



سنشرح في هذا الفصل كيفية كتابة التعليمات والتركيبات الأساسية للغة **Access VBA**، وتشمل الثوابت والمتغيرات والمصفوفات، وكيفية تحديد الشروط واتخاذ القرارات واستخدام الدورات بانتهاء هذا الفصل ستتعرف على:

- ◆ قواعد كتابة البرنامج.
- ◆ كتابة التعليقات.
- ◆ أنواع المتغيرات وكيفية الإعلان عنها.
- ◆ المصفوفات.
- ◆ الإعلان عن الثوابت وبيان كيفية استخدامها.
- ◆ تحديد الشروط واتخاذ القرارات.
- ◆ التكرارات والدورات.

سنشرح في هذا الفصل أهم أوامر وتركيبات لغة **Access VBA** التي يحتاج إليها المبرمج في كتابة البرامج ، ولذلك فإننا نعتبر هذا الفصل ضروري جدا لمن يريد أن يتعلم هذه اللغة. ولهذا فإننا نرجو أن تولى هذا الفصل عناية خاصة، وأن تعتبره مرجعا ترجع إليه من حين لآخر كلما احتجت للتعرف على قواعد كتابة التعليمات وكيفية كتابتها. ولأن الأمثلة الواردة في هذا الفصل تتعلق بكتابة التعليمات التي يتكون منها البرنامج، فيجب كتابتها في نافذة البرمجة لخاصية معينة أو من خلال وحدة نمطية.

## كيفية كتابة التعليمات

عند كتابة البرامج يتم إدخال كل تعليمه في سطر وتعد خطوة من خطوات البرنامج، ومع ذلك يسمح **Access VBA** بإدخال أكثر من تعليمه في نفس السطر. وفيما يلي نوضح كيفية كتابة التعليمات بكل من الطريقتين

قد يطلق علي تعليمات **Access VBA** أوامر أو جمل أو عبارات وكلها بمعنى واحد.



**الطريقة الأولى :** كتابة كل تعليمة في سطر مستقل كما يلي:

**Value1% = -6**

**Value2% = 10**

**Value3% = 0**

لاحظ في هذه الطريقة أن كل سطر يحتوي على تعليمة واحدة فقط ، حيث يحتوي السطر الأول على متغير عددي هو **Value1** . يشير المعامل % على أن القيمة المخصصة لهذا المتغير سوف تكون قيمه عددية. كما يستخدم المعامل "=" لتخصيص القيمة **6-** إلى المتغير **Value1**. ويحتوي 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 بينها باستخدام المعامل " : " كما يلي:

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

### التعليقات (Comments)

التعليقات هي الملاحظات التي تكتبها داخل البرنامج دون أن يكون لها أى تأثير عند تنفيذ البرنامج، بعبارة أخرى لا يقوم Access VBA بتنفيذها ولا بمراجعتها. ويتميز المبرمج الماهر بالبراعة في استخدام التعليقات في شرح خطوات البرنامج بصورة جيدة تساعده في تذكر البرنامج والرجوع إليه للتعديل أو التطوير في أقل زمن ممكن.

### إدخال التعليقات

هناك طريقتان لاستخدام التعليقات في برنامج Access VBA

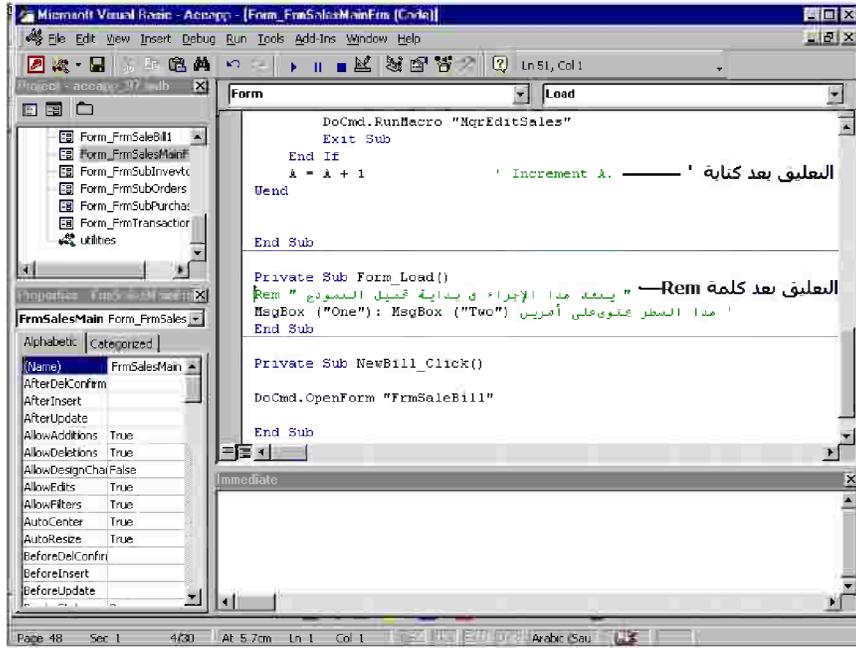
**الطريقة الأولى :** استخدام الأمر REM وهو اختصار للكلمة REMARK .

لاستخدام هذه الطريقة اكتب الأمر REM أولاً ثم اكتب التعليق بعد ذلك، سيعتبر Access VBA كل ما يرد بعد كلمة REM تعليقا. تصلح هذه الطريقة في حالة إضافة سطر كامل كتعليق داخل البرنامج.

**الطريقة الثانية :** استخدام الحرف (') قبل التعليق، وهذه الطريقة تصلح في حالة الرغبة في إضافة التعليق على نفس السطر.

### مثال

في المثال التالي يخصص السطر الأول كله كتعليق داخل البرنامج ولا يقوم Access VBA بتنفيذه ولا يؤثر على نتيجة البرنامج، بينما يشتمل السطر الثاني على تعليق بالإضافة إلى تعليمات أخرى، كما يظهر في شكل ١-٣.



شكل ٣-١ كتابة التعليقات داخل البرنامج

## المتغيرات Variables

المتغير (Variable) عبارة عن مكان يتم حجزه في ذاكرة الحاسب ويخصص له اسم ويحمل قيمة قد تتغير أثناء تنفيذ البرنامج. ، فمثلا 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. في هذه الحالة تستخدم متغير لتضع فيه اسم العميل. انظر المثال التالي:

**HisName\$=InputBox\$("اكتب اسم العميل")**

في هذا المثال سيعرض Access VBA على المستخدم مربع حوار نتيجة لتنفيذ أمر **InputBox\$** يطالبه فيه بكتابة اسم العميل ويقوم بحفظ اسم العميل الذى يدخله المستخدم في المتغير **HisName\$** ، ويبقى المتغير **HisName\$** يحمل هذا الاسم حتى يقوم المستخدم بتغييره ، ويتم تغيير القيمة التى يحملها المتغير بوضع قيمة أخرى داخله فيقوم

**Access VBA** باستبدال القيمة القديمة بالقيمة الجديدة.

ويجب أن يراعى عند اختيار اسم للمتغير الشروط التالية:

- يجب أن يبدأ اسم المتغير بحرف أبجدي.
- ألا يزيد عدد حروف اسم المتغير عن ٤٠ حرفاً.
- يجب ألا يتضمن كلمة من الكلمات المحجوزة ، وهي الكلمات التي تستخدم في الأوامر والعبارات التي يستخدمها **Access VBA** ، فمثلاً لا يسمح باستخدام كلمة **Print** كاسم للمتغير ، فإذا احتجت لتسمية متغير باسم مثلها يمكن أن تكتب الكلمة كجزء من اسم المتغير هكذا **PrintText** .

### أنواع المتغيرات

يوجد في لغة **Access VBA** عشرة أنواع للمتغيرات نوضحها في الجدول التالي:

نوع المتغير	معناه	حجمه	مداه	الرمز
Byte	حرف	1 byte	من 0 إلى 256	لا يوجد
Boolean	منطقي	1 byte	True أو False	لا يوجد
Integer	عدد صحيح صغير نسبياً	2 byte	من -32768 إلى 32767	%
Long	عدد صحيح كبير نسبياً	4 byte	من -2147483648 إلى 2147483674	&
Single	عدد حقيقي صغير نسبياً (يحتوى على علامة عشرية) عائمة Floating (Point)	4 byte	من -3.402823E38 إلى -1.401298E-45 (قيم سالبة) من 1.401298E-45 إلى 3.402823E38 (قيم موجبة)	!

نوع المتغير	معناه	حجمه	مداه	الرمز
Double	عدد حقيقي (يحتوى على علامة عشرية عائمة) كبير نسبياً	8 byte	رقم هائل	#
Currency	عدد حقيقي كبير نسبياً (يحتوى على علامة عشرية ثابتة)	8 byte	رقم هائل	@
String	سلسلة من الحروف ثابتة الطول		من صفر إلى ٢ بليون حرف تقريبا	\$
Date	التاريخ والوقت		-January 1, 100 December 31, 9999	لا يوجد
Variant	الوقت/التاريخ، أو عدد ذو علامة عشرية عائمة ، أو سلسلة حروف		التاريخ من ١ يناير ٠٠٠٠ إلى ٣١ ديسمبر ٩٩٩٩ . وفي الأعداد مثل Double وفي الحروف مثل String	لا يوجد

تختلف القيمة "15" عن القيمة 15 حيث أن القيمة 15 قيمة عددية (حسابية) يمكن إجراء أى عملية حسابية عليها مثل الجمع والطرح و... الخ. بينما تعامل القيمة "15" معاملة الحروف ولا يمكن التعامل معها على كقيمة عددية إلا بعد تحويلها.



### كيفية الإعلان عن المتغيرات

الإعلان عن المتغير عبارة عن أمر يجزى Access VBA باسم المتغير ونوعه كى

يمكن من حجز المساحة اللازمة من ذاكرة الحاسب لهذا المتغير وقتهته . ويتم الإعلان عن أى متغير باستخدام إحدى الطرق الآتية :

عملية الإعلان عن المتغير التي سنشرحها هنا عملية اختيارية، بمعنى أنك يمكنك استخدام متغير دون أن يسبقه الإعلان عنه، وفي هذه الحالة سيتولى Access VBA حجز مساحة الذاكرة اللازمة للمتغير وقتهته تلقائيا ويكون من النوع Variant. إلا أننا ننصح دائما بالإعلان عن المتغير قبل استخدامه لكي تتجنب الوقوع في أخطاء يصعب عليك اكتشافها.



#### الإعلان بإضافة حرف مميز

تستخدم هذه الطريقة في تمييز نوع أى متغير وذلك بإضافة حرف معين إلى اسم المتغير ، والجدول التالي يبين شكل هذه الأحرف والنوع المقابل لها.

الحرف المستخدم	يستخدم مع المتغير
%	Integer
&	Long
!	Single
#	Double
@	Currency
\$	String

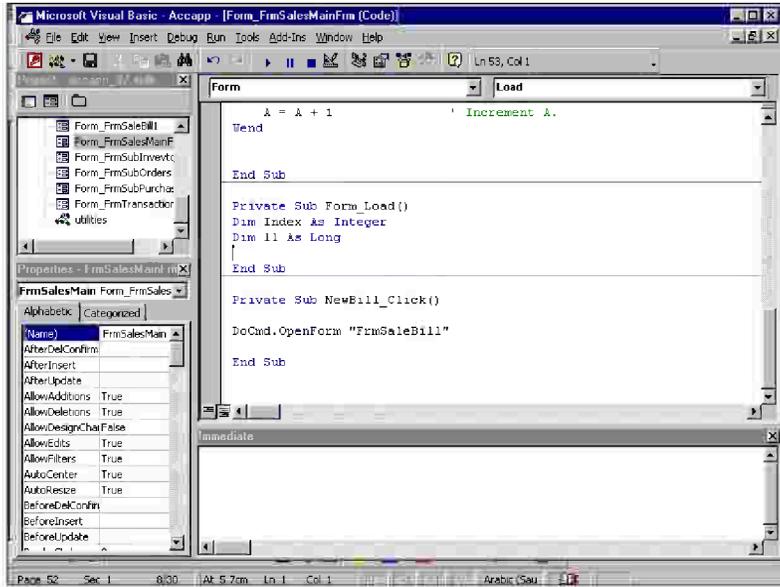
فمثلا الأمر

```
CompanyName$="CompuScience "
```

يعلن عن متغير من النوع النصي String

#### الإعلان باستخدام AS

يمكن تمييز نوع المتغير باستخدام الوظيفة AS مع أحد الأوامر Dim , Redim , Static , Global (سوف نشرح استخدامات هذه الأوامر بعد قليل) حيث يتم كتابة الأمر ثم اسم المتغير ثم كتابة الوظيفة AS ثم كتابة نوع المتغير كما في المثال التالي (شكل ٢-٣).



شكل ٢-٣ استخدام As للإعلان عن المتغير

### الإعلان باستخدام أمر تعريف دالة

في هذه الطريقة يتم استخدام أحد الأوامر التالية :

(CCur , CLng , CDbI , CInt, CStr , CSng, CVar)

للأنواع الآتية على التوالي

**Currency, Long, Double, Integer, String, Single, Variant**

وعند كتابة أى حرف بعد أى من هذه الأوامر تتحول كل المتغيرات التي تبدأ

بهذه الأحرف إلى نفس النوع المعلن عنه في الأمر المستخدم، وفي المثال التالي نستخدم الأمر

**CInt** للإعلان عن جميع المتغيرات الموجودة في البرنامج والتي تبدأ بحرف A على إنها من

النوع **Integer**.

**CInt A**

أما في المثال التالي فيتم الإعلان عن المتغيرات الموجودة في البرنامج والتي تبدأ

بأحد الأحرف التالية **B** أو **C** أو **D** على أنها من النوع **String**

**Cstr B-D**

لاحظ أنه لا ينبغي كتابة أكثر من حرفين متصلين بدون العلامة (-) بعد الأمر.



للإعلان عن أى متغير يبدأ بالحرف (A) أو بأحد الحروف من (D) إلى (F) أو من (X) إلى (Z) على أنه من النوع Double اكتب الأمر بالصورة التالية :

CDbl A , D-F , X-Z

### الالتزام بالإعلان عن المتغير

رغم أنه بالإمكان استخدام المتغير بدون الإعلان عنه ، إلا أن هذا الأمر ينطوى على مخاطر الوقوع في أخطاء خفية ، فإذا أردت أن تتجنب ذلك فعليك بالزام Access VBA بعدم قبول أى متغير بدون الإعلان عنه مسبقاً ويتم ذلك بأن تضع الأمر Option Explicit في قسم الإعلانات (Declaration Section) في الإجراء الذى تريد استخدامه فيه. في هذه الحالة سيعرض عليك Access VBA رسالة خطأ عند ورود أى متغيرات لم يسبق الإعلان عنها، وستتضمن رسالة الخطأ اسم المتغير. ينحصر تأثير أمر Option Explicit على الإجراء الذى ورد به فقط، ولذلك يجب وضعه في قسم الإعلانات في الإجراء الذى تريد تنفيذه عليه.

### مدى استخدام المتغير وعمره Lifetime and Scope of Variable

يقصد بمدى استخدام المتغير Scope of Variable الإجراءات والوحدات النمطية التى ستأثر به ، أى الأماكن التى يمكن أن يستخدم فيها هذا المتغير داخل البرنامج. أما عمر المتغير Lifetime of variable فيقصد به المدة التى سيبقى المتغير خلالها محتفظاً بقيمته الحالية داخل الذاكرة دون أن يفقدها، وتنقسم المتغيرات من حيث مدة بقائها في الذاكرة ومداتها إلى متغيرات عامة، ومتغيرات على مستوى الوحدة النمطية، ومتغيرات على مستوى الإجراء. وفيما يلي نوضح كل نوع من هذه الأنواع الثلاثة والأمر الذى يستخدم للإعلان عنه

#### ١ . المتغيرات العامة

هى المتغيرات التى يمكنك استخدامها من أى مكان داخل البرنامج أو التطبيق،

وتبقى في ذاكرة الحاسب طوال فترة عمل البرنامج، فإذا انتهى البرنامج تحذف من الذاكرة. ولذلك يجب أن يعلن عن المتغير العام من خلال الوحدة النمطية، لكي تتعرف عليه جميع الإجراءات الموجودة في جميع الوحدات النمطية بالبرنامج أو التطبيق.

يستخدم الأمر **public** للإعلان عن المتغيرات العامة. في المثال التالي يتم الإعلان عن متغير عام لكي تستخدمه جميع الإجراءات في جميع الوحدات النمطية من نوع **Integer** واسمه **ABC**:

```
public ABC AS integer
```

## ٢ . المتغيرات على مستوى الوحدة نمطية

يتمكنك الإعلان عن متغير وتقييده على مستوى وحدة نمطية. في هذه الحالة لن تستطيع استخدام المتغير إلا من خلال الوحدة النمطية التي أعلنت عنه فيها، ولن تستطيع استخدامه خارجها. فترة عمل هذا النوع من المتغيرات هي أيضا فترة عمل البرنامج، أي أن الفرق بينها وبين المتغيرات العامة هو في المدى الذي تستخدم فيه فقط.

للإعلان عن متغير من هذا النوع استخدم الأمر **Private** بدلا من الأمر **public** في المثال التالي يتم الإعلان عن متغير من نوع **String** واسمه

```
CompanyName لكي يستخدم فقط مع الوحدة النمطية التي يوجد بها  
Private CompanyName AS String
```

وهذا الأمر يمكن إدخاله من خلال الإجراء . لإدخال الأمر على مستوى الوحدة النمطية استخدمه بنفس الطريقة التي تستخدمها للإعلان عن المتغير العام ، مع فارق واحد وهو استخدام أمر **Private** بدلا من أمر **public**.

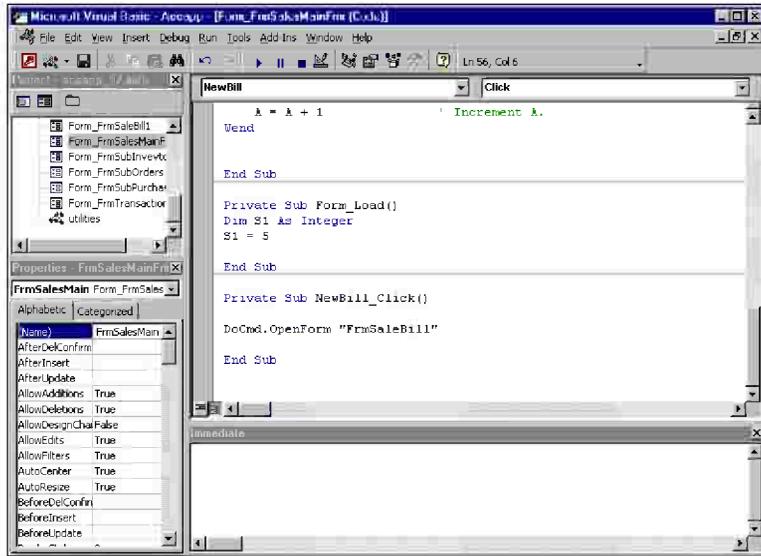
## ٣ . متغيرات على مستوى النموذج أو التقرير

ويتم تعريف هذه المتغيرات داخل قسم الإعلان بالنموذج أو التقرير وبالتالي يتم استخدامها حينما يتم فتح هذا النموذج أو ذلك التقرير وينتهي عمرها داخل الذاكرة بإغلاق النموذج أو التقرير.

#### ٤ . متغيرات على مستوى الإجراء (متغيرات محلية)

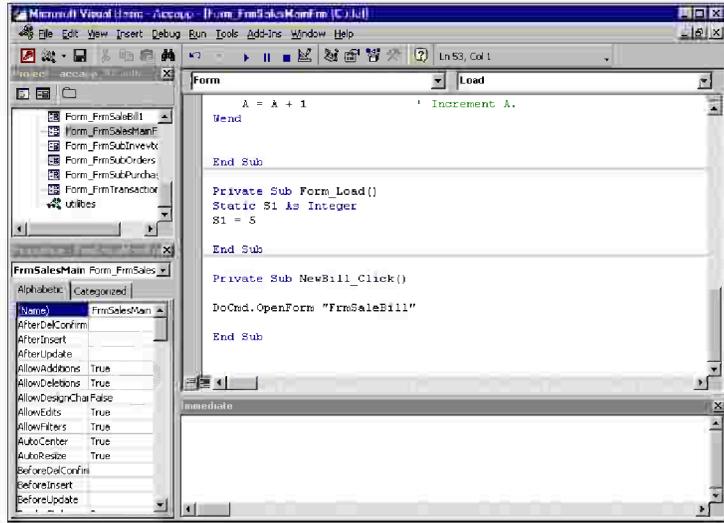
يقتصر مدى هذه المتغيرات على الإجراء الموجودة به فقط ، ولا يمكن استخدامها في أى مكان غيره، وهى بهذا تعتبر أقل المتغيرات مدىً. أما من حيث عمرها فهى تبقى موجودة بالذاكرة حتى بعد أن ينتهى الإجراء الذى أعلن فيه عنها. وبهذا يتضح أن الفرق بين هذه المتغيرات والمتغيرات العامة أو المتغيرات على مستوى الوحدة النمطية في مداها فقط ، حيث لا يتعدى مداها الإجراء الذى أعلن عنها فيه . يستخدم لهذا الغرض الأمر **Static** ويتضح ذلك من المثالين التاليين. في المثال الأول يظل المتغير **S1** محتفظاً بقيمته فترة تنفيذ الإجراء **Load Form** وبمجرد الخروج من الإجراء ستكون قيمته تساوى صفر بينما في حالة استخدام المثال الثانى فإن المتغير سيحتفظ بقيمته بعد تنفيذ نفس الإجراء السابق.

#### المثال الأول (شكل ٣-٣)



شكل ٣-٣ تعريف المتغير بحيث يصبح صفراً بمجرد الانتهاء من الإجراء.

#### المثال الثانى (شكل ٣-٤)



شكل ٤-٣ تعريف المتغير بحيث يحتفظ بقيمته حتى بعد الانتهاء من الإجراء.

يوفر عليك تحديد مدى المتغيرات وعمرها استهلاك مساحة من الذاكرة بدون داع. فمثلا إذا كنت تريد استخدام متغير في أكثر من وحدة نمطية ، فيجب أن تعلن عنه كمتغير عام بالأمر **public** ، وإذا كنت تحتاج للمتغير في وحدة نمطية واحدة فقط، استخدم الأمر **Private** للإعلان عنه. أما إذا كنت تحتاج للمتغير مؤقتا في هذا الإجراء فقط ، استخدم أمر **Static** ليبقى مداه داخل الإجراء فقط.



### استخدام المصفوفات Arrays

المصفوفة هي سلسلة من المتغيرات ترتبط مع بعضها وتسمى بنفس الاسم وعند استدعاء أى متغير فيها يتم الإشارة إليه باستخدام الرقم المسلسل الخاص به داخل المصفوفة. تستخدم كل لغات البرمجة تقريبا فكرة المصفوفات، لأنها توفر وقت وجهد المبرمج كما أنها تؤدي إلى صغر حجم البرنامج وسهولة متابعته. ويتضح ذلك من المثال التالي:

إذا أردت تصميم برنامج لشئون العاملين بشركتك بدون استخدام المصفوفات وبفرض أن عدد الموظفين في الشركة ١٠٠ موظف . فان الحل باستخدام المتغيرات العادية

يتطلب الإعلان عن ١٠٠ متغير ثم تكرار الأوامر التي تتعامل مع الموظفين المائة ، ولاشك أن هذه مسألة شاقة ومطولة.

أما الحل الأمثل في هذه الحالة فهو استخدام مصفوفة تتكون من ١٠٠ عنصر، والصيغة التي تحقق هذا الغرض لهذه المصفوفة كما يلي

### Private Names(99) AS String

وعن هذه الصيغة نوضح مايلي:

- لأننا نريد أن نحتفظ بأسماء الموظفين طوال فترة تنفيذ البرنامج، فقد استخدمنا الأمر **Private** للإعلان عن المصفوفة باعتبارها مصفوفة عامة ، ويتم تحديد مدى احتياجك لأي مصفوفة باستخدام أمر **Dim** أو **Static** بنفس الطريقة التي أوضحناها عند شرح مدى المتغير وعمره فإذا كنت داخل الوحدة النمطية فقط استخدام الأمر **Dim** ، وإذا كنت داخل إجراء استخدم الأمر **Static** (راجع مدى استخدام المتغير وعمره في البند السابق).

- اسم المصفوفة هو **Names**.
- إجمالي عدد عناصر المصفوفة (عدد الموظفين) هو ١٠٠ وتكتب ٩٩ بين قوسين حيث أن المصفوفة تبدأ دائما من الصفر.
- متغيرات هذه المصفوفة من النوع **String** وذلك لأن الاسم ما هو إلا سلسلة من الحروف.

فإذا أردت التعامل مع الموظف رقم ١ في المسلسل العام وبفرض أن اسمه "أنس عبدالرازق" ، فإنك ستكتب الأمر التالي

**Names(0) = "أنس عبد الرازق"**

وللإشارة إلى الموظف رقم ٢ في المسلسل وهو "جهاد عبدالرازق" ، قم بكتابة الأمر بنفس الطريقة كما يلي:

**Names(1) = "جهاد عبد الرازق"**

لتحديد بداية الرقم المسلسل للمصفوفة من رقم ١ بدلاً من رقم صفر استخدم

الأمر التالي في قسم الإعلانات وقبل الإعلان عن أى مصفوفة داخل البرنامج

### Option Base 1

حيث أن الأمر **Option Base 1** أمر افتراضي داخل اللغة يتم بمجرد تشغيل

البرنامج دون أن يكتب شيئاً.

بعد تحديد بداية الرقم المسلسل في المصفوفة على أنه الرقم (1) يجب استبدال

الأوامر السابقة والخاصة بالإشارة إلى الموظفين رقم 1 ، 2 بالأمرين التاليين:

**Names(1) = "أنس عبدالرازق"**

**Names(2) = "جهاد عبدالرازق"**

### استخدام المصفوفات الديناميكية *Dynamic Arrays*

كما لاحظت في المثال السابق أننا استخدمنا مصفوفة عدد عناصرها ثابت وتسمى

**Fixed Size Array**، وعند الإعلان عن هذا النوع من المصفوفات يقوم **Access**

**VBA** بحجز مساحة من الذاكرة تتسع لجميع عناصر المصفوفة ، فإذا كانت عناصر

المصفوفة غير مستغلة كلها فإن هذا يعنى استغلال سئ للذاكرة. والحل في هذه الحالة هو

استخدام المصفوفة الديناميكية. تعتمد فكرة المصفوفة الديناميكية على استخدام عدد من

العناصر يتحدد بحسب حاجة البرنامج فقط

يتم الإعلان عن المصفوفات الديناميكية (أى متغيرة الحجم) بدون تحديد حجمها كما

في المثال التالي

**Dim AA ( )**

وعندما ترغب في تحديد حجمها على أنه 50 عنصر استخدم الأمر التالي

**Redim AA (50)**

ولإعادة تحديد حجمها إلى 100 عنصر بدلاً من 50 استخدم الأمر التالي

**Redim AA (100)**

تظهر فائدة استخدام الأمر **Redim** الآن في استغلال الجزء المطلوب فقط من الذاكرة

دون زيادة.

لا يستخدم الأمر **Redim** إلا داخل إجراء فقط (أى لا يستخدم في قسم الإعلانات مثل الأمر **Dim**).



### المصفوفات متعددة الأبعاد *Multidimensional Arrays*

تكلمنا فيما سبق عن المصفوفات ذات البعد الواحد وطرق تثبيت حجمها وتغييره داخل البرنامج والآن نتكلم عن المصفوفات ذات الأبعاد المتعددة وكيفية الإعلان عنها والتعامل معها.

يتم كتابة المصفوفات ذات الأبعاد المتعددة بالترتيب التالي: اسم المصفوفة ثم إجمالي عدد عناصر البعد الأول. (كما في الحالات السابقة) ثم إجمالي عدد عناصر البعد الثاني وتكتب بهذه الصورة

#### **Dim BB (5,6) AS integer**

حيث **BB** : هو اسم المصفوفة ، **5** : عدد عناصر البعد الأول **6** : عدد عناصر البعد الثاني ، وجميع القيم داخل المصفوفة من النوع **Integer**. ويكون مجموع العناصر داخل المصفوفة هو **٤٢** عنصراً ، أى حاصل ضرب البعد الأول (من صفر إلى **٥** يساوى **٦** عناصر ) في البعد الثاني (من صفر إلى **٦** يساوى **٧** عناصر)

يفضل كتابة هذا الأمر : **Option Base 1** قبل الإعلان عن المصفوفة حتى تكون بداية المصفوفة هي القيمة **(١,١)** بدلاً من **(٠,٠)** وهذا يساعدك على تحديد رقم المتغير بطريقة سريعة وسهلة.



وتتم الإشارة إلى العناصر داخل المصفوفة باستخدام البعدين هكذا:

**A= BB(2,3)**

ومعناها العنصر الموجود عند التقاء الصف الثالث والعمود الرابع.

### الثوابت *Constants*

الثوابت كما هو واضح من اسمها عبارة عن اسم يحمل قيمة ثابتة لا تتغير أثناء تنفيذ البرنامج ، وهي عكس المتغيرات التي تتغير قيمتها تبعاً للمدخلات. ومع ذلك فهي تتشابه مع المتغيرات في أمرين هما اسم الثابت ومداه كما سيتضح بعد قليل. ولتوضيح

فكرة الثابت نضرب المثال التالي:

إذا كان عملك يتطلب مجموعة من العمليات الحسابية ترتبط بوحدة ثابتة مثل وحدة القياس "المتر" الذى يساوى مائة سنتيمتر ، فيمكنك الإعلان عن ذلك بالأمر التالى:

**Const Meter = 100**

وهذا يفيدك عندما تكون جميع حساباتك بالنسبة للوحدة سنتيمتر. فبدلاً من حساب قيمة المتر وكتابة الرقم (١٠٠) في كل مرة سيتم كتابته الثابت **Meter** في جميع التعليمات المطلوبة داخل البرنامج وهى فائدة كبيرة تجعل برنامجك سهلاً وبسيطاً.

فائدة أخرى يمكن الحصول عليها من استخدام الثوابت، فمثلاً في حالة تعديل كل حساباتك لتصبح منسوبة لوحدة الملليمتر بدلاً من السنتيمتر (والمعروف أن المتر = ١٠٠٠ ملليمتر) فبدلاً من إجراء هذا التعديل في جميع إجراءات برنامجك (وهو كتابة الرقم ١٠٠٠ بدلاً من الرقم ١٠٠)، فيكفى أن تعدل الرقم ١٠٠ ليصبح ١٠٠٠ في نفس الأمر كالتالى

**Const Meter = 1000**

وبذلك تتم عملية التعديل مرة واحدة فقط لتعطى النتيجة المطلوبة.

*تسمية الثابت*

يخضع اسم الثابت لنفس الشروط التى شرحناها عند اختيار اسم المتغير وهى ألا يزيد عدد حروفه عن ٤٠ حرفاً ، وأن يبدأ بحرف هجائى ، وألا يستخدم إحدى الكلمات

المحجوزة لـ **Access VBA**

*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

**ثوابت عامة:** إذا أردت أن يكون الثابت عاماً أى يمكن استخدامه من أى مكان في البرنامج فيجب أن تعلن عنه في الوحدة النمطية بشرط أن يسبق الإعلان عنه كلمة

**public** هكذا

**public Const Comp\_Name= "CompuScience"**

ثوابت على مستوى الوحدة النمطية: لكي تستخدم الثابت في وحدة نمطية فقط ، يجب

أن تعلن عنه في قسم الإعلانات في هذه الوحدة النمطية تسبقه كلمة **private** هكذا

**private Const N0\_EMP=20**

ثوابت على مستوى الإجراء: لكي تستخدم الثابت مؤقتا داخل إجراء معين ، أعلن عن

الثابت داخل هذا الإجراء بنفس الطريقة السابقة.

### تعميد الشروط واتخاذ القرارات

غالبا ما يكون التسلسل الطبيعي لكتابة أوامر الإجراءات هو الطريقة الدائمة

لتنفيذ هذه الأوامر فيتم تنفيذ أوامر السطر الأول ثم أوامر السطر الثاني وهكذا إلى نهاية

الإجراء. إلا أنه في بعض الحالات الخاصة قد تحتاج للخروج عن هذا التسلسل الطبيعي

لأوامر الإجراءات وفي هذه الحالة يجب أن نستخدم أحد أدوات التفرع أو الشرط

الموجودة في Access VBA .

### أنواع التفرع

يستخدم Access VBA نوعان من التفرع هما :

١. التفرع غير المشروط.

٢. التفرع المشروط.

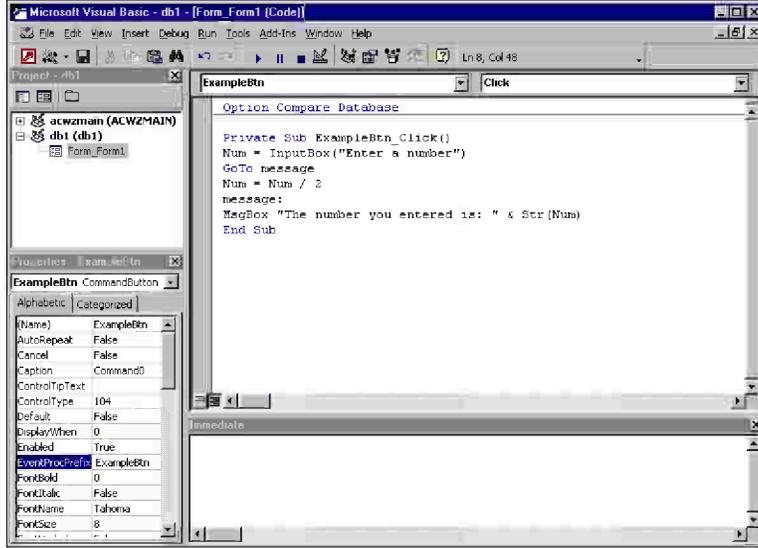
وفيما يلي نوضح بالتفصيل المقصود بكل نوع مع إعطاء الأمثلة المناسبة.

### التفرع غير المشروط

ومعناه الانتقال إلى مكان آخر داخل البرنامج، بعبارة أخرى تغيير ترتيب تنفيذ التعليمات

الواردة بالبرنامج أو الإجراء بدون شرط، كأن تقول اذهب إلى أمر كذا. ويوضح المثال

التالي (شكل ٣-٥) فكرة التفرع غير المشروط باستخدام الأمر **GoTo** .



شكل ٥-٣ مثال للتفرع غير المشروط

في هذا المثال يتم تنفيذ أمر **GOTO** الغير مشروط وينتقل التنفيذ إلى العنوان **Message** ولذلك يبقى المتغير **Num** محتفظا بالرقم الذي أدخلته ،، ولذلك سيظهر في مربع الرسالة الرقم الذي أدخلته وليس نصف قيمته . بعبارة أخرى ستقفز **Access** من السطر الثاني إلى السطر الرابع بدون تنفيذ الأمر التالي :-

**Num = Num / 2**

## التفرع المشروط

يتطلب وجود تعبير منطقي يحتمل أن يكون صحيحا أو خاطئا ، فإذا كان صحيحا يتم اتخاذ إجراء معين أو تنفيذ أمر ما ، أما إذا كان خاطئا فيتم تنفيذ إجراء أو أمر آخر ، فمثلا الأمر

**If password="CS" Then**

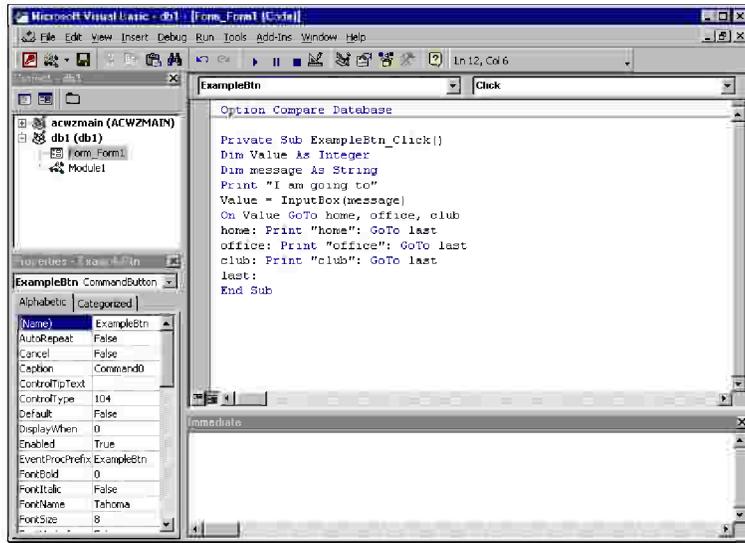
يحتوى على تعبير منطقي أو شرط فإذا كان المتغير **password** يحتوى على القيمة **CS** فان الشرط يعد صحيحا (**True**) وبالتالي يتم تنفيذ الأمر أو الأوامر التي تلى كلمة **Then** ، أما إذا لم يشتمل المتغير **password** على القيمة **CS** فان الشرط يعد خاطئا

(False) وبالتالي لن تنفذ الأوامر التي تلي كلمة Then.

ولأن التفريع المشروط ينطوى دائماً على شرط يحتمل الصواب أو الخطأ، أى أن البرنامج يقارن قيمتين ليتحقق من وقوع الشرط صحيحاً من عدمه، فإن عملية المقارنة تتم باستخدام أحد عوامل المقارنة أو العوامل المنطقية التي سيرد شرحها بالتفصيل في الفصل الثامن من هذا الكتاب.

ويوضح المثال التالي (شكل ٦-٣) فكرة التفريع المشروط باستخدام الأمر ON...

## GOTO



شكل ٦-٣ مثال للتفريع غير المشروط

وهذا المثال يوضح الفرق بين الأمر GoTo للتفريع غير المشروط والذي سبق وبينه وبين الأمر ON...GoTo للتفريع المشروط. وفيما يلي توضيح لفكرة هذا الإجراء.

- في السطر الأول والثاني يتم تعريف المتغير Value1 على أنه من النوع Integer والمتغير message على أنها رسالة من النوع String.
- في السطر الثالث يتم طباعة الرسالة "I am going to"
- في السطر الرابع يتم إدخال قيمة المتغير message حرفياً من خلال لوحة المفاتيح

- ويتحول إلى قيمة رقمية توضع في المتغير Value1 عن طريق الدالة Input Box .
- في السطر الخامس يتم تنفيذ التفرع المشروط ON...GO TO على أساس أن القيمة المدخلة للمتغير Value1 هي "١" أو "٢" أو "٣" .
- في السطر السادس عند إدخال قيمة المتغير = ١ يتم الانتقال إلى العنوان home وهذا تفرع مشروط بقيمة المتغير الذى أدخلته باستخدام الأمر ON...GoTo لطباعة كلمة "Home" .
- عند إدخال قيمة المتغير = ٢ يتم الانتقال إلى العنوان office وهذا تفرع مشروط باستخدام الأمر ON...GoTo لطباعة كلمة "Office" .
- عند إدخال قيمة المتغير = ٣ يتم الانتقال إلى العنوان Club وهذا تفرع مشروط باستخدام الأمر ON...GoTo لطباعة كلمة "Club" .
- بعد طباعة واحدة من الرسائل الموجودة في الأسطر السادس أو السابع أو الثامن ، يتم تنفيذ التفرع غير المشروط goto last لينتقل التنفيذ إلى نهاية الإجراء .

## أدوات الشرط

يستخدم Access VBA نوعان من أدوات الشرط هما :

١ . أداة الشرط IF

٢ . أداة المقارنة Select Case

## أداة الشرط IF

دائما ترتبط بتحقيق شرط معين ، فإذا وقع الشرط صحيحا يتم تنفيذ تعليمة أو مجموعة من التعليمات، وتستخدم بتركيبات 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 صحيحا فان التركيب يأخذ الصورة العامة التالية:

### If <condition> Then <command>

وفي هذا التركيب يتم تقييم الشرط (Condition) الوارد بالتعليمة ، فإذا كان صحيحا ينفذ Access VBA الأمر <Command> الذى يلي كلمة Then ، وإلا فإنه يتجاهله ، ويصلح هذا التركيب عندما تريد تنفيذ أمر واحد في حالة تحقق الشرط .  
والمثال التالى يوضح هذه الفكرة

**IF A=5 Then print "OK "**

في هذا المثال الجملة الشرطية عبارة عن أداة شرط هي IF الشرط هو مقارنة محتويات المتغير A بالقيمة 5 ويرتبط جواب الشرط في البداية بكلمة Then ويتم تنفيذ طباعة الرسالة "OK" (جواب الشرط) في حالة التحقق من تساوى المتغير A للقيمة 5 .  
لاحظ أن الجملة 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 صحيحا فان تركيب If...Then يأخذ الصورة العامة التالية:

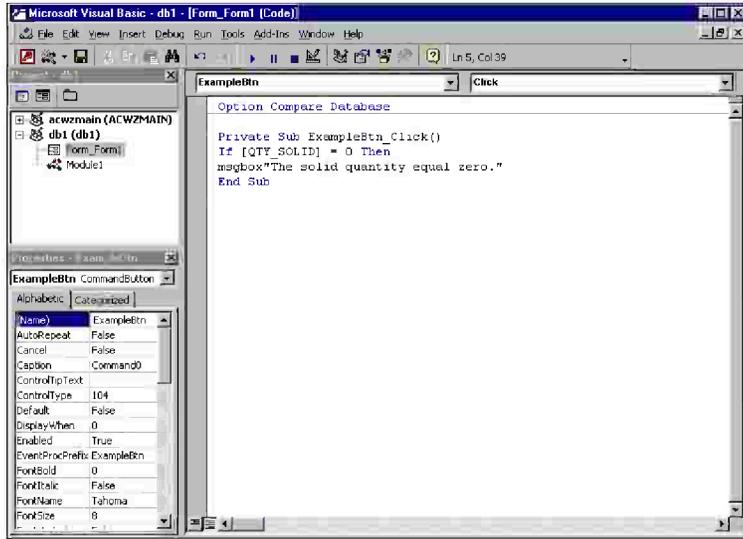
**If <condition> Then**

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

.....

**End if**

وكما تلاحظ في هذه الصورة العامة أنه لا توجد أوامر بعد كلمة Then وأن التركيب ينتهى بعبارة End if كما في هذا مثال (شكل ٧-٣) :



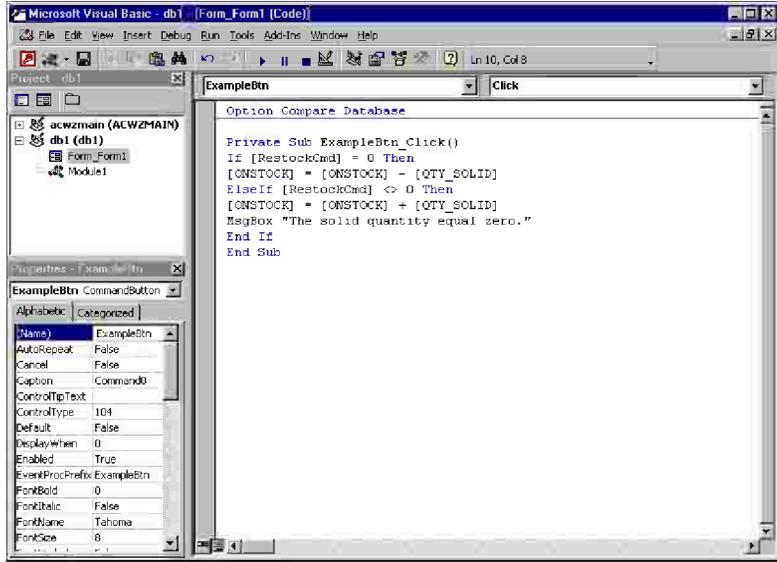
شكل ٧-٣ استخدام جملة IF

### التركيب IF...Then...Else

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

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

وفي هذا التركيب يتم تقييم الشرط (Condition) الوارد به ، فإذا كان صحيحا ينفذ VBA Access الأمر أو الأوامر (Commands) التي تلي كلمة Then ، وإلا فإنه ينفذ الأمر أو الأوامر التي تلي كلمة Elseif ، ويصلح هذا التركيب عندما تريد تنفيذ مجموعة أوامر إذا وقع الشرط صحيحا ومجموعة أخرى إذا وقع الشرط خاطئا.. والمثال التالي (شكل ٨-٣) يوضح هذه الفكرة.



شكل ٨-٣ استخدام التركيب If...Then...Else

في المثال السابق يتم اختبار قيمة [RestockCmd] إذا كانت القيمة تساوي صفراً سيتم طرح الكمية المباعة من كمية المخزون أما إذا كانت غير ذلك سيتم الجمع.

استخدام جملة IF لعمل أكثر من مقارنة

في المثال التالي (شكل ٩-٣) يقوم الإجراء بحساب معدل ويقوم باختياره فإذا كان خارج النطاق من ٠ إلى ١٠٠ تظهر الرسالة "رقم غير ملائم" وفي حالة إدخال معدل جديد غيره يتم مقارنته مرة أخرى وطباعة الرسالة المناسبة (امتياز أو جيد جداً أو جيد أو مقبول).

```

Option Compare Database

Private Sub ExampleBtn_Click()
    Dim score As Integer
    Dim message, title As String
    message = "أدخل الرقم من ٠ إلى ١٠٠"
    title = "تصفح حساب النتيجة"
    score = InputBox(message, title)

    If score > 100 Or score < 0 Then
        Print "رقم غير ملائم"
    Else
        If score >= 85 Then
            Print "النتيجة ممتاز"
        ElseIf score >= 75 Then
            Print "النتيجة جيد جدا"
        ElseIf score >= 65 Then
            Print "النتيجة جيد"
        Else: Print "النتيجة مقبول"
        End If
    End If
    Print "أعد المحاولة"
End Sub
    
```

شكل ٩-٣ استخدام جملة If لعمل أكثر من مقارنة

في مثل هذه الحالات التي تتعدد فيها الاحتمالات ننصح باستخدام التركيب Select Case للمقارنة والذي سنشرحه بعد قليل.



### استخدام التركيب Select Case للمقارنة

تصلح تركيبات جملة IF إذا كان جواب الشرط عبارة عن احتمال واحد أو احتمالين. أما إذا كنت تتوقع عند تقييمك لشرط معين عدة احتمالات ، فمن الأفضل أن تستخدم التركيب Select Case ، وأبسط صورة للتركيب Select Case هي:

```

Select Case VarName
    [Case Expressionlist1
        [Statementblock1] ]
    [Case Expressionlist2
        [Statementblock2] ]
    -----
    -----
    [Case Else
        [Statementblockn] ]
End Select
    
```

حيث:

اسم المتغير الذي سيتم البحث في مجموعة الحالات التالية عما يساويه، ويمكن أن يكون تعبيراً حرفياً أو رقمياً	<b>VarName</b>
تعبير حرفي أو رقمي ، ويجب أن يتطابق نوع التعبير مع نوع التعبير المذكور في <b>VarName</b> . يمكن أن يكون تعبيراً واحداً أو أكثر من تعبير مفصولين بعلامة <b>Comma</b>	<b>Expressionlist1</b>
الجمل التي تنفذ إذا تساوت قيمة <b>VarName</b> مع التعبير أو التعبيرات المذكورة في <b>Expressionlist1</b>	<b>Statementblock1</b>
تعبير حرفي أو رقمي ، ويجب أن يتطابق نوع التعبير مع نوع التعبير المذكور في <b>VarName</b> . يمكن أن يكون تعبيراً واحداً أو أكثر من تعبير مفصولين بعلامة <b>Comma</b>	<b>Expressionlist2</b>
الجمل التي تنفذ إذا تساوت قيمة <b>VarName</b> مع التعبير أو التعبيرات المذكورة في <b>Expressionlist2</b>	<b>Statementblock2</b>
الجمل التي تنفذ في حالة عدم تحقق في أى من الحالات السابقة ونوضح أن <b>Expressionlist</b> يمكن أن تأخذ أحد الأشكال الآتية:	<b>Statementblockn</b>
• قيمة أو أكثر يتم مقارنتهم مع القيمة المذكورة في <b>VarName</b> . إذا استخدمت أكثر من قيمة ، افصل بينهم بعلامة <b>Comma</b> مثل:	
Case "!" , "?" , "." , ";"	• مجموعة قيم تقع في مدى معين ويفصل بينها بكلمة <b>To</b> مثل:
Case 10000 To 49999.99	• المعامل <b>IS</b> متبوعاً بعامل علائقي مثل <b>=</b> , <b>&lt;</b> , <b>&gt;</b> , <b>&lt;&gt;</b> , <b>=</b> مثل:
Case IS < 10000	

مثال ١

يستخدم المثال التالي التركيب **Select Case** في حالة تعدد الاختيارات باستخدام تعبير رقمي.

**Sales = InputBox ("Please Enter Sales Amount")**

**Select Case Val(Sales)**

**Case 10000 To 49999.99**

**Msg = "يجب زيادة المبيعات"**

**Case 50000 To 100000**

**Msg = "مبيعات مقبولة"**

**Case IS < 10000**

**Msg = "مبيعات منخفضة"**

**Case Else**

**Msg = "مبيعات عالية"**

**End Select**

**MsgBox Msg**

وفي هذا المثال إذا كان رقم المبيعات الذي أدخله المستخدم يقع في المدى من ١٠٠٠٠ إلى ٤٩٩٩٩.٩٩ ، فإن الرسالة التي ستظهر له هي " يجب زيادة المبيعات ". أما إذا كان يقع في المدى من ٥٠٠٠٠ إلى ١٠٠٠٠٠ فإن الرسالة ستكون "مبيعات مقبولة" . وإذا كان رقم المبيعات أقل من ١٠٠٠٠ فإن الرسالة ستكون "مبيعات منخفضة" . وإذا لم تقع أي من الحالات الثلاث المذكورة صحيحة فإن الرسالة التي ستظهر ستكون " مبيعات عالية"

مثال ٢

المثال التالي يستخدم التركيب **Select Case** لتقييم تعبير حرفي ، وهي تقييم كود آسكي (ASCII) المقابل للحرف الأول من السلسلة.

**Select Case letter\$**

**Case "A" To "Z"**

**Chartype\$ = "Upper Case"**

**Case "a" To "z"**

**Chartype\$ = "Lower Case"**

**Case "0" To "9"**

```
Chartype$ = "Number"  
Case "!", "?", ".", ";", ":"  
Chartype$ = "Punctuation"  
Case " "  
Chartype$ = "Empty"  
Case < 32  
Chartype$ = "Special Character"  
Case Else  
Chartype$ = "Unknown Character"
```

## التكرار والدوران Loop

ونعنى بالتكرار والدوران تكرار مجموعة من الأوامر عدد معين من المرات. ويتم تمييز مجموعة الأوامر المطلوب تكرارها داخل البرنامج وتحديد عدد مرات تكرارها باستخدام الأمر **For .. Next** ، أو حتى يتحقق شرط معين باستخدام الأمر **Do .. Loop** ، أو باستخدام الأمر **While .. Wend**.

### الأمر For .. Next

يتم استخدام الأمر **For .. Next** لتكرار مجموعة من الأوامر عدد معين من المرات ويأخذ الصورة التالية :

**For** الشرط

مجموعة الأوامر المطلوب تكرارها

-----  
**Next** اسم المتغير

فمثلا لطباعة الأرقام من ١ إلى ٥ يمكنك تكرار أمر الطباعة خمسة مرات كالاتى :

```
Print 1  
Print 2  
Print 3  
Print 4
```

**Print 5**

أما باستخدام الأمر **For .. Next** فسيتم الاستغناء عن هذا التكرار كالتالي :

**Dim I AS Integer**

**For I = 1 To 5**

**Print I**

**Next I**

وعن هذا التركيب نوضح مايلي:

- أعلننا عن المتغير (I) للحصول على أرقام متغيرة من 1 إلى 5.
- الأمر **For .. Next** يعمل على زيادة قيمة المتغير بمقدار (1) 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 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) في

المتغير (I) وهذا يعنى 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 التي

ستتم على المتغير في كل دورة. ففي الحالة السابقة لم نذكر الوظيفة **Step** في الأمر

**For I = 1 To 5**

لأن الزيادة التلقائية في كل دورة مقدارها 1 ما لم 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 في الأرقام 1 إلى 10 استخدم الأمر التالي :

**For I = 2 To 10 Step 2**

## الأمر While .. Wend

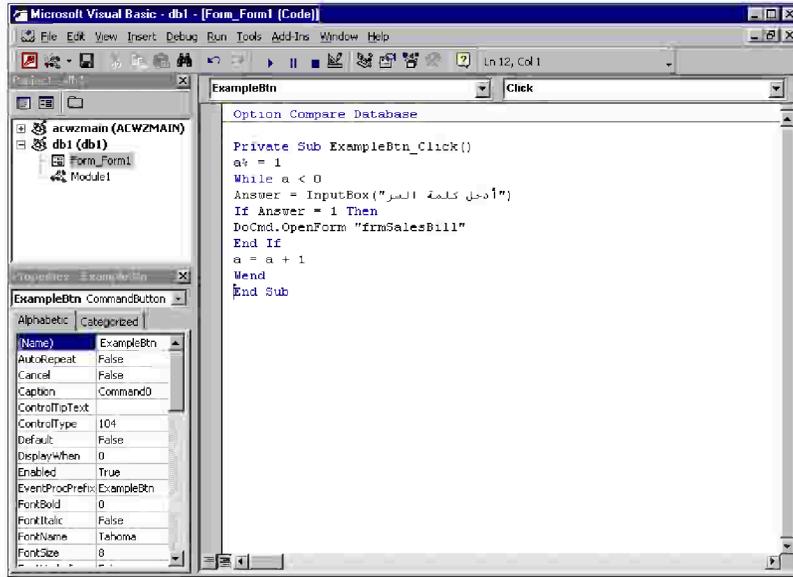
يتم استخدام الأمر While .. Wend لتكرار مجموعة من الأوامر طالما أنها تحقق شرط معين ويأخذ الصورة التالية :

**While** شرط

مجموعة الأوامر المطلوب تكرارها

-----  
**Wend**

يستخدم المثال في شكل ٣-١٠ أمر While...Wend لإعطاء المستخدم الفرصة حتى ٤ مرات لإدخال كلمة السر.



شكل ٣-١٠ أمر While... Wend لاختبار كلمة السر

## الأمر Do .. Loop

يستخدم الأمر Do .. Loop لتكرار مجموعة من الأوامر، ويستخدم بصيغ كثيرة

نوضحها فيما يلي

**الصيغة الأولى :** يتم التكرار طالما أنها تحقق شرط معين كما في الأمر **While .. Wend** الذي ذكرناه من قبل وهي تأخذ الشكل التالي :

**الشرط Do While**

مجموعة الأوامر

-----  
**Loop**

**الصيغة الثانية :** يتم تكرار مجموعة الأوامر حتى يتحقق الشرط. وهي تأخذ الشكل التالي:

**الشرط Do Until**

مجموعة الأوامر

-----  
**Loop**

مثال (شكل ١١-٣) التعليمات التالية تعطي نفس نتيجة أمر **Do While** السابق:

```

Microsoft Visual Basic - db1 - [Form_Form] [Code]
File Edit View Insert Debug Run Tools Add-Ins Window Help
Unit4, Col 1
Project - db1
  db1
    Form_Form
      Module1
Properties - ExampleBtn
ExampleBtn CommandButton
  Name: ExampleBtn
  AutoRepeat: False
  Cancel: False
  Caption: Command0
  ControlTipText:
  ControlType: 104
  Default: False
  DisplayWhen: 0
  Enabled: True
  EventProcPrefix: ExampleBtn
  FontBold: 0
  FontItalic: False
  FontName:Tahoma
  FontSize:8
  Option Compare Database
Private Sub ExampleBtn_Click()
a = 1
While a < 0
Do Until a < 0
Answer = InputBox("أدخل كلمة السر")
If Answer = 1 Then
DoCmd.OpenForm "frmSalesBill"
End If
a = a + 1
Loop
End Sub

```

شكل ١١-٣ أمر **Do Until...Loop**

ضع أمر **Do While** مكان أمر **Do Until** ستحصل علي نفس النتيجة

*الصيغة الثالثة* : يتم التكرار طالما أنهما تحقق شرط معين وهي تأخذ الشكل التالي :

**Do**

مجموعة الأوامر

-----

**Loop While** الشرط

*الصيغة الرابعة* : يتم تكرار مجموعة الأوامر حتى يتحقق الشرط. وهي تأخذ الشكل التالي :

**Do**

مجموعة الأوامر

-----

**Loop Until** الشرط

