المحاضرة الخامسة

الأخطاء Bugs

تسمى أخطاء البرامج بـ bugs و عملية تنقية البرنامج منها بـ debugging.

أنواع الأخطاء Bugs

١. خطأ هجائي Syntax error
٢. خطأ أثناء التشغيل Runtime error
٣. خطأ منطقي Logical error

- الأخطاء في الأوامر البرمجية Syntax Errors وهي الأخطاء التي تحدث في بنية الجمل البرمجية، مثل (الأخطاء المطبعية) الخطأ في كتابة خاصية أو متغير ما (في الـ Spelling)، لكن الفيجوال بيسك سوف يقوم بتلوين الخطأ مباشرة ولن يسمح لك بتنفيذ البرنامج إذا لم تقوم بتصحيح الخطأ.

- الأخطاء وقت تشغيل البرنامج Run-Time Errors وهي الأخطاء التي تحدث بعد تنفيذ البرنامج وفي وقت تشغيله وتسبب للبرنامج التوقف عن العمل، هذه الأخطاء تحدث عندما يحدث أمر غير متوقع وقت البرمجة أو قد يكون هناك خطأ في الأوامر من النوع الأول Syntax Errors يجبر هذا الخطأ البرنامج على التوقف عن العمل، فمثلاً قد تخطئ في كتابة مسار قاعدة البيانات أو مسار صورة سيقوم البرنامج بوضعها في صندوق الصورة، أو قد تأمر البرنامج بفتح القلوبي Floppy وحيث لا يوجد فلوبي في الجهاز. هذه الأخطاء تسمى أخطاء وقت التشغيل أو Run-Time Errors وقد توقف البرنامج عن العمل.

- الأخطاء المنطقية Logic Errors وهي أخطاء يكون سببها المبرمج وهذه الأخطاء قد تتسبب بظهور نتيجة خاطئة بسبب أخطاء في البرمجة، فمثلاً قد يحدث خطأ في حفظ إجمالي الفاتورة لإن المبرمج لم ينتبه إلى مسألة الخصم، ومعظم عمليات معالجة الأخطاء تختص بهذا النوع من الأخطاء وطرق معالجتها.

الخطأ الهجائي Syntax error

١. تسمى بأخطاء وقت التصميم
٢. من ابسط الأنواع.
٣. يظهر سريعا في error list أسفل الشاشة و يسهل علاجه.
٤. يحدث عند كتابة خطأ في هجاء الأوامر أو خطأ في قواعد اللغة.
٥. يكتشف بعضها فى محرر فيجوال (شاشة الكود) بينما يتعرف على البقية فى التنفيذ.

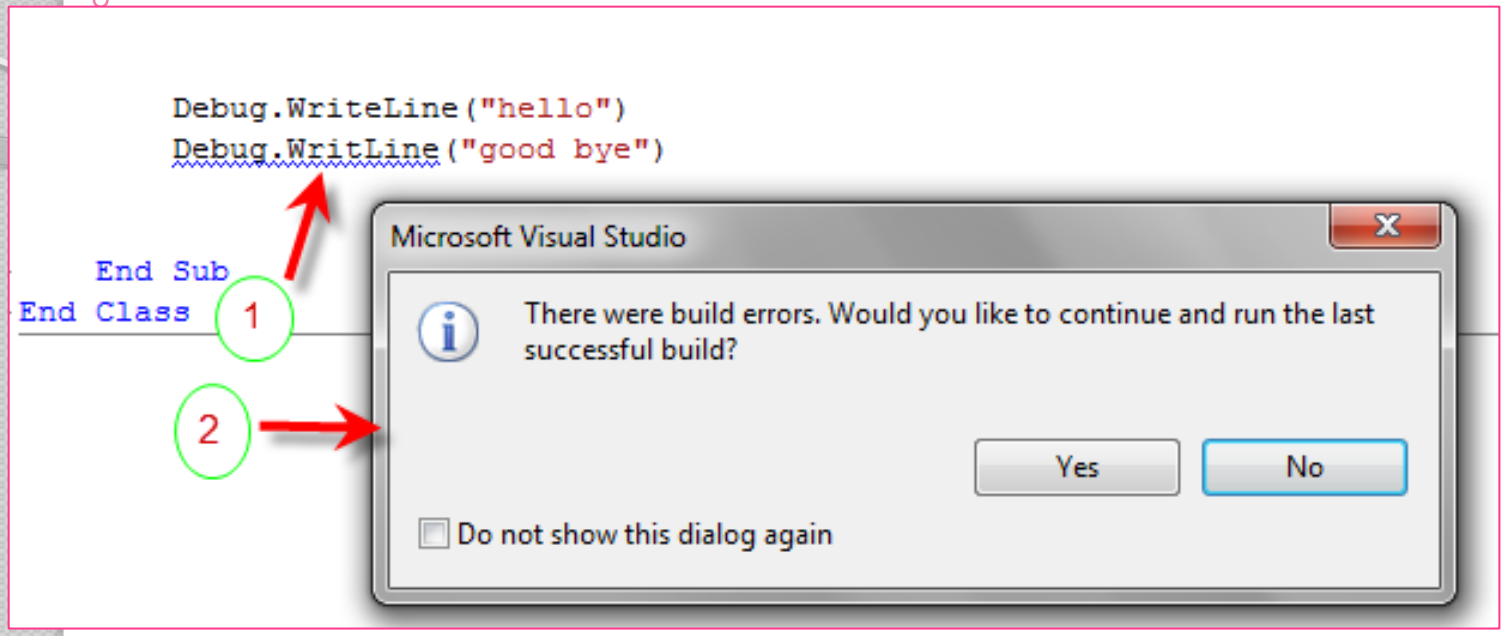
مثل :

if بدون end if

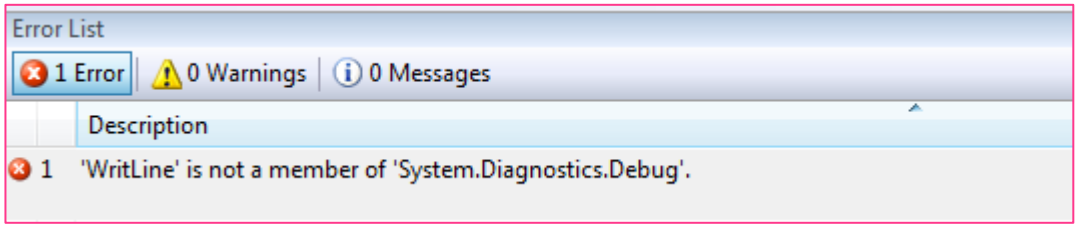
msgbox نكتبها messgbox

مثال:

عند كتابة الخطأ (١) و قمنا بتنفيذ البرنامج فتنتج رسالة (٢)



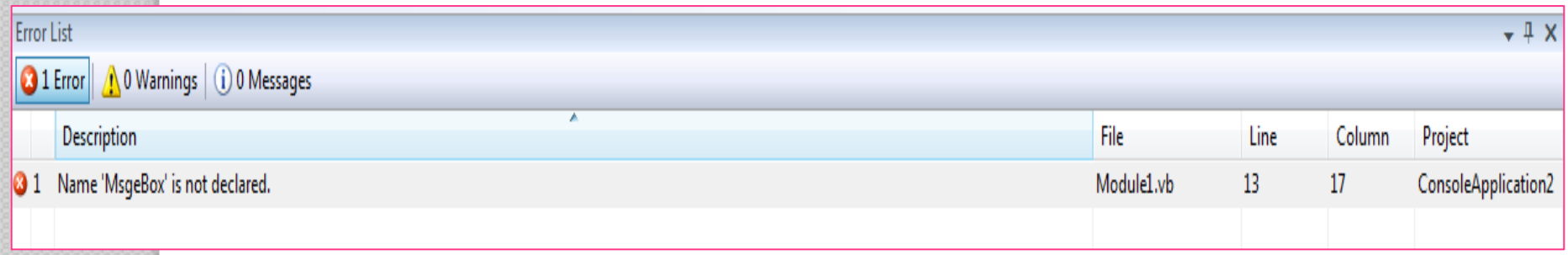
يظهر نوع الخطأ فى قائمة الاخطاء



مثال:

عند كتابة msgebox (.....)

```
MsgeBox(" the element is found in position : " & I + 1)
```



مثال:

```
Public Class Form1

    Private Sub Form1_Load(ByVal
        Select Case
            Csae  1
        End
    End Sub
End Class
```

Name 'Csae' is not declared.

فالرسالة في المستطيل ذو اللون الأصفر وتقول Name 'Csae' is not declared (وتعني الكلمة Csae غير معرفة).

ملاحظة:

اللون الأخضر تحت الكلمة يعني تحذير، اللون الأحمر يعني أن هناك خطأ في الأمر البرمجي Syntax Error، اللون الأزرق يعني أن المترجم الـ Compiler حدد الخطأ، أما اللون الأرجواني Purple فيعني أن هناك خطأ آخر.

طريقة العلاج

- 2-click على الخطأ في error list ينتقل إلى مكان الخطأ لتصحيحه.
- في الإصدارات الحديثة يضع خطا أسفل الخطأ الهجائي.
- الإكمال التلقائي للكود.

خطأ أثناء التشغيل Runtime error

١- تسمى بالاستثناءات exceptions.
٢- يظهر عند تشغيل البرنامج نتيجة قيام المبرمج بكتابة اوامر تحقق عملية غير محققه أثناء التشغيل او عدم قدرة فيجوال على التعرف عليها مثل :

- ✓ عند صدور أمر فتح ملف من مشغل القرص و المشغل غير جاهز فتظهر رسالة runtime error.
- ✓ عند طلب فتح قاعدة بيانات غير موجودة.
- ✓ القسمة على صفر (خطأ overflow).

هذه الأخطاء تسبب قطع البرنامج و الخروج الى بيئة التشغيل.

طرق العلاج

١. عبارة on error resume next.
٢. عبارة ..... on error go to.
٣. مصفوفة الأخطاء.
٤. التركيب try ... catch.

عبارة on error resume next

تسبب تجاهل تنفيذ الجملة الخطأ و الانتقال للجملة التالية مع عدم ظهور رسالة خطأ.

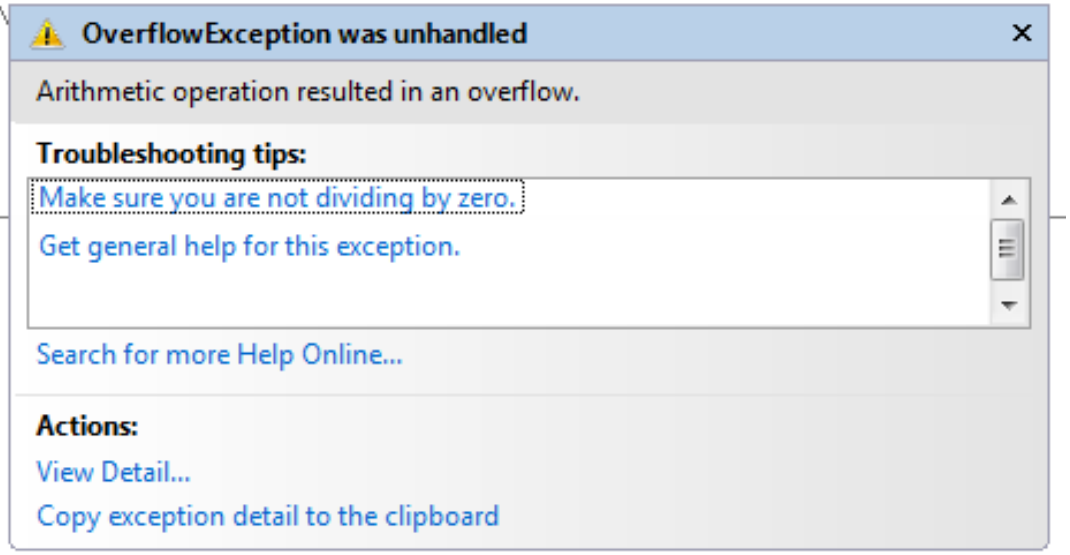
مثال:

```
Dim d, i, r As Integer
i = 0 : d = 100

r = d / i

MsgBox(r)
```

✓ عند تنفيذ البرنامج فيتوقف التنفيذ وينتج مربع رسالة يخبرنا عن الـ error

```
Dim d, i, r As Integer
i = 0 : d = 100
r = d / i
MsgBox(r)
```

```
Sub
ule
```



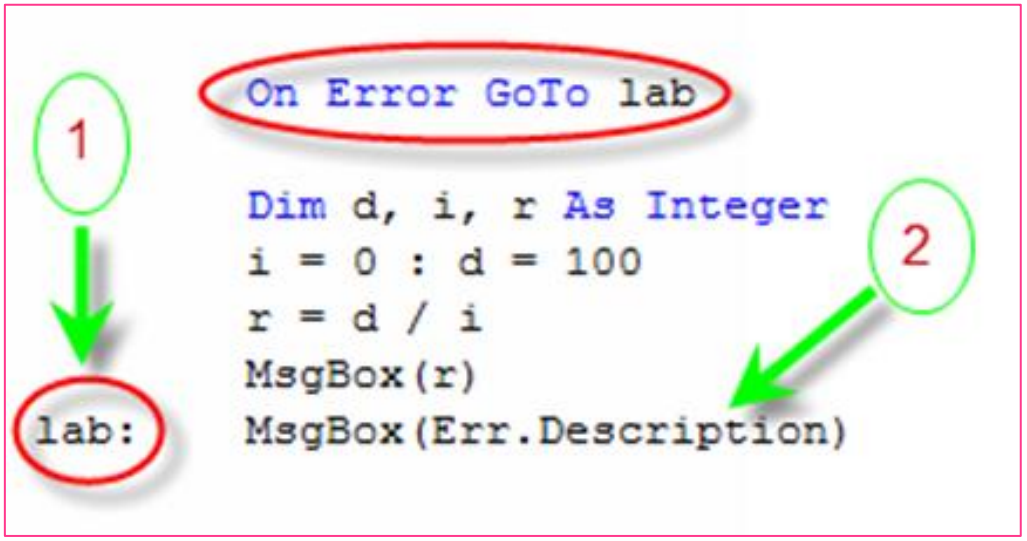
✓ بعد كتابة on resume error ينفذ البرنامج و لا يتوقف لكنه لا ينفذ جملة الخطأ.

```
On Error Resume Next

Dim d, i, r As Integer
i = 0 : d = 100
r = d / i
MsgBox(r)
```

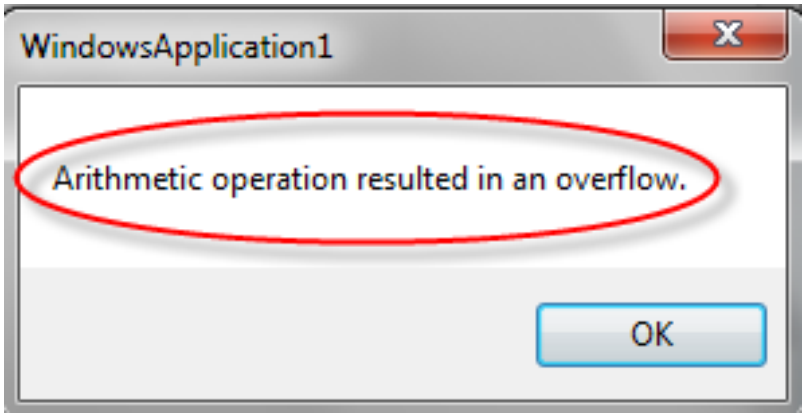
عبارة on error goto .....

لتوجيه تنفيذ البرنامج إلى مجموعة سطور تظهر رسالة الخطأ لنعالجه.

مثال:



فى التنفيذ يظهر مربع رسالة يخبرنا عن نوع الخطأ نتيجة للأمر Err.Description



لكل خطأ يوجد رقم حيث خطأ القسمة على صفر له الرقم 6. فمن الممكن إظهار رسالة الخطأ بالعربي نتيجة لرقم الخطأ.

```
...
On error goto s
Dim d , i , r as integer
d=200
i=0
r=d/i
MsgBox(r)
S:
If err . number=6 then
        MsgBox("تم القسمة على صفر")
        err.Clear    // تنظيف الكائن err من الأخطاء الموجودة به
end if
```

مصفوفة الأخطاء

نقوم بإنشاء مصفوفة تحتوى على رسائل الخطأ بشرط وضع الرسالة في مكان العنصر الذي يشير إلى نفس رقم الخطأ كما هو موضح بالمثال التالي.

مثال:

```
dim  erraray (50) as string
erraray (6) ="تم القسمة على صفر"

On error goto s
Dim d , i , r as integer
d=200
i=0
r=d/i
MsgBox(r)
S:

MsgBox ( erraray ( err . number ) )
```

1

2

3

نستنتج انه أيا كان رقم الخطأ يتم عرض الرسالة المقابلة له في المصفوفة لكن لابد من معرفة أرقام الأخطاء ووضع رسائل في الأماكن المقابلة بالمصفوفة.

تركيب Try .. catch

الصيغة العامة

```
try
      Stat1.
      Stat2.
      ...
Catch  exception1
      action1
Catch  exception2
      action2
  ...
End try
```

```
Try

الجملة البرمجية التي قد يحدث بسببها خطأ

Catch

الجملة التي نريد تنفيذها حال حدوث الخطأ

Finally (اختياريه)

جملة أخيرة نريد تنفيذها سواء حدث الخطأ أم لم يحدث

End Try
```

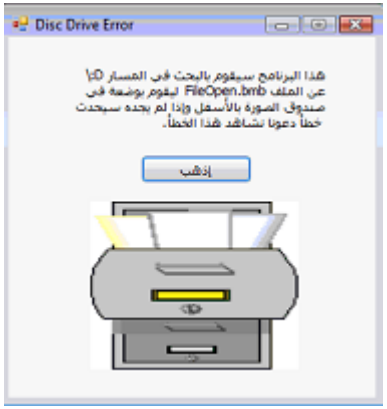
يحاول تنفيذ الجمل التي تلي try (stat1,stat2,...) فإذا نفذت بنجاح يذهب إلى end try ، إذا ظهرت مشاكل اذهب لجمل catch (للتعامل مع الخطأ) فإذا كان الخطأ من نوع exception1 نفذ action1 و إذا كان الخطأ من نوع exception2 نفذ action2 (تشبه select case).

مثال تطبيقي على التعامل مع المسارات والملفات:

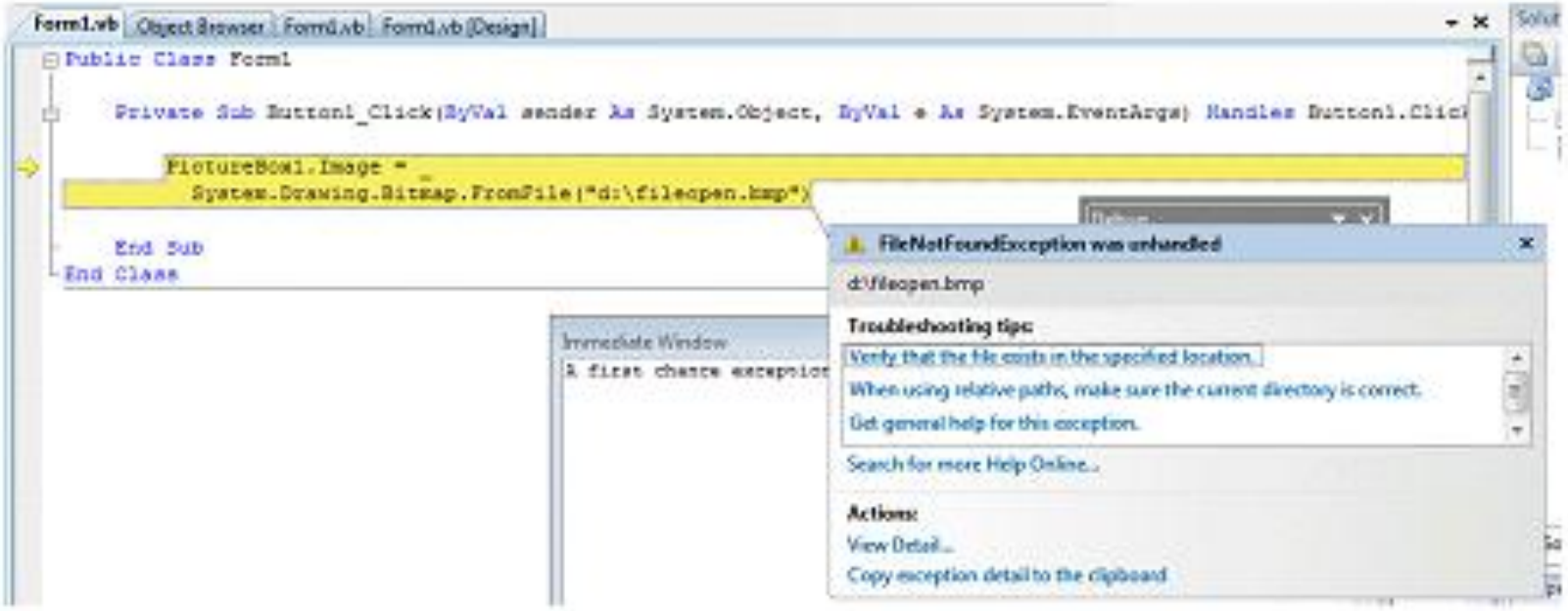
لنأخذ الآن مثال فيه احتمال على حدوث خطأ ما هذا المثال هو التعامل مع الملفات والمسارات حيث سنصمم فورم فيه زر Button ومربع للصورة عند الضغط على الزر سيذهب البرنامج إلى مسار ما وسيقوم بتحميل صورة معينة في مربع الصورة إذا وجدها وإذا لم يجدها فسيحدث الخطأ.

```
PictureBox1.Image = _
    System.Drawing.Bitmap.FromFile("d:\fileopen.bmp")
```

وهذا الكود يجعل البرنامج يقوم بالذهاب إلى القرص D:\ ليبحث عن الملف الصورة fileopen.bmp ويضعها في مربع الصورة PictureBox1 وبما أننا قد وضعنا الصورة في القرص D:\ نقوم بتشغيل البرنامج وضغط الزر ليقوم البرنامج بوضع الصورة في مربع الصورة كما في الصورة التالية:



الآن وبعد أن رأينا كيف قام البرنامج بتحميل الصورة من المسار المحدد نذهب إلى القرص D:\ ونلغي الصورة Fileopen.bmp ونعيد تشغيل البرنامج ونضغط على الزر والآن يحدث الخطأ كما في الصورة (لأن البرنامج لم يجد الصورة في المسار المحدد في الكود):

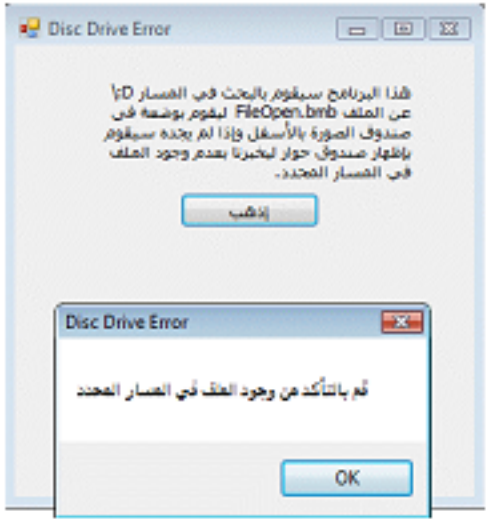


مثال على معالجة الأخطاء باستخدام Try و Catch

```
Try
    PictureBox1.Image = _
      System.Drawing.Bitmap.FromFile("d:\fileopen.bmp")
Catch
    MsgBox("قم بالتأكد من وجود الملف في المسار المحدد")
End Try
```

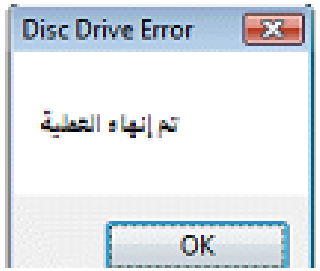
لنجرب الآن تشغيل البرنامج مع عدم وجود الملف fileopen.bmp في D:\ ونرى ماذا سيحدث

للبرنامج لنرى الصورة:



```
Try
    PictureBox1.Image = _
     System.Drawing.Bitmap.FromFile("d:\fileopen.bmp")
Catch
    MsgBox("قم بالتأكد من وجود الملف في المسار المحدد")
Finally
    MsgBox("تم إنهاء العملية")
End Try
```

بعد تنفيذ البرنامج وظهر رسالة الخطأ أو حتى مع عدم ظهور رسالة الخطأ (أي عند وجود الملف وتحميله إلى مربع الصورة) سيظهر لنا صندوق الحوار التالي:



التعامل مع الكائن Err

للعلم الكائن Err كان متوفراً مع النسخ القديمة من الفيجوال بيسك والنسخ الأقدم لكن الفرق هنا هو أن الكائن في فيجوال بيسك 2008 قد تم تحديثه وإضافة العديد من الميزات له من ضمنها إعطاء تقرير أفضل عن الأخطاء وأرقامها وغيرها من التفاصيل كما يوجد العديد من الكائنات التي تتعامل مع الأخطاء مثل الكائن Exception ذو القدرات العالية في التعامل مع الأخطاء.

من ضمن الميزات التي يحويها الكائن Err هي رقم الخطأ Err.Number ووصف الخطأ Err.Description فرقم الخطأ يحتوي على رقم أخر خطأ حدث ووصف الخطأ يحتوي على وصف قصير لذلك الخطأ وباستخدام الخاصيتين Err.Number و Err.Description يمكننا الحصول على رقم الخطأ ووصف الخطأ ووضع أكواد برمجية للتعامل مع كل خطأ بطريقة معينة أو وضع أكواد للتوضيح للمستخدم ماذا يجب القيام به للتعامل مع مثل تلك الأخطاء. نستطيع مسح تقرير الخطأ في الكائن Err بواسطة الكود Err.Clear في الجدول التالي يوجد أرقام لبعض الأخطاء التي قد تحدث في البرنامج ويقوم الكائن Err بإظهار أرقامها إذا حدثت

رقم الخطأ	رسالة الخطأ
5	خطأ في الاستدعاء Procedure call or Argument is not valid
6	الإغراق Overflow
7	أكبر من قوة الذاكرة Out of memory
9	Subscript out of range
11	القسمة على صفر
13	عدم التطابق النوعي Type mismatch
48	خطا عند تحميل مكتبة الـــ dll Error in loading DLL
51	خطأ داخلي Internal error
52	اسم ملف خاطئ أو رقم Bad fi le name or number
53	لا يوجد الملف File not found
55	الملف حالياً مفتوح File already open
57	Device I/O error
58	الملف موجود مسبقاً File already exists
61	القرص ممتلئ
62	Input past end of file
67	توجد ملفات كثيرة

```
Try
    PictureBox1.Image = _
      System.Drawing.Bitmap.FromFile("d:\fileopen.bmp")
Catch When Err.Number = 53
    MsgBox("تأكد من الملف ومسار الملف")
Catch When Err.Number = 7
    MsgBox("لا يمكن فتح الصورة بسبب تجاوزها لإمكانيات الذاكرة", , Err.Description)
Catch
    MsgBox("حدث خطأ عند تحميل الملف", , Err.Description)
End Try
```

استخدام جُمَل Try...Catch المتداخلة

```
Try
    PictureBox1.Image = _
     System.Drawing.Bitmap.FromFile("d:\fileopen.bmp")
Catch
    MsgBox("تأكد من وجود الملف في المسار المحدد ثم إضغط موافق")
    Try
        PictureBox1.Image = _
         System.Drawing.Bitmap.FromFile("d:\fileopen.bmp")
    Catch
        MsgBox("لا يمكنك المحاولة مرة أُخرى")
        Button1.Enabled = False
    End Try
End Try
```

استخدام Exit...Try

إذا راجعت الأكواد التي صممناها في هذا الفصل لصيد الأخطاء ستلاحظ بأننا نغلق جملة Try...Catch بالعبارة End Try والتي بدورها تقوم بإغلاق جملة صيد الأخطاء وينتقل البرنامج للجملة التي تليها في الكود ليقوم بتنفيذها (إذا وجدت).

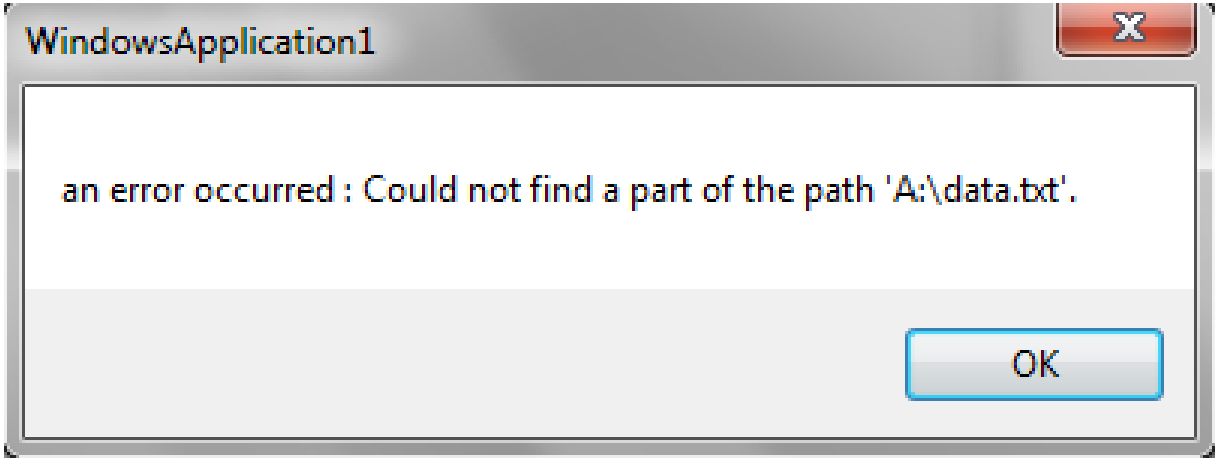
لكن عند استخدامنا للجملة Exit Try في جملة صيد الأخطاء Try...Catch يقوم البرنامج مباشرة بمغادرة جملة صيد الأخطاء عند الوصول إليها (Exit Try) بدون مراجعة بقية الأكواد التي قد تكون بين ثنايا جملة صيد الأخطاء Try...Catch وحتى قبل أن يصل البرنامج إلى العبارة End Try.

```
Try
    If PictureBox1.Enabled = False Then Exit Try
    PictureBox1.Image = _
      System.Drawing.Bitmap.FromFile("d:\fileopen.bmp")
Catch
```

مثال:

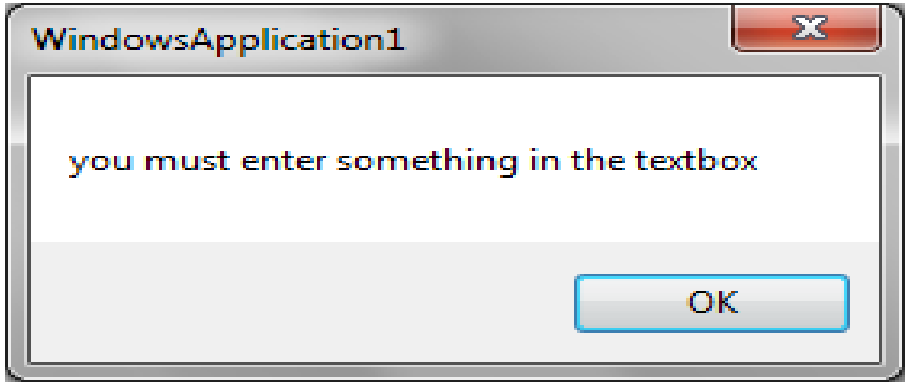
فتح الملف data.txt من القرص a: .

```
Dim filest As System.IO.StreamReader          // لفتح الملف
Try
    filest = New System.IO.StreamReader("a:data.txt")
Catch ex As IO.FileNotFoundException      // نوع الخطأ الملف غير موجود
    MsgBox("the file could not be found")
    Exit Sub
Catch ex As IO.IOException                    // القرص غير موجود
    MsgBox("an error occurred : " & ex.Message)
    Exit Sub
End Try
```



مثال:

```
Dim i As Long
Try
        i = 100 / CLng(TextBox1.Text)
        MsgBox("100/" & TextBox1.Text & " is " & i)

Catch objexception As System.InvalidCastException     // نوع الخطأ عدم ادخال سلسلة فى مربع النص
        MsgBox("you must enter something in the textbox")

Catch objexception As Exception        // نوع الخطأ لا يوجد نوع محدد (اى نوع غير النوع السابق)
        MsgBox("caught an exception that was not an invalid cast")

End Try
```



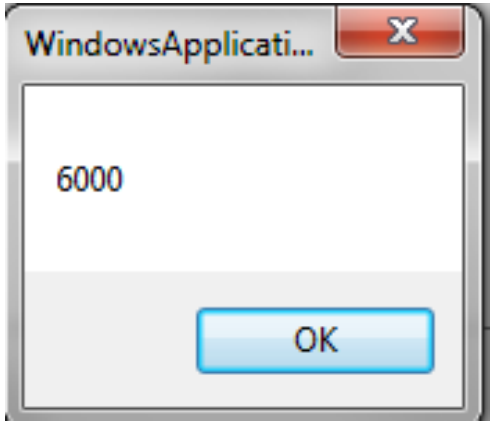
أخطاء منطقية logical errors

ليست أخطاء نحويه في تعليمات البرنامج وليست أخطاء تظهر في التنفيذ بل الخطأ هنا في النتيجة الغير متوقعه للبرنامج .

مثال:

اكتبي إجراء يقوم بحساب اجمالى القيمة المطلوبة لشراء ألف جهاز كمبيوتر حيث سعر الجهاز ٥ آلاف ريال.

```
Dim x1, y1 As Integer
x1 = 1000
y1 = 5000
MsgBox(x1 + y1)
```



في التنفيذ يعطى نتيجة خطأ غير المتوقع نتيجة للعملية + بدلا من *

تحديد الأخطاء المنطقية Logic Errors

الأخطاء المنطقية في البرنامج تعتبر من أصعب الأخطاء عند التصحيح، وذلك بسبب طبيعتها المنطقية حيث تستلزم المراجعة المنطقية لكل عناصر الكود، تذكر بأن الفيجوال بيسك ليس السبب في مثل هذه الأخطاء ولكن الخطأ من عند المبرمج. لنأخذ هذا المثال:

```
Dim Age As Single
If Age > 13 And Age < 20 Then
    TextBox2.Text = "أنت في عمر المراهقة"
Else
    TextBox2.Text = "لست في عمر المراهقة"
End If
```

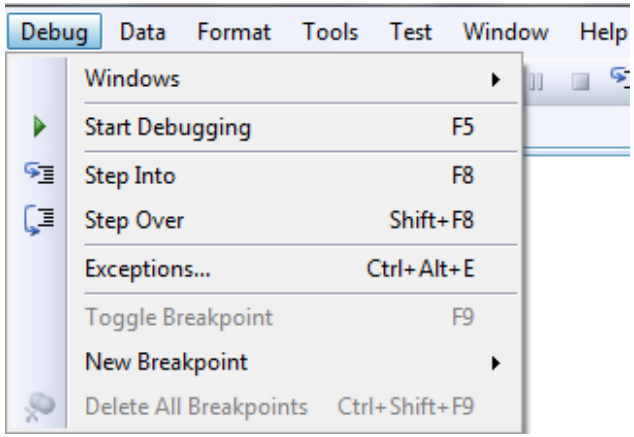
عند مراجعة الكود أعلاه وبما أننا نعرف أن عمر المراهقة يكون من عمر 13 حتى 20 سنة من عمر الشخص، فإن الجملتين أعلاه صحيحتين لكن ماذا إذا أدخلنا العمر 13 أي رسالة ستظهر لنا، بالطبع ستظهر لنا الرسالة التي تقول "لست في عمر المراهقة" مع إن العمر 13 هو من عمر المراهقة، طيب! ما هي المشكلة أو الخطأ، الخطأ هنا أنه لا بد من تحديد العمر بالإشارة >= أكبر أو تساوي 13 وليس بالإشارة > أكبر من 13 فقط، انظر تفاهة هذا الخطأ، كن على ثقة أن معظم الأخطاء البرمجية تأتي من الأخطاء المنطقية ومن جمل مثل هذه (معظم الأخطاء) لكن ليس من السهولة بمكان كشف الأخطاء المنطقية مثل هذه المرة.

طرق الاكتشاف

١. التحكم في سير البرنامج من خلال :
   - شريط debug هناك أدوات لمتابعة سير التحكم.
   - أو من نقاط التوقف breaks (ctrl+b) حيث يتوقف تنفيذ البرنامج عندها لمشاهدة قيم المتغيرات.
٢. إظهار معلومات عن متغيرات / تعبيرات فى نوافذ خاصة.

إجراء عملية التصحيح (تعقب الاخطاء و تصحيحها اثناء run)

من قائمة debug



Step into: تنفيذ البرنامج للسطر الأول ثم التوقف فإذا كان السطر استدعاء لدالة يتم توجيه التحكم إلى الدالة المستدعاة ثم التوقف داخلها قبل تنفيذ السطر الأول بها.

Step over: تنفيذ البرنامج للسطر الأول ثم التوقف فإذا كان السطر استدعاء لدالة يتم توجيه التحكم إلى الدالة المستدعاة لينفذها بالكامل ثم الرجوع للجملة التالية للاستدعاء ثم يتوقف.

من داخل الدائرة الحمراء في الصورة أعلاه نختار الزر Step Into، ليقوم الفيجوال بيسك بتنفيذ الأمر البرمجي وبسبب أن الكود مترابط مع بعضة البعض سينتقل اللون الأصفر إلى السطر التالي ويتوقف، نختار Step Into مرة ثانية، سينتقل اللون الأصفر إلى السطر الذي يليه، مع ملاحظة أن السطر المظلل باللون الأصفر لا يقوم البرنامج بتنفيذه إلا بعد الضغط على Step Into، نقوم بالضغط على Step Into مرة ثالثة، الآن وبما أن العمر الذي وضعناه في صندوق النص هو 16 أي أنه يتوافق مع الشرط الأول في جملة If الشرطية، لذلك لن يقوم البرنامج بالتحقق من الشرط الثاني وإنما سيقوم بالانتقال مباشرة إلى End If. وإذا ضغطنا Step Into للمرة الرابعة نلاحظ أن اللون الأصفر ينتقل إلى End Sub وعند الضغط على Step Into للمرة الأخيرة نلاحظ أن البرنامج يقوم بإظهار الرسالة "أنت في عمر المراهقة" في صندوق النص المحدد لذلك. لعلنا الآن قد عرفنا كيف يتسلسل التطبيق تطبيق الكود البرمجي في برامجنا. دعونا نعيد العملية السابقة مرة أُخرى وبعد وضع العمر 16 والضغط على الزر "قارن العمر" وعند توقف اللون الأصفر فوق الكود:

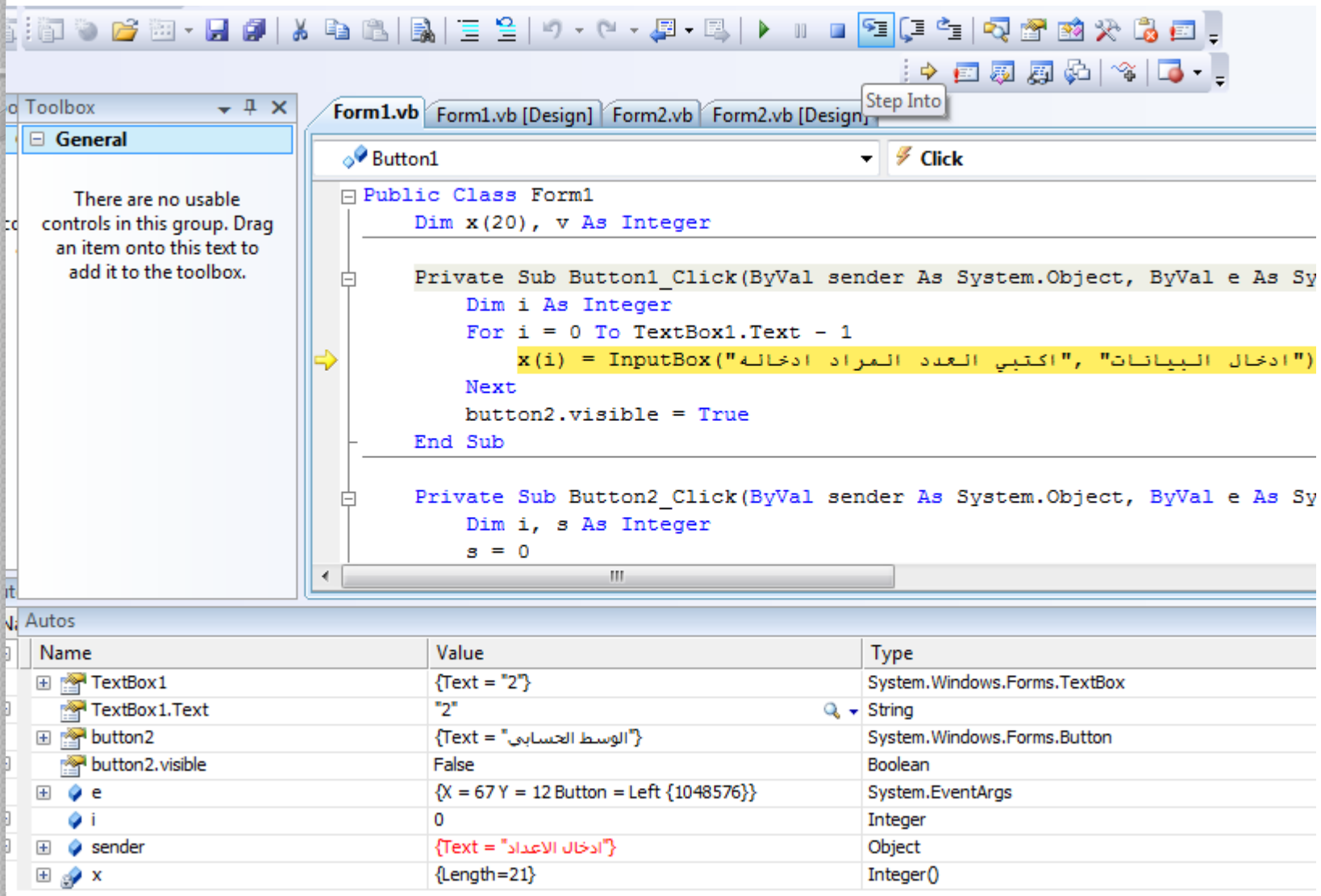
```
Dim Age As Single = TextBox1.Text
```

نذهب الماوس إلى فوق المتغير Age نشاهد الكتابة التالية تحت مؤشر الماوس (Age |0.0) لماذا وماذا تعني هذه القيمة، للتوضيح مادمت في وضع الـــ Debugging إذا وضعت الماوس فوق أي متغير من المتغيرات على طول الكود فسيكتب لك محرر الكود كم هي قيمة المتغير في تلك اللحظة، وقد وضع القيمة 0.0 للمتغير Age لإن المتغير حتى هذه اللحظة لم تتم تعبئته بالقيمة الموجودة في صندوق النص TextBox1.Text ، هذه الميزة ستستفيد منها في البرامج العملاقة والمعقدة، لنعرف متى يتم تعبئة المتغير بالقيمة المحددة له.

ونلاحظ أننا بعد الضغط على Step Into للمرة الثانية ثم نضع الماوس فوق المتغير Age نلاحظ ظهور الكتابة (Age |16.0) فوق الماوس لتعلمنا بأنه قد تمت تعبئة المتغير Age بالقيمة 16 عند الانتقال من السطر الأول في الكود إلى الذي يليه.

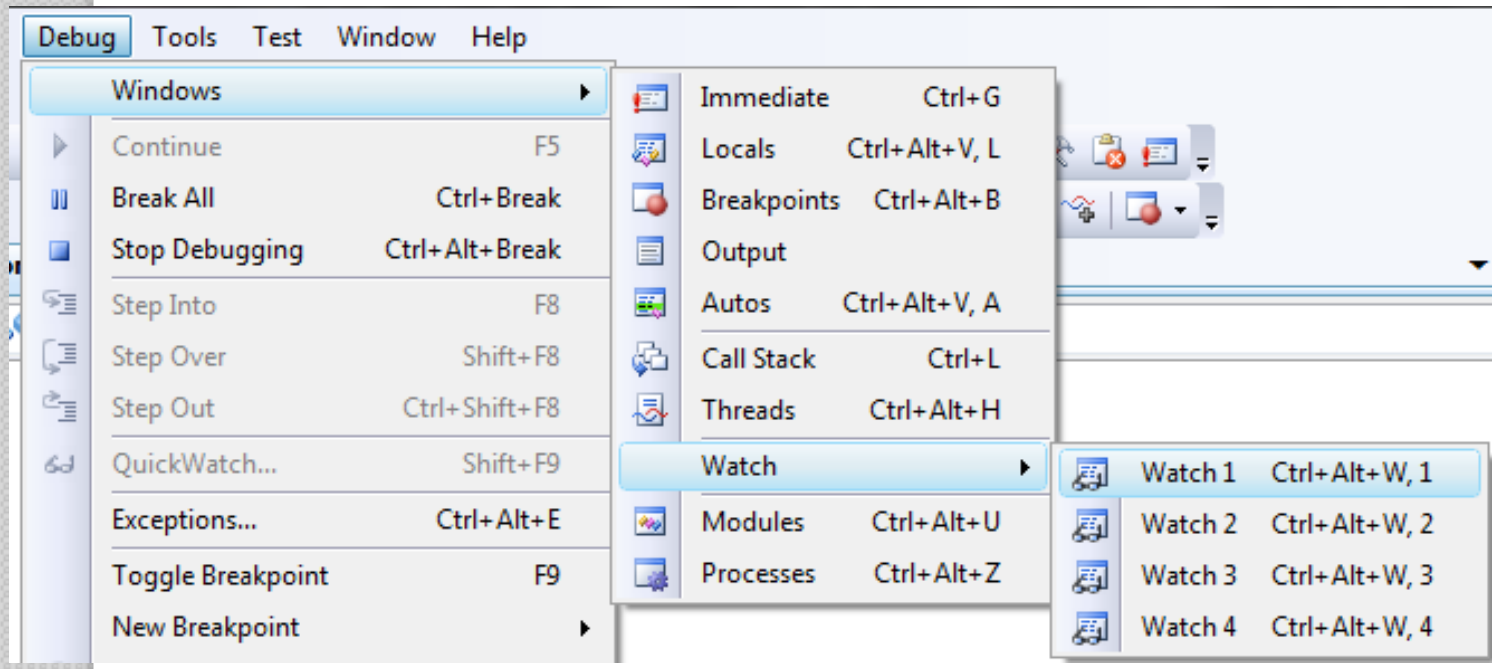
شاشة Autos

- تظهر مع step into اسفل الكود (شاشة عائمة) لعرض معلومات حول قيم المتغيرات.



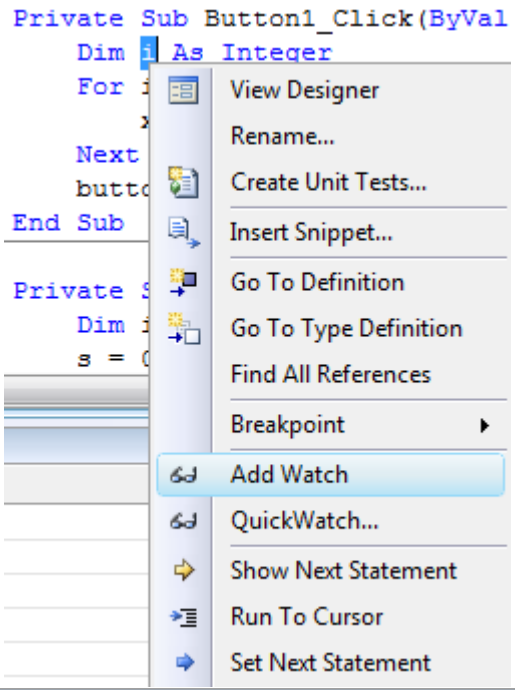
شاشة watch

- تظهر في مرحلة ال debug (run) بعد الضغط علي step into و يمكن استخدام اربع شاشات watch لمراقبة قيم المتغيرات في الاكواد حيث ان شاشة Autos لا يظهر بها جميع قيم المتغيرات بل تظهر قيم المتغيرات التي تغيرت قيمتها عند تطبيق كود معين (السطر المظلل بالأصفر).
- اضفنا شاشة watch و بعد ذلك نضيف لها التعبير المراد تتبع قيم متغيراته r-click علي التعبير ثم add watch فنجد انه اضيف في شاشة ال watch لمراقبة قيمة هذا التعبير.



في فيجوال 2008 فبإمكاننا فتح أربع نوافذ مرقمة 1، 2، 3، 4. لإضافة Watch Window لا بد أن يكون التطبيق في مرحلة الـــ Debugging نختار Debug ثم Windows ثم Watch ثم نختار واحدة من الأربع. ونستطيع إضافة تعبير معين لها لمراقبته فمثلاً يمكننا إضافة التعبير Age >= 13 لتقوم نافذة الـــ Watch Window بمراقبة المتغير Age >= 13.

لنقوم بتظليل المتغير المراد مراقبته ثم الضغط Right-Click على المتغير ثم اختيار Add Watch سنقوم ببيئة التطوير مباشرة بإضافة المتغير إلى الـــ Watch Window فمثلاً نظلل المتغير Age ونختار Add Watch وكذلك TextBox2 وبعد إضافتهم نختار Step Into أو نضغط على F8 لنقوم بنفس العمل ماذا نلاحظ سنلاحظ بأن بيئة التطوير قد قامت بتغيير القيمة للمتغير Age إلى 13 (التي كتبناها في صندوق النص في الفورم) وكذلك لونت القيمة باللون الأحمر لتنتبه بأن هذه القيمة تغيرت في هذه اللحظات. نستطيع إضافة أي بند إلى Watch Window بعمل Right-Click على البند ثم اختيار Add Watch. ويمكننا حذف بنود الـــ Window Watch باختيار أحد البنود ثم الضغط على Delete.



Watch 1

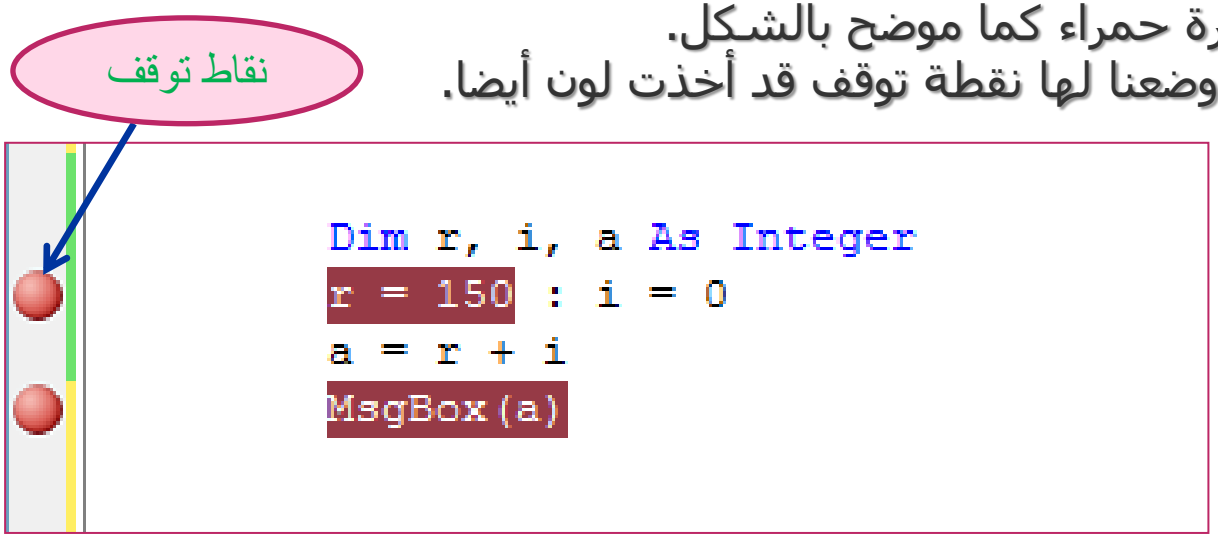
Name	Value
x	Nothing
x	Nothing
i = 0	Name 'i' is not declared.
i2 > 5	Name 'i2' is not declared.
i	Name 'i' is not declared.
s	Name 's' is not declared.
i	Name 'i' is not declared.

نقاط التوقف Breakpoint

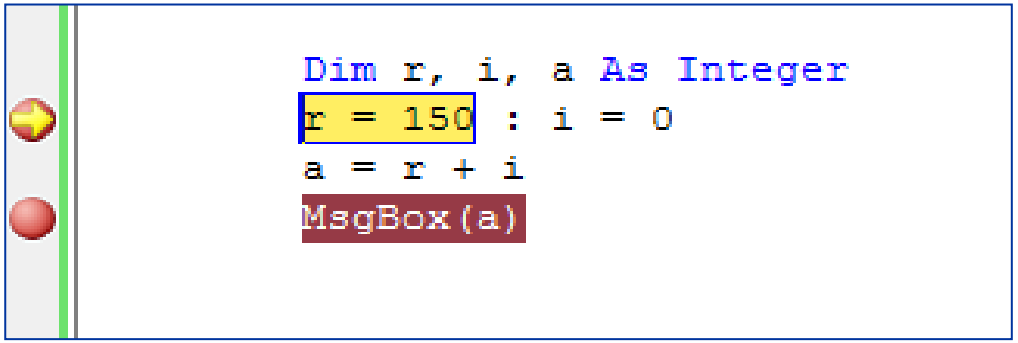
نقاط تضاف لسطور البرنامج عن طريق ctrl+b أو r-click ثم breakpoint ثم insert breakpoint

تظهر نقاط التوقف كدائرة حمراء كما موضح بالشكل.

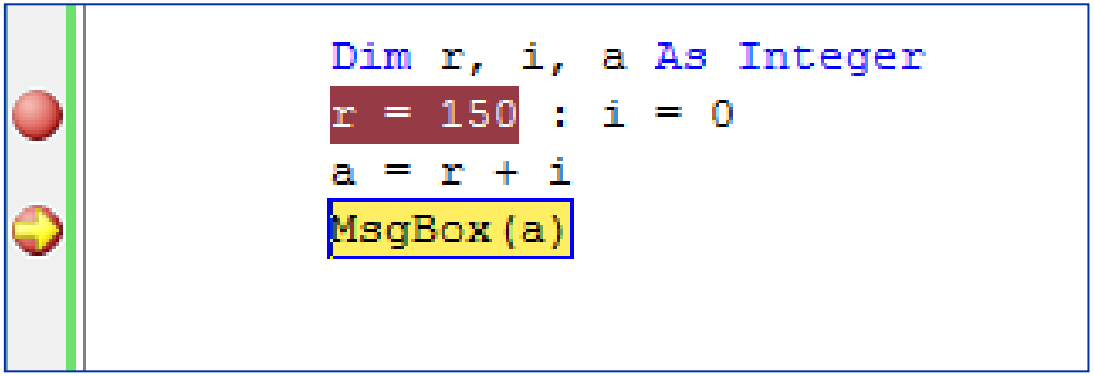
و نجد أن الجملة التي وضعنا لها نقطة توقف قد أخذت لون أيضا.



عند الضغط f5 (run) فنجد أن البرنامج توقف عند نقطة التوقف الأولى لنرى قيمة المتغير r و نجد أن لون الجملة قد تغير للأصفر و شكل نقطة التوقف تغيرت.



عند الضغط على f5 لإكمال التنفيذ نجد أن البرنامج يتوقف عند نقطة التوقف الثانية لنرى قيمة المتغير a.



عند الضغط على f5 لإكمال التنفيذ نجد أن البرنامج أعطى الناتج النهائي و هو 150.