سنشرح في هذا الفصل كيفية كتابة التعليمات والتركيبات الأساسية للغة Access VBA ،وتشمل الثوابت والمتغيرات والمصفوفات، وكيفية تحديد الشروط واتخاذ القرارات واستخدام الدوارات 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 والدوارات.

سنشرح في هذا الفصل أهم أوامر وتركيبات لغة Access VBA التى يحتاج إليها المبرمج في كتابة البرامج ، ولذلك فإننا نعتبر هذا الفصل ضرورى جدا لمن يريد أن يستعلم 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 التى 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 يلى 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 يلى:

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

لاحظ في هذه الطريقة أن كل سطر يحتوى على تعليمة واحدة فقط ، حيث يحتوى السطر الأول على متغير عددى هو Value1 . يشير المعامل % على أن القيمة المخصصة لهذا المتغير سوف تكون قيمه عددية. كما يستخدم المعامل "=" لتخصيص القيمة 6- إلى المتغير Value1. ويحتوى السطر الثانى 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 بينـــها باســـتخدام المعامل" : " كما يلى:

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

التعليقات (Comments)

التعليقات هى الملاحظات التى تكتبها داخل البرنامج دون أن يكون لها أى تأثير عند تنفيذ البرنامج، بعبارة أخرى لا يقـــوم Access VBA بتنفيـــذها ولا بمراجعتـــها. ويتميز المبرمج الماهر بالبراعة في استخدام التعليقات في شرح خطوات البرنـــامج بصـــورة جيدة تساعده في تذكر البرنامج والرجوع إليه للتعديل أو التطوير في أقل زمن ممكن.

إدخال التعليقات

هناك طريقتان لاستخدام التعليقات في برنامج Access VBA

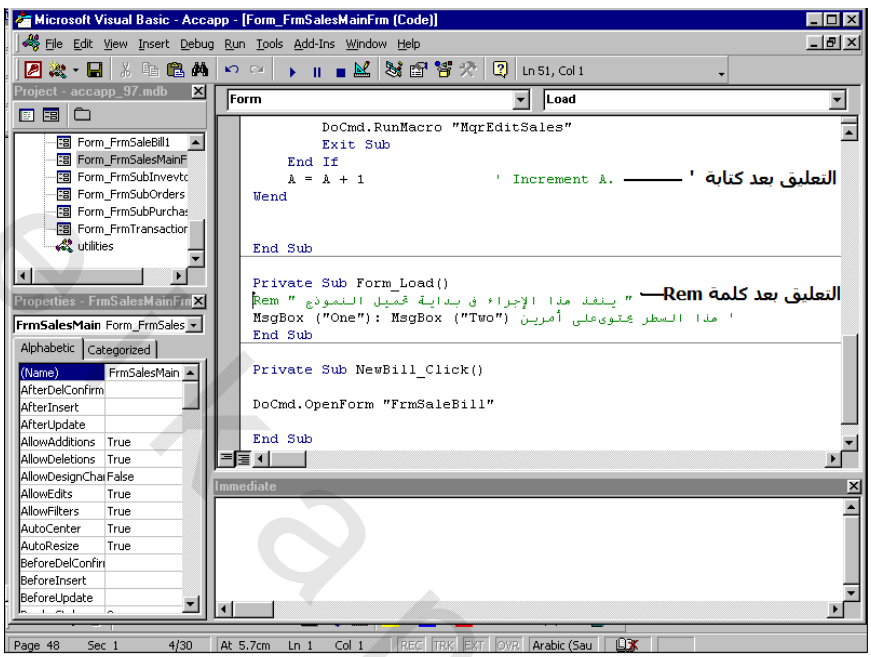
الطريقة الأولى : استخدام الأمر REM وهو اختصار للكلمة REMARK .

لاستخدام هذه الطريقة اكتب الأمر REM أولا ثـــم اكتب التعليـــق بعد ذلك، سيعتبر Access VBA كل ما يرد بعـــد كلمـــة REM تعليقا. تصلح هذه الطريقة في حالة إضافة سطر كامل كتعليق داخـــل البرنامج.

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

مثال

في المثال التالى يخصص السطر الأول كله كتعليق داخـــل البرنـــامج ولا يقـــوم Access VBA بتنفيذه ولا يؤثر على نتيجة البرنامج، بينما يشتمل السطر الثانى علـــى تعليق بالإضافة إلى تعليمات أخرى، كما يظهر في شكل ١-٣.



شكل ١-٣ كتابة التعليقات داخل البرنامج

المتغيرات Variables

المتغير (**Variable**) عبارة عن مكان يتم حجزه في ذاكرة الحاسب ويخصص لـه 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 هـو هـذا العميل الذى سيقع عليه اختيار المستخدم. في هذه الحالة تستخدم متغير لتضع فيه اسـم العميل. انظر المثال التالي:

HisName$=InputBox$("اكتب اسم العميل")

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

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 حرفا.
- يجب ألا يتضمن كلمة من الكلمات المحجوزة ،وهى الكلمات التى تستخدم في الأوامر والعبارات التى يستخدمها Access VBA ، فمثلا 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 التالى:

نوع المتغير	معناه	حجمه	مداه	الرمز
Byte	حرف	1 byte	من 0 إلى 256	لا يوجد
Boolean	منطقى	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 حقيقى 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 حقيقى (يحتوى على علامة عشرية عائمة) كبير نسبياً	8 byte	رقم هائل	#
Currency	عدد حقيقى 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 . وفى الأعداد مثل Double وفى الحروف مثل String	لا يوجد

تختلف القيمة "15" عن القيمة 15 حيث أن القيمة 15 قيمة عددية (حسابية) يمكن إجراء أى عملية حسابيه 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 الـــمتغير عبارة عن أمر يخبر Access VBA باسم المتغير ونوعه كى

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

عملية الإعلان عن المتغير التى سنشرحها هنا عملية اختيارية، بمعنى أنك يمكنك استخدام متغير دون أن يسبقه الإعلان عنه، وفى هذه الحالة سيتولى Access VBA حجز مساحة الذاكرة اللازمة للمتغير وتهيئته تلقائيا ويكون من النوع Variant. ألا أننا ننصح دائما بالإعلان عن المتغير قبل استخدامه لكى 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 التالى يبين شكل هذه الأحرف والنوع المقابل لها.

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

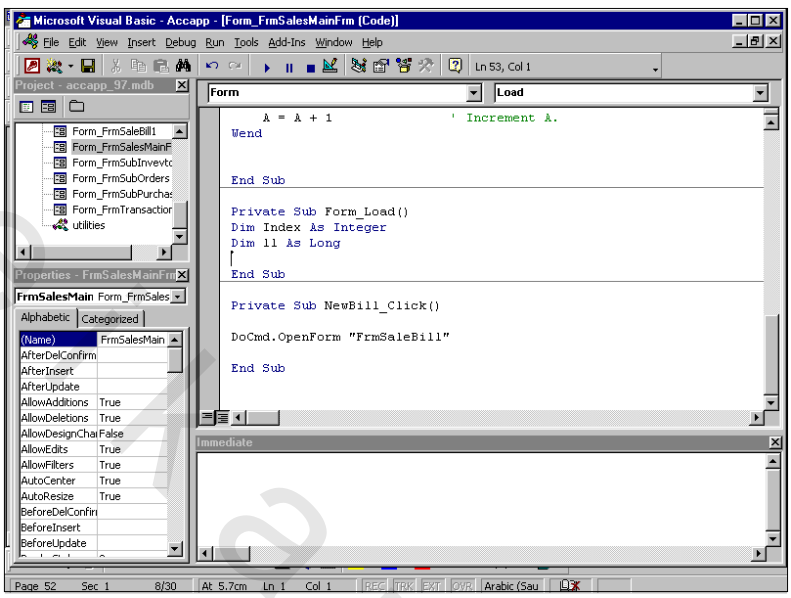
فمثلا الأمر

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

يعلن عن متغير من النوع النصى 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 التالى (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 , CDbl , CInt, CStr , CSng, CVar)

للأنواع الآتية على التوالى

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

وعند كتابة أى حرف بعد أى من هذه الأوامر تتحول كل المتغيرات التى تبـــدأ بهذه الأحرف إلى نفس النوع المعلن عنه في الأمر المستخدم، وفى المثال التالى نستخدم الأمر **CInt** للإعلان عن جميع المتغيرات الموجودة في البرنامج والتى تبدأ بحرف A على إنها مـــن النوع **Integer**.

CInt A

أما في المثال التالى فيتم الإعلان عن المتغيرات الموجودة في البرنامج والـــتى تبـــدأ بأحد الأحرف التالية **B** أو **C** أو **D** على أنها من النوع **String**

Cstr B-D

لاحظ أنه لا ينبغى 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 بإلزام 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 المتغيرScope of Variable الإجراءات والوحدات النمطية التى ستتأثر 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 محتفظا بقيمته الحالية داخل الذاكرة دون أن يفقدها، وتنقسم المتغيرات من حيث مدة بقائها في الذاكرة ومداها إلى متغيرات عامة، ومتغيرات على مستوى الوحدة النمطية، ومتغيرات على مستوى الإجراء. وفيما يلى 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، لكى تتعرف عليه جميع الإجراءات الموجودة في جميع الوحدات النمطية بالبرنامج أو التطبيق.

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

```
public ABC AS integer
```

٢ . *المتغيرات على مستوى الوحدة نمطية*

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

للإعلان عن متغير من هذا النوع استخدم الأمر **Private** بدلا من الأمر **public**

في المثال التالى يتم الإعلان عن متغير من نوع **String** واسمه **CompanyName** لكى يستخدم فقط مع الوحدة النمطية التى يوجد بها

```
Private CompanyName AS String
```

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

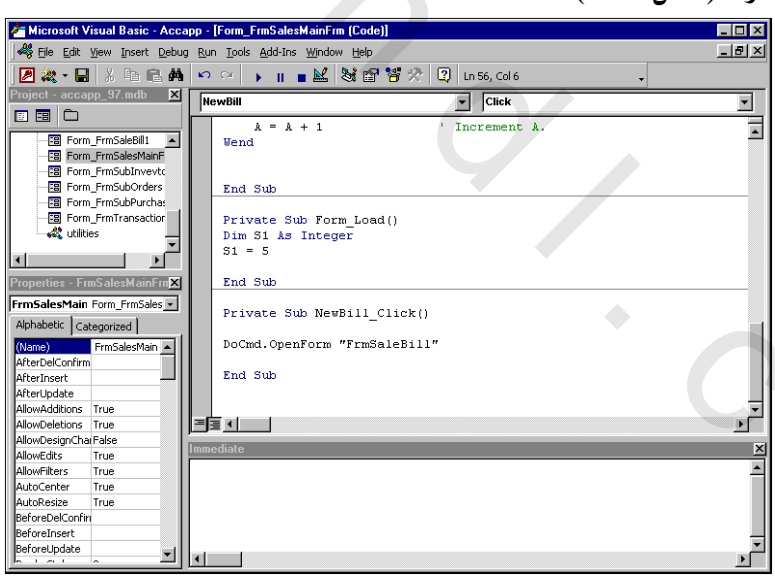
٣ . *متغيرات على مستوى النموذج أو التقرير*

ويتم تعريف هذه المتغيرات داخل قسم الإعلان بالنموذج أو التقرير وبالتالى يتم استخدامها حينما يتم فتح هذا النموذج أو ذاك التقرير وينتهى عمرها داخل الذاكرة بإغلاق النموذج أو التقرير.

٤ . متغيرات على مستوى الإجراء (متغيرات محلية)

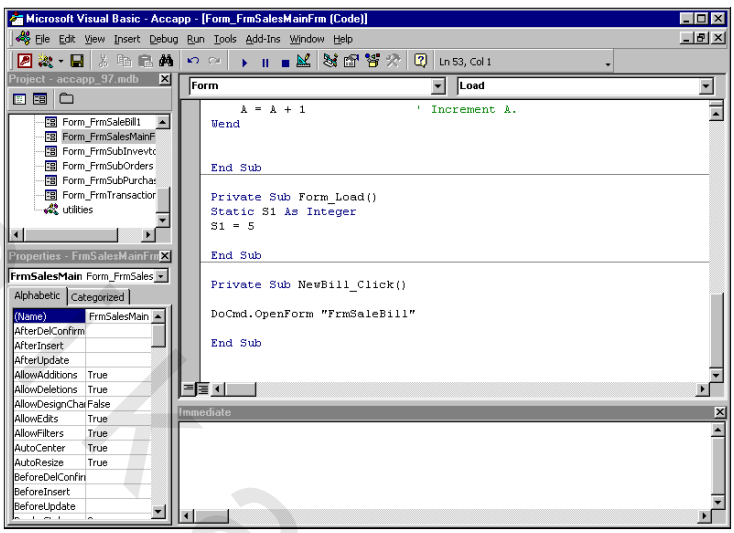
يقتصر مدى هذه المتغيرات على الإجراء الموجودة به فقط ، ولا يمكن استخدامها في أى مكان غيره، وهى بهذا تعتبر أقل المتغيرات مدىً. أما من حيث عمرها فهى تبقى موجودة بالذاكرة حتى بعد أن ينتهى الإجراء الذى أعلن فيه عنها. وبهذا يتضح أن الفرق بين هذه المتغيرات والمتغيرات العامة أو المتغيرات على مستوى الوحدة النمطية 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** محتفظا 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 ٤-٣)



شكل٤-٣ تعريف المتغير بحيث يحتفظ بقيمته حتى بعد الانتهاء من الإجراء.

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

استخدام المصفوفات Arrays

المصفوفة هى 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 تؤدى إلى صغر حجم البرنامج وسهولة متابعته. ويتضح ذلك من المثال التالى:

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

يتطلب الإعلان عن ١٠٠ متغير ثـــم تكرار الأوامر التى تتعامل مـــع المـــوظفين المائـــة ، ولاشك أن هذه مسألة شاقة ومطولة.

أما الحل الأمثل في هذه الحالة فهو استخدام مصفوفة تتكون من ١٠٠ عنصـــر، والصيغة التى تحقق هذا الغرض لهذه المصفوفة كما يلى

```
Private Names(99) AS String
```

وعن هذه الصيغة نوضح مايلى:

- لأننا نريد أن نحتفظ بأسماء الموظفين طوال فترة تنفيذ البرنامج، فقد استخدمنا الأمـــر **Private** للإعلان عن المصفوفة باعتبارها مصفوفة عامة ، ويتم تحديد مدى احتياجك لأى مصفوفة باستخدام أمر **Dim** أو **Static** بنفس الطريقة التى أوضحناها عنـــد 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 بـــين قوســـين 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 اسمـــه "أنـــس عبدالرازق" ، فإنك ستكتب الأمر التالى

```
Names(0)= "أنس عبد الرازق"
```

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

```
Names(1)= "جهاد عبد الرازق"
```

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

الأمر التالى 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 تشـــغيل البرنامج دون أن يكتب شيئا.

بعد تحديد بداية الرقم المسلسل في المصفوفة على أنه الرقم (١) يجب استبدال الأوامر السابقة والخاصة بالإشارة إلى الموظفين رقم ١، ٢ 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 يقـــوم **Access VBA** بحجز مساحة من الذاكرة تتسع لجميع عناصر المصفوفة ، فـــإذا كانـــت عناصـــر المصفوفة غير مستغلة كلها فان هذا يعنى استغلال سئ للذاكرة. والحل في هذه الحالة هـــو استخدام المصفوفة الديناميكية. تعتمد فكرة المصفوفة الديناميكية على استخدام عدد مـــن العناصر يتحدد بحسب حاجة البرنامج فقط

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

```
Dim AA ( )
```

وعندما ترغب في تحديد حجمها على أنه ٥٠ عنصر استخدم الأمر التالى

```
Redim AA (50)
```

ولإعادة تحديد حجمها إلى ١٠٠ عنصر بدلاً من ٥٠ استخدم الأمر التالى

```
Redim AA (100)
```

تظهر فائدة استخدام الأمر **Redim** الآن في استغلال الجزء المطلوب فقط من الـــذاكرة 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 والأن نتكلم عن المصفوفات ذات الأبعاد المتعددة وكيفية الإعلان عنها والتعامل معها.

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

```
Dim BB (5,6) AS integer
```

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

يفضل كتابة هذا الأمر : Option Base 1 قبل الإعلان عن المصفوفة حتى تكون بداية المصفوفة هى 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 ، وهى عكس المتغيرات التى تتغير قيمتها تبعا للمدخلات. ومع ذلك فهى تتشابه مع المتغيرات في 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 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 كتابه الثابت **Meter** في جميع التعليمات المطلوبة داخل البرنامج وهى فائدة كبيرة تجعل برنامجك سهلا وبسيطا.

فائدة أخرى يمكن الحصول عليها من استخدام الثوابت، فمثلا في حالة تعديل كل حساباتك لتصبح منسوبة لوحدة المليمتر بدلا 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 كالآتى

```
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 حرفا ، 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 عاما 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:لكى 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:لكى تستخدم الثابت مؤقتا داخل إجراء معين ، أعلن عن الثابت داخل هذا الإجراء بنفس الطريقة السابقة.

تحديد الشروط واتخاذ القرارات

غالبا ما يكون التسلسل الطبيعى لكتابة أوامر الإجراءات هو الطريقة الدائمة لتنفيذ هذه الأوامر فيتم تنفيذ أوامر السطر الأول ثم أوامر السطر الثانى وهكذا إلى نهاية الإجراء. إلا أنه في بعض الحالات الخاصة قد تحتاج للخروج عن هذا التسلسل الطبيعى لأوامر الإجراءات وفى هذه الحالة يجب أن نستخدم أحد أدوات التفرع أو الشرط الموجودة في **Access VBA** .

أنواع التفرع

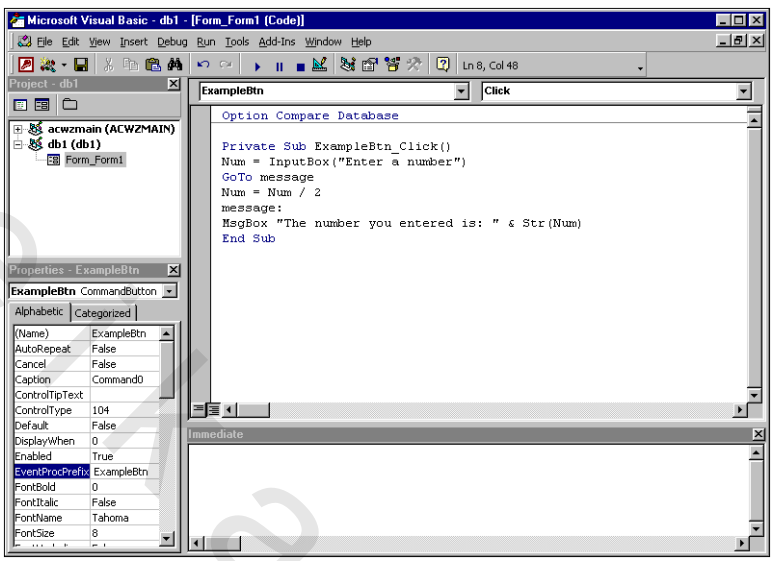
يستخدم **Access VBA** نوعان من التفرع هما :

١. التفرع غير المشروط.
٢. التفرع المشروط.

وفيما يلى 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 التالى (شكل ٥-٣) فكرة التفرع غير المشروط باستخدام الأمر **GoTo** .



شكل ٥-٣ مثال للتفريع غير المشروط

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

Num = Num / 2

التفرع المشروط

يتطلب وجود تعبير منطقى 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

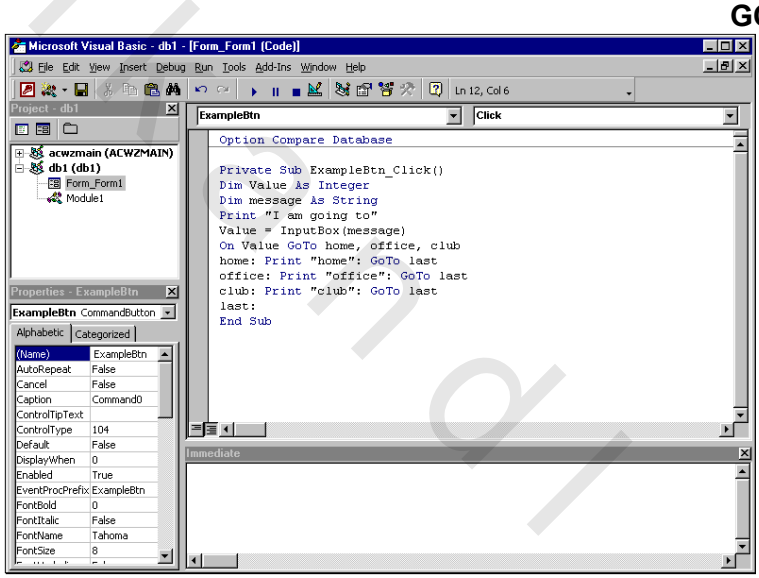
If password="CS" Then

يحتوى على تعبير منطقى أو شرط فإذا كان المتغير**password** يحتوى على القيمة **CS** فان الشرط يعد صحيحا (**True**) وبالتالى يتم تنفيذ الأمر أو الأوامر التى تلى كلمـــة **Then** ، أما إذا لـــم يشتمل المتغير **password** على القيمة **CS** فان الشرط يعد خاطئا

(False) وبالتالى لن تنفذ الأوامر التى تلى كلمة Then.

ولأن التفريع المشروط ينطوى دائما على شرط يحتمل الصواب أو الخطأ، أى أن البرنامج يقارن قيمتين ليتحقق من وقوع الشرط صحيحا من عدمه ، فان عملية المقارنة تتم باستخدام أحد عوامل المقارنة أو العوامل المنطقية التى سيرد شرحها بالتفصيل في الفصـــل الثامن من هذا الكتاب.

ويوضح المثال التالى (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 والذى ســبق وبيناه وبين الأمر ON...GoTo للتفريع المشروط. وفيما يلى توضيح لفكرة هذا الإجراء.

- في السطر الأول والثانى 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 .

- فى السطر الخامس يتم تنفيذ التفرع المشروط ON...GO TO على أساس أن القيمة المدخلة للمتغيرValue1 هى "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 يلى:

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>

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

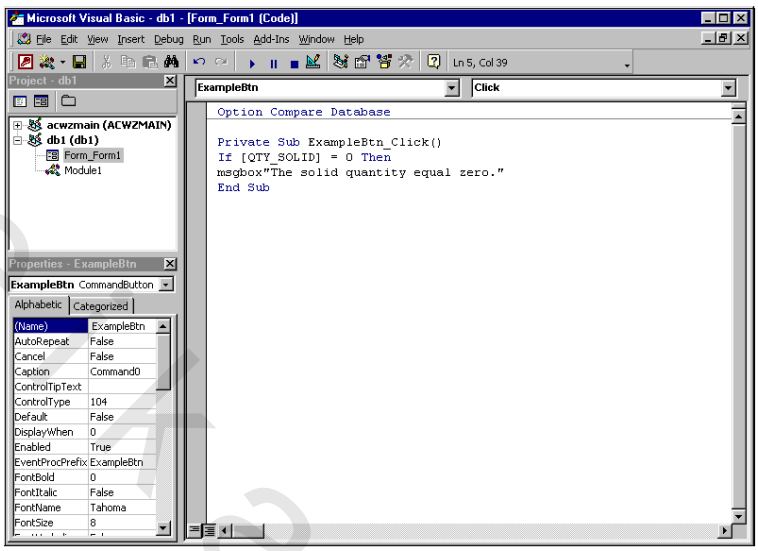
IF A=5 Then print "OK "

في هذا المثال الجملة الشرطية عبارة عن أداة شرط هى IF الشرط هـــو مقارنـــة محتويات المتغير **A** بالقيمة **5** ويرتبط جواب الشرط في البداية بكلمة **Then** ويتم تنفيـــذ 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 كلمـــة **Then** وأن التركيب ينتهى بعبارة **End if** كما فى هذا مثال (شكل ٧-٣) :



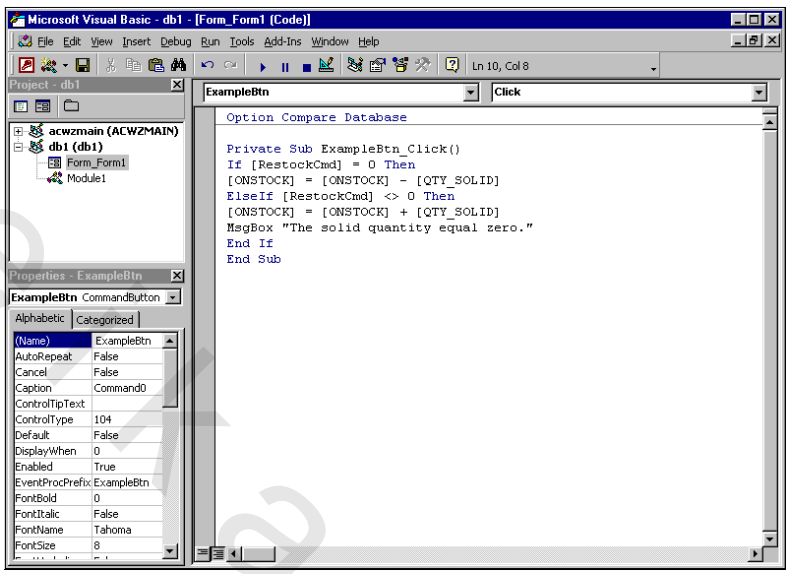
شكل ٧-٣ استخدام جملة IF

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
```

وفى هذا التركيب يتم تقيم الشرط (Condition) الوارد به ، فإذا كان صحيحا ينفذ VBA Access الأمر أو الأوامر (Commands) التى تلى كلمة Then ، وإلا فانه ينفذ الأمر أو الأوامر التى تلى كلمة ElseIf ، ويصلح هذا التركيب عندما تريد تنفيذ مجموعة أوامر إذا وقع الشرط صحيحا ومجموعة أخرى إذا وقع الشرط خاطئا.. والمثـــال التالى (شكل ٨-٣) يوضح هذه الفكرة.

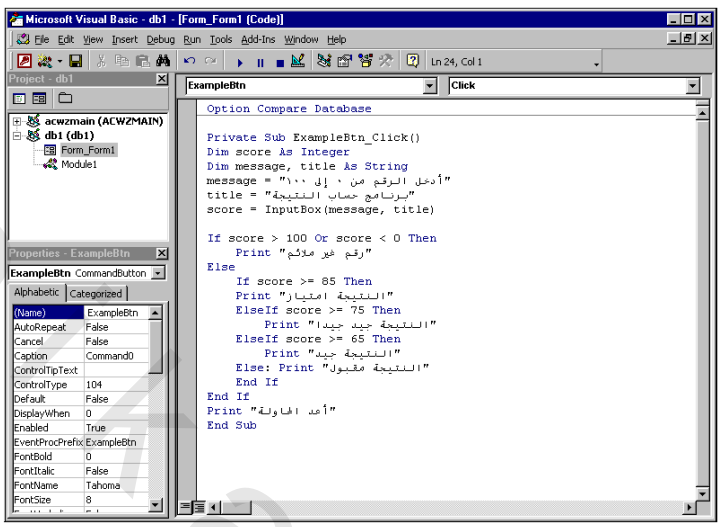


شكل ٨–٣ استخدام التركيب If...Then...Else

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

استخدام جملة IF لعمل أكثر من مقارنة

في المثال التالى (شكل ٩–٣) يقوم الإجراء بحساب معدل ويقوم باختباره فـــإذا كان خارج النطاق من ٠ إلى ١٠٠ تظهر الرسالة "رقم غير ملائم" وفى حالة إدخال معدل جديد غيره يتم مقارنته مرة أخرى وطباعة الرسالة المناسبة (امتياز أو جيد جدا أو جيد أو مقبول).



شكل ٩-٣ استخدام جملة If لعمل أكثر من مقارنة

في مثل هذه الحالات التى تتعدد فيها الاحتمالات ننصح باستخدام التركيب **Select Case** للمقارنة والذى 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
        [Casse Expressionlist1
                [Statementblock1] ]
        [Case Expressionlist2
                [Statementblock2] ]
        --------------------
        --------------------
        [Case Else
                [Statementblockn] ]
End Select
```

حيث:

VarName	اسم المتغير الذي سيتم البحث في مجموعة الحالات التالية عما يساويه، ويمكن أن يكون تعبيرا حرفيا أو رقميا
Expressionlist1	تعبير حرفي أو رقمي ، ويجب أن يتطابق نوع التعبير مع نوع التعبير المذكور في **VarName** . يمكن أن يكون تعبيرا واحدا أو أكثر من تعبير مفصولين بعلامة **Comma**
Statementblock1	الجمل التي تنفذ إذا تساوت قيمة **VarName** مع التعبير أو التعبيرات المذكورة في **Expressionlist1**
Expressionlist2	تعبير حرفي أو رقمي ، ويجب أن يتطابق نوع التعبير مع نوع التعبير المذكور في **VarName** . يمكن أن يكون تعبيرا واحدا أو أكثر من تعبير مفصولين بعلامة **Comma**
Statementblock2	الجمل التي تنفذ إذا تساوت قيمة **VarName** مع التعبير أو التعبيرات المذكورة في **Expressionlist2**
Statementblockn	الجمل التي تنفذ في حالة عدم تحقق في أى من الحالات السابقة

ونوضح أن **Expressionlist** يمكن أن تأخذ أحد الأشكال الآتية:

- قيمة أو أكثر يتم مقارنتهم مع القيمة المذكورة في **VarName** . إذا استخدمت أكثر من قيمة ، افصل بينهم بعلامة **Comma** مثل:

```
Case "!" , "?", "." , ";"
```

- مجموعة قيم تقع في مدي معين ويفصل بينها بكلمة To مثل:

```
Case 10000 To 49999.99
```

- المعامل **IS** متبوعا بعامل علائقي مثل =< ,< ,> ,<> ,= 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 ، فان الرسالة التي ستظهر له هي " يجب زيادة المبيعات". أما إذا كان يقع في المدي من ٥٠٠٠٠ إلى ١٠٠٠٠٠ فان الرسالة ستكون "مبيعات مقبولة" . وإذا كان رقم المبيعات أقل من ١٠٠٠٠ فان الرسالة ستكون "مبيعات منخفضة" . وإذا لـــم تقع أي من الحالات الثلاث المذكورة صحيحة فان الرسالة التي ستظهر ستكون " مبيعات عالية"

مثال ٢

المثال التالي يستخدم التركيب **Select Case** لتقييم تعبير حرفي ، وهي تقـــيم 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 كالآتى :

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

```
Print 5
```

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

```
Dim I AS Integer
        For I = 1 To 5
                Print I
        Next I
```

وعن هذا التركيب نوضح مايلى:

- أعلنا عن المتغير **(I)** للحصول على أرقام متغيرة من 1 إلى **5**.
- الأمر **For .. Next** يعمل على زيادة قيمة المتغير بمقدار (١) في كل مرة دون الحاجة إلى كتابة الأمر التالى 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 التى تمت لمجموعة الأوامر المطلوب تكرارها هو **(5)**.

استخدام الوظيفة Step

يتم استخدام الوظيفة **Step** مع الأمر **For .. Next** لمعرفة مقدار الزيادة التى 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 ١ 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 ١ إلى ١٠ استخدم الأمر التالى :

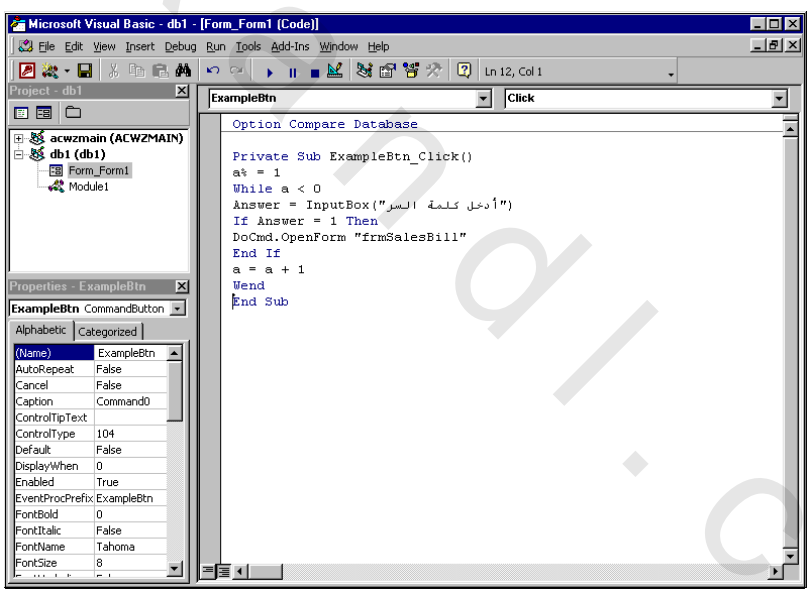
```
For I = 2 To 10 Step 2
```

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



شكل ١٠-٣ 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** الذى ذكرناه من قبل وهى تأخذ الشكل التالى :

Do While الشرط

مجموعة الأوامر

Loop

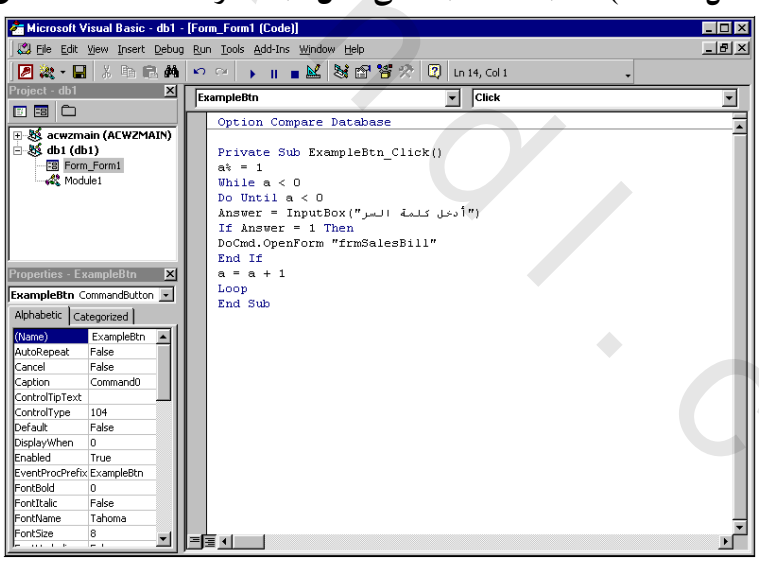
الصيغة الثانية: يتم تكرار مجموعة الأوامر حتى يتحقق الشرط. وهى تأخذ الشكل التالى:

Do Until الشرط

مجموعة الأوامر

Loop

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



شكل ١١-٣ أمر **Do Until...Loop**

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

الصيغة الثالثة: يتم التكرار طالما أنها تحقق شرط معين وهى تأخذ الشكل التالى :

Do

مجموعة الأوامر

Loop While الشرط

الصيغة الرابعة:يتم تكرار مجموعة الأوامر حتى يتحقق الشرط. وهى تأخذ الشكل التالى:

Do

مجموعة الأوامر

Loop Until الشرط