Written & Published by Prelware Association

In the name of Allah

بسم الله الرحمن الرحيم

Learn The Basics In Visual Basic 6.0

بسم الله الرحمن الرحيم

نسخة مهداة إلى موقع الفريق العربي للبرمجة مجانية للجميع

كتاب تعليم الأساسيات في فيجوال بيسك 6.0/ الفصل الأول:

محتوى الفصل الأول من الكتاب: مقدمة عن لغات البرمجة, والمصطلحات الأساسية المستخدمة والمتغيرات و الثوابت و خصائص العناصر وصفاتها و تعاريف شائعة وأساليب التعامل مع vb6....

محتوى الفصل الثاني من الكتاب: شرح عن بنى التحكم و كيفية استخدامها...

القسم الأول

Written By Doried-a-a

تأليف وإعداد : دريد عبد الله

مقدمة عن لغات البرمجة والمصطلحات الأساسية المستخدمة:

مُحَرَّمٌ ذُوالقِعْدَة مُحَرَّمٌ مقدمة عن الحاسب وبرمجته:

كما نعلم: الحاسب هو إحدى الآلات القابلة للبرمجة,ويتم برمجة الحاسب بما يسمى لغات البرمجة,وهناك ثلاثة أنواع للغات البرمجة.

مُحَرَّمٌ ذُوالقِعْدَة مُحَرَّمٌ ذُوالقِعْدَة مُحَرَّمٌ أنواع لغات البرمجة:

النوع الأول: لغات الآلة:

هي اللغة التي يفهمها الحاسوب مباشرة,ويمكن تعريف هذه اللغة على أنها اللغة المعرفة من قبل البنية الصلبة للحاسب.تتمثل لغة الآلة برموز قد تكون أصفار أو واحدات وتشكل أوامر يفهمها الحاسوب.وتعتبر البرمجة بلغات الآلة صعبة جدا وتستغرق الكثير من الوقت.

النوع الثاني: لغات المجمع:

هي عبارة عن مترجمات للبرامج,حيث يكتب البرنامج بلغة برمجة قد تكون قريبة من الإنكليزية و تقوم هذه المجمعات بترجمتها إلى لغة الآلة,وطبعا كانت لغة المجمع أبسط وأوضح بكثير من لغة الآلة,لكنها بقيت صعبة قليلاً.

النوع الثالث: لغات البرمجة عالية المستوى:

وهي لغات برمجة سهلة جدا مقارنة بلغة الآلة و لغة المجمع,ويقبلها العقل البشري ويفهمها أكثر من اللغتين السابقتين,تسمى البرامج التي تترجم النصوص البرمجية من لغات البرمجة عالية المستوى إلى لغة الآلة بالمترجمات Compilers.وطبعاً تعد لغة الفيجوال بيسك 6.0 من لغات البرمجة عالية المستوى,بالإضافة إلى الكثير من اللغات الأخرى مثل C++ و C وغيرها..

2-1 مقدمة عن الفيجوال بيسك 6.0:

إن لغة الفيجوال بيسك بنظري لغة برمجة سهلة جدا مقارنة مع لغات أخرى,وكذلك هي من أقوى لغات البرمجة,ولا تحتاج إلى أكواد كبيرة مثل بقية اللغات,وهي كما ذكرت من لغات البرمجة عالية المستوى.

عندما تفتح مترجم الفيجوال بيسك قد يطلب منك اختيار نوع المشروع الذي تريده, نحن في هذه الكتاب سنتعامل مع النوع Standard EXE.
سوف ترى إن شاء الله خصائص هذا النوع واختلافه عن بقية الأنواع.

1-2-1 بيئة التطوير في فيجوال بيسك: من مكونات بيئة التطوير في VB6.0:

لديك في فيجوال بيسك مربع يدعى **مربع أو شريط أدوات التحكم** شكل 1-1,يحتوي هذا المربع على بعض أدوات التحكم التي سيرد ذكرها, تستخدم أدوات التحكم لتكوين البرنامج,فالزر هو أداة تحكم,كذلك مربع النص والتسمية و مربع الصورة وخانة الاختيار وغيرها..

شكل 1-1 مربع أدوات التحكم

لعرض مربع أدوات التحكم,انقر على القائمة View ثم انقر على الأمر Toolbox.

كما تحتوي بيئة التطوير أيضا على **نافذة الخصائص**, سيرد ذكر معنى الخصائص في هذا القسم إن شاء الله.تساعدك نافذة الخصائص على ضبط خصائص عنصر تحكم أثناء التصميم, دون الحاجة إلى كتابة كود ليتم ضبطه أثناء تنفيذ البرنامج Run-time. سيرد شرح معاني هذه المصطلحات كلها إن شاء الله.لعرض نافذة الخصائص إن لم تكن موجودة اضغط على الزر F4 أو انقر على القائمة View ثم على Properties Window .

لدينا أيضاً **شريط الأدوات القياسي** والذي يحوي عدة أوامر كفتح ملف و حفظ ملف و تشغيل البرنامج لتجربته و محرر القوائم(ويساعدك في إنشاء قوائم كما سترى لاحقا) وغيرها من الأوامر..

لدينا أيضا **مستعرض المشروع**,وهو يعرض لك العناصر المكونة للمشروع من فورمات قياسية وغيرها و modules وغيرها..لعرض هذه النافذة انقر على القائمة View ثم على Project explorer.ولفتح عنصر ما من مستعرض المشروع قم بالنقر المزدوج عليه.

3-1 مصطلحات برمجية أساسية وهامة:

سوف تمر معنا بعد المصطلحات التي سيرد شرحها في هذه الفقرة, إن كنت مبتدأ في مجال البرمجة, فدقق في هذه الفقرة, وافهمها جيداً.

مصطلحات أساسية مستخدمة:

هناك الكثير من المصطلحات في فيجوال بيسك,نذكر منها:

- **الكود(الشيفرة البرمجية) Code:** وهي عبارة عن الأوامر التي تعطى للكمبيوتر على شكل نص,والتي يفسرها مترجم الفيجوال بيسك,ولكل فورم كود خاص به,ولكل وحدة نمطية كود خاص بها,وقد يكون الكود مقسما إلى كائنات Objects, دالات وتوابع Functions إجراءات Procedures...
- **الكائن Object:** قد يكون تعريف لكود أداة تحكم كزر الأمر(سيمر معنا), ويكون للكائن عدة إجراءات.
- **الإجراء Procedure:** هو عملية أو عدة عمليات ينفذها الكمبيوتر عند تحقق حدث معين, كالنقر على زر أو غيره..
- **الحدث Event:** هو عبارة عن فعل يقوم به المستخدم,كالنقر على زر أمر, أو تغيير نص في مربع نص أو حتى تحريك الماوس....ويرافقه تنفيذ الإجراء المتعلق به.

1-3-1 المتغيرات و الثوابت:

من الطبيعي أن يحتاج أي برنامج لتخزين بيانات بشكل مؤقت, وبالإمكان تخزين البيانات في ملفات,واستعادتها, ولكن هذه الطريقة ببساطة هي طريقة بطيئة و قد تؤثر على أداء الكمبيوتر, لذلك كان هناك ما يسمى بالمتغيرات والثوابت.

ملاحظة: يمكنك في بعض الأحيان تخزين البيانات في أحد الأماكن من الذاكرة دون الحاجة إلى تعريف متغير(عندما يكون مجال الرؤية على مستوى إجراء مثلا), وعندها يكون نوع البيانات شامل(يمكن للنوع String أن يكون شامل لأغلب الأنواع(ستعرف أنواع المتغيرات وطرق تخزينها في هذه الفقرة))

المتغيرات: هي عبارة عن مواقع في الذاكرة تخزن فيها بيانات بشكل مؤقت وتمحى عندما لا يعود هناك حاجة إليها, ويمكن تغيير قيمتها أثناء تنفيذ البرنامج, مثلا يمكن لبرنامج أن يخزن في المتغير Var1 القيمة 1 ثم يخزن القيمة 2, والقيمة التي تسترجع من المتغير هي آخر قيمة تم تخزينها فيه, وللمتغيرات أنواع كما سترى.وستتعلم كيفية تخزين البيانات في متغيرات بعد قليل.

الثوابت: هي عبارة عن مواقع في الذاكرة تخزن فيها بيانات بشكل مؤقت وتمحى عندما لا يعود هناك حاجة إليها, ولا يمكن تغيير قيمتها أثناء تنفيذ البرنامج, مثلا يمكن لبرنامج أن يخزن في المتغير ؤCon1 القيمة 1 لكنه لا يستطيع تغييرها, وإن حاولت تغييرها سيعطيك المترجم رسالة خطأ أثناء تنفيذ البرنامج. والقيمة التي تسترجع من الثابت هي القيمة التي أعطيت له أثناء تعريفه, وللثوابت أيضا أنواع كما سترى.وستتعلم كيفية تخزين البيانات في الثوابت بعد قليل.

تعريف المتغيرات أو الثوابت وتخزين البيانات فيها:

لتعريف متغير تستخدم القاعدة التالية:

Dim VariableName As Type

الكلمات التي تحتها خط تكتب كما هي

مثال: Dim T1 As String

حيث VariableName هو اسم المتغير(ويمكنك استخدام إي اسم للمتغير شريطة أن لايكون كلمة مفتاحية أو معرفاً في فيجوال بيسك أو معرفا من قبلك مسبقا ضمن نفس مجال الرؤية).
و Type هو نوع البيانات التي ستخزن بالمتغير, ونوع البيانات يحدد ما هي البيانات المسموح لها بأن تخزن في هذا المتغير أرقام مثلا أو حروف..سترى أنواع البيانات واستخداماتها بعد قليل.

لتخزين البيانات في متغير تكتب ما يلي:

VariableName = Data
حيث VariableName هو اسم المتغير,أما Data فهي البيانات المراد تخزينها في المتغير.

ولتعريف ثابت وتخزين البيانات فيه تستخدم القاعدة التالية:

Const ConstantName As Type=Data

الكلمات التي تحتها خط تكتب كما هي

مثال: Const C1 As Integer=1

حيث ConstantName هو اسم الثابت(ويمكنك استخدام إي اسم للثابت شريطة أن لايكون كلمة مفتاحية أو معرفاً في فيجوال بيسك أو معرفا من قبلك مسبقا ضمن نفس مجال الرؤية).
و Type هو نوع البيانات التي ستخزن بالثابت, ونوع البيانات يحدد ما هي البيانات المسموح لها بأن تخزن في هذا الثابت: أرقام مثلا أو حروف..سترى أنواع البيانات واستخداماتها بعد قليل.
كما مر معنا,لا يمكن تغيير البيانات في ثابت.

مصطلح مستخدم في هذه الفقرة:
تذييل البيانات: هو إضافة البيانات الجديدة بعد نهاية البيانات المخزنة مسبقا,دون عمليات جمع أو استبدال أو غيرها, وإشارة التذييل هي &,مثلا:

Data1=45
Data1=Data1 & 5
فتكون Data1=455
أما عملية الجمع + تؤدي إلى جمع البيانات,مثال:
Data1=45
Data1=Data1 + 5
فتكون Data1=50
ويمكن تذييل البيانات سواء أكانت أعداد أو حروف.

2-3-1 شرح عن أهم أنواع البيات في Visual Basic 6.0:

إليك جدول ببعض أنواع البيانات:

نوع المتغير	Integer	String	Double	Boolean
نوع البيانات	أعداد صحيحة	محارف	أعداد ذات فاصلة	True\False
أمثلة	56-25-10-9	Prelware1	5.544-1.6-12	True-False

جدول 1

-التعامل مع النوع String:

- إن البيانات الموضوعة بين علامتي اقتباس,تدل على النوع String أي على النص,لكن إذا تم إسناد قيمة عددية موضوعة بين علامتي اقتباس إلى متغير من النوع العددي Integer Or Double فإنه عادة يتم تجاهل علامتي الإقتباس.
- عند إضافة بيانات إلى متغير من النوع **String** فيتم إضافة البيانات إلى يمين البيانات الحالية(تذييل البيانات) حتى لو استخدمت إشارة الجمع +, لتوضيح الفكرة,نأخذ مثال:

```
Dim S1 As String
S1="Doried "
S1=S1 + "Abd-Allah"
```

فإن قيمة المتغير S1 النهائية تكون "Doried Abd-Allah"
- ملاحظة: يمكننا الإستغناء عن الإشارة + بالإشارة & لتذييل البيانات,سواءً في تذييل البيانات العددية أو الحرفية,لكننا استخدمنا الإشارة + في المثال السابق لنريك عملها مع الأنواع String.
على الرغم من ذلك نحن ننهي عن استخدام الإشارة + لتذييل البينات حتى لو كان تعاملنا مع النوع String ,فهناك حالات خاصة تؤدي إلى الحصول على نتائج غير متوقعة(كما سترى في هذه الفقرة بمقارنة المثالين 1-3-1 و 1-3-2).

- يمكن للنوع String الخاص بتمثيل المحارف بشكل عام أن يخزن فيه بيانات من النوع Integer أو حتى Double ويمكنه إجراء عمليات حسابية باستخدامها إن لم تكن موضوعة بين علامتي اقتباس,مثلا :

```
Dim A1 as String
A1=50.6
A1=A1+4
فتكون A1=54.6
```

مثال 1-3-1

```
Dim A1 as String
A1=50.6
A1=A1+"4"
فتكون A1=50.64
```

مثال 1-3-2

نلاحظ أن في المثال 1-3-1 كانت قيمة A1 النهائية تساوي 54.6 فقد تم اعتبار قيمة A1 من النوع Integer لخلوها من الحروف وتم جمع القيمة 4 معها, لأن العدد 4 مجرد من علامات الإقتباس التي تميز النوع String.
أما في المثال 1-3-2 كانت قيمة A1 النهائية تساوي 50.64 فقد تم اعتبار قيمة A1 من النوع String وتم إضافة المحرف 4 إلى يمينها(تذييلها) لأن المحرف "4" موضوع بين علامتي الإقتباس اللتان تميزان النوع String.

-التعامل مع الأنواع Integer & Double :

-يمكن للنوع Integer أو النوع Double أن يقبل الأرقام حتى لو كانت ضمن علامتي اقتباس.
-يمكن للنوع String أن يحتوي أرقام وأن يخصص للأرقام,لكنه لايمكن للنوع المخصص للأرقام مثل Integer Or Double أن يحتوي حروف.
-عند إسناد(تخزين) قيمة تحتوي فاصلة عشرية(من النوع Double) إلى متغير من النوع Integer فإنه يتم تقريب القيمة إلى أقرب عدد صحيح ثم يتم تخزينها,مثال:

```
Dim A1 as integer
A1=24.3
Msgbox A1
```

فعندها تظهر رسالة تبين قيمة A1 وهي بعد التقريب تساوي 24.

- ملاحظة : الأمرMsgbox يفيد في إظهار رسالة, سيرد شرحها فيما بعد.
- **تنبيه..!**: لا تقم بجمع الأعداد باستخدام الإشارة(التعليمة) & لأن ذلك يؤدي إلى تذييل البيانات.

- التعامل مع النوع Boolean:

- إن المتغيرات والثوابت من النوع Boolean تختزن فيها إحدى القيمتينTrue أوFalse.

- بإمكاننا إسناد القيمة إلى المتغير أو الثابت من النوع Boolean على أساس قيمة من النوع String وذلك بكتابتها على الشكل "True" أو "False" بدون الإهتمام بحالة الأحرف.
- وبإمكاننا إسناد القيمة لها بمجرد كتابة كلمة True أو False بدون علامتي إقتباس, وهي الطرقة الأفضل.
- وبإمكاننا إسناد القيمة إليها بمجرد كتابة رقم حيث:
الرقم 0 يعني False
أما بقية الأرقام فتعني True.

مجالات رؤية المتغيرات والثوابت:

مجالات رؤية متغير أو ثابت هي الأماكن التي يمكن أن يستدعى منها هذا المتغير وتستعاد البيانات منه فيها, أي المتغير الذي يكون معرفا ضمن إجراء(ستعرف لاحقا مامعنى الإجراء),تنمحي بياناته عند الإنتقال إلى إجراء آخر,بالتالي لايمكن الإستفادة من قيمته إلا ضمن الإجراء المعرف به.
أما المتغيرات المعرفة في بداية الكود في الفورم قبل أي إجراء,فيمكن استدعائها والإستفادة منها من أي إجراء أو أي مكان في كود الفورم المعرفة به.
أما المتغيرات المعرفة في Module فيمكن الإستفادة منها من أي مكان في البرنامج.
ملاحظة: لايمكن تعريف متغيرين بنفس الاسم في نفس مجال الرؤية.

4-1 الكائنات في فيجوال بيسك:

سيمر معنا في هذا الكتاب بعض المصطلحات,منها الخصائص, والطرق أو الدوال و الكود,و الإجراء, ستفهم معانيها في هذه الفقرة إن شاء الله.

- أدوات التحكم:

هي الأجزاء المكونة للبرنامج,فكل برنامج يتكون من أدوات تحكم,كزر الأمر ومربع النص وغيرها...
ولكل أداة تحكم عدة عناصر خاصة بها وتحدد سلوكها و شكلها وتميزها عن غيرها.

مصطلح هام :

الفورم Form : هو عبارة عن إطار(نافذة),تكون أدوات التحكم موجودة بداخله, ومجموعة الفورمات بما تحويه من أدوات تحكم تشكل واجهة البرنامج, فمثلا, في برنامج الوورد مثلا انقر على القائمة تنسيق ثم على "خط" ستظهر نافذة ,هذه النافذة تدعر فورم وهذه النافذة تحتوي على(تبويبات وقوائم منسدلة و خانات اختيار وأزرار تحكم) وكلها عبارة عن أدوات تحكم Controls.

أهم أدوات التحكم وخصائصها:

- **أداة زر التحكم Command Button :** وهي إحدى الأدوات الرئيسية في البرامج,عند النقر على زر التحكم,يقوم بتنفيذ إجراء معين.
- **أداة مربع النص Text Box :** تستخدم للحصول على معلومات من المستخدم, تميزها الخاصة Text التي تستخدم لتحديد النص المكتوب فيها(سيرد ذكر كيفية استخدام الخواص).
- **أداة التسمية Label :** تستخدم لإظهار تسمية توضيحية لعنصر ما كمربع النص,أو تستخدم لإظهار نتيجة,تميزها الخاصة Caption.
- **أداة خانة الإختيار Check Box:** تستخدم عادة للحصول على قيمة من المستخدم,وهذه القيمة إما True أو False.خاصتها الأساسية هي Value وتأخذ 3 قيم.
- **صندوق الصورة Picture Box:** يستخدم أحيانا لإظهار صورة,كما يمكنه أن يتضمن أدوات تحكم أخرى.
- **الإطار Frame :** يمكنه أن يتضمن عناصر تحكم أخرى.
- **المؤقت Timer:** يقوم بتنفيذ الإجراء الخاص به كل مدة محددة, تحددها الخاصية Interval حيث كل 1000 = ثانية(أي تحدد المدة بأجزاء الثانية).
- **الصورة Image:** تستخدم لإظهار صورة,تميزها خاصتان: Stretch وتحدد فيما إذا كان يجب تكبير أو تصغير الصورة المحتواة ضمن أداة التحكم هذه بحيث تناسب حجم أداة التحكم.أما الخاصة الثانية فهي Picture,وتستخدم لتحديد الصورة(سيرد ذكر طرق تحديد الصورة أثناء تشغيل البرنامج).

طريقة الدخول إلى خاصية لتعديلها:

يمكنك أن تعدل بعض الخصائص أثناء تنفيذ البرنامج,وذلك من خلال القاعدة التالية:

ControlName.Property = Value

لكني أنصحك **بالقاعدة العامة**:

FormName.ControlName.Property = Value

قاعدة 3

حيث:

FormName هو اسم الفورم الذي يحتوي أداة التحكم التي تريد تغيير خصائصها(إن كانت أداة التحكم ضمن نفس الفورم الذي تريد تعديل خاصيتها منه, فاكتب كلمة Me بدلا من FormName,وإذا كانت أداة التحكم خارج الفورم الموجود فيه الكود فاكتب اسم الفورم الموجودة فيه أداة التحكم).

ControlName هي أداة التحكم التي تريد تغيير خصائصها.

Property هي الخاصية التي تريد تعديلها.

كل ماعليك هو كتابة المعلومات السابقة والفصل بينها بالنقطة(.).

لاتقلق إن لم تفهم بعض الأشياء,فمن خلال الأمثلة التي ستصادفها في هذا الكتاب,ستفهم كل شيء إن شاء الله.

أهم الخصائص الشائعة ووظائفها:

لديك في الفيجوال بيسك عدد من الخصائص المشتركة بين معظم أدوات التحكم,نذكر منها:

- **الخاصية Name :** وهي الخاصية الأساسية لأي أداة تحكم, تعرفها عند لغة البرمجة,وبها يتم تمييز أداة التحكم من غيرها من قبل لغة البرمجة أو المترجم,ويتم تعديل بقية خواص أداة التحكم بوساطتها,كذلك يتم الاعتماد على اسم أداة التحكم لتحديد الإجراءات المتعلقة بها وغيرها, أي هذه الخاصية بالنسبة لأداة تحكم مثل الاسم بالنسبة للإنسان,فمثلا : إذا أراد الأب من ابنه أن يطلب من أحد الأشخاص فعل شيء معين,فإنه يقول له: اذهب إلى **فلان** و قل له أن يفعل كذا, نلاحظ أن الأب استخدم اسم الشخص لكي يعلم ولده من هو الشخص المقصود.. ستفهم كل شيء من خلال الأمثلة إن شاء الله. تكون قيمة هذه الخاصية عبارة عن محارف شرط أن لا تبدأ برقم وأقصر حد هو 127 محرف, ولا يمكن تعديل هذه الخاصية من خلال الكود لأنها تكون للقراءة فقط أثناء تشغيل البرنامج.

- **الخاصية Caption:** تتوافر لأغلب أدوات التحكم التي تظهر للمستخدم والتي يتعامل المستخدم معها,مثل زر الأمر أو خانة الإختيار,وهي ترجع أو تحدد(تضبط) تسمية عنصر التحكم التي ستظهر للمستخدم.
- **الخاصية Enabled :** ترجع(ستعرف معنى كلمة ترجع) أو تحدد(تضبط) فيما إذا كانت أداة التحكم قابلة للاستخدام من قبل المستخدم أم لا, (لاحظ أنه أحيانا ترى زر أمر أو قائمة بلون رمادي, وإن نقرت عليها لا يحدث أي شيء).هذه الخاصية تأخذ قيمتان:
إما قيمة **True:** وتعني أن أداة التحكم قابلة للاستخدام(أي تنفيذ أحد الأحداث المتعلقة بها يتم تنفيذ الكود الخاص به).
أو قيمة **False:** وتعني أن أداة التحكم غير مفعلة,في هذه الحالة لا يحدث أي شيء عند النقر عليها أو تنفيذ أحد الأحداث المتعلقة بها(ستعرف معنى الأحداث فيما بعد إن شاء الله),وفي هذه الحالة يكون لون أداة التحكم رمادي.
- **الخاصية Visible:** ترجع أو تحدد ما إذا كانت أداة التحكم مرئية أو لا, أي هل تظهر هذه الأداة للمستخدم أم لا,قد تظن أنه لا داعي لاستخدام هذه الخاصية,لكنها خاصية هامة,وتأخذ هذه الخاصية إما قيمة **True** أي هذه الأداة مرئية أو القيمة **False** أي هذه الأداة غير مرئية.

الخاصية Back Color: وتحدد أو ترجع لون خلفية أداة تحكم,تتوفر هذه الخاصية لكثير من أدوات التحكم. تستطيع ضبطها أثناء تصميم البرنامج وذلك من نافذة الخصائص وتستطيع ضبطها من خلال الكود وذلك بأن تطبق قاعدة الدخول إلى خاصية لتعديلها(القاعدة 3), حيث تكون القيمة Value هي اللون, ويمكنك تمثيل اللون بعدة طرق:
الطريقة الأولى هي كتابته كرمز, فمثلا الرمز &H80000012 يعني اللون الأسود ,أما الرمز&H000000FF& وهو نفسه &HFF& يعني الأحمر...(يمكنك نسخ رمز اللون من نافذة الخصائص). أما الطريقة الثانية لتحديد اللون هي كتابته كرقم,فمثلا الرقم 255 يمثل اللون الأحمر أما الرقم 0 فيمثل اللون الأسود,بينما الرقم 16711680يمثل الأزرق...
أما الطريقة الثالثة تستخدم لتحديد الألوان الشائعة والمعرفة لدى لغة Visual Basic 6.0 وهي بكتابة اللون ككلمة حيث كل كلمة تمثل رقم يفهمه المترجم, فمثلا الكلمة vbRed للون الأحمر و vbBlue للون الأزرق..وهذه الطريقة تستخدم فقط في الكود.

الخواص الخاصة بموقع الأداة:

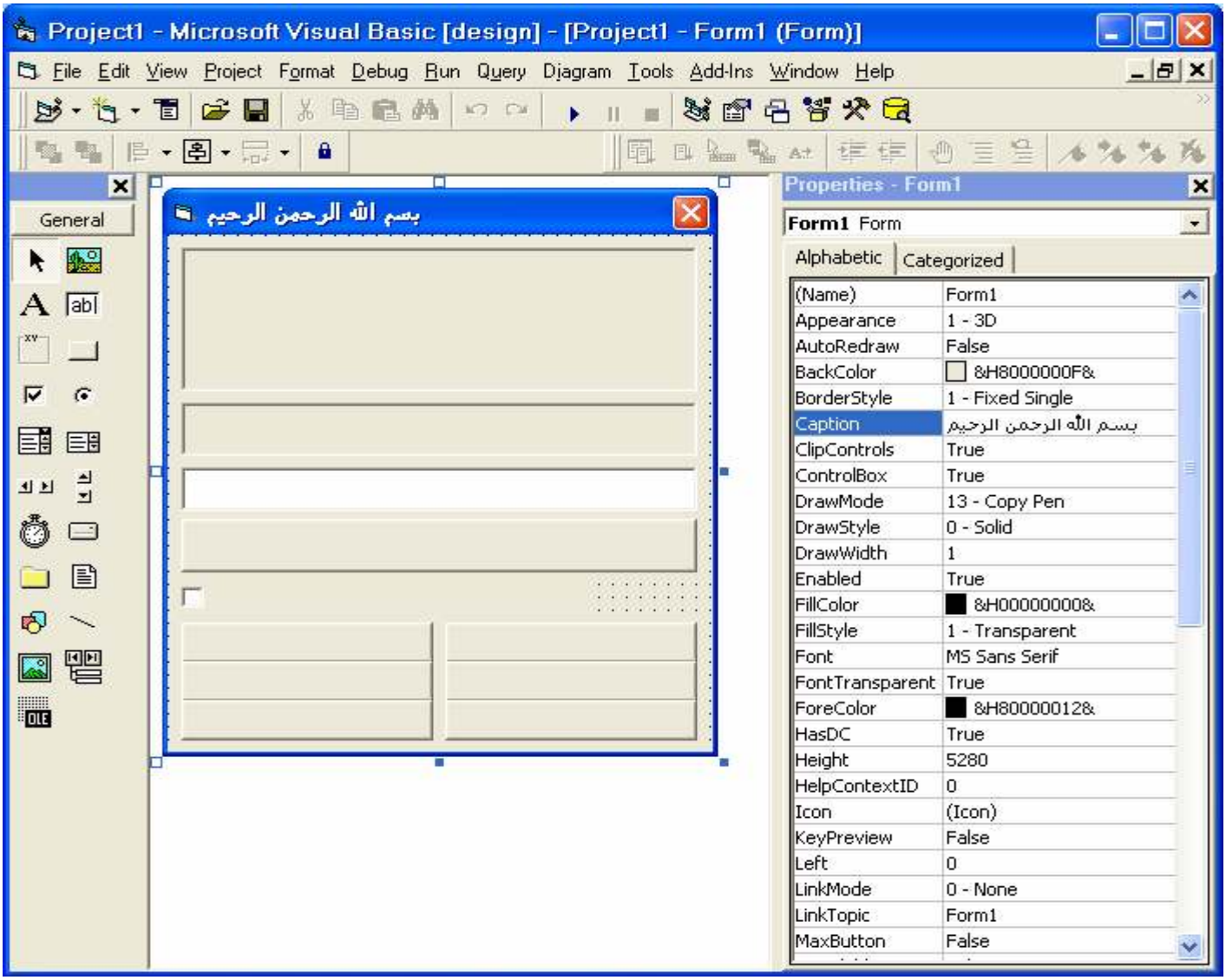
- **الخاصية Left:** ترجع أو تحدد بعد الأداة عن يسار الإطار الذي يحتويها (سواء أكان هذا الإطار Frame أو Form) مقدر بالبيكسل.

- **الخاصية Top:** ترجع أو تحدد بعد الأداة عن أعلى الإطار الذي يحتويها (سواء أكان هذا الإطار Frame أو Form) مقدر بالبيكسل.
- **الخاصية Width :** ترجع أو تحدد عرض الأداة مقدر بالبيكسل.
- **الخاصية Height :** ترجع أو تحدد ارتفاع الأداة مقدر بالبيكسل.

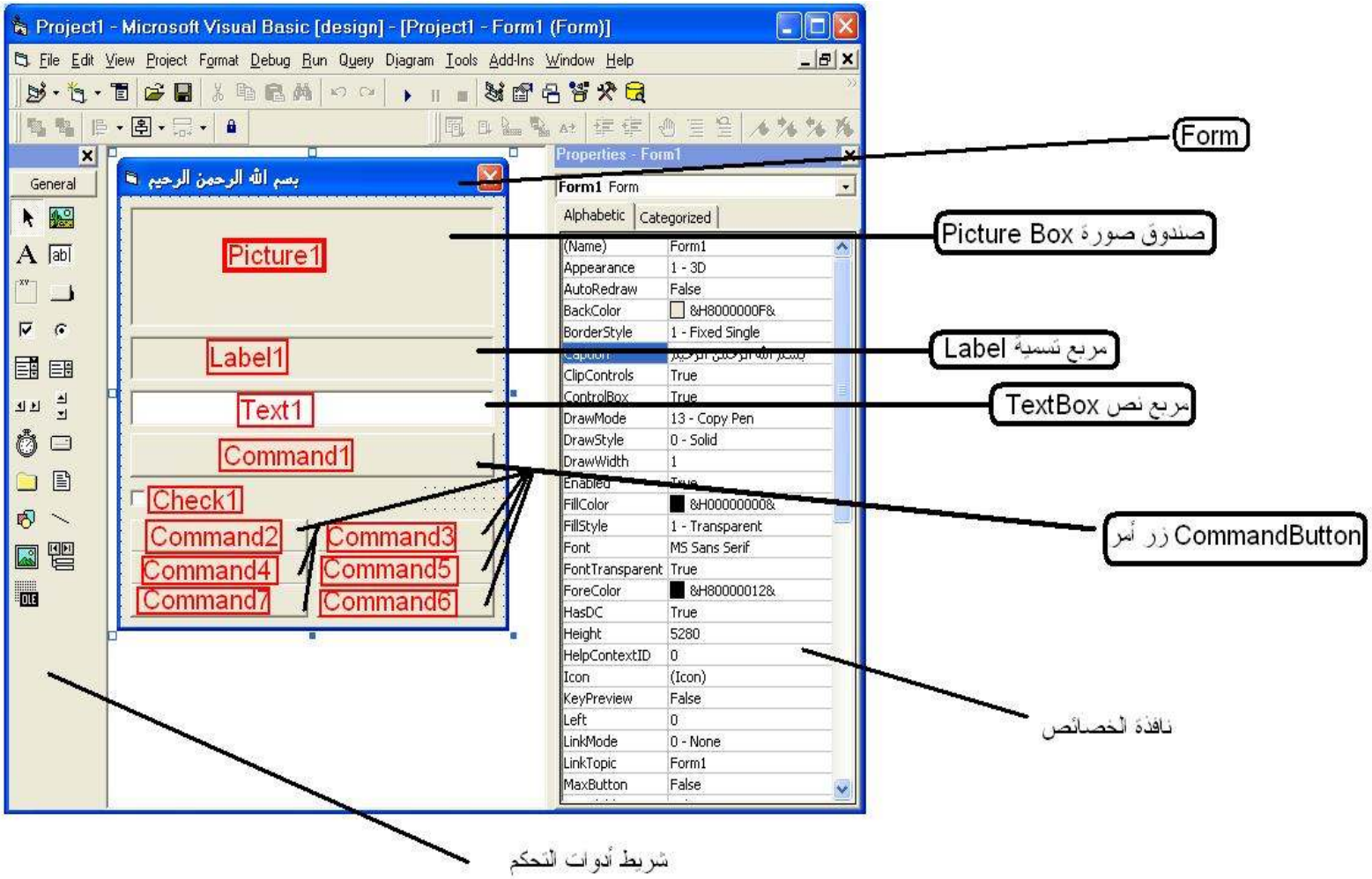
ليست هذه كل الخواص,إنما هناك خواص كثيرة ,سيمر بعضها مهنا في الكتاب من خلال الأمثلة,وسنشرح كل خاصة تمر معنا إن شاء الله.

مثال شامل للخواص المذكورة

مثال 1-4-1



لاحظ من خلال الشكل السابق بيئة التطوير في فيجوال بيسك.
سنقوم بإنشاء برنامج بسيط,مكون من عدة أزرار تحكم و مربع نص وأداة تسمية وصندوق صورة وخانة اختيار. انظر المسميات وتعرف على بيئة فيجوال بيسك في الشكل التالي لكي نشرع بإنشاء البرنامج:



لاحظ شريط أدوات التحكم,ونافذة الخصائص.
فلنبدأ بإنشاء البرنامج:
افتح برنامج الفيجوال بيسك, فتظهر رسالة تسألك عن نوع البرنامج الذي ترغب بإنشائه(إن لم تظهر: انقر على القائمة File ثم على(New Project),انقر على Standard EXE ثم على Ok.
قم بإضافة العناصر الموجودة في الصورة السابقة(_لقد قمت بتسمية نوع كل عنصر باللون الأسود_) من شريط أدوات التحكم(موجود في الصورة السابقة على يسار الشاشة).
ملاحظة: لإضافة عنصر تحكم قم بالنقر عليه في شريط أدوات التحكم ثم ارسمه على الفورم أو انقر عليه نقرة مزدوجة في شريط أدوات التحكم ثم اسحبه بعد أن يظهر في الفورم..
العناصر المطلوبة هي : 7 أزرار تحكم, 1 خانة اختيار,1 صندوق صورة, 1 مربع نص,1تسمية(Label).

بعد إضافة العناصر, ستتم تسميتها كما هي مسماة في الصورة السابقة باللون الأحمر,اعتمد على التسميات في الصورة السابقة لترتيب العناصر.
اختصار: في الصورة السابقة : اللون الأسود يحدد أنواع العناصر التي يحويها البرنامج الذي سننشئه, واللون الأحمر يحدد أسماء تلك العناصر ويساعدك في تنسيقها وترتيبها في الفورم.
بعد إنشاء العناصر وترتيبها,لا تهتم بالتسميات,انقر على القائمة View أو عرض ثم على Code,فتظهر نافذة الكود, إن كان بداخلها أي كود فامسحه, ثم اكتب الكود التالي:

```
Private Sub Command1_Click()
On Error Resume Next
Dim color As ColorConstants 'تعريف متغير باسم color
color = Text1.Text 'إسناد النص الموجود في مربع النص إلى المتغير color
Picture1.BackColor = color 'تغيير لون خلفية عنصر التحكم إلى القيمة المسندة بالمتحول
End Sub

Private Sub Command2_Click() 'لاحظ أننا استخدمنا اسم زر التحكم لتعيين الإجراء الخاص به
Check1.Value = Checked 'لاحظ أننا لم نستخدم اسم الفورم لأن أداة التحكم موجودة ضمن نفس الفورم
End Sub

Private Sub Command3_Click()
Check1.Value = Unchecked
End Sub

Private Sub Command4_Click()
Check1.Enabled = True
End Sub

Private Sub Command5_Click()
Check1.Enabled = False
End Sub

Private Sub Command6_Click()
Check1.Visible = False
End Sub

Private Sub Command7_Click()
Check1.Visible = True
End Sub

Private Sub Form_Load()
Text1.Text = ""
Label1.Caption = "اكتب رقما تريده في مربع النص ثم انقر على الزر لإظهار اللون الذي يمثله هذه الرقم"
Command1.Caption = " إظهار اللون الذي يمثله الرقم"
Command2.Caption = "تحديد خانة الإختيار"
Command3.Caption = "عدم تحديد خانة الإختيار"
Command4.Caption = "تفعيل خانة الإختيار"
Command5.Caption = "عدم تفعيل خانة الإختيار"
Command6.Caption = "إخفاء خانة الإختيار"
Command7.Caption = "إظهار خانة الإختيار"
Check1.Caption = "خانة اختيار تجريبية"
End Sub
```

ملاحظة: النص المكتوب بالأخضر هو عبارة عن ملاحظات وليس له علاقة بالكود,يمكن إضافة الملاحظات إلى الكود في فيجوال بيسك وذلك بأن تضع العلامة (') قبل الملاحظة,عندها يقوم المترجم بتجاهل كل ما هو مكتوب بعد تلك العلامة, وقد قمت أنا بوضع هذه العلامة في الكود السابق.
لاحظ من خلال المثال السابق,كيف استخدمنا لسم كل عنصر (أداة) تحكم في تحديد الإجراء الخاص بها,أو في ضبط قيمة إحدى خاصياتها..أي اسم عنصر التحكم ضروري جدا, لأنه مثلا في هذا البرنامج, هو الذي يميز الزر الأول عن الزر الثاني عن ....
لاحظ أنه عندما قمنا بتحديد الخصائص لم نطبق القاعدة العامة(لأننا لم نذكر اسم الفورم) وذلك لأننا قمنا بضبط خصائص عناصر التحكم من خلال كود الفورم الذي يحتويها(أي لأنها موجودة في نفس الفورم) ولكن لو أننا نريد ضبط خاصية لعنصر موجود في فورم آخر,كان يجب علينا ذكر اسم الفورم متبوع بنقطة قبل اسم عنصر التحكم.
فلنقم بتجريب البرنامج وذلك بالضغط على القائمة Run أو تشغيل ثم Start أو بالضغط على رمز المثلث الأزرق الصغير في شريط الأدوات القياسي.
يمكنك الحصول على ملف مكتوب بالفيجوال وذلك بفتح هذا الملف دون الحاجة إلى أي اتصال بإنترنت,لاتقلق فالملف آمن وهو يحوي المثال السابق جاهز:


المثال الأول.exe

الفصل الثاني

بنى التحكم:

مقدمة عن بنى التحكم:

محرم ذو القعدة جمادى أول

قمنا في المثال الأخير في الفصل الأول من هذا الكتاب بعرض كود بسيط جداً, ولم نستخدم في ذاك الكود أية بنية تحكم غير البنية التسلسلية.
يمكن القول أن بنية التحكم هي طريقة نقل التحكم من مكان إلى آخر في الكود فمثلا يمكنك أن تأمر البرنامج أن يفعل جزء معين من الكود في حال تحقق شرط معين وأن يفعل جزءاً آخرا منه في حال عدم تحققه,كما يمكنك أن تأمر البرنامج أن يستمر في تكرار تنفيذ جزء من الكود حتى يتحقق شرط معين أو طالما بقي شرط معين محققاً..ويمكنك أن تترك البرنامج ينفذ التعليمات بالتسلسل(تعليمة تعليمة) وهي بنية التحكم الافتراضية.

بنية التحكم التسلسلية:

في هذه الطريقة,يتم تنفيذ التعليمات تعليمة تعليمة, و**كمثال على ذلك:** افتح برنامج الفيجوال بيسك,أضف فورم,ضع فيه عنصر تحكم من النوع مربع النص واترك الخاصية Name له كما هي(Text1) ثم أضف زر تحكم(زر أمر) واترك الخاصية Name كما هي, بإمكانك تغيير تسمية الزر (الخاصية Caption) إلى "أظهر النص".
اكتب الكود التالي:

```
Private Sub Command1_click()
Text1.text = ""
Text1.Text = "Prel"
Text1.Text = Text1.text & "ware"
Text1.Text = Text1.text & " Association"
End Sub
```

مثال 2-1

لاحظ من خلال الكود السابق أنه تم في البداية جعل نص مربع النص يساوي "" أي لا شيء,ثم ينتقل البرنامج لتنفيذ الأمر الذي يليه وهو جعل النص "Prel" ثم يضيف للنص السابق كلمة "ware" ثم يضيف للنص السابق كلمة " Association" لاحظ أن التعليمات نفذت بشكل متسلسل مما جعل النص الموجود في مربع النص يكون "Prelware Association".

بنى التحكم الشرطية:

تقوم بنى التحكم الشرطية بتنفيذ أجزاء معينة من الكود حسب صحة شرط معين,سنتعرف على ذلك, وهناك عدة أنواع للبنى الشرطية.

عبارة If→Then:

تستخدم عبارة If Then لتنفيذ كود معين في حالة تحقق شرط معين, وإن لم يتحقق ذلك الشرط فإن البرنامج يقوم بتجاهل الكود المتعلق بكتلة الشرط.
والشكل العلم لعبارة If Then هو:

```
If Condition Then
  Statement1
  Statement2
  Statement..
End If
```

القاعدة 2-1

الكلمات التي تحتها خط تكتب كما هي
حيث Condition هو الشرط الواجب تحققه لتنفذ كتلة ال If.
أما Statement(s) هي التعليمة/التعليمات التي تنفذ في حال تحقق الشرط Condition.
أما Then فكل مل يليها يتم تنفيذه حتى نصل إلى عبارة End If(في حال تحقق الشرط).
أما End If فتدل على أن الكتلة الشرطية انتهت.
الكود المحصور بين عبارتي Then و End If هو الكود الذي يتم تنفيذه في حال تحقق الشرط.
في حال عدم تحقق الشرط,فأن التحكم ينتقل مباشرة إلى التعليمة التي تلي عبارة End if ويتم تنفيذها.
إذا: في البداية يقوم البرنامج بالتحقق من صحة الشرط,إذا كان الشرط محققا فإن التعبير الشرطي يأخذ القيمة True ثم يتم تنفيذ الكود الواقع بين كلمتي Then و End If وإن كان

غير محقق فإن البرنامج يقوم يتجاوز التعبير الشرطي ويقوم بتنفيذ التعليمات التي تلي عبارة End If.
ملاحظة: يسبب نسيان عبارة End if أو Then الوقوع في خطأ قواعدي.

عبارة If→Then→Else:

تستخدم عبارة If Then→Else لتنفيذ كود معين في حالة تحقق شرط معين, وإن لم يتحقق ذلك الشرط فإن البرنامج يقوم بتنفيذ كود آخر.
والشكل العلم لعبارة If Then→Else هو:

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

القاعدة 2-2

الكلمات التي تحتها خط تكتب كما هي
حيث Condition هو الشرط الواجب تحققه لتنفذ كتلة ال If.
أما Statement(s) هي التعليمة/التعليمات التي تنفذ في حال تحقق الشرط Condition.
أما Then فكـــل مل يليها يتم تنفيذه حتى نصل إلى عبارة Else(في حال تحقق الشرط).
أمـا Else فكـــل ما يليها يتم تنفيذه حتى نصـــل إلى عبارة End If (في حال عدم تحقق الشرط).
أما End If فتدل على أن الكتلة الشرطية انتهت.
الكود المحصور بين عبارتي Then و Else هو الكود الذي يتم تنفيذه في حال تحقق الشرط.
في حال عدم تحقق الشرط,فسوف يتم تنفيذ الكود المحصور بين عبارتي Else و End If.

إذا: في البداية يقوم البرنامج بالتحقق من صحة الشرط,إذا كان الشرط محققا فإن التعبير الشرطي يأخذ القيمة True ثم يتم تنفيذ الكود الواقع بين كلمتي Then و Else وإن كان

غير محقق فإن البرنامج يقوم يتجاوز الكود الواقع بين Then و Else و ينفذ الكود الواقع بين Else و End If.

ملاحظة: يسبب نسيان عبارة End if أو Then الوقوع في خطأ قواعـــدي.

ويسبب نسيان Else في تعبير شرطي من هذا النوع الوقوع في خطأ منطقي.

(الخـطأ المنطقي: هو خطأ لا يقوم البرنامج بكشفه, لأنه ليس خطأ إملائي أو قواعدي إنما هو كود صحيح لكن ينتج عنه نتائج غير النتائج المطلوبة من تنفيذه) لأنه في حـــــال عدم الفصل بين الكود الذي ينفذ عند تحقق الشرط والكود الذي ينفذ عند عدم تحقق الشرط بكلمة Else فسيتم اعتبار الكود الثاني جزءاً من الأول.

عبارة ..If→Then→Else→ElseIf:

تقوم هذه العبارة بعمل مشابه لعمل if→then→else→ إلا أنها تقوم باختبار تحقق عدة شروط,أي إن لم يحقق الشرط الأول تقوم باختبار الشرط الثاني ثم الثالث وهكذا حتى يتحقق أحد الشروط فتقوم بتنفيذ كوده وإن لم يتحقق أي من الشروط فإنها تقوم بتنفيذ الكود المتعلق بكلمة Else.

توضع كلمة Else والكود المتعلق في نهاية الكتلة الشرطية قبل End If.

عند تحقق الشرط الأول فإن البرنامج ينفذ الكود المحصور بين كلمة Then وكلمة ElseIf الأولى وفي حال تحقق شرط متعلق بإحدى كلمات ElseIf فإنه يتم تنفيذ الكود الواقع بينه و بين كلمة ElseIf التي تليه,وفي حال عدم وجود ElseIf بعده يتم تنفيذ الكود حتى كلمة Else وفي حال عدم وجودها أيضا يتم تنفيذ الكود حتى كلمة End If.

وإن لم يتحقق أي من الشروط فإنها تقوم بتنفيذ الكود المتعلق بكلمة Else.

أما الشكل العام لعبارة ..If→Then→Else→ElseIf فهو:

```
If Condition Then
  Statement1
  Statement..
ElseIf Condition2 Then
Statement1
Statement..
Else If Condition.. Then
...
End If
```

القاعدة 2-3

عبارة Select Case:

تستخدم هذه الطريقة في حال وجود عدة قيم واحتمالات لعبارة معينة واحدة.
مثلا يقوم المستخدم بإدخال رقم أصغر من 5 فيقوم الحاسوب بطباعة رسالة بحسب الرقم المدخل,فعوضا عن تكرار استخدام عبارات If أو ElseIf فإننا نقوم باستخدام Select Case.
نقوم بتحديد العبارة التي سترجع لنا القيمة وذلك بوضعها بعد عبارة Select Case ثم نقوم بوضع كل احتمال بعد كلمة Case وتحتها الكود المرتبط بهذا الاحتمال.
والشكل العام لـSelect Case هو:

```
Select Case Expression
 Case Value1
   Statement(s)
 Case Value2
   Statement(s)
 Case .........
 Case Else
   Statement(s)
End If
```

القاعدة 2-4

حيث أن Expression هي العبارة التي ستختبر قيمتها,مثلا Text1.text أو ...
أما Value(1,2,3,..) فهي قيم العبارة Expression.
وهذه بمثابة If Expression=ValueX Then Statement(s)
أي أن البرنامج يقوم بإرجاع قيمة Expression ويقوم باختبار ما إذا كانت تساوي إحدى القيم الموجودة بعد إحدى كلمات Case وفي حال كانت كذلك فإنه يقوم بتنفيذ الكـــود الذي يقع بين القيمـــة الموجودة يمين كلمة Case و كلمة Case التي تليها(أو End select أو Case Else).
في حال لم تتوافق واحدة أو أكثر من القيم المحددة بعد كلمات Case مع قيمة Expression فإن البرنامج يقوم بتنفيذ الكود المتعلق بكلمة Case Else.
ملاحظة : ليس من الضروري كتابة عبارة Case Else إلا في حال تطلب البرنامج ذلك.
يمكنك استخدام كلمة To مع بعض أنواع القيم كالأرقام مثلا,حيث يمكنك مثلا أن تكتب Case 2 To 5 بدلا من Case 2.....Case3.....Case4.....Case5..... .

مثال:

```
()Private Sub Form_Load
("s = InputBox("D
Select Case s
Case 1
MsgBox "واحد"
Case 2
MsgBox "اثنان"
Case 3 To 5
MsgBox "بين 3 و 5"
Case Is > 5
MsgBox "أكبر من 5"
Case Else
MsgBox "أصغر من 1"
End Select
End Sub
```

مثال 2-2

بنى التحكم التّكرارية:

تستخدم بنى التحكم التكرارية للاستمرار في تنفيذ كتلة معينة من الكود طالما بقي شرط محققا أو حتى تحقق شرط أو لعدد_ يحدده المستخدم_ من المرات.

عبارة While➔Wend:

تستخدم هذه الطريقة للاستمرار في تنفيذ كود معين و إعادة تنفيذه..طالما بقي الشرط المتعلق بها محققا, وفي حال عدم تحقق الشرط فإن التحكم ينتقل من البنية التكرارية إلى التعليمة التي تليها.

والشكل العلم لعبارة While➔Wend هو:

```
While Condition
  Statement1
  Statement2
  Statement..
Wend
```

القاعدة 2-5

مثال: إليك برنامج يقوم بإظهار رسالة للمستخدم 5 مرات ويكتب بالرسالة ترتيب التكرار:

أنشئ فورم واكتب فيه الكود التالي:

```
()Private Sub Form_Load
Dim Counter As Integer
Counter = 0

While Counter < 6
MsgBox Counter
Counter = Counter + 1
Wend
End Sub
```

مثال 2-3

لاحظ في المثال السابق:

استخدمنا المتحول الصحيح Counter.

بدأنا الحلقة التكرارية في السطر الرابع,ولاحظنا أن شرط تكرار الحلقة هو أن تكون قيمة المتحول Counter أصغر من 6.بالتالي عندما ستصبح قيمة هذا المتحول تساوي 6 سيخرج البرنامج من هذه الحلقة ويتوقف عن تكرارها.

لاحظ أنه عند تنفيذ الحلقة في كل مرة يتم طباعة رسالة تحتوي قيمة المتحول Counter وبعد ذلك يقوم البرنامج بإضافة 1 إلى هذا المتحول.

تسمى بنى التحكم التي تحوي على متحول من هذا النوع ويرتبط استمرار تكرار هذه البنى بهذا المتحول بـ(البنى التكرارية ذات العداد).

ملاحظة: يجب أن تحتوي البنية التكرارية من هذا اللنوع على شرط يقوم بإيقاف تنفيذهـــا عند تحققه ويقع بعد كلمة While وقد تستخدم الشرط كشرط داخلي وذلك بأن تضع الشرط داخل جسم الحلقة وتضع أمام كلمة While شرط غير متحقق(كأن تضع رقما غير 0).

والمثال التالي هو نفسه المثال السابق لكن بعد تعديله:

```
()Private Sub Form_Load
Dim Counter As Integer
Counter = 0
While 1
If Counter=6 Then Exit Sub
MsgBox Counter
Counter = Counter + 1
Wend
End Sub
```

مثال 2-4

لاحظ أن Exit Sub تؤدي للخروج من الإجراء كله ليس فقط من البنية التكرارية,بينما تتيح لك البنية التالية الخروج من بنية التكرار فقط.

عبارة Do➔Loop:

تقوم بنية التحكم هذه بالاستمرار في تكرار تنفيذ الكود المرافق لها.فبالتالي يجـــب على المبرمج أن يضع الشرط الذي يؤدي إلى إيقافها ضمن الكود المرفق بها,وذلك باستخدام التعليمة Exit Do.

يتم تكرار تنفيذ الكود المحصور بين Do وLoop,والقاعدة العامة هي:

```
Do
  Statement1 {May it is  Loop Exit Condition}
  Statement..
Loop
```

القاعدة 2-6

ولنأخذ المثال السابق بعد تعديله:

```
()Private Sub Form_Load
Dim Counter As Integer
Counter = 0
Do
If Counter=6 Then Exit Do
MsgBox Counter
Counter = Counter + 1
Loop
End Sub
```

مثال 2-5

لاحظ أنه في حال حذفنا السطر الخامس سيستمر البرنامج في إظهار هذه الرسالة,ذلك لأن الشرط الذي يوقف التكرار غير موجود.(سوف نتعلم بعد قليل طرق أخرى لتحديد شرط إيقاف التكرار).
لاحظ أن كلمة Exit Do تخرج فقط من البنية التكرارية ليس من الإجراء كله كما تفعل كلمة Exit Sub.وللبرهان على ذلك قم بكتابة المثال السابق كما يلي:

```
()Private Sub Form_Load
Dim Counter As Integer
Counter = 0
Do
If Counter=6 Then Exit Do
MsgBox Counter
Counter = Counter + 1
Loop
Msgbox "The Loop Ended"
End Sub
```

مثال 2-6
لاحظ أن البرنامج بعد أن انتهى من الحلقة التكرارية قام بإظهار الرسالة,مما يدل على أنه لم يخرج من الإجراء كله,بل خرج من الحلقة فقط, بينما لو أننا وضعنا الرسالة بعد الحلقة في المثال 1-4 لوجدنا أنها لن تنفذ,ذلك لأن التعليمة End Sub تخرج التحكم من الإجراء بكامله.
نتيجة:إذاً لابد من استخدام **Exit Do** أو غيرها مع بنى Do➔loop المجردة من While و Until.
وسنتعلم أشكال خاصة لعبارة Do➔Loop.

أشكال خاصة لعبارة Do➔Loop:

Do➔Loop While...:

تقوم هذه الطريقة بتكرار الكود المرافق لها طالما بقي الشطر الموجود على يمين While محققا, وعند عدم تحققه فإن البرنامج يخرج من البنية إلى التعليمة التي تليها.
أي عملها مشابه لعمل While➔Wend تقريباً.
القاعدة العامة:

```
Do
  Statement1
  Statement2
  Statement..
Loop While Condition
```

القاعدة 2-7

حيث أنه يتم تكرير تنفيذ الكود الموجود بين كلمتي Do و Loop ثم يتم اختبار الشرط, وتعاد الكرة حتى عدم تحقق الشرط Condition.ولنأخذ المثال التالي:

```
()Private Sub Form_Load
Dim Counter As Integer
Counter = 0
Do
MsgBox Counter
Counter = Counter + 1
Loop While Counter<=6
Msgbox "The Loop Ended"
End Sub
```

مثال 2-7

لاحظ أن البرنامج ظل يرسل الرسالة طالما كانت قيمة المتحول Counter أصغر أو تساوي 6.

لاحظ أننا استخدمنا في هذا المثال التعليمة =< وليس لها علاقة بالبنية,إنما الغـــرض هو تعريفك ببعض الإشارات الموجودة في الفيجوال بيسك.وتعني هذه التعليمة(الموجــودة في السطر السابع) أن شرط التكرار هو أن تكون قيمة Counter أصـغر من أو تساوي الستة وهو الشرط الذي يتوقف التكرار في حال عدم تحققه(أي عندما تصـــبح قيمة Counter تساوي 7).

لاحظ لو أنك كتبت الإشارة =< على الشكل <= سيصححها المترجم تلقائياً.

:Do➔Loop Until...

تعد هذه الطريقة معاكسة للطريقة السابقة , حيث أنها تقوم بتكرار الكود المرافق لها حتى يصبح الشرط الموجود على يمين While محققا, أي أن البرنامج سيستمر في تكرير تنفيذ التعليمات في حال عدم تحقق الشرط Condition.

أي عملها معاكس لعمل While➔Wend تقريباً.

القاعدة العامة:

```
Do
  Statement1
  Statement2
  Statement..
Loop Until Condition
```

القاعدة 2-8

حيث أنه يتم تكرير تنفيذ الكود الموجود بين كلمتي Do و Loop ثم يتم اختبار الشرط, وتعاد الكرة حتى تحقق الشرط Condition.ولنأخذ المثال التالي:

```
Private Sub Form_Load()
Dim Counter As Integer
Counter = 0
Do
MsgBox Counter
Counter = Counter + 1
Loop Until Counter>=7
Msgbox "The Loop Ended"
End Sub
```

مثال 2-8

لاحظ أن البرنامج ظل يرسل الرسالة عندما كانت قيمة المتحول Counter أصغر من 7,أما عندما أصبحت قيمة المتحول السابق تساوي 7 خرج البرنامج من الحلقة.
لاحظ أننا استخدمنا في هذا المثال التعليمة =< وليس لها علاقة بالبنية أيضا.وتعني هذه التعليمة(الموجودة في السطر السابع) أن شرط التوقف عن التكرار هو أن تكون قيمة Counter أكبر من أو تساوي السبعة وهو الشرط الذي يتوقف التكرار في حال تحققه(أي عندما تصبح قيمة Counter تساوي 7).
لاحظ لو أنك كتبت الإشارة =< على الشكل <=سيصححها المترجم تلقائياً أيضا,وهذه من مزايا مترجم الفيجوال بيسك 6.
كذلك لاحظ أنه يقوم بتصحيح حالة الأحرف, أي الكبيرة و الصغيرة,فعندما نكتب do يقوم المترجم بتحويلها إلى Do بمجرد انتقالنا إلى سطر آخر أو تشغيل البرنامج....

Do While ➔Loop:

يشبه عملها عمل ..Do➔Loop While إنما تختلف عنها بالصيغة وبشيء آخر,ألا وهو:

- في الصيغة Do➔Loop While يتم تنفيذ الكود(كود الحلقة أو كتلتها) ثم اختبار الشرط Condition.
- أما في الصيغة Loop➔Do While فإنه يتم اختبار الشرط ثم تنفيذ الكود.

والشكل العام كما يلي:

```
Do While Condition
  Statement1
  Statement2
  Statement..
Loop
```

القاعدة 2-9

مثال 1- 7 بعد تعديله:

```
Private Sub Form_Load()
Dim Counter As Integer
Counter = 0
Do While Counter<=6
MsgBox Counter
Counter = Counter + 1
Loop
Msgbox "The Loop Ended"
End Sub
```

مثال 2-9.

Do Until ➔Loop:

يشبه عملها عمل Do➔Loop Until.. إنما تختلف عنها بالصيغة وبشيء آخر,ألا وهو:

- في الصيغة Do➔Loop Until يتم تنفيذ الكود(كود الحلقة أو كتلتها) ثم اختبار الشرط Condition.
- أما في الصيغة Loop➔Do Until فإنه يتم اختبار الشرط ثم تنفيذ الكود.

والشكل العام كما يلي:

```
Do Until Condition
  Statement1
  Statement2
  Statement..
Loop
```

القاعدة 2-10

مثال 1- 8 بعد تعديله:

```
()Private Sub Form_Load
Dim Counter As Integer
Counter = 0
Do Until Counter>=7
MsgBox Counter
Counter = Counter + 1
Loop
Msgbox "The Loop Ended"
End Sub
```

مثال 2-10

استخدام العبارة For

تستخدم العبارة for عادة لتنفيذ كتلة معينة من الشيفرة(الكود) عدد(غالبا ما يكون محددا) من المرات, ويمكن القول أن بإمكاننا الاستعاضة عن البنية for ببنية while أو ..do.

عبارة For➔Next:

في هذه العبارة, يتم تنفيذ الكتلة المتعلقة بها,ثم تتم عملية إضافة مقدار معين(يكون افتراضيا 1) إلى متحول يمثل عداد,ويمكن تحديد مقدار التزايد.
لاتقلق إن لم تفهم ما ذكر,سنقوم بشرح هذه العبارة بشكل وافي, انظر القاعدة العامة لعبارة For➔Next

```
For CounterName = Start To End Step Increment
  Statement1
  Statement2
  Statement..
Next Counter
```

القاعدة 2-11

الكلمات التي تحتها خط في القاعدة تكتب كما هي
يعبر CounterName عن اسم المتحول الذي سنستخدمه كعداد للتكرار.
أما Start فهي القيمة التي يأخذها المتحول عند بداية التكرار.
و End هي القيمة التي تنهي التكرار, حيث أنه عندما تجتاز قيمة متحول التكرار Counter القيمة النهائية End يتم الخروج من بنية For وتنفيذ التعليمة التي تلي كلمة Next.

أما Increment فهي مقدار الزيادة في متحول التكرار Counter بعد كل مرة تنفذ فيها الكتلة المتعلقة بالبنية For (كما ذكرنا إن لم تذكر تكون 1 بشكل افتراضي).
أما Next فعندما يصل البرنامج إليها تتم إضافة Increment (مقدار التزايد في متحول التكرار) إلى متحول التكرار Counter.
يجدر القول بأنه يمكن أن تكون القيمة Increment قيمة سالبة, في هذه الحالة سوف يتناقص متحول التكرار بدلاً من تزايده, ويمكن استخدام هذه الطريقة,ولكن بهذه الحالة يجب أن تكون القيمة End أصغر من القيمة Start بحيث أن التكرار يبدأ بالقيمة Start ثم ينقص منها المقدار Increment (يقوم البرنامج بزيادة Increment إلى Counter لكن إذا كانت Increment سالبة فإنه يقوم بإنقاصها, فمثلا : 2 + (-1) = 1 وهي كلها عمليات رياضية) وعندما تصبح قيمة Counter أصغر من قيمة End فإن البرنامج يخرج من البنية For وينفذ التعليمة التي تليها.

دورة البنية For:

- يقوم البرنامج باختبار فيما إذا كانت قيمة Counter تساوي القيمة Start أو End أو واقعة بينهما وفي حال لم تكن كذلك فإنه يخرج من البنية وفي حال كانت كذلك فإن البرنامج ينفذ كتلة البنية For(الكود المحصور بين Increment وNext وقد لا تكون Increment موجودة عندها يتم تنفيذ الكود الواقع بين End وNext).
- عندما يصل البرنامج إلى كلمة Next يقوم بإضافة القيمة Increment (والتي قد تكون سالبة كما ذكرنا) إلى متحول التكرار Counter(أو المتحول المذكور بعدها).

تنبيه..! : لا تذكر اسم أي متحول بعد الكلمة Next في حالة استخدام بنى For المتداخلة (والتي ستتعرف عليها إن شاء الله) إلا إذا كان ذلك مطلوبا وضروريا.

مثال بسيط:

```
()Private Sub Form_Load
For i = 6 To 1 Step -1
" Msgbox إن قيمة متحول التكرار هي : " & i
Next i
End Sub
```

مثال 2-11
لاحظ أن قيمة متحول التكرار i كانت متناقصة,حيث أن البرنامج قام بجعل قيمته تساوي 6 ثم كان يعرض الرسالة التي تحوي قيمة المتحول i ثم يضيف له القيمة -1 (أي ينقص منه 1) ومن ثم يعود وينفذ الحلقة حتى تصبح قيمة المتحول i أصغر من 1 عندها يخرج من البنية For.

وفي حال أردنا جعل قيمة متحول التكرار متزايدة فإننا نكتب المثال السابق كما يلي:

```
Private Sub Form_Load()
For i = 1 To 6 Step 1
Msgbox " إن قيمة متحول التكرار هي : " & i
Next i
End Sub
```

مثال 2-12

لاحظ أنه ليس من الضروري ذكر قيمة التزايد في السطر الثاني لأنها 1 بالتالي ليس من الضروري ذكر الكلمة Step 1.

ولاحظ أيضا أنه ليس من الضروري ذكر اسم المتحول بعد الكلمة Next في السطر الرابع من مثالنا 1-12.

عبارة For Each➔Next:

تقوم هذه العبارة بتنفيذ عملية على مجموعة من العناصر المشتركة في صفة ما (كالمصفوفات التي ستتعلمها فيما بعد إن شاء الله) كمجموعة من عناصر التحكم التي تنتمي لنوع واحد و تملك نفس الاسم وتكون مصفوفة(هذا على سبيل المثال الحصر).

و في هذه الطريقة لا داعي لتحديد القيمة الأولية Start ولا النهائية End.

ولترى ذلك,قم بتنفيذ المثال التالي:

أنشئ فورم, وضع فيه مجموعة من عناصر التحكم(كالأزرارCommand Buttons و التسميات Labels و خانات الإختيار وغيرها..) ثم افتح نافذة الكود وذلك بالنقر المزدوج على أحد عناصر الفورم, وامسح الكود الموجود مسبقا, واكتب ما يلي:

```
Private Sub Form_Load()
On Error Resume Next
Dim a As Integer
a = 1
For Each Control In Form1
Control.Caption = "I'm Control " & a
a = a + 1
Next
End Sub
```

مثال 2-13

لاحظ أننا استخدمنا العبارة On Error Resume Next :

وهي تأمر البرنامج : إذا حدث خطأ فتجاهله و نفذ التعليمة التي تليه.
وهي ضرورية هنا, لأنه ليس من الضروري أن تتوفر الخاصية Caption لكل العناصر التي أضفتها إلى الفورم.
قمنا بتعريف متحول صحيح a وخزنّا فيه القيمة 1.
ثم استخدمنا البنية For Each➔Next:
حيث كان العنصر المراد تطبيق العملية عليه هو Control ويعني أي عنصر تحكم مهما كان,أما التعليمة المطلوب تنفيذها هي تعديل الخاصية Caption وجعل قيمتها كما هو مذكور,من ثم إضافة 1 لـ a.
الشكل العام لعبارة For Each➔Next هو:

```
For Each Element In Array
  Statement1
  Statement2
  Statement..
Next Counter
```

القاعدة 12-2

حيث أن Array تعني مصفوفة من العناصر أو ما شابه.
يستمر البرنامج بتكرار الحلقة حتى يطبق التعليمات الموجودة في كتلة الحلقة حتى يتم تطبيقها على جميع العناصر الموجودة في المصفوفة.

بنى التحكم المتداخلة

في كثير من الأحيان نحتاج إلى حلقة في كل مرة تنفذ فيها هذه الحلقة تنفذ حلقة أخــرى, مثلا نريد من البرنامج أن يكتب في مربع نص 6 سطور, في كل سطر يكتب 4 كلمــات فإننا في هذه الحالة نستخدم حلقة أولى(خارجية) تعمل 6 دورات و تقوم بإضافة سطر في كل مرة و حلقة ثانية تعمل 4 دورات وتقوم بإضافة كلمة في كل دورة ويجب أن تكون الحلقة الثانية(الداخلية) ضمن كتلة الحلقة الأولى(الخارجية) حيث أنه في كل دورة تنفذ فيها الحلقة الأولى تنفذ الحلقة الداخلية(الثانية) 4 دورات,بالتالي ستنفذ الحلقة الثانية 6*4 أي 24 مرة في البرنامج كله.
ولتوضيح الفكرة نأخذ المثال التالي:
سننشئ برنامجاً يقوم بطباعة الأرقام من 1 إلى 6 في مربع نص 7 مرات,وبعد كل مرة يقوم بطباعة كلمة تعلم المستخدم ما هو ترتيب هذه المرة.

أنشئ فورم جديد, وضع فيه مربع نص واجعل قيمة الخاصية MultiLine تساوي True وذلك من نافذة الخصائص(انقر على مربع النص ثم انقر F4 وعدل القيمة).
ثم اكتب الكود التالي:

```
Private Sub Form_Load()

Me.Height = 2655                ' السطر الأول
Me.ScaleHeight = 2145           ' السطر الثاني
Me.Width = 7575
Me.ScaleWidth = 7455

Text1.Text = ""                 'السطر الخامس
Text1.FontSize = 20
For i = 1 To 7
Text1.Text = Text1.Text & "Tour " & i & " : "
For j = 1 To 6
Text1.Text = Text1.Text & j & " "
Next
Text1.Text = Text1.Text + " || "
Next
End Sub

Private Sub Form_Resize()
Text1.Left = 0
Text1.Top = 0
Text1.Height = Me.Height
Text1.Width = Me.Width
End Sub
```

مثال 1-14
لاتقلق بشأن أول 4 أسطر و لا آخر 4 أسطر فهي لأجل ضبط حجم الفورم ومربع النص وموقعه.
قمنا في السطر الخامس(نقصد في السطر السادس لكننا لا نعتبر السطر الأول سطر لأنه ليس تعليمة) بحذف النص الموجود في مربع النص.
أما في السطر السادس فجعلنا حجم النص الذي سيكتب في مربع النص =20 من خلال الخاصية FontSize.

لاحظ أننا دخلنا في الحلقة الخارجية ذات المتحول i عندها يضيف البرنامج إلى مربع النص عبارة تعبر عن رقم العداد i ثم بعدها يدخل بحلقة داخلية ذات متحول j و يطبع قيمة المتحول j ويعيدها 6 مرات ثم يعود ليعيد الحلقة الخارجية فتعاد الكرة حتى تنفذ الحلقة الخارجية 7 مرات.
أما الحدث Form_Resize يتحقق عند تغيير حجم الفورم,وعندما يبدأ البرنامج أيضاً.
وفي الإجراء الأخير يقوم البرنامج بجعل حجم مربع النص مساويا لحجم الفورم,جرب أن تغير حجم الفورم عند تنفيذ البرنامج ولاحظ ذلك.

انتهى - بعونه تعالى – الفصلان الأول و الثاني من كتاب:

تعلم الأساسيات في الفيجوال بيسك للمبتدئين

ترقبوا الفصل الثالث على أحد المواقع التالية:

- **مجموعة الكمبيوتر والبرمجة** **groups.google.com/group/ComputerAndProgramming**
- **مجموعة جمعية بريل وير** **groups.google.com/group/Prelware-Association**
- **أو من الرابط** **Prelware.250free.com**

مع تمنياتنا بالتوفيق والإفادة..!

ملاحظة: Visual Basic 6.0 هي لغة برمجة لشركة مايكروسوفت,والكاتب يحترم جميع الحقوق .