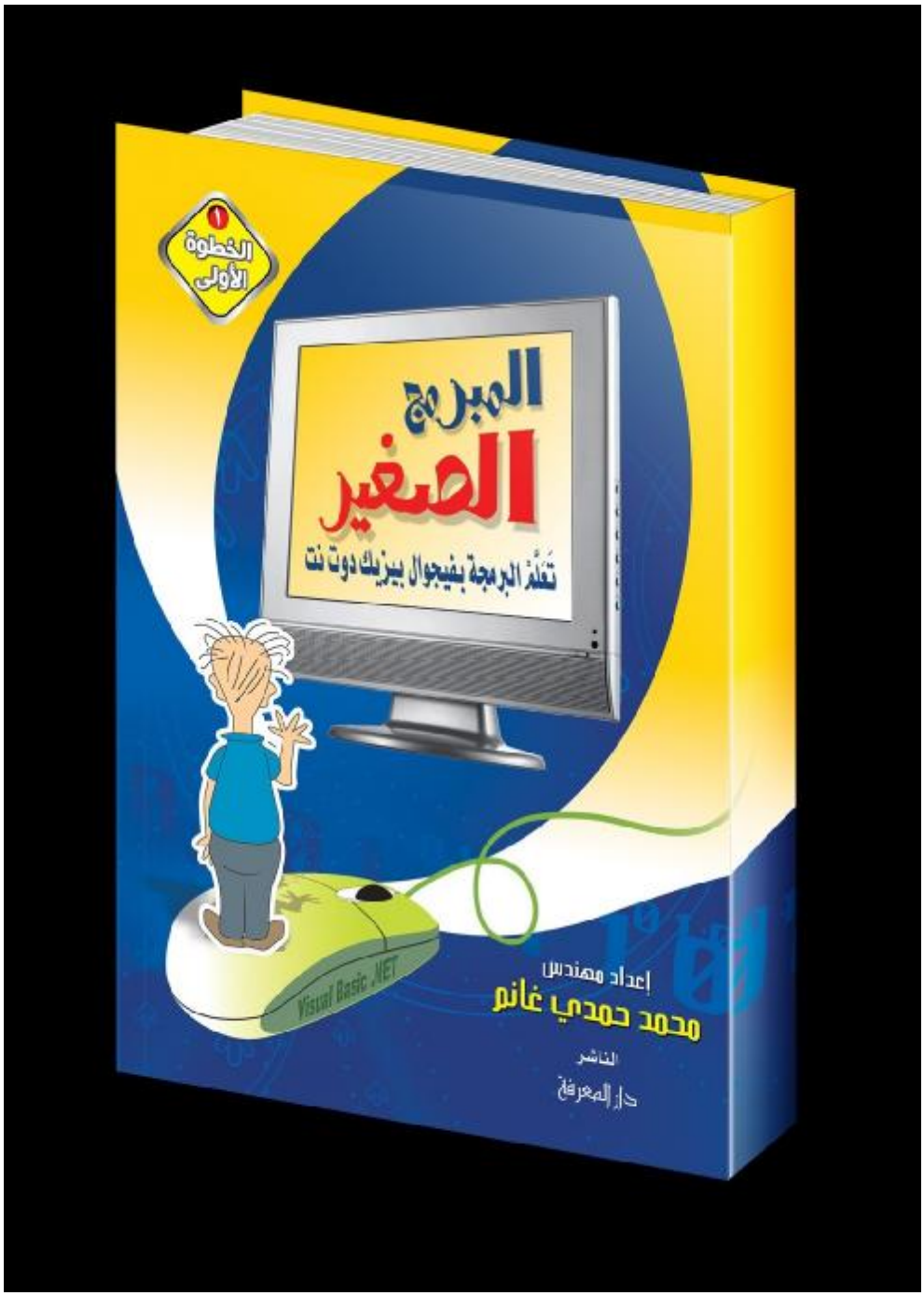

الخطوة الأولى
1
المبرمج الصغير
تعلّم البرمجة بفيجوال بيزيك دوت نت
Visual Basic .NET
إعداد مهندس
محمد حمدي غانم
الناشر
دار المعرفة

المبرمج الصغير

تعلم البرمجة بفيجوال بيزيك دوت نت

منهج الصف الثالث الإعدادي

الفصل الدراسي الأول

إعداد

م / محمد حمدى غانم

أسألكم الدعاء لأبي رحمه الله

اللهم اغفر لأبي واجعل مثواه الجنة

عن الكاتب:

- مهندس محمد حمدي غانم.
- من مواليد محافظة دمياط ١٩٧٧.
- خريج هندسة الاتصالات، جامعة القاهرة.
- عمل مبرمجا وكاتبا تقنيا، وله ١٤ كتابا متخصصا في البرمجة تشرح لغتي VB.NET و C#.

للتواصل مع الكاتب:

- بريدي الالكتروني:

msvbnet@hotmail.com

- مدونتي:

http://mhmdhmdy.blogspot.com

- قناتي على يوتيوب (تحتوي على إلقاء أكثر من ٦٠ قصيدة بصوتي):

http://www.youtube.com/user/mhmdhmdy

- صفحتي الأدبية على فيسبوك:

https://www.facebook.com/Poet.Mhmd.Hmdy

- كتبي في مجال البرمجة بلغتي فيجوال بيزيك وسي شارب:

http://mhmdhmdy.blogspot.com/2010/09/blog-post_9555.html

- صفحة فيجوال بيزيك وسي شارب:

https://www.facebook.com/vbandcsharp

السيرة الأدبية:

- نشرت له مسرحية "مجرّد طريقة للتفكير" في العدد ١٦ من "آفاق المسرح" من إصدارات قصور الثقافة، عام ٢٠٠٠، كما نشرت له مسرحية "بين قوسين من الخلود" ضمن إصدارات المنتدى الأدبي بجامعة القاهرة.
- صدر له ديوان "انتهاك حدود اللحظة"، عن مكتبة دار المعرفة، ٢٠١٠.
- صدر له ديوان "دلال الورد" عن قصر ثقافة دمياط، ٢٠١٣.
- يكتب القصص القصيرة والروايات الرومانسية وسلاسل الخيال العلمي، ونشر بعضها الكترونيا على شبكة المعلومات الدولية Internet.

كتب مجانية للكاتب للتنزيل:

- كتب برمجية على موقع كتب:

http://www.kutub.info/library/author/محمد%٢٠حمدي%٢٠غانم

- كتاب: "خرافة داروين، حينما تتحول الصدفة إلى علم":

http://mhmdhmdy.blogspot.com/2013/11/blog-post_29.html

- رواية "حائرة في الحب:

http://www.mediafire.com/?hd1jy6ca4ay3m9w

- رواية "حب في القطار (عمو)":

http://mhmdhmdy.blogspot.com.eg/2015/11/blog-post_39.html

- ديوان دلال الورد:

http://www.mediafire.com/?n1qte7j9hdv1l98

- ديوان فنجان وجع:

http://www.mediafire.com/download/gzivkgedtvx2e4j

- ديوان امرأة تسكن في زحل:

http://www.mediafire.com/download/o0lu67bfatdpqm7

- ديوان انتهاك حدود اللحظة:

http://www.mediafire.com/file/c5ctl13srqcvniy

لتحميل كتاب الأسئلة والإجابة للفصل الدراسي الأول:
http://www.kutub.info/library/book/2599

لتحميل كتاب الشرح للفصل الدراسي الثاني:
http://www.kutub.info/library/book/5804

لتحميل كتاب الأسئلة والإجابة للفصل الدراسي الثاني:
http://www.kutub.info/library/book/3393

أسألكم الدعاء لأبي رحمه الله

اللهم اغفر لأبي واجعل مثواه الجنة

-١-

مقدمة للبرمجة

قبل أن نتعلم لغة فيجيوال بيزيك، نحتاج إلى أن نتعرف أولا على عالم البرمجة الذي نخطو أولى خطواتنا فيه.. وفي هذا الفصل سنتعرف على بعض المفاهيم الأساسية والعمليات التي يستطيع الحاسب القيام بها.

نظام المعلومات Information System:
هو أي نظام يجمع بين الأفراد والحاسب الآلي، ويسمح بجمع وتخزين البيانات واستخلاص المعلومات منها.
وتمتلك معظم الشركات الحديثة نظام معلومات تستخدمه لحفظ بيانات الموظفين والعملاء والأجور والفواتير، مما يتيح لها الحصول على أية معلومات تريدها من هذه البيانات.

البيانات Data:
هي الحقائق المجردة التي يتم جمعها بواسطة نظام المعلومات، مثل:
- الأرقام: كدرجات الحرارة والأسعار ودرجات الطلاب في الاختبار.
- والحروف: كأسماء الطلبة والمدن والدول.
- والتواريخ: كتاريخ ميلادك.
- والصور: كصورة أخيك الصغير.
- والأصوات: كتلاوات القرآن الكريم.
- والفيديو: كالأفلام.

دعنا نأخذ نظام معلومات إحدى الشركات كمثال.. في هذا النظام تكون هناك الكثير من البيانات، مثل مواعيد الحضور والانصراف اليومية للموظف، وفواتير الواردات والصادرات للشركة، وبيانات العملاء المتعاملين مع الشركة، وهكذا.

المعلومات Information:
هي النتائج التي يمكن استنتاجها من البيانات بعد إجراء بعض العمليات عليها، كالحسابات الرياضية والمقارنة والترتيب وغير ذلك.. وتأخذ هذه المعلومات أشكالا متعددة، مثل:
- التقارير.
- الجداول.
- القوائم.
- الرسوم البيانية.

فلو قلنا مثلا إن: **أحمد طوله ١٥٠ سم**، فإن هذا بيان مجرد، لكن يمكن تحويله إلى معلومة إذا طبقنا عليه القاعدة التالية:
إذا كان طول الشخص أصغر من ٦٠ سم فهو قصير، وإذا كان أطول من ١٧٠ سم فهو طويل، وإذا كان محصورا بينهما فهو متوسط الطول.
لتطبيق هذه القاعدة نحتاج لمقارنة البيان الذي يمثل طول أحمد (وهو ١٥٠ سم) بالعددين (٦٠ و ١٧٠).. إذا فعلنا هذا فسنجد أن ١٥٠ أكبر من ٦٠ وأصغر من ١٧٠، وهذا يعطينا المعلومة التالية: **أحمد متوسط الطول.**
لاحظ أننا استطعنا إجراء هذه العملية بأنفسنا دون الحاجة إلى جهاز حاسب آلي، وذلك لأننا نتعامل مع بيان واحد فقط.. لكن ماذا لو أردت أن تعرف عدد سكان مصر متوسطي الطول؟
إن تطبيق عملية المقارنة على ٨٠ مليون شخص هي عملية مجهدة للغاية وتحتاج إلى جيش من الموظفين لإجرائها يدويا.. هنا تبرز بوضوح أهمية الحاسب الآلي ونظم المعلومات، فلو كانت كل بيانات أطوال سكان مصر مخزنة في نظام معلومات، فبضغطة زر واحدة يمكن الحصول على عدد سكان مصر متوسطي الطول، في زمن لا يتعدى بضع دقائق أو أقل!

***

ولو عدنا إلى مثال نظام معلومات الشركة، فسنجد أن الشركة تستطيع الحصول على المعلومات التالية من البيانات المخزنة في نظامها عن الموظفين والعملاء:

- عدد مرات غياب كل موظف الشهر.
- عدد ساعات العمل الإضافية التي عملها كل موظف في الشهر.
- أرباح وخسائر الشركة في آخر أربعة شهور.
- الفائض أو العجز في المخازن في هذا الشهر.
- الشركات التي تأخرت عن سداد ديونها للشركة خلال نصف العام الماضي.

وهكذا...

البرامج Programs:
هي مجموعة من الأوامر الموجهة إلى الحاسوب، لجعله يتعامل مع البيانات المقدمة إليه بالشكل المناسب للحصول على المعلومات المطلوبة.. هذه الأوامر تكون مكتوبة بصيغة معينة تختلف تبعا للغة البرمجة المستخدمة.
إن وورد Word مثلا هو برنامج مهمته كتابة الوثائق، بينما أكسيس Access هو برنامج مهمته إنشاء قواعد البيانات لتخزين البيانات وإجراء الاستعلامات عليها للحصول على المعلومات المطلوبة منها.. بل إن فيجيوال ستديو دوت نت VS.NET هي نفسها برنامج، وإن كانت مهمتها أن تمنحك عددا من لغات البرمجة (مثل فيجيوال بيزيك دوت نت) لتتيح لك كتابة البرامج الخاصة بك!
وللبرنامج عدة أسماء أخرى:

- فهو يسمى مشروعا Project.. هذا الاسم يطلق على البرنامج أثناء تصميمه وتنفيذه، لأن البرنامج ما زال "مشروعا" لم يكتمل بعد.
- ويسمى أيضا تطبيقا Application، لأن المبرمج يطبق وينفذ بهذا البرنامج فكرة معينة أو وظيفة ما.
- كما يمكن أن يسمى البرنامج حلا Solution، لأن المبرمج يستخدم البرنامج لحل مشكلة تواجهه.. لكن الحل Solution أعم من المشروع Project، فالحل يمكن أن يتكون من مشروع واحد أو أكثر من مشروع.

المبرمج Programmer:

هو الشخص الذي يحلل المشكلة، ويقترح خطوات حل لها يمكن تنفيذها بواسطة الحاسوب.. وهو يسمى أيضا بالمطور Developer، لأنه يطور البرامج والأفكار الجديدة.

ويمكن أن يكتب المبرمج الكود بنفسه باستخدام لغة البرمجة، وقد يكتفي بكتابة خطوات الحل وترك كتابة الكود لأي شخص يجيد إحدى لغات البرمجة، والذي يسمى بكاتب الكود Coder.

ويستطيع المبرمج أن يتعامل مع أنواع مختلفة من المشاكل في مجالات مختلفة وأن يقترح حلولا مبتكرة ومبدعة لها.. لا تنس أن البرمجة تتعامل مع كل مجالات الحياة، لهذا قد يواجه المبرمج مشكلة تتعلق بالهندسة أو الكيمياء أو الأحياء أو المحاسبة أو اللغات البشرية كاللغة العربية.. إلخ.. كما قد يبتكر المبرمج لعبة، أو برامج لإنشاء الرسوم الثابتة أو المتحركة، وغير ذلك من المجالات التي يمكن التعامل معها بالحاسوب.

ويواجه المبرمجون اليوم تحديات في المجالات التي يتفوق فيها المخ البشري على الحاسوب، مثل فهم المعاني والترجمة من لغة بشرية إلى أخرى، والتعرف على الكائنات الموجودة في الصور والفيديو، وتمييز الأصوات المتداخلة، وغير ذلك من المجالات التي تندرج تحت فرع الذكاء الصناعي.

وينبغي على المبرمج أن يجيد إحدى لغات البرمجة أو أكثر، لأنه كثيرا ما يكتب الكود بنفسه.. وحتى لو لم يكتب المبرمج أي كود بنفسه، فإن تعلم إحدى لغات البرمجة يجعله يدرك حدود الإمكانيات التي يتيحها له الحاسب، حتى لا يتجاوزها هو يبتكر حلا للمشكلة التي تواجهه.

لهذا لو أرت أن تكون مبرمجا جيدا، فعليك أن تضع أمامك الأهداف التالية:

١- تعلم لغة برمجة واحدة على الأقل، ولا مانع من تعلم أكثر من لغة إن أمكن.. ولا تنسَ أهمية التدريب المستمر بكتابة الكثير من البرامج بيدك، فهذا يعطيك خبرة كتابة الكود الأمثل وتخليصه من الأخطاء.. وها أنت ذا تدرس لغة برمجة سهلة وقوية وهي فيجيوال بيزيك دوت نت، لهذا كل ما عليك أن تغتنم الفرصة وتستمتع بها.. فالبرمجة أشبه بلعبة شطرنج، فيها تحدٍ وذكاء ومتعة، والمبرمج يشعر بلذة الفوز حينما ينجح في إنجاز برنامج تعب في كتابته وحل مشاكله.

٢- الاطلاع على خبرات وعلوم البرمجة، سواء بالدراسة المنهجية في الكليات المتخصصة ككلية الهندسة وكلية الحاسبات والمعلومات، أو حتى بالاطلاع الحر على علوم الحاسب والاستمتاع بالبرمجة كهواية، بجوار التخصص في مجال آخر.. هذا يتيح لك ابتكار برامج جديدة في تخصصك.. فمثلا لو كنت متخصصا في اللغة الفرنسية، فيمكنك أن تنتج برامج للترجمة من الفرنسية إلى العربية والعكس.. ولو كنت متخصصا في التاريخ، فيمكنك أن تبتكر برامج رسومية تعرض التاريخ بصورة شيقة.. ولو كنت طبيبا فيمكنك أن تنتج برامج خبيرة تستنتج المرض من مجموعة الأعراض التي يعاني منها المريض.. وهكذا.

٣- ومن الأفضل أن تجيد اللغة الإنجليزية بدرجة معقولة، وهذا أيضا متاح لك من خلال الدراسة، وليس عليك سوى أن تتقن ما تدرسه من اللغة الإنجليزية.. السبب في هذا أننا للأسف تابعون للغرب في البرمجة وكل العلوم الحديثة، والجديد دائما يظهر عندهم ونضطر نحن إلى القراءة فيه بلغتهم.. وإلى أن يظهر جيل عربي مسلم مبدع يعيد إلى اللغة العربية مجدها باعتبارها لغة العلم كما كانت في عهد الأندلس، فلا مفر من أن تتقن اللغة الإنجليزية إذا أردت التعمق في مجالات متقدمة في البرمجة، لأن هذه المجالات غير مغطاة في حركة الترجمة والتعريب إلى الآن بكل أسف.. أما إذا كنت ستكتفي باستخدام البرمجة في التطبيقات العملية والتجارية، فستجد ما تحتاجه من الكتب العربية في هذا المجال.

كما ترى.. كل المطالب متاحة وميسرة.. عليك فقط أن تحب البرمجة، وتعتبرها لعبة ممتعة وشيقة، وسيدهشك ما أنت قادر على إنجازه بها.

المستخدم User:

هو الشخص الذي يستخدم البرنامج الذي صنعه المبرمج، لهذا يسمى أيضا مشغل البرنامج.

أنت مثلا مستخدم لنظام الويندوز، ولبرنامج الوورد، ولفيجيوال ستديو دوت نت التي تكتب من خلالها برامج فيجيوال بيزيك.

مدخل البيانات Data Entry:

هو موظف أو مستخدم يعمل على نظام المعلومات، مهمته إدخال البيانات إليه وحفظها فيه.

مدير النظام System Administrator:

هو شخص مسئول عن إدارة نظام المعلومات، وهو يمتلك صلاحيات واسعة تتيح له التحكم في نظام المعلومات واستخدام كل إمكانياته، وحذف أية بيانات منه أو إضافتها إليه، والسماح للموظفين بالتعامل مع بعض إمكانيات نظام المعلومات من خلال اسم كل مستخدم وكلمة المرور الخاصة به والصلاحيات الممنوحة له.

الخلاصة:

لو شبهنا **البيانات** بالمادة الخام، فإن:

- **المعلومات** هي المنتَج الذي يتم تصنيعه من هذه المادة.
- **ونظام المعلومات** هو المصنع الذي تتم فيه هذه العملية.
- **والحاسبات** هي الآلات التي تشغل هذا المصنع.
- **والبرامج** هي تروس ومحركات آلات هذا المصنع!
- **ومدخلو البيانات** القائمون على تشغيل هذه البرامج وإدخال البيانات إليها هم عمال هذا المصنع.
- **والمبرمجون** الذين يطورون هذه البرامج هم المهندسون في هذا المصنع.
- **ومدير النظام** هو رئيس إدارة هذا المصنع.

تدريب:

أكمل الجمل التالية، باختيار الكلمة المناسبة من الكلمات التالية (بيان، معلومة، نظام معلومات):

- مشروع الحكومة الالكترونية في مصر هو .....
- الرقم القومي لأي مواطن في مصر هو .....
- عدد المواطنين الذين ينتمون إلى شريحة عمرية تقع بين ١٥ و ٣٠ سنة هو .....

العمليات التي يقوم بها الحاسوب:

يقوم الحاسوب بست عمليات أساسية، وهي:

١. إدخال البيانات (قراءة المعطيات).
٢. تخزين البيانات في الذاكرة الداخلية.
٣. إجراء عمليات حسابية على البيانات.
٤. مقارنة قيمتين واتخاذ القرار بناء على نتيجة المقارنة.
٥. تكرار تنفيذ أمر معين أو مجموعة من الأوامر لأي عدد من المرات .
٦. إخراج المعلومات (طباعة النتائج).

دعنا نتعرف على هذه العمليات بتفصيل أكثر:

إدخال البيانات (قراءة المعطيات):

أول خطوة في معالجة البيانات، هي تغذية الحاسوب بها.. ويمكن فعل هذا بأي من الوسائل التالية:

١. يدويا، بواسطة لوحة المفاتيح (كأن يكتب مستخدم البرنامج اسمه مثلا)، أو بواسطة الفأرة (كأن يختار المستخدم مدينته من قائمة بها أسماء المدن).

٢. آليا، باستخدام بعض الأجهزة التي تقرأ البيانات وتحولها إلى الصيغة الرقمية Digital التي يفهمها الحاسوب.. مثال هذا الماسح الضوئي Scanner الذي يحول الصورة الورقية إلى صور رقمية، وكارت الصوت Sound Card الذي يحول الصوت القادم إليه عبر لاقط الصوت Microphone إلى بيانات رقمية.. وكذلك الحال مع كارت الفيديو وكارت الشاشة وكارت طبق الاستقبال وكارت التلفاز، التي تحول الصورة إلى بيانات رقمية، سواء كانت قادمة عبر الكاميرا أو جهاز الفيديو أو هوائي استقبال الإرسال التلفزيوني أو طبق استقبال الإرسال الفضائي.. إلخ.
٣. كما أن البيانات قد تكون قادمة عبر وسيط تخزين تم حفظها عليه من قبل، مثل القرص المرن Floppy Disk، والقرص الصلب Hard Disk، والأقراص الضوئية (CDs أو DVDs)، أو الذاكرة النقالة Flash Memory.
٤. أو قد تأتي البيانات إلى الحاسب عبر شبكة معلومات خاصة Network أو شبكة المعلومات الدولية Internet، ويتم استقبالها عبر كارت الشبكة أو كارت الفاكس تبعا لنوع الاتصال المستخدم.

أيا كانت الطريقة المستخدمة، فالمهم هو أن يحصل الحاسب على البيانات اللازمة، في صورة رقمية يستطيع فهمها والتعامل معها.

لاحظ أن كل وسيلة من تلك التي شرحناها (مثل الفأرة ولوحة المفاتيح ومحرك الأقراص المرنة Floppy Driver ومحرك الأقراص الضوئية CD Driver، والكروت المختلفة) تسمى وحدة إدخال Input Unit، فمن خلالها تدخل البيانات إلى الحاسب.

تخزين البيانات في ذاكرة الكمبيوتر الداخلية:

يحتوي كل حاسوب على ذاكرة مؤقتة RAM.. معنى أنها مؤقتة هي أنها تفقد المعلومات الموجودة بها عند إغلاق الجهاز.. هذا هو السبب في احتياج كل حاسوب إلى أنواع أخرى من الذاكرة تستطيع أن تحفظ البيانات لفترات أطول، مثل القرص المرن Floppy Disk، والقرص الصلب Hard Disk والقرص الضوئي (CD أو DVD).

ومن المهم أن تعرف أن هناك فوارق كبيرة في سرعة كل نوع من أنواع هذه الوسائط.. ولو أردنا ترتيبها من الأسرع إلى الأبطأ فستكون كالتالي:

١- الذاكرة المؤقتة RAM.
٢- القرص الصلب Hard Disk.
٣- القرص الضوئي CD.
٤- القرص المرن Floppy Disk.

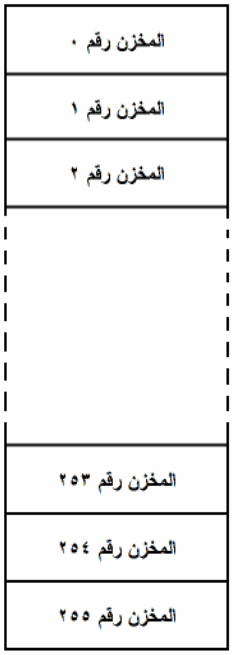
هذا الفارق في السرعات يفرض نفسه على طريقة عمل الحاسب، وكيفية كتابة المبرمج للبرامج.. فلكي يكون الحاسب بالسرعة الهائلة التي نعرفها، فعليه أن يجري كل عملياته الحسابية داخل الذاكرة RAM لما تمتاز به من سرعة كبيرة.. ونظرا لأن هذه الذاكرة صغيرة المساحة ولا تستطيع حفظ البيانات بعد إغلاق الجهاز، فإن تنفيذ العمليات في الحاسب يتبع الخطوات التالية:

١- تتم قراءة البيانات المراد معالجتها من وسيط التخزين (القرص الصلب، أو القرص المرن، أو القرص الضوئي) ونقلها إلى جزء من الذاكرة.
٢- يقوم الحاسب بالتعامل مع هذه البيانات في الذاكرة.. وقد يستخدم أجزاء أخرى من الذاكرة لوضع النتائج فيها إذا تطلب الأمر ذلك.
٣- بعد انتهاء العملية تماما، يتم نقل النتائج إلى الجهة النهائية.. قد يتم عرض هذه النتائج مباشرة على الشاشة أو طباعتها عبر الطابعة أو نقلها عبر الإنترنت، أو قد يتم إعادة حفظها على وسيط تخزين مرة أخرى (كالقرص الصلب، أو القرص المرن)، وذلك تبعا لكيفية عمل البرنامج الذي ينفذه الحاسب.

بهذا التنظيم، يضمن المبرمجون الحصول على أفضل كفاءة وأعلى سرعة من الحاسب.

لكن: كيف يتعامل المبرمج مع الذاكرة لحفظ وقراءة البيانات؟

تتكون الذاكرة من مجموعة من المخازن، يستطيع كل منها حفظ رقم محصور بين الصفر و ٢٥٥.. هذا المخزن يسمى بوحدة الذاكرة Byte.



رسم مبسط يوضح تركيب ذاكرة الحاسب

ولكل مخزن في الذاكرة عنوان.. هذا العنوان هو رقم.. مثلا: الذاكرة التي سعتها ٢٥٦ بايت، تحتوي على مخازن تبدأ من العنوان رقم صفر إلى العنوان رقم ٢٥٥.

وقديما كان المبرمج يستخدم لغات برمجة منخفضة المستوى Low Level Languages، وكان مضطرا إلى التعامل مع الذاكرة من خلال الأرقام التي تمثل عناوين مخازن الذاكرة.. بطبيعة الحال كان هذا الأمر يشكل صداعا للمبرمجين، خاصة مع التطور السريع في أحجام الذاكرة، والتي تجاوزت الشريحة الواحدة منها اليوم في الحاسب الشخصي ١ جيجيا بايت، أي ما يزيد عن مليار بايت، أو مليار مخزن!.. حاول أن تتخيل حجم المعاناة لو كنت مضطرا إلى التعامل مع مليار عنوان من عناوين الذاكرة!!

لمثل هذا جاءت لغات البرمجة عالية المستوى High Level Language، فعزلت المبرمج عن تركيب الذاكرة وباقي مكونات الجهاز المادية، ومنحته العديد من الأوامر السهلة التي تمكّنه من تنفيذ ما يريده بسرعةٍ وكفاءة.

والآن صار المبرمج يستطيع تسمية المخزن الذي سيستخدمه في الذاكرة باسم خاص به، بدلا من استخدام العناوين الرقمية.. هذا الاسم يسمى بالمتغيّر Variable، وهو يعتبر ساعي البريد بين المبرمج والذاكرة، حيث يمكن أن يكتب فيه المبرمج البيانات ويقرأها منه، دون أن يحتاج إلى معرفة عنوان الذاكرة الأصلي.

وبهذا صار كل المطلوب من المبرمج، أن يعطي المتغير اسما يدل على وظيفته في البرنامج ليسهل عليه فهمه وتذكره.. على سبيل المثال، يمكن تسمية متغير بالاسم Student_Name إذا كان سيستخدم ليحفظ في الذاكرة اسم الطالب، كما يمكن تسمية

متغير آخر بالاسم Grade إذا كان سيستخدم ليحفظ في الذاكرة، الدرجة التي حصل عليها الطالب في الامتحان.
ويستطيع المبرمج قراءة القيمة من المتغير، كما يمكنه تغييرها في أي وقت، ولأي عدد من المرات خلال البرنامج.. إن المتغير هو مخزن خاص بالمبرمج في الذاكرة، ومن حقه أن يغير محتويات هذا المخزن في أية لحظة على حسب احتياجه.. هذا هو السبب في تسمية المتغير بهذا الاسم، فالمبرمج يستطيع **تغيير** قيمته في أية لحظة كما يشاء.. يمكنك مثلا أن تضع في المتغير Student_Name الاسم "مجدي عادل"، ثم بعد ذلك تغيره إلى الاسم "أشرف محمود"، ثم إلى "محمد سعيد".. يحدث هذا في فيجوال بيزيك كالتالي:

```
' حجز متغير في الذاكرة
Dim Student_Name As String
' وضع قيمة في المتغير
Student_Name = "مجدي عادل"
' تغيير قيمة المتغير
Student_Name = "أشرف محمود"
' تغيير قيمته مرة أخرى
Student_Name = "محمد سعيد"
```

وسنتعرف على كيفية تعريف المتغيرات في فيجيوال بيزيك بالتفصيل لاحقا.

إجراء العمليات الحسابية على البيانات:
يستطيع الحاسوب القيام بالعمليات المنطقية والحسابية الأساسية.. العمليات المنطقية الأساسية هي:
- "ليس" Not.
- "و" And.
- "أو" OR.

والعمليات الحسابية الأساسية هي:
- الجمع Addition.
- الطرح Subtraction.
- الضرب Multiplying.
- القسمة Division.

وتعتبر العمليات المنطقية والحسابية كافية لمعالجة أي نوع من أنواع البيانات.. قد يصعب عليك تخيل هذا، لكنك ستجده أمرا منطقيا عندما تتذكر أن الحاسوب يتعامل مع الأرقام فحسب، وأن أي شيء نتعامل معه في الحياة الواقعية لا يمكن للحاسب التعامل معه إلا إذا

تم تحويله أولا إلى بيانات رقمية.. وسنتعرف على هذا بالتفصيل عند التعامل مع كل نوع من أنواع البيانات في فيجيوال بيزيك.

مقارنة قيمتين واتخاذ القرار بناء على نتيجة المقارنة:
يستطيع الحاسب فحص القيم والمقارنة بينها.. تسمى عملية المقارنة بين قيمتين "شرطا" Condition، والبرمجة الفعلية تبدأ من قدرة المبرمج على كتابة الشروط واتخاذ القرار المناسب عند تحقق كل شرط منها.. دعنا نأخذ حالة طول الشخص التي شرحناها سابقا كمثال.. لو أردنا أن نكتب فكرة هذا البرنامج فستكون كالتالي:

اقرأ الطول الذي يمدك به المستخدم.
إذا كان الطول أصغر من ٦٠ سم إذن:
اكتب على الشاشة: "الشخص قصير"
غير ذلك: إذا كان الطول أكبر من ١٦٠ سم إذن:
اكتب على الشاشة: "الشخص طويل"
غير ذلك:
اكتب على الشاشة: "الشخص متوسط الطول"
نهاية الشرط.

ما كتبناه أعلاه يسمى خوارزمية Algorithm، وهي نسبة إلى العالم المسلم "أبي بكر الخوارزمي" واضع علم الجبر.. هذه الخوارزمية تمثل طريقة تفكير منظمة وواضحة، للخطوات التي يجب اتباعها لمقارنة طول أي شخص، لمعرفة هل هو قصير أم طويل أم متوسط الطول.. لتقرير هذا احتجنا إلى ثلاث عمليات مقارنة (ثلاثة شروط)، وفي كل حالة منها اتخذنا قرارا مختلفا.
وسنتعرف لاحقا على كيفية كتابة الكود الذي ينفذ هذه الخوارزمية بلغة فيجيوال بيزيك.

الخوارزمية Algorithm:
هي مجموعة من خطوات التفكير، مرتبة ترتيبا منطقيا وواضحا، إذا تتبعناه نصل إلى حل المسالة التي نفكر فيها.
وينبغي على المبرمج أن يكتب خوارزمية البرنامج قبل أن يحاول كتابة الكود، لأن هذا يقلل من الأخطاء المحتملة عند البرمجة، كما أنه يساعد على الوصول إلى أبسط وأقصر وأسرع طريقة لتنفيذ المطلوب، بدلا من استخدام أكواد تضيع الوقت أو لا لزوم لها أصلا.

تكرار تنفيذ أمر معين أو مجموعة من الأوامر لأي عدد من المرات:
لو اقتصر الأمر على مقارنة عدة قيم واتخاذ القرارات المناسبة، لما كانت هناك حاجة إلى الحاسب، فالإنسان أفضل من الحاسب ملايين المرات في الفهم والاستيعاب والمقارنة

واتخاذ القرار.. لكن حاجة الإنسان إلى الحاسب تنبع أساسا من قدرة الحاسب على تكرار تنفيذ العمليات لآلاف وملايين المرات دون كلل أو ملل، وبسرعات هائلة.
تخيل مثلا لو أن الحكومة المصرية قررت طباعة مجلد تذكاري ضخم يضم كل أسماء الشعب المصري في القرن العشرين، وأن هذا الأمر كان سيتم يدويا، فكم ألف موظف في نظرك سيتطلبهم هذا الأمر، وكم سنة سيستغرقون لإنهائه؟.. بينما لو تم استخدام الحاسب، فلن يتطلب الأمر سوى موظف واحد لإعطاء الأمر للحاسب الذي يتعامل مع نظام معلومات الحكومة الالكترونية، لطباعة كل أسماء المواطنين المخزنة لديه، وسيعتمد وقت طباعة هذه النتائج على عدد الطابعات المستخدمة في هذه العملية وعلى سرعتها.
لاحظ أن المبرمج الذي كتب البرنامج الذي يطبع أسماء الشعب المصري لم يكتب ٨٠ مليون أمر لطباعة اسم كل فرد على حدة، لأنه لو فعل هذا، فلا ريب أنه كان سيبدأ كتابة هذا البرنامج منذ عهد الملك مينا موحد القطرين!!
لكن الأمر على العكس في منتهى البساطة، فكل المطلوب هو قراءة اسم أحد المواطنين من قاعدة البيانات وطباعته، ثم الانتقال إلى المواطن التالي وتكرار نفس الأمر إلى حين الوصول إلى آخر مواطن في قاعدة البيانات.. لو أردت أن تكتب هذا في شكل خوارزمية، فستبدو كالتالي:

انتقل إلى سجل المواطن الأول في قاعدة البيانات.
كرر ما يلي إلى حين الوصول إلى آخر موضع في قاعدة البيانات:
اقرأ اسم المواطن الموجود في السجل الحالي.
اطبع هذا الاسم.
انتقل إلى سجل المواطن التالي في قاعدة البيانات.
نهاية التكرار.

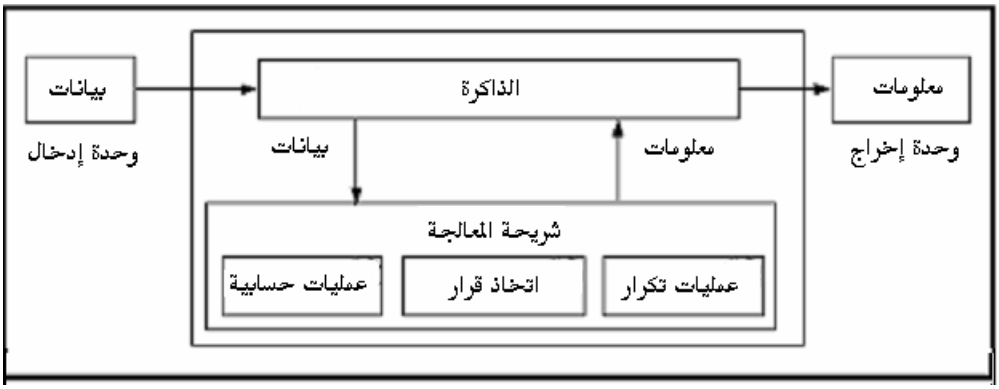
كما ترى: لن يحتاج المبرمج إلا إلى كتابة خمسة أوامر فحسب، لطباعة أسماء ٨٠ مليون مواطن.. هذا هو ما يميز الحاسوب عن كتبة الدواوين في عهد الملك مينا!
وسنتعرف لاحقا على كيفية كتابة كود التكرارات Loops في فيجيوال بيزيك.

إخراج المعلومات (طباعة النتائج):
بعد معالجة الحاسب للبيانات يتم الحصول على المعلومات المطلوبة.. قد يكتفي الحاسب عند هذه الخطوة بحفظ المعلومات الناتجة على أي وسيط تخزين (كالقرص الصلب أو القرص المرن)، وقد يحتاج المستخدم إلى الحصول على هذه البيانات.. ونظرا لأن المستخدم لا يهتم بالأرقام كما يهتم بها الحاسب، فإن على الحاسب أن يحول هذه الأرقام إلى الصورة التي يفهمها المستخدم.. يحدث هذا عبر ما يسمى بوحدات الإخراج Output Units، وهي المسئولة عن إخراج البيانات وتسليمها إلى المستخدم.. ولهذه الوحدات أنواع عديدة، مثل:

- كارت الشاشة، وهو يحول الأرقام إلى نصوص وصور وفيديو ويرسلها إلى الشاشة لعرضها.
- كارت الصوت، وهو يحول الأرقام إلى أصوات ويرسلها إلى السماعة.
- الطابعة Printer، وهي تطبع النصوص والصور على الورق.
- كارت الفاكس Fax Modem، وهو يحول البيانات إلى إشارات كهربية ويرسلها عبر خطوط الهاتف إلى أي جهاز آخر.

وهكذا...

والشكل التالي يلخص العمليات الست التي يستطيع الحاسب القيام بها.



لغات البرمجة Programming Languages:

لكل مخلوق في الكون لغة خاصة به حتى لو لم نفهمها نحن.. ففي البشر والثدييات تكون لغة التخاطب صوتية.. وفي الحشرات لغة التخاطب تتم برقصات معينة أو إفرازات كيميائية، بالإضافة إلى الأصوات.. لكن مدلول كل صوت وحركة ورائحة يختلف من نوع إلى آخر، تماما كما تختلف معاني الكلمات من لغة إلى أخرى عند البشر.

ولا يختلف الأمر كثيرا إذا تكلمنا عن الحاسوب، لكن كما ذكرنا من قبل، فإن الحاسب لا يفهم إلا لغة الآلة، وهي لغة تتكون من نبضات كهربية تتحكم في كيفية عمل أجزائه المختلفة.. ونظرا لأن هذه النبضات لها قيمتان فقط (Off، On)، فقد تم تمثيلهما رقميا بالرقمين (٠، ١).. فإذا كان الرمز أ هو الشكل المكتوب للصوت "ألف"، فإن الرقم صفر هو الشكل المكتوب للجهد الكهربي ٠ فولت (Off) والرقم ١ هو الشكل المكتوب للجهد الكهربي ٥ فولت (On).. باختصار: الرقمان ٠ و ١ هما الأبجدية الخاصة التي نكتب بها لغة الآلة، وباستخدامها كتب المبرمجون الأوائل الأوامر الموجهة إلى الحاسب.

وكما رأينا في العرض التاريخي الموجز في المقدمة، فإن هذه اللغة بدائية وصعبة للغاية، لهذا فكر المبرمجون في الكتابة بلغة أسهل قليلا، على أن تتم ترجمة هذه اللغة بعد ذلك إلى لغة الآلة.. وقد سميت هذه اللغة بلغة التجميع Assembly، وكانت تستخدم كلمات إنجليزية مختصرة يسهل فهمها في كتابة الأوامر (مثل ADD و SUB و MOV).

ونظرا لأن ترجمة هذه الأوامر يدويا من لغة التجميع إلى لغة الآلة لن يحقق الهدف المنشود، فقد ابتكر أليك جليني Alick Glennie في عام ١٩٥٢م برنامجا أسماه مترجم الكود الآلي AutoCode-Compiler، لتحويل كود لغة التجميع إلى لغة الآلة.. وهكذا ظهر إلى الوجود أول مترجم Compiler.

المترجم Compiler:
هو برنامج يحول كل الأوامر المكتوبة بلغة البرمجة إلى لغة الآلة ليمكن للحاسب تنفيذها.. ويسمى البرنامج المكتوب بلغة البرمجة مشروعا Project، بينما يسمى البرنامج بعد تحويله إلى لغة الآلة ملفا تنفيذيا Executable File، ويكون له الامتداد Exe.، ويمكن للمستخدم تشغيله مباشرة على الويندوز بدون الحاجة إلى وجود لغة البرمجة التي كتب بها الكود الأصلي على الجهاز.

وقد ظهرت العديد من لغات البرمجة وتطورت مع الزمن، لتبتعد تدريجيا عن لغة الآلة، وتصير أكثر سهولة بالنسبة لمستخدمها.. وقد كانت لغات البرمجة الأولى تسمى باللغات منخفضة المستوى Low Level، لأنها كانت أقرب ما تكون إلى لغة الآلة، وكان المبرمج مضطرا إلى فهم تركيب مكونات الجهاز، لأن أوامر لغة البرمجة كانت تتعامل معها مباشرة.. هذا هو السبب الذي من أجله كانت البرمجة حكرا على المهندسين في البداية!.. لكن لغات البرمجة تطورت لتبتعد شيئا فشيئا عن تركيب الجهاز المادي، فصارت تسمى لغات برمجة عالية المستوى High Level، ولم يعد المبرمج بالضرورة مهندسا، بل صار بالإمكان أن يتعلم البرمجة أي صبي أو طفل في العاشرة من عمره لديه معرفة بسيطة باللغة الإنجليزية!

وتوجد العديد من لغات البرمجة، مثل كوبول COBOL وفورتران FORTRAN وسي بلاس بلاس C++ وجافا Java وبيزيك Basic.. وقد رأينا كيف تطورت إصدارات البيزيك على نظام الدوس DOS إلى أن ظهرت فيجيوال بيزيك على نظام الويندوز، وكيف ظهرت فيجيوال بيزيك على بيئة دوت نت منذ نهاية عام ٢٠٠١.

كما تتفرد لغة فيجيوال بيزيك عن سائر لغات البرمجة بأن لها مترجما ومفسرا.

المفسر Interpreter:
هو برنامج يحول كل أمر في البرنامج على حدة من لغة البرمجة عالية المستوى إلى لغة الآلة.. بمعنى أن المفسر يحول الأمر الأول إلى لغة الآلة ثم ينفذه مباشرة، وبعد ذلك يحول الأمر الثاني ثم ينفذه.. وهكذا حتى ينتهي البرنامج.

وقد كانت إصدارات فيجوال بيزيك السابقة لدوت نت (كفيجيوال بيزيك ٦ مثلا) تستخدم مفسرا Interpreter أثناء تصميم واختبار البرنامج، لهذا كان المبرمج يستطيع تشغيل البرنامج بدون أن ينشئ له ملفا تنفيذيا Exe. .. السبب في هذا أن المفسر ــ كما قلنا ــ

يترجم أمرا واحدا في كل مرة وينفذه مباشرة، لهذا لا يحتاج إلى ملف تنفيذي.. هذه الطريقة جعلت مبرمج فيجيوال بيزيك ٦ قادرا على تشغيل البرنامج بدون التأكد من خلوه تماما من الأخطاء، مما يعني سرعة بدء تشغيل البرنامج مهما كان حجمه كبيرا، فلا يوجد أي وقت ضائع في فحص كل الأخطاء وترجمة كل الأوامر.. لكن في مقابل سرعة بدء التشغيل، فإن المفسر كان يجعل تنفيذ البرنامج أبطأ، لأن عملية الترجمة تحدث لكل أمر أثناء التشغيل ومن ثم يتم تنفيذه.

وإذا صادف المفسر أثناء التشغيل خطأ في أحد الأوامر فإنه يتوقف عن تحويله ويعرض رسالة تفيد بوجود الخطأ.. وإذا أصلح المبرمج الخطأ فإن المفسر يستأنف تحويل وتنفيذ الأوامر من حيث توقف.. هذا يعني أن من أهم مميزات هذه الطريقة سهولة اكتشاف الخطأ وإصلاحه أما أبرز عيوبها فهو بطء التشغيل.

أما المترجم Compiler، فإنه يقوم بتحويل البرنامج بالكامل إلى لغة الآلة وإنشاء ملف تنفيذي قبل أن يبدأ في تنفيذ أي أمر.. لهذا السبب يكون تشغيل المبرمج للبرنامج لأول مرة أبطأ لأن المترجم يترجم البرنامج كاملا قبل تشغيله، لكن تنفيذ البرنامج بعد ذلك يكون أسرع، لأن كل الأوامر تمت ترجمتها من قبل.. أيضا لا تتم عملية الترجمة إلا بعد تصحيح جميع الأخطاء الموجودة في الكود.. لهذا تكون عملية تصحيح الأخطاء أصعب.

والجدول التالي يعرض مقارنة بين المفسر والمترجم:

المقارنة	المترجم	المفسر
طريقة الترجمة	تتم ترجمة البرنامج كاملا.	تتم ترجمة كل أمر في البرنامج على حدة.
طريقة التنفيذ	يوضع كود لغة الآلة الناتج من الترجمة في ملف تنفيذي له الامتداد exe. ثم يتم تنفيذه.	يتم تنفيذ كل أمر بعد ترجمته مباشرة، لهذا لا يوجد داع لوجود ملف تنفيذي.
بدء تشغيل البرنامج	أبطأ.	أسرع.
تنفيذ البرنامج	أسرع.	أبطأ.
فحص الأخطاء	يتم فحص كل الأخطاء قبل الترجمة، وتفشل الترجمة في حالة وجود أي خطأ، لهذا يجب على المبرمج تصحيح الأخطاء أولا ثم إعادة عملية الترجمة.	لا حاجة لفحص كل أخطاء البرنامج، فكل أمر يتم فحصه قبل ترجمته وتنفيذه، ولو تم اكتشاف أي خطأ يتوقف المفسر عند السطر الذي به الخطأ ويتيح للمبرمج تصحيحه ثم مواصلة تنفيذ البرنامج.
تصحيح الأخطاء	أصعب.	أسهل.

وتستخدم فيجيوال بيزيك ٦ المترجم والمفسر معا، فالمبرمج يستخدم المفسر في مرحلة كتابة الكود واختباره وتصحيح أخطائه، وبذلك يضمن تشغل البرنامج بسرعة لاختباره،

مع سهولة تصحيح الأخطاء الموجودة به.. وبعد أن ينتهي المبرمج نهائيا من كتابة البرنامج ويضمن خلوه من الأخطاء، يطلب من المترجم الخاص بفيجيوال بيزيك ٦ ترجمة البرنامج كاملا، وذلك للحصول على الملف التنفيذ للبرنامج exe. ، حتى يمكن توزيع البرنامج إلى المستخدمين، حيث يضمن المبرمج في هذه الحالة أن البرنامج سيعمل بشكل أسرع عند المستخدم، لأنه ملف مترجم Compiled وخال من الأخطاء.. وبهذا يستفيد المبرمج من مميزات المفسر والمترجم مع تلافى عيوب كل منهما.

لكن.. ماذا عن فيجيوال بيزيك دوت نت؟

في الحقيقة، يختلف الأمر في بيئة دوت نت عن فيجيوال بيزيك ٦:

١- فعند تشغيل البرنامج في فيجيوال بيزيك دوت نت يتأكد المترجم من عدم وجود أية أخطاء في البرنامج، ومن ثم يقوم بترجمة الكود كاملا وإنشاء ملف تنفيذي وتشغيله.. إذن ففيجيوال بيزيك دوت نت تستخدم المترجم بشكل أساسي.

٢- لكن فيجيوال بيزيك دوت نت تمنح المبرمج أيضا بعض إمكانيات المفسر، فبعد أن يكتب المبرمج سطرا من الكود وينتقل إلى سطر جديد، ودون أن يقوم المبرمج بتشغيل البرنامج، تقوم فيجيوال بيزيك دوت نت بفحص أخطاء هذا السطر وتنبيه المبرمج إليها.. وإذا حدث خطأ في البرنامج أثناء تشغيله يتم إيقاف البرنامج عند السطر الذي سبب الخطأ، حيث يمكن للمبرمج تصحيح هذا الخطأ ومواصلة تنفيذ البرنامج.

٣- لكن الملحوظة الهامة التي يجب أن تنتبه إليها، هي أن مترجم فيجيوال بيزيك دوت نت لا يترجم الكود إلى لغة الآلة، بل إلى لغة تسمى "اللغة الوسيطة" Intermediate Language أو اختصارا IL.. وتعرف أيضا باسم لغة ميكروسوفت الوسيطة MSIL.. وعند تشغيل الملف التنفيذي المكتوب بهذه اللغة، يقوم جزء من بيئة دوت نت يسمى إطار العمل Framework، بتحويل اللغة الوسطية إلى لغة الآلة.. لهذا السبب يجب أن يكون الإصدار الثاني من إطار عمل دوت نت NET Framework 2.0. موجودا على الجهاز الذي تريد تشغيل برنامجك عليه.. الحكمة من وراء هذا هي جعل برنامجك قادرا على العمل على أي نظام تشغيل يوجد عليه إطار العمل، بدلا من أن يقتصر عمل برنامجك على نظام الويندوز فحسب، فإطار العمل سيترجم برنامجك إلى لغة الآلة المناسبة التي يفهمها نظام التشغيل.

-٢-

المخططات (خرائط التدفق) Flowcharts

التفكير البرمجي:

لا يستطيع الحاسوب أن يفكر.. لقد سمي بالحاسوب لأنه يُجري العمليات الحسابية، ولو كان يستطيع التفكير لأسموه العقل الآلي بدلا من الحاسب الآلي!

ويمتاز البشر عن الحواسيب بامتلاكهم الخيال الذي يولد القدرة على الابتكار.. كما يمتلك البشر الإرادة التي تدفعهم إلى التفكير في حلول المشاكل التي تواجههم، وتسهل لهم الوصول إلى أهدافهم.. وعليك أن تحمد الله أن الحواسيب ليست كذلك، وإلا لما كانت هناك ضرورة لوجود مبرمجين يفكرون بدلا من الحواسيب!

وفي هذا الفصل سنخطو أولى خطواتنا العملية في عالم البرمجة، حيث سنتعلم كيف نحل المشاكل بالطريقة التي يفهمها الحاسوب، بدون أن نستخدم أية لغة برمجة.

لقد عرفنا في الفصل السابق العمليات الأساسية الست التي يستطيع الحاسب القيام بها.. ولكي نكون مبرمجين ناجحين، فعلينا أن نجد طريقة مناسبة لحل أية مشكلة تواجهنا باستخدام هذه العمليات الست.. بمعنى آخر: على المبرمج أن يتقمص شخصية الحاسب ويعامله على قدر (ذكائه).. وكما لا تستطيع أن تطلب من طفل أن يتعلم النظريات العلمية المعقدة، عليك أيضا ألا تطلب من الحاسب شيئا يفوق طاقته، كأن تتوقع منه ابتكار حلول ذكية من عنده مثلا!

خطوات حل المسائل البرمجية:

الخطوات الأساسية لحل أية مسألة أو مشكلة برمجية، لا تختلف في شيء عن خطوات حل أية مسألة واجهتك في الحساب.. وهذه الخطوات هي:

١- تعريف المسألة:
قم بتحديد المطلوب بالضبط، وذلك بقراءة المسألة بدقة وفهم.. لاحظ أن فهم المطلوب بشكل خاطئ سيجعلك تحل المسألة بشكل خاطئ.

٢- إهمال البيانات الزائدة، وغير الداخلة في الحل.

٣- تعريف المتغيرات:
بإعطاء أسماء واضحة للبيانات التي ستستخدم في حل المسألة.

٤- إيجاد العلاقات بين المتغيرات ووضعها في صورة معادلات:
بهذه الطريقة تستطيع الحصول على المعلومات المطلوبة من البيانات المتاحة.

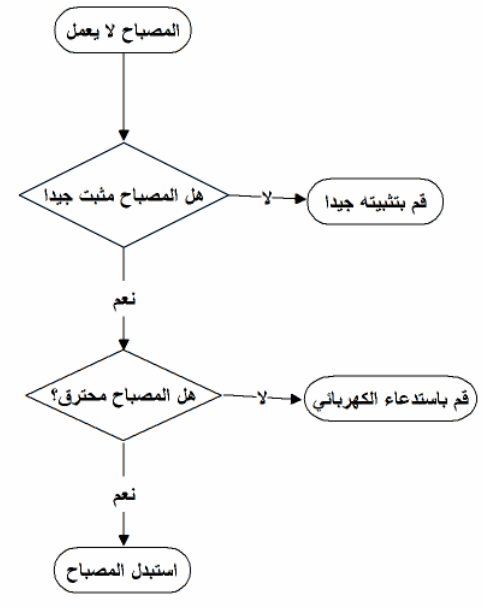
٥- كتابة خوارزمية Algorithm توضح خطوات الحل:
في هذه الخوارزمية سنستخدم البيانات والمعادلات والمتغيرات لحساب المطلوب.

فلنأخذ مثالا:
افترض أنك صحوت من النوم في السادسة صباحا، وضغطت زر الإضاءة فلم يعمل المصباح.. ما الذي يمكن أن تفعله لحل هذه المشكلة؟
في البداية يجب أن نحلل معا معطيات المشكلة، ونستبعد أية بيانات ليس لها قيمة في الحل.
فالبيان الأساسي هنا هو:
- المصباح لا يعمل رغم ضغط زر الإضاءة.

أما البيانان التاليان فلا قيمة لها:
- أنك كنت نائما وصحوت.
- أن المشكلة حدثت في السادسة صباحا.

نريد الآن أن نكتب الخطوات التي ستتبعها لحل هذه المشكلة.. هذه الخطوات ستكون كالتالي:

إذا كان المصباح غير مثبت جيدا إذن:
قم بتثبيته جيدا.
غير ذلك إذا كان المصباح محترقا إذن:
قم بتركيب مصباح جديد.
غير ذلك:
استدع الكهربائي لمعرفة سبب العطل.
نهاية الشرط.



ما كتبناه أعلاه يسمى خوارزمية Algorithm، وهي ــ كما هو واضح ــ شرح منظم للخطوات التي يجب اتباعها لحل المشكلة.
الآن يمكنك أن تعطي هذه الخوارزمية إلى أخيك الصغير لتعلمه كيف يتصرف إذا تعطل المصباح.. إذن فالخوارزمية هي طريقة منظمة ومختصرة لنقل الأفكار.. لهذا السبب اتفق المبرمجون فيما بينهم على وضع قواعد ثابتة لكتابة الخوارزميات، بدلا من أن يكتبها كل منهم بطريقته الخاصة فيصعب فهمها على الآخرين.. وسنتعلم في هذا الكتاب طريقتين:
- الطريقة الأولى تستخدم الرسوم الإيضاحية لكتابة الخوارزمية.. هذه

الطريقة تسمى "مخطط التنفيذ" Flowchart، أو تبعا للترجمة الحرفية "خريطة التدفق".

- الطريقة الثانية تستخدم جملا إنجليزية بسيطة تشبه الكود، لكنها لا تنتمي إلى أية لغة برمجة محددة.. لهذا تسمى هذه الطريقة بالكود الوهمي أو الكود الزائف Pseudo Code.. وسنتعرف عليها في الفصل التالي.. والصورة السابقة توضح مخطط خوارزمية إصلاح المصباح.

مخطط التنفيذ Flowchart:

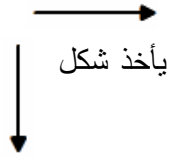
في المقاطع القادمة سنتعرف على مخطط التنفيذ وسنحل أمثلة كثيرة تساعدنا على فهمه.. وقد رأيت للتسهيل أن أكتب المخططات في هذا الفصل باللغة العربية، لتسهيل فهمه عليك.. وفي الفصل القادم بإذن الله سنعود إلى كتابة المخططات باللغة الإنجليزية.. لاحظ أن المخطط ليس لغة برمجة، وبالتالي لا يهم إن كتبناه بأية لغة بشرية، فهو مجرد طريقة لتمثيل الأفكار البرمجية.. لكن نظرا لأن معظم الكتب والشروح مكتوبة بالإنجليزية، فمن المهم التدرب على قراءة وكتابة المخطط بالإنجليزية، وبهذا يسهل عليك التواصل مع باقي مبرمجي العالم.. لكن دعنا نبدأ تعلمه أولا بالعربية لنسهل الأمور على أنفسنا.

المخطط Flowchart:
هو تمثيل بياني يعتمد على الرسم لتوضيح ترتيب العمليات اللازمة لحل المسألة. بطريقة أخرى: المخطط هو خوارزمية مكتوبة بالصور لا بالكلمات.

الرموز الأساسية للمخططات:

هناك خمسة أشكال أساسية ترسم بها أغلب المخططات، وهي:

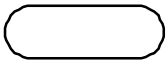
١- خط الاتجاه Flow Line:



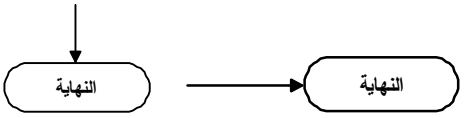
يستخدم هذا الرمز لرسم خطوط تصل بين باقي رموز المخطط، وهو يأخذ شكل خط في نهايته سهم يشير إلى اتجاه الحركة عبر المخطط.

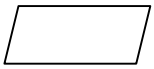
٢- الرمز الطرفي Terminal:

يستخدم هذا الرمز لتوضيح موضع بداية ونهاية المخطط.. هذا هو السبب في تسميته بالرمز الطرفي، لأنه يوجد في الأطراف.. لاحظ أن البرنامج يبدأ من نقطة واحدة فقط، ويكون لها أحد الشكلين التاليين:



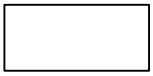
لكن من الممكن أن توجد أكثر من نهاية للبرنامج، وذلك إذا كان يحتوي على أكثر من مسار، ففي هذه الحالة ستوجد نهاية واحدة في كل مسار، كما سنرى فيما بعد.. ويكون لرمز النهاية أحد الشكلين التاليين:





٣- إدخال أو إخراج Input/Output:

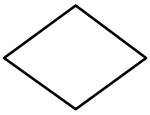
يستخدم هذا الرمز لقراءة بيانات من المستخدم، أو إخراج معلومات للمستخدم.



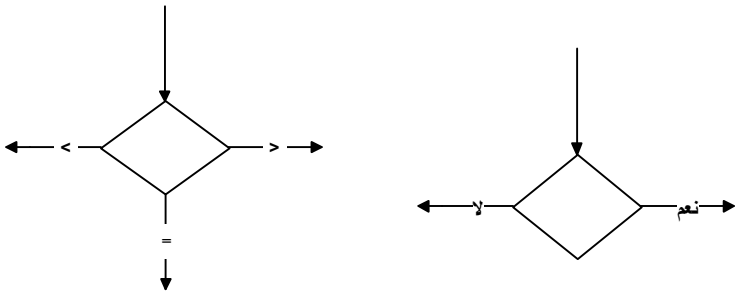
٤- معالجة / عملية Process:

يستخدم هذا الرمز للتعبير عن الأوامر التي ينفذها البرنامج، مثل العمليات الحسابية وغيرها.

٥- اتخاذ قرار Decision:



يستخدم هذا الرمز لإجراء عمليات المقارنة.. ويختلف هذا الرمز عن الرموز السابقة في خروج سهمين منه وأحيانا ثلاثة، حيث يمثل كل من هذه الأسهم أحد الاحتمالات الناتجة عن المقارنة، كما هو واضح في الصورة:



والآن، دعنا نأخذ بعض الأمثلة على مخططات التنفيذ، لنفهمها بشكل عملي:

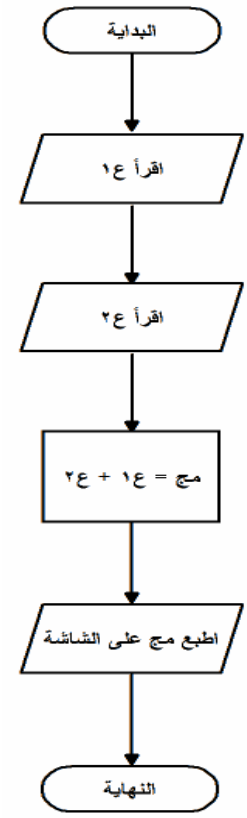
مثال ١:
ارسم مخططا لجمع أي عددين يكتبهما المستخدم، وعرض الناتج على الشاشة.

الحل:
في هذه المسألة نحتاج إلى استخدام ثلاثة متغيرات Variables:

١. الأول اسمه ع١، وسنضع فيه العدد الأول الذي يكتبه المستخدم.
٢. والثاني اسمه ع٢، وسنضع فيه العدد الثاني الذي يكتبه المستخدم.
٣. والثالث اسمه مج، وسنضع فيه مجموع العددين.

ملحوظة:
تسمح فيجيوال بيزيك بتعريف متغيرات بحروف إنجليزية أو عربية.. إذن فليست هناك مشكلة من استخدام متغيرات أسماؤها ع١ و ع٢ و مج.. وعلى كل حال، مخطط التنفيذ هو طريقة مجردة لتبسيط الحل، والمبرمج ليس مجبرا على استخدام نفس أسماء المتغيرات المستخدمة هنا.

والشكل المقابل يريك مخطط الحل.

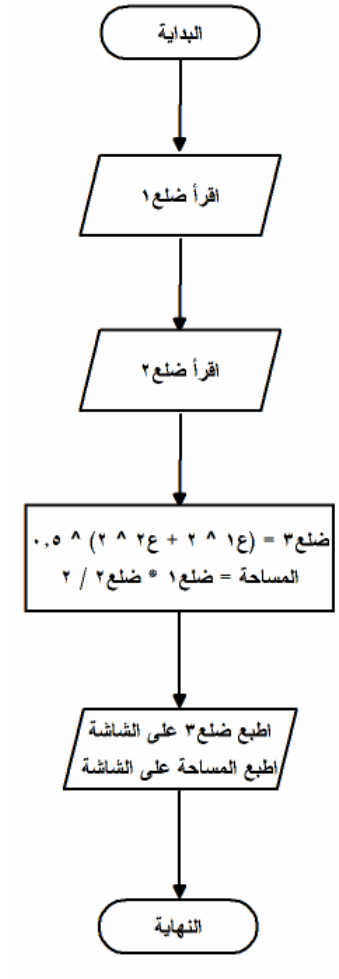


مثال ٢:
ارسم مخطط برنامج يسمح للمستخدم بإدخال ضلعي الزاوية القائمة في مثلث قائم الزاوية، ومن ثم يقوم البرنامج بحساب طول الوتر ومساحة سطح المثلث، ويعرضهما على الشاشة.

الحل:
في هذه المسألة سنستخدم قاعدة رياضية معروفة، وهي قاعدة فيثاغورث، والتي تقول إن مجموع مربعي ضلعي الزاوية القائمة يساوي مربع الوتر.. أي أن:
طول الوتر = الجذر التربيعي لمجموع مربعي ضلعي القائمة.

لاحظ أننا في البرمجة نستخدم الرمز ^ للتعبير عن الأس.. فمثلا: ٢ ^ ٣ تعني ٢ أس ٣ .. وأنت تعرف أن الجذر هو مقلوب الأس، لهذا يمكننا كتابة الجذر التربيعي للعدد ٤ كالتالي: (٤ أس ١/٢) أو (٤ أس ٠,٥)، ويمكن أن نكتب هذا برمجيا كالتالي: (٤ ^ ٠,٥).

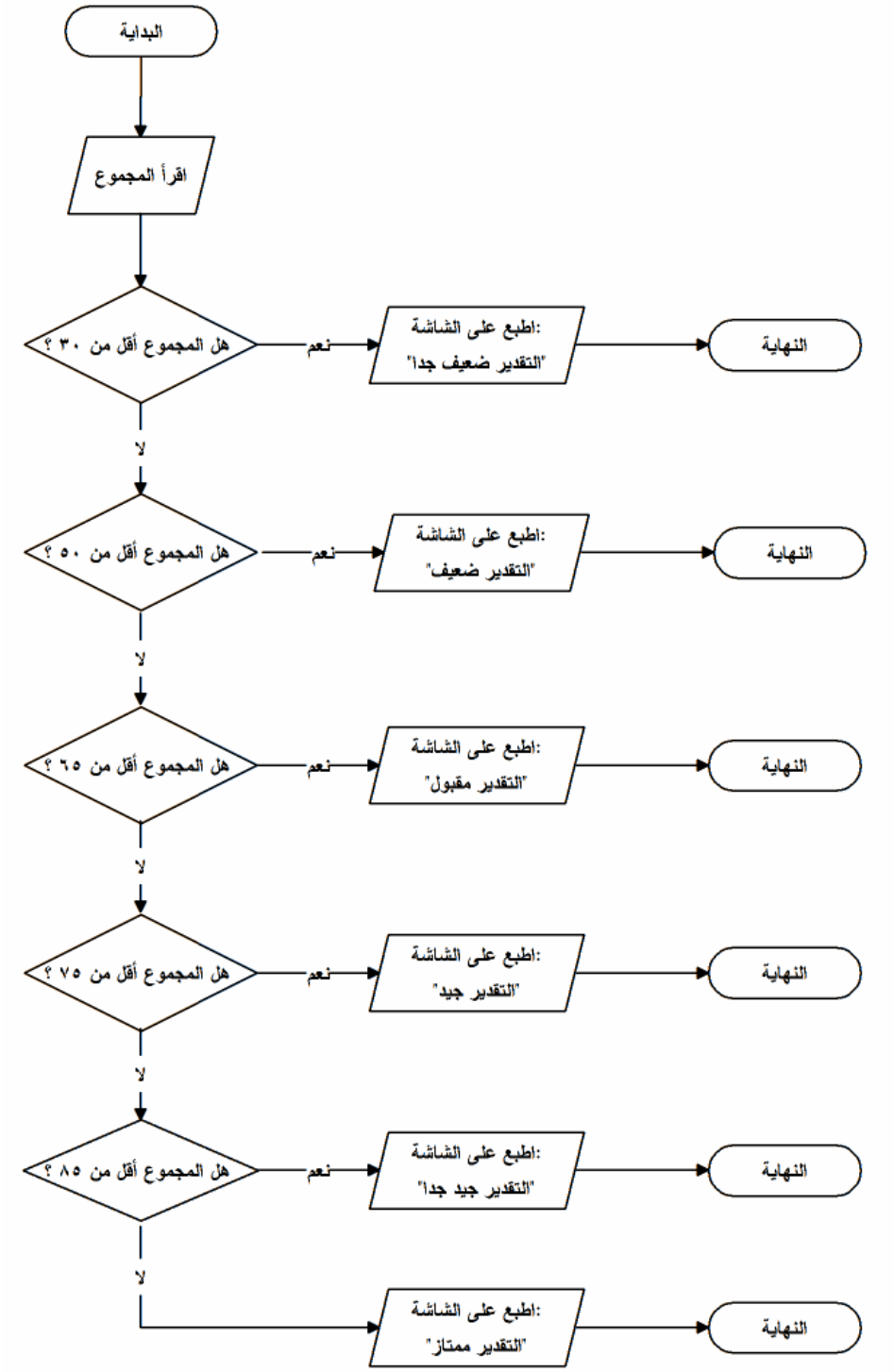
رموز العمليات الحسابية المستخدمة في البرمجة	
الجمع	+
الطرح	−
الضرب	*
القسمة	/
الأس	^



وسنحتاج في برنامجنا هذا إلى أربعة متغيرات:
ضلع١ و ضلع٢: ضلعا الزاوية القائمة اللذان سيمدنا بهما المستخدم.
ضلع٣: وتر المثلث قائم الزاوية الذي سنحسب طوله.
المساحة: مساحة سطح المثلث قائم الزاوية.. تعرف طبعا أن مساحة المثلث = نصف القاعدة × الارتفاع.. وبما أنه مثلث قائم الزاوية، فهذا يعني أن ضلعي القائمة هما القاعدة والارتفاع، لهذا ستكون مساحة سطحه = نصف حاصل ضرب ضلعي القائمة.. أي أن:
المساحة = ١/٢ * ضلع١ * ضلع٢.

مثال ٣:
ارسم مخططا لبرنامج يسمح للمستخدم بإدخال مجموع درجات الطالب، ويعرض له التقدير الذي يشير إليه هذا المجموع.. اعتبر أن المجموع النهائي هو ١٠٠ درجة، وأن المستخدم يدخل عددا لا يزيد عن ١٠٠.

الحل:
مخطط هذا المثال مرسوم في الصفحة المقابلة.. دعنا نفهم معا كيف رسمنا هذا المخطط.
نحتاج في هذا البرنامج إلى استخدام جمل الشرط، لمقارنة مجموع الطالب بالتقديرات المختلفة.. سنستخدم هنا متغيرا اسمه المجموع يمثل درجات الطالب، وستكون الخوارزمية كالتالي:

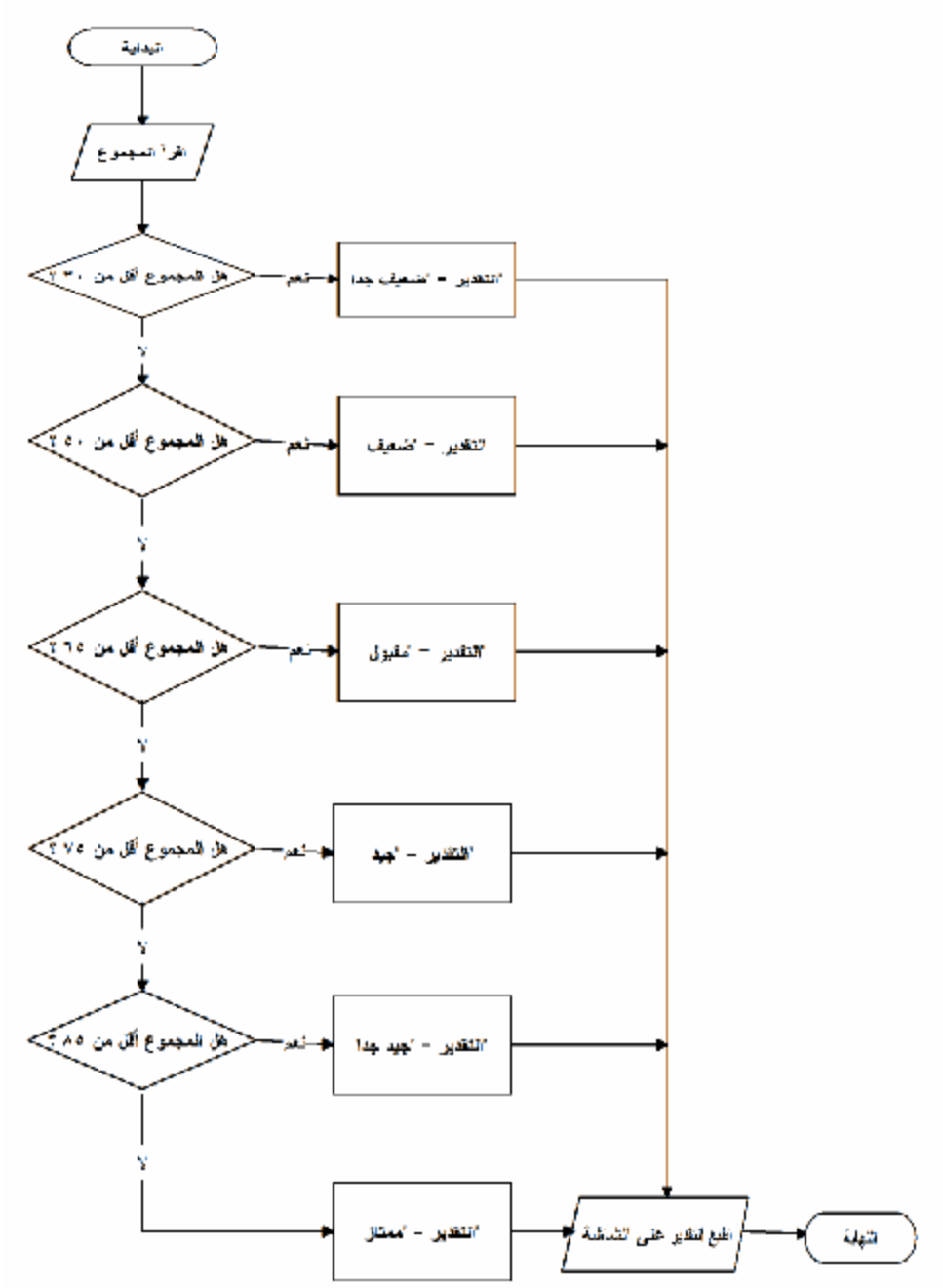

البداية
اقرأ المجموع
هل المجموع أقل من ٣٠ ؟
نعم
اطبع على الشاشة:
"التقدير ضعيف جدا"
النهاية
لا
هل المجموع أقل من ٥٠ ؟
نعم
اطبع على الشاشة:
"التقدير ضعيف"
النهاية
لا
هل المجموع أقل من ٦٥ ؟
نعم
اطبع على الشاشة:
"التقدير مقبول"
النهاية
لا
هل المجموع أقل من ٧٥ ؟
نعم
اطبع على الشاشة:
"التقدير جيد"
النهاية
لا
هل المجموع أقل من ٨٥ ؟
نعم
اطبع على الشاشة:
"التقدير جيد جدا"
النهاية
لا
اطبع على الشاشة:
"التقدير ممتاز"
النهاية

إذا كان المجموع أصغر من ٣٠ إذن:
اطبع على الشاشة "التقدير ضعيف جدا".
غير ذلك إذا كان المجموع أصغر من ٥٠ إذن:
اطبع على الشاشة "التقدير ضعيف".
غير ذلك إذا كان المجموع أصغر من ٦٥ إذن:
اطبع على الشاشة "التقدير مقبول".
غير ذلك إذا كان المجموع أصغر من ٧٥ إذن:
اطبع على الشاشة "التقدير جيد".
غير ذلك إذا كان المجموع أصغر من ٨٥ إذن:
اطبع على الشاشة "التقدير جيد جدا".
غير ذلك:
اطبع على الشاشة "التقدير ممتاز".
نهاية الشرط.

حل آخر:

نستطيع تحسين هذا المخطط قليلا، بتحسين الخوارزمية التي نستخدمها.. فلو كانت طريقة الطباعة على الشاشة تحتاج إلى كود كثير، فهذا يعني أننا سنكرر كتابة هذا الكود في كل جملة من جمل الشرط.. يمكن تلافي هذا باستخدام متغير نصي String Variable وليكن اسمه **التقدير**، حيث سنضع فيه التقدير الذي يناسب المجموع في كل جملة شرط، وبعد انتهاء جمل الشرط نطبع النص الموجود في هذا المتغير.. هكذا ستكون الخوارزمية المعدلة:

إذا كان المجموع أصغر من ٣٠ إذن:
التقدير = "ضعيف جدا".
غير ذلك إذا كان المجموع أصغر من ٥٠ إذن:
التقدير = "ضعيف".
غير ذلك إذا كان المجموع أصغر من ٦٥ إذن:
التقدير = "مقبول".
غير ذلك إذا كان المجموع أصغر من ٧٥ إذن:
التقدير = "جيد".
غير ذلك إذا كان المجموع أصغر من ٨٥ إذن:
التقدير = "جيد جدا".
غير ذلك:
التقدير = "ممتاز".
نهاية الشرط.
اطبع التقدير على الشاشة.



لاحظ أن النص الذي نضعه في المتغير النصي String Variable يوضع بين علامتـــي تنصيص ""، للتفريق بينه وبين أسماء المتغيرات.. بهذا نعرف أن "**ممتاز**" هو نص عادي،

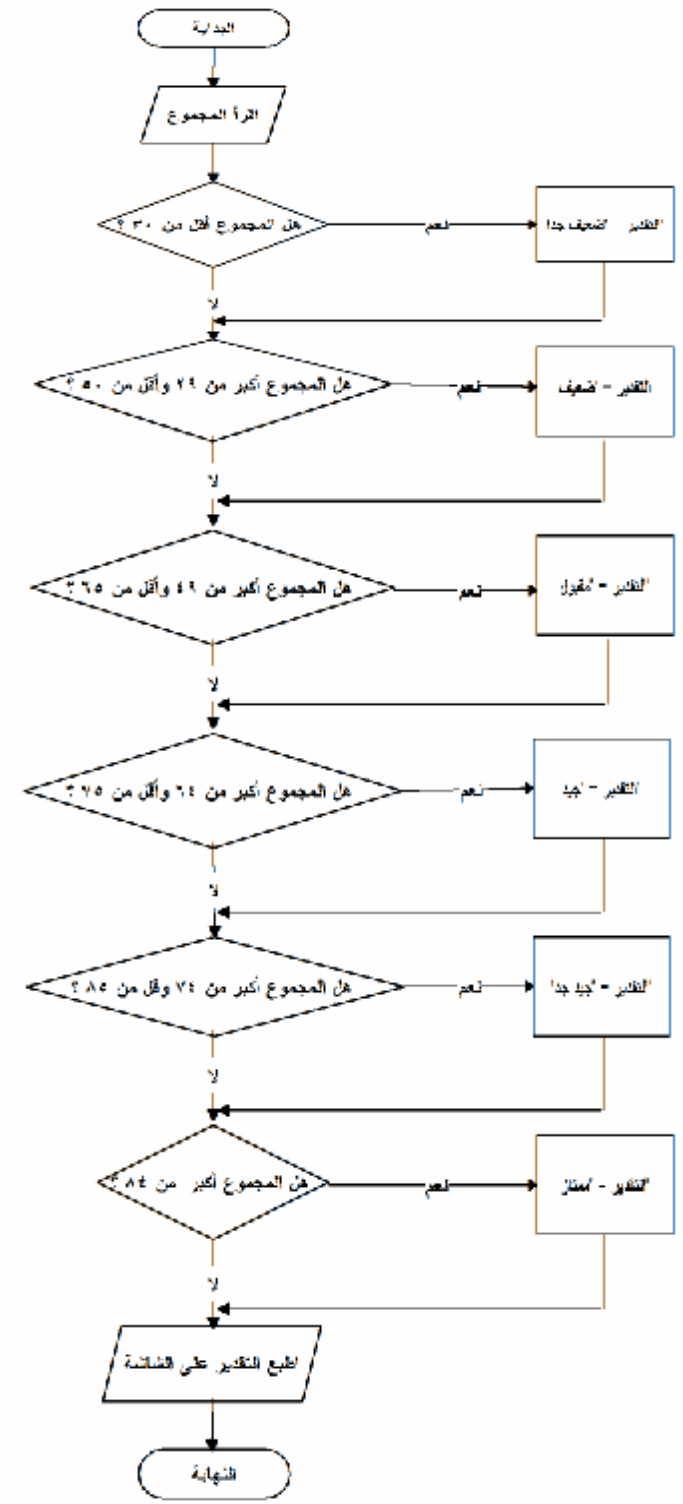
بينما **التقدير** هو اسم متغير.. لاحظ أنك تستطيع جعل المتغير فارغا باستخدام علامتي تنصيص فارغتين ""‌.. يسمى هذا بالنص الفارغ Empty String.. مثل: **التقدير** = ""
تلاحظ في المخطط أيضا أن مضلع المخرجات مرسوم مرة واحدة، وأن كل الشروط تشير إليه.. يمكنك استخدام هذه الطريقة في الحالات التي تريد فيها تلافي تكرار أي جزء من المخطط أكثر من مرة بلا فائدة.
ما زالت لدينا ملاحظة مهمة حول هذا البرنامج.. ففي الخوارزمية التي كتبناها تلاحظ أن الشروط مرتبة.. ولو حاولت تغيير موضع أي شرطين فستحصل على نتائج خاطئة.. انظر إلى هذا مثلا:

إذا كان المجموع أصغر من ٧٥ إذن:
التقدير = "جيد".
غير ذلك إذا كان المجموع أصغر من ٣٠ إذن:
التقدير = "ضعيف جدا".
نهاية الشرط.

تعال نرى ما سيحدث لو كان المجموع ٢٠ درجة مثلا.. عند اختبار أول شرط سيجده البرنامج صحيحا (لأن المجموع ٢٠ أصغر من ٧٠ فعلا) وبالتالي سيخبر برنامجك المستخدم أن الطالب حصل على تقدير جيد، بينما الحقيقة أن الطالب راسب وتقديره ضعيف جدا!!
حدث هذا لأن الشرط الثاني لن يتم تنفيذه أبدا.. السبب في هذا هو أنه تم تنفيذ أي شرط لا يتم فحص الشروط التالية له.. أو بتعبير آخر: لا يتم فحص أي شرط إلا إذا كان الشرط السابق له خاطئا.. هذه هي وظيفة الجملة **غير ذلك**، (والتي يقابلها Else في فيجوال بيزيك كما سنرى فيما بعد).

حل ثالث (حل غير عملي.. نعرضه للمقارنة فقط):
إذا أردت أن تكتب جمل الشرط بدون استخدام طريقة "غير ذلك"، فعليك أن تستخدم شروطا مستقلة عن بعضها.. في هذه الحالة يجب أن تكتب شرط كل تقدير كاملا.. فالطالب يحصل على جيد جدا مثلا إذا كان مجموعه أكبر من ٦٤ درجة وأقل من ٧٥ درجة.. إذن يجب أن تكون الخوارزمية الجديدة كالتالي:

إذا كان المجموع أصغر من ٣٠ إذن: التقدير = "ضعيف جدا".
إذا كان المجموع أكبر من ٢٩ وأصغر من ٥٠ إذن: التقدير = "ضعيف".
إذا كان المجموع أكبر من ٤٩ وأصغر من ٦٥ إذن: التقدير = "مقبول".
إذا كان المجموع أكبر من ٦٤ وأصغر من ٧٥ إذن: التقدير = "جيد".
إذا كان المجموع أكبر من ٧٤ وأصغر من ٨٥ إذن: التقدير = "جيد جدا".
إذا كان المجموع أكبر من ٨٤ إذن: التقدير = "ممتاز".
اطبع التقدير على الشاشة.


البداية
اقرأ المجموع
هل المجموع أقل من ٣٠ ؟
نعم
التقدير ضعيف جدا
لا
هل المجموع أكبر من ٢٩ وأقل من ٤٠ ؟
نعم
التقدير = ضعيف
لا
هل المجموع أكبر من ٤٩ وأقل من ٦٥ ؟
نعم
التقدير = مقبول
لا
هل المجموع أكبر من ٦٤ وأقل من ٧٥ ؟
نعم
التقدير = جيد
لا
هل المجموع أكبر من ٧٤ وقل من ٨٥ ؟
نعم
التقدير = جيد جدا
لا
هل المجموع أكبر من ٨٤ ؟
نعم
التقدير = ممتاز
لا
اطبع التقدير على الشاشة
النهاية

كما تلاحظ، يختلف مخطط هذا الحل عن مخطط الحل الأول والثاني في نقطة جوهرية، فسواء كانت إجابة كل شرط نعم أم لا، فيجب أن يتجه كل من السهمين إلى الشرط التالي.. ولو نظرت إلى المخطط جيدا، فسترى أن السهم "لا" الخارج من كل شرط غير مستغل، فهو يشير فقط إلى الشرط التالي بدون تنفيذ أي كود خاص.. هذا هو السبب الرئيسي في أن هذه الطريقة في تصميم جمل الشرط تهدر وقت وطاقة البرنامج، حيث يبدو الشرط كالأعرج الذي يسير على ساق واحدة.

هذا يقول بوضوح إن كتابة جمل الشر منفصلة عن بعضها يحتوي على العيوب التالية:

١- متعبة في كتابتها ورسمها لأنها أطول.

٢- معظم الشروط صارت تحتوي على عمليتي مقارنة مع عددين بدلا من عملية مقارنة مع عدد واحد، مما يعني أن تنفيذ البرنامج سيكون أبطأ.

٣- كل الشروط سيتم فحصها، رغم أن واحدا منها فقط هو الشرط الصحيح.. فمثلا: لو حصل الطالب على ٢٠ درجة، فإن أول شرط سيتحقق وسيكون تقديره ضعيف جدا.. لكن رغم هذا سيستمر البرنامج في فحص باقي الشروط بلا طائل، مما يضيع المزيد من الوقت.

هذا يوضح لك فوائد تصميم الشروط باستخدام المقطع "غير ذلك" Else.. ولا يجب أن يستخدم المبرمج جمل الشرط المستقلة إلا إذا كانت الشروط فعلا غير مترابطة وليس بينها أية علاقة، كما سنرى في المثال التالي.

مثال ٤:

ارسم مخططا لبرنامج يسأل الطالب ثلاثة أسئلة، ويعطيه درجة على كل إجابة صحيحة، ويعرض مجموع الدرجات في النهاية.

الحل:

سنستخدم في هذا البرنامج متغيرا اسمه **المجموع**، حيث سنجمع عليه درجة في كل مرة يجيب فيها المستخدم بإجابة صحيحة.. لاحظ أن البرمجة تستخدم الصيغة التالية لجمع ١ على المتغير:

المجموع = المجموع + ١

هذه الجملة ستبدو لك مربكة للوهلة الأولى، لأنك ستصرخ في استهجان: كيف يمكن أن يساوي الشيء نفسه بعد أن نجمع عليه ١ ؟!!

طيب.. دعني أكتب لك هذه العملية بشكل آخر:

س = المجموع + ١

المجموع = س

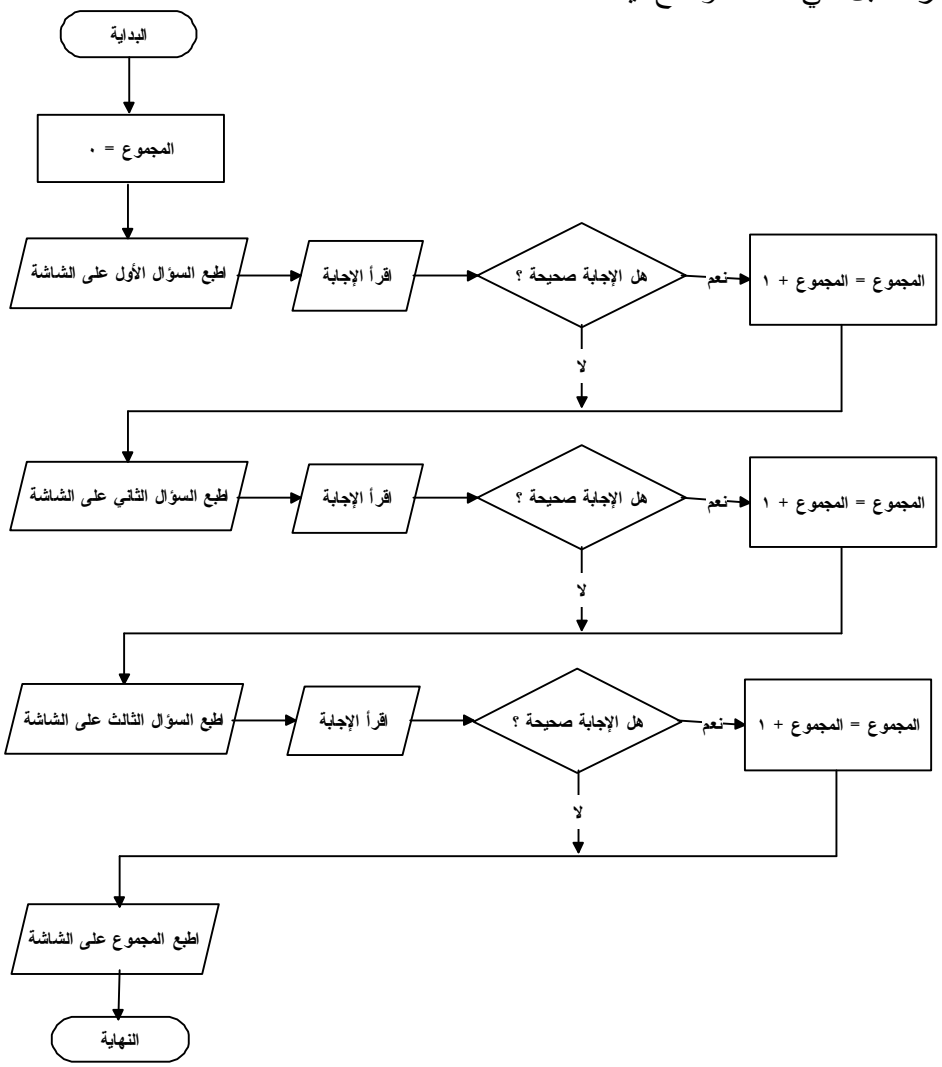
ما فعلناه هو تنفيذ العملية على خطوتين:

١- في الخطوة الأولى استخدمنا متغيرا مؤقتا اسمه س ووضعنا فيه المجموع + ١.

٢- في الخطوة الثانية نقلنا القيمة الموجودة في المتغير س إلى المجموع.

من هاتين الخطوتين سنلاحظ حقيقة هامة، وهي أن المكتوب هنا ليس معادلات رياضية، بل هو نقل للقيم من متغير إلى آخر.. هذه العملية تسمى وضع القيمة في المتغير

Assignment، وتستخدم فيها العلامة = .. إذن فالعلامة = المستخدمة هنا ليست علامة مقارنة، بل هي علامة وضع قيمة.



وقديما كان المبرمج مضطرا إلى تغيير قيمة نفس المتغير في خطوتين، كما فعلنا في الكود:

س = المجموع + ١

المجموع = س

ولكن نظرا لأن هذه العملية تتكرر كثيرا، فقد اختصرتها لغات البرمجة في خطوة واحدة بحذف المتغير الوسيط س، ليصير الكود:

المجموع = المجموع + ١

هذا مجرد تسهيل على المبرمج، يوفر له وقتا وجهدا عند كتابة الكود.. ولكي لا يكون هذا الكود مربكا لك، فعليك أن تقرأه كالتالي:

قيمة المجموع بعد تنفيذ الكود = قيمة المجموع قبل تنفيذ الكود + ١

ويمكنك تعميم هذه الطريقة على عمليات الطرح والضرب والقسمة، مع إمكانية استخدام العدد ١ أو ٢ أو أي عدد، أو حتى أي متغير آخر في هذه العمليات، كما في الأمثلة التالية:

المجموع = المجموع / ٢
المجموع = المجموع – س
المجموع = س + المجموع /٢

وهكذا...

وكما ترى في المخطط في الصفحة السابقة: كل شرط مستقل عن الآخر، وأيا كانت نتيجة الشرط، وسواء كانت الإجابة صحيحة أم لا، فلا بد من التوجه إلى الشرط التالي لطرح السؤال التالي وتقييمه.. باختصار: إجابة الطالب عن كل سؤال لا علاقة لها بالسؤال التالي له، فكل ما يعنينا هو المجموع النهائي.

تدريب:

بعد عرض المجموع على الشاشة في المخطط السابق استخدم جمل الشرط لتقييم إجابات الطالب.. اعتبر أن ممتاز = ٣ درجات، وجيد = ٢، ومقبول = ١، وضعيف جدا = ٠.

إرشاد:

واضح أنك ستستخدم نفس الطريقة التي اتبعناها في المثال رقم ٣ لمعرفة تقدير الطالب، وعليك استخدام "غير ذلك" Else في هذا الجزء، لأنها الطريقة الأمثل لحل الشروط المترابطة.

مثال ٥:

ارسم مخططا لبرنامج يقرأ ٤ أعداد ويعرض على الشاشة أكبر عدد فيها Maximum Number.. ثم وضح على الرسم مسار التنفيذ إذا كانت الأعداد التي أدخلها المستخدم هي ١٤ و ٥ و ٢٠ و ٩.

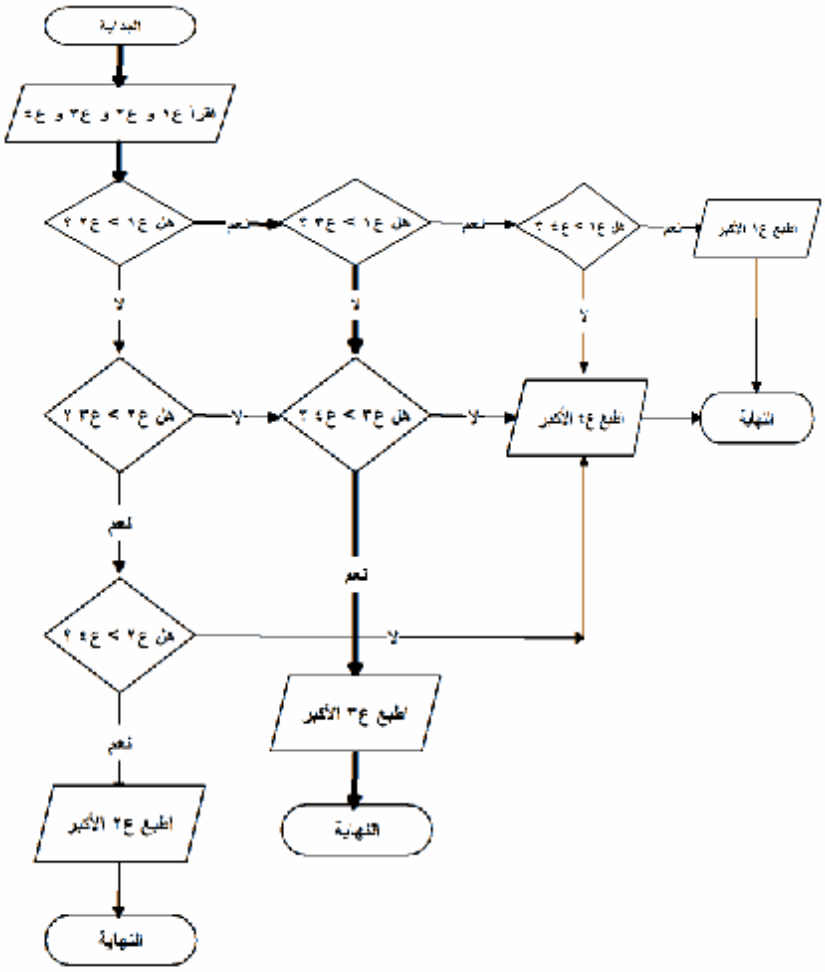
الحل:

كل ما سيفعله هذا البرنامج هو مقارنة الأعداد بعضها ببعض.. لاحظ أن معرفة أكبر عدد من بين أربعة أعداد يحتاج بالضبط إلى إجراء ثلاث عمليات مقارنة، لأنا لا نحتاج إلى مقارنة العدد بنفسه.

دعنا نرمز للأعداد بالمتغيرات ع١، ع٢، ع٣، ع٤.. هذه هي الخوارزمية:

إذا كان ع١ > ع٢ إذن:
إذا كان ع١ > ع٣ إذن:
إذا كان ع١ > ع٤ إذن:
اطبع على الشاشة: أكبر عدد هو ع١
غير ذلك:
اطبع على الشاشة: أكبر عدد هو ع٤
نهاية الشرط
غير ذلك إذا كان ع٣ > ع٤ إذن:
اطبع على الشاشة: أكبر عدد هو ع٣
غير ذلك:
اطبع على الشاشة: أكبر عدد هو ع٤
نهاية الشرط
غير ذلك إذا كان ع٢ > ع٣ إذن:
إذا كان ع٢ أكبر من ع٤ إذن:
اطبع على الشاشة: أكبر عدد هو ع٢
غير ذلك:
اطبع على الشاشة: أكبر عدد هو ع٤
نهاية الشرط.
غير ذلك إذا كان ع٣ > ع٤ إذن:
اطبع على الشاشة: أكبر عدد هو ع٣
غير ذلك:
اطبع على الشاشة: أكبر عدد هو ع٤
نهاية الشرط

تبدو الخوارزمية معقدة نوعا بسبب تداخل جمل الشرط.. لكن الحقيقة أن فكرتها بسيطة للغاية، ففي كل خطوة نقارن عددين، ونستبعد أصغرهما من المقارنات التالية، ونستمر في مقارنة العدد الكبير بباقي الأعداد.
على كل حال، المخطط سيجعل الأمر أبسط وأوضح، وهذه هي فائدته.. ها هو ذا:



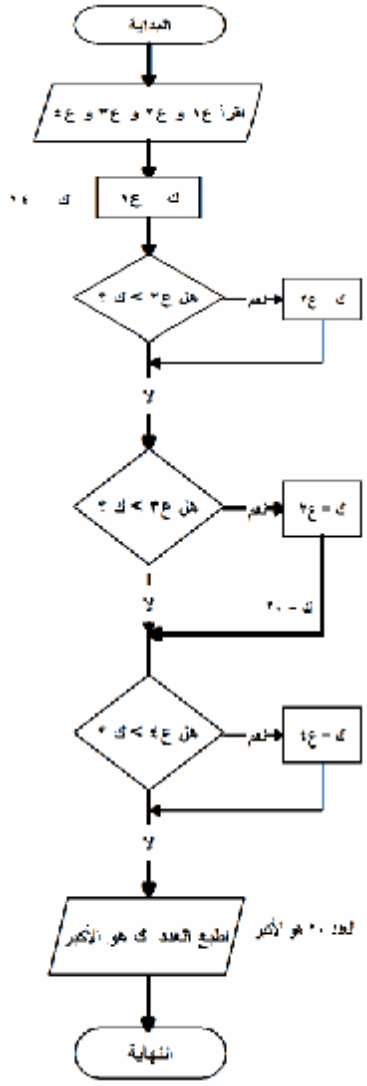
لاحظ أم المسار المحدد بخط سميك هو مسار تنفيذ البرنامج إذا كان ع١= ١٤ و ع٢ = ٥ و ع٣ = ٢٠ و ع٤ = ٩.. تلاحظ كما قلنا أن هناك ثلاث عمليات مقارنة فقط على هذا المسار.. حاول تغيير الأرقام وتتبع مسارات أخرى.. ستجدها جميعا تحتوي على ثلاث عمليات مقارنة فحسب.

حل آخر:

رغم أن الحل السابق سريع التنفيذ، إلا أنه معقد في كتابته.. ستتأكد من هذا لو حاولت أن تستخدم نفس هذه الطريقة لمقارنة ٥ أعداد، حيث ستجد أن جمل الشرط قد صارت معقدة بشكل كبير، وستزداد تعقيدا كلما زاد عدد الأعداد التي تتم مقارنتها.. دعنا إذن نستخدم خوارزمية أخرى للحل.. في هذه الخوارزمية سنستخدم متغيرا وليكن اسمه ك.. في البداية سنضع في ك العدد الأول، ثم سنقارن ك بباقي الأعداد، ولو وجدنا عددا أكبر من ك

سنضع قيمته في ك لنضمن أن ك سيظل دائما يحتوي على قيمة اكبر عـــدد.. هـــذه هـــي الخوارزمية:

ك = ع١.
إذا كان ع٢ > ك إذن: ك = ع٢.
إذا كان ع٣ > ك إذن: ك = ع٣.
إذا كان ع٤ > ك إذن: ك = ع٤.
اطبع على الشاشة: أكبر عدد هو ك



بسيطة جدا، ومختصرة جدا.. أليس كذلك؟
تعال نجربها على الأعداد ١٤ و ٥ و ٢٠ و ٩:

- في البداية سنضع فـــي ك العـــدد الأول، لهـــذا سيكون ك = ١٤.
- سنقارن ك بالعدد الثاني فنجد ك أكبر من ٥ لهذا لن نفعل شيئا.
- سنقارن ك بالعدد الثالث فنجد ك أصغر مـــن ٢٠ لهذا سنجعل ك = ٢٠ لأن هذا أكبر عدد وصلنا إليه حتى الآن.
- سنقارن ك بالعدد الرابع فنجد ك أكبر من ٩ لهذا لن نفعل شيئا.
- انتهت عمليات المقارنة، والمتغير ك الآن يحتوي على أكبر عدد، وهو ٢٠.. يمكننا أن نطبعه على الشاشة.

لاحظ أن هذه الطريقة ســـهلة التعمـــيم.. فلـــو أردت مقارنة ٥ أعداد فستضيف عملية مقارنة واحدة جديدة مع ك وهذا لا يحتوي على أي تعقيـــد.. ولـــو أردت مقارنة ٦ أعداد فستضيف عملية مقارنـــة أخـــرى.. وهكذا...

وحينما نتعلم المصفوفات Arrays وجمل التكرار Loops في فيجيوال بيزيك ستجد أنـــك تستطيع تعميم هذه الطريقة لأية مجموعة من الأرقام، ولن يزيد طول الكود الذي ســـتكتبه عن أربعة أسطر، مهما كان عدد هذه الأرقام!
والمخطط يوضح لك هذه الخوارزمية، مع ملاحظة أن المسار السميك عليـــه هـــو مســـار مقارنة الأعداد ١٤ و ٥ و ٢٠ و ٩.

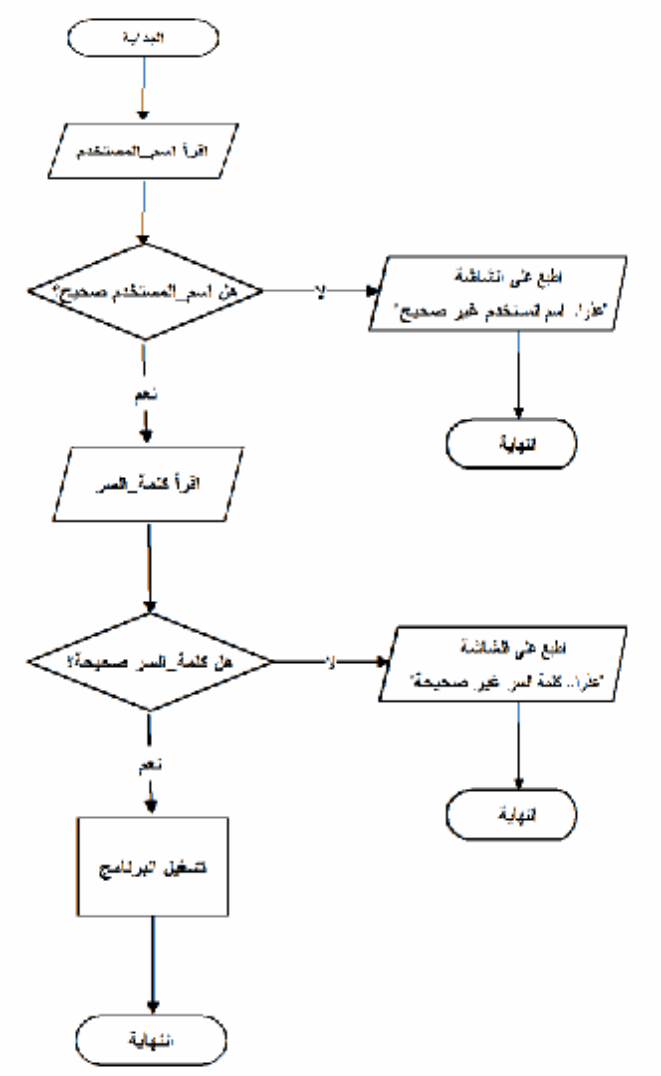
مثال ٦:
ارسم مخططا لبرنامج لا يبدأ العمل إلا بعد قراءة اسم المستخدم وكلمة السر والتأكد من أنهما صحيحان.

الحل:
سنستخدم في هذا البرنامج متغيرين:
١. اسم_المستخدم: سنقرأ في هذا المتغير اسم المستخدم الذي يكتبه مشغل البرنامج.
٢. كلمة_المرور: سنقرأ في هذا المتغير كلمة المرور التي يكتبها مشغل البرنامج.
لاحظ أن اسم أي متغير يجب ألا يحتوي على مسافات.. لهذا إذا أردت أن تستخدم اسم متغير يتكون من كلمتين، فيمكنك أن تربطهما معا باستخدام العلامة _ .. هذه العلامة تسمى الشرطة المنخفضة Underscore، وهي توجد مع علامة الطرح على نفس الزر، حيث يمكنك كتابتها بضغط زر التحويل Shift مع هذا الزر.
ستكون خوارزمية هذا البرنامج كالتالي:

إذا كان اسم_المستخدم خاطئا إذن:
اطبع على الشاشة: "عذرا.. اسم المستخدم غير صحيح".
غادر البرنامج.
نهاية الشرط.

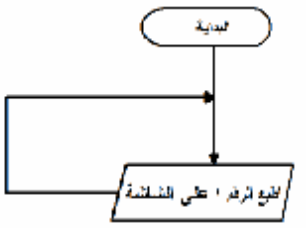
إذا كانت كلمة_السر خاطئة إذن:
اطبع على الشاشة: "عذرا.. كلمة السر غير صحيحة".
غادر البرنامج.
نهاية الشرط.

الخوارزمية بسيطة وواضحة.. وكذلك سيكون مخططها.. هذا هو:



مثال ٧:
ارسم مخططا لطباعة العدد ١ على الشاشة لعدد لا نهائي من المرات.

الحل:



كل ما سنفعله في هذا البرنامج هو استخدام التكرار Loop.
فكرة التكرار Loop بسيطة، فكل ما عليك هو القفز إلى خطوة سابقة لإعادة تكرارها.. هذا هو المخطط:
لاحظ أن رسم التكرار Loop على المخطط في منتهى

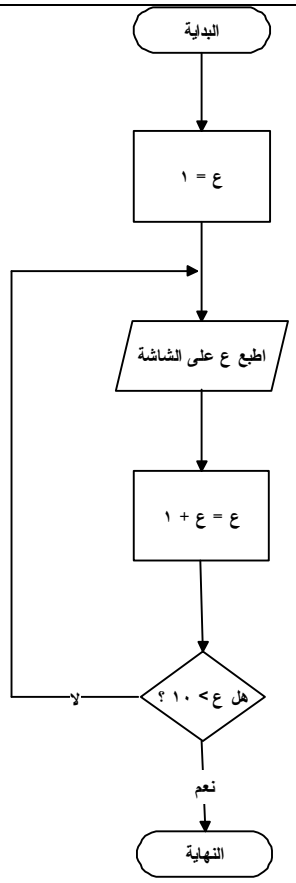
السهولة، فأنت ترسم سهما يبدأ من نقطة نهاية التكرار إلى نقطة بداية التكرار، بحيث تكون كل الخطوات التي يجب تكرارها محصورة بين هاتين النقطتين.
لاحظ كذلك أن المخطط لا يحتوي على رمز النهاية، لأن تنفيذه فعلا لن ينتهي أبدا!.. السبب في هذا هو أننا نقفز إلى الخلف لنعيد طباعة العدد ١، ثم نقفز إلى الخلف لنعيد طباعته مجددا وهكذا... هذا أشبه بالدوران في دائرة، لذا يسمى بعملية اللف Loop أو التكرار.
طبعا لا يمكن استخدام هذا التكرار بهذا الشكل في أي برنامج عملي، لأنه لن ينتهي أبدا.. لهذا يجب وضع شرط أو أكثر داخل الجزء الذي يتم تكراره، ليحدد متى يجب أن نتوقف.. انظر للمثال التالي.

مثال ٨:
اسم مخططا لطباعة الأعداد من ١ إلى ١٠ على الشاشة.

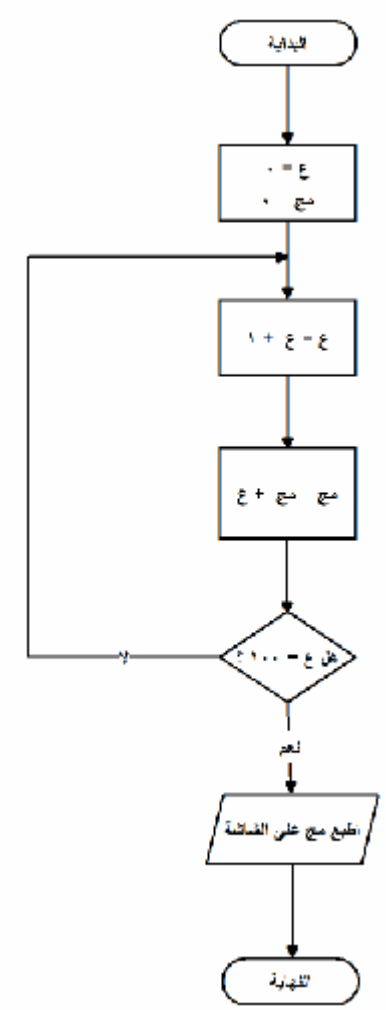
الحل:
سنستخدم في هذا المخطط متغيرا اسمه ع يعبر عن العدد الذي سنطبعه عن الشاشة.. هذا المتغير سيبدأ من العدد ١.. وسيتكون المقطع الذي سيتم تكراره من أمرين:
الأول: طباعة ع على الشاشة.
والثاني: زيادة ع بواحد.
وفي نهاية مقطع التكرار سنستخدم جملة شرط للتأكد إن كان ع أكبر ١٠، فإن لم يكن نواصل عملية التكرار، ولا نتوقف إلا إذا صار ع أكبر من ١٠.. أي أننا سنتوقف عندما تكون قيمة ع = ١١.
هذه هي خوارزمية البرنامج:

```
ع = ١ .
كرر:
    اطبع ع على الشاشة.
    ع = ع + ١ .
    إذا كان ع > ١٠ إذن:
        غادر البرنامج.
    نهاية الشرط.
نهاية التكرار.
```

والمخطط المجاور يوضح هذه الخوارزمية.



مثال ٩:
ارسم مخططا لبرنامج يقوم بجمع الأعداد من ١ إلى ١٠٠.



الحل:
هناك قانون رياضي لجمع الأعداد من ١ إلى ١٠٠ مباشرة، لكننا لا نريد استخدامه هنا.. فبدلا من هذا يمكن أن نستخدم التكرار Loop.. وسنستخدم في هذا البرنامج متغيرين:

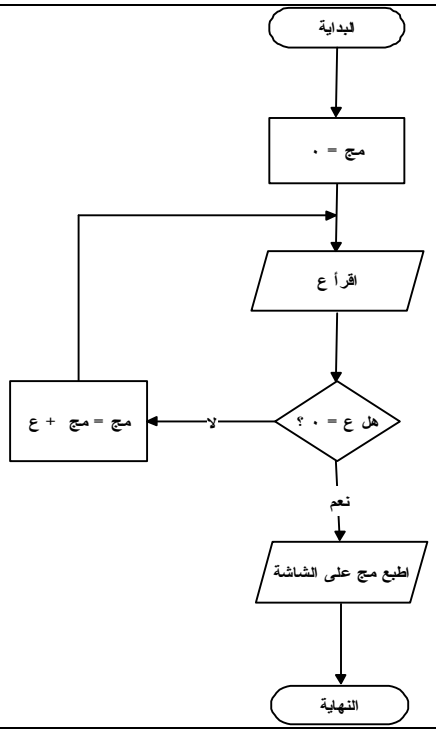
١. ع: عدد يبدأ بصفر، ويزيد بمقدار ١ في بداية كل لفة في جملة التكرار.. بهذه الطريقة سيأخذ هذا المتغير القيم: ١، ٢، ٣، ٤، ..... ، ١٠٠ بالترتيب.
٢. مج: متغير سنضع فيه مجموع الأعداد، وستبدأ قيمته بصفر، وفي كل لفة سنجمع عليه قيمة المتغير ع.. هذا يعني أننا في أول مرة سنجمع على هذا المتغير ١ فتصير قيمة مج = ١، وفي المرة الثانية ستصير قيمة ع = ٢ وحينما نجمعها على المجموع ستصير قيمة مج = ٣.. وفي اللفة الثالثة سصير ع = ٣ وسنجمعه على المجموع فيصيرمج = ٦ (١ + ٢ + ٣).. بهذه الطريقة سيتراكم مجموع الأعداد في المتغير مج إلى أن نصل إلى ع = ١٠٠ ونجمعه على مج فنحصل بهذا على الناتج النهائي.

لاحظ أننا نستطيع إدخال تعديل صغير على فكرة هذا البرنامج.. فبدلا من أن يبدأ ع ومج بالقيمة صفر، يمكن أن يبدأ كلاهما بالقيمة ١ مباشرة، فنختصر بهذا لفة واحدة من التكرار لا ضرورة لها.

مثال ١٠:
ارسم مخططا لبرنامج يقرأ الأعداد التي يدخلها المستخدم ويجمعها معا، ويستمر في فعل هذا إلى أن يدخل المستخدم العدد ٠ ، فيقوم البرنامج عندئذ بطباعة المجموع الكلي للأعداد على الشاشة.

الحل:
في هذا البرنامج أيضا سنستخدم المتغيرين ع ومج، ولكننا لن نزيد قيمة ع بواحد في كل لفة، بل سنضع فيه العدد الذي يكتبه المستخدم، ثم نفحص قيمته: فإن كانت صفرا نعرض

الناتج على الشاشة وننهي البرنامج، وإن كانت أي عدد غير الصفر نضيفه إلى المجموع القديم الموجود في المتغير مج.
المخطط موضح في الصورة المجاورة.



مثال ١١:
ارسم مخططا لجمع أول ٢١ عددا طبيعيا زوجيا.

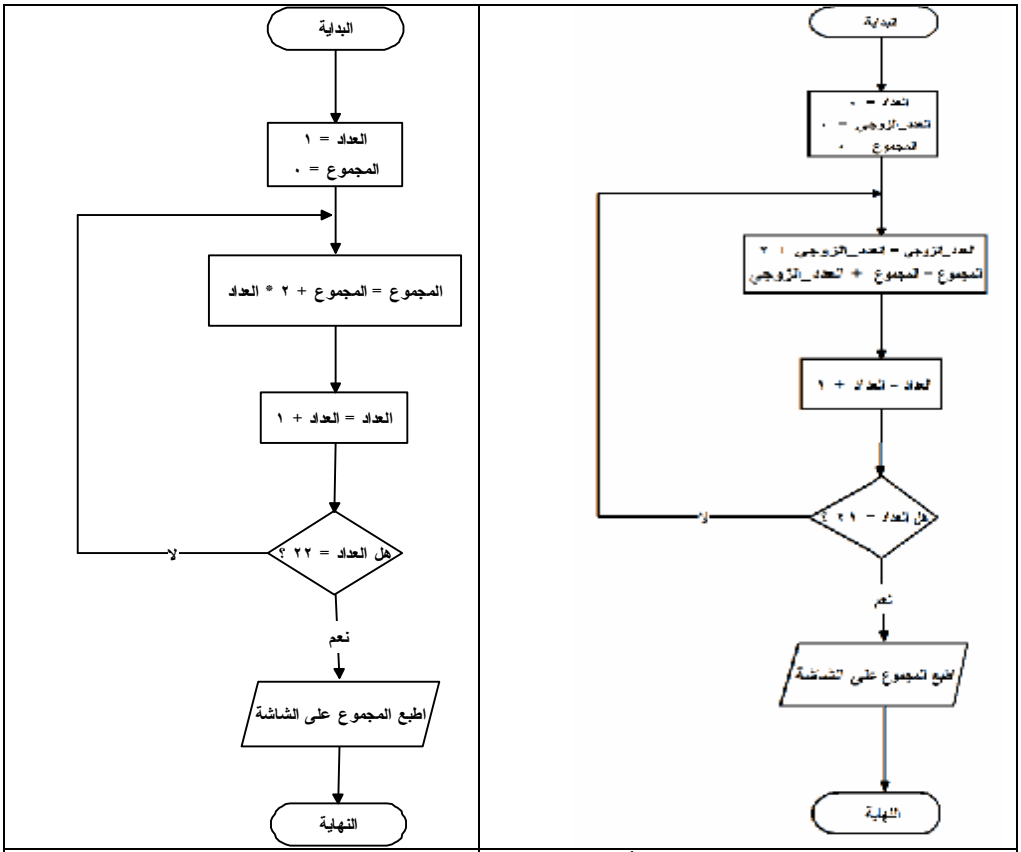
الحل:
سنستخدم في هذا الحل ثلاثة متغيرات:

١. العداد: سنستخدمه للعد من ١ إلى ٢١.
٢. العدد_الزوجي: سنضع به العدد الزوجي الذي وصلنا إليه، ومع كل زيادة للعداد سنضيف ٢ على هذا المتغير لنصل إلى العدد الزوجي التالي.. بهذه الطريقة ستكون قيم هذا المتغير هي: ٢، ٤، ٦، ..... ٤٠، ٤٢.
٣. المجموع: سنضع في هذا المتغير مجموع الأعداد الزوجية.

المخطط موجود على يمين الصفحة المقابلة.

حل آخر:
يمكن الاستغناء عن المتغير المسمى العدد_الزوجي في الحل السابق، وذلك لأننا نستطيع استنتاج العدد الزوجي من العداد مباشرة، لأنه ببساطة = ٢ × العداد.. لاحظ أن هذا يستدعي تعديلا صغيرا، فالعداد يجب أن تكون قيمته ١ في أول مرة لأن عدد زوجي طبيعي هو العدد ٢.. لهذا سنجعل القيمة الابتدائية للعداد = ١ في المخطط الجديد.. هذا بدوره يستدعي تطوير الشرط، فلو تركنا الشرط القديم كما هو فسينتهي التكرار عندما تصل قيمة ع إلى ٢١ دون أن نجمع ٢ * ٢١ على المجموع.. لهذا يجب تطوير الشرط

ليصير: "هل العدد = ٢٢ ؟" أو بطريقة أخرى: "هل العدد أكبر من ٢١ ؟"
المخطط الجديد موجود على يسار هذه الصفحة.



مخطط مثال ١١ (الحل الأول) | **مخطط مثال ١١ (الحل الثاني)**

تدريب:

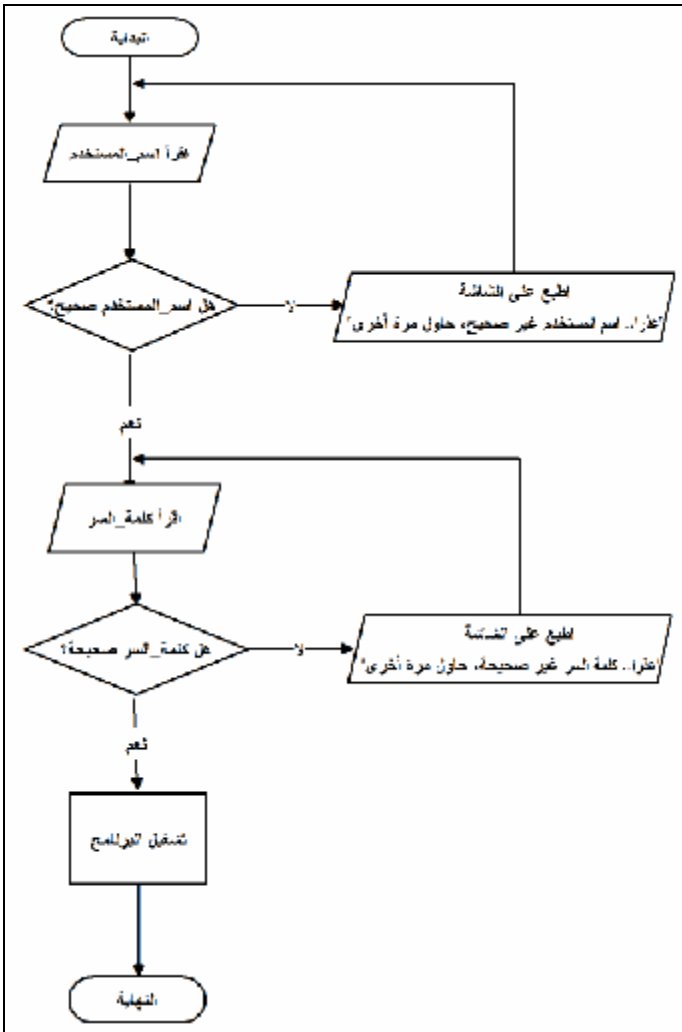
ارسم مخططا لجمع أول ٣٠ عددا طبيعيا فرديا.. لاحظ أن العدد الفردي سيرتبط بالعداد بالعلاقة التالية: العدد الفردي = ٢ * العداد −١
فمثلا: لو كان العداد = ١، فسيكون العدد الفردي = ٢ −١ = ١ .. ولو كان العداد = ٢ فسيكون العدد الفردي = ٤ − ١ = ٣... وهكذا.

مثال ١٢:

طور المثال رقم ٦، ليسمح للمستخدم بإعادة إدخال اسم المستخدم أو كلمة السر إن اخطأ فيهما، إلى أن يكتبهما بشكل صحيح.

الحل:

كل ما سنفعله هو استخدام التكرار في المخطط الذي رسمناه في المثال رقم ٦.. فبدلا من إنهاء البرنامج عندما يخطئ المستخدم في اسم المستخدم أو كلمة السر، سنرسم سهما يقفز بنا إلى الخلف إلى خطوة قراءة اسم المستخدم أو كلمة السر من جديد..هذا هو المخطط:

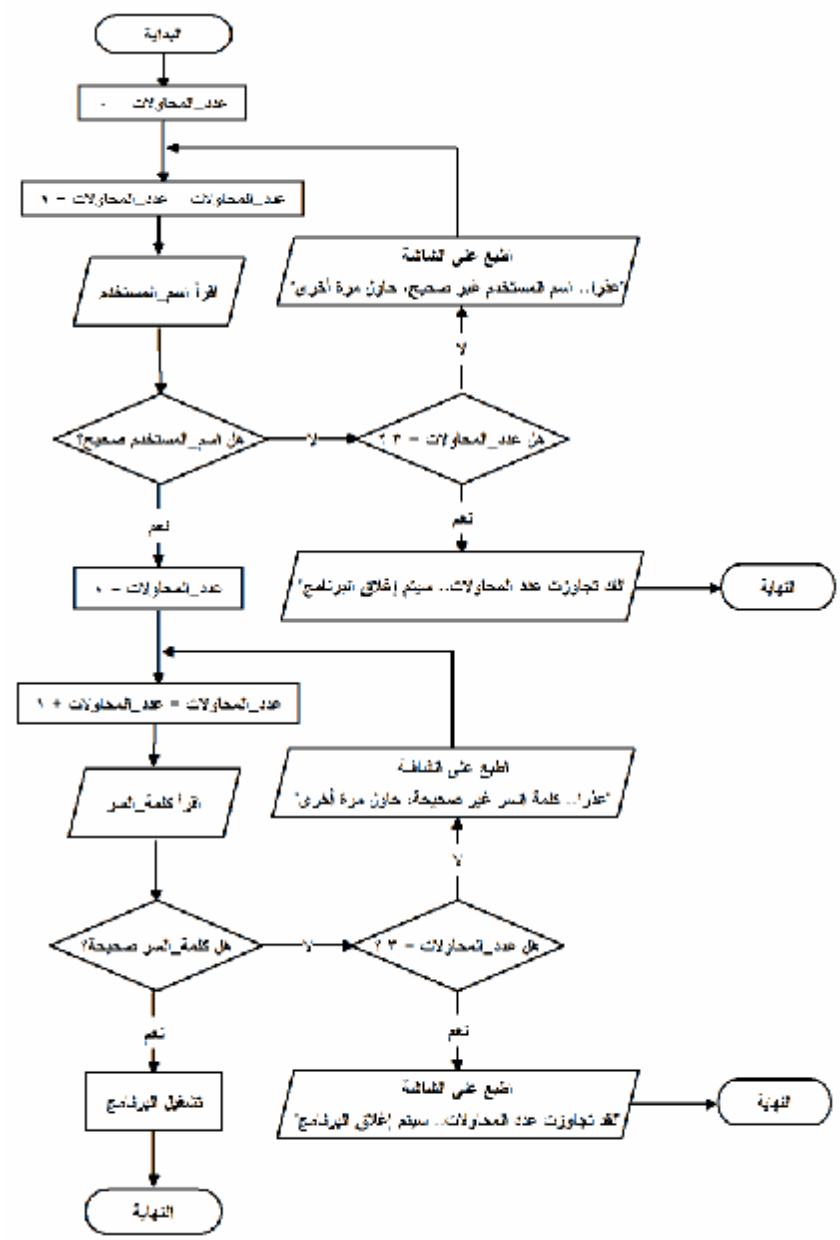


مثال ١٣:

قد يحاول بعض القراصنة Hackers تجريب إدخال عدد كبير من أسماء المستخدمين وكلمات المرور إلى أن ينجحوا في الوصول إلى الحل الصحيح وفتح البرنامج.. لتفويت هذه الفرصة عليهم، طور المثال السابق لجعل عدد مرات إدخال اسم المستخدم لا تزيد عن ٣ مرات، وكذلك الحال بالنسبة لكلمة السر.. ارسم المخطط الجديد الذي يفعل هذا.

الحل:

سنحتاج في هذا البرنامج المعدّل إلى متغير جديد اسمه عدد_المحاولات، وستكون قيمته المبدئية صفرا، وسنزيد قيمته في كل مرة يكتب فيها مشغل البرنامج اسم المستخدم.. وسنفحص قيمة هذا المتغير في كل مرة نكتشف فيها أن اسم المستخدم خطأ، فإن كانت قيمته = ٣ أغلقنا البرنامج، وإن كانت أقل من ٣ سمحنا للمستخدم بمواصلة المحاولة.
لاحظ أننا سنتبع نفس الخطوات بالنسبة لكلمة المرور، لهذا لا داعي لتكرار شرحها هنا.. هذا هو المخطط الجديد:



مثال ١٤: ابحث مع الشرطة:
يريد المفتش ذكي القبض على مجرم خطير، أفاد الشهود أن له المواصفات التالية:
- شعره مجعد.
- أنفه ضخم.
- شفتاه غليظتان.

ونظرا لأن المفتش ذكي مشغول بالكثير من القضايا، وهناك ١٠٠٠٠ مشتبه به توجد صورهم في قاعدة بيانات الشرطة، فهو يريد منك كتابة برنامج لمساعدته في التعرف على صورة المجرم من بين صور المشتبه فيهم.. هل تستطيع رسم مخطط لهذا البرنامج، بافتراض أن هناك شخصا واحدا فقط من بين المشتبه فيهم هو الذي تنطبق عليه هذه الصفات؟

الحل:
في هذا البرنامج سنتعامل مع الصور، وهي عملية تتسم بالبطء خاصة عندما يكون عدد الصور ضخما.. لهذا فليس من الذكي أن نفحص الصفات الثلاث للمجرم معا في كل صورة، لأن هذا سيضيع الكثير من الوقت.. الأذكى من هذا أن نفحص كل صفة على حدة، فلو لم تكن هذه الصفة موجودة في صورة المشتبه فيه، فلا داعي لفحص باقي الصفات، لأن عدم وجود صفة واحدة يكفي لاستبعاد المشتبه فيه.
أيضا، ليس من المنطقي أن نستمر في فحص باقي صور المشتبه فيهم بعد العثور على المجرم.. افرض مثلا أننا عثرنا على المجرم عند فحص الصورة العاشرة، فهل من المنطقي أن نستمر في فحص ٩٩٩٠ صورة بلا جدوى؟
بناء على هذا يمكننا أن نكتب الخوارزمية كالتالي:

لكل صورة من صور المشتبه فيهم، كرر ما يلي:
إذا كان الشعر مجعدا إذن:
إذا كان الأنف ضخما إذن:
إذا كانت الشفتان غليظتين إذن:
اطبع على الشاشة صورة المجرم وبياناته.
غادر البرنامج.
نهاية الشرط
نهاية الشرط
نهاية الشرط
نهاية التكرار.
اطبع على الشاشة "لم يتم العثور على المجرم"

في هذه الخوارزمية نلاحظ ما يلي:

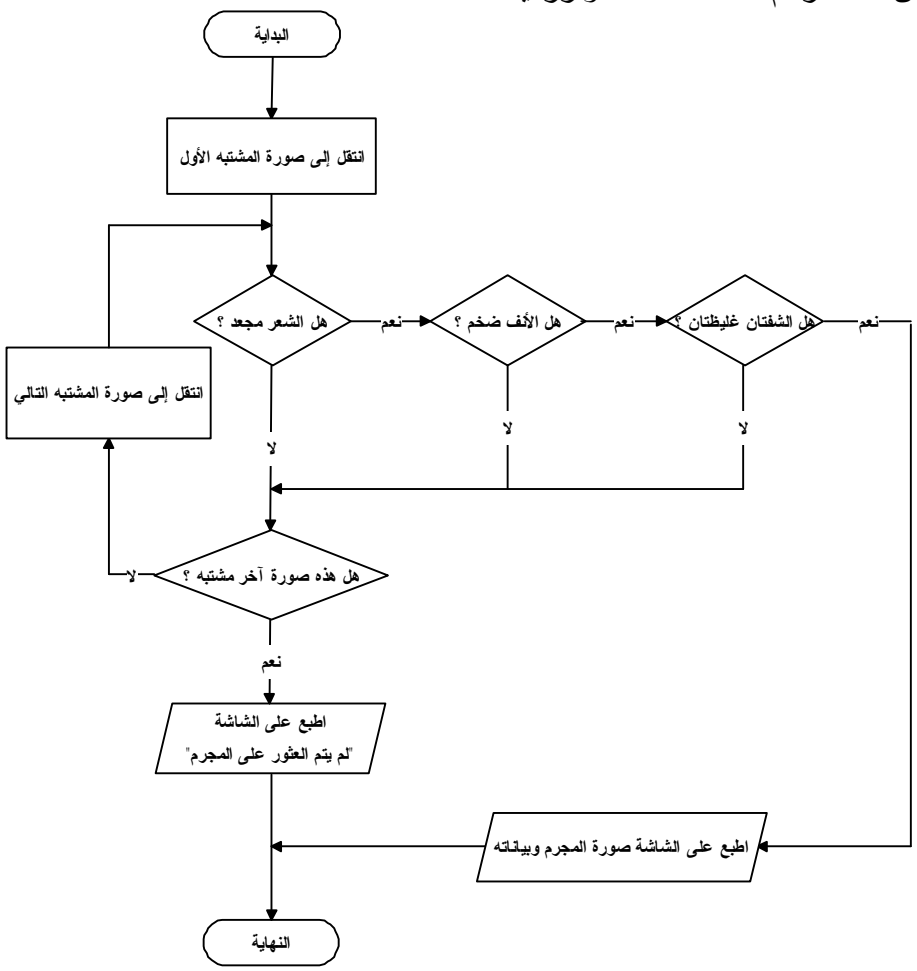
١- استخدمنا ثلاثة شروط متداخلة Nested If.. هذه الطريقة تضمن عدم تنفيذ أي شرط إلا إذا كان الشرط السابق له صحيحا.. هذا عكس استخدام جملة "غير ذلك" Else.

٢- إذا فشل أي شرط من الشروط الثلاثة فلن يتم فحص الشرط التالي له.. هذا يوفر الكثير من الوقت عند فحص الصور.

٣- إذا تحققت الشروط الثالثة معا، فستتم طباعة صورة المجرم وبياناته، ومغادرة البرنامج.. هذا يضمن عدم مواصلة فحص باقي صور المشبه فيهم، كما سيضمن عدم طباعة الجملة "لم يتم العثور على المجرم".. هذا لأن مغادرة البرنامج تعني عدم تنفيذ أي شيء آخر فيه.

٤- إذا وصل البرنامج إلى آخر صورة ولم يعثر على المجرم، فسيخرج من جملة التكرار، وبالتالي سيطبع على الشاشة الجملة: "لم يتم العثور على المجرم".. هذا يعني أن هذا المجرم ليس له سجل إجرامي سابق لدى الشرطة.

والآن دعنا نرسم مخطط هذه الخوارزمية:

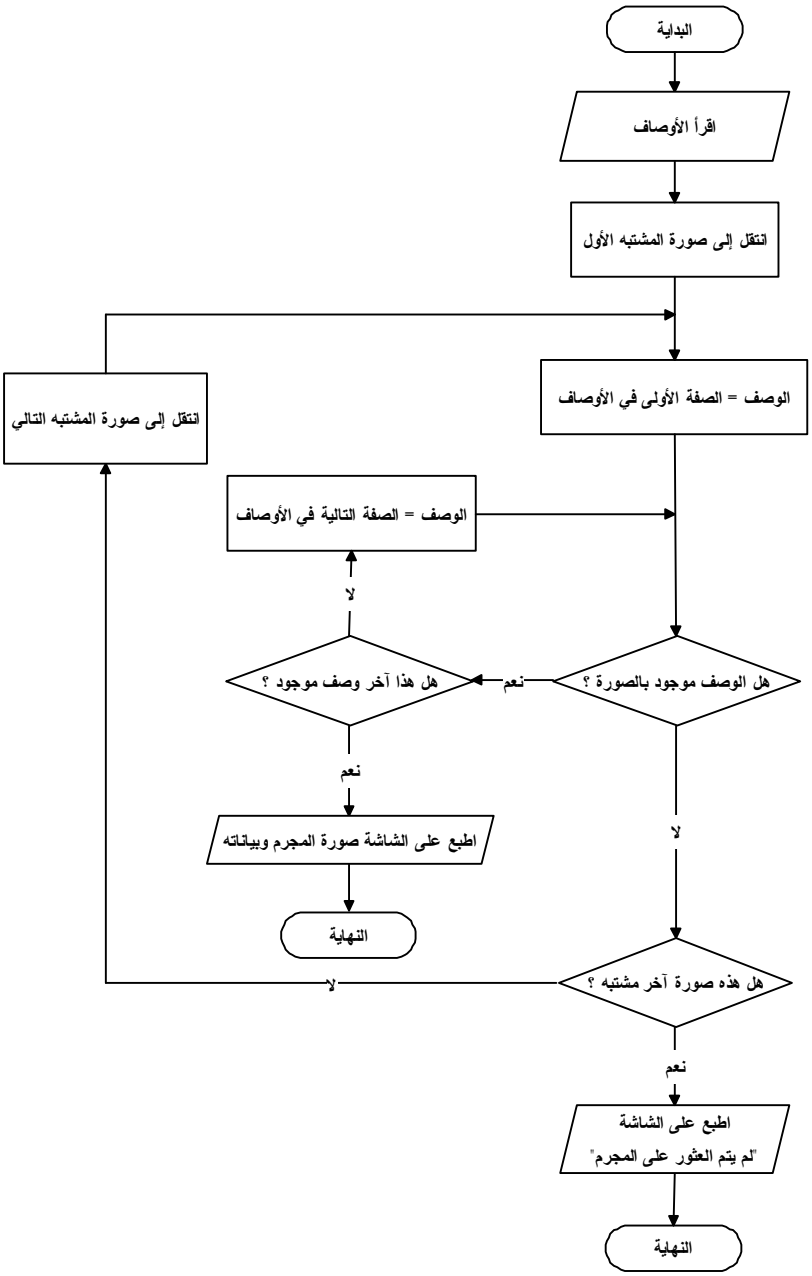


مثال ١٥:
المفتش ذكي ممتن منك كثيرا لمساعدته في القبض على المجرم الخطير، ولكنه يطلب منك تطوير البرنامج السابق، فهو لا يصلح للبحث إلا عن مجرم واحد وقد تم القبض عليه بالفعل!.. لهذا يحلم المفتش بوجود برنامج يسمح له بإدخال أي عدد من الأوصاف لأي مجرم أدلى بها الشهود.. هل تستطيع رسم مخطط جديد يحقق للمفتش ذكي ما يحلم به؟

الحل:
يمكن تطوير المثال السابق بقراءة مجموعة من الأوصاف من المفتش ذكي، ثم المرور عبر كل وصف منها وفحص إن كان هذا الوصف موجودا في صورة المشتبه الحالي أم لا.. فإن كان أحد الأوصاف غير موجود في الصورة الحالية فلا داعي لإكمال فحص باقي الأوصاف، وعلينا الانتقال إلى صورة المشتبه التالي وإعادة نفس العملية.. أما إن عثرنا على كل الأوصاف في نفس الصورة، فهذا يعني أننا وصلنا إلى المجرم الذي نبحث عنه، وعلينا طباعة بياناته ومغادرة البرنامج.
لاحظ أن تنفيذ هذا يحتاج إلى استخدام جملة تكرار Loop خاصة بالأوصاف، وأن هذه الجملة التكرارية ستكون داخل جملة التكرار التي تمر عبر صور كل المشتبه فيهم.. يسمى هذا بجمل التكرار المتداخلة Nested Loops.
مخطط البرنامج المعدل في الصفحة المقابلة.

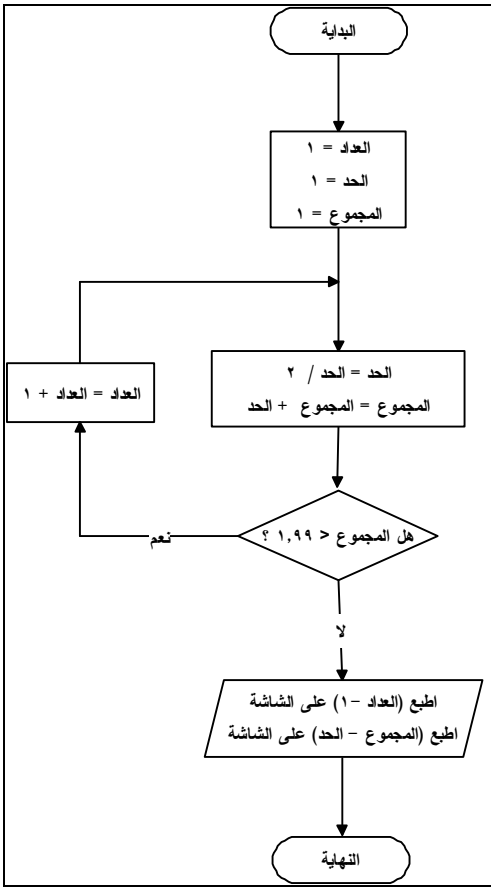
مثال ١٦:
ارسم مخططا لبرنامج يعرض على الشاشة عدد الحدود التي مجموعها أقل من ١,٩٩ في المتتابعة التالية: ١ + ١/٢ + ١/٤ + ١/٨ + ١/١٦ + ١/٣٢ + ...
ويعرض البرنامج مجموع هذه الحدود أيضا.

الحل:
بملاحظة حدود المتتابعة اللانهائية، سنجد أن المقام في أي حد هو ضعف مقام الحد السابق له.. إذن فنحن نضرب المقام في ٢ باستمرار، أو بمعنى آخر: الحد الجديد = الحد السابق مقسوما على ٢.. هذا يعني أن:
الحد = الحد / ٢
يمكننا إذن استخدام المتغيرات التالية:
١- العداد: يبدأ من صفر ونزيد قيمته بواحد في كل لفة.
٢- الحد: تبدأ قيمته بواحد، ونقسمه على ٢ في كل لفة.
٣- المجموع: يبدأ من صفر ونجمع عليه الحد في كل لفة.
لاحظ أننا نتعامل مع كسور عشرية، ولا نضمن أن نحصل على مجموع = ١,٩٩ بالضبط، لهذا لا نستخدم الشرط: "هل المجموع = ١,٩٩؟"، لأنه قد لا يتحقق أبدا مما يؤدي إلى استمرار تنفيذ البرنامج إلى ما لا نهاية!


البداية
اقرأ الأوصاف
انتقل إلى صورة المشتبه الأول
الوصف = الصفة الأولى في الأوصاف
انتقل إلى صورة المشتبه التالي
الوصف = الصفة التالية في الأوصاف
لا
هل الوصف موجود بالصورة ؟
نعم
هل هذا آخر وصف موجود ؟
نعم
اطبع على الشاشة صورة المجرم وبياناته
النهاية
لا
هل هذه صورة آخر مشتبه ؟
لا
نعم
اطبع على الشاشة
"لم يتم العثور على المجرم"
النهاية

وبدلا من هذا سنتحقق في نهاية كل لفة إن كان المجموع قد زاد عن ١,٩٩ أم لا.. فإن تحقق هذا الشرط فهذا يعني أننا لا نحتاج إلى آخر حد من المتتابعة جمعناه على المجموع لأنه هو سبب تجاوز القيمة ١,٩٩.. لهذا السبب سنطرح ١ من العداد لنحصل على عدد الحدود المطلوبة، كما سنطرح من المجموع آخر حد (وقيمته ما زالت محفوظة في المتغير المسمى: الحد) وبهذا نحصل على مجموع الحدود الأصغر من ١,٩٩ في المتتابعة.
المخطط في الصورة المجاورة.

ملحوظة:
هناك صيغة عامة لكل حد، وهي:
الحد = الحد / ٢ أس العداد
ويمكنك استخدامها بدلا من العلاقة:
الحد = الحد / ٢



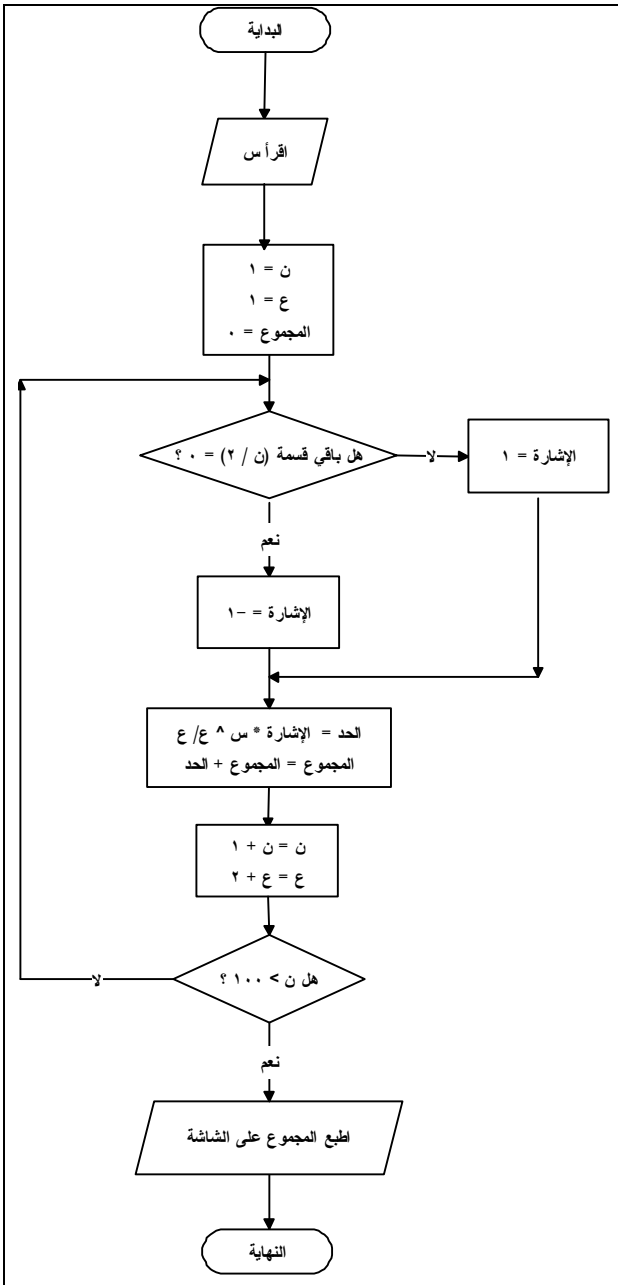
مثال ١٧:
ارسم مخططا لحساب قيمة الدالة ص بناء على قيمة س التي يدخلها المستخدم، علما بأن ص = س – س^{٣} / ٣ + س^{٥} / ٥ – س^{٧} / ٧ + ...
وعلما بأن عدد حدود هذه المتابعة = ١٠٠ حد.

الحل:
- نلاحظ أن الأس يساوي المقام في كل حد من حدود المتتابعة، وأنهما يبدآن بالعدد ١ ثم يزيدان بمقدار ٢ في كل حد (١، ٣، ٥، ٧، .....).. بناء على هذه الملاحظة، وللحصول على الأس والمقام، استخدم متغيرا اسمه ع تبدأ قيمته بـ ١، وفي نهاية كل لفة اجمع عليه ٢ لتصير قيمته ٣، ثم ٥، ثم ٧، .....

- تتبقى إشارة كل حد.. كما تلاحظ، فإن إشارة كل حد تتناوب بين الموجب والسالب، فهناك حد موجب يليه حد سالب يليه حد موجب وهكذا... دعنا نستخدم متغيرا اسمه ن ليمثل رقم الحد، حيث ستبدأ قيمته بواحد للإشارة إلى الحد الأول، ثم ستزيد في كل لفة بمقدار ١.. ويمكن استخدام ن لمعرفة الإشارة، فالحد الزوجي هو الذي يقبل القسمة على ٢ بدون باقي، لهذا سنقسم ن على ٢ ونفحص الناتج، فإن كان الناتج بدون باق سنجعل الإشارة سالبة (لأن الحدود الزوجية في هذه المتتابعة سالبة).. أما إن كان هناك باق فسنجعل الإشارة موجبة:

إذا كان باقي قسمة (ن / ٢) = ٠ إذن:
الإشارة = −١
غير ذلك:
الإشارة = ١
نهاية الشرط

المخطط موضح في الصورة المجاورة.



تدريب ١:

ارسم مخططا لحساب قيمة الدالة ص بناء على قيمة س التي يدخلها المستخدم، علما بأن ص = س^{٢٨} / ٢ – س^{٤٨} /٤ + س^{٦٨} / ٦ – س^{٨٨} /٨+ .....

وعلما بأن عدد حدود هذه المتابعة = ١٠٠ حد.

مساعدة:

لاحظ أن كل أسس هذه المتتابعة زوجية، لهذا سيكون الحل مماثلا لحل مثال ١٧، لكنك ستجري فقط تغييرا واحدا، هو أنك ستجعل القيمة الابتدائية للمتغير ع = ٢.. سأترك لك رسم المخطط الجديد بنفسك للتدريب.

تدريب ٢:

ارسم مخططا لحساب قيمة الدالة ص بناء على قيمة س التي يدخلها المستخدم، علما بأن ص = ١ + س – س^{٣٨} / ٣ + س^{٥٨} /٥ – س^{٧٨} / ٧ + ...

وعلما بأن عدد حدود هذه المتابعة = ١٠٠ حد.

مساعدة:

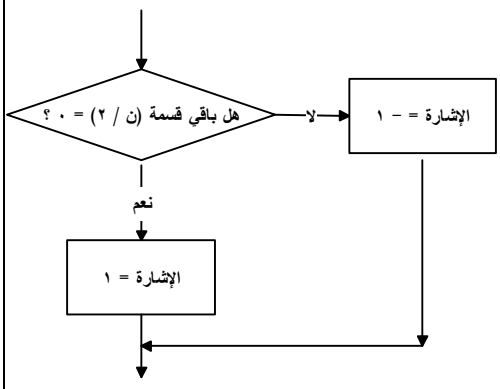
هذه المسألة مشابهة للمثال ١٧، مع فارق واحد فقط، هو أن العدد ١ موجود في بداية المتتابعة.. وبتفكير بسيط ستجد أن الحل هو نفس حل مثال ١٧، لكن المجموع النهائي سيزيد بواحد فقط.. لهذا يمكنك أن تجعل القيمة الابتدائية للمجموع = ١ بدلا من صفر في بداية المخطط، ولن تحتاج إلى تغيير أي شيء آخر في الحل!

تدريب ٣:

ارسم مخططا لحساب قيمة الدالة ص بناء على قيمة س التي يدخلها المستخدم، علما بأن ص = ١ – س + س^{٣٨} / ٣ – س^{٥٨} /٥ + س^{٧٨} / ٧ – .....

وعلما بأن عدد حدود هذه المتابعة = ١٠٠ حد.

مساعدة:



هذه المسألة مماثلة للمثال ١٧، ما عدا وجود فارقين:

الأول: أن هناك ١ في بداية المتتابعة، وقد رأينا في تدريب ٢ كيف يمكننا إضافته للمجموع بجعل القيمة الابتدائية للمجموع = ١.

الثانية: أن إشارات الحدود معكوسة، لهذا كل ما ستفعله هو تعديل بسيط في المخطط لعكس الإشارة في جملة الشرط، كما في الصورة.

حل آخر للمثال ١٧:

يمكن حل أية متتابعة بمعرفة الصيغة العامة لأي حد فيها، حيث سنحسب قيمة كل حد ونضيفه إلى المجموع في كل لفة.. بتأمل هذه المتابعة سنجد أن الحد العام لها هو:

$$\frac{(-1)^{ن-1} \times س^{٢ن-١}}{٢ن - ١}$$

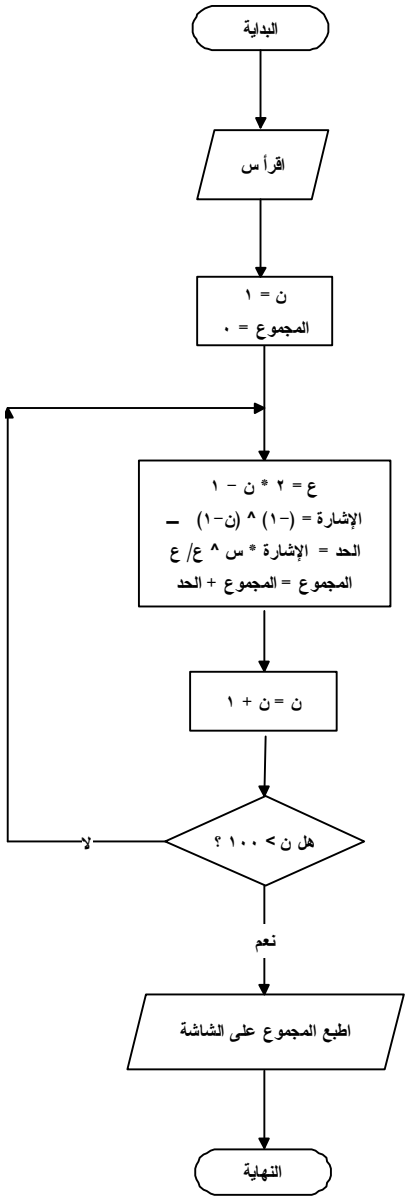
حيث ن هو رقم الحد.

لكي تفهم فكرة هذا الحد العام لاحظ ما يلي:

- ٢ ن – ١ هي الصيغة التي تضمن لنا الحصول على عدد فردي دائما.. هذه الصيغة ستعطينا الأس والمقام في كل من حدود المتتابعة.
- لاحظ أنك لو كنت تتعامل مع متتابعة كل أسسها زوجية، فيمكنك استخدام الصيغة العامة (٢ن) للحصول على أعداد زوجية دائما.
- تتبقى صيغة الإشارة.. لاحظ أنه لو كانت لدينا متتابعة حدودها الفردية سالبة وحدودها الزوجية موجبة، فإن صيغة الإشارة ستكون (–١) أس ن.. السبب في هذا أن العدد السالب عند رفعه لأس فردي يعطي عددا سالبا، وعند رفعه لأس زوجي يعطي عددا موجبا.. لهذا ستعطينا هذه الصيغة –١ ، ١، –١ ....

لكن في مسألتنا هذه، الحدود الفردية إشارتها موجبة والحدود الزوجية إشارتها سالبة.. أنت تعرف أن كل عدد زوجي يسبقه عدد فردي ويليه عدد فردي.. إذن يكفي لتعديل الصيغة أن نطرح ١ من ن أو نضيف ١ إلى ن، لنحصل على أس معاكس لـ ن في نوعه، فإن كانت ن فردية حصلنا على أس زوجي، وإن كانت ن زوجية حصلنا على أس فردي.. إذن فالصيغة العامة التي تعطينا إشارة المتتابعة هنا هي (–١) أس (ن –١).

والآن بعد أن عرفنا الحد العام، من السهل علينا أن نستخدم جملة تكرارية تبدأ من ن = ١ إلى ن = ١٠٠، ولكل قيمة ن نحسب الحد ونجمعه على المجموع، كما في الشكل.



مثال ١٨:

ارسم مخططا لبرنامج يقرأ البيانات التالية:

- عدد ساعات العمل التي عملها الموظف.
- وأجر الموظف في الساعة.
- عدد ساعات تأخر الموظف في الحضور إلى العمل.

ومن ثم يحسب البرنامج إجمالي راتب الموظف بعد خضم الضرائب، علمًا بأن:

- الضرائب تخصم بمعدل ١٥% من إجمالي الراتب.
- وأن العامل يستحق ١,٢ من أجر الساعة عن كل ساعة عمل إضافية بعد أول ١٥٠ ساعة عمل.
- وأن كل ساعة تأخير في الحضور يخصم مقابلها ١,٥ من أجر الساعة.

الحل:

نحتاج إلى تحليل معطيات هذه المسالة وتعريف متغيراتها أولا لنستطيع حلها:

١- ساعات_العمل: سنقرأ في هذا المتغير عدد ساعات عمل الموظف.

٢- ساعات_التأخير: سنقرأ في هذا المتغير عدد ساعات تأخر الموظف في الحضور إلى العمل.

٣- أجر_الساعة: سنقرأ في هذا المتغير الأجر الذي يتقاضاه الموظف في الساعة.

٤- الخصم: سنضع في هذا المتغير ما سيتم خصمه من الموظف بسبب التأخير، وهو يحسب كالتالي:

الخصم = ساعات_التأخير × ١,٥ × أجر_الساعة.

٥- الأجر: سنضع في هذا المتغير الأجر المستحق عن ساعات العمل.. لدينا هنا حالتان:

أ. إذا كان عدد ساعات العمل أقل من أو يساوي ١٥٠ ساعة، فسيكون أجر الموظف كالتالي:

الأجر = ساعات_العمل × أجر_الساعة.

ب. إذا كان عدد ساعات العمل أكبر من ١٥٠ ساعة، فسيستحق الموظف أجرا مقداره ١,٢ من أجر الساعة عن كل ساعة عمل إضافية.. بناء على هذا سيكون الأجر هو الأجر العادي عن أول ١٥٠ ساعة مضافا إليه الأجر الإضافي عن الساعات الزائدة.. أي أن:

الأجر = ١٥٠ × أجر_الساعة + (ساعات_العمل −١٥٠) × ١,٢ × أجر_الساعة.

لا توجد مشكلة في استخدام هذه الصيغة كما هي.. لكن لو أردت، فيمكنك تبسيط هذه الصيغة رياضا إلى:

الأجر = [١٥٠ + (ساعات_العمل −١٥٠) × ١,٢] × أجر_الساعة.

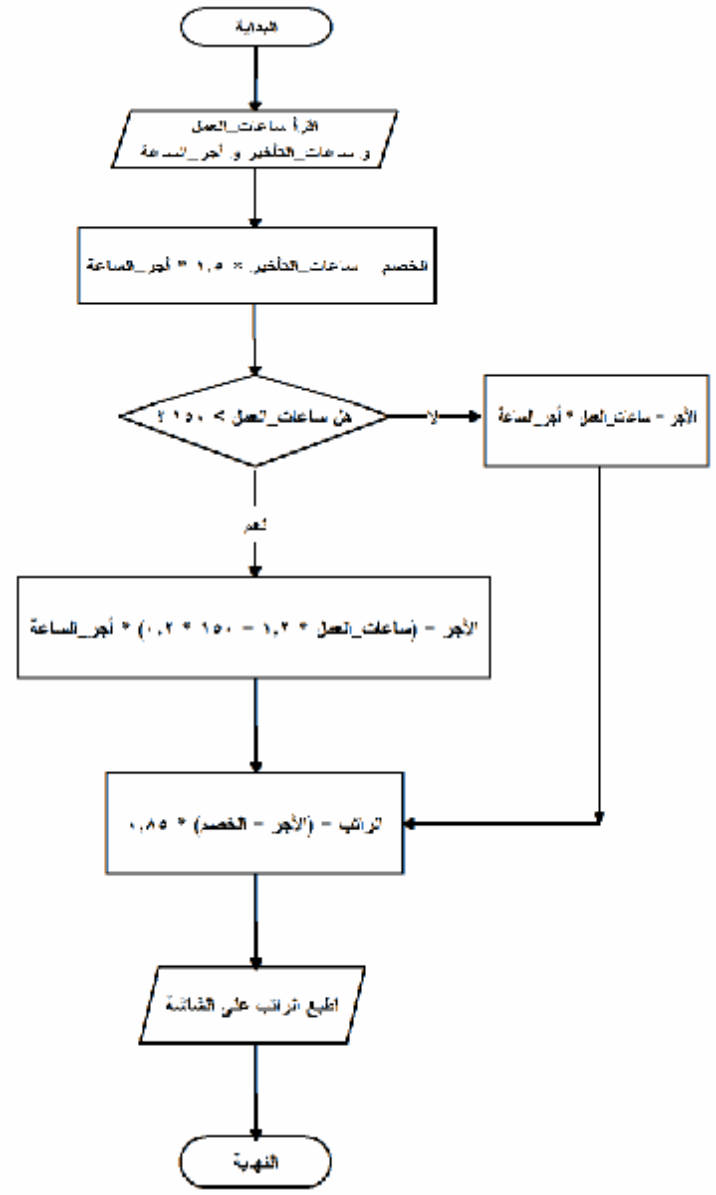
وهو ما يمكن تبسيطه مرة أخرى ليصير:

الأجر = (ساعات_العمل × ١,٢ − ١٥٠ × ٠,٢) × أجر_الساعة.

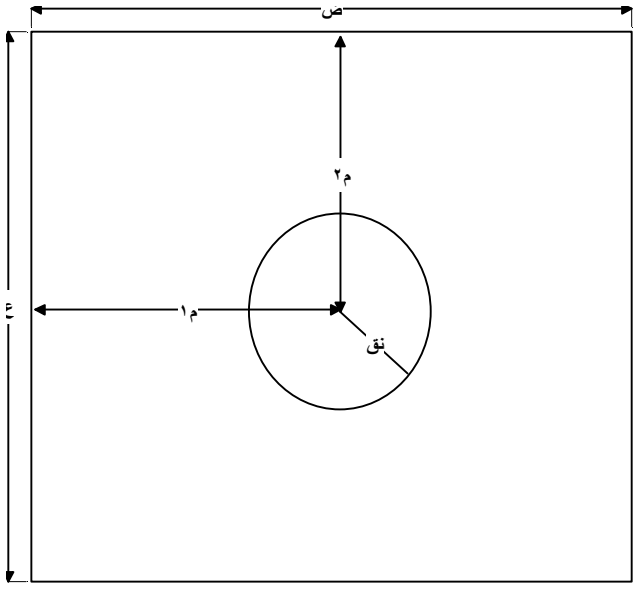
ج. الراتب: سنضع في هذا المتغير صافي الراتب بعد الخصم والضرائب.. فإذا كانت الضرائب تساوي ١٥% فإن صافي الراتب سيكون ٨٥% من الأجر بعد الخصم.. أي أن:

الراتب = (الأجر – الخصم) × ٠,٨٥

والمخطط التالي يوضح كيفية تنفيذ هذه الخطوات.



مثال ١٩ (للمتفوقين):
ارسم مخططا يوضح حركة كرة نصف قطرها نق داخل نافذة عرضها ض وارتفاعها ع، علما بأن الكرة تتحرك في كل خطوة بمقدار س في الاتجاه الأفقي و ص في الاتجاه الرأسي، وعند اصطدامها بأي حافة من حواف النافذة ترتد عنها في الاتجاه العكسي.. اعتبر النقطة (صفر، صفر) هي رأس النافذة العلوي الأيسر.



الحل:
هذا المثال يعلمك كيف تبرمج جسما متحركا.
لكي يستمر هذا الجسم في الحركة، فإنا نغير موضعه الأفقي والرأسي داخل جملة تكرار Loop ليظهر متحركا باستمرار.
بالنسبة للكرة، فإننا نقيس موضعها بمعرفة الموضع الأفقي م١ والرأسي م٢ لنقطة مركزها.. وفي كل لفة سنزيد هذا الموضع بإضافة س إلى م١، و ص إلى م٢.. لاحظ أن س و ص يحددان سرعة الكرة واتجاه حركتها.. فكلما زادت قيمة س و ص زادت سرعة الكرة.. السؤال هو: كيف تؤثر س وص على الاتجاه؟
أنت تعرف من حساب المثلثات أن الكرة ستتحرك بزاوية هـ يمكن حسابها من العلاقة: ظا هـ = ص / س.. على كل حال هذا لا يؤثر في كتابة الكود، ولكنه يوضح لك كيف سيؤثر تغيير قيمتي س و ص على زاوية تحرك الكرة.
ما يهمنا هنا هو أن نتأكد أن الكرة لم تصطدم بأية حافة بعد تحريكها.. لدينا هنا أربعة حواف للنافذة، وهي:

١- الحافة اليسرى، وأية نقطة عليها موضعها الأفقي = صفر.. ستصطدم الكرة بهذه الحافة إذا كانت المسافة بين مركز الكرة والحافة أصغر من صفر.. أي:

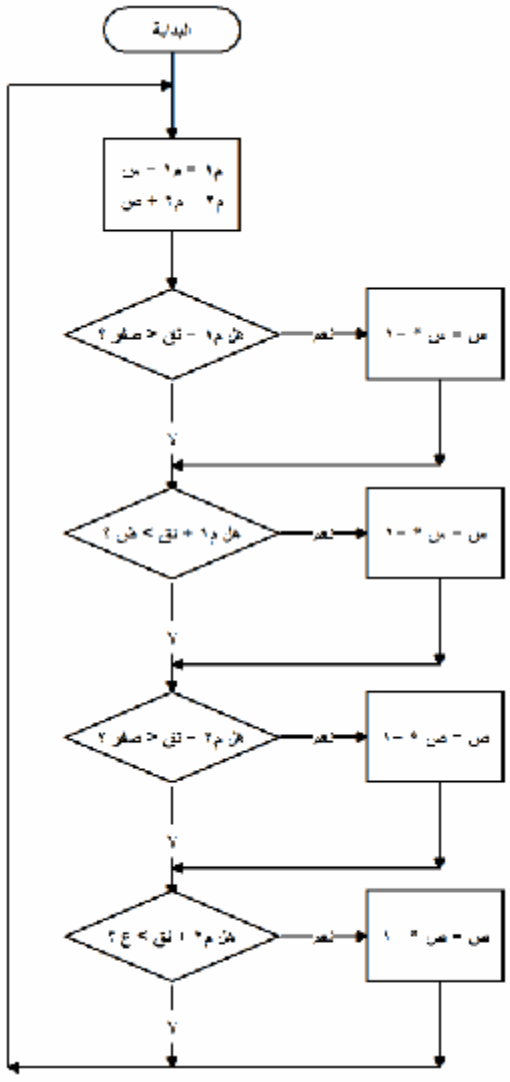
م١ – نق < صفر.

ولو اصطدمت الكرة بهذه الحافة فعلينا عكس اتجاه حركتها الأفقي، لكن دون تغيير سرعة الكرة.. يحدث هذا بضرب س × –١ لعكس إشارة مقدار الحركة الأفقية.. تذكر أن م١ + (–س) = م١ – س.. هذا يوضح لك كيف يؤدي عكس الإشارة إلى تغيير اتجاه الحركة.

٢– الحافة العليا، وأية نقطة عليها موضعها الرأسي = صفر.. ستصطدم الكرة بهذه الحافة إذا كان: م٢ – نق < صفر. لو اصطدمت الكرة بهذه الحافة فعلينا عكس اتجاه حركتها الرأسي، دون تغيير سرعة الكرة.. يحدث هذا بضرب ص × –١ لعكس إشارة مقدار الحركة الرأسية.

٣– الحافة اليمنى، وأية نقطة عليها موضعها الأفقي = ض.. ستصطدم الكرة بهذه الحافة إذا كان: م١ + نق > ض.. لو اصطدمت الكرة بهذه الحافة فعلينا عكس اتجاه حركتها الأفقية بضرب س × –١.

٤– الحافة السفلى: وأية نقطة عليها موضعها الرأسي = ع.. ستصطدم الكرة بهذه الحافة إذا كان: م٢ + نق > ع.. لو اصطدمت الكرة بهذه الحافة فعلينا عكس اتجاه حركتها الرأسية، بضرب ص × –١.



دعنا نرى كيف سيكون هذا المخطط.

لاحظ عدم وجود رمز نهاية البرنامج، لأن الكرة ستستمر في الحركة بلا توقف.

مميزات المخطط:

١- وسيلة اتصال سهلة، لنقل الأفكار البرمجية وشرح حلول المسائل للآخرين.
٢- وسيلة فعالة وواضحة لتحليل المسألة واكتشاف الحلول.
٣- أداة هامة لتوثيق البرامج وحفظها على الورق.
٤- تحقق كفاءة أعلى للمبرمج عند كتابة الكود، حيث تفصل حل المشكلة عن كتابة الكود، مما يبسط كتابة الكود.
٥- تسهل تتبع خطوات الحل، مما يساعد على كشف أية أخطاء موجودة فيه.
٦- تسهل إصلاح البرنامج، لأن الرجوع إلى المخطط بعد فترة طويلة يسهل على المبرمج تذكر البرنامج وفهم كيفية عمله، مما يسهل عليه تعديل البرنامج أو إصلاح أخطائه.

عيوب المخطط:

١- إذا كانت المسألة معقدة، يصير المخطط معقدا وتصعب متابعته.
٢- قد يحتوي المخطط على الكثير من التفاصيل، مما يجعل تتبع حل بعض المسائل صعبا.
٣- يتطلب إجراء تعديلات في الفكرة إعادة رسم المخطط من جديد.. لاحظ أن هذه المشكلة خاصة بالرسوم الورقية فحسب، لكن هناك برامج حاسوبية لرسم المخططات، والتعديل فيها في منتهى السهولة.
٤- صعوبة نسخ المخططات.. هذا أيضا خاص بالرسوم الورقية، لأن برامج رسم المخططات تتيح لك تسهيلات كثيرة رائعة.

-٣-

الكود الزائف (سودو كود) Pseudo Code

تعريف: الكود الزائف Pseudo Code:
يمكن ترجمة الكلمة "سودو كود" أيضا إلى الكود الوهمي، أو الكود غير الحقيقي، أو الكود التقريبي، أو شِبه الكود.. السبب في هذا هو أنه طريقة كلامية للتعبير عن خطوات حل مسألة معينة، لكن بدون استخدام كود حقيقي من أكواد لغات البرمجة المعروفة.

قواعد الكود الزائف Pseudo Code:

١- اختيار أسماء متغيرات واضحة ليسهل فهم وظيفتها.
٢- كتابة الأوامر بصورة سهلة وواضحة ولا لبس فيها.
٣- تنسيق المقاطع التي لها بداية ونهاية بطريقة واضحة.

مكونات الكود الزائف Pseudo Code:

أسهل طريقة لتعلم هذا الكود التقريبي، هي ترجمة مخططات التنفيذ Flowcharts إلى كلمات تشرحها.. لاحظ أننا سنستخدم هنا اللغة الإنجليزية المبسطة، لأن هذا هو المتعارف عليه، ولأن مفردات لغات البرمجة مكتوبة بالإنجليزية.

وسأريك كيفية التعبير عن كل جزء في المخططات باستخدام الكود الزائف Pseudo Code.

والآن دعنا نتعرف على مكونات الكود الزائف:

١- ترقيم الخطوات:

كل خطوة من خطوات البرنامج المكتوب بالكود الزائف يجب أن يكون لها رقم.. فأول خطوة تأخذ الرقم ١، والثانية تأخذ الرقم ٢، وهكذا...

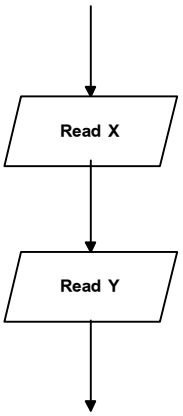
٢- القفز من أية خطوة إلى أخرى:

إذا كان المخطط Flowchart يحتوي على سهم يقفز من رمز إلى رمز آخر ليس تاليا له مباشرة، فيمكنك التعبير عن هذا بالجملة:

Go to step N

حيث N هو رقم الأمر الذي تريد الذهاب إليه.. وسنرى كيف نفعل هذا في الأمثلة القادمة.

٣- قراءة المدخلات:



إذا وجدت في المخطط رمز الإدخال مثل الموضح في الصورة، فيمكنك التعبير عنه باستخدام الكلمة Input أو الكلمة Read كالتالي:

1- Input the value of X
2- Input the value of Y.

أو:

1- Read the value of X
2- Read the value of Y.

٤- طباعة المخرجات:

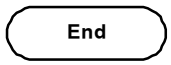
إذا وجدت في المخطط رمز الإخراج، فيمكنك التعبير عنه باستخدام الكلمة Print:

Print "Student Failed"	5- Print an output line saying: "Student Succeeded"
Print "Max. no. is X"	9- Print an output line showing that the maximum number is the value of X.

لاحظ أن استخدام الرقمين ٥ و ٩ هو لمجرد تنبيهك إلى أهمية وجود رقم السطر في البداية، لكنه سيختلف طبعا على حسب موضع أمر الطباعة في البرنامج.
ويمكن أيضا أن تستخدم بعض الجمل البديلة للكلمة Print.. مثلا إذا أردت أن تطبع تقريرا يحتوي على اسم الطالب وعمره ورقم فصله، يمكنك أن تكتب ذلك كالتالي:

12- Write a report for the student containing his Name, Age, and Class

٥- نهاية البرنامج:



استخدم الجملة التالية للتعبير عن رمز نهاية البرنامج في المخطط:

Stop processing

وهي تعني: إيقاف المعالجة أو إيقاف التنفيذ.

٦- المقارنات والشروط:

تستخدم جملة الشرط IF Statement للتعبير عن رمز المقارنة واتخاذ القرار Decision في المخطط.. وتكتب جملة IF بالصيغة التالية:

```
3- IF Condition is TRUE THEN
     Block1
   ELSE
     Block2
   END IF
```

لاحظ ما يلي:

١- جملة IF كلها أخذت ترقيما واحدا، مهما كان الكود المكتوب فيها كبيرا.. لقد أعطيناها هنا الرقم ٣ لضرب المثال.

٢- الشرط Condition هو أية عملية مقارنة تريد إجراءها.. وتستخدم في المقارنة الكلمات التالية:

equals	يساوي
not equals	لا يساوي
greater than	أكبر من
less than	أقل من
greater than or equals	أكبر من أو يساوي
less than or equals	أقل من أو يساوي

٣- المقطع الأول Block1 هو سطر من الكود أو عدة سطور من الكود، لا يتم تنفيذها إلا إذا كان الشرط Condition صحيحا TRUE.

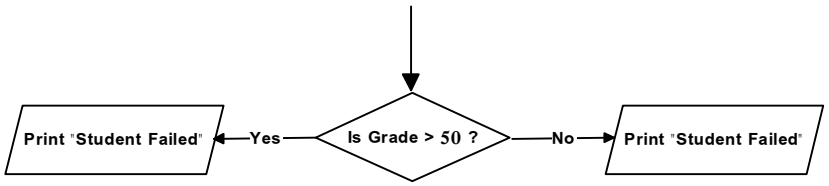
٤- المقطع الثاني Block2 هو سطر من الكود أو عدة سطور من الكود، لا يتم تنفيذها إلا إذا كان الشرط Condition خاطئا FALSE.. هذه هي وظيفة الكلمة "غير ذلك" ELSE، فهي تعني أن الشرط السابق لها ليس صحيحا، وبالتالي سيتم اتخاذ تصرف آخر.

٥- تنتهي جملة الشرط بالتعبير END IF منعا للالتباس، حتى يستطيع من يقرأ الكود الزائف أن يميز نهاية المقطع ELSE عن الكود التالي لها.

٦- يمكن ألا تحتوي جملة الشرط على المقطع ELSE.. في هذه الحالة ستكون لها الصيغة:

```
IF Condition is TRUE THEN
   Block1
END IF
```

لاحظ وجود العبارة END IF أيضا، لتميز نهاية مقطع الشرط عن الكود التالي لها.. مثال:




```
1- IF the value of the Grade is greater than 50 then
      Print an output line saying: "Student Succeeded"
   ELSE
      Print an output line saying: "Student Failed"
   END IF
```

٧- العمليات:

استخدم جملة مختصرة تصف العملية التي تقوم بها.. هذه الجملة يمكن أن تستخدم كلمات مثل:

Initialize Reinitialize	ضع القيمة الابتدائية أعد وضع القيمة الابتدائية
Add	اجمع
Subtract	اطرح
Multiply	اضرب
Divide	اقسم
Increase Increment	أزد القيمة زيادة القيمة
Decrease Decrement	أنقص القيمة إنقاص القيمة
Calculate	احسب قيمة

مثلا:

1- Initialize the number and the sum:

N = 0

Sum = 0

2- Increase the number:

N = N + 1

3- Add the number to the sum:

Sum = Sum + N

4- Calculate the average:

Avg = Sum / N

لاحظ أن المتوسط الحسابي average = مجموع الأعداد ÷ عددها.. طبعا هذا المثال سيعطي المتوسط الحسابي ١ لأننا جمعنا عددا واحدا فقط هو ١ وقسمناه على ١.. وسنرى في المثال التالي كيف نحسب المتوسط الحسابي لعدد كبير من الأعداد.

٨- التكرار Loop:

عرفنا أن التكرار في مخطط التنفيذ يتم برسم سهم من إحدى الخطوات يعود إلى الخلف إلى خطوة سابقة.. ويكتب الكود الزائف Pseudo Code المعبر عن هذا كالتالي:

Perform steps N1 through N2 until Condition is TRUE

هذه الجملة تطلب الاستمرار في تنفيذ الخطوات بدءا من الخطوة رقم N1 وحتى الخطوة رقم N2 إلى أن يتحقق الشرط Condition، وهو أي شرط تريده.

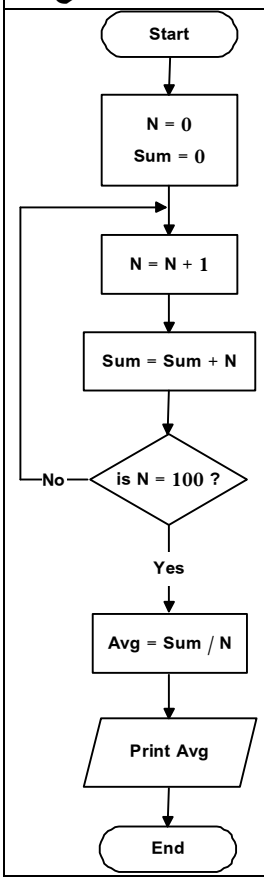
ويمكنك كتابة الجملة السابقة بصورة أخرى مكافئة كالتالي:

Perform steps N1 through N2 While Condition is FALSE

هذه الجملة تطلب الاستمرار في تنفيذ الخطوات بدءا من الخطـوة رقـم N1 وحتـى الخطوة رقم N2، طالما أن الشرط Condition خاطئ ولم يتحقـق.. لاحـظ جيـدا الفارق بين "حتى" Until و "بينما" While، فكل منهما عكس الأخرى في المعنى.

دعنا نأخذ مثالا جيدا، وهو برنامج حساب المتوسط الحسابي للأعـداد مـن ١ إلـى ١٠٠.. هذا هو:

مخطط البرنامج	الكود الزائف Pseudo Code

	1- Initialize the number and the sum:
	N = 0
	Sum = 0
	2- Increase the number:
	N = N + 1
	3- Add the number to the sum:
	Sum = Sum + N
	4- IF the value of N equals 100 THEN
	Go to Step 5
	ELSE
	Perform steps 2 through 3 until the value of N = 100
	END IF
	5- Calculate the average:
	Avg = Sum / N
	6- Print an output line showing that the value of the Avg variable
	7- Stop processing

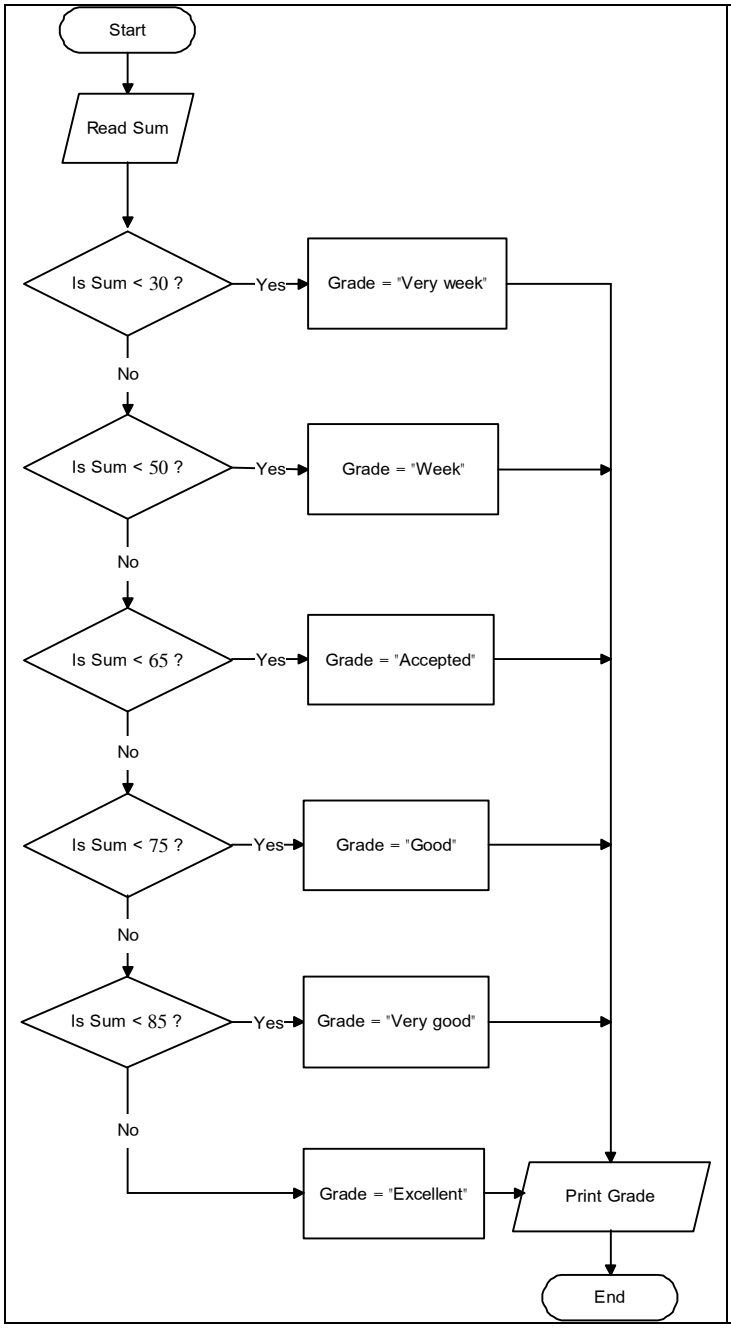
ويمكنك إعادة كتابة جزء التكرار باستخدام While كالتالي:

4- IF the value of N < 100 THEN
Go to Step 5
ELSE
Perform steps 2 through 3 while the value of N < 100
END IF

أمثلة على الكود الزائف Pseudo Code:

سنتناول في هذا المقطع بعض الأمثلة التي شرحناها في الفصل السابق.. سنعيد رسم مخطط التنفيذ لهذه الأمثلة لكن باللغة الإنجليزية هذه المرة، وسنكتب الكود الزائف المناظر لهذه المخططات.

مثال ١:
اكتب الكود الزائف Pseudo Code الذي يعبر عن المخطط الموضح في الصورة، وهو لبرنامج يحسب تقدير الطالب بناء على مجموع درجاته.

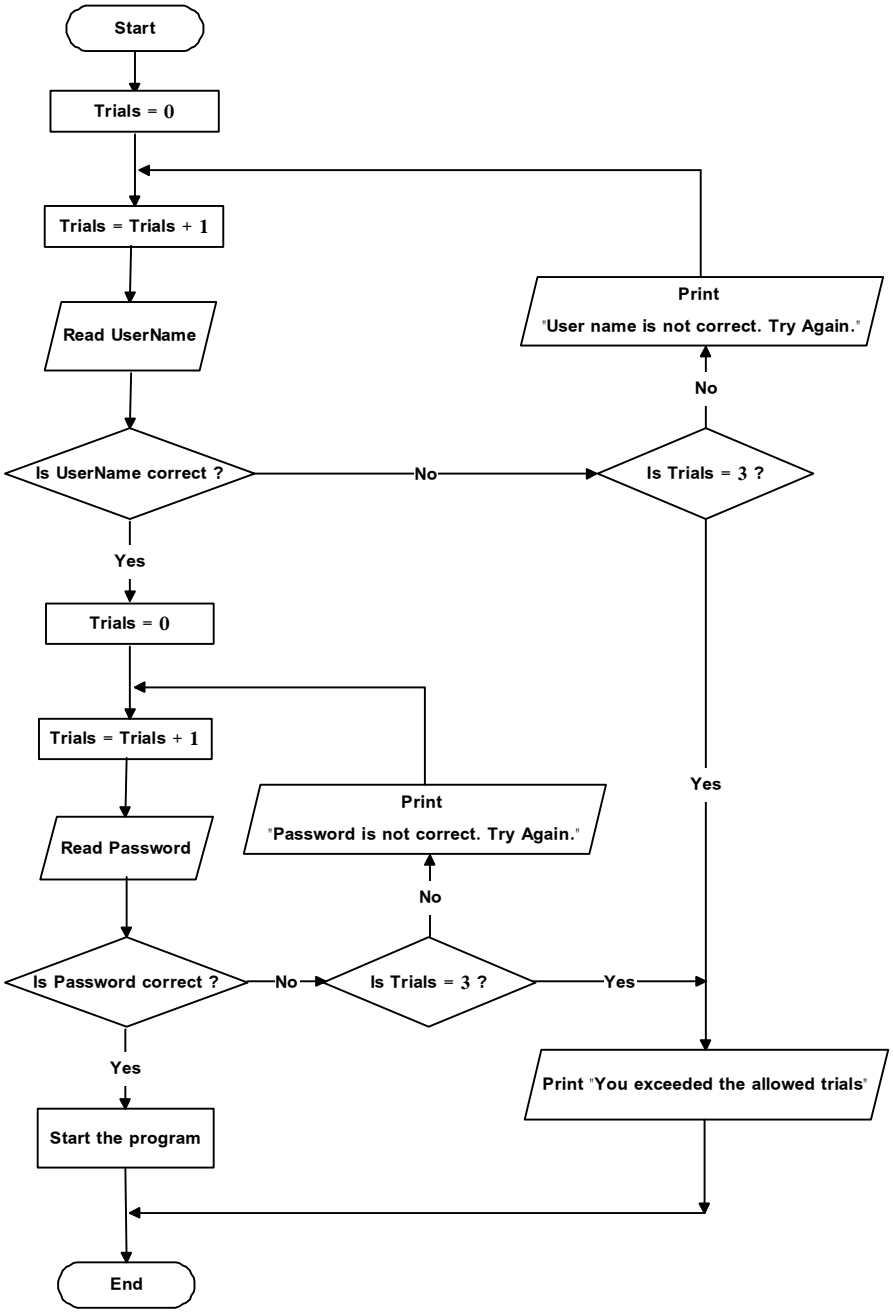


الحل:

```
1-  Input the value of the Sum
2-  IF the value of the Sum < 30 THEN
        Grade = "Very week"
        Go to 7
    END IF
3-  IF the value of the Sum < 50 THEN
        Grade = "week"
        Go to 7
    END IF
4-  IF the value of the Sum < 65 THEN
        Grade = "Accepted"
        Go to 7
    END IF
5-  IF the value of the Sum < 75 THEN
        Grade = "Good"
        Go to 7
    END IF
6-  IF the value of the Sum < 85 THEN
        Grade = "Very Good"
        Go to 7
    ELSE
        Grade = "Excellent"
    END IF
7-  Print an output line showing the Grade
8-  Stop processing
```

مثال ٢:

اكتب الكود الزائف Pseudo Code الذي يعبر عن المخطط الموضح في الصـــورة، وهـــو لبرنامج يفحص اسم المستخدم وكلمة السر.


Start
Trials = 0
Trials = Trials + 1
Read UserName
Is UserName correct ?
No
Is Trials = 3 ?
Print
"User name is not correct. Try Again."
No
Yes
Trials = 0
Trials = Trials + 1
Read Password
Is Password correct ?
No
Is Trials = 3 ?
Yes
Print
"Password is not correct. Try Again."
No
Yes
Yes
Start the program
Print "You exceeded the allowed trials"
End

الحل:

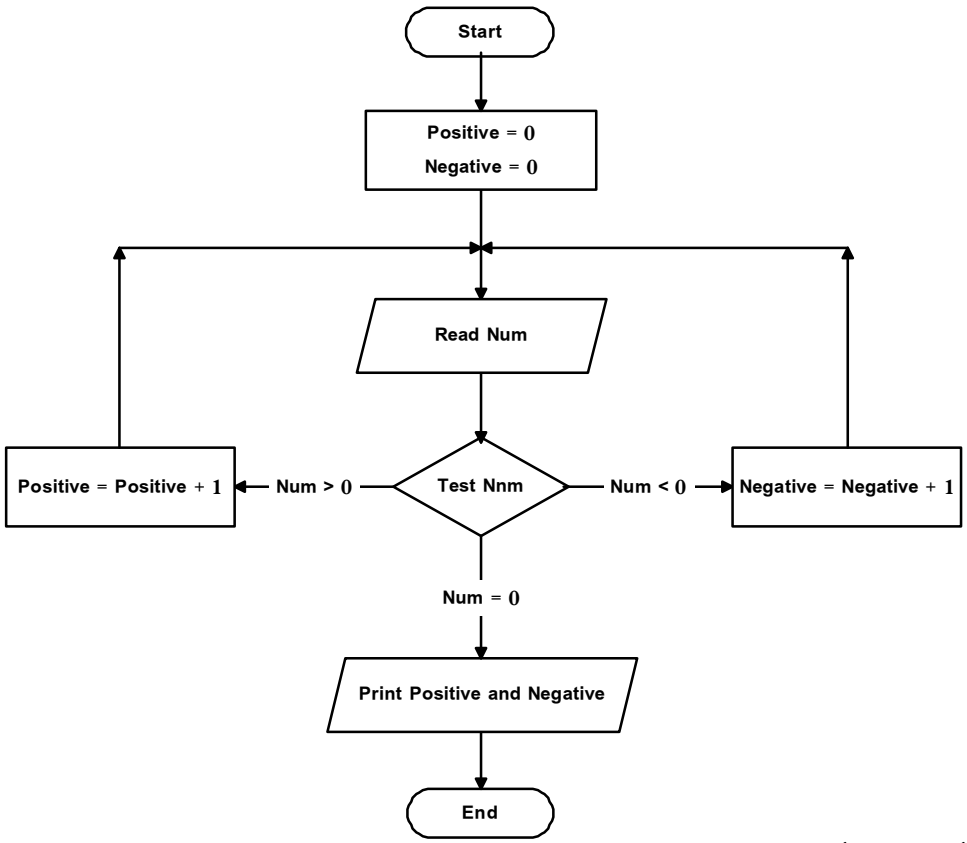
```
1- Initialize the number of user trials:
   Trials = 0
2- Increase number of user trials:
   Trials = Trials + 1
3- Input a string that represents the UserName
4- IF the string of the UserName does not match the correct user
   name THEN
      IF the No. of  Trials = 3 THEN
         Print an output line saying: "You exceeded the allowed
         trials."
         Go to 10
      ELSE
         Print an output line saying: "User name is not correct. Try
         again."
         Perform steps 2 through 3 until UserName matches the
         correct user name
      END IF
   END IF
5- Reinitialize the no of user trials:
   Trials = 0
6- Increase number of user trials:
   Trials = Trials + 1
7- Input a string that represents the Password
8- IF the string of the Password does not match the correct password
   THEN
      IF the no of  Trials = 3 THEN
         Print an output line saying: "You exceeded the allowed
         trials."
         Go to 10
      ELSE
         Print an output line saying: "Password is not correct. Try
         again."
         Perform steps 6 through 7 until Password matches the
         correct password
      END IF
   END IF
9- Start the program
10- Stop processing
```

مثال ٣:
ارسم مخططا يستمر في قراءة مجموعة من الأعداد من المستخدم إلى أن يكتب المستخدم الرقم ٠، ومن ثم يعرض البرنامج للمستخدم عدد الأعداد السالبة وعدد الأعداد الموجبة التي أدخلها.. اكتب الكود الزائف لهذا البرنامج.

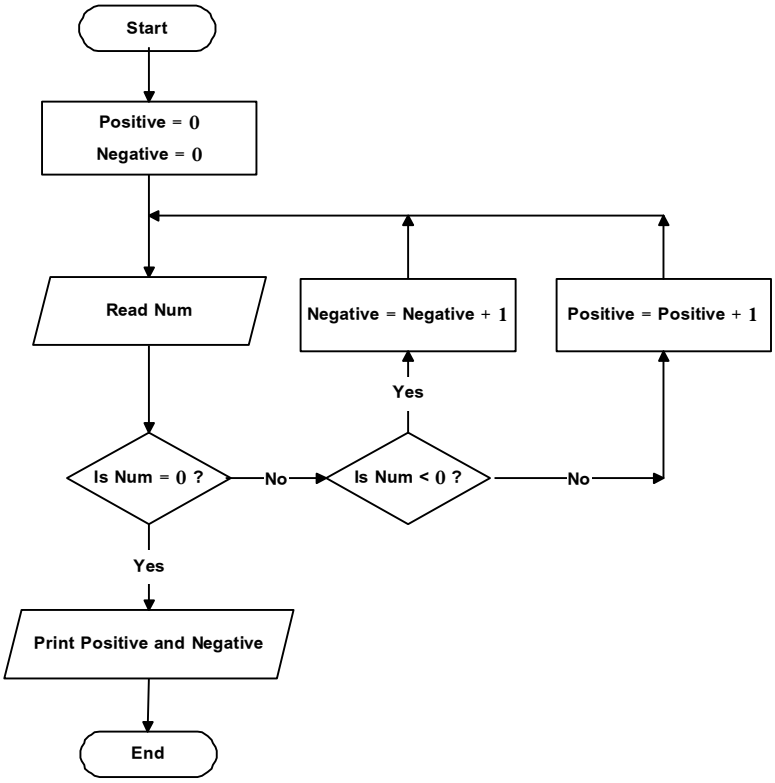
الحل:
سنستخدم في هذا البرنامج ثلاثة متغيرات:

١- Num: سنقرأ فيه العدد الذي كتبه المستخدم.
٢- Positive: سنضع فيه عدد الأعداد الموجبة.. ستبدأ قيمة هذا المتغير بصفر، وسنزيدها بواحد كلما وجدنا أن Num أكبر من الصفر.
٣- Negative: سنضع فيه عدد الأعداد السالبة.. ستبدأ قيمة هذا المتغير بصفر، وسنزيدها بواحد كلما وجدنا أن Num أصغر من الصفر.

هذا هو المخطط:



أو بطريقة أخرى:

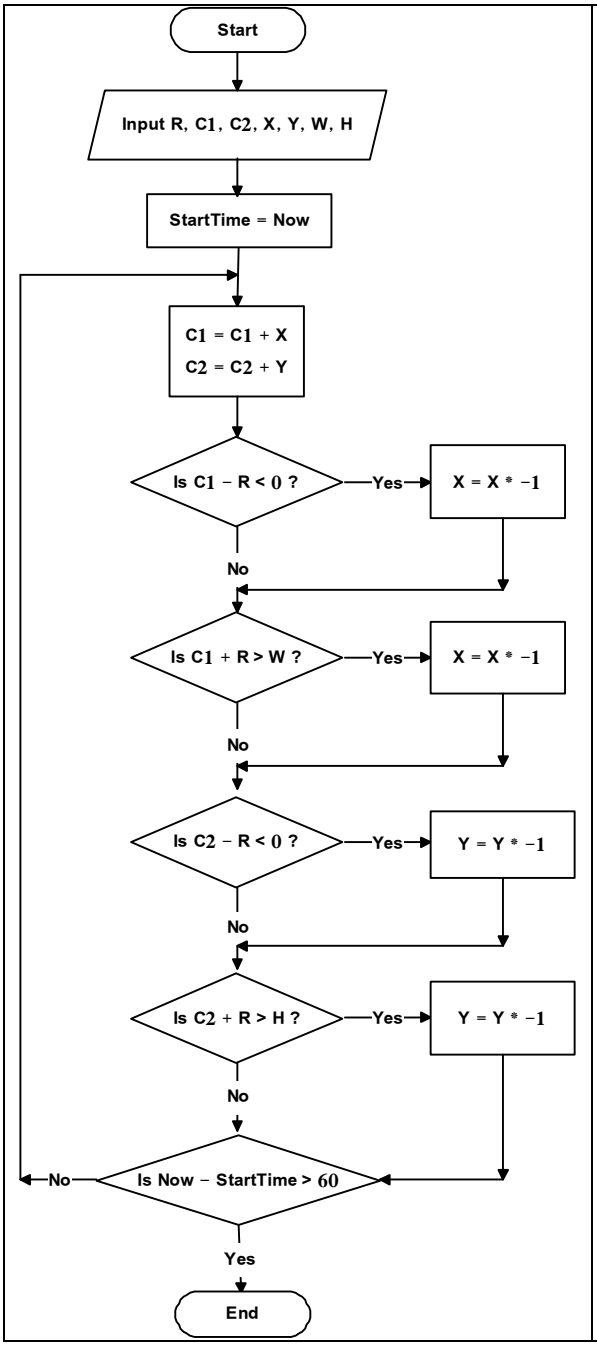


وهذا هو الكود الزائف:

```
1-  Initialize the counters of positive and negative numbers:
    Positive = 0
    Negative = 0
2-  Input Num
3-  IF Num = 0 THEN
        Go to 4
    ELSE
        IF Num < 0 THEN
            Increase the value of the negative numbers counter:
                Negative = Negative + 1
            Perform steps 2 through 3 until Num = 0
        ELSE
            Increase the value of the positive numbers counter:
                Positive = Positive + 1
            Perform steps 2 through 3 until Num = 0
        END IF
    END IF
4-  Print an output line showing the values of Negative and Positive
5-  Stop processing
```

مثال ٤:

اكتب الكود الزائف Pseudo Code الذي يعبر عن المخطط الموضح في الصورة، وهو لبرنامج يحرك كرة على الشاشة لمدة ٦٠ ثانية.



الحل:

```
1- Input the ball radius (R), the initial position of the ball
   center (C1, C2), the step by which that the ball moves (X, Y), and
   the window size (W, H)
2- Initialize the start time to the current time:
   StartTime = Now
3- Increase the horizontal and vertical positions of the center of the
   ball (C1, C2), by adding the horizontal and vertical steps (X , Y):
   C1 = C1 + X
   C2 = C2 + Y
4- IF C1 – R < 0 THEN
       X = X * -1
   END IF
5- IF C1 + R > W THEN
       X = X * -1
   END IF
6- IF C2 – R < 0 THEN
       Y = Y * -1
   END IF
7- IF C2 + R > H THEN
       Y = Y * -1
   END IF
8- IF Now – StartTime > 60 seconds THEN
      Go to 9
   ELSE
       Perform steps 3 through 7 until Now – StartTime > 60
       seconds
   END IF
9- Stop processing
```

مميزات الكود الزائف Pseudo Code:

١- سهل الفهم، لأنه يستخدم كلمات إنجليزية بسيطة.

٢- قريب في صياغته من لغات البرمجة، لهذا يسهل تحويله إلى أية لغة برمجة.

٣- لا يحتوي على أشكال ورسوم خاصة، وليست له قواعد صعبة خاصة بـــه، لهـــذا يكون أسهل وأسرع في كتابته.

٤- يسهل تعديله وتطويره.

ويوجد عيب واحد في الكود الزائف Pseudo Code، هو أنه قد يكون طويلا جـــدا فـــي المسائل المعقدة.

-٤-

مقدمة عن البرمجة الشيئية (الكائنية)

Object Oriented Programming (OOP)

البرمجة الخطية Linear Programming:

- يعتبر هذا النوع من البرمجة بدائيا وتقليديا.
- وكان موجودا في لغات البرمجة الأولى مثل Basic و C و FORTRAN و Pascal، التي كانت تعمل على نظم التشغيل البدائية مثل الدوس DOS.
- هذا النوع من البرامج هو الذي درسناه من خلال المخططات Flowcharts.
- كان على المبرمج أن يكتب البرنامج كله في سطور متتالية، ويسأل المستخدم عن كل خطوة يريد عملها، أو يطلب من المستخدم كتابة معلومة معينة كاسمه أو تاريخ ميلاده، ثم يقرأ هذه المعلومة وعلى أساسها يتخذ باقي قرارات البرنامج، وفي النهاية تتم طباعة النتائج على الشاشة.. هذا هو سبب تسميتها بالبرمجة الخطية، فهي تسير على خط واحد كل خطواته متتالية ومتتابعة.
- وبعض هذه البرامج تستخدم "سطور الأوامر" Command Lines، حيث يستطيع المستخدم كتابة بعض الأوامر لينفذ البرنامج وظيفة معينة.. لهذا يمكن أن نفهم الاسم Linear Programming باعتباره البرمجة السطرية أو البرمجة بالسطور، لأن البرنامج كان ينفذ بسطور الأوامر!.. لو أردت تجربة هذا مثلا، فافتح قائمة البداية Start menu من سطح المكتب، واضغط الأمر Run، واكتب اسم البرنامج Command.. ستظهر لك نافذة سوداء تشبه نافذة الدوس القديمة.. هذا البرنامج يمنحك إمكانيات شبيهة بإمكانيات متصفح الويندوز Explorer، حيث يمكنك تصفح المجلدات والملفات، ونسخها أو نقلها أو حذفها.. لكن كل هذا يتم بسطور الأوامر، والتي يتم تنفيذ كل منها بضغط زر الإدخال Enter.. جرب مثلا أن تكتب cls واضغط زر الإدخال.. هذا الأمر سيمحو كل الكلمات المكتوبة على الشاشة.. جرب أيضا أن تكتب Dir وتضغط زر الإدخال.. سيعرض هذا على الشاشة كل محتويات المجلد الحالي.. وهكذا...

- تسمى هذه البرمجة أيضا بالبرمجة الإجرائية Procedural Programming، فهي تستخدم الإجراءات وأوامر البرمجة لتنفيذ مجموعة من الوظائف.

عيوب البرمجة الخطية:

١- أطول كودا.
٢- أعقد في برمجتها.
٣- أصعب في تصحيح أخطائها.
٤- إعادة استخدام أجزاء من الكود تعطي أقل فائدة.

البرمجة الخطية Linear Programming:
تستخدم الأسلوب التقليدي للبرمجة، حيث يعتمد البرنامج على الترتيب الدقيق للأوامر والعمليات، ويحدد المبرمج للمستخدم الخطوات الإجبارية التي يجب عليه اتباعها أثناء تشغيل البرنامج، ويستجيب لكل منها في خطوات متتابعة.

البرمجة الشيئية OOP:

- يترجمها البعض أيضا بالبرمجة الموجهة بالكائنات، أو اختصارا: البرمجة بالكائنات.
- تعتمد عليها لغات البرمجة الحديثة، مثل VB.NET و C# و VC++ التي تعمل على نظم التشغيل الحديثة كالويندوز.
- تتسم بأنها أكثر تنظيما من البرمجة الخطية، فالأكواد التي تؤدي وظائف ذات صلة يتم جمعها في قالب واحد يسمى التصنيف أو الفئة Class.
- يمكن تعريف عدد لا نهائي من المتغيرات من نفس الفئة، يسمى كل منها بالكائن Object.

- لكل كائن مجموعة من الخصائص Properties، والوسائل Methods، والأحداث Events، ومن خلالها يتحكم المبرمج في وظيفة الكائن، وهذا بدلا من استخدام الإجراءات وأوامر الكود المنفصلة في البرمجة الخطية.
- استخدام الكائنات يوفر الوقت والجهد ويضمن أقصى استفادة من الكود.
- توفر بعض الفئات أدوات مرئية يتحكم فيها مشغل البرنامج باستخدام لوحة المفاتيح والفأرة، مما يجعل للبرنامج شكلا جذابا ويتيح للمستخدم حرية تنفيذ الجزء الذي يريده من البرنامج في الوقت الذي يريده.

البرمجة الشيئية OOP:
تستخدم مجموعة من الكائنات Objects لأداء وظائف البرنامج المختلفة.. ولا يتبع تنفيذ البرنامج التسلسل المنطقي، حيث يتم تقسيم البرنامج إلى أجزاء صغيرة، يتم تنفيذ كل منها كاستجابة لحدث معين من الأحداث التي يقوم بها المستخدم.

البرمجة الموجهة بالأحداث Event Driven Programming:

- تقترن هذه التقنية بالبرمجة الموجهة بالكائنات OOP التي تمنح واجهة مرئية للبرنامج.
- تعتمد هذه التقنية على قدرة الويندوز على تشغيل أكثر من برنامج في نفس الوقت، وتشغيل أكثر من شاشة في البرنامج الواحد، حيث يتنقل المستخدم بين البرامج والشاشات باستخدام الفأرة ولوحة المفاتيح.
- يستطيع المستخدم أن ينفذ ما يريده من وظائف البرنامج من خلال ما تعرضه كل شاشة من أزرار واختيارات.. خذ برنامج الوورد Word كمثال.. في هذا البرنامج يستطيع المستخدم أن يفتح ملفا جديدا، ويكتب فيه ما يشاء من النصوص ويعدلها ويحذفها، وينسخ أي جزء منها ويلصقه، ويضيف صورة أو يرسم مخططا، ويحفظ الملف أو يلغيه.. إلخ.. كل هذه الأفعال لا يتدخل فيها المبرمج ولا يجبر المستخدم عليها، فالمستخدم هنا هو سيد قراره ويمتلك الحرية الكاملة في التعامل مع البرنامج.
- لكي يحدث كل هذا لا يكتب المبرمج كل الكود كبرنامج واحد مترابط يجب تنفيذه على التوالي، بل يفعل ما يلي:

١- يقسم المبرمج برنامجه إلى نوافذ.

٢- يضع المبرمج على كل نافذة مجموعة من الأدوات، كالأزرار ومربعات النصوص وغيرها.

٣- يكتب المبرمج في كل نافذة أجزاء صغيرة من الكود يعتبر كل منها برنامجا مستقلا بذاته، ينفذ كل منها إحدى وظائف النافذة أو الأدوات الموجودة عليها.

٤- لا يتم تنفيذ أي جزء من الكود إلا عندما يقوم المستخدم بحدث Event معين باستخدام لوحة المفاتيح أو الفأرة، لتنفيذ إحدى وظائف النافذة أو أية أداة موجودة عليها.. فلو ضغط المستخدم زر "فتح ملف"، يتم تنفيذ الكود الذي يفتح نافذة جديدة

في البرنامج، ولو ضغط زر الحفظ، يتم تنفيذ الكود الذي يحفظ النص الذي كتبه المستخدم في ملف.. وهكذا.

البرمجة الموجهة بالأحداث Event Driven Programming:
في هذا النوع من البرمجة، لا يتم تنفيذ أي جزء من البرنامج إلا كاستجابة لحدث معين قام به المستخدم باستخدام لوحة المفاتيح أو الفأرة، لهذا يقسم المبرمج برنامجه إلى أجزاء صغيرة تسمى معالجات الحدث أو المستجيبات للحدث Event Handlers.. هذا يجعل كتابة الكود أسهل وأبسط، كما يمنح مستخدم البرنامج حرية وسهولة وقدرات أكبر.

وسنأخذ في هذا الفصل فكرة عامة عن البرمجة الكائنية، وسنتعرف على الكائنات والفئات والخصائص والوسائل والأحداث، لأن اللغات الحديثة ومنها فيجيوال بيزيك دوت نت تعتمد عليها بشكل تام، وهو ما سنتعلمه بتفصيل أكبر في باقي فصول هذا الكتاب.

الكائن Object:

- الكائن هو شيء له وجود وكيان، مثل قلمك وحاسبك، والعصفور الذي تربيه، وصديقك أحمد.. لاحظ أن القلم والحاسب والعصفور والإنسان هي أسماء عامة تدل على نوع الكائن وليست كائنات في حد ذاتها.. لهذا تسمى هذه الأنواع فئات Classes كما سنعرف بعد قليل.
- يمكن أن يكون الكائن جزءا من كائن آخر.. فلوحة المفاتيح مثلا هي كائن، وهي جزء من كائن أكبر يسمى الحاسوب.
- هذا يعني أن الكائن الواحد قد يتكون من عدة كائنات أصغر، فالحاسوب مثلا يحوي على لوحة مفاتيح وفأرة وشاشة وقرص صلب... إلخ.
- وفي لغات البرمجة الشيئية هناك آلاف الكائنات.. فالنافذة Window التي تراها على الشاشة هي كائن، وكل زر أو مربع نص تراه عليها هو كائن.. إلخ.
- يتم الحوار بين المبرمج والكائن باستخدام الخصائص والوسائل والأحداث التي يمتلكها الكائن.

الكائن Object:
هو شيء له وجود، ويمتلك مجموعة من الخصائص Properties، والوسائل Methods، والأحداث Events، ويمكن أن يتكون من مجموعة من الكائنات الأصغر، أو يكون جزءا من كائن أكبر.

الخصائص Properties:

- الخاصية هي صفة من صفات الكائن.
- وللخاصية اسم وقيمة.. يشرح الاسم وظيفة الخاصية، بينما تحدد القيمة درجة الصفة التي تمنحها هذه الخاصية للكائن.
- فإذا أخذناك أنت كمثال باعتبارك كائنا، فأنت تمتلك هذه الصفات: الاسم، العمر، الطول، الوزن.. إلخ، حيث يوضح اسم كل خاصية من هذه الخصائص معناها.. فإذا أراد شخص ما أن يصفك فسيقول إن اسمك هو فلان، وعمرك هو ١٤ عاما، وطولك هو ١٣٠ سم ووزنك هو ٧٠ كجم.. إلخ.
- كل الكائنات من نفس النوع تملك نفس أسماء الخصائص، لكنها قد تختلف في قيمها.. فكل صديق من أصدقائك يمتلك خصائص الاسم والعمر والطول والوزن، لكنّ كلا منهم له اسم مختلف وطول مختلف ووزن مختلف وهكذا.
- وفي البرمجة، يمتلك كائن النافذة مثلا عدة خصائص مثل: الاسم Name والنص Text والعرض Width والارتفاع Height ولون الخلفية BackColor.. إلخ.. وباختلاف قيم هذه الخصائص تختلف مواصفات كل نافذة تراها على شاشة جهازك في البرامج المختلفة، رغم أنها جميعا تستخدم نوافذ من نوع واحد اسمه النموذج Form.
- ولاستخدام الخاصية في البرمجة، تستخدم الصيغة التالية:

ObjectName.PropertyName = PropertyValue

حيث إن:

- ObjectName: هو اسم الكائن.
- PropertyName: هو اسم الخاصية.
- PropertyValue: هي قيمة الخاصية.

فمثلا: لو لدينا كائن يمثل أحد الطلاب، وكان اسم هذا الكائن Sudent1، فيمكن تغيير خصائص الاسم والعمر والطول لهذا الطالب كالتالي:

```
Sudent1.Name = "Ahmad"
Sudent1.Age = 14
Sudent1.Length = 135
```

ولو لدينا كائن نافذة اسمه Form1، فيمكننا تغيير خصائص النص والعرض والارتفاع الخاصة به كالتالي:

```
Form1.Text = "Open File Window"
Form1.Width = 700
Form1.Height = 500
```

الخاصية Property:
صفة من صفات الكائن تحدد شكله وسمته.. ولكل كائن مجموعة من الخصائص.

الوسائل Methods:

- الوسيلة هي وظيفة يمكن أن يقوم بها الكائن.
- فكر في الوسيلة باعتبارها فعل أمر توجهه إلى الكائن لتنفيذ وظيفة معينة.. فلو كنت تقود سيارة مثلا، فإنك تأمرها بأن تعمل عن طريق إدارة مفتاحها.. لهذا يقال إن إدارة المفتاح هو أمر للسيارة لكي تعمل، أو بطريقة أخرى: هو **وسيلة** لتشغيل السيارة.. كذلك فإن ضغط دواسة الوقود هو وسيلة لجعل السيارة تتحرك، بينما ضغط دواسة الفرامل هي وسيلة لإيقاف السيارة.. لهذا لو كتبت كودا لتمثيل السيارة في لعبة سباق السيارات مثلا، فإنك ستجعل لكائن السيارة الوسائل التالية:

Start	لتشغيل السيارة
Move	لتحريك السيارة
Stop	لإيقاف السيارة

- وفي البرمجة، يمتلك كل كائن عدة وسائل، يستطيع المبرمج استخدمها لجعل الكائن ينفذ وظائف معينة.. فالنافذة مثلا تمتلك الوسائل التالية:

Show	لعرض النافذة
Hide	لإخفاء النافذة لكن دون إغلاقها.
Close	لإغلاق النافذة نهائيا.

- ويمكنك توجيه الأوامر إلى الكائن (أو بمعنى آخر: استخدام وسائل الكائن)، باستخدام الصيغة التالية:

ObjectName.MethodName( )

فمثلا: لو لديك نافذة اسمها Form1، فإنك تستطيع عرضها وإغلاقها باستخدام الوسيلتين التاليتين:

Form1.Show( )
Form1.Close( )

لاحظ وجود قوسين بعد اسم الوسيلة.. بعض الوسائل لا تستخدم هذين القوسين لهذا يتم تركهما فارغين.. لكن هناك وسائل أخرى تستخدم هذين القوسين لاستقبال بعض المعلومات التي تؤثر على طريقة عمل الوسيلة.. هذه المعلومات تسمى المعاملات Parameters، وهي تتحكم في طريقة تنفيذ الوظيفة التي تقوم بها الوسيلة.. فمثلا، لو كان لديك كائن سيارة اسمه MyCar، وكانت لهذا الكائن وسيلة اسمها Move، فإن هذه الوسيلة تحتاج إلى معلومتين من قائد السيارة: سرعة الحركة واتجاه الحركة، وهما المعاملان المطلوبان لهذه الوسيلة.. في هذه الحالة يمكنك استخدام هذه الوسيلة برمجيا لتحريك السيارة بسرعة ٥٠ كم إلى الأمام كالتالي:

MyCar.Move(50, Forward)

وسنتعرف على المعاملات في فصل لاحق بإذن الله.

الوسيلة Method:
هي سلوك معين أو وظيفة معينة يقوم بها الكائن.

الأحداث Events:

رن جرس الهاتف فجأة وأنت تجلس في الصالة، وكان صوت الرنين مرتفعا.. إذا اعتبرنا الهاتف كائنا Object، فإن:

- صوت الرنين يسمى خاصية Property، وقيمتها = مرتفعة.
- أما انطلاق الرنين فيسمى حدثا Event، وهدفه تنبيهك إلى وجود مكالمة هاتفية لك.
- في هذه الحالة ستتجه إلى الهاتف وترفع السماعة وتجيب.. يسمى رفع السماعة وسيلة Method ووظيفتها أن تفتح الخط لتبدأ المكالمة.
- وعند انتهاء المكالمة ستقوم بوضع السماعة.. هذه أيضا وسيلة من وسائل الهاتف، ووظيفتها إغلاق الخط.
- يسمى ردك على الهاتف "استجابة أو معالجة للحدث" Handling the Event.. لاحظ أنك قد تقرر عدم الرد على الهاتف، أو قد تكون نائما ولا تسمع رنينه فلا ترد.. في هذه الحالة يقال إنك لم تقم بالاستجابة للحدث، لهذا لم يترتب على الحدث أي تصرف خاص.

- ما ينطبق على الهاتف يمكن تعميمه على الكائنات البرمجية.. فعندما يريد المبرمج شيئا من الكائن، فإنه يخاطبه باستخدام الخصائص والوسائل.. وعندما يريد الكائن أن يخبر المبرمج شيئا فإنه يخاطبه باستخدام الأحداث.
- الحدث هو طريقة لتنبيه المبرمج إلى أن شيئا قد تغير في داخل الكائن، وفي الغالب يكون هذا بسبب فعل معين قام به المستخدم، سواء باستخدام لوحة المفاتيح (مثل ضغط حرف معين)، أو باستخدام الفأرة (مثل التحرك بها أو ضغط أحد أزرارها).
- هناك أحداث كثيرة للكائن، فالنافذة مثلا تمتلك حدث ضغط الفأرة Click وحدث ضغط زر لوحة المفاتيح KeyPress.. لكن المبرمج لا يحتاج للاستجابة إلى كل هذه الأحداث في كل برنامج يكتبه، بل يستجيب فقط للأحداث التي تناسب وظائف برنامجه.
- يستجيب المبرمج للحدث بكتابة بعض الأوامر الخاصة بهذا الحدث.. هذا الجزء يسمى المستجيب للحدث أو معالج الحدث Event Handler.. وفي الغالب يكتب المبرمج في معالج الحدث أكوادا تستخدم خصائص ووسائل الكائن الذي أطلق الحدث، أو خصائص ووسائل كائن آخر مرتبط به.
- فمثلا: إذا أردت كتابة برنامج للرسم بالفأرة فوق النافذة، فإنك تستجيب لحدث حركة الفأرة MouseMove، وتستخدم خصائص ووسائل النافذة للرسم فوقها.

الحدث Event:
الحدث هو فعل يقع على الكائن ويستجيب له. أو بعبارة أخرى: الحدث هو تنبيه يطلقه الكائن لإخبار المبرمج بأن فعلا معينا قد وقع.

الفئة أو التصنيف Class:

- يرسم المهندس المعماري مخططا معماريا للمبنى قبل إنشائه، ثم يسلم هذا المخطط للمهندس المدني لإنشاء أي عدد من الوحدات من هذا المبنى، كلها بنفس التصميم، لكن قد يختلف أحد المباني عن الآخر في بعض خصائصه، كلون جدرانه مثلا.
- يسمى المخطط المعماري للمبنى في هذه الحالة بالفئة Class، بينما يسمى كل مبنى بالكائن Object.. هذا يعني أن كل فئة يمكن إنشاء كائن أو أكثر منها، تختلف فيما بينها فقط في قيم بعض الخصائص.
- وفي البرمجة، تكون الفئة Class هي الكود الذي يكتبه المبرمج لأداء وظائف معينة، وهو يكتبه في صورة خصائص ووسائل وأحداث.. وبعد أن ينتهي المبرمج من كتابة الفئة، فإنه يستطيع تعريف متغير من هذه الفئة.. يسمى هذا المتغير بالكائن Object، أو النسخة الفورية Instance من الفئة، ومن خلاله يستطيع تنفيذ الكود المكتوب داخل الفئة، وذلك باستخدام الخصائص والوسائل والأحداث.
- مثال هذا النوافذ التي تراها في كل برامج الويندوز.. كل هذه النوافذ هي كائنات Objects تم تعريفها من فئة تسمى فئة النموذج Form Class.
- لاحظ أنك تستطيع أن تسكن في المبنى لكنك لا تستطيع السكن في المخطط الهندسي له.. بالمثل يمكنك تكبير حجم أية نافذة، لكن لا يمكنك تكبير حجم فئة النموذج!.. فالفئة هي كود خام، بينما الكائن هو المنتج الذي تم تصنيعه بتطبيق هذا الكود.. أو بعبارة أخرى: الكائن هو التطبيق العملي لفكرة الفئة.

الفئة أو التصنيف Class:
هي مقطع من الكود يُستخدم لتعريف نوع جديد من أنواع الكائنات مستقل بذاته، يحتوي على خصائص ووسائل وأحداث.. وبعد تعريف نوع الكائن يمكن تعريف أي عدد من الكائنات منه.

الكائن Object:
هو متغير Variable تم تعريفه من الفئة. أو هو نسخة Instance تم إنشاؤها من الفئة. ويشغل الكائن مساحة من ذاكرة الحاسب، ومن خلاله يمكن تنفيذ الكود المكتوب في الفئة، واستخدام الخصائص والوسائل والأحداث التي تم تعريفها داخل الفئة. وإذا شبهنا الفئة بمخطط التصميم، فإن الكائن هو المنتج الذي تم تنفيذه من هذا المخطط.

التغليف Encapsulation:

كلنا نستخدم التلفاز والهاتف والسيارة.. لكن لا أحد فينا يفحص تركيبها الداخلي قبل استخدامها، فكل ما يعنينا هو استخدامها لأداء الوظيفة المطلوبة دون الخوض في تفاصيل لا تهمنا.

يسمى هذا بالتغليف Encapsulation، بمعنى وضع كل التفاصيل الداخلية للكائن في غلاف يخفيه عمن يستخدمه.. هذا يضمن للمستخدم سهولة الاستخدام، ويضمن للمصمم حماية التفاصيل الداخلية لمنتجه.

وفي البرمجة استخدم ملايين المبرمجين فئة النموذج Form Class لتصميم النوافذ في برامجهم، لكن أحدا من هؤلاء المبرمجين لم يرَ الكود المكتوب داخل فئة النموذج!.. لقد قام كل مبرمج بتعريف كائنات من فئة النموذج واستخدم خصائصها ووسائلها وأحداثها في تنفيذ ما يريده من وظائف برنامجه، دون أن يشغل ذهنه بالكود المختفي داخل هذه الفئة.

لاحظ أن هناك تفاصيل داخل كل كائن لا يمكن لنا جميعا أن نصل إليها.. فأنت مثلا تستطيع إدارة عجلة المذياع الخارجية لتغيير المحطة، لكنك لا تستطيع تغيير قيم المقاومات الموجودة داخل المذياع، إلا إذا ذهبت إلى أحد الفنيين لضبطها لك.. في هذه الحالة يقال إن لديك صلاحية استخدام عجلة المذياع الخارجية، لكن ليس لديك صلاحية استخدام المقاومات الداخلية.. لهذا يقال إن لعجلة المذياع صلاحية عامة Public، بينما لكل مقاومة داخلية صلاحية خاصة Private.

وفي البرمجة أيضا، توجد داخل الكائن خصائص ووسائل بصلاحيات مختلفة، بعضها عام Public يمكنك استخدامه، وبعضها خاص Private لا يمكنك استخدامه، وهو يستخدم فقط في الكود الداخلي للفئة، المكتوب بواسطة المبرمج المختص الذي صمم الفئة.. يعرف هذا باسم إخفاء البيانات Data Hiding، فمبرمج الفئة يريد الاحتفاظ ببعض التفاصيل الخاصة به مختفية عنا، ليحمي أفكاره من السرقة.

التغليف Encapsulation:
هو إخفاء البيانات داخل الكائن، بحيث لا يتم الوصول إليها إلا بصلاحيات معينة.

الوراثة Inheritance:

لا ريب أنك لعبت كرة القدم على الحاسب أو أي مشغل ألعاب فيديو من قبل.. في هذه اللعبة هناك أربعة أنواع من الأشخاص:

١- اللاعب.
٢- حارس المرمى.
٣- الحكم.
٤- مراقب الخطوط.

دعنا الآن نترك موضعنا كمستخدمين يستمتعون بلعبة كرة القدم، ونشرع في التفكير كمبرمجين، ونسأل أنفسنا هذا السؤال:

هل سنقوم بكتابة أربعة فئات مختلفة لبرمجة هذه الأنواع الأربعة من الأشخاص؟

إن كل هؤلاء يشتركون في النهاية في أن لهم هيئة البشر، ويحركون أيديهم وأرجلهم، ويجرون... إلخ.. لهذا سيكون تكرار كتابة كل هذه الوظائف في كل فئة إهدارا للوقت والجهد، كما أنه سيجعل عملية تصحيح الأخطاء وتطوير الكود أعقد، بسبب اضطرار المبرمج إلى إجراء نفس التعديل في أربع فئات مختلفة!

إذن ماذا نفعل؟

لحل هذه المشكلة، دخل مفهوم الوراثة Inheritance إلى البرمجة.. بهذه الطريقة يمكنك كتابة فئة عامة تحتوي على جميع الخصائص والوسائل والأحداث المشتركة بين اللاعب وحارس المرمى والحكم ومراقب الخطوط.. هذه الفئة تسمى:

- Parent Class: بمعنى الفئة الأم (أو التصنيف الأب).
- أو Base Class: بمعنى الفئة الأساسية أو الفئة الرئيسية.

افترض مثلا أن اسم هذه الفئة Human Class بمعنى فئة البشر، لأنها تجمع كل السمات البشرية التي يتسم بها أشخاص اللعبة.. بعد ذلك يمكنك إنشاء الفئات التالية:

١- فئة اللاعب Player Class.
٢- فئة الحارس Keeper Class.
٣- فئة الحكم Referee Class.
٤- فئة مراقب الخطوط Linesman Class.

كل فئة من هذه الفئات سترث الفئة الأم Human Class.. هذا يعني أنها ستحتوي على كل خصائص ووسائل وأحداث هذه الفئة الأم مباشرة بدون إعادة كتابتها من جديد.. في هذه الحالة كل فئة من هذه الفئات تسمى:

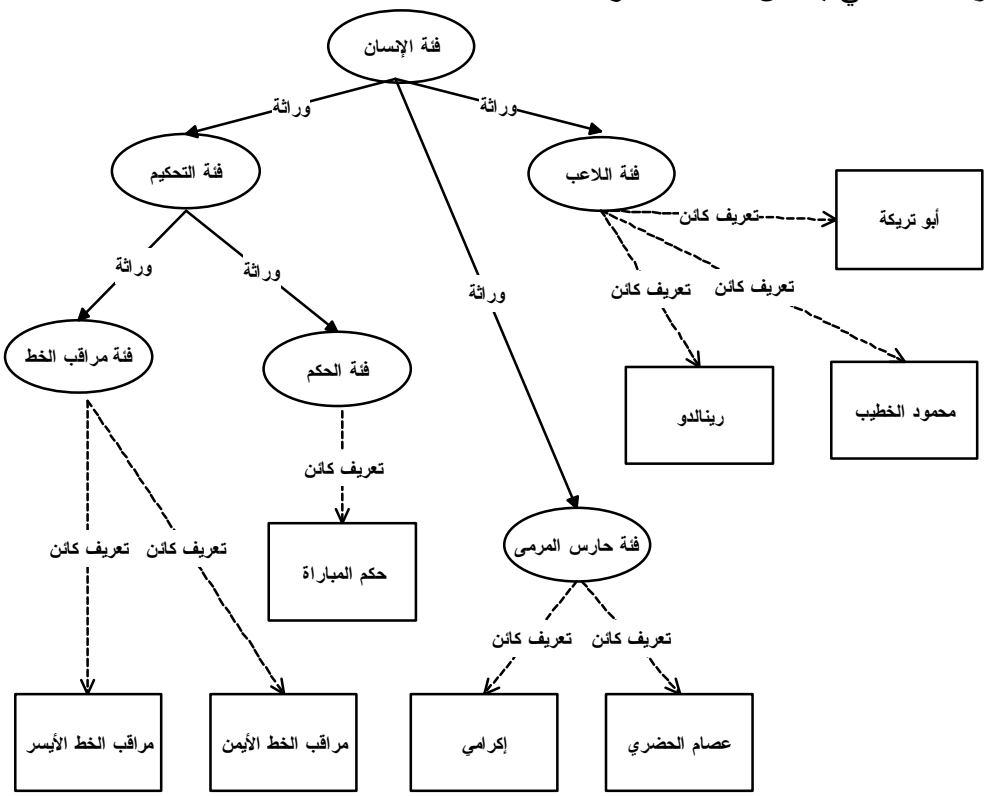
- Child Class: بمعنى الفئة الابنة (أو التصنيف الابن).
- أو Derived Class: بمعنى الفئة الفرعية أو الفئة المشتقة.

لاحظ أن المبرمج يستطيع إضافة أي عدد من الخصائص والوسائل والأحداث الجديدة إلى كل فئة من الفئات المشتقة.. هذه العناصر الجديدة لن تؤثر على الفئة الأم، لكنها ستكون خاصة فقط بالفئة المشتقة.

الجميل في الأمر أن المبرمج يستطيع إنشاء فئة جديدة ترث فئة فرعية.. هذا يعني أنه سيوجد أكثر من فئة أم لهذه الفئة الجديدة.. مثال هذا هو فئة الحكم، فهي تشترك مع فئة مراقب الخطوط في معظم صفاتها، مع وجود اختلافات بينهما.. لهذا يمكن إنشاء فئة ترث فئة الإنسان وتضيف إليها الصفات المشتركة بين الحكم ومراقب الخطوط، وليكن اسمها فئة التحكيم، ومنها ترث كل من فئة الحكم ومراقب الخطوط العناصر المشتركة بينهما.. لاحظ أن هذا يعني أن كلا من فئة الحكم ومراقب الخطوط سترث ضمنيا فئة الإنسان Human Class، وذلك من خلال فئة التحكيم.. بهذا التنظيم الدقيق يستطيع المبرمج توفير وقته وجهده إلى أقصى حد، فهو ليس مضطرا إلى إعادة كتابة أي الكود أكثر من مرة.

لاحظ أن الفروق بين اللاعبين هي مجرد فروق في قيم الخصائص، مثل اسم اللاعب، واسم فريقه، وبلده، وسنه، ولون قميصه ورقمه، وصورة وجهه، وسرعة الجري، وموضعه في الملعب... إلخ.. لهذا فإنّ كل اللاعبين هم في النهاية مجرد كائنات تم تعريفها من فئة اللاعب Player Class.. هذا يعني أنك تستطيع تعريف ٢٠ كائنا للاعبين الأساسيين وما شئت من اللاعبين الاحتياط من الفريقين.. كما تستطيع استخدام فئة حارس المرمى لتعريف حراس المرمى الأساسيين والاحتياط.

والشكل التالي يلخص لك هذا الأمر:



دعنا نأخذ مثالا في فيجيوال بيزيك.. سنتعرف في الفصل السادس من هذا الكتاب على بعض الأدوات المستخدمة في البرمجة، مثل فئة النموذج Form Class وفئة الزر Button Class، وفئة اللافتة Label Class، وفئة مربع النص TextBox Class، وسنتعرف على بعض الخصائص والوسائل والأحداث المشتركة بين هذه الفئات.. السبب في وجود هذه العناصر المشتركة، هو أن جميع هذه الأدوات ترث فئة تسمى فئة أداة التحكم Control Class، سواء بطريقة مباشرة أو غير مباشرة.. لهذا تعتبر فئة الأداة Control Class هي الفئة الأم لكل الأدوات المستخدمة في تصميم نوافذ الويندوز.

الوراثة Inheritance:

يقصد بها امتلاك فئة مشتقة Derived Class كل خصائص ووسائل وأحداث الفئة الأم Base Class، بالإضافة إلى امتلاك الفئة المشتقة بعض الخصائص والوسائل والأحداث الإضافية الخاصة بها.

وفائدة الوراثة هي إعادة استخدام ما تم تصميمه من فئات، والتعديل فيها حسب الحاجة، بدلا من إعادة كتابة كل الكود من جديد.. لاحظ أن هذه هي أهم السمات التي تميز البرمجة بالكائنات عن البرمجة الخطية التقليدية.

-٥-

بيئة التطوير المتكاملة IDE

فيجيوال ستديو دونت نت VS.NET:

فيجوال بيزيك هي إحدى مكونات حزمة برمجية تسمى فيجيوال ستديو دوت نت Visual Studio .NET.. هذه الحزمة تتكون من:

١- **إطار العمل NET Framewor. :**

وهو قلب دوت نت، وهو المسئول عن أداء جميع الوظائف البرمجية، ويحتوي على مكتبة هائلة من الفئات التي يستخدمها المبرمج في كتابة برامجه.

٢- **بيئة التطوير المتكاملة IDE:**

تمثل بيئة التطوير المتكاملة Integrated Development Environment (IDE) الواجهة المرئيّة المشتركة التي تستخدمها لغات دوت نت.. بعبارة أخرى: بيئة التطوير هي البرنامج الذي سنستخدمه للتعامل مع مشاريع فيجيوال بيزيك دوت نت.

٣- **لغات البرمجة الأساسية:**

تحتوي حزمة فيجيوال ستديو دوت نت ٢٠٠٥ على أربع لغات أساسية، وهي:

أ- فيجيوال بيزيك دوت نت Visual Basic .NET (وتكتب اختصارا VB.NET).

ب- فيجيوال سي بلاس بلاس Visual C++.

ج- سي شارب CSharp (وتكتب اختصارا C#).

د- جا شارب JSharp (وتكتب اختصارا J#).. وقد اختفت وقد اختفت لغة J# من إصدار دوت نت ٢٠٠٨ وما تلاه من إصدارات.

ويمكنك أثناء إعداد البرنامج اختيار إعداد أي من هذه اللغات على جهازك أو إزالتها.

بيئة التطوير المتكاملة IDE:

صُمّمت بيئة التطوير المتكاملة بحيث لا تكون مرتبطة بلغةِ برمجةٍ بعينِها، لهذا فإن نفس الأدواتِ التي ستستخدمُها لتطوير تطبيقات فيجيوال بيزيك، ستجدُ أنَّ مبرمجي باقي لغات دوت نت يستخدمونَها بنفس الكيفية!

إنّ جعلَ بيئةِ التطويرِ مستقلة عن لغاتِ البرمجة يسهل على المبرمجينَ تعلم أية لغة من لغات دوت نت، حيث لن يضطرّوا إلى تعلم البيئة الخاصة بكل لغة جديدة يُقدمون على تعلمها، مما يوفر عليهم بالتأكيد وقت التعلم.. لهذا عليك أن تعلم جيدا أن كل ما ستتعلمه

في هذا الكتاب عن كيفية إنشاء المشاريع وكيفية رسم الأدوات وكيفية تشغيل البرامج، سيفيدك لو أردت أن تتعلم لغة أخرى من لغات دوت نت مثل سي شارب مثلا.. فالفارق الوحيد بين سي شارب وفيجيوال بيزيك هو طريقة كتابة الكود فقط!
ليس هذا فحسب، بل إن جعل بيئة التطوير مشتركة بين كل لغات دوت نت يتيح كذلك إنشاء برامج تستخدم أكوادا مكتوبة بلغات برمجة مختلفة، مع إمكانيّة فتح كلّ هذه الأكواد معا في وقت واحد في نفس الواجهة!
وتتضمن بيئة التطوير مجموعة متكاملة من الأدوات والتسهيلات في كل مراحل كتابة المشاريع.. من هذه الأدوات ما يلي:

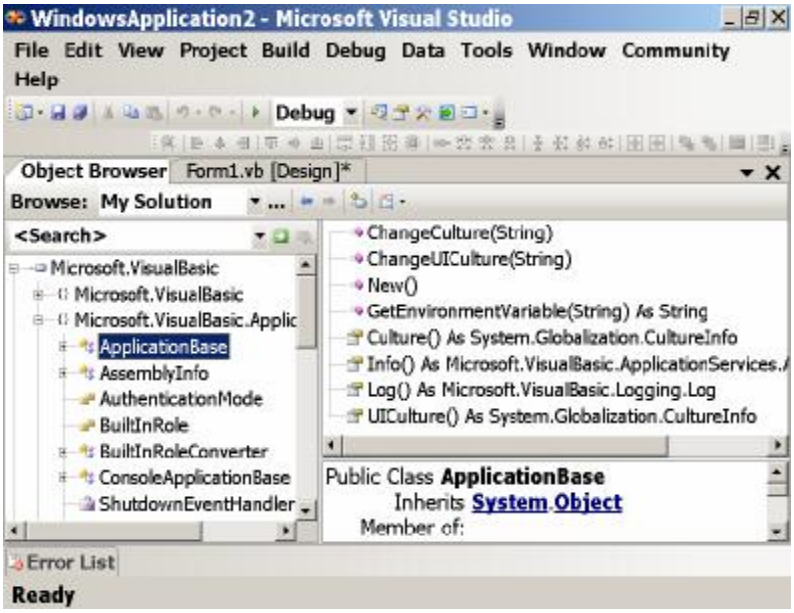
١- مصمم النماذج Form Designer:
للمساعدة في إنشاء واجهة البرنامج.. وتسمى واجهة البرنامج بالنموذج Form، ويساعدك مصمم النماذج على وضع الأدوات على النموذج وتنسيقها وضبط خصائصها.

٢- محرر الكود Code Editor:
لتسهيل عملية كتابة البرنامج وتصحيح أخطاء صياغة الكود.

٣- مترجم الكود Compiler:
لترجمة كود فيجيوال بيزيك إلى كود لغة ميكروسوفت الوسطية MSIL، وإنشاء الملف التنفيذ للبرنامج exe. .

٤- مصحح الأخطاء Debugger:
تمنحك بيئة التطوير المتكاملة مصححا Debugger لتتبع وتصحيح الأخطاء التي تحدث أثناء تشغيل برنامجك.
إن الكلمة Bug في اللغة الإنجليزية تعني "البق" أو "الحشرة" بوجه عام.. والكلمة Debug تعني إزالة البق ــ من حشية الفراش مثلا.. ونظرا لأن وجود الأخطاء في برنامجك أسوأ من وجود البق في فراشك، فقد صار هذا المصطلح يرمز لتتبع الأخطاء في البرامج واصطيادها وتصحيحها.. لهذا سنترجم الكلمة Debug إلى "تصحيح الأخطاء" واختصارًا إلى "تصحيح"، وسنترجم الكلمة Debugger إلى "المصحح".

٥- متصفح الكائنات Object Browser:
تتيح لك هذه النافذة عرض الفئات Classes، التي تستخدمها في برنامجك، وعرض تفاصيلها الداخلية: وسائلها Methods وخصائصها Properties وأحداثها Events، مع عرض صيغة تعريف كل من هذه الأجزاء، ونبذة عن وظيفتها وكيفية استخدامها.
ويمكن عرض هذه النافذة بفتح القائمة العلوية View وضغط الأمر Object Browser.



٦- ملفات الاستعلام والمساعدة Help:

تسمى هذه الملفات بالاسم MSDN، وهي تحتوي على كل المعلومات اللازمة التي تشرح مكونات بيئة التطوير وأوامر فيجوال بيزيك وفئات إطار العمل وكيفية استخدامها.. ويمكنك عرض ملفات الاستعلام في أي وقت بضغط الزر F1 من لوحة المفاتيح، حيث ستظهر نافذة تعطيك معلومات عن الجزء الذي تتعامل معه في تلك اللحظة في دوت نت.

فتح بيئة التطوير المتكاملة:

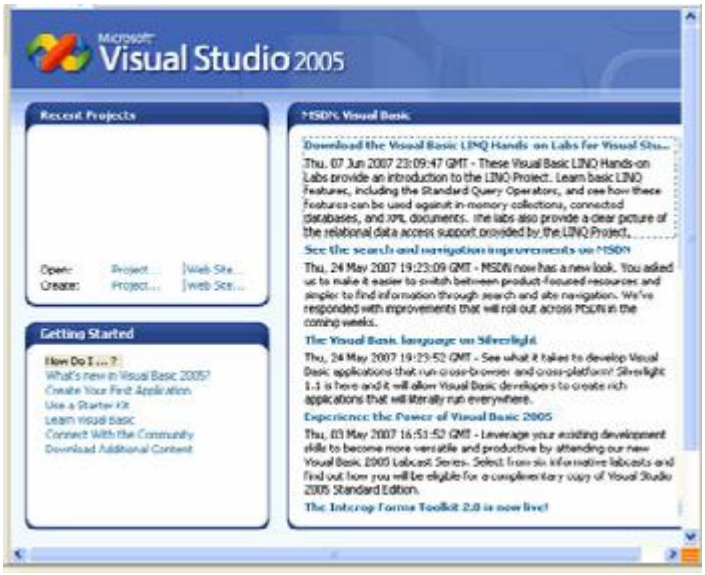
بعد أن تقوم بإعداد فيجيوال ستديو دونت نت على جهازك، يمكنك أن تفتح بيئة التطـــوير باتباع الخطوات التالية:

- من سطح المكتب Desktop الخاص بجهازك اضغط قائمة البداية Start Menu.
- اضغط قائمة البرامج Programs، أو All Programs تبعا لطريقة عرض قائمـــة البداية لديك.
- اختـــر القائمـــة Microsoft Visual Studio 2005.. أو Microsoft Visual Studio 2008، تبعا للإصدار المتوفر لديك.
- اضغط الأمـــر Microsoft Visual Studio 2005.. أو Microsoft Visual Studio 2008، تبعا للإصدار المتوفر لديك.. سيؤدي هذا إلى فتح شاشـــة البدايـــة في فيجيوال ستديو.

إن كانت هذه أول مرة يتم فيها فتح فيجيوال استديو دوت نت على هذا الجهاز، فـــأول مـــا سيواجهك هو نافذة تطلب منك تحديد طريقة التعامل مع الواجهة.. اختر Visual Basic Developer Profile لتخبر الواجهة أنك مبرمج فيجيوال بيزيك واضـــغط زر الموافقـــة Ok.

صفحة البداية Start Page:

أول نافذة ستراها في بيئة التطوير هي صـــفحة البدايـــة Start Page، الموضـــحة فـــي الصورة التالية:



وتتكون صفحة البداية من عدة أجزاء رئيسية، هي:

١ – أحدث المشاريع Recent Projects:

يوجد هذا القسم أعلى يسار صفحة البداية، وتظهر فيه أسماء آخر مشاريع أنشأتها أو فتحتها بفيجيوال بيزيك، مرتبة من الأحدث إلى الأقدم.. ويمكنك الضغط بالفأرة على اسم أي مشروع لفتحه في بيئة التطوير.

ويوجد أسفل هذا القسم عنوانان رئيسيان آخران، هما:

أ. "فتح" Open: وهو يتيح لك إعادة فتح مشروع تم إنشاؤه وحفظه سابقا على الجهاز.

ب. و"إنشاء" Create: وهو يتيح لك إنشاء مشروع جديد.

وبجوار كل من هذين العنوانين يوجد نوعان من البرامج التي يمكنك فتحها أو إنشاؤها:

أ. "مشروع" Project: اضغط هذا الاختيار لإنشاء مشروع فيجيوال بيزيك من أي نوع.. هذا هو النوع الذي سنستخدمه هنا.

ب. موقع ويب Web Site: اضغط هذا الاختيار لإنشاء موقع يعمل على الإنترنت.. لن نتطرق إلى هذا النوع من المشاريع في هذا الكتاب.

٢ – هيا نبدأ Getting Started:

يوجد هذا القسم أسفل قسم أحدث المشاريع، وهو يحتوي على عدة روابط Links لصفحات موجودة على شبكة المعلومات الدولية Internet، تقدم لك معلومات عن:

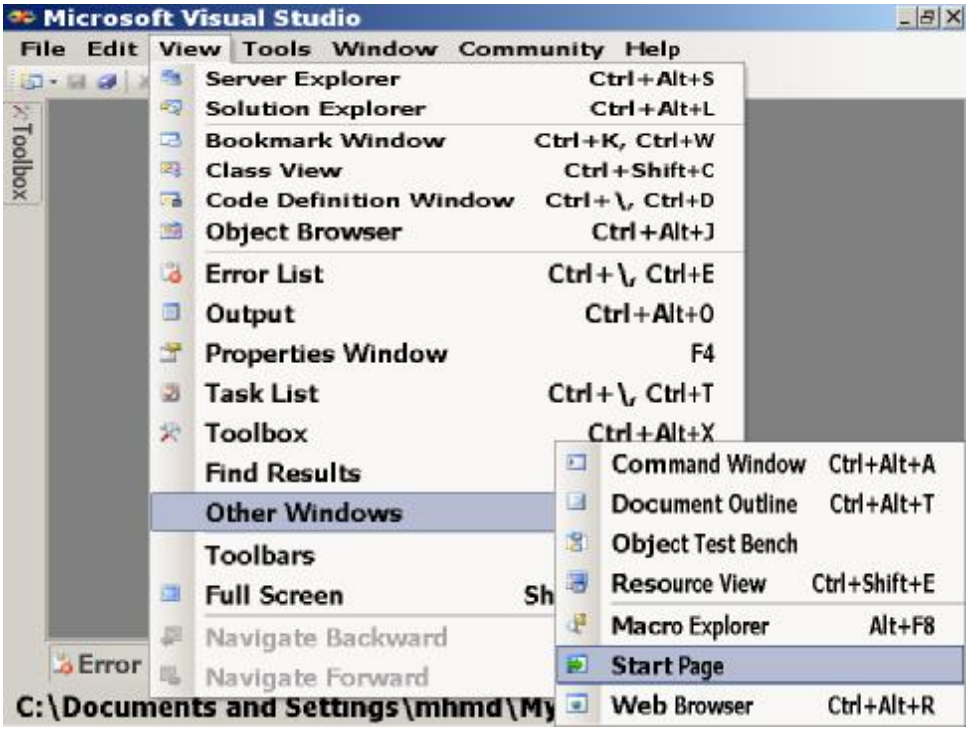
- الجديد في دوت نت.
- كيف تنشئ أول مشروع لك.
- استخدام حزمة البداية Starter Kit.
- تعلم فيجيوال بيزيك.
- التواصل مع عالم البرمجة.
- تنزيل مكونات إضافية.
- ماذا أفعل.... How Do I؟
- مركز المطورين Developer Center.
- بعض الروابط المفيدة لمطوري خادم قاعدة البيانات SQL Server.

لاحظ أنك قد لا تجد بعض هذه الروابط، وقد تجد بعض الروابط غيرها.. السبب في هذا هو أن هذه الروابط يتم تحديثها إذا كان جهازك متصلا بالإنترنت.

٣ – أخبار مطوري فيجيوال ستديو Visual Studio Developers News:

يوجد هذا القسم في الجانب الأيمن من صفحة البداية، وهو يعرض أحدث الأخبار والعروض من موقع شبكة مطوري ميكروسوفت Microsoft Developer Network المعروف اختصار باسم MSDN.

ويمكنك إغلاق صفحة البداية بضغط العلامة X الموجودة أعلى يمين النافذة، ولو أردت إعادة عرضها، فافتح قائمة العرض View Menu أعلى نافذة بيئة التطوير، واضغط الأمر Start Page.. لاحظ أنك قد لا تجد هذا الأمر في بعض نسخ دوت نت تحت القائمة View.. في هذه الحالة ستجده في قائمة فرعية تحت قائمة العرض اسمها "نوافذ أخرى" Other Windows كما هو موضح في الصورة.

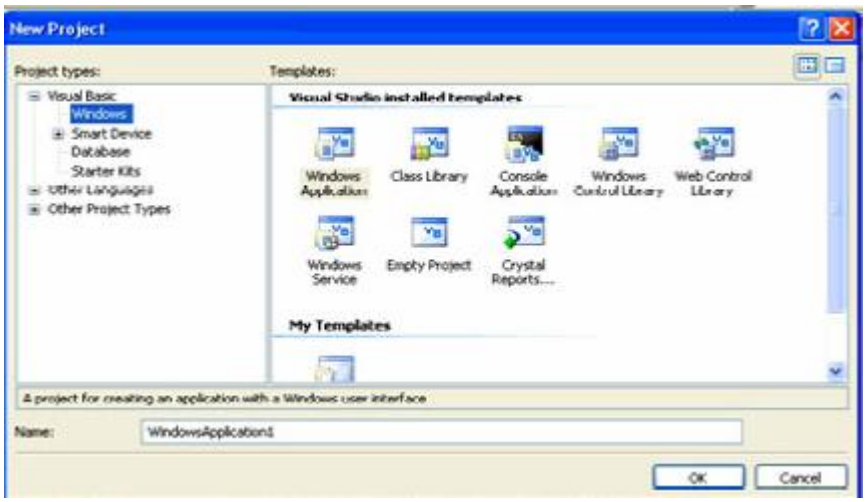


بدء مشروع جديد:

يمكنك أن تبدأ مشروع فيجيوال بيزيك جديدا، باستخدام أي من الطرق التالية:

- اضغط على Create New Project في صفحة البداية Start Page.
- اضغط القائمة الرئيسية "ملف" File أعلى نافذة بيئة التطوير IDE، واضغط الأمر "مشروع جديد" New Project.. لاحظ أن العنصر New Project يوجد في بعض إصدارات دوت نت في قائمة فرعية اسمها New تحت القائمة الرئيسية File.
- يمكنك استخدام لوحة المفاتيح مباشرة لفتح مشروع جديد، وذلك بضغط زر التحكم Control مع زر التحويل Shift مع الزر N (يكتب هذا باختصار على الصورة Ctrl+Shift+N).

أيا كانت الطريقة التي اتبعتها، فسيظهر لك مربع حوار "مشروع جديد" New Project.. في هذا المربع سترى على اليسار قائمة باللغات المتاحة في دوت نت.. اختر Visual Basic.. على اليمين ستعرض القائمة أنواع المشاريع التي يمكن إنشاؤها بفيجيوال بيزيك.



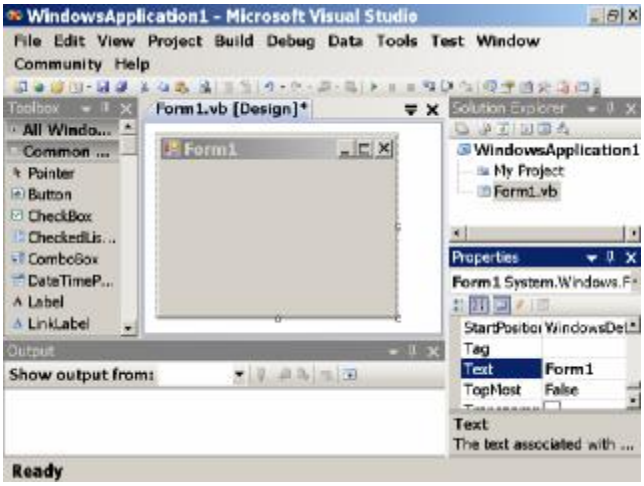
اختر النوع "تطبيق ويندوز" Windows Application.. ستقترح فيجيوال بيزيك اسمًا افتراضيًا للمشروع WindowsApplication1، ستجده في خانة اسم المشروع Name أسفل النافذة.. غيّره إلى ما تحب أو اتركه كما هو لو أردت.. أنصحك بأن تسمي هذا المشروع Hello.

ملحوظة:
رغم أن تسمية المشروع والنماذج والمتغيرات بحروف عربية مسموح به، إلا أنني أنصحك بتسميتها بحروفٍ أجنبية، وذلك حتى لا تُضطر أثناء كتابة الكود إلى الانتقال بين اللغةِ العربية والإنجليزية، كما أنّ بعض أنظمة التشغيل تسبب مشاكل مع الحروف العربية.. ويمكنك في المقابل أن تسّمى عناصرك بمنطوقٍ عربيٍّ لكن بحروف أجنبية.

اضغط زر موافق OK، لإغلاق مربع الحوار.

في بيئة التطوير IDE:

ما تراه أمامك الآن هو بيئة التطبيق وقد عرضت مصمم النماذج Form Designer للمشروع الجديد.



نحن الآن في وضع يسمى وقت التصميم Design Time، وهو يسمى كذلك لأن المبرمج يستطيع تصميم شكل النموذج والأدوات وكتابة الكود.. هذا يختلف عن وضع آخر يسمى وقت التشغيل Run Time، وفيه يشغل المبرمج البرنامج ليرى نتيجة عمله، ويختبره ليتأكد من عدم وجود أخطاء في البرنامج.

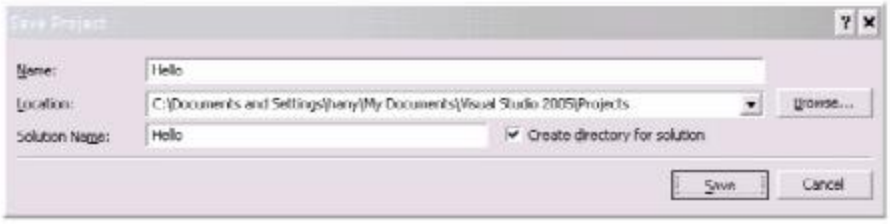
وفي وضع التصميم يمكنك أن ترى الأجزاء التالية في بيئة التطوير:

- في المنتصف ترى نموذجا Form اسمه Form1.. هذه هي شاشة البرنامج الخاص بك، ويمكنك أن تضع عليها ما تريد من الأدوات.
- وعلى اليسار يبدو صندوق الأدوات Toolbox.. إذا لم يكن هذا الصندوق ظاهرا يمكنك عرضه بضغط الأمر Toolbox من القائمة الرئيسية View في أعلى النافذة.
- في الأسفل تلوح نافذة المخرجات Output Window.. هذه النافذة تعرض بعض البيانات عند تشغيل البرنامج وإغلاقه.. لو لم تكن هذه النافذة ظاهرة، فيمكنك عرضها بفتح القائمة الرئيسية View وضغط الأمر Output Window.. لو لم تجد هذا الأمر مباشرة، فستجده تحت القائمة الفرعية Other Windows.
- أما على اليمين فهناك نافذة متصفح المشاريع Solution Explorer، التي تعرض الملفات الموجودة في المشروع الحالي، ومنها ملف النموذج Form1.vb.. لاحظ أنك تستطيع ضغط اسم أي عنصر في القائمة ليتم فتحه.. وإذا لم تكن هذه النافذة ظاهرة يمكنك عرضها بضغط الأمر Solution Explorer من القائمة الرئيسية View في أعلى النافذة.

- وأسفل نافذة متصفح المشاريع هناك نافذة الخصائص Properties Window.. هذه النافذة تعرض لك خصائص الكائن المحدد حاليا في مصمم النماذج.. الكائن المحدد Selected هو الذي تضغطه بالفأرة مرة واحدة ليكون هو الكائن الفعال.. فمثلا، لو ضغطت فوق سطح النموذج فسيكون هو الكائن المحدد، وستعرض هذه النافذة خصائصه، حيث يمكنك تغيير قيم هذه الخصائص مما يوفر عليك كتابة الكثير من الكود لفعل هذا.

حفظ البرنامج:

حتى الآن لم يتم حفظ البرنامج على القرص الصلب.. لفعل هذا افتح القائمة File واضغط الأمر Save All.. هذا الأمر يؤدي إلى حفظ كل الملفات التي حدث فيها تغيير في المشروع.. ونظرا لأن ملف المشروع نفسه لم يتم حفظه حتى الآن، فسيؤدي هذا إلى عرض مربع حوار "حفظ المشروع" Save Project، كما في الصورة التالية:



ستجد أن اسم المشروع يظهر في الخانة العلوية Name، وهو هنا Hello.. أمامك الآن فرصة لتغيير هذا الاسم لو أردت.

تحت خانة اسم المشروع توجد خانة الموقع Location، حيثُ يمكنك هنا أن تكتب مسار واسم المجلد الذي سيتم حفظ البرنامج فيه، أو يمكنك اختيار المجلد من على جهازك مباشرة، بضغط الزر التصفح Browse.

لاحظ أن كل مشرع يتم حفظه في مجلد خاص به له نفس اسم المشروع، ويتمّ إنشاؤه داخل المجلد الذي حدّدته أنت في خانة الموقع.. فمثلا: لو اخترت الموقع C:\، فسيتم إنشاء مجلد اسمه Hello على هذا المسار، ويتم حفظ المشروع به.

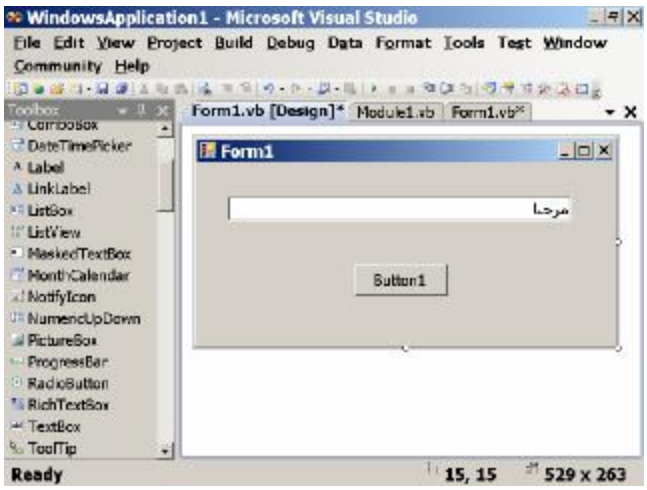
تحذير:
يقترح عليك مربع الحوار مسارا افتراضيا في خانة الموقع.. أنصحكُ أن تغيّرَ هذا المجلد، لأنه في الغالب سيكونُ في القسم من القرص الصلب الذي تم إعداد الويندوز عليه، مّما يهدّد بضياعه وكلّ ما يحتويه من برامج في حالة حدوث خللٍ في الويندوز يحتاجُ إلى إعادةِ تنصيبه على الجهاز.. لهذا تخيّر الموقعَ على أيٍّ جزءٍ آخرَ من القرص الصُلبِ بعيدٍ عن نظام التشغيل.. يمكنك مثلا أن تنشئ مجلدا خاصا بك على المحرك D:\ اسمه My VB Projects وتحفظ فيه كل برامجك.

بعد أن تحدد المسار اضغط موافق لتنفيذ عملية الحفظ وإغلاق النافذة.
دعنا الآن نعود إلى مشروعنا، لنرى كيف سنتعامل معه.

الفئات في كل مكان:

أوّل معلومة أحبّ أن أخبرك بها هي أن النموذج ــ وكذلك جميع الأدوات التي توضع عليه ــ هي فئات Classes جاهزة يقدمها لنا إطار عمل دوت نت .NET Framework، ليسمح لنا ببناء تطبيقات ويندوز لها نفس الشكل والطابع الذي تتميز به البرامج التي نعرفها ونستخدمها على نظام تشغيل الويندوز، مثل وورد Word والرسام Paint وغير ذلك.

- فالنموذج مثلا هو نافذة تظهر على الشاشة وتسمح برسم الأدوات عليها.. هذه النافذة ترسمها وتتحكم في أدائها فئة اسمها Form.. هذا يعني أن Form1 هو كائن Object تم تعريفه من فئة النموذج Form Class.
- والزر هو مساحة على النموذج، ترسمها وتتحكم فيها فئة اسمها Button.. ويأخذ الزر شكل مستطيل بارز، وعند الضغط عليه بالفأرة يأخذ شكلا غائرا ليبدو كزر تم ضغطه.. ويعرض الزر نصا توضيحيا ليدل على وظيفته، كما يقوم الزر بتنفيذ بعض الكود في حالة ضغط المستخدم له، وسنعرف بعد قليل كيف يمكن فعل هذا.
- ومربع النص هو مساحة من النموذج تسمح للمستخدم بالكتابة فيها.. وهذه المساحة يرسمها ويتحكم فيها فئة اسمها TextBox.
- ويسمى كل من الزر ومربع النص "أداة تحكم" Control.. إذن فأداة التحكم: هي كائن يتم رسمه على النموذج لأداء وظيفة معينة.. ويعتبر النموذج أيضا أداة تحكم، لكنها أداة تحكم من نوع خاص، فالنموذج هو الأرضية التي يتم رسم باقي الأدوات عليها، لهذا يظهر مباشرة على شاشة الحاسوب.



و توضح الصورة نموذجا في وضع التصميم Design Time وعليه مربع نص وزر.
ويسمى النموذج وما عليه من أدوات باسم الواجهة المرئية Visual Interface للبرنامج الذي تقوم بإنشائه.. ويمكنك بواسطة مصمم النماذج Form Designer أن تصمم هذه الواجهة، حيثُ يمكنكُ أن تضعَ كل الأدوات Controls التي تريد أن تعرضها لمستخدم برنامجك في وقت التشغيل Runtime.

صندوق الأدوات Toolbox:

قم بتحريك مؤشر الفأرة فوقَ شريط صندوق الأدوات Toolbox Tab في أقصى يسار الشاشة.. سيبرز إليك صندوقُ الأدوات.. وإذا لم يكن هذا الصندوق ظاهرا يمكنك عرضه بضغط الأمر Toolbox من القائمة الرئيسية View في أعلى النافذة، أو بضغط الأيقونة الخاصة به على شريط الأدوات.

ويحتوي صندوق الأدوات على أيقونةٍ Icon لكلِّ أداةٍ يمكنك استخدامها على نموذجك، وقد تم ترتيبُ وتبويب الأدوات في صندوق الأدوات تحت عدة شرائط Tabs، على حسب نوع الأداة ووظيفتها.

اضغط بالفأرة الشريط المسمى Common Controls لتظهر لك الأدوات شائعة الاستخدام.. لاحظ أن الأدوات مرتبة أبجديا تحت كل شريط على حسب أسمائها.

ماذا تفعلُ الآن لوضع الأدوات على نموذجك؟

ما عليك إلا أن تنقر مرتين بالفأرة Double-click على الأيقونة التي تمثل الأداة المطلوبة (ولتكن مربع النص TextBox)، وفي الحال ستوضع على النموذج نسخة جديدة من هذه الأداة، ذات موضع وأبعاد افتراضيّة.

يمكنُك بعد ذلك أن تغيّر موضع مربع النص بضغطه بزر الفأرة الأيسر مرة واحدة وعدم رفع إصبعك عن الزر مع التحرك بالفأرة.. سيؤدي هذا إلى سحب الزر إلى أي موضع تريده.. اترك زر الفأرة لترك الزر في الموضع المفضل لديك.

كما يمكنُك أن تغيّر أبعاد مربع النص بأن تتحرّكَ بالفأرةِ فوقَ جوانبه، وتتوقف فوقَ المربّعات الصغيرةِ على محيطه.. هذه المربعات تسمى المقابض Handles.. وعندما يكون مؤشر الفأرة فوقها يتحوّل شكله إلى سهم ذي رأسين.. عندما يحدث هذا اضغط زرّ الفأرةَ الأيسر واسحب في أيّ اتجاه أثناء ضغطك.. بهذا تستطيعُ أن تزيد أو تنقص من طولِ مربع النص في هذا الاتجاه.. لاحظ أن بإمكانك استخدام نفس الطريقة لتغيير حجم النموذج نفسه.

وتوجد طريقة أخرى لوضع نسخة من مربع النص على النموذج، وذلك بضغط أيقونته في صندوق الأدوات بالفأرة مرة واحدة، ثم الانتقال إلى النموذج وضغط زر الفأرةِ الأيسر والتحرك بها فوقَ النموذج مع استمرار الضغط.. ستجد أنّ مستطيلا يمثل الإطار الخارجيَّ لمربع النص يرتسمُ على النموذج وتتغيّر أبعاده مع حركة الفأرة.. عندما ترى أن أبعاد المستطيل ملائمة لما تريده اترك زر الفأرة.. في الحال سيرتسم مربع النص داخلَ الإطارِ الذي حدّدته.. يمكنكُ بالطبع تغيير أبعاد مربع النص وموضعه من جديد في أية لحظةٍ كما ذكرنا سابقا.

لديك الآن مربعا نص على الشاشة.. ولحذف أحدهما، قم بتحديده وذلك بضغطه بالفأرةِ مرةً واحدة، ثم اضغط زر الحذف Delete من لوحة المفاتيح.

نافذة الخصائص Properties Window:

تظهر خصائص كل أداةٍ في نافذة الخصائص Properties Window، التي توجد في أقصى يمين الشاشة، وبواسطتها يمكنك عرض وتغيير خصائص الأداة المحددة حاليا على النموذج.

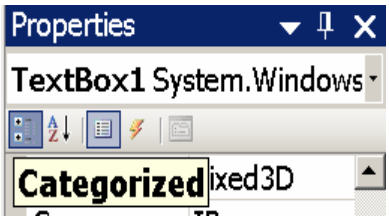
وإذا لم تكن نافذة الخصائص ظاهرة، فاضغط القائمة الرئيسية "عرض" View ثم اختر منها "نافذة الخصائص" Properties Window.. كما يمكنك أن تضغط بزر الفأرة الأيمن Right-Click فوق أية أداة لعرض القائمة الموضعية Context Menu، واختيار الأمر Properties Window منها.. أو يمكنك عرض نافذة الخصائص مباشرة بضغط الزر F4 من لوحة المفاتيح.

وإذا لم تكن هناك أداة محددة ولم يكن النموذج محددًا، فستعرض نافذة الخصائص خصائص العنصر الحالي في متصفح المشاريع Solution Explorer، الذي يعرض أسماء الملفات الموجودة في المشروع الحالي.

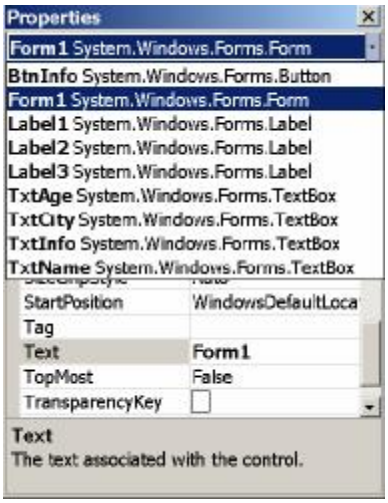
والآن اضغط علي مربع النص بزر الفأرة الأيمن، ومن القائمة الموضعية اختر "الخصائص" Properties.

في نافذة الخصائص، يمكنك أن ترى الخصائص التي تحدد كيف تظهر كل أداة على النموذج، وفي بعض الأحيان كيف تعمل.

وتوجد طريقتان لترتيب الخصائص في نافذة الخصائص:



- الطريقة الأولى هي ترتيب الخصائص تبعا لوظيفتها، حيث يتم جمع كل الخصائص المتشابهة في وظيفتها تحت عنوان يميزها.. ويمكنك تفعيل هذه الطريقة بضغط زر التصنيف Categorized، وهو أول زر على اليسار أعلى نافذة الخصائص.



- الطريقة الثانية هي ترتيب الخصائص أبجديا على حسب أسمائها.. وأنا أنصحك باستخدام هذه الطريقة لأنها تسهل عليك الوصول إلى أية خاصية بسرعة.. ويمكنك تفعيل هذه الطريقة بضغط زر الترتيب الأبجدي Alphabetical المجاور لزر التصنيف.

لاحظ أن هناك قائمة أعلى نافذة الخصائص يظهر بها معلومات عن الأداة التي يتم عرضها حاليا في نافذة الخصائص.. هذه المعلومات تبدأ باسم الأداة (مثل TextBox1) ويليها نوع الأداة (مثل System.Windows..Forms.TextBox الذي يمثل نوع أداة مربع النص).. وتسمى هذه القائمة

بمربع الكائنات Object Box، ولو ضغطت الزر الموجود في جانبها الأيمن فستنسدل القائمة وتعرض لك أسماء كل الأدوات الموجودة على النموذج، بما في ذلك اسم النموذج نفسه.. ويمكنك أن تختار أية أداة من هذه الأدوات بضغطها بالفأرة، ليتم تحديدها على النموذج وعرض خصائصها في نافذة الخصائص.
وعند تحديد اسم أية خاصية في نافذة الخصائص، يظهر تعريف مختصر لها في أسفل نافذة الخصائص ليساعدك على فهم وظيفتها.. في الصورة مثلا ترى أن الخاصية Text محددة في نافذة الخصائص، وفي أسفل النافذة هناك نص يقول:

Text
The text associated with the control.

بمعنى: النص المرتبط بالأداة.
كما يمكنك أن تضغط الزر F1 من لوحة المفاتيح لتظهر لك نافذة المساعدة Help Window، التي تشرح لك كل التفاصيل المتعلقة بهذه الخاصية، والأخطاء التي يحتمل أن يسببها وضع بعض القيم الخاطئة بها، وأحيانًا أمثلة على استخدامها.. حاول أن تعتاد قراءة هذه المعلومات، ولا تجزع من كونها بالإنجليزية، فهي إنجليزية مبسطة، ومعظمها مصطلحات برمجية سريعا ما ستألفها.
تعال نجرّب بعض خصائص مربع النص:

خاصيّة الاسم Name:

تحدد هذه الخاصية اسم الأداة الذي سنستخدمه في البرمجة.. لاحظ أن هذه الخاصية لا تظهر في موضعها المفترض عند ترتيب الخصائص أبجديا، ولكن يتم وضعها بين قوسين (Name) وعرضها في أعلى النافذة.. السبب في هذا هو أن هذه أهم وأول خاصية يبحث عنها المبرمج، لأن الاسم الذي يكتبه فيها يستخدمه في كتابة الكود.. لهذا تم تسهيل الأمر عليه ووضعها أعلى نافذة الخصائص حتى لا يضيع وقتا في البحث عنها.
دعنا نغير قيمة الخاصية Name لمربع النص الخاص بنا إلى TxtWelcome بدلا من النص الافتراضي "TextBox1".. لفعل هذا احذف النص المكتوب في الخانة الموجودة على يمين الخاصية Name، واكتب النص TxtWelcome بدلا منه.

ملحوظة:
إذا كانت قيمة الخاصية قابلة للكتابة، فإن النقر مرتين بالفأرة فوق اسم أية خاصية أو قيمتها يؤدي إلى تحديد النص المكتوب في خانة القيمة مباشرة. أما إذا كانت للخاصية مجوعة محدودة من القيم، فإن النقر المزدوج على القيمة الحالية للخاصية يؤدي إلى تغييرها واختيار القيمة التالية لها مباشرة.

خاصيّة النص Text:

والخاصية Text هي المسئولة عما يُعرض في مربع النص من كلمات، وأي نص يُكتب بها ستجده مكتوبا داخل مربع النص.
ابحث عن خاصية النص Text في نافذة الخصائص، وضع فيها القيمة: هذا هو مربع النص الخاص بي.. ولا تضع أية علامات تنصيص في بداية ونهاية النص.. علامات التنصيص تستخدم فقط في الكود، وليس في نافذة الخصائص.

ملحوظة:
لن ترى أي تغيير يحدث في مربع النص بعد تغيير قيمة الخاصية Text إلا بعد أن تغادر خانة الخاصية Text إلى خاصية أخرى أو إلى النموذج أو أية أداة عليه.. ولو شئت أن ترى التغيير دون أن تغادر خانة الخاصية Text، فعليك أن تضغط Ctrl+S من لوحة المفاتيح لحفظ التغييرات.

نريد الآن وضع زر على النموذج.
عُد إلى صندوق الأدوات، وانقر مرتين بالفأرة على الأداة Button.. سيؤدي هذا إلى وضع زر على النموذج.
حرّك الزر المرسوم على النموذج ليصبح أسفل منتصف مربع النص.. اذهب إلى نافذة الخصائص، وحدد الخاصية Text الخاصة بالزر واجعل قيمتها "ترحيب".. غير كذلك خاصية الاسم Name إلى BtnWelcome.

تشغيل البرنامج:

هناك عدة طرق لتشغيل البرنامج، منها:

- فتح القائمة الرئيسية "تصحيح" Debug أعلى النافذة، وضغط الأمر "بدء التصحيح" Start Debugging.
- أو ضغط زر Start Debugging على شريط الأدوات العلوي الموضح في الصورة.

- أو ضغط الزر F5 من لوحة المفاتيح.

لكي يعمل البرنامج، يجب أن يقوم المترجم Compiler بترجمة الكود إلى لغة ميكروسوفت الوسيطة MSIL أولا، وبعد لحظاتٍ قليلةٍ سيتم تشغيل البرنامج، حيث سيظهر على الشاشة نموذج يحتوي على مربع نص وزر.. هذا هو نفس ما سيراه المستخدم عندما يشغل برنامجك.
أدخل بعض النصوص في مربع النص.. اكتب مثلا: "مرحبا".. حدد هذه الكلمة بضغط زر الفأرة الأيسر مع التحرك يمينا أو يسارا قبل ترك الزر.. اضغط Ctrl+C من لوحة المفاتيح لنسخ النص الذي حددته.. إن بإمكانك الآن أن تضغط Ctrl+V، لتلصق هذا النص المنسوخ في أي موضع من مربع النص الحالي.. يمكنك أيضا أن تلصق هذا النص

في أية أداة أخرى تقبل الكتابة، موجودة على النموذج الحالي أو على أي نموذج آخر، في نفس البرنامج أو في برامج أخرى.. والعكس أيضًا صحيح، فالنصوص المنسوخة من أي تطبيق آخر، يمكن لصقها في مربع النص الحالي.
ترى.. كيف يحدث هذا، ونحن لم نكتب أي كود؟
قلت لك إنك تستخدم فئة جاهزة من فئات إطار العمل وهي فئة مربع النص TextBox Class، وهي تمنحك العديد والعديد من الإمكانيات التي تحيل حياتك كمبرمج إلى نعيم!
وفكرة النسخ واللصق بسيطة، فكل ما تفعله فئة مربع النص هو حفظ النص المنسوخ في مخزن موجود في نظام الويندوز اسمه لوحة القصاصات Clipboard، وهو مخزن عام يستطيع أي برنامج أن يكتب فيه ويقرأ منه، لذلك يُعد من أبسط الوسائل لنقل النصوص والصور بين البرامج المختلفة.
بعد أن تفرغ من تجريب البرنامج، قم بإغلاقه.. لفعل هذا يمكنك اتباع أية طريقة من الطرق المختلفة التالية:

- قم بإغلاق النموذج.. إن إغلاق النموذج الرئيسي يؤدي إلى إغلاق البرنامج.. ونظرا لأن برنامجنا يحتوي على نموذج واحد فقط فهو يعتبر النموذج الرئيسي له.
- أو عد إلى بيئة التطوير، واضغط القائمة الرئيسية "تصحيح" Debug، واضغط الأمر "إيقاف التصحيح"Stop Debugging.
- أو اضغط زر إيقاف البرنامج Stop Debugging من شريط الأدوات، كما هو موضح بالصورة.
- أو اضغط Shift+F5 من لوحة المفاتيح.

تشغيل البرنامج من خارج بيئة التطوير IDE:
بمجرد تشغيل البرنامج في بيئة التطوير تتم ترجمته كاملا وإنشاء ملف تنفيذي للبرنامج Executable File.. ولكن أين يوجد هذا الملف بالضبط؟
دعنا نأخذ المشروع Hello كمثال.. لو استخدمت متصفح الويندوز Explorer، وذهبت إلى المسار الذي حفظت عليه هذا المشروع، فستجد ما يلي:

- هناك مجلد اسمه Hello، يحوي على ملف المشروع Hello.sln.. لو ضغطت هذا الملف مرتين بالفأرة فسيتم فتح بيئة التطوير وعرض المشروع Hello.
- داخل المجلد Hello ستجد مجلدا آخر اسمه Hello أيضا.. في هذا المجلد ستجد العديد من الملفات الخاصة بالمشروع، بعضها خاص بمصمم النموذج، وبعضها خاص بالكود الذي تكتبه في المشروع.. لكن ما يهمنا هنا هو مجلد اسمه bin.. افتح هذا المجلد.. ستجد بداخله مجلدا آخر اسمه Debug.. لو فتحت هذا المجلد فستجد بداخله ملفا امتداده Exe. وله نفس اسم المشروع (Hello.exe).. هذا هو الملف التنفيذي للبرنامج.
- انقر الملف التنفيذي مرتين بالفأرة.. سيؤدي هذا إلى فتح البرنامج مباشرة من خارج بيئة التطوير.

أظننا قد أخذنا خلفية مناسبة عن مصمم النماذج حتى الآن.. نريد إذن أن ننتقل إلى كتابة بعض الكود.. هيا بنا.

لغة فيجيوال بيزيك:

قبل أن نكتب أول كود لنا في فيجيوال بيزيك، علينا أن نتعرف على القليل من الحقائق عن طريقة كتابة الكود في هذه اللغة.

١- نهاية الأمر في فيجوال بيزيك:

أصغر جزء من الكود يكتبه المبرمج هو الأمر Command.. والأمر في فيجيوال بيزيك ينتهي بانتهاء السطر.. مثلا: السطران التاليان يمثلان أمرين كاملين موجهين إلى فيجيوال بيزيك:

```
Dim X As Integer = 0
Dim Y As Integer = 1
```

الأمر الأول يطلب تعريف متغير اسمه X ونوعه عدد صحيح Integer وقيمته المبدئية صفر، والثاني يعرف متغيرا اسمه Y نوعه عدد صحيح، وقيمته المبدئية ١.
وكما ترى: تمت كتابة كل أمر في سطر مستقل.. وطبعا من الخطأ وضع الأمرين في سطر واحد كالتالي:

```
Dim X As Integer = 0 Dim Y As Integer = 1
```

لكن لو كنت مصرا على فعل هذا لهدف في نفسك (كأن تكون بخيلا وتريد توفير عدد السطور ☺)، فاستخدم النقطتين العموديتين (:) للفصل بين الأمرين كالتالي:

```
Dim X As Integer = 0 : Dim Y As Integer = 1
```

ومن الخطأ أيضا تجزئة الأمر الواحد على سطرين أو أكثر كالتالي:

```
Dim X
As Integer = 0
```

لكن لو كنت مصرّا، فاستخدم علامة تكملة السطر.. هذه العلامة تتكون من مسافة تليها العلامة _ .. هذه العلامة تسمى الشرطة التحتية أو الشرطة المنخفضة Underscore.. هكذا يمكن فصل الأمر الواحد في فيجوال بيزك على سطرين:

```
Dim X _
As Integer = 0
Dim Y As Integer _
= 1
```

ولو شئت حتى، يمكنك فصل الأمر الواحد على أكثر من سطرين:

```
Dim S _
As String = _
"My name is Ali"
```

لاحظ أن هذا التقسيم لضرب المثل، وليس له دلالة خاصة هنا، ولكنه ليس خاطئا من وجهة نظر فيجيوال بيزيك.. هذه الميزة تتيح لك تقسيم الأمر الطويل جدا إلى أكثر من سطر، بحيث لا يتجاوز عرض كل سطر منها عرض الشاشة، وبالتالي تستطيع قراءة الكود بدون إرهاق نفسك بتحريك المُنزلق الأفقي Horizontal Scroll Bar يمينا ويسارا.
بالنسبة لنا في هذا الكتاب، سنستخدم علامة تكملة الأسطر لتجزئة السطور الأطول من عرض الصفحة إلى أكثر من سطر.
لاحظ أيضا أنك لا تستطيع وضع علامة تقسيم السطر داخل علامتي التنصيص.. الكود التالي على سبيل المثال غير صحيح:

```
Dim S As String = "My name is _
Ali"
```

والصواب هو أن تضع كل جزء من النصين المجزئين بين علامتي تنصيص، وتجمعهما معا باستخدام العلامة +، وبهذا تستطيع أن تستخدم علامة تكملة السطر قبل أو بعد العلامة +.. هكذا مثلا:

```
Dim S As String = "My name is " _
+ "Ali"
```

٢- وضع التعليقات Comments:

لكي يكون الكود واضحا، يلجأ المبرمج إلى وضع بعض التعليقات هنا أو هناك.. يتم هذا في فيجيوال بيزيك بإحدى طريقتين:

١- باستخدام العلامة ' كالتالي:

```
سنعرّف أحد المتغيرات '
نوع هذا المتغير عدد صحيح ' Dim X As Integer
```

٢- أو باستخدام التعبير :Rem كالتالي:

```
سنعرّف أحد المتغيرات :Rem
نوع هذا المتغير عدد صحيح:Dim X As Integer Rem
```

لاحظ أن التعليق ليس أمرا، ولن يتم تنفيذه، بل لن يتم ترجمته عند إنشاء الملف التنفيذي للبرنامج Exe.
وتقوم فيجيوال بيزيك بتلوين التعليق باللون الأخضر لتمييزه عن باقي الكود.

٣- حالة الأحرف Character Casing:

على عكس باقي لغات البرمجة، فإن المتغيرات في لغة البيزيك غير حساسة لحالة الأحرف Case-insensitive، بمعنى أنها لا تهتم إن كان الحرف الإنجليزي كبيرا Capital أم صغيرا Small.. فالأسماء username و userName و UserName كلها متكافئة، وتشير إلى نفس المتغيّر.. معنى هذا أنك لا تستطيع

استخدام هذه الكلمات لتعريف ثلاثة متغيرات مختلفة، فكلها تُعتبر اسما واحـدا.. كمـا يمكنك أن تكتب في الكود أي شكل منها بدون أن يحدث خطأ.. يمكنك أن تكتب مثلا:

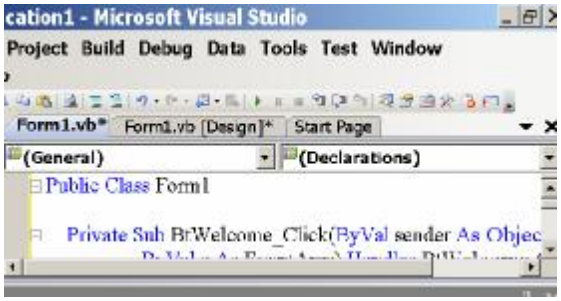
```
Dim UserName As String
username = "محمد"
```

في لغة أخرى (مثل سي شارب C#) سيعطيك مترجم الكود خطأ.. لكـن هـذا لـن يحدث في فيجيوال بيزيك، بل على العكس، ستقوم فيجيوال بيزيك بتصـحيح الاسـم الذي كتبته تلقائيا بمجرد مغادرة السطر، ليصير:

```
UserName = "محمد"
```

كتابة بعض الكود:

نريد الآن أن نكتب بعض الكود في التطبيق Hello، بحيث إذا ضغط المسـتخدم الـزر، نجعل مربع النص يعرض النص "مرحبا بك في فيجيوال بيزيك دوت نت".



سيدهشك كم أن الأمر بسيط.. كلّ ما عليك فعله هو النقر بالفـأرة مـرتين Double-Click، فـوق الـزر.. سيؤدي هذا إلى الانتقال إلى فتح نافذة محرر الكود Code Editor، وفيهـا سـترى الكـود الخـاص بـالنموذج الحالي.

ولو نظـرت إلـى أعلـى الشاشـة، فسترى ثلاثة أشرطة:

- **الشريط الأول Form1.vb [Design]:**

عند ضغط هذا الشريط بالفأرة يتم عرض مصمم النماذج Form Designer، حيـثُ يمكنك رؤية النموذج Form1 والأدوات الموجودة عليه.

- **الشريط الثاني Form1.vb:**

عند ضغط هذا الشريط بالفأرة يتم عرض محرر الكود Code Editor، حيث يمكنـك رؤية الكود الخاص بالنموذج Form1.

- **الشريط الثالث Start Page:**

عند ضغط هذا الشريط بالفأرة يتم عرض صفحة البداية.

اضغط Form1.vb لتعود إلى صفحة الكود.. ستجد في هذه الصفحة الكود التالي:

```
Public Class Form1
    Private Sub BtnWelcome_Click(ByVal sender As Object, _
        ByVal  e As EventArgs) Handles BtnWelcome.Click

    End Sub
End Class
```

هذا الكود يمثل تعريف فئة Class اسمها Form1، وبداخلها إجراء اسمه BtnWelcome_Click.. أي كود ستكتبه داخل هذا الإجراء سيتم تنفيذه عند ضغط الزر.
دعنا إذن نستخدم هذا الإجراء لنكتب جملة الترحيب في مربع النص.. كما قنا: مربع النص ــ مثل معظم الأدوات ــ له الخاصية Text التي يمكن استخدامها لقراءة النص المعروض في مربع النص، أو لتغيير هذا النص.. هذا هو الكود الذي يعرض جملة الترحيب:

TxtWelcome.Text = "مرحبا بك في فيجيوال بيزيك دوت نت"

هذا هو كل شيء.. لم نكتب أكثر من سطر واحد من الكود!.. لكن دعني أذكرك مجددا أن هذا السطر سيوضع داخل الإجراء BtnWelcome_Click.. إذن فسيكون كود النموذج بعد كتابة هذا الكود كالتالي:

```
Public Class Form1
    Private Sub BtnWelcome_Click(ByVal sender As Object, _
            ByVal e As EventArgs) Handles BtnWelcome.Click
        TxtWelcome.Text = "مرحبا بك في فيجيوال بيزيك دوت نت"
    End Sub
End Class
```

الآن، شغل البرنامج بضغط الزر F5 من لوحة المفاتيح، واضغط زر الترحيب.. سترى أن جملة الترحيب قد ظهرت في مربع النص.
هنيئا لك.. لقد خطوت أولى خطواتك في برمجة الويندوز.. ربما تبدو هذه خطوة صغيرة للبرمجة، لكنها قفزة هائلة لمبرمج ☺!

تنبيه:

الكود الذي كتبته في المشروع لم يتم حفظه بعد على القرص الصلب.. لاحظ أن كل تغيير تقوم به في برنامجك لا يتم حفظه إلا إذا طلبت أنت هذا.. لهذا يجب عليك بين الحين والآخر أن تحفظ التغييرات التي قمت بها في برنامجك.. لا تنسَ فعل هذا، لأن انقطاع التيار الكهربي أو حدوث أية مشكلة غير متوقعة في دوت نت أو الويندوز قد تؤدي في ثانية واحدة إلى ضياع كل الكود الذي كتبته في ساعات!
ويمكنك حفظ التغييرات باستخدام الأمر Save All من القائمة File.. هذا يحفظ التغييرات التي تمت في جميع ملفات البرنامج.. أما لو شئت حفظ التغييرات التي حدثت في الملف الذي تتعامل معه حاليا (كملف النموذج Form1 مثلا)، فستجد أمرا آخر في القائمة File اسمه:

Save Form1.vb

مهمة هذا الأمر هي حفظ ملف النموذج Form1.. لاحظ أن اسم هذا الأمر يختلف تبعا للملف المفتوح حاليا في البرنامج.. فلو أضفت نموذجا آخر إلى برنامجك اسمه Form2 وفتحته، فستجد أن هذا الأمر قد صار:

Save Form2.vb

وهكذا..

وعموما، يمكنك أن تريح نفسك من هذه الأسماء، وتضغط Ctrl+S من لوحة المفاتيح مباشرة، لتحفظ التغييرات التي أجريتها على الملف الذي تتعامل معه حاليا أيا كان اسمه.. كما يمكن أن تفعل نفس الشيء باستخدام زر الحفظ الموجود على شريط الأدوات، وهو يحمل أيقونة تمثل قرصا مرنا دلالة على عملية الحفظ، وبجواره زر حفظ كل الملفات، وهو يحمل مجموعة من الأقراص دلالة على حفظ عدد كبير من الملفات.

أخطاء الصياغة Syntax Errors:

لكل لغة قواعد.. فاللغة العربية مثلا لها قواعد صرف ونحو وإملاء.. هذا ينطبق أيضا على لغات البرمجة، فلكي تكتب أوامر أية لغة برمجة يجب أن تلتزم بقواعد هذه اللغة.. هذه القواعد تسمى طريقة الصياغة Syntax.

ولقد تعرفنا بالفعل على بعض قواعد صياغة فيجيوال بيزيك.. من هذه القواعد مثلا:

١- الأمر يكتب في سطر واحد.
٢- لا يجوز تقسيم الأمر على سطرين إلا باستخدام علامة تكملة السطر _ .
٣- لا يجوز كتابة أكثر من أمر في نفس السطر إلا بوضع النقطتين : بينهما.
٤- لكتابة تعليق يجب وضع العلامة ' في أوله.
٥- يمكن وضع التعليق في نفس السطر بعد انتهاء الأمر، بدون وضع : بينهما.
٦- لا يجوز لصق الكلمات ببعضها عند كتابة الأمر، لهذا تستخدم المسافة للفصل بين أسماء المتغيرات والكلمات الأساسية في فيجيوال بيزيك.
٧- لا يجوز الخطأ في كتابة اسم متغير أو أداة، بل تجب كتابتهما بنفس الحروف التي تم تعريفهما بها.
٨- لا يهم أن تكون حروف المتغيرات والأدوات كبيرة أو صغيرة، لأن فيجيوال بيزيك غير حساسة لحالة الأحرف.
٩- يجب وضع أي نص بين علامتي تنصيص.
١٠- لاستخدام إحدى خصائص الكائن، تجب كتابة نقطة بين اسم الخاصية واسم الكائن بدون وضع أية مسافات.. مثلا:

TxtWelcome.Text = "مرحبا بك في فيجيوال بيزيك دوت نت"

وهكذا...

هذه قواعد بسيطة كما ترى، لكنها بالطبع ليست كل القواعد التي تجب مراعاتها في فيجيوال بيزيك، فمع كل جزء جديد تتعلمه في فيجيوال بيزيك ستتعلم قواعد الصياغة الخاصة به.

الشيء المهم معرفته هنا، هو أن أية لغة برمجة ــ بما في ذلك فيجيوال بيزيك ــ لا تسمح بمخالفة قواعد الصياغة الخاصة بها.. لهذا عليك أن تلتزم بهذه القواعد دائما، وإلا حدث خطأ في الصياغة Syntax Error يمنع تشغيل برنامجك.

تعال نجرب.. دعنا نخطئ في الصياغة عن عمد، لنرى ماذا سيحدث.. مثلا: احذف علامتي التنصيص اللتين تحيطان بجملة الترحيب، كالتالي:

TxtWelcome.Text = مرحبا بك في فيجيوال بيزيك دوت نت

لاحظ أن فيجيوال بيزيك تفحص صياغة كل أمر بمجرد مغادرة السطر الذي كتبته فيه إلى أي سطر آخر.. ومن الطريف أنك لو حذفت علامة التنصيص الأخيرة وحدها فستضعها فيجيوال بيزيك بنفسها آليا نيابة عنك.. لكن حذف علامة التنصيص الأولى أو العلامتين معا سيسبب خطأ في الصياغة ولن تصححه لك فيجيوال بيزيك تلقائيا، بل ستضع خطا متعرجا تحت الكلمات التي سببت خطأ الصياغة، ولو حلقت فوق هذا الخط بالفأرة وسكنت للحظة، فسيظهر تلميح على الشاشة يخبرك بنوع الخطأ.

```
Private Sub BtWelcomw_Click(ByVal sender As System.Ob
    TxtTest.Text = مرحبا بك في فيجيوال بيزيك دوت نت
                   Name 'مرحبا' is not declared.

End Sub
```

في حالتنا هذه مثلا، سيكون لدينا نوعان من الخطأ.. الأول هو:

Name "مرحبا" is not declared.

هذا الخطأ يخبرك بأن المتغير الذي اسمه مرحبا لم يتم تعريفه بعد.. هذا الخطأ سببه اعتقاد فيجيوال بيزيك أن الكلمة "مرحبا" هي اسم أحد المتغيرات، لكنها لم تجد تعريفا لهذا المتغير من قبل.. السبب في هذا الاعتقاد هو عدم وجود الكلمة "مرحبا" بين علامتي تنصيص، فهما ما يفرق بين الكلمة العادية وأسماء المتغيرات من وجهة فيجيوال بيزيك.

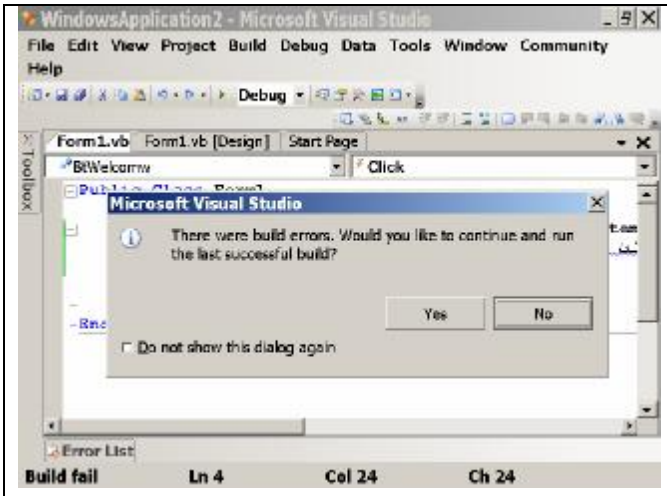
أما الخطأ الثاني فسيتضح لك إذا حلقت بالفأرة فوق أية كلمة بعد الكلمة "مرحبا"، حيث ستخبرك فيجيوال بيزيك أن:

End of statement expected.

هذا الخطأ يخبرك أن فيجيوال بيزيك تتوقع نهاية الجملة، لكنها تفاجأت بوجود هذه الكلمات في هذا الموضع.. هذا الخطأ مترتب على الخطأ الأول، فقد فهمت فيجيوال بيزيك أن مرحبا هو اسم متغير، وبهذا فإن الكود في هذا السطر يطلب وضع قيمة المتغير مرحبا في خاصية النص Text التابعة لمربع النص TxtWelcome، وبناء على هذا لا يجب أن يكون هناك أي شيء آخر بعد هذا في نفس السطر.

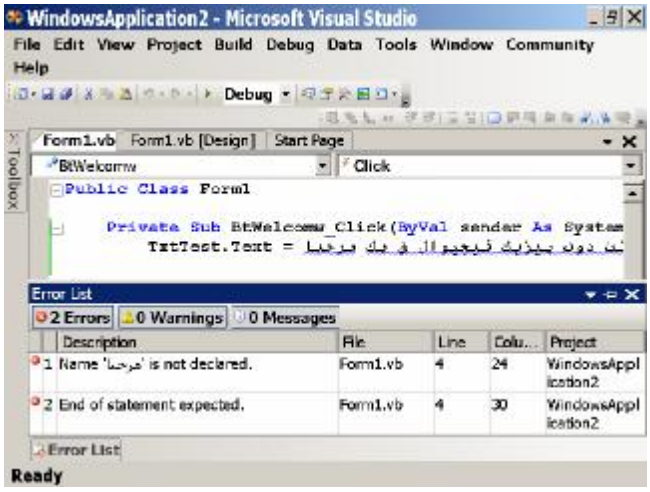
المفروض الآن أن نصحح الصياغة للتخلص من هذين الخطأين.. لكن قبل هذا دعنا نرى ماذا سيحدث لو حاولنا تشغيل البرنامج الآن.

لو ضغطت F5 فستظهر لك رسالة خطأ تخبرك أن البرنامج به أخطاء لهذا لا يمكن تشغيله، وتسألك إن كنت تريد تشغيل آخر نسخة صحيحة من البرنامج.. يمكن أن تجيب بنعم، لتقوم فيجيوال بيزيك بتشغيل آخر نسخة من الملف التنفيذي للبرنامج تم إنشاؤها قبل حدوث هذه الأخطاء، لكن لا توجد فائدة لهذا، والأذكى أن تصحح الأخطاء الموجودة في البرنامج أولا.. لهذا عليك أن تجيب بلا.



لاحظ وجود مربع اختيار Check Box في أسفل رسالة الخطأ.. يمكنك أن تضغط هذا المربع بالفأرة لوضع علامة الاختيار ✓ في هذا المربع.. هذا يعني أنك لا تريد ظهور هذه الرسالة مرة أخرى، وأن الاختيار الذي ستتخذه الآن سيتم تنفيذه تلقائيا في كل مرة.. فلو ضغطت الزر Yes فسيتم تشغيل آخر نسخة صحيحة مترجمة من البرنامج، وإن ضغطت الزر No فسيتم إلغاء عملية تشغيل البرنامج لتصحيح الأخطاء.. الاختيار الأول سيصّعب عليك تصحيح برامجك، بينما الاختيار الأخير هو الأفضل.

قائمة الأخطاء Error List:



تظهر أخطاء البرنامج في نافذة تسمى "قائمة الأخطاء" Error List في أسفل نافذة محرر الكود.. لو لم تكن ظاهرة منذ البداية فستظهر بعد أن تضغط الزر No في رسالة الخطأ السابقة، كما يمكنك أن تعرضها في أي وقت بفتح القائمة الرئيسية View وضغط الأمر Error List.

هذه النافذة تعرض كل الأخطاء Errors الموجودة في البرنامج كما تعرض بعض التحذيرات Warnings إن وجدت.. التحذير ليس خطأ ويمكنك أن تتجاهله، ويمكنك تمييزه من وجود علامة تعجب داخل مثلث أصفر بجواره.. أما الخطأ فتوجد بجواره العلامة × في دائرة حمراء.. لاحظ أنك تستطيع إخفاء التحذيرات من القائمة، وذلك بالضغط على زر التحذيرات أعلى قائمة الأخطاء.. ولو ضغطت عليه مرة أخرى فستعود التحذيرات للظهور في القائمة.

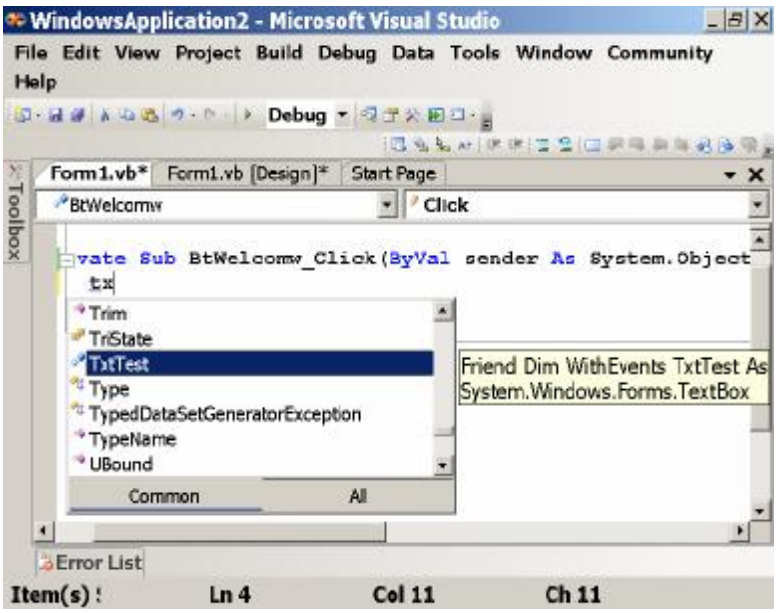
ولتصحيح أي خطأ أو تحذير، يمكنك النقر المزدوج Double-Click بالفأرة فوق هذا الخطأ في قائمة الأخطاء، ليتم الانتقال إلى السطر الذي يوجد به هذا الخطأ في الكود، ومن ثمّ يمكنك تصحيحه.
الآن، ضع علامتي التنصيص حول جملة الترحيب من جديد.. بمجرد مغادرة السطر سيختفي الخط المتعرج وستختفي الأخطاء من قائمة الأخطاء، وستسمح لك فيجيوال بيزيك بتشغيل البرنامج من جديد.
لاحظ أن حذف علامتي التنصيص قد يجعل كلمات الجملة المكتوبة باللغة العربية تظهر بترتيب معكوس، كأنها مكتوبة من اليسار إلى اليمين، كما هو واضح في الصورة السابقة التي توضح خطأ الصياغة.. لا تجعل هذا يخدعك وأنت تضع علامتي التنصيص.. أفضل طريقة لفعل هذا هي أن تضغط بالفأرة بعد العلامة = لتضع مؤشر الكتابة في هذا الموضع، ثم تتحرك بالأسهم إلى اليسار إلى أن تصل إلى الموضع السابق لأول حرف في الجملة العربية، ومن ثم تكتب علامة التنصيص الأولى.. بعد هذا اضغط الزر End من لوحة المفاتيح للتحرك إلى نهاية السطر، وهناك ضع علامة التنصيص الثانية.. هنا سترى أن ترتيب كلمات الجملة العربية قد تغير لتظهر بصورة صحصحة.

الاستشعار الذكي IntelliSense:
قدمت ميكروسوفت خدمة جليلة إلى المبرمجين في نهاية عام ١٩٩٧، مع الإصدار الخامس من فيجيوال ستديو Visual Studio، وهو الإصدار الذي كان يحتوي على الإصدار الخامس من فيجيوال بيزيك "VB5"، وذلك حينما منحتهم تقنية الاستشعار الذكي IntelliSense (اختصارا للتعبير Intelligence Sense).
فقبل ذلك الحين كان على المبرمج أن يكتب كل شيء بيده: الكلمات الأساسية للغة Keywords، أسماء المتغيرات، أسماء الفئات والدوال والخصائص والوسائل والأحداث.. كل شيء.. لك أن تتخيل كيف كان هذا يضيع وقت المبرمج، بسبب أخطاء الصياغة التي تحدث نتيجة خطئه في تهجئه إحدى الكلمات.. فقبل ظهور الاستشعار الذكي كان على المبرمج أن ينسخ ويلصق أسماء المتغيرات الكثيرة التي يستخدمها في البرنامج حتى لا يخطئ في هجائها، كما كان يضيع معظم الوقت في فتح ملفات الاستعلام لمراجعة أسماء الدوال والخصائص، والتأكد من أنواع معاملات الدوال وترتيبها الصحيح وغير ذلك.. وحتى لو لم يخطئ في الهجاء، فقد كان المبرمج مضطرا إلى كتابة الكثير من الكلمات كاملة كأنه يعمل كاتبا في جلسة محكمة!
وقد كان ظهور الاستشعار الذكي مع الإصدار VB5 مفاجأة مذهلة ومبهجة لكل المبرمجين، ففجأة صارت كتابة الكود متعة، ولم يعد المبرمج مضطرا إلى كتابة كلمة واحدة كاملة!

- فبمجرد كتابة حرفين أو ثلاثة من الكلمة يستطيع المبرمج عرض قائمة تحتوي على كل الكلمات البرمجية وأسماء المتغيرات التي تبدأ بهذه الحروف، ليختار منها الكلمة المناسبة.. ولو كانت هناك كلمة واحدة فقط فلن تظهر القائمة، بل سيكمل الاستشعار الذكي الكلمة تلقائيا ويقوم بكتابتها كاملة مباشرة.. يمكنك أن تجرب هذا بنفسك لو

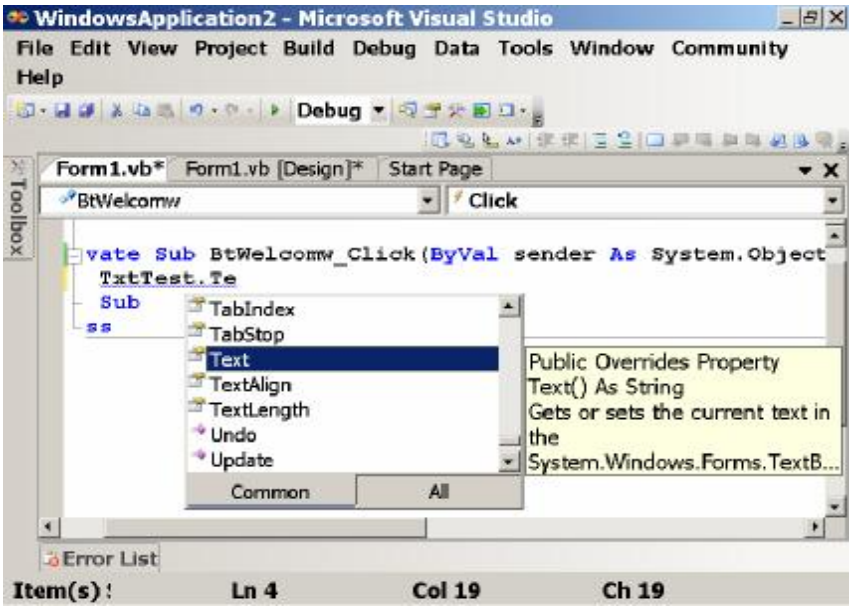
أردت.. في حدث ضغط الزر اكتب في بداية سطر جديد الحرفين tx، ثم اضغط الزر Ctrl مع حرف المسافة من لوحة المفاتيح.. ستجد أن الحرفين tx قد تم إكمالهما تلقائيا إلى الكلمة TxtWelcome.. شيء رائع.. أليس كذلك؟

- يمكنك أيضا أن تؤدي نفس المهمة بضغط الزر Alt مع السهم الأيمن من لوحة المفاتيح.. جرب هذا مثلا بعد كتابة الحرف t بمفرده.. ستجد أن قائمة ستظهر وبها كل الكلمات التي يمكنك استخدامها في برنامجك، وقد تم تحديد أول كلمة تبدأ بحرف t في هذه القائمة.



ويمكنك أن تتحرك في قائمة الاستشعار الذكي باستخدام أسهم لوحة المفاتيح أو بتحريك المنزلق الرأسي Vertical Scroll الموجود في جانب القائمة الأيمن، إلى أن تجد الكلمة المنشودة وتقوم بتحديدها بضغطها مرة واحدة بالفأرة.. أو يمكنك أن تكتب حرفا إضافيا بعد الحرف t ليقل عدد الكلمات التي تختار من بينها.. فكلما كتبت حرفا إضافيا بعد الحرف t ستجد أن الكلمة المحددة تتغير لتصير أول كلمة تبدأ بالحروف التي كتبتها، إلى أن تصل إلى الكلمة التي تريدها.. في هذه اللحظة يمكنك أن تنقر مرتين بالفأرة فوق الكلمة المحددة لكتابتها في الموضع الحالي من الكود، أو يكفي أن تضغط مسافة من لوحة المفاتيح لكتابتها.. كما أن كتابة النقطة . أو = في الموضع الحالي يؤدي إلى كتابة الكلمة المحددة وبعدها النقطة أو = .. أما لو شئت إخفاء هذه القائمة فكل ما عليك هو ضغط زر الإلغاء Esc من لوحة المفاتيح، أو الضغط بالفأرة في أي موضع آخر من صفحة الكود.

- وليس هذا كل شيء.. فعند تحديد أية كلمة في قائمة الاستشعار الذكي، يظهر تلميح على الشاشة يخبرك بصيغة تعريف هذه الكلمة.. وفي الصورة السابقة ترى أن التلميح يوضح أن المتغير TxtWelcome هو من نوع الفئة TextBox.
- وتظهر قائمة الاستشعار الذكي أيضا عند كتابة نقطة . بعد اسم أي كائن Object.. ولأنها ذكية فعلا فإن هذه القائمة ستعرض فقط كل الأعضاء Members الخاصة بهذا الكائن.. والمقصود بالأعضاء هنا كل الخصائص والوسائل والأحداث التي تنتمي إلى هذا الكائن.. لهذا تسمى قائمة الاستشعار الذكي أحيانا بقائمة الأعضاء Members List.. وبكتابة حرف أو حرفين بعد النقطة ستصل إلى العنصر الذي تريده في القائمة ومن ثم يمكنك ضغط مسافة لإكمال اسمه تلقائيا.. هذا مع ظهور معلومات عن تعريف هذا العنصر ترشدك إلى نوعه.. كما أن الأيقونة المعروضة بجوار كل عنصر في القائمة تدل على نوعه، فأيقونة الخاصية تختلف عن أيقونة الوسيلة وعن أيقونة الثوابت... إلخ.



ويمكنك تجربة هذه الإمكانية بكتابة اسم الكائن TxtWelcome متبوعا بنقطة، حيث ستظهر قائمة أعضاء مربع النص.. اكتب بعد هذا الحرف t لتصل إلى أول خاصية تبدأ بهذا الحرف.. لو كتبت بعد هذا الحرف e فسيتم تحديد الخاصية Text.. اكتب = وستجد أن اسم الخاصية Text قد تم إكماله آليا متبوعا بالعلامة = .

- كما ترى: باستخدام الاستشعار الذكي صار المبرمج يكتب نصف الحروف التي كان يكتبها قديما وأحيانا أقل من ذلك بكثير، بالإضافة إلى أنه لم يعد مضطرا إلى حفظ

هجاء أية كلمة، ولم تعد حياته تتحول إلى جحيم بسبب أخطاء الصياغة الناتجة عن أخطاء الهجاء!

- ولا تتوقف قدرات الاستشعار الذكي عند هذا، فهي تقدم للمبرمج معلومات عن تعريف الإجراءات والدوال ومعلومات عن معاملاتها وكيفية استخدامها.. وسنتعرف على هذه الإمكانية بتفصيل أكثر في الفصل القادم.

أنا Me:

في التطبيق Hello كتبنا الكود الذي يعرض جملة الترحيب في مربع النص كالتالي:

TxtWelcome.Text = "مرحبا بك في فيجيوال بيزيك دوت نت"

في الحقيقة، هناك صيغة أخرى لكتابة نفس هذا الكود، وذلك باستخدام الكلمة Me كالتالي:

Me.TxtWelcome.Text = "مرحبا بك في فيجيوال بيزيك دوت نت"

يا ترى: ما معنى كلمة Me هذه؟

كلمة Me تعني أنا، لهذا فهي تشير إلى نسخة الكائن المعرف من الفئة الحالية، التي توجد كلمة Me داخلها.. وبما أننا نكتب الكود داخل الفئة Form1، فإن كلمة Me تشير إلى الكائن المعرف من هذه الفئة، والذي يعرض النموذج Form1 على الشاشة:

```
Public Class Form1
    Private Sub BtWelcomw_Click(ByVal sender As Object, _
            ByVal e As System.EventArgs) _
            Handles BtWelcomw.Click
        Me.TxtWelcome.Text = "مرحبا بك في فيجيوال بيزيك دوت نت"
    End Sub
End Class
```

وبما أن مربع النص موجود على النموذج، إذن فهو عنصر من عناصر النموذج، لهذا يمكن استخدامه من خلال الكلمة Me بوضع نقطة بينهما، كأنه مجرد خاصية من خصائص النموذج.

لكن، لماذا لم نستخدم اسم النموذج Form1 مباشرة بدلا من Me؟

لو لاحظت، فإن النموذج Form1 معرف باعتباره فئة Class.. وقد قلنا من قبل إن الفئة لا تستخدم بنفسها فهي مجرد مخطط لم يتم تنفيذه بعد، لهذا يجب أن يتم تعريف كائنات من هذه الفئة أولا ليمكن التعامل معها من الكود.. لكن المبرمج وهو يكتب كود أية فئة لا يعرف أسماء الكائنات التي سيتم تعريفها من هذه الفئة، لهذا يستخدم الكلمة Me للدلالة على أي كائن معرف من الفئة الحالية.. وعلى كل حال، هذه الكلمة اختيارية، حيث يمكنك كتابتها أو لا، كما رأينا في حالة التعامل مع مربع النص.

إذن فما فائدة الكلمة Me؟

هناك فوائد لهذه الكلمة لا نستطيع التطرق إليها هنا، لهذا سنكتفي بفائدة واحدة تتعلق بالاستشعار الذكي.. فبعض الناس يرون أن استخدام الكلمة Me يوفر الوقت والجهد على

المبرمج، فبمجرد كتابة نقطة بعدها، تظهر قائمة الاستشعار الذكي، وتعرض أعضاء الفئة الحالية ليستطيع المبرمج الاختيار منها، مما يمنع أخطاء الكتابة.. في الحقيقة هذا رأي مشكوك فيه، فقد رأينا كيف يمكننا إجبار قائمة الاستشعار الذكي على الظهور بضغط Ctrl+Space، أو ضغط Alt + السهم الأيمن.. بل إن إصدار دوت نت ٢٠٠٨ طور الاستشعار الذكي بحيث صارت قائمة الاستشعار الذكي تظهر تلقائيا بعد كتابة أي حرف في أي موضع!

لهذا فإن استخدام الكلمة Me في واقع الحال ليس بالأهمية القصوى التي يتصورها البعض، وكثير من المبرمجين لا يستخدمونها إلا في ضرورات أخرى.. الفائدة الوحيدة فقط لاستخدام Me مع الاستشعار الذكي، هي أن عدد العناصر التي تظهر في القائمة يكون أقل بكثير، لأنها في هذه الحالة ستحتوي على أعضاء النموذج الحالي، وليس كل الكلمات البرمجية.

تصميم الأدوات:

تقدم بيئة التطوير للمبرمج تسهيلات رائعة لتصميم الأدوات على النموذج، ليضمن الحصول على أجمل تصميم بأقل مجهود.. وتنقسم هذه التسهيلات إلى ثلاثة أنواع:

١- قائمة التحرير Edit Menu.

٢- خطوط المحاذاة Snap Lines.

٣- قائمة التنسيق Format Menu.

ولكي نجرب معا هذه التسهيلات، دعنا أولا ننشئ مشروعا جديدا New Project ــ بنفس الطريقة التي تعلمناها في هذا الفصل ــ ونعطيه الاسم Alignment.. ضع زرا Button على النموذج وتعال نرى ماذا سنفعل به باستخدام طرق التصميم المختلفة، مع ملاحظة أن ما سنفعله مع الزر يمكن فعله مع أية أداة أخرى.

١- قائمة التحرير Edit Menu:

يمكنك تحديد الزر على النموذج بضغطه مرة واحدة بزر الفأرة، حيث سيظهر حوله إطار التحديد.. هنا يمكنك أن تستخدم معه إمكانيات قائمة التحرير Edit Menu.. لو فتحت هذه القائمة فستجد فيها العديد من الأوامر.. دعنا نتعرف على هذه الأوامر، مع ملاحظة أن هذه الأوامر موجودة أيضا في القائمة الموضعية Context Menu التي تظهر عند ضغط زر الفأرة الأيمن فوق الزر Button1.. كما أن هذه الأوامر توجد أيضا على شريط الأدوات العلوي.. وهذه الأوامر هي:

- **نسخ Copy:**

هذا الأمر سينسخ الزر المحدد إلى لوحة القصاصات Clipboard.. ويمكنك فعل نفس الشيء بضغط Ctrl+C مباشرة من لوحة المفاتيح.

- **لصق Paste:**

استخدم هذا الأمر للصق نسخة جديدة من الزر على النموذج.. ويمكنك فعل نفس الشيء بضغط Ctrl+V مباشرة من لوحة المفاتيح.
لاحظ أن الزر الجديد الذي سيوضع على النموذج، مماثل طبق الأصل للزر الأول في المقاييس واللون والنص المكتوب عليه وكل التفاصيل، وذلك لأن له نفس قيم خصائص الزر الأول، فيما عدا الخاصية Name بالطبع، وذلك لأنه لا يمكن أن يوجد أكثر من أداة بنفس الاسم.
وتتضح أهمية النسخ واللصق عندما تحتاج إلى تصميم نموذج عليه عدد كبير من الأدوات المتشابهة، كالآلة الحاسبة مثلا، فكل أزرار الأرقام متماثلة تماما، ما عدا اختلاف الرقم المكتوب على كل زر.. في هذه الحالة يقوم المبرمج بالخطوات التالية:

أ. تصميم زر واحد فقط وضبط مقاييسه وجميع خصائصه.
ب. نسخ الزر.
ت. لصق أي عدد يريده من النسخ من هذا الزر.
ث. تغيير النص المكتوب على كل زر من الأزرار بعد ذلك.

هذا يوفر على المبرمج الكثير من الوقت والجهد.
لاحظ أنك تستطيع أن تؤدي عملية النسخ واللصق بطريقة أسرع، وذلك بسحب الزر الذي تريد نسخه بالفأرة، مع ضغط الزر Ctrl من لوحة المفاتيح أثناء السحب.. سيؤدي هذا إلى ظهور العلامة + بجوار مؤشر الفأرة، مما يعني أنك تؤدي عملية لصق، ولو رفعت إصبعك عن الزر Ctrl فستختفي هذه العلامة مما يعني أنك تحرّك الزر من موضوعه إلى موضع آخر فحسب.. فإذا كنت تسحب مع ضغط Ctrl، ووصلت إلى الموضع المناسب وتركت زر الفأرة أولا ثم تركت زر Ctrl، فسيتم لصق نسخة من الزر إلى هذا الموضع.. ميزة هذه الطريقة أنها أسرع وأسهل وتتيح لك التحكم في موضع اللصق، كما أنها لا تنسخ الزر إلى لوحة القصاصات، وهذا مفيد إذا كنت تريد الاحتفاظ بشيء آخر نسخته إلى لوحة القصاصات.

- **قص Cut:**

استخدم هذا الأمر لقص الزر المحدد من على النموذج.. ويمكنك فعل نفس الشيء بضغط Ctrl+X مباشرة من لوحة المفاتيح.
لاحظ أن القص يعني نسخ الزر أولا إلى لوحة القصاصات ثم حذفه من على النموذج.. لهذا يمكن إعادة لصق هذا الزر من جديد على النموذج.
وتتضح أهمية القص واللصق عندما تريد نقل بعض الأدوات من نموذج إلى آخر، سواء كان النموذجان في نفس البرنامج أو في برنامجين مختلفين.

- **حذف Delete:**

استخدم هذا الأمر لحذف الزر المحدد من على النموذج.. ويمكنك فعل نفس الشيء بضغط الزر Delete مباشرة من لوحة المفاتيح.

- **تراجع Undo:**

استخدم هذا الأمر للتراجع عن آخر عملية قمت بها.. فمثلا، لو كنت حذفت أحد الأزرار، فيمكنك إعادته مجددا باستخدام هذا الأمر.. ويمكنك فعل نفس الشيء بضغط Ctrl+Z مباشرة من لوحة المفاتيح.

لاحظ أنك تستطيع تكرار عملية التراجع لأكثر من مرة، لاستعادة وضع الأدوات على النموذج قبل آخر أوامر نفذتها.

- **إعادة Redo:**

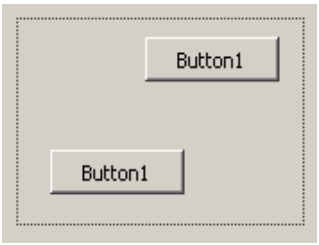
استخدم الأمر Redo لإعادة آخر أمر تراجعت عنه باستخدام الأمر Undo.

لاحظ أنك تستطيع تكرار عملية الإعادة لأكثر من مرة، لاستعادة تأثير كل الأوامر التي تراجعت عنها.

- **تحديد الكل Select All:**

استخدم هذا الأمر لتحديد كل الأدوات الموجودة على النموذج مرة واحدة.. ويمكنك فعل نفس الشيء بضغط Ctrl+A مباشرة من لوحة المفاتيح.

طرق أخرى لتحديد الأدوات:



- تستطيع تحديد أكثر من أداة باستخدام الفأرة، وذلك بضغط زر الفأرة الأيسر والتحرك بالفأرة.. سيؤدي هذا إلى رسم مستطيل التحديد على النموذج.. كل الأدوات التي توجد داخل هذا المستطيل سيتم تحديدها بعد أن تترك زر الفأرة ويختفي المستطيل.. لاحظ أن ضغط الزر ESC أثناء رسم المستطيل يؤدي إلى إخفائه وإلغاء العملية.
- ويمكنك أن تحدد الأدوات بطريقة أخرى، وذلك بضغط الزر Ctrl مع ضغط كل أداة بزر الفأرة الأيسر.. هذه الطريقة تحافظ على تحديد الأدوات المحددة سابقا، وتحدد معها الأداة التي تم ضغطها.. أما لو لم تضغط الزر Ctrl فسيتم تحديد الأداة المضغوطة فقط، وستتم إزالة التحديد من الأدوات الأخرى.. ميزة هذه الطريقة أنها تتيح لك تحديد أدوات بعيدة عن بعضها، وتفصلها أدوات لا تريد تحديها، مما يمنعك من استخدام مستطيل التحديد.

ولإزالة التحديد من جميع الأدوات اضغط الزر Esc من لوحة المفاتيح، أو

اضغط بزر الفأرة فوق أي موضع خال من الأدوات على النموذج.. ولإزالة التحديد من أداة واحدة فقط، اضغط الزر Ctrl واضغط هذه الأداة بزر الفأرة الأيسر.. ولإعادة تحديدها كرر نفس العملية، فهي تعكس حالة التحديد أو عدم التحديد.

الجدير بالذكر أن عمليات القص والنسخ واللصق ستؤثر على جميع الأدوات المحددة.. هذا يعني أنك تستطيع نسخ أكثر من أداة ولصقها دفعة واحدة على النموذج.

ملحوظة:

كل أوامر قائمة التحرير التي نفذناها على الأدوات في مصمم النماذج، يمكن تنفيذها على النصوص المكتوبة في نافذة محرر الكود، تماما كما تفعل في برنامج Notepad و Word وغيرهما.. لقد ركزنا هنا على تحرير الأدوات خصوصا لأن هذا شيء جديد عليك ولم تألفه في برامج أخرى.
ولعلك لاحظتَ أنَّ كل طريق التحرير والتحديد التي شرحناها مماثلة لطرق التعامل مع الملفاتِ والمجلدات في متصفح الويندوز Windows Explorer، ومع النصوص في برامج تحرير النصوص، وغير ذلك.. هذه هي روعة الويندوز، حيثُ إنّ أساليبه قياسيّة، بحيث تستطيع أن تطبّق معظم ما تعرفه في برنامج ما، على باقي البرامج.. لهذا عليك أن تتعامل مع النصوص في محرر الكود تماما كما تتعامل مع النصوص في برنامج Word وغيره، مستفيدا من نفس إمكانيات التحرير والبحث والتحرك بالأسهم وغير ذلك.

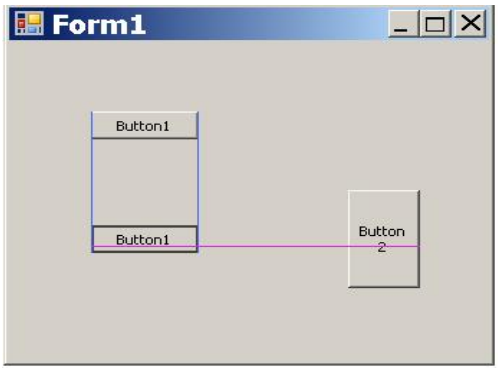
٢ – خطوط المحاذاة Snap Lines:

يمكنك استخدام لوحة المفاتيح للتحكم في موضع وحجم أية أداة محددة على النموذج، وذلك كالتالي:

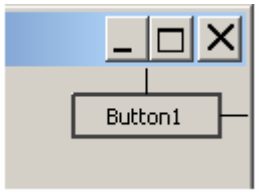
- يمكنك تحريك الأداة على النموذج بتحديدها وضغط أزرار الأسهم من لوحة المفاتيح.. هذا يؤدي إلى تحريك الأداة حركات صغيرة في الاتجاه الذي يشير إليه السهم المضغوط.
- ولو ضغطت زر التحكم Ctrl مع أحد الأسهم فإن هذا يؤدي إلى قفز الأداة قفزة كبيرة لمحاذاتها بأقرب أداة إليها، وإن لم تكن هناك أدوات قريبة فستتم محاذاتها مع حافة النموذج.
- ولو ضغطت زر التحويل Shift مع أحد الأسهم، فسيتم تصغير الأداة أو تكبيرها على حسب اتجاه السهم، فالسهم العلوي يعمل على تصغير الارتفاع، والسفلي يعمل على تكبيره، بينما السهم الأيمن يعمل على تكبير العرض، والأيسر يعمل على تصغيره.

وقد عرفنا من قبل أن تحريك أية أداة على النموذج يتم أيضا بسحبها على النمـــوذج إلى موضع آخر وتركها هناك.. عند استخدام هذه الطريقة يمنحك مصمم النمـــوذج إمكانية رائعة اسمها خطوط المحاذاة Snap Lines:

- فعند تحريك أية أداة على النموذج واقترابها من أية أداة أخرى، تظهر خطـــوط رأسية أو أفقية على النمـــوذج لتصل بـــين الأداة المتحركـــة والأداة المحاذية لها فـــي تلـــك اللحظة.



- وإذا كان عرض الأداة مســـاويا لعرض أداة أخرى، فإن خطين رأسيين يظهران ليشـــيرا إلـــى محاذاة حافتي الأداتين معا.
- إذا كانت محاذاة الأداتين مـــن الحافة فإن الخط الذي يظهر يكون أزرق اللون.. أما إذا كانـــت المحـــاذاة مـــن منتصف الأداتين، فإن الخط يكون أحمر اللون.
- كما تظهر خطوط المحاذاة أيضا أثناء تغيير حجم أي أداة، وذلـــك إذا أدى هـــذا التغيير إلى محاذاة حافة من حوافها مع حافة أداة أخرى.
- وتظهر خطوط المحاذاة كذلك عند استخدام صـــندوق الأدوات لرســـم الأدوات على النموذج.



- كما أن لخطوط المحاذاة وظيفة أخرى، فهي تنبهـــك إلى أفصل هامش تتركه بين الأداة وحافـــة النمـــوذج عندما تتحرك بالأداة قرب حواف النموذج، كما هـــو موضح في الصورة.

واضح طبعا كيف أن مثل هذا التنظيم يسهل عليك محاذاة الأدوات بصريا، بأيســـر وأسرع طريقة.

٣- قائمة التنسيق Format Menu:

توجد بالقوائم الرئيسية العلوية قائمة خاصة لتنسيق الأدوات على النمـــوذج اســـمها Format.. لكي نجرب هذه القائمة، ضع أربعة أزرار على النموذج، وتعال نـــرى كيف ننسقها معا.. لاحظ أنك تستطيع التراجع عن آخر عملية تنســـيق قمـــت بهـــا بالطرق التقليدية للتراجع مثل ضغط Ctrl+Z.. كما يمكنك إعـــادة العمليـــة التـــي تراجعت عنها أيضا باستخدام أمر الإعادة Redo من قائمة التحرير Edit.

الآن، حددّ الأزرار الأربعة على النموذج، وذلك بإحدى الطريقتين اللتين شـــرحناهما سابقا:

أ- اضغط زر الفأرة الأيسر على منطقة خالية من النموذج وتحرّك به دون ترك الزر.. ستجد أنّ إطارًا مستطيلا يرتسم على النموذج، إحدى رءوسِـــه حيـــثُ

بدأتَ الضغط، ورأسه المقابلة حيثُ يوجدُ مؤشّـرُ الفـأرةِ الآن.. حـاولْ أن تتحّركَ بهذا الإطار حتى يحتوي الأزرار الأربعة، ثـمّ اتـرك زرّ الفـأرة.. ستجدُ أنّ الأزرار الأربعَة قد تم تحديدها.

ب- اضغط زر الفأرةِ الأيسر على أحد الأزرار لتحديده، ثم اضـغط الـزر Ctrl من لوحة المفاتيح ولا تتركه، واضغط باقي الأزرار بالفأرة.. بهذه الطريقـة سيتم تحديد كلّ زر تضغطه، بدون إزالة تحديد ما سبقه من أزرار.. لاحـظ أنك لو أردت إزالة تحديد أي زر، فكل ما عليك هو إعادة ضـغطه بالفـأرة مرّة أخرى أثناء ضغط الزر Ctrl.

والآن، دعنا نتعرف على العناصر الموجودة في قائمة التنسيق Format Menu:

- **القائمة الفرعية "محاذاة" Align:**

تحتوي قائمة التنسيق على قائمة فرعية خاصة بمحاذاة الأدوات اسمها Align، تحتوي على عدة أوامر لمحاذاة حواف الأدوات المحددة.. وهذه الأوامر هي:

Lefts	لمحاذاة الحواف اليسرى للأدوات.
Centers	تقوم بمحاذاة مراكز الأدوات (محاذاة كل الأدوات رأسـيا مـن منتصف عرضها).
Rights	لمحاذاة الحواف اليمنى للأدوات.
Tops	لمحاذاة الحواف العليا للأدوات.
Middles	لمحاذاة منتصفات الأدوات (محاذاة كـل الأدوات أفقيـا مـن منتصف ارتفاعها).
Bottoms	لمحاذاة الحواف السفلى للأدوات.
To Grid	لمحاذاة الأدوات إلى شبكة التصميم الوهمية التي تقسم النموذج إلى مربعات صغيرة لأغراض التنسيق.. هذا الأمر غير متـاح في الوضع الافتراضي، ولن يعمـل إلا إذا اختـرت اسـتخدام خطوط الشبكة من خيارات بيئة التطوير.

السؤال المهم الآن هو: أيّ من الأزرار المحددة سيثبت في مكانه، لتتمّ محـاذاة باقي الأدوات عليه؟

لو نظرت إلى النموذج، فستلاحظ أنَّ ثلاثة أزرار توجد على حوافهـا مقـابض سوداء Black Handles، بينما الزر الرابع له مقابض بيضاء.. هذا الزر هـو الأداة المرجعيّة التي ستتمُّ المحاذاة بالنسبة لها، وهي في الغالب أول أداة قمـت بتحديدها.. لكن لو شئت تغييرها، فاضغط بالفأرةِ ضغطة واحدة (بدون ضـغط Ctrl) على الأداةِ التي تريد أن تجعلها مرجعا للمحاذاة، حيـث سـيتغير لـون مقابضها إلى الأبيض وستتم محاذاة باقي الأدوات المحددة بالنسبة لها.

- **القائمة الفرعية "مساواة الأبعاد" Make Same Size:**

تتيح لك هذه القائمة جعل جميع الأدوات المحددة مساوية في أبعادها للأداة المرجعية.. والجدول التالي يلخص أوامر هذه القائمة الفرعية:

Hieght	لجعل الأدوات متساوية في الارتفاع.
Width	لجعل الأدوات متساوية في العرض.
Both	لجعل الأدوات متساوية في كل من العرض والارتفاع معا.
Size to Grid	لتغيير عرض وارتفاع الأدوات لتصبح الأدوات محتواة بالكامل داخل خانات شبكة التصميم.

- **القائمة الفرعية "المسافات الأفقية" Horizontal Spacing:**

تتيح لك هذه القائمة التحكم في المسافات الأفقية بين الأدوات، وهي تحتوي على الأوامر التالية:

Make Equal	يجعل هذا الأمر المسافات الأفقية بين الأدوات متساوية.
Increase	يعمل هذا الأمر على زيادة المسافات الأفقية بين الأدوات.
Decrease	يعمل هذا الأمر على تقليل المسافات الأفقية بين الأدوات.
Remove	يعمل هذا الأمر على إزالة المسافات الأفقية بين الأدوات، بمعنى جعلها متلاصقة.

- **القائمة الفرعية "المسافات الرأسية" Vertical Spacing:**

تتيح لك هذه القائمة التحكم في المسافات الرأسية بين الأدوات، وهي تحتوي على الأوامر التالية:

Make Equal	يجعل هذا الأمر المسافات الرأسية بين الأدوات متساوية.
Increase	يعمل هذا الأمر على زيادة المسافات الرأسية بين الأدوات.
Decrease	يعمل هذا الأمر على تقليل المسافات الرأسية بين الأدوات.
Remove	يعمل هذا الأمر على إزالة المسافات الرأسية بين الأدوات، بمعنى جعلها متلاصقة.

- **القائمة الفرعية "توسيط على النموذج" Center in form:**

تتيح لك هذه القائمة توسيط الأدوات على النموذج، وهي تحتوي على الأمـــرين التاليين:

Horizontally	يتم توسيط الأدوات أفقيا على النموذج.
Virtically	يتم توسيط الأدوات رأسيا على النموذج.

- **الأمر "إغلاق الأدوات" Lock Controls:**

يؤدي ضغط هذا الأمر إلى تثبيت كل الأدوات الموجودة على النموذج.. معنـــى هذا أنك لن تستطيع تغيير حجمها أو موضعها باســـتخدام الفـــأرة فـــي وقـــت التصميم، لكن سيظل بإمكانك التحكم فيها باستخدام نافذة الخصائص.

ويمكنك تنفيذ نفس الأمر بضغط زر الفأرة الأيمن فوق سطح النموذج، واختيار الأمر Lock Controls من القائمة الموضعية Context Menu.

ولفك تثبيت كل الأدوات، اضغط الأمر Lock Controls مجددا، ســـواء مـــن القائمة Format أو من القائمة الموضعية.. لاحظ أن شـــكل القفـــل المرســـوم بجوار الأمر يتغير في كل مرة تضغطه فيها، فحينمـــا تكـــون الأدوات حـــرة الحركة يكون القفل بارزا، وحينما تكون مثبتة يكون القفل غائرا.

ويمكنك أن تجعل بعض الأدوات حرة الحركة وبعضها مثبتة، وذلك باســـتخدام نافذة الخصائص لتغيير قيمة الخاصية Locked الخاصة بكل أداة، فإذا جعلـــت قيمتها True يتم تثبيت الأداة، وإذا جعلت قيمتها False يتم فك تثبيتها.

والآن.. لقد تعرفنا بشكل جيد على البيئة التي نصمم برامجنا ونكتب الكـــود فيهـــا.. الآن نحن جاهزون للتعرف بشكل أفضل على لغة فيجيوال بيزيك.. فهيا بنا.

-٦-

مقدمة إلى الأدوات Controls

سنأخذ في هذا الفصل فكرة عامة على الأدوات، وسنتعرف بشيء من التفصيل على ثلاثة أدوات هامة، وهي:

- **الزر Button:** وهو يسمح للمستخدم بضغطه لتنفيذ مهمة معينة.
- **مربع النص TextBox:** وهو يسمح للمستخدم بالكتابة فيه لإدخال بيانات معينة.
- **اللافتة (المبين) Label:** وهي تعرض نصا للمستخدم، ولا تسمح له بتعديله.. هذا مفيد عند عرض تنبيه معين، أو نتيجة عملية قام بها البرنامج، أو عند عرض عنوان بجوار مربع نص ليشرح وظيفته للمستخدم.

لكن قبل أن نتعرف على هذه الأدوات، دعنا نتعرف على بعض المفاهيم الأساسية التي سنستخدمها ونحن نتعامل مع الأدوات.

التطبيق الفعال Active Application:

يمكن تشغيل أكثر من برنامج في نفس الوقت على نظام تشغيل الويندوز.. لكن برنامجا واحدا منها فقط هو الذي يمكنك التعامل معه باستخدام لوحة المفاتيح.. هذا البرنامج يسمى **التطبيق الفعال Active Application، وهو التطبيق الذي يستقبل الأزرار التي تضغطها من لوحة المفاتيح ويستجيب لها**.. ويمكنك تفعيل أي برنامج بضغط نافذته بالفأرة، أو بالانتقال إليه بضغط Alt+Tab من لوحة المفاتيح.

النموذج الفعال Active Form:

يمكن أن يحتوي كل برنامج على أكثر من نموذج، لكن واحدا منها فقط يمكنه التعامل مع لوحة المفاتيح والاستجابة للأزرار التي يتم ضغطها منها.. هذا النموذج يسمى **النموذج الفعال Active Form، وهو النموذج الذي يستقبل الأزرار التي تضغطها من لوحة المفاتيح ويستجيب لها.**

لاحظ أن النموذج لا يكون فعالا إلا لو كان البرنامج الذي يوجد به فعالا.. افترض أن لدينا برنامجا اسمه "برنامج ١" به نموذجان: "نموذج ١" "ونموذج ٢".. لو كان "نموذج١" فعالا وتم تفعيل برنامج آخر غير "برنامج١"، فإن "نموذج ١" يصير غير فعال، لكن عند تفعيل "برنامج١" مرة أخرى سيعود "نموذج ١" ليصير هو النموذج الفعال.. وسيظل الأمر هكذا إلى أن يتم تفعيل "نموذج ٢".

ويمكنك تفعيل أي نموذج بضغطه بالفأرة، أو بالانتقال إليه بضغط Ctrl+Tab من لوحة المفاتيح.

الأداة الفعالة Active Control:

يمكن أن يحتوي كل نموذج على أكثر من أداة، لكن أداة واحدة منها فقط يمكنها أن تتعامل مع لوحة المفاتيح وتستجيب للأزرار المضغوطة منها.. هذه الأداة تسمى **الأداة الفعالة Active Control أو الأداة المحددة Selected Control، وهي الأداة التي تستقبل الأزرار التي تضغطها من لوحة المفاتيح وتستجيب لها.**

وتختلف الأدوات فيما بينها من حيث إمكانية تفعيلها، وكيفية تعاملها مع لوحة المفاتيح:

- فهناك أدوات لا يمكن تفعيلها على الإطلاق، لأنها لا تتعامل مع لوحة المفاتيح نهائيا، مثل اللافتة Label.
- وهناك أدوات يمكن تفعيلها والتعامل معها من لوحة المفاتيح، لكن دون إمكانية الكتابة فيها، مثل الأزرار.. هذه الأدوات يقال إنها تستقبل المؤشر Focus لهذا عندما تكون فعالة يقال إنها تمتلك المؤشر Focused.. هذا المؤشر يبدو على هيئة إطار سميك يحيط بالأداة، مع وجود إطار متقطع خفيف داخل إطار الأداة ليلفت نظر المستخدم إلى أنها فعالة وبها المؤشر.. والصورة المقابلة توضح أن الزر Button2 هز الزر الفعال وبه المؤشر Focus، بينما الزر Button1 غير فعال وليس به المؤشر.. ولو ضغط المستخدم زر المسافة أو زر الإدخال Enter من لوحة المفاتيح في هذه اللحظة فسيتم ضغط الزر Button2.

- وهناك أدوات يمكن تفعيلها والتعامل معها من لوحة المفاتيح، كما يمكنك الكتابة فيها، مثل مربع النص TextBox والقائمة المركبة ComboBox وغيرهما.. هذه الأدوات تستقبل المؤشر Focus لكن لا يتم رسم إطار حولها، بل يتم وضع علامة ضوئية Caret بها.. هذه العلامة تبدو كخط رأسي يظهر ويختفي في ومضات متتابعة، ليدل المستخدم على موضع الكتابة في الأداة.. وتختفي هذه العلامة الضوئية عندما تصير الأداة غير فعالة (يقال عندئذ إن الأداة فقدت المؤشر Lost Focus).

ويمكنك تفعيل أية أداة قابلة للتفعيل بضغطها بالفأرة، أو بالانتقال إليها بضغط Tab من لوحة المفاتيح.

بعض الخصائص المشتركة بين الأدوات:

توجد الكثير من الخصائص المشتركة بين الأدوات، وهي كما ذكرنا منت قبل موروثة من الفئة الأم لجميع الأدوات Control Class.. لهذا من الأفضل أن نتعلم هذه الخصائص المشتركة مرة واحدة بدلا من أن نكرر شرحها ثلاث مرات أو أكثر!

ومن أهم الخصائص المشتركة بين الأدوات:

الاسم Name:

كما عرفنا من قبل، فإن اسم الأداة هو اسم الكائن الذي يتم تعريفه من فئة هذه الأداة لاستخدامه في البرمجة.

لاحظ أن فيجيوال بيزيك تعطي كل أداة اسما افتراضيا يبدأ باسم الأداة يتبعه رقم ما (١، ٢، ٣ ....).. لهذا إذا وضعت مربع نص على النموذج فستجد اسمه الافتراضي هو TextBox1.. لا ننصحك بإبقاء هذه الأسماء الافتراضية، لأن استخدامها في الكود سيصعب عليك فهمه فيما بعد.. تخيل مثلا لو أن لديك في البرنامج عشرة مربعات نصوص تأخذ الأسماء TextBox11 و TxtBox2 و ..... و TxtBox10.. كيف في نظرك ستتذكر وظيفة كل مربع نص منها وأنت تكتب الكود؟

لهذا من الأذكى أن تعطي كل أداة اسما يدل على وظيفتها.. مثلا، لو كان لديك مربع نص يكتب فيه المستخدم اسمه، فيمكنك تسميته UserName أو User_Name، ولو لديك مربع نص يكتب فيه المستخدم رقمه فيمكنك تسميته UesrID أو User_ID، ولو لديك مربع نص يكتب فيه المستخدم تاريخ ميلاده، فيمكنك تسميته BirthDate أو Birth_Date.

لكن تظل هناك نقطة مهمة عند تسمية الأداة، هي وضع بادئة قبل الاسم تدل على نوع الأداة.. هذه البادئة تتكون من حرفين أو ثلاثة حروف مأخوذة من اسم الأداة، وبهذا يسهل على عين المبرمج تمييزها ومعرفتها بمجرد النظر.. هذا مهم للأسباب التالية:

١- لأنك قد تحتاج إلى تعريف متغيرات بنفس اسم الأداة لتنفيذ وظيفة معينة في الكود.. عليك أن تعرف هنا أنه من غير الممكن تسمية متغير بنفس اسم الأداة في نفس النموذج، لهذا فإن بادئة الأداة ستجعل هناك اختلافا بين اسمها واسم المتغير.. مثلا: الأداة TxtBirthDate تختلف عن المتغير BirthDate.. وواضح من البادئة Txt أن هذه الأداة هي مربع نص.

٢- أو قد توجد أكثر من أداة من أنواع مختلفة تشترك معا في أداء نفس الوظيفة في برنامجك، وأيضا لا يمكن أن نسميها جميعا بنفس الاسم.. هنا تميز البادئة بين كل أداة منها.. مثل: TxtBirthDate و LblBirthDate و BtnBirthDate.

٣- لأن وجود بادئة تدل على نوع الأداة يسهل عليك كتابة الكود الخاص بها، وكذلك فهم الكود بعد ذلك عند قرائته لتصحيحه أو تطويره.

لكل هذا، من الأفضل أن تسمي مربعات النصوص السابق ذكرها TxtUserName و TxtUserID و TxtBirthDate.. لاحظ أن كل مقطع في اسم المتغير يبدأ بحرف كبير Capital ليسهل على العين تمييز الكلمات المتلاصقة المكونة للاسم.. هذا أكثر اختصارا من وضع شرطة منخفضة _ بين أجزاء الكلمات.

ملحوظة:

أحيانا يضع المبرمج على النموذج أدوات للعرض فقط، حيث يضبط خصائصها مرة واحدة في وقت التصميم باستخدام نافذة الخصائص، ولا يحتاج إلى كتابة أي كود يتعامل معها بعد ذلك.. يحدث هذا كثيرا مع اللافتات Labels التي يستخدمها المبرمج لعرض نص يشرح وظيفة أدوات أخرى على النموذج.. في هذه الحالة لا داعي لأن يرهق المبرمج نفسه بتغيير أسماء هذه الأدوات، فأسماؤها لن تستخدم في الكود أبدا.. بل إن المبرمج قد يمنع تعريف متغير يحمل اسم هذه الأداة في البرنامج أصلا، وذلك باستخدام نافذة الخصائص لتغيير قيمة خاصية اسمها GenerateMember بمعنى "إنشاء متغير للأداة".. هذه الخاصية لها القيمة الافتراضية True، ولو غيرها المبرمج إلى القيمة False فلن يتم إنشاء متغير للأداة داخل النموذج، وبهذا يصير من المستحيل التعامل معها من الكود!

والآن دعنا نتفق على استخدام البادئات التالية لأسماء الأدوات في هذا الكتاب:

الأداة	البادئة	مثال للاسم
Form	Frm	FrmHello
TextBox	Txt	TxtCity
Label	Lbl	Lb1Result
Button	Btn	BtnOk
ListBox	Lst	LstCountries
ComboBox	Cmb	CmbJobs

والآن ابدأ مشروعا جديدا بنفس الطريقة التي تعلمناها في الفصل السابق، وأطلق عليه الاسم ControlsIntro.. ضع على النموذج مربع نص TextBox ولافتة Label وزرا Button، وغير أسماءها كما هو موضح في الجدول:

الأداة	الاسم Name
مربع النص	TxtTest
اللافتة	LblTest
الزر	BtnTest

كما ترى: أسمينا كل الأدوات Test لأنها أدوات سنستخدمها لاختبار الخصائص المشتركة بين الأدوات.. ولم يميز بين اسم كل أداة والأخرى إلا البادئة الخاصة بكل منها، والتي تدل على نوعها.

نسق الأدوات الثلاث لتبدو كما في الصورة:

لفعل هذا اتبع الخطوات التالية:

١- اضغط Ctrl+A من لوحة المفاتيح لتحديد كل الأدوات الموجودة على النموذج.
٢- افتح القائمة Format ثم القائمة الفرعية Center in form ومنها اضغط الأمر Horizontally.. هذا سيضع كل أداة في منتصف النموذج أفقيا.
٣- كرر الخطوة السابقة، لكن اختر الأمر Vertically لوضع كل أداة في منتصف النموذج رأسيا.
٤- افتح القائمة Format ثم القائمة الفرعية Vertical Spacing ومنا اضغط الأمر Make Equal.. هذا سيجعل المسافة الرأسية بين الأدوات متساوية.

النص Text:

تغير هذه الخاصية النص المعروض على الأداة.. لاحظ أن النموذج يعرض النص الخاص به على شريطه العلوي، بينما يظهر النص فوق اللافتة والزر، ويظهر داخل منطقة الكتابة في مربع النص.

دعنا نعرض النصوص التالية على أدواتنا:

الأداة	النص Text
مربع النص	تمكنك الكتابة هنا
اللافتة	مرحبا بك
الزر	اضغطني
النموذج	نموذج اختبار

مرئية Visible:

القيمة الافتراضية لهذه الخاصية True، فإذا جعلتها False وشغلت البرنامج، فلن ترى الأداة على سطح النموذج.. فائدة هذه الخاصية إخفاء الأدوات في بعض الأوقات وإظهارها عند الحاجة.

فعّالة Enabled:

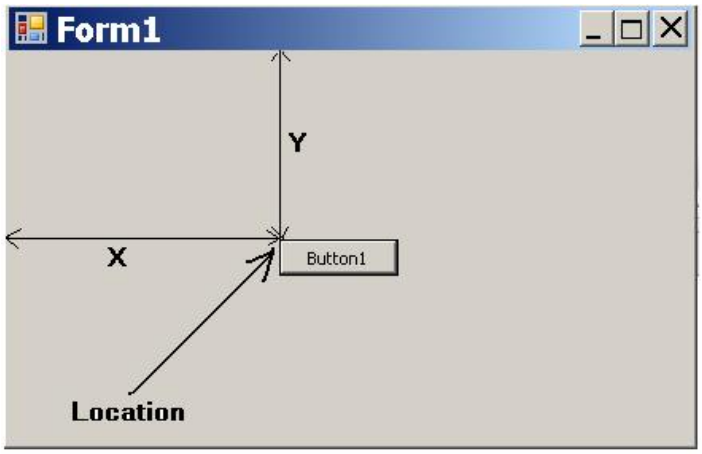
القيمة الافتراضية لهذه الخاصية هي True، وهذا يعني أن الأداة فعالة Enabled، أي أن المستخدم يستطيع التعامل معها بواسطة الفأرة أو لوحة المفاتيح.. أما لو جعلت قيمة هذه الخاصية False فسيكتسب لون الأداة تأثيرا رماديا، كأنما صارت فوقها طبقة عازلة تمنع المستخدم من التعامل معها.. في هذه الحالة تكون الأداة غير فعالة أو معطلة Disabled.. جرب أن تعطل الزر ومربع النص بأن تجعل للخاصية Enabled الخاصة بكل منها القيمة False، ثم شغل البرنامج.. حاول أن تضغط الزر بالفأرة.. ستجد أنه لا ينضغط ولا يستجيب لك.. حاول أن تضغط داخل مربع النص لتضع فيه مؤشر الكتابة.. ستجده يرفض ذلك مما يعني أنك لن تستطيع الكتابة فيه أو نسخ النص الموجود فيه حاليا.

الموضع Location:

تمثل هذه الخاصية موضع الركن العلوي الأيسر للأداة على النموذج.. هذا الموضع يتم تعريفه بواسطة معلومتين:

- X : وهي تمثل الموضع الأفقي للأداة (على المحور س).. بمعنى آخر، تمثل المسافة من بداية النموذج إلى حافة الأداة اليسرى.
- Y : وهي تمثل الموضع الرأسي للأداة (على المحور ص).. بمعنى آخر، تمثل المسافة من قمة النموذج إلى حافة الأداة العليا.

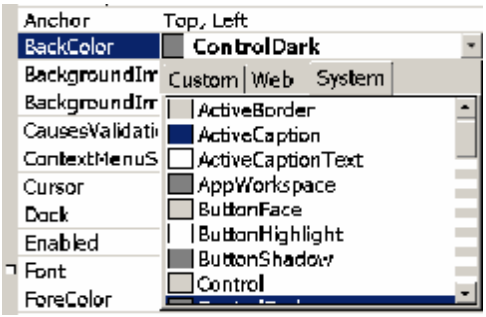
ويمكنك كتابة هاتين القيمتين من نافذة الخصائص بضغط العلامة "+" المجاورة لخاصية الموضع Location، ووضع القيم التي تريدها في الخاصيتين الفرعيتين X و Y.. هذا يتيح لك دقة أكبر في التحكم في موضع الأداة على النموذج.



الحجم Size:

تمثل هذه الخاصية عرض الأداة Width وارتفاعها Height، ويمكنك كتابة هاتين القيمتين من نافذة الخصائص بضغط العلامة "+" المجاورة لاسم الخاصية.

لون الخلفية BackColor:



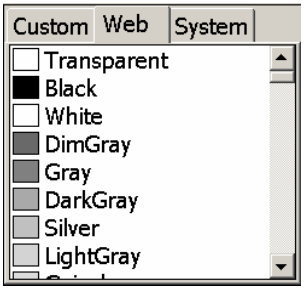
تمكنك هذه الخاصية من اختيار لون خلفية الأداة.. لاحظ أن خانة القيمة لهذه الخاصية يوجد بها زر صغير عليه سهم يشير إلى الأسفل.. اضغط هذا الزر لعرض نافذة اختيار اللون.

هذه النافذة تتيح لك اختيار اللون بثلاث طرق مختلفة، بواسطة أشرطة الاختيار Tabs العلوية، وهي:

١- ألوان النظام System:

يعرض لك هذا الشريط ألوان مكونات نوافذ الويندوز، مثل ألوان النصوص والعناوين والأشرطة وغيرها.. لاحظ أن كل عنصر في هذا الشريط موجود بجواره مربع صغير يوضح لونه.. هذه الألوان غير ثابتة، بل يستطيع المستخدم تغييرها من نظام الويندوز، سواء بتغيير السمات Themes أو بتغيير مظهر الويندوز Appearance.

لاحظ أن الميزة في جعل مظهر برنامجك يعتمد على ألوان النظام، هي ترك الحرية لمستخدم برنامجك في الحصول على الألوان التي تناسبه تبعا لاختياراته هو لألوان الويندوز، والتي يستطيع تغييرها في أية لحظة، مما يعني أنك لن تفرض ذوقك الخاص على مستخدمي برنامجك.



٢- ألوان صفحات المواقع Web:

يعرض هذا الشريط مجموعة كبيرة من الألوان بأسمائها الإنجليزية، وبجوار كل اسم مربع صغير يعرض اللون.

٣- ألوان مخصصة Custom:

يتيح لك هذا الشريط اختيار اللون المفضل لك من قائمة من درجات الألوان المختلفة، كما هو موضح بالصورة.

أيا كانت الطريقة التي تفضلها، فسيتم إغلاق نافذة اختيار اللون فور أن تضغط اللون بالفأرة، حيث ستتلون خلفية الأداة باللون الذي اخترته.

لاحظ أنك تستطيع استعادة اللون الافتراضي للأداة بضغط زر الفأرة الأيمن فوق اسم الخاصية وضغط الأمر "استعادة الوضع الأصلي" Reset من القائمة الموضعية Context Menu.

لون الكتابة (لون الحروف) ForeColor:

تتيح لك هذه الخاصية تغير لون الكتابة (لون النص الذي تعرضه الأداة).. ويمكنك اختيار اللون بنفس الكيفية التي أوضحناها في الخاصية السابقة.

والآن، دعنا نعطي الأدوات الألوان التالية:

الأداة	لون الخلفية BackColor	لون الكتابة ForeColor
مربع النص	أزرق Blue	أبيض White
اللافتة	أحمر Red	أصفر Yellow
الزر	برتقالي Orange	أخضر Green
النموذج	ذهبي Gold	(بلا تأثير)

لاحظ أن خاصية لون الكتابة ForeColor لا تغير لون النص المكتوب على شريط النموذج، لهذا فهي بلا تأثير على النموذج.. ولا يظهر تأثير هذه الخاصية إلا في حالة واحدة، وذلك عندما يكون على النموذج بعض الأدوات التي لم تقم أنت بتغييـر لونها الافتراضي بعد، ففي هذه الحالة ستأخذ جميعها لون الكتابة الذي أعطيته للنموذج.

صورة الخلفية BackGroundImage:

تتيح لك هذه الخاصية وضع صورة في خلفية الأداة.. لتجربة هذا، حدد الزر BtnTest، وافتح نافذة الخصائص، واختر الخاصية BackGroundImage.. اضغط زر الاختيار الموجود في خانة القيمة لهذه الخاصية.. ستظهر لك نافذة اختيار الصورة.

تقدم لك هذه النافذة طريقتين لحفظ الصورة:

١- حفظ الصورة كمصدر محلي Local Resource ضمن النموذج الذي وضعت عليه الأداة.

٢- حفظ الصورة كمصدر عام للمشروع Project Resource، حيث يمكن لأية أداة على أي نموذج استخدام الصور المحفوظة في هذا المصدر، وبهذا لا يتم تكرار حفظ نفس الصورة في أكثر من نموذج، مما يوفر مساحة التخزين.

وبغض النظر عن الطريقة التي ستستخدمها، فإن الصور التي ستضعها على النموذج أو الأدوات بهذه الطريقة سيتم تضمينها داخل الملف التنفيذي Exe. للبرنامج عند ترجمة الكود Compilation.. لهذا ميزة، وهي أنك لا تحتاج إلى القلق بخصوص نقل الصور مع البرنامج إلى جهاز المستخدم، فهي بالفعل موجودة داخل ملف البرنامج نفسه.. وإن كان هذا في المقابل يؤدي إلى كبر حجم الملف التنفيذي، تبعا لحجم الصور والمعلومات المضمنة بداخله.

اضغط بزر الفأرة فوق الكلمة Local resource لاختيار حفظ الصورة ضمن مصادر النموذج.. سيؤدي هذا إلى تفعيل زر الاستيراد Import الموجود تحت هذه الجملة مباشرة.. اضغط هذا الزر لعرض مربع حوار فتح ملف، حيث يمكنك تصفح محتويات جهازك واختيار الصورة التي تريدها.

بعد اختيار الصورة اضغط زر الفتح Open.. سيؤدي هذا إلى عرض الصورة في قسم المعاينة على يمين النافذة.. هذه هي الصورة التي سيتم وضعها في خلفية الأداة.. يمكنك أن تغيرها في أية لحظة بإعادة ضغط زر الاستيراد واختيار صورة أخرى.. كما يمكنك حذف الصورة نهائيا من خلفية الأداة بضغط زر المحو Clear.

والآن بعد أن اخترت الصورة المناسبة، اضغط زر الموافقة OK لإغلاق النافذة.. لاحظ أن زر الإلغاء Cancel يؤدي أيضا إلى إغلاق النافذة، لكنه يلغي أية خطوات قمت باتخاذها بعد أن فتحت النافذة.. هذا يعني عدم وضع الصورة في خلفية الأداة.
الآن سترى الصورة التي اخترتها وقد ظهرت على خلفية الأداة في وقت التصميم Design Time، ولو شغلت البرنامج (بضغط الزر F5) فسترى أنها موجودة أيضا في خلفية الأداة في وقت التشغيل Run Time.
أوقف تشغيل البرنامج، وحدد الزر BtnTest مجددا، ثم انتقل إلى نافذة الخصائص، واختر الخاصية BackGroundImage.. ستلاحظ أن خانة القيمة تحتوي على النص System.Drawing.Bitmap.. يمكنك تحديد هذا النص وضغط الزر Delete من لوحة المفاتيح لحذف الصورة الموجودة في خلفية الزر.

تنسيق صورة الخلفية BackGroundImageLayout:
تتيح لك هذه الخاصية اختيار طريقة عرض الصورة داخل الأداة، حيث تقبل إحدى القيم التالية:

بدون تأثير None	يتم عرض الصورة كما هي على الأداة بدون أي تغيير، حيث ستظهر الصورة بدءا من الركن العلوي الأيسر للأداة، وإن زاد حجمها عن حجم الأداة فلن يظهر الجزء الزائد على الأداة. 
توسيط Center	يتم توسيط الصورة داخل حدود الأداة. 
مطّ Stretch	يتم مطّ الصورة لتملأ كامل مساحة الأداة.. ولو كانت الصورة أكبر من مساحة الأداة، فسيتم تصغيرها لتتلائم مساحة الأداة.. عيب هذه الطريقة أنها في الغالب تشوه الصورة بسبب عدم التناسب في مقاييسها عند تكبيرها أو تصغيرها. 

تحجيم Zoom	مشابهة لطريقة العرض السابقة، إلا أن الصورة في هذه الطريقة يتم تكبيرها أو تصغيرها مع المحافظة على التناسب بين عرضها وارتفاعها.. هذا يعني أن الصورة في معظم الأحيان ستأخذ طول الأداة فقط أو ارتفاعها فقط، بينما قد يتبقى جزء من أرضية الأداة لم تتم تغطيته. 
تبليط Tile	هذه هي القيمة الافتراضية، حيث يتم ملأ مساحة الأداة بالصورة، لكن بدون مطها، بل بتكرار الصورة عدة مرات أفقيا ورأسيا إذا استلزم الأمر ذلك.. تماما كأن الصورة قطعة بلاط يتم كسوة الأرضية بها. 

مؤشر الفأرة Cursor:

تتيح لك هذه الخاصية اختيار شكل مؤشر الفأرة عند مرور الفأرة فوق الأداة.. ويمكنك اختيار المؤشر الذي يناسبك من القيم التالية:

AppStarting	مؤشر بدء تشغيل البرنامج	
Arrow	سهم	
Cross	صليب	
Default	المؤشر الافتراضي.. يختار المستخدم شكل هذا المؤشر من خيارات الفأرة من لوحة التحكم في نظام الويندوز.. في الغالب يكون المؤشر الافتراضي هو السهم.	
IBeam	حرف I	
No	ممنوع	
SizeAll	تغيير الحجم في جميع الاتجاهات	
SizeNESW	تغيير الحجم في اتجاهي الشمال الشرقي والجنوب الغربي	
SizeNS	تغيير الحجم في اتجاهي الشمال والجنوب	
SizeNWSE	تغيير الحجم في اتجاهي الشمال الغربي والجنوب الشرقي	
SizeWE	تغيير الحجم في اتجاهي الغرب والشرق	
UpArrow	سهم للأعلى	

WaitCursor	مؤشر انتظار انتهاء التنفيذ	
Help	استعلام (مساعدة)	
HSplit	فاصل أفقي	
VSplit	فاصل رأسي	
NoMove2D	ممنوع الحركة في الاتجاهين	
NoMoveHoriz	ممنوع الحركة أفقيا	
NoMoveVert	ممنوع الحركة رأسيا	
PanEast	توجه إلى الشرق	
PanNE	توجه إلى الشمال الشرقي	
PanNorth	توجه إلى الشمال	
PanNW	توجه إلى الشمال الغربي	
PanSE	توجه إلى الجنوب الشرقي	
PanSouth	التوجه إلى الجنوب	
PanSW	التوجه إلى الجنوب الغربي	
PanWest	التوجه إلى الغرب	
Hand	يد	

تعال نجرب هذه الخاصية.. جرب مثلا جعل المؤشر على شكل يد Hand فوق الزر.. اضغط F5 لتشغيل البرنامج، وتحرك بالفأرة فوق النموذج.. ستجد أن شكل مؤشر الفأرة هو السهم العادي طالما كنت فوق سطح النموذج، وعندما تحلق فوق سطح الزر فستجد شكل المؤشر قد تحول إلى يد.

من اليمين إلى اليسار RightToLeft:

هذه الخاصية موجودة من أجلنا نحن المستخدمين العرب، فهي تجعل النص المعروض في اللافتة ومربع النص يظهر من اليمين إلى اليسار ليكون مناسبا للغة العربية.. ولهذه الخاصية ثلاثة قيم ممكنة، وهي:

لا No	يظهر النص من اليسار إلى اليمين.
نعم Yes	يظهر النص من اليمين إلى اليسار.
موروثة Inherit	تأخذ هذه الخاصية نفس القيمة الموضوعة بخاصية RightToLeft في النموذج أو الأداة الحاوية للأداة الحالية.

ويمكنك تجربة هذه الخاصية مع مربع النص.. سيسعدك أخيرا أن يظهر النص العربي من اليمين إلى اليسار كما اعتدنا عليه.

الخط Font:

تسمح لك هذه الخاصية باختيار الخط الذي سيظهر به النص على أي أداة.. ولتغيير قيمة هذه الخاصية من نافذة الخصائص لديك ثلاث طرق:

١- الطريقة الأولي هي أن تكتب معلومات الخط في خانة القيمة يدويا.. مبدئيا، ستجد في خانة القيمة النص "Tahoma, 8pt".. هذا يعني استخدام خط اسمه تاهوما Tahoma حجمه أو سمكه ٨ نقاط، حيث pt هي اختصار الكلمة Points.. ويمكنك أن تغير اسم الخط مثلا إلى "الخط العربي المبسط" Simplified Arabic وحجمه إلى ١٠ نقاط، بوضع النص التالي في خانة القيمة: "Simplified Arabic, 10pt".. لكني لا أنصحك باستخدام هذه الطريقة فهي أصعب وتعرضك للخطأ، كأن تخطئ في كتابة اسم الخط مثلا، ناهيك عن ضرورة حفظ أسماء الخطوط، وهو عبء لا ضرورة له.

٢- الطريقة الثانية هي أن تضغط العلامة "+" المجاورة لكلمة Font.. سيؤدي هذا إلى إسدال خصائص كائن الخط، مثل:

أ. **خاصية اسم الخط Name.**

في خانة القيمة لهذه الخاصية سترى زرا صغيرا عليه سهم للأسفل.. لو ضغطت هذا الزر فستظهر قائمة منسدلة فيها أسماء كل الخطوط المتاحة على جهازك، وبجوار كل خط صورة لحرف مكتوب بهذا الخط على سبيل المعاينة السريعة.. كل ما عليك هو أن تختار الخط الذي يعجبك من هذه القائمة.

ب. **خاصية حجم الخط Size:**

اكتب في هذه الخانة رقما يدل على حجم الخط الذي تريده.. الأحجام الصغيرة تنحصر بين ٨ و ١٢ نقطة، والأحجام العادية تنحصر بين ١٤ و ٢٠ نقطة، والأحجام الكبيرة تنحصر بين ٢٢ إلى ٧٢ نقطة.

ج. **خط سميك Bald:**

هذه الخاصية تأخذ إحدى قيمتين: True لجعل الخط **سميكا**، وFalse لجعله خطا عاديا (غير سميك).

د. **خط مائل Italic:**

هذه الخاصية تأخذ إحدى قيمتين: True لجعل الخط *مائلا*، وFalse لجعله خطا عاديا (غير مائل).

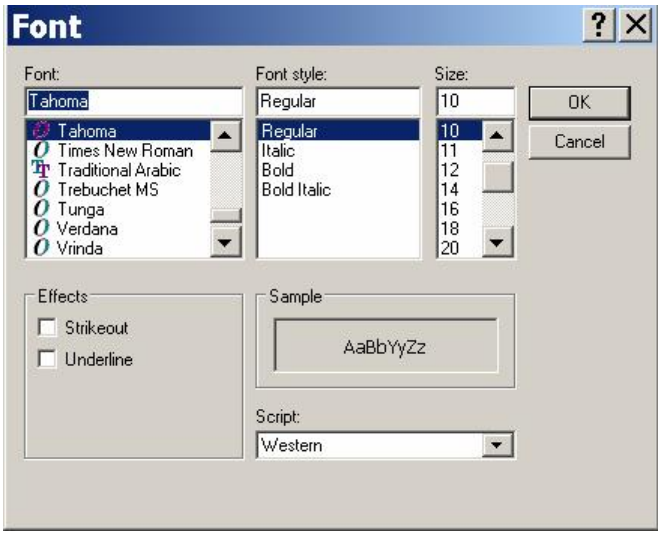
ه. **يعبره خط Strikout:**

هذه الخاصية تأخذ إحدى قيمتين: True لرسم خط يعبر النص، وFalse لجعله خطا عاديا.

و. **تحته خط UnderLine:**

هذه الخاصية تأخذ إحدى قيمتين: True لوضع خط تحت النص، وFalse لجعله خطا عاديا.

٣- الطريقة الثالثة ــ وهي الأسهل ــ هي ضغط زر الاختيار الموجود في خانة القيمة للخاصية Font نفسها.. سيؤدي هذا إلى ظهور مربع حوار اختيار الخط الموضح في الصورة:



تقدم لك هذه النافذة طريقة مرئية لاختيار الخط ومعاينة تأثير اختياراتك على النص المكتوب في قسم المعاينة Sample.

تعال نتعرف على أجزاء هذه النافذة:

أ. **حجم الخط Size:**

تستطيع كتابة الرقم الذي تريده أن يكون حجما للخط في مربع النص مباشرة، أو اختياره من القائمة الموجودة أسفله.

ب. **طراز الخط Font Style:**

تستطيع كتابة الطراز الذي تريده للخط في مربع النص مباشرة، أو اختياره من القائمة الموجودة أسفله.. تقدم لك هذه القائمة الخيارات التالية:

Regular	خط عادي (ليس سميكا ولا مائلا).
Italic	خط *مائل*.
Bold	خط **سميك**.
Italic Bold	خط ***سميك مائل***.

ج. **الخط Font:**

تستطيع كتابة اسم الخط الذي تريده في مربع النص مباشرة، أو اختياره من القائمة الموجودة أسفله، والتي تعرض لك كل أسماء الخطوط المتاحة على جهازك.

د. تأثيرات الخط Effects:

يقدم لك هذا القسم خيارين، حيث يمكنك الضغط بالفأرة فوق مربع الاختيار Check Box الموجود على يسار كل اختيار لوضع العلامة ✓ أو إزالتها.. وهذان الخياران هما: "تحته خط" Underline و"يعبره خط" ~~Strikeout~~.. ولا يوجد ما يمنع من اختيار التأثيرين معا.. سيبدو هذا التأثير كما في الكلمة التالية: ~~مثال~~.

ه. لغة المستند Script:

تتيح لك هذه القائمة اختيار اللغة التي ستكتب الخط بها.. هذا الخيار يؤثر على جزء المعاينة Sample، حيث يعرض الحروف المناسبة للغة التي تستخدمها.. مثلا: لو اخترت اللغة العربية Arabic من القائمة فسيظهر النص أبجدهوز aAbB في مربع المعاينة، مما يتيح لك رؤية تأثير اختياراتك بصورة أفضل على النصوص العربية.

والآن، بعد أن تنتهي من اختيار الخط وتأثيراته ومعاينتها، اضغط زر الموافقة OK لتنفيذ ذلك، أو زر الإلغاء Cancel لإهمالها.

يمكنك الآن أن تجرب أنواعا مختلفة من الخطوط لكل من اللافتة ومربع النص والزر، باستخدام الطريقة التي تناسبك من هذه الطرق.

الأحداث Events:

تعتمد البرمجة في نظام تشغيل الويندوز على الأحداث Events.. هنا يجب أن نتعرف على ثلاثة مفاهيم أساسية:

١ – سبب الحدث:

هو التصرف الذي قام به المستخدم بواسطة الفأرة أو لوحة المفاتيح.

٢ – إطلاق الحدث Firing the Event:

هو محاولة الأداة أن تخبر المبرمج بالحدث الذي تسبب فيه المستخدم.

٣ – الاستجابة للحدث (معالجة الحدث) Handling the Event:

هي محاولة المستخدم للرد على الحدث.. في كثير من الأحيان يتجاهل المستخدم انطلاق الحدث لأنه غير مهتم به، لهذا لا يحدث شيء في البرنامج.. مثلا: لو شغلت البرنامج ControlsIntro وجربت أن تضغط الزر BtnTest فلن يحدث شيء.. السبب في هذا أن هناك ضلعا ناقصا من المثلث.. فضغط المستخدم للزر هو الفعل الذي قام به المستخدم وسبب الحدث.. هذا الحدث جعل الزر يقوم **بإطلاق حدث** اسمه Click، لكن المبرمج لم يستجب لهذا الحدث، فنحن لم نكتب أي كود يفعل هذا حتى الآن!

إذن، لكي يكون هناك تأثير للحدث، لا بد أن يكتب المبرمج الكود الذي يستجيب له.. هذا الكود يسمى معالج الحدث أو المستجيب للحدث Event Handler.

دعنا إذن نرى كيف يمكنك كتابة هذا المستجيب.. أبسط طريقة لفعل هذا، هي النقر المزدوج بالفأرة على الزر في وضع التصميم.. سيؤدي هذا إلى فتح نافذة الكود، وقد تمت كتابة المستجيب للحدث فيها آليا.. هذا المستجيب يسمى إجراء Sub وله التعريف التالي:

```
Private Sub BtnTest_Click(ByVal sender As Object, _
        ByVal e As EventArgs) Handles BtnTest.Click

End Sub
```

يمكنك الآن أن تكتب أي كود تريده داخل هذا الإجراء، ليتم تنفيذه عندما يضغط المستخدم الزر.. لكن قبل أن تفعل هذا، لاحظ ما يلي:

١- الإجراء Sub: هو مقطع من الكود له وظيفة محددة، يتم تعريفه بسطر بداية فيه الكلمة Sub، وسطر نهاية فيه الجملة End Sub.

٢- لكل إجراء اسم خاص به.. هذا الاسم يوجد بعد الكلمة Sub مباشرة، وهو هنا BtnTest_Click.. واضح أن هذا الاسم مكون من مقطعين تفصلهما الشرطة المنخفضة _ .. هذا المقطعان هما اسم الزر BtnClick واسم الحدث Click.. ولا يوجد ما يمنعك من تغير هذا الاسم إلى أي اسم آخر تريده، مثل BtnClick أو غير ذلك.

٣- لكي يكون الإجراء متسجيبا لحدث ما، فلا بد أن توجد في نهاية سطر تعريفه الكلمة "يستجيب" Handles، متبوعة باسم الحدث الذي سيستجيب له الإجراء.. ولا بد أن ينسب الحدث إلى الكائن الخاص به، بوضع نقطة بينهما.. وفي مثالنا هذا تم استخدام الجملة التالية لجعل الإجراء يستجيب لحدث ضغط الزر:

```
Handles BtnTest.Click
```

نريد الآن أن نكتب شيئا في هذا الإجراء.. دعنا مثلا نكتب أي جملة في مربع النص، مثل:

```
TxtTest.Text = "لقد قمت بضغط الزر"
```

لو كتبت هذه الجملة داخل مقطع الإجراء (قبل السطر End Sub) وشغلت البرنامج وضغطت الزر، فستظهر في مربع النص العبارة: "لقد ضغطت الزر".

وهكذا تكون أضلاع المثلث قد اكتملت: سبب الحدث ــ إطلاق الحدث ــ الاستجابة للحدث.

وتوجد العديد من الأحداث المشتركة بين الأدوات.. دعنا نلقي نظرة سريعة عليها بدون الغوص في التفاصيل.. هذه الأحداث تنقسم إلى أربعة أنواع رئيسية، وهي:

١- أحداث الفأرة:

هذه الأحداث تنطلق نتيجة حركة الفأرة أو ضغط أحد أزرارها أو كليهما معا.. والجدول التالي يلخص هذه الأحداث:

ضغط الفأرة MouseClick	ينطلق هذا الحدث بعد ضغط زر الفأرة فوق الأداة وترك الزر.
النقر المزدوج بالفأرة MouseDoubleClick	ينطلق هذا الحدث عند نقر الأداة مرتين سريعتين بالفأرة.
هبوط زر الفأرة MouseDown	ينطلق هذا الحدث بمجرد ضغط زر الفأرة دون انتظار تركه لهذا فهو يسبق انطلاق الحدث MouseClick.
صعود زر الفأرة MouseUp	ينطلق هذا الحدث بعد ترك زر الفأرة المضغوط، وهو يلي انطلاق الحدث MouseClick.
تحرك عجلة الفأرة MouseWheel	ينطلق هذا الحدث عند تحريك عجلة الفأرة.
دخول الفأرة MouseEnter	ينطلق هذا الحدث عندما يعبر مؤشر الفأرة من خارج الأداة إلى داخلها، وهو ينطلق مرة واحدة فور دخول مؤشر الفأرة، ولا ينطلق مرة أخرى حتى يغادرها ثم يدخلها من جديد.
مغادرة الفأرة MouseLeave	ينطلق هذا الحدث عندما يخرج مؤشر الفأرة عن حدود الأداة.
تحرك الفأرة MouseMove	ينطلق هذا الحدث أثناء حركة الفأرة فوق الأداة، وهو ينطلق مرة كل حوالي ١٢٠ ملي ثانية (الملي = ١ / ١٠٠٠)، أي أنه ينطلق حوالي ٨ مرات في الثانية الواحدة.
تحليق الفأرة MouseHover	ينطلق هذا الحدث عند سكون مؤشر الفأرة وتوقفه عن الحركة فوق الأداة لفترة زمنية قصيرة.
حدث استقبال المؤشر GotFocus	ينطلق عندما يضغط المستخدم الأداة بالفأرة فتصير فعالة Focused.
حدث فقد المؤشر LostFocus	ينطلق عندما يضغط المستخدم أداة أخرى بالفأرة فتصير الأداة الحالية غير فعالة بانتقال المؤشر Focus منها إلى الأداة التي تم ضعطها.

٢- أحداث لوحة المفاتيح:

تنطلق هذه الأحداث بسبب ضغط المستخدم زرا أو أكثر من لوحة المفاتيح Keyboard.. وهي:

هبوط الزر KeyDown	يحدث عندما يضغط المستخدم أي زر من لوحة المفاتيح.
ضغط الزر KeyPress	يحدث عندما يضغط المستخدم أزار الأرقام والحروف الهجائية من لوحة المفاتيح.
صعود الزر KeyUp	يحدث عندما يرفع المستخدم إصبعه عن الزر المضغوط.

٣- أحداث يمكن أن تحدث بالفأرة أو لوحة المفاتيح:

يمكن أن تنطلق هذه الأحداث بسبب استخدام الفأرة أو لوحة المفاتيح.. وهذه الأحداث هي:

حدث الدخول Enter	ينطلق هذا الحدث عندما تستقبل الأداة المؤشر Focus وتصير نشيطة Active.. يحدث هذا عند ضغط الأداة بالفأرة، أو ضغط زر الجدولة Tab من لوحة المفاتيح للانتقال من أداة إلى أخرى.
حدث الخروج Leave	ينطلق هذا الحدث عندما يغادر المؤشر الأداة، وتصبح غير نشيطة.. سواء بسب ضغط أداة أخرى بالفأرة، أو ضغط زر الجدولة Tab من لوحة المفاتيح للانتقال إلى أداة أخرى.
الضغط Click	ينطلق هذا الحدث نتيجة ضغط المستخدم زر الفأرة فوق الأداة.. ولو كان هناك زر على النموذج، وكان فعالا (يمتلك المؤشر Focus) فيمكن ضغطه باستخدام لوحة المفاتيح، بضغط زر المسافة Space أو زر الإدخال Enter.
النقر المزدوج DoubleClick	ينطلق هذا الحدث عند نقر الأداة مرتين سريعتين بالفأرة، أو عند ضغط بعض أزرار الاختصار من لوحة المفاتيح. لاحظ أن انطلاق هذا الحدث يسبقه انطلاق الحدث Click، فهو ينطلق بعد أول ضغطة لزر الفأرة.

٤- أحداث يسببها المبرمج:

هناك أحداث تطلقها الأداة، بسبب كتابة المبرمج لكود قام فيه بتغيير بعض خصائصها.. مثلا: لو غير المبرمج الخاصية BackColor من الكود، فإن الأداة تطلق حدثا اسمه BackColorChanged.. وفي الغالب لا تكون هذه الأحداث هامة بالنسبة للمبرمج ولا يستخدمها، لأنه هو السبب فيها، ولا يحتاج إلى اتخاذ رد فعل لها.

فئة الزر Button Class:

يستخدم الزر Button لتنفيذ شيء معين عند ضغطه.

ويمتلك الزر كل الخصائص والأحداث المشتركة بين الأدوات التي شرحناها سابقا، وبالإضافة إلي هذا يمتلك بعض الخصائص الخاصة به، مثل:

حجم تلقائي AutoSize:

القيمة الافتراضية لهذه الخاصية هي False.. هذا يعني أنك تستطيع تغيير عرض وارتفاع الزر كما تشاء، بغض النظر عن حجم النص المكتوي عليه.. فقد تجعل أبعاد الزر أكبر من أبعاد النص، وقد تجعل أبعاد الزر أصغر من النص بحيث لا تظهر بعض

الحروف عليه.. أما إذا جعلت قيمة هذه الخاصية True فسيتم تغيير أبعاد الزر تلقائيا لتناسب أبعاد النص المكتوب عليه، بحيث يحتوي الزر هذا النص احتواء كاملا.. ولو حاولت تكبير أو تصغير الزر، فسيعود دائما إلى الحجم المناسب للنص المكتوب عليه.

الصورة Image:

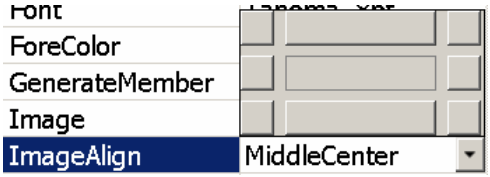
تتيح لك هذه الخاصية اختيار الصورة التي سيتم عرضها على الزر.. ويمكنك اختيار الصورة من نافذة الخصائص بنفس الطريقة التي تعرفنا عليها مع الخاصية BackGroundImage.

وتختلف الخاصية Image عن الخاصية BackGroundImage في أن الخاصية BackGroundImage تتحكم في تنسيق الصورة من خلال الخاصية BackgroundImageLayout، بينما الخاصية Image تتيح تحكم في محاذاة الصورة من خلال الخاصية ImageAlign.

محاذاة الصورة ImageAlign:

تتيح لك هذه الخاصية التحكم في محاذاة الصورة المعروضة على الزر بواسطة الخاصية Image.

ولتغيير قيمة هذه الخاصية في نافذة الخصائص، يمكنك إسدال زر الاختيار في خانة القيمة، حيث ستظهر لك تسع مربعات توضح كل مواضع الصورة المحتملة على الزر:



ويمكنك ضغط أي مستطيل بالفأرة لاختياره، وعندئذ ستجد أن قيمة الخاصية المكتوبة نصيا في الخانة قد تغيرت.. على كل حال، هذه الخاصية تأخذ إحدى القيمة التالية:

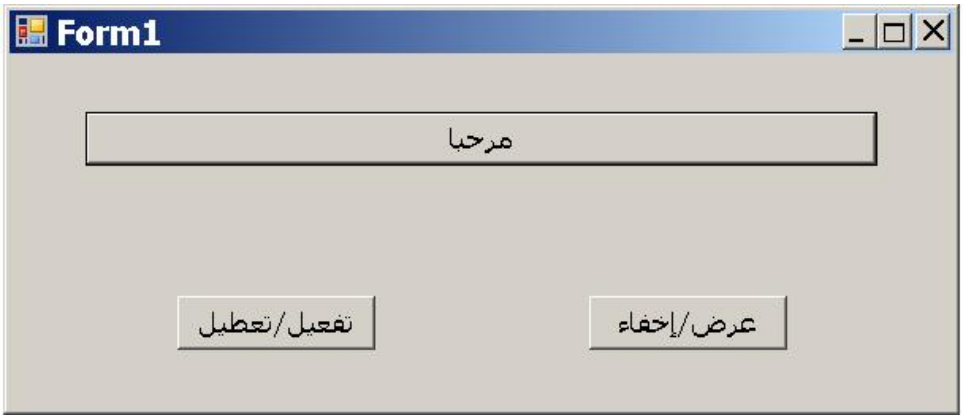
BottomCenter	محاذاة أسفل مركز الزر.
BottomLeft	محاذاة أسفل يسار الزر.
BottomRight	محاذاة أسفل يمين الزر.
MiddleCenter	محاذاة أوسط مركز الزر.
MiddleLeft	محاذاة أوسط يسار الزر.
MiddleRight	محاذاة أوسط يمين الزر.
TopCenter	محاذاة أعلى مركز الزر.
TopLeft	محاذاة أعلى يسار الزر.
TopRight	محاذاة أعلى يمين الزر.

محاذاة النص TextAlign:
تتيح لك هذه الخاصية التحكم في موضع النص على الزر، ويمكن تغيير قيمتها بنفس التي شرحناها في الخاصية السابقة ImageAlign.

علاقة النص والصورة TextImageRelation:
تتيح لك هذه الخاصية التحكم في كيفية عرض النص والصورة معا، وهي تأخذ إحدى القيم التالية:

ImageAboveText	تظهر الصورة أعلى الزر، ويظهر النص تحتها.
TextAboveImage	يظهر النص أعلى الزر، وتظهر الصورة تحته.
ImageBeforeText	تظهر الصورة على يسار الزر، ويظهر النص بعدها على اليمين.
TextBeforeImage	يظهر النص على يسار الزر، وتظهر الصورة بعده على اليمين.
Overlay	النص يظهر كأنه مكتوب فوق الصورة.. في هذه الحالة تتحكم الخاصيتان TextAlign و ImageAlign في موضع ظهور النص على الصورة.

دعنا نأخذ مثالا على استخدام الأزار.. أنشئ مشروعا جديا اسمه Buttons، وضع على النموذج ثلاثة أزرار، ورتبها لتبدو كالتالي:



غير خاصيتي الاسم والنص لكل زر تبعا للجدول التالي:

Text	Name
مرحبا	BtnWelcome
عرض/إخفاء	BtnVisible
تفعيل/تعطيل	BtnEnabled

اضغط F5 لتشغيل البرنامج وجرب ضغط أي زر.. طبعا لن يحدث أي شيء لأننا لم نكتب الإجراء المعالج Handler لحدث ضغط هذه الأزرار بعد.. أغلق البرنامج وتعال نكتب هذه الإجراءات.

انقر مرتين بالفأرة على الزر "مرحبا" واكتب ما يلي في حدث ضغطه:

```
Me.Text = "مرحبا"
```

هذا الكود يكتب الكلمة مرحبا في الخاصية Text الخاصية بالنموذج الحالي، والذي اشرنا إليه بالكلمة Me.

عد إلى تصميم النموذج بضغط الشريط العلوي Form1.vb[Design]، وانقر مرتين بالفأرة على الزر "عرض/إخفاء"، واكتب ما يلي في حدث ضغطه:

```
BtnWelcome.Visible = Not BtnWelcome.Visible
```

نريد أن نفهم هذا الكود.. هذا الكود يستخدم المعامل Not لعكس حالة الخاصية Visible لزر الترحيب.. لاحظ أن عملية التساوي يتم تنفيذها من اليمين إلى اليسار، بمعنى أن الكود الموجود على يمين العلامة = يتم تنفيذه أولا، ثم يوضع الناتج في الكود الموجود على يسار العلامة =.. هذا معناه أن فيجيوال بيزيك تنفذ هذا الكود كالتالي:

١- تقرأ قيمة الخاصية BtnWelcome.Visible الموجودة على يسار العلامة = .. هذه الخاصية كما عرفنا لها قيمتان فحسب: إما True إذا كان الزر مرئيا، وإما False إذا كان الزر مختفيا.

٢- يقوم المعامل Not بعكس القيمة التي تمت قراءتها.. فلو كانت هذه القيمة True فستتحول إلى False.. ولو كانت False فستتحول إلى True.

٣- يتم وضع الناتج الذي حصلنا عليه في الكود الموجود على يسار علامة التساوي.. هذا الكود يشير إلى الخاصية BtnWelcome.Visible أيضا، مما يعني أن القيمة الناتجة ستوضع فيها.. هذا يعني أننا قرأنا قيمة الخاصية وعكسناها ووضعناها في نفس الخاصية، وبذلك نكون قد عكسنا قيمة الخاصية في خطوة واحدة.. ولكي لا يربكك هذا الأمر، فعليك أن تقرأ هذا الكود كالتالي:
قيمة الخاصية Visible بعد تنفيذ الكود = عكس قيمة الخاصية Visible قبل تنفيذ الكود.

يمكنك أن تجرب البرنامج الآن.. لو ضغطت الزر "مرحبا" فسيظهر على شريط النموذج النص "مرحبا".. ولو ضغطت الزر "عرض/إخفاء" فسيختفي الزر "مرحبا".. ولو ضغطت نفس الزر مرة أخرى فسيظهر الزر "مرحبا" من جديد.. وهكذا بالتبادل: ضغطة تخفي الزر وضغطة تظهره.. يسمى الزر في هذه الحالة "مفتاح التبديل" Toggle أو Switch.

أوقف تشغيل البرنامج، وتعال نكتب كود الزر "تفعيل/تعطيل".. هذا الكود مماثل لكود الزر "عرض/إخفاء" مع اختلاف واحد، هو أننا سنعكس قيمة الخاصية Enabled:

```
BtnWelcome.Enabled = Not BtnWelcome.Enabled
```

مرة أخرى، دعنا نقرأ هذا الكود معا كالتالي:

قيمة الخاصية Enabled بعد تنفيذ الكود = عكس قيمة الخاصية Enabled قبل تنفيذ الكود.
جرب البرنامج الآن.. اضغط الزر "تفعيل/تعطيل".. ستجد أن الزر "مرحبا" قد تم تعطيله، ولو حاولت أن تضغط الزر "مرحبا" فلن يستجيب لك.. اضغط الزر "تفعيل/تعطيل" مرة أخرى.. ستجد أن الزر "مرحبا" قد تم تفعيله من جديد، ولو ضغطته فستظهر الكلمة مرحبا على شريط النموذج.
كل شيء على ما يرام ولله الحمد.. أوقف تشغيل البرنامج، ودعنا ننتقل إلى الخطوة التالية.
نريد الآن تطوير هذا البرنامج قليلا.. فبدلا من أن نستخدم النصين "عرض/إخفاء" و "تفعيل/تعطيل"، نريد أن نكتب على الزر الوظيفة التي سيقوم بها بدقة:

- فنكتب على الزر الأول "إخفاء" إن كان ضغطه سيخفي الزر، و"عرض" إن كان ضغطه سيظهره.
- ونكتب على الزر الثاني "تعطيل" إن كان ضغطه سيعطل الزر، و"تفعيل" إن كان ضغطه سيفعله.

قد تقول لي إن بالإمكان استخدام المعامل Not لعكس عنوان كل زر.. للأسف هذا غير متاح، لأن المعامل Not يتعامل مع القيم المنطقية Boolean التي لها القيمتان True و False فحسب، بينما عنوان الزر نص لا يمكن عكسه بطريقة مباشرة.
إذن فما الحل؟
علينا هنا أن نستخدم طريقة أخرى، وهي جملة الشرط If.. Then.. دعنا نفكر معا أولا كيف ستكون خوارزمية الحل، ولنأخذ زر العرض والإخفاء كمثال:

إذا كان الزر BtnWelcome مرئيا إذن:
قم بإخفاء الزر BtnWelcome
اجعل عنوان الزر BtnVisible = "عرض"
غير ذلك:
قم بعرض الزر BtnWelcome
اجعل عنوان الزر BtnVisible = "إخفاء"
نهاية الشرط

الفكرة واضحة طبعا.. دعنا إذن نكتبها بفيجيوال بيزيك.. اتبع معي هذه الخطوات:

١- اعرض مصمم النماذج بضغط الشريط الخاص به.
٢- اضغط Ctrl+A لتحديد كل الأزرار على النموذج.
٣- اضغط Ctrl+C لنسخ الأزرار إلى لوحة القصاصات.
٤- اضغط Ctrl+S لحفظ النموذج الحالي.
٥- ابدأ مشروعا جديدا اسمه Buttons2 بالطريقة المألوفة.

٦- عند ظهور النموذج في المشروع الجديد غير أبعاده ليكون له نفس حجم النمـــوذج في المشروع القديم.
٧- اضغط Ctrl+V للصق الأزرار التي نسخناها من المشروع القديم.. بهذه الطريقة سنوفر على أنفسنا إعادة تصميم النموذج والأزرار من البداية.. لاحظ أن كل زر سيكون له نفس الاسم والعنوان والشكل التي كانت للأزرار في المشروع القـــديم.. لكن لن يكون هناك أي كود مع هذه الأزرار.
٨- غير النص المكتوب على زر العرض والإخفاء باستخدام الخاصـــية Text إلـــى "إخفاء".. السبب في هذا أن الزر "مرحبا" سيكون مرئيا عند تشغيل البرنامج، لهذا ستكون مهمة هذا الزر هي إخفاؤه.
٩- غير النص المكتوب على زر التفعيل والتعطيل باستخدام الخاصـــية Text إلـــى "تعطيل".. السبب في هذا أن الزر "مرحبا" سيكون فعالا عند تشـــغيل البرنـــامج، لهذا ستكون مهمة هذا الزر هي تعطيله.

والآن، دعنا نكتب كود زر العرض والإخفاء.. انقر مرتين على الزر واكتب ما يلي فـــي حدث ضغطه:

```
If BtnWelcome.Visible = True Then
   الزر مرئي.. سنخفيه '
   BtnWelcome.Visible = False
   BtnVisible.Text = "عرض"
Else
   الزر غير مرئي.. سنعرضه '
   BtnWelcome.Visible = True
   BtnVisible.Text = "إخفاء"
End If
```

لا أظن الكود يحتاج إلى شرح، فهو يفعل نفس ما شرحناه في الخوارزمية بالضبط.
بالمثل، يمكننا كتابة كود زر التفعيل والتعطيل.. سيكون كما يلي:

```
If BtnWelcome.Enabled = True Then
   الزر فعال.. سنعطله '
   BtnWelcome.Enabled = False
   BtnEnabled.Text = "تفعيل"
Else
   الزر غير فعال.. سنفعله '
   BtnWelcome.Enabled = True
   BtnEnabled.Text = "تعطيل"
End If
```

جرب البرنامج الآن.. اضغط الزر "إخفاء".. ستجد أن الزر "مرحبا" قد اختفي، وصار عنوان الزر "عرض".. ولو ضغطته مرة أخرى، فسيظهر الزر "مرحبا" وسيتحول العنوان مرة أخرى إلى "إخفاء".. وهكذا دواليك.
نفس الأمر مع الزر "تعطيل"، فسيؤدي ضغطه إلى تعطيل زر الترحيب مع تحول العنوان إلى "تفعيل".. ولو ضغطته مرة أخرى فسيتم تفعيل زر الترحيب، وسيعود العنوان إلى "تعطيل".. وهكذا دواليك.
برنامج رائع.. أليس كذلك؟
أرجو أن تكون قد استمتعت به.

فئة اللافتة Label Class:

تستخدم اللافتة لعرض بعض المعلومات للمستخدم، وهي تمتلك كل الخصائص المشتركة التي شرحناها سابقا، وبالإضافة إليها تمتلك بعض الخصائص التي يمتلكها الزر Button، مثل:

- حجم تلقائي AutoSize.
- الصورة Image.
- محاذاة الصورة ImageAlign.
- محاذاة النص TextAlign.

لهذا لن نكرر شرحها هنا.
كما تمتلك اللافتة بعض الخصائص الخاصة بها، مثل:

طراز الإطار BorderStyle:

تتحكم هذه الخاصية في شكل إطار اللافتة، وهي تأخذ إحدى القيم التالية:

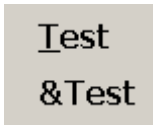
None	تظهر الأداة بدون أي إطار.. هذه هي القيمة الافتراضية.
FixedSingle	يظهر إطار اللافتة في هيئة مستطيل أسود مسطح.
Fixed3D	تظهر اللافتة الأداة مجسمة (ثلاثية الأبعاد).

والصورة التالية توضح كيف نظهر اللافتة في كل حالة:

استخدام حرف تذكيري UseMnemonic:
تتيح لك الخاصية UseMnemonic تحديد إذا كنت تريد استخدام حرف تذكيري أم لا.

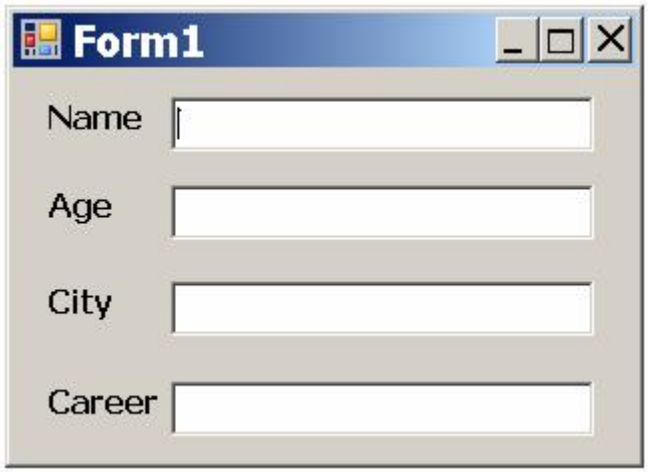
الحرف التذكيري Mnemonic Key:
يسمى أيضا بحرف الاختصار Shortcut key أو حرف الوصول Access Key، وهو الحرف الذي إذا ضغطه المستخدم مع حرف Alt من لوحة المفاتيح يتم تفعيل اللافتة.

ويمكنك تعريف حرف تذكيري للافتة باستخدام الخاصية Text، وذلك بوضع العلامة & قبل أي حرف ضمن النص المكتوب في هذه الخاصية.. حينما تفعل هذا ستجد أن العلامة & لم تظهر على اللافتة، وبدلا من هذا تم وضع خط أسفل الحرف التالي لها، مما يلفت نظر المستخدم إلى أنه حرف اختصار.
لاحظ أن هذا يعمل لأن الخاصية UseMnemonic قيمتها True، لكن إذا جعلت قيمتها False فستتم كتابة العلامة & على اللافتة كما هي باعتبارها حرفا عاديا.
كمثال: ضع لافتة على النموذج وافتح نافذة الخصائص.. غير اسم اللافتة إلى LblTest، واكتب في الخاصية Text الخاصة بها النص "&Test".. انتقل إلى النموذج واضغطه بالفأرة مرة واحدة لترى تأثير هذه الخاصية.. ستجد أن اللافتة تعرض النص التالي: Test، حيث تم وضع خط تحت الحرف T، مما يعني أنه حرف الاختصار (الحرف التذكيري)، وأن المستخدم لو ضغط Alt+T من لوحة المفاتيح فسيتم الانتقال إلى اللافتة LblTest.



عد إلى نافذة الخصائص وغير قيمة الخاصية UseMnemonic إلى False.. ستجد أن اللافتة تعرض النص "&Test" كما تمت كتابته بالضبط في الخاصية Text.
الجدير بالذكر أنك تستطيع استخدام نفس حرف الاختصار لأكثر من لافتة.. في هذه الحالة يستطيع المستخدم ضغط حرف الاختصار للانتقال إلى أول لافتة تستخدم هذه الحرف، ولو ضغط حرف الاختصار مرة أخرى فسينتقل إلى اللافتة التالية لها التي تستخدم نفس الحرف، وهكذا...
ولكن لحظة من فضلك:
ألم نذكر من قبل أن اللافتة لا يمكن تفعيلها ولا يوضع بها المؤشر Focus؟... فكيف يمكن الانتقال إليها باستخدام حرف الاختصار؟
في الواقع لا يتم الانتقال إلى اللافتة نفسها.. فعند ضغط مفتاح الاختصار من لوحة المفاتيح، ينتقل المؤشر Focus إلى الأداة التالية للافتة مباشرة!
هل تتعجب من هذا؟.. سيزول عجبك إن علمت السر وراء هذا التصرف:
أنت تعلم أننا نستخدم اللافتة لشرح وظيفة بعض الأدوات كمربع النص TextBox، حيث نضع اللافتة أعلى مربع النص أو بجواره مباشرة.. في هذه الحالة يكون مربع النص هو الأداة التالية للافتة، ولو ضغط المستخدم حرف الاختصار الخاص باللافتة من لوحة

المفاتيح، فإنه ينتقل مباشرة إلى مربع النص التالي لها.. هـذا يسـهل علـى مسـتخدمي البرنامج كثيرا، خاصة عندما يحتوي النموذج على عدد كبير من مربعات النصوص.
تعال نجرب هذا:
ابدأ مشروعا جديدا اسمه Shortcuts، وصمم النموذج ليبدو كما في الصورة:



غير خاصية النص Text لكل لافتة كما في الجدول.. لاحظ أن العمود الثالث يوضح لـك حرف الاختصار الناتج عن الخاصية Text:

اللافتة	الخاصية Text	حرف الاختصار
Label1	&Name	Alt+N
Label2	&Age	Alt+A
Label3	&City	Alt+C
Label4	Ca&reer	Alt+R

لاحظ أننا لم نهتم بتغيير أسماء الأدوات لأننا لن نكتب أي كود هنا.. هـذا مجـرد مثـال لتجربة حروف الاختصار.
شغل البرنامج بضغط F5.. اضغط حروف الاختصار الموضحة في الجدول وراقب كيف سيتم الانتقال إلى مربع النص المجاور للافتة التي يخصها هذا الاختصار.

ملحوظة:
يمكن استخدام حرف تذكيري (حرف اختصار) للزر أيضا بنفس الطريقة.. لكن وظيفـة حرف الاختصار تختلف في الزر عن اللافتة، فضغط حـرف الاختصـار مـن لوحـة المفاتيح سيؤدي إلى تنفيذ حدث ضغط الزر.. هذا يختصر الوقت على المستخدم عنـدما يكون على النموذج العديد من الأزرار.

ترتيب الوصول TabIndex:

قد تواجهك مشكلة عند تجربة البرنامج السابق، فقد تضغط حرف الاختصار فتجد أن مؤشر الكتابة قد ظهر في مربع نص لا يجاور اللافتة التي يخصها حرف الاختصار!

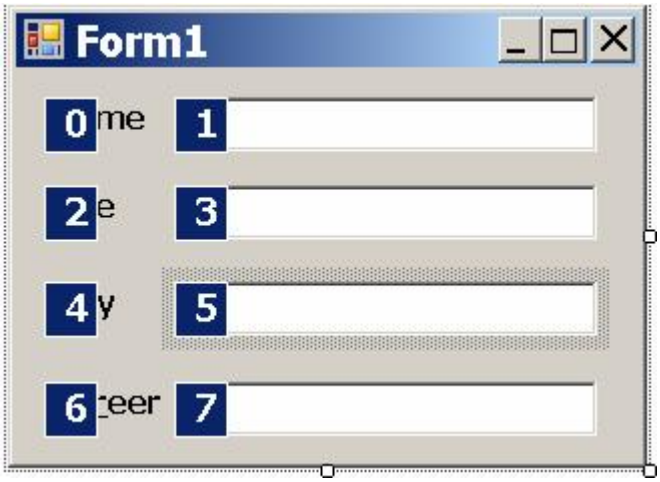
السبب في هذا الخطأ، هو أن مربع النص ليس الأداة التالية للافتة، حتى ولو كان يظهر بجوار اللافتة مباشرة!

فما يتحكم في ترتيب الأدوات على النموذج، هو الخاصية TabIndex الخاصية بكل أداة.. لهذا لكي يعمل البرنامج السابق بشكل صحيح، يجب أن تتأكد أن قيمة هذه الخاصية لكل مربع نص تزيد على قيمتها للافتة المجاورة به بمقدار ١.

لفعل هذا، حدد كل أداة على النموذج، وضع في الخاصية TabIndex الخاصة بها القيمة الموضحة في الجدول:

الأداة	الخاصية TabIndex
Label1	٠
TextBox1	١
Label2	٢
TextBox1	٣
Label3	٤
TextBox1	٥
Label4	٦
TextBox1	٧

الجدير بالذكر أنك تستطيع تغيير ترتيب الأدوات بطريقة مرئية أسهل وأسرع، وذلك بضغط قائمة العرض View Menu واختيار الأمر Tab Order منها.. ستجد أن رقم كل أداة في الترتيب قد كتب عليها، كما في الصورة التالية:



الآن وبمنتهى البساطة، اضغط هذه الأرقام بالفأرة بالترتيب الذي تريده، ليصير هذا هو ترتيب التنقل بين الأدوات بالزر Tab، مع ملاحظة أن ضغط نفس الرقم أكثر من مرة يعمل على زيادته باستمرار إلى أن يصل إلى أكبر رقم ممكن (وهو عدد الأدوات الموجودة على النموذج)، ومن ثم يعود إلى الصفر من جديد.
بعد أن تنتهي من وضع أرقام كل الأدوات بالترتيب الذي تريده، اضغط القائمة الرئيسية View واختر الأمر Tab Order منها من جديد لإخفاء هذه الأرقام والعودة إلى تصميم النموذج التقليديّ.. لو فحصت الخاصية TabIndex لكل أداة بعد هذه العملية فستجد أنها تغيرت إلى الترتيب الذي قمت به.

ملحوظة:
تمت تسمية الخاصية TabIndex بهذا الاسم، لأنها تتحكم في ترتيب الوصول إلى الأداة عند ضغط زر الجدولة Tab من لوحة المفاتيح أكثر من مرة.. لقد ذكرنا من قبل أن ضغط الزر Tab من لوحة المفاتيح يؤدي إلى الانتقال إلى الأداة التالية للأداة الفعالة الحالية Active Control.. أيضا يؤدي ضغط Shift+Tab إلى الانتقال إلى الأداة السابقة للأداة الفعالة الحالية.. وباستخدام هذين الاختصارين يمكن للمستخدم التحرك بين الأدوات للأمام أو الخلف دون الحاجة إلى استخدام الفأرة.

فئة مربع النص TextBox Class:

مربع النص هو مساحة لعرض وتحرير النص.. وقد رأينا كيف نستخدم الخاصية Text لتغيير النص المعروض في مربع النص.. بخلاف هذا يمتلك مربع النص كل الخصائص المشتركة بين الأدوات التي تعرفنا عليها في بداية هذا الفصل، كما يشترك مع اللافتة في امتلاكه خاصية طراز الإطار BorderStyle التي تتحكم في شكل إطار مربع النص..
كما يمتلك مربع النص بعض الخصائص التي يمتلكها الزر Button، مثل:

- الصورة Image.
- محاذاة الصورة ImageAlign.

وبالإضافة إلى هذا يمتلك مربع النص الخصائص التالية:

محاذاة النص TextAlign:

هذه الخاصية موجودة في كل من الزر واللافتة، لكن قيم هذه الخاصية في مربع النص تحتلف عن قيمها الممكنة في الزر واللافتة، فمحاذاة النص المكتوب في مربع النص تأخذ القيم التالية فحسب:

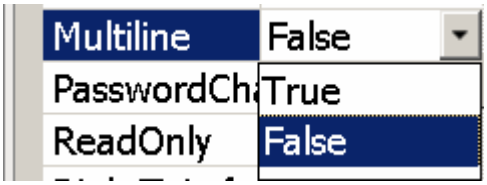
Left	محاذاة النص إلى اليسار
Right	محاذاة النص إلى اليمين
Center	محاذاة النص إلى المنتصف

لاحظ أنك تستطيع استخدام الخاصية RightToLeft لعرض النص العربي من اليمين إلى اليسار، لكن هذا سيعكس تأثير قيمة خاصية محاذاة النص TextAlign، لأن الخاصية RightToLeft تعمل كالمرآة، فتعكس اتجاه مربع النص، لهذا تصير محاذاته إلى اليسار معكوسة فكأنما تمت محاذاته إلى اليمين!.. الجدول التالي يلخص تأثير هذه الخاصية:

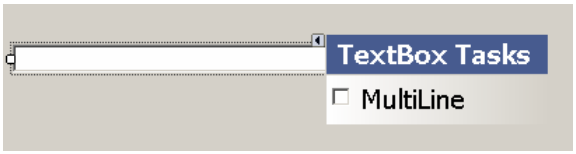
القيمة	تأثيرها إذا كانت RightToLeft = False	تأثيرها إذا كانت RightToLeft = True
Left	محاذاة النص إلى اليسار	محاذاة النص إلى اليمين (معكوسة)
Right	محاذاة النص إلى اليمين	محاذاة النص إلى اليسار (معكوسة)
Center	محاذاة النص إلى المنتصف	محاذاة النص إلى المنتصف

"متعدد الأسطر" MultiLine:

في الوضع التلقائيّ، يرفض مربع النصّ كتابة أكثر من سطر واحدٍ فيه.. لتغيير ذلك، حدد مربع النص واضغط F4 لفتح نافذة الخصائص، وابحث فيها عن الخاصية "متعدد الأسطر" MultiLine.. ستجد أن قيمتها الحالية هي False، بمعنى أن مربع النص ليس متعدد الأسطر، أي أنه يتعامل مع سطر واحد فقط.. لتغيير قيمة هذه الخاصية قم بالنقر المزدوج Double-click على الكلمة False لتتحول إلى True.. يمكنك أيضا أن تضغط الزر المجاور للكلمة False ليتم إسدال قائمة بها كل القيم الممكنة لهذه الخاصية.. من هذه القائمة اختر القيمة True.



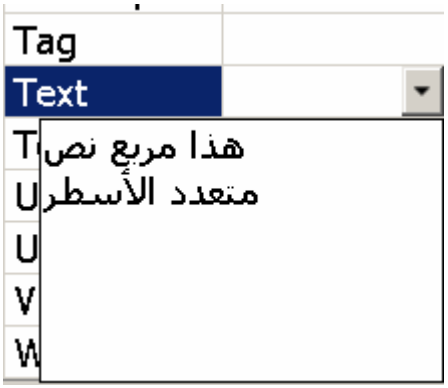
كما أن هناك طريقة ثالثة لتغيير هذه الخاصية.. حدد مربع النص على النموذج.. سترى مثلثا صغيرا على الجهة اليمنى من حافته العلوية.. اضغط هذا المثلث بالفأرة لتظهر لك قائمة مهام مربع النص كما يبدو في الصورة:



الآن، كل ما عليك هو ضغط مربع الاختيار الموجود بجوار الكلمة MultiLine لوضع علامة الاختيار ✓ به، ثم أضغط في أي موضع على النموذج.. هذا سيجعل مربع النص متعدد الأسطر.. إذا أردت أن تعيده مفرد السطر، استخدم نفس الطريقة واضغط مربع الاختيار لإزالة العلامة ✓ منه.

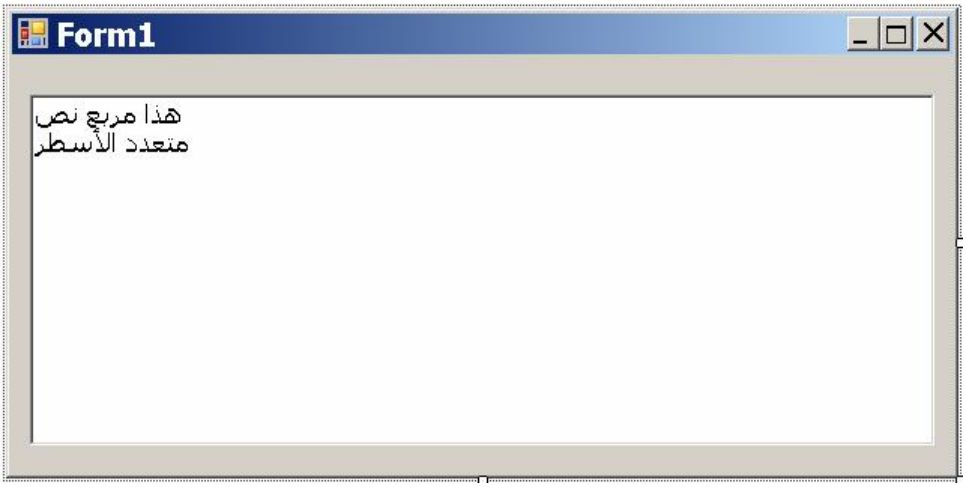
لاحظ أن مربع النص مفرد السطر يرفض تغيير ارتفاعه، وكلما حاولت زيادة ارتفاعه عاد كما كان، حيث يأخذ الارتفاع المناسب لحجم الخط المستخدم به.. لهذا يمكنك تغيير

عرضه فقط.. أما مربع النص متعدد الأسطر فيمكنك زيادة ارتفاعه كما تريد، ليستطيع المستخدم رؤية الأسطر المكتوبة به.



ويمكنك كتابة السطور في مربع النص في وقت التصميم من خلال الخاصية Text نافذة الخصائص.. اضغط زر الإسدال الموجود في خانة هذه الخاصية في نافذة الخصائص، واكتب السطور التي تريدها، كما في الصورة.

بعد أن تكتب كل السطور، اضغط بالفأرة فوق أي موضع على النموذج، لترى التغييرات في مربع النص، كما في الصورة التالية:

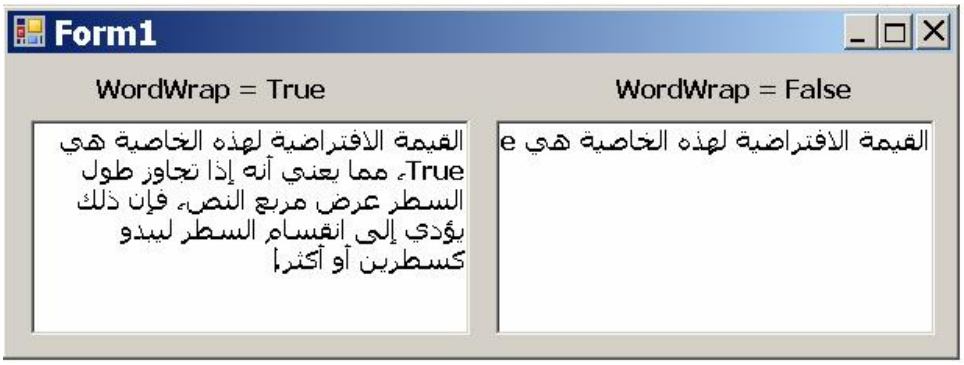


ويمكنك استخدام الخاصية Font لتغيير الخط الذي سيعرض به النص في مربع النص.

التفاف الأسطر WordWrap:

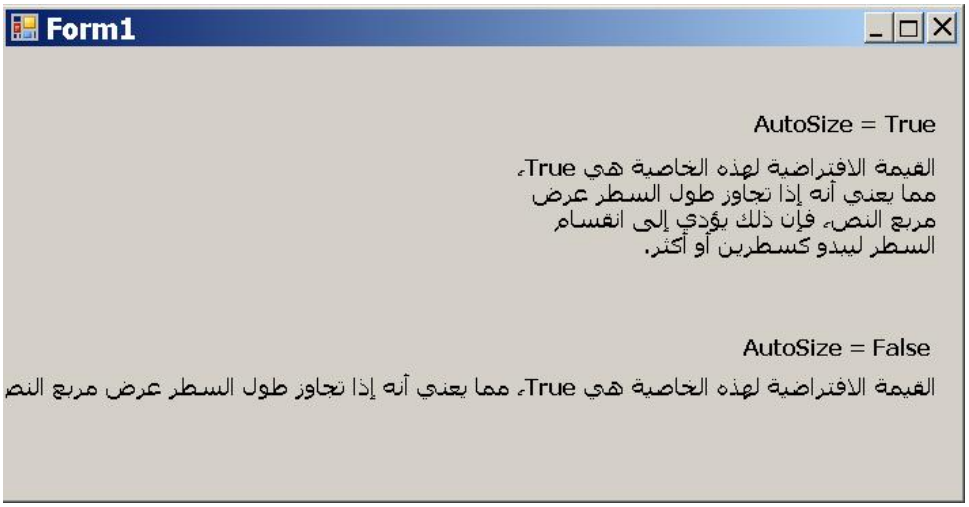
القيمة الافتراضية لهذه الخاصية هي True، مما يعني أنه إذا تجاوز طول السطر عرض مربع النص، فإن ذلك يؤدي إلى انقسام السطر ليبدو كسطرين أو أكثر.

أما إذا جعلت هذه الخاصية False، فإن المستخدم يستطيع الكتابة في نفس السطر إلى أن يتجاوز عرض السطر، وفي هذه الحالة يجب عليك أن تعرض المنزلق الأفقي في مربع النص، حتى يستطيع المستخدم تحريكه يمينا ويسارا لعرض أي جزء يريده من السطر.

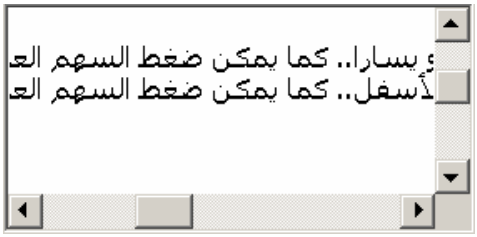


ملحوظة:

لا تمتلك اللافتة خاصية التفاف الأسطر، لكنها تستطيع القيام بنفس الوظيفة بطريقة أخرى، وذلك باستخدام خاصية الحجم التلقائي AutoSize التي تعتبر عكس الخاصية WordWrap، فلو جعلت قيمتها False فإن اللافتة ستسمح لك بتغيير عرضها، ولو صغرت هذا العرض عن عرض السطر المكتوب فيها، فسيتلف على أكثر من سطر، لكن لا تنس تكبير ارتفاع اللافتة لترى هذا.



المنزلقان ScrollBars:



يمكن أن يزيد عدد السطور في مربع النص عن ارتفاعه، كما يمكن أن يزيد عرض أي سطر عن عرض مربع النص.. في هذه الحالة تتيح لك هذه الخاصيّة اختيار إظهار المنزلق الأفقي أو الرأسي أو كليهما.

المنزلق (شريط اللف) ScrollBar:
يوجد منه نوعان: ١- الأفقي: وهو شريط أفقي يظهر على حافة مربع النص السفلية، وعليه رأس متحرك يمكن ضغطه بالفأرة وتحريكه يمنا أو يسارا.. كما يمكن ضغط السهم العلوي أو السفلي لتغيير موضع الرأس المتحرك. ٢- الرأسي: وهو شريط رأسي يظهر على حافة مربع النص اليمنى، وعليه رأس متحرك يمكن ضغطه بالفأرة وتحريكه لأعلى أو لأسفل.. كما يمكن ضغط السهم العلوي أو السفلي لتغيير موضع الرأس المتحرك.

وتأخذ الخاصية ScrollBars إحدى القيم التالية:

None	لا يتم عرض أيّ من المنزلقين.
Vertical	يتم عرض المنزلق الرأسي فحسب.. لاحظ أن هذا المنزلق لن يظهر إلا إذا كان مربع النص متعدد الأسطر (MultiLine = True).
Horizontal	يتم عرض المنزلق الأفقي فحسب.. لاحظ أن هذا المنزلق لن يظهر إلا إذا كان مربع النص متعدد الأسطر (MultiLine = True)، وكانت لخاصية التفاف السطر WordWrap القيمة False.
Both	يتم عرض كلا المنزلقين الرأسي والأفقي معا.

طول النص TextLength:

تخبرك هذه الخاصية بعدد حروف النص الموجود في مربع النص.. ولا يمكنك تغيير قيمة هذه الخاصية بنفسك، فهي تتغير تلقائيا مع تغير عدد الحروف المكتوبة في مربع النص.. هذا هو السبب الذي يجعل هذه الخاصية لا تظهر في نافذة الخصائص.

ولاستخدام هذه الخاصية، افتح مشروعا جديدا، وضع على النموذج مربع نص اسمه TxtTest ولافتة اسمها LblTxtLen، وزرا اسمه BtnTest.. انقر الزر مرتين واكتب في حدث ضغطه الكود التالي:

LblTxtLen.Text = TxtTest.TextLength

هذا الكود سيعرض في اللافتة عدد الحروف المكتوبة في مربع النص.. شغل البرنامج واضغط الزر.. ستعرض اللافتة العدد ٠ .. اكتب بعض الحروف في مربع النص واضغط الزر.. سيظهر في اللافتة عدد هذه الحروف.

أقصى طول MaxLength:

القيمة الافتراضيّة لهذه الخاصيّة هي ٣٢٧٦٧.. هذا يعني أن مستخدم برنامجك يستطيع كتابة أكثر من ٣٢ ألف حرف في مربع النص، وهو عدد كبير فعلا!

لكنك أحيانا لا تريد أن يكتب المستخدم نصا بهذا الطول.. مثلا: إذا استخدمت مربع نص في برنامجك ليكتب فيه المستخدم اسمه، فليس من المتوقع أن يزيد هذا الاسم عن ٥٠ حرفا، لهذا يمكنك أن تجعل للخاصية MaxLength القيمة ٥٠.. في هذه الحالة سيقبل

مربع النص أول ٥٠ حرفا يكتبها المستخدم، وأي حرف سيكتبه بعد ذلك سيرفضه مربع النص مع إصدار رنة تنبيه من سماعة الجهاز.
لاحظ أنك تستطيع أن تجعل قيمة هذه الخاصية صفرا.. هذا لا يعني منع المستخدم من الكتابة في مربع النص، بل على العكس تماما، هو يعني عدم وضع أية قيود على الطول الأقصى المسموح للمستخدم بإدخاله في مربع النص، مما يتيح له كتابة حوالي ٢ مليار حرف في مربع النص!

حالة الأحرف CharacterCasing:
تتيح لك هذه الخاصية التحكم في حالة الحروف الإنجليزية التي يكتبها المستخدم في مربع النص، وهي تأخذ إحدى القيم التالية:

Normal	تتم كتابة حروف عادية (كما يكتبها المستخدم).
Lower	كل الحروف ستكون صغيرة Small.. فمثلا لو كتب المستخدم AbGf، فإنها ستظهر في مربع النص abgf.
Upper	كل الحروف ستكون كبيرة Capital.. فمثلا لو كتب المستخدم AbGf، فإنها ستظهر في مربع النص ABGF.

لاحظ أن اختيار الحروف الكبيرة Upper أو الصغيرة Lower سيلغي قدرة المستخدم على التحكم في حالة الحرف الذي يكتبه من لوحة المفاتيح، حيث لن يكون لضغط الزر Shift أو Caps Lock أي تأثير.. لاحظ كذلك أن هذه الخاصية تؤثر على النص الموجود في مربع النص حاليا، وعلى كل الحروف التي ستتم كتابتها فيما بعد.. معنى هذا أنها تضمن لك أن كل حروف النص ستكون كبيرة معا، أو صغيرة معا.. هذا مفيد في حالة التعامل مع كلمات المرور Passwords، لتضمن أن لكل الحروف حالة واحدة (ولتكن الحالة الصغيرة) لضمان عدم حدوث خطأ بسبب ضغط المستخدم لزر الأحرف الكبيرة Caps Lock سهوا.

تمّ تعديله Modified:
عندما يُجري المستخدم أيّ تغيير على محتويات مربّع النصّ بالكتابة أو الحذف، فإن قيمة هذه الخاصيّة تتغيّر تلقائيّا لتصبح True.

للقراءة فقط ReadOnly:
إذا جعلت قيمة هذه الخاصيّة True، فسيتغير لون خلفية مربع النص إلى الرمادي بدلا من الأبيض، ولن يتمكن المستخدم من تغيير النص المعروض في مربع النص أثناء تشغيل البرنامج.. هذا يجعل مربع النص شبيها باللافتة، لكن سيظل هناك فارقا بينهما.. فاللافتة لا تسمح للمستخدم بنسخ النص المعروض عليها، بينما يسمح مربع النص للمستخدم بنسخ النتيجة أو المعلومة المعروضة فيه.

استخدم نافذة الخصائص لتغيير قيمة هذه الخاصية إلى True وقم بتشغيل البرنامج ومحاولة الكتابة في مربع النص.. ستجد أنه يرفض أي حرف تكتبه.
ولو شئت استخدام هذه الخاصية من الكود، فضع مربع نص على النموذج اسمه TxtTest، وضع زرا على النموذج وانقره مرتين بالفأرة، واكتب في حدث ضغطه ما يلي:

```
TxtTest.ReadOnly = Not TxtTest.ReadOnly
```

هذا الكود يطلب عكس حالة الخاصية ReadOnly عند ضغط الزر.. لهذا لو جربت البرنامج وضغطت الزر فسيتحول مربع النص إلى مربع نص للقراءة فقط، ولو أعدت ضغط الزر فسيمكنك الكتابة في مربع النص من جديد.

حرف كلمة المرور PasswordChar:

بعض البرامج تستخدم شاشة الدخول Log-in Window، وهي تعرض الأدوات التالية:

١- مربع نص لإدخال اسم المستخدم TxtUserName.
٢- مربع نص لإدخال كلمة السر TxtPassword.
٣- زر الموافقة BtnOk.
٤- زر الإلغاء BtnCancel.

كما في الصورة التالية:

كما ترى في الصورة، يمكنك قراءة اسم المستخدم "أحمد محمود" المكتوب في مربع النص TxtUserName، لكنك لا تستطيع قراءة كلمة السر المكتوبة في مربع النص TxtPassWord، لأنها تظهر في الصورة ******.. فائدة هذا هو حماية كلمة السر، بحيث إذا كان المستخدم يجلس في مكان عام، لا يرى المحيطون به الحروف الحقيقيّة التي يكتبها.

لكن كيف نفعل هذا في فيجيوال بيزيك؟

الأمر بسيط.. كل ما علينا هو استخدام الخاصية PasswordChar لفعل هذا.

ابدأ أولا بإنشاء مشروع جديد اسمه LogIn، وصمم النموذج ليكون كالموضح في الصورة، وأعط الأدوات الأسماء التي ذكرناها، ثم حدد مربع النص TxtPassword

الخاص بكلمة السر، واضغط F4 لفتح نافذة الخصائص.. اذهب إلى الخاصية PasswordChar، واكتب في خانة القيمة الحرف * .. يمكنك أيضا أن تضع في الخاصية MaxLength القيمة ٦ لتجبر المستخدم على كتابة كلمة سر لا تزيد عن ٦ حروف فقط.

شغل البرنامج بضغط F5، واكتب أي شيء في مربع النص الخاص بكلمة السر.. ستجد أن كل الحروف التي كتبتها قد تم تحويلها إلى الحرف * .

حدد الحروف التي كتبتها، وحاول نسخها بضغط (Ctrl+C) أو قصها بضغط (Ctrl+X).. لن تنجح في هذا.. لقد منعت الخاصية PasswordChar عمليتي النسخ والقص لمزيد من حماية كلمة السر، فقد يقوم المستخدم من على الجهاز قبل أن يغلق النافذة، فيأتي شخص متلصص ويحاول نسخها إلى أي مكان آخر ليستطيع استخدامها، لكنه لحسن الحظ لن ينجح في هذا!

لاحظ أن هذه الخاصيّة لا تؤثر على قيمة النص الموجود في مربع النصّ، فالحروف التي تراها مكتوبة هي مجرد طريقة للعرض، أما النص الأصلي فهو موجود في خاصية النص Text، ويمكنك من خلالها التعامل مع النص الأصليّ الذي أدخله المستخدم.

تعال نجرب هذا.. دعنا نفترض أن اسم المستخدم الصحيح هو "مجدي عادل"، وأن كلمة السر الصحيحة هي "مصر".. انقر مرتين بالفأرة على الزر "موافق"، واكتب ما يلي في حدث ضغطه:

```
If TxtUserName.Text = "مجدي عادل" Then
   If TxtPassword.Text = "مصر" Then
      MsgBox("اسم المستخدم صحيح وكلمة السر صحيحة")
   Else
      MsgBox("كلمة السر غير صحيحة")
   End If
Else
   MsgBox("اسم المستخدم غير صحيح")
End If
```

لاحظ ما يلي:

١- إذا كان اسم المستخدم صحيحا، يمكننا عندئذ أن نفحص كلمة السر.. أما إذا كان اسم المستخدم خاطئا فلا داعي للمواصلة، وسنعرض رسالة خطأ للمستخدم باستخدام الدالة MsgBox.

٢- إذا كان اسم المستخدم صحيحا وفحصنا كلمة السر وكانت صحيحة، فيجب تشغيل البرنامج.. لكن نظرا لأن هذا مجرد مثال نتعلم من خلاله، ولا يوجد برنامج فعلي لتشغيله، فسنكتفي بعرض رسالة للمستخدم نخبره فيها بأن اسم المستخدم صحيح وكلمة السر صحيحة.. أما إذا لم تكن كلمة السر صحيحة، فسنعرض رسالة نخبر فيها المستخدم بذلك.

بقي أن نبرمج زر الإلغاء.. عد إلى مصمم النموذج وانقر هذا الزر مرتين بالفأرة، واكتب في حدث ضغطه:

Me.Close( )

هذا السطر من الكود يطلب إغلاق النموذج باستخدام الوسيلة Close، وذلك لأن إلغاء العملية يعني أن المستخدم لا يريد تشغيل البرنامج.. مجددا أذكرك أن Me تشير إلى الكائن الحالي، وبما أنها مكتوبة داخل فئة النموذج Form1، فهي تشير إلى كائن النموذج الحالي.

اضغط F5 الآن لتشغيل البرنامج واستمتع بتجربته.

تنبيه:
لا تنسَ أن تحول لغة الكتابة إلى اللغة العربية لتستطيع إدخال اسم المستخدم وكلمة السر بالعربية.. لفعل هذا اضغط زر Alt الأيمن مع زر Shift الأيمن من لوحة المفاتيح.. أما إذا أردت تحويل لغة الكتابة إلى اللغة الإنجليزية، فاضغط زر Alt الأيسر مع زر Shift الأيسر من لوحة المفاتيح.

خصائص التعامل مع النص المحدد Selected Text في مربع النص:

سنتعرف على ثلاث خصائص تتعامل مع النص المحدد في مربع النص، وهي بداية التحديد SelectionStart، وطول التحديد SelectionLength ونص التحديد SelectionText.

النص المحدّد Selected text:
هو جزء من النص الموجود في مربع النص، يبدو ملونا باللون الأزرق، ويكون هو الجزء الفعال في مربع النص، حيث إن: ١- ضغط الزر Delete أو Back Space من لوحة المفاتيح سيحذف كل النص المحدد. ٢- كتابة أي حرف من لوحة المفاتيح سيحذف النص المحدد من مربع النص، وسيكتب الحرف الجديد بدلا منه. ٣- ضغط Ctrl+C من لوحة المفاتيح سينسخ النص المحدد إلى لوحة القصاصات Clipboard. ٤- ضغط Ctrl+X من لوحة المفاتيح سينسخ النص المحدد إلى لوحة القصاصات Clipboard ويحذفه من مربع النص. ٥- ضغط Ctrl+V من لوحة المفاتيح سيحذف النص المحدد من مربع النص، وسيلصق بدلا منه النص الموجود في لوحة القصاصات. ويكفي لمعرفة النص المحدد معرفة تامة، معرفة موضع أول حرف فيه، وطول النص المحدد (أو بمعنى آخر: عدد الحروف المحددة).

ملحوظة:
كل حرف في مربع النص له ترقيم معين.. هذا الترقيم يبدأ من الرقم صفر وينتهي عند الرقم الذي يساوي طول النص -١.. معنى هذا أن: - أول حرف في مربع النص موجود في الموضع رقم ٠ . - ثاني حرف في مربع النص موجود في الموضع رقم ١ . - ثالث حرف في مربع النص موجود في الموضع رقم ٢ . - ..... - آخر حرف في مربع النص موجود في الموضع رقم N، حيث يمكن حساب N بطرح ١ من طول النص كالتالي: N = TextBox1.TextLength - 1

بداية التحديد SelectionStart:

استخدم هذه الخاصية لعمل ما يلي:

١- معرفة رقم أول حرف في النص المحدّد.. يمكنك فعل هذا بقراءة قيمة هذه الخاصية.. هذا الكود مثلا سيعرض رسالة بها رقم أول حرف محدد:

```
MsgBox(TextBox1.SelectionStart)
```

لاحظ أنه لو لم يكن هناك أي نص محدد حاليا في مربع النص، فإن هذه الخاصية ستخبرك بالموضع الذي يوجد به مؤشّر الكتابة حاليّا في مربع النص، مع ملاحظة أن الموضع ٠ يعني أن المؤشّر موجود قبل أوّل حرفٍ في مربع النصّ.

٢- تغيير موضع أول حرف في النص المحدد.. يمكنك فعل هذا بتغيير قيمة هذه الخاصية.. هذا الكود مثلا سيجعل أولا حرف محدد هو الحرف الثالث (الحرف رقم ٢):

```
TextBox1.SelectionStart = 2
```

لاحظ أنك لو وضعت في هذه الخاصية رقما أصغر من طول النص بواحد فإن موضع المؤشر سيكون في الموضع السابق لآخر حرف في النص مباشرة، أما لو وضعت فيها رقما أكبر من أو يساوي طول النص فسيوضع مؤشر الفأرة بعد آخر حرف في النص، ولن يحدث أي خطأ.. لكن الخطأ سيحدث إذا وضعت رقما سالبا في هذه الخاصية، فلا يوجد موضع في النص قبل الموضع رقم صفر.. هل سمعت من قبل عن حرف في الموضع رقم -١ مثلا؟!

دعنا نأخذ مثالا.. ابدأ مشروعا جديدا اسمه TextSelection، وضع مربع نص على النموذج اسمه TxtSel، وضع زرا على النموذج اسمه BtnIncIncSelStart.. هذا الاسم هو اختصار الجملة: Increase Selection Start، بمعنى زيادة موضع بدائة التحديد.. هذا طبعا مع البادئة Btn التي تشير إلى الزر Button.

انقر هذا الزر مرتين بالفأرة واكتب ما يلي في حدث ضغطه:

```
TxtSel.SelectionStart = TxtSel.SelectionStart + 1
```

هذا الكود سيجعل موضع بداية التحديد يزيد بواحد كلما تم ضغط الزر.. وعلى فكرة، هناك صيغة مختصرة لكتابة نفس هذا الكود، وهي كالتالي:

TxtSel.SelectionStart += 1

الرمز =+ يعني زيادة القيمة الموجودة في المتغير أو الخاصية الموجودة على يسار =+، بمقدار القيمة الموجودة على يمين =+ .. وتستطيع استخدام الصيغة التي تستريح لها أكثر من هاتين الصيغتين.

اضغط F5 لتشغيل البرنامج.. اضغط الزر أكثر من مرة بينما مربع النص ما زال فارغا.. لن يحدث أي شيء.. اكتب بعض الحروف في مربع النص، وضع مؤشر الكتابة قبل أول حرف فيها.. اضغط الزر أكثر من مرة.. لن ترى شيئا يحدث!

السبب في هذا أن ضغط الزر يجعله هو الأداة الفعالة، وبالتالي لن ترى مؤشر الكتابة Caret في مربع النص لتشعر بتأثير ضغط الزر.. أغلق النموذج وتعال نحل هذه المشكلة.. لدينا هنا وسيلة من وسائل مربع النص اسمها وضع المؤشر Focus، وكل الأدوات عموما تمتلك هذه الوسيلة.. مجرد استدعاء هذه الوسيلة من خلال مربع النص سينقل مؤشر الكتابة إليه مباشرة.. تعال نعدل كود ضغط الزر ليصير:

TxtSel.SelectionStart += 1

وضع المؤشر في مربع النص ' ( )TxtSel.Focus

اضغط F5 لتشغيل البرنامج، واكتب بعض الحروف في مربع النص، ثم جرب ما يلي:

١- حدد النص كله، واضغط الزر عدة مرات.. ستلاحظ أن بداية التحديد ستزيد باستمرار، لكن مع المحافظة على باقي النص المحدد.. هذا سيجعل طول التحديد بسبب خروج أول حرف من التحديد مع كل ضغطة للزر.. وسيستمر هذا إلى أن يصل بداية موضع التحديد إلى نهاية النص، حيث سيختفي التحديد، وسيوضع مؤشر الكتابة بعد آخر حرف في النص.

٢- حدد عددا من الحروف في منتصف النص، واضغط الزر عدة مرات.. ستجد أن موضع بداية التحديد يزيد مع كل ضغطة، لكن طول التحديد يظل ثابتا، مما يجعل التحديد كله يتحرك إلى الأمام.. هذا يحدث لأن أول حرف سيخرج من التحديد، بينما سيدخل حرف جديد بدلا منه من نهاية التحديد.. وعندما تصل نهاية التحديد إلى نهاية النص، سنعود إلى أول حالة جربناها، حيث سيخرج حرف من بداية التحديد، لكن دون أن يدخل مكانه أي حرف في نهاية التحديد، مما سيجعل طول التحديد يقل باستمرار إلى أن يختفي.

٣- اضغط بالفأرة قبل أول حرف في مربع النص لوضع مؤشر الفأرة هناك لكن دون تحديد أي نص.. اضغط الزر عدة مرات.. ستجد أن موضع مؤشر الكتابة يزيد باستمرار إلى أن يصل إلى نهاية النص.

يمكنك أيضا أن تضع زرا على النموذج اسمه BtnDecSelStart.. مهمته إنقاص موضع بداية التحديد.. لاحظ أن الحروف Dec في اسم الزر هي اختصار الكلمة Decrement بمعنى إنقاص.. انقر هذا الزر مرتين، واكتب ما يلي في حدث ضغطه:

```
If TxtSel.SelectionStart > 0 Then
   TxtSel.SelectionStart -= 1
   وضع المؤشر في مربع النص ' ( )TxtSel.Focus
End If
```

يمكنك استنتاج أن السطر:

```
TxtSel.SelectionStart -= 1
```

يكافئ السطر:

```
TxtSel.SelectionStart = TxtSel.SelectionStart - 1
```

وكلاهما يعمل على إنقاص موضع بداية التحديد بواحد.. أما الشرط الذي فحصناه في البداية، فهو للتأكد أن موضع بداية التحديد أكبر من صفر، حتىّ يمكن طرح ١ منه.
هذا الشرط يهدف إلى حماية البرنامج من حدوث أي خطأ، فلو كان موضع بداية التحديد يساوي صفرا، فطرح ١ منه يعني أن الموضع الجديد سيكون −١ وهذا خطا.
شغل البرنامج وجرب تأثير زيادة وإنقاص موضع البداية على النص المحدد في مربع النص.
لاحظ أنه لن يحدث أي شيء إذا كان موضع بداية التحديد = صفرا وضغطنا الزر.. هذا أداء جيد، ولن تحدث أية أخطاء في البرنامج.

طول التحديد SelectionLength:

تستخدم هذه الخاصية لمعرفة طول النص المحدّد Selected Text في مربع النص، أو بمعنى آخر: تستخدم لمعرفة عدد الحروف المحددة.. مثال:
لو وضعت الكود التالي في حدث ضغط أي زر، فإن ضغط الزر سيعرض رسالة فيها عدد الحروف المحددة في مربع النص:

```
MsgBox(TextBox1.SelectionStart)
```

كما تستخدم هذه الخاصية أيضا لتغيير طول النص المحدد في مربع النص.. هذه الجملة مثلا تحدد أربعة حروف:

```
TextBox1.SelectionStart = 4
```

لاحظ أن هذه الخاصية لن تتسبب في حدوث خطأ إذا وضعت بها عددا أكبر من طول النص، لأنها في هذه الحالة ستصحح قيمتها تلقائيا ليكون آخر حرف في النص هو آخر حرف في التحديد.
لكن انتبه جيدا، فهذه الخاصية ستسبب خطأ إذا وضعت بها رقما سالبا، وهذا أمر منطقي، فلم يسمع أحد من قبل عن عدد حروف سالب!.. لكن من المقبول أن تضع فيها الرقم صفر، مما يعني انه لا يوجد أي نص محدد في مربع النص حاليا.
تعال نجرب هذه الخاصية عمليا.. أضف زرا جديدا إلى النموذج في المشروع TextSelection، واجعل اسمه BtnSelLen، واكتب في حدث ضغطه ما يلي:

```
TxtSel.SelectionLength += 1
TxtSel.Focus( )
```

هذا الكود يطلب زيادة طول التحديد بواحد كلما تم ضغط الزر، ويضع مؤشر الكتابة في مربع النص ليمكنك رؤية التغيير.
اضغط F5 لتشغيل البرنامج، واكتب بعض النص في مربع النص.. ضع مؤشر الكتابة في بداية النص ثم اضغط الزر BtnIncSelLen.. ستجد أن أول حرف قد تم تحديده.. اضغط الزر مجددا.. سيمتد التحديد ليشمل الحرف الثاني مع الحرف الأول.. وكلما ضغطت الزر فسيمتد التحديد ليشمل حرفا جديدا.. هذا يعني أن تغيير هذه الخاصية يحافظ على موضع بداية التحديد.
يمكنك أيضا أن تضع زرا آخر على النموذج اسمه BtnDecSelLen، مهمته إنقاص طول التحديد، ويجب هنا أن نتأكد أولا أن طول التحديد الحالي أكبر من صفر قبل إنقاصه بواحد، حتى لا نحصل على رقم سالب ويحدث خطأ.. هذا هو كود حدث ضغط هذا الزر:

```
If TxtSel.SelectionLength > 0 Then
   TxtSel.SelectionLength -= 1
   TxtSel.Focus( )
End If
```

تحديد كل النص:

ضع زرا على النموذج اسمه BtnSelAll1، وانقره مرتين بالفأرة، واكتب ما يلي في حدث ضغطه:

```
TxtSel.SelectionStart = 0
TxtSel.SelectionLength = TxtSel.TextLength
TxtSel.Focus( )
```

هذا الكود يجعل بداية التحديد قبل أول حرف، ويجعل عدد الحروف المحددة مساوية لطول النص الموجود في مربع النص.. هذا يعني أنه سيتم تحديد كل النص الموجود في مربع النص.
الجدير بالذكر أن هناك طريقة مختصرة لتحديد كل النص، وذلك باستخدام الوسيلة SelectAll.. ضع زرا آخر على النموذج اسمه BtnSelAll2 واكتب ما يلي في حدث ضغطه:

```
TxtSel.SelectAll( )
TxtSel.Focus( )
```

لو شغلت البرنامج وجربت أيا من الزرين، فستجده يحدد كل النص الموجود في مربع النص.

النصّ المحدّد SelectedText:

استخدم هذه الخاصية لقراءة الجزء المحدّد حاليا من النصّ في مربع النص.
ضع زرا جديدا على النموذج في المشروع TextSelection، واجعل اسمه BtnReadSelText، واكتب ما يلي في حدث ضغطه:

Me.Text = TxtSel.SelectedText
TxtSel.Focus( )

هذا الكود سيعرض النص المحدد في مربع النص، على شريط النموذج العلوي.. لاحظ أنه لو لم يكن هناك نص محدد في النص، فإن هذه الخاصية ستعيد نصا فارغا، ولن يظهر أي شيء على شريط النموذج العلوي.

وتتيح لك هذه الخاصية أيضا تغيير النص المحدد في مربع النص.. هذا معناه أن النص المحدد سيتم محوه، ووضع النص الجديد بدلا منه في نفس الموضع في مربع النص.. وليس شرطا أن يكون للنص الجديد نفس طول النص المحدد، فقد يكون أطول أو أقصر.. لاحظ أنه لم يكن هناك نص محدد، فإن النص الذي تضعه في هذه الخاصية ستتم كتابته في الموضع الحاليّ لمؤشّر الكتابة.

مثلا، يمكنك مسح النصّ المحدّد حاليا في مربع النص بوضع نص فارغ مكانه، وذلك بالجملة التالية:

TextBox1.SelectedText = ""

تعال نجرب هذا في المشروع TextSelection.. ضع زرا جديدا على النموذج اسمه BtnChangeSelText.. يمكنك تنظيم الأزرار لتبدو كما في الصورة:

انقر الزر BtnChangeSelText مرتين واكتب ما يلي في حدث ضغطه:

TxtSel.SelectedText = "نص جديد"
TxtSel.Focus( )

شغل البرنامج الآن.. لو ضغطت هذا الزر ومربع النص فارغ، فستتم كتابة "نص جديد" فيه.. لو وضعت مؤشر الكتابة في أي موضع من النص وضغط الزر فستتم كتابة "نص جديد" في هذا الموضع.. لو حددت أي جزء من النص وضغطت الزر، فستتم كتابة "نص جديد" بدلا من النص المحدد.

أهم وسائل مربع النص TextBox Methods:

يمتلك مربع النص بعض الوسائل الهامة، التي تستطيع من خلالها تنفيذ وظائف معينة خاصة بمربع النص.. لقد رأينا كيف نستخدم الوسيلة Focus لوضع المؤشر في مربع النص، وها هي ذي مزيد من الوسائل:

محو النص Clear:

تمحو هذه الوسيلة كل النص الموجود في مربع النص.. جرب هذا الكود في حدث ضغط اي زر:

```
TxtSel.Clear( )
```

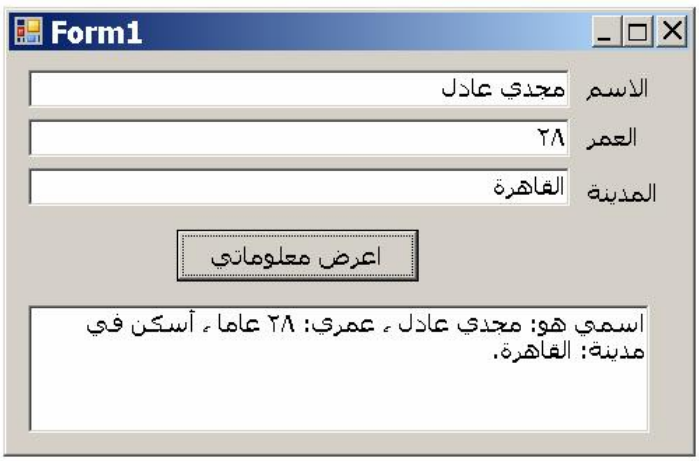
إضافة نص AppendText:

استخدم هذه الوسيلة لإضافة أي نص بعد آخر حرف مكتوب في مربع النص.. هذا النص يتم إرساله داخل القوسين ( ) التاليين لاسم الوسيلة، كالتالي:

```
TextBox1.AppendText("نص جديد")
```

الكود السابق يضيف الجملة "نص جديد" إلى النص الموجود حاليا في مربع النص، بعد آخر حرف مباشرة.. دعنا نأخذ مثالا.

ابدأ مشروعا جديدا اسمه TextBoxAppend، وصمم النموذج كما في الصورة:



فكرة هذا النموذج أن يكتب المستخدم اسمه في مربع النص الأول (TxtName)، وعمره في مربع النص الثاني (TxtAge)، ومدينته في مربع النص الثالث (TxtCity)، ويضغط الزر (BtnInfo) لعرض معلوماته في مربع النص السفلي (TxtInfo)، والذي جعلناه متعدد الأسطر (MultiLine = True).. وستتم كتابة معلومات المستخدم على الصيغة:

اسمي هو: ---- ، عمري: ---- عاما ، أسكن في مدينة ---- .

دعنا نكتب كود الزر.. انقره مرتين بالفأرة، واكتب ما يلي في حدث ضغطه:

```
TxtInfo.Clear( )
TxtInfo.AppendText("اسمي هو: ")
TxtInfo.AppendText(TxtName.Text)
TxtInfo.AppendText(" ، عمري: ")
TxtInfo.AppendText(TxtAge.Text)
TxtInfo.AppendText(" عاما")
TxtInfo.AppendText(" ، أسكن في مدينة: ")
TxtInfo.AppendText(TxtCity.Text)
TxtInfo.AppendText(".")
```

لاحظ أننا استخدمنا الوسيلة Clear أولا لمحو أي نص موجود سابقا في مربع النص.. لو لم نفعل هذا وضغط المستخدم الزر مرتين فستظهر معلوماته مرتين متتاليتين في مربع النص.

بعد هذا استخدمنا الوسيلة AppendText لإضافة النصوص إلى نهاية مربع النص، وبهذا استطعنا لصق النصوص بجوار بعضها:

- في البداية أضفنا الجملة "اسمي هو: ".
- ثم أضفنا بعدها النص الموجود في مربع النص TxtName.
- ثم بعد ذلك لصقنا الجملة " ، عمري: ".. لاحظ اهتمامنا بوضع علامات الترقيم كالفاصلة والنقطتين.. نحن أحرار في تنسيق النص بالشكل الذي نريده.
- وبنفس الطريقة أضفنا عمر الشخص ومدينته، ووضعنا نقطة توقف في النهاية.

شغل البرنامج بضغط F5، واكتب بياناتك في مربعات النصوص، واضغط الزر لترى معلوماتك في مربع النص السفلي.

لاحظ أنه لا يجب على المستخدم الكتابة في مربع النص TxtInfo، لهذا من الأذكى أن تجعله للقراءة فقط، بأن تجعل للخاصية ReadOnly القيمة True.

تراجع Undo:

استخدم هذه الوسيلة للتراجع عن آخر تغيير تم في مربع النص، بحيث يعرض مربع النص، آخر نص الذي كان موجودا فيه قبل إجراء عملية التغيير.

لاحظ أن استدعاء هذه الوسيلة مرة ثانية يعتبر عملية إعادة Redo!

تعال نأخذ مثالا عمليا لنفهم هذا الأمر جيدا.. ابدأ مشروعا جديدا اسمه EditOperations، وضع مربع نص على النموذج اسمه TxtEditor، وضع زرا على النموذج اسمه BtnUndo، ولنقره مرتين بالفأرة، واكتب ما يلي في حدث ضغطه:

```
TxtEditor.Undo( )
TxtEditor.Focus( )
```

اضغط F5 لتشغيل البرنامج.. اكتب الكلمة "مصر" في مربع النص ثم اضغط الزر.. سيؤدي هذا إلى محو الكلمة "مصر" من مربع النص.. هذه عملية تراجع عن الكتابة..

اضغط الزر مرة أخرى.. سيؤدي هذا إلى عرض الكلمة "مصر" في مربع النص مرة أخرى.. هذه عملية تراجع عن التراجع، أو بمعنى آخر: عملية إعادة Redo.

نسخ Copy:

تقوم هذه الوسيلة بنسخ النص المحدد حاليا في مربع النص إلى لوحة قصاصات الويندوز Clipboard.. لتجربة هذا، ضع زرا اسمه BtnCopy على النموذج في المشروع EditOperations، واكتب ما يلي في حدث ضغطه:

```
TxtEditor.Copy( )
TxtEditor.Focus( )
```

قص Cut:

تقوم هذه الوسيلة بقص النص المحدد حاليا في مربع النص، ووضعه في لوحة قصاصات الويندوز ClipBoard.. لتجربة هذا، ضع زرا اسمه BtnCut على النموذج في المشروع EditOperations، واكتب ما يلي في حدث ضغطه:

```
TxtEditor.Cut( )
TxtEditor.Focus( )
```

لصق Paste:

تقوم هذه الوسيلة بلصق النص المحفوظ في لوحة القصاصات إلى موضع مؤشر الكتابة الحالي في مربع النص، ولو كان هناك نص محدد في مربع النص في تلك اللحظة، فسيتم حذفه ولصق النص من لوحة القصاصات بدلا منه.

لتجربة هذا، ضع زرا اسمه BtnPaste على النموذج في المشروع EditOperations، واكتب ما يلي في حدث ضغطه:

```
TxtEditor.Paste( )
TxtEditor.Focus( )
```

اضغط F5 الآن وشغل البرنامج.. اكتب أي نص في مربع النص، وحدد جزءا منه وجرب أزرار النسخ والقص واللصق.

أهم أحداث مربع النص:

يمتلك مربع النص عددا من الأحداث Events، وفيما يلي نتعرف على أهمها:

حدث تغير النص TextChanged Event:

ينطلق هذا الحدث عند تغير النص الموجود في مربع النص، سواء عند قيام المبرمج بتغيير قيمة الخاصية Text من الكود، أو بسبب تغيير المستخدم للنص الموجود بمربع النص، بالحذف أو الإضافة أو التعديل أو القص أو اللصق.

دعنا نأخذ مثالا على هذا الحدث.. ابدأ مشروعا جديدا اسمه TextChanged_Sample، وضع على النموذج ثلاثة مربعات نصوص، واضبط خصائصها كما هو موضح في الجدول التالي:

الخاصية	مربع النص الأول	مربع النص الثاني	مربع النص الثالث
Name	TxtNormal	TxtUpper	TxtLower
ReadOnly	False	True	True
CharacterCasing	Normal	Upper	Lower
ملحوظة حول الوظيفة	يعرض حروف النص في حالتها الطبيعية كما يدخلها المستخدم	يعرض حروف النص في صورة كبيرة Capital	يعرض حروف النص في صورة صغيرة Small

بعد أن تنتهي من ضبط خصائص مربعات النص، انقر مرتين بالفأرة فوق مربع النص الأول.. سينقلك هذا إلى صفحة الكود، حيث سترى معالج حدث تغير النص كالتالي:

```
Private Sub TxtNormal_TextChanged(ByVal sender As Object, _
        ByVal e As EventArgs) Handles TxtNormal.TextChanged

End Sub
```

هذا هو المكان الذي سنكتب فيه الكود الخاص بنا.. هذا الكود سيكون بسيطا للغاية، فهذا الحدث ينطلق كلما كتب المستخدم حرفا في مربع النص الأول، لهذا سنستخدمه لنسخ نفس النص الموجود في مربع النص الأول إلى مربعي النصين الثاني والثالث.. هكذا يجب أن يبدو كود هذا الحدث:

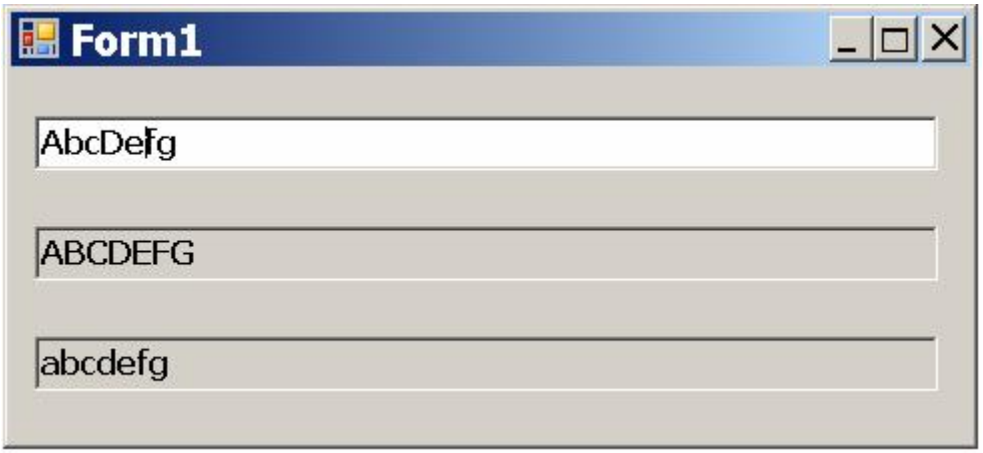
```
Private Sub TxtNormal_TextChanged(ByVal sender As Object, _
        ByVal e As EventArgs) Handles TxtNormal.TextChanged
    TxtUpper.Text = TxtNormal.Text
    TxtLower.Text = TxtNormal.Text
End Sub
```

الكود في منتهى البساطة، وهو يتكون من سطرين اثنين فحسب:

- الأول يقرأ النص الموجود في الخاصية Text لمربع النص الأول (TxtNormal) ويضعه في الخاصية Text لمربع النص الثاني (TxtUpper).
- والثاني يقرأ النص الموجود في الخاصية Text لمربع النص الأول (TxtNormal) ويضعه في الخاصية Text لمربع النص الثالث (TxtLower).

شغل البرنامج الآن بضغط F5.. اكتب أي حرف إنجليزي في مربع النص الأول.. سيؤدي هذا إلى استدعاء حدث تغير النص TextChanged، وسيتم تنفيذ الكود الذي كتبناه في معالج هذا الحدث، لهذا سيظهر الحرف في صورة كبيرة في مربع النص

الثاني، وفي صورة صغيرة في مربع النص الثالث.. وسيتكرر حدوث هذا مع كل حرف تكتبه في مربع النص الأول.



دعنا نطور هذا المثال قليلا.. فبدلا من عرض الحروف الكبيرة والصغيرة في اثنين من مربعات النص تم ضبط خصائصهما ليكونا للقراءة فقط، دعنا نعرضهما في لافتتين.

ابدأ مشروعا جديدا اسمه TextChanged_Sample2، وضع على النموذج الأدوات التالية:

١- مربع نص اسمه TxtNormal.

٢- لافتة Label اسمها LblUpper لعرض الحروف الكبيرة.. واجعل للخاصية BorderStyle القيمة Fixed3D لجعل هذه اللافتة مجسمة.

٣- لافتة Label اسمها LblLower لعرض الحروف الصغيرة.. واجعل للخاصية BorderStyle القيمة Fixed3D لجعل هذه اللافتة مجسمة.

سنستخدم هنا حدث تغير النص TextChanged لعرض النص في اللافتتين.. لكن المشكلة التي ستواجهنا هي أن اللافتة لا تملك خاصية حالة الحروف CharacterCasing، وهذا يعني أن علينا تغيير حالة الأحرف بأنفسنا.. لفعل هذا يمكننا استخدام الوسيلتين التاليتين، وهما من وسائل فئة النص String Class التي سنتعرف عليها فيما بعد:

ToUpper	تقوم بتحويل كل حرف النص إلى حروف كبيرة.
ToLower	تقوم بتحويل كل حرف النص إلى حروف صغيرة.

والآن، انقر مرتين بالفأرة على مربع النص TxtNormal، واكتب في حدث تغير النص ما يلي:

```
LblUpper.Text = TxtNormal.Text.ToUpper
LblLower.Text = TxtNormal.Text.ToLower
```

هذا هو نفس الكود الذي كتبناه سابقا، مع اختلاف واحد، هو استخدام الوسيلتين ToUpper و ToLower.
لاحظ أن الخاصية Text تعيد كائنا من نوع فئة النص String Class، لهذا استخدمنا الوسيلتين ToUpper و ToLower التابعتين لهذا الكائن مباشرة بعد اسم الخاصية Text.. هذا اختصار على المبرمج يوفر عليك كتابة الكثير من الكود، ولو لم نستخدم هذا الاختصار، لصار الكود كالتالي:

```
Dim S As String
S = TxtNormal.Text
LblUpper.Text = S.ToUpper
LblLower.Text = S.ToLower
```

حيث استخدمنا متغيرا نصيا String Variable اسمه S، ووضعنا فيه قيمة خاصية النص Text التابعة لمربع النص، وبعد ذلك استخدمنا الوسيلتين ToUpper و ToLower من خلال المتغير S.. واضح طبعا أن الكود الأول أكثر اختصارا من هذا الكود، لكنّ كليهما يؤدي الوظيفة نفسها، وأنت حر في استخدام الكود الذي يريحك أكثر.
يمكنك الآن تجربة البرنامج.. ستجده يعمل بنفس كفاءة البرنامج السابق.

حدث ضغط الزرّ KeyPress Event:
ينطلق هذا الحدث كلما ضغط المستخدم زرا من لوحة المفاتيح.

ملحوظة هامة:
ينطلق الحدث KeyPress بعد ضغط الحرف، لكن قبل كتابته في مربع النص، لهذا يمكن استخدام الحدث KeyPress لإلغاء كتابة الحرف.. بينما الحدث ينطلق TextChanged بعد كتابة الحرف في مربع النص وتغير قيمة الخاصية Text.

حدث الدخول Enter:
ينطلق هذا الحدث عندما يستقبل مربع النص العلامة الضوئية (مؤشر الكتابة) ويصبح نشيطا Active.. يحدث هذا بالطرق التالية:

١- بضغط زر الفأرة الأيسر فوق مربع النص.
٢- بضغط زر Tab من لوحة المفاتيح للتنقل بين الأدوات والوصول إلى مربع النص.
٣- باستخدام الوسيلة Focus في الكود لوضع مؤشر الكتابة في مربع النص.

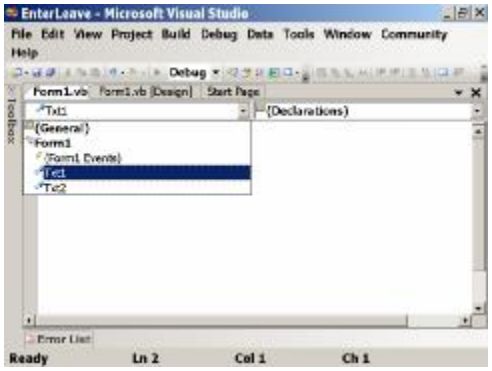
حدث الخروج Leave:
ينطلق هذا الحدث عندما تغادر العلامة الضوئية مربع النص، ويصبح غير نشيط.. لاحظ أن المؤشر بعد مغادرة مربع النص الحالي قد يدخل مربع نص آخر.. هذا يعني أن الحدث Leave سنطلق أولا لإعلان مغادرة مربع النص الأول، وسيليه مباشرة انطلاق الحدث Enter الخاص بمربع النص الثاني.

دعنا الآن نأخذ مثالا عمليا شيقا، نستخدم فيه الحدثين Enter و Leave.. في هذا المثال سنستخدم الحدث Enter لتلوين خلفية مربع النص الفعال (الذي يوجد به مؤشر الكتابة) باللون الأصفر، وسنستخدم الحدث Leave لإعادة لون الخلفية مربع النص إلى اللون الأبيض عند مغادرة المؤشر له.. والآن اتبع معي هذه الخطوات:

١- ابدأ مشروعا جديدا اسمه EnterLeave.

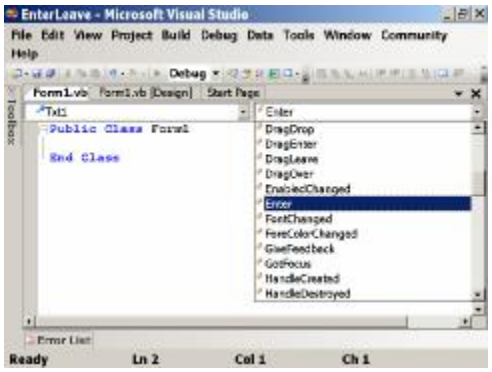
٢- ضع على النموذج نسختين من الأداة TextBox، وأعطهما الاسمين Txt1 و Txt2.

٣- اضغط بزر الفأرة الأيمن فوق أي جزء خال من سطح النموذج، ومن القائمة الموضعية Context Menu اضغط الأمر View Code.. سيؤدي هذا إلى عرض نافذة الكود الخاصة بالنموذج.



٤- في أعلى نافذة الكود ستجد قائمة على اليسار، مكتوب فيها Form1.. اضغط هذه القائمة لإسدالها.. ستجد في هذه القائمة جميع أسماء الأدوات الموضوعة على النموذج.. اختر منها اسم مربع النص الأول Txt1 بضغطه مرة واحدة بالفأرة.

٥- توجد أعلى نافذة محرر الكود قائمة أخرى على اليمين.. هذه القائمة تعرض أحداث الأداة التي تم اختيارها في القائمة الموجود على اليسار.. ونظرا لاختيارنا مربع النص Txt1 من القائمة اليسرى، فإن القائمة اليمنى ستعرض كل أحداثه. اضغط هذه القائمة لإسدالها، واختر منها الحدث Enter بضغطه مرة واحدة بالفأرة.. سيؤدي هذا إلى كتابة كود معالج هذا الحدث في صفحة الكود، وسيكون كالتالي:




```
Private Sub Txt1_Enter(ByVal sender As Object, _
        ByVal e As EventArgs) Handles Txt1.Enter

End Sub
```

كل ما علينا الآن هو أن نكتب داخل هذا الإجراء الكود الذي يغير لون خلفية مربع النص Txt1 إلى اللون الأصفر.. سيكون هذا الكود كالتالي:

```
Private Sub Txt1_Enter(ByVal sender As Object, _
        ByVal e As EventArgs) Handles Txt1.Enter
    Txt1.BackColor = Color.Yellow
End Sub
```

كما ترى.. لم نحتج إلى أكثر من سطر واحد من الكود، استخدمنا فيه الخاصية BackColor لمربع النص Txt1 حيث وضعنا فيها اللون الأصفر Color.Yellow.

لاحظ أن العشرات من أسماء الألوان موجودة كخصائص داخل الكائن Color.. والجدول التالي يلخص لك بعض أسماء الألوان الشهيرة:

Color.Yellow	أصفر
Color.Red	أحمر
Color.Blue	أزرق
Color.Green	أخضر
Color.Orange	برتقالي
Color.Gray	رمادي
Color.White	أبيض
Color.Brown	بني
Color.Black	أسود

٦- تأكد أن الأداة Txt1 ما زالت محددة في القائمة اليسرى، ومن القائمة اليمنى اضغط اسم الحدث Leave.. هذا سيؤدي إلى كتابة معالج هذا الحدث في صفحة الكود.. كل ما سنفعله هو أن نكتب في هذا المعالج سطرا واحدا من الكود يعيد لون خلفية مربع النص إلى اللون الأبيض.. هكذا يجب أن يكون كود هذا المعالج:

```
Private Sub Txt1_Leave(ByVal sender As Object, _
        ByVal e As EventArgs) Handles Txt1.Leave
    Txt1.BackColor = Color.White
End Sub
```

يمكنك تشغيل البرنامج الآن وتجربته.. ستجد أن خلفية مربع النص الأول تصير صفراء اللون عندما تضع فيه مؤشر الكتابة، ثم تصير بيضاء اللون حينما تنقل المؤشر إلى مربع النص الثاني.

٧- نريد الآن أن نفعل المثل مع مربع النص الثاني.. اتبع نفس الخطوات السابقة، مع تغيير شيء واحد فقط، هو اختيار مربع النص Txt2 من القائمة اليسرى.. هكذا يجب أن يكون كود الحدثين Enter و Leave لمربع النص الثاني:

```
Private Sub Txt2_Enter(ByVal sender As Object, _
        ByVal e As EventArgs) Handles Txt2.Enter
    Txt2.BackColor = Color.Yellow
End Sub

Private Sub Txt2_Leave(ByVal sender As Object, _
        ByVal e As EventArgs) Handles Txt2.Leave
    Txt2.BackColor = Color.White
End Sub
```

شغل البرنامج الآن وجربه.. ستجد أن خلفية مربع النص الأول تصير صفراء اللون عندما تضع فيه مؤشر الكتابة بينما تكون خلفية مربع النص الثاني بيضاء، وعندما تنقل المؤشر إلى مربع النص الثاني ستصير خلفيته صفراء بينما ستعود خلفية مربع النص الأول إلى اللون الأبيض.

مربع الرسالة Message Box:

لعلك لاحظت وأنت تستخدم برامج الويندوز، أنها تتحاور معك من خلال مربع الرسالة Message Boxe، الذي يظهر ليخبرك بأمر معين كحدوث خطأ مثلا، أو تنبيهك إلى أنك ستحذف ملفا أو ما إلى ذلك.

- بعض مربعات الرسائل يعرض رسالة وزر OK فقط.
- وبعضها يعرض زر OK أو زر Yes للموافقة على العملية وزر Cancel أو زر No لإلغائها.
- وبعضها يعرض زر Yes للموافقة، وزر No للرفض وزر Cancel للإلغاء.

ملحوظة:

يمكن عرض نص الرسالة للمستخدم في لافتة Label أو في مربع رسالة.. لكن هناك اختلافات جوهرية بين الطريقتين:

- فمن الممكن ألا ينتبه المستخدم إلى النص المكتوب في اللافتة، ويواصل تشغيل البرنامج بطريقة عادية.
- بينما مربع الرسالة يظهر فوق النموذج ويمنع المستخدم من مواصلة التعامل مع البرنامج إلا لو ضغط أحد أزرار مربع الرسالة لإغلاقه.. لهذا يُستخدم مربع الرسالة لعرض رسائل الخطأ والتحذيرات التي يجب ألا ينتبه إليها المستخدم أولا، كتذكيره مثلا بأنه لم يحفظ ما كتبه في ملف أو ما شابه.

نريد الآن أن نتعلم كيف نعرض هذه الرسائل للمستخدم، وكيف نعرف الزر الذي ضغطه المستخدم إن كان مربع الرسالة يعرض أكثر من زر.
هناك طريقتان في فيجيوال بيزيك دوت نت لفعل هذا:

١- باستخدام الدالة MsgBox وهي إحدى دوال فيجيوال بيزيك القديمة التي استخدمها المبرمجون قبل ظهور نظام دوت نت، وما زالت متاحة للاستخدام حتى اليوم في فيجيوال بيزيك دوت نت.

٢- باستخدام الفئة MessageBox، وهي إحدى فئات إطار العمل Framework، لهذا يمكن استخدامها من أية لغة على نظام دوت نت، مثل فيجيوال بيزيك دوت نت وسي شارب وغيرهما.

وفيما يلي، سنتعرف على هاتين الطريقتين، لكن قبل أن نفعل هذا نحتاج إلى التعرف على بعض التعريفات الأساسية.

الدالة Function:

هي مقطع من الكود يؤدي وظيفة محددة وله اسم معين، يمكن تنفيذه بكتابة هذا الاسم.. ويمكن استدعاء الدالة أكثر من مرة من أكثر من موضع في البرنامج.
ويمكن تعريف الدالة داخل الفئة Class.. في هذه الحالة تسمى الدالة وسيلة Method.. إذن فالوسائل التي تعرفنا عليها هي دوال مكتوبة داخل الفئات.

المعامل Parameter:

ويسمى أيضا Argument، وهو بيان Data أو قيمة Value يتم إرسالها إلى الدالة لتجري عليها حساباتها.. ويرسل المبرمج هذه القيم داخل قوسين ( ) بعد اسم الدالة.
ويمكن أن يكون للدالة معامل واحد.. مثلا: الدالة AppendText المكتوبة في فئة مربع النص TextBox تحتاج إلى معامل واحد لترسل من خلاله إلى الدالة، النص الذي سيتم إضافته في نهاية مربع النص.. مثال:

```
TextBox1.AppendText("new string")
```

وقد تحتاج الدالة إلى أكثر من معامل، وفي هذه الحالة يجب أن يرسل المبرمج البيانات بنفس ترتيب المعاملات مع وضع فاصلة , بينها.. مثال:

```
MsgBox("Hello my friend.", MsgBoxStyle.YesNo, "Hello")
```

وأحيانا قد لا تحتاج الدالة إلى أية معاملات.. فمثلا: الدالة Close المكتوبة في فئة النموذج لا تحتاج إلى أية بيانات لإغلاق النموذج Form، لهذا ليست لها معاملات:

```
Me.Close( )
```

المعامل الاختياري Optional Parameter:

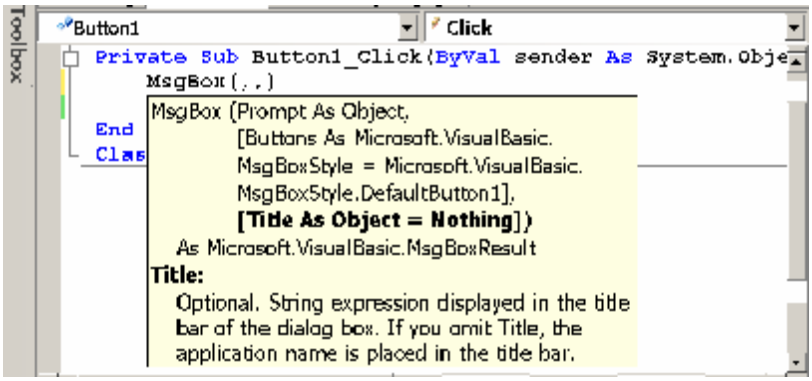
هو معامل يمكنك أن ترسل قيمته إلى الدالة أو لا ترسلها، وفي حالة عدم إرسالها تستخدم الدالة قيمة افتراضية Default Value لهذا المعامل معرفة لديها مسبقا.

استخدام الاستشعار الذكي IntelliSense مع الدوال:

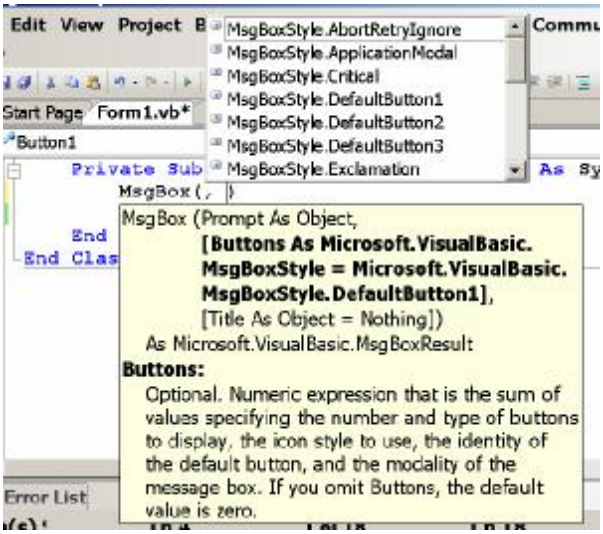
يمكنك استخدام الاستشعار الذكي IntelliSense لإكمال اسم الدالة التي تكتبها.. فبعد كتابة أول حرف أو حرفين يمكنك ضغط Ctrl+Space من لوحة المفاتيح، لتظهر قائمة بكل الأسماء الممكنة التي تبدأ بالحروف التي كتبتها.

كما يمنحك الاستشعار الذكي IntelliSense تسهيلات كبيرة في التعامل مع معاملات الدوال:

- فبمجرد كتابة قوس البداية تظهر لافتة تعرض لك صيغة تعريف المعاملات، وبهذا تعرف أسماءها وأنواعها وترتيبها.
- وكلما كتبت فاصلة يظهر شرح موجز أسفل اللافتة يشرح وظيفة المعامل الذي ستقوم بإرسال البيانات إليه في هذا الموضع، وللإيضاح فإن تعريف هذا المعامل في صيغة الدالة يظهر بخط سميك.. في الصورة التالية مثلا يظهر بخط سميك معامل اسمه Title، وفي أسفل اللافتة يوجد سرح لوظيفة هذا المعامل.. تلاحظ أيضا أن هذا المعامل موضوع بين قوسين مضلعين [ ].. هذا لتنبيهك إلى أنه معامل اختياري، وتستطيع ألا ترسله إلى الدالة.. لاحظ أن الرمز [ ] لا يستخدم في الكود، لكنه يستخدم في وصف المعاملات فقط.



- وإن كانت هناك قيم خاصة للمعامل الحالي، فإن الاستشعار الذكي يعرض قائمة خاصة، تظهر فيها كل هذه القيم ليمكنك اختيارها منها، وذلك كما هو موضح في الصورة التالية:



دالة مربع الرسالة MsgBox:

المقطع Msg في اسم هذه الدالة هو اختصار الكلمة الإنجليزية Message بمعنى رسالة، والمقطع Box هو الكلمة الإنجليزية التي تعني صندوق في ترجمتها الحرفية، ونترجمها نحن إلى مربع لأنها أكثر دلالة.

وأبسط صيغة للدالة MsgBox تقبل معاملا واحدا، يسمى معامل التوجيه Prompt Argument، وهو يستقبل نص الرسالة التي سيتم عرضها للمستخدم لتخبره ما المطلوب منه بالضبط.

ابدأ مشروعا جديدا اسمه MsgBoxSample وضع زرا على النموذج، وجرب الجملة التالية في حدث ضغط الزر:

MsgBox("مرحبا بك في برنامجنا")

هذه الجملة ستعرض رسالة عليها الزر Ok وعنوانها المكتوب على شريط النموذج العلوي هو MsgBoxSample وهو اسم برنامجنا، وتعرض النص "مرحبا بك في برنامجنا".

وللدالة MsgBox معاملان اختياريان، يمكن أن ترسلهما إليها إذا أردت، لتتحكم من خلالهما في التفاصيل المعروضة على مربع الرسالة، وهذان المعاملان هما:

- المعامل الثاني ويسمى معامل الأزرار Buttons Arguments.
- المعامل الثالث، ويسمى معامل العنوان Title Argument.

معامل الأزرار Buttons Arguments:

يسمح لنا هذا المعامل بالتحكم في شكل مربع الرسالة، واتجاه نص الرسالة (من اليسار إلى اليمين أم العكس) وعدد الأزرار والأيقونات Icons التي يعرضها مربع الرسالة.. ويأخذ هذا المعامل إحدى القيم التالية:

١- قيم تتحكم في عدد ونوع الأزرار:

MsgBoxStyle.OkOnly	تعرض الرسالة زر الموافقة Ok وحده.
MsgBoxStyle.OkCancel	تعرض الرسالة زر الموافقة Ok وزر الإلغاء Cancel.
MsgBoxStyle.YesNo	تعرض الرسالة زر "نعم" Yes وزر "لا" No.
MsgBoxStyle.YesNoCancel	تعرض الرسالة ثلاثة أزرار: Yes و No و Cancel.
MsgBoxStyle.AbortRetryIgnore	تعرض الرسالة ثلاثة أزرار: إلغاء Abort وإعادة المحاولة Retry وتجاهل Ignore.
MsgBoxStyle.MsgBoxHelp	تعرض الرسالة زرا إضافيا بجانب أزرارها الأخرى، هو زر المساعدة أو الاستعلام Help، لتسمح للمستخدم عند ضغطه بالحصول على معلومات تشرح سبب المشكلة.

٢- قيم تتحكم في الزر الافتراضي:

الزر الافتراضي هو الزر الذي يكون محددا Selected على مربع الرسالة، بحيث يتم ضغطه إذا ضغط المستخدم مسافة أو Enter من لوحة المفاتيح.

MsgBoxStyle.DefaultButton1	أول زر في الرسالة هو الزر الافتراضي.
MsgBoxStyle.DefaultButton2	ثاني زر في الرسالة هو الزر الافتراضي.
MsgBoxStyle.DefaultButton3	ثالث زر في الرسالة هو الزر الافتراضي.

٣- قيم تتحكم في اتجاه نص الرسالة:

MsgBoxStyle.MsgBoxRight	أزرار الرسالة تظهر من اليمين إلى اليسار، ومحاذاة نص الرسالة من اليمين إلى اليسار.
MsgBoxStyle.MsgBoxRtlReading	هذا الثابت مفيد عند كتابة نص الرسالة بالعربية، حتى تظهر الأقواس وعلامات الترقيم وأية حروف أجنبية ضمن النص العربي بشكل صحيح.

قيم تتحكم في الأيقونات التي يعرضها مربع الرسالة:

MsgBoxStyle.Critical 	تعرض الرسالة أيقونة الموقف الحرج، وهي على شكل علامة × داخل دائرة حمراء للدلالة على أن العملية التي يريد المستخدم القيام بهـا خطيـرة، كحذف الملفات مثلا.
MsgBoxStyle.Exclamation 	تعرض الرسالة أيقونة التعجب، وهي على شـكل علامة تعجب داخل مثلث أصفر على سبيل التنبيه والتحذير.
MsgBoxStyle.Information 	تعرض الرسالة أيقونة المعلومات، وهـي علـى شكل حرف I داخل دائرة للدلالة على أنها تقـدم معلومة هامة.
MsgBoxStyle.Question 	تعرض الرسالة أيقونة الاستفهام، وهي على شكل علامة استفهام ؟ داخل دائرة للدلالة على أن هذا السؤال يحتاج إلى إجابة من المستخدم.

لعل السؤال الذي يدور في ذهنك الآن هو: ماذا لو أردت التحكم في هذه الأنواع الأربعـة من القيم معا؟.. كيف يمكن مـثلا أن أعـرض على الرسالة الزرين Yes و No مـع أيقونـة التعجب؟

والإجابة بسيطة: فأنت تستطيع دمج أكثـر مـن قيمة من هذه القيم معا باستخدام علامة الجمـع +.. لهذا يمكنك عـرض Yes و No وأيقونـة التعجب معا كما ترى في الصورة كالتالي:

```
MsgBox("هل تريد إغلاق البرنامج؟", _
        MsgBoxStyle.YesNo + MsgBoxStyle.Exclamation)
```

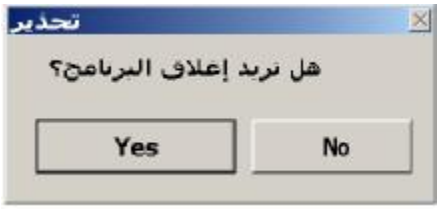
ملحوظة:

يمكن استخدام المعامل Or بدلا من علامة الجمع لتأدية نفس الوظيفة، مع ملاحظـة أن هذا سيجعل الاستشعار الذكي IntelliSense يعرض قائمة بـالقيم المختلفـة لمعامـل الأزرار بعد كتابة الكلمة Or، بينما لا يحدث هذا عند استخدام العلامة +.

```
MsgBox("هـل تريـد إغـلاق البرنـامج؟", MsgBoxStyle.YesNo Or
MsgBoxStyle.Exclamation)
```

معامل العنوان Title Argument:

يستقبل هذا المعامل العنوان الذي سيتم عرضه على الشريط العلوي لمربع الرسالة.. أما إذا لم ترسل أية قيمة إلى هذا المعامل، فستحصل الدالة MsgBox تلقائيا على اسم البرنامج الذي تعمل فيه، وتعرضه كعنوان لمربع الرسالة.



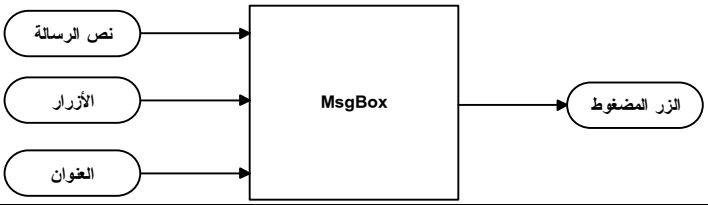
مثال: الكود التالي يعرض العنوان "تحذير" على مربع الرسالة:

MsgBox("هل تريد إغلاق البرنامج؟", MsgBoxStyle.YesNo, "تحذير")

ملحوظة:
إذا لم تكن تريد إرسال أية قيمة إلى المعامل الثاني لتعرض الرسالة زر الموافقة OK فقط، فيجب عليك أن تضع الفاصلة الخاصة بالمعامل الثاني دون أن تكتب بعدها شيئا.. مثال: **MsgBox("هل تريد إغلاق البرنامج؟", , "تحذير")** لو لم تفعل هذا فسيحدث خطأ.. مثلا لو كتبت: **MsgBox("هل تريد إغلاق البرنامج؟", "تحذير")** ستفهم فيجيوال بيزيك من هذا الكود أنك ترسل المعاملين الأول والثاني إلى الدالة، لهذا سيحدث خطأ عند تنفيذ البرنامج، لأن المعامل الثاني لا يقبل أية بيانات من نوع النص String.. بينما لو وضعت فاصلة إضافية فستحفظ موضع المعامل الذي لم ترسله، وبهذا تفهم فيجيوال بيزيك المعاملات بالترتيب الصحيح.

القيمة العائدة من الدالة MsgBox:

إذا نظرنا إلى الدالة كآلة في مصنع، فإن المواد الخام التي يتم تقديمها إلى هذه الدالة هي المعاملات Parameters، بينما الناتج الذي تعيده إلينا هذه الدالة هو القيمة العائدة Return Value، كما هو موضح في الرسم المبسط التالي:



تعريف: القيمة العائدة Return Value:
هي نتيجة الدالة، التي نقرؤها منها.

وتعيد الدالة MsgBox إحدى القيم التالية:

MsgBoxResult.Ok	المستخدم ضغط الزر OK.
MsgBoxResult.Cancel	المستخدم ضغط زر الإلغاء Cancel.
MsgBoxResult.Yes	المستخدم ضغط زر Yes.
MsgBoxResult.No	المستخدم ضغط الزر No.
MsgBoxResult.Abort	المستخدم ضغط زر الإلغاء Abort.
MsgBoxResult.Ignore	المستخدم ضغط زر التجاهل Ignore.
MsgBoxResult.Retry	المستخدم ضغط الزر Retry .

وعليك فحص القيمة العائدة من الرسالة للتأكد من نوع الزر المضغوط باستخدام جملة If، وبالتالي اتخاذ رد الفعل المناسب.

مثال:

افترض أنك قابلت شخصا وشعرت أن ملامحه مألوفة لك، وتظن أنه صديق قديم اسمه هاني.. أفضل ما تفعله عندئذ هو أن تتجه نحوه وتسأله بأدب: "هل أنت هاني؟".. فإن أجاب بنعم، قلت له بسعادة: "أهلا يا هاني"، وإن أجاب بلا قلت له معتذرا: "آسف جدا.. ظننتك شخصا أعرفه".

اكتب برنامجا يعبر عن هذا الحوار، مفترضا الآتي:

- الشخص الذي تكلمه هو مستخدم البرنامج.
- اللقاء بينكما يتم عندما يضغط المستخدم الزر Button في برنامجك.
- الحوار بينكما يتم باستخدام مربعات الرسائل.

الحل:

دعنا في هذا حل هذا المثال نكتب خطوات الحل كاملة.. هذا هو ما يجب فعله في كل مسألة، لكننا فقط كنا نختصر الخطوات لتوفير الصفحات!

هذه هي الخطوات كاملة:

- من القائمة File، اضغط الأمر New Project، وفي قائمة لغات البرمجة اختر Visual Basic، وفي قائمة أنواع المشاريع اختر Window Forms، وفي خانة الاسم Name اكتب اسم المشروع MyFriend، واضغط الزر Ok.
- من القائمة File اضغط Save All، وفي نافذة الحفظ حدد المسار الذي تريد حفظ البرنامج عليه، واضغط الزر OK.
- من القائمة View اضغط الأمر Toolbox لعرض صندوق الأدوات، وافتح الشريط Common Controls وانقر مرتين بالفأرة على أيقونة الزر Button لإضافته إلى النموذج.
- حدد الزر واضغط F4 لعرض نافذة الخصائص.. غير اسم الزر Name إلى BtnMsg، وغير عنوانه Text إلى: مرحبا.

- انقر الزر مرتين بالفأرة Double-Click.. سيعرض هذا صفحة محـــرر الكـــود، وفيها معالج حدث ضغط الزر BtnMsg_Click.. اكتب في هذا المعالج ما يلي:

```
If MsgBox("هل أنت هاني", MsgBoxStyle.YesNo, _
          "لحظة من فضلك") = MsgBoxResult.Yes Then
    MsgBox("أهلا يا هاني")
Else
    MsgBox("آسف جدا.. ظننتك شخصا أعرفه")
End If
```

لاحظ أن السطر الأول يفحص القيمة العائدة من الدالة MsgBox ويتأكد إن كان الـــزر المضغوط هو Yes.. لكي يحدث هذا، لا بد أن يتم عرض مربع الرسالة أولا ولا بد أن يضغط المستخدم زر Yes أو No.. وكما ترى في الكود، فإن الرسالة تسأله هل اســـمه هاني، فإن ضغط Yes، فإن الشرط سيكون صحيحا، وبالتالي سيتم عرض رسالة تقـــول له "مرحبا يا هاني"، وإن ضغط No فإن الشرط يكون خاطئا ويتم تنفيذ المقطـــع Else، حيث يتم عرض الرسالة: "آسف جدا.. ظننتك شخصا آخر".

فئة مربع الرسالة MessageBox Class:

تمتلك هذه الفئة وسيلة واحدة، هي وسيلة العرض Show، وهـــي تقـــوم بعـــرض مربـــع الرسالة على الشاشة.

ولهذه الوسيلة العديد من الصيغ، كل منها تعطيك تحكما ببعض أجزاء مربع الرسالة، لهـــذا يمكنك اختيار الصيغة المناسبة على حسب احتياجك.

وأبسط صيغة للوسيلة Show لها معامل واحد،، هو نص الرسالة التي سيعرضها مربـــع الرسالة.. مثلا: الكود التالي يعرض رسالة تقول لك "مرحبا":

```
MessageBox.Show("مرحبا")
```

وتوجد صيغة ثانية للوسيلة Show لها معامل إضافي، هو عنوان مربع الرسالة.. مـــثلا: الكود التالي يعرض رسالة تقول لك "مرحبا" وعنوانها "أهلا بك":

```
MessageBox.Show("مرحبا", "أهلا بك")
```

وتوجد صيغة ثالثة للوسيلة Show لها معامل إضافي، هو أزرار مربع الرسالة.. ويأخـــذ هذا المعامل أية قيمة من القيم التالية:

MessageBoxButtons.Ok	تعرض الرسالة زر الموافقة Ok وحده.
MessageBoxButtons.OkCancel	تعـــرض الرســـالة زر الموافقـــة Ok وزر الإلغاء Cancel.
MessageBoxButtons.RetryCancel	تعرض الرســـالة زر الإعـــادة Retry وزر الإلغاء Cancel.

MessageBoxButtons.YesNo	تعرض الرسالة زر "نعم" Yes وزر "لا" No.
MessageBoxButtons.YesNoCancel	تعرض الرسالة ثلاثة أزرار: Yes و No و Cancel.
MessageBoxButtons.AbortRetryIgnore	تعرض الرسالة ثلاثة أزرار: إلغاء Abort وإعادة المحاولة Retry وتجاهل Ignore.

مثال:

الكود التالي يعرض رسالة عنوانها "تأكيد"، تسأل المستخدم إن كان يريد إغلاق البرنامج، وعليها الزران Yes و No:

```
MessageBox.Show("هل تريد إغلاق البرنامج", "تأكيد", _
    MessageBoxButtons.YesNo)
```

وهناك صيغة رابعة للوسيلة Show، لها معامل رابع، تستطيع من خلاله التحكم في الأيقونة التي يعرضها مربع الرسالة، وذلك باستخدام أي من القيم التالية:

None	لا يتم عرض أية أيقونات على مربع الرسالة.
MessageBoxIcon.Error أو MessageBoxIcon.Stop 	تعرض الرسالة أيقونة الموقف الحرج، وهي على شكل علامة × داخل دائرة حمراء للدلالة على أن العملية التي يريد المستخدم القيام بها خطيرة، كحذف الملفات مثلا.
MessageBoxIcon.Exclamation أو MessageBoxIcon.Warning 	تعرض الرسالة أيقونة التعجب، وهي على شكل علامة تعجب داخل مثلث أصفر على سبيل التنبيه والتحذير.
MessageBoxIcon.Information 	تعرض الرسالة أيقونة المعلومات، وهي على شكل حرف I داخل دائرة للدلالة على أنها تقدم معلومة هامة.
MessageBoxIcon.Question 	تعرض الرسالة أيقونة الاستفهام، وهي على شكل علامة استفهام ؟ داخل دائرة للدلالة على أن هذا السؤال يحتاج إلى إجابة من المستخدم.

مثال:

الكود التالي يعرض نفس الرسالة كما في المثال السابق، لكن مع إظهار أيقونة الاستفهام عليها:

```
MessageBox.Show("هل تريد إغلاق البرنامج", "تأكيد", _
    MessageBoxButtons.YesNo, MessageBoxIcon.Question)
```

ملحوظة هامة:

المعاملات الثاني والثالث والرابع في الوسيلة Show ليست معاملات اختيارية.. إن المعاملات الاختيارية هي معاملات خاصة بدوال فيجيوال بيزيك القديمة فقط مثل الدالة MsgBox، لكنها غير مستخدمة في فئات إطار العمل Framework، وبدلا منها تستخدم طريقة تسمى تعدد صيغ الدالة Function Overloading.. في هذه الطريقة يتم تعريف أكثر من صيغة لنفس الدالة بنفس الاسم، تختلف كل منها عن الأخرى في بعض المعاملات.

انتبه جيدا لهذا الفارق الجوهري، لأنك لا تستطيع ترك موضع معامل فارغا في الدوال متعددة الصيغ كالدالة MessageBox.Show، بينما تستطيع فعل هذا مع المعاملات الاختيارية في الدالة MsgBox.. على سبيل المثال، هناك صيغة خامسة للوسيلة Show لها معامل خامس يتحكم في الزر الافتراضي.. لو أردت استخدام هذا المعامل فأنت مجبر على إرسال قيمة المعاملات الثاني والثالث والرابع، وسيحدث خطأ لو تركت موضع أي منها فارغا.. مثال:

```
MessageBox.Show("البرنامج إغلاق تريد هل", "تأكيد", _
    MessageBoxButtons.YesNo, MessageBoxIcon.Question, _
    MessageBoxDefaultButton.Button1)
```

القيمة العائدة من الوسيلة Show:

تخبرك القيمة العائدة من الوسيلة MessageBox.Show بالزر الذي ضغطه المستخدم.. وهي تعيد إحدى القيم التالية:

DialogResult.None	لم يضغط المستخدم أي زر.. مربع الرسالة ما زال معروضا.
DialogResult.Ok	المستخدم ضغط الزر OK.
DialogResult.Cancel	المستخدم ضغط زر الإلغاء Cancel.
DialogResult.Yes	المستخدم ضغط زر Yes.
DialogResult.No	المستخدم ضغط الزر No.
DialogResult.Abort	المستخدم ضغط زر الإلغاء Abort.
DialogResult.Ignore	المستخدم ضغط زر التجاهل Ignore.
DialogResult.Retry	المستخدم ضغط الزر Retry .

ما رأيك أن نعيد كتابة الحوار بينك وبين صديقك باستخدام الوسيلة MessageBox.Show؟.. هذا هو الكود:

```
If MessageBox.Show("هل أنت هاني", "لحظة من فضلك", _
    MessageBoxButtons.YesNo) = DialogResult.Yes Then
  MessageBox.Show("أهلا يا هاني")
```

```
Else
    MessageBox.Show("آسف جدا.. ظننتك شخصا أعرفه")
End If
```

ويمكنك أن تعيد كتابة هذا الكود بطريقة أخرى، وذلك بتعريف متغير من النوع "نتيجة مربع الحوار" DialogResult، وتضع فيه القيمة العائدة من الوسيلة Show، ثم تفحص قيمة هذا المتغير بجملة If.. هذا مفيد خاصة عندما تتعامل مع مربع حوار عليه ثلاثة أزرار مثل Yes و No و Cancel، لأنك تحتاج إلى استخدام أكثر من عملية مقارنة.. والمثال التالي يسأل المستخدم إن كان يريد حجز تذكرة القطار صباحا.. فإن أجاب المستخدم بنعم فسيتم الحجز صباحا، وإن أجاب بلا فسيتم الحجز مساء، وإن ضغط زر الإلغاء فسيتم إلغاء الحجز:

```
Dim Result As DialogResult
Result = MessageBox.Show("هل تريد الحجز صباحا ","موعد الحجز" , _
         MessageBoxButtons.YesNoCancel)
If Result = DialogResult.Yes Then
   MessageBox.Show("تم الحجز صباحا")
ElseIf Result = DialogResult.No Then
   MessageBox.Show("تم الحجز مساء")
Else
   MessageBox.Show("تم إلغاء الحجز")
End If
```

- تم بحمد الله -

لتحميل كتاب الأسئلة والإجابة للفصل الدراسي الأول:
http://www.kutub.info/library/book/2599

لتحميل كتاب الشرح للفصل الدراسي الثاني:
http://www.kutub.info/library/book/5804

لتحميل كتاب الأسئلة والإجابة للفصل الدراسي الثاني:
http://www.kutub.info/library/book/3393

أسألكم الدعاء لأبي رحمه الله

اللهم اغفر لأبي واجعل مثواه الجنة