همس

في الحاسب الآلي

(شرح)

للشهادة الإعدادية

إعداد الأستاذ / خالد هارون

٠١٠٩٥٧٩٩٥٢٢

٠١١٥٢٢٦٣٧٤٤

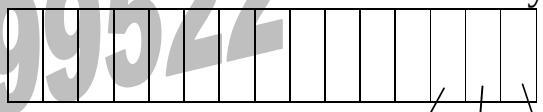
الفصل الأول
البيانات

* مقدمة :

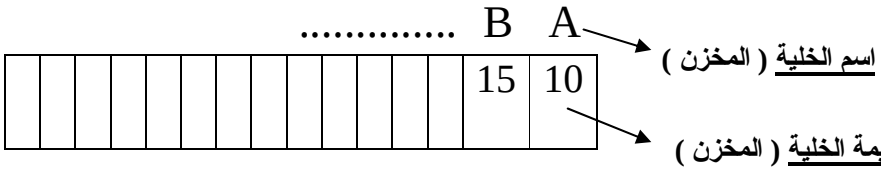
- إن أي بيانات يتم التعامل معها في أي برنامج تكون مخزنة في ذاكرة الجهاز الداخلية (RAM) .

ذاكرة الجهاز R A M

- وتتكون ذاكرة الجهاز الداخلية RAM من ملايين الخلايا (الأماكن) المتساوية ، وتسمى كل خلية من هذه الخلايا Byte .



- وتسمح كل لغات البرمجة عالية المستوى بما فيها لغة Visual Basic بإعطاء أسماء للخلايا التي سيتم تخزين بيانات بها ، حتى يسهل تتبعها والتعامل معها داخل البرنامج .



* والسؤال الآن : هل يمكن أن نسمي الخلايا بأي اسم ؟
- والإجابة : لا . لأن هناك شروط يجب مراعاتها عند تسمية الخلايا (المخازن – المتغيرات) هي:

*** قواعد تكوين الأسماء (المخازن - المتغيرات) في Visual Basic :**

١- يجب أن يبدأ اسم المخزن (الخلية) بحرف من حروف الهجاء الإنجليزية a – z أو علامة underscore (_) .
٢- بعد الحرف الأول يمكن أن يأتي أي عدد من الحروف أو الأرقام أو الرمز Underscore(_) بأي ترتيب .
٣- لا يسمح أن يحتوي الاسم على أي رموز أو علامات خاصة ، المسافة والنقطة وعلامة الاستفهام وعلامة التعجب .
٤- لا يسمح بأن يكون الاسم من الكلمات المحجوزة في Visual Basic مثل كلمة Dim / Integer / Double / Const ..... إلخ .
٥- يفضل اختيار اسم ذات معنى مناسب للمخزن ليسهل عليك تذكر وكيفتة (أي أن يعبر اسم المخزن عن محتواه) .
٦- إذا اضطررت لتكوين اسم من مقطعين ، فيمكن أن تبدأ كل مقطع بحرف كبير (Capital) مثل (StudentName) و (TeacherSalary) .

*** أمثلة لأسماء مخازن (متغيرات) صحيحة :**

Adam2013	Spent_Money	FirstName
_2013	A	_ElAhly

*** أمثلة لأسماء مخازن (متغيرات) غير صحيحة :**

السبب	الاسم	السبب	الاسم
يحتوي على علامة خاصة	Tom&Jerry	بدأ برقم	25January
يحتوي على علامة خاصة	ElAhly-Team	يحتوي على علامة خاصة	Spent.Money
كلمة محجوزة	Integer	كلمة محجوزة	Single

* أنواع البيانات :

- يمكن أن نخزن داخل المخازن (الخلايا) أنواع عديدة من البيانات كما يلي :

١) البيانات الرقمية :

أ) صحيحة : (Byte – Short – Integer – Long)

ب) عشرية : (Single – Double – Decimal)

٢) البيانات الحرفية :

أ) حرف واحد : Char ب) مجموعة أحرف : String

٣) البيانات المتنوعة :

أ) صواب / خطأ : Boolean ب) تاريخ / وقت : Date ج) كائن : Object

* ملاحظات هامة :

١) **كل نوع بيان له :**
- (**حيز تخزين**) : مثل Integer عند استخدامه يشغل ٤ بايت .
- (**مدى**) : وهي حدود القيم المخزنة ، مثل نوع البيان Byte يخزن قيم من ٠ – ٢٥٥

٢) **عند اختيار نوع البيان** يجب مراعاة : حجم البيانات - ونوع البيانات التي سيتم تخزينها .

٣) الخاصية **Text** : نوعها String .

٤) الخاصية **Autosize** : نوعها Boolean .

٥) الخاصية **Width** : نوعها Integer .

٦) الخاصية **Backcolor** : نوعها System.Drawing.color

٧) **الخصائص** هي مكان تخزين البيانات ولها انواع .

٨) توفر لنا لغة Visual Basic إمكانية تحويل القيم إلى نوع البيان المتوافق مع المتغير أو الخاصية وهي ما يطلق عليها (**التحويل الضمني**) .

* المتغيرات Variables:

* **تعريف المتغيرات** : هي عبارة عن أماكن محجوزة بذاكرة الكمبيوتر (RAM) يمكن أن تتغير قيمتها أثناء سير البرنامج .

*** الصيغة العامة للإعلان عن المتغيرات :**

Dim اسم المتغير As نوع البيان

مثال :

```
Dim V_Name As String
```

وهنا تم الإعلان (أي إخبار البرنامج) عن فتح مخزن متغير باسم V_Name من النوع String أي سلسلة حرفية ، وذلك باستخدام الأمر Dim .

* جملة التخصيص Assignment :

- هي عبارة عن جملة من طرفين بينهما علامة (=) .
- ويتم إنشاء جملة التخصيص كما يلي :

الطرف الأيمن	علامة (معامل) التخصيص	الطرف الأيسر
قيمة مجردة (مثل الرقم ٨) **أو** **قيمة من متغير** (مثل Y) **أو** **قيمة خاصية** (مثل Red) **أو** **قيمة من تعبير** (مثل S+V)	=	**متغير** (مثل **X**) **أو** **خاصية** (مثل **Backcolor**)

*** أمثلة على جملة التخصيص :**

مثال ١) :

```
Dim Number As Integer
Number = 5
```

في هذا المثال تم الإعلان عن متغير باسم (Number) من النوع (Integer) في السطر الأول ، وفي السطر الثاني تم تخصيص القيمة المجردة (٥) للمتغير Number .

مثال ٢) :

```
Dim X As Single = 5.6
```

في هذا المثال تم الإعلان عن متغير بإسم (X) من النوع (Single) وتم تخصيص القيمة (5.6) للمتغير (X) في سطر واحد .

مثال ٣) :

```
Dim A , B As Integer
```

في هذا المثال تم الإعلان عن متغيرين (A) و (B) في سطر واحد ، بشرط أن يفصل بينهما بعلامة (,) Comma .

*** ملاحظات هامة :**

١) **علامة (=)** لا يقصد بها التساوي الحسابي : (مثل 10 = 10) ، وإنما يقصد بها إسناد القيمة الموجودة جهة اليمين للمتغير الموجود جهة اليسار (مثل X=10+5 أي إسناد القيمة ١٥ للمتغير X).

٢) تستخدم **علامة (&)** في الربط بين النصوص :

مثال :

```
Dim X , Y As String
Y="Ali"
MsgBox(X & Y)
```

هنا تم إظهار صندوق رسالة يحتوي على القيمة الحرفية "AhmedAli" حيث تم ربط قيمة المتغير X مع قيمة المتغير Y باستخدام علامة الربط بين النصوص & .

٣) **الكلمة المحجوزة (vbCrLf)** تستخدم في إنشاء سطر جديد .

مثال :

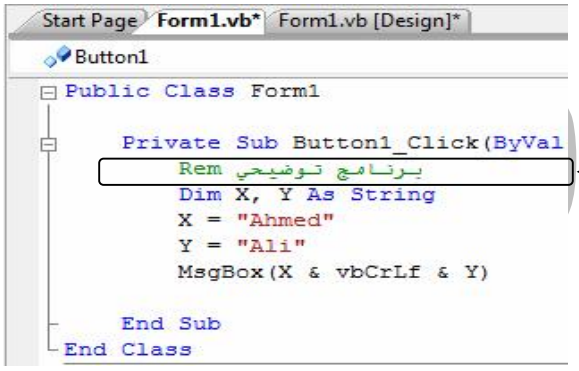
```
Dim X , Y As String
X= "Ahmed"
Y="Ali"
MsgBox(X & vbCrLf & Y)
```

هنا تم إظهار صندوق رسالة يحتوي على القيمة الحرفية "Ahmed" في سطر ، والقيمة الحرفية "Ali" في سطر آخر ، وذلك لأنه تم الفصل بينهما بكلمة vbCrLf .

٤) **تستخدم علامة (_)** حتى يمكن كتابة سطر الكود على أكثر من سطر في حالة إذا كان سطر الكود طويل بعض الشيء ، وذلك لتنظيم وتسهيل عملية قراءة الكود .

٥) **كلمة Me** تعبر عن نافذة النموذج (Form) الحالية .

٦) **تستخدم كلمة Rem أو علامة (')** عند كتابة الملاحظات في الكود ، حيث أن ما يكتب بعدها لا يعتبر أكواد ويتم إهماله بواسطة البرنامج .



السطر الموجود به الملاحظة في الكود

* الثوابت Constants :

* **تعريف الثوابت** : هي عبارة عن أماكن محجوزة بذاكرة الكمبيوتر (RAM) ، يتم تخصيص قيمة لها أثناء الإعلان عنها فقط ، وهذه القيمة لا يمكن أن تتغير أثناء سير البرنامج .

*** الصيغة العامة للإعلان عن الثوابت :**

Const اسم الثابت As نوع البيان = القيمة

أمثلة :

مثال ١) :

Const C_Name As String = "جمهورية مصر العربية"

- في هذا المثال تم الإعلان عن ثابت باسم (C_Name) من النوع الحرفي (String) وتم تخصيص القيمة النصية "جمهورية مصر العربية " له أثناء الإعلان .
- **ويجب ملاحظة أن علامتي التنصيص " " تستخدم في حالة ما إذا أردنا كتابة نص .**

مثال ٢) :

Const Pi As Single = 3.14

- في هذا المثال تم الإعلان عن ثابت اسمه (Pi) له نوع بيان Single وتم تخصيص القيمة الرقمية (٣,١٤) له أثناء الإعلان .

مثال ٣) :

Const BirthDay As Date = # 1/25/2011 #

- في هذا المثال تم الإعلان عن ثابت بإسم (BirthDay) له نوع بيان (Date) وتم تخصيص قيمة التاريخ #1/25/2011# له أثناء الإعلان .
- **ويجب ملاحظة أن علامتي # # تستخدم في حالة ما إذا أردنا كتابة تاريخ أو وقت .**

- **ملاحظة** :

- المتغيرات والثوابت لا نستطيع استخدامها إلا في نطاق إعلانهما ، فإذا تم الإعلان عنهما في نطاق إجراء الحدث (مثل Button1 , Button2) فإنه لن يمكن استخدامهما إلى في نطاق الحدث فقط . أما إذا تم الإعلان عنهما على مستوى التصنيف Class فعندها يمكن استخدامهما في أي مكان داخل البرنامج ولن نحتاج للإعلان عنهما عند كل نطاق إجراء حدث .

* الأخطاء Errors :

أولاً : الأخطاء الإملائية والنحوية Syntax Errors :

- وتحدث عند كتابة الكود بصورة غير سليمة (كأن نكتب Din بدلاً من Dim) أو (أن يتم الإعلان ثابت دون تخصيص قيمة له) .
- وللتغلب على هذا النوع من الأخطاء فإن بيئة IDE في برنامج Visual Basic لا تسمح بوجود أي خطأ من هذا النوع .

ثانياً : الأخطاء المنطقية Logical Errors :

- وتحدث نتيجة استخدام تعبيرات حسابية بناؤها غير سليم في جمل التخصيص (كأن تقوم بجمع الطول + العرض عند حساب مساحة مستطيل مثلاً بدلاً من أن تضرب الطول * العرض) ، وبالتالي فإن البرنامج سوف يعمل دون أن يعطي أي رسائل خطأ ولكن النتائج ستكون غير سليمة .
- وللتغلب على هذا النوع من الأخطاء يجب مراجعة الكود المكتوب جيداً ، واختبار البرنامج على بيانات سبق التأكد من صحتها .

ثالثاً : الأخطاء عند (أثناء) التشغيل Runtime Errors :

- وتحدث في حالتين :

أ) تخصيص قيمة أكبر من مدى نوع البيان المستخدم (كأن يتم الإعلان عن متغير من النوع Byte ثم نقوم بتخصيص القيمة ٤٥٠ له ، وهذا خطأ ، حيث أن مدى نوع البيان Byte هو من صفر إلى ٢٥٥) .

ب)تخصيص قيمة نوعها يختلف عن نوع البيان المستخدم (كأن يتم الإعلان عن متغير من النوع Integer مثلاً ، ونقوم بعدها بتخصيص القيمة الحرفيه "Ali" لهذا المتغير ، وهذا خطأ ، حيث أن المتغير من النوع Integer هو بيان عددي ، أما القيمة "Ali" فهي قيمة حرفية) .

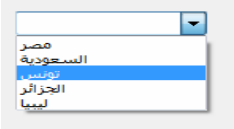
- ويمكن التغلب على هذا النوع من الأخطاء بإحدى الطريقتين التاليتين :

١) باستخدام جملة Try/Catch :

```
Try
    الكود المحتمل أن يظهر منه خطأ أثناء التشغيل
Catch
        الكود الخاص بمعالجة الخطأ إذا وقع
EndTry
```

* **ملاحظة** : في حالة وجود خطأ يتم تنفيذ الأوامر التي تلي Catch ، أما في حالة عدم وجود خطأ فيتم تنفيذ الأوامر التي تلي EndTry .

٢) عن طريق توفير الأدوات للمستخدم مع ضبط خصائصها حتى لا ندع له مجالاً للوقوع في أخطاء . ففي الComboBox التالي لا يستطيع المستخدم اختيار دولة ليست موجودة في القائمة ، وبهذا نكون قد جنبناه الوقوع في الأخطاء .



* أولويات تنفيذ العمليات الحسابية :

* عند إعطاءك أي عملية حسابية لتقوم بحلها ، فعليك مراعاة أولويات تنفيذها حيث نقوم بحل المسألة بالترتيب التالي :

١- العمليات بين الأقواس (ونبدأ بفك الأقواس الداخلية في حالة وجود أقواس متداخلة) .
٢- عمليات الأسس ^ .
٣- عمليات الضرب * والقسمة / .
٤- عمليات باقي القسمة Mod .
٥- عمليات الجمع + والطرح - .

*** ملاحظة هامة** : نبدأ حل المسألة دائماً من جهة اليسار .

*** مثال :**

١- ما هو ناتج العملية الحسابية التالية :

4*2+3^(1+2)Mod4/ 2

- نبدأ أولاً بفك الأقواس ... (1+2) = 3
- الأسس ... 3^3 = 27
- الضرب ...2*4 = 8
- القسمة ... 4/2 = 2
- باقي القسمة ... 27Mod2 = 1
- الجمع ... 8+1 = 9

4*2+3^3Mod4/ 2
4*2+27Mod4/ 2
8+27Mod4/2
8+27Mod 2
8+1
9

*** ملاحظة** : عدم مراعاة أولويات تنفيذ العمليات الحسابية ينتج عنه أخطاء من النوع المنطقي .

الفصل الثاني
التفرع

* معاملات المقارنة :

المعامل	المعنى	المعامل	المعنى
=	يساوي	< >	لا يساوي
>	أكبر من	<	أصغر من
>=	أكبر من أو يساوي	=<	أصغر من أو يساوي

* المعاملات المنطقية :

أولاً : المعامل And (و) :

A	B	الناتج (A And B)
True	True	True
True	False	False
False	True	False
False	False	False

- يتضح من الجدول السابق أن المعامل And (و) يعطي ناتج نهائي (صواب True) في حالة واحدة فقط وهي أن يكون ناتج كافة الشروط (صواب True) .

ثانياً : المعامل Or (أو) :

A	B	الناتج (A Or B)
True	True	True
True	False	True
False	True	True
False	False	False

- يتضح من الجدول السابق أن المعامل Or (أو) يعطي ناتج نهائي (صواب True) في حالة أن يكون ناتج أحد الشروط (صواب True) .

ثالثاً : المعامل Not :

A	Not (A)
True	False
False	True

- المعامل Not هو معامل منطقي يقوم بعكس الناتج ، فإذا كانت قيمة A صواب يقوم المعامل Not بعكسها لتصبح False ، وهكذا ....

* التعبيرات الشرطية :

التعبير الشرطي هو جزء من كود البرمجة يكون ناتجه إما صواب (True) أو خطأ (False) وذلك بناءاً على قيمة خاصية أو متغير أو بيان آخر .

أمثلة :

١) ناتج التعبير الشرطي 100< >100 ..... False ..... لأن القيمتين متساويتين
٢) ناتج التعبير الشرطي 600 => 300 .... True..... لأن ال٣٠٠ أقل من أو تساوي ٦٠٠
٣) ناتج التعبير الشرطي 6.5 > 4 ..... False ..... لأن ال ٤ ليست أكبر من ال ٦,٥

*** كثيراً ما نحتاج إلى التفرع (أي تنفيذ مجموعة خطوات بناءاً على سؤال معين أو شرط) . وللتعبير عن التفرع برمجياً نستخدم جمل معينة في لغة البرمجة هي :**

أولاً : التفرع باستخدام جملة If…Then (البسيطة) :

*** الصيغة العامة :**

```
If شرط Then
   Code
End If
```


- في جملة If…Then البسيطة يتم تنفيذ الأوامر التي تلي Then في حالة تحقق الشرط ، أما في حالة عدم تحقق الشرط فيتم تنفيذ الأوامر التي تلي End If .

- مثال ١) :

```
Dim X As Byte
X=60
If  X>=50 Then
    MsgBox("ناجح")
End If
```

- ففي المثال السابق تم اختبار الشرط (X>=50) ، ونجد أن الشرط تحقق ، أي أن ناتجه True ، لأن ال ٦٠ أكبر من أو يساوي ال ٥٠ ، ولذلك سيتم تنفيذ الأوامر التي تلي Then وهي إظهار صندوق رسالة يحتوي على كلمة "ناجح" .

- مثال ٢) :

```
Dim X As Byte
X=20
If  X>=50 Then
    MsgBox("ناجح")
End If
```

- ففي المثال السابق تم اختبار الشرط (X>=50) ، ونجد أن الشرط لم يتحقق ، أي أن ناتجه False ، لأن ال ٢٠ ليست أكبر من أو يساوي ال ٥٠ ، ولذلك سيتم تنفيذ الأوامر التي تلي End If وهنا لا توجد أوامر تلي End If لذلك لن ينفذ البرنامج أي أمر .

ثانياً : التفرع باستخدام جملة If …Then…Else :

- تختلف هذه الجملة عن السابقة في وجود كود يتم تنفيذه في حالة أن يكون ناتج التعبير الشرطي False

*** الصيغة العامة :**

```
If شرط Then
   Code
Else
   Code
End If
```


- في جملة If …Then…Else يتم تنفيذ الأوامر التي تلي Then في حالة تحقق الشرط ، أما في حالة عدم تحقق الشرط فيتم تنفيذ الأوامر التي تلي Else .

- مثال ١) :

```
Dim X As Byte
X=60
If  X>=50 Then
    MsgBox("ناجح")
Else
    MsgBox("راسب")
End If
```

- ففي المثال السابق تم اختبار الشرط (X>=50) ، ونجد أن الشرط تحقق ، أي أن ناتجه True ، لأن ال ٦٠ أكبر من أو يساوي ال ٥٠ ، ولذلك سيتم تنفيذ الأوامر التي تلي Then وهي إظهار صندوق رسالة يحتوي على كلمة "ناجح" .

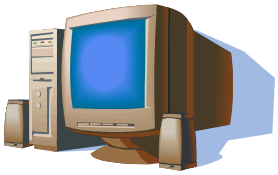
- مثال ٢) :

```
Dim X As Byte
X=20
      If  X>=50 Then
            MsgBox("ناجح")
      Else
            MsgBox("راسب")
      End If
```


- وفي المثال السابق تم اختبار الشرط (X>=50) ، ونجد أن الشرط لم يتحقق ، أي أن ناتجه False ، لأن ال ٢٠ ليست أكبر من أو يساوي ال ٥٠ ، ولذلك سيتم تنفيذ الأوامر التي تلي Else وهي إظهار صندوق رسالة يحتوي على كلمة "راسب" .

ثالثاً : التفرع باستخدام جملة If …Then…ElseIf :

- وتستخدم هذه الجملة عند وجود أكثر من تعبير شرطي .



*** الصيغة العامة :**

```
If     شرط    Then
   Code
ElseIf  شرط Then
   Code
ElseIf  شرط Then
   Code
.
.
.
Else
    Code
End If
```


- في جملة If …Then…ElseIf يتم اختبار الشرط الأول فإذا تحقق ينفذ ما يلي Then ، وإذا لم يتحقق يقوم باختبار الشرط الثاني ، فإذا تحقق ينفذ ما يلي Then ، وإذا لم يتحقق يختبر الشرط الذي يليه ..... وهكذا حتى نهاية الجملة .

مثال:

```
Dim Degree As Single
Degree = 12
      If Degree=0 Then
            MsgBox("درجة الحرارة تساوي الصفر")
      ElseIf Degree<0 Then
            MsgBox("درجة الحرارة أقل من الصفر")
      Else
            MsgBox("درجة الحرارة أكبر من الصفر")
      End If
```


- في المثال السابق سيتم إظهار صندوق رسالة يحتوي على "درجة الحرارة أكبر من الصفر" وذلك لأن قيمة Degree المخزنة = ١٢

رابعاً : التفرع باستخدام جملة Select … Case :

- جملة Select … Case تشبه جملة If … Then …ElseIf غير أنها فعالة بشكل أكثر عندما يكون التفرع معتمداً على قيمة متغير واحد ، كما أنها تجعل الكود مفهوم بشكل أكثر .

*** الصيغة العامة :**

```
Select Case  Variable
  Case Valu1
     Code
  Case Valu2
     Code
  Case Valu3
     Code
   .
   .
Case Else
    Code
End Select
```

مثال:

```
Dim Degree As Single
Degree = 12
    Select Case Degree
        Case Is = 0
            MsgBox("درجة الحرارة تساوي الصفر")
        Case Is <0
            MsgBox("درجة الحرارة أقل من الصفر")
        Case Else
            MsgBox("درجة الحرارة أكبر من الصفر")
    End Select
```

*** بعض الملاحظات الهامة :**

١) الأمر **Setfocus** هو وسيلة Method خاصة بصندوق النص TextBox وتعني نقل التركيز إليه ووضع المؤشر بداخل صندوق النص .

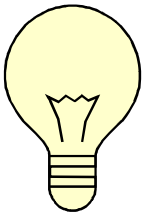
٢) إجراء الحدث **SelectedIndexChanged** هو الحدث الافتراضي للأداة ListBox ويتحقق هذا الحدث عند اختيار عنصر من عناصر الأداة ListBox .

٣) يتم إضافة عناصر للأداة ListBox عن طريق الخاصية **Items** ، ويتم ترتيب عناصر القائمة بحيث يكون العنصر الأول ترتيبه صفر ، والثاني ترتيبه ١ ، وهكذا ..... .

٤) يتم معرفة العنصر المحدد عن طريق الخاصية **SelectedIndex** للأداة ListBox .

٥) عند جعل الخاصية **Multiline** للأداة TextBox تساوي True فهذا يعني الكتابة على أكثر من سطر داخل الأداة TextBox .

الفصل الثالث
الحلقات التكرارية والمؤقتات



- تستخدم الحلقات التكرارية في تكرار كود محدد لعدد من المرات .

أولاً : استخدام الجملة For ... Next :

- تستخدم هذه الجملة في حالة **معرفة عدد مرات التكرار مسبقاً** .

*** الصيغة العامة :**

```
For Variable=Start To End [Step n]
    Code
Next [Variable]
```

- تبدأ الحلقة التكرارية بكلمة (For) وتنتهي بكلمة (Next) ، ويحدد مع (For) اسم متغير يطلق عليه العداد (Counter) له قيمة بداية (Start) وله قيمة نهاية (End) ، ويتم تكرار الكود (Code) داخل الحلقة حتى الوصول إلى قيمة النهاية .
- من الصيغة العامة تم وضع بعض الكلمات بين قوسين بهذا الشكل [ ] وهذا للدلالة على أنها اختيارية ، أي يمكن عدم كتابتها ، ولكن يجب معرفة أنه في حالة عدم كتابتها تأخذ القيمة الافتراضية لها . فمثلاً عند عدم كتابة (Step) تكون قيمتها ١ ، وعدم كتابة (Variable) مع (Next) تكون بنفس اسم المتغير المحدد مع (For) .

مثال ١):

```
For I = 1 To 10
    MsgBox( I )
Next
```

في هذا المثال سيقوم البرنامج بإظهار صناديق رسائل تحتوي على الأعداد (١ ، ٢ ، ٣ ، ٤ ، ٥ ، ٦ ، ٧ ، ٨ ، ٩ ، ١٠) بالترتيب .
وهنا يجب ملاحظة أن مقدار الزيادة ستكون ١ ، وذلك لعدم كتابة قيمة لـ Step .

مثال ٢) :

```
For I = 1 To 10  Step  2
    MsgBox( I )
Next
```

في هذا المثال سيقوم البرنامج بإظهار صناديق رسائل تحتوي على الأعداد الفردية (١ ، ٣ ، ٥ ، ٧ ، ٩) بالترتيب .
وهنا يجب ملاحظة أن بداية المتغير I ستكون بـ ١ ، ومقدار الزيادة ٢ في كل مرة .

مثال ٣):

```
For I = 2 To 10   Step  2
      MsgBox( I )
Next
```

في هذا المثال سيقوم البرنامج بإظهار صناديق رسائل تحتوي على الأعداد الزوجية (٢ ، ٤ ، ٦ ، ٨ ، ١٠) بالترتيب .
وهنا يجب ملاحظة أن بداية المتغير I ستكون بـ ٢ ، ومقدار الزيادة ٢ في كل مرة .

مثال ٤):

```
For I = 5 To 30   Step  5
      MsgBox( I )
Next
```

في هذا المثال سيقوم البرنامج بإظهار صناديق رسائل تحتوي على الأعداد (٥ ، ١٠ ، ١٥ ، ٢٠ ، ٢٥ ، ٣٠) بالترتيب .
وهنا يجب ملاحظة أن بداية المتغير I ستكون بـ ٥ ، ومقدار الزيادة ٥ في كل مرة .

مثال ٥):

```
For I = 10 To 1   Step  -1
      MsgBox( I )
Next
```

في هذا المثال سيقوم البرنامج بإظهار صناديق رسائل تحتوي على الأعداد (١٠ ، ٩ ، ٨ ، ٧ ، ٦ ، ٥ ، ٤ ، ٣ ، ٢ ، ١) بالترتيب .
وهنا يجب ملاحظة أنه إذا كانت قيمة بداية المتغير أكبر من قيمة النهاية ، فيجب أن تكون قيمة Step بالسالب .

مثال ٦):

```
For I = 1 To 5   Step  0.5
      MsgBox( I )
Next
```

في هذا المثال سيقوم البرنامج بإظهار صناديق رسائل تحتوي على الأعداد (١ ، ١,٥ ، ٢ ، ٢,٥ ، ٣ ، ٣,٥ ، ٤ ، ٤,٥ ، ٥) بالترتيب .
وهنا يجب أيضاً ملاحظة أنه يمكن أن تكون مقدار الزيادة قيمة عشرية .

ثانياً : استخدام الجملة Do While … Loop :

- تستخدم هذه الجملة لتكرار كود معين **لعدد من المرات غير معروف مسبقاً** بناءاً على شرط معين .

*** الصيغة العامة :**

```
Do While Condition
   Code
Loop
```

- ملاحظة هامة : هنا في جملة Do While … Loop يتم اختبار الشرط (Condition) فإذا كان متحققاً (أي كان ناتج الشرط True) يتم تنفيذ الأوامر التي تلي الشرط. أما في حالة عدم تحقق الشرط (أي كان ناتج الشرط False) فيتم تنفيذ الأوامر التي تلي Loop .
- وهذا يعني أن الكود سيظل يتكرر طالما كان ناتج الشرط صواب True .

مثال ١):

```
Dim X As Byte
X = 5
Do while X <= 7
  MsgBox (X)
X=X+1
Loop
```

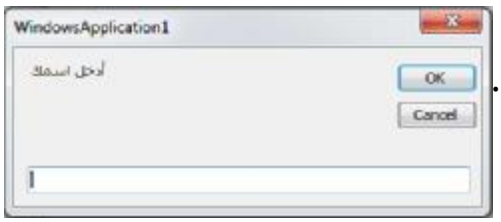
- وفي هذا المثال سيتم إظهار ٣ صناديق رسائل متتالية تحتوي على الأعداد (٥) ، (٦) ، (٧) بالترتيب

مثال ٢):

```
Dim X As Byte
X = 5
Do while X <= 7
X=X+1
Loop
  MsgBox (X)
```

- وفي هذا المثال سيتم إظهار صندوق رسالة يحتوي على العدد (٨) . **لماذا (٨) ؟ (فكر)**

*** ملاحظة : الدالة (InputBox)** وظيفتها إظهار صندوق لاستقبال قيمة من مستخدم البرنامج .



*** استخدام الأداة Timer :**

- تقع الأحداث (Events) عندما يقوم مستعمل البرنامج بعمل ما ، كالضغط على زر الـ Button. ولكن في بعض الأحيان نكون في حاجة لأن نجعل بعض الأحداث تقع بعد فترة زمنية معينة وبدون تدخل من مستعمل البرنامج ... وهذا نستطيع عمله باستخدام الكائن Timer والحدث Tick . **لذا يكون استخدام الـ Timer مفيداً جداً عندما يكون تكرار الكود مرتبطاً بالوقت** .
- **والكائن Timer هو عبارة عن ساعة توقيت غير مرئية** تتيح لك التعامل مع ساعة الحاسب.
- **الحدث الافتراضي للأداة Timer هو الحدث (Tick)** ويعني مرور الفترة الزمنية المحددة في الخاصية (Interval) .

*** خصائص الأداة Timer :**

الخاصية	القيمة الافتراضية	الوظيفة
Enabled	**False**	وتعني **تمكين** عمل المؤقت بجعلها (**True**) ، أو **إيقاف عمله** بجعلها (**False**) ، ويمكن ضبطها برمجياً من خلال الكود .
Interval	**100**	ووظيفتها التحكم في (تحديد) الفترات الزمنية لنشاط الـ Timer وتحدد هذه الفترة الزمنية (**بالمللي ثانية**) ، **حيث الثانية الواحدة تساوي ١٠٠٠ مللي ثانية** .

مثال١) : **الكود اللازم كتابته لإظهار التاريخ والوقت** داخل مربع العنوان Label1 يكون كالتالي :

```
Private Sub Timer1_Tick (ByVal ……..
    Label1.text = Now ( )
End Sub
```

مثال٢) : **الكود اللازم لعرض الوقت فقط** داخل مربع العنوان Label1 يكون كالتالي :

```
Private Sub Timer1_Tick (ByVal ……..
    Label1.text = TimeOf Day ( )
End Sub
```

مثال٣) : **الكود اللازم لإيقاف تشغيل المؤقت (Timer)** أثناء البرمجة يكون كالتالي :

```
Private Sub Timer1_Tick (ByVal ……..
    Timer1.Enabled = False
End Sub
```

الفصل الرابع
الإجراءات

* **الإجراء Procedure :** هو مجموعة من الأوامر والتعليمات تحت اسم ما (أي يتم تسمية مجموعة الأوامر بإسم ما) ، عند استدعاء هذا الاسم يتم تنفيذ هذه الأوامر والتعليمات .

- **وهذه الإجراءات إما أن تكون :**
أ) **Sub** : لا تعود بقيمة . **أو** ب) **Function** : تعود بقيمة .

أولاً : الإعلان عن الإجراء Sub :

* نلجأ للإعلان عن الإجراء من النوع (Sub) في حالة إذا ما كان لدينا كود سيتكرر كتابته في أكثر من موضع داخل التصنيف (Class) بهدف تنظيم كتابة الكود وتسهيل قراءته وفهمه .

*** الصيغة العامة للإعلان عن الإجراء من النوع (Sub) :**

```
Sub Name(Parameter)
    Code
End Sub
```

*** حيث أن :**
١- (Name) : يعبر عن اسم الإجراء .
٢- (Parameter) : عبارة عن القيم التي سوف تستقبل عند استدعاء الإجراء .
٣- (Code) : مجموعة الأوامر والتعليمات التي ستنفذ عند استدعاء الإجراء (Sub) .

*** ملاحظات :**
- يتم الإعلان عن الإجراء من النوع (Sub) مرة واحدة فقط .
- يمكن استدعاء الإجراء من النوع (Sub) أي عدد من المرات .
- يمكن عند الإعلان عن الإجراء من النوع (Sub) استخدام أكثر من وسيط (Parameter) .
- تستخدم الوسائط (Parameters) كوسيلة استقبال قيم من خارج الإجراء غير معلومة مسبقاً ، ويتم تحديد هذه القيم عند استدعاء هذا الإجراء .
- ينتهي الإجراء من النوع (Sub) بكلمة (End Sub) .
- الإجراء من النوع (Sub) لا يجوز استخدامها في أي جملة تخصيص .
- إجراء الحدث يعتبر إجراء من النوع (Sub) .
- وسيط الإجراء إما أن يكون قيمة مجردة ، أو متغير أو ثابت أو دالة .

ثانياً : الإعلان عن الدالة Function :

* نلجأ للإعلان عن (Function) في حالة إذا ما كان لدينا كود سينتج منه قيمة نحتاجها .

*** الصيغة العامة للإعلان عن الدالة Function :**

```
Function Name (Parameters) As Data Type
    Code
Return Value
End Function
```


*** حيث أن :**

١- (Name) : تعبر عن اسم الدالة (Function) .

٢- (Data Type) : تحدد نوع البيان الخاص بالقيمة الراجعة من الدالة (Function) .

٣- (Parameters) : تمثل الوسائط التي سوف تستخدم في الكود .

٤- (Code) : مجموعة الأوامر والتعليمات التي ستنفذ عند استدعاء الدالة (Function) .

٥- (Value) : هي القيمة الراجعة من الدالة (Function) .

*** ملاحظات :**

- الدوال (Function) : لا يمكن تخصيص قيم لها ، وإنما تُستدعى فتنتج قيمة تخزن بها .
- جميع الدوال (Function) تستخدم في الطرف الأيمن من معادلة التخصيص .
- أي دالة (Function) لا بد وأن يكون لها ناتج .
- هناك دوال لا تأخذ وسائط مثل الدالة Now( ) .
- يفضل تسمية الدوال (Function) تسمية متعلقة بوظيفتها .

* الدوال المعرفة :

- الدوال المعرفة هي دوال معروفة لدى لغة الفيجوال بيزك يتم استدعاؤها مباشرة دون الحاجة للإعلان عنها .

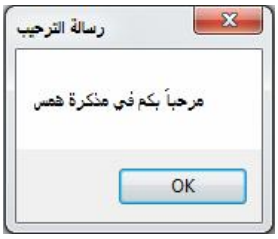
- من أنواع الدوال المعرفة ما يلي :

١- الدالة (Show) :

- من خلالها يمكن إظهار صندوق رسالةMessageBox يتحدد محتوياته حسب الوسائط المعطاة.

*** مثال :**

MessageBox.Show("همس في مذكرة مرحباً بكم" , "رسالة الترحيب")



٢- الدالة (IsNumeric) :

- يمكن من خلالها اختبار قيمة ، هل هي رقمية أم لا ، فإذا كانت القيمة رقمية فإن الدالة IsNumeric تعطي ناتج (True) ، وإذا كانت القيمة غير رقمية فإن الناتج يكون False .

*** أمثلة :**

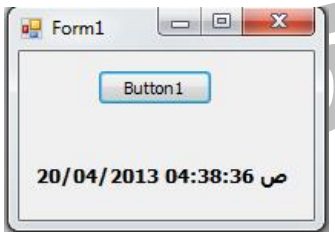
Label1.text = IsNumeric(5) ..... يكون الناتج ..... (True)
Label1.text = IsNumeric("Five") ..... يكون الناتج ..... (False)

٣- الدالة (Now) :

- يمكن من خلالها عرض التاريخ والوقت الحاليين المخزنين في نظام الكمبيوتر .

*** مثال :**

Label1.text = Now( )



- ملحوظة : الدالة (**TimeOf Day**) تستخدم لعرض الوقت فقط .