الفصل الثالث عشر

أنواع البيانات والثوابت والمتغيرات

نتعرف في هذا الفصل والفصول التالية على كيفية كتابة التعليمات والتركيبات الأساسية للغة **Visual Basic**، حيث يتناول هذا الفصل حافظات البيانات (**Data Holders**) وهي المتغيرات والثوابت، وكيفية معالجة هذه البيانات باستخدام عبارة التخصيص. بالانتهاء من هذا الفصل ستكتسب المعارف وتتدرب على المهارات التي تجعلك قادرا على:

- قواعد كتابة التعليمات
- كتابة عنوان للتعليمات
- كتابة التعليقات
- فهم أنواع البيانات
- أنواع المتغيرات وكيفية الإعلان عنها
- الإعلان عن الثوابت وبيان كيفية استخدامها

خلفية ضرورية

قبل أن نتعرض لأنواع البيانات ولشرح المتغيرات والثوابت التي هي موضوع هذا الفصل سنتعرض لخلفية ضرورية لابد من فهمها لكي تفهم البرمجة باستخدام لغة **Visual Basic**. هذه المقدمة تهم المبتدئين ومن يستخدمون لغة **Visual Basic** لأول مرة.

كيفية كتابة التعليمات

عند كتابة البرامج يتم إدخال كل تعليمه في سطر وتعد خطوة من خطوات التطبيق، ومع ذلك يسمح **Visual Basic** بإدخال أكثر من تعليمه في نفس السطر. وفيما يلى نوضح كيفية كتابة التعليمات بكلٍ من الطريقتين.

لاحظ أن جميع الأمثلة التي سنشرحها في هذا الفصل يجب أن تكتب داخل إجراء من خلال نافذة الكود.

الطريقة الأولى: كتابة كل تعليمة فى سطر مستقل كما يلى:

obeikandl.com

```
Dim X As Integer = 99
Dim str As String
Dim MyName As String
```

لاحظ في هذه الطريقة أن كل سطر يحتوى على تعليمة واحدة فقط، حيث يحتوى السطر الأول على متغير عددي هو X وذلك باستخدام كلمة Dim وهى اختصار Dimension وتستخدم للإعلان عن المتغيرات. وتستخدم العلامة "=" لتخصيص القيمة ٩٩ إلى المتغير X . ويحتوى السطر الثاني والثالث على متغيرات أخرى ويتم التعامل معها بنفس الأسلوب.(سوف نتكلم عن كيفية التعامل مع هذه المتغيرات لاحقا في هذا الفصل).

الطريقة الثانية: كتابة التعليمات الثلاثة في سطر واحد ويتم الفصل بينهم باستخدام العلامة " : " كما يلى:

```
Value1 = 0 : Value2 = 10 : Value3 = -6
```

عنوان السطر (Label)

عنوان السطر (Label) من الطرق التى يستخدمها Visual Basic للإشارة إلى التعليمة التى سيتم تنفيذها فى التطبيق .وتستخدم هذه الطريقة غالباً فى حالة الرغبة فى الانتقال إلى سطر خارج التسلسل الطبيعى دون شرط.

ويتم كتابة العنوان فى أول السطر قبل التعليمات المراد تنفيذها ويشتمل على مجموعة من الأحرف والأرقام لا تقل عن واحد ولا تزيد عن ٤٠ حرف.ويتم كتابة الحرف (:) بعنوان السطر كما فى الأمثلة التالية :

```
A:
btnExit:
FrmEdit:
```

التعليقات (Comments)

التعليقات هى الملاحظات التى تكتبها داخل التطبيق دون أن يكون لها أى تأثير عند تنفيذ التطبيق، بعبارة أخرى لا يقوم Visual Basic بتنفيذها ولا بمراجعتها. ويتميز المبرمج الماهر بالبراعة فى استخدام التعليقات فى شرح خطوات التطبيق بصورة جيدة تساعده فى تذكر التطبيق والرجوع إليه للتعديل أو التطوير فى أقل زمن ممكن.

كتابة التعليقات

هناك طريقتان لاستخدام التعليقات فى برنامج **Visual Basic**

الطريقة الأولى: استخدام الأمر **REM** وهو اختصار للكلمة **Remark**.

لاستخدام هذه الطريقة اكتب الأمر **REM** أولاً ثم اكتب التعليق بعد ذلك، سيعتبر **Visual Basic** كل ما يرد بعد كلمة **REM** تعليقاً. وتصلح هذه الطريقة فى حالة إضافة سطر كامل كتعليق داخل التطبيق.

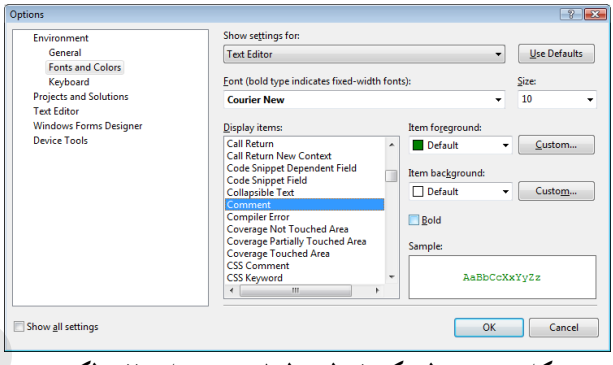
الطريقة الثانية: استخدام الحرف (`) قبل التعليق، وهذه الطريقة تصلح فى حالة الرغبة فى إضافة التعليق على نفس السطر.

مثال: فى المثال التالى يخصص السطر الأول كله كتعليق داخل التطبيق ولا يقوم **Visual Basic** بتنفيذه ولا يؤثر على نتيجة التطبيق، بينما يشتمل السطر الثانى على تعليق بالإضافة إلى تعليمات أخرى.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    REMينفذ هذا الإجراء فى بداية تحميل النموذج
    MessageBox.Show("one") : MessageBox.Show ("two") ' يحتوى
هذا السطر على أمرين
End Sub
```

لاحظ ظهور التعليق فور كتابتك له داخل التطبيق وانتقالك إلى أى سطر بلون مختلف غالباً هو اللون الأخضر إلا إذا قمت بتغييره بنفسك من داخل المربع الحوارى **Options** وذلك كما يلى:

١. افتح قائمة **Tools** من شريط القوائم ثم اختر **Options**، يظهر المربع الحوارى **Options** (انظر شكل ١٣-١).



شكل ١٣–١ التحكم فى لون التعليقات داخل نافذة الكود.

٢. انقر المجلد **Environment** من العمود الأيسر بالمربع الحوارى ثم اختر **Fonts and Colors** من القائمة المنسدلة.

٣. اختر **Comment** من مربع السرد **Display Items** بالجزء الأيمن من المربع الحوارى.

٤. قم بتغيير لون التعليق من القائمة المنسدلة **Item Foreground** وكذلك خلفيته من القائمة المنسدلة **Item Background**.

٥. يمكنك أيضاً تغيير الخط المستخدم من القائمة المنسدلة **Font** وكذلك حجم الخط من القائمة المنسدلة **Size**.

٦. انقر زر **Ok** لإغلاق المربع الحوارى.

المتغيرات *Variables*

المتغير (**Variable**) عبارة عن مكان يتم حجزه فى ذاكرة الحاسب ويخصص له اسم ويحمل قيمة قد تتغير أثناء تنفيذ التطبيق. فمثلاً إذا أردت أن تسأل عن اسم العميل الذى سيدخله المستخدم، فان اسم العميل قيمة متغيرة، لأنك لا تعرف من هو هذا العميل الذى سيقع عليه اختيار المستخدم. فى هذه الحالة تستخدم متغير لتضع فيه اسم العميل. انظر المثال التالي:

```
Dim HisName As String
HisName = InputBox ("اكتب اسم العميل")
```

فى هذا المثال سيعرض **Visual Basic** على المستخدم مربع حوار نتيجة لتنفيذ أمر **InputBox** يطالبه فيه بكتابة اسم العميل ويقوم بحفظ اسم العميل الذى يدخله المستخدم فى المتغير **HisName**، ويبقى المتغير **HisName** يحمل هذا الاسم حتى يقوم المستخدم بتغييره، ويتم تغيير القيمة التى يحملها المتغير بوضع قيمة أخرى داخله فيقوم **Visual Basic** باستبدال القيمة القديمة بالقيمة الجديدة وهكذا.

أيضاً من أمثلة استخدام المتغير: لو أن لديك برنامج يقوم بعد الموظفين في قاعدة بيانات مثلا فإنك تحتاج إلى متغير لحفظ هذا العدد الذى يتغير أولاً بأول ليعكس العدد الفعلي للموظفين. ومن طبيعة البيانات، نعرف أن هذا المتغير ينبغي أن يكون عدد صحيح وليس عدد كسري وأنه ينبغي أن يصلح لحفظ قيم كبيرة تفوق العدد المتوقع للموظفين ، هذه هي المعلومات المطلوبة للبدء في كتابة الكود والتي يحتاجها مترجم اللغة حين إنشاؤه للملف التنفيذي، بعد ذلك وعند تشغيل التطبيق لا يبقى إلا مكان في الذاكرة يتم ملؤه بالقيم المحددة في التطبيق.

من هذا نفهم أن للمتغير ثلاثة خصال تحدده وهي:

- اسم المتغير (والذي يشير إلى موقعه في الذاكرة)
- نمط المتغير، ويحدد تعامل المترجم معه والمساحة التخزينية المحجوزة له
- البيانات الفعلية المخزنة في المتغير

وكما يدل الاسم "متغير"، يمكن تغيير قيمة المتغير في أي وقت أثناء تشغيل التطبيق ، ويمكننا التفاعل ثنائي الاتجاه معه، أي استرجاع القيم المخزنة أو تخزين قيم جديدة، وهذا يحتاج كما سبق إلى أن يكون للمتغير اسم مميز ليمكننا التعامل معه.

لعلك لاحظت من قبل عبارة كالآتية:

```
Dim i As Integer
```

هذه العبارة تستخدم للإعلان **Declaration** عن متغير يسمى **i** ومن النمط**Integer** أى يحمل أرقاماً صحيحة. ومن وجهة نظر مترجم اللغة فإنه يفهم منها ما يلى:

- المبرمج سيستخدم متغيراً يسمى i وعليه سيقبل المترجم وجود هذا الرمز وسط الكود وسيقبل تغييره أثناء التشغيل ولكن بقيمة من نفس النمط.
- المتغير من النمط integer (أى يحتوى على قيمٍ صحيحة) وتخزين إحدى القيم فى هذا المتغير تحتاج إلى 4 bytes، لذا يتم حجز هذه المساحة وعند تشغيل التطبيق يقوم بإخبار نظام التشغيل باحتياجاته من الذاكرة ليمده بها.
- يسمح المترجم فقط بدخول هذا المتغير في عبارات Expressions تحتوي على نمط متوافق مع نمط هذا المتغير (مثلا يجب أن تكون المتغيرات الأخرى رقمية وليست حرفية).
- من الأشياء الأخرى الأقل وضوحا والتي يفهمها المترجم من وجود العبارة Dim أن المتغير سيستخدم في الملف أو نموذج النافذة أو الإجراء الذي يتم الإعلان من خلاله ولن يسمح برؤية أو تغيير قيمة المتغير من خارجه. فإن أردت خلاف ذلك يمكنك استخدام Public بدلا من Dim كما سنوضح فيما بعد.
- أيضاً من الأشياء غير الواضحة والتي يفهمها المستخدم عندما يكون الإعلان في إجراء أو دالة أن هذا المتغير يتم حجز الذاكرة له فقط عند نداء الإجراء أو الدالة، وفيما عدا ذلك فليس له وجود. فضلاً عن ذلك كلما تم نداء الدالة يتم وضع قيمة "صفر" في المتغير أن كان رقميا.فإن أردت ألا يحدث ذلك يمكنك استخدام كلمة static كما سيأتي.

وكما ترى فإن العبارة البسيطة تحمل كثيراً من المعلومات المفيدة للمترجم ولك أيضاً، فبالرغم من كون الإعلان عن المتغيرات غير إجباري في Visual Basic مثل لغات Visual C++ أو C# إلا أنه غير إفادته يخلصك من كثير من المشاكل كما سيتضح فيما بعد.

نبدأ بتسمية المتغيرات، ويعطيك Visual Basic حرية كبيرة في تسمية المتغيرات آخذا في الاعتبار القيود الآتية:

- يجب أن يبدأ اسم المتغير بحرف وليس برقم أو أي رمز آخر.
- يجب ألا يحتوي اسم المتغير على النقطة (.) أو المسافة space أو أي رمز خلاف الأحرف اللاتينية الكبيرة والصغيرة والأرقام والشرطة التحتية (_) المسماة underscore أي أن الأحرف العربية غير مسموح بها.

- يجب أن تكون الأسماء مميزة وليس فيها تماثل داخل نطاق المتغير.
- غير مسموح باستخدام كلمات Visual Basic المحجوزة مثل Dim و For وأمثالها (تسمى هذه الكلمات Reserved Words أو Keywords وعند كتابتها في نافذة الكود في Visual Basic ستجد أن لونها يتحول إلى الأزرق – يمكنك تغيير اللون– ويمكنك الاستدلال بذلك على أنك كتبت الكلمة بصورة صحيحة وأيضا عند اختيارك لاسم متغير هل هو كلمة محجوزة أم لا).

فيما عدا ذلك يمكنك التسمية بحرية ولكن راعى أن يكون الاسم ذو معنى وغير طويل ليسهل حفظه وتذكر دلالته خاصة في المشروعات الكبيرة.

أمثلة على أسماء متغيرات غير صحيحة:

اسم المتغير	سبب عدم صلاحيته
3rdQuarter	يبدأ برقم
مخزن_دمنهور	حروف غير لاتينية
Case	كلمة محجوزة في لغة Visual Basic
Number One	يحتوي مسافات
P.T.Value	يتضمن نقاط

كثير من المبرمجين يستخدم أسلوب يسمي بأسلوب التسمية المجري (Hungarian Naming Convention) نسبة إلى مبتكرها ذو الأصل المجري، وفيه يحتوي اسم المتغير على بادئة تبين نمط المتغير تتكون من حرفين أو ثلاثة ثم اسم المتغير مبدوءاً بحرف كبير. مثلاً استخدمنا في الفصول السابقة البادئة st والتي تدل على سلسلة بيانات أو متغير حرفيString وفيما يلي نذكر البادئات الأكثر استخداما:

نمط المتغير	البادئة	مثال
String	st	StName
Integer	in	InCount
Long integer	lg	LgPopulation
Single	sg	SgTemperature
Double	db	Dbdistance
Boolean	bl	BlValid

لا عليك الآن من معاني واستخدامات الأنماط المختلفة فسوف نشرحها بعد قليل، إنما يهمنا أن تتذكر هذه البادئات عند الحاجة إليها.

أنواع المتغيرات

السؤال الآن هو ماذا يمكننا أن نخزن في المتغير ؟ تقريبا يمكننا تخزين أي نوع من البيانات ، يمكن للمتغير أن يحمل رقماً صحيحاً أو كسرياً، أو قيمة نصية في متغير حرفي، أو أحد أفراد كائن مثل نافذة أو أداة تحكم. إلا أن المتغيرات غالباً ما تستخدم للبيانات البسيطة لذا سنركز عليها في هذا الفصل.

بالنسبة للمبتدئين، يمكن أن يتناسوا تماما موضوع النمط هذا، ويستخدمون متغيراً لتخزين أي نوع من البيانات فيه. هذه الطريقة سيئة ومليئة بالمشاكل، فضلا عن أنك، بعد قراءتك للفصول السابقة لا يجدر بك أن تسلك هذا السلوك في برامجك. ينبغي أن تضع كل قيمة في مكانها الصحيح، أي في المتغير المناسب، بأن تحدد النمط المناسب للمتغير. عندما تخبرVisual Basic بنمط المتغير فإنه يمنحك تطبيقات أسرع وأقل عرضة للأخطاء. هناك العديد من الأنماط في Visual Basic وهي في الحقيقة لغة غنية من حيث تعدد الأنماط.

يوضح جدول ١٣-١ التالى كافة الأنماط مع استخدامها وحجم المتغير في الذاكرة ونطاق المتغير أو مداه من القيم.

جدول ١٣-١ الأنماط المختلفة للبيانات التى يتعامل معها Visual Basic

نوع المتغير	معناه	حجمه	مثال
Integer	عدد صحيح صغير نسبياً	4 bytes	0, 1, 000, -1, -2000
Long	عدد صحيح طويل نسبياً	8 bytes	0, 1, 33000, -2, -33000 أى عدد يتكون من ١٩رقم
Single	عدد حقيقي يحتوى على علامة عشرية	4 bytes	0.000123
Short	أرقام صحيحة	2 bytes	32,123
Double	عدد حقيقى يحتوى على علامة عشرية عائمة (كبير نسبياً)	8 bytes	1.23E-10

نوع المتغير	معناه	حجمه	مثال
Decimal	عدد ذو علامة عشرية يبلغ حتى ١٥ رقم صحيح و٤ أرقام عشرية	16 bytes	19.95D
String	سلسلة من الحروف ثابتة الطول	يعتمد على الطول	"AB", "15", "AAA111"
Byte	عدد صحيح	1 Byte	0-255
Boolean	قيمة منطقية	2 Bytes	True-False
Date	التاريخ والوقت	8 Bytes	12/12/2002 02:00 PM
Object	يحتوى على مرجع لكائن من أى نوع	4 Bytes	–
Char	يحتوى على حرف واحد فقط	2 Bytes	A,B,C
SByte	يحتوى على قيم سالبة أو موجبة فى حدود البايت	1Byte	-128 - 127
UInteger	يحتوى على قيم صحيحة موجبة فقط	4 Bytes	0- 4,294,967,295
ULong	يحتوى على رقم من النوع Long فى المدى الموجب فقط	8 Bytes	0- 18,446,744,073,709,551,615
UShort	يحتوى على رقم من النوع Short ولكن فى المدى الموجب فقط	2 Bytes	0- 65,535

obeikandl.com

تم حذف نوع البيانات **Currency** الموجود فى الإصدارات السابقة من**Visual Basic**، وتوصى ملفات المساعدة باستخدام نوع البيانات الجديد **Decimal** عوضاً عنه. كما تم حذف نوع البيانات **Variant** وذلك لأن جميع أنواع البيانات مشتقة أساساً من التصنيف **Object**، لذا يتم استخدام هذا التصنيف كنوع عام للبيانات عوضاً عن النوع **Variant**.

كيف تختار نمط المتغير

نمط المتغير هو نوع بيانات المتغير. ومن الأشياء الهامة التى تساعدك في كتابة إجراءات سليمة كيفة اختيار نمط للمتغير الذي ستستخدمه.

هناك بعض الحالات التى لا يوجد فيها اختيار، مثل تخزين قيم مالية، في برنامج للمبيعات على سبيل المثال (غير قابلة للتقريب للمحافظة على دقة القيمة المالية)، هنا نختار **Decimal**، حيث لو استخدمنا **Single** على سبيل المثال فإن جمع الأرقام الممثلة للقيم المالية قد يؤدي إلى أخطاء عند تقريبها. أيضا عند الرغبة في تخزين قيم تواريخ، كتطبيق به مفكرة مثلاً فإن النمط المناسب الوحيد هو **Date**. كذلك لو احتجنا متغير به قيمة منطقية (نعم/لا) نحتفظ بها طول التطبيق لفعل معين، مثلا في برنامج للطباعة قد نسأل المستخدم "هل ترغب في الطباعة بالطريقة التالية؟" ثم نحتفظ بإجابته لتوجيه عملية الطباعة بالطريقة المطلوبة. هنا نستخدم النمط **Boolean** الذى يحتوى على قيمة منطقية، (الاسم **Boolean** بالمناسبة نسبة إلى العالم بوليان الذي وضع الجبر المنطقي وسمي باسمه **Boolean Algebra**) أخيرا لو احتجنا لحفظ نص أو سلسلة حرفية كما يسمى فإننا نستخدم النمط**string** .

عندما نأتي بعد ذلك للقيم الرقمية، نسأل أنفسنا السؤال التالي: هل القيمة المراد وضعها في المتغير بها كسور؟ إن كانت الإجابة بنعم فإننا نستخدم **Single** أو **Double** ، الأول عندما لا نكون في حاجة إلى دقة كبيرة بل إلى سرعة في التطبيق. و الثاني للأعداد الضخمة و الدقة العالية (الاسم **Double** يقصد به **Double Precision** أي دقة مضاعفة أي من النوع **Single** ذو الدقة الأحادية.)

إن كنا في غير حاجة للكسور فإننا نستخدم **byte** أو **Integer** أو **Long** وهما مرتبين بدءاً بالأقل في مدى الأرقام والمساحة التخزينية حتى أكبرها. الذى يحدد استخدام أيٍ منها هو مدى القيمة المراد تخزينها. مثلاً لو أردنا تخزين عدد يعبر عن عدد مشغلات الأقراص في جهاز مثلا، فلن تتجاوز ٢٥٦ بأي حال من الأحوال. و عليه من الأوفر استخدام النمط **Byte** .

إن كنت ستحتاج أعداد (موجبة أو سالبة) لا تتجاوز النصف مليون (مثل عدد الطلاب في مدرسة مثلا) فإن النمط **Integer** مناسب تماما. إذا زاد الرقم عن ذلك (مثل عدد سكان دولة) حيث نحتاج رقم بالملايين نستخدم النمط (**long**) يمكنه حمل رقم يصل إلى مليارين). الأرقام الصحيحة الأكبر من ٢ مليار لا يدعمها **Visual Basic** ونضطر هنا لاستخدام الأنماط الحقيقية **Single** و **Double** لتمثيلها مع فقدان بعض الدقة في تمثيل الرقم (من الممكن أن يقل أو يزيد الرقم مئات أو عشرات عند تخزينه ثم استرجاعه نتيجة الطريقة المستخدمة في تخزين الأنماط الكسرية)،كما في المثال التالي:

```
Dim x As Single
x = 2000000001
Text1.Text = x
```

الإعلان عن المتغيرات

الإعلان عن المتغير عبارة عن أمر يخبر **Visual Basic** باسم المتغير ونوعه، ليتمكن **Visual Basic** من حجز المساحة اللازمة من ذاكرة الحاسب لهذا المتغير وتهيئته. كما سبق و ذكرنا تستخدم العبارة **Dim** للإعلان عن المتغيرات، إن لم تعلن عن المتغير وذكرت اسمه مباشرة في الكود فإن **Visual Basic** سيفترض أنه من النوع **Object** ويمكنه أن يحتوي على أي نوع من البيانات. هذا الكلام ما لم تطلب الإعلان الإجباري من خلال الخاصية **Option Explicit** بالتبويب **Compile** بمربع الخصائص الخاص بالمشروع (راجع الفصل السابق).

ويجب أن تعلم أن النمط الافتراضي **Object** ذو مساوئ عديدة منها استنفاذ مساحة كبيرة من حجم الذاكرة، والبطء في التعامل معه، و أيضا يحد من استخدام المتغير مع

بعض الدوال (الوظائف) التي لا تتعامل مع هذا النمط. لذا فمن المفضل دائما الإعلان عن المتغيرات بل وجعل المترجم يجبرك على ذلك.

ويتم الإعلان عن أى متغير باستخدام عبارة **Dim** أو أحد مكافئاتها مثل **Private** و **Public**و **Static** وذكر نمط المتغير صراحة بعد اسمه ، والصورة العامة للإعلان هي:

```
Dim varname1 As type1, varname2 As type2,...
```

حيث يتم استبدال **varname** باسم المتغير و**Type** بنوع بياناته. كما ترى يمكن الإعلان عن عدة متغيرات في نفس السطر، كما يجب ملاحظة الكلمة **As** والتي لابد من ذكرها مع النمط.

من أمثلة الإعلان الصريح عن المتغيرات

```
Private stName As String
Public iNewProp As Integer
Dim x  → Object  يفترض أن نمطه
```

في المثال الأول تم الإعلان عن متغير من نوع سلسلة بيانات **String**، وفي المثال الثاني متغير من النوع الصحيح **Integer**، وفي المثال الثالث لم نحدد نوع أو نمط المتغير، وإنما اكتفينا بذكر اسمه فقط، وفي هذه الحالة سيفترض **Visual Basic** أن نمط المتغير من النوع **Object** أى متنوع ويختلف باختلاف محتوياته.

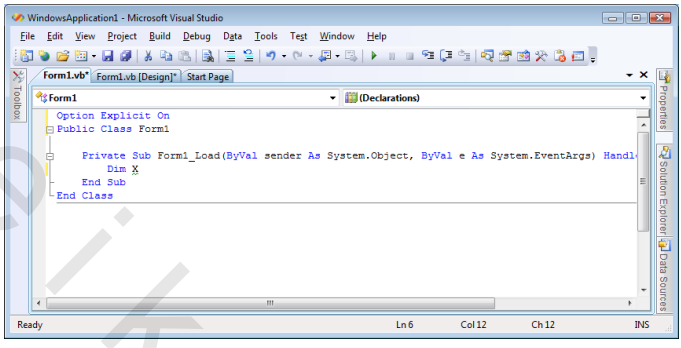
الالتزام بالإعلان عن المتغير

رغم أنه بالإمكان استخدام المتغير بدون الإعلان عنه، إلا أن هذا الأمر ينطوى على مخاطر الوقوع فى أخطاء خفية، فإذا أردت أن تتجنب ذلك فعليك بإلزام **Visual Basic** بعدم قبول أى متغير بدون الإعلان عنه مسبقاً ويتم ذلك بإحدى طريقتين:

الطريقة الأولى:

أن تضع الأمر **Option Explicit** فى قسم الإعلانات (**Declaration Section**) فى النموذج أو الملف الذى تريد استخدامه فيه (انظر شكل ١٣-٢). فى هذه الحالة سيعرض عليك **Visual Basic** رسالة خطأ عند ورود أى متغيرات لم يسبق الإعلان عنها، وستتضمن رسالة الخطأ اسم المتغير. ينحصر تأثير أمر **Option Explicit** على

النموذج أو الملف الذى ورد به فقط، ولذلك يجب وضعه فى قسم الإعلانات فى النموذج أو الملف الذى تريد تنفيذه عليه. إذا أردت أن يمتد تأثير الأمر ليشمل المشروع كله، استخدم الطريقة الثانية.

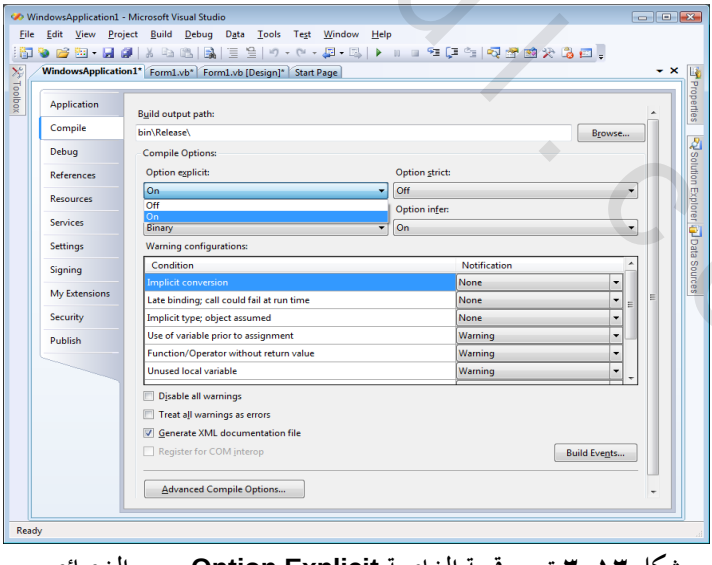


شكل ١٣–٢ عبارة **Option Explicit** تظهر في قسم الإعلان العام للنموذج.

الطريقة الثانية:

باستخدام الخيار **Option Explicit**. لأداء ذلك، تابع معنا الخطوات الآتية:

١. انقر المشروع بزر الفأرة الأيمن داخل نافذة المستكشف ثم اختر **Properties** من القائمة الموضعية، يظهر مربع خصائص المشروع (انظر شكل ١٣–٣).



شكل ١٣–٣ تعيين قيمة الخاصية **Option Explicit** بمربع الخصائص.

٢. نشط التبويب Compile من الجانب الأيسر من المربع.
٣. تحتوى القائمة المنسدلة Option Explicit على قيمتين هما On وOff. اختر القيمة On إذا أردت تعريف المتغيرات صراحةً أو القيمة Off إذا أردت أن يقوم المترجم باختيار النوع المناسب للمتغير نيابةً عنك.

بعد ذلك لو أخطأت في كتابة اسم أحد المتغيرات أثناء استخدامه، أي ظهر اسمه في التطبيق دون أن تعلن عنه صراحة وكانت قيمة الخاصية Option Explicit هى On فستظهر لك رسالة الخطأ الآتية:

```
Name x is not declared
```

حيث x عبارة عن اسم المتغير المستخدم.

مثلا لو أعلنت عن المتغير stName كالتالي:

```
Dim stName as string
```

ثم أخطأت في كتابة اسمه وكتبت:

```
stNsme= "Ahmed"
```

فستظهر رسالة الخطأ وهذا يحمينا كما هو واضح من الأخطاء التي قد تنتج عن ذلك، لاحظ أنك لو لم تستخدم Option Explicit فإن استخدام أي اسم غير معلن عنه لن يسبب خطأ و إنما ستستخدم القيم الافتراضية للمتغيرات (صفر للأنماط الرقمية ، و "" للمتغيرات الحرفية و هكذا) مما قد يسبب أخطاءً كبيرة في البرامج التي تنشئها.

لو قمت بكتابة بعض أحرف اسم المتغير في الإعلان عنه بحروف كبيرة ثم كتبتها صغيرة أثناء استخدامه في التطبيق فسيقوم المصحح الفوري لمحرر الكود بكتابتها بحروف كبيرة بمجرد أن تغادر السطر و هذا يدلك على أنك كتبت الاسم صحيحا.

مدى استخدام المتغير وعمره Lifetime and Scope of Variable

يقصد بمدى استخدام المتغيرScope of Variable الإجراءات والنماذج والملفات التى ستتأثر به، أى الأماكن التى يمكن أن يستخدم فيها هذا المتغير داخل التطبيق. أما عمر المتغير Lifetime of variable فيقصد به المدة التى سيبقى المتغير خلالها محتفظا

obeikandl.com

بقيمته الحالية داخل الذاكرة دون أن يفقدها، وتنقسم المتغيرات من حيث مدة بقائها فى الذاكرة ومداها إلى متغيرات عامة، ومتغيرات على مستوى النموذج أو الملف (متغيرات محلية)، ومتغيرات على مستوى الإجراء وأخيراً متغيرات على مستوى كتلة من الكود. وفيما يلى نوضح كل نوع من هذه الأنواع الأربعة والأمر الذى يستخدم للإعلان عنه.

المتغيرات العامة

هى المتغيرات التى يمكنك استخدامها من أي مكان داخل المشروع، وتبقى فى ذاكرة الحاسب طوال فترة عمل المشروع، فإذا انتهى التطبيق تحذف من الذاكرة.

تستخدم عبارة **Public** أو عبارة **Friend** للإعلان عن المتغيرات العامة. فى المثال التالى يتم الإعلان عن متغير عام لكى تستخدمه جميع ملفات ونماذج التطبيق من نوع **Integer** واسمه **ABC**:

```
Public ABC AS integer
```

المتغيرات على مستوى الملف أو النموذج

يمكنك الإعلان عن متغير وتقييده بملف أو نموذج أو إجراء. فى هذه الحالة لن تستطع استخدام المتغير إلا من خلال النموذج أو الملف الذى أعلنت عنه فيه، ولن تستطع استخدامه خارجها. فترة عمل هذا النوع من المتغيرات هى فترة عمل النموذج أو الملف وليس التطبيق كله، أى أن الفرق بينها وبين المتغيرات العامة هو فى المدى الذى تستخدم فيه فقط.

للإعلان عن متغير من هذا النوع استخدم عبارة **Dim** أو **Private** داخل جزء الإعلان الخاص بالتصنيف أو النموذج **Module** الخاص بالملف.

فى المثال التالى يتم الإعلان عن متغير من نوع **String** واسمه **CompanyName** لكى يستخدم فقط مع الملف أو النموذج الذى يوجد به:

```
Dim CompanyName As String
```

متغيرات على مستوى الإجراء

يقتصر مدى هذه المتغيرات على الإجراء الموجودة به فقط، ولا يمكن استخدامها فى أى مكان غيره، وهى بهذا تعتبر أقل المتغيرات مدىً. أما من حيث عمرها فهى تبقى موجودة بالذاكرة حتى ينتهى الإجراء الذى أعلن فيه عنها وبهذا يتضح أن الفرق بين هذه المتغيرات

والمتغيرات العامة أو المتغيرات على مستوى الملف أو النموذج فى مداها فقط، حيث لا يتعدى مداها الإجراء الذى أعلن عنها فيه. يستخدم لهذا الغرض عبارة **Dim**. أما إذا أردت الاحتفاظ بقيمة المتغير بعد الخروج من الإجراء، فيمكنك استخدام كلمة **Static** ويتضح ذلك من المثالين التاليين. فى المثال الأول يظل المتغير **S1** محتفظا بقيمته فترة تنفيذ الإجراء **btnTest_Click** وبمجرد الخروج من الإجراء ستكون قيمته تساوى صفر بينما فى حالة استخدام المثال الثانى فإن المتغير سيحتفظ بقيمته بعد تنفيذ نفس الإجراء السابق.

المثال الأول:

```
Private Sub btnTest_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnTest.Click
Dim S1 As Integer
S1 = 5
End Sub
```

المثال الثانى:

```
Private Sub btnTest_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnTest.Click
Static S1 As Integer
S1 = 5
End Sub
```

يوفر عليك تحديد مدى المتغيرات وعمرها استهلاك مساحة من الذاكرة بدون داع. فمثلاً إذا كنت تريد استخدام متغير فى أكثر من نموذج، فيجب أن تعلن عنه كمتغير عام بعبارة **Public**، وإذا كنت تحتاج للمتغير فى نموذج أو ملف واحد فقط، استخدم الأمر **Dim** أو **Private** داخل النموذج أو الملف. أما إذا كنت تحتاج للمتغير مؤقتاً فى هذا الإجراء فقط، استخدم أمر **Dim** داخل الإجراء ليبقى مداه داخل الإجراء فقط.

الثوابت Constants

الثوابت كما هو واضح من اسمها عبارة عن اسم يحمل قيمة ثابتة لا تتغير أثناء تنفيذ التطبيق، وهى عكس المتغيرات التى تتغير قيمتها من حينٍ إلى آخر تبعاً للمدخلات. ومع

ذلك فهى تتشابه مع المتغيرات فى أمرين هما اسم الثابت ومداه كما سيتضح بعد قليل.
ولتوضيح فكرة الثابت نضرب المثال التالى:
إذا كان عملك يتطلب مجموعة من العمليات الحسابية ترتبط بوحدة ثابتة مثل وحدة قياس المتر وهو يساوى مائة سنتيمتر فيمكن الإعلان عن ذلك بالأمر التالى:

```
Const Meter As Integer = 100
```

وهذا يفيدك عندما تكون جميع حساباتك بالنسبة للوحدة سنتيمتر فبدلاً من حساب قيمة المتر وكتابة الرقم (١٠٠) فى كل مرة، يتم كتابه الثابت **Meter** فى جميع التعليمات المطلوبة داخل التطبيق وهى فائدة كبيرة تجعل برنامجك سهلا وبسيطا.
فائدة أخرى يمكن الحصول عليها من استخدام الثوابت، فمثلا فى حالة تعديل كل حساباتك لتصبح منسوبة لوحدة المليمتر بدلا من السنتيمتر (والمعروف أن المتر = ١٠٠٠ مليمتر) فبدلاً من إجراء هذا التعديل فى جميع إجراءات برنامجك (وهو كتابة الرقم ١٠٠٠ بدلاً من الرقم ١٠٠).
يكفى أن تعدل الرقم ١٠٠ ليصبح ١٠٠٠ فى نفس الأمر كالآتى:

```
Const Meter As Integer = 1000
```

وبذلك تتم عملية التعديل مرة واحدة فقط لتعطى النتيجة المطلوبة.
من الاستخدامات الهامة الأخرى توفير كثير من خطوات التغيير في التطبيق، مثلاً يمكن تخزين رقم إصدار التطبيق **Version number** ، أو الاسم الذي ستطلقه على التطبيق في ثابت في أحد الملفات ثم تقوم باستخدامه طيلة التطبيق في أماكن متفرقة ، ثم إذا أردت تغيير الاسم أو رقم الإصدار، فليس عليك سوى تغيير الثابت في مكان واحد فقط.
مثال لذلك:

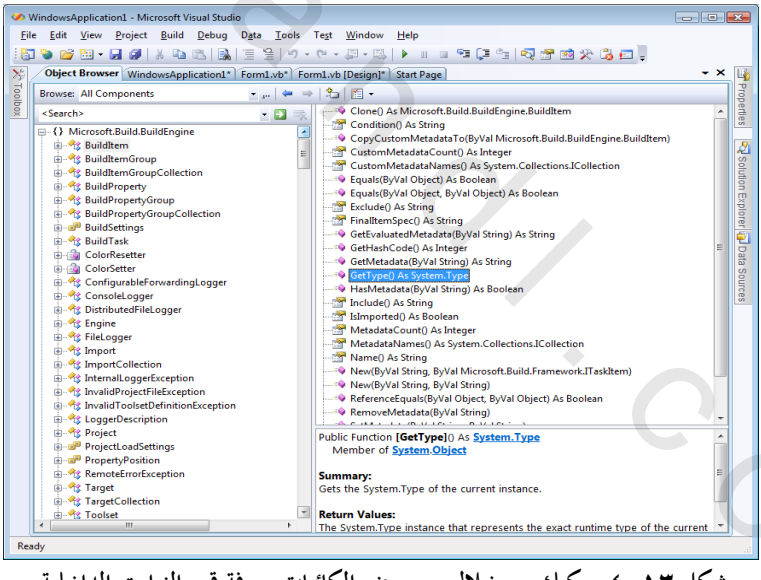
```
Public Const ProgName= "CompuScience Demo Prog"
Public Const ProgVers= "4.5"
```

الكود التالي مثلا يقوم بالتحويل من سنتيمتر إلى بوصة باستخدام أحد الثوابت:

```
Const CM_PER_INCH = 2.54
Length_cm = (Length_Inch) * CM_PER_INCH
```

ثوابت Visual Basic

يمدك Visual Basic بكمية كبيرة من الثوابت المعرفة التي يمكنك استخدامها مباشرة مع برنامجك دون تعريف حيث أنها معرفة داخليا ولذلك تسمى الثوابت الداخلية intrinsic constants مثل MessageBox.Buttons.Ok والتى تسهل كثيراً من استخدام الوظائف والخصائص داخل الكود. عموما عند قراءتك لملف التعليمات help عن دالة ستقرأ عن الثوابت التي تستخدم معها وكيفية استخدامها. إذا أردت معرفة قيمة ثابت معين يمكنك استخدام مستعرض الكائنات Object Browser ، ويمكنك فتحه بفتح قائمة View من شريط القوائم ثم اختيار Object Browser من القائمة المنسدلة أو بضغط مفتاح F2 وعند اختيارك لثابت داخل النافذة تظهر قيمته أسفل النافذة (انظر شكل ١٣-٤).



شكل ١٣-٤ يمكنك من خلال مستعرض الكائنات معرفة قيم الثوابت الداخلية.

تعريف ثوابت جديدة

يتم تعريف ثابت في أي مكان من الكود بالعبارة:

```
Const CONSTANT_NAME [As constantType] = value
```

والعبارة تشبه عبارة الإعلان عن متغير في أن الاسم (إجباريا) و النمط (اختياريا) يتم تحديدهما إلا أن هناك كلمة Const بدلا من Dim وكذلك وجود علامة التخصيص = يليها القيمة. و ما أن يتم التعريف بهذه العبارة لا يمكن بحال من الأحوال تغيير قيمة المتغير بأي طريقة. حيث سيرفض المترجم ذلك. ومما يجب ملاحظته أن الثوابت الحرفية Stringيجب وضعها بين علامتي اقتباس " " مثل "Hassan" ، بينما ثوابت التاريخ (من النمطDate) يتم وضعها بين علامتي # # ، مثل # ١٩٧٤/٨/١ # .

```
NAME = “Ahmed”
BIRTH_DATE = # 1/8/1974 #
```

استخدام الحروف الكبيرة كأسماء للثوابت كما يظهر في السطور السابقة عادة شائعة بين مبرمجي Visual Basic. ويخضع اسم الثابت لنفس الشروط التى شرحناها عند اختيار اسم المتغير.

مدى الثوابت

تتبع الثوابت نفس القواعد التى تحدد مدى المتغيرات، حيث يحدد مدى الثابت بالمكان الذى تعلن فيه عن هذا الثابت. وتوضيح ذلك كما يلى:

- إذا أردت أن يكون الثابت عاما أى يمكن استخدامه من أى مكان فى التطبيق فيجب أن تعلن عنه فى منطقة إعلان النموذج بشرط أن يسبق الإعلان عنه كلمة Public هكذا:

```
Public Const Comp_Name= “CompuScience”
```

- لكى تستخدم الثابت فى ملف أو نموذج واحد فقط، يجب أن تعلن عنه فى قسم الإعلانات فى هذا الملف أو النموذج بدون كلمة Public هكذا:

```
Const N0_EMP = 20
```

- لكى تستخدم الثابت مؤقتا داخل إجراء معين، قم بالإعلان عن الثابت داخل هذا الإجراء بنفس الطريقة السابقة.