الفصل الخامس عشر
استخدام عبارات التحكم

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

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

يستخدم Visual Basic نوعان من عبارات الشرط هما:

- عبارة الشرط IF .
- عبارة المقارنة Select Case.

عبارة الشرط IF

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

التركيب IF...Then

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

```
If <condition> Then <command>
```

وفى هذا التركيب يتم تقييم الشرط (Condition) الوارد بالتعليمة، فإذا كان صحيحا ينفذ Visual Basic الأمر< Command> الذي يلى كلمة Then، وإلا فانه يتجاهله، ويصلح هذا التركيب عندما تريد تنفيذ أمر واحد فى حالة تحقق الشرط. والمثال التالى يوضح هذه الفكرة:

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

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

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

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

وكما تلاحظ فى هذه الصورة العامة أن الأوامر تكتب في السطر التالي 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 فانه 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 خطئا.. والمثال التالى يوضح هذه الفكرة

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

فى المثال السابق تتم مقارنة محتويات المتغير InAge بالقيمة ٨٠ فإذا كانت InAge تساوي 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، مثلا:

```
If Not (stPasswd=txtPasswd) 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 هذا الجزء ينفذ في حالة التحقق
        ……
Else
        Statement1
        Statement2   ≺condition=False ينفذ في حالة عدم تحقق الشرط
        ……
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** بناءاً 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 عسيرا 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.

عبارة If المتداخلة Nested If

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 لفئته 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 شابا ")
ElseIf  inAge<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 Not وAnd .

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 اختباره 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 ١٥-١ التالي أشهر أنماط المتغيرات المستخدمة مع عبارة Select Case.

جدول ١٥-١ 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 ستتم 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 أيا من القيم السابقة أو إذا لم يقع الشرط صحيحا، فإن التعليمات التى تلي Else هي التى 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 لعددها:

```
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 ،ومدىً 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 ١٥ في آخر حالة. في حالة استخدام Case مع القوائم ومدىً من القيم لم نستخدم العامل IS .

علاوةً 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
```

الدوارات 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 والمحاسبية كأن تطبق وظيفة ما على كل سجلات قاعدة بيانات ، مثل طباعة مرتبات العاملين أو حساب الخصومات على كافة الموظفين و هكذا. وتعرف الكلمات الخاصة التي توضع في بداية و نهاية العبارات المطلوب تكرارها بالدوارات.

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

الدوارة For...Next

الدوارات ذات العداد، هي الدوارات 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 ، ويمكننا 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 الدوارة يتوقف وينتقل التنفيذ إلى أول جملة بعد Next . إذا كانت القيمة النهائية أصغر من القيمة المبدئية فإن الدوارة 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، وعليه 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 مسببا 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
    MessageBox.Show(i)
Next i
```

وعن هذا المثال نوضح مايلى:

- أعلنا عن المتغير (I) للحصول على أرقام متغيرة من 1 إلى 5.
- تعمل الدوارة For .. Next على زيادة قيمة المتغير بمقدار (١) فى كل مرة دون الحاجة إلى كتابة الأمر التالى فى كل دورة.

```
i = i + 1
```

وذلك لأن الشرط المحدد فى 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 دائما مع أمر For...Next أى كتابة بداية العداد بعد علامة = ونهايته بعد كلمة To.

- يتم الخروج تلقائياً من الأمر For .. Next بمجرد الوصول إلى العدد (5) فى المتغير (I) وهذا يعنى أن عدد مرات التكرار التى 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 ١مالم 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 ١٥-١ التالى.

شكل ١٥-١ طباعة الأرقام الزوجية فقط باستخدام الوظيفة Step.

مثال ٢ :

المثال الآتي مثال هام، حيث يبين كيفية استخدام الدوارات ، وكيفية الخروج الطارئ منها، هذا خلافا لكونه ذو فائدة عملية. المثال يبحث في مصفوفة حرفية باسم NameArray عن اسم معين هو stsearch ، وبمجرد العثور على هذا الاسم، يتم الخروج من الدوارة وينتقل التنفيذ للسطر التالي 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, NameArray(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 الدالة ترجع الحرف الذي يبدأ عنده النص المطلوب (١ أو أكثر) و ترجع الدالة صفرا 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 أيضا يوضح العلاقة الوثيقة بين المصفوفات والدوارات، 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 الدوارات التكرارية والدوارات 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 دالة 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 الدوارات المشروطة هما **Do While** و **Do Until** وسنشرحهما بالتفصيل في الفقرة التالية.

دوارة Do... While

يتم تنفيذ الدوارة **Do While** طالما ظل الشرط متحققا، وعندما لا يتحقق، يتم نقل

التنفيذ إلى العبارة التالية للدوارة.

هناك طريقتين أو شكلين لكتابة هذه الدوارة، كالتالي:

الشكل الأول

```
Do While condition
        statement
        ......
Loop
```

الشكل الثاني

```
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 الثاني 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** و لا يزال **Visual Basic** يدعمها حتى الآن إلا أنه من المستحسن استخدام الطريقة الجديدة في كتابة الدوارة.

مثال٣:

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
```

و لهما 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:

التعليمات التالية تعطى 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 فى الصيغة:

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

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

```
Do UNTIL i > 5
```

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

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

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

من المشاكل المصاحبة للدوارات مشكلة الدوارات اللانهائية و التي تنتج إذا أخطأت في تحديد الشروط، فتظل الدوارة 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 مستقلا، 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. والذي يمكنك من خلاله إغلاق البرنامج المتعطل.

الدوارة For...Each

يمكنك استخدام الدوارة For ... Each لتنفيذ عدد من العبارات على مجموعة من الكائنات المرتبطة، لذا فهى غالباً ما تستخدم مع المصفوفات Arrays ومجموعات البيانات DataSet وأدوات التحكم. للتعرف على طريقة عمل هذه الدوارة، دعنا نرى الكود التالى:

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

حيث يتم اختبار جميع عناصر المصفوفة ومن ثمَّ طباعة الأرقام الأكبر من ٥ (انظر شكل ١٥-٢).

شكل ١٥-٢ 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 الحالى، يمكنك استخدام الكود التالى:

```
Dim CtrlVal As Control
For Each CtrlVal In Controls
    CtrlVal.Visible = False
Next CtrlVal
```

وقد قمنا فى هذا الكود بدايةً بتعريف المتغير **CtrlVal** 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 التي ذكرناها. المثال الذي نقدمه هو إنشاء عارض ملفات نصوص **Text** والتي لها الاختصار **txt** والتي ينشئها محرر النصوص في **Windows** وتجده على القرص المدمج المرفق بالكتاب باسم **TextView** .

إنشاء واجهة عارض الملفات

١. قم بإنشاء مشروع نوافذى جديد ثم اجعل اسمه **TextView** .

٢. غير اسم النموذج الرئيسي إلى **frmView** 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 كالتالي:

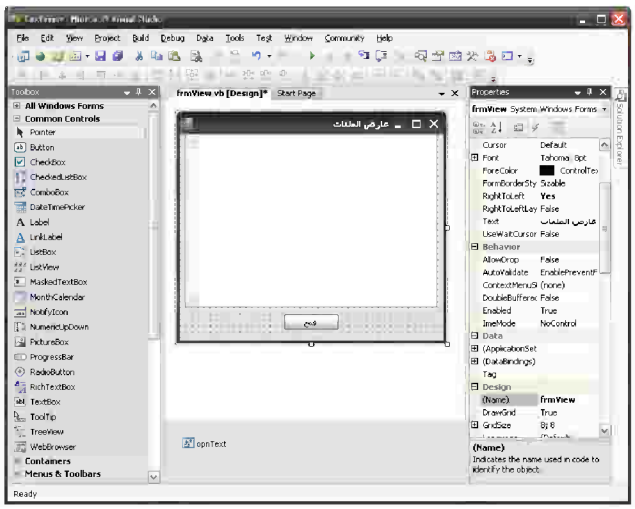
Name = txtView
MultiLine = True
ScrollBars = Both

٤. أضف أداة مربع فتح الملفات **OpenFileDialog** من مربع الأدوات إلى النموذج، تظهر الأداة أسفل النموذج كالمعتاد.

٥. قم بإعادة تسمية الأداة من خلال الخاصية **Name** إلى **opnText** .

٦. أضف زر أمر وقم بتغيير عنوانه إلى "فتح" و اسمه إلى **btnOpen** .

يجب أن يظهر النموذج الآن كما فى شكل ١٥-٣ التالى.



شكل ١٥-٣ مظهر النموذج عند إضافة عناصر الواجهة الأساسية إليه.

إضافة الكود للبرنامج

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

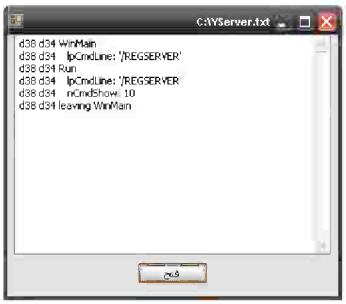
```
Dim F As String 'اسم الملف
Dim N As Integer     ' رقم الملف
Dim S As String      ' مخزن لسطر
Dim Buffer As String ' مخزن للملف بالكامل
OpnText.Filter = "Text Files|*.txt"
OpnText.ShowDialog()
F = OpnText.FileName
Me.Text = F      ' تغيير عنوان النافذة
N = FreeFile()
FileOpen(N, F, OpenMode.Input)
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 ووضعه في المتغير F ومن ثمّ وضع اسم الملف على شريط عنوان النموذج. بعد ذلك يتم حجز رقم متاح باستخدام الدالة FreeFile()، يلي ذلك فتح الملف من خلال الدالة FileOpen() ثم ملء المتغير Buffer بسطور الملف النصي وذلك باستخدام الدوارة:

```
Do Until EOF(N)
        S = LineInput(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 الدالة

LineInput() والثاني يقوم بإضافة هذا السطر (مع علامة بداية السطر CR وسطر 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 الدالة FileClose() (انظر شكل ١٥-٤).



شكل ١٥-٤ نافذة برنامج عارض الملفات أثناء عمله وبه أحد الملفات النصية.

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

```
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 لحثك على اختيار أحد الملفات.

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