الفصـــل الخامس

أساليب البرمجة المتقدمة بلغة Visual Basic

٥-٠ تمهيد

بعد أن تعرفنا على أساسيات البرمجة بلغة Visual Basic فى الفصل السابق، حان الوقت لكى نلقى الضوء على بعض المفاهيم والأساليب المتقدمة فى البرمجة عموما، وفى لغة VB بشكل محدد. وسوف نركز بشكل محدد على أربعة موضوعات رئيسة، تشكل المربع الذهبى عند بناء البرامج المعقدة. ولسهولة هضم هذه المفاهيم المتقدمة، حرصنا ـ عزيزى القارئ ـ على أن نقدم تدريبات عملية، يمكنك تطبيقها مباشرة على الكمبيوتر.

الأهداف التعليمية

بنهاية هذا الفصل ، يجب أن يكون الدارس قادرا على:

- تعريف الدوال والإجراءات الفرعية Functions and Subs.
- تنفيذ جمل التكرار Looping المختلفة.
- تنفيذ جمل اتخاذ القرار Decision Making المختلفة.
- معالجة أخطاء Handling Errors التنفيذ داخل الكود.

١ـ٥ الدوال والإجراءات الفرعية Functions and Subs

سوف نتعرف معا ـ عزيزى القارئ ـ فى هذا الجزء على إنشاء الإجراءات. ويمكن تعريف الإجراء Procedure بأنه كود يمكن تشغيله من داخل كود آخر. كما سوف نتعرف على كيفية إعداد البيانات الضرورية parameters اللازمة للإجراء.

تتعلق وظيفة الإجراء بإعطاء البرنامج تعليمات تنفيذ شيء ما. وربما لم تلحظ ـ عزيزى القارئ ـ أنك بالفعل استخدمت الإجراءات فى الفصل الرابع. على سبيل المثال، تعتبر الدالة MsgBox إجراء يقوم بعرض رسالة على شاشة الكمبيوتر.

تحتوى لغة Visual Basic على العديد من الإجراءات الجاهزة للاستخدام. إلا أنه ـ فى بعض الأحيان ـ تستدعى الضرورة قيام المبرمج ببناء إجراء معين؛ نظرا إلى عدم توفره فى لغة VB. على سبيل المثال، لا تستطيع الدالة MsgBox عرض صندوق حوار يحتوى على صورة.

١ـ١ـ٥ فائدة الإجراءات

تخيل معى ـ عزيزى القارئ ـ هذا المثال: لديك إجراء يطلق عليه PlaySound يحتوى على الكود الضرورى لتشغيل ملف wave (أحد أنواع الملفات الصوتية). على الرغم من أنه يمكنك كتابة كود تشغيل الصوت فى كل مرة تحتاج فيها من البرنامج إحداث صوت، إلا أنه من المنطقى إنشاء إجراء واحد single procedure يمكنك استدعاؤه من أى مكان بالبرنامج.

يتم تشغيل الإجراء عن طريق استدعائه داخل الكود. فمثلا لتشغيل الإجراء

PlaySound يجب إضافة سطر ضمن كود البرنامج يحتوى على اسم الإجراء، على النحو التالى:

```
PlaySound
```

وعندما يصل البرنامج إلى هذا السطر، ينتقل الكود إلى الإجراء PlaySound وينفذ الكود الموجود داخله، ثم يعود الكود ثانية إلى السطر التالى للسطر الذى تم استدعاء الإجراء فيه.

يمكنك استدعاء أى عدد من الإجراءات كما تشاء، وسوف يتم تنفيذها جميعا بترتيب الاستدعاء ذاته. على سبيل المثال، نفترض أنه يوجد أيضا إجراء يطلق عليه DisplayResults؛ لكى تقوم بتنفيذه بعد تنفيذ الإجراء PlaySound يجب أن تقوم باستدعاء الإجراءين بالشكل التالى:

```
PlaySound

DisplayResults
```

٥ـ١ـ٢ أنواع الإجراءات

يوجد نوعان من الإجراءات: الدوال Functions والإجراءات الفرعية Subroutines أو اختصارا Subs. تقوم الدالة باسترجاع قيمة للإجراء الذى قام باستدعائها. فى حين تقوم الدالة فقط بتنفيذ كود معين. يتم استدعاء الإجراءات الفرعية Subs بإضافة سطر من الكود يحتوى على اسم الإجراء Sub كما فى المثال التالى:

```
DisplayResults
```

لكن الدوال مختلفة؛ لأنها لا تقوم فقط بتنفيذ الكود، لكنها تسترجع قيمة معينة. على سبيل المثال تخيل أن هناك دالة يطلق عليها GetDayOfWeek تقوم باسترجاع رقم صحيح Integer يدل على يوم من أيام الأسبوع. يقوم المبرمج باستدعاء هذه

الدالة بالإعلان أولا عن متغير variable لكى تخزن فيه القيمة المسترجعة، ثم تخصيص القيمة المسترجعة لهذا المتغير للاستخدام اللاحق، كما فى المثال التالى:

```
Dim Today As Integer

Today = GetDayOfWeek
```

فى المثال السابق، يتم تخزين القيمة المسترجعة من الدالة فى المتغير Today بهدف استخدامها لاحقا.

٥-١-٣ كتابة الإجراءات Writing Procedures

تتم كتابة الإجراءات عن طريق الإعلان عنها أولا procedure declaration. فى هذا الإعلان، نحدد ما إذا كان هذا الإجراء دالة function أو إجراء فرعيا sub. ثم نقوم بتسمية الإجراء، وبعد ذلك نحدد أية بيانات ضرورية parameters قد يحتاجها الإجراء. انظر المثال التالى للإعلان عن إجراء:

```
Sub MyFirstSub( )
End Sub
```

تدل الكلمة المفتاحية Sub على أن الإجراء عبارة عن Sub وبالتالى لن يسترجع قيمة. يأتى بعد ذلك اسم الإجراء الفرعى MyFirstSub. أما الأقواس الهلالية الفارغة فتشير إلا أنه لا توجد بيانات ضرورية parameters يحتاجها الإجراء. وأخيرا تشير الكلمة المفتاحية End Sub إلى نهاية الإجراء الفرعى. وبين هذين السطرين يتم كتابة الكود المطلوب من الإجراء الفرعى تنفيذه.

يتم وبالطريقة ذاتها، الإعلان عن الدوال functions، لكن مع ضرورة تحديد نوع القيمة المسترجعة (.Integer, String, etc). على سبيل المثال، الدالة التى تقوم باسترجاع رقم صحيح Integer قد تبدو على النحو التالى:

```
Function MyFirstFunction( ) As Integer
End Function
```

وللحصول على القيمة المسترجعة بواسطة الدالة، نستخدم الكلمة المفتاحية Return كما فى المثال التالى:

```
Function GetTheNumberOne( ) As Integer
  Return 1
End Function
```

تدريب عملى

١. من قائمة File اختر New Project

٢. اختر Windows Application ثم اكتب اسم البرنامج MyFirstProcedures.

٣. اضغط مرتين على سطح النموذج لفتح نافذة الكود.

٤. اكتب الكود التالى مباشرة قبل عبارة End Class:

```
Function GetTime( ) As String
  Return CStr(Now)
End Function
```

تستخدم هذه الدالة الإجراء الجاهز فى لغة VB وهو Now للحصول على التوقيت الحالى، وبعد ذلك تستخدم الدالة CStr لتحويل القيمة المسترجعة بواسطة Now إلى نص String. وأخيرا يتم استرجاع هذه القيمة النصية كنتيجة للدالة.

٥. فوق الدالة التى قمت بكتابتها فى الخطوة السابقة، أضف الإجراء الفرعى التالى:

```
Sub DisplayTime( )
  MsgBox(GetTime)
End Sub
```

يقوم هذا الإجراء الفرعى باستدعاء الدالة GetTime ويعرض النتيجة المسترجعة بواسطتها فى صندوق رسالة.

٦. وأخيرا أضف السطر التالى لإجراء الحدث Form1.Load وهو يقوم باستدعاء الإجراء الفرعى DisplayTime كما بالشكل التالى:

```
DisplayTime( )
```

٧. اضغط F5 لتشغيل البرنامج.

عندما يبدأ البرنامج، يتم تنفيذ إجراء الحدث Form1.Load. يقوم هذا الإجراء باستدعاء الإجراء الفرعى DisplayTime، ولذلك ينتقل تنفيذ الكود إلى الإجراء الفرعى DisplayTime. وفى المقابل يقوم هذا الإجراء باستدعاء الدالة GetTime. تسترجع هذه الدالة القيمة النصية المقابلة للتوقيت إلى الإجراء الفرعى DisplayTime، الذى يعرض هذه القيمة النصية فى صندوق رسالة. وبعد انتهاء عملية تنفيذ هذا الإجراء، يستكمل البرنامج العمل بشكل طبيعى، ويعرض النموذج.

٥-١-٤ البيانات الضرورية للدوال والإجراءات الفرعية

فى بعض الأحيان، سوف تحتاج إلى توفير معلومات إضافية للإجراءات التى تقوم بإنشائها. على سبيل المثال، فى الإجراء PlaySound قد تحتاج إلى تشغيل ملف معين من عدة ملفات صوتية. ومن هنا يمكنك تزويد الإجراء بمعلومات حول هذا الملف عن طريق خاصية المعلومات الضرورية parameters.

تشبه الـ parameters كثيرا المتغيرات variables. كل منهما له نوع واسم، كما يقومان بتخزين معلومات. ومن هنا تستخدم الـ parameters كمتغيرات فى الإجراءات الفرعية procedures. لكن هناك فرقين أساسيين بين الـ parameters والمتغيرات:

- يتم الإعلان عن الـ parameters من خلال الإعلان عن الإجراء procedure declaration، وليس فى أسطر منفردة من الكود.
- الـ parameters قابلة للاستخدام فقط داخل الإجراء المرتبطة به.

يتم تحديد المعلومات الضرورية بين الأقواس الهلالية بعد اسم الإجراء وأثناء الإعلان عن الإجراء. وتستخدم الكلمة المفتاحية As قبل تحديد النوع type، ويسبق كل parameter الكلمة المفتاحية ByVal. يقوم VB بإضافة هذه الكلمة أوتوماتيكيا فى حالة عدم كتابتها من قبل المبرمج. انظر المثال التالى لإجراء فرعى Sub يحتوى على parameters:

```
Sub PlaySound(ByVal SoundFile As String, ByVal Volume As_
Integer)
  My.Computer.Audio.Play(SoundFile, Volume)
End Sub
```

وبعد ذلك، يجب عليك استدعاء الإجراء الفرعى مزودا بقيم الـ parameters:

```
PlaySound("Startup.wav", 1)
```

يتم تحديد الـ parameters بالنسبة إلى الدوال functions بالطريقة المتبعة ذاتها مع الإجراءات الفرعية subs.

تدريب عملى

١. من قائمة File اختر New Project

٢. اختر Windows Application واكتب اسم البرنامج parameters

٣. من صندوق الأدوات Toolbox ضع صندوقى نص TextBox 2 وزر أمر 1 button

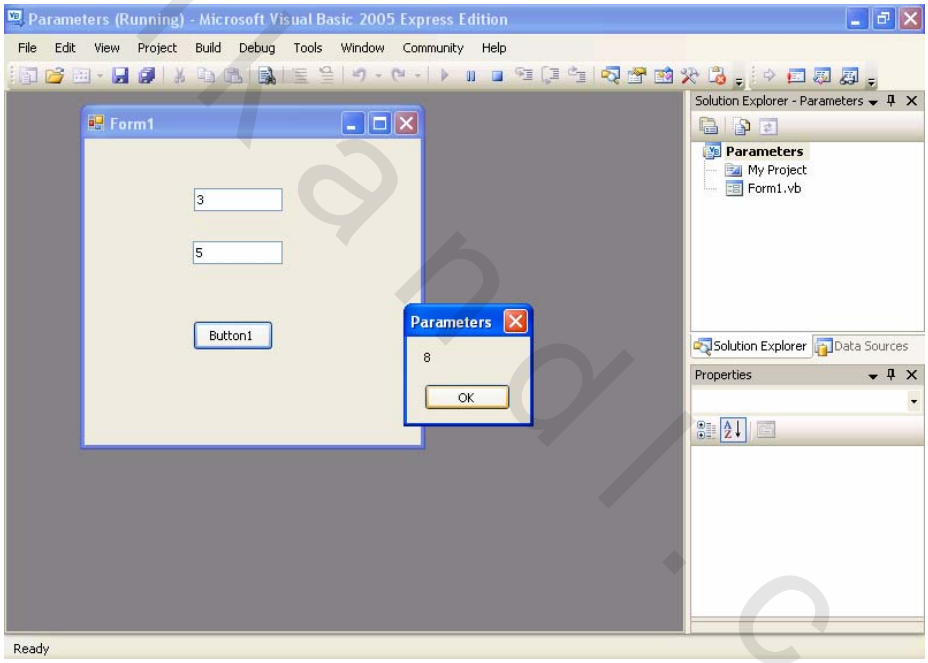
٤. مباشرة بعد عبارة End Sub الخاصة بإجراء الحدث Button1.Click أضف الإجراء التالى:

```
Function AddTwoNumbers(ByVal N1 As Integer, ByVal N2_ As
Integer) As Integer
  Return N1 + N2
End Function
```

٥. وفى الإجراء Button1.Click اكتب الكود التالى:

```
Dim aNumber As Integer = CInt(Textbox1.Text)
Dim bNumber As Integer = CInt(Textbox2.Text)
MsgBox(AddTwoNumbers(aNumber, bNumber))
```

فى هذا الكود، أعلنا عن رقمين صحيحين integers وقمنا بتحويل النص الموجود فى صندوقى النص إلى قيم رقمية صحيحة. وبعد ذلك قمنا بتمرير هذه القيم إلى الدالة AddTwoNumbers مع عرض النتيجة فى صندوق رسالة.



شكل (٥-١) استخدام Parameters مع الدوال والإجراءات

٦. اضغط F5 لتشغيل البرنامج.

٧. اكتب قيمة رقمية فى كل من صندوقى النص، واضغط زر الأمر. تلاحظ ظهور نتيجة جمع الرقمين فى صندوق رسالة.

٥ـ٢ التكرار Looping

فى هذا الجزء، سوف نتعرف معا ـ عزيزى القارئ ـ على كيفية استخدام جملة For Next ... لتكرار تعليمات معينة فى البرنامج. على سبيل المثال، افترض أننا نريد كتابة برنامج يعرض سلسلة من الأرقام على الشاشة. يجب أن تحدد للبرنامج ضرورة تكرار (سكر) معين من الكود عدد من المرات.

تسمح الحلقة التكرارية For ... Next تحديد رقم، ثم تكرار الكود الموجود داخل الحلقة عددا من المرات يساوى هذا الرقم. انظر المثال التالى:

```
Dim i As Integer = 0

For i = 1 To 10

DisplayNumber(i)

Next
```

فى المثال السابق، بدأت الحلقة التكرارية بالمتغير العداد i. تستخدم الحلقة التكرارية هذا المتغير لمعرفة عدد مرات تنفيذ الكود الموجود داخل الحلقة. وفى السطر التالى (For i = 1 to 10) عدد مرات تكرار الحلقة، والقيم التى يأخذها المتغير i.

عندما يصل الكود إلى داخل الحلقة التكرارية، يأخذ المتغير i أولى القيم (فى هذه الحالة ١). يقوم البرنامج بعد ذلك بتنفيذ الكود الموجود بين For و Next؛ حيث يقوم باستدعاء الدالة DisplayNumber مع المعلومات الضرورية parameter الخاصة بالمتغير I (فى هذه الحالة أيضا الرقم ١).

وعندما يصل الكود إلى سطر Next، تتم إضافة ١ إلى I ثم يعود التنفيذ مرة أخرى إلى سطر For. وهنا يتكرر الكود حتى تصبح قيمة I أكبر من الرقم الثانى فى

سطر For، فى هذه الحالة رقم ١٠. وعندما يحدث ذلك، ينتقل تنفيذ الكود إلى السطر التالى لجملة Next.

تدريب عملى

١. من قائمة File اختر New Project

٢. اختر Windows Application واكتب اسم البرنامج ForNext

٣. من صندوق الأدوات Toolbox ضع صندوق نص TextBoxوزر أمر Button

٤. اضغط مرتين على زر الأمر لفتح نافذة الكود.

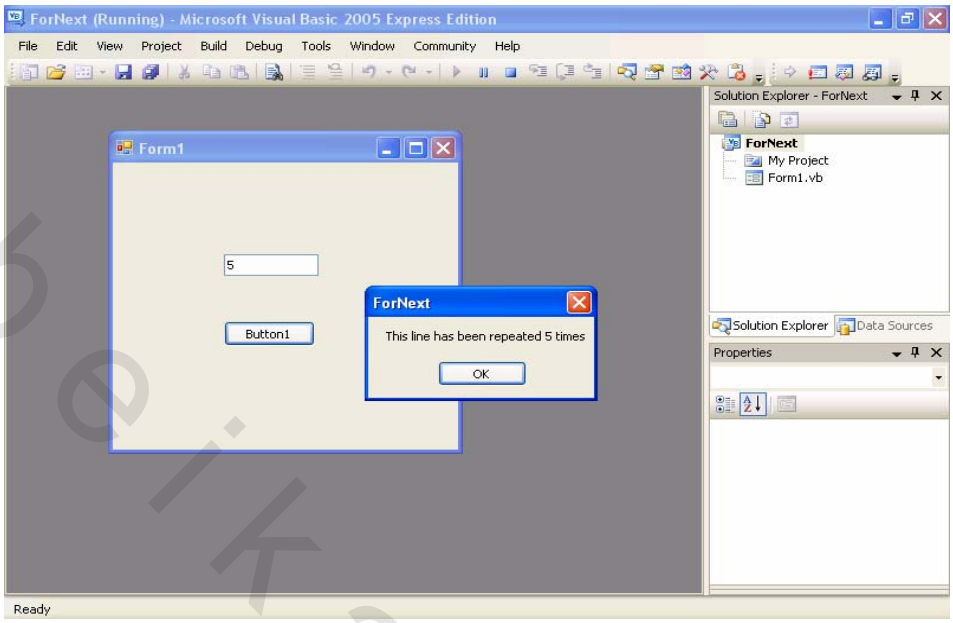
٥. اكتب الكود التالى فى إجراء الحدث Button1.Click

```
Dim i As Integer = 0

Dim NumberOfRepetitions As Integer = CInt(Textbox1.Text)

For i = 1 To NumberOfRepetitions

  MsgBox("This line has been repeated " & i & " times")

Next
```

٦. اضغط F5 لتشغيل البرنامج.

٧. اكتب رقما ما فى صندوق النص، ثم اضغط زر Ok

٨. يظهر صندوق الرسالة عددا من المرات يعادل الرقم الموجود فى صندوق النص.



شكل (٥-٢) التكرار بجملة For .. Next

٥-٢-١ التكرار باستخدام Do While

فى هذا الجزء، سوف نتعرف معا ـ عزيزى القارئ ـ على جمل أخرى لتنفيذ الحلقات التكرارية بلغة Visual Basic. فقد تعرفنا ـ فى الجزء السابق ـ على جملة For Next ... التى تقوم بتنفيذ الكود عددا معينا من المرات. لكن ماذا لو اختلف عدد مرات التكرار هذه حسب توافر شروط معينة؟.

من هنا تأتى أهمية جملة Do While التى تسمح بتنفيذ الكود الموجود داخل الحلقة التكرارية مادام While كان شرطا معينا صحيحا True. وهناك أيضا جملة Do Until التى تسمح بتكرار الكود الموجود داخل الحلقة حتى Until ليتحقق شرط معين True.

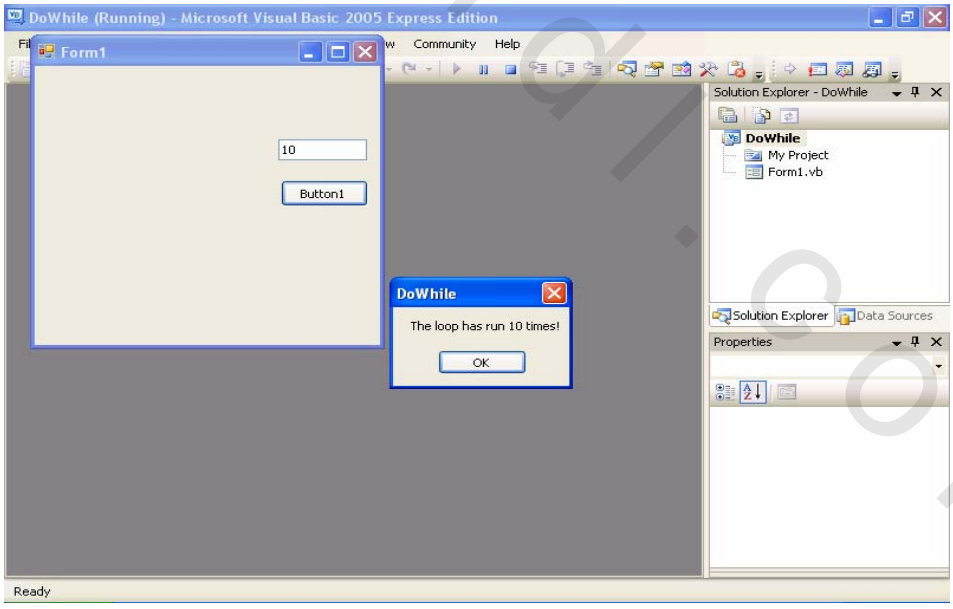
على سبيل المثال، افترض أن هناك برنامجا يقوم بجمع سلسلة من الأرقام، بشرط ألا يتجاوز حاصل الجمع الرقم ١٠٠. فى هذه الحالة يمكن استخدام جملة Do While على النحو التالى:

```
Dim sum As Integer = 0
Do While sum < 100
sum = sum + 10
Loop
```

فى المثال السابق، يقوم سطر Do While بالتحقق من المتغير sum إذا كان أقل من ١٠٠. فإذا كان الأمر كذلك، يتم تنفيذ السطر التالى من الكود. أما إذا كان الأمر غير ذلك (أى كانت قيمة sum أكبر من أو تساوى الرقم ١٠٠)، فإن الكود سوف ينتقل إلى السطر التالى لجملة Loop. أما وظيفة الكلمة المفتاحية Loop فهى تحويل تنفيذ الكود مرة أخرى إلى بداية الحلقة التكرارية (سطر Do While) للتحقق من القيمة الجديدة للمتغير sum.

تدريب عملى

١. من قائمة File اختر New Project
٢. اختر Windows Application واكتب اسم البرنامج DoWhile
٣. ضع صندوق نص TextBox وزر أمر Button على سطح النموذج



شكل (٥-٣) التكرار باستخدام جملة Do While

٤. اضغط مرتين على زر الأمر لفتح نافذة الكود.

٥. اكتب الكود التالى فى إجراء الحدث Button1.Click.

```
Dim sum As Integer = 0
Dim counter As Integer = 0
Do While sum < 100
 sum = sum + CInt(Textbox1.Text)
counter = counter + 1
Loop
MsgBox("The loop has run " & CStr(counter) & " times!")
```

٦. اضغط F5 لتشغيل البرنامج.

٧. اكتب رقما معينا فى صندوق النص، ثم اضغط زر الأمر.

٨. يظهر صندوق رسالة، ويعرض عدد المرات التى أضيف فيها الرقم إلى نفسه، قبل أن يصل إلى الرقم ١٠٠.

٩. من قائمة Debug اختر Stop Debugging لإيقاف البرنامج. اترك البرنامج مفتوحا كما هو، فسوف نضيف إليه فى الجزء التالى.

٥ـ٢ـ٢ التكرار باستخدام Do Until

تقوم جملة DoWhile بتنفيذ الحلقة التكرارية مادام قد ظل الشرط صحيحا True ، لكن هناك حالات معينة قد تريد فيها تنفيذ الحلقة التكرارية حتى يصبح الشرط صحيحا. وفى هذه الحالة يبرز استخدام جملة DoUntil على النحو التالى:

```
Dim sum As Integer = 0
Do Until sum >= 100
 sum = sum + 10
Loop
```

يشبه الكود السابق كود Do While أعلاه، فيما عدا أنه فى هذه المرة يقوم الكود بالتحقق من المتغير sum إذا كان يساوى أو أكبر من الرقم ١٠٠.

تدريب عملى

١. أضف الكود التالى تحت سطر MsgBox مباشرة:

```
Dim sum2 As Integer = 0
Dim counter2 As Integer = 0
Do Until sum2 >= 100
  sum2 = sum2 + CInt(Textbox1.Text)
  counter2 = counter2 + 1
Loop
MsgBox("The loop has run " & CStr(counter2) & " times!")
```

٢. اضغط F5 لتشغيل البرنامج.

٣. اكتب رقما معينا فى صندوق النص، ثم اضغط زر الأمر. يظهر صندوق رسالة ثان، ويعرض عدد مرات إضافة الرقم إلى نفسه قبل أن يساوى ١٠٠ أو أكثر.

٣ـ٥ اتخاذ القرار Decision Making

فى هذا الجزء، سوف نتعرف معا ـ عزيزى القارئ ـ على استخدام جملة If ... Then لتنفيذ كود معين بناء على شرط ما.

تحتاج البرامج إلى تنفيذ تعليمات مختلفة استجابة لشروط معينة. على سبيل المثال، قد تريد من البرنامج التحقق من اليوم بين أيام الأسبوع، ثم تنفيذ تعليمات مختلفة على حسب اليوم. ومن هنا تأتى جملة If ... Then التى تتحقق من شرط معين، ثم تقوم بتنفيذ تعليمات مختلفة على حسب نتيجة التحقق من ذلك الشرط.

يوضح المثال التالى استخدام جملة If ... Then:

```
If My.Computer.Clock.LocalTime.DayOfWeek = _
DayOfWeek.Monday Then
    MsgBox("Today is Monday!")
End If
```

عند تنفيذ هذا الكود، يتم التحقق من الشرط (الواقع بين كلمتى If و Then)؛ فإذا كان صحيحا True، يتم تنفيذ السطر التالى من الكود؛ حيث يظهر صندوق الرسالة؛ أما إذا كان خاطئا False، فإن الكود ينتقل إلى سطر End If. وبعبارة أخرى، ينص الكود على أنه إذا كان اليوم هو الاثنين، فاعرض، إذن، الرسالة.

تدريب عملى

١. من قائمة File اختر New Project.

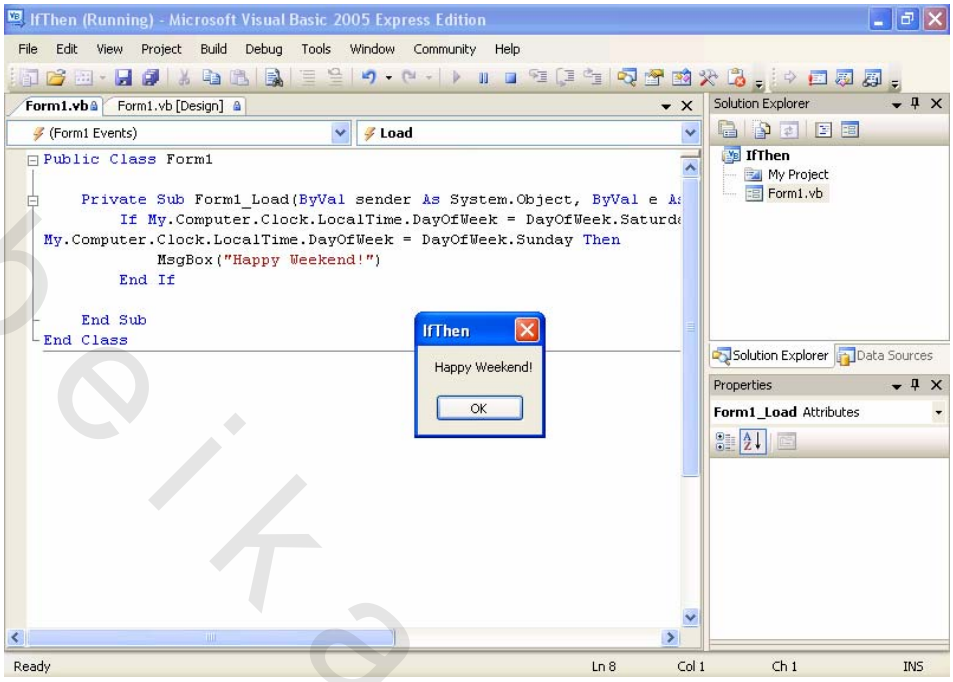
٢. اختر Windows Application واكتب اسم البرنامج IfThen.

٣. اضغط مرتين على سطح النموذج لفتح نافذة الكود.

٤. فى إجراء الحدث Form1.Load اكتب الكود التالى:

```
If My.Computer.Clock.LocalTime.DayOfWeek =
DayOfWeek.Saturday Or _
   My.Computer.Clock.LocalTime.DayOfWeek =
DayOfWeek.Sunday Then
    MsgBox("Happy Weekend!")
End If
```

٥. اضغط F5 لتشغيل البرنامج.



شكل (٥ـ٤) جملة If Then لاتخاذ القرار

٦. إذا كان اليوم هو السبت أو الأحد، تظهر رسالة Happy Weekend! ، وإذا كان الأمر غير ذلك، فلن يظهر شيء.

٧. من قائمة Debug اختر Stop Debugging لإنهاء البرنامج. اترك المشروع مفتوحا كما هو، فسوف نضيف إليه فى الجزء التالى.

ربما تكون قد لاحظت ـ عزيزى القارئ ـ من خلال المثال السابق أن جملة If ... Then قد استخدمت المعامل Or للتحقق من شروط متعددة. ويمكن للمبرمج استخدام المعاملات Or و And للتحقق من شروط متعددة، داخل جملة If ... Then

٥ـ٣ـ١ جملة Else

لقد رأيت ـ عزيزى القارئ ـ استخدام جملة If...Then لتنفيذ الكود إذا تحقق الشرط، لكن ماذا لو أردت تنفيذ كود معين إذا تحقق الشرط، وتنفيذ كود آخر إذا لم

يتحقق هذا الشرط؟. فى هذه الحالة، يمكنك استخدام عبارة Else. تسمح هذه العبارة بتخصيص كود معين يتم تنفيذه فى حالة عدم تحقق الشرط. انظر المثال التالى:

```
If My.Computer.Clock.LocalTime.DayOfWeek =
DayOfWeek.Friday Then
  MsgBox("Today is Friday!")
Else
  MsgBox("It isn't Friday yet!")
End If
```

فى هذا المثال، إذا تحقق الشرط، يتم تنفيذ السطر التالى لكلمة Then: حيث تعرض رسالة !Today is Friday، وإذا لم يتحقق، يتم تنفيذ السطر التالى لكلمة Else: حيث تظهر رسالة !Today isn't Friday yet.

تدريب عملى

١. غير الكود الموجود فى جملة If … Then بحيث يصبح على النحو التالى:

```
If My.Computer.Clock.LocalTime.DayOfWeek =
DayOfWeek.Saturday Or _
My.Computer.Clock.LocalTime.DayOfWeek =
DayOfWeek.Sunday Then
  MsgBox("Happy Weekend!")
Else
  MsgBox("Happy Weekday! Don't work too hard!")
End If
```

٢. اضغط F5 لتشغيل البرنامج. سوف يعرض البرنامج الآن رسالة تحدد ما إذا كان اليوم هو يوم عطلة Weekend أو يوما من أيام الأسبوع Weekday.

ملحوظة

يمكنك تغيير اليوم بالضغط مرتين على الساعة الموجودة فى الركن الأيمن أسفل الشاشة؛ وذلك لتجربة الكود الموجود فى البرنامج السابق.

٥ـ٣ـ٢ جملة Select Case

هناك شكل آخر لاتخاذ القرار داخل كود البرنامج فى حالة وجود شروط متعددة، وهو جملة Select Case. فعلى الرغم من أنه يمكن تقييم أكثر من شرط باستخدام جملة If ... Then من خلال العبارة ElseIf؛ فإن جملة Select Case توفر طريقة أفضل لهذا الغرض.

تسمح جملة Select Case بالتحقق من شروط (حالات Cases) كثيرة كيفما تريد، الأمر الذى يسهل عملية كتابة الكود فى حالة وجود خيارات متعددة. على سبيل المثال، افترض أن البرنامج يستخدم متغيرا من نوع String لتخزين اللون الذى اختاره المستخدم، وتحتاج أنت إلى التعرف على هذا اللون من خلال الكود. فى هذه الحالة يمكن كتابة الكود على النحو التالى:

```
Select Case Color
  Case "red"
    MsgBox("You selected red")
  Case "blue"
    MsgBox("You selected blue")
  Case "green"
    MsgBox("You selected green")
End Select
```

عند تنفيذ هذا الكود، يقوم سطر Select Case بتحديد قيمة اللون، ومقارنتها بالقيمة الموجودة فى أول حالة. إذا كانت هناك مطابقة، يتم تنفيذ السطر التالى

مباشرة من الكود، ثم ينتقل التنفيذ إلى جملة End Select. وإذا لم تحدث المطابقة، ينتقل التنفيذ إلى الحالة الثانية للمقارنة، وهكذا.

يمكن لجملة Case أن تأخذ أشكالا مختلفة. ففى المثال السابق، الجملة عبارة عن String. إلا أنها يمكن أن تتعامل مع أى نوع بيانات.

على سبيل المثال، يمكن التحقق من مجموعة أرقام، باستخدام الكلمة المفتاحية To على النحو التالى:

```
Case 1 To 10
```

فى المثال السابق، تحدث المطابقة إذا كان الرقم بين ١ و ١٠.

كما يمكن، كذلك، تقييم خيارات متعددة فى جملة Case واحدة؛ وذلك بالفصل بين هذه الخيارات بفاصلة، على النحو التالى:

```
Case "red", "white", "blue"
```

فى المثال السابق، تحدث المطابقة إذا كانت القيمة أيا من الخيارات الثلاثة.

وكذلك يمكن استخدام المقارنات مع الكلمة المفتاحية Is على النحو التالى:

```
Case Is > 9
```

فى المثال السابق، تحدث المطابقة إذا كان الرقم أكبر من ٩.

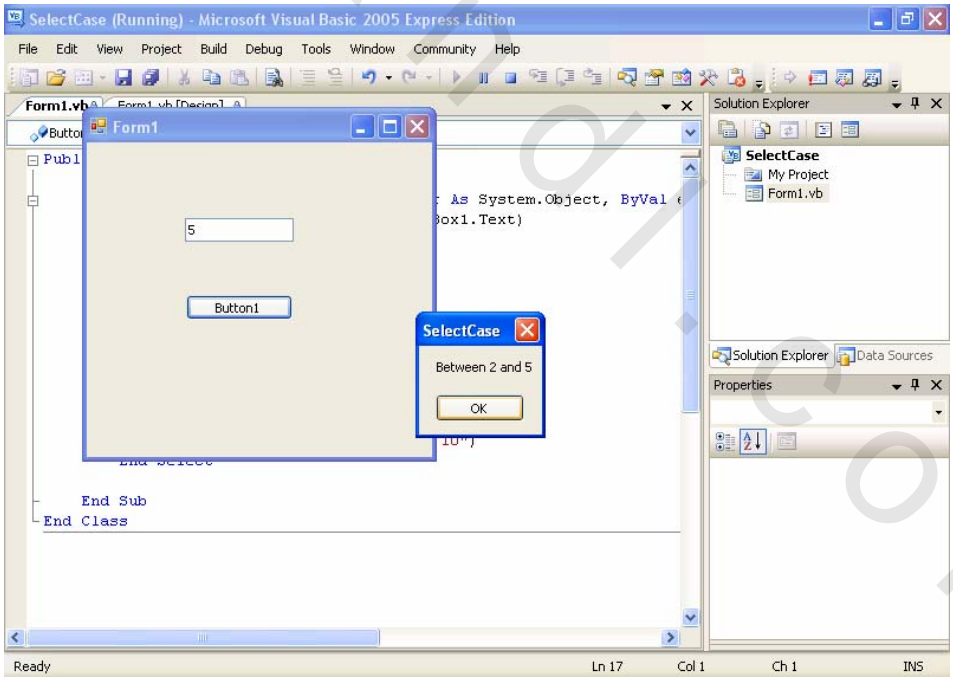
٥ـ٣ـ٣ جملة Case Else

يستخدم المثال السابق عند معرفة جميع الحالات الممكنة. لكن ماذا لو كان هناك شرط غير معروف؟ على سبيل المثال، إذا كان اللون أصفر، يقوم الكود بتقييم الحالات الثلاث من دون الحصول على مطابقة، ومن ثم لن يظهر صندوق الرسالة.

تستخدم جملة Case Else فى تنفيذ كود معين فى حالة عدم وجود مطابقة، وذلك على النحو التالى:

```
Select Case Color
  Case "red"
     MsgBox("You selected red")
  Case "blue"
     MsgBox("You selected blue")
  Case "green"
     MsgBox("You selected green")
  Case Else
     MsgBox("Please choose red, blue, or green")
End Select
```

فى المثال السابق، إذا كان اللون أصفر، يقوم الكود بمقارنته مع أسطر Case الثلاثة الأولى، من دون الحصول على مطابقة، وعند الوصول إلى سطر Case Else يتم تنفيذ السطر التالى من الكود قبل الانتقال إلى سطر End Select.



شكل (٥-٥) جملة Select Case لاتخاذ القرار

تدريب عملى

١. من قائمة File اختر New Project.

٢. اختر Windows Application واكتب اسم البرنامج SelectCase.

٣. ضع صندوق نص TextBox وزر أمر على سطح النموذج.

٤. اضغط مرتين على زر الأمر لفتح نافذة الكود.

٥. فى إجراء الحدث Button1.Click اكتب الكود التالى:

```
Dim Number As Integer = CInt(Textbox1.Text)
Select Case Number
  Case 1
    MsgBox("Less than 2")
  Case 2 To 5
    MsgBox("Between 2 and 5")
  Case 6, 7, 8
    MsgBox("Between 6 and 8")
  Case 9 To 10
    MsgBox("Greater than 8")
  Case Else
    MsgBox("Not between 1 and 10")
End Select
```

٦. اضغط F5 لتشغيل البرنامج.

٧. اكتب رقما معينا فى صندوق النص، ثم اضغط زر الأمر.

٨. تظهر رسالة توضح الرقم الذى كتبته، على حسب على جملة Case.

٥ـ٤ معالجة الأخطاء Handling Errors

فى هذا الجزء سوف نتعرف معا ـ عزيزى القارئ ـ على كيفية التعامل مع الأخطاء من خلال الكود.

حتى أفضل البرامج تصميما لا تخلو من الأخطاء. فهناك أخطاء فى كتابة الكود، وهذه يسهل التعرف عليها وعلاجها. وهناك أخطاء أخرى تعتبر طبيعية فى البرامج، كأن يحاول البرنامج مثلا فتح ملف، ويكون هذا الملف مفتوحا أصلا. فى مثل هذه الحالات، من الممكن توقع الأخطاء، لكن لا يمكن منعها. ومن ثم فإن إحدى وظائف المبرمج الناجح هى توقع الأخطاء، والعمل على علاجها مسبقا من خلال الكود.

٥ـ٤ـ١ أخطاء وقت التشغيل

يطلق على الخطأ الذى يحدث أثناء تشغيل البرنامج "خطأ وقت التشغيل run-time error". تحدث مثل هذه الأخطاء عندما يحاول البرنامج تنفيذ مهام معينة ليست مصممة للقيام بها. على سبيل المثال عندما يحاول البرنامج القيام بتنفيذ عملية غير منطقية؛ كتحويل قيمة غير رقمية إلى قيمة رقمية .. هنا يحدث خطأ أثناء التنفيذ.

عند حدوث مثل هذه الأخطاء، يصدر البرنامج استثناء exception. تقوم الاستثناءات بالبحث عن الكود الخاص بالتعامل مع الخطأ؛ فإن لم تجده، يتوقف البرنامج، ويجب إعادة تشغيله من جديد. وحيث إن ذلك قد يؤدى إلى فقدان البيانات، فمن الأهمية بمكان كتابة الكود الخاص بالتعامل مع الأخطاء، حيثما يتوقع المبرمج حدوثها أو وقوعها.

٥ـ٤ـ٢ جملة Try…Catch…Finally

تستخدم جملة Try…Catch…Finally للتعامل مع أخطاء وقت التنفيذ من خلال الكود، حيث تتم أولا محاولة Try تنفيذ تعليمات برمجة معينة، فإذا حدث استثناء ما، ينتقل التنفيذ إلى التعليمات الأخرى الموجودة بعد جملة Catch، وبعد

الانتهاء من ذلك، يتم تنفيذ أية تعليمات برمجية موجودة بعد جملة Finally. تنتهى جملة Try…Catch…Finally بعبارة End Try. والمثال التالى يوضح استخدام هذه الجملة:

```
Try
  ' Code here attempts to do something.
Catch
  ' If an error occurs, code here will be run.
Finally
  ' Code in this block will always be run.
End Try
```

فى البداية، يقوم البرنامج بتنفيذ الكود الموجود فى جملة Try. فإذا تم التنفيذ من دون أخطاء، يتخطى البرنامج المقطع Catch ليقوم بتنفيذ الكود الموجود فى جملة Finally. أما إذا حدث خطأ ما عند تنفيذ جملة Try، فينتقل التنفيذ مباشرة إلى المقطع Catch ويقوم بتنفيذ الكود الموجود داخله، وبعد ذلك يتم تنفيذ الكود الموجود بجملة Finally.

تدريب عملى

١. من قائمة File اختر New Project
٢. اختر Windows Application واكتب اسم البرنامج TryCatch
٣. ضع صندوق نص TextBox وزر أمر Button على سطح النموذج.
٤. اضغط مرتين على زر الأمر لفتح نافذة الكود.
٥. اكتب الكود التالى فى إجراء الحدث Button1.Click:

```
Try
  Dim aNumber As Double = CDbl(Textbox1.Text)
  MsgBox("You entered the number " & aNumber)
Catch
  MsgBox("Please enter a number.")
Finally
```

```
MsgBox("Why not try it again?")
End Try
```

٦. اضغط F5 لتشغيل البرنامج.

٧. اكتب رقما معينا فى صندوق النص، ثم اضغط زر الأمر. تظهر رسالة توضح الرقم الذى كتبته. تتبعها رسالة أخرى تدعوك إلى تكرار المحاولة.

٨. اكتب قيمة غير رقمية فى صندوق النص، كلمة مثلا، ثم اضغط زر الأمر. فى هذه المرة، عندما يحاول البرنامج تحويل النص الموجود بصندوق النص إلى قيمة رقمية، لن يستطيع ذلك، وهنا يحدث الخطأ. وبدلا من توقف التنفيذ، يقوم البرنامج بتنفيذ الكود الموجود فى المقطع Catch، ومن ثم تظهر رسالة تطلب منك إدخال رقم. وأخيرا يتم تنفيذ الكود الموجود بجملة Finally، والذى يظهر رسالة تدعوك إلى تكرار المحاولة.

إلى هنا ـ عزيزى القارئ ـ نكون قد وصلنا معا إلى نهاية هذا الفصل. أرجو أن تكون قد بدأت تشعر بالمواطنة فى هذه الأرض الجديدة. الفصل التالى يأخذك فى جولة ترويحية فى واجهة التعامل مع المستخدم user interface. فكن معنا حيث نقترب أكثر وأكثر من لحظات حاسمة مع الفصل السابع – بعد التالى – حيث محاولة ترويض لغة Visual Basic.Net للتعرف على الطرق المختلفة لبرمجة قواعد البيانات.

ملخص

فى هذا الفصل، قدمنا لك ـ عزيزى القارئ ـ بعضا من أساليب البرمجة المتقدمة، وكيفية استخدامها فى تطبيقات Visual Basic. وكان التركيز على موضوعات الدوال Functions والإجراءات الفرعية Procedures أو Subs، وكذلك جمل التكرار الأساسية؛ مثل For … Next و While … Do و Until … Do. هذا فضلا عن اتخاذ القرار داخل الكود؛ وفيه تعرفنا على جملة If… Then الشهيرة، وأيضا جملة Select…Case لاتخاذ قرار من بين عدة بدائل.

كما ألقينا الضوء على موضوع مهم بالنسبة إلى جميع المبرمجين المحترفين؛ وهو معالجة الأخطاء الموجودة بالكود. وجدير بالذكر هنا أن الأسلوب الجديد الذى قدمته لغة Visual Basic .NET وهو جملة Try .. Catch ..Finally للتعامل مع الاستثناءات exceptions داخل كود البرنامج، قد قدمت أسلوبا راقيا وفعالا فى الوقت ذاته؛ لمعالجة الأخطاء، مقارنة بجملة GoTo … OnError التى اعتمدتها عائلة BASIC على مدار سنوات طويلة.

تمارين

١. وضع الفرق بين الدوال Functions والإجراءات الفرعية Subs.

٢. عند الرغبة فى تنفيذ تعليمات برمجية معينة عددا محددا من المرات، يمكن الاعتماد على جملة .....

٣. أى من الجملتين التاليتين يتم استخدامها فى التكرار حتى يتحقق شرط معين (اختر واحدة):

a. جملة Do … While

b. جملة Do … Until

٤. هل يمكن استخدام جملة If … Then فى اتخاذ قرار من بين عدة بدائل؟.

٥. يعتقد بعض المبرمجين أنه يجب استخدام جملة Try .. Catch .. Finally فى جميع البرامج. ناقش.