# الفصل الأول

# المتغيرات Variables

# الفصل الأول

## المتغيرات Variables

المتغيرات هي أحد أهم الركائز الأساسية في جميع لغات البرمجة ، وعليه فإن بدون فهم المتغيرات وإستيعابها جيداً لا تتوقع أن تصبح مبرمجاً في يوم من الأيام ... ويمكن تعريف المتغير على أنه عبارة عن مخزن يقوم المبرمج بحجزه في الذاكرة لتخزين بعض القيم أو البيانات به ثم إستدعاء هذه البيانات مرة أخرى والإستفادة منها أثناء سير البرنامج .

### أنواع المتغيرات :

عندما تريد أن تقوم بحجز متغير ( مخزن ما في الذاكرة ) يجب أن يكون هذا المتغير مناسب لنوعية البيانات التي ستضعها داخله ، فعلى سبيل المثال إذا كانت القيم أو البيانات التي ستوضع داخل المتغير عبارة عن بيانات نصية فيجب أن تقوم بحجز متغير من النوع النصي ( String ) أما إذا كانت هذه البيانات عبارة عن أرقام أو بيانات رقمية فيجب أن تقوم بحجز متغير من النوع الرقمي وهكذا لا بد أن يتوافق نوع المتغير مع نوع البيانات كما بالشكل التالي :

وتوفر لنا لغة الفيجوال بيسك العديد والعديد من أنواع المتغيرات التي يمكن أن نحتاج إليها أثناء عمل أي برنامج ، ويمكنك التعرف على أشهر أنواع هذه المتغيرات وأكثرها استخداما من خلال الجدول التالي .

| نوع المتغير | البيانات التي يقوم المتغير بتخزينها | المساحة التي يستغلها المتغير في الذاكرة |
|---|---|---|
| String | بيانات حرفية | 1 بايت لكل حرف |
| Byte | رقم صحيح | 1 بايت |
| Integer | رقم صحيح | 2 بايت |
| Long | رقم صحيح | 4 بايت |
| Single | رقم عشرى | 4 بايت |
| Double | رقم عشرى | 8 بايت |
| Currency | عملة | 8 بايت |
| Boolean | قيم منطقية False أو True | 2 بايت |
| Date | التاريخ والوقت | 8 بايت |
| Object | صور أو بيانات البرامج المدمجة مع VB | 4 بايت |
| Variant | جميع أنواع البيانات | 16 بايت |

ملاحظة هامة :

قد يرى كثير من المبتدئين أنه إذا كان المتغير Variant يمكنه التعامل مع جميع أنواع البيانات فليس هناك ما يدعو إلى الاهتمام بباقي أنواع المتغيرات أو ضرورة استخدام أي منها ! وهذا من الأخطاء الشائعة التي يمكن أن يقع فيها الكثير عند بداية التعامل مع المتغيرات ولكى تتعرف على حجم هذا الخطأ والمشاكل التي يمكن أن تنتج عنه يكفى أن تعيد قراءة الجدول السابق بشيء من التدقيق وبالتحديد الجزء الخاص بالمساحة التي يشغلها كل متغير داخل الذاكرة فستجد أن أكبر مساحة يتم إستغلالها وشغلها من الذاكرة هي تلك المساحة الخاصة بالمتغير Variant وبالتالى فعند استخدام هذا النوع من المتغيرات فقط فإن ذلك سيتسبب في زيادة حجم المساحة المستغلة والمستقطعة من الذاكرة مما يؤثر على سرعة البرنامج وأدائه بصفة عامة خاصةً إذا البرنامج يحتوى على عدد كبير من المتغيرات .

تحديد أسماء المتغيرات :

عندما تريد الإعلان عن متغير داخل الذاكرة لا بد من إعطاء هذا المتغير أسم ما حتى تستطيع بعد ذلك التعامل معه عن طريق هذا الأسم مثل إعطاء قيمة له له أو إستدعاء القيمة المخزنة بداخله وتخضع عملية إعطاء أسم للمتغير إلى عدد من الشروط نستعرضها فيما يلى :

1- أن يبدأ أسم المتغير بحرف هجائى وليس رقم .

2- ألا يحتوى أسم المتغير على مسافات أو علامات خاصة مثل ( @ أو # أو ٪ أو & أو ..... )

3- ألا يكون أسم المتغير من الكلمات المحجوزة باللغة مثل ( If أو End ) .

4- ألا يزيد أسم المتغير عن 255 حرف .

## طريقة الإعلان عن المتغيرات :

عندما تريد استخدام أو حجز متغير ما داخل الذاكرة عليك أن تقوم أولا بالإعلان عن هذا المتغير عن طريق كلمة Dim المحجوزة داخل اللغة ويكون الإعلان عن المتغير كما يلى :

```
Dim VariableName As DataType
```

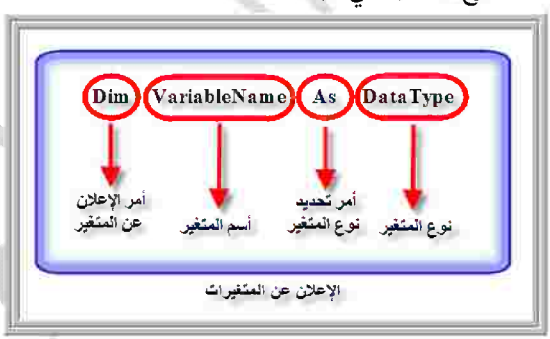
حيث تشير المعاملات السابقة إلى ما يلى :

**Dim** : أمر الإعلان عن المتغير .

**VariableName** : تحديد أسم المتغير الذي نريد الإعلان عنه .

**As** : أمر تحديد نوع المتغير .

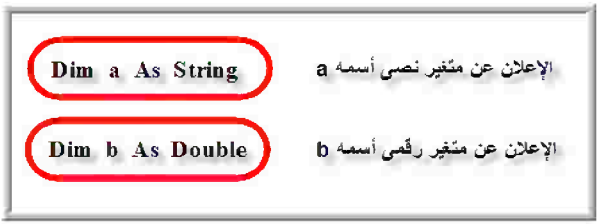
**DataType** : نوع المتغير التي نريد استخدامه .




فعلى سبيل المثال يمكنك أن تقوم بالإعلان عن أحد المتغيرات كما يلى :

```
Dim a As String
```

في الكود السابق قمنا بالإعلان عن متغير أسمه a من النوع String .




لاحظ ... أنه يمكنك أن تقوم بالإعلان عن أكثر من متغير في سطر واحد كما يلي :

```
Dim a, b, c As String
```

في الكود السابق قمنا بالإعلان عن ثلاثة متغيرات بالاسم ( a و b و c ) من النوع String .

ملاحظة هامة :

عند الإعلان عن متغير دون تحديد نوعه ستقوم الفيجوال بيسك بالتعامل مع هذا المتغير على أنه من النوع Variant وهو النوع الافتراضي داخل اللغة كما بالمثال التالي :

```
Dim a
```

في الكود السابق قمنا بالإعلان عن متغير بالاسم a ولم نقو بتحديد نوع هذا المتغير لذلك ستم معاملة هذا المتغير من قبل اللغة على أنه متغير Variant .

**تمرير قيمة للمتغير :**

يمكنك أن تقوم بوضع قيم للمتغيرات التي تقوم بالإعلان عنها عن طريق كلمة Let كما بالشكل التالي :

```
Let أسم المتغير = القيمة
```

هذا بالطبع مع ملاحظة أن تكون القيمة التي ستوضع داخل المتغير تتناسب مع نوع المتغير كما يتضح لك من الشكل التالي :

```
Dim a As String
Dim b As Double

Let a = "محمد"
Let b = 5
```

لاحظ ... أننا عند إرسال قيمة إلى متغير نصى لا بد أن تكون هذه القيمة بين علامتين " " كما بالشكل السابق .

لاحظ أيضاً ... أنه يمكنك أن تقوم بوضع قيمة داخل المتغير بصورة مباشرة دون استخدام كلمة Let كما بالشكل التالي :

```
Dim a As String
Dim b As Double

a = "محمد"
b = 5
```

**أماكن الإعلان عن المتغيرات :**

1- الإعلان عن المتغير داخل أحد الإجراءات .

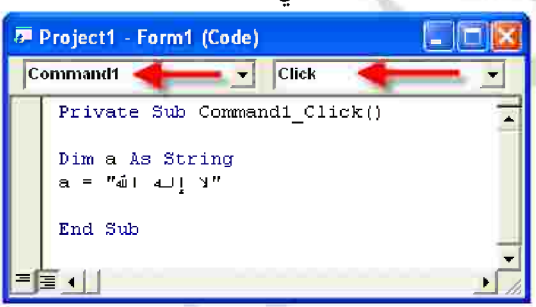
2- الإعلان عن المتغير داخل منطقة التعريفات العامة للفورم General Declarations .

3- الإعلان عن المتغير داخل موديول Module .

وفيما يلى سنقوم باستعراض هذه الأماكن بشيء من التفصيل والتعرف على الفروق بين كل منها .

أولاً : الإعلان عن المتغير داخل أحد الإجراءات :

عند الإعلان عن متغير ما داخل أحد الإجراءات فسيصبح هذا المتغير خاص بهذا الإجراء فقط يبدأ مع بداية الإجراء وينتهى بنهايته وبمعنى أخر لا يمكن استخدام هذا المتغير خارج هذا الإجراء كما بالمثال التالي :




في المثال السابق قمنا بالإعلان عن متغير بالاسم a داخل الحدث Click للمفتاح CommandButton .

ثانياً : الإعلان عن المتغير داخل منطقة التعريفات العامة :

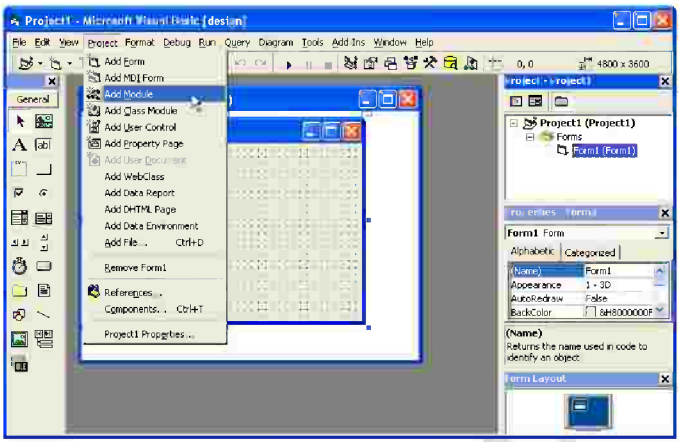
عند الإعلان عن المتغير داخل منطقة التعريفات العامة للفورم General Declarations سيصح هذا المتغير معروفاً على مستوى الفورم كلها وبالتالى يمكن استخدامه داخل أي مكان أو إجراء بالفورم كما بالمثال التالي :


في المثال السابق قمنا بالإعلان عن متغير بالإسم a داخل منطقة التعريفات العامة للفورم General Declarations وقمنا بعد ذلك نفس المتغير من داخل أكثر من مفتاح CommandButton .

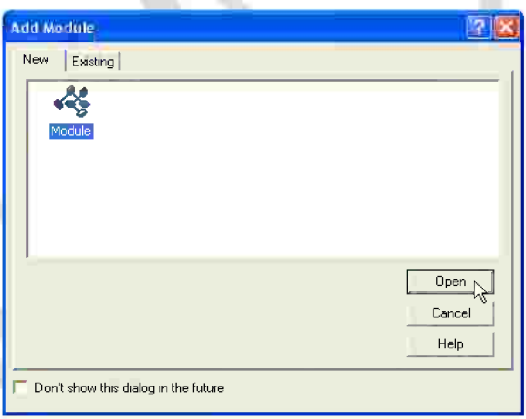
ثالثاً : الإعلان عن المتغير داخل موديول **Module** :

إذا كان المشروع أو البرنامج الذي تعمل به يحتوى على أكثر من Form وتريد الإعلان عن متغير ما يتم استخدامه داخل أي Form منها ففى هذه الحالة عليك أن تقوم بالإعلان عن هذا المتغير داخل موديول Module وقبل أن نتعرف على طريقة القيام بذلك تعالى أولاً نتعرف على كيفية إنشاء موديول Module من خلال الخطوات التالية :

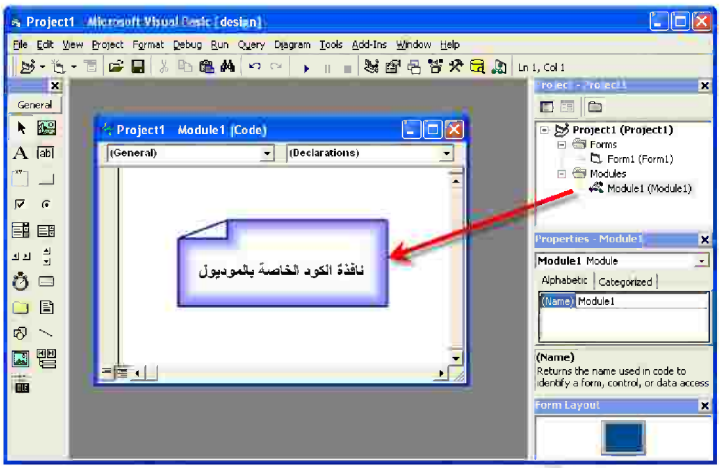
قم باختيار Add Module من قائمة Project كما بالشكل التالي :



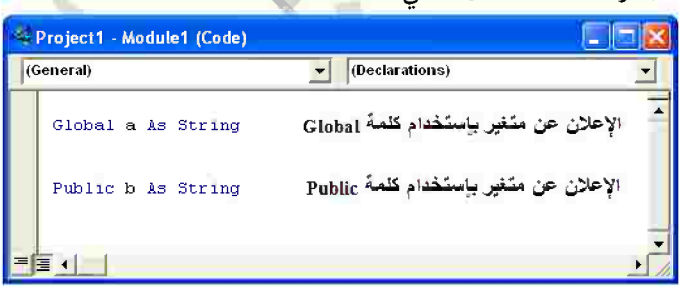

لتظهر النافذة التالية :




من النافذة السابقة قم بـالنقر فـوق زر Open ليـتم إضـافة موديـول داخـل نافـذة مستكشف المشروع Project Explorer كما بالشكل التالي :




يمكنك الآن أن تقوم بالإعلان عن المتغير داخل نافذة الكود الخاصة بالموديول والتي تظهر لك بالشكل السابق مع مع ملاحظة أنه عند الإعلان عن المتغير داخل موديول سيتم استخدام كلمة Global أو كلمة Public بدلاً من كلمة Dim التي كنا نقوم بإستخدامها وذلك كما بالشكل التالي :




## استخدام الرموز في الإعلان عن المتغيرات :

الأسلوب السابق هو الأسلوب الشائع والمتعارف عليه عند الإعلان عن المتغيرات ولكن توجد بعض المتغيرات يمكن الإعلان عنها واستخدامها بطريقة أخرى مختلفة عن هذه

الطريقة التقليدية ويتم ذلك من خلال بعض الرموز الخاصة التي يمكن التعرف عليها من خلال الجدول التالي :

| نوع المتغير | الرمز الخاص بالمتغير |
|---|---|
| String | $ |
| Integer | % |
| Long | & |
| Single | ! |
| Double | # |
| Currency | @ |

يمكنك الآن أن تتجاهل كلمة Dim وتقوم بالإعلان عن المتغير واستخدامه كذلك بصورة مباشرة عن الرمز الخاص به كما يلى :

```
a$ = "لا إله إلا الله محمد رسول الله"
```

في الكود السابق قمنا بالإعلان عن متغير باسم a ومن النوع String وقمنا بوضع قيمة لهذا المتغير ( **لا إله إلا الله محمد رسول الله** ) .

مثال على استخدام المتغيرات :

سنقوم في هذا الجزء بتطبيق مثال عملى بسيط على استخدام المتغيرات داخل الفيجوال بيسك وهذا المثال عبارة آلة حاسبة تقوم ببعض العمليات الحسابية الأساسية كما بالشكل التالي :


إذا كنت قرأت الجزء الأول من هذه السلسلة ( فيجوال بيسك المهام الأساسية ) ستجد أننا قمنا بتناول نفس المثال ولكن دون استخدام المتغيرات فعلى سبيل المثال لإجراء عملية الجمع قمنا **باستخدام** الكود التالي :

```
Text3.Text = Val(Text1.Text) + Val(Text2.Text)
```

أما لإجراء عملية الطرح فقد قمنا **باستخدام** الكود التالي :

```
Text3.Text = Val(Text1.Text) - Val(Text2.Text)
```

ولإجراء عملية الضرب قمنا **باستخدام** الكود التالي :

```
Text3.Text = Val(Text1.Text) * Val(Text2.Text)
```

وكذلك الكود الخاص بإجراء عملية القسمة كان كما يلى :

```
Text3.Text = Val(Text1.Text) / Val(Text2.Text)
```

كما تلاحظ أننا كنا نقوم **باستخدام** دالة Val لكى تتعامل اللغة مع محتويات TextBox على أنها أرقام وتقوم بإجراء العمليات الحسابية عليها بطريقة صحيحة ، فتعالى معى الآن لنقوم بتطبيق نفس المثال **باستخدام** المتغيرات ...

قم بإدخال الكود التالي داخل الحدث Click لمفتاح الجمع .

```
Dim a, b As Doublea = Text1.Text
```

```
b = Text2.Text
Text3.Text = a + b
```

**شرح الكود :**

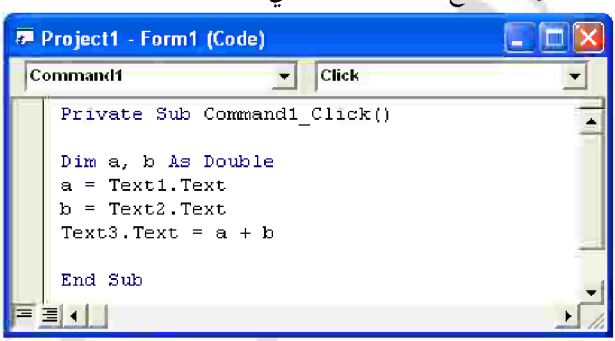
في السطر الأول قمنا بتعريف متغيرين من النوع Double .

في السطر الثاني قمنا بجعل قيمة المتغير a هي محتويات Text1 .

في السطر الثالث قمنا بجعل قيمة المتغير b هي محتويات Text2 .

في السطر الرابع قمنا بجمع المتغير a مع المتغير b ووضع الناتج في Text3 .

لاحظ أن نافذة الكود ستصبح كما بالشكل التالي :




لاحظ ... أننا بذلك وبعد استخدام المتغيرات الرقمية لم نعد بحاجة إلى استخدام دالة Val داخل الكود .

قم بعد ذلك بإدخال الكود الخاص بعملية الطرح كما يلي :

```
Dim a, b As Double
a = Text1.Text
b = Text2.Text
Text3.Text = a – b
```

**شرح الكود :**

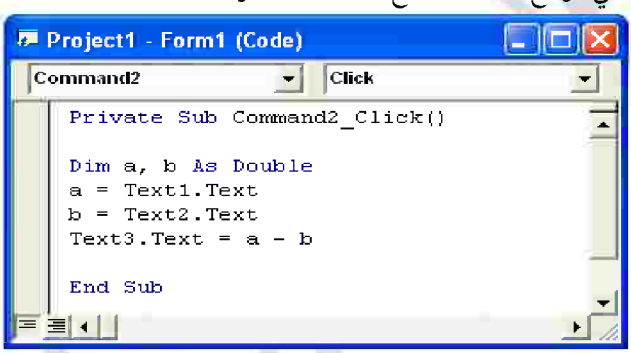
في السطر الأول قمنا بتعريف متغيرين من النوع Double .

في السطر الثاني قمنا بجعل قيمة المتغير a هي محتويات Text1 .

في السطر الثالث قمنا بجعل قيمة المتغير b هي محتويات Text2 .

في السطر الرابع قمنا بطرح محتويات المتغير b من محتويات المتغير a ووضع الناتج في Text3 .

والشكل التالي يوضح لك ما ستصبح عليه نافذة الكود :




قم بعد ذلك بإدخال الكود الخاص بعملية الضرب كما يلي :

```
Dim a, b As Double
a = Text1.Text
b = Text2.Text
Text3.Text = a * b
```

**شرح الكود :**

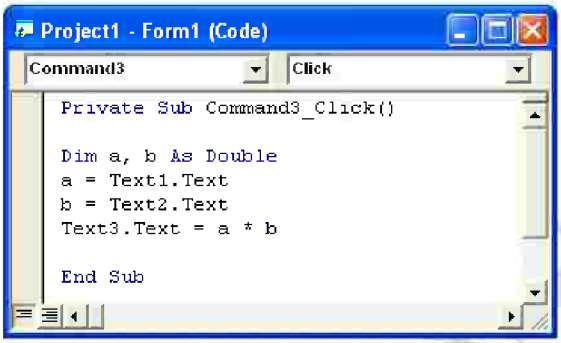
في السطر الأول قمنا بتعريف متغيرين من النوع Double .

في السطر الثاني قمنا بجعل قيمة المتغير a هي محتويات Text1 .

في السطر الثالث قمنا بجعل قيمة المتغير b هي محتويات Text2 .

في السطر الرابع قمنا بضرب المتغير a في المتغير b ووضع الناتج في Text3.

والشكل التالي يوضح لك ما ستصبح عليه نافذة الكود :




قم بعد ذلك بإدخال الكود الخاص بعملية القسمة كما يلى :

```
Dim a, b As Double
a = Text1.Text
b = Text2.Text
Text3.Text = a / b
```

**شرح الكود :**

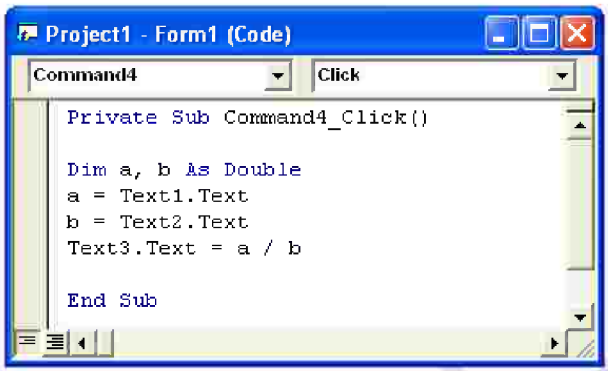
في السطر الأول قمنا بتعريف متغيرين من النوع Double .

في السطر الثاني قمنا بجعل قيمة المتغير a هي محتويات Text1 .

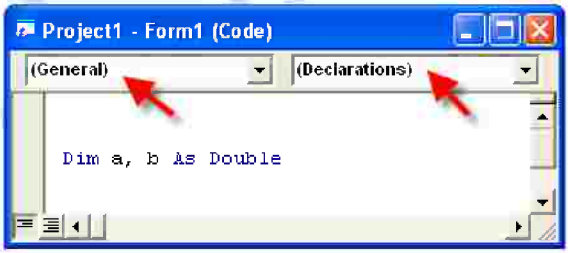
في السطر الثالث قمنا بجعل قيمة المتغير b هي محتويات Text2 .

في السطر الرابع قمنا بعملية قسمة محتويات المتغير a على محتويات المتغير b ووضع الناتج في Text3 .

والشكل التالي يوضح لك ما ستصبح عليه نافذة الكود :



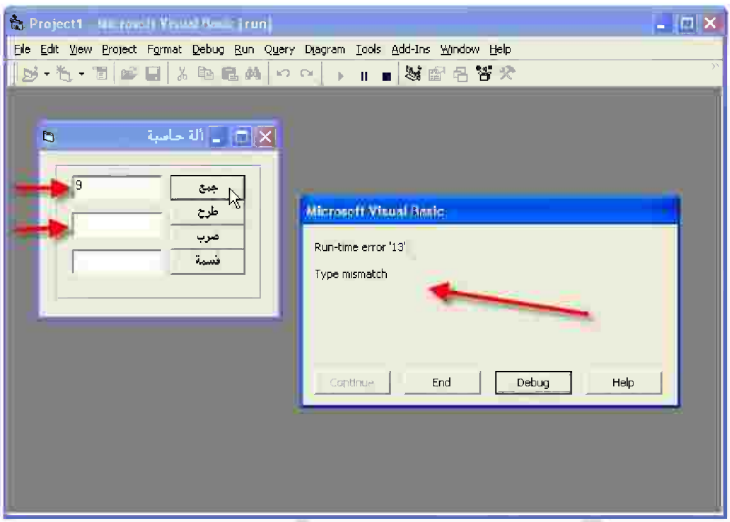

لاحظ ... أننا في جميع الأكواد السابقة كنا نقوم بتعريف متغيرين من النوع Double داخل كل إجراء من الإجراءات الخاصة بالعمليات الحسابية لذلك وبما أننا نحتاج إلى استخدام هذان المتغيران داخل أكثر من إجراء فلا داعي إذا لتكرار تعريفهما مع كل إجراء وعلينا أن نقوم بتعريفهما داخل منطقة التعريفات العامة General Declarations لكي نستطيع بعد ذلك استخدامها بطريقة مباشرة من داخل أي إجراء كما يلي :




وبذلك يمكننا استخدام هذان المتغيران من داخل أي إجراء دون الحاجة إلى تكرار تعريفهما كما بالشكل التالي :


## ملاحظات هامة حول المتغيرات :

لاحظ ... عند استخدام المتغير الرقمي لا بد أن تقوم بإرسال قيمة له فعلى سبيل المثال في التطبيق السابق ستجد أنك إذا قمت بإدخال رقم في TextBox1 ولم تقم بإدخال رقم في TextBox2 ثم قمت بمحاولة إجراء أي عملية حسابية ستظهر لك رسالة خطأ كما بالشكل التالي :



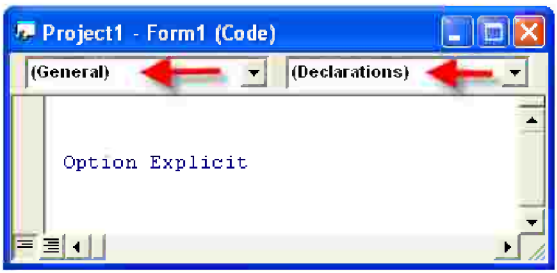

لاحظ ... أن لغة الفيجوال بيسك في الوضع الإفتراضي تتيح لك استخدام المتغيرات مباشرةً دون أن تقوم بالإعلان عنها كما يلي :

```
a = "محمد"
```

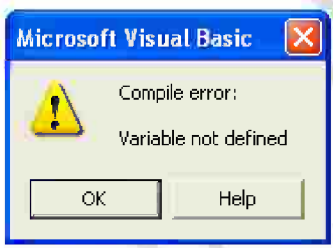
ولكن عند التعامل مع عدد كبير من المتغيرات يمكن أن يتسبب ذلك في العديد من المشاكل لذلك فمن الأفضل أن تجعل لغة الفيجوال تقوم بالتحذير والاعتراض إذا قمت **باستخدام** متغير ما دون أن تقوم بالإعلان عنه وذلك عن طريق كتابة الكود التالي في منطقة التعريفات العامة :

```
Option Explicit
```

لتصبح نافذة الكود كما بالشكل التالي :




وفى هذه الحالة وعند محاولة استخدام متغير دون أن تقوم بالإعلان عنه ستظهر لك رسالة الخطأ التالية :




لاحظ ... أننا بعد استخدام المتغيرات في تطبيق الآلة الحاسبة السابق قد يرى البعض أن المتغيرات تسببت في زيادة حجم الكود وقد يرى أيضاً أننا بإمكاننا تجاهل المتغيرات والاستغناء داخل البرامج ! وهذا بالطبع خطأ كبير لأنني في التطبيق السابق أردت توصيل مفهوم المتغيرات لك في أبسط صورة ممكنة ولكن عند التعامل مع برامج كبيرة سيظهر لك مدى أهمية المتغيرات وفائدتها وما يمكن أن توفره لنا من مجهود وإختصار لعشرات الأسطر من الأكواد كما ستجد أن هناك بعض المهام لا يمكن لنا القيام بها بحال من الأحوال دون وجود المتغيرات كما سيأتى في الفصول المختلفة للكتاب .

## المتغيرات الاستاتيكية Static :

المقصود بالمتغيرات الإستاتيكية هي المتغيرات التي تحتفظ بقيمتها طوال فترة عمل البرنامج ، لذلك يمكن اعتبار أن المتغير العام ( الذي يتم تعريفه في منطقة التعريفات العامة ) هو من نوعية المتغيرات الإستاتيكية .

## عيوب المتغيرات الاستاتيكية Static :

من عيوب المتغيرات الاستاتيكية أنها تقوم بحجز مساحة من الذاكرة لفترة طويلة ( طول فترة عمل البرنامج ) دون الحاجة لذلك .

أيضاً يعيب هذا النوع من المتغيرات أنها أبطأ في التعامل من المتغيرات الديناميكية التي تعملنا معها سابقاً .

## طريقة الإعلان عن المتغير الاستاتيكى :

يتم الإعلان عن المتغير الإستاتيكى عن طريق كلمة Static كما يلى :

```
Static VariableName As DataType
```

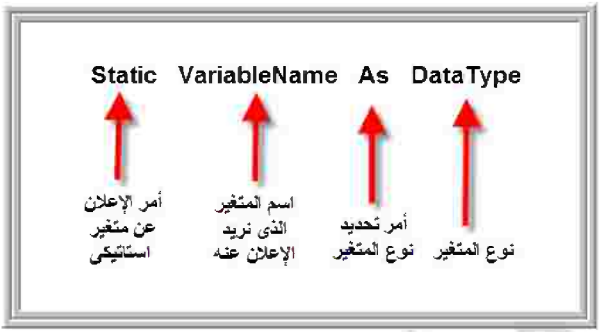
حيث تشير المعاملات السابقة إلى ما يلى :

**Static** : أمر الإعلان عن المتغير الإستاتيكى .

**VariableName** : تحديد أسم المتغير الذي نريد الإعلان عنه .

**As** : أمر تحديد نوع المتغير .

**DataType** : نوع المتغير التي نريد استخدامه .




مثال للإعلان عن متغير استاتيكى :

```
Static a As String
```

في الكود السابق قمنا بالإعلان عن متغير استاتيكى اسمه a من النوع String . لاحظ ... أن المتغيرات الاستاتيكية يتم التعامل معها بنفس طريقة التعامل مع المتغيرات الديناميكية .

## الثوابت Constants :

الثوابت تتشابه كثيراً مع المتغيرات للدرجة التي جعلت بعض المبرمجين يطلقون عليها المتغيرات الثابتة ! فهي أيضاً عبارة عن مكان داخل الذاكرة يستخدم في تخزين المعلومات به وتختلف الثوابت عن المتغيرات في شيئين أساسيين .

1- لا يمكن تغيير قيمة الثابت داخل المشروع .

2- يجب تمرير قيمة للثابت عند الإعلان عنه .

## أنواع الثوابت :

تنقسم الثوابت داخل لغة الفيجوال بيسك إلى نوعيين كما يلى :

1- ثوابت خاصة بلغة الفيجوال بيسك نفسها .

2- ثوابت يقوم المبرمج بالإعلان عنها .

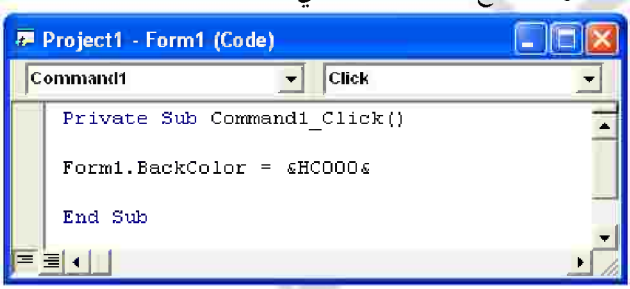
وفيما يلى سنتعرف على كل نوع منهما بشيء من التفصيل .

## أولاً : ثوابت اللغة :

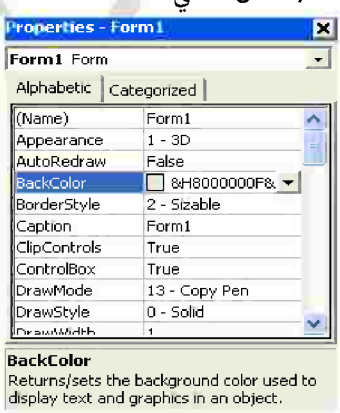
من أبسط وأشهر الأمثلة التي تذكر عند الحديث عن ثوابت اللغة هي الثوابت الخاصة بالألوان ولتوضيح الفكرة بصورة أكثر تخيل أنك تريد تغيير لون خلفية الفورم إلى اللون الأخضر عن طريق الكود من خلال النقر فوق أحد المفاتيح فعند ذلك سنقوم **باستخدام** الكود التالي :

```
Form1.BackColor = &HC000&
```

أي أن نافذة الكود ستصبح كما بالشكل التالي :




بالطبع كلنا نعرف أن المسئول عن ذلك هي خاصية BackColor ولكن السؤال الهام هو كيف عرفنا قيمة اللون الأخضر ؟ وأجيبك بأننا عرفنا ذلك عن طريق خاصية BackColor نفسها كما بالشكل التالي :

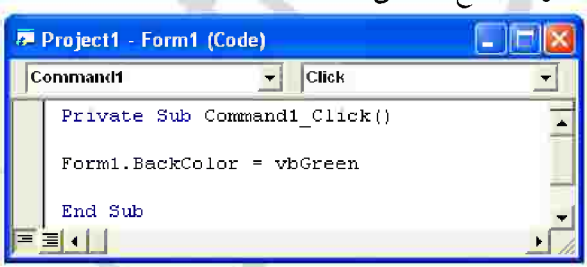


كما ترى بالشكل السابق أن القيمة التي تظهر بجوار خاصية BackColor هي القيمة الخاصة باللون الافتراضي للفورم ويمكنك أن تقوم بنقر السهم الخاص بالخاصية وتقوم بتحديد اللون الذي تريده ليظهر لك بجوار الخاصية رقم يمثل القيمة الخاصة باللون إلى اخترته فقم بنسخه واستخدامه داخل الكود .

ما سبق يمثل احد الطرق التي يمكنك من خلالها التعامل مع الألوان وهي الطريقة التقليدية ولكن يمكنك تجاهل هذه الطريقة بأكملها واستخدام بعض الثوابت الخاصة بالألوان فعلى سبيل المثال الثابت الخاص باللون الأخضر هو vbGreen وعليه فيمكن استخدام الكود التالي لتغير خلفية الفورم إلى اللون الأخضر .

```
Form1.BackColor = vbGreen
```

أي أن نافذة الكود ستصبح كما يلى :




ويمكنك التعرف على المزيد من الثوابت الخاصة بالألوان من خلال الجدول التالي :

| اللون | الثابت |
|---|---|
| أبيض | vbWhite |
| أسود | vbBlack |
| أخضر | vbGreen |
| أزرق | vbBlue |
| أصفر | vbYellow |

| أحمر | vbRed |
|---|---|

**ثانياً : الثوابت التي يقوم المبرمج بتعريفها :**

يقوم المبرمج بالإعلان عن الثوابت واستخدامها داخل البرنامج إذا أراد تثبيت أحد القيم وعدم تغييرها أثناء عمل البرنامج ويتم الإعلان عن الثوابت من خلال كلمة Const كما يلي :

```
Const VaraibleName As DataType = Value
```

حيث يمكنك التعرف على المعاملات السابقة من خلال ما يلي :

**Const** : أمر الإعلان عن الثابت .

**VaraibleName** : أسم الثابت المعلن عنه .

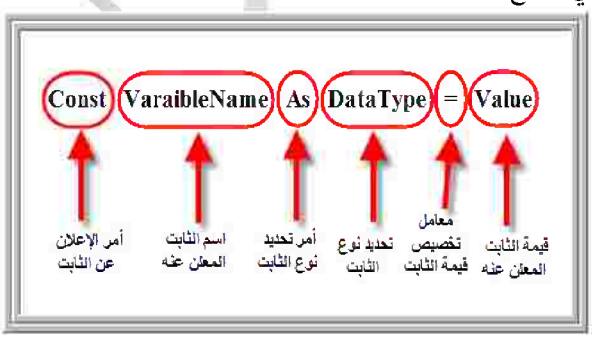
**As** : أمر تحديد نوع الثابت .

**DataType** : نوع الثابت .

**=** : معامل تخصيص قيمة الثابت المعلن عنه .

**Value** : قيمة الثابت المعلن عنه .

والشكل التالي يوضح لك طريقة الإعلان عن الثوابت .




فتأمل معى الكود التالي :

**Const a As String = "سبحان الله وبحمده"**

في الكود السابق قمنا بالإعلان عن ثابت بالاسم a من النوع String وقمنا بتمرير قيمة له سبحان الله وبحمده .

لاحظ ... أنه لا بد من إرسال قيمة للثوابت عند الإعلان عنها .

مثال على استخدام الثوابت :

سنقوم في هذا الجزء بتطبيق مثال عملي على استخدام الثوابت وذلك من خلال برنامج صغير يقوم بحساب مساحة الدائرة ، ومن المعروف أن قانون حساب مساحة الدائرة هو كما يلي :

مساحة الدائرة = الثابت ( ط ) × ( نصف القطر )^2

حيث أن الثابت ( ط ) = 3.14 وهي قيمة ثابتة لا تتغير .

فتعالى معي نتعرف على كيفية القيام بذلك من خلال الخطوات التالية :

1- قم بفتح مشروع جديد ثم قم باضافة أداتين TextBox وأداتين Label وأداة CommandButton .

2- قم بتغيير خصائص تلك الأدوات كما بالجدول التالي :

| الأداة | الخاصية | القيمة الافتراضية | القيمة الجديدة |
|---|---|---|---|
| CommandButton | Caption | Command1 | موافق |
| Label1 | Caption | Label1 | نصف القطر |
| | AutoSize | False | True |
| Label2 | Caption | Label2 | مساحة الدائرة |
| | AutoSize | False | True |

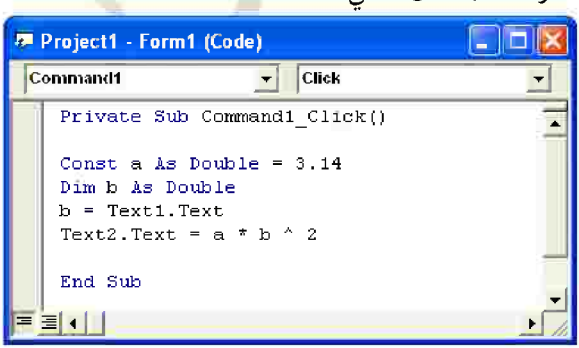
| | | | |
|---|---|---|---|
| | Text1 | Text1 | TextBox |
| | Text2 | Text2 | |

لتصبح الفورم كما بالشكل التالي :


3- قم بإضافة الكود التالي لحدث Click لمفتاح موافق .

```
Const a As Double = 3.14
Dim b As Double
b = Text1.Text
Text2.Text = a * b ^ 2
```

لتصبح نافذة الكود كما بالشكل التالي :



## شرح الكود :

في السطر الأول قمنا بتعريف ثابت بالاسم a من النوع Double مع تحديد قيمته بـ 3.14 ( قيمة الثابت ط ) .

في السطر الثاني قمنا بتعريف متغير بالاسم b من النوع Double ليقوم باستقبال قيمة نصف القطر التي سيقوم المستخدم بإدخالها .

في السطر الثالث قمنا بجعل قيمة المتغير b هي ما سيقوم المستخدم بإدخاله في Text1 .

في السطر الرابع قمنا بتنفيذ قانون مساحة الدائرة وإظهار الناتج في Text2 .

قم الآن بتشغيل البرنامج ثم قم بإدخال قيمة نصف قطر الدائرة التي تريد حساب مساحتها في Text1 ثم أنقر مفتاح موافق لتظهر لك مساحة الدائرة في Text2 كما بالشكل التالي :