المملكة العربية السعودية
المؤسسة العامة للتدريب المهني والتقني
الإدارة العامة للتدريب المهني والأهلي
معهد التميز الكندي العالي للتدريب

اساسيات برمجة الحاسب

معهد التميز الكندي

اساسيات برمجة الحاسب الالي

برمج ١١١

تخصص برمجيات مستوى ثالث

الدرس الأول

مبادئ وأساسيات البرمجة:

قبل البدء بتعلم أي من لغات البرمجة، أو الدخول إلى عالم البرمجة والخوارزميات، من الضروري جدا التعرف على مبادئ البرمجة بشكل مجرد ومستقل عن لغات البرمجة. مبادئ البرمجة هي عرض للأدوات المنطقية التي تم تطويرها واستخدامها في لغات البرمجة، حيث تشترك معظم أو جميع لغات البرمجة في هذه الأدوات ولكن تقوم كل لغة بالتعبير عن هذه الأدوات بطريقتها الخاصة.

هذه الأدوات عموما صممت وطورت لتكون البنية الأساسية للغات متوسطة وعالية المستوى.

أهمية مبادئ وأساسيات البرمجة :

في مادة أساسيات البرمجة يتم التعرف على لبنات بناء البرنامج والتحليل المنطقي للمسائل وتجزئتها ومحاولة تمثيل كل جزء بما يقابله من الأدوات البرمجية المتاحة.

إذا كنت في بداية الطريق لعلوم الكمبيوتر بشكل عام، والبرمجة بشكل خاص أو إذا كنت تريد التعرف على مفاهيم البرمجة دون التعمق بلغات البرمجة؛ فهذا هو المكان المناسب لتبدأ منه.

مصطلحات برمجية:

المبرمج :(computer programmer)هو الذي يكتب البرنامج بعد أن يفهم المشكلة ويقترح الحل وينفذه لحل هذه المشكلة.

ويجب أن يكون البرنامج صحيحاً وواضحاً وليس فيه غموض.

البرمجيات : (Software) هي التي تسهل للمستخدم استخدام المكونات المادية (Hardware) بكفاءة وراحة ويمكن تقسيم البرمجيات إلى : برامج التشغيل. برامج التطبيقات. لغات البرمجة.

برامج التشغيل : (Operating System) هي عبارة عن برامج تقوم بدور الوسيط بين المستخدم والمكونات المادية.

من وظائفها : تمكن المستخدم من استخدام المكونات المادية للحاسب بكفاءة وبراحة. تساعد المستخدم في إنشاء نظام الملفات وغيرها. امثلة لبرامج التشغيل : من برامج التشغيل ما يصلح للعمل في الشبكات مثل , Unix , Windows ومنها الذي يستخدم مع الحاسب فقط مثل. Dos

برامج التطبيقات : (Application Programs) هي برامج تساعد في إنشاء كثير من التطبيقات، مثل: إنشاء قاعدة البيانات والرسم باستخدام الحاسب وغيرها.

لغات البرمجة : (Programming Languages) هي التي تستخدم في بناء البرامج المختلفة وهي تتراوح من اللغات التي تتعامل مباشرة مع المكونات المادية للحاسب والأخرى التي

تتطلب تحويلها من صورتها التي تكتب بها إلى صورة أخرى يستطيع الحاسب التعامل معها. تقسم لغات البرمجة إلى: لغة الآلة. لغات التجميع' .لغات المستوى العالي.

لغة الآلة : (Machine Languages) هي اللغة الوحيدة التي يفهمها الحاسب ويستطيع التعامل معها، وهي تعتبر لغة خاصة لكل حاسب وقد تختلف من حاسب لآخر لأنها تعتمد على المكونات المادية للحاسب نفسه. لغة الآلة تتكون من مجموعة أرقام من بين ٠,١ التي تعطي تعليمات للحاسب للقيام بمعظم العمليات الأساسية واحدة بعد الأخر لغة الآلة من اللغات الصعبة في التعلم للإنسان حتى بالنسبة للمبرمجين لأنها مجموعة من الأرقام ٠,١ فقط. للتغلب على هذه الصعوبة تم اقتراح لغة أخرى تعتمد على استخدام اختصارات معبرة من اللغة الانجليزية للتعبير عن العمليات الأولية التي يقوم بها الحاسب وهذه اللغة هي لغة التجميع.

لغة التجميع : (Assembly Languages) هي لغة تستخدم اختصارات معبرة من اللغة الانجليزية لتعبر بها عن العمليات الأولية التي يقوم بها الحاسب ، مثل إضافة (Add) وحفظ (Store)و طرح (Sub)وغيرها.

نظراً لان هذه اللغة تستخدم كلمات مختصرة من اللغة الانجليزية فإنها تحتاج محولاً لكي يحولها إلى لغة الآلة وهو ما يسمى المجمع (assembler) الذي يقوم بتحويل لغة التجميع إلى لغة الآلة كي يفهمها الحاسب ويستطيع تنفيذها.

ولكن بالرغم من كل ذلك ولكن مازال هناك توجد مشقة عند حل ابسط المسائل لان ذلك يتطلب معرفة وكتابة العديد من التعليمات، وهذا ما دفع المبرمجين للتفكير في لغات أخرى تقلل المجهود الكبير اللازم لكتابة الكثير من التعليمات فكانت لغات البرمجة ذات المستوى العالي.

لغة البرمجة ذات المستوى العالي : (High level Languages) هذه اللغات كتبت بحيث تستخدم بعض الكلمات الانجليزية العادية بنفس معانيها حيث يقوم كل أمر منها بتنفيذ العديد من الواجبات ، وهذه اللغة كسابقاتها تحتاج الى **(مترجمات compilers)** التي تقوم بتحويل التعليمات (الأوامر) إلى لغة الآلة ، وهذه اللغات تستخدم العلاقات والعوامل الرياضية المتعارف عليها مثال:

SUM = A+B+C

مفهوم البرمجة المرئية

من المعروف أن اللغات التقليدية (التي لا تعتمد أسلوب البرمجة المرئية) تستخدم النصوص لبرمجة الحاسوب، أما لغات البرمجة المرئية يمكن أن نستخدم الرسم والرسومات وواجهة رسومية

(Graphical User Interface : GUI) لإصدار تعليمات للحاسوب، ومن التوضيحات المنطقية لهذا الأسلوب، هو ان الإنسان يشاهد ما حوله من الظواهر المختلفة على شكل صور، ثم تتم ترجمة ما يرى الى نصوص معبرة عنها، وبالتالي، لماذا لا تعطى هذه الصور كتعليمات للحاسوب بدلا من المرور بمرحلة نقلها الى نص. كما أن كثيرا من التطبيقات العلمية وغيرها، وبرامج التعلم التفاعلية تحتاج الى البيئة المرئية لتقوم بدورها بشكل أكثر فاعلية.

إن هذا النوع من البرمجة يوسع دائرة استخدام الحاسوب، ويزيد من قوة وفاعلية البرمجة ويطورها، وقد شهدت العقود الثلاثة الأخيرة تطورات واسعة في هذا الميدان، ويمكن القول ان حقل البرمجة المرئية نما وترعرع نتيجة تمازج رائع بين حقول ثلاثة هي لغات البرمجة (Programming Languages)، الرسم بالحاسوب (Computer Graphics)، وتفاعل الإنسان مع الحاسوب (HCI).

ولنستعرض التعريفات التالية وصولا الى تعريف لغة البرمجة المرئية:

- أيقونة: كينون دو مدلولين، المدلول المنطقي (Logical) وهو المعنى المقصود منه، والمدلول الفيزيائي (Physical) وهو الصورة التي تمثله.
- نظام أيقوني: مجموعة تركيبية من أيقونات ذات علاقة مع بعضها البعض.

- جملة أيقونية: تنظيم من الأيقونات من نظام أيقوني.
- لغة البرمجة المرئية: مجموعة من الجمل الايقونية بنيت بقواعدية (Syntax) والتي يرتبط معها تحليل لهذه الجمل لتحديد تركيبها القواعدي، ومعنى (Semantic) والتي يرتبط معها تحليل لهذه الجمل لتحديد المعنى الذي يقصد منها.

وفي لغات البرمجة المرئية، يمكن ان نميز بين نوعين من الأيقونات، أيقونات العمليات (Process Icons) والذي تدل على حسابات، الايقونات الكينونية (Object Icons)، والتي تتكون بدورها من نوعين هما البسيطة (Elementary)، والتي تمثل كينونات اساسية والمركبة (Composite)، والتي تمثل كينونات مركبة، يتم تكوينها من عدة ايقونات بسيطة باستخدام عمليات محددة.

ان البرمجة المرئية هي عبارة عن اسلوب حديث نسبيا للبرمجة، تستخدم فيه برامج مساعدة لتصميم واجهة استخدام رسومية (الازرار والنصوص...) وربطها بالشيفرة البرمجية (Code)، وتسمى هذه البرامج المساعدة ببيئة التطوير المتكاملة (Integrated Development Environment: IDE)، ويستخدم هذا النوع من البرمجة تعبيرات مرئية (Visual Expressions) في عملية البرمجة، او تعالج معلومات مرئية، او قد تدعم التفاعل المرئي بين المستخدم وجهاز الحاسوب.

في السابق كانت البرامج تستخدم ما يسمى بسطر الاوامر(Line Command)، فكان يظهر البرنامج في صورة نصوص من عدة أسطر، ويمكن القول ان لغات البرمجة التي تعتمد للنصوص فقط تصعب على العديد من المستخدمين، أما الان ومع الواجهات الرسومية، تظهر

امام المستخدم مجموعات من الازرار والخيارات والقوائم وغيرها، ولا يمكن للبرنامج ان يتوقع ما الذي سيحدث في الخطوة التالية، لذا فإن البرنامج يقسم الى عدة اجزاء، ولكل جزء وظيفة محددة، ينفذ كلا منها عند تنفيذ ما يسمى بالحدث (Event)، فعلى سبيل المثال، تعتبر نقرة زر الفأرة حدثا، وضغط احد المفاتيح يعتبر حدثا، الاتصال بالانترنت يعتبر حدثا، كل هذه تعتبر احداث، وتسمى الدالة التي تعمل عند حدوث الحدث بالدالة المرتبطة بالحدث، ومما يعزز اهمية البرمجة المرئية كذلك هو ان الكثير من الناس يستخدمون فكرة الصور (Pictures) في تفكيرهم وتوضيحاتهم، وان كثيرا من التطبيقات تتوافق كثيرا مع استخدام الرسومات والواجهات الرسومية، وباسلوب اخر فان العقل البشري سريع التأثر بالصورة والتعامل معها، وربما نقلت صورة معلومات اكثف بكثير مما تنقله النصوص، كما ان هذا النوع من البرمجة يسهل عملية البرمجة لغير المختصين مما يوسع انتشار الحواسيب وبرمجتها.

ومن اشهر بيئات التطوير الرسومية Visual C++، Visual J++، Delphi، Visual Basic، Visual Basic.Net، Java Builder وغيرها الكثير، وتستخدم هذه البرامج نسخ محسنة من لغات البرمجة العادية وتدمجها في بيئة التطوير الخاصة بها، لذلك فأن Delphi مثلا ليست لغة برمجة بمعنى الكلمة، وانما هي بيئة تطوير تستخدم نسخة محسنة من Pascal تتميز باستخدام الكينونات (Objects)، وميزات اخرى.

وتعرف لغة البرمجة بأكثر من اسلوب، فيمكن القول انها لغة تستخدم الفأرة (Mouse) والايقونات (Icons) والرموز التي على الشاشة وقوائم الاختيار (Menus) لأعداد او تطوير البرامج. كما يمكن القول انها لغة تستخدم التمثيل المرئي (Visual Representation) لكينونات منطقية (Objects Logical) لمعالجة معلومات مرئية (Visual Information)

وتدعم التفاعل المرئي في عملية البرمجة (Visual Interaction) وتستخدم التعبيرات المرئية (Visual Expressions).

مزايا البرمجة المرئية:

تتصف عملية البرمجة المرئية بمزايا ابجابية نذكر منها :

- افكار البرمجة فيها اقل من غيرها.
- تتصف بالتماسك الكبير بين اجزاءها.
- العلاقة بين اجزائها تظهر بشكل صريح.
- مشاهدة النتائج المرئية لها بشكل فوري وسريع.
- لا يشترط تصور البرامج فيها بشكل متسلسل.
- يمكن الاستغناء عن خطوات او مراحل وسطية.
- اهتمام اقل بالامور القواعدية.
- للبرامج فيها هيكلية قابلة للاستعراض (Navigable) .
- يمكن تنفيذ اجزاء محددة من البرامج.
- تكاملية القواعد الصورية لها.
- وجود مكونات (Component) قابلة لاعادة الاستخدام، حيث تدمج مع غيرها لانتاج نظام تطبيقي كامل.

وبالرغم من وجود كل هذه المزايا، فلا يخلو الامر من وجود سلبيات، مثل:

أ. حاجة البرنامج لمساحة كبيرة لعرض اجزاءه الرسومية.

ب. وجود اشكالية في المسميات لبعض الاجزاء الرسومية، كأن نحتار في تسمية الجزء (Stop) على انه اجراء (Action) امر بالتوقف، او انه كينون (Object) لاشارة توقف مثلا.

مسرد المصطلحات:

لغة البرمجة المرئية	Visual Programming Language
مجموعة من الجمل الايقونية بنيت بقواعدية (Syntax) والتي يرتبط معها تحليل لهذه الجمل لتحديد تركيبها القواعدي، ومعنى (Semantic) والتي يرتبط معها تحليل لهذه الجمل لتحديد المعنى الذي يقصد منها.	
البرمجة المرئية	Visual Programming
هي عبارة عن اسلوب حديث نسبيا للبرمجة، تستخدم فيه برامج مساعدة لتصميم واجهة استخدام رسومية (الازرار والنصوص...) وربطها بالشيفرة البرمجية (Code)، وتسمى هذه البرامج المساعدة ببيئة التطوير المتكاملة (Integrated Development Environment: IDE)، ويستخدم هذا النوع من البرمجة تعبيرات مرئية (Visual Expressions) في عملية البرمجة، او تعالج معلومات مرئية، او قد تدعم التفاعل المرئي بين المستخدم وجهاز الحاسوب.	
واجهة المستخدم الرسومية	Graphical User Interface GUI
شاشة تبدأ بها الكثير من البرمجيات الحديثة وتفرغاتها، وتوفر للمستخدم مجموعة من الايقونات والادوات والخيارات التي تسهل عليه كثيرا عملية استخدام البرمجية.	
الرسم بالحاسوب	Computer Graphics
استخدام الحاسوب من معدات وبرمجيات لانجاز رسومات متنوعة وجذابة.	
تفاعل الانسان مع الحاسوب	Human Computer Interface
رغم انه لا يوجد اتفاق على تعريف واضح لهذا الموضوع، الا انه يمكن القول بأنه حقل يعني بتصميم وتقييم وتنفيذ انظمة الحسابات التفاعلية المجهة لاستخدام الانسان، كما يعنى بدراسة مختلف الظواهر المحيطة بهذه الانظمة.	

ايقونة	Icon
قواعد تراكيب اللغة	Syntax
مجموعة من التراكيب لكتابة جمل اللغة، يرتبط معها تحليل لهذه الجمل لتحديد تركيبها القواعدي وصحة توافقه مع تراكيب اللغة.	
معنى تراكيب اللغة	Semantic
المعنى المرتبط بتراكيب جمل اللغة يحدد ما يفترض أن تؤديه الجملة، ويرتبط بتحليل لهذه الجمل لتحديد المعنى الذي يقصد منها.	
بيئة التطوير المتكاملة	Integrated Development Enuironment
وهي بيئة لاحتضان عملية البرمجة، حيث يتوفر فيها مجموعة من الادوات المختلفة والنماذج التي توفر للبرمجة تسهيلا كبيرا في بناء تطبيقاته البرمجية.	
التعبير المرئي	Visual Expression
استخدام نماذج مرئية مختلفة، خاصة الصور، للتعبير عن معلومة معينة، مثل الشعارات (Logos)، بما في ذلك استخدام الالوان والحركة.	
المعلومات المرئية	Visual Information
معلومات تأتي على شكل مصور مرئي ناتجة عن التحليل المرئي للبيانات، مما يزيد في فهم العلاقة بين المصادر المتوفرة والاداء المتعلق بها.	
التفاعل المرئي	Visual Interaction
تفاعل الانسان مع الحاسوب دون اللجوء الى الاسلوب الكتابي، ولكن باستخدام الاشارة او تعابير بعض اعضاء الانسان كالعين وغيرها، للطلب من الحاسوب والمعلومة المطلوبة.	

الدرس الثاني

المتغيرات داخل فيجوال بيسك (Visual Basic)

قس هذا الموضوع سوف نتعرف ما هي المتغيرات ووظيفتها وانواعها وما هي الطرق المناسبة لكتابة المتغيرات داخل فيجوال بيسك ، لعل اهم ما يميز لغات البرمجة قدرتها على التعامل مع القيم بأشكال مختلفة واجراء العمليات الحسابية عليها في وضع مرن لكي لا تفقد قوتها كمحرك وبديل لعقل الانسان لذا جاءت المتغيرات التي تجعل من ادخالات المستخدم او القيم المخصصة لها في صوره تمكنها من ادارة العمليات المطلوبة بيسر وسهوله. اذاً المتغيرات في الـ . vb.net هي عبارة عن مخازن مؤقتة في الذاكرة تخزن فيها البيانات أثناء تنفيذ خطوات البرنامج لتجري عليها العمليات المطلوبة وكذلك تخزن فيها نتائج تلك العمليات.
والمتغيرات هي أحد القواعد الأساسية في كل لغات البرمجة سواء في الإنترنت أم في لغات برمجة الحاسب . لدرجة انك لا تجد برنامج مهما كان حجمه أو وظيفته ويخلو من متغير أو اكثر.

ما هي المتغيرات:

هي عبارة عن المخازن الموجودة في الذاكرة التي يحتجزها المبرمج ليضع فيها بعض القيم التي سيحتاجها في تنفيذ مشروعه.

تعتبر المتغيرات هي الأساس التي يعتمد عليه المبرمج في عمل البرنامج الذى يريد.

ماهي أنواع المتغيرات:

حيث تنقسم المتغيرات الى عدة انواع ،ولكن سنأخذ نوعين فقط من هذه المتغيرات ,
وهي المتغيرات الحرفية – المتغيرات العددية

١. Integer: يشغل ٢ بايت وهى ارقام صحيحة صغيرة المدى وتتراوح بين ٣٢٧٦٧ الى -٣٢٧٦٧
٢. long: يشغل ٤ بايت وهى ارقام صحيحة كبيرة المدى وتتراوح بين 2147483648الى-٢١٤٧٤٨٣٦٤٨

٣. single: يشغل ٤ بايت وهى ارقام حقيقية ذات دقة بسيطة

٤. Double:
يشغل ٨ بايت وهى ارقام حقيقية ذات كسر عشرى وذات دقة عاليه

٥. متغيرات عمله ارقام ماليه currency :
تستخدم فى البرامج المالية الدقيقة وتشغل ٨ بايت فى الذاكرة
٦. متغيرات حرفية string :
تستخدم لتخزين قيم البيانات الحرفية مثل الاسماء وتشمل كل حروف لوحة المفاتيح
٧. متغيرات منطقية Boolean :
تستخدم لتخزين قيم البيانات المنطقية (yes,no) او(true,false)

٨. byte
يستخدم هذا المتغير عند استخدام ارقام محددة وتتراوح بين ٠,٢٥٥
٩. variant
يستخدم لتخزين أي قيمة لأى بيانات حسب نوعية البيانات نفسها بمعنى ان هذا المتغير يتشكل حسب الحاجة وحسب نوع البيانات
١٠. default وهي القيمة (الابتدائية – الافتراضية) في حالة عدم تحديد النوع لأى متغير.

الإعلان عن المتغيرات ، حجز وتعريف المتغيرات: Declaration of variables

الإعلان عن المتغير أن تخبر "فيجول بيسك" عن اسم المتغير ونوعه ، مثل

Dim Efham As String

الصورة العامة للأمر: dim اسم المتغير as نوع المتغير
يجب ان يتم تعريف أي متغير قبل استخدامه في البرنامج بأمر dim وهى اختصار لكلمة dimension
وتعنى ابعاد او مواصفات المتغير
اسم المتغير : هو اسم للمتغير ويجب ان يبدا بحرف هجائي ولا يشمل حروف خاصة او مسافات
نوع المتغير : احد انواع المتغيرات السابقة

الشروط الواجب توافرها عند اختيار اسم للمتغير ..

- يجب أن يبدأ اسم المتغير بحرف أبجدي وليس رقما.
- ألا يزيد عدد حروف اسم المتغير عن ٤٠ حرفا.
- ويجب ألا يحتوي علي أي مسافات أو نقاط وإذا كان اسم المتغير يحتوي علي كلمتين فأنصحك باستخدام Under Score (_) للفصل بينهما ..

و يجب أيضا إلا يتضمن كلمة من الكلمات المحجوزة وهي الكلمات التي تستخدم في الأوامر و العبارات التي يستخدمها فيجول بيسك فمثلا لا يسمح باستخدام كلمة Print كاسم للمتغير فإذا احتجت لتسمية متغير مثل كلمة Print فيمكنك أن تكتب الكلمة كجزء من اسم المتغير PrintTextمثلا.

فائدة : (1)

يفضل الإعلان عن نوع المتغير لزيادة سرعة التعامل معه .المتغيرات التي لم تحدد نوعها يعمل فيجول بيسك علي الإعلان عنها تلقائيا من النوع Variant وهو أبطأ أنواع المتغيرات .

فائدة : (2)

يوجد نوعان من المتغير من النوع String وهي متغيرات ثابتة الطول Fixed Length و متغيرات متغيرة الطوال . Variable Length و المتغيرات الحرفية ثابتة الطوال وكما واضح من اسمها هي متغيرات محدد لها عدد الحروف في أثناء التصريح عنها ولا يمكن أن يتغير طولها .

- أوامر الإعلان عن المتغيرات :

أمر الإعلان : Dim يستخدم لتعريف متغير ديناميكي Dynamic Variables ضمن الأجراء ويكون مجال رؤية هذا المتغير داخل الأجراء فقط وعمرة من عمر الأجراء أي أن عندما ينتهي الأجراء ينتهي معه مفعول المتغير المحلي ويصبح لا قيمة له أو صفر ولهذا سمي أمر الإعلان هذا بديناميكية لأنه يوفر في الذاكرة بشكل جيد.

أمر الإعلان : Redim تستخدم لتعريف مصفوفة ديناميكية Dynamic Array غير معرفة لعدد العناصر أي يمكنك تغير أبعادها أثناء عمل البرنامج ويعمل أمر الإعلان Redim عندما ترغب في تحديد حجم المصفوفة وتظهر الفائدة من أمر الإعلان Redim في استغلال الجزء المطلوب فقط من الذاكرة دون زيادة . وأمر الإعلان هذا يجعلنا ندخل إلي عالم آخر وهو عالم المصفوفات وإذا تكلمنا عن المصفوفات في هذا الجزء الصغير فبذلك نكون قد لا نوفيها حق قدرها .

أمر الإعلان : Static يستخدم لتعريف متغير ستاتيكي Static Variables ضمن الأجراء ويكون مجال رؤية هذا المتغير داخل الأجراء فقط وعمرة من عمر الوحدة التي بداخلها الأجراء

سواء كانت الوحدة هذه Form أو Module أو Class أي أن عندما ينتهي الأجراء يظل المتغير ساكن في الذاكرة وكذلك قيمته موجودة ولكنك لا يمكنك الوصول إلية ألا من داخل الأجراء التابع له.

أمر الإعلان : Private يستخدم لتعريف متغير عام General Variables ولكنه يكون علي مستوي الوحدة سواء كانت Form أو ملف برمجة BAS أو فئة Class وبالطبع عمرة يكون من عمر الوحدة.

أمر الإعلان : Public يستخدم لتعريف متغير عام General Variables ولكنه يكون علي مستوي المشروع ككل ويمكن الوصول إليه من جميع الوحدات الموجودة في المشروع وعمرة من عمر البرنامج ككل ويظل في الذاكرة حتى ينتهي البرنامج.

أمر الإعلان : Global أمر الإعلان أو الكلمة المحجوزة Global كانت تستخدم في الإصدارات القديمة للفيجول بيسك ومازالت حتى الآن تستخدم وهي تؤدي نفس وظيفة أمر الإعلان Public ولكنك لن تستطيع التصريح عنها ألا في الوحدات النمطية فقط.

الإعلان بإضافة رمز مميز :

تستخدم هذه الطريقة في نوع أي متغير وذلك بإضافة حرف معين إلي اسم المتغير و هذه الطريقة تسهل عليك معرفة نوع المتغير المستخدم مع اسم المتغير و الجدول التالي يبين شكل هذه الأحرف و النوع المقابل لها..

نوع المتغير : Integer الرمز المستخدم" % "
نوع المتغير : Long الرمز المستخدم" & "
نوع المتغير : Single الرمز المستخدم" ! "
نوع المتغير : Currency الرمز المستخدم" # "
نوع المتغير : Double الرمز المستخدم" @ "
نوع المتغير : String الرمز المستخدم" $ "

فمثلا الأمر..

```
MyName$="TafTaf"
```

يعلن عن متغير من نوع String سلسة من الحروف

الاعلان باستخدام الوظيفة AS

وهنا تفيد الوظيفة AS في تميز نوع المتغير الذي يأتي بعد الوظيفة AS مع أحد الأوامر Redim. Dim. Static. Global Private. Public حيث يتم كتابة الأمر ثم اسم المتغير ثم كتابة الوظيفة AS ثم كتابة نوع المتغير . انظر المثال التالي.

كود VB :

```
Dim TafTaf  AS String
```

بعض من الامثلة عن حجز المتغيرات:

dim n as string هنا نحجز مكان في الذاكرة اسمه n ونوعه string حرفي

dim a as integer كما تعلمنا ان الاسم يأتي بعد كلمة dim اذن اسم المتغير هنا a ونوعه integer

dim a1 as integer هنا المتغير اسمه a1 والنوع مثل السابق

dim abc as double اسم المتغير abc ونوعه double

dim name as single اسم المتغير name ونوعه single

ملاحظة: يفضل فى المسائل الكبيرة والبرامج المتطورة استخدام اسم للمتغير له علامة بمحتوياته

امثلة لمتغيرات خطأ لتفاديها :

dim 1ahmed as integer خطا لأنه يبدا برقم

dim a 1 as integer خطا لأنه يحتوى على مسافة

الإعلان عن الثوابت:

يتم الإعلان عن الثوابت باستخدام (Const)أو(Public Const)
مستوى الإعلان :

-1إذا تم الإعلان داخل إجراء يعتبر الثابت المعلن عنه محليا Const a=5

-2وإذا تم الإعلان في القسم العام للنافذة يصبح عاما لجميع إجراءات تلك النافذة Const a=5

-3وإذا تم الإعلان في القسم العام للملف يصبح عاما لجميع إجراءات ذلك الملف Const a=5

-4أما إذا تم الإعلان في القسم العام للملف - كما سبق - لكن سبقت كلمة Const بكلمة Publicفسيصبح الثابت شاملا لجميع ملفات البرنامج ونوافذه

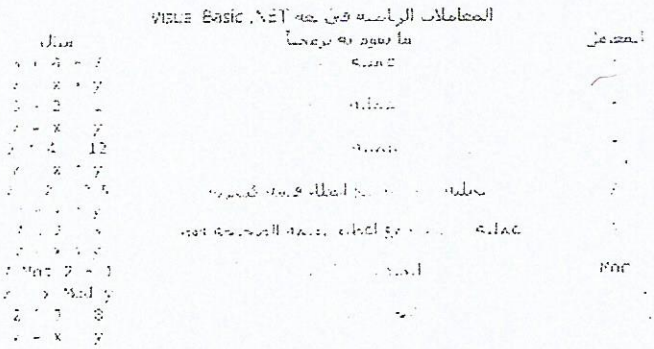
يتم الإعلان عن الثابت الشامل هكذا(Public Const a=5)

ماهي المعاملات أو: Operators

المعاملات أو Operators بشكل عام هي تلك الرموز (مثل '+') أو الكلمات (مثل 'New') التي نستخدمها لأداء عملية معينة سواء كانت عملية رياضية كالجمع والطرح أو عملية منطقية وغيرها ، ولها أيضاً أغراض أخرى.

أولاً : المعاملات الحسابيةArithmetic Operators

تمكننا لغة فيجوال بيسك .نت من أداء بعض العمليات الحسابية بشكل مباشر وهي : الجمع (+) والطرح (-) والضرب (*) والقسمة (/) وإيجاد الأس (^) وإيجاد باقي القسمة. (Mod)
وهناك معامل آخر للقسمة (\) وهو يقوم بحساب ناتج القسمة الصحيح فقط ولا يعطينا قيمة كسرية.
الجدول التالي يوضع المعاملات الحسابية في لغة فيجوال بيسك .نت:



ثانياً : معاملات المقارنة Comparison Operators أو Relational

تستطيع بسهولة اجراء عمليات المقارنة بشكل مباشر مع معاملات المقارنة التي عرفتها أيام المدرسة الابتدائية !! وهناك فرق بسيط هو أننا في البرمجة نقرأ الجمل من اليسار إلى اليمين مثلاً الجملة "x>5" نقرأها : هل x أكبر من ٥ ؟ وقد تكون صائبة True أو خاطئة. False
معاملات المقارنة الأساسية المعرفة في فيجوال بيسك .نت هي : أكبر من , (>) و أصغر من , (<)و أكبرمن أو يساوي , (>=) و أصغر من أو يساوي , (<=) و يساوي (=) ، و لايساوي (<>) .
لا ننسى أيضاً المعامل (Like) المستخدم في مقارنة السلاسل الحرفية - (String) مع استخدامات اخرى - وهو يقارن بينها من ناحية عدد الأحرف وحتى حالة الأحرف فمثلاً السلسلة "khalid"تعتبر مختلفة عن , "KHALID" انظر للمثال:

```
Dim result As Boolean
```

```
result = "Ahmed" Like "Ahmed"
Console.WriteLine(result)  'True
```

مثال آخر:

```
Dim result As String
result = "khalid" Like "KHALID"
Console.WriteLine(result)  'False
```

هناك أيضاً المعاملين (Is) و (IsNot) ولكن لن اتناولهما الآن لأنهما خاصان بالكائنات Objectsكل شي في وقته زين.

ثالثاً : المعاملات المنطقيةLogical Operators

تساعدنا المعاملات المنطقية مع معاملات المقارنة بشكل كبير في التحكم بسير البرنامج حسب شروط تضعها أنت عبر جمل التحكم مثل جملة If ... Else وجمل التكرار مثل جملة For , و هذه المعاملات هي:

المعاملNot

يقوم المعامل Not بمهمة نفي القضايا المنطقية (') وكما تعلم فإن نفي القضية يحولها من صائبة إلى خاطئة ومن خاطئة إلى صائبة ، والمثال التالي يريك طريقة استخدامه:

```
Dim result As Boolean
result = Not 3 > 2
Console.WriteLine(result)    'False
```

مثال آخر :

```
Dim result As Boolean
result = Not 4 < 2
Console.WriteLine(result)    'True
```

المعاملAnd

يستخدم المعامل And لتمثيل الرابط AND المستخدم في القضايا المنطقية ، والقضية المستخدم فيها الرابط AND تكون صائبة اذا وفقط اذا كان الطرفان صائبان ، مثال بسيط:

```
Dim result As Boolean
result = 3 > 2 And 5 > 1
Console.WriteLine(result)     'True
```

مثال آخر :

```
Dim result As Boolean
result = 3 = 2 And 5 > 1
Console.WriteLine(result)     'False
```

المعامل Or

يستخدم المعامل Or لتمثيل الرابط OR المستخدم في القضايا المنطقية ، والقضية المستخدم فيها الرابط OR تكون صائبة اذا كان أحد الطرفين صائب أو كلاهما صائبان ، سنعدل على المثال السابق:

```
Dim result As Boolean
result = 3 > 2 Or 5 > 1
Console.WriteLine(result)     'True
```

مثال آخر :

```
Dim result As Boolean
result = 3 = 2 Or 5 > 1
Console.WriteLine(result)     'True
```

المعامل Xor

يقوم المعامل Xor بمهمة الرابط XOR , والقضية المستخدم فيها الرابط XOR تكون صائبة اذا كان أحد الطرفين صائب والآخر خاطئ ، نفس المثال السابق بعد التعديل:

```
Dim result As Boolean
result = 3 > 2 Xor 5 > 1
Console.WriteLine(result)     'False
```

مثال آخر :

```
Dim result As Boolean
```

```
result = 3 = 2 Xor 5 > 1

Console.WriteLine(result)     'True
```

المعامل AndAlso

يقوم المعامل AndAlso بنفس وظيفة المعامل And ولكن مع اختلاف بسيط ، فالمعامل AndAlso يفحص الشرط الأول فاذا كان صائب يفحص الشرط الثاني ولكن اذا كان الشرط الأول خاطئ فإنه يتوقف مباشرة ولا يفحص الشرط الثاني لأن النتيجة معروفة وهي False فلا داعي لفحص الشرط الثاني ، مثال:

```
Dim result As Boolean

result = 3 > 7 AndAlso 2 < 5

Console.WriteLine(result)    'False
```

المعامل OrElse

يقوم المعامل OrElse بنفس وظيفة المعامل Or ولكن مع اختلاف بسيط ، فالمعامل OrElse يفحص الشرط الأول فاذا كان صائب فإنه يتوقف ولا يفحص الشرط الثاني لأن النتيجة معروفة وهي True فلا داعي لفحص الشرط الثاني ، ولكن اذا كان الشرط الأول خاطئ فإنه يفحص الشرط الثاني ، مثال:

```
Dim result As Boolean

result = 3 > 7 OrElse 2 < 5

Console.WriteLine(result)    'True
```

رابعاً : معاملات الإسناد: Assignment Operators

أول هذه المعاملات هو المعامل المعروف (=) والذي نستخدمه لاسناد القيم للمتغيرات بأنواعها وهو غني عن التعريف.

هناك نوع آخر من معاملات الاسناد وهي معاملات تمكنك من القيام بعملية حسابية مع اسناد القيمة في نفس الوقت وهي : معامل الجمع مع الاسناد (=+) ومعامل الطرح مع الاسناد (=-) ومعامل الضرب مع الاسناد (=*) ومعامل القسمة مع الاسناد بنوعيها الإثنين (=/) و (=\) ، وأيضاً معامل القوة أو الأس (=^) ، وإليك مثال للتوضيح:

```
Dim x As Integer = 5

x += 3    'x = x + 3
```

```
Console.WriteLine(x)    '8
```

مثال آخر :

```
Dim y As Integer = 4
y ^= 2    'y = y ^ 2
Console.WriteLine(y)    ' 16
```

هناك معامل آخر من معاملات الاسناد وهو المعامل (&=) ويستخدم مع السلاسل الحرفية (String)ووظيفته أنه يقوم باضافة سلسلة حرفية إلى سلسلة حرفية أخرى ، مثال:

```
Dim str1 As String = "Hello"
Dim str2 As String = " World"
str1 &= str2    'str1 = str1 & str2
Console.WriteLine(str1)    'Hello World
```

مثال آخر :

```
Dim str1 As String = "Welcome"
str1 &= " to my program"
Console.WriteLine(str1)    'Welcome to my program
```

الدرس الرابع

مراحل حل المشكلة:Problem Solving Stages

١. تحديد المشكلة : بمعنى تحديد المخرجات والمدخلات المتوفرة وعمليات المعالجة الحسابية أو المنطقية.
٢. إعداد خطوات الحل الخوارزمية :(Algorithm) هي مجموعة من الخطوات المرتبة ترتيبا منطقيا والتي يتم تنفيذها للوصول إلى هدف أو ناتج محدد من معطيات محددة .
بمعنى آخر هي أي طريقه تهدف لحل المسألة على صورة خطوات مرتبه ترتيباً منطقياً وإذا اتبعناه نصل لحل المسالة.
٣. تصميم البرنامج على الكمبيوتر :(Program Design) بعد الانتهاء من عمل خريطة التدفق (Flowchart ولحل المشكلة باستخدام الكمبيوتر نقوم بترجمتها إلى احدى لغات البرمجة.
٤. اختبار صحة البرنامج وتصحيح أخطائه :(Program Testing) وذلك عن طريق ادخال بيانات للبرنامج معروف نتائجها مسبقا حتى نتمكن من مقارنة النتائج التي نحصل عليها بالنتائج الفعلية وبذلك يمكن أن نكتشف الأخطاء ونقوم بتصحيحها.
٥. توثيق البرنامج :(Program Documentation) وذلك عن طريق كتابة جميع الخطوات التى اتخذت لحل المشكلة من مدخلات ومخرجات وأوامر البرنامج وتاريخ آخر تعديل للبرنامج ومن شارك فى عمل البرنامج للاحتفاظ به موثق للرجوع اليه في أي وقت بهدف التصحيح

الخوارزميات Algorithms :

إن عملية تحليل المسائل تعتمد على أربعة خطوات رئيسية المذكورة سابقاً ، ولقد تم التعرف على الخطوة الثانية والتي هي وضع طريقة للحل أي وضع خطوات متسلسلة وواضحة ومترابطة للوصول إلى النتيجة المطلوبة هذه الخطوات تسمى بالخوارزمية والتي يمكن تعريفها أيضاً (وصف مفهوم لطريقة حل المسائل بخطوات محدد وثابتة تتوفر فيها قواعد أساسية ثابتة تستطيع الآلة أو الإنسان فهمها ثم إطاعتها ، أي تنفيذها . يتكون هذا الوصف من خطوات متسلسلة وواضحة تحتوي على عمليات ثابتة ومحددة) .

خصائص الخوارزمية :-

يشترط توفر الخصائص التالية في خطوات أي خوارزمية :

١- يجب أن تكون كل خطوة في الخوارزمية واضحة تماماً دون شك أو غموض عن العملية المقصودة بتلك الخطوة .

٢- يجب أن تكون خطوات الخوارزمية متسلسلة ومحددة بعددها بحيث تشكل وحدة متكاملة تؤدي بمجموعها إلى إنجاز عمل معين أو التوصل إلى نتيجة أو نهاية .

٣- يجب أن تكون الخوارزمية كاملة بحيث تأخذ بنظر الاعتبار جميع الظروف والاحتمالات التي يمكن أن تجابه طريق التنفيذ .

مثال / أكتب خوارزمية لإجراء مكالمة هاتفية ؟

١- ابدأ .

٢- أرفع السماعة .

٣- أدر قرص الهاتف حسب الرقم المطلوب .

٤- أنتظر رنين الهاتف حتى يرفع الطرف الآخر السماعة .

٥- ابدأ بالتكلم إلى أن تنهي مكالمتك .

٦- ضع السماعة على الجهاز .

٧- توقف .

رسم التدفق البياني (Flowchart)

يمكن أن يعبر عن الخوارزمية عادةً على شكل مخطط يمكن من خلاله ملاحظة الخطوات بوضوح .

إن Flowchart هو نوع من الأنواع الرسوم البيانية التي تستخدم لتمثيل مجموعة من العمليات من البداية إلى النهائية باستخدام أشكال مختلفة متصلة بأسهم، ولأنه لا يوجد تعريب موحد لمصطلح Flowchart تعددت المسميات العربية لهذه الكلمة نذكر منها:

- مخطط انسيابي.
- خارطة انسياب أو خارطة انسيابية.
- خارطة تدفق العمل أو خارطة سير العمل.

استخدامات المخططات الانسيابية

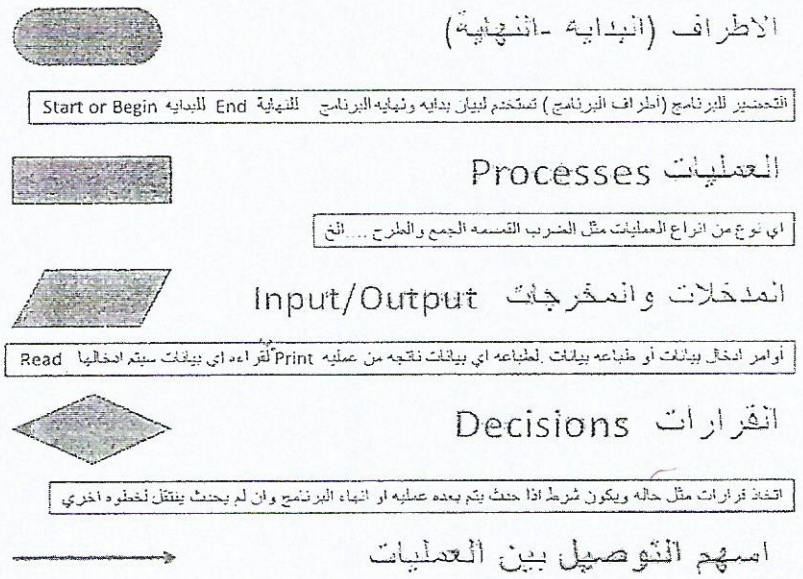
تستخدم المخططات الانسيابية لعدة أغراض:

- توثيق الإجراءات.
- تحليل العمليات.
- تتبع تدفق المعلومات.

– تعقب خطوات سير العمل.
– عرض حل لمشكلة ما خطوة بخطوة.

تعلم دلالات الرموز (الأشكال):

في المخططات الانسيابية يتم استخدام الأشكال الهندسية بحيث كل شكل يعتبر رمزا يحدد نوع العملية أو الخطوة. مثال على ذلك:



الدوائر والأشكال البيضاوية والمستطيلات الدائرية تستخدم كرموز للبداية والنهاية.
– المستطيلات تستخدم كرموز للعمليات أو الخطوات.
– شكل المعين شكل الألماس يستخدم كرمز لاتخاذ القرار.
– شكل متوازي الأضلاع يستخدم كرمز لعملية إدخال أو إخراج البيانات
– شكل شبه المنحرف يستخدم كرمز لعملية يدوية.
وهكذا... أما الأسهم والخطوط فتدل على تسلسل الخطوات واتجاهها، والعلاقة بين كل عملية وأخرى.

لذلك من المهم معرفة رموز هذه الأشكال ودلالاتها لتساعدنا على فهم المخططات الانسيابية، ولنكون أيضا قادرين على إنشاء أو رسم المخططات الانسيابية.

ارشادات يجب اتباعها عند رسم خرائط التدفق

يجب التأكد من ان الخريطة لها بداية واحدة ونهاية واحدة وان الدخول من عند البداية يجب ان يؤدى بنا الى النهاية تحت اى حال او ظرف.

من الافضل اختبار صلاحية الخريطة باستخدام مدخلات معروف قيمة مخرجاتها مسبقا.

مميزات خرائط التدفق

١.الاتصال

تتكون خرائط التدفق من اشكال نمطية وهذا يمثل وسيلة سهلة لشرح خطوات حل المشكلات للآخرين.

٢.تحليل الافعال

باستخدام خرائط التدفق فان المسألة يمكن تحليلها بصورة اكثر فعالية.

٣.التوثيق

تعتبر خرائط التدفق للبرامج من الادوات الهامة لتوثيق البرنامج.

٤.كتابة الشفرات بكفاءة

تساعد خرائط التدفق بعد رسمها على كتابة البرامج بطريقة فعالة.

٥. تصحيح الاخطاء

تساعد خرائط التدفق بعد رسمها على تتبع خطوات الحل لاكتشاف الاخطاء.

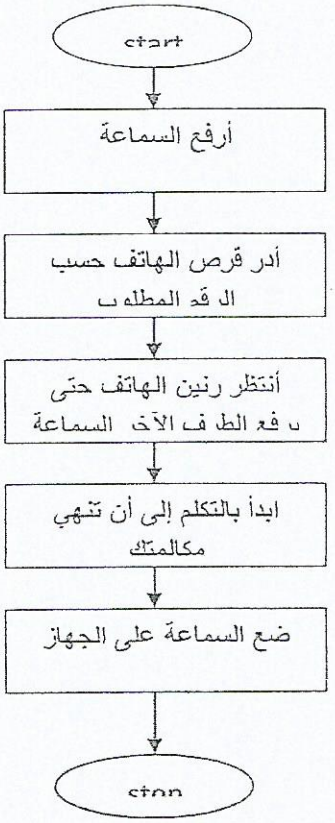
٦. كفاءة اصلاح البرنامج

يصبح اصلاح البرنامج سهلا بمساعدة خرائط التدفق.

أمثلة على رسم التدفق البياني:

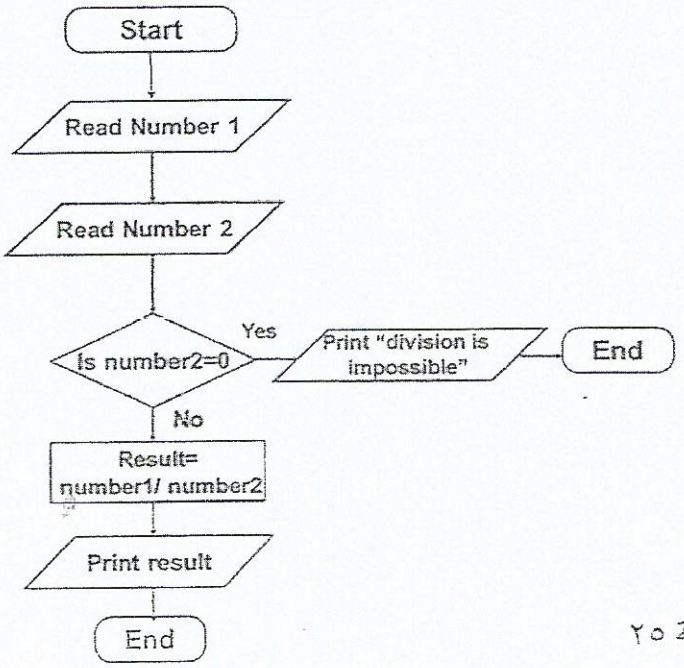
مثال (١):

تدفق بياني لإجراء مكالمة هاتفية باستخدام المخططات الانسيابية ؟



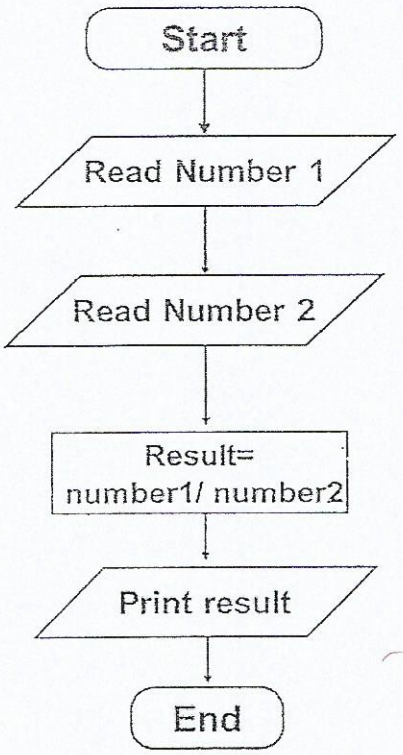
مثال (٢):

تدفق بياني يقرأ عددين، ثم يقوم بقسمتهم ثم يطبع الناتج.



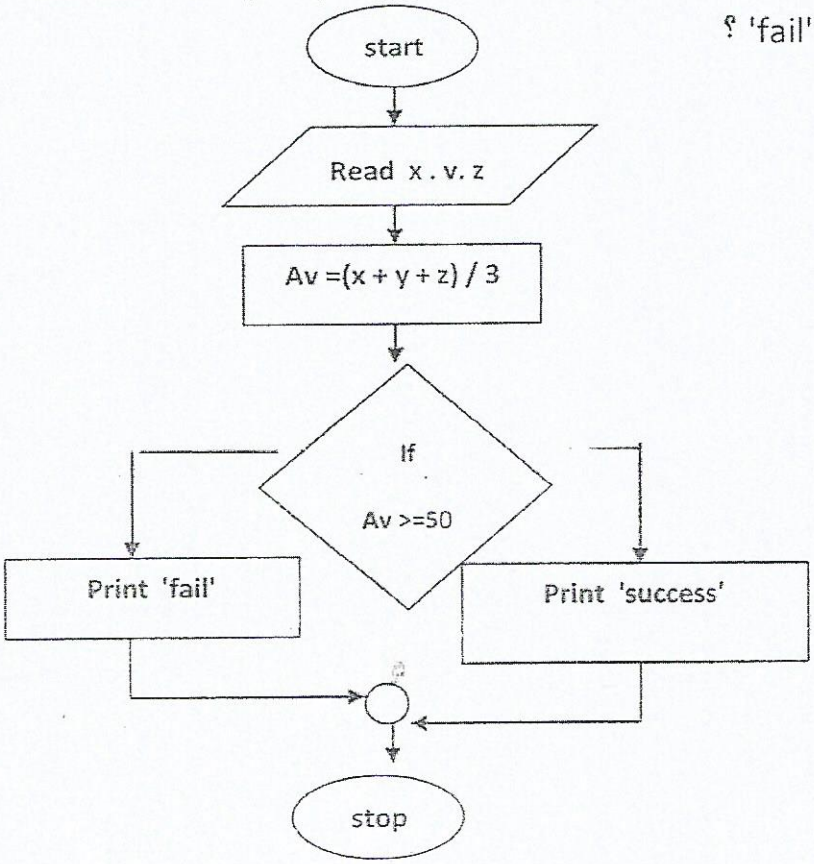
مثال (٣):

رسم تدفق بياني يقوم بقراءة عددين، جمعهم، ثم طباعة النتيجة.



مثال (٤)

رسم تدفق بياني تجد المعدل لثلاث درجات فإذا كان المعدل ناجح تطبع كلمة 'success' وعكس ذلك تطبع كلمة 'fail' ؟



تمارين:

١- اكتب خوارزمية تقرأ عدد فإذا كان العدد سالب تطبع كلمة 'negative' أما إذا كان العدد موجب تطبع كلمة 'positive' ؟

٢- اكتب خوارزمية تقرا ثلاث درجات فإذا كان المعدل مابين 80..100 تطبع كلمة 'very good' وإذا كان مابين 50..79 اطبع كلمة 'Good' أما إذا كان اقل من 50 تطبع 'fail' ؟

٣-اكتب خوارزمية تقرأ ثلاثة أعداد ومن ثم تطبع العدد الأكبر ؟

٤- ما هو الإخراج للخوارزمية التالية عندما x=3 و y=-2 ؟

الدرس الخامس

انواع الاخطاء في البرمجة

من الطبيعي جدا لأى مبرمج سواء كان مبتدئ أو محترف أن يكون لدية أخطاء برمجية أثناء عملية التطوير، وتتوقف مدى سهولة حل هذا الخطأ طبقاً لنوعه

تنقسم الأخطاء البرمجية إلى نوعين أساسين هما:

١. Static Programming Errors

وهي عبارة عن أخطاء يمكن للـ compiler اكتشافها بدون الحاجة لعملية run للكود البرمجي، وتنقسم لنوعين هما

- Syntax Programming Errors

وتعنى خطأ نحوى باللغة ولم تُحسن كتابة الأمر البرمجي كاملا هو مطلوب وهو أسهلهم حلاً، على سبيل المثال حينما تنسى semi colon ; في نهاية الأمر البرمجي

- Semantic Programming Errors

وهى قريبة من الأخطاء النحوية ولكن تعتمد على مدى صحة تسلسل او ترتيب الكود، على سبيل المثال حينما تحاول طباعة متغير لم تقم بتعرفيه مسبقاً

٢. Dynamic Programming Errors

وهى عبارة عن أخطاء لا يمكن اكتشافها الا بعد عمل run للكود البرمجة، وتنقسم أيضا لنوعين هما

- Run-time Errors

وهى عبارة عن حدوث خطأ وقت تنفيذ البرنامج على سبيل المثال حاولت ان تقرأ من ملف فى البرنامج وهو ليس موجود في الأساس، او محاولة قسمة ١ على ٠ ، حاولت ان تتصل بقاعدة بيانات لم تقم بتعريفها

- Logical Errors

وهذا النوع هو أصعبهم على الأطلاق لأنه لا يحدث أي صدد وقتي امامك وقد يستغرق سنين لاكتشافه في بعض الحالات، على سبيل المثال أنت تبرمج كود يستقبل رقمين من المستخدم ويقوم البرنامج بجمعهم ثم عرض النتيجة وبدلاً من ان تضع علامة الجمع وضعت علامة الطرح فكل مر تقوم بعمل run للبرنامج فأنه سيعمل بشكل طبيعي جدا ولكن ستشعر بالأمر اذا قارنت المدخلات بالمخرجات فأنتبهوا لهذا الأمر

الدرس السادس

مقدمة إلى لغة Visual Basic:

التنصيب والتشغيل :

لأول مرة عام ١٩٩١، ومنذ ذلك الحين وحتى Visual Basic لقد ظهرت لغة الآن تجري تعديلات على هذه اللغة وذلك بظهور إصدارات مختلفة، وآخر .وهو الإصدار الذي سنعتمده في هذا المقرر Visual Basic 6.0 إصدار هو

ملاحظة :

يمكنك تنفيذ معظم التطبيقات المذكورة في هذا الكتاب باستخدام Visual Basic 5.0

تعتبر لغةVisual Basic من لغات برمجة ويندوز، فهي تستخدم لتصميم برامج تعمل تحت نظام التشغيل Windows وبالتالي يجب على من يريد تعلم هذه اللغة أن يكون ملماً بطريقة التعامل مع نظام التشغيل ويندوز ، ويفضل أن يكون على دراية كافية بلغة البرمجة) Basic فالمتحولات وبنى التحكم و الملفات في Visual Basic تشابه وبشكل كبير مثيلاتها في (Basicولكننا سنذكر القارئ ببعض الأمور المتعلقة بلغة Basic في حال الضرورة .

إن Visual Basic من اللغات المسيرة بالأحداث شأنها في ذلك شأن معظم لغات برمجة ويندوز Delphi)و(Visual C++ ، واللغة المسيرة بالأحداث هي اللغة التي تعتمد فكرة تجزئة البرنامج إلى برامج جزئية تنفذ عند وقوع حدث ما كالضغط فوق أحد الأزرار أو تحريك مؤشر الفأرة فوق النافذة أو مرور فترة من الزمن . وبالتالي يجب عند البدء بالبرنامج تحديد الأحداث وكيفية الاستجابة لكل منه (إذا ضُغط زر كذا أفعل كذا وإذا تحرك مؤشر الفأرة فوق النافذة افعل كذا . (....

وأخيراً نقول : إن البرمجة بلغة Visual Basic هي برمجة ممتعة حقاً فمن خلال وقت قصير جداً نستطيع إنشاء برامج جيدة ومفيدة، وخصوصاً أن لغة

Visual Basicسهلة التعلم مقارنةً مع لغات مثل Visual C++ أو Java .

تنصيب Visual Basic:

إن عملية تنصيب Visual Basic عملية بسيطة جداً تشبه عملية تنصيب أي برنامج تطبيقي، ولتنصيب Visual Basic تحتاج إلى القرص الليزري الذي يحوي على هذه اللغة،ومن الجدير بالذكر أن هذه اللغة تأتي مع مجموعة لغات أخرى من Microsoft تسمى مجموعة Microsoft Visual Studio 6.0، وهذه المجموعة عبارة عن ٦ أقراص ليزري تحوي عدة لغات مثل :

- Microsoft Visual Basic 6.0
- Microsoft Visual C++ 6.0
- Microsoft Visual J++ 6.0
- Microsoft Visual FoxPro 6.0
- MSDN

إن MSDN ليست لغة برمجة، وإنما هي برنامج ضخم جداً يحوي تعليمات ومساعدة عن جميع اللغات السابقة، لتنصيب MSDN أنت بحاجة إلى القرص الخامس والسادس من مجموعة Microsoft Visual Studio 6.0. لتنصيب Visual Basic 6.0 ضع القرص الأول من مجموعة Microsoft Visual Studio 6.0ثم اتبع التعليمات التي ستظهر على الشاشة .

تتوفر Visual Basic على قرص ليزري واحد فقط، ولكن غالباً ما يكون هذا القرص هو القرص الأول من مجموعة Microsoft Visual Studio 6.0

عند تنصيب Visual Basic فإنها ستوضع – افتراضياً - ضمن المسار :
C:Program FilesMicrosoft Visual StudioVB98
إلا في حال قمنا بتغيير المسار أثناء عملية التنصيب .
ستجد داخل المسار السابق الملف التنفيذي للغة Visual Basic وهو Vb6.exe، بالإضافة إلى بعض الملفات المساعدة وبعض أمثلة قواعد المعطيات المعدة مسبقاً .

في الحقيقة، عند تنصيب نسخة Visual Basic 6.0 يتم تنصيب مجموعة من المجلدات (المكتبات) المساعدة، مثل مجلد إعداد برنامج التنصيب Setupومجلد الأدوات Tools ومجلد الرسومات Graphics ، وستجد كل هذه المجلدات داخل المسار :

C:Program FilesMicrosoft Visual StudioCommon

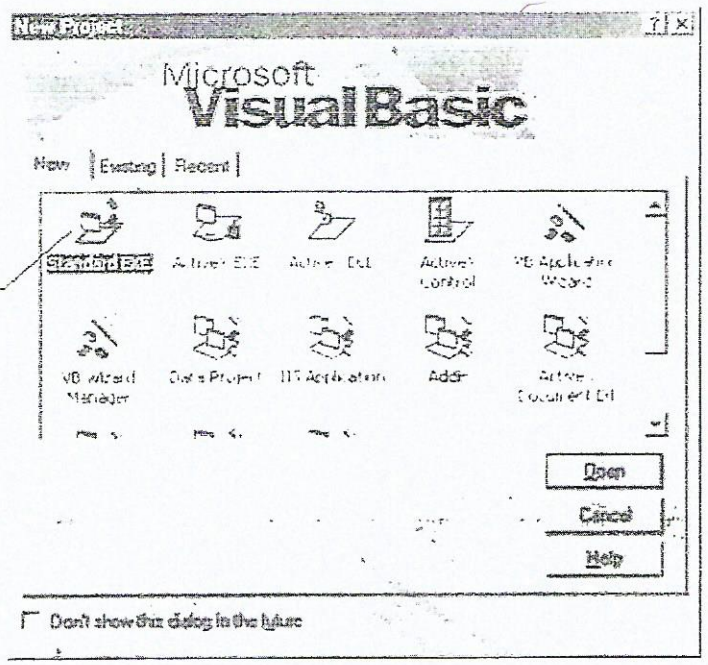
إذا لم تجد أحد هذه المجلدات فمعنى ذلك أنك نسيت تنصيبها أثناء تنصيب Visual Basicلذلك أعد عملية التنصيب وقم بإضافة ما تريد .

تشغيل Visual Basic:

بعد عملية التنصيب تضاف مجموعة اختصارات إلى قائمة بدء التشغيل وهي Microsoft Visual Studio 6.0 وداخل هذه المجموعة ستجد الاختصار Microsoft Visual Basic 6.0 المسؤول عن تشغيل Visual Basic.

إذا لم تجد الختصار السابق ضمن قائمة البرامج تستطيع تشغيل Visual Basicمن الملف التنفيذي VB6.exe.

عند تشغيل Visual Basic 6.0 تظهر النافذة التالية :



شكل (١-١) نافذة اختيار نوع المشروع

نلاحظ في هذا الشكل وجود ثلاث بوابات (صفحات) هي :

١. صفحة : New وتستخدم للبدء بمشروع جديد وهي تحوي -كما نرى- العديد من الأيقونات، كل أيقونة خاصة بنوع من المشاريع، ومن هذه

المشاريع :

- المشاريع القياسية (البرامج)، وهي الأكثر استخداماً .
- مشاريع تصميم الأدوات الإضافية ActiveX.
- مشاريع تصميم المكتبات DLL.
- مشاريع قواعد المعطيات .
- مشاريع أخرى .

٢.صفحة :Existing وتستخدم لفتح مشروع سابق .

٣.صفحة :Recent وتستخدم لعرض قائمة المشاريع التي تم العمل بها مؤخراً .

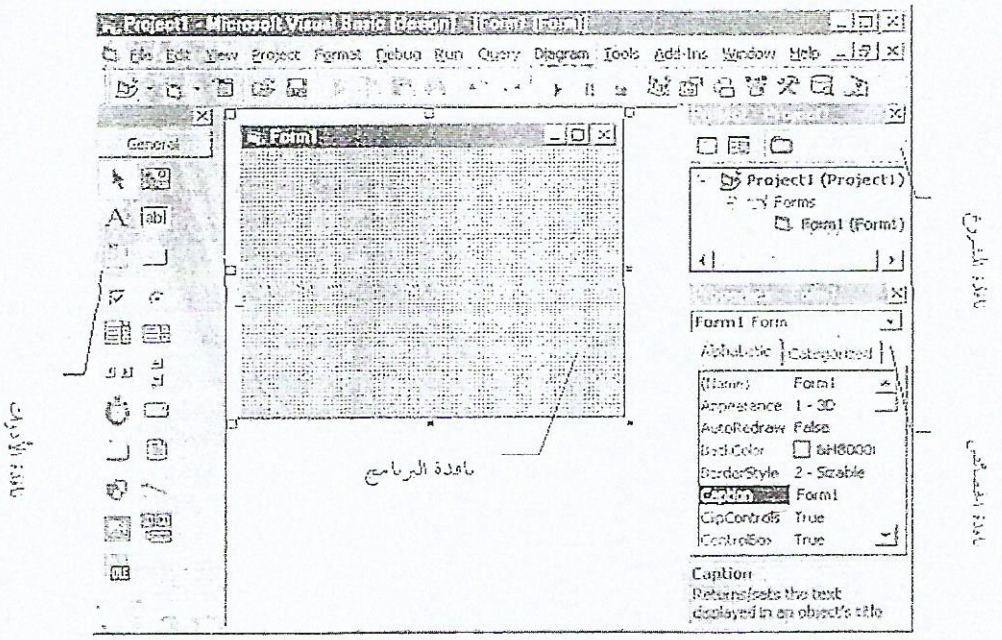
الآن اختر البوابة New ثم اختر منها الأيقونة Standard EXE أي مشروع قياسي.

بيئة تطوير Visual Basic (واجهة العمل):

عند البدء بمشروع قياسي تظهر بيئة التطوير كما في الشكل (١-٢) التالي :

نافذة VB الرئيسية:

وهي النافذة التي تحتوي الاتي:-



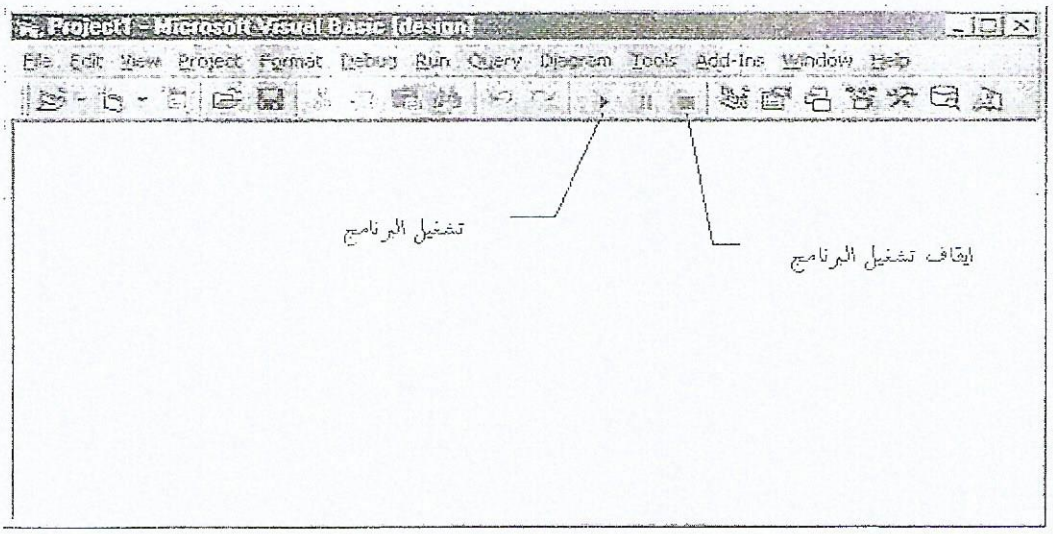
الشكل (١-٢) بيئة تطوير Visual Basic

١-شريط القوائم:

ويحوي عدداً من القوائم (File, Edit, Project) التي تضم عدة أوامر مثل فتح مشروع ، حفظ مشروع، إضافة نافذة ..الخ .

٢- أشرطة الأدوات:

غالباً ما يظهر شريط أدوات واحد هو شريط الأدوات القياسي كما يوضح الشكل(١-٣) ويحوي عدداً من الأزرار مثل زر الفتح والحفظ والتشغيل وإيقاف التشغيل .



الشكل (١-٣) نافذة Visual Basic الرئيسية

٣- نافذة أدوات Visual Basic:

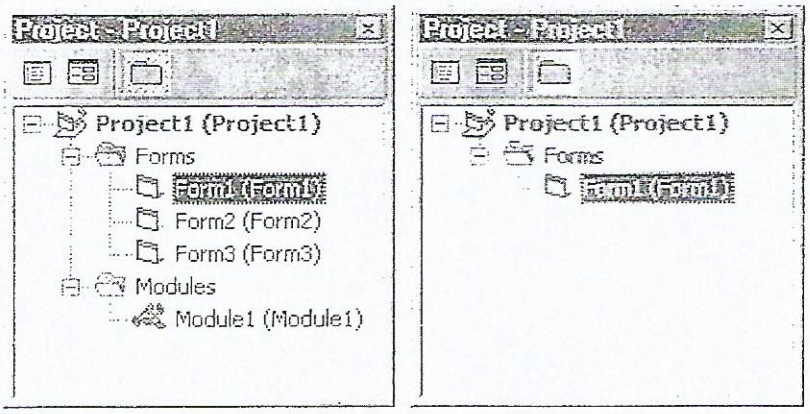
وتحوي جميع الأدوات اللازمة لبناء التطبيقات، وتظهر غالباً على يسار الشاشة، وفي حال عدم ظهورها يمكنك إظهارها باختيار الأمر Tool Box من القائمةView ، أو بالضغط على الزر المناسب في شريط الأدوات .

٤- نافذة البرنامج:

وهي النافذة التي ستشكل البرنامج وعليها ستتوضع كافة أدوات Visual Basicوهي كما نرى في الشكل (١-٢) كأي نافذة من نوافذ Windows فهي تملك شريط عنوان، وزر تكبير وتصغير وإغلاق وأيقونة تحكم .

4-نافذة المشروع:

ربما أن كلمة مشروع هي كلمة جديدة بالنسبة لك فقد كنت فيما سبق تستخدم كلمة برنامج والمعنى واحد .
والمشروع في Visual Basic ليس ملف واحد، ولكنه عبارة عن عدة ملفات مرتبطة فيما بينها، فالمشروع يتألف من عدد من النوافذ Forms وملفات البرمجة .Modules
ولتسهيل عملية الانتقال بين النوافذ وملفات البرمجة وجدت نافذة المشروع حيث يتم فيها عرض جميع النوافذ وملفات البرمجة المستخدمة في المشروع. والشكل (١-٤) يوضح نافذتين الأولى لمشروع يحوي نافذة واحدة والأخرى لمشروع يحوي ثلاث نوافذ وملف برمجة .



الشكل (١-٤) نافذة المشروع

لا تهتم الآن بكيفية إضافة نافذة جديدة لأننا سنفصل ذلك فيما بعد، ولكن إذا احببت أن تضيف نافذة جديدة فما عليك إلا أن تختار الأمر Project-Add Form.

لرؤية أي نافذة نضغط على اسمها – من نافذة المشروع- ضغطاً مزدوجاً، أو نحددها ونضغط الزر View Object الموجود في نافذة المشروع .

في عدم ظهور نافذة المشروع اختر الأمر Project Explorer من القائمة Viewأو اضغط على الزر المناسب من شريط الأدوات .

لحفظ المشروع يجب عليك حفظ النافذة في ملف، والمشروع ككل في ملف، أي أن المشروع يتكون من ملفين هما (ملف النافذة وملف المشروع .(

لذلك وعند محاولة حفظ المشروع باختيار الأمر Save Project As من القائمة File سترى أنك ستطالب باسم ملف النافذة ومن ثم اسم ملف المشروع .

٥- نافذة الخصائص :

وتستخدم لتغيير خصائص الأدوات كما سنرى، يمكنك إظهار هذه النافذة باختيار الأمر Properties Window من القائمة View.

تأكد أنك في البداية ستجد صعوبة في التعامل مع هذا العدد الكبير من النوافذ، وكثيراً ماستقف حائراً ولا تدري ما يجب عليك فعله، ولكن لا تقلق لأنك ومع الوقت ستعتاد على هذه النوافذ وكيفية التعامل معها .

مفهوم البرمجة المسيرة بالأحداث :

قبل أن تبدأ بكتابة أي برنامج في Visual Basic عليك أن تفهم معنى البرمجة المسيرة بالأحداث، تسمى برمجة ويندوز بالبرمجة المسيرة بالأحداث (Event Driven Programming) والحدث هو ما يقوم به المستخدم من أفعال على البرنامج، مثل الضغط على زر أو اختيار أمر من قائمة أو تحريك الماوس أو ضغط مفتاح ما من لوحة المفاتيح الخ .

عند وقوع أي حدث يقوم Windows بتسليم الحدث إلى البرنامج المنفذ في هذه اللحظة ويعطيه رسالة عن طبيعة الحدث الذي وقع في حدود نافذته .

يقوم البرنامج بتحليل الرسالة ثم يتخذ الإجراء الذي يراه مناسباً لهذا الحدث وبعد أن ينتهي من ذلك تعود السيطرة لنظام التشغيل .

ليس بالضرورة أن يستجيب البرنامج لكل الأحداث فمثلاً تحريك الماوس فوق نافذة البرنامج يعتبر حدث ولكن ليس بالضرورة الإستجابة لهذا الحدث إلا في برامج الرسم .

عند تصميم برنامجك عليك أن تتفهم طريقة البرمجة المسيرة بالأحداث وأن تقوم بتصميم البرنامج على أساس أن تعرض على المستخدم الخيارات المختلفة وتترك له حرية التصرف والانتقال من نافذة إلى أخرى ثم العودة وهذا لن يأتي إلا بأن تصمم برنامجك بحيث يستجيب للأحداث المختلفة التي يقوم بها المستخدم وليس على أنه سلسلة متصلة من التعليمات تنفذ من البداية إلى النهاية .

إن البرمجة المسيرة بالأحداث تؤدي إلى تجزئة البرنامج إلى عدة أجزاء كل منها يستجيب إلى حدث معين فتقوم بكتابة شيفرة كل جزء بشكل مستقل، و Visual Basicتسهل لنا هذه المهمة فهي تقوم تلقائياً بتقسيم البرنامج إلى عدة أجزاء كل جزء يدعى إجراء وكل إجراء مختص بحدث معين على أداة معينة .

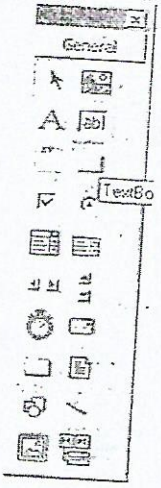
ما هي الأدوات ؟

كثيراً على ما يسمى بالأدوات، والأدوات Visual Basic تعتمد البرمجة في عبارة عن أجزاء برامج جاهزة للاستخدام، مثل أزرار الأوامر وخانات Tools النص والقوائم وغيرها.

توفر الأدوات علينا وقتاً وجهداً كبيرين، بل في بعض الأحيان تقوم هذه الأدوات بمعظم العمل بمجرد وضعها على النافذة بدون الحاجة إلى كتابة أي تعليمات.

Visual Basic 6.0 تحوي على عشرين أداة جاهزة للاستخدام، وجميع هذه الأدوات موجودة في نافذة الأدوات، ولكل أداة اسم محدد يميزها عن باقي الأدوات، ولمعرفة هذا الاسم ماعليك إلا وضع مؤشر الماوس فوق الأداة لفترة زمنية قصيرة.

ملاحظة: إن المؤشر الموجود في أعلى يسار نافذة الأدوات ليس أداةً.



إضافة الأدوات إلى النافذة :

إذا أردت أن تستخدم أي أداة اضغط فوق هذه الأداة من نافذة الأدوات

شكل الأداة	الاسم الشائع	اسم الأداة	وظيفة الأداة واستخدامها
	PictureBox	خانة الصورة	عرض الصور أو وعاء لباقي الأدوات أو لوحة رسم
	Label	أداة العنوان	عرض النصوص الثابتة التي لا يستطيع المستخدم تعديلها
	TextBox	خانة النص	طلب المعلومات من المستخدم أو عرض المعلومات
	Frame	الإطار	وعاء لباقي الأدوات
	CommandButton	زر الأوامر	ينقر عليه المستخدم لتنفيذ أمر معين
	CheckBox	خانة التحقق	اختيار مجموعة من مجموعة (اختيار الألوان المفضلة مثلاً)
	OptionButton	زر الخيار	عرض مجموعة خيارات لاختيار أحدها (اختيار اللغة مثلاً)

ToolBoxثم ارسمها على نافذة البرنامج وكأنك ترسم مستطيلاً، وهناك طريقة أخرى لإضافة الأداة وهي الضغط المزدوج فوقها من نافذة الأدوات وعندها ستتوضع هذه الأداة في مركز نافذة البرنامج وبحجم معين .

إن لكل أداة من هذه الأدوات وظيفة معينة تختلف عن وظيفة الأدوات الأخرى، وفي هذا الجدول نبين أسماء الأدوات ووظائفها .

جدول بأسماء الأدوات ووظائفها :

	Timer	المؤقت	تنفيذ مجموعة من التعليمات كل فترة زمنية محددة
	DriveListBox	قائمة الأقراص	عرض قائمة بالأقراص الموجودة بالجهاز
	DirListBox	قائمة الأدلة	عرض قائمة بالأدلة الموجودة في القرص الحالي
	FileListBox	قائمة الملفات	عرض قائمة بالملفات الموجودة في الدليل الحالي
	Shape	أداة الأشكال	رسم أشكال هندسية (مربع، دائرة، قطع)
	Line	أداة الخطوط	رسم الخطوط
	Image	أداة الرسم	عرض الصور فقط
	Data	أداة قواعد البيانات	ربط برنامج Visual Basic مع قواعد البيانات
	OLE	أداة ربط الكائنات وتضمينها	وضع كائنات من تطبيقات أخرى (مثل مستند وورد) في البرنامج واستخدام التطبيقات الأصلية في تحريرها

	ComboBox	القائمة المركبة	عرض قائمة من العناصر لاختيار أحداها أو إدخال اختياره كتابةً
	ListBox	القائمة	عرض قائمة من العناصر لاختيار أحدها
	HScrollBar	شريط تمرير أفقي	زيادة أو إنقاص قيمة ما كتدرج اللون أو حجم الصوت
	VScrollBar	شريط تمرير عمودي	نفس استخدام شريط التمرير الأفقي

-1تحليل البرنامج على الورق .
-2تصميم واجهة البرنامج وضبط خصائص الأدوات .
-3كتابة الشفرة المناسبة لكل أداة .

بعد تحليل البرنامج على الورق يجب تحديد الأدوات اللازمة للبرنامج وتحديد عدد كل منها (٥ أزرار، ٣ خانات نص...) وبعد ذلك يجب إضافة هذه الأدوات إلى النافذة ووضعها في مكانها المناسب (ربما تحتاج في برنامجك إلى أكثر من نافذة وهذا ما سوف نشرحه لاحقاً ولكن الآن سنتعامل مع نافذة واحدة فقط)، وبعد ذلك يتم ضبط خصائص هذه الأداوت بما يناسب البرنامج،

خصائص الأدوات (أنواعها وأهمها)

تعريف الخصائص:

الخصائص هي مجموعة من المواصفات التي تغيير من سلوك ومظهر الأدوات، لكل أداة في Visual Basic - بما في ذلك نافذة البرنامج Form - مثل: لون الأداة، عنوان الأداة، Properties مجموعة محددة من الخصائص حجم الأداة، موقع الأداة الخ.

تقوم بضبط Visual Basic عندما تقوم بإضافة أداة ما إلى نافذة البرنامج فإن خصائص هذه الأداة على قيم افتراضية، وبعد ذلك تستطيع تعديل هذه الخصائص كيفما تريد.

تغير (ضبط) الخصائص :

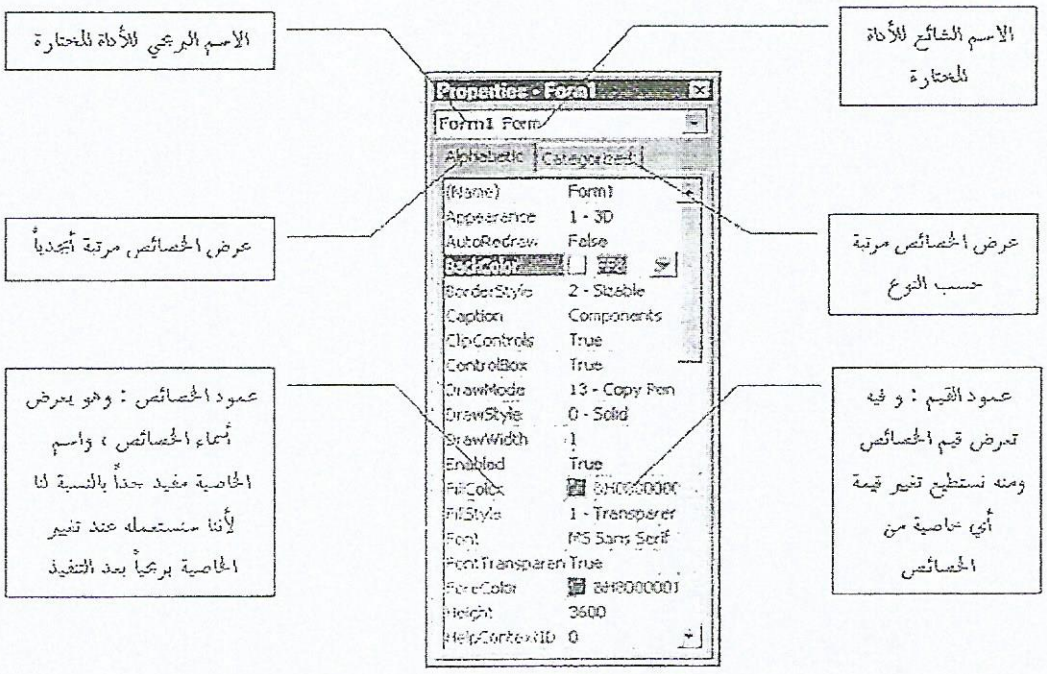
تتم عملية ضبط خصائص الأدوات أثناء تصميم البرنامج فقط باستخدام نافذة الخصائص Properties Window ، وهناك ثلاث خطوات تمر بها عملية تغيير الخصائص وهي :

1- اختيار الأداة التي نريد ضبط خصائصها من نافذة البرنامج .

2- اختيار الخاصية التي نريد تغييرها من نافذة الخصائص .

3- إدخال القيمة الجديدة .

ولإظهار نافذة الخصائص لأداة ما ، نقوم أولاً بتحديد الأداة ثم نضغط F4 فتظهر كما الشكل (٢-١)



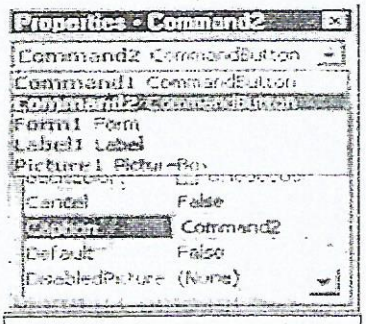
الشكل (٢-١) نافذة الخصائص

انقر على الصورة لتكبيرها

خانة اسم الأداة:

وهي تعرض اسم الأداة النشطة ونوعها، فإذا أردت أن تعدل خصائص أداة أخرى غير تلك المعروضة في هذه الخانة، فما عليك سوى أن تنقر فوق هذه الأداة من نافذة البرنامج فتتحدث محتويات نافذة الخصائص لتعرض خصائص الأداة الجديدة المختارة .

أو تختار هذه الأداة من خانة اسم الأداة فتصبح هي النشطة وتُعرض خصائصها .



اختيار الأداة من نافذة الخصائص

تغير قيمة خاصة ما :

لتغير قيمة خاصية ما نضغط فوق اسم الخاصية مما يؤدي إلى تظليلها هي

وقيمتها ، وبعد ذلك نغير قيمة هذه الخاصية من عمود القيم، وعند تغيير قيمة خاصية ما يجب التمييز بين الأنواع التالية من الخصائص وهي :

.1الخاصية النصية : وهي الخاصية التي قيمتها عبارة عن سلسة محارف كخاصية الـ Caption, Name.

.2الخاصية الرقمية : وهي الخاصة التي قيمتها عبارة عن رقم مثل خصائص الأبعادWidth ، Height.

.3الخاصية المنطقية : وهي الخاصية التي قيمتها إما) True : والتي تدل على أن الخاصية فعالة (

أو) False : والتي تدل على أن الخاصية غير فعالة .(

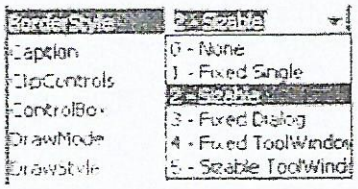
ومن الخصائص المنطقية خاصية Visible التي تتحكم بإظهار الأداة عندما تكون قيمتها True وإخفاء الأداة عندما تكون قيمتها False.

.4الخاصية اللونية : وهي خاصية قيمتها عبارة عن لون محدد بإحدى توابع الألوان Qbcolor أو RGB أو عدد ست عشري، ومن أهم الخصائص اللونية خاصية BackColor.

.5الخاصية التي قيمتها عبارة عن ملف: مثل خاصية Picture و Icon و ...

.6الخاصية واحد من مجموعة: وفيها علينا منح الخاصية إحدى قيم القائمة التي تنسدل أمامنا ولا يمكننا إعطاء قيمة من خارجها .

وكمثال على هذه الأنواع من الخصائص، خاصية BorderStyle التابعة للنافذة والتي تأخذ قيمة من ست قيم فقط موجودة في القائمة التي تنسدل عند الضغط على السهم الصغير بجوارها.



تغيير الخصائص لأكثر من أداة :

إذا أردت ضبط الخصائص لأكثر من أداة في آن واحد، قم بتحديد هذه الأدوات معاً، ثم اضغط المفتاح F4 لإظهار نافذة الخصائص التي ستعرض الخصائص المشتركة فقط بين الأدوات المحددة ، غير الخاصية التي تريد وستلاحظ أن هذا التغيير سيؤثر على جميع الأدوات المحددة (جرب ذلك على الخاصية Caption لثلاثة أزرار مثلاً .(

هناك بعض الخصائص التي لا يظهر تأثيرها عند التصميم ولكن يظهر فقط عند التنفيذ مثل الخاصية Mouse icon التي تتحكم بشكل المشيرة عند مرورها فوق الأداة .

لتنفيذ البرنامج نضغط على المفتاح F5 ، ولإيقاف تنفيذ البرنامج نضغط على زر الأغلاق الخاص بنافذة البرنامج

الخصائص الشائعة :

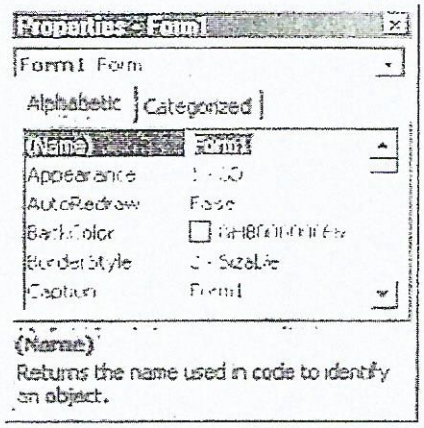
هناك مجموعة من الخصائص الشائعة الاستخدام والمتوفرة لمعظم الأدوات، سنقوم الآن بشرح أهم هذه الخصائص وسنؤجل الحديث عن باقي الخصائص لوقت آخر .

1- الخاصية Name :

تعتبر هذه الخاصية من أهم الخصائص على الإطلاق، وهي متوفرة لجميع الأدوات دون استثناء، وهذه الخاصية تحدد
الاسم البرمجي للأداة، وهو الاسم الذي يستخدم عند كتابة شفرة تخص هذه الأداة مثل :

Form1.Caption="Test"

حيث Form1 تمثل اسم النافذة .



عندما تضع أداة جديدة أو تضيف نافذة جديدة يتم وضع الخاصية Name افتراضياً لهذه الأداة وذلك بذكر اسم الأداة يليها رقم مثل Form1 و Form2 و Label1 و.. Label2 الخ .

والآن إذا كنت ترغب في تغيير هذا الاسم الافتراضي فعليك تذكر ما يلي :

يجب أن يبدأ الاسم بحرف ولا يجوز أن يبدأ برقم، ويجوز أن يتخلله أرقام .

يفضل أن يكون الاسم باللغة الإنكليزية، وذلك لتجنب المشاكل التي يمكن أن تحدث عند استخدام الأسماء العربية .

يجب أن لا يتجاوز الاسم ٤٠ حرفاً .

لا يجوز استخدام بعض المحارف مثل النقطة و الفراغ و الفاصلة و ...
لا يجوز استخدام الكلمات المحجوزة مثل FOR : و WHILE و FUNCTION و ...
يفضل استخدام الأسماء التي تدل على وظيفة الأداة، وتجنب الأسماء العشوائية .
مثلاً: يمكنك تسيمة النافذة "F" بدلاً من "Form1" و لكن عندها ستصبح الشفرة على الشكل :
F.Caption="Test"

الخاصية Name متوفرة أثناء التصميم فقط، أي من المستحيل تغيير الخاصية Name ضمن الشيفرة وهناك العديد من الخصائص الأخرى تشترك معها بهذه الصفة .

2-الخاصية BackColor (ون الأرضية)

تحدد هذه الخاصية لون أرضية الأداة، وعند محاولة تغيير هذه الخاصية يظهر مربع صغير يحوي سهم ، عند الضغط على هذا المربع يظهر لوح الألوان الذي يمكننا من اختيار اللون الذي نريد .
ونلاحظ في مربع الألوان وجود بوابتين الأولى Palette ومنها نختار ألوان ثابتة ، والثانية System ومنها نختار ألوان يستخدمها النظام . Windows

3-الخاصية Caption (لعنوان)

وهي تحدد النص الذي سيظهر على الأداة كعنوان لها، ويجب أن لا يتجاوز النص ٢٥٥ حرفاً بما في ذلك الفراغات .

4-الخاصية Enabled (تمكين)

تحدد هذه الخاصية فيما إذا كانت الأداة ستتأثر بالأحداث (النقر أو حركة الماوس) أم لا، حيث تأخذ القيمتين True تتأثر أو False لا تتأثر. لن يظهر تأثير هذه الخاصية إلا بعد تنفيذ البرنامج .

5-الخاصية Font(الخط)

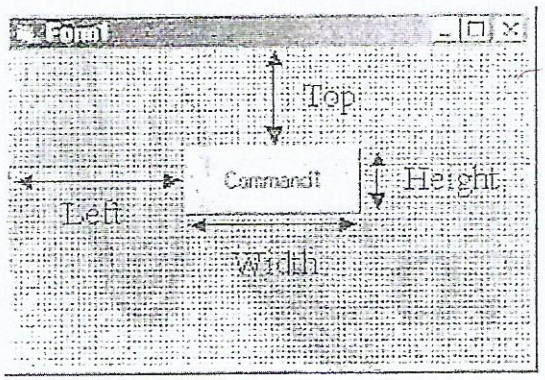
تستخدم من أجل تحديد شكل ونوع وحجم الخط الذي سيظهر به عنوان الأداة .

6-الخاصية ForeColor(لون الخط):

وهي تحدد لون الخط الذي سيكتب به عنوان الأداة .

7-الخاصية Height

تحدد ارتفاع الأداة مقدراً بـ .Twip
Twip=1/15 Picxel



خصائص الأبعاد التابعة للأداوت

الخاصيةWidth تحدد عرض الأداة

9-الخاصية Left تحدد مقدار بعد الطرف الأيسر للأداة عن الطرف الأيسر للنافذة .

10-الخاصيةTop تحدد مقدار بعد الطرف العلوي للأداة عن الطرف العلوي للنافذة .

11-خاصية Pictur وتستخدم لتحميل صورة ووضعها كخلفية للأداة، ومن الجدير بالذكر أن Visual Basic تستطيع التعامل مع عدد كبير من أنواع الصور أشهرها صور BMP.*: و WMF.* و DIB.* و * ICO. و .JPG.*
لإزالة الصورة من على النافذة نقوم بتحديد قيمة الخاصية Picture ثم

نضغط على المفتاح .Delete

12- خاصية Visible وتستخدم لإظهار أو إخفاء الأداة أثناء التنفيذ. ففي بعض الأحيان نضطر لإخفاء الأداة لسبب معين. لن يظهر تأثير هذه الخاصية إلا بعد تنفيذ البرنامج .

من المهم الآن أن تقوم بإضافة الأدوات وتجريب الخصائص السابقة، وتصميم واجهات مختلفة تتخيلها .

استخدام أدوات وأوامر

في الحقيقة هناك صعوبة كبيرة في تحديد نقطة البداية، وهناك صعوبة أكبر في تحديد المسار الذي يجب علينا سلوكه لتعلم Visual Basic، فالأدوات والأوامر كثيرة جداً، والمواضيع متعددة ومتشعبة.

كما هو معروف إن أفضل طريقة لتعلم أي لغة برمجة، هي من خلال تطبيق عدد كبير من الأمثلة والتمارين في هذه اللغة. ولكننا لا نستطيع اتباع هذه الطريقة مباشرةً في Visual Basic، لأننا بحاجة إلى معرفة الكثير حول الأدوات وكيفية استخدامها، ولذلك سنقوم في هذا الفصل بعرض جميع أدوات Visual Basic القياسية (استخدامها، خصائصها، أحداثها) وبالطبع لن نكرر ما سبق وذكرناه عن الأحداث والخصائص العامة، ولكي لا تبقى الأمور مجردة ومملة قمنا بكتابة العديد من التطبيقات العملية التي تستخدم هذه الأدوات، وذلك لدعم الفكرة وتوضيح المعنى.

تعتبر الفقرات القادمة بمثابة دليل لاستخدام أدوات Visual Basic، لذلك يمكنك تجاهل عرض الأدوات، والانتقال مباشرةً إلى التطبيقات العملية

أولاً : نافذة البرنامج Form:

كما سبق وذكرنا نافذة البرنامج هي حجر البناء الأساسي للبرنامج، فعليها تتوضع جميع الأدوات الأخرى.

خصائص النافذة Form:

- الخاصية Appearance (المظهر):

تحدد هذه الخاصية شكل النافذة ومتوفرة في زمن التصميم فقط وتقبل قيمتين هما :

Flat (٠): وعندها ستظهر النافذة مسطحة.

3D (١):وعندها ستظهر النافذة بأثر ثلاثي الأبعاد.

- الخاصية Border Style: تحدد هذه الخاصية شكل إطار النافذة وهي قابلة للتغيير وقت التصميم فقط ولها ست وضعيات أهمها:

None (0): تظهر النافذة بدون حد أو شريط عنوان وبالتالي لا توجد أزرار تحكم بالنافذة .

FixedSingle(١): تظهر النافذة بحد مفرد بحيث لا يمكن تغيير حجم النافذة من أحد حدودها ويظهر شريط عنوان لا يحوي أزرار التكبير والتصغير.

Sizable (2): وهي القيمة الافتراضية للنافذة حيث يمكن تغيير حجم النافذة باستخدام السحب من الحدود ويظهر شريط العنوان وعليه كافة أزراره.

- الخاصية ControlBox: تستخدم هذه الخاصية للتحكم بإظهار أزرار الإغلاق والتكبير والتصغير وأيقونة التحكم ولها قيمتان:

True: وهي القيمة الافتراضية حيث تظهر أزرار التكبير والتصغير والإغلاق وأيقونة التحكم.

False: وعندها لا تظهر أزرار التكبير والتصغير والإغلاق وأيقونة التحكم على شريط العنوان.

- الخصائص (DrawMode , DrawStyle , DrawWidth): تتحكم هذه الخصائص بأسلوب الرسم على النافذة وهذا سيتم شرحه لاحقاً.

- الخصائص Fillcolor ، FillStyle : الأولى لون التعبئة والثانية شكل التعبئة، وستشرح لاحقاً.

- الخاصية Font: تستخدم من أجل تحديد شكل ونوع وحجم الخط الذي يستخدم للطباعة عند استخدام الأمر Print.

- الخاصية ForeColor: وهي تحدد لون الكتابة عند استخدام الأمر Print ولون الرسم عند استخدام أوامر الرسم.

- الخاصية Icon: تحدد شكل أيقونة التحكم التي تظهر على شريط العنوان للنافذة حيث يمكن تغيير شكلها باستخدام ملفات الأيقونات ذات الامتداد ico أو cur.

- الخاصية MaxButton: تستخدم هذه الخاصية من أجل تفعيل زر التكبير، وليس لهذه الخاصية تأثير في حال كانت الخاصية ControlBox تملك القيمة False أو كانت BorderStyle تملك القيمة None.

- الخاصية MinButton: تستخدم هذه الخاصية من أجل تفعيل زر التصغير، وليس لهذه الخاصية تأثير في حال كانت الخاصية ControlBox تملك القيمة False أو كانت BorderStyle تملك القيمة None.

في حال عدم تفعيل كل من MaxButton أو MinButton فإن VB ستقوم بإخفاء هذين الزرين.

- الخاصية Moveable: تستخدم لمنع المستخدم من تحريك النافذة بعد التنفيذ ولها قيمتين:

True: عندها يمكن تحريك النافذة بسحبها من شريط العنوان.

False: عندها من المستحيل تحريك النافذة.

- خاصية RightToLeft :وتستخدم لتعريب النافذة وهذا التعريب يشمل :

ظهور عنوان النافذة على يمين شريط العنوان.

الطباعة بالأمر Print ستتم بدءاً من اليمين.

القوائم –في حال وجودها– ستظهر من على اليمين.

- خاصية StartUpPostion :وتستخدم لتحديد موقع النافذة بالنسبة للشاشة عن بدء البرنامج ،ولها القيم :

None (١): عندها ستظهر النافذة كما هو محدد لها من خصائص Left و Top .

CenterScreen (٢) :وعندها ستظهر النافذة في منتصف الشاشة.

- خاصية MousePointer: تحدد شكل مؤشر الماوس عند مروره فوق النافذة (Form) ، وهناك ١٦ قيمة يمكن الاختيار منها، وهناك القيمة ٩٩-custom والتي تسمح لنا بتحديد شكل المؤشر اعتماداً على خاصية MouseIcon والتي تستخدم لتحديد صورة لتظهر بدلاً من مؤشر الماوس.

- خاصية Picture: وتستخدم لتحميل صورة ما ووضعها على النافذة، ومن الجدير بالذكر أن VBتستطيع التعامل مع عدد كبير من الصور أهمها
*.BMP: و *.WMF و *.DIB و *.ICO و *.JPG...

لإزالة الصورة من على النافذة نقوم بتحديد قيمة الخاصية Picture ثم نضغط على المفتاح Delete.

- خاصية WindowState : تستخدم لتحديد حجم النافذة عند تشغيل البرنامج ولها ثلاث قيم هي:

Normal (0): وعندها ستظهر النافذة بالحجم المحدد لها من خاصتي Width و Heightللنافذة.

Minimized (1): وعندها ستظهر النافذة مصغرة على شريط المهام.

Maximized (2) : وعندها ستظهر النافذة ملء الشاشة.

ثانياً : زر الأوامر Command:

يمكن لزر الأوامر أن يأخذ أحد الشكلين: Standard (قياسي)مثل باقي أزرار ويندوز، Graphical (رسومي) أي أن الزر يمكن أن يقبل الصور و الألوان.

خصائص زر الأوامر Command:

- الخاصية Cancel: تستخدم هذه الخاصية لربط المفتاح Esc مع الزر أي أن الضغط على المفتاح Escسيكافئ الضغط على الزر ويتم ذلك بضبط الخاصية على القيمة True، ويكثر استخدام هذه الخاصية مع زر إنهاء البرنامج.
- الخاصية Default: تستخدم هذه الخاصية لربط المفتاح Enter مع الزر، أي أن الضغط على المفتاحEnter سيكافئ الضغط على الزر ويتم ذلك بضبط الخاصية على القيمة True، ويكثر استخدام هذه الخاصية مع زر موافق البرنامج.
- الخاصية DisabelePicture: وتستخدم لتحديد الصورة التي ستظهر على الزر عندما يصبح غير ممكن، وليس لهذه الخاصية تأثير إلا إذا كانت خاصية Style مضبوطة على القيمة Graphical.
- الخاصية DownPicture: وتستخدم لتحديد الصورة التي ستظهر على الزر عندما يكون مضغوطاً، وليس لهذه الخاصية تأثير إلا إذا كانت خاصية Style مضبوطة على القيمة Graphical
- الخاصية Picture: وتستخدم لتحديد الصورة التي ستظهر على الزر، وليس لهذه الخاصية تأثير إلا إذا كانت خاصية Style مضبوطة على القيمة Graphical
- الخاصية Style: تستخدم لتحديد نوع الزر ولها قيمتين Standard أي أن الزر هو زر قياسي لا يقبل الصور والألوان ، والثانية هي Graphical أي أن الزر يقبل صورة .

- الخاصية MousePionter: تستخدم لتغيير شكل مؤشر الماوس عند مروره فوق الزر وهي تحوي القيمة99 التي تستخدم عندما نريد وضع أيقونة كمؤشر للماوس وذلك باستخدام خاصية MouseIcons.

- الخاصية TabIndex: تستخدم من أجل ترتيب الأدوات من حيث حصولها على التركيز عند الضغط على المفتاح Tab، ويبدأ الترقيم من 0 حسب اختيار الأداة.

تضع VB قيم هذه الخاصية للأدوات بحسب ترتيب وضع الأدوات على النافذة.

- الخاصية TabStop: تحدد فيما إذا كانت الأداة ستحصل على التركيز أثناء الضغط على المفتاح Tabأم لا.

ثالثاً: خانة النص Text:

تستخدم هذه الأداة من أجل عرض المعلومات على المستخدم أو طلب المعلومات من المستخدم، وهذه الأداة تتسع لـ ٦٥٥٣٦ حرف.

خصائص الأداة Text:

- Alignment: تحدد هذه الخاصية موضع النص داخل الأداة، وهي تملك ثلاث قيم (Center، Right ،Left).

- خاصية Text: وهي تمثل محتويات خانة النص.

- MaxLength: تحدد هذه الخاصية العدد الأعظمي للمحارف الذي يمكن أن تتسع له هذه الأداة، والقيمة صفر تعني أن عدد الأحرف أعظمي أي ٦٥٥٣٦ حرف.

- MultiLine: تحدد هذه الخاصية إن كانت الأداة ستتقبل أكثر من سطر أم لا ولها قيمتين هما:

False: وتعني أن الأداة ستقبل سطر واحد فقط.

True: وتعني أن الأداة ستتقبل أكثر من سطر، وبهذا تصبح شبيهة ببرنامج المفكرة.

- Locked: تستخدم هذه الخاصية لمنع المستخدم من تغيير محتويات هذه الأداة ويتم ذلك بوضع القيمةTrue لهذه الخاصية.

يجب منع المستخدم من تغيير محتويات الأداة عندما نستخدمها لعرض المعلومات.

- Passwordchar: تستخدم هذه الخاصية من أجل تشفير الإدخال بحسب الرمز الذي نريد، وتستخدم في كلمات السر وما شابه. ليس لهذه الخاصية تأثير إلا إذا كانت خاصية MultiLine مضبوطة على القيمة False.

- ScrollBars : وتستخدم لعرض أو إخفاء أشرطة التمرير الخاصة بالأداة عندما تكون خاصيةMultiLine مضبوطة على القيمة True وهي تملك أربع قيم هي :

0-None وعندها ستظهر الأداة بدون أشرطة تمرير .

1-Horizontal وعندها سيظهر شريط تمرير أفقي فقط.

2-Vertical وعندها سيظهر شريط تمرير عمودي فقط.

3-Both وعندها سيظهر شريط تمرير أفقي وشريط تمرير عمودي بنفس الوقت.

الأداة Label:

تستخدم هذه الأداة لعرض العناوين.

الخصائص التابعة للأداة Label:

الخاصية Borderstyle: تستخدم من إظهار أو إخفاء حد الأداة وتملك قيمتين :

None عندها ستظهر اللافتة بدون حد.

FixedSingle عندها ستظهر الأداة بحد مفرد.

الخاصية BackStyle: وتستخدم لتحديد شكل خلفية الأداة عاتمة أو شفافة.

الخاصية BorderStyle: تستخدم لعرض الأداة مسطحة أم مجوفة مثل الأداة Text.

الخاصية Alignment: تحدد هذه الخاصية جهة عرض النص في يمين الأداة أو وسطها أو إلى اليسار.

- الخاصية Autosize: تحدد فيما إذا كانت الأداة ستأخذ حجم النص الذي تحويه أو بالحجم الذي نحدده أثناء التصميم. حيث تملك قيمتين True ، False.

- الخاصية ToolTipText: تستخدم لكتابة نص سيظهر عند مرور مؤشر الماوس فوق الأداة والنصوص التي تكتب في هذه الخاصية هي توضيحية توضح عمل الأداة.

خامساً: الأداة Frame:

تستخدم هذه الأداة كوعاء لباقي الأدوات الأخرى وذلك من أجل فرز العمليات وتحسين المظهر ويجب أن نستخدم طريقة الرسم عند وضع الأدوات عليها لأن طريقة النقر المزدوج غير فعالة.

ليس لها أي خصائص أو أحداث خاصة.

سادساً: الأداة Checkbox:

تستخدم هذه الأداة من أجل اختيار مجموعة خيارات من مجموعة.

خصائص Checkbox:

الخاصية Alignment: تحدد مكان توضع الكتابة يميناً أو يساراً.

الخاصية Style: تحدد شكل ظهور الأداة تأخذ هذه الخاصية إحدى القيمتين:

1-Standard حيث تظهر الأداة بالشكل الافتراضي.

2-Graphical حيث تظهر الأداة على شكل زر يمكن ضغطه للأسفل ويبقى مضغوطاً و بالضغطة الأخرى الثانية يرتفع للأعلى. وفي هذه الحالة ترث هذه الأداة معظم خصائص زر الأوامر.

الخاصية Value : و التي تملك القيم التالية:

0-Unchecked أي أن الأداة تحمل القيمة False.

1-Checked أي أن الأداة تحمل القيمةTrue .

أداة الاختيارOptionButton

تفيد من أجل اختيار عنصر من مجموعة عناصر، وهنا من المستحيل اختيار أكثر من عنصر في آن واحد.

خصائص OptionButton:

خصائص هذه الأداة تشبه تماماً خصائص الأداة CheckBox إلا أن القيمة Value لها هي إما True أوFalse أي أن أداة واحدة فقط من بين جميع الأداوت OptionButton تأخذ القيمة True.

الأداة:ComboBox

تُستخدم هذه الأداة من أجل عرض مجموعة من البنود متيحة بذلك للمستخدم اختيار أحد هذه البنود.

خصائص ComboBox:

List: وهي قائمة البنود التي ستعرض ضمن الأداة.

Locked: تستخدم من أجل الحماية وتأخذ قيمتين: True عدم السماح للمستخدم بالكتابة داخل الأداة.

False السماح للمستخدم بالكتابة داخل الأداة.

Sorted: تستخدم من أجل ترتيب العناصر داخل الأداة أبجدياً.

Style: تحدد طريقة عمل الأداة وفق ما يلي:

DropDwon Combo: وهو الوضع الافتراضي وتتيح اختيار العنصر عن طريق النقر فوقه أو كتابة اسمه فوق الأداة.

Simple Combo: تصبح مشابهة لخانة النص (أي يختفي السهم) وهنا نكتب الخيار فيها ويمكن التنقل بواسطة الأسهم.

DropDwon List: وهي قائمة لا يمكن الكتابة داخلها نهائياً بل علينا اختيار أحد العناصر الموجودة حصراً.

Text: وهي القيمة الافتراضية التي ستظهر داخل الأداة، ولا يمكن تغيير قيمتها إذا كانت الخاصيةStyle مضبوطة على القيمة (٢).

أحداث الأداة ComboBox:

Change: هذا الحدث أصبح معروفاً لدينا ولكنه لا يقع إلا عند الكتابة على الأداة أما اختيار قيمة من القيم الجاهزة لا يجعل هذا الحدث يقع.

Click:يقع عند اختيار عنصر من عناصر الأدوات بواسطة الماوس أو الأسهم.

إضافة العناصر إلى الأداة برمجياً: تتم إضافة العناصر إلى الأداة Combo باستخدام الأمر AddItem أي الشكل العام:

```
Combo1.AddItem "text"
```

وغالباً ما تكتب هذه الشيفرة في الحدث LoadForm أما لحذف كافة عناصر الأداة Combo فنكتب التعليمة:

```
Combo1.Clear
```

الأداةListBox

ستخدم لعرض لائحة من البنود ليقوم المستثمر باختيار واحدة منها أو أكثر، إذا تجاوز عدد البنود العدد الذي يمكن أن يعرض، يضاف شريط التمرير بشكل آلي إلى هذه الأداة.

خصائص الأداة ListBox:

- List: متوفرة في زمن التصميم والتنفيذ تعيد أو تضع البنود المحتواة في جزء لائحة لأداة وهي من نوعString.
- MultiSelect: تأخذ إحدى القيم التالية:
- None: تظهر في وضعها الافتراضي فلا يمكن اختيار أكثر من بند واحد.
- Simple: يمكن اختيار أكثر من بند بمجرد النقر على بند آخر.
- Extented: يمكن اختيار أكثر من بند واحد ولكن باستخدام أحد المفتاحين Ctrl أو Shift.
- Sorted: إذا وضعت على القيمة True فإن البنود المحتواة في اللائحة ستظهر مرتبة أبجدياً.
- Style: إذا وضعت على القيمة CheckBox سيظهر بجانب كل بند مربع اختيار يشبه الأداةCheckBox.

وهنالك بعض الخصائص المتوفرة في زمن التنفيذ فقط (أي لن نراها ضمن قائمة الخصائص المتوفرة في زمن التصميم) وأهمها:

- ListCount: تعيد عدد البنود الموجودة ضمن اللائحة.
- ListIndex: تعيد أو تضع رقم ترتيب البند المختار من اللائحة وننوه هنا أن قيمة هذه الخاصية بالنسبة للبند الأول من اللائحة هي ٠ ، أم البند الأخير فيأخذ القيمة ListCount-1 .

ملاحظات:

- الترقيم ضمن أية قائمة يبدأ من الصفر.
- معرفة البند المختار من اللائحة يتم بالشيفرة List1.List(List1.ListIndex) ، أو بشكل أسهلList1.Text .
- إضافة بند إلى اللائحة يتم باستخدام المنهج(List1.AddItem "Text").
- لحذف عنصر نجد المنهج (رقم العنصر المراد حذفه List1.RemoveItem) .

الأداة PictureBox:

تستخدم هذه الأداة لعرض الصور حيث تتوضع الصورة في الزاوية العليا اليسرى من هذه الأداة، كما يمكن استخدام هذه الأداة كوعاء، لباقي الأدوات مثل الأداة Frame، ويمكن الرسم عليها باستخدام أوامر الرسم.

الخصائص:

- الخاصية Picture: وتستخدم لتحديد الصورة التي ستظهر داخل الأداة من تحديد ملف هذه الصورة.
- الخاصية Autosize: وتأخذ إحدى القيمتين True : عندها ستتمدد الأداة لتأخذ حجم الصورة.

  False : عندها سيظهر القسم الممكن من الصورة.
- الخاصية Align : وتستخدم لتحديد موقع الـ Picture على النافذة (أعلى – أسفل – يسار – يمين).

١٢- الأداة Image :

تستخدم لعرض الصور فقط.

الخصائص:

- الخاصية Picture: وتستخدم لتحديد الصورة التي ستظهر داخل الأداة من تحديد ملف هذه الصورة.

- الخاصية Stretch : تأخذ قيمتين True : عندها سيتغير حجم الصورة ليناسب حجم الأداة.

  False : عندها سيتغير حجم الأداة ليناسب حجم الصورة.

ملاحظة: تحميل الصورة برمجياً إلى الأداة Image1 يتم كما يلي:

Image1.Picture = LoadPicture ("اسم الصورة مع مسارها الكامل").

الأداة: شريط التمرير:ScrollBar

ضم نافذة الأدوات أداتين من هذه الأداة HscrollBar شريط أفقي و VscrollBar شريط عمودي وهما متشابهان تماماً لذلك سنتكلم عن واحد فقط منهما، إن الفائدة من هذه الأداة هي زيادة أو إنقاص شقيمة مدرجة مثل التحكم بالصوت، التحكم بالإضاءة، التحكم بالأبعاد، التحكم بحجم صفحة الكتابة.

الخصائص:

- Max: وهي القيمة العظمى التي يمكن أن يصل إليها الشريط.
- Min: وهي القيمة الصغرى التي يمكن أن يصل إليها الشريط.
- SmallChange: وهي مقدار الزيادة التي ستطرأ على القيمة Value عند الضغط على أحد سهمي شريط التمرير.
- LargeChange: مقدار التغير عند الضغط على مجرى الشريط.
- RightToLeft: وهي اتجاه الزيادة في القيمة من اليسار إلى اليمين أو العكس.
- Value: وهي القيمة الابتدائية التي سيأخذها الشريط عند التنفيذ.

نفذ البرنامج واضغط على شريط التمرير لتحصل على ألوان مختلفة.

الأداة Timer وأمثلة عنها

وتعمل على تنفيذ مجموعة من الأوامر بعد كل فترة زمنية تحدد من قبل المبرمج،

خصائصها:

- **Interval**: وهي الفترة الزمنية التي يجب انتظارها حتى يتم تنفيذ الأوامر ضمن الأداة، وهي مقدرةبالميلي ثانية.
- **Enabled**: تمكين أو عدم تمكين الأداة.

للأداة حدث واحد هو Timer ويقع كلما انقضت الفترة الزمنية المحددة بالخاصية **Interval**.

مثال١: لنضع على النافذة الأداة Label و الأداة Timer حيث Interval=1000 ولنكتب الشيفرة التالية في الحدث Timer:

```
Label1.Caption =Timer $
```

مثال٢: ضع الأداة Timer على النافذة Form حيث أن Interval=1000 وعرف في Generalمتحول color1 من نوع Integer بالشكل:

```
Dim color1 as integer
```

واكتب الشيفرة التالية في الحدث Timer:

```
Form1.BackColor = Qbcolor(color1)

color1=color1+1

IF  color1=15 then  color1=0
```

مثال٣: إنهاء البرنامج بعد ٦٠ ثانية:

١- نعرف N كمتحول عام على مستوى النافذة.

٢- في الحدث Timer1_Timer نكتب:

```
N=N+1
```

IF N=60 then End

مثال٤: (شاشة توقف)

١- ابدأ بمشروع جديد.

٢- غير خصائص النافذة Form كما يلي:

الخاصية	القيمة
BorderStyle	0-None
BackColor	أسود
WindowStat	2- Maximized

٣- في الحدث Form_Dblclick اكتب تعليمة إنهاء البرنامج *End*.

٣- ضع Timer على النافذة، واضبط خاصية Interval على القيمة 100.

٤- اكتب الشيفرة التالية في الحدث Timer1_Timer:

```
Dim w As Long

Dim h As Long

w = Form1.Width

h = Form1.Height

Circle (Rnd * w, Rnd * h), 200, QBColor(Rnd * 15)
```

٥- نفذ البرنامج ولاحظ كيف سيتم رسم دوائر بألوان عشوائية على النافذة.

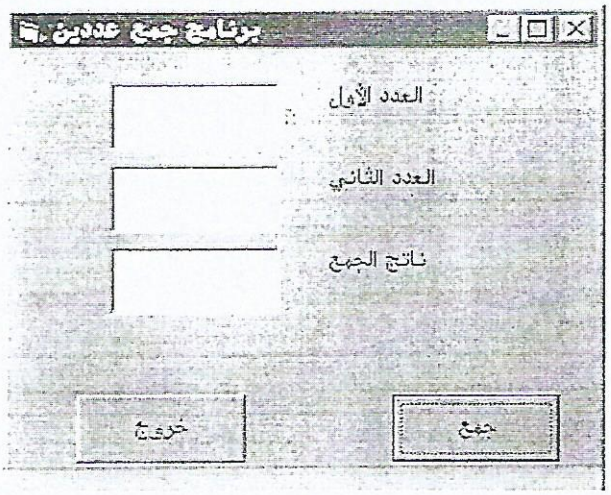
الدرس السابع

تطبيقات عددية:

برنامج جمع عددين:

الأدوات المستخدمة: زر أوامر CommandBox، خانة نص TextBox، أداة عنوان Label.

تصميم واجهة التطبيق:



١- ضع زر أوامر Command1 وغير خاصية Caption له إلى "جمع"، ثم ضع زر أوامر ثاني Command2 وغير خاصية Caption له إلى "خروج".

٢- ضع ثلاث خانات نص، وقم بحذف محتوياتهم وذلك من خلال خاصية Text.

ملاحظة: إن خاصية محتوى خانة النص هي Text وليست Caption كمعظم الأدوات.

٣- ضع ثلاث أداوات عنوان Label وغير خاصية Caption لهم إلى "العدد الأول"، "العدد الثاني"، "ناتج الجمع" بالترتيب.

٤- غير خاصية Caption التابعة للنافذة إلى "برنامج جمع عددين".

٥- قم برصف الأدوات بحيث تظهر بشكل مشابه للشكل السابق.

كتابة الشفرة:

قبل أن نكتب الشفرة يجب علينا أن نحدد كيف سيعمل هذا التطبيق، أي ماذا يجب على المستخدم أن يفعل وكيف سأرد على أفعاله.

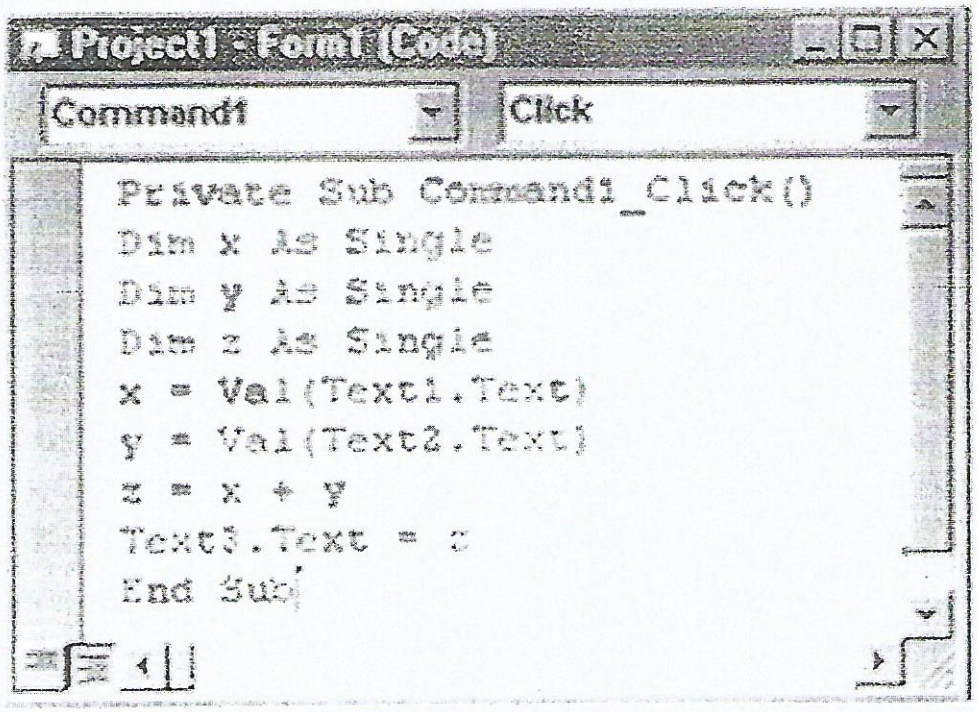
عند تنفيذ البرنامج سيقوم المستخدم بإدخال العدد الأول في خانة النص الأولى Text1، والعدد الثاني في خانة النص الثانية Text2، ومن ثم سيضغط على الزر "جمع" ليتم جمع العددين ووضع الناتج في Text3.

١- إذاً كل ما علينا فعله هو عملية جمع العددين المدخلين ووضع الناتج في خانة النصText3، ويتم ذلك في الحدث Command1_Click، لذلك اكتب الشفرة التالية في هذا الحدث:

```
Dim x As Single
Dim y As Single
Dim z As Single
x = Val ( Text1.Text )
y = Val ( Text2.Text )
z = x + y
Text3.Text = z
```

٢- في الحدث Command2_Click اكتب تعليمة إنهاء البرنامج وهي *End*.

شرح الشفرة السابقة:



لقد قمنا في الشفرة السابقة بتعريف ثلاث متحولات x,y,z من نمط عدد كسري فردي الدقة، حيث x وy هما العددين و z هو ناتج الجمع.

قمنا في السطر الرابع باسناد القيمة العددية Val لمحتويات خانة النص الأولى Text1 إلى المتحول x، وقمنا في السطر الخامس باسناد القيمة العددية Val لمحتويات خانة النص الأولىText2 إلى المتحول y.

قمنا بعد ذلك بجمع العددين ووضع الناتج في خانة النص Text3.

ملاحظة: إن السطر

x = Val (Text1.Text)

واسندها للمتحولText1سيتكرر كثيراً فيما بعد، ومعناه خذ القيمة العددية لمحتويات خانة النص x.

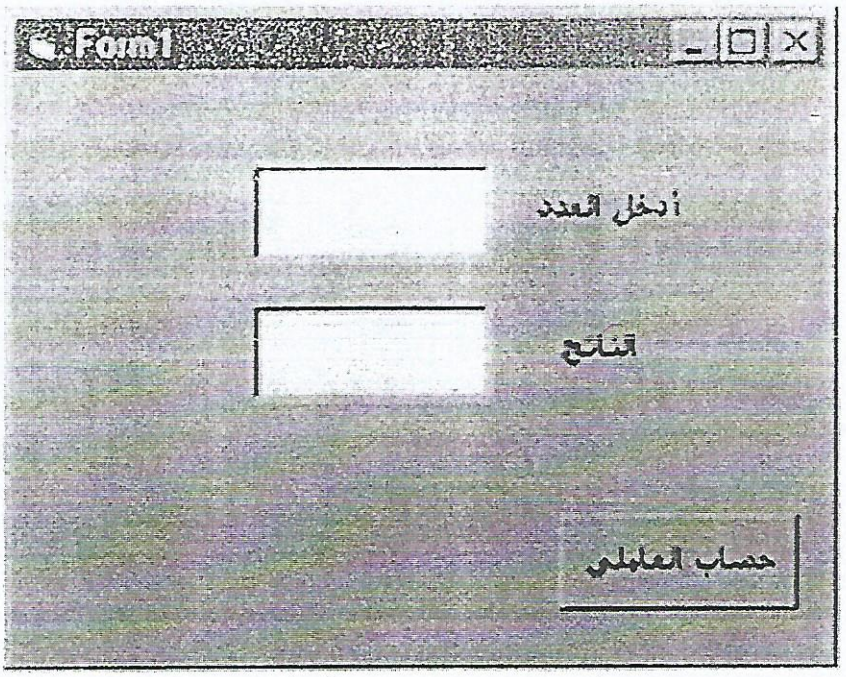
تطبيق لحساب n!:

١- ابدأ بمشروع جديد.

٢- ضع خانتي نص، وأداتي عنوان وزر أوامر.

٣- صمم واجهة البرنامج كما يلي:

٤- اكتب الشفرة التالية في الحدث Command1_Click:




```
Dim n As Integer
Dim i As Integer
Dim s As Long
s = 1
n = Val(Text1.Text)
For i = 1 To n
 s = s * i
```

```
Next i

Text2.Text = s
```

٥- نفذ التطبيق، واختبره من أجل مجموعة من الأعداد.

ملاحظة: لقد استخدمنا الحلقة For في الشفرة السابقة لحساب العاملي لعدد ما.

عيوب البرنامج:

١- يقبل هذا البرنامج إدخال الأحرف، في حين يجب أن لايقبل إلا الأرقام.

٢- يرمي هذا البرنامج خطأً عند إدخال عدد أكبر من ١٢.

الحلول:

١- لحل مشكلة قبول إدخال الأحرف قم بإضافة الشفرة التالية في الحدثText1_KeyPress:

```
If (KeyAscii < 48 Or KeyAscii > 57) And KeyAscii <> 8 Then

  KeyAscii = 0

End If
```

الرقم ٨ في جدول الآسكي يقابل مفتاح الحذف BackSpace.

٢- أما بالنسبة لمشكلة الخطأ الذي سيحدث في البرنامج عند محاولة حساب العاملي لعدد أكبر من ١٢ فيمكن حلها بإضافة بعض الأسطر إلى شفرة حساب العاملي بحيث تصبح على الشكل:

```
Dim n As Integer

Dim i As Integer

Dim s As Long

s = 1

n = Val(Text1.Text)

IF n > 12 Then
```

```
    MsgBox "العدد كبير، لا يمكن حساب العاملي"
Else
    For i = 1 To n
    s = s * i
    Next i
    Text2.Text = s
End If
```

الإضافة تظهر بالأسود العريض.

التوابع أو الدوال:

ما هي الدوال : الدوال هي أسماء محجوزة ومعرفة من قبل الفيجوال بيسك لتقوم بعمل معين مثل المصفوفات والقيم المطلقة وغيرها..
أو بمعني ابسط واعم هي عبارة عن برنامج صغير مكتوب مسبقا و محفوظ داخل لغة الفيجوال بيسك يمكن استدعاءه من خلال برنامجك والاستفادة منه ، إضافة إلى ذلك يمكنك أنت أن تكتب دالة وتستدعيها أكثر من مرة داخل برنامجك . والدوال في الفيجوال بيسك تنقسم إلى:

١. الدوال الرياضيةMathematics Functions .
٢. دوال سلاسل البياناتString Functions .
٣. دوال الوقت و التاريخDate and Time Functions
٤. دوال التحقق من أنواع البياناتData Type Inspection Functions
٥. دوال المدخلات و المخرجاتProgram Output and User Input Functions
٦. دوال ماليةFinancial Functions
٧. دوال التحويلاتConversion Function
٨. دوال التعامل مع الفهارس
٩. دوال التعامل مع الملفات
١٠. دوال المصفوفات
١١. و دوال متنوعة أخري
١٢. دوال معرفة من قبل المبرمج(UDF) User Defined Functions .

ونلاحظ أن نوع الدوال من رقم (١) حتى رقم (٧) أنها دوال داخلية ضمن اللغة، أي تأتي مُعرفة في لغة الفيجوال بيسك و مبرمجة مسبقاً لكي تستخدمها مباشرة أما النوع الأخير فيتم تعريفه بواسطة المبرمج، أي بواسطتك أنت.

أولا: الدوال الرياضية Mathematics Functions .

الدالة Abs: ترجع القيمة المطلقة لأي عدد وترجعه من نفس نوع البيانات المعطى للدالة والمقصود بالقيمة المطلقة هي قيمة العدد بدون إشارة فالقيمة المطلقة ل (-١٣) مثلا هي (١٣) وهكذا، فمثلا لو كتبنا الكود التالي
رمز:

```
MyNumber=Abs(-45.6) Text1.Text=MyNumber
```

فإن نتيجة تنفيذ الدالة هيMyNumber=45.6
ولاحظ أن القيمة المدخلة للدالة لابد أن تكون عدد أو تعبير عددي فإذا كانت القيمة المدخلة للدالة Nullستكون النتيجة Null وإذا كانت القيمة المدخلة للدالة متغير فارغ أو لم يتم تعيين قيمة له ستكون النتيجة (٠)

الدالة Sqr :تستخدم هذه الدالة في تحديد الجذر التربيعي لرقم معين وتأخذ الصورة العامة التالية:

رمز:
MyNumber=Sqr(10) Text1.Text=MyNumber
فإن نتيجة تنفيذ الدالة هيMyNumber=3.1622776

الدالة : Log تستخدم هذه الدالة في تحديد قيمة اللوغاريتم العشري لرقم وتأخذ الصورة العامة التالية:

رمز:
MyNumber=Log (20)Text1.Text=MyNumber
فإن نتيجة تنفيذ الدالة هيMyNumber=2.9957327

الدالة Exp : تستخدم هذه الدالة في تحديد القيمة (e) وهي قاعدة اللوغاريتم الطبيعي مرفوعة بقوة الرقم الذي تتضمنه حيث (e) تساوي تقريبا ٢,٧١٨٢٨١٨ وتأخذ الصورة العامة التالية:

رمز:

MyNumber=Exp (رقم)

الدالة Rnd : وتستخدم هذا الدالة في توليد أرقام عشوائية تقع ما بين الصفر و واحد بحد أقصي ١٥ رقما عشريا وتأخذ الصورة العامة التالية:

رمز:

MyNumber=Rnd (عدد)

فمثلا الدالة Rnd(10) قد تعطي رقما مثل ٠,٧٠٥٥٤٧٥ وعند تشغيل الدالة مرة أخري ينتج رقما آخر مثل ٠,٥٣٣٤٢٤ وهكذا.
التصريح : Randomize يعمل هذا التصريح مع الدالة Rnd ونستفيد منه عدم التكرارية و

الحصول علي عشوائية افضل وذلك لأنه يعتمد علي ساعة النظام لديك . ويأخذ الصورة التالية.

رمز:

RandomizeMsgBox Int((10 * Rnd) + 1)

وهنا نعمل علي توليد أرقام عشوائية من واحد إلى عشرة.
أما إذا أردت أن تجعل هذه العشوائية نصوص وليس أرقاما فلك طريقتين لتحايل علي هذا أما أن تضع النصوص في قاعدة بيانات وفي شكل سجلات وتعضي لكل سجل رقم أو تضعه في مصفوفة وتعضي لكل نص رقم أيضا وأنا افضل الطريقة الأولى لسرعتها

الدالة Int :وتستخدم هذه الدالة لحساب الجزء الصحيح فقط من رقم يشتمل علي أرقام صحيحة وعشرية أو بعبارة أخر لحذف الأرقام العشرية الموجودة بعد العلامة العشرية بدون تقريب وتأخذ الصورة التالية:

رمز:
MyNumber=Int (332.54)
فإن نتيجة تنفيذ الدالة هيMyNumber=332

الدالة : Atn : تستخدم هذه الدالة في حساب مقلوب ظل الزاوية "ظتا" للرقم الذي تشتمل علية مقدار بالتقدير الدائري وتأخذ الصورة العامة التالية:

رمز:

MyNumber=Atn (رقم)

الدالة : : Tan تستخدم هذه الدالة في تحديد قيمة ظل زاوية معينة وتأخذ الصورة العامة التالية:

رمز:

MyNumber=Tan (رقم)
وتستخدم هذه الدالة في تحديد قيمة جيب تمام الزاوية معينة وتأخذ الصورة العامة Cos : الدالة التالية:

رمز:

MyNumber=Cos (رقم)

الدالة : Sin تستخدم هذه الدالة في تحديد قيمة جيب زاوية معينة وتأخذ الصورة العامة التالية:

رمز:

MyNumber=Sin (رقم)

الدالة : Round وهي دالة التقريب التي من خلالها يمكنك تحديد عدد الأرقام العشرية وتأخذ الصورة التالية:

رمز:
MyNumber= Round(4.52696,2)
حيث ٤,٥٢٦٩٦ الرقم المراد تقريبه أما الرقم ٢ فهو عدد التي ستقرب بعد العلامة العشرية وسيكون الناتج ٤,٥٣

الدالة Fix : وهي تشبه الدالة Int تماما أي أنها تستخدم لحساب الجزء الصحيح فقط وتأخذ الصورة التالية:

رمز:
MyNumber= Fix(4.52696)
فإن نتيجة تنفيذ الدالة هيMyNumber=4

ثانيا : دوال سلاسل البيانات String Functions .

الدالة : Array تحول عدة بيانات مدخلة كنصوص تفصلها فاصلة إلى مصفوفة يكون رقم أول عنصر فيها (Lower Bound) هو (٠) فمثلا لو أدخلنا النص التالي إلى الدالة.

كود:

```
"Frist","Second","Third","Forth"
MyNumber= Array("First" , "Second" , "Third" , "Forth" )
Print MyNumber(0)
Print MyNumber(1)
Print MyNumber(2)
Print MyNumber(3)
```

نرى أن النتيجة هي

كود:

```
First
Second
Third
Forth
```

أي أن المتغير MyNumber يحمل مصفوفة ممتلئة بالنصوص المدخلة بعد أن تحولت إلى عناصر للمصفوفة

لاحظ أن المتغير MyNumber كان من نوع Variant لأننا لم نعلن عنه ويمكن للمتغير من هذا النوع أن يحمل مصفوفة وكذلك يمكن أن تكون المصفوفة من أي نوع آخر فالمصفوفة المستعملة في المثال هي مصفوفة نصية بينما لو كانت العناصر المدخلة أرقما (لاستعملنا علامات التنصيص مع الأرقام حتى لا تعتبر نصوصا) لكانت مصفوفة من نوع Integer مثلا.

الدالة : Asc ترجع كود الرمز المدخل (ASCII Code) فمثلا الحرف A له الكود ٦٥ فعند إدخال حرف A مثلا ستكون النتيجة ٦٥ وهكذا (لاحظ أن حرف A يختلف عن حرف a في الكود)

كود:

```
MyNumber = Asc("a")
```

تجد أن المتغير MyNumber أصبح يحمل القيمة ٩٧ . وهكذا .. انتبه فيما لو غيرة الحرف من صغير إلى كبير ستتغير القيمة إلى ٦٥

الدالة : UCase وظيفتها بسيطة جدا فهي ببساطة تحول حالة النص المدخل من حالة الأحرف الصغيرة (Small Letters) إلى حالة الأحرف الكبيرة(Capital Letters)

فالمثال التالي يحول "taftaf1267" إلى"TAFTAF1267"

كود:

```
1   Ucase("taftaf1267")
```

الدالة : LCase تعيد الدالة LCase نسخة من النص string تكون فيه جميع الحروف صغيرة Lowercase المتغيرة myText في المثال التالي ستحتوي على عبارة: it works

كود:

```
1   myText = "It Works"
2   myText = LCase(myText)
```

الدالة : chr تقوم الدالة chr بأخذ قيمة بين ٠ و ٢٥٥ وتعيد الحرف الممثل لهذه القيمة في جدول رموز ASCII ، على سبيل المثال العبارة التالية :

كود:

```
1   Hi,
2   I'm "TafTaf"
```

ستجد أنك لا تستطيع كتابتها بهذا الشكل:

كود:

```
1   myText = "Hi," & vbCrLf & "I'm "TafTaf""
```

لأن البرنامج سيعتقد بأن نهاية السلسة النصية السابقة هي عند علامات الاقتباس التي تقع مباشرة قبل كلمة TafTaf وستظهر لك رسالة خطأ، لذلك فإننا نلجأ لاستخدام الدالة chr حيث أن رمز علامة الاقتباس المزدوجة في جدول ASCII هو ٣٤، فتكون الصياغة الصحيحة للعبارة البرمجية السابقة كالتالي:

كود:

```
1   myText = "Hi," & vbCrLf & "I'm " & chr(34) & "TafTaf" & chr(34
```

الدالة : Len ستجد فيما بعد حاجة في كثير من الأحيان لمعرفة طول السلسلة النصية (عدد الأحرف)، ولعمل ذلك استخدم الدالة Len ، مرر إليها النص وستعيد لك عدد الحروف.

كود:

```
1   myLength = Len("TafTaf")
```

الدالة : InStr يمكنك بواسطة هذه الدالة معرفة أول مكان يظهر فيه نص ما ضمن نص آخر أكبر منه. الوسيطة الأولى هي وسيطة اختيارية تحدد مكان بدء البحث، أما الوسيطة الثانية string1 فتحدد السلسلة النصية التي سيتم البحث فيها، والوسيطة الثالثة string2 تحدد السلسلة النصية التي سيتم البحث عنها في السلسلة الأولى، أما الوسيطة الأخيرة فهي اختيارية أيضا وتحدد نوع المقارنة التي يجب إجرائها وهي تأخذ أحد الثوابت التالية:

0 - vbBinaryCompare
1 - vbTextCompare

والفرق بينهما هو أن الأولى تراعي حالة الأحرف والثاني لا تراعي حالة الأحرف.
في المثال التالي الدالة j ستحتوي على القيمة ١:

كود:

```
1   i = InStr("TafTaf","T")
```

وأما المثال التالي فستحتوي i فيه على القيمة ٣:

كود:

```
1   i = InStr("aafTaf","T")
```

حيث أن الدالة في المثال السابق ستبحث عن الحرف T بادئه من الحرف الثاني ولذلك فهي لن تجد الحرف الأول.

الدالة : InStrRev وهي شبيهه بدالة (InStr وقد سبق شرحها من قبل) ولكن تكون عملية البحث عن الحرف عكسية أي تبدأ من نهاية القيمة . وهي حساسة لحالة الأحرف وتأخذ الصورة التالية:

كود:

```
1   Print InStrRev("Mostafa", "a")
2   Print InStr("Mostafa", "a")
```

بالنسبة للحالة الأولى سيكون الناتج رقم ٧ لأنها تبدأ من نهاية القيمة.
أما في الحالة الثانية فسيكون الناتج ٥ لأنها تبدأ من بداية القيمة.

الدالة : Str قد تبدو الدالة Str متشابهة مع الدالةchr ، إلا أنها تؤدي وظيفة مختلف تماما، فهي تحول الأرقام إلى سلاسل نصية، وهي تفيد مثلا في حال أردت أن تقوم بالتحام بين رقمين فتقوم بتحويل كل منهما إلى سلسلة نصية وتطبق بينهما جمع السلاسل (&) الذي يختلف عن جمع الأرقام وبالتالي تحصل على سلسلة جديدة يمكنك أن تحولها إلى رقم من جديد، على سبيل المثال الدالة myNumber تحتوي على القيمة ١٢٣٤٥٦.

كود:

```
1   myNumber = Str(123) & Str(456)
```

ستواجهك مشكلة في الشفرة السابقة حيث يقوم فيجوال بيسيك بإضافة مسافة قبل كل سلسلة نصية تنتج من الدالة السابقة.

الدالة : Val وهي تحول السلاسل النصية إلى قيمة رقمية وهي عكس الدالة (Str التي تحول الأرقام إلى سلاسل نصية وتأخذ الصورة التالية:

كود:

```
1   myNumber = Val (Text1.text)
```

وهنا تعمل الدالة علي تحويل القيمة الموجودة في Text1 إلى قيمة رقمية.

الدالة : Left حيث تقوم بوضع سلسلة نصية string وتحدد الجزء الذي تريد اقتطاعه من بداية السلسلةlength ، والتعبير بكلمة من بداية السلسلة أدق من يسار السلسلة لأن هذا قد يحدث اشتباها في السلاسل النصية للغات التي تكتب من اليمين إلي اليسار مثل العربية، هنا تعيد السلسلة العدد المحدد من الحروف من اليمين أي أنها لا تهتم لاتجاه ظهور

أحرف السلسة وإنما اتجاه تخزينها، وللتخلص من هذه المشكلة سنقول بداية السلسلة.

الدالة : Right : مطابقة للدالة Left في كل شيء، إلا أنها تأخذ العدد المحدد من الحروف من نهاية السلسلة.

الدالة : Mid : تعيد الدالة Mid عددا من الأحرف قدره length بدءا من حرف معين هوstart ، في سلسلة نصية string لاحظ أيضا أن الوسيطة length اختيارية وإذا لم تمرر بها أي قيمة فإن الدالة ستعيد الأحرف إلى نهاية السلسلة.
في المثال التالي ستحتوي المتغيرة myText على العبارة: I'm Taf

كود:

```
1   myText = Mid("I'm TafTaf", 1, 7)
```

أما في المثال التالي فستحتوي على الكلمة: TafTaf

كود:

```
1   myText = Mid("I'm TafTaf", 5)
```

التصريح : Mid : يقوم التصريح Mid باستبدال مقطع محدد من النص بنص آخر، وهو يكتب في صورة مشابهة جدا لطريقة كتابة دالة Mid ولكن توضع بعده علامة مساواة وبعدها العبارة الجديدة، في المثال التالي ستحتوي المتغيرة myText على القيمة: I'm TafTaf

كود:

```
1   name = "TafTaf"
2   myText = "I'm name"
3   Mid(myText, 5) = name
```

حيث سيتم حذف الجزء المحدد بالخاصية Mid ويوضع الجزء الذي بعد علامة المساواة في مكان الجزء المحذوف.

الدالة : strReverse تعيد السلسلة string معكوسة، أي تبدأ من البداية وتنتهي من النهاية، المتغيرة myText في المثال التالي ستحتوي على العبارة: emoclew

كود:

```
1   myText = strReverse("welcome")
```

الدالة : Replace وتقوم باستبدال النص string2 بالنص string3 ضمن السلسلةstring1 ، أي أنها تبحث فبالنص string1 عن النص string2 وعندما تجده فإنها تحذفه منه وتضع مكانه string3، ويمكن تحديد نقطة بداية البحث بالوسيطةstart ، وعدد مرات الاستبدال القصوى بالوسيطةcount ، ضع القيمة -١ لاستبدال الكل، وطريقة المقارنة بالوسيطة compare كما ذكر سابقا.
المتغيرة myText في المثال التالي ستحتوي على القيمة : Hi Everyone

كود:

```
1   myText = Replace("Welcome Everyone,", "Welcome", "Hi", 1, -1)
```

الدالة : Format وتقوم هذه الدالة بتنسيق رقم أو سلسلة حروف أو التاريخ/الوقت الموجود في التعبير تبعا لتعليمات التنسيق الموجودة في الدالة نفسها . وتأخذ الصورة التالية:

كود:

```
1   MyDate = Format(Date, "dd-mmm-yyyy")
2   Text1.Text = MyDate
```

الدالة : Trim تقوم هذه الدالة بحذف الفراغات الموجودة في الجهة اليسري و اليمني من سلسلة وتأخذ الصورة التالية:

كود:

```
1   MyText = Trim("     TafTaf     ")
2   Text1.Text = MyText
```

الدالة : Rtrim تقوم هذه الدالة بحذف الفراغات الموجودة في الجهة اليمني من سلسلة وتأخذ الصورة التالية:

كود:

```
1   MyText = Rtrim("TafTaf     ")
```

الدالة : Ltrim وتقوم هذه الدالة بحذف الفراغات الموجودة في الجهة اليسري و اليمني من سلسلة . وتأخذ الصورة التالية:

كود:

```
1   MyText = Ltrim("     TafTaf ")
```

الدالة : Space وتقوم هذه الدالة سلسلة فراغات محدده بالعدد الموجود بين الأقواس

كود:

```
1   MyText ="I'm"+Space (2)+"TafTaf"
```

الدالة : StrComp تقارن سلسلتين لتحدد هل هما متساويتين ثم تعيد رقما بنتيجة المقارنة وتأخذ الصورة العامة التالية:

كود:

```
1   MyText =StrComp("TafTaf","TafTaf")
```

الدالة : Ascw وهي عكس الدالة Asc فالدالة Ascw ترجع كود الرمز المدخل إلى الترميز Unicodeوتأخذ الصورة التالية:

كود:

```
1   MsgBox Ascw("ت")
```

الدالة : Chrw وهي عكس الدالة Chr تعيد الحرف الممثل لهذه القيمة في الترميز Unicode وتأخذ الصورة التالية:

كود:

```
1   MsgBox ChrW(1578(
```

توضيح اكثر للدوال الأربع :

كود:

```
1   Private Sub Command1_Click()
2   MsgBox Asc("ت")
3   MsgBox AscW("ت")
4   MsgBox Chr(202(
5   MsgBox ChrW(1578)
6   End Sub
```

الدالة : String وتعمل هذه الدالة علي تكرار الحرف المار أليها وناخذ الصورة التالية.

كود:

```
1   MyString = String(10, "T")
2   MsgBox MyString
```

هنا عملنا علي تكرار حرف T عشر مرات.

كود:

```
1   mystring = String(2, 13)
2   ss = "Welcome To V.B World" & mystring & "Welcome To V.B World"
3   MsgBox ss
```

ثالثا: دوال الوقت و التاريخDate and Time

الدالة : Now تعيد هذه الدالة تاريخ اليوم و الوقت الحالي باستخدام ساعة الحاسب الذي تعمل علية وتكون بالصورة التالية:

كود:

```
1   Label1.Caption = Now()
```

الدالة : Time تعيد هذه الدالة الوقت الحالي حسب ما هو مدون بساعة الحاسب الذي تعمل علية وتكون بالصورة التالية

كود:

```
1   Label1.Caption = Time()
```

الدالة: Date تعيد التاريخ الحالي حسب ما هو مسجل بالحاسب ويكون بالصورة التالية:

كود:

```
1   Label1.Caption = Date()
```

الدالة Day والدالة: Month تعيد الدالة ()Day رقما يقع بين ١ ، ٣١ يمثل ترتيب اليوم المقابل لقيمة تاريخية بينما تعيد الدالة ()Month رقما يقع بين ١ ، ١٢ يمثل ترتيب الشهر المقابل لقيمة تاريخية

كود:

```
Label1.Caption =Day(Date())

Label1.Caption =Month(Date())[/CODE]
```

الدالة : DateSerial تعيد هذه الدالة التاريخ المسلسل المقابل لليوم والشهر والسنة المدون بين الأقواس وتأخذ الصورة التالية:

[CODE]GetDate=DateSerial(2003,4,11)

MsgBox GetDate

الدالة : TimeSerial تعيد هذه الدالة الوقت المسلسل المقابل للساعة و الدقيقة و الثانية المدونة بين الأقواس وتأخذ الصورة التالية:

كود:

```
1  Get_Time=TimeSerial(12,59,20)
2  MsgBox Get_Time
```

الدالة : DateValue تعيد هذه الدالة التاريخ المقابل لتعبير معين وتأخذ الصورة التالية:

كود:

```
1  RelVal = DateValue(Now – 1)
2  MsgBox RelVal
3  Select Case RelVal
4      Case Is < Int(Now): Verb   "كان "=
5      Case Is > Int(Now): Verb =  "سيكون"
6      Case Else: Verb =  "اليوم هو"
7  End Select
8  WhatDay = Format(RelVal, "dddd")
9  MsgBox UserDate & Verb & WhatDay
```

يبن هذا المثال تحديد يوم من أيام الأسبوع (أمس) بعد تحديد قيمة تاريخ اليوم (بفرض : الاثنين الموافق ٢سنة ١٩٩٧) ثم طرح (١) منة يظهر الناتج في صورة رسالة بان أمس "كان الأحد. "

دوال :Hour,Minute,Second,TimeValue

كود:

```
1  Hour(رقم)
2  Minute(رقم)
3  Second(رقم)
4  TimeValue(تعبير)
```

تعيد هذه الدوال قيم الساعة ، الدقيقة ، الثانية أو الوقت.

كود:

```
1   Midnight=TimeValue("23:59:59")
2   HourDiff=Hour(Midnight)-Hour(Now)
3   MinuteDiff=Minute(Midnight)-Minute(Now)
4   SecondDiff=Scond(Midnight)-Second(Now)+1
5   If Second Diff=60 Then
6       MinuteDiff=MinuteDiff+1
7       SecondDiff=0
8   End If
9   If MinuteDiff=60 Then
10      HourDiff=HourDiff+1
11      MinuteDiff=0
12  End If
13  TotalMinDiff=(HourDiff*60)+MinuteDiff
14  Total SecDiff=(Total MinDiff*60)+SeconDiff
15  Msg="إجمالي الوقت المتبقي علي منتصف الليل هو"& Format(Total SecDiff,"#,##0")
```

```
16  Msg=Msg&"ثانية ، يمكن ترجمتها إلي"
17  Msg=Msg & HourDiff & " ساعة، " & MinuteDiff
18  Msg=Msg & " دقيقة، و " & SecondDiff & "ثانية"
19  Msg BoxMsg
```

يستخدم هذا المثال الدوال السابقة لتحويل الوقت المتبقي علي منتصف الليل إلى ثواني ثم ترجمة هذا الوقت إلى ساعات ودقائق وثواني في رسالة كهذه "إجمالي الوقت المتبقي علي منتصف الليل هو ٧٩٩٧٢ ثانية يمكن ترجمتها إلى ٢٢ ساعة ١٢ دقيقة ٢٥ ثانية"

الدالة : Weekday تعيد هذه الدالة ترتيب اليوم بين أيام الأسبوع من التاريخ المدون بحقل التاريخ أو القيمة التاريخية باعتبار أن يوم الأحد رقم ١ والاثنين ٢ ... وهكذا

كود:

```
1  Label1.Caption = Weekday(#1/4/97#)
```

الدالة : Year تعيد هذه الدالة رقما يعبر عن السنة لتاريخ معين.

كود:

```
1  Label1.Caption = Year(#1/4/97#)
```

الدالة : DateAdd تعيد هذه الدالة تاريخ جديد بعد أن تضيف له قيمة جديدة وبصرف عن اسم هذه الدالة فهي تعمل مع كل من التاريخ و الوقت . ويوضح الجدول التالي المقصود بالمعامل "الفاصل الزمني " و القيم التي يمكن تخصيصها له:

الفاصل الزمني	التوضيح
yyyy	سنة
q	ربع سنة
m	شهر
y	يوم في سنة
d	يوم
w	يوم من الأسبوع (الأحد ١ ، والاثنين ٢ ... وهكذا
ww	أسبوع
h	ساعة
n	دقيقة
s	ثانية

كود:

```
1   Label1.Caption = DateAdd("yyyy", -10, Date)
```

وفي هذا المثال تعيد هذه الدالة السنة الحالية ٢٠٠٣ إلى ١٠ سنوات مضت ١٩٩٣

الدالة : DateDiff تقوم هذه الدالة بإعادة الفرق بين تاريخين.

```
Label1.Caption = DateDiff("y", 2000, 2003(
```

بينما المثال التالي يحدد عدد الأسابيع وعدد الأيام ما بين أول أيام سنة ٢٠٠٣ وتاريخ اليوم

كود:

```
1   Label1.Caption = DateDiff("ww", #4/5/2003#, Now())
2   Label2.Caption = DateDiff("y", #4/10/2003#, Now())
```

الدالة : DatePart تقوم هذه الدالة بإعادة جزء من التاريخ (مثل اليوم أو الشهر أو الأسبوع أو الساعة) ويتم تعين هذا الجزء بواسطة الفاصل الزمني.
تعيد الدالة في هذا المثال رقم اليوم المحدد في حقل تاريخ الاعتماد(OrderDate)

كود:

```
1   Label1.Caption = DatePart("w", OrderDate)
```

الجمل الشرطية

لا شك أن جميع المبرمجين يدركون أهمية الجمل الشرطية في بناء شيفرة البرنامج أو النص البرمجي، وعلى الأخص الجملة ذات البادئة... IF

وكثيرا ما نصادف الألغاز التي تتمحور فكرتها حول تنفيذ مهمة معينة دون الاعتماد على الجمل الشرطية، وهذا ما

يقودنا إلى أهمية هذه الجمل في تكوين الشيفرات...

أولا: الجملة الشرطيةIF

استخدامها:

تستخدم الجملة الشرطية IF عند حاجتنا لاتخاذ قرارات مختلفة حسب شرط معين.

مثال :

إذا كانت درجة الطالب أكبر من أو يساوي ٦٠ فهو ناجح ، وإذا كانت أقل من ٦٠ فهو راسب

أ) صيغةIF – THEN

IFالشرط المطلوب اختباره THEN الأمر الذي سيتم تنفيذه اذا كان الشرط صحيح

مثال:

```
IF grade>=60 then result = "ناجح"
```

ب) صيغةIF – THEN – ENDIF

```
IF    الشرط المطلوب اختباره THEN
الأوامر أو الجمل التي سيتم تنفيذها اذا كان الشرط صحيح
ENDIF
```

مثال:

```
IF grade>=60 then
"ناجح result = "
"مبروكTEXTBOX1.TEXT="
ENDIF
```

ج) صيغةIF – THEN – ELSE

IFالشرط المطلوب اختبارهTHEN

مجموعة الجمل لتي سيتم تنفيذها اذا كان الشرط صحيح

ELSE

مجموعة الجمل التي سيتم تنفيذها إذا كان الشرط خاطيء

ENDIF

مثال:

IF grade>=60 then

result = "ناجح"

ELSE

result =" غير مجتاز "

ENDIF

د) صيغةIF – THEN – ELSEIF

IFالشرط الأول المطلوب اختبارهTHEN

مجموعة الجمل لتي سيتم تنفيذها اذا كان الشرط الأول صحيح

ELSEIFالشرط الثاني لمطلوب اختبارهTHEN

مجموعة الجمل التي سيتم تنفيذها إذا كان الشرط الثاني صحيح

ELSE

مجموعة الجمل التي سيتم تنفيذها إذا كانت كل الشروط خاطئة

ENDIF

مثال:

```
IF grade>=90 then
 result = "ممتاز"
ELSEIF grade>=80 then
 result = "جيدجدا"
ELSEIF grade>=70 then
 result = "جيد"
ELSEIF grade>=60 then
 result = "مقبول"
ELSE
result =" غير مجتاز "
ENDIF
```

ثانيا: الجملة الشرطية SELECT CASE

تستخدم هذه الجملة إذا كان هناك عدة احتمالات للشرط ، فهي تقوم بنفس عمل جملة IF ولكن بطريقة أسهل.

الصيغة العامة لها:

SELECT CASEالتعبير المراد اختبار قيمته أو تقييمه

CASEالاحتمال الأول

يتم تنفيذ مجموعة الجمل1 -

CASEالاحتمال الثاني

يتم تنفيذ مجموعة الجمل2

CASE ELSE

الجمل التي سيتم تنفيذها في حالة عدم تحقق أي شرط من الشروط السابقة

END SELECT

مثال علىSELECT CASE

```
SELECT CASE grade
CASE 90 TO 100
Result = " ممتاز "
CASE 80 TO 89
Result = "جيد جدا"
CASE 70 TO 79
Result = " جيد "
CASE 60 TO 69
Result = " مقبول "
case else
Result = " غير مجتاز "
END SELECT
```

جمل التكرار:

جمل التكرار -: باختصار هي جزء من البرنامج يتكرر
فمثلاً اذا اردنا ان نضيف على Combo box ال قائمة المنسدلة الارقام من 0 إلى100
فهل من المعقول ان نقوم بكتابة 100 سطر لأجل ذلك ؟؟
بالطبع لا....
وسينتج لنا كود كبير جداً وبدون اي فائدة منه :- مثل

: كود

```
ComboBox1.Items.Add(1)
ComboBox1.Items.Add(2)
ComboBox1.Items.Add(3)
.
.
.
ComboBox1.Items.Add(99)
ComboBox1.Items.Add(100)
```

سنجرب الان ان نضيف اليه الارقام ولكن باستخدام جملة for مثلاً-:

: كود

```
For i = 1 to 100
ComboBox1.Items.Add(i)
next
```

بهذه الكود هنا سوف يقوم بعمل اضافة للأعداد من ١ إلى ١٠٠ ولا حظ ان كلمة next تأتي مع حلقة for دائماً

أقسام جمل التكرار:
-1جمل التكرار باستخدام For.... Next
-2جمل التكرار باستخدام Do... Loop

اولاً :- جملة التكرارfor --- next

يمكننا استخدام هذه الجملة عندما نريد تكرار التعليمة حسب المراد

فمثلاً كما في المثال السابق حددنا الارقام من ١ إلى ١٠٠ وبهذا صار التكرار معلوماً لدينا

وتكون الصيغة العامة لجملة التكرار for .... next بالشكل التالي-:

كود:

```
For i = (StartValue) to (EndValue)
Condition
Next i
```

مثلاً الكود التالي يستخدم لإضافة مصفوفة جديدة ومن خلال جملة التكرار اريدها ان تضع
الاسماء التي بداخل المصفوفة
وتكتب في الكونسول

: كود

```
Dim vb4rab() As String = {"abdullah", "ali", "mohammed", "samer",
"some one"}

For i = 0 To 4

Console.WriteLine("the name is  " & vb4rabrab(i))

Next
Console.ReadKey()
```

شرح الشفرة:

- قمنا بتعريف مصفوفة جديد وادخلنا فيها القيم التالية

: كود

{"abdullah", "ali", "mohammed", "samer", "some one"}

for استخدمنا الكلمة المحجوزة

قمنا بتنفيذ تكرار جديد واعطينا قيمة له هي ان يبدأ بالعد من الصفر وينتهي ب اربعة
اي ٠ ١ ٢ ٣ ٤ اي خمس مرات

- الصفر رقم منفصل
- الواحد رقم منفصل
- الاثنين رقم منفصل
- الثلاثة رقم منفصل
- الاربعة رقم منفصل

يمكنك تجريب النتيجة بنفسك الان لعمل ذلك
ولا حظ هنا اننا كتبنا هذا الكود

: كود

Console.WriteLine("the name is " & vb4arab(i))

وهو الكود الذي سيقوم بتكرار نفسه خمس مرات على التوالي

ملاحظة هامة جداً-:

سيقوم تكرار العد على العداد الذي نقوم بتعريفه في جملة التكرار
فمثلاً قيمة i
لأول مرة سيكون0
وبعد وصوله للكلمة المحجوزةnext
سوف يعود لبداية التكرار
بعد ان تصبح قيمته ٢
وهكذا حتى تصل إلى خمسة
اذا وصل العداد إلى الرقم 5 خرج من جملة التكرار

ونفذ الشفرات التالية...

ولا تنسوا انه يمكننا الحصول على القيمة التي بداخل اي مصفوفة من خلال الكود التالي

كود:

```
Textbox1.text = vb4arab(1)
```

وسيظهر لنا في text اسم abdullah

حسنا! لماذا في جملة التكرار لم نقم بكتابة عدد داخل فهرس المصفوفة

كما تلاحظون كتبنا داخل فهرس المصفوفة العداد، الذي عرفناه على انه جملة تكرار تبدأ من الصفر وتنهي عند الاربعة، نعم المسألة بسيطة اكثر مما تتصور
فكما قلنا ان العداد سوف يبدأ من القيمة صفر وهي اول قيمة في المصفوفة ايضاً
لان المصفوفة أيضا قيمها تبدأ من العدد ر قم صفر وعند وصوله اول مرة إلى كلمةnext
سوف يذهب إلى الforوتكون قيمته1 وسيعرض لنا القيمة الاولى في المصفوفة التي تأتي بعد الصفر وهكذا.

ثانياً :- جملة التكرارDo...Loop

القسم الأول:توضيح الحلقةDo...Loop

هذا التكرار لا يقيدنا بقيم البداية أو النهاية بل يتحدد فقط بتحقق الشرط وبالنسبة للحلقة For..Nextنحدد مقدار الزيادة أو النقصان في مقدار العداد

الفرق بين ال for next و Do...Loop

الفرق كما ذكرنا في التعريف ان جملة ال for next قمنا بتحديد مقدار الاضافة ومقدار العداد له اما جملة التكرار do while فهي غير محددة وينتهي تكرارها بالاعتماد على الشرط الذي اعطيتها اياه.

مثلاً في هذا الكود هنا

كود:

```
For x = o to 2
Msgbox x
Next
```

ستظهر لنا ثلاث رسائل فيها التالي-:

الاولى رقم ١
الثانية رقم ٢
الثالثة رقم ٣
واما الكود التالي لجملة التكرار ال do ..Loop

كود :

```
Dim x As Integer
x = 0
Do While Not x = 4
MsgBox(x)
x = x + 1
Loop
```

سيقوم الكود التالي بطباعة المسج الرسالة 3 msgbox مرات على التوالي:

وبهذا //؟

اعتقد ان الفرق اصبح واضح الان

القسم الثانى: الحلقة Do...Loop

الحلقةDo While...Loop
الحلقةDo Until...Loop

أولا:الحلقةDo While...Loop

باختصار تقوم هذه الحلقة بتنفيذ التعليمات المحتواة طالما تحقق الشرط وعند فشل الشرط تتوقف هذه الحلقة
مثال بسيط:

كود:

```
Dim I As Integer = 0
    Do While I < 10
        Console.Write(I)
        I = I + 1

        Console.WriteLine()

    Loop
    Console.ReadKey()
```

الآن سيقوم البرنامج باختبار الشرط عند اول عملية تنفيذ وسيرى أن الشرط قد تحقق لان صفر اقل من عشرة وهكذا الى ان يصل الرقم تسعة فيزيد البرنامج ١ ويفشل الشرط وبهذا تنتهي الحلقة
والناتج:

0
1
2
3
4
5
6
7

8
9

ثانيا:الحلقةDo Until...Loop

تقوم هذه الحلقة بتنفيذ التعليمات داخل كتلة Do...Loop طالما لم يتحقق الشرط وبمجرد تحققه تتوقف الحلقة(اذن كما نرى فهي تعمل عكس الحلقةDo While...Loop
مثال:

كود:

```
Dim I As Integer = 0
    Do Until I >= 10
       Console.Write(I)
       I = I + 1

       Console.WriteLine()

    Loop
    Console.ReadKey()
```

كما نرى فان الشرط هو مضاد تماما للشرط في الحلقة السابقة لكن نتج نفس النتائج بالضبط!!كيف ذلك؟

الآن ستبدأ الحلقة باختبار الشرط ووجده خاطئ لذلك ينفذ التعليمات الى ان يتحقق الشرط وعندما يصل الرقم تسعة فبزيادة ١ تصبح عشرة وبالتالي يصبح الشرط صحيحا وتتوقف الحلقة
وناتج الحلقة هو

0
1
2
3
4
5
6

7
8
9

القسم الثالث: الفحص قبل الحلقة أو بعد الحلقة

الجزء الأول: الفحص المبدئي Pre-Test :
يقوم البرنامج بفحص الشرط ثم ينفذ الحلقة بعد التحقق من صحة الشرط
مثال: جميع أمثلة Do..Loop السابقة هي من هذا النوع من الفحص

الجزء الثاني: الفحص بعد تنفيذ الحلقة Posted-Test :
يقوم البرنامج بتنفيذ التعليمات لأول مرة ثم يقوم بعد تنفيذ التعليمات للمرة الأولى بفحص الشرط
أي أنه ينفذ التعليمات لأول مرة بالرغم من صحة الشرط المعطى:
مثال: نظام (هل تريد اعادة البرنامج في تطبيقات الكونسول)
الآن سأوضح الفكرة بهذا الكود:

كود:

```
Dim again As String

    Do
       Console.Write("Do you Want to re-start The Program?(Y/N)")
       again = Console.ReadLine()
    Loop While (again = "Y" Or again = "y")
```

الآن نخبر البرنامج أول مرة أن يقوم بتنفيذ التعليمات وبعد ذلك يقوم البرنامج بالتحقق من قيمة againالتي أدخلناها وقت التنفيذ فاذا كانت تساوي y أو Y فنخبره بتكرار الحلقة وان لم يكن يتخطى الحلقة

ثالثاً:

طبعاً النوع هذا لا يعتبر نوع بحد ما يعتبر طريقة استثنائية ثانية فمثلاً اذا اردنا ان نقوم بتعبئة الكونسول 10 مرات ali بدون ذكر اي من الكلمات for اوdo while
فسنعمل طريقة التالية لتعبئة الكونسول بدون استخدام اي من جمل التكرار المعروفة
وهذه هي الطريقة_:

كود:

```
Dim s As Integer

s = 0
lo2i:

s = s + 1

Console.WriteLine("ali")

If s < 5 Then GoTo lo2i

Console.ReadKey()
```

شرح الكود السابق-:

السطر الاول-:
كود:

```
Dim s As Integer
```

عرفنا متغير من نوع رقمي وللتعلم المزيد حول المتغيرات والثوابت ادخل هنا
السطر الثاني-:
كود:

```
s = 0
```

اسندنا قيمة للمتغير s وهي صفر.

السطر الثالث -:

كود:

lo2i:

السطر الخامس -:

كود:

```
Console.WriteLine("ali")
```

الكود الذي تستطيع من خلاله كتابة اي شيء في الكونسول
وهنا اخترنا اسم عليali

السطر السادس -:

كود:

```
If s < 5 Then GoTo lo2i
```

جملة شرطية واعتقد انها واضحة بعد ما قمت بقراءة هذا الدرس هنا ومضمون هذه الجملة الشرطية انه اذا كانت قيمة المتغيرs اذهب إلى كلمة لؤيlo2i وابدأ بالقراءة من هناك...
**********.

السطر السابع -:

كود:

```
Console.ReadKey()
```

هذا الكود اذا لم تضعه فسوف تفتح لك شاشة الكونسول وتقفل بدون ان ترى اي شيء كتب فيها
... لذلك يجب عليك ان تقوم بكتابته لكي تستطيع القراءة من الكونسول

Goto اي انه باستخدام جملة ال
يمكنك البدء من سطر جديد في الكود ام من اجراء معين
يكتب بعدها : نقطتان رئسيتان Lo2i ولا حظ ان كلمة
-: كما هو موضح في الصورة التالية

<u>هذا مثال بسيط على الكونسول وجمل التكرار والمصفوفات والمتغيرات:</u>

مجموع الدرجات جميعها وسنخرج منها ايضاً سننشأ مصفوفة جديدة لنضع فيها درجات الطلاب
.. متوسط العلامات اي عدد الطلاب قسمة الدرجات العلامة الاكبر من بين العلامات

كود :

```
Dim marks(4) As Integer
Dim sum As Double
Dim avg As Double
Dim big As Double
sum = 0
avg = 0
big = 0
For x = 0 To 4
marks(x) = InputBox("ادخل الدرجة هنا ")
sum = sum + marks(x)
Next
For s = 0 To 4
Console.WriteLine("number of marks " & s & "  = " & marks(s))
Next
Console.WriteLine("----------------------------")
Console.WriteLine("the sum is  = " & sum)
avg = sum / 5
Console.WriteLine("----------------------------")
Console.WriteLine("the avg of numbers is  = " & avg)

For m = 0 To 4
If marks(m) > big Then big = marks(m)
Next
Console.WriteLine("----------------------------")
Console.WriteLine("the bigger number is  = " & big)
Console.WriteLine("----------------------------")
Console.ReadKey()
```

شرح الكود السابق-:

عرفنا مصفوفة من نوع رقم وفيها خمس مناطق للتخزين
لانها تبدأ من الصفر وتنتهي عند الاربعة
يعني ٠ ١ ٢ ٣ ٤

السطور التي بعدها هي لتعريف المتغيرات وهي
Sum & avg & big

= Sumليحمل بداخله مجموع الدرجات جميعها
= Avgليحمل بداخله متوسط الدرجات
= Bigليحمل بداخله اكبر درجة

واسندنا قيمة بدائية لكل من المتغيرات الثلاثة-:

اقتباس:
Sum=0
Avg=0
Big=0
اول جملة تكرار في الكود
عرفنا عداد اسمه x على انه يبدأ من صفر وينتهي عند الاربعة
الان بتدأ باسناد القيمة لكل من الغرف الموجودة في المصفوفةmarks
بهذا الكود

كود:

```
Marks(x)=inputbox("ادخل الدرجة هنا")
```

الان كبداية.. المتغير قيمته صفر ونحن نقول له
اجمع قيمتك القديمة التي هي صفر مع مجموع الدرجة التي ادخلت من قبل المستتخدم

الان سوف نقوم بكتابة جملة تكرار لتكرار مجموع الدرجات وطباعتها على الكونسول الخاص بنا

كود:

```
For s  0 To 4
Console.WriteLine("number of marks " & s & "  = " & marks(s))
Next
```

عرفنا عداد جديد للتكرار اسمهs
ويحمل بداخله قيمة من صفر إلى4

وسيقوم بإعطاء هذا الكود

كود:

```
Console.WriteLine("number of marks " & s & "  = " & marks(s))
```

لماذا الخمسة ؟؟؟؟ الخمسة مجموع القيم الموجودة في المصفوفة

و هي التي تبدأ من الصفر وتنتهي بالأربعة

وجملة التكرار الثالثة

هي لتحديد الدرجة الاكبر من خلال الدرجات المدخلة جميعها
اي عندما تكون
الدرجة اكبر من ال درجة التي قبلها سوف يحفظها في المتغير

وسنقوم استدعاء المتغير من خلال الكود التالي-:

كود:

```
Console.WriteLine("the bigger number is  = " & big)
```

طبعا لا ننسى الكود الذي لايعمل البرنامج بدونه
وهو ال
كود:

```
Console.readkey()
```

المصفوفات(Arrays)

ما هي المصفوفات: Arrays

المصفوفة هي عبارة عن سلسلة من البيانات من نفس النوع ، لتعريف Array من الارقام طولها ٥ عناصر نكتب الكود التالي:

كود:

```
Dim intarray As Integer() = New Integer(4)
```

يبدأ الترقيم في المصفوفات من الصفر وحتى ٤ ، لقراءة احد عناصر المصفوفة نكتب كود مثل التالي :

كود:

```
Console.WriteLine(intarray(3))
```

ولقراءة جميع العناصر يمكن استخدام حلقات التكرار بالشكل التالي:

كود:

```
For i As Integer = 0 To 4
   Console.WriteLine(intarray(i))
Next
```

تكوين المصفوفات:

الطريقة الاسهل لادخال البيانات إلى المصفوفة بالشكل التالي مثلاً:

كود:

```
array(0) = 15
array(1) = 20
array(2) = 13
```

أو عن طريق حلقه تكرار ايضاً . إلا ان هناك طريقه أخرى لادخال البيانات إلى المصفوفه بالشكل التالى مثلاً:

كود:

```
Dim intarray As Integer() = New Integer() {15, 20, 13}
```

المصفوفات متعددة الابعاد:

جميع المصفوفات السابقة هي مصفوفات احادية البعد one dimensional ، هناك انواع اخرى من المصفوفات ثنائية او متعددة الابعاد Multi dimensional ، هذا مثال على مصفوفة ثنائية الابعاد - تسمى باسم: - Matrix

كود:

```
Dim matrix As Integer = New Integer(2, 2)
```

سيكون شكل المصفوفة بالشكل التالي (افتراضي):

0 0 0
0 0 0
0 0 0

ويمكن ادخال البيانات إلى نقطة من المصفوفة بالشكل التالي:

كود:

```
matrix(1, 2) = 20
```

يمكن عمل حلقة تكرار لادخال البيانات ، وليكن عن طريق المستخدم بالشكل التالي مثلاً:

كود:

```
Dim matrix As Integer = New Integer(2, 2) {}
For i As Integer = 0 To 2
   For j As Integer = 0 To 2
      matrix(i, j) = Console.ReadLine()
   Next
Next
```

والطباعة بنفس الشكل ايضاً.

عمل مصفوفة من المصفوفات:

يمكن عمل مصفوفة يحتوي كل عنصر منها على مصفوفة بالشكل التالي:

كود:

```
Dim complexarray As Integer()() = New Integer(4)()
```

ويمكن الوصول لأي عنصر فيها عن طريق الكود التالي مثلاً:

كود:

```
Console.WriteLine(complexarray(1)(4))
```

وهذا ما يعني العنصر رقم ٤ من المصفوفة الأولى في المصفوفة. complexarray

يمكنك عمل دالة لطباعة محتويات مصفوفة بالشكل التالي:
كود:

```
Private Shared Sub print(ByVal arr As Integer())
   For i As Integer = 0 To arr.Length - 1
      Console.WriteLine(arr(0))
   Next
End Sub
```

افتراضياً ، لذا أي تعديل في الدالة سيؤثر على المصفوفة byref لا تنسى طبعاً انه يتم التعامل معها الأساسية .

: ، نكتب الكود بالشكل التالي return هي ما نود اعادته من الدالة array في حالة كون ال

كود:

```
Private Shared Function read() As Integer()

   Dim arr As Integer() = New Integer(2) {}
   For i As Integer = 0 To 2
      arr(i) = Console.Read()
   Next
End Function
```

خصائص المصفوفات الرئيسية:

تحتوي المصفوفات على بعض خصائص ودوال قد تساعدك في العمل عليها ، اشهرها واكثرها استخداماً هي الخاصية Lenght والتي تحدد طول عناصر المصفوفة ، الخاصية Rank تحدد عدد الابعاد في المصفوفة.

الدالة Sort تقوم بترتيب عناصر المصفوفة Reverse تقوم بعكس ترتيب عناصر المصفوفة ، وأخيراً الدالة ToString لتحويل المصفوفة إلى متغير نصي.

الدالة REDEM و الدالة : PRESEVE

تمكننا من تغيير حجم المصفوفة داخل البرنامج باستخدام الكلمة ReDim مع ملاحظة ان البيانات التي كانت موجودة سوف تلغى
Redim x( )
مع ذلك من الممكن الحفاظ على البيانات عند تغيير الحجم ولكن فقط من الممكن تغيير حجم البعد الاخير باستخدام كلمة Preserve
Redim Preserve x( )
من الممكن تدمير المصفوفة الديناميكية من الذاكرة باستخدام كلمةErase
Erase x
من الدوال التي تستخدم مع المصفوفات هي ()UBound تعود برقم العنصر الاخير في المصفوفة (x)UBound
*تستخدم هذه الدالة كثيرا فيloops