصة بزر الخيارات Option Button، أو تعبير به مقارنة مثل inAge > 25، أو متغير من النوع Boolean. هناك نوعين من الدورات المشروطة هما Do While و Do Until وسنشرحهما بالتفصيل في الفقرة التالية.

### دوارة Do... While

يتم تنفيذ الدوارة Do While طالما ظل الشرط متحققاً، وعندما لا يتحقق، يتم نقل التنفيذ إلى العبارة التالية للدوارة.

هناك طريقتين أو شكلين لكتابة هذه الدوارة، كالتالى:

الشكل الأول

**Do While condition  
statement**

.....

**Loop**

الشكل الثانى

**Do  
statement**

.....

**Loop While condition**

الفرق بين الشكلين هو "متى يتم اختبار تحقق الشرط"، في الشكل الأول الاختبار يتم قبل إجراء التعليمات في الدوارة، وعليه فلو كان الشرط غير متحقق لن يتم تنفيذ العبارات في الدوارة ولا مرة. بينما في الشكل الثانى يتم الاختبار بعد إجراء التعليمات في الدوارة وعليه فإن هذه التعليمات تنفذ مرة واحدة على الأقل حتى ولو كان الشرط غير متحقق. واحد فقط من الشكلين يمكن استخدامه، أي لا يمكن وضع **While** في أول وآخر الحلقة. واستخدام أي منهما يعتمد على التطبيق والوظيفة التي تريدها، وكلا النوعين قد يكون مفيد لك.

في الإصدارات القديمة من **Visual Basic** كانت الكلمة **Wend** تستخدم بدلا من **Loop** ولا يزال **Visual Basic** يدعمها حتى الآن إلا أنه من المستحسن استخدام الطريقة الجديدة في كتابة الدوارة.

مثال ٣:

التعليمات التالية تعطى نفس نتيجة المثال الذي شرحناه لطباعة الأرقام من ١ الى ٥ :

```
Dim i As integer
i = 1
Do While i <= 5
    Debug.WriteLine(i)
    i = i + 1
Loop
```

### الدوارة Do...Until

تعد الدوارة **Do Until** عكس الدوارة السابقة **Do While** ، حيث أن تنفيذ الدوارة هنا يستمر ما لم يتحقق الشرط فإن تحقق يتم الخروج من الدوارة إلى التعليمة التالية. واستخدامها مماثل تماما للدوارة السابقة **Do While** ، و هناك أيضا شكلين لكتابتها كالتالي:  
الشكل الأول:

```
Do Until condition  
statement
```

.....

```
Loop
```

الشكل الثاني:

```
Do  
statement
```

.....

```
Loop Until condition
```

ولهما نفس المعنى السابق، حيث أن الشكل الثاني ينفذ مرة على الأقل قبل الخروج من الحلقة. أكثر استخدام للدوارة **Do Until** هو معالجة الملفات ذات السجلات المتعاقبة ، أو قواعد البيانات.

مثال ٤ :

التعليقات التالية تعطي نفس نتيجة المثال السابق:

```
Dim i As integer
```

```
i = 1
```

```
Do Until i > 5
```

```
Debug.WriteLine(i)
```

```
i = i + 1
```

```
Loop
```

لاحظ الفرق بين المثالين السابقين تجده يتركز في الصيغة:

```
Do While i <= 5
```

ومعناه أنه سيتم التكرار طالما أن المتغير (i) أصغر من أو يساوي القيمة 5 بينما الصيغة التالية:

```
Do UNTIL i > 5
```

تعني أنه سيتم التكرار حتى تصبح قيمة المتغير (i) أكبر من خمسة.

لاحظ أن العلاقة (<=) هي عكس العلاقة (>).

على أية حال الدورات المشروطة أكثر مرونة من الدورات ذات العداد. ويمكن تحويل ذات العداد إلى دوائر مشروطة بأن نستخدم متغير ولكن في هذه الحالة يجب أن نقوم بزيادة قيمته بأنفسنا، الأمر الذي يتم تلقائياً في حالة الدورات ذات العداد.

من المشاكل المصاحبة للدورات مشكلة الدورات اللانهائية و التي تنتج إذا أخطأت في تحديد الشروط، فتظل الدائرة تعمل إلى لا نهاية مسببة عطل تام للنظام. لكي تنجو من ذلك الفخ إن كنت تعمل في بيئة Visual Basic اضغط **Ctrl +Break** لإيقاف عمل البرنامج. أما إن حدث ذلك في برنامج تم ترجمته إلى EXE وتقوم بتشغيله مستقلاً، فإن الحل للخروج من هذه الحالة ( دون الاضطرار لإعادة تشغيل الجهاز) أن تقوم بضغط **Ctrl +Alt +Delete** ليظهر لك مربع مدير المهام Task Manager الذى يمكنك من خلاله إغلاق البرنامج المتعطل.

### الدائرة For...Each

يمكنك استخدام الدائرة For...Each لتنفيذ عدد من العبارات على مجموعة من الكائنات المرتبطة، لذا فهي غالباً ما تستخدم مع المصفوفات Arrays ومجموعات البيانات Data Set وأدوات التحكم. للتعرف على طريقة عمل هذه الدائرة، دعنا نرى الكود التالي:

```
Dim MyNumber As Integer
Dim MyArray() As Integer = {5,7,9,3,1,6,8}
For Each MyNumber In MyArray
    If MyNumber > 5 Then Debug.WriteLine(MyNumber)
Next My Number
```

حيث يتم اختبار جميع عناصر المصفوفة ومن ثم طباعة الأرقام الأكبر من 5 (انظر شكل

١١-٢).



شكل ١١-٢ استخدام الدوارة For Each.

ومن أهم استخدامات الدوارة For Each، تنفيذ مجموعة من العمليات على عدد من أدوات التحكم الموجودة بالنموذج. فعلى سبيل المثال، لإخفاء جميع أدوات التحكم الموجودة بالنموذج الحالي، يمكنك استخدام الكود التالي:

```
Dim Ctrl ValAs Control
```

```
For Each Ctrl ValIn Controls
```

```
Ctrl Val. Visible = False
```

```
Next Ctrl Val
```

وقد قمنا في هذا الكود بدايةً بتعريف المتغير Ctrl Val الذى سيحتوى على أدوات التحكم ثم استخدام هذا المتغير داخل الدوارة، التى يتم فيها تخصيص القيمة False للخاصية Visible المصاحبة لكل أداة من أدوات التحكم الموجودة بالنموذج.

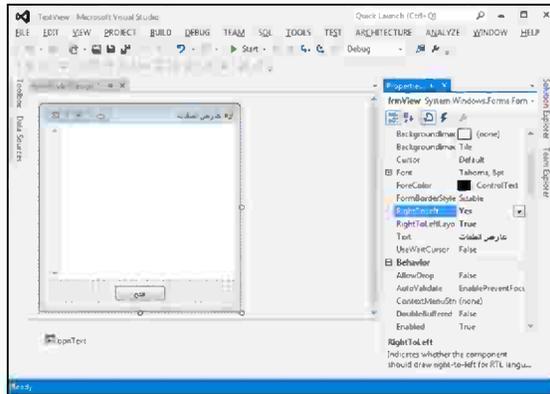
### مثال تطبيقي

بعد الأمثلة البسيطة التى قدمناها لكل نوع من العبارات السابقة، نوضح الآن مثالا أكبر وأكثر فائدة يحتوي على العديد من العبارات التى ذكرناها. المثال الذى نقدمه هو إنشاء عارض ملفات نصوص Text والتي لها الاختصار txt والتي ينشئها محرر النصوص في Windows وتجده على القرص المدمج المرفق بالكتاب باسم Text View.

إنشاء واجهة عارض الملفات

1. قم بإنشاء مشروع نوافذى جديد ثم اجعل اسمه Text View.
2. غير اسم النموذج الرئيسي إلى frm View من خلال الخاصية Name وغير عنوانه إلى "عارض الملفات" من خلال الخاصية Text وقم أيضاً بتغيير قيمة الخاصية Right to Left إلى Yes.

٣. أضف مربع نص **Text Box** إلى النموذج وحدد خصائصه كالتالى:
- Name=txtView**  
**MultiLine=True**  
**ScrollBars=Both**
٤. أضف أداة مربع فتح الملفات **Open File Dialog** من مربع الأدوات إلى النموذج، تظهر الأداة أسفل النموذج كالمعتاد.
٥. قم بإعادة تسمية الأداة من خلال الخاصية **Name** إلى **txtOpen**. قم أيضاً بحذف القيمة الموجودة بالخاصية **File Name**.
٦. أضف زر أمر **btnOpen** و قم بتغيير عنوانه إلى "فتح" و اسمه إلى **btnOpen**. يجب أن يظهر النموذج الآن كما فى شكل ٣-١١ التالى.



شكل ٣-١١ مظهر النموذج عند إضافة عناصر الواجهة الأساسية إليه.

### إضافة الكود للبرنامج

كود البرنامج بسيط للغاية، انقر الزر "فتح" نقرأ مزدوجاً ثم قم بإدخال الكود التالى داخل إجراء النقر الخاص بالزر:

```
Dim F As String ' اسم الملف
Dim N As Integer ' رقم الملف
Dim S As String ' مخزن لسطر
Dim Buffer As String ' مخزن للملف بالكامل
```

```
OpnText.Filter = "Text Files|.txt"
```

```
OpnText.ShowDialog()
```

```
F = OpnText.FileName
```

```
Me.Text = F ' تغيير عنوان النافذة
```

```
N = FreeFile()
```

```
FileOpen(N, F, OpenMode.Input)
```

```
Do Until EOF(N)
```

```
S = LineInput(N)
```

```
Buffer = Buffer &vbCrLf& S
```

```
Loop
```

```
txtView.Text = Buffer
```

```
FileClose(N)
```

يستخدم البرنامج أربعة متغيرات كما ترى تعليقا عليها بجانبها. وكافة المهام يتم تنفيذها عند ضغط زر `btnOpen` ولذا تجدها في الإجراء `Click` لهذا الزر. يبدأ الإجراء بإعداد الخاصية `Filter` لمربع الحوار `Open` بحيث تظهر الملفات النصية `txt` فقط. بعد ذلك يتم فتح المربع و اختيار اسم الملف ووضع اسم المتغير `F` ومن ثم وضع اسم الملف على شريط عنوان النموذج. بعد ذلك يتم حجز رقم متاح باستخدام الوظيفة `(Free File)`، يلي ذلك فتح الملف من خلال الوظيفة `(File Open)` ثم ملء المتغير `Buffer` بسطور الملف النصي وذلك باستخدام الدوارة:

```
Do Until EOF(N)
```

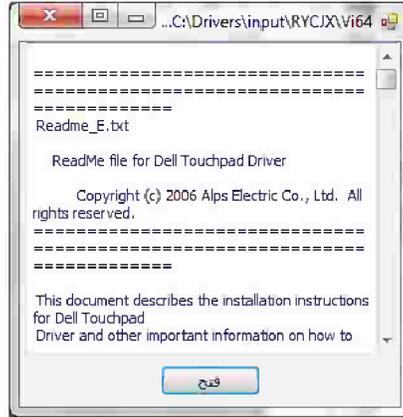
```
S = Line Input(N)
```

```
Buffer = Buffer &vbCrLf& S
```

```
Loop
```

كما ترى استخدمنا دوارة مشروطة `Do Until` ووضعنا الشرط في البداية لأن الملف قد لا يحتوي على أية سطور، الشرط هنا هو `EOF(N)` أو العثور على علامة انتهاء الملف. بداخل الدوارة سطرين الأول يقوم بقراءة سطر من الملف في المتغير `S` باستخدام الوظيفة `(Line Input)` والثاني يقوم بإضافة هذا السطر (مع علامة بداية السطر `CR` و سطر جديد `LF`) ونستخدم لهما الثابت `VbCrLf` إلى المتغير `Buffer`. هنا قد يتبادر سؤال للذهن: لماذا لا نكتب مباشرة في مربع النصوص بدلا من استخدام `Buffer` ؟ ، للإجابة على هذا السؤال قم بإعادة تنفيذ البرنامج بدون `Buffer` وستجد فرقا هائلا في سرعة تنفيذ البرنامج،

يرجع ذلك إلى أن الوصول إلى المتغير البسيط أسرع بمراحل من الوصول إلى خاصية في أداة تحكم. أخيراً يتم كتابة المتغير **Buffer** في مربع النصوص وإغلاق الملف من خلال الوظيفة (**File Close**) (انظر شكل ١١-٤).



شكل ١١-٤ نافذة برنامج عارض الملفات أثناء عمله وبه أحد الملفات النصية.

هناك قصور في هذا البرنامج و هو أنك إذا لم تختار ملف من مربع الحوار فسيحدث خطأ. للتغلب على هذه المشكلة نستخدم العبارة الشرطية **If** كالتالى:

```
If F = "" Then
  MessageBox.Show("You must select a file")
Else
  Me.Text = F      تغيير عنوان النافذة
  N = FreeFile()
  FileOpen(N, F, OpenMode.Input)
  Do Until EOF(N)
    S = LineInput(N)
    Buffer = Buffer &vbCrLf & S
  Loop
  txtView.Text = Buffer
  FileClose(N)
End If
```

بعد ذلك لن تحدث مشكلة إذا لم تقم باختيار ملف حيث يتم إظهار مربع رسالة لخطك على اختيار أحد الملفات.

وعلى الرغم من بساطة المثال السابق عرضه إلا أنه يوضح طريقة فعلية لاستخدام الدورات والعبارات الشرطية و لا أظن أن الكثير من الأمثلة ستفيد هنا بل الأفضل أن تقوم بنفسك باستخدام الدورات و العبارات الشرطية، و التعلم من أخطائك مع الرجوع لملف المساعدة كلما أمكن.

