البرمجة والحواسيب /٢/

كلية الهندسة الميكانيكية

قسم الطاقة + ميكانيك الغزل والنسيج

السنة الثانية

الفصل الثاني

البرمجة والحواسيب /٢/

جامعة
حلب
١٩٥٨
UNIVERSITY
OF
ALEPPO

منشورات جامعة حلب
كلية الهندسة الميكانيكية

البرمجة والحواسيب /٢/

الدكتور المهندس
محمد صقر حماد
أستاذ مساعد في قسم هندسة الطاقة

لطلاب السنة الثانية
قسم هندسة الطاقة

مديرية الكتب والمطبوعات الجامعية
١٤٣٩هـ - ٢٠١٨م

جامعة
حلب
١٩٥٨
UNIVERSITY
OF
ALEPPO

الفهرس

الفصل الثاني

أنواع البيانات

Data Types

الفصل الثالث

تعليمات وأدوات الإدخال والإخراج

Input and Output Instructions and Tools

الفصل الرابع

تعليمات التحكم وأدوات التحكم

Control Instructions and Control Tools

الفصل الخامس

تعليمات وحلقات التكرار

Repetition Instructions and Loops

الفصل السادس

المصفوفات

Arrays

الفصل السابع

التوابع

Functions

الفصل الثامن

الألوان

Colors

الفصل التاسع

الرسم في الفيجوال البيزك

Drawing in VB

الفصل العاشر

المؤقت

Timers

الفصل الحادي عشر

محرر القوائم

Menu Editor

الفصل الثاني عشر

أحداث الماوس

Mouse Events

الفصل الثالث عشر

أحداث لوحة المفاتيح

KeyBoard Events

الفصل الرابع عشر

النماذج

Forms

الفصل الخامس عشر

مربعات الحوار الشائعة

Common Dialog Box

الفصل السادس عشر

عناصر تحكم الملفات

File – System Controls

الفصل السابع عشر

أداة الجدول المرن

Ms – Flex Grid Control

الفصل الثامن عشر

أمثلة محلولة

Resolved Examples

المفاهيم والمصطلحات الهندسية والمراجع العلمية

مُقَدِّمَة

PREFACE

تُعَرَّفْ البرمجة بأنها عملية كتابة تعليمات وأوامر لجهاز الحاسوب أو أي جهاز آخر، لتوجيهه وإعلامه بكيفية التعامل مع البيانات أو كيفية تنفيذ سلسلة من الأعمال المطلوبة تسمى خوارزمية.

نعيش اليوم في عالم رقمي متجدد ومتغير، وأصبحنا وبكل ما يحيط بنا من أعمال وتجارة وتسوق وعلوم واختراعات وصحة وطيران وحكومات وتدريس المقررات وإصدار النتائج ... الخ نحتاج إلى برمجيات تُدِيرُها وتَتَحَكَّمُ بها، وانطلاقاً من ذلك نحن بحاجة إلى التفكير بعمق في تدريس وتعليم الطلاب علوم الحاسبات والتفكير الخوارزمي (الحسابي) والبرمجة في جميع مراحل التعليم.

وإذا أخذنا بعين الإعتبار أنه ليس بالضرورة أنَّ كل من درس البرمجة سوف يصبح مبرمجاً، إلا أن كتابة البرمجية المطلوبة ثم تجربيها من قبل الطلاب يوفر تغذية راجعة فورية سواءً كانت تلك البرمجية صحيحة أم خاطئة، كما أن قدرة الطالب على أداء ما هو مطلوب منه والتفكير في تحويل المفاهيم وكتابتها للكمبيوتر لكي يقوم بتنفيذها هو أهم بكثير وأعمق من تفاصيل لغة البرمجة نفسها.

يهدف هذا الكتاب إلى توفير المزيد مِنَ المراجع العلمية المتخصصة بلغة بسيطة لطلبتنا ومهندسينا، حيث تم مناقشة المبادئ الأساسية المطلوبة لفهم المسألة وكتابة الخوارزمية الملائمة والأكواد التي يحتاجها الطالب لحل المسألة والإستخدام الأمثل لأدوات البرنامج والتي تضمن حل المسألة كما يرغب بها الطالب.

لقد جاءت مادة الكتاب في تسعة عشر فصلاً لتضم كلّ المحاضرات التي ألقيت على طلاب السنة الثانية من قسم هندسة الطاقة في كلية الهندسة الميكانيكية وهو يغطي بالكامل مفردات المقرر والمبادئ الأساسية ويتضمَّن التمارين والأمثلة والتطبيقات الهندسية بحيث يشكل منهجاً دراسياً كاملاً في البرمجة.

تم في الفصل الأول التعرض لمفهوم البرمجة المرئية والذي يهتم بتصميم واجهة البرنامج التي سيتعامل معها مستثمر البرنامج بشكل مباشر عن طريق إدخال المعطيات إلى البرنامج أو إخراجها منه، وفيها قدرُ كبير من المتعة وتعتمد

بكثرة على النقر بالماوس. بينما في المرحلة الكودية نستخدم محرر النصوص لكتابة البرنامج وتتألف البرامج من عبارات مكتوبة بلغة VB وتتصف عملية كتابة البرامج هنا بالسهولة مقارنة مع اللغات الأخرى.

وتم في الفصل الثاني التحدث عن أنواع البيانات والتي تتضمن الثوابت والمتغيرات والمعاملات الرياضية والإشارات الحسابية وقواعد تسمية المتحولات وأولوية تنفيذ العمليات الحسابية بلغة الفيجوال البيزك.

تحدث الفصل الثالث عن تعليمات وأدوات الإدخال والإخراج مثل أداة مربع النص وأداة مربع اللائحة وتعليمة الطباعة وإجراء مقارنة بينهما. وتطرق الفصل الرابع إلى تعليمات التحكم وأدوات التحكم حيث إن التعليمات الشرطية أو تعليمات التحكم هي التعليمات التي تغير التسلسل الطبيعي لتنفيذ تعليمات البرنامج بناءً على شرط معين. أما الفصل الخامس فتحدث عن تعليمات وحلقات التكرار والتي تسمح بتكرار تنفيذ مجموعة من التعليمات عدداً من المرات.

وتحدث الفصل السادس عن المصفوفات والتي تمثل مجموعة من المتغيرات من نفس النمط ترتبط مع بعضها، وتم التمييز بين المصفوفات الستاتيكية والديناميكية من جهة وبين المصفوفات أحادية البُعد والمتعددة الأبعاد من جهة أخرى، وتم في الفصل السابع التحدث عن التوابع بنوعيها: التوابع المعرفة ضمن اللغة البرمجية والتوابع المبنية ضمن البرنامج، كما تم التحدث عن أهم التوابع التي يمكن أن يستخدمها الطالب في دراسته الجامعية في قسم هندسة الطاقة الميكانيكية.

وفي الفصل الثامن تم التحدث عن طرق تمثيل الألوان وخصوصاً في المراحل التنفيذية كرسم المخططات والمنحنيات البيانية والتي تم التطرق إليها في الفصل التاسع والذي تم فيه التحدث عن الرسم في الفيجوال البيزك والأدوات المستخدمة للرسم وإظهار الرسوم كأداة مربع الصورة وأداة عرض الصور وأداة الأشكال، وتم التطرق لطرق الرسم المنظومة الإحداثية في مربع الصورة مثل تعليمة رسم نقطة وتعليمة رسم خط وتعليمة رسم الدائرة وتعليمة مقياس الرسم.

أما في الفصل العاشر فتم التحدث عن أداة المؤقت والتي تجعل البرنامج يؤدي عملاً تلقائيّاً معيناً وعلى فترة زمنية معينة ودون تدخل من المستخدم.

وفي الفصل الحادي عشر تم التحدث عن محرر القوائم والذي نستطيع من خلاله استخدام أشرطة القوائم والأدوات للمساعدة في عملية التبويب بشكل أحسن بالإضافة إلى الجمالية التي تضفيها على واجهة البرنامج أو الإطار.

تم في الفصـــلين الثاني عشـــر والثالث عشـــر التحدث عن أحداث الماوس وأحداث لوحة المفاتيح على اعتبار أنه هناك نوعان رئيسـيان من الأحداث: أحداث يسـببها المسـتخدم وأخرى يسـببها النظام، حيث إنَّ الأحداث التي يثيرها المسـتخدم هي التي تنتج عن فعل المستخدم مثل ضغط مفتاح أو نقر زر الماوس.

وتحدث الفصـل الرابع عشـر عن النماذج حيث إن النموذج عبارة عن كائن يعمل كحاوية للكائنات الأخرى كالعناوين ومربعات النص ومربعات الرســـم التي تتكون منها في النهاية واجهة المستخدم. أمَّا الفصل الخامس عشر فتطرق لمربعات الحوار الجاهزة ومسبقة التعريف ومربعات الحوار المخصصة والشائعة.

وفي الفصـل السـادس عشـر تم التحدث عن عناصـر تحكم الملفات وباختيار الملفات من محركات الأقراص المختلفة، وباختيار السواقة من محركات الأقراص. وفي الفصـل السـابع عشـر تم التحدث عن أداة الجدول المرن والتي تحتوي جدول ذي صـفوف أفقية وأعمدة عمودية والتي تسـتخدم للتحرير الشـبكي كالمصـفوفات والجداول.

في الفصـل الثامن عشـر تم التطرق لمجموعة من المسـائل المحلولة لتثبيت المعلومات في ذهن الطلاب والتي أتت كأمثلة مباشرة لجميع الفصول السابقة.

كذلك تضـــمن الكتاب بعض المفاهيم والمصـــطلحات الهندســـية المهمة والمراجع العلمية.

أخيراً أود أنْ أشـكر كلّ مَنْ سـاهم بمجهود في إعداد هذا الكتاب ومن يقدم ملاحظات ومقترحات علمية حول تقويم وتصـحيح مضـمونه وصـولاً إلى الأفضـل في طبعات الكتاب القادمة ... وفقنا الله إلى خدمة وطننا الحبيب.

حلب الأحد المصادف في ٢٠١٨/٠٩/٠٩

المؤلف

جامعة
حلب
١٩٥٨
UNIVERSITY
OF
ALEPPO

الفصل الأول
البرمجة المرئية
Visual Programming

تعريف:

هي لغة برمجة متقدمة وعالية المستوى وتتعامل بشكل جيد مع قواعد البيانات. والمقصود بذلك أن هذه اللغة تستقبل كلمات لغوية مفهومة لدى الإنسان وتحتاج إلى مترجم ليترجمها لتصبح مفهومة من قبل الحاسب.

مقدمة Introduction:

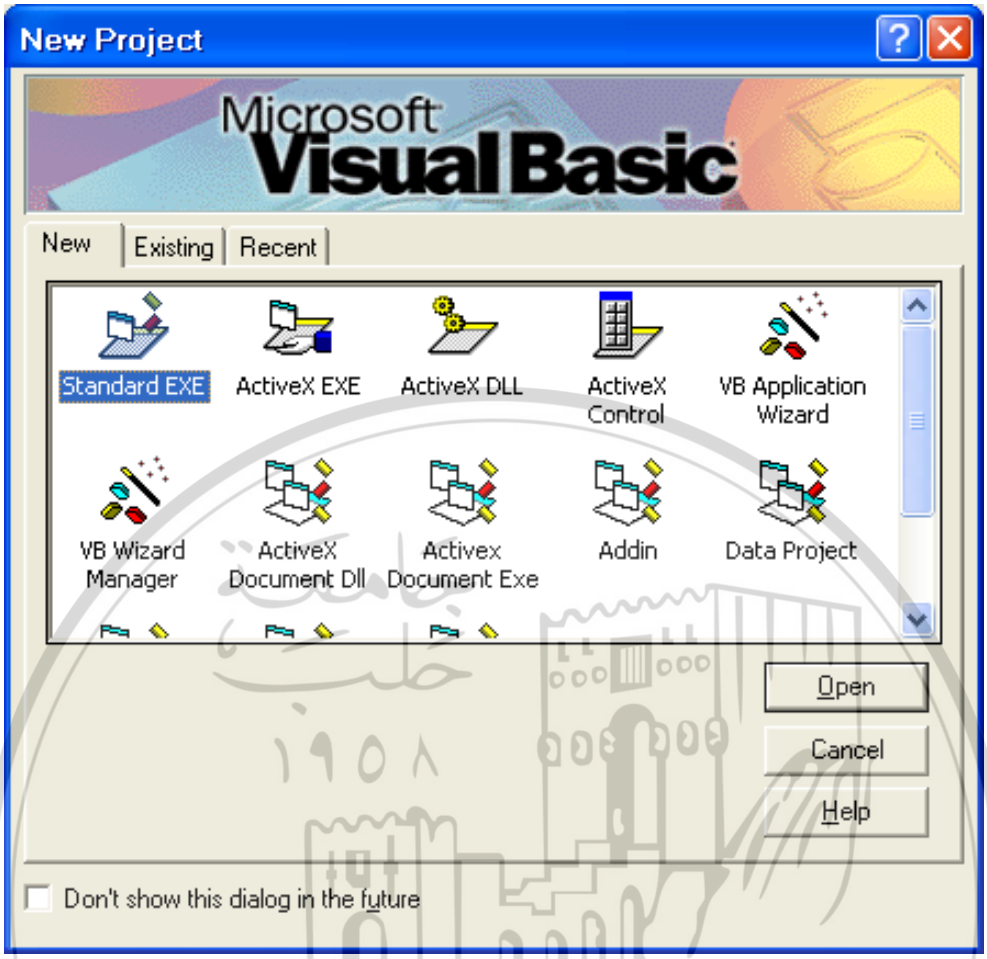
يتم في البرمجة المرئية تصميم واجهة البرنامج التي سيتعامل معها مستثمر البرنامج بشكل مباشر عن طريق إدخال المعطيات إلى البرنامج أو إخراجها منه، وفيها قدرُ كبير من المتعة وتعتمد بكثرة على النقر بالماوس. بينما في المرحلة الكودية نستخدم محرر النصوص لكتابة البرنامج وتتألف البرامج من عبارات مكتوبة بلغة VB وتتصف عملية كتابة البرامج هنا بالسهولة مقارنة مع اللغات الأخرى.

تشغيل Visual Basic: لتشغيل VB يجب إتباع الخطوات التالية:

- من شريط المهام وبالنقر على زر "ابدأ" فتظهر قائمة "البرامج" (كافة البرامج) ومن هنا تظهر قائمة تحتوي على كافة البرامج الموجودة ضمن نظام التشغيل ومن ضمنها القائمة الفرعية Microsoft Visual Basic 6 ومن هذه القائمة تنبثق قائمة فرعية أخرى نجد فيها الملف التنفيذي Visual Basic 6 فيتم تشغيل البرنامج.

"***Start → All programs → Microsoft visual studio 6.0 → Microsoft Visual Basic v.6***"

- يظهر مربع حوار "مشروع جديد New Project" والذي نختار منه نوع البرنامج الذي نريد إنشاؤه. يشتمل مربع الحوار هذا على ثلاث صفحات تبويب أو ثلاثة خيارات أو ثلاثة أفرع تظهر كما في الشكل (١):



الشكل (١) – مربع الحوار New Project

- New: يسمح هذا التبويب باختيار نوع المشروع الجديد الذي تريد إنشاؤه، ويحتوي على مجموعة من القوالب الجاهزة والتي تمكننا من إنشاء أنواع مختلفة من المشاريع Projects.
- Existing: يسمح لنا هذا التبويب باستعراض وفتح المشروعات الموجودة والمخزّنة مسبقاً لدينا على أحد أقراص التخزين.
- Recent: ويظهر في هذا التبويب أسماء آخر مشروعات تم تشغيلها (وهو كخيار المستندات الموجودة في قائمة ابدأ أي يحتوي على مسار تشغيل آخر المشاريع التي تم التعامل معها).
- في المشاريع الجديدة القياسية نأخذ دوماً من علامة التبويب New في مربع الحوار الأساسي New Project ملفاً تنفيذيّاً من النوع "ملف قياسي StandardEXE والذي يُعطينا إطار خالي من كل الأدوات ونستطيع من خلاله بدء العمل.

- بعد اختيار النموذج المطلوب ننقر عليه مرتين بزر الماوس الأيسر فيتم فتحه أو ننقر بزر الماوس الأيسر مرة واحدة فيتم تحديده ومن ثم نقوم بفتحه بالنقر على زر أو أيقونة (فتح – Open) الموجودة في مربع الحوار وهكذا ستظهر النافذة الأساسية للبرنامج التي تحتوي على واجهة البرنامج وفيها منطقة العمل الرئيسية.
- يمكن إلغاء خيار ظهور نافذة "مربع الحوار New Project" من خلال وضع العلامة √ في خانة الاختيار Don't Show This Dialog in the Future.
- المشروع هو مجموعة من الأشكال مع البرمجة التابعة لها أو هو مجموعة من نوافذ التصميم. وكلمة شكل مأخوذة من كلمة Form وتعطي اسم المشروع وأسماء الفورمات (نوافذ التصميم) الموجودة والمشروع قد يكون مؤلف من عدد من الفورمات.

منطقة العمل Visual Basic work area:

- فيما يلي نوضح أهم العناصر التي تظهر في واجهة البرنامج:

١) شريط الاسم Title Bar.

٢) شريط القوائم Menu Bar.

٣) أشرطة الأدوات Tool Bars.

٤) مربع الأدوات (أدوات التحكم) Tool Box.

٥) نافذة مستكشف المشروع Project Explorer.

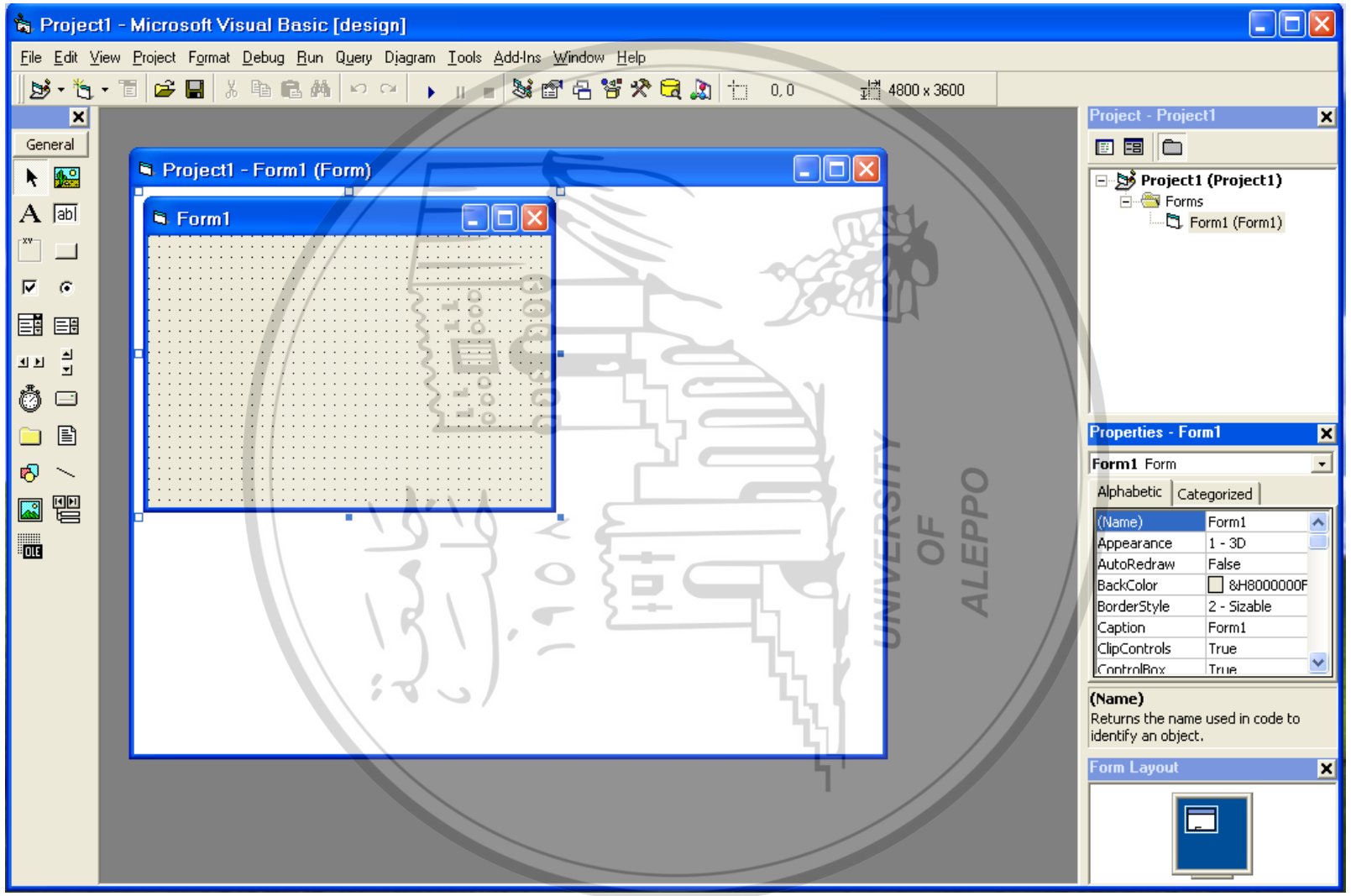
٦) نافذة الخصائص (خصائص الأدوات) Properties Window.

٧) نافذة توضع الإطار Form Layout Window.

٨) نافذة إطار المشروع Project Form والتي تحتوي بداخلها على نافذة الإطار Form والتي تعتبر ساحة العمل الأساسية والتي سيتم توضع الأدوات الأساسية في المشروع عليها.

شريط الاسم Title Bar: ويحتوي كما في الشكل (٢) على:

الشكل (٢) – شريط الاسم Title Bar



"*Start → All programs → Microsoft visual studio 6.0 → Microsoft Visual Basic v.6*"

a) رمز أيقونة البرنامج.

b) اسم البرنامج أو اسم التطبيق الذي يتم العمل عليه.

c) الأزرار الأساسية وهي:

- زر التصغير إلى شريط المهام.
- زر التكبير والاستعادة.
- زر إغلاق البرنامج.

شريط القوائم Menu Bar:

يحتوي على 13 قائمة تحتوي كل قائمة منها على مجموعة من الأوامر (الاختيارات) المختلفة كما في الشكل (٣).

تستخدم مثلاً قائمة File للتعامل مع ملفات المشاريع المختلفة بينما تستخدم قائمة Edit في كتابة البرامج وتصميم النماذج وتستخدم قائمة View للتحكم في عرض أو إخفاء عناصر بيئة التصميم، وتستخدم القائمة Run لتنفيذ أو إيقاف البرنامج وتستخدم القائمة Debug لاكتشاف وتصحيح الأخطاء وهكذا ...

الشكل (٣) – شريط القوائم Menu Bar

وفيما يلي تعريف بأهم القوائم الموجودة في شريط القوائم:

- **القائمة File:**

تحتوي هذه القائمة على أوامر أساسية خاصّة بالمشاريع بشكل عام، كإنشاء مشروع جديد New Project، حفظ محتويات المشروع Save Project، طباعة محتويات المشروع Print، وترجمة المشروع وتحويله إلى ملف تنفيذي Make Project1.exe والذي يسمح بتشغيل المشروع حتى دون تنصيب برنامج الفيجوال بيزك في نسخة غير قابلة للتعديل. والميّزة التي أضيفت لـ VB6 هي إمكانية فتح أكثر من مشروع في نسخة واحدة من البيئة وهي ميّزة تعرف بالمشاريع المُتعدّدة Multiple Projects ويظهر في أسفل القائمة أسماء آخر أربعة مشاريع تم فتحها والتعامل معها.

- **القائمة Edit:**

تحوي هذه القائمة على أوامر التحرير القياسية كالقص، النسخ واللصق بالإضافة إلى أوامر خاصة بقواعد البيانات في حالة وجود قاعدة بيانات في نافذة عرض البيانات Data View. إنَّ معظم الأوامر الواردة في أسفل هذه القائمة تُستخدم مع نافذة محرر الأكواد Code Window.

- **القائمة View:**

للتحكم بخصائص عرض النوافذ في واجهة البرنامج الرئيسية. وتسمح بإظهار أشرطة الأدوات والتي تحتوي على أهم الأوامر الأكثر تكراراً والموجودة في شريط القوائم.

- **القائمة Project:**

تختص معظم أوامرها بمحتويات المشاريع، فهي تمكننا من إضافة عنصر أو عناصر من عناصر المشروع كنوافذ النماذج Forms، وملفات البرمجة Module والفئات Classes ... الخ. كما يمكننا إضافة أدوات تحكم ActiveX Controls إضافية عن طريق الأمر Components أو تضمين مكتبات ActiveX DLL خارجية عن طريق الأمر Reference.

- **القائمة Format:**

تختص أوامر هذه القائمة بتنسيق الأدوات التي نضعها على نافذة النموذج من ناحية موقعها على النافذة، فتوجد أوامر مرنة تفيد في محاذاة الأدوات أو توسيطها على النافذة، بالإضافة إلى تغيير ترتيب ظهور الأدوات أي وضع أداة فوق الكل أو أداة خلف الكل. ويمكن استخدام الأمر الأخير Lock Controls إذا أردنا تصميم الأدوات ومنع الآخرين من تغيير أحجامها أو مواقعها عن طريق الخطأ، ونستطيع التراجع عن هذا الخيار (فتح القفل) بكل بساطة باختيار نفس الأمر مرة أخرى.

- **القائمة Debug:**

معظم أوامر التنقيح وضعت في هذه القائمة، ومن هذه الأوامر اختيار طريقة تنفيذ البرنامج كتنفيذ سطر واحد منه Step Into أو إجراء كامل Step Over أو أمر سابق Step

Out أو التنفيذ حتى الوصول إلى السطر الذي يوجد عليه مؤشر الكتابة Run to Cursor وبالنسبة لنقاط القطع Breakpoints فهي علامات تظهر مبدئيّاً باللون الأحمر على سطر معين بحيث تتم عملية الإيقاف المؤقت للبرنامج عند الوصول إلى هذه العلامات.

- **القائمة Run:**

تمكننا هذه القائمة من تنفيذ البرنامج ومن اختيار الأوامر الأخرى كالإيقاف المؤقت Break أو إنهاء عملية تنفيذ البرنامج End. إن استخدام الأمر Start with Full Compile مشابه لأمر التنفيذ Start ولن نحتاجه إلا نادراً.

- **القائمة Query:**

أوامر هذه القائمة ممكنة بعد إنشاء جملة استعلام SQL باستخدام الأداة Microsoft Query Builder.

- **القائمة Diagram:**

إن استخدام أوامر هذه القائمة غير ممكناً إلا عند التعامل مع قاعدة بيانات SQL Server أو ORACLE.

- **القائمة Tools:**

تحتوي هذه القائمة على أوامر مختلفة كمحرر القوائم Menu Editor ومسهل كتابة الإجراءات Add Procedure وغيرها. ويفيد الأمر Options في الوصول إلى صندوق الحوار Options والذي يوفر عشرات الخيارات الخاصة بتخصيص (تغيير إعدادات) بيئة التطوير المتكاملة IDE.

- **القائمة Add-Ins:**

إنّ أوامر هذه القائمة عبارة عن برامج مستقلة نسميها بالإضافات Add-Ins والتي هدفها توفير خدمات إضافية لبيئة التطوير والتي تزيد من مرونتها.

- **القائمة Windows:**

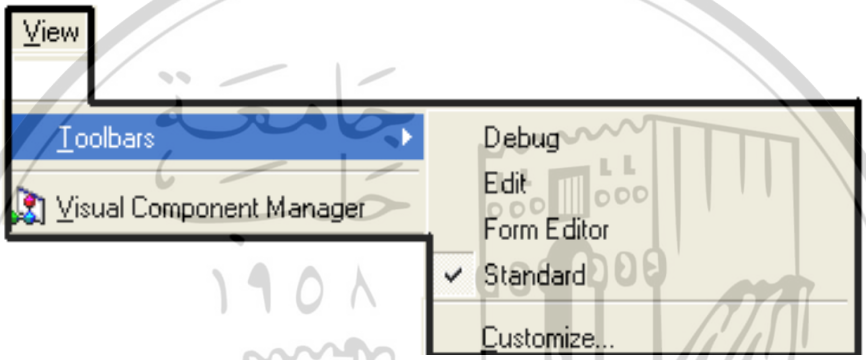
تعمل هذه النافذة عند وجود أكثر من إطار Windows مع بعضهما البعض.

- **القائمة Help:**

تفيد في الوصول إلى التعليمات الفورية في حالة تنصيب مكتبة MSDN أي Microsoft Developer Network.

أشرطة الأدوات Toolbars:

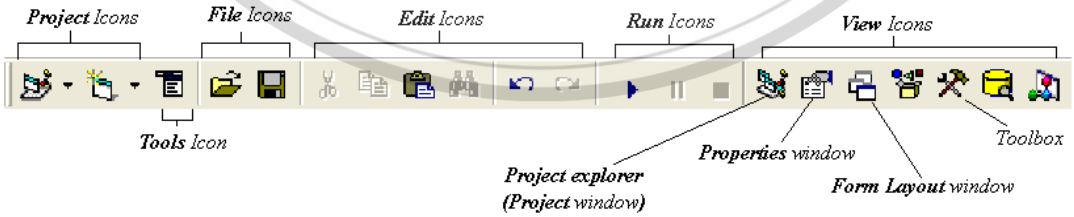
- تظهر أشرطة الأدوات من قائمة View من الأمر Toolbars.
- يشتمل VB على أربعة أشرطة ألأدوات هي كما في الشكل (٤).



الشكل (٤) – أشرطة الأدوات Tool Bar

يضم مجموعة من الأيقونات تشير إلى الأوامر الأكثر استخداماً.

- جميع الأزرار الموجودة في أشرطة الأدوات منسوخة من القوائم السابقة، أما الغرض من نسخها فهو تسريع عملية اختيار الأمر. ويمكن التحكم في أشرطة الأدوات وتحريرها كما في تطبيقات MS-Office وذلك من خلال النقر بزر الماوس الأيمن على شريط الأدوات واختيار الأمر Customize من القائمة المنسدلة. وتظهر الأزرار في أشرطة الأدوات وحسب القوائم التي أُخذت منها كما هو مبيّن في الشكل (٥):



الشكل (٥) – أزرار أشرطة الأدوات حسب القوائم التي أُخذت منها

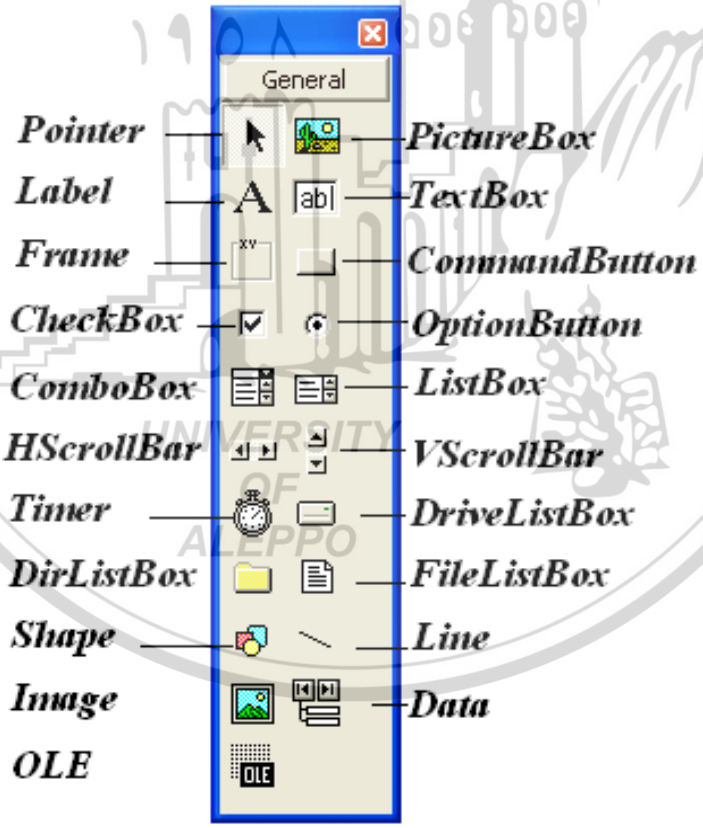
- يمكن إظهار وإخفاء أشرطة الأدوات هذه عن طريق الضغط على قائمة View ثم Toolbars ثم تحديد أو إلغاء تحديد (لإخفاء) شريط الأدوات المطلوب.

- يظهر شريط الأدوات Standard Toolbar دوماً ثابتاً تحت شريط القوائم، أما الأشرطة الثلاثة الأخرى فتظهر عائمة على سطح البرنامج ويمكن تثبيتها أيضاً.
- تحتوي أشرطة الأدوات على مجموعة من الأزرار يقوم كل منها مقام وظيفة يمكن استدعاؤها مباشرة بمجرد النقر عليها بدلاً من فتح القوائم واختيار الأوامر منها. وتظهر هذه الأزرار كما في الشكل (٦):

الشكل (٦) – أزرار أشرطة الأدوات

مربع أدوات التحكم Control Tool Box:

يشتمل على مجموعة أدوات تُسَهِّلْ التعامل مع VB وتستخدم في إعداد وتصميم واجهات البرامج. ويمكن نقل مربع الأدوات إلى أي مكان داخل الشاشة.



الشكل (٧) – مربع أدوات التحكم

من أهم الأدوات الموجودة في مربع أدوات التحكم والمستخدمة في تصميم واجهات البرامج وكما هو مبيّن في الشكل (٧) نجد:

- **المؤشر Pointer:**

يفيد في تغيير أبعاد وتحريك العناصر على النموذج وهو ليس أداة تحكم.

- **مربع الصورة Picture Box:**

أداة تمكننا من عرض الصور ورسم وتوليد الصور.

- **اللافتة Label:**

أداة لعرض النصوص غير القابلة للتعديل من قبل المستخدم وتستخدم للإعلان أو لتوضيح شيء ما.

- **مربع النص Text Box:**

أداة تستخدم لإدخال وعرض (إظهار) النصوص.

- **الإطار Frame:**

أداة تمكننا من فرز وترتيب العناصر في مجموعات.

- **زر الأوامر Command Button:**

يستخدم في الغالب لكتابة البرمجة تحت عنوانين وشكله يشبه الزر نتمكن من خلال من النقر والضغط عليه من الحصول على فعل معين.

- **صندوق الاختيار Check Box:**

تفيد خانة أو صندوق أو أداة الاختيار في أخذ خيار ما (اختياره) أو رفضه (عدم اختياره) من خلال وضع إشارة صح أو خطأ والتي على ضوئها تكون قيمته صفر أو واحد وهو مربع صغير جداً.

- **زر الاختيار Option Button:**

يستخدم في مجموعات الأوامر والتي يكون مطلوب فيها اختيار أحد هذه الأوامر فقط.

- **صندوق أو مربع اللائحة List Box:**

أداة تقوم بإظهار أو إخراج مجموعة من الحدود أو القيم.

- **الخانة المركبة Combo Box:**

أداة تقوم بإدخال أو إخراج مجموعة قصيرة من القيم أو الخيارات.

- **شريط التمرير الأفقي HScrollBar:**

شريط التمرير (السحاب) الأفقي Horizontal Scroll Bar والذي يساعد في رؤية العناصر بشكل أفقي.

- **شريط التمرير العمودي VScrollBar:**

شريط التمرير (السحاب) العمودي Vertical Scroll Bar والذي يساعد في رؤية العناصر بشكل عمودي.

- **المؤقت الزمني Timer:**

ويساعد البرنامج على القيام بعمل معين خلال فترات زمنية محددة دون أي تدخل من المستخدم، وهي أداة ظاهرة على الإطار في المرحلة المرئية (التصميمية) وغير مرئية للمستخدم في المرحلة التنفيذية.

- **أداة التحكم بالسواقات Drive List Box:**

تفيد في اختيار السواقة المطلوبة من بين السواقات الموجودة في الحاسب.

- **أداة التحكم بالمجلدات DirListBox:**

تفيد أداة التحكم بالمجلدات Directory List Box في اختيار المجلد المطلوب.

- **أداة التحكم بالملفات FileListBox:**

وتفيد في عملية إظهار كل الملفات الموجودة في دليل معين.

- **أداة الأشكال Shape:**

وهي أداة لرسم الدوائر والمستطيلات والمساحات والقطوع.

- **أداة الخط Line:**

أداة لرسم الخطوط.

- **أداة الصورة Image:**

أداة الصورة وهي أداة لعرض الصور وهي أقل قدرة على التعامل مع الصور من الأداة Picture Box.

- **أداة البيانات Data:**

أداة التعامل مع البيانات.

- **أداة التحكم OLE:**

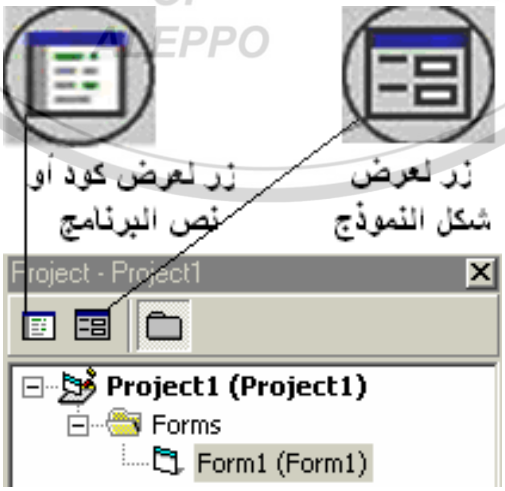
أداة التحكم والتي تمكننا من التفاعل مع التطبيقات الأخرى لبرامج Windows.
يتم إظهار مربع أدوات التحكم من قائمة View ومن ثم اختيار Toolbox.

نافذة المشروع Project window:

توجد في الجزء الأيمن العلوي من واجهة البرنامج وتضم مجموعة الملفات والنماذج الخاصة بالمشروع ويمكن رؤية أي عنصر إما بالنقر المزدوج عليه أو بتنشيطه ثم نقر زر View Code، الأمر الذي يسمح لنا برؤية الأوامر الكودية أو زر View Form وهو الأمر الذي يمكننا من رؤية النافذة المرئية. وتظهر هذه النافذة كما هو مبيّن في الشكل (٨) باستخدام إحدى الطريقتين:

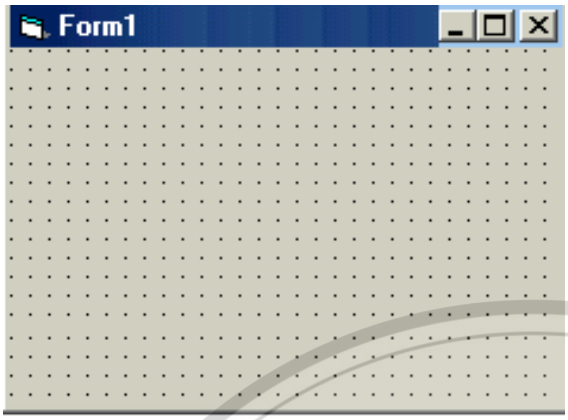
١) من قائمة **View** الأمر **Project Explorer**.

٢) من التركيب **[Ctrl + R]**.



الشكل (٨) – نافذة المشروع

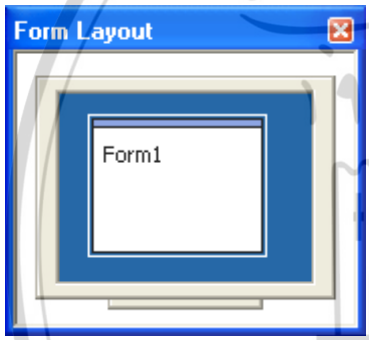
النموذج Form:



الشكل (٩) – الإطار أو النموذج Form

يظهر في منتصف الشاشة ويستخدم لتصميم واجهات البرنامج حيث توضع عليه جميع الأدوات المستخدمة ويحتوي البرنامج على إطار (نموذج) واحد أو أكثر من هذه النماذج. يظهر النموذج في واجهة البرنامج كما في الشكل (٩).

نافذة توضع الإطار Form Layout Window:



الشكل (١٠) – نافذة توضع الإطار

وهي النافذة التي نستطيع من خلالها تحديد مكان ظهور الإطار في المرحلة التنفيذية بالنسبة لشاشة جهاز الحاسب.

وتظهر هذه النافذة في واجهة البرنامج كما في الشكل (١٠).

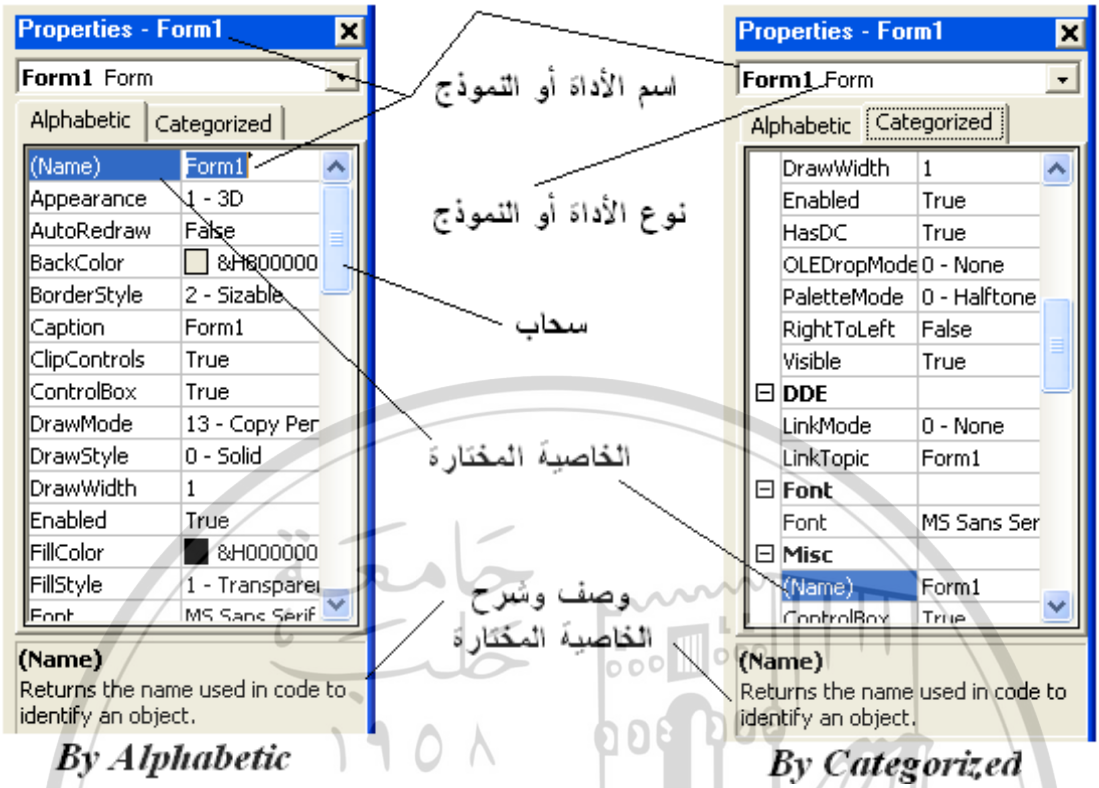
مربع الخصائص Properties Window:

يستخدم للتحكم بخصائص الأدوات التي وضعت على نافذة التصميم. حيث يعرض مجموعة الخصائص التي تخص كائن معين أو أداة ما موجودة في واجهة البرنامج. وتظهر الخصائص مبوبة بإحدى طريقتين كما في الشكل (١١):

- حسب ترتيب واسم Name هذه العناصر By Alphabetic.
- حسب مظهر Appearance وسلوك Behavior هذه العناصر By Categorized.

تظهر نافذة خصائص العناصر باستخدام إحدى الطريقتين:

- قائمة View أمر Properties Window.
- باستخدام المفتاح الوظيفي F4.



الشكل (١١) – مربع خصائص الأدوات

٥. ٤. خصائص الأدوات Tools Properties:

إن لكل أداة من هذه الأدوات خاصية مستقلة تميّزها عن غيرها من الأدوات ولكن لأغلب هذه الأدوات مجموعة من الخصائص المشتركة نذكر منها:

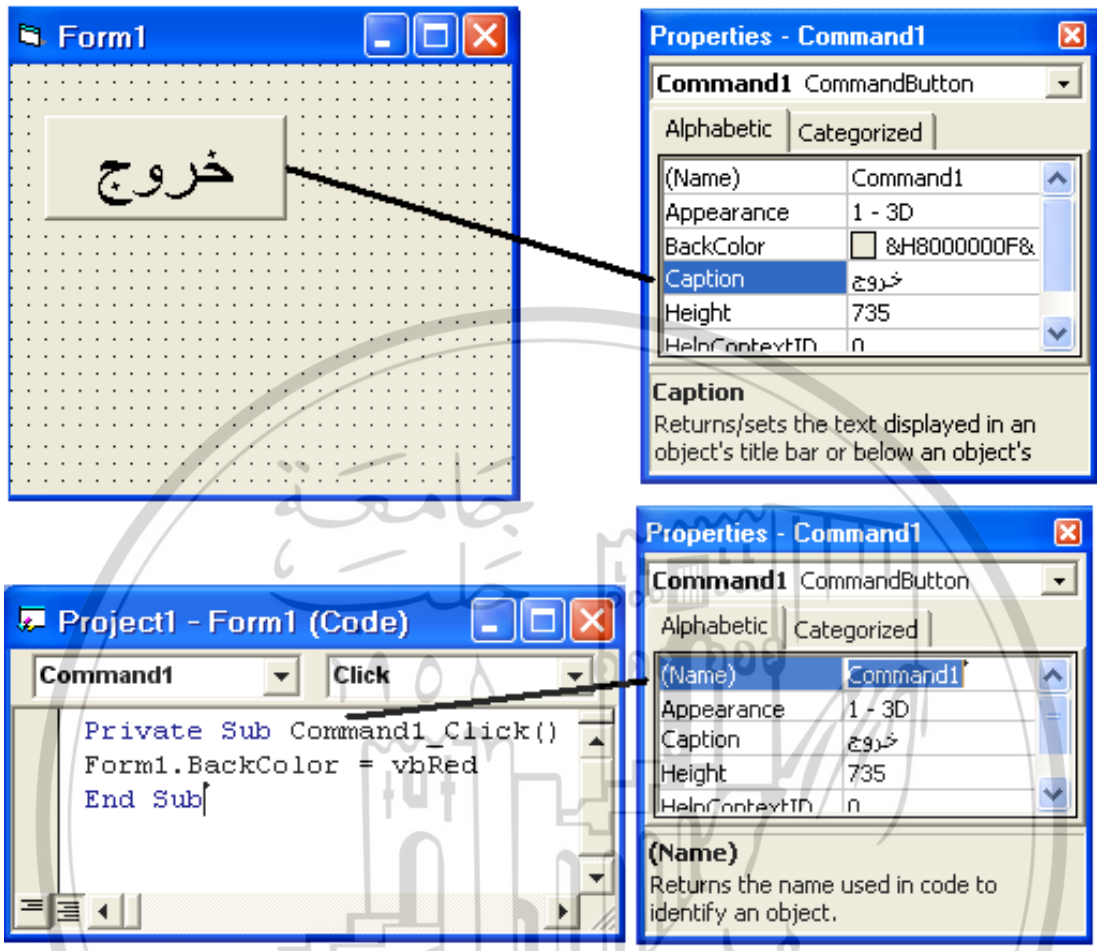
خاصية الاسم الكودي Name:

وهي الخاصية التي يتم من خلالها تمييز أداة من الأدوات عن غيرها في البرمجة الكودية والتي ستمكننا من التعامل معها في المرحلة التنفيذية.

خاصية الاسم المرئي Caption:

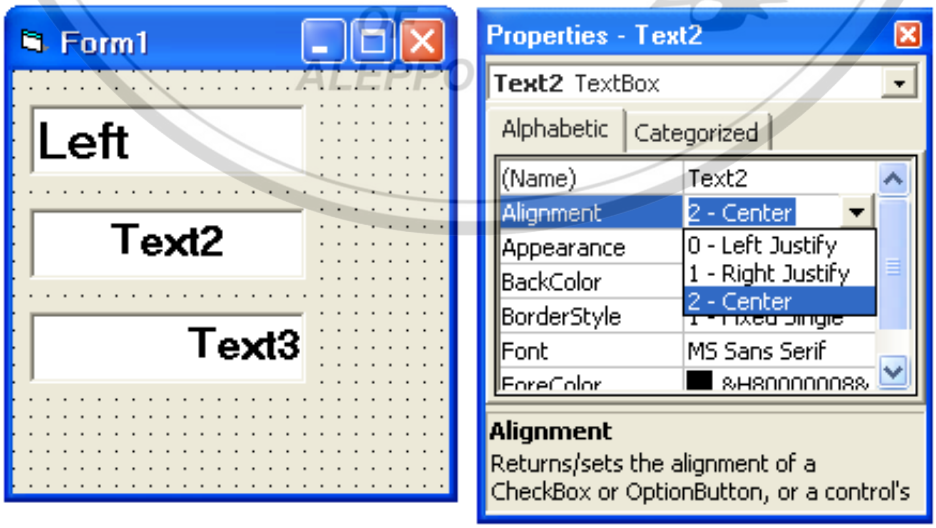
وهو الاسم المرئي الذي سيظهر على الأداة في المرحلة المرئية أو التنفيذية ويمكن تعديل هذه الخاصيّة في المرحلتين.

يبيّن الشكل (١٢) الخاصيتين السابقتين (الاسم الكودي والمرئي). من الواضح أنّ Command1 هو الاسم الكودي لهذه الأداة لأنّه مرتبط بالكود أمّا "خروج" فهو الاسم المرئي والذي يميّز هذه الأداة من غيرها من الأدوات.



الشكل (١٢) – خاصية الاسم الكودي والاسم المرئي

خاصية الضبط أو المحاذاة Alignment:

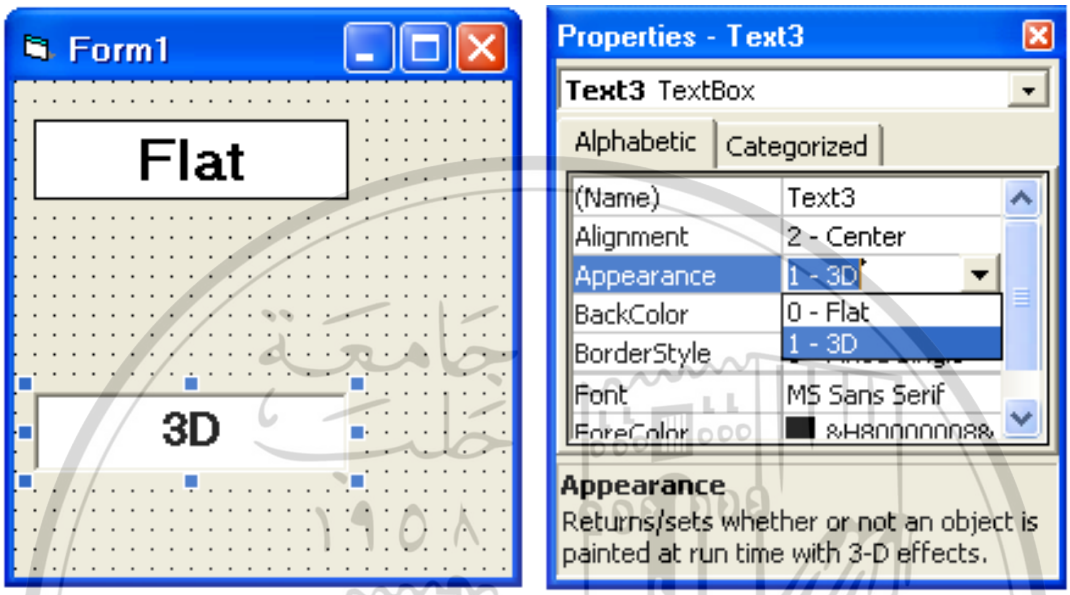


الشكل (١٣) – الخاصية Alignment للضبط والمحاذاة

خاصية المظهر Appearance:

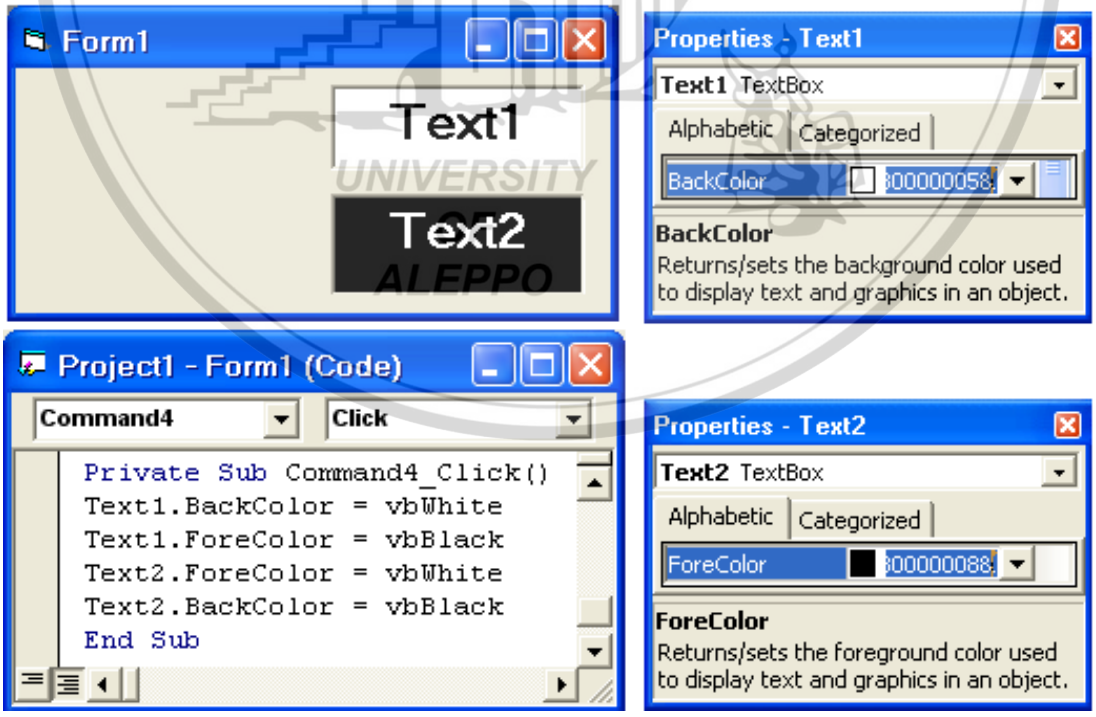
طبيعة ظهور الأداة على النموذج (عائمة أو مسطحة Flat أم ثلاثية الأبعاد 3D).

ويبيّن الشكل (١٤) تأثير هذه الخاصيّة على طريقة ظهور الأداة:



الشكل (١٤) – الخاصية Appearance للمظهر

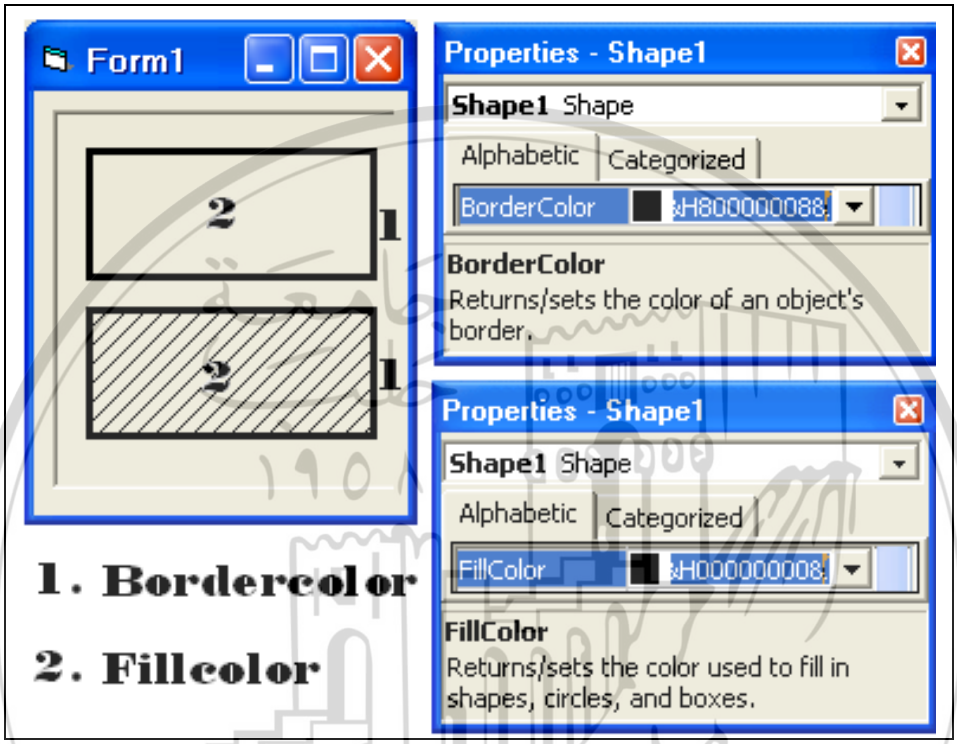
خصائص اللون Color:



الشكل (١٥ – a) – خصائص الألوان

وهناك مجموعة من الخصائص التي لها علاقة باللون Color. ويظهر على الشكلين (a, b ،١٥) الاختلاف بين هذه الخصائص.

يظهر على الشكل (a–١٥) تأثير الخاصيتين BackColor, ForColor كما يظهر على الشكل (b–١٥) تأثير الخاصيتين BorderColor, FillColor :



الشكل (١٥ – b) – خصائص الألوان

a) لون الخلفية Backcolor:

وهي خاصية تمكننا من تغيير لون خلفية الأداة.

b) لون الكتابة أو الرسم Forecolor:

لون الكتابة أو لون الأشكال المرسومة ضمن أداة من الأدوات.

c)لون الحافة أو الإطار المحيط Bordercolor:

لون الحافة أو الإطار أو البرواز الذي يحيط بأداة ما ويحدد محيطها.

d) لون التعبئة أو الملء Fillcolor:

لون تعبئة أو ملء أداة من الأدوات أو لون التهشير فيها.

خصائص أبعاد الأداة ومكانها:

مجموعة من الخصائص تتعلق بأبعاد الأداة ومكان تواجدها على الإطار ويظهر على الشكل (١٦) خصائص الأبعاد Dimentions والمكان Placement. مثل:

a) البعد العلوي أو الحافة العلوي Top:

بعد الجزء العلوي للأداة عن الطرف العلوي للنموذج. ومن الواضح أنه يزداد كلّما اتجهنا إلى الأسفل.

b) البعد الأيسر أو الحافة اليسرى Left:

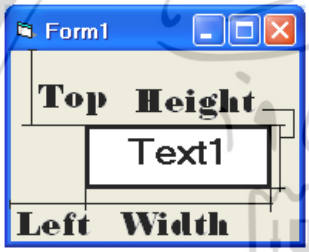
بعد الجزء الأيسر للأداة عن الطرف الأيسر للنموذج. والذي يزداد كلّما اتجهنا إلى اليمين.

c)الارتفاع Height:

ارتفاع الأداة.

d) العرض Width:

عرض الأداة.



الشكل (١٦) – خصائص الأبعاد والمكان

الخط Font:

يسمح بإظهار مربع حوار للتحكم بنوع وحجم ونمط وتنسيق الخط المستخدم.

خصائص الرسم Drawing Properties:

وهي مجموعة خصائص لها علاقة بالأدوات التي تسمح بالرسم مثل:

a) DrawMode:

تحدد طبيعة طريقة ظهور الرسوم والمخططات.

b) DrawStyle:

تحدد شكل الخطوط التي يتم فيها الرسم (مستمرة – منقطة – ... الخ).

c) DrawWidth:

تحدد عرض خط الرسم أو سماكته.

مرئية وفعالية واختيار الأدوات:

خصائص لها علاقة بمرئية وفعالية واختيار بعض الأدوات، ويمكن أن تطبّق هذه الخصائص على بعض الأدوات ويمكن أن لا تتقبلها بعض الأدوات الأخرى:

a) المرئية Visible:

ظهور أو عدم ظهور الأداة.

b) التمكين والتفعيل Enabled:

إمكانية التعامل مع هذه الأداة أو عدم الإمكانية (تعطيلها).

c)التحديد أو الاختيار Checked:

وتحدد اختيار هذه الخاصية أو الأمر أم لا، وتستخدم هذه الخاصيّة مع القوائم.

مثال: لدينا ثلاثة أزرار أوامر وثلاث قوائم تظهر كما في الشكلين (١٧- a, b):

الشكل (١٧ – a) – خصائص مرئية واختيار وفعالية الأدوات

في الجزء الأيسر من القوائم الثلاثة Mnu2, Mnu3, Mnu4 داخل القائمة Mnu1 بدون أي إضافات على الشكل الإفتراضي الذي تظهر فيه. أمّا في الجزء الأيمن فلقد تم تغيير بعض الخصائص كما يلي:

- القائمة Mnu1 كانت ظاهرة وفعّالة ولم تكُن مختارة وبقيت كما هي. إذاً كل الخصائص الإفتراضية للقائمة Mnu1 بقيت كما هي ولم يتم تغييرها.
- القائمة Mnu2 كانت ظاهرة واختفت، إذا تم تغيير خاصية ظهورها Visible وأخذت القيمة المنطقية False بدلاً من True (أي أصبحت غير ظاهرة). ويمكن القيام بذلك في المرحلة التنفيذية إذا كتبنا الكود التالي:

```
Mnu2.Visible = False
```

- القائمة Mnu3 لم تكن مُختارة، أمّا الآن فلقد تم اختيارها، إذا تم تغيير خاصية اختيارها Checked وأخذت القيمة المنطقية True بدلاً من False (أي أصبحت مختارة). ويمكن القيام بذلك في المرحلة التنفيذية إذا كتبنا الكود التالي:

```
Mnu3.Checked = True
```

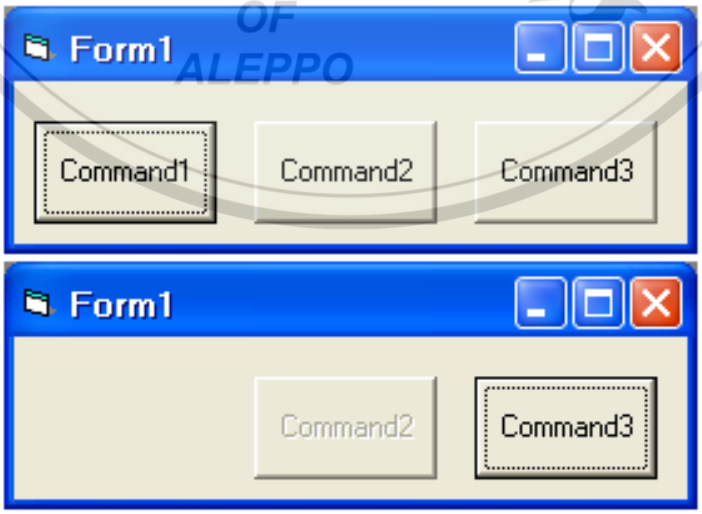
- القائمة Mnu4 كانت فعّالة، أمّا الآن فلقد أصبحت باهتة (غير فعّالة أو مُعطّلة) أي تمّ تعطيلها، إذا تم تغيير خاصية فعاليتها Enabled وأخذت القيمة المنطقية False بدلاً من True (أي أصبحت باهتة ومعطّلة). ويمكن القيام بذلك في المرحلة التنفيذية إذا كتبنا الكود التالي:

```
Mnu4.Enabled = False
```

الآن إذا قمنا بتنظيم جدول بخصائص القوائم الظاهرة في الشكل سنجد:

Checked	Enabled	Visible	القائمة
False	True	True	Mnu1
False	True	False	Mnu2
True	True	True	Mnu3
False	False	True	Mnu4

الآن في الجزء العلوي من الشكل (b-١٧) تظهر أزرار الأوامر الثلاثة Command1, Command2, Command3 كما هي في الحالة الإفتراضية بدون أي تغيير أو تعديل. أمّا في الجزء السفلي فلقد تم تغيير بعض الخصائص كما يلي:



الشكل (١٧ – b) – خصائص مرئية واختيار وفعالية الأدوات

- زر الأوامر Command1 كان ظاهراً واختفى، إذا تم تغيير خاصية ظهوره Visible وأخذت القيمة المنطقية False بدلاً من True (أي أصبح غير ظاهراً). ويمكن القيام بذلك في المرحلة التنفيذية إذا كتبنا الكود التالي:

```
Command1.Visible = False
```

- زر الأوامر Command2 كان فعّالاً وتم تعطيله، إذا تم تغيير خاصية فعاليته Enabled وأخذت القيمة المنطقية False بدلاً من True (أي أصبح غير معطلاً). ويمكن القيام بذلك في المرحلة التنفيذية إذا كتبنا الكود التالي:

```
Command2.Enabled = False
```

- زر الأوامر Command3 بقي كما هو ولا يمكن تطبيق الخاصية Checked على هكذا نوع من الأدوات. الآن بتنظيم جدول بخصائص القوائم الظاهرة سنجد:

أزرار الأوامر	Visible	Enabled	Checked
Command1	False	True	-
Command2	True	False	-
Command3	True	True	-

القيم التي تأخذها بعض الأدوات أثناء العمل:

وهي مجموعة القيم الأعظمية والأصغرية والحالية لأداة من الأدوات. قد توجد مثل هكذا خصائص لبعض الأدوات مثل أشرطة التمرير Scroll Bars وقد لا تكون موجودة لأدوات أخرى. ومن أهم هذه الخصائص نجد:

a) القيمة الأصغرية للأداة Min:

القيمة الأصغرية التي تأخذها الأداة.

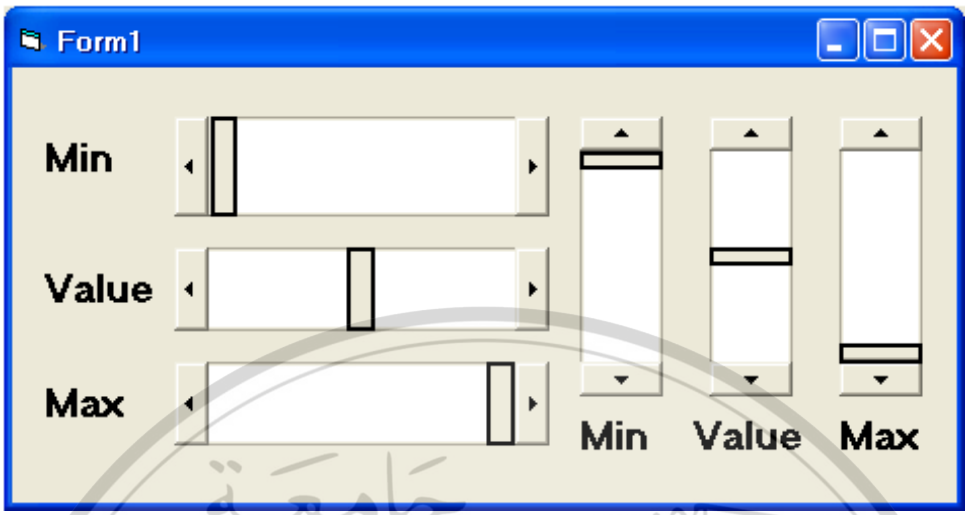
b) القيمة الأعظمية Max:

القيمة الأعظمية التي تأخذها الأداة.

c)القيمة الحالية Value:

قيمة الأداة في هذه اللحظة.

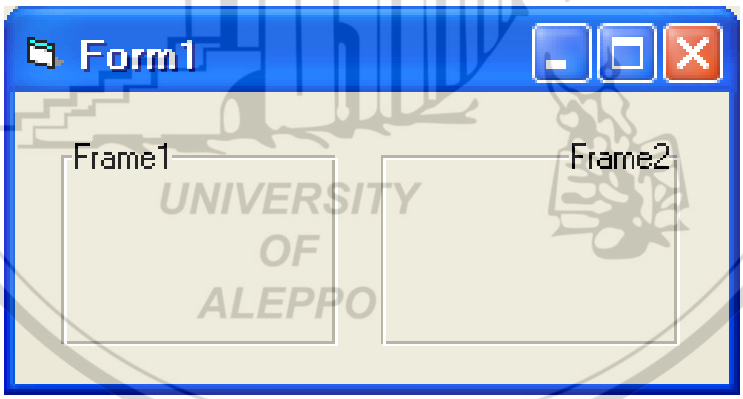
ويبيّن الشكل التالي بعضاً من قيم أشرطة التمرير الأفقية والعمودية أثناء العمل:



الشكل (١٨) – القيم التي تأخذها أداة "شريط التمرير" أثناء العمل

خاصية اليمين لليسار RightToLeft:

خاصية الكتابة أو الظهور من اليمين لليسار وبالعكس. تسمح هذه الخاصية بتغيير جهة الكتابة من اليمين (كما في نظام الكتابة باللغة العربية أو بالعكس)، وتشبه هذه الخاصيّة إلى حدٍّ ما خاصيّة المحاذاة أو الضبط. والشكل التالي يبيّن هذه الخاصيّة.



الشكل (١٩) – خاصية اليمين لليسار

خاصية تحميل الصورة Picture:

تستخدم هذه الخاصيّة مع الأدوات التي تتعامل مع الصور والأيقونات، وتُحدّد وبشكل دقيق المسار الذي يجب أن يتم تحميل ملف الصورة أو الأيقونة منه. ويمكن القيام بذلك من خلال مربع حوار "Load Picture" والذي يظهر عند اختيار الخاصية Picture للأداة كما في الشكل التالي:

الشكل (٢٠) – تحميل الصورة

ويمكن القيام بذلك في المرحلة التنفيذية إذا كتبنا الكود التالي:

$$Picture1.Picture = LoadPicture("E:\Darina\Darina3.gif")$$

المقياس Scale:

المقياس وهناك مجموعة من الخصائص التي لها علاقة بها مثل:

a) عرض المقياس ScaleWidth:

تحدد قياس المنطقة التي تحدد عرض كائن أو مقياس البعد من جهة العرض وهي أصغر من عرض الكائن بمقدار سمك البرواز.

b) ارتفاع المقياس ScaleHeight:

تحدد قياس المنطقة التي تحدد ارتفاع كائن أو مقياس البعد من جهة الارتفاع وهي أصغر من ارتفاع الكائن بمقدار سمك البرواز.

c)نوع المقياس ScaleMode:

تحدد واحدة القياس التي ستتعامل معها الأداة (Pixel, Point, Inch).

خاصيّة حالة النافذة WindowState:

تحدد طبيعة ظهور نافذة النموذج ويمكن أن تكون:

a) الحجم الأعظمي Maximized:

تظهر مكبرة وهي تملأ حجم الشاشة بالكامل.

b) الحجم الأصغري Minimized:

تظهر وهي مصغرة في شريط المهام.

c)الحجم العادي Normal:

تظهر بالحجم الموجودة فيه في الحالة المرئية (التصميمية).

خاصيّة مؤشر الماوس MousePointer:

تحدد شكل مؤشر الماوس عند اقترابه من أداة ما أو عند مروره فوق أحد هذه الأدوات.

ملاحظة: إن أغلب هذه الخصائص مشتركة بين معظم الأدوات ويمكن لبعض هذه الخصائص أن تكون موجودة في بعض الأدوات وغير موجودة في البعض الآخر.

وحتى لا نُخطئ في كتابة الأوامر الكودية تم اعتماد بادئات Prefix خاصة بكل أداة من الأدوات Control والتي تم تنظيمها في الجدول التالي:

جدول (٥-١) اللواحق المعتمدة بالنسبة لأدوات التحكم

Prefix	Control	Prefix	Control
cbo	Combo box	lbl	Label
chk	Check box	lin	Line
cmd	Command button	lst	List box
dir	Directory list box	mnu	Menu
drv	Drive list box	ole	OLE client
fil	File list box	opt	Option button
fra	Frame	pic	Picture box
frm	Form	shp	Shape

grd	Grid	tmr	Timer
hsb	Horizontal scrollbar	txt	Text box
img	Image	vsb	Vertical Scrollbar

مصطلحات هامّة وتعاريف:

تحتوي لغة VB على بعض المصطلحات الهامّة والتي تستخدم أثناء كتابة أكواد البرنامج أو أثناء العمل مع البرنامج أو أثناء تصميم واجهة البرنامج ومن أهمها:

أدوات التحكم Controls:

كائنات برمجية Objects تم تصميمها مرة واحدة وتستخدم مرات عديدة، وهي تمثل القطع المكونة لواجهة البرامج المصمّمة بـ VB. ومن الأمثلة على أدوات التحكم نجد مربع النص Text Box وزر الأمر Command Button ... الخ.

الحدث Event:

هو فعل يقوم به المستخدم أو نظام التشغيل أو البرنامج نفسه. ومن الأمثلة على الأحداث نجد حدث ضغط أحد مفاتيح أو حدث نقر احد أزرار الماوس ... الخ.

الوظائف Methods:

أفعال محددة يمكن للكائن أن يقوم بها.

الكائن Object:

الكائن البرمجي هو أحد العناصر الأساسية للبرنامج، يحتوى على خصائص Properties تحدد مميزاته، ووظائف Methods تحدد مهامه التي يمكن إنجازها. ومجموعة من الأحداث Events والتي يمكن أن يشعر بها. ومن الأمثلة على الكائنات المستخدمة في VB نجد النماذج Forms وأدوات التحكم Controls ... الخ.

الإجراءات Procedures:

هي مقاطع من التعليمات (نطلق عليها الكود) والتي تكتب لغرض معين وغالبًا ما تكون مقترنة بحدث فتسمّى عندئذٍ بالإجراءات الحدثية. في هذه الحالة سيتم تنفيذ الأكواد المكتوبة في الإجراء في لحظة وقوع الحدث المطلوب.

الخصائص Properties:

هي مزايا الكائن مثل حجمه وموقعه على الشاشة، ولونه ونوع الخط المستخدم في الكتابة عليه، أي أن الخصائص تحدد مظهر الكائن. وهناك بعض الخصائص الأخرى والتي تقوم بتحديد سلوك الكائن أيضاً.

مراحل البرمجة:

تمر برمجة مسألة معيّنة بعد تحليل المسألة وفهمها وتحديد الأدوات المطلوب التعامل معها بثلاث مراحل مهمّة هي مرحلة التعامل مع الأدوات (وضع الأدوات على الإطار وإعطائها الأماكن والأبعاد المناسبة)، ومرحلة كتابة الأكواد والمرحلة التنفيذية. هذه المراحل مهمة ويتم العمل فيها كما يلي:

١) مرحلة البرمجة المرئيّة:

١. تبدأ مرحلة البرمجة المرئية بإدخال أدوات لغة الـ VB الموجودة في شريط الأدوات ووضعها على الإطار Form1.

٢. بعد إدخال الأدوات المطلوبة إلى إطار النموذج سنقوم بتحريكها ووضعها في المكان المطلوب ومن ثم تكبيرها وتصغيرها إلى أن تأخذ الحجم المطلوب.

٣. لكل أداة من الأدوات مجموعة من الخصائص تميّزها عن غيرها مثل Caption وName والتي تحمل العنوان أو التسمية الافتراضية التي يعطيها برنامج الـ VB لهذه الأدوات، لذا يجب تغيير بعض الخصائص لهذه الأدوات لسهولة التعرف عليها ولتمييزها عن بعضها البعض وخاصّةً في حالات الاستخدام المتكرر.

٢) مرحلة البرمجة الكودية:

- تمتاز مرحلة البرمجة الكودية بكتابة النصوص للكائنات (العناصر) التي تم توضيعها على الإطار Form1 حيث تمثل هذه النصوص الأوامر (الأكواد) التي يجب تنفيذها استجابة لحادثة ما ستُطبق على العنصر المرئي كالنقر عليه بأحد أزرار الماوس Click أو الضغط على أحد مفاتيح لوحة المفاتيح Key Down عندما يكون هذا الكائن محدداً.

- يقوم VB بإضافة سطرين يعبران عن بداية ونهاية الإجراء الخاص بالكائن المحدد والحادثة المختارة عليه. فمثلا يتم التعامل مع زر الأوامر Command Button عن طريق النقر عليه Click بزر الماوس ولذا عند محاولة برمجته كوديّاً سيظهر معه السطران التاليان:

```
Private Sub Command_Click ( )
   VB – Code
End Sub
```

- إن السطر الأول يبدأ بالكلمتين التاليتين Private Sub:
- الكلمة Sub: عبارة عن كلمة محجوزة في لغة VB والتي تدل على أن الإجراء Procedure سيبدأ من هنا.
- الإجراء Procedure: فهو عبارة عن نص برنامج مكرّس لحادثة خاصة، والذي هنا هو إجراء النقر Command_Click ( ).
- السطر الأخير من النص أيضا كتب من قبل لغة الـ VB وتشير العبارة End Sub على نهاية الإجراء.
- اسم الإجراء Command_Click ( ): أتى من عملية النقر Click على الأداة أو زر الأوامر والذي اسمه الكودي Command ومعنى ذلك أن لغة VB ستقوم بربط هذه الأداة بنص برمجي (كود) يتم كتابته في الإجراء Procedure والذي سيتم تنفيذ مدلوله أو محتواه عند النقر (الحدث Click) بزر الماوس على الأداة التي اسمها الكودي Command.
- يفصل بين القسمين الأول والثاني من الاسم رمز الخط التحتي ونلاحظ أن آخر رمزين في اسم الإجراء هما القوسين ( ).
- نقوم الآن بين هذين السطرين بكتابة النص الواجب تنفيذه من قبل البرنامج عند استخدام هذه الأداة المرئية وباستخدام الإجراء اللازم Procedure وهنا تنتهي مرحلة البرمجة الكوديّة.

٣) مرحلة البرمجة التجريبيّة أو التنفيذيّة:

- وهي المرحلة التي يتم فيها تجريب البرنامج أو التأكد من عمل البرنامج وبالتالي تنفيذه. وتبدأ هذه المرحلة إذا أخذنا الأمر Start من القائمة Run أو زر التشغيل Start في

شريط الأدوات أو باستخدام مفتاح الوظيفة F5 أو التركيب [Shift + F5] ويمكن تنفيذ البرنامج سطراً سطراً باستخدام المفتاح F8.

- عند وجود أخطاء برمجية أو رياضية في البرنامج، سنجد أنّ المدقّق النحوي واللغوي Compiler الموجود ضمن البرنامج سيتعرف على هذه الأخطاء وسيتوقف عند مكان الخطأ وسيعطي تعليقاً على ذلك. بعد تصحيح الخطأ ننهي هذه المرحلة وننتقل إلى المرحلة الأخيرة وهي المرحلة التنفيذيّة.

ملاحظة: قد يظهر أحياناً أخطاء في تنفيذ البرنامج والتي قد تؤدي إلى توقف النظام وانهياره وهذا ما سيجبرنا على إعادة تشغيل Windows أو VB مرة أخرى وهذا يعني أن عدم حفظ البرنامج قد يتسبب بضياعه وبالتالي سنضطر إلى بدء عملية البرمجة من جديد ولذا يفضل في أغلب الأحيان حفظ العمل قبل تنفيذه ولذلك نأخذ Save Project من شريط القوائم (قائمة ملف File) ثم نقوم بتنفيذ البرنامج وإذا حدث خطأ الآن سيكون بإمكاننا تشغيل VB مجدداً وفتح المشروع المخزن وتصحيح الخطأ ومعاودة تنفيذ التطبيق من جديد.

إيقاف تنفيذ البرنامج:

عند عمل البرنامج بالشكل الصحيح وبعد إنتهاء المرحلة التنفيذية وتحليل وقبول النتائج نقوم بإنهاء المرحلة التنفيذية بإحدى الطرق التالية:

١. إشارة X في الزاوية اليمنى العليا في الشاشة.
٢. أو من القائمة Run نختار الأمر Stop.
٣. أو من القائمة Run نختار الأمر End.
٤. أو من الزر End الموجود في شريط الأدوات.
٥. أو من خلال زر أوامر مخصص لإنهاء عمل البرنامج بعد أن نكتب فيه الأمر الكودي End.

حفظ البرنامج:

لحفظ البرنامج من المفضَّل إنشاء مجلد لحفظ ملف المشروع Project و ملفات الإطارات Forms بداخله ولأجل ذلك نستخدم إحدى الطرق التالية:

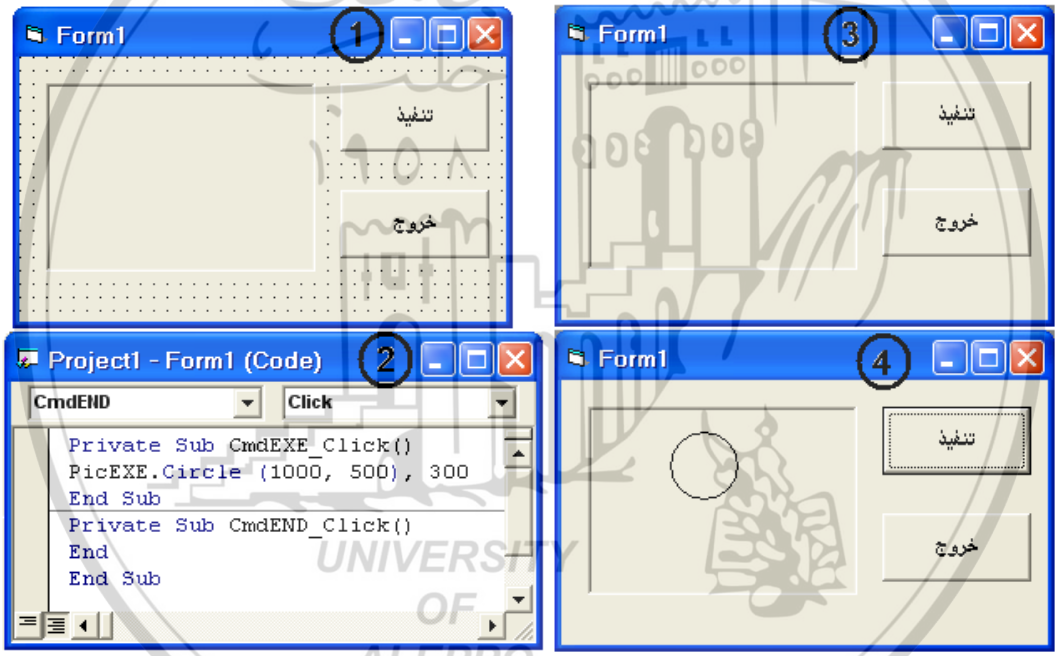
١. من القائمة File نأخذ الخيار Save Project والخيار Save Form.

٢. أو من الزر Save Project الموجود في شريط الأدوات.

٣. عند محاولة الخروج من البرنامج دون الحفظ سيظهر مربع يقترح علينا حفظ التعديلات والخروج.

سيظهر داخل المجلد أربع ملفات الأول ملف المشروع Project والثاني ملف الإطار Form وإثنان مكتوبان بلغة الآلة وهما ضروريان لعمل البرنامج.

مثال: نريد تصميم برنامج يقوم برسم دائرة ضمن مربع الصورة عند النقر بزر الماوس الأيمن على زر أوامر ويُنهي البرنامج عند النقر على زر أوامر آخر:



الشكل (٢١) – مراحل البرمجة

في المرحلة المرئية تم وضع ثلاثة أدوات على الإطار Form وهي مربع الصورة وزري أوامر كما تمّ تغيير أسماء هذه الأدوات الكودية والمرئية. بعد ذلك تم كتابة الكود في المرحلة الثانية وفي المرحلة الثالثة تم البدء بتنفيذ البرنامج.

عند النقر على زر الأوامر والذي اسمه الكودي CmdEXE سيتم تنفيذ الإجراء المكتوب بين سطري هذا الإجراء ومن الواضح أنّ هذا الكود هو عبارة عن دائرة مركزها (1000, 500) ونصف قطرها 300.

أدوات التحكم في فيجوال بيزك: VISUAL BASIC CONTROL TOOLS

أدوات التحكم هي كائنات معدة لوظائف خاصة ولها مجموعة من الخصائص والوظائف والأحداث. يأتي VB6 بفئة أساسية من أدوات التحكم والتي تجتمع داخل مربع واحد يسمّى مربع الأدوات والذي يبقى ظاهراً طوال فترة تصميم البرامج.

تشترك معظم أدوات التحكم بمجموعة من الخصائص بينما تتميز كل أداة من الأدوات بمجموعة من الخصائص الأخرى والتي تُميّزها عن غيرها وتمكّنها من القيام بوظيفة معينة داخل الواجهة لأنها قد تكون الأنسب للاستعمال في هذه الحالة.

إضافة أدوات التحكم إلى النموذج:

إن أول خطوات استخدام الأدوات هي إضافتها إلى النموذج ويتم ذلك إما بالنقر على الأداة في مربع الأدوات ثم سحبها إلى النموذج أو بالنقر المزدوج على الأداة فيظهر عندها عنصر بحجم افتراضي في وسط النموذج يمثّل هذه الأداة.

اختيار الأداة:

- قبل إجراء أي عملية من عمليات نقل أو تحجيم الأدوات أو نسخها أو حذفها يجب اختيار الأداة أو الأدوات ثم إجراء العملية المطلوبة عليها. إن اختيار الأداة يعني تحديدها أو تنشيطها بحيث تصبح جاهزة لاستقبال الحدث الذي سيطبق عليها.
- لاختيار أو تنشيط أداة بعد وضعها على الواجهة ننقر عليها بزر الماوس وعندها ستظهر ثمانية مربعات (مقابض أو مماسك) حول الأداة لتدل على اختيارها وستصبح هذه الأداة جاهزة لاستقبال وتنفيذ الأوامر.
- عند الرغبة بتنفيذ عملية ما على أكثر من أداة فلا بد من اختيار الأدوات كلها قبل تنفيذ العملية ويتم ذلك بطريقتين:
- إذا كانت الأدوات متجاورة فيمكن اختيارها جميعاً عن طريق النقر بزر الماوس في مكان فارغ على النافذة ثم سحب المؤشر وأثناء السحب يظهر مستطيل منقط (مربع التحديد) والذي سيحوي أو سيحدّد الأدوات التي تمّ اختيارها وبمجرد تحرير زر الماوس سيتم اختيار كل الأدوات التي تمّ احتواءها أثناء سحب الماوس.

- إذا كانت الأدوات متباعدة نحدد أول أداة ثم نضغط مفتاح Shift ثم نقوم بتحديد باقي الأدوات الأخرى المطلوبة (مع استمرار الضغط على المفتاح Shift). عند اختيار الأدوات نقوم بتحرير المفتاح Shift.

نقل الأدوات:

لنقل أداة من مكانها إلى مكان آخر نقوم بالنقر على هذه الأداة ثم نسحبها إلى المكان الجديد، وأثناء السحب سيتحرك مستطيل فارغ بنفس حجم الأداة، عندها نستمر بتحريك مؤشر الماوس وعند الوصول إلى المكان الجديد نقوم بتحرير زر الماوس، عندها ستظهر الأداة في المكان الجديد.

نسخ الأداة:

لنسخ أداة موجودة نتبع أحد الطرق المعروفة بالنسخ (قائمة تحرير أو بزر الماوس الأيمن أو من تراكيب لوحة المفاتيح ... نقوم بنسخ الأداة إلى الحافظة ومن ثم لصقها، عندها ستظهر نسخة من الأداة في أعلى النافذة من جهة اليسار.

حذف الأدوات:

لحذف أداة واحدة أو أكثر نختار الأداة أو الأدوات المطلوب حذفها ثم نضغط على المفتاح Delete من لوحة المفاتيح أو من القائمة Edit نختار الأمر Delete، كما يمكن القيام بذلك بأي طريقة من طرق الحذف الأخرى المعروفة.

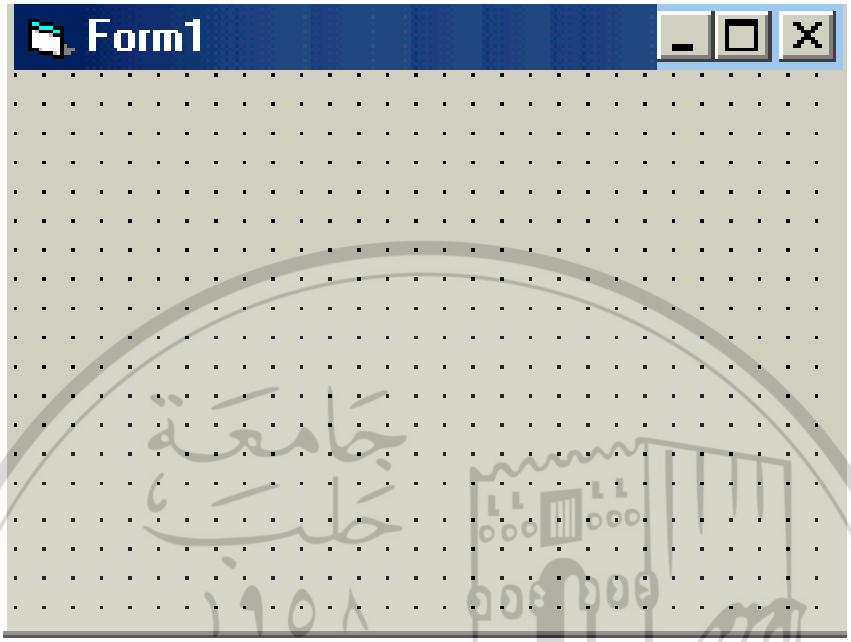
تغيير حجم الأداة:

لتغيير حجم الأداة ننقر عليها ليتم تنشيطها ثم ننقر فيتم ظهور المقابض حول الأداة للدلالة على تحديدها. نقترب بالمؤشر من أحد المقابض عندها سيتغير شكل المؤشر إلى سهم ذو رأسين. بعد ذلك ننقر بزر الماوس الأيسر ونقوم بتحريك الماوس عندها سنلاحظ تغيّر حجم المستطيل أثناء حركة الماوس، وعندما نصل إلى الحجم المطلوب نقوم بتحرير زر الماوس.

الإطار (النافذة – النموذج) Form:

وتمثل النافذة الرئيسية التي يتم التعامل من خلالها مع معطيات الإدخال والإخراج.

يمكن التحكم بها عن طريق قائمة الخصائص Properties التي تظهر على الشاشة عن طريق القائمة المنسدلة View الأمر Properties Window:



الشكل (٢٢) – نافذة الإطار Form

- النموذج كائن يعمل كحاوية للكائنات الأخرى كالعناوين ومربعات النص ومربعات الرسم التي تتكون منها في النهاية واجهة المستخدم.
- يحتوي النموذج على كل العناصر التي توجد على نوافذ البرنامج أثناء تشغيله، فهو يحتوي على شريط عنوان وقائمة التحكم وعدة أزرار للتحكم (تصغير، تكبير واستعادة وإغلاق).
- تظهر أرضية النموذج أثناء التصميم على هيئة شبكة نقطية Grid والتي تسمح بمحاذاة العناصر على النموذج.

التحكم في نوافذ النماذج:

يمكن التحكم في النموذج أثناء تصميمه من خلال سحبه من مقابض التحجيم في مرحلة التصميم أو بأن تحدد قيم الخاصتين Height, Width من مربع الخصائص كما يمكن التحكم في مكان وحجم النموذج أثناء تشغيل البرنامج (أي في المرحلة التنفيذية من خلال الأوامر المكتوبة بالمرحلة الكودية).

فتح نموذج آخر:

يظهر في بيئة تصميم VB داخل مربع المشروع أسماء النماذج المستخدمة في المشروع.

ولإظهار أي نموذج غير النموذج المعروض في بيئة التصميم ننقر نقراً مزدوجاً فوق اسم النموذج، عندها ستظهر نافذة النموذج الآخر بحجمها المناسب.

إظهار النموذج:

عندما يحتوي البرنامج على عدة نماذج فلا بد أن نعرف كيف ومتى تظهر النوافذ المختلفة ويتم التحكم في ذلك من خلال الحدثين Load, Unload والوظيفتين Show, Hide الخاصتين بالنافذة.

- تقوم العبارة Load بوضع النافذة في الذاكرة إلا أنها لا تظهرها على الشاشة. ويتم تحميل النافذة إذا تم ذكر اسمها ضمنياً عند طلب أحد خصائصها أو وظائفها وسيتمكن المستخدم بتيادل البيانات بين هذه النافذة والنوافذ الأخرى.
- لإظهار النافذة نستعمل الوظيفة Show وهي ستقوم بتحميل النافذة إذا لم يكن قد سبق تحميلها، أو ستقوم بإظهارها إذا تم تحميلها دون إظهارها سابقاً.
- هناك معامل اختياري يحدد حالة النافذة عند إظهارها (هل هي من النوع Modal أم Modeless). فإذا تمّ إظهار الإطار على شكل Modal عندها لن نتمكن من متابعة العمل في البرنامج أو الوصول إلى أي نافذة حتى يتم معالجة الأوامر الموجودة في هذه النافذة. أمّا إذا تمّ إظهار هذا النموذج على شكل Modeless، عندها نستطيع التفاعل مع النوافذ الأخرى أثناء ظهور هذه النافذة ويمكن أن تغطيها النوافذ الأخرى.
- إذا لم نعد بحاجة إلى نافذة ما، فيمكن استبعادها من الذاكرة وتحرير المساحة التي تشغلها بالأمر Unload.

التعامل مع أحداث النموذج:

هناك خمسة أحداث رئيسية بالنسبة للنموذج يمكن التعامل معها وهي:

- **Load:** ويحدث بعد تحميل النموذج في الذاكرة.

- **Activate**: ويحدث عند أول ظهور للنموذج ثم بعد ذلك عندما يتحول المستخدم إلى النافذة لتنشيطها.
- **Deactivate**: ويحدث عند تنشيط نموذج آخر من نفس البرنامج.
- **Unload**: ويحدث قبل إفراغ الذاكرة من النافذة.
- **Initialize**: ويقع مرة واحدة فقط لكل نموذج حتى إذا تم إفراغ الذاكرة منه ثم إعادة تحميله لأنه يقع عند تسجيل بيانات النافذة كصنف جديد من النوافذ.

الفصل الثاني
أنواع البيانات
Data Types

قسم التصاريح العامة General Declaration :

الأخطاء المحتملة عند عدم تعريف متحول ما كثيرة من أهمها أن البرنامج سيقوم بإعطاء القيمة 0 إلى أي قيمة غير معرفة مسبقاً.

تستخدم العبارة Option Explicit في قسم التصاريح العامة للنموذج وذلك لإجبار VB على عدم قبول أي متحولات غير مصرح عنها وتكتب أول عبارة في الإطار وقبل البرمجة الكودية لأي عنصر من العناصر.

```
Option Explicit
Dim A As Integer
Dim M(44) As Integer
Dim Par As String
```

إضافة تعليق أو ملاحظة Rem:

نستطيع في VB إضافة تعليقات ضمن نص البرنامج باستخدام الكلمة Rem من الكلمة Remark أو باستخدام رمز الفاصلة العلوية:

```
Rem This Is My First Program
'  ***  Hi Every Body  ***
'  This Program Designed By ……..
Rem This Program To Resolve The Equation  Ax2 + Bx + C = 0
```

تعتبر عادة التعليقات ضمن نص البرنامج عادة حسنة لأنها تسهل قراءة وتنقيح البرنامج وتحديثه وفهمه من قبل نفس المبرمج أو من قبل مستثمر جديد للبرنامج.

المتغيرات Variables:

المتحول هو اسم يأخذ قيم مختلفة أثناء تنفيذ البرنامج. أو هو اسم يمكن تغيير قيمته أثناء تنفيذ البرنامج، والمتغير عبارة عن مكان يتم حجزه في ذاكرة الحاسب ويخصص له اسم ويحمل قيمة قد تتغير أثناء تنفيذ البرنامج.

وللمتغيرات أنواع مختلفة فقد تكون (صحيحة، حقيقية، رمزية، ... الخ) ويمكن أن يكون لها أطوال مختلفة (Integer, Long, Single, Double, ...). وللتصريح عن متغير نحتاج إلى ثلاث قيم تحدده وهي:

```
Public Item_num As Integer
```

- اسم المتغير والذي يشير إلى موقعه في الذاكرة Item_num.
- نمط المتغير ويحدد تعامل المترجم معه والمساحة التخزينية المطلوبة له Integer.

أنواع البيانات Data Types:

يمكن للمتغير أن يحمل رقماً صحيحاً أو كسريّاً أو حقيقيّاً أو قيمة نصية في متغير حرفي أو أحد مكونات كائن مثل نافذة أو أداة تحكم.

١. **متحول صحيح Integer:**

يأخذ القيم الصحيحة من $-2^{15} = -32768$ وحتى $2^{15} = 32768$ وحجمه 2 byte ويأخذ 15 خانة. وكل رقم تتوقع قيمه خارج المجال (32768- , 32768) فلا يجب أن يعرف بالمتحول الصحيح لأن ذلك خطأ.

٢. **المتحول الطويل Long (صحيح وطويل):**

يأخذ القيم الصحيحة من $-2.1x10^9 = -2^{31}$ وحتى $2.1x10^9 = 2^{31}$ وحجمه 4 byte ويأخذ 10 خانات.

٣. **نوع Single:**

يأخذ قيم كسرية بدقة 6 خانات في الذاكرة بعد الفاصلة، ويأخذ 4 بايت في ذاكرة الحاسب.

٤. **المتحول الكسري Double:**

متحول كسري بـ 12 خانة بعد الفاصلة ويأخذ 8 بايت في الذاكرة.

٥. **متحول منطقي Boolean:**

يأخذ القيم (1 و 0) وحجمه 2 byte.

٦. **متحول حرفي String:**

كل حرف 1 بايت.

٧. **متحول من نوع Date تاريخ:**

يأخذ 8 بايت.

٨. **متحول الهدف:**

ويأخذ 8 بايت.

٩. **متحول عام Variant:**

حجمه 16 بايت. أي أنه عند تعريف المتحول Variant نستطيع تخزين أي متحول.

ملاحظة: يوجد متحولات تأخذ القيم (صح أو خطأ) أو (0 أو 1) وتعرف هذه المتحولات بالمتحولات المنطقية.

وبشكل عام يمكن استخدام متغيرات من أحد الأنواع التالية:

```
Dim item1 As Currency
Dim Date1 As Date
Dim item2 As Boolean
Dim item3 As Single
Dim item4 As Double
Dim item5 As Byte
Dim item6 As Integer
Dim item7 As Long
```

- لتخزين قيمة مالية يمكن اختيار النوع Currency.
- لتخزين قيم تواريخ يمكن اختيار النوع Date.
- لتخزين قيمة منطقية يمكن اختيار النوع Boolean.
- للمتغير الذي سيحمل قيمة رقمية بها كسور يمكن اختيار النوع Single أو Double. الأول عند عدم الحاجة إلى دقة كبيرة بل إلى سرعة في البرنامج والثاني للأعداد الضخمة والدقة العالية.
- أما إذا كان المتغير سيحمل قيمة رقمية ليس بها كسور فإنك تستخدم Byte أو integer أو long وهما مرتبين حسب الأقل في مدى الأرقام والمساحة التخزينية.

والجدول التالي يبين أنواع عديدة من البيانات التي يدعمها الفيجوال بيزك:

نوع البيانات	البايتات المطلوبة	المجال	الاختصار
Integer	2 Bytes	From -32,768	%

	To 32,767		
&	From -2,147,483,648 To 2,147,483,647	4 Bytes	Long
!	From 1.401298 E-45 To 3,402823 E38	4 Bytes	Single-positive
!	From -3,402823 E38 To -1.401298 E-45	4 Bytes	Single-negative
#	From 4,94065645841247 D-24 To 1,79769313486232 D308	8 Bytes	Double-positive
#	From - 1,79769313486232 D308 To -4,94065645841247 D-24	8 Bytes	Double-negative
@	From -922337203685477,5808 To 922337203685477,5807	8 Bytes	Currency
$	من صفر إلى حوالي 2 بليون حرف	يعتمد على عدد الأحرف	String
	From 1 / June / 100 To 31 / December / 9999	8 Bytes	Date
		يعتمد على نوع البيانات التي يخزنها.	Variant

✓ يمكن التصريح عن متحول بالإشارة إلى نوعه أو من خلال رمزه لذا يمكن القول:

Dim I As Integer

أو بالشكل:

Dim I %

إذ يعتبر الرمز % التمثيل المختصر للعبارة As Integer.

يأخذ المتحول من النوع الصحيح قيمة ما من ضمن المجال 32767- وحتى 32768.

تشغل كل قيمة صحيحة 2 Byte (بايتين) من الذاكرة.

✓ وبشكل مشابه يمكن التصريح عن متحول ما بأنه صحيح طويل كما يلي:

Dim N As Long

أو بالشكل:

Dim N &

الرمز & هو التمثيل المختصر للعبارة As Long.

يأخذ المتحول من النوع الصحيح الطويل أي قيمة من ضمن المجال 2,147,483,648- وحتى 2,147,483,648.

يشغل كل متحول طويل 4 Byte (4 بايتات) من الذاكرة.

- ✓ وبشكل مشابه يمكن التصريح عن متحول المضاعف الدقة كما يلي:

Dim Z As Double

أو بالشكل:

Dim Z #

المتحول المضاعف الدقة قد يكون رقماً موجباً أو سالباً ويشغل 8 Byte من الذاكرة.

- ✓ يمكن أن يشير المتحول Variant إلى تاريخ أو وقت أو سلسلة كتابية أو متحول ذي فاصلة عائمة.
  - فمثلاً عند التصريح عن متحول بالشكل:

Dim I As Integer

عندها سيشكل VB متحولاً يدعى I من نوع صحيح.

- أما عند التصريح عن متحول I بالشكل:

Dim I

فسوف يشكل VB متحول I من نوع متغير Variant أي أن VB لا يعلم بماذا سيستخدم المتحول I ، هل هو سلسلة أم متحول صحيح أم متحول طويل أم أي نوع آخر. فعندما نستخدمه بالعبارة I = "My String" سيعالج كسلسلة نصية وعندما نستخدمه بالعبارة I = 2 + 3 سيعالج كمتحول صحيح.

إن عيب استخدام البيانات Variant هو التسبب في بطء عمل البرنامج بالقياس مع البرامج التي تحدد نوع المتحول بدقة.

مدى استخدام المتغير:

يقصد بمدى استخدام المتغير ضمن الإجراءات والنماذج والملفات التي ستتأثر به أي الأماكن التي يمكن أن يستخدم فيها هذا المتغير داخل البرنامج. ففي بعض الأحيان نعرف متغيّراً ما ضمن وحدة برمجية معينة ويصبح صالحاً للاستخدام داخل هذه الوحدة البرمجية، وقد يتم تعريفه بشكل يصبح استخدامه ممكناً في كل الوحدات البرمجية ضمن النموذج الواحد وعندها لن نستطيع استخدامه في النماذج الأخرى من المشروع، ويمكن أن

يتم تعريفه بشكل أوسع وأعم بحيث يصبح استخدامه ممكنا داخل كل النماذج الموجودة في والوحدات البرمجية في المشروع.

١) المتغيرات العامة Public:

هي المتغيرات التي يمكن استخدامها من أي مكان داخل البرنامج وتبقى في ذاكرة الحاسب طوال فترة عمل البرنامج . ويتم الإعلان عنها باستخدام الكلمة public ويفضل الإعلان عنه في ملف برمجي Module.

$$Public\ mydata\ As\ String$$

٢) المتغيرات على مستوى الملف والنموذج Dim:

هي متغيرات تنفيذ بملف أو نموذج وتبقى في الذاكرة طوال فترة عمل الملف أو النموذج، ويتم الإعلان عنها باستخدام الكلمة Dim خارج أي إجراء من الإجراءات الموجودة في الملف أو النموذج.

٣) المتغيرات على مستوى الحدث Static:

وهي المتغيرات التي يتم التعرف عليها والتعامل معها ضمن الحدث الإجرائي نفسه ولا يتم التعامل معها خارج حدود هذا الحدث.

الثوابت Constants:

هي عبارة عن اسم يحمل قيمة ثابتة لا تتغير أثناء تنفيذ البرنامج، ويجب أن لا تظهر على يسار تعليمة تعيين لأن قيمتها لا تتغير، وبشكل عام ينطبق على اسم الثابت جميع الشروط التي تطبق على اسم المتحول.

تتشابه الثوابت مع المتغيرات في أمرين هما اسم الثابت ومجال استخدامه أو مداه. تتبع الثوابت نفس القواعد التي تحدد مدى المتغيرات حيث يحدد مدى الثابت بالمكان الذي تعلن فيه عن هذا الثابت.

يتم الإعلان عن الثوابت حسب الطريقة الموضحة.

$$\boldsymbol{Const\ pi\ double\ =\ 3.14}$$

Const: للدلالة على الثابت و Pi: اسم الثابت و Double نوع أو طراز الثابت و 3.14 قيمة الثابت.

أبجدية لغة الفيجوال بيزك:

تستخدم لغة الفيجوال بيزك عموماُ الأبجدية التالية:

١. جميع الأحرف الإنكليزية الكبيرة (A ... Z) والصغيرة (a ... z).
٢. الأرقام العربية العشرية (0 – 9) والأرقام الست عشرية (0 ... F).
٣. بعض الرموز الخاصـــة، كالرموز التي تدل على الأقواس مثلاً وعلامات الترقيم و@ و ... الخ.

الرموز التي تدل على نوع المعطيات Data-Type Suffixes:

نوع المتحول Data–type	الرمز الذي يدل عليه Suffix	الدلالة على المتحول (شرحه)
Integer	%	صحيح
Long-integer	&	صحيح طويل
Single-precision	!	ذو دقة بسيطة
Double-precision	#	ذو دقة مضاعفة
String	$	حرفي – نصّي

المعاملات الرياضية Mathematical Operators:

المعامل الرياضي	المعنى الرياضي
*	معامل الضرب
-	إشارة السالب
/	علامة القسمة
=	معامل المساواة
>	أكبر من
+	إشارة الموجب
.	الفاصلة العشرية
<	أصغر من
\	علامة القسمة الصحيحة
^	علامة الرفع إلى قوّة

الرموز الخاصّة Special Indexes:

الرمز	المعنى	الشرح
.	النقطة	وتستخدم للفصل بين اسم المتحول واسم خاصية أو صفة المتحول
'	الفاصلة العلوية	وتستخدم لكتابة سطر الملاحظات أو التعليق
" "	الحاصرتين العلويتين	وتستخدم لكتابة النصوص والتعليقات
;	الفاصلة المنقوطة	تستخدم للفصل بين أسماء المتحولات أو الوسائط أي تتحكم بصيغة وإخراج تعليمات الدخل والخرج
,	الفاصلة	تستخدم للفصل بين أسماء المتحولات أو الوسائط أي تتحكم بصيغة وإخراج تعليمات الدخل والخرج
:	النقطتين	تفصل بين التعليمات الواقعة على سطر واحد
?	إشارة الاستفهام	دليل يستخدم مع تعليمة الإدخال
_	إشارة تحت السطر	وتستخدم لإكمال تعليمة على أكثر من سطر

مواصفات البرنامج المكتوب بلغة البيزك:

١. يتألف البرنامج المكتوب بلغة الفيجوال بيزك Visual Basic من مجموعات عبارات وتعليمات يقوم الحاسب من خلالها بمعالجة المعطيات الموجودة للحصول على النتائج المطلوبة.

٢. يتم كتابة هذه التعليمات بمجموعة أسطر متتالية تشكل البرنامج. يتضمن كل سطر من هذه الأسطر تعليمه كاملة أو جزءاً من تعليمه تمتد على أكثر من سطر. ويمكن للسطر الواحد أن يحتوي على أكثر من تعليمه بشرط أن تفصل نقطتان علويتان ":" بين كل تعليمتين متتاليتين.

٣. تكتب التعليمات بأحرف إنكليزية كبيرة أو صغيرة دون فراغ بينهما. وتصاغ التعليمات من كلمات مأخوذة من اللغة الإنكليزية أو من اختصار لكلماتها مثل: - Print Input – If.

٤. تستخدم أثناء كتابة البرنامج المحارف والأرقام وعلامات الترقيم المختلفة ضمن تركيبات معينة تفرضها قواعد كتابة اللغة.

٥. يُكتب النص بواسطة محرر النصوص (Text Editor) الخاص المدمج في المترجم الفوري (Compiler) للغة والذي من خلاله تستطيع التعديل في البرنامج والإضافة إليه والحذف منه.

٦. يحدث لكل برنامج جديد ملف خاص به يكتب نصه فيه ويمكن تعديله وفق الحاجة.

وعند كتابة برنامجاً بلغة البيزك نقوم بالتسلسل التالي:

١. نقوم بكتابة البرنامج و تدقيقه و تعديل الأخطاء.

٢. نقوم بتدقيق البرنامج عن طريق Compiler فيكشف الأخطاء الموجودة .

٣. نجهز مثال محلول على ورق لنتأكد من صحة البرنامج.

٤. نستخدم تعليمة RUN أو نضغط مفتاح SHIFT+F5 معاً فيظهر على الشاشة إطار جديد مخصص لإظهار نتائج التنفيذ. بعد ذلك يمكن العودة إلى محرر نصوص البيزك للتعامل مع نص البرنامج من جديد وتعديله.

تسمية المتحولات:

تخضع تسمية المتحولات لعدة قواعد:

١. أن يبتدئ الاسم بحرف من حروف اللغة الإنكليزية بعدها يمكن أن يليه عدد من الحروف أو الأرقام أو كليهما معاً.

- (التسمية M55 صحيحة كون الاسم قد ابتدأ بحرف).
- (التسمية 5xy خاطئة كون الاسم ابتدأ برقم).

٢. ألا يتضمن فراغاً أو إشارة ترقيم و ينصح باستعمال التسميات ذات الدلالة الواضحة:

- PRICE تسمية صحيحة.
- MY BOOK تسمية خاطئة (لوجود فراغ ضمن الاسم).
- X.Y تسمية خاطئة (لوجود علامة ترقيم ضمن الاسم).

٣. يمكن استعمال الحروف الكبيرة والصغيرة في التسميات ولا فرق بينهما أي (ABC = abc).

٤. يجب تفادي استعمال الكلمات المكونة لمفردات لغة البرمجة والتي تسمى الكلمات المحجوزة Reserved Words كأسماء المتحولات مثل INPUT - PRINT .

٥. ينتهي اســم المتحول بمحدد نوع المتحول، فمثلاً A$ هو متحول نصـــي بينما B% متحول صـــــحيح، وإذا لم تتم كتابة نوع المتحول فأن المتحول يكون أحادي الدقة أي من النوع Single.

٦. ألا يكون مكرراً.

٧. أكبر طول ممكن لاسم متحول هو 255 حرفاً.

ملاحظة: يجب الانتباه إلى كتابة اسم المتحول بنفس الطريقة عند استخدامه في عدة أماكن وإلا سيقوم مترجم البيزك باعتباره متحولاً آخراً.

الإشارات الحسابية في لغة الفيجوال بيزك:

إن الإشارات أو العمليات الحسابية المستخدمة في لغة البيزك تختلف عن الإشارات المستخدمة في العمليات الحسابية:

العملية	الرمز	مثال
الجمع	+	x = a + b
الطرح	-	x = a - b
الضرب	*	x = a * b
القسمة العادية	/	x = a / b
القسمة الصحيحة	\	x = a \ b
الرفع إلى قوة	^	x = a ^ b

إشارات المقارنة في لغة البيزك:

إن إشارات المقارنة المستخدمة في لغة البيزك هي:

العملية	الرمز	مثال	الشرح
أكبر من	>	a > b	a أكبر من b
أصغر من	<	a < b	a أصغر من b
المساواة	=	a = b	a يساوي b
أكبر أو يساوي	>=	a >= b	a أكبر أو يساوي b
أصغر أو يساوي	<=	a <= b	a أصغر أو يساوي b
لا يساوي (ليست أكبر وليست أصغر)	<>	a <> b	a لا يساوي b

العمليات الحسابية وأولوية تنفيذها بلغة البيزك:

أثناء حل المعادلات الحسابية ننطلق وفقاً للقواعد التالية:

١) الأقواس:

إن الأقواس لا تمثل أي عملية حسابية لكنها تستخدم لتحديد أن العملية الموجودة بين قوسين يتم تنفيذها أولاً (أي تأخذ الأولوية) وفي حال تعدد الأقواس يتم فك الأقواس الداخلية أولاً ثم الأقواس الخارجية. وتستخدم الأقواس الصغيرة فقط ولا يجوز استخدام الأقواس المتوسطة والكبيرة وفي حال وجود أقواس تحمل نفس الأفضلية يتم البدء من اليسار إلى اليمين.

٢) تعتبر الأولوية لتنفيذ العمليات الحسابية كما يلي:

١. الرفع إلى قوة.
٢. الضرب والقسمة ولهما نفس القوة.
٣. القسمة الصحيحة (/) إي إهمال الكسور أو القيم العشرية.
٤. باقي القسمة (Mod) والذي يعطي باقي القسمة.
٥. الجمع والطرح ولهما نفس القوة.
٦. إشارة الدمج (&) والتي تقوم بجمع المتحولات النصيّة.
٧. عمليات المقارنة: < ، > ، =< ، >= ، <> ، = .
٨. العمليات المنطقية: (ليس Not) ، (أو Or) ، (و And).

في حال تماثل العمليات في سلم الأفضليات (أي عندما يكون لها نفس الأولوية) فإن البرنامج يقوم بالتنفيذ بدءاً من اليسار إلى اليمين.

العمليات الحسابية وأولويتها		
Exponentiation	^	الرفع إلى قوة
Multiplication	*	الضرب
Floating point Division	/	القسمة
Integer Division	\	القسمة الصحيحة
Addition	+	الجمع
Subtraction	-	الطرح

قواعد الكتابة:

عند كتابة التعابير الرياضية يجب الانتباه إلى ما يلي:

- يجب ألا نستخدم في التعبير الرياضي إشارتين حسابيتين متتاليتين:

$$x = a +* b \qquad z = 5\text{^} - 2$$

- يجب إغلاق جميع الأقواس أي أنه يجب أن يكون عدد الأقواس اليسارية مساوياً للأقواس اليمينية.
- إن التعبير $\mathbf{a/b * c}$ يعادل $\mathbf{a * c/b}$ إذ يقوم الحاسب أولاً بعملية التقسيم من اليسار فيقسم $\mathbf{a/b}$ ثم يضرب الناتج بالمتحول $\mathbf{c}$.
- إن التعبير $\mathbf{a/(b * c)}$ يعادل $\frac{a}{b*c}$ إذ يقوم الحاسب أولاً بعملية الضرب $\mathbf{(b * c)}$ ثم يقوم بتقسيم $\mathbf{a}$ على ناتج عملية القسمة.
- كما يمكن كتابة التعبير على الشكل التالي $\mathbf{a/b/c}$ حيث يقوم أولاً بعملية التقسيم $\mathbf{a/b}$ ثم يقسّم الناتج على $\mathbf{c}$.
- إن التعبير $\mathbf{a + b/c * d}$ يعادل $a + \frac{b*d}{c}$ حيث يقوم الحاسب أولاً بعملية التقسيم $\mathbf{b/c}$ والناتج يضربه بالمتحول $\mathbf{d}$ ومن ثم يقوم بإضافة $\mathbf{a}$.

المتحولات المنطقية والعمليات المنطقية:

المتحول المنطقي:

هو متحول يأخذ فقط إحدى القيمتين "صح" أو "خطأ". نرمز للصح بـ T من كلمة True وللخطأ بـ F من كلمة False. فإذا كان لدينا تعبيران حسابيان وقمنا باستخدامهما بعملية مقارنة فالنتيجة ستكون متحولا منطقياً وسيأخذ إحدى القيمتين إما T أو F . وإذا رمزنا للمتحول المنطقي بـ R وباستخدام عملية المقارنة نجد:

- $\mathbf{R = (a = b)}$ فإنه سيكون $\mathbf{R = T}$ إذا كان $\mathbf{a = b}$.

$\mathbf{R = F}$ إذا كان $\mathbf{a = b}$.

أي أن نتيجة R ستكون صحيحة عندما a تساوي b فقط وغير ذلك ستكون خاطئة.

- $\mathbf{R = (a < b)}$ فإنه سيكون $\mathbf{R = T}$ إذا كان $\mathbf{a < b}$.

$\mathbf{R = F}$ إذا كان $\mathbf{a > b}$.

أي أن نتيجة R ستكون صحيحة عندما a أصغر تماماً من b فقط وغير ذلك ستكون خاطئة.

العمليات المنطقية:

يمكن أن نربط المتحولات المنطقية بعمليات منطقية مثل النفي والتقاطع والجمع وعمليات أخرى كما هو مبيَّن:

١) النفي أو "لا" ونرمز لها بـ ~ أو Not:

إذا كان لدينا متحول منطقي $\boldsymbol{R}$ فإن نفي $\boldsymbol{R}$~ هو متحول منطقي آخر ويعرف كما يلي:

R	Not R ~R
F	T
T	F

٢) التقاطع يرمز له بـ ∩ أو And ويعرف كما يلي:

إذا كان لدينا متحولان منطقيان $\boldsymbol{R, S}$ فان تقاطعهما $\boldsymbol{Q}$ هو متحول منطقي آخر يعرف كما يلي:

R	S	Q = R And S
T	T	T
T	F	F
F	T	F
F	F	F

أي أن النتيجة ستكون صحيحة فقط عند تقاطع وتحقق جميع الشروط.

٣) الجمع يرمز له بـ ∪ أو Or ويعرف كما يلي:

إذا كان لدينا متحولان منطقيان $\boldsymbol{R, S}$ فان جمعهما $\boldsymbol{Q}$ هو متحول منطقي آخر يعرف كما يلي:

R	S	Q = R Or S
T	T	T
T	F	T
F	T	T
F	F	F

أي حتى تكون النتيجة صحيحة يكفي فقط تحقق أحد هذه الشروط.

٤) العملية Xor:

تكون نتيجة العملية ***True*** إذا كان واحد من التعبيرين التاليين صحيحاً وتكون النتيجة خطأ ***False*** إذا كان كلاهما صح أو خطأ.

R	S	Q = R Xor S
T	T	F
T	F	T
F	T	T
F	F	F

أي أن النتيجة ستكون صحيحة عند اختلاف الشروط.

٥) العملية Eqv:

تكون نتيجة العملية ***False*** إذا كان واحد من التعبيرين التاليين صحيحاً وتكون النتيجة ***True*** إذا كان كلاهما صح أو خطأ.

R	S	Q = R Eqv S
T	T	T
T	F	F
F	T	F
F	F	T

أي أن النتيجة ستكون صحيحة عند توافق الشروط كأن تكون جميعها محققة أو جميعها غير محققة.

أمثلة:

مثال (١): أوجد ناتج ما يلي: $X = \frac{8}{4} + 15\ x\ 2 - 3^3$

1- $X = 8/2 + 15 * 2\ - 27$

2- $X = 2 + 15 * 2\ - 27$

3- $X = 2 + 30 - 27$

4- $X = 32 - 27 = 5$

الأولوية هي االرفع إلى قوة ثم لقسمة ثم الضرب (لأن القسمة أقرب إلى إشارة المساواة) بعد ذلك الجمع ثم الطرح (لأن الجمع أقرب إلى إشارة المساواة).

مثال (٢): أوجد ناتج ما يلي: $Y = (9 + 3)^2$

1- $Y = 12\text{^}2$

2- $Y = 144$

مثال (٣): اكتب العبارة التالية على الحاسب: $Y = \frac{X+1}{X} + 5$

1- $Y = (X + 1)/\ X\ + 5$ عبارة صحيحة

2- $Y = \big((X + 1)/\ X\big) + 5$ عبارة صحيحة

3- $Y = X + 1/X\ + 5$ عبارة خاطئة

لأن العبارة الثالثة تعبر عن المعادلة $Y = X + \frac{1}{X} + 5$

مثال (٤): اكتب العبارة التالية على الحاسب: $Z = \frac{X}{Y+3}$

1- $Z = X/(Y + 3)$ عبارة صحيحة

2- $Z = X/Y + 3$ عبارة خاطئة

مثال (٥): حل المعادلة التالية $5^2 + \frac{20}{4} - 3^2$ **حسب أولوية العمليات فيها:**

الرفع إلى قوة والجمع والقسمة والطرح والقوة	$5^2 + \frac{20}{4} - 3^2$
١. الرفع إلى قوة من جهة اليسار.	5^2 + 20/4 - 3^2
٢. الرفع إلى قوة من جهة اليمين.	1. 25 + 20/4 - 3^2
٣. القسمة أقوى.	2. 25 + 20/4 – 9
٤. الجمع أقوى (من اليسار).	3. 25 + 5 – 9
٥. الطرح	4. 30 – 9
	5. 21

مثال (٦): حل المعادلة التالية $5^2 + \left(\frac{20}{4}\right) - 3^2$ حسب أولوية العمليات فيها:

الرفع إلى قوة والجمع والقسمة والطرح والقوة ١. الأقواس لها الأولوية. ٢. الرفع إلى قوة من جهة اليسار ٣. الرفع إلى قوة من جهة اليمين. ٤. الجمع أقوى (من اليسار). ٥. الطرح	$5^2 + \left(\frac{20}{4}\right) - 3^2$ 5^2 + (20/4) - 3^2 1. 5^2 + 5 - 3^2 2. 25 + 5 – 3^2 3. 25 + 5 – 9 4. 30 – 9 5. 21

لقد استحوذت عملية القسمة على الأولوية بسبب وجود الأقواس، ولولا وجود الأقواس لكانت الأولوية للرفع إلى قوة.

تظهر المعادلة ضمن كود الفيجوال بيزك كما في الشكل:

```
5 ^ 2 + (20 / 4) - 3 ^ 2

1)  5 ^ 2 + 5 - 3 ^ 2
2)  25 + 5 - 3 ^ 2
3)  25 + 5 - 9
4)  30 - 9
5)  21
```

مثال (٧): حل المعادلة التالية حسب أولوية العمليات فيها:

```
x = 25 + (10 ^ 2) - (30 \ 4 + 2) + 3 ^ (5 - 2)
```

$x = 25 + (10\text{^}2) - (30\backslash 4 + 2) + 3\text{^}(5 - 2)$

$x = 25 + 100 - (30\backslash 4 + 2) + 3\text{^}(5 - 2)$

$x = 25 + 100 - (7 + 2) + 3\text{^}(5 - 2)$

$x = 25 + 100 - 9 + 3\text{^}(5 - 2)$

$x = 25 + 100 - 9 + 3\text{^}3$

$x = 25 + 100 - 9 + 27$

$x = 125 - 9 + 27$

$x = 116 + 27$

$x = 143$

مثال (٨): حل المعادلة التالية حســب أولوية العمليات فيهاإذا كانت عميلتي القسمة (عاديتين – صحيحتين – عادية وصحيحة – صحيحة وعادية):

$$x = 8^2 + \frac{7.4}{2} + \frac{17.6}{2} + \sqrt{9}$$

يمكن كتابة المعادلة بأحد الأشكال التالية:

$x = 8\text{^}2 + 7.4/2 + 17.6/2 + SQR(9)$

$x = 8\text{^}2 + 7.4/2 + 17.6/2 + 9\text{^}(1/2)$

$x = 8\text{^}2 + 7.4/2 + 17.6/2 + (9\text{^}(1/2))$

يمكن حل المعادلة بالحالات الأربع كما يلي:

صحيحة ثم عادية:

$x_1 = 8\ \text{^}\ 2 + 7.4 \setminus 2 + 17.6 / 2 + 9\ \text{^}\ (1 / 2) \Rightarrow x_1 = 78.8$

صحيحتين:

$x_2 = 8\ \text{^}\ 2 + 7.4 \setminus 2 + 17.6 \setminus 2 + 9\ \text{^}\ (1 / 2) \Rightarrow x_2 = 79$

عاديتين:

$x_3 = 8\ \text{^}\ 2 + 7.4 / 2 + 17.6 / 2 + 9\ \text{^}\ (1 / 2) \Rightarrow x_3 = 79.5$

عادية ثم صحيحة:

$x_4 = 8\ \text{^}\ 2 + 7.4 / 2 + 17.6 \setminus 2 + 9\ \text{^}\ (1 / 2) \Rightarrow x_4 = 79.7$

مثال (9): حل المعادلة التالية حسب أولوية العمليات فيهاإذا كانت عميلتي القسمة (عاديتين – صحيحتين – عادية وصحيحة – صحيحة وعادية):

$$x = 8^2 * \left(\frac{7.4}{2} + \frac{17.6}{2}\right) + \sqrt{9}$$

يمكن كتابة المعادلة بأحد الأشكال التالية:

$x = 8\text{^}2 * (7.4/2 + 17.6/2) + SQR(9)$

$x = 8\text{^}2 * (7.4/2 + 17.6/2) + 9\text{^}(1/2)$

يمكن حل المعادلة بالحالات الأربع كما يلي:

$x_1 = 8\text{^}2 * (7.4/2 + 17.6/2) + SQR(9) = 803$

$x_2 = 8\text{^}2 * (7.4\setminus 2 + 17.6\setminus 2) + SQR(9) = 771$

$x_3 = 8\text{^}2 * (7.4/2 + 17.6\setminus 2) + SQR(9) = 815.8$

$x_4 = 8\text{^}2 * (7.4\setminus 2 + 17.6/2) + SQR(9) = 758.2$

مثال (10): حل المعادلة التالية حســب أولوية العمليات فيهاإذا كانت عميلتي القسمة (عاديتين – صحيحتين – عادية وصحيحة – صحيحة وعادية):

$$x = 6^2 + \frac{7.4}{2} + \frac{17.6}{2} + \sqrt{25}$$

يمكن كتابة المعادلة بأحد الأشكال التالية:

$x = 6\text{^}2 + 7.4/2 + 17.6/2 + 25\text{^}(1/2)$

$x = 6\text{^}2 + 7.4/2 + 17.6/2 + SQR(25)$

يمكن حل المعادلة بالحالات الأربع كما يلي:

عاديتين	$X = 53.5$	$X = 6\text{^}2 + 7.4/2 + 17.6/2 + 25\text{^}(1/2)$ $X = 6\text{^}2 + 7.4/2 + 17.6/2 + 25\text{^}(1/2)$ $X = 36 + 3.7 + 8.8 + 5$
صحيحتين	$X = 53$	$X = 6\text{^}2 + 7.4\backslash 2 + 17.6\backslash 2 + 25\text{^}(1/2)$ $X = 6\text{^}2 + 7\backslash 2 + 18\backslash 2 + 25\text{^}(1/2)$ $X = 36 + 3 + 9 + 5$
عادية ثم صحيحة	$X = 53.7$	$X = 6\text{^}2 + 7.4/2 + 17.6\backslash 2 + 25\text{^}(1/2)$ $X = 6\text{^}2 + 7.4/2 + 18\backslash 2 + 25\text{^}(1/2)$ $X = 36 + 3.7 + 9 + 5$
صحيحة ثم عادية	$X = 52.8$	$X = 6\text{^}2 + 7.4\backslash 2 + 17.6/2 + 25\text{^}(1/2)$ $X = 6\text{^}2 + 7\backslash 2 + 17.6/2 + 25\text{^}(1/2)$ $X = 36 + 3 + 8.8 + 5$

مثال (11): أكتب بلغة البيزك المعادلة التالية:

$$y = \frac{m + \sqrt{|Cos^3(\beta + \alpha)|}}{\sqrt{n}.(Cos\beta^3 + Sin2\alpha)}$$

يمكن كتابة المعادلة بالشكل التالي:

$$Y = (M + SQR(ABS(COS(B + A)\text{^}3)))/SQR(N)/(COS(B\text{^}3) + SIN(2 * A))$$

الفصل الثالث
تعليمات وأدوات الإدخال والإخراج
Input and Output Instructions and Tools

تعليمات الإدخال والإخراج Input and Output Instructions:

مربع أو تابع الإدخال *InputBox*:

تابع يستخدم من أجل الإدخال، حيث يظهر صندوق إدخال يتم إدخال قيمة المتحول ضمنه. والصيغة العامة لهذا التابع:

$$Var = InputBox(Prompt\ [,Title][,Default][,Xpos][,Ypos]\ [,HelpFile,Context])$$

- ***Prompt*** – تعبير رمزي يعرض كرسالة في صندوق الحوار، وهو إجباري.
- ***Title*** – تعبير رمزي يعرض كعنوان صندوق الحوار، وإذا لم تُذكر هذه القيمة فيعرض اسم المشروع بدلاً منها.
- ***Default*** – تعبير رمزي يعرض في خانة النص كقيمة قياسية، وإذا حذفت تكون الخانة فارغة.
- ***Xpos, Ypos*** – تعبيران عدديان يحددان موضع صندوق الحوار من حواف الشاشة.
- ***HelpFile*** – تعبير رمزي يعرف ملف المساعدة، إذا ذكر، يجب ذكر ***Context***.
- ***Context*** – تعبير عددي يدل على رقم موضوع المساعدة ضمن ملف مساعدة يحدده المبرمج.
- ***Var*** – القيمة التي يأخذها التابع بعد أن يضغط المستخدم على أحد الزرين ***Ok, Cancel***.

يمكن استخدام التابع الوظيفي لتلقي معلومات كتابية من قبل المستخدم. يُظهِر التابع الوظيفي ***InputBox*** ( ) مربع حوار مع رسالة وزرين هما ***Ok*** و ***Cancel*** ويستطيع المستخدم إدخال نص ما في الحقل النصي وإغلاق مربع الحوار بالنقر على الزر ***Ok***.

لدى الضغط على الزر ***Ok*** أو عند الضغط على المفتاح ***Enter*** يأخذ المتحول القيمة التي تم إدخالها في خانة النص. وعند الضغط على الزر ***Cancel*** فإن المتحوّل

يأخذ القيمة الفارغة " ". حتى لو كان قد تم إدخال قيمة ما ضمن خانة الإدخال. أي باختصار يعيد التابع الوظيفي ***InputBox*** ( ) ما أدخله المستخدم في الحقل النصّي الافتراضي ***Default***. إن القيمة المعادة تساوي لا شيء "***Null***" إذا لم يكتب المستخدم أي شيء في مربع الإدخال أو عند الضغط على زر ***Cancel*** "إلغاء الأمر".

إذا كانت الرسالة أو العنوان باللغة العربية فيمكن أن تظهر بعض المشاكل في العرض، ويمكن التغلب على ذلك بتخزين الرسالة وعنوان الصندوق ضمن متغيرات رمزية والتعامل معها.

Msg1$ = "أدخل الاسم " : Msg2$ = "أدخل المعلومات "

$$MyName\$ = InputBox(Msg1\$, Msg2\$, "DrMHammad", 500,500, "Demo.Hlp", 10)$$

$$MyName\$ = InputBox(Msg1\$, Msg2\$, "Dr M Hammad", 500,500)$$

طبعاً يمكن مقارنة البارامترات والوسائط مع الصيغة العامة:

$$Var = InputBox(Prompt\ [,Title][,Default][,Xpos][,Ypos][,HelpFile,Context])$$

- تبعد الحافة اليسارية لمربع الحوار بمقدار **500 *twips*** عن الحافة اليسارية للشاشة.
- تبعد الحافة العلوية لمربع الحوار بمقدار **500 *twips*** عن قمة للشاشة (حافتها العليا).

يمكن تستخدم العبارة If للتحقق من شرط الإدخال ومن القيمة المُدخَلَة وبالتالي للتحكم في سلوك البرنامج. ويمكن أن تستخدم القيم المُدخلة بشكلها الرمزي (النصّي) أو العددي:

```
Dim y As String
y = InputBox("Dr M Hammad","Aleppo University","Color",200,400)
If y = "Red" Then
 Form1.BackColor = vbRed
ElseIf y = "Green" Then
 Form1.BackColor = vbGreen
ElseIf y = "Blue" Then
 Form1.BackColor = vbBlue
End If
```

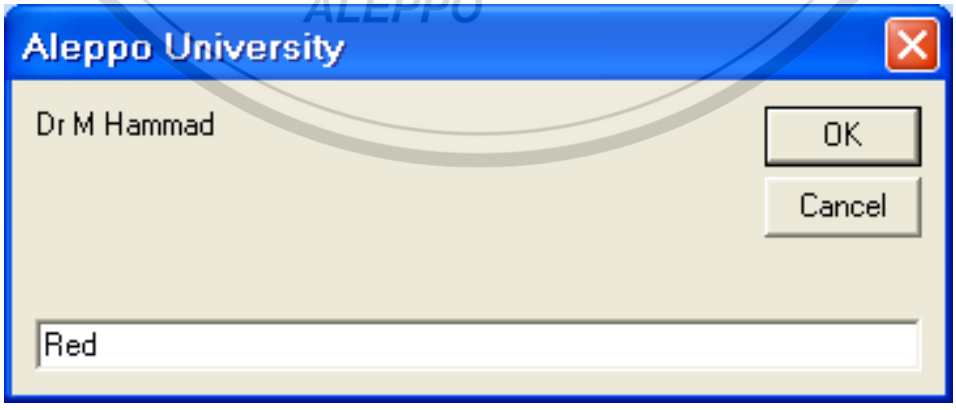
طبعاً يمكن مقارنة البارامترات والوسائط مع الصيغة العامة:

$$Var = InputBox(Prompt\ [,Title][,Default][,Xpos][,Ypos][,HelpFile,Context])$$

مع ملاحظة أنَّه يمكن إهمال الوسائط والعبارات الموجودة بين الأقواس المتوسطة:

يمكن قبول القيمة الإفتراضية:

أو إدخال قيمة أخرى بدلاً من القيمة الإفتراضية (اللون الأحمر):

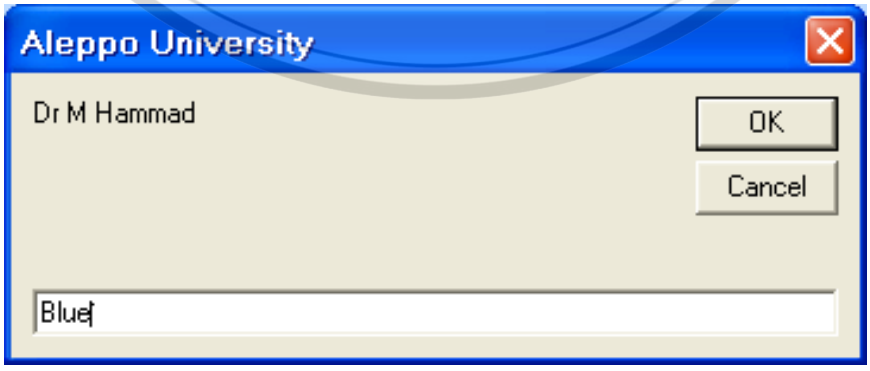


عندها ونتيجة تطبيق الكود سنحصل على النتيجة وهي هنا اللون الأحمر:

وإذا تم إدخال قيمة أخرى بدلاً من القيمة الإفتراضية (اللون الأخضر):

عندها ستكون نتيجة تطبيق الكود وهو هنا اللون الأخضر:

أو يتم إدخال قيمة أخرى بدلاً من القيمة الإفتراضية (اللون الأزرق):



عندها ستكون نتيجة تطبيق الكود وهو هنا اللون الأزرق كما هو مبيَّن:

القيمة الافتراضـــية التي ســـتظهر في مربع الحوار هي $Color$ ولدينا ثلاثة احتمالات للإجابة:

١) الآن يمكن أن نوافق عليها بالضغط على موافق أو $\boldsymbol{Enter}$.

٢) أو أن نكتب قيمة جديدة ثم نوافق عليها بالضغط على موافق أو $\boldsymbol{Enter}$.

٣) عند الضـــغط على المفتاح $Cancel$ أو إلغاء الأمر ســـيتم إدخال قيمة فارغة " " بغض النظر عن القيمة الموجودة حاليّاً في مربع الإدخال.

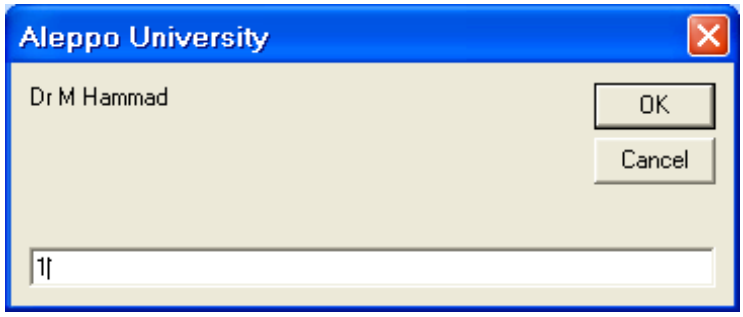
تبعد الحافة اليســارية لمربع الحوار بمقدار $200\ twips$ عن الحافة اليســارية للشــاشــة.

وتبعد الحافة العلوية لمربع الحوار بمقدار $400\ twips$ عن قمة للشاشة (حافتها العليا).

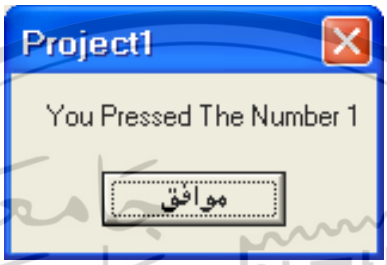
```
Dim y As Integer
y = InputBox("Dr M Hammad","Aleppo University","Integer",200,400)
If y = 1 Then
   MsgBox("You Pressed The Number 1")
Else
   MsgBox("You Pressed The Another Number")
End If
```

ســـيتوقف البرنامج عند مربع الإدخال الذي يطلب قيمة ســـيتم إظهارها عن طريق مربع إخراج أو مربع رسالة كما هو مبيَّن:

إذا أدخلنا الرقم **(1)** بدلاً من القيمة الإفتراضية $\boldsymbol{Integer}$:



سيتم تنفيذ الكود الأول من التعليمة الشرطية ***If*** وسيظهر مربع الرسالة كما هو مبيَّن:



إذا أدخلنا الرقم **(23)** بدلاً من القيمة الإفتراضية ***Integer***:

سيتم تنفيذ الكود الثاني من التعليمة الشرطية ***If*** وسيظهر مربع الرسالة كما هو مبيَّن:

أمثلة أخرى للتابع *InputBox*( ):

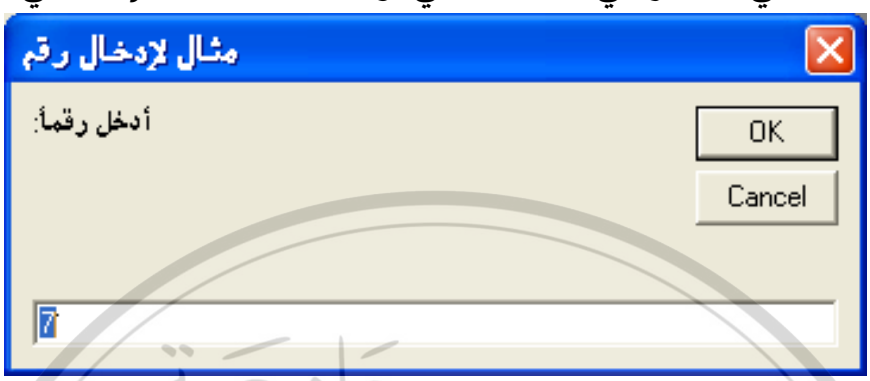
يجب أن الصيغة العامة لتابع الإدخال هي كما يلي:

$$Var = InputBox(Prompt\ [,Title][,Default][,Xpos][,Ypos]\ [,HelpFile,Context])$$

يمكن استخدام وسائط اختيارية أخرى مع التابع المذكور كما يلي:

Number = InputBox ("أدخل رقماً:", "مثال لإدخال رقم", "7", 100, 200)

- الرسالة التي سيتم إظهارها في الخاصية ***Prompt*** هي "أدخل رقماً:". أما عنوان مربع الإدخال والذي سيتم إظهاره في الخاصية ***Title*** هي "مثال لإدخال رقم". والقيمة الافتراضية التي ستظهر في الحقل النصي أو الخاصية ***Default*** هي "7" .

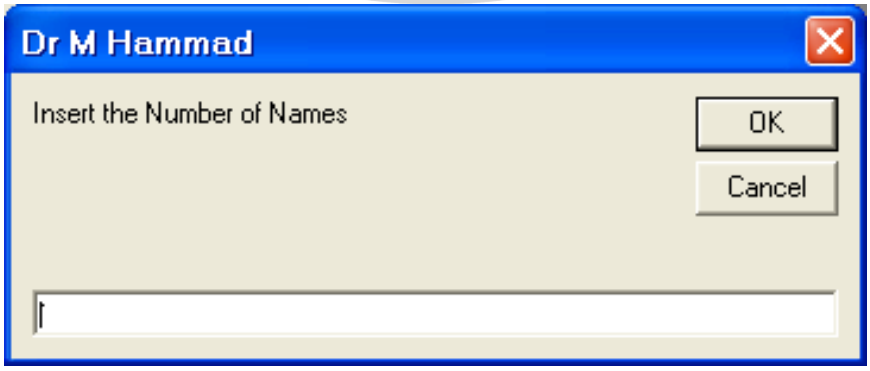


- تبعد الحافة اليسارية لمربع الحوار بمقدار $x = 100\ twips$ عن الحافة اليسارية للشاشة. كما تبعد الحافة العلوية لمربع الحوار بمقدار $y = 200\ twips$ عن قمة للشاشة.
- يمكن للمستخدم النقر على ***Ok*** في مربع الحوار أو يمكنه إدخال رقماً آخر غير الرقم 7 ثم ينقر على ***Ok***.

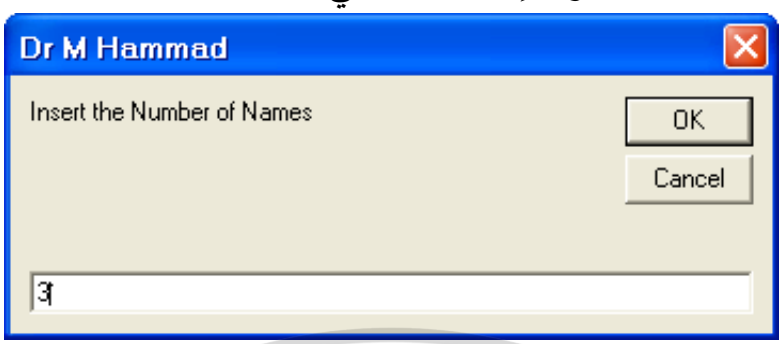
يمكن استخدام التابع لإدخال مجموعة من القيم من خلال حلقات التكرار كما يلي:

```
n = InputBox("Insert the Number of Names","Dr M Hammad")
For I = 1 To n
  Nam = InputBox("Insert the Name","the Names")
  List1.AddItem Nam
  List2.AddItem Nam
Next
```

- عند البدء بالمرحلة التنفيذية يظهر مربع الحوار التالي والذي ينفذ فيه السطر الأول أي يطلب عدد الأسماء المطلوبة:

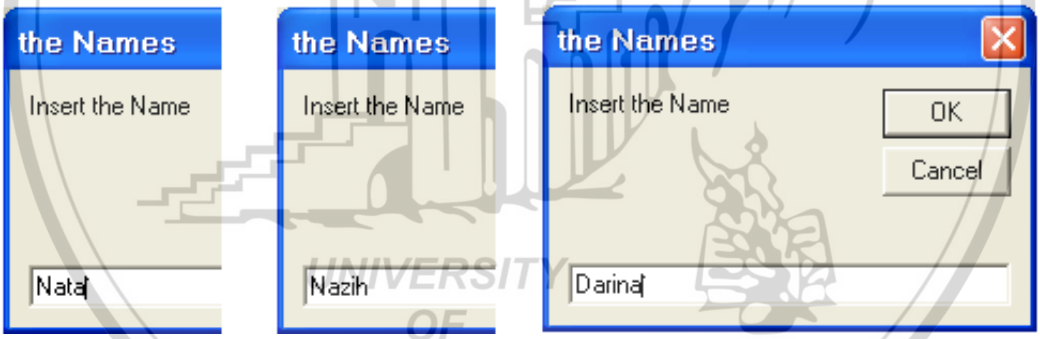


- ندخل عدد الأسماء المطلوب إدخالها كما يلي:



- ندخل الاسم الأول كما يلي:

- ثم ندخل الاسم الثاني والثالث كما يلي:



- تظهر النتيجة كما يلي:

- نلاحظ هنا أن الخاصية $\boldsymbol{Sorted = False}$ في صندوق اللائحة الأول $\boldsymbol{List1}$ لذا لم يتم ترتيب الأسماء.
- ونلاحظ أن الخاصية $\boldsymbol{Sorted = True}$ في صندوق اللائحة الأول $\boldsymbol{List2}$ لذا تم ترتيب الأسماء.

ملاحظة:

من خلال عملية الإدخال تبيّن أنه عند إدخال الأسماء كانت العبارات المرافقة دوماً هي العبارة ***the Names*** في الخاصية أو الوسيط ***Title*** والعبارة ***Insert the Name*** في الخاصية أو الوسيط ***Prompt***.

إن هذا النوع من الإدخال جامداً جداً وقد يوقعنا في كثير من الأخطاء عندما يكون عدد الأسماء المطلوب إدخاله كبيراً جداً، عندها سنقع في حيرة دائمة عندما نتساءل عن الاسم المطلوب إدخاله (أي ما هو ترتيبه) لذا نجد من الضروري ومن المفيد جداً أخذ قيمة ديناميكية تعبّر عن تسلسل الاسم المطلوب إدخاله وتجعل العملية مرتبطة ديناميكيّاً بعداد حلقة التكرار. لأجل ذلك نرى من الضروري إدخال عدّاد حلقة التكرار ضمن وسائط مربع الإدخال.

يمكن أن تتم عملية الربط في عبارة الوسيط ***Prompt*** أو في عبارة الوسيط ***Title*** بشكل من الأشكال بحيث يحتوي الوسيط على جزأين، الجزء الأساسي رمزي (سلسلة نصية) والجزء الثاني متعلق بعدّاد حلقة التكرار ويتم الربط بينهما من خلال الرموز التي تسمح بربط السلاسل ببعضها البعض (الرمز & مثلاً). في هذا الحالة عند إجراء عملية الإدخال سيتكرر الجزء الأساسي النصّي بشكل دائم ويتغيّر الجزء المتعلق بعدّاد حلقة التكرار مع تغيّر قيمته في كل مرحلة من مراحل تنفيذ هذه الحلقة.

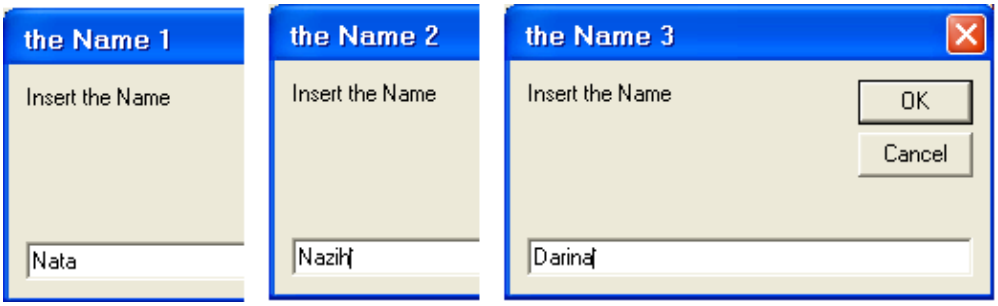
يمكن أن تتم عملية الربط بعدة أشكال وعلى سبيل الأمثلة نقترح تبديل السطر المحدد في العبارة الكودية بأحد الأسطر التالية وستكون النتيجة كما هو مبيّن في مربعات الإدخال المبيّنة:

$$Nam = InputBox("Insert\ the\ Name", "the\ Names")$$

- الصيغة الأولى المُقتَرَحَة للتعديل:

$$Nam = InputBox("Insert\ the\ Name", "\ the\ Name\ " \ \&\ I)$$

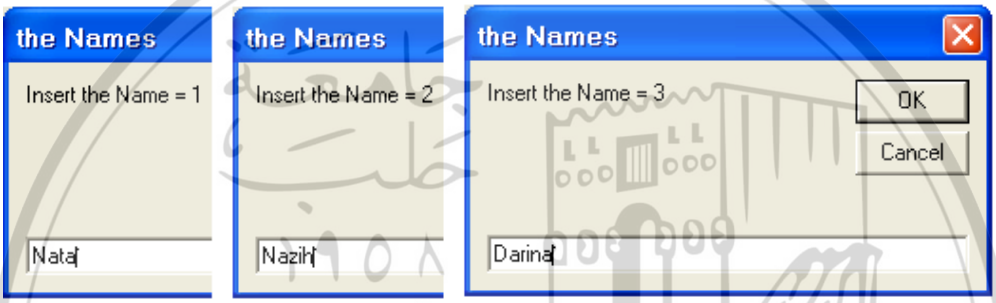
- تظهر مربعات الإدخال كما يلي:



- الصيغة الثانية المُقتَرَحَة للتعديل:

$$Nam = InputBox("Insert\ the\ Name = " \& I, "the\ Names")$$

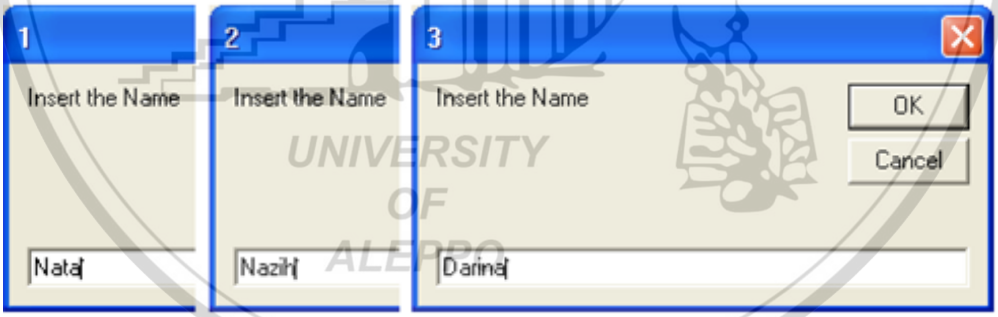
- تظهر مربعات الإدخال كما يلي:



- الصيغة الثالثة المُقتَرَحَة للتعديل:

$$Nam = InputBox("Insert\ the\ Name", I)$$

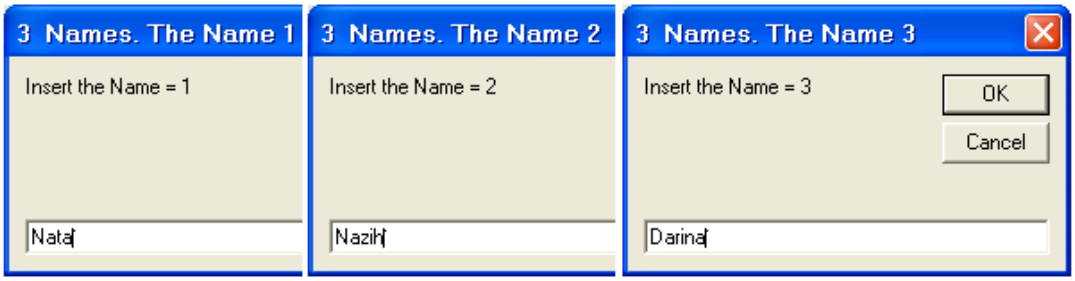
- تظهر مربعات الإدخال كما يلي:



- الصيغة الرابعة المُقتَرَحَة للتعديل:

$$Nam = InputBox("Insert\ the\ Name = " \& I, n \& " Names." \& "The\ Name " \& I)$$

- تظهر مربعات الإدخال كما يلي:



مربع أو تابع الإظهار *MsgBox*:

هو تابع يستخدم لإعطاء المستخدم فرصة لاتخاذ قرار ما بخصوص أمر ما، حيث يتم عرض صندوق الحوار والذي يتضمن رسالة معيّنة، تُظهر هذه الرسالة نصّاً مناسباً ومجموعة من الأزرار والتي تبيّن الخيارات المتاحة أمام المستخدم.

إذا يمكن استخدام الأمر ***MsgBox*** لإظهار رسائل للمستخدم كرسائل التأكيد والتنبيه والترحيب وغير ذلك. وحسب الرسالة التي تمّ إظهارها يكون لدينا احتمالين:

١) قراءة الرسالة الموجودة فقط ولا يوجد حاجة لمعرفة رد فعل المستخدم. وهنا يتم قراءة الرسالة والموافقة عليها من خلال الضغط على الخيار الوحيد الظاهر وهو الموافقة وبعد ذلك متابعة خطوات البرنامج حسب التعليمات التي تلي هذه التعليمة.

٢) قراءة الرسالة والتي على ضوءها نحتاج لمعرفة رد المستخدم والذي سيتم من خلال النقر على أزرار إضافية تظهر على الأداة.

<u>الصيغة الأولى:</u>

إن الشكل العام لهذه الصيغة كما في الشكل التالي:

$MsgBox$ (" $Prompt\ String$")

- حيث ***Prompt String*** هي عبارة نصيّة يتم كتابتها بين إشارتي التنصيص. ستظهر هذه العبارة داخل مربع الحوار ويظهر معها زر موافق ***Ok***. بعد قراءة العبارة والضغط على زر موافق سيقوم البرنامج بالانتقال إلى التعليمات التالية وسيتم تنفيذها حسب قواعد لغة ***VB***.

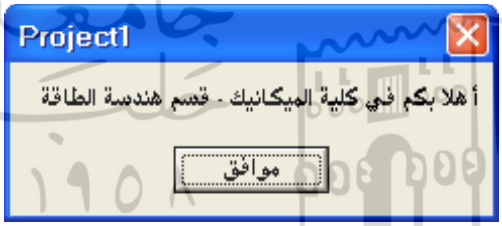
$MsgBox$ (" ")
$MsgBox$ ("$Prompt\ String$")
$MsgBox$ ("$Aleppo\ University, Dr\ Mohammad\ Hammad$")
$MsgBox$ ("أهلا بكم في كلية الميكانيك – قسم هندسة الطاقة")
MsgBox ("لغة الفيجوال بيزك – الدكتور محمد حماد")

وتظهر النتيجة كما في الشكل التالي:

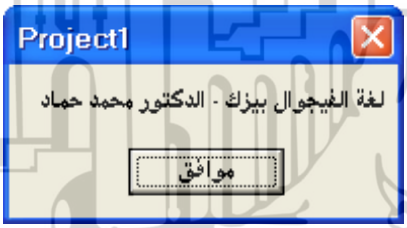
السطرين الكوديين الأوليين: عدم إظهار أي شي يظهر اسم المشروع ***Project***. أو طباعة قسم الرسالة ***Prompt String***.

السطر الثالث: إظهار ***Aleppo University, Dr Mohammad Hammad***.

السطر الرابع: إظهار أهلا بكم في كلية الميكانيك – قسم هندسة الطاقة.



السطر الخامس: إظهار لغة الفيجوال بيزك – الدكتور محمد حماد.



الصيغة الثانية:

إن الشكل العام لهذه الصيغة كما في الشكل التالي:

$$Var = MsgBox\ (Prompt[, Buttons][, Title][, Helpfile, Context])$$
$$Var = MsgBox\ ("Message", Integer\ Number\ , "Title")$$
$$\boldsymbol{Var = MsgBox\ ("Prompt"\ , Buttons\ , "Title")}$$

أما معنى هذه الوسائط فهو:

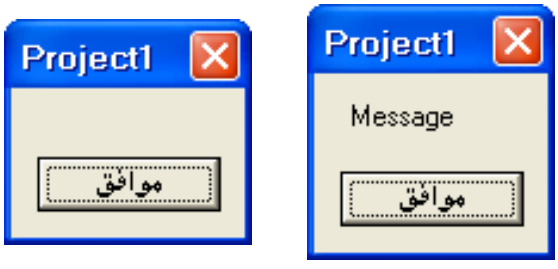
- ***Prompt***: هي عبارة عن تعبير نصي (رسالة) والذي يُظهِر مضمون الرسالة التي ستظهر في مربع الحوار. إن الطول الأعظمي يجب أن لا يزيد عن 1024 حرفاً والذي يعتمد على عرض الحروف المستخدمة. إذا كان طول الرسالة أكبر من طول خط ما نستطيع تجزئة الرسالة إلى أسطر باستخدام التابعين $\boldsymbol{Chr(10)}$ & $\boldsymbol{Chr(13)}$ واللذان يقومان بفتح سطر جديد والانتقال بالمؤشر إلى بداية السطر التالي.

- ***Buttons***: وهو تعبير عددي قيمته صحيحة والذي سيحدد عدد ونوع الأيقونات التي ستظهر على مربع الرسالة. عندها إهمال هذه القيمة سيعتبرها ***VB*** مساوية للصفر أي ***Buttons = 0*** وهنا سيتم عرض زر واحد فقط كما مر سابقاً وهو "موافق" أو "***Ok***".
- *Title* – عنوان مربع الرسالة الذي سيظهر في منطقة الاسم. عند إهمال هذه القيمة سيظهر اسم التطبيق في مكانها ***Project1***.
- ***Helpfile*** – اسم ملف المساعدة الذي سيظهر مع مربع الرسالة. نلاحظ أن استعمال هذا الوسيط يحتم علينا استعمال الوسيط الأخير ***Context***.
- ***Context*** – تعبير عددي يدل يشير إلى عبارة المساعدة التي يبينها الوسيط ***Helpfile*** وهو مرتبط بوجوده أيضاً.
- ***Var*** – هو متحول من النوع الصحيح ***Integer*** ويمثل رقم الزر الذي تم ضغطه.

ملاحظات:

- عند استخدام العبارة ***Message*** لا يتم إحاطة الوسائط بقوسين بعكس التابع الوظيفي ***MsgBox()*** الذي يتوجب فيه إغلاق الوسائط بقوسين.
- يجب استخدام القيمة المعادة من قبل التابع الوظيفي لإسنادها إلى متحول.
- تدل القيمة التي يعيدها التابع ***MsgBox()*** على الزر الذي تم النقر أو الضغط عليه.
- يؤدي عدم تحديد الوسيطين الثاني والثالث لعبارة ***MsgBox*** إلى ظهور مربع حوار يحمل زر موافق فقط والعبارة ***Message*** الموجودة في المكان المخصص *Prompt*.
- يؤدي عدم تحديد الوسائط الثلاث لعبارة ***MsgBox*** إلى ظهور مربع حوار يحمل زر موافق فقط ودون أي رمز.

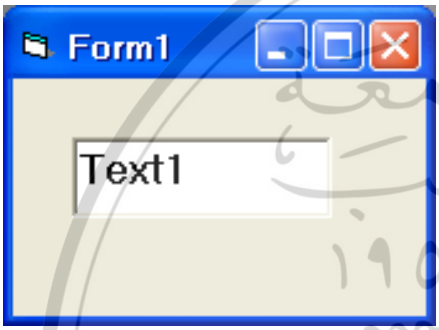
MsgBox (" Message")
MsgBox *(" ")*



أداة مربع النص TextBox Tool:

أداة تستخدم للإدخال والإخراج وتستخدم لكتابة وتعديل نص معين فتستخدم كمحرر النصوص ويمكن من خلالها إدخال المتحولات سواء كانت نصية أم رقمية ويجب الانتباه إلى أن هذه الأداة تتعامل مع المعطيات دوماً على أنها سلاسل حرفية ولذا إذا أردنا التعامل مع مضمونها الرقمي فلابد من أخذ القيمة العددية لمحتواها باستخدام التابع ***Val***.

يوضح الشكل التالي صندوق النص ***TextBox*** والذي يمتلك إمكانية إدخال قيم متغيرة إلى البرنامج أو إخراجها منه.



يتم إلحاق البند ***Text*** إلى يسار اسم صندوق النص ***Text*** للدلالة على المحتوى النصي للتعليمة، فإذا وقعت التعليمة ***Text1.Text*** مثلاً على يسار إشارة المساواة عندها ستكون هذه التعليمة تعليمة إخراج:

$$Text1.Text = "Ali"$$

هذا يعني أنّ السلسلة الرمزية ***Ali*** سوف تظهر في صندوق النص ***Text1*** عند تنفيذ البرنامج. أمّا إذا وقعت على يمين المساواة فهذا يعني أنها ستكون تعليمة إدخال:

$$Age = Val(Text1.Text)$$

هذا يعني أنّ القيمة المُدخلة في مربع النص سيتم إسنادها إلى المتحول المعرّف ***Age***.

تحتوي الخاصيّة ***Text*** على النص المكتوب داخل مربع النص. افتراضيًا يكون النص من سطر واحد، ويمكن جعل مربع النصوص عديد الأسطر باستخدام الخاصية ***Multiline*** بجعل قيمتها ***True***. وإذا أردنا ظهور أشرطة تمرير فإن الخاصية ***Scrollbar*** هي التي تحدد وجود هذه الأشرطة من عدمه وهل سيوجد شريطان أفقي ورأسي أم أحدهما فقط.

يمكن أن تستخدم الأداة ***Text Box*** بدل التعليمتين ***Input, Print*** المُستخدمتان في لغة البيزك العادية.

خصائص مربع النص Text Box:

الخاصية $Text$:

وتبين المحتوى الموجود ضمن مربع النص والذي يمكن أن يكون رقمياً أو حرفياً.

تستخدم في الإدخال والإخراج كما ذكرنا فإذا وقعت التعليمة $Text1.Text$ على يسار إشارة المساواة عندها ستكون هذه التعليمة تعليمة إخراج، وإذا وقعت على يمين إشارة المساواة فهذا يعني أنها ستكون تعليمة إدخال.

تعليمة إدخال والمضمون سلسلة حرفية	$A\$ = Text1.text$
تعليمة إدخال والمضمون قيمة عددية	$X = Val(Text1.text)$
أداة إخراج وستظهر المضمون الحرفي	$Text1.Text = A\$$
أداة إخراج وستظهر المضمون الرقمي	$Text1.Text = X$
أداة إخراج وستظهر المضمون الرقمي للعملية الحسابية	$Text1.Text = 8 - 4$
أداة إخراج وستمسح ما هو بداخل مربع النص	$Text1.Text = ""$

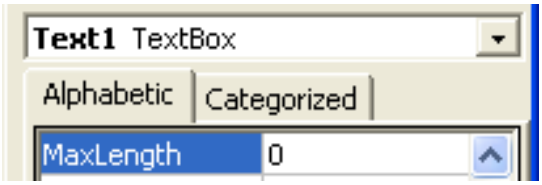
الخاصية $Alignment$:

تستخدم للتحكم بجهة ومكان ظهور النص ويمكن التحكم بهذه الخاصية في المرحلة المرئية كما يلي:

ويمكن التحكم بهذه الخاصية في المرحلة الكودية وتأخذ إحدى القيم التالية:

$Text1.Alignment = vbLeftJustify$ $Text1.Alignment = 0$
$Text1.Alignment = vbRightJustify$ $Text1.Alignment = 1$
$Text1.Alignment = vbCenter$ $Text1.Alignment = 2$

الخاصية *MaxLength*:



في كثير من الأحيان تحتاج إلى تحديد طول معين للقيم المطلوب إدخالها، أي يجب تحديد قيمة لعدد حروف السلسلة الحرفية المُشكّلة لمحتوى النص ويمكن أن تأخذ القيمة 0 أو قيمة رقمية معينة أخرى.

عندما تكون القيمة مساوية للصفر فهذا يعني أن الأداة بدون حد أقصى من الحروف ويمكن أن نكتب العدد المطلوب. وإذا أردنا أن لا يتجاوز عدد الأحرف المُدخلة العشرة حروف عندها يجب أن نكتب:

$$Text1.MaxLength = 10$$

في هذه الحالة سيسمح مربع النص بإدخال عشرة حروف وسوف يتم تجاهل أي أحرف يتم إدخالها بعد ذلك.

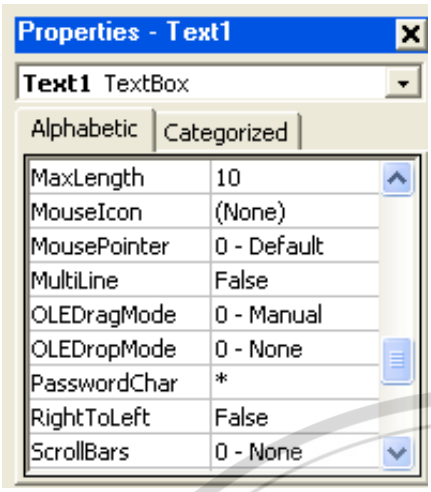
إن كل خصائص مربع النص تستخدم في المرحلة المرئية والكودية إلا هذه الخاصية أي ***MaxLength*** فهي ممكنة في المرحلة المرئية فقط.

الخاصية *PasswordChar*:

في كثير من الأحيان قد نضطر للتعامل مع سرية معينة للبرامج لذا لن نسمح بالدخول للبرنامج إلا باستخدام كلمة سر. ولمنع الآخرين من رؤية أو معرفة كلمة السر المدخلة كان من المفيد التفكير بطريقة تشفير معيّنة لكلمة السر المدخلة.

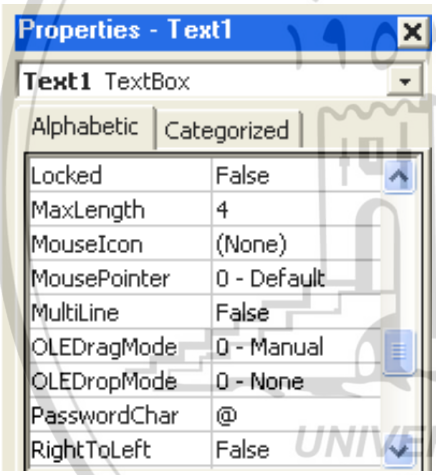
تستخدم هذه الخاصية لمنع ظهور كلمة السر على الشاشة واستبدالها بأحرف أو رموز أو أرقام أخرى.

يمكن مثلاً جعل عدد المحارف المسموح استخدامها في الأداة ***Text1*** مساوية إلى عشرة. ويمكن تغيير شكل ظهورها إلى * والتي تُكتب مرة واحدة في الخاصية ***PasswordChar*** كما يلي:



السماح في المرحلة التنفيذية باستخدام عشرة محارف وسنشفرها بالرمز *

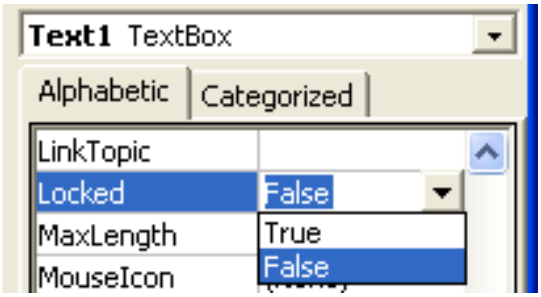
ويمكن مثلاً جعل عدد المحارف المسموح استخدامها في الأداة ***Text1*** مساوية إلى أربعة. ويمكن تغيير شكل ظهورها إلى @ والتي تُكتب مرة واحدة في الخاصية ***PasswordChar*** كما يلي:



السماح في المرحلة التنفيذية باستخدام أربعة محارف وسنشفرها بالرمز @

يجب التنويه إلى أن الخاصية ***Text*** لمربع النص ستتعامل مع المضمون أو المحتوى الفعلي الذي تم إدخاله وليس مع المضمون أو المحتوى الذي ظهر بعد عملية التشفير.

الخاصية Locked:



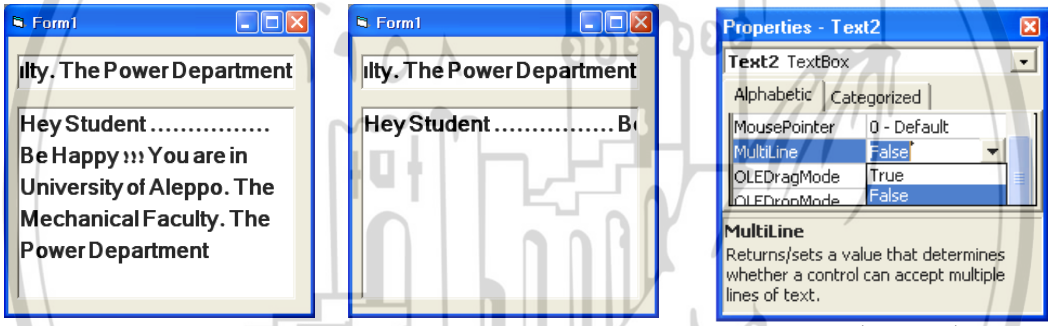
تستخدم الخاصية ***Locked*** لمنع المستخدم من تحرير النص (أي التعديل على النص أو مسح ما في داخله).

- يجب ملاحظة أنه مازال مسموح لنا تعليم جزء أو كل المحتوى الموجود في مربع النص ونسخه إلى الحافظة.

الخاصية *Multiline*:

الخاصية التي تسمح بعرض النص على سطر واحد أو على عدة أسطر. بالحالة الافتراضية تكون قيمة الخاصية $Multiline = False$ وهذا يعني أن العبارات المكتوبة ضمن مربع النص ستظهر ضمن سطر واحد مهما كانت أبعاد صندوق (مربع) النص.

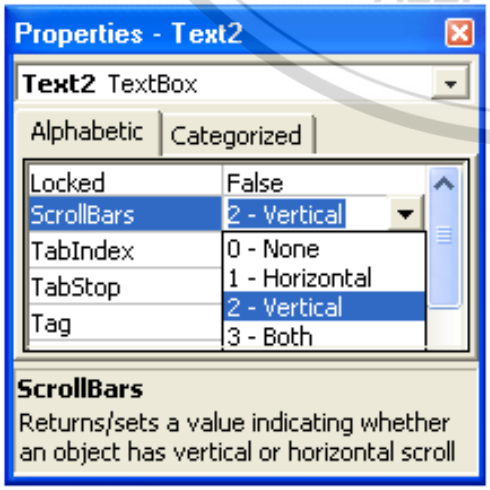
أما إذا كانت قيمة الخاصية $Multiline = True$ فهذا يعني أن أداة مربع النص تسمح بعرض النص على أكثر من سطر واحد وهذا سيكون متعلق بوجود أو عدم وجود أشرطة تمرير.



MultiLine = True　　MultiLine = False　　MultiLine اختيار قيمة الخاصية

الخاصية ScrollBars:

وهي الخاصية التي تسمح بتحريك نافذة الرؤية من أجل رؤية أجزاء النص غير الظاهرة وذلك بسبب أبعاد صندوق النص. تأخذ هذه الخاصية أربع قيم وهي:



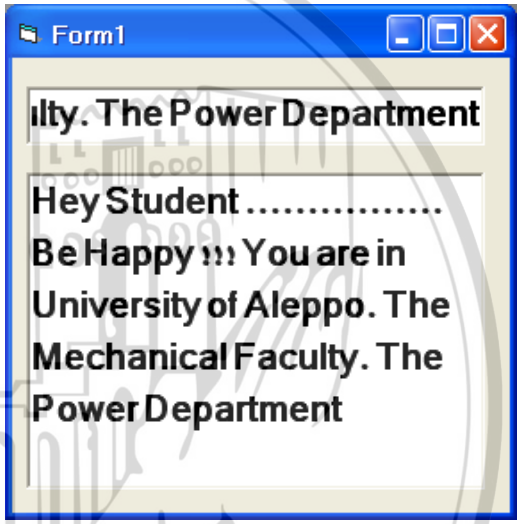
- *0-None* لا تسمح بإظهار أشرطة التمرير.
- *1-Horizontal* يظهر شريط التمرير الأفقي.
- *2-Vertical* يظهر شريط التمرير العمودي.
- *3-Both* يظهر كل من شريطي التمرير الأفقي والعمودي.

١) **حالة عدم وجود أشرطة تمرير أي ScrollBars = 0 – None:**

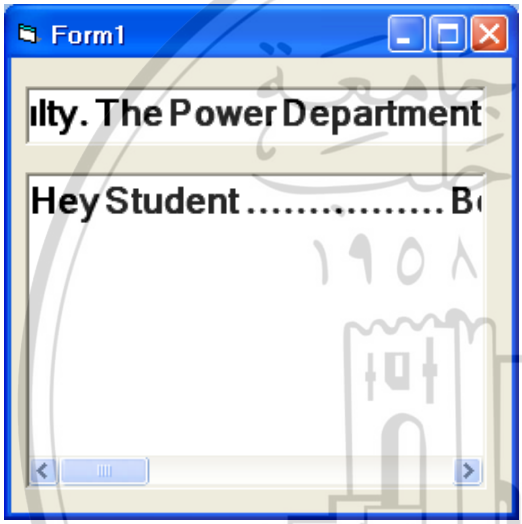
عند عدم وجود أشرطة التمرير، سيظهر النص على عدة أسطر ويمكن التنقل بداخله ورؤيته كاملاً من خلال أسهم الحركة في لوحة المفاتيح.

٢) **حالة وجود شريط تمرير أفقي أي ScrollBars = 1 – Horizontal:**

وجود شريط تمرير أفقي. هنا سيظهر النص على سطر واحد ويمكن التنقل بداخله ورؤيته كاملاً من خلال أسهم الحركة في لوحة المفاتيح بالإضافة إلى تحريك شريط التمرير إلى اليمين واليسار.



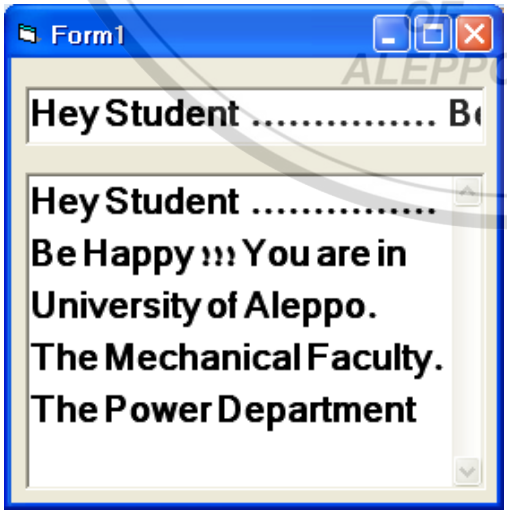
ScrollBars = 0 - None



ScrollBars = 1 - Horizontal

٣) **حالة وجود شريط تمرير عمودي أي ScrollBars = 2 – Vertical:**

عند وجود شريط تمرير عمودي، سيظهر النص على عدة أسطر ويمكن التنقل بداخله ورؤيته كاملاً من خلال أسهم الحركة في لوحة المفاتيح بالإضافة إلى تحريك شريط التمرير إلى الأعلى والأسفل. عندما يتسع مربع النص للنصوص المكتوبة بداخله يظهر لونه باهتاً (أي لا نكون بحاجة إليه).



ScrollBars = 2 – Vertical

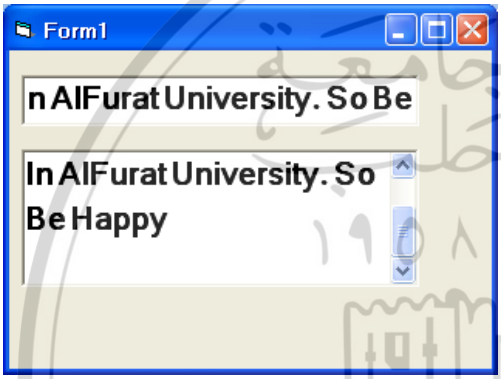
وجود شريط تمرير عمودي – التنقل داخل مربع النص:

عندما لا يتسع مربع النص للنصوص المكتوبة بداخله يظهر لونه أسوداً غامقاً.

لرؤية الجزء السفلي من النص المكتوب نحرك السحاب (شريط التمرير) إلى الأسفل.

$$ScrollBars = 2 - Vertical$$

وجود شريط تمرير عمودي – التنقل داخل مربع النص:



عندما لا يتسع مربع النص للنصوص المكتوبة بداخله يظهر لونه أسوداً غامقاً.

لرؤية الجزء العلوي من النص المكتوب نحرك السحاب (شريط التمرير) إلى الأعلى.

$$ScrollBars = 2 - Vertical$$

٤) حالة وجود شريطي التمرير الأفقي والعمودي عمودي أي $ScrollBars = 3 - Both$:

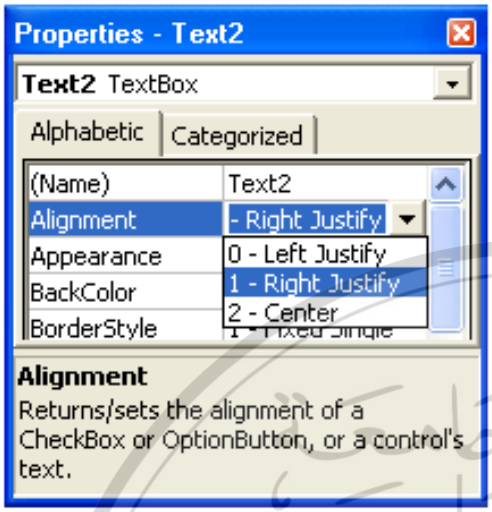
سيظهر النص على عدة أسطر ويمكن التنقل بداخله ورؤيته كاملاً من خلال أسهم الحركة في لوحة المفاتيح بالإضافة إلى تحريك شريط التمرير الأفقي إلى اليمين وإلى اليسار وشريط التمرير العمودي إلى الأعلى والأسفل.

تظهر النصوص على سطر واحد حتى الضغط على مفتاح الإدخال Enter والذي يجعل النص يبدأ من سطرٍ جديدٍ.

عندما يتسع مربع النص للنصوص المكتوبة بداخله تظهر ألوان أشرطة التمرير باهتة، أي لا نكون بحاجة إليها.

إن الذي يتحكم بظهور شريط التمرير الأفقي خاصية المحاذاة *Alignment*.

الخاصية Alignment:



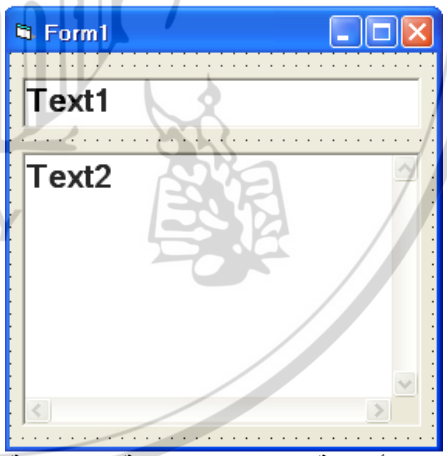
وتأخذ هذه الخاصية ثلاث قيم وهي:

0 - Left Justify ظهور شريطا التمرير بشكل عادي.

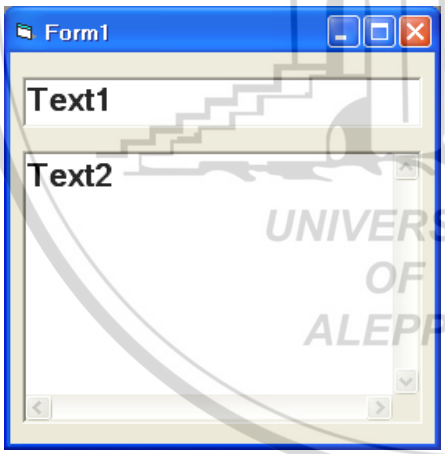
1- Right Justify سيظهر شريط التمرير العمودي ويبقى مكان شريط التمرير الأفقي فارغاً. يكون البعد العمودي لمربع النص مساوٍ للبعد العمودي لشريط التمرير العمودي.

2-Center سيظهر شريط التمرير العمودي ويبقى مكان شريط التمرير الأفقي فارغاً. يكون البعد العمودي لمربع النص مساوٍ للبعد العمودي لشريط التمرير العمودي مضافاً إليه ارتفاع شريط التمرير الأفقي.

١) الحالة الأولى *Left Justify* – 0 = *Alignment*:



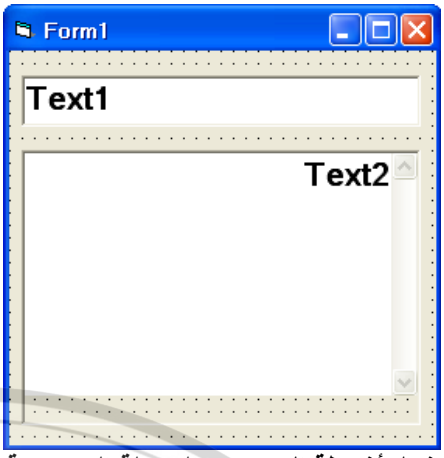
شكل أشرطة التمرير – المرحلة التصميمية



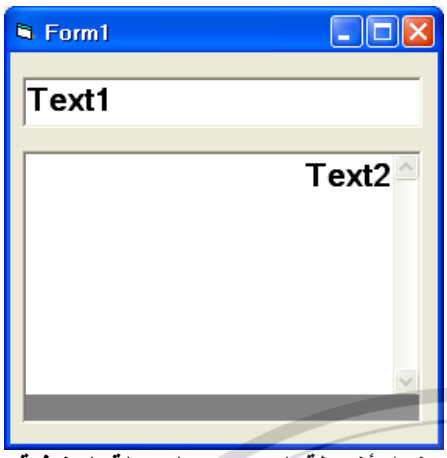
شكل أشرطة التمرير – المرحلة التنفيذية

٢) الحالة الثانية *Right Justify* – 1 = *Alignment*:

سيظهر شريط التمرير العمودي ويبقى مكان شريط التمرير الأفقي فارغاً. يحل جزء من الإطار مكان الفراغ الذي أحدثه زوال شريط التمرير الأفقي أي يكون البعد العمودي لمربع النص مساوٍ للبعد العمودي لشريط التمرير العمودي.



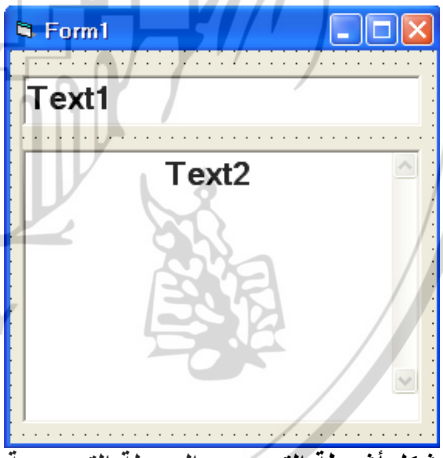
شكل أشرطة التمرير – المرحلة التصميمية



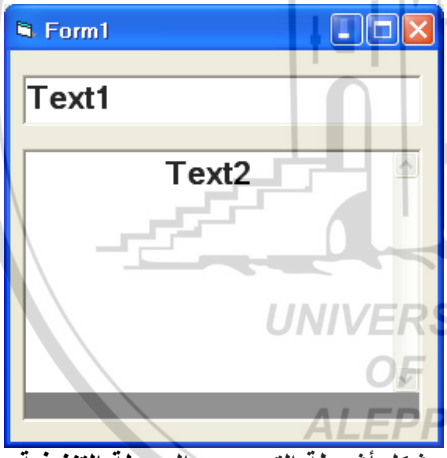
شكل أشرطة التمرير – المرحلة التنفيذية

٣) الحالة الثالثة $Alignment = 2 - Center$:

سيظهر شريط التمرير العمودي ويبقى مكان شريط التمرير الأفقي فارغاً. يحل جزء من الإطار مكان الفراغ الذي أحدثه زوال شريط التمرير الأفقي أي يكون البعد العمودي لمربع النص مساوٍ للبعد العمودي لشريط التمرير العمودي مضافاً إليه ارتفاع شريط التمرير الأفقي.



شكل أشرطة التمرير – المرحلة التصميمية



شكل أشرطة التمرير – المرحلة التنفيذية

الخاصية ToolTip:

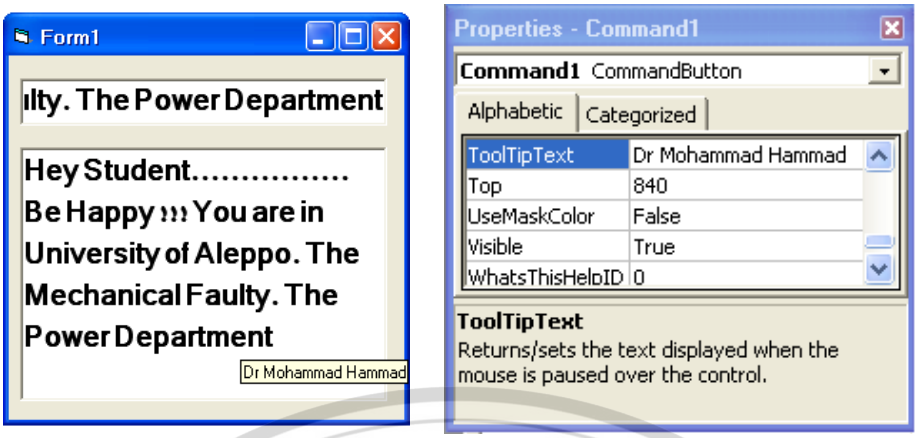
تُعيد أو تضبُط التلميح المرتبط مع الأداة.

يتم ضبط هذه الخاصية في المرحلة التصميمية في مربع الخصائص الخاص بالأداة المطلوب تطبيقه عليها.

ويمكن أن تستخدم في المرحلة التنفيذية وتكون صيغتها العامة كما يلي:

$ControlName.ToolTipText\ [= String]$

$Command1.ToolTipText = "Dr\ Mohammad\ Hammad"$



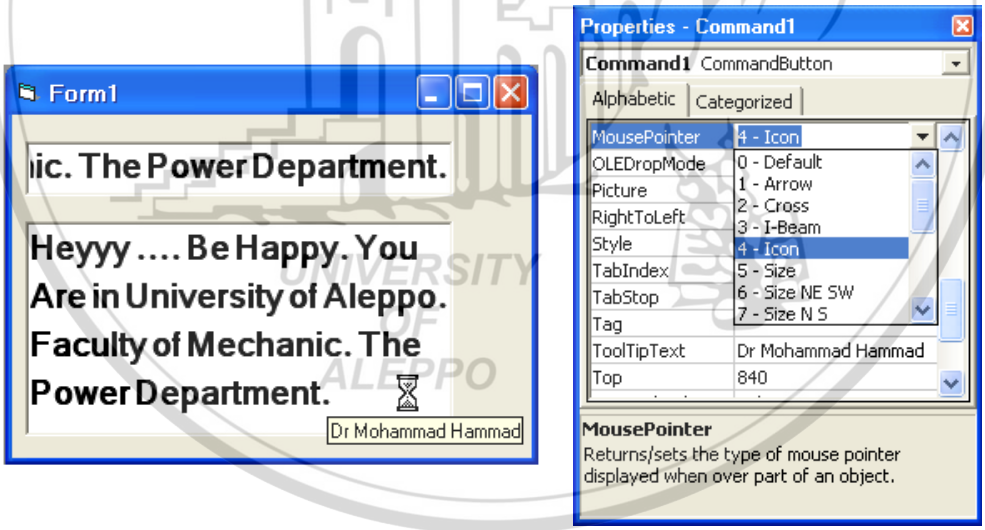
الخاصية MousePointer:

تُعيد أو تضع قيمة معيّنة يحدد من خلالها نوع المؤشر الذي سيظهر عندما تكون الماوس فوق الكائن الذي تم تغيير هذه الخاصية له.

يتم ضبط هذه الخاصية في المرحلة المرئية في مربع الخصائص الخاص بالأداة المطلوب تطبيقه عليها. ويمكن أن تستخدم في المرحلة التنفيذية وتكون صيغتها العامة كما يلي:

$$ControlName.MousePointer\ [=\ Value]$$

$$Command1.MousePointer\ =\ 11$$



يمكن تطبيق هذه الخاصية على العديد من الأدوات. وتستخدم عند الرغبة في تغيير عمل مؤشر الماوس أثناء المرور فوق أداة من الأدوات الموجودة على النموذج أو صندوق حوار. فمثلاً إن ضبط القيمة على الرقم **11-*vbHourglass*** والتي تُظهر الساعة الرملية يستخدم عند الرغبة في الإشارة إلى أنّه على المستخدم أن ينتظر قليلاً حتى تنتهي العملية أو الأمر المطلوب.

تأخذ الخاصية ***MousePointer*** قيماً مختلفة تؤدي إلى ظهور أشكال مختلفة للماوس كما هو مبيّن:

vbValue	***Value***	***Description***
vbDefault	0	الشكل الافتراضي للماوس
vbArrow	1	سهم
vbCrosshair	2	صليب
vbIbeam	3	شعاع I
vbIconPointer	4	أيقونة (مربع صغير داخل مربع)
vbSizePointer	5	حجم (أربع أسهم بمؤشرات تشير إلى الشمال، الجنوب، الغرب، والشرق)
vbSizeNESW	6	حجم NESW (سهم مضاعف يشير إلى الشمال الغربي والجنوب الشرقي)
vbSizeNS	7	حجم NS (سهم مضاعف يشير إلى الشمال الغربي والجنوب الشرقي)
vbSizeNWSE	8	حجم NWSE (سهم مضاعف يشير إلى الشمال الجنوب)
vbSizeWE	9	حجم WE (سهم مضاعف يشير إلى الغرب والشرق)
vbUpArrow	10	سهم للأعلى
vbHourglass	11	ساعة رملية (انتظار)
vbNoDrop	12	لا قطرة
vbArrowHourglass	13	سهم وساعة رملية
vbArrowQuestion	14	سهم وعلامة استفهام
vbSizeAll	15	حجم كلّي
vbCustom	99	أيقونة مخصصة يمكن تحميلها من خلال زر الاستعراض

الخاصية Font:

توجد الخاصية *Font* في كل العناصر التي تتعامل مع النصوص وهي الخاصية المسؤولة عن الخط المستخدم في عرض النصوص على الشاشة أو على الطابعة. وعلى الرغم أن الخط خاصية لعناصر مختلفة إلا أنه في حد ذاته كائن مستقل له خصائص تتحكم في كيفية عرض النصوص وهي:

- **الخاصية *FontName*:**

وتحدد اسم الخط المستخدم في عرض النص. ويجب تحديد نوع الخط قبل البدء بالتكلم عن الخصائص الأخرى للخط ***Font*** مثل الحجم ***Size*** والتغميق ***Bold***. وهناك مجموعة من الخطوط الأجنبية والعربية مثل ... ***Times New Romans, Andalus***.

ويكتب الأمر الكودي كما يلي:

$Text1.FontName = "Simplified\ Arabic"$
$Text1.FontName = "Arial"$

- **الخاصية *FontBold*:**

وتحدد ما إذا كان الخط المراد إظهاره سميك أم عادي.

- **الخاصية *FontItalic*:**

وتحدد هل الأحرف مائلة أم لا .

- **الخاصية *FontUnderline*:**

وتحدد هل سيتم إظهار الأحرف مسطرة بسطر رفيع أسفلها .

- **الخاصية *FontSize*:**

وتحدد حجم الخط المراد إظهاره.

- **الخاصية *FontStrikethru*:**

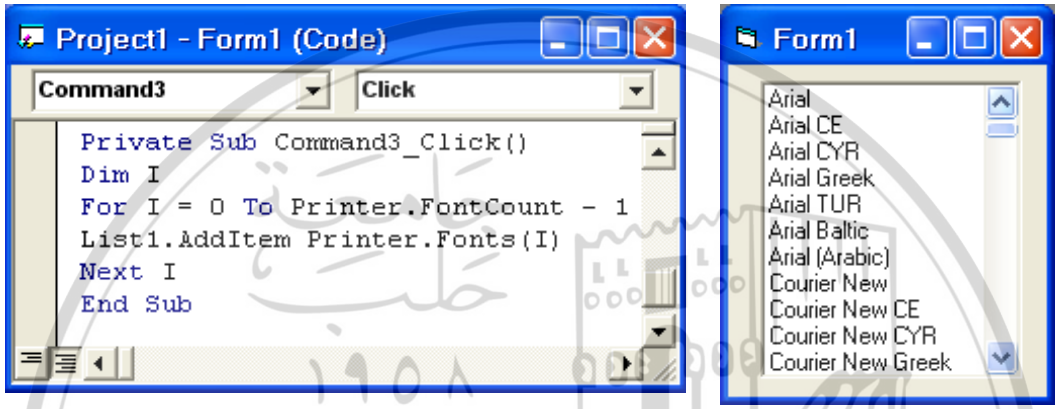
وتحدد فيما إذا كان سيظهر خط في منتصف النص المطلوب إظهاره.

ويكتب الأمر الكودي لهذه الخصائص كما يلي:

$Text1.FontItalic = Not\ Text1.FontItalic$
$Text1.FontBold = Not\ Text1.FontBold$
$Text1.FontSize = 24$

- **الخاصية *Fonts*:**

تحدد جميع أسماء الخطوط المعرفة والمتوفرة على الجهاز الحالي أو الطابعة الفعالة:



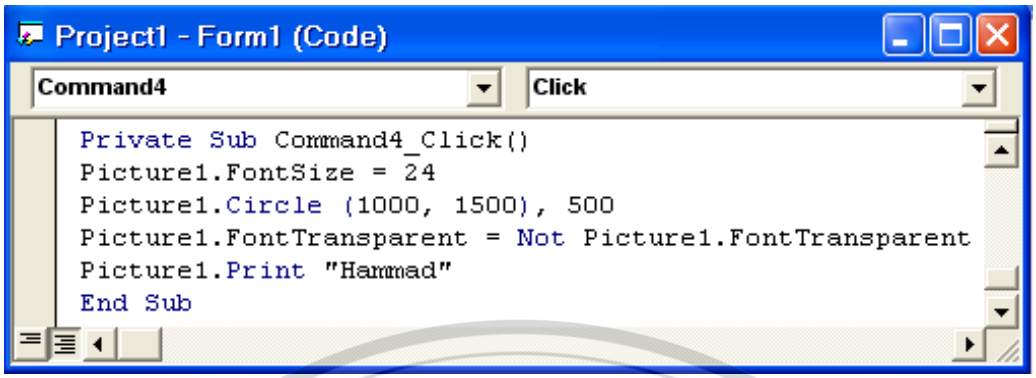
- **الخاصية *FontTransparent*:**

وتبين تأثير خلفية النص أو الرسم على النموذج أو أداة ***PictureBox*** أو أي أداة أخرى. ويكتب الأمر الكودي بإحدى الشكلين:

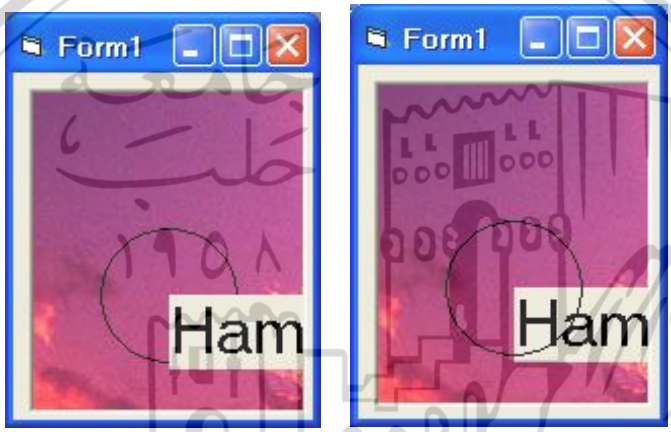
$$Picture1.FontTransparent = Not\ Picture1.FontTransparent$$
$$Picture1.FontTransparent = False$$
$$Picture1.FontTransparent = True$$

إن تغيير الخاصية ***FontTransparent*** في المرحلة التنفيذية لن يؤثر على الرسوميات والنصوص التي رسمت وكتبت للتو على النموذج أو الأداة ***PictureBox*** أو الطابعة.

هنا تم استخدام العبارة ***Not*** عدد من المرات.



وكانت نتيجة استخدام البرمجة الكودية كما في الشكل:



الحدث Change:

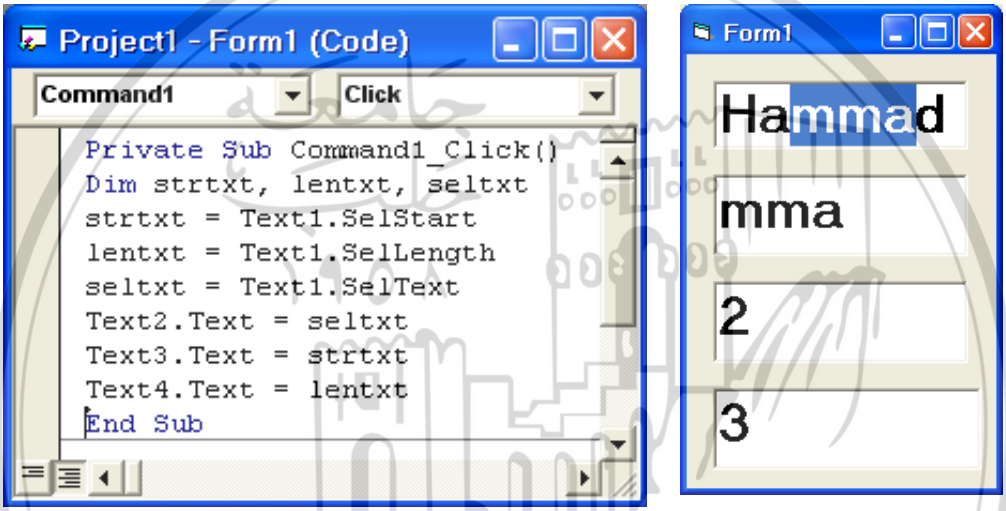
يتأثر مربع النص بأنواع كثيرة من الأحداث إلا أن الحدث الأكثر أهمية هو حدث التغيير. ويتم تنفيذ الإجراء الكودي المتعلق بهذا الحدث كلما حدث تغيير على محتوى خانة النص سواءً كان ذلك من خلال لوحة المفاتيح أو من خلال تغيير الخاصية *Text* للأداة.

تعديل محتويات مربع النص Editing the Text Box:

قد نضطر لتحديد جزء من النص المعروض في مربع النص بغية نسخه أو إجراء أي تعديل عليه لذا لابد أن نعرف من أين سيتم التحديد وعدد الرموز أو المحارف المحدد وما هو النص الذي تم تحديده وذلك بغية حذفه أو تعديله لذا نتعامل مع الأوامر الكودية التالية:

- ***Selstart*** لمعرفة بداية الجزء الذي تم تحديده من قبل المستخدم نستخدم الخاصية والتي تعطى رقم يعبر عن ترتيب الحرف الذي بدأ عنده الاختيار وهو الحرف غير المُعلّم الذي سبق الرموز المُعَلَّمة.

- لمعرفة طول الجزء الذي تم تحديده من قبل المستخدم نستخدم الخاصية ***SelLength*** والتي تعطى رقم يعبر عن عدد الحروف التي تم تحديدها أي تعليمها في مربع النص.
- لمعرفة الجزء النصّي الذي تم تحديده من قبل المستخدم نستخدم الخاصية ***SelText*** والتي تعطى المحتوى الحرفي الذي تم تحديده.
- بعد عملية تعليم (تحديد) نص معين نستطيع حذفه أو استبداله بأي مضمون نصي آخر.



وهناك عدد كبير من الخصائص الأخرى لمربع النص ولا مجال لذكرها جميعاً الآن.

إن من أهم العيوب التي ترافق هذه الأداة هو:

- استخدامه لسطر واحد ولكن حتى ولو قمنا بتغيير خاصية ***MultiLine*** وإعطائها القيمة *True* والتي تمكننا من استخدام أكثر من سطر إلا أن ذلك يبقى نوعاً ما محدوداً.
- أن أي تغيير يتم تطبيقه على أي مقطع من النص سيتم تطبيقه على كل ما تحتويه الأداة من كلمات ولذا ولضرورة التعامل بشكل منفصل مع الكلمات المكتوبة كان لا بد من التفكير في أداة أخرى وهي أداة النص غني التنسيق أو *Rich Text Box*.

أداة مربع اللائحة *ListBox Tool* :

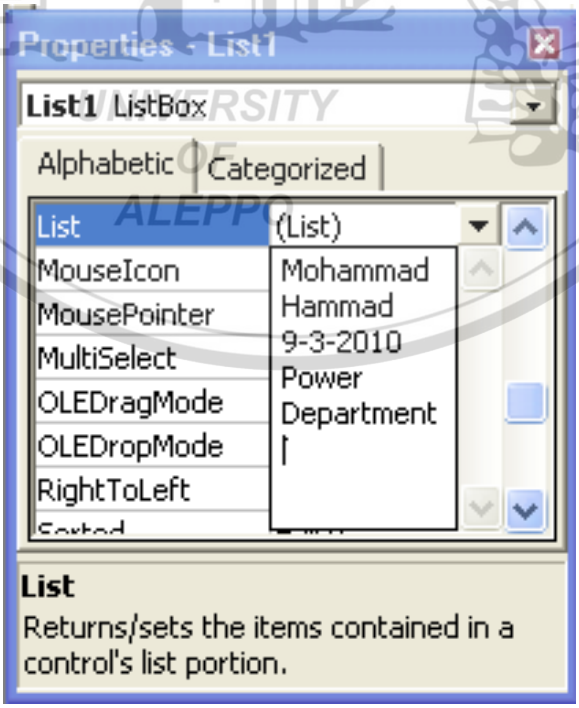
وهي أداة إخراج فقط، بعكس أداة صندوق النص ***TextBox*** تستخدم للإدخال والإخراج. تستطيع الأداة عرض لائحة من البنود أو العناصر ويمكن للمستخدم أن ينتقي واحداً منها أو أكثر من بنداً واحداً.

تتمتع الأداة ***ListBox*** بمجموعة من الخصائص المشابهة لأغلب أدوات ***VB*** إلاّ أنها تتمتع ببعض الخصائص التي تختلف عن الأدوات الأخرى نذكر منها:

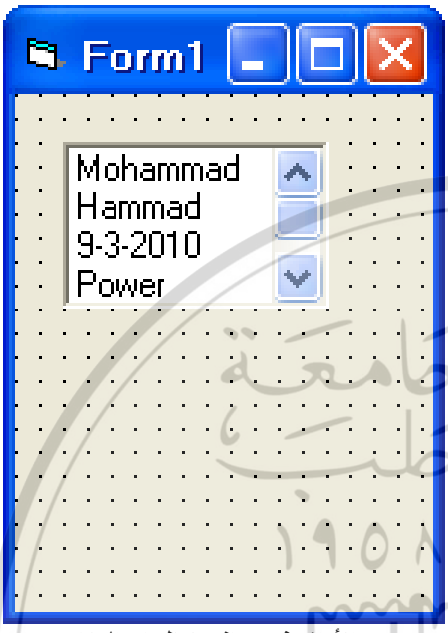
الخاصية List:

وهي الخاصية التي تسمح بإدخال العناصر – المطلوب إظهارها في الأداة – في المرحلة التصميمية.

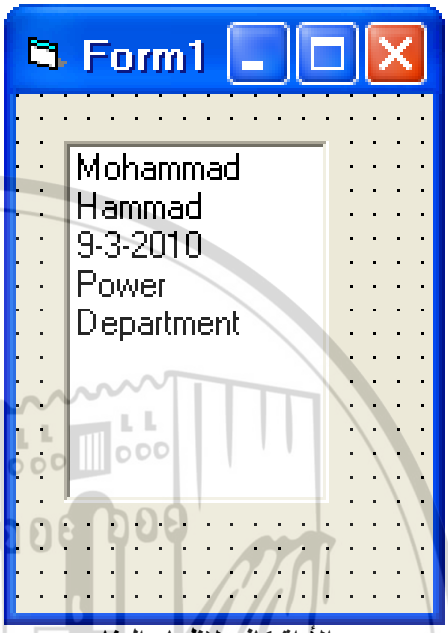
- يتم إدخال العناصر في المرحلة التصميمية من خلال النقر على الخيار ***List*** الموجود بجانب الخاصية ***List***.
- يتم كتابة العنصر الأول وهو العنصر الذي دليله *(0)*.
- إذا أردنا إدخال عنصر آخر نستخدم التركيب ***[Ctrl + Enter]***عندها ينتقل المؤشر إلى السطر التالي. أما لإنهاء عملية إدخال العناصر في المرحلة المرئية نستخدم المفتاح ***Enter***.



- تظهر كل العناصر أو البنود داخل الأداة إلا إذا كان عدد العناصر المطلوب عرضها أكبر من حجم الأداة، وفي هذه الحالة سيظهر (سيتم إضافة) شريط تمرير داخل الأداة بشكل آلي والذي من خلاله نستطيع التنقل بين العناصر.

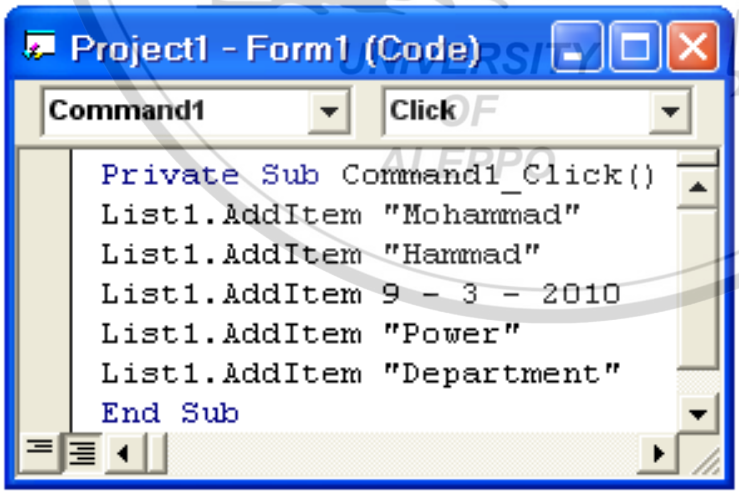


حجم الأداة غير كافٍ لإظهار العناصر



حجم الأداة كافٍ لإظهار العناصر

يتم إدخال العناصر – المطلوب إظهارها في الأداة – في المرحلة الكودية ضمن كود معين أو في المرحلة التنفيذية باستخدام توابع وأدوات الإدخال الأخرى. وتظهر في المرحلة التنفيذية كما يلي:



لقد قام البرنامج بالسطر الثالث بإجراء عملية حسابية ولم يظهر التاريخ بشكل صحيح والسبب كوننا لم نستخدم إشارتي التنصيص للدلالة على أن القيمة نصيّة.

لهذه الأداة مجموعة من الخصائص الكودية (لا تظهر بالمرحلة التصميمية) وتفيد في تحديد ترتيب وعدد العناصر الموجودة في الأداة منها:

✓ $\boldsymbol{ListIndex}$ – تحدد الفهرس للعنصر الذي تمّ انتقاؤه بصورة عامة في الأداة وهي خاصية غير متوفرة في المرحلة التصميمية.

$$Text1.Text = List1.ListIndex$$

- تكون قيمة $\boldsymbol{ListIndex = -1}$ إذا لم يتم انتقاء أي عنصر من اللائحة.
- تكون قيمة $\boldsymbol{ListIndex = 0}$ إذا لم يتم انتقاء العنصر الأول من اللائحة.
- تكون قيمة $\boldsymbol{ListIndex = 1}$ إذا لم يتم انتقاء العنصر الثاني من اللائحة.
- تكون قيمة $\boldsymbol{ListIndex = n - 1}$ إذا لم يتم انتقاء العنصر رقم n من اللائحة.

✓ $\boldsymbol{ListCount}$ – عدد العناصر الذي يظهر في الأداة.

$$Text1.Text = List1.ListCount$$

- إن قيمة الخاصية $\boldsymbol{ListCount}$ هي دائماً زيادة واحد عن القيمة الأكبر لـ $\boldsymbol{ListIndex}$.
- أي تكون قيمة $\boldsymbol{ListCount = n + 1}$ إذا كان عدد العناصر مساوٍ لـ n عنصر في اللائحة.

الخاصية $\boldsymbol{Style}$:

وهي الخاصية التي تسمح بتحديد نمط إظهار وسلوك أداة اللائحة بشكلها العادي الافتراضي أو التي تسمح بإظهار أداة التحكم $\boldsymbol{CheckBox}$ داخل أداة اللائحة.

تأخذ هذه الخاصية إحدى القيمتين التاليتين:

١) الحالة الأولى $\boldsymbol{Style = 0 - vbListBoxStandard}$:

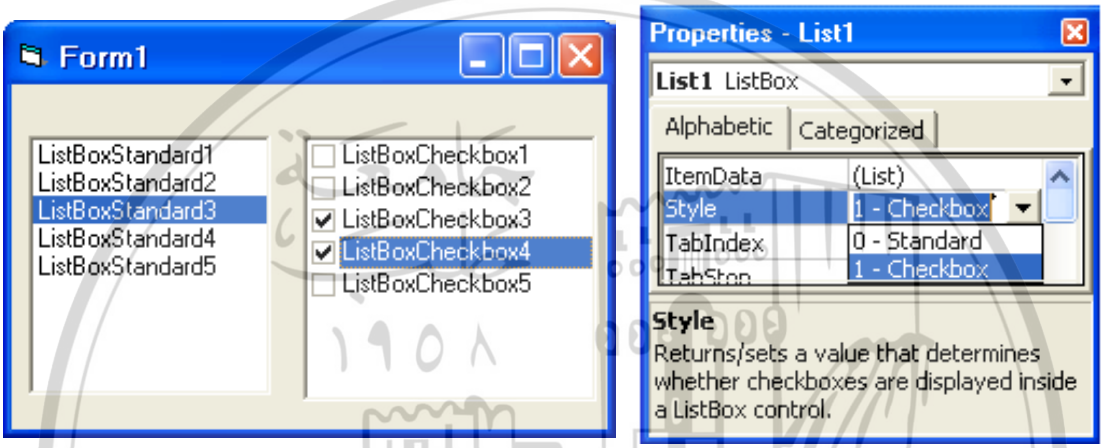
وهي الخاصية الافتراضية أو القياسية. تظهر الأداة $\boldsymbol{ListBox}$ كلائحة من الفقرات النصيّة.

يمكن انتقاء أو تحديد عنصر واحد أو فقرة واحدة من اللائحة، وعند محاولة انتقاء أو تحديد عنصر آخر سيتم إلغاء تحديد العنصر السابق.

٢) الحالة الثانية ***vbListBoxCheckBox*** **– 1 =** ***Style***:

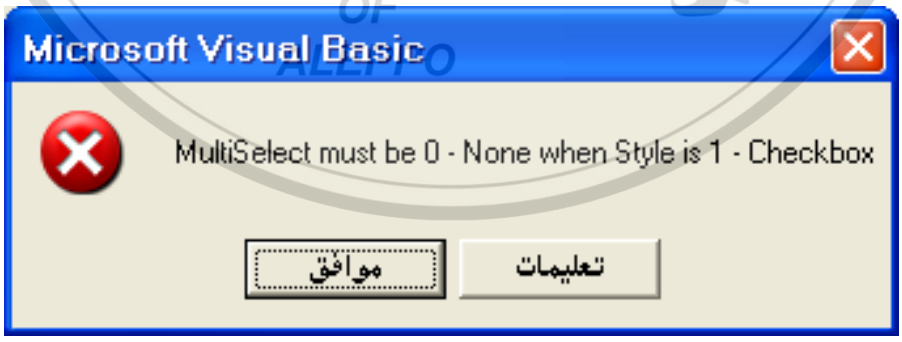
تظهر الأداة ***ListBox*** وبجانبها صندوق اختيار ***CheckBox*** مرتبط بكل بند أو عنصر من العناصر.

يمكن انتقاء أو تحديد عنصر واحد أو فقرة واحدة من اللائحة، وعند محاولة انتقاء أو تحديد عنصر آخر سيتم إلغاء تحديد العنصر السابق. لكن الأداة ***CheckBox*** تسمح بانتقاء أكثر من عنصر في الأداة ***ListBox*** من خلال النقر عليها وتحديدها.



الخاصية *MultiSelect*:

تحدد فيما إذا كنا نرغب بتحديد أكثر من عنصر واحد من عناصر اللائحة. هذه الخاصية تعمل عندما ***Style*** **= 0 –** ***Standard*** ولا تعمل عندما ***Style*** **=** ***CheckBox*** **– 1** وإذا أردنا تغيير الخاصية بشكل غير صحيح يظهر مربع الحوار التالي:



تأخذ هذه الخاصية إحدى القيم الثلاث التالية:

١) قيمة الخاصية ***None*** **– 0 =** ***MultiSelect***:

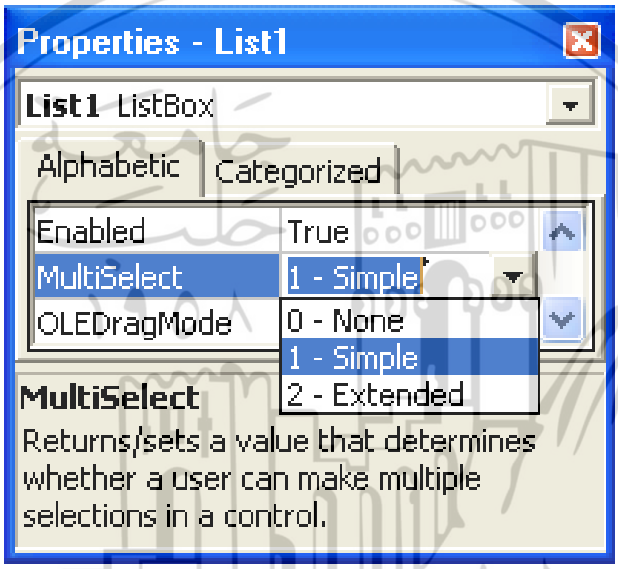
لا تسمح بالاختيار المتعدد.

٢) قيمة الخاصية: $MultiSelect = 1 - Simple$

تسمح بالاختيار المتعدد. ويمكن تنفيذ الاختيار أو إلغاء الاختيار للعناصر بواسطة:

- زر الماوس.

يستخدم زر الماوس الأيسر للقيام بعملية تحديد أو إلغاء تحديد العناصر أو البنود. عند النقر بزر الماوس الأيسر على عنصر ما سيتم تحديده، وعند النقر عليه مرة ثانية سيتم إلغاء التحديد.



- من خلال مفتاح المسطرة $Space\ Bar$:

عند الضغط على المفتاح $Space$ والتركيز على عنصر ما عندها سيتم تحديد هذا العنصر. وعند الضغط على المفتاح $Space$ والتركيز على العنصر الذي قد تم تحديده، عندها سيتم إلغاء تحديد هذا العنصر.

٣) قيمة الخاصية $MultiSelect = 2 - Extended$:

تسمح بالاختيار المتعدد. ويمكن اختيار أو إلغاء اختيار العناصر بواسطة:

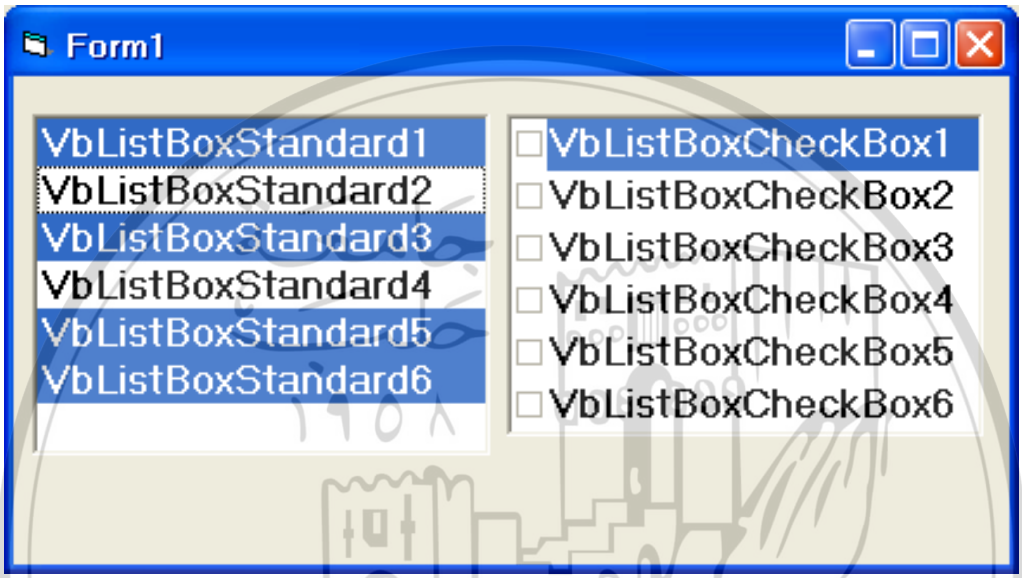
- زر الماوس والمفتاح $Space\ Bar$ كما ذكرنا سابقاً.
- من خلال المفاتيح $Shift, Ctrl$ من لوحة المفاتيح:

المفتاح $Shift$:

- يستخدم المفتاح $Shift$ مع زر الماوس الأيسر لتحديد العناصر المتجاورة.
- يستخدم المفتاح $Shift$ مع مفاتيح الأسهم لتحديد العناصر المتجاورة.

المفتاح ***Ctrl***:

- يستخدم المفتاح ***Ctrl*** مع زر الماوس الأيسر لتحديد العناصر غير المتجاورة.
- يستخدم المفتاح ***Ctrl*** لإلغاء تحديد العناصر المحددة بأي طريقة من الطرق السابقة.



الخاصية Sorted:

لا تعمل في المرحلة التنفيذية لذا يمكن تغيير قيمتها في المرحلة التصميمية فقط.

عملية الفرز نصية ، أي يتعامل مع الأرقام كحروف ولذا فإن الأرقام الثلاثة التالية (1, 2, 11) ستظهر كما يلي (1, 11, 2) أي الرقم 1 أولاً ثم يظهر بعده الرقم 11 ثم الرقم 2.

وتظهر العناصر المدخلة بأي ترتيب تم إدخالها كما يلي:

$Ham1, Ham10, Ham18, Ham3, Ham31, Ham4, Ham6$

الخاصية *Columns*:

قيمة صحيحة تسمح بعرض العناصر عموديّاً أم لا.

القيمة الافتراضية هي $\boldsymbol{Columns = 0}$ أي عرض العناصر عموديّاً.

عندما تكون قيمة $\boldsymbol{Columns > 0}$ عندها ستعني هذه القيمة عدد الأعمدة الأفقية المستخدمة لعرض العناصر.

تعليمة الطباعة Print:

تستخدم هذه التعليمة للطباعة على النموذج أو بعض الكائنات الأخرى الموجودة عليه مع إمكانية التحكم بشكل وبطريقة الطباعة. يمكن باستخدام ***Print*** طباعة القيمة العددية لمتحول ما أو مجموعة من المحارف والتي تشكل متحول نصّي أو كليهما معاً (عددي ونصّي). إضافةً إلى إمكانية إظهار قيمة أكثر من متحول وناتج عملية حسابية ما معاً.

والشكل العام للتعليمة هو:

Print [***ExpressionList***] [{ ; | , }]

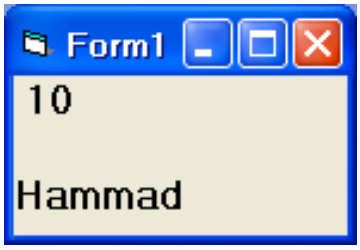
حيث:

- ***ExpressionList*** – هي مجموعة من الثوابت أو المتحولات أو التعابير الحسابية. ويمكن للثوابت أن تكون عددية أو حرفية.
- [{; | ,}] – تعني أنّه يمكن استخدام الفاصلة المنقوطة أو العادية.
- الفاصلة المنقوطة (;) بين قيمتين تؤدي إلى طباعة القيمة الثانية بجانب الأولى تماماً.
- الفاصلة العادية (,) تؤدي إلى طباعة القيمة الثانية بدءاً من المنطقة الطباعية التالية أي على مسافة *14* حرف من القيمة الأولى.

ملاحظات على استخدام تعليمة *Print*:

- إن تعليمة ***Print*** بدون أي قيمة بعدها تؤدي إلى طباعة سطر فارغ.
- إن تعليمة ***Print*** وبعدها ثابت عددي تؤدي إلى طباعة الرقم على الشاشة.
- إن تعليمة ***Print*** وبعدها ثابت حرفي (أي جملة محاطة بإشارتي تنصيص) تؤدي إلى طباعة الجملة.

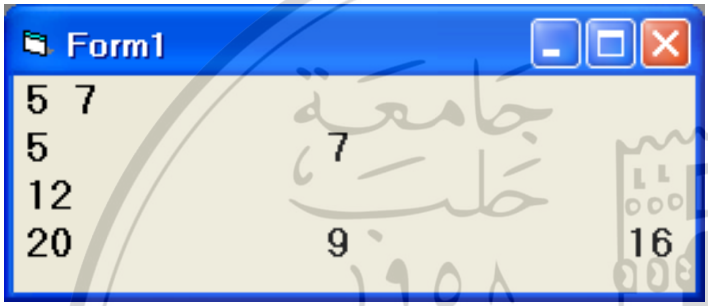
```
Print 10
Print
Print "Hammad"
```



- إن تعليمة ***Print*** وبعدها تعبير حسابي (أرقام أو متحولات أو الاثنين معاً) يؤدي إلى طباعة نتيجة العملية الحسابية:

```
X = 5
Y = 7
Print X; Y
Print X,Y
Print X + Y
Print 4 * X,2 + Y,4 ^ 2
```

وتكون النتيجة كما يلي:



- إن تعليمة *Print* وبعدها اسم متحول حرفي أو رقمي تؤدي إلى طباعة قيمة المتحول:

```
X = 5
Name1$ = "Hammad"
Print X; Name1$
Print X,Name1$
```

- يمكن الاستعاضة عن تعليمة الطباعة *Print* بإشارة استفهام.

```
Print 2 * 5    هذا يكافئ    ? 2 * 5
```

- يمكن وضع أكثر من قيمة بعد تعليمة ***Print*** ونفصل بينهما بفواصل عادية أو منقوطة:
- الفاصلة المنقوطة (;) بين قيمتين تؤدي إلى طباعة القيمة الثانية بجانب الأولى تماماً.
- الفاصلة العادية (,) تؤدي إلى طباعة القيمة الثانية على مسافة **14** حرف من القيمة الأولى.
- يمكن استخدام أكثر من فاصلة عادية عندها سيترك الحاسب مجموعة من المسافات الطباعية التي طول كل منها *14* حرفاً وذلك حسب عدد الفواصل العادية.

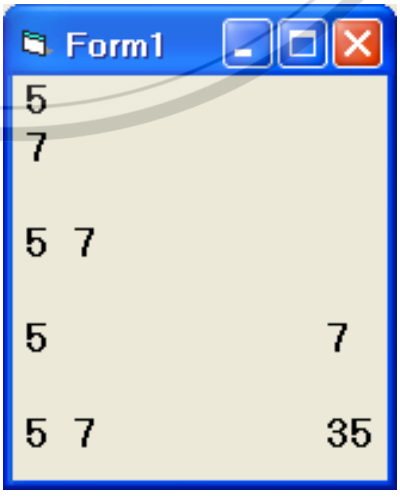
- إن استخدام فاصلة منقوطة واحدة أو اثنتين أو أكثر سيؤدي إلى نفس الغرض ولذا لا داعي إلى استخدام أكثر من فاصلة منقوطة واحدة.

```
X = 5
Y = 7
Print X; Y
Print X; ; Y
Print X; ; ; Y
Print X,Y
Print X,,Y
Print X,,,Y
```

وتكون النتيجة كما يلي:

- يمكن استخدام أكثر من تعليمة ***Print*** على سطر واحد شرط أن يفصل بينهما النقطتين العلويتين (:).
- مع كل تعليمة ***Print*** تبدأ الطباعة من سطر جديد، إلا إذا كان أمر الطباعة السابق يحتوي على فاصلة منقوطة أو عادية في نهاية السطر، عندئذٍ سوف تتم الطباعة على السطر القديم بجانب القيمة السابقة تماماً وتسمى طباعة متقاربة في حالة الفاصلة المنقوطة، وعلى مسافة *14* حرفاً تسمى منطقة طباعية في حالة الفاصلة العادية.

```
X = 5 : Y = 7
Print X: Print Y
Print
Print X;
Print Y
Print
Print X,
Print Y
Print
Print X;
Print Y,
Print X * Y
```

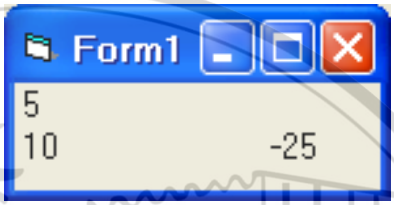


مثال:

```
X$ = "Hello": Y% = 20
Print X$: Print Y%
Print X$, Y%
Print X$; Y%;
Print 25 + 15 * 5
```

مثال:

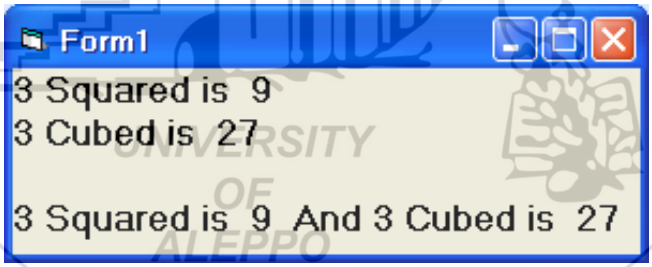
```
X = 5
Print X
Print X + 5, X * (-5)
```



مثال:

```
X = 3
Print X; " Squared is "; X ^ 2
Print X; " Cubed is "; X ^ 3
Print
Print X; " Squared is "; X ^ 2; " And ";
Print X; " Cubed is "; X ^ 3
```

وتكون النتيجة كما يلي:



- نلاحظ أن تعليمة ***Print*** الخامسة قد طبعت القيم على نفس السطر بسب وجود الفاصلة المنقوطة في نهاية تعليمة ***Print*** الرابعة.

مثال:

```
X$ = " We Are "
Y% = 3
Z$ = " Friends "
Print X$; Y%; Z$
Print X$, Y%; Z$
Print X$, Y%, Z$
```

حالات استخدام أخرى لتعليمة *Print*:

a) تعليمة الطباعة على عمود محدد $Print\ Tab(N)$:

تستخدم هذه التعليمة لتحديد العمود المراد الطباعة عليه. والشكل العام لهذه التعليمة هو:

Print Tab(N)

حيث ***N*** رقم يشير إلى رقم العمود التي سوف تتم الطباعة عليه.

```
A = 7
Print Tab(15); 5
Print Tab(18); "A"
Print Tab(12); A
```

- تؤدي تعليمة الطباعة الأولى إلى طباعة الرقم **5** على العمود رقم **15**.
- تؤدي تعليمة الطباعة الثانية إلى طباعة المحرف ***A*** على العمود رقم **18**.
- تؤدي تعليمة الطباعة الثالثة إلى طباعة قيمة المتحول ***A*** على العمود رقم **12**.

b) الطباعة مع التقريب $PRINT\ USING\ "###.##";\ A$:

الشكل العام لهذه التعليمة:

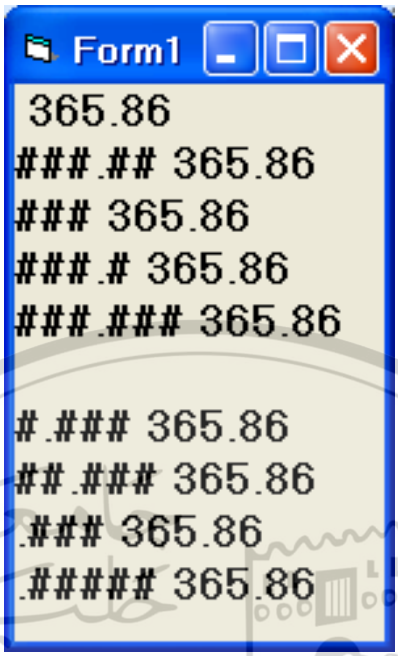
```
PRINT USING "###.##"; A
```

- **#** تمثل هذه الإشارة موضع طباعة الخانة الرقمية.
- **.** تمثل النقطة موضع الفاصلة العشرية.

<u>مثال:</u>

```
A = 365.86
PRINT A
PRINT USING "###.##"; A
PRINT USING "###"; A
PRINT USING "###.#"; A
PRINT USING "###.###"; A
PRINT
PRINT USING "#.###"; A
PRINT USING "##.###"; A
PRINT USING ".###"; A
PRINT USING ".#####"; A
```

وتكون النتيجة كما في الشكل:



- نلاحظ عند استخدام الأمر ***A*** ;"#####."***PRINT USING*** يجب زيادة أصفار مكان كل خانة عشرية إضافية.

c) تعليمة الإظهار أو الطباعة على الطابعة *LPRINT*:

- يقوم هذا الأمر بالطباعة إلى الطابعة بدلاً من الشاشة، وينطبق عليه كل ما ينطبق على تعليمة ***PRINT***.

الفصل الرابع
تعليمات التحكم وأدوات التحكم
Control Instructions and Control Tools

مقدمة:

بالحالة العادية يتم تنفيذ تعليمات برنامج معين بشكل متسلسل سطراً سطراً إلا إذا تغير التسلسل بناء على تحقق أو انتقاء شرط محدد وهذا يحدث باستخدام التعليمات الشرطية. إذاً التعليمات الشرطية هي التعليمات التي تغير التسلسل الطبيعي لتنفيذ تعليمات البرنامج بناء على شرط معين.

تعليمات التحكم Control Instructions:

١) تعليمة إذا الشرطية $If - Instruction$:

تسمح هذه التعليمة باتخاذ القرارات ضمن البرنامج اعتماداً على تحقق شرط معين، أو بكلمة أخرى تسمح هذه التعليمة بتنفيذ إحدى مجموعتين من التعليمات بناءً على تحقق شرط معين أو عدم تحققه. والشكل العام لهذه التعليمة هو:

```
If Expression1 Then
        VB Code1
[Else If Expression2 Then
        VB Code2]
[Else If expression2 Then
        VB Code3]
- - - - - - - - - - -
[Else
        VB Code4]
End If
```

حيث:

If بداية العبارة الشرطية.

Expressions تعابير منطقية رياضية تحدد سير وجهة التنفيذ.

VB Code's كتل التعليمات أو مجموعة الأكواد التي يتم تنفيذها في المرحلة التنفيذية للبرنامج.

Else If وإلا إذا.

Else وإلا.

End If نهاية الشرط.

ملاحظة: العبارات الموجودة ضمن أقواس يمكن حذفها.

ملاحظات على استخدام التعليمة *If*:

a) التعليمة ذات الخيار الواحد:

إن أبسط أشكال هذه التعليمة هو عند حذف الأقواس المتوسطة فيبقى بداية الشرط ونهايته مع حذف كل الاحتمالات الإختيارية، أي يتم اختيار حالة واحدة إذا كانت صحيحة نفذت الحلقة وإلا يتم الخروج من الحلقة.

```
If X > 0 Then
  Y = 5 * X
End If
```

b) التعليمة ذات السطر الواحد:

يمكن كتابة التعليمة على سطر واحد، عندها يمكن إهمال كلمة ***End If*** وتسمى عندها تعليمة ***If*** ذات السطر الواحد.

```
X = Y / 4
If X < 0 Then Text1.Text = "MINUS" Else Text1.Text = "PLUS"
```

c) استخدام *Else* :

عندها ممكن من نتيجة عملية المقارنة تطبيق حالة أولى في حالة الصح وحالة ثانية في حالة الخطأ.

```
A = 3 * Z
IfA > 0 Then
  Y = A + 5
Else
  Y = A - 5
End If
```

d) استخدام $Else\ If$**:**

يساعد على تنفيذ إحدى مجموعة تعليمات (كتلة تعليمات) من بين عدد كبير من كتل التعليمات المطروحة أو المختلفة.

```
A = 3 * Z
If A > 0 Then
 Text1.Text = "Larger from the Zero"
Else If A < 0 Then
Text1.Text = "Smaller from the Zero"
Else
 Text1.Text = "Equal the Zero"
End If
```

e) حالات If **المتداخلة:**

يمكن للتعليمة الشرطية أن تضم تعليمه شرطية أخرى شرطية ألا تتقاطع التعليمات الداخلية مع الخارجية. في هذه الحالة تتم مناقشة الشرط الداخلي وتنفيذ أحد خياراته بناء على الشرط الذي يتم اختباره وفي ضمن أحد الاحتمالات الممكنة من الشرط الخارجي.

```
A = 3 * Z
If A = 0 Then
   Text1.Text = "Equal"
Else
    IF A < 0 Then
   Text1.Text = "Smaller"
    Else
   Text1.Text = "Larger"
   End If
End If
```

٢) العبارة الشرطية $Select\ Case$**:**

في كثير من الأحيان يكون لدينا عدد كبير جداً من الاحتمالات تصبح عندها تعليمة If طويلة جداً ولذا فإن استخدام عبارة التعليمة ($If - Else\ If - Else - End$) والذي سيؤدي إلى مناقشة شرطها الأول ثم الثاني ثم الثالث وهكذا سيصبح الأمر متعباً ومُملاً وطويلاً، ولذا وفي مثل هذه الحالات يكون استخدام العبارة $Select\ Case$ أكثر ملائمة من استخدام التعليمة If. والشكل العام لهذه التعليمة هو:

```
Select Case A
Case 1
```

```
 Text1.Text = "There Are No Solution"
Case 2
 Text1.Text = "There Are Double Solution"
[Case Else
 Text1.Text = "There Are Tow Solution"]
End Select
```

هنا بناء على قيمة المتحول **A** يتم تنفيذ كتلة التعليمات المتعلقة باحتمال واحد فقط من بين هذه الاحتمالات ويخرج خارج الحلقة أي ينتقل إلى ما بعد العبارة **End Select**.

وإن لم تكن قيمة **A** موجودة بين الخيارات الموجودة فإنه لن يدخل إلى داخل حلقة **Select** على الإطلاق ويقوم بتنفيذ السطر الذي يليها مباشرةً أي ينتقل إلى ما بعد العبارة **End Select**. ويمكن استخدام تعليمة **Select** بعدة أشكال أخرى كما هو مبيّن:

```
A = InputBox ("Select Your Case")
Select Case A
 Case 1
 Text1.Text = "Your Select Is " & A
 Case 3 To 5
 Text1.Text = "Your Select Is Between 3 And 5 , It Is = " & A
 Case 6, 9, 14
 Text1.Text = "Your Select Is One of This Numbers 6 ,9 , 14 , It Is = " & A
 Case Else
 Text1.Text = "Your Select Is Difference of 1,3 ... 5, 6, 9 , 14, It Is = "&A
End Select
```

- **Case 1**: أي تحديد قيمتها مباشرة بشكل صريح وهنا مساوية للواحد.
- **Case 3 To 5**: أي تحديد قيمتها بين مجال من القيم.
- **Case 6, 9, 14**: أي تحديد قيمتها بين مجموعة من القيم المحددة.
- **Case Else**: عندما تكون قيمتها مخالفة لكل القيم المذكورة سابقاً.
- ومن الواضح أن أي قيمة لـ **A** ستنفذ أحد الاحتمالات وذلك لوجود **Case Else**.
- إذا حذفنا تعليمة **Case Else** ستبقى الاحتمالات هي **1, 3, 4, 5, 6, 9,14,**

ويمكن أن نستخدم الصيغة التالية:

```
Case 1 To 4, 7 To 9, 11, 13
```

- وهذا يعني إذا تحقق أحد الحالات من **1** إلى **4** أو من **7** حتى **9** أو **11** أو **13** عندها سيتم تنفيذ الكود المرافق ... الخ.

٣) تعليمة القفز غير المشروط $GoTo$:

وهي التعليمة التي ستنقلنا قسريّاً إلى السطر الذي يحمل الرقم $Line\ Number$ ومن ثم يستمر البرنامج بتنفيذ تعليماته ابتداءً من هذا السطر. والشكل العام لهذه التعليمة هو:

$$GoTo\ Line\ Number$$

قد يكون رقم السطر المطلوب الانتقال إليه قبل تعليمة $GoTo$ أو بعدها، لكنه حتماً ستقع ضمن الوحدة البرمجية الواحدة وهذا يعني أن تعليمة $GoTo$ ستنقل تنفيذ أوامر البرنامج إلى الخلف أو ستنقله إلى الأمام.

```
10  A = InputBox(" Insert the Number  A")
    B = InputBox(" Insert the Number  B")
    C = InputBox(" Insert the Number  C")
    D = B ^ 2  −  4  ∗  A  ∗  C
    If D  <  0 Then
      MsgBox "There Are No Solution"
      GoTo 10
    Else
      MsgBox "There Are Two Solution"
      GoTo 100
    End If
100 X1 = (−B  +  Sqr(D)) / 2  ∗  A : Text1.Text = X1
    X2 = (−B  −  Sqr(D)) / 2  ∗  A : Text2.Text = X2
```

٤) تعليمة انتقاء التفرع $On\ k\ GoTo$:

وهي التعليمة التي ستنقلنا قسريّاً إلى سطر ما من بين مجموعة من الأسطر وتشبه إلى حد ما كل من التعليمتين $GoTo$ وتعليمة $Select\ Case$. والشكل العام لهذه التعليمة هو:

$$On\ k\ GoTo\ n1, n2, n3$$

تتألف هذه التعليمة من التعليمة On يليها رقم فهرس k ومن ثم تعليمة $GoTo$ ولائحة بأرقام الأسطر (التي من الممكن اختيارها) ويتم انتقاء الأسطر وفقاً لقيمة الفهرس. فإذا كان الفهرس مساوياً للواحد يتم انتقاء الفرع الأول وإذا كان الفهرس مساوياً اثنين فيتم انتقاء التفرع الثاني وهكذا.

ملاحظة: إذا كان الفهرس مساوياً للصفر أو رقم أكبر من عدد التفرعات الموجودة ينتقل التنفيذ إلى السطر الذي يلي تعليمة الانتقاء دون تنفيذ أي تفرع من هذه التفريعات. وإذا كان الفهرس سالباً أو أكبر من العدد **255** نحصل على عبارة الخطأ التالية:

```
Run-time error '5':
Invalid procedure call or argument
```

تمرين: على استخدام تعليمة انتقاء التفرع *On k GoTo*.

```
20  num = Val(InputBox("Insert the Select Number"))
    On num GoTo 80, 100
    MsgBox "You Selected Number Not in The List"
    MsgBox "Your Program Will End Now"
    End
80  MsgBox "You Are at Line 80"
    GoTo 20
100 MsgBox "You Are at Line 100"
    GoTo 20
```

٥) تعليمة الانتقال غير المشروط إلى البرامج الفرعية (الروتينات الفرعية) *GoSub*:

وهي التعليمة التي ستنقلنا قسريّاً إلى سطر ما يبدأ عنده برنامج فرعي (روتين فرعي) معيّن يتم في هذا الروتين تنفيذ مجموعة من الأسطر والتعليمات وبعده يتم إجبار البرنامج إلى العودة إلى التعليمة التي تلي تعليمة النقل القسري. والشكل العام للتعليمة هو:

```
GoSub NUMBER LINE
- - - -
 VB Code 1
- - - -
NUMBER LINE
- - - -
 VB Code 2
- - - -
Return
```

- تعتبر الروتينات الفرعية (التفرعات) برامج جزئية ضمن البرنامج الأصلي وتساعد على تجزئة البرنامج إلى مجموعات وظيفية مع إمكانية الوصول إليها تكراراً عند اللزوم.
- يتم التفرع إلى البرنامج الجزئي باستخدام التعليمة ***GoSub*** يليها رقم سطر يشير إلى بداية البرنامج الجزئي.

- يتم تنفيذ جميع الأسطر الموجودة داخل البرنامج الجزئي سطراً سطراً.
- تتم العودة إلى رقم السطر الذي يلي تعليمة استدعاء البرنامج وذلك عند الوصول إلى سطر يحوي تعليمة ***Return***.
- يمكن الإشارة إلى روتين جزئي باسم ***Label*** وليس برقم وعندها يجب إضافة نقطتين علويتين بعد الاسم ***Label*** في مكان وجوده بالبرنامج.
- من المفيد إدراج ملاحظة قبل تعليمة الانتقال إلى الروتين الجزئي تشير إلى اسم الروتين الجزئي (أو وظيفة أو مهمة أو مضمون أو رقم سطر الروتين الجزئي). الذي سيتم الانتقال إليه أو تحديد مهمته أو مضمونه.

تمرين: برنامج لحساب أس (قوة) عدد ما يتم إدخاله.

١. **حل التمرين باستخدام التعليمة الشرطية *If – Else – End If* وتعليمات *GoTo* و *GoSub*.**

```
    A = InputBox("Input The Base A")
    N = InputBox("Input The Power N")
    If N > 0 Then
      GoSub 10
      GoTo 20
    Else
      GoSub 10
    Text1.Text = A & " ^ " & N & " = " & 1 / F
    End If
    GoTo 100
10  F = 1
    For i = 1 To Abs(N)
       F = F * A
    Next i
    Return
20  Text1.Text = A
    For i = 1 To N - 1
       Text1.Text = Text1.Text & " * " & A
    Next i
    Text1.Text = Text1.Text & " = " & F
100 Text2.Text = "End"
```

٢. حل التمرين باستخدام التعليمة الشرطية If ذات السطر الواحد وتعليمات $GoTo$ و $GoSub$.

```
    A = InputBox("Input The Base A")
    N = InputBox("Input The Power N")
    If N > 0 Then GoSub 10: GoTo 20
    GoSub 10
    Text1.Text = A & " ^ " & N & " = " & 1 / F
    GoTo 100
10  F = 1
    For i = 1 To Abs(N)
       F = F * A
    Next i
    Return
20  Text1.Text = A
    For i = 1 To N - 1
       Text1.Text = Text1.Text & " * " & A
    Next i
    Text1.Text = Text1.Text & " = " & F
100 Text2.Text = "End"
```

٣. حل التمرين باستخدام التعليمة الشرطية $If-Else-End\ If$ وتعليمات $GoTo$ و $GoSub$.

تعليمة $GoSub$ باستخدام $Label = Hammad$:

```
    A = InputBox("Input The Base A")
    N = InputBox("Input The Power N")
    If N > 0 Then
      GoSub Hammad
      GoTo 20
    Else
      GoSub Hammad
    Text1.Text = A & " ^ " & N & " = " & 1 / F
    End If
    GoTo 100
Hammad:
    F = 1
    For i = 1 To Abs(N)
       F = F * A
    Next i
```

```
    Return
20  Text1.Text = A
    For i = 1 To N - 1
       Text1.Text = Text1.Text & " * " & A
    Next i
    Text1.Text = Text1.Text & " = " & F
100 Text2.Text = "End"
```

٤. حل التمرين باستخدام التعليمة الشرطية *If* ذات السطر الواحد وتعليمات *GoTo* و *GoSub*.

تعليمة *GoSub* باستخدام *Label = Hammad*:

```
    A = InputBox("Input The Base A")
    N = InputBox("Input The Power N")
    If N > 0 Then GoSub Hammad: GoTo 20
    GoSub Hammad
    Text1.Text = A & " ^ " & N & " = " & 1 / F
    GoTo 100
Hammad:
    F = 1
    For i = 1 To Abs(N)
       F = F * A
    Next i
    Return
20  Text1.Text = A
    For i = 1 To N - 1
       Text1.Text = Text1.Text & " * " & A
    Next i
    Text1.Text = Text1.Text & " = " & F
100 Text2.Text = "End"
```

٦) تعليمة الانتقال المشروط إلى البرامج الفرعية (الروتينات الفرعية) *On k GoSub*:

تسمح هذه التعليمة بانتقاء أحد الخيارات المتاحة بعد ***GoSub*** بناء على قيمة الفهرس ***k***. والشكل العام للتعليمة هو:

```
On k GoSub N1, N2, N3
N1 VB Code1
- - - -
Return
```

```
N2 VB Code2
– – – –
Return
N3 VB Code3
– – – –
Return
```

تتألف هذه التعليمة من التعليمة **On** يليها رقم فهرس **k** ثم **GoSub** ولائحة بأرقام الأسطر ويتم انتقاء الروتين الفرعي وذلك وفقاً لقيمة الفهرس. فإذا كان الفهرس مساوياً لواحد **1** سيتم انتقاء الروتين الفرعي الأول الذي يبدأ سطره بالرقم **N1** وإذا كان الفهرس مساوياً لاثنين 2 فسيتم انتقاء الروتين الفرعي الثاني الذي يبدأ بالرقم **N2** وهكذا دواليك. وإذا كان رقم الفهرس مساوياً للصفر أو رقم أكبر من عدد أرقام السطور يستمر التنفيذ إلى السطر الذي يلي سطر تعليمة التفرع. وإذا كان الفهرس سالباً أو أكبر من العدد **255** نحصل على عبارة الخطأ التالية:

```
Run-time error '5':
Invalid procedure call or argument
```

مثال على تعليمة الانتقال المشروط إلى البرامج الفرعية (الروتينات الفرعية) On k GoSub:

```
A = InputBox("Insert the Index A")
On A GoSub 10, 20, 30
Text1.Text = B
GoTo 100
10 B = 5
Return
20 B = 6
Return
30 B = 7
Return
100
```

```
A = InputBox("Insert the Index A")
On A GoSub 10, 20, 30
Text1.Text = B
GoTo 100
10 B = 5 : Return
20 B = 6 : Return
30 B = 7 : Return
100
```

عند إدخال القيمة **2** سيتم انتقاء الروتين الثاني الذي يبدأ من السطر **20** وحتى **Return** التي تليه، بعدها يعود البرنامج إلى السطر الذي يلي تعليمة **GoSub** فيتم إظهار قيمة **B** في مربع النص ثم يتابع إلى السطر رقم **100** بتوجيه من التعليمة **GoTo**. طبعاً من الواضح أننا نستطيع استخدام تعليمة **Return** بعد أي تعليمة أخرى ولكن بشرط أن يفصل بينهما النقطتين العلويتين.

مثال: أكتب برنامجاً يستخدم التعليمتين *On k GoSub* و *Sgn* لحساب *Y* وفقاً لما يلي:

$$y = \begin{cases} x^2 + 3 & If \quad x < 0 \\ Y = 5 & If \quad x = 0 \\ e^x - 1 & If \quad x > 0 \end{cases}$$

الحل:

```
    X = InputBox("Insert the Index X")
    N = Sgn(X) + 2
    On N GoSub 10, 20, 30
    GoTo 100
10  Y = X ^ 2 + 3
    Text1.Text = " Y = x ^ 2 + 3 = " : Text2.Text = Y
    Return
20  Y = 5
    Text1.Text = " Y = " : Text2.Text = Y
    Return
30  Y = Exp(X) - 1
    Text1.Text = "Y = Exp (x) - 1 = " : Text2.Text = Y
    Return
100
```

مثال: برنامج انتقال تفرعي يرمز فيه للبرنامج باسم *Label* وليس برقم السطر الذي يبدأ به.

يتم إدخال قيمة معينة ***N*** والتي يجب أن تكون إمّا **1** أو **2** حتى يتم تنفيذ تعليمة ***GoSub*** وذلك لوجود احتمالين فيها هما ***AA, BB***.

```
N = InputBox("Insert Number N")
On N GoSub AA, BB
GoTo 100
AA:
MsgBox "Dr M Hammad Says"
Return
BB:
MsgBox " Hello My Good Students "
Return
100
```

```
N = InputBox("Insert Number N")
On N GoSub AA, BB
GoTo 100
AA: MsgBox "Dr M Hammad Says"
Return
BB:
MsgBox " Hello My Good Students "
Return
100
```

الآن إذا أدخلنا قيم مخالفة أي ($N = 0, 3, 4, \ldots$) عندها سيقوم البرنامج بتنفيذ التعليمة التي تلي سطر ***GoSub*** والتي سترسلنا إلى السطر رقم **100** والذي لا يحوي على أي كود أي أن البرنامج سيصل إليه ولا ينفّذ أي عملاً آخراً.

إذا أدخلنا $N = 1$ سيتم اختيار الاحتمال الأول أي سيتم الانتقال إلى الروتين الجزئي الذي اسمه ***AA*** فينفذه تعليماته حتى التعليمة ***Return*** والتي ستعيده إلى التعليمة التي بعد ***GoSub*** والتي سترسلنا إلى السطر رقم **100**. أمّا إذا أدخلنا $N = 2$ سيتم اختيار الاحتمال الثاني أي سيتم الانتقال إلى الروتين الجزئي الذي اسمه ***BB*** فينفذه تعليماته حتى التعليمة ***Return*** والتي ستعيده إلى التعليمة التي بعد ***GoSub*** والتي سترسلنا إلى السطر رقم **100**.

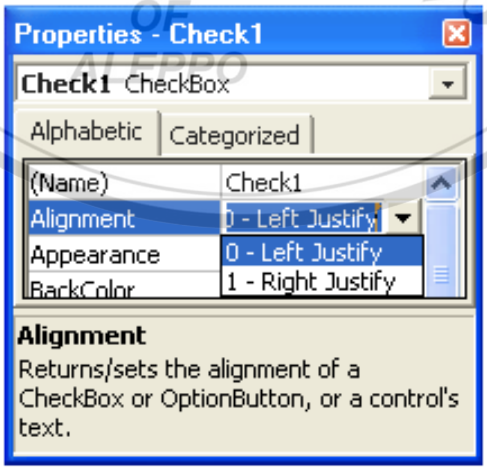
أدوات التحكم *Control Tools*:

خانة الاختيار *Check Box*:

تستخدم في حالات عديدة، لكن أهم استخدام لها هي عندما نرغب في اختيار حالة واحدة أو عدة خيارات معاً. ويمكن عن الاختيار أن نختار خيار واحد أو كل الخيارات دون أن يؤثر أحد الاختيارات على الاختيارات الأخرى والموجودة في نفس المجموعة. ومن أهم خصائص هذه الأداة:

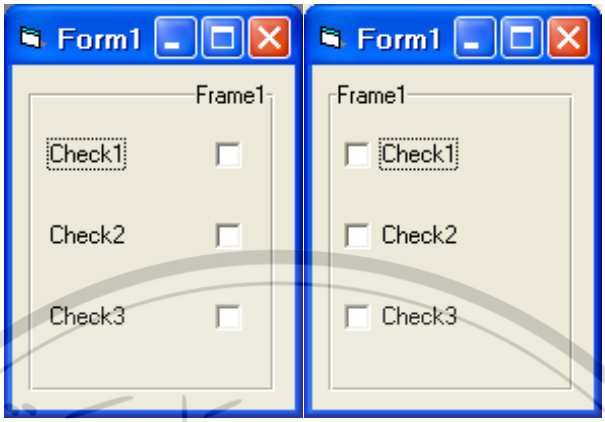
الخاصية *Alignment*:

وهي الخاصية المسؤولة عن محاذاة الكتابة ومكانها بالنسبة للأداة ولها احتمالين هما:



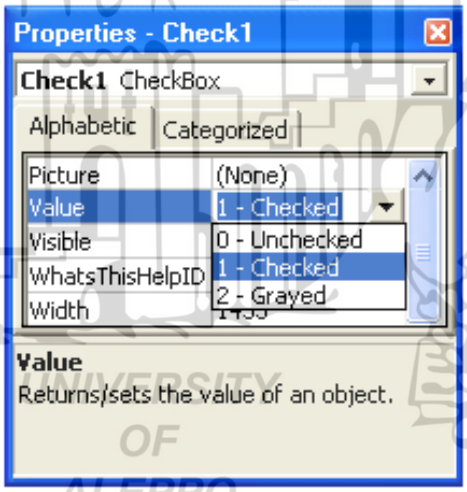
- عندما تكون قيمة الخاصية: ***Alignment = 0 – Left Justify*** عندها سيكون التنسيق لليسار أي أن خانة الاختيار ستظهر على يسار النص.

- عندما تكون قيمة الخاصية: $\boldsymbol{Alignment = 1 - Right\ Justify}$ عندها سيكون التنسيق لليمين أي أن خانة الاختيار ستظهر على يمين النص.



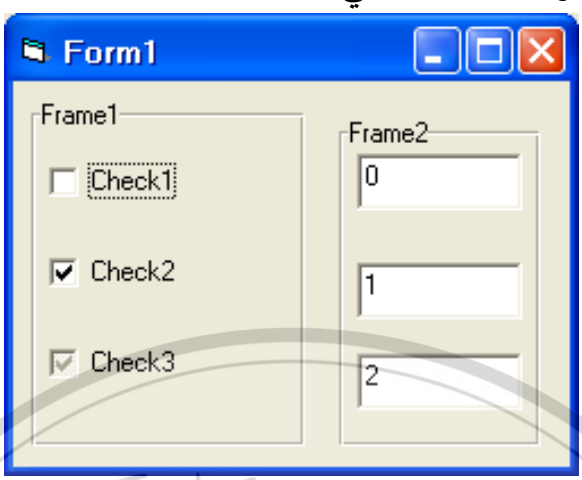
الخاصية $\boldsymbol{Value}$:

وهي الخاصية المسؤولة عن حالة اختيار هذه الأداة. ومن خلال هذه الخاصية يمكن معرفة إذا تم اختيار هذه الخانة أم لا. ولها احتمالات ثلاثة:



- عندما تكون قيمة الخاصية: $\boldsymbol{Value = 0 - Unchecked}$ أي أن الخانة ستظهر من دون علامة الاختيار (أي أننا لم نقوم باختيار هذه الخانة).
- وعندما تكون قيمة الخاصية: $\boldsymbol{Value = 1 - Checked}$ أي أن الخانة ستظهر معلّمة بالعلامة √ (أي أننا قمنا باختيار هذه الخانة).
- وعندما تكون قيمة الخاصية: $\boldsymbol{Value = 2 - Grayed}$ أي أن الخانة ستظهر معلّمة بالعلامة √ ولكن لونها باهت غير فعال (أي أنها غير متاحة في هذه المرحلة من البرنامج ولا نستطيع استخدامها).
- تستخدم هذه الخاصيّة في المرحلة التصميمية وفي المرحلة الكودية (التنفيذية).

وتظهر في البرمجة المرئية بالشكل التالي:



وفي البرمجة الكودية:

```
If Check1.Value = 0 Then
        [VB Code1]  ' الأداة غير مختارة
ElseIf Check1.Value = 1 Then
        [VB Code2]  '  الأداة مختارة
Else
        [VB Code3]  ' الأداة غير متاحة للاستخدام
End If
```

ويمكن صياغة البرمجة الكودية كما يلي:

```
If Check1.Value = VbUnchecked Then
        [VB Code1]  ' الأداة غير مختارة
ElseIf Check1.Value = VbChecked Then
        [VB Code2]  '  الأداة مختارة
Else Check1.Value = VbGrayed Then
        [VB Code3]  ' الأداة غير متاحة للاستخدام
End If
```

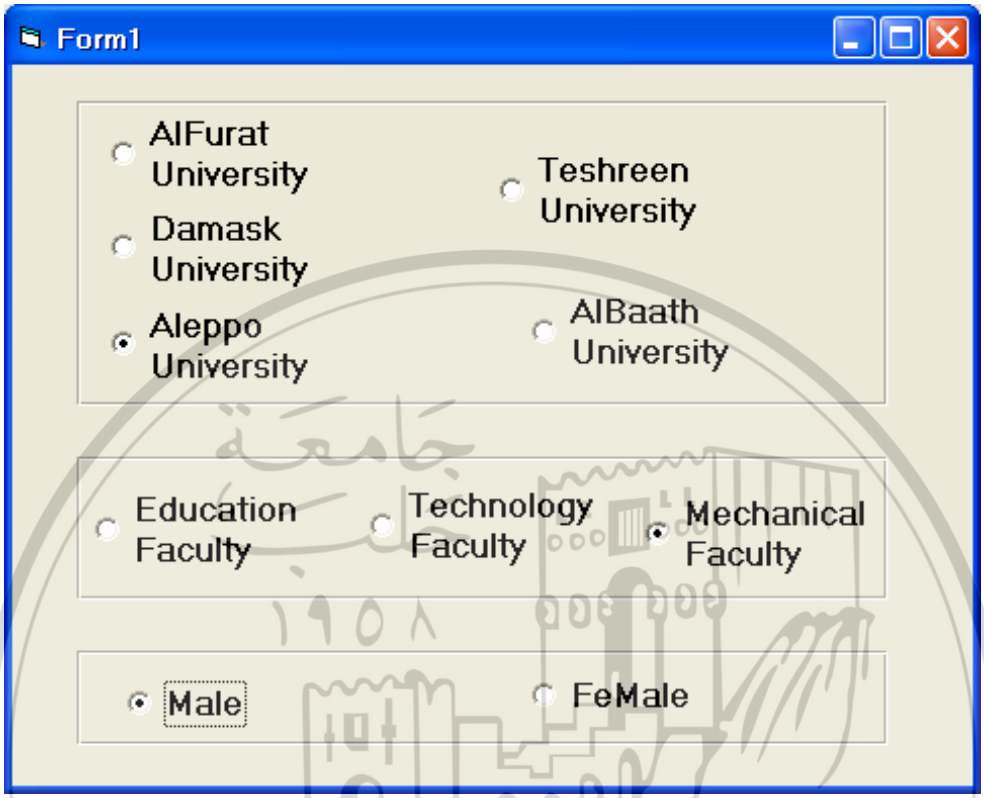
فهي تتضمن إذاً تعليمة ***If*** الشرطية.

- تأخذ خانة الاختيار القيمة (1) في حالة الاختيار، بينما تأخذ القيمة (0) في حالة عدم اختيارها. إذاً وظيفة مربع الاختيار ***Check box*** أخذ الإجابة بصـــورة محددة من المستخدم (نعم أو لا) .

زر الخيار *OptionButton*:

وهو بعكس خانة الاختيار، حيث يتم اختيار حالة وحيدة من مجموعة من الخيارات المعروضة. يجب أن تتوضع أزرار الخيارات ضمن الأداة ***Frame*** أثناء البرمجة المرئية،

وذلك من أجل تمييز مجموعة خيارات عن أخرى أثناء العمل البرمجي ففي حالة وجود مجموعة خيارات وحيدة لا يكون بالضرورة وضعها ضمن الأداة ***Frame***.



إذاً عندما تتعدد الاختيارات لا يصلح مربع الاختيار وإنما نلجأ إلى أزرار الاختيار والتي تستعمل دائما في مجموعات، عند اختيار أحدها يتم تعطيل الاختيار عن بقيتها.

تأخذ البرمجة الكودية لزر الخيار الشكل التالي:

```
If Option1.Value = True Then
  VB Code 1
Else
  VB Code 2
End If
```

- يأخذ زر الخيار القيمة ***True*** في حالة الاختيار، بينما يأخذ القيمة ***False*** في حالة عدم اختياره.

نتيجة:

- تأخذ خانة الاختيار قيمة عددية، بينما يأخذ زر الخيار قيمة منطقية.

تذكرة:

- يتم تغيير عناوين الأدوات عن طريق صندوق الخصائص عبر الخاصية ***Caption***.

الخانة المركبة ComboBox :

استخدام أداة الخانة المركبة:

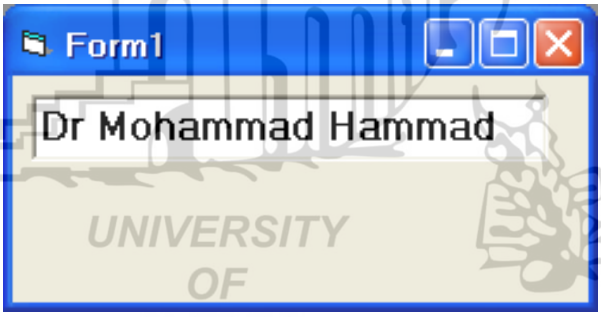
تجمل هذه الأداة الملامح الأساسية الموجودة في الأداة النصية ***TextBox*** وأداة اللائحة ***List***. فتستخدم للإدخال وللإخراج إذ يمكن إدخال معطيات جديدة في خانة النص أو اختيار عنصر موجود في عناصر لائحتها.

إضافة وإزالة العناصر من الخانة المركبة:

يمكن إضافة عنصر ما في المرحلة التصميمية بإستخدام الخاصية ***List*** فيها ويتم إدخال أكثر من سطر بإستخدام التركيب [***Ctrl + Enter***] وأثناء التنفيذ يتم إختيار هذه العناصر من القائمة المنسدلة. كما يمكن إضافة وحذف عنصر في المرحلة التنفيذية بإستخدام ***AddItem*** و ***RemoveItem*** كما يلي:

$$CmbBox1.AddItem\ ''Dr\ Mohammad\ Hammad''$$

يقوم الأمر المكتوب بإضافة العبارة المبيَّنة في الخانة المركبة وتظهر في المرحلة التنفيذية كما يلي:



$$CmbBox1.AddItem\ ''My\ Name\ is\ Hammad\ ''$$

تقوم بإضافة العبارة المبين في الخانة المركبة.

$$Combo1.Text\ =\ ""$$

مسح أو تنظيف أو عدم إظهار أي كتابة في منطقة النص في الخانة المركبة.

تظهر التعابير داخل أداة الخانة المركبة بشكل مرتب من الأعلى إلى الأسفل، ويقوم ***VB*** بإعطاء دليل أو ***Index*** لكل تعبير أو عنصر من العناصر الموجودة داخل أداة الخانة المركبة. ويكون دليل العنصر الأول **0** ودليل العنصر الثاني **1** وهكذا ... الخ.

$$Combo1.Text\ =\ Combo1.List(1)$$

تقوم بإظهار العنصر الثاني في منطقة النص في الخانة المركبة. وفي هذه العبارة تم إظهار العنصر الثاني عند اختيار الـ $(\boldsymbol{Index = 1})$ وذلك كون الـ **0** يدل على العنصر رقم واحد.

$Combox.RemoveItem\ index$

تقوم بإزالة العنصر الذي ترتيبه هو الرقم الموجود مضافاً إليه واحد.

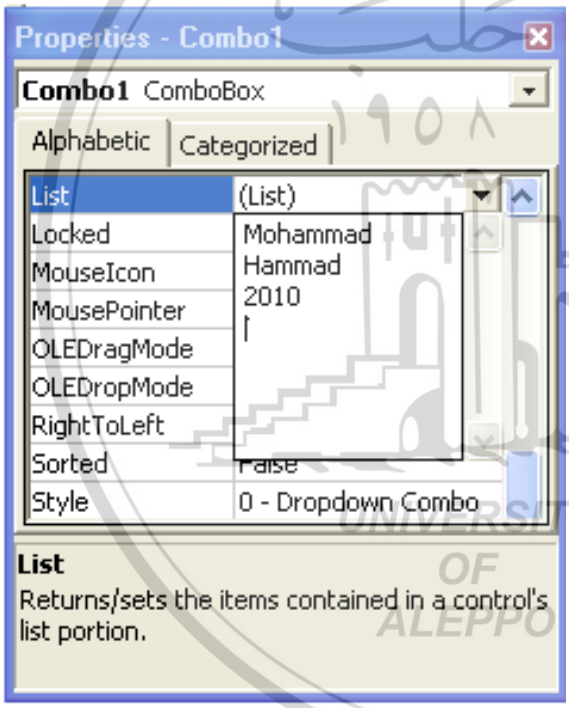
$ComboBox.Clear$

تقوم بإزالة كل العناصر الموجودة في الخانة المركبة (القائمة).

ومن أهم خصائص هذه الأداة وحسب ورودها بالتسلسل الأبجدي في نافذة الخصائص:

Name: وهو الاسم الكودي الذي يتعامل معه البرنامج لهذه الأداة.

List: الخاصية التي تسمح بإدخال العناصر إلى الخانة المركبة أثناء التصميم.



- يتم إدخال العناصر في المرحلة المرئية من خلال النقر على الخيار $List$ الموجود بجانب الخاصية $List$.
- يتم كتابة العنصر الأول وهو العنصر الذي دليله (0).
- إذا أردنا إدخال عنصر آخر نستخدم التركيب $[Ctrl + Enter]$ عندها ينتقل المؤشر إلى السطر التالي.
- أما لإنهاء عملية إدخال العناصر في المرحلة المرئية نستخدم المفتاح $Enter$.

$Locked$: تسمح هذه الخاصية باختيار عناصر أو لا أثناء التنفيذ ولها قيمة منطقية $(T\ or\ F)$.

$Sorted$: تسمح هذه الخاصية بترتيب العناصر أبجدياً أثناء التنفيذ ولها قيمة منطقية $(T\ or\ F)$.

$Style$: تحدد هذه الخاصية الشكل الذي ستظهر فيه الأداة. ولها ثلاثة أشكال:

١) القيمة $Style = 0$:

وتظهر فيها أداة النص وقائمة منسدلة تنسدل عند النقر على السهم ليتم الاختيار من داخلها. ومن ميزات هذه الأداة في هذه الحالة:

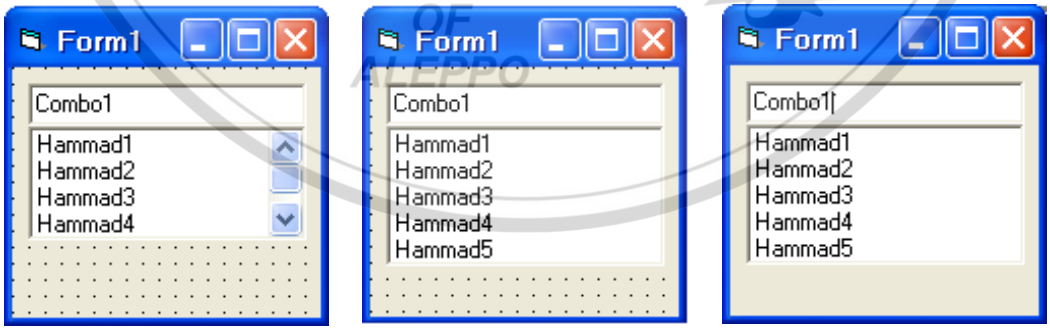
- يمكننا القراءة والكتابة في الخانة في هذه الحالة والنقر على القائمة المنسدلة.
- لا يمكن تغيير ارتفاع الأداة.

0 - Dropdown Combo **0 - Dropdown Combo** **0 - Dropdown Combo**

٢) القيمة $Style = 1$:

وتظهر فيها أداة النص وقائمة ثابتة مفتوحة دوماً. ومن ميزات هذه الأداة في هذه الحالة:

- يمكننا القراءة والكتابة في الخانة في هذه الحالة ولا يمكننا النقر على القائمة المنسدلة لأنها غير موجودة.
- يمكن تغيير ارتفاع الأداة ونلاحظ ظهور شـــريط التمرير عندما لا يكون ارتفاع الأداة كافٍ لإظهار كل العناصر فيها.

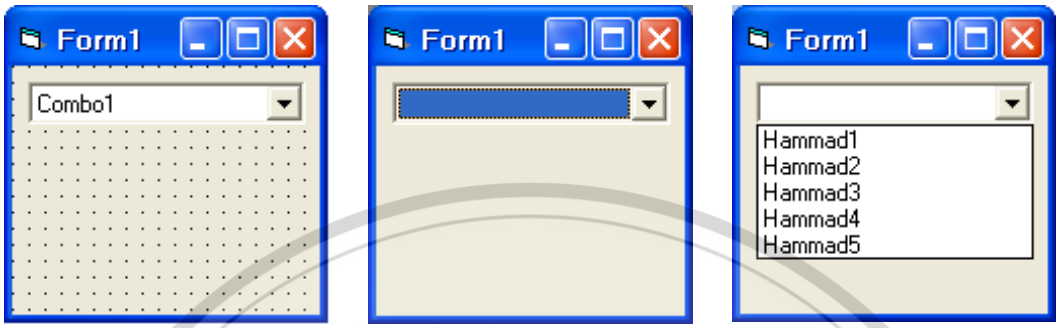


1 – Simple Combo Box **1 – Simple Combo Box** **1 – Simple Combo Box**

٣) القيمة $Style = 2$:

وتظهر فيها قائمة منســـدلة بدون خانة نص اي يمكن الإنتقاء منها ولا يمكن الكتابة فيها. ومن ميزات هذه الأداة في هذه الحالة:

- القيم الموجودة في الخاصـــية $Text$ القيم الموجودة في الأداة ســـتكون متوفرة للقراءة فقط أثناء التنفيذ ولا يمكن اختيارها أو تغييرها أثناء التصميم والتنفيذ.
- لا يمكن تغيير ارتفاع الأداة.



2 – Drop-Down List Box **2 – Drop-Down List Box** **2 – Drop-Down List Box**

ملاحظة: تبيّن الأشكال الثلاثة حالة الأداة في المرحلة المرئية (الأيسر)، وحالة الأداة عند لحظة البدء بالمرحلة التنفيذية (الأوسـط) وحالة الأداة بعد النقر عليها. نلاحظ أن ارتفاع الأداة ثابت عندما تأخذ الخاصية $Style$ القيم $0, 2$ وتتغير عندما $Style = 1$.

$Text$: تحدد هذه الخاصية النص الذي سيظهر في الخانة المركبة. وهذه القيمة ستكون للقراءة فقط عندما تأخذ الخاصـــية $Style$ القيمة $2 - Drop - down\ list\ box$ ولن نتمكن من تغيير قيمتها أثناء التنفيذ والتصميم.

$Visible$: وهي الخاصـــية التي تســـمح برؤية الأداة أثناء التنفيذ أو لا ولهما قيمتان $False$ و $True$.

الأحداث على الأداة:

هناك مجموعة من الأحداث التي يمكن تطبيقها على الأداة $ComboBox$ من أهمها الحدث $DropDown$:

يقع هذا الحدث عندما ينقر المستثمر على زر الاسم الموجود في الخانة المركبة وقبل أن تنسدل القائمة الخاصة بها. ولا يقع هذا الحدث إذا أخذت الخاصية $Style$ القيمة 1 لأن الحالة التصميمية الإفتراضية لهذه الأداة في هذه الحالة أن تكون مفتوحة دوماً.

جامعة
حلب
١٩٥٨
UNIVERSITY
OF
ALEPPO

الفصل الخامس
تعليمات وحلقات التكرار
Repetition Instructions and Loops

١) حلقة التكرار For – Next:

تعليمة التكرار: هي التعليمة التي تسمح بتنفيذ مجموعة من التعليمات عدداً من المرات.

حلقة التكرار: مجموعة التعليمات المكورة المحصورة بين كلمتي ***For*** و ***Next***.

متحول حلقة التكرار: متحول خاص مرتبط بتعليمة التكرار له قيمة بدائية وخطوة تزايد (أو تناقص) وقيمة نهائيه. (ولا يمكن استخدام عنصر مصفوفة كعدّاد للحلقة) ويتم تكرار العملية ما دامت قيمة المتحول لم تتجاوز الحد الأعلى.

والشكل العام لهذه التعليمة هو:

For *Counter* = *Start* ***To*** *End* [*Step Increment*]
    VB Code
Next *Variable*

حيث:

- ***Counter*** – اسم متحول حلقة التكرار.
- ***Start*** - القيمة البدائية التي يأخذها متحول حلقة التكرار.
- ***End*** – القيمة النهائية التي قد يأخذها (ولا يمكن أن يتجاوزها) متحول حلقة التكرار.
- ***Increment*** -قيمة الخطوة ***Step*** التي يتزايد أو يتناقص بقدرها متحول التكرار من ***Start*** إلى ***End***.
- *Statement – Block* – كتلة التعليمات التي يجري تكرارها ضمن الحلقة.

يبدأ تنفيذ حلقة التكرار بأن يأخذ المتحول القيمة البدائية و يبدأ بالتنفيذ التسلسل كتلة المعلومات حتى يصل إلى نهاية الحلقة عند ***Next*** عندها يأخذ قيمة جديدة (زيادة أو نقصان بمقدار الخطوة ***Step***) ثم يقوم البرنامج بفحص القيمة الجديدة فإذا لم تكن أكبر (أو أصغر في حالة الحلقات المتناقصة) من القيمة النهائية ***End*** عندها سيعود إلى بدايتها من جديد عن ***For*** ثم يبدأ بتنفيذ كتلة التعليمات من جديد حتى ***Next***

وهكذا حتى تصبح قيمة المتحول أكبر (أو أصغر) من قيمة الحد الأعلى (القيمة النهائية) فيخرج من الحلقة.

```
For I = 5 To 10 Step 2
    y = x + I
Next I
```

ملاحظات على تعليمة حلقة التكرار For − Next:

- يجب أن نحرص على كتابة الخطوة بشكل جيد إلا إذا كانت قيمتها تساوي الواحد فيمكن إهمالها.

```
For I = 5 To 10 Step 1
    y = x + I
Next I
```
الحلقة صحيحة

```
For I = 5 To 10
    y = x + I
Next I
```
الحلقة صحيحة

- عند تنفيذ الحلقة بشكل متناقص يجب أن نعطي للخطوة قيمة سالبة ولا يمكن إهمالها.

```
For I = 10 To 5 Step − 1
    y = x + I
Next I
```
الحلقة صحيحة

```
For I = 10 To 5
    y = x + I
Next I
```
الحلقة خاطئة

- عند استخدام خطوة سالبة يجب أن نتأكد أن قيمة التعبير البدائي قبل To أكبر من النهائي بعدها.

```
For I = 10 To 5 Step − 1
    y = x + I
Next I
```
الحلقة صحيحة

```
For I = 5 To 15 Step − 1
    y = x + I
Next I
```
الحلقة خاطئة

- يمكن أن نستخدم دليل الحلقة في تعليماتها على أن لا نغير من قيمته داخل الحلقة.

```
For I = 5 To 10
    y = x + I
    z = I * I
Next I
```
الحلقة صحيحة

```
For I = 5 To 10
    y = x + I
    I = I + 2
Next I
```
الحلقة خاطئة

```
For I = 5 To 10
    y = x + I
    I = I − 2
Next I
```
الحلقة خاطئة

- يمكن أن تكون القيمة البدائية والنهائية والخطوة تعابير حقيقية.

```
For I = 0.5 To 6.5
    y = x + I
Next I
```
الحلقة صحيحة

```
For I = 5 To 10 Step 0.1
    y = x + I
Next I
```
الحلقة صحيحة

- يمكن أن تكون القيمة النهائية أو البدائية تعبيراً قيمته صفر.

```
For I = 15 To 0 Step -0.1
    y = x + I
Next I
```
الحلقة صحيحة

```
For I = 0 To 6.5
    y = x + I
Next I
```
الحلقة صحيحة

- إذا كانت القيمة الابتدائية والقيمة النهائية متساويتان، فيتم تنفيذ الحلقة مرة واحدة فقط.

```
For I = 7 To 7 Step -0.1
    y = x + I
Next I
```
الحلقة صحيحة

```
For I = 5 To 5
    y = x + I
Next I
```
الحلقة صحيحة

- يمكن وضع حلقات **For – Next** متداخلة مع بعضها البعض (وهي مهمة جداً في المصفوفات)، أي يمكن وضع حلقة **For** ضمن حلقة ثانية.
- لكي يتم التنفيذ بشكل صحيح يجب استخدام أسماء متحولات مختلفة لعدادات (متحولات) الحلقة كما يجب الانتباه إلى عدم التقاطع في الحلقات.

(( ))

```
For I = 1 To 3
    For J = 2 To 4
        List1.AddItem I * J
    Next I
Next J
```
الشكل الخاطئ للحلقات المتقاطعة

(( ))

```
For I = 1 To 3
    For J = 2 To 4
        List1.AddItem I * J
    Next J
Next I
```
الشكل الصحيح للحلقات المتقاطعة

- يمكن استخدام كلمة **Next** دون كتابة الدليل إلا في الحلقات المتداخلة.

```
For I = 5 To 5
    y = x + I
Next
```
الحلقة صحيحة

```
For I = 5 To 5
    y = x + I
Next I
```
الحلقة صحيحة

- يمكن استخدام كلمة **Next** دون كتابة الدليل إلا في الحلقات المتداخلة.

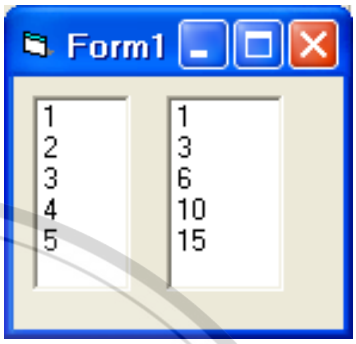
```
For I = 1 To 3
  For J = 2 To 4
    List1.AddItem I * J
  Next
Next
```
الشكل الخاطئ للحلقات المتقاطعة

```
For I = 1 To 3
  For J = 2 To 4
    List1.AddItem I * J
  Next J
Next I
```
الشكل الصحيح للحلقات المتقاطعة

أمثلة على أشكال الحلقة For – Next:

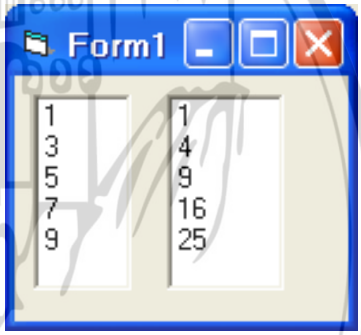
١) جمع الأعداد في المجال من 1 وحتى 5 (إن عدم ذكر الخطوة يعني أنّها مساوية للواحد).

```
S = 0
For x = 1 To 5
  S = X + S
  List1.AddItem X
  List2.AddItem S
Next X
```



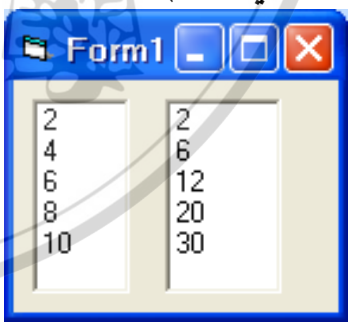
٢) جمع الأعداد الفردية في المجال من 1 وحتى 10 (يجب ذكر الخطوة والبدء من العدد الفردي الأول).

```
S = 0
For x = 1 To 10 Step 2
  S = X + S
  List1.AddItem X
  List2.AddItem S
Next X
```



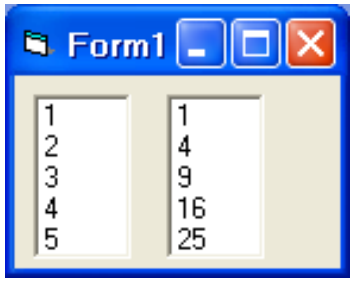
٣) جمع الأعداد الزوجية في المجال من 1 وحتى 10 (يجب ذكر الخطوة والبدء من العدد الزوجي الأول).

```
S = 0
For x = 2 To 10 Step 2
  S = X + S
  List1.AddItem X
  List2.AddItem S
Next X
```



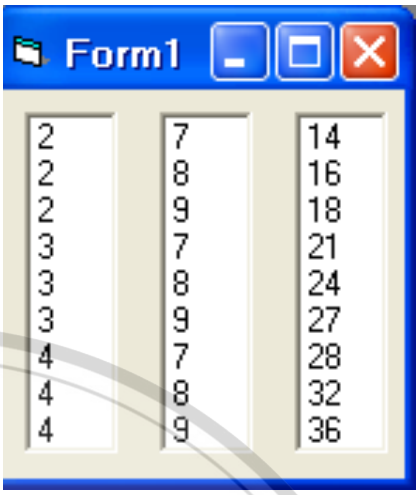
٤) مثال العدد ومربعه:

```
S = 0
For x = 1 To 5
  S = X + S
  List1.AddItem X
  List2.AddItem x ^ 2
Next X
```



٥) جدول الضرب الحلقات المتداخلة:

```
For A = 2 To 4
  For B = 7 To 9
    List1.AddItem A
    List2.AddItem B
    List3.AddItem A * B
  Next B
Next A
```



العبارة Exit For:

نستطيع الخروج من الحلقة $For - Next$ باستخدام العبارة $Exit\ For$.

مثلاً عند جمع الأعداد من 1 وحتى 10 فمن الواضح أنه إذا وصلنا إلى الرقم 7 أي إذا كان العدد 7 من بين الأعداد المطلوب جمعها فإن الحلقة For سيتم إنهاؤها ولذلك ستقوم العبارة $Exit\ For$ بإنهاء الحلقة قبل أن تصل الحلقة إلى نهايتها.

```
For I = 1 To 10
  Y = I + X
  If I = 7 Then
    Exit For
  End If
Next I
```

٢) تعليمة التكرار Do - Loop:

لها شكلان و لكل منها صيغتان:

a) الشكل الأول:

يكتب شرط الحلقة في البداية هنا يتم اختيار شرط الحلقة في البداية قبل تنفيذ تعليماتها لذلك قد لا تنفذ التعليمات على الإطلاق إن لم يتحقق الشرط من المرة الأولى.

١. الصيغة الأولى:

يتم تنفيذ الحلقة ما دام الشرط محقق.

```
Do While expression
    Statement – Block
Loop
```

٢. الصيغة الثانية:

يتم تنفيذ الحلقة حتى يتحقق الشرط.

```
Do Until expression
    Statement – Block
Loop
```

b) الشكل الثاني:

يكتب شرط الحلقة في النهاية لذلك يتم إخبار شرط الحلقة بعد تنفيذ مرة واحدة و إذا لم يتحقق شرط الحلقة نضمن تنفيذ الحلقة مرة واحدة على الأقل.

١. **الصيغة الأولى:**

يتم تنفيذ الحلقة ما دام الشرط محقق.

```
Do
    Statement − Block
Loop While expression
```

٢. **الصيغة الثانية:**

يتم تنفيذ الحلقة حتى يتحقق الشرط.

```
Do
    Statement − Block
Loop Until expression
```

مثال: أكتب برنامجاً يقوم بضرب مجال من الأعداد بمجال آخر.

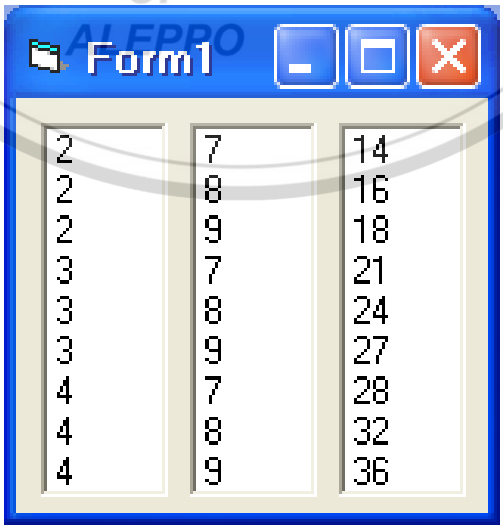
```
A = Val (InputBox("Insert A"))
Do Until A = 5
  For B = 7 To 9
    List1.AddItem A
    List2.AddItem B
    List3.AddItem A * B
  Next
  A = A + 1
Loop
```

برنامج يستخدم الصيغة الثانية (Until)

```
A = Val (InputBox("Insert A"))
Do While A < 5
  For B = 7 To 9
    List1.AddItem A
    List2.AddItem B
    List3.AddItem A * B
  Next
  A = A + 1
Loop
```

برنامج يستخدم الصيغة الأولى (While)

الآن عند تنفيذ البرنامج بالصيغتين نحصل على نفس النتيجة. إذاً نلاحظ أنه عند تنفيذ البرنامج بحلقة $Do - Until$ أنّه لن ينفذ الشرط الأخير من البرنامج.



الحلقات اللامنتهية:

قد نقع في بعض الأحيان في خطأ ما يجعل البرنامج يقوم بتنفيذ الحلقة إلى ما لا نهاية. هذا يتم عندما لا تصل الحلقة إلى قيمة الشرط النهائية ولذلك تبقى الحلقة تعمل بدون أن تنهي عملها إلى الأبد مثل حالة تغيير قيمة العداد داخل حلقة ***For – Next*** فإن البرنامج قد لا ينهي عمله أبداً.

```
For I = 10 To 20
   Y = I * X
   I = I - 3
Next I
```

- يبدأ البرنامج عندما يدخل إلى الحلقة بإعطاء القيمة $I = 10$.
- يتم ضرب قيمة I بـ X وهذا الأمر مسموح (استخدام الدليل).
- ثم يقوم بتخفيض قيمة I بمقدار 3 (وهذا الأمر غير مسموح) لأنه يتم تغيير قيمة الدليل حيث تصبح قيمته هنا مساوية 7 وفي نهاية الحلقة يضيف الخطوة المساوية للواحد هنا فتصبح $I = 8$ ولذا فإن قيمة I لن تصبح أبداً مساوية لـ 20 والتي تعبرعن القيمة النهائية للحلقة ولذلك سيعمل البرنامج إلى أن نوقفه بشكل قسري.

```
A = 0
Do Until A = 9
   Y = A ^ 2
   A = A + 2
Loop
```

- يبدأ البرنامج عندما تكون قيمة $A = 0$ فيدخل إلى الحلقة وتتزايد قيمة A بمقدار 2.
- بما أن البداية 0 والتزايد زوجياً بمقدار 2 فإننا لن نصل أبد إلى قيمة لـ A فردية أي أن قيمة A لن تصبح مساوية لـ 9 على الإطلاق وبالتالي فإن الحلقة ستعمل بشكل دائم ولذلك سيعمل البرنامج إلى أن نوقفه بشكل قسري.

جدول الضرب باستخدام الحلقات المتداخلة

(حلقات التكرار For – Next):

```
10 X1 = Val (InputBox ("Insert X1"))
   X2 = Val (InputBox ("Insert X2"))
   If X1 > X2 Then
      MsgBox ("Error")
      GoTo 10
   End If
20 Y1 = Val (InputBox ("Insert Y1"))
```

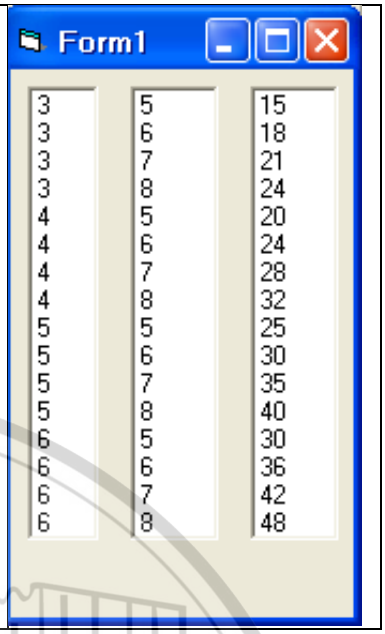
تم إدخال ما يلي:

$X1 = 3, X2 = 6$

$Y1 = 5, Y2 = 8$

يتم الوصول إلى القيم النهائية كما في الشكل:

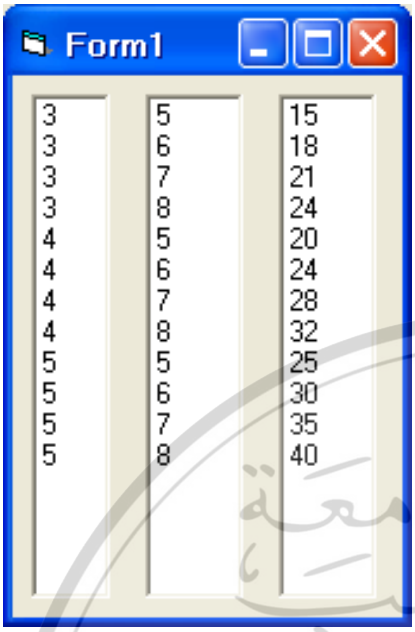
```
    Y2 = Val (InputBox ("Insert Y2"))
    If Y1 > Y2 Then
       MsgBox ("Error")
       GoTo 20
    End If
    For I = X1 To X2
      For J = Y1 To Y2
        Z = I * j
        List1.AddItem I
              : List2.AddItem J
        List3.AddItem Z
      Next J
    Next I
```



جدول الضرب باستخدام الحلقات المتداخلة (حلقات التكرار *Do - Loop*):

```
10  X1 = Val (InputBox ("Insert X1"))
    X2 = Val (InputBox ("Insert X2"))
    If X1 > X2 Then
       MsgBox ("Error")
      GoTo 10
    End If
20  Y1 = Val (InputBox ("Insert Y1"))
    Y2 = Val (InputBox ("Insert Y2"))
    If Y1 > Y2 Then
       MsgBox ("Error")
       GoTo 20
    End If
    I = X1
    Do Until I = X2
      J = Y1
      Do
        Z = I * J
       List1.AddItem I : List2.AddItem J
       List3.AddItem Z
        J = J + 1
      Loop While J < Y2 + 1
      I = I + 1
    Loop
```

كيفية تنفيذ كود البرنامج وتعليمات حلقة التكرار:



تم إدخال ما يلي:

$X1 = 3, X2 = 6$
$Y1 = 5, Y2 = 8$

نلاحظ أنه في حلقة $\boldsymbol{Do - Loop}$ الخارجية لم يتم تنفيذ الشرط الأخير أي لم ينفذ حلقة $\boldsymbol{Do}$ عندما $\boldsymbol{I = 6}$.

في الحلقة الداخلية تم تنفيذ كل القيم كوننا استخدمنا الشرط حتى $J < Y2 + 1$ ولذا نفذ القيم حتى $J = 8$.

٣) تعليمة الحلقة While ... Wend:

الشكل العام للتعليمة هو:

```
While Condition
    VB Code
Wend
```

يتم تنفيذ الحلقة طالما الشرط محقق.

مثال:

```
A = InputBox("Insert A")
Text1.Text = A
If A >= 11 Then
Text2.Text = "Error"
GoTo 100
End If
Sum = 0
While A < 11
 Sum = Sum + A
 Text2.Text = Sum
 A = A + 1
Wend
100
```

```
A = InputBox("Insert A")
Text1.Text = A
Sum = 0
While A < 11
 Sum = Sum + A
 A = A + 1
Wend
If A > 11 Then
Text2.Text = "Error"
Else
Text2.Text = Sum
End If
```

يمكن إجراء مقارنة بين الصيغتين واستقراء أيهما أفضل والأكثر ملاءمة وذلك حسب شروط المسألة.

٤) تعليمة With ... End With:

تمكن هذه التعليمة من إسناد القيم إلى عدة خصائص لأداة ما دون تكرار كتابة اسم الأداة لكل خاصية.

مثال:

```
With Picture1
   .BackColor = RGB(255, 0, 0)
   .ForeColor = vbBlue
   .Width = 1000
   .Height = 2300
   Picture1.Line (0, 0) – (.Width, .Height)
End With
```

تعرف التعليمات السابقة أربعة خواص لأداة الصورة **Picture1** .

نلاحظ أنّ هذه التعليمة سمحت لنا باختصار زمن الإدخال في مرحلة البرمجة الكودية، فبدلاً من أن نكتب اسم الأداة مع كل خاصية يكفي كتابة اسم الأداة لمرة واحدة فقط على سطر التعليمة **With**.

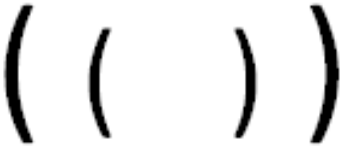
تسمح هذه التعليمة باختصار اسم الأداة عند التعبير عن خصائصها ولكن لا تسمح بذلك عند كتابة الأوامر والطرائق الأخرى كما هو مبيّن في طريقة رسم الخط **Line Method** المُبَيَّن في الكود:

Picture1.Line (0, 0) – (.Width, .Height)

ملاحظات عن الحلقات المتداخلة:

الحلقات المتداخلة هي حلقات تشترك بنفس الأوامر وتكون إحداها داخل الأخرى ومن شروطها:

١. أن تكون أدلة الحلقات مختلفة.
٢. الحلقة التي تبدأ أولاً تنتهي آخراً.
٣. الحلقة التي تبدأ آخراً تنتهي أولاً.



الشكل الصحيح للحلقات المتداخلة

الشكل الخاطئ للحلقات المتداخلة

كيفية تنفيذ تعليمات الحلقات المتداخلة:

يتم تنفيذ تعليمات الحلقات المتداخلة وفق الآلية التالية:

١. عند الدخول إلى الحلقة الأولى يأخذ دليل الحلقة الأولى أول قيمة ويقف ثم يتم الدخول إلى الحلقة الثانية فيأخذ دليل الحلقة الثانية كل القيم ويخرج منها.

٢. يأخذ بعد ذلك دليل الحلقة الأولى ثاني قيمة ويقف ثم يتم الدخول إلى الحلقة الثانية فيأخذ دليل الحلقة الثانية كل القيم ويخرج منها.

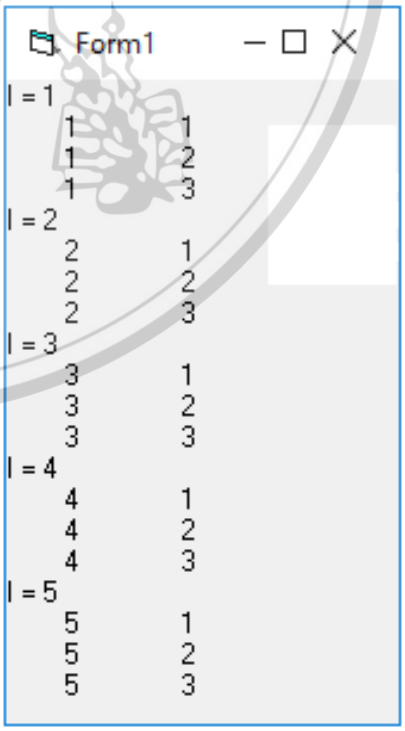
٣. نســـتمر بذلك حتى يأخذ دليل الحلقة الأولى آخر قيمة ويقف ثم يتم الدخول إلى الحلقة الثانية فيأخذ دليل الحلقة الثانية كل القيم ويخرج منها.

٤. يتم الخروج من الحلقة الأولى وهكذا تنتهي الحلقتين معاً.

الحلقات المتداخلة { بداية الأولى؛ { بداية الثانية؛ نهاية الثانية }؛ نهاية الأولى }	***Nested Loops*** { ***Start First Loop***; { ***Start Second Loop***; ***End Second Loop*** }; ***End First Loop*** }

مثال:

```
Private Sub Command1_Click()
  For I = 1 To 5
    Print "I = "; I
    For J = 1 To 3
      Print "     "; I,J
    Next J
  Next I
End Sub
```



جامعة
حلب
١٩٥٨
UNIVERSITY
OF
ALEPPO

الفصل السادس
المصفوفات
Arrays

تعريف المصفوفة:

هي مجموعة من المتغيرات من نفس النمط ترتبط مع بعضها وتسمى بنفس الاسم. أو هي متحول يأخذ نفس الاسم لكن بعدة قيم كل قيمة تخزن تحت اسم المتحول نفسه ولكن بدليل مختلف ويجب أن يوضع الدليل بين قوسين. يسمى كل متغير منها عنصر أو Element ولكل متغير داخل المصفوفة دليل Index مختلف يحدد ترتيبه داخل المصفوفة.

ويتم تعريف وحجز المصفوفة كالتالي:

$$Dim\ M(50) As\ Integer$$

- الاسم: M.
- الأبعاد: (50).
- النوع: Integer.

عند استدعاء أي متغير فيها يتم الإشارة إليه باستخدام الدليل (index) أو الرقم المسلسل الخاص به داخل المصفوفة. ويمكن العودة إلى أي عنصر من عناصر المصفوفة بواسطة اسم المصفوفة ودليل العنصر فمثلاً M(1) يدل على العنصر الأول أما M(39) فيدل على العنصر 39.

الإعلان عن المصفوفة:

يسمى الإعلان عن المصفوفة أيضًا تحديد أبعاد المصفوفة وهو يشبه الإعلان عن المتغير العادي ويكون على الصيغة التالية:

$$Public\ M(99)$$

يتم الإعلان عن المصفوفة بإحدى العبارات Dim أو Public أو Private أوStatic:

$$Private\ M1(99)$$
$$Public\ M2(99)$$
$$Dim\ M3(99)$$

- Static: تستخدم للتصريح المحلي وتكتب ضمن إجراء حدثي. ولذا يمكن استخدامها ضمن الإجراء الذي تم التصريح عنها فيه فقط.
- Private: تجعل المصفوفة محلية أي إذا تم الإعلان عنها في إجراء ما فإنها لن ترى إلا في هذا الإجراء. وإن تم الإعلان عنها في قسم الإعلان العام في نموذج أو وحدة برمجية فإنها لن ترى إلا في هذا النموذج أو هذه الوحدة.
- Public: لا تستخدم إلا في قسم الإعلان العام وتعنى أن المتغير يصبح متاحاً في أي مكان من البرنامج.
- Dim: تشبه Private عند استخدامها في إجراء أو في قسم الإعلان العام في نموذج بينما تقوم بعمل Public عند استخدامها في قسم الإعلان العام في وحدة برمجية.

هناك صـــيغة أخرى للإعلان عن المصـــفوفة تتيح حرية أكبر في تحديد الحد الأدنى والأقصى للدليل وهي كالتالي:

$$Dim\ M(2\ To\ 99)$$

بشكل عام عند التصريح (الإعلان) عن مصفوفة يجب أن تشتمل المعلومات على:

- اسم المصفوفة: الاسم الذي ستستعمله المصفوفة في البرنامج.
- نوع البيانات: نوع البيانات التي ستخزنها المصفوفة.
- عدد الأبعاد الذي ستحتوي عليه المصفوفة. قد تكون أحادية البعد أو ثنائية وهكذا.
- عدد العناصر: عدد العناصر الذي ستحتويه المصفوفة. يستنتج عدد العناصر مباشرة من صيغتها وبشكل افتراضي تحمل أول خانة الفهرس 0 (صفر).

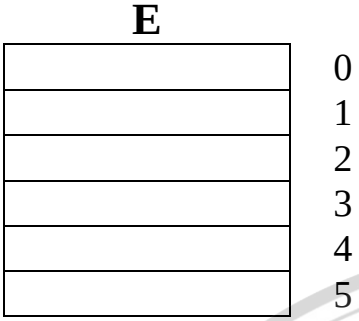
وبشكل عام نعرف مصفوفة كما يلي:

$$Public\ ArrayName\ (Dim1Elements, Dim2Elements, \dots)\ As\ DataType$$

- Public: الكلمة الدليلية التي تدل على مصفوفة عامة.
- ArrayName: اسم المصفوفة.
- Dim1Elements: عدد العناصر في البعد الأول في المصفوفة.
- Dim2Elements: عدد العناصر في البعد الثاني في المصفوفة.
- DataType: الكلمة الدليلية التي تحدد نوع البيانات التي ستخزن في المصفوفة.

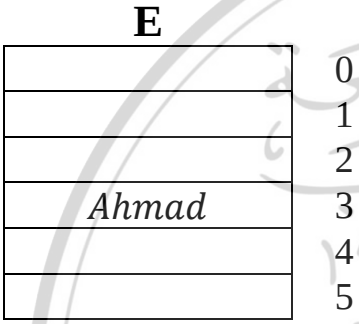
مثال:

$Dim\ E(5) As\ String$



هـذه مصـفوفة عموميـة (Public) اسـمها E تتسـع لسـتة عناصـر مـن النـوع النصـي (String).

يتم ترقيم العناصر من 0 وحتى 5



عنـد الإشـارة إلـى أي عنصـر مـن المصـفوفة كمـا في العبارة:

$E(3) = "Ahmad"$

عندها سيظهر هذا العنصر كما يلي:

النهايتان العليا والسفلى للمصفوفة:

يتحدد العنصران الأول والأخير من المصفوفة من عبارة التصريح عن المصفوفة نفسها:

مثال:

$Dim\ M(0\ To\ 35) As\ Long$

هذه العبارة تصرح عن مصفوفة من الأرقام من النوع الطويل.

- العنصر الأول أو الحد الأول LBound (النهاية السفلى): هو M(0).
- العنصر الأخير العنصر أو الحد الأخير UBound (النهاية العليا): هو M(35).

كما يمكن التصريح باختصار كما يلي:

$Dim\ M(35) As\ Long$

يفسـر هذا التصـريح كما يلي: العنصـر الأول من المصـفوفة هو M(0) والعنصـر الثاني هو M(1) والثالث هو M(2) والعنصر الأخير هو M(35).

عيب هذا التصريح (الشكل المختصر) هو أن القيمة الافتراضية للنهاية السفلى هو الصفر دوماً أي ينبغي أن نتذكر أن العنصـر الأول هو ذي الترتيب صـفر والعنصـر الثاني ذي الترتيب 1 ... وهكذا.

استخدام عبارة اختيار الأساس Option Base:

عادةً يبدأ دليل المصفوفة بالقيمة صفر وإن حجز المصفوفات يتم عادة من الصفر أي أن المصفوفة:

$Dim\ M(50)$

تحتوي 51 عنصراً وليس 50. وإذا وجدت المصفوفة بهذا الشكل فمن الأفضل أن يبدأ الدليل من 1 بدلاً من صفر لذلك يتم استخدام العبارة Option Base في قسم الإعلان العام للوحدة البرمجية:

$Option\ Base\ 1$

إن العبارة Option Base هي التي تحدد الحد الأصغري من كلا البعدين (السطر والعمود) وتأخذ إحدى القيمتين إما 0 أو 1.

يمكن التعامل مع الدليل كرمز متغير وليس كرقم: فمثلاً Marks(i) يدل على العنصر i.

المصفوفة متغيرة الحجم (الديناميكية):

عند التصريح عن مصفوفة بالعبارة:

$Static\ M(10000)As\ Long$

كل عنصر من عناصر هذه المصفوفة من النوع الطويل Long والذي سيشغل 4 Byte ولذلك فإن هذه المصفوفة ستشغل ما قيمته 40004 Byte من الذاكرة. إن هذه القيمة ليست قليلة ولكن لو تخيلنا وجود عشر مصفوفات مثل هذه المصفوفة عندها سيتم استهلاك 400040 Byte من ذاكرة الحاسب وهذا ما سيؤدي إلى عمل الحاسب بشكل بطيء.

وبشكل عام ذكرنا سابقاً أنه يمكن تعريف المصفوفة كما يلي:

$Private\ M1(99)$
$Public\ M2(99)$
$Dim\ M3(99)$

أثناء تعريف المصفوفة بهذه الطريقة فإننا سنقوم بحجز 99 مكاناً في الذاكرة لتعبئة هذه العناصر وهذا غير مفيد عندما لن نحتاج أو لن نستخدم كل هذه العناصر وذلك كوننا سنهتلك جزء كبير من الذاكرة بدون فائدة. إذاً يفضل تعريف المصفوفة بشكل آخر ولذلك لجأنا إلى المصفوفة الديناميكية.

والمصفوفة الديناميكية Array Dynamic:

هي مصفوفة وهمية غير محددة الأبعاد (العناصر) تساعد على الاستخدام الأمثل للذاكرة عن طريق الحجز اللاحق للأبعاد المناسبة تماماً للمصفوفة.

ففي المثال:

$$Dim\ A(100)As\ Integer$$

هنا ليس بالضرورة أن يكون عدد العناصر المدخلة 100 وبالتالي سيتم حجز مكان لمئة عنصر في الذاكرة.

أما حين نكتب:

$$Dim\ A(\ \ )As\ Integer$$

فإن البرنامج يقوم بحجز مصفوفة غير محددة الأبعاد أو العناصر وهذا ما ندعوه التعريف الديناميكي للمصفوفات Array Dynamic Allocation. إذاً يجب الإعلان عن هذه المصفوفات بدون تحديد لعدد العناصر وإلا لن يتم السماح بتغير عددها فيما بعد.

من الواضح أن هذه المصفوفة لا تحتوي على أية عناصر وأي استخدام لها سيسبّب ظهور خطأ وحين معرفة بعد هذه المصفوفة الافتراضية أثناء عمل البرنامج نقوم بإعادة حجز المصفوفة بالتعليمة التالية:

$$ReDim\ A(n)$$

- نبدأ العبارة بكلمة ReDim وهي عبارة برمجية وليست إعلاناً أي يجب استخدامها داخل إجراء ولا يمكن استخدامها في قسم الإعلان العام أو قسم التصاريح العامة باعتبار أنه يجب تحديد أبعادها قبل الاستخدام بعبارة ReDim وبالتالي نستطيع حجز وإعادة حجز أي مصفوفة ديناميكية أكثر من مرة في العمل البرمجي الواحد.
- نلاحظ أن عدد عناصر المصفوفة عبارة عن قيمة ثابتة ضمن العمل (نستطيع تغييرها عن طريق تغيير قيمة المتغير n فقط) لذلك يجب تعريف قيمة المتغير ضمن المصفوفة كرقم ثابت.
- كما نلاحظ أنه عند إعادة حجز المصفوفة الديناميكية لم نعرّف نوعها لأن البرنامج يفترض أنها من نفس النوع الذي تم تعريف المصفوفة الديناميكية به من قبل، وفي حالة إعادة تعريفها أيضاً ولكن بشكل مغاير سيُظهر البرنامج وجود خطأ.

نستطيع مباشرة حجز المصفوفة الديناميكية كالتالي:

$ReDim\ C(I)As\ Integer$

وبشكل عام يمكن أن نكتب:

$ReDim\ Preserve\ M(\ 1\ To\ 99)$

- كلمة Preserve اختيارية وتعنى الاحتفاظ بالعناصر الموجودة حاليًا في المصفوفة نظرًا لأن عبارة ReDim يمكن استخدامها أكثر من مرة مع نفس المصفوفة فقد يتم استخدامها والمصفوفة تحتوي على عناصر بالفعل.
- من الواضح أنه لو لم يتم وضع Preserve فإن العناصر كلها سيتم وضعها بالقيمة صفر أما إذا تم استخدام Preserve فإن القيم الحالية سيتم الاحتفاظ بها.

حجم المصفوفة:

إن الحجم الأعظمي الذي يمكن أن تصله المصفوفة متعلق بنظام التشغيل وبإمكانيات الحاسب نفسه. يبدأ البرنامج باستخدام الذاكرة RAM وعندما تُستخدم كل الذاكرة RAM المتاحة في الحاسب سيبدأ النظام باستخدام القرص الصلب وكأنه امتداد للذاكرة RAM وعندها سيُدعى الجزء من القرص الصلب والمستخدم كذاكرة RAM بالذاكرة الوهمية Virtual Memory. كما أن استخدام مصفوفات كبيرة والتي تستهلك الذاكرة المتاحة سيؤدي إلى تدني سرعة عمل البرنامج وسيعاني البرنامج من مشاكل في الأداء عند استهلاك كامل الذاكرة المتاحة RAM.

المصفوفة متعددة الأبعاد:

نلاحظ أن المصفوفة السابقة أحادية البعد، ويمكن أن تكون ثنائية أو ثلاثية ... وهكذا. يتيح لنا Visual Basic أن تقوم بتعريف مصفوفات يصل حجمها حتى 60 بعداً.

المصفوفة ذات بعدين تشبه الجدول تمامًا إذ يمكن الوصول إلى أي عنصر فيها باستخدام دليلين وليس دليلاً واحدًا، الدليل الأول هو رقم الصف والثاني هو رقم العمود.

تسمى المصفوفة ذات m سطراً و n عموداً بالمصفوفة m x n أو m by n.

يرمز للمصفوفة ذات البعدين بالرمز $U(i,j)$ ، حيث U اسم المصفوفة و i رقم يدل على عدد أسطر المصفوفة أو بعدها الأول Element1 و j رقم يدل على عدد أعمدة المصفوفة أو بعدها الثاني Element2.

وكما ذكرنا سابقاً نرى أنّه بالحالة الافتراضية تأخذ العبارة Option Base القيمة 0 لذلك إذا أردنا التعبير عن مصفوفة بثلاثة أسطر وأربعة أعمدة فمعنى ذلك أنه يجب أن يكون الرقم الدال إلى الأسطر يساوي [العدد المطلوب (مطلوب 3 صفوف) ناقص واحد أي 2]، كذلك يجب أن يكون الرقم الدال إلى الأعمدة يساوي [العدد المطلوب (مطلوب أربع أعمدة) ناقص واحد أي 3] أي نكتب المصفوفة بالشكل U(2, 3) أما إذا كتبنا مصفوفة بالشكل U(3, 4) فمعنى ذلك أن عدد الصفوف أو الأسطر هنا هو 4 أربعة وعدد الأعمدة هو 5 خمسة. ويمكن تمثيل المصفوفة كما يلي:

ArrayM	Column0	Column1	Column2	Column3	Column4
Row 0	M (0 , 0)	M (0 , 1)	M (0 , 2)	M (0 , 3)	M (0 , 4)
Row 1	M (1 , 0)	M (1 , 1)	M (1 , 2)	M (1 , 3)	M (1 , 4)
Row 2	M (2 , 0)	M (2 , 1)	M (2 , 2)	M (2 , 3)	M (2 , 4)
Row 3	M (3 , 0)	M (3 , 1)	M (3 , 2)	M (3 , 3)	M (3 , 4)

كما يمكن للإعلان عن مصفوفة ذات بعدين نستخدم الصيغة التالية:

$$Dim\ M(2,3)\ As\ Integer$$

وستكون بنود هذه المصفوفة كما يلي:

Array M	Column 0	Column 1	Column 2	Column 3
Row 0	M (0 , 0)	M (0 , 1)	M (0 , 2)	M (0 , 3)
Row 1	M (1 , 0)	M (1 , 1)	M (1 , 2)	M (1 , 3)
Row 2	M (2 , 0)	M (2 , 1)	M (2 , 2)	M (2 , 3)

يمكن الإعلان عن مصفوفة ذات بعدين نستخدم الصيغة التالية:

$$Dim\ M(0\ To\ 3, 1\ To\ 4)$$

وستكون بنود هذه المصفوفة كما يلي:

Array M	Column 0	Column 1	Column 2	Column 3
Row 0	M (0 , 1)	M (0 , 2)	M (0 , 3)	M (0 , 4)
Row 1	M (1 , 1)	M (1 , 2)	M (1 , 3)	M (1 , 4)
Row 2	M (2 , 1)	M (2 , 2)	M (2 , 3)	M (2 , 4)
Row 3	M (3 , 1)	M (3 , 2)	M (3 , 3)	M (3 , 4)

عند التصريح عن مصفوفة ثنائية الأبعاد سيحجز لها VB مكاناً في الذاكرة كما يلي:

$$Public\ Z(1,4)\ As\ Variant$$

هي مصفوفة عامة ($Public$) اسمها Z لها بعدين:

- البعد الأول عدد عناصرها 2 (من 0 وحتى 1).
- البعد الثاني عدد عناصره 5 (من 0 وحتى 4).
- العناصر من النوع المتغير (***Variant***).

Z

$Colomns$

4	3	2	1	0	
					0 $Rows$
					1

بشـــكل مماثل يمكن أن نكتب القيمة أو نخزن القيمة في المكان الموجود في الصـــف 0 والعمود 2 كما يلي: $Z\ (0,2)\ =\ 23$.

Z

$Colomns$

4	3	2	1	0	
		23			0 $Rows$
					1

وللإعلان عن مصفوفة ذات ثلاثة أبعاد نستخدم الصيغة التالية:

$Dim\ M(0\ To\ 1\ ,1\ To\ 2\ ,5\ To\ 6)\ As\ Integer$

وستكون عناصر هذه المصفوفة كما يلي:

$M(0\ ,1\ ,5)$	$M(0\ ,1\ ,6)$
$M(0\ ,2\ ,5)$	$M(0\ ,2\ ,6)$
$M(1\ ,1\ ,5)$	$M(1\ ,1\ ,6)$
$M(1\ ,2\ ,5)$	$M(1\ ,2\ ,6)$

المصفوفات المتعددة الأبعاد تشبه أحادية البعد من حيث الاستخدام إذا ينطبق عليها عبارة $Option\ Base$.

كذلك ينطبق على هذا النوع اســـتخدام العبارة $ReDim$ إذ يمكن تغيير الأبعاد كأي مصفوفة أحادية البعد ولكن لو استخدمت $Preserve$ مع المصفوفة متعددة الأبعاد فإن البعد الأخير فقط هو الذي يمكن تغيير عدد عناصره فقط.

كما نستطيع أيضاً حجز مصفوفة ديناميكية ذات بعدين أو أكثر كما يلي:

```
Dim C(  ) As Long
I = ⋯
J = ⋯
ReDim C(I ,J)
```

يمكن احياناً أن يستخدم عداد الحلقة For Next في تحديد أبعاد وعناصر مصفوفة:

```
Dim I
Dim M(  ) As Integer
ReDim M(1 To 15) As Integer
For I = 1 To 15 Step 1
M(1 ) = I
Next I
ReDim M(1 To 4) As Integer
```

إن التصريح M( ) يدل على أن المصفوفة M( ) هي مصفوفة ديناميكية. والعبارة:

$ReDim\ M(1\ To\ 15)\ As\ Integer$

تعين حجم المصفوفة باستخدام $ReDim$. تحدد هذه العبارة حجم المصفوفة بـــ 15 عنصر وكل عنصر من عناصرها ينتمي إلى النوع الصحيح.

تملأ الحلقة For بنود هذه المصفوفة الـ 15 ثم تستخدم العبارة $ReDim$ مرة ثانية لحجز 4 عناصر.

إن الحجم الذي أخذته المصفوفة الأولى من الذاكرة هو $(2\ x\ 15\ =\ 30\ Byte)$ لأنها حجزت خمسة عشر عنصراً وبعد تنفيذ العبارة الثانية يتبدل حجم المصفوفة M( ) إلى أربع عناصر وبالتالي ستشغل من الذاكرة حجماً أقل من الأولى والذي مقداره 8 = 2x4) (Byte. وهكذا نستخدم هذه الطريقة للمحافظة على أكبر كمية غير مستخدمة من الذاكرة.

عملية البحث في المصفوفات Searching Arrays:

يمكن أن نتعامل أحياناً مع كم كبير جداً من المعلومات والتي نرتّبها أو نخزنها في مصفوفات رقمية أو اسمية أو من أي نوع، وقد نضطر في كثير من الأحيان التعامل مع عنصر ما من العناصر المخزَّنة لذا نقوم بالبحث عنه بين العناصر التي تمّ تخزينها.

إن عملية البحث عن عنصر ما في مصفوفة معيّنة قد تكون خطيّة أو ثنائية وهذا مُتعلِّق بعدد عناصر المصفوفة من جهة وبكون هذه العناصر مُرَتَّبة أو لا من جهة أخرى.

البحث الخطّي Linear Search:

تستخدم عملية البحث هذه في المصفوفات الصغيرة والمصفوفات غير المرتَّبة. حيث تتم عملية البحث عن عنصر ما في هذه الطريقة بأن نقوم بمقارنة عنصر البحث مع كل عنصر من عناصر المصفوفة ابتداءً من العنصر الأول وانتهاءً بالعنصر الأخير. وعلى اعتبار أن المصفوفة غير مُرتبة فقد يكون العنصر في المصفوفة هو العنصر المطلوب وقد يكون العنصر الأخير وكحالة وسطية نعتبر أنّه سيتم بحث نصف عدد العناصر الموجودة. ومن الواضح أنّه سيتم مقارنة عناصر المصفوفة كاملةً إن لم يكون عنصر البحث موجوداً فيها حتى يتم التأكّد من ذلك.

من الواضح أنّ هذه الطريقة ملائمة في المصفوفات الصغيرة وفي المصفوفات غير المرتَّبة، وهي غير فعَّالة في المصفوفات الكبيرة لأن عملية البحث ستبدأ من أول عنصر وتنتهي بآخر عنصر وهذا سيحتاج إلى جهدٍ وزمنٍ كبيرين.

البرمجة الثنائية Binary Search:

تستخدم هذه الطريقة بشكل رائع في المصفوفات الكبيرة والمُرتبة ترتيباً تصاعديّاً أو تنازليّاً. وتكون آلية البحث عن عنصر معين من خلال مقارنة عنصر البحث مع العنصر الوسط في المصفوفة وسيكون لدينا نتيجة المقارنة ثلاثة احتمالات منطقية وهي:

١) أن يكون هو العنصر المطلوب البحث عنه وتنتهي عملية البحث.
٢) أن يكون أكبر من عنصر البحث لذا يتم إهمال كل العناصر الأكبر من عنصر البحث ابتداءً من هذا العنصر وحتى آخر عناصر المصفوفة ويتم البحث في الجزء الآخر.
٣) أن يكون أصغر من عنصر البحث لذا يتم إهمال كل العناصر الأصغر من عنصر البحث ابتداءً من هذا العنصر وحتى آخر عناصر المصفوفة ويتم البحث في الجزء الآخر.

بكل الأحوال في الحالتين الثانية والثالثة فإن عملية المقارنة ستزيل نصف عدد العناصر مباشرةً من المرّة الأولى وسيذهب البحث إلى النصف الآخر المُتَبَقِّي من العناصر. وفي

عملية المقارنة الثانية سيتم إزالة نصف عدد العناصر من الجديد أي يبقى الربع وفي عملية المقارنة الثالثة سيتم إزالة نصف عدد العناصر المتبقي ويبقى الثُمن ... وهكذا.

الآن تستمر عملية البحث حتى يصبح الجزء المتبقي من المصفوفة ثلاثة عناصر وفي هذه الحالة ستتم مقارنة العنصر الوسط فإمّا أن يكون مساوٍ لعنصر البحث وتنتهي العملية أو يكون أكبر وبالتالي العنصر الأصغر يمكن أن يكون هو العنصر المطلوب وتنتهي العملية أو يكون أصغر وبالتالي العنصر الأكبر قد يكون هو العنصر المطلوب وتنتهي العملية.

عند استمرار عملية البحث حتى يصبح الجزء المتبقي من المصفوفة عنصراً واحداً، فمن الواضح أن هذا العنصر سيكون حكماً مخالفاً لعنصر البحث وعندها سنصل إلى نتيجة أن عنصر البحث غير موجود.

من الواضح هنا أن عملية البحث تتم بسرعة كبيرة جداً مقارنةً مع عملية البحث الخطّي.

مثال:

لدينا المصفوفة M(n) المؤلفة من 20 عنصراً.

- ما هو دليل العنصر السادس في هذه المصفوفة ومن هو العنصر M(12).
- اشرح بالتفصيل طرق البحث عن عنصر ما في مصفوفة ما، ابحث كمثال عن العنصر M(16) في هذه المصفوفة.

الحل:

يتم الإعلان عن المصفوفة بإحدى العبارات Dim أو Public أو Private أوStatic.

$Dim\ M(20)$

يتحدد العنصران الأول والأخير من المصفوفة من العبارة Option Base والتي تستخدم في قسم التصاريح العامة. إن العبارة Option Base هي التي تحدد الحد الأصغري وتأخذ إحدى القيمتين إما 0 أو 1.

إن التصريح عن مصفوفة بدون استخدام هذه العبارة يعني أن العنصر الأول من المصفوفة هو M(0) والعنصر الثاني هو M(1) والثالث هو M(2) والعنصر الأخير هو M(20) ومن الواضح هنا أن عدد عناصر المصفوفة الكلي هو 21 عنصراً.

عيب هذا التصريح هو أن القيمة الافتراضية للنهاية السفلى هو الصفر دوماً أي ينبغي أن نتذكر أن العنصر الأول هو العنصر ذي الترتيب صفر والعنصر الثاني ذي الترتيب 1 وهكذا ...

إن التصريح عن مصفوفة باستخدام هذه العبارة (أي نكتب Option Base 1) يعني أن العنصر الأول من المصفوفة هو M(1) والعنصر الثاني هو M(2) والثالث هو M(3) والعنصر الأخير هو M(20) ومن الواضح هنا أن عدد عناصر المصفوفة الكلي هو 20 عنصراً.

ويمكن التصريح عن المصفوفة نفسها بأن يتحدد العنصران الأول والأخير من المصفوفة من عبارة التصريح عن المصفوفة نفسها كما في العبارة التالية:

$Dim\ M(10\ To\ 20)$

- العنصر أو الحد الأول LBound (النهاية السفلى) هو M(10).
- العنصر أو الحد الأخير UBound (النهاية العليا) هو M(20).

ما هو دليل العنصر السادس في هذه المصفوفة ومن هو العنصر M(12).

إن لم نُعَرِّف العبارة Option Base فمعنى ذلك أن مصفوفتنا المؤلفة من 20 عنصراً يجب أن تُعَرَّف كما يلي:

$Dim\ M(19)$

ويمكن تعريف المصفوفة كما يلي:

$Dim\ M(0\ To\ 19)$

في هذه الحالة سيكون العنصر الأول M(0) والعنصر الأخير M(19) والعنصر السادس هو M(5) ودليل هذا العنصر هو 5 أمّا العنصر M(12) فهو العنصر الثالث عشر.

ويمكن باستخدام العبارة Option Base 1 تعريف مصفوفتنا المؤلفة من 20 عنصراً كما يلي:

$Option\ Base\ 1$
$Dim\ M(20)$

ويمكن تعريف المصفوفة كما يلي:

$Dim\ M(1\ To\ 20)$

في هذه الحالة سيكون العنصر الأول M(1) والعنصر الأخير M(20) والعنصر السادس هو M(6) ودليل هذا العنصر هو 6 أمّا العنصر M(12) فهو العنصر الثاني عشر.

البحث عن العنصر M(16):

المصـــفوفة تحتوي على 20 عنصـــراً وعلى فرض أنّه أخذ العبارة Option Base فإن العنصــر الأول من المصــفوفة هو M(1) والعنصــر الأخير هو M(20) وأن العنصــر المطلوب البحث عنه موجود وهو العنصر M(16) وأن المصفوفة مرتبة ترتيباً تصاعديّاً.

من الواضح أن البحث الخطي سيقوم بفحص العناصر كلها أي سيقوم بعملية المقارنة 16 مرة حتى يتأكّد من أن عنصر البحث هو العنصر M(search) = M(16).

أمّا عملية البحث الثنائي فستتم كما يلي:

M(1)	M(2)	M(3)	M(4)	M(5)	M(6)	M(7)	M(8)	M(9)	M(10)
M(11)	M(12)	M(13)	M(14)	M(15)	M(16)	M(17)	M(18)	M(19)	M(20)

تتم عملية المقارنة مع العنصــر M(10) ونجد أن M(search) > M(10) لذا ســيتم إهمال عنصر المقارنة وكل العناصر الأصغر منه أي تبقى العناصر التالية:

M(11)	M(12)	M(13)	M(14)	M(15)	M(16)	M(17)	M(18)	M(19)	M(20)

تتم عملية المقارنة مع العنصــر M(15) ونجد أن M(search) > M(15) لذا ســيتم إهمال عنصر المقارنة وكل العناصر الأصغر منه أي تبقى العناصر التالية:

M(16)	M(17)	M(18)	M(19)	M(20)

تتم عملية المقارنة مع العنصــر M(18) ونجد أن M(search) < M(18) لذا ســيتم إهمال عنصر المقارنة وكل العناصر الأكبر منه أي تبقى العناصر التالية:

M(16)	M(17)

لنفترض أن المقارنة تمت مع العنصر M(16) فهذا يعني أن عملية البحث قد انتهت.

لنفترض أن المقارنة تمت مع العنصر M(17) فهذا يعني أن عملية ستُلغي هذا العنصر ويبقى العنصر M(16) فتتم المقارنة معه وتنتهي عملية البحث.

مسح محتويات مصفوفة:

الشكل العام: $Erase\ ArrayName$.

$ArrayName$ اسم المصفوفة التي نرغب بمسح قيمها باستخدام تعليمة $Erase$.

أمثلة عن المصفوفات واستخداماتها:

مثال: أكتب برنامجا يقوم بإدخال خمس قيم في مصـــفوفة ومن ثم يحســـب مجموع ومتوسـط هذه القيم ويطبعها. ثم أكتب برنامجا يقوم بإدخال N قيمة في مصـفوفة ومن ثم يحسب مجموع ومتوسط هذه القيم ويطبعها.

الحل: نكتب الكود أولاً من أجل خمس قيم.

```
Private Sub Command1_Click()
Dim Y(5) As Single
Dim Sum As Single, N As Single
Dim I As Integer
For I = 1 To 5
Y(I) = Val(InputBox("Y" & I))
Next I
Sum = 0
For I = 1 To 5
Sum = Sum + Y(I)
Next I
M = Sum / 5
Print "Sum = "; Sum, "Average = "; M
End Sub
```

ويمكن كتابة الكود بشكل عام لـ N قيمة.

```
Private Sub Command2_Click( )
Dim Y( ) As Single
Dim Sum As Single, N As Single
Dim I As Integer
N = Val(InputBox("Input Enter the size of Array"))
ReDim Y(N)
For I = 1 To N
Y(I) = Val(InputBox("Y" & I))
Next I
Sum = 0
For I = 1 To N
Sum = Sum + Y(I)
Next I
M = Sum / N
Print "Sum = "; Sum, "Average = "; M
End Sub
```

مثال: أكتب برنامجا يقوم بإدخال مجموعة من الأعداد داخل مصـــفوفة ثم يقوم بإيجاد العنصر الأصغر منها.

```
Private Sub Command3_Click( )
Dim Y( ) As Single
Dim Min As Single
Dim I As Integer
Dim N As Integer
N = Val(InputBox("Enter the Size of Array"))
ReDim Y(N)
For I = 1 To N
Y(I) = Val(InputBox("Y" & I))
Next I
Min = Y(1)
For I = 2 To N
If Min > Y(I) Then Min = Y(I)
Next I
Print "Minimum = "; Min
End Sub
```

ملاحظة: نفترض أن العنصر الأصغر هو أول عنصر من العينة ونقارنه مع عناصر العينة عنصراً وإذا وجدنا أصغر منه نستبدله به.

مثال: أكتب برنامجا يقوم بإدخال مجموعة من الأعداد داخل مصـــفوفة ثم يقوم بإيجاد العنصر الأكبر منها.

```
Private Sub Command4_Click( )
Dim Y( ) As Single
Dim Max  As Single
Dim I As Integer
Dim N As Integer
N = Val(InputBox("Enter the Size of Array"))
ReDim Y(N)
For I = 1 To N
Y(I) = Val(InputBox("Y" & I))
Next I
Max = Y(1)
For I = 2 To N
If Min > Y(I) Then Min = Y(I)
Next I
```

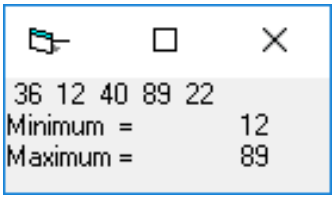
```
Print "Maximum = "; Max
End Sub
```

ملاحظة: نفترض أن العنصر الأكبر هو أول عنصر من العينة ونقارنه مع عناصر العينة عنصراً وإذا وجدنا أكبر منه نستبدله به.

مثال: أكتب برنامجاً يقوم بإدخال مجموعة من الأعداد داخل مصـــفوفة ثم يقوم بإيجاد العنصر الأكبر والأصغر منها.

```
Private Sub Command5_Click( )
Dim Y( ) As Single
Dim Min, Max As Single
Dim I, N As Integer
Dim N As Integer
N = Val(InputBox("Enter the Size of Array "))
ReDim Y(N)
For I = 1 To N
Y(I) = Val(InputBox("Y" & I))
Print Y(I); ;
Next I
Print
Max = Y(1)
Min = Y(1)
For I = 2 To N
If Max < Y(I) Then Max = Y(I)
If Min > Y(I) Then Min = Y(I)
Next I
Print "Minimum = "; Tab(20); Min
Print "Maximum = "; Tab(20); Max
End Sub
```

ملاحظة: نفترض أن العنصر الأصغر هو أول عنصر من العينة ونسميه Min ونقارنه مع عناصر العينة عنصراً وإذا وجدنا أصغر منه نستبدله به. وبنفس الوقت نفترض أن العنصر الأكبر هو أول عنصر من العينة ونسميه Max ونقارنه مع عناصر العينة عنصراً وإذا وجدنا أكبر منه نستبدله به. ويكون الناتج كما هو مبيَّن:



استخدام الحلقات المتداخلة في المصفوفات ثنائية البعد (متعددة الأبعاد):

تستخدم الحلقات المتداخلة لإدخال قيم المصفوفات ثنائية البعد، حيث يجب وضع دليلين يفصل بينهما فاصل.

يرمز للمصفوفة ثنائية البعد رياضياً بـ xij حيث إن i تمثل رقم السطر و j رقم العمود فإذا كان لدينا مصفوفة ثنائية البعد تبين علامات أربع مواد (لغة، رياضيات وبرمجة) لأربعة من الطلاب (خالد، عمر، حسن وعلي). عندها يمكن أن نرمز لأسماء المقررات بـ i ولأسماء الطلاب بـ j. وهكذا يمكن أن نرمز لعلامة الطالب الرابع (علي) بالمقرر الأول (لغة) بالرمز X_{14} ويمكن أن نرمز لعلامة الطالب الثاني (عمر) بالمقرر الثالث (برمجة) بالرمز X_{32}.

عند ذلك يمكن تمثيل وإدخال عناصر المصفوفة كما يلي:

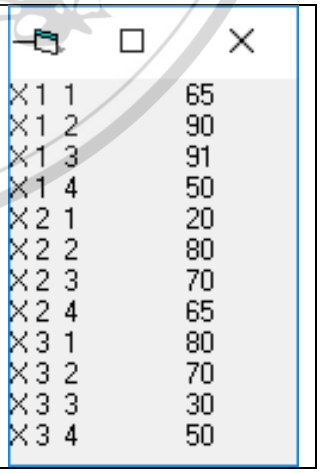
$$[X_{ij}] = \begin{bmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{21} & X_{22} & X_{23} & X_{24} \\ X_{31} & X_{32} & X_{33} & X_{34} \end{bmatrix}$$

وذلك وفق الجدول المبيّن:

	Khaled	***Omar***	***Hasan***	***Ali***
Language	$X_{11} = 65$	$X_{12} = 90$	$X_{13} = 91$	$X_{14} = 50$
Math	$X_{21} = 20$	$X_{22} = 80$	$X_{23} = 70$	$X_{24} = 65$
VB	$X_{31} = 80$	$X_{32} = 70$	$X_{33} = 30$	$X_{34} = 50$

عندها يمكن كتابة البرنامج كما يلي:

```
Private Sub Command2_Click( )
Dim X(3, 4) As Single
Dim I As Integer, J As Integer
For I = 1 To 3
For J = 1 To 4
X(I, J) = Val(InputBox("X" & I & J))
Print "X"; I; J, X(I, J)
Next J
Next I
End Sub
```



كما يمكن تمثيل وإدخال عناصر المصفوفة كما يلي:

$$[X_{ij}] = \begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \\ X_{41} & X_{42} & X_{43} \end{bmatrix}$$

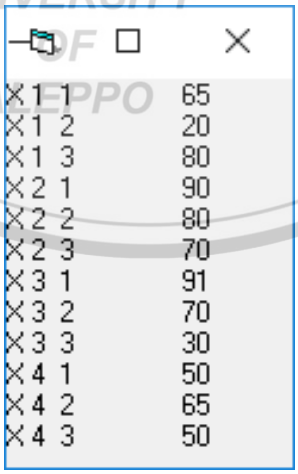
وذلك وفق الجدول المبيّن:

	Language	***Math***	***VB***
Khaled	$X_{11} = 65$	$X_{12} = 20$	$X_{13} = 80$
Omar	$X_{21} = 90$	$X_{22} = 80$	$X_{23} = 70$
Hasan	$X_{31} = 91$	$X_{32} = 70$	$X_{33} = 30$
Ali	$X_{41} = 50$	$X_{42} = 65$	$X_{43} = 50$

عندها يمكن كتابة البرنامج كما يلي:

```
Private Sub Command3_Click( )
Dim X(4, 3) As Single
Dim I As Integer, J As Integer
For I = 1 To 4
For J = 1 To 3
X(I,J) = Val(InputBox("X" & I & J))
Print "X"; I; J, X(I,J)
Next J
Next I
End Sub
```

ويكون الناتج كما هو مبيَّن:

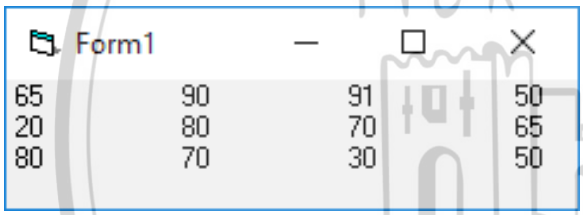


طبعاً هنا تم إظهار جميع النتائج في عمودين اثنين الأول يتضـــمن رمز المقرر والثاني الدرجة المُستحقة باستخدام الفواصل العادية والمنقوطة.

وإذا أردنا ألا تكون النتائج في عمود واحد وإنما في سطر واحد يكفي أن نضع فاصلة بعد أمر الطباعة لتطبع على ســـطر واحد وبفراغات متباعدة أو فاصـــلة منقوطة لتطبع على سطر واحد وبفراغات متقاربة. ويكون كود البرنامج كما يلي:

```
Private Sub Command4_Click( )
Dim X(3,4) As Single
Dim I As Integer,J As Integer
For I = 1 To 3
For J = 1 To 4
X(I,J) = Val(InputBox("X" & I & J))
Print X(I,J),
Next J
Print
Next I
End Sub
```

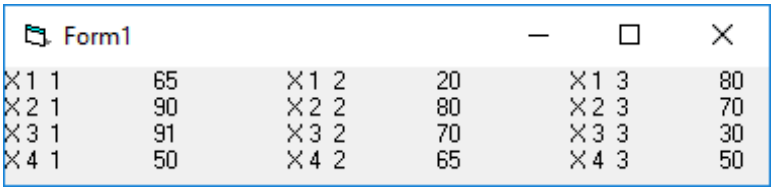
ويكون الناتج كما هو مبيَّن:



وإذا كتبنا كود البرنامج كما يلي:

```
Private Sub Command5_Click( )
Dim X(4,3) As Single
Dim I As Integer,J As Integer
For I = 1 To 4
For J = 1 To 3
X(I,J) = Val(InputBox("X" & I & J))
Print "X"; I; J,X(I,J),
Next J
Print
Next I
End Sub
```

عندها سيكون الناتج كما هو مبيَّن:



مثال: أكتب برنامجاً لإدخال قيم في مصفوفة مربعة ذات أبعاد 4x4 بحيث يقوم بما يلي:

١. حساب مجموع عناصر المصفوفة.

٢. حساب مجموع العناصر التي تقع فوق القطر الرئيسي.

٣. حساب مجموع العناصر التي تقع تحت القطر الرئيسي.

٤. حساب العناصر التي تقع على القطر الرئيسي وأيضا يطبع المصفوفة بشكل جدول.

الحل: نكتب كود البرنامج كما هو مبيّن:

```
Private Sub Command6_Click( )
Dim X(4, 4) As Single
Dim I As Integer, J As Integer
Dim Sum As Single, Su As Single
Dim Sd As Single, So As Single
Sum = 0: Su = 0: Sd = 0: So = 0
For I = 1 To 4: For J = 1 To 4
X(I, J) = Val (InputBox("x(" & I & "," & J & ")"))
Next J: Next I
For I = 1 To 4: For J = 1 To 4
Sum = Sum + X(I, J)
Next J, I
For I = 1 To 4: For J = 1 To 4
If I < J Then Su = Su + X(I, J)
Next J: Next I
For I = 1 To 4: For J = 1 To 4
If I > J Then Sd = Sd + X(I, J)
Next J, I
For I = 1 To 4: For J = 1 To 4
If I = J Then So = So + X(I, J)
Next J, I
Print " ----------"
Print "Sum = "; Sum, "Su = "; Su, "Sd = "; Sd, "So = "; So
Print " ----------"
For I = 1 To 4: For J = 1 To 4
Print X(I, J),
Next J
Print
Next I
End Sub
```

إن عناصر مصفوفة مربعة ذات أبعاد 4x4 تظهر كما يلي:

$$[X_{ij}] = \begin{bmatrix} \boldsymbol{X_{11}} & X_{12} & X_{13} & X_{14} \\ X_{21} & \boldsymbol{X_{22}} & X_{23} & X_{24} \\ X_{31} & X_{32} & \boldsymbol{X_{33}} & X_{34} \\ X_{41} & X_{42} & X_{43} & \boldsymbol{X_{44}} \end{bmatrix}$$

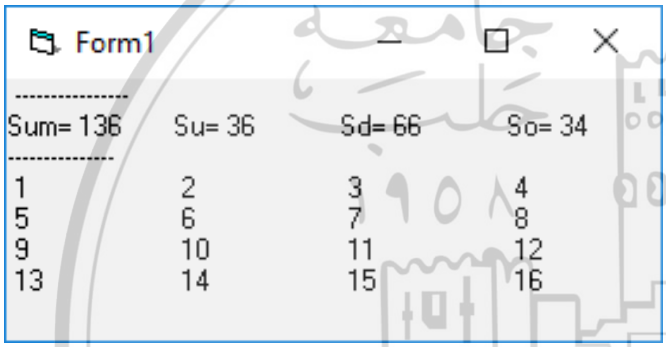
صفوف المصفوفة:

$$i = 1\ To\ m = 4$$

أعمدة المصفوفة:

$$j = 1\ To\ n = 4$$

عندها سيكون الناتج كما هو مبيَّن:



تحليل وتوضيح الناتج كما هو مبيَّن:

$$\boldsymbol{Sum} = All\ Numbers.$$

$$\boldsymbol{Sum} = \sum X_{ij}$$

$$\boldsymbol{Sum - Up} = \boldsymbol{Su} = The\ Numbers\ Up\ the\ Main\ Diameter.$$

$$\boldsymbol{Su} = \sum X_{ij} \quad if\ i < j$$

$$\boldsymbol{Sum - Down} = \boldsymbol{Sd} = The\ Numbers\ Down\ the\ Main\ Diameter.$$

$$\boldsymbol{Sd} = \sum X_{ij} \quad if\ i > j$$

$$\boldsymbol{Sum - On} = \boldsymbol{So} = The\ Numbers\ On\ the\ Main\ Diameter.$$

$$\boldsymbol{So} = \sum X_{ij} \quad if\ i = j$$

$$\boldsymbol{Sum} = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14 + 15 + 16 = \boldsymbol{136}$$

$$\boldsymbol{Sum - Up} = \boldsymbol{Su} = 2 + 3 + 4 + 7 + +8 + 12 = \boldsymbol{36}$$

$$\boldsymbol{Sum - Down} = \boldsymbol{Sd} = 5 + 9 + 10 + 13 + 14 + 15 = \boldsymbol{66}$$

$$\boldsymbol{Sum - On} = \boldsymbol{So} = 1 + 6 + 11 + 16 = \boldsymbol{34}$$

مثال: أكتب برنامجاً لإدخال قيم في مصفوفة مربعة ذات أبعاد 4x4 بحيث يقوم بما يلي:

١. حساب مجموع عناصر المصفوفة.

٢. حساب مجموع العناصر التي تقع فوق القطر الثانوي.

٣. حساب مجموع العناصر التي تقع تحت القطر الثانوي.

٤. حساب العناصر التي تقع على القطر الثانوي وأيضا يطبع المصفوفة بشكل جدول.

الحل: نكتب كود البرنامج كما هو مبيّن:

```
Private Sub Command7_Click( )
Dim X(4, 4) As Single
Dim I As Integer, J As Integer
Dim Sum As Single, Su As Single
Dim Sd As Single, So As Single
Sum = 0: Su = 0: Sd = 0: So = 0
For I = 1 To 4: For J = 1 To 4
X(I, J) = Val (InputBox ("x(" & I & "," & J & ")"))
Next J: Next I
For I = 1 To 4: For J = 1 To 4
Sum = Sum + X(I, J)
Next J, I
For I = 1 To 4: For J = 1 To 4
If I + J < 4 + 1 Then Su = Su + X(I, J)
Next J: Next I
For I = 1 To 4: For J = 1 To 4
If I + J > 4 + 1 Then Sd = Sd + X(I, J)
Next J, I
For I = 1 To 4: For J = 1 To 4
If I + J = 4 + 1 Then So = So + X(I, J)
Next J, I
Print " - - - - - - - - - - - - - - - - -"
Print "Sum = "; Sum, "Su = "; Su, "Sd = "; Sd, "So = "; So
Print " - - - - - - - - - - - - - - - - -"
For I = 1 To 4: For J = 1 To 4
Print X(I, J),
Next J
Print
Next I
End Sub
```

$$[X_{ij}] = \begin{bmatrix} X_{11} & X_{12} & X_{13} & \boldsymbol{X_{14}} \\ X_{21} & X_{22} & \boldsymbol{X_{23}} & X_{24} \\ X_{31} & \boldsymbol{X_{32}} & X_{33} & X_{34} \\ \boldsymbol{X_{41}} & X_{42} & X_{43} & X_{44} \end{bmatrix}$$

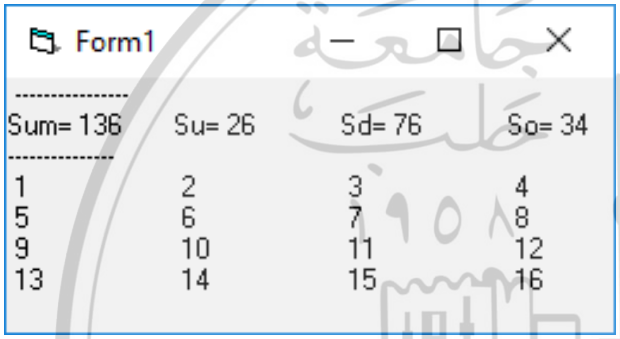
صفوف المصفوفة:

$$i = 1\ To\ m = 4$$

أعمدة المصفوفة:

$$j = 1\ To\ n = 4$$

عندها سيكون الناتج كما هو مبيَّن:



تحليل وتوضيح الناتج كما هو مبيَّن:

$$\boldsymbol{Sum} = All\ Numbers.$$

$$\boldsymbol{Sum} = \sum X_{ij}$$

$$\boldsymbol{Sum - Up} = \boldsymbol{Su} = The\ Numbers\ Up\ the\ \boldsymbol{MMain}\ Diameter.$$

$$\boldsymbol{Su} = \sum X_{ij} \quad if \quad i + j < m + 1$$

$$\boldsymbol{Sum - Down} = \boldsymbol{Sd} = The\ Numbers\ Down\ the\ \boldsymbol{MMain} Diameter.$$

$$\boldsymbol{Sd} = \sum X_{ij} \quad if \quad i + j > m + 1$$

$$\boldsymbol{Sum - On} = \boldsymbol{So} = The\ Numbers\ On\ the\ \boldsymbol{MMain} Diameter.$$

$$\boldsymbol{So} = \sum X_{ij} \quad if \quad i + j = m + 1$$

$$\boldsymbol{Sum} = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14 + 15 + 16 = \boldsymbol{136}$$

$$\boldsymbol{Sum - Up} = \boldsymbol{Su} = 1 + 2 + 3 + 5 + 6 + 9 = \boldsymbol{26}$$

$$\boldsymbol{Sum - Down} = \boldsymbol{Sd} = 8 + 11 + 12 + 14 + 15 + 16 = \boldsymbol{76}$$

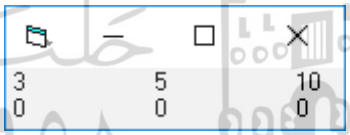
$$\boldsymbol{Sum - On} = \boldsymbol{So} = 4 + 7 + 10 + 13 = \boldsymbol{34}$$

مثال: أكتب برنامجاً لإدخال قيم في مصفوفة ثم طباعتها ثم مسحها ومن ثم طباعتها كما يلي:

الحل: نكتب كود البرنامج كما هو مبيّن:

```
Private Sub Command9_Click( )
Dim A(5) As Integer
A(1) = 3
A(2) = 5
A(3) = 10
Print A(1), A(2), A(3)
Erase A
Print A(1), A(2), A(3)
End Sub
```

عندها سيكون الناتج كما هو مبيَّن:

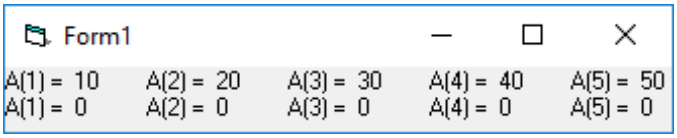


مثال: أكتب برنامجاً لإدخال قيم في مصفوفة ثم طباعتها ثم مسحها ومن ثم طباعتها كما يلي:

الحل: نكتب كود البرنامج كما هو مبيّن:

```
Private Sub Command8_Click( )
Dim A(5) As Integer
For I = 1 To 5
 A(I) = Val(InputBox("A(" & I & ") = "))
Next I
For I = 1 To 5
 Print "A(" & I & ") = "; A(I),
Next I
Erase A
Print
For I = 1 To 5
 Print "A(" & I & ") = "; A(I),
Next I
Print
End Sub
```

عندها سيكون الناتج كما هو مبيَّن:

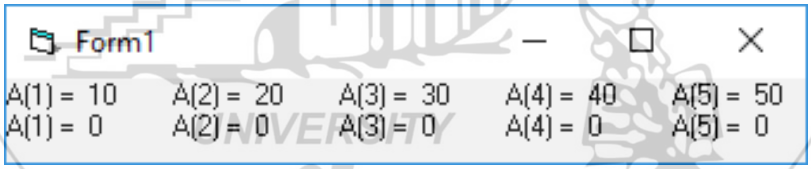


مثال: أكتب برنامجاً لإدخال قيم في مصفوفة ثم طباعتها ثم مسحها ومن ثم طباعتها كما يلي:

الحل: نكتب كود البرنامج كما هو مبيّن:

```
Private Sub Command10_Click( )
Dim A( ) As Integer, I, N As Integer
N = Val(InputBox("The Total Numbers"))
ReDim A(N)
For I = 1 To N
A(I) = Val(InputBox("A(" & I & ") = "))
Next I
For I = 1 To N
 Print "A(" & I & ") = "; A(I),
Next I
Erase A
Print
ReDim A(N)
For I = 1 To N
 Print "A(" & I & ") = "; A(I),
Next I
Print
End Sub
```

عندها سيكون الناتج كما هو مبيَّن:



جامعة
حلب
١٩٥٨
UNIVERSITY
OF
ALEPPO

الفصل السابع
التوابع
Functions

مقدمة:

التوابع نوعان:

- توابع معرفة ضمن اللغة البرمجية: أمنتها لغة VB لتسهيل العمل على المبرمج.
- توابع مبنية ضمن البرنامج.

يعطي التابع عادة قيمة واحدة، ولكل تابع مواصفات هي الاسم والمتحول أو الوسيط.

وشكله العام:

$$Var = Function\ Name\ (Par)$$

ويجب أن تكون قيمة المتحول أو الوسيط (Par) معرفة (طبعاً).

ملاحظة:

كلمات المتغيرات يجب أن لا تكون كلمات محجوزة ضمن البرنامج نفسه لتمييز وظيفتها عن وظيفة كلمات البرنامج المحجوزة مسبقاً.

التوابع الهامة:

تابع القيمة المطلقة $Abs(Number)$:

الشكل العام للتابع:

$$= Abs(Number)$$

يعيد هذا التابع القيمة المطلقة لرقم ما، رقم بدون إشارة.

$Number$: الرقم المراد قيمته المطلقة.

$$y = Abs(-2.1) = 2.1 \qquad y = Abs(2.1) = 2.1$$
$$y = Abs(-2.6) = 2.6 \qquad y = Abs(2.6) = 2.6$$

تابع الجزء الصحيح $Fix(Number)$:

يعيد هذا التابع الجزء الصحيح من التعبير العددي الحقيقي:

$y = Fix(2.1) = 2$ $y = Fix(-2.1) = -2$

$y = Fix(2.6) = 2$ $y = Fix(-2.6) = -2$

تابع التقريب (التدوير) $CInt(Number)$:

يعمل هذا التابع على تقريب العدد الحقيقي إلى أقرب عدد صحيح:

$y = CInt(2.1) = 2$ $y = CInt(-2.1) = -2$

$y = CInt(2.6) = 3$ $y = CInt(-2.6) = -3$

تابع التقريب الصحيح $Int(Number)$:

يعيد هذا التابع أكبر عدد صحيح أصغر أو يساوي التعبير العددي الحقيقي:

$y = Int(2.1) = 2$ $y = Int(-2.1) = -3$

$y = Int(2.6) = 2$ $y = Int(-2.6) = -3$

توابع النسب المثلثية $Sin(\theta),\ Cos(\theta),\ Tan(\theta)$:

وهي تمثل التوابع المثلثية حيث تكون الزاوية مقدرة بالراديان حصراً.

$Tan\left(\frac{\pi}{4}\right) = 1$ $Cos\left(\frac{\pi}{2}\right) = 0$ $Sin\left(\frac{\pi}{6}\right) = 0.5$

- يعيد التابع Sin قيمة من النوع Double محددة لجيب الزاوية. تقع النتيجة في المجال (-1) إلى (+1).
- يعيد التابع Cos قيمة من النوع Double محددة لتجيب الزاوية. تقع النتيجة في المجال (-1) إلى (+1).
- يعيد التابع Tan قيمة من النوع Double محددة لظل الزاوية.
- يكون المعامل x من النوع المضاعف أو أي تعبير عددي صحيح يمثل زاوية ما بالراديان.
- لتحويل الدرجات إلى راديان نضرب بالقيمة $Pi/180$. وللتحويل من الراديان إلى الدرجات نضرب الراديان بالقيمة $180/Pi$.
- نحسب قيمة Pi من العلاقة $Pi = 22/7$ أو من العلاقة $Pi = 4 * Atn(1)$

تابع الظل العكسي $Atn(Number)$:

وهو عبارة عن تابع ظل التمام (الظل العكسي) لرقم ما.

$\pi = 4 * Atn(1)$ $\frac{\pi}{4} = Atn(1)$ $Tan\left(\frac{\pi}{4}\right) = 1$

- يكون المعامل x من النوع المضاعف أو أي تعبير عددي صحيح.
- إن ناتج هذا التابع مقدر بالراديان وينحصر في المجال $\left]-\frac{\pi}{2} \ldots + \frac{\pi}{2}\right[$.
- يجب أن نمّيز بين التابع Atn والتابع Cotangent والذي يعبر عن قيمة مقلوب التابع Tan أي (1/Tan).

تابع اللوغاريتم الطبيعي $Log(Number)$**:**

يعيد هذا التابع اللوغاريتم الطبيعي للتعبير العددي الحقيقي (وليس اللوغاريتم العشري). يجب أن يكون العدد من النوع المضاعف Double أو أي تعبير رقمي صحيح أكبر من الصفر. والشكل العام للتابع هو:

$$Log(Number)$$

اللوغاريتم الطبيعي هو اللوغاريتم للأساس e. أما قيمة الثابت e فهي e = 2.718282 تقريباً.

يمكن أن نحسب اللوغاريتمات ذات الأساس n لأي رقم x من تقسيم اللوغاريتم الطبيعي للعدد x على اللوغاريتم الطبيعي للأساس n. لذا يمكن أن نكتب اللوغاريتم العشري (الذي أساسه العدد 10) من الصيغة التالية:

$$Log10(x) = \frac{Log(x)}{Log(10)}$$

ولذلك يمكن أن نكتب مجموعة من القيم كما يلي:

$$Log(x) = y \qquad Log(10) = 2.30258509299405$$
$$Log(2.718282) = 1$$
$$Log(1) = 0$$

ويمكن أن نكتب:

$$Logn(x) = \frac{Log(x)}{Log(n)} \qquad Log10(10) = \frac{Log(10)}{Log(10)} = 1$$

$$Logn(x) = \frac{Log(x)}{Log(n)} \qquad Log10(10) = \frac{Log(10)}{Log(2.718282)} = 2.30258509299405$$

إذا ببساطة يقوم التابع Log (x) بإعطاء اللوغاريتم الطبيعي لعدد معين x والذي يجب أن يكون أي عدد موجب. والشكل العام لهذا التابع:

$$Log(1) = 0 \qquad Log(2.718282) = 1 \qquad Log\big(Exp(1)\big) = 1$$

إذا أردنا الحصول على اللوغاريتم العشري نقوم بتقسيم اللوغاريتم الطبيعي للقيمة على اللوغاريتم الطبيعي للعدد 10.

$$y = Ln(x) = \frac{Log(x)}{Log(10)}$$

$$Log(5) = 1.609438 \quad Log(10) = 2.302585 \quad Log(5)/Log(10) = 0.69897$$

تابع الجذر التربيعي $Sqr(Number)$:

يعيد هذا التابع الجذر التربيعي للتعبير العددي الحقيقي:

$$y = Sqr(9) = 3 \qquad y = Sqr(100) = 10$$

معامل القسمة العادية / أي (Floating-Point Division, the Forward Slash /):

معامل القسمة العادية ويعطي ناتج القسمة العادية (العدد العشري):

$$A = I/J = 5/2 = 2.5 \qquad J = 2 \qquad I = 5$$

معامل القسمة الصحيحة \ أي (Integer Division, the Back Slash \):

معامل القسمة الصحيحة ويعطي الجزء الصحيح فقط من ناتج القسمة الصحيحة:

$$A = I\backslash J = 5\backslash 2 = 2 \qquad J = 2 \qquad I = 5$$

باقي القسمة الصحيحة Mod أي المودول:

معامل باقي القسمة الصحيحة ويعطي الباقي فقط من ناتج القسمة الصحيحة:

$$A = I\ Mod\ J = 5\ Mod\ 2 = 1 \qquad J = 2 \qquad I = 5$$

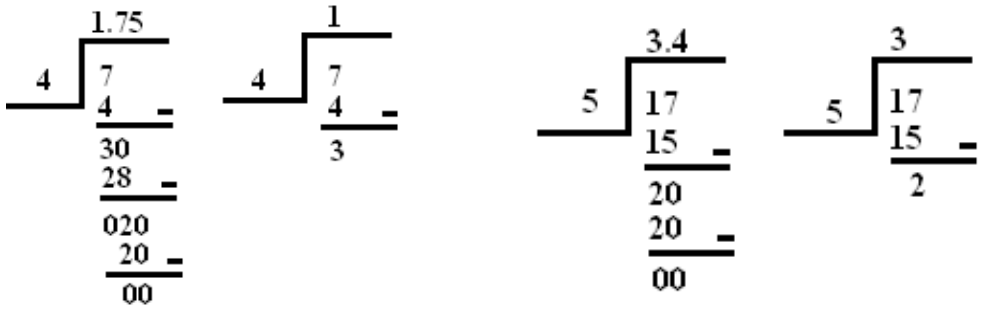
أيضاً نجد:

$$A = I\backslash J = 7/4 = 1.75$$
$$A = I\backslash J = 7\backslash 4 = 1 \qquad J = 4 \qquad I = 7$$
$$A = I\ Mod\ J = 7\ Mod\ 4 = 3$$

أيضاً نجد:

$$A = I\backslash J = 17/5 = 3.4$$
$$A = I\backslash J = 17\backslash 5 = 3 \qquad J = 5 \qquad I = 17$$
$$A = I\ Mod\ J = 17\ Mod\ 5 = 2$$

تظهر عملية القسمة كما في الشكل:



- عند إجراء عملية القسمة الصحيحة لعدد عشري فإنّه سيتم تقريب العدد العشري إلى أقرب عدد صحيح قبل إجراء القسمة الصحيحة ثم تتم عملية القسمة الصحيحة.
- عند إجراء عملية القسمة للحصول على الباقي لعدد عشري فإنّه سيتم تقريب العدد العشري إلى أقرب عدد صحيح قبل إجراء القسمة للحصول على الباقي ثم تتم عملية القسمة للحصول على الباقي.
- عند إجراء عملية القسمة للحصول على الباقي للعدد x على العدد y وكان الناتج مساوياً للصفر، فهذا يعني أن العدد x يقبل القسمة على العدد y أو نقول أن العدد x من قواسم العدد y أو نقول أنه من مضاعفاته.

$I = 7.7 \quad J = 4 \quad A = I \backslash J = 7.7 \backslash 4 = 2$

$I = 8 \quad J = 4 \quad A = I \backslash J = 8 \backslash 4 = 2$

$I = 7.7 \quad J = 2 \quad A = I\ Mod\ J = 7.7\ Mod\ 2 = 0$

$I = 8 \quad J = 2 \quad A = I\ Mod\ J = 8\ Mod\ 2 = 0$

- نلاحظ من المثال السابق أنَّه عند تقريب العدد 7.7 سنحصل على العدد 8.

$I = 7.3 \quad J = 4 \quad A = I \backslash J = 7.3 \backslash 4 = 1$

$I = 7 \quad J = 4 \quad A = I \backslash J = 7 \backslash 4 = 1$

$I = 7.3 \quad J = 2 \quad A = I\ Mod\ J = 7.3\ Mod\ 2 = 1$

$I = 7 \quad J = 2 \quad A = I\ Mod\ J = 7\ Mod\ 2 = 1$

- نلاحظ من المثال السابق أنَّه عند تقريب العدد 7.3 سنحصل على العدد 7.

التابع $Exp(Number)$:

الشكل العام لهذا التابع:

$y = Exp(x)$

يعيد هذا التابع القيمة $\boldsymbol{x}$ بالشكل $\boldsymbol{e^x}$ (أي أساس اللوغاريتم الطبيعي $\boldsymbol{e}$ مرفوعة للقوة $\boldsymbol{x}$) ويمكن أن تكون $\boldsymbol{x}$ أي تعبير عددي مضاعف صحيح أي من النوع Double:

$$a = Val(Text1.Text)$$
$$b = Exp(a)$$
$$Text2.Text = b$$

يقوم التابع $Exp(x)$ بإعطاء قيمة $\boldsymbol{e}$ المرفوعة لقوّة معيّنة $\boldsymbol{x}$ أي $\boldsymbol{e^x}$ حيث $\boldsymbol{e}$ هي أساس اللوغاريتم الطبيعي وقيمة $\boldsymbol{x}$ يجب أن تكون أصغر أو تساوي 709.782712893. وإذا تجاوزت $\boldsymbol{x}$ القيمة المسموحة سيحدث خطأ تجاوز الحد العلوي المسموح به أو ما يسمى خطأ الفيضان أو الطفح العلوي أي OverFlow.

Run-time error '6':
Overflow

يعتبر التابع Exp تابعاً عكسيّاً للتابع Log:

$$Exp(0) = 1 \qquad Log(1) = 0 \qquad Log\big(Exp(0)\big) = 0$$
$$Exp(1) = 2.718282 \qquad Log(2.718282) = 1 \qquad Log\big(Exp(1)\big) = 1$$

تابع الإشارة $Sgn(x)$:

يعطي هذا التابع $y = Sgn(x)$ قيمة صحيحة تمثل إشارة العدد الحقيقي. والشكل العام للتابع هو:

$$y = Sgn(x)$$

ونحصل على القيم التالية:

$y = 1$	إذا كانت قيمة x أكبر من الصفر	$y = 1 \; if \; x > 0$
$y = 0$	إذا كانت قيمة x تساوي من الصفر	$y = 0 \; if \; x = 0$
$y = -1$	إذا كانت قيمة x أصغر من الصفر	$y = -1 \; if \; x < 0$

التابع $Rnd(x)$:

يولد هذا التابع عدداً عشوائياً قيمته أكبر من أو تساوي الصفر وأصغر من الواحد أي:

$$y = Rnd(x) \; Then \; (0 \leq y < 1)$$

الصيغة العامة $Rnd(n)$ حيث المعامل الاختياري n هو عدد من النوع Single أو تعبير عددي صحيح. إذاً يعطي التابع $Rnd(x)$ سلسلة من الأعداد العشوائية أحادية الدقّة أكبر

من الصفر وأصغر من الواحد وهي ناتجة عن مجموعة عمليات رياضية مُعَقَّدة تعطي سلسلة من الأرقام العشوائية غير الموجّهة. وإذا تمّ تنفيذ البرنامج أكثر من مرّة سنحصل على نفس النتائج.

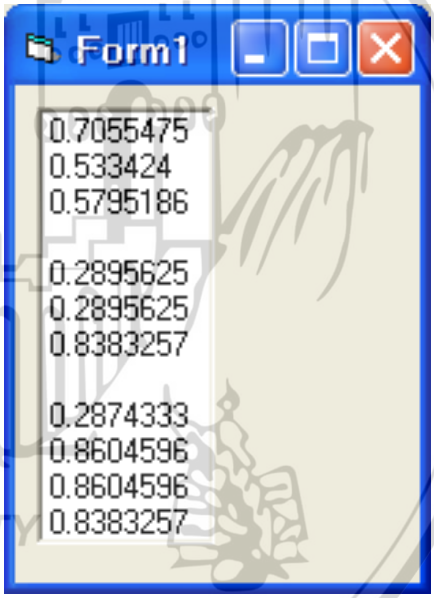
والشكل العام للتابع هو:

$$y = Rnd(n)$$

n – قيمة تبين كيف يقوم التابع Rnd بتوليد الرقم العشوائي التالي ويكون كما يلي:

- $n < 0$ سيقوم بتوليد نفس الرقم لأي n.
- $n > 0$ أو بدون وجود n (بإهمال n) سيولد الرقم العشوائي التالي.
- $n = 0$ سيولد الرقم العشوائي السابق.

```
List1.AddItem Rnd
List1.AddItem Rnd
List1.AddItem Rnd
List1.AddItem ""
List1.AddItem Rnd(2)
List1.AddItem Rnd(0)
List1.AddItem Rnd(-5)
List1.AddItem ""
List1.AddItem Rnd
List1.AddItem Rnd(2)
List1.AddItem Rnd(0)
List1.AddItem Rnd(-5)
```



العبارة $Randomize$:

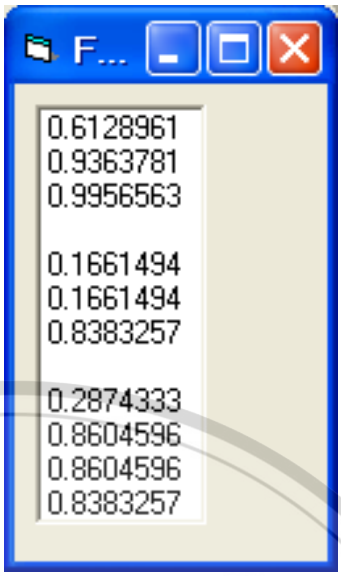
تفيد في تغيير عملية توليد الأرقام. والشكل العام لهذا التابع هو:

$$Randomize[Num]$$

Num – رقم اختياري يسهل عملية توليد الأرقام من النوع المتغير Variant أو أي تعبير عددي صحيح.

تستخدم العبارة $Randomize$ الرقم Num لتؤسس مولد الأرقام العشوائية للتابع Rnd وذلك بإعطائه قيمة أساسية جديدة. وإذا لم تتم كتابته فإن العبارة $Randomize$ ستستخدم القيمة العائدة من قبل مؤقت النظام كقيمة أساسية جديدة.

```
Randomize
List1.AddItem Rnd
List1.AddItem Rnd
List1.AddItem Rnd
List1.AddItem ""
List1.AddItem Rnd(2)
List1.AddItem Rnd(0)
List1.AddItem Rnd(-5)
List1.AddItem ""
List1.AddItem Rnd
List1.AddItem Rnd(2)
List1.AddItem Rnd(0)
List1.AddItem Rnd(-5)
```

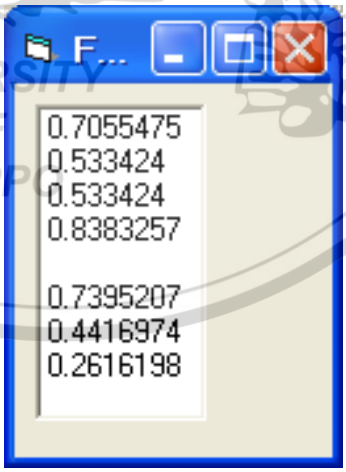


عند عدم استخدام العبارة $Randomize$ فإن التابع Rnd والذي لا يحتوي على متحولات سيستخدم نفس الرقم كقيمة أساسية في أول مرة يُستدعى فيها، وبناءً عليه سيستخدم الرقم الأخير المُوَلَّد كقيمة أساسية.

ملاحظة:

لتكرار مجموعة متعاقبة من الأرقام العشوائية نقوم باستدعاء التابع Rnd فوراً مع معامل سالب قبل استخدام $Randomize$ مع معامل عددي. ونلاحظ أن استخدام $Randomize$ مع نفس القيمة لـ Num سوف لن يكرر التعاقب السابق.

```
List1.AddItem Rnd
List1.AddItem Rnd(2)
List1.AddItem Rnd(0)
List1.AddItem Rnd(-5)
List1.AddItem ""
Rnd (-5)
Randomize (3)
List1.AddItem Rnd
List1.AddItem Rnd
List1.AddItem Rnd
```

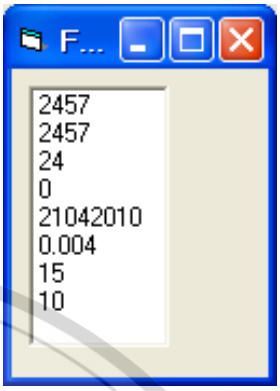


التابع $Val(String)$:

يعطي قيمة الأعداد الموجودة في السلسلة $String$ إلى قيمة عددية بالشكل المناسب.

أي يعيد هذا التابع القيمة العددية لسلسلة من الرموز $String$، ويجب أن يكون أول رمز عبارة عن رقم أو فاصلة عشرية ورقم أو إشارة ورقم وإلا فسيعطي هذا التابع قيمة صفرية:

$List1.AddItem\ Val("2457")$
$List1.AddItem\ Val("2\ \ 45\ \ 7")$
$List1.AddItem\ Val("24\ And\ 57")$
$List1.AddItem\ Val("And\ 57")$
$List1.AddItem\ Val("21\ 04\ 2010\ Aleppo")$
$List1.AddItem\ Val("0.004\ and\ 10")$
$List1.AddItem\ Val("\&H\ F")$
$List1.AddItem\ Val("\&O\ 12")$



ملاحظة:

يتوقف التابع Val عن قراءة السلسلة عند رمز أو حرف لا يستطيع تمييزه كجزء من العدد. وغالباً ما يأخذ هذا التابع بعين الاعتبار الرموز والحروف كجزء من القيم العددية أمّا إشارات الدولار والفواصل فلا يميّزها.

يميّز هذا التابع السابقة $\&O$ من أجل النظام الثماني $Octinary$ وكذلك $\&H$ للنظام الست عشري $Hexinary$. أمّا الفراغات والقفزات ورموز السطر الجديد $(Linefeed)$ فتُزال من المعامل.

يميّز التابع Val فقط الفاصلة (.) كفاصلة عشرية صحيحة.

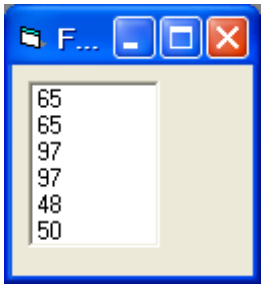
التابع $Asc(String)$:

يعيد هذا التابع قيمة عددية صحيحة تمثل شفرة المحرف $Character$ المطابق للحرف الأول في سلسلة ما $String$. والصيغة العامة هي:

$$y = Asc(String)$$

إذاً يعيد هذا التابع ترميز المحرف المناظر للحرف الأول في السلسلة الرمزية x فمثلاً:

$List1.AddItem\ Asc("A")$
$List1.AddItem\ Asc("Apple")$
$List1.AddItem\ Asc("a")$
$List1.AddItem\ Asc("a10")$
$List1.AddItem\ Asc("0")$
$List1.AddItem\ Asc("259")$



ملاحظات:

يمكن للتابع أن يمثل 256 محرفاً أو $Character$ وستكون القيمة التي تنتج عن استخدام التابع محصورة ضمن المجال من 0 وحتى 255 على الأنظمة غير النظام $DBCS$ وقد تكون في المجال من -32768 إن مجال القيمة التي يعطيها التابع وحتى 32767 على نظم $DBCS$. حيث DBCS: Double-byte character set.

يجب أن يكون المعامل $String$ تعبيراً سلاسليّاً، وسيحدث خطأ أثناء التنفيذ إذا لم يحتوي هذا المعامل على محارف. فمثلاً استخدام الكود المبيّن سيؤدي إلى وقوع الخطأ:

$List1.AddItem\ Asc("")$

Run-time error '5':
Invalid procedure call or argument

التابع $Chr(x)$:

يعيد هذا التابع سلسلة متضمنة المحرف المرتبط مع شفرة المحرف المحدّد. والصيغة العامة لهذا التابع هي:

$$y = Chr(ChrCode)$$

حيث يُمثل المعامل اللازم $ChrCode$ بقيمة من النوع $Long$ لتعريف المحرف. أي يعيد هذا التابع المحرف المناظر للرمز x المعطى فمثلاً:

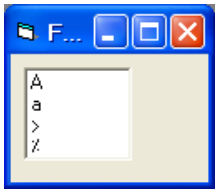
$List1.AddItem\ Chr(65)$
$List1.AddItem\ Chr(97)$
$List1.AddItem\ Chr(62)$
$List1.AddItem\ Chr(37)$
$List1.AddItem\ Chr(0)$
$List1.AddItem\ Chr(2)$
$List1.AddItem\ Chr(10)$
$List1.AddItem\ Chr(13)$

ملاحظات:

تتشابه المحارف ذات الشفرة من 0 وحتى 31 مع جدول القياس لشفرة $ASCII$ فالقيمة $Chr(10)$ تُعبّر عن محرف (رمز) الانتقال إلى السطر التالي. والمجال الطبيعي للمعامل $ChrCode$ يتراوح بين 0 و 255 ضمناً ويصبح هذا المجال من -32768 إلى 65536 لأنظمة مجموعة المحارف ذات البايت المضاعف $DBCS$.

إن استخدام التابع مع سلاسل رمزية عددية سيؤدي إلى نفس المضمون لأن التابع سيتعرف على قيمها العددية:

```
List1.AddItem Chr("65")
List1.AddItem Chr("97")
List1.AddItem Chr("62")
List1.AddItem Chr("37")
```



يجب أن يكون المعامل $ChrCode$ يتراوح بين 0 و 255 ضمناً ، وسيحدث خطأ أثناء التنفيذ إذا تم استخدامه لسلاسل رمزية أو أرقام خارج المجال المحدد.

```
List1.AddItem Chr("")
List1.AddItem Chr("Hammad")
List1.AddItem Chr(300)
```

Run-time error '13':
Type mismatch
Run-time error '5':
Invalid procedure call or argument

التابع $Str(Num)$:

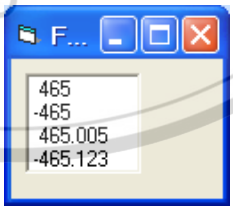
يعيد هذا التابع سلسلة أو متغيّر ممثل لعدد ما، أي يحول هذا التابع قيمة عددية Num إلى سلسلة رمزية. والصيغة العامة هي:

$$y = Str(Num)$$

ملاحظة:

عند تحويل الأعداد إلى سلاسل يُحجز دائماً الفراغ اليساري ليستخدم مع إشارة العدد فإذا كان العدد موجباً سنجد أن السلسلة التي تُعبر عن هذا العدد بعد تحويله تتضمن فراغاً من اليسار دون أي محرف (للتعبير عن أنّ العدد موجب)، أمّا إذا كان سالباً فستتوضّع الإشارة في هذا الفراغ للدلالة على أن العدد سالب.

```
List1.AddItem Str(465)
List1.AddItem Str(−465)
List1.AddItem Str(465.005)
List1.AddItem Str(−465.123)
```



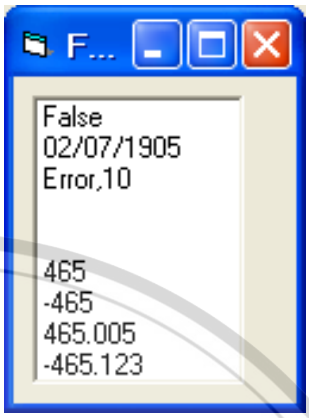
التابع $CStr(Num)$:

يحول هذا التابع قيمة عددية x إلى سلسلة رمزية دون أن يقوم بحجز مكان للإشارة، فإذا كانت الإشارة موجبة يظهر العدد بدون الإشارة وبدون مكان فارغ مخصص للإشارة وإذا كان العدد سالب يظهر العدد وأمامه إشارة السالب. والصيغة العامة للتابع:

$$y = CStr(Num)$$

فمثلاً:

```
Dim B As Boolean, D As Date
x = 5 : y = 4
B = x = y
D = 2010
List1.AddItem CStr(B)
List1.AddItem CStr(D)
List1.AddItem CStr("Error, 10")
List1.AddItem CStr("")
List1.AddItem CStr(Error10)
List1.AddItem CStr(465)
List1.AddItem CStr(−465)
List1.AddItem CStr(465.005)
List1.AddItem CStr(−465.123)
```



ملاحظة:

إذا كان المتحول منطقيّاً يعطي قيمته المنطقية وإذا كان من النوع $Date$ سيظهره بشكل $Date$ وإذا كانت العبارة موجودة بين إشارتي تنصيص سيظهرها كما هي وإذا كانت سلسلة أو متحول رمزي لكن بدون إشارتي تنصيص سيتجاهلها وسيطبع متحول قيمته فراغ.

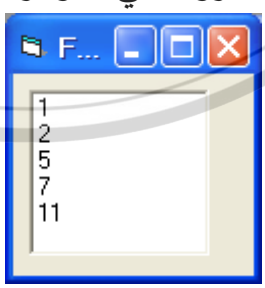
التابع $Len(x)$:

يعيد هذا التابع عدد الرموز في المتغير الرمزي x ، ويتضمن هذا العدد الفراغات. والشكل العام للتابع:

$$y = Len(x)$$

لاحظ وجود الفراغات بين الحروف في الأوامر الكودية:

```
List1.AddItem Len("h")
List1.AddItem Len("ha")
List1.AddItem Len("h  am")
List1.AddItem Len("h a m m")
List1.AddItem Len("Hello World")
```



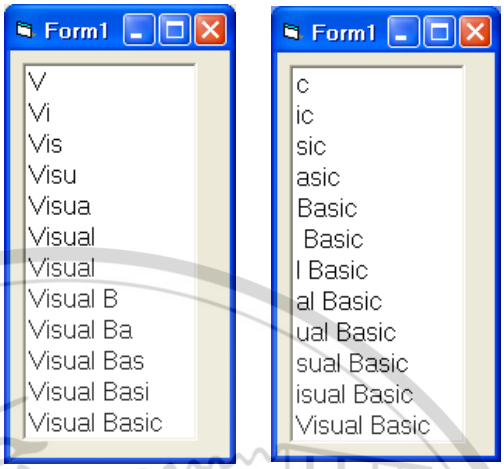
يوجد فراغين بين الحرفين
يوجد فراغ بين كل لحرفين
يوجد فراغ بين الكلمتين

التابع $Left(x, y)$:

يعيد هذا التابع متغير رمزي يتألف من y رمز من على يسار المتغير الرمزي x. يجب أن تكون قيمة y بين 0 و 255 .

- إذا كانت قيمة y أكبر من عدد رموز المتغير الرمزي x، تعاد قيمة كامل المتغير.
- إذا كانت قيمة y تساوي الصفر يعاد متغير رمزي بطول صفر.

```
Dim A As String
x = "Visual Basic"

For I = 1 To Len(x)
 List1.AddItem Left(x,I)
Next

For I = 1 To Len(x)
 List2.AddItem Right(x,I)
Next
```



التابع $Right(x,y)$:

يعيد هذا التابع متغير رمزي يتألف من y رمز من على يمين المتغير الرمزي x. يجب أن تكون قيمة y بين 0 و255.

- إذا كانت قيمة y أكبر من عدد رموز المتغير الرمزي x، تعاد قيمة كامل المتغير.
- إذا كانت قيمة y تساوي الصفر يعاد متغير رمزي بطول صفر.

التابع $Mid(x,y\ [,k])$:

يعيد هذا التابع متغير رمزي من وسط المتغير الرمزي x يتألف من k رمز ابتداءً من الرمز y. والشكل العام للتابع:

$$String = Mid(Stringvar, Start\ [, Length])$$

ملاحظات:

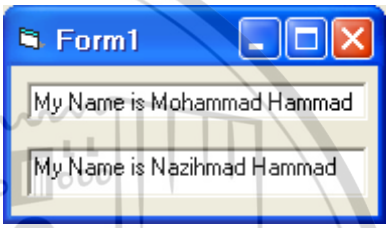
- يجب أن تكون قيمة y و k بين 1 و255.
- إذا حذفت k، أو كان عدد الرموز التي ابتداءً من y وعلى يمينها أقل من k، تعاد جميع رموز السلسلة ابتداءً من y وعلى يمينها.
- إذا كانت قيمة y أكبر من عدد رموز x يعاد متغير رمزي بطول صفر.

```
Dim x As String
x = "My Name is Mohammad Hammad"
Text1.Text = x
List1.AddItem Mid(x,12,8)
```

$List1.AddItem\ Mid(x, 12, 4)$
$List1.AddItem\ Mid(x, 6, 4)$
$List1.AddItem\ Mid(x, 2, 4)$
$List1.AddItem\ Mid(x, 30)$
$List1.AddItem\ Mid(x, 4)$

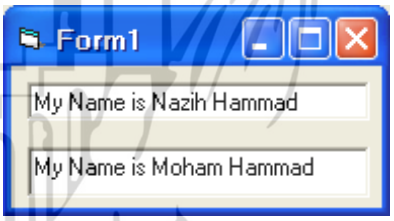
- كما يمكن باستخدام هذا التابع استبدال جزء من وسط متغير رمزي بسلسلة رمزية أخرى:

$x = "My\ Name\ is\ Mohammad\ Hammad"$
$Text1.Text\ =\ x$
$Mid(x, 12, 8)\ =\ "Nazih"$
$Text2.Text\ =\ x$



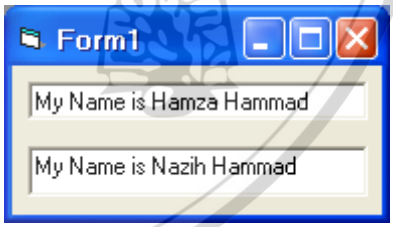
- قد تكون السلسلة الجديدة أطول من القديمة أو العكس (لاحظ الفرق):

$x = "My\ Name\ is\ Nazih\ Hammad"$
$Text1.Text = x$
$Mid(x, 12, 5) = "Mohammad"$
$Text2.Text = x$



- قد تكون السلسلة الجديدة أطول من القديمة أو العكس (لاحظ الفرق):

$x = "My\ Name\ is\ Hamza\ Hammad"$
$Text1.Text = x$
$Mid(x, 12, 8) = "Nazih"$
$Text2.Text = x$



- إذا كانت قيمة y مساوية للصفر ستظهر عبارة الخطأ:

$Text2.Text = Mid(x, 0, 8)$

Run-time error '5':
Invalid procedure call or argument

التابع $String(Num, y)$:

يعيد هذا التابع متغير رمزي بطول Num، جميع رموزه تكرار للمحرف الذي يكون رمزه في جدول $ASCII$ هو y إذا كانت قيمة y رقمية، أما إذا كان الوسيط y متغير رمزي، تكون جميع رموزه تكرار للمحرف الأول في المتغير y.

والشكل العام للتابع:

$$String = String(Num, y)$$

يمكن استخدام التابع Val أو عدم استخدامه كما يلي:

$List2.AddItem\ String(5, Val(65))$
$List2.AddItem\ String(5, "65")$
$List2.AddItem\ String(5, (65))$
$List2.AddItem\ String(5, Val(aaaaa))$
$List2.AddItem\ String(5, "aaaaa")$
$List2.AddItem\ String(5, (aaaaa))$
$List2.AddItem\ String(5, "abcde")$
$List2.AddItem\ String(5, "ab\ 24")$

ملاحظات:

- الحالة الأولى حول قيمة الكود 65 إلى الحرف A وقام بطباعته خمس مرات.
- الحالة الثانية قام بمعرفة المحرف الأول من السلسلة الرمزية "65" وهو الرقم 6 وقام بطباعته خمس مرات.
- الحالة الثالثة حول قيمة الكود 65 إلى الحرف A وقام بطباعته خمس مرات.
- الحالة الرابعة يحاول معرفة قيمة عددية لكنها وجد أحرف فأصبحت القيمة العددية لديه مساوية للصفر لذا لم يطبع أي شيء.
- الحالة الخامسة قام بمعرفة المحرف الأول من السلسلة الرمزية "aaaaa" وهو الحرف a وقام بطباعته خمس مرات.
- الحالة السادسة يحاول معرفة قيمة عددية لكنها وجد أحرف فأصبحت القيمة العددية لديه مساوية للصفر لذا لم يطبع أي شيء.
- الحالة السابعة قام بمعرفة المحرف الأول من السلسلة الرمزية "abcde" وهو الحرف a وقام بطباعته خمس مرات.
- الحالة الثامنة قام بمعرفة المحرف الأول من السلسلة الرمزية "ab 24" وهو الحرف a وقام بطباعته خمس مرات.

التابع $Space(x)$:

يعيد هذا التابع متغير رمزي مكون من الفراغات بطول x. يجب أن تكون قيمة x عددية صحيحة وبين 0 و 255، وهي تقرب إذا كانت حقيقية.

$x = Space(10)$
$Text1.Text = x$
$y = "Hello" \& Space(10) \& "world"$
$Text2.Text = y$

- ستكون قيمة المتغير المطبوعة في مربع النص الأول عبارة عن عشر فراغات.
- ستكون قيمة المتغير المطبوعة في مربع النص الثاني هي الكلمتين وبينهما عشر فراغات.

التابع $InStr([j,]x\ ,y\ [,c])$:

يعيد هذا التابع قيمة من النوع Variant (Long) حيث يُحدّد مكان التواجد (الحدوث) الأول لسلسلة واحدة ضمن سلسلة أخرى. والصيغة العامة للتابع هي:

$InStr([Start,]\ String1, String2\ [,Compare])$

حيث:

$Start$ – قيمة اختيارية، وهي عبارة عن تعبير رقمي يحدد المكان الذي سيتم ابتداء عملية البحث منه. إذا حُذِفَ هذا الوسيط فإن عملية البحث ستبدأ من موضع المحرف الأول، وإذا احتوى $Start$ على $Null$ سيحدث خطأ. ويُعتبر وجود هذا العامل ضرورياً إذا تم استخدام الوسيط $Compare$.

$String1$– وسيط إجباري وهو عبارة عن السلسلة التي يتم البحث فيها.

$String2$– وسيط إجباري، وهو عبارة عن السلسلة التي يتم البحث عنها.

$Compare$ – وسيط اختياري، ويعيّن نوع المقارنة للسلسلة. يمكن أن نحذف هذا الوسيط ويمكن أن يأخذ أحد القيم التالية:

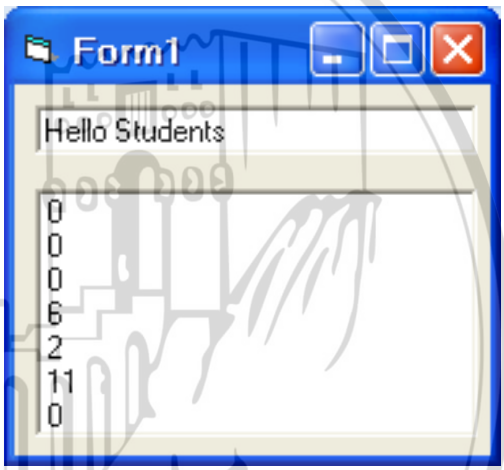
- 0 – وهي القيمة الافتراضية وتستخدم لتنفيذ مقارنة ثنائية.
- 1 – وتستخدم لإجراء مقارنة نصيّة غير حسّاسة.
- 2 – وهي مقارنة تعتمد على معلومات محتواة ضـــمن قاعدة بيانات معيّنة، وتستخدم من أجل Microsoft Access.

- إذا كانت قيمة $Compare = Null$ عندئذ سيحدث خطأ. وإذا حذف الوسيط $Compare$ فإن وضع $OptionCompare$ هو الذي يحدد نوع المقارنة.

إذاً يبحث هذا التابع عن أول حدوث للمتغير الرمزي y في المتغير الرمزي x. ويعيد قيمة عددية صحيحة تدل على موضع التطابق.

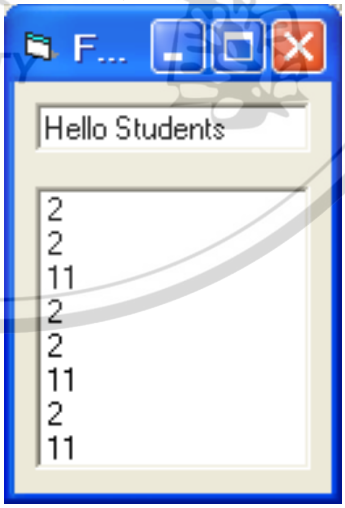
- في حال ذكر j فإن عملية البحث تبدأ ابتداءً من الموضـــع j، يجب أن تكون قيمة j صحيحة وضمن المجال من 1 ولغاية 255، وفي حال غير ذلك سيتولد خطأ استدعاء غير قانوني للتابع.
- إذا كانت قيمة j أكبر من طول x، يعيد هذا التابع قيمة 0.

$x = "Hello\ Students"$
$Text1.Text = x$
$List1.AddItem\ InStr("","e")$
$List1.AddItem\ InStr(0,"e")$
$List1.AddItem\ InStr(x,0)$
$List1.AddItem\ InStr(x," ")$
$List1.AddItem\ InStr(x,"e")$
$List1.AddItem\ InStr(3,x,"e")$
$List1.AddItem\ InStr(30,x,"e")$



يجب أن يكون $Start$ صحيحاً وإذا كان حقيقيّاً سيتم تقريبه إلى أقرب عدد صحيح.

$x = "Hello\ Students"$
$Text1.Text = x$
$List1.AddItem\ InStr(x,"e")$
$List1.AddItem\ InStr(1,x,"e")$
$List1.AddItem\ InStr(7,x,"e")$
$List1.AddItem\ InStr(1.1,x,"e")$
$List1.AddItem\ InStr(1.9,x,"e")$
$List1.AddItem\ InStr(9.1,x,"e")$
$List1.AddItem\ InStr(2.1,x,"e")$
$List1.AddItem\ InStr(2.7,x,"e")$



إذا كانت قيمة $Start$ معدومة أو مهملة يعطي إشارة الخطأ:

$List1.AddItem\ InStr("",x,"e")$

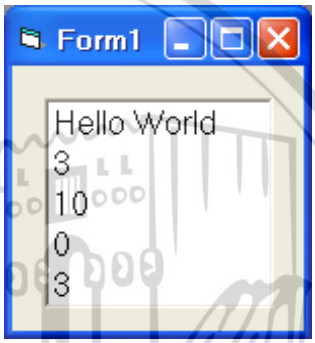
Run-time error '13':
Type mismatch

لاحظ البحث عن المقطع أو السلسلة "Wo":

```
x = "Hello World"
List1.AddItem x
List1.AddItem InStr (3, x, "Wo")
List1.AddItem InStr (6, x, "Wo")
List1.AddItem InStr (20, x, "Wo")
List1.AddItem InStr (x, "Wo")
```

لاحظ البحث عن المحرف أو السلسلة "l":

```
x = "Hello World"
List1.AddItem x
List1.AddItem InStr (3, x, "l")
List1.AddItem InStr (6, x, "l")
List1.AddItem InStr (20, x, "l")
List1.AddItem InStr (x, "l")
```



التابع $Trim(x)$:

يعيد هذا التابع كتابة متغير رمزي ما بعد حذف كل الفراغات التي على يمينه ويساره.

التابع $LTrim(x)$:

يعيد هذا التابع كتابة متغير رمزي ما بعد حذف كل الفراغات التي على يساره.

التابع $RTrim(x)$:

يعيد هذا التابع كتابة متغير رمزي ما بعد حذف كل الفراغات التي على يمينه.

```
Dim x As String
x = "   Hammad   "
List1.AddItem Trim("")
List1.AddItem Trim(x)
List1.AddItem "Hello" & Trim(x) & "Hello"
List1.AddItem LTrim(x)
List1.AddItem LTrim(x) & "Hello"
List1.AddItem RTrim(x)
List1.AddItem "Hello" & RTrim(x)
```

التوابع الزمنية:

التابع $Time$:

يعطي هذا التابع الوقت الحالي مكتوب فيه الساعات والدقائق والثواني.

التابع $Date$:

يعطي هذا التابع التاريخ الحالي مكتوب فيه اليوم والشهر والسنة.

التابع Now:

يعطي هذا التابع التاريخ كما في التابع السابق وبجانبه الوقت.

التابع $Day\ (Now)$:

يعطي هذا التابع رقم اليوم الحالي.

التابع $Month\ (Now)$:

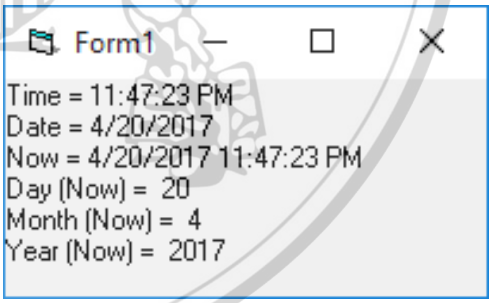
يعطي هذا التابع رقم الشهر الحالي.

التابع $Year\ (Now)$:

يعطي هذا التابع رقم السنة الحالية.

أمثلة على التوابع الزمنية:

```
Private Sub Command1_Click()
Print "Time = "; Time
Print "Date = "; Date
Print "Now = "; Now
Print "Day (Now) = "; Day(Now)
Print "Month (Now) = "; Month(Now)
Print "Year (Now) = "; Year(Now)
End Sub
```



التوابع المبنية ضمن البرنامج و عملية إنشاء التوابع

إن التوابع عادة تعطي قيمة معينة لدى استعمالها ضمن الكود البرمجي، فهي عبارة عن تعليمات حسابية معرفة ضمن اسم مفروض ومعرف لدينا مسبقاً يستعمل لاختصار السطور البرمجية والتقليل من احتمال وجود الأخطاء.

سبق وأن استعرضنا التوابع المعرفة ضمن اللغة البرمجية وذكرنا أن الشكل العام للتوابع يتألف من اسم ووسيط يتم وضعه بين قوسين، أي:

$$Var = Function\ Name\ (Par)$$

Var – اسم المتحول y.

$Function$ – اسم التابع f.

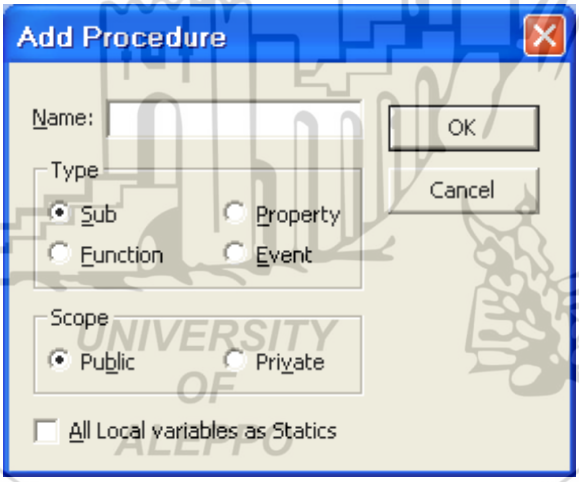
Par – قيمة أو اسم الوسيط x.

إن إنشاء التوابع يكون من أجل علاقات تعطي في خرجها قيمة واحدة فقط. ففي مثال حل المعادلة من الدرجة الثانية لا نستطيع إنشاء تابع واحد يعطي احتمالات حل هذه المعادلة.

الآن سنقوم بإنشاء التوابع لوحدنا، لذلك يجب اختيار اسم التابع بحيث يعبر عن وظيفته.

وبعد الانتقال إلى شاشة البرمجة في البرنامج نستطيع إنشاء التابع كما يلي:

من قائمة الأوامر المنسدلة نختار Tools – Add Procedure فيظهر مربع حوار نجعل شكله كالتالي:



إن الكلمات الموجودة في مربع الحوار هي:

Name – اسم التابع.

Type – نوع التابع:

- Sub – تابع فرعي
- Function – تابع وظيفي
- Property – تابع إحتمالي
- Event – للدلالة على حدث

Scope - نوع التعليق:

- Public – عام لكافة إطارات المشروع ووحداته.
- Private – للإجراءات في نفس الملف فقط.

إن الفرق بين التصريحين Private & Public هو أن الإجراء في حالة Public يكون عاماً ولذلك يمكن إستخدامه من قبل أي إجراء آخر وفي أي موقع من ملفات المشروع، إذ أنه يمكن للبرنامج الواحد أن يتألف من عدة ملفات (وربما عدة نماذج) وبالتالي عند التصريح عن إجراء بأنه Public فمعنى ذلك أنه بمقدور كل الملفات إستخدامه. أمّا عند إستخدام Private فلا يمكن إستخدام الإجراء إلا من قبل الإجراءات الموجودة ضمن نفس الملف.

نكتب اسم التابع والذي يفترض أن يعبر عن معنى هذا التابع ثم نضغط الزر OK فتظهر البرمجة الكودية التالية:

```
Private Function FName()
    [VB Code]
End Function
```

نستطيع مباشرة إما قبل أو بعد كتابة البرمجة الكودية المطلوبة للبرنامج أن نبدأ كتابة برمجة التابع الذي نريد. ففي المثال الذي سيتبع كلامنا سنقوم بإنشاء التابع Fact وذلك كالتالي:

```
Private Function Fact(M)
If M = 0 Then
   Fact = 1
Else
   Fact = 1
   For i = 2 To M
      Fact = Fact * i
   Next i
End If
End Function
```

نلاحظ ما يلي:

- تم اختيار الاسم بحيث لا يكون معرفاً ضمن اللغة البرمجية (أو حتى ضمن المشروع) وبحيث يعبّر عن الغاية من استخدامه (حساب عاملي عدد صحيح موجب).
- تم تعريف متغير وهمي للتابع M ، وتم حصره بين قوسين هلاليين ( ).

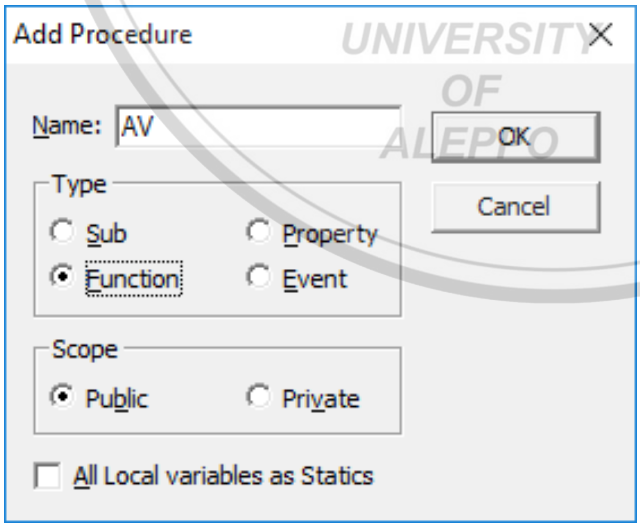
- يمكن أن نكتب اسم التابع (بدون إلحاقه بالوسيط) على يسار تعليمة الإسناد أي يمكن كتابة الإسم Fact بدلاً من التسمية الطويلة Fact(M) ولكن يجب على اسم التابع أن يظهر على الأقل مرة واحدة على يسار تعليمة التعيين أي على الشكل Fact(M) وإلا لن يعمل البرنامج الذي يعتمد عليه.
- في حالة استخدام البرمجة الكودية بإستخدام تابع فرعي Function ستنقسم البرمجية إلى:
  - الوحدة البرمجية الأساسية: وهي الوحدة الداعية التي تستدعي البرامج الفرعية ويمثلها سطر البرمجة الكودية ضمن البرنامج الرئيسي.
  - الوحدة البرمجية الفرعية: والتي يتم إستدعاؤها من قبل الوحدة الأساسية الداعية وتمثلها تعليمات التابع الفرعي Function الذي نقوم بإنشائه.

أمثلة على التوابع المبنية ضمن البرنامج و عملية إنشاء التوابع:

مثال:

المطلوب كتابة الكود اللازم لحساب المتوسط الحسابي لثلاثة أعداد بحيث تتم عملية الحساب داخل تابع والذي يتم استدعائه برمجياً.

من القوائم المنسدلة نختار $Tools - Add\ Procedure$ فيظهر مربع حوار نحدد فيه الوسائط كما يلي:



نكتب اسم التابع: AV.

نكتب نوع التابع: Function.

نحدد كيفية استخدام التابع: Public.

ثم نضغط على الخيار Ok فيظهر الكود المبيّن:

$Public\ Function\ AV( )$

$End\ Function$

نعرف الوسائط المستخدمة في اسم البرنامج كما يلي:

$AV(X\ \boldsymbol{As\ Single}, Y\ \boldsymbol{As\ Single}, Z\ \boldsymbol{As\ Single})$

نكتب الكود البرمجي المطلوب تنفيذه كما يلي:

$AV = (X + Y + Z) / 3$

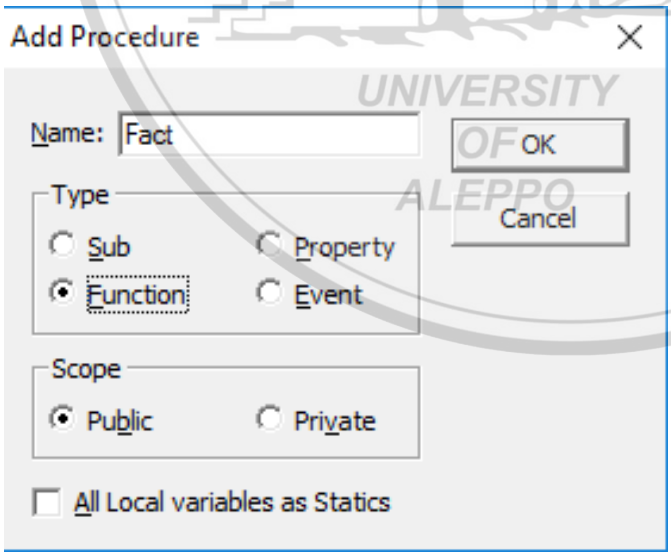
نستخدم زر أوامر نكتب فيه الأوامر الكودية المطلوب لاستدعاء التابع وتنفيذ المطلوب ونستخدم مربع نص لإظهار النتيجة كما يلي:

```
Private Sub Command1_Click()
  Dim X As Single, Y As Single, Z As Single
  Dim F As Single
  X = Val(InputBox("Insert the 1 – st Number X = "))
  Y = Val(InputBox("Insert the 2 – nd Number Y = "))
  Z = Val(InputBox("Insert the 3 – rd Number Z = "))
  F = AV(X, Y, Z)
  Text1.Text = F
End Sub
```

مثال:

المطلوب كتابة الكود اللازم لحساب عاملي عدد محدد بحيث تتم عملية الحساب داخل تابع والذي يتم استدعائه برمجياً.

من القوائم المنسدلة نختار Tools – Add Procedure فيظهر مربع حوار نحدد فيه الوسائط كما يلي:



نكتب اسم التابع: Fact.

نكتب نوع التابع: Function.

نحدد كيفية استخدام التابع:

Public.

ثم نضغط على الخيار Ok

فيظهر الكود المبيّن:

```
Public Function Fact( )

End Function
```

نعرف الوسائط المستخدمة في اسم البرنامج كما يلي:

```
Fact (N As Integer)
```

نكتب الكود البرمجي المطلوب تنفيذه كما يلي:

```
Public Function Fact (N As Integer)
 Dim I As Integer, S As Integer
 S = 1
 For I = 1 To N
  S = S * I
 Next I
 Fact = S
End Function
```

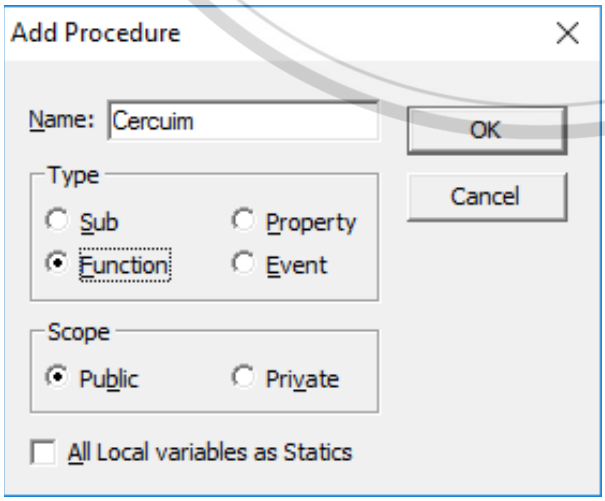
نستخدم زر أوامر نكتب فيه الأوامر الكودية المطلوب لاستدعاء التابع وتنفيذ المطلوب ونستخدم الإطار لإظهار النتيجة كما يلي:

```
Private Sub Command1_Click()
 Dim K As Integer, R As Integer
 K = Val(InputBox("Enter any Integer Number"))
 R = Fact(K)
 Print "K = "; K, "Fact = "; R
End Sub
```

مثال:

المطلوب كتابة الكود اللازم لحساب محيط دائرة بحيث تتم عملية الحساب داخل تابع والذي يتم استدعائه برمجياً.

من القوائم المنسدلة نختار Tools – Add Procedure فيظهر مربع حوار نحدد فيه الوسائط كما يلي:



نكتب اسم التابع: Cercuim.

نكتب نوع التابع: Function.

نحدد كيفية استخدام التابع:

Public.

ثم نضغط على الخيار Ok

فيظهر الكود المبيّن:

```
Public Function Cercuim
End Function
```

نعرف الوسائط المستخدمة في اسم البرنامج كما يلي:

$Cercuim\ (N\ \boldsymbol{As\ Integer})$

نكتب الكود البرمجي المطلوب تنفيذه كما يلي:

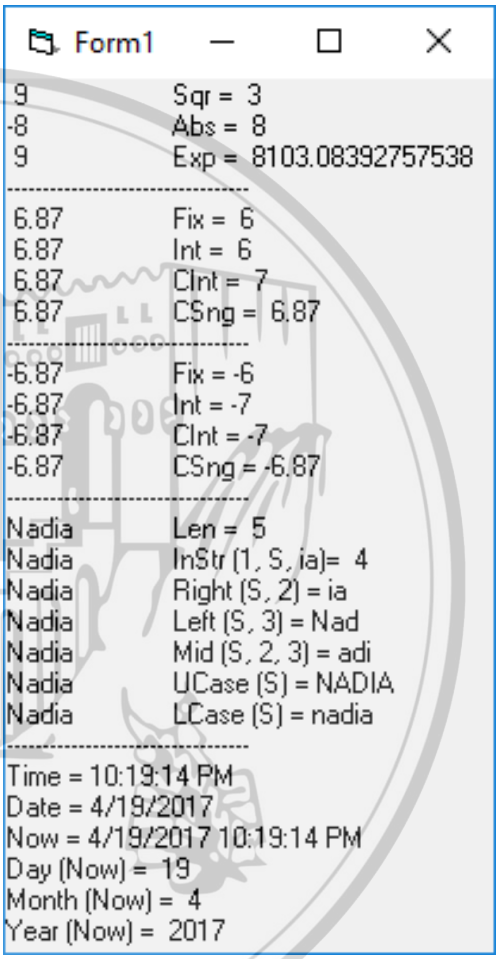
```
Public Function Cercuim (N As Integer)
 C = 2 * (22 / 7) * N
 Cercuim = C
End Function
```

نستخدم زر أوامر نكتب فيه الأوامر الكودية المطلوب لاستدعاء التابع وتنفيذ المطلوب ونستخدم الإطار لإظهار النتيجة كما يلي:

```
Private Sub Command1_Click()
 Dim K As Integer, R As Integer
 K = Val(InputBox("Enter The Radius"))
 R = Cercuim (K)
 Print "K = ", K, " Cercuim = ", R
End Sub
```

أمثلة إضافية على التوابع:

```
Private Sub Command1_Click()
  Dim X As Integer, Y As Integer
  Dim Z As Single, S As String
  X = 9
  Print X, "Sqr = "; Sqr(X)
  Y = -8
  Print Y, "Abs = "; Abs(Y)
  Print X, "Exp = "; Exp(X)
  Print "----------"
  Z = 6.87
  Print Z, "Fix = "; Fix(Z)
  Print Z, "Int = "; Int(Z)
  Print Z, "CInt = "; CInt(Z)
  Print Z, "CSng = "; CSng(Z)
  Print "----------"
  Z = -Z
  Print Z, "Fix = "; Fix(Z)
  Print Z, "Int = "; Int(Z)
  Print Z, "CInt = "; CInt(Z)
  Print Z, "CSng = "; CSng(Z)
  Print "----------"
  S = "Nadia"
  Print S, "Len = "; Len(S)
  Y = InStr(1, S, "ia")
  Print S, "InStr (1, S, ia) = "; Y
  Print S, "Right (S, 2) = "; Right(S, 2)
  Print S, "Left (S, 3) = "; Left(S, 3)
    Print S, "Mid (S, 2, 3) = "; Mid(S, 2, 3)
  Print S, "UCase (S) = "; UCase(S)
  Print S, "LCase (S) = "; LCase(S)
  Print "----------"
  Print "Time = "; Time
  Print "Date = "; Date
  Print "Now = "; Now
  Print "Day (Now) = "; Day(Now)
  Print "Month (Now) = "; Month(Now)
  Print "Year (Now) = "; Year(Now)
End Sub
```



الفصل الثامن
الألوان
Colors

مقدمة *Introduction*:

في كثير من الأحيان وأثناء استعمال أدوات VB كنّا نرى أن بعض من خصائص النافذة أو من خصائص بعض الأدوات في المرحلة المرئية (التصميمية) كانت تحتاج إلى الألوان مثل $(BackColor, ForeColor, FillColor)$ بالإضافة إلى بعض الأوامر الكودية أي في المرحلة التنفيذية مثل $(PSet, Line, Circle)$ والتي كانت تستعمل دالة أو وسيطة معينة من هذه الوسائط لها علاقة بالألوان.

في كثير من الأحيان أيضاً ولعدم الحاجة الماسة للون الأداة أو الخاصية كنّا نهمل ميزة اللون أو الدليل الذي يتحدث عن اللون عندها يؤخذ اللون الافتراضي للخاصية $ForeColor$. لكن نحتاج كثيراً وخصوصاً في المراحل التنفيذية كرسم المخططات مثلاً إلى اختيار وإعداد الألوان، في هذه الحالة يحدد لكل أداة أو لكل خاصية منها اللون المناسب ويكون الإجراء الكودي مثلاً:

$$ControlName.PropertyName = Value$$

حيث:

$ControlName$ هو اسم الأداة.

$PropertyName$ اسم الخاصية.

$Value$ القيمة التي يأخذها اللون.

إذا انتبهنا إلى خاصية اللون $BackColore$ في خصائص النافذة سنجد أنها عبارة عن قيم صحيحة طويلة ولا يمكن بسهولة فهم هذه الأرقام غير العادية فمثلا من الصعب جداً فهم أن القيمة $\&H00FFFF00\&$ تدل على اللون الأزرق السماوي ولذلك كان لا بد من التفكير بطريقة صحيحة لتمييز واختيار الألوان.

طرق تمثيل الألوان وهي :

يزودنا VB بثلاث طرق مختلفة لتمثيل ووصف الألوان وهي :

١) الثوابت الرمزية $Symbolic\ Constants$.

٢) الخاصية أو التابع $QBColor$ أي $QBColor\ Function$.

٣) الخاصية أو التابع RGB أي $RGB\ Function$.

١. الثوابت الرمزية *Symbolic Constants*:

يستخدم في الـ VB ثماني ثوابت رمزية لتمثيل الألوان الرئيسية ويمكن استخدام كل ثابت من هذه الثوابت كوسيط أو دالة يمثل لون ما في رسم المخططات أو كقيمة لخاصية ما. يمكن تمثيل هذه الثوابت الرمزية مع الألوان التي تمثلها في الجدول التالي:

Constant	Color	Constant	Color
$vbBlack$	$Black$	$vbBlue$	$Blue$
$vbRed$	Red	$vbMagenta$	$Magenta$
$vbGreen$	$Green$	$vbCyan$	$Cyan$
$vbYellow$	$Yellow$	$vbWhite$	$White$

٢. الخاصية أو التابع *QBColor* أي *QBColor Function*:

يمكن أن لا تكون الألوان الثمانية كافية في بعض الأحيان لذلك يمكن الإستعاضة عن طريقة الثوابت الرمزية بطريقة أخرى هي التابع $QBColor$ والتي تعطينا ستة عشر لوناً وهنا تعطى قيمة للون على شكل تابع $QBColor$ لمتحول Index بالشكل $QBColor(Index)$. إن قيمة الدليل $Index$ تتغير من الـ 0 وحتى الـ 15 وتتغير معها الألوان الدالة عليها والجدول التالي يظهر هذه القيم:

Index	Color	Index	Color
0	$Black$	8	$Gray$
1	$Blue$	9	$Light\ Blue$
2	$Green$	10	$Light\ Green$
3	$Cyan$	11	$Light\ Cyan$
4	Red	12	$Light\ Red$
5	$Magenta$	13	$Light\ Magenta$
6	$Brown$	14	$Yellow$
7	$White$	15	$Light\ (Bright)\ White$

٣. الخاصية أو التابع RGB أي RGB Function:

قد تكون الألوان الستة عشرة غير كافية أحياناً لذلك لا بد عندها من استخدام طريقة جديدة ولذا يمكن استخدام الخاصية أو التابع الوظيفي RGB والذي يزودنا بأكثر من 16 مليون لون ممكن وهنا من الأكيد أن أي لون سنحتاجه سيكون موجوداً.

إن اسم التابع الظاهر باستخدام الميزة RGB سيكون $RGB(Red, Green, Blue)$ حيث تمثل القيم $Red, Green, Blue$ شدة (كثافة) كل لون من هذه الألوان من اللون الأساسي لها حيث تتراوح هذه القيم بين القيمة 0 التي تمثل أقل كثافة ممكنة من هذا اللون وبين 255 والتي تمثل الكثافة العظمى من هذا اللون ومن الواضح أن أي قيمة من لون ما ستؤثر على الألوان الأخرى وبذلك نرى أن اللون الأحمر المطلق هو $RGB(255, 0, 0)$ وإذا أنشأنا جدولاً بالألوان الأساسية سنجد :

اللون Color	التابع RGB	اللون Color	التابع RGB
الأحمر Red	$RGB(255, 0, 0)$	السماوي Cyan	$RGB(0, 255, 255)$
الأخضر Green	$RGB(0, 255, 0)$	البنفسجي Magenta	$RGB(255, 0, 255)$
الأزرق Blue	$RGB(0, 0, 255)$	الأصفر Yellow	$RGB(255, 255, 0)$
الأبيض White	$RGB(255, 255, 255)$	الأسود Black	$RGB(0, 0, 0)$

ومن الواضح أن وجود لونين معينين يؤدي إلى مزجهما وبالتالي سيظهر لون ناتج من خليطهما (ناتج عملية المزج) ويتأثر فيهما فمثلا البنفسجي هو مزيج اللونين الأزرق والأحمر وبالتالي سنكتب التابع RGB له بالشكل التالي $RGB(255, 0, 255)$ وهكذا.

وهكذا إذا أردنا أخذ خلفية الإطار باللون الأحمر مثلا يمكن أن نكتب الكود المبيَّن بإحدى الطرق الثلاثة:

$$Form.BackColor = vbRed$$
$$Form.BackColor = QBColor(4)$$
$$Form.BackColor = RGB(255, 0, 0)$$

جامعة
حلب
١٩٥٨
UNIVERSITY
OF
ALEPPO

الفصل التاسع
الرسم في الفيجوال البيزك
Drawing in VB

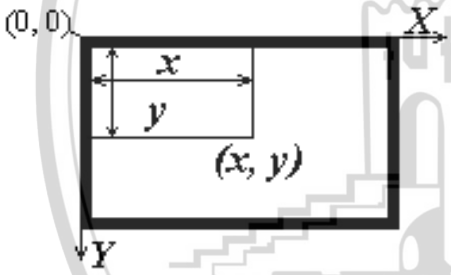
وحدة القياس *Twip*:

وهي وحدة لقياس المسافة تستخدم لتحديد مواقع العناصر الرسومية وأماكنها وأبعادها. وتقرأ تويب وتقاس كما يلي:

$1\ Inch\ =\ 1440\ Twip$

يتم تعريف وحدة القياس هذه لكي نتمكن من تحديد أماكن توضّع ووجود الكائنات على مربعات الصور أو الإطارات ويستخدم أحياناً واحدات مثل (نقطة، بكسل، رمز، إنش، مم، سم) ولكن الأكثر شيوعاً هنا هي التويب.

نظام الإحداثيات:



يمكن إنشاء الرسوم ضمن الأدوات Form أو Picture ولكل منها نظام إحداثياته الخاص به حيث تقع نقطة الأصل (0, 0) والتي تنسب إليها إحداثياته عند الركن الأيسر العلوي للأداة، وتزداد بالاتجاه الأيمن (المحور x) وإلى أسفل (المحور y) كما هو موضح بالشكل.

محور الإحداثيات هو النقطة العليا اليسارية الممكن الرسم فيها.

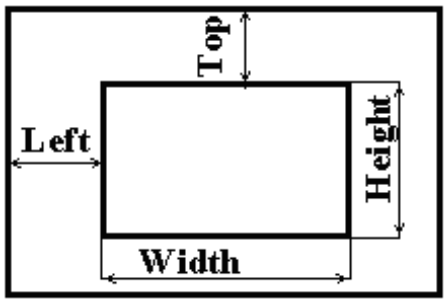
إحداثيات نقطة ما (x, y) هو بعد هذه النقطة عن محور الإحداثيات.

تحريك الصورة أو الكائن:

عند ظهور أي كائن ضمن إطار أو مربع صور ما فإن توضعه ضمن هذه الإطار سيتحدد بمجموعة من الإحداثيات وهي :

١. بعد الطرف اليساري لهذا الكائن أو العنصر عن الطرف اليساري للإطار Left.
٢. بعد الطرف العلوي لهذا الكائن عن الطرف العلوي للإطار Top.
٣. عرض هذا الكائن ضمن الإطار Width.

٤. ارتفاع هذا الكائن ضمن الإطار Height.



يمكننا تحريك الصور أو الكائنات الموجودة على الإطار باستخدام إحدى الطريقتين التاليتين:

١. بتغيير الخاصيتين Left و Top.
٢. باستخدام الطريقة Move.

الخاصيتين Left و Top:

في هذه الحالة يتم تحريك الكائن أو العنصر عن طريق تغيير بعده عن الطرف الأيسر أو العلوي للإطار ويتم ذلك عن طريق إعطائه قيماً متغيرة فسيتجه باتجاه اليمين عند تزايد القيمة Left وسيتجه باتجاه اليسار عند تناقص القيمة Left وبالمقابل فإنه سيتجه إلى الأعلى عند تناقص القيمة Top أو سيتجه باتجاه الأسفل عند تزايد هذه القيمة.

م – كل هذا سيتم في المرحلة التنفيذية.

وتكون الأوامر بالشكل التالي:

$$ControlName.Left = ControlName.Left + 500$$
$$ControlName.Top = ControlName.Top - 100$$

وهذا يعني إزاحة الصورة (الأداة) 500 وحدة إلى اليمين و 100 وحدة باتجاه الأعلى.

الطريقة Move:

وهي طريقة أخرى لتحريك العناصر عن طريق التلاعب بأبعاد هذه العناصر أو بالإحداثيات التي تحدد مكان ظهورها على الإطار أو مربع الصورة والشكل العام لصيغة هذه الطريقة هو :

$$ControlName.Move\ newLeft, newTop, newWidth, newHeight$$

أي أننا نستطيع تحريكه وتغيير عرضه وارتفاعه. وجميع هذه الوسائط اختيارية عدا الوسيط Left الذي يعتبر ضرورياً.

م – عملية التكبير والتصغير تتم على كائنات الصور فقط وفي هذه الحالة يجب أن تأخذ الخاصية Stretch القيمة True مما يعني أن البرنامج يستطيع تكبير أو تصغير حجم الصورة.

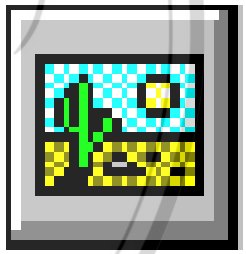
م – تستخدم طريقة Move لتحريك أي كائن Object باستثناء القوائم، وقد تكون بعض عناصر التحكم غير مرئية (Timer) ولهذا لا يحمل تحريكها أي معنى.

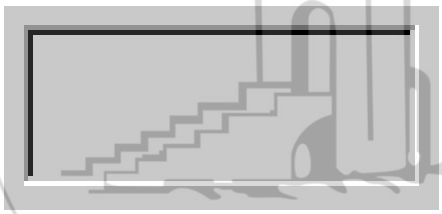
مربع الصورة Picture Box:

تعتبر أداة صندوق الصورة من أهم وأكثر الأدوات استعمالا في الـ VB لأنها تستخدم لعرض الصور وللتحكم بشكل وإطار وبتوليد عدة أشكال رسومية ولاستعمالات أخرى.

تؤخذ هذه الأداة Picture Box من مربع الأدوات Toolbox وتظهر بالشكل التالي:

في شريط الأدوات **على الإطار (الخصائص الافتراضية)**



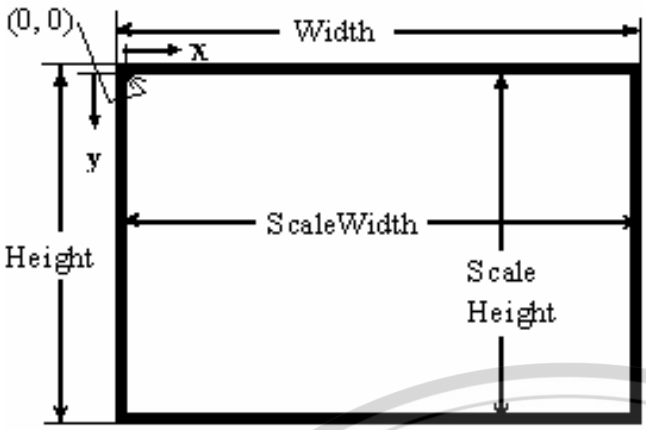


خصائص مربع الصورة *Picture Box Properties*:

هناك خصائص كثيرة لمربع الصورة يصعب التحدث عنها بإسهاب في محاضرة واحدة. سندرس ببعض التفصيل بعض من أهم الخصائص والخصائص الأخرى سنتحدث عنها أثناء حل التمارين:

Name: إسم الأداة.

Picture: يظهر الصورة المطلوب عرضها أو تحميلها في المربع.



AutoSize: الحجم التلقائي (الافتراضي).

BackColor: لون الخلفية.

ForeColor: لون إطار الأداة.

DrawStyle: نمط الرسم.

DrawWidth: عرض خط الرسم.

FillColor: اللون المستخدم للتعبئة.

FillStyle: نمط النقش أو التهشير (Pattern).

BorderStyle: نمط الإطار.

Left: بُعد الركن العلوي اليساري للصورة عن الجهة اليسرى لأداة الصورة PictureBox.

Top : بُعد الركن العلوي اليساري للصورة عن الجهة العليا لأداة الصورة PictureBox .

Width: عرض مربع الصورة PictureBox.

Height: ارتفاع مربع الصورة PictureBox.

ScaleWidth: مقياس البعد من جهة العرض.

ScaleHeight : مقياس البعد من جهة الارتفاع.

Enabled : حالة ظهور مربع الصورة والأحداث المرافقة له أثناء التنفيذ.

Visible: ظهور أو عدم ظهور مربع الصورة أثناء التنفيذ.

يجب التمييز بين خاصيتي Height والتي تمثل الارتفاع الحقيقي لمربع الصورة وبين مقياس الارتفاع ScaleHeight والذي يمثل الطول الحقيقي للمنقطة التي يتم فيها الرسم وهي دوماً أصغر من منطقة Height بمقدار البعد الذي يأخذه نمط الإطار BorderStyle ويمكن أن يتساوى البعدين أي Height مع البعد ScaleHeight فقط عندما تكون قيمة نمط الإطار "None" أي مساوية للصفر ومعنى ذلك أن كل مساحة مربع الأداة أو البعد ScaleHeight كارتفاع مسموحاً استخدامه في منطقة الرسم. الكلام ينطبق على الخاصيتين ScaleWidth و Width.

الآن نلاحظ أنه إذا رسمنا زر أوامر أو أي أداة خارج مربع الصورة ونقلناه إلى داخل المربع فلن يكون جزءاً من مربع الصورة ويدل على هذه عندما نحرك مربع الصورة فلن يتحرك بداخله الأداة الموضوعة. بينما إذا رسمنا أو حملنا أداة ما داخل أداة مربع الصورة سنلاحظ أن هذه الأداة ستتحرك مع مربع الصورة الذي نقوم بتحريكه.

أنواع الصور graphics formats التي يمكن تحميلها في مربع الصورة:

Bmp : وهي الملفات من النوع Bitmap تظهر أثناء التحميل بحجمها الحقيقي.

JPEG : وهي ملفات من النوع Joint Photographic Experts Group

Gif : وهي الملفات من النوع Graphic Interchange Format

Metafile: وهي الملفات التي تحتوي على مجوعة من الكائنات الرسومية مثل (الخطوط والدوائر والمستطيلات أو كثيرات الأضلاع). تأخذ هذه الملفات اللاحقة wmf

Icon: وهي نوع من أنواع الملفات Bmp ولكن الحجم الأعظمي لها هو 32 x 32 pixels.

طرق الرسم Graphics Methods :

إن الطريقة Method هي تابع أو إجراء يقوم بنقل فعل ما إلى أداة ما. تستخدم الطريقة في الأداة Picture Box لرسم شيء ما وتتم الطريقة فقط في الحالة التنفيذية. والإجراء الكودي للطريقة هو:

$$ControlName.MethodName\ [optional\ arguments]$$

حيثControlName هو اسم الأداة التي نتحدث عنها والتي نقوم بالرسم فيها.

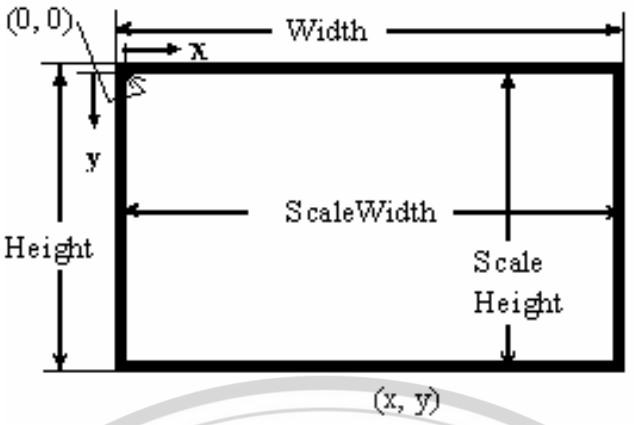
MethodName اسم الطريقة التي نقوم بتنفيذها.

Arguments وهي وسيطات قد نحتاجها أثناء استخدام الأمر أو الطريقة.

الطرائق المستخدمة بشكل عام هي رسم نقطة، خط، مربعات ودوائر في مربع الصورة مع كيفية استخدام بعض من خصائص أداة مربع الصورة مع هذه الطرائق.

المنظومة الإحداثية Coordinate System في مربع الصورة:

يتم الرسم عادة في إطار ما Form أو في مربع صورة Picture Box أو في أداة صورة Image وفي كل الحالات نحتاج إلى التذكير بمحور الإحداثيات الذي سيتم بداخله الرسم والذي سيكون كما في الشكل:



إن عملية الرسم تتم ضمن منطقة الرسم الموضحة في الشكل والتي عرضها ScaleWidth وطولها ScaleHeight والتي تختلف عن حجم مربع الرسم الكلي الذي يشغله ضمن الإطار والمحدد بالأبعاد Width و Height. قد تصبح قيمة Width وScaleWidth متساويتين وأيضاً القيمتين Height وScaleHeight وهذا متعلق بقيمة الخاصية Border Style والتي تأخذ إحدى القيمتين:

$Border\ Style\ =\ 0\ None$ عندها تكون القيم متساوية.

$Border\ Style = 1 Fixed\ Single$ يظهر اختلاف بالقيمة بمقدار سمك الإطار

يتضح هنا شكل مربع الصورة بالإضافة إلى الخاصيتين Height مع ScaleHeight وإلى الخاصيتين Width و ScaleWidth وتكون قيمة كل منهما متساويتين عندما تأخذ الخاصية BorderStyle القيمة None.

محور الإحداثيات هو النقطة العليا اليسارية الممكن الرسم فيها

إحداثيات نقطة ما (x, y) هو بعد هذه النقطة عن محور الإحداثيات.

تعليمة رسم نقطة PSet :

إن أسهل ما يمكن رسمه في المخططات هو نقطة والتي تستخدم في رسم نقطة بسيطة ما والتي أحيانا تعتبر كنقطة إرتكاز أو أساس لرسم أشكال هندسية أخرى. الشكل العام للتعليمة:

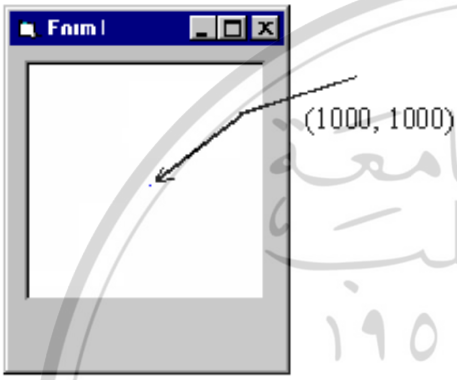
$PBName.PSet\ [step](x, y), Color$

- PBName هو إسم مربع الصورة الذي سنقوم برسم النقطة فيه.
- Pset أمر رسم النقطة.

- Step قيمة اختيارية فإذا أهملت ستكون قيم (x, y) منسوبة إلى مركز محور الإحداثيات أو ما يسمى نقطة الأصل للأداة وإذا كتبت ستكون قيم الإحداثيتين (x, y) منسوبين إلى آخر إحداثيتين تم استخدامها.
- (x, y) إحداثيات النقطة المطلوب تحديدها أو رسمها.
- Color هو اللون المنتقى لرسمها وفي حال إهمال هذه القيمة فمعنى ذلك أن اللون الافتراضي المعرف بالخاصية ForeColor هو اللون الذي سيتم أخذه أثناء الرسم.

مثال

رسم نقطة تبعد عن محور الإحداثيات بقيمة 1000 إلى اليمين و 1000 إلى الأسفل.




```
Private Sub Form_Click()
 PBox1.PSet (1000, 1000)
End Sub
```

مثال: $PBox1.Pset\ Step\ (250, 500)$

يعني رسم نقطة نبعد بمقدار 250 Twip إلى اليمين و بمقدار 500 Twip إلى الأسفل عن آخر نقطة رسمت في مربع الصورة PBox1.

$PBox1.Pset\ Step\ (250,500), RGB(255,0,0)$
$PBox1.Pset\ Step\ (250,500), VbBlack$

يعني رسم نقطة نبعد بمقدار 250 Twip إلى اليمين و بمقدار 500 Twip إلى الأسفل عن آخر نقطة رسمت في مربع الصورة $PBox_1$ بحيث يكون لون الرسم أحمراً في الأمر الأول وأسوداً في الثاني.

تعليمة رسم خط Line Method:

وهي طريقة متعددة الاستعمالات وبأشكال متعددة مثل رسم قطعة مستقيمة أو صندوق فارغ أو مصمت والصيغة العامة لها هي:

$ControlName.Line\ [Step](x1, y1) - [Step]\ (x2, y2), [Color], [B]\ [F]$

- ✓ Step قيمة اختيارية فإذا أهملت ستكون قيم (x, y) منسوبة إلى مركز محور الإحداثيات أو ما يسمى نقطة الأصل للأداة وإذا كتبت ستكون قيم الإحداثيتين (x, y) منسوبين إلى آخر إحداثيتين تم استخدامها.
- ✓ $(x_1, y_1), (x_2, y_2)$ إحداثيات نقاط بداية ونهاية المستقيم المطلوب رسمه.
- ✓ Color هو اللون المنتقى للرسم وفي حال إهمال هذه القيمة فمعنى ذلك أن اللون الافتراضي المعرف بالخاصية ForeColor هو اللون الذي سيتم أخذه أثناء الرسم.
- ✓ [B] اختيارية وهي تدل على أن المطلوب رسمه صندوق فارغ وفي هذه الحالة ستكون قيم كل من $(x_1,y_1), (x_2,y_2)$ ممثلة لإحداثيات قطر الصندوق.

[F] اختيارية وتستخدم فقط إذا تم استخدام B وهي تعني أن الصندوق المطلوب رسمه يجب أن يكون مصمتاً ويكون لونه هو لون FillColor الجاري وبما أنه لا يمكن أن تستخدم F بدون B لذلك لا توجد فاصلة بينهما.

١) رسم قطعة مستقيمة (x1, y1) – (x2, y2)

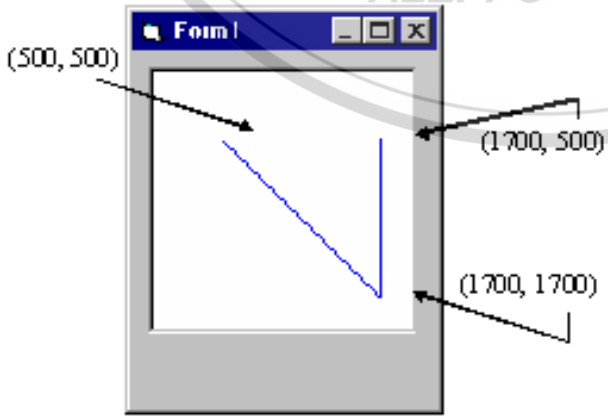
$$PBName.Line\ (x1, y1)\ -\ (x2, y2), Red$$

بداية المستقيم (x_1, y_1) و نهاية المستقيم (x_2, y_2). لون خط الرسم أحمر.

٢) رسم مستقيم يبدأ من نقطة محددة أو من نهاية مستقيم آخر (x3, y3) -

$$PBox.Line\ (x1, y1)\ -\ (x2, y2), RGB(0,0,255)$$
$$PBox.Line\ -\ (x3, y3), VbGreen$$

هنا سيتم رسم مستقيم أول يبدأ من النقطة (x_1, y_1) وينتهي بالنقطة (x_2, y_2) ولونه أزرق ومن نهاية هذا المستقيم سيتم رسم مستقيم لونه أخضر ينتهي بالنقطة (x_3, y_3).



مثال:

```
Private Sub Form_Click()
PBox.Line (500, 500)-
(1700, 1700)
PBox.Line -(1700, 500)
End Sub
```

مثال : $PBox.LineStep(100,200) - Step(850,700)$

يؤدي الأمر إلى رسم خط مستقيم يبدأ بإزاحة قيمتها 100 إلى اليمين و 200 إلى الأسفل عن آخر نقطة تم التعامل معها وينتهي بإزاحة قيمتها 850 لليمين و 700 إلى الأسفل عن نقطة البدء المذكورة سابقاً.

مثال : $PBox.Line\ (100,200) - Step(850,700)$

مثال : $PBox.LineStep(100,200) - \ (850,700)$

يؤدي الأمر الأول إلى رسم خط مستقيم يبدأ بإزاحة مطلقة قيمتها 100 إلى اليمين و 200 إلى الأسفل عن مركز الإحداثيات وينتهي بإزاحة قيمتها 850 لليمين و 700 للأسفل عن نقطة البدء المذكورة.

يؤدي الأمر الثاني إلى رسم خط مستقيم يبدأ بإزاحة قيمتها 100 إلى اليمين و 200 إلى الأسفل عن آخر نقطة تم التعامل معها وينتهي بنقطة إزاحتها المطلقة 850 لليمين و 700 للأسفل عن نقطة مبدأ الإحداثيات.

٣) رسم مستطيل أو صندوق Box

الصندوق أو المستطيل يبدأ بنقطة وينتهي قطرياً بنقطة أخرى لذلك فالأمر الكودي له مثل المستقيم مع إضافة الحرف B من كلمة Box للدلالة على الصندوقية.

$PBox.Line\ (x1,y1) - (x2,y2), Color, B$

الركن الأول من الصندوق (x_1, y_1) و الركن الثاني من الصندوق (x_2, y_2).

يمكن إهمال اللون عندها سيكون اللون هو لون التعبئة FillColor وبذلك نكتب الأمر كما يلي:

$PBox.Line\ (x1,y1) - (x2,y2), \ \ , B$

مثال : $PBox.Line\ (100,200\ ) - \ (850,700), VbYello, B$

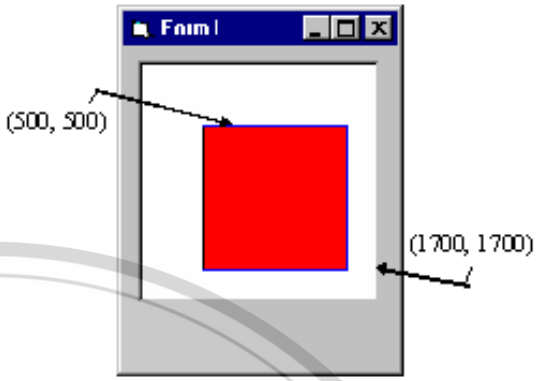
يؤدي هذا الأمر إلى رسم مستطيل إحداثيات قطره هي النقطة الأولى التي إحداثياتها المطلقة 100 إلى اليمين و 200 إلى الأسفل عن مركز الإحداثيات والنقطة الثانية إزاحتها المطلقة 850 لليمين و700 إلى الأسفل عن نقطة مبدأ الإحداثيات وهو فارغ ولونه أصفر.

مثال : $PBox.Line\ (100,200) - \ (850,700), VbYello, BF$

يؤدي هذا الأمر إلى رسم مستطيل إحداثيات قطره هي النقطة الأولى التي إحداثياتها المطلقة 100 إلى اليمين و 200 إلى الأسفل عن مركز الإحداثيات والنقطة الثانية إزاحتها

المطلقة 850 لليمين و 700 إلى الأسفل عن نقطة مبدأ الإحداثيات وهو مصمت ولونه أصفر. لاحظ المثال:

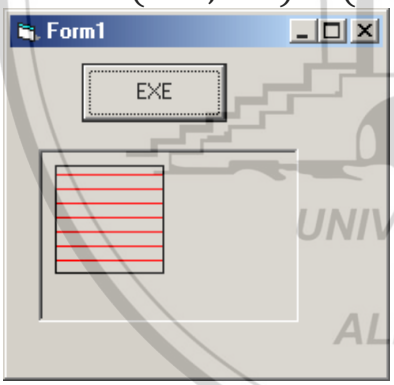
```
Private Sub Form_Click()
PBox.Line (500,500)
– (1700,1700),,BF
End Sub
```



هنا أخذ اللون أحمر مع العلم أنه تم إهماله وأخذ اللون من الخاصية ForeColor وكان مصمتاً لأننا أخذنا F في صيغة الأمر.

يمكن ملء المستطيل باستخدام الخاصيتين FillStyle و FillColor للواجهة Form دون استخدام الوسيط F كما يلي:

```
PBox.FillStyle = 2
PBox.FillColor = RGB(255,0,0)
PBox.Line (100,100) – (1000,1000),RGB(0,0,0),B
```



ترسم التعليمات التالية مستطيلاً باللون الأسود كما هو موضح في تعليمة رسم المستقيم RGB(0,0,0) ويكون مملوءاً بخطوط أفقية (القيمة FillStyle = 2) لون هذه الخطوط أحمر RGB(255,0,0).

ويبين الجدول التالي القيم الممكنة للخاصية *FillStyle*:

القيمة	الشكل
0	لون خالص مصمت
1	لون شفاف غير مرئي (القيمة الافتراضية)
2	خطوط أفقية
3	خطوط عمودية

4	خطوط مائلة للأعلى
5	خطوط مائلة للأسفل
6	خطوط رأسية وأفقية متقاطعة
7	خطوط مائلة متقاطعة

تعليمة رسم الدائرة Circle Method :

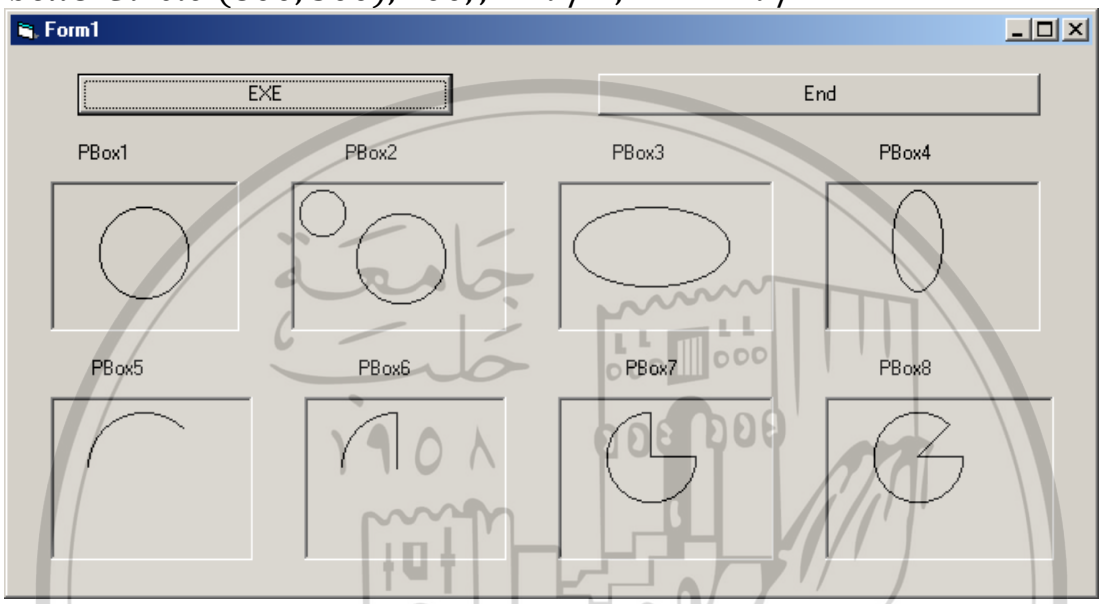
تستخدم هذه الطريقة لرسم الدوائر أو القطوع الناقصة أو الأقواس. الشكل العام لهذه التعليمة هو:

$$ControlName.Circle\ [Step](x,y), Radius, [color, Start, End, Aspect]$$

- ✓ Step اختياري يستخدم لرسم مركز الدائرة في إحداثيات منسوبة إلى آخر نقطة وقعت ضمن الأداة. إذا حذفت تكون إحداثيات مركز الدائرة مطلقة.
- ✓ x, y إحداثيات مركز الدائرة.
- ✓ Radius تحدد قيمة نصف قطر الدائرة.
- ✓ Color اختياري وهو يدل على اللون الذي سترسم به الدائرة ويمكن استخدام التابع RGB أو التابع QBColor أو التابع VbColor لتحديد اللون.
- ✓ Start, End اختيارية تحددا نقطتي بداية رسم القوس ونهايتها ويعبر عنهما بزوايا مقاسه بالراديان وقيمها بين $(-2\pi, 2\pi)$ بطريقة رياضية طبيعية أي ابتداءً من اليمين وبعكس عقارب الساعة. وإذا كانت كل منهما أو كلتاهما مستخدمتين بإشارة سالبة يوصل القوس عند الزاوية المستخدم معها الإشارة السالبة مع مركز الدائرة بمستقيم.
- ✓ Aspect اختياري ويستخدم هذا الوسيط لرسم قطوع ناقصة قيمته القياسية تساوي 1 حيث ترسم دائرة مكتملة. أما إذا كانت قيمته أقل من 1 فيرسم قطعاً ناقصاً محوره الكبير الأفقي وإذا كانت قيمته أكبر من 1 فيرسم قطعاً ناقصاً محوره الكبير رأسي وقيمته تحدد نسبة القطر باتجاه المحور y على القطر باتجاه المحور x.
- ✓ لاحظ الأمثلة التالية حيث سنقوم باستخدام الأمر Circle برسم دوائر وأقواس وقطاعات زاوية وقطوع وسيتم تمثيل هذه الحالات على مربعات صور مختلفة:

$$Const\ Pi\ =\ 3.14159265$$
$$PBox1.Circle\ (800, 600), 400\ :\ Pbox2.Circle\ (250, 250), 200$$

$Pbox2.Circle\ Step(700,400),400$
: $Pbox3.Circle\ (800,550),700,,,,0.5$
$Pbox4.Circle\ (800,500),450,,,,2$
: $Pbox5.Circle\ (800,600),500,,Pi\ /\ 4,2\ *\ Pi\ /\ 2$
$Pbox6.Circle\ (800,600),500,,-2\ *\ Pi\ /\ 4,2\ *\ Pi\ /\ 2$
$Pbox7.Circle\ (800,500),400,,-2\ *\ Pi\ /\ 4,-4\ *\ Pi\ /\ 2$
$Pbox8.Circle\ (800,500),400,,-Pi\ /\ 4,-4\ *\ Pi\ /\ 2$



طريقة المسح Cls (Clean Screen):

تستخدم هذه الطريقة لتنظيف مربع الصورة أو الأداة التي عليها الرسم من كل ما هو عليه من رسوم متولّدة في المرحلة التنفيذية ولا نحتاج هنا إلى أي دلائل أو وسيطات. وشكلها العام هو:

$ControlName.cls$

أي أن التعليمة ستقوم بمسح أي رسوم متولّدة عليها.

تعليمة لون نقطة Point:

تستخدم هذه التعليمة لإيجاد لون نقطة ضوئية عند إحداثي معين وشكلها العام:

$ControlName.Point(x,y)$

أي تتعرف على لون النقطة التي إحداثياتها (x, y).

- ✓ يمكن أن تستخدم كتابع يحدد لون النقطة المرسومة بنفس لون نقطة إحداثياتها محددة مسبقاً.

$PBox.Pset(200,400),Point(800,1000)$

أي أخذ لون النقطة (800,1000) وتطبيقه على النقطة المطلوب رسمها بالإحداثيات (200,400).

✓ يمكن أن تستخدم لتحديد إحدى خواص أداة الرسم:

$PBox.ForeColor = PBox.Point(500,500)$

أي أخذ لون الرسوم ForeColor ضمن الأداة PBox كما هو للنقطة (500,500).

الخاصية DrawStyle:

تحدد نوع النسيج المستخدم في الرسم وهي من خصائص أداة الرسم وتأخذ إحدى القيم التالية:

القيمة	الشكل
0	خط مصمت Solid
1	خط مكون من شُرط متتابعة Dashed
2	خط منقّط Dotted
3	خط مكون من نقط وشُرط متتابعة Dash - Dot
4	خط مكون من شُرط ونقطتين على التتابع Dash – Dot – Dot
5	خط شفاف Transparent
6	خط مصمت من الداخل Inside Solid

تغيير الألوان ونسيج الملء للأشكال المرسومة:

يتم ذلك باستخدام الخصائص التالية:

- ForeColor لون الرسم.
- BackColor لون الخلفية.
- FillColor لون الملء.
- FillStyle نسيج الملء.

أداة عرض الصور Image Control:

تستخدم هذه الأداة لإظهار رسم بياني ما Graphic. وهي قادرة على إظهار الرسوم البيانية والصور من ملفات الصور النقطية Bitmaps والأيقونات Icon أو الملفات

المتحولة metafile بالإضافة إلى الملفات المحسنة advanced metafile والملفات من النوع GIF, JPEG.

Type	File Extension	Description
Gif Image	. gif	Graphics Interchange Format image type that supports 256 or fewer colors.
Bitmap Image	. bmp	Device-dependent bitmap. Image type that supports 1-bit, 4-bit, 8-bit, 16-bit and 24-bit color depths.
Bitmap Image	. dib	Device-independent bitmap that supports 1-bit, 4-bit, 8-bit, 16-bit and 24-bit color depths.
Icon Image	. ico	Icon. Image type composed of approximately 16 colors. Typically sized at 32-by-32 pixels.
Cursur image	. cur	Cursor. A special bitmap type commonly used to represent the mouse pointer.
Windows meta file and Enhanced windows meta file	. wmf . emf	Graphical images that describe a picture in terms of geometric shapes (i.e., lines, circles, etc.)
RLE Images	. rle	Run-Length Encoding. A compressed image type.
JPEG Images	. jpg	Joint Photographic Experts Group. Supports true color images and palette images.

إن شكل هذه الأداة في شريط الأدوات وفي الإطار Form كما يلي:

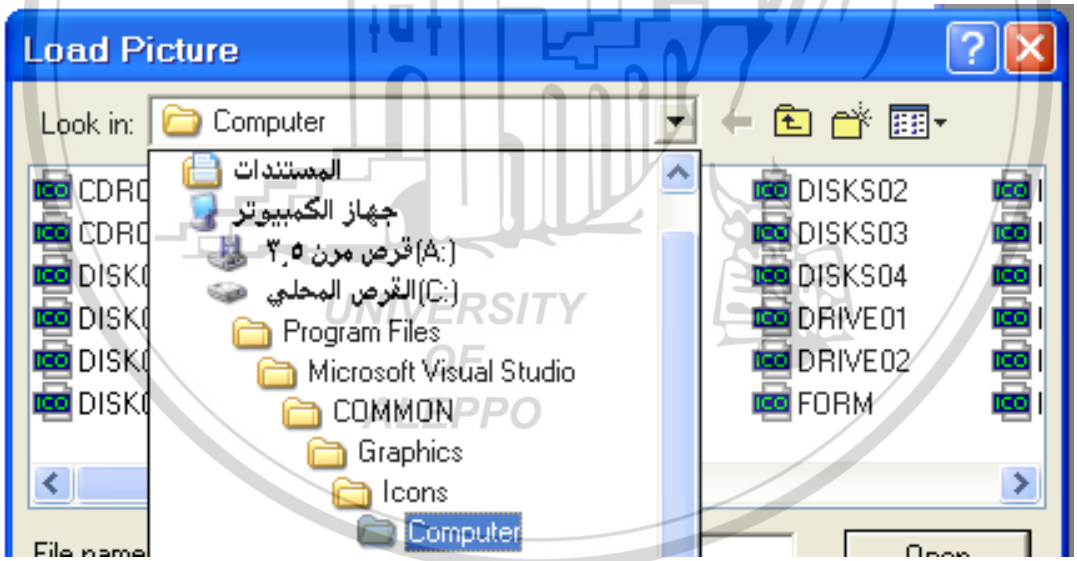
تستخدم هذه الأداة لعرض الصور فقط وبشكل مشابه لأداة الصورة Picture Box ولكنها تستخدم موارد النظام (الذاكرة مثلاً) بشكل أقل وأسرع منها لأن عدد الخصائص والأحداث المتوافرة لهذه الأداة أقل من تلك المتوافرة لأداة الصورة.

إن أغلب الخصائص الموجودة في هذه الأداة مشابهة لمثيلاتها في الأدوات الأخرى وخصوصاً أداة الصور Picture Box إلاّ أنها تتمتع ببعض الخصائص التي تميّزها عن غيرها من الأدوات مثل:

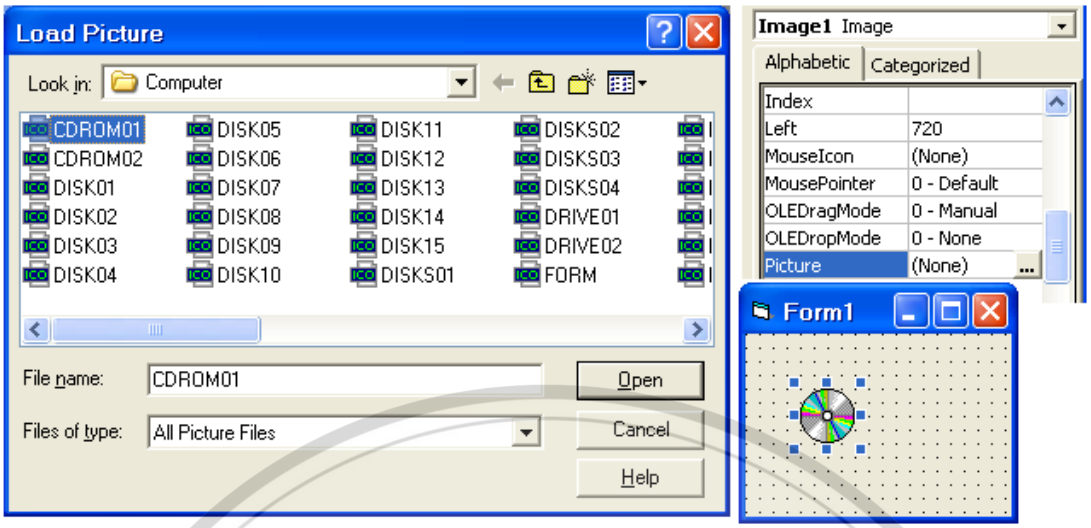
الخاصية Picture:

وهي مشابهة تماماً للخاصية Picture في أداة الصورة. تستخدم هذه الخاصية في المرحلة التصميمية (المرئية) والمرحلة التنفيذية ونستطيع من خلالها تحميل صورة وعرضها في الأداة Image بعد كتابة مسارها بشكل كامل ومفصل وصحيح.

في المرحلة المرئية يتم اختيار قيمة للخاصية Picture فيظهر مربع حوار نذهب من خلاله إلى مكان وجود الصورة.



بعد ذلك نختار الصورة أو الأيقونة المطلوبة:



في المرحلة التنفيذية يتم استخدام الكود الذي من خلاله نبين مسار الصورة بالكامل كما يلي:

$$Image1.Picture = LoadPicture(The\ Full\ Path\ of\ the\ Picture")$$

مثال:

$$Image1.Picture = LoadPicture("C:\Program\ Files \Microsoft\ Visual\ Studio\Common\Graphics\Icons \Computer\CDROM01.Ico")$$

فيتم تميل الصورة في أداة عرض الصور Image Control وتظهر بداخلها كما هو مبين في الشكل.

الخاصية Stretch:

تحدد هذه الخاصية من سيغير حجمه ليلائم الآخر (أداة عرض الصورة أم الصورة الموجودة بداخلها). تأخذ هذه الخاصية قيماً منطقية Boolean فإما أن تكون True أو تكون False.

- إذا كانت الخاصية Stretch = True فإن الصورة ستتغير بحيث يتم احتواؤها في أداة عرض الصورة بحجمها الموجود وهذا يعني أن الصورة قد تتمدد (تكبر) أو تنكمش (تصغر) لكي تصبح بحجم أداة عرض الصورة Image control.
- إذا كانت الخاصية Stretch = false وهي القيمة الافتراضية فهذا يعني أنّ أداة عرض الصورة هي التي سوف يتغير حجمها تبعاً لحجم الصورة. أي أن حجم الصورة سيبقى ثابتاً وستتغير أبعاد الأداة (زيادة ونقصاناً) لكي تستطيع احتواء الصورة بشكل دقيق.

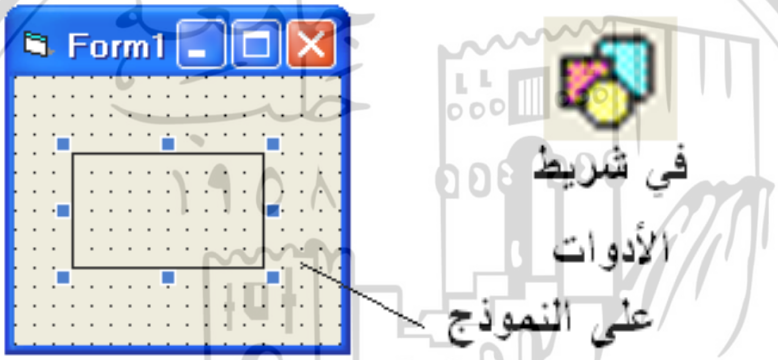
- إن الخواص المسؤولة عن أبعاد الصورة هي: Left, Top, Width, Height ولكن عند تغيير حجم الأداة تبقى الخاصيتين Left و Top بدون تغيير ويتم تغيير الخاصيتين Width و Height.

إن الخاصيتين Picture, Stretch متوفرتان في المرحلة المرئية والمرحلة التنفيذية.

أداة الأشكال Shape Control:

تستخدم هذه الأداة لإنشاء أشكال هندسية عديدة كالمستطيلات، والمربعات، والقطوع الناقصة، والدوائر، والمستطيلات ذات الدوائر المستديرة، والمربعات ذات الدوائر المستديرة.

تظهر هذه الأداة في شريط الأدوات وعلى النموذج كما يلي:



إن أغلب الخصائص الموجودة في هذه الأداة مشابهة لمثيلاتها في الأدوات الأخرى وخصوصاً أداة الصور Picture Box إلاّ أنها تتمتع ببعض الخصائص التي تميّزها عن غيرها من الأدوات مثل:

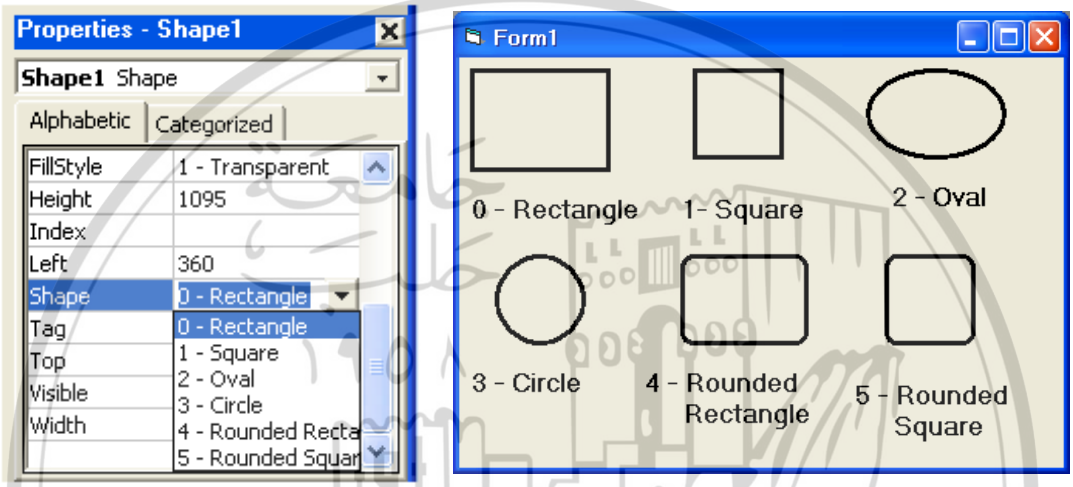
الخاصية Shape:

وهي الخاصية التي تبين نوع الشكل المرسوم داخل الأداة. وإن الأشكال التي يمكن توليدها داخل هذه الأداة هي:

Shape Property	Description	Vb Name	Vb Value
Rectangle	مستطيل	*vbShapeRectangle*	0
Square	مربع	*vbShapeSquare*	1
Oval	قطع ناقص (إهليلج)	*vbShapeOval*	2
Circle	دائرة	*vbShapeCircle*	3

Rounded Rectangle	مستطيل ذو زوايا مستديرة	*vbShapeRoundedRectangle*	4
Rounded Square	مربع ذو زوايا مستديرة	*vbShapeRoundedSquare*	5

يمكن تغيير هذه الخاصية واختيار الشكل المطلوب في المرحلة المرئية بأن نأخذ الخاصية Shape ونأخذ الشكل المطلوب كما يلي:



بينما في المرحلة التنفيذية فباستخدام الأوامر الكودية كما يلي:

لرسم مستطيل $Shape1.Shape = 0$

لرسم مستطيل $Shape1.Shape = VbShapeRectangle$

لرسم دائرة $Shape1.Shape = 3$

لرسم دائرة $Shape1.Shape = VbShapeCircle$

خصائص الألوان Colors Properties:

وهي ثلاث خصائص:

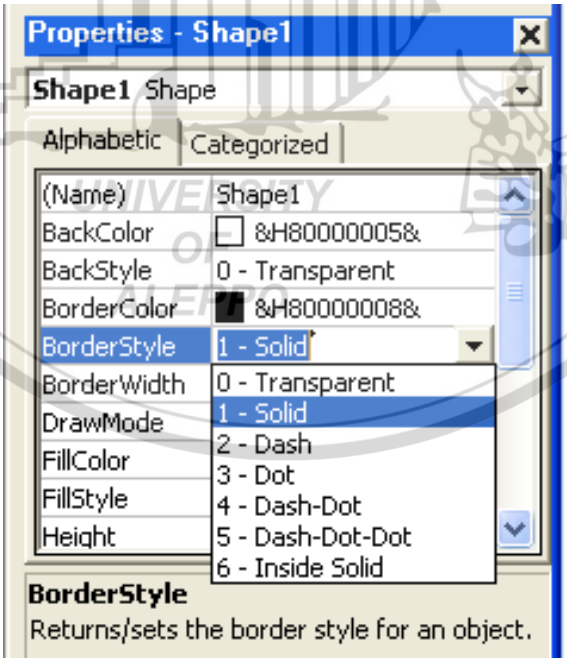
- لون الإطار BorderColor.
- لون الملء FillColor.
- لون الخلفية BackColor.

خصائص النسيج Style Properties:

وهي ثلاث خصائص أيضاً:

- نسيج الإطار BorderStyle. وتأخذ إحدى القيم التالية:

Vb Constant	Type	الشكل	القيمة
VbTransparent	*Transparent*	خط شفاف	0
VbBSSolid	*Solid*	خط مصمت وهو الافتراضي	1
VBSDash	*Dash*	خط مكون من شرط	2
VbBSDot	*Dot*	خط مكون من نقط	3
VbBSDashDot	*Dash – Dot*	خط مكون من شُرط ونقط متتابعة	4
VbBSDashDotDot	*Dash – Dot – Dot*	خط مكون من شُرط ونقطتين على التتابع.	5
VbBSInsideSolid	*Patterrn is Inside Solid*	خط مصمت من الداخل	6

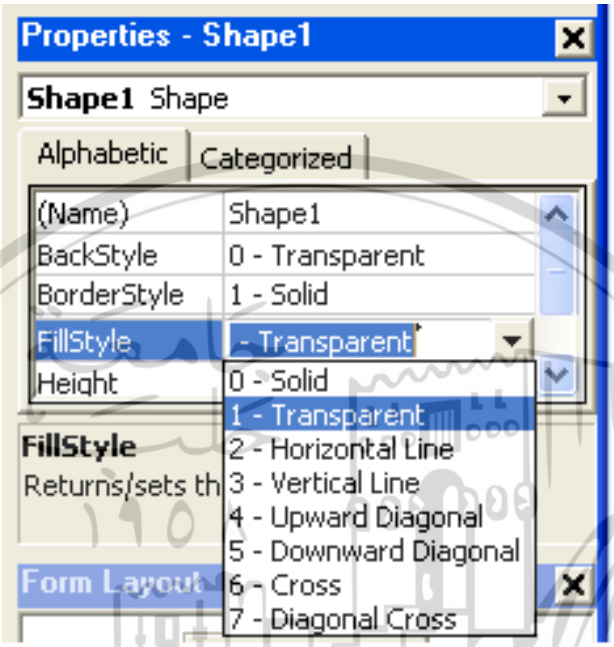


ومن الواضح أنه يمكن تغيير هذه الخاصية في المرحلة المرئية وفي المرحلة التنفيذية باستخدام الأوامر الكودية كما يلي:

$$Shape1.BorderStyle = 0$$
$$Shape1.BorderStyle = vbBSSolid$$

- نسيج الملء FillStyle.

وتأخذ هذه الخاصية في المرحلة المرئية إحدى القيم التالية:



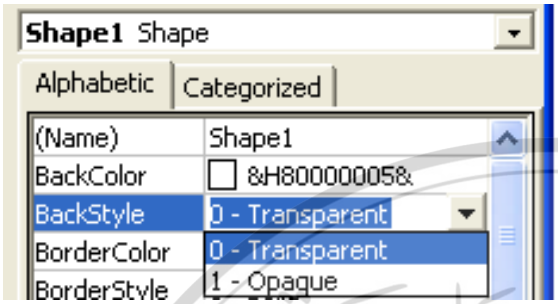
وتظهر القيم الكودية كما في الجدول:

Type	Value	الشكل
Solid	0	لون خالص مصمت
Transparent	1	لون شفاف غير مرئي (القيمة الافتراضية)
Horizontal Lines	2	خطوط أفقية
Vertical Lines	3	خطوط عمودية
Upward Diagonal	4	خطوط مائلة للأعلى
Downward Diagonal	5	خطوط مائلة للأسفل
Cross	6	خطوط رأسية وأفقية متقاطعة
Diagonal Cross	7	خطوط مائلة متقاطعة

ومن الواضح أنه يمكن تغيير هذه الخاصية في المرحلة المرئية وفي المرحلة التنفيذية باستخدام الأوامر الكودية كما يلي:

$$Shape1.FillStyle = 1$$
$$Shape1.FillStyle = vbTransparent$$

- نسيج الخلفية BackStyle.



✓ Transparent – 0 أو نسيج شفاف.

✓ Opaque – 1 أو نسيج مصمت من الداخل.

ومن الواضح أنه يمكن تغيير هذه الخاصية في المرحلة المرئية وفي المرحلة التنفيذية باستخدام الأوامر الكودية كما يلي:

$$Shape1.backStyle = 1$$

أداة رسم المستقيم Line Control:

يمكن استخدام هذه الأداة لرسم مستقيمات على نافذة واجهة المشروع Form. تستخدم هذه الأداة في المرحلة التصميمية وفي المرحلة التنفيذية بدلاً من طريقة الرسم Line Method.

تظهر هذه الأداة في شريط الأدوات وعلى النموذج كما يلي:

إن أغلب الخصائص الموجودة في هذه الأداة مشابهة لمثيلاتها في الأدوات الأخرى وخصوصاً أداة الصور Picture Box إلاّ أنها تتمتع ببعض الخصائص التي تميّزها عن غيرها من الأدوات مثل:

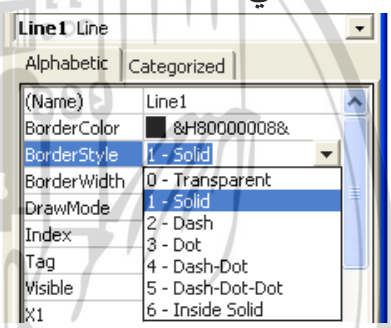
الخاصية BorderStyle وتأخذ إحدى القيم التالية:

القيمة	الشكل	Type	Vb Constant
0	خط شفاف	*Transparent*	*VbTransparent*
1	خط مصمت وهو الافتراضي	*Solid*	*VbBSSolid*

2	خط مكون من شرط	*Dash*	*VBSDash*
3	خط مكون من نقط	*Dot*	*VbBSDot*
4	خط مكون من شُرط ونقط متتابعة	*Dash – Dot*	*VbBSDashDot*
5	خط مكون من شُرط ونقطتين على التتابع.	*Dash-Dot-Dot*	*VbBSDashDotDot*
6	خط مصمت من الداخل	*Patterrn is Inside Solid*	*VbBSInsideSolid*

ويمكن تغييرها في المرحلة المرئية وفي المرحلة التنفيذية باستخدام الأوامر الكودية كما يلي:

$$Line1.BorderStyle = 1$$
$$Line1.BorderStyle = VbSDash$$



الخاصية BorderWidth:

وتحدد هذه الخاصية سماكة المستقيم المرسوم على نافذة واجهة المشروع. ومن خلالها نستطيع الفصل بين الأدوات المختلفة. تأخذ هذه الخاصية إحدى القيم المحصورة في المجال بين $1 - 32767$. عندما تأخذ هذه الخاصية القيمة 1 أي:

$$Line1.BorderStyle = 1$$

عندها نستطيع رؤية تأثير الخصائص الأخرى أما إذا أخذت الخاصية قيمة أكبر من الواحد فإن بعض خصائص هذه الأداة سيتم تجاهلها أو سوف لن يتم تنفيذها بشكل صحيح.

الخاصية BorderColor:

تمكننا من تغيير لون الخط المرسوم إلى أحد الألوان الممكنة في لغة الفيجوال بيزك.

الخاصية Visible:

تمكننا هذه الخاصية من إخفاء المستقيم المرسوم عند عدم الحاجة إليه وإظهاره في اللحظات التي يصبح وجوده فيها ضروريّاً. تأخذ هذه الخاصية قيمة منطقية فإما أن تكون $True$ أو $False$.

خصائص موقع ومكان الأداة:

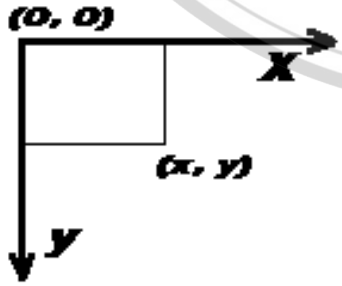
وهي الخصائص المتعلقة بمكان وجود هذا المستقيم. ومن الواضح أن هذه الخصائص هي نقطتي بداية ونهاية المستقيم والتي إحداثيتيهما (x1, y1) لنقطة البدء و(x2, y2) لنقطة النهاية.

مقياس الرسم Scale

تستخدم هذه التعليمة لنقل مركز الإحداثيات من الركن العلوي الأيسر إلى مكان ما من الإطار أو أي أداة أخرى من الأدوات (غالباً ما تكون أدوات الرسم وخصوصاً الأداة Picture Box).

عند نقل المركز نحدّد إحداثيات مركز الإحداثيات الجديد بالنسبة للمركز القديم أو نحدد موقعة انطلاقاً من أربع خصائص هي: ScaleTop, ScaleLeft, ScaleWidth, ScaleHeight. حيث ستُحدّد هذه الإحداثيات بُعد الركن العلوي اليساري وبعد الركن السفلي اليميني عن مركز الإحداثيات الجديد. ويتحدد الاتجاه الموجب للمحاور من خلال إعطاء قيم موجبة أو سالبة بالنسبة لهذه القيم.

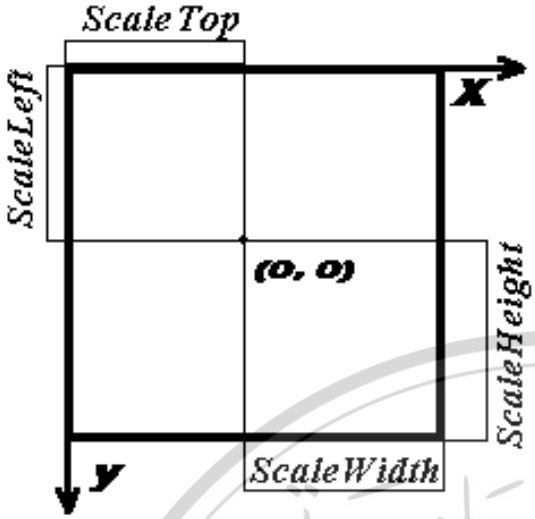
إتجاه المحاور الإحداثية العادية في أدوات VB للأسفل واليمين كما يلي:



x – لليمين.

y – للأسفل.

(0, 0) – هو الركن العلوي اليسر.



الآن إذا وضعنا مركز الإحداثيات الجديد في مكان ما من الأداة فإن الخصائص الأربعة السابقة ستحدّد بُعد الركن العلوي الأيسر والسفلي اليمن عن المركز الجديد كما يلي:

(0, 0) – مركز الإحداثيات الجديد.

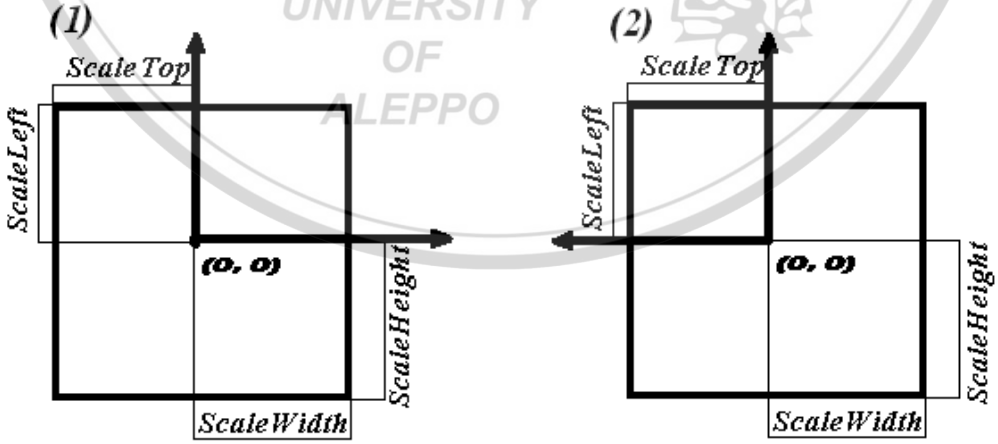
ويكون الأمر كما يلي:

$$ControlName.Scale\ (scaleTop, ScaleLeft) - (ScaleWidth, ScaleHeight)$$

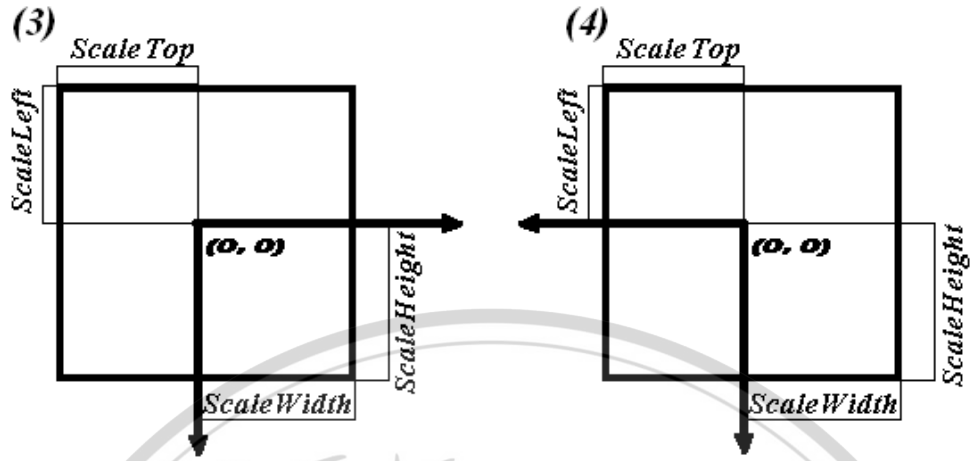
الآن تحديد اتجاه المحاور الجديدة يتم من خلال توضيع هاتين النقطتين بالنسبة للمركز الجديد (أعلى وأسفل) أو (يمين ويسار) وبما أن اتجاه المحاور القديمة لليمين (x) وللأسفل (y) لذا يمكن رسم المحاور افحداثية بالحالات الأربع كما يلي:

$$ControlName.Scale(-ScaleTop, ScaleLeft) - (ScaleWidth, \quad -ScaleHeight)$$
$$ControlName.Scale(ScaleTop, ScaleLeft) - (-ScaleWidth, \quad -ScaleHeight)$$
$$ControlName.Scale(-ScaleTop, -ScaleLeft) - (ScaleWidth, \quad ScaleHeight)$$
$$ControlName.Scale(ScaleTop, -ScaleLeft) - (-ScaleWidth, \quad ScaleHeight)$$

تظهر نتيجة الكودين الأول والثاني كما يلي:



تظهر نتيجة الكودين الثالث والرابع كما يلي:



الآن إذا أردنا رسم مستقيم ودائرة حسب الإحداثيات الجديدة بحيث يكون واقعاً في الربع الأول من كل منهما لذلك سنكتب الأوامر الكودية كما يلي:

$PBox.Scale\ (-500, 250) - (500, -250)$

1) $PBox.Line\ (0,0) - (250, 250)\ , VbGreen$
   $PBox.Circle\ (0,0)\ , 200\ , VbRed$

الأمر الأول يقوم بنقل مركز الإحداثيات إلى مكان جديد يبعد بمقدار x = 500 و y = 250 عن الركن الأيسر العلوي.

بما أن x فيه سالبة لذا اتجاه محور الإحداثيات معاكس لمكان وجود هذه النقطة وبما أن النقطة في اليسار لذا اتجاه المحور إلى اليمين.

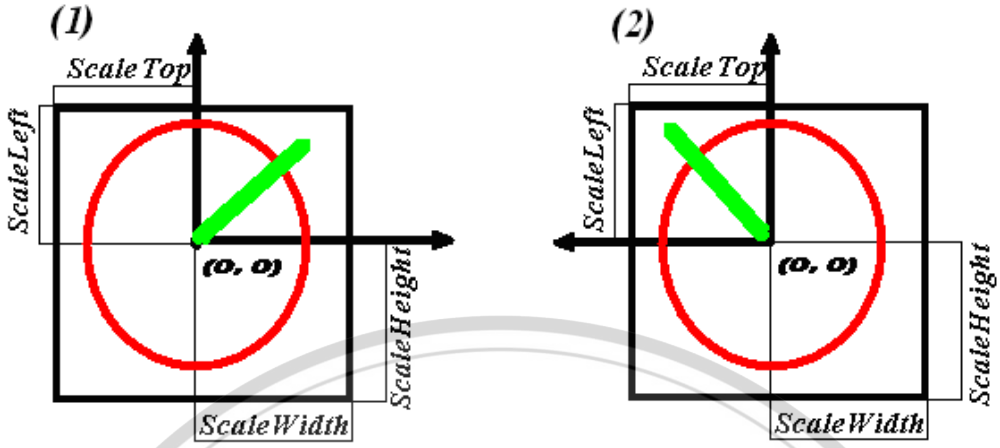
بما أن y فيه موجبة لذا اتجاه محور الإحداثيات موافق لمكان وجود هذه النقطة وبما أن النقطة في الأعلى لذا اتجاه المحور إلى الأعلى.

تتم المقارنة نفسها بالنسبة للركن الأيمن السفلي.

الأمر الثاني يقوم برسم خط يبدأ من النقطة التي إحداثياتها x = 0 و y = 0 وينتهي في النقطة التي إحداثياتها x = 250 و y = 250 ولونه أخضر.

الأمر الثالث يقوم برسم دائرة مركزها النقطة التي إحداثياتها x = 0 و y = 0 ونصف قطرها r = 200 ولونها أحمر.

عند تنفيذ الأمر سنجد أن المستقيم والدائرة سيظهران في كل شكل كما يلي:



$PBox.Scale\ (500, 250) - (-500, -250)$

2) $PBox.Line\ (0, 0) - (250, 250)\ , VbGreen$
$PBox.Circle\ (0, 0)\ , 200\ , VbRed$

الأمر الأول يقوم بنقل مركز الإحداثيات إلى مكان جديد يبعد بمقدار x = 500 و y = 250 عن الركن الأيسر العلوي.

بما أن x فيه موجبة لذا اتجاه محور الإحداثيات موافق لمكان وجود هذه النقطة وبما أن النقطة في اليسار لذا اتجاه المحور إلى اليسار.

بما أن y فيه موجبة لذا اتجاه محور الإحداثيات موافق لمكان وجود هذه النقطة وبما أن النقطة في الأعلى لذا اتجاه المحور إلى الأعلى.

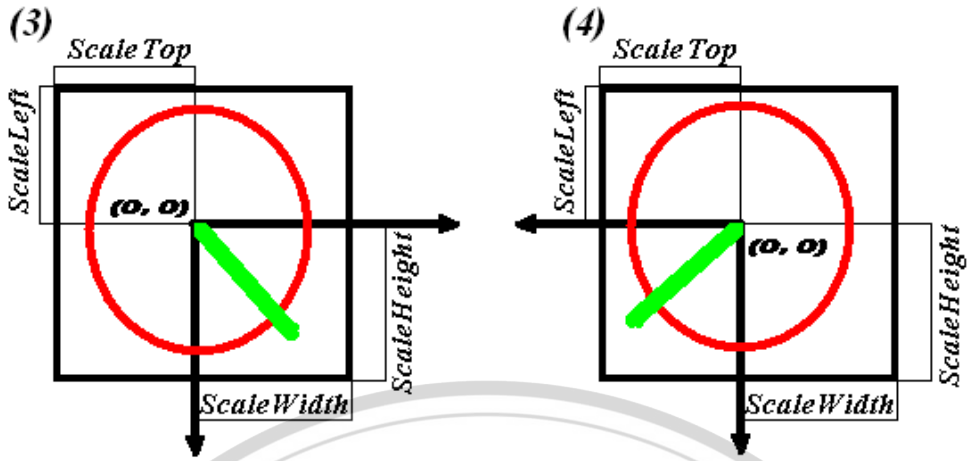
$PBox.Scale\ (-500, -250) - (500, 250)$

3) $PBox.Line\ (0, 0) - (250, 250)\ , VbGreen$
$PBox.Circle\ (0, 0)\ , 200\ , VbRed$

الأمر الأول يقوم بنقل مركز الإحداثيات إلى مكان جديد يبعد بمقدار x = 500 و y = 250 عن الركن الأيسر العلوي.

بما أن x فيه سالبة لذا اتجاه محور الإحداثيات معاكس لمكان وجود هذه النقطة وبما أن النقطة في اليسار لذا اتجاه المحور إلى اليمين.

بما أن y فيه سالبة لذا اتجاه محور الإحداثيات معاكس لمكان وجود هذه النقطة وبما أن النقطة في الأعلى لذا اتجاه المحور إلى الأسفل.



$PBox.Scale\ (500, -250) - (-500, 250)$

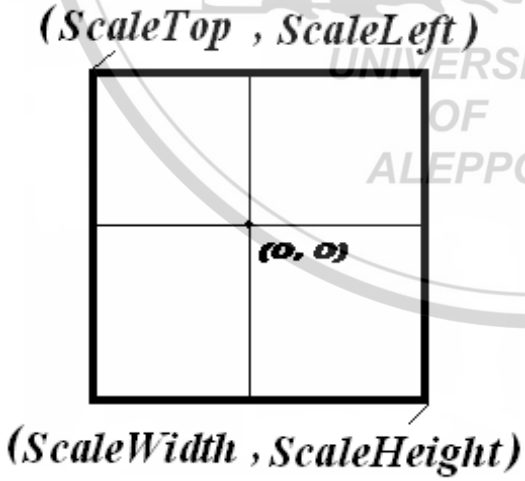
4) $PBox.Line\ (0,0) - (250, 250)\ , VbGreen$
$PBox.Circle\ (0,0)\ , 200\ , VbRed$

الأمر الأول يقوم بنقل مركز الإحداثيات إلى مكان جديد يبعد بمقدار x = 500 و y = 250 عن الركن الأيسر العلوي.

بما أن x فيه موجبة لذا اتجاه محور الإحداثيات موافق لمكان وجود هذه النقطة وبما أن النقطة في اليسار لذا اتجاه المحور إلى اليسار.

بما أن y فيه سالبة لذا اتجاه محور الإحداثيات معاكس لمكان وجود هذه النقطة وبما أن النقطة في الأعلى لذا اتجاه المحور إلى الأسفل.

ملاحظة:



أي إحداثية موجبة لركن من الأركان ستكون نظيرتها سالبة بالنسبة للركن الآخر والعكس صحيح.

على (x) نجد أن ScaleTop و ScaleWidth قيمتان متعاكستان.

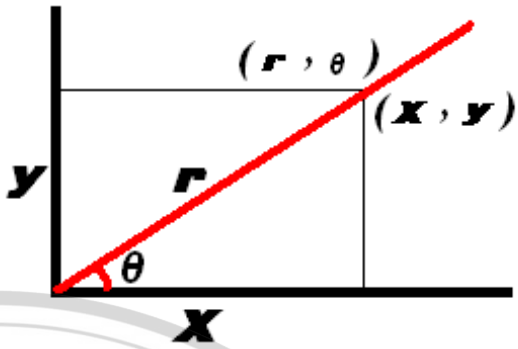
على (y) نجد أن ScaleLeft و ScaleHeight قيمتان متعاكستان.

العلاقة بين الإحداثيات القطبية والإحداثيات الديكارتية:

$$r^2 = x^2 + y^2 \ , \ r = \sqrt{x^2 + y^2}$$

$$x = r.Cos(\theta) \ , \ y = r.Sin(\theta)$$

$$tn(\theta) = \frac{y}{x} \ , \theta = Atn\left(\frac{y}{x}\right)$$



الفصل العاشر
المؤقت
Timers

مقدمة Introduction:

المؤقت أداة تجعل البرنامج يؤدي عملاً تلقائياً معيناً وعلى فترة زمنية معينة ودون تدخل من المستخدم. يعمل في خلفية المشروع مولداً حدثاً ما أو أحداثاً حسب الفترة الزمنية التي نختارها. يستخدم المؤقت بشكل كبير في الكائنات الرسومية عندما يتطلب الأمر تحديث الشاشة وتغذية مضمونها بأشكال وأشياء جديدة وبفواصل زمنية معينة.

يؤخذ المؤقت من شريط الأدوات ويظهر على الإطار كما يلي:

في شريط الأدوات على الإطار (الشكل الافتراضي)

من أهم ميزات زر المؤقت أنه غير قابل للتكبير أو التصغير وذلك كونه لا يحتوي على خاصيتي الارتفاع والعرض. ويعمل المؤقت في خلفية الشاشة ولا يظهر على الإطلاق عند تنفيذ البرنامج ولذلك ليس من المهم مكان وجوده على إطار البرنامج.

خصائص المؤقت:

للمؤقت مجموعة غير كبيرة من الخصائص من أهمها خاصيتان أساسيتان هما:

١. المسافة أو الفاصل الزمني $Interval$.
٢. خاصية التفعيل والتعطيل $Enabled$.

١. الخاصية *Enabled*:

إن الخاصية $Enabled$ تدل على نشاط المؤقت وتأخذ إحدى القيمتين فإما أن تكون نشطة $True$ أو تكون معطلة $False$. بالحالة العادية يكون المؤقت نشطاً أي أن الخاصية $Enabled = True$ وللمبتدئين ينصح بجعلها معطلة مع إمكانية التحكم بها

عن طريق الأوامر الكودية. في هذه الحالة من الأفضل أخذ زر أوامر وجعل خاصيته $Caption$ تظهر عبارة تعطيل عندما يكون فعالاً وتفعيل عندما يكون معطلاً.

عندما تكون الخاصية $Enabled = True$ فمعنى ذلك أن المؤقت فعال وسيقوم بتكرار الحدث المطلوب منه كودياً كل فترة زمنية معينة أي أن VB سيقوم بتوليد الحدث $Event$ الموجود في الإجراء الكودي للمؤقت $Timer\ Procedure$ والذي سيكون بالشكل:

```
Private Sub timExample_Timer( )
  أمر كودي يتم تنفيذه كل مدة زمنية مقدراً بالميللي ثانية
End Sub
```

٢. الخاصية $Interval$:

وهي خاصية الفاصل الزمني وهي عبارة عن المسافة الزمنية التي خلالها يكرر المؤقت الحدث المطلوب منه وتقدر المسافة الزمنية بواحدة الميللي ثانية وتتراوح بين $[msec]$ 65535 ... 0. فإذا كان الميللي ثانية مساوياً لـ 1/1000 ثانية فمعنى ذلك أنه يمكن أن نقوم بتوليد الأحداث على مسافة زمنية تبدأ من الصفر وحتى $[sec]$ 65.

فإذا أخذت الخاصية $Interval$ القيمة 0 فمعنى ذلك أن المؤقت معطل ويبدو هذا وكأننا جعلنا قيمة الخاصية $Enabled = False$. أما إذا أردنا تفعيل المؤقت N مرة في الثانية فمعنى ذلك أننا سنقوم بتقسيم القيمة 1000 على عدد المرات N ونضعها في الخاصية $Interval$. حيث N عدد مرات تكرار الحدث في الثانية:

$$Interval = (1000 \ / \ N)$$

مثلاً إذا أردنا تكرار الحدث 4 مرات في الثانية فمعنى ذلك أننا سنأخذ:

$$Interval = (1000 \ / \ 4) = 250$$

أي أنه كل 250 ميللي ثانية سيقوم المؤقت بتوليد الحدث.

وإذا أردنا تكرار الحدث مرة واحدة كل N ثانية نكتب:

$$Interval = (1000 * \ N)$$

N عدد مرات الثواني التي يتكرر فيها الحدث كل مرة.

الفصل الحادي عشر
محرر القوائم
Menu Editor

مقدمة:

يمكن في كثير من الأحيان أن يقوم البرنامج بعدد كبير من العمليات ولذلك استعمال عدد كبير من الكائنات (العناصر) وهذا ما قد يسبب في بعض الأحيان ضغط كبير على الـ ***Form*** من حيث المعلومات وقد يجعل هذا واجهة البرنامج تفقد جماليتها لذا نجد من الضروري في هكذا حالات استخدام طريقة أفضل تساعد في عملية التبويب بشكل أحسن بالإضافة إلى الجمالية التي تضفيها على واجهة البرنامج أو الإطار. إن من أفضل الطرق لحل هذه المشكلة هي إستخدام أشرطة القوائم والأدوات.

إنشاء أشرطة القوائم *Menus*:

لإنشاء قائمة أو شريط قوائم أو أشرطة قوائم نستخدم الأمر ***Menu Editor*** من القائمة ***Tools*** فيظهر مربع الحوار ***Menu Editor*** والذي سيساعدنا على تصميم القوائم المطلوبة.

مربع الحوار *Menu Editor* :

يقسم مربع الحوار إلى منطقتين :

١) منطقة خصائص عناصر التحكم.

٢) منطقة بنود عناصر التحكم.

١. منطقة خصائص عناصر التحكم:

- *Caption*: نكتب الإسم المرئي والذي سيظهر على الإطار، ويمكن أن يكون باللغة العربية أو بالإنكليزية ويمكن أن يحتوي على فراغات وما شابه ذلك.
- *Name*: الإسم الكودي والذي سيستخدم أثناء كتابة البرمجة الكودية، ويجب أن يكون باللغة الإنكليزية حكماً.

- $Shortcut$: مفاتيح الإختزال المتعلقة بهذا البند.

يستخدم المربع ***Shortcut*** لإضافة مفاتيح الإختزال إلى القوائم وهذا ما سيمكننا من إستخدام هذه المفاتيح من لوحة المفاتيح بالإضافة إلى إمكانية تنفيذ هذه الأوامر عن طريق النقر على القائمة بإستخدام الماوس أيضاً.

- $Checked$: لإمكانية تحديد القائمة أثناء التنفيذ وتحدد بأن تظهر بجانبها الإشارة √.
- $Enabled$: لتمكين ظهور العنصر بشكل فعال (لونه أسود) أو معطل (لونه باهت).
- $Visible$: تمكين ظهور العنصر أثناء التنفيذ أم لا.

٢. منطقة بنود عناصر التحكم:

يوجد في هذه المنطقة مجموعة أزرار أهمها:

- أسـهم تحريك القوائم للأعلى والأسـفل لترتيب ظهورها في الإطار ولليمين واليسـار من أجل جعلها أساسية أو ضمنية (فرعية – أي قائمة داخل قائمة).
- $Next$: للإنتقال إلى القائمة التالية.
- $Insert$: لإدراج قائمة جديدة وتدرج دوما إلى الأعلى من القائمة المختارة.
- $Delete$: لحذف قائمة.

منطقة التسمية والتي تظهر فيها أسماء القوائم كما تم إنشاؤها بحيث نستدل على كيفية توضع العناصر وعلى وجود مفاتيح الإختزال فيها أو لا.

يمتلك كل بند من بنود القوائم (بغض النظر عن مستواه) على حادثة نقر ***Click Event***، تنفّذ هذه الحادثة لدى النقر على هذه القائمة (أي لدى إختيارها).

رغم أن الحادثة تدعى ***Click*** (أي النقر) إلا أنها تنفّذ سواء تم إختيار القائمة بزر الماوس أم عن طريق إستخدام مفاتيح الإختزال من لوحة المفاتيح.

- لا يوفر VB ولا النظام ويندوز نفسه مفاتيح إختزال باللغة العربية.
- لايمكن اختيار الخط الفاصـــل أثناء تنفيذ البرامج ولهذا فإنه لايمكن تطبيق أي إجراء عليه والغاية من وجوده فقط الناحية الجمالية. يمكن أن يتواجد عدة خطوط فاصـــلة وحسبما نريد.
- يجب تسمية كل بنود القائمة حتّى ولو لم تقم بأي فعل (مثل فواصل الأسطر).

يمكن أيضا تنفيذ أمر القائمة وذلك بإستخدام مفاتيح تسمى المفاتيح الساخنة أو Hot Keys والتي بإستخدام التركيب (المفتاح الساخن + مفتاح Alt).

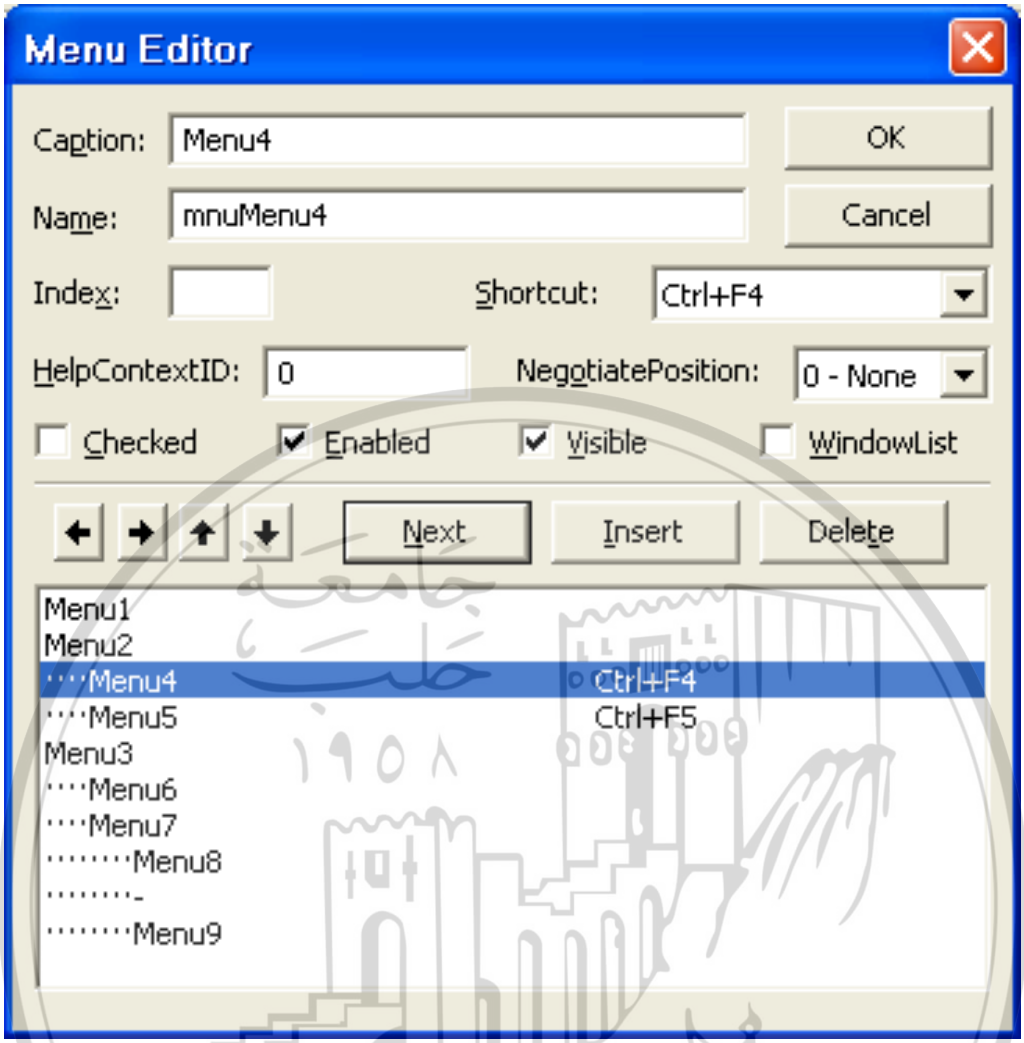
البرمجة الكودية للعناصر:

بعد الانتهاء من عملية التنسيق المرئي لأشرطة القوائم (أي بعد كتابة مواصفاتها وأسمائها سنتمكن من كتابة النصوص التي تدل على الأوامر التي سيقوم البرنامج بتنفيذها أثناء النقر على هذه القوائم.

مثال:

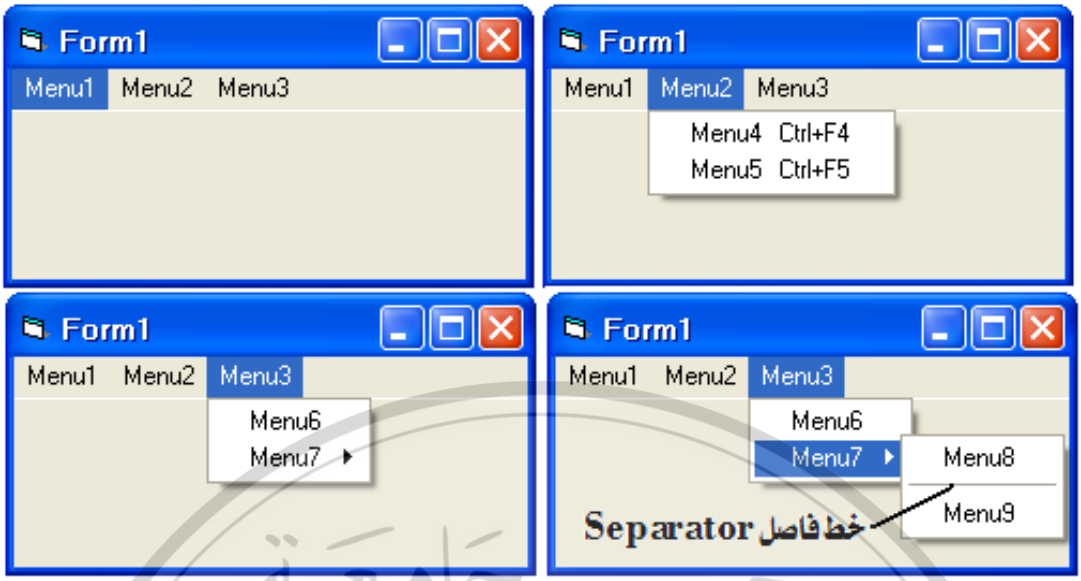
نريد إنشاء شريط قوائم يتألف من ثلاث قوائم وهي $Menu1, Menu2, Menu3$ بحيث تتألف القائمة $Menu2$ من قائمتين هما $Menu4, Menu5$ كما وتتألف القائمة $Menu3$ من قائمتين هما $Menu7, Menu8$ وتتألف القائمة الفرعية $Menu7$ بدورها من قائمتين فرعيتين $Menu8, Menu9$ والذي يفصل بينهما خط فاصل. يضاف للقائمتين $Menu4, Menu5$ مفاتيح إختزال هي $[Ctrl + F4], [Ctrl + F5]$.

من القائمة $Tools$ نأخذ القائمة الفرعية $Menu\ Editor$ فيظهر مربع الحوار $Menu\ Editor$ والذي سنضع فيه مواصفات القوائم:



نأخذ القائمة ***Menu1*** ثم ***Next*** ونأخذ القائمة ***Menu2*** ولكي ننشئ بداخلها قائمتين فرعيتين سنأخذ Next ثم السهم اليميني ليزيح القائمة لليمين للدلالة على فرعيتها ثم نكتب ***Menu4*** ثم نضيف مفتاح إختزال [***Ctrl + F4***] لها ثم نأخذ ***Next*** ونكتب ***Menu5*** ثم نضيف مفتاح إختزال [***Ctrl + F5***] لها ثم نأخذ ***Next*** وللعودة إلى قائمة أعلى نأخذ السهم اليساري ثم نكتب ***Menu3*** ثم نأخذ ***Next*** ثم السهم اليميني ثم نكتب القائمتين ***Menu6, Menu7*** ثم ***Next*** ومن ثم نكتب القائمة ***Menu8*** بعد أن نزيح السهم لليسار مرة ثانية لليمين وبعد ذلك نأخذ خط فاصل بأن نكتب العلامة – (إشارة الناقص العادية) في مربع ***Caption*** ومن ثم ***Next*** وفي النهاية القائمة ***Menu9*** ثم Ok.

بعد ذلك ستظهر القوائم ضمن الإطار الرئيسي كما يلي:



القوائم المعطلة والخاصية Enabled :

أثناء تنفيذ بعض البرامج من الممكن أن يبدأ عمل البرنامج بأحد الخيارات الفعالة (أي بشكل إفتراضي سيعمل هذا الأمر مع بدء الإيعاز بتنفيذ البرنامج) فإذا كان هناك قائمة تدل على عمل هذا الخيار فيجب أن تظهر القائمة بشكل معطل (لون باهت) كون البرنامج يعمل منذ البداية مع هذا الخيار ولا داعي لإختياره كونه مختاراً بشكل آلي. مثلا قد نظهر النافذة بلون معين أو بحجم معين أو ... الخ.

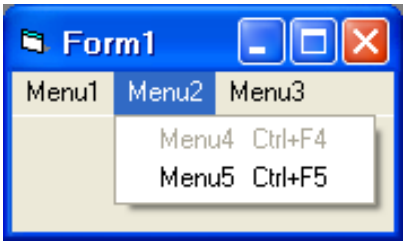
في هذه الحالة يجب تعطيل هذه القائمة كودياً مع البقاء على الخيارات الأخرى المناقضة فعالة. يتم تعطيل القائمة بأن نجعل الخاصية ***Enabled*** لهذه القائمة غير فعالة أي ***False*** وهي في الحالة الإفتراضية ستكون ***True***.

يجب أن تكون الخيارات الأخرى مناقضة وبشكل إفتراضي أيضاً (أي أثناء بدء التنفيذ).

الآن عند إختيار أحد الخيارات النشطة في القوائم فمن المؤكد أن الخيار الذي تم تنشيطه يجب أن يتعطل والخيار الذي كان معطّلاً سيتم تفعيله وهكذا ...

من الخصائص الممكن تغييرها لون خلفية الإطار ***BackColor*** وحجم النافذة ***WindowState***. فمثلا إذا أردنا تعطيل عمل القائمة ***Menu4*** أثناء إختيار القائمة ***Menu2*** نقوم بإضافة سطر كودي يقوم بتعطيل القائمة ***Menu4*** ويمنعها من الظهور:

$$Menu4.Enabled = False$$



الآن أثناء تنفيذ البرنامج وعند انتقاء القائمة ***Menu2*** ستظهر القائمة ***Menu4*** غير نشطة كما في الشكل التالي:

لإظهارها عند أي خيار آخر لا بد من إضافة السطر الكودي التالي إلى هذا الخيار:

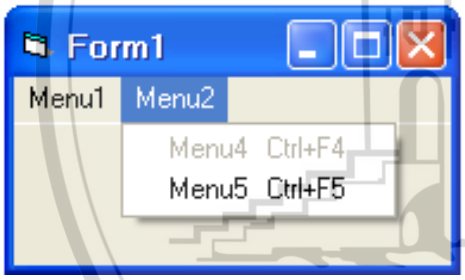
$$Menu4.Enabled = True$$

كل الخيارات المطلوب تعطيلها إفتراضيّاً لا بد من وجودها في السطر الكودي للنموذج ***Form*** أي أنها ستكون مكتوبة في الإجراء ***Form_Load*** ( ).

جعل أحد العناصر مخفيّاً (غير مرئي) والخاصية Visible:

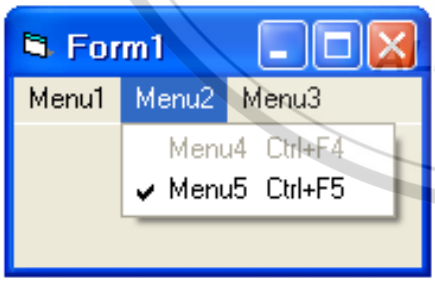
قد نحتاج إلى إخفاء أحد بنود القائمة كليّاً في بعض الأحيان وليس مجرد جعله باهتاً. لأجل ذلك نأخذ الخاصية ***Visible*** ونعطيها القيمة ***False***. فإذا أردنا إخفاء البند ***Menu3*** مثلاً سنقوم بإضافة الأمر ***False*** والذي يعطل ظهور هذه القائمة:

$$Menu3.Visible = False$$



معنى ذلك أن القائمة ستختفي عند التنفيذ وستزاح كل البنود التي تحتها إلى الأعلى وستظهر القائمة وكأن هذا البند غير موجود من الأصل.

علامات الاختيار *Check Marks* والخاصية *Checked*:



نحتاج في بعض الأحيان إلى وضع إشارة على أحد البنود للدلالة على إختياره مثلاً، لأجل ذلك نستخدم ***Check Marks*** أو ما يسمّى علامات الاختيار لإنجاز هذه الغاية.

تستخدم الخاصية ***Check*** لوضع أو رفع علامة الاختيار. فعند إسناد القيمة True للخاصية ***Checked*** التابعة لأحد عناصر تحكم القائمة فإن علامة الإختيار √ ستظهر بجانبه، أما ***False*** فسيؤدي إلى إزالتها.

$$Menu5.Checked = True$$
$$Menu5.Checked = False$$

- بشـكل إفتراضــي فإن كل القائم سـتظهر غير مختارة أثناء التنفيذ إلاّ إذا تم التنويه غير ذلك كودياً.

يوجد في مربع الحوار Menu Editor خيار Check Box والمسمى Checked والذي باختياره سيظهر البند أو القائمة وبشكل إفتراضي مختاراً إلا إذا تم إزالة الإختيار كودياً بإستخدام الخاصية False.

Property	***Description***
Checked	Indicates whether a menu item has a checkmark next to it. Generally, you'll add checkmarks to menu options that perform on or off actions, such as a View menu that contains a Highlighted command. The checkmark appears when you, at design time or through code, set the menu item's Checked property to True. The checkmark goes away (indicating that the item is no longer active or selected) when you set the Checked property to False.
HelpContextID	This is a code that matches a help file description if and when you add help files to your application.
Index	If you create a menu control array rather than name individual menu items separately, this `Index` property specifies the menu item's subscript within the control array.
Shortcut	This is a drop-down list of Ctrl+ keystroke access keys that you can add to any pull-down menu item.
Window List	Specifies whether the menu item applies to an advanced application's MDI (multiple-document interface) document. The menus that you create for this book don't require the use of MDI features.

جامعة حلب
١٩٥٨
UNIVERSITY
OF
ALEPPO

الفصل الثاني عشر
أحداث الماوس
Mouse Events

مقدمة:

إن فلسفة البرمجة المسيَّرة بالأحداث *Event Driven Programming* تقتضي عملية تنفيذ الأكواد عند حالات معينة تعرف بوقوع الأحداث أو حدوث الأحداث. فالأكواد التي نضعها لن يتم تنفيذها إلا عند وقوع الحدث عليها. والأحداث عبارة عن إجراءات *Sub's* أسماؤها تتبع الصيغة التالية: اسم الكائن _ الحدث

Form_ Click ( )
Command 1 _Click ( )

ملاحظة:

يستخدم التعبير تفجير (حدوث – وقوع – اندلاع) الحدث *Fire Event* عوضاً عن التعبير استدعاء الحدث لأن:

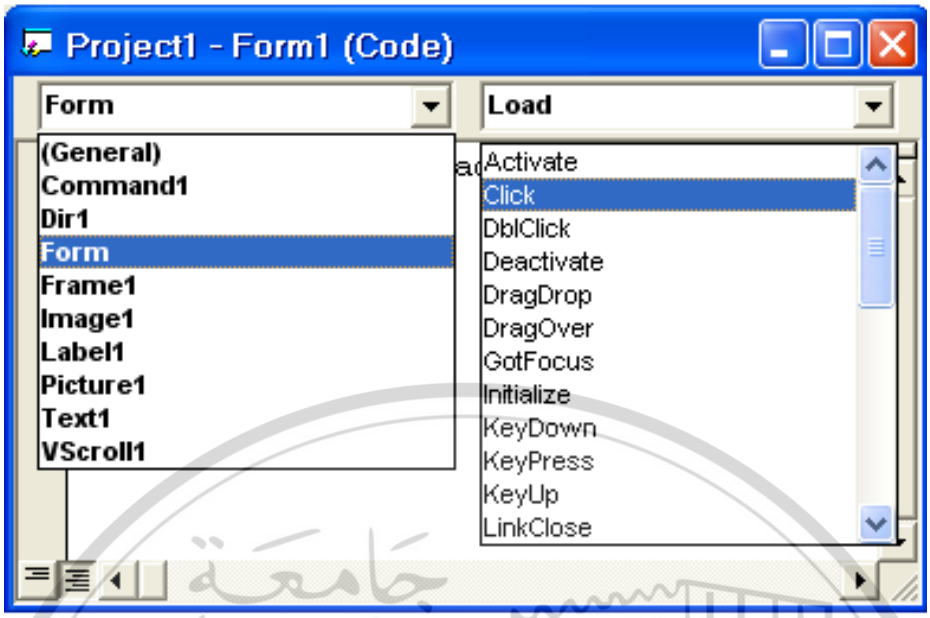
- استدعاء الحدث هي عملية كتابة اسم الحدث لتنفيذه كما تفعل مع الإجراءات.
- أما تفجير (حدوث) الحدث فهي عملية استدعاء الحدث من قبل نظام التشغيل و *VB*.

ملاحظة:

بالنسبة لنافذة النموذج، تسمية أحداثها دائما ما تبدأ بالكلمة *Form* وليس اسم النموذج الموجود في الخاصية *Name*.

كتابة إجراءات الحدث والاستجابة له:

عند الرغبة في استجابة أداة ما لحدث معين لابد أن نكتب الكود المطلوب تنفيذه في إجراء الحدث الخاص بهذه الأداة. نلاحظ في نافذة الكود مربعان منسدلان الأيسر يستعرض كل الكائنات الموضوعة على نموذج النافذة بما فيها النافذة *Form* نفسها والأيمن يستعرض الأحداث المتاحة للكائن المختار حاليًا في المربع الأيسر.



يتم إدخال التعليمات التي نريد أن ينفذها *VB* عند وقوع الحدث من خلال إجراء أو *Procedure*.

لمعرفة استخدام نافذة البرمجة وكتابة الإجراءات نقوم بالخطوات التالية:

- بعد تشغيل *Visual Basic* تظهر الواجهة *Form1* تلقائيًا داخل بيئة التصميم عند بداية التشغيل.
- من مربع الأدوات ننقر فوق أداة ما فتظهر الأداة على النموذج.
- ننقر فوق الأداة لتنشيطها ثم اضغط مفتاح *F4* يظهر مربع الخصائص وبه خصائص هذه الأداة ومن ثم نقوم بتغيير الخصائص المرغوب في تغييرها (مثل *Caption* & *Name*) وهكذا.
- ننقر نقراً مزدوجاً على الأداة فتظهر نافذة برمجية تستخدم لكتابة التعليمات *Codes* التي تتسبب في الاستجابة للحدث وتنفيذ ما تريده عندما تنقر زر الأمر.
- يبدأ الإجراء عادة بكلمة *sub* وينتهي بعبارة *End sub* ويظهر داخله النافذة البرمجية.
- يتم الإعلان عن المتغيرات حيث يتم Visual Basic بأسماء المتغيرات التي سنستخدمها وأنماط البيانات التي ستحتويها هذه المتغيرات *Integer, Data, String*. وتتكون صيغة الإعلان من متغير من الكلمة *Dim* يليها اسم اختياري للمتغير يليها *As* ثم نمط المتغير.

أنواع الأحداث:

وهناك نوعان رئيسيان من الأحداث أحداث يسببها المستخدم وأخرى يسببها النظام. الأحداث التي يثيرها المستخدم هي التي تنتج عن فعل المستخدم مثل ضغط مفتاح أو نقر زر الماوس.

الأحداث الشائعة:

هناك أحداث شائعة تستجيب لها معظم الأدوات ومنها:

١) النقر Click:

يقع عندما ينقر المستخدم فوق الأداة نقرة واحدة.

٢) النقر المزدوج DblClick:

يقع عندما ينقر المستخدم نقرًا مزدوجًا (نقرتين الفاصل بينهما قصير جداً) فوق الأداة.

٣) تحريك الماوس MouseMove:

يقع عندما يحرك المستخدم الفأرة فوق الأداة.

٤) ضغط زر الماوس للأسفل MouseDown:

يقع أثناء عملية النقر أي عندما ينقر المستخدم زر الفأرة وقبل أن يحرره.

٥) تحرير زر الماوس MouseUp:

يقع بعد عملية النقر.

٦) سحب فوق DragOver:

يقع عندما يمر المستخدم أثناء لسحب أداة فوق أداة أخرى.

٧) إفلات أو إلقاء DragDrop:

يقع عندما تستقر الأداة في المكان المطلوب في نهاية السحب.

٨) حصل تركيز GotFocus:

يقع عند تنشيط الأداة.

٩) فقدان التركيز LostFocus:

يقع عند تنشيط أداة أخرى غير النشطة.

١٠)ضغط مفتاح KeyPress:

يقع عندما يضغط المستخدم أحد الحروف الأبجدية من لوحة المفاتيح.

١١)ضغط مفتاح لأسفل KeyDown:

يقع أثناء ضغط المفتاح من لوحة المفاتيح وقبل تحريره.

١٢)تحرير مفتاح KeyUp:

يقع بعد ضغط المفتاح (أي أثناء تحرير المفتاح).

أحداث الماوس:

إن الغالبية العظمى من الأوامر الكودية المستجابة تكون ردة فعل لأعمال قام بها المستخدم بالماوس. أول حدث تعرضه لنا معظم الأدوات عند النقر عليها هو الحدث *Click* والذي يحدث لحظة النقر على الأداة بزر الفأرة الأيسر. والحدث *DblClick* الذي يمثل النقر المزدوج.

من الأساليب الخاطئة التي يتبعها قليل من المبرمجين هي كتابة أكواد في كلا الحدثين *Click* و *DblClick* لنفس الأداة (رغم أننا نستطيع عمل ذلك بلغة *VB*) إلا أنها طريقة غير مستحبة و تسبب التشويش. فلو قام المستخدم بالنقر المزدوج على الأداة، فان الحدث *Click* سيتم تنفيذه أولا ومن ثم تنفيذ الحدث *DblClick*. وفي حالة قيام المستخدم بالنقر المزدوج *Double Click* على الأداة، فان ترتيب وقوع الأحداث سيتم على النحو التالي:

MouseDown ⇨ ***MouseUp*** ⇨ ***Click*** ⇨ ***MouseMove*** ⇨ ***DblClick*** ⇨ ***MouseUp*** ⇨ ***MouseMove***

وبشكل عام يمكن أن نقول أن أحداث الماوس (الفأرة) هي الأحداث الناتجة عن ضغط أو تحرير زر ماوس أي ***Clicking And Releasing Buttons*** أو نتيجة تحريك هذه الماوس (الفأرة) أي ***Moving the Mouse***.

١) حدث ضغط زر الماوس *MouseDown Event*:

يختلف هذا الحدث عن حدث النقر *Click* لأن النقر مقصود به ضغط أحد الأزرار ثم تركه بينما ضغط زر الماوس سينفذ الإجراء قبل ترك زر الماوس أي أثناء ضغط أحد هذه الأزرار للأسفل وفي مكان خال من النموذج عدا شريط العنوان. أو بتعبير آخر هو الحدث الذي يترافق مع ضغط زر ما من الماوس عند وجود المؤشر فوق أداة ما.

والإجراء الكودي لهذا الحدث هو:

Private Sub ControlName_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)

VB Code

End Sub

للإجراء الكودي أربع وسيطات أو أدلة هي:

Button: وهي قيمة صحيحة تدل على أي زر من أزرار الماوس ويمكن أن تكون:

- $vbLeftButton$: أي زر الماوس الأيسر مضغوط.
- $vbMiddleButton$: أي زر الماوس الوسط مضغوط.
- $vbRightButton$: أي زر الماوس الأيمن مضغوط.

أي أن الوسيط *Button* سيأخذ قيمة صحيحة أثناء ضغط أحد أزرار الماوس وتكون قيمته كما يلي:

1 – إذا كان الزر الأيسر مضغوطاً.

2 – إذا كان الأيمن مضغوطاً.

4 – إذا كان الأوسط مضغوطاً.

Shift: وهو من نوع العدد الصحيح أيضاً وهو يدل على حالة مفاتيح لوحة المفاتيح *keyboard* الثلاثة وهي *Shift, Ctrl, Alt* إذا كانت مضغوطة أم لا ويأخذ أحد القيم التالية:

قيمة الوسيط Shift	نوع المفتاح المضغوط
1	$Shift$
2	$Ctrl$
3	$Shift + Ctrl$
4	Alt
5	$Shift + Alt$
6	$Ctrl + Alt$
7	$Alt + Shift + Ctrl$

القيم (x, y): من النوع *Single* ويشيران إلى إحداثي مؤشر الماوس عند موقع مؤشر الماوس عند الضغط على أحد أزرارها. وهي تمثل موقع مؤشر الماوس بالنسبة للأداة نفسها وليس للشاشة، حيث تمثل النقطة *(0, 0)* الزاوية العلوية اليسرى للأداة، وتزداد قيمة X كلما اتجه مؤشر الفأرة إلى جهة اليمين وتزداد قيمة Y كلما اتجه مؤشر الفأرة إلى الأسفل.

٢) حدث تحرير زر الماوس *MouseUp Event*:

وهو عكس الحدث *MouseDown* وينفذ إجراؤه لحظة تحرير زر الماوس عند وجود المؤشر فوق أداة ما. والإجراء الكودي لهذا الحدث هو:

Private Sub ControlName_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)

VB Code

End Sub

أي له أربع وسائط أيضاً إلا أن الوسيط *Button* يأخذ ثماني قيم وهي:

قيمة الوسيط Button	نوع المفتاح المضغوط	قيمة الوسيط Button	نوع المفتاح المضغوط
0	عدم الضغط على أي زر	**4**	الأوسط مضغوط
1	الأيسر مضغوط	**5**	الأيسر والأوسط مضغوطان
2	الأيمن مضغوط	**6**	الأيمن والأوسط مضغوطان
3	الأيسر والأيمن مضغوطان	**7**	الأزرار الثلاثة مضغوطة

٣) حدث تحريك الماوس *MouseMove Event*:

وهو الحدث الذي يترافق مع تحريك مؤشر الماوس فوق النموذج أو فوق أداة ما.

العبارة (الإجراء الكودي) لهذا الحدث هي:

Private Sub ControlName_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

VB Code

End Sub

٤) حدث السحب والإفلات *Drag And Drop*:

إن عملية السحب *Drag* هي عملية الضغط على الزر الأيسر للماوس (عندما يكون مؤشر الماوس فوق عنصر ما) والتحرك بمؤشر الماوس مع الإبقاء على الزر مضغوطاً.

إن عملية الإفلات (الإلقاء) *Drop* هي عملية رفع اليد عن الزر الأيسر من الماوس والمضغوط (عندما يكون مؤشر الماوس فوق عنصر ما) بعد عملية السحب *Drag*.

ينفذ هذا الإجراء بمجرد رفع اليد عن الزر الأيسر للماوس بعد سحب عنصر التحكم.

والإجراء الكودي لهذا الأمر هو:

Private Sub Command1_DragDrop(Source As Control, X As Single, Y As Single)
VB Code
End Sub

من الواضح أن له ثلاث وسائط هي:

Source – عنصر التحكم الذي يجري سحبه و إلقاؤه ويخزن فيه تلقائياً اسم الأداة *Name* التي يتم سحبها.

(x, y) الإحداثيات الحالية لمؤشر الماوس منسوباً إلى إحداثيات النموذج نفسه.

٥) حدث السحب فوق *DragOver*:

ينفذ هذا الإجراء عند سحب عنصر التحكم فوق النموذج أو أداة ما. والإجراء الكودي لهذا الأمر هو:

Private Sub Command1_DragOver(Source As Control, X As Single, Y As Single, State As Integer)
VB Code
End Sub

ومن الواضح أن له أربع وسائط هي:

- *Source* – عنصر التحكم الذي يجري سحبه ويخزن فيه تلقائياً اسم الأداة *Name* التي يتم سحبها.
- *(x, y)* – الإحداثيات الحالية لمؤشر الماوس منسوباً إلى إحداثيات النموذج نفسه.
- *State* – من النوع الصحيح *Integer* ويأخذ أحد القيم الثلاثة التالية:
- *0* – عندما يتم سحب عنصر تحكم من نقطة خارج النموذج إلى نقطة داخل النموذج.
- *1* – عندما يتم سحب عنصر تحكم من نقطة داخل النموذج إلى نقطة خارج النموذج.
- *2* – عندما يتم سحب عنصر تحكم من نقطة داخل النموذج إلى نقطة داخل النموذج أيضاً.

يتم تفجير الحدث ***MouseMove*** بمجرد أن يقوم المستخدم بتحريك المؤشر فوق الأداة، ونهاية الحدث تكون لحظة خروج المؤشر عن حدود الأداة. أما في حالة الالتقاط

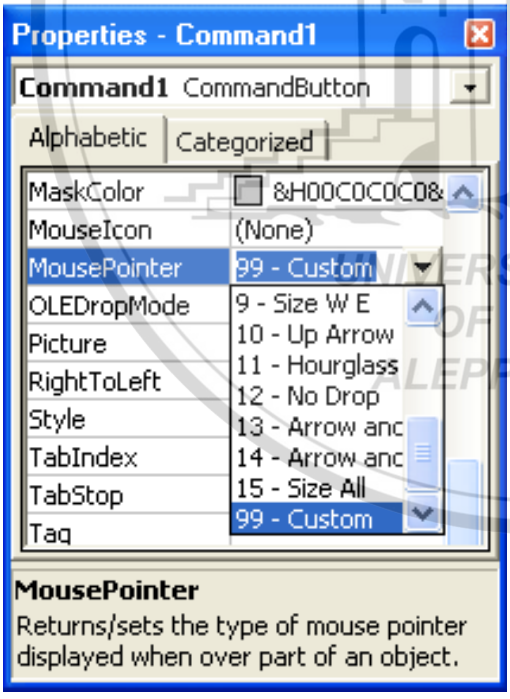
Capturing فإن الحدث ***MouseMove*** سيتم تفجيره حتى لوتعدّى المؤشر حدود الأداة مما يترتب عنه قيم سالبة للإحداثيات *X* و *Y* في حالة كون مؤشر الفأرة قد تحرك إلى يسار أو فوق الأداة.

ملاحظة:

كلمة الالتقاط *Capturing* هي عملية الضغط بزر الفأرة على الأداة مع استمرار الضغط على الزر.

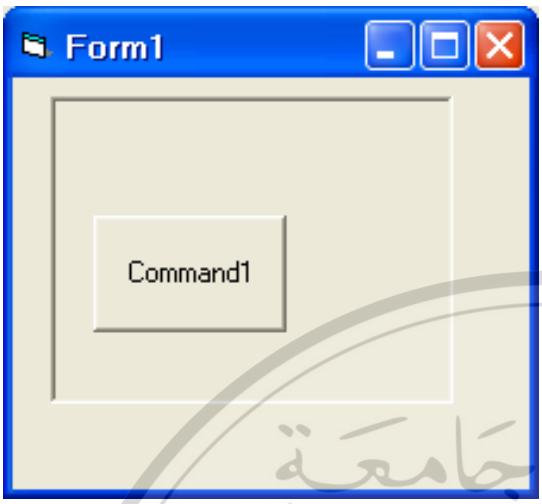
بالنسبة للحدثين *MouseDown* و *MouseUp* فسيتم وقوعهما بمجرد الضغط على زر الفأرة وتحرير الزر على التوالي حتى لو اختلفت الأزرار، فلو قمنا بالضغط على زر الفأرة الأيسر – وأبقيناه مضغوطاً – ومن ثم قمنا بالضغط على زر الفأرة الأيمن، سيقوم *VB* بتفجير الحدث *MouseDown* مرتين، وعند تحرير الأزرار، فان حدوثين للحدث *MouseUp* مقبلان (أي أن الحدث سيقع مرتين).

خصائص مؤشر الماوس:



- خصائص مؤشر الماوس *MousePointer* و *MouseIcon* تحددان الشكل المطلوب لمؤشر الفأرة *Mouse Cursor*.
- توفر لنا الخاصية *MousePointer* ستة عشر *16* مؤشراً قياسيّاً *(0, 1, 2, ... 15)* يوفرها نظام التشغيل.
- وعند الرغبة في تخصيص رمز معين مغاير للقيم الموجودة نختار القيمة *99-Custom* من الخاصية *MousePointer* السابقة مع تحميل ملف المؤشر في الخاصية *MouseIcon* في المرحلة المرئية أو كتابة الكود التالي في المرحلة التنفيذية:

$$Command1.MousePointer = vbCustom$$
$$Command1.MouseIcon = LoadPicture("C:\Test.ICO")$$



هنا لن نلاحظ أي تغيير للمؤشر إلا إذا مرر المستخدم مؤشر الفأرة فوق الأداة .مع ذلك، هناك عدة عوامل تمنع *VB* من تغيير شكل المؤشر.

إذا كان لدينا إطار عليه مربع صورة وبداخل مربع الصورة زر أوامر كما هو مبيَّن.

هنا لدينا ثلاثة أدوات هي *Form1, Picture1, Command1* والتي يمكن تزويد كل منها مرئياً أو كوديّاً بمؤشر ماوس خاص ونلاحظ ما يلي:

مؤشر الماوس الافتراضي هو الذي قيمته *Default – 0* أي مؤشر الماوس العادي وعندها لن يحصل أي تغير على أي أداة من الأدوات عند المرور فوقها.

- عند تحريك الماوس فوق أداة من الأدوات سيتغير شكل المؤشر وحسب الشكل المعطى بناء على الاحتمالات التالية:

١) الاحتمال الأول:

عندما تأخذ خصائص المؤشر للأداتين في المرحلة المرئية أو الكودية القيم التالية:

$$Picture1.MousePointer = 2$$
$$Command1.MousePointer = 0$$

عند المرور فوق أداة الصورة *Picture1* سيتغير شكل الماوس (حسب قيمته المساوية إلى 2) ويبقى هو ذاته حتى عند المرور فوق زر الأوامر *Command1* الموجود أصلاً فوق مربع الصورة (كون قيمة المؤشر هي الافتراضية في زر الأوامر أي مساوية للصفر).

٢) الاحتمال الثاني:

عندما تأخذ خصائص المؤشر للأداتين في المرحلة المرئية أو الكودية القيم التالية:

$$Picture1.MousePointer = 2$$
$$Command1.MousePointer = 5$$

عند المرور فوق أداة الصورة *Picture1* سيتغير شكل الماوس (حسب قيمته المساوية إلى *2*) حتى الوصول إلى مكان وجود زر الأوامر.

عند المرور فوق زر الأوامر *Command1* الموجود أصلاً فوق مربع الصورة سيتغير شكل مؤشر الماوس (حسب قيمته المساوية إلى *5*) وفي هذه المنطقة أصبحت الأفضلية لشكل مؤشر الماوس للأداة *Command1*.

٣) الاحتمال الثالث:

عندما تأخذ خصائص المؤشر للأداتين في المرحلة المرئية أو الكودية القيم التالية:

$$\boldsymbol{Picture1.MousePointer = 0}$$
$$\boldsymbol{Command1.MousePointer = 0}$$

عند المرور فوق أداة الصورة *Picture1* أو فوق أداة زر الأوامر *Command1* لن يحصل أي تغيير وسيبقى شكل مؤشر الماوس كما هو محدد للأداة *Form1*.

ملاحظة:

يجب عدم القيام بتغيير شكل المؤشر إلا عند الحاجة لتغييره، كتحويله إلى صورة يد عند المرور فوق رابط الموقع على الانترنت، أو على شكل الأسهم في حالة التحجيم، ومن المستحسن تحويله إلى شكل ساعة رملية عند بداية كل إجراء حتى يعلم المستخدم أن عليه الانتظار.

قيمة الوسيط ***Button*** إنطلاقاً من زر الماوس المضغوط:

الأزرار المضغوطة			قيمة الوسيط *Button*	
الأيسر	الأيمن	الأوسط	العددية	التابعية
√			1	*vbLeftButton*
1	0	0		
	√		2	*vbMiddleButton*
0	1	0		
		√	4	*vbRightButton*
0	0	1		

قيمة الوسيط ***Shift*** إنطلاقاً من أزرار التحكم المضغوطة من لوحة المفاتيح:

قيمة الوسيط *Shift* العددية	المفاتيح المضغوطة		
	Alt	Ctrl	Shift
1			√
	0	0	1
2		√	
	0	1	0
3		√	√
	0	1	1
4	√		
	1	0	0
5	√		√
	1	0	1
6	√	√	
	1	1	0
7	√	√	√
	1	1	1

قيمة الوسيط *Button* العددية	الأزرار المضغوطة		
	الأوسط	الأيمن	الأيسر
0			
	0	0	0
1			√
	0	0	1
2		√	
	0	1	0
3		√	√
	0	1	1
4	√		
	1	0	0
5	√		√
	1	0	1
6	√	√	
	1	1	0
7	√	√	√
	1	1	1

جامعة
حلب
١٩٥٨
UNIVERSITY
OF
ALEPPO

الفصل الثالث عشر
أحداث لوحة المفاتيح
KeyBoard Events

مقدمة:

هناك نوعان رئيسيان من الأحداث أحداث يسببها المستخدم وأخرى يسببها النظام. الأحداث التي يسببها المستخدم هي التي تنتج عن فعل المستخدم مثل ضغط مفتاح أو نقر زر الماوس.

فيما يخص لوحة المفاتيح نرى نوعين من الأحداث:

١) أحداث لوحة المفاتيح الناتجة ضـــغط أو تحرير مفتاح ما من لوحة المفاتيح *Clicking and Releasing Keys Events*.

تترافق فعاليات لوحة المفاتيح مع الأحداث التالية:

١. ضغط مفتاح للأسفل ***KeyDown***.
٢. تحرير مفتاح للأعلى ***KeyUp***.
٣. ضغط مفتاح آسكي ***KeyPress***.
٤. حدث التغيير ***Change***.

ثلاثة أحداث مرنة يوفرها لنا VB ناتجة من لوحة المفاتيح هي KeyPress, KeyDown, KeyUp فعندما يقوم المستخدم بالضغط على أي زر من أزرار لوحة المفاتيح، فسيقع الحدث KeyDown ثم يقوم VB بتحويل المفتاح المدخل إلى مقابله في جدول ASCII ثم يتم حدوث الحدث KeyPress وبعد أن يرفع المستخدم إصبعه عن المفتاح يبدأ الحدث KeyUp بالوقوع.

بالنسبة للحدثين KeyUp, KeyDown فيحدثان مع كل المفاتيح أما الحدث KeyPress فيحدث في حالة قيام المستخدم بالضغط على المفاتيح [Enter, BackSpace, Esc, Ctrl] وحروف لوحة المفاتيح الأخرى أما مفاتيح الأسهم ومفاتيح الوظائف وغيرها فلا تؤدي إلى وقوع الحدث KeyPress.

٢) أحداث التركيز الناتجة عن تنشيط أداة ما أو فقد تنشيط هذه الأداة $Focus\ Events$.

نلاحظ أن التنشيط يمكن أن يتم من خلال لوحة المفاتيح أو من خلال الماوس أيضاً.

أحداث لوحة المفاتيح:

١) حدث ضغط المفتاح للأسفل $Key\ Down\ Event$:

تحصل هذه الحادثة عند ضغط أحد المفاتيح على لوحة المفاتيح إلى الأسفل. والإجراء الكودي للحدث هو:

```
Private Sub ControlName_KeyDown
        (KeyCode As Integer, Shift As Integer)
    VB Code
End Sub
```

أثناء التنفيذ وبعد التركيز على المفتاح $ControlName$ أي سيظهر حوله مستطيل منقط للدلالة على امتلاكه تركيز لوحة المفاتيح. بعد ذلك وعند الضغط على أي مفتاح $KeyDown$ فإنه سيحدث الأمر الكودي الموجود.

حيث:

- $ControlName$: اسم الأداة التي تخضع للتركيز عند الضغط على أحد أزرار لوحة المفاتيح.
- $KeyDown$: اسم الحادثة على أحد مفاتيح لوحة المفاتيح.
- $KeyCode$: عبارة عن عدد صحيح يمثل شفرة المسح Scan Code للمفتاح المضغوط.
- الوسيط $Shift$ ويمثل حالة المفاتيح $Shift, Ctrl, Alt$.

بالنسبة إلى قيمة المفتاح المدخل – المتمثلة في المتغير $KeyCode$ – هي القيمة الفيزيائية للمفتاح في لوحة المفاتيح إلا أنها لا تمثل نوعية الحرف المدخل سواء كان صغيراً $Small\ Letter$ أو علامات أو رموزاً أخرى كـ ؟ ... @#% الخ أو حتى حروف عربية مثل ا، ب ، ت ... الخ. فهي ترسل دائماً القيمة للحرف الإنجليزي الكبير $Capital\ Letter$ كـ A, B, C.

أما بالنسبة لقيمة الوسيط $Shift$ فتتحدد من الجدول التالي:

قيمة الوسيط Shift	حالة المفتاح Alt	حالة المفتاح Ctrl	حالة المفتاح Shift
0	غير مضغوط	غير مضغوط	غير مضغوط
1	غير مضغوط	غير مضغوط	مضغوط
2	غير مضغوط	مضغوط	غير مضغوط
3	غير مضغوط	مضغوط	مضغوط
4	مضغوط	غير مضغوط	غير مضغوط
5	مضغوط	غير مضغوط	مضغوط
6	مضغوط	مضغوط	غير مضغوط
7	مضغوط	مضغوط	مضغوط

مثال:

```
Private Sub ControlName_KeyDown
            (KeyCode As Integer, Shift As Integer)
Lb1.Caption = ""
Lb1.Caption = "KeyCode" + Str (KeyCode) + vbCrLf
            + "Shift" + Str (Shift)
End Sub
```

في هذا الكود وبعد التركيز على المفتاح ControlName وعند الضغط على أي مفتاح فإنه شفرته أي KeyCode ستظهر على اللافتة Lb1 في السطر الأول بعد عبارة KeyCode =. وفي السطر الثاني ستظهر قيمة الوسيط الثاني انطلاقاً من حالة المفاتيح Alt و Ctrl و Shift بعد عبارة Shift =.

لتحديد المفتاح المضغوط يمكننا مقارنة الوسيط KeyDown مع ثوابت لوحة المفاتيح في VB فمثلاً إذا ضغطنا على المفتاح Num Lock فإن قيمة الوسيط KeyCode سوف تساوي قيمة الثابت VbKeyNumLuck وبشكل مشابه فإن الضغط على المفتاح F1 يعني أن قيمة الوسيط KeyCode ستساوي VbKeyF1 والضغط على المفتاح 2 يعني أن قيمة الوسيط ستساوي VbCode2 والضغط على المفتاح A يعني أن الوسيط KeyCode ستساوي قيمة الثابت VbCodeA ... وهكذا.

- يســـتخدم التابع الوظيفي ( )Str لتحويل القيمة الرقمية إلى قيمة نصـــية. ولذلك فإن التابع الوظيفي Str(KeyCode) ســـيحول الوســـيط KeyCode من قيمة رقمية من النوع الصحيح Integer إلى قيمة نصية من النوع String وكذلك فإن التابع الوظيفي Str(Shift) سيحول الوسيط Shift من قيمة رقمية من النوع الصحيح Integer إلى قيمة نصية من النوع String.
- التابع VbCrLf يستخدم لفتح سطر جديد وبالتالي الانتقال بالمؤشر إلى بداية السطر الثاني.

```
Private Sub Form_KeyDown
        (KeyCode As Integer, Shift As Integer)
If Shift And vbShiftMask Then
المفتاح [SHIFT] مضغوط'
End If
If Shift And vbCtrlMask Then
المفتاح [CTRL] مضغوط'
End If
If Shift And vbAltMask Then
المفتاح [ALT] مضغوط'
End If
End Sub
```

٢) حدث تحرير المفتاح للأعلى Key Up Event:

تحصل هذه الحادثة عند تحرير أحد المفاتيح على لوحة المفاتيح.

مثال عند الضغط على أي مفتاح من لوحة المفاتيح وعندما يكون التركيز موجوداً على زر أمر ControlName وباستخدام الإجراء الكودي ( )ControlName_keyUp وسيكون الإجراء كما يلي:

```
Private Sub ControlName_KeyUp
        (KeyCode As Integer, Shift As Integer)
   [Visual Basic Code]
End Sub
```

أثناء التنفيذ وبعد التركيز على المفتاح ControlName أي سيظهر حوله مستطيل منقط للدلالة على امتلاكه تركيز لوحة المفاتيح. بعد ذلك وعند تحرير أي مفتاح مضغوط من لوحة المفاتيح أي KeyUp فإنه سيحدث الأمر الكودي الموجود. حيث:

- ControlName: اسم الأداة التي تخضع للتركيز عند الضغط على أحد أزرار لوحة المفاتيح.
- KeyUp: اسم الحادثة على أحد مفاتيح لوحة المفاتيح.
- KeyCode: عبارة عن عدد صحيح يمثل شفرة المسح Scan Code للمفتاح المحرر.
- Shift: ويمثل حالة المفاتيح Shift, Ctrl, Alt.

```
Private Sub ControlName_KeyUp
                (KeyCode As Integer, Shift As Integer)
Lb1.Caption = ""
Lb1.Caption = "KeyCode" + Str(KeyCode) + vbCrLf
            + "Shift" + Str(Shift)
End Sub
```

٣) حدث استمرار ضغط المفتاح Key Press Event:

تحصل هذه الحادثة عند الضغط على مفتاح ما له شفرة ASCII مرافقة. والإجراء الكودي له:

```
Private Sub ControlName_KeyPress
                (KeyAscii As Integer)
[Visual Basic Code]
[Visual Basic Code]
End Sub
```

فمثلاً عند وضع التركيز على أداة ما ControlName وأثناء الضغط على مفتاح من المفاتيح التي لها شفرة آسكي فإنه سيتم تنفيذ الإجراء KeyPress( ) أما إذا ضغطنا على مفتاح ليس له شفرة آسكي فلن يتم تنفيذ أي أمر أو أي إجراء كودي.

حيث:

- ControlName: اسم زر الأوامر الذي سيتم عليه التركيز Focus.
- KeyPress: اسم الحادثة على مفاتيح لوحة المفاتيح.
- KeyAscii: وسيط يمثل قيمة ASCII للمفتاح المضغوط.

عند الضغط على المفتاح A سنجد أنه سيتم تنفيذ الإجراء KeyPress( ) وذلك كون المفتاح A يمتلك على شفرة ASCII مرافقة بينما إذا ضغطنا على المفتاح F1 أو أي سهم من أسهم الحركة فلن يتم تنفيذ أي أمر كون المفتاح F1 لا يملك شفرة مرافقة.

مثال:

```
Private Sub ControlName_KeyPress(KeyAscii As Integer)
Dim Char
Lb1.Caption = " "
Char = Chr(KeyAscii)
Lb1.Caption = "KeyAscii" + Str(KeyAscii) + vbCrLf
            + "Char = " + Char
End Sub
```

- يحول التابع الوظيفي ( )Chr القيمة ASCII الصحيحة إلى حرف:

```
Char = Chr(KeyAscii)
```

- المتحول Char يحتفظ الآن برمز المفتاح المضغوط.

يؤمن الكود رؤية رمز المفتاح المضغوط إضافةً إلى قيمة ASCII المرافقة له.

مثال:

```
Private Sub Form_ KeyPress( KeyAscii As Integer)
Print Chr$( KeyAscii) & " = " & KeyAscii
End Sub
```

يرسل الحدث KeyPress قيمة عددية من النوع Integer متمثلة في متغير عددي بالاسم KeyAscii تمثل المقابل العددي للحرف المدخل في جدول ASCII.

المتغير KeyAscii مرسل بالمرجع وليس القيمة أي يمكنك تعديل قيمته مما يترتب عليه مرونة كبير في التحكم في مدخلات المستخدم.

مثال:

```
Private Sub Text 1 _KeyPress( KeyAscii As Integer)
KeyAscii = Asc( UCase( Chr$( KeyAscii)))
End Sub
```

هذا الكود لتحويل الحروف المدخلة في أداة النص إلى حروف كبيرة Capital Letters.

مثال:

```
Private Sub Text 1 _KeyPress( KeyAscii As Integer)
If KeyAscii < Asc(" 0") Or KeyAscii > Asc(" 9") Then
KeyAscii = 0
End If
End Sub
```

- وإذا أسندت قيمة الصفر إلى هذا المتغير فهذا يعني أنه تمت عملية إرسال قيمة المفتاح إلى الأداة المستقبلة له.
- هذا الكود مثلا يمنع المستخدم من كتابة أي شيء في أداة النص عدا الأعداد من صفر حتى تسعة أي 0, 1, 2, … 9.

ملاحظة:

تم اعتماد الدالتين Asc, Chr$ فيما سبق ويمكن الاستغناء عنهما إذا كنا نعرف المقابل العددي للحرف المطلوب في جدول ASCII.

٤) حدث التغيير Change Event:

يتم تفجير حدث التغيير Change بمجرد القيام بتغيير محتويات الأداة كتغير النص الظاهر في الخاصية Caption أو الخاصية Text ولكن الاعتماد على هذا الحدث فيه شيء من الخطأ، فعند تغيير قيمة الخاصية Value للأداتين CheckBox و OptionButton لن يقوم VB بإحداث هذا الحدث، كذلك عند تغيير الشكل الظاهري للأدوات كحجمها أو ألوانها لن يتم تفجير هذا الحدث.

٥) أحداث التركيز Focus Event:

إن الكائن الذي يملك تركيز لوحة المفاتيح هو الكائن الذي سيستجيب لإدخالها أو للتأثير عليها. وعندما يكون تركيز لوحة على كائن ما (أداة) فإنه سوف يبدي تغيّراً في شكله على الشاشة كأن يظهر مستطيل منقط حوله (زر أوامر) أو أن يومض مؤشّره (شريط تمرير) … وهكذا.

من خلال ما سبق يمكن ومن خلال الرؤية معرفة العنصر الذي يملك التركيز. كما يمكن كوديّاً معرفة العنصر الذي يخضع حاليا للتركيز من خلال حدثين هما:

١. الحدث GotFocus (حصل على التركيز).
٢. الحدث LostFocus (أي فقد التركيز).

يتم تفجير الحدث GotFocus عندما تستقبل الأداة التركيز، والحدث LostFocus عندما تفقد الأداة التركيز، سواء كان ذلك بالفأرة أو لوحة المفاتيح أو برمجيا . أما بالنسبة لنافذة النموذج، فهذه الأحداث تعمل جيدا بها شريطة عدم وجود أي أداة قابلة لاستقبال التركيز.

ملاحظة:

لن تتم عملية حدوث الأحداث بالطريقة المتوقعة إذا فقدت النافذة تركيزها بسبب الانتقال إلى تطبيق آخر أو استقبلت تركيزها بعد الانتقال من تطبيق آخر. باختصار، أحداث التركيز لا تعمل إلا بين نوافذ وأدوات برنامجك فقط.

a) حدث وصول التركيز Got Focus:

الإجراء الكودي لهذا الحدث هو:

```
Private Sub ControlName_GotFocus ( )
Lb1.Caption = "ControlName سيظهر التأثير عند وصول التركيز إلى الأداة"
End Sub
```

أي هنا ستظهر عبارة "سيظهر التأثير عند بداية الضغط على المفتاح للأسفل" على اللافتة Lb1 فور وصول التركيز على الزر KeyDown.

b) حدث مغادرة التركيز Lost Focus:

الإجراء الكودي لهذا الحدث هو:

```
Private Sub ControlName_LostFocus ( )
Lb1.Caption = "ControlName سيختفي التأثير عند التركيز عن الأداة"
End Sub
```

أي هنا ستظهر عبارة "سيختفي التأثير عن هذا العنصر" على اللافتة Lb1 فور زوال (فقدان) التركيز عن الزر ControlName.

ملاحظة: يمكن مقارنة العملية بعملية تحديد عنصر على سطح المكتب أو نافذة ما والذي عند ذلك سيتغير لونه وإن أي أمر مثل قص أو نسخ سيتم عليه بكونه هو العنصر المحدد أو الخاضع للتأثير أو التركيز.

عملية تحويل رموز ASCII إلى حروف كبيرة (تحويل الحروف الصغيرة إلى كبيرة) والعكس (تحويل الحروف الكبيرة إلى صغيرة):

قد يحتاج المستخدم في بعض البرامج إلى تحويل الحروف الصغيرة إلى كبيرة فقط أو صغيرة فقط بغض النظر عن حالة المفتاحين Shift و Caps Lock.

يوجد تابع وظيفي هو التابع UCase( ) من العبارة (Upper Case) والذي وظيفته تحويل الأحرف الصغيرة إلى كبيرة. كما أن التابع الوظيفي LCase( ) من العبارة (Lower Case) هو تابع وظيفي معاكس للسابق إذ يقوم بتحويل الأحرف إلى صغيرة دائماً.

ملاحظة:

ليس لهذه الوظائف أي تأثير على النصوص المكتوبة باللغة العربية وتعيد النص كما هو بدون تغيير.

مثال: المطلوب استخدام مربع النص وباستخدام الإجراء KeyPress لتحويل الحروف إلى حروف كبيرة كما يلي:

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
Dim Char
Char = Chr(KeyAscii)
KeyAscii = Asc(UCase(Char))
Lb1.Caption = Lb1.Caption + vbCrLf + "KeyAscii"
            + Str(KeyAscii) + " Char = " + Char
End Sub
```

- Text1: تدل على أن الإجراء يطبق على مربع نص.
- KeyPress: تدل على أن الإجراء المطبق على مربع النص سيتم تطبيقه عند الضغط على مفاتيح لوحة المفاتيح.
- التابع ( )Chr يقوم بتحويل شفرة Code الحرف المضغوط إلى رمز أو حرف:

$$Char = Chr(KeyAscii)$$

- يحول الرمز نفسه إلى رمز كبير (حرف كبير) من خلال التابع الوظيفي ()UCase.
- من الواضح أن: $UCase("a") = A$ كما أن $UCase("A") = A$
- يقوم التابع الوظيفي ( )Asc بتحويل القيمة المعادة من التابع الوظيفي ()UCase إلى قيمة صحيحة تمثّل قيمة ASCII للوسيط الممرر له:

$$KeyAscii = Asc(UCase(Char))$$

- باختصار يزوّد الإجراء ( )Text1_KeyPress قيمة صحيحة تمثل قيمة آسكي ASCII للمفتاح المضغوط ثم تحوّل هذه القيمة إلى حرف ثم يحّول الحرف إلى حرف كبير ثمّ يُسند قيمة آسكي ASCII للحرف الكبير إلى الوسيط KeyAscii مرة أخرى وهكذا يعتقد مربع النص أن المستخدم ضغط على حرف (رمز) كبير.
- طبعاً من الممكن إعادة تنفيذ التمرين وإجبار البرنامج على كتابة الحروف جميعها على شكل حروف صغيرة من خلال اسبدال التابع في الكود المبيَّن:

$$KeyAscii = Asc(UCase(Char))$$

الخاصية Tab و TabIndex:

- يسمح لنا ويندوز بالانتقال من عنصر تحكم إلى عنصر تحكم آخر بواسطة المفتاح Tab (نقل تركيز لوحة المفاتيح من عنصر تحكم إلى آخر) بينما يقوم بنقل التركيز بالاتجاه المعاكس. $[Shift + Tab]$
- تحدد الخاصية TabIndex الترتيب الذي تستقبل وفقه عناصر التحكم التركيز في لوحة المفاتيح فمثلاً إذا كان تركيز لوحة المفاتيح حاليّاً عند عنصر تحكم ما وكانت قيمة الخاصية TabIndex له هي 5 فإن الضغط على المفتاح Tab سينقل تركيز لوحة المفاتيح إلى العنصر الذي تساوي فيه قيمة الخاصية TabIndex إلى 6 بينما إذا ضغطنا على $[Shift + Tab]$ سيتم نقل التركيز إلى العنصر الذي فيه $TabIndex = 4$.
- يعمل ترتيب الانتقال وفق Tab بطريقة دائرية فإذا كان عنصر التحكم الذي يمتلك حاليّاً التركيز في لوحة المفاتيح يحمل أعلى قيمة للخاصية TabIndex فإن الضغط مجدداً على Tab سيؤدي إلى انتقال تركيز لوحة المفاتيح إلى العنصر الذي تساوي قيمة الخاصية TabIndex فيه إلى الصفر وبالعكس عند الضغط على $[Shift + Tab]$ والتركيز موجود في عنصر تحكم قيمة الخاصية TabIndex فيه تساوي الصفر ينتقل تركيز لوحة المفاتيح إلى عنصر التحكم الذي يحمل أعلى قيمة للخاصية TabIndex.
- يسند VB أرقام متسلسلة إلى الخاصية TabIndex فهذه الخاصية لأول عنصر تساوي الصفر وتساوي 1 للعنصر الذي يليه وهكذا. عموماً يمكن تغيير الخاصية TabIndex لتحديد الترتيب الذي يناسب المستخدم ضمن البرنامج.

ملاحظة:

لا تتقبل بعض عناصر التحكم تركيز لوحة المفاتيح مثل عنصر تحكم اللافتة Label بالرغم من وجود خاصية TabIndex لها. وعند نقل التركيز من أداة إلى أخرى باستخدام مفتاح $[Tab]$ أو باستخدام التركيب $[Shift + Tab]$ فإنه سيتم تجاوز الأداة Label بغض النظر عن قيمة الخاصية TabIndex العائدة لها. كما أنَّه من غير الضروري أن تكون القيم متزايدة بتسلسل محدد، والمهم فقط أن تكون متزايدة.

الخاصية Cancel والخاصية Default:

تستخدم هذه الخاصية Cancel لتوفير استجابة للضغط على المفتاح Esc وعادة يتم ربط هذه الخاصية مع مفتاح الخروج.

- نقوم بتغيير الخاصـــية Cancel للزر خروج والمطلوب اســـتعمال Esc معه ونجعلها True.
- نقوم بتنفيذ البرنامج.
- نضغط على المفتاح Esc على لوحة المفاتيح.
- يستجيب البرنامج للمفتاح Esc كما لو أن المستخدم ضغط على زر الأمر خروج هذا بغض النظر عن عنصر التحكم الذي يستحوذ على تركيز لوحة المفاتيح كوننا أسندنا القيمة True للخاصية Cancel لزر الخروج.

بشكل مشابه يمكن إسناد القيمة True إلى الخاصية Default للزر خروج عندها سيستجيب البرنامج للضغط على المفتاح Enter وكأنه نقر على الزر خروج مما سيؤدي إلى إنهاء البرنامج وهذا أيضاً وبغض النظر عن أين يقع تركيز لوحة المفاتيح.

الخاصية KeyPreview للإطار Form:

أخيرا، أحداث لوحة المفاتيح KeyDown, KeyPress, KeyUp يتم حدوثها عندما يكون التركيز على الأداة المكتوب فيها الكود، وإذا وجدت أحداث إضافية تابعة لنافذة النموذج عندها ستكون الأولوية كما يلي فإن كانت قيمة الخاصية KeyPreview التابعة لنافذة النموذج تساوي True فان أحداث النافذة ستقع أولا ومن ثم الأداة التي عليها التركيز، أما إن كانت قيمة هذه الخاصية False فإن نافذة النموذج ستتجاهل هذه الأحداث وكأنها غير موجودة، ولن تحدث إلا أحداث الأداة فقط.

جامعة
حلب
١٩٥٨
UNIVERSITY
OF
ALEPPO

الفصل الرابع عشر
النماذج
Forms

مقدمة – تعريف النموذج:

يظهر في منتصف الشاشة تقريبًا ويستخدم لتصميم واجهات البرنامج حيث توضع عليها جميع الأدوات المستخدمة ويحتوي البرنامج على واحد أو أكثر من هذه النماذج. **إذاً النموذج عبارة عن كائن يعمل كحاوية للكائنات الأخرى كالعناوين ومربعات النص ومربعات الرسم التي تتكون منها في النهاية واجهة المستخدم.**

يحتوى النموذج على كل العناصر التي توجد في نوافذ البرنامج حال تشغيله فهو يحتوي على شريط عنوان وقائمة التحكم وعدة أزرار للتحكم (تكبير، تصغير، إغلاق).

تظهر أرضية النموذج أثناء التصميم على هيئة شبكة نقطية Grid والتي تسمح لنا بمحاذاة العناصر على النموذج والتحكم في نوافذ النماذج.

يمكن التحكم في النموذج أثناء تصميمه بأن نسحبه من مقابض التحجيم أو بأن نحدد قيم الخاصتان Width, Height من مربع الخصائص، كما يمكننا التحكم في مكان وحجم النموذج أثناء تشغيل البرنامج (أي في المرحلة الكودية).

من المهم معرفة كيفية تطوير برنامج ما من خلال إنشاء تأثيرات قوية وبرامج متينة ومتماسكة، لذا يجب أن نتعلم أكثر عن عملية إضافة مزيد من النماذج إلى الواجهة لمعالجة الإدخال والإخراج والانتقال إلى الخيارات المتعددة التي قد تفرضها كل حالة من الحالات.

في الكثير من الحالات يكون النموذج الواحد كافٍ لإنجاز كل ما هو مطلوب من البرنامج. لكن عندما نريد تقديم معلومات أكثر إلى (أو الحصول على معلومات أكثر من) المستخدم، عندها سنضطر أن نرهق النموذج الوحيد بعدد كبير من العناصر والأوامر الكودية والإجراءات الحدثية. يمكن بشكل أفضل أن نضيف نماذج أخرى للبرنامج وهو ما يعطي تفصيل ووضوح أكثر للبرنامج ويسهل عمله واستخدامه من قبل المستخدم. يعتبر كل نموذج جديد أنه كائن ويملك كائنات وخصائص وإجراءات حدثية خاصة به.

يسمى النموذج الأول في البرنامج Form1 والنماذج التالية Form2 و Form3 ... وهكذا.

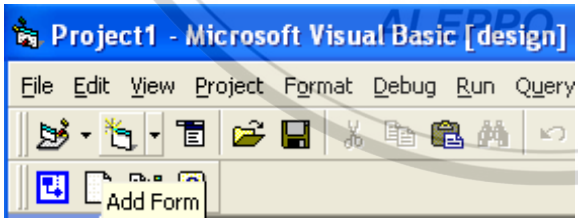
يمكن استخدام النماذج في كثير من الحالات مثل:

استخدام النموذج	الشرح
الشاشة الإفتتاحية	شاشة تعرض رسالة ترحيب أو أعمالاً فنية أو معلومات عن حقوق النشر عندما يبدأ البرنامج بالعمل.
إرشادات البرنامج	شاشة تعرض معلومات وتلميحات عن كيفية عمل البرنامج.
مربعات الحوار	مربعات حوار مخصصة تقبل الإدخال وتعرض الإخراج في البرنامج.
محتويات المستند ورسومه الجاهزة	شاشة تعرض محتويات ملف أو أكثر مستعمل في البرنامج.

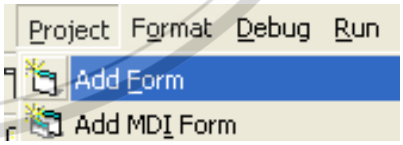
النماذج الفارغة و المسبقة التصميم:

إضافة نموذج فارغ:

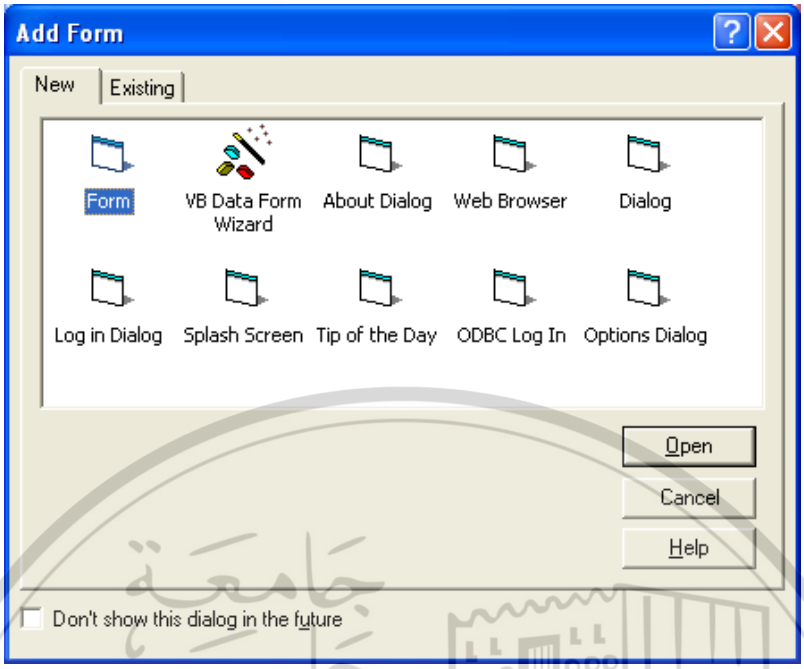
يمكن إنشاء نموذج جديد باختيار الأمر Add Form من القائمة Project. عندها سيظهر مربع حوار يسأل عن نوع النموذج المطلوب، وهنا نستطيع اختيار نموذج جديد فارغ (علامة التبويب New) ونضغط على Open. أو من زر إضافة نموذج في شريط الأدوات ثم نقر Form.



Project ⇨ Add Form ⇨ New ⇨ Open



Standard Tool Bar ⇨ Add Form



كيفية استعمال النماذج:

يمكن جعل كل النماذج في البرنامج مرئية في الوقت نفسه، أو يمكن تحميلها وإلغاء تحميلها عند الحاجة إلى ذلك. عند الحاجة إلى العمل مع أكثر من نموذج في وقت واحد يمكن أن نسمح للمستخدم التبديل بين هذه النماذج أو يمكن ترتيبها والتحكم بترتيب استعمالها.

عند التبديل بين النماذج نميز بين نوعين منها النوع الأول يسمى النموذج المشروط Modal وهو النموذج الذي يحافظ على التركيز إلى أن يتم استخدامه (بالنقر على Ok أو Cancel أو بطريقة ما أخرى) أما النموذج الذي يستطيع المستخدم تبديله من دون أن يستخدمه يسمى نموذجاً غير مشروطاً NonModal.

إن أغلب برامج النظام Windows تستخدم نماذج غير مشروطة عندما تعرض المعلومات كونها تعطي المستخدم حرية أكبر على صعيد الاستعمال، لذا فالنوع الافتراضي عند إنشاء نموذج هو النوع غير المشروط.

يمكن ضبط أي خاصية من خصائص النموذج الجديد بشكل مستقل عن الذي قبله بما في ذلك تسميته التوضيحية وحجمه ونمط حدوده وألوانه الأمامية والخلفية ونوع خطه وصورته الخلفية.

التعامل مع أحداث النموذج:

هناك خمسة أحداث رئيسية بالنسبة للنموذج يمكن التعامل معها بكتابة إجراء حدثي معين وهي:

١) الحدث Load:

ويحدث بعد تحميل النموذج في الذاكرة.

```
Private Sub Form_Load()
  VB Code
End Sub
```

٢) الحدث Activate:

يحدث عند أول ظهور للنموذج ثم بعد ذلك عندما يتحول المستخدم إلى النافذة لتنشيطها.

```
Private Sub Form_Activate()
  VB Code
End Sub
```

٣) الحدث Deactivate:

يحدث عند تنشيط نموذج آخر من نفس البرنامج.

```
Private Sub Form_Deactivate()
  VB Code
End Sub
```

٤) الحدث Unload:

يحدث قبل إفراغ الذاكرة من النافذة.

```
Private Sub Form_unload(Cancel As Integer)
  VB Code
End Sub
```

٥) الحدث Initialize:

يقع مرة واحدة فقط لكل نموذج حتى إذا تم إفراغ الذاكرة منه ثم إعادة تحميله لأنه يقع عند تسجيل بيانات النافذة كصنف جديد من النوافذ.

```
Private Sub Form_Initialize()
  VB Code
End Sub
```

التعامل مع النماذج:

١) تحميل النموذج Load Form:

تستخدم هذه العبارة لتحميل نموذج جديد إلى ذاكرة البرنامج:

$Load\ FormName$

عند تحميل النموذج (طبعا لن يتم عرضه إلى الآن) سنستطيع استعماله في أي إجراء حدثي ونستطيع الوصول إلى أي خاصية أو طريقة نريد استعمالها معه.

$Load\ Form2$
$Form2.Caption = "Hello"$

لضبط الخاصية Caption للنموذج Form2 بعد تحميله وجعلها Hello.

٢) إظهار النموذج Show Form:

عندما يصبح النموذج في ذاكرة البرنامج سيصبح جاهزاً للظهور، عندها سنستطيع أن نستدعيه باستعمال الطريقة Show مع التحديد بجعل ظهور النموذج مشروطاً أم غير مشروط.

$FormName.Show\ Mode$

تأخذ Mode القيمة 0 للنموذج غير المشروط (الافتراضي) والقيمة 1 إذا كان مشروطاً.

$\boldsymbol{If}\ Mode\ =\ 0\ \boldsymbol{Then}\ FormName\ is\ Modal$
$\boldsymbol{If}\ Mode\ =\ 1\ \boldsymbol{Then}\ FormName\ is\ NonModal$

لاحظ:

$Load\ Form2$

تم تحميل النموذج إلى ذاكرة البرنامج.

$Form2.Show$

إظهار النموذج من دون تحديد نمط ظهوره الذي سيكون افتراضياً غير مشروطاً.

$Form2.Show\ 1$

إظهار النموذج مع تحديد نمط ظهوره الذي سيكون مشروطاً.

يمكن صياغة هذا الموضوع من خلال الثوابت المعبرة عن معنى هذه القيم أي Mode.

فإذا كانت Mode=1 أو أخذت القيمة modal فلن يمكن لنافذة أو حتى كود أن ينفذ ما لم يتم إغلاق هذه النافذة.

أما إذا كانت قيمة Mode = 0 أو أخذت الثابت modeless فذلك يعني إمكانية التفاعل مع النوافذ الأخرى أثناء ظهورها ويمكن أن تغطيها نوافذ أخرى.

```
Private Sub Form_Load()
Form2.Show vbmodel
End Sub
```

٣) إخفاء النموذج Hide Form:

إن عبارة Hide هي عكس عبارة Show تماماً إذ تقوم بإخفاء النموذج عن الواجهة (جعله غير مرئياً) إلا أنه يبقى في ذاكرة البرنامج ليُستعمل لاحقاً في البرنامج.

إن اخفاء النموذج مطابق لجعله غير مرئياً باستعمال الخاصية Visible.

```
Private Sub Form_Load()
Form2.Hide
End Sub
```

٤) إلغاء تحميل النموذج Unload Form:

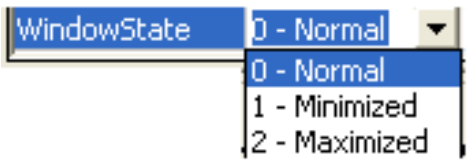
إن إلغاء تحميل نموذج ما Unload هو عكس عملية تحميله Load.

عند إلغاء تحميل نموذج سيتم إزالته من ذاكرة البرنامج وتحرير هذه الذاكرة التي كان يتم استعمالها من قبل النموذج لتخزين كائناته ورسومه، لكن لا يتم تحرير المساحة التي تستعملها إجراءات النموذج الحدثية فهذه الإجراءات تبقى في الذاكرة دوماً.

```
Private Sub Form_Load()
Unload Form2
End Sub
```

حجم النموذج WindowState:

يمكن تصغير النموذج (وضعه بشريط المهام) أو تكبيره (توسيعه ليملأ الشاشة بأكملها) أو إعادته إلى الوضع العادي (الحجم الذي تم قبوله في مرحلة البرمجة المرئية) وذلك باستعمال الخاصية WindowState كما يلي:



- Form. WindowState = 1 لتصغير النموذج في شريط المهام.
- Form. WindowState = 2 لتكبير النموذج وجعله يأخذ أبعاد الشاشة كاملةً.

- Form. WindowState = 0 لإعادة النموذج إلى حجمه العادي الافتراضي.

يمكن أن نستخدم العبارات الواضحة بدلاً من القيم التي تدل عليها فمثلا العبارة:

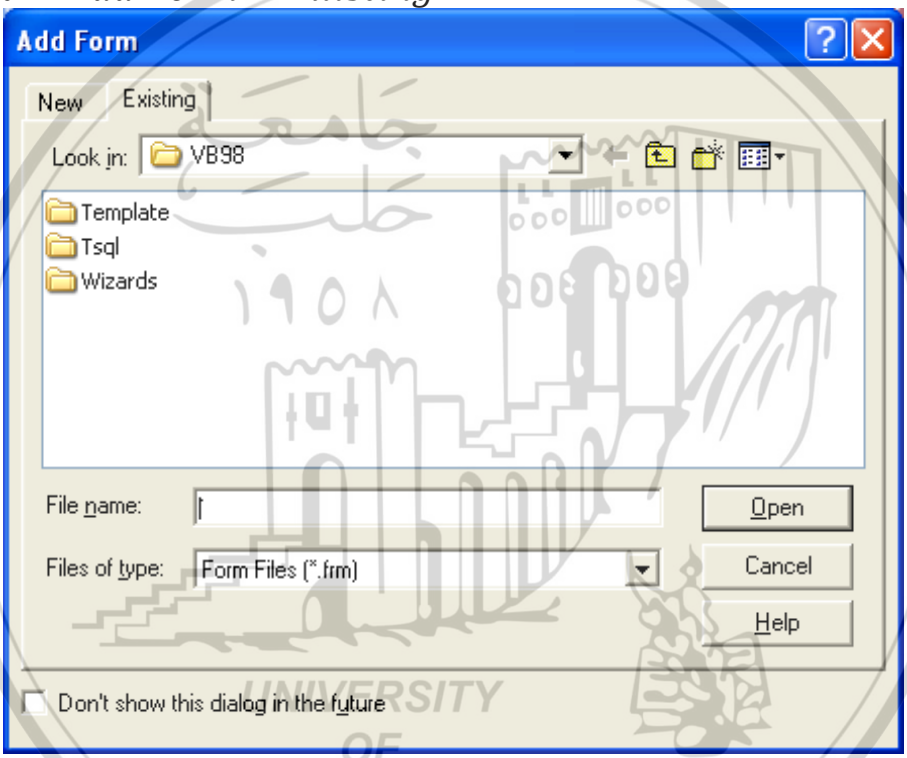
$$Form.WindowState = VbMaximized$$

تستخدم لتكبير النموذج وجعله يأخذ أبعاد الشاشة كاملةً.

إضافة نموذج موجود مسبقاً Adding An Existing Form:

يسمح لنا VB باستخدام نماذج مُنشأة سابقاً في مشاريع برمجية جديدة كما يلي:

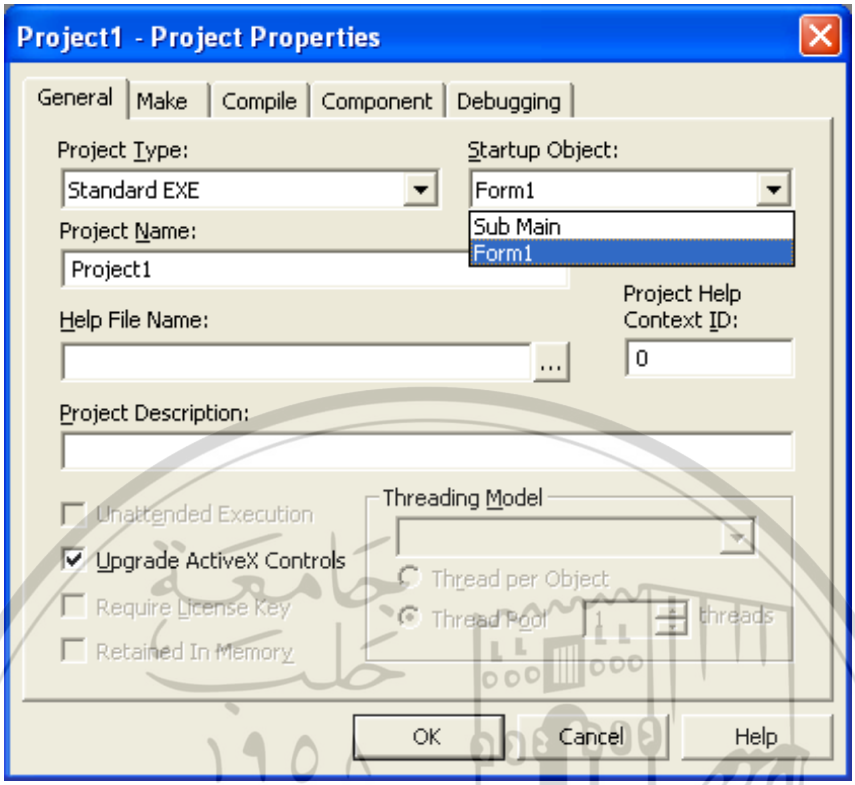
$$Project \Rightarrow Add\ Form \Rightarrow Existing$$



نأخذ الخيار Add Form من القائمة Project ثم علامة التبويب Existing بعدها سيظهر مربع الحوار Existing يسرد فيه كل النماذج الموجودة في المجلد الحالي (النماذج تأخذ اللاحقة frm.).

لإضافة نموذج موجود مسبقاً نبحث عنه في المجلد الموجود فيه ومن ثم ننقر عليه نقراً مزدوجاً في مربع الحوار بعدها سيضيفه VB إلى المشروع ونستطيع بعدها معاينته وتعديل إجراءاته الحدثية بنقر زري رؤية الكائنات ورؤية الشيفرة في إطار المشروع.

يمكن تحديد نموذج بدء التشغيل من الأمر Properties من القائمة Project ثم علامة التبويب General ومن ثم اختيار النموذج المناسب في مربع السرد Startup Object.

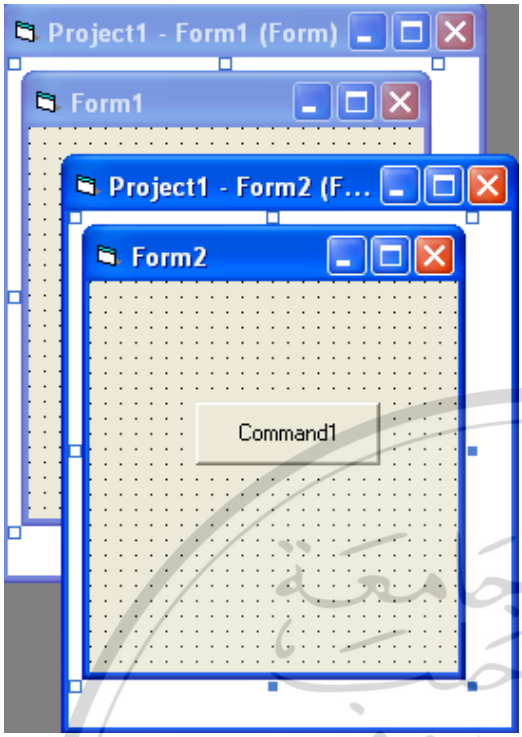


إذا استخدمنا نموذجاً قديماً في برنامجاً جديداً وأجرينا أي تعديلات عليه فإنه سوف لن يعمل بشكل صحيح في البرنامج القديم لذا من الواجب أولا بعد إضافة نموذج قديم أن نحفظه باسمه ومكانه الجديد ثم إجراء أي تعديل نريد عليه كي لا يؤثر ذلك على النموذج الأصلي وبالتالي البرنامج الأصلي.

إغلاق نموذج غير أساسي:

يمكن إضافة زر أوامر Command ضمن أي نموذج Form2 ونضيف فيه عبارة Close لإغلاقه عندما ننتهي منه. إن الإجراء الحدثي Command1_Click مقترن بالزر الأول من النموذج Form2 وليس على النموذج Form1 لذا يمكن إغلاقه مع إبقاء عمل Form1 على حاله.

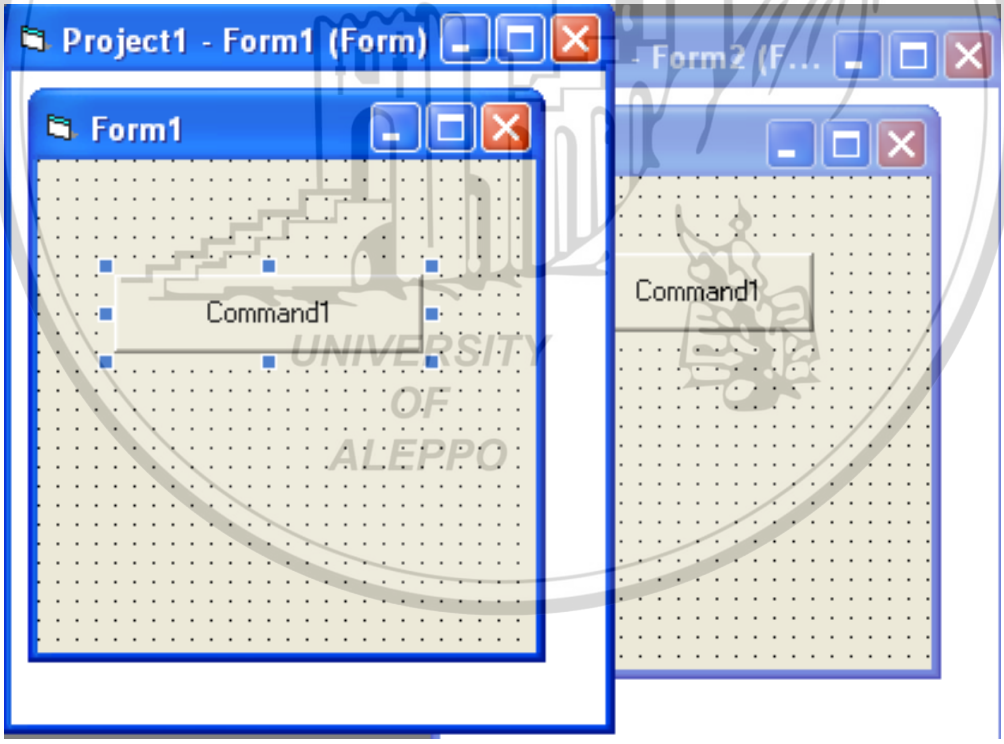
يمكن إضافة زر أوامر Command ضمن أي نموذج Form2 ونضيف فيه عبارة Close لإغلاقه عندما ننتهي منه. إن الإجراء الحدثي Command1_Click مقترن بالزر الأول من النموذج Form2



وليس على النموذج Form1 لـذا يمكن إغلاقـه مع إبقـاء عمـل Form1 على حاله.

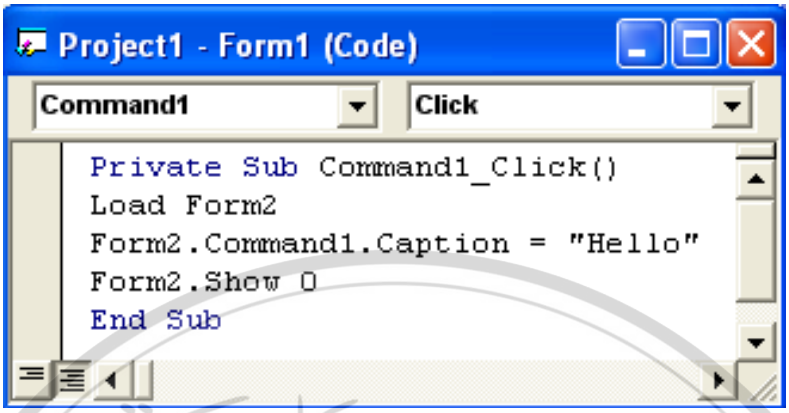
```
Private Sub Command1_Click( )
  Close
End Sub
```

ملاحظة:

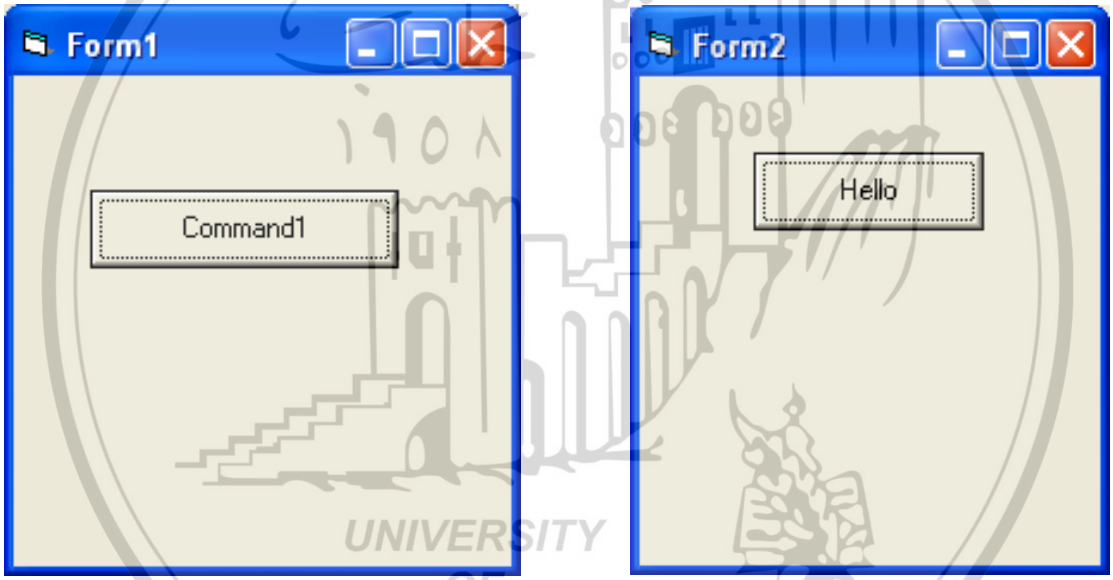


يمكن للكائنات الموجودة على نماذج مختلفة أن تحمل الاسم نفسه لأن VB يستطيع التمييز بينها. لكن من المفضل أن تحمل الكائنات أسماء مختلفة بحيث يكون اسمه منطقياً بحيث نتذكر وظيفته بسهولة وأن يكون متعلقاً بالنموذج الموجود عليه وبطبيعة الوظيفة

الملقاة على عاتقه. وعند تغيير اسم كائن ما على نموذج ما يجب أن نتأكد من تغيير اسمه في الشيفرة الكودية بما يتلاءم مع التسمية الجديدة.



وعند تنفيذ البرنامج تظهر النتيجة كما يلي:



واجهة المستخدم متعددة المستندات MDI – Multi Document Interface:

تتألف برامج MDI من عدة مستندات، وكل مستند منها موجود ضمن إطار منفصل والكل موجود على سطح النافذة الرئيسية (النافذة الأم)، والتي تضم على سطحها كل المستندات المفتوحة الحالية مثل برامج Word و Excel وغيرها. وبشكل عام يمكن القول أن النماذج MDI أطر تجمعها علاقة أصل – فرع.

قد يكون لدينا نموذجاً قياسياً نقوم فيه بالقيام بمعظم أعمالنا ويكون نقطة الإنطلاق دوماً. وقد نضطر لإضافة نماذج أخرى لغايات خاصة تعالج عمليات الإدخال والإخراج في البرنامج. يمكن من خلال VB إعداد تسلسل هرمي وهو علاقة خاصة بين النماذج

التي تعمل على نحو أفضل بصفتها مجموعة واحدة تسمى هذه النماذج MDI (واجهة متعددة المستندات) وهي تتميز بدورها كنماذج أصل وفرع.

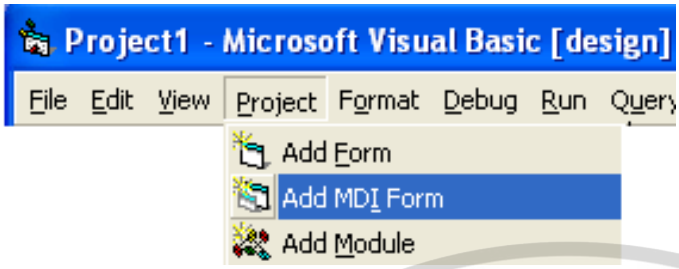
يتم إنشاء نموذج أصل باختيار الأمر Add MDI Form من القائمة Project ونموذجاً آخر فرع باختيار الأمر Add Form من القائمة Project ثم ضبط خاصية النموذج MDIChild فيه عند True.

تعمل النماذج MDI كالنماذج العادية خلال التشغيل فيما عدا الحالات الاستثنائية التالية:

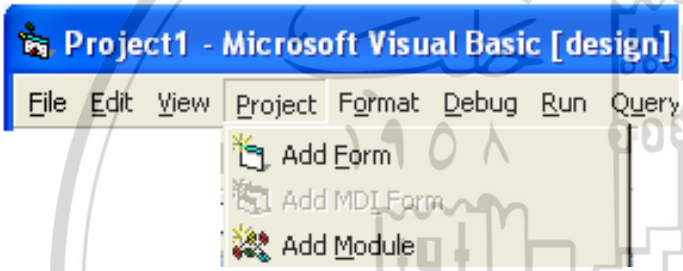
- تُعرض كل النماذج الفروع ضمن مساحة إطار نموذجها الأصلي.
- عند تصغير نموذج فرع، يتضاءل حجمه إلى شريط عنوان صغير على النموذج الأصل بدلاً من ان يبدو كزر في شريط المهام.
- عند تصغير النموذج الأصل، يظهر مع كل نماذجه الفروع كزر واحد على شريط المهام.
- تُعرض كل قوائم النموذج الفرع على شريط قوائم النموذج الأصل وعند تكبير نموذج فرع، تظهر تسميته التوضيحية في شريط عنوان النموذج الأصل.
- يمكن إظهار كل النماذج الفروع بضبط الخاصية AutoShowChildren للنموذج الأصل عند True.
- يمكن سحب أي نموذج من النماذج الأبناء إلى أي مكان على سطح النافذة الرئيسية ولكن لا يمكن سحبه خارج حدود النافذة الرئيسية.
- تفيد النماذج الأصل والفرع عند العمل على برامج تتمحور حول مستند واحد والتي تُستعمل فيها عدة أطر لعرض المستند أو تحريره.
- في المرحلة التنفيذية وعند تصغير احد النوافذ الأبناء (بنفس طريقة تصغير النوافذ القياسية بنقر الرمز (–) الموجود أعلى يمين كل نافذة) أو (اختيار البند تصغير من قائمة النظام الفرعية والتي تظهر بالنقر على رمز النافذة الموجود أعلى يسار النافذة) بعد عملية التصغير ستصبح نافذة النموذج الأم كما يلي:

إضافة نموذج MDI:

تتم الإضافة من قائمة **Project** الأمر **Add MDI Form**:



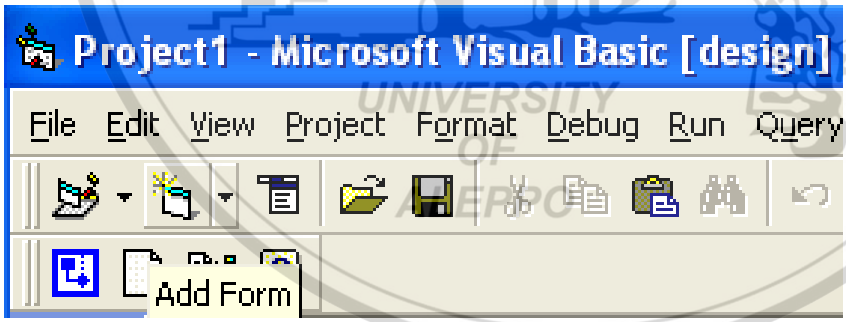
يمتلك كل برنامج MDI نافذة رئيســـية واحدة فقط لذلك نلاحظ أنه بعد أن نضـــيف هذه النافذة مرة واحدة نلاحظ أن الخيار Add MDI Form يصبح باهتاً (غير فعالاً) في قائمة Project كما هو موضح بالشكل:



أما النماذج الأبناء فيمكن إضافة العدد الذي نريد بإحدى الطريقتين:

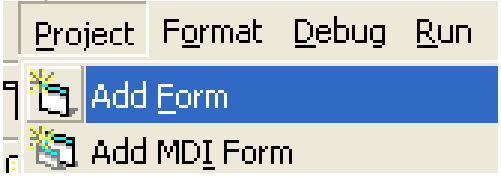
١) **من شريط الأزرار القياسية (شريط الأدوات).**

Standard Tool Bar → Add Form

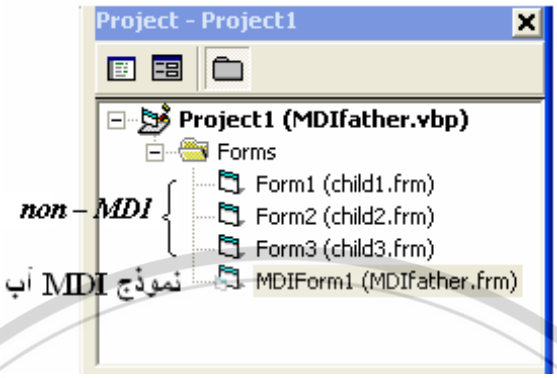


٢) **من القائمة Project أمر Add Form:**

Project → Add Form → New → Open



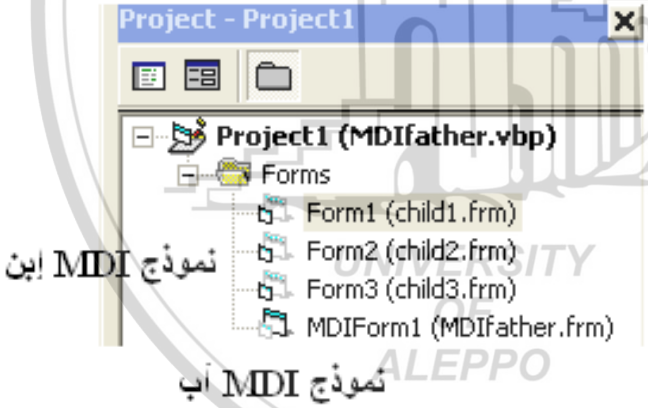
نلاحظ من خلال الشكل فارق بين رمز النموذج MDI الأب وبين رمز النماذج القياسية العادية:



فمثلاً رمز النموذج القياسي (non – MDI) عبارة عن صورة نموذج يقف على حافته وأمامه اسم النموذج العادي.

أما رمز نموذج MDI فهو عبارة عن صورة نموذج يقف على حافته وأمامه نموذج أصغر خافت.

نلاحظ من خلال الشكل فارق بين رمز النموذج MDI الأب وبين رمز نماذجه الأبناء:



يمكن تغيير خاصية MDIChild للنموذج من False للنموذج القياسي بالحالة الافتراضية إلى True للدلالة على أنها MDI form.

عندها سيتغير شكل رموز النوافذ الأبناء فأصبح الرمز عبارة عن صورة نموذج يقف على حافته وخلفه نموذج أكبر خافت كما في الشكل:

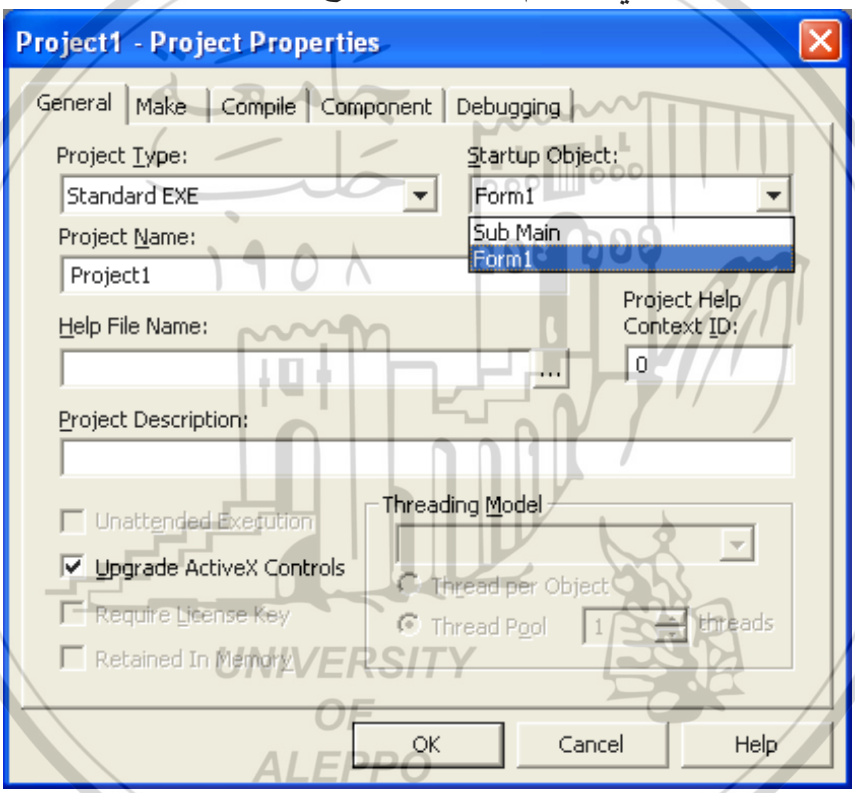
النموذج الذي سيحمل أولاً:

يمكن لأي برنامج أن يحوي على نموذج MDI رئيسي أب (واحد فقط) وعدة نماذج أبناء. إذاً يمكن للبرنامج أن يحتوي على نموذج رئيسي أب واحد فقط وعلى عدة نماذج أبناء ولكن قد يحتوي أيضاً على عدة نماذج قياسية Standard Form (ليست أبناء). في

هذه الحالة من المفترض التحديد اي منهما سيقلع أولاً (سيتم تحميله وإظهاره أولاً). هل هو النموذج الأب الرئيسي MDI أم نموذج من النماذج القياسية؟.

نستطيع تحديد ذلك من خلال الأمر Project Properties من القائمة Project وبأخذ علامة التبويب General ووضع النموذج الذي سيتم تحميله فوراً في حقل Startup Object.

عند اختيار نافذة قياسية ستُحَمَّل هي أولاً وعند أخذ نافذة إبن فمن الواضح أنه حتى يتم تحميله سيكون من البديهي أن يتم تحميل النموذج MDI الأب أولاً.



إضافة قائمة بالنوافذ المفتوحة:

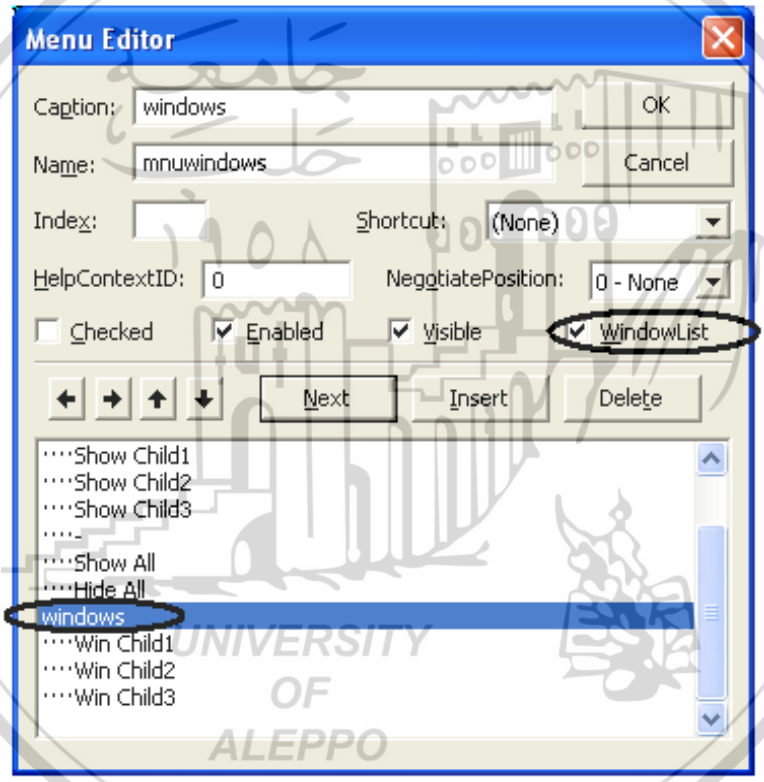
تتضمن معظم برامج MDI الاحترافية قائمة باسم Window أو بالعربي إطار. تتضمن قائمة إطار لائحة بأسماء النوافذ الأبناء المفتوحة حالياً.

عند كتابة القائمة إطار Window في مربع الأدوات Menu Editor نضع علامة الإختيار في مربع الإختيار WindowList وهو الخيار الذي سيُظهر عناوين جميع النوافذ الأبناء المفتوحة في النموذج الرئيسي كما في الشكل المبين.

تختصر ميزة القائمة Window الكثير من الأعمال البرمجية.

يشير كل بند من بنود القائمة نافذة Window إلى عنوان نموذج إبن فرعي ويَظهر على يسار (يمين) النموذج النشط الحالي علامة اختيار (حسب جهة الكتابة من اليسار لليمين أو العكس).

نستطيع من خلال هذه الخاصية إغلاق نموذج مفتوح أو فتح نموذج مغلق وكل ما علينا فعله هو وضع علامة الإختيار في مربع الإختيار WindowList في مربع الحوار Menu Editor لأحد البنود.



ترتيب النوافذ Arrange Windows:

يمكن إضافة البنود "ترتيب متتالي" و "ترتيب متجانب" و "ترتيب الأيقونات" للقائمة نافذة حيث تساعد هذه الخيارات على ترتيب النوافذ الأبناء بعدة أشكال على سطح النافذة الرئيسية. ويتم ذلك كما يلي:

- من مربع الحوار Menu Editor نستطيع إضافة بنود خاصة لقائمة Window في تطبيقات MDI.

- نضـــيف خط فاصـــل ثم نضـــيف الخيارات Arrange Cascade للترتيب المتتالي و Vertical Arrange للترتيـب العمودي و Horizontal Arrange للترتيـب الأفقي أو Arrange All لترتيب الكل.
- بعد ذلك نكتب في كل قائمة من هذه القوائم الأمر الذي يحقق معناها.

فمثلاً الأمر:

MDIForm1.Arrange VbCascade

يستعمل هذا الأمر لترتيب النماذج بشكل متتالي. وفي هذا الأمر نقول أن الطريقة Arrange تستخدم على النموذج MDIForm1 مع الثابت vbCascade لترتيب نماذج الأبناء المبعثرة بشكل متتالي.

أما الأمر:

MDIForm1.Arrange VbTileVertical

فيستعمل لترتيب النماذج بشكل عمودي، وفي هذا الأمر نقول أن الطريقة Arrange تستخدم على النموذج MDIForm1 مع الثابت البرمجي VbTileVertical لترتيب نماذج الأبناء المبعثرة بشكل متجانب عمودياً.

أما الأمر:

MDIForm1.Arrange VbTileHorizontal

فيستعمل لترتيب النماذج بشكل أفقي، وفي هذا الأمر نقول أن الطريقة Arrange تستخدم على النموذج MDIForm1 مع الثابت البرمجي VbTileHorizontal لترتيب نماذج الأبناء المبعثرة بشكل متجانب أفقياً.

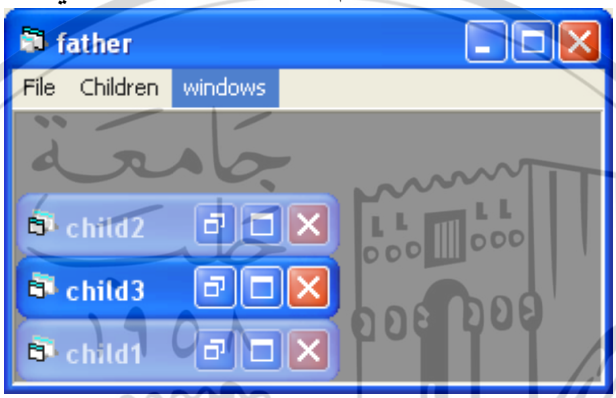
أما الأمر:

MDIForm1.Arrange VbArrangeIcons

فيستعمل لترتيب أيقونات النماذج المصغرة بشكل عام. وفي هذا الأمر نقول أن الطريقة Arrange تستخدم على النموذج MDIForm1 مع الثابت البرمجي VbArrangeIcons لترتيب أيقونات النماذج المصغرة بشكل عام. ولا يعمل هذا البند إلا إذا كانت جميع النوافذ الأبناء مصغرة للحد الأدنى.

- تتم عملية ترتيب الأيقونات للنماذج بشكل يتلائم مع حجم النموذج الأب الرئيسي.
- لاحظ الأشكال التالية:

- أيقونات متواجدة بشكل مبعثر وغير منظم بعد عملية تصغيرها في النموذج الأب.



- أيقونات النوافذ المصـــغرة بعد عملية الترتيب. ونلاحظ أن عرض النافذة الأب قليل لذا فإن الإيقونات توضعت بشكل طولاني.

- أيقونات النوافذ المصـغرة بعد عملية الترتيب. نلاحظ أن عرض النافذة الأب قليل نسـبياً لذا فإن الإيقونات توضعت بشكل طولاني وعرضاني.

- أيقونات النوافذ المصـــغرة بعد عملية الترتيب. و نلاحظ أن عرض النافذة الأب كبير فســمح للإيقونات من أن تتوضع بشكل عرضاني تماماً.

إضافة نماذج أبناء جديدة في المرحلة التنفيذية:

إن معظم البرامج المرتبطة بـ Windows مزودة بخاصية Multiple-Document Interface – MDI وهي البرامج التي تحتوي على عدة نوافذ (أبناء) والتي تقع ضمن نافذة رئيسية (نافذة أم) ومن أهم هذه البرامج برنامج محرر النصوص Microsoft Word وبرنامج الجداول الإلكترونية Microsoft Excel.

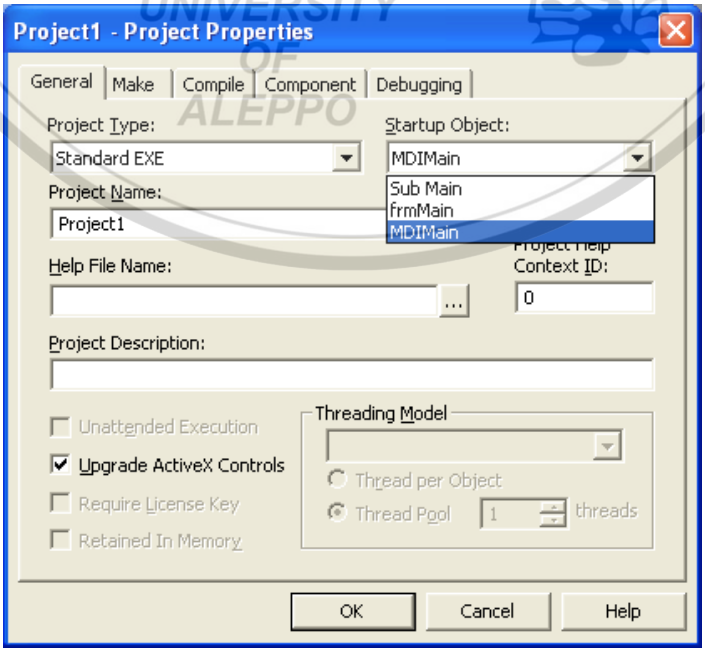
لعمل البرنامج لا بد من تزويده بنافذة نموذج أب رئيسي MDIMain ونموذج قياسي عادي frmMain.frm. لأجل ذلك نقوم بتسمية النافذة الإفتراضية القياسية frmMain ونجعل الخاصية MDIChild = True ومن ثم نقوم بإضافة نموذج MDI من القائمة Project أو من شريط الأدوات كما مر سابقاً كما يلي:

١) **من قائمة Project.**

٢) **من شريط الأدوات.**

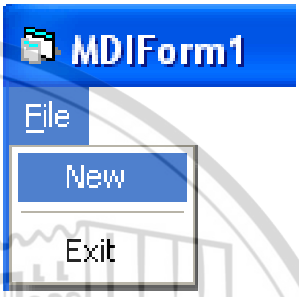
Standard Tool Bar → Add Form
Project → Add Form → New → Open

بعد ذلك نجعل خيار الإقلاع الإفتراضي للنموذج الأب وذلك من خلال علامة التبويب General من الأمر Project Properties من القائمة Project.



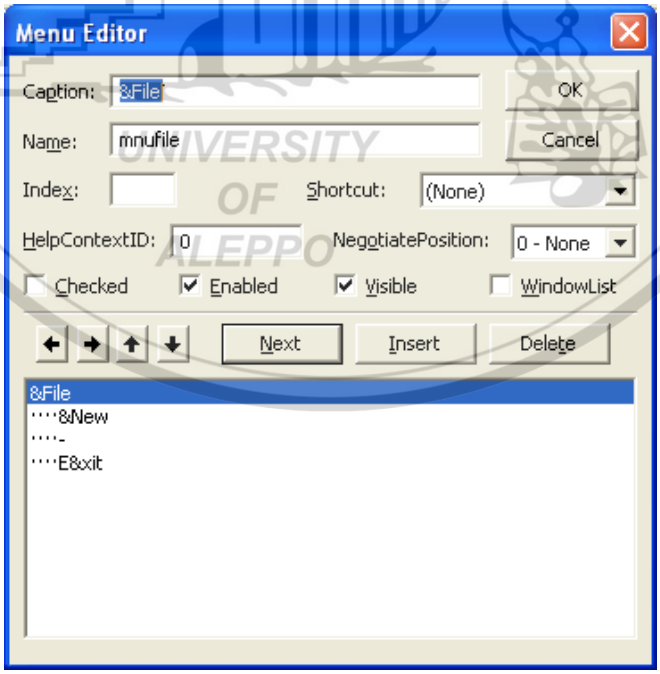
يمكن إضافة الأوامر الكودية اللازمة من خلال أدوات تحكم عادية على النموذج أو من خلال قائمة لذلك ولجمالية العمل نقوم من خلال محرر القوائم Menu Editor بإنشاء شريط قوائم فيه قائمة File التي تحتوي على خيارات من أهمها (ما يخصنا الآن) الخيار أو القائمة جديد (New).

نصمم القائمة بالشكل التالي:

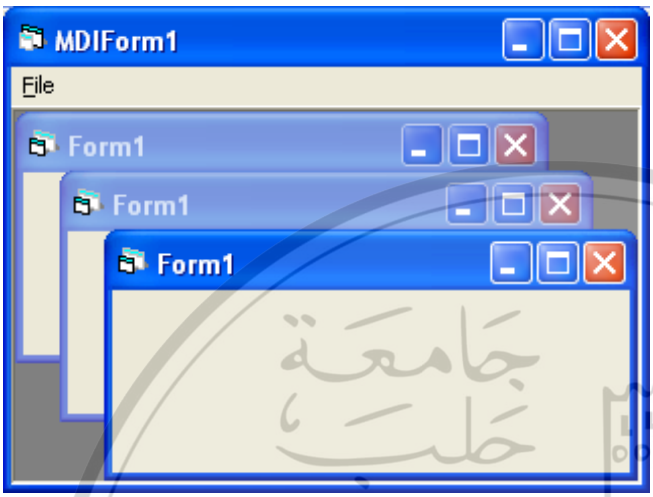



```
Private Sub mnuexit_Click()
    End
End Sub
Private Sub mnuName_Click()
    Dim frmNew As New frmMain
    frmNew.Show
End Sub
```

تصرح العبارة Dim frmNew As New frmMain من هذا الإجراء عن متحول اسمه frmNew وعلى أنه نسخة مطابقة بمواصفاتها للنموذج الإبن frmMain الذي تم إنشاؤه في المرحلة التصميمية أي تم إنشاء نسخة مطابقة للنموذج الذي تم إنشاؤه بالمرحلة التصميمية. و تؤدي العبارة frmNew.Show إلى إظهار هذا النموذج وجعله نشطاً (فعالاً).



إن الرمز & يستخدم في الحقل النصي Caption فقط. يعني هذا الرمز أن الحرف الذي يأتي وراءه سيظهر وتحته خط ويسمى مفتاحاً ساخناً وسيتميز عن غيره بأنه سنستطيع تنفيذ قائمته بالضغط على تركيب مؤلف من [هذا المفتاح + Alt].



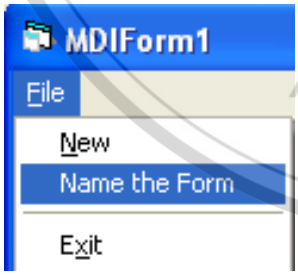
نلاحظ أنه عند تنفيذ البرامج وعند الضغط على القائمة جديد ستظهر نافذة جديدة مسماة Form1 وهو الإسم الإفتراضي الذي يأخذه أي نموذج جديد.

في الحقيقة سيتم إنشاء نموذج جديد كنسخة طبق الأصل عن النموذج الأصل في كل مرة نختار فيها البند جديد (New) من القائمة ملف (File) وستظهر هذه النماذج على النموذج الأب الرئيسي بحيث لا تتعدى حدودها حدوده وتحمل نفس الإسم كما هو مبين.

الخاصية ActiveForm:

تمثل هذه الخاصية النموذج النشط الحالي. وإذا لم يكن هناك نموذج نشط حالياً فإن الخاصية ActiveForm فإنها ستشير إلى آخر نموذج نشط تم استخدامه.

تغيير اسم النموذج الفعال:



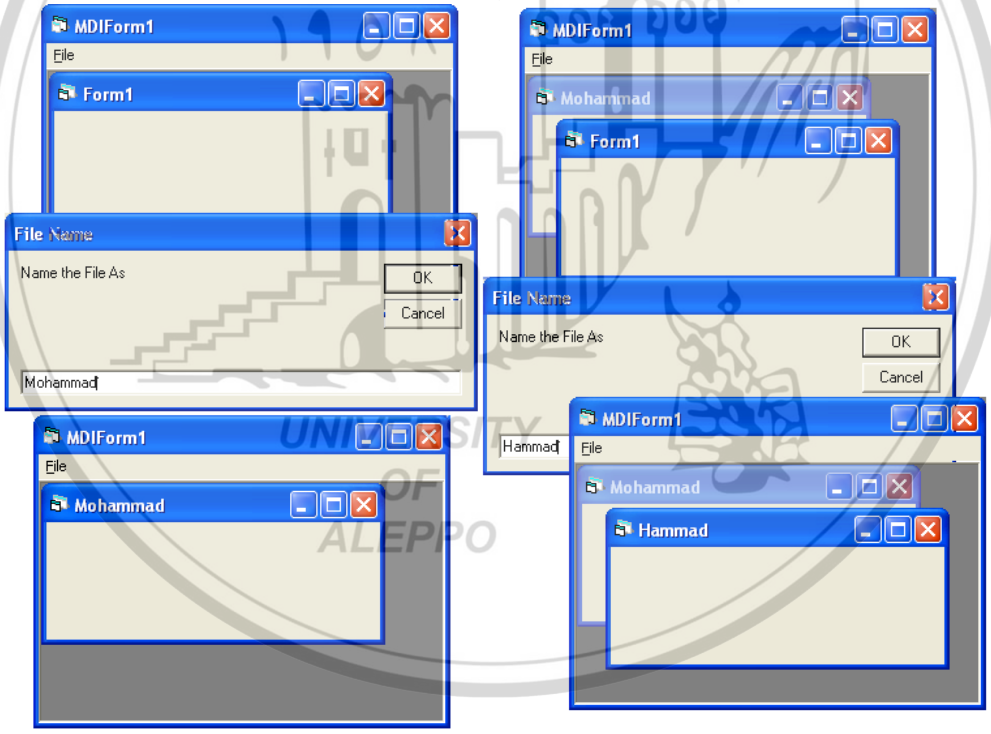
يمكن تغيير اسم النموذج الفعال بإضافة قائمة تطلب منا اسم هذا النموذج عن طريق مربع حوار كما يلي:

نقوم بتعريف متحول نصي اسمه docName ويقوم الإجراء الحدثي للقائمة بما يلي:

```
Private Sub frmName_Click()
  Dim docName As String
  docName = InputBox("Name the File As","File Name")
  MDIMain.ActiveForm.Caption = docName
End Sub
```

سـيظهر مربع إدخال InputBox يطلب اسـم النموذج المطلوب إدخاله ومن ثم يعطيه للنموذج النشـط (الفعال) وهذه ما تفسره العبارة التي تحتوي على الخاصـية ActiveForm في الإجراء الحدثي.

- العبارة Dim docName As String تعريف متحول نصي اسمه docName.
- العبارة docName = InputBox("Name the File As", "File Name") لإدخال قيمة من خلال مربع إدخال InputBox وإسنادها إلى المتحول docName.



- العبـارة MDIMain.ActiveForm.Caption = docName تعني إسـنـاد القيمـة docName للخاصـيـة Caption للنموذج الابن النشـط ActiveForm من أبن أبناء النموذج الأب MDIMain.
- بعد كل عملية إنشـاء نموذج جديد بالقائمة New نسـتطيع تغيير اسـمه وعندها سـتصـبح النوافذ الأبناء كما هو مبيّن في الشكل.

هذه العملية صحيحة ولكن ينقصها حس برمجي جميل.

- نستطيع من خلال الأوامر الكودية إعطاء اسم ثابت دوما مع رمز أو رقم يدل على تسلسل النموذج الجديد. في محرر النصـــوص تظهر لدينا Document1 أولاً ثم Document2 ثانياً و ... وهكذا.

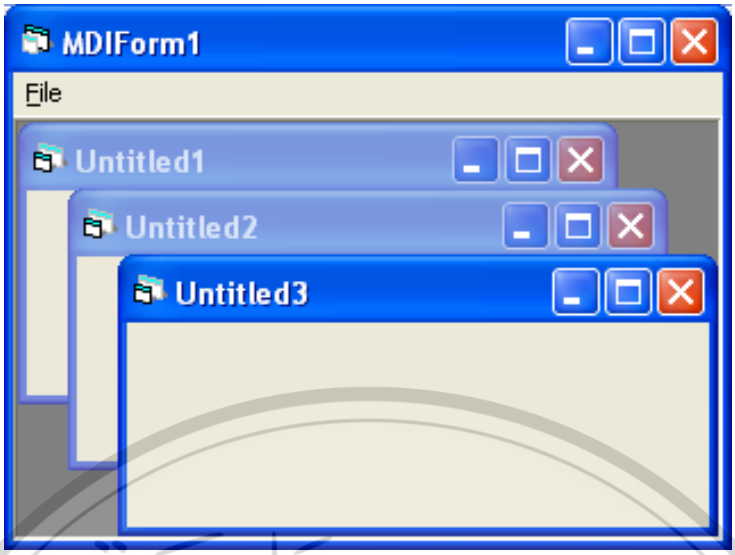
يظهر Form1 ثم Form2 ثم ... وهكذا.

- سنعطي هنا التسمية المطلوبة للمستند ونقوم بتمييزها بالرقم الملحق والدال على تسلسل النافذة المفتوحة.
- إذا افترضـــنا أن التســـمية العامة هي Untitled ونتبعها برقم فتصـــبح Untitled1 ثم Untitled2 وهكذا.

تتم هذه العملية من خلال إضافة كود معين إلى القائمة New نفسها كما يلي:

```
Private Sub mnuName_Click()
Static docName
Dim frmNew As New frmMain
If docName = 0 Then
docName = 1
frmNew.Show
MDIMain.ActiveForm.Caption
            = "Untitled" + CStr(docName)
Else
docName = docName + 1
frmNew.Show
MDIMain.ActiveForm.Caption
            = "Untitled" + CStr(docName)
End If
End Sub
```

- يقوم الإجراء الحدثي بتعريف متحول اسمه docName. ثم يصـرح عن متحول يدل على نموذج مواصفاته مثل النموذج القياسي frmMain.
- الحلقة If – Else – End If مسؤولة عن أخذ عداد يدل على رقم وتسلسل للنموذج الجديد وإظهار هذا النموذج ثم تغيير الخاصية Caption للنموذج كما في الشكل.



الكلمة المحجوزة Me:

تعتبر الكلمة Me من الكلمات المحجوزة في لغة الفيجوال بيزك وهي ببساطة عبارة عن متحول يحوي قيمة اسم النموذج الحالي (الذي يتم في الإجراء أو العمل).

الحادثة Resize:

تحصل الحادثة Resize عند ظهور النموذج أول مرة، وعند تغيير أبعاد النموذج من قبل المستخدم.

```
Private Sub MDIForm_Click()
Me.ActiveForm.Height = Me.ScaleHeight
Me.ActiveForm.Width = Me.ScaleWidth
End Sub
Private Sub mnuresize_Click()
Me.ActiveForm.Height = Me.ScaleHeight
Me.ActiveForm.Width = Me.ScaleWidth
End Sub
```

- نلاحظ من الإجراء الأول أنه عند ظهور النموذج MDIForm والذي دلت عليه الكلمة المحجوزة Me فإن الخاصية ScaleHeight للنموذج الفعال ستساوي الخاصية Height للنموذج MDIForm. والأمر نفسه ينطبق على العرض في السطر الثاني.
- نلاحظ من الإجراء الثاني أنه استعمال القائمة Resize فإن الخاصية ScaleHeight للنموذج الفعال ستساوي الخاصية Height للنموذج MDIForm والذي يتم فيه الإجراء الكودي والذي دلت عليه الكلمة المحجوزة Me. والأمر نفسه ينطبق على العرض في السطر الثاني.

- فـإن أبعـاد النموذج الفعـال MDIForm والـذي دلـت عليـه الكلمـة المحجوزة Me فـإن الخاصـــية ScaleHeight للنموذج الفعال ســـتســـاوي الخاصـــية Height للنموذج MDIForm. والأمر نفسه ينطبق على العرض في السطر الثاني.

الفصل الخامس عشر
مربعات الحوار الشائعة
Common Dialog Box

مقدمة:

تستخدم مربعات الحوار لإظهار والحصول على المعلومات من المستخدم. وتوجد ثلاثة أنواع من مربعات الحوار في VB وهذه الأنواع هي:

١. مربعات الحوار مسبقة التعريف Predefined Dialog Boxes أو (مربعات الحوار الجاهزة).
٢. مربعات الحوار المخصصة Custom Dialog Boxes.
٣. مربعات الحوار الشائعة Common Dialog Boxes.

مربعات الحوار مسبقة التعريف:

من اسمها هي مربعات معرفة مسبقاً بواسطة فيجوال بيزك. ولإظهار مربع مسبق التعريف، تستخدم عبارة ذات وسائط معينة تحدد كيف ومتى يتوجب إظهار مربع الحوار.

يمكن إظهار مربع حوار مسبق التعريف بواسطة ما يلي:

- عبارة MsgBox والتابع ()MsgBox.
- التابع الوظيفي ( )InputBox.

يمكن باستخدام العبارة MsgBox والتابع الوظيفي ()MsgBox بإظهار الرسائل للمستخدم والحصول على استجابته أو استجوابه (نتيجة الإستجواب هي إمّا نعم Yes أو لا No).

إظهار مربع حوار بواسطة العبارة MsgBox:

تستخدم العبارة MsgBox لإظهار مربعات حوار وتستخدم ثلاثة وسائط كما يلي:

MsgBox Message, ButtonsAndIcon, Title

حيث:

- Message هي الرسالة المطلوب إظهارها (وهي سلسلة كتابية) ويستخدم كأول وسيط في العبارة MsgBox.
- ButtonsAndIcon وهي الأزرار والرموز المراد ظهورها في مربع الرسالة وهو المتحول الثاني في العبارة MsgBox وهو عبارة عن متحول (قيمة عددية صحيحة من النوع Integer) ومن خلال هذه القيمة نستطيع تحديد الزر أو الأزرار المطلوب إظهارها على مربع الحوار بالإضافة إلى رمز أو إشارة (أيقونة) تظهر على مربع الحوار.
- Title وهي عبارة عن عنوان مربع الرسالة (وهي سلسلة كتابية) وهو الوسيط الثالث في العبارة MsgBox.

إن الشكل السابق هو الشكل البسيط لإستخدام الوظيفة MsgBox إلا أن الشكل الأكثر شمولية هو:

$$MsgBox(prompt[, buttons][, title][, helpfile, context])$$

أما معنى هذه الوسائط فهو:

- prompt: هي عبارة عن تعبير نصي والذي يُظهِر مضمون الرسالة التي ستظهر في مربع الحوار. إن الطول الأعظمي يجب أن لايزيد عن 1024 حرفاً والذي يعتمد على عرض الحروف المستخدمة. إذا كان طول الرسالة أكبر من طول خط ما نستطيع تجزئة الرسالة إلى أسطر باستخدام التابعين $Chr(13)$ & $Chr(10)$ واللذان يقومان بفتح سطر جديد والإنتقال بالمؤشر إلى بداية السطر التالي.
- buttons: وهو تعبير عددي قيمته صحيحة والذي سيحدد عدد ونوع الأيقونات التي ستظهر على مربع الرسالة. عند إهمال هذه القيمة سيعتبرها VB مساوية للصفر أي Buttons = 0.
- Title – عنوان مربع الرسالة الذي سيظهر في منطقة الإسم. عند إهمال هذه القيمة سيظهر اسم التطبيق في مكانها.
- helpfile – اسم ملف المساعدة الذي سيظهر مع مربع الرسالة. نلاحظ أن استعمال هذا الوسيط يحتم علينا استعمال الوسيط الأخير context.

- context – تعبير عددي يدل يشـــير إلى عبارة المســـاعدة التي يبينها الوســـيط helpfile وهو مرتبط بوجوده أيضاً.

إن القيم التي تأخذها الأزرار buttons تكون كما يلي:

Constant	Value	Description
vbOKOnly	0	Display OK button only.
vbOKCancel	1	Display OK and Cancel buttons.
vbAbortRetryIgnore	2	Display Abort, Retry, and Ignore buttons.
vbYesNoCancel	3	Display Yes, No, and Cancel buttons.
vbYesNo	4	Display Yes and No buttons.
vbRetryCancel	5	Display Retry and Cancel buttons.
vbCritical	16	Display Critical Message icon.
vbQuestion	32	Display Warning Query icon.
vbExclamation	48	Display Warning Message icon.
vbInformation	64	Display Information Message icon.
vbDefaultButton1	0	First button is default.
vbDefaultButton2	256	Second button is default.
vbDefaultButton3	512	Third button is default.
vbDefaultButton4	768	Fourth button is default.
vbApplicationModal	0	Application modal; the user must respond to the message box before continuing work in the current application.
vbSystemModal	4096	System modal; all applications are suspended until the user responds to the message box.

ويمكن أن يكون هناك قيمة من مجموعتين أو أكثر ولا يمكن أن يكون هناك قيمة من جمع قيمتين من مجموعة واحدة.

يمكن تقسيم هذه الرموز إلى عدة مجموعات:

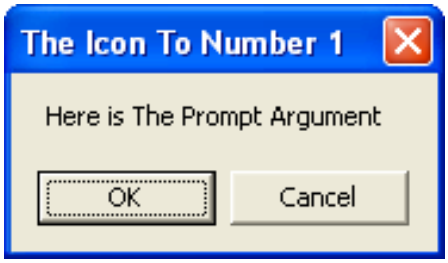
١. مجموعة الأرقام 0, 1, 2, 3, 4, 5 والتي تصف عدد ونوع الأزرار على مربع الحوار.
٢. المجموعة 16, 32, 48, 64 والتي تصف نمط الأيقونة على مربع الحوار.
٣. المجموعة 0, 256, 512, 768 تحدد أي الأزرار سيكون إفتراضياً.
٤. المجموعة 0, 4096 تحدد فيما إذا كان مربع الحوار من النوع Modal أم لا أي هل من الممكن تجاوزه دون المرور عليه والإنتقال إلى خيارات أخرى أم لا.

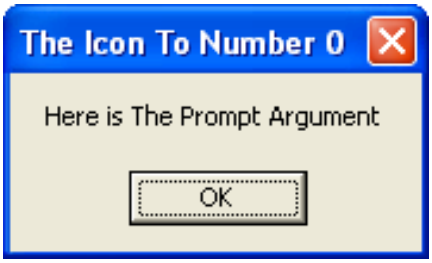
ملاحظات:

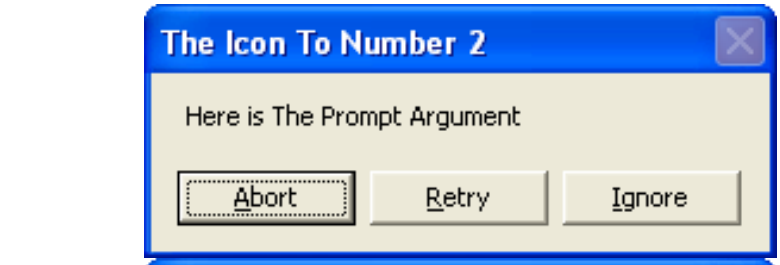
- لا يمكن استخدام إلا رقماً واحداً من كل مجموعة من المجموعات.
- عند تزويد مربع الحوار بالقيمتين helpfile و context نستطيع الضغط على المفتاح F1 للحصول على المساعدة أي أن الضغط على المفتاح سيأخذنا إلى ملف المساعدة.
- إذا ظهر الزر Cancel على مربع حوار معين فإننا نستطيع باستخدام المفتاح Esc الحصول على نفس النتيجة التي يقدمها المفتاح Cancel.
- إذا إحتوى مربع الحوار على زر المساعدة Help فإذا استخدمنا المساعدة أم لم نستخدمها فإن مربع الحوار لن يأخذ قيمة ويتابع البرنامج إلا إذا ضغطنا على زراً آخراً.

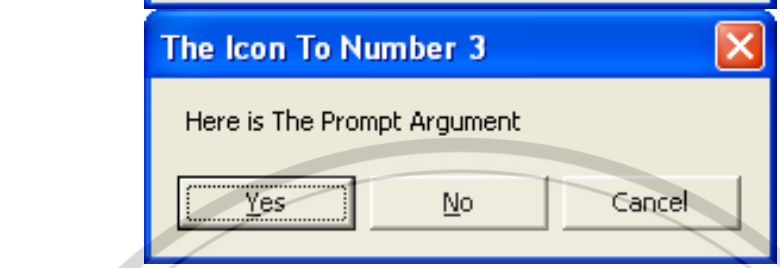
وتكون أشكال مربعات الحوار الموافقة لهذه الأرقام كما يلي:

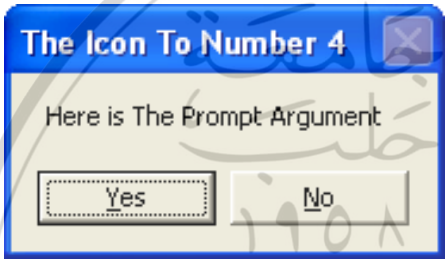
المجموعة الأولى :

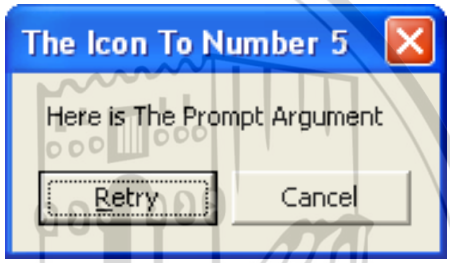






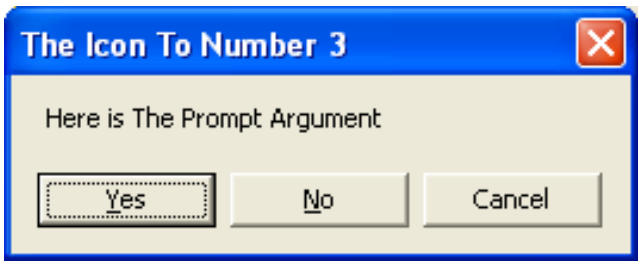


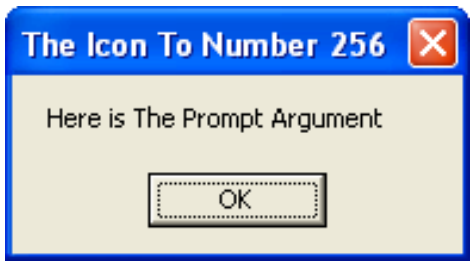


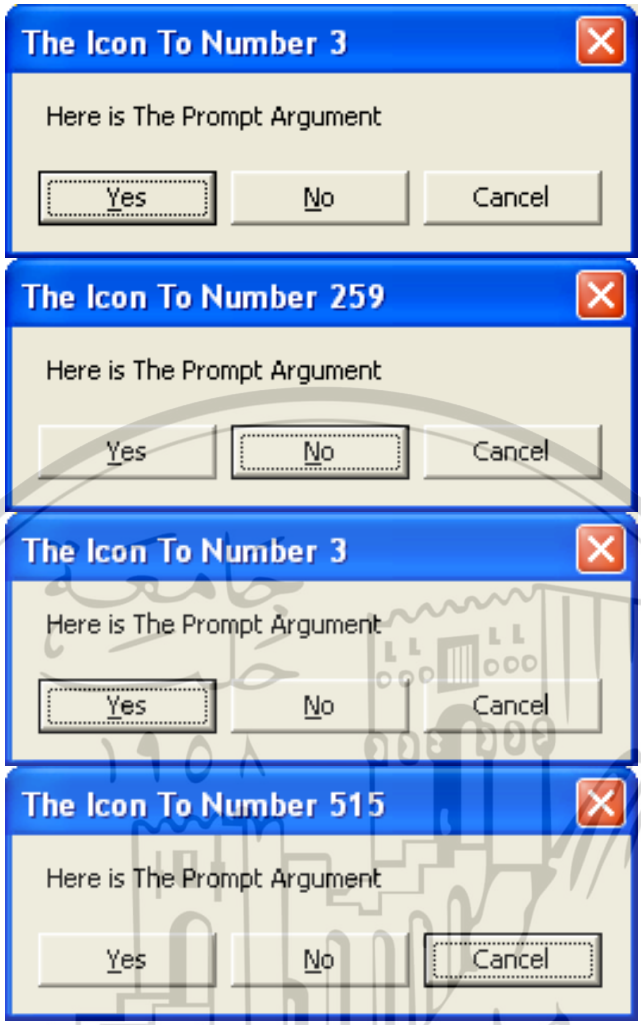


المجموعة الثانية:

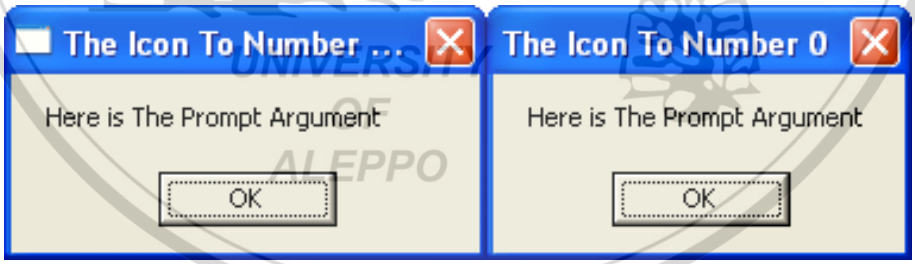
المجموعة الثالثة:







المجموعة الرابعة :



يبين الجدول التالي ثوابت أزرار مربع حوار الرسالة MsgBox:

Constant	Value	Button
vbOKOnly	0	موافق
vbOkCancel	1	موافق وإلغاء
vbAbortRetryIgnore	2	إحباط وإعادة المحاولة وتجاهل

vbYesNoCancel	3	نعم و لا وإلغاء
vbYesNoCancel	4	نعم و لا
vbRetryCancel	5	إعادة المحاولة وإلغاء

كما يبين الجدول التالي الثوابت المستخدمة في مربع حوار الرسالة MsgBox:

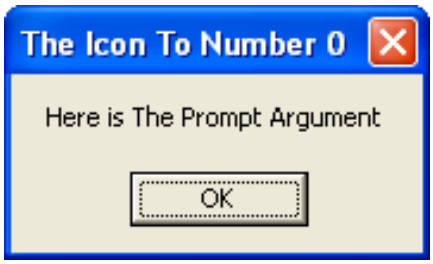
	اسم الثابت وأيقونته	القيمة Value	معنى الثابت أو وصفه
	vbCritical	16	Display Critical Message icon. يعرض رمز الخطأ الحرج
	vbQuestion	32	Display Warning Query icon. يعرض رمز إشارة الاستفهام
	vbExclamation	48	Display Warning Message icon. يعرض رمز التعجب
	vbInformation	64	Display Information Message icon. يعرض رمز المعلومات

يمكن تشكيل وسائط buttons من خلال هذه الوسائط شرط أن لا يكون هناك وسيطين من مجموعة واحدة.

مثلاً يمكن أن نرى أشكال مربعات الحوار الناتجة من جمع الوسيط 64 من المجموعة الثانية مع عناصر المجموعة الأولى:

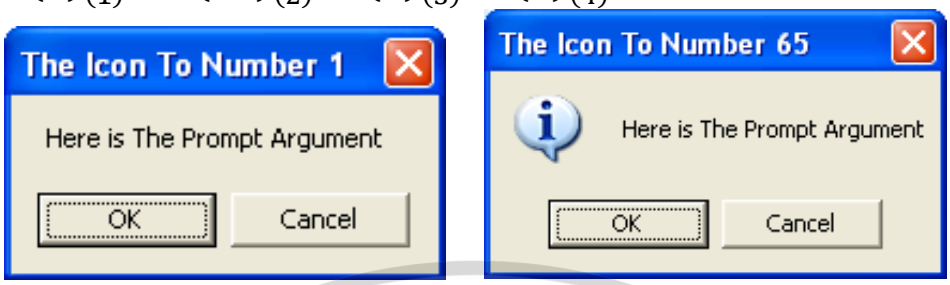
١. جمع الوسيط 64 مع القيمة 0 من المجموعة الأولى:

$$64 = (0)_{(1)} + (64)_{(2)} + (0)_{(3)} + (0)_{(4)}$$



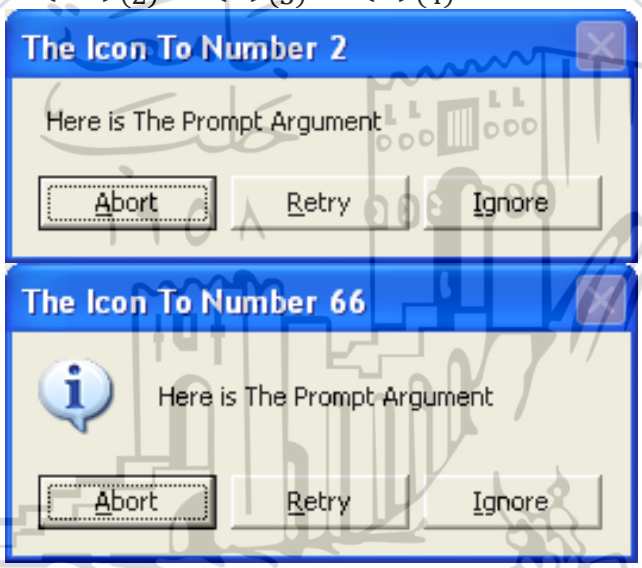
٢. جمع الوسيط 64 مع القيمة 1 من المجموعة الأولى:

$$65 = (1)_{(1)} + (64)_{(2)} + (0)_{(3)} + (0)_{(4)}$$



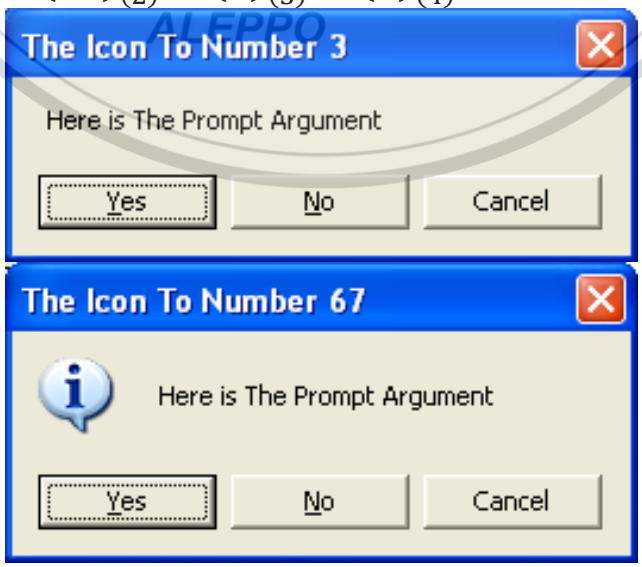
٣. جمع الوسيط 64 مع القيمة 2 من المجموعة الأولى:

$$66 = (2)_{(1)} + (64)_{(2)} + (0)_{(3)} + (0)_{(4)}$$



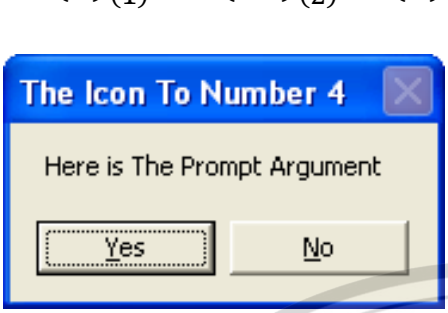
٤. جمع الوسيط 64 مع القيمة 3 من المجموعة الأولى:

$$67 = (3)_{(1)} + (64)_{(2)} + (0)_{(3)} + (0)_{(4)}$$



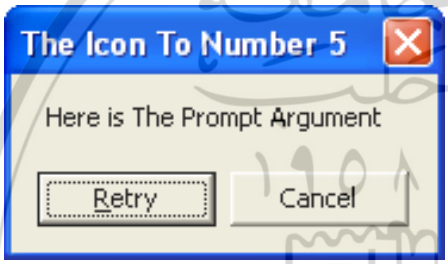
٥. جمع الوسيط 64 مع القيمة 4 من المجموعة الأولى:

$$68 = (4)_{(1)} + (64)_{(2)} + (0)_{(3)} + (0)_{(4)}$$



٦. جمع الوسيط 64 مع القيمة 5 من المجموعة الأولى:

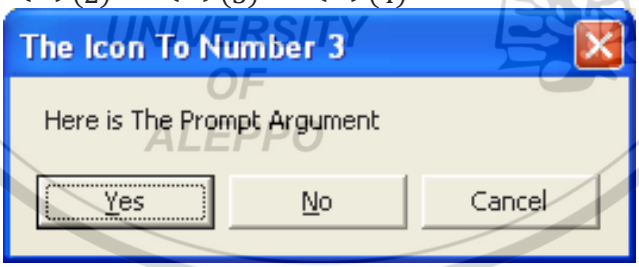
$$69 = (5)_{(1)} + (64)_{(2)} + (0)_{(3)} + (0)_{(4)}$$



أيضاً يمكن أن نرى رأينا أشكال مربعات الحوار الناتجة من جمع الوسيط 3 من المجموعة الأولى مع عناصر المجموعة الثالثة كما يلي:

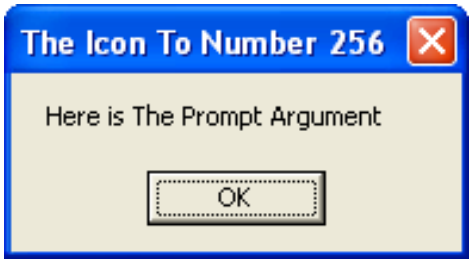
١. القيمة 3 من المجموعة الأولى تظهر كما يلي:

$$3 = (3)_{(1)} + (0)_{(2)} + (0)_{(3)} + (0)_{(4)}$$



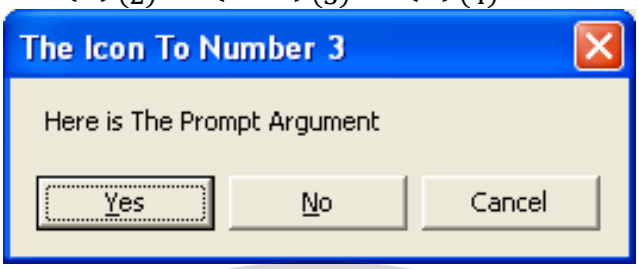
٢. القيمة 256 من المجموعة الثالثة تظهر كما يلي:

$$256 = (0)_{(1)} + (0)_{(2)} + (256)_{(3)} + (0)_{(4)}$$



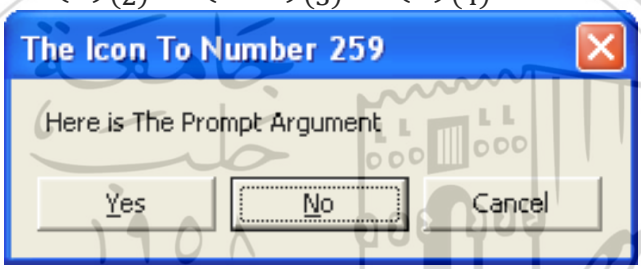
٣. جمع الوسيط 0 من المجموعة الثالثة مع القيمة 3 من المجموعة الأولى:

$$259 = (3)_{(1)} + (0)_{(2)} + (256)_{(3)} + (0)_{(4)}$$



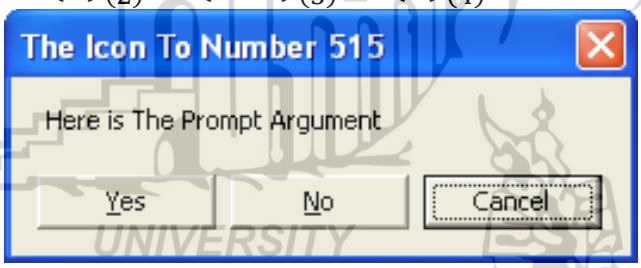
٤. جمع الوسيط 256 من المجموعة الثالثة مع القيمة 3 من المجموعة الأولى:

$$259 = (3)_{(1)} + (0)_{(2)} + (256)_{(3)} + (0)_{(4)}$$



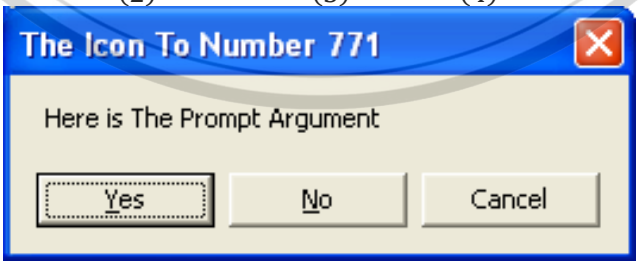
٥. جمع الوسيط 512 من المجموعة الثالثة مع القيمة 3 من المجموعة الأولى:

$$515 = (3)_{(1)} + (0)_{(2)} + (512)_{(3)} + (0)_{(4)}$$



٦. جمع الوسيط 768 من المجموعة الثالثة مع القيمة 3 من المجموعة الأولى:

$$771 = (3)_{(1)} + (0)_{(2)} + (768)_{(3)} + (0)_{(4)}$$



إن استخدام الوسيط سينقل التركيز إلى الزر الرابع، وعلى اعتبار وجود ثلاثة أزرار فإنَّه تم نقلالتركيز إلى الزر الأول. وهذا يعني انعدام تأثير الوسيط الرابع من المجموعة الثالثة بسبب وجود ثلاثة أزرار فقط.

أيضاً يمكن أن نرى رأينا أشكال مربعات الحوار الناتجة من جمع الوسيط 4096 من المجموعة الرابعة مع بعض عناصر المجموعة الأولى كما يلي:

١. جمع الوسيط 4096 من المجموعة الرابعة مع القيمة 0 من المجموعة الأولى:

$$4096 = (0)_{(1)} + (0)_{(2)} + 0_{(3)} + (4096)_{(4)}$$

٢. جمع الوسيط 4096 من المجموعة الرابعة مع القيمة 1 من المجموعة الأولى:

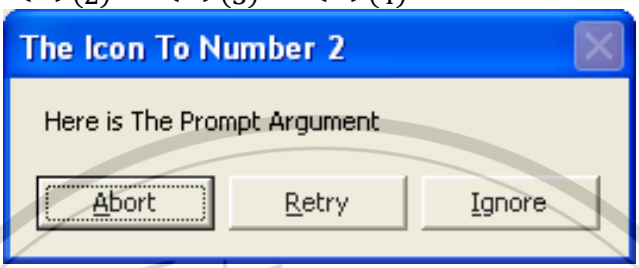
٣. جمع الوسيط 4096 من المجموعة الرابعة مع القيمة 2 من المجموعة الأولى:

٤. جمع الوسيط 4096 من المجموعة الرابعة مع القيمة 3 من المجموعة الأولى:

كما يمكن أن نرى أشكال مربعات الحوار الناتجة من جمع عدة وسائط كما في المثال التالي:

١. القيمة 2 من المجموعة الأولى تظهر كما يلي:

$$2 = (2)_{(1)} + (0)_{(2)} + (0)_{(3)} + (0)_{(4)}$$



٢. القيمة 32 من المجموعة الثانية تظهر كما يلي:

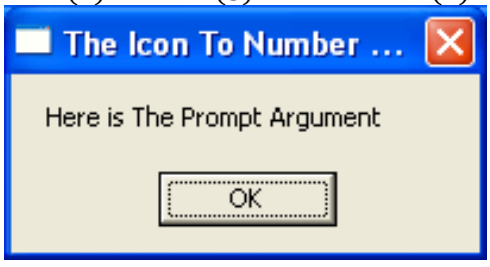
$$32 = (0)_{(1)} + (32)_{(2)} + (0)_{(3)} + (0)_{(4)}$$

٣. القيمة 512 من المجموعة الثالثة تظهر كما يلي:

$$512 = (0)_{(1)} + (0)_{(2)} + (512)_{(3)} + (0)_{(4)}$$

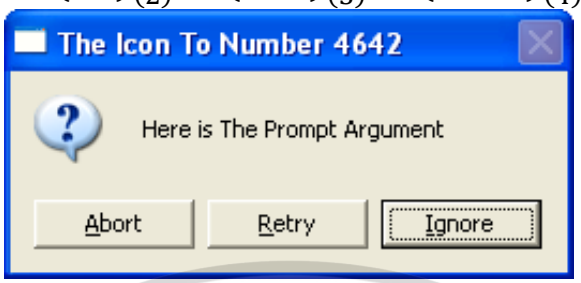
٤. القيمة 4096 من المجموعة الرابعة تظهر كما يلي:

$$4096 = (0)_{(1)} + (0)_{(2)} + (0)_{(3)} + (4096)_{(4)}$$



٥. القيمة 4642 ستكون عبارة عن جمع الأرقام السابقة كلها وستظهر كما يلي:

$$4642 = (2)_{(1)} + (32)_{(2)} + (512)_{(3)} + (4096)_{(4)}$$



إظهار مربع حوار بواسطة التابع الوظيفي MsgBox():

يستخدم هذا التابع لإظهار مربع حوار يحمل إشارة وأزرار ويأخذ نفس الوسائط التي تأخذها العبارة MsgBox والإختلاف الوحيد بينهما هو أن التابع MsgBox() يعيد قيمة تشير إلى الزر الذي تم النقر عليه في مربع الحوار (أي أن عملية النقر على اي زر ستحدد إتجاه عمل البرنامج أو ستحدد وجهة البرنامج).

$$MsgBox(prompt[, buttons][, title][, helpfile, context])$$

يبين الجدول التالي ثوابت أزرار مربع حوار التابع الوظيفي MsgBox():

Constant	Value	Button	
vbOK	1	OK	موافق
vbCancel	2	Cancel	إلغاء
vbAbort	3	Abort	إحباط
vbRetry	4	Retry	إعادة المحاولة
vbIgnore	5	Ignore	تجاهل
vbYes	6	Yes	نعم
vbNo	7	No	لا

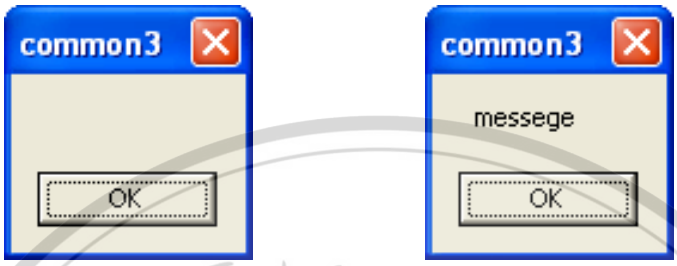
ملاحظات:

- عند استخدام العبارة MsgBox لا يتم إحاطة الوسائط بقوسين بعكس التابع الوظيفي MsgBox() الذي يتوجب فيه إغلاق الوسائط بقوسين.
- يجب استخدام القيمة المعادة من قبل التابع الوظيفي لإسنادها إلى متحول.
- تدل القيمة التي يعيدها التابع MsgBox() على الزر الذي تم النقر أو الضـــغط عليه.

- يؤدي عدم تحديد الوســيطين الثاني والثالث لعبارة MsgBox إلى ظهور مربع حوار يحمل زر موافق فقط ودون أي رمز.

نلاحظ أن مربع الحوار أخذ اسم المشروع وذلك لعدم وجود قيمة للوسيط Title:

$$A = MsgBox(" ") \quad A = MsgBox(" messege")$$



التابع الوظيفي InputBox():

يمكن استخدام التابع الوظيفي لتلقي معلومات كتابية من قبل المستخدم. يُظهِر التابع الوظيفي InputBox() مربع حوار مع رسالة وزرين هما Ok و Cancel ويستطيع المستخدم إدخال نص ما في الحقل النصي وإغلاق مربع الحوار بالنقر على الزر Ok.

الوسيط الأول من التابع الوظيفي هو رسالة مربع الحوار والوسيط الثاني هو عنوان مربع الحوار.

يعيد التابع الوظيفي InputBox() ما أدخله المستخدم في الحقل النصّي. إن القيمة المعادة تساوي لا شيء "null" إذا لم يكتب المستخدم اي شيء في مربع الإدخال أو عند الضغط على زر Cancel "إلغاء الأمر".

تستخدم العبارة If للتحقق من شرط الإدخال ومن القيمة المُدخَلَة.

استخدام الوسائط الأخرى للتابع InputBox():

يمكن استخدام وسائط إختيارية أخرى مع التابع المذكور كما يلي:

$$Number = InputBox\ ("رقماً أدخل", "مثال", "7", 100, 200)$$

- عنوان المربع هو أدخل رقماً.
- عنوان العبارة التي ستظهر هي مثال.
- القيمة الإفتراضية التي ستظهر في الحقل النصي هي 7.
- تبعد الحافة اليسارية لمربع الحوار بمقدار 100 twips عن الحافة اليسارية للشاشة.

- تبعد الحافة العلوية لمربع الحوار بمقدار 200 twips عن قمة للشاشة.

نتيجة ذلك سيظهر مربع الحوار عند احداثيي الشاشة $x = 100$ و $y = 200$ كما أن القيمة الإفتراضية الموجودة في الحقل النصّي تساوي 7. يمكن للمستخدم النقر على Ok في مربع الحوار أو يمكنه إدخال رقماً آخر بدلاً من الرقم 7 ويتبعه بالنقر على Ok.

مربعات الحوار المخصصة:

يتم تصميم مربعات الحوار المخصصة من قبل المستخدم (بينما كانت مربعات الحوار مسبقة التعريف، تسمح بوضع أشياء محددة مثل الأزرار والرموز).

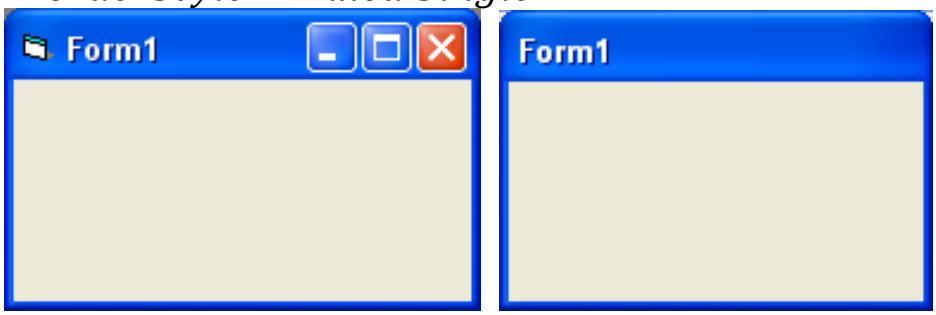
إن عملية تصميم مربع حوار مشابهة لعملية تصميم النماذج. في الواقع إن مربع الحوار المخصص هو نموذج عادي يستخدم لإظهار المعلومات للمستخدم أو إظهار المعلومات للمستخدم وبمجرد الإنتهاء من تصميم مربع الحوار المخصص، يصبح بالإمكان استخدامه من قبل أي برنامج مكتوب بلغة VB.

خصائص مناسبة لمربعات الحوار:

من المهم أن يحافظ دوماً مربع الحوار على شكله وخصائصه لذا يجب أن نقوم بتغيير بعض الخصائص له بحيث نمنع تغيير شكل ظهوره أثناء زمن التنفيذ.

الخاصية BorderStyle:

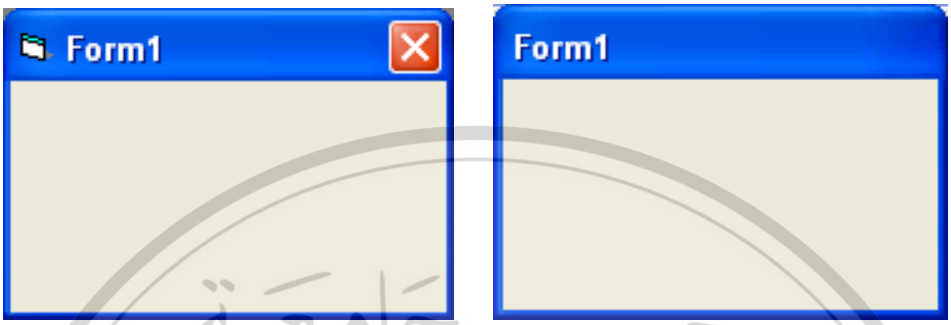
$$Form.BorderStyle = Fixed\ Single$$



إن جعل خاصية النموذج بهذا الشكل لا يسمح بتغيير حجمه (حجم مربع الحوار) أثناء زمن التنفيذ.

الخاصية ControlBox:

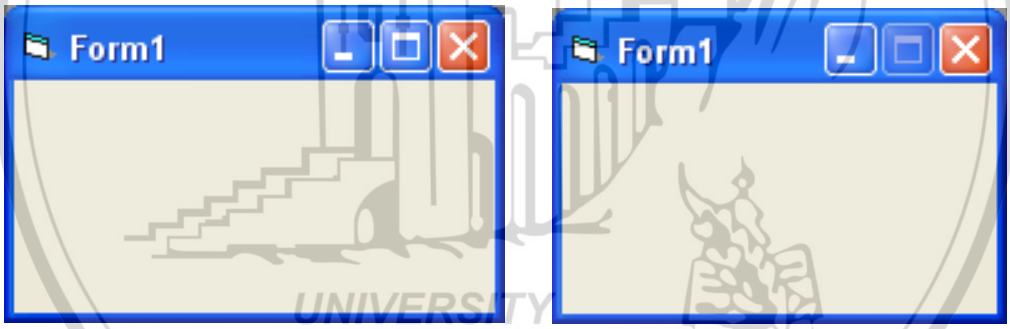
$Form.ControlBox = False$



هذا يعني أن النموذج (مربع الحوار) سيظهر بدون مربع قائمة التحكم أي Control Menu Box (وهو ذلك المربع الذي يظهر في الزاوية اليسرى العليا من النموذج).

الخاصية MaxButton:

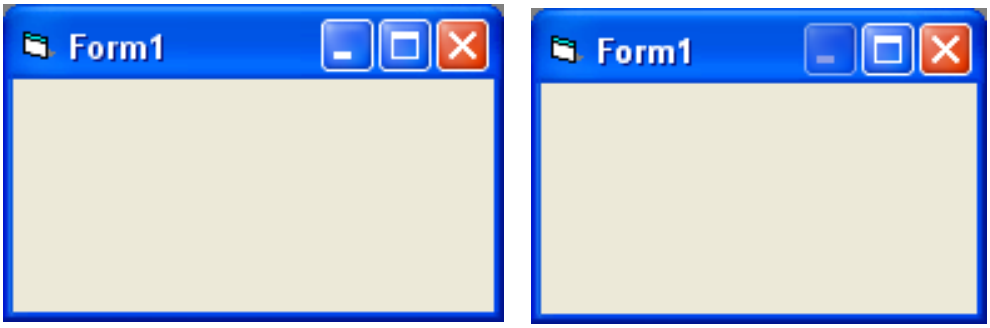
$Form.MaxButton = False$



هذا يعني أن النموذج (مربع الحوار) سيظهر بدون زر التكبير وهو الزر الأوسط في الزاوية العليا اليمنى وهذا ما سيمنع المستخدم من تكبير مربع الحوار أثناء زمن التنفيذ.

الخاصية MinButton:

$Form.MinButton = False$



هذا يعني أن النموذج (مربع الحوار) سيظهر بدون زر التصغير في زاويته العليا اليمنى وهذا ما سيمنع المستخدم من تصغير مربع الحوار أثناء زمن التنفيذ.

الخاصيتان Cancel و Default لأزرار الأوامر:

نضيف للنموذج زري أوامر ونقوم بتغيير خاصية Default لأحدهما وخاصية Cancel للآخر كما يلي:

$$Form.Default = True$$
$$Form.Cancel = True$$

يؤدي تعيين الخاصية Default لزر الأمر على القيمة True إلى جعل زر الأمر وكأنه زر إفتراضي Default وهو ذلك الزر الذي يُعتبر أنه قد نقر على عليه إذا تم الضغط على المفتاح Enter.

يؤدي تعيين الخاصية Cancel لزر الأمر على القيمة True إلى جعل هذا الزر بمثابة مفتاح الهروب Escape (وهو الزر الذي يتم إختياره نتيجة الضغط على المفتاح Esc الموجود في لوحة المفاتيح).

إظهار وإخفاء مربع الحوار:

يمكن إظهار مربع الحوار وإخفائه بالأوامر:

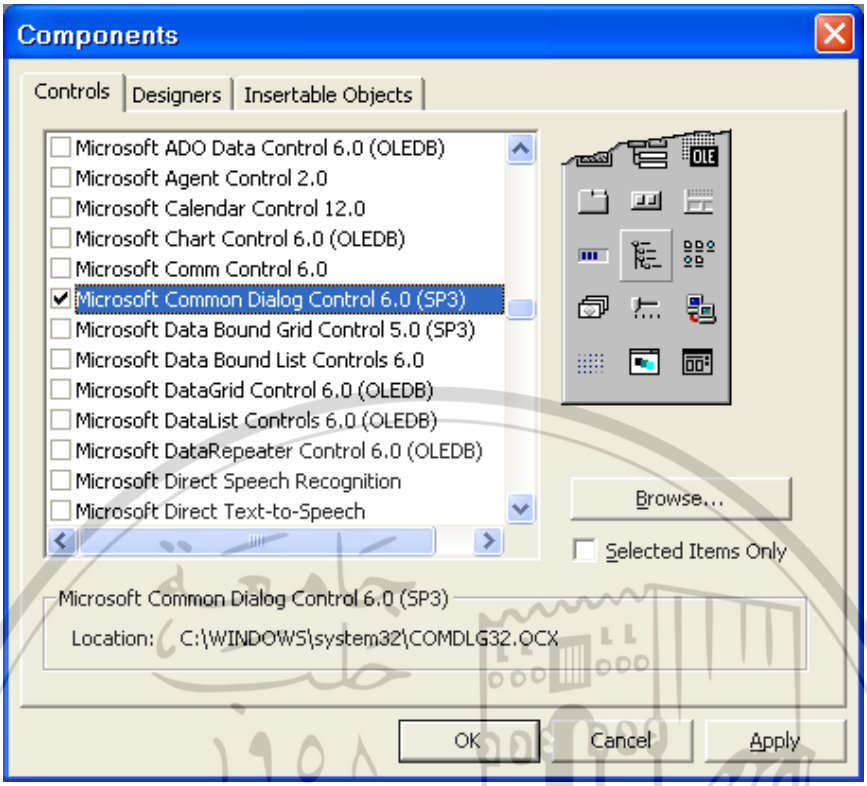
$$Form.Show$$
$$Form.Hide$$

المفروض في مربع حوار أن لا يسمح للبرنامج بأن يستكمل العمل دون المرور عليه أو التعامل معه لذا سنجعله من النوع Modal (أي أن المستخدم يجب أن يستجيب له قبل أن يتمكن من استئناف تنفيذ البرنامج) ويمكن أن نستخدم وسيط يعبر عن ذلك كما يلي:

$$Form.Show\ 1$$

مربعات الحوار الشائعة Common Dialog Box: 

يتضمن VB عنصر تحكم اسمه CommonDialog (أي مربعات الحوار الشائعة). يمكن استخدام هذا العنصر لإظهار مربعات الحوار الشائعة أثناء زمن التنفيذ بمجرد تعيين خصائصها. يستخدم مربع الحوار الشائع غالباً كمربع حوار يسمح للمستخدم باختيار وحفظ الملفات.



يتم إضافة هذه الأدام من خلال القائمة Project → Components بعد ذلك يظهر مربع الحوار كما في الشكل التالي :

نختار الخيار :

Microsoft Common Dialog Control 6.0 (Sp3)

ثم Ok فتُضاف أداة مربع الحوارات الشائعة إلى شريط الأدوات كما في الشكل:

بعد ذلك وكما في أي أداة نقوم بوضع هذه الأداة على النموذج وعندها سيظهر اسمها الإفتراضي كما يلي: CommonDialog1 وإذا وضعنا الأداة مرة ثانية سيكون اسمها CommonDialog2 ... الخ.

أنواع مربعات المستخدمة The Common Dialog Box Methods:

Method	Dialog displayed	Action Property
ShowOpen	Open	1
ShowSave	Save As	2
ShowColor	Color	3
ShowFont	Font	4
ShowPrinter	Print	5

مربعات حوار الملفات Adding the File Dialog Boxes:

إن أدوات مربعات الحوار لا تظهر أثناء المرحلة التنفيذية ولكن تظهر فوراً أثناء استدعائها من قبل حدث أو إجراء معين.

فتح ملف Open File:

```
Private Sub mnuOpen_ Click ( )
Dim Filter As String
On Error GoTo OpenError
Filter = "كل الملفات (*.*) | *.*|"
Filter = Filter + "الملفات النصية (*.txt) | *. txt |"
Filter = Filter + "الملفات الدفعية (*.bat) | *. bat |"
CommonDialog1.Filter = Filter
CommonDialog1.FilterIndex = 2
CommonDialog1. ShowOpen
MsgBox "لقد اخترت الملف: " & CommonDialog1.filename
Exit Sub
OpenError:
MsgBox "لقد ألغيت مربع حوار فتح ملف"
Exit Sub
End Sub
```

تشغيل متصيد الأخطاء لمعرفة أن المستخدم ضغط زر إلغاء الأمر في مربع حوار فتح ملف:

Private Sub mnuOpen_ Click ( )

تحديد أنواع الملفات المراد فتحها:

Filter = "كل الملفات (.*) | *.*|"*

Filter=Filter+"الملفات النصية (.txt) | *. txt |"*

Filter=Filter+"الملفات الدفعية (.bat) | *. bat |"*

$CommonDialog1.Filter = Filter$

تحديد نوع الملفات الإفتراضي وهو الملفات النصية:

$CommonDialog1.FilterIndex = 2$

إظهار مربع حوار فتح ملف:

$CommonDialog1.ShowOpen$

إظهار اسم الملف الذي تم اختياره:

$MsgBox$ "الملف اختر لقد : " & $CommonDialog1.filename$

الخروج من الإجراء:

$Exit\ Sub$

الكود OpenError:

ينفذ هذا الجزء من الإجراء عندما يضغط المستخدم على زر إلغاء الأمر.

ظهور مربع حوار مضمونه: "لقد ألغيت مربع حوار فتح ملف"

MsgBox "لقد ألغيت مربع حوار فتح ملف"

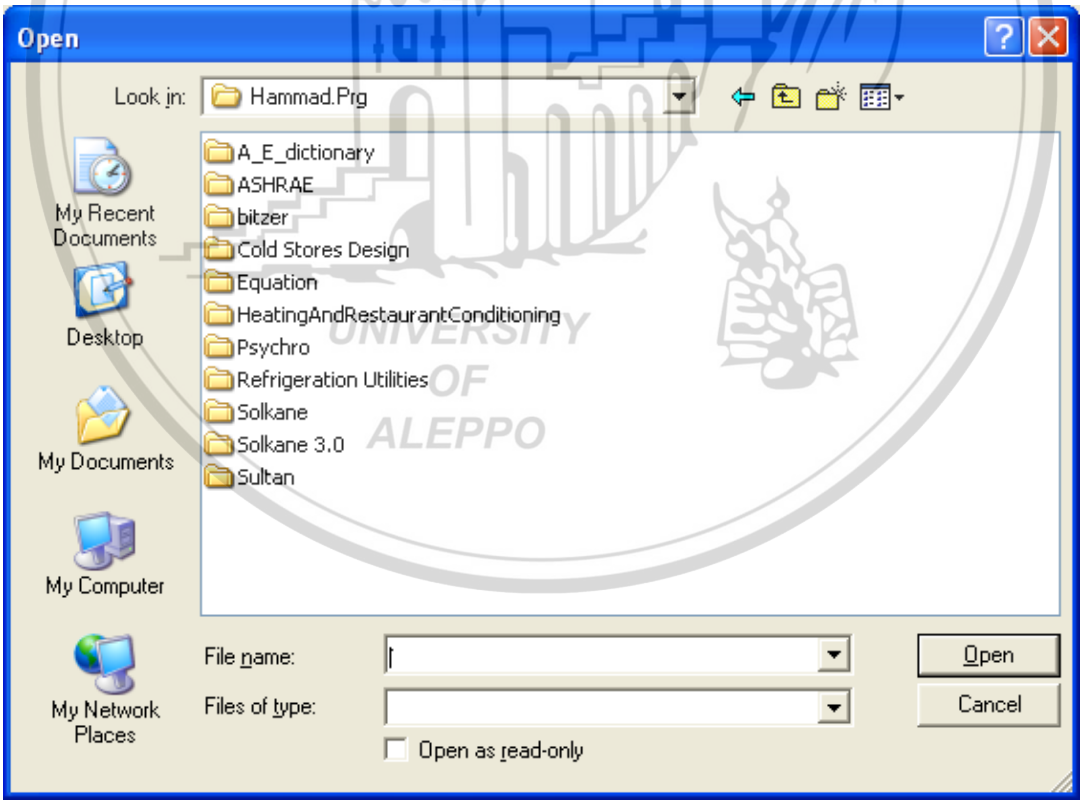
الخروج من الإجراء:

$Exit\ Sub$

إنهاء الإجراء:

End Sub

يستجيب البرنامج بإظهار مربع الحوار فتح.



عند إختيار دليل محدد من القرص الصلب، واختيار البند كل الملفات من مربع الملفات وبعد أن نكتب *.* في مربع اسم الملف ومن ثم نختار ملفاً وننقر على الزر فتح في مربع

الحوار فتح، عندها سيستجيب البرنامج ويقوم بإظهار رسالة تخبرنا عن اسم الملف الذي اخترناه.

نتيجة مرحلة التصيد فإن النقر على زر إلغاء الأمر سيؤدي أثناء ظهور مربع الحوار فتح إلى توليد خطأ أثناء مرحلة التنفيذ مما يؤدي إلى إنتقال التنفيذ إلى اللافتة OpenError.

يستخدم البرنامج بعد مرحلة التصيد هذه الخاصية Filter لعنصر التحكم CommonDialog1 لملء بنود مربع سرد أنواع الملفات الموجودة في مربع فتح.

يُملأ مربع سرد أنواع الملفات بثلاثة أنواع:

```
Filter = "كل الملفات (*.*) | *.*|"
Filter = Filter + "الملفات النصية (*.txt) | *. txt |"
Filter = Filter + "الملفات الدفعية (*.bat) | *. bat |"
CommonDialog1.Filter = Filter
```

تُملأ العبارات السابقة الخاصية Filter لعنصر التحكم CommonDialog1 بسلسلة تُحَدِّد نصوص البنود التي ستظهر في مربع سرد الملفات، إضافة إلى أنواع هذه الملفات (*.bat, *.txt, *.*).

مثلاً:

إن أول بند يظهر في مربع سرد الملفات بالسلسلة التالية:

```
Filter = "كل الملفات (*.*) | *.*|"
```

تتألف السلسلة من جزأين مفصولان بالحرف (|) (يظهر باستخدام [\ + Shift].

يشير الجزء الأول من السلسلة (*.*) كل الملفات إلى النص الذي يظهر في مربع سرد الملفات. وبمعنى آخر سيؤدي اختيار هذا البند إلى ظهور الملفات ذات الهيكلية (*.*) كل الملفات في مربع الحوار.

قامت العبارات السابقة بملء الخاصية Filter لعنصر التحكم CommonDialog1 على مراحل وذلك بهدف التوضيح.

بعد ذلك أسندت قيمة المتحول Filter هذا إلى الخاصية Filter لعنصر التحكم CommonDialog1.

بعد الإنتهاء من ملء مربع سرد الملفات الموجود في مربع الحوار فتح، سيتم تحديد البند الإفتراضي الذي سيظهر في مربع سرد الملفات وهنا أخذنا البند الثاني وذلك بعد ما أخذنا القيمة 2 إلى الخاصية FilterIndex لعنصر مربع الحوار CommonDialog1.

وباعتبار أن البند الثاني في مربع سرد الملفات هو الملفات النصية (*.txt) فالملفات الإفتراضية التي ستندرج في مربع الحوار فتح هي الملفات ذات الإمتداد txt.

إذا تم نقر زر إلغاء الأمر أثناء ظهور مربع الحوار فإن البرنامج سيُنَفِّذ الجزء الواقع تحت اللافتة OpenError والذي سيُظهِر رسالة من خلال MsgBox ومن ثم يقوم بإنهاء البرنامج.

أما الجزء الثاني *.* فيشير إلى هيكلية الملفات التي ستظهر عند إختيار هذا البند أي عند إختيار *.*.

مربعات حوار الألوان The Color Dialog Box:

```
Private Sub mnuColor_Click()
On Error GoTo ColorError
CommonDialog1.ShowColor
Form1.BackColor = CommonDialog1.Color
Exit Sub
ColorError:
MsgBox "لقد ألغيت مربع حوار الألوان"
Exit Sub
End Sub
```

تشغيل متصيد الأخطاء لمعرفة أن المستخدم ضغط:

Private Sub mnuColor_Click( )

زر إلغاء الأمر في مربع حوار الألوان :

On Error GoTo ColorError

'إظهار مربع حوار الألوان:

CommonDialog1.ShowColor

'تغيير لون خلفية البرنامج إلى اللون الذي تم اختياره:

$Form1.BackColor = CommonDialog1.Color$

الخروج من الإجراء:

$Exit\ Sub$

الكود OpenError:

ينفذ هذا الجزء من الإجراء عندما يضغط المستخدم على زر إلغاء الأمر.

ظهور مربع حوار مضمونه: "لقد ألغيت مربع حوار الألوان"

$MsgBox$ "لقد ألغيت مربع حوار الألوان"

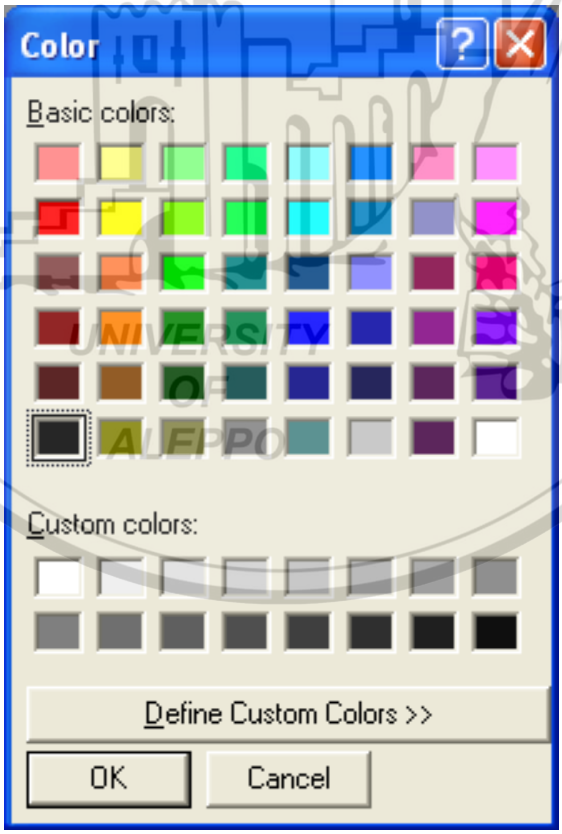
الخروج من الإجراء:

$Exit\ Sub$

إنهاء الإجراء:

End Sub

يستجيب البرنامج عند الطلب بإظهار مربع الحوار لون ونستطيع إنتقاء أي لون من الألوان منه.



إن وظيفة المصيدة On Error GoTo ColorError هي مراقبة (أو تصيد) الخطأ أثناء إظهاره مربع الحوار.

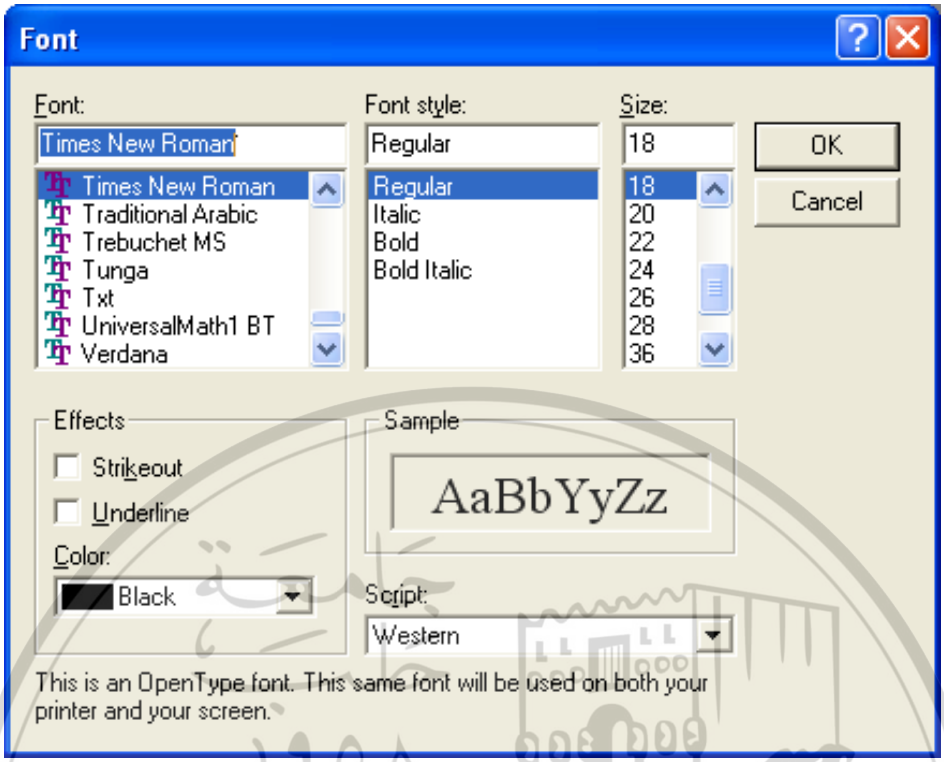
لنفترض جدلاً أننا وضعنا الخاصية CancelError لعنصر التحكم المخصص CommonDialog1 أثناء المرحلة المرئية (التصميمية) على الحالة True وهكذا سيؤدي ضغط الزر إلغاء الأمر أثناء ظهور مربع الحوار إلى توليد خطأ. ونتيجة إصطياد الخطأ سيتحول تنفيذ البرنامج إلى اللافتة ColorError.

يؤدي نقر الزر إلغاء الأمر أثناء ظهور مربع الحوار إلى توليد خطأ، ونتيجة لذلك سيُنَفَّذ الإجراء الواقع تحت اللافتة ColorError عنها سيُظهِر البرنامج الرسالة المكتوبة في MsgBox ثم يُنهِي البرنامج.

إن لم يكن هناك أخطاء فإن البرنامج سيتخطّى عبارة التصيد وسينتقل إلى إظهار مربع الحوار لون. بعد إنتقاء اللون سيستجيب مربع الحوار ويقوم بإغلاق مربع الحوار لون وتبديل لون النموذج إلى اللون المطلوب.

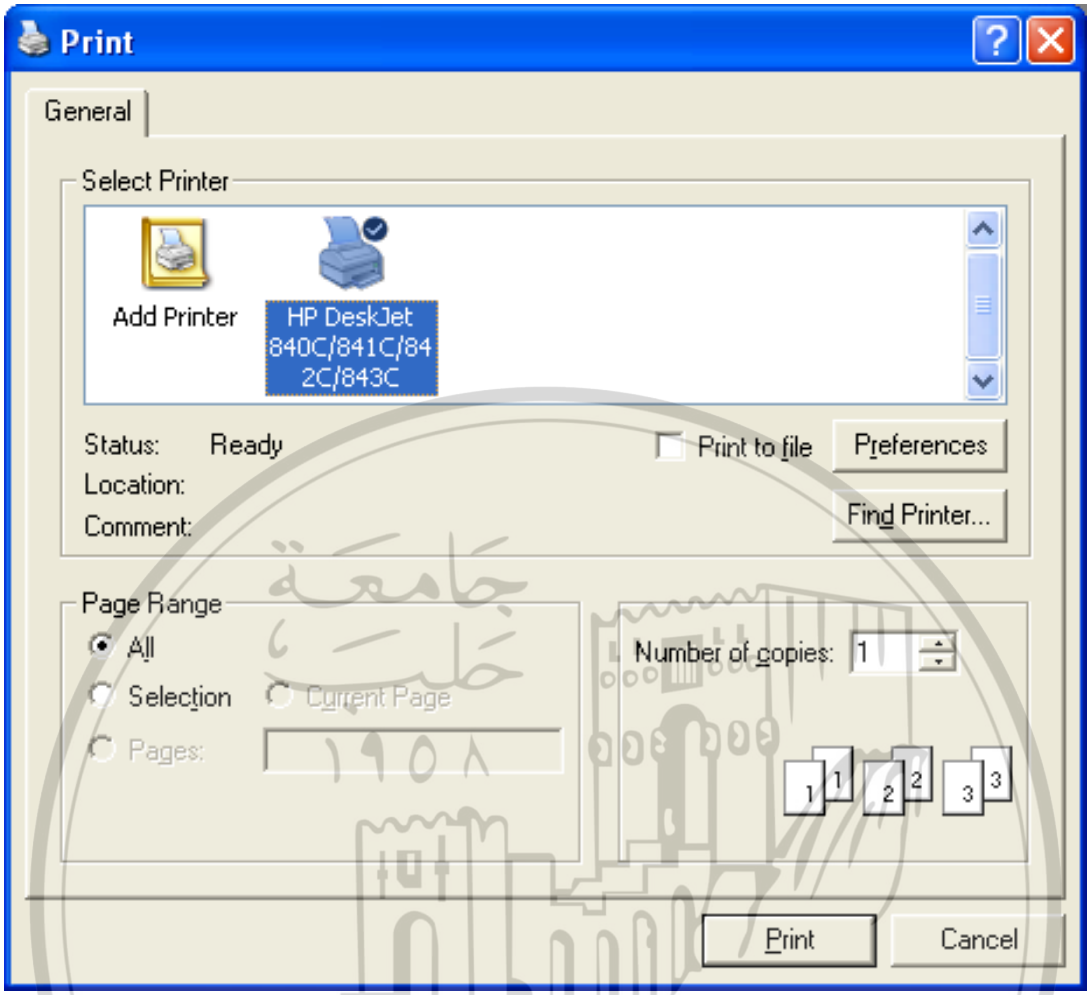
مربع حوار الخطوط The Font Dialog Box:

```
Private Sub mnuFormatFont_Click()
  ' Set Cancel to True.
  CommonDialog1.CancelError = True
  On Error GoTo FontError
  ' Set the Flags property.
  CommonDialog1.Flags = cdlCFBoth Or cdlCFEffects
  ' Display the Font dialog box.
  CommonDialog1.ShowFont
  ' Set text properties according to user's selections.
  txtEditBox.Font.Name = CommonDialog1.FontName
  txtEditBox.Font.Size = CommonDialog1.FontSize
  txtEditBox.Font.Bold = CommonDialog1.FontBold
  txtEditBox.Font.Italic = CommonDialog1.FontItalic
  txtEditBox.Font.Underline
            = CommonDialog1.FontUnderline
  txtEditBox.FontStrikethru
            = CommonDialog1.FontStrikethru
  txtEditBox.ForeColor = CommonDialog1.Color
  Exit Sub
FontError:
  ' User pressed Cancel button.
  Exit Sub
End Sub
```



مربع حوار الطابعة The Printer Dialog Box:

```
Private Sub mnuprinter_Click ()
  Dim BeginPage, EndPage, NumCopies, Orientation, i
  ' Set Cancel to True.
  CommonDialog1.CancelError = True
  On Error GoTo PrintError
  ' Display the Print dialog box.
  CommonDialog1.ShowPrinter
  ' Get user – selected values from the dialog box.
  BeginPage = CommonDialog1.FromPage
  EndPage = CommonDialog1.ToPage
  NumCopies = CommonDialog1.Copies
  Orientation = CommonDialog1.Orientation
  For i = 1 to NumCopies
  ' Put code here to send data to your printer.
  Next
  Exit Sub
PrintError:
  ' User pressed
  ' Cancel button.
  Exit Sub
End Sub
```



فيما سبق لاحظنا أنه نستطيع إظهار أي مربع حوار من خلال ذكر اسمه كما يلي:

CommonDialog1.ShowPrinter *لإظهار الطابعة*

CommonDialog1.ShowColor *لإظهار الألوان*

CommonDialog1.ShowFont *لإظهار الخطوط*

CommonDialog1.ShowOpen *لإظهار مربع فتح*

لكن في الواقع يمكن أن نستعيض عن اسمه بخاصية من خصائصه اسمها Action والتي كما ذكرنا سابقاً تأخذ أحد القيم التالية 1, 2, 3, 4, 5 ولذا يمكن أن نكتب:

CommonDialog1. Action = 5 *لإظهار الطابعة*

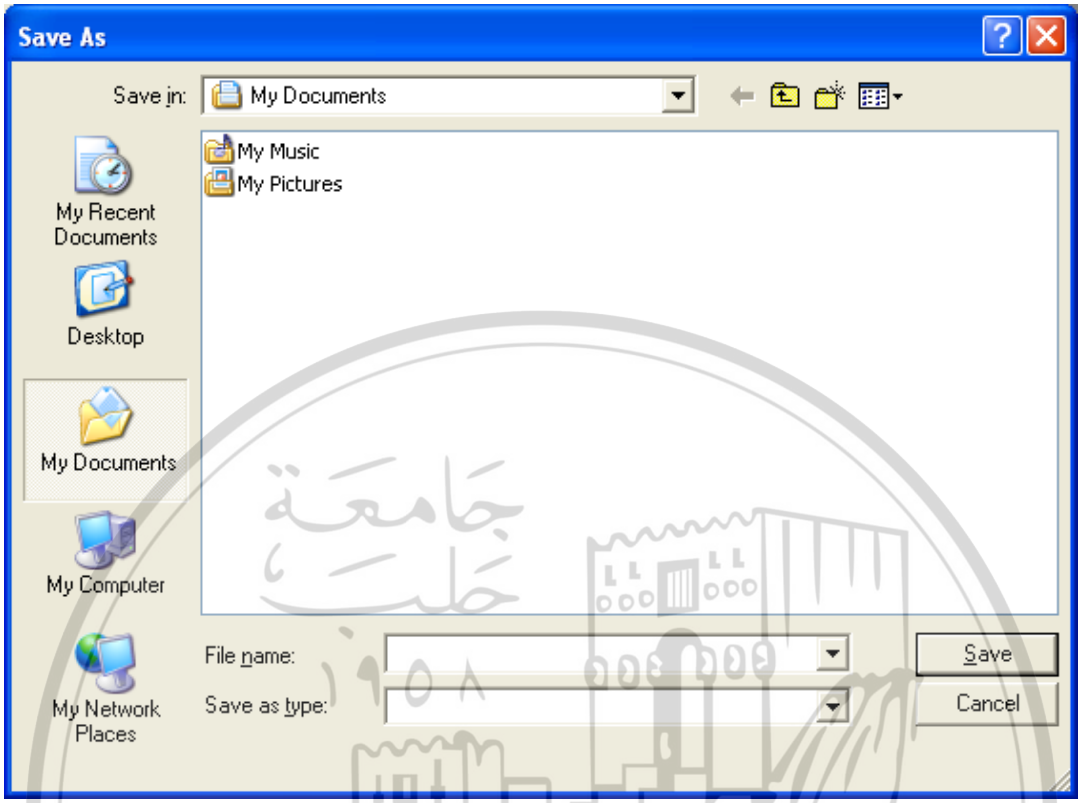
CommonDialog1. Action = 3 *لإظهار الألوان*

CommonDialog1. Action = 4 *لإظهار الخطوط*

CommonDialog1. Action = 1 *لإظهار مربع فتح*

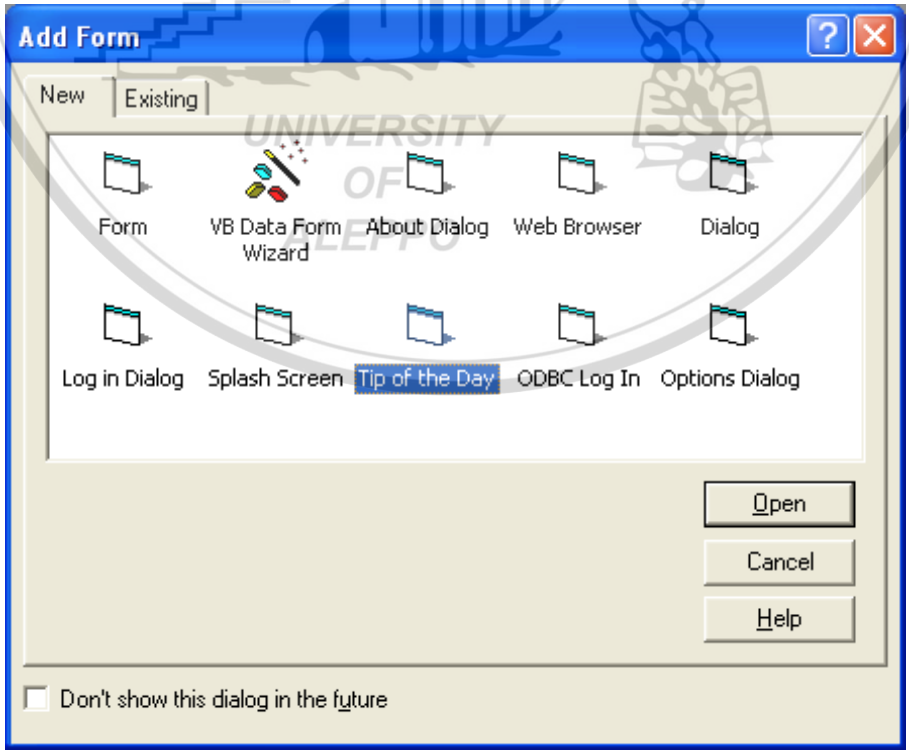
CommonDialog1. Action = 2 *لإظهار مربع حفظ بإسم*

حفظ ملف Save File:



نماذج و مربعات حوار أخرى:

هناك طريقة أخرى هي استخدام النماذج مسبقة التحضير Prepared Forms.

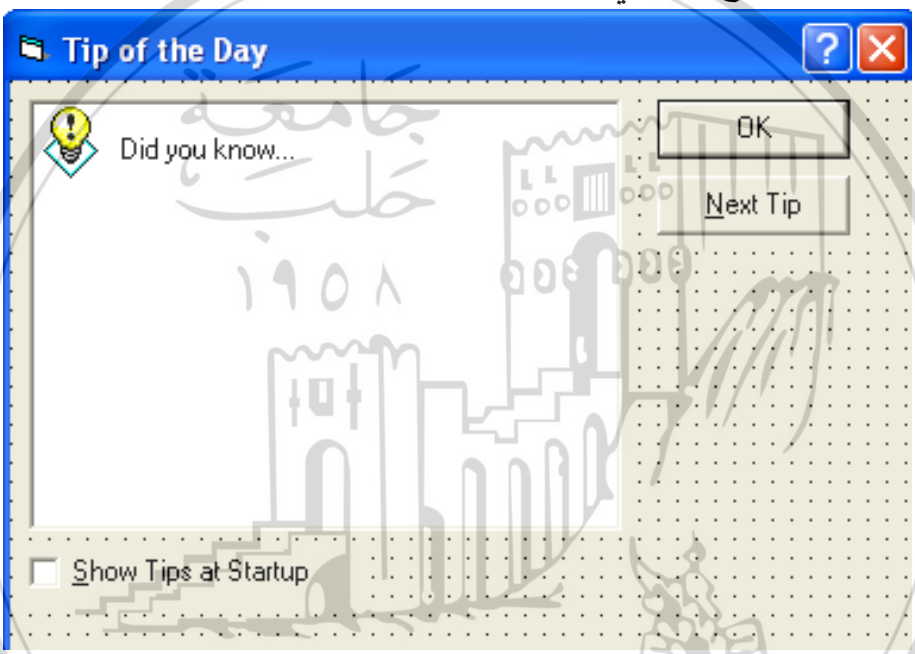


يُعتبر استخدام النماذج مُسبَقَة التحضير مفيداً أحياناً نتيجة الإستفادة من الإمكانيات والأوامر والإجراءات الموجودة بداخله. طبعاً سنكون أقدر على الإستفادة من النماذج الجاهزة عندما نُلِم بشكل صحيح بلغة VB لأننا سنكون أقدر على فهم النصوص البرمجية المكتوبة لهذه النماذج.

مثال على هذه النماذج النموذج Tip of the Day والتي يتم الحصول عليه من خلال:

$$Project \rightarrow Add\ Form \rightarrow New \rightarrow Tip\ of\ the\ Day$$

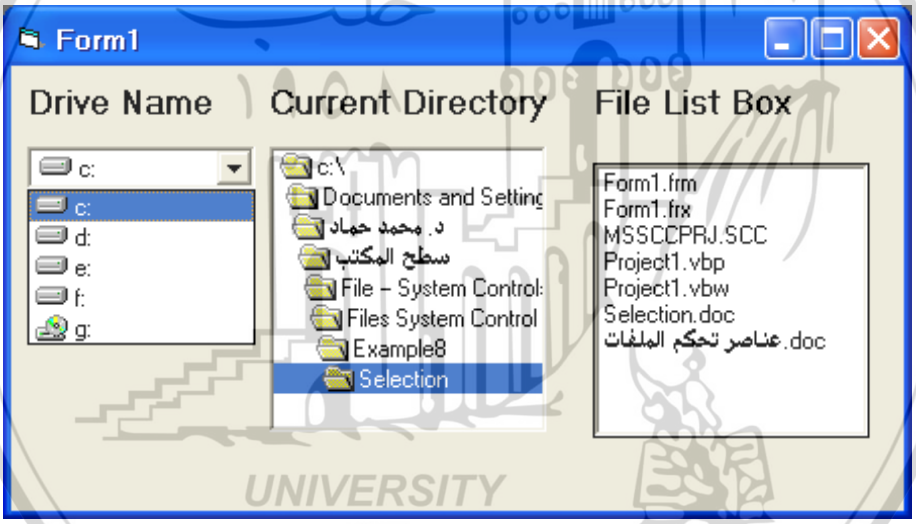
يظهر شكل هذا النموذج كما يلي:



الفصل السادس عشر
عناصر تحكم الملفات
File – System Controls

يمكن باستخدام أوامر التحكم بنظام الملفات اختيار ملف ما من مكان وجوده على أي قرص من الأقراص وعناصر التحكم هي ثلاثة هي:

- أمر اختيار السواقة *Directory List Box*.
- أمر اختيار الدليل *Drive List Box*.
- أمر اختيار الملف *File List Box*.

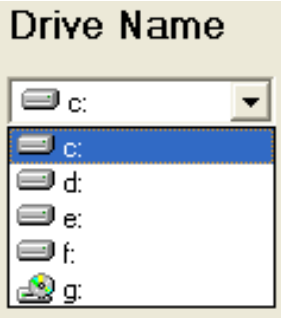


تستخدم هذه الأوامر بطريقة تسمح باختيار الملفات من محركات الأقراص المختلفة. إذ نتمكن من خلال مربعات حوار معينة اختيار الملف المطلوب من خلال عناصر التحكم الثلاثة هذه.

يتم تخصيص مربع يظهر كل السواقات الموجودة ومربع يظهر كل الأدلة الموجودة في السواقة التي تم اختيارها ومربع آخر يظهر كل الملفات الموجودة في الدليل الذي تم اختياره.

مربع اختيار السواقة The Drive List Box:

إن مربع اختيار السواقة عبارة عن مربع منسدل للأسفل وتظهر السواقة التي عليها النظام بشكل افتراضي.



عند وجود التركيز على الأداة $Drive\ List\ Box$ فإننا نستطيع كتابة اسم السواقة لينتقل إليها أو يمكن التنقل بين السواقات عن طريق النقر على القائمة المنسدلة اليمينية فتظهر قائمة للأسفل وتظهر كل السواقات الموجودة. عند اختيار أي من السواقات فإنها ستظهر في أعلى صندوق الاختيار.

الإجراء الكودي لانتقاء السواقة:

```
Private Sub drvName_chang ( )
DirName.Path = drvName.Drive
End Sub
```

حيث:

- $drvName$ اسم الأداة التي يتم عليها تطبيق الخاصية $Drive$ أي انتقاء السواقة فيها.
- Drive الخاصية المطلوب تطبيقها على الأداة $drvName$.
- $DirName$ اسم أداة اختيار الدليل التي ستظهر عليها الأدلة الموجودة على السواقة المنتقاة.

عند اختيار السواقة C مثلا فمعنى ذلك أن الخاصية $Drive$ للأداة $drvName$ ستأخذ القيمة $C:\backslash$ كما يلي:

$$drvName.Drive = "C:\backslash"$$

وبالمقابل فإن مربع اختيار السواقة سيظهر وبشكل دائم اسم السواقة الفعالة الحالية. ويمكن استعمال الخاصية $Drive$ أي السواقة الحالة على النظام وذلك باستعمالها كوسيط في الأمر $ChDrive$ كما يلي:

$$ChDrive\ drvname.Drive$$

مربع اختيار الدليل *The Directory List Box*:

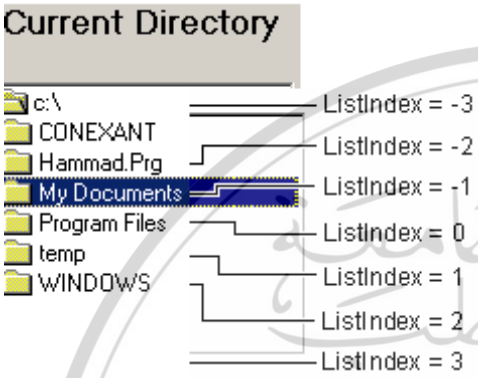
يعرض مربع الأدلة تركيبة وبناء الدليل الحالي على النظام، مبتدئاً من المستوى العلوي للدليل. مبدئياً فإن اسم الدليل الحالي سيكون معلماً من بين الأدلة وسيظهر في شكل متسلسل وهرمي عن الأدلة التي فوقه من الخلف إلى الأعلى. بينما ستتفرع الأدلة الفرعية

في داخله للأسفل وكلما تنقلنا باتجاه الأعلى أو الأسفل بين الأدلة فإن عملية التعليم ستنعكس (أي الدليل المعلم يزول تعليمه وغير المعلّم سيتم تعليمه.

ترتيب عناصر الدليل *Identifying Individual Directories*:

يتم إعطاء قيم صحيحة لكل الأدلة لكي يتم تمييزها عن بعضها.

- بشكل افتراضي سيأخذ الدليل المنتقى الرمز أو القيمة -1 وتفهم كما يلي:



$$Dir1.Path.ListIndex = -1$$

- الدليل الذي سيأتي فوقها مباشرة سيأخذ القيمة -2 والذي فوقه -3 وهكذا.
- الدليل الفرعي الأول من هذا الدليل سيأخذ القيمة 0 أي:

$$DirName.Path\ listIndex = 0$$

وإذا كان هناك أدلة فرعية تحت الدليل الفرعي الأول ستأخذ القيم $1, 2, \dots$ وهكذا.

اختيار الدليل الحالي *Setting the Current Directory*:

يمكن استخدام الخاصية $Path$ في أي أداة Dir لتغيير اسم الدليل الحالي كما يلي:

$$Dir1.Path = "C:\windows"$$

سيقوم هذا الأمر بجعل الدليل الحالي على الأداة $Dir1$ هو الدليل $Windows$ الموجود على السواقة C بغض النظر عن السواقة المختارة في مربع اختيار السواقة.

ويمكن أيضاً تغيير الدليل الحالي من الأمر:

$$ChDir\ dirName.Path$$

في التطبيقات التي تحتوي على أدوات ملفات نجد أن الأمرين:

$$ChDrive\ App.path\ , ChDir\ App.Path$$

يستعملان مع الملفات التنفيذية ونلاحظ أن الخاصية $Path$ متواجدة فقط في المرحلة التنفيذية.

النقر على الأدلة *Clicking a Directory Item*:

عند النقر مرة واحدة على دليل ما فإنه سيتم تعليمه فقط بينما عند النقر عليه مرتين سيتم إعطاء الخاصية $Path$ في مربع الدليل وستأخذ الخاصية $ListIndex$ فيه القيمة -1 وسيتم رسم مربع اختيار الدليل من جديد بحيث تأخذ العناصر الجديدة قيمها الصحيحة.

الإجراء الكودي في مربع اختيار الدليل Codes In the Directory List Box:

```
Private Sub dirName_chang()
  FileName.Path = dirName.Path
End Sub
```

حيث:

- $dirName$ اسم الأداة التي يتم عليها تطبيق الخاصية Dir أي اختيار الدليل فيها.
- $filename.Path$ مسار الملفات المطلوب إظهارها في الأداة $dirFile$.

إيجاد الأدلة بالنسبة للدليل الحالي Finding a Directory's Relative Position:

بما أن ترقيم وعد العناصر يتم بالنسبة للدليل الحالي ولا يمثل العدد الكلي لذا يمكن تحديد الدليل المطلوب التعامل معه بمقدار بعده عن الدليل الحالي والذي تأخذ الخاصية $ListIndex$ دوماً فيه القيمة -1.

مربع اختيار الملفات *The File List Box*:

يظهر مربع اختيار الملفات كل الملفات التي خاصية path في الوقت الحالي من مربع اختيار الأدلة. يتم ذلك باستخدام الأمر الكودي:

$$filenames.Path = DirName.Path$$

حيث تمثل $FileNames$ أسماء الملفات التي سيتم إظهارها في مربع اختيار الملفات انطلاقاً من الأمر المنتقى في الأداة $DirName$.

يمكن اختيار نوع الملفات من الخاصية Pattern أي الملفات ذات النموذج أو النوع المحدد.

مثال:

$$File1.pattern = "*.frm; *.bas"$$

سيقوم VB بإظهار الملفات من النوع (التي لها الامتداد أو اللاحقة) frm و bas فقط.

العمل مع خاصية الفرز في الملفات Working with File Attributes:

في الحالة العادية يتم عرض الملفات من الأنواع (Archive, Normal, System, Hidden, Read-Only) إلا إذا أردنا غير ذلك.

في الحالة الافتراضية تكون الخاصية $Attribute$:

- $True$ للملفات من النوع $Normal, Archive, And\ Read-Only$.
- $False$ للملفات من النوع $System\ and\ Hidden$.

لإظهار الملفات من النوع $Read-Only$ فقط نكتب ما يلي:

$$File1.ReadOnly = True$$
$$File1.Archive = False$$
$$File1.Normal = False$$
$$File1.System = False$$
$$File1.Hidden = False$$

افتراضيا يتم تعليم ملف واحد وإذا أردنا تعليم عدة ملفات نجعل الخاصية $MultiSelect = True$.

استخدام عناصر تحكم الملفات مع بعض – Using File System Controls Together:

عند استخدام هذه الأدوات مع بعضها يجب أن نحرص على حدوث الأوامر المتعلقة ببعضها في زمن واحد كما يلي.

١. نأخذ تسميات متماثلة للأدوات كأن نسميها $Drive1, Dir1, File1$.

٢. نختار سواقة من الأداة $Drive1$.

٣. سيقع الحدث $Drive1\_change$ وسيقوم بتنفيذ الأوامر المرتبطة به ويعيد عرض عناصر السواقة $Drive1$.

٤. سيقوم الإجراء الكودي للحدث $Drive1\_change$ بتحديد خاصية جديدة $Drive$ للأداة $Drive1$ وهذا سيؤدي إلى تغيير الخاصية $Path$ للأداة $Dir1$ كما يلي:

```
Private Sub Drive1_chang ( )
    Dir.Path = Drive1.Drive
End Sub
```

٥. سـيقع الحدث $Dir1\_change$ وسـيقوم بتنفيذ الأوامر المرتبطة به ويعيد عرض عناصر الدليل $Dir1$.

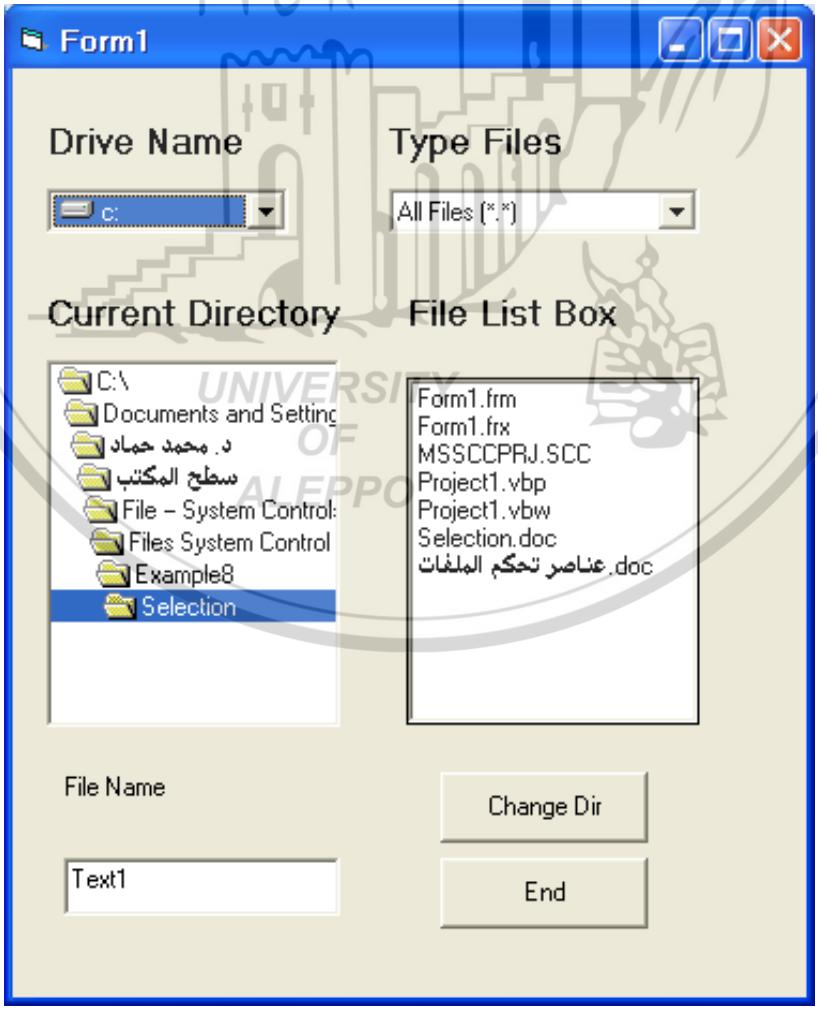
٦. سـيقوم الإجراء الكودي للحدث $Dir1\_change$ بتحديد خاصـية جديدة $Path$ للأداة $Dir1$ وهذا سيؤدي إلى تغيير الخاصية $Path$ للأداة $File1$ كما يلي:

```
Private Sub Dir1_chang ( )
  File1.Path = Dir1.Path
End Sub
```

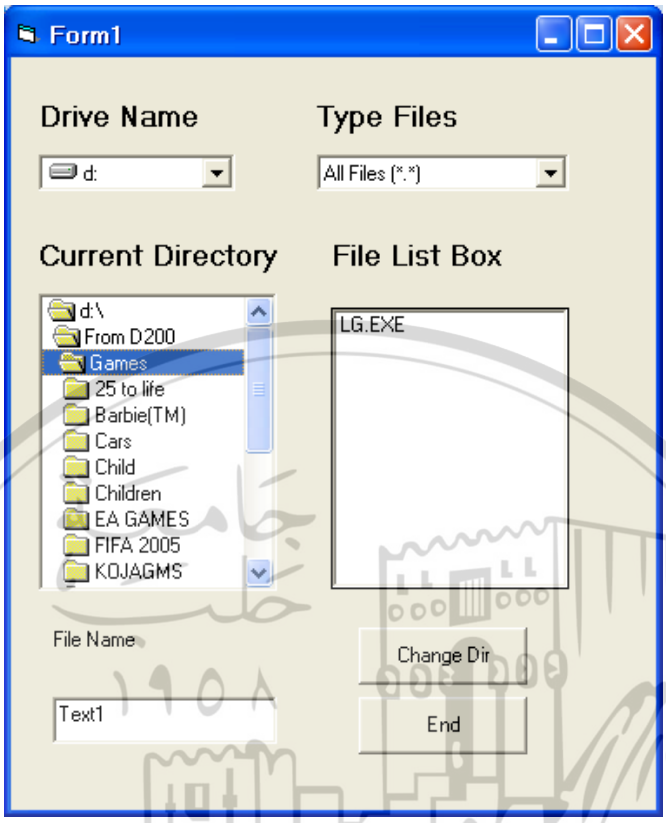
٧. إن تغيير الخاصية $Path$ للأداة $File1$ سيؤدي إلى عرض لائحة بالملفات المتعلقة بالخاصـية الجديدة $Path$ للأداة $Dir1$ في مربع اختيار الملفات $File1\ List\ Box$.

صمم الواجهة المرئية كما يلي:

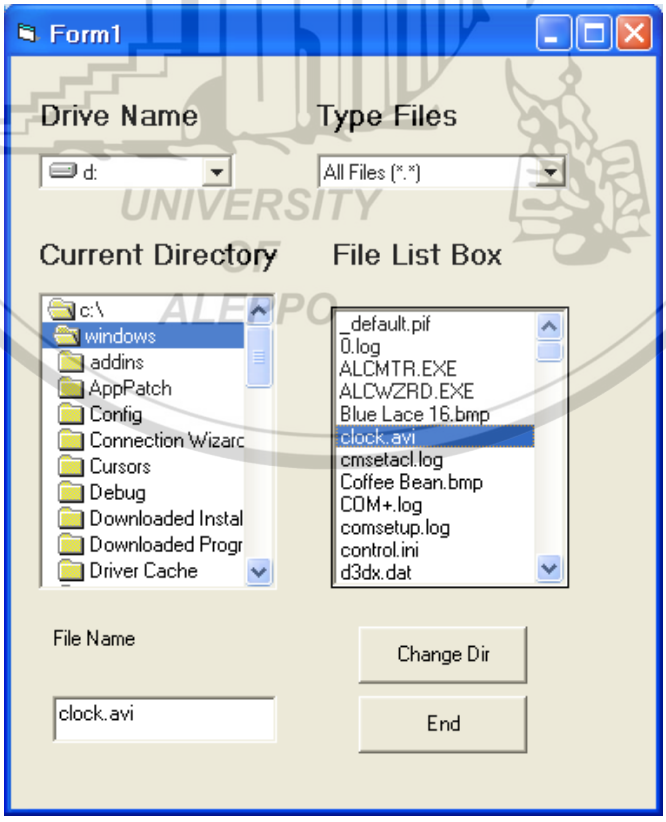
الأداة Combo لتحديد نوع الملفات.



زر الأوامر *Change Dir* للانتقال إلى المجلد *Windows* من السواقة *C*.



عند تحديد أو انتقاء أي ملف يظهر اسمه في مربع النص:



```
Private Sub Form_Load ( )
Combo1.AddItem "All Files (*.*)"
Combo1.AddItem "Text Files (*.Txt)"
Combo1.AddItem "Doc Files (*.Doc)"
Combo1.ListIndex = 0
End Sub
```

```
Private Sub Drive1_Change ( )
Dir1.Path = Drive1.Drive
End Sub
```

```
Private Sub Combo1_Click()
Select Case Combo1.ListIndex
Case 0
 File1.Pattern = "*.*"
Case 1
 File1.Pattern = "*.Txt"
Case 2
 File1.Pattern = "*.Doc"
End Select
End Sub
```

```
Private Sub Dir1_Change ( )
File1.Path = Dir1.Path
End Sub
```

```
Private Sub File1_Click ( )
Text1.Text = File1.FileName
End Sub
```

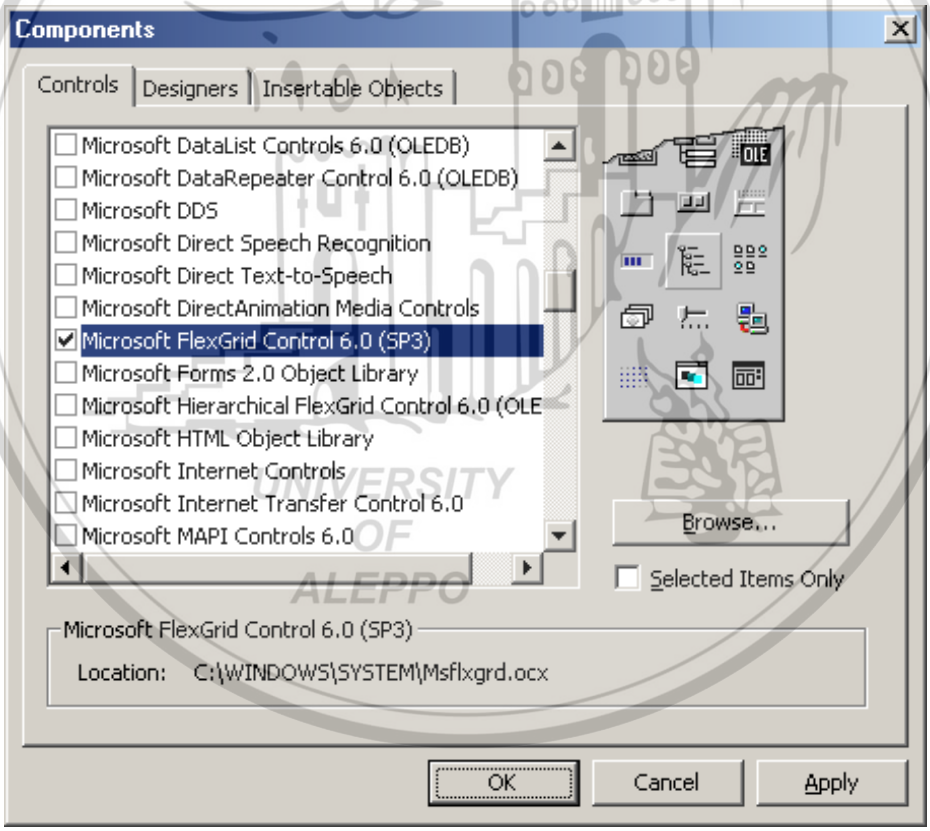
```
Private Sub Command1_Click ( )
Dir1.Path = "C:\windows"
End Sub
```

```
Private Sub Command2_Click()
End
End Sub
```

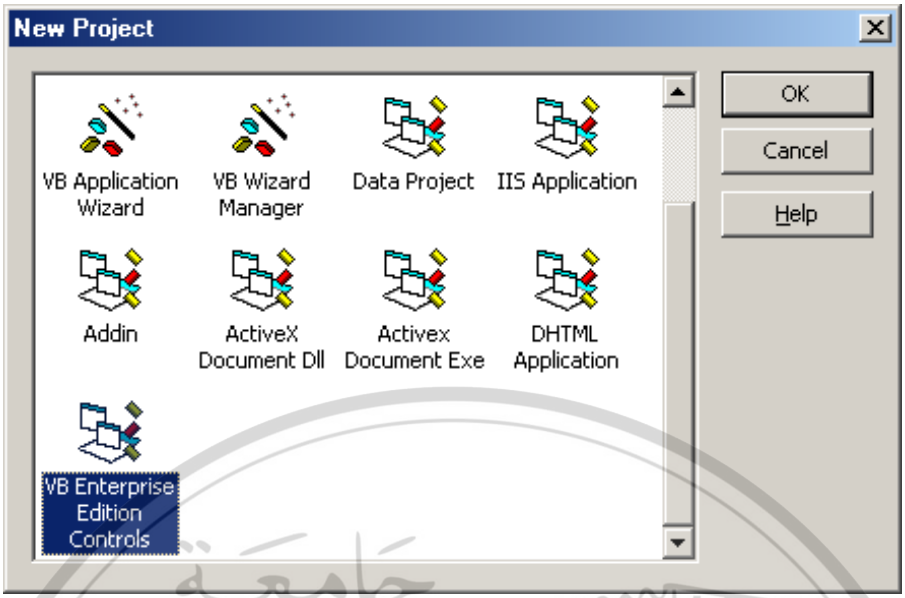
الفصل السابع عشر
أداة الجدول المرن
Ms – Flex Grid Control

إضافة الأداة إلى شريط الأدوات:

يمكن إضافتها عن طريق القائمة Project - Components. يظهر مربع الحوار Components نختار منه الأداة Microsoft flex Grid Control 6.0 (SP3) من مربع الأدوات ثم نضغط على زر Ok. ستظهر الأداة داخل مربع الأدوات بعد إغلاق مربع Components كما هو مبين في الشكل:

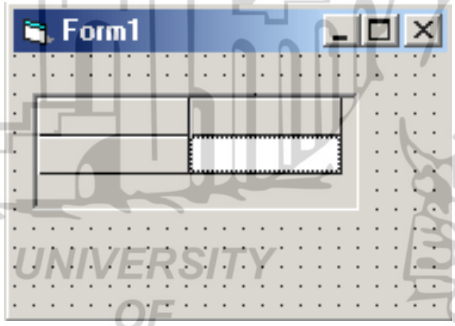


ويمكن إضافتها عن طريق إضافتها عن طريق القائمة File - New Project. يظهر مربع الحوار New Project نأخذ منه VB Enterprise Edition controls عندها سيتم إضافة مجموعة من الأدوات ToolBox الإضافية إلى شريط الأدوات ومن بينها الأداة MsFlex Grid.



ماهية الصفوف والأعمدة:

عند النقر على هذه الأداة يظهر في الإطار Form مستطيل فيه جدول أبعاده 2 x 2 أي فيه صفان وعمودان مع مساحة خالية تسمى Background ويرمز لها BKG تستخدم للتحرير الشبكي (مصفوفات – جداول).



إن شبكة ورقة العمل الناتجة عن الأداة MsFlexGrid تحتوي جدول ذي صفوف أفقية وأعمدة عمودية. وبشكل افتراضي يكون الصف العلوي والعمود اليساري محجوزان لعناوين الصفوف والأعمدة ويتم عرضها بخلفية رمادية. نستعمل الخاصية Rows لضبط عدد الصفوف في الجدول والخاصية Cols لضبط عدد الأعمدة. ويمكن زيادة عدد الصفوف والأعمدة بالشكل الذي نريد.

يتم التعامل مع البيانات الجدولية في الأداة MsFlexGrid كمصفوفة ثنائية الأبعاد، البعد الأول في الجدول هو أرقام الصفوف والبعد الثاني هو أرقام الأعمدة (عمود، صف).

مثلاً الخلية المحددة هي (0 ,0) التي تدل على أنها موجودة في الصف رقم صفر والعمود رقم صفر.

وتستخدم الخاصية AllowUserResizing للسماح أو عدم السماح للمستخدم بتغيير عرض الأسطر والأعمدة أو أحدها بالسحب وذلك عند الوقوف على الخط الفاصل بين هذه الأسطر أو بين هذه الأعمدة. ونستخدم ذلك عندما نرى أن بعض الكلمات في سطر ما أو عمود ما غير ظاهرة، ولإظهارها نوقف الماوس على الحدود الفاصلة بين الأعمدة أو بين الأسطر ثم نسحب الأعمدة أو الأسطر بالماوس لتكبيرها. وتأخذ الخاصية القيم التالية:

القيمة	المعنى
0 – flexResizNone	لا يسمح بتغيير أبعاد الأسطر أو الأعمدة
1 – flexResizColumns	يسمح بتغيير أبعاد الأعمدة فقط
2 – flexResizRows	يسمح بتغيير أبعاد الأسطر فقط
3 – flexResizBoth	يسمح بتغيير أبعاد الأسطر و الأعمدة

يمكن في المرحلة التنفيذية أن نقوم بأحد الأوامر التي نستخدمها في التعامل مع الجدول:

- الأمر الأول: تحديد عدد صفوف وأعمدة الجدول.
- الأمر الثاني: إدراج البيانات إلى الجدول.
- الأمر الثالث: إضافة صفوف جديدة إلى الجدول.

تحديد عدد صفوف وأعمدة الجدول:

يتم تحديد عدد الصفوف من خلال الخاصية Rows وعدد الأعمدة من خلال الخاصية Cols ويمكن أن يتم ذلك في المرحلة التصميمية وافتراضيا يكون عدد الأسطر 2 وعدد الأعمدة 2. ويمكن أن يتم ذلك في المرحلة التنفيذية كما يلي:

```
Private Sub Form_Load()
Grid1.Rows = 3
Grid1.Cols = 4
End Sub
```

جعل عدد الأعمدة مساوياً لـ 4 و جعل عدد الأسطر (الصفوف) مساوياً لـ 3.

إدراج البيانات في الجدول:

هناك طرق لإدراج البيانات في الجدول هي:

الطريقة الأولى:

تحديد خلية cell معينة بالانتقال إليها بتحديد الخاصيتين Row ، Col ثم يلي ذلك كتابة النص المراد فيها بالخاصية Text.

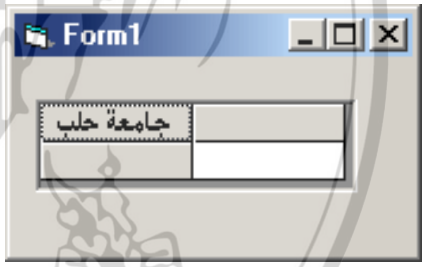
Grid1.Row = 0 : Grid1.Col = 0 : Grid1.Text = "جامعة حلب"

هنا سيتم إضافة عبارة "جامعة حلب" في الخلية الواقعة على الصف 0 (صفر) من العمود 0 (صفر).

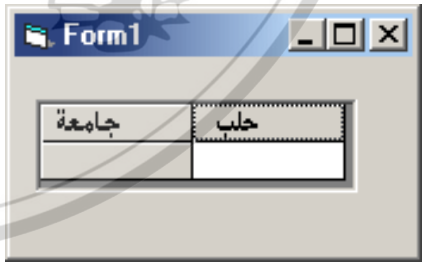
Grid1.Row = 0 : Grid1.Col = 0 : Grid1.Text = " جامعة "
Grid1.Col = 1 : Grid1.Text = " حلب "

أي نضيف كلمة "جامعة" في الخلية الواقعة في الصف 0 والعمود 0 وبما أننا لم نذكر في السطر الثاني قيمة الصف فمعنى ذلك أنه نفس الصف أي الصف رقم 0 وبالتالي فإن كلمة "حلب" ستضاف إلى الخلية الواقعة في الصف 0 والعمود1.

```
Private Sub Form_Load( )
Grid1. Row = 0:Grid1.Col = 0
Grid1.Text =  "جامعة حلب"
End Sub
```




```
Private Sub Form_Load()
Grid1.Row = 0: Grid1.Col = 0
Grid1.Text = " جامعة"
Grid1.Col = 1: Grid1.Text =  " حلب"
End Sub
```



الطريقة الثانية:

تحديد نطاق من الخلايا وذلك بالانتقال إلى الخلية المحددة لأحد أركان النطاق بـ Row و Col ومن ثم تحديد الركن المقابل بالخاصتين Rowsel و Colsel ثم نقوم بملئه بعبارة واحدة هي الوظيفة Clip للجدول حيث تملأ النطاق باستخدام نص أو تابع نصي String. ولتوزيع محتويات النص على الخلايا المختلفة والصفوف المختلفة يتم ذلك بإدراج أحرف

للتحكم ضمن النص، الحرف VbTab والذي يفصل بين الأعمدة المختلفة بينما VbCr يقوم بالفصل بين صف وآخر (أي لفتح سطر جديد).

```
Private Sub Form_Load ( )
Grid1.Row = 0: Grid1.Col = 0
Grid1.RowSel = 0: Grid1.ColSel = 3
Grid1.Clip = "الدكتور" & vbTab & "محمد" & vbTab & "حماد"
End Sub
```

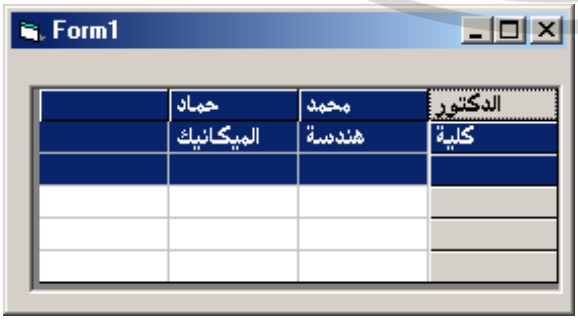
بدأ التحديد من الخلية الواقعة في الصف 0 والعمود 0 وأنهى التحديد عند الخلية الواقعة في الصف 0 والعمود 3 أي حدد سطر مؤلف من أربع خلايا.

بعد ذلك تم إضافة الكلمات "الدكتور" ثم "محمد" ثم "حماد" إلى الخلايا المحددة وبما أنه هناك ثلاث كلمات لذلك ظهرت الخلية الرابعة المحددة فارغة.

إذا كان طول خلية الشبكة FlexGrid غير كافي لعرض كل الخلايا في المرحلة التنفيذية فإن شريط تمرير سيظهر بالاتجاه المطلوب وهذا واضح هنا في الوضع الأفقي والعمودي.

هنا أخذنا الخاصية RightToLeft مساوية للقيمة False لذا ظهرت الخلايا من اليسار إلى اليمين.

```
Private Sub Form_Load()
Grid1.Row = 0: Grid1.Col = 0
Grid1.RowSel = 2: Grid1.ColSel = 3
Grid1.Clip = "الدكتور" & vbTab & "محمد" & vbTab & "حماد" & Chr(13)_
 &"كلية" & vbTab & "هندسة" & vbTab & "الميكانيك"
End Sub
```



بدأ التحديد من الخلية الواقعة على الصف 0 والعمود 0 وأنهى التحديد عند الخلية الواقعة في الصف 2 والعمود 2 أي قام بتحديد سطرين كل منهما مؤلف من أربع خلايا.

بعد ذلك تم إضافة الكلمات "الدكتور" ثم "محمد" ثم "حماد" إلى الخلايا المحددة في السطر الأول وانتقل بالتابع $\boldsymbol{Chr(13)}$ إلى الصف الثاني الذي كتب فيه الكلمات "كلية" ثم "هندسة" ثم "الميكانيك" في السطر الثاني.

وبما أنه هناك ثلاث كلمات في كل سطر لذلك ظهرت الخلية الرابعة المحددة في كل سطر فارغة.

هنا طول أداة الشبكة FlexGrid كافٍ لعرض كل الخلايا في المرحلة التنفيذية لذا لم يظهر أي شريط تمرير.

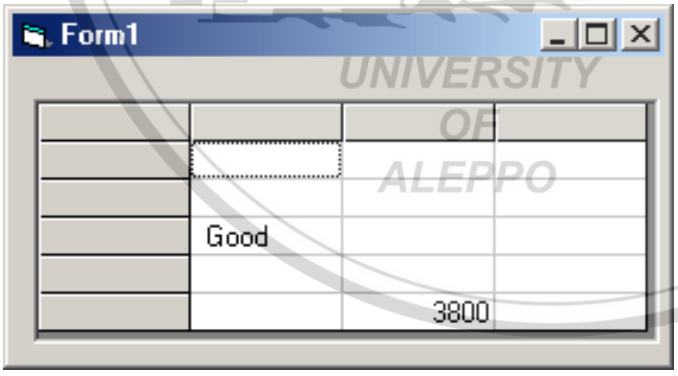
هنا أخذنا الخاصية RightToLeft مساوية للقيمة True لذا ظهرت الخلايا من اليمين لليسار.

الطريقة الثالثة:

استخدام TextMatrix وهي دالة تمكنك من الكتابة في خلية معينة مباشرة بعبارة واحدة وذلك من خلال تحديد الصف والعمود للدالة .

لوضع قيمة في خلية MsFlexGrid سواءً كانت نصية أو رقمية باستخدام الخاصية TextMatrix ونحدد الخلية المطلوب الكتابة فيها:

$$Grid1.TextMatrix\ (3,1) = "Good"$$
$$Grid1.TextMatrix\ (5,2) = "3800"$$



كتابة النص Good في الخلية الواقعة في الصف الثالث والعمود الأول.

والرقم 3800 في الخلية الواقعة في الصف الخامس والعمود الثاني.

ملاحظة:

يجب أن لا ننسى أن أعلى صف هو الصف ذو الرقم (0) والعمود في أقصى اليسار هو العمود (0).

الصف ذو الرقم (0) والعمود ذو رقم (0) هي للعناوين.

إضافة صفوف جديدة إلى الجدول:

يتم إضافة صفوف جديدة إلى الجدول باستخدام الوظيفة AddItem مع تحديد محتويات الصف بنص يشبه تمامًا النص المستخدم مع Clip .

الصف الجديد يوضع اختياريًا في نهاية الجدول. وهناك معامل اختياري لهذه الوظيفة يمكننا من إضافة الصف الجديد في مكان آخر خلاف النهاية.

```
Private Sub Form_Load( )
Grid1.Row = 0: Grid1.Col = 0
Grid1.RowSel = 2: Grid1.ColSel = 3
Grid1.Clip = "الدكتور" & vbTab & "محمد" & vbTab & "حماد"
Grid1.AddItem "كلية" & vbTab & "هندسة" & vbTab & "الميكانيك"
End Sub
```

تصميمياً كان عدد الأسطر 3 وعدد الأعمدة 4.

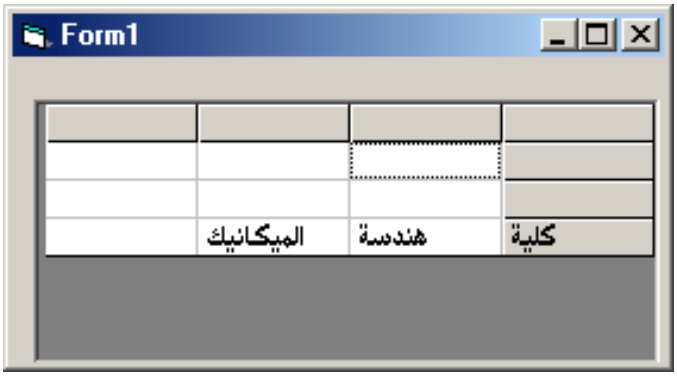
تم تحديد الخلايا من الخلية (0,0) وحتى الخلية (2,3) ثم تم إضافة الكلمات الثلاثة الأولى فيها.

بعد ذلك تم إضافة سطراً مطابقاً لأسطر الأداة ومن ثم أضفنا فيه الكلمات الثلاث الأخرى في خلاياه الأولى.

ويمكن ببساطة كتابة ما يلي:

```
Private Sub Form_Load ( )
Grid1.AddItem "كلية" & vbTab & "هندسة" & vbTab & "الميكانيك"
End Sub
```

وعندها سنحصل على:



التحكم في مظهر الجدول المرن:

خصائص التحكم في ألوان الكتابة وهي:

- ForeColor لتحديد لون النص في الخلية العادية.
- ForeColorFixed لتحديد لون النص في الخلية الثابتة.
- ForeColorSel لتحديد لون النص في الخلية المعلمة.

خصائص للتحكم بتحديد لون الخلفية في مناطق مختلفة من الجدول:

- BackColor للخلية العادية.
- BackColorFixed للخلية الثابتة.
- BackColorSel للخلية المعلمة.
- BackColorBKG للمنطقة الخلفية من الجدول التي لا تحتوي على أية خلية.

خصائص التحكم في لون خلية منفردة أو مجموعة معلمة من الخلايا:

- CellForeColor: لتحديد لون الإطار في الخلية المحددة.
- CellBackColor: لتحديد لون الخلفية للخلية المحددة.

خصائص الخطوط في الجدول:

- خاصية التحكم في خلايا الجدول كلها Font.
- للتحكم في خط أحد الخلايا على انفراد باستخدام CellFontName و CellFontSize ومجموعة شبيهة من الخواص تبدأ بـ CellFont.

تزود الأداة Flex Grid عدة ميّزات تنسيق قياسية تتضمن ميزات مثل الأسود العريض والمائل والتسطير ومحاذاة النصوص في الأعمدة وأسماء الخطوط وأحجامها والألوان الأمامية والخلفية. وأهم هذه الخيارات:

$$MsFlexGrid1.CellFontBold = True$$
$$MsFlexGrid1.CellFontItalic = True$$
$$MsFlexGrid1.CellFontUnderline = True$$
$$MsFlexGrid1.CellAlignment = FlexAlignmentRightCenter$$
$$MsFlexGrid1.CellFontName = "Courier\ New"$$
$$MsFlexGrid1.CellFontSize = 14$$
$$MsFlexGrid1.CellForColor = "Red"$$
$$MsFlexGrid1.CellBackColor = "Blue"$$

وهناك مجموعة من الخصائص تتحكم في خطوط الشبكة التي تفصل خانات الجدول وتبدأ كلها بـ Grid وهي:

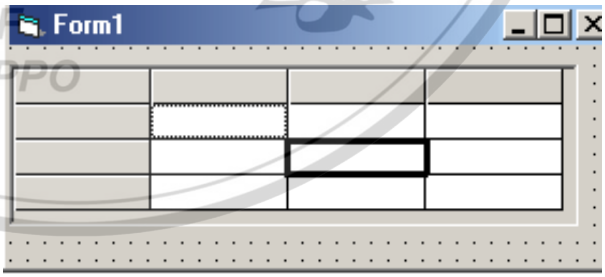
- GridColor لتحديد لون الخطوط بين الخلايا العادية.
- GridColorFixed لتحديد لون الخطوط بين المحيط بالجدول.
- GridLines تتحكم في ظهور الخطوط من عدمه للخلايا المنتقاة.
- GridLinesFixed تتحكم في ظهور الخطوط من عدمه بين الخلايا الثابتة.
- GridLineWidth لتحديد سمك الخطوط المحيطة بالجدول.

انتقاء الخلايا:

حتى نتمكن من تنسيق الخلايا كان لابد من أن ننتقيها في البداية، والانتقاء يمكن أن يكون فرديّاً (خلية واحدة) أو نطاقاً (كتلة متجاورة) من الخلايا بواسطة الشفرة البرمجية.

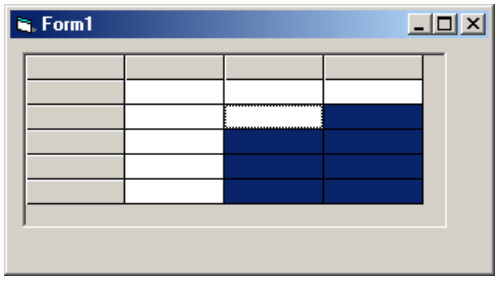
- لانتقاء خلية فردية نقوم بضبطها عن طريق تحديد الصف وكذلك العمود الواقعة فيهما أي باستخدام الخاصيتين Row و Col كما يلي:

$$MsFlexGrid1.Row = 2$$
$$MsFlexGrid1.Col = 2$$



هنا تم تحديد الخلية الواقعة في العمود لثاني والصف الثاني.

- لانتقاء نطاق من الخلايا نحتاج إلى تحديد نقطة بداية ونقطة نهاية للانتقاء. نقطة البداية ننتقيها بالخاصيتين Row و Col ونقطة نهاية الانتقاء يتم تحديدها بواسطة الخاصيتين RowSel و ColSel.



$$MsFlexGrid1.Row = 2$$
$$MsFlexGrid1.Col = 2$$
$$MsFlexGrid1.RowSel = 5$$
$$MsFlexGrid1.ColSel = 3$$

هنا تمّ تحديد الخلايا من (2, 2) وحتى (5, 3) أي من الخلية الواقعة على الصف الثاني والعمود الثاني حتى الخلية الواقعة على الصف الخامس والعمود الثالث.

بعد عملية الانتقاء وإذا كنا نريد تنسيق الخلايا فيجب التنويه بعملية التنسيق إلى أنّ التنسيق سيتم على خلية واحدة أو على النطاق وذلك باستخدام الخاصة FillStyle الواجب تطبيقه. هل هو فرديا (أي خلية واحدة) وهي القيمة الافتراضية وعندها يكون 0-FlexFillSingle أو باستخدام 1-FlexFillRepeat والتي تتيح للأداة FlexGrid تنسيق أكثر من خلية منتقاة في الوقت نفسه. بعد ذلك نصبح جاهزين لبدء التنسيق سواءً كخلية فردية أو كنطاق.

$$MsFlexGrid1.FillStyle = FlexFillRepeat$$
$$MsFlexGrid1.FillStyle = 0$$

يمكن في البداية انتقاء الخلايا ومن ثم تنسيق هذه الخلايا كتغيير نمط النص أو غير ذلك. كما يمكن التحكم بكيفية استعمال الشبكة أثناء التشغيل من خلال ضبط الخاصية SelectionMode بالشكل:

$0 - FlexSelectionFree$ انتقاء عادي

$1 - FlexSelectionByRow$ انتقاء صفوف فقط

$2 - FlexSelectionByColumn$ انتقاء أعمدة فقط

إدراج الرسوم في الخلايا:

يمكن إدراج صورة أو رسم ضمن خلايا MsFlexGrid شريطة أن تكون الصورة أو الرسم عبارة عن:

- Bmp: أي صورة نقطية.
- Ico: أي أيقونة أو رمزية.
- Wmf: ملفاً من ملفات تعريف ويندوز.

تضاف هذه الرسوم باستخدام الجملة Set والخاصية CellPicture والدالة LoadPicture.

مثال:

$$SetMsFlexGrid1.CellPicture = LoadPicture("d:\VB6.0\boy.bmp")$$

بعد إضافة الصورة يجب ملائمة أبعاد الخلية مع أبعاد الرسم وذلك بتعديل ارتفاع وعرض الخلية التي تحوي على هذا الرسم. مثلاً إذا كان حجم الرسم هو 1000 x 1000 twip وهي موجودة في الصف الرابع والعمود الثالث فإننا سنلائم حجم خلايا هذا الصف والعمود لملائمة حجم الصورة (الرسم) المدرجة.

$$MsFlexGrid1.RowHeight(4) = 1000$$
$$MsFlexGrid1.ColWidth(3) = 1000$$

إن وضع هذه الجمل مباشرةً فوق الجملة Set والتي تحمل الرسم في الخاصية CellPicture سيؤدي إلى تكبير الخلية بما يكفي لإظهار الرسم بأكمله.

عند تغيير حجم خلية ما في الشبكة سيتغير حجم كل الصف وكل العمود التابعين لتلك الخلية.

التحكم في سلوك الجدول:

من أبرز الخصائص الخاصة بتغيير سلوك الجدول ما يلي:

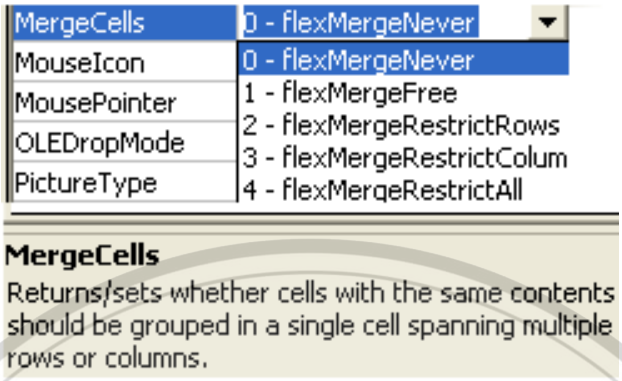
- AllowBigSelection: تحدد إمكانية تحديد عمود بنقر عنوانه الرئيس أم لا ونفس الحال مع الصف.
- AllowUserResizing: تحدد إمكانية تغيير المستخدم لأبعاد الأعمدة والصفوف ديناميكيا أثناء عمل البرنامج أم لا.
- FillStyle: الخصائص التي تبدأ بـ Cell مثل CellFontName وغيرها وتستخدم لتغيير كافة الخلايا المحددة إذا كانت الخاصية FillStyle بالقيمة واحد أما إذا كان قيمتها صفر فإن تأثير الخصائص المشار إليها لا يتعدى الخلية الفعالة وحدها.
- MergeCells: يحدد السماح بدمج الخلايا، هذا الدمج يتم آليّاً إذا كانت قيم الخلايا المتجاورة متشابهة، قد نسمح بالدمج بين الأعمدة فقط أو بين الصفوف فقط أو بين كليهما أو نمنعه تمامًا.
- SelectionMode: يحدد هل يمكن تحديد الخلايا في أي مكان من الجدول أم أن التحديد سيمتد إجباريا ليشمل عمود بالكامل أو صف بالكامل.

الأحداث الخاصة بالجدول المرن: إن أهم الأحداث التي يطلقها الجدول هي:

- GridLines تتحكم في ظهور الخطوط من عدمه للخلايا المنتقاة.
- EnterCell: ينطلق هذا الحدث في كل مرة يتم انتقال التركيز إلى الخلية.
- LeaveCell: عكس الحدث السابق فهو يقع عند فقدان الخلية للتركيز.
- RowColChange: ينطلق عند انتقال التركيز من خلية إلى أخرى.
- SelChange: ينطلق عندما يتم تغيير نطاق التحديد.
- بالإضافة إلى أهم الأحداث الأخرى المطبقة على الأدوات الأخرى مثل: Click, DblClick, Scroll ... الخ.

أهم خصائص المتعلقة في الجدول MsFlexGrid:

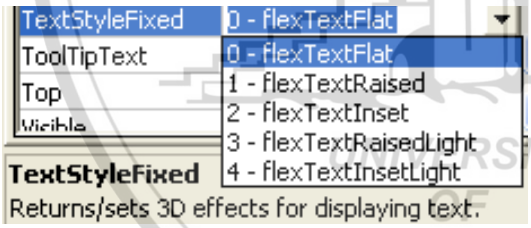
- خاصية دمج الخلايا MergeCells:

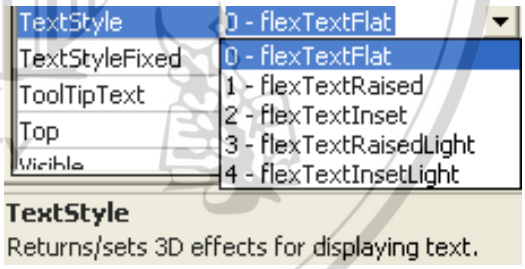


وهي خاصية تمكن المستخدم من القيام بدمج مجموعة من الخلايا بخلية واحدة وقد يكون نطاق الخلايا أفقي أو عمودي أو أفقي وعمودي.

Property Name	***Property Value***
MergeCells	*1 - FlexMergeFree*
	2 - FlexMergeNever
	3 - FlexMergeNever
	4 - FlexMergeNever
	5 – FlexMergeNever

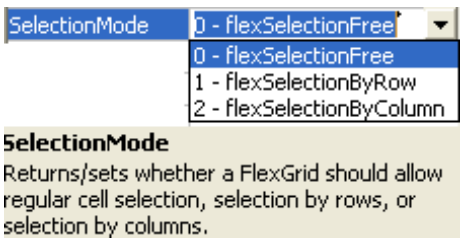
- خاصية تأثيرات ثلاثية الأبعاد على النص TextStyle:

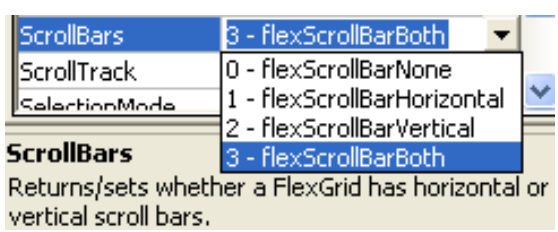




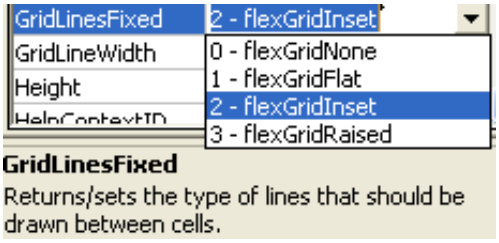
وهي خاصية تمكن المستخدم من القيام بكتابة الخلايا على شكل ثلاثية الأبعاد أو بالشكل العادي.

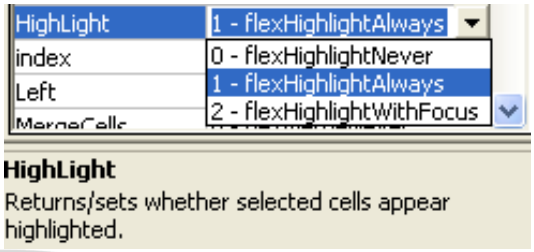
- خاصية السحابات الأفقية و العمودية ScrollBars:
- خاصية تحديد الخلايا SelectionMode:



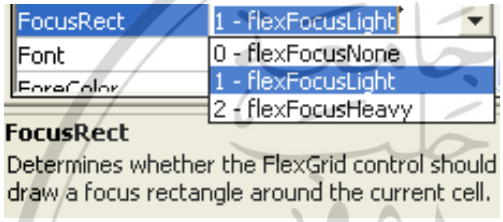


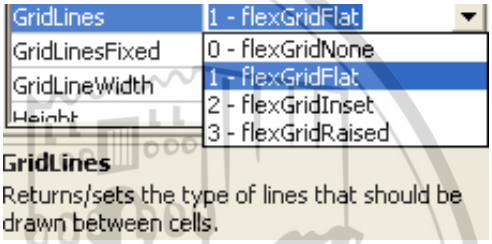
- خاصية نمط الخطوط بين الخلايا GridLinesFixed:
- خاصية تعليم الخلايا HighLight:



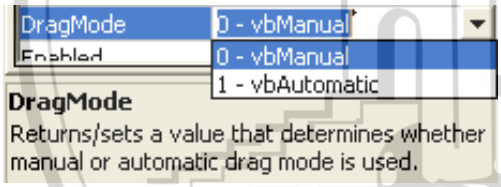


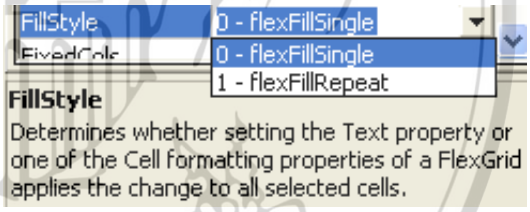
- الخاصية GridLines:
- الخاصية FocusRect:



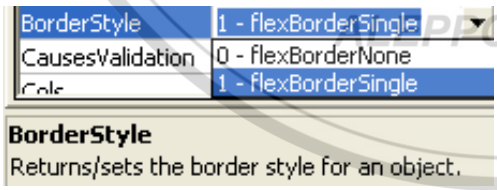


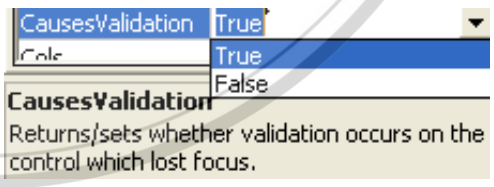
- الخاصية FillStyle:
- الخاصية DragMode:



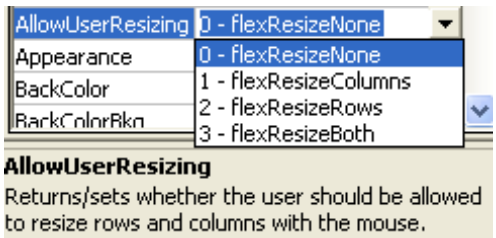


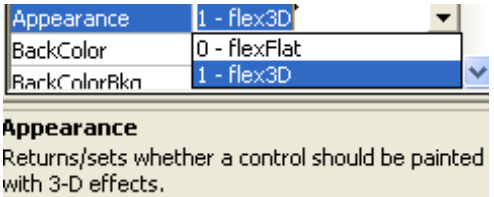
- الخاصية CausesValidation:
- الخاصية BorderStyle:



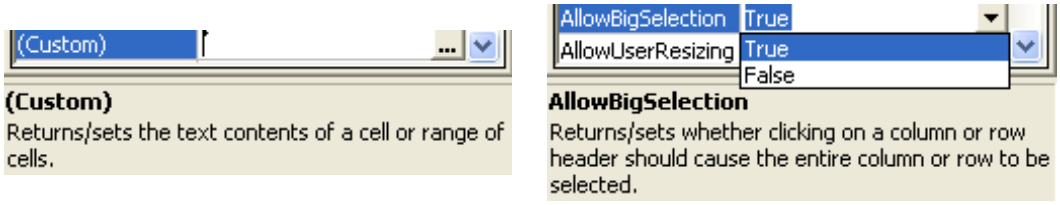


- الخاصية Appearance:
- الخاصية AllowUserResizing:



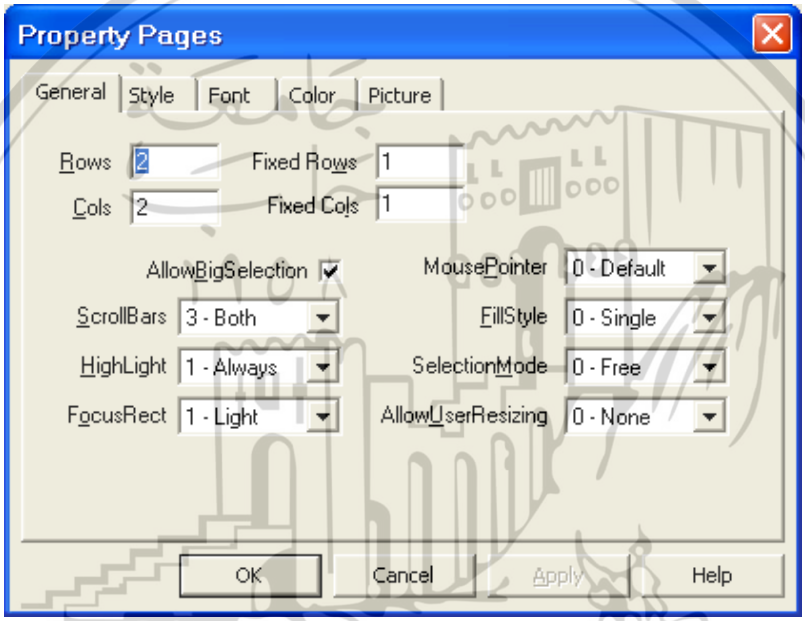


- الخاصية AllowBigSelection:
- الخاصية Custom:

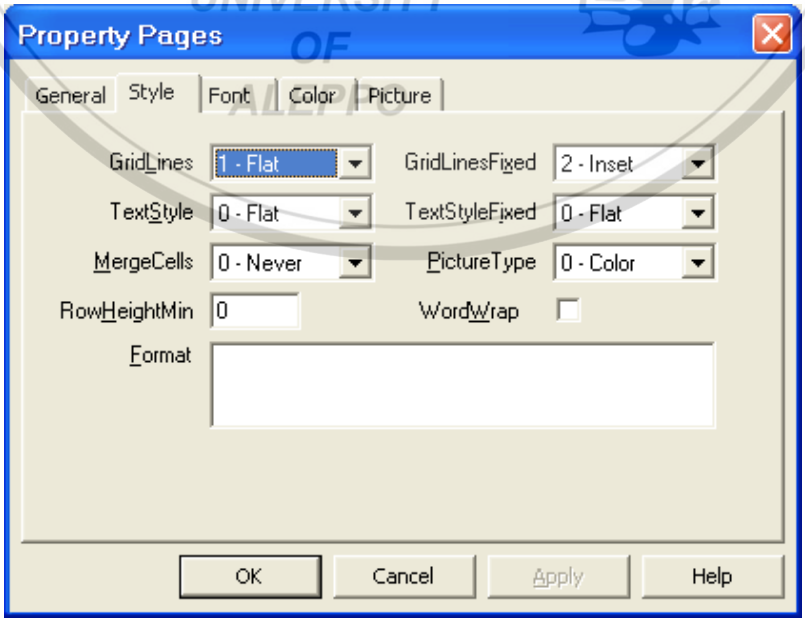


أهم مربعات الحوار في الجدول MsFlexGrid:

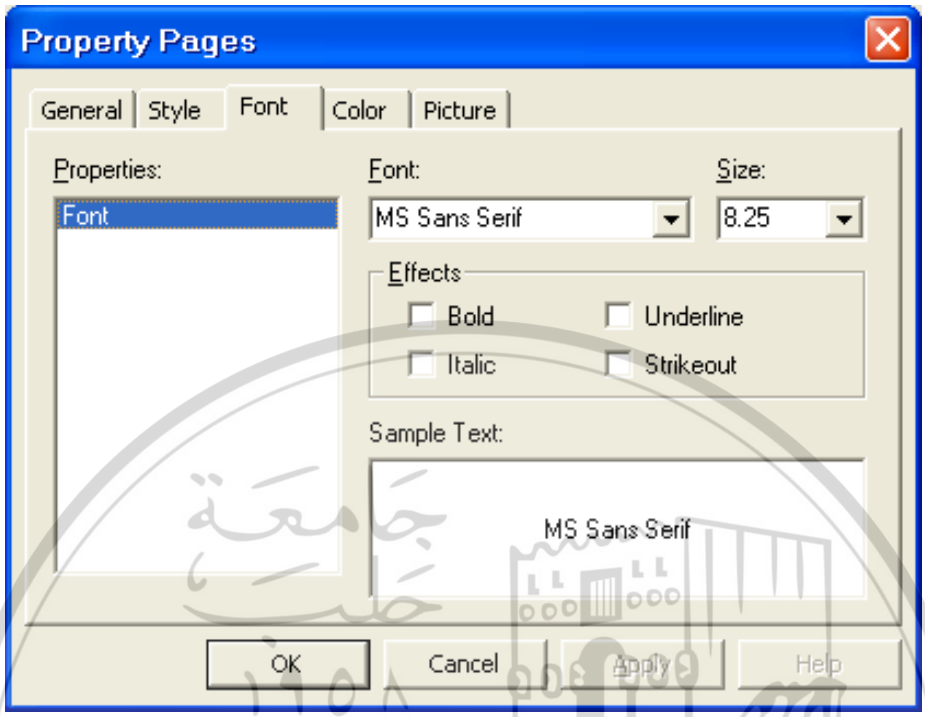
- مربع الحوار Property Pages - الصفحة عام General:



- مربع الحوار Property Pages - الصفحة نمط Style:

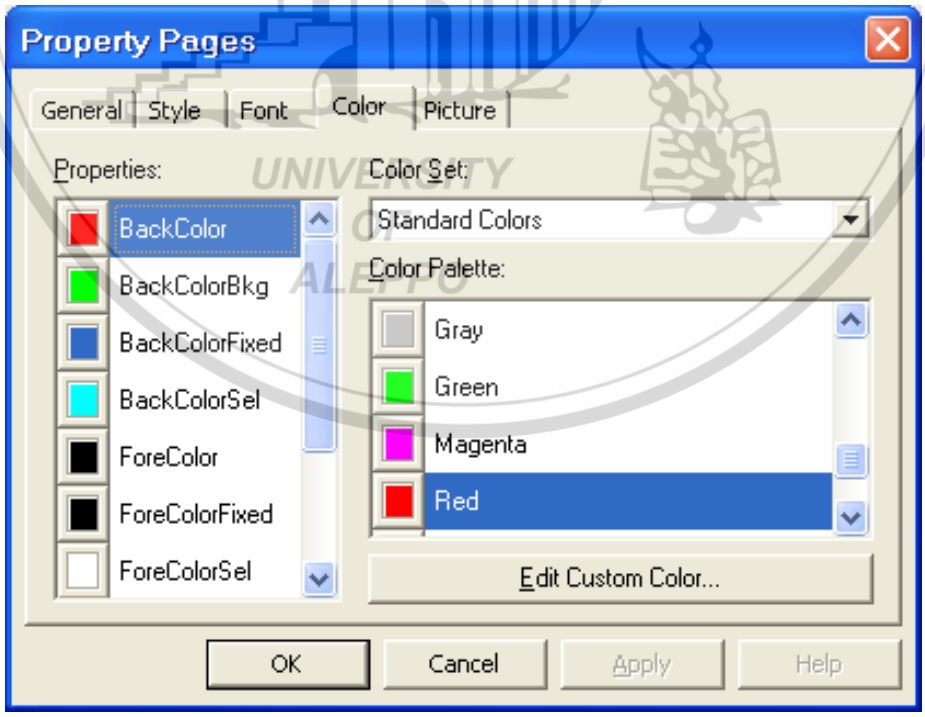


- مربع الحوار Property Pages - الصفحة خط Font:



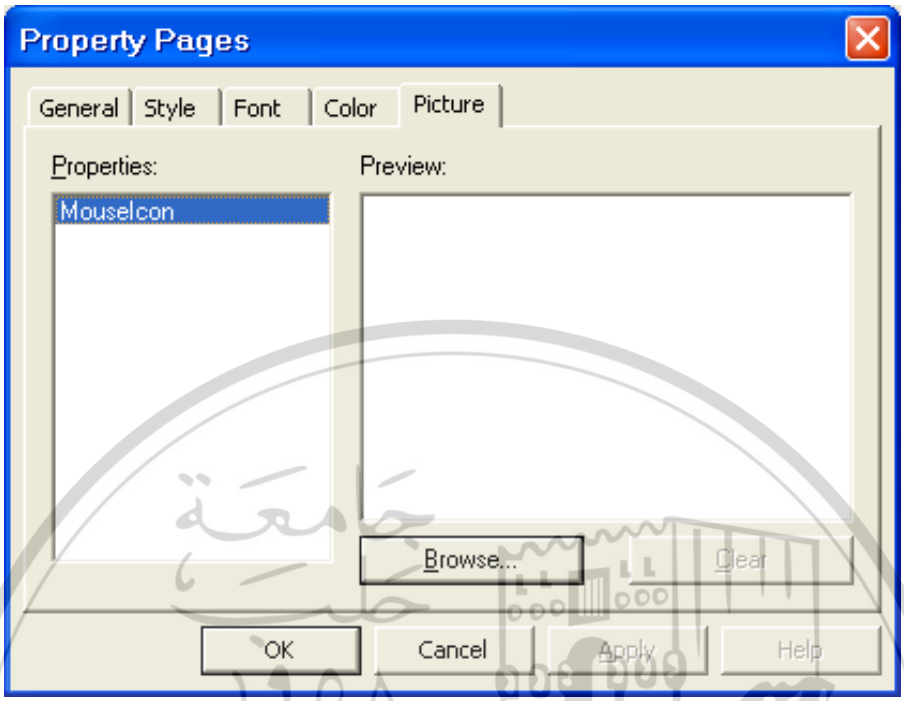
وهي خاصية تمكن المستخدم من القيام بتغيير مواصفات الخط مثل نوع الخط وحجم الخط ولون الخط والتأثيرات الإضافية والتنسيقية الأخرى الممكن تعديلها على الخط.

- مربع الحوار Property Pages - الصفحة لون Color:



وهي خاصية تمكن المستخدم من القيام بتغيير خيارات الممكنة.

- مربع الحوار Property Pages - الصفحة صورة Picture:



جامعة حلب
١٩٥٨
UNIVERSITY
OF
ALEPPO

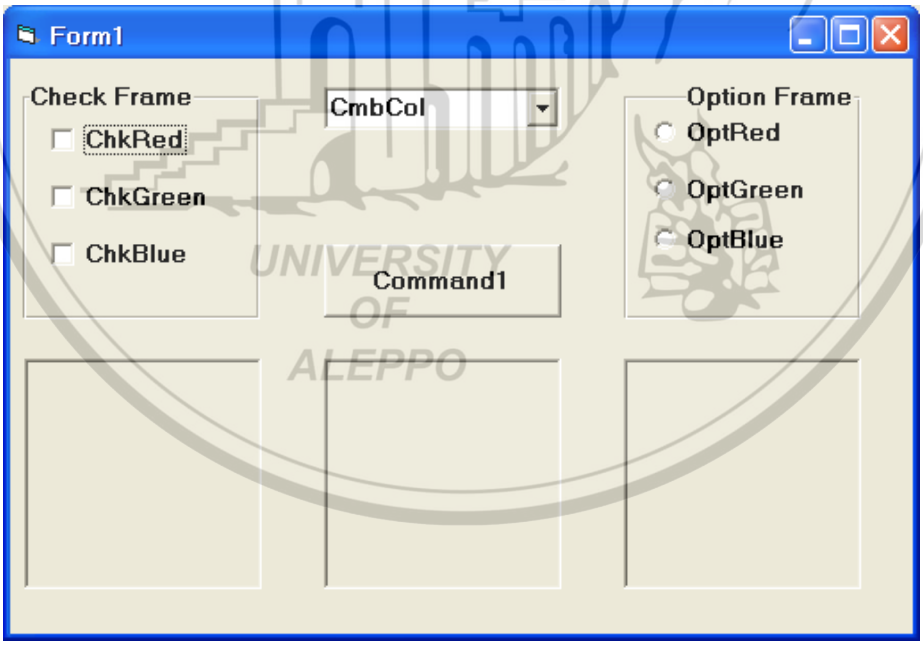
الفصل الثامن عشر
أمثلة محلولة
Resolved Examples

التمرين (١) – الألوان وأدوات التحكم وأدوات الإدخال والإخراج:

الهدف من التمرين:

١) التدرب على أدوات التحكم والتي تقوم مقام التعليمات الشرطية.

- تسمح أداة التحكم $Option\ Button$ باستخدام خيار وحيد من بين الخيارات المتاحة.
- تسمح أداة التحكم $Check\ Box$ بأخذ خيار وحيد أو عدة خيارات من بين الخيارات المتاحة.
- تسمح أداة التحكم $Combo\ Box$ بانتقاء خيار من بين عدة خيارات ممكنة ويمكن أن يتم ذلك انتقاءً أو كتابة حسب خصائص الأداة.

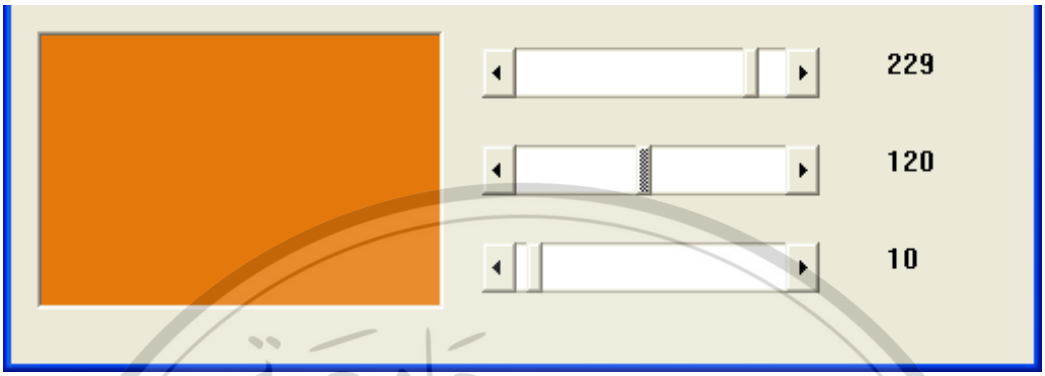


٢) التدرّب على طريقة إدخال القيم من خلال مربعات الإدخال والحصول على الملاحظات والتنويهات من خلال مربعات الإخراج.

٣) التدرب على استخدام الألوان بالطرق الثلاثة المعروفة.

ملاحظة (١):

يمكن تطوير التمرين واستخدام أداة السحابات $Scroll\ Bar$ لاختيار شدة أو كثافة اللون المطلوب أثناء عمليات مزج الألوان.



ملاحظة (٢):

التمرين طويل جداً إذا قمنا بكتابة الأوامر المطلوبة كلها. لكن نلاحظ أوامر متشابهة والاختلافات قليلة فيما بينها لذا يمكن ببساطة وبسرعة إنجاز التمرين إذا أحسننا استخدام أوامر النسخ واللصق والتعديل.

المرحلة الأساسية من التمرين:

```
Dim RCol, GCol, BCol As Integer
Private Sub ChkRed_Click( )
If ChkRed.Value = 0 Then
  RCol = 0
ElseIf ChkRed.Value = 1 Then
  RCol = 255
End If
PicCheck.BackColor = RGB(RCol, GCol, BCol)
End Sub
```

```
Private Sub ChkGreen_Click( )
If ChkGreen.Value = 0 Then
  GCol = 0
ElseIf ChkGreen.Value = 1 Then
  GCol = 255
End If
PicCheck.BackColor = RGB(RCol, GCol, BCol)
End Sub
```

```
Private Sub ChkBlue_Click()
If ChkBlue.Value = 0 Then
  BCol = 0
ElseIf ChkBlue.Value = 1 Then
  BCol = 255
End If
PicCheck.BackColor = RGB(RCol, GCol, BCol)
End Sub
```

```
Private Sub CmbCol_Change()
If CmbCol.Text = "Red" Then
  PicCombo.BackColor = vbRed
ElseIf CmbCol.Text = "Green" Then
  PicCombo.BackColor = vbGreen
ElseIf CmbCol.Text = "Blue" Then
  PicCombo.BackColor = vbBlue
End If
End Sub
```

```
Private Sub CmbCol_Click()
If CmbCol.Text = "Red" Then
  PicCombo.BackColor = vbRed
ElseIf CmbCol.Text = "Green" Then
  PicCombo.BackColor = vbGreen
ElseIf CmbCol.Text = "Blue" Then
  PicCombo.BackColor = vbBlue
End If
End Sub
```

```
Private Sub CmbCol_DropDown()
If CmbCol.Text = "Red" Then
  PicCombo.BackColor = vbRed
ElseIf CmbCol.Text = "Green" Then
  PicCombo.BackColor = vbGreen
ElseIf CmbCol.Text = "Blue" Then
  PicCombo.BackColor = vbBlue
End If
End Sub
```

```
Private Sub OptRed_Click( )
If OptRed.Value = True Then
  PicOption.BackColor = vbRed
ElseIf OptGreen.Value = True Then
  PicOption.BackColor = vbGreen
ElseIf OptBlue.Value = True Then
  PicOption.BackColor = vbBlue
End If
End Sub
```

```
Private Sub OptGreen_Click( )
If OptRed.Value = True Then
  PicOption.BackColor = vbRed
ElseIf OptGreen.Value = True Then
  PicOption.BackColor = vbGreen
ElseIf OptBlue.Value = True Then
  PicOption.BackColor = vbBlue
End If
End Sub
```

```
Private Sub OptBlue_Click( )
If OptRed.Value = True Then
  PicOption.BackColor = vbRed
ElseIf OptGreen.Value = True Then
  PicOption.BackColor = vbGreen
ElseIf OptBlue.Value = True Then
  PicOption.BackColor = vbBlue
End If
End Sub
```

```
Private Sub Command1_Click( )
Num = InputBox
    ("Insert Number Between 0 and 15","Select Color",0)
For I = Num To 15
  MsgBox ("The Index is  " & I)
  Form1.BackColor = QBColor(I)
Next
End Sub
```

المرحلة المُتقدمة من التمرين:

تتم هذه المرحلة باستخدام الأدوات الجديدة التالية:

- مربع صورة جديد اسمه $PicScroll$.
- ثلاثة سحابات أفقية تمثل كثافة الألوان الثلاثة المشهورة.
- ثلاث لافتات للتعبير عن شدة الألوان المأخوذة من قيم السحابات المقابلة.

```
Private Sub hscRed_Scroll( )
lblRed.Caption = Val(hscRed.Value)
lblGreen.Caption = Val(hscGreen.Value)
lblBlue.Caption = Val(hscBlue.Value)
PicScroll.BackColor
= RGB(hscRed.Value, hscGreen.Value, hscBlue.Value)
End Sub
```

```
Private Sub hscRed_Change( )
lblRed.Caption = Val(hscRed.Value)
lblGreen.Caption = Val(hscGreen.Value)
lblBlue.Caption = Val(hscBlue.Value)
PicScroll.BackColor
= RGB(hscRed.Value, hscGreen.Value, hscBlue.Value)
End Sub
```

```
Private Sub hscGreen_Scroll( )
lblRed.Caption = Val(hscRed.Value)
lblGreen.Caption = Val(hscGreen.Value)
lblBlue.Caption = Val(hscBlue.Value)
PicScroll.BackColor
= RGB(hscRed.Value, hscGreen.Value, hscBlue.Value)
End Sub
```

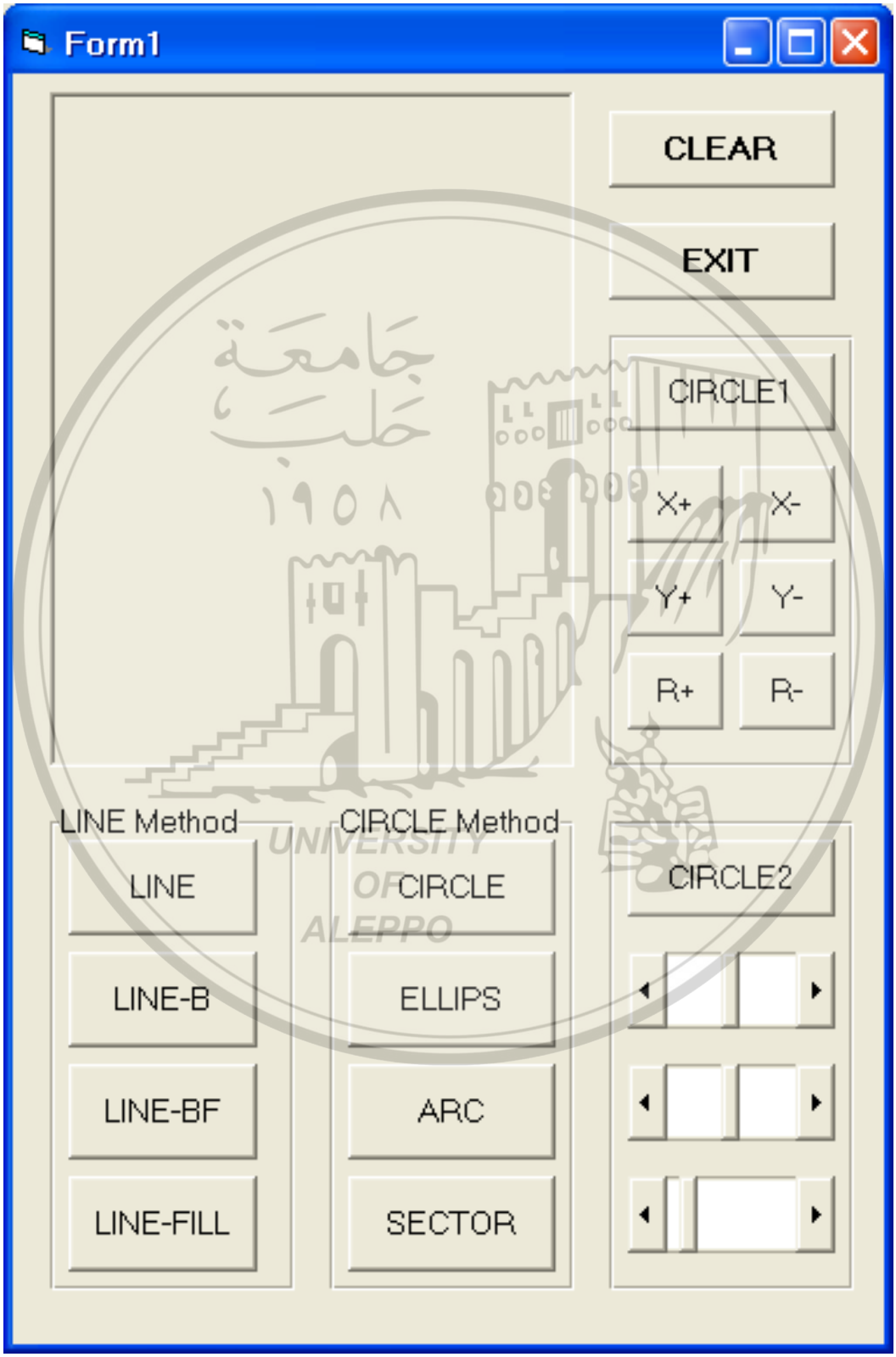
```
Private Sub hscGreen_Change( )
lblRed.Caption = Val(hscRed.Value)
lblGreen.Caption = Val(hscGreen.Value)
lblBlue.Caption = Val(hscBlue.Value)
PicScroll.BackColor
= RGB(hscRed.Value, hscGreen.Value, hscBlue.Value)
End Sub
```

```
Private Sub hscBlue_Scroll( )
lblRed.Caption = Val(hscRed.Value)
lblGreen.Caption = Val(hscGreen.Value)
lblBlue.Caption = Val(hscBlue.Value)
PicScroll.BackColor
= RGB(hscRed.Value, hscGreen.Value, hscBlue.Value)
End Sub
```

```
Private Sub hscBlue_Change( )
lblRed.Caption = Val(hscRed.Value)
lblGreen.Caption = Val(hscGreen.Value)
lblBlue.Caption = Val(hscBlue.Value)
PicScroll.BackColor
= RGB(hscRed.Value, hscGreen.Value, hscBlue.Value)
End Sub
```

التمرين (٢) – طرائق الرسم *Drawing Methods*:

الهدف من التمرين:



١) التدرب على أوامر الرسم.

- طريقة رسم قطعة مستقيمة أو صندوق $Line\ Method$ وجميع حالاتها الخاصة.
- طريقة رسم دائرة أو قوس أو قطاع أو قطع ناقص $Circle\ Method$ وجميع حالاتها الخاصة.
- استخدام أنماط مختلفة من التعبئة.

٢) التدرّب على استخدام أنماط مختلفة من التعبئة.

٣) التدرب على استخدام الألوان بالطرق المعروفة.

ملاحظة:

التمرين طويل جداً وهناك أوامر متشابهة والاختلافات فيما بينها قليلة جداً، لذا يمكن ببساطة ولسرعة إنجاز التمرين استخدام أوامر النسخ واللصق والتعديل.

حل التمرين (٢) – طرائق الرسم *Drawing Methods*:

تعريف إحداثيات الدائرة. (يتم التعريف في قسم التصاريح العامة).	$Dim\ x, y, r$ $'Const\ pi\ =\ 3.14159265$ $'Const\ pi\ =\ 22\ /\ 7$

الخروج وإنهاء البرنامج	تنظيف مربع الصورة
Private Sub CmdEXIT_Click() *End* ***End Sub***	***Private Sub CmdCLEAR_Click***() *PBOX.Cls* ***End Sub***

١) أوامر رسم الدائرة (*Circle Method*).

- رسم دائرة في مربع الصورة.

```
Private Sub CmdCircle_Click()
PBOX.Circle (1000, 1000), 500
End Sub
```

- رسم قطع ناقص في مربع الصورة. يمكن للقطع الناقص أن يكون أفقيّاً أو عموديّاً.

```
Private Sub CmdEllips_Click()
'PBox.Circle (1000, 1000), 500, vbRed,,, 2
PBOX.Circle (1000, 1000), 500, vbRed,,, 0.5
End Sub
```

- رسم قوس في مربع الصورة. يمكن للقوس أن يكون قوس من قطع ناقص.

```
Private Sub CmdARC_Click( )
PBOX.Circle (2000, 2000), 1500, vbRed, pi / 4, pi, 2
End Sub
```

- رسم قطاع في مربع الصورة. يمكن للقطاع أن يكون قطاع من قطع ناقص.

```
Private Sub CmdSector_Click()
PBOX.Circle (2000, 2000), 1500, vbGreen, −pi, −pi / 4, 2
End Sub
```

٢) أوامر رسم الخط (*Line Method*).

- رسم خطين مستقيمين، النقطة الثانية تختلف في كل منهما. إحداها منسوبة إلى نقطة الأصل والأخرى منسوبة إلى نقطة أخرى.

```
Private Sub CmdLine_Click()
PBOX.Line (400, 800) − (1400, 1800)
PBOX.Line (400, 800) − Step(1400, 1800), vbBlue
End Sub
```

- رسم صندوقين مفرغين، النقطة الثانية تختلف في كل منهما. إحداها منسوبة إلى نقطة الأصل والأخرى منسوبة إلى نقطة أخرى.

```
Private Sub CmdLineB_Click()
PBOX.DrawWidth = 2
PBOX.Line (200, 400) − (1200, 2400), vbRed, B
PBOX.Line (200, 400) − Step(1200, 2400), vbBlue, B
End Sub
```

- رسم صندوقين أحدهما مصمت والآخر مفرغ، النقطة الثانية تختلف في كل منهما. إحداها منسوبة إلى نقطة الأصل والأخرى منسوبة إلى نقطة أخرى.

```
Private Sub CmdLineBF_Click()
PBOX.DrawWidth = 2
PBOX.Line (200, 400) − (1200, 2400), vbRed, BF
PBOX.Line (200, 400) − Step(1200, 2400), vbBlue, B
End Sub
```

- صندوقين أحدهما مصمت والآخر مفرغ، النقطة الثانية تختلف في كل منهما. إحداها منسوبة إلى نقطة الأصل والأخرى منسوبة إلى نقطة أخرى. لاحظ استخدام التعبئة.

```
Private Sub CmdLineFill_Click()
PBOX.DrawWidth = 3
PBOX.FillStyle = 3
PBOX.FillColor = RGB(0, 255, 0)
```

```
PBOX.Line (400,800) − (1400,1800), vbRed, B
'PBox.Line (400,800) − Step(1400,1800), vbBlue, BF
End Sub
```

- رسم دائرة أبعادها معروفة.

```
Private Sub CmdCIR1_Click()
x = 500
y = 500
r = 500
PBOX.Circle (x,y), r
End Sub
```

- تغيير نصف قطر الدائرة زيادة ونقصاناً (يظهر أعداد كبيرة من الدوائر).
- يمكن استخدام أوامر مسح (في بداية الكود) فتظهر الدوائر وكأنها تكبر أو تصغر.

```
Private Sub RPLUS_Click()
r = r + 10
PBOX.Circle (x,y), r
End Sub
```

```
Private Sub RMIN_Click()
r = r − 10
PBOX.Circle (x,y), r
End Sub
```

- تغيير مركز الدائرة إلى اليمين وإلى اليسار (يظهر أثناء الحركة دائرة مصمتة تتحرك أفقيّاً إلى اليسار وإلى اليمين).
- يمكن استخدام أوامر مسح (في بداية الكود) فتظهر الدوائر وكأنها تتحرك.

```
Private Sub XPLUS_Click()
x = x + 10
PBOX.Circle (x,y), r
End Sub
```

```
Private Sub XMIN_Click()
x = x − 10
PBOX.Circle (x,y), r
End Sub
```

- تغيير مركز الدائرة إلى إلى الأعلى والأسفل (يظهر أثناء الحركة دائرة مصمتة تتحرك عموديّاً إلى الأعلى وإلى الأسفل).
- يمكن استخدام أوامر مسح (في بداية الكود) فتظهر الدوائر وكأنها تتحرك.

```
Private Sub YPLUS_Click()
y = y + 10
PBOX.Circle (x,y), r
End Sub
```

```
Private Sub YMIN_Click()
y = y − 10
PBOX.Circle (x,y), r
End Sub
```

- يمكن رسم الدائرة وتغيير أبعادها من خلال أزرار أوامر.
- ويمكن نسب مركز الدائرة أو نصف قطرها إلى مربع الصورة أو إلى سحابات.

- ويمكن استخدام أوامر المسح أيضاً.

```
Private Sub CmdCIR2_Click()
'x = PBOX.ScaleWidth / 2
'y = PBOX.ScaleHeight / 2
x = HX.Value
y = HY.Value
r = HR.Value
'PBOX.Cls
PBOX.Circle (x,y),r
End Sub
```

- تغيير مركز الدائرة إلى اليمين وإلى اليسار (يظهر أثناء الحركة دائرة مصمتة تتحرك أفقيّاً إلى اليسار وإلى اليمين).

```
Private Sub HX_Scroll()
x = HX.Value
y = HY.Value
r = HR.Value
'PBOX.Cls
PBOX.Circle (x,y),r
End Sub
```

```
Private Sub HX_Change()
x = HX.Value
y = HY.Value
r = HR.Value
'PBOX.Cls
PBOX.Circle (x,y),r
End Sub
```

- تغيير مركز الدائرة إلى إلى الأعلى والأسفل (يظهر أثناء الحركة دائرة مصمتة تتحرك عموديّاً إلى الأعلى وإلى الأسفل).

```
Private Sub HY_Scroll()
x = HX.Value
y = HY.Value
r = HR.Value
'PBOX.Cls
PBOX.Circle (x,y),r
End Sub
```

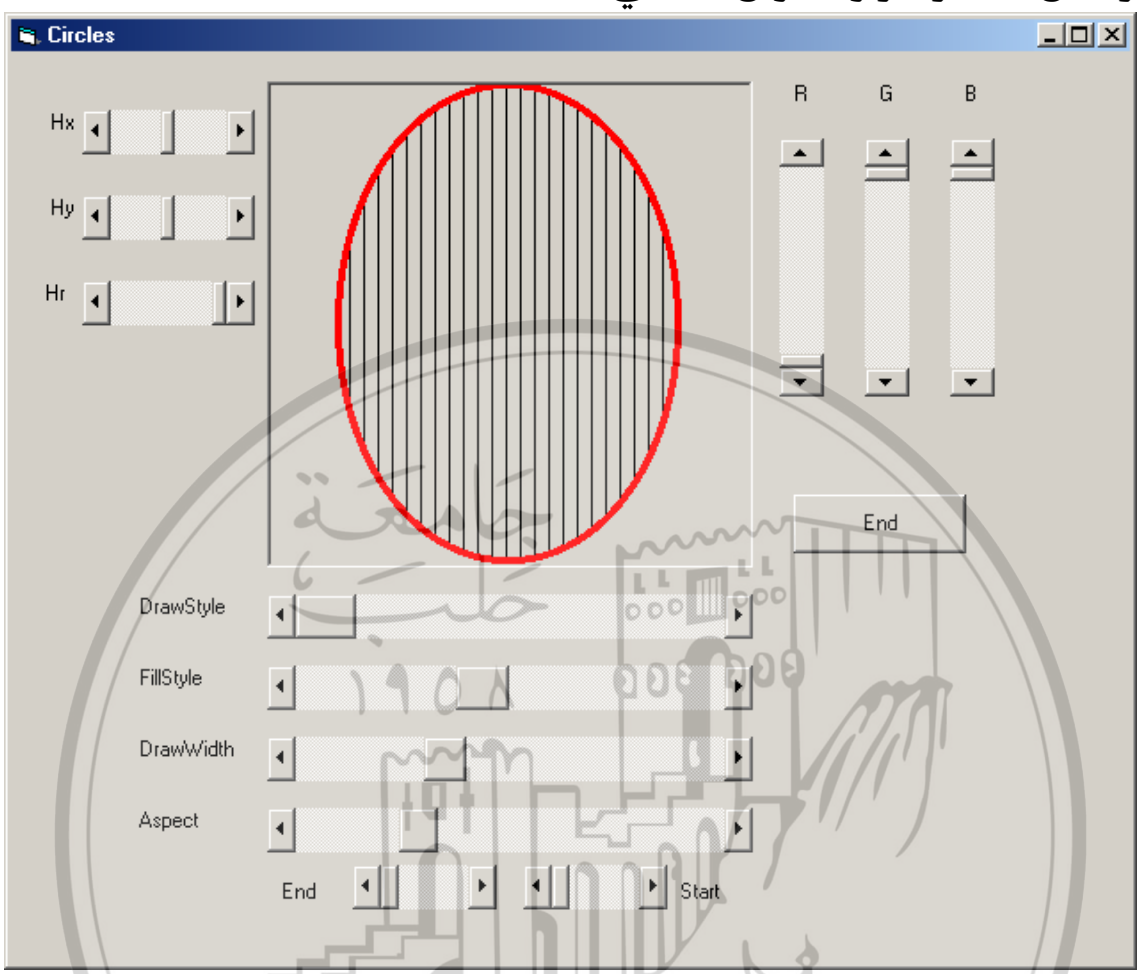
```
Private Sub HY_Change()
x = HX.Value
y = HY.Value
r = HR.Value
'PBOX.Cls
PBOX.Circle (x,y),r
End Sub
```

- تغيير نصف قطر الدائرة زيادة ونقصاناً (يظهر أعداد كبيرة من الدوائر).

```
Private Sub HR_Change()
x = HX.Value
y = HY.Value
r = HR.Value
'PBOX.Cls
PBOX.Circle (x,y),r
End Sub
```

```
Private Sub HR_Scroll()
x = HX.Value
y = HY.Value
r = HR.Value
'PBOX.Cls
PBOX.Circle (x,y),r
End Sub
```

ويمكن تنفيذ وتطوير التمرين كما يلي:



الكائن	الخاصية	القيمة
Picture Box	Name	PBox1
	DrawStyle	0
	DrawWidth	1
	FillStyle	1
	Height	4095
	Width	4095
Horizontal Scroll Bar	Name (x,y)	Hx , Hy
	Min	1
	Max	4000
	Small Chang	10

	Large Chang	100
	Value	2000
Horizontal Scroll Bar	Name (radius)	Hr
	Min	1
	Max	2000
	Small Chang	10
	Large Chang	100
	Value	2000
Vertical Scroll Bar	Name (R, G , B)	VR , VG , VB
	Min	0
	Max	255
	Small Chang	1
	Large Chang	10
	value	0
Horizontal Scroll Bar	Name (Draw Width)	Hds
	Min	0
	Max	6
	Small Chang	1
	Large Chang	1
	value	0
Horizontal Scroll Bar	Name (Aspect)	Hxy
	Min	1
	Max	50
	Small Chang	1
	Large Chang	5
	value	10
Horizontal Scroll Bar	Name (FillStyle)	Hfs
	Min	0
	Max	7
	Small Chang	1
	Large Chang	1

	value	0
Horizontal Scroll Bar	Name (Draw Width)	Hdw
	Min	1
	Max	10
	Small Chang	1
	Large Chang	1
	value	1
Horizontal Scroll Bar	Name (Start , End)	HArc1 , Harc2
	Min	0
	Max	360
	Small Chang	10
	Large Chang	90
	Value	0

```
Dim x, y, r
Const pi = 3.14159265
'Const pi = 4 * Atn(1)

Private Sub PBox1_Click()
PBox1.Cls
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
                    RGB(VRed.Value, VGreen.Value, VBlue.Value)
End Sub

Private Sub Hx_Change()
PBox1.Cls
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
                    RGB(VRed.Value, VGreen.Value, VBlue.Value)
End Sub

Private Sub Hx_Scroll()
PBox1.Cls
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
                    RGB(VRed.Value, VGreen.Value, VBlue.Value)
End Sub
```

Private Sub Hy_Change()
$PBox1.Cls$
$PBox1.Circle\ (Hx.Value, Hy.Value), Hr.Value,$
$RGB(VRed.Value, VGreen.Value, VBlue.Value)$
End Sub

Private Sub Hy_Scroll()
$PBox1.Cls$
$PBox1.Circle\ (Hx.Value, Hy.Value), Hr.Value,$
$RGB(VRed.Value, VGreen.Value, VBlue.Value)$
End Sub

Private Sub Hr_Change()
$PBox1.Cls$
$PBox1.Circle\ (Hx.Value, Hy.Value), Hr.Value,$
$RGB(VRed.Value, VGreen.Value, VBlue.Value)$
End Sub

Private Sub Hr_Scroll()
$PBox1.Cls$
$PBox1.Circle\ (Hx.Value, Hy.Value), Hr.Value,$
$RGB(VRed.Value, VGreen.Value, VBlue.Value)$
End Sub

Private Sub VRed_Change()
$PBox1.Cls$
$PBox1.Circle\ (Hx.Value, Hy.Value), Hr.Value,$
$RGB(VRed.Value, VGreen.Value, VBlue.Value)$
End Sub

Private Sub VRed_Scroll()
$PBox1.Cls$
$PBox1.Circle\ (Hx.Value, Hy.Value), Hr.Value,$
$RGB(VRed.Value, VGreen.Value, VBlue.Value)$
End Sub

Private Sub VGreen_Change()

```
PBox1.Cls
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
                    RGB(VRed.Value, VGreen.Value, VBlue.Value)
End Sub

Private Sub VGreen_Scroll()
PBox1.Cls
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
                    RGB(VRed.Value, VGreen.Value, VBlue.Value)
End Sub

Private Sub VBlue_Change()
PBox1.Cls
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
                    RGB(VRed.Value, VGreen.Value, VBlue.Value)
End Sub

Private Sub VBlue_Scroll()
PBox1.Cls
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
                    RGB(VRed.Value, VGreen.Value, VBlue.Value)
End Sub

Private Sub Hds_Change()
PBox1.Cls
PBox1.DrawStyle = Val(Hds.Value)
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
                    RGB(VRed.Value, VGreen.Value, VBlue.Value)
End Sub

Private Sub Hds_Scroll()
PBox1.Cls
PBox1.DrawStyle = Val(Hds.Value)
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
                    RGB(VRed.Value, VGreen.Value, VBlue.Value)
End Sub

Private Sub Hfs_Change()
PBox1.Cls
```

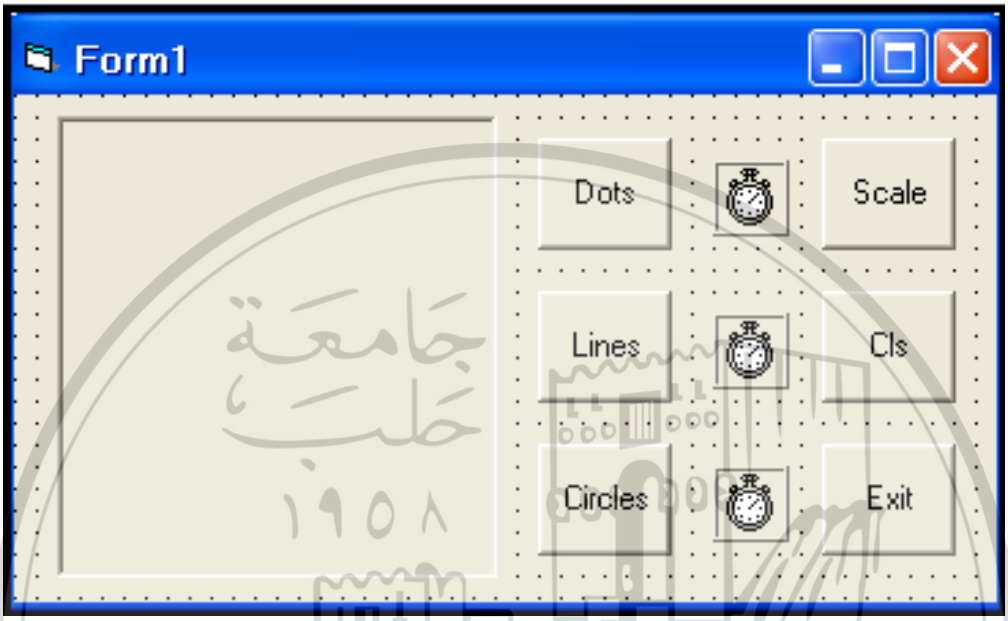
```
PBox1.DrawStyle = Val(Hds.Value)
PBox1.FillStyle = Val(Hfs.Value)
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
                    RGB(VRed.Value, VGreen.Value, VBlue.Value)
End Sub

Private Sub Hfs_Scroll()
PBox1.Cls
PBox1.DrawStyle = Val(Hds.Value)
PBox1.FillStyle = Val(Hfs.Value)
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
                    RGB(VRed.Value, VGreen.Value, VBlue.Value)
End Sub

Private Sub Hdw_Change()
PBox1.Cls
PBox1.DrawStyle = Val(Hds.Value)
PBox1.FillStyle = Val(Hfs.Value)
PBox1.DrawWidth = Val(Hdw.Value)
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
                    RGB(VRed.Value, VGreen.Value, VBlue.Value)
End Sub

Private Sub Hdw_Scroll()
PBox1.Cls
PBox1.DrawStyle = Val(Hds.Value)
PBox1.FillStyle = Val(Hfs.Value)
PBox1.DrawWidth = Val(Hdw.Value)
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
                    RGB(VRed.Value, VGreen.Value, VBlue.Value)
End Sub

Private Sub Hxy_Change()
PBox1.Cls
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
        RGB(VRed.Value, VGreen.Value, VBlue.Value), , , Hxy / 10
End Sub

Private Sub Hxy_Scroll()
PBox1.Cls
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
```

```
        RGB(VRed.Value, VGreen.Value, VBlue.Value), , , Hxy / 10
End Sub

Private Sub Harc1_Change()
PBox1.Cls
PBox1.DrawStyle = Val(Hds.Value)
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
 RGB(VRed.Value, VGreen.Value, VBlue.Value), Harc1.Value * -2
            * pi/360, Harc2.Value * -2 * pi/360, Hxy/10
End Sub

Private Sub Harc1_Scroll()
PBox1.Cls
PBox1.DrawStyle = Val(Hds.Value)
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
 RGB(VRed.Value, VGreen.Value, VBlue.Value), Harc1.Value * -2
            * pi/360, Harc2.Value * -2 * pi/360, Hxy/10
End Sub

Private Sub Harc2_Change()
PBox1.Cls
PBox1.DrawStyle = Val(Hds.Value)
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
 RGB(VRed.Value, VGreen.Value, VBlue.Value), Harc1.Value * -2
            * pi/360, Harc2.Value * -2 * pi/360, Hxy/10
End Sub

Private Sub Harc2_Scroll()
PBox1.Cls
PBox1.DrawStyle = Val(Hds.Value)
PBox1.Circle (Hx.Value, Hy.Value), Hr.Value,
 RGB(VRed.Value, VGreen.Value, VBlue.Value), Harc1.Value * -2
            * pi/360, Harc2.Value * -2 * pi/360, Hxy/10
End Sub

Private Sub Command1_Click()
End
End Sub
```

التمرين (٣) – طرائق الرسم *Drawing Methods – Scale*:

الهدف من التمرين:

صمم الواجهة المرئية التالية وقم بتنفيذ الأوامر كما هو مبيّن:



١) رسم النقاط:

- المطلوب رسم نقاط عشوائية يصل عددها حتى 1000 نقطة واقعة ضمن منطقة حدود الرسم في مربع الصورة.
- اجعل هذه النقاط عشوائية اللون.
- غيّر أحجام هذه النقاط.
- اجعل هذه النقاط منسوبة إلى مركز الإحداثيات (نقطة الأصل) تارةً ومنسوبة إلى منتصف أداة الرسم تارةً أخرى.
- استخدم زر أوامر يقوم بتنشيط وإلغاء تنشيط مؤقت زمني. ويقوم بتغيير خاصية الاسم المرئي بحيث يتلاءم مع طبيعة تشغيل أو عد تشغيل المؤقت.

٢) رسم الخطوط:

- استخدام زر أوامر يقوم برسم 1000 قطعة مستقيمة عشوائية تبدأ من مركز الإحداثيات.
- يمكن جعل ألوان هذه الكائنات عشوائية.

⇦ يمكن أن تكون هذه الكائنات منسوبة إلى نقطة الأصل أو إلى مركز الإحداثيات الجديد الواقع في منتصف أداة الرسم. قارن !!

⇦ يمكن استخدام مؤقت يقوم بتنشيط أو توقيف تنشيط رسم الخطوط.

٣) رسم الدوائر:

⏲ رسم 100 دائرة مكان مركزها متغير عشوائيّاً ونصف قطرها ثابت.

⏲ يمكن تغيير لون الدوائر ونسبتها إلى نقطة الأصل أو مركز الإحداثيات الواقع في منتصف أداة مربع الصورة.

⏲ رسم 5000 دائرة مكان مركزها ثابت ونصف قطرها متغيّر تكبر من الداخل إلى الخارج أو تصغر من الخارج إلى الداخل.

⏲ يمكن تغيير لون الدوائر ونسبتها إلى نقطة الأصل أو مركز الإحداثيات الواقع في منتصف أداة مربع الصورة.

⏲ يمكن جعل تغير نصف القطر عشوائي أيضاً.

⏲ يمكن استخدام مؤقت يقوم بتنشيط أو توقيف تنشيط رسم الخطوط.

٤) رسم التابع الرياضي:

♫ يجب إضافة زر أوامر لرسم مجموعة من النقاط العشوائية الألوان والتي إحداثياتها (X, Y) والتي يمثِّلها التابع الرياضي:

$$\theta_{tot} = 2\pi$$
$$a = 500 * Rnd$$
$$r = a * Sin(2\theta)$$
$$x = r * Cos(\theta)$$
$$y = r * Sin(\theta)$$

♫ يمكن استخدام مؤقت لرسم هذه النقاط أو لتفعيل أو تنشيط زر الأوامر الذي يقوم بذلك.

♫ يمكن تغيير القيم المتعلقة برسم هذا التابع (غيّر – قارن – استمتع) !!!

♫ لاحظ عمدما تكون قيم زوجية أو فردية.

معلومات لا بد من التدرب عليها وفهمها لإنجاز التمرين:

معلومات عن أوامر الرسم:

- رسم نقطة لا لون لها (أي اللون افتراضي)
- رسم نقطة لونها محدد

$P1.PSet\ (X,Y)$

$P1.PSet\ (X,Y),\quad COLOR$

- رسم قطعة مستقيمة لونها افتراضي تبدأ من مركز الإحداثيات
- رسم قطعة مستقيمة لونها افتراضي
- رسم قطعة مستقيمة لونها محدد تبدأ من مركز الإحداثيات

$P1.Line\ (0,0)\ -\ (X,Y)$

$P1.Line\ (X1,Y1)\ -\ (X2,Y2)$

$P1.Line\ (0,0)\ -\ (X,Y)\ ,COLOR$

- رسم دائرة لونها افتراضي مركزها هو مركز الإحداثيات
- رسم دائرة لونها افتراضي مركزها مختلف عن مركز الإحداثيات
- رسم دائرة لونها محدد مركزها مختلف عن مركز الإحداثيات

$P1.Circle\ (0,0)\ ,R$

$P1.Circle\ (X,Y)\ ,R$

$P1.Circle\ (X,Y)\ ,R\ ,COLOR$

معلومات عن نقل مركز الإحداثيات إلى نقطة غير نقطة الأصل:

- جعل مركز إحداثيات الصورة في منتصف مربع الصورة ونسب مقياس الرسم إلى قيم محددة

$P1.Scale\ (-100,100)\ -\ (500,-500)$

$P1.Scale\ (-500,500)\ -\ (500,-500)$

- جعل مركز إحداثيات الصورة في منتصف مربع الصورة ونسب مقياس الرسم إلى أبعاد مربع الصورة

$P1.Scale\ (-P1.ScaleWidth\ /\ 2, P1.ScaleHeight\ /\ 2)$
$-\ (\ P1.ScaleWidth\ /\ 2, -\ P1.ScaleHeight\ /\ 2)$

معلومات عن القيم العشوائية:

- تابع العشوائية ($0 \leq \text{Rnd} < 1$) يولد قيمة عشوائية أكبر أو تساوي الصفر وأصغر تماماً من الواحد.
- تابع التقريب Int يعطي أكبر قيمة صحيحة أكبر أو يساوي التعبير العددي الحقيقي.
- انتقاء نقطة عشوائية واقعة ضمن حدود أداة مربع الصورة:

$$Let\ X\ =\ Int(P1.ScaleHeight\ *\ Rnd)$$
$$Let\ Y\ =\ Int(P1.ScaleWidth\ *\ Rnd$$

- انتقاء ثلاث قيم صحيحة عشوائية قيمة كلٍّ منها أكبر أو تساوي الصفر وأصغر تماماً من 256 لتمثيل اللون باستخدام الخاصية RGB أي:
- $0 \leq RC, GC, BC \leq 255$

$$Let\ RC\ =\ Int\ (256\ *\ Rnd)$$
$$Let\ GC\ =\ Int\ (256\ *\ Rnd)$$
$$Let\ BC\ =\ Int\ (256\ *\ Rnd)$$

تنشيط وتفعيل مؤقت ما وتغيير خصائص زر أوامر:

- استخدم زر أوامر يقوم بتنشيط وإلغاء تنشيط مؤقت زمني. ويقوم بتغيير خاصية الاسم المرئي بحيث يتلاءم مع طبيعة تشغيل أو عد تشغيل المؤقت.

```
Private Sub Stars_Click()
If Timer1.Enabled = True Then
  Stars.Caption = "GO"
  Timer1.Enabled = False
Else
  Timer1.Enabled = True
  Stars.Caption = "STOP"
End If
End Sub
```

ملاحظات:

☺ على الطالب أن يقوم بقراءة التعليمات وفهمها قبل تنفيذ التمرين.

☺ على الطالب أن يقوم بكل التعديلات والتغييرات والتي من خلالها يستطيع رؤية احتمالات أكثر والتي تدعم فهم وتركيز التمرين.

المرحلة الكودية:

١. رسم النقاط:

a) رسم النقاط باللون الافتراضي (إحداثيات النقاط عشوائية):

```
Private Sub CmdDot1_Click()
For Dot1 = 1 To 100
 Let X = Int(Picture1.ScaleHeight * Rnd)
 Let Y = Int(Picture1.ScaleWidth * Rnd)
 Picture1.PSet (X,Y)
Next Dot1
End Sub
```

b) رسم النقاط (إحداثيات النقاط عشوائية وألوانها عشوائية):

```
Private Sub CmdDot2_Click()
Picture1.DrawWidth = 3
For Dot2 = 1 To 100
 Let X = Int(Picture1.ScaleHeight * Rnd)
 Let Y = Int(Picture1.ScaleWidth * Rnd)
 Let RC = Int(256 * Rnd)
 Let GC = Int(256 * Rnd)
 Let BC = Int(256 * Rnd)
 Picture1.PSet (X,Y),RGB(RC,GC,BC)
Next Dot2
End Sub
```

c) رسم النقاط (إحداثيات النقاط عشوائية وألوانها عشوائية)- استخدام المؤقت:

```
Private Sub TimDot_Timer()
Picture1.DrawWidth = 3
For Dot2 = 1 To 100
 Let X = Int(Picture1.ScaleHeight * Rnd)
 Let Y = Int(Picture1.ScaleWidth * Rnd)
 Let RC = Int(256 * Rnd)
 Let GC = Int(256 * Rnd)
 Let BC = Int(256 * Rnd)
 Picture1.PSet (X,Y),RGB(RC,GC,BC)
Next Dot2
End Sub
```

d) زر تنشيط وإيقاف المؤقت وتغيير خصائص زر الأوامر:

```
Private Sub CmdDot_Click()
If TimDot.Enabled = True Then
```

```
  CmdDot.Caption = "GO"
  TimDot.Enabled = False
Else
  TimDot.Enabled = True
  CmdDot.Caption = "STOP"
End If
End Sub
```

٢. رسم الخطوط:

a) رسم الخطوط باللون الافتراضي (إحداثيات النقطة الثانية عشوائية):

```
Private Sub CmdLin1_Click()
For Lin1 = 1 To 100
 Let X = Int(Picture1.ScaleHeight * Rnd)
 Let Y = Int(Picture1.ScaleWidth * Rnd)
 Picture1.Line (0,0) - (X,Y)
Next Lin1
End Sub
```

b) رسم الخطوط (النقطة الثانية عشوائية وألوان الخطوط عشوائية):

```
Private Sub CmdLin2_Click()
Picture1.DrawWidth = 3
For Lin2 = 1 To 100
 Let X = Int(Picture1.ScaleHeight * Rnd)
 Let Y = Int(Picture1.ScaleWidth * Rnd)
 Let RC = Int(256 * Rnd)
 Let GC = Int(256 * Rnd)
 Let BC = Int(256 * Rnd)
 Picture1.Line (0,0) - (X,Y),RGB(RC,GC,BC)
Next Lin2
End Sub
```

c) رسم الخطوط (النقطة الثانية عشوائية وألوان الخطوط عشوائية)- استخدام المؤقت:

```
Private Sub TimLin_Timer()
Picture1.DrawWidth = 3
For Lin2 = 1 To 100
 Let X = Int(Picture1.ScaleHeight * Rnd)
 Let Y = Int(Picture1.ScaleWidth * Rnd)
 Let RC = Int(256 * Rnd)
 Let GC = Int(256 * Rnd)
 Let BC = Int(256 * Rnd)
 Picture1.Line (0,0) - (X,Y),RGB(RC,GC,BC)
```

```
Next Lin2
End Sub
```

d) زر تنشيط وإيقاف المؤقت وتغيير خصائص زر الأوامر:

```
Private Sub CmdLin_Click()
If TimLin.Enabled = True Then
  CmdLin.Caption = "GO"
  TimLin.Enabled = False
Else
  TimLin.Enabled = True
  CmdLin.Caption = "STOP"
End If
End Sub
```

٣. رسم الدوائر عشوائية المركز وثابتة نصف القطر:

a) رسم الدوائر باللون الافتراضي (إحداثيات مركز الدائرة عشوائي):

```
Private Sub CmdCir1_Click()
For Cir1 = 1 To 100
 Let X = Int(Picture1.ScaleHeight * Rnd)
 Let Y = Int(Picture1.ScaleWidth * Rnd)
 Picture1.Circle (X,Y), 500
Next Cir1
End Sub
```

b) رسم الدوائر (إحداثيات مركز الدائرة وألوان الدوائر عشوائية):

```
Private Sub CmdCir2_Click()
For Cir2 = 1 To 100
 Let X = Int(Picture1.ScaleHeight * Rnd)
 Let Y = Int(Picture1.ScaleWidth * Rnd)
 Let RC = Int(256 * Rnd)
 Let GC = Int(256 * Rnd)
 Let BC = Int(256 * Rnd)
 Picture1.Circle (X,Y), 500, RGB(RC,GC,BC)
Next Cir2
End Sub
```

c) رسم الدوائر (إحداثيات مركز الدائرة وألوان الدوائر عشوائية) – استخدام المؤقت:

```
Private Sub TimCir12_Timer()
For Cir12 = 1 To 100
 Let X = Int(Picture1.ScaleHeight * Rnd)
 Let Y = Int(Picture1.ScaleWidth * Rnd)
```

```
 Let RC = Int(256 * Rnd)
 Let GC = Int(256 * Rnd)
 Let BC = Int(256 * Rnd)
 Picture1.Circle (X,Y),500,RGB(RC,GC,BC)
Next Cir12
End Sub
```

d) زر **تنشيط وإيقاف المؤقت وتغيير خصائص زر الأوامر:**

```
Private Sub CmdCir12_Click()
If TimCir12.Enabled = True Then
  CmdCir12.Caption = "GO"
  TimCir12.Enabled = False
Else
  TimCir12.Enabled = True
  CmdCir12.Caption = "STOP"
End If
End Sub
```

٤. رسم الدوائر ثابتة المركز والمتغيرة نصف القطر بشكل متدرج:

a) رسم الدوائر باللون الافتراضي (نصف قطر الدائرة متغير):

```
Private Sub CmdCir3_Click()
For Cir3 = 10 To 5000 Step 50
  Picture1.Circle (0,0),Cir3
Next Cir3
End Sub
```

b) رسم الدوائر (نصف قطر الدائرة متغير وألوان الدوائر عشوائية):

```
Private Sub CmdCir4_Click()
For Cir4 = 10 To 5000 Step 50
 Let RC = Int(256 * Rnd)
 Let GC = Int(256 * Rnd)
 Let BC = Int(256 * Rnd)
 Picture1.Circle (X,Y),Cir4,RGB(RC,GC,BC)
Next Cir4
End Sub
```

c) رسم الدوائر (نصف قطر الدائرة متغير وألوان الدوائر عشوائية) – استخدام المؤقت:

```
Private Sub TimCir34_Timer()
For Cir34 = 10 To 5000 Step 50
 Let RC = Int(256 * Rnd)
 Let GC = Int(256 * Rnd)
```

```
 Let BC = Int(256 * Rnd)
 Picture1.Circle (X,Y),Cir34,RGB(RC,GC,BC)
Next Cir34
End Sub
```

d) زر تنشيط وإيقاف المؤقت وتغيير خصائص زر الأوامر:

```
Private Sub CmdCir34_Click()
If TimCir34.Enabled = True Then
  CmdCir34.Caption = "GO"
  TimCir34.Enabled = False
Else
  TimCir34.Enabled = True
  CmdCir34.Caption = "STOP"
End If
End Sub
```

٥. تغيير مركز الإحداثيات:

```
Private Sub CmdScale_Click()
Picture1.Scale (-Picture1.ScaleWidth / 2,Picture1.ScaleHeight/2)
              - (Picture1.ScaleWidth / 2,-Picture1.ScaleHeight / 2)
End Sub
```

٦. مسح محتويات مربع الصورة:

```
Private Sub CmdCls_Click()
Picture1.Cls
End Sub
```

٧. إنهاء البرنامج:

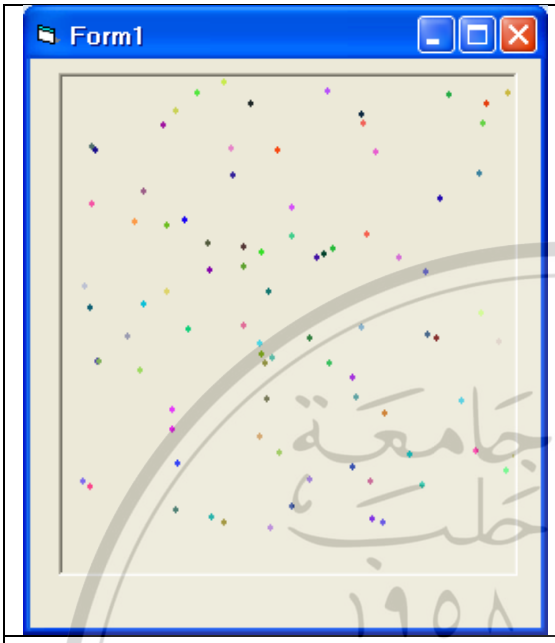
```
Private Sub CmdExit_Click()
End
End Sub
```

٨. إمكانية استخدام هذا الأمر لجعل نصف قطر الدوائر عشوائي:

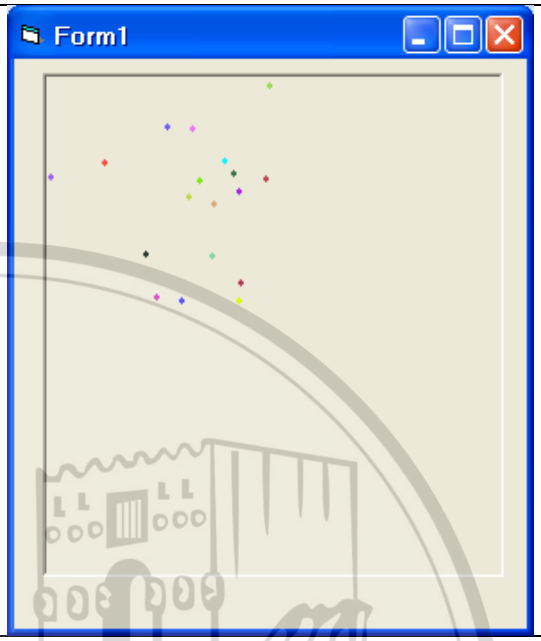
```
Let Radius = Int(Picture1.ScaleWidth * Rnd)
```

المرحلة التنفيذية:

١) رسم النقاط:

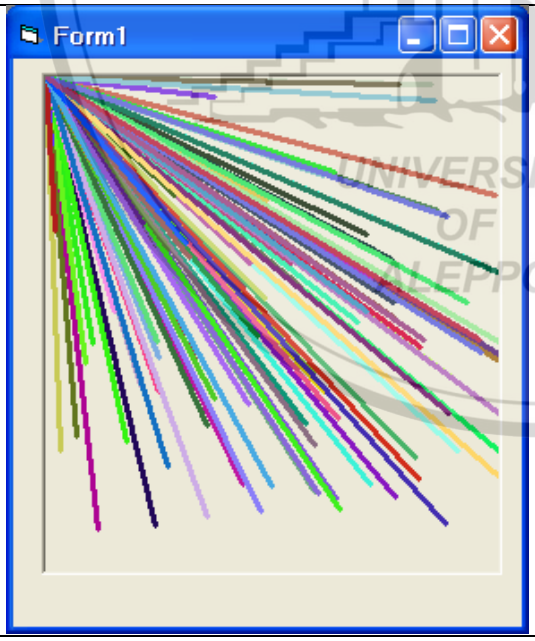


بعد نقل مركز الإحداثيات إلى منتصف أداة الرسم

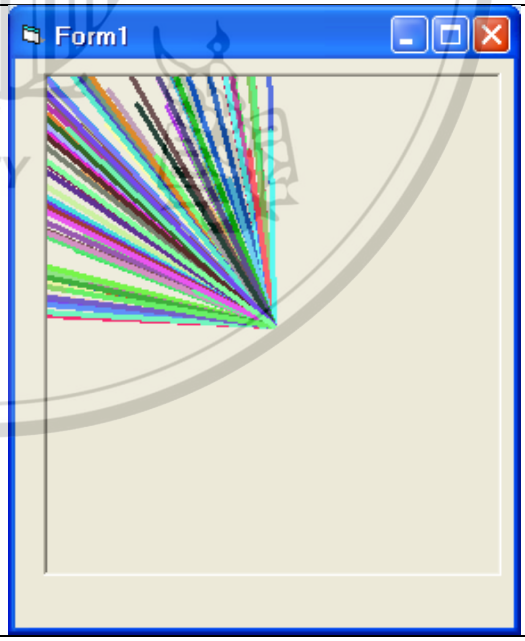


مركز الإحداثيات هو نقطة الأصل

٢) رسم الخطوط:



بعد نقل مركز الإحداثيات إلى منتصف أداة الرسم



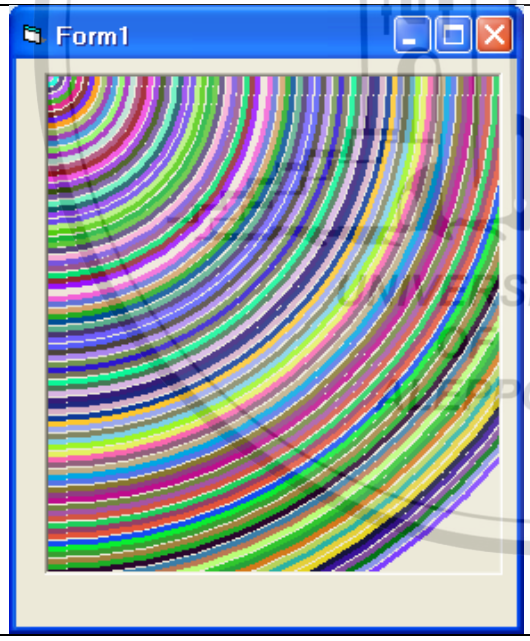
مركز الإحداثيات هو نقطة الأصل

٣) رسم الدوائر عشوائية المركز:

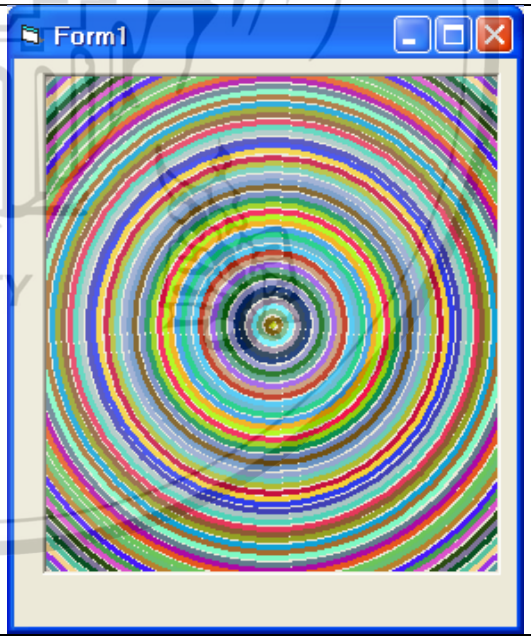
مركز الإحداثيات هو نقطة الأصل

بعد نقل مركز الإحداثيات إلى منتصف أداة الرسم

٤) رسم الدوائر المتحدة المركز:



مركز الإحداثيات هو نقطة الأصل



بعد نقل مركز الإحداثيات إلى منتصف أداة الرسم

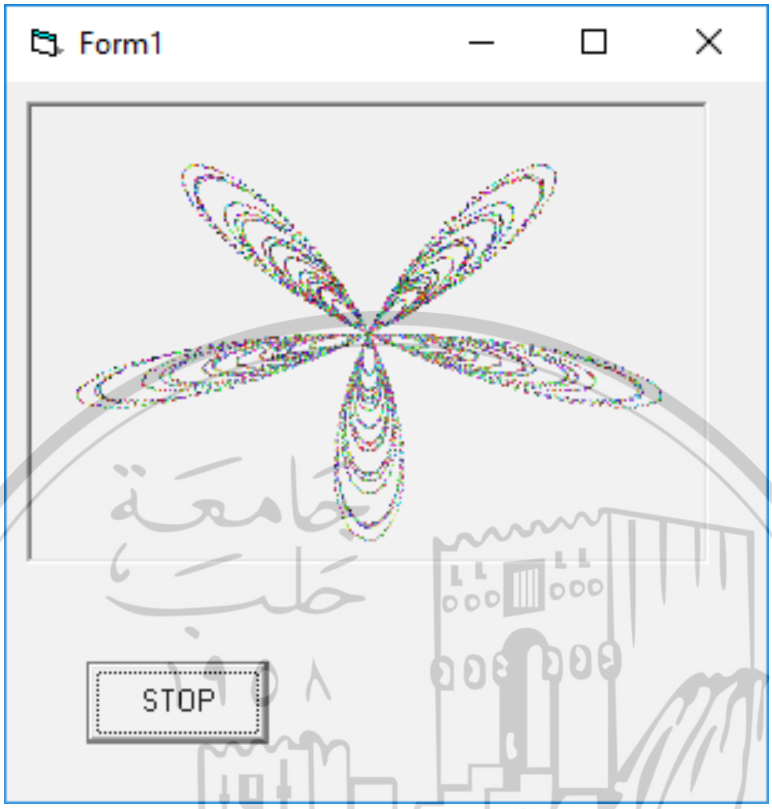
تطوير تمرين – طرائق الرسم $Drawing\ Methods - Scale - Timer$:

- **صمم الواجهة المرئية التي تحتوي زر أوامر ومؤقت:**
- **اســتخدام زر أوامر لرســم مجموعة من النقاط العشــوائية الألوان والتي إحداثياتها (X, Y) والتي يمثّلها التابع الرياضي:**

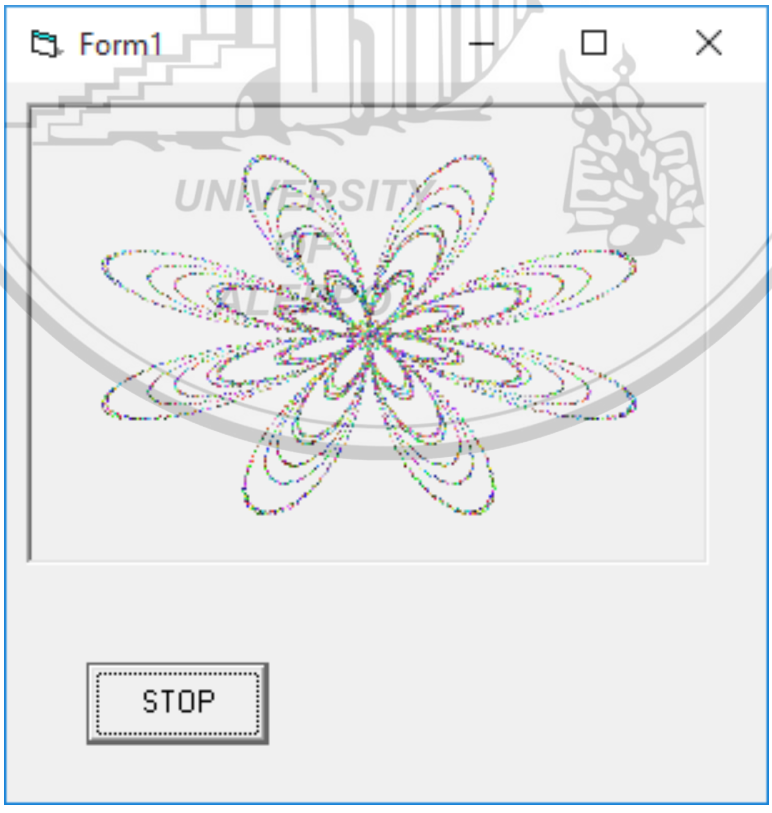
$$\theta_{tot} = 2\pi$$
$$a = 500 * Rnd$$
$$r = a * Sin(2\theta)$$
$$x = r * Cos(\theta)$$
$$y = r * Sin(\theta)$$

```
Private Sub Stars_Click( )
If Timer1.Enabled = True Then
  Stars.Caption = "GO"
  Timer1.Enabled = False
Else
  Timer1.Enabled = True
  Stars.Caption = "STOP"
End If
End Sub

Private Sub Timer1_Timer( )
Picture1.Scale (500, −500) − (−500, 500)
totrad = 8 * Atn(1)
a = 500 * Rnd()
For teta = 0 To totrad Step 0.01
  r = a * Sin(4 * teta)
  X = r * Cos(teta)
  Y = r * Sin(teta)
  re = Rnd * 255
  ge = Rnd * 255
  be = Rnd * 255
  Picture1.PSet (X,Y), RGB(re, ge, be)
Next
End Sub
```

عند جعل $a = 5$ أي فردي نجد أن عدد الوريقات يساوي $a = 5$:



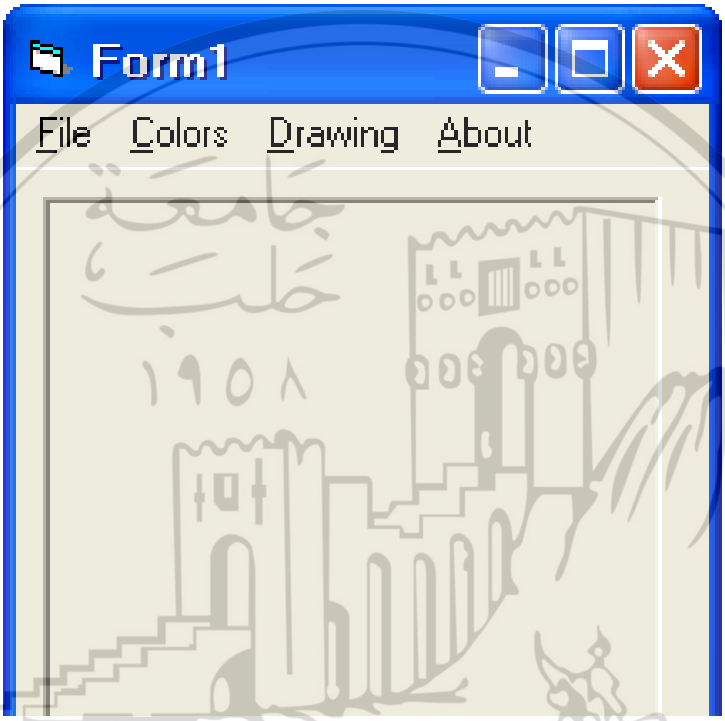
عند جعل $a = 4$ أي فردي نجد أن عدد الوريقات يساوي $a = 8 * 2$:



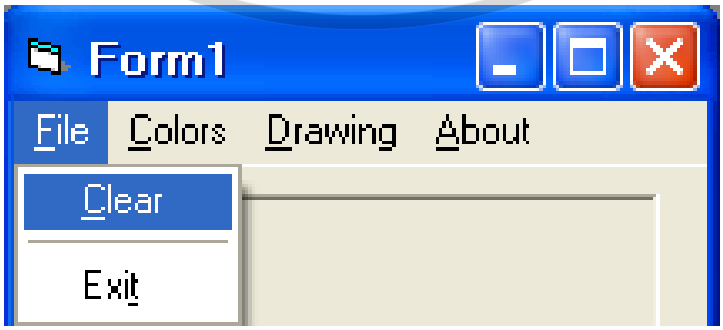
التمرين (٤) – محرر القوائم والرسم *Menu Editor and Drawing*:

- **المطلوب تصـميم الواجهة المرئية والمؤلفة من شـريط القوائم ومربع الصـورة كما هو مبيّن.**
- **لاحظ المفاتيح الساخنة Hot Keys في القوائم المبيَّنة:**

File **C**olors **D**rawing **A**bout



- **القائمة File:**
  - تتألف من الأمر Clear والذي يستخدم لتنظيف مربع الصورة والأمر Exit والذي يستخدم لإنهاء البرنامج.
  - (لاحظ المفاتيح الساخنة C, t)



- **القائمة Colors:**

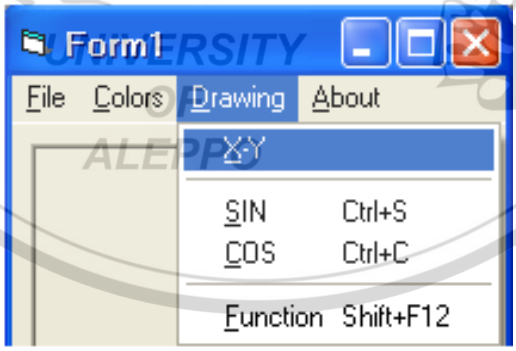
- تتألف من الأمر *Select Colors* والذي تتفرع منه ثلاثة أوامر تستخدم للتبديل بين الألوان وتحدد الأمر المستخدم وتجعله غير قابلاً للاستخدام حاليّاً.
- لاحظ المفاتيح الساخنة ومفاتيح الاختصار.

- **القائمة Drawing:**

تتألف من الأوامر التالية:

- X–Y– لنقل مركز الإحداثيات ورسم المحاور الإحداثية الجديدة.
- SIN – لرسم تابع الجيب بلون أخضر.
- COS – لرسم تابع التجيب بلون أحمر.
- Function – لرسم التابع الذي تريده بلون مختلف.

لاحظ المفاتيح الساخنة ومفاتيح الاختصار.



- **القائمة About:**

- تظهر رسالة تبيّن اسم الطالب الذي قام بتنفيذ التمرين.

ملاحظة مهمة للطلاب:

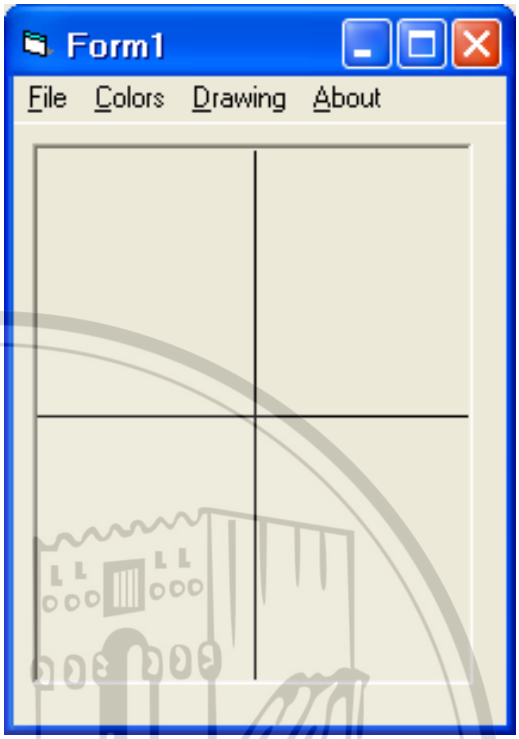
- **لا يُسمح بإدخال الحل إلى مخبر البرمجة ويسمح بإدخال نص التمرين فقط.**

المرحلة الكودية:

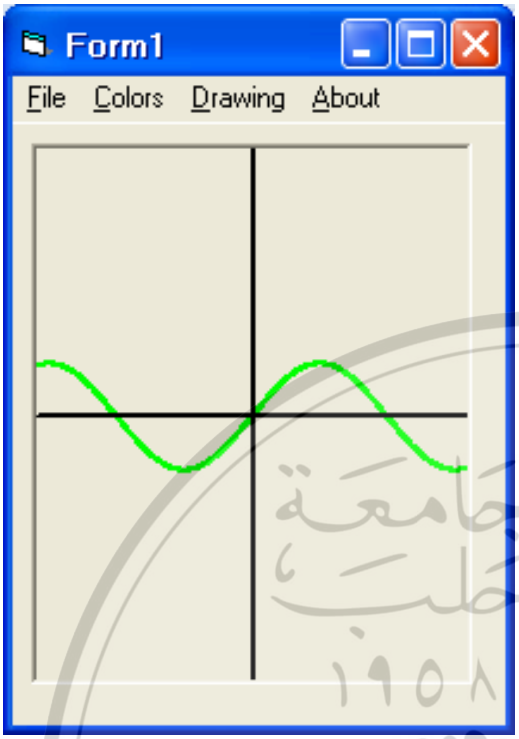
```
Private Sub mnuClear_Click()
Picture1.Cls
End Sub

Private Sub MnuExit_Click()
End
End Sub

Private Sub mnuXY_Click()
Picture1.Scale (-100,100)-(100,-100)
Picture1.Line (-100,0)-(100,0)
Picture1.Line (0,-100)-(0,100)
End Sub

Private Sub mnuSin_Click()
Picture1.Scale (-5,5)-(5,-5)
pi = 4 * Atn(1)
For teta = -8 * pi To 8 * pi Step pi / 180
y = Sin(teta)
Picture1.DrawWidth = 2
Picture1.PSet (teta,y),vbGreen
Next
End Sub

Private Sub mnuCos_Click()
Picture1.Scale (-5,5)-(5,-5)
pi = 4 * Atn(1)
For teta = -8 * pi To 8 * pi Step pi / 180
y = Cos(teta)
Picture1.DrawWidth = 2
Picture1.PSet (teta,y),vbRed
Next
End Sub

Private Sub mnuFunction_Click()
Picture1.Scale (-10,10)-(10,-10)
pi = 4 * Atn(1)
For x = -10 To 10 Step 0.0001
y = 1 / 10 * (Exp(x) + Exp(-x))
```

```
Picture1.DrawWidth = 2
Picture1.PSet (x, y), vbRed
Next
End Sub

Private Sub MnuRed_Click()
Form1.BackColor = vbRed
mnuGreen.Enabled = True
mnuBlue.Enabled = True
MnuRed.Enabled = False
MnuRed.Checked = True
mnuBlue.Checked = False
mnuGreen.Checked = False
End Sub

Private Sub MnuGreen_Click()
Form1.BackColor = vbGreen
mnuGreen.Enabled = False
mnuGreen.Checked = True
mnuBlue.Checked = False
MnuRed.Checked = False
mnuBlue.Enabled = True
MnuRed.Enabled = True
End Sub

Private Sub MnuBlue_Click()
Form1.BackColor = vbBlue
mnuGreen.Enabled = True
mnuBlue.Enabled = False
MnuRed.Enabled = True
mnuBlue.Checked = True
MnuRed.Checked = False
mnuGreen.Checked = False
End Sub

Private Sub MnuAbout_Click()
mnuColors.Visible = False
MsgBox ("This Program by Designed Dr Hammad")
End Sub
```

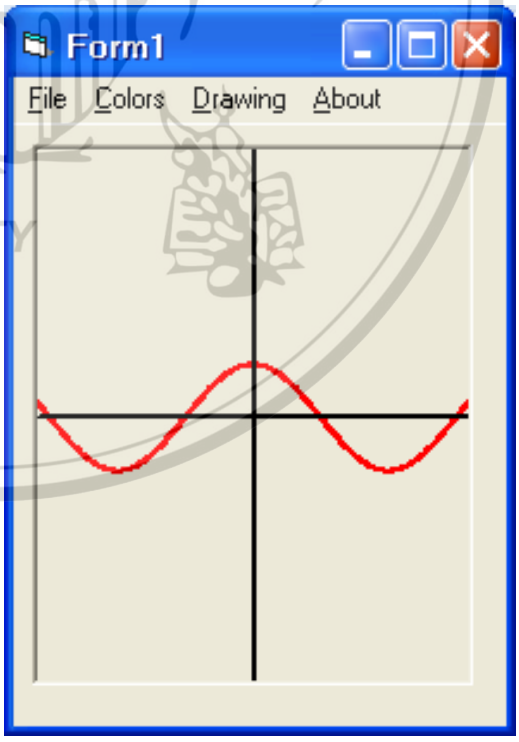
المرحلة التنفيذية:



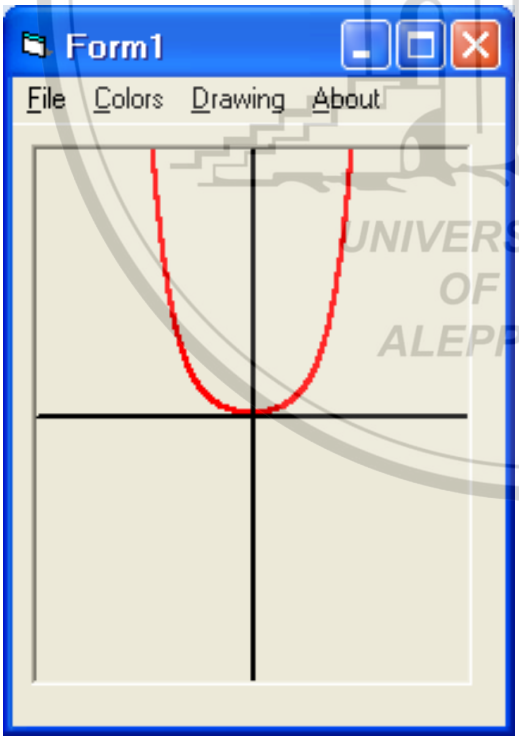
نقل مركز الإحداثيات رسم المحاور الإحداثية



رسم تابع الجيب

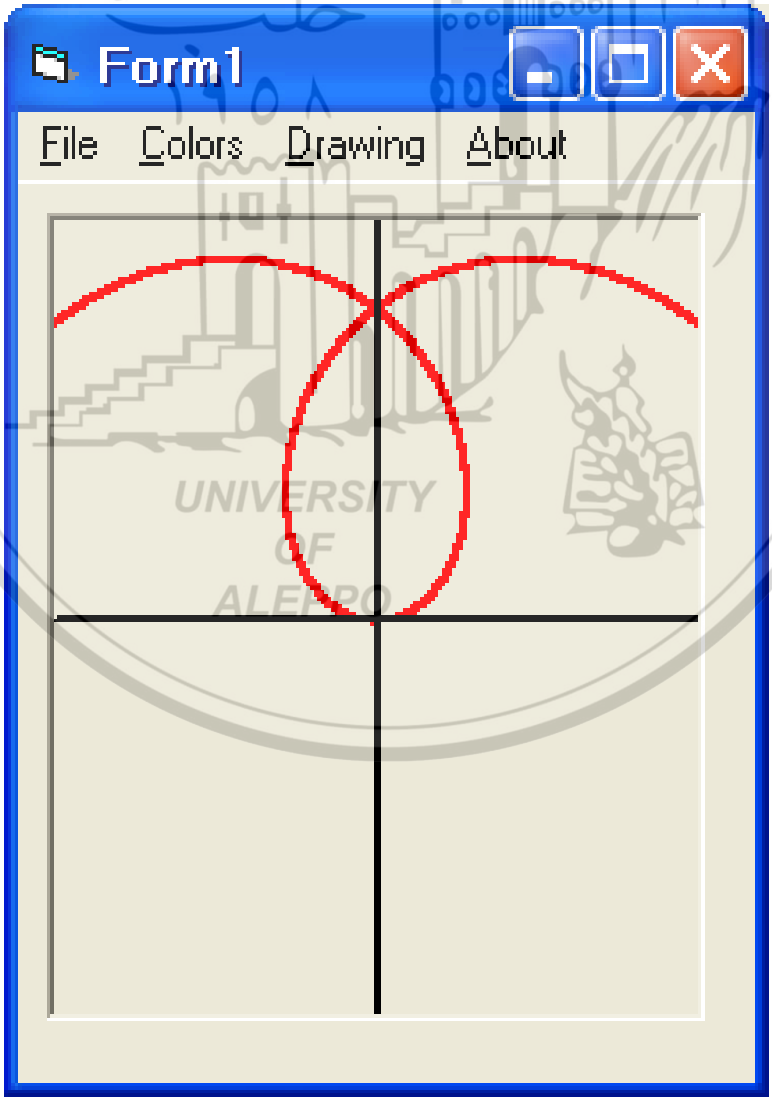


رسم تابع التجيب

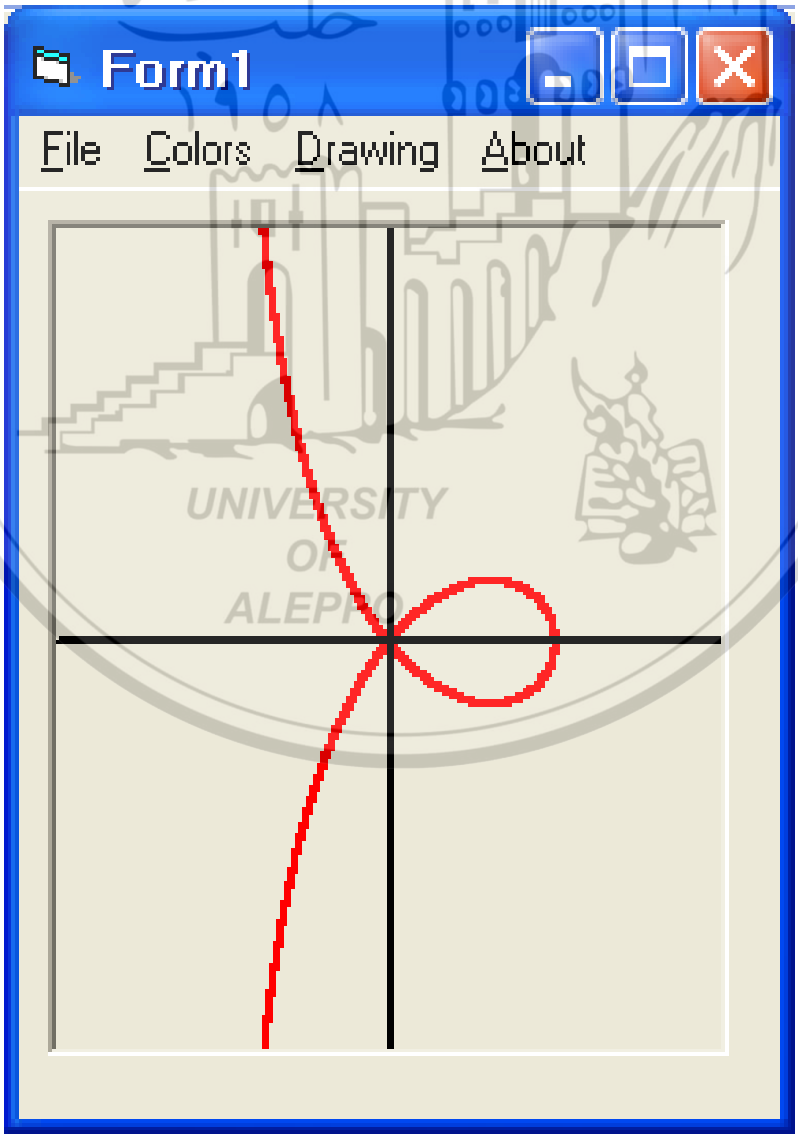


رسم تابع مختلف (هنا قطع مكافئ)

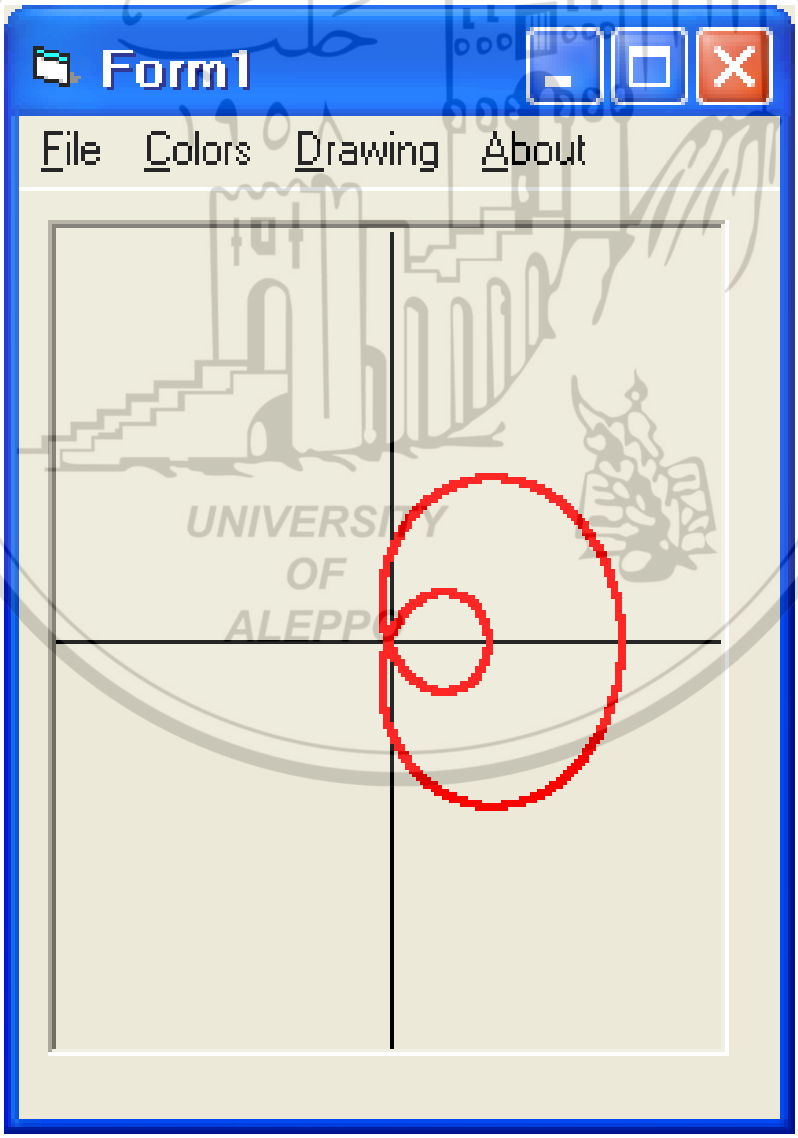
```
Private Sub mnuFunction2_Click( )
Picture1.Scale (−10, 10) − (10, −10)
pi = 4 * Atn(1)
For teta = −8 * pi To 8 * pi Step 0.0001
r = 5 * teta
x = r * Cos(teta)
y = r * Sin(teta)
Picture1.DrawWidth = 2
Picture1.PSet (x, y), vbRed
Next
End Sub
```



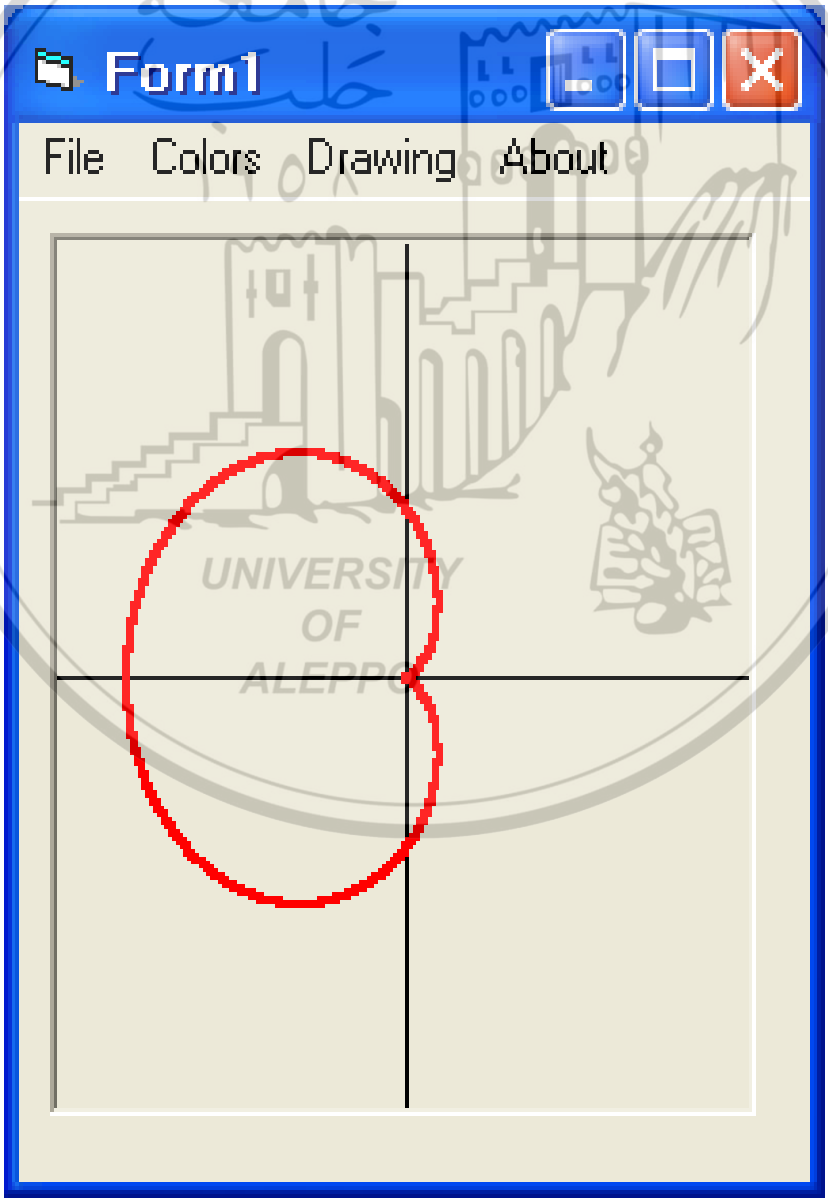
```
Private Sub mnuFunction3_Click()
Picture1.Scale (-10, 10) - (10, -10)
pi = 4 * Atn(1)
For teta = 0 To 8 * pi Step 0.0001
r = 5 * Cos(2 * teta) / Cos(teta)
x = r * Cos(teta)
y = r * Sin(teta)
Picture1.DrawWidth = 2
Picture1.PSet (x, y), vbRed
Next
End Sub
```



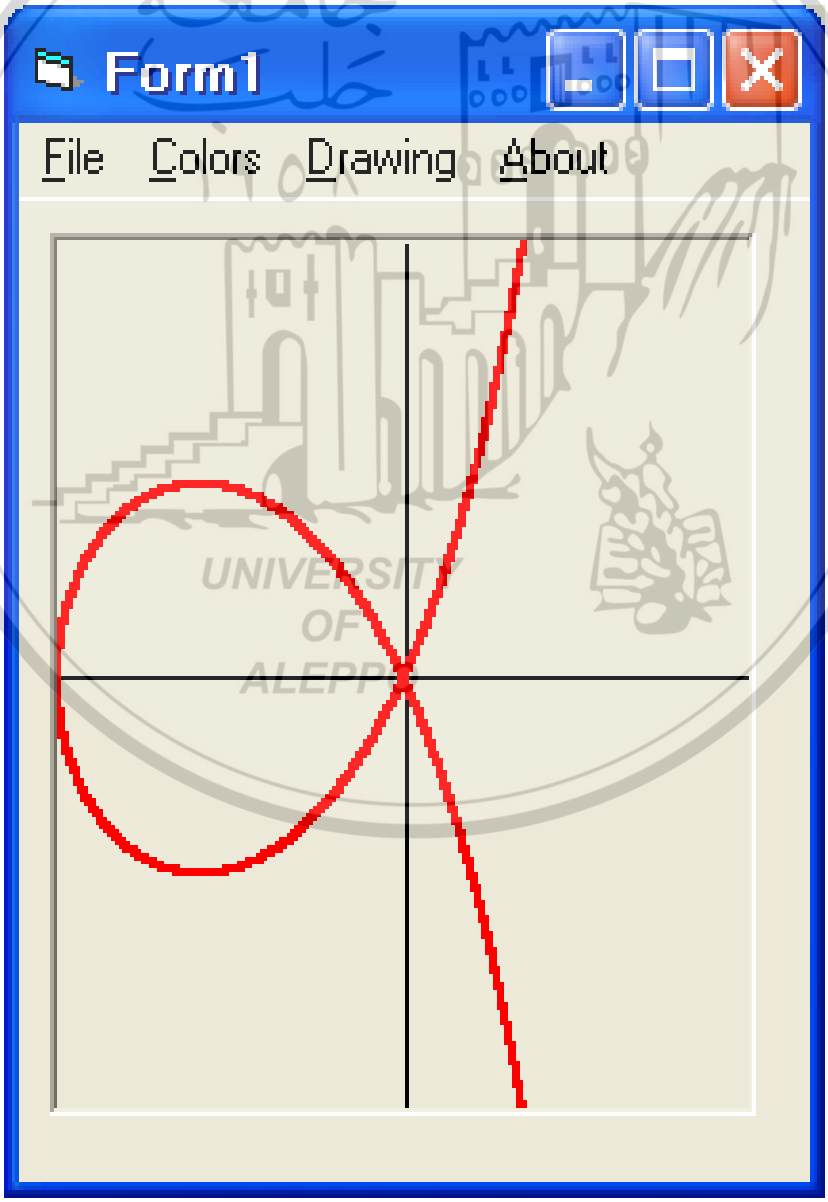
```
mnuFunction4_Click()
Picture1.Scale (-10,10)-(10,-10)
pi = 4 * Atn(1)
For teta = 0 To 8 * pi Step 0.0001
r = 5 * Cos(teta) + 2
'r = 5 * Cos(teta) - 2
x = r * Cos(teta)
y = r * Sin(teta)
Picture1.DrawWidth = 2
Picture1.PSet (x,y),vbRed
Next
End Sub
```



```
Private Sub mnuFunction5_Click()
Picture1.Scale (-10, 10) - (10, -10)
pi = 4 * Atn(1)
For teta = -8 * pi To 8 * pi Step 0.0001
r = 2 * 2 * (1 - Cos(teta))
x = r * Cos(teta)
y = r * Sin(teta)
Picture1.DrawWidth = 2
Picture1.PSet (x, y), vbRed
Next
End Sub
```



```
Private Sub mnuFunction6_Click()
Picture1.Scale (-10, 10) - (10, -10)
pi = 4 * Atn(1)
For teta = -8 * pi To 8 * pi Step 0.001
r = (10 / Cos(teta)) + 20
x = r * Cos(teta)
y = r * Sin(teta)
Picture1.DrawWidth = 2
Picture1.PSet (x, y), vbRed
Next
End Sub
```

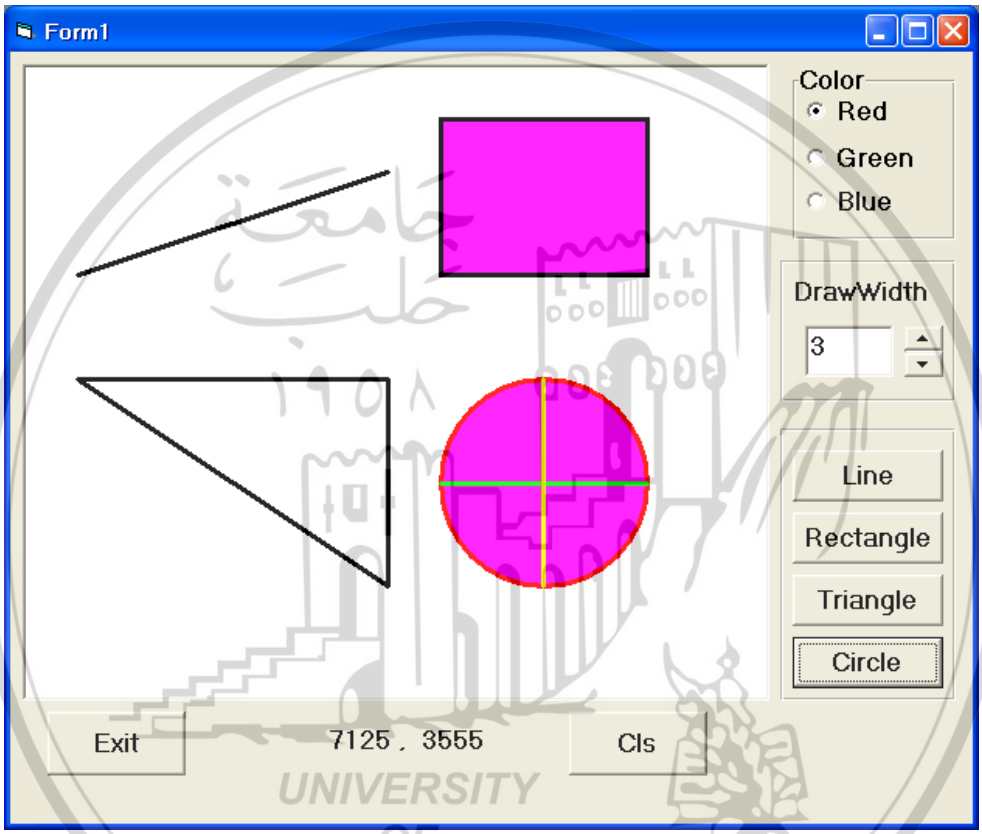


التمرين (٥) – أحداث الماوس *Mouse Events*:

المطلوب رسم مربع صورة PBox كسبورة ويتم على مرحلتين تنفيذ ما يلي:

١. المرحلة الأولى والتي لا بد من تنفيذ كل الأوامر فيها وتشمل:

- تصميم إطار فيه ثلاث ألوان للرسم (قابلة للتطوير إلى عدد أكبر وبطرق مختلفة كأن نقوم بإضافة اللون عن طريق أشرطة السحابات.



- يتم انتقاء اللون من خلال أزرار الخيارات OptRed, OptGreen, OptBlue.
- سحاب ومربع نص يسمح بجعل عرض خط الرسم في المجال 5 ... 1.
- لافتة Label X, Y نستطيع خلالها إظهار إحداثيات الماوس أثناء حركتها فوق مربع الصورة فقط.
- باستخدام أوامر الماوس (الأيمن – الأيسر – ضغط – تحرير) رسم نقاط وخطوط مستمرة أو مربعات مصمتة.
- زر أوامر يقوم بتنظيف الشاشة.
- زر أوامر يقوم بإنهاء البرنامج.

٢. المرحلة التطويرية والتي تعتبر مرحلة إضافية وتشمل:

- إضافة إطار يحتوي على أربع أزرار أوامر مسئولة عن رسم خط ومستطيل ومثلث ودائرة.
- الخط بين نقطتين. أو تكون هاتين النقطتين قطر لمستطيل.
- مثلث تعطى فيه أبعاد النقطة الأولى وبعد الثانية عنها والثالثة عن الثانية.

ملاحظة: يمكن قبول أي تعديلات أخرى يقوم بها الطالب.

المرحلة الكودية – المرحلة الأولى:

```
Option Explicit
Dim MousePress As Boolean
Dim x1, y1, x2, y2, x3, y3
Private Sub Form_Load()
Pbox.BackColor = vbBlack
Pbox.ForeColor = vbWhite
Pbox.DrawWidth = 1
twidth.Text = 1
End Sub
Private Sub cls_Click()
Pbox.Cls
End Sub
Private Sub cmdExit_Click()
End
End Sub
Private Sub VSwidth_Change()
Pbox.DrawWidth = vswidth.Value
twidth.Text = vswidth.Value
End Sub
Private Sub OptRed_Click()
OptRed.Value = True
Pbox.ForeColor = vbRed
End Sub
Private Sub OptGreen_Click()
OptGreen.Value = True
Pbox.ForeColor = vbGreen
End Sub
```

```
Private Sub OptBlue_Click()
OptBlue.Value = True
Pbox.ForeColor = vbBlue
End Sub
Private Sub pbox_MouseDown(Button As Integer,
                           Shift As Integer, x As Single, y As Single)
If Button = vbLeftButton Then
 MousePress = True
 Pbox.PSet (x, y)
ElseIf Button = vbRightButton Then
 Pbox.PSet (x, y)
End If
End Sub
Private Sub pbox_MouseMove(Button As Integer,
                           Shift As Integer, x As Single, y As Single)
If MousePress = True Then
 Pbox.Line -(x, y)
End If
lbl.Caption = Str(x) + "," + Str(y)
End Sub
Private Sub pbox_MouseUp(Button As Integer,
                           Shift As Integer, x As Single, y As Single)
If Button = vbLeftButton Then
 MousePress = False
ElseIf Button = vbRightButton Then
 Pbox.Line -(x, y),,B
End If
End Sub
```

ملاحظة: يمكن قبول أي تعديلات أخرى يقوم بها الطالب.

المرحلة الكودية – المرحلة التطويرية:

- إضافة إطار يحتوي على أربع أزرار أوامر مسؤولة عن رسم خط ومستطيل ومثلث ودائرة.
- الخط بين نقطتين. أو تكون هاتين النقطتين قطر لمستطيل.
- مثلث تعطى فيه أبعاد النقطة الأولى وبعد الثانية عنها والثالثة عن الثانية.

- دائرة ندخل إحداثيات مركزها ونصف قطرها ونرسم بداخلها خطوط متعامدة.

```
Private Sub Line_Click()
MsgBox "insert The Coordenat of The First point Of The Line"
x1 = Val(InputBox("Insert The X1"))
y1 = Val(InputBox("Insert The Y1"))
MsgBox "insert The Coordenat of The Second point Of The Line"
x2 = Val(InputBox("Insert The X2"))
y2 = Val(InputBox("Insert The Y2"))
Pbox.Line (x1,y1)-(x2,y2)
End Sub

Private Sub Rectangle_Click()
MsgBox "insert The Coordenat of The First point Of The Rectangle"
x1 = Val(InputBox("Insert The X1"))
y1 = Val(InputBox("Insert The Y1"))
MsgBox "insert The Coordenat of The Second point Of The Rectangle"
x2 = Val(InputBox("Insert The X2"))
y2 = Val(InputBox("Insert The Y2"))
Pbox.Line (x1,y1)-(x2,y2),,B
End Sub

Private Sub Triangle_Click()
MsgBox "insert The Coordenat of The First point Of The Triangle"
x1 = Val(InputBox("Insert The X1"))
y1 = Val(InputBox("Insert The Y1"))
MsgBox "insert The Distance of The Second point Of The Triangle"
x2 = Val(InputBox("Insert The X2"))
y2 = Val(InputBox("Insert The Y2"))
Pbox.Line (x1,y1)-Step(x2,y2)
MsgBox "insert The Distance of The Third point Of The Triangle"
x3 = Val(InputBox("Insert The X3"))
y3 = Val(InputBox("Insert The Y3"))
Pbox.Line -Step(x3,y3)
Pbox.Line -(x1,y1)
End Sub

Private Sub Circle_Click()
Dim x,y,r
x = Val(InputBox(" Insert x"))
y = Val(InputBox(" Insert y "))
r = Val(InputBox(" Insert r "))
```

$Const\ pi\ =\ 3.14159265$
$Pbox.Circle\ (x,y),r,vbRed$
$Pbox.Line\ (x\ -\ r,y)-(x\ +\ r,y),vbGreen$
$Pbox.Line\ (x,y\ -\ r)-(x,y\ +\ r),vbYellow$
End Sub

المرحلة التنفيذية:

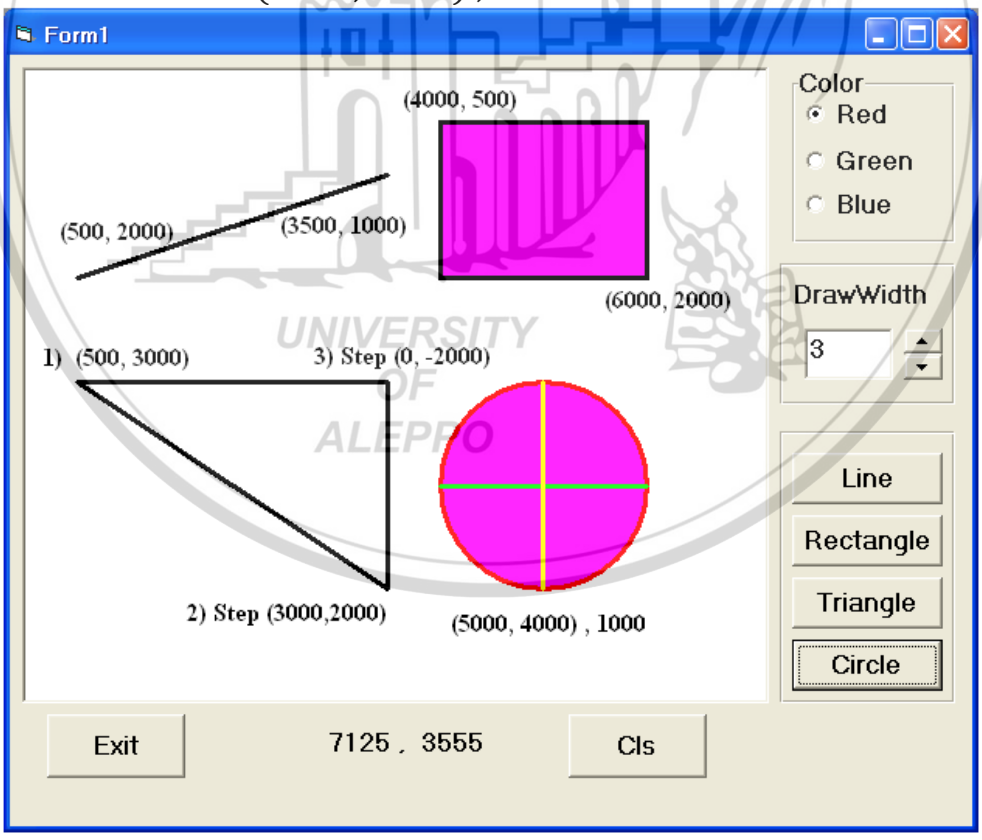
الآن بإدخال القيم الرقمية التالية نحصل على:

$The\ Line:\quad Line\ (500,2000)-(3500,1000)$

$The\ Rectangle:\quad Line\ (4000,500)-(6000,2000),,B$

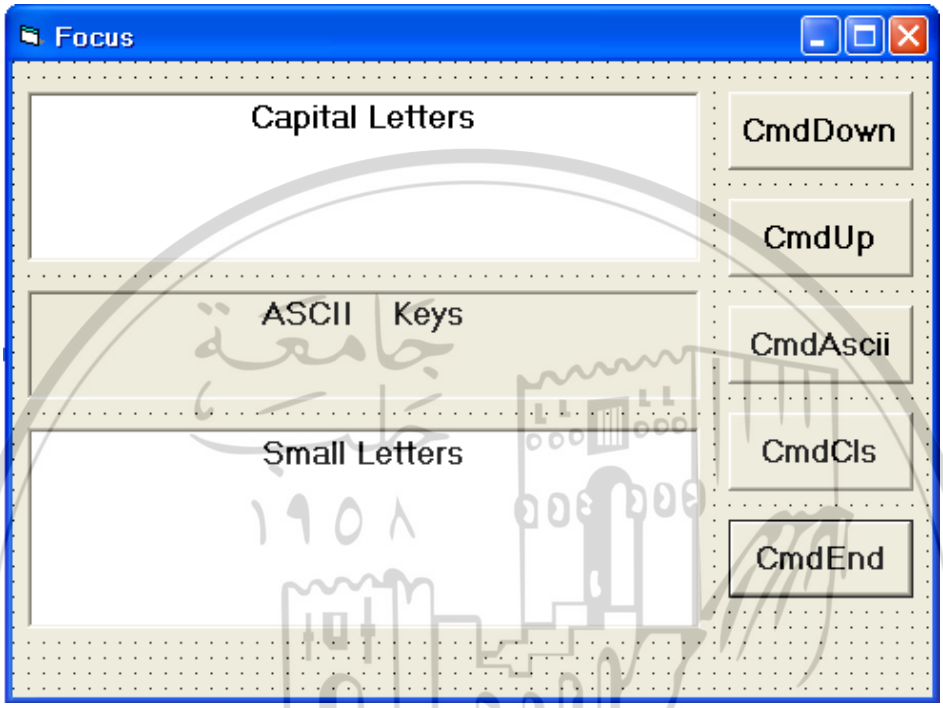
$The\ Triangle:$
$The\ Line\ 1-2:\quad Line\ (500,3000)-Step(3000,2000)$
$The\ Line\ 2-3:\quad Line-Step\ (0,-2000)$
$The\ Lin\ 3-1\quad Line-(500,3000)$

$The\ Circle:\quad Circle\ (5000,4000)\ ,1000$



التمرين (٦) – أحداث لوحة المفاتيح *Keyboard Events*:

المطلوب تصـــميم برنامج يظهر أحداث لوحة المفاتيح وهي الضــغط واســتمرار الضغط وتحرير المفاتيح وأحداث التركيز



١) عند إقلاع البرنامج وتحميل الفورم سيتم مسح الكتابات في مربعات النصوص وتلوين خلفية الفورم باللون السماوي.

٢) تغيير الخاصية *KeyPreview* بحيث نرى تأثير التراكيز على الأدوات الأخرى مباشرة أو بعد تأثيرات الفورم.

٣) رؤية الأحداث عن وصول التركيز أو مغادرته لبعض الأدوات. استخدم مربعات النصوص لإظهار العبارات التي تدل على وصول التركيز للأدوات أو مغادرته لها.

٤) استخدم أزرار الأوامر المبينة لإظهار الأحداث عن ضغط واستمرار الضغط وتحرير أزرار لوحة المفاتيح (لاحظ الفرق بين أو *KeyDown* و *KeyPress*.

٥) استخدم الأمر *Change* على أداة النص للحروف الكبيرة.

٦) استخدم التوابع التي تبين الحروف الكبيرة والصغيرة.

٧) بيّن الخصائص *Cancel, Default*.

٨) بين الخاصية *TabIndex* وعملية انتقال التركيز بين الأدوات.

المرحلة الكودية:

- سيحدث هذا الأمر عند إقلاع البرنامج وتحميل الفورم.

```
Private Sub Form_Load( )
Form1.BackColor = vbCyan
Text1.Text = ""
Text2.Text = ""
Lb1.Caption = ""
End Sub
```

سيظهر تأثير هذا الأمر عندما تصبح الخاصية $Key\ Preview = True$.

- سيحدث هذا الأمر بدء الضغط على مفتاح ما (تركيز الفورم أقوى من أي أداة).

```
Private Sub Form_KeyDown
                    (KeyCode As Integer, Shift As Integer)
Form1.BackColor = vbRed
Text1.Text = " The KeyPreviw of Form Is True Now "
End Sub
```

سيظهر تأثير هذا الأمر عندما تصبح الخاصية $Key\ Preview = True$.

- سيحدث هذا الأمر بدء الضغط على مفتاح ما (تركيز الفورم أقوى من أي أداة).

```
Private Sub Form_KeyUp
                    (KeyCode As Integer, Shift As Integer)
Form1.BackColor = vbCyan
Text1.Text = " The KeyPreviw of Form Is True Now "
End Sub
```

سيظهر تأثير هذا الأمر عندما تصبح الخاصية $Key\ Preview = True$.

- سيحدث هذا الأمر بدء الضغط على مفتاح ما (تركيز الفورم أقوى من أي أداة).
- وبعد إزالة إشارة الملاحظة للتفريق بين عملية استمرار الضغط والضغط.

```
'Private Sub Form_KeyPress(KeyAscii As Integer)
'Form1.BackColor = vbGreen
'Text1.Text = " The KeyPreviw of Form Is True Now "
'End Sub
```

- سيحدث هذا الأمر وصول التركيز للأداة.

```
Private Sub Text1_GotFocus( )
Lb1.Caption = " Your Focus is on the Capital Letters text Box "
End Sub
```

- سيحدث هذا الأمر وصول التركيز للأداة.

```
Private Sub Text2_GotFocus( )
Lb1. Caption = " Your Focus is on the Small Letters text Box "
End Sub
```

- سيحدث هذا الأمر وصول التركيز للأداة.

```
Private Sub CmdDown_GotFocus( )
Lb1. Caption = " سيظهر التأثير عند بداية الضغط على المفتاح للأسفل "
Text1. Text = " The KeyPreviw of Form Is False Now "
End Sub
```

- سيحدث هذا الأمر وصول التركيز للأداة.

```
Private Sub CmdUp_GotFocus( )
Lb1. Caption = "سيظهر التأثير عند تحرير المفتاح"
Text1. Text = " The KeyPreviw of Form Is False Now "
End Sub
```

- سيحدث هذا الأمر وصول التركيز للأداة.

```
Private Sub CmdASCII_GotFocus( )
Lb1.Caption = " سيظهر التأثير عند بداية الضغط على المفتاح للأسفل"
End Sub
```

سيظهر تأثير هذا الأمر عندما تصبح الخاصية $Key\ Preview = False$.

- سيحدث هذا الأمر بدء الضغط على مفتاح ما ووجود التركيز على هذه الأداة.

(أصبح تركيز الفورم معدوماً).

```
Private Sub CmdDown_KeyDown
                    (KeyCode As Integer, Shift As Integer)
Lb1. Caption = " "
Lb1. Caption = "KeyCode" + Str(KeyCode) + vbCrLf + " Shift"
          + Str(Shift)
End Sub
```

سيظهر تأثير هذا الأمر عندما تصبح الخاصية $Key\ Preview = False$.

- سيحدث هذا الأمر بدء الضغط على مفتاح ما ووجود التركيز على هذه الأداة.

(أصبح تركيز الفورم معدوماً).

```
Private Sub CmdUp_KeyUp
```

```
                                    (KeyCode As Integer, Shift As Integer)
Lb1.Caption = " "
Lb1.Caption = "KeyCode" + Str(KeyCode) + vbCrLf + " Shift"
            + Str(Shift)
End Sub
```

سيظهر تأثير هذا الأمر عندما تصبح الخاصية $Key\ Preview = False$.

- سيحدث هذا الأمر بدء الضغط على مفتاح ما ووجود التركيز على هذه الأداة.

(أصبح تركيز الفورم معدوماً).

```
Private Sub CmdAscii_KeyPress(KeyAscii As Integer)
Dim Char
Lb1.Caption = " "
Char = Chr(KeyAscii)
Lb1.Caption = "KeyAscii" + Str(KeyAscii) + vbCrLf + " Char "
            + Char
End Sub
```

سيظهر تأثير هذا الأمر عندما تصبح الخاصية $Key\ Preview = False$.

- سيحدث هذا الأمر بدء الضغط على مفتاح ما ووجود التركيز على هذه الأداة.

(أصبح تركيز الفورم معدوماً).

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
Dim Char
Lb1.Caption = " You Write in Capital Letter "
Char = Chr(KeyAscii)
KeyAscii = Asc(UCase(Char))
    Lb1.Caption = Lb1.Caption + vbCrLf + " KeyAscii "
                + Str(KeyAscii) + " Char = " + Char
End Sub
```

سيظهر تأثير هذا الأمر عندما تصبح الخاصية $Key\ Preview = False$.

- سيحدث هذا الأمر بدء الضغط على مفتاح ما ووجود التركيز على هذه الأداة.

(أصبح تركيز الفورم معدوماً).

```
Private Sub Text2_KeyPress(KeyAscii As Integer)
Dim Char
Lb1.Caption = " You Write in Small Letter "
Char = Chr(KeyAscii)
KeyAscii = Asc(LCase(Char))
```

```
Lb1.Caption = Lb1.Caption + vbCrLf + " KeyAscii"
              + Str(KeyAscii) + " Char = " + Char
End Sub
```

- سيحدث هذا الأمر بدء الضغط الكتابة في الأداة $Text2$ أي عند بدء حدوث الحدث $Change$ ووجود التركيز على هذه الأداة.

```
Private Sub Text2_Change( )
If Text2.Text = "h" Then
Text1.BackColor = vbRed
ElseIf Text2.Text = "he" Then
Text1.BackColor = vbGreen
ElseIf Text2.Text = "hel" Then
Text1.BackColor = vbBlue
ElseIf Text2.Text = "hell" Then
Text1.BackColor = vbMagenta
ElseIf Text2.Text = "hello" Then
Text1.BackColor = vbYellow
Else
Text1.BackColor = vbWhite
End If
End Sub
```

- سيحدث هذا الأمر بدء الضغط على هذا المفتاح.

```
Private Sub CmdCls_Click( )
Text1.Text = ""
Text2.Text = ""
Lb1.Caption = ""
End Sub
```

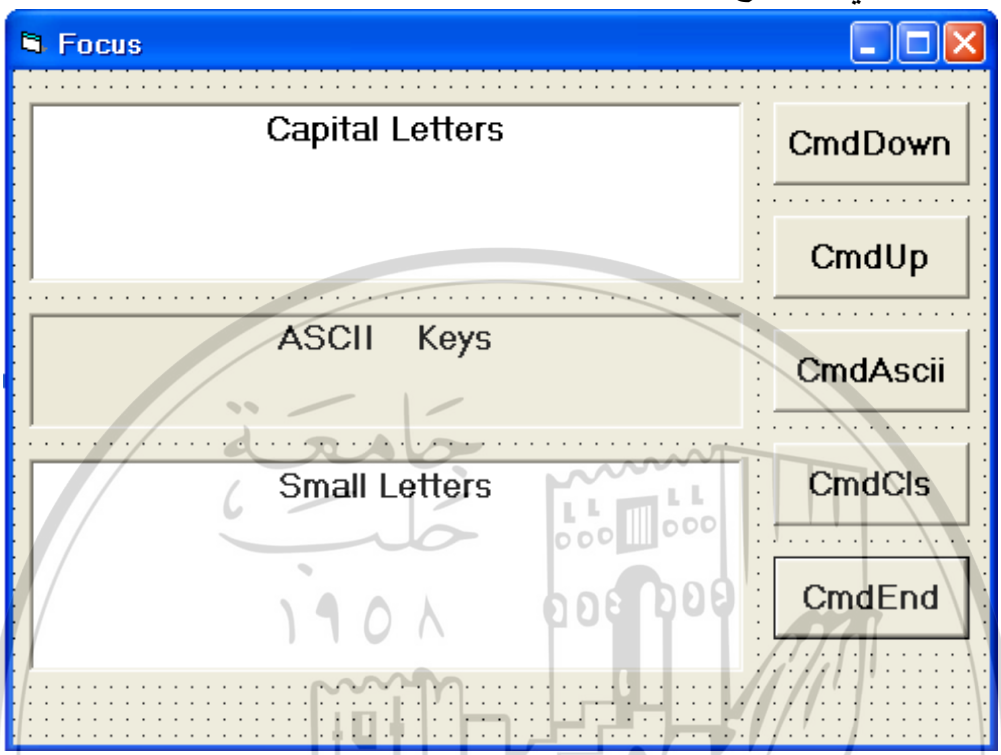
- سيحدث هذا الأمر بدء الضغط على هذا المفتاح.

```
Private Sub CmdEnd_Click( )
End
End Sub
```

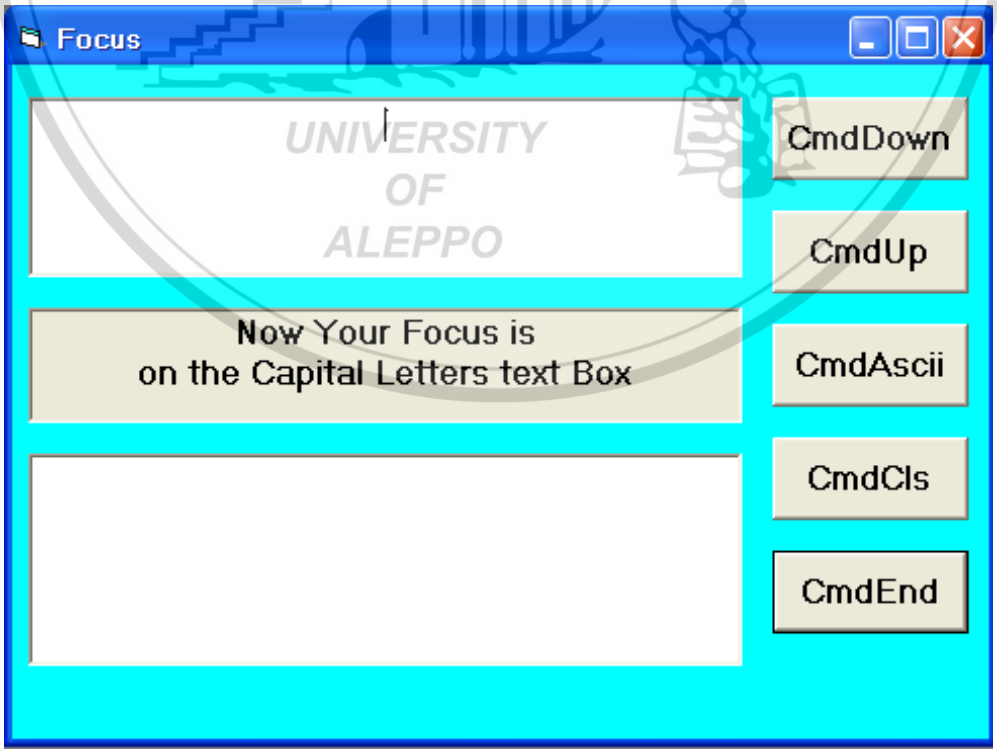
- يمكن ملاحظة تأثير عملية تغيير الخصائص $Cancel$ و $Default$ على هذه الأداة وعلاقتهما بالمفتاحين ESC و $ENTER$.

المرحلة التنفيذية:

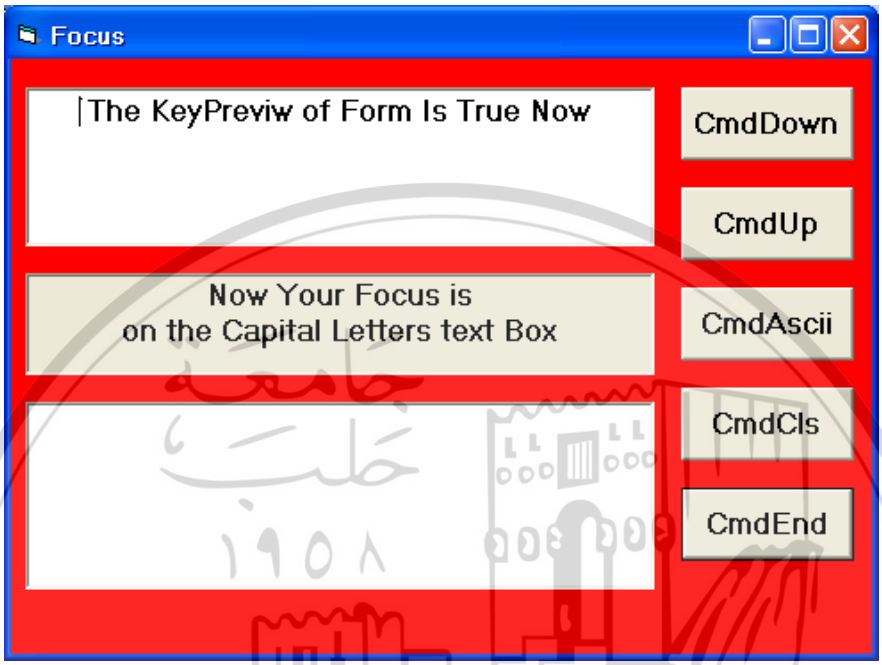
- الشكل المرئي للبرنامج قبل البدء بالمرحلة الكودية



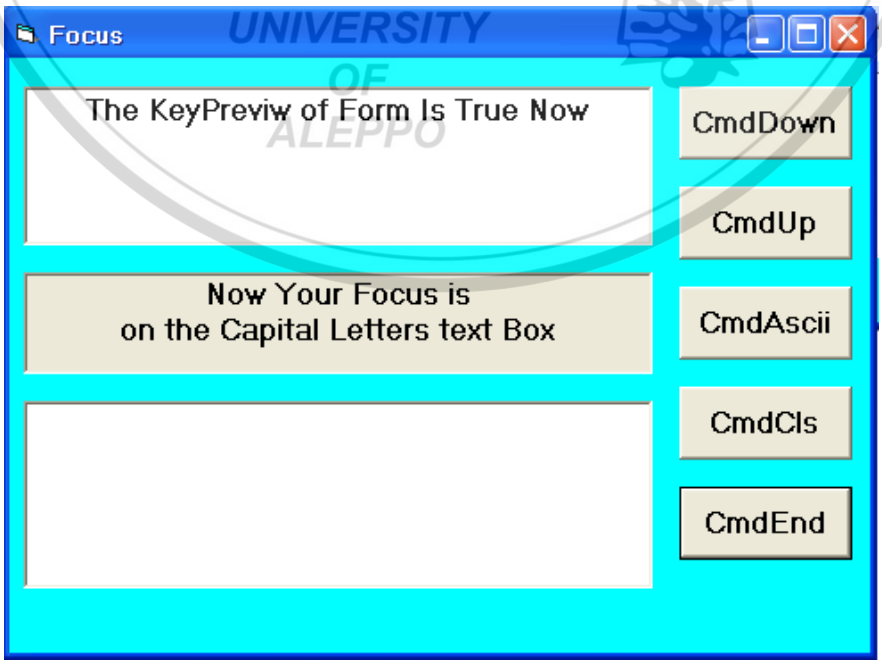
- بدء إقلاع البرنامج



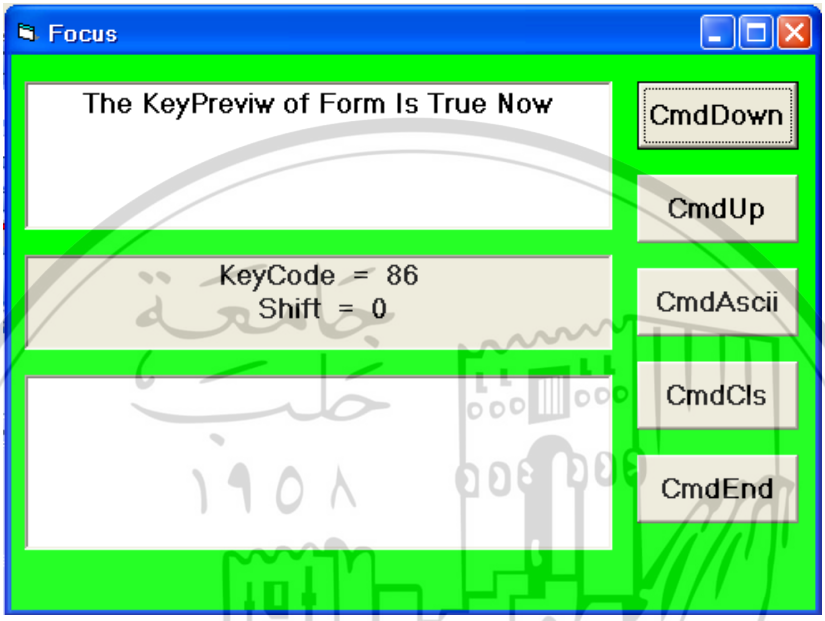
- سيقع هذا الحدث عند بدء الضغط على أي مفتاح كون خاصية الأداة (الإطار– $Form$) هي $Key\ Preview = True$
- الأفضلية للنموذج أو الإطار $Form$.



- سيقع هذا الحدث عند تحرير المفتاح المضغوط كون خاصية الأداة (الإطار– $Form$) هي: $Key\ Preview\ =\ True$
- الأفضلية للنموذج أو الإطار $Form$.

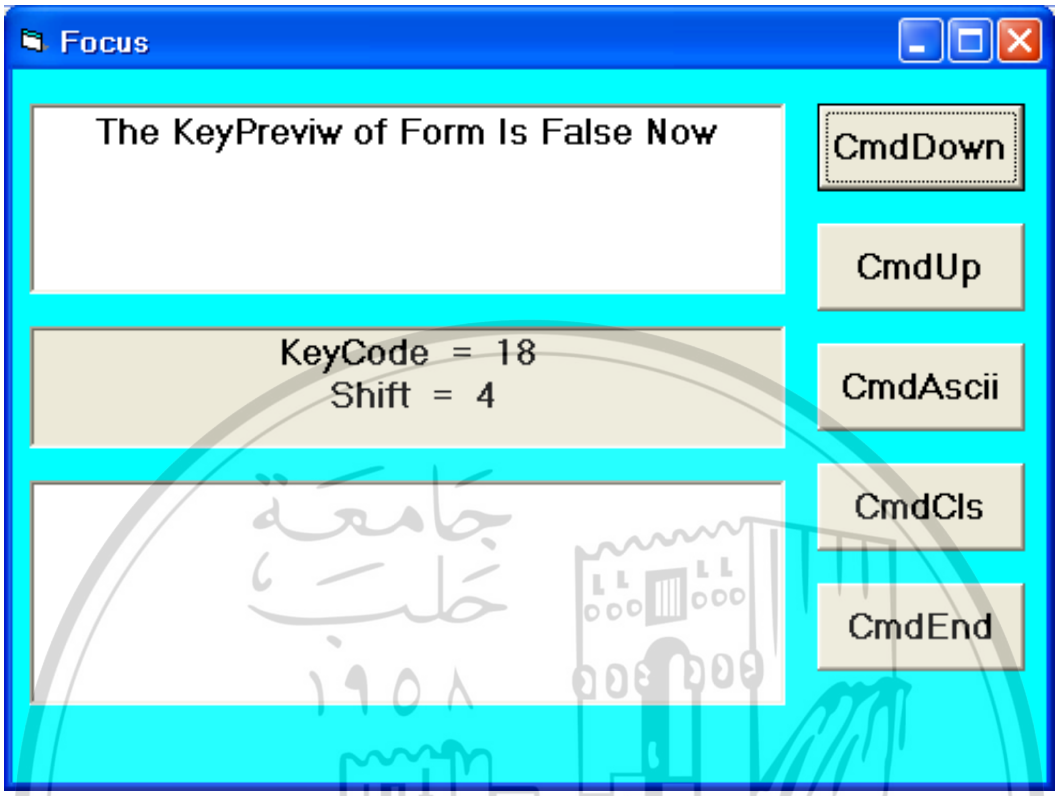


- سيقع هذا الحدث عند استمرار ضغط مفتاح ما كون خاصية الأداة (الإطار – $Form$) هي $Key\ Preview = True$
- الأفضلية للنموذج أو الإطار $Form$.



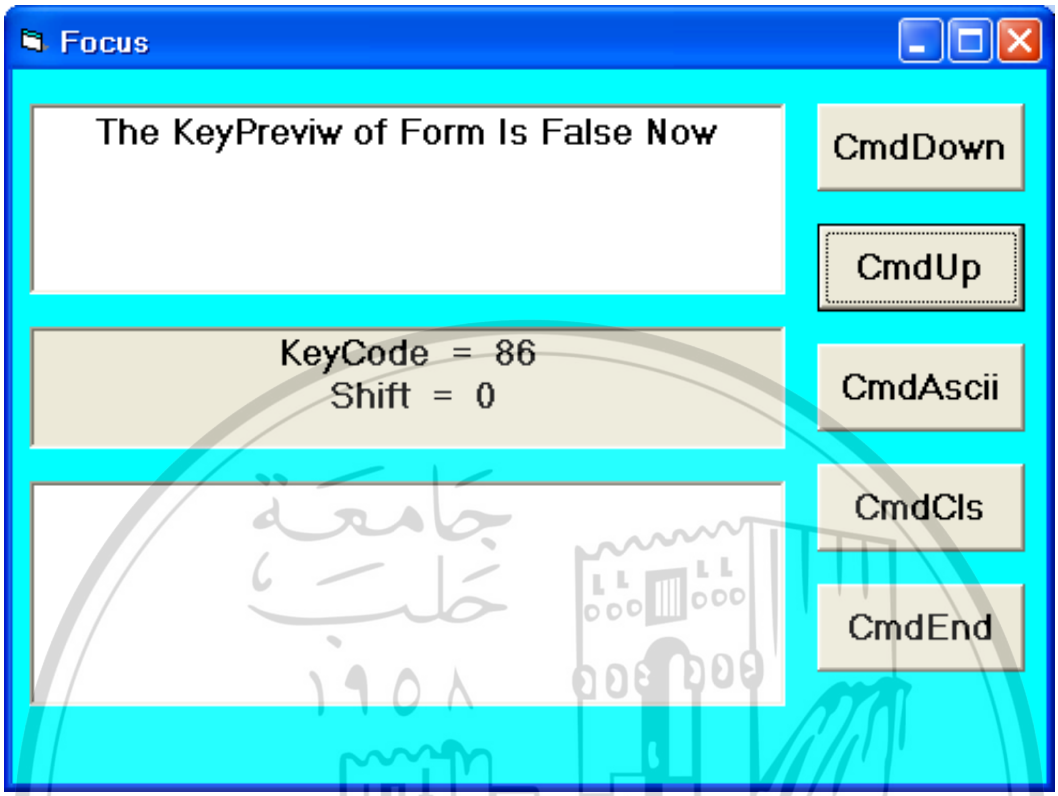
- سيظهر وجود هذا الحدث بعد إزالة إشارة الملاحظة $Remark$ عن الأمر المتعلق بعملية استمرار الضغط على أداة ما والنموذج فعال أي للحدث $Form\_KeyPress$
- سنقوم بتغيير الخاصية $KeyPreview = False$ للإطار.
- وصول التركيز للأداة $CmdDown$.

- الضغط على مفتاح ما والتركيز على الأداة $CmdDown$.



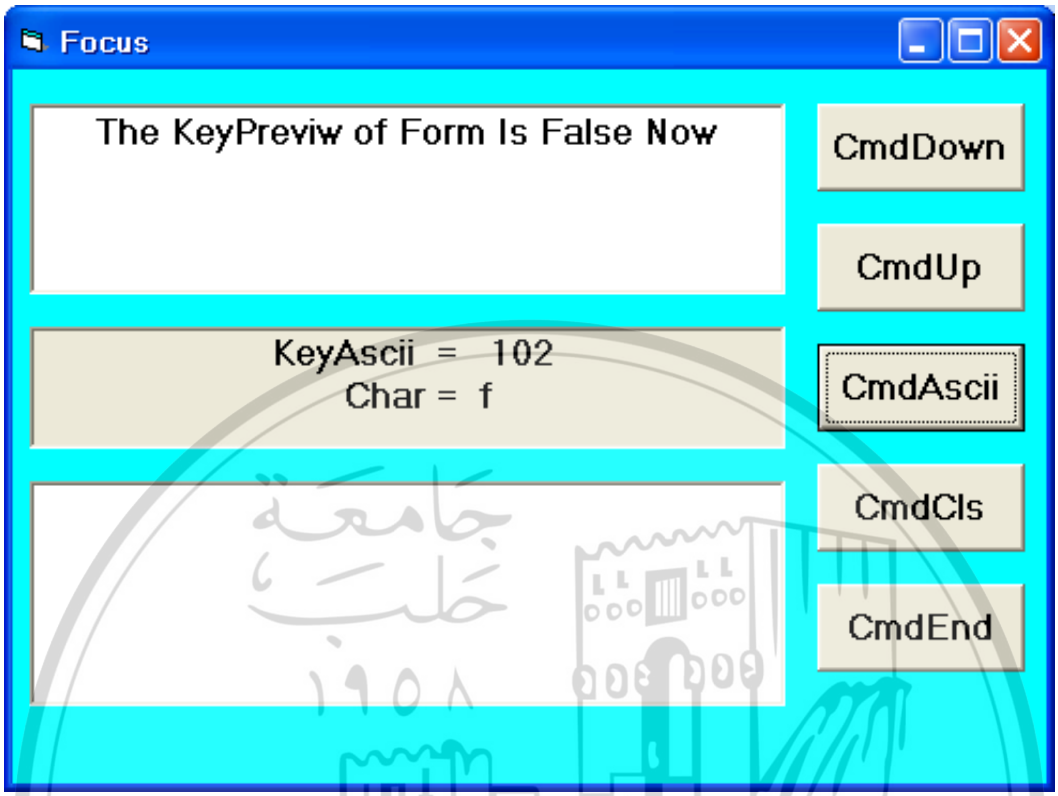
- وصول التركيز للأداة $CmdUp$.

- تحرير مفتاح ما والتركيز على الأداة $CmdUp$

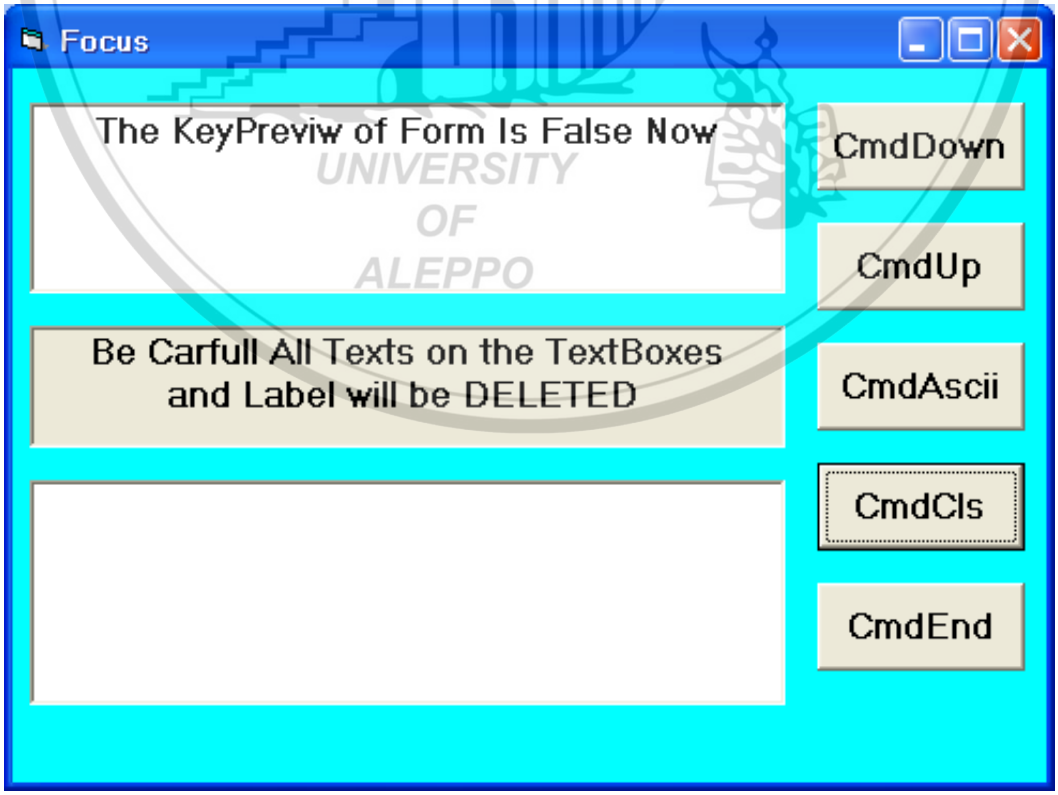


- وصول التركيز للأداة $CmdAscii$

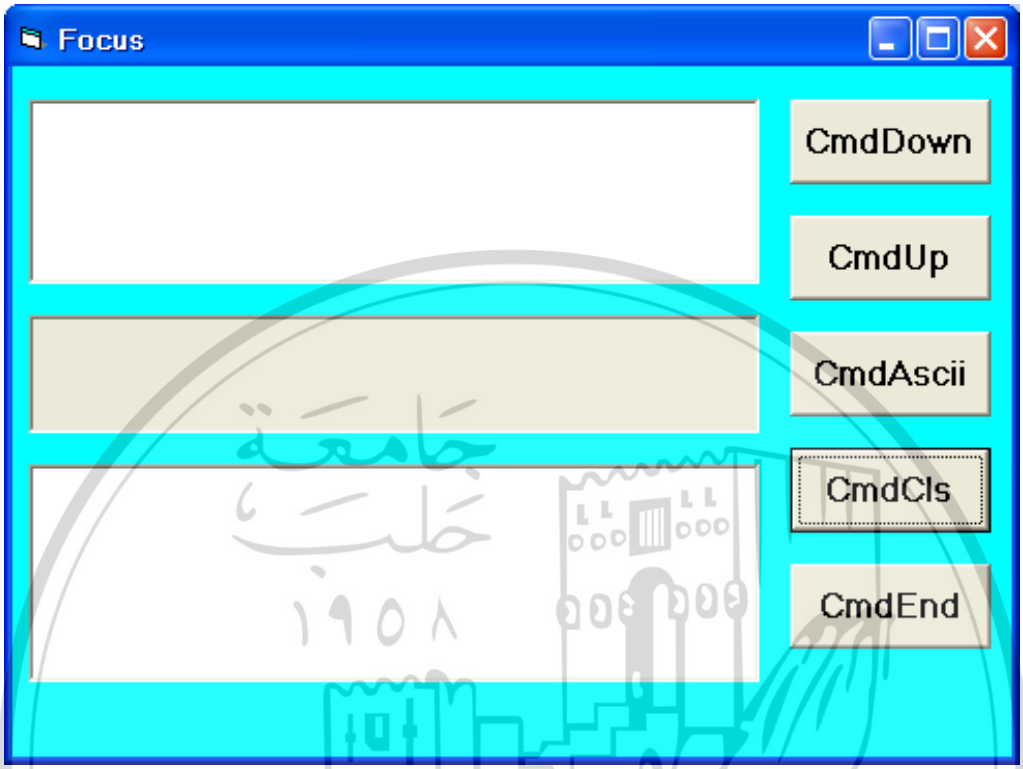
- استمرار الضغط على مفتاح ما والتركيز على الأداة $CmdAscii$



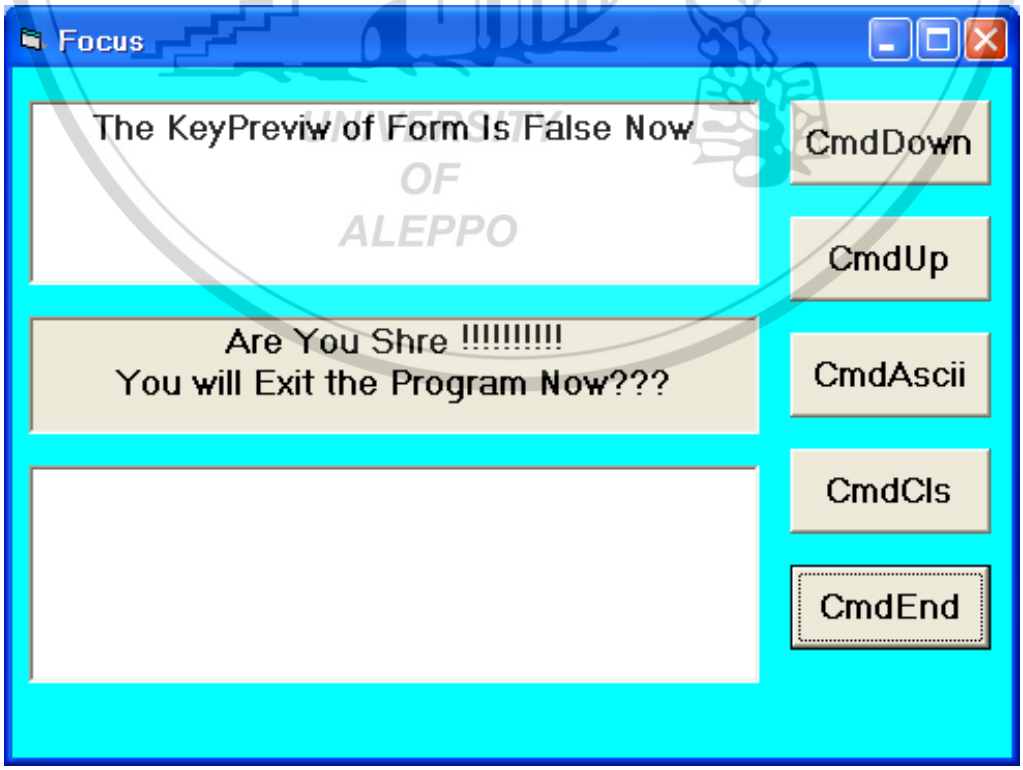
- وصول التركيز للأداة $CmdCls$



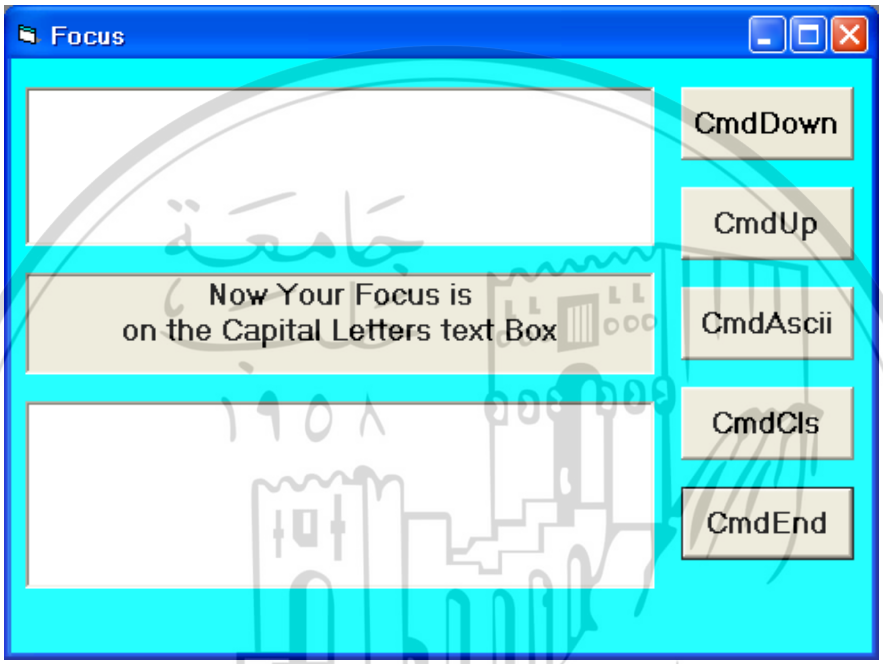
- النقر بالماوس على $CmdCls$ أو استخدام المفتاح $Enter$ أو المفتاح $Space$ والتركيز موجود على هذه الأداة.



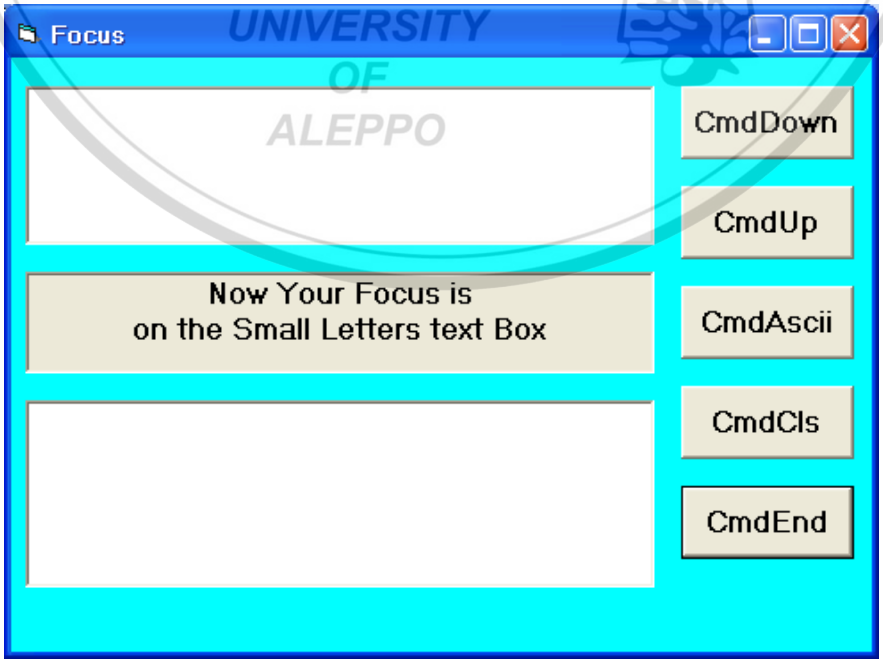
- وصول التركيز للأداة CmdEnd



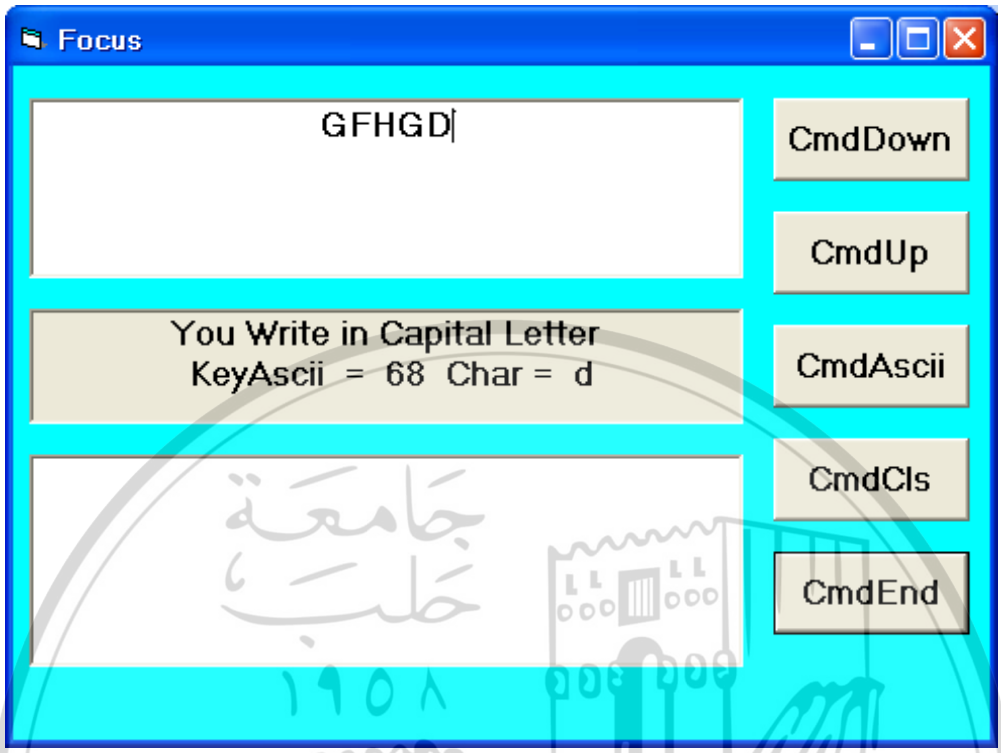
- إن وجود الخاصية $Cancel = True$ على هذه الأداة يجعل إمكانية تنفيذ هذا الأمر عند الضغط على المفتاح ESC ووجود التركيز على أي أداة.
- إن وجود الخاصية $Default = True$ على هذه الأداة يجعل إمكانية تنفيذ هذا الأمر عند الضغط على المفتاح $Enter$ ووجود التركيز على هذه الأداة فقط.
- وصول التركيز الكتابة إلى مربع نص الحروف الكبيرة:



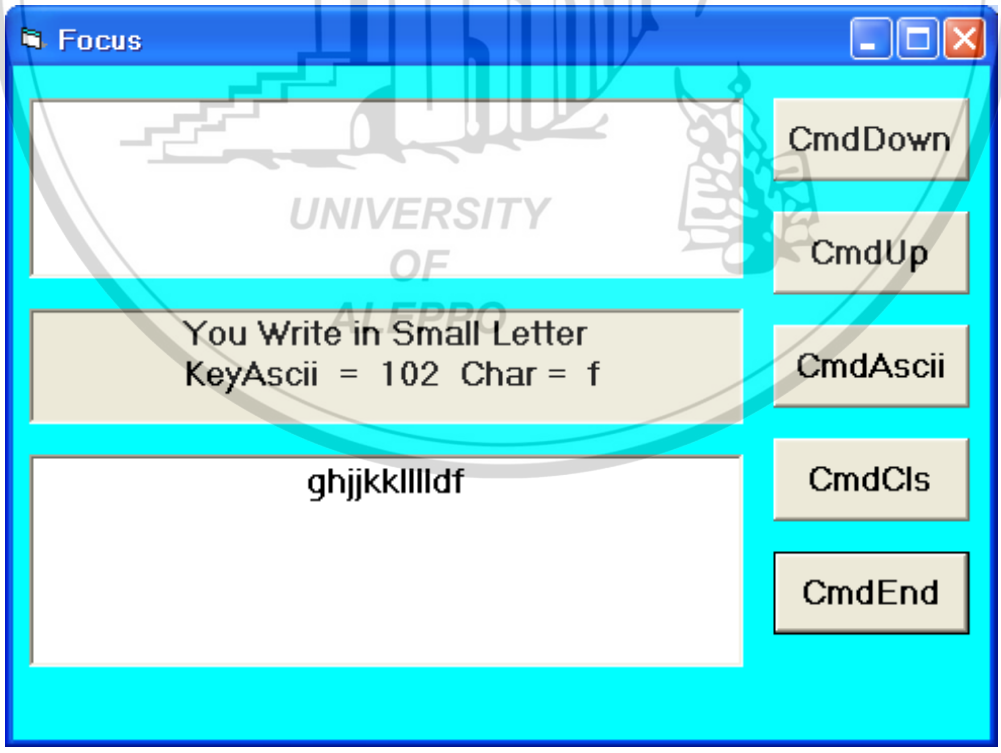
- وصول التركيز الكتابة إلى مربع نص الحروف الصغيرة:



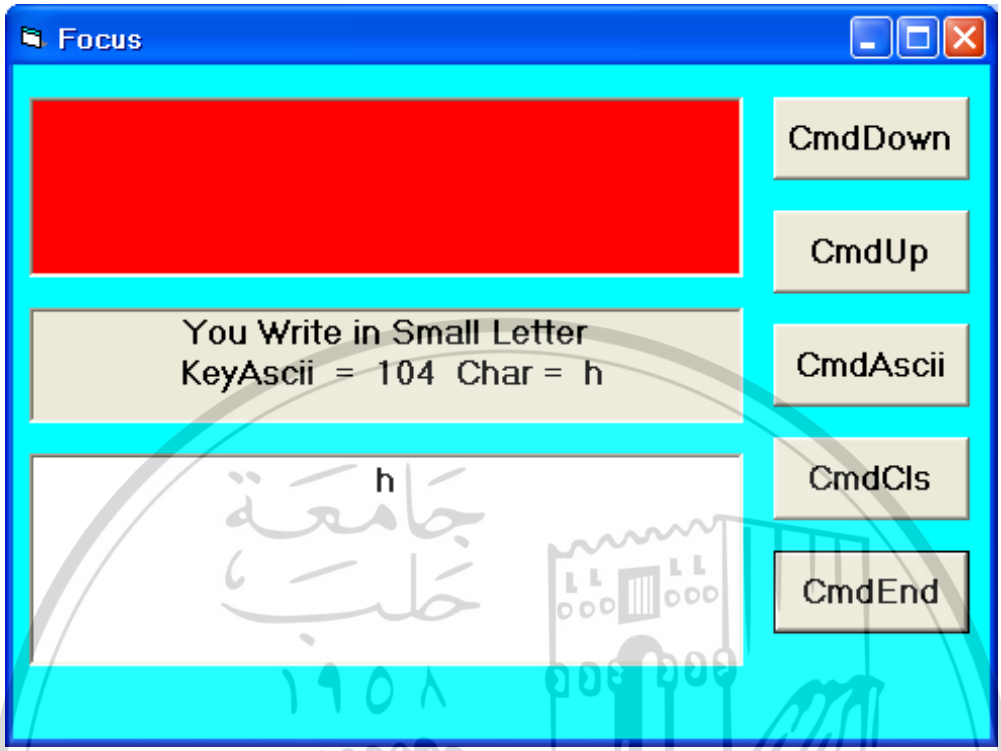
- الكتابة ضمن مربع نص الحروف الكبيرة:



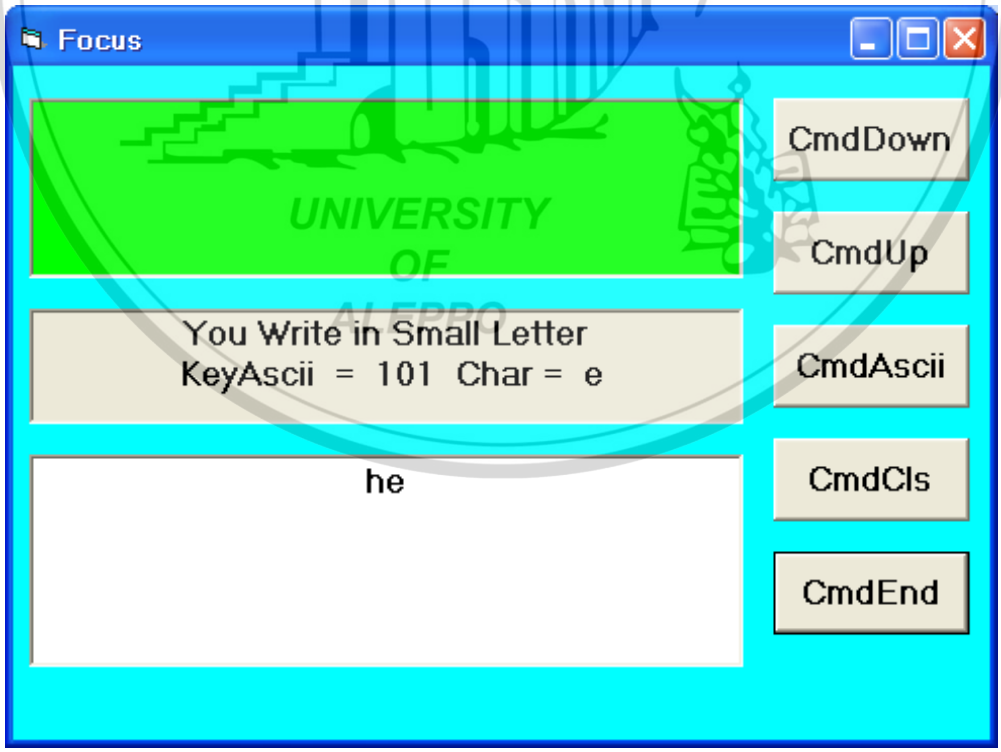
- الكتابة ضمن مربع نص الحروف الصغيرة:



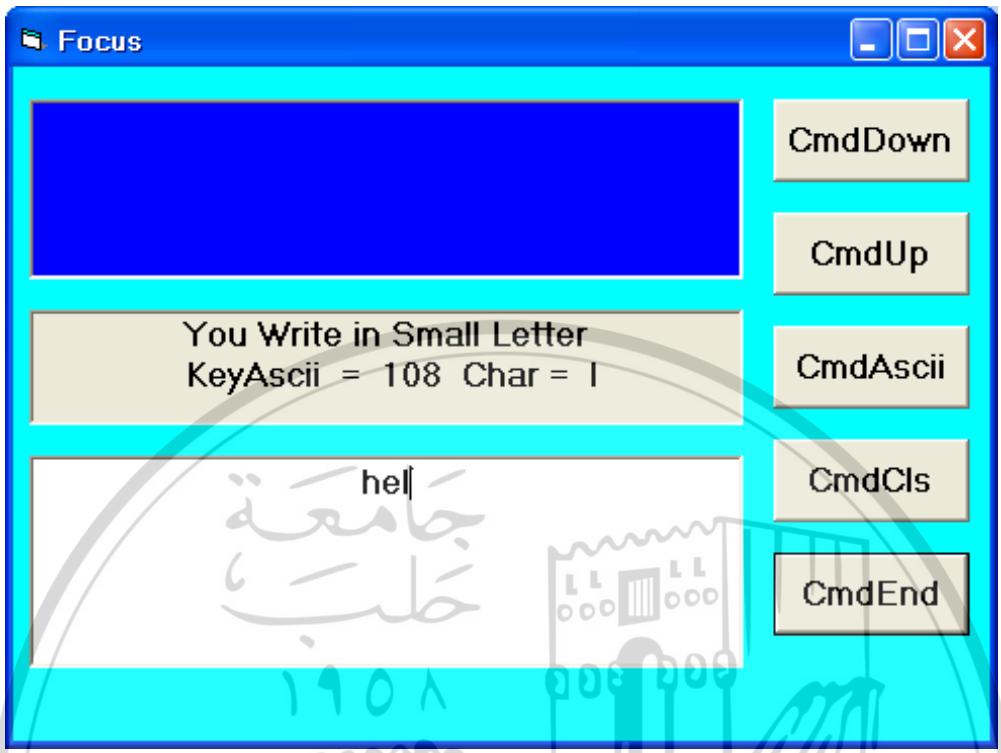
- الكتابة ضمن مربع نص الحروف الصغيرة: كلمة h = $hello$



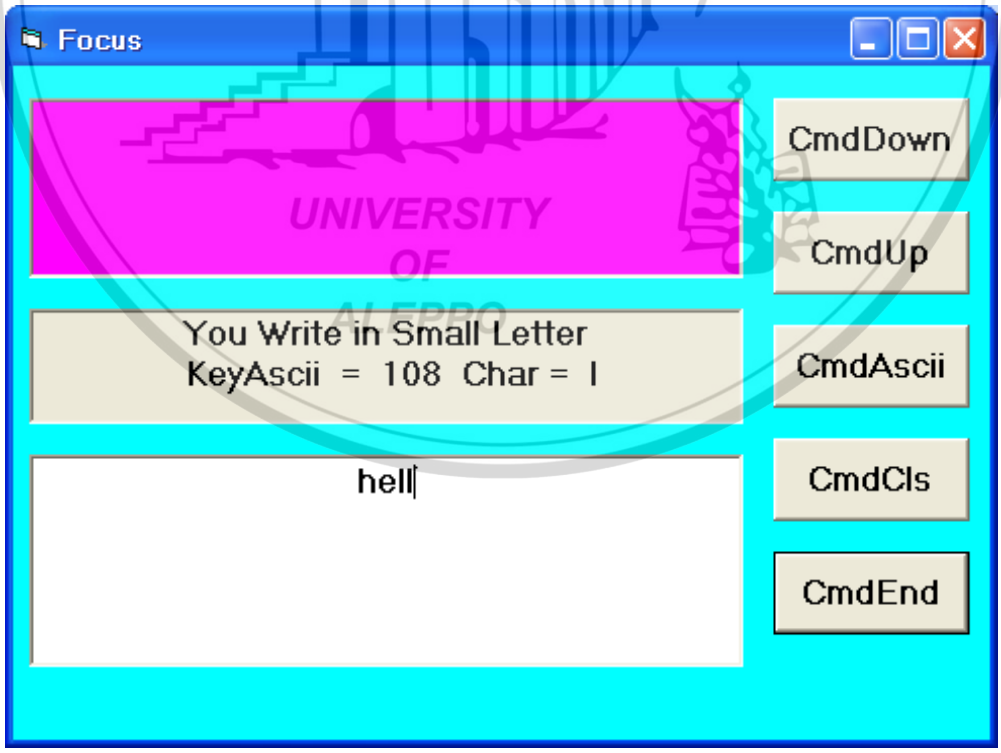
- الكتابة ضمن مربع نص الحروف الصغيرة: كلمة he = $hello$



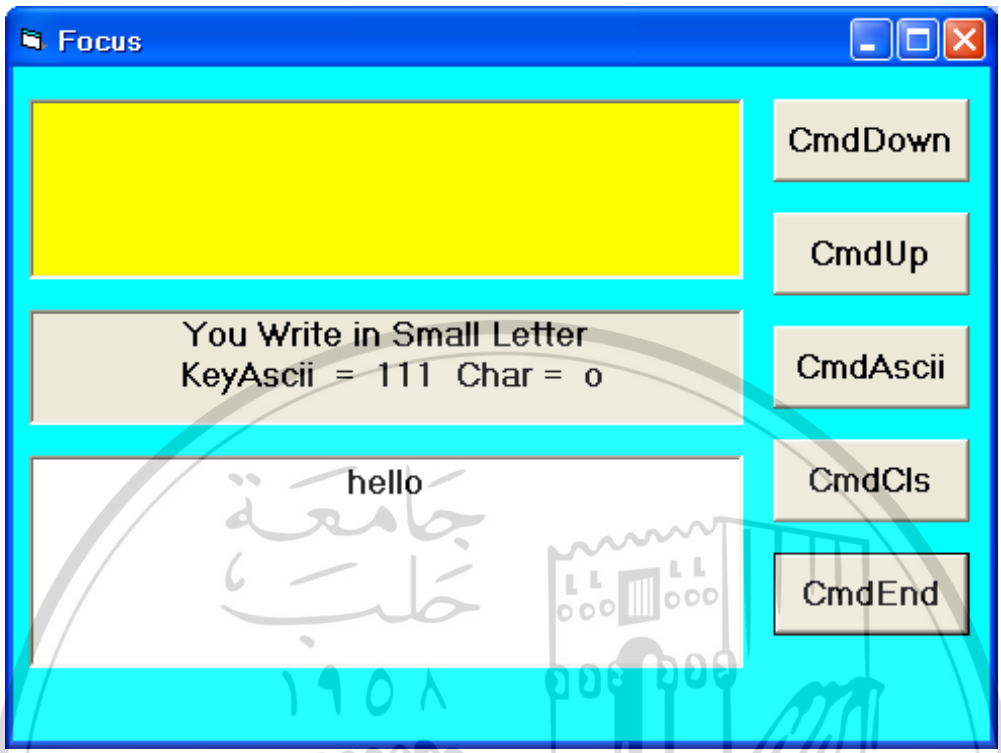
- الكتابة ضمن مربع نص الحروف الصغيرة: كلمة ***hello*** = ***hel***



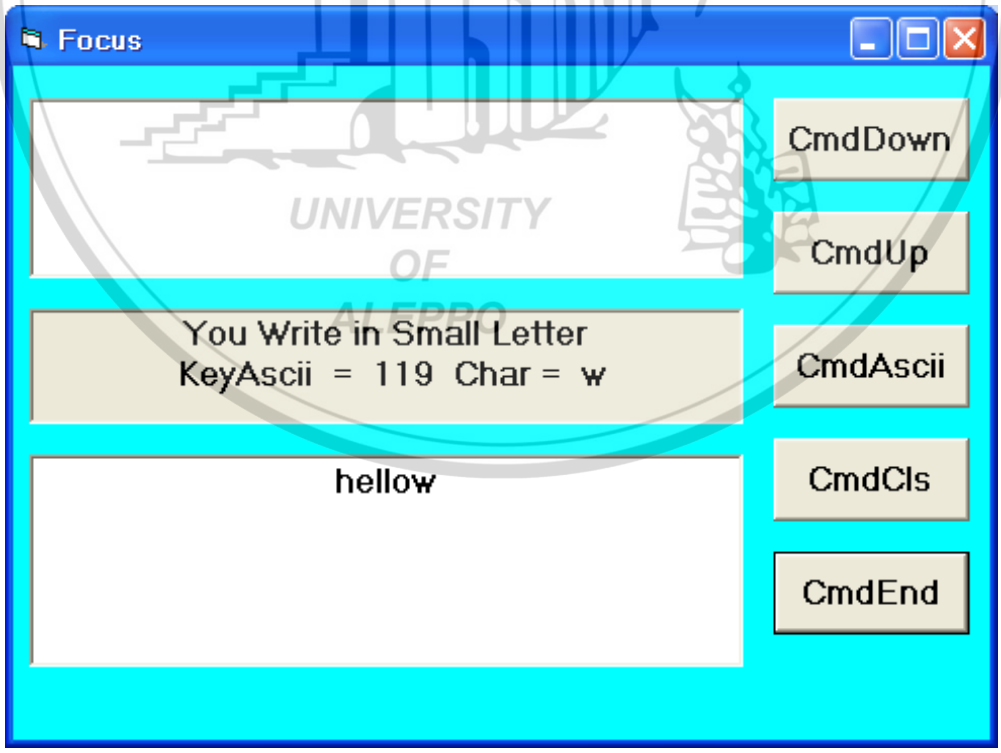
- الكتابة ضمن مربع نص الحروف الصغيرة: كلمة ***hello*** = ***hell***



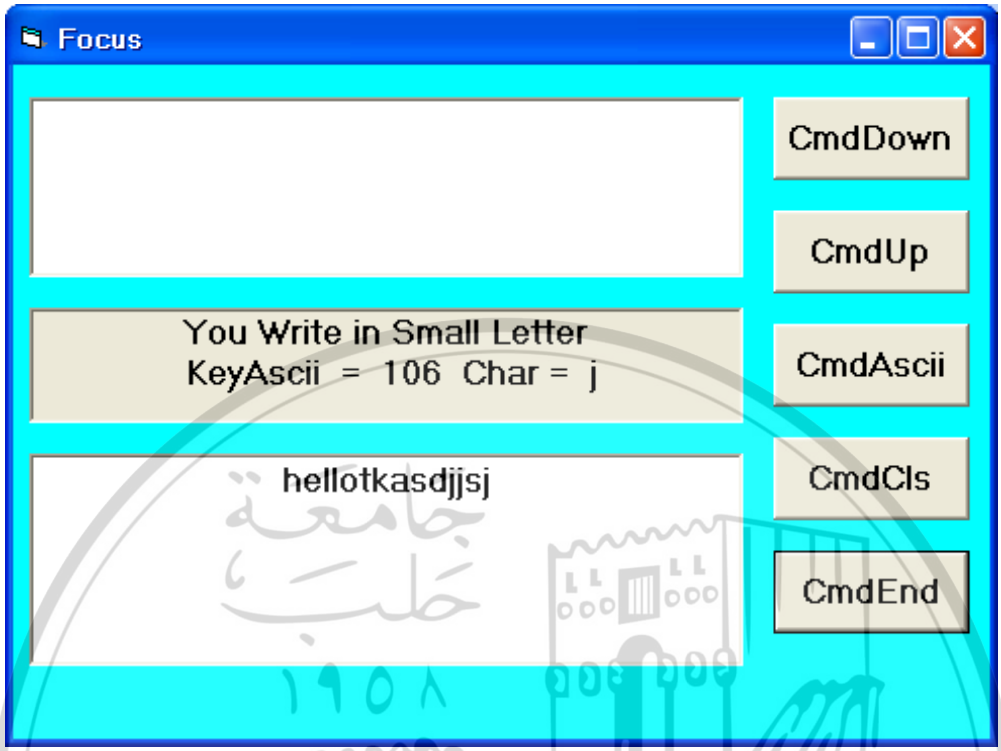
- الكتابة ضمن مربع نص الحروف الصغيرة: كلمة ***hello*** = ***hello***



- الكتابة ضمن مربع نص الحروف الصغيرة: كلمة ***hello*** = ***hellow***



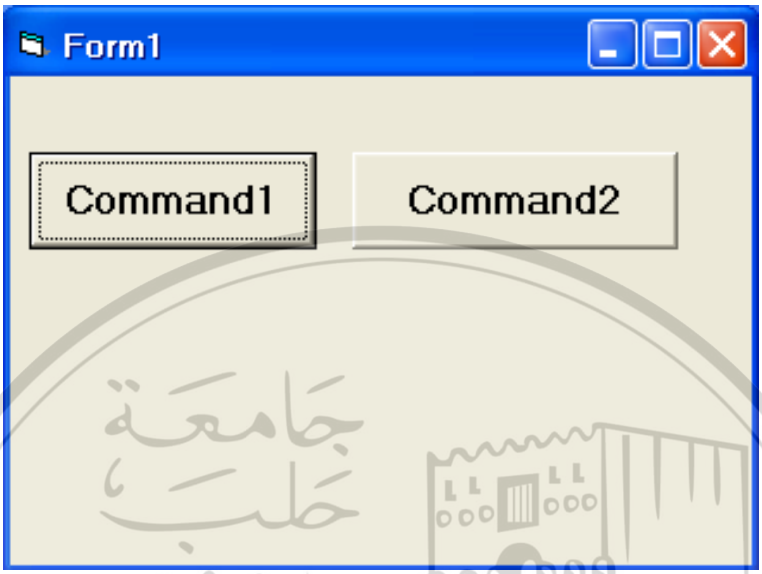
- الكتابة ضمن مربع نص الحروف الصغيرة: كلمة ***hellotkasdjjsj***



- يمكن ملاحظة تغيير انتقال التركيز بين العناصر بشكل تصاعدي باستخدام المفتاح Tab وبشكل تنازلي باستخدام التركيب $[Shift + Tab]$
- القيم الدالة على ترتيب انتقال التركيز موجودة في الخاصية $TapIndex$.

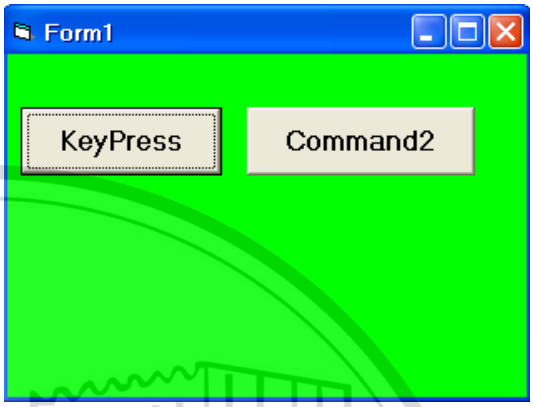
أمثلة إضافية على أحداث لوحة المفاتيح

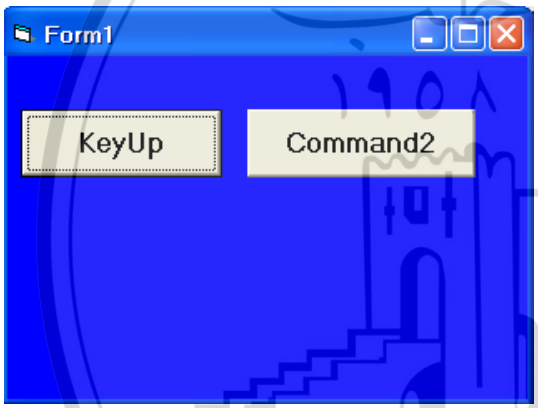
مثال (1):

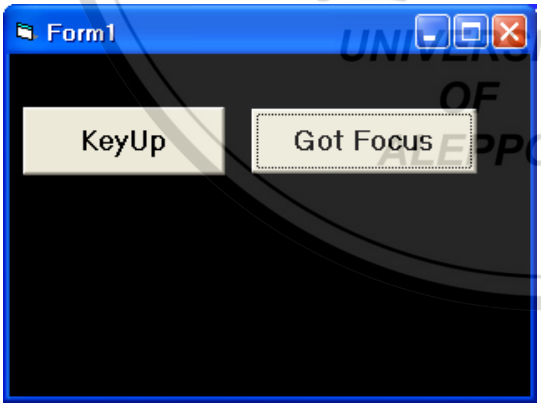


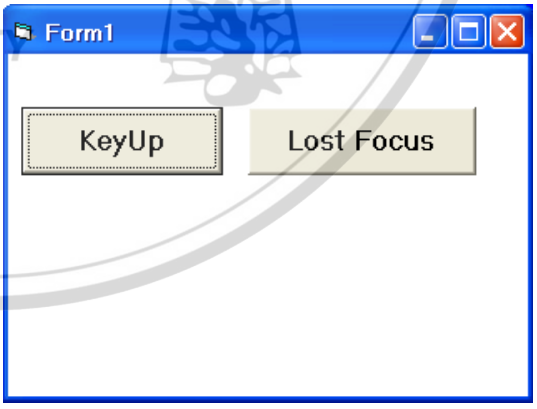

```
Private Sub Command1_KeyDown(KeyCode As Integer,
                             Shift As Integer)
Command1.Caption = "KeyDown"
Form1.BackColor = vbRed
End Sub

Private Sub Command1_KeyPress(KeyAscii As Integer)
Command1.Caption = "KeyPress"
Form1.BackColor = vbGreen
End Sub

Private Sub Command1_KeyUp(KeyCode As Integer,
                           Shift As Integer)
Command1.Caption = "KeyUp"
Form1.BackColor = vbBlue
End Sub

Private Sub Command2_GotFocus()
Form1.BackColor = RGB(0,0,0)
Command2.Caption = "Got Focus"
End Sub
```

Private Sub Command2_LostFocus()
$Form1.BackColor = RGB(255,255,255)$
$Command2.Caption = "Lost\ Focus"$
End Sub

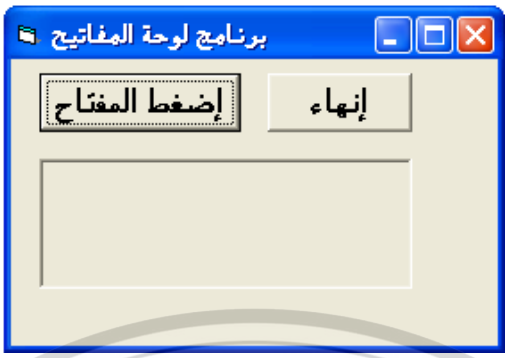




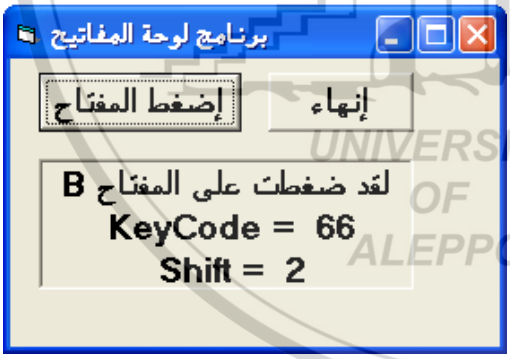


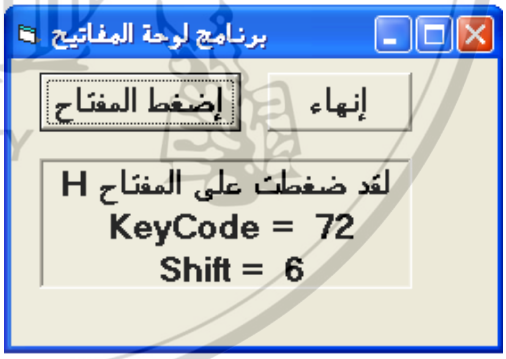


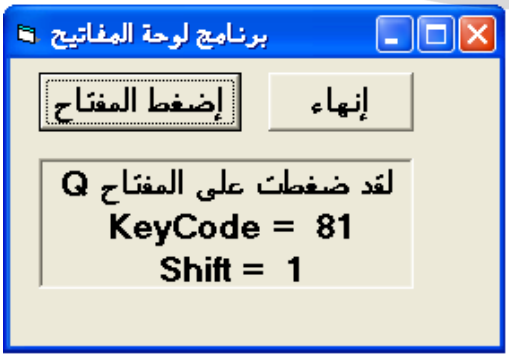
مثال (2):

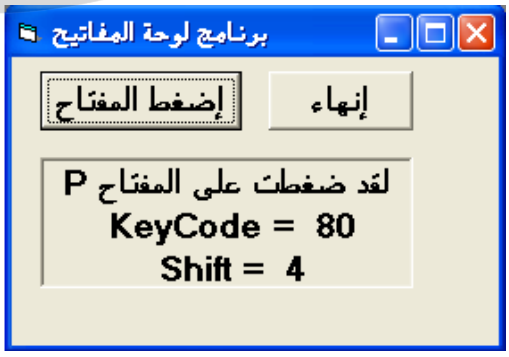



```
Option Explicit
Private Sub Command1_KeyDown(KeyCode As Integer,
                             Shift As Integer)
Lb1.Caption = " لقد ضغطت على المفتاح "
Lb1.Caption = Chr(KeyCode) + Lb1.Caption
Lb1.Caption = Lb1.Caption + vbCrLf
Lb1.Caption = Lb1.Caption + " KeyCode = " + Str(KeyCode)
Lb1.Caption = Lb1.Caption + vbCrLf
Lb1.Caption = Lb1.Caption + " Shift = " + Str(Shift)
End Sub
Private Sub Command2_Click()
End
End Sub
```









مثال (3):


```
Option Explicit
Dim SCA$
Private Sub Command1_KeyDown(KeyCode As Integer,
                             Shift As Integer)
If Shift = 1 Then
 SCA$ = "Shift"
ElseIf Shift = 2 Then
 SCA$ = "Ctrl"
ElseIf Shift = 3 Then
 SCA$ = "Shift + Ctrl"
ElseIf Shift = 4 Then
 SCA$ = "Alt"
ElseIf Shift = 5 Then
 SCA$ = "Shift + Alt"
ElseIf Shift = 6 Then
 SCA$ = "Ctrl + Alt"
ElseIf Shift = 7 Then
 SCA$ = "Shift + Ctrl + Alt"
Else
 SCA$ = "No Shift Keys"
End If
Lb1.Caption = Chr(KeyCode) + " لقد ضغطت على المفتاح "
Lb1.Caption = Lb1.Caption + vbCrLf
Lb1.Caption = Lb1.Caption + "KeyCode = " + Str(KeyCode)
Lb1.Caption = Lb1.Caption + vbCrLf
Lb1.Caption = Lb1.Caption + SCA + vbCrLf
Lb1.Caption = Lb1.Caption + "Shift = " + Str(Shift)
```

```
End Sub

Private Sub Command3_KeyUp(KeyCode As Integer,
                           Shift As Integer)
Lb1.Caption = Chr(KeyCode) + " لقد حررت المفتاح "
Lb1.Caption = Lb1.Caption + vbCrLf
Lb1.Caption = Lb1.Caption + " KeyCode = " + Str(KeyCode)
Lb1.Caption = Lb1.Caption + vbCrLf
Lb1.Caption = Lb1.Caption + SCA + vbCrLf
Lb1.Caption = Lb1.Caption + " Shift = " + Str(Shift)
End Sub

Private Sub Command4_KeyPress(KeyAscii As Integer)
Dim char
char = Chr(KeyAscii)
Lb1.Caption = char + " أنت تضغط الآن على المفتاح "
Lb1.Caption = Lb1.Caption + vbCrLf
Lb1.Caption = Lb1.Caption + "KeyAscii" + Str(KeyAscii)
              + vbCrLf
Lb1.Caption = Lb1.Caption + " Char = " + char
End Sub

Private Sub Command2_Click()
End
End Sub
```

ملاحظة:

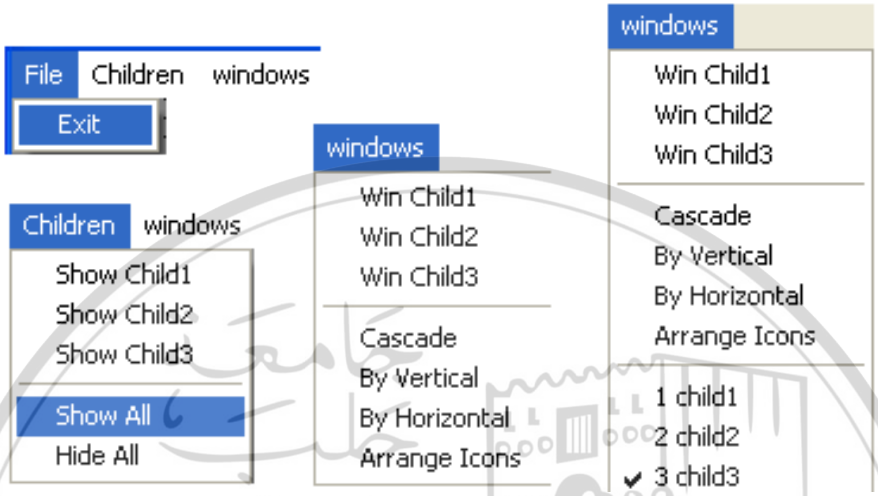
- إن قيمة ***KeyCode*** هي نفسها للحرف في حال كان كبيراً أم صغيراً ولا يأخذ قيماً في حالة الحروف العربية. بينما قيمة ***KeyAscii*** تختلف بين الحروف الكبيرة والصغيرة.
- هنا تم الضغط على المفتاح A في كل الحالات ولكن تم تنفيذ الأوامر الثلاثة مرة والحرف بحالة كبيرة ومرة بحالة صغيرة ولكن الحروف بنظام اللغة الإنكليزية.

- تم تحويل لوحة المفاتيح إلى نظام الحروف العربية والضـغط على المفتاح A والذي يوجد عليه الحرف "ش" بدون استخدام $Shift$.

- تم تحويل لوحة المفاتيح إلى نظام الحروف العربية والضـغط على المفتاح A والذي يوجد عليه الحرف " ٍ " أي "تنوين الكسر" مع استخدام $Shift$.

التمرين (٧) – النماذج *Forms*:

المطلوب تصـــميم برنامج يظهر الأحداث على النماذج *Forms* والواجهة المتعددة النماذج *MDI*.

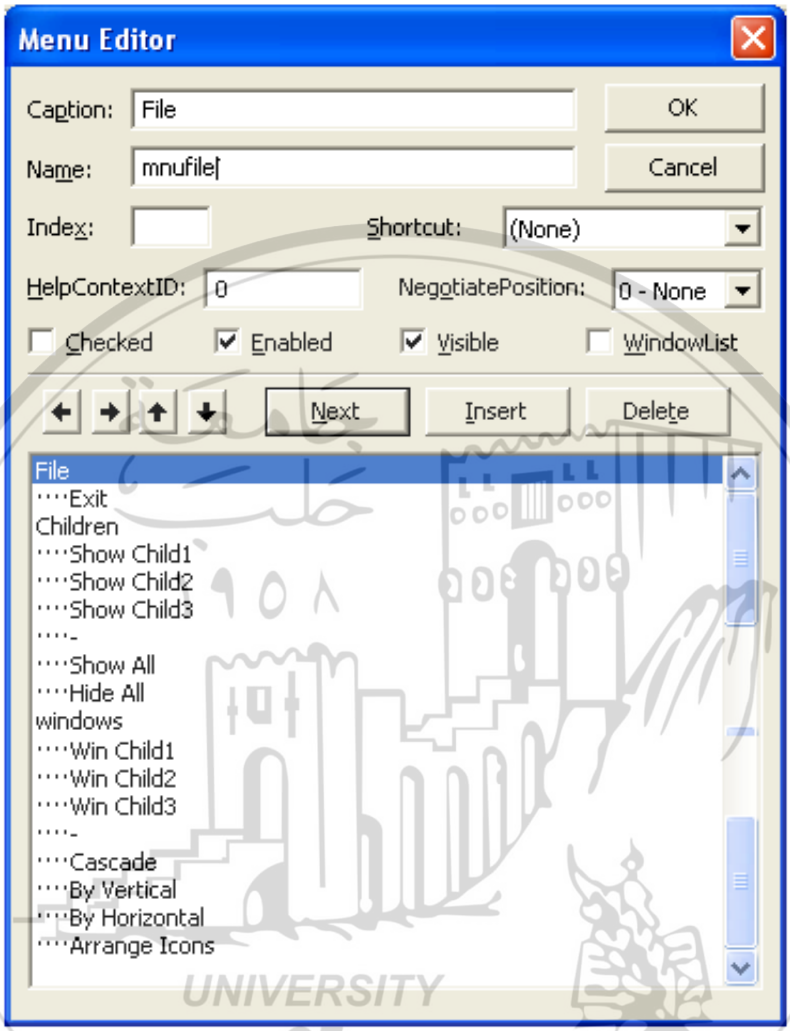


يقوم البرنامج بما يلي:

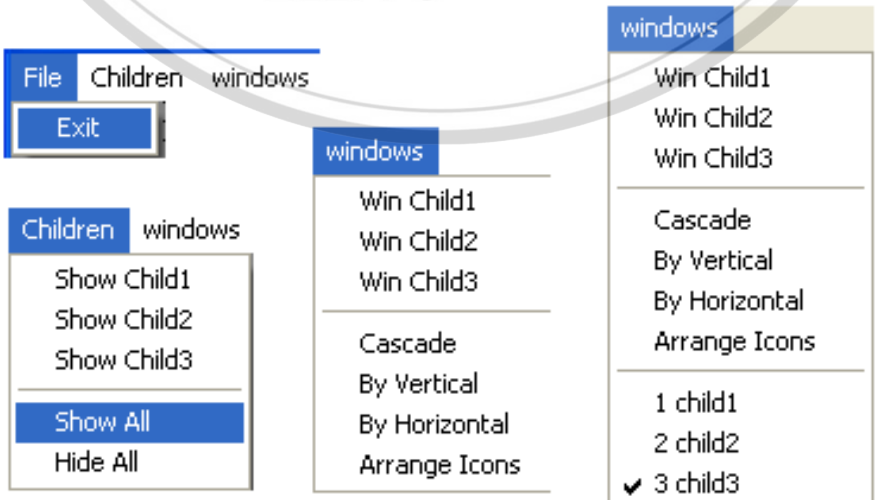
- **إظهار كل النماذج الأبناء عند إقلاع البرنامج:**
  - ⇦ إنهاء البرنامج عند استخدام الأمر ***Exit***.
- **من القائمة *Children* نقوم بما يلي:**
  - ⇦ إظهار نافذة النموذج الأول عند استخدام الأمر *Show Child1*. وبالمثل للنماذج الأخرى.
  - ⇦ إظهار كل النوافذ عند استخدام الأمر *Show All*. وإخفاءها جميعها بالأمر *Hide All*.
- **القائمة windows لترتيب نوافذ الأبناء المفتوحة في نافذة الأب كما يلي:**
  - ⇦ *Cascade* – ترتيباً متتالياً.
  - ⇦ *By Vertical* – ترتيباً متجانباً عمودياً.
  - ⇦ *By Horizontal* – ترتيباً متجانباً أفقياً.
  - ⇦ *Arrange Icons* – ترتيب الأيقونات للنوافذ المصغرة.
  - ⇦ تفيد الأوامر *Win Child1, WinChild2, …* للتنقل بين النوافذ المفتوحة.

برنامج يظهر الأحداث على النماذج *Forms* والواجهة المتعددة النماذج *MDI*.

⇦ الواجهة المرئية وتصميم شريط القوائم *Menu Editor*:



⇦ القوائم الفرعية:



البرمجة الكودية:

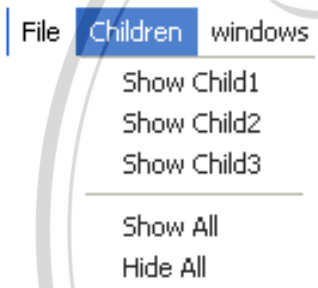
⇦ **إظهار كل النماذج الأبناء عند إقلاع البرنامج:**

```
Private Sub MDIForm_Load()
  fChild1.Show
  fChild2.Show
  fChild3.Show
End Sub
```

⇦ **إنهاء البرنامج:**

```
Private Sub mnuExit_Click()
  End
End Sub
```

⇦ **إظهار نوافذ الأبناء:**



- **إظهار النافذة الابن الأولى:**

```
Private Sub mnuchild1_Click()
fChild1.Show
End Sub
```

- **إظهار النافذة الابن الثانية:**

```
Private Sub mnuchild2_Click()
fChild2.Show
End Sub
```

- **إظهار النافذة الابن الثالثة:**

```
Private Sub mnuchild3_Click()
fChild3.Show
End Sub
```

- **إظهار كل النماذج الأبناء:**

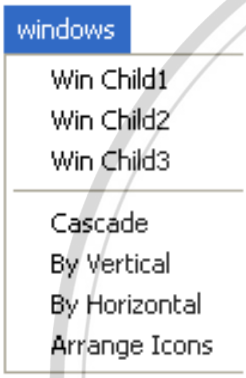
```
Private Sub mnuShowAll_Click()
fChild1.Show
```

```
fChild2.Show
fChild3.Show
End Sub
```

- إخفاء كل النوافذ الأبناء:

```
Private Sub mnuHideAll_Click()
Unload fChild1
Unload fChild2
Unload fChild3
End Sub
```

⇦ ترتيب نوافذ الأبناء ضمن نافذة الأب:



- ترتيب متتالي للنوافذ المفتوحة:

```
Private Sub mnuCascade_Click()
MDIForm1.Arrange vbCascade
End Sub
```

- ترتيب النوافذ المفتوحة بشكل متجانب عموديّاً:

```
Private Sub mnuvertical_Click()
MDIForm1.Arrange vbTileVertical
End Sub
```

- ترتيب النوافذ المفتوحة بشكل متجانب أفقيّاً:

```
Private Sub mnuhorizontal_Click()
  MDIForm1.Arrange vbTileHorizontal
End Sub
```

- ترتيب الأيقونات للنوافذ المصغرة:

```
Private Sub mnuarrange_Click()
  MDIForm1.Arrange vbArrangeIcons
End Sub
```

⇦ **ظهور النوافذ المتعلق بالقائمة Window:**

```
Private Sub mnuwin1_Click()
  fChild1.Show
End Sub

Private Sub mnuwin2_Click()
  fChild2.Show
End Sub

Private Sub mnuwin3_Click()
  fChild3.Show
End Sub
```

⇦ **أمر إلغاء تحميل النوافذ:**

- **أمر إلغاء تحميل النافذة الأولى من زر أوامر النافذة الأولى:**

```
Private Sub CmdChild1_Click()
  Unload fChild1
End Sub
```

- **أمر إلغاء تحميل النافذة الثانية من زر أوامر النافذة الثانية:**

```
Private Sub CmdChild2_Click()
  Unload fChild2
End Sub
```

- **أمر إلغاء تحميل النافذة الثالثة من زر أوامر النافذة الثالثة:**

```
Private Sub CmdChild3_Click()
Unload fChild3
End Sub
```

المرحلة التنفيذية:

خطوات بناء البرنامج:

١. تشغيل الفيجوال فتظهر نافذة أولى $Form1$.
٢. إضافة نموذج أب من النوع MDI نسميه $MDIForm1$.
٣. إضافة نموذجين أبناء $Form2$ و $Form3$.
٤. تغيير خصائص النماذج الأبناء وتسمياتها.
٥. نغير الخاصية $Backcolor$ (نختار للأول اللون الأحمر والثاني أخضر والثالث أزرق).
   ☹ نسميها $fchild1$ و $fchild2$ و $fchild3$
   ☹ نغير الخاصية $MDIChild = True$ لكل النماذج الأبناء.
٦. نصمم شريط القوائم كما هو مبين بالشكل ونختار $WidowList$ للقائمة $Window$.
٧. نكتب الأوامر الكودية للقوائم على النموذج الأب.
٨. نكتب البرمجة الكودية لزر الأمر $Close$ على كل النوافذ الفرعية.

خطوات تنفيذ البرنامج:

١. التدرب على إقلاع البرنامج وإغلاقه.
٢. إظهار نافذة وكل النوافذ وإخفاء نافذة وكل النوافذ.
٣. التدرب على التنقل بين النوافذ الأبناء المفتوحة.
٤. التدرب على أنواع الترتيبات المطروحة.

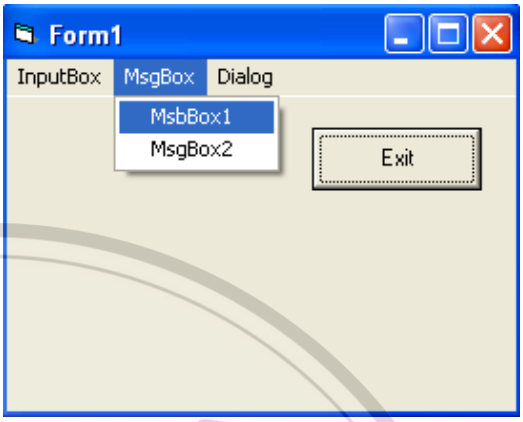
☺ لاحظ أن خيار ترتيب الأيقونات لن يتفعل مالم تكن كل النوافذ مصغرة في شريط النافذة الأب.

☺ دون كل الملاحظات.

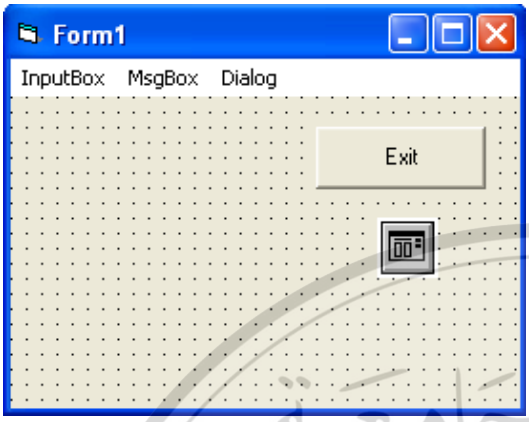
☺ ما هي الإمكانيات الممكن تطويرها على هذا التمرين.

التمرين (٨) – مربعات الحوار الشائعة *Common Dialog Boxes*:

المطلوب تصميم برنامجاً لإظهار مربعات الحوار الشائعة واجهته المرئية كما يلي:



النافذة أثناء المرحلة التنفيذية



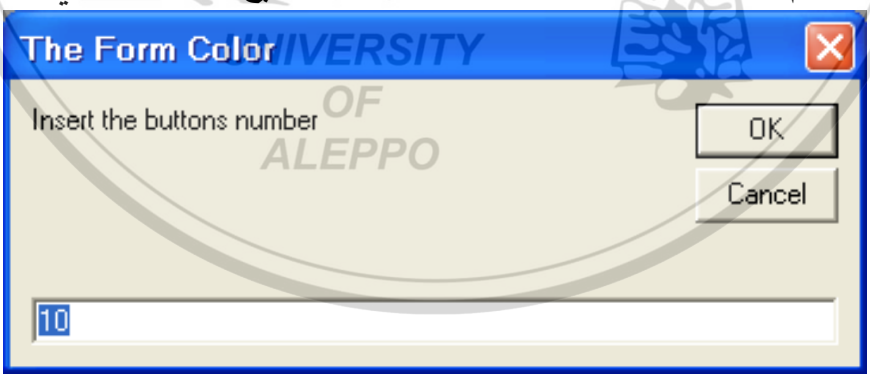
النافذة أثناء المرحلة التصميمية

يعمل البرنامج كما يلي:

١) مفتاح *Exit* لإنهاء عمل البرنامج.

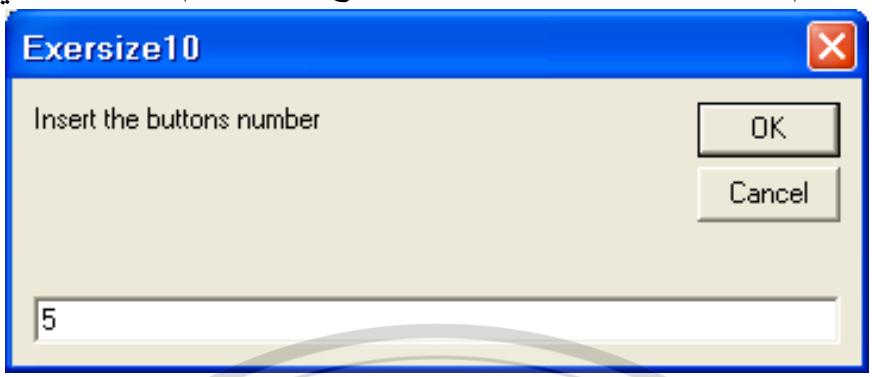
٢) عند استخدام القائمة الرئيسية *Dialog* يظهر مربع إدخال يسمح بإدخال أحد الأرقام $1, 2, \ldots 5$ ويظهر مربع الحوار الملائم (الذي له الخاصية *Action* تساوي الرقم المُدخل). وفي حال تم إدخال رقم غير هذه الأرقام يعطي رسالة توضيحية أن الأرقام المُدخلة يجب أن تكون ضمن المجال المُحدد.

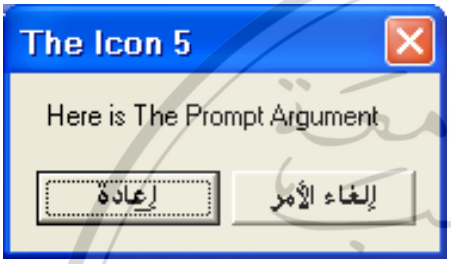
٣) عند استخدام القائمة الرئيسية *InputBox* يظهر مربع الإدخال التالي:



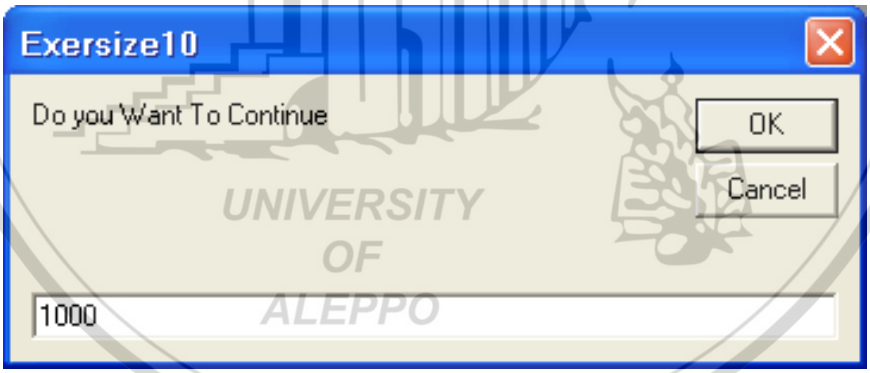
إذا كان الرقم المُدخل يساوي 15 يجعل لون خلفية الإطار أخضر وإلا يظهر العبارة المبيَّنة. ثم يعيد عملية الإدخال من جديد.

٤) عند استخدام القائمة الفرعية $MsgBox1$ يسمح بإدخال رقم معين كما يلي:





- يظهر هذا الرقم بجانب العبارة "$The\ Icon$" في عنوان مربع الحوار التالي ويظهر معنى هذا الرقم على مربع الحوار من خلال الرموز الذي تدل على هذا الرقم.
- يسأل بعد ذلك عن الرغبة في الاستمرار وحسب الرقم الذي نستخدمه فإما يُنهي البرنامج إذا كان الرقم المُدخل يساوي 1000 أو يعيد إدخال رقم آخر ويُظهر مربع حوار أيقوناته ملائمة للرقم المُدخل من جديد كما مر سابقاً.

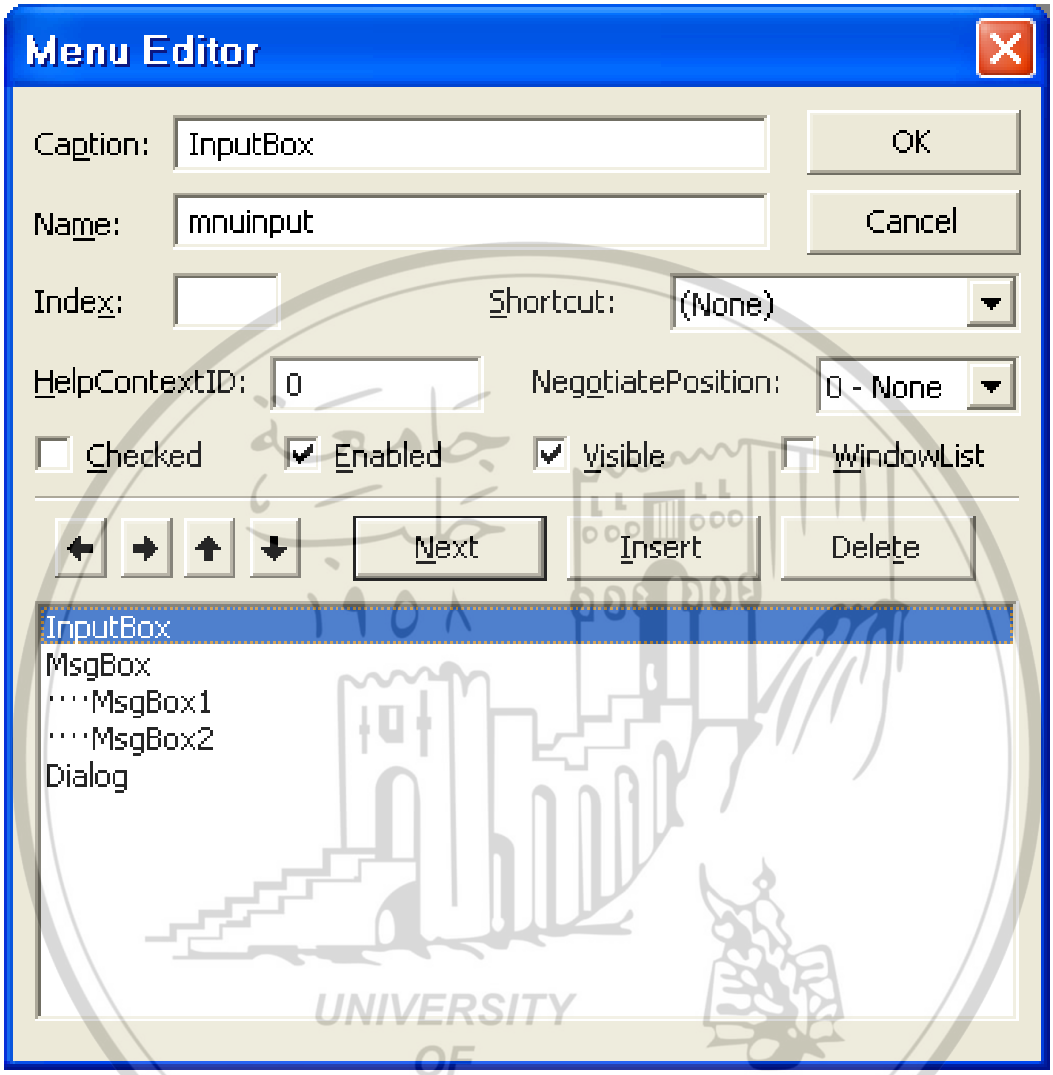


٥) عند استخدام القائمة الفرعية MsgBox2 يظهر مربع الحوار التالي والذي يعمل كما يلي:

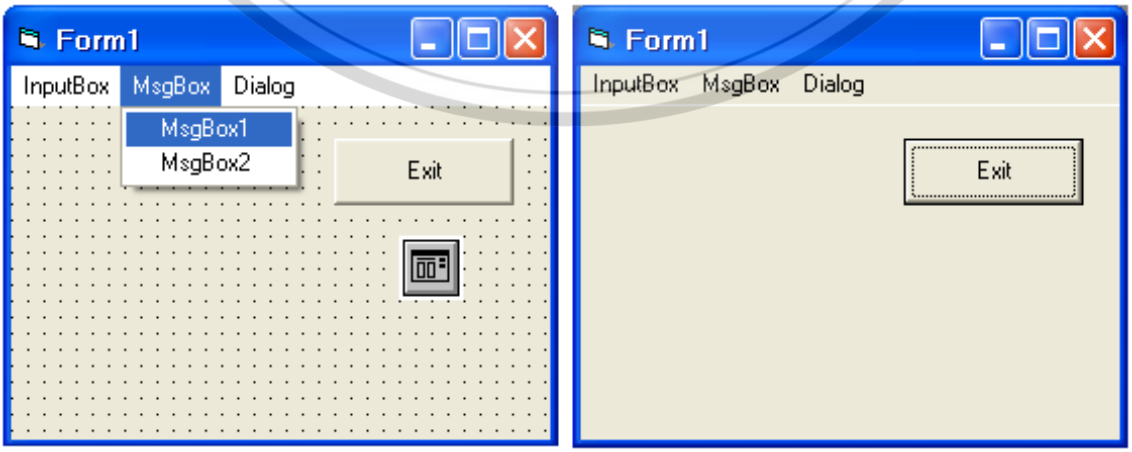
- عند النقر على "موافق" يجعل الإطار أحمر.
- عند النقر على "إلغاء الأمر" يجعل الإطار أزرق.

مربعات الحوار الشائعة *Common Dialog Boxes*:

⇦ الواجهة المرئية وتصميم شريط القوائم *Menu Editor*:



⇦ القوائم الفرعية:

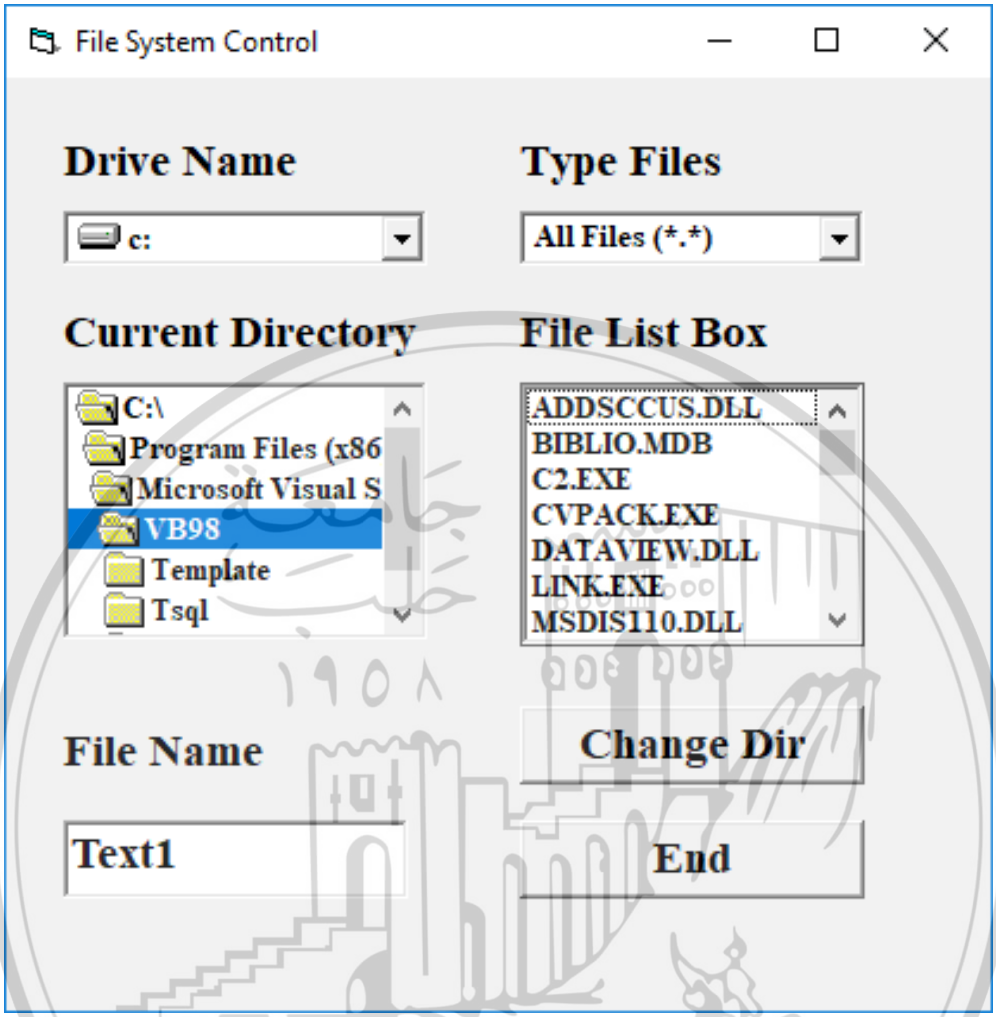


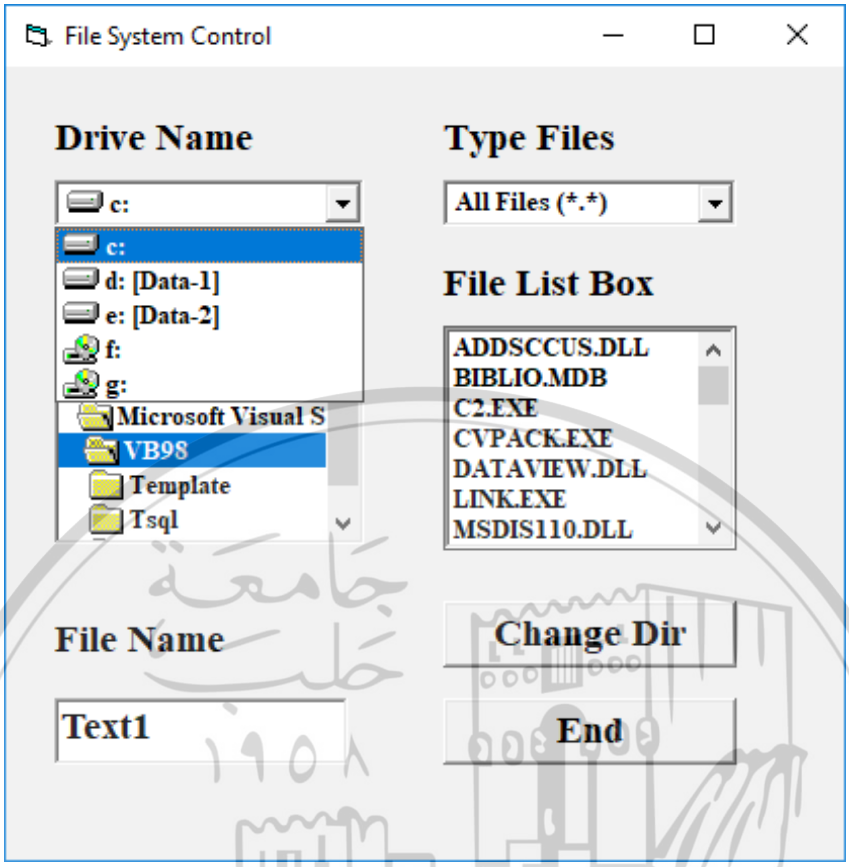
المرحلة الكودية:

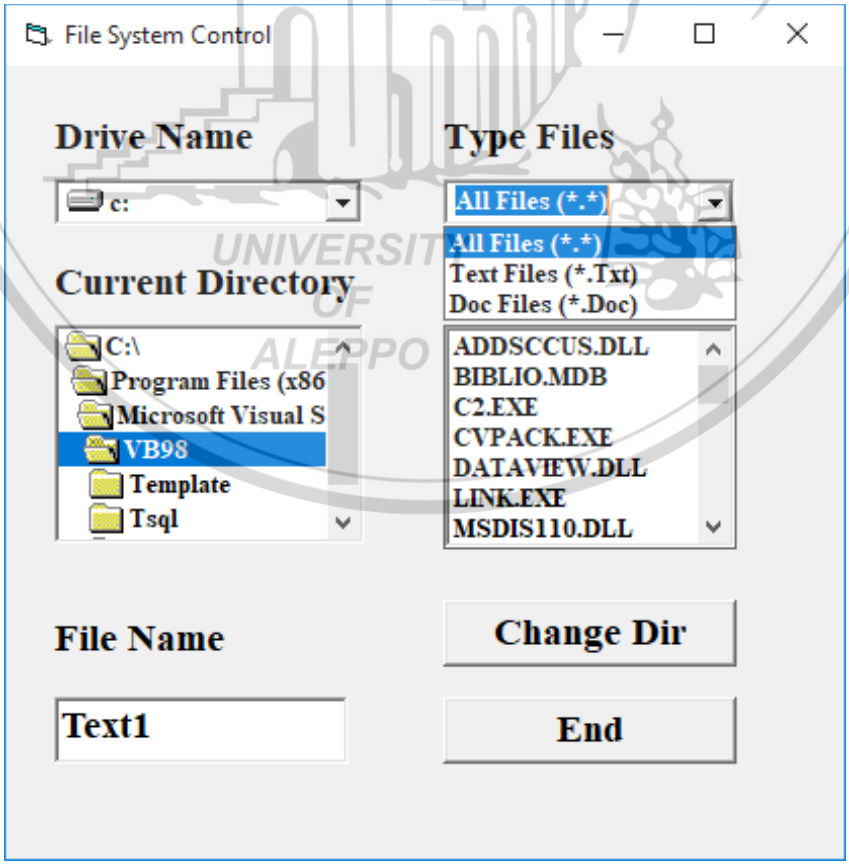
```
Private Sub mnudialog_Click()
  c = InputBox ("Inser Number Between 1 And 5")
  If  c = 1 Then
     CommonDialog1.Action = 1
  ElseIf c = 2 Then
     CommonDialog1.Action = 2
  ElseIf c = 3 Then
     CommonDialog1.Action = 3
  ElseIf c = 4 Then
     CommonDialog1.Flags = cdlCFBoth Or cdlCFEffects
     CommonDialog1.Action = 4
  ElseIf c = 5 Then
     CommonDialog1.Action = 5
  Else
     MsgBox "Insert Number Between 1 And 5"
  End If
End Sub

Private Sub mnuinput_Click()
10 B = InputBox("Insert the buttons number",
                                    "The Form Color", 10)
   If B = 15 Then
      Form1.BackColor = vbGreen
   Else
      MsgBox "your Number is Wrong"
      GoTo 10
   End If
End Sub

Private Sub mnumsg1_Click()
20 B = InputBox("Insert the buttons number")
      Title = "The Icon " + B
      Argument = "Here is The Prompt Argument"
      A = MsgBox(Argument, B, Title)
      c = InputBox("Do you Want To Continue")
   If c = 1000 Then
      End
```

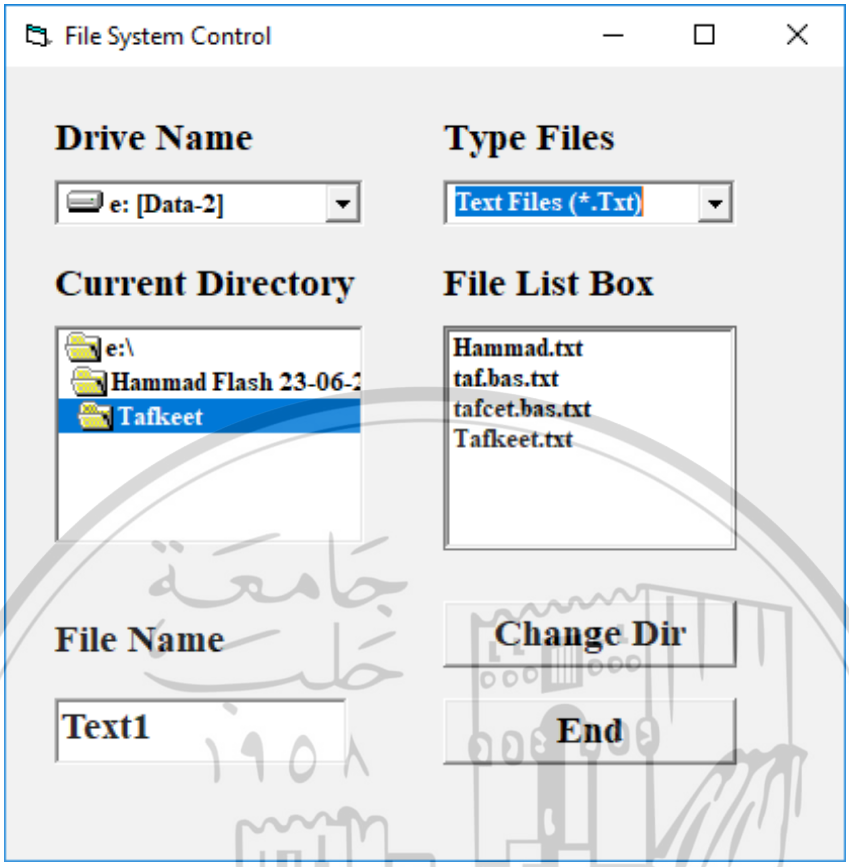
```
    Else
        GoTo 20
    End If
End Sub

Private Sub mnumsg2_Click()
    Dim MyVar
    MyVar = MsgBox("Hello World!", 65, "MsgBox Example")
    ' MyVar contains either 1 or 2,
    ' depending on which button is clicked.
    If MyVar = 1 Then
        Form1.BackColor = vbRed
    Else
        Form1.BackColor = vbGreen
    End If
End Sub

Private Sub Command1_Click()
    End
End Sub
```

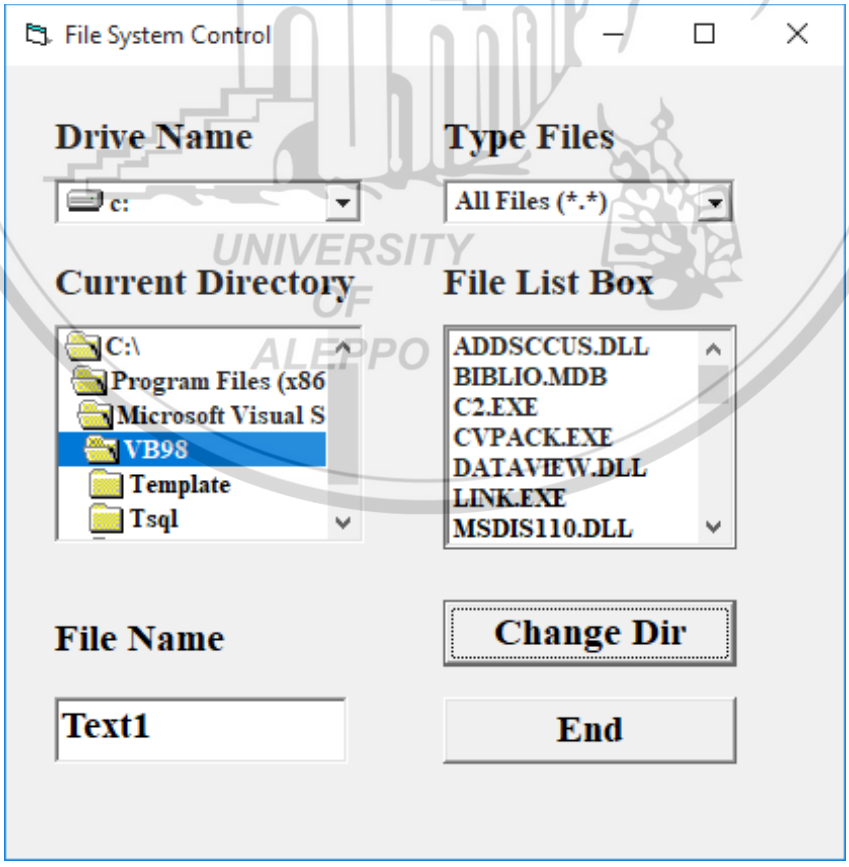
التمرين (٩) – عناصر التحكم بالملفات $File - System\ Control$:

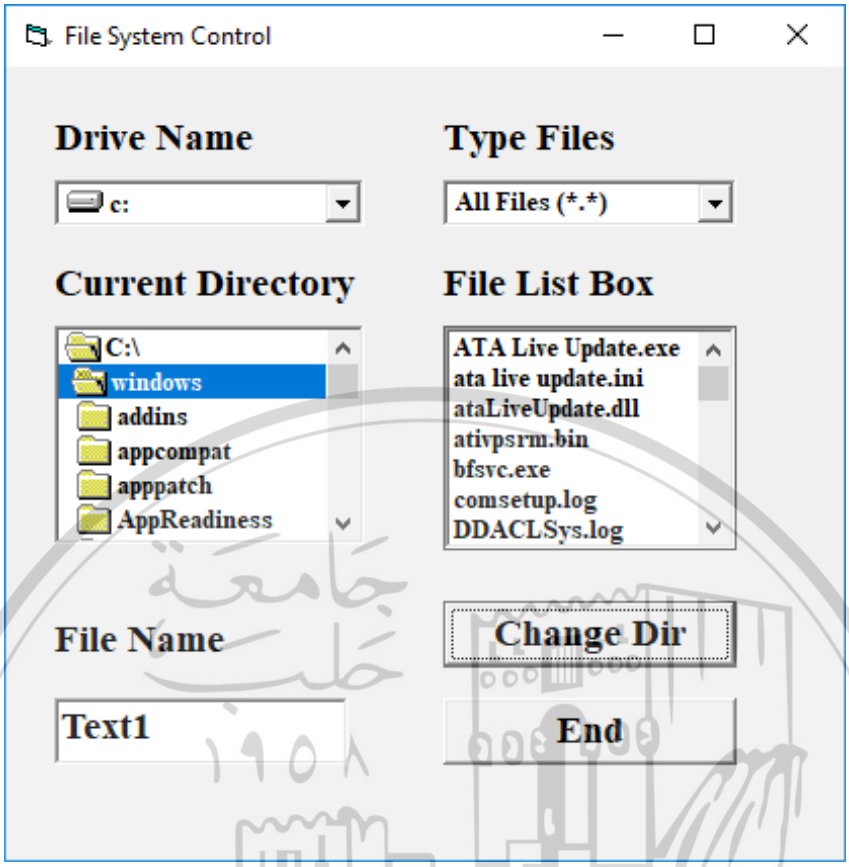


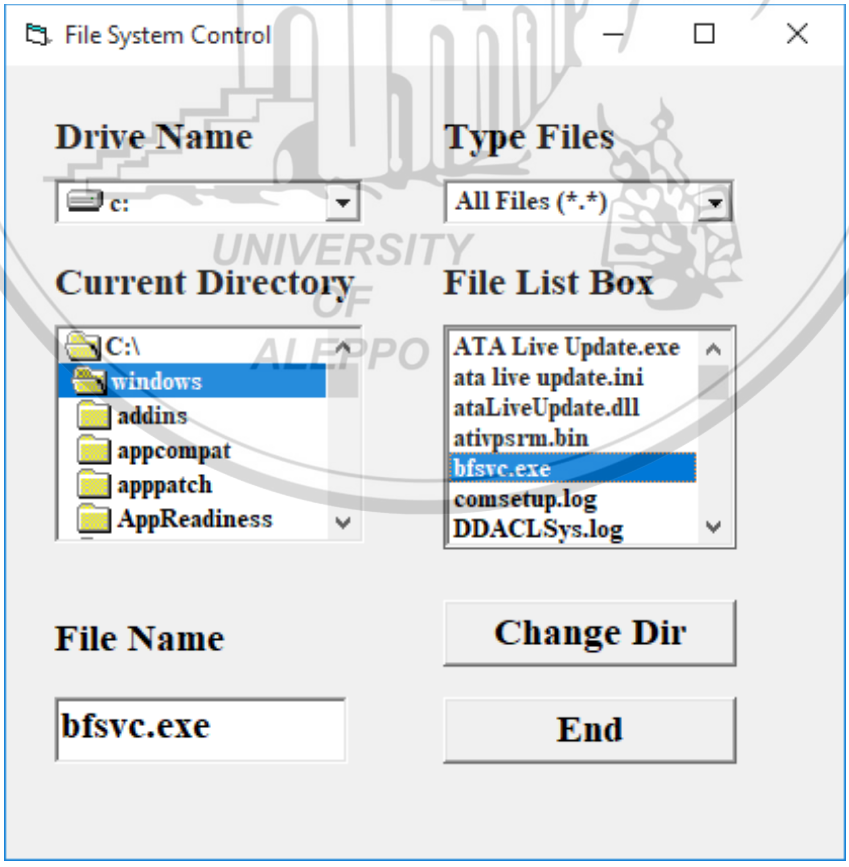










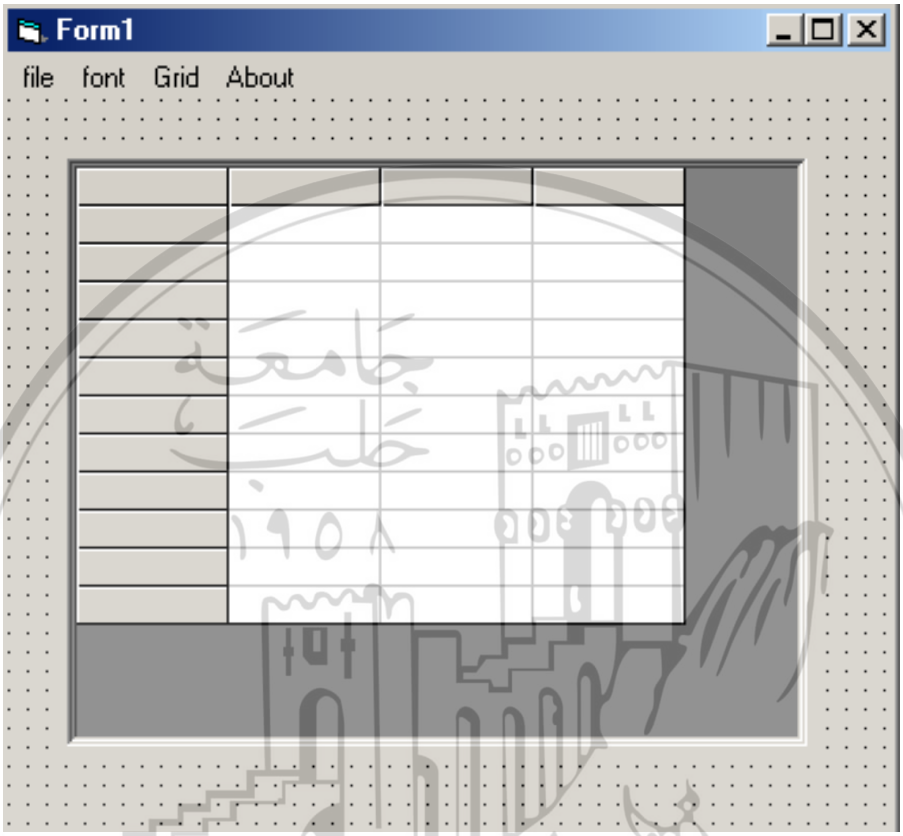


```
Private Sub Form_Load()
Combo1.AddItem "All Files (*.*)"
Combo1.AddItem "Text Files (*.Txt)"
Combo1.AddItem "Doc Files (*.Doc)"
Combo1.ListIndex = 0
End Sub

Private Sub Drive1_Change()
Dir1.Path = Drive1.Drive
End Sub

Private Sub Combo1_Click()
Select Case Combo1.ListIndex
Case 0
 File1.Pattern = "*.*"
Case 1
 File1.Pattern = "*.Txt"
Case 2
 File1.Pattern = "*.Doc"
End Select
End Sub

Private Sub Dir1_Change()
File1.Path = Dir1.Path
End Sub

Private Sub File1_Click()
Text1.Text = File1.FileName
End Sub

Private Sub Command1_Click()
Dir1.Path = "C:\windows"
End Sub

Private Sub Command2_Click()
End
End Sub
```

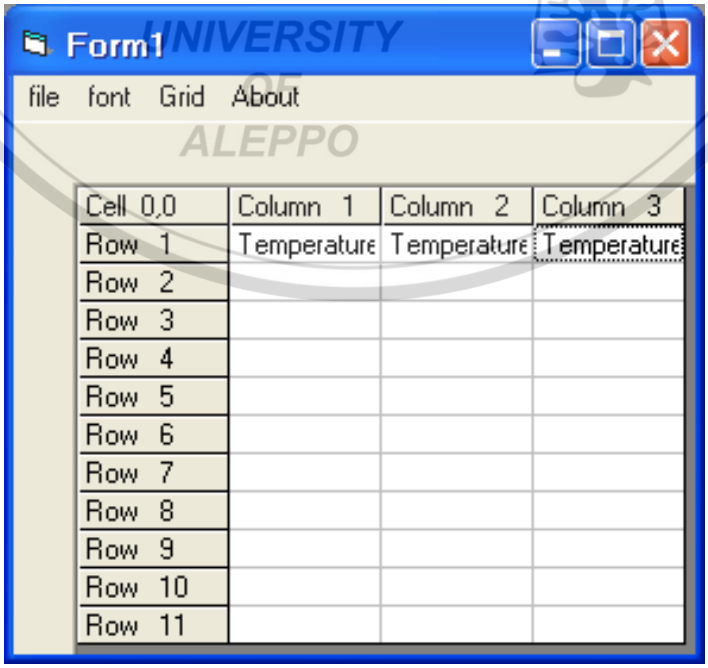
التمرين (١٠) – أداة الشبكة $Ms - Flex\ Grid$:

المطلوب تصميم برنامجاً لإظهار يحتوي أداة الشبكة واجهته المرئية كما يلي:

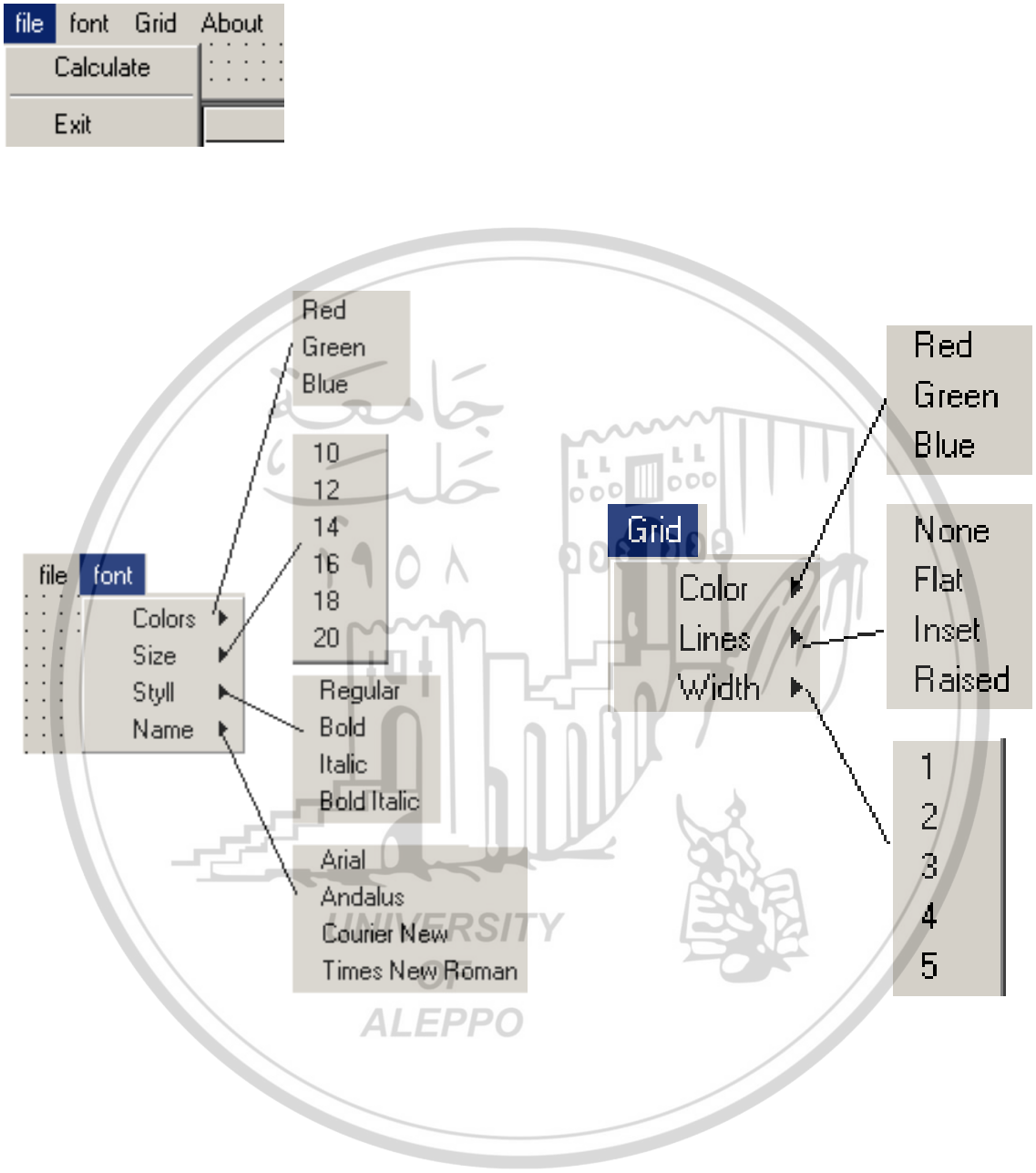
⇦ **تظهر الواجهة في المرحلة المرئية كما في الشكل:**



⇦ **تظهر الواجهة في المرحلة التنفيذية كما في الشكل:**



⇦ **تظهر القوائم الفرعية في المرحلة المرئية كما في الشكل:**



المرحلة الكودية:

```
Private Sub Form_Load( )
Grid1.Rows = 12
Grid1.Cols = 4
Grid1.Row = 0: Grid1.Col = 0: Grid1.Text = "Cell 0,0"
For i = 1 To Grid1.Rows - 1
   Grid1.TextMatrix(i, 0) = "Row " + Str(i)
Next
For j = 1 To Grid1.Cols - 1
   Grid1.TextMatrix(0, j) = "Column " + Str(j)
Next
Grid1.Row = 1: Grid1.Col = 1: Grid1.Text = "Temperature C "
Grid1.Row = 1: Grid1.Col = 2: Grid1.Text = "Temperature K "
Grid1.Row = 1: Grid1.Col = 3: Grid1.Text = "Temperature F "
End Sub

Private Sub Calc_Click( )
tempc = 0
For i = 2 To 11
  tempc = tempc + 5
  Grid1.Col = 1: Grid1.Row = i
  Grid1.Text = tempc
  Grid1.Col = 2: Grid1.Row = i
  tempk = tempc + 273
  Grid1.Text = tempk
  Grid1.Col = 3: Grid1.Row = i
  tempf = (9 / 5) * tempc + 32
  Grid1.Text = tempf
Next
End Sub

Private Sub exit_Click( )
End
End Sub

Private Sub fRed_Click( )
Grid1.CellForeColor = RGB(255, 0, 0)
End Sub
```

Private Sub fGreen_Click( )
$Grid1.CellForeColor = RGB(0, 255, 0)$
End Sub

Private Sub fBlue_Click( )
$Grid1.CellForeColor = RGB(0, 0, 255)$
End Sub

Private Sub s10_Click()
$Grid1.CellFontSize = 10$
End Sub

Private Sub s12_Click( )
$Grid1.CellFontSize = 12$
End Sub

Private Sub s14_Click( )
$Grid1.CellFontSize = 14$
End Sub

Private Sub s16_Click( )
$Grid1.CellFontSize = 16$
End Sub

Private Sub s18_Click( )
$Grid1.CellFontSize = 18$
End Sub

Private Sub s20_Click( )
$Grid1.CellFontSize = 20$
End Sub

Private Sub FRegular _Click( )
$Grid1.CellFontRegular = True$
End Sub

Private Sub FBold _Click( )
$Grid1.CellFontBold = True$

End Sub

Private Sub FItalic _Click( )
$Grid1.CellFontItalic = True$
End Sub

Private Sub FBoldItalic _Click( )
$Grid1.CellFontBoldItalic = True$
End Sub

Private Sub fArial _Click( )
$Grid1.CellFontName = "Arial"$
End Sub

Private Sub fAndalus _Click( )
$Grid1.CellFontName = "Andalus"$
End Sub

Private Sub fcnew _Click( )
$Grid1.CellFontName = "New Courier"$
End Sub

Private Sub ftnroman _Click( )
$Grid1.CellFontName = "Times New Roman"$
End Sub

Private Sub Gred _Click( )
$Grid1.GridColor = RGB(255,0,0)$
End Sub

Private Sub GGreen _Click( )
$Grid1.GridColor = RGB(0,255,0)$
End Sub

Private Sub GBlue _Click( )
$Grid1.GridColor = RGB(0,0,255)$
End Sub

Private Sub lflat _Click( )

```
Grid1.GridLines = flexGridFlat
End Sub

Private Sub linset _Click( )
Grid1.GridLines = flexGridInset
End Sub

Private Sub lnone _Click( )
Grid1.GridLines = flexGridNone
End Sub

Private Sub lraised _Click( )
Grid1.GridLines = flexGridRaised
End Sub

Private Sub w1_Click( )
Grid1.GridLineWidth = 1
End Sub

Private Sub w2_Click( )
Grid1.GridLineWidth = 2
End Sub

Private Sub w3_Click( )
Grid1.GridLineWidth = 3
End Sub

Private Sub w4_Click( )
Grid1.GridLineWidth = 4
End Sub

Private Sub w5_Click( )
Grid1.GridLineWidth = 5
End Sub

Private Sub About_Click( )
MsgBox ("This Program Designed By Dr Hammad")
End Sub
```

المفاهيم والمصطلحات الهندسية والمراجع العلمية
المصطلحات العلمية

A	
ABS – Absolute	تابع القيمة المطلقة
Absolute References	المرجعية المطلقة
Accept	يوافق
ACOS	تابع عكس التجيب
Activate	تنشيط
Add	أضف
Add Constraint	إضافة قيد
Add Trendline	إضافة خط اتجاه
Adding An Existing Form	إضافة نموذج موجود مسبقاً
Addition	عملية الجمع
Additivity	قابلة الإضافة
Address	عنوان
advanced metafile	الملفات المحسنة
Air	هواء
Algorithm	خوارزمية
Alignment	خاصية الضبط أو المحاذاة
AND	تابع التقاطع (وجوب تحقق عدة شروط)
AND	عملية AND المنطقية
Angle	زاوية

Answer	جواب
Apostrophe	فاصلة علوية
Appearance	خاصية المظهر
Appearance 3D Appearance	خاصية المظهر ثلاثية الأبعاد
Apples	تفاح
Approximation Method	طريقة التقريب
Area	مساحة
Arithmetic	حسابي
Arithmetic operand	معامل حسابي
Arithmetic Operators	معاملات حسابية
Arrange Cascade	الترتيب المتتالي
Arrange Windows	ترتيب النوافذ
Array	مصفوفة
Array element	عنصر مصفوفة
Arrays	المصفوفات
ASIN	تابع عكس الجيب
Assembly language	لغة الآلة
Assignment operators	عمليات الإسناد
Assistant	المساعد
ATAN	تابع عكس الظل
AVERAGE	متوسط

B

Basic Concept	مفاهيم عامة
Binary	ثنائي
Binary Search	البحث الثنائي
Bit	خانة ثنائية
Bitmaps Files	ملفات الصور النقطية
Block	كتلة
Book	مصنف
Boundary Condition	الشروط الحدية
Byte	بايت

C

Calculation	حسابات
Calculation Order	أولوية العمليات الحسابية
Call	استدعاء
Capital	كبير
Capital Letters	حروف الكبيرة
Caption	خاصية الاسم المرئي
Central	مركزي
Change	تعديل – تغيير
CHAR – Character	تابع المحرف
Characters	محارف
Check Box	صندوق الاختيار
Check Marks	علامات الاختيار

Checked	خاصية التحديد أو الاختيار
Circle Method	تعليمة رسم الدائرة
Classes	الفئات
Click Event	حجث النقر
Clicking a Directory Item	النقر على الأدلة
Clicking And Releasing Buttons	ضغط و تحرير الأزرار
Clicking and Releasing Keys Events	أحداث لوحة المفاتيح الناتجة ضغط أو تحرير مفتاح ما من لوحة المفاتيح
Cls (Clean Screen)	طريقة المسح
CODE	تابع الأعداد في نظام الآسكي
Code	شيفرة
Code Window	نافذة محرر الأكواد
Coefficient	معامل
Colors in VB	الألوان في الفيجوال البيزك
Column	عمود
Combo Box	الخانة المركبة
Comma	فاصلة
Command Button	زر الأوامر
Comments	تعليقات
Common Dialog Boxes	مربعات الحوار الشائعة
Comparison	مقارنة
Comparison Operators	معاملات المقارنة

Compiler	مترجم
Computer organization	بنية الحاسب
CONCATENATE	تابع السلاسل النصية
Concatenately	سلاسلي
Concatenation	إشارة الدمج (الربط)
Conditional operator	العملية الشرطية
Conditions	شروط
Constant	ثابت
Constants	الثوابت
Constraint	قيد
constructor	باني
Control Instructions	تعليمات التحكم
Control Instructions and Control Tools	تعليمات التحكم وأدوات التحكم
Control structures	بنى التحكم
Control Tool Box	مربع الأدوات (أدوات التحكم)
Control Tools	أدوات التحكم
Control unit	وحدة تحكم
Converge	غير منته
Convergence	درجة التقارب
Conversion Factor	عامل التحويل
Coordinate System	المنظومة الإحداثية
Correct	يصحح

Correction	تصحيح
Correction Factor	عامل التصحيح
Correlation Factor	معامل الارتباط
COS	تابع التجيب
Covariance	تباين
Criteria	شرط – معيار
Critical	حرج – حدي
Current	حالي
Custom Dialog Boxes	مربعات الحوار المخصصة

D

Dash – Dot – Dot Line	خط مكون من شُرط ونقطتين على التتابع
Dash - Dot Line	خط مكون من نقط وشُرط متتابعة
Dashed Line	خط مكون من شُرط متتابعة
Data	قيم
Data	أداة البيانات
Data Types	أنواع البيانات
Data View	نافذة عرض البيانات
Data-Type Suffixes	الرموز التي تدل على نوع المعطيات
DATE	تابع التاريخ
DAY	تابع اليوم
Deactivate	فقدان التنشيط
Decimal	عشري

Decimals	الأجزاء العشرية
Decrement operators	عمليات النقصان بواحد
Default	افتراضي
DEGREES	تابع الدرجات
Delete	حذف
Derivative	مشتق – اشتقاقي
Diameter	القطر
Difference Approximation	التقريب التفاضلي
Difference Equations	المعادلات التفاضلية
Digit	رقم
Digital	رقمي
Dimentions	خصائص الأبعاد
Directory List Box	أداة التحكم بالمجلدات
Divisibility	قابلية القسمة
Divisible	قابل للقسمة
Division	عملية القسمة
Divisor	مقسوم عليه
Dotted Line	خط منقّط
Double	مضاعف
Double Click Event	حدث النقر المزدوج
Drag Drop Event	حدث إفلات أو إلقاء
Drag Over Event	حدث سحب فوق

Draw Mode	خاصية ظهور الرسوم والمخططات
Draw Style	خاصية تحدد شكل الخطوط
Draw Width	خاصية تحدد عرض خط الرسم
Drawing in VB	الرسم في الفيجوال البيزك
Drawing Properties	خصائص الرسم
Drive List Box	أداة التحكم بالسواقات
Dynamic Array	المصفوفة الديناميكية

E

Editing the Text Box	تعديل محتويات مربع النص
Effective	فعال
Efficiency	فعالية
Elements	عناصر
Enabled	خاصية التفعيل والتعطيل
Encountered	تعارض
Equal	يساوي لـ – مساوٍ لـ
Equal to	يساوي
Equation	مساواة
Erase Array	مسح محتويات مصفوفة
Error	خطأ
Estimate	يقدر – يخمن
EVEN	تابع الأعداد الزوجية
Event	الحدث

Event Driven Programming	البرمجة المسيَّرة بالأحداث
Event Function	تابع حدثي
Executable statement	تعليمة تنفيذية
Execution	تنفيذ
Existing Project	مشروع موجود مسبقاً
EXP - EXPONENTIAL	تابع الرفع إلى قوة (الأساس العدد النبري)
EXPONENTIAL	عملية الرفع إلى قوة
Exponentiation	الرفع إلى قوة
Expression	تعبير
Expressions	تعابير

F

FACT	تابع العاملي
Feasible	ممكن
Feasible Solution	حل ممكن
File	ملف
File – System Controls	عناصر تحكم الملفات
File List Box	أداة التحكم بالملفات
FIND	تابع البحث عن السلاسل النصية
Find – Text	النص المراد البحث عنه
Finding a Directory's Relative Position	إيجاد الأدلة بالنسبة للدليل الحالي
Fire Event	تفجير الحدث
FIXED	تابع التقريب

Flat Appearance	خاصية المظهر المسطحة
Floating point Division	القسمة العادية (العائمة)
FLOOR	تابع المضاعفات
Flowchart	المخطط الصندوقي
Focus Events	أحداث التركيز الناتجة عن تنشيط أداة ما أو فقد تنشيط هذه الأداة
Font	الخط
Form	الإطار
Form Layout Window	نافذة توضع الإطار
Format	تعليمة الواصفات
Formatting Toolbar	شريط أدوات التنسيق
Forms	النماذج
Forward	مستقيم
Frame	الإطار
Function	تابع – دالة
Function overloading	التحميل الزائد للتوابع
Function Scope	نوع تعليق التابع
Function Type	نوع التابع
Functions	التوابع
FALSE	خطأ – خاطئ – نتيجة خاطئة
FALSE	عملية FALSE المنطقية

G

General	عام
General Declaration	قسم التصاريح العامة
Geometry	الشكل الهندسي
Global	شامل
Got Focus Event	حدث حصل تركيز
Graphic	رسم بياني
Graphics Methods	طرق الرسم
Greater than	أكبر من
Greater than or Equal to	أكبر من أو يساوي لـ
H	
Header files	ملفات رأسية
Heap	مكدس
Height	الارتفاع
Help	مساعدة
hiding	إخفاء
Horizontal Arrange	الترتيب الأفقي
Horizontal Scroll Bar	شريط التمرير الأفقي
Hot	ساخن
HOUR	تابع الساعة
I	
Icon Files	ملفات الأيقونات
Identifying Individual Directories	ترتيب عناصر الدليل

Image Control	أداة عرض الصور
Increase operators	عمليات الزيادة بواحد
Index	مؤشر دليل أس
Inequality	متراجحة
Initial Condition	الشروط الداخلية
Initial Temperature	درجة الحرارة الداخلية
Initialization	قيمة ابتدائية
Initialize	حدث تسجيل بيانات النافذة
Inline function	التوابع السطرية
Input and Output Instructions	تعليمات الإدخال والإخراج
Input and Output Instructions and Tools	تعليمات وأدوات الإدخال والإخراج
InputBox	تابع الإدخال
Inside Solid Line	خط مصمت من الداخل
INT	تابع القيم الصحيحة
Integer	صحيح
Integer Division	القسمة الصحيحة
Integer Programming	البرمجة الصحيحة
International	دولي
Interval	المسافة أو الفاصل الزمني
J	
Joul	جول
K	

Key Board Events	أحداث لوحة المفاتيح
Key Down Event	حدث ضغط مفتاح لأسفل
Key Press Event	حدث ضغط مفتاح
Key Up Event	حدث تحرير مفتاح
Known	معروف

L

Label	اللافتة
LARGE	تابع القيم الأعظمية
Left	البعد الأيسر أو الحافة اليسرى
Length	طول
Less than	أصغر من
Less than or Equal to	أصغر من أو يساوي لـ
Letter	حرف
Limit	حد
Line	أداة الخط
Line Method	تعليمة رسم خط
Linear	خطي
Linear Programming	البرمجة الخطية
Linear Regression	تراجع خطي
Linear Search	البحث الخطّي
Linearity	خطية
Liner	خطي

List Box	صندوق أو مربع اللائحة
LN	تابع اللوغاريتم الطبيعي
Load	تحميل
Load Model	تحميل موديل
Local	محلي
LOG	تابع لوغاريتم عدد للأساس المحدد
LOG10	تابع اللوغاريتم العشري
Logic	منطق
Logical	منطقي
Logical operators	العمليات المنطقية
Lost Focus Event	حدث فقدان التركيز
LOWER	تابع الحروف الصغيرة
LPRINT	تعليمة الإظهار أو الطباعة على الطابعة

M

Mathematical	رياضي
Mathematical Expressions	تعابير رياضية
Mathematical Operators	المعاملات الرياضية
Maximized	خاصيّة الحجم الأعظمي
Memory	ذاكرة
Menu Bar	شريط القوائم
Menu Editor	محرر القوائم
metafile	الملفات المتحولة

Method	طريقة
Methods	الوظائف
Minimized	خاصيّة الحجم الأصغري
Minus	علامة السالب
MINUTE	تابع الدقيقة
Modeling	بناء الموديلات
Module	ملفات البرمجة
MONTH	تابع الشهر
Mouse Down Event	حدث ضغط زر الماوس للأسفل
Mouse Events	أحداث الماوس
Mouse Move Event	حدث تحريك الماوس
Mouse Pointer	خاصيّة مؤشر الماوس
Mouse Up Event	حدث تحرير زر الماوس
Moving the Mouse Events	الأحداث الناتجة عن تحريك هذه الماوس
MSDN (Microsoft Developer Network)	
Ms-Flex Grid Control	أداة الجدول المرن
MsgBox	تابع الإظهار
Multi Document Interface – MDI	واجهة المستخدم متعددة المستندات
Multiple	الضرب
Multiple	متعدد
Multiple Projects	المشاريع المُتعدّدة
Multiplication	عملية الضرب

Multiplicity	قابلة الضرب
MultiSelect	متعدد الاختيار
N	
Name	خاصية الاسم الكودي
Name Box	مربع الاسم
Name of variable	اسم المتحول
Negation	إشارة السالب
Negative	سلبي
New Project	مشروع جديد
NO - Commas	بدون فاصلة
Non – Linear	غير خطي
Non – Negative	عدم السلبية
Normal	خاصيّة الحجم العادي
NOT	عملية NOT المنطقية
Not Equal to	لا يساوي لـ
NOW	تابع الآن
Number	عدد
Numeric	عددي
Numeric Functions	التوابع العددية
Numerical	عددي
Numerical Method for Solving Systems of Equation	الطريقة العددية لحل جملة المعادلات
O	

Objective	حيادي – موضوعي
objects	أغراض
Objects	كائنات برمجية
ODD	تابع الأعداد الفردية
OLE	أداة التحكم
One dimensional array	مصفوفة أحادية البعد
One-Dimensional	أحادي البعد
Open System	نظام مفتوح
Operation	عمليات
Operation Research	بحوث العمليات
Operator	معامل
Operator Precedence	أفضلية المعاملات
Optimal	مثالي
Optimality	المثالية
Optimization	دراسة النهايات الحدية
Optimum	القيمة الحدية أو المُثلى
Option Button	زر الاختيار
OR	تابع الجمع (كفاية تحقق أحد شروط)
OR	عملية OR المنطقية
Order	نظام – أولوية
Orientation	التوجيه

P

Parallel	متوازي
Parameters	بارامترات
Parentheses	الأقواس
Passing arrays	تمرير المصفوفات
Percent	النسبة المئوية
PI	تابع القيمة (π٣,١٤=)
Picture	خاصية تحميل الصورة
Picture Box	مربع الصورة
Pixel Set	تعليمة رسم نقطة
Placement	خصائص المكان
Plus	علامة الموجب
Point	تعليمة لون نقطة
Pointer	المؤشر
Pointer operator	عمليات التأشير
POWER	تابع القوة – عدد مرفوع إلى أس محدد
Precedence	أفضلية
Precision	دقة
Predefined Dialog Boxes	مربعات الحوار مسبقة التعريف
Prediction	تنبؤ
Print	تعليمة الطباعة
Private Scope	إجراء في الملف
Problem	مشكلة – خطأ

Procedures	الإجراءات
Process	عملية
PRODUCT	تابع الجداء
Programming	البرمجة
Project	مشروع
Project Explorer	نافذة مستكشف المشروع
Project Form	نافذة إطار المشروع
Project window	نافذة المشروع
PROPER	تابع الحروف الكبيرة في بداية الكلمات
Properties	الخصائص
Properties	خصائص
Properties Window	نافذة الخصائص (خصائص الأدوات)
Property	خاصية
Property Function	تابع احتمالي
Proportional	نسبي
Proteins	البروتينات
Public Scope	إجراء في جميع ملف المشروع
Pull Down Menus	القوائم المنسدلة
Pull Down Menus Bar	شريط القوائم المنسدلة

Q

Quality	النوعية

R

RADIANS	تابع الراديان
Radius	نصف القطر
RAND	تابع العشوائية
Random	عشوائي
Random number	الأعداد العشوائية
RANDOMIZE	العشوائية
Range	نطاق – مجال
Ratio	نسبة
Ready	جاهز
Real	حقيقي
Real Numbers	أعداد حقيقية
Recent Project	مشروع مفتوح مؤخراً
Recursion	العودية
Reference	مرجعي
Reference Operators	المعاملات المرجعية
References	المراجع
Relative References	المرجعية النسبية
Rem	تعليق أو ملاحظة
Remark	تعليق أو ملاحظة
Repetition Instructions and Loops	تعليمات وحلقات التكرار
Repetition structure	بنية التكرار
REPLACE	تابع استبدال السلاسل النصية

Reports	تقارير
REPT - Repeat	تابع تكرار السلاسل النصية
Research	بحوث
Reserved Words	الكلمات المحجوزة
Reset	مسح وإعادة تعريف
Results	نتائج
Return	تعليمة العودة
RIGHT	تابع اقتطاع السلسلة من اليمين
Right To Left	خاصية اليمين لليسار
ROUND	تابع التقريب
Row	صف

S

Scale	المقياس
Scale Height	ارتفاع المقياس
Scale Mode	نوع المقياس
Scale Width	عرض المقياس
Scroll Bar	شريط التمرير (السحاب)
SEARCH	تابع البحث عن مواقع السلاسل النصية
Search	يبحث
Search	البحث
Searching Arrays	عملية البحث في المصفوفات
SECOND	تابع الثواني

Section	مقطع
Selection	اختيار
Setting the Current Directory	اختيار الدليل الحالي
Shape	أداة الأشكال
Shape	شكل
Shape Control	أداة الأشكال
Shape Factor	عامل الشكل
Shortcut Keys	مفاتيح الاختزال
Shortcut Keys	مفاتيح الإختزال
Side	جانب
SIGN	تابع الإشارة
Significance	مضاعفات
Similarity	التشابه
SIN	تابع الجيب
Situation	حالة
SMALL	تابع القيم الأصغرية
Small Letters	حروف الصغيرة
Solid Line	خط مصمت
Sort	فرز
Sorting arrays	مصفوفات الفرز
Source	مصدر
Special Indexes	الرموز الخاصّة

Spherical	كروي
SQRT	تابع الجذر التربيعي
Standard Toolbar	شريط الأدوات القياسي
Start	بداية – يبدأ
Statistics	احصاء
Status Bar	شريط الحالة
Straight Line	خط مستقيم
Stretch	الخاصية الإمتداد أو ملائمة الحجم
String	سلسلة
Structured programming	البرمجة الهيكلية
Sub Function	تابع فرعي
SUBSTITUTE	تابع استبدال ضمن السلاسل النصية
Subtraction	عملية الطرح
Subtraction	الطرح
SUM	تابع دالة الجمع
SUMIF	تابع دالة الجمع الشرطي
Supposition	افتراض
Surface	سطح
Surface Temperature	درجة الحرارة السطحية
Surrounded	محاط
Symmetry	التناظر

T

TAN –Tangent	تابع الظل
Tangent	ظل – مماس
TEXT	سلسلة نصية
Text Box	مربع النص
Text Concatenation Operators	معاملات السلاسل النصية
Text Editor	محرر النصوص
Text Functions	توابع السلاسل النصية
The Directory List Box	مربع اختيار الدليل
The Drive List Box	مربع اختيار السواقة
The File List Box	مربع اختيار الملفات
Thickness	سماكة
TIME	تابع الوقت
Time	الزمن
Timer	المؤقت الزمني
Title Bar	شريط الاسم أو العنوان
Toleranece	تفاوت
Tool Bars	أشرطة الأدوات
Top	البعد العلوي أو الحافة العلوية
Transfer	ينقل
Transparent Line	خط شفاف
Trendline	خط اتجاه
Type	نوع – نموذج – موديل

Type statement	تعليمة النوع
TRUE	حقيقة – صحيح – نتيجة صحيحة
TRUE	عملية TRUE المنطقية
U	
Unary	أحادي
Unload	تفريغ التحميل
Unstable	غير ثابت
UPPER	تابع الحروف الكبيرة
Use of Parentheses	استخدام الأقواس
Using File-System Controls Together	استخدام عناصر تحكم الملفات مع بعض
V	
Value	قيمة
Variable	متحول
Variable	متغير
Varies Linearly	تغير بشكل خطي
Vector	شعاع
Vertical Arrange	الترتيب العمودي
Vertical Scroll Bar	شريط التمرير العمودي
Visible	خاصية المرئية
VISUAL BASIC CONTROL TOOLS	أدوات التحكم في فيجوال بيزك
Visual Basic work area	منطقة العمل في الفيجوال بيزك
W	

Width	العرض
Window State	خاصيّة حالة النافذة
Working with File Attributes	العمل مع خاصية الفرز في الملفات
X	
X- Rays	أشعة إكس
XOR	عملية XOR المنطقية
Y	
YEAR	تابع العام

المراجع العلمية
Bibliography

المراجع العربية

١ – جقل، أحمد. حماد، محمد – ٢٠٠٩. البرمجة والحاســـبات /١/ – مديرية الكتب والمطبوعات الجامعية – جامعة حلب، ٤٨٨ صفحة.

٢– ســـلام ســـلوم. إلهام بلان. برمجة "١"، مديرية الكتب والمطبوعات الجامعية، جامعة البعث 2600-2005م.

٢– زياد عبد الكريم القاضي. عودة النشوان. المرجع في كويك بيسك, دار صفاء للطباعة والنشر، عمان – الأردن، الطبعة الأولى 1997.

٣– أيمن عودة. نوال منصور. دليل لغة والعلوم، الطبعة الأولى

Visual Basic.1998 ، دار شعاع للنشر

٤– كويك بيسك تأليف الدكتور ستيفن تاميروف – ترجمة د. سرور علي ابراهيم سرور – الدار الدولية للنشر والتوزيع.

٥– ســـهيل خواتمي. حمدو النجار. يحيى النجار. أيمن نعّال. محمد حماد. المعلوماتية لغير المختصين، مديرية الكتب والمطبوعات الجامعية، جامعة حلب ٢٠٠٨م.

٦– أمل منا. برمجة (١) "لغة بيســـك المرئية للمهندســـين"، مديرية الكتب والمطبوعات الجامعية، جامعة حلب 2003م.

٧– نشــاوي محمد أســعد وأحمد وضــاح عطّار، موســوعة مبرمجي فيجوال بيزيك٦، دار الكتب العلمية للنشر والتوزيع، 2000.

٨– محاضرات الدكتور محمد حماد، كلية الهندسة الميكانيكية، جامعة حلب.

٩– محاضرات الدكتور أحمد جقل، كلية الهندسة الميكانيكية، جامعة حلب.

١٠– مواقع ومقالات ومجلات علمية من الانترنيت.

المراجع الأجنبية

1. Deitel, H. M, and P, J, Deitel., C++ How to Progam (Second Edition), Englewood Cliffs, NJ: Prentice-Hall-1998.
2. Weiskamp, K., and B. Flaing, The complete C++ Primer, Second Edition, Academic Press,1992.
3. Ladd, S. R., C++ Components and Algorithm, M&T Books,1993.
4. Deitel, H. M., Operating Systems, Second Edition. Reading, MA: Addison-Wealey,1990.
5. Borland International, the world of C++: The Fastest Way to Become aC++ Programmer, Scotts, CA: Borland International, Inc., -1991.
6. Deitel & Deitel T. R. Nieto, Visual Basic 6 "How to program, International Edition, Prentice-Hall, 2004.

تم تدقيق الكتاب علمياً من قبل:

الدكتور	الدكتور	الدكتور
محمد خليل الصباغ	أحمد جقل	محمد يوسف الهاشم

تم تدقيق الكتاب لغوياً من قبل:

الدكتور

أحمد دواليبي

جامعة
حلب
١٩٥٨
UNIVERSITY
OF
ALEPPO