بسم الله الرحمن الرحيم

شرح للمتغيرات و الثوابت و المعاملات في الفيجول بيسك

شرح : طارق العبسي – اليمن

لمزيد من الدروس البرمجية

http://www.14soft.com/vb

http://www.14soft.net

مقدمـــــــــــة :

البيانات في أي لغة من لغات البرمجة بما فيها فيجول بيسك أما أن تكون متغيرات Variables أو ثوابت Constantsو المتغير هو مكان في الذاكرة يتم تخصيصه لك لتقوم بوضع عنوان له ويخزن المعلومة التي تريدها بداخلة كما يمكنك تغير هذه المعلومة بمعلومة أخري في نفس المكان ويظل عنوانه ثابت.
أما الثابت فمثله مثل المتغير تماما إلا انك لن تحتاج لتغير هذه المعلومة . وكما هو واضح من الاسم عبارة عن اسم يحمل قيمة ثابتة لا تتغير أثناء تنفيذ البرنامج.

ولكي هذان المثالين ليوضح لك وظيفة كل من المتغيرات والثوابت..

مثال علي المتغيرات والثوابت :
المتغيرات . إذا أردت أن تسال عن اسم العميل الذي سيدخله المستخدم فان اسم العميل قيمة متغيرة لأنك لا تعرف من هو هذا العميل الذي سيقع عليه اختيار المستخدم في هذه الحالة تستخدم متغير لتضع فيه اسم العميل انظر المثال التالي:

code:
```
HisName$=InputBox$("اكتب اسم العميل")
</TD< tr>
```

في هذا المثال سيعرض فيجول بيسك علي المستخدم مربع حوار نتيجة لتنفيذ أمر InputBox$ يطالبه فيه بكتابه اسم العميل ويقوم بحفظ اسم العميل الذي يدخله المستخدم في المتغير HisName$ ويبقي المتغير HisName$ يحمل هذا الاسم حتى يقوم المستخدم بتغيره ويتم تغير القيمة التي يحملها المتغير HisName$ بوضع قيمة أخري داخلة فيقوم الفيجول بيسك باستبدال القيمة القديمة بالقيمة الجديدة.
الثوابت . إذا كان عملك يتطلب مجموعة من العمليات الحسابية ترتبط بوحدة ثابتة مثل وحدة القياس المتر وهو يساوي مائه سنتمتر فيمكن الإعلان عن ذلك بالأمر التالي:
code:
```
Const Meter=100
</TD< tr>
```

وهذه يفيدك عندما تكون جميع حساباتك بالنسبة للوحدة سنتمتر فبدلا من قيمة المتر وكتابة الرقم (١٠٠) في كل مرة سيتم كتابه الثابت Meter في جميع التعليمات المطلوبة داخل البرنامج وهي فائدة كبيرة تجعل برنامجك سهلا وبسيطا.
فائدة أخري يمكن الحصول عليها من استخدام الثوابت ، فمثلاً في حالة تعديل كل حساباتك لتصبح منسوبة لوحدة المليمتر بدلا من السنتمتر (والمعروف أن المتر = ١٠٠٠ ملليمتر) فبدلا من إجراء هذا التعديل في جميع إجراءات برنامجك (وهو كتابة الرقم ١٠٠٠ بدلا من الرقم ١٠٠) يكفي أن تعدل الرقم ١٠٠ ليصبح ١٠٠٠ في نفس الأمر كآلاتي.
code:
```
Const Meter=1000
</TD< tr>
```

وبذلك تتم عملية التعديل مرة واحدة فقط لتعطي النتيجة المطلوبة.
و ألان وقد تعرفنا علي وظيفة كل من المتغيرات و الثوابت و الفائدة المرجوة من استخدامهم ولكن بقي لنا أن نوضح كيفية التعامل مع المتغيرات و الثوابت ليمكننا استخدامهم بصورة صحيحة وبشكل أفضل.
ولعدم التشتت سنترك الثوابت قليلا وسنتكلم عن المتغيرات بشكلا من التفصيل.

لعلك يا أخي تتسأل عن الكثير من النقاط الغامضة التي لم يكشف عنها بعد ولعلك حاولت تطبيق الدرس السابق ولم تعمل معك المتغيرات أو الثوابت بشكل صحيح فمهلا يا أخي و لا تتعجل فهناك الكثير لكي تعرفه عن المتغيرات و الثوابت و سنبدأ بأنواع المتغيرات.

أنواع المتغيرات :

يوجد في الفيجول بيسك أنواع كثيرة للمتغيرات نوضح منها ما يلي.
نوع المتغير : Integer عدد صحيح صغير نسبيا ـ حجمه ٢ Byte ـ مداه من -٣٢٧٦٨ إلي ٣٢٧٦٧
نوع المتغير : Long عدد صحيح كبر نسبيا ـ حجمه ٤ Byte ـ مداه من -٢١٤٧٤٨٣٦٤٨ إلي214783674
نوع المتغير : Single عدد حقيقي صغير نسبيا) يحتوي علي علامة عشرية عائمة (Floating Point ـ حجمه ٤ Byte ـ مداه من -3.402823E38 إلي -١,٤٠١٢٩٨E-45) و من ١,٤٠١٢٩٨ E-45 إلي ٣,٤٠٢٨٢٣E38) قيم موجبة(
نوع المتغير : Double عدد حقيقي كبير نسبيا) يحتوي علي علامة عشرية عائمة) ـ حجمه ٨ Byte ـ مداه رقم هائل
نوع المتغير : Currency عدد حقيقي كبير نسبيا (يحتوي علي علامة عشرية ثابتة) ـ حجمه ٨ Byte ـ مداه رقم هائل
نوع المتغير : Byte عدد صغير جدا أو بيانات ثنائية ـ حجمه ١ Byte ـ مداه من الصفر إلي 255
نوع المتغير : Boolean متغير منطقي يحمل قيمتان فقط صفر أو واحد . أو True or False ـ حجمه ٢ Byte
نوع المتغير : Data يحمل قيم تاريخه و وقت وهو من نفس نوع المتغير Double ـ حجمه ٨ Byte ـ مداه من التاريخ ١ يناير ١٠٠ إلي ٣١ ديسمبر ٩٩٩٩ ومن الساعة ٠٠:٠٠:٠٠ إلي 23:59:59
نوع المتغير : Object كائنات أو فئاتClasses
نوع المتغير : String سلسلة من الحروف ـ مداه من صفر إلي ٦٥٥٠٠ حرف تقريبا
نوع المتغير : Variant الوقت/التاريخ أو عدد ذو علامة عشرية عائمة أو سلسة حروف ـ حجمه ١٦ Byte ـ مداه التاريخ من ١ يناير ٠٠٠٠ إلي ٣١ ديسمبر ٩٩٩٩ وفي الأعداد مثل Double وفي الحروف مثل String

و الآن وقد تعرفنا علي أنواع المتغيرات)ارجع إلي كتاب الأستاذ تركي العسيري فيه شرح لأنواع المتغيرات بشكل من التفصيل .(
ولكن بقي لنا شئ آخر وهو كيف نعلن عن هذه المتغيرات في فيجول بيسك . ومعني الإعلان عن المتغير عبارة عن أمر يخبر فيجول بيسك باسم المتغير ونوعه ليتمكن الفيجول بيسك من حجز المساحة اللازمة من ذاكرة الحاسب لهذا المتغير وتهيئته . وقبل الإعلان عن أي متغير يجب إن نعرف شئ مهم جدا وهي..

الشروط الواجب توافرها عند اختيار اسم للمتغير ..

يجب أن يبدأ اسم المتغير بحرف أبجدي وليس رقما.
ألا يزيد عدد حروف اسم المتغير عن ٤٠ حرفا.
ويجب ألا يحتوي علي أي مسافات أو نقاط وإذا كان اسم المتغير يحتوي علي كلمتين فأنصحك باستخدام Under Score (_) للفصل بينهما ..
و يجب أيضا إلا يتضمن كلمة من الكلمات المحجوزة وهي الكلمات التي تستخدم في الأوامر و العبارات التي يستخدمها فيجول بيسك فمثلا لا يسمح باستخدام كلمة Print كاسم للمتغير فإذا احتجت لتسمية متغير مثل كلمة Print فيمكنك أن تكتب الكلمة كجزء من اسم المتغير PrintText مثلا.

فائدة (١ :)

يفضل الإعلان عن نوع المتغير لزيادة سرعة التعامل معه .المتغيرات التي لم تحدد نوعها يعمل فيجول بيسك علي الإعلان عنها تلقائيا من النوع Variant وهو أبطأ أنواع المتغيرات .

فائدة (٢ :)

يوجد نوعان من المتغير من النوع String وهي متغيرات ثابتة الطول Fixed Length و متغيرات متغيرة الطوال . Variable Length و المتغيرات الحرفية ثابتة الطوال وكما واضح من اسمها هي متغيرات محدد لها عدد الحروف في أثناء التصريح عنها ولا يمكن أن يتغير طولها وتكون بصورة التالية :

code:

```
Dim TafTaf As String * 10
```

```
</TD< tr>
```

و المتغيرات الحرفية متغيرة الطول تستخدم في حالة عدم معرفتك بطول المتغير المار إليها ومن عيوبها إنها تلتهم حجم كبير جدا من الذاكرة.

كيفية الإعلان عن المتغيرات :

عند استخدام المتغير في فيجول بيسك فأن فيجول بيسك تتعرف علي المتغير بمجرد استخدامه في الكود وهذه الطريقة مريحة لأنك لان تحتاج إلي تعريف كل متغير قبل استخدامه ألا إنها يعاب عليها شئ خطير جدا وهو انك إذا أخطأت في كتابة اسم المتغير فأن فيجول بيسك سيعتبره متغيرا جديدا فمثلا إذا أنشأنا متغيرا مثل هذا ..

code:

```
Value = 10                '   ١٠ = قيمةValue  هنا أعطينا
المتغير
Text1.Text = Valu  'اسم المتغير  في كتابة أخطأنا  و هنا
</TD< tr>
```

و لحل هذه المشكلة الإعلان عن المتغير بأمر Dim مثلا) . سنتكلم عن أوامر الإعلان عن المتغيرات لاحقا)

code:

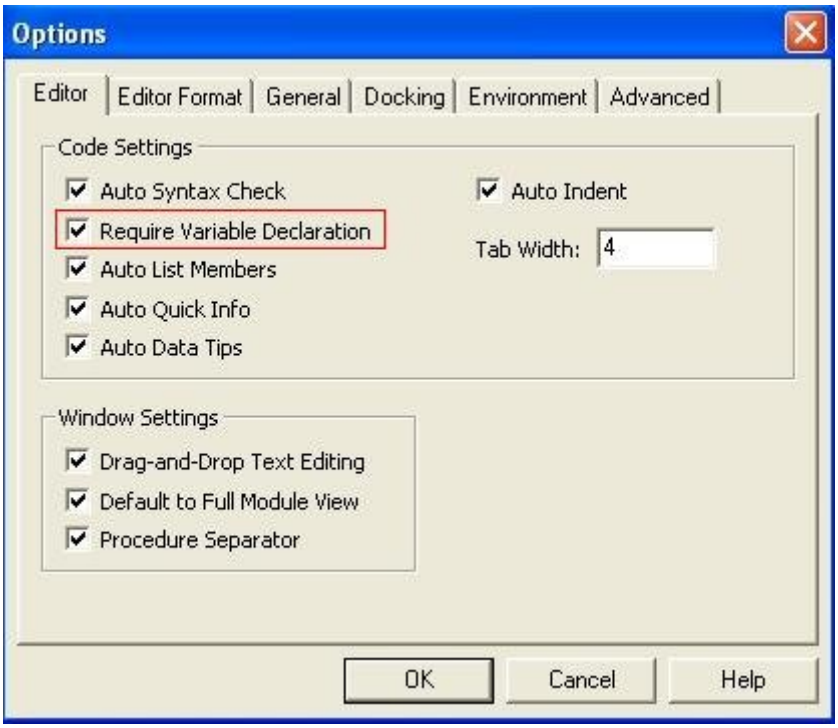
```
Dim Value As Integer
</TD< tr>
```

ولجعل فيجول بيسك لا يستخدم متغيرا إلا قبل الإعلان عنه أو بمعني اصح يجبرك للإعلان عن كل متغير تستخدمه بكتابة هذه العبارة في قسم التصاريح العامة . General Declaration بإظهار رسالة خطأ..

code:

```
Option Explicit
</TD< tr>
```

ولجعل هذه العبارة تكتب تلقائيا . من قائمة Tools اختر أمر Options وضع علامة صح في صندوق الاختيار أمام عبارة Require Variable Declarations من صفحة Editor أي طلب الإعلان عن أي متغير قبل استخدامه..



تعلمنا ألان فائدة الإعلان عن المتغيرات و المشاكل التي تترتب عليها إذا لم نعلن عن أي متغير نضيفه في برنامجنا ولكن السؤال هنا الذي يطرح نفسه هو كيف نعلن عن المتغيرات وما هي أوامر الإعلان وهذا يذكرني بمن تعلم القيادة ولكنه لا يعلم شئ عن قواعد المرور التي سترشده إلي الطريق الصحيح .
و لكي نتحكم في المتغيرات بشيئا من الفاعلية أكثر يجب نتعرف علي أوامر الإعلان حتى يمكننا الإعلان عن المتغير بشكل سليم واستغلال الذاكرة بشكل أفضل..

أوامر الإعلان عن المتغيرات :

أمر الإعلان : Dim يستخدم لتعريف متغير ديناميكي Dynamic Variables ضمن الأجراء ويكون مجال رؤية هذا المتغير داخل الأجراء فقط وعمرة من عمر الأجراء أي أن عندما ينتهي الأجراء ينتهي معه مفعول المتغير المحلي ويصبح لا قيمة له أو صفر ولهذا سمي أمر الإعلان هذا بديناميكية لأنه يوفر في الذاكرة بشكل جيد.
أمر الإعلان : Redim تستخدم لتعريف مصفوفة ديناميكية Dynamic Array غير معرفة لعدد العناصر أي يمكنك تغير أبعادها أثناء عمل البرنامج ويعمل أمر الإعلان Redim عندما ترغب في تحديد حجم المصفوفة وتظهر الفائدة من أمر الإعلان Redim في استغلال الجزء المطلوب فقط من الذاكرة دون زيادة . وأمر الإعلان هذا يجعلنا ندخل إلي عالم آخر وهو عالم المصفوفات وإذا تكلمنا عن المصفوفات في هذا الجزء الصغير فبذلك نكون قد لا نوفيها حق قدرها . ولكن من يعلم قد يأتي يوم الأيام ونتكلم عن المصفوفات إنشاء الله.

فائدة (٣ :)

أن أمر الإعلان Redim لا يستخدم ألا داخل أجراء فقط أي لا يستخدم في قسم الإعلانات مثل الأمر Dim

أمر الإعلان : Static يستخدم لتعريف متغير ستاتيكي Static Variables ضمن الأجراء ويكون مجال رؤية هذا المتغير داخل الأجراء فقط وعمرة من عمر الوحدة التي بداخلها الأجراء سواء كانت الوحدة هذه Form أو Module أو Class أي أن عندما ينتهي الأجراء يظل المتغير ساكن في الذاكرة وكذلك قيمته موجودة ولكنك لا يمكنك الوصول إلية ألا من داخل الأجراء التابع له.

نصيحة (١ :)

حاول أن لا تكثر من استخدام أمر الإعلان Static ألا في أضيق الحدود لأنه يستغل الذاكرة طوال فترة عمل البرنامج .
أمر الإعلان : Private يستخدم لتعريف متغير عام General Variables ولكنه يكون علي مستوي الوحدة سواء كانت Form أو ملف برمجة BAS أو فئة Class وبالطبع عمرة يكون من عمر الوحدة.
أمر الإعلان : Public يستخدم لتعريف متغير عام General Variables ولكنه يكون علي مستوي المشروع ككل ويمكن الوصول إليه من جميع الوحدات الموجودة في المشروع وعمرة من عمر البرنامج ككل ويظل في الذاكرة حتى ينتهي البرنامج.
أمر الإعلان : Global أمر الإعلان أو الكلمة المحجوزة Global كانت تستخدم في الإصدارات القديمة للفيجول بيسك ومازالت حتى الآن تستخدم وهي تؤدي نفس وظيفة أمر الإعلان Public ولكنك لن تستطيع التصريح عنها ألا في الوحدات النمطية فقط.

الإعلان بإضافة رمز مميز :

تستخدم هذه الطريقة في نوع أي متغير وذلك بإضافة حرف معين إلي اسم المتغير و هذه الطريقة تسهل عليك معرفة نوع المتغير المستخدم مع اسم المتغير و الجدول التالي يبين شكل هذه الأحرف و النوع المقابل لها..

نوع المتغير : Integer الرمز المستخدم" % "
نوع المتغير : Long الرمز المستخدم" & "
نوع المتغير : Single الرمز المستخدم" ! "
نوع المتغير : Currency الرمز المستخدم" # "
نوع المتغير : Double الرمز المستخدم" @ "
نوع المتغير : String الرمز المستخدم" $ "

فمثلا الأمر..

code:

```
MyName$="TafTaf"
</TD< tr>
```

يعلن عن متغير من نوع) String سلسلة من الحروف (

الإعلان باستخدام الوظيفة : AS

وهنا تفيد الوظيفة AS في تميز نوع المتغير الذي يأتي بعد الوظيفة AS مع أحد الأوامر Redim. Dim. Static. Global Private. Public حيث يتم كتابة الأمر ثم اسم المتغير ثم كتابة الوظيفة AS ثم كتابة نوع المتغير . انظر المثال التالي.

code:

```
Dim TafTaf  AS String
</TD< tr>
```

فائدة (٤ :)

أوامر الإعلان السابق ذكرها هي تعتبر ضمن الكلمات المحجوزة مثل الجملة Option Explicit أو أمر Print والكلمات المحجوزة هي الكلمات التي يحتفظ بها الفيجول بيسك لنفسه ولا يمكنك استخدامها كاسم للمتغير أو للثابت .

الإعلان باستخدام أمر تعريف دالة :

في هذه الطريقة يتم استخدام أحد الأوامر التالية:
(CCur , CLng , CDbl , CInt, CStr , CSng, Cvar)
للأنواع الآتية علي التوالي.
Currency , Long , Double , Integer , String , Single , Variant
وعند كتابة أي حرف بعد أي من هذه الأوامر تتحول كل المتغيرات التي تبدأ بهذه الأحرف إلي نفس النوع المعلن عنه في الأوامر المستخدم ، وفي المثال التالي نستخدم الأمر CInt للإعلان عن جميع المتغيرات الموجودة في البرامج والتي تبدأ بحرف A علي إنها من النوع. Integer

code:

```
CInt A
</TD< tr>
```

أما في المثال التالي فيتم الإعلان عن التغيرات الموجودة في البرامج و التي تبدأ بأحد الأحرف التالية B أو C أو D علي أنها من النوع. String

code:

```
CStr B-D
</TD< tr>
```

لاحظ أنه لا ينبغي كتابة أكثر من حرفين متصلين بدون العلامة (-) بعد الأمر .
للإعلان عن أي متغير يبدأ بالحرف (A) أو بأحد الحروف من (D) إلي (F) أو من (X) إلي (Z) علي أنه من النوع Double اكتب الأمر بالصورة التالية:

code:

```
CDbl A, D-F , X-Z
</TD< tr>
```

علاقة المصفوفات بالمتغيرات :

المصفوفات هي في حد ذاتها عبارة عن سلسلة من المتغيرات لها نفس الاسم ولكن يكون لها رقم وهنا تختلف عن المتغيرات بوجود هذا الرقم الذي يوفر لك الوقت و الجهد .
إليك توضيح أكثر .. إذا أردت تصميم برنامج لشئون الموظفين بشركتك بدون استخدام المصفوفات وبفرض أن عدد الموظفين في شركتك ١٠٠ موظف فان الحل باستخدام المتغيرات العادية يتطلب الإعلان عن ١٠٠ متغير ثم تكرار الأوامر التي تتعامل مع الموظفين المائة ولا شك أن هذه مسألة شاقة ومطولة.

أما الحل الامثل في هذه الحالة فهو استخدام مصفوفة تتكون من ١٠٠ عنصر و الصيغة التي تحقق هذا الغرض لهذه المصفوفة كما يلي:

code:

```
Private Names(99)  As String
```

كان درسنا السابق عن أنواع المتغيرات وكيفية الإعلان عنها ولكن بقي لنا شئ مهم جدا لنعرفه عن المتغيرات وهو مدي استخدام المتغير وعمرةLifetime and Scope of Variable

مدي استخدام المتغير وعمرة : Lifetime and Scope of Variable

ويقصد بمدي استخدام المتغير Scope Of Variable والإجراءات و الوحدات النمطية التي ستتأثر به أي الأماكن التي أن يستخدم فيها هذا المتغير داخل البرنامج أما عمر المتغير Lifetime Of Variable فيقصد به المدة التي سيبقي المتغير خلالها محتفظا بقيمته الحالية داخل الذاكرة دون أن يفقدها وتنقسم المتغيرات من حيث مدة بقائها في الذاكرة ومداها إلي متغيرات عامة ومتغيرات علي مستوي الوحدة النمطية ومتغيرات علي مستوي الأجراء وفيما يلي نوضح كل نوع من هذه الأنواع الثلاثة والأمر الذي يستخدم للإعلان عنه ..

.1المتغيرات العامة ..

هي المتغيرات التي يمكنك استخدامها من أي مكان داخل البرنامج أو التطبيق وتبقي في الذاكرة الحاسب طوال فترة عمل البرنامج فإذا انتهي 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

code:

```
Public ABC AS Integer
</TD< tr>
```

.2المتغيرات علي مستوي الوحدة النمطية ..

بإمكانك الإعلان عن متغير وتقيده علي مستوي 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 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 TafTaf لكي يستخدم فقط مع الوحدة النمطية التي يوجد بها.

code:

```
Private TafTaf As String
</TD< tr>
```

وهذا الأمر يمكن إدخاله من خلال الأجراء لإدخال الأمر علي مستوي الوحدة النمطية استخدمه بنفس الطريقة التي تستخدمها للإعلان عن المتغير العام مع فارق واحد وهو استخدام أمر Private بدلا من أمر Public .

.3متغيرات علي مستوي الأجراء ..

يقتصر مدي هذه المتغيرات علي الأجراء الموجودة به فقط ولا يمكن استخدامه في أي مكان غيرة وهي بهذا تعتبر اقل المتغيرات مدي من حيث عمرها فهي تبقي موجودة بالذاكرة حتى بعد أن ينتهي الأجراء الذي أعلن فيه عنها وبهذا يتضح أن الفرق بين هذه المتغيرات و المتغيرات العامة أو المتغيرات علي مستوي الوحدة النمطية في مداها فقط حيث لا يتعدى مداها الأجراء الذي أعلن عنها فيه . يستخدم لهذا الغرض الأمر Static ويتضح ذلك من المثالين التالين في المثال الأول يظهر المتغير S1 محتفظا بقيمته فترة تنفيذ الأجراء Load Form وبمجرد الخروج من الأجراء ستكون قيمته تساوي صفر بينما في حالة استخدام المثال الثاني فإن المتغير سيحتفظ بقيمته بعد تنفيذ نفس الأجراء السابق.
المثال الأول :

code:

```
Private Sub Form_Load (Cancel As Integer)
Dim S1 As Integer
S1 = 5
End Sub
</TD< tr>
```

المثال الثاني:

code:

```
Private Sub Form_Load (Cancel As Integer)
Static S1 As Integer
S1 = 5
End Sub
</TD< tr>
```

فائدة (٥ :)

يوفر عليك مدي 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 الأجراء فقط استخدم أمر Static ليبقي مداه داخل الأجراء فقط .

مستويات الإعلان عن المتغيرات) : هذا الجزء منقول عن موضوع العقل الصناعي : الثوابت و المتغيرات للأخ (BSC.

في القسم العام لملفات البرمجة Dim عام ، Private عام ، Public شامل ، Global شامل
في القسم العام للنوافذ Dim عام ، Private عام ، Public عام
داخل إجراءات الملفات Dim ساكن
داخل إجراءات النوافذ Dim محلي ، Static ساكن

المصطلحات :

-شامل : عام لجميع نوافذ البرنامج وملفاته
-2عام : عام لجميع إجراءات النافذة أو الملف الذي تم الإعلان داخلها (الشامل أعم)
-3المحلي والساكن : كلا منهما خاص بالإجراء الذي تم الإعلان داخله

الفرق بين المحلي والساكن :

-المتغير المحلي يفقد قيمته عند الخروج النهائي من الإجراء
-2الساكن يحتفظ بقيمته طوال مدة تشغيل البرنامج

كما ذكرنا من قبل أن الثوابت مثل المتغيرات تماما ألا أنها وكما واضح من اسمها ثابتة أي لا يمكن تغير محتوها أو القيمة المضافة إليها إثناء تنفيذ البرنامج عكس المتغير الذي يمكننا تغير قيمته كما نريد تبعا للمدخلات ومع ذلك فالثوابت تتشابه مع المتغيرات في أمرين هما اسم الثابت ومداه .

تسمية الثابت :

يخضع اسم الثابت لنفس الشروط التي شرحناها عند اختيار اسم المتغير وهي ألا يزيد عدد حروفه عن 40حرفا، وأن يبدأ بحرف هجائي وألا يستخدم إحدى الكلمات المحجوزة للفيجول بيسك.

مدي الثوابت :

تتبع الثوابت نفس القواعد التي تحدد مدي المتغيرات حيث يحدد مدي الثابت بالمكان الذي تعلن فيه عن هذا الثابت . وتوضيح ذلك كما يلي.

ثوابت عامة : إذا أردت أن يكون الثابت عاما أي يمكن استخدامه من أي مكان في البرنامج فيجب أن تعلن عنه في الوحدة النمطية بشرط أن يسبق الإعلان عنه كلمة Public هكذا.

code:

```
Public Const My_Name As String = "TafTaf"
</TD< tr>
```

ثوابت علي مستوي الوحدة النمطية : لكي تستخدم الثابت في وحدة نمطية فقط يجب أن تعلن عنه في قسم الإعلانات في هذه الوحدة النمطية تسبقه كلمه Private هكذا.

code:

```
Private Const My_Age As Integer = 25
</TD< tr>
```

ثوابت علي مستوي الأجراء : : لكي تستخدم الثابت مؤقتا داخل إجراء معين أعلن عن الثابت داخل هذا الإجراء بنفس الطريقة السابقة .
و نكتشف من الكلام السابق أن الثوابت مشابه إلي حد كبير للمتغيرات في طريقة الإعلان عنها بأمر الإعلان Const أو حتى الإعلان عن نوع الثابت لزيادة سرعة التعامل معه وأيضا مجال رؤية الثابت وعمرة.
وللثوابت أنواع منها ثوابت عددية وثوابت حرفيه String وللإعلان عن ثابت حرفي يجب وضعة بين علامتي تنصيص. " "

الإعلان عن الثوابت) : هذا الجزء منقول عن موضوع العقل الصناعي : الثوابت و المتغيرات للأخ (BSC

يتم الإعلان عن الثوابت باستخدام (Const)أو(Public Const)

مستوى الإعلان :

1-إذا تم الإعلان داخل إجراء يعتبر الثابت المعلن عنه محليا Const a=5

2-وإذا تم الإعلان في القسم العام للنافذة يصبح عاما لجميع إجراءات تلك النافذة Const a=5

-3وإذا تم الإعلان في القسم العام للملف يصبح عاما لجميع إجراءات ذلك الملف Const a=5

-4أما إذا تم الإعلان في القسم العام للملف - كما سبق - لكن سبقت كلمة Const بكلمة Public فسيصبح الثابت شاملا لجميع ملفات البرنامج ونوافذه

يتم الإعلان عن الثابت الشامل هكذا(Public Const a=5)

الثوابت و : API

كلنا نعرف مكتبات الربط الديناميكي (Dynamic Link Libraries (DLL و الفائدة الكبيرة التي نستفيد منها في برامجنا من التوابع (الدوال) و التي يطلق عليها واجهة برمجة التطبيقات Application Programming Interface (API) .
ولكن عند استخدام هذه التوابع يوجد بعض الثوابت التي يجب تمريرها لهذه التوابع لتقوم بعملها المطلوب منها علي أكمل وجهه لان الكل يعرف القاعدة الذهبية تقول أن لكل دالة من دوال API لها ثوابت خاصة بها . انظر المثالي التالي:
يصرح بهذه الدالة في قسم التصاريح العامة:

code:

```
Private Declare Function MessageBox Lib "user32" _
Alias "MessageBoxA" (ByVal hwnd As Long, ByVal _
lpText As String, ByVal lpCaption As String, ByVal _
wType As Long) As Long
Const MB_OK = 0
</TD< tr>
```

اكتب هذا الكود في حدث Click للـ: CommandButton

code:

```
Private Sub Command1_Click()
MessageBox Me.hwnd, "Hallow", "Example", MB_OK
End Sub
</TD< tr>
```

الكود السابق هو كود بسيط لعرض رسالة عن طريق توابع الـ API .
في البداية عملنا علي التصريح بالدالة في قسم التصاريح العامة بأمر الإعلان Private وبعد ذلك صرحنا عن ثابت وهو ثابت معرف مسبقا و موجود في الفيجول بيسك واسم الثابت MB_Ok وبالطبع لك مطلق الحرية في تسمية الثابت كما تعلمنا سابقا .
والخطوة الثانية العمل علي تمرير الثابت ليؤدي وظيفته المطلوبة منه في حدث Click للـ CommandButtonأو في أي حدث تريده . ووظيفة هذا الثابت هي إظهار الزر موافق vbOkOnly للرسالة .

فائدة (٦ :)

من الممكن الاستغناء عن أمر الإعلان Private إذا كان الإعلان علي مستوي الوحدة أو الأجراء

code:

```
Private Const MB_OK = 0
Const MB_OK = 0
</TD< tr>
```

يؤديان نفس الوظيفة ..

و ألان وقد انتهينا من شرح كل من المتغيرات و الثوابت واكتشفنا مدي أهميتهم لنا كمبرمجين لتوفيرهم لنا الوقت والجهد وبعد انتهاءنا من المتغيرات والثوابت سنتكلم عن موضوع لا يقل أهمية عنهما وهو عن المعاملات أو عوامل التشغيل. Operators
وعوامل التشغيل أو المعاملات Operators لها فائدة عظيمة في توفيرها لنا الوقت و الجهد مثل المتغيرات و الثوابت فهي تجعلنا نختزل الكود ونستطيع من خلالها المقارنة بين قيمتين أو اكثر أو حتى

مزج قيم مع بعضها فالمبرمج الجيد الذي يستطيع استغلال هذا و الذي يجعل الكود الذي يكتبه صغير جدا.