الفصل الثالث
لغة البرمجةBASIC

ان Visual Basic شخصية اصيلة معتزة بامجادها وتاريخها، فما زالتVisual Basic محتفظة بسمات لغة البرمجة BASIC التي تصنف من لغات البرمجة العليا High level programming language ، لغة BASIC هي روح لغة البرمجة ، Visual Basic وهي اللغة التي زادت من اسهم وشعبية Visual Basicالى جانب مصمم النماذج Form Designer. فمعظم الصيغ Syntax التي ظهرت بها اللغة منذ بداية الستينات مازالت مدعومة بشكل جيد في احدث اصدارات . Visual Basic ليس هذا فقط، بل اضيفت اليها العشرات من الدوال والصيغ البرمجية حتى تلائم قوةVisual Basic وتحاكي تطبيقات Windowsفي امكانياتها.

المتغيرات والثوابت

المتغيرات والثوابت هي اساس أي لغة برمجة . إن استيعاب انواع المتغيرات من المسائل الضرورية التي تمكنك من اختيار الانواع المناسبة للمتغيرات سواء لارسالها الى الدوال او لإجراء العمليات الحسابية عليها .بودي التحدث عن مبدئا قابلية الرؤية وعمر الحياة قبل الخوض في تفاصيل المتغيرات.

قابلية الرؤية وعمر الحياة

قابلية الرؤية وعمر الحياة من احد المبادئ الضرورية في جمي ع لغات البرمجة، و Visual Basic يعتبر لغة برمجة حقيقة تدعم هذان المبدئان. قابلية الرؤية– Visibility او المدى - Scope للمتغير تمثل قدرة البرنامج على الوصول الى المتغير واستخدامه، فالمتغير X الموجود في الكود التالي لا يمكن الوصول اليه خارج الاجراء MySub1

```
Sub MySub1 ()
   Dim X As Integer
   X = 20
End Sub
Sub MySub2 ()
  Print X     لا يمثل المتغير X السابق
End Sub
```

اما عمر الحياة LifeTime للمتغير، فهي تمثل الفترة التي يظل فيها المتغير محتفظا بقيمته، فالمتغير X الموجود في الكود السابق، سينتهي ويزال تلقائيا من الذ اكرة بمجرد الخروج من الاجراء Sub1. ولكي تفهم الاسلوب الذي يتبعه Visual Basic لتطبيق مبدأ قابلية الرؤية وعمر المتغيرات، عليك معرفة انواع المتغيرات من منظور الرؤية وعمر الحياة:

المتغيرات المحلية الديناميكية:

المتغيرات المحلية الديناميكية Dynamic Local Variables هي متغيرات تولد مع السطر الذي تعلن عنها فيه داخل الاجراء وتموت بعد نهاية الاجراء مباشرة ويتم تحرير المساحة التي حجزتها هذه المتغيرات في الذاكرة، وبالنسبة لقابلية الرؤية فلن تستطيع الوصول الى هذه المتغيرات الى في نفس الاجراء الذي صرح فيه المتغير . تستخدم الكلمة المحجوزة Dim لتصريح المتغير مع كتابة اسمه ونوعه:

```
Dim sName As String
Dim iAge As Integer
```

اذا كانت الكلمة المحجوزة Option Explicit موجودة في اعلى منطقة الاعلانات العامة لنافذة النموذج او ملف البرمجة BAS ، فعليك اللالتزام بالتصريح آما في الصيغة السابقة، وان لم تكن الكلمة المحجوزة Option Explicit مسطورة فيمكنك تعريف المتغير مباشرة دون الالتزام بعملية التصريح باسناد قيمة ابتدائية له:

```
sName = "  Ali “
iAge = 99
```

صحيح ان الكود السابق يوفر عليك عناء تصريح المتغير لا انه غير محبذ بشكل كبير لدى المبرمجين الجادين، قد يعرض هذا المثال احد الاسباب:

```
sCompanyName = "  الشركة التجارية”
Print sConpanyName     0    ‘ الناتج
```

الناتج من عملية الطباعة Print في الكود السابق لن يكون كما هو متوقع" الشركة التجارية"، فالمتغير المستخدم في السطر الثاني هو sConpanyName وليس sCompanyName وهذا الخطأ كفيل في نمو الشوائب البرمجية Bugs في برامجك.
سبب آخر قد يجعلك تحبذ الالتزام بعملية التصريح وهو ان جميع المتغيرات تكون من النوع Variant ان لم يتم تصريح نوع غير ذلك، والنوع Variant هو ابطأ انواع المتغيرات كما سيأتي لاحقا.
في مثالنا السابق؛ يؤدي فرض الاعلان عن المتغيرات Option Explicit إلى الاعلان عن خطأ و توقف البرنامج . وفي جميع الحالات فإن الخطأ في كتابة اسم المتغير أو اسناد قيمة إلى متغيرات لم يتم الاعلان عنها مسبقاً سيتتسبب في الإعلان عن خطأ، وسيتوقف البرنامج ايضاً.

ملاحظة :توفر لك بيئة التطوير المتكاملة IDE خيار يلزمك بعملية التصريح أي بكتابة الكلمة المحجوزة Option Explicitفي جميع وحدات برامجك آنوافذ النماذج، ملفات البرمجة .... الخ .لتفعيل الاختيار، حدد الاختيار Require Variable Declaration من خانة التبويبEditor في صندوق الحوارOptions.

اخيرا، القيمة الابتدائية للمتغير العددي المصرح هي0 ، والحرفي يكون قيمة حرفية "" ، اما الكائنات فهي لا شئ Nothing.

المتغيرات على مستوى الوحدة:

نقصد بالوحدة هي الوحدة البرمجية Module المتمثلة في ملف برمجة او نافذة نموذج Form او فئة .... Class الخ. من الوحدات المكونة للمشروع . يمكنك تصريح متغير على مستوى الوحدة في منطقة الاعلانات العامة للوحدة أي خارج الاجراءات. قابلية الرؤية لهذا النوع من المتغيرات يكون عام لجميع اكواد الوحدة في حالة استخدام الكلمة المحجوزة Dim او Private:

```
    Dim sName As String
    Dim iAge As Integer
Sub SetData ()
    sName = “ studnt1”
    iAge = 99
End Sub
Sub PrintData ()
    Print sName
    Print iAge
End Sub
```

اما اذا كنت تريد تعريف متغيرات عامة قابلة للوصول من جميع انحاء المشروع ، فالكلمة المحجوزة Public تفي بالغرض:

```
في ملف برمجة ‘
Public iNumberOfUsers As Integer
```

```
' في نافذة نموذجForm1
Public sCurrentUser As String

' في نافذة النموذجForm2
Private Sub Form_Load()
If iNumberOfUsers <= 0 Then
Exit Sub
Else
Me.Caption = Form1.sCurrentUser
End If
End Sub
```

اما عمر الحياة لهذا النوع من المتغيرات فيكون مرافق لعمر حياة الكائن التابع له والمصرح فيه– كعمر حياة المتغيرات الستاتيكية، وبالنسبة للمتغيرات العامة المصرحة في ملفات البرمجة ، فستظل محتفظة بقيمتها حتى نهاية تنفيذ البرنامج.

المتغيرات

نستطيع ان نعرف المتغيرات بمنظورين، بالمنظور الرياضي يعرف المتغير على انه مجهول س يحتوي على قيمة معينة، اما بالمنظور البرمجي– وهو الاهم – يعرف المتغير على انه قيمة تحفظ في ذاكرة الجهاز . وتختلف المساحة المحجوزة لحفظ هذه القيمة باختلاف نوع المتغير، فمتغيرمن النوع Byte لا يستهلك سوى بايت واحد من ذاكرة الحاسب، في حين أن متغير من نوع String قد يحجز مساحة تصل الى 2 جيجابايت. وفيما يلي عرض لجميع انواع المتغيرات المدعومة من قبل Visual Basic:

المتغيرات من النوعByte:

يستطيع هذا النوع الاحتفاظ باي قيمة صحيحة ضمن المجال العددي [255 , 0] وهو اصغر انواع المتغيرات اذ لا يحتجز سوى 1 بايت .

المتغيرات من النوع Integer:

اسند أي قيمة عددية صحيحة في المجال [32,767 , 32,768 -] للمتغيرات من النوع Integer فهي تحجز مساحة 2 بايت. تفيدك المتغيرات من هذا النوع عند التعامل مع الاعداد الصحيحة .

المتغيرات من النوعLong:

المتغيرات من نوع Long تستطيع حمل قيم عددية صحيحة في المجال [2,147,483,647 , 2,147,483,648] فهي تحجز مساحة قدرها 4 بايت للمتغير الواحد، فهي تحمل قيم كبيرة جدا مقللة الخوف من ظهور خطأ وقت التنفيذ Overflow .

المتغيرات من النوعBoolean:

المتغيرات من النوع Boolean هي نفس المتغيرات من النوع Integer ولكن القيم التي تمكنك من اسنادها اليها تكون اما False 0 او True 1 - ، حجم المتغيرات من النوع Boolean مثل حجم المتغيرات من النواع Integer أي 2 بايت، الا انها لا تستخدم سوى 1 بت متجاهلة ال 15 بت الاخرى .صحيح ان الحجم 2 بايت يعتبر زيادة غير مستخدمة، الا ان المتغيرات من النوع Boolean تسهل عليك عملية قراءة وفهم الاكواد.

المتغيرات من النوعSingle:

مجال القيم التي يمكن للمتغيرات من النوع Single احتوائها هو الاعداد الموجبة من 1.401298e-45 الى 3.402823e38 او الاعداد السالبة من 3.402823e38- الى 1.401298e-45-وتستهلك مساحة 4 بايت. ربما يفضل معظم مبرمجي Visual Basic النوع Single على النوع Double لاعتقادهم ان الأول اسرع في التنفيذ من الثاني، هذا الاعتقاد صحيح في النوع القديم من المعالجات والتي لا تحتوي على مساعد رياضي ،Math Coprocessor
اما اغلب المعالجات الجديد تحتوي على المساعد الرياضي وهو خاص بالعمليات الحسابية للاعداد ذات الفاصلة العائمة Floating Point مما يجعل السرعة متقاربة جدا بين النوعين Single و Double .

المتغيرات من النوعDouble:

مجال القيم التي يمكن للمتغيرات من النوع Double احتوائها هو الاعداد الموجبة من 4.9406564581247e-324 الى 1.79769313486232e308 او الاعداد السالبة من 4.9406564581247e-324- الى 1.79769313486232e308- وتستهلك
مساحة 8 بايت. معظم دوال Visual Basic الخاصة بالاعداد تعود بقيمة من النوع Double لذ لك هو النوع المفضل دائما، الا ان عيبه الوحيد هو في المساحة الكبير التي يحتجزها، وقد يظهر هذا العيب جليا في المصفوفات الكبيرة من النوع Double.

المتغيرات من النوعCurrency:

يمكن للمتغيرات من النوع Currency الاحتفاظ بقيم عشرية للفاصلة الثابتة Fixed-Point شريطة ان تكون محصورة في داخل المجال 922,337,203,685,477.5808 -] ،
[922,337,203,685,477.5808وحجمها 8 بايت ايض ا .يوفر هذا النوع من المتغيرات عناء التقريب باستخدام دوال التقريب ك Round ، .... Fix الخ والتي تستخدمها بكثرة مع المتغيرات من النوع Double و Single مما يبطئ العمليات الحسابية، مع ذلك الاستخدام المجرد للمتغيرات من النوع Currency ابطأ خمس او اربع مرات من المتغيرات Double و Single فلا تستخدمها بكثرة في حالة تطبيق آلاف العمليات الحسابية عليها.

المتغيرات من النوع Date:

هذا النوع من المتغيرات يحمل قيم تاريخية تبدأ من التاريخ 1 كانون الثاني 100 الى 31 كانون الاول 9999 ويشمل نفس المتغير وقت يبدأ من الساعة 00:00:00 ص حتى الساعة 23:59:59م وتستهلك مساحة 8 بايت .

المتغيرات من النوع String:

لماذا لغة ال BASIC سهلة؟ والجواب بسبب المتغيرات الحرفية من نوع ! String اذا كنت من مبرمجي C فانسى كل شئ متعلق بقضية حجز المساحة في الذاكرة سواء كان ديناميكيا او ستاتيكيا باستخدام المصفوفات، او التحقق من طول النص وغيرها من الامور التي تتطلب 3 او 6 سطور لاسناد قيمة الى متغير حرفي، ف Visual Basic هو المتكفل بهذه الامور تلقائيا بمجرد تصريح متغير من النوع String او اسناد قيم حرفية له.
منذ الاصدار– VB4 نسخة عيار 32 بت - اصبحت المتغيرات الحرفية Strings تعتمد ترميز UNICODE وليس . ASCII بصفة عامة، يوجد نوعان من انواع المتغيرات الحرفية يوفرهما Visual Basic لك هما المتغيرات ثابتة الطول Fixed-length والمتغيرة الطول Variable-Length. المتغيرات ثابتة الطول هي متغيرات حرفية عدد حروفها محدد في اثناء تصريحها ولا يمكن ان يتغير:

```
Dim FixedStr As String * 13
sFixedStr=  “Computer dep“
```

فالعدد الاقصى من الحروف التي يمكن للمتغير FixedStr ان يحمله هو 13 مما يؤدي الى استهلاك مساحة قدرها 24 بايت– لا تنسى ان UNICODE يستهلك 2 بايت للحرف الواحد .
بالنسبة للمتغيرات المتغيرة الطول Variable-Length فهي باختصار تغطي على جميع عيوب النوع السابق، الا انها تحتجز مساحة تعادل ضعف عدد الحروف 10 + بايتات اضافية تحوي معلومات عن المتغير الحرفي كحجمه وغيرها من التفاصيل التي يخفيها Visual Basic عنك، والعدد الاقصى من الحروف التي يمكن حفظها في هذا النوع يصل إلى 2 جيجا بايت.

المتغيرات من النوع Object:

معظم المتغيرات التي تمثل كائنات سواء صرحت بالنوع Object او بنوع فئات هي متغيرات من النوع Object:

```
Dim X As Object
Dim Y As Form
Dim Z As Text
```

نستطيع اسناد كائن الى كائن عن باستخدام العبارة Set:

```
Set X = New MyClass
Set Y = Form1
Set Z = Text1
```

المتغيرات من النوع Variant:

ظهرت المتغيرات من النوع Variant في الاصدار VB3 ، ويستطيع حمل جميع انواع البيانات السابق ذكرها مثل Long :، Date، .... String الخ. الحجم الذي يستهلكه المتغير Variant هو 16 بايت،البايت الاول يحدد نوع القيمة الموجودة في المتغير، والبايتات من 2 الى 7 لا تستخدم الا في حالة كون القيمة من النوع Decimal ، اما البايتات من 8 الى 15 فهي تمثل القيمة التي يحملها المتغير.
الميزة التي تتميز بها المتغيرات من نوع Variant ليس فقط في امكانية اشتمالها على انواع مختلفة من البيانات بل واجراء العمليات الحسابية او المنطقية عليها، حيث يقوم Visual Basic باختبار نوع المتغيرات ومن ثم اجراء العملية الحسابية او المنطقية المناسبة لها:

```
Dim X As Variant
Dim Y As Variant
Dim Z As Variant
X = 2000 ‘ قيمة من النوعInteger
Y = CLng(2000) ‘ قيمة من النوعLong
Z = X + Y ‘ قيمة من النوعLong
X = CDbl(2.5) ‘ قيمة من النوعDouble
Z = X + Y ‘ قيمة من النوعDouble
```

لا تحاول الاعتماد على الطرق السابقة بشكل استثنائي، فقد تعطيك نتائج غير متوقعة، كما المتغيرات من هذا النوع ابطئ انواع المتغيرات.

الثوابت

ابسط انواع الثوابت هي الثوابت العددية والتي يمكنك كتابتها مباشرة بالنظام العشري Decimal او باضافة البادئة &H للنظام الستعشري Hexadecimal او البادئة &O للنظام الثماني:

```
جميع الاعداد التالية تساوي‘ 15
```

```
Print 15
Print &HF
Print &O17
```

من الضروري ان انبه هنا بان جميع الاعداد المستخدمة في النظام الستعشري Hexadecimal آ0 ، 1، F ، E، 2 ....والنظام الثما ني Octal والتي تكتبها في اكوادك تعتبر في نظر Visual Basic اعداد من النوع Integer

فكرة الثوابت المسماة شبيهه بفكرة المتغيرات، ويكمن الفرق بينهما في أن قيم الثوابت لايمكنك تعديلها وقت التنفيذ لانها قيم ليست موجودة بالذاكرة كقيم المتغيرات، وانما يتم استبدال هذه الاسماء بقيمتها الفعلية في الكود اثناء عملية الترجمة Compiling ، فالثوابت تحفظ مباشرة في الملف التنفيذي EXE للبرنامج.

تستطيع تعريف ثابت جديد باستخدام العبارةConst:

```
Const PI = 3.14
Print PI
```

كما يفضل تعريف نوع الثابت لزيادة سرعة التعامل معه:

```
Const PI As Double = 3.14
Const PROGRAMMER_NAME As String = “Computer department”
Const SPECIAL_VALUE As Long = &H32FE&
```

العمليات الحسابية

الجدول التالي يبين العمليات الحسابية حسب الاسبقية.

رمز العملية (Operation Code)	العملية(Operation)
()	الاقواس
^	الرفع
*، /	القسمة والضرب حسب الاسبقية
Mod	باقي القسمة
-، +	الجمع والطرح حسب الاسبقية
&	الدمج(الجمع الحرفي)
=، >، =>، <، =<، <>	الأسبقية من اليسار الى اليمين
AND , OR	تعابيرمنطقية

التحكم في سير البرنامج

90%من الاجراءات لن تكون ذات قيمة معنوية كبيرة مالم تستخدم عبارات التفرع If و Select او الحلقات التكرارية ك For ... Next او Do ... Loop لتتحكم في سير البرنامج، الفقرات التالية تشرح عبارات التفرع Branch Statements وعبارات التكرار Looping Statements.

التفرع باستخدامIF

جملة If الجميلة لا يستغني عنها اي مبرمج، ليس في Visual Basic وحسب وانما في جميع لغات البرمجة . ومما لا شك فيه تعتبر If من اكثر العبارات استخدام ا في البرنامج، وهي تنجز اما في سطر واحد او– المفضل -عدة سطور:

في سطر واحد'

If *Condition* **Then** *statement*
If *Condition* **Then** *statement1* **Else** *statement2*

```
If X > 0 Then Y = 0
If X > 0 Then Y = 0 Else Y = X
```

```
في عدة سطور'
If Condition Then
   statement
End If
```

```
If X > 0 Then
   Y = 0
End If
```

```
If Condition Then
   Statement1
Else
   Statement2
End If
```

```
If M > 0 Then
   T = 1
Else
   T = -1
End If
```

```
If Condition1 Then
   Statement1
ElseIf Condition2 Then
   Statement2
Else
   Statement3
End If
```

```
If M > 0 Then
   T = 1
ElseIf M < 0 Then
   T = -1
Else
   T = 0
End If
```

عبارة Select Case

تصلح عبارة الشرط if إذا كان جواب الشرط عبارة عن احتمالين أو ثلاثة أما إذا كنت تتوقع عند تقييمك لشرط معين احتمالات كثيرة فمن الأفضل أن نستخدم عبارة Select Case وتكون صيغتها العامة ما يلى : تبدأ العبارة بـ Select Case يليها اسم المتغير او التعبير الذى سيتم اختباره .تأتي بعد ذلك الاحتمالات Case بعد كل منها احدى قيم المتغير الذى ستتم مقارنته ثم يعقبها التعليمات التى ستنفذ إذا كان الشرط صحيحًا أو كان المتغير بهذه القيمة . واخيرًا يأتى Case else ومعناها إذا كان المتغير لا يساوي أيًا من القيم السابقة أو إذا لم يكن الشرط صحيحًا فإن التعليمات التى تلى Else هي التى تنفذ .

```
Select [ Case ] testexpression
    [ Case expressionlist
        [ statements ]
    ]

    [ Case Else
        [ elsestatements ]
    ]

End Select
```

```
Dim number As Integer
number = 7
Select Case number
  Case 1 To 5
    MsgBox "Between 1 and 5, inclusive"
    ' The following is the only Case clause that evaluates to True.
  Case 6, 7, 8
    MsgBox "Between 6 and 8, inclusive"
  Case 9 To 10
    MsgBox "Equal to 9 or 10"
  Case Else
     MsgBox "Not between 1 and 10, inclusive"
End Select
```

عبارة التكرار For

وتستخدم لتكرار أمر ما أكثر من مرة ، بتعيين قيمة البداية والنهاية (ومقدار العد إن أردت) ، ويكون بالشكل التالي:

For *Counter*= *StartValue* **To** *endValue* [**Step** number]

Statements

Next *Counter*

```
For i = 1 To 10
  Print i
Next i

For i = 1 To 10 Step 2
  Print i
Next i

For i = 10 To 1 Step -1
 Print i
Next i
```

مثال : اكتب برنامج بلغة فجوال بيسك لحل المتتالية التالية

X= 2 + 4+ 6 + 8 …..+ n

```
n = Val(InputBox("Enter N", "Sum", 1))
 For i = 1 To n
   Sum = Sum + i
Next i
MsgBox Sum
```

الايعاز Exit For

يمكن الخروج من الحلقة التكرارية for باستخدام هذا الايعاز كما في المثال التاليي:

```
Dim iCounter As Integer
For iCounter = 0 To 100
    IF MsgBox(“ Are you want to Stop” , vbYesNo)= vbYes then
        Exit For
    End if
 ...
Next
```

اما حلقة For Each فهي تطبق على كائنات المجموعاتCollections:

```
Dim X(100) As Integer
Dim Y As Variant
 كود لاسناد قيم للمصفوفة‘
...
 طباعة محتوياتها‘
For Each Y In X
    Print Y
Next
```

الحلقات التكرارية Loop ... Do و Until

الحلقات التكرارية Loop ... Do و Until هي اكثر مرونة من الحلقة التكرارية For لانها لا تحد عدد مرات التكرار لكن تحدد شرط معين لايقاف عملية التكرار .

```
Do While MsgBox( “Are you want to Stop” , vbYesNo) = vbYes
        ……
Loop

Do Until MsgBox( “Are you want to Stop” , vbYesNo) = vbNo
        ……
Loopو
```

ستتم عملية تنفيذ الحلقة مادامت الجملة الشريط صحيحة True في حال استخدام الكلمة المحجوزة While او False في حال استخدام الكلمة المحجوزة Until واذا اردت تنفيذ الحلقة التكرارية مرة واحد على الاقل، ضع حمل الشرط في اسفل الحلقة ، كذالك يمكن استخدام الايعاز Exit Do داخل عبارة شرطية للخروج من الحلقة التكرارية.

طريقة تحويل For loop الى Do While

يمكن تحويل بين عبارات التكرار الغير مشروطة for الى عبارات التكرار المشروطة Do While كما في المثال التالي

```
For i=0 to 10
      ----
Next i

Dim i as integer
 i=0
Do while  i < 10
   -------
Loop
```

المصفوفات

يمكنك Visual Basic من انشاء والتعامل مع المصفوفات Arrays سواء كانت احادية البعد او متعددة الابعاد– قد تصل الى 60 بعدا:

```
Dim OneDim (99) As Intger          100عنصر
Dim TwoDim (4, 9) As Integer       ثنائية الابعاد
Dim ThreeDim (2, 2, 2) As Integer  ثلاثية البعد

Dim OneDArray(0 To 10) As String
Dim TwoDArray(0 To 10, 0 To 10) As Long
Dim OneDArray(15 To 22) As String
```

تستطيع البدء في عملية اسناد القيم بمجرد تصريح المصفوفة مع العلم ان فهرس المصفوفة Array Indexيبدأ من صفر مالم تستخدم الكلمة المحجوزة OptionBase 1 في منطقة الاعلانات العامة للوحدة البرمجية فانه سيبدأ بواحد:

```
OneDim (0) = 100
OneDim (1) = 200
TwoDim (0, 0) = (100, OneDim (0) + OneDim (1))
```

ولمعرفة رقم العنصر الاول استخدم الدالة LBound بينما الدالة UBound تعود برقم العنصر الاخير:

```
Dim lCounter As Long
    For lCounter = LBound (OneDim) To UBound (OneDim)
      Print OneDim (lCounter)
Next
```

المصفوفات السابقة TwoDim ، OneDim و ThreeDim هي مصفوفات ستاتيكية أي ثابتة الحجم لا تتغير في وقت التنفيذ، لذلك فالمرونة الحقيقة ستكون مع المصفوفات الديناميكية Dynamic Arraysالتي تتيح لك التحكم في حجم المصفوفات كلما دعت الحاجة، وتصريحها يكون بدون ذكر حجمها:

```
Dim DynamicArray () As String
```

قبل ان تبدأ في عملية اسناد القيم، عليك استخدام الكلمة المحجوزة ReDim اولا مع ذكر الحجم:

```
ReDim DynamicArray (2)
DynamicArray (0) = “computer”
DynamicArray (1) = “Student”
DynamicArray (2) = “Ali”
```

لو اردت زيادة او تقليص حجم المصفوفة، استخدم ReDim مرة اخرى وعليك معرفة ان جميع محتويات المصفوفة سوف تلغى:

```
ReDim DynamicArray(4)
DynamicArray(3) =”Ahmed”
DynamicArray(4) = “Hassen”
Print DynamicArray(4)     ‘     تبطبع  Ahmed
Print DynamicArray(2)     ‘ لا تطبع شئ
```

اما اذا اردنا تغيير حجم المصفوفة دون المخاطرة بفقد البيانات الموجودة فيها، فالكلمة المحجوزة Preserve يمكن استخدامها للاستخدام:

```
ReDim Preserve DynamicArray(4)
DynamicArray(3) =”Ahmed”
DynamicArray(4) = “Hassen”
Print DynamicArray(4)     ‘     تبطبع  Hassen
Print DynamicArray(2)     ‘  تطبع Ali
```

نقطة اخيرة حول المصفوفات الديناميكة وهي امكانية تدميرها باستخدام العبارة Erase:

```
Erase OneDim
```