في الفصول السابقة تحدثنا عن مقدمة إلى لغـــة **Access VBA** *وكيفية استخدامها لكتابة التعليمات ، وفي هـــذا الفصـــل نشرح كيفية تصحيح أخطاء البرنامج. ولأن الأخطاء عادة تكتشف عند تنفيذ البرنامج فســـنبدأ بشـــرح متـــرجم* **Access VBA** *واستخدام الإطار المباشر للحصول على نتيجة التنفيذ. بانتهاء هذا الفصل سنتعرف على :*

- *مترجم* **Access VBA**
- *استخدام النافذة الفورية*
- *تصحيح أخطاء البرنامج*
- *استخدام نقطة إيقاف لإرجاء تنفيذ البرنامج*
- *الخطو خلال البرنامج*
- *استخدام أمر* **On Error** *لتصحيح أخطاء وقت التشغيل*

مترجم Access VBA

تستخدم لغات البرمجة التقليدية مثل لغةPascal أو++ C ما يسمى Compiler أو مترجم ليقوم بترجمة التعليمات التى يكتبها المبرمج من "اللغة المصدرية" وتسمى Source Code إلي اللغة التى يفهمها الكمبيوتر وتسمي Object Code أو "لغة الهدف". بعد عملية الترجمة من اللغة المصدرية التي يستخدمها المبرمج إلى لغة الهدف التى يفهمها الكمبيوتر ، يتطلب الأمر ربط البرنامج مع واحدة من المكتبات الموجودة عادة ضمن لغة البرمجة في صورة Object للحصول على ملف جاهز قابل للتنفيذ يسمى Executable file . تستخدم Access أيضا مكتبات تقوم بنفس الوظيفة ، من هذه المكتبات WIZARD.MDA . والملف التنفيذى يخصص له عادة الامتداد EXE. ويمكن تشغيله كتطبيق مستقل لا يحتاج وجود لغة البرمجة التي كتب بها في الأصل. تستخدم اللغات الحديثة مثل Dbase ما يسمى Interpreter أو المفسر لتنفيذ التعليمات التى تكتب باللغة المصدرية. ويقوم المفسر بقراءة التعليمات وترجمتها إلى تعليمات يفهمها الكمبيوتر ، وبعد ذلك تطلب من الكمبيوتر تنفيذ التعليمات.

والفرق بين الحالتين ، حالة استخدام المترجم Compiler مع بعض اللغات ، وحالة استخدام المفسر Interpreter مع لغات أخرى يتضح عندما تعرف أن البرامج التى تستخدم المفسر لا يمكن تشغيلها باستقلالية عن اللغة التى كتبت بها ، فمثلا برامج Qbasic لابد من تشغيلها مع وجود لغة Qbasic . ومن مزايا استخدام المفسر أنه يراجع القواعد اللازمة لكتابة التعليمات بمجرد كتابتها وينبه عن الخطأ في كتابة التعليمات إن وجد ، وهذا بعكس اللغات التى تستخدم المترجم ، حيث لا يمكن اكتشاف أخطاء كتابة التعليمات إلا بعد ترجمتها. وهذا معناه أن التعديلات التى تتم على البرامج التي تستخدم المفسر تتم على اللغة المصدرية فقط دون حاجة لترجمتها وربطها مع المكتبات مرة أخرى ، أما إذا أردت تعديل برنامج يحتاج إلى مترجم فلابد من تعديل التعليمات المصدرية ثم عمل ترجمة (Compiling) وربط (Linking) للبرنامج بعد

إجراء أى تعديل على لغة المصدر . أما عن عيوب استخدام المفسر ، فهي بطء تنفيذ البرامج مقارنة بالبرامج التي تستخدم المترجم ، والموجودة في صورة قابلة للتنفيذ.

تستخدم Access الملامح الموجودة في كل من المفسر والمترجم ، حيث يراجع المفسر التعليمات التي تكتبها بمجرد إنهاء السطر بالضغط على مفتاح Enter ويكتشف أخطاءها ويصححها إن أمكن أو ينبهك عن الخطأ في تركيب التعليمة. وبعد الانتهاء من كتابة تعليمات المصدر Source Code تقوم Access بترجمة هذه التعليمات في صورة تجمع بين التعليمات المفسرة (Interpreted Code) وتعليمات الهدف (Object Code)، تسمى pseudo-code، ويتم ترجمة التعليمات بهذه الطريقة عند استخدامها لأول مرة داخل برامجك، وفي مرحلة الترجمة يتم اكتشاف الأخطاء التي لا تكتشف عند كتابة التعليمات بواسطة المفسر. وميزة هذه الصورة أنها أسرع من التعليمات المفسرة في التنفيذ. ويمكنك ترجمة كل الإجراءات والبرامج مرة واحدة باختيار أمر Compile من قائمة Debug .

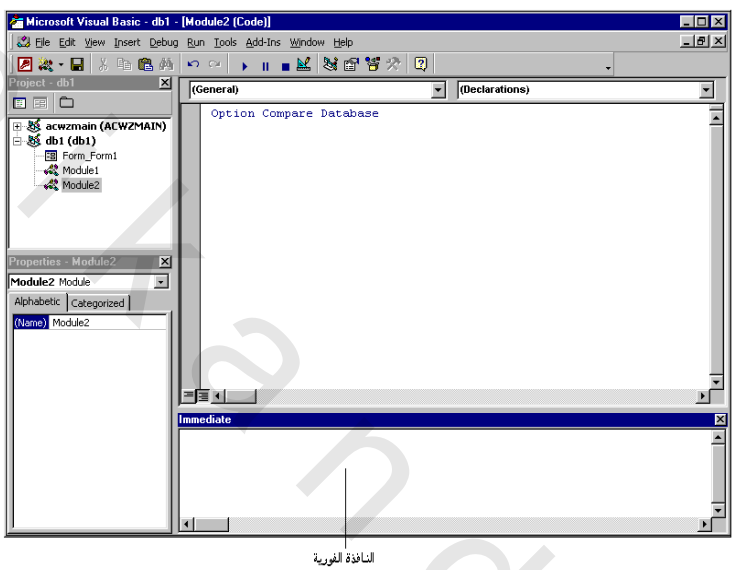
استخدام النافذة الفورية

عند كتابة برامج باستخدام Access VBA داخل وحدة نمطية، يمكنك استخدام النافذة الفورية للمساعدة في اكتشاف أخطاء كتابة التعليمات وتصحيحها. نوضح فيما يلى خطوات استخدام النافذة الفورية لتجربة بعض الدوال والتأكد من صحة استخدامها:

١. من نافذة قاعدة البيانات ومن مجموعة Other داخل التبويب Create انقر زر الوحدة النمطية Modules ، تقوم Access بفتح نافذة وحدة نمطية جديدة وتخصص لها اسم Module1 إذا كانت هذه أول مرة تفتح نافذة وحدة نمطية جديدة خلال جلسة العمل.

٢. انقر زر تكبير النافذة إذا لزم الأمر.

٣. إذا لم تظهر أمامك النافذة الفورية، افتح قائمة **View** ثم انقر أمر **Immediate window** لتنشيطه . تظهر النافذة الفورية كما في شكل ١-٤.



شكل ١-٤ النافذة الفورية داخل إطار الوحدة النمطية

٤. داخل الإطار المباشر للنافذة وحيث تقف نقطة الإدراج اكتب:

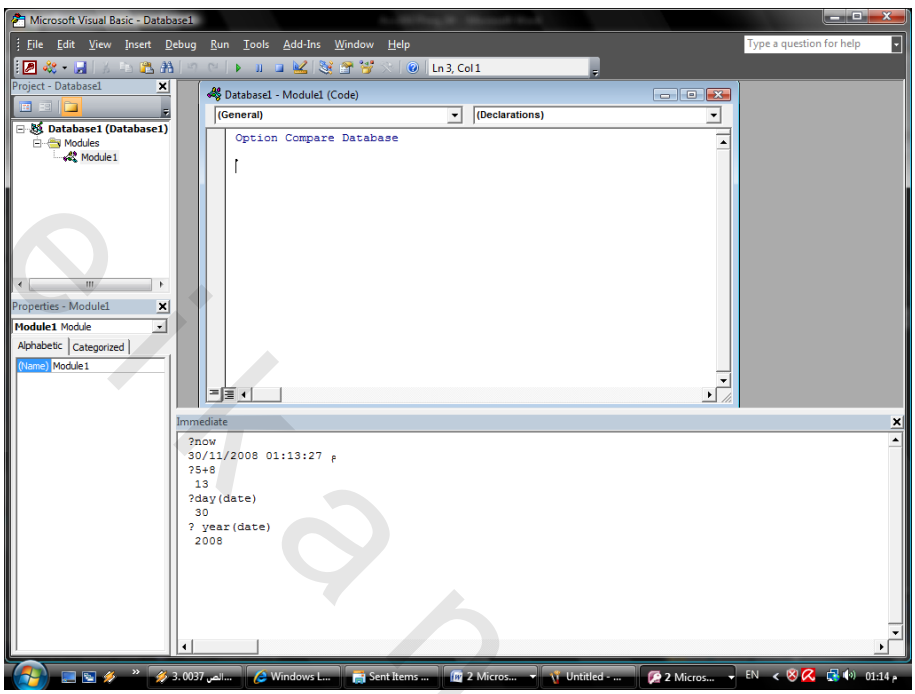
```
?Now( )
```

ثم اضغط مفتاح الإدخال

يظهر تاريخ اليوم والوقت الحالي المسجل بحاسبك في السطر التالي مباشرة.

علامة ? هنا أمر بمعنى اطبع (**Print**) وتستخدم لإظهار قيمة دالة أو متغير أو نتيجة تعبير. أما **NOW()** فهي إحدى الدوال المبنية داخل **Access** والتي سنشرحها فيما بعد ومعناها ما هو التاريخ والتوقيت الحالي. إذا لم يسبق الدالة أمر ? فستحصل على رسالة خطأ.

٥. أكمل كتابة الأوامر الموجودة بشكل ٢-٤



شكل ٢–٤ استخدام النافذة الفورية لتجربة الدوال والتعبيرات.

تصحيح أخطاء البرنامج

مهما كانت خبرتك بتعليمات **Access VBA** فلن يخلو الأمر من أخطاء تقع فيها وتحتاج لتصحيحها عندما تكتشفها. والتصحيح هو العملية التي تستخدمها لإيجاد الأخطاء وحلها في برنامج **Access VBA** ، هناك ثلاثة أنواع من الأخطاء التي قد تصادفها أثناء تشغيل برنامجك والتى يمكن بيانها كما يلى:

- *أخطاء وقت الترجمة*(**Compiling Error**) : وتحدث نتيجة لبناء البرنامج بشكل غير صحيح. فقد تنسي موازنة عبارتين (مثل **IF** و**End IF** أو **For** و**Next**)، وقد يكون لديك خطأ برمجة لا يراعي قواعد برمجة **Access VBA**(خطأ إملائي في كلمة، أو فاصل غير موجود، أو خطأ تطابق الكتابة). ومنها كذلك أخطاء بناء الجملة، وهي أخطاء النحو أو علامات الترقيم. وتتضمن هذه

الأخطاء الأقواس غير المتطابقة أو عدد غير صحيح من الوسائط التي تـــم تمريرها إلى دالة. مثل هذه الأخطاء تقوم بتصحيحها ثـــم إعادة الترجمة إلى أن يتم تنقيـــة البرنامج من الأخطاء.

- *الأخطاء المنطقية*(Logical Error): وتحدث عندما لا يعمل البرنامج التطبيقي بالشكل المنتظر مما يؤدي إلى نتائج غير صحيحة. هذه الأخطاء تتطلب منك جهدا كبيرا لأن Access VBA لا تنبهك عنها مثل الأخطاء التى تحدث وقت الترجمة .
- *أخطاء وقت التشغيل* (Run-Time Error) : وتحدث بعد بدء تشغيل البرنامج التطبيقي. وتتضمن أمثلة هذه الأخطاء العمليات غير القانونية، مثل القسمة علـــى صفر أو الكتابة إلى ملف غير موجود.

وفيما يلي نوضح ثلاث طرق لاكتشاف الأخطاء وتصـــحيحها ، وهـــي الطـــرق الـــتي يستخدمها معظم مبرمجي Access VBA وهي:

- *استخدام نقطة إيقاف (Breakpoint) لإرجاء تنفيذ البرنامج.*
- *الخطو خلال البرنامج.*
- *استخدام أمر On Error لتصحيح أخطاء وقت التشغيل.*

استخدام نقطة إيقاف لإرجاء تنفيذ البرنامج

تبدو أهمية استخدام نقطة التوقف عنـــدما تتبـــع تنفيـــذ برنـــامج Access VBAمكتوب بواسطة شخص آخر ، أو عند تتبع وتصحيح أخطاء برنامجك . عندما ترجئ تنفيذ برنامج Access VBA، يظل البرنامج يعمل ولكنه يتوقف أثناء التشـــغيل عنـــدما يصل إلي نقطة التوقف. ولإيقاف تنفيذ Access Basic، يمكنك ضبط نقطـــة إيقـــاف Breakpoint بإتباع الخطوات التالية:

١. في إطار الوحدة النمطية، انقل نقطة الإدراج إلى سطر من البرنامج لا يحتوى أساساً على نقطة إيقاف وتريد أن يتوقف تنفيذ البرنامج عندما يصل إليه.
٢. انقر بزر الفأرة أقصى يسار هذا السطر خارج إطار نافذة الكود أو اضغط مفتـــاح

F9 أو انقر زر نقطة التوقف  من قائمة Debug. تتولى Access تعيين نقطة إيقاف وعلامة ذلك أن خط السطر يتحول إلي أحمر غامق.

عند تشغيل البرنامج، إذا وصل البرنامج إلى السطر المحدد عنده نقطة الإيقاف، سيتوقف وسيظهر نافذة الوحدة النمطية ويظهر هذا السطر محاطا بمستطيل مضاء ليوضح لك أن هنا نقطة إيقاف.

لمسح نقطة الإيقاف، انقل نقطة الإدراج إلى سطر من البرنامج تم إعداد نقطة الإيقاف عليه، قم بتكرار نفس الخطوات السابقة.

الخطو خلال برنامج Access VBA

يمكن أن يساعدك الخطو داخل برنامج Access VBA على تعريف مكان حدوث الخطأ. كما يمكنك مشاهدة ما إذا كان كل سطر في البرنامج ينجز النتائج التي تتوقعها. تابع الخطوات التالية:

١. قم بإرجاء تنفيذ البرنامج. اتبع تعليمات إرجاء تنفيذ البرنامج التي شرحناها في البند السابق. يعرض Access سطر البرنامج في مكان إرجاء التنفيذ.

٢. قم بتنفيذ واحد مما يلي:

- للخطو خطوة واحدة فى كل مرة يتم فيها استدعاء إجراء بواسطة إجراء آخر(خطوة خاصة)، انقر فوق Step into  من قائمة Debug أو اضغط F8 .
- للخطو خطوة واحدة فى كل مرة مع تشغيل أي إجراء يتم استدعاؤه كوحدة واحدة (خطوة عادية)، انقر فوق Step Over  في قائمة Debug أو اضغط SHIFT+F8 .
- لتشغيل السطر الذي يسبق الإجراء الحالى، ثم التوقف بعد ذلك حتى يمكنك الخطو في كل سطر من البرنامج أو (خطوة للمؤشر)، انقر فوق Step Out  في قائمة Debug أو اضغط CTRL+ SHIFT +F8.

يمكنك التبديل بين هذه الأنواع من الخطو. ويتوقف نوع الخطو الذي تقوم به على جزء البرنامج الذي تريد تحليله.

استخدام أمر On Error لتصحيح أخطاء وقت التشغيل

يستخدم أمر **On Error** لمعالجة الأخطاء وتوضيح ما يمكن عمله إذا وقعت. فمثلا في حالة حدوث خطأ يمكن أن تخبر **Access** أن تذهب إلى مكان معين داخل البرنامج أو أن تتجاهل هذا الخطأ. ما لم تستخدم علي الأقل إجراء واحد لتصبح أخطاء وقت التشغيل، فان برنامجك أو تطبيقك سينتهي فجأة بمجرد حدوث خطأ.

يأخذ أمر **On Error** إحدى الصيغ التالية:

```
On Error Goto <Line label/Line number>
On Error Resume Next
On Error Goto  0
```

وفيما يلي نوضح معني كل صيغة من الصيغ الثلاث:

- *معني أمر*

```
On Error Goto <Line label/Line number>
```

إذا حدث خطأ أثناء التشغيل، فان التنفيذ ينتقل إلى السطر المذكور عنوانه أو رقمه في الجزء **Goto** من أمر **On Error** ، ويجب أن يكون السطر المطلوب الانتقال إليه داخل نفس الإجراء ، وإلا سيحدث خطأ في الترجمة ، وعادة يتضمن الأمر الذي يلي العنوان استدعاء إجراء يتولى عملية التصحيح ، ويصبح هو الإجراء النشط . كما سيتضح من المثال بعد قليل.

- *أما الصيغة*

```
Resume Next
```

فمعناها عندما يقع خطأ أثناء التشغيل ، يتم تجاهل الخطأ ويستمر البرنامج في تنفيذ التعليمات الصحيحة ، بعبارة أخري ينتقل التنفيذ إلى الجملة التي تلي الجملة التي وقع فيها الخطأ.

- *والصيغة*

GoTo 0

تقوم بتعطيل أي معالج أخطاء في الإجراء الحالي.

انظر المثال التالي:

```
On Error GoTo ErrHandler
…
[RepeatCode:
(Code using Errproc to handle errors)]
…
GoTo SkipHandler
ErrHandler:
Call Errorproc
[GoTo RepeatCode]
SkipHandler:
…
(Additional Code)
```

وعن هذا المثال نوضح مايلي:

- يتسبب الأمر **On Error GoTo** في انتقال تنفيذ البرنامج إلى العنوان **ErrHandler** الذي يستدعى الإجراء **Errorproc** للتنفيذ والذى يقوم بمعالجة الخطأ الذي يحدث. وعادة توجد تعليمات معالجة الخطأ في نهاية الإجراء.
- إذا اشتمل البرنامج على أكثر من إجراء لمعالجة الخطأ ، أو إذا وضعت إجراء معالجة الخطأ في وسط مجموعة من التعليمات داخل البرنامج ، يجب أن تتجاهله إذا كانت التعليمات التي تسبقه خالية من الأخطاء.
- استخدمنا أمر **GoTo SkipHandler** الذي يتسبب في تجاهل التعليمات التي تلي العنوان **ErrHandler** .
- لكي يتكرر الأمر الذي سبب الخطأ بعد الانتهاء من تنفيذ الإجراء **Errorproc** ، أضفنا العنوان **RepeatCode** في بداية التعليمات المكررة، ليتم الانتقال إلى التعليمات التي تلي العنوان **ErrHandler** .