

2019-2018

C# من البداية حتى الاتقان

By Eng27 – Hasan M. al-Fahl

هـام

بعض المصطلحات في هذا الكتاب هي ملكيات خاصة لمالكها والكاتب
يحترمها ولا ينسبها إليه!

إهداء

إلى كل طموح لم تُسعفه الإمكانيّات..

إلى ذاك الصديق الذي أوصلني إلى هنا بلا قصد..

إلى خالد، وتركي..

وأخيرًا إلى 2506 

جعله الله في ميزان حسناتهم جميًعاً

اٰہداء 2

إلى أرواح بعض دكاتري في الجامعة، الملحقين بحرف الدال والألف والـ"ما"، الذين حولوا الهندسة إلى مجرد مصطلحات نظرية وتعدادات وتعاريف، الذين تاجروا بمستقبلنا بضمير ميت

إلى أولئك المترفون في مكاتبهم الدّافئة، المكيّفة، المُخدّمة بكافة وسائل الراحة، الذين نهبوا وقتنا وسرقوا مخصصاتنا، وتركوا نحارب البرد والحر والظروف، لنكتب ما هو مكتوب، ونحفظ ما هو محفوظ بغض النظر عن منطقيته أو مصداقيته، ليعطونا علامات بقدر "حفظنا" للمسائل العلمية ورضاهم عنا

أهديكم هذا الكتاب لأنه لولاكم لما انتقلت إلى البرمجة هربًا من فسادكم
(لم يموتووا بعدُ، لكن إن شاء الله بأقرب فرصة)

{وَمَا أُوتِيتُمْ مِنَ الْعِلْمِ إِلَّا قَلِيلًا}

"إِنَّ اللَّهَ يُحِبُّ إِذَا عَمِلَ أَحَدًا كُمْ عَمَلًا أَنْ يُتَقْنَهُ"

رَسُولُ اللَّهِ مُحَمَّدٌ، صَلَّى اللَّهُ عَلَيْهِ وَسَلَّمَ

**"Teach me how to make a program,
instead of giving me one!"**

**"بإمكانك تغيير العالم بأسره!
ولكن يجب أولا الحصول على السورس كود
الخاص به.."**

Programming logic:



COMPUTER

Common Operating Machine
Particularly Used for Technical,
Education and Research

TECH8BC.COM

تنويه هام

قبل أن تنتقد أو تمدح هذا الكتاب أو الكاتب أو أسلوبه وأفكاره، اعلم أن هذا العمل ليس مجرد 510 صفحة فقط.. هذه الـ 510 صفحة هي عبارة عن عشرة أشهر عشرة أيام من العمل المتواصل حيناً والمترافق حيناً آخر.. هي عبارة عن ثلاثة مئة وسبعة عشر يوماً.. هي عبارة عما يقارب الـ 1400 عملية تعديل.. هي عبارة عن 70280 كلمة مقسمة على 473 صفحة..

(على فكرة الأعداد السابقة ليست تأليقاً، برنامج وورد أحصاها)

كما أنه كُتب في زمن حرب قيل إنها سبع عجاف في حين أنها قرن أعجف، وغاب عنه - عن الكتاب - الدعم كمصادر واستشارات، ودعم فني وتقني..

الآن امدح أو انتقد على راحتكم

وفي سياق متصل، لي رجاء عندك، إذا أحببت مشاركة أي كود أو فكرة أو فقرة من محتويات هذا الكتاب فاذكر المصدر ولا تنسبها لنفسك وذلك للأسباب أعلاه.

2019/2/13، حلب

بقراءتك لهذا التنويه فإنك توافق على شروط الاستخدام ✓

رقم الصفحة	المحتويات
20	مقدمة
21	نبذة عن المؤلف
22	تاريخ لغات البرمجة
25	الفكرة والبرمجة
26	الذكاء، الإدراك، الوعي والفهم
27	مجالات استخدام C#
28	لمن هذا الكتاب؟
30	ما هو أفضل، التعليم المقرؤ أم المرئي؟
31	خطة الكتاب
33	تنسيق الكتاب
35	الجزء الأول Console Application
37	الفصل صفر - أساسيات
39	المنطق البرمجي
41	أكواد أساسية - الفئة Console
42	• استخدام خصائص وطرق الفئات
43	• ايقاف البرنامج - انتظار أحد المفاتيح
43	• تغيير عنوان نافذة المشروع
43	• تغيير لون نافذة المشروع
43	• طباعة عبارة نصية
46	• رموز طباعة خاصة
48	أكواد متقدمة - الفئة MessageBox
52	الفصل الأول - المتغيرات
54	أنواع المتغيرات
55	• التصريح عن المتغيرات
56	• إسناد القيمة للمتغيرات
57	• التصريح عن الثوابت
58	• قراءة المعطيات
59	• التحويل بين أنواع المتغيرات
60	أنواع المتغيرات بعمق
60	• bool
60	• char

63	DateTime	•
63	decimal	•
63	double	•
64	float	•
64	int	•
64	long	•
64	string	•
67	كائن StringBuilder	•
69	تنسيق العبارات النصية	•
70	التحكم بالأخطاء Exception Handling	
72	• التحكم باستخدام try – catch	
74	الروابط	
77	التوابع الرياضية Math	
79	سلسل العمليات الرياضية	
80	فكرةأخيرة - المبرمج لا يعلم كل العلوم!	
82	أمثلة تطبيقية	
90	الفصل الثاني - بني التحكم	
90	بنية الشرط If	
94	If .. Else بنية الشرط	
96	بنية الشرط المتعددة	
98	بنية الشرط المختصر	
98	Switch بنية الاختيار	
105	أمثلة تطبيقية	
113	الفصل الثالث - بني التكرار	
113	حلقة For	
117	حلقة do – while	
117	حلقة while	
119	حلقة label goto	
120	التحكم بالحلقات	
120	• باستخدام continue	
120	• باستخدام break	
121	أمثلة تطبيقية	
134	الفصل الرابع - المصفوفات	
134	• التصريح عن المصفوفات	
135	• إسناد القيم لعناصر المصفوفات	
136	• التعامل مع المصفوفات	
137	• استخدام Array	
138	• الحلقات في المصفوفات	

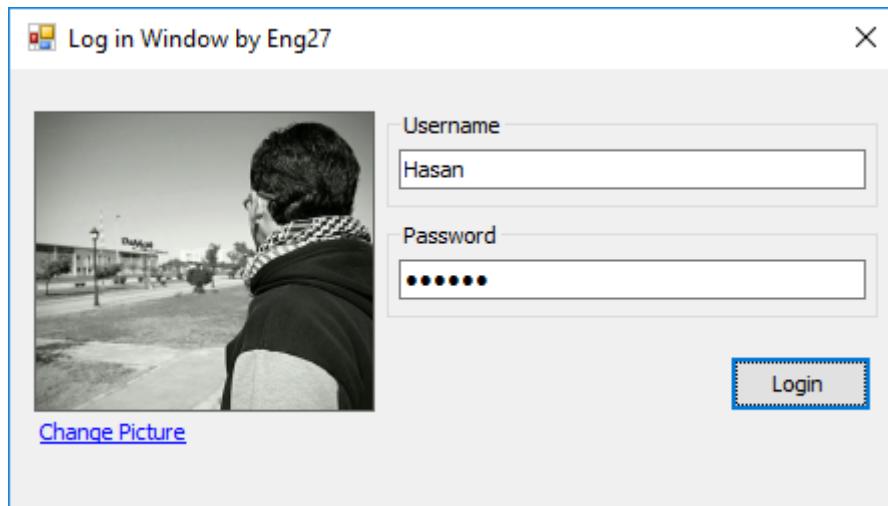
139	اللواچ Lists
139	• التصريح عن اللواچ
139	• التعامل مع اللواچ
141	القواميس Dictionaries
141	• التصريح عن القواميس
141	• إضافة العناصر للقواميس
142	• طباعة عناصر القواميس
143	• البحث عن العناصر في القواميس
143	• حذف العناصر في القواميس
143	المكدسات Stacks
144	• التصريح عن المكدسات
144	• إضافة عناصر للمكدسات
144	• طرح العناصر من المكدسات
147	أمثلة تطبيقية
161	الفصل الخامس - أنواع بيانات خاصة بك!
161	التراكيب Struct
163	• التراكيب المتفرعة Nested Structs
163	• مصفوفة من التراكيب Array of Structs
164	• طرق خاصة بالتراكيب
165	المعدادات Enum
172	الفصل السادس - الطرق، التوابع والإجراءات
176	تمرير المتغيرات
176	• التمرير بالقيمة
176	• التمرير بالمرجع
176	• التمرير بالإخراج
179	تمرير عدد غير محدد من الوسطاء كمصفوفة
180	إرجاع التوابع كمصفوفة
183	الفصل السابع - الفئات Classes
186	المجموعات Assemblies
186	مجالات الأسماء Namespaces
187	• إنشاء مجال أسماء
188	مجال الرؤية
189	• التعريف الافتراضي
189	Public •
189	Private •
189	Protected •
189	Internal •
189	static • الكلمة

190	التابع البناء Constructor
191	التابع الهدم Destructor
191	الخصائص Properties
194	الوراثة Inheritance
198	الفئات المجردة abstract
199	الفئات المغلقة sealed
199	الوظائف الوهمية virtual Methods
201	تعدد التعريف Overloading
202	تعدد الأشكال Polymorphism
203	الواجهات Interfaces
205	المفهومات Delegates
206	التقسيم المتعدد Multicasting
207	الأحداث Events
209	الوظائف المجهولة Anonymous Methods
210	العبارة لامدا Lamda expression
212	مكتبات الارتباط الحيوي dll
214	• إنشاء مشروع dll
216	• إضافة المكتبة إلى مراجع المشروع
218	• مكتبات API الجاهزة
220	الجزء الثاني Windows Form Applications
222	الفصل الثامن - أساسيات 2
222	• متصفح المشروع Solution Explorer
223	• صندوق الأدوات ToolBox
223	• الخصائص Properties
223	• قائمة الأخطاء Error List
223	• النموذج Form
223	• محرر الأكواد
225	• استيراد المراجع
226	الخصائص Properties
229	الأحداث Events
232	الفصل التاسع - مشاريع تطبيقية
233	• مشروع TextToSpeech
238	• مشروع EmailSender
244	• مشروع WindowsExplorer
253	• مشروع NotePadDataBase
259	• مشروع SpeakToComputer
268	• مشروع TabbedNotePad

280	• مشروع WebBrowser
284	• مشروع PopupMessage
288	• مشروع MP3Reader
291	• مشروع تحويل الأعداد إلى كلمات
309	الفصل العاشر - الدوس و C#
310	ما هو موجه الأوامر cmd؟
311	أوامر دوس
326	• عمليات متقدمة على الأوامر
329	• أخطاء قد تواجهك
330	الملفات الدفعية Batch Files
336	• تغليف الملفات الدفعية
337	حماية الملفات والبيانات
338	• تشفير النصوص
341	• حماية الملفات داخل صورة
353	• حماية الملفات داخل قواعد البيانات
354	• حماية الملفات بتحويلها إلى ملفات نظام مخفية
356	الفصل الحادي عشر - الرجساري
359	التعامل مع القيم والمفاتيح
359	• نظرة عامة
360	• تطبيق 1 - نسخ الملف كمسار!
362	• تطبيق 2 - نسخ المجلد كمسار
362	• تطبيق 3 - نسخ تقرير عن محتويات المجلد
365	• تطبيق 4 - نسخ محتوى الملفات النصية!
366	• تطبيق 5 - نسخ محتوى الملفات النصية 2
368	• تطبيق 6 - خصائص الملفات باختصار
370	نشر المفاتيح، تصديرها واستيرادها
373	مسح المفاتيح
373	التسجيل الصامت!
373	الرجساري من خلال C#
374	• القيم والمفاتيح برمجيّاً
375	• تطبيق 7 - التحقق من المستخدم!
382	• تطبيق 8 - فكرة، نسخة محدودة..
383	صيغة ملفات خاصة بك!
383	• تطبيق 9 - ربط برنامجك بصيغة خاصة به
397	الفصل الثاني عشر - قواعد بيانات SQL
398	إنشاء قاعدة بيانات
402	بعض أوامر قواعد البيانات
403	• عرض البيانات select

405	• عرض البيانات وفق شرط where
408	• عرض البيانات وفق ترتيب معين order by
409	• الإضافة والحذف والتعديل insert, delete, update
410	• التوابع الجاهزة
411	• تجميع البيانات group by
412	• النسخ الاحتياطي
412	إنشاء اتصال مع قاعدة البيانات
413	• الاتصال باستخدام كائن بناء الاتصال
414	القراءة من قاعدة البيانات
415	• إضافة وحذف بيانات إلى قاعدة البيانات
416	• تعديل بيانات قاعدة البيانات
418	• استخدام SqlParameter
419	عرض البيانات في جدول عرض
421	التنقل بين البيانات والعمليات عليها
424	إجراءات المخزنة
429	• SQL Injection
429	الربط بين أكثر من جدول
436	طبقة الوصول للبيانات DAL
437	• الاتصال وقطع الاتصال بقاعدة البيانات
437	• حذف البيانات
442	الفصل الثالث عشر - قواعد بيانات ACCESS
443	إنشاء قاعدة بيانات
444	إنشاء اتصال مع قاعدة البيانات
447	إضافة سجل جديد
449	حذف السجلات
450	تعديل السجلات
451	سهولة الوصول إلى البيانات
453	البحث عن السجلات
454	تجميع البيانات
456	إظهار البيانات في قائمة عرض ListView
459	إظهار البيانات في مخططات
462	حفظ الصور في قاعدة البيانات
468	النسخ الاحتياطي
469	حماية قاعدة البيانات
471	• إعادة ضبط كلمة السر
473	الفصل الرابع عشر - التقارير Reports
474	• تصميم التقرير
477	• ربط أداة عرض التقارير بالتقدير

477	• عرض تقرير عن كافة المحتويات
478	• عرض تقرير عن السجل المحدد
480	• عرض تقرير عن نتائج البحث
483	الفصل الخامس عشر - تقنيات مفيدة ☺
483	محارف غير موجودة في كيبوردك!
487	التحكم بمدخلات صناديق الإدخال
488	تأكيد الخروج من البرنامج
489	تغيير خط إحدى الأدوات
489	عمليات مفيدة على ListBox
491	الأيقونات في التطبيقات مثل الألوان في اللوحات!
493	الملحقات والمراجع
495	ملحق أ - جدول ASCII
496	ملحق ب - أكواد Alt
500	ملحق ج - الكلمات المحجوزة في C#
506	ملحق د - رموز الكائنات في الفيجوال ستوديو
507	المراجع
509	رسالة الكتاب





مقدمة

طالما استهونتني C# في بداياتي، وبدورى غضضت بصيرتي عنها كرامَةً للغти الأم الفيجوال بيسك، وضعفًا مني مادياً ومعنوياً، بالإضافة إلى أسباب أخرى تتعلق بالزمان وأسباب متعلقة بالإمكانيات. لم تسعنـي لغتي في مسائلـي ومشاكلـي البرمجـية، ولم تواكبـني مع تطورـات العـصر، ولم تشفـ غـليلـي بأبـسط أدـوات التـحكم حتـى.. لكن لم أهـاجر إـلى C# لهـذا السـبـب، لا أـحب حالـات الطـلاق ولا فـراق الـانتقام 😊.

في بداياتي كنت أخاف من أي شيء يحوي كلمة "Net". وأنحاشـاه عـلـى صفحـاتـ الانترنت وقنـواتـ الـيوـتيـوبـ وـحتـىـ فيـ كلمـاتـيـ، وهـذـاـ ماـ قـللـ اـهـتمـاميـ بـ C#. وبـمواـزاـةـ ذـلـكـ كانـ لـفـيـجوـالـ بـيـسـكـ دـوـتـ نـتـ ذاتـ النـصـيبـ منـ المـعـادـاةـ، هـيـ لـغـةـ أـخـرىـ مـخـتـلـفـةـ تـمـامـاـ عـنـ لـغـتـيـ وـلاـ تـحـمـلـ منـهاـ إـلـاـ اسمـهاـ معـ لـقـبـ عـائـلـةـ أـخـرىـ، وـلـمـ أـكـنـ أـرـىـ ذـلـكـ إـلـاـ تـبـلـيـاـ مـنـهاـ.

توغلـتـ فـيـ C#ـ شـيـئـاـ فـشـيـئـاـ، وـظـهـرـ لـيـ أـنـهـ اـبـنـةـ عـالـمـ وـنـاسـ وـمـحـترـمـةـ وـأـدـمـيـةـ (ـوـكـافـةـ صـفـاتـ الـلـطـفـ الـأـخـرىـ..)، سـاعـدـتـنـيـ خـلـفـيـتـيـ فـيـ C++ـ فـيـ تـخـطـيـ غالـبـيـةـ الـمـبـادـئـ، وـكـانـ لـلـغـتـيـ الأمـ فـضـلـ آخرـ ذـوـ مـذـاقـ مـخـتـلـفـ..ـ ماـهـيـ إـلـاـ شـهـورـ حتـىـ اـكـتـسـبـ ثـقـافـةـ كـافـيـةـ لـأـقـولـ أـنـنـيـ أـعـتـنـقـ الـ C#ـ بـجـانـبـ الـفـيـجوـالـ بـيـسـكـ. وـبـالـمـصـادـفـةـ، اـكـتـشـفـتـ أـيـضـاـ أـنـ Visual Basic.Netـ اـبـنـةـ عمـ C#.Netـ

خلال مشوارـيـ فـيـ هـذـاـ الـكـتـابـ اـقـتـبـسـتـ كـثـيرـاـ مـنـ درـوسـ خـالـدـ السـعـدـانـيـ عـلـىـ الـيـوـتيـوبـ، كـلـ الشـكـرـ لـهـ، وـجـعـلـهـ اللـهـ فـيـ مـيزـانـ حـسـنـاتـهـ، بـالـإـضـافـةـ إـلـىـ بـعـضـ الـكـتـبـ الـأـخـرىـ. وـالـشـكـرـ أـيـضـاـ لـتـرـكـيـ العـسـيرـيـ عـظـيمـ الـفـيـجوـالـ بـيـسـكـ، فـلـهـ فـضـلـ الـبـداـيـةـ، وـلـلـهـ الـحـمـدـ أـوـلـاـ وـآـخـراـ.



هذا الكتاب ليس مرجعا في C# ولا دليل استخدام ولا يمثل إلا نفسه، ويمكن اعتباره ملخص صغير للأكواود الأكثر تكرارا والأفكار والشروحات الأكثر تواجدا بين سطور أي برنامج C#.

بشكلأساسي، سوف أقوم بتطبيق الأكواود على فيجوال ستوديو 2012. ولن أشرح كيفية تنزيل أو تثبيت أو تكريك هذا البرنامج – أو البيئة البرمجية – على اعتبار أن هذا الكتاب ليس مرجعا كما اتفقنا ☺. كما أنني لن أشرح كل شاردة وواردة في C# بكل ما تحتوي من طرق وفئات.

قد يخطر ببالك: "أنت بدأت بالكتاب في بدايات 2018، لماذا تقدم أمثلة على برنامج قديم؟"، سأجيبك بأن "الذي يعلم يعلم، والذي لا يعلم يقول كف عدس!"¹، جهاز الكمبيوتر لدى عندما ابتدأت الكتاب مواصفاته: Core 2 Due، 4 Ram، 1 VideoCard، وفهمك كفاية 😊، وبعد فترة اقتنيت i7-2670، وانتقلت من SONY SP إلى GrandPimePlus لترافقني رسالة "الذاكرة ممتلئة امسح بعض التطبيقات"!! (متخيلكم أن حالي يائسة؟).

نبذة عن المؤلف

اسمي حسن الفحل، سوري من مدينة حلب وأدرس في كلية الآداب الميكانيكية.²

أحب السباحة وكرة القدم ولا أجدهما، كما أنني أحب القراءة إلا أنه لا وقت لدى (حالياً على أقل تقدير).

أنا كلاسيكي للغاية، وأوصلك الفكرة من أوسع أبوابها: خلفية سطح مكتب على Windows 7 وWindows 10 – وما بعدهما إن شاء الله – هي خلفية Windows XP الشهيرة.

لا أملك أية امتيازات أو شهادات في البرمجة، ومعلوماتي مرجعيتها فيديوهات يوتيوب مختلفة وكتب إلكترونية لأناس هواة.. وحتى تصلك الفكرة: معلومات هذا الكتاب أشبه بمعلومات طالب علم أسس علمه على مقالات ويكيبيديا. لذلك فـ "اعلم ممن تأخذ علمك"، تأكد وابحث وتحقق من كل معلومة تقرؤها حتى لا تأتي يوم وتقول "حسن قال المعلومة الغلانية واتضح أنها خاطئة".

¹ يقال أن هذه المقوله هي مثل شعبي.

² عندما تكون كلية الهندسة عبارة عن مكان لتحفيظ التعريف والتعدادات، وحفظ المسائل بأرقامها، ومحاربة العقول النابغة مما يدفعه للسفر خارجاً "لتتفع الأجنبي كما يقال"، في هذه الحالة هي كلية آداب وليس كلية هندسة..



تاريخ لغات البرمجة

لغة سي شارب C# هي من لغات البرمجة القوية، وهي إحدى بीئات الدوت نت التي أنتجتها وطورتها شركة ميكروسوفت، "وذلك من أجل الخروج من ورطة الجافا والقضية الشهيرة التي رفعتها عليها شركة SUN".

تعال لأحكي لك حكاية من تأليفي مبنية على راوٍ ثقة: كان ياما كان، في قديم الزمان، مجموعة من الناس يعملون في مجال البرمجيات وأمور أخرى مشابهة وبطريقة ما اخترعوا حاسباً بحجم غرفة كبيرة، كان قد أثار ضجة كبيرة في ذلك الوقت بسبب توفيره الوقت الكبير المطلوب أثناء الكتابة على الورق أو من أجل العمليات الحسابية الأربعية..

كان هذا الحاسوب يتعامل مع برامج مكونة من عدد كبير من الأصفار والوحدات، ثم تم اختراع ما يسمى بلغة التجميع Assembly.. لكن العمل معه كان صعباً للغاية، ويطلب حفظ الكثير من الأسطر البرمجية للقيام بأمور عادية وصغيرة..

في هذه الأثناء كانت هناك شركة تسمى IBM تصمم حواسيب آلية عليها نظام تشغيل، لكنه كان بسيطاً ولا يؤدي الكثير من الوظائف، الأمر الذي أدى إلى الحاجة إلى تطويره..

ومع موازاة هذه الأحداث، تم تطوير نظام تشغيل DOS، وهو اختصار لثلاث كلمات Disk Operating System قرص نظام التشغيل..

وبسبب صعوبة التعامل مع لغة التجميع، تم إنشاء مفهوم جديد وهو الإجراءات Procedures، وهي عبارة عن مكتبات تضم إجراءات ودوال تمثل الأسطر البرمجية الكثيرة التي قلنا إنها للقيام بأعمال عادية وصغيرة.. وهذا كان بداية ظهور لغات عرفت بالجيل الثالث للغات البرمجة مثل لغة سي والفورتن..

كانت هناك شركة اسمها Apple Macintosh، قامت بتطوير نظام يستخدم الصور للدلالة على الكائنات، مثل أيقونات المجلدات والملفات والصور والفيديو ... إلخ، ليظهر بذلك نظام يتعامل مع الواجهات بدل موجه الأوامر ..Command Prompt

بالإضافة إلى الشركة تلك، كان هناك شركة اسمها SUN MicroSystems، لا أدرى إذا ما كان اسمها اختصاراً لثلاث كلمات أم أن معناه الكلمة "ابن" أم أن معناها يوم الأحد.. المهم أنها أطلقت نظاماً مفتوحاً المصدر يسمى يونيكس، ثم أنشئ نظام اللينكس والذي يستخدم الواجهات في عمله..



بالإضافة إلى الشركاتتين السابقتين، كان هناك شركة اسمها Microsoft، كانت تعتمد على موجه الأوامر، لذلك فقد كان سوقها ضعيفاً.. فقامت هذه الشركة بإطلاق تطبيق يعتمد الواجهة الرسمية وهو Windows 1.X وذلك في 13/11/1985 والذي كان يتطلب 320KB من الذاكرة رام كحد أدنى، و720KB من مساحة القرص الصلب (يعني كتابي هذا حجمه أكبر من هذا النظام بأضعاف 😂). كان يحوي على آلة حاسبة، ساعة، محرر نصوص، لعبة Reversi، تقويم بالإضافة إلى إمكانية إضافة مهام شخصية يذكر بها الكمبيوتر، وبعض الأمور الأخرى، باختصار كان هذا النظام أحدث من نظام هاتف Nokia 3310 بقليل.

ثم تالت النسخ فجأة Windows 2.X ليكون تطويراً لسابقه، ثم جاء Windows 3.X ليتعامل مع الأيقونات بشكل أوسع وأفضل، بذاكرة رام 2MB كحد أدنى، و20MB من مساحة القرص الصلب (لاحظ الفرق في الحجم وما يتربّع عليه من إضافات، عم أحكي جد).

في هذه الأثناء ظهر مفهوم البرمجة كائنية التوجه، أو ما يعرف بـ OOP، مما أدى لظهور الجيل الرابع من لغات البرمجة مثل سي ++ وفيجوال بيسك ..

ثم أُطلق نظام Windows 95 والذي كان يعتمد على DOS، ثم Windows 97، ثم Windows 98.. وعندما تم إنشاء برمجيات Microsoft Office ..

ولجذب المبرمجين والعملاء - والذين يمثلون الزبائن بالنسبة للشركة - قامت Microsoft بإطلاق مكتبات API هي نفسها التي يعمل بها النظام، بحيث يمكن لأي مبرمج أو عميل الحصول على هذه المكتبات مجاناً.. وبدلاً من كتابة أسطر برمجية طويلة للحصول على واجهات تشابه واجهات نظام التشغيل الخاص بـ Microsoft فإنه من الممكن الآن استخدام تلك المكتبات وتوفير الوقت والجهد.. حركة ذكية صراحةً، والحركة الأذكى أن هذه البرامج لن تعمل إلا على نظام Windows :D

زادت أرباح الشركة بشكل هائل، وحتى تلحق بها شركة SUN، أطلقت لغة اسمها JAVA، وعندما ظهرت مكتبات SDK من إنتاج الشركة الأولى، وJDK من إنتاج الثانية..

مع تطور البرمجيات ظهرت الحاجة للاهتمام بقواعد البيانات، فقامت شركة اسمها Fox Pro بالاهتمام بذلك.. وحتى لا تخسر شركة Microsoft حضورها اشتريت الشركة 😊.. وبعدها مباشرةً أنشأت مكتبة خاصة لقاعدة البيانات اسماها SQL..



وباستخدام قواعد البيانات أنشأت Microsoft لغات مثل Visual C++ و Visual Fox Pro و Visual Basic.. ثم طورت لغة خاصة لمعالجة صفحات الانترنت سميت InterDev تعامل مع الملفات ذات الامتداد ..ASP.

في هذه الأثناء كانت شركة SUN تعمل على تطوير لغتها JAVA دون الحاجة إلى وضع لغات عديدة وتفرعات بلا طعمة، بحيث يمكن لمبرمج JAVA برمجة قواعد البيانات والتعامل مع الانترنت من اللغة نفسها!..

قامت شركة Microsoft بإطلاق لغة Visual J++ والغاية منه والله أعلم جذب انتباه الناس إليها بعيدا عن JAVA، لم يعجب شركة SUN الموضوع كثيرا ورفعت دعوة قضائية بكل صدر رحب على Microsoft.. وكسبت القضية، ونتيجة لذلك دفعت شركة Microsoft مiliارات الدولارات ومنعت من استخدام اللغة الأخيرة أو حتى تطويرها..

بسبب تعدد لغات البرمجة ظهرت مشاكل عديدة مثل وجود أكواد في لغة Visual Basic على امتداد نسخها وهي غير موجودة في C++ مثلا، والعكس أيضا.. في حين أن هذه المشكلة ليست موجودة لدى شركة SUN لأنها ليست طمامعة كثيرا.. مما اضطر شركة Microsoft إلى أن تقلد شركة SUN وتطلق لغة برمجة موحدة يستخدمها جميع زبائن الشركة..

بعد استشارات واتفاقات، ارتأت الشركة إلى أن تكون هذه اللغة تعمل تحت إطار اسمته إطار Net.. وهو يجمع جميع لغات البرمجة التي أطلقتها الشركة.. وقف أحد أعضاء الشركة - وكان لديه بعد نظر - وقال أنه من المحتمل خسارة الشركة لمليين المبرمجين حول العالم، إذ أن لغة Visual Basic كانت الأكثر انتشاراً، وأضاف ستكون خطتنا على النحو التالي: نضع لغة نهتم بها C#.Net وهي محور اهتمامنا، ونضع لغة تمثالية حال تجذب المبرمجين القدامى حتى لا نخسرهم، واسمها Visual Basic.Net، ليعتقدوا أنها امتداد لسلسة لغات الفيجوال بيسك من 1 وحتى 6، وهذا يتطلب أن ننشئ لغة C# بكل ما فيها ونضع رؤانا فيها (مع ترك بعض الرؤى للنسخ القادمة حتى يبقى المبرمج متطلع لأمور جديدة 😊)، ثم ننشئ لغة مطابقة تماماً للغة C#.Net بحيث تبدو أنها امتداد لـ Visual Basic. وفي الواقع، لا علاقة لـ C#.Net بـ Visual Basic.

¹ جرت العادة في الشركات العالمية والبرامج المحترمة - خصوصاً التي تصدر بشكل سنوي - ألا يتم وضع جميع الأفكار في النسخة الحالية وذلك حتى يضمنوا صدور نسخ جديدة بأفكار جديدة.

² راجع كتاب "دليل المختصر - الهجرة من الفيجوال بيسك إلى الفيجوال سي شارب والعكس". لخالد السعداني، حيث يتضمن نبذة عن اللغتين مع أهم الفروقات بينهما.



هذا من جهة، ومن جهة أخرى فإن الفيجوال ستوديو - والذي يجمع لغات دوت نت ضمن برنامج واحد - يحوي لغات كثيرة ليست فقط فيجوال بيسك وسي#, وعلى اعتبار أن الكثير من المبرمجين ينحازون للغات برمجة محددة ولا يرغبون بالتخلي عنها لتعلم لغة أخرى فإنه من المناسب جمع جميع لغات ميكروسوفت ضمن بيئة واحدة بحيث يمكن للفريق البرمجي أن يحتوي على مبرمجين للغات مختلفة. كما أن بعض اللغات تتميز عن غيرها بأمور معينة، فإذا ما تم جمع أكثر من لغة لتعمل على ذات البيئة فإن ذلك سيفتح مجالاً للخروج بمنتجات من أكثر من لغة واحدة.

الفكرة والبرمجة

يقال أن الفكرة في البرمجة هي كل شيء، كما أنها المفتاح الأساسي لتعلم البرمجة بشكلٍ صحيح. وأصحاب هذا المقال يعتقدون أن هناك احتمالية لإضاعة فترة كبيرة من عمرك هباءً وأنت تتعلم البرمجة فقط لأنك لا تملك فكرة تعمل عليها بعد تعلمك.. والفكرة برأيهم لن تتعلمها أو تبحث عنها في مكان ما. الفكرة قد تأتي صدفة أو بعد تفكير عميق. والسؤال هو هل يجب إيجاد الفكرة قبل تعلم البرمجة أم تعلم البرمجة ثم إيجاد الفكرة؟ ويرأيي المتواضع أن الفكرة موجودة دائماً، ولغة البرمجة هي أسلوب - أو وسيلة - حياة، قد تصل لغاياتك بهذه الوسيلة أو بغيرها، وقد تدعم اللغة طرق الوصول لهذه الغاية ولكنك لا تعرف استخدامها لذلك..

قبل تعلم البرمجة ستكون الفكرة إنسانية بحثة وخيالية، وعندما ستكون - أنت - عندما تتعامل مع الفكرة، مثلـي ومثلـك عندما نخطط لما سنفعله في الفضاء أو في أعماق البحار.

أثناء تعلم البرمجة ستشعر أن الفكرة التي تبنيها غبية ومعاقبة وساذجة وستستغرب لماذا اخترعوا لغات بهذا الغباء، وكيف أن البرامج المشهورة والعالمية قد وصلت لهذه الشهرة والقوة بهذه المنطق الغبي. كالكود الذي تتعرض له جميع الكائنات البشرية التي تتعلم البرمجة: كود تبديل رقمين ببعضـما، والذي تحتاج لمتغير ثالـث بلا طعة لأداء العملية.

أما بعد تعلم البرمجة فإنـك ستعيش الفكرة بشـكلـين، ستراها أمامـك وفق منظورـين، أحدهـما برمجي والآخر إنساني.. ستتخيل الطريق للوصول لـلـفـكـرة بـرمـجـيا وفي نفسـ الوقت ستـتخـيل النـتـائـج مـحسـوـسة كما لو أنها أمامـك في لـحظـتها وكـيف سـيـتـعـامل معـها ويـفـهـمـها المستـخـدمـون العـادـيون. ستـشـعـر أنـك عـازـف بيـانـو محـترـف ومنـ حـولـك لا يـفـقـهـون شيئاً بـآلات العـزـف إـلا أنـهم يـحـبـون نـغمـاتـ الـبيـانـو..



الذكاء، الإدراك، الوعي والفهم

قال لي أحد دكاتري في الجامعة: الكمبيوتر عبدٌ غبي!!

ويقال أيضاً إن الإنسان أذكي من الكمبيوتر مهما بلغ الأخير من قوىًّا ودقةً وسرعةً، لكن بالمقابل فإن مسألةً يحلها الكمبيوتر في لحظات قد لا تتجاوز ثانيةً أو اثنتين يستغرق الإنسان حلها ساعات وربما أيام، هذا يعني أن الكمبيوتر أذكي.. ومع هذا فلا تسمح الكلمات بالتحايل عليك، فلعبة الكلمات سهلة للغاية، الفكرة أن الإنسان يمكنه اختراع كمبيوتر بينما الكمبيوتر لا يمكنه اختراع إنسان!

فيلسوف أمريكي اسمه جون سيرل John Searle طرح تجربة فلسفية، مفادها أنه لدينا غرفة فيها إنسان ومعه أكواام من الأوراق مكتوب عليها عبارات باللغة الصينية، هذا الإنسان لا يعلم شيئاً عن الصينية فهي عبارة عن رموز عديمة المعنى بالنسبة له. يوجد في الغرفة أيضاً مجموعة من الجداول تمثل إرشادات مكتوبة باللغة التي يفهمها هذا الإنسان.

يوجد خارج الغرفة شخص آخر يرسل لمن في الغرفة مجموعة بيانات مكتوبة باللغة الصينية - والتي لا يفهمها صديقنا داخل الغرفة - على أوراق ليستلم غيرها من الغرفة كجواب لها. لتبسيط الموضوع يقوم الشخص خارج الغرفة بإرسال ورقة عليها سؤال، وصديقنا داخل الغرفة عليه إعطاءه جواباً لهذا السؤال على ورقة أخرى.. لكن لا تنسى فصديقنا لا يفهم الصينية!

تقول تجربة سيرل أن صديقنا سيستلم الورقة من خارج الغرفة، ويعالجها باستخدام جداول الإرشادات والتي يبحث من خلالها على رموز مشابهة للرموز الموجودة في البيانات المستلمة، أي أن الإرشادات تقولُ لو رأيت الشكلَ الفلانيَّ في السؤال فاذهبْ إلى المنطقةِ الفلانيةِ في الغرفة وأحضر ورقةً أخرى بعنوانِ كذا ورقمِ كذا، هذه الورقة هي جواب السؤال المطلوب. وفعلاً دونَ أن يكذبَ خبراً يقوم صديقنا بالبحث عن الرمز المكتوب باللغة الصينية والتي لا يفهمها، وعندما يجد جواباً يحضر الورقة التي تحتوي على رمز آخر باللغة الصينية والتي أيضاً لا يفهمها!! أي أنه لم يفهم السؤال ولا الجواب حتى..

الشخص خارج الغرفة عندما يستلم ورقة الإجابة، ويشعر أنها منطقية وهي الحل المطلوب سيعتقدُ أنَّ من في الغرفة قد فهمَ السؤال والإجابة معاً! لكن في الواقع الموضوع عبارة عن مطابقة رموز ليس إلا..



صديقنا داخل الغرفة هو الكمبيوتر، ليس لديه صفة الفهم أو الإدراك أو الوعي، وإنما يقوم بتنفيذ أوامر محددة بناءً على معطيات وبيانات محددة..

وحتى من أجل أعقد وأدق أنظمة الذكاء الاصطناعي فلا يمكن إعطاء الكمبيوتر ميزة الوعي والإدراك والفهم، الأمر الذي يمتلكه الإنسان بشكل فطريّ!

لكن بالمقابل، فلا سفة آخرون قالوا إنه طالما هناك نتائج صحيحة لمعطيات ما فإنه من الممكن إعطاء صفة الفهم للكمبيوتر، ولو حتى فهم جزئي. يعني أننا لا يمكن أن نقول إن الإنسان داخل الغرفة يفهم الصينية، لكن النظام ككل - الإنسان + جدول الإرشادات - يفهمه! وهذا منطقى نوعاً ما، فالكمبيوتر لا يعلم ماذا تمثل أو تعنى القيمة 1 عند إسنادها للمتغير المسمى a، ولا 0 عند إسنادها للمتغير b، ولا 9 عند إسنادها للمتغير c، لكن بالمقابل هو يعطيك ورقة مكتوب عليها العبارة: " $3 = -3 \times x_1$ ", هو فقط يعلم أن a و b و c هي متغيرات عدديّة، وأن هناك ما يربط المتغيرات مع بعضها - مجموعة من المعادلات وبنى الشرط والتكرار - ولكنه لا يعلم ماذا تمثل في الواقع!

مجالات استخدام C#

قد يسأل سائل ما هي المجالات التي يمكن استخدام C# خاللها !?
في الواقع جميع لغات البرمجة يمكنها القيام بجميع الأمور التي تحتاجها في حياتك وذلك يعتمد على خبرتك ومهاراتك في لغة البرمجة التي ستخدمها وانتباها للمسألة أو القضية التي تواجهها. طبعاً يتوقف الأمر على قدرة اللغة أساساً على الخوض في المجال الذي تحتاج تحليله وبرمجته، بالتأكيد لا يمكنك استخدام موجّه الأوامر لبرمجة قواعد البيانات على سبيل المثال، لذلك يجب الأخذ بعين الاعتبار قابلية لغتك البرمجية على تحليل ومناقشة الموضوع الذي تتعامل معه.

يمكن من خلال C# تطوير البرامج والتطبيقات التي تعمل على أنظمة تشغيل ويندوز، وهي تدعم البرمجة كائنية التوجه OOP، وتعتمد على مكتبات إطار الدوت نت، وهي ميزة تُسهل كتابة البرامج المعقدة دون الاعتماد على مصادر خارجية.

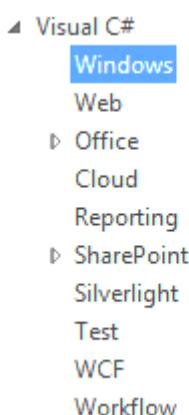
الجدير بالذكر أن جميع اللغات التي تعمل ضمن بيئه الدوت نت بإمكانها الوصول لذات النتيجة، واختيار لغة البرمجة لهذا الغرض مبني على ميولك ورغباتك وراحتك تجاه هذه اللغة، بحيث أن المهارة تكمن في إمكاناتك للوصول لهذه النتيجة على أكثر من لغة تعمل وفق بيئه الدوت نت..



يمكن تلخيص مجالات استخدام C# كما يلي:

- صناعة وبرمجة تطبيقات نظام ويندوز.
- برمجة تطبيقات الانترنت، وذلك عبر منصة ASP.NET.
- برمجة الـ Graphics والوسائط المتعددة.
- برمجة الألعاب وذلك باستخدام بيئات مشغلات الألعاب.
- برمجة تطبيقات تتعامل مع قواعد البيانات باستخدام مكتبة ADO.NET.
- برمجة تطبيقات إدارة المحتوى.
- برمجة تطبيقات الاندرويد وذلك باستخدام برنامج Xamarin.
- تشكيلاً أوسع في الداخل 😊.

عموماً، يمكنك القيام بما يلي في C#, حيث لكل فرع من هذه الفروع¹:



لمن هذا الكتاب؟

إذا كنت قد اطلعت على أكثر من كتاب للبرمجة فالتأكيد قد لاحظت أن الكثير منها قد ابتدأ بهذه الفقرة، ذلك لأن للبرمجة مستويات، وأن المبرمجين متباينون في المستوى. شخصياً أصنف المبرمجين وفق أربعة مستويات: المبتدئين والمتوسطين والمتقنين والمحترفين. وعلى هذا الأساس اخترت اسم هذا الكتاب.

مشكلة المحتوى العربي أنه يفتقر لمراجع وكتب تتحدث عن البرمجة للمحترفين، وأن جل الكتب موجهة للمبتدئين، اللهم إلا قلة قليلة من الكتب لمبرمجين يُعتبرون من نجوم البرمجة العربية. ومن الممكن أن يكون

¹ بعض محتويات هذه الفقرة مأخوذة من الانترنت بتصرف.



هذا الأمر - أن أغلب الكتب موجهة للمبتدئين - منطقياً على اعتبار أن المحترفين أو المتوسطين بإمكانهم البحث والاطلاع في المنتديات والمواقع وقنوات اليوتيوب بحثاً عن ضالتهم، أما المبتدئين فيحتاجون من يمسك بأيديهم وينقلهم من اللا شيء إلا المستوى المبتدئ أو من المبتدئ إلا المتوسط. ولكن ما أحawل قوله هو ماذا لو كانت هناك مراجع عربية تهتم بالاحتراف؟؟ بالتأكيد كان حالنا سيكون أفضل وبرامجنا ستكون منتشرة بشكل أوسع وكان سيُحسب لنا حساب في عالم التكنولوجيا والمعلومات وسيكون لنا وجود وحضور، إلا أن ما يحدث أن المحترفين في البرمجة لدينا إما مُحتكرون ويكتفون بما جنوا لأنفسهم، أو أن لا يزال لديهم في الكتابة أو التأليف ويرغبون بالتعليم الشخصي - وجهاً لوجه - أو بقنوات اليوتيوب أو المنتديات.. وأكرر، اللهم إلا بعض نجوم البرمجة العربية فعطاؤهم معروف وفضلهم مشهور !!

هذا الكتاب موجه لكل من المستويات التالية:

للمبتدئين فهو يعرّفهم على مفاهيم أساسية في البرمجة وينتقل معهم خطوة خطوة حتى الوصول للنتيجة المطلوبة، فيعطيهم أهم ما يحتاجونه للبدء باستخدام C# بدءاً بالمتغيرات وانتهاءً بالتتابع.

للمتوسطين فهو يسرد لهم المفاهيم التي يعرفونها ويشرح لهم مبادئ البرمجة كائنية التوجه OOP، كما يزودهم بمشاريع تطبيقية وأمثلة تطبيقية مختلفة وفي مجالات واسعة لتكون ذخراً لهم لإنشاء تطبيقات خاصة بهم. كما ينقلهم لبرمجة قواعد البيانات وأهم ما يحتاجونه لذلك.

للمتقنيين فهو يمثل سرداً سريعاً لمعلوماتهم وترتيباً لأفكارهم ومهاراتهم، كما يمكن اعتباره مرجعاً لتطبيقاتهم (حتى ولو أني لا أحبذ ذلك لغياب الدعم العلمي على محتوى الكتاب).

ومع الأسف فهو لا يشكل شيئاً للمحترفين لأن الشريحة المستهدفة من الكتاب هي الشريحة المتعلمة لا المعلمة، كما أني لما أصل لهذه المرحلة بعد 😊.

وبالنسبة لأسلوب طرح الكتاب وفقراته فمع الأسف فهي متسلسلة متتابعة يتعلقها آخرها بأولها بشكل كبير - خصوصاً في الفصول الأولى - مما قد يشتت بعض القارئين، وأقول مع الأسف لأنني من الكتاب كانت شرح كلّ صغيرة وكبيرة بالتفصيل الممل إلا أن ذلك أدى إلى التكرار في كثير من المواضيع، لذلك فعليك التحلي بالصبر وقراءة الفصول التي تشترك مرة واثنتين وثلاثة قبل أن تدعوا علي.



حاولت - قدر الإمكان - أن أضفي على الكتاب أسلوب الكتابة المشوب ببعض الكلمات العامية لشحذ همتك، كما أن هذا أسلوبني حتى في دفاتري في الجامعة. وبالتأكيد لا يكتمل الأسلوب دون وجود سمايلات تعبر عن حالي.

ما هو أفضل، التعليم المقرؤ أم المرئي؟

تتميز البرمجة عن غيرها من العلوم بأن متعلمتها ليس بالضرورة أن يكون اختصاصياً، أي أنه ليس بالضرورة أن تكون طالباً في كلية المعلوماتية أو الحواسيب أو أي كلية تتعامل مع البرمجة لتعلم البرمجة. لذلك فهناك الكثير من يتعلم البرمجة دون مرجعية (معلم يرجع إليه)، أي بالتعليم الذاتي، وهذا ما يضع المتعلم أمام خيارين: أيهما أفضل، التعليم المقرؤ أم المرئي؟؟

التعليم المقرؤ هو تعليم من الكتب، والتي تعتمد بشكل كبير على درجة ثقافة القارئ، وهي أكاديمية غالباً. أما المرئي منه فهو التعليم بمشاهدة فيديوهات من اليوتيوب بشكل أساسي، غالباً هو تطبيقي بامتياز لذلك فهو الخيار الأمثل لأغلب المتعلمين.

شخصياً فقد تعلمت البرمجة من اليوتيوب واعتمدت على كثير من الكتب كمراجعة، ويرأسي فالفيديوهات مناسبة لاكتساب المهارات والكتب مناسبة لاكتساب المعلومات. بمعنى أنه عليك - خصوصاً إن كنت جديداً على البرمجة - أن تشاهد دروساً على اليوتيوب للبرمجة ثم تطبقها على كمبيوترك ثم تقرأ عنها من الكتب لكتساب المعلومات النظرية وترتب أفكارك وتكتسب المزيد والمزيد من الأمثلة والتطبيقات. ويمكن تطبيق هذا الأسلوب على جميع لغات البرمجة.

ولا بأس بمشاهدة فيديوهات لأناس وقراءة كتب لأناس آخرين، مع أنه يفضل في بداياتك أن تكتفي بمصدر واحد تتعلم منه.. وعندما تتجاوز مرحلة "المبرمج متوسط المستوى"، لا مشكلة في تعدد المصادر فيبلغوك لهذه المرحلة بإمكانك تمييز المحتوى الجيد من الرديء، وبإمكانك ربط الأفكار مع بعضها سواءً أشرحت أو قرأت من مصدر واحد أو من عدة مصادر. ففي هذا المرحلة - وما بعدها - أنت لديك الأفكار والمهارات وتحتاج فقط لترتيبها واكتساب المزيد وضمها لما تملك.



خطة الكتاب

يناقش الكتاب أساسيات لغة C# وبعضاً من المواقف المتقدمة فيها، ويعرك على كل ماتحتاجه لبناء تطبيقات صغيرة متكاملة، وذلك بتعريفك على مبادئ وأساسيات البرمجة ثم ينقلك للبرمجة كائنية التوجه والنوافذ وقواعد البيانات ومواقف أخرى مختلفة موزعة على ستة عشر فصلاً، مقسمة على جزأين، الأول يتحدث عن بيئة Console والثاني يتحدث عن النوافظ والمواقف المتقدمة مثل قواعد البيانات وغيرها.

الفصل صفر يتحدث عن بعض الأكواد التي ستحتاجها على امتداد الجزء الأول من الكتاب، ويهيئك لتدخل إلى البرمجة وفهم مفاهيمها.

الفصل الأول يتحدث عن المتغيرات وما هي وما بينها والفرق بين نوع آخر، وكيفية استخدامها والروابط بينها، والتحكم بالأخطاء. كما يشرح كيفية التعامل مع المسائل الرياضية في البرمجة. ويناقش بعض الأمثلة التطبيقية.

الفصل الثاني يتحدث عن بنى التحكم وأهميتها في البرمجة، بدءاً من بنية الشرط if وانتهاءً ببنية الاختيار switch. ويعرض بعض الأمثلة التطبيقية.

الفصل الثالث يتحدث عن الحلقات والفرق بين نوع آخر منها وأهميتها، وكيفية استخدامها والتحكم بها. ويشرح بعض الأمثلة التطبيقية.

الفصل الرابع يتحدث عن طرق تجميع البيانات وذلك عبر المصفوفات وغيرها. ويعرض بعض الأمثلة المفيدة.

الفصل الخامس يناقش كيفية إنشاء أنواع بيانات خاصة بك وهو مفيد جداً خصوصاً عند انتهاءك من الجزء الأول كاملاً وبالتحديد عند إتقانك للفصل السابع، وهو أعنى فصول الكتاب.

الفصل السادس يعرفك على التوابع والإجراءات وهي مفاهيم مهمة للغاية إذ إنها حولت مجراً تاريخ البرمجة عند اختراعها.

الفصل السابع يناقش مبادئ البرمجة كائنية التوجه بدءاً من الفئات ومروراً بالكائنات وانتهاءً بمكتبات الارتباط الحيوي DLL.

الفصل الثامن يتحدث عن أساسيات النوافظ، ويشرح لك بعض الأدوات وخصائصها والأحداث التي تمر عليها لتصبح جاهزاً لإنشاء وتصميم تطبيقات مفيدة - صغيرة كبداية - والفصل التاسع غير تطبيق عملي لما سترؤه هنا.



الفصل التاسع يشرح لك عدة مشاريع تطبيقية تعطيك فكرة عن كيفية إنشاء تطبيقات سطح المكتب، وهي تطبيقات عملية على الفصل الثامن. أساساً كانت خطة الكتاب القديمة تقضي بأنّ الهدف من الكتاب هو شرح مشاريع شاملة عن مواضيع مختلفة في C# ولكن هذا كان صعباً فتم اختصار المشاريع لتعطيك أفكار محددة وتم خفض عدد المشاريع (لو أن الكتاب مؤلف ليناقش فقط المشاريع دون التطرق للمفاهيم كان من الممكن القيام بذلك).

الفصلان العاشر والحادي عشر يعطيانك فكرة عن كيفية التحكم بتطبيقك من خلال الرجسستري والدوس، وذلك عبر تطبيقات عديدة قد تكون مفيدة.

الفصلان الثاني عشر والثالث عشر يدخلانك إلى قواعد البيانات وأهم أوامرها وذلك من خلال أمثلة بسيطة ت Howell لإنشاء تطبيقات صغيرة مفيدة.

الفصل الرابع عشر يعطيك لمحة عن كيفية إنشاء التقارير.

الفصل الخامس عشر ينهي الكتاب بتقنيات من المحتمل أن تحتاجها في برمجك، وهي من الأمور التي يبحث عنها في البرمجة على الانترنت بكثرة.

تنسيق الكتاب

شخصياً أفضل استخدام خط Tahoma في كتبتي تيمّنا بأستاذي، حجم الخط المستخدم 14، وفي العناوين العريضة 20، وفي الأكواود 12 إلا ما ندر.

الصيغ والقواعد المستخدمة توضع داخل جداول من الشكل:

--

أما الملاحظات فقد وضعت في جدول من الشكل:

ملاحظة



وأخيراً، الأكواود وضعت في جدول من الشكل:

--



التواصل مع المؤلف

للحصول على أي دعم أو استشارة، أو لاقتراح تعديلات على الكتاب أو الأعمال القادمة – إن قدر الله لي ذلك – أو اقتراح أعمال بإمكانك التواصل معي على بريد الصفحة الخاصة بالبرمجة :Eng27 – Programming

- فيسبوك <https://www.facebook.com/Eng27Programming/>
- تلغرام عبر المعرف @Eng27Channel

كما يمكنك الاشتراك بالبوت Eng27 على التلغرام الذي أضع فيه بعضًا من منتجاتي، وبعض الكتب التي من الممكن أن تكون مراجعة جيدة يمكن الاستفادة منها، وذلك عبر المعرف @Eng27Bot، أو عبر قناتي على اليوتيوب Hasan M. al-Fahl والتي سأقوم بشرح كتابي عليها كل فقرة على حدة، كما سأنشر شروحاتي التي لم أتمكن من وضعها في هذا الكتاب.

أو يمكنك التواصل معي عبر الواتساب أو التلغرام أو المكالمات الهاتفية عبر الرقم +963954796627.





الجزء الأول

Console Application





الفصل صفر - أساسيات

لن يقدم لك هذا الكتاب تعاريفاً نظرية وتعدادات كلاسيكيةً عن لغة C#, قد تتفق معي وقد لا تتفق، لكن من وجهة نظري فهذا الكتاب ليس مرجعاً للغة، وإنما عمل متواضع يختصر بعض المفاهيم والمواضيع التي تتكرر مع مبتدئي ومتسطوي المستوى في C#. كما أن الكتاب يفتقر بشكل كبير للمعلومات النظرية وهو مليء بالأخطاء النحوية والعلمية لذلك لا تشدّ يدك فيه كثيراً.¹.

أحببت ابتداء فصول الكتاب بالفصل صفر تيمناً بالمبدأ البرمجي، حيث يبدأ العد من الصفر بشكل افتراضي. إذا كان هذا الكتاب هو بداياتك مع البرمجة فأتمنى أن يعطيك انطباعاً عنها ويوصلك لمبدئها.

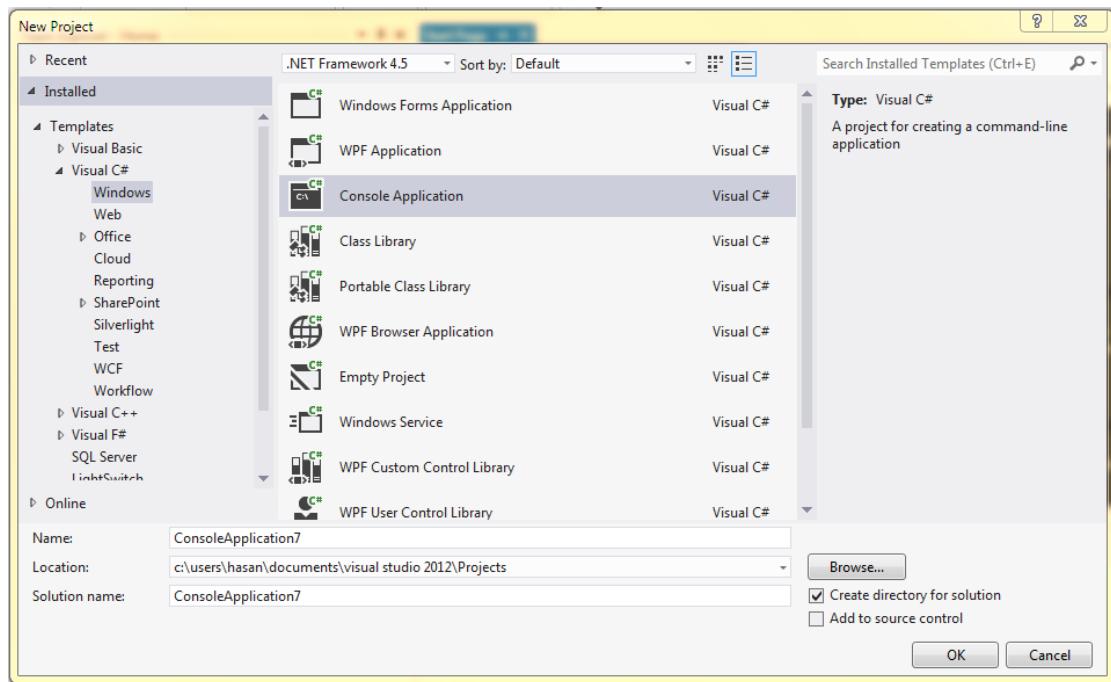
جرت العادة في كتب البرمجة، أن يتم إضافة شرح موجز عن الفيوجوال ستوديو، لن أدخل بكل هذه التفاصيل، سأهتم بالعنب بعض الشيء وسأتحدث عن الناطور بشيء من الإيجاز لاحقاً إن احتجنا لذلك.

أنشئ مشروعًا جديداً من قائمة النماذج Templates حدد مشاريع لغة C#، اختر نوع النوافذ Windows واختر Console Application، سُمّ مشروعك واضغط OK للحصول على مشروع جديد.

ملاحظة

- افتراضياً، اسم المشروع هو ConsoleApplication1، ويزاد الرقم تلقائياً مع كل مشروع جديد.

¹ مصطلح سوري بمعنى أنه لا يعتمد عليه.



مباشرةً بعد أن يُنشأ المشروع، سيتم إعطاءك الأسطر البرمجية التالية كهدية، لا تفوت فرصة فحصها، فكل سطر برمجي له وظيفة محددة لن تعمل دونه.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication7
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}

```

كما تلاحظ تم تقسيم هذه الأسطر إلى فئتين، الأولى مكتبات من خلالها تتمكن من تنفيذ الأكواد، ويمكن استخدام مكتبة ما من خلال البادئة using والتي تعني "باستخدام"، أي باستخدام المكتبة الفلانية، قم بتنفيذ أكواد برمجي. الفئة الثانية، هي البرنامج بحد ذاته.

بالعودة للبادئة using، هي كلمة محفوظة وظيفتها إخبار C# أنك ستستخدم إحدى طرق وفئات فئة رئيسية ما أسميناها مكتبات دون تكرار اسم المكتبة الأمل سواءً أضفت المكتبة إلى مراجع البرنامج أم أنها من



المراجع الرئيسية فيه، إذا كنت جديداً على C# لاتشغل بالك كثيراً بها ومع مرور الأكوا德 ستؤلف كتاباً بها إن أردت، ماعليك إلا أن تفهم المبدأ الآن فقط.

تعتبر C# من اللغات كائنية التوجه OOP، وفي مثل هذه اللغات يعتبر كل شيء عبارة عن كائن Object، سنتعامل لاحقاً مع مربعات رسائل، قوائم منسدلة، وأدوات مختلفة، كل ما سبق عبارة عن كائنات بما فيها برنامجك هو الآخر كائن. ولكل كائن خصائص تميزه عن الكائنات الأخرى وأفعال يقوم بها وأحداث تمر بها. في برماج # يتمثل الكائن بفئة Class – أو قد يترجم للغة العربية بكلمة خلية – ويسبق باسم الفئة، محتويات كل فئة تحصر ما بين قوسين معقوفين { }.

إن كل فئة تحوي أقساماً وفئاتٍ داخلها، ولاستخدام الفئات الفرعية يُكتب اسم الفئة، نقطة، الفئة الفرعية، نقطة، .. وهكذا حتى الوصول للفئة أو الطريقة أو الخاصية المطلوبة.

سترى لاحقاً أن البرنامج مؤلف من فئات أو صفوف Classes، وهذه الفئات تحتوي على أفعال وخصائص كما أشرت، هذه الأفعال تسمى طرقاً Methods، وأي برنامج يجب أن يحوي الطريقة Main، حيث أن المترجم يقوم بالانتقال مباشرةً إلى هذه الطريقة، ثم ينتقل للطرق والفئات الأخرى التي تنشأها أنت إذا كان هناك حاجة لذلك.

ملاحظة

- المترجم هو جزء من النظام، بالاشتراك مع البيئة البرمجية التي تتعامل معها، يقوم بترجمة أكوادك من اللغة البرمجية التي تفهمها، إلى لغة الآلة التي يفهمها الحاسوب، أي مجموعة أرقام مكونة من 0 و 1 مكررة بشكل كبير جداً.

المنطق البرمجي

في بداية هذا الكتاب تم عرض صورة معنونة بـ Programming logic، ومفادها أن البرمجة تعتمد على البيانات المخزنة، المعطيات، الحقائق والثوابت وكل ما هو معرف.. على عكس المنطق البشري الذي يتميز بالمرونة والتعلم الذاتي واكتشاف المجهول..

قبل أن ندخل في عالم البرمجة سنطرح مثلاً يمكن لأي كائن حي مشتق من فئة البشر أن يفهمه..



لنفرض أنك أردت من أحدthem أمرما، فلتطلب منه فإنك تأمره، والأمر يأتي على أكثر من نوع.. فيكون دعاءً إذا كان من الأدنى للأعلى، كما يكون أمراً إذا كان من الأعلى للأدنى.. وأنت عندما تتعامل مع الكمبيوتر - كمبرمج أو مستثمر - فإنك تأمره، لذلك فأوامرك مطاعة (إلى حدٍ ما، وبشروط)..

لنعد لفرضيتنا، إذا أردت أمر أحدthem أن يعطيك كوب ماءٍ مثلاً، فإنك ستستخدم الأمر التالي:

;() أحضر_الماء;

لاحظ أن إحضار الماء هو عبارة عن فعل، والأفعال هي الطرق Methods في C#، لذلك فعليك عند إصدار الأمر وضع قوسين ()، يمكنك إصدار تفاصيل أخرى مع الأمر وذلك بوضعها بين هذين القوسين.

وعندما ترغب بشرب الماء فإنك ستستخدم الأمر التالي:

;() شرب.الكوب;

الكوب هو كائن مشتق من فئة `أواني` منزلية، لشرب الماء منه فإنك ستقوم بإحدى الأفعال الممكنة مع الأكوب وهي شرب الماء، وبشكل منطقي لا يمكنك مثلًا القيام بأفعال مثل الدردشة مع الكوب 😊، أي أن هناك أفعال بشكل منطقي وفطري وظيفي بإمكانك القيام بها مع الكوب، والتي ستختصر على بالك بمجرد التفكير بالكوب أو مشاهدته، وفي البرمجة لمشاهدة كائن أو التفكير به عليك كتابة اسمه ثم نقطة فيظهر كل ما يميزه عن الكائنات الأخرى من صفات وأفعال وظروف تمر عليه.

وأما إذا أردت تعبئة الكوب بالعصير مثلاً فإنك ستستخدم الأمر:

;() عصير).تعبئة.الكوب;

ومعناها أنك ترغب بتعبئة الكوب بالعصير، حيث أن ما بين قوسين هي تفاصيل تحدد مميزات الفعل الذي يتم القيام به كما اتفقنا.

لاحظ الكلمات العربية البسيطة هذه، بتغييرها إلى الإنكليزية فقط فإنك قد كُونت أكواداً برمجيةً ذات معنى!!

لكن لحظة، كيف يفهم الكمبيوتر علينا؟؟ أو بمعنى آخر هذه الكلمات المكتوبة باللغة الإنكليزية، كيف يعرف الكمبيوتر أنها أكواود؟؟ لأنه بالتأكيد



ليست أية كلمة كوداً برمجياً.. حتماً هناك كلمات محددة تكتب بطريقة محددة هي فقط أكواود اللغة. في الواقع فالجواب على هذا السؤال يحتاج شرح مفهومين أساسيين في أي لغة، هما Compiler و IDE.

الـ IDE أو بيئة التطوير المتكاملة هي أقرب ما تكون إلى محرر نصوص يستقبل منك الكلمات الإنكليزية ويعالجها ليتأكد من أنها أكواود ويحلل نوعها فقد تكون متغيرات أو كلمات محجوزة أو طرق أو عبارات نصية، أو قد لا تمثل شيئاً أصلاً..

كما أن بعض بيئات التطوير تعطيك ميزات أكثر من الأخرى، فقد تجد في بعض اللغات أنه عند كتابة أسطر برمجية فإنها ستتحاذى بشكل معين.. في حين أن هذا الأمر غير موجود في لغات أخرى.

والجدير بالذكر أن بيئة التطوير تعطيك إشارة على مكان وجود خطأ ضمن أسطرك البرمجية، أي أنها ستكون يدك اليمنى طوال حياتك..

ومن هنا فإن بيئة التطوير المتكاملة وظيفتها استقبال الأكواود منك، وبعدها يأتي دور الـ Compiler أو المترجم والذي شرحناه في ملاحظة سابقة.

وبشكل عام فإن أي لغة تتكون من مصطلحات - هي كلمات محجوزة - وقواعد كتابة، بالربط بينها تحصل على لغة البرمجة.. كما هو الحال في اللغات الإنسانية. وبالمناسبة: بإمكانك إنشاء لغة برمجة 😊. كما أن هناك لغات برمجة عربية (فاجأتك صحيح؟¹).

أكواود أساسية – الفئة Console

عند تعاملك مع المشاريع من النوع Console Application فالفئة هي أكثر كائن ستعامل معه، رأينا منذ سطور كيفية استخدام طرق الفئات، وذلك عبر استخدام نقاط تفصل بين الفئات والطرق الفرعية فيها، وذلك حتى الوصول للطريقة أو الخاصية المطلوبة.

إن الفكرة من وجود الفئات هو تسهيل البرمجة إلى حد كبير، بحيث تُعطى أنت الأكواود الجاهزة على البارد المستريح²، وتقوم ببرامجه على أكمل وجه بنفس الفكرة التي أعطت ميكروسوفت مكتبات API الخاصة بها لزيانها، حيث إن هذه الأكواود الجاهزة لا تراها وإنما مكتوبة ضمن ملفات معينة.

¹ لم أفاحنك؟؟ غريب 😊.

² البارد المستريح: بلا عناء



استخدام خصائص وطرق الفئات

سأشرح فكرتين نظريتين مرت معنا في الفقرات السابقة:

الفكرة الأولى: لاستخدام أي كائن جاهز في نظام التشغيل دون أن تبرمجه من الصفر عليك استخدام طرق وخصائص الكائن System، وهو كائن يمثل نظام التشغيل، جعلته ميكروسوفت متداولاً ليتمكن الجميع من إنشاء كائنات مشابهة لكتائنهما. لولاه – ولولا كائنات أخرى – كان عليك برمجة كل شيء من الصفر، حتى النوافذ والأزرار وصناديق النصوص وكل ما تأخذة بنقرة زر.

الصيغة التالية توضح كيفية استخدام خصائص وطرق الطرق والكائنات:

System.Console؛
خاصية أو طريقة;

الصيغة السابقة معناها ككود أنك تطلب من C# أن يتعامل مع System ضمن الفئة Console ليأتي بالخاصية أو الطريقة المعينة ل تقوم بأمر ما.

الفكرة الثانية: إنه من الصعب وغير المجدى أن نكرر الكلمات في الأكواد خصوصاً إذا كانت هذه الكلمات ستكون في الكود بأكمله، حيث أن لغات البرمجة عالية المستوى كان أحد أهدافها تقليل عدد الكلمات التي تكتبها لإنجاز نتيجة ما. بالتأكيد برنامجك لن يحتوي على كود واحد، وسيكون من مئات الأكواد كحد وسطي، هل أنت متخيلاً كتابة كلمة System مئات المرات؟

الصيغة التالية توضع كيفية استخدام خصائص وطرق الكائنات أو الفئات وذلك بكتابة اسم الكائن أو الفئة الأم مرة واحدة:

في بداية البرنامج //
using System;
. .
/Main
Console؛
خاصية أو طريقة;

وهذا ما سنناقشه في الفصل السابع.

هناك الكثير من الطرق والخصائص التي من الممكن أن تستفيد منها في حياتك البرمجية وسأسرد لك أكثرها تكراراً:



إيقاف البرنامج - انتظار أحد المفاتيح

ابتدأت أكوادي بهذا الكود الذي ستستخدمه في جميع برامجك، عند عدم وجوده فإن البرنامج سوف يُترجم ثم يغلق فوراً، جرب استخدام أي كود برمجي تالي لهذا الكود ضمن هذا الفصل - أو الجزء بشكل عام - دون استخدام هذا الكود وسترى أن نافذة مشروعك ستغلق فوراً.

```
Console.ReadKey();
```

ملاحظة

- جميع الأكواد في لغات الـ C يجب أن تنتهي بفاصلة منقوطة، عدا بعض البني التي سترتها لاحقاً.
- **أغلب مشاكلك وأخطاءك البرمجية هي فاصلة منقوطة** 😊.

تغيير عنوان نافذة المشروع

سيقوم الكود التالي بتغيير عنوان مشروعك إلى العبارة داخل التنصيص.

```
Console.Title = "Eng27's Project!";
```

ملاحظة

- افتراضياً عنوان المشروع هو مسار المشروع.
- الطرق - وهي توابع وإجراءات - تلحق بقوسین ()، أما الخصائص فلا تلحق. هل لاحظت أن هذه الملاحظة تكررت بشكل كبير 😊.

تغيير لون نافذة المشروع

يمكنك تغيير لون الخط أو النافذة كما يحلو لك:

```
Console.BackgroundColor = ConsoleColor.Black; // لون النافذة
Console.ForegroundColor = ConsoleColor.Green; // لون الخط
Console.Clear();
```

طباعة عبارة نصية

لا أعني بالطباعة عملية نسخ محتوى نص أو صورة أو ملف ما وطباعتها على ورقة أو مجموعة من الأوراق، في البرمجة تستخدم كلمة طباعة للدلالة على إخراج عبارة نصية إلى المستخدم على الشاشة.



```
Console.WriteLine("Eng27");
```

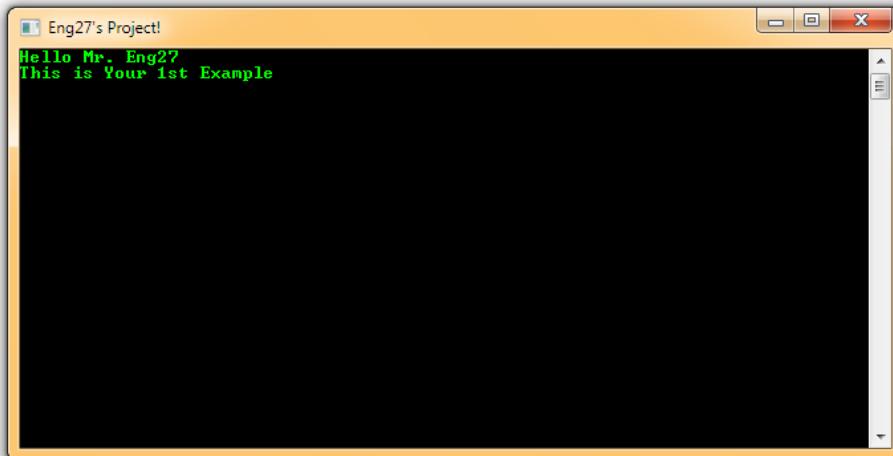
يقوم الكود السابق بطباعة عبارة نصية، مع إبقاء مؤشر الكتابة على السطر نفسه الذي يتم الكتابة عليه، أما للانتقال للسطر التالي (طف سطر) فالكود التالي سوف يشفى غليلك.

```
Console.WriteLine("Eng27");
```

لنقم بمثال شامل لما سبق:

```
Console.Title = "Eng27's Project!";  
Console.BackgroundColor = ConsoleColor.Black;  
Console.ForegroundColor = ConsoleColor.Green;  
Console.Clear();  
Console.Write("Hello Mr.");  
Console.WriteLine(" Eng27");  
Console.WriteLine("This is Your 1st Example");  
Console.ReadKey();
```

ضع هذا الكود – الأسطر البرمجية – داخل التابع Main، ثم ترجم المشروع لتحصل على ما يلي:

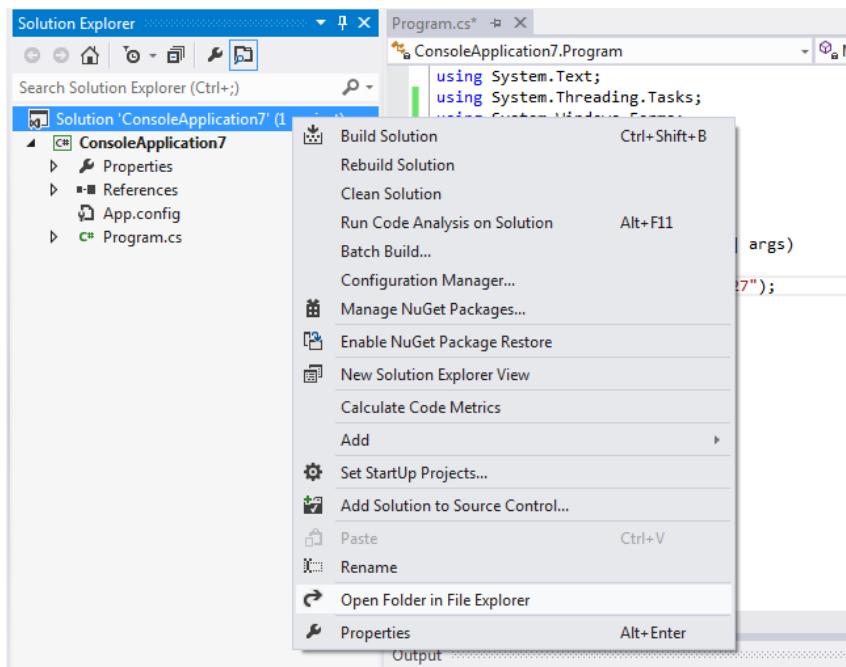


ملاحظة

- يمكن ترجمة المشروع عن طريق الضغط على زر F5 أو الزر Start ضمن القائمة Debug.
- هذا أول مشروع لك، وبهذا أكون قد قضيت على فرصة أن يكون أول برنامج لك هو Hello World.



يمكن الحصول على نسخة تنفيذية exe من البرنامج وذلك بالانتقال إلى موقع المشروع ثم مجلد bin ثم Debug، ستجد بعض الملفات ومنها الملف التنفيذي المطلوب، شاب شعر رأسي حتى اكتشفت الطريقة.



كما يمكن طباعة أكثر من عبارة نصية، كما يلي:

```
string name = "Hasan";
Console.WriteLine("Hello Mr: " + name);
```

لكن انتبه، معامل الجمع (إشارة +) قد تستخدم للجمع وقد تستخدم للربط بين النصوص، وهذا ماستفهمه مع مرور الأكواد بإذن الله.

وفي العبارات الكثيرة، هناك صيغة مختصرة، لاحظ المثال:

```
string car = "Kia", model = "Spectra";
int cost = 9000;
Console.WriteLine("Car Name: {0}\nModel: {1}\ncost: {2}",
    car,
    model,
    cost);
Console.ReadKey();
```

ونتيجة تنفيذ الكود ستكون مماثلة للكود الذي تربط فيه العبارات النصية بإشارة +، والذي سيكون أطول بكثير وأكثر عرضة للأخطاء.



رموز طباعة خاصة - تنسيق الطباعة

تُستخدم هذه الأحرف داخل العبارات النصية وتسمى عادةً مفاتيح الهروب كترجمة حرفية لها، أي أنها تحاط بإشاراتي تنسيص، ويمكن أن تُضمن داخل عبارة نصية موجودة مسبقاً، ويعرف عليها المترجم بوجود المحرف \ قبلها.

الوظيفة	الحرف
إعطاء صوت، تنبيه	\a
مسح حرف للخلف	\b
سطر جديد	\n
مسافة أفقية	\t
مسافة شاقولية	\v
طباعة '	'
طباعة "	"
طباعة \	\\
طباعة محرف بمعرفة كود Unicode خاص به	\u
كتابة مسار ملف أو مجلد، عوضاً عن استخدام الرمز \.	@"Path"

قد تحتاج في بعض الأحيان لطباعة الرمز ' \ ' أو ' \u '، وعند استخدامك لهذه الرموز منفردة فإن العبارة النصية ستكون مختلفة، لذلك تم اختيار الترکيبة الموجودة في الجدول السابق آخر سطرين، كل الشكر لمخترعيها ^_^.

المثال التالي يبين لك الفرق بين استخدام الرموز السابقة من عدمها:

```
Console.WriteLine ("Menu:");
Console.WriteLine ("      A) Option1");
Console.WriteLine ("      B) Option2");
Console.WriteLine ("      C) Option3");
```

ال코드 السابق يمكن أن يختصر كما يلي:

```
string A = "Menu:\n\t A) Option1\n\t B) Option2\n\t C) Option3";
Console.WriteLine(A);
```



فاضل إعلاني

بالنسبة لتطبيقات Console، المبرمجة لأغراض تعليمية أو تجريبية، يمكن استخدام تطبيق Dcoder، حيث يدعم الكثير من اللغات البرمجية التي تتعامل مع بيئة Console. الصورة التالية توضح مثلاً بسيطاً تم تنفيذه باستخدام هذا التطبيق.

The screenshot shows the Dcoder app interface on an Android device. At the top, there's a status bar with signal strength, battery level (57%), and time (20:18). Below it is the app's header with the title 'Dcoder' and icons for file operations. The main area has a dropdown menu set to 'C#' and a 'Input' toggle switch. Below this is a code editor window containing the following C# code:

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text.RegularExpressions;
5
6 /*Please dont change namespace, Dcoder
7 and class must not be public*/
8
9 //Compiler version 4.0, .NET Framework 4.5
10
11
12 namespace Dcoder
13 {
14     public class Program
15     {
16         public static void Main(string[] args)
17         {
18             //Your code goes here
19             Console.WriteLine("Hello");
20             Console.Write("Mr ");
21             Console.Write("Eng27");
22         }
23     }

```

Below the code editor is an 'Output' section with three colored dots (red, green, yellow) followed by the text 'Output'. Underneath is a scrollable text area showing the program's output: 'Hello', 'Mr ', and 'Eng27'. At the bottom of the app screen are standard Android navigation buttons.

ملاحظة

- تطبيق Dcoder لا يعمل إلا إذا كنت متصلًا بالإنترنت. كما أنه يوفر لك إمكانية حفظ مشاريعك وتطبيقاتك.

هام

- يدعم تطبيق Dcoder اللغات البرمجية والبيانات التالية: C , C++ , C# , Java , Python , VB.Net , Pascal , Html , Css والكثير الكثير من اللغات الأخرى التي لم تسمع عنها إطلاقاً.
- للأمانة أنا لا أقوم بدعاية لهذا التطبيق وإنما أدل القارئ على أساليب تراعي ظروفه وتساعده على العمل والبرمجة والتعلم 😊.

كما بإمكانك استخدام موقع DotNetFiddle للبرمجة من خلال هاتفك.
بالعودة للـ Console والـ C# وأمور البرمجة، ماذا يعني مصطلح؟



بيئة Console لا تحوي نوافذ، ولا أدوات أو قوائم، لا وجود لصناديق النصوص أو أزرار الأوامر، فقط يوجد أكواد – نسميها أسطر برمجية – تُنفَّذ سطراً سطراً مع إمكانيات كثيرة لقيادة وإدارة البرنامج أثناء التنفيذ.

أكواد متقدمة – الفئة MessageBox

من الأمور الهامة وشائعة الوجود في أيّ برنامج هو صناديق الرسائل، فهي تلعب دور الوسيط بين برنامجك والمستخدم (أو بالأحرى بينك وبين المستخدم).

من الأمور المرجعية التي تساعدك على إنشاء صناديق رسائل هي إضافة مكتبة System.Windows.Forms إلى مشروعك، وذلك من خلال إضافة مكتبة System.Windows.Forms إلى Solution Explorer.

ملاحظة

- في بعض لغات البرمجة، يكون صندوق الرسائل من الأمور القياسية المعطاة مع النموذج، وفي حين أن عليك إضافته في C#.

من أكواد المشروع، قل له أنك تود استخدام محتويات المكتبة التي أضفناها، وذلك بال코드 التالي:

```
using System.Windows.Forms;
```

ثم استخدم الأمر التالي في برنامجك لإظهار مربع الرسالة:

```
string name;  
name = "Eng27";  
MessageBox.Show ("Welcome Mr. " + name);
```





لاحظ أن مربع الرسالة قديم (من أيام Windows XP وماقبل، انتبه للزر)، كما أنه لا يحوي عنوان أو أي رموز أو أزرار إضافية. وهذا هو أبسط شكل من أشكال مربعات الرسائل..

اعتقد أنه بات لديك تصور واضح عن كيفية ومتى نستخدم `using`، وما الغرض منها، الآن لنتناقش عن تنسيق مربع الرسائل، الإضافات عليه، ومتغيراته.

لعرض صندوق رسائل، طلبنا من C# أن يستدعي الطريقة `Show`، لاحقاً ستفهم ما يعني هذا وستكون على دراية جيدة به، بحيث تتمكن من إنشاء دوال وصفوف – فئات – لها أفعال عديدة ممثلة بطرق `Methods` تستدعيها عن الحاجة. الفصل السادس يوضح ذلك.

إن وسطاء الطريقة `Show` كثيرة، أهمها النص المعروض، والوسطاء هذه هي من ستحدد تنسيق صندوق الرسائل كما سنرى.

```
string name;
name = "Eng27";
MessageBox.Show ("Welcome Mr. " + name, // المحتوى
    "MessageBox", // عنوان صندوق الرسائل
    MessageBoxButtons.YesNo, // الأزرار
    MessageBoxIcon.Asterisk, // رمز الرسالة
    MessageBoxDefaultButton.Button1, // الزر الافتراضي
    MessageBoxOptions.ServiceNotification); // خيارات الصندوق
Console.ReadKey();
```

ملاحظة

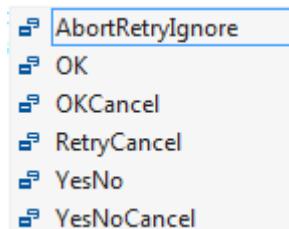
- يتم وضع الوسطاء الكثيرة للتتابع والإجراءات – الطرق – عادةً في أسطر منفصلة حتى تسهل قراءة الكود!
- بإمكانك ذلك بوضع فاصلة بعد كل وسيط ثم الانتقال للسطر التالي عبر الضغط على زر `Enter`.

عند تنفيذ الكود السابق نحصل على مربع الرسالة التالي:

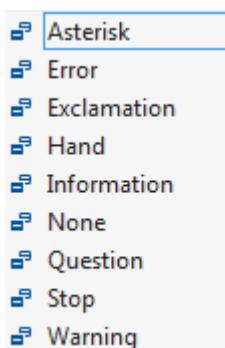


لاحظ أنه بات للرسالة عنوان، ورمز يعبر عن مزاج البرنامج وحالته، بالإضافة إلى أزرار عديدة، مع إمكانية تحديد الزر الافتراضي، بالإضافة إلى أنه لا يمكن الضغط على زر الخروج.. عد للصندوق القياسي وقارن بينهما..

- محتوى الصندوق يوضع في بداية الوسائط وهو قيمة نصية.
- عنوان الصندوق يلي اسمه، وهو قيمة نصية أيضاً.
- أزرار الرسالة، وتأخذ إحدى الحالات:



- رمز الرسالة، ويعبر عن مزاج برنامجك، وله الحالات:



- الزر الافتراضي، وهو الأول أو الثاني أو الثالث.
- خيارات الصندوق، وفيها يمكن ضبط محاذة محتوى الصندوق أو الصندوق بأكمله..



كما يمكنك التعامل مع ماسيدخله المستخدم في صندوق الرسائل بالصيغة التالية (سيلزملك في الجزء الثاني من الكتاب، ضع علامة مرجعية على هذه الصفحة):

```
DialogResult result = MessageBox.Show(title,  
                                         message,  
                                         MessageBoxButtons.YesNo,  
                                         MessageBoxIcon.Exclamation);  
  
if (result == DialogResult.Yes)  
{  
    //C#'s code  
}
```





الفصل الأول – المتغيرات

عند تعاملك مع بيانات تأخذ مجالاً من القيم يجب أن تتعامل مع متغيرات، هذه المتغيرات تمثل بيانات ومعلومات تطبيقك، حيث إن الأخير يستجيب وينفذ أوامر برمجية محددة بناءً على هذه المتغيرات.

تلعب المتغيرات دوراً حيوياً في تطبيقاتك، فهي تخزن المعلومات، وتعمل خلال كامل فترة تنفيذ تطبيقك، وعند إيقاف التطبيق أو الخروج من الفئة أو الطريقة أو الحلقة التي تتبع لها فإنها تفني!

إن وجود المتغيرات وتنوعها وإحاطتها ببرامحك تعطيه قوة وسيطرة ودقة في العمل، يزداد الذكاء الصناعي لبرامحك مع ازدياد متغيراتك والعلاقات فيما بينها وبين التحكم في هذا البرنامج (والتي ستناقش في فصول لاحقة).

وبالمناسبة أيضا فالمتغيرات تمثل ذاكرة برامحك، والأكواد التي تسيّر هذه المتغيرات هي عقل برامحك، لذلك فهو علاقتك بها :

من أبسط الأمثلة وأكثرها شيوعا هو التبديل بين قيم المتغيرات، لديك متغيران x و y ، ولكل قيمة، ولغاية ما أردت أن يأخذ المتغير x قيمة y ، وبالعكس.. الكود التالي يوضح ذلك:

```
int x, y;  
x = 5; y = 6;  
int temp; // متغير مؤقت  
temp = x; // temp = 5  
x = y; // x = 6 (not 5)
```



```
y = temp; // y = 5 (not 6)
Console.ReadKey();
```

كما أن هناك إمكانية للحصول على ذات النتيجة بمتغيرات أقل، لكن الطريقة هذه تحتاج تشغيل مخ:

```
int x, y;
x = 5; y = 6;
x = x + y; // x = 5 + 6 = 11
y = x - y; // y = 11 - 6 = 5
x = x - y; // x = 11 - 5 = 6
Console.ReadKey();
```

أنواع المتغيرات

اتفقنا في مقدمة هذا الفصل على أنه لتخزين بيانات برنامجك سواءً تلك التي يدخلها المستخدم أو التي يحصل عليها البرنامج من تلقاء نفسه يجب أن تخزن في ما يسمى بالمتغيرات.

وبشكل منطقي فالمتغيرات يجب أن تتبع للبيانات التي ستتخزن داخلها. والبيانات من الممكن أن تكون نصية مثلاً - مجموعة من المحارف والرموز - أو رقمية صحيحة أو رقمية عشرية، كما توجد بيانات تمثل التاريخ والوقت، وأخرى منطقية تحمل قيمتين 1 أو 0، كما يمكنك تكوين نوع بيانات خاص بك وهذا ما سنراه في الفصل الخامس إن شاء الله.

الجدول التالي يبين لك أنواع البيانات والتي سنستخدم بعضها في هذا الكتاب إن شاء الله:

قيمتها	نوع المتغير
قيمة منطقية <code>false</code> أو <code>true</code>	<code>bool</code>
عدد صحيح من 0 حتى 255	<code>byte</code>
حرف ¹	<code>char</code>
فئة خاصة بالتاريخ والوقت	<code>DateTime</code>
قيمة عشرية	<code>decimal</code>
قيمة عشرية	<code>double</code>
قيم ثابتة خاصة بك كمبرمج	<code>enum</code>
قيمة عشرية	<code>float</code>

¹ المحرف هو رمز أو حرف.



قيمة صحيحة	int
قيمة صحيحة	long
قيمة صحيحة	short
نوع بيانات خاص بك كمبرمج	struct
قيمة نصية	string
قيمة منوعة (ترك اختيار نوع البيانات للكمبيوتر على ذوقه)	var

ملاحظة

- العدد الصحيح هو العدد الذي يمكن كتابته دون استخدام كسors أو فواصل عشرية.
- العدد الطبيعي هو عدد صحيح موجب.
- العدد الحقيقي هو عدد يمكن كتابته مع أو دون استخدام كسors أو فواصل عشرية.

المتغيرات تمثل أماكن في ذاكرة الكمبيوتر والتي يحتاجها الأخير لكي تعمل البرامج عليه بشكل كامل، لذلك فحاول أن تكون متغيراتك اقتصادية قدر الإمكان حتى لا تسبب ببطء للكمبيوترات التي ستستخدم برنامحك، لكن لا تنسى أن المتغيرات التي اخترتها يجب أن تكفي للبيانات التي ستخزنها!

التصريح عن المتغيرات

للتصريح عن المتغيرات اكتب نوع المتغير يليه اسمه.

هناك قواعد عديدة تحدد اسم المتغير أهمها ألا يكون كلمة محجوزة¹ وألا يبدأ برقم، وألا يحتوي على فراغات أو بعض الرموز مثل نقطة أو فاصلة منقوطة، وألا يحتوي إلا على أحرف لاتينية، كما أن C# حساسة لحالة الأحرف. هذا بالإضافة إلى قواعد أخرى كثيرة، والجدير بالذكر هنا أن جميع لغات البرمجة لها ذات المبدأ بالنسبة لأسماء المتغيرات، مع فوارق بسيطة. في الحقيقة الكثير من البنى والقواعد والمبادئ مشتركة في جميع لغات البرمجة مع فوارق بسيطة، إلا أن الجوهر ذاته.

ملاحظة

- في C#, عند وجود إشارة تحت اسم المتغير باللون الأحمر فهذا يعني أنه قد يكون اسم المتغير خاطئ، أما عند وحودها باللون الأخضر فهذا يعني أن هذا المتغير لم يستخدم بعد.

¹ راجع ملحقات الكتاب، الملحق ج - الكلمات المحجوزة في C#



- في الواقع، الإشارة باللون الأحمر تعني أن السطر البرمجي أو التعليمية فيها خطأ، قد يكون فاصلة منقوطة مفقودة!

في C# لا يمكنك استخدام المتغيرات مالم تقم بالتصريح عنها، على عكس لغات أخرى كثيرة.

إن التصريح عن المتغيرات يكافي استنساخ فئة تمثل نوع المتغير، سنتناقش هذه الفكرة عندما نتطرق لأنواع المتغيرات بعمق.

إسناد القيم للمتغيرات

يمكن إسناد القيم للمتغيرات فور التصريح عنها، ويمكن لاحقاً عند الحاجة. لاحظ أنه يجب تطابق نوع القيمة المسندة مع نوع المتغير، بمعنى أن تسد قيم صحيحة لمتغيرات صحيحة، وقيم خاطئة لمتغيرات خاطئة (عم أمزح 😂).

لاحظ المثال التالي:

```
int a = 5;
double b = 2.7;
char c;
String d;
c = "Eng27"; // خطأ 1
D = "Eng27"; // خطأ 2
d = "Eng27";
c = 'N';
a = b = 5; // ملاحظة
```

- خطأ 1: لا يمكن إسناد قيمة نصية إلى متغير حرفـي، بمعنى أن المتغير c يجب أن يحوي أحرف فقط على اعتبار أنه من النوع char، ويمكن إسناد القيم له بوضعها مابين إشارتي ' '.
- خطأ 2: لا يوجد متغير اسمه D، هناك متغير اسمه d، حاول التفريق بينهما.
- ملاحظة: يمكن إسناد أكثر من متغير لقيمة واحدة شرط أن تقبل جميع المتغيرات هذه القيمة.

ملاحظة

- توفر جميع لغات البرمجة إمكانية ترك تعليقات يُستفاد منها أثناء البرمجة فقط، ولا يقرأها المترجم (هل تذكر المترجم؟ راجع الصفحات الأولى من هذا الكتاب). تستخدم التعليقات عادة لعنونة



- الأكواد أو المتغيرات ولتعريفها ولترك ملاحظات عن أكواد معينة، ويمكن أن يكون التعليق على أكثر من سطر واحد.
- حاول الإكثار من التعليقات في برامحك، فعندما تعتاد عليها وتصبح برامحك مكونة من مئات أوآلاف الأكواد عندها قد تكتشف المشاكل من خلالها وتتطور برامحك بشكل أفضل.

يمكن إسناد قيمة لمتغير من نوع `DateTime` كما يلي:

```
DateTime MyDate;
MyDate = DateTime.Now();
```

كما يمكن الاستفادة من قيمة معينة من الوقت أو التاريخ وذلك بإسناد قيمتها إلى متغير من النوع `int`.

```
int MyDate = DateTime.Now.Hour();
```

يقوم الكود السابق بإسناد قيمة الساعة الحالية للوقت إلى المتغير `MyDate`، ويمكن الحصول على كل من الثواني الحالية أو الدقائق أو الساعات أو الشهر الحالي أو السنة الحالية.. والمزيد في الداخل.

التصريح عن الثوابت

على غرار المتغيرات، يمكن لتطبيقك أن يحتاج إلى قيم ثابتة على طول فترة تنفيذه، يمكن التصريح عنها بإضافة الكلمة الممحوزة `const` إلى ما قبل نوع المتغير أثناء التصريح، مع إسناد قيمة. لاحظ المثال:

```
const double Pi = 3.14;
```

الثوابت لا يمكن أن يُعدل عليها أو يسند إليها قيم!

مثال آخر:

```
if (f == 32)
{
    C#'s Code
}
```



الكود السابق سيء برمجياً، ماذا تعني 32؟! بالنسبة لك كمنشئ لهذا الشرط فإنك تعلم معنى القيمة الثابتة 32، وتعلم ماذا سيحدث لو أن f كانت لها القيمة 32، لكن أنا مثلاً عندما أقرأ كود برنامجك لن أفهمه، لذلك فهو سيء برمجياً. فالكود الجيد هو الكود المكتوب بأكسل الطرق بأسهل العبارات . ماذا لو أردت تغيير القيمة في المستقبل وكان هذا الشرط موجوداً بأكثر من مكان في برنامجك، ماذا لو نسيت تعديل أحدها؟؟ سيختل البرنامج بالتأكيد. (لا تشغل بالك بما يعنيه الشرط وكيف يُكتب الآن، افهم الفكرة من الثوابت فقط).

لجعل الكود يبدو بشكل أفضل استخدم الثوابت:

```
const int FREEZING = 32;
if (f == FREEZING)
{
    C#'s Code
}
```

ألم يصبح الكود أفضل برأيك؟؟ بإمكانك تغيير قيمة الثابت أثناء البرمجة في أي وقت شئت، وبالتالي فإن أي كود يحوي هذا الثابت سيعتمد عليه وعلى قيمته التي برمجتها عليه. ثم إن عبارة `FREEZING = f` أوضح من تلك المتمثلة بـ `f = 32`، فالآن أفهم أن المتغير f وظيفته تخزين قيمة درجة الحرارة أو مقارنتها (إلا إذا كان ميولي أدبياً).

قراءة المعطيات

بعد تصريحك عن المتغيرات سيبدأ برنامجك سلسلةً من المجريات الدراماتيكية التي قد تؤدي إلى احتياج برنامجك لقيم خارجية يعطيها المستخدم للبرنامج، فمثلاً بإمكانك برمجة تطبيق يقوم بحساب مسائل بسيطة أو معقدة، بناءً على قيم يعطيها له المستخدم، هذه القيم هي المعطيات الأولية للمسألة (طلاب الجامعات، ركزوا على هذه الفكرة "هذه القيم هي المعطيات الأولية للمسألة"، كروها عشر مرات).

قراءة المعطيات باختصار هي قراءة القيم المدخلة من لوحة المفاتيح.

```
string a;
Console.Write ("Type Your Name Please: ");
a = Console.ReadLine( );
Console.Write ("Welcome Mr " + a);
Console.ReadKey( );
```



أما لقراءة معطيات رقمية، فاستخدم الكود التالي:

```
Console.Write ("Type Your Age Please: ");
int age = int.Parse(Console.ReadLine( ));
// أو
// int age = Convert.ToInt32 (Console.ReadLine( ));
Console.Write ("Your Age is " + age);
Console.ReadKey( );
```

كما لاحظت، فإنه من غير الممكن قراءة معطيات رقمية من خلال الطريقة `ReadLine`، فهي تعيد قيمة نصية.. لكن بإمكانك تحويل العبارات النصية التي يقرأها هذا الأمر إلى نوع آخر وذلك باستخدام الفئة `Convert`.

```
string s = Console.ReadLine();
int i = Convert.ToInt32(s); // convert to int
long l = Convert.ToInt64(s); // convert to long
float f = Convert.ToSingle(s); // convert to float
double d = Convert.ToDouble(s); // convert to double
decimal c = Convert.ToDecimal(s); // convert to decimal
```

كما يمكن استخدام الطريقة `Parse`:

```
Long A = Int64.Parse(s) // convert to long
Float A = Single.Parse(s) // convert to float
Decimal A = Decimal.Parse(s) // convert to decimal
double A = double.Parse(s) // convert to double
Int A = int.Parse(s) // convert to int
```

ملاحظة

- إذا كنت جديدا على البرمجة، لابد أنه قد تكررت معك كلمات مثل إجراءات أو طرق، لا تقلق ستتعرف عليها مع مرور الأكواد.

التحويل بين أنواع المتغيرات

كما لاحظت - عزيزي - فالبرمجة حساسة لنوع المتغيرات، لذلك فإذا كانت لديك متغيرات من نوع معين وأردت أجراء عمليات من نوع آخر عليها فعليك تحويل البيانات الواردة منها أو إليها لتتم العملية المطلوب.

طيب السؤال هو كيف أعرف متى يجب أن أحول من نوع إلى آخر؟؟؟
بشكل أساسي يجب عليك معرفة ما يجب إدخاله للطرق وما سينتج عنها.



ما معنى هذا الكلام؟! عد إلى مثال استقبال قيم من المستخدم باستخدام الطريقة ReadLine، إن ما ستحصل عليه من هذه الطريقة هو قيمة نصية حصرًا، لذلك فعليك تحويلها لنوع المتغير المراد إسناد القيمة الناتجة إليه إذا لم يكن نوعه نصيّاً.

يمكن التحويل من أي نوع لأي نوع آخر عن طريق الفئة Convert مستخدماً الطريقة المطلوب التحويل إليها، فلتتحويل لنوع double استخدام الدالة `Convert.ToDouble` التابع للفئة Convert أي `Convert.ToDouble`.

أما للت转换 من قيمة نصية إلى أي نوع آخر فيمكن عن طريق الفئة Convert أو عن طريقة الدالة Parse، فإذا أردت تحويل عبارة نصية إلى قيمة من نوع double استخدام `double.Parse`.

وللتحويل من قيمة من أي نوع إلى قيمة نصية يمكن استخدام Convert ويمكن استخدام الطريقة `ToString`، فمثلاً لتحويل قيمة المتغير الرقمي الصحيح إلى قيمة نصية استخدام `.x.ToString`.

أنواع المتغيرات بعمق

تتميز المتغيرات عن بعدها بحجم البيانات التي يامكانها تخزينه، والحجم الذي ستستهلكه من الذاكرة، ودقتها في حالة أرقام عشرية. وإليك أكثر المتغيرات شيوعاً بين أسطر أي برنامج:

bool

تلعب المتغيرات المنطقية دوراً هاماً في برامجك، فهي تمثل الحالتين المتصادتين: الشمس والقمر، الليل والنهار، الأسود والأبيض، المليء والفارغ، الكامل والناقص، الناجح والرااسب، إلخ ... حيث يتمثل كل هذا بقيمتيين true للقيمة الإيجابية و false للقيمة السلبية.

تستخدم المتغيرات المنطقية بشكل واسع مع الشروط، فلو أردت بعض الأمثلة عنها ستجدها في الفصل الثاني.

char

هذا المتغير يخزن المحارف، ولكل محرف في كمبيوترك كود خاص به يسمى كود ASCII وآخر Unicode، بإمكانك مراجعة الملحق أ - جدول ASCII والملحق ب - أكواد Alt في آخر الكتاب لتتعرف على الموضوع أكثر.



إذا انتقلت لهناك وألقيت نظرة على الأحرف المتاحة وشيفراتها، فبالتأكيد قد لاحظت أن لكل حرف رقم يمثله بالنظام العشري وآخر بالنظام السنتعشري. فمثلاً الحرف N له الشيفرة 78 بالنظام العشري و4e بالنمط السنتعشري، جرب الكود التالي:

```
Console.WriteLine('N');
Console.WriteLine("\u004e"); //Unicode
Console.WriteLine((char)78); //Ascii
Console.ReadKey();
```

تأخذ المحارف - المتغيرات من نوع char - قيمًا تمثل بحروف أو رموز محاطة بإشارة تنسيص واحدة، بينما المتغيرات من نوع string فتحاط بإشارة تنسيص ثنائية. لاحظ أن جميع القيم المسندة للمتغير الموضح بالمثال التالي تتالف من محرف واحد فقط:

```
char letter;
letter = 'N';
letter = '1';
letter = '!';
```

بشكل عام فأنواع المتغيرات - مثل string و int و char وغيرها - ماهي إلا فئات جاهزة تمت برمجتها وبرمجة الخدمات التي يمكنك الحصول عليها عند استخدامك إياها، وأنت تستخدمها ببرودة أعصاب ..

عندما نتحدث عن الفئات - وكما سنرى في الفصل السابع - فإن هناك أسطر برمجية كثيرة أنت كمبرمج لا تعلم عنها شيئاً، لأن هناك مبرمجاً آخر قام بإنشاءها وأعطاك منها النتيجة فقط.. وكما سنرى أيضاً فإن الفئات عادةً ما يضاف إليها طرقاً لإعطاء نتائج محددة، أي أنه ستكتب اسم الكائن الذي استنسخت الفئة إليه ثم نقطة ثم التواعي والإجراءات والتي سمح للكمبرمج الذي اخترعها باستخدامها.

وعندما تصل لمستوىً جيد في البرمجة وتعود إلى هذه الصفحات ستقول لي أنا عند تعريف المتغيرات لا نقوم باستنساخ فئات، فكيف تقول أن هذه المتغيرات هي كائنات؟؟ وسأجيبك أن عملية تعريف المتغيرات ماهي إلا صيغة مختصرة لاستنساخ الفئات، أي أن السطر الأول هو اختصار لما يليه:

```
char letter;
// char letter = new char();
```



طبعا في حال كنت جديدا على البرمجة وقرأت السطور الحالية وارتبتكت فليست مشكلة، فهذه الأفكار غير موجهة لك (أقصد الفئات والكائنات والاستنساخ وغيرها). وعلى سيرة الطرق الجاهزة التي بإمكانك استخدامها، المتغيرات الحرفية تعطيك الإمكانيات التالية:

- لمعرفة ما إذا كانت قيمة المتغير هي حرف كبير أم صغير:

```
char letter = 'N';
Console.WriteLine(char.IsUpper(letter));
```

- لمعرفة ما إذا كانت قيمة المتغير هي رقم:

```
char letter = 'N';
Console.WriteLine(char.IsDigit(letter));
```

- لمعرفة ما إذا كانت قيمة المتغير هي حرف:

```
char letter = 'N';
Console.WriteLine(char.IsLetter(letter));
```

- لمعرفة ما إذا كانت قيمة المتغير هي أداة ترقيم:

```
char letter = 'N';
Console.WriteLine(char.IsPunctuation(letter));
```

- لإرجاع قيمة المتغير في حالة حرف كبيرة:

```
char letter = 'N';
Console.WriteLine(char.ToUpper(letter));
```

- لإرجاع قيمة المتغير في حالة حرف صغيرة:

```
char letter = 'N';
Console.WriteLine(char.ToLower(letter));
```

وحتى لا ترتبك وتتشتت فالامثلة السابقة هي فقط "أمثلة"، أي أن الغاية منها إيصال فكرة محددة، لتجربتها أضف () Console.ReadKey() إلى نهاية الكود المطلوب تجربته.



DateTime

هذا ليس نوع متغير وإنما فئة، والفئة هي مجموعة من الطرق والمتغيرات مجمّعة مع بعضها لأداء وظائف معينة، وستراها لاحقاً إن شاء الله. إلا أنها ستناقشه هنا حتى تصبح لديك فكرة عن كيفية التعامل مع قيم للتاريخ أو الوقت.

يمكن إنشاء كائن من نوع التاريخ والوقت كما يلي:

```
DateTime date = new DateTime(1996, 5, 3);
```

كما يمكن التعامل مع الوقت عبر كائنات من الفئة TimeSpan (لا تقلق إذا لم تفهم بعض المصطلحات، تأمل كيفية كتابة الأكواد فقط):

```
TimeSpan time = new TimeSpan(10, 27, 0);
```

كما يمكن إجراء عمليات الطرح والإضافة عن طريق الطرق Add و Subtract :

```
TimeSpan time1 = new TimeSpan(10, 27, 0);
TimeSpan time2 = new TimeSpan(1, 0, 0);
TimeSpan time3 = new TimeSpan();
time3 = time1.Subtract(time2);
```

decimal

هذا المتغير يخزن بيانات عشرية ذات دقة عالية تصل حتى 29 رقم بعد الفاصلة، و المجال قيمها تصل إلى 28 خانة (المليون 7 خانات، مابالك بـ 28 خانة?!). المتغيرات من هذا النوع مناسبة للبيانات المصرفية والأموال.

double

يخزن هذا المتغير بيانات عشرية أيضاً، إلا أن دقتها أقل تصل حتى 16 رقم بعض الفاصلة، والمميّز فيه أن مجال قيم بياناتيه يمكن أن تصل إلى 324 خانة. البيانات من هذا النوع مناسبة للعمليات العلمية والحسابات الدقيقة.

المتغيرات من هذا النوع أسرع من النوع السابق بالنسبة للمعالج.



float

يعتبر هذا لمتغير الشقيق الأصغر لـ double، دقته تصل إلى 7 أرقام بعد الفاصلة فقط، ويخزن بيانات خاناتها 28 رقم بعد الفاصلة.

int

من أشهر المتغيرات التي يعرفها جميع سكان الكره الأرضية والكواكب المجاورة هو هذا المتغير، وهو متغير صحيح لا يحوي فاصلة عشرية، ومجال قيم بياناته تصل حتى 10 خانات، أي مليار تقريباً.

long

الشقيق الأكبر لـ int هو هذا المتغير، شهرته أقل من المتغير السابق، كما أن مجال قيم بياناته تصل حتى 19 خانة، وللصراحة لا أعلم ما هو الرقم الذي له 19 خانة 😊.

string

المتغيرات النصية تخزن قيمةً عبارة عن مجموعة من الحروف والرموز، وحجمها في الذاكرة يختلف بحسب حجم العبارة النصية. لاحظ المثال:

```
string a = "hello";
string b = "h";
b += "ello";
Console.WriteLine(a == b);
Console.WriteLine((object)a == (object)b);
Console.ReadKey();
```

والنتيجة ستكون True ثم False لأن محتوى كل من a و b متساوي، لكنهما لا يمثلان نفس الكائن.

قد تتساءل عن سبب أن النتيجة منطقية، أي أنها True وFalse.. والسبب في أن النتيجة بهذا الشكل هو أن الروابط تحول العبارة من أي نوع إلى منطقية، وهذا ما سنراه في فقرة لاحقة.

بإمكانك استخدام المتغيرات النصية لتخزين العبارات النصية (يا للمصادفة 😂)، ويقولو البرمجة صعبة 😂:

```
string word;
word = "This is a simple string";
```



كما يمكنك تخزين بيانات نصية مكونة من أكثر من سطر كما رأينا في الفصل السابق:

```
string word;
word = "This is a simple string/nThis is other line";
```

ويمكنك تخزين المسارات داخل المتغيرات النصية أيضًا:

```
string path;
path = "d:\\my files\\e.txt";
// Or path = @"d:\\my files\\e.txt";
```

توفر لك C# إمكانية كبيرة في التعامل مع المتغيرات من شتى أنواعها، وذلك عبر توابع عديدة جاهزة، مع مرور الأكواد ستفهم ماهية التوابع، كيفية صناعتها وكيفية استخدامها. الأمر مشترك من أجل جميع لغات البرمجة، مع اختلاف في طريقة سرد الأكواد فقط.

سأناقش بعض هذه التوابع، وطريقة استخدامها هي كتابة اسم المتغير ثم نقطة ثم الخاصية المطلوب التعامل معها.

- لتعريف متغير نصي والقيام بالعمليات عليه:

```
string s = "Hasan";
```

والآن تابع معي العمليات التالية على المتغير الذي عرفناه منذ ثانية ..

- للحصول على عدد أحرف عبارة نصية استخدم التابع Count:

```
Console.WriteLine(s.Count()); // Prints 5
```

- للحصول على أول وأخر حرف استخدم التابعين First وLast :

```
Console.WriteLine(s.First()); // Prints H
Console.WriteLine(s.Last()); // Prints n
```

- لإضافة أحرف إلى عبارة نصية استخدم التابع Insert:

```
Console.WriteLine(s.Insert(2,"ss")); // Prints Hasssan
```



- للحصول على طول عبارة نصية استخدم الخاصية `:Length`

```
Console.WriteLine(s.Length); // Prints 5
```

ملاحظة

- هناك فرق بين عدد العناصر `Count`، والطول `Length`.
- هناك فرق بين التابع والخاصية، التابع يلحق بإشارتي () أما الخاصية فلا تلحق.

- لتصغير جميع أحرف عبارة نصية أو تكبيرها استخدم التابعين `ToLower` و `ToUpper`:

```
Console.WriteLine(s.ToLower()); // Prints hasan
Console.WriteLine(s.ToUpper()); // Prints HASAN
```

- لمعرفة فيما إذا كان محتوى المتغير يحوي قيمة نصية معينة:

```
bool b = s.Contains("s"); //True
```

- لمعرفة مكان وجود قيمة نصية ضمن قيمة نصية أخرى:

```
int i = s.IndexOf("s"); //2
```

- بإمكانك استبدال قيمة نصية ضمن قيمة نصية أخرى:

```
string s2 = s.Replace("n", "N"); //s2 = HasaN
```

كما يمكن الاستفادة من الاستبدال وذلك لإزالة قيم نصية محددة ضمن قيمة نصية أخرى:

```
string s2 = s.Replace("n", string.Empty); //s2 = Hasa
```

- لإزالة الفراغات التي لا طعم لها:

```
s += " ";
string s2 = s.Trim(); //s2 = Hasan
```



كما يمكنك إزالة الفراغات من نهاية القيمة النصية فقط أو من نهايتها فقط استخدم الطرق TrimStart و TrimEnd.

- بإمكانك تقسيم محتوى عبارة نصية إلى عناصر مصفوفة حسب قيمة ما:

```
string x = "This is sample text, we will split it by some punctuations. Try it";
string [] sss = x.Split(new char [] {',', '.' });
foreach (string m in sss)
    Console.WriteLine(m);
Console.ReadKey();
```

والنتيجة ستكون ثلاثة أسطر، كل سطر فيه جملة نصية واحدة مع بعض الفراغات في بداية كل سطر.

- ولدمج النصوص استخدم الطريقة Concern، والتي هي نفسها معامل الجمع +.

كائن StringBuilder

قلنا قبل قليل أن التصريح عن المتغيرات يعني استنساخ فئة تمثل نوع المتغير، وناقشنا مثلاً عن متغير حرف.. ومالم أخبرك عنه هو لو أنه تم تغيير قيمة المتغير فإنه سيعاد تعريفه من جديد، أي أن:

```
string s = "";
s = "Statement";
s += ":";
```

يكافئ:

```
string s = new string();
s = "";
s = new string();
s = "Statement";
s = new string();
s += ":";
```

طبعاً من جديد الكود غير موجه لك، لكن الفكرة والغاية منه موجهة لك بالتأكيد. عد إلى الكود الأول المكون من ثلاثة أسطر، السطر الأول كافأناه بسطرين في الكود المكافئ وقلنا أنه يعني استنساخ فئة نوع المتغير ثم إسناد القيمة للمتغير، أما السطر الثاني من الكود الأول فقلنا أنه يمثل

السطرين الثالث والرابع من الكود المكافئ والذي يتمثل بإعادة استنساخ الفئة ثم إعطاء قيمة جديدة، والسطر الثالث من الكود الأول يكافئه السطرين الآخرين من الكود المكافئ والذي يحتوي على إعادة استنساخ.

طبعاً المشكلة ليست في أن كل سطر يكافئه سطرين - فبالإمكان اختصار الكود المكافئ لتصبح ثلاثة أسطر - وإنما في إعادة الاستنساخ، وهذه مشكلة في حد ذاتها. طبعاً كود مكون من سطر أو اثنين لا يمثل مشكلة إطلاقاً ولكن إذا ماتم تكرار العملية مئات أوآلاف المرات فبالتأكيد هناك مشكلة، والكود التالي يقوم بتنفيذ كود 1000 مرة باستخدام حلقات تكرار سنراها لاحقاً (لا تشغلي بالك بها كثيراً الآن):

```
string s = "";
for (int i = 0; i < 1000; i++)
    s += "*";
Console.WriteLine(s);
Console.ReadKey();
```

حيث يقوم الكود السابق بإضافة الرمز "*" للمتغير S ألف مرة.. تخيل ألف مرة يُعاد فيها التعريف!!

وعوضاً عن ذلك استخدام كائنات من نوع `:StringBuilder`

```
StringBuilder sb = new StringBuilder();
for (int i = 0; i < 1000; i++)
    sb.Append("*");
string s = sb.ToString();
Console.WriteLine(s);
Console.ReadKey();
```

لإنشاء كائن من نوع `StringBuilder` عليك استنساخ واحداً في البداية:

```
StringBuilder s = new StringBuilder();
```

:Appendix لإسناد القيم له استخدم الطريقة

```
s.Append(".");
Console.WriteLine(s.ToString()); //Prints a dot
```



كما يمكنك إسناد مصفوفة له:

```
s.Clear();
char[] c = { '<', 'A', '>' }; // char[]
s = s.Append(c);
Console.WriteLine(s.ToString());
Console.ReadKey();
```

وإمكانيك إسناد حرف معين مكرراً عدداً من المرات:

```
s.Clear();
char c = '.';
s = s.Append(c,5);
Console.WriteLine(s.ToString()); Prints ....
Console.ReadKey();
```

كما يمكنك إضافة قيم للكائن هذا في سطر جديد باستخدام الطريقة AppendLine كما هو حال الطريقة WriteLine.

وكما قلنا فهذا الكائن لا يقوم باستنساخ نفسه عند كل إسناد لذلك فهو أفضل!

تنسيق العبارات النصية

سبق وأن رأينا إمكانية التحويل من متغير إلى آخر، في بعض الأحيان تحتاج لتنسيق المتغيرات لتأخذ شكلًا ما بحيث تفهمه أنت والبرنامج على حد سواء.. الجدول التالي يوضح رموز التنسيق:

الوصف	الرمز
تهيئة الرقم ليكون بشكل العملة المحلية، يتعلق بإعدادات كمبيوترك.	c
تهيئة الرقم بعدد خانات معين، وهو مختص بالأعداد الصحيحة فقط.	d
تهيئة الرقم بواسطة الفاصلة ، للألاف والنقطة . للفواصل، بدقة رقمين بعد الفصلة العشرية.	n
تهيئة الرقم ليظهر بالشكل العلمي، بدقة ست أرقام بعد الفاصلة العشرية.	e
تهيئة الرقم بعدد ثابت من الأرقام العشرية.	f
تهيئة عامة	g
تهيئة الرقم ليظهر بالنظام السنتي عشربي.	x



لاحظ المثال التالي:

```
Console.WriteLine("{0:c}", 27.1); //27.1 $  
Console.WriteLine("{0:d1}", 27); //27  
Console.WriteLine("{0:d5}", 27); //00027  
Console.WriteLine("{0:e}", 2700); //2.700000e +003  
Console.WriteLine("{0:f0}", 27); //27  
Console.WriteLine("{0:f3}", 27); //27.000  
Console.WriteLine("{0:g}", 2.7); //2.7  
Console.WriteLine("{0:n}", 2700); //2,700.00  
Console.WriteLine("{0:x}", 27); //1b  
Console.ReadKey();
```

التحكم بالأخطاء Exception Handling

رأينا كيف نصرّح عن المتغيرات وننسقها، رأينا أيضًا كيفية التحويل من نوع إلى آخر. لكن قد تحدث أخطاء في وقت التنفيذ Run-Time أي عند تشغيل البرنامج، وهذه الأخطاء تنتج بسبب سوء استخدام المتغيرات، أو بسبب أخطاء منطقية كالقسمة على صفر مثلاً.

سوء استخدام المتغيرات معناه أن تسند قيمة نصية مثلاً إلى رقمية أو العكس، أو أي نوعين آخرين من البيانات. قد تقول لي أنتا في فقرة من الفقرات أسندا قيمة نصية لمتغير رقمي بعد تحويلها إليه، أي باستخدام الفئة Convert، وسأجيك بأن هذا الكلام صحيح جزئياً، صحيح أنتا أسندا قيمة نصية "123" إلى متغير رقمي بعد تحويلها لقيمة رقمية 123، لكن القيمة النصية هذه ليست إلا سلسلة من المحارف لأرقام. والخطأ الذي يحصل نتيجة سوء استخدام المتغيرات هي أن تسند قيمة نصية "abc" إلى رقمية.

خلال الأمثلة السابقة لم نأخذ بعين الاعتبار الاستخدام السيء للمتغيرات، وافتراضنا أن المستخدم سيفهم عليك وسيتصرف بمثالية، والواقع أن البرنامج الناجح هو الذي يبتعد عن المثاليات ويتجه نحو الاحتمالات المتشعبة.

خلال الفصول التالية لن نأخذ بعين الاعتبار أيضاً الاستخدام السيء للمتغيرات إلا ما ندر، وذلك حتى لا تضيع الفكرة ويتشتت انتباحك عزيزي القارئ.

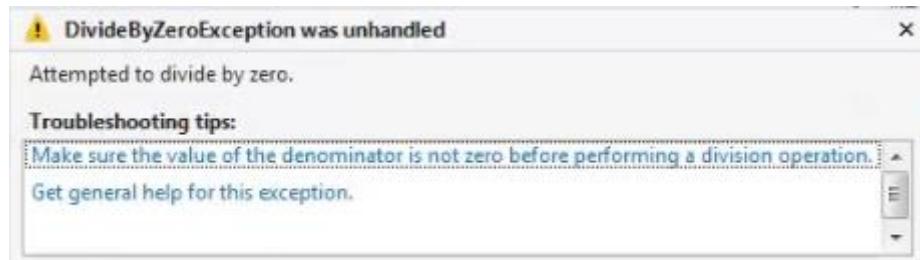
لنستعرض بعض الأمثلة التي قد تحدث فيها أخطاء وقت التنفيذ:



المثال الأول:

```
int x, y;  
x = 5; y = 0;  
double z = x / y; //ERROR!
```

عند تنفيذ البرنامج ستحصل مباشرة على خطأ بالشكل:

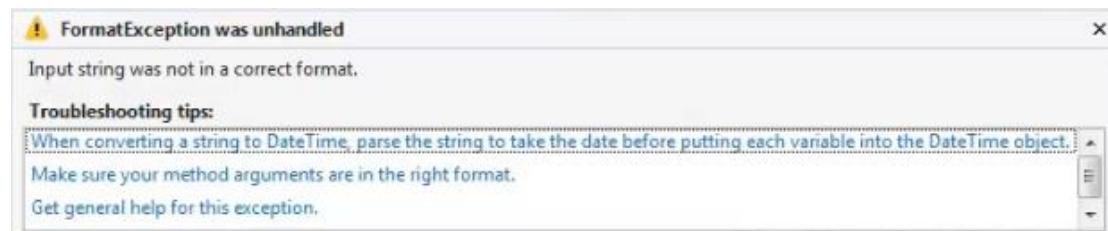


ومعناها أنك تحاول القسمة على صفر.

المثال الثاني:

```
int x;  
Console.Write ("Enter Number :");  
x = Convert.ToInt32 (Console.ReadLine());
```

عند تنفيذ البرنامج وإدخال القيمة "sss" مثلا، ستحصل مباشرة على خطأ بالشكل:

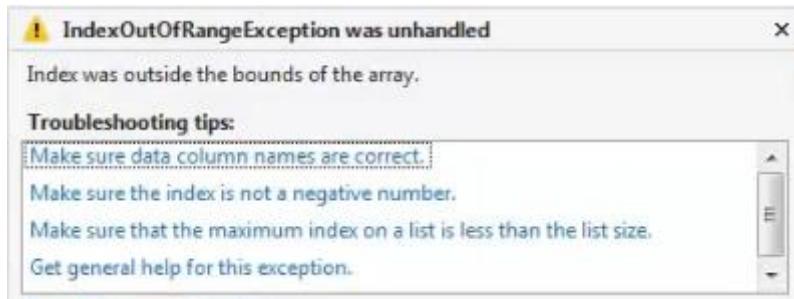


ومعناها أن هناك خطأ في التنسيق وأنواع المتغيرات.

المثال الثالث:

```
int [] x = new int[5];  
x[17] = 27; //ERROR!!
```

عند تنفيذ البرنامج ستحصل مباشرة على خطأ بالشكل:



ومعناها أن هناك خطأ في الدليل وهو خارج حدود المصفوفة (المصفوفات هي مجموعة من المتغيرات تحمل ذات النوع وتعرف ضمن مجموعة واحدة، الفصل الرابع يتحدث عنها).

التحكم باستخدام try – catch

للحصول على برامج ندية وخالية من الأخطاء قدر الإمكان لا بد من استخدام صيغة تخبر فيها C# أن يقتصر لك الخطأ فور حدوثه، وذلك في الأماكن التي تتوقع فيها حدوث أخطاء، ويكون ذلك وفق الصيغة التالية:

```
try
{
    C#'s Codes
    .
    .
}
catch (EXCEPTION)
{Solution of the Exception error}
finally
{C#'s Code to do anyway}
```

ومعنى الصيغة السابقة أنك تخبر C# أن يحاول بالكود المرفق بين أقواس try، فإذا وجد خطأً من النوع المعرف داخل أقواس catch يقوم بما هو داخل أقواسها، وأخيراً عند الخروج من try – catch سينفذ البرنامج محتويات finally سواءً أكان هناك خطأ أم لم يكن.

بإمكانك استخدام أكثر من catch لرصد أكثر من نوع خطأ، وبإمكانك أيضاً عدم استخدام finally.

لتحكم بالأخطاء التي حدثت في الأمثلة الأخيرة:



- رصد الخطأ دون تحديد نوعه:

```
int x, y;
try
{
    x = 5; y = 0;
    double z = x / y;
}
catch
{
    Console.WriteLine ("Can not divide!");
}
Console.ReadKey();
```

- رصد الخطأ من نوع القسمة على صفر، ثم طباعة رسالة تعطي سبب وجود الخطأ:

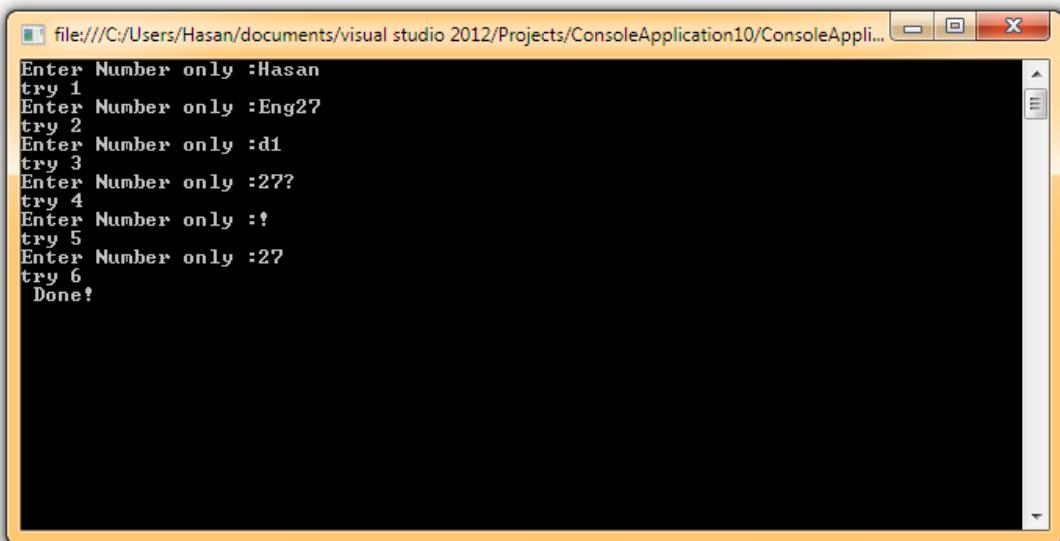
```
int x, y;
try
{
    x = 5; y = 0;
    double z = x / y;
}
catch (DivideByZeroException a)
{
    Console.WriteLine(a.Message);
}
Console.ReadKey();
```

- تكرار المحاولة حتى عدم حصول الخطأ:

```
int x;
int s = 1;
a:
Console.Write("Enter Number only :");
try
{
    x = Convert.ToInt32(Console.ReadLine());
}
catch
{
    goto a;
}
finally
```



```
{
    Console.WriteLine("try " + s);
    s += 1;
}
Console.WriteLine("\b Done!");
Console.ReadKey();
```



وهكذا يامكانك التعامل مع أكثر من نوع من الأخطاء، وذلك بتحديد نوعها في `.catch`.

الروابط

تقسام الروابط إلى الأنواع التالية: روابط رياضية، منطقية، مقارنة وإسناد. حاول التفريق بين الأنواع المختلفة للروابط، وافهم كل أداة ربط ووظيفتها، مع مرور الأكواد ستلاحظ الفرق بنفسك وستتمكن من استخدام أدوات الرابط دون الرجوع لهذه الفقرة حتى.

تستخدم الروابط الرياضية من أجل العمليات الأساسية كالجمع والطرح والجداء والقسمة، الرابط الجديد هو باقي القسمة، ويستخدم عند قسمة عددين بحيث لا يكون الناتج صحيحاً، كقسمة 10 على 3.

الروابط المنطقية تعيد قيمة `True` أو `False`، صح أو خطأ. أكثر منها في برمجك وأعطيها مكاناً مرموقاً فيه، مع أن لها قيمتين فقط إلا أنها تعني الكثير، وفي حياتنا اليومية هناك الكثير والكثير من الأمور التي تأخذ قيمتين فقط، فمثلاً باب الغرفة يأخذ قيمتين مفتوح ومغلق، المصباح يأخذ القيمتين



يعلم أو لا يعلم، الزر مضغوط أو غير مضغوط، صديقك في المنزل أو ليس في المنزل، أو أنه جائع أو شبعان، أو أنك تحبه أو لا تطيقه، وهكذا يمكنك إسقاط المنظور البرمجي هذا على الكثير من مجريات حياتك.

روابط المقارنة تستخدم للمقارنة بين المتغيرات، وتعامل مع أنواع المتغيرات أو قيمها.

روابط الإسناد تستخدم لإسناد القيم للمتغيرات.

الجدول التالي يبين الفرق بين الروابط المختلفة ويحدد استخدام كل رابط:

الوظيفة	أداة الرابط	النوع
الجمع	+	روابط رياضية
الطرح	-	
الجداء	*	
القسمة	/	
باقي القسمة الصحيحة	%	
و (And)	&&	روابط منطقية
أو (Or)		
التحقق من نوع المتغير	Is	روابط المقارنة
أكبر	>	
أصغر	<	
أكبر أو يساوي	>=	
أصغر أو يساوي	<=	
لا يساوي	!=	
يساوي	==	
إسناد	=	روابط الإسناد
إضافة قيمة للمتغير بمقدار 1	++	
طرح قيمة من المتغير بمقدار 1	--	
إضافة قيمة للمتغير بمقدار ما	+=	
طرح قيمة من المتغير بمقدار ما	-=	
جداء المتغير بمقدار ما	*=	
قسمة المتغير على مقدار ما	/=	



ملاحظة

- هناك تقارب كبير بين لغات C المختلفة، من يفهم مبادئ لغة منها بإمكانه استيعاب باقي اللغات ببساطة.
- الأمر مماثل لمن يتعامل مع لغات برمجة ليست من عائلة C، بعض المبادئ مشتركة، أو متقاربة، مثل تعامل نظام التشغيل مع المتغيرات، أو التعامل مع الروابط. لكن هذا لا يعني أنها متماثلة في كل شيء، فقد تشتراك بالطبع، ولكنها مختلفة في الأسلوب.

يمكنك القيام بالعمليات الرياضية المختلفة ضمن متغير، ثم إظهار النتيجة عبر الطريقة WriteLine، ويمكن القيام بها دون وجود متغيرات لحفظ الناتج، لاحظ الأمثلة:

```
Int a, b, c;
a = 5;
b = 6;
c = a + b;
Console.WriteLine(c); // 11
Console.ReadKey();

Int a, b;
a = 5;
b = 6;
Console.WriteLine(a + b); // 11
Console.ReadKey();

Int a, b;
a = 5;
b = 6;
Console.WriteLine(a + b + 4); // 15
Console.ReadKey();

Int a, b, c;
a = 5; b = 6;
c = a + b;
c = c + 4;
c += 4; // c = c + 4;
Console.WriteLine(c); // 19
Console.ReadKey();
```

الكود الأخير رياضيا خاطئ، فلا توجد معادلة من الشكل $c = c + 4$ فمعناها أن $0 = 4$ بعد اختصار c، وهذا مستحيل. أما برمجيا فمعناه أن قيمة المتغير c الجديدة هي قيمته القديمة مضافة لها القيمة 4.



بالمثل يمكن مناقشة كل من الطرح والجداء والقسمة. لكن انتبه، قد تحتاج لتحويل أنواع المتغيرات التي تتعامل معها إلى أنواع أخرى حتى تعطيك الناتج المطلوب. فمثلاً:

```
Console.WriteLine(3 / 2); // 1
Console.WriteLine(3.0 / 2.0); // 1.5
Console.WriteLine(3.0 / 2); // 1.5
```

وبالمثل:

```
Int a = 3, b = 2;
Console.write(a / b); // يطبع القيمة 1
Double a = 3, b = 2;
Console.write(a / b); // تطبع القيمة 1.5
Int a = 3; double b = 2;
Console.write(a / b); // تطبع القيمة 1.5
```

التابع الرياضية Math

كما هو معروف في الرياضيات فإنه لا يكفي فقط القيام بالعمليات الحسابية الأساسية، توفر لك C# مجموعة من التوابع من أجل القيام بالعمليات الرياضية المتقدمة.

ربما تتساءل بعد كل هذه الفقرات ماهي التتابع؟! لا تستعجل على رزقك، سُتُشرح لاحقاً بالتفصيل إن شاء الله، لكن حتى لا يأكلك الفضول سأشرحها لك بإيجاز: التابع هو برنامج جزئي، يحتوي على أكواد معينة تقوم بوظيفة معينة، بحيث يقل عدد أكواد البرنامج الرئيسي Main، وتسهل عملية مشاركة الأكواد الخاصة بهذا التابع مع المبرمجين، كما أن عملية اكتشاف الأخطاء ستكون أسهل لأن العمل المجزأ يسهل تنقيحه وتدقيقه. كما أن التتابع هي مجموعة من الأكواد موجودة بشكل منفصل عن البرنامج تحت عنوان محدد - هو اسم التابع - عند استدعاءها يتم تنفيذ أوامرها. الآن لنترك المواقع المتقدمة لوقتها ولنناقش التابع الرياضية الجاهزة.

الجدول التالي يبين عدة توابع مستخدمة بكثرة مع تفصيلها رياضياً. التابع المثلثية تستقبل زوايا بالراديان حسراً، وهذا الأمر مشترك من أجل جميع لغات البرمجة على حد علمي.



المعنی الرياضي	التابع
$Y = x $	$Y = \text{Math.Abs}(x)$
$Y = x^z$	$Y = \text{Math.Pow}(x,z)$
$Y = \sqrt{x}$	$Y = \text{Math.Sqrt}(x)$
$Y = \ln x$	$Y = \text{Math.Log}(x)$
$Y = \log x$	$Y = \text{Math.Log10}(x)$
$Y = e^x$	$Y = \text{Math.Exp}(x)$
$Y = \sin x$	$Y = \text{Math.Sin}(x)$
$Y = \cos x$	$Y = \text{Math.Cos}(x)$
$Y = \tan x$	$Y = \text{Math.Tan}(x)$
$Y = \sin^{-1} x$	$Y = \text{Math.Asin}(x)$
$Y = \cos^{-1} x$	$Y = \text{Math.Acos}(x)$
$Y = \tan^{-1} x$	$Y = \text{Math.Atan}(x)$
عدد a من الأرقام بعد الفاصلة	$Y = \text{Math.Round}(x,[a])$

كما يمكن التحويل إلى أنظمة العد وفق ما المثال:

```
int num = 27;
Console.WriteLine(Convert.ToString (num, 2)); //11011
Console.WriteLine(Convert.ToString (num, 8)); //33
Console.WriteLine(Convert.ToString (num, 10)); //27
Console.WriteLine(Convert.ToString (num, 16)); //1b
Console.ReadKey();
```

مثال على النسب المثلثية:

```
Console.Write("Enter Angle in deg to calc:");
double deg = Convert.ToDouble (Console.ReadLine());
double rad = deg * 3.14159 / 180;
Console.WriteLine("deg \tSin \tCos \tTan");
double sin = Math.Round(Math.Sin(rad), 2);
double cos = Math.Round(Math.Cos(rad), 2);
double tan = Math.Round(Math.Tan(rad), 2);
Console.WriteLine(deg + "\t" + sin + "\t" + cos + "\t" + tan);
Console.ReadKey();
```



```
file:///C:/Users/Hasan/documents/visual studio 2012/
Enter Angle in deg to calc: 27
deg      Sin      Cos      Tan
27      0.45     0.89     0.51
```

```
file:///C:/Users/Hasan/documents/visual studio 2012/
Enter Angle in deg to calc: 30
deg      Sin      Cos      Tan
30      0.5      0.87     0.58
```

كما يمكنك استخدام `using System.Math` في بداية كودك حتى تصل لطرق مجال الأسماء `System.Math` بشكل أسرع من كتابته في كل مرة.

تسلسل العمليات الرياضية

قد تعاني في بداياتك في البرمجة مع المنطق الرياضي. إذا كنت من طلاب الهندسة أو الرياضيات فهذا يعني أن الموضوع شبه بسيط بالنسبة لك، وإذا لم تكن ذو منحى علمي فرّكز في محتوى هذه الفقرة (هذا لا يعني بطبيعة الحال أن طلاب الهندسة والرياضيات لا يجب أن يركزوا فيها، لا أحد كاملُ العلم).

عند كتابتك للمعادلات الرياضية راعِ التسلسل الموضح في الجدول التالي:

ترتيب التنفيذ	العملية	الأولوية
إذا احتوى كودك على أقواس متفرعة فإنه تنفذ الأقواس الداخلية فالخارجية فالخارجية وهكذا حتى القوس الرئيسي. وفي حال وجود عدة أقواس متغيرة فإن التنفيذ يكون من اليسار إلى اليمين.	الأقواس ()	1
عند وجود أكثر من عملية فإن التنفيذ يكون من اليسار إلى اليمين.	الضرب، القسمة، باقي القسمة	2
عند وجود أكثر من عملية فإن التنفيذ يكون من اليسار إلى اليمين.	الجمع والطرح	3

لاحظ المثال التالي:

$$y = \frac{5(x - 2)}{16} + 22 \left(\frac{(x - 9)^2}{x} + x - 1 \right)$$

الأمر ليس سهلاً أليس كذلك؟؟؟؟

$Y = 5 * (x - 2) / 16 + 22 * (\text{Math.Pow}(x - 9, 2) / x) + x - 1;$
--



لل وهلة الأولى قد تعتقد أن بعض الأرقام والمتغيرات تُجمع أو تطرح مع أرقام ومتغيرات أخرى، مثل $x + 22 + 16$ في المثال السابق، لكن لا تنس أن الأولوية هي للأقواس، ثم الجداء والقسمة، ثم الجمع والطرح مع الأخذ بعين الاعتبار التسلسل من اليسار إلى اليمين. لاحظ كيف تتم العملية في عقل الكمبيوتر إن صح التعبير:

بفرض أن $x = 5^1$:

$$y = \frac{5(5 - 2)}{16} + 22 \left(\frac{(5 - 9)^2}{5} + 5 - 1 \right)$$

$$y = \frac{5(3)}{16} + 22 \left(\frac{(-4)^2}{5} + 5 - 1 \right)$$

$$y = \frac{5(3)}{16} + 22(3.2 + 5 - 1)$$

$$y = \frac{5(3)}{16} + 22(7.2)$$

$$y = \frac{15}{16} + 158.4$$

$$y = 0.9375 + 158.4$$

$$y = 159.3375$$

فكرة أخيرة – المبرمج لا يعلم كل العلوم!

ذات يومٍ، وأثناء تصفحي لمنشورات الفيسيوك وقعت عيني على منشور ضمن أحد مجتمعات البرمجة لفتاة واضح أنها طالبة جامعية انضمت لتجد حلولاً لوظائفها.. المنصور كان عبارة عن صورة تقول بالحرف الواحد:

أوجد جذر المعادلة $0 = 3 - 2x^3$ بطريقة التنصيف معأخذ (1.4) وحساب ثلاثة دورات؟ برماج بلغة السي بطريقة التنصيف.. وقد عنونت المنصور بـ "السلام عليكم عاوزه طريقة حل المعادله بطريقه التنصيف بلغه السي"

¹ لا يوجد شيء اسمه متغير بمجهول في البرمجة على عكس الرياضيات، لا يمكنك إدخال معادلة من الدرجة الثانية لـ x وتطلب من الحاسوب حلها، عليك إيجاد معادلات فيها متغيرات قيمها معلومة وليس متوجهة، وبناءً عليها يتم إيجاد حلول المعادلات المطلوبة.. قد تقول "كيف تكون متغيرات وقيمها معلومة؟!" وسأجيبك بأن هذه المتغيرات تتغير قيمها من قيمة لأخرى، لكن هذا لا يعني أنها مجهولة، كيف تريد من الحاسوب أن يؤسس حلاً مبنياً على مجاهيل؟ بالآخر هو مجرد آلة ولا يمكنه فعل المعجزات.



لا أعلم لماذا يستخدمون حرف الهاء مكان الناء المربوطة.. في حين أنهم يستخدمونها في أماكن أخرى!! المهم، أجبتها في تعليق موجّهٍ لمجهول:

"عندما تعطى سؤالاً ما لمبرمج، سواءً أكان السؤال في الفيزياء أو الرياضيات أو أي مجال علمي، لا تفترض أنه يعلم طريقة حل السؤال.. هذا بالنسبة للمبرمج.. أما نحن، الذين نتعلم البرمجة ولم تمض سنون قليلة على بدننا فيها، فلا خبرة لدينا.. الزبدة: الحق سؤالك بطريقـة حل بالتنصيف" وسوف نحل لك سؤالك من عيوننا".

حاول دائمًا إلـحـاق طـرـيقـة حل المسـأـلة أو خطـوـاتـها أو خـواـرـزمـيـة تـدـعمـها معـها.. حتـى إنـ كـانـتـ المسـأـلةـ منـ ضـمـنـ المـوـادـ التـيـ يـدـرـسـهـاـ الطـلـابـ، أعـطـ العـلـاقـاتـ الـأسـاسـيـةـ وـالـافـتـراـضـاتـ التـيـ تـدـعـمـ ظـرـوفـ المسـأـلةـ..ـ الفـكـرـةـ لـيـسـ فـيـ مـهـارـةـ الـمـبـرـمـجـ -ـ أوـ الـطـالـبـ -ـ فـيـ تـلـكـ الـعـلـومـ،ـ وإنـماـ فـيـ كـيـفـيـةـ تـوـظـيـفـ تـلـكـ الـعـلـومـ وـقـوـانـينـهاـ فـيـ الـبـرـمـجـةـ وـأـكـوـادـهاـ..ـ وـهـذـاـ مـاـيـفـتـقـرـ إـلـيـهـ نـظـامـ الـتـعـلـيمـ -ـ بـشـتـىـ أـقـسـامـهـ وـفـرـوعـهـ -ـ الـقـائـمـ عـلـىـ اـخـتـيـارـ حـفـظـ الـطـلـابـ وـلـيـسـ فـهـمـهـمـ.ـ فـكـلـيـةـ مـثـلـ كـلـيـةـ الـمـيـكـانـيـكـ مـثـلـاـ وـالـتـيـ يـفـتـرـضـ أـنـ تـخـرـجـ مـهـنـدـسـيـنـ قـادـرـيـنـ عـلـىـ تـوـظـيـفـ وـخـلـقـ الـحـلـوـلـ لـلـمـشـاـكـلـ التـيـ تـوـاجـهـهـمـ بـهـدـفـ بـنـاءـ الـمـجـتمـعـ وـإـصـلـاحـهـ وـتـطـوـيرـهـ عـنـدـمـاـ تـمـتـحـنـ الـطـلـابـ بـمـدـىـ حـفـظـهـمـ لـلـقـوـانـينـ -ـ وـالـتـيـ مـنـ الـمـفـتـرـضـ أـنـ تـعـطـىـ فـيـ نـصـ الـمـسـأـلةـ،ـ إـذـ إـنـ الـمـهـنـدـسـ النـاجـحـ هوـ ذـلـكـ الـذـيـ يـخـلـقـ مـنـ الـقـوـانـينـ حـلـوـلـ لـمـشـاـكـلـهـ وـلـيـسـ مـنـ يـبـدـعـ بـحـفـظـهـاـ -ـ أـوـ تـسـأـلـهـ عـنـ تـعـدـادـاتـ وـتـعـارـيفـ سـقـمـ وـعـمـىـ¹ـ كـ "ـعـدـدـ مـوـاصـفـاتـ الـعـاـمـلـ"²ـ أـوـ "ـعـرـفـ أـداـةـ الـقـطـعـ"ـ أـوـ "ـاـشـرـحـ كـيـفـ تـمـ اـسـتـنـتـاجـ الـقـانـونـ الـفـلـانـيـ"ـ هـيـ كـلـيـةـ آـدـابـ بـاـمـتـيـازـ.

بالعودـةـ لـحـديـثـنـاـ،ـ لـاحـقاـ..ـ عـنـدـمـاـ تـصـبـحـ مـبـرـمـجاـ وـتـعـطـىـ مـسـائـلـ -ـ أـوـ طـلـباتـ وـأـفـكـارـ لـتـصـعـبـهاـ فـيـ بـرـامـجـ مـعـيـنةـ -ـ لـاـ تـطـبـقـ مـحتـويـاتـ هـذـهـ الفـقـرـةـ عـلـىـ نـفـسـكـ وـتـطـلـبـ الـخـطـوـاتـ الـعـلـمـيـةـ لـحـلـ الـبـرـمـجـةـ إـلـاـ إـنـ اـسـتـعـصـيـ عـلـيـكـ ذـلـكـ،ـ وـإـنـماـ حـاـوـلـ الـبـحـثـ عـنـهـاـ وـتـعـلـمـهـاـ..ـ سـيـكـسـبـكـ هـذـاـ مـهـارـةـ الـبـحـثـ،ـ وـمـهـارـةـ الـعـلـمـ الـذـيـ تـقـومـ بـالـبـرـمـجـةـ فـيـهـ..ـ

¹ سـقـمـ وـعـمـىـ:ـ سـيـئـةـ لـلـغـاـيـةـ.

² الـعـاـمـلـ:ـ هـوـ ذـاكـ الـإـنـسـانـ الـذـيـ يـعـملـ تـحـتـ إـشـرـافـ الـمـهـنـدـسـيـنـ.ـ تـخـيلـ لـهـذـهـ الـدـرـجـةـ تـعـلـيـمـنـاـ سـيـئـ.



أمثلة تطبيقية

1- برنامج يحسب محيط ومساحة مستطيل بمعرفة بعدها.

$$A = a \cdot b$$

$$P = 2 \cdot (a + b)$$

حيث A مساحة المستطيل، P محطيه، a و b بعدها.

```
double a, b;
Console.Title = "Rectangle..";
Console.Write("Enter width of rectangle: ");
a = Convert.ToDouble(Console.ReadLine());
Console.Write("Enter length of rectangle: ");
b = Convert.ToDouble(Console.ReadLine());
Console.WriteLine("A = " + Math.Round(a * b, 3));
Console.WriteLine("P = " + Math.Round(2*(a + b), 3));
Console.ReadKey();
```



2- برنامج يحسب مساحة ومحيط مثلث بمعرفة طول ضلعين فيه وزاوية بينهما.

$$A = 0.5 \cdot a \cdot b$$

$$P = a + b + c$$

$$c = \sqrt{a^2 + b^2 - 2 \cdot a \cdot b \cdot \cos \alpha}$$



حيث A مساحة المثلث، P محبيطه، a و b و c أطوال أضلاعه، α الزاوية بين ضلعين a و b.

```
const double pi = 3.14159;
double a, b, c, angle;
Console.Title = "Triangle..";
Console.Write("Enter one length of Triangle: ");
a = Convert.ToDouble(Console.ReadLine());
Console.Write("Enter another length of Triangle: ");
b = Convert.ToDouble(Console.ReadLine());
Console.Write("Enter the angle between them (deg): ");
angle = Convert.ToDouble(Console.ReadLine());
angle=angle*pi/180;
double area; area = Math.Round(0.5*a * b* Math.Sin(angle));
Console.WriteLine("A = " + area, 3));
c = Math.Sqrt(a * a + b * b - 2 * a * b * Math.Cos(angle));
Console.WriteLine("P = " + Math.Round(a+b+c, 3));
Console.ReadKey();
```





ملاحظة

- عند تعاملك مع الزوايا ضع في بالك أن لغات البرمجة تتعامل مع الرadian، لذلك عليك التحويل من الدرجات إليه.
- قد تحتاج في بعض الأحيان لتعريف متغيرات تستخدمنها بشكل مؤقت، ستلاحظ ذلك مع مرور الأكواد.

-3- برنامج يطلب من المستخدم درجة الحرارة (بالدرجة المئوية)، ثم يطبع مقابلاتها بالكلفن والفهرنهايت.



$$k = C + 273$$

$$f = C \cdot \frac{9}{5} + 32$$

```
Console.WriteLine("Enter Temperature in Celsius :");
int C = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("k = {0}", C + 273);
Console.WriteLine("f = {0}", C * 9/5 + 32);
Console.ReadKey();
```

-4- يمكن حساب المسافة التي قطعتها سيارة خلال فترة زمنية، حيث تعطى المسافة بعلاقة من الدرجة الثانية:

$$s = 0.5 \cdot a \cdot t^2 + v \cdot t + s_0$$

حيث s المسافة المقطوعة، a تسارع السيارة، v سرعة السيارة لحظة بدء الحركة أي عند s_0 و t_0 . (إن كل من a و v ثوابت من أجل الحركات المتغيرة بانتظام، أما t فمتغير يتعلق باللحظة المدروسة، و s_0 المسافة عند لحظة ما t).

البرنامج التالي يحسب المسافة التي قطعتها سيارة خلال فترات زمنية مختلفة، الفاصل بين كل زمن وأخر هو 0.1 مثلا، إذا ما كان للتسارع قيمة ثابتة:

```
Console.Write("a = ");
double a = double.Parse(Console.ReadLine());
Console.Write("v = ");
double v = double.Parse(Console.ReadLine());
Console.Write("t = ");
double t = double.Parse(Console.ReadLine());
double s;
Console.WriteLine("t\ts");
Console.WriteLine("-----");
for (double i = 0; i <= t; i += 0.1)
```



```
{
    s = 0.5 * a * i * i + v * i + 0; // فرضاً
    Console.WriteLine("{0}\t{1}", i, Math.Round(s, 3));
}
Console.ReadKey();
```

5- تعطى معادلة حركة جسم يسقط تحت تأثير ثقله (سرعته الابتدائية معدومة) بالعلاقة التالية:

$$h = 0.5 \cdot g \cdot t^2$$

أما سرعة الجسم عندما يصل للأرض:

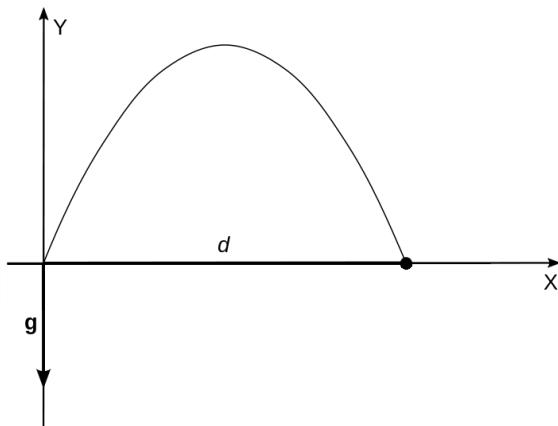
$$v = g \cdot t$$

البرنامج التالي يحسب الزمن اللازم لوصول الجسم للأرض إذا ما أُسقِطَ من ارتفاع h - يدخله المستخدم - وسرعته عندئذ. (علاقة

الزمن مع الارتفاع بناءً على العلاقة المعطاة هي: $(t = \sqrt{\frac{2h}{g}})$

```
Console.Write("h = ");
double h = double.Parse(Console.ReadLine());
const double g = 9.81;
double t = Math.Sqrt(2 * h / g);
double v = g * t;
Console.WriteLine("v = {0} m/s", Math.Round(v, 3));
Console.WriteLine("t = {0} ss", Math.Round(t, 3));
Console.ReadKey();
```

6- تُطلق قذيفة بسرعة ابتدائية v_0 وبزاوية α ، فتقطع مسافة d بزمن طيران t لتبلغ أعلى ارتفاع تصله h ، وفق العلاقات التالية:



$$h = \frac{v_0^2 (\sin \alpha)^2}{2g}$$

$$t = \frac{2v_0 \sin \alpha}{g}$$

$$d = \frac{v_0^2 \sin 2\alpha}{g}$$

حيث $g = 9.81 \text{ m/s}^2$.

البرنامج التالي يحسب المسافة التي تقطعها القذيفة والزمن المستغرق، وأعلى ارتفاع تصل إليه والزمن اللازم لذلك (نصف الزمن الكلي):



```

double a, v0, h, th, t, d;
const double g = 9.81;
Console.WriteLine("v0 (in m/s) = ");
v0 = double.Parse(Console.ReadLine());
Console.WriteLine("angle (in degrees) = ");
a = double.Parse(Console.ReadLine());
double sin = Math.Sin(a * Math.PI/180);
double sin2 = Math.Sin(2 * a * Math.PI/180) ;
d = v0 * v0 *sin2 / g;
t = 2 * v0 * sin / g;
h = v0 * v0 * sin * sin / g / 2;
th = 0.5 * t;
Console.WriteLine("-----");
Console.WriteLine("distance = {0} m", Math.Round(d, 3));
Console.WriteLine("time = {0} s", Math.Round(t, 3));
Console.WriteLine("max height = {0} m", Math.Round(h, 3));
Console.WriteLine("time to get max height = {0} s", Math.Round(th, 3));
Console.ReadKey();

```

```

file:///C:/Users/Hp/Documents/Visual Studio 2012/Projects/ConsoleApplication1/ConsoleApplication1/bin/Debug/C...
v0 (in m/s) = 20
angle (in degrees) = 60
-----
distance = 35.312 m
time = 3.531 s
max height = 15.291 m
time to get max height = 1.766 s

```

تم الحساب من أجل سرعة 20 m/s 20 km/h 72 و من أجل زاوية 60 درجة فكانت النتيجة أن القذيفة ستقطع ما يزيد عن 35 متر خلال أقل من 4 ثوان، لنصل لارتفاع أعظمي 15.3 متر بعد 1.8 ثانية. (حاول دائماً تذوق النتائج، جرب طعمتها و معناها).

7- يمكن حساب النمو في السكان وفق العلاقة الهندسية:

$$p = a(1 + r)^n$$

أو العلاقة الأنبية:

$$p = a \cdot e^{r \cdot n}$$



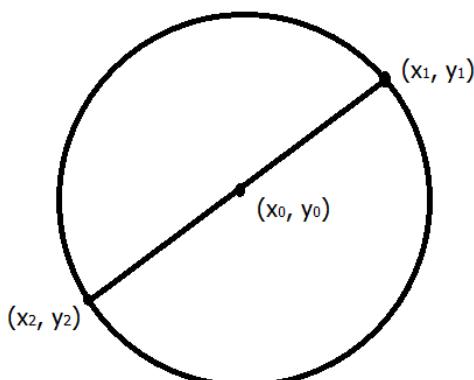
حيث a عدد السكان في بداية الفترة الزمنية المدروسة، r نسبة النمو السكاني، n عدد السنوات المدروسة.

```
Console.WriteLine("a = ");
double a = double.Parse(Console.ReadLine());
Console.WriteLine("r = ");
double r = double.Parse(Console.ReadLine());
Console.WriteLine("n = ");
double n = double.Parse(Console.ReadLine());
double Peng = a * Math.Pow(1 + r, n);
double Pexp = a * Math.Exp(r*n);
Console.WriteLine("Peng = {0}, Pexp = {1}",
    Math.Round(Peng, 2),
    Math.Round(Pexp, 2));
Console.ReadKey();
```

8- تعطى معادلة دائرة بمعرفة نصف قطرها ومركزها بالعلاقة:

$$(x - x_0)^2 + (y - y_0)^2 = R^2$$

كما تعطى إحداثيات مركز الدائرة بمعرفة إحداثيات قطريها بالعلاقة:



$$x_0 = \frac{x_1 + x_2}{2}$$

$$y_0 = \frac{y_1 + y_2}{2}$$

حيث x_1, y_1 إحداثيات النقطة الأولى،
 x_2, y_2 إحداثيات النقطة الثانية. كما

يعطى نصف القطر بالعلاقة: $R = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$

برنامـج لإيجـاد إحداثيات مركز الدائرة ونصف قطرـها:

```
Console.WriteLine("x1 : ");
double x1 = double.Parse(Console.ReadLine());
Console.WriteLine("y1 : ");
double y1 = double.Parse(Console.ReadLine());
Console.WriteLine("x2 : ");
double x2 = double.Parse(Console.ReadLine());
Console.WriteLine("y2 : ");
double y2 = double.Parse(Console.ReadLine());
double x0 = (x1 + x2) / 2;
```



```
double y0 = (y1 + y2) / 2;
double R = Math.Sqrt(Math.Pow(x1 - x0, 2) + Math.Pow(y1 - y0, 2));
Console.WriteLine("A0({0},{1}), R = {2}",
    Math.Round(x0, 2),
    Math.Round(y0, 2),
    Math.Round(R, 2));
Console.ReadKey();
```

```
file:///C:/Users/Hp/Documents/Visual Studio 2012/Projects/ConsoleApplication1/...
x1 : -1
y1 : -1
x2 : 3
y2 : 3
A0(1,1), R = 2.83
```





الفصل الثاني - بنى التحكم

بشكل طبيعي فإن أي برنامج على وجه هذا الكوكب الصغير مصمم على أي لغة برمجة سينفذ سطراً سطراً وبالترتيب المتسلسل الممل. قد تحتاج في بعض الأحيان للتغيير مجريات برنامجك، وذلك اعتماداً على حالة معينة أو مجموعة من الحالات، قد تحتاج لتكرار مجموعة من الأسطر البرمجية مرات عديدة، أو تجاهل أحد الأسطر عند حالة معينة، عندها نستخدم ما يسمى ببني التحكم.

الجيد في بني التحكم أنها قريبة من اللغة البشرية، تستخدم عبارات مثل **إذا حدث شيء ما، عندها سيتتنفيذ شيء ما آخر.**

بنية الشرط If

كثيراً ما تصادف حالات لمتغيرات برنامجك تفترض فيها إمكانية حصول أكثر من حالة، ولكل حالة ناتج تنفيذ معين، ببساطة بإمكانك اختبار قيمة الحالة التي يمر بها برنامجك وفق بنية شرط **If**. عندما يتحقق الشرط ستُنفذ أوصار بنية الشرط. الصيغة العامة لبنيه الشرط هي:

```
If (حالة معينة)  
{  
C#'s Codes  
. . .  
}
```



ملاحظة

- عند وجود أمر واحد ضمن أوامر بنية الشرط C#'s Codes، يمكن الاستغناء عن الأقواس الكبيرة {}.

لأخذ مثلاً واقعياً لفهم الصيغة السابقة، لاحظ الكود الواقعي التالي:

```
if (أحمد في المنزل)
    سوف أزوره;
Console.ReadKey();
```

بسطة أليس كذلك؟ هل تذكر فقرة الروابط في الفصل الأول؟ قلنا عن المتغير المنطقي أنه يأخذ قيمتين.. بنية if تتعامل مع المتغيرات المنطقية، بمعنى أنها تتعامل مع قيمتين فقط، صح أو خطأ.. إذا كان الشرط محققاً فنفذ محتويات البنية، وإلا فلا تنفذ.

قد تقول لي أنه بإمكانك أن تكتب بنية شرط if تتعامل مع الأعداد أو الأحرف أو النصوص أو.. إلخ من أنواع المتغيرات.. وسأجيك بأنه كلامك صحيح بإمكانك فعل ذلك لكن بالنتيجة بنية الشرط تتعامل مع النتيجة المنطقية لنتائج مقارنة متغير أو قيمة ما مع متغير أو قيمة ما آخران.

```
int x; x = 5;
if (x > 0) // True
    Console.WriteLine("True");
Console.ReadKey();
```

بالتأكيد لاحظت أن x هو متغير يحمل قيمة صحيحة، و0 هو قيمة صحيحة، لكن إشارة المقارنة < هي ما حولت العبارة السابقة إلى قيمة منطقية.

على سبيل المثال، أنت أستاذ في مدرسة ولديك طلاب، قدّم الطلاب امتحاناً ما، واحتاجت بناء برنامج يحدد لك فيما إذا كان الطالب ناجحين أم لا. (علامة النجاح 60 من مئة).

قد يخطر ببالك هذا الخاطر: لماذا سأقوم بتصميم برنامج كامل من أجل تقرير ما إذا كان الطالب ناجحين أم لا، بإمكانني مقارنة علاماتهم مع علامة النجاح بنفسي دون أن أتعذب وأصمم برامج بلا طعة.

عزيزي، لا تؤاخذني بهذه الكلمة، شكلك جديد على البرمجة، جميع برامج وأمثلة هذا الكتاب (وجميع برامج وأمثلة جميع الكتب في مجال البرمجة) يمكن اعتبارها برامج جزئية لها أهداف ووظائف ما. في الواقع، من الصعب



تصميم برنامج واحد كقطعة واحدة وإنما يُصمّم على دفعات، هذه الدفعات هي عبارة عن برامج جزئية - تسمى توابع - يمكن برمجتها منفردة، ويمكن لأكثر من مبرمج التعاون لبرمجة عدة برامج جزئية توزّع عليهم بطريقة ما.

بالعودة للمثال، فكّر معي قليلاً، نحتاج لمقارنة علامات طالب ما مع القيمة الافتراضية 60، فإذا كانت أقل منها ستظهر رسالة خطأ. هذا يعني أننا نحتاج متغيرين، الأول يحمل قيمة علامات طالب، والآخر ثابت وهو 60.
لاحظ:

```
const int Default = 60;
int Marks;
Console.Title = "IsFailed?";
Console.Write("Enter Student Mark: ");
Marks = Convert.ToInt16(Console.ReadLine());
if (Marks < Default)
    Console.WriteLine("Failed");
Console.ReadKey();
```

وકشرح بسيط للبرنامج، فالسطر الأول يقوم بتعريف (التصريح عن) متغير ثابت من نوع `int` قيمته 60 (لا يمكن أن تتغير قيمته لاحقاً). السطر الثاني يقوم بتعريف متغير من نوع `int` كذلك الأمر. الثالث يغير عنوان البرنامج. الرابع والخامس يظهران عبارة نصية تطلب من المستخدم إدخال قيمة، ثم يسند أحدهما القيمة للمتغير `Marks` بعد تحويلها من نصية إلى صحيحة `.int`. الزيدة بالسطر السادس، والذي يحوي بنية شرط تتحقق من أن علامة `int` طالب `Marks` هي أصغر من العلامة الافتراضية `Default`، فإذا كان الشرط محققاً كانت نتيجته `True`. والسطر السابع ينفّذ في حال تحقق الشرط السابق، وهو إظهار عبارة مفادها أن الطالب راسب. السطر الأخير يوقف النافذة مؤقتاً ريثما يتكرم علينا المستخدم ويضغط على أحد الأزرار.

مثال آخر لجامعة الرياضيات، لنأخذ التابع الرياضي التالي:

$$y = \ln(x - 1) : x > 1$$

المعروف في الرياضيات أن اللوغاريتم لا يستقبل قيمة سالبة، لذلك عند إجراء حسابات على توابع لوغاريمية يجب التتحقق من أن دخل التابع اللوغاريتمي هو موجب تماماً. تتحقق الفرضية السابقة عندما يكون x أكبر تماماً من الواحد، على اعتبار أن دخل اللوغاريتم هو $1-x$.



بناءً على مسابق، نحتاج مايلي:

- متغير y يحمل قيمة التابع اللوغاريتمي.
- متغير x سوف يدخله المستخدم.
- تابع لوغاريتمي أساسه نّاري (Math.Log())
- كوب كابتشينو.

```
double x, y;
Console.WriteLine("Calculating y = ln(x-1)");
Console.WriteLine("Note that x > 1");
Console.Write("Enter x =");
x = Convert.ToDouble(Console.ReadLine ());
if (x > 1)
{
    y = Math.Log(x - 1);
    Console.Write("y = " + y);
}
Console.ReadKey();
```

الفكرة من المثال ليس فقط تجربة بنية الشرط 😐، لاحظ أن عدد أوامر بنية الشرط إذا كان أكثر من واحد فيجب استخدام أقواس وهذا ماذكرناه سابقاً، وعند عدم استخدامها فإن أول تعليمة بعد أول سطر ستُعتبر أمر بنية الشرط الوحيد.

لنأخذ مثلا آخر من لبّ الـ C#, إذا كانت الساعة أكبر من 6 وأصغر من 10، يطبع لك البرنامج رسالة صباح النور :

```
int Time;
Time = DateTime.Now.Hour;
if (Time > 6 && Time < 13)
    Console.Write("Good Morning :)");
Console.ReadKey();
```

لا داعي لأنسح أنّه يمكن تعدد الشروط داخل شرط واحد أليس كذلك؟ الأمر مماثل من أجل جميع الروابط المنطقية. أعتقد الآن أنه باتت لديك فكرة واضحة عن الفرق بين الروابط المنطقية وروابط المقارنة.



بنية الشرط If .. Else

تعطيك بنية الشرط هذه إمكانية التعامل مع حالة ما بشرطين متعاكسين، بحيث تحدد الشرط الأول وتحدد أковاده كما في بنية الشرط If، ثم تحدد أ Kovad الحالة المعاكسة ضمن بنية Else، وذلك وفق الصيغة:

```
If (حالة معينة)
{
    C#'s Codes
    .
}
Else
{
    C#'s Codes
    .
}
```

ملاحظة

- لا يجب تحديد شرط الحالة الثانية، لغة البرمجة بطبيعة الحال تعكس الشرط الأول.

لأخذ أمثلة بنية الشرط الأساسية، المثال الأول يقول:

أنت أستاذ في مدرسة ولديك طلاب، قدم الطلاب امتحاناً ما، واحتاجت بناء برنامج يحدد لك فيما إذا كان الطالب ناجحين أم لا. (علامة النجاح 60 من مئة). هذه المرة سنحدد فيما إذا كان الطالب ناجح أم راسب، وليس فقط تحديد الطالب الراسب.

```
const int Default = 60;
int Marks;
Console.Write("Enter Student Mark:");
Marks = Convert.ToInt16(Console.ReadLine());
if (Marks < Default)
    Console.WriteLine("Failed");
Else
    Console.WriteLine("Passed");
Console.ReadKey();
```



المثال الآخر كان عن الرياضيات، سنأخذ العلاقة التالية:

$$y = \begin{cases} \frac{-\sqrt{x} + 1}{x}, & x < 0 \\ \sqrt{2 - x}, & x \geq 0 \end{cases}$$

```
double x, y;
Console.WriteLine("Calculating y");
Console.Write("Enter x =");
x = Convert.ToDouble(Console.ReadLine ());
if (x >= 0)
{
    y = Math.Log(2 - x);
    Console.Write("y = " + y);
}
Else
{
    y = (-Math.Sqrt(x) + 1) / x;
    Console.Write("y = " + y);
}
Console.ReadKey();
```

أما بالنسبة لمثال الوقت:

```
int Time;
Time = DateTime.Now.Hour;
if (Time > 6 && Time < 13)
    Console.Write("Good Morning :)");
Else
    Console.Write("Good Evening :)");
Console.ReadKey();
```

مثال آخر يخبرك فيما إذا كان اليوم الحالي هو فردي أم زوجي:

```
bool b;
int day = DateTime.Now.DayOfYear;
إسناد قيمة العبارة الممثلة بباقي القسمة إلى المتغير المنطقي //
b = (day % 2 == 0);
if (b)
    Console.WriteLine("day is an even number");
else
    Console.WriteLine("day is an odd number");
Console.ReadKey();
```



والمثال التالي يريك كيفية إنشاء بنية شرط متفرعة، لمعرفة الحرف المدخل هل هو كبير أم صغير في حال كان حرفًا أصلًا، أما في حال لم يكن حرفًا يظهر لك عبارة بذلك:

```
Console.WriteLine("Enter a character: ");
char c = (char)Console.Read();
if (Char.IsLetter(c))
{
    if (Char.IsLower(c))
        Console.WriteLine("The character is lowercase.");
    else
        Console.WriteLine("The character is uppercase.");
}
else
    Console.WriteLine("Not an alphabetic character.");
Console.ReadKey();
```

بنية الشرط المتعددة

غالباً ما نواجه في برامجك حالات عديدة كثيرة ومتفرعة لحالة ما، قد لا تشع رغبتك بنية If بسيطة ولا حتى بنية If .. Else .. العدد الذي تريده من بنى شرط داخل بنية أساسية. الصيغة العامة لها:

```
If (حالة معينة)
{
    C#'s Codes
}

Else If
{
    C#'s Codes
}

Else
{
    C#'s Codes
}
```



ملاحظة

- بإمكانك عدم استخدام بنية Else .If
- في بنية Else If يجب عليك وضع شرط كما هو الحال في

لأخذ مثلاً عملياً تمر به جميع الكائنات الحية تقريباً. حل معادلة من الدرجة الثانية 😊، تأخذ المعادلات من الدرجة الثانية الصيغة التالية:

$$ax^2 + bx + c = 0$$

حيث أن كل من a و b و c ثوابت تختلف من معادلة لأخرى.

إن تصميم الأمثل للبرنامج يعتمد على خوارزميتك وبناءك لخطوات سير البرنامج. من الطبيعي عند حل معادلة من الدرجة الثانية أن نمر بالخطوات التالية:

1. تحديد كل من a و b و c .
2. حساب المميز Δ .
3. مناقشة حالات المميز (سالب، موجب أو معدوم).
4. حساب حلول المعادلة بناءً على المناقشة في الخطوة السابقة.

```
Console.Title = "Equation Solver By Eng27";
Console.WriteLine("Equation : A x^2 + B x + C");
Console.WriteLine("Please Enter A,B And C:");
Console.Write("A = ");
double A = Convert.ToDouble (Console.ReadLine());
Console.Write("B = ");
double B = Convert.ToDouble (Console.ReadLine());
Console.Write("C = ");
double C = Convert.ToDouble (Console.ReadLine());
double delta = B * B - 4 * A * C;
if (delta>0)
{
    double x1 = (- B + Math.Sqrt (delta))/(2*A);
    double x2 = (- B - Math.Sqrt (delta))/(2*A);
    Console.WriteLine("x1 = " + Math.Round (x1,2));
    Console.WriteLine("x2 = " + Math.Round(x2, 2));
}
else if (delta == 0)
{
    double x = -B / (2 * A);
    Console.WriteLine("x1 = x2 = " + Math.Round(x, 2));
}
else
```



```
Console.WriteLine("Cannot be Solved!");
Console.ReadKey();
```

لا تحاول دائما الاعتماد على المشاريع مفتوحة المصدر، أو الدروس والمشاريع الجاهزة، أو مكتبات الأكواد، حاول فقط أن تستوحى الفكرة من هؤلاء جميعاً مع ابتكار أفكار وأساليب خاصة بك، مع الحفاظ على مراجع ثابتة تعود إليها عند الضرورة، ولا بأس أن تكون هذه المشاريع مفتوحة المصدر أو الدروس والمشاريع الجاهزة أو مكتبات الأكواد هي هذه المراجع، فليس بالضور أن تكون المراجع دائما نظرية جافة..

أمر آخر خذه بعين الاعتبار، الأمثلة التي أضعها في هذا الكتاب بعيدة جدًا عن الواقعية وهي مسيّسة قدر الإمكان لخدمة الفقرة الراهنة، فلم أتناول جميع الاحتمالات على المسائل المدروسة عند حل المسألة حتى لا أشتت تركيزك، أو أهول لك الفكرة وأعقدّها.. في المثال الأخير ماذا لو أدخل المستخدم قيمة غير رقمية مثلا ؟ أو كان الثابت A معدوم ؟؟ ماذا لو ترك قيمة المتغيرة فارغة ؟؟ البرنامج الناجح هو ذاك الحاوي على أكثر عدد من الاحتمالات بأبسط وأكسل الطرق، والبرنامج الأنجح هو ذاك الحاوي على توابع وفئات تجزء البرنامج إلى أكبر قدر ممكن بحيث يصبح البرنامج حلقات متكاملة يمكن استخدام أي حلقة بمعنى عن الحلقات الأخرى في أي برنامج آخر، الأسطر السابقة هي باختصار الفكرة من التوابع والتي سنتناولها بالتفصيل في الفصل السادس إن شاء الله.

قبل الانتقال إلى بنى جديدة لابد من التنويه إلى أن C# يربط else بآخر if قبلها، لذلك كون صداقة مع الأقواس {}.

لاحظ المثال:

```
int x, y;
x = 6; y = 3;
if (x > 4) //True
    if (y > 4) //False
        Console.WriteLine ("x and y > 4");
else
    Console.WriteLine ("x is <= 4");
Console.ReadKey();
```

انسخ الكود السابق إلى فيجوال ستوديو وستلاحظ أنه حاذى else مستوى if الفرعية، لتعتبر حالة ثانية لها.



وعند تنفيذ البرنامج، وطالما أن $x > 4$ فالبنية الفرعية سوف تعالج، وبما أن y ليست أكبر من 4 فالشرط غير متحقق وستتم تنفيذ أكواد else والتي تقول أن x أصغر أو تساوي 4، وفي الواقع هي !!6

ولتلافي المشكلة استخدم الأقواس:

```
int x, y;
x = 6; y = 3;
if (x > 4) //True
{
    if (y > 4) //False
        Console.WriteLine ("x and y > 4");
}
else
    Console.WriteLine ("x is <= 4");
Console.ReadKey();
```

بنية الشرط المختصر

هي اختصار لبنيّة if else، وتكتب بالصيغة التالية:

أمر2 : أمر1 ? شرط ما

إن الأمر1 ينفَّذ إذا كان الشرط متحققاً، بمعنى أنه إذا كانت نتيجته True، بينما الأمر2 فهو معاكس له.

لاحظ المثال الجميل التالي:

```
const int Default = 60;
int Marks;
Console.Write("Enter Student Mark:");
Marks = Convert.ToInt16(Console.ReadLine());
Console.WriteLine (Marks < 60 ? "Failed" : "Passed");
Console.ReadKey();
```

بنية اختيار Switch

على غرار بنية الشرط if، تشغل بنية الاختيار Switch أهمية كبيرة لدى إجراء مقارنة لحالة متغير ما. تفتقر بنية الاختيار للديناميكية والمناورة التي تؤمنُها بنية الشرط، إذ إنّها تستخدم لحالات ومتغيرات وأغراض محدودة.



الصيغة التالية توضح بنية الاختيار:

Switch (متغير معين)

{

Case 1:
 ثابت:

C#'s Codes

.

Break;

Case 2:
 ثابت:

C#'s Codes

.

Break;

.

.

Default:
 C#'s Codes

.

Break;

}

عند استخدام بنية الاختيار فإنه سيحصل ما يلي:

- الدخول إلى البنية بمتغير معين (وليس بحالة معينة كما في الشرط، إذ إننا لن نقارن متغيرات مع بعضها البعض، وإنما سنناقشه قيمة متغير معين).
- المرور بالقيم التي نفترض أن تتحقق، حيث إن كل قيمة مسبوقة بالكلمة المحوزة Case.
- عند تحقق إحدى القيم فإن مجموعة أковاد ستتتندف، ثم يتم الخروج من بنية الاختيار لوجود التعليمية Break.

ملاحظة

- عند عدم وجود التعليمية Break، فإن بنية الاختيار لن يتم الخروج منها مالم يتم المرور بجميع حالات البنية أو المرور بتعليمية Break.

- عند عدم تحقق أي من القيم المفروضة، فإن الحالة Default الافتراضية ستتحقق، وعند عدم وجودها لن يتم تنفيذ شيء (عند عدم تتحقق أي من القيم المفروضة).



عند استخدامك لبنيـة **Switch**, ضع في بالـك أن المتغير الذي تناقـشه في البنـية هو من النوع **int** أو **char**, ولا يجوز مناقـشة متغيرات من الأنواع الأخرى (وإن كنت مضطراً لها فاستخدم بنـية الشرط). كما أن الـقيم التي توضع بعد الكلـمات المحـوزة **Case** يجب أن تكون ثوابـت حـصراً، ولا يجوز أن تكون متـغيرات (وإن كنت مضطراً لها فاستخدم بنـية الشرط).

لـنأخذ المـثال التـالـي:

في أحد الفـنادـق تم إـنشـاء بـرـنامج يـقوم بإـظهـار أسـعار الغـرف من أـجل يوم واحد، بحيث يـختار الزـبـائن الغـرف عـبر الحـوـاسـيب مـثـلاً.

يـقوم البرـنامج بإـظهـار ثلاثة خـيـارات عـلى سـبـيل المـثال، وبـحـسب اختيار الزـبـون يتم إـظهـار سـعر الغـرفة من أـجل يوم واحد فـقط.

```
Console.WriteLine("Menu:\n");
Console.WriteLine("A) Room with one bed");
Console.WriteLine("B) Room with tow beds");
Console.WriteLine("C) Room with three beds");
begin: Console.Write("Choose A,B or C:");
char choise = Convert.ToChar(Console.ReadLine());
switch (choise) {
    case 'a':
    case 'A':
        Console.WriteLine ("Cost of this room is 25$/day");
        break;
    case 'b':
    case 'B':
        Console.WriteLine ("Cost of this room is 40$/day");
        break;
    case 'c': case 'C':
        Console.WriteLine ("Cost of this room is 60$/day");
        break;
    default:
        Console.WriteLine ("Choose A,B or C ONLY"); goto begin;
}
Console.ReadKey();
```

مـلاحظـات حول المـثال السـابـق:

- بنـية الاختـيار تـناقـش متـغير من نوع **Char** وهو متـغير حـرـفي.
- لوـحة المـفاتـيح تـنـدـخـل قـيم من نوع نـصـي **String** لذلك يـجب التـحـويل للـنـوع الحـرـفي.



- قد يدخل الزبون حرف كبير أو حرف صغير لذلك يجب وضع القيمتين بالحساب، إذ إن الكمبيوتر يميز بين الأحرف الكبيرة والأحرف الصغيرة.

ملاحظة

- عند وضع حالة بدون أكواد، فإن الحالة التالية ستتفذ!

- قد يدخل الزبون حرفاً غير موجود من ضمن الخيارات، لذلك يفضل أن يعاد إقلاع البرنامج من أسطرته الأولى (يفضل تطوير الكود السابق بحيث يتم مسح الشاشة وإظهار رسالة خطأ ثم البدء من جديد).

لاحظ الصور التالية، الأولى تم تنفيذ البرنامج دون أية أخطاء (في الاختيار)، الثانية والثالثة تم اختيار غرفة غير موجودة ضمن الخيارات ثم اختيار واحدة موجودة.

```

file:///C:/Users/Hasan/documents/visual studio 2012/Projects/C...
Menu:
A> Room with one bed
B> Room with tow beds
C> Room with three beds
Choose A,B or C:a
Cost of this room is 25$/day

file:///C:/Users/Hasan/documents/visual studio 2012/Projects/...
Menu:
A> Room with one bed
B> Room with tow beds
C> Room with three beds
Choose A,B or C:d
Choose A,B or C ONLY
Choose A,B or C:
Cost of this room is 25$/day

file:///C:/Users/Hasan/documents/visual studio 2012/Projects/...
Menu:
A> Room with one bed
B> Room with tow beds
C> Room with three beds
Choose A,B or C:d
Choose A,B or C ONLY
Choose A,B or C:B
Cost of this room is 40$/day

```

مثال آخر شائع في البرمجة هو آلة حاسبة بسيطة تجري العمليات الحسابية الأربعية وذلك اعتماداً على اختيار المستخدم للعملية الحسابية.

```

Console.WriteLine("Enter tow Numbers:");
Console.Write("num1= ");
double num1 = Convert.ToDouble(Console.ReadLine());
Console.Write("num2= ");
double num2 = Convert.ToDouble(Console.ReadLine());

```



```

Console.WriteLine("Choose The Operator:\n");
Console.WriteLine("1: -");
Console.WriteLine("2: +");
Console.WriteLine("3: x");
Console.WriteLine("4: ÷");
Console.Write("Choose 1,2,3 or 4: ");
int choice = Convert.ToInt16(Console.ReadLine());
switch (choice) {
    case 1:
        Console.WriteLine("Result = " + Convert.ToString(num1 - num2));
        break;
    case 2:
        Console.WriteLine("Result = " + Convert.ToString(num1 + num2));
        break;
    case 3:
        Console.WriteLine("Result = " + Convert.ToString(num1 * num2));
        break;
    case 4:
        if (num2 != 0)
            Console.WriteLine("Result = " + Convert.ToString(num1 / num2));
        else
            Console.WriteLine("Cannot devide by zero!");
        break;
}
Console.ReadKey();

```

في الحالة الرابعة، عندما choice يساوي 4 والتي تعني عملية القسمة، أي كائن حي على وجه هذا الكوكب الجميل يعلم أن القسمة على صفر مستحيلة، لذلك فعليك اختبار حالة المقام، إذا لم يكن معدوماً أجري عملية القسمة، وإلا فلا.

قد تسألني عن إمكانية استخدام بنية اختيار Switch بدلاً من بنية شرط If لتحديد ما إذا كان المقام معدوماً أم لا، سأجيبك بنعم تستطيع ذلك لكن هناك احتمال كبير أن يكون المقام عدداً عشرياً، وقد أشرت سابقاً إلى عدم إمكانية مناقشة الأعداد غير الصحيحة.

قد تتساءل لماذا لا يمكن القسمة على صفر؟! لماذا يمثل هذا العدد كابوساً عند إجراء القسمة. جميع من مرّ على مسائل رياضية وكتب المسائل التي تفترض $0 \neq x$ ، لماذا هذا التمييز برأيك؟!



في الواقع وكما تعلم عزيزي فالمقام كلما صغر زادت قيمة الكسر، وعند الوصول لقيمة قريبة من الصفر، فإن الكسر سيأخذ قيمة كبيرة جداً قد لا تخيلها حتى.

سأشرح لك أكثر، لنقسم العدد 20 على 5، الناتج ببساطة شديدة هو 4. بإمكانك الحصول على هذه القيمة ذهنياً أو باستخدام آلة حاسبة علمية أو بسيطة أو مبرمجة أو أن تسأل أحد العارفين في الرياضيات، أو أن تسأilyني وتقنعني بها. لكن كمبدأ إن قسمة عدد على آخر يعني عدد المرات التي تُنقص فيها العدد الآخر من العدد الأساسي حتى الوصول للصفر، لاحظ:

$$20 - 5 = 15$$

$$15 - 5 = 10$$

$$10 - 5 = 5$$

$$5 - 5 = 0$$

واضح أن العملية استغرقت 4 خطوات!؟ الآن لنقسم 20 على 0، لاحظ:

$$20 - 0 = 20$$

$$20 - 0 = 20$$

$$20 - 0 = 20$$

⋮

$$20 - 0 = 20$$

ستبقى تُنقص القيمة 0 من العدد 20 إلى أن يأخذ الله أمانته منك ولن تنتهي. الموضوع أعقد من $\frac{20}{0}$ التي علمنا إياها في المدارس، الموضوع منطق ومبدأ وليس فقط معادلات ومسلّمات، وهذا المنطق سائد في الرياضيات والفيزياء والبرمجة واللغات حتى.



أمثلة تطبيقية

1- برنامج يطلب من المستخدم اسمه وعمره، بالإضافة إلى جنسه، وبعدها يقوم بطباعة رسالة ترحيبية مسبوقة بـ "أهلاً سيد فلان" إذا كان المستخدم ذكراً، و"أهلاً سيدة فلانة" إذا كانت أنثى. مع إظهار الوقت الذي قام بتشغيل البرنامج فيه.

```
// 1
string name;
int age;
char gender;
string time;
// 2
time = DateTime.Now.ToShortTimeString();
Console.Title = "Welcome to C# Program";
// 3
Console.Write("Enter Your Name : ");
name = Console.ReadLine();
Console.Title = "Logged in by " + name;
Console.Write("Enter Your Age : ");
age = Convert.ToInt16(Console.ReadLine());
Console.Write("Press [m] for Male, [f] for Female : ");
gender = Convert.ToChar(Console.ReadLine());
// 4
if (gender == 'm')
{
    Console.WriteLine("Welcome Mr. " + name);
    Console.WriteLine ("Your age is "+ age);
}
else if (gender == 'f')
{
    Console.WriteLine ("Welcome Mrs. " + name);
    Console.WriteLine ("Your age is " + age);
}
else
    Console.WriteLine ("Error 101..");
// 5
Console.WriteLine("you started the program at " + time);
Console.ReadKey();
```

شرح البرنامج:

- تعریف أربع متغيرات، لحفظ اسم المستخدم وعمره، ووقت دخوله للبرنامج، بالإضافة إلى جنسه (متمثلاً بحرف).



- إسناد قيمة الوقت الحالي للمتغير المخصص لحفظ قيمة الوقت بصيغة الوقت القصير #:#:# ص/م، بالإضافة إلى تغيير عنوان البرنامج.
- الطلب من المستخدم اسمه وإسناده مباشرة للمتغير الخاص بحفظ اسم المستخدم، وفي ذات اللحظة تغيير عنوان البرنامج إلى "تم تسجيل الدخول من قبل .."، ثم الطلب من المستخدم عمره وجنسه مع إسناد المدخلات إلى المتغيرات المخصصة لهذا الغرض. (لا تنس تحويل نوع بيانات المدخلات من النصية إلى غيرها عند الحاجة).
- مقارنة المتغير الحاوي على جنس المستخدم بقيم ثابتة مرجعية، فإذا كان "m" أظهر عبارة "أهلا سيد فلان"، وإذا كان "f" أظهر عبارة "أهلا سيدة فلانة"، وإلا أظهر عبارة خطأ.
- إظهار الوقت الذي تم تشغيل البرنامج فيه.

```
Logged in by Hasan
Enter Your Name : Hasan
Enter Your Age : 22
Press [m] for Male, [f] for Female : m
Welcome Mr. Hasan
Your age is 22
you started the program at 06:34 PM
```

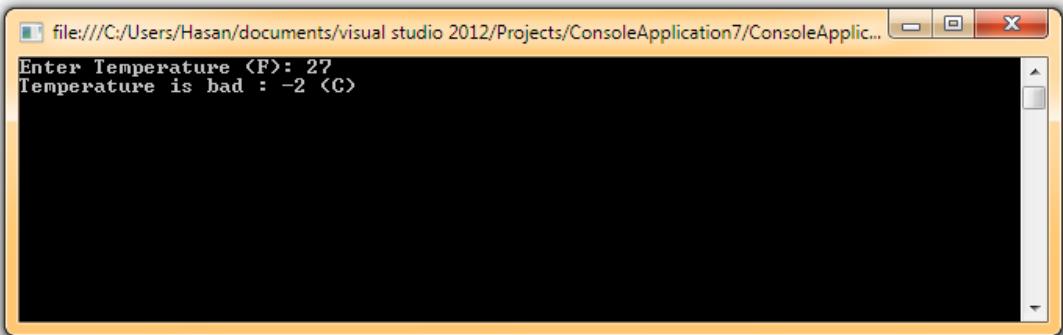
2- برنامج يدخل درجة الحرارة مقدرة بالفهرنهايت، يقوم بتحديد فيما إذا كانت مناسبة للجو أم لا، ثم يقوم بطباعة مايساويها في الفهرنهايت. حيث أن سلم درجات الحرارة بالفهرنهايت يتراوح ضمن المجال (212 - 32) ودرجة حرارة الغرفة النظامية هي ضمن المجال (15 - 25) درجة مئوية. والعلاقة التي تربط بين مقاييس درجات الحرارة هي:

$$C = \frac{(F - 32) \cdot 5}{9}$$

```
int c, f;
Console.Write("Enter Temperature (F): ");
```



```
f = Convert.ToInt16(Console.ReadLine());
if (f < -32 || f > 212)
    Console.WriteLine("Temperature is out of range!");
else
{
    c = (f - 32) * 5 / 9;
    if (c >= 15 && c <= 25)
        Console.WriteLine("Temperature is good : " + c + " (C)");
    else
        Console.WriteLine("Temperature is bad : " + c + " (C)");
}
Console.ReadKey();
```



لا أظن أنه من الضروري شرح الأمثلة التطبيقية على غرار المثال الأول، فقد أصبحتم كباراً الآن بوصولكم لنهاية هذا البحث.

3- برنامج يُدخل إحداثيات نقطة (x,y)، ثم يحدد ما إذا كانت واقعة على أحد المحاور الإحداثية أو ضمن أحد الأرباع أو على نقطة المبدأ..

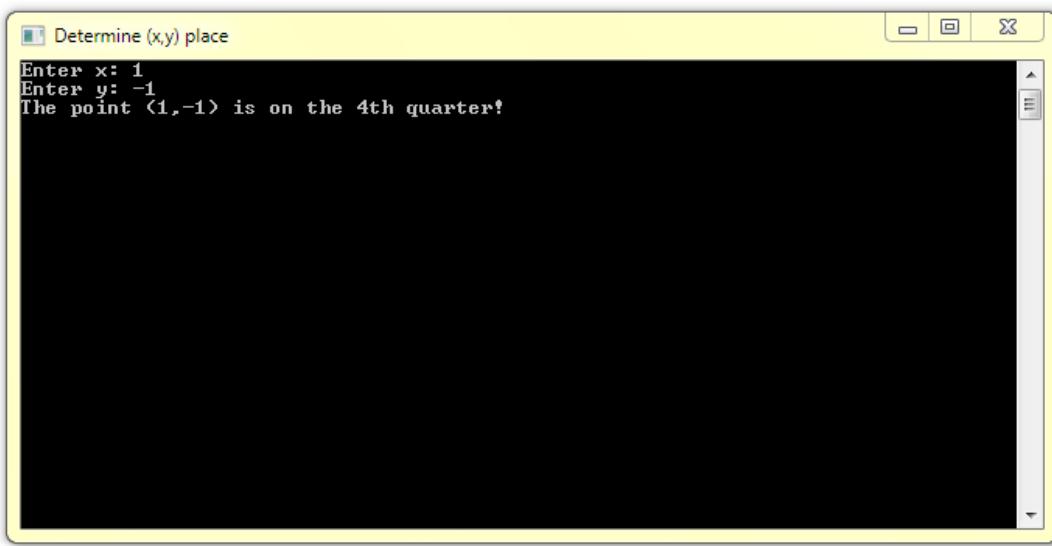
```
double x, y;
Console.Title = "Determine (x,y) place";
Console.Write("Enter x: ");
x = Convert.ToDouble(Console.ReadLine());
Console.Write("Enter y: ");
y = Convert.ToDouble(Console.ReadLine());
if (x > 0 && y > 0)
    Console.WriteLine("The point (" + x + "," + y + ") is on the 1st quarter!");
else if (x < 0 && y > 0)
    Console.WriteLine("The point (" + x + "," + y + ") is on the 2nd quarter!");
else if (x < 0 && y < 0)
    Console.WriteLine("The point (" + x + "," + y + ") is on the 3rd quarter!");
else if (x > 0 && y < 0)
    Console.WriteLine("The point (" + x + "," + y + ") is on the 4th quarter!");
```



```

else if (x == 0 && y != 0)
    Console.WriteLine("The point (" + x + "," + y + ") is on y'y");
else if (x != 0 && y == 0)
    Console.WriteLine("The point (" + x + "," + y + ") is on x'x");
else
    Console.WriteLine("The point (" + x + "," + y + ") is O");
Console.ReadKey();

```



4- برنامج يدخل ثلاثة أطوال صحيحة، ومن ثم يتحقق من إمكانية تشكيل مثلث من هذه الأطوال، فإذا كان ممكناً يحدد هل هو قائم أم لا.

يمكن تحديد إمكانية تشكيل مثلث من عدمه من القاعدة التالية: "إن أي ضلع في مثلث هي أصغر من مجموع الצלعين الآخرين" وهذا يقتضي أن يكون $a < b + c$ حيث a أكبر الأضلاع، و b و c الصلعين الآخرين. مما سبق نستنتج الحالات التالية:

- يمكن تشكيل مثلث $a < b + c$.
- يكون المثلث قائم عندما $a^2 = b^2 + c^2$
- لا يمكن تشكيل مثلث $a \geq b + c$

```

Console.Title = "TRIANGLE CHECKER!";
int a, b, c;
Console.WriteLine("Enter a,b and c: ");
Console.Write("a = ");
a = Convert.ToInt16(Console.ReadLine());
Console.Write("b = ");
b = Convert.ToInt16(Console.ReadLine());
Console.Write("c = ");

```



```
c = Convert.ToInt16(Console.ReadLine());
// *
int temp;
if (a < b)
{
    temp = a;
    a = b;
    b = temp;
}
if (a < c)
{
    temp = a;
    a = c;
    c = temp;
}
if (a >= c + b)
    Console.WriteLine("Triangle can't be formed!");
else if (a * a == b * b + c * c)
    Console.WriteLine("Right Triangle can be formed!");
Else
    Console.WriteLine("Triangle can be formed!");
Console.ReadKey();
```

ملاحظة

- يجب تحديد أكبر ضلع من أضلاع المثلث لتطبيق القاعدة المذكورة في نص المسألة.





5- يمكن معرفة ما إذا كان مستقيم ما قاطع دائرة من خلال حساب أقصر بعد لمركز الدائرة عن المستقيم. وأقصر بعد بين مستقيم ونقطة هو الخط العمودي على المستقيم والمدار من النقطة، والذي يحسب من العلاقة التالية:

$$L = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$$

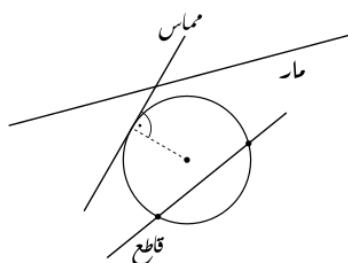
فإذا كان البعد L أكبر من نصف قطر الدائرة كان المستقيم غير مار بالدائرة (هناك خطأ في الصورة الموضحة لحالات مستقيم مع دائرة بالنسبة لهذه الحالة)، وإذا كان مساوياً لنصف القطر كان مماساً للدائرة، وإلا فكان قاطعاً لها.

حيث A, B و C ثوابت المستقيم:

$$Ax + By + C = 0$$

وكل من x_0 و y_0 إحداثيات مركز الدائرة. البرنامج التالي يحدد الوضع النسبي بالنسبة لمستقيم مع دائرة يدخلهما المستخدم:

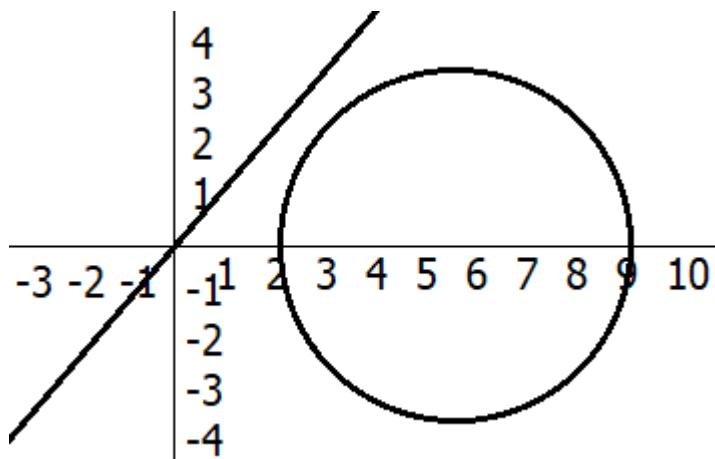
```
Console.WriteLine("Enter A, B and C in Ax + By + C = 0");
Console.Write("A = ");
double A = double.Parse(Console.ReadLine());
Console.Write("B = ");
double B = double.Parse(Console.ReadLine());
Console.Write("C = ");
double C = double.Parse(Console.ReadLine());
Console.WriteLine("Enter x0 and y0 in A0(x0, y0), and R");
Console.Write("x0 = ");
double x0 = double.Parse(Console.ReadLine());
Console.Write("y0 = ");
double y0 = double.Parse(Console.ReadLine());
Console.Write("R = ");
double R = double.Parse(Console.ReadLine());
double L = Math.Abs(A * x0 + B * y0 + C) / Math.Sqrt(A * A + B * B);
if (L > R)
    Console.WriteLine("Line is outside the circle");
else if (L == R)
    Console.WriteLine("Line is tangent to circle");
else
    Console.WriteLine("Line breaks circle");
Console.ReadKey();
```





```
file:///C:/Users/Hp/Documents/Visual Studio 2012/Projects/ConsoleApplication1/ConsoleApplication1/...
Enter A, B and C in Ax + By + C = 0
A = -1
B = 1
C = 0
Enter x0 and y0 in A0(x0, y0), and R
x0 = 5.5
y0 = 0
R = 3.5
Line is outside the circle
```

والشكل التالي يوضح مثلاً بيانياً للقيم السابقة. مع العلم أن المستقيم منصف للربع الأول أي أن معادلته $y = x$ والتي يجب أن تكتب $x - y = 0$ لتصبح بالشكل المستخدم في الحل.







الفصل الثالث - بنى التكرار

كثيرةً هي الحالات التي تتطلب تنفيذ كود أو مجموعة من الأكواد بشكل مكرر. بغض النظر عن عدد مرات هذا التكرار فكتابه الكود للمرة الثانية أو الثالثة سيكون مملاً حتى مع إمكانية النسخ واللصق. وبعدم غض النظر عن عدد مرات التكرار فوجود بنية تسمح بتكرار تنفيذ الكود من أجل عدد معين من المرات وتحت شروط معينة هو من الأمور العظيمة التي اخترع في البرمجة (لا تستهون فيها).

على غرار بنى الشرط، فإن بنى التكرار لها أكثر من صيغة واحدة، ولكلٍ استعمال وفضيل.

حلقة For

تعتبر من أشهر بنى التكرار والحاضي والماضي¹ يعرفها، وهي مشتركة في كثير من لغات البرمجة على اختلاف أساليب التصريح عنها من أجل ذات المبدأ. الصيغة العامة لها:

```
(خطوة عداد الحلقة; شرط التكرار; القيمة الابتدائية) For
{
C#'s Codes
.
.
}
```

¹ الحاضي والماضي: الجميع، كل الناس.



إن الصيغة السابقة تعني - بلغة إنسانية أكثر منها برمجية - أنه من أجل القيمة الابتدائية لعداد //الحلقة سيتم تنفيذ أكواد الحلقة بشكل متكرر من أجل شرط التكرار، مع الأخذ بعين الاعتبار خطوة عداد الحلقة.

إن خطوة عداد الحلقة هي معدل زيادة أو نقصان عداد الحلقة.

ملاحظة

- عند وجود أمر واحد ضمن أوامر حلقة التكرار C#'s Codes، يمكن الاستغناء عن الأقواس الكبيرة {}.
- القيمة الابتدائية هي قيمة عداد الحلقة.

لأخذ مثلاً عن طباعة عبارة نصية خمس مرات باستخدام حلقة تكرار ودون استخدام، لاحظ:

```
n = DateTime.Now.ToString();
Console.WriteLine("Welcome Eng27 Time is: " + n);
n = DateTime.Now.ToString();
Console.WriteLine("Welcome Eng27 Time is: " + n);
n = DateTime.Now.ToString();
Console.WriteLine("Welcome Eng27 Time is: " + n);
n = DateTime.Now.ToString();
Console.WriteLine("Welcome Eng27 Time is: " + n);
n = DateTime.Now.ToString();
Console.WriteLine("Welcome Eng27 Time is: " + n);
Console.ReadKey();
```

يقوم هذا الكود بطباعة العبارة داخل التنصيص، ملحوظة بالوقت الحالي (تنسيق وقت قصير HH:MM AM/PM). لاحظ الفرق عند استخدام بنية تكرار:

```
int i; string n;
for (i = 1; i <= 5; ++i)
{
    n = DateTime.Now.ToString();
    Console.WriteLine("Welcome Eng27 Time is: " + n);
}
Console.ReadKey();
```

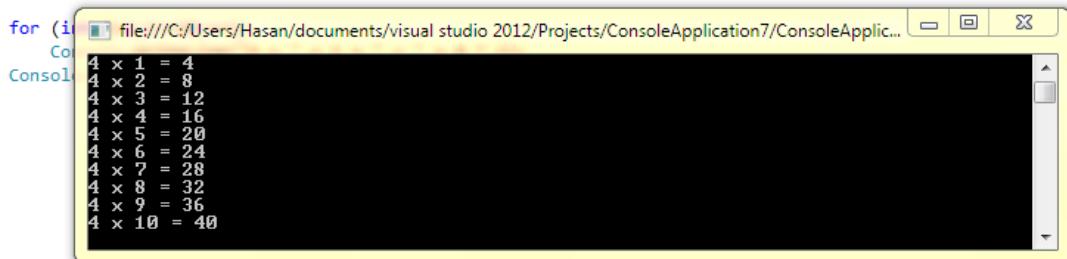


مثال آخر، لنطّبع جدول ضرب أحد الأعداد 😊.

فكرة معي، كم متغير لدينا؟؟؟؟ لدينا متغير واحد، العدد الذي سنكتب جدوله ثابت، أما الأعداد التي سنضربها به فهي متغيرة (من 1 حتى 10).

إذا فكرت معي أكثر ستلاحظ أنه بالإمكان الاستفادة من عداد الحلقة في إنشاء هذا الجدول، وبالنسبة للشرط فهو بسيط جدا، طالما عداد الحلقة أصغر من 9 استمر في العمل. لاحظ:

```
for (int i=1;i<=10;++i)
    Console.WriteLine("4 x " + i + " = " + 4 * i);
Console.ReadKey();
```



يمكنك تنفيذ هذا المثال على جميع لغات C بنفس الأكواد حتى (مع تغيير بسيط في أكواد الإدخال والإخراج و..)

مارائیک لو کتیا کل سطر سطرہ دون استخدا م حلقات؟

```
Console.WriteLine("4 x 1 = " + 4 *1);
Console.WriteLine("4 x 2 = " + 4 *2);
Console.WriteLine("4 x 3 = " + 4 *3);
Console.WriteLine("4 x 4 = " + 4 *4);
Console.WriteLine("4 x 5 = " + 4 *5);
Console.WriteLine("4 x 6 = " + 4 *6);
Console.WriteLine("4 x 7 = " + 4 *7);
Console.WriteLine("4 x 8 = " + 4 *8);
Console.WriteLine("4 x 9 = " + 4 *9);
Console.WriteLine("4 x 10 = " + 4 *10);
Console.ReadKey();
```

إعماقة إلى آخر الحدود.



مثال آخر، لنقم بجدول الضرب كاملاً! 😊😊

ماذا تغير لدينا؟ لدينا متغيران، الأول هو العدد الذي سنحسب جدول الضرب له، الآخر هو الأعداد التي سنضربها بالعدد الأول.

من البديهي من المناقشة السابقة أن نقول أنه لدينا حلقتا `for`، لكل منها عداد منفصل، وبالتفكير مليأً بالموضوع نلاحظ أن عداد الحلقة الأولى يزداد عند تغيير قيمة عداد الحلقة الثانية من القيمة 1 وحتى 10، بمعنى آخر فإن الحلقة الثانية يجب ألا يتحقق شرطها حتى تزداد قيمة عداد الحلقة الأولى بمقدار خطوة واحدة.

عادةً في بعض كتب شرح لغات البرمجة أمثل `c++` وغيرها يتم إنشاء فقرة كاملة محتواها "حلقات `for` المتداخلة"، ومفادها أن إحدى الحلقات ستتكرر بشكل كامل من أجل خطوة واحد للحلقات الأخرى، لن أستخدم هذا الأسلوب وسأكتفي بمثال:

```
for (int i = 1; i <= 10; ++i)
{
    for (int j = 1; j <= 10; ++j)
        Console.WriteLine(i + " x " + j + " = " + i * j);
    Console.Write("\n");
}
Console.ReadKey();
```

لا داعي لتجربة كتابة الأكواد يدوياً أليس كذلك؟؟؟ وعلى غرار المثال السابق بإمكانك تنفيذ هذا المثال على لغات C مختلفة.

```
file:///C:/Users/Hasan/documents/visual studio 2012/Projects/ConsoleApplication7/ConsoleApplic...
8 x 9 = 72
8 x 10 = 80

9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90

10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
```



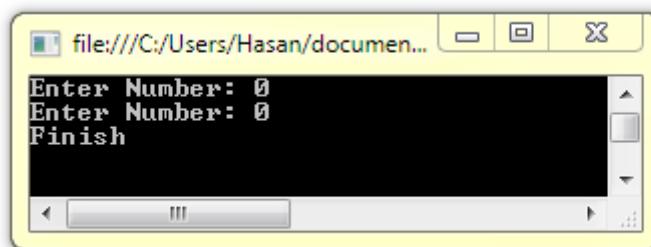
حلقة do – while

لا تختلف كثيراً هذه الحلقة عن حلقة for، إلا أن هذه الحلقة تنفذ التعليمات مرة واحدة على الأقل. الصيغة العامة لها:

```
Do
{
    C#'s Codes
    .
    .
} While (شرط التكرار);
```

كمثال، لنطلب من المستخدم أن يدخل عدداً ما، فإذا كان موجباً أعدنا عليه الطلب:

```
int a;
Console.Write("Enter Number: ");
a = Convert.ToInt32(Console.ReadLine());
do
{
    Console.Write("Enter Number: ");
    a = Convert.ToInt32(Console.ReadLine());
} while (a > 0);
Console.WriteLine("Finish");
Console.ReadKey();
```



حلقة while

تشابه حلقة for كثيراً وتحتاج إليها بطريقة التنفيذ، كما أنها تختلف عن حلقة do – while بأنها لا تنفذ التعليمات إلا إذا تحقق الشرط، ولها الصيغة:



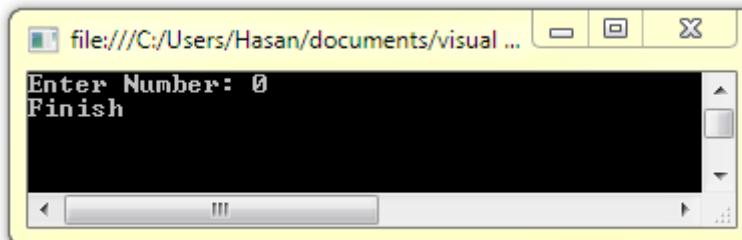
(شرط التكرار)

{C#'s Codes

.}

لأخذ نفس المثال السابق:

```
int a;
Console.WriteLine("Enter Number: ");
a = Convert.ToInt32(Console.ReadLine());
while (a > 0)
{
    Console.WriteLine("Enter Number: ");
    a = Convert.ToInt32(Console.ReadLine());
}
Console.WriteLine("Finish");
Console.ReadKey();
```



هل لاحظت الفرق بين الحلقتين؟ في المثالين تعمدت إسناد القيمة 0 للمتغير a، في الأولى تم تنفيذ الكود رغم أن القيمة ليست موجبة تماماً، وفكرة البرنامج - أو الكود - كما لاحظت هو إعادة طلب الرقم مادام موجباً، وهذا مالم يؤخذ بعين الاعتبار في الحلقة الأولى لمرة واحدة فقط، بينما تم مراعاته في الثانية.

عادةً نستخدم حلقة while لنفس الغرض وللحصول على نفس النتيجة بفارق بسيط، كما لاحظت في الحلقة الأولى تم تنفيذ الكود دون التحقق من الشرط، ثم تم التتحقق من الشرط والتكرار في حال تتحققه. ماذا لو لم يكن الشرط متحققاً من بداية الحلقة؟ الجواب هو أن الكود سيتم تنفيذه مرة واحدة، وهو ماليس موجوداً في حلقة while.

بمعنى آخر، في بعض الأحيان تحتاج لتنفيذ كود ما، ثم تكراره عند تحقق الشرط. عوضاً عن كتابة الكواد خارج الحلقة ثم كتابته من جديد داخل



حلقة التكرار، أو استخدام التوابع كما سترى في البحث السادس،
.do while حلقة نستخدم.

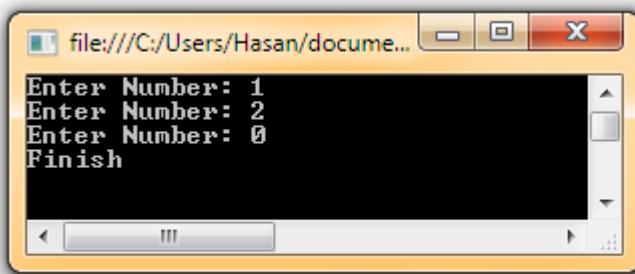
حلقة label goto

الكلمة الممحوza goto تستخدم للتحكم أكثر ما هي للتكرار، هذا لا ينفي قابلية استخدامها للتكرار، صيغتها العامة:

```
Label:  
C#'s Codes  
. .  
Goto label;
```

لاحظ نفس المثال السابق هنا:

```
int a;  
Eng27:  
Console.WriteLine("Enter Number: ");  
a = Convert.ToInt32(Console.ReadLine());  
if (a > 0)  
{  
    goto Eng27;  
}  
Console.WriteLine("Finish");  
Console.ReadKey();
```



هناك حلقة أخرى ضمن حلقات التكرار هي foreach تستخدم مع المصفوفات، سأتحدث عنها في فصل المصفوفات إن شاء الله تعالى.



التحكم بالحلقات

توفر لك العديد من لغات البرمجة وسائل للمناورة في أكوادها، ومن هذه الوسائل هي المناورة أثناء تنفيذ الحلقات التكرارية، قد تحتاج لتجاهل إحدى التكرارات أو إيقاف الحلقة عند تكرار معين، وسنناقش وسليتين في كتابنا هذا:

باستخدام **continue**

تستعمل لتجاهل التكرار الذي تحقق فيه شرط ما، وهي موضحة بالمثال التالي:

```
for (int i = 1; i<=10; ++i)
{
    if (i == 5)
        continue;
    Console.WriteLine (i);
}
Console.ReadKey();
```

سيطبع جميع الأعداد عدا 5.

باستخدام **break**

تستعمل للخروج من الحلقة عند تحقق شرط، وهي موضحة بالمثال التالي:

```
for (int i = 1; i<=10; ++i)
{
    if (i == 5)
        break;
    Console.WriteLine (i);
}
Console.ReadKey();
```

سيطبع جميع الأعداد الأصغر 5.



أمثلة تطبيقية

1- جمع الأعداد داخل مجال معين.

سيطلب البرنامج من المستخدم إدخال عددين، بحيث يشكلان مجالاً ليجمع الأعداد داخله.

```
int a, b, sum = 0;
Console.Write ("Enter the first number : ");
a = Convert.ToInt32 (Console.ReadLine ());
Console.Write ("Enter the second number : ");
b = Convert.ToInt32 (Console.ReadLine ());
if (a > b)
    for (int i = b; i < a; ++i)
        sum = sum + i;
else
    for (int i = a; i < b; ++i)
        sum = sum + i;
MessageBox.Show ("The Sum of Numbers is " + sum,
    "Sum Between " + a + " and " + b);
```

لا تنسَ إضافة مكتبة `NuGet` كمرجع للبرنامج لتمكن من عرض صندوق الرسالة، راجع الفصل صفر ضمن فقرة الإخراج بصندوق رسائل.

2- شخص لديه 1000 سهم، وكل سهم قيمته دولار واحد، والشركة التي يستثمر فيها لها عوائد ثابتة تقريرياً كل سنة، ومقدارها 5% من قيمة السهم. وهذا الشخص يقوم بشراء أسهم جديدة كل سنة بالمبالغ التي يحصل عليها من الشركة. والمطلوب حساب قيمة الأسهم مقدرة بالليرة السورية مع نهاية كل سنة، ولمدة عشر سنين (اعتبر أن كل سهم يساوي دولاراً واحداً لمدة عشر سنين، وسعر صرف الدولار بالسنة لليرة السورية يدخله المستخدم وافتراض أن قيمته ثابتة طيلة المدة المدرستة).

$$A = Q (rate + 1)^y$$

حيث:

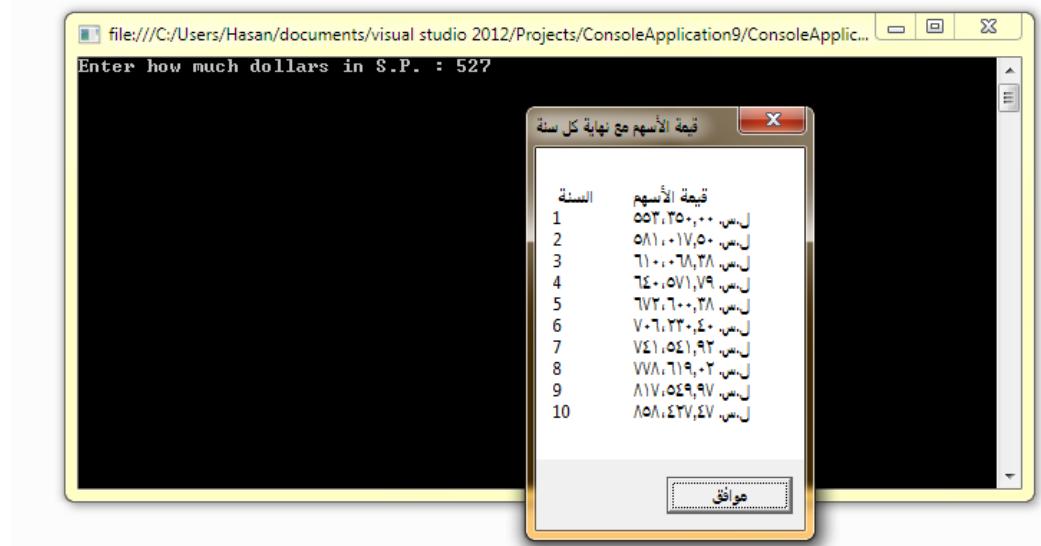
A قيمة الأسهم مع نهاية السنة.

Q القيمة الأصلية.

Rate معدل الدخل كل سنة و Year عدد السنين.



```
double A, Q = 1000;
double rate = 0.05;
double DollarToSP;
string Result;
Console.Write("Enter how much dollars in S.P. : ");
DollarToSP = Convert.ToDouble(Console.ReadLine());
Result = "الأسهم قيمة \t السنة \n";
for (int year = 1; year <= 10; year++)
{
    A = DollarToSP * Q * Math.Pow(1 + rate, year);
    Result += year + "\t" + String.Format("{0:c}", A) + "\n";
}
MessageBox.Show(Result, "قيمة الأسهم مع نهاية كل سنة");
```



3- كتابة العدد على شكل جداء عوامل!

هناك طريقة رياضية للحصول على عوامل الأعداد، وذلك بقسمتها بشكل متتال على الأعداد الأولية¹ (2، 3، 5، 7، 11، 13، ..). برمجيا يمكنك الحصول عليها من البرنامج التالي:

```
int number, d = 2;
Console.Write("Enter Number to get its prime factors: ");
number = Convert.ToInt32(Console.ReadLine());
while (number > 1)
{
    while (number % d == 0)
    {
        Console.Write(d);
        number /= d;
    }
}
```

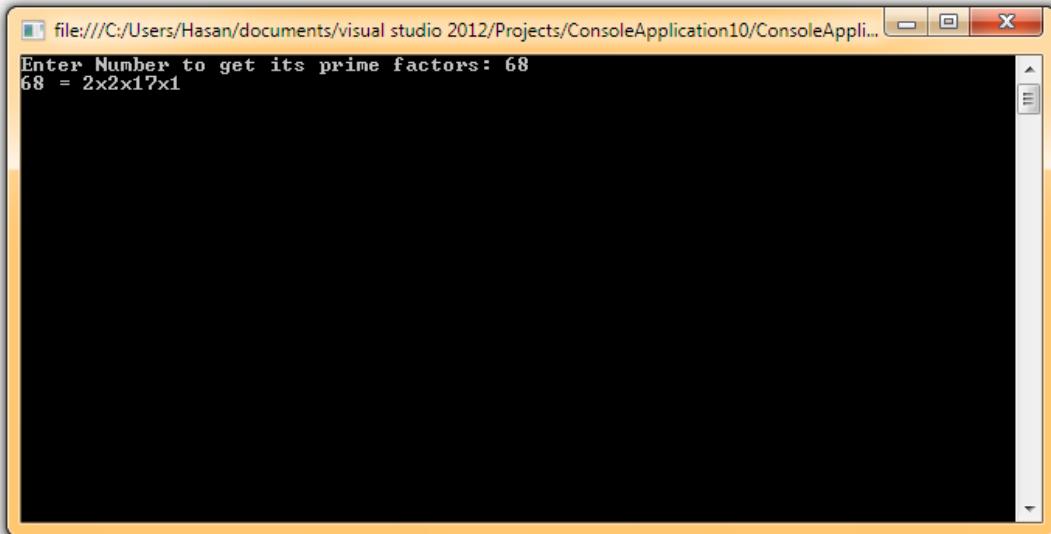
١ العدد الأولي هو العدد الذي لا يقبل القسمة إلا على ١ أو نفسه.



```

        Console.WriteLine("x");
        number /= d;
    }
    d++;
}
Console.WriteLine(1);
Console.ReadKey();

```



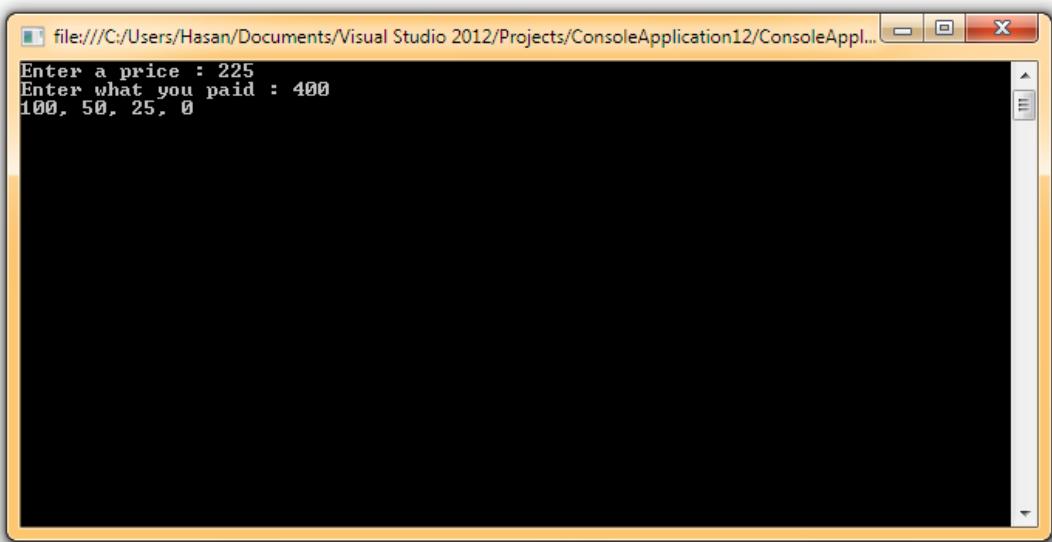
معنى البرنامج السابق هو: لدينا عددين أحدهما العامل الأولي 2، بعد إدخال العدد المراد تحليله إلى جداء عوامل أولية وطالما أن العدد موجب وطالما أنه لا يوجد باقي قسمة عند قسمة العدد على العدد الأولي فقسمه عليه، عندما لا يتحقق الشرط الثاني زد العدد الأولي بمقدار واحد وحرب من جديد، عندما لا يتحقق الشرط الأول والخروج من الحلقة اكتب العدد 1 (وذلك حتى لا يبقى آخر رمز ضمن الصيغة هي x جداء، جرب عدم استخدام 1).

بإمكانك تطوير البرنامج السابق ليشتمل على عنوان وإمكانية تكرار الإدخال، والتأكد من صلاحية القيم. سبق وتم التنوية على أن هذا الجزء من الكتاب ما هي إلا أفكار يمكن الاستفادة منها والاعتناء بشكلها في الجزء الثاني الخاص بالنواذ.

4- برنامج يستعمل في المولات والأمور التجارية، يستقبل ثمن السلعة، ثم المقدار المدفوع، ثم يعيد لك العملات النقدية القياسية التي يجب أن تعيدها للزبون في حال كان ما دفعه أكبر من ثمن القطعة.



```
Console.Write("Enter a price : ");
int price = Convert.ToInt32(Console.ReadLine());
Console.Write("Enter what you paid : ");
int paid = Convert.ToInt32(Console.ReadLine());
int change= paid - price;
while (change >= 1000)
{
    Console.Write ("1000, "); change -= 1000;
}
while (change >= 500)
{
    Console.Write("500, "); change -= 500;
}
while (change >= 200)
{
    Console.Write("200, "); change -= 200;
}
while (change >= 100)
{
    Console.Write("100, "); change -= 100;
}
while (change >= 50)
{
    Console.Write("50, "); change -= 50;
}
while (change >= 25)
{
    Console.Write("25, "); change -= 25;
}
while (change >= 10)
{
    Console.Write("10, "); change -= 10;
}
while (change >= 5)
{
    Console.Write("5, "); change -= 5;
}
while (change >= 2)
{
    Console.Write("2, "); change -= 2;
}
while (change >= 1)
{
    Console.Write("1, "); change -= 1;
}
Console.Write("0");
Console.ReadKey();
```



بإمكانك تطبيق هذا المثال بشكل احترافي أكثر فيما لو أمكنت المستخدم من تحديد الفئات النقدية القياسية التي يملكونها. وبإمكانك أيضاً تطوير البرنامج ليعطيك عدد الوحدات النقدية القياسية بدلاً من تكرارها. لاحظ الكود:

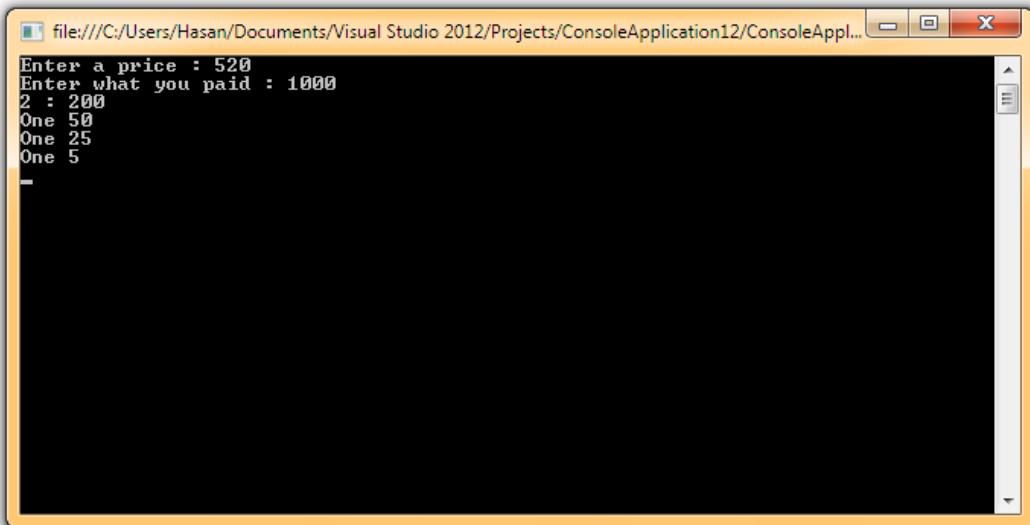
```
Console.WriteLine("Enter a price : ");
int price = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("Enter what you paid : ");
int paid = Convert.ToInt32(Console.ReadLine());
int change = paid - price;
int c1000, c500, c200, c100, c50, c25, c10, c5;
c1000 = c500 = c200 = c100 = c50 = c25 = c10 = c5 = 0;
while (change >= 1000)
{
    c1000 += 1; change -= 1000;
}
while (change >= 500)
{
    c500 += 1; change -= 500;
}
while (change >= 200)
{
    c200 += 1; change -= 200;
}
while (change >= 100)
{
    c100 += 1; change -= 100;
}
while (change >= 50)
{
    c50 += 1; change -= 50;
}
while (change >= 25)
{
    c25 += 1; change -= 25;
}
while (change >= 10)
{
    c10 += 1; change -= 10;
}
while (change >= 5)
{
    c5 += 1; change -= 5;
}
if (c1000 == 1) Console.WriteLine("One 1000");
else if (c1000 > 1) Console.WriteLine(c1000 + " 1000");
```



```

if (c500 == 1) Console.WriteLine("One 500");
else if (c500 > 1) Console.WriteLine(c500 + " : 500");
if (c200 == 1) Console.WriteLine("One 200");
else if (c200 > 1) Console.WriteLine(c200 + " : 200");
if (c100 == 1) Console.WriteLine("One 100");
else if (c100 > 1) Console.WriteLine(c100 + " : 100");
if (c50 == 1) Console.WriteLine("One 50");
else if (c50 > 1) Console.WriteLine(c50 + " : 50");
if (c25 == 1) Console.WriteLine("One 25");
else if (c25 > 1) Console.WriteLine(c25 + " : 25");
if (c10 == 1) Console.WriteLine("One 10");
else if (c10 > 1) Console.WriteLine(c10 + " : 10");
if (c5 == 1) Console.WriteLine("One 5");
else if (c5 > 1) Console.WriteLine(c5 + " : 5");
Console.ReadKey();

```



يمكن تنظيم المتغيرات بشكل أفضل وذلك باستخدام مفهوم المصفوفات وأمثالها من أساليب تجميع البيانات، الفصل الرابع يشرح لك ذلك.

5- برنامج لحساب عدد الكلمات في ملف نصي.

```

//using System.IO;
//طبعاً لازم يكون الملف موجود
StreamReader file = File.OpenText("E:\\MyText.txt");
string line;
int count = 0;
do
{
    line = file.ReadLine();

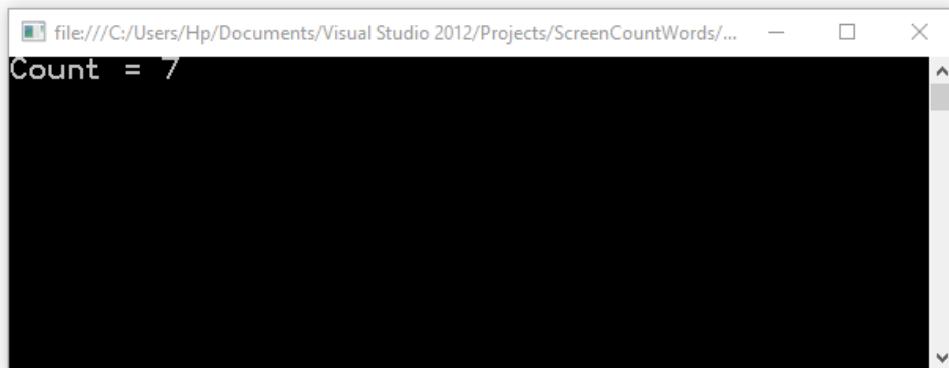
```



```

if (line != null)
{
    string[] words = line.Split(' ');
    count += words.Length;
}
} while (line != null);
file.Close();
Console.WriteLine("Count = {0}", count);
Console.ReadKey();

```



من الممكن الاستفادة من هذا البرنامج في إنشاء قواميس مفردات يستخدمها البرنامج كمصدر له، كبرامج سنراها لاحقا في الجزء الثاني من هذا الكتاب والتي تستخدم للتواصل مع الكمبيوتر أو التي فيها أدوات تقوم بعملية الإكمال التلقائي للنصوص (مثل صندوق بحث غوغل، تكتب كلمة أو جملة فيكملها عنك بشحمة ولحمة¹ قبل أن تنتهي أنت من كتابتها).

6- برنامج لحساب عدد مرات تكرار حرف ما ضمن ملف نصي:

```

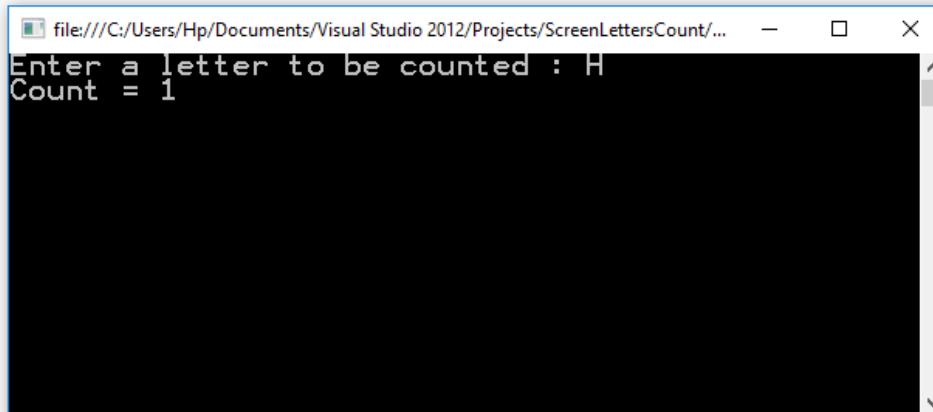
//using System.IO;
//نفس المثال السابق، لازم يكون الملف موجود
StreamReader file = File.OpenText("E:\\\\MyText.txt");
Console.Write("Enter a letter to be counted : ");
string letter = Console.ReadLine();
string line;
int count = 0;
do
{
    line = file.ReadLine();
    if (line != null)
    {
        for (int i = 0; i < line.Length; i++)

```

¹ بشحمة ولحمة: كاملة دون زيادة أو نقصان.

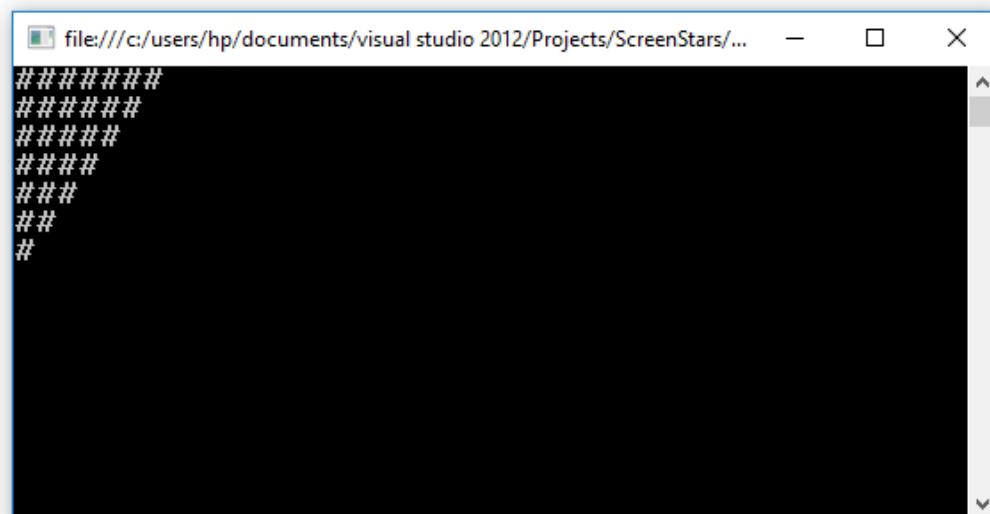


```
if (line.Substring(i, 1) == letter)
    count++;
}
} while (line != null);
file.Close(); Console.WriteLine("Count = {0}", count); Console.ReadKey();
```



7- برنامج لطباعة مجموعة من المحارف على شكل مثلث.

```
int width, height;
char n;
width = height = 7;
n = '#';
for (int row = 0; row < height; row++)
{
    for (int column = 0; column < width; column++)
        Console.Write(n.ToString());
    Console.WriteLine();
    width--;
}
Console.ReadKey();
```





```

int width, height; char n;
width = 0; height = 7; n = 'p';
int max = height;
for (int row = 0; row < height; row++)
{
    for (int column = 0; column < width; column++)
        Console.Write(" ");
    for (int i = 0; i < max; i++)
        Console.Write(n.ToString());
    Console.WriteLine(); width++; max--;
}
Console.ReadKey();

```

من المحتمل وإذا لم يخب ظني – وإذا كنت تتأمل وتراجع هذا المثال أكثر من مرة – فإنك طالب جامعي طلب منه كتابة برنامج يطبع مجموعة من النجوم وفق شكل ما باستخدام الحلقات.

لاحظ هذا الكود:

```

int n;
Console.Write("n = ");
n = int.Parse(Console.ReadLine());
Console.WriteLine();
if (n % 2 == 1 && n < 20 && n > 0)
{
    for (int i = 1; i <= n; i += 2)
    {
        for (int j = i; j <= n; j += 2)
            Console.Write(" ");
        for (int j = i; j > 0; --j)
            Console.Write("*");
    }
}

```



```
Console.WriteLine();
}
for (int i = n - 2; i >= 1; i -= 2)
{
    for (int j = i; j <= n; j += 2)
        Console.Write(" ");
    for (int j = i; j > 0; --j)
        Console.Write("*");
    Console.WriteLine();
}
else
    Console.WriteLine("Only odd numbers between 0 and 20 can be used");
Console.ReadKey();
```

```
n = 15
*
 ***
 *****
 **** ***
 **** *
 **** ***
 **** ***
 **** *
 **** ***
 **** *
 ****
 ***
 *
 *
```

```
n = 18
Only odd numbers between 0 and 20 can be used
```

الشرط (ما بين 0 و 20 وأن يكون فردياً يمكن تغييره، هو فقط مثال).



8- برنامج لطباعة مجموعة من المحارف.

```
for (char i = 'H'; i <= 'N'; i++)
    Console.Write(i + " ");
Console.ReadKey();
```

9- برنامج لحساب العامل لعدد موجب:

```
Console.WriteLine("x = ");
int x = int.Parse(Console.ReadLine());
if (x < 0)
    Console.WriteLine("x >= 0 Only!");
else if (x == 0)
    Console.WriteLine("0! = 1");
else
{
    int Fact = 1;
    for (int i = 1; i <= x; ++i)
        Fact *= i;
    Console.WriteLine("{0}! = {1}", x, Fact);
}
Console.ReadKey();
```

10- يمكن حساب قيمة تقريرية للتكامل المحدود للتتابع وفق علاقه تسمى العلاقة العامة لأشباه المنحرفات، والتي تعطى بالصيغة التالية:

$$\int_a^b f(x)dx \approx \frac{h}{2}(y_0 + 2y_1 + \dots + 2y_{n-1} + y_n)$$

حيث $h = \frac{b-a}{n}$ هي دقة التجزيء.

البرنامج التالي يقوم بحساب قيمة تقريرية للتكامل التالي:

$$\int_0^1 \frac{dx}{1+x^2}$$

.($h = 0.25$ أي $n = 4$)

```
int n = 4;
double [] x = new double[n+1];
double a, b;
a=1; b=0;
double h = (a-b)/n;
int i=0;
```



```
double d = 0;
do {
    x[i] = d;
    d+=h;
    ++i;
} while (d <= a);
double L=0;
for (i = 0; i<=n;++i)
{
    if (i == 0 | i == n)
        L += 1/(1+x[i]*x[i]);
    else
        L += 2*1/(1+x[i]*x[i]);
}
L *= h / 2;
Console.WriteLine("L = " + L);
Console.ReadKey();
```

والنتيجة ستكون $L = 0.78279$ في حين أن القيمة الدقيقة هي 0.7854 (في علم الرياضيات التطبيقية - وهو فرع من الرياضيات يهتم بالحلول العددية التقريرية - يمكن أن يكون هناك هامش خطأ بين القيمة الحقيقية والقيمة المحسوبة بالطرق التقريرية بهامش خطأ ع يتعلق بالمسألة المدروسة وأهميتها).





الفصل الرابع – المصفوفات

قد تتعامل في بعض الأحيان مع مجموعة من المتغيرات من نفس النوع، بحيث لا تختلف عن بعضها إلا بالقيمة، أو قد تحتاج لجمع مجموعة من القيم أو المتغيرات ضمن مجموعة معينة، أو حالات ودواع أخرى كثيرة، مهما كانت الحالة أو الغاية التي تمر بها فمن الأفضل والأُناسب والأُرتب والأبسط استخدام ما يسمى بالمصفوفات.

المصفوفة هي عبارة عن مجموعة من المتغيرات، تشتراك فيما بينها بالنوع وتختلف بالقيمة. من الممكن أن تكون المصفوفات أحادية البعد أو ثنائية الأبعاد أو ثلاثية الأبعاد أو ...، لكن كلما زاد عدد أبعادها أصبحت أكثر للتخيل والتعامل.

التصريح عن المصفوفات

الصيغة التالية تستخدم لتعريف المصفوفات أحادية البعد، عدد عناصرها n :

نوع $new [n];$ اسم المصفوفة [] نوع

ملاحظة

- أول عنصر في المصفوفات ترتيبه 0، وهذا شائع في جميع لغات البرمجة، هذا يعني أن آخر عنصر ترتيبه أقل من عدد عناصر المصفوفة بمقدار واحد.
- من الممكن في بعض لغات البرمجة تغيير ترتيب أول عنصر بجعله 1 بدلاً من 0.



إسناد القيم لعناصر المصفوفات

لا تختلف كثيراً طريقة إسناد القيم لعناصر المصفوفات عن المتغيرات العادية، اللهم إلا بفروقات بسيطة كترتيب العنصر في المصفوفة.

```
string[] day = new string[7];
day[0] = "Saturday";
day[1] = "Sunday";
day[2] = "Monday";
day[3] = "Tuesday";
day[4] = "Wednesday";
day[5] = "Thursday";
day[6] = "Friday";
Console.WriteLine("Enter Number of day [1-7]: ");
int a = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("you choosed " + day[a - 1]);
Console.ReadKey();
```

لاحظ - عزيزي القارئ - أنني صرحت عن مصفوفة عدد عناصرها 7 من نوع نصي، اسمها `day`، عند التعامل مع عناصر المصفوفة تعتبر أول عنصر ترتيبه 0، وعندما تتعامل مع المستخدم - الذي لا يفقه شيئاً في البرمجيات وترتيب العناصر والمصفوفات - فإنك ستناقشه على أساس المنطق الواقعي المتمثل بأن أول عنصر ترتيبه 1. حاول التفريق بين المستخدم وبين الكمبيوتر، بين أكواذك وبين القيم التي يعطيك إياها المستخدم..

في حال كانت عناصر المصفوفة مكونة من سلسلة معروفة أو ناتجة عن عمليات متلاحقة أو مدخلة بالتتابع من قبل المستخدم فتستخدم عادةً حلقات التكرار.

```
double[] num = new double[10];
for (int i = 1; i <= 10; ++i)
{
    num[i-1] = i * i;
    Console.WriteLine(i + "^2 = " + num[i-1]);
}
Console.ReadKey();
```

لا تنسَ أن أول عنصر ترتيبه 0، وأن المستخدم لن يتعامل معك بالمنطق البرمجي، لا تنسَ هذا أبداً!



التعامل مع المصفوفات

توفر لك C# إمكانيات كبيرة للتعامل مع المصفوفات والتي سأناقش بعضها، وفي سبيل ذلك سأعرّف مصفوفة مكونة من مجموعة من الأعداد وأقوم ببعض العمليات عليها:

```
double[] num = new double[10];
for (int i = 1; i <= 10; ++i) num[i-1] = i;
عملية من العمليات الموجودة بعد هذا الكود، على المصفوفة //
Console.ReadKey();
```

- للحصول على متوسط الأعداد، استخدم التابع `Average`

```
Console.WriteLine("Average = " + num.Average());
```

- لمعرفة إن كانت قيمة ما موجودة ضمن عناصر المصفوفة، استخدم التابع `Contains`، حيث أنه يعيد قيمة `True` إذا كانت النتيجة إيجابية و`False` إذا لم تكن:

```
Console.Write("Enter number to find: ");
double a = Convert.ToDouble(Console.ReadLine());
if (num.Contains(a)) Console.WriteLine(a + " exists!");
else Console.WriteLine(a + " does not exist!");
```

ملاحظة

- في بني الشرط، وعند مقارنة متغيرات منطقية بالقيمة `True`، لا داعي لذكر الشرط كاملاً، بمعنى أن `(a = True)` if `(a)` تكافئ `(a == True)` لأن ناتج عملية المقارنة هو قيمة منطقية، وهذا ما تسعى بني الشرط للتعامل معه.
- في بني الشرط من الممكن وضع تعليمية الشرط أمام جملة الشرط في حال كانت تعليمية واحدة، ويجب وضع أقواس في حال كانت أكثر من تعليمية.

- لمعرفة عدد عناصر مصفوفة استخدم التابع `Count`:

```
Console.WriteLine("Number of items is " + num.Count());
```



- للحصول على أول عنصر ضمن مصفوفة وأخر عنصر استخدم التابعين :Last First

```
Console.WriteLine("First item is " + num.First());
Console.WriteLine("Last item is " + num.Last());
```

- للحصول على طول المصفوفة استخدم الخاصية :Length

```
Console.WriteLine("Last item is " + num.Length);
```

- للحصول على أكبر وأصغر قيمة استخدم التابعين :Min Max

```
Console.WriteLine("Max item is " + num.Max());
Console.WriteLine("Min item is " + num.Min());
```

- للحصول على رتبة المصفوفة (بالنسبة لجامعة الرياضيات 😊) استخدم الخاصية :Rank

```
Console.WriteLine("Rank item is " + num.Rank);
```

- للحصول على مجموع مصفوفة استخدم التابع :Sum

```
Console.WriteLine("Sum item is " + num.Sum());
```

استخدام Array

يعتبر System.Array أحد أصناف – فئات – لغة C#, والذي يحوي توابع وطرق للتعامل مع المصفوفات، لذلك فأعتقد أنه من الأنسب وضع هذه الفقرة كتابع للفقرة السابقة وهذا ما قمت به 😊.

- لمسح عناصر محددة بدءاً من عنصر فلاني وحتى عنصر فلاني آخر من عناصر مصفوفة ما استخدم الإجراء Clear. يجب تحديد اسم المصفوفة، ترتيب عنصر البداية وعنصر النهاية بحيث لا يتجاوز عنصر النهاية عدد عناصر المصفوفة:

```
Array.Clear(num, 2, 3);
```



عند تنفيذ الإجراء السابق فإنه سيتم استبدال ثلاثة عناصر من المصفوفة num بدءاً من العنصر الثاني بالقيمة 0، ولو تم التعامل مع مصفوفة نصية فإنه سيتم مسح العنصر (لن يتم إسناد القيمة 0 له).

- لعكس ترتيب العناصر، استخدم الطريقة Reverse والذي يستخدم مع المصفوفات الأحادية فقط:

```
Array.Reverse (num);
```

- لترتيب العناصر أبجدياً (للعناصر النصية) وتصاعدياً (للعناصر الرياضية)، استخدم الطريقة Sort:

```
Array.Sort (num);
```

الحلقات في المصفوفات

في الفصل الثالثتناولنا بنى تكرار متعددة وأجلنا أحدى البنى وهي بنية foreach، والتي تستخدم مع المصفوفات. الصيغة العامة لها:

```
Foreach (اسم مصفوفة VAR نوع متغير)
{
    C#'s Codes
    .
    .
}
```

VAR هو متغير سيحمل قيم جميع عناصر المصفوفة، بحيث نستخدمه لعمليات مختلفة.

من أجل مثال الفقرة السابقة نفسها، سأظهر جميع محتويات المصفوفة وذلك بالكود التالي:

```
foreach (double n in num)
    Console.WriteLine(n);
```

حيث n متغير من نوع بيانات المصفوفة (من نوع المصفوفة ذاتها)، لتتضح الفكرة عليك أكثر لاحظ المثال التالي:



```
string[] name = new string[5];
name[0] = "Ahmad";
name[1] = "Mohammad";
name[2] = "Mahmoud";
name[3] = "Khaled";
name[4] = "Ali";
foreach (string n in name)
    Console.WriteLine(n); // س يتم طباعة جميع عناصر المصفوفة
Console.ReadKey();
```

اللواچ Lists

لا تختلف اللواچ كثيراً عن المصفوفات، فال مهمة المسندة إليهما ذاتها وقد تم توضيحيها في بداية الفصل.

المصفوفات تتميز بتنوع أبعادها، وهذه الميزة تستعمل مع العمليات المعقدة ويستفاد منها عند برمجة الألعاب والتطبيقات الضخمة، في حين أن اللواچ لا تملك هذه الميزة.

اللواچ هي أبسط شكل للمصفوفات، وتتميز بالдинاميكية والتي تمكّنك من إضافة أو حذف العناصر بسهولة شديدة، وذلك لأن طولها ليس ثابتاً كما هو الحال في المصفوفات، والتي لا يوجد فيها إمكانية إضافة أو حذف عناصر، فعناصر المصفوفة تبقى موجودة خلال فترة استخدامها في برنامجك مع إمكانية تغيير قيمها فقط.

لذلك فاللواچ شائعة أكثر مع البرامج البسيطة، والتي ستراها طيلة حياتك البرمجية 😊.

التصريح عن اللواچ

الصيغة التالية تستخدم لتعريف اللواچ:

```
List <نوع> = اسم اللائحة <نوع>();
```

التعامل مع اللواچ

بشكل أساسي تتميز اللواچ عن المصفوفات بإمكانية الإضافة والحذف، وذلك من خلال الطريقـ Add و Remove و RemoveAt.



- إضافة عنصر -

تختلف اللوائح عن المصفوفات شكلاً بالأقواس، ولا نقوم بإسناد القيم للعناصر وإنما بإضافتها لأنها غير موجودة أصلاً، حيث نضيف العنصر وقيمتها. المثال التالي يوضح عملية التعامل مع ستة مواد دراسية مدخلة مسبقاً، ويناقش البرنامج عدد المواد الراسبة ومعدل المواد معاً:

```
double Sum = 0;
List<int> Marks = new List<int>();
Marks.Add(90);
Marks.Add(80);
Marks.Add(55);
Marks.Add(90);
Marks.Add(40);
Marks.Add(66);
foreach (int m in Marks)
    if (m < 60)
        Sum = Sum + 1;
Console.WriteLine("Average = " + Math.Round(Marks.Average(), 2));
Console.WriteLine("Failed in " + Sum + " Subjects!");
Console.ReadKey();
```

النتيجة بالتأكيد ستكون راسب بمادتين، والمثال سيكون واقعي أكثر فيما لو تم إدخال المواد من قبل المستخدم أو من خلال استيراد القيم من ملف خارجي أو قاعدة بيانات.

- حذف العناصر -

ويكون باستخدام رتبة العنصر أو قيمته، وهذه الميزة غير موجودة بالمصفوفات. لاحظ المثال:

```
List<int> Marks = new List<int>();
Marks.Add(90);
Marks.Add(80);
Marks.Add(55);
Marks.Add(90);
Marks.Add(40);
Marks.Add(66);
Marks.Remove(55);
Marks.RemoveAt(2);
foreach (int m in Marks)
    Console.WriteLine(m);
Console.ReadKey();
```



ونتيجة الكود ستكون بالتأكيد حذف العنصر 55 والعنصر 90، لاحظ الجدول التالي:

الرتبة	العناصر الأساسية	عند الحذف الأول	عند الحذف الثاني
0	90	90	90
1	80	80	80
2	55	90	40
3	90	40	66
4	40	66	لا يوجد عنصر
5	66	لا يوجد عنصر	لا يوجد عنصر

ملاحظة

- أول عنصر في اللوائح ترتيبه 0.

Dictionaries القواميس

قد تحتاج في كثير من الأحيان إلى تخزين بيانات مشتركة بالنوع، وفهرستها لتسهيل وتنظيم الوصول إليها. تعطيك المصفوفات واللوائح فهارس تلقائية تتمثل بأعداد صحيحة تبدأ بالصفر، أما القواميس فتعطيك إمكانية فهرسة البيانات وفق أي نوع ترغب به.

التصريح عن القواميس

```
Dictionary<عنصر, مفتاح> new Dictionary<اسم القاموس>;
```

حيث المفتاح هو نوع القيمة التي ستفهرس عناصر القاموس من خلالها ومن الممكن ألا تكون رقمية صحيحة، العنصر وهو نوع العناصر المراد فهرستها.

إضافة عناصر للقواميس

تشابه القواميس اللوائح من حيث أنها ديناميكية وغير محددة الحجم، فكما هو الحال في اللوائح فإن إضافة العناصر للقواميس يتم عن طريق الإجراء :Add

```
Dictionary<int, string> Names = new Dictionary<int, string>();
Names.Add(0, "Ahmad");
Names.Add(1, "Ali");
```



```
Names.Add(2, "Hasan");
Names.Add(3, "Khaled");
Names.Add(4, "Mahmoud");
```

كما يمكن أن يكون المفتاح قيمة نصية مثلا:

```
Dictionary<string, string> Names = new Dictionary<string, string>();
Names.Add("Name1", "Ahmad");
Names.Add("Name2", "Ali");
Names.Add("Name3", "Hasan");
Names.Add("Name4", "Khaled");
Names.Add("Name5", "Mahmoud");
```

كما يمكن غيرها من الأنواع، لكن مع الأسف ستزداد صفحات الكتاب
لتصبح أكثر من 1000 صفحة إذا أردت كتابة مثال عن كل حالة واردة 😊.

طباعة عناصر القواميس

```
Console.WriteLine("NameID\tName");
Console.WriteLine("-----");
foreach (KeyValuePair<string, string> m in Names)
{
    //View all
    Console.WriteLine(m.Key + "\t" + m.Value);
}
```

ملاحظة

- أعلم أن `KeyValuePair<string, string>` لم تعجبك، لذلك سأخبرك أنه يمكنك استبدالها بـ `var`، حيث أن `var` متغير غير محدد. فالمحرر يقرر نوعه طبقاً للعملية الجارية والمتغيرات الممررة معه.. حيث إن الأولى هي تركيب `struct` خاص بالمجموعات.¹

كما يمكن طباعة عنصر بمعرفة ترتيبه ضمن القاموس من خلال التابع `ElementAt` والذي يستقبل متغير رقمي صحيح:

```
Console.WriteLine(Names.ElementAt(2)); //Prints [Name3, Hasan]
```

¹ المجموعات هي عبارة عن المصفوفات أو اللوائح أو .. (إلخ من طرق تجميع البيانات المتشابهة معاً).



البحث عن العناصر في القواميس

بإمكانك البحث عن الفهارس أو العناصر وذلك باستخدام التابعين `ContainsKey` و `ContainsValue`:

```
string key = "Name3", name = "Ali";
//Search key
if (Names.ContainsKey(key)) Console.WriteLine(Names[key] + " Found!");
//Search value
if (Names.ContainsValue(name)) Console.WriteLine(name + " Found!");
```

حذف العناصر في القواميس

لحذف عنصر ما استخدم الإجراء `Remove`، ولحذف جميع العناصر `:Clear`

```
//Delete value
Names.Remove("Name2");
//Delete all values
Names.Clear();
```

المكدسات Stacks

المُكَدِّسات هي نوع من أنواع تجميع البيانات مثل المصفوفات واللوائح والقواميس، إلا أنها تختلف عنها بطريقة الاستخدام أو الفكرة من وجودها.

المصفوفات موجهة للأنظمة المعقدة والمنظمة والمرتبة بوضعية معينة، بينما اللوائح موجهة للحالات التي فيها بيانات يراد تجميعها معًا بغرض القيام بعمليات عليها بحيث تكون أحادية البعد، واللوائح تعامل مع عمليات مثل إضافة عناصر `Add` وحذف عناصر `Remove`. كما أن القواميس تشترك مع اللوائح بالعمليتين السابقتين إلا أنها موجهة للفهرسة أكثر منها لتجمیع البيانات فقط.

المكدسات تختلف عن كل ماضي، فعملياتها الرئيسية هي `Push-On` لتكميل - إضافة - العناصر، و `Pop-Off` لطرح العناصر. كما يمكنك بالتأكيد إجراء عمليات مثل معرفة عدد عناصر المكدس وما إلى ذلك. كما أنها تختلف عن أنواع المجموعات الأخرى بأنها ضمن مجال الأسماء `.System.Collections.Generic` بينما البقية ضمن `.System.Collections`.

عند إضافة العناصر للمكدسات فإن العنصر الجديد سيكون أمام أو أعلى من العنصر القديم، بمعنى أنه بإمكانك الوصول للعناصر الأحدث أولاً. هذا يعني



أن عملية طرح العناصر ستكون من العنصر الأخير إلى العنصر الأول. وهذا ما يسمى بـTechnique LIFO وهي اختصار لـLast Input First Output.

التصريح عن المكدسات

```
Stack Stack = new Stack();
```

إضافة عناصر للمكدسات

يتم إضافة العناصر عن طريق الإجراء Push:

```
تعريف مكدس جديد //  
Stack myStack = new Stack();  
// إضافة عناصر  
myStack.Push("Item1");  
myStack.Push("Item2");  
myStack.Push("Item3");  
myStack.Push("Item4");  
myStack.Push("Item5");
```

بات لدينا في الذاكرة المكدس التالي ذو العناصر الخمسة:

myStack
Item5
Item4
Item3
Item2
Item1

طرح العناصر من المكدسات

طرح العناصر لا يعني التخلص منها وإنما إسنادها إلى متغير آخر. تتم عملية طرح العناصر عن طريق التابع Pop والذي يقوم فعلياً بحذف العنصر من المكدس وإسناده إلى متغير ما. وكما اتفقنا فعملية الطرح تكون وفق العنصر الأخير.

```
string x = myStack.Pop().ToString(); //x = Item5
```

.WriteLine عن طباعة قيمة x عن طريق الإجراء



كما يمكنك طرح العناصر من المكدسات دون حذفها منها عن طريق التابع :Peek

```
x = myStack.Peek().ToString(); //x = Item4
```

كما يمكنك معرفة عدد عناصر المكدس عن طريق الخاصية Count ومسح جميع العناصر عن طريق الطريقة .Clear.

جرب كل ما سبق في مثال واحد:

```
Stack myStack = new Stack();
myStack.Push("Item1");
myStack.Push("Item2");
myStack.Push("Item3");
myStack.Push("Item4");
myStack.Push("Item5");
string x = myStack.Pop().ToString();
Console.WriteLine("Popping off " + x);
Console.WriteLine("myStack has now {0} Item\\Items", myStack.Count);
x = myStack.Pop().ToString();
Console.WriteLine("Popping off " + x);
Console.WriteLine("myStack has now {0} Item\\Items", myStack.Count);
x = myStack.Peek().ToString();
Console.WriteLine("Peeking out " + x);
Console.WriteLine("myStack has now {0} Item\\Items", myStack.Count);
myStack.Clear();
Console.WriteLine("Clearing myStack...");
Console.WriteLine("myStack has now {0} Item\\Items", myStack.Count);
Console.ReadKey();
```

والنتيجة ستكون:

Popping off Item5

myStack has now 4 Item\Items

Popping off Item4

myStack has now 3 Item\Items

Peeking out Item3

myStack has now 3 Item\Items

Clearing myStack...

myStack has now 0 Item\Items

للمكذسات تطبيقات عدّة أهمّها الآلات الحاسبة العلمية والبرامج التي تعامل مع معادلات وكل تطبيق يحوي رموز معينة يقوم الكمبيوتر بتحليلها إلى أصناف وأنواع، ويفرزها بحسب نوعها "رمز"، "حرف"، "رقم"، إلخ..

ختاماً، لا تقتصر عملية تخزين المتغيرات المتشابهة على المصفوفات واللواحق والقواميس، هناك أيضاً Collection و ArrayList و SortedList و الكثير أيضاً، كما أن هناك نوع آخر هو Hashtable، والذي لا يختلف كثيراً عن القواميس.



أمثلة تطبيقية

ملاحظة

- غير مطلوب منك فهم المعادلات الواردة في هذا الفصل - أو بقية الفصول - وإنما عليك فهم كيفية استخدامها وكيفية تكييفها. هناك أناس متخصصون في الرياضيات سيفهمون هذه المعادلات، النقاش والاستنتاجات موجهة لهم.. أنت عليك الأكواذ فقط. توجه مباشرة للكود وخذ النتيجة النهائية من كل استنتاج.

1- برنامج لحل جملة معادلتين بمحضتين!

فكرة الحل:

سبق واتفقنا على أنه لحل أي مسألة في البرمجة يجب تجريد المطلوب والحصول على صيغة نهائية للمتغيرات المطلوبة، بحيث يتم التعويض مباشرة فيها للحصول على النتيجة.

إن الشكل العام لجملة معادلتين بمحضتين هو:

$$\begin{cases} a_1 x + a_2 y = a_3 \\ b_1 x + b_2 y = b_3 \end{cases}$$

حيث x و y محضلان، والبقية ثوابت.. لنأخذ مثالاً ونحله رياضياً لاستنتاج طريقة لأتمتها!

$$\begin{cases} -5x + 2y = -5 \\ 1x + 2y = 13 \end{cases}$$

لحل جملة المعادلتين السابقتين هناك ثلاثة طرق، إحداها طرح المعادلتين من بعضهما.. لاحظ:

أولاً: نضرب المعادلة الأولى بأمثال أول حد من المعادلة الثانية، والثانية بأمثال أول حد من الأولى فنحصل على جملة المعادلتين التالية:

$$\begin{cases} -5x + 2y = -5 \\ -5x + -10y = -65 \end{cases}$$

ثانياً: نطرح الأولى من الثانية، فنحصل على المعادلة التالية:

$$0 + 12y = 60 \rightarrow y = \frac{60}{12} = 5$$



ثالثاً: نعوض في إحدى المعادلتين من أي جملة لدينا، لنحصل على x بعد عزلها:

$$x = \frac{-5 - 2y}{-5} = \frac{-5 - 2 \cdot 5}{-5} = 3$$

وهكذا أوجدنا حل جملة المعادلتين، الآن سنتنقل من التعبير الرياضية إلى تلك البرمجية، وللسهولة سنشوّل ذلك مرحلة..

أولاً: كما لاحظت فإن المتغير في المسألة هو الأمثل، لذلك سننشئ مصفوفة لأمثال السطر الأول وأخرى لأمثال السطر الثاني. على اعتبار أن السطر الأول يحوي أمثال L_x وآخر L_y ، وقيمة ثلاثة لمابعد المساواة، فإن المصفوفة يجب أن تحوي ثلاثة عناصر.. سنسمى الأولى A والثانية B تيمنا بالصيغة العامة..

الآن المصفوفتين هما:

$$A = [a_1, a_2, a_3]$$

$$B = [b_1, b_2, b_3]$$

لضرب المصفوفتين بأمثال بعض، سننشئ مصفوفتين تمثلانهما بعد الجداء، وسنجعل اسميهما المصفوفتين الجديدين – المكافئتين – يشبهان المصفوفات الأساسية:

$$AA = A \times b_1 = [a_1, a_2, a_3] \times b_1$$

$$BB = B \times a_1 = [b_1, b_2, b_3] \times a_1$$

ثانياً: لنطرح المصفوفتين من بعضهما ونشئ مصفوفة نضع فيها نتيجة الطرح:

$$\text{Result} = BB - AA$$

وعلى اعتبار أن أول عنصر من AA نفسه أول عنصر من BB فأمثال أول مجهول ستكون معدومة.. (أول عنصر من كلا المصفوفتين هو $a_1 \times b_1$).

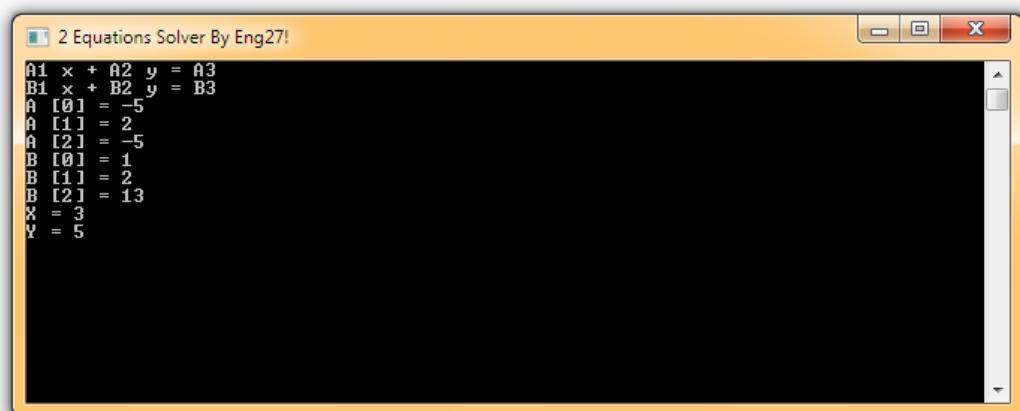
بناءً عليه يمكن حساب المتغير الثاني وذلك بعد قسمة ما بعد المساواة على أمثاله..

ثالثاً: نعزل من A أو B أو AA أو BB علاقة L_x ونحوذ فيها قيمة y .



بعد المناقشة السابقة أصبح لدينا البرنامج التالي:

```
Console.Title = "2 Equations Solver By Eng27!";
double [] a = new double[3]; // لتخزين أمثال السطر الأول
double [] b = new double[3]; // لتخزين أمثال السطر الثاني
double [] aa = new double[3]; // لتخزين السطر الأول بعد التعديل عليه
double [] bb= new double[3]; // لتخزين السطر الثاني بعد التعديل عليه
double [] result = new double [3];
Console.WriteLine("A1 x + A2 y = A3");
Console.WriteLine("B1 x + B2 y = B3");
for (int i = 0; i<3;++i) // لإدخال السطر الأول
{
    Console.Write ("A [" + i + "] = ");
    a[i] = Convert.ToDouble (Console.ReadLine());
}
for (int i = 0; i < 3; ++i) // لإدخال السطر الثاني
{
    Console.Write("B [" + i + "] = ");
    b[i] = Convert.ToDouble(Console.ReadLine());
}
for (int i = 0; i < 3; ++i) // لحساب الفرق بين السطرين بعد توحيدهما
{
    aa[i] = b[0] * a[i];
    bb[i] = a[0] * b[i];
    result[i] = bb[i] - aa[i];
}
double X, Y;
Y = result[2] / result[1]; // لاحظ المثال المحلول رياضيًّا
X = (aa[2] - aa[1] * Y) / aa[0]; // نفس الملاحظة السابقة
Console.WriteLine("X = " + Math.Round(X, 3));
Console.WriteLine("Y = " + Math.Round(Y, 3));
Console.ReadKey();
```





ملاحظة

- كان من الممكن أن يكون البرنامج السابق أفضل في حال عرضت المعادلتين بعد إدخال أمثلهما (كل من A وB). على كل حال هكذا برامج وأمثلة تطبيقية يستفاد منها في النوافذ أكثر من هنا..

2- برنامج لحل جملة ثلاثة معادلات بثلاث مجاهيل!!!

لا أظن أن هناك حاجة لشرح الطريقة.. تفك بالكود، تذوقه ..

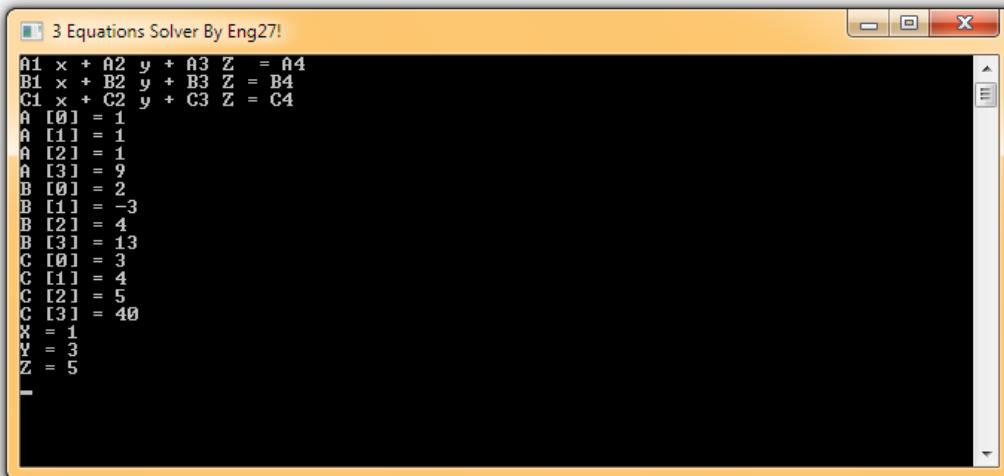
```
Console.Title = "3 Equations Solver By Eng27!";
تعريف المصفوفات التي ستتحوي قيم الأمثل في الأسطر الثلاثة//
double [] a = new double[4];
double [] b = new double[4];
double [] c = new double[4];
تعريف المصفوفات التي ستتحوي الأسطر بعد عزلها//
double [] aa = new double[3];
double [] bb = new double[3];
double [] cc = new double[3];
double [] result = new double [3];
إظهار شكل جملة المعادلات//
Console.WriteLine("A1 x + A2 y + A3 Z = A4");
Console.WriteLine("B1 x + B2 y + B3 Z = B4");
Console.WriteLine("C1 x + C2 y + C3 Z = C4");
إدخال الثوابت، كل سطر بسطره//
for (int i = 0; i<4; ++i)
{
    Console.Write ("A [" + i + "] = ");
    a[i] = Convert.ToDouble (Console.ReadLine());
}
for (int i = 0; i < 4; ++i)
{
    Console.Write("B [" + i + "] = ");
    b[i] = Convert.ToDouble(Console.ReadLine());
}
for (int i = 0; i < 4; ++i)
{
    Console.Write("C [" + i + "] = ");
    c[i] = Convert.ToDouble(Console.ReadLine());
}
عزل المعادلات//
aa[0] = -a[1]/a[0];
aa[1] = -a[2]/a[0];
aa[2] = a[3] / a[0];
bb[0] = b[1] + b[0] * aa[0];
bb[1] = b[2] + b[0] * aa[1];
```



```

bb[2] = b[3] - b[0] * aa[2];
cc[0] = c[1] + c[0] * aa[0];
cc[1] = c[2] + c[0] * aa[1];
cc[2] = c[3] - c[0] * aa[2];
// سنعرف متغيران بدل أن نقوم بتعريف مصفوفتان إضافيتان
// مع العلم أنه كان بالإمكان التعامل مع حالة جملة معادلتان هكذا..
double l = bb[0], m= cc[0];
for (int i = 0; i < 3; ++i)
{
    cc[i] = l * cc[i];
    bb[i] = m * bb[i];
    result[i] = bb[i] - cc[i];
}
double X, Y, Z;
Z = result[2] / result[1];
Y = (bb[2] - bb[1] * Z) / bb[0];
X = (-a[1] * Y - a[2] * Z + a[3])/a[0];
Console.WriteLine("X = " + Math.Round(X, 3));
Console.WriteLine("Y = " + Math.Round(Y, 3));
Console.WriteLine("Z = " + Math.Round(Z, 3));
Console.ReadKey();

```



3- يمكن إيجاد تابع رياضي ممثّل لعملية ما من خلال كثير حدود¹ الاستيفاء الداخلي للاغرانيج - وهو أحد الأشياء القلائل التي استفادتُها من الجامعة، الحقُّ يقال - من خلال العلاقة التالية:

$$L_n(x) = \sum_{i=1}^n y_i \frac{(x - x_1) \cdots (x - x_{i-1}) \cdot (x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_1) \cdots (x_i - x_{i-1}) \cdot (x_i - x_{i+1}) \cdots (x_i - x_n)}$$

¹ كثير الحدود هو تركيب جبري يتكون من واحد أو أكثر من المعاملات والمتغيرات، يتم بناؤه باستخدام عمليات الجمع والطرح والضرب والأسس الصحيحة غيرالسالبة. (المصدر: ويكيبيديا - متعدد الحدود).



حيث:

- x مجهول، ور تابع له.. والعملية المدروسة هي عملية دخلها المجهول وخرجها التابع.
- L_n التابع الرياضي الذي يعبر عن العملية المدروسة، وفق كثير حدود الاستيفاء الداخلي للاغرانيج.
- n عدد الحالات المعروفة خرجها ودخلها.

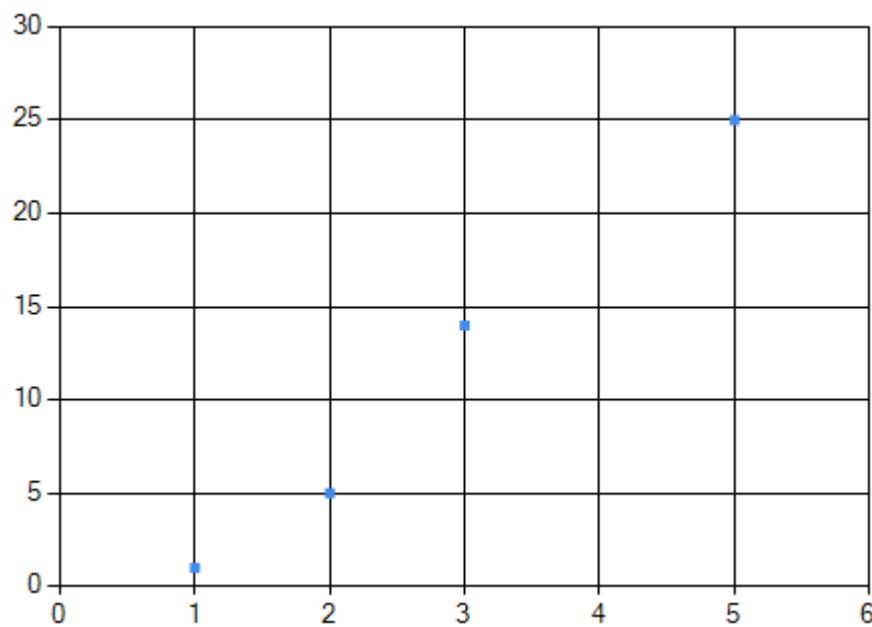
سأبسط لك الموضوع أكثر.. لدينا مسألة نعلم فيها نقاط فقط، خذ المسألة التالية:

سجلت شركة أرباحاً مختلفة خلال سنة، حيث تمأخذ القراءات كل أربعة أشهر، وكانت الأرباح تتبع لمقدار الرأس مال الذي تضعه الشركة وفق الجدول التالي:

الفترة	3-1	6-4	9-7	12-10
رأس المال	1	2	3	5
الأرباح	1	5	14	25

حيث أن كل من رأس المال والأرباح مقدرة بـمليون ليرة سورية.

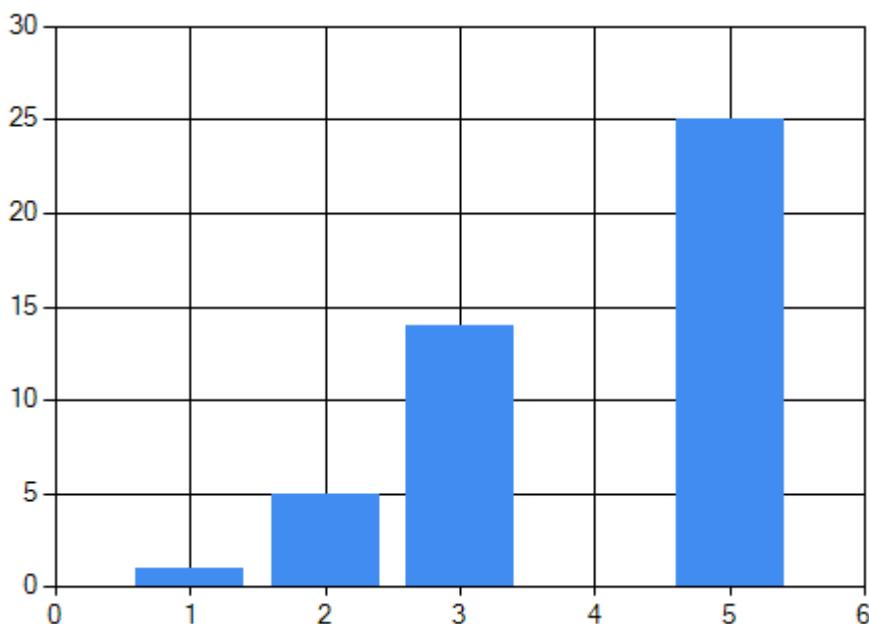
لنقم بتعيين النقاط على جملة محاور إحداثية كما يفعل الاقتصاديون:





والذي نريده هو رسم منحنى بياني تقريري يمر من كافة النقاط السابقة ومعرفة تابعه الرياضي، بحيث يمكن معرفة قيمة الأرباح (المحور الشاقولي) بمجرد معرفة أو توقيع قيمة رأس المال (المحور الأفقي)، كما أن التابع الرياضي يعطي توقعات أدق من الرسم البياني. وإحدى طرق معرفة التابع الرياضي هي طريقة الاستيفاء الداخلي للأغراض، والتي تعتمد على نقاط يمر منها التابع الرياضي (جذور التابع الرياضي)، وبكلمات أخرى: النقاط التي تحقق معادلة التابع الرياضي.

عادةً ما تشاهد مخططات على شكل أعمدة:



المخططات على شكل أعمدة لا تعطيك مجالاً لتوقع نقاط أخرى وإنما تسمح لك بمقارنة النقاط المعلومة فقط.

المطلوب منك - عزيزي القارئ - أن تتوقع الأرباح من أجل رأس مال مختلف عما هو موجود ضمن المسألة، ورياضياً يمكن تمثيل أي عملية بتتابع كثير حدود بحيث أنه يمكن توقيع أي نتيجة للخرج بمعرفة قيمة ما للدخل.

طيب، ما علاقة كثير حدود لاغرائج بالمصفوفات؟؟؟ لاحظ أن المسألة تحوي مجموعة من القيم تمثل الدخل x ومجموعة من القيم تمثل الخرج y ، وفي حالة المسائل التي تحوي مجموعات من القيم المتشابهة - بالنوع - نستخدم المصفوفات. وعادةً ماتكون قوانين المسائل التي تحوي مجموعات من القيم تحمل إشارة المجموع Σ .

لأنأخذ أمثلة رياضية لنفهم الفكرة أكثر.. وسنبدأ بالمثال السابق:



بما أن الجدول يحوي أربعة قيم فهذا يعني أن تابع لاغرانج له الشكل:

$$L_4(x) = \sum_{i=1}^4 y_i \frac{(x - x_0) \cdots (x - x_{i-1}) \cdot (x - x_{i+1}) \cdots (x - x_3)}{(x_i - x_0) \cdots (x_i - x_{i-1}) \cdot (x_i - x_{i+1}) \cdots (x_i - x_3)}$$

5	3	2	1	x
25	14	5	1	y

وبالتالي:

$$\begin{aligned} L_4(x) &= 1 \frac{(x-2)(x-3)(x-5)}{(1-2)(1-3)(1-5)} + 5 \frac{(x-1)(x-3)(x-5)}{(2-1)(2-3)(2-5)} \\ &\quad + 14 \frac{(x-1)(x-2)(x-5)}{(3-1)(3-2)(3-5)} + 25 \frac{(x-1)(x-2)(x-3)}{(5-1)(5-2)(5-3)} \end{aligned}$$

وبنشر كل كثير حدود على حدة نجد:

$$\begin{aligned} (x-2)(x-3)(x-5) &= (x^2 - 5x + 6)(x-5) \\ &= x^3 - 5x^2 - 5x^2 + 25x + 6x - 30 \\ &= x^3 - 10x^2 + 31x - 30 \end{aligned}$$

$$\begin{aligned} (x-1)(x-3)(x-5) &= (x^2 - 4x + 3)(x-5) \\ &= x^3 - 5x^2 - 4x^2 + 20x + 3x - 15 \\ &= x^3 - 9x^2 + 23x - 15 \end{aligned}$$

$$\begin{aligned} (x-1)(x-2)(x-5) &= (x^2 - 3x + 2)(x-5) \\ &= x^3 - 5x^2 - 3x^2 + 15x + 2x - 10 \\ &= x^3 - 8x^2 + 17x - 10 \end{aligned}$$

$$\begin{aligned} (x-1)(x-2)(x-3) &= (x^2 - 3x + 2)(x-3) \\ &= x^3 - 3x^2 - 3x^2 + 9x + 2x - 6 = x^3 - 6x^2 + 11x - 6 \end{aligned}$$

وبالتالي:

$$\begin{aligned} L_4(x) &= 1 \frac{x^3 - 10x^2 + 31x - 30}{-1 \cdot -2 \cdot -4} + 5 \frac{x^3 - 9x^2 + 23x - 15}{1 \cdot -1 \cdot -3} \\ &\quad + 14 \frac{x^3 - 8x^2 + 17x - 10}{2 \cdot 1 \cdot -2} + 25 \frac{x^3 - 6x^2 + 11x - 6}{4 \cdot 3 \cdot 2} \end{aligned}$$

$$L_4(x) = -0.92x^3 + 8x^2 - 13.6x + 7.5$$



للتأكد من صحة الحل، جرب عوض قيم x لتحصل على y ، وبالتالي ستكون قيمة تقريبية منها لأن التابع هوتابع تقريري.

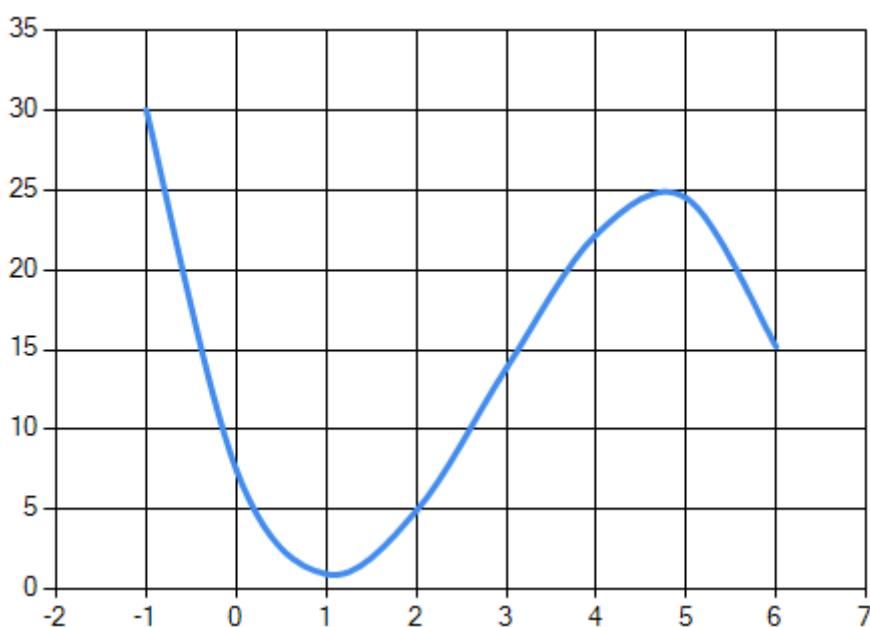
الآن لنتوقع أرباحاً معينة لرأس مال 4 و 4.5 و 6 مثلاً:

$$L_4(4) = 22.22$$

$$L_4(4.5) = 24.67$$

$$L_4(6) = 16.2$$

وبرسم المحنبي البياني المعبر عن المعادلة السابقة:



لكن لحظة، كيف سنقوم بهذا بالبرمجة؟! الصفحات السابقة ليست إلا مقدمة، الآن بدأ العمل.. وأعيid تذكيرك إذا لم تكن مختصاً بالرياضيات أو من محبيها فلا تعذّب نفسك بقراءة المناقشة وكيفية الاستنتاج، لا أريد أن تخصص وقتاً من أدعيتك للدعاء على

أما إذا كانت الاستنتاجات تروق لك، فأحضر ورقة وقلماً، وتحلّ ببعض الصبر، وفكّر معي:

في البداية لنستنتج ما هو ناتج نشر معادلة من الدرجة الثالثة (أي معلوم منها ثلاثة نقاط)، وهي من الشكل: $(x - a)(x - b)(x - c)$

أنشر الأقواس لتحصل على مايلي:



$$\begin{aligned}
 (x - a)(x - b)(x - c) &= (x^2 + (-a - b)x + ab)(x - c) \\
 &= x^3 - cx^2 + (-a - b)x^2 - (-a - b)cx + abx - abc \\
 &= x^3 + (-a - b - c)x^2 + ((a + b)c + ab)x - abc
 \end{aligned}$$

ألف مبروك لقد استنتجت منشور معادلة من الدرجة الثالثة بمعرفة ثلاثة جذور منها، بغض النظر ما إذا كان هناك من سبقك في استنتاجه أو سجله تحت اسمه، انتقل للفيسبوك مباشرًة وأنشئ صفحة باسم الباحث والمفكر في الرياضيات وابداً بنشر أفكارك ونظرياتك..

وبالعودة لصيغة لاغرانج، فإنك تحتاج لمصفوفة x للمجاھيل، و L للتواع، ولـ L كثير حدود لاغرانج نضع فيها ثوابت المعادلة وبالتأكيد فإنك تحتاج لمصفوفة تضع فيها نتيجة كثيرات الحدود الجزئية.

```

int n = 4;
int[] x = new int[n];
int[] y = new int[n];
x[0] = 1; x[1] = 2; x[2] = 3; x[3] = 5;
y[0] = 1; y[1] = 5; y[2] = 14; y[3] = 25;
double[] L = new double[n]; //أمثال كثير حدود لاغرانج
double[,] L1 = new double[n, n]; //أمثال المعادلات الجزئية
double[,] L2 = new double[n]; //مقامات المعادلات الجزئية
double a = 0, b = 0, c = 0;
for (int i = 0; i < n; ++i)
{
    L2[i] = 1;
    for (int j = 0; j < n; ++j)
    {
        حساب المقام //تحديد البسط
        if (x[i] != x[j])
            L2[i] *= (x[i] - x[j]);
        else
        {
            if (j == 0)
                { a = x[1]; b = x[2]; c = x[3]; }
            else if (j == 1)
                { a = x[0]; b = x[2]; c = x[3]; }
            else if (j == 2)
                { a = x[0]; b = x[1]; c = x[3]; }
            else
                { a = x[0]; b = x[1]; c = x[2]; }
            L1[i, 0] = 1;
            L1[i, 1] = -a - b - c;
            L1[i, 2] = (a + b) * c + a * b;
            L1[i, 3] = -a * b * c;
        }
    }
}

```



```

    }
}
for (int j = 0; j < n; ++j)
    for (int i = 0; i < n; ++i)
        L[j] += y[i] * L1[i, j] / L2[i];
Console.WriteLine("L(x) = L1 x^3 + L2 x^2 + L3 x + L4");
int l = 0;
foreach (double d in L)
    Console.WriteLine("L{0} = {1}", ++l, Math.Round(d, 1));
Console.ReadKey();

```



وفي حالة المسائل التي علم فيها خمسة نقاط، فإن نشر المعادلة يكون بالشكل:

$$\begin{aligned}
 (x - a)(x - b)(x - c)(x - d) \\
 &= (x^2 + (-a - b)x + ab)(x - c)(x - d) \\
 &= (x^3 - cx^2 + (-a - b)x^2 - (-a - b)cx + abx - abc)(x \\
 &\quad - d) \\
 &= x^4 + (-a - b - c - d)x^3 \\
 &\quad + ((a + b)c + ab + (a + b + c)d)x^2 + (((a + b)c \\
 &\quad + ab)d - abd)x + abcd
 \end{aligned}$$

4- يمكن إيجاد المماس للأي تابع بمعرفة نقطة التماس بالعلاقة التالية:

$$y = m(x - x_1) + y_1$$

حيث m ميل هذا المستقيم، والذي يساوي إلى مشتق التابع المراد حساب المماس له. أي أن: $m = y'$



البرنامج التالي يقوم باستيقاظ تابع صحيح معطى من قبل المستخدم، ثم يقوم بإيجاد المماس بعد أن يقوم المستخدم بإدخال نقطة التماس.

أي أن خوارزمية البرنامج ستكون:

- أن يدخل المستخدم درجة تابع كثير الحدود والذي سينوجد مماساً له في نقطة تماس يختارها المستخدم.
 - أن يدخل المستخدم ثوابت كثير الحدود، ويحفظها ضمن مصفوفة.
- فإذا كانت درجة كثير الحدود 3 مثلا، احتجنا أربعة ثوابت:

$$f(x) = ax^3 + bx^2 + cx + d$$

- أن يدخل المستخدم نقطة التماس.
- أن يحدد البرنامج معادلة المشتق وذلك بضرب كل ثابت بأس المجهول الذي يجاوره، ومن أجل معادلة من الدرجة الثالثة يكون المشتق:

$$f'(x) = 3ax^2 + 2bx + 1c + 0$$

بمعنى أن ثوابت المشتق هي نفسها ثوابت التابع مضروبة بأرقام تبدأ بدرجة التابع، وآخر عنصر فيها 0 لأن آخر حد في التابع ثابت.

- أن يحسب البرنامج قيمة المشتق عند نقطة التماس.
- إذا أدخل المستخدم إحداثي x فقط يقوم البرنامج بحساب الإحداثي الآخر من معادلة التابع. ولكن سأترك هذه الخطوة لك مع أنها ليست صعبة.

- أن يظهر البرنامج معادلة المماس بعد أن يختزلها لتصبح بالشكل:

$$y = mx + c$$

حيث $m = y'$ و $c = -mx_1 + y_1$ بناءً على ما سبق.

```

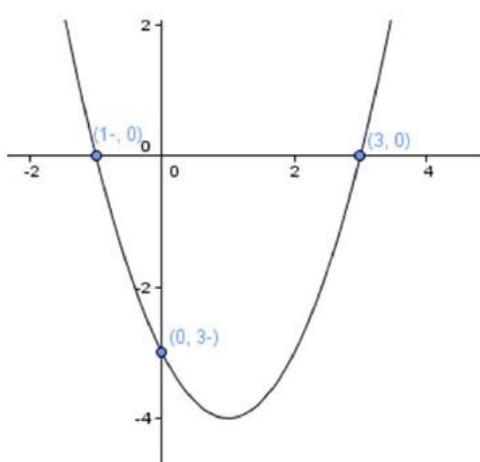
تحديد درجة كثير الحدود (المعادلة)//
Console.WriteLine("Enter n = ");
int n = int.Parse(Console.ReadLine());
double[] x = new double[n+1];
تحديد ثوابت المعادلة//
for (int i = 0; i <= n; ++i)
{
    Console.Write("X[{0}] = ", i + 1);
    x[i] = double.Parse(Console.ReadLine());
}
تحديد نقطة التماس//
Console.WriteLine("Enter tangency point:");
Console.Write("x1 = ");
double x1 = double.Parse(Console.ReadLine());
Console.Write("y1 = ");
double y1 = double.Parse(Console.ReadLine());

```



```
// إيجاد ثوابت معادلة المشتق //
double [] dy = new double[n+1];
for (int i = n; i > 0; --i)
    dy[n-i] = i * x[n-i];
حساب الميل (المشتقة عند نقطة التماس)//
double m = 0;
for (int i = 0; i <= n; ++i)
{
    m += dy[i] * Math.Pow(x1, n - i-1);
}
عرض معادلة المماس// 
double c = -m * x1 + y1;
if (c>0)
    Console.WriteLine("y = {0} x + {1}", m,c);
else if (c <0)
    Console.WriteLine("y = {0} x {1}", m, c);
Console.ReadKey();
```

لنجرب البرنامج السابق رياضيًّا، وبموازاة ذلك جرب الأرقام التي اخترناها وتحقق من النتائج.



$$y = x^2 - 2x - 3$$

$$y' = 2x - 2$$

نقطة التماس مثلاً:

$$m = 2(3) - 2 = 4$$

وبالتالي:

$$y = 4x - 12$$

```
file:///C:/Users/Hp/Documents/Visual Studio 2012/Projects/ConsoleApplication1/Cons...
Enter n = 2
x[1] = 1
x[2] = -2
x[3] = -3
Enter tangency point:
x1 = 3
y1 = 0
y = 4 x - 12
```





الفصل الخامس - أنواع بيانات خاصة بك!

قد تحتاج لأنواع بيانات خاصة بك، قد تكون هذه الأنواع ثابتة ومحددة الاستخدام ولا يُحتاج إليها إلا في برنامحك أو مسألك، أو قد تكون هذه الأنواع لها بيانات فرعية تعرّفها أنت. للتعامل مع البيانات الثابتة نستخدم ما يسمى المعدّات **Enumerations** أما المتغيره متعددة الفروع من البيانات فنستخدم التراكيب **Structures**.

الstrukت

لإنشاء أنواع مركبة تحوي أنواع عديدة من المتغيرات نستخدم التراكيب، والتي تغنيك عن استخدام متغيرات مختلفة لاختزان بيانات متعلقة ببعضها، كبيانات طالب في مدرسة، اسمه وعمره وعلاماته. نصّرّح عن التركيب كالتالي:

```
اسم_التركيب struct
{
    Public struct's Vars;
    .
    .
}
```

ملاحظة

- يجب أن تكون متغيرات التركيب عامة حتى نستخدمها في كافة توابع وإجراءات المشروع.



لأخذ المثال التالي:

```
struct Student
{
    public string Home;
    public int Age;
}

static void Main(string[] args)
{
    Student Ali = new Student();
    Ali.Home = "Aleppo";
    Ali.Age = 27;
}
```

الغاية من المثال إيصال الفكرة من التراكيب، وشرحه كالتالي:

نعرف تركيب باسم `Student` يحوي متغيرين أحدهما `Home` والثاني `Age` عددي صحيح. في الدالة `Main` عرفنا متغير `Ali` على أنه من النوع `Student` الذي ألقناه منذ ثانية، للوصول لمتغيرات التركيب نضع اسم المتغير المعرف على أنه تركيب، ثم نقطة، ثم المتغير المطلوب. وهكذا بإمكاننا استخدام التراكيب للإدخال والإخراج.

حاول ربط التراكيب - والمواضيع المتقدمة التي سنراها في الفصول اللاحقة إن شاء الله، وخصوصاً التي تتعامل مع OOP - بالحياة الاجتماعية الواقعية، أو من الممكن أن نقول بالحياة الإنسانية. بسّط المفهوم البرمجي دائماً إلى مفهوم إنساني سهل ومريح لتحظى بالفائدة الكبيرة مما تقوم به. لاحظ كيف سأشرح المثال الأخير:

في الحالة العامة (الإجراء `Main`) لدينا كائن يسمى `Ali`، هو طالب، منزله (`Ali`, أو بلغة إنسانية `Ali's Home`) هو `Aleppo`, وعمره (`Ali.Age`, أو `Ali's Age`) هو 27. حيث أن الموطن `Home` وال عمر `Age` هي البيانات المطلوب تعريفها عند إضافة كائن ما على أنه طالب.

ملاحظة

- عند تعريف متغير ما على أنه تركيب، فإنه سيتم حجز مكان بالذاكرة لجميع متغيرات التركيب على أنها خاصة بهذا المتغير.



Nested Structs المترابطة

بشكل عام فإن التراكيب ماهي إلا كتل أو بنى برمجية مركبة من عدة متغيرات وطرق، وفي بعض الأحيان قد تربط عدة تراكيب بعضها بحيث يجعل أحد التراكيب من مكونات تركيب ما. لاحظ المثال:

```
struct Student
{
    public string Home;
    public int Age;
    public BirthDay Birth;
}
struct BirthDay
{
    public int Day, Month, Year;
}
static void Main(string[] args)
{
    Student Ali = new Student();
    Ali.Home = "Aleppo";
    Ali.Age = 27;
    Ali.Birth.Day = 3;
    Ali.Birth.Month = 5;
    Ali.Birth.Year = 1992;
}
```

مصفوفة التراكيب Array of Structs

وعلى غرار المتغيرات العاديّة - الموجودة افتراضياً في لغة البرمجة - فإنه من الممكن تشكيل مصفوفة من التراكيب. المثال التالي يوضح ذلك:

```
Student[] S = new Student[2];
// التصريح عن مصفوفة مكونة من تركيبين
// يتم التعامل مع المصفوفات عادةً من خلال الحلقات
for (int i = 0; i < 2; ++i)
{
    Console.Write("Enter Student [{0}] Home: ", i);
    S[i].Home = Console.ReadLine();
    Console.Write("Enter Student [{0}] Age: ", i);
    S[i].Age = int.Parse(Console.ReadLine());
    Console.Write("Enter Student [{0}] Day of Birth: ", i);
    S[i].Birth.Day = int.Parse(Console.ReadLine());
    Console.Write("Enter Student [{0}] Month of Birth: ", i);
    S[i].Birth.Month = int.Parse(Console.ReadLine());
    Console.Write("Enter Student [{0}] Year of Birth: ", i);
```



```
S[0].Birth.Year = int.Parse(Console.ReadLine());
}
```

طرق خاصة بالتراتيب

بإمكانك إنشاء بعض الطرق لتؤدي بعض الوظائف على التراتيب، لاحظ:

```
struct Student
{
    public string Home;
    public int Age;
    public BirthDay Birth;
    public override string ToString()
    {
        string str;
        str = string.Format("Home: {0}, Age: {1}, BirthDay: {2}/{3}/{4}",
            Home,
            Age,
            Birth.Day,
            Birth.Month,
            Birth.Year);

        return str;
    }
}
struct BirthDay
{
    public int Day, Month, Year;
}
static void Main(string[] args)
{
    Student Ahmad = new Student();
    Ahmad.Home = "Aleppo";
    Ahmad.Age = 25;
    Ahmad.Birth.Day = 22;
    Ahmad.Birth.Month = 5;
    Ahmad.Birth.Year = 1994;
    Console.WriteLine(Ahmad.ToString());
    Console.ReadKey();
}
```

عند تنفيذ البرنامج ستحصل على عبارة نصية محتواها محدد بالقيمة المعادة من التابع `ToString`. (هكذا تم إنشاء التابع `ToString` الذي تراه في مختلف المتغيرات والتوابع والإجراءات).



المعدّات Enum

تستخدم المعدّات عند وجود مجموعة من القيم محددة لا تتغيّر ولا تزداد ولا تنقص، ومصرّح عنها سابقاً، ومرتبة وفق ترتيب معين يبدأ من 0. نصرّح عن المعدّد كما يلي:

```
اسم المعدّد enum
{
    enum's Values;
    .
    .
}
```

ملاحظة

- نكتب كل من التراكيب والمعدّات خارج الدالة Main، فالدوال لا يمكن أن تحتوي على تراكيب، في حين أن التراكيب من الممكن أن تحتوي على دوال.

المعدّات هي شكل آخر للثوابت والتي تحدثنا عنها في الفصل الأول، والتي قلنا أنه في حال كان لدينا معلومات ثابتة فيفضل إسنادها لمتغيرات ثابتة. وكنا وقتها نتعامل مع قيمة ثابتة واحدة أي متغير ثابت واحد. أما لو أردنا التعامل مع مجموعة من الثوابت المتعلقة ببعضها فالمحددات هي ضالتنا.

في الفصل الأول أخذنا مثلاً عن درجات الحرارة، وقلنا أنه لو كانت لدينا حالة ثابتة هي درجة التجمد بالفهرنهايت 32 مثلاً، وللتعامل معها فإننا أنشأنا متغيراً ثابتاً يمثل هذه الدرجة (انتقل لهناك اقرأ الفقرة وتذكر فكرتها وعد إلى هنا لتدخل بالجو أكثر).

لأخذ المثال التالي:

```
// Defining Enum
enum Day
{
    Saturday, Sunday,
    Monday, Tuesday,
    Wednesday,
    Thursday,
    Friday,
```



```
{
// Using Enum
static void Main(string[] args)
{
    Console.Write (Day.Saturday); //Prints Saturday
    Console.ReadKey ();
}
```

يجهل كثير من المبرمجين - الجدد - الفائدة من المعدّات، فمنهم من يقول أنه يمكن الاستغناء عنها واستخدام متغيرات من نوع string أو bool مثلاً عوضاً عن كتابة أسطر برمجية عديدة لحصر المستخدم في خيارات محددة.

في البداية أحب أن أخبرك أن enum هي للمبرمجين وليس للمستخدمين. عند كتابة برنامج طويل عريض وتنذيرك بوجود حالات معينة لقيمة أو متغير ما فإن المعدّات ستساعدك في ذلك.

ثم إنه قد ترغب بنشر كودك، قد تعطيه لصديق لك. إذا كانت لديك متغيرات تأخذ قيمة محدودة فعليك عندها إرفاق مجموعة من التعليمات عبر ورقة أو رسالة أو اتصال مفادها أن المتغير الفلاني يأخذ القيم الفلانية فقط. ماذا لو نسي هذه التعليمات واستخدم قيمة من عنده؟؟ سيحظى البرنامج بكم جيد من الأخطاء عندها خصوصاً إذا كانت هناك توابع ومتغيرات مبنية على هذا المتغير الذي أدخلت قيمة خاطئة له.

ولا تنسى أن نشر الأكواد - الكبيرة منها خصوصاً - يتم عادةً عبر مكتبات dll والتي ستناقشها في الفصل السابع، أي أن صديقك لا يمكنه التبني بمحفوظات الكود (هو فقط سيستطيع الوصول إلى بعض الخصائص والطرق والتي ستخترها أنت برمجيا¹).

والأكثر من ذلك، ماذا لو قمت بنشر الكود على نطاق واسع؟؟؟ مثلاً عبر موقع ما أو مجموعة من مجموعات التواصل الاجتماعي أو أي مجتمع برمجي، هل ستتصل بجميع المبرمجين وتخبرهم بالإمكانيات المتاحة حتى لا يشتموك عندما يواجهون المشاكل والأخطاء! أم أنك ستترك لهم ملفاً نصياً باسم ReadMe بحيث يقرؤونه قبل استخدام الكود؟؟؟؟

¹ أصلاً لو كانت الأسطر البرمجية متاحة بين يديه وقرأها لوجد الإمكانيات المتاحة التي يمكن للمتغيرات أن تأخذها وذلك عبر بنى الشرط التي تحكم وتدبر البرنامج بناءً على قيم المتغيرات (والتي قلنا أنها محدودة).



لاحظ معي أنه عند استخدام المعدات فإن C# ستحصر المتغير بقيم محددة، فهنا لدينا سبعة خيارات لا ثامن لها!

```
Day d = Day.Saturday;
if (d == |
Console.R
```

Day

- DayMisalignedException
- DateTime
- DateTimeKind
- DateTimeOffset
- Day

وهذا أفضل من استخدام متغيرات نصية مثلا للدلالة على أيام الأسبوع مثلًا:

```
string d = "July";
if (d == "Saturday")
```

لاحظ أن الشرط لن يتحقق بحياته. ستقول لي أن الخطأ مني لأنني أنسندت لـ d القيمة "July" والتي ليست أصلاً من أيام الأسبوع، وسأقول لك أنني لو استخدمت المعدات لما سمح لي البرنامج من إسناد قيمة لا تمت بصلة لأيام الأسبوع أصلاً، صح الا لا (على قوله أحمد مساد).

يعتقد بعض المبرمجين أن المعدات وظيفتها إخبار المستخدم بخيارات محددة، مثلاً في الألعاب لديك ثلاثة متغيرات (Hard, Medium, Easy). وهذا اعتقاد خطأ لأن المعدات - كسائر المتغيرات - لا تظهر للمستخدم بشكل مباشر وإنما هي عبارة عن ذاكرة برمجك والتي يحتفظُ فيها ببياناته وأفكاره. أما ما يظهر للمستخدم هو ناتج أوامر الطباعة فقط!

عد إلى فصل الأساسية وتحديداً في فقرة الأكواد الأساسية ثم فقرة تغيير لون نافذة المشروع، لاحظ الكود الموجود، فيه ثوابت هي الألوان، هذا يعني أن الألوان ماهي إلا معدات.

```
Console.ForegroundColor = ConsoleColor.gr
```

ConsoleColor

- DarkGray
- DarkGreen
- Gray
- Green

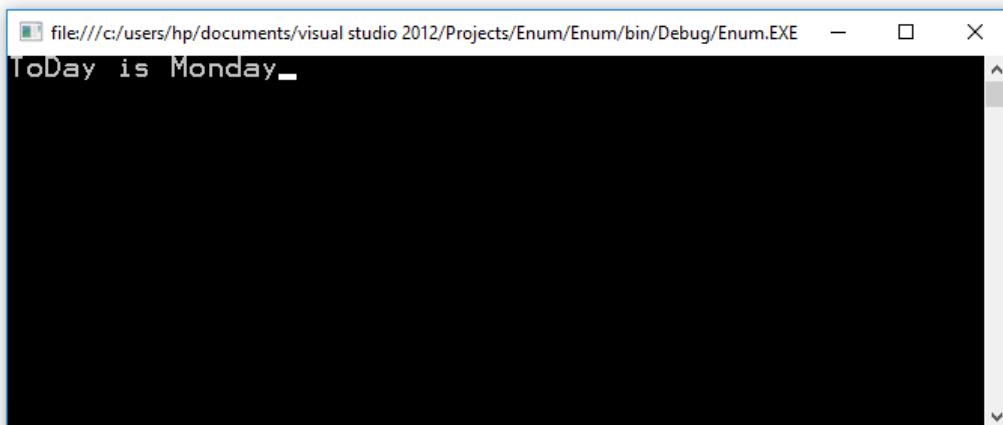


جرب أن تكتب السطر التالي وستظهر رسالة منبثقه مفادها أن المترجم يضحك عليك:

```
Console.ForegroundColor = "Green";
```

في الواقع فإن المعدادات ماهي إلا متغيرات رقمية صحيحة، تبدأ بالرقم 0:

```
Day d = Day.Sunday;  
Console.WriteLine("ToDay is {0}", ++d);
```



وفي كثير من الأحيان فإنه قد ترغب بتحزين المتغيرات من نوع enum بأرقام على كيفك، وفق حاجتك ومسئلك:

```
enum Grades  
{  
    first = 1,  
    second, third  
}  
static void Main(string[] args)  
{  
    Grades g = Grades.first;  
    int num = (int)g;  
    if (num == 1)  
        Console.WriteLine("Winner!");  
    Console.ReadKey();  
}
```

```
enum Grades  
{  
    first = 1,
```



```

        second, third
    }
static void Main(string[] args)
{
    Grades g;
    g = (Grades)2;
    Console.WriteLine(g); //Prints second
    Console.ReadKey();
}

```

مع العلم أن مقدار الزيادة لباقي العناصر ستكون 1 (أي أن second ستخزن على أنها 2 وthird على أنها 3).

كما يمكنك تخزين أكثر من متغير على نفس الرقم:

```

enum Answer
{
    Yes = 1, Ok = 1, True = 1,
    No = 0, False = 0,
}
static void Main(string[] args)
{
    Answer a = Answer.Ok;
    if (a == Answer.Yes)
        Console.WriteLine("Accepted!");
    else
        Console.WriteLine("Refused!");
    Console.ReadKey();
}

```

طبعا النتيجة ستكون Accepted! لأن كلاً من Ok وYes وTrue مساوية لبعضها.

كما يمكن ألا تخزن العناصر بأرقام متعلقة ببعضها أصلا:

```

enum Coin
{
    penny = 1, //قرش
    nickle = 5, //خمس قروش
    dime = 10, //عشر قروش
    quarter = 25, //ربع دولار
    half = 50, //نصف دولار
    dollar = 100, //دولار واحد
}

```



وبالعودة لدرجات الحرارة:

```
enum Temp
{
    FREEZING = 32,
    LUKEWARM = 65,
    HOT = 105,
    BOILING = 212
}
```

حيث أن الكود السابق يكافئ بالنسبة للثوابت الكود التالي:

```
const int FREEZING = 32;
const int LUKEWARM = 65;
const int HOT = 105;
const int BOILING = 212;
```

بشكل افتراضي المعدادات هي قيم صحيحة `int`, لكن بإمكانك تخصيص نوع آخر لها. لاحظ الكود التالي:

```
enum Temp : byte
{
    FREEZING = 32,
    LUKEWARM = 65,
    HOT = 105,
    BOILING = 212
}
```

أي أن عناصر المعدد قيمها من النوع `byte` وليس `int`, وإشارة النقطتين تعني أن ما قبلها يرث ما بعدها، والوراثة مصطلح سيناقشه في الفصل السابع.

أنواع البيانات التي يمكن للمعدادات أن ترث منها هي التي من النوع الرقمي بطبعتها، مثل `short`, `byte` و `long`.





الفصل السادس - الطرق، التوابع والإجراءات

الطرق Methods هي مجموعة من الأوامر المجمّعة تحت اسم معين، وعند استدعاءها - ذكر اسمها - يتم تنفيذ الأوامر التابعة لها. الغاية منها تبسيط البرنامج بأكمل شكل ممكن (لا أدرى لماذا أحب استخدام هذا التعبير)، بحيث يسهل التعديل عليه وإيجاد الأخطاء واستخدامه في برامج أخرى.

عند الاستخدام الكمي للكود يكون من الأفضل استخدام الطرق وذلك لتجنب تكرارها، تخيل معي أنك استخدمت كودًا ما في برنامجك في أكثر من موضع، وعند التجريب كان يعمل بالشكل الصحيح لكنك وجدت له شكلاً أفضل بحيث يعطي نتائج أفضل، هل من المعقول البحث عنه سطراً سطراً والتعديل عليه، وأثناء بحثك قد تعدل على أمور أخرى خطأً، كما قد تنسى إحدى النسخ من هذا الكود دون تعديل! ماذا لو وضعت هذا الكود في إحدى زوايا برنامجك، وكلما احتاج برنامجك لهذا الكود تقول له، إذهب إلى الموضع الفلاني، وطبق الكود الموجود هناك، وتعالَ لتكمل مسيرتك؟ يعبر عن هذه العملية بوضع الكود في مايسمي بالطريقة، والتي قد تحتاج متغيرات وقد لا تحتاج، والتي قد تعطيك قيمةً وقد لا تعطيك.

مرّ علينا سابقاً مجموعة من الطرق استخدمناها بكثرة مثل الطريقة `Convert.ToInt32()` أو `Console.WriteLine()` أو `Console.ReadKey()` .`Math.Round()` أو `MessageBox.Show()`



حيث كل من ReadKey و WriteToInt32 هي طرق تابعة للفئة¹. Console، و Round هي طريقة للفئة Math، و Show هي طريقة للفئة MessageBox. حيث إن الفوائل الموضحة مابين الأقواس تمثل عدد المتغيرات اللازمة لإعطاء النتيجة المطلوبة، مع الأخذ بعين الاعتبار أنه من الممكن تجاهل بعض المتغيرات أحياناً.

وتتمثل الطرق بالدوال Procedures والإجراءات Functions، الأولى تستقبل متغيرات وتجري عليها بعض العمليات ثم تعطي نتيجة (return) من نفس نوع الدالة (.., int, string, double, bool ..)، أما الثانية فلا تعيد نتيجة (لا تملك الكلمة return)، ولا نوع لها (تسبق بالكلمة المحجوزة void).

سأعيد الفكرة عليك: تتمثل الطرق بالدوال Procedures والإجراءات Functions، الأولى تستقبل متغيرات وتجري عليها بعض العمليات ثم تعطي نتيجة (return) من نفس نوع الدالة (.., int, string, double ..)، أما الثانية فلا تعيد نتيجة (لا تملك الكلمة return)، ولا نوع لها (تسبق بالكلمة المحجوزة void).

تكريم وأعد قراءتها من جديد. وصلت الفكرة 😊؟ وبعبارة صريحة أكثر: حفظتها؟!!



وتعُرف أيضًا التوابع والإجراءات بأنها ليست إلا قطعة من البرنامج، تحتوي على مجموعة من الأكواد، بتنفيذها نحصل على نتائج ما، سواءً أكانت النتيجة ستعاد أم لا.

الصيغة العامة للتصرح عن الدوال:

```
(متغيرات_الدالة) اسم_الدالة نوع_الدالة2  
أكواد الدالة {  
.  
. .  
return; قيمة}
```

ملاحظة

- القيمة المعادة بعد كلمة return هي عبارة عن متغير يحمل قيمة نتيجة إجراء بعض العمليات، ومن الممكن وضع علاقة بحيث يُعاد نتيجتها.

¹ الفئة: هي تعبير آخر للصنف – Class.

² تم وضع الخط السفلي لتمييز العبارات عن بعضها.



أما الصيغة العامة للتصریح عن الإجراء:

```
void (متغيرات_الإجراء) اسم_الإجراء
{
    أکواد_الإجراء
    .
    .
}
```

ملاحظة

- كما هو الحال في بني التحكم وبعض حلقات التكرار، فالطرق لا تلحق بفاصلة منقوطة.
- في الإجراءات لا توجد الكلمة `return` لأنها لا تعيد أية قيمة.

خلال الفصول السابقة لم نتعامل مع التوابع أو الإجراءات أو الوظائف بشيء من التفصيل، وإنما اهتممنا بالأكواد بحد ذاتها. بالعودة لأول شيء تعطاه في مشاريعك وهو التابع `Main`, عد إلى الصفحات الأولى من هذا الكتاب وتحديداً في بدايات الفصل صفر، أو افتح مشروعًا جديداً في الفيجوال ستوديو وخلصت القصة، ستجد أن التابع `Main` سُبق بكلمتين، `static void`، الأخيرة نعلم قصتها والغاية منها، أما الأولى فتعني أن أي متغير أو إجراء أو تابع خارج التابع `Main` غير معرف بالكلمة `static` لن يتم التعرف عنه داخلها.. جرب ذلك بنفسك ولاحظ النتيجة.

لاحظ هذا المثال لفهم التوابع والإجراءات:

```
static void procANS (int answer) //answer = 5 * 2
{
    Console.WriteLine ("A = " + answer);
}
static int funANS (int x, int y) //x = 5, y = 2
{
    return x * y;
}
static void Main (string [] args)
{
    int a, b; a = 5; b = 2;
    procANS (a * b);
    int ans = funANS (a, b);
    Console.WriteLine ("A = " + ans); Console.ReadKey ();
}
```



إن نتيجة الكود السابق ستكون $A = 10$ مكررة بسطرين، وهو يلخص الإجراءات والدوال معاً.

كشرح موجز على الكود السابق يمكن كتابة ما يلي:

لدينا إجراء يسمى procANS اختصاراً لـ "الجواب بطريقة الإجراءات"، يتلقى متغيراً صحيحاً، ويظهره مع عبارة نصية. كما أنه لدينا تابع يسمى funANS اختصاراً لـ "الجواب بطريقة الدوال"، يتلقى متغيرين صحيحين، ويعيد ناتج جدائهما.

كما أنه يوجد لدينا التابع Main، والذي يحوي متغيرين صحيحين، لهما قيمتين صحيحتين، تم فيه استدعاء طرفيتين إحداهما إجرائية والأخرى تعيد قيمة، وذلك بالشكل التالي:

في السطر الثالث من التابع، تم مناداة الإجراء وإعطائه ناتج جداء المتغيرين، ليقوم بما فيه من أكواد بغض النظر عنها، وبغض النظر من أن المتغيرات التي أرسلناها ستلزمه أم لا. (أعد قراءة الجملة الأخيرة)

في السطر الذي يليه، تم مناداة التابع وإعطائه قيمة المتغيرين، ليسندهما إلى متغيرات أخرى مناظرة لهذه المتغيرات، ثم يعيد قيمة سندها إلى متغير من نفس نوع الدالة، لنظهر النتيجة فيما بعد.

الزبدة:

- لا يلزمنا تعريف متغيرات لاستقبال نتيجة الإجراءات، لأنها لن تعيد قيمة أصلاً. وفي حال كانت النتيجة ستظهر في خرج البرنامج عندها يجب إخراجها في الإجراء نفسه أو إسناد النتيجة من داخل الإجراء لمتغير يعمل في كافة الإجراءات ليتم إظهارها فيما بعد.

- يجب تعريف متغيرات لاستقبال نتيجة الدوال، أو استدعاء التابع كمتغير لتابع آخر مثل استخدام الكود Console.WriteLine (funANS(a, b)) والذي يعني عن الكود المستخدم في المثال.

- لاحظ أن النتيجة ذاتها في كلا الطريقيتين، الإجراء أو التابع. واستخدام هذه الطريقة أو تلك تعتمد على حاجتك لها والظروف الراهنة بالإضافة للمتغيرات التي لديك ومقدار راحتك وسعادتك بالتعامل معها، وهذا مشابه لأن تقول لأخيك مثلاً: أنا عطشان (فيذهب ويحضر لك كأس ماء)، أو أن تقول: أحضر لي كأس ماء بعد إذنك (فيذهب ويحضر لك كأس ماء). هل اختلفت النتيجة في كلا الطريقيتين؟!



تمرير المتغيرات

لاستخدام إحدى الطرق - الممثلة بالتتابع والإجراءات - لا يكفي فقط ذكر اسم الطريقة المطلوب استدعائها، وإنما قد تحتاج لمتغيرات تمثل وسطاء يتم استخدامها للحصول على نتيجة ما. عند استدعائك لتتابع أو إجراء - وذلك بذكر اسمه - سترسل معه المتغيرات التي ستتمثل وسطاءه، عملية الإرسال هذه تسمى تمريراً.

التمرير بالقيمة

وهي أبسط أنواع التمرير. وفيها يتم إرسال قيمة المتغيرات إلى التابع أو الإجراء، ليقوم ببعض العمليات باستخدام قيمها. لا تتغير قيمة هذه المتغيرات عند استدعاء الطرق، ف مجال رؤيتها لا يتعدى مكان تمريرها.

التمرير بالمرجع

وفيه يتم تمرير مرجع المتغير وهو العنوان الموجود فيه المتغير في الذاكرة، وعندها فإن أي تغيير في قيمة المتغير في الدالة أو الإجراء المستدعي يصحب معه تغير في قيمة المتغير الممرر مرجعه.

التمرير بالإخراج

وهو مماثل للتمرير بالمرجع ولكنه يختلف عنه في أن المتغير الممرر مرجعه من الممكن ألا يحوي على قيم أثناء التمرير، في حين أن هذا الأمر غير متوفّر عند استخدام أسلوب التمرير السابقين.

لاحظ المثال التالي:

```
static void valProc(int x, int y)
{
    y = x = 11;
    Console.Write("in method: ");
    Console.WriteLine(" x = " + x);
    Console.WriteLine(" y = " + y);
}
static void refProc(ref int y)
{
    y = 11;
    Console.Write("in method: ");
    Console.WriteLine(" y = " + y);
}
static void outProc(out int z)
{
```



```

z=12;
Console.WriteLine("in method: ");
Console.WriteLine (" z = " + z);
}
static void Main(string[] args)
{
    int x;
    Console.WriteLine("Before Passing:");
    x = 10;
    Console.WriteLine("in Main: x = " + x);
    Console.WriteLine("Passing with value:");
    valProc(x,2*x);
    //x = 10
    Console.WriteLine("in Main: x = " + x);
    Console.WriteLine("Passing with reference:");
    //Don't forget that x is 10 in Main Procedure!
    refProc(ref x);
    Console.WriteLine("in Main: x = " + x);
    //Now x is 11 Because we have changed it from the method
    Console.WriteLine("Passing with output:");
    int z; //There is no value to z
    outProc(out z);
    //Now z has value
    Console.WriteLine("in Main: z = " + z);
    Console.ReadKey();
}

```

ماحدث في المثال هو كالتالي:

ابتدأنا البرنامج على بركة الله في الإجراء Main، عرّفنا متغيراً عددياً صحيحاً x ثم أظهرنا قيمته قبل التمرير.

ثم استدعينا الإجراء valProc وأعطيته القيم التالية لسندتها لوسطائه: 10 و20، وهناك - في الإجراء المذكور - أُسندنا القيمة 11 للمتغيرين x وy والذين يعتبران وسطاء هذا الإجراء، ثم أظهرنا قيمهما، وعدنا لمكان الاستدعاء في الإجراء Main. (لا تنسَ أن قيمة x المتغير الموجود بالإجراء لا تمت بصلة لقيم x أو y الموجودين في الإجراء valProc، مجرد تشابه بالأسماء لا أكثر ولا أقل)

ثم استدعينا الإجراء refProc وأعطيته مرجع المتغير x ليتسنى للإجراء المستدعى الاستفادة من x من جذورها في الذاكرة، وهناك - في الإجراء المذكور - أُسندنا القيمة 11 لـ y والذي يعتبر الوسيط الوحيد لهذا الإجراء، ثم أظهرنا قيمته، وعدنا لمكان الاستدعاء في الإجراء Main.



ثم عرّفنا متغيراً z واستدعينا outProc مع إعطاءه مرجع z ك وسيط لكن مرناه بالإخراج، لأنه لا يحوي أية قيمة، وفي الإجراء أسنداً للمتغير z قيمة، والذي بدوره سيؤثر على المتغير z الموجود في الإجراء الرئيسي، لأن لكليهما نفس العنوان في الذاكرة.

```
file:///C:/Users/Hasan/documents/visual studio 2012/Projects/ConsoleApplication10/ConsoleApp...
Before Passing:
in Main: x = 10
Passing with value:
in method: x = 11 y = 11
in Main: x = 10
Passing with reference:
in method: y = 11
in Main: x = 11
Passing with output:
in method: z = 12
in Main: z = 12
```

إذا كانت الأفكار السابقة صعبة عليك، سأعطيك مثالاً آخر:

```
static void MyProc(int y)
{
    y += 10;
}
static void Main(string[] args)
{
    int x = 10;
    MyProc(x);
    Console.WriteLine(x); //Prints 10
    Console.ReadKey();
}
```

قيمة x ستبقى 10 سواءً قبل استدعاء الطريقة MyProc أو بعدها.. أما باستخدام المرجع:

```
static void MyProc(ref int y)
{
    y += 10;
}
static void Main(string[] args)
{
```



```
int x = 10;
MyProc(ref x);
Console.WriteLine(x); //Prints 20
Console.ReadKey();
}
```

الآن قيمة `x` أصبحت 20 بعد استدعاء الطريقة `MyProc`, وهذا كل ما أريدك أن تعلمه من هذا البحث. بقية الأفكار تحصيل حاصل ستحصل عليها مع مرور الأكوا德.

طيب قد يخطر على بالك السؤال التالي: لماذا سأحتاج لاستخدام الإجراءات - الطرق التي لا تعيد قيمة، والتي تبدأ بـ `void` - بينما التوابع - الطرق التي تعيد قيمة، وتنتهي بـ `return` - هي أبسط؟؟

في الواقع فإن التوابع تعيد قيمة واحدة وهي القيمة المعادة بالكلمة `return`, في حين أنه في بعض الأحيان فإنك قد تحتاج لإرجاع أكثر من قيمة من الطريقة. فمثلاً إذا أنشأت طريقة تعطيها قيمة زاوية `x` وترغب بأن تعيد لك قيمة `sin` و `cos` و `tan` هذه الزاوية - وتفاصيل أخرى إذا رغبت - فإن التوابع لن تشبع رغبتك. أنشئ إجراءً يمكن استدعاؤه بالكود التالي:

```
double sin, cos, tan;
Math(x, out sin, out cos, out tan);
```

تمرير عدد غير محدد من الوسطاء كمصفوفة

يمكنك تمرير العدد الذي ترغب به من الوسطاء دون الحاجة لتخصيص عددها طالما هي من ذات النوع وذلك باستخدام البادئة `:params`

```
static int Sum(params int[] nums)
{
    int x = 0;
    foreach (int n in nums)
        x += n;
    return x;
}
static void Main(string[] args)
{
    Console.WriteLine(Sum(1, 10, 1000)); //Prints 1011
    Console.WriteLine(Sum(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)); //Prints 48
    Console.ReadKey();
}
```



وماحدث هو أن الطريقة `Sum` باتت تستقبل عدداً غير محدد من البيانات، من الممكن إدخال العدد الذي ترغب به من الوسطاء دون الحاجة لتحديدها بشكل مفصل.

لكن ماذا لو أردت تمثيل أنواع أخرى من البيانات مع البيانات السابقة غير محددة العدد؟؟

بإمكانك ذلك بتخصيص وسطاء لاستقبالها وذلك بشرط وضع المصفوفة التي ستنطبق على البيانات المتماثلة غير محددة العدد آخر الوسطاء.

```
static string ContactInfo(string name, bool married, params char[]  
PhoneNumber)  
{  
    string contact = string.Format("Name {0}", name);  
    if (married)  
        contact += ", Married";  
    else  
        contact += ", Single";  
    contact += ", PhoneNumber: ";  
    foreach (char c in PhoneNumber)  
        contact += c;  
    return contact;  
}  
static void Main(string[] args)  
{  
    Console.WriteLine(ContactInfo("Elias", false, '0', '9', '5', '4', '9', '6', '6', '2', '7'));  
    Console.ReadKey();  
}
```

والنتيجة ستكون:

Name Elias, Single, PhoneNumber: 0954796627

إرجاع التوابع كمصفوفة

اتفقنا منذ البداية على أن الطرق نوعان، إجراءات أو توابع.. واتفقنا على أن الإجراءات من الممكن أن تستقبل قيمة ولا تعيد - لا ترجع - شيئاً، في حين أن التوابع من الممكن أن تستقبل قيمة ثم تعيد قيمة تمثل نتيجة تنفيذ التابع، حيث يتم إرجاع القيمة باستخدام الكلمة `return`، كما أن نوع القيمة المعاودة يكون من نوع التابع.



ناقشنا أيضاً إمكانية تمرير عدد غير محدد من القيم على شكل مصفوفة باستخدام البادئة params، والآن سيناقش كيفية إرجاع التابع كمصفوفة، أي أن القيمة المعادة من التابع هي مصفوفة!

لاحظ المثال التالي:

```
static int[] MultipleBy2(params int[] nums)
{
    for (int i = 0; i < nums.Length; i++)
        nums[i] *= 2;
    return nums;
}
static void Main(string[] args)
{
    int[] numbers = new int[3];
    numbers = MultipleBy2(1, 10, 1000);
    foreach (int n in numbers)
        Console.WriteLine(n);
    Console.ReadKey();
}
```

التابع MultipleBy2 يستقبل مجموعة غير محددة من العناصر الرقمية الصحيحة كمصفوفة، ويعيد مصفوفة أخرى على أنها ناتج ضرب الأولى باثنان.





الفصل السابع – الفئات Classes

تعتبر الفئات مدخلًك للبرمجة كائنية التوجّه OOP، وفيها قلنا أن أي شيء يعتبر كائناً Object، وهي تقرير لمبدأ الحياة الحقيقية على الحياة البرمجية، فللائن خصائص Properties تميزه عن الكائنات الأخرى، وأفعال هي طرق Methods يقوم بها.

في الحياة الحقيقة هناك كائنات مختلفة تتبع لفئات (أنواع) مثل أنس أو سيارات أو آلات أو .. إلخ، ولهذه الكائنات مجموعة من الصفات مثل الاسم والعمر ولون البشرة ولون الشعر وما إذا كان أيمن أو أيسر بالنسبة للأناس مثلاً، وبإمكان هؤلاء الناس أن يقوموا بمجموعة من الأفعال مثل الكتابة والقراءة والكلام والسمع والاستيقاظ، كما أنه تمر عليهم لحظات و مجريات يشعرون فيها بالجوع أو الإرهاق أو السعادة.. وبإسقاط هذا المفهوم على مكونات البيئة البرمجية نحصل على مجموعة من الخصائص للكائنات التي نتعامل معها مثل Title و Name و Text و Visible، ومجموعة من الأفعال مثل ReadKey و Write و Clear و Enter و Leave و KeyPress. كما أنه بإمكانك إضافة خصائص أو أفعال خاصة بك وفق حاجتك.

لنأخذ مثالاً لتقرير الفكرة: لدينا كائن اسمه حسن، يتبع للفئة إنسان، لديه خصيتان الاسم وال عمر كمثال بسيط، يقوم هذا الكائن بأخذ قيم هاتين الخاصيتين وفق الطريقة Initialize والتي تعني تحضير (بإمكانك تغيير اسم هذه الطريقة إلى Setup مثلاً)، ويقوم أيضاً بعرض تفاصيله وفق الطريقة ShowDetails.



على غير العادة - التي جرت في الفصول السابقة - لن أقوم بسرد صيغة عامة للفئات وسأكتفي بمثال يوضح أبسط أشكالها لغاية في نفسي..

ملاحظة

- الكائن هو تعابير خاص بينما الفئة هي تعابير عام. فمثلا الكتاب يعتبر فئة، لكن "C# من البداية حتى الإتقان" يعتبر كائناً.
- تسمى عملية تعريف الكائنات باستنساخ الفئات.

الطريقة Initialize هي إجراء تأخذ متغيرين، الاسم وال عمر، أما ShowDitails فلا تأخذ أية وسطاء ولا تعيد قيمة (أي أنها إجراء أيضاً).

بدايةً سننشئ ملفاً للفئة Human، وذلك من Class Project ثم Add Class. سيرجّع بجانب ملفات المشروع. بإمكانك استخدام ملف هذه الفئة في برامج أخرى وذلك عن طريقة استيرادها عبر الأمر Add Existin Item من القائمة Project نفسها.

```
class Human
{
    string cNAME;
    int cAGE;
    public void Initialize(string Name, int Age)
    {
        cNAME = Name;
        cAGE = Age;
    }
    public void ShowDitails()
    {
        Console.WriteLine ("Name = {0}, Age = {1}",
                           cNAME,
                           cAGE );
    }
}
```

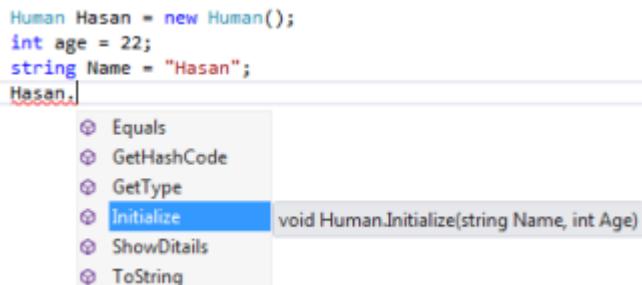
يفضل دائما التفريق بين متغيرات الفئة نفسها ومتغيرات الفئات الأخرى، يمكن ذلك من خلال استخدام حرف كبير أو صغير أو تسبيق أسماء المتغيرات بلواحق تعتمد عليها دائما في برماجك. بالعودة للإجراء Main، اكتب الكود التالي:

```
Human Hasan = new Human() // استنساخ الفئة
int age = 22;
string Name = "Hasan";
```



```
Hasan.Initialize(Name, age);
Hasan.ShowDitails();
Console.ReadKey();
```

عند إضافة الفئة Human إلى مشروعك، وكتابة Hasan ثم نقطة، فإنك ستحصل على:



يعنى أنك ستحصل على جميع طرق الفئة المضافة عند استنساخها إلى كائن جديد، عملية استنساخ الفئة ليست إلا السطر البرمجي الأول من الكود الأخير. عند تحديد الطريقة المطلوبة فإنك ستحصل معها على البارامترات المطلوبة كوسطاء لاستدعاء هذه الطريقة.

ملاحظة

- بإمكانك كتابة الفئة داخل ملف مشروعك وذلك بجوار فئة البرنامج الذي تعمل فيه.

أساساً حتى أنواع البيانات هي بالواقع كائنات، وقد نوهنا إلى ذلك في بداية الكتاب وقلنا أنه لنا وقفة مع أنواع البيانات عند الحديث عن الكائنات.

عند تعريف متغير فإنك بال الواقع تستنسخ فئة تمثل نوع المتغير هذا، أي أن الكودين التاليين متكافئين:

```
string s;
string s = new string();
```

أي أن كل من int و double و string وبقية أنواع البيانات ليست إلا System.Double و System.String و System.Int32 وكلها تتبع للفئة System.Object. وعلى اعتبارها كائنات، بإمكانك استدعاء طرق الكائنات من خلال هذه المتغيرات كما يلي:



```
int s;
s.GetType(); //return System.Int32
s.ToString(); //return a string
s.Equals(m); //return whether s and m are equals AS OBJECTS!
```

المجموعات Assemblies

هو ملف يضم كل ما يوجد في برنامجك من ملفات وفئات وغيرها، ويكون امتداده exe أو dll وذلك تبعا لنوع المشروع الذي تتعامل معه. ويمكنك عرض خصائص المجمع الخاص بمشروعك من Properties الخاص بالمشروع، وفيه بإمكانك ضبط الخصائص الأساسية لمجمع مشروعك وما إلى ذلك مثل نسخة المشروع Version.

مجالات الأسماء Namespaces

مجالات الأسماء هي مجالات تضم العديد من الفئات والأنواع، ولاستخدام فئة ما يجب تضمين مجال الأسماء الذي تتبع له هذه الفئة.

تكلمنا بإيجاز في بداية هذا الكتاب عن مجالات الأسماء دون أن ننوه لها، وقلنا أنه لاستخدام الفئة Console يجب تضمين System في مشروعك. الآن عد إلى ثاني صورة من الفصل صفر ولننكلم عن الموضوع جزءاً جزءاً:

أولاً: يعتبر System هو مجال أسماء، وConsole هو إحدى فئاته، لذلك يجب تضمين System في مشروعك لتتمكن من استخدام طرق وخصائص Console. ويمكن تضمين أي مجال أسماء باستخدام الكلمة using.

ثانياً: إذا لم ترغب في تضمين مجال أسماء ما لسبب من الأسباب - مثل قلة حاجتك له - فإنه عليك كتابة اسم مجال الأسماء أولاً، نقطة، الفئة المطلوبة، نقطة، الخاصية أو الطريقة المطلوبة وهكذا..

ثالثاً: من الممكن لمجال أسماء أي يحتوي على مجال أسماء آخر، مثل System.Text.

رابعاً: هناك مجالات أسماء أخرى لا تدرج افتراضيا مع إنشاء مشروع جديد، ولا يمكنك استخدام بعض الكائنات والفئات لولاهما، في هذه الحالة عليك إضافتها ك System.Windows.Forms.

خامساً: مجال الأسماء هو وحدة برمجية متكاملة مكونة من عدد من الفئات، فإذا اعتبرنا أن العجلة فئة والباب فئة والنافذة فئة و.. إلخ من



العناصر المكونة للسيارات، فيمكن اعتبار السيارة مجال أسماء لأنه يحوي مجموعة من الفئات تؤدي مع بعضها وظيفة معينة كما يمكن استخدام فئة دون أخرى. كما يمكن أن يضم مجال الأسماء مجالاتٍ أسماء أخرى، فبدوره الباب مثلاً مكون من فئة تمثل الزجاج وفئة تمثل المعدن وفئة تمثل المرأة و.. إلخ من العناصر المكونة للباب. كما أن المرأة هي مجال أسماء يمكن محاكاتها بذات الطريقة، فمجالات الأسماء - كما هو حال الفئات - هي تمثيل لكائنات الحياة الواقعية الحية منها وغير الحياة، الملمسة منها والمعنوية، على شكل كائنات برمجية.

إنشاء مجال أسماء

عند إنشاء مشروع ما فإنه سيتم إنشاء مجال أسماء خاص بمشروعك وباسم مشروعك أيضاً، قد تحتاج لإنشاء مجال أسماء تضع فيه بعض الفئات لاستخدامها في أكثر من برنامج، وهي ذات الفكرة التي أنشؤوا من خلالها مجالات الأسماء الجاهزة مثل IO، فمن خلال استخدام فئاتها ستحصل على نتائج أكواد موضوعة داخلها أنت بغني عن الخوض في تفاصيلها طالما أن النتيجة متاحة.

لإنشاء مجال أسماء اكتب الكلمة namespace ثم اسم مجال الأسماء، تليها فئاتها محاطة بإشارتي {} تحديدان بداية ونهاية مجال الأسماء هذا. ولاستخدام مجال الأسماء الذي أنشأته في مجال أسماء آخر عليك تضمينه في بداية البرنامج بالكلمة using.

وعلى سبيل المثال، بإمكانك إنشاء واستخدام مجالات الأسماء وفئاتها كما يلي:

```
using System;
using School; // المجال الخاص بنا
namespace ConsoleApplication11
{
    class Program
    {
        static void Main(string[] args)
        {
            Students Ali = new Students();
            Teachers Khaled = new Teachers();
            Subjects Math = new Subjects();
        }
    }
}
```



```
namespace School
{
    class Students
    {

    }
    class Teachers
    {

    }
    class Subjects
    {

    }
}
```

وبالبرمجة الصحيحة والخوارزميات المتقدمة بإمكانك استخدام مجال الأسماء School السابق على سبيل المثال، في الكثير من البرامج، كما أنه بإمكانك توزيعها على المبرمجين.

والفكرة من هذا مجال أسماء، هو إنشاء كتلة برمجية لمرة واحدة تتضمن أفكاراً وتطبيقاتٍ عديدة، واستخدامها برمجياً في أكثر من برنامج، فعند برمجة طرق وخصائص جميع فئات مجال الأسماء هذا، فإنه يصبح جاهزاً للاستخدام. وعند استنساخ هذه الفئات إلى كائنات في الإجراء main، فإن جميع الطرق التي أُلفتها ستتصبح متاحة لهذه الكائنات، مما يعطي برامج قوية يسهل فيها إصلاح الأخطاء بعد اكتشافها.

مجال الرؤية

ويقصد بمجال الرؤية، إمكانية الوصول إلى الفئات والكائنات، الطرق والخصائص والمتغيرات، سواءً من البرنامج أو الفئة أو الطريقة أو الحلقة حتى.

كل متغير له مجال رؤية محدد، أي إمكانية وصول محددة، بمعنى آخر أنه من خلال مجال الرؤية نحن نعطي الصلاحيات للفئات والطرق الأخرى من البرنامج بالوصول لمتغيراتنا.

ويتم ضبط مجال الرؤية وذلك عند تعريف هذا الكائن أو الطريقة أو المتغير، وذلك بالأقسام التالية:



التعريف الافتراضي

ويحدث عند التعريف بالطريقة العادمة، وفيه تكون الكائنات معروفة فقط على مستوى الوحدة البرمجية التي تنتهي إليها (الفئة أو الطريقة أو البنية التي تم تعريف الكائنات فيها).

public

وهو أشمل أنواع التعريف، وفيه تكون الكائنات معروفة في كافة فئات البرنامج.

private

وهو نفسه التعريف الافتراضي، الكائنات معروفة فقط على مستوى الوحدة البرمجية التي تنتهي إليها.

protected

الكائنات معروفة على مستوى الفئات التي تنتهي إليها والفئات التي ترثها.

internal

الكائنات معروفة على مستوى ملف التجميع assembly فقط.

static

لهذه الكلمة دور هام في مجال رؤية الطرق والمتغيرات، فالمتغير المعرف بنوع ما ومبني على الكلمة static يمكن رؤيته في جميع أجزاء البرنامج والتي تحمل هذه الكلمة بالإضافة لاستخدامه مباشرة دون استنساخ الكائن الحاوي عليه.

خذ المثال التالي عن أسلوب التعريف الافتراضي:

```
int x = 0;
++x; // لا توجد مشكلة
for (int i = 0; i < 10; ++i)
{
    x += i; // لا توجد مشكلة
    int y = x / 10; // لا توجد مشكلة
}
Console.WriteLine(x); // لا توجد مشكلة
Console.WriteLine(y); // هناك مشكلة!!!!!
Console.ReadKey();
```



بإمكانك الوصول لـ x من خلال الحلقة لأنها تنتمي للطريقة التي تم تعريف x فيها، أما y فلا يمكنك الوصول إليها إلا من خلال الحلقة..

التابع البناء Constructor

هو أسلوب برمجي يقوم بإعطاء القيم الابتدائية لعناصر فئة ما، وله نفس اسمها، وهو لا يعد إجراء ولا تابعاً، ولكنه أشبه ما يكون لهما، فهو يستقبل قيمةً لذلك فهو يشبه الطرق، لكنه لا يعيد قيمةً لذلك ليس تابعاً، ولا يعرف باستخدام void لذلك ليس إجراءً أيضاً.

في بداية هذا الفصل وعند تعريفنا للفئات ناقشنا مثلاً عن الفئات، واستخدمنا إجراءً لتعيين قيم الإجراء، وعند استخدام التابع البناء فلا داعي لاستخدام أية طرق وسيطة.

لاحظ نفس المثال عند استخدام التابع البناء، كود الفئة:

```
class Human
{
    string cNAME;
    int cAGE;
    public Human(string Name, int Age)
    {
        cNAME = Name;
        cAGE = Age;
    }
    public void ShowDitails()
    {
        Console.WriteLine ("Name = {0}, Age = {1}",
                           cNAME,
                           cAGE );
    }
}
```

كود البرنامج:

```
int age = 22;
string Name = "Hasan";
Human Hasan = new Human(Name, age); // استنساخ الفئة
Hasan.ShowDitails();
Console.ReadKey();
```

والنتيجة ذاتها..



عند استنساخ كائن ما من إحدى الفئات فإن التابع البناء ينفذ فوراً دون أن تناهيه أصلاً، لذلك ضع الأكواود التي ترغب بتنفيذها عند استنساخ الكائن في التابع البناء الـ `Constructor`.

التابع المدّام Destructor

وهو نظير التابع البناء من حيث المبدأ، إلا أنه يُنفذ عند فناء الكائن. لذلك فعندما ترغب بتنفيذ أكواود معينة عند فناء الكائن - كالخروج من البرنامج مثلاً - فضعها في التابع المدّام والذي يأخذ الشكل التالي:

```
~Human()
{
}
```

الكود الأخير هو تابع هدام لفئة اسمها `Human` (المستخدمة في الأمثلة الأخيرة).

الخصائص Properties

قد تحتاج في بعض الأحيان للوصول إلى خاصية ما لإحدى كائنات مشروعك، وعلى اعتبار أن الكائن ما هو إلا تصغير (أو تخصيص) لمفهوم الفئة، فجميع خصائص الفئة ستُنسخ للكائن.

بمعنى آخر، عند استنساخ فئة (إنشاء كائن)، فإن مكونات ومحتويات هذه الفئة بما فيها الخصائص والتوابع والإجراءات ستنسخ في ذاكرة الحاسوب وتعطى للكائن المستنسخ. وبالعودة للمثال الأول ضمن هذا الفصل وتحديداً في الصورة المرفقة رأينا أنه عند كتابة اسم الكائن ثم نقطة، ستظهر قائمة منبثقة تحوي الطرق الممكن استخدامها، ولكن ماذا عن الخصائص؟؟؟؟ بالتأكيد عندما أنشأنا الفئة صرّحنا عن متغيرات تمثل خصائص الفئة، السؤال هو لماذا لم تظهر؟؟؟؟

في الواقع هذه المتغيرات لا تمثل خصائص الفئة وإنما تمثل متغيرات مستخدمة كوسطاء للحصول على نتائج معينة داخل الفئة ولا يمكن للفئات الأخرى التعامل معها. وفي كثير من الفئات الجاهزة المتوفرة في C# عند استنساخ كائنات منها، وعند كتابة اسم الكائن ثم نقطة فإنك تحصل على عدد من الطرق والخصائص، هذه الخصائص هي تمثيل أكبر لبعض المتغيرات داخل الفئة.



الخلاصة: إذا كانت الأسطر السابقة قد أربكتك فاقرأ ما يلي: في بعض الأحيان قد تحتاج للوصول لبعض متغيرات فئاتك وكائناتها، ويمكن ذلك من خلال استخدام مفهوم الخصائص المتمثل ب getter و setter. والموضحة بالصيغة التالية:

```
اسم_الخاصية نوع_الخاصية
{
set { return متغير; }
get { this.المتغير = value; }
}
```

ملاحظة

- تستخدم الكلمة this في C# للدلالة على الفئة الحالية، قد يكون لديك متغيرات عامة تحمل ذات الاسم في أكثر من فئة، وحتى لا تختلط عليك الأمور بإمكانك كتابة اسم المتغير بعد هذه الكلمة لتحصل على متغير هذه الفئة. (لمبرمجي VB، هذه الكلمة مناظرة للكلمة (Me))
- في أمثلة هذا الفصل استخدمنا الحرف c في بداية متغيرات الفئة للدلالة على أنها تابعة لها، مثل cName.

لنكتب المثال السابق مع إضافة إمكانية وجود خصائص:

```
class Human
{
    private string cNAME;
    private int cAGE;
    public Human()
    {
        // ملاحظة
    }
    public Human(string Name, int Age)
    {
        cNAME = Name;
        cAGE = Age;
        // ملاحظة 2
    }
    public void ShowDitails()
    {
        Console.WriteLine ("Name = {0}, Age = {1}",
                           cNAME,
                           cAGE );
    }
}
```

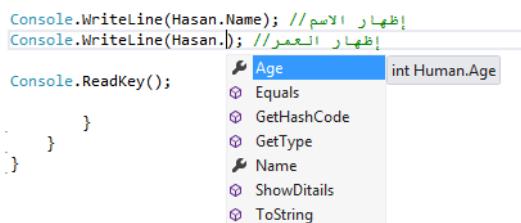


```
public string Name
{
    get { return cNAME; }
    set { cNAME = value; }
}
public int Age
{
    get { return cAGE; }
    set { cAGE = value; }
}
```

* ملاحظة: جرت العادة عند إنشاء فئات، أن يكون التابع البُنَاء عديم الأكواد ولا يحتاج بaramترات، أي أنه يستخدم فقط لاستنساخ الكائن. وفي الغالب يتم وضع تابعين بـnائين - أو يمكن أكثر من ذلك - أحدهما خالي الوسطاء والآخر يحوي وسطاء وعمليات إسناد وذلك ليتمكن المبرمج الذي سيستخدم الفئة من استعمال إحدى الطريقتين على راحته ووفقاً للظروف المارة بين يديه. أما الكود في التابع الرئيسي فهو:

```
استنساخ الفئة //()
إسناد القيم لمتغيرات الكائن //()
Hasan.Name = "Hasan";
Hasan.Age = 22;
اظهار كافة النتائج //()
إظهار الاسم //()
Console.WriteLine(Hasan.Name);
إظهار العمر //()
Console.WriteLine(Hasan.Age);
Console.ReadKey();
```

لاحظ إمكانية الوصول للخصائص من الفئات الأخرى:



* ملاحظة 2: لم أستخدم الكلمة `this` قبل المتغير كما في الصيغة العامة وذلك لأن متغيرات الفئة مُميّزة بالاسم عن متغيرات باقي فئات البرنامج، وبالتالي هذا يعني أنه يمكنني معرفة متغيرات الفئة من سواها بسهولة تامة.



الوراثة Inheritance

هي أحد أهم تطبيقات مبدأ البرمجة كائنية التوجه OOP، وتعني نقل خصائص (صفات) ووظائف (طرق) كائن ما إلى كائن آخر. ومن خلالها تعطيك إمكانية التعامل مع الكائنات البرمجية كما نتعامل مع الكائنات في الواقع، أو بعبارة أخرى: أن تتعامل مع الكائنات من منطق إنساني أكثر من المنطق البرمجي!

يختصر مبدأ الوراثة كتابة الأكواود الكثيرة، فيكتفي كتابة أكواود الفئة الأساسية (وتسمى أيضا الفئة الأم)، ومنها نورث أكواودها للفئات الأبناء.

لأخذ المثال الذي ناقشناه على امتداد هذا الفصل ونطبق عليه مبدأ الوراثة. لدينا فئة أساسية هي إنسان Human، ولتكن لدينا ثلاثة فئات أخرى هي بعض مما يمكن أن يكونه هذا الإنسان وهي موظف، مدرس، ومهندس على سبيل المثال لا الحصر.

في مثال هذا الفصل كان لدينا خصيتان Name و Age وطريقة واحدة ShowDetails، وهي مكونات الفئة Human. عند توريث الفئة Human إلى الفئات Employee و Teacher و Engineer فإن هذه المكونات ستتملكها هذه الفئات أيضا دون الحاجة لكتابتها داخلها، بالإضافة إلى أن لهذه الفئات مكونات خاصة بها.

لاحظ:

المكونات	الفئة
هي فئة أساسية (لا ترث من أحد - فئة أم) جميع الفئات التي سترث من هذه الفئة ستحصل على الحقليين التاليين من الجدول، بالإضافة إلى حقولها. لها الخصائص Name • Age •	Human
لها الطرق ShowDetails •	
ترث من الفئة Human لذلك فلها خصائصها وطرقها. لها الخصائص Institution • StudyLevel • (من الفئة الأم) Name • (من الفئة الأم) Age •	Employee



	لها الطرق Work • GetPromoted • (رشوة D) Bribe • ShowDetails • ترث من الفئة Human لذلك فلها خصائصها وطرقها.	
	لها الخصائص School • TeachLevel • Name (من الفئة الأم) • Age (من الفئة الأم) • Teacher	
	لها الطرق Teach • MakeTest • ShowDeatails • ترث من الفئة Human لذلك فلها خصائصها وطرقها.	
	لها الخصائص Company • Specialty • Name (من الفئة الأم) • Age (من الفئة الأم) • Engineer	
	لها الطرق Work • ShowDetails • ترث من الفئة Human لذلك فلها خصائصها وطرقها.	

الآن سنحول الجدول السابق إلى كود:

```
class Human
{
    string cNAME;
    int cAGE;
    public void Initialize(string Name, int Age)
    {
        cNAME = Name;
        cAGE = Age;
    }
    public void ShowDitails()
    {
        Console.WriteLine ("Name = {0}, Age = {1}",
                           cNAME,
                           cAGE );
    }
}
```



```
}  
class Teacher : Human  
{  
    string cSchool, cTeachLevel;  
    public Teacher(string School, string TeachLevel)  
    {  
        //This is Constructor!  
        cSchool = School;  
        cTeachLevel = TeachLevel;  
    }  
    public void Teach()  
    {  
        //C#'s Codes!  
    }  
    public void MakeTest()  
    {  
        //C#'s Codes!  
    }  
}  
class Employee : Human  
{  
    string cInstitution, cStudyLevel;  
    public Employee(string Institution, string StudyLevel)  
    {  
        //This is Constructor!  
        cInstitution = Institution;  
        cStudyLevel = StudyLevel;  
    }  
    public void Work()  
    {  
        //C#'s Codes!  
    }  
    public void Bribe()  
    {  
        //C#'s Codes!  
    }  
    public void GetPromoted()  
    {  
        //C#'s Codes!  
    }  
}  
class Engineer : Human  
{  
    string cCompany, cSpecialty;  
    public Engineer()  
    {}  
    public Engineer(string Company, string Specialty)
```



```

    {
        //This is Constructor!
        cCompany = Company;
        cSpecialty = Specialty;
    }
    public void Work()
    {
        //C#'s Codes!
    }
}
class Program
{
    static void Main(string[] args)
    {
        string Name = "Hasan";
        int Age = 22;
        string Company = "None";
        string Speciality = "Mechanical";
        Engineer Hasan = new Engineer(Company, Speciality);
        Hasan.Initialize(Name, Age);
        Hasan.ShowDitails();
        Hasan.Work();
        Console.ReadKey();
    }
}

```

لتوريث فئة أخرى نستخدم الرمز ":" للربط بين الفئة الوارثة والفئة الموروث منها، بحيث تكون الوارثة في البداية.

لا يختلف إنشاء وتكوين الفئات العادية سواءً أكانت سترث من فئات أخرى أم لا، الفارق ستجده عند استخدام هذه الفئات في فئات أخرى، وسيتجلى هذا الفارق في أن محتويات الفئة ومكوناتها ستضم طرق وخصائص أكثر.

يفضل دائماً وضع أكثر من تابع بناءً، وذلك لنسمح للمبرمج الذي سيستخدم فئتنا من إدخال وسطاء تعريف الفئة عند استنساخ كائن، أو استنساخ كائن فقط دون إدخال الوسطاء. لاحظ الصورة التالية: عند استنساخ كائن Hasan من الفئة Engineer فإنه لديك خيارات (لاحظ مكتوب 2 of 1)، ومن أجل ذلك أنشأنا تابعين بثنائيين. في بعض الفئات التي تأتيك هدية مع لغة البرمجة – أقصد الفئات الجاهزة – قد تصل الخيارات إلى العشرات!



لكن ضع في ذهنك، عندما تستخدم تابعاً بناءً خالياً الوسطاء مثل الذي في الصورة، عليك استخدام مفهوم الخصائص Properties. وهذا مالم يوجد في المثال الأخير لذلك تم استخدام التابع البناة الآخر والذي يلزم المبرمج إدخال الوسطاء.

```
Hasan.Initialize();
Hasan.ShowDetails();
Hasan.Work();
Console.ReadKey();
```

الفئات المجردة abstract

الفئات المجردة هي فئات غير موجودة على أرض الواقع وإنما تمثل مفهوماً عاماً لشيء ما، وهي فئات لا يمكن استنساخ كائنات منها.

قلنا أن البرمجة كائية التوجه أنت لتجعل من مفهوم البرمجة أقرب ما يكون للحياة الواقعية، والتي يعتبر كل مافيها كائنات. في الحياة الواقعية هناك كائنات غير موجودة فعلياً لكنها تعتبر نوعاً لكائنات أخرى، فوسائل النقل هي كائن لكنه غير موجود، أما السيارات والطائرات والships والقطارات هي كائنات مشتقة من الكائن وسيطة نقل ولها مميزات خاصة بها تميزها عن الكائنات الأخرى من نفس النوع. معنى أنه لا يمكنك أن ترى وسيطة نقل أو تستخدمها لكن بإمكانك استخدام ورثية السيارات، وسيطة النقل عبارة عن فكرة مجردة أما السيارة فهي كائن موجود!

للإعلان عن فئة على أنها مجردة نستخدم الكلمة `abstract`، وكما اتفقنا فالفئات المجردة لا يمكن استنساخ كائنات منها، لكن يبقى لك حرية استنساخ كائنات من فئات وارثة منها فلهذه الفئات إمكانية الوراثة، وبطبيعة الحال فالفئات الوراثة تملك جميع مكونات الفئات الرئيسية.

```
abstract class Transport
{
    C#'s Codes
}
class Car
{
    C#'s Codes
}
class Bus
{
    C#'s Codes
}
```



```
class Program
{
    static void Main(string[] args)
    {
        //Transport Kia = new Transport(); خطأ!
        Car Kia = new Car(); //() ممكن
    }
}
```

الفئات المغلقة sealed

الفئات المغلقة هي عكس الفئات المجردة تماماً فهي فئات لا يمكن الوراثة منها، أما الفئات المجردة فال فكرة منها هي إمكانية وراثتها ولو لا الوراثة لما كان لها معنى أو داعٍ.. أما فكرة وجود الفئات المغلقة: أنها عقيمة.

الوظائف¹ الوهمية virtual Methods

قد تحتوي بعض الفئات على طرق تقوم بوظائف معينة، لكن هذه الوظائف تكون بصيغة عامة. وعند توريث هذه الفئات إلى فئات أخرى قد تحتاج لتخصيص إحدى الطرق، كإعطاء تفاصيل إضافية أو تغيير محتوى الطريقة كل. لجعل طريقة ما وهمية استخدم الكلمة virtual عند التصريح عنها، وللتعديل عليها في الفئات الوارثة استخدم إحدى الكلمتين new أو override. وللنفوذ لطرق الفئة الرئيسية يمكن استخدام الكلمة base.

```
class Human
{
    virtual public void Job()
    {
        Console.Write("I work, ");
    }
}

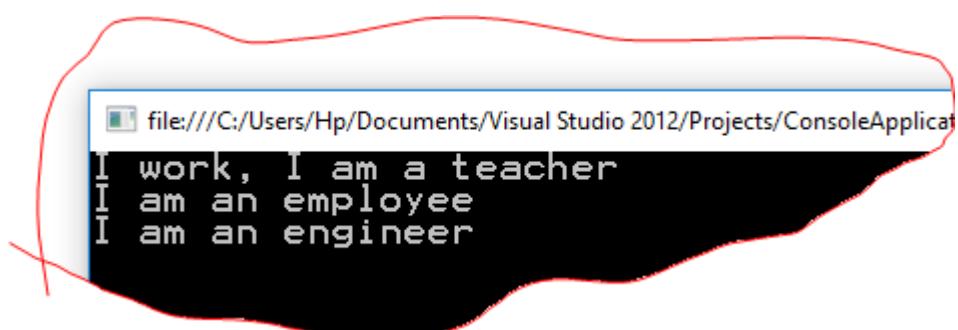
class Teacher : Human
{
    public override void Job()
    {
        base.Job();
        Console.WriteLine("I am a teacher");
    }
}
```

¹ الوظيفة هي تعبير آخر للطريقة.



```
}  
  
class Employee : Human  
{  
    public override void Job()  
    {  
        //base.Job();  
        Console.WriteLine("I am an employee");  
    }  
}  
  
class Engineer : Human  
{  
    new public void Job()  
    {  
        Console.WriteLine("I am an engineer");  
    }  
}  
  
class Program  
{  
    static void Main(string[] args)  
    {  
        Human Ahmad = new Teacher();  
        Human Ali = new Employee();  
        Engineer Khaled = new Engineer();  
  
        Ahmad.Job();  
        Ali.Job();  
        Khaled.Job();  
  
        Console.ReadKey();  
    }  
}
```

والنتيجة ستكون كما يلي:





ملاحظة

- عند اشتقاء فئة من فئة أخرى (الوراثة منها)، فإنه يمكن الوصول للفئة الرئيسية عن طريق الكلمة `base`. فكما أن الكلمة `this` تمثل الفئة الحالية، فإن الكلمة `base` تمثل الفئة الرئيسية.

تعدد التعاريف Overloading

يمكن لبعض الطرق أن تأتي بأكثر من صيغة وأن تعامل مع أكثر من حالة، بمعنى أنها تأخذ أكثر من حالة للوسطاء. مر معنا مفهوم تعدد التعاريف كثيراً لكن لم ننوه له، وهذه الفقرة الأنسب لذلك. وقد تستغرب إذا أخبرتك بالتفاصيل، فأنت تستخدم هذا المفهوم يومياً في البرمجة!

```
Console.WriteLine (
    ▲ 1 of 19 ▼ void Console.WriteLine()
    Writes the current line terminator to the standard output stream.
```

الصورة تقول أن الطريقة `WriteLine` يمكن أن تجدها بتسعة عشرة حالة، أي أن مبرمجها قام بكتابتها 19 مرة لتصل إلى يديك بالشكل الحالي!

الجيد في تعدد التعاريف أن المترجم يفهم الحالة التي تحتاج التعامل معها من خلال نوع المتغيرات الذي ترسله عند نداء الطريقة، وليس هناك حاجة للتقليل بين الحالات الـ 19 السابقة الممكنة للوصول على الحالة المطلوبة، لذلك فأنت تدين للمترجم بالكثير. هل تذكر المترجم في بداياتك مع الكتاب ؟

الآن لنكتب مثلاً لإيضاح الفكرة أكثر:

```
class Human
{
    string cNAME;
    int cAGE;
    //Constructor overloading
    public Human(string Name, int Age)
    {
        cNAME = Name;
        cAGE = Age;
    }
    public Human()
    {
    }
```



```
//Method overloading
public void AddFriend()
{
    //C#'s Codes
}
public void AddFriend(int Age)
{
    //C#'s Codes
}
public void AddFriend(string Name)
{
    //C#'s Codes
}

class Program
{
    static void Main(string[] args)
    {
        Human Ahmad = new Human("Ahmad alAari", 20);
        Ahmad.AddFriend("Eng27");
        Console.ReadKey();
    }
}
```

لاحظ كيف استنسخنا كائناً بالتتابع البناء الثاني، ممررين الاسم وال عمر.

Human Ahmad = new Human("Ahmad alAari", 20)
▲ 2 of 2 ▼ Human.Human(string Name, int Age)

وهنا استدعيينا الطريقة AddFriend وذلك باستخدام الطريقة الثالثة،
ممررين متغيراً نصياً.

Ahmad.AddFriend ("Eng27")
▲ 3 of 3 ▼ void Human.AddFriend(string Name)

تعدد الأشكال Polymorphism

يشابه هذا المفهوم مفهوم تعدد التعريف كثيراً، ويعطيك إمكانية التعامل مع الفئات المتراثة كما لو أنها واحدة..



```

class Human
{
    public Human()
    {
        //C#'s Codes
    }

    public void AddFriend()
    {
        //C#'s Codes
    }
}

class Teacher : Human
{
    //C#'s Codes
}

class Program
{
    static void Work(Human h)
    {
        h.AddFriend();
    }

    static void Main(string[] args)
    {
        Teacher Ahmad = new Teacher();
        Work(Ahmad);
        Console.ReadKey();
    }
}

```

لاحظ أننا أرسلنا الكائن ك وسيط للإجراء Work، ووفق هذا المفهوم في إمكانك تمرير أي فئة مشتقة من الفئة الرئيسية Human ك وسيط.

الواجهات Interfaces

تدعم بعض لغات البرمجة مبدأ الوراثة المتعددة، أي إمكانية وراثة فئة ما لأكثر من فئة أخرى. في C# لا يمكن للفئة أن ترث إلا من فئة واحدة.. وبالتالي للوراثة من أكثر من فئة فنستخدم مفهوم الواجهات.

يشابه مفهوم الواجهات بشكل كبير مفهوم الفئات المجردة، والتي لا يمكن استساغ كائنات منها، بالإضافة لعدم إمكانية كتابة أي كود عملي داخل الواجهات وإنما فقط الوظائف والخصائص التي ستستعمل هذه الواجهة.



يمكن لفئة واحدة أن ترث من أكثر من واجهة، وهو الأمر غير الممكن بالنسبة للوراثة من الفئات، وتسمى عملية الوراثة من واجهة بـ Implementation.

لاحظ المثال:

```
interface Human
{
    int Age { get; set; }
    string Name { get; set; }
    int BirthYear();
}

public Teacher()
{
}

class Teacher : Human
{
    private string cNAME;
    private int cAGE;
    public Teacher(string Name, int Age)
    {
        cNAME = Name;
        cAGE = Age;
    }

    int Human.Age
    {
        get { return cAGE; }
        set { cAGE = value; }
    }

    string Human.Name
    {
        get { return cNAME; }
        set { cNAME = value; }
    }

    int Human.BirthYear()
    {
        return DateTime.Now.Year - cAGE;
    }
}

class Program
{
    static void Main(string[] args)
    {
```



```

Human Hasan = new Teacher();
Hasan.Name = "Hasan M. alFahl";
Hasan.Age = 22;
Console.WriteLine("Year of Birth is {0}", Hasan.BirthYear());
Console.ReadKey();
}
}

```

المفوّضات Delegates

وتسّمى لغةً بالمندوبات، والمندوب هو النائب أو الممثل. ومن خلالها يمكن أن ينوب عن فئة ما مفوض أو مندوب، بحيث يتعامل مع وظيفة ما من وظائف الفئةشرط أن تكون من نفس نوعه. تستخدّم المفوّضات بشكل واسع للتعامل مع الأحداث، وهو ما سنراه لاحقاً بإذن الله. لاحظ المثال:

```

class Numbers
{
    public string IsEven(int Number)
    {
        return (Number % 2 == 0 ? "Even" : "Odd");
    }

    public string PrimeFactors(int Number)
    {
        int d = 2;
        string factors = Number + " = ";
        while (Number > 1)
        {
            while (Number % d == 0)
            {
                factors += d;
                factors += "x";
                Number /= d;
            }
            d++;
        }
        factors += "1";
        return factors;
    }
}

class Program
{
    public delegate string NumCase(int Num); // التصريح عن مندوب
}

```



```

static void Main(string[] args)
{
    Numbers N = new Numbers(); // استنساخ كائن
    NumCase Num = new NumCase(N.isEven); // إنشاء مفوض

    Console.WriteLine("Enter Number : ");
    int number = int.Parse(Console.ReadLine());
    Console.WriteLine("{0} is {1}", number, Num(number));

    Num = new NumCase(N.PrimeFactors); // التفويض بتتابع آخر من توابع الفئة
    Console.WriteLine(Num(number));

    Console.ReadKey();
}

```

لاحظ أن المفوض هو كائن، جعلناه نائباً عن الكائن N للتعامل مع طرقها.

التفويض المتعدد

أعتقد أنك لاحظت أن عملية التفويض غير مجديّة عند التعامل مع توابع وإجراءات كثيرة، فكل وظيفة عليك تجديد المفوض أو إنشاء واحد جديد.

```

class Numbers
{
    public void isEven(int Number)
    {
        Console.WriteLine ( Number % 2 == 0 ? "Even" : "Odd");
    }
    public void PrimeFactors(int Number)
    {
        int d = 2;
        string factors = Number + " = ";
        while (Number > 1)
        {
            while (Number % d == 0)
            {
                factors += d;
                factors += "x";
                Number /= d;
            }
            d++;
        }
        factors += "1";
        Console.WriteLine ( factors);
    }
}

```



```

    }
}
class Program
{
    public delegate void NumCase(int Num);
    static void Main(string[] args)
    {
        Numbers N = new Numbers();
        NumCase Num1 = new NumCase(N.isEven);
        NumCase Num2 = new NumCase(N.PrimeFactors);
        Console.Write("Enter Number : ");
        int number = int.Parse(Console.ReadLine());
        Console.Write("{0} is ",number);
        NumCase Multi = Num1 + Num2;
        Multi(number);
        Console.ReadKey();
    }
}

```

الأحداث Events

الأحداث هي المجريات والمشاعر والأحساس التي تمر على برنامجك، فضغط الأزرار يفجر حدثاً، واتصال جهاز بالكمبيوتر يفجر حدثاً، ووصول التركيز لأحد الأدوات في النموذج يفجر حدثاً، كما أن إدخال الحروف من لوحة المفاتيح يفجر حدثاً.. وهناك الكثير والكثير من الأحداث التي ستترافق عليك في حياتك البرمجية عند برمجتك للتطبيقات الحاوية على نوافذ..

ملاحظة

- مصطلح تفجير الحدث يقصد به حدوث الحدث، وهي ترجمة حرفية للمصطلح الأجنبي.. فعندما نقول تم تفجير الحدث الفلاني فإننا نقصد أنه تم حدوثه.

إن مبدأ عمل الأحداث هو استدعاء طريقة ما عند تفجير حدث ما، لذلك فتعاملنا سيكون مع الطرق في نهاية الأمر وهذه الطرق الخاصة بالأحداث تسمى Handlers. كما أن للأحداث علاقة وطيدة مع المفروضات، فكما قلنا فالمفروضات تستعمل للتعامل مع الأحداث.. لذلك فعليك إنشاء مفروضات ستتربّع على الأحداث وتعامل معها.

لاحظ المثال:



التصريح عن مفوض//

```
public delegate void delegEvent(); // التصريح عن مفوض//
```

```
class Store
{
    public event delegEvent Selling; // التصريح عن حدث جديد//
```

```
    private string className;
    public string Name
    {
        get { return className; }
        set { className = value; }
    }
```

```
    private int classQuantity;
    public int Quantity
    {
        get { return classQuantity; }
        set { classQuantity = value; }
    }
```

```
    public Store(string Name, int Mark)
    {
        className = Name;
        classQuantity = Mark;
    }
    public Store()
    {
```

```
    public void Sell(int Number)
    {
        if (Number > Quantity) Selling(); // تفجير الحدث// في حال لم يتفجر الحدث يتم انقاص الكمية// else
        else Quantity -= Number;
    }
```

```
    public void SoldError()
    {
        Console.WriteLine("The quantity is not enough!");
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Store Product1 = new Store();
```



```

//ربط الحدث بالإجراء وذلك عن طريق المفهوم //
Product1.Selling += new delegEvent(Product1.SoldError);

Product1.Name = "C# Book";
Product1.Quantity = 100;

Console.WriteLine("Enter number of items to sell : ");
int Number = int.Parse(Console.ReadLine());

Product1.Sell(Number); //عملية البيع
Console.WriteLine("You Have {0} of {1}",
    Product1.Quantity,
    Product1.Name );
Console.ReadKey();
}
}

```

لا تنسَ أن الحدث Selling سينفجّر في حال تحقق الشرط، وعندها سيتم تنفيذ الـHandler الخاص به، وهو الإجراء .SoldError.

الوظائف المجهولة Anonymous Methods

يسمح لك هذا المفهوم بإنشاء الوظائف عند تفجير الحدث ومن ثم تمرييرها للمفهوم في نفس الوقت، قارن المثال السابق مع هذا المفهوم:

```

public delegate void delegEvent(); // التصريح عن مفهوم

class Store
{
    public event delegEvent Selling; // التصريح عن حدث جديد

    private string className;
    public string Name
    {
        get { return className; }
        set { className = value; }
    }

    private int classQuantity;
    public int Quantity
    {
        get { return classQuantity; }
        set { classQuantity = value; }
    }
}

```



```

public Store(string Name, int Mark)
{
    className = Name;
    classQuantity = Mark;
}
public Store()
{
}

public void Sell(int Number)
{
    if (Number > Quantity) Selling();
    else Quantity -= Number;
}
}

class Program
{
    static void Main(string[] args)
    {
        Store Product1 = new Store();
        //ربط الحدث بالإجراء وذلك عن طريق المفهوم
        Product1.Selling += delegate()
        {
            Console.WriteLine("The quantity is not enough!");
        };

        Product1.Name = "C# Book";
        Product1.Quantity = 100;

        Console.Write("Enter number of items to sell : ");
        int Number = int.Parse(Console.ReadLine());

        Product1.Sell(Number); //عملية البيع
        Console.WriteLine("You Have {0} of {1}",
            Product1.Quantity,
            Product1.Name );
        Console.ReadKey();
    }
}

```

العبارة لاما | Lambda expression

لا تختلف كثيراً العبارة لاما عن الوظائف المجهولة إلا بالشكل، إلا أنها تعامل مع اللوائح بشكل جيد. لاحظ:



```

public delegate void delegEvent(); // مفهوم التصريح عن

class Store
{
    public event delegEvent Selling; // التصريح عن حدث جديد

    private string className;
    public string Name
    {
        get { return className; }
        set { className = value; }
    }

    private int classQuantity;
    public int Quantity
    {
        get { return classQuantity; }
        set { classQuantity = value; }
    }

    public Store(string Name, int Mark)
    { className = Name; classQuantity = Mark; }
    public Store()
    {
    }
    public void Sell(int Number)
    {
        if (Number > Quantity) Selling();
        else Quantity -= Number;
    }
}

class Program
{
    static void Main(string[] args)
    {
        Store Product1 = new Store();
        // ربط الحدث بالإجراء وذلك عن طريق المفهوم
        Product1.Selling += () =>
        {
            Console.WriteLine("The quantity is not enough!");
        };

        Product1.Name = "C# Book";
        Product1.Quantity = 100;
        Console.Write("Enter number of items to sell : ");
        int Number = int.Parse(Console.ReadLine());
    }
}

```



```
Product1.Sell(Number); //عملية البيع
Console.WriteLine("You Have {0} of {1}",
    Product1.Quantity,
    Product1.Name );
Console.ReadKey();
}
}
```

وبالنسبة للوائح:

```
List<string> Employees = new List<string>
{
    "Ahmad", "Ali", "Khaled", "Zaid"};
Console.WriteLine("Employees are:");
Employees.ForEach(name =>
{
    Console.WriteLine("- {0}", name);
});
Console.ReadKey();
```

مكتبات الارتباط الحيوي DLL

تلعب مكتبات الارتباط الحيوي Dynamic Link Libraries دورا هاما في حياة برمج وتطبيقات ويندوز، وعند استخدامك للطرق الجاهزة في C# - أو أي لغة برمجة أخرى - فإنك تستخدم مكتبة DLL موجودة على كمبيوترك.



إذهب إلى متصفح ملفات ويندوز وانتقل إلى مجلد Windows في قرص النظام، وابحث عن الكلمة ".dll". وتأمل النتائج. ستحصل على آلاف الملفات والتي بلا شك لم تأتِ بالصدفة إلى هنا، ولولا هذه الملفات لما استطاع نظام التشغيل من العمل وما تمكنت برامحك -

التي أنشأتها أنت أو التي حصلت عليها من الإنترنت أو من مكان ما - من استخدام الكثير من الطرق والأدوات الجاهزة، فالأدوات التي تجدها على البرامج والتطبيقات مثل الأزرار وصناديق النصوص والمؤقتات وغيرها ماهي إلا ملفات صممت من قبل شركات أو أشخاص ووصلت إلى كمبيوترك أثناء تنزيل نظام التشغيل أو تثبيت برنامج جديد، والطرق التي تستخدمنها للحصول على نتائج ما مثل Math التابعة للفئة Show أو Console التابعة



للفئة MessageBox ماهي إلا طرق مكتوبة في فئات محفوظة بملف موجود على كمبيوترك يستفيد منها برنامجك ولا تعلم ما فيها أصلا، وبشكل عام أنت بغضن عن معرفة كيفية عملها، المهم أنها تعطيك نتائج ترضيك في برامحك..

لكن بالمقابل قد لا تشبع رغباتك الفئات الجاهزة، لذلك فأنت مضطرك لإنشاء فئات خاصة بك كما تعلمتها في فقرات هذا الفصل. وعندما تكون هذه الفئات مطلوبة في برامحك بشكل كبير فإنك تحتاج لنسخ الكود كاملاً من أحد المشاريع القديمة والتي تحتوي عليه ووضعها في مشروع جديد لك. وعواضاً عن عملية نسخ الكود والعمل غير المنظم، بإمكانك وضع هذه الفئات في ملف ما يسمى مكتبة الارتباط الحيوي، وإسناده لمراجع مشاريعك في كل مشروع تحتاجه فيه، وهكذا في حال تعديلك على مكتبتك، فإن جميع مشاريعك التي تعتمد على هذه المكتبة ستحصل على التعديل الجديد.

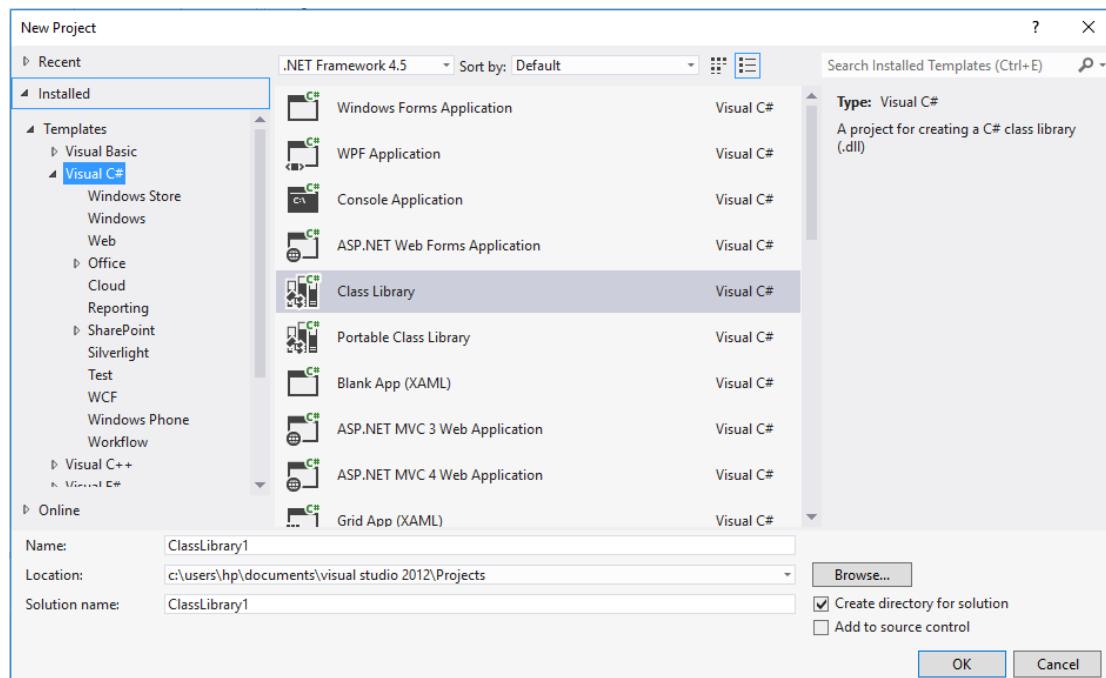
وفضلاً عن هذا كله، فاستخدام مكتبة `dll` لوضع أكوادك الخاصة فيها - المتمثلة بالفئات الخاصة بك - يجعل برامحك أكثر احترافية، فكلما كان برامحك مجزءاً أكثر، كلما قل احتمال وجود أخطاء فيه لأن الخطأ من الممكن اكتشافه بسهولة ويسراً في حال البرامج المجزأة. وهذا ماتسعى إليه البرمجة كائنية التوجه.

في مصطلحات عالم البرمجة، "المكتبة" هي مجموعة من الإجراءات البرمجية المستقلة التي يمكن استدعاؤها من قبل البرنامج لتنفيذ وظائف تكمل عملها. ويحتوي ملف المكتبة نصوصاً برمجية لأداء عمليات مختلفة على النظام، إلى جانب متغيرات تمثل بيانات أو بعض مكونات النظام. وهي محتويات يتم استخدامهما من قبل البرامج العاملة. هذا الأمر يسمح بمشاركة نفس الوظائف والمكونات البرمجية بين أكثر من تطبيق، ويزيد من حرية القيام بتعديلها، لوجودها في ملفات خارجية منفصلة عن ملفات البرامج الأساسية. أغلب المكتبات لا يمكنها العمل كبرامج تنفيذية، ولكن البعض منها لا يختلف عن البرامج التنفيذية في شيء، فيتم تشغيلها بصورة ذاتية مستقلة. كل من البرامج التنفيذية والمكتبات تصنف روابط مرجعية فيما بينها، يتم ذلك في مرحلة تجميع مكونات البرنامج، التي ينفذها برنامج الرابط.



إنشاء مشروع dll

أنشئ مشروعًا جديداً وهذه المرة Class Library، وكما في مشاريع ClassLibrary1 فإن اسم المشروع سيكون بشكل افتراضي Console ويزداد الرقم تلقائيًا مع كل مشروع جديد.



ستحصل على هذه الأكواد بشكل تلقائي (بتنذكر أيام الفصل صفر؟):

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ClassLibrary1
{
    public class Class1
    {
    }
}

```

في مشاريع dll تهمنا أسماء الفئات (على عكس مشاريع Console)، لأن اسم الفئة هنا سنجده في مشاريعنا العادية، تخيل استخدام مكتبة فيها فئات، Class1 و Class2 و Class3 بدل Add و Subtract و Multiple ؟! عندما تشاهد الأسماء الافتراضية ستشعر أن لا نفس لك للبرمجة وأن الموضوع مائع زيادة..



الآن أنت أصبحت كبيراً ولا داعي لأن أقول لك أن المشروع هنا عبارة عن قسمين، الأول مكتبات والثاني مجال أسماء خاص بمشروعك وأن القسم الأول ستضيف إليه مجالات الأسماء التي ستتعامل معها – وغير موجودة ضمن مجالات الأسماء الافتراضية – وأن القسم الثاني سنضع فيه الفئات الخاصة بنا وداخلها طرقها وخصائصها وأكوادها، أليس كذلك؟؟؟

لنشئ مكتبةً كمثال، نقوم فيها بحل معادلة رياضية من الدرجة الأولى:

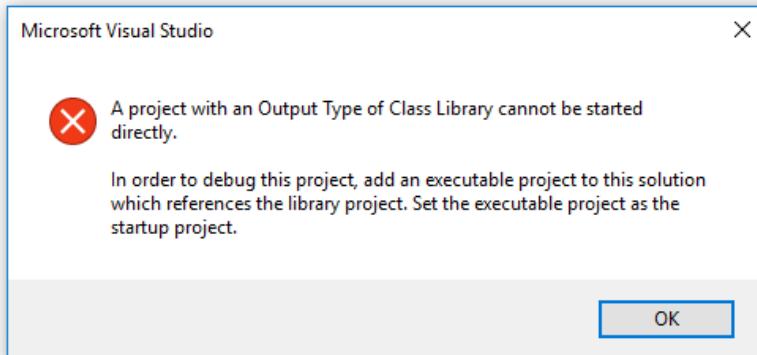
$$\text{الحل فيها من الشكل } x = \frac{-B}{A}$$

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ClassLibrary1
{
    public class Equation1
    {
        private double cA, cB;
        public Equation1()
        {
        }

        public Equation1(double a, double b)
        { cA = a; cB = b; }
        public double A
        {
            get { return cA; }
            set { cA = value; }
        }
        public double B
        {
            get { return cB; }
            set { cB = value; }
        }
        public double Solve ()
        { return (-cB / cA); }
    }
}
```

إذا حاولت تنفيذ المشروع ستحصل على الرسالة التالية:

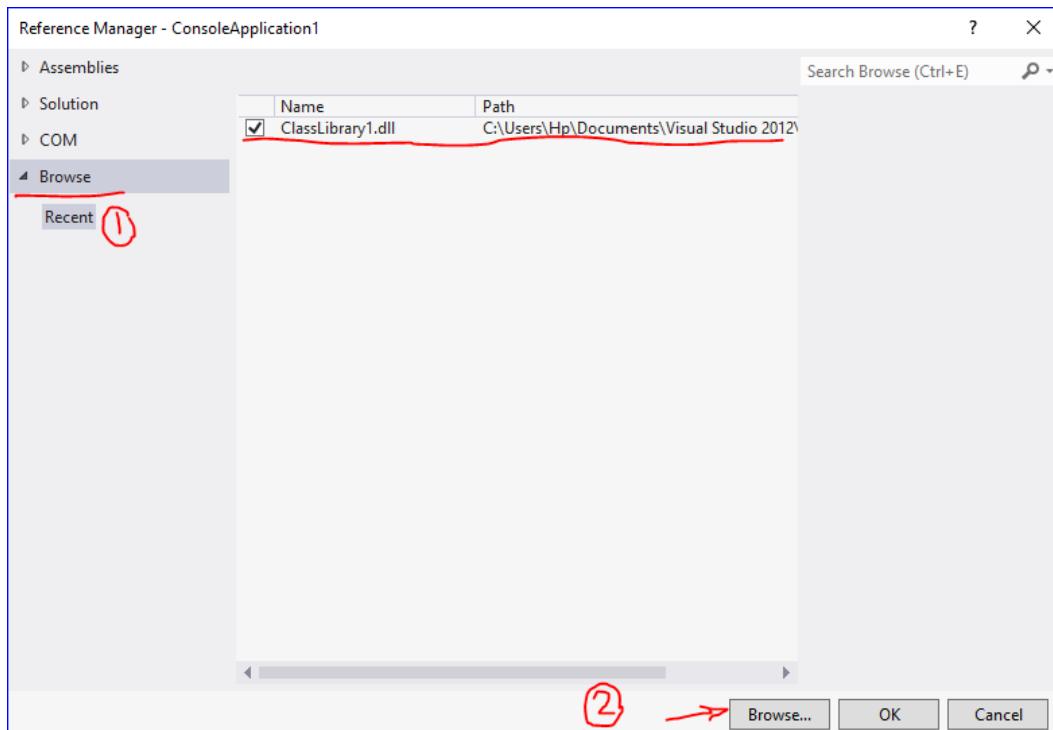


ومفادها أنه لا يمكن تنفيذ هذا المشروع إلا من خلال إضافة مشروع تنفيذي واستيراد هذا المشروع إلى قائمة المراجع الخاصة بالمشروع التنفيذي..

إضافة المكتبة إلى مراجع المشروع

لإنشاء مشروع تنفيذيا من نوع Console ونستورد هذا المشروع كمراجع للمشروع التنفيذي.

من القائمة Solution Explorer وبالضغط على References بالزر الأيمن، ثم ابحث عن مكتبتنا في القرص الصلب ثم أضافها.



ثم استخدم الكود التالي:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
استيراد المكتبة برمجيا
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Equation1 Equ = new Equation1(5, 6); //Means 5x + 6 = 0
            Console.WriteLine("x = " + Equ.Solve().ToString()); //Prints x = -1.2
            Console.ReadKey();
        }
    }
}
```

أو من الممكن أن يكون الكود بالشكل التالي:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ClassLibrary1;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Equation1 Equ = new Equation1();
            Equ.A = 5;
            Equ.B = 6;
            Console.WriteLine("x = " + Equ.Solve().ToString());//Prints x = -1.2
            Console.ReadKey();
        }
    }
}
```



مكتبات API الجاهزة

من المحتمل أنك قد سمعت بهذا المصطلح بشكل واسع عند تسركعك من منتدى برمجي لآخر ومن موقع لآخر، والسؤال هنا ماهي هذه المكتبات API الجاهزة؟؟

في بداية الكتاب وعند سرد تاريخ البرمجة تم ذكر أن شركة ميكروسوفت قد طرحت مكتبات API مجانية لجميع المبرمجين لجذب انتباهم وكسب ولائهم، فلا أعتقد - ولا هم يعتقدون - أن مبرمجا سيفوت عليه فرصة الحصول على سر مصلحة شركة عريقة لاستخدامه - سر المصلحة - في برامجه كما يحلو له وبالشكل الذي يريد.

مكتبات APIs أو Application Programming interface هي مجموعة من هياكل البيانات والفنات وغيرها من المكونات البرمجية التي تقدمها مكتبات محددة بالتعاون مع نظام التشغيل لدعم بناء البرامج.

وتقوم الشركات عادةً بنشر مكتبات APIs خاصة بها كدعاية على منتجاتها، وأنت باستخدامك لهذه المكتبات تستطيع تطوير برامحك بناءً عليها بالشكل الذي ترغب به، أي أنك تبدأ من حيث انتهى الآخرون، بدلاً من البدء بالبرمجة من الصفر.





الجزء الثاني

Windows Form Applications





الفصل الثامن - أساسيات 2

أهلا بك بالجزء الثاني من هذا الكتاب، والذي يهتم بالمشاريع التنفيذية بشكل خاص، النوافذ، مكتبات الارتباط الحيوي `AIO` وقواعد البيانات بإيجاز. لن يحتوي هذا الجزء على قواعد البرمجة، وإنما سيعتمد على محتويات الجزء الأول. كما أنه لن يحتوي على شرح لأدوات ونوافذ بيئة التطوير الخاصة بـ C#, وإنما سيعتبرك تعرفها أو على اطلاع عليها¹.

قد يتم إيجاز بعض الأمور الهامة المتعلقة بالأدوات المستخدمة في تصميم مشاريع الكتاب والتي يجب الإلمام بها أو الاطلاع عليها بشكل جيد. ولمن يحتاج المزيد من الشرح والإيضاح والتفصيل فهناك العديد من الكتب والفيديوهات على الانترنت تهتم بهذا الموضوع، ولا أرىفائدة من إنشاء كتابٍ يوجد ما يغني عنه، وعوضاً عن ذلك فمن الجيد - كفكرة - أن يتتوفر كتابٌ يشرح مشاريع تطبيقية ويحاول أن يكون مكتبة مقرؤة لمشاريع مفتوحة المصدر، وهذا ما ينوي الكتاب أن يحويه.

وكبداية سأورد باختصار تفاصيل بيئة التطوير - أو قد تسمى بيئة التصميم - الخاصة بلغتنا، وهي كما يلي:

Solution Explorer

وفيه يتم عرض جميع ملفات المشروع ومصادره وخصائصه، ويمكنك من الوصول إلى جميع طرقه وكائناته.

¹ راجع كتاب "سييلك المختصر إلى تعلم لغة C#.Net، برمجة الواجهات" لخالد السعداني، فقد كفّي ووفى، في كتابه شرحاً عن أكثر الأدوات التي ستستخدمها في برامحك، ليغطي 36 أداة شائعة الاستخدام، جزاه خيراً ❤.



صندوق الأدوات ToolBox

وفيه توجد الأدوات التي ستستخدمها لتصميم مشروعك، وهي مرتبة وفق مجموعات.

الخصائص Properties

وفيه تُعرض جميع خصائص الأدوات المستخدمة في المشروع، والخصائص هي الصفات والميزات التي تتسم بها كل أداة. كما يمكن من خلال صندوق الخصائص الولوج إلى أحداث الأدوات.

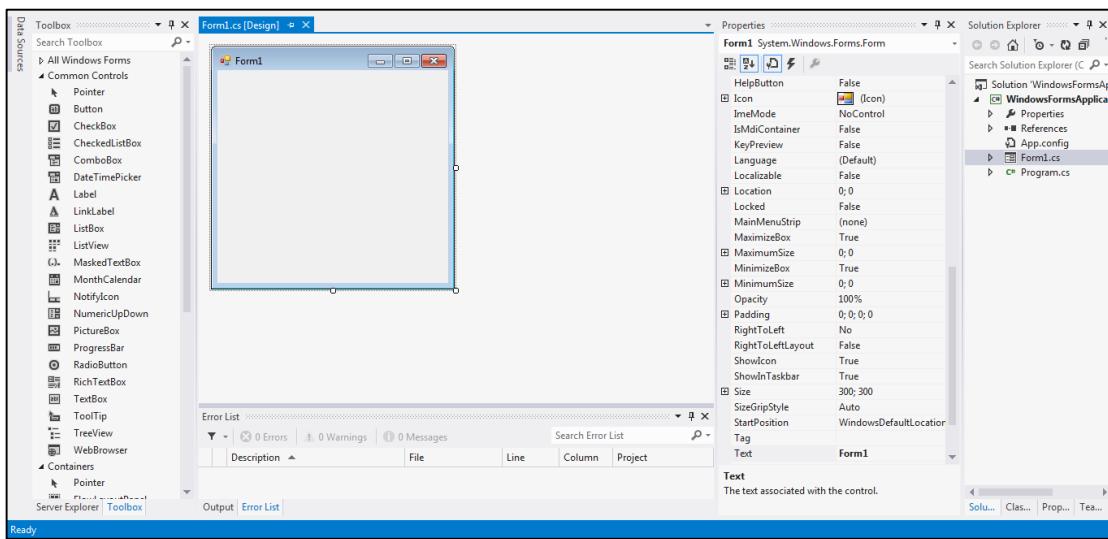
قائمة الأخطاء Error List

و فيها يتم عرض أخطائك البرمجية لتمكن من معالجتها، حيث أنك ستتعامل مع هذه القائمة أكثر من أي قائمة شاهدتها في حياتك!

النموذج Form

وهو عبارة عن إطار (أو نافذة) تضع عليه الأدوات التي تختارها لأداء وظائف معينة، وفق ترتيب معين. وتسمى عملية انتقاء الأدوات وترتيبها بـ تصميم النافذة أو المشروع أو النموذج.

الصورة التالية توضح بيئه التطوير التابعة لـ C#.



محرر الأكواد

وهو عبارة عن محرر نصوص يقوم باستقبال أكوادك ومعالجتها وتنقيحها، ويمكن الوصول إليه عبر القائمة View، ثم الأمر Code. أو بالضغط على أي أداة لتنقل إلى محرر الأكواد وتحديداً في الإجراء الافتراضي لهذه الأداة.



ملاحظة

- عند الضغط على أداة ما وكانت تحوي كوداً تحت أحد إجراءاتها، فإن محرر الأكواد سينتقل إليه مباشرة حتى لو لم يكن هذا الإجراء هو الافتراضي.

عند إنشاء مشروع جديد ستحصل على الأسطر البرمجية التالية:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication5
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}

```

في الجزء الأول عندما ناقشنا محتويات أول مشروع، قلنا أنه مؤلف من قسمين، سأفترض أنك تعرف وتفهم كليهما، لكن سأشرحهما باختصار للتذكير:

في القسم العلوي هناك مجالات أسماء (أسميناها مكتبات) ماهي إلا ملفات تحوي فئات تؤدي وظائف معينة عن طريق مجموعة من الطرق والخصائص التابعة لها.

في القسم السفلي هناك مجال خاص بمشروعك، وفيه ستكتب أكوادك. قد يشير اهتمامك للإجراء InitializeComponent، لا عليك صادقه في برامحك وأعطيه بعض الاحترام، هو إجراء يقوم بتهيئة أدوات مشروعك عند بداية تنفيذ البرنامج. كيف عرفت أنه إجراء؟¹

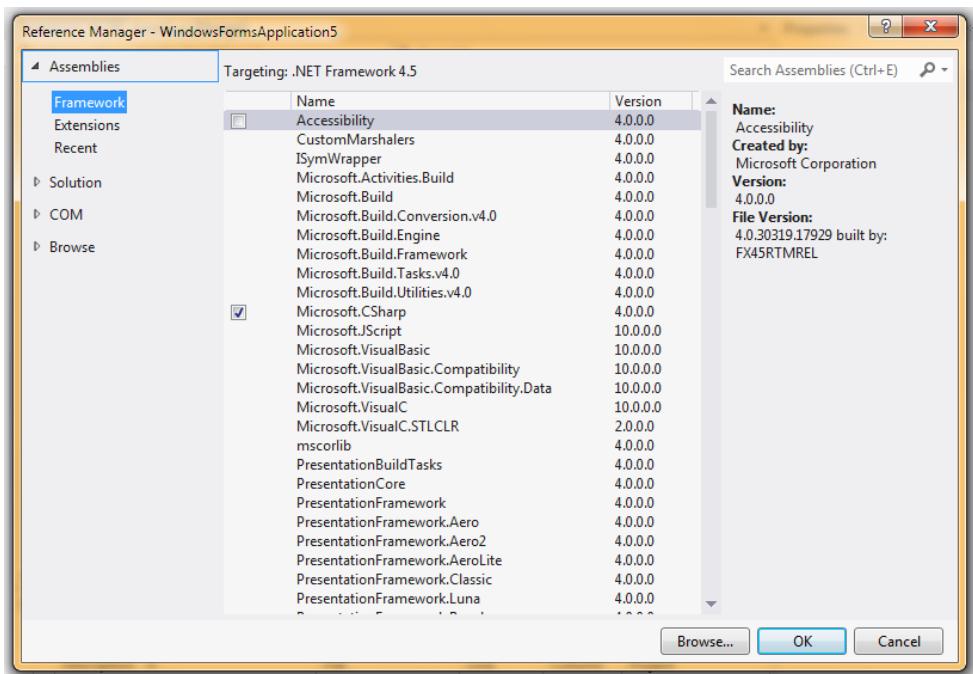
إذا كنت قد أكملت الجزء الأول من الكتاب بشكل جيد، أعتقد أنك تعرف معنى Form1 و () public Form1 : Form يعني public Form1 () : Form.. أما إذا لم تكمله أو لم تبدأ به فأكمله أو ابدأ به 😊😊😊. لم أكتبه لتقرأه جدتي 😞.

¹ يعرف الإجراء بأنه تابع أو طريقة لا تعيد قيمة، لذلك لا نسنده لمتغير أو لوسطاء طريقة أخرى.



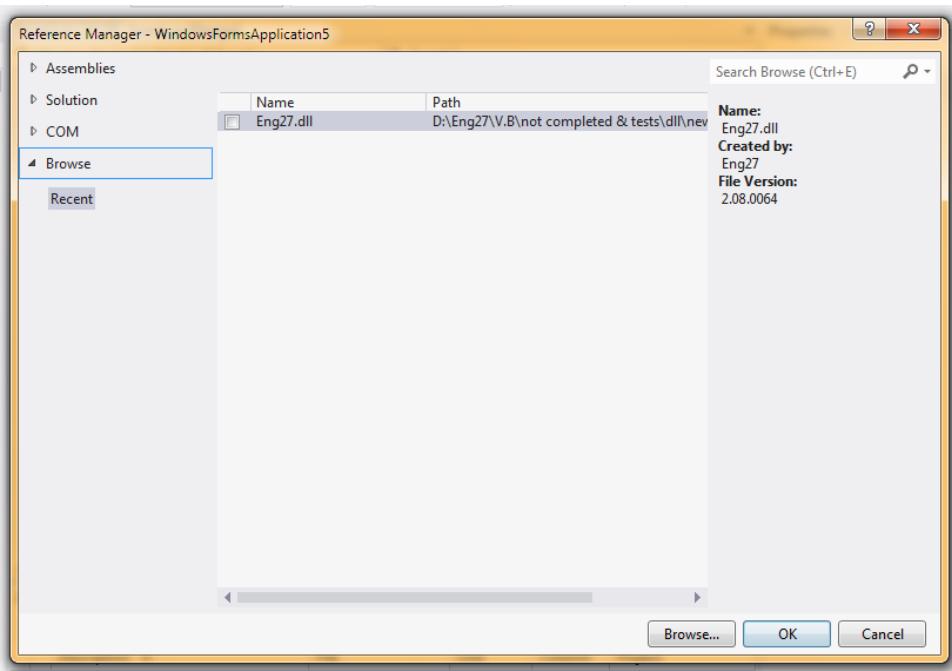
استيراد المراجع

قد لا تشبع رغباتك البرمجية الأدوات والفئات التي تحصل عليه مع كل مشروع جديد. قد تحتاج للتعامل مع الملفات مثلاً، أو البريد الإلكتروني، أو غيرها من الحاجات. بإمكانك إضافة مكتبة تمثل مجال أسماء تأتي هدية مع نسخة الفيجوال ستوديو، أو قد يكون لديك مكتبة خاصة بك أو بأحد أصدقائك، أو أنك حملتها بالنت، و تقوم بوظيفة ما، وترغب بإضافتها إلى المشروع لتخصر الأكواد أو لتحصل على نتيجة ما أنت لا تعلم كيفية الوصول إليها أو لا تهتم بالطريقة المسليكة للوصول إليها ولكنك تعلم نتيجتها. أيّاً كانت حاجتك وغايتها، بإمكانك إضافة مرجع ما عن طريق Add Reference، ثم البحث ضمن القائمة على المكتبة المطلوبة.¹



أو بإمكانك إضافة مكتبة خاصة بك – أنت برمجتها أو غيرك، المهم أنها على قرصك الصلب، قرص كمبيوترك الصلب أقصد :Browse 😐 – من

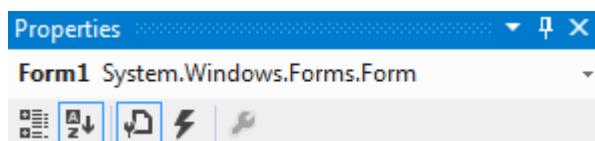
¹ تم شرح استيراد المراجع سابقاً في الجزء الأول من الكتاب باسم "إضافة المكتبة إلى مراجع المشروع"، ولأهميةه في هذا الجزء، ولأنه من المحتمل أن يكون بعض القراء قد انتقل تلقائياً إلى الجزء الثاني تم إعادة التنوية لذلك.



بالمناسبة: مكتبة Eng27.dll في الصورة السابقة تعمل على VB6 ولا تعمل على بيئة Net، وتؤدي وظائف لـ 8 مجالات مختلفة، كل مجال ممثل بفئة Class، وكل فئة فيها العديد من التوابع والإجراءات. مقصدي أنه عند استيراد مكتبة فإن لغة البرمجة لا تعرف أنها ستعمل عليه أمر لا.

الخصائص Properties

تتميز الأدوات عن بعضها بالخصائص التي تتمتع بها والأحداث التي قد تجري عليها. أما الأحداث فهي ظروف برمجية تمر على الأدوات وستناقشها لاحقاً إن شاء الله، وأما الخصائص فهي ماتميز كائن عن آخر، وبهذه الأداة - شريط الخصائص - بإمكانك إدارة أدوات مشروعك:



لديك خمسة أزرار في شريط الخصائص، الأول للتجميع بحسب الوظيفة، الثاني للتجميع بحسب الاسم، الثالث عرض الخصائص والرابع لعرض الأحداث. الجدول التالي يبين أهم الخصائص التي قد تحتاج تبيانها:

الوظيفة	الخاصية
الاسم البرمجي، ويختلف عن الاسم الظاهر للمستخدم المتمثل بالخاصية Text، حاول التفريق!	(Name)



ارتباط الأدوات بحواف حاوياتها ¹ ، وهي خاصية مهمة تحدد فيما إذا كانت الأداة ستبعد بمقدار ثابت عن حواف ما يحويها.	Anchor
تحدد فيما إذا كانت الأداة ستظهر عليها أشرطة تمرير بشكل تلقائي عندما يكون حجم الأدوات المحتواة أكبر من حجم هذه الأداة.	AutoScroll
الحجم التلقائي، وفيها يتغير حجم الأداة تبعاً للنص المحتوى داخلها.	AutoSize
لون الخلفية.	BackColor
Normalوضع الطبيعى، أحرف كبيرة وصغيرة. Upper تكبير جميع الأحرف. Lower تصغير جميع الأحرف.	CharacterCasting
أداة القائمة المنشقة المرتبطة مع هذه الأداة، لظهور معها عند النقر باليمنى عليها.	ContextStripMenu
شكل مؤشر الفأرة عند المرور بهذه الأداة.	Cursor
تحدد أي من الحدود متصلة بالأداة الحاوية، والتي ستتغير من حجم الأداة الحالية لتصبح بحجم حاويتها.	Dock
تحدد ما إذا كانت الأداة مفعولة أم لا، بحيث تظهر للمستخدم مع / مع عدم إمكانية استخدام الأداة.	Enabled
للأدوات التي خاصية FlatStyle لها هي Flat، تحدد عرض الحدود.	FlatAppearance
يكون فيها سطح الأداة مستويا. Popup يكون سطحها مستوياً ويتغير عند النقر. Standard القيمة القياسية. System الأدوات الشائعة لنظام.	FlatStyle
تحدد مواصفات الخط المستخدم في الأداة.	Font
تحدد لون الخط.	ForeColor
تحدد أيقونة الفورم (النموذج).	Icon
تحدد النص المحتوى على شكل مصفوفة نصوص.	Line
تحدد موقع الأداة بالنسبة للأداة الحاوية لها.	Location
تحدد فيما إذا كانت الأداة مفغولة أم لا، والأداة المفغولة هي أداة مفعولة (Enabled = True) ولكنها للقراءة فقط، لا يمكن تحريكها أو تغيير حجمها.	Locked
تحدد الفراغات بين هذه الأداة والأدوات المجاورة.	Margin
تحدد فيما إذا كان زر التكبير مفعلاً أم لا.	MaximizeButton
تحديد أكبر حجم ممكن للأداة الوصول إليه.	MaximumSize
عدد الأحرف الأعظمي الذي يمكن للأداة أن تحويه.	MaxLength

¹ قد تحتوى الأداة ضمن أداة أخرى مثل GroupBox أو أي أداة من أدوات الاحتواء، أو من قبل النافذة .Form



تحدد فيما إذا كان زر التصغير مفعلاً أم لا.	MinimizeButton
تحدد أصغر حجم ممكّن للأداة الوصول إليه.	MinimumSize
تحدد فيما إذا كانت الأداة متعددة الأسطر أم لا.	MultiLines
الشفافية، وتؤخذ كنسبة مئوية، القيمة 0 هي 0%， والقيمة 1 هي 100%.	Opacity
تحدد النموذج الأَب لهذا النموذج.	Parent
تشفيـر المحتوى النصـي بـرمـز.	PasswordChar
المحاذاة من اليمين لليسار. وهو مالم يكن يتخيـله العـرب حتـى، في لـغات البرـمـجة القـديـمة.	RightToLeft
تحدد محاذاة النموذج كـلـ، من اليمـين للـيسـار أو العـكـسـ، بـمعـنى بـرامـج عـربـية أو أجـنبـيةـ. وهو ما لم يكن يـحـلمـ به العـربـ حتـىـ، في لـغـاتـ البرـمـجةـ القـديـمةـ.	RightToLeftLayout
أشرطة التمرير، وهي أدوات تستخدم للانتقال من جـزـءـ لـآخـرـ منـ الأـدـاءـ عـنـدـماـ تكونـ متـعـدـدـةـ الأـسـطـرـ، منـ المـمـكـنـ أنـ تـكـوـنـ أـفـقـيـةـ أوـ شـاقـولـيـةـ أوـ كـلـاهـمـاـ.ـ فـيـ حـالـ لـمـ يـتـمـ تـحـديـدـهـاـ فـإـنـهـ لـنـ يـتـمـ إـظـهـارـ أـشـرـطـةـ التـمـرـيرـ حتـىـ لوـ كـانـ عـدـدـ أـسـطـرـ المـحـتـوىـ أـكـبـرـ مـنـ عـدـدـ أـسـطـرـ الأـدـاءـ المـمـكـنـ عـرـضـهـ.	ScrollBars
تحدد فيما إذا كانت الأيقونة ستظهر على شريط العنوان للفورم أم لا.	ShowIcon
تحدد فيما إذا كان هذا النموذج سيظهر في شريط المهام أم لا.	ShowInTaskbar
تحدد حجم الأداة.	Size
موقع بداية ظهور الفورم، ويفضل ضبطه في منتصف الشاشة.	StartPosition
تحدد ترتيب الأداة للانتقال إليها عند الضغط على زر Tap، أول أداة ترتيبها 0.	TapIndex
تحدد فيما إذا كانت الأداة سينتقل إليها عند الضغط على Tap أم لا.	TapStop
تحدد النص المحتوى في الأداة.	Text
تحدد محاذاة النص.	TextAlign
تحدد فيما إذا كانت الأداة ظاهرة أم مخفية.	Visible
تحدد فيما إذا كان النموذج سيظهر عند تحميله بـحـالـةـ تـكـبـيرـ أـمـ تصـغـيرـ أوـ الحـالـةـ المـصـمـمـ عـلـيـهـ.	WindowState
تحدد فيما إذا كان النص سينتقل لـسـطـرـ جـدـيدـ أـمـ لـعـدـدـ لـأـسـطـرـ عـنـدـمـ إـمـكـنـهـ اـحـتوـاؤـهـ، بـالـنـسـبـةـ لـلـأـدـوـاتـ مـتـعـدـدـةـ الـأـسـطـرـ.	WordWrap



الأحداث Events

الأحداث ماهي إلا إجراءات تُنفَّذ عند حدوث أمر ما، وهي إحدى الأمور التي تتميز بها الكائنات، فقد قلنا أن للكائن أفعال يقوم بها (أسميناها طرق)، وسمات يتميز بها (أسميناها خصائص)، وظروف تمر به (وهي الأحداث).

إن كل حدث مرتبط بطرف معين يمر بالكائنات، والكائنات التي سنتعامل مع أحداثها هي أدوات التحكم، مثل الزر وصندوق النص واللائحة وحتى النموذج وغيرها من الأدوات. تقع على هذه الأدوات ظروف مثل مرور مؤشر الماوس فوق الأداة، أو الضغط بأحد مفاتيح الكيبورد عندما يكون التركيز عليها، أو فقدان التركيز من الأداة وغيرها الكثير من الأحداث التي سأشرح أكثرها تكرارا في مشاريع هذا الكتاب.

الحدث	الوظيفة
Click	يحدث عندما يتم الضغط بزر الفأرة الأيسر (أو الافتراضي) على الأداة.
DoubleClick	يحدث عندما يتم الضغط مزدوجا على الأداة.
FormClosed	يحدث عندما يقوم المستخدم بإغلاق النموذج.
FormClosing	يحدث أثناء عملية إغلاق النموذج.
InputLanguageChanged	يحدث عند تغيير اللغة.*
KeyDown	يحدث عند بداية ضغط أحد مفاتيح الكيبورد.**
KeyPress	يحدث عند استمرار الضغط على أحد مفاتيح الكيبورد.*
KeyUp	يحدث عند نهاية ضغط أحد مفاتيح الكيبورد (رفع اليد عن المفتاح).**
MouseClick	يحدث عند النقر بالماوس، ويمكن الاستفاده منه في معرفة الزر المصغوط (يمين أو يسار أو الوسط).*
MouseDown	يحدث عندما يُنقر على زر الماوس، وأيضاً تستفاد منه في معرفة الزر المصغوط.**
MouseEnter	تحدد عندما تدخل الماوس المجال المرئي للأداة.
MouseMove	يحدث عند حركة مؤشر الماوس على إحدى الأدوات.***
TextChanged	يحدث عند تغيير المحتوى النصي للأداة (الخاصة بالنصوص).



* يمكن معرفة اللغة التي تم التغيير إليها ضمن الحدث `InputLanguageChanged` وذلك باستخدام قيمة المتغير `e` المعرف ضمن بارامترات الحدث، واستخدام خاصياته `Culture` و `DisplayName`، وذلك وفق الكود:

```
MessageBox.Show(e.Culture.DisplayName);
```

** يمكن الاستفادة من بارامترات أحداث لوحة المفاتيح في معرفة المفتاح المضغوط، أو مقارنة المفتاح المضغوط مع قيمة مرجعية.

```
//KeyEvents
MessageBox.Show(e.KeyCode.ToString()); // اسم المفتاح
MessageBox.Show(e.KeyData.ToString()); // اسم المفتاح
MessageBox.Show(e.KeyValue.ToString()); // شفرة المفتاح
//KeyPressEvents
// مع الأرقام يفضل استخدامها/
MessageBox.Show(e.KeyChar.ToString());
```

*** يمكن معرفة الزر المنقول في الماوس باستخدام الكود التالي:

```
//MouseEvents
if (e.Button == MouseButtons.Left)
    MessageBox.Show("الزر الأيسر");
if (e.Button == MouseButtons.Right)
    MessageBox.Show("الزر الأيمن");
if (e.Button == MouseButtons.Middle)
    MessageBox.Show("الزر المتوسط");
```

**** يمكن معرفة إحداثيات الماوس بالنسبة لإحدى الأدوات (الفورم Form) مثلًا عن طريق الكود:

```
//MouseMove
label1.Text = "X: " + e.X + ", Y:" + e.Y;
```





الفصل التاسع - مشاريع تطبيقية

بوصولك لهذا الفصل وانتهاءك من جميع محتويات الفصول السابقة فأنت رسمياً قد اجتازت مرحلة "المبتدئ" بنجاح وبإمكانك القيام بما لا يخطر على بالك وذلك في طريقك للوصول لمرحلة الإتقان والتي اختيارت تكون عنواناً لهذا الكتاب، تشجيعاً لقارئيه، ورغبةً شخصيةً من كاتبه.

هذا الكتاب وابتداءً من هذا الفصل وحتى نهاية الكتاب سيعطيك أفكاراً وأساليب للوصول لما عُنِونَ به، وأثناء ذلك تحتاج بعض الصبر، القليل من الوقت، ونت على مدار الساعة، والكثير الكثير من المشاريع المتوفرة على الانترنت، المchorة منها والمكتوبة.

حاول تقليل المشاريع - التي تجدها في هذا الكتاب أو على الانترنت - مرة ومرتين وثلاثة أو أكثر حتى تفهمها كاملاً، فإذا أتقنتها أعد صياغتها بأسلوبك وأعطيها لمساتك، وأضف عليها بعض الأفكار وألغِ غيرها، واحصل على بعض التشجيع والدعم من هذا ذاك، أو مني على أقل تقدير بتواصلك معي إذا كان لك رغبة بذلك!

المشاريع الموجودة في هذا الفصل متعلقة بشكل كبير بمحتويات الجزء الأول من هذا الكتاب، لذلك لن يتم شرح أي قاعدة أو صيغة إطلاقاً، وسيقتصر على شرح الأكواد المستخدمة، الجاهزة منها والمؤلفة.

أمر آخر أود التنويه له، مواضع الجزء السابق متربطة ومتتابعة ويُتوقع منك أن تكون قد تعلمتها - أو قرأتها - بشكل متسلسل، أما محتويات هذا الجزء - وخصوصاً هذا الفصل - فليست متعلقة ببعضها وقد تجد أكواداً غير مذكورة أو مشروحة أو منوه عنها في الكتاب، لذلك فعليك البحث والتحري.



مشروع TextToSpeech

الفكرة من المشروع هي تحويل النص إلى كلام، وفيه يقوم الكمبيوتر بنطق المحتوى النصي. وستتعامل مع المجال System.Speech.

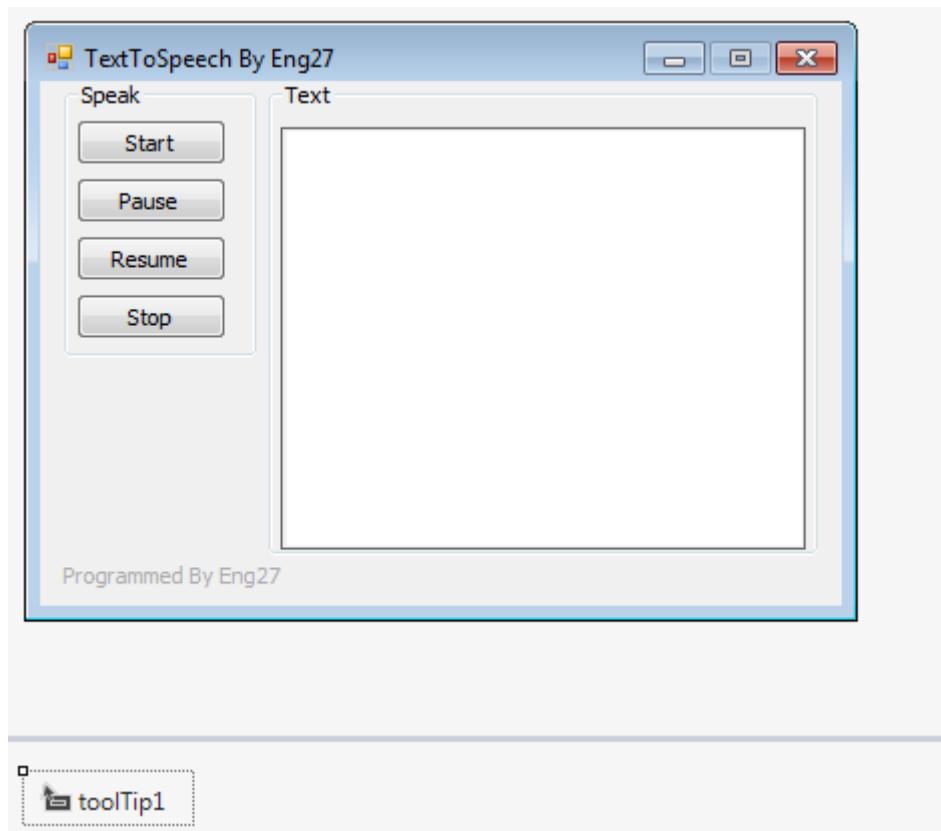
- الأدوات التي ستتعامل معها:

- زر Button
- لائحة Label
- صندوق تجميع GroupBox
- صندوق نص غني RichTextBox

- المكتبات التي ستتعامل معها:

- System.Speech
- System.Speech.Synthesis

أنشأ مشروعًا جديداً وأضف إلى مراجعه المجال السابق، ثم صمم النافذة وفق الشكل التالي:



أضف صندوقي تجميع GroupBox، أربعة أزرار Button، لائحتين Label، صندوق نص غني RichTextBox، وأداة عرض التلميحات ToolTip.



أضبط الخصائص التالية:

القيمة	الخاصية	الأداة
False	MaximizeBox	Form1
416; 300	MaximumSize	
	MinimumSize	
	Size	
CenterScreen	StartPosition	
TextToSpeech By Eng27	Text	
Speak	Text	GroupBox1
Text	Text	GroupBox2
Speak	Text	Button1
False	Enabled	
Pause	Text	Button2
False	Enabled	
Resume	Text	Button3
False	Enabled	
Stop	Text	Button4
False	Enabled	
FixedSingle	BorderStyle	Label1
6; 23	Location	
262; 211	Size	
Programmed By Eng27	Text	Label2
None	BorderStyle	RichTextBox1
9; 25	Location	
257; 207	Size	
Type the text you want to get spoken by the computer here!	ToolTip on ToolTip1	

خصائص النافذة تضمن لكبقاء حجمها ثابتا. واللائحة Label1 وظيفتها إعطاء حواف لصندوق النص عند جعله عديم الحواف، حيث نضع قيم الحجم والموقع متقاربة لكلا الأداتين. وبالنسبة لإلغاء تمكين الأزرار فيجب عدم تفعيلها إلا إذا احتوى صندوق النص على عبارات نصية.

ملاحظة

- لجعل النافذة بحجم واحد لا يتغير، أضبط الخصائص MaximumSize و Size على نفس المقدار، و MaximizeBox و MinimumSize على قيمة False.



انتهت المرحلة التصميمية، وبرنامتك الآن عبارة عن جثة هامدة، لبّ
الروح فيه استخدم الكود التالي:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

// مجالات الأسماء الجديدة اللازمة للمشروع
using System.Speech;
using System.Speech.Synthesis;

namespace TextToSpeech
{
    public partial class Form1 : Form
    {
        // استنساخ كائن جديد يمثل لسان الحاسوب
        SpeechSynthesizer Reader = new SpeechSynthesizer();
        // تهيئة مكونات النموذج
        public Form1()
        {
            InitializeComponent();
        }
        // التأكد من أن صندوق النص يحوي عبارات نصية
        // إذا لم يحوي على نصوص يجب عدم تمكين الأزرار من العمل
        private void richTextBox1_TextChanged(object sender, EventArgs e)
        {
            if (richTextBox1.Text != "")
            {
                button1.Enabled = true;
                button2.Enabled = true;
                button3.Enabled = true;
                button4.Enabled = true;
            }
            else
            {
                button1.Enabled = false;
                button2.Enabled = false;
                button3.Enabled = false;
                button4.Enabled = false;
            }
        }
    }
}
```



```
// تحدث
private void button1_Click(object sender, EventArgs e)
{
    Reader.Dispose();
    Reader = new SpeechSynthesizer();
    Reader.SpeakAsync richTextBox1.Text);
}

// إيقاف مؤقت
private void button2_Click(object sender, EventArgs e)
{
    if (Reader != null)
    {
        if (Reader.State == SynthesizerState.Speaking)
            Reader.Pause();
    }
}

// استئناف
private void button3_Click(object sender, EventArgs e)
{
    if (Reader != null)
    {
        if (Reader.State == SynthesizerState.Paused )
            Reader.Resume ();
    }
}

// إيقاف
private void button4_Click(object sender, EventArgs e)
{
    if (Reader != null)
    {
        Reader.Dispose();
    }
}
```

عند استنساخ الفئة `SpeechSynthesizer` إلى كائن ول يكن `Reader` مثلاً، فإن جميع طرق وخصائص الفئة سيمتلكها الكائن. وعلى اعتبار أن هذه الفئة هي لسان الكمبيوتر، فالكائن أصبح كذلك.

عند الضغط على زر البدء على الكائن أن يتوقف أولاً، ثم ينشئ نفسه من جديد ويتحدث بمحتويات النص.



عند الضغط على زر الإيقاف المؤقت، وإذا لم يكن الكائن لا شيء (أي أنه مازال على قيد الحياة)، وإذا كانت حالة الكائن هي الكلام، فعليه أن يتوقف.

بالمثل عند الضغط على زر الاستئناف، لكن الحالة معاكسة.

وأخيرًا، عند الضغط على زر الإيقاف فعلى الكائن أن يستدعي التابع `Dispose` والذي يوقفه من الكلام.

هل تذكر ماهية الطرق والخصائص؟؟ قلنا إنها أفعال يقوم بها الكائن وصفات يتميز بها. إن الطريقة `Dispose` هي فعل يقوم به الكائن بإيقاف الكلام، كما أن الخاصية `State` هي صفة تميزه فيما إذا كان متوقفا عن الكلام أم متحدثا.

ملاحظة

- لا يمكنك نسخ الأكواد بالكامل ولصقها في الفيجوال ستوديو عندك لأنها لن تعمل.. وذلك بسبب أن الأحداث الناتجة عن الأدوات مثل حدث ضغط الزر أو تغيير النص أو غيرها من الخصائص لا يمكن أن تنشأ إلا إذا قمت أنت بإنشاءها. طبعا لا تحدث عن إنشاء الأحداث برمجيا والذي تكلمنا عنه في الجزء الأول من الكتاب، وإنما أقصد أن تنشئ الحدث من خلال الأداة، فمثلا بالضغط مرتين على الزر `button1` يمكن أن تُنشأ حدث `button1_Click` ثم نضع بداخله الكود المناظر له في هذا الكتاب. وهكذا بالنسبة لجميع الأكواد...



مشروع EmailSender

يمكنك هذا المشروع من إرسال الرسائل الإلكترونية عبر تطبيقك، ولكنك قد تحتاج لأذونات من الحساب الذي ترغب بالإرسال منه.

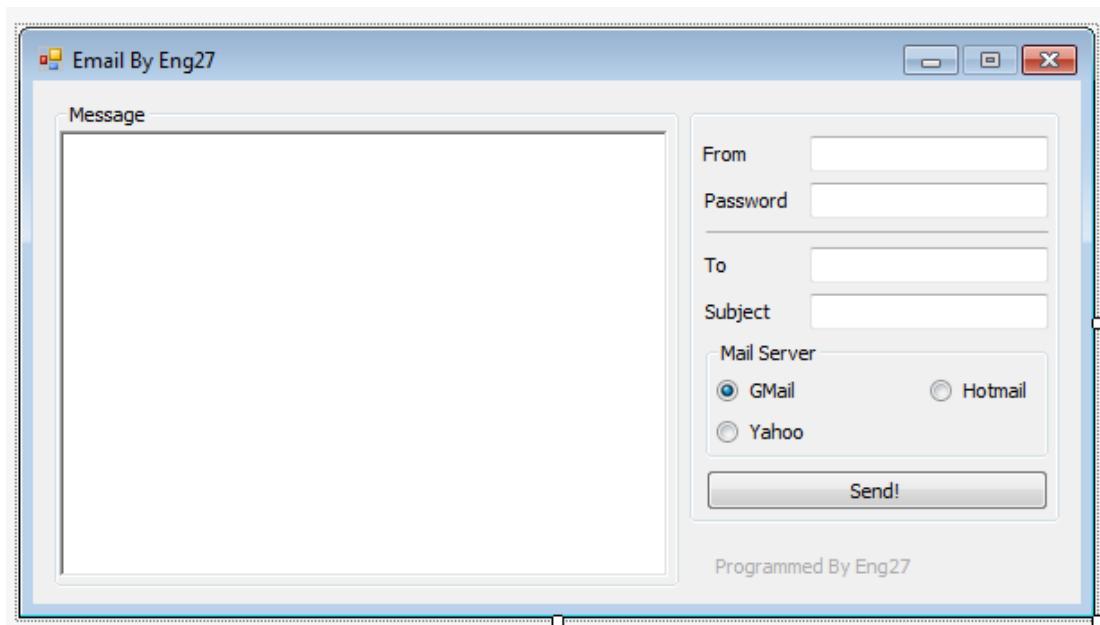
- الأدوات التي ستتعامل معها:

- زر Button
- صندوق نص TextBox
- قائمة Label
- صندوق تجميع GroupBox
- أداة اختيار RadioButton
- صندوق نص غني RichTextBox

- المكتبات التي ستتعامل معها:

- System.Web
- System.Net
- System.Net.Mail

صمم الواجهة التالية:



تحتاج ثلاثة صناديق تجميع `GroupBox`، أربعة صناديق نصوص `TextBox`، صندوق نص غني `RichTextBox`، خمسة لوائح `Label`، زر واحد `Button`، ثلاثة أدوات اختيار `RadioButton`، بالإضافة إلى خط `lineShape`.



اصبِطِ الخصائص التالية:

القيمة	الخاصية	الأداة
False	MaximizeBox	Form1
600; 330	MaximumSize	
	MinimumSize	
	Size	
CenterScreen	StartPosition	GroupBox1
Email By Eng27	Text	
	Text	GroupBox2
Message	Text	GroupBox3
Mail Server	Text	Button1
Send!	Text	
False	Enabled	
From	Text	
Password	Text	Label1
To	Text	Label2
Subject	Text	Label3
Programmed By Eng27	Text	Label4
	Text	Label5
None	BorderStyle	RishTextBox1
Gmail	Text	RadioButton1
True	Checked	
Yahoo	Text	
Hotmail	Text	RadioButton2
	Text	RadioButton3

استخدم الكود التالي:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

مجالات الأسماء المطلوبة لإتمام المشروع//

```
using System.Web;
using System.Net;
using System.Net.Mail;
```



```

namespace EmailSender
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        // عند الضغط على زر الإرسال
        private void button1_Click(object sender, EventArgs e)
        {
            // إنشاء كائن لإرسال رسالة من خلاله
            MailMessage Mail = new MailMessage( from.Text,
                to.Text,
                subject.Text,
                body.Text);
            string SMTP;
            // اختيار نوع السرفر بحسب شركة البريد
            // مع إعطاء قيمة ابتدائية حتى لا يحصل أخطاء
            SMTP = "smtp.gmail.com";
            if (radioButton1.Checked)
                SMTP = "smtp.gmail.com";
            else if (radioButton2.Checked)
                SMTP = "smtp.mail.yahoo.com";
            else if (radioButton3.Checked)
                SMTP = "smpt.live.com";
            // إنشاء كائن يمثل الخادم ليتواصل مع البورت الخاص بالشركة
            SmtpClient Client = new SmtpClient(SMTP,587);
            Client.UseDefaultCredentials = false;
            Client.Credentials = new NetworkCredential(from.Text,
                password.Text);
            Client.EnableSsl = true;
            try
            {
                // محاولة إرسال البريد
                Client.Send(Mail);
                MessageBox.Show("Mail : " + subject.Text + " sent successfully!",
                    "Message Sent",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information,
                    MessageBoxDefaultButton.Button1,
                    MessageBoxOptions.DefaultDesktopOnly);
                body.Text = "";
            }
            catch (Exception ex)
            {
                // في حال وجود خطأ، أظهر رسالة بذلك!
            }
        }
    }
}

```



```
        MessageBox.Show(ex.Message,
                    "Message Cann't Sent",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error,
                    MessageBoxDefaultButton.Button1,
                    MessageBoxOptions.DefaultDesktopOnly);
    }
}

// يجب عدم تفعيل زر الإرسال مالم يتم التحقق من أن جميع المعطيات مدخلة
```

```
private void body_TextChanged(object sender, EventArgs e)
{
    CheckDATA();
}

private void password_TextChanged(object sender, EventArgs e)
{
    CheckDATA();
}

private void to_TextChanged(object sender, EventArgs e)
{
    CheckDATA();
}

private void subject_TextChanged(object sender, EventArgs e)
{
    CheckDATA();
}

private void CheckDATA()
{
    if (body.Text == "" &&
        from.Text == "" &&
        password.Text == "" &&
        to.Text == "" &&
        subject.Text == "") button1.Enabled = false;
    else button1.Enabled = true;
}
```

ومعنى الكود السابق، أن على البرنامج أن يستخدم ثلاث مكتبات، وعند الضغط على زر إرسال فإنه سيتم إنشاء كائن يقوم باستقبال البريد



المرسل منه والمرسل إليه وموضع الرسالة وجسمها. ثم يحدد نوع السرفر وذلك بناءً على نوع مخدم البريد الذي سترسل منه ويتمثل هذا بعبارة نصية SMTP، وبناءً عليه يتم إنشاء كائن ليتواصل مع شركة البريد بعد تحديد رقم البورت. وللقيام بالإرسال يجب تحديد بريد وكلمة سر المرسل، ثم محاولة الإرسال وإعطاء رسالة للمستخدم بالفشل أو النجاح.

متخيل أن كلّ هذا يحدث في برنامج الرسائل الإلكترونية؟؟ من الآن وصاعداً قبل انتقادك لبرنامج البريد الإلكتروني تذكّر ما يقوم به أولاً ..

يجب عدم تفعيل زر الإرسال مالم يتم التأكد من أن جميع البيانات قد تم إدخالها، ولذلك أنشأنا إجراءً أسميناه CheckDATA، وهو إجراء يقوم بالتأكد من أن جميع الحقول ليست فارغة. نستدعي هذا الإجراء في حدث تغيير النص لكل من صناديق النصوص المستخدمة العادية منها والغنية.

يمكن معرفة تفاصيل SMTP لكل شركة وذلك بالبحث بالإنترنت عنها، وعند البحث ستحصل على مايلي:

Gmail's Default SMTP Settings

- Gmail SMTP server address: **smtp.gmail.com**
- Gmail SMTP username: **Your Gmail address** (e.g. *example@gmail.com*)
- Gmail SMTP password: **Your Gmail password**
- Gmail SMTP port (TLS): **587**
- Gmail SMTP port (SSL): **465**
- Gmail SMTP TLS/SSL required: **yes**

Yahoo SMTP Server Settings

The SMTP server settings are the same for both POP and IMAP accounts. In most cases, you enter them in the Settings section of the email provider when you set up the Yahoo account. Enter the following information in the email program you plan to use to send Yahoo Mail:

- Yahoo Mail SMTP server address: **smtp.mail.yahoo.com**
- Yahoo Mail SMTP username: Your full **Yahoo Mail email address** (including "@yahoo.com")
- Yahoo Mail SMTP password: Your **Yahoo Mail password**
- Yahoo Mail SMTP port: **465** (587 is an alternative)
- Yahoo Mail SMTP TLS/SSL required: **yes**



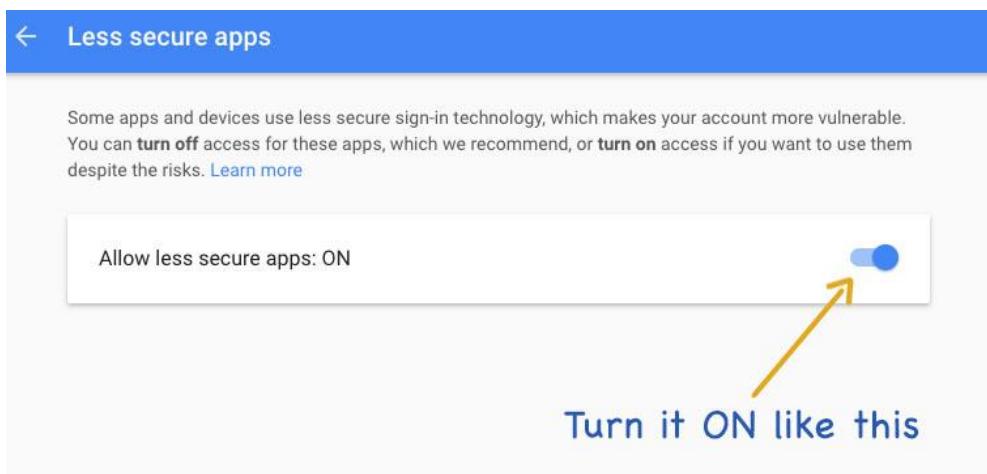
Windows Live Hotmail SMTP Server Settings

These are the outgoing SMTP server settings for sending mail using Windows Live Hotmail from any email program, mobile device, or another email service:

- Hotmail SMTP Server: **smtp.live.com**
- Hotmail SMTP Username: Your complete **Windows Live Hotmail email address** (e.g. *me@hotmail.com* or *me@live.com*)
- Hotmail SMTP Password: Your **Windows Live Hotmail password**
- Hotmail SMTP Port: **587**
- Hotmail SMTP TLS/SSL Required: **yes**

Tip: You can also use the [Outlook.com SMTP server settings](#) for your Hotmail account since, as you can read below, the two services are now the same.

في حال وجود أخطاء جرب أن تسمح بالتطبيقات قليلة الأمان من استخدام بريدك الإلكتروني وذلك من إعدادات حسابك. كما في الصورة:



- لإرسال ملفات مع الرسالة بإمكانك إدراج الملفات إلى الرسالة بالكود التالي:

```
Mail.Attachments.Add(new Attachment("C:\\\\MyFile1.txt"));
Mail.Attachments.Add(new Attachment("d:\\\\MyFile2.mp3"));
// بإمكانك إضافة ملفات أكثر
// بإمكانك إنشاء صناديق إدخال لوضع مسار الملفات فيها
```

ضع الكود بعد استنساخ الكائن Mail.



مشروع Windows Explorer

يقدم لك هذا المشروع فكرة بسيطة عن مبدأ عمل متصفحات الملفات، والتي من خلالها بإمكانك الوصول للملفات والمجلدات في أقراصك، التعامل معها، البحث عنها وحتى فتحها..

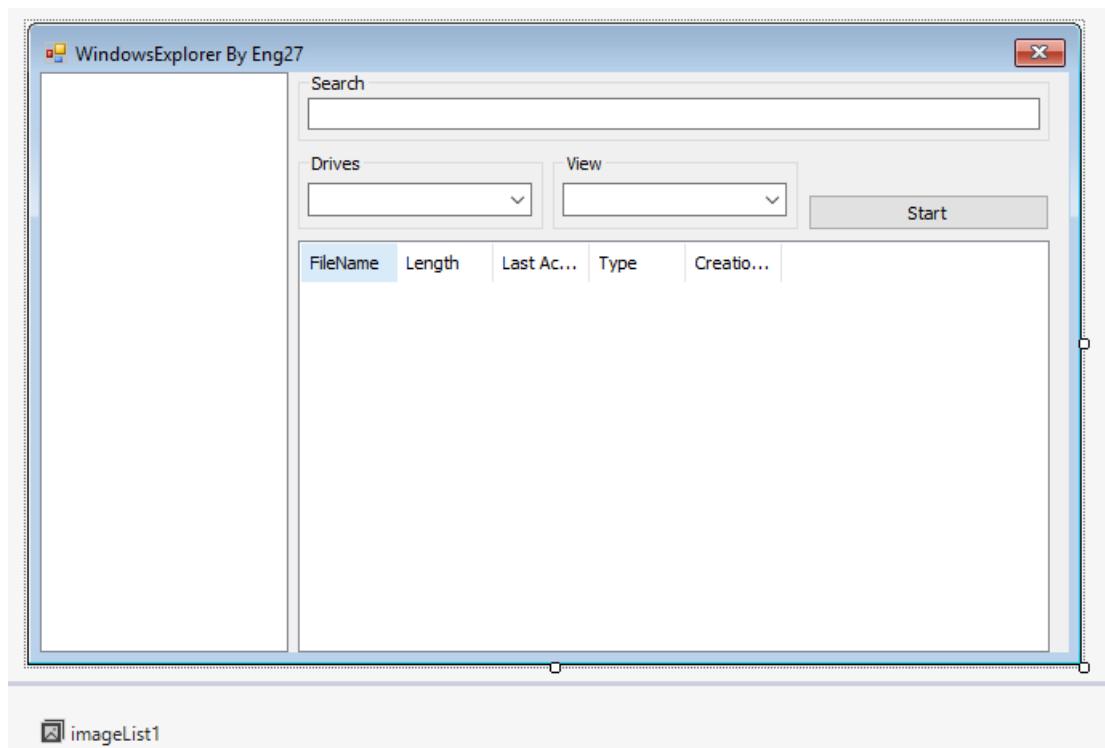
- الأدوات التي ستتعامل معها:

- زر Button
- صندوق نص TextBox
- صندوق تجميع GroupBox
- صندوق الاختيار ComboBox
- شجرة العرض TreeView
- قائمة العرض ListView
- قائمة الصور ImageList

- المكتبات التي ستتعامل معها:

- System.IO
- System.Diagnostics

صمم الواجهة التالية:



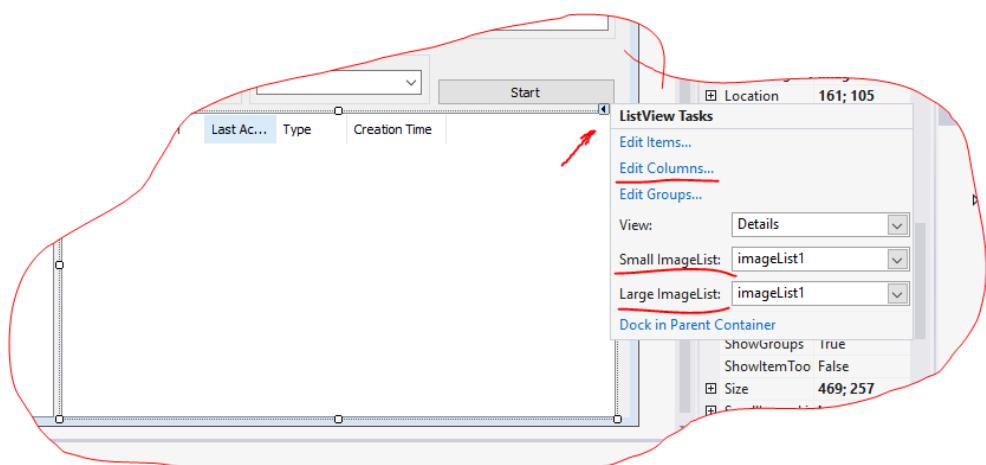


أضف ثلاثة صناديق تجميع **GroupBox**، زر واحد **Button**، صندوق نص واحد **TextBox**، صندوق اختيار **ComboBox**، وشجرة عرض **TreeView**، قائمة عرض **ListView**، قائمة صور **ImageList**

ثم اضبط الخصائص التالية:

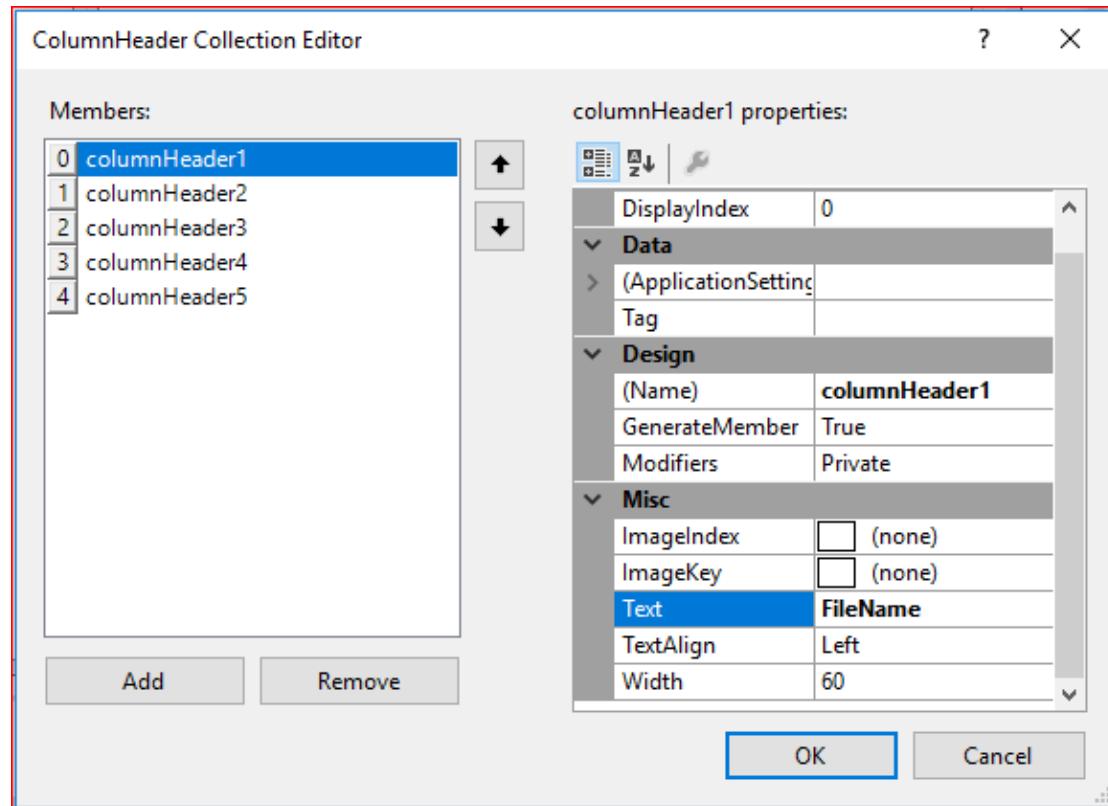
القيمة	الخاصية	الأداة
False	MaximizeBox	Form1
False	MinimizeBox	
WindowsExplorer By Eng27	Text	
Drives	Text	GroupBox1
View	Text	GroupBox2
Search	Text	GroupBox3
Start	Text	Button1
False	Enabled	
Left	Dock	TreeView1
ImageList1	ImageList	
True	Scrollable	
FileName	Text	ListView1
Length	Text	
Last Access	Text	
Type	Text	
Creation Time	Text	
ImageList1	LargeImageList	ListView1
ImageList1	SmallImageList	
True	Scrollable	

يمكن ضبط خصائص أداة قائمة العرض بصورة سريعة ومختصرة كما يلي:

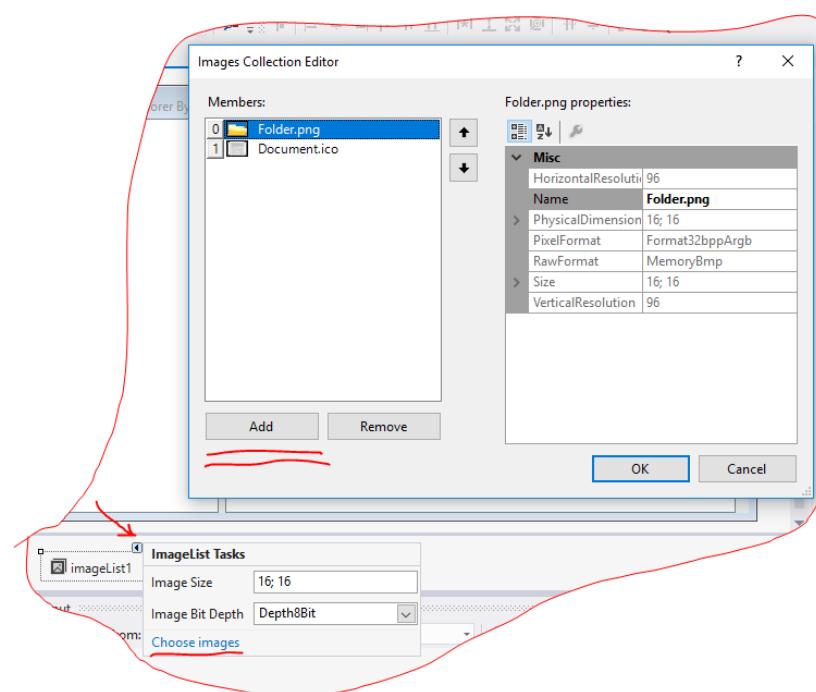




كما يمكن ذلك من شريط الخصائص. لإضافة عناوين للأعمدة ادخل إلى خصائص الخاصية Culomns ثم أضف خمسة قوائم، واجعل خاصية النص لها كما هو في جدول الخصائص في الصفحة السابقة.



أما بالنسبة للأيقونات والصور الخاصة ببرنامتك والتي ستعرض في أدوات العرض، فاضبطها كما يلي:





أما بالنسبة للكود المستخدم فهو كما يلي:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

المكتبات المستخدمة//
using System.IO; //للتعامل مع المجلدات والملفات
using System.Diagnostics; //للتعامل مع البرامج والعمليات

namespace WindowsExplorer
{
    public partial class Form1 : Form
    {
        string SelectedDrive, SelectedPath;

        public Form1()
        {
            InitializeComponent();
            getDrives(); comboBox1.Text = "C:\\";
            comboBoxView(); comboBox2.Text = "Details";
        }

        //تهيئة صندوق الاختيار الخاص بالعرض
        void comboBoxView()
        {
            comboBox2.Items.Add ("LargeIcon");
            comboBox2.Items.Add ("Details");
            comboBox2.Items.Add ("SmallIcon");
            comboBox2.Items.Add ("List");
            comboBox2.Items.Add ("Tile");
        }

        //تهيئة صندوق الاختيار الخاص بعرض الأقراص المتوفرة
        void getDrives()
        {
            foreach (string drive in Directory.GetLogicalDrives ())
            {
                comboBox1.Items.Add (drive);
            }
        }
}
```



```

الحصول على مجلدات القرص المحدد//  

void getFolders()  

{  

    SelectedDrive = comboBox1.Text;  

    DirectoryInfo Dir = new DirectoryInfo(SelectedDrive);  

    treeView1.Nodes.Clear();  

    foreach (DirectoryInfo Folder in Dir.GetDirectories())  

    {  

        treeView1.Nodes.Add("", Folder.Name, 0, 0);  

    }  

}  

الحصول على ملفات القرص المحدد//  

void getFiles(string strPath)  

{  

    ListViewItem lvi;  

    DirectoryInfo Dir = new DirectoryInfo(SelectedDrive + strPath);  

    listView1.Items.Clear();  

    try  

    {  

        foreach (FileInfo files in Dir.GetFiles())  

        {  

            lvi = listView1.Items.Add(files.Name, 1);  

            double FileLength = files.Length;  

            lvi.SubItems.Add(Math.Round ( FileLength / 1000,3) + " KB");  

            lvi.SubItems.Add(files.LastAccessTime.ToString());  

            lvi.SubItems.Add(files.Extension);  

            lvi.SubItems.Add(files.CreationTime.ToString());  

        }  

    }  

    catch (Exception ex)  

    {  

        MessageBox.Show(ex.Message);  

    }  

}  

عند تغيير العنصر المحدد، يجب الحصول على مجلداته//  

private void comboBox1_SelectedIndexChanged(object sender,  

EventArgs e)  

{  

    getFolders();  

}  

عند النقر على أحد عناصر شجرة العرض، يجب الحصول على ملفاته//  

private void treeView1_AfterSelect(object sender, TreeViewEventArgs e)  

{  

    SelectedPath = e.Node.FullPath ;
}

```



```

getFiles(SelectedPath);
TreeNode node;
DirectoryInfo Dir = new DirectoryInfo(SelectedDrive + SelectedPath);
try
{
    foreach (DirectoryInfo folder in Dir.GetDirectories())
    {
        node = new TreeNode(folder.Name, 0, 0);
        e.Node.Nodes.Add(node);
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
SelectedPath = SelectedDrive + SelectedPath;
}

// عند تغيير العنصر المحدد، يجب تغيير طريقة العرض
private void comboBox2_SelectedIndexChanged (object sender,
EventArgs e)
{
    if (comboBox2.Text == "LargeIcon")
        listView1.View = View.LargeIcon;
    else if (comboBox2.Text == "Details")
        listView1.View = View.Details;
    else if (comboBox2.Text == "SmallIcon")
        listView1.View = View.SmallIcon;
    else if (comboBox2.Text == "List")
        listView1.View = View.List;
    else
        listView1.View = View.Tile;
}

// تشغيل عملية ما بعد تحديدها من قائمة العرض
// هذه العملية ستعمل بالعملية الافتراضية لها
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        Process.Start(SelectedPath + "\\\"+ listView1.FocusedItem.Text);
    }
    catch (Exception ex)
    { MessageBox.Show(ex.Message); }
}

```



```

عندما تكسب قائمة العرض التركيز يتم تفعيل الزر //  

private void listView1_Enter(object sender, EventArgs e)  

{ button1.Enabled = true; }  

وعندما تكسب شجرة العرض التركيز يتم إلغاء تفعيل الزر //  

private void treeView1_Enter(object sender, EventArgs e)  

{  

    button1.Enabled = false;  

}  
  

عند تغيير المحتوى النصي لصندوق النص يتم البحث بشكل تلقائي //  

هذا يضمن لك ديناميكية عمل الأداة، وهو أفضل من وضع زر للبحث //  

private void textBox1_TextChanged(object sender, EventArgs e)  

{  

    if (File.Exists(SelectedPath + "\\\" + textBox1.Text))  

        MessageBox.Show("Exists!");  

}  

}  

}

```

والكود السابق معناه:

أنا نرغب بالتعامل مع مكتبة الملفات والمجلدات ومكتبة العمليات (البرامج) في ويندوز، لذلك ألقنهاهما في مشروعنا.

عّرفنا متغيرين سنضع فيهما القرص الذي اختاره المستخدم (من صندوق الاختيار) والمجلد الذي انتقل إليه المستخدم (في شجرة العرض).

عند تحميل النموذج (بداية تشغيله) وبعد تهيئة مكونات المشروع، يجب الحصول على جميع أقراص نظام التشغيل، وذلك من خلال إجراء getDrives، ثم نجعل النص المحدد في صندوق الاختيار هو القرص "C:\\" على اعتبار أنه فعلياً جميع أنظمة التشغيل أول قرص فيها هو القرص C، ثم نقوم بملئ عناصر صندوق الاختيار الثاني بأشكال العرض الممكنة، وذلك من خلال إجراء comboView، ثم جعل النص المحدد في صندوق الاختيار هذا هو "Details" على اعتبار أنها صمنا الأداة على أن تعرض الملفات بالشكل تفاصيل.

ملاحظة

- لا تنسَ أنه لكتابة الرمز \ في عبارة نصية يجب كتابته مكرراً لأنه أحد رموز تنسيق العبارات النصية.
- يمكن استخدام @ في بداية العبارة النصية الممثلة للمسارات بدلاً من تكرار \ التي تفصل بين المسارات.



يقوم الإجراء comboView بإضافة عناصر لصندوق الاختيار، وهي عناصر معروفة (أشكال عرض الملفات)، أما getDrives فإنه يقوم بإضافة أسماء الأقراص المتوفرة في نظام التشغيل وذلك من خلال حلقة foreach متغيرها متغير نصي يمثل ناتج تنفيذ الإجراء GetLogicalDrives التابع للفئة Directory.

ملاحظة

- قد تحصل على رسالة خطأ عند فتح أحد الأقراص مفادها أن القرص غير جاهز، وسببها أن هذا القرص قد يكون قرص وهمي غير مستخدم.
- تستخدم الأقراص الوهمية عادة عند تثبيت أو تشغيل بعض الألعاب أو البرامج الضخمة والتي تحتاج لملفات خارجية يستصعب تواجدها في الأقراص العادية.

للحصول على مجلدات القرص المحدد (ووضعها في شجرة العرض)، فيتم استدعاء الإجراء getFolder، وفيه يتم إسناد اسم القرص المحدد إلى متغير نصي SelectedDrive والذي عرفناه قبل قليل، ثم ننشئ كائنا Dir يمثل مجلدات القرص الحالي الممثل بالمتغير النصي الأخير، ثم تممسح مكونات شجرة العرض لإضافة عناصر جديدة، ثم تتم إضافة جميع المجلدات الموجودة داخل المجلد Dir إلى شجرة العرض وذلك من خلال حلقة foreach متغيرها كائن folder يمثل مجلدات الكائن Dir.

أما للحصول على ملفات المجلد المحدد - ووضعها في قائمة العرض - يتم استدعاء الإجراء getFiles والذي له وسيط وحيد وهو مسار المجلد المحدد، وفيه يتم إنشاء كائن يمثل عنصر قائمة عرض *lvi* للقيام بعمليات على عناصر قائمة العرض بعد قليل دون تكرار وإكثار الأكواد، ثم أنشأنا كائنا Dir وهو استنساخ للفئة مجلد، وهذا الكائن هو عبارة عن مسار المجلد المحدد لعرض ملفاته، ثم تممسح مكونات قائمة العرض لإضافة عناصر جديدة، ثم تتم محاولة إضافة جميع الملفات الموجودة داخل المجلد Dir إلى قائمة العرض وذلك من خلال حلقة foreach متغيرها كائن من files نوع ملف يمثل ملفات الكائن Dir، ومن أجل كل عنصر يجب إضافة اسمه أولاً، ثم تفاصيل أخرى عبر SubItems. وعند وجود خطأ يتم عرض رسالة نصية تشرحه ثم يكمل البرنامج مسيرته للحصول على ملفات أخرى.

لتشغيل عملية ما من عمليات ويندوز - وهي أي برنامج أو تطبيق - يتم استخدام الإجراء Start ضمن الفئة Process التابعة لمجال الأسماء Diagnostics، وهو إجراء يستقبل قيمة نصية تمثل مسار العملية.



والجدير بالذكر أن الطريقة Start تضمن لك تشغيل العملية بالمشغل الافتراضي، مثل تشغيل ملف mp3 بمشغل أغاني، أو pdf بقارئ ملفات إلكترونية، أو txt بمستعرض ملفات نصية..

عندما تصبح أداة قائمة العرض هي الأداة الفعالة (عندما تكسب التركيز بلهجة الفيجوال بيسبك) فإنه يتم تفعيل زر تشغيل العملية، وهذا الإجراء وضعيته لمحاولة إقلال عدد الأخطاء، تخيل لو أن المستخدم لم يحدد الملف المراد تشغيله وضغط زر تشغيل العملية ماذا تعتقد أن يحدث؟ بالتأكيد رسالة خطأ تودي بحياة برنامجك مالم تتحكم بها. ومع هذا فللخطأ نصيب محتمل ولكن باحتمالية أقل.

عندما تصبح أداة شجرة العرض هي الأداة الفعالة فإنه من البديهي أن أداة قائمة العرض ليست الأداة الفعالة في المشروع حالياً (وجميع الأدوات الأخرى بطبيعة الحال)، لذلك يجب عدم تفعيل زر تشغيل العملية لأنه ما من ملف محدد.. هل فهمت الفكرة من آخر إجرائين؟؟؟

ملاحظة

- إذا كانت المشاريع صعبة بالنسبة لك فلا عليك هي ليست سهلة بالتأكيد ولا يمكن لمبتدئ أن يتلقنها، بإمكانك تطبيق الكود عدة مرات أو سؤال أحد المختصين أو ذوي الخبرة، أو العودة لأفكار الجزء الأول لمحاولة تعويض المعلومات المبهمة.
- إذا كانت المشاريع سهلة بالنسبة لك فيإمكانك الإنقال لمشاريع ذات تطبيقات أوسع وأكثر واقعية مثل تطبيقات إدارة المبيعات أو ما شابهها، كما أنه بإمكانك ومع بعض الأفكار القيام بالكثير من التطبيقات المفيدة.



NotePadDataBase مشروع

يمكنك هذا المشروع من اخزن قيم معينة في ملف ذو امتداد معين، وهو مشابه لفكرة قواعد البيانات لأنه يخزن مجموعة من البيانات في ملف واحد. هذا المشروع يسمح للمستخدم بإدخال مجموعة من القيم النصية دون تكرارها، بالإضافة لحذف قيمة معينة أو حذف جميع القيم.

لا يضمن المشروع لك ارتباط العناصر وفق نسق معين، مثل رقم id مثلاً كما في قواعد البيانات العادية، لكنه يسمح لك بترتيب العناصر تصاعدياً أو تناظرياً، دون إمكانية العودة للترتيب القديم.

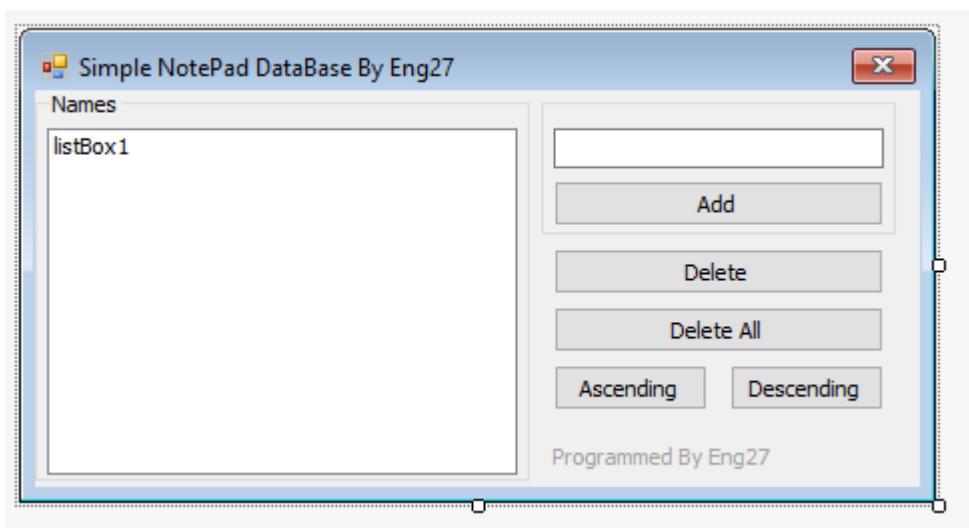
- الأدوات التي ستتعامل معها:

- زر Button
- صندوق نص TextBox
- قائمة Label
- صندوق تجميع GroupBox
- أداة القائمة ListBox

- المكتبات التي ستتعامل معها:

System.IO •

صمم الواجهة التالية:



أضف صندوقي تجميع GroupBox، خمسة أزرار Button، قائمة واحدة .ListBox، صندوق نص واحد TextBox، وأداة قائمة واحدة Label



اصبِطِ الخصائص التالية:

القيمة	الخاصية	الأداة
CenterScreen	StartPosition	Form1
Simple NotePad DataBase By Eng27	Text	
Names	Text	GroupBox1
	Text	
Add	Text	Button1
False	Enabled	
Delete	Text	Button2
False	Enabled	
Delete All	Text	Button3
False	Enabled	
Ascending	Text	Button4
False	Enabled	
Descending	Text	Button5
False	Enabled	
Programmed By Eng27	Text	Label1

استخدم الكود التالي:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

//المكتبة المطلوبة في المشروع
using System.IO;

namespace NotepadDatabase
{
    public partial class Form1 : Form
    {
        //متغيرات ستلزمنا على امتداد المشروع
        string Path = Application.StartupPath + @"\Data.dat";
        string SelectedData;
```



```

int SelectedIndex;
List<string> Names = new List<string>();
public Form1()
{
    InitializeComponent();
    إذا كان الملف موجودا//
    if (File.Exists (Path))
        حمل البيانات//();
    وإن لم يكن موجودا//
    {
        var file = File.Create(Path);//
        ثم أغلقه حتى نستطيع فتحه للقراءة//();
        //File.Create(Path).Close() is the same //
        الآن حمل البيانات بعد أن أنشأنا الملف//();
    }
}

تحميل البيانات من الملف//
private void LoadData ()
{
    يجب أولا مسح أدوات الإدخال والإخراج والتخزين//
    textBox1.Text = string.Empty;
    listBox1.Items.Clear();
    Names.Clear();
    foreach (string Name in File.ReadLines(Path, Encoding.UTF8))
    {
        listBox1.Items.Add (Name);
        Names.Add(Name);
    }
}

التأكد من أن الاسم المراد إضافته غير موجود//
private bool IsData(string NameToCheck)
{
    bool check =false ;
    foreach (string Name in File.ReadLines(Path, Encoding.UTF8))
    {
        if (NameToCheck.ToLower() == Name.ToLower())
        {
            check = true;
            break;
        }
        else
            check = false;
    }
    return check;
}

```



```

// إضافة اسم
private void button1_Click(object sender, EventArgs e)
{
    استدعاء إجراء التحقق من وجود الاسم//
    bool CheckName = IsData(textBox1.Text);
    اذا كانت النتيجة سلبية هذا يعني أن الاسم غير موجود//(
    {
        File.AppendAllText(Path, textBox1.Text + Environment.NewLine );
    }
    وإن لم تكن سلبية معناها أن الاسم موجود ويجب إدخال اسم جديد//
    MessageBox.Show("Try another name!", "Name exists!");
    وعلى كل الأحوال حمل البيانات من جديد//
}
}

إذا كان مربع النص الخاص بإضافة اسم جديد فارغ//(
يجب عدم تفعيل زر إضافة اسم جديد
private void textBox1_TextChanged(object sender, EventArgs e)
{
    if (textBox1.Text != string.Empty) button1.Enabled = true;
    else
        button1.Enabled = false;
}

عند الضغط على زر الحذف//
private void button2_Click(object sender, EventArgs e)
{
    اذا كان العنصر المحدد موجود ضمن أداة القائمة من الممكن حذفه//
    if (listBox1.Items.Contains(SelectedData))
    {
        var x = MessageBox.Show("Do you want to delete " +
            SelectedData +
            "?",
            "Delete data?",
            MessageBoxButtons.YesNo);
        if (x == System.Windows.Forms.DialogResult.Yes)
        {
            File.Delete(Path);
            Names.Remove(SelectedData);
            foreach (string Name in Names)
                File.AppendAllText (Path, Name + Environment.NewLine);
            LoadData();
        }
    }
    اذا كان عدد عناصر أداة القائمة معدوما يجب إلغاء تفعيل زر الحذف//
    if (listBox1.Items.Count == 0)
    { button2.Enabled = false; button3.Enabled = false; }
}

```



```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    SelectedData = listBox1.Text; // نص القائمة
    if (SelectedData != string.Empty)
    {
        SelectedData = listBox1.SelectedItem.ToString();
        SelectedIndex = listBox1.SelectedIndex;
        listBox1_Leave(sender, e);
    }
}

private void listBox1_Leave(object sender, EventArgs e)
{
    if (SelectedData != null)
        if (listBox1.Items.Contains(SelectedData))
        {
            button2.Enabled = true;
            button3.Enabled = true;
        }
        else
        {
            button2.Enabled = false;
            button3.Enabled = false;
        }
}

// حذف جميع العناصر
private void button3_Click(object sender, EventArgs e)
{
    var x = MessageBox.Show("Do you want to delete all data??",
                           "Delete data?",
                           MessageBoxButtons.YesNo);
    if (x == System.Windows.Forms.DialogResult.Yes)
    {
        File.Delete(Path);
        Application.Restart(); // إعادة تشغيل التطبيق
    }
}

// ترتيب تصاعدي
private void button4_Click(object sender, EventArgs e)
{
    File.Delete(Path);
    Names.Sort();
    foreach (string Name in Names)
        File.AppendAllText(Path, Name + Environment.NewLine);
    LoadData();
}
```



```
{
    }
    ترتيب تناظلي //
    private void button5_Click(object sender, EventArgs e)
    {
        File.Delete(Path);
        Names.Reverse();
        foreach (string Name in Names)
            File.AppendAllText(Path, Name + Environment.NewLine);
        LoadData();
    }
}
}
```

وفي الكود السابق، عرّفنا عدة متغيرات ستلزمها خلال المشروع، وهي كما يلي:

مسار الملف Path: وهو متغير نصي، سيأخذ قيمة مسار التطبيق، ويتبعه باسم للملف ول يكن Data ذو اللاحقة .dat.

العنصر المحدد SelectedData: وهو متغير نصي سيأخذ قيمة العنصر المحدد من أداة القائمة.

ترتيب العنصر المحدد SelectedIndex: وهو متغير رقمي سيأخذ قيمة العنصر المحدد من أداة القائمة للاستفادة منه عند الحذف.

قائمة بالاسماء Names: وهي قائمة (تشبه المصفوفة لكنها متغيرة)، ستحتفظ بالعناصر لإدراجهما في ملف جديد، أو عند حذف بعض العناصر.

ملاحظة

- يمكن الحصول على متغيرات كثيرة تربط تطبيقك بالبيئة المحيطة به وذلك من خلال الفئة Application، ومنها مسار تشغيل التطبيق.
- لكتابة مسار ما في جملة نصية نستخدم الرمز @، وذلك عوضاً عن تكرار الرمز \\.
- مع الملفات ومحفوتها، ينصح باستخدام الخاصية NewLine التابعة للفئة Environment عوضاً عن "\n".

بقية الأمور سهلة نوعاً ما ولا تحتاج لشرح..

حاول قراءة الأسطر البرمجية كما في المشاريع السابقة، حاول قراءتها بلغة إنسانية أكثر منها برمجية!



مشروع SpeakToComputer

يمكنك هذا المشروع من التواصل مع كمبيوترك بشكل مبسط جداً، ويتمثل ذلك بإمكانية إعطاء الأوامر الصوتية له وبدوره يستجيب لك وفقاً لأمرك.

يقسم المشروع إلى جزئين: الأول نطق الكلام وقد تم مناقشته في مشروع سابق وما سيأتي هنا ما هو إلا تتمة وتطوير له، والثاني التعرف على الكلمات وذلك من خلال حفظ بعض الكلمات والأفعال في ملفات معينة يقوم التطبيق بفتحها عند بداية عمله، ثم يتعرف على الكلمات الموجودة في هذه الملفات فقط. بالإمكان استخدام معاجم وقواميس جاهزة وتضمن جميع كلمات لغة معينة مثل اللغة الإنجليزية لكن هذا يتطلب مهارة ودقة في نطق الأحرف، وبدلاً من ذلك بإمكانك تصميم الموضوع لتجعل التطبيق يتعرف على كلمات معينة، الأمر الذي لا يتطلب دقة في نطق الأحرف.

الأفكار الموجودة في هذا المشروع يمكن تطويرها للحصول على برنامج خدمي جيد ومتوازن.

- الأدوات التي ستتعامل معها:

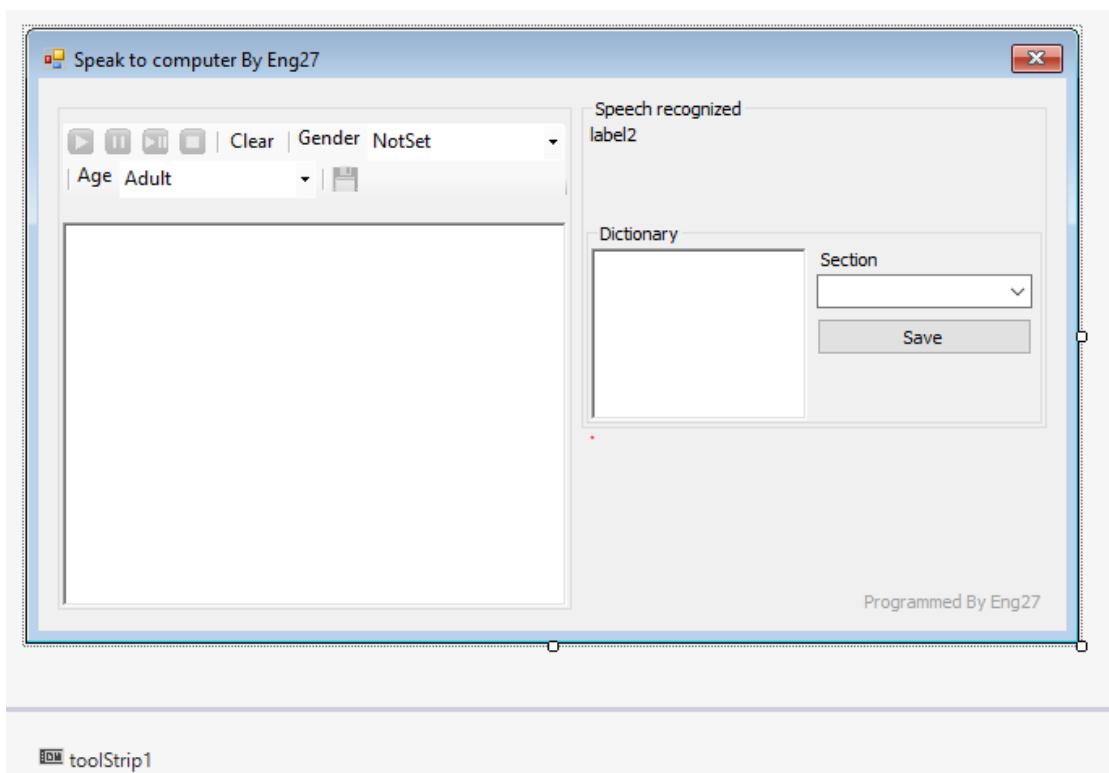
- زر Button
- صندوق اختيار ComboBox
- قائمة Label
- صندوق تجميع GroupBox
- صندوق نص غني RichTextBox
- شريط الأدوات ToolStrip

- المكتبات التي ستتعامل معها:

- System.Speech
- System.Speech.Synthesis
- System.Speech.Recognition
- System.IO
- System.Diagnostics



صمم الواجهة التالية:



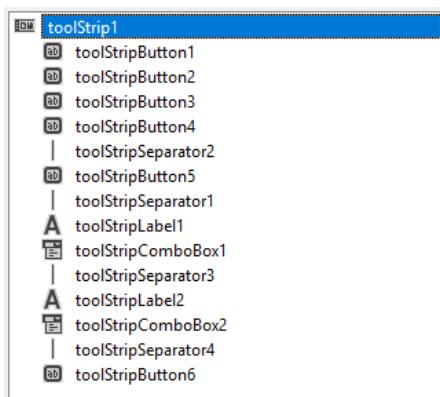
أضف ثلاثة صناديق تجميع **GroupBox**، زر واحد **Button**، لائحتين **ComboBox**، صندوق اختيار واحد **CheckBox**، شريط أدوات واحد **ToolStrip**، ونصين **RichTextBox**.
غنّيّن.

ثم اضبط الخصائص كما يلي:

القيمة	الخاصية	الأداة
False	MaximizeBox	Form1
False	MinimizeBox	
650; 380	MaximumSize	
	MinimumSize	
	Size	
CenterScreen	StartPosition	
Speak to computer By Eng27	Text	
	Text	GroupBox1
Speech recognized	Text	GroupBox2
Dictionary	Text	GroupBox3
Save	Text	Button1



Programmed By Eng27	Text	Label1
label2	Text	Label2
Section	Text	Label3
Verbs Words	Items	ComboBox1
NotSet Male Female Neutral	Items	ToolStripComboBox1
NotSet	Text	
Adult Child NotSet Senior Teen	Items	ToolStripComboBox2
Adult	Text	
False	Enable	ToolStripButton1
False	Enable	ToolStripButton2
False	Enable	ToolStripButton3
False	Enable	ToolStripButton4
Clear	Text	ToolStripButton5
False	Enable	ToolStripButton6
Gender	Text	ToolStripLabel1
Age	Text	ToolStripLabel2



بالنسبة للأزرار الأربع الأولى والزر الأخير، اضبطها على أن تظهر بشكل صورة فقط، أما الزر الخامس فاضبطه على أن يظهر بشكل نص فقط، وذلك من الخاصية `DisplayStyle`.

ثم استخدم الكود التالي:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```



```

using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

// مجالات الأسماء الجديدة اللازمة للمشروع
using System.Speech;
using System.Speech.Synthesis;
using System.Speech.Recognition;
using System.IO;
using System.Diagnostics;

namespace SpeaktoComputer
{
    public partial class Form1 : Form
    {
        // استنساخ فئات القراءة والاستماع
        SpeechSynthesizer Reader = new SpeechSynthesizer();
        SpeechRecognitionEngine Engine = new SpeechRecognitionEngine();

        public Form1()
        {
            InitializeComponent();
            comboBox1.SelectedIndex = 0;
            comboBox1.SelectedIndex = 1;
            Recognizing(); // بدء الاستماع
        }

        private void Recognizing()
        {
            Engine.SetInputToDefaultAudioDevice(); // ضبط جهاز تسجيل الصوت
            Grammar g = MyGrammar(); // تهيئة قواعد اللغة
            Engine.LoadGrammar(g); // تحميل قواعد اللغة
            // الإصغاء للكنة بالوضع المتعدد
            Engine.RecognizeAsync(RecognizeMode.Multiple);
            // جمع الكلام الذي تم الإصغاء إليه وإرساله لحدث النص المُصنَّف إليه
            Engine.SpeechRecognized += new
            EventHandler<SpeechRecognizedEventArgs>(EngineSpeechRecognized);
        }

        // حدث النص المُصنَّف إليه (الذي تم استماعه من الميكروفون)
        private void EngineSpeechRecognized(object sender,
        SpeechRecognizedEventArgs e)
        {
            // تعريف متغيرات سنتعامل معها في حدث الكلام المستمع إليه
            string TextRecognized = e.Result.Text, TextReplaced;
        }
    }
}

```



```

label2.Text = TextRecognized; // إظهار الكلام الذي تم الاستماع إليه
// تنفيذ عمليات معينة بحسب الكلام المستمع إليه
if (TextRecognized.Contains("goto"))
{
    TextReplaced= TextRecognized.Replace("goto ", string.Empty);
    Process.Start("https://www.google.com/#q=" + TextReplaced);
    richTextBox1.Text = string.Empty;
    richTextBox1.Text = "Just a moment and I'll search";
    toolStripButton1_Click(sender, e);
}

if (TextRecognized.Contains("run"))
{
    try
    {
        TextReplaced= TextRecognized.Replace("run ", string.Empty);
        Process.Start(TextReplaced + ".exe");
        richTextBox1.Text = string.Empty;
        richTextBox1.Text = "Ok I'll do sir!";
        toolStripButton1_Click(sender, e)
    }
    catch
    {
    }
}

if (TextRecognized == "hello computer")
{
    richTextBox1.Text = "Hello sir, How are you??";
    toolStripButton1_Click(sender, e);
}

if (TextRecognized == "how are you")
{
    richTextBox1.Text = "I'm fine, and you??";
    toolStripButton1_Click(sender, e);
}

if (TextRecognized.Contains ("clear"))
{
    richTextBox1.Text = string.Empty;
}
}

تابع يقوم بتهيئة قواعد اللغة
private Grammar MyGrammar()
{
    string[] verbs = Arr("Verbs");
    string[] words = Arr("Words");
}

```



```

GrammarBuilder gb = new GrammarBuilder(new Choices(verbs));
gb.Append(new Choices(words));
GrammarBuilder v = new GrammarBuilder(new Choices(verbs));
Grammar g = new Grammar(new Choices(gb, v));
return g;
}
private string [] Arr(string Section)
{
    comboBox1.SelectedItem = Section;
    return richTextBox2.Lines; // إرجاع أسطر صندوق النص كمصفوفة نصية
}
private void toolStripButton5_Click(object sender, EventArgs e)
{
    richTextBox1.Text = string.Empty;
}

// التأكد من أن صندوق النص يحوي عبارات نصية
// إذا لم يحو على نصوص يجب عدم تمكين الأزرار من العمل
private void richTextBox1_TextChanged(object sender, EventArgs e)
{
    if (richTextBox1.Text != "")
    {
        toolStripButton1.Enabled = true;
        toolStripButton2.Enabled = true;
        toolStripButton3.Enabled = true;
        toolStripButton4.Enabled = true;
        toolStripButton6.Enabled = true;
    }
    else
    {
        toolStripButton1.Enabled = false;
        toolStripButton2.Enabled = false;
        toolStripButton3.Enabled = false;
        toolStripButton4.Enabled = false;
        toolStripButton6.Enabled = false;
    }
}

private void toolStripButton1_Click(object sender, EventArgs e)
{
    Reader.Dispose();
    Reader = new SpeechSynthesizer();
    int VoiceGender = toolStripComboBox1.SelectedIndex;
    // استنساخ كائن من معدّد
    System.Speech.Synthesis.VoiceGender Gender = new VoiceGender();
    switch (VoiceGender)
    {

```



```

case 0:
    Gender = System.Speech.Synthesis.VoiceGender.NotSet ;
    break;
case 1:
    Gender = System.Speech.Synthesis.VoiceGender.Male ;
    break;
case 2:
    Gender = System.Speech.Synthesis.VoiceGender.Female ;
    break;
case 3:
    Gender = System.Speech.Synthesis.VoiceGender.Neutral ;
    break;
}

int VoiceAge = toolStripComboBox2.SelectedIndex;
System.Speech.Synthesis.VoiceAge Age = new VoiceAge();
switch (VoiceAge)
{
    case 0:
        Age = System.Speech.Synthesis.VoiceAge.Adult;
        break;
    case 1:
        Age = System.Speech.Synthesis.VoiceAge.Child;
        break;
    case 2:
        Age = System.Speech.Synthesis.VoiceAge.NotSet;
        break;
    case 3:
        Age = System.Speech.Synthesis.VoiceAge.Senior;
        break;
    case 4:
        Age = System.Speech.Synthesis.VoiceAge.Teen;
        break;
}

// القراءة بحسب العمر والجنس
Reader.SelectVoiceByHints(Gender, Age);
Reader.SpeakAsync(textBox1.Text);
}

private void toolStripButton2_Click(object sender, EventArgs e)
{
    if (Reader != null)
    {
        if (Reader.State == SynthesizerState.Speaking) Reader.Pause();
    }
}

```



```
private void toolStripButton3_Click(object sender, EventArgs e)
{
    if (Reader != null)
    {
        if (Reader.State == SynthesizerState.Paused)
            Reader.Resume();
    }
}

private void toolStripButton4_Click(object sender, EventArgs e)
{
    if (Reader != null)
    {
        Reader.Dispose();
    }
}

// حفظ التسجيل الصوتي بملف
private void toolStripButton6_Click(object sender, EventArgs e)
{
    SaveFileDialog SaveDialog = new SaveFileDialog();
    SaveDialog.Filter = "Wav Files | *.wav";
    SaveDialog.Title = "Save voice to a file";
    if (SaveDialog.ShowDialog() == DialogResult.OK)
    {
        FileStream file = new FileStream(SaveDialog.FileName,
            FileMode.Create,
            FileAccess.Write);
        Reader.SetOutputToWaveStream(file);
        Reader.Speak(richTextBox1.Text);
    }
}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    try
    {
        richTextBox2.Lines = File.ReadAllLines(Application.StartupPath +
            @"/" +
            comboBox1.Text +
            ".dat");
    }

    catch
    {
        var file = File.Create(Application.StartupPath +
            @"/" +
```

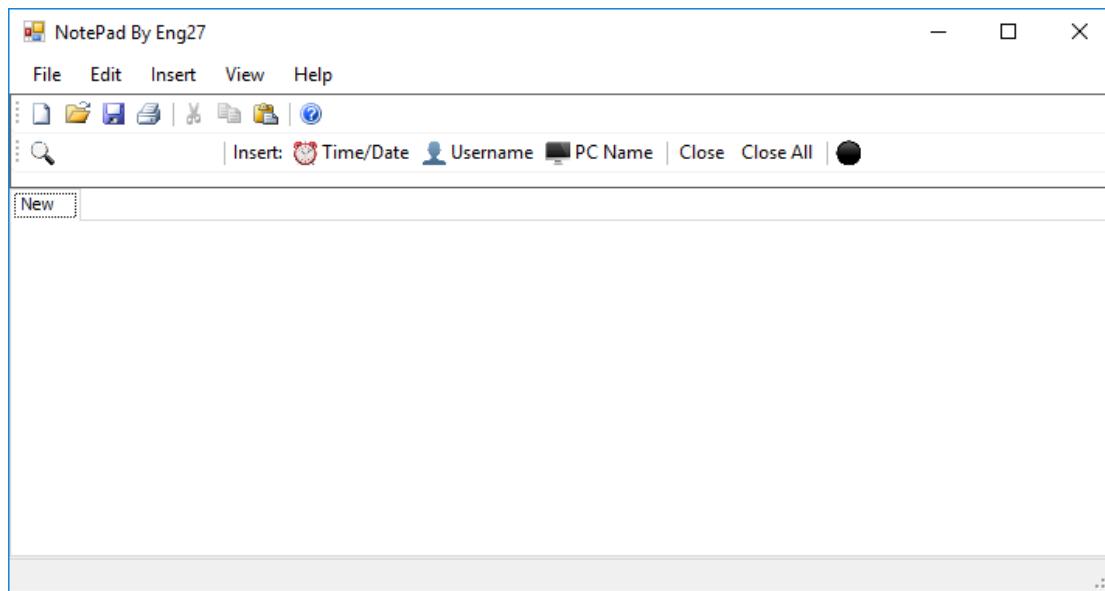


```
comboBox1.Text +  
    ".dat");  
 بهذه الطريقة نغلق الملف بعد فتحه //;  
 File.WriteAllText(Application.StartupPath +  
     @"/" +  
     comboBox1.Text +  
     ".dat",  
     "Write your " +  
     comboBox1.Text +  
     " here!");  
 }  
 }  
 private void richTextBox2_TextChanged(object sender, EventArgs e)  
 {  
     if (!richTextBox2.Lines.Contains (""))  
         File.WriteAllLines(Application.StartupPath +  
             @"/" +  
             comboBox1.Text +  
             ".dat",  
             richTextBox2.Lines);  
 }  
 private void button1_Click(object sender, EventArgs e)  
 {  
     MessageBox.Show("The Application will restart to save changes!",  
         "Save to dictionary!",  
         MessageBoxButtons.OK ,  
         MessageBoxIcon.Information ); Application.Restart();  
 }  
 private void Form1_Load(object sender, EventArgs e)  
 {  
     if (DateTime.Now.Hour <= 12 && DateTime.Now.Hour >= 6)  
         richTextBox1.Text = "Good morning sir!";  
     else if (DateTime.Now.Hour >= 12 && DateTime.Now.Hour <= 14)  
         richTextBox1.Text = "Good afternoon sir!";  
     else if (DateTime.Now.Hour >= 17 && DateTime.Now.Hour <= 19)  
         richTextBox1.Text = "Good evening sir!";  
     else  
         richTextBox1.Text = "How are you sir!";  
     toolStripButton1_Click(sender, e);  
 }  
 }  
 }
```



مشروع TabbedNotePad

يعطيك هذا المشروع لمحة عن كيفية التعامل مع محررات النصوص ومبدأ عملها وبعض مزاياها، ومن خلاله بإمكانك إنشاء محرر نصوص كامل بحيث يحتوي على الكثير من المزايا المطلوبة في أي محرر نصوص.



يحتوي المشروع على مجموعة من القوائم التي تسمح للمستخدم بالتعامل مع الملفات والتعديل على المحتوى النصي وإمكانية إضافة بعض الأدوات بالإضافة إلى قوائم المساعدة والعرض. كما يتيح لك إضافة أشرطة أدوات لأداء وظائف مختلفة، مع إمكانية الحصول على بعض التفاصيل من خلال شريط الحالة .StatusBar.

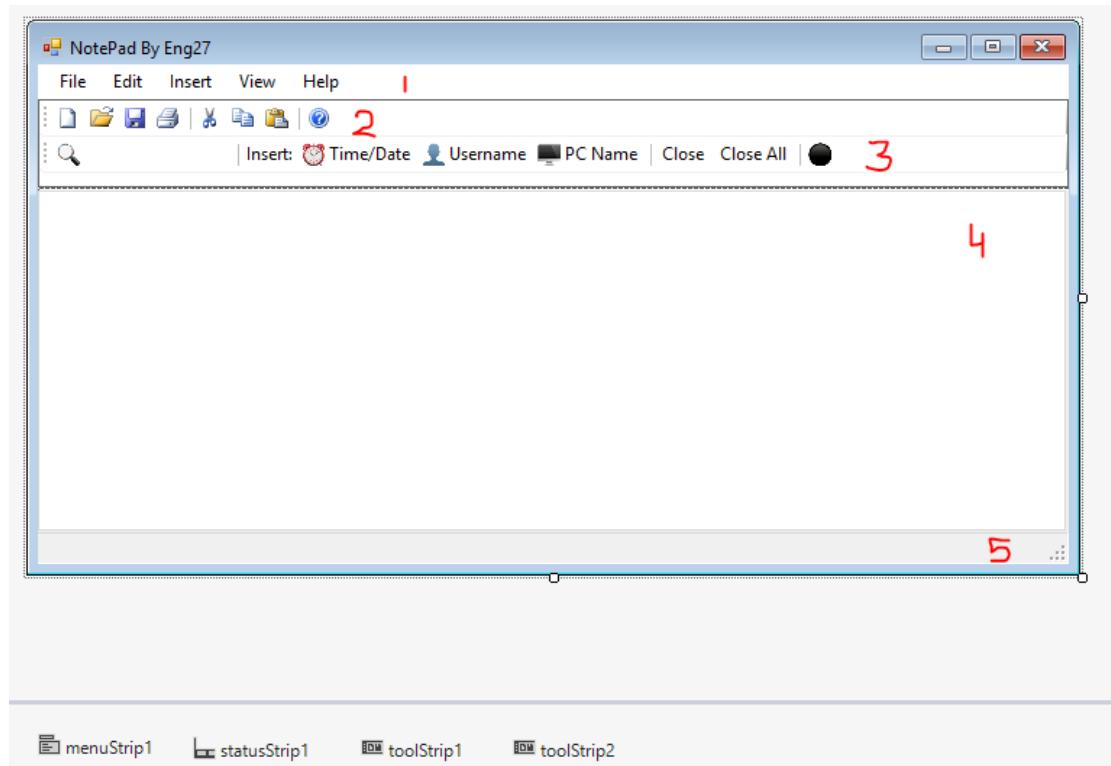
ابتدأً من هذا المشروع ومروراً بجميع المشاريع التي تليه لن يتم بيان خصائص الأدوات المستخدمة كما في المشاريع السابقة، وذلك لأنك أصبحت كبيراً الآن - أكثر من كل مرة قلت لك بأنك صرت كبيراً فيها - وبات بإمكانك اكتشاف خصائص الأدوات المستخدمة لوحدهك.. ولكن في بعض الأحيان قد يتم توضيح بعض الخصائص لأهميتها أو لأنها غير واضحة بشكل مباشر.

هذا وسيتم استخدام هذا الأسلوب مع الأكواد كذلك الأمر، فلن يتم شرح إلا الشيء الهام والحيوي في المشروع (كما لاحظت في آخر مشروع)، وسيترك لك مهمة اكتشاف جميع وظائف ومهام الأكواد المستخدمة، حيث أن شرح الأكواد كان عن كل شاردة وواردة في بداية الكتاب - وتحديداً في بداية هذا الفصل - لأنه تم افتراض أنك جديد على المشاريع، ولإعطاءك

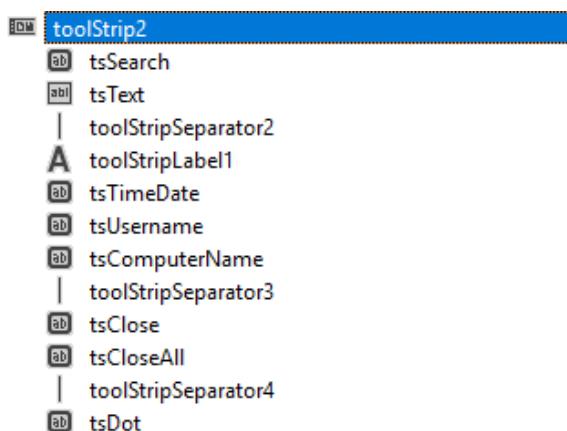


الأسلوب الذي يناقِش ويحلّل فيه الكتابُ المسائلَ. والآن عليك التفكير في المسائل بالأسلوب الذي ترتاح له والأقرب لتفكيرك واهتمامك.

صمم النافذة التالية وفق ماتراه مناسباً:



في البداية تم إضافة أداتي Panel وهي من أدوات التجميع. حيث تم ضبط خاصية Dock للأولى على Top وـBorderStyle على FixedSingle، والخاصية Dock للثانية على Fill وذلك بعد ضبط خصائص الأداة الأولى. ثم تم إضافة شريط أدوات ToolStrip، والمشار له بالرقم 2 وذلك في اللوحة الأولى Panel1. حيث تم إدراج الأوامر القياسية للشريط الأول وذلك بالضغط عليه بالزر الأيمن ثم Insert Standard Items. أما الثاني والمشار إليه بالرقم 3 فقد تم إضافة العناصر يدوياً وهي كما يلي:





حيث تم تغيير الأسماء البرمجية Name لجميع العناصر إلى المهمة التي سيؤديها العنصر.

ملاحظة

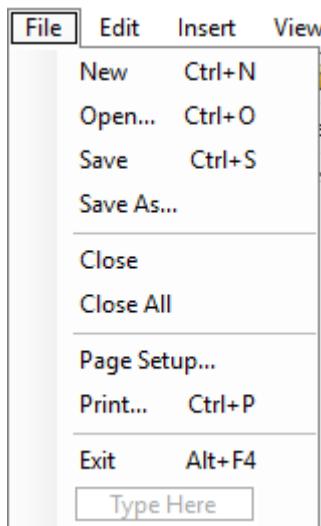
- يمكن ضبط صورة للعنصر في شريط الأدوات ToolStrip بالضغط عليه باليمين ثم Set Image.
- يمكن تهيئة العنصر من نوع Button في شريط الأدوات من خلال DisplayStyle.

ثم تم إضافة أداة TabControl ومسح تبويباتها (لذلك فالشاشة فارغة في الصورة في المنقطة المشار إليها بالرقم 4).

وبعدها تم إضافة شريط الحالة المشار إليه بالرقم 5.

وأخيراً تم إضافة القوائم، والمشار إليها بالرقم 1 وهي كما يلي:

القائمة File: والتي تتعامل مع الملفات:



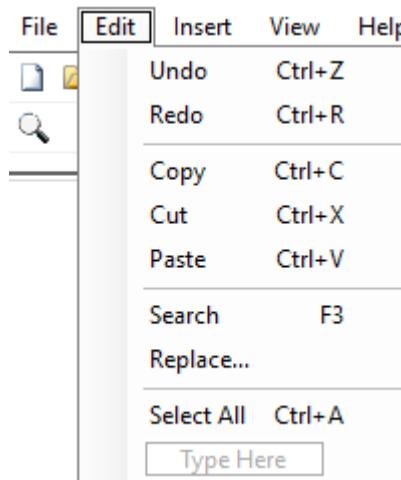
طبعاً تم تغيير الأسم البرمجي Name لجميع العناصر هنا أيضاً لتصبح الأكواد أسهل (صدقني ستتصبح الأكواد شوربة لو تركنا الأسماء الافتراضية، فجميعها متشابهة!).

ملاحظة

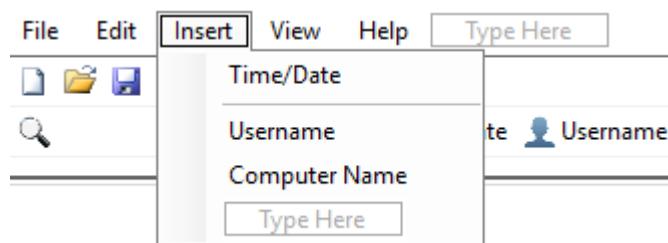
- يتم عادةً إضافة ثلاثة نقاط أمام الأوامر أو القوائم التي ستُظهر نوافذ جديدة.



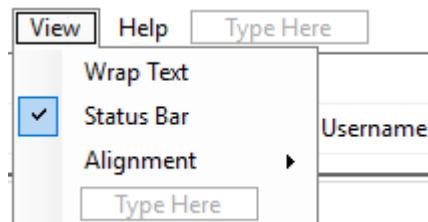
القائمة Edit: والتي تتعامل مع المحتوى النصي:



القائمة Insert: والتي تقوم بإضافة بعض المتغيرات إلى المحتوى النصي:

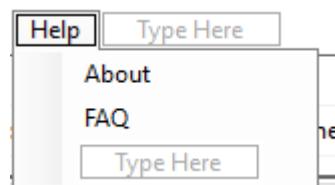


القائمة View: والتي تعطي المستخدم إمكانيات عرض مختلفة:



فعّل الخاصية CheckOnClick لكل من القائمتين الفرعيتين .WrapText و .Statusbar، وفعل الخاصية Checked للقائمة الفرعية StatusBar.

القائمة Help: والتي تشرح للمستخدم عن البرنامج:





ثم استخدم الأمر التالي:

```
المتغيرات التي ستلزمنا على امتداد المشروع//  
RichTextBox RTB;  
TabPage TB;  
public Form1()  
{  
    InitializeComponent();  
}  
  
private void Form1_Load(object sender, EventArgs e)  
{  
    عند تحميل النموذج لابد من إنشاء صفحة جديدة والتحقق من إمكانية التعديل//  
    New_Click(sender, e);  
    CanEdit();  
}  
  
private RichTextBox GetRichTextBox()  
{  
    هذا التابع يعطيك إمكانية الوصول لصندوق النص في التبويبية الحالية//  
    RTB = null;  
    TB = tabControl1.SelectedTab;  
    if (TB != null)  
        RTB = TB.Controls[0] as RichTextBox;  
    return RTB;  
}  
  
private void New_Click(object sender, EventArgs e)  
{  
    إنشاء تبويبية جديدة//  
    TB = new TabPage("New");  
    RTB = new RichTextBox();  
    RTB.Dock = DockStyle.Fill;  
    RTB.BorderStyle = BorderStyle.None;  
    TB.Controls.Add(RTB);  
    tabControl1.TabPages.Add(TB);  
}  
  
private void CanEdit()  
{  
    هذا الإجراء يتحقق من إمكانيات التعديل (النسخ واللصق والقص و...)//  
    يمكن التعديل في حال كان هناك قوائم//  
    if (tabControl1.TabPages.Count > 0)  
    {  
        if (GetRichTextBox().SelectedText == string.Empty)  
        {  
            إذا لم يكن هناك نص محدد لا يمكن النسخ ولا القص//  
            Copy.Enabled = false;
```



```

Cut.Enabled = false;
tsCopy.Enabled = false; // الأمر الموجود في شريط الأدوات
tsCut.Enabled = false; // الأمر الموجود في شريط الأدوات
}
else
{
    Copy.Enabled = true;
    Cut.Enabled = true;
    tsCopy.Enabled = true;
    tsCut.Enabled = true;
}

if (Clipboard.ContainsText())
{
    إذا كان هناك نص في حافظة النظام يمكن اللصق //
    Paste.Enabled = true;
    tsPaste.Enabled = true;
}
else
{
    Paste.Enabled = false;
    tsPaste.Enabled = false;
}

إذا كان هناك إمكانية تراجع يمكن التراجع //
Undo.Enabled = true;
else
    Undo.Enabled = false;

إذا كان هناك إمكانية تكرار يمكن التكرار //
Redo.Enabled = true;
else
    Redo.Enabled = false;
}

الأوامر الأساسية//
private void Copy_Click(object sender, EventArgs e)
{
    GetRichTextBox().Copy(); // لاحظ للوصول إلى النص الحالي نستدعي التابع
}

private void Paste_Click(object sender, EventArgs e)
{
    GetRichTextBox().Paste();
}

```



```
private void Cut_Click(object sender, EventArgs e)
{
    GetRichTextBox().Cut();
}

private void Undo_Click(object sender, EventArgs e)
{
    GetRichTextBox().Undo();
}

private void Redo_Click(object sender, EventArgs e)
{
    GetRichTextBox().Redo();
}

private void SelectAll_Click(object sender, EventArgs e)
{
    GetRichTextBox().SelectAll();
}

private void Search_Click(object sender, EventArgs e)
{
    tsText.Focus(); // البحث يتم التركيز على صندوق البحث
}

private void Exit_Click(object sender, EventArgs e)
{
    Application.Exit();
}

private void StatusBar_CheckedChanged(object sender, EventArgs e)
{
    if (StatusBar.Checked)
        statusStrip1.Visible = true;
    else
        statusStrip1.Visible = false;
}

private void WrapText_CheckedChanged(object sender, EventArgs e)
{
    if (WrapText.Checked)
        GetRichTextBox().WordWrap = true;
    else
        GetRichTextBox().WordWrap = false;
}
```



```
private void Edit_DropDownOpened(object sender, EventArgs e)
{
    CanEdit(); // عند فتح القائمة تعديل سيتم التحقق من إمكانيات التعديل الممكنة
}

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    // عند إغلاق النموذج يتم تأكيد الخروج
    DialogResult result = MessageBox.Show("Do you want really exit?", "Exit", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
    if (result == DialogResult.No)
        e.Cancel = true;
}

private void RL_Click(object sender, EventArgs e)
{
    // تحويل اتجاه النافذة والقوائم والنص إلى العربي
    GetRichTextBox().RightToLeft = RightToLeft.Yes;
    this.RightToLeft = RightToLeft.Yes;
    this.RightToLeftLayout = true;
    // عليك إضافة قاعدة بيانات تغيير الأسماء وجميع العبارات والرسائل إلى العربية
}

private void LR_Click(object sender, EventArgs e)
{
    // تحويل اتجاه النافذة والقوائم والنص إلى الإنجليزي
    GetRichTextBox().RightToLeft = System.Windows.Forms.RightToLeft.No;
    this.RightToLeft = System.Windows.Forms.RightToLeft.No;
    this.RightToLeftLayout = false;
    // عليك إضافة قاعدة بيانات تغيير الأسماء وجميع العبارات والرسائل إلى الإنجليزية
}

private void TimeAndDate_Click(object sender, EventArgs e)
{
    GetRichTextBox().AppendText(" " + DateTime.Now.ToString() + " ");
}

private void UserName_Click(object sender, EventArgs e)
{
    GetRichTextBox().AppendText(" " + Environment.UserName + " ");
}

private void ComputerName_Click(object sender, EventArgs e)
{
    GetRichTextBox().AppendText(" " + Environment.MachineName + " ");
}
```



```
private void Close_Click(object sender, EventArgs e)
{
    if (tabControl1.TabPages.Count > 0)
    {
        إذا كان هناك تبويبات سيتم مسح التبويبة الحالية //  

        int SelectedTab = tabControl1.SelectedIndex;  

        tabControl1.TabPages.Remove(tabControl1.SelectedTab);  

        tabControl1.SelectedIndex = SelectedTab;
    }
}

private void CloseAll_Click(object sender, EventArgs e)
{
    while (tabControl1.TabPages.Count > 0)
    {
        مسح جميع التبويبات، ستتكرر عملية المسح طالماها هناك تبويبات //  

        tabControl1.TabPages.Remove(tabControl1.SelectedTab);
    }
}

// أوامر شريط الأدوات //  

// أغلاقها ستنفذ أوامر أخرى //  

private void tsUsername_Click(object sender, EventArgs e)
{
    UserName_Click(sender, e);
}

private void tsComputerName_Click(object sender, EventArgs e)
{
    ComputerName_Click(sender, e);
}

private void tsNew_Click(object sender, EventArgs e)
{
    New_Click(sender, e);
}

private void tsOpen_Click(object sender, EventArgs e)
{
    Open_Click(sender, e);
}

private void tsSave_Click(object sender, EventArgs e)
{
    Save_Click(sender, e);
}
```



```
private void tsPrint_Click(object sender, EventArgs e)
{
    Print_Click(sender, e);
}

private void tsCut_Click(object sender, EventArgs e)
{
    Cut_Click(sender, e);
}

private void tsCopy_Click(object sender, EventArgs e)
{
    Copy_Click(sender, e);
}

private void tsPaste_Click(object sender, EventArgs e)
{
    Paste_Click(sender, e);
}

private void tsClose_Click(object sender, EventArgs e)
{
    Close_Click(sender, e);
}

private void tsCloseAll_Click(object sender, EventArgs e)
{
    CloseAll_Click(sender, e);
}

private void tsSearch_Click(object sender, EventArgs e)
{
    Search_Click(sender, e);
}

private void tabControl1_SelectedIndexChanged(object sender, EventArgs e)
{
    GetRichTextBox().Focus(); // نصها
    // عند تغيير التبويبية يجب التركيز على نصها
}

private void toolStrip1_MouseMove(object sender, MouseEventArgs e)
{
    // عند المرور فوق شريط الأدوات الأول سيتم التحقق من إمكانني التحرير
    // CanEdit();
}
```



```

private void tsText_TextChanged(object sender, EventArgs e)
{
    // عند تغيير نص البحث
    int index = 0;
    string temp = GetRichTextBox().Text;
    GetRichTextBox().Text = "";
    GetRichTextBox().Text = temp;
    while (index < GetRichTextBox().Text.LastIndexOf(tsText.Text))
    {
        GetRichTextBox().Find(tsText.Text,
            index,
            GetRichTextBox().TextLength,
            RichTextBoxFinds.None);
        GetRichTextBox().SelectionBackColor = Color.Cyan;
        index = GetRichTextBox().Text.IndexOf(tsText.Text, index) + 1;
    }
}

private void tsDot_Click(object sender, EventArgs e)
{
    // إضافة نقطة تعداد إلى بداية الجملة النصية أو إزالتها
    if (tsDot.Checked)
    {
        GetRichTextBox().SelectionBullet = true;
        tsDot.Checked = true;
    }
    else
    {
        GetRichTextBox().SelectionBullet = false;
        tsDot.Checked = false;
    }
    // عليك إنشاء حدث تغيير نص صندوق النص الغني، والذي هو تابع!
}

private void tsText_Leave(object sender, EventArgs e)
{
    string temp = GetRichTextBox().Text;
    GetRichTextBox().Text = "";
    GetRichTextBox().Text = temp;
}

private void tsTimeAndDate_Click(object sender, EventArgs e)
{
    TimeAndDate_Click(sender, e);
}

```



جميع الأكواد التالية سيتم شرحها في فصل آخر//

```
private void Open_Click(object sender, EventArgs e)
{
}

private void Save_Click(object sender, EventArgs e)
{
}

private void SaveAs_Click(object sender, EventArgs e)
{
}

private void PageSetup_Click(object sender, EventArgs e)
{
}

private void Print_Click(object sender, EventArgs e)
{}
```

هناك بعض الأوامر والأكواد لم يتم وضعها في الكود مثل أكواد الطباعة أو فتح وحفظ الملفات أو المساعدة أو الاستبدال أو شريط الحالة... لأن المشروع يركز على أوامر تعديل النصوص فقط وكيفية التعامل مع محرر نصوص مبوب .TabbedNotepad



مشروع WebBrowser

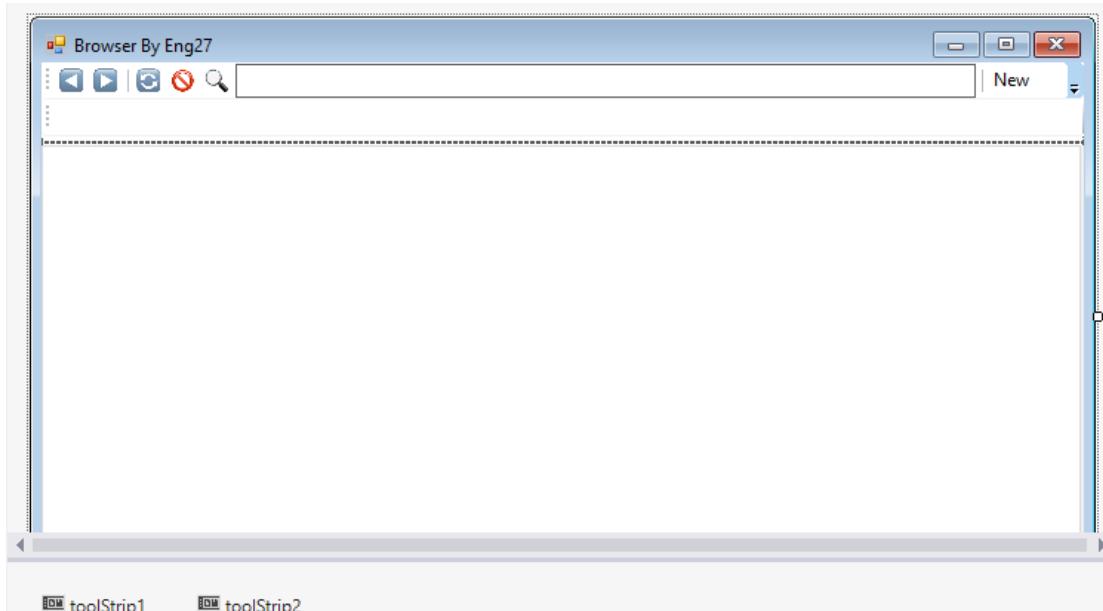
على غرار المشروع السابق فإننا سنتعامل مع تبويبات بحيث يمكن المستخدم من استخدام برنامجك لتصفح أكثر من صفحة في الوقت ذاته.

طبعاً الفكرة من المشروع هو التعامل مع الإجراءات الأساسية في متصفح الويب، كالتحديث والتالي والسابق وفتح تبويبات جديدة، بالإضافة إلى إمكانية حفظ صفحات الانترنت المفضلة.



طبعاً المشروع يحتاج الكثير والكثير من العناية والتحديث والتطوير لينتفيأ منه، كإمكانية حفظ سجل النشاط أو تنزيل الملفات أو غيرها من الميزات التي تجعل من متصفحك متصفحًا مطلوباً أو مفيداً على أقل تقدير!

صمم النافذة التالية، والتي تحتوي على أداتي Panel كما في مشروع المفكرة المبوبة - واحدة لها خاصية Dock للأعلى والثانية لها ذات الخاصية على وضع الإملاء - بالإضافة إلى شريطي أدوات ToolStrip الأول لوضع أزرار الأوامر عليه والثاني لوضع الصفحات المفضلة للمستخدم فيه:



استخدم الكود التالي:

```
متغيرات ستلزمنا//  
WebBrowser WB;  
TabPage TB;  
ToolStripButton TSB;  
  
public Form1()  
{  
    InitializeComponent();  
}  
  
private WebBrowser GetWebBrowser()  
{  
    // هذا التابع يعطيك إمكانية الوصول لمتصفح الويب في التبويبة الحالية//  
    WB = null;  
    TB = tabControl1.SelectedTab;  
    if (TB != null)  
        WB = TB.Controls[0] as WebBrowser;  
    return WB;  
}  
  
private void GetToolStripButton()  
{  
    // هذا التابع يعطيك إمكانية الوصول للزر المضغوط من شريط الأدوات الثاني//  
    TSB = null;  
    TSB = new ToolStripButton();  
    TSB.Text = GetWebBrowser().DocumentTitle;  
    TSB.ToolTipText = GetWebBrowser().Url.ToString();  
    toolStrip2.Items.Add(TSB);
```



```

إنشاء حدث مرتبط بالزر، ويتفجر عندما يتم الضغط عليه//  

toolStrip2.ItemClicked += toolStrip2_ItemClicked;  

}  
  

void toolStrip2_ItemClicked(object sender, ToolStripItemClickedEventArgs e)  

{  

    عند الضغط على الزر فإنه سيتم الانتقال على الموقع الممثل بخاصية تلميح الزر//  

    GetWebBrowser().Navigate(e.ClickedItem.ToolTipText);  

}  
  

private void Form1_Load(object sender, EventArgs e)  

{  

    إنشاء تبويب جديد عند تحميل النموذج //  

    GetWebBrowser().Navigate("https://www.google.com");  

    عند اكتمال تحميل الصفحة، سيتغير اسم التبويب إلى اسم الصفحة//  

    ولذلك علينا إنشاء حدث مرتبط بانتهاء تحميل الصفحة//  

    GetWebBrowser().DocumentCompleted += Form1_DocumentCompleted;  

}  
  

private void Form1_DocumentCompleted(object sender,  

WebBrowserDocumentCompletedEventArgs e)  

{  

    عند اكتمال تحميل الصفحة سيتفجر هذا الحدث//  

    tabControl1.SelectedTab.Text = GetWebBrowser().DocumentTitle;  

}  
  

private void tsNew_Click(object sender, EventArgs e)  

{  

    TB = new TabPage("New");  

    WB = new WebBrowser();  

    WB.Dock = DockStyle.Fill;  

    TB.Controls.Add(WB);  

    tabControl1.TabPages.Add(TB);  

    tabControl1.SelectedIndex = tabControl1.TabCount - 1;  

    GetWebBrowser().Navigate("https://www.google.com");  

    GetWebBrowser().DocumentCompleted += Form1_DocumentCompleted;  

}  
  

private void tsSearch_Click(object sender, EventArgs e)  

{  

    if(tsText.Text.Contains(".com") ||  

        tsText.Text.Contains(".net") ||  

        tsText.Text.Contains(".org"))  

        GetWebBrowser().Navigate(tsText.Text);  

    else  

        GetWebBrowser().Navigate("https://www.google.com/search?q=" +  

        tsText.Text);}  

}

```



```
private void tsAddToFav_Click(object sender, EventArgs e)
{
    GetToolStripButton();
}

private void tsBack_Click(object sender, EventArgs e)
{
    if (GetWebBrowser().CanGoBack)
        GetWebBrowser().GoBack();
}

private void tsText_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyValue == 13) tsSearch_Click(sender, e);
}

private void tsStop_Click(object sender, EventArgs e)
{
    GetWebBrowser().Stop();
}

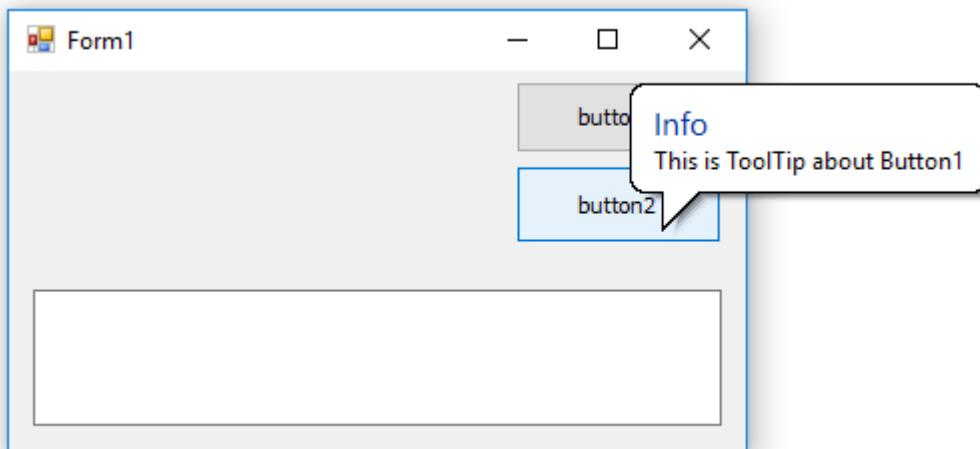
private void tsReload_Click(object sender, EventArgs e)
{
    GetWebBrowser().Refresh();
}

private void tsForward_Click(object sender, EventArgs e)
{
    if (GetWebBrowser().CanGoForward)
        GetWebBrowser().GoForward();
}
```



مشروع PopupMessage

توفر لك C# إمكانية عرض تلميحات عند مرور مؤشر الفأرة على أحد مكونات نافذة برنامجك وذلك عبر الأداة ToolTip كما يلي:



وفي هذا المشروع سنقوم بإنشاء رسالة منبثقة مع تفاصيل إضافية مثل إمكانية إضافة صور أو غيرها من التفاصيل.

أنشئ مشروعًا من نوع Class وسمه PopupDLL، ثم استخدم الكود التالي:

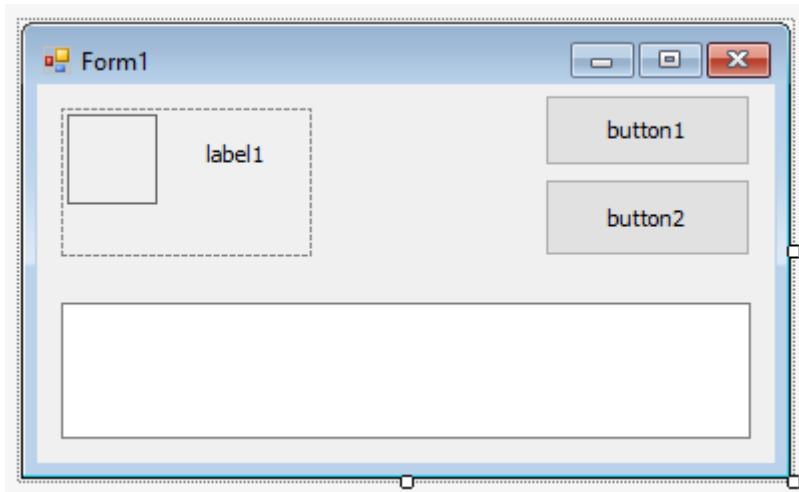
```
using System;
using System.Windows.Forms;
namespace PopupDLL
{
    class Popup
    {
        //خصائص الفئة
        private Panel cPanel;
        public Panel Panel
        {
            get { return cPanel; }
            set { cPanel = value; }
        }
        private Label cLabel;
        public Label Label
        {
            get { return cLabel; }
            set { cLabel = value; }
        }
        private PictureBox cPicture;
        public PictureBox Picture
```



```
{
    get { return cPicture; }
    set { cPicture = value; }
}
//تتابع البناءة للفئة
public Popup()
{
}
public Popup(Panel Panel, Label Label, PictureBox Picture)
{
    cPanel = Panel;
    cLabel = Label;
    cPicture = Picture;
}
//إجراءات الفئة
public void Load()
{
    cPanel.BringToFront();
    cPanel.BorderStyle = BorderStyle.FixedSingle;
    cLabel.AutoSize = true;
    cLabel.AutoEllipsis = true;
    cPicture.Left = cPicture.Top = 3;
    cPicture.SizeMode = PictureBoxSizeMode.StretchImage;
    cPicture.Height = cPicture.Width = 45;
    cPanel.Visible = false;
}
public void MouseMove(Control Control, int x, int y, string Text)
{
    cPanel.Visible = true;
    cPanel.Left = Control.Left + x + 15;
    cPanel.Top = Control.Top + y + 15;
    cPanel.Width = cLabel.Width + 60;
    int H;
    if (cPicture.Height < cLabel.Height) H = cLabel.Height;
    else H = cPicture.Height;
    cPanel.Height = H + 10;
    cLabel.Text = Text;
    cLabel.Top = 0; cLabel.Left = 55;
    Control.MouseLeave += Control_MouseLeave;
}
void Control_MouseLeave(object sender, EventArgs e)
{
    cPanel.Visible = false;
    cPicture.Image = null;
}
}
```



ثم أنشئ مشروعًا من نوع WindowsFormApplications، وصمم النافذة التالية:



اضغط باليمنى على النموذج ثم View Code، واتكتب في محرر الأكواد ما يلى:

```
using System;
using System.Windows.Forms;
using PopupDLL;
namespace Popup_test
{
    public partial class Form1 : Form
    {
        Popup p = new Popup();
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            p.Label = label1;
            p.Panel = panel1;
            p.Picture = pictureBox1;
            p.Load();
        }

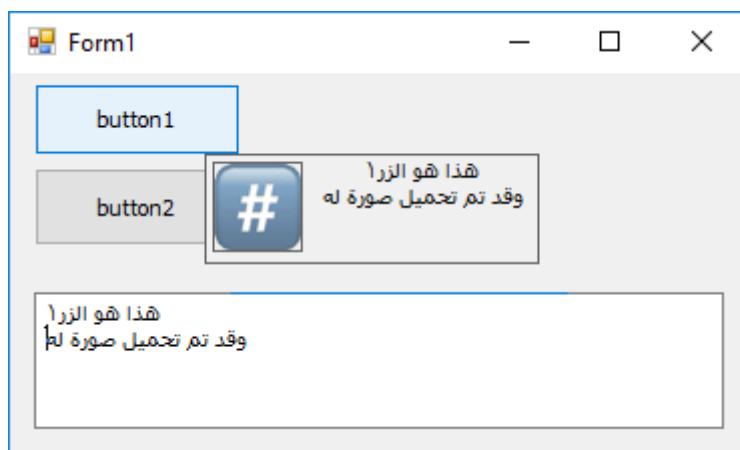
        private void button1_MouseMove(object sender, MouseEventArgs e)
        {
            p.Picture.Image = Properties.Resources.e0001;
            p.MouseMove(button1, e.X, e.Y, textBox1.Text);
        }
    }
}
```



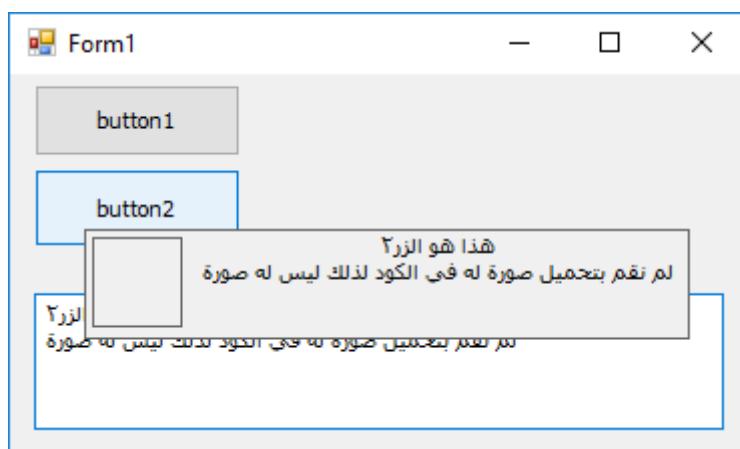
```
private void button2_MouseMove(object sender, MouseEventArgs e)
{
    //p.MouseMove(button2, e.X, e.Y, textBox1.Text);
}
}
```

قبل تشغيل البرنامج قم بإضافة صورة ما ولتكن باسم 0001.jpg إلى مصادر المشروع.

بعد تشغيل البرنامج:



ومن أجل الزر الثاني:



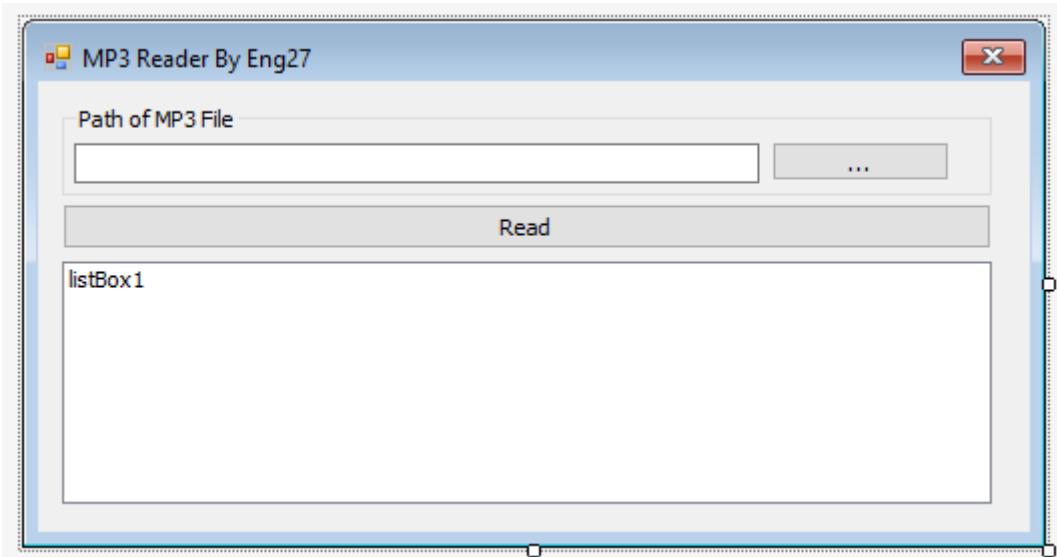
طبعاً بإمكانك تغيير لون الخط أو الخلفية أو إضافة مكونات أو إزالتها من الرسالة كإدراج الصور أو الجداول أو الفيديوهات أو أية تفاصيل أخرى، وستستفيد من هكذا فكرة في البرامج التي تحوي قواعد بيانات ضخمة بحيث تحصل على تفاصيل المكونات الحالية بمجرد مرور مؤشر الفأرة على أحد عناصر قاعدة البيانات.



مشروع MP3Reader

يعطيك هذا المشروع فكرة عن كيفية قراءة الملفات ببياناته الأولية، ويمكن استثماره ب مجالات عديدة منها البحث المتقدم وغيرها.

صمم النافذة التالية:



استخدم الكود التالي لاستعراض الملفات من نوع MP3 :

```
private void button1_Click(object sender, EventArgs e)
{
    OpenFileDialog open = new OpenFileDialog();
    open.Filter = "MP3 Files (*.mp3) | *.mp3";
    if (open.ShowDialog() == DialogResult.OK)
        textBox1.Text = open.FileName;
}
```

والكود التالي لقراءة الملفات:

```
private void button2_Click(object sender, EventArgs e)
{
    const int size = 128;
    byte[] data;
    string name = textBox1.Text;
    if (!File.Exists(name))
    {
        MessageBox.Show("No MP3 File");
        return;
    }
}
```



```
try
{
    FileStream file = File.Open(name, FileMode.Open);
    data = new byte[size];

    file.Seek(-128, SeekOrigin.End);
    file.Read(data, 0, size);

    file.Close();

    byte b1 = data[0];
    byte b2 = data[1];
    byte b3 = data[2];

    if (Convert.ToChar(b1) != 'T' || Convert.ToChar(b2) != 'A' ||
Convert.ToChar(b3) != 'G')
    {
        MessageBox.Show("Invalid MP3 File");
        return;
    }

    int i = 3;
    string title = "";
    for (; i < 33; i++)
    {
        if (data[i] != 0)
            title += Convert.ToChar(data[i]);
    }

    string author = "";
    for (i = 33; i < 63; i++)
    {
        if (data[i] != 0)
            author += Convert.ToChar(data[i]);
    }

    string album = "";
    for (i = 63; i < 93; i++)
    {
        if (data[i] != 0)
            album += Convert.ToChar(data[i]);
    }

    string year = "";
    for (i = 93; i < 97; i++)
    {
        if (data[i] != 0)
```



```
        year += Convert.ToChar(data[i]);
    }

    string comments = "";
    for (i = 97; i < 127; i++)
    {
        if (data[i] != 0)
            comments += Convert.ToChar(data[i]);
    }
    listBox1.Items.Clear();
    listBox1.Items.Add("title : " + title);
    listBox1.Items.Add("author : " + author);
    listBox1.Items.Add("album : " + album);
    listBox1.Items.Add("year : " + year);
    listBox1.Items.Add("comments : " + comments);
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
```

لا تنسى تضمين مجال الأسماء System.IO



مشروع تحويل الأعداد إلى كلمات

كخاتمة لهذا الكتاب أحببت أن أنهيه بهذا المشروع، والذي يستخدم بكثرة في برامج المحاسبة والبرامج الأدارية.

وકأمثلة على المشروع (لتدخل إلى جو الفكرة منه):

- العدد 125 يُكتب مئة وخمسة وعشرون.
- العدد 1005 يُكتب ألف وخمسة.
- العدد 6523544 يُكتب ست ملايين وخمس مائة وثلاث وعشرون ألف وخمس مائة وأربعة وأربعون. وهكذا..

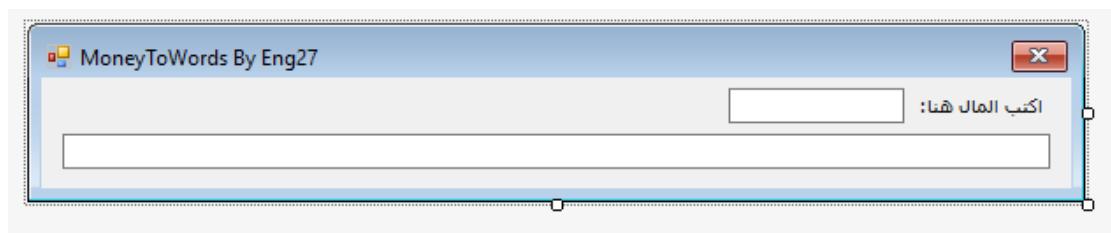
هذا المشروع سيقوم بهذا عنك .

ملاحظة

- الرقم هو حرف أو رمز يعبر عن مجموعة من الأعداد وفق نظام عد ما، وهي 0، 1، 2، 3، 4، 5، 6، 7، 8، 9 في نظام العد العشري. وهي عبارة عن أبجدية الأعداد.
- العدد هو مجموعة أرقام تكتب مع بعضها لتدل على قيمة ما، مثل 10، 600، 1000 وغيرها الكثير الكثير من الأعداد. فالإعداد أشبه بالكلمات إذا ما اعتبرنا الأرقام حروف.
- مشروعنا هذا يأخذ العدد ويحلله لأرقامه التي شكلته ثم يدرس قيمة كل خانة مع أخرى، ويدرس كيفية تسميتها مع بعضها.

ويعتمد هذا المشروع على تحليل العدد المدخل إلى أرقامه الأولية، ثم معرفة اسم الخانة الموافقة لكل رقم وفق الخانة التي يشغلها.

صمم الواجهة التالية:



استخدم الكود التالي:

```
using System;
using System.Windows.Forms;
```



```

namespace MoneyToWords
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        static string ConvertCurrency(string Money, string FullCurrency, string
PartCurrency)
        {
            string MA = " " + FullCurrency;
            string MI = " " + PartCurrency;
            string X = Money;
            بدون فاصلة // بدون فاصلة
            double B = Math.Round(double.Parse(X), 2);
            double B2 = B - Math.Floor(B); B2 *= 100; // القروش
            string r = SConvertCurrency(N);
            string t;
            t = r.ToString() + MA + " و " + Math.Round(Math.Ceiling(B2), 0) + MI;
            return t
        }

        static string SConvertCurrency(double Money)
        {
            تحويل المال إلى عبارة نصية // وذلك لتحويلها إلى مصفوفة حرفية في السطر التالي لتحليلها إلى أرقامها //
            char[] MoneyChars = C.ToCharArray();
            ثم عكسها // عدد أرقام العدد
            int ArrayCount = cMoney.Count(); // متغير سيحمل نتيجة العملية
            string R = "";
            for (int i = 1; i <= ArrayCount; ++i)
            {
                if (i == 1)
                {
                    R = MoneyOnes(cMoney);
                }
                if (i == 2)
                {
                    if (cMoney[0] != char.Parse("0") &
                        cMoney[1] > char.Parse("1"))
                        R = MoneyOnes(cMoney) + " و " + MoneyTens(cMoney);
                    if (cMoney[1] == char.Parse("0"))
                        R = MoneyOnes(cMoney);
                    if (cMoney[0] == char.Parse("0") &
                        cMoney[1] == char.Parse("1"))
                }
            }
        }
    }
}

```



```

R = MoneyTens(cMoney) + "ة";
if (cMoney[0] == char.Parse("1") &
    cMoney[1] == char.Parse("1"))
    R = "عشر أحد";
if (cMoney[0] == char.Parse("2") &
    cMoney[1] == char.Parse("1"))
    R = "عشر اثنا";
if (cMoney[0] > char.Parse("2") &
    cMoney[1] == char.Parse("1"))
    R = MoneyOnes(cMoney) + " " + MoneyTens(cMoney);
if (cMoney[0] == char.Parse("0") &
    cMoney[1] != char.Parse("0")) // أصفار آحاد
    R = MoneyTens(cMoney);
}
if (i == 3)
{
    if (MoneyHundreds(cMoney) != "") {
        if (cMoney[0] != char.Parse("0") &
            cMoney[1] != char.Parse("0"))
            R = MoneyHundreds(cMoney) + " و" + R;
        if (cMoney[0] == char.Parse("0") &
            cMoney[1] != char.Parse("0"))
            R = MoneyHundreds(cMoney) +
                " و" + MoneyTens(cMoney);
        if (cMoney[0] != char.Parse("0") &
            cMoney[1] == char.Parse("0"))
            R = MoneyHundreds(cMoney) +
                " و" + MoneyOnes(cMoney);
        if (cMoney[0] == char.Parse("0") &
            cMoney[1] == char.Parse("0"))
            R = MoneyHundreds(cMoney);
    }
}
if (i == 4)
{
    if (MoneyThousands(cMoney) != "") {
        if (SConvertCurrency(OtherUnits(cMoney)) != "") {
            R = MoneyThousands(cMoney) +
                " و" + SConvertCurrency(OtherUnits(cMoney));
        }
    }
}
if (i == 5)
{

```



```

string TenThousands = SConvertCurrency(
double.Parse(cMoney[4].ToString() + cMoney[3].ToString())));
if (TenThousands != "") {
{
    if (TenThousands == "عشر") TenThousands += "آلفة";
    else TenThousands += "آلف";
    if (SConvertCurrency(OtherUnits2(cMoney)) != "") {
        R = TenThousands +
            " و" + SConvertCurrency(OtherUnits2(cMoney));
    } else R = TenThousands;
}
}
if (i == 6) {
{
    string HundredThousands = SConvertCurrency(
double.Parse(cMoney[5].ToString() + cMoney[4].ToString() +
cMoney[3].ToString()) + "آلف";
    if (HundredThousands != "") {
        if (cMoney[4] == '0' & cMoney[3] != '0') {
{
            if (cMoney[3] == '1') //إذا واحد يساوي الآلاف
                HundredThousands =
SConvertCurrency((double.Parse(cMoney[5].ToString() + "00")) + "ألف ";
            else if (cMoney[3] == '2')
                HundredThousands =
SConvertCurrency((double.Parse(cMoney[5].ToString() + "00")) + "ألفان ";
            else
                HundredThousands =
SConvertCurrency((double.Parse(cMoney[5].ToString() + "00")) + " و +
SConvertCurrency(double.Parse(cMoney[3].ToString())) + "آلف";
}
        }
        if (SConvertCurrency(OtherUnits3(cMoney)) != "") {
            R = HundredThousands +
                " و" + SConvertCurrency(OtherUnits3(cMoney));
        } else
            R = HundredThousands;
    }
}
}
if (i == 7) {
{
    if (MoneyMillions(cMoney) != "") {
{
        if (SConvertCurrency(OtherUnits(cMoney)) != "") {
            R = MoneyMillions(cMoney) +
                " و" + SConvertCurrency(OtherUnits(cMoney));
        }
}
}
}

```



```

        else
            R = MoneyMillions(cMoney);
        }
    }
    if (i == 8)
    {
        string TenMillions = SConvertCurrency(
double.Parse(cMoney[7].ToString() + cMoney[6].ToString()));
        if (TenMillions != "")
        {
            if (TenMillions == "عشر") TenMillions += "ملايين و";
            else TenMillions += "مليون";
            if (SConvertCurrency(OtherUnits2(cMoney)) != "")
                R = TenMillions +
                    " و" + SConvertCurrency(OtherUnits2(cMoney));
            else R = TenMillions;
        }
    }
    if (i == 9)
    {
        string HundredMillions = SConvertCurrency(
double.Parse(cMoney[8].ToString() + cMoney[7].ToString() +
cMoney[6].ToString()) + "مليون";
        if (HundredMillions != "")
        {
            if (cMoney[7] == '0' & cMoney[6] != '0')
            {
                if (cMoney[6] == '1')
                    HundredMillions =
SConvertCurrency((double.Parse(cMoney[8].ToString() + "00")) + "ومليون و";
                else if (cMoney[6] == '2')
                    HundredMillions =
SConvertCurrency((double.Parse(cMoney[8].ToString() + "00")) + "و"
"ومليونان";
            else
                HundredMillions =
SConvertCurrency((double.Parse(cMoney[8].ToString() + "00")) + " و" +
SConvertCurrency(double.Parse(cMoney[6].ToString()) + "ملايين و");
            }
        }
        if (SConvertCurrency(OtherUnits3(cMoney)) != "")
            R = HundredMillions +
                " و" + SConvertCurrency(OtherUnits3(cMoney));
        else
            R = HundredMillions;
    }
}

```



```

if (i == 10)
{
    if (MoneyBillions(cMoney) != "")
    {
        if (SConvertCurrency(OtherUnits(cMoney)) != "")
            R = MoneyBillions(cMoney) +
                " و " + SConvertCurrency(OtherUnits(cMoney));
        else
            R = MoneyBillions(cMoney);
    }
}

if (i == 11)
{
    string TenBillions = SConvertCurrency(
double.Parse(cMoney[10].ToString() + cMoney[9].ToString()));
    if (TenBillions != "")
    {
        if (TenBillions == "عشر") TenBillions += "مليارات و ";
        else TenBillions += "مليار و ";
        if (SConvertCurrency(OtherUnits2(cMoney)) != "")
            R = TenBillions +
                " و " + SConvertCurrency(OtherUnits2(cMoney));
        else R = TenBillions;
    }
}
if (i > 11)
{
    R = "";
}
return R;
}

static string MoneyOnes(params char[] c) //الأحاداد
{
    switch (int.Parse(c[0].ToString())) //أول خانة
    {

        case 1:
            return "واحد";
        case 2:
            return "اثنان";
        case 3:
            return "ثلاث";
        case 4:
            return "أربع";
    }
}

```



```

        case 5:
            return "خمس";
        case 6:
            return "ست";
        case 7:
            return "سبع";
        case 8:
            return "ثمان";
        case 9:
            return "تسعة";
        default:
            return "";
    }
}
الآحاد بطريقة أخرى // MoneyOnes(int Ones)
{
    switch (Ones)
    {

        case 1:
            return "واحد";
        case 2:
            return "اثنان";
        case 3:
            return "ثلاث";
        case 4:
            return "أربع";
        case 5:
            return "خمس";
        case 6:
            return "ست";
        case 7:
            return "سبع";
        case 8:
            return "ثمان";
        case 9:
            return "تسعة";
        default:
            return "";
    }
}
العشرات // MoneyTens(params char[] c)
{
    ثانية خانة // int.Parse(c[1].ToString()))
    {
        case 1:
            return "عشر";

```



```

        case 2:
            return "عشرون";
        case 3:
            return "ثلاثون";
        case 4:
            return "أربعون";
        case 5:
            return "خمسون";
        case 6:
            return "ستون";
        case 7:
            return "سبعون";
        case 8:
            return "ثمانون";
        case 9:
            return "تسعون";
    default:
        return "";
    }
}
المئات//MoneyHundreds(params char[] c)
{
    switch (int.Parse(c[2].ToString()))
    {
        case 1:
            return "مئة";
        case 2:
            return "مئتان";
        //جميع الحالات التالية لها نفس النتيجة
        case 3:
        case 4:
        case 5:
        case 6:
        case 7:
        case 8:
        case 9:
        {
            return MoneyOnes(int.Parse(c[2].ToString())) + " مئة";
        }
    default:
        return "";
    }
}
الآلاف//MoneyThousands(params char[] c)
{
    switch (int.Parse(c[3].ToString()))
    {

```



```

case 1:
    return "ألف";
case 2:
    return "ألفان";
case 3:
case 4:
case 5:
case 6:
case 7:
case 8:
case 9:
{
    return MoneyOnes(int.Parse(c[3].ToString())) + "آلاف";
}
default:
    return "";
}
}

static string MoneyMillions(params char[] c) // الملايين
{
    switch (int.Parse(c[6].ToString())) // سادس خانة
    {
        case 1:
            return "مليون";
        case 2:
            return "مليونان";
        case 3:
        case 4:
        case 5:
        case 6:
        case 7:
        case 8:
        case 9:
        {
            return MoneyOnes(int.Parse(c[6].ToString())) + "ملايين";
        }
    }
    default:
        return "";
}
}

static string MoneyBillions(params char[] c) // المليارات
{
    switch (int.Parse(c[9].ToString())) // سابع خانة
    {
        case 1:
            return "مليار";
        case 2:

```



```

        return "ملياران ";
    case 3:
    case 4:
    case 5:
    case 6:
    case 7:
    case 8:
    case 9:
    {
        return MoneyOnes(int.Parse(c[9].ToString())) + "مليارات ";
    }
    default:
        return "";
    }
}

الخانات التالية للألف والمليون // (c[])
{
    string t = "";
    for (int j = c.Length; j > 1; --j)
        t += c[j - 2].ToString(); //$$$$ >> $ is needed (or #####)
    return double.Parse(t);
}

الخانات التالية لعشرات الآلاف // (c[])
{
    string t = "";
    for (int j = c.Length; j > 2; --j) //###$$$ >> $ is needed
        t += c[j - 3].ToString();
    return double.Parse(t);
}

الخانات التالية لمئات الآلاف // (c[])
{
    string t = "";
    for (int j = c.Length; j > 3; --j) //####$$$ >> $ is needed
        t += c[j - 4].ToString();
    return double.Parse(t);
}

private void textBox1_TextChanged(object sender, EventArgs e)
{
    textBox2.Text = ConvertCurrency(textBox1.Text, "ليرة", "قرش");
}
}

```

وكبداية أعتذر منك جدًا جدًا عن رداءة الكود لأن عرض الصفحة لا يكفي ليظهر الكود كاملا، في بعض الأ קוד وزعتها على سطرين بحيث يتم



محاذاة السطر الثاني بعد 4 فراغات من الأول، وبعضها اكتفيت بمحاذاة الكود كاملاً إلى اليسار، وبقية الأكواد تركتها كما هي على الصفحة التالية تشرح لك ما جرى في الكود. كما أني جاهز لإعطيك هذا المشروع - وأي مشروع آخر تحتاجه - وذلك عن طريق أي وسيلة من وسائل الاتصال الموجودة في بداية هذا الكتاب.

في البداية ستقوم بالمناداة على التابع `ConvertCurrency` معطياً إياه العملة وأجزاءها، ليقوم بدوره بالمناداة على تابع آخر `SConvertCurrency` والذي يقوم بإدارة عملية تحويل كل رقم ضمن العدد إلى القيمة النصية التي يشغلها.

التابع `SConvertCurrency` يستقبل قيمة عشرية تمثل العدد المراد تحويله إلى كلمات، ليقوم بإسنادها مباشرةً لمتغير نصي، ليتم تحويلها إلى مجموعة من المحارف تمثل أرقام العدد. فالعدد 5220764 - والذي يمثل هاتفي الأرضي - يصبح "5220764" ثم "'4','6','0','7','2','5','2'" والذي يبدأ بالحرف 5 وينتهي بـ 2. ولكننا سنتعامل مع العدد من آحادها إلى عشراتها ثم مئاته وهكذا إلى أن ينتهي، لذلك سنعكس العدد.

وبعدها سنقوم بدراسة كل رقم برقمه، وذلك عبر حلقة تكرار تبدأ من القيمة 1 (أول خانة) وتنتهي بقيمة تمثل عدد أرقام العدد، وفي حالتنا 7 أرقام، أي أن الحلقة ستتكرر 7 مرات، من 1 وحتى 7.

إذا كان عدد الحلقة i يساوي 1، أي أنها ندرس أول خانة، فإننا سنستدعي التابع `MoneyOnes` والذي يقوم بتحويل الآحاد إلى أرقام، وبدوره التابع `MoneyOnes` يقوم باستقبال عدد غير محدد من الوسطاء كمصفوفة حرافية، ليقوم باختبار قيمة العنصر الأول ذي الترتيب 0، فإذا كانت 1 يعيد القيمة "واحد"، وإذا كانت 2 يعيد القيمة "اثنان"، وهكذا حتى 9. وفي مثالنا ومن أجل قيمة الآحاد 4 سيعيد التابع القيمة "أربع". وبالعودة للتابع الذي استدعي `MoneyOnes` فإن R ستصبح قيمتها "أربع".

ينتقل الكود سطراً سطراً ولا ينفذ أي سطر بعد بنية الشرط الأولى لأن جميع الشروط التالية غير محققة. ليزداد عدد الحلقة بمقدار 1 ويعيد تنفيذ محتواها من جديد.

الآن عدد الحلقة له القيمة 2، لذلك فالشرط الذي ينص على أن $i == 2$ سيتم تنفيذه، وفيه سنلاحظ الحالات التالية (وهي حالات وضعتها بنفسي) ومن المحتمل أن توجد حالات أخرى لم تخطر على بالي، والموضوع مماثل لبني الشرط الأخرى):



- أن يكون الآحاد لا يساوي الصفر والعشرات أكبر من الواحد كالأعداد الأكبر من عشرين بحيث يكون آحادها لا يساوي الصفر مثل 27 مثلا، عندها يكون العدد سبع وعشرون، أو في مثالنا السابق أول رقمين 64 فيكون العدد بالكلمات هو أربع وستون. لاحظ، نكتب الآحاد ثم "و" ثم العشرات.. لذلك سنستدعي تابعين الأول للآحاد MoneyTens والثاني للعشرات MoneyOnes وهو مماثل للتتابع الأول إلا أنه يدرس العنصر الثاني من مصفوفة الأرقام ويعيد قيم مثل "عشر"، "عشرون"، وهكذا حتى "تسعون".
- أن يكون العشرات معدوم، مثل العدد 101 فيه خانة العشرات معدومة عندها R تؤول إلى حالة الآحاد.
- أن يكون الآحاد معدوم والعشرات تساوي 1 وهو حالة العدد 10 فقط، وعندما R تساوي "عشرة".
- أن يكون كل من الآحاد والعشرات مساوين للقيمة 1 وهو حالة العدد 11 فقط، عندها R تساوي "أحد عشر".
- أن يكون الآحاد له القيمة 2 والعشرات له القيمة 1 وهو حالة العدد 12 فقط، عندها R تساوي "اثنا عشر".
- أن يكون الآحاد أكبر من 2 والعشرات له القيمة 1 مثل الأعداد من 13 وحتى 19 عندها يكون العدد بالكلمات مصافا، مثلا 13 يكتب ثلاث عشر، آحاد ثم عشرات.
- أن يكون الآحاد معدوم والعشرات لا يساوي 1، مثل 20 أو 30 أو 40، فتكون R هي ناتج تنفيذ التابع MoneyTens فقط.

عندما يصبح عدد الحلقة 3، وفي حالة لم تكن خانة المئات معدومة، تكون لدينا الحالات التالية:

- ألا يكون الآحاد أو العشرات معدوما مثل 764 في مثالنا القديم، عندها يكون الناتج سبع مئة وأربع وستون، أي قيمة نصية مقابلة لخانة المئات، ثم "و"، ثم قيمة R القديمة. القيمة المقابلة لخانة المئات نحصل عليها من التابع MoneyHundreds والذي يعيد القيمة "مئة" للعدد 100، "مئتان" للعدد 200، أما من أجل أعداد أكبر من ذلك مثل 300 ومما فوق فإنه يستدعي تابع الآحاد ويضيف إلى نهاية القيمة المعادة القيمة "مئة".



- أن يكون الآحاد معدوم والعشرات غير معدوم مثل العدد 720 وعندما R ستكون سبع مئة وعشرون، لاحظ قيمة 7 بالآحاد ثم "مئة" ثم "و" ثم قيمة 2 بالعشرات.
- أن يكون الآحاد غير معدوم والعشرات معدوم مثل 702 وعندما R ستكون سبع مئة واثنان، لاحظ قيمة 7 بالآحاد ثم "مئة" ثم "و" ثم قيمة 2 بالآحاد.
- أن يكون الآحاد والعشرات معدومان وعندما R ستكون ناتج إرجاع تابع المئات فقط.

عندما يصبح عدد الحلقة 4، أي عند دراسة الآلاف فإنه ومن أجل قيمة للآلاف غير معدومة تكون لدينا الحالات التالية:

- إذا كان ما أمام خانة الآلاف (خانة الآحاد والعشرات والمئات) غير معدوم – ويمكن ذلك عن طريق استدعاء التابع SConvertCurrency ليعيد فقط القيمة المقابلة للأرقام المكونة من الآحاد والعشرات والمئات وهذه القيمة تحصل عليها من التابع OtherUnits والذي يقوم بتكون عبارة نصية مكونة من الخانات الثلاث فقط – فعندما يُستدعي تابع الآلاف ويضاف للقيمة المعادة من تابع الآلاف القيمة "و" ثم ناتج استدعاء SConvertCurrency من أجل ناتج OtherUnits أعلم أن الموضوع مربك نوعاً ما لذلك تأمل الأمثلة التالية:

العدد 1250 يكتب ألف ومئان وخمسون، لاحظ ألف ثم "و" ثم ناتج تحويل القيمة 250 إلى كلمات وذلك باتباع الخطوات عندما عدنا حلقة يأخذ القيم 1 و 2 و 3.

العدد 7020 يكتب سبع آلاف وعشرون، لاحظ سبع آلاف – القيمة المعادة من تابع الآلاف MoneyThousands – ثم "و" ثم ناتج تحويل 020 إلى كلمات.

- إذا كان ما أمام خانة الآلاف معدوم أي 000 تكون R لها القيمة المعادة من تابع الآلاف فقط.

وبالنسبة لـ OtherUnits فإنه يعمل بالشكل التالي:

يستقبل مصفوفة حرفية – غير معلومة عدد العناصر – ويقوم بإرجاع جميع المحارف ما عدا الحرف الأخير. فمثلاً 2501 يعاد منه 501، 1111 يعاد منه 111 وهكذا. هذه القيمة المكونة من ثلاث خانات تقوم بدراستها من جديد لنكتبه على شكل كلمات. فالعدد 2501



يُكتب ألفان - ناتج استدعاء تابع الألوف - ثم " و " ثم يستدعي تابع تحويل العدد لكلمات من أجل العدد 501 ليكون الناتج خمس مئة وواحد. ليصبح الناتج النهاية ألفان وخمس مئة واحد.

استغرقت في برمجة هذا المشروع ثلاثة أيام - بمعدل أربع ساعات يومياً - لأحصل على هذا التطبيق، وقد ابتدأت بأصل 0.00% وأن أصل ل نهايته، لأنني عزيزي القارئ أنه لا صعب مع المحاولة والإصرار والرغبة بوجودة المعرفة والتوكيل على الله.

على أيام VB6 كنت قد طورت كودا حملته من إحدى الصفحات - أو من أحد برامج موسوعات الأكواد الجاهزة لا أذكر - يقوم بتحويل الأعداد إلى كلمات.

وكانت نيتها - في بداية الكتاب - أن أنشئ فصلاً بالاعتماد على كتاب لخالد السعداني يشرح كيفية الانتقال من لغة برمجة إلى أخرى، وإحدى لغات البرمجة التي كنت بصدده الحديث عنها هي الفيجوال بيسك، وأحد المشاريع التي كنت سأتناولها هو هذا المشروع.. لكن الوقت لم يسعفي كما أتمنى تعبت للغاية فقررت اختصار بعض الفصول وإلغاء أخرى. فكانت خطتي السابقة وضع مشاريع قواعد بيانات شاملة ومناقشتها، والحديث أكثر عن المشاريع الجاهزة - الخاصة بهذا الفصل - وطرح المزيد منها، كما كانت لدي نية بإدخال فصل إلى الكتاب يتحدث عن كيفية الاستفادة من موقع الانترنت المختلفة لإنشاء تطبيقاتك، وذلك عبر إنشاء برامج كاملة من الصفر وحتى التعامل مع معالج التثبيت Setup Wizard، لكن كما أسلفت فالوقت قد قطعني لأنني لم الحق أن أقطعه على اعتبار أنه كالسيف.

ما علينا¹، قررت في النهاية وضع المشروع في نهاية هذا الفصل كهدية للقارئ - لأنه بنظري من أكثر المشاريع قيمةً وتطبيقاً واقعياً - وكدعائية وأهداف شخصية أخرى. لكنني صدمت مرة أخرى عندما حاولت الانتقال من أسلوب الأكواد الإجرائي الذي تعامل معه VB إلى أسلوب أكواد بيئة دوت نت، ومع بعض التجارب الفاشلة حصلت على هذا التطبيق الناجح!! ولمن يرغب بالاطلاع أو الاستفادة فهذا هو الكود بلغة Visual Basic 6:

```
Public Function ConvertCurrency(Money, FullCurrency, _ PartCurrency As _
String)
On Error GoTo 1
Ma = " " & FullCurrency
```

¹ ماعلينا: على أي حال.



```

MI = " " & PartCurrency
X = Money
N = Int(X)
B = Val(Format(X, "000000000000.00"), 2))
r = SConvertCurrency(N)
If r <> "" And B > 0 Then result = r & Ma & " و " & B & MI
If r <> "" And B = 0 Then result = r & Ma
If r = "" And B <> 0 Then result = B & MI
ConvertCurrency = result
Exit Function
1
ConvertCurrency = "يجب إدخال أرقام حصرًا"
End Function

Private Function SConvertCurrency(X)
N = Int(X)
C = Format(N, "000000000000")
C1 = Val(Mid(C, 12, 1))
Select Case C1
Case Is = 1: Letter1 = "واحد"
Case Is = 2: Letter1 = "اثنان"
Case Is = 3: Letter1 = "ثلاثة"
Case Is = 4: Letter1 = "أربعة"
Case Is = 5: Letter1 = "خمسة"
Case Is = 6: Letter1 = "ستة"
Case Is = 7: Letter1 = "سبعة"
Case Is = 8: Letter1 = "ثمانية"
Case Is = 9: Letter1 = "تسعة"
End Select
C2 = Val(Mid(C, 11, 1))
Select Case C2
Case Is = 1: Letter2 = "عشر"
Case Is = 2: Letter2 = "عشرون"
Case Is = 3: Letter2 = "ثلاثون"
Case Is = 4: Letter2 = "أربعون"
Case Is = 5: Letter2 = "خمسون"
Case Is = 6: Letter2 = "ستون"
Case Is = 7: Letter2 = "سبعون"
Case Is = 8: Letter2 = "ثمانون"
Case Is = 9: Letter2 = "تسعون"
End Select
If Letter1 <> "" And C2 > 1 Then Letter2 = Letter1 + " و " + _Letter2
If Letter2 = "" Then Letter2 = Letter1
If C1 = 0 And C2 = 1 Then Letter2 = Letter2 + "ة"
If C1 = 1 And C2 = 1 Then Letter2 = "إحدى عشر"
If C1 = 2 And C2 = 1 Then Letter2 = "اثنتي عشر"
If C1 > 2 And C2 = 1 Then Letter2 = Letter1 + " " + Letter2

```



```

C3 = Val(Mid(C, 10, 1))
Select Case C3
Case Is = 1: Letter3 = "مائة"
Case Is = 2: Letter3 = "مئتان"

Case Is > 2: Letter3 = Left(SConvertCurrency(C3), _
Len(SConvertCurrency(C3)) - 1) + "لهم"
End Select

If Letter3 <> "" And Letter2 <> "" Then Letter3 = Letter3 + " و" _ + Letter2
If Letter3 = "" Then Letter3 = Letter2
C4 = Val(Mid(C, 7, 3))
Select Case C4
Case Is = 1: Letter4 = "ألف"
Case Is = 2: Letter4 = "ألفان"
Case 3 To 10: Letter4 = SConvertCurrency(C4) + "آلاف"
Case Is > 10: Letter4 = SConvertCurrency(C4) + "ألف"
End Select

If Letter4 <> "" And Letter3 <> "" Then Letter4 = Letter4 + " و" _ + Letter3
If Letter4 = "" Then Letter4 = Letter3
C5 = Val(Mid(C, 4, 3))
Select Case C5
Case Is = 1: Letter5 = "مليون"
Case Is = 2: Letter5 = "مليونان"
Case 3 To 10: Letter5 = SConvertCurrency(C5) + "ملايين"
Case Is > 10: Letter5 = SConvertCurrency(C5) + "مليون"
End Select

If Letter5 <> "" And Letter4 <> "" Then Letter5 = Letter5 + " و" _ + Letter4
If Letter5 = "" Then Letter5 = Letter4
C6 = Val(Mid(C, 1, 3))
Select Case C6
Case Is = 1: Letter6 = "مليار"
Case Is = 2: Letter6 = "ملياران"
Case Is > 2: Letter6 = SConvertCurrency(C6) + "مليار"
End Select

If Letter6 <> "" And Letter5 <> "" Then Letter6 = Letter6 + " و" _ + Letter5
If Letter6 = "" Then Letter6 = Letter5
SConvertCurrency = Letter6
End Function

```

ومع الأسف وبعد ما انتهيت من التطبيق اكتشفت أن هناك من صمم واحداً وهو المهندس محمد حمدي غانم، وهو من نجوم البرمجة العربية مثل تركي العسيري وأحمد جمال خليفة.

والرائع في التطبيق الذي أنشأه المهندس محمد حمدي غانم هو أنه يتعامل مع التأثير والتذكير ويعطيك إمكانية لإحصاء الأموال أو غير الأموال



مثل إحصاء الناس أو الأدوات أو القطع مثلًا، كما أنه يشتمل على مجال أوسع من الأعداد..

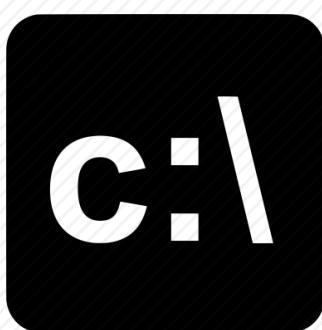
مشكلتي أنني لا أبحث إلا باللغة الإنكليزية، لذلك لم أصادف الكود (صادفت بعض المشاريع التي تحويل الأعداد إلى كلمات إنكليزية).





الفصل العاشر – الدوس و C#

ذكرنا في بداية الكتاب أن أنظمة الحواسيب كانت يومًا ما هي نظام الدوس DOS OS، ولاحقًا تم اعتماد نظام ويندوز Windows مع إبقاء الدوس كأداة من أدواته.. وحتى اليوم، الكثير من المبرمجين يعتمدون في برامجهم على نظام الدوس لأداء وظائف معينة، فليس كل ماتراه أمامك في البرامج هو كل ما تم برمجته، هناك عمليات كثيرة لا تراها كمستخدم!



يتمثل الدوس في ويندوز بموجة الأوامر Command Prompt، كما يعرف بـ cmd، ويتميز بأن ما يصدر عنه أمرٌ مطاع، فهو مدلل للغایة 2، ويندوز لا يجرؤ على رفض أمر من أوامره بحجة أنه لا يملك إمكانية الوصول مثلاً، أو أن الملف محمي أو غيرها من الحجج، فبعض الأوامر في cmd بإمكانك إزالة أقوى الحمايات والقيام بالعديد من الأمور بسهولة ويسر. مشكلته الوحيدة هي

أن عليك كتابة الأمر بدقة شديدة، لذلك ومع امتداد هذا الفصل توقع الحصول على كثير من الأخطاء مثل أن الملف غير موجود أو أنه مستخدم أو أن الأمر غير معروف وغيرها من الأخطاء، خلاف ذلك لا مشكلة جمیع أمورك محلولة.

في هذا الفصل سنتحدث عن أكثر الأوامر شيوعا في نظام الدوس والتي سنستخدم بعضها في برامجنا.



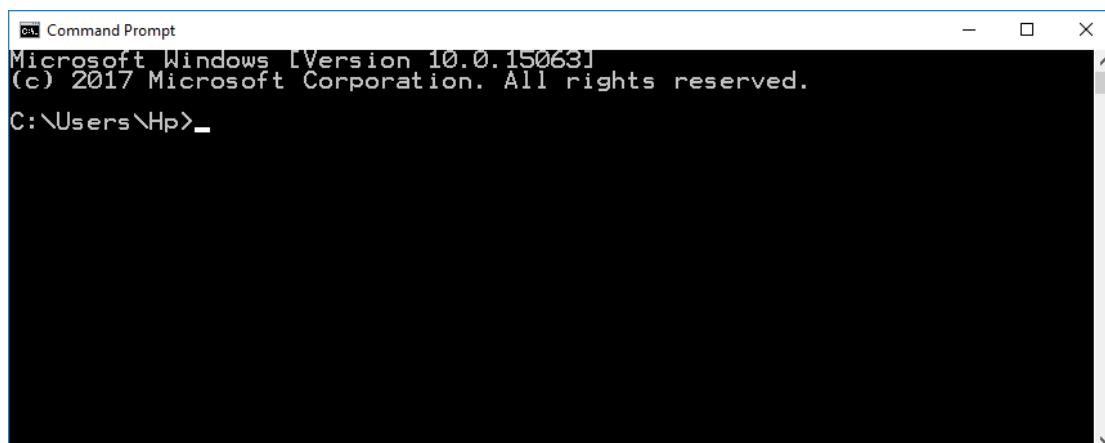
ما هو موجه الأوامر cmd؟

"يعد 'موجه الأوامر' إحدى ميزات Windows التي توفر نقطة إدخال لكتابة أوامر MS-DOS (نظام تشغيل أقراص Microsoft) وأوامر الكمبيوتر الأخرى. أهم ما ينبغي معرفته في هذا الشأن أنه عن طريق كتابة الأوامر، يمكنك تنفيذ المهام على جهاز الكمبيوتر بدون استخدام واجهة Windows الرسومية. عادة ما يستخدم 'موجه الأوامر' من قبل المستخدمين المتقدمين فقط."¹.

طيب لماذا تناولناه في هذا الكتاب؟ لأن هناك الكثير من الأمور التي بإمكاننا القيام بها في C# بمساعدة موجه الأوامر، توجد أكثر من فكرة بإمكاننا الاستعانة بموجه الأوامر لأداءها في C#, والفقرات التالية ستوضح ذلك. وعلى اعتبار أنه من المحتمل أنك - عزيزي القارئ - لا تعلم كثيراً عن موجه الأوامر، فكان من الجيد شرحه لك بشيء من الإيجاز لأكثر من غاية، الأولى هي تحضيرك وتدريبك على كيفية عمل موجه الأوامر لتسليط فهم الأفكار التي قلنا إنها تعتمد عليه، الثانية هي إطلاعك على بيئة الدوس والتي قد تقييدك في تطبيقات خاصة بك، الثالثة هي إكسابك مرجعاً صغيراً مختصراً عن موجه الأوامر إن كان أحد اهتماماتك.².

والكتاب هذا يحاول قدر الإمكان احتواء أكبر عدد من المواضيع الممكنة المتعلقة بـ C#، فقد تحدثنا عن المفاهيم الأساسية والمتقدمة، وناقشتنا بعض المشاريع التطبيقية، وسنناقش الآن كيفية التعامل مع موجه الأوامر من خلال C#, ولاحقاً كيفية التعامل مع الرجستري من خلال C#, كما وسنتحدث عن قواعد البيانات أيضاً إن شاء الله.

لتتغيل موجه الأوامر افتح قائمة ابدأ ثم اكتب cmd:



¹ مقتبس من التعليمات والدعم في ويندوز 7.

² إذا لم تكن ترغب بتعلم نظام الدوس فانتقل مباشرةً لفقرة حماية الملفات والبيانات هناك ستجد C#.



أوامر دوس

كما هو الحال في أي بيئة برمجية - أو بكلام أدق بيئه تتعامل مع أوامر برمجية - فإنه عليك كمستثمر لهذه البيئة سواءً أكانت برماجاً أو لغة برمجة أو غيرها من البرمجيات أن تعرف الكلمات الخاصة بهذه البيئة وأن تفهم طريقة تطبيق هذه الكلمات أهم من معرفتها أساساً.. هذه الكلمات هي نفسها الكلمات الممحوزة في لغات البرمجة.

موجه الأوامر يتعامل مع أوامر محددة، بمتغيرات محددة، وأي خلل في الأمر أو المتغيرات يعرضك لخطأ أنت بمعنى عنه. فلا تتوقع مثلاً أن تكتب له مثلاً (Hello bro) فيجيب تحينك! بالعكس تماماً ستظهر لك عبارة ستمر معك طيلة فترة حياتك وستكره عمرك منها، هذه العبارة مفادها: "ياحبيبي، 'Hello' لا يعتبر من الأوامر الداخلية أو الخارجية، وليس ببرنامجاً قابلاً للفتح أو ملفاً دفعياً"، لا تأخذ مني إلا الكلام، افتح موجه الأوامر وجرب الكود السابق:

```
C:\Users\Hp>Hello bro :)
'Hello' is not recognized as an internal or external command,
operable program or batch file.
```

الجملة السابقة تخبرك أن هناك مكانة خاصة للملفات الدفعية في قلب موجه الأوامر 😊. ما علينا¹، مبدئياً ضع في ذهنك شيئين: الأول أنه يجب عليك معرفة الأوامر التي يفهمها موجه الأوامر، الثاني أن نتيجة هذا الأمر أو استجابة موجه الأوامر لهذا الأمر على كيف (وفق رغبة) موجه الأوامر نفسه، لذلك ستعاني نوعاً ما 😊. لكن لحظة، ماهي الملفات الدفعية؟ لا تستعجل على رزقك ستتعرف عليها في فقرة لاحقة.

قبل التعرف على أوامر cmd اقرأ الملاحظة جيداً:

ملاحظة

- عليك الانتقال لفهرس المجلدات أو الملفات التي ستتعامل معها وذلك باستخدام أمر معين سنراه لاحقاً، وإلا فعليك كتابة مسار هذه الملفات أو المجلدات بشكل كامل.
- فهرس يعني مجلد أو مسار ☺ (فاجأتك صحيح 😊?).
- في حال وجود فراغات في مسارات أو أسماء المجلدات أو الملفات التي ستتعامل معها عليك إحياطتها بإشارة " .
- المحرف \ يستخدم مع المسارات أما / فيستخدم مع خيارات ومتغيرات الأوامر.

¹ ماعلينا: على أي حال.



سأشرح لك بعض الأمثلة لكيفية استخدام موجه الأوامر:

افتح موجه الأوامر لتلاحظ وجود ثلاثة أسطر بها كلمات، أول سطرين يصفان لك نسخة موجه الأوامر المثبتة على جهازك. السطر الثالث فارغ، السطر الرابع مكتوب فيه مسار مستخدمك الشخصي في كمبيوترك ثم إشارة >، وأمامها مؤشر يظهر ويختفي كل أقل من ثانية.. سنقوم الآن ببعض العمليات الأساسية والتي لا غنى عنها عند التعامل مع cmd.

- لتغيير عنوان موجه الأوامر اكتب title ثم العنوان المطلوب.
- لطباعة عبارة نصية على شاشة موجه الأوامر اكتب echo ثم العبارة المراد طباعتها.
- لتغيير لون شاشة ولون خط موجه الأوامر استخدم الأمر color، مثلا color 02.
- للتنقل بين الأاقراص نستخدم التعليمية cd. مثلا للانتقال لمجلد ما اكتب cd ثم مساره، وفي حال كان المجلد داخل الفهرس الذي أنت فيه يكفي كتابة cd ثم اسمه. مثلا للانتقال لمجلد Games موجود في القرص d اكتب cd d:\games. أما للانتقال لقرص ما اكتب اسم القرص ثم نقطتين، مثلا d: لتنقل للقرص d. ولعرض اسم الفهرس الحالي اكتب فقط cd.

ملاحظة

- موجه الأوامر ليس حساسا لحالة الأحرف، ف games نفسها .Cd وكذلك الحال بالنسبة للأوامر cd نفسها CD.

- لعرض محتويات الفهرس الحالي اكتب dir، مثلا لعرض محتويات القرص e انتقل له باستخدام الأمر e: ثم اكتب dir.
- لمسح محتويات الشاشة اكتب cls.

والآن سأتركك مع الأوامر الأكثر شيوعا في cmd:

الأمر	وصفه
Assoc	عرض أو إضافة اسم مشغل ملف معين في الرجسستري. لمعرفة جميع المشغلات في الكمبيوتر: assoc لمعرفة مشغل الملفات النصية مثلا: Assoc.txt والنتيجة ستكون txt = txtfile ، ولمعرفة مسار هذه



<p>المشغل استخدم الأمر <code>ftype</code>. إضافة مشغل جديد إلى مشغلات الملفات في الكمبيوتر، مثلاً في فصل الرجساري سنتعامل مع ملفات نحن سننشئها وسنسمّي لاحقّتها <code>:myra</code> <code>Assoc.myra = myrafile</code></p>	
<p>عرض وتعديل خصائص الملفات والمجلدات، والبحث عن الملفات. ويأخذ المتغيرات التالية: <code>R</code> للقراءة فقط. <code>H</code> مخفي. <code>S</code> ملف نظام. <code>+ إضافة المتغير (السمة)</code> ، <code>- إزالته</code>. وله الخيارات: <code>S</code> لتطبيق الأمر على الملفات الفرعية. <code>D</code> لتطبيق الأمر على المجلدات أيضًا. لجعل الملف للقراءة فقط:</p>	
<p><code>Attrib d:\File.txt +r</code> لإخفاء ملف:</p>	
<p><code>Attrib d:\File.txt +h</code> لجعل ملف ما من ملفات النظام:</p>	
<p><code>Attrib d:\File.txt +s</code> لإزالة إحدى الصفات السابقة اكتب <code>- بدل +</code>. لتطبيق السمات السابقة على كافة محتويات فهرس ما:</p>	Attrib
<p><code>Attrib d:\Folder +r /s /d</code> سيتم تطبيق السمات على الملفات الفرعية مباشرة في المجلد <code>Folder</code>.</p>	
<p>يمكن تنفيذ هذا الأمر مع مجلد بعينه لكن لا يمكن تنفيذه مع عدة مجلدات كما هو الحال مع الملفات، كما أن خاصية القراءة فقط لا يمكن تطبيقها على المجلدات أصلًا، افتح خصائص أحد المجلدات ولاحظ ماذا كتب مابين قوسين أمام خاصية القراءة فقط، فالملجد كائن لا يملك خاصية للقراءة فقط!</p>	
<p>إعادة تسمية المجلد أو الملف لا تعتبر من عمليات الكتابة.. جرب تغيير اسم ملف للقراءة فقط. لحماية ملف أو مجلد بإمكانك إخفاءه وجعله من ملفات النظام في ذات الوقت بحيث لا يظهر حتى لو تم تفعيل خيار عرض الملفات والمجلدات المخفية:</p>	
<p><code>Attrib d:\Folder +h +s</code> وفي هذه الحالة فإن خيارة الإخفاء سيتم إلغاؤه. لاستعادة الملف أو المجلد اكتب <code>- بدل +</code>. كما يمكن</p>	



	إظهار المجلدات والملفات المخفية من خيارات المجلد في مستعرض ويندوز وذلك بإلغاء تفعيل خيار "إخفاء ملفات نظام التشغيل المحمية (مستحسن)" والموافقة على رسالة التحذير.
	تفعيل أو إلغاء تفعيل إمكانية مقاطعة موجه الأوامر عند عمله، في الأوامر التي تأخذ وقتاً طويلاً أثناء عملها يمكن مقاطعتها عبر الضغط على Ctr + C، جرب اكتب الأمر tree والذي سيعرض لك محتويات الفهرس الحالي على شكل شجرة متفرعة، لاحظ أنه سيأخذ بعض الوقت، وفي هذه الحالة قد تحتاج لإيقافه. الحالة الافتراضية للأمر break on أي أنه يمكنك مقاطعة الأوامر، ولإلغاء تفعيلها اكتب break off.
	يقوم بتشغيل ملف دفعي موجود على القرص الصلب. سنرى الملفات الدفعية لاحقاً.
	للتنقل بين الفهارس في موجه الأوامر. وهو نفسه .chdir للانتقال للقرص الأب: Cd\ للانتقال للفهرس الأب (خطوة للخلف): cd.. للانتقال للفهرس الجد (المجلد ما قبل السابق): Cd ..\.. لعرض الفهرس الحالي: Cd للانتقال لفهرس ما: Cd e:\C# في حال كان مسار الفهرس يحوي فراغات يجب تضمينه بين إشارتي " " كما في المتغيرات النصية. في الملفات الدفعية للانتقال للفهرس الحاوي على الملف الدفعي: cd \d "%~dp0"
	يستخدم لاستقبال خيارات عدديّة في الملفات الدفعية. ويأخذ الخيارات التالية: C إضافة خيارات للمستخدم M إضافة رسالة تظهر على شكل نص. N عدم إظهار الخيارات الافتراضية بعد نص الخيار M. D الخيار الافتراضي. T الوقت الافتراضي والذي إذا مضى سيتم أخذ الخيار الافتراضي المحدد بالخيار D. ويأخذ القيم من



<p>0 وحتى 9999 . الخيارات حساسة لحالة الأحرف. الخيارات T و Dg يأتيان معاً. لاحظ الأمثلة:</p> <p>CHOICE /C YN /M "Press Y for Yes, N for No." CHOICE /T 10 /C ync /CS /D y CHOICE /C ab /M "Select a for option 1 and b for option 2."</p> <p>بعد إدخال أحد الخيارات فإن هناك متغير افتراضي في موجه الأوامر يسمى errorlevel يأخذ قيمًا عدديًا.. هذا المتغير سينستخدمه لتحديد ما يجب على موجه الأوامر تنفيذه وذلك من خلال الأمر if . إذا لم يتم إدخال أحد الخيارات الموضعة من خلال الخيار c فإن موجه الأوامر لن يسمح لك بالتقدم للأوامر التالية (مثلاً لديك الخيارات 123 وقمت بالضغط على 4).</p>																	
<p>لمسح محتويات الشاشة.</p>	Cls																
<p>نسخ ناتج تنفيذ أمر ما إلى الحافظة. مثلاً:</p>	Clip																
<p>Dir clip</p> <p>تغيير لون شاشة او خط موجه الأوامر، ويأخذ أحد القيم:</p> <table style="margin-left: 100px; margin-top: 10px;"> <tr><td>0 = Black</td><td>8 = Gray</td></tr> <tr><td>1 = Blue</td><td>9 = Light Blue</td></tr> <tr><td>2 = Green</td><td>A = Light Green</td></tr> <tr><td>3 = Aqua</td><td>B = Light Aqua</td></tr> <tr><td>4 = Red</td><td>C = Light Red</td></tr> <tr><td>5 = Purple</td><td>D = Light Purple</td></tr> <tr><td>6 = Yellow</td><td>E = Light Yellow</td></tr> <tr><td>7 = White</td><td>F = Bright White</td></tr> </table> <p>حيث يُكتب حرفان بعد الأمر color الأول للون الخط والثاني للون الخلفية، ولو تم كتابة حرف واحد فسيتم تغيير لون الخط فقط.</p>	0 = Black	8 = Gray	1 = Blue	9 = Light Blue	2 = Green	A = Light Green	3 = Aqua	B = Light Aqua	4 = Red	C = Light Red	5 = Purple	D = Light Purple	6 = Yellow	E = Light Yellow	7 = White	F = Bright White	Color
0 = Black	8 = Gray																
1 = Blue	9 = Light Blue																
2 = Green	A = Light Green																
3 = Aqua	B = Light Aqua																
4 = Red	C = Light Red																
5 = Purple	D = Light Purple																
6 = Yellow	E = Light Yellow																
7 = White	F = Bright White																
<p>يستخدم لنسخ الملفات ☺، إلا أنه لا يت suction المجلدات.. لنسخ المجلدات استخدم xcopy . لنسخ ملف موجود في القرص d إلى القرص c: D: Copy File1.txt c:\ أو يمكن ذلك دون الانتقال للقرص d: Copy d:\File1.txt c:\ لنسخ جميع ملفات مجلد ما إلى القرص c: Copy d:\Folder*. * c:\ كما يمكن نسخ ملفات من نوع واحد: Copy d:\Folder*.txt c:\</p>	Copy																



<p>ومن الخيارات الرئيسية فيه الخيار y لعدم إظهار رسالة تأكيد قبل النسخ: Copy /y d:\File1.txt c:\ لحفظ كل ما يكتب في موجه الأوامر في ملف اكتب: Copy con d:\file.txt الأوامر . . لإنهاء الإدخال اضغط .Ctrl + z.</p>	
<p>يستخدم للعرض التاريخ مع إمكانية التعديل. في حال أردت فقط العرض اضغط Enter دون إدخال قيمة جديدة للتاريخ.</p>	Date
<p>يستخدم لحذف الملفات، إلا أنه لا يحذف المجلدات.. لحذف المجلدات استخدم rd. الخيارات الرئيسية فيه: P التنبيه قبل حذف أي ملف. F إجبار حذف الملفات للقراءة فقط. S حذف الملفات من المجلدات الفرعية. Q حذف سريع، دون تنبيه. A اختيار نوع معين من الملفات وله المتغيرات: R - للقراءة فقط. H - المخفية. A - أرشيف. S - ملفات النظام. في حال تم استخدام اللاحقة "-" أمام أي من السمات السابقة فإنها سيتم حذف الملفات التي لا تحتوي هذه السمة. لحذف ملف معين:</p>	Del
<p>Del D:\Folder\File1.txt لحذف ملف تم الانتقال لفهرسه: D: Cd d:\Folder Del File1.txt لحذف جميع الملفات من جميع الأنواع داخل مجلد معين، دون تنبيه مع حذف الملفات في المجلدات الفرعية: Del d:\Folder*.* /s /q حذف جميع الملفات المخفية من جميع الأنواع داخل مجلد معين، مع إظهار تنبيه عند حذف كل ملف: Del d:\Folder*.* /p /s /a : h</p>	



<p>عرض محتويات الفهرس الحالي، ويأخذ الخيارات التالية:</p> <p>A اختيار نوع معين من الملفات وله المتغيرات:</p> <ul style="list-style-type: none"> - R للقراءة فقط. - H المخفية. - A أرشيف. - S ملفات النظام. - في حال تم استخدام اللاحقة "-". أمام أي من السمات السابقة فإنها سيتم حذف الملفات التي لا تحتوي هذه السمة. <p>L عرض الملفات والمجلدات بأحرف صغيرة.</p> <p>O فرز الملفات بحسب المتغيرات التالية:</p> <ul style="list-style-type: none"> - N بحسب الاسم. - E بحسب الامتداد (بشكل أبجدي). - G تجميع الملفات أولاً. - S بحسب الحجم (الأصغر أولاً). - D بحسب التاريخ (الأقدم أولاً). - في حال تم استخدام اللاحقة "-". أمام أي من المتغيرات السابقة فإنه سيتم عكسها (يعني مثلاً S - تعني بحسب الحجم الأكبر). <p>Q عرض أسماء مالكي الملفات.</p> <p>S عرض الملفات والمجلدات الفرعية أيضاً.</p> <p>W عرض الملفات والمجلدات بشكل عرضي.</p> <p>لعرض محتويات الفهرس الحالي:</p> <p>Dir</p> <p>لعرض محتويات فهرس ما مرتبة بحسب الاسم وبشكل عرضي:</p> <p>Dir d:\o:n/w</p>	<p>Dir</p>
<p>عرض العبارات النصية في موجه الأوامر وإيقاف أو تفعيل عرض المسارات، وهو مكافئ لـ <code>Console.WriteLine</code> بالنسبة لوظيفته الأولى..</p> <p>المسارات هي تلك العبارات التي تراها في كل سطر من أسطر موجه الأوامر والتي تخبرك أين أنت في القرص الصلب، حتى تعرف أين أنت.. هذه العبارات تسمى المحت، لا أدرى من سماها هكذا ولماذا، تفاصيل أكثر عنها في الأمر <code>prompt</code>.</p> <p>عند التعامل مع الملفات الدفعية أنت لست بحاجة لعرض هذه المسارات لأنك أصلاً عند إنشاءك للملف الدفعي قمت بدراسة وتحطيط المسارات التي ستنتقل إليها، لذلك لا داعي لعرضها.</p>	<p>Echo</p>



<p>لعرض المسارات استخدم echo on ولإيقافها echo off، ومع هذا فإن المسار الموجود في سطر أمر إيقاف عرض المسارات لا زال موجوداً، وإزالته استخدام البادئة @، أي اكتب @Echo Off.</p> <p>لعرض عبارة نصية:</p>	
<p>Echo Choose [a, b, c or d]:</p> <p>لعرض سطر فارغ:</p> <p>Echo.</p> <p>إظهار محتوى متغير:</p> <p>Echo %proframfiles%</p> <p>كتابة محتوى متغير في جملة:</p> <p>Echo your Username: %username&</p>	
<p>الخروج من موجه الأوامر أو الملف الدفعي الحالي.</p>	Exit
<p>البحث عن عبارة نصية ضمن ملف ما. وله الخيارات:</p> <ul style="list-style-type: none"> 7 عرض الأسطر التي لا تحوي جملة البحث. C عرض فقط عدد الأسطر التي تحوي جملة البحث. N عرض الأسطر مع أرقامها. I تجاهل حالة الأحرف. <p>للبحث عن "ss" في ملف ما مع عرض أرقام الأسطر ودون مراعاة حالة الأحرف:</p>	Find
<p>Find "ss" File.txt /I /n</p> <p>كما يمكن البحث عن جملة البحث في عدة نصوص.</p> <p>للبحث عن الأسطر التي لا تحتوي على العبارة السابقة في ملفين مثلا:</p> <p>Find "ss" File1.txt File2.txt /v /n /i</p>	
<p>عرض أو تعديل مشغلات أنواع الملفات.. حيث يمكن عرض أو إضافة أنواع الملفات الموجودة في الكمبيوتر من خلال الأمر assoc، فعند كتابة assoc. txt ستحصل على txt = txtfile .. وهذا يعني أن اسم مشغل الملفات من نوع txt هو txtfile.</p> <p>لعرض مسار مشغل ملفات معين:</p>	Ftype
<p>Ftype txtfile</p> <p>ستحصل على مسار برنامج المفكرة، متبوعا بإشارة %1، والتي تخبر نظام التشغيل أن عليه تشغيل الملف الحالي من خلال برنامج المفكرة، إذ إن %1 هو متغير يحمل مسار الملف الذي يتم التعامل معه.</p> <p>لضبط مشغل ملفات جديد، مثلا ملفات من نوع myra والتي ستتحدد عنه في فصل الرجساري:</p> <p>Ftype myrafile="E:\REG\RegApp.exe" "%1"</p> <p>وذلك بعد إنشاء الصيغة في رجساري الكمبيوتر من</p>	Ftype



خلال الأمر assoc .myra = myrafile كما رأينا في أمر سابق.	
توجيهي موجه الأوامر للانتقال لسيطرة معنون من خلال إشارة :، مثلا:	
:a . .	Goto
Goto a	
عرض معلومات عن أوامر موجه الأوامر. وهذا الأمر أحد مصادر لي لهذا الفصل.	
لعرض أوامر موجه الأوامر بالترتيب:	
Help	Help
Help dir	عرض معلومات عن أمر معين مثلا dir
Dir /?	أو
بنية شرط لها ثلاثة أشكال:	
IF [NOT] ERRORLEVEL number command	
IF [NOT] string1==string2 command	
IF [NOT] EXIST filename command	
الكلمة not في حال وضعت تعني نفي الشرط،	
والأمر errorlevel رأيناه في الأمر choice إذ إنه يحوي	
قيمة عدديّة تشير إلى ترتيب الخيار الذي تم	
اختياره. command هو الأمر المطلوب تنفيذه. أما	
filename فهو مسار ملف ما.	
تستخدم بنية الشرط عادةً في الملفات الدفعية،	
وهي تتحكم بطريقة تنفيذ الملف الدفعي.	
الصيغة الأولى تستخدم مع الأمر choice، مثلا:	If
CHOICE /C YN /M "Press Y for Yes, N for No."	
If errorlevel 1 goto yes	
If errorlevel 2 goto no	
وبالنسبة للصيغة الثانية فهي تستخدم مع المتغيرات، لاحظ:	
If %a% == new echo welcome	
والتي تعني إذا كان المتغير a مساو للقيمة new	
فاكتبه أهلا. يتم ضبط المتغيرات من خلال الأمر set.	
أما الصيغة الثالثة فتستخدم عند التعامل مع	
الملفات، وبإمكانك استخدام بنية if else كالتالي:	
IF EXIST d:\file.txt (
del d:\file.txt	



<pre>) ELSE (echo d:\file.txt missing.) كما يمكن استخدام الخيار a مع الأمر if لتصبح بالشكل: IF [/I] string1 compare-op string2 command IF CMDEXTVERSION number command IF DEFINED variable command حيث compare-op يأخذ إحدى القيم التالية: Equ مساوي. Neq ليس مساوي. Lss أقل. Leq أقل أو يساوي. Gtr أكبر. Geq أكبر أو يساوي. لاحظ المثال: IF %ERRORLEVEL% LEQ 2 goto okay والذي يعني إذا كانت القيمة المختارة أكبر أو تساوي 2 انتقل للسطر المعنون بـ okay.</pre>	
<p>عرض وتعديل اسم محركات الأقراص. لعرض اسم القرص d:</p> <p>Label d:</p> <p>لضبط اسم جديد للقرص d:</p> <p>Label d: MyDisk</p> <p>لإزالة اسم القرص (استعادة الافتراضيات)، قم بعرض اسم القرص من خلال الأمر الأول، ثم اضغط Enter ثم y.</p>	Label
<p>إنشاء مجلد. وهو نفسه mkdir.</p> <p>في حال كنت في مسار المجلد الذي ترغب بإنشاء مجلد فيه يكفي كتابة اسمه.</p>	Md
<p>تنسيق موجه الأوامر، من حيث الأبعاد. لتغيير أبعاد نافذة موجه الأوامر:</p> <p>Mode 50,30</p>	Mode
<p>نقل الملفات وإعادة تسمية الملفات أو المجلدات. ويأخذ الخيارات التالية: - عدم إظهار رسالة تأكيد عند وجود نفس الملف. - إظهار رسالة تأكيد عند وجود نفس الملف.</p>	Move
<p>التحكم بالشبكة المحلية، وهو ضخم جدا يحتاج لكتاب لشرحه.. لعرض الأقسام التي يتعامل معها الأمر:</p>	Net



<p>Net</p> <p>عرض معلومات عن الأمر: Net help</p> <p>عرض معلومات عن قسم معين مثلا :user</p> <p>Net help user</p> <p>عرض قائمة بأسماء الأجهزة المتصلة بالشبكة وأسماء مستخدمي الكمبيوتر:</p> <p>Net user</p> <p>إضافة مستخدم: Net user Name /add</p> <p>حذف مستخدم: Net user Name /delete</p> <p>تغيير كلمة سر المستخدم: Net user Name Pass</p> <p>حيث Name اسم المستخدم وPass كلمة السر.</p>	
<p>أمر مخصص بال شبكات، وهو أضخم من القبليه.</p> <p>عرض أسماء الشبكات المحفوظة في الكمبيوتر:</p> <p>Netsh wlan show profiles</p> <p>عرض معلومات عن شبكة معينة اسمها WL:</p> <p>Netsh wlan show profiles WL</p> <p>كما أنه لعرض كلمة السر يكتب .key = clear</p>	Netsh
<p>إيقاف موجه الأوامر بشكل مؤقت وإظهار العبارة:</p> <p>إضغط أي مفتاح للاستمرار....، وذلك في الملفات الدفعية.</p>	Pause
<p>تغيير العبارة التي تظهر في كل سطر قبل الأوامر والتي قلنا أن اسمها المحت، وافتراضيا هي مسار الفهرس الحالي.</p> <p>يمكن للمحت أن يأخذ القيم التالية:</p> <p>\$a إشارة & \$b إشارة \$c قوس مفتوح) \$d التاريخ الحالي \$e مفتاح الهروب (مفتاح أسكى له 27) \$f قوس مغلق () \$g إشارة أكبر > \$h مسح حرف للخلف \$i إشارة أصغر < \$n القرص الحالي \$p المسار الحالي (بما فيه القرص الحالي) \$q إشارة مساواة =</p>	Prompt



	\$ فراغ \$ الوقت الحالي \$ إصدار ويندوز \$ سطر جديد \$ إشارة الدولار \$ لجعل المحت عبارة نصية ول يكن Eng27 Prompt Eng27 كما يمكن استخدام أكثر من قيمة معًا: Prompt Eng27\$g و لإعادته لوضعه الافتراضي وهو \$P\$G اكتب .prompt	
حذف المجلدات.	Rd	
.rd	Rmdir	
ترك ملاحظة في الملفات الدفعية، وهو مكافئ للمحرف \\ المستخدم في C# لترك الملاحظات. والملاحظات هي أسطر لا يقرأها المترجم، وكما هو معروف فإن الملاحظات تستخدم لعنونة الأكواد ولترك تلميحات ومذكرات يعاد إليها عند الحاجة.	Rem	
إعادة تسمية الملفات. حيث يوضع مسار الملف المراد إعادة تسميته ثم الاسم الجديد. ren e:\Folder\file1.txt File.txt حيث تم تغيير اسم الملف الواقع في المجلد Folder ضمن القرص e من الاسم file1.txt إلى File.txt كما يمكنك عدم كتابة مسار الملف كاملاً إذا كنت في فهرسه. كما يمكن استخدام الأمر move لإعادة التسمية.	Ren	
.ren	Rename	
التحكم بالمتغيرات الخاصة ببيئة دوس وإظهارها. وهو أحد الأوامر التي من الممكن أن تستفيد منها خلال حياتك. وذلك بالشكل التالي: إظهار المتغيرات الموجودة في نظام dos: Set		
عرض المتغيرات التي تبدأ بحرف ما أو مجموعة من الحروف ول يكن :an	Set	
Set an إنشاء متغير وإسناد قيمة له:		
Set a = 5 استقبال قيمة من المستخدم وتخزينها في متغير: Set /p a = Enter a: والناتج سيكون عبارة نصية تطالبك بإدخال قيمة ل a.		



التحكم بالملف الدفعي باستخدام حلقة if بناءاً على قيمة متغير، المثال التالي هو ملف دفعي:

```
@echo off
```

```
Set /p ans = Choose A, B or C:
```

```
If %ans% = a goto a
```

```
If %ans% = b goto b
```

```
If %ans% = c goto c
```

```
Echo you can choose a, b or c ONLY!
```

```
a:
```

```
echo you have choosed a!
```

```
goto d
```

```
b:
```

```
echo you have choosed b!
```

```
goto d
```

```
c:
```

```
echo you have choosed c!
```

```
d:
```

```
pause
```

أما المتغيرات الداخلية فهي كثيرة ومفيدة، ويمكن استخدامها بإحاطتها بإشارة % من الطرفين. أهم هذه المتغيرات:

AppData مسار مجلد هام يستخدمه نظام التشغيل في حياته اليومية وهو Roaming اسم الكمبيوتر.

ComputerName مسار موجه الأوامر. ComSpec قرص النظام.

HomeDrive مسار المستخدم. HomePath

.LocalAppData مسار مجلد هام آخر اسمه Local. Number_Of_Processors عدد أنواع المعالج.

ProgramFiles مسار مجلد البرامج. ProgramFiles(x86) مسار مجلد البرامج 32.

PROMPT شكل المحت.

Temp أو TEMP مسار المجلد .Temp. UserName اسم المستخدم.

UserProfile مسار مجلد المستخدم. WinDir مسار مجلد Windows.

كما يوجد متغيرات أخرى مثل: Date التاريخ الحالي.

Time الوقت الحالي.

Random متغير عشوائي يأخذ قيمة صحيحة من 0



وحتى 32727.	
الملف الدفعي يتمثل كمسار بمتغير باسم 0، حيث يمكن استخدامه بكتابة إشارة % واحدة قبله، لاحظ: Copy %0 d:\Hasan%random%.txt والكود السابق يأمر الملف الدفعي أن ينسخ نفسه إلى القرص d ويفير اسمه إلى Hasan ويليه رقم عشوائي ويجعل صيغته .txt.	
خيارات الطاقة في ويندوز. ويأخذ المتغيرات التالية: S إيقاف تشغيل الكمبيوتر. R إعادة تشغيل الكمبيوتر. L تسجيل الخروج (لا يمكن استخدامه مع C أوا). I عرض مربع حوار خيارات الطاقة للشبكات. T إجراء العملية بعد زمن ما (مقدرا بالثانية). C ترك تعليق للمستخدم. A إحباط العملية في حالة كانت مجدولة. F إيقاف قسري لجميع البرامج قيد العمل. والجدير بالذكر أن الزمن في حال لم يتم تحديده من خلال المتغير t هو 60 ثانية.	Shutdown
تشغيل برنامج في نافذة جديدة. وله الخيارات: Min تشغيل البرنامج في وضع التصغير. Max تشغيل البرنامج في وضع التكبير. Low التشغيل بأولوية ضعيفة. BelowNormal التشغيل أقل من عادية. Normal التشغيل بأولوية عادية. AboveNormal التشغيل بأولوية أكثر من عادية. High التشغيل بأولوية عالية. RealHigh التشغيل بأولوية أعلى من العالية. Wait تشغيل البرنامج بعد إصدار أمر جديد. "Title" إظهار عنوان لبعض البرامج. يمكن تشغيل البرامج التالية من خلال هذا الأمر: - البرامج العادية (بكتابة مسارها). - البرامج الموجودة داخل System32، مثل calc explorer cmdg notepad وغيرها.	Start
معلومات عامة عن الكمبيوتر.	Systeminfo
قائمة بالبرامج قيد العمل.	Tasklist
إيقاف العمليات الجارية.. هذا الأمر والذي سبقه معا يشكلان Alt + Ctrl + Del.	Taskkill
تغير اسم الملف الدفعي أو موجه الأوامر الحالي.	Title
عرض أو تعديل الوقت الحالي. لعرض الوقت فقط اضغط Enter بعد استخدام الأمر.	Time



<p>إيقاف موجه الأوامر أو الملف الدفعي عن العمل لمدة ما. وله الخيارات:</p> <ul style="list-style-type: none"> T المدة الزمنية التي سيتم الإيقاف المؤقت عليها. NoBreak عدم إمكانية كسر الأمر من قبل المستخدم باستخدام C. Ctrl + C + Ctrl + C 	<p>Timeout</p>
<p>عرض محتويات الفهرس الحالي أو فهرس ما. وله الخيارات:</p> <ul style="list-style-type: none"> F إظهار أسماء الملفات أيضا. A استخدام أرقام أسكبي. <p>حفظ ناتج تنفيذ الأمر في ملف ما:</p> <p>Tree d:\Folder /f /a > e:\File.txt</p>	<p>Tree</p>
<p>عرض محتويات الملفات (فتحها للقراءة فقط).</p>	<p>Type</p>
<p>معرفة نسخة نظام التشغيل.</p>	<p>Ver</p>
<p>لمعرفة معلومات عن القرص الصلب.</p>	<p>Vol</p>
<p>نسخ ملف أو أكثر أو مجلد أو أكثر من مكان إلى آخر.</p> <p>وله الخيارات:</p> <ul style="list-style-type: none"> A نسخ الملفات التي لها صفحة الأرشيف فقط. P التأكيد قبل النسخ. S نسخ جميع الملفات والمجلدات الفرعية بما فيها الفارغة. E نفسه S دون الفارغة. V التتحقق من كل ملف جديد. W إعطاء تنبيه ليدخل المستخدم أي زر للبدء. C الإكمال في حال وجود أخطاء. Q عدم إظهار أسماء الملفات أثناء النسخ. F إظهار المسار وأسماء الملفات أثناء النسخ. L إظهار الملفات التي سيتم نسخها. G السماح بنسخ الملفات المشفرة. H نسخ ملفات النظام والملفات المحمية إن وجدت. R الكتابة فوق الملفات التي تحمل صفة للقراءة فقط. n نسخ الملفات الموجودة في المجلد الهدف. K نسخ خصائص الملفات مع الملفات. Y عدم التنبيه عند وجود نفس الاسم. ـ- التنبيه عند وجود نفس الاسم (الافتراضي). 	<p>Xcopy</p>

جرب جميع الأوامر السابقة لتأخذ فكرة عنها وأي أمر يستعصي عليك اكتب اسمه ثم ?, لتحصل على شرح مفصل عنه.



ملاحظة

- قد تحتاج لتشغيل موجه الأوامر كمسؤول لتتمكن من تنفيذ بعض الأوامر.

عمليات متقدمة على الأوامر

- حفظ ناتج أمر في ملف خارجي

في بعض الأحيان قد تحتاج لحفظ ناتج أحد أوامرك في ملف خارجي تستفيد منه كتقرير لعملية ما، خصوصا عند أنتمة برامحك، أو قد لا تتمكن من عرض ناتج تنفيذ أحد الأوامر في cmd لضخامتها فتضطر لحفظ الناتج في ملف خارجي وقراءته هناك، أو قد ترغب باستعراض عضلاتك في نظام dos أمام أحد معارفك. أيا كان هدفك فإمكانك القيام بهذه العملية بالصيغة التالية:

cmd's code > مسار

فمثلا انتقل للقرص e عبر الأمر: e ثم اكتب dir > c:\EFiles.txt، ليتم إنشاء ملف نصي في القرص c باسم EFiles مكتوب فيه محتويات e.

لكن انتبه، ففي حال وجود الملف سيتم استبداله عند استخدام هذا الأمر، أي أن بياناتك ستتحذف، لذلك فكن حذرا مع هذا الأمر. وعوضا عن ذلك إذا أردت الكتابة إلى نهاية ملف موجود مسبقا فاستخدم الإشارة <>، وفي كلا الحالتين سيتم إنشاء ملف جديد في حال عدم وجوده.

ملاحظة

- عند حفظ ناتج تنفيذ أمر ما في ملف خارجي فالناتج لن يعرض في شاشة موجه الأوامر، لذلك عند استخدامك إياه يجب أن تكون قد درست ناتج الأمر أولا بتطبيقه دون استخدام هذه العملية.
- عندما لا يكون الأمر الذي تستخدمه من الأوامر التي يتعارف عليها نظام dos - على الأقل على كمبيوترك - فإنه سيتم كتابة عبارة وجود خطأ في الملف الخارجي، مع تطبيق البند الأول من هذه الملاحظة.

- تنفيذ أكثر من أمر في سطر واحد

يوفر لك موجه الأوامر إمكانية تنفيذ أكثر من كود في سطر واحد وذلك من خلال الربط بين الأوامر بإشارة &، حيث بإمكانك تنفيذ العديد والعديد من



الأوامر بوجود هذا الرمز، لكن انتبه، عند وجود خطأ ما (أمر لا يعرفه نظام التشغيل)، فإن الأوامر ما بعد الأمر الخاطئ لن تنفذ، أما الأوامر التي قبله فستنفذ بسلسة.

هذا يعني أن الأمر الخاطئ إذا كان أولاً لن يتم تنفيذ شيء، وإذا كان آخر أمر ضمن سطرك المدمج فسيتم تنفيذ جميع الأوامر إلا واحداً.

- التنقل بين الأوامر

يمكن التنقل بين الأوامر باستخدام الأسهم في لوحة المفاتيح، بإمكانك التنقل بين الأسطر التالية والسابقة باستخدام مفتاحي للأعلى والأسفل، وإمكانك التنقل بين أحرف الأمر الحالي باستخدام مفتاحي يمين ويسار.

بالإضافة لهذه التنقلات البسيطة بإمكانك التنقل باستخدام بعض الأزرار الأخرى، والتي سأستعرضها لك كما يلي:

المفتاح F1: يقوم بكتابة آخر أمر تم استخدامه، وذلك بمعدل حرف واحد من اليسار إلى اليمين من أجل كل ضغطة للمفتاح F1.

المفتاح F2: يعطيك مربع حوار يطلب منك إدخال حرف، فيقوم بنسخ جميع أحرف الأمر السابق حتى الحرف السابق للحرف الذي أدخلته.

المفتاح F3: يقوم بكتابة آخر أمر تم استخدامه (مماثل لاستخدام مفتاح التنقل للأعلى لكن لمرة واحدة).

المفتاح F4: يقوم بشيء ما

المفتاح F5: هو نفسه المفتاح للأعلى.

المفتاح F6: يطبع ^Z، وهذا التركيب مماثل لإدخال المفاتحين Z و Ctrl، ويحمل صفات هامة عن موجه الأوامر الحالي مثل حالة Echo هل هي On أم Off، أو بالإمكان فتح موجه أوامر جديد عند استخدام Start ^Z، وأمور أخرى.

المفتاح F7: يعرض الأوامر في ذاكرة موجه الأوامر، هل سألت نفسك كيف يتذكر موجه الأوامر الأوامر السابقة؟! بالتأكيد هناك ذاكرة وهنا بالإمكان التعامل معها. عند اختيار أحد الأوامر المحفوظة في الذاكرة فإنه سينفذ فوراً.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Hp>eng27
'eng27' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Hp>echo off
echo eng27
eng27
```

The screenshot shows a Windows Command Prompt window. The user has typed 'eng27' and received an error message stating it's not a recognized command. Then, they typed 'echo off' followed by 'eng27' again, which is displayed in the history at the bottom of the window.

المفتاح F8: مشابه لاستخدام مفتاح التنقل للأسفل.

المفتاح F9: يقوم بتنفيذ أحد الأوامر الموجودة في الذاكرة بناء على رقم الأمر (لاحظ أن أول عنصر رقمه 0).

بقية الأزرار لا وظيفة لهم (على حد علمي).

- التعامل مع عدة ملفات

جميع الأوامر التي تعرفت عليها كانت تعامل مع ملف واحد بعينه، قد تحتاج للتعامل مع عدد كبير من الملفات أو نوعية معينة من الملفات، لاحظ:

للتعامل مع جميع الملفات ذات امتداد واحد ولتكن txt:

cmd's code *.txt

للتعامل مع جميع الملفات التي تحمل اسم معين من جميع الامتدادات:

cmd's code A.*

للتعامل مع جميع الملفات من جميع الامتدادات:

cmd's code *.*

للتعامل مع الملفات المسماة بعدد معين من الأحرف وجميع الامتدادات:

cmd's code ???.*



للتعامل مع الملفات المسممة بعدد معين من الأحرف وبامتداد معين ولتكن :txt

cmd's code ????.txt

في المثالين الآخرين، سيتم التعامل مع جميع الملفات التي عدد أحرف اسمها ثلاثة، ومن نوع نصي أو من أجل جميع أنواع الملفات.

أخطاء قد تواجهك

- Bad command or filename -
أمر خاطئ أو ملف غير موجود.
- Access denied -
لا يمكن الوصول إلى الملف بسبب الحماية.
- Abort, Retry, Fail -
يعني أنك تحاول القراءة من قرص غير قابل للقراءة أو غير موجود.
- Devide overflow -
قسمة غير ممكن (القسمة على 0).
- Drive not ready -
الجهاز غير جاهز، قد يمشي الحال¹ إذا نزعته وأعدت اتصاله.
- Duplicate filename or file not found -
إعادة تسمية ملف غير موجود.
- File not found -
الملف غير موجود كما تلاحظ يا عزيزي.
- Insufficient disk space -
لا يوجد مساحة كافية للأسف.
- Invalid filename or file not found -
لم يتمكن دوس من إيجاد الملف في المجلد المحدد.

هذا بالإضافة إلى العبارة الذي ستتكرر معك خلال حياتك:

'ss' is not recognized as an internal or external command,
operable program or batch file.

والتي تعني أنك تكتب أمر غير معترف عليه.

¹ يمشي الحال: يكتمل، يكمل بالنجاح..



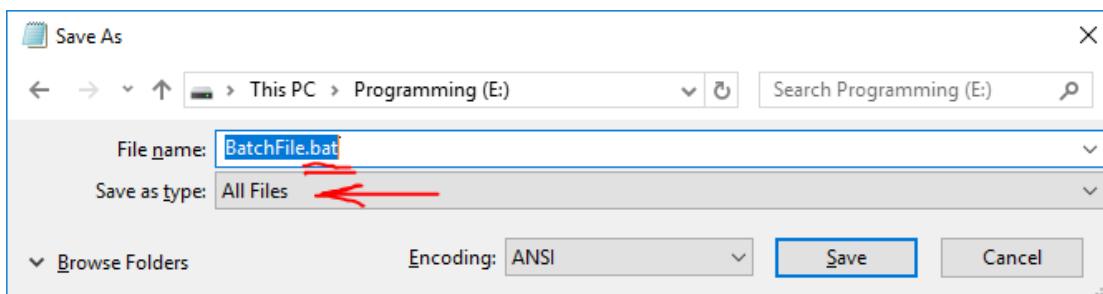
الملفات الدفعية



log off

تم التنويه سابقاً إلى أن الملفات الدفعية هي ملفات تقوم بتنفيذ أوامر موجه الأوامر دفعه واحدة.. وقد تحتاج لذلك عند أتمتة العمليات البرمجية أو لإنشاء تطبيقات صغيرة تقوم بمهام محددة.

لإنشاء ملف دفعي قم بفتح مستند نصي جديد ثم احفظه بالشكل التالي:



يفضل دائماً الابداء بالأمر @Echo off في الملفات الدفعية لإخفاء المحت، حيث يقوم الأمر Echo off بإلغاء ظهور المحت للأوامر التالية أما الرمز @ فيلغيه للأمر الحالي.. هذه الحركة تعطيك شاشة فارغة محتوياتها فقط ستكون نواتج تنفيذ الأوامر.

في حال كان ملفك الدفعي سيجري محادثة مع المستخدم، يجب إنهاء ملفك الدفعي بالأمر pause والذي يوقف شاشة موجه الأوامر ريثما يقوم المستخدم بإدخال أحد المفاتيح (مناظر ل() Console.ReadKey في C#).

وما بين الأمرين السابقين ستتوارد أوامر تطبيقاتك والتي ستناقشها في الأمثلة التالية:

- معرفة المشغلات الموجودة في الكمبيوتر وحفظها في ملف نصي:

```
@echo off
echo Your Computer Associations: > "e:\File Types.txt"
Assoc >> "e:\File Types.txt"
echo ----- >>"e:\File Types.txt"
echo Done at %time% >>"e:\File Types.txt"
```



وما سيحدث هو أن ملفًا نصيًّا سُيُبَدِّأ بعبارة نصية، ثم يُكتب بعدها مشغلات ملفات كمبيوترك، ثم خطوط ثم عبارة تقول للمستخدم أنه تم تنفيذ الأمر في الوقت الفلاني.

لاحظ أنه استخدمنا إشارة < مرة واحدة عند أول أمر وإشارتين عند بقية الأوامر، والسبب في ذلك أن الأولى تقوم باستبدال الملف في حال وجوده، أما الثانية فتكتب في نهاية الملف في حال وجوده.

قد تقول لي: أين الأمر pause؟! وسأجيبك أنه لا يهمني أن تظهر شاشة موجه الأوامر للمستخدم.. أنا فقط تهمني النتائج التي ستحفظ في ملف نصي.

- حفظ سمات الملفات في فهرس ما في ملف خارجي:

```
@echo off
echo File attributes in D: > "File Attribs.txt"
attrib d:\*.* >> "File Attribs.txt"
echo ----- >>"File Attribs.txt"
echo Done.. (%time%) >>"File Attribs.txt"
```

- إخفاء جميع الملفات النصية والتي عدد الأحرف المكونة لاسمها هو 3، الإخفاء سيكون بشكل متقدم بحيث لا يمكن عرض الملفات بتفعيل خيار عرض الملفات والمجلدات المخفية:

```
@echo off
attrib ????.txt +h +s
```

- قد تحول جميع ملفات أحد الأقراص إلى ملفات نظام مخفية (كما في المثال السابق لكن على مستوى القرص ككل)، ويكون ذلك غالباً بفعل فيروس مايغ (ساذج بلغة أفلام الكرتون)، ولاستعادتها إلى وضعها الأصلي يمكنك استخدام الكود التالي:

```
@echo off
attrib /s -h -s
REM Don't forget to use parameter 's' with 'h'
```

- جرب الكود التالي هدية مني لك:

```
@echo off
title matrix By Eng27
```



```
color 02
:start
echo
%random%%random%%random%%random%%random%%random%%random%%ran
dom%%random%%random%%random%%random%%random%
goto start
```

- يمكن إنشاء ملف ليعطيك إمكانيات إيقاف التشغيل في كمبيوترك:

```
@echo off
:start
title PowerTools by Eng.27
color 02
echo Welcome %username% to PowerTools By Eng.27
echo Programmed by Eng27
echo -----
echo Menu:
echo a) Shutdown
echo b) Restart
echo c) Log off
echo d) Exit
set/p answer=Choose [a-d]:
if %answer% == a goto a
if %answer% == b goto b
if %answer% == c goto c
if %answer% == d goto d
echo you have to choose a, b, c or d only
pause
goto start
:a
cls
echo Your computer will Shutdown in 10 sec.
echo you can abort the operation by typing Shutdown -a in cmd or RUN
shutdown -s -t 11 -c "Shutdown, By Eng.27"
goto d
color 4
:b
color 4
echo Your computer will Restart in 10 sec.
echo you can abort the operation by typing Shutdown -a in cmd or RUN
shutdown -r -t 11 -c "Restart, By Eng.27"
goto d
:c
color 2
echo Your computer will Log off Now
shutdown -l
```



```
:d
color 2
pause
```

سيتم إجراء العمليات السابقة إما مباشرة أو بعد 11 ثانية، يمكن تغيير الموضوع حسب الحاجة. كما يمكن إحباط العملية من خلال الأمر -a shutdown، يمكن إضافته إلى القائمة السابقة بنفس الملف الدفعي ويمكن تنفيذه عن طريق Run أو من خلال موجه الأوامر.

كما يمكن إنشاء اختصار يقوم بإيقاف التشغيل (أو أي مهمة أخرى)، الجميل فيه هو إمكانية تغيير أيقونته إلى أي أيقونة تريد لأنه اختصار:

أنشئ اختصاراً جديداً، واتكتب المسار التالي في مسار عمله:

Shutdown.exe -s -t 10 -c "your computer will shut down in 10 s"

ثم غير أيقونته من خصائصه، واجعلها مثلاً أيقونة زر الطاقة.



وبالمناسبة فإن هذه الحركة (تغيير الأيقونة) بإمكانك القيام بها على الملفات الدفعية عموماً بشكل غير مباشر، أنشئ ملفاً دفعياً واحفظه في مكان ما مثلاً في القرص E، أرسله إلى سطح المكتب كاختصار، ثم غير اسم وأيقونة الاختصار وفقاً لاحتياجك.



- بإمكانك أيضاً إنشاء ملف دفعي يفتح لك برامج أو مجلدات معينة..
وبنفس فكرة المثال السابق فإننا سنستفيد من بنية الشرط، لكن سنتطبق فكرة جديدة وهي إنشاء ملف دفعي فرعي يقوم بتنفيذ الأوامر لاحظ:

```
@echo off
title ProgramsShell by Eng27
echo Menu:
echo a) Calc
echo b) Notepad
echo c) cmd
set/p ans = Choose [a-c]:
if %ans% == a goto a
if %ans% == b goto b
if %ans% == c goto c
you can choose a, b or c only!
pause
:a
call e:\calc.bat
exit
:b
call e:\notepad.bat
exit
:c
call e:\cmd.bat
```

Cmd

Notepad

calc

بات لديك أربعة ملفات دفعية:

 BatchFile	٢٠١٨/١٢/٢٢ ١٠:٣٤	Windows Batch File	1 KB
 calc	٢٠١٨/١٢/٢٢ ١٠:٣٤	Windows Batch File	1 KB
 cmd	٢٠١٨/١٢/٢٢ ١٠:٣٤	Windows Batch File	1 KB
 notepad	٢٠١٨/١٢/٢٢ ١٠:٣٤	Windows Batch File	1 KB

أرسل الملف الأول كاختصار لسطح المكتب وغير أيقونته لما يناسب عمله، ثم جربه. (هل لاحظت أن هذه العملية أشبه بالإجراءات والتوابع؟)



- يمكنك إنشاء وحذف المستخدمين:

```
إضافة مستخدم Rem  
Net user MyNewUser /add
```

```
حذف مستخدم Rem  
Net user MyNewUser /delete
```

```
ضبط كلمة مرور Rem  
Net user MyNewUser *  
[اكتب كلمة السر]  
[أكدها]
```

أنا كمستخدم Windows 10 Pro لم أتمكن من إضافة مستخدم آخر لا أدرى لماذا، فالحل كان باستخدام موجه الأوامر وتحديداً بالأسطر البرمجية السابقة..

- نسخ أي ملف أو برنامج إلى بدء التشغيل، وبالتالي فإنه يعمل عند تشغيل المستخدم:

```
Copy e:\File.txt  
"C:\Users\%username%\AppData\Roaming\Microsoft\Windows\Start  
Menu\Programs\Startup
```

كما يمكن وضع الملفات في سطح المكتب لكافه المستخدمين مثلا:

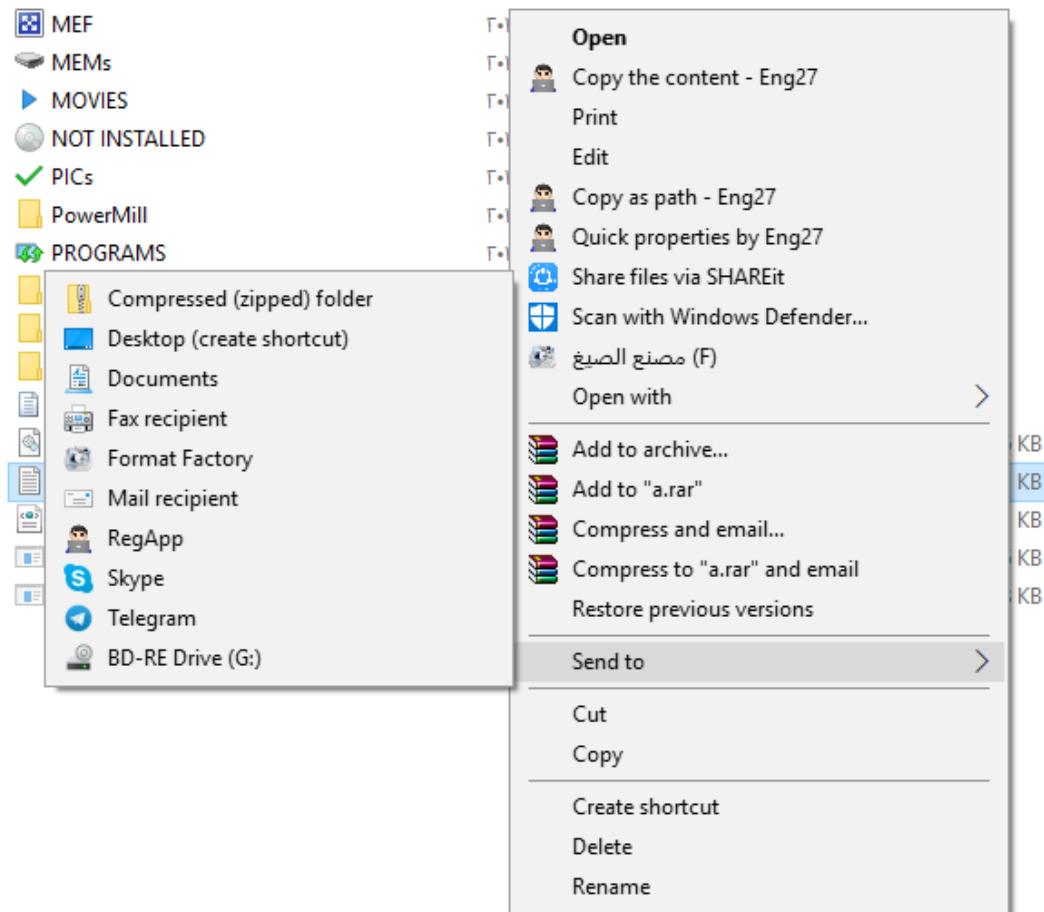
```
Copy e:\File.txt C:\Users\Public\Desktop
```

قد تستفيد من هكذا حركة في تطبيقاتك، فعند نشرها يمكن إرسال نسخة من التطبيق كاختصار إلى موقع مختلفة من كمبيوترك.

من المسارات التي قد تلزمك:

```
C:\Users\%username%\AppData\Roaming\Microsoft\Windows\SendTo
```

وهذا المسار توضع فيه البرامج التي من الممكن إرسال الملفات من خلالها، كما هو موضح بالصورة التالية:



لاحظ أن محتويات هذه القائمة موجودة بجانب بعضها كما يلي:

📁 Compressed (zipped) folder	٢٠١٧/٠٣/١٨ ٢٣:٠١	Compressed (zipp...)	1 KB
💻 Desktop (create shortcut)	٢٠١٧/٠٣/١٨ ٢٣:٠١	Desktop Shortcut	1 KB
📄 Documents	٢٠١٨/٠٦/٥ ١٨:١٢	MyDocs Drop Tar...	0 KB
📠 Fax recipient	٢٠١٧/٠٣/١٨ ٢٢:٥٩	Shortcut	2 KB
📠 Format Factory	٢٠١٨/٠٦/٦ ١٢:١٨	Shortcut	2 KB
✉️ Mail recipient	٢٠١٧/٠٣/١٨ ٢٣:٠١	Mail Service	1 KB
👤 RegApp	٢٠١٨/١٢/٢٢ ١٤:٣٣	Shortcut	1 KB
♫ Skype	٢٠١٨/٠٦/٦ ١١:٠٥	Shortcut	3 KB
📠 Telegram	٢٠١٨/٠٨/٢٣ ٠٩:٣٩	Shortcut	2 KB

هذه الحركة سنسخدمها في الفصل الذي يتحدث عن الرجسٰtri إن شاء الله.

تغليف الملفات الدفعية

كما لاحظت فإن الملفات الدفعية يمكن اكتشاف محتوياتها بسهولة، فهي بسيطة لا يمكن المناورة بها.. ومع قليل من الخبرة ستتمكن من فهم الغاية من أي ملف دفعي تصادفه أو على أقل تقدير ستتمكن من إيجاد



طريقة لفهم هذا الملف وذلك عبر البحث على الانترنت على بعض الأسطر الموجودة فيه أو الاطلاع على بعض المراجع وغيرها من الأساليب..

إن عملية تغليف الملفات الدفعية هي عبارة عن تحويلها إلى ملفات تنفيذية exe، وهذا لا يتم بطريقه مباشرة.. هناك أكثر من طريقة لذلك:

- من خلال ضغط الملف الدفعي وتحويل الملف المضغوط الناتج إلى ملف تنفيذى exe، وذلك بتفعيل خيار Create SFX archive.

- من خلال لغات البرمجة، وذلك بإنشاء مشروع برمجي يقوم بإنشاء ملف دفعي يكتب فيه الأسطر البرمجية المطلوب تنفيذها، ثم يقوم المشروع بتشغيل هذا الملف عبر كائن مستنسخ من الفئة Process في C# مثلاً، وذلك عبر الطريقة Start.File.Delete، ويفضل انتظار العملية ريثما تنتهي عبر الطريقة WaitForExit (التابعة للفئة Process) والتي تعطي أمراً لبرنامجك ليترىث قليلاً حتى انتهاء الملف الدفعي من تنفيذ أوامرها.. وبعدها يقوم بحذفه.

حماية الملفات والبيانات

مع ازدياد حاجة الإنسان للحواسيب واعتماده عليها كان لا بد من إيجاد طرق وأساليب لحماية البيانات المخزنة على هذه الحواسيب. وطرق الحماية كثيرة، فبإمكانك حماية ملفاتك بإخفاءها أو تشفيرها أو ضغطها وحمايتها بكلمة سر وغيرها من الطرق (شخصياً أحمي ملفاتي بوضعها في سلة المهملات ).

طبعاً عندما نقول حماية الملفات بإخفاءها فإننا لا نقصد إخفاءها بالطريقة العادية بتاتاً بتاتاً، وإنما نقصد إخفاءها بشكل كامل بحيث لا يمكن إظهارها حتى لو تم تفعيل خيار "إظهار الملفات والمجلدات المخفية" (إلا إذا كانت الغاية من الحماية هي إبعاد الملفات عن متناول الأطفال ففي هذه الحالة لا بأس باستخدام الطريقة العادية .

من الممكن أيضاً حماية الملفات بتغيير صيغها، فمثلاً بإمكانك إنشاء ملفات تخزن فيها بيانات نصية وحفظها على أنها ملفات mp3 مثلاً، في الواقع هذه الملفات ليست إلا ملفات نصية لكن الظاهر (لنظام التشغيل وللمستخدم) أن هذه الملفات هي ملفات mp3 تالفة.

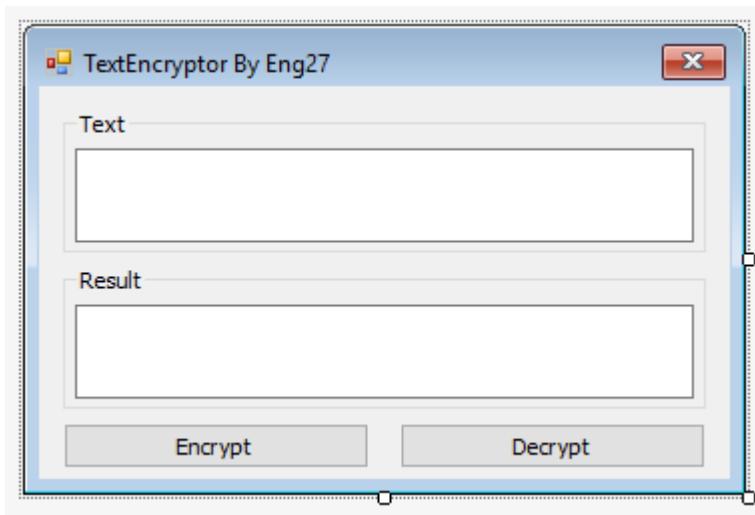
سنناقش طرقاً عديدة لحماية الملفات والبيانات وذلك عبر عدة مشاريع، بعضها يعتمد على موجه الأوامر وبعضها لا يعتمد..



تشغير النصوص

يمكن تشغیر النصوص بأكثر من طريقة وأكثر من خوارزمية، تختلف قوتها من خوارزمية لأخرى وفقاً لصعوبة اختراقها أو صعوبة اكتشاف خوارزميتها¹.

صمم النافذة التالية:



استخدم الكود التالي:

```
namespace Encrypt1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Encrypt_Click(object sender, EventArgs e)
        {
            textBox2.Clear();
            List<char> text = new List<char>();

            for (int i = 0; i < textBox1.TextLength; i++)
            {
                text.Add (Convert.ToChar(textBox1.Text.Substring(i, 1)));
            }
        }
    }
}
```

¹ هناك طرق جاهزة للتشغیر، إذا كانت هذه غايتك فأعتذر منك ربما قد غرّك عنوان الفقرة، فما سأناقشه هو كيفية إنشاء برنامج تشغیر بسيط.

أكرر هناك طرق جاهزة للتشغیر وأنا لن أناقشها.



```
foreach (char oldL in text)
{
    int byteL = (int)oldL;
    intL += 1;
    char newL = (char)intL;
    textBox2.Text += newL.ToString();
}
}

private void Decrypt_Click(object sender, EventArgs e)
{
    textBox2.Clear();
    List<char> text = new List<char>();
    for (int i = 0; i < textBox1.TextLength; i++)
    {
        text.Add(Convert.ToChar(textBox1.Text.Substring(i, 1)));
    }
    foreach (char oldL in text)
    {
        int byteL = (int)oldL;
        intL -= 1;
        char newL = (char)intL;
        textBox2.Text += newL.ToString();
    }
}
```

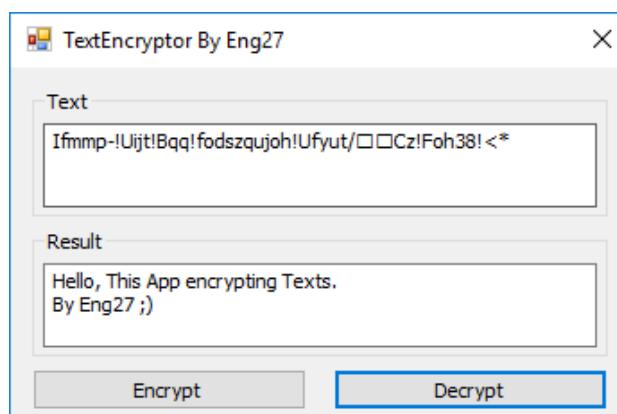
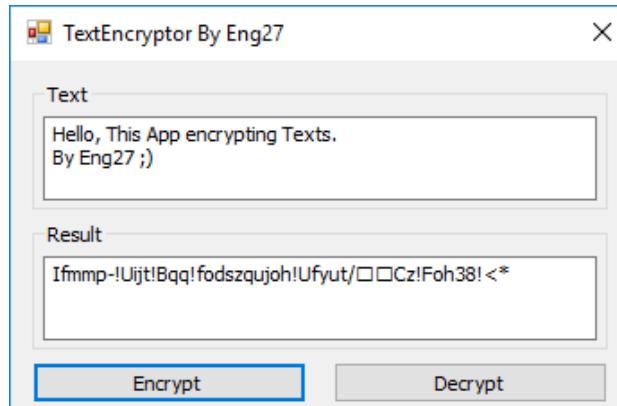
وال فكرة من الكود هو إسناد جميع أحرف عبارة نصية ما إلى لائحة أو مصفوفة ثم الحصول على الأحرف التالية لها وإظهارها في عبارة نصية جديدة. لاحظ المثال:

إن ناتج تشفير الكلمة ENG27 هي FOH38، بمعنى أن خوارزميتنا في هذا المثال مبنية على الحصول على المحرف التالي للمحرف الحالي. طبعاً يمكانك بناء خوارزميات كثيرة وفق رغبتك..

القيمة المشفرة	القيمة المراد تشفيرها
F	E
O	N
H	G
3	2
8	7



لاحظ الأمثلة التالية في البرنامج:





من الممكن استخدام هذا التطبيق لتشفيـر الرسائل أو البيانات النصية أو غيرها، فمثلاً بإمكانك تشفـير البيانات المرسلة إلى قاعدة بيانات على سبيل المثال وفك تشفـير البيانات المستوردة منها، كالبيانات الشخصية أو الإدارية مثلاً. صحيح أنه يمكن حماية وسائل التخزين كقواعد البيانات بكلمات سر لكن لا يوجد شيء غير قابل للاختراق ☺.

يـجدر الانتـباـه إـلـى أـنـ المـحـارـفـ مـرـتـبـةـ وـفـقـ جـدـولـ معـيـنـ يـسـمـىـ جـدـولـ ASCII¹، فـحـرـفـ Aـ مـثـلـاـ لـاـ يـلـيـهـ aـ وـإـنـماـ Bـ، وـهـكـذـاـ. أـوـ بـالـأـخـرـيـ فـإـنـ الرـمـوزـ وـالـمـحـارـفـ الـيـونـانـيـةـ مـرـتـبـةـ وـفـقـ هـذـاـ الجـدـولـ، أـمـ أـنـكـ أـرـسـلـتـهـاـ عـبـرـ بـالـبـرـيدـ أـوـ أـيـةـ وـسـيـلـةـ نـقـلـ إـلـىـ أـحـدـ أـصـدـقاءـكـ..ـ (ـأـوـ جـرـبـ سـلـةـ الـمـهـمـلـاتـ 😊ـ).

حماية الملفات داخل صورة

إـحـدىـ تـطـبـيقـاتـ نـظـامـ الدـوـسـ هـيـ إـخـفـاءـ الـمـلـفـاتـ دـاخـلـ مـلـفـ صـوـرـةـ،ـ وـالـذـيـ يـتـيـحـ لـكـ حـمـاـيـةـ مـلـفـاتـكـ مـنـ الغـرـاءـ سـوـاءـ أـرـدـتـ تـخـزـينـ الـمـلـفـاتـ عـلـىـ كـمـبـيـوتـرـكـ وـأـنـتـ مـرـتـاحـ الـبـالـ أـنـهـاـ لـنـ يـتـمـ الـوصـولـ إـلـيـهـاـ،ـ أـمـ أـنـكـ أـرـسـلـتـهـاـ عـبـرـ بـالـبـرـيدـ أـوـ أـيـةـ وـسـيـلـةـ نـقـلـ إـلـىـ أـحـدـ أـصـدـقاءـكـ..ـ (ـأـوـ جـرـبـ سـلـةـ الـمـهـمـلـاتـ 😊ـ).

وـكـمـ أـسـلـفـتـ فـإـنـ العـمـلـيـةـ تـتـمـ عـلـىـ نـظـامـ الدـوـسـ بـشـكـلـ كـامـلـ،ـ أـيـ فـيـ مـوـجـهـ الـأـوـامـرـ وـالـمـلـفـاتـ الـدـفـعـيـةـ.ـ سـتـقـولـ لـيـ مـادـوـرـ C#ـ هـنـاـ؟ـ؟ـ؟ـ فـيـ الـوـاقـعـ دـورـ C#ـ هـنـاـ ثـانـويـ،ـ فـوـظـيـفـتـهـ إـعـطـاءـ النـوـافـذـ لـبـرـنـامـجـكـ وـتـنـظـيمـ الـمـدـخـلـاتـ وـالـمـخـرـجـاتـ وـالـتـوـاسـطـ بـيـنـ الـمـسـتـخـدـمـ وـنـظـامـ الدـوـسـ حـتـىـ تـكـلـلـ الـعـمـلـيـةـ بـالـنـجـاحـ.ـ فـنـظـامـ الدـوـسـ وـC#ـ فـيـ هـذـاـ تـطـبـيقـ كـالـعـلـاقـةـ بـيـنـ htm~lـ وـcssـ فـيـ تـطـبـيقـاتـ وـمـوـاـقـعـ الـوـيـبـ!ـ أـوـ كـالـعـلـاقـةـ بـيـنـ الـفـتـاةـ وـالـكـوـافـيرـ 😍ـ.

كـبـدـاـيـةـ سـنـقـومـ بـالـعـمـلـيـةـ فـيـ نـظـامـ الدـوـسـ (ـأـوـ بـالـأـخـرـيـ باـسـتـخـدـامـ الـمـلـفـاتـ الـدـفـعـيـةـ عـلـىـ اـعـتـبـارـ أـنـنـاـ نـتـعـاـمـلـ مـعـ نـظـامـ وـيـنـدـوـزـ):

أـنـشـئـ مـلـفـاـ نـصـياـ جـديـداـ وـاـكـتـبـ الـكـوـدـ التـالـيـ دـاخـلـهـ:

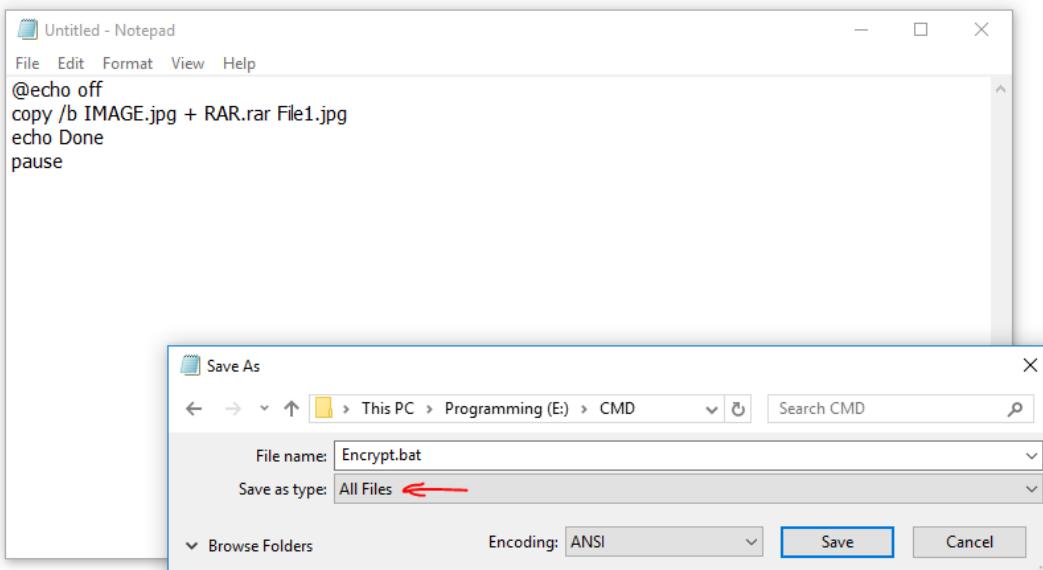
```
@echo off
copy /b IMAGE.jpg + RAR.rar File1.jpg
echo Done
pause
```

¹ راجع ملحقات الكتاب، الملحق أ – جدول ASCII.

² راجع الملحق الذي يلي الملحق السابق.



احفظ الملف كما يلي:



نوع الملفات يجب أن يكون "جميع الملفات" ولاعبة الملف هي bat (لاحقة الملفات الدفعية).

أنشئ ملفاً دفعيا آخر واتكتب فيه:

```
@echo off  
ren File1.jpg File1.rar  
echo Done  
pause
```

واحفظه بذات الأسلوب باسم .Decrypt.bat

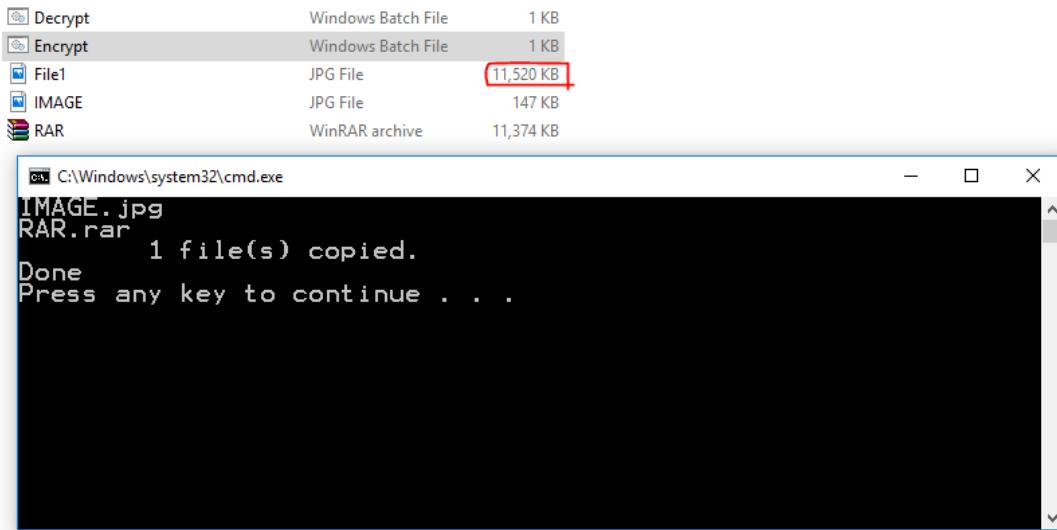
اختر صورة بريئة تخفي بها ملفاتك وسمها IMAGE، كهذا:



أضف الملفات المطلوب إخفاءها إلى ملف مضغوط وسمه RAR. أصبح لديك الملفات التالية:

	Decrypt	Windows Batch File	1 KB
	Encrypt	Windows Batch File	1 KB
	IMAGE	JPG File	147 KB
	RAR	WinRAR archive	11,374 KB

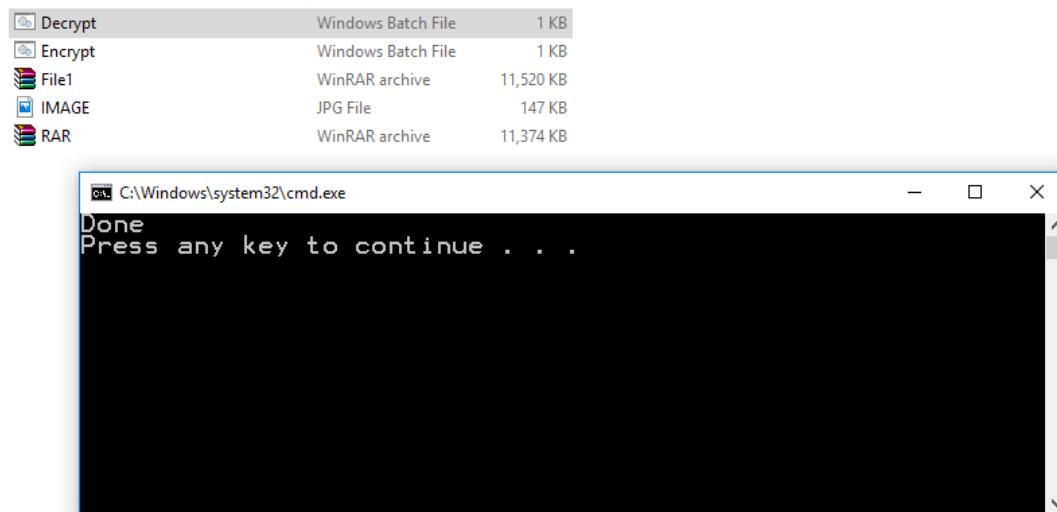
شغل ملف التشفير Encrypt ليزداد الملفات ملفاً ويظهر أمامك ما يلي:



افتح الصورة File1 لتجد الصورة البريئة التي اخترتها ولكن بحجم رهيب، 11.5 ميغابايت!!

الآن الصورة File1 هي الصورة التي تحوي بياناتك المخفية عن الأنظار - بطريقة متقدمة - والتي بإمكانك تخزينها وأنت مرتاح البال أو إرسالها إلى أي شخص تريده بحيث يكون لديه الملف لفك تشفير الصورة.

لفك تشفير الصورة شغل الملف Decrypt:



حيث تم تحويل الصورة File1 إلى ملف مضغوٌط، في الواقع تمت إعادة تسميتها فقط تحت بيئة دوس..

شايف العمليات التي قمنا بها للتو؟؟ سنجعل C# ينظم ويدير هذه الملفات للحصول على برنامج تفاعلي يعطي ذات النتيجة.



لكن قبل ذلك نحتاج لضغط الملفات قبل تشفيرها، ولا أعتقد أنه من كرم الضيافة أن تطلب من المستخدم - في حال نشرت تطبيقك - أن يقوم بضغط الملفات ثم إعطائك إياها، وإنما من الأنسب أن تقوم أنت - كمبرمج - بضغطها، وللقيام بذلك سنتعيين بنظام الدوس مرة أخرى وسنضغط الملفات من خلاله، أنشئ ملفاً نصياً جديداً واكتبه فيه:

Rar a RAR Files

احفظه باسم RarFiles.bat بنفس الأسلوب السابق، حيث أن هذا الملف سيقوم بضغط المجلد المسمى Files في ملف مضغوط باسم RAR، لذلك أنشئ مجلداً باسم Files تضع فيه الملفات المطلوب تشفيرها.

لكن لحظة، نظام الدوس لا يمكنه ضغط الملفات، وإنما يقوم بتوجيه برنامج ضغط الملفات لهذا الأمر. لذلك فعليك الحصول على برنامج ضغط الملفات مدعومة بمكتبة تحوي فئات خاصة بضغط الملفات. برأيك من هو أفضل من يضغط الملفات؟؟؟ أعتقد أن WinRAR يجول في خاطرك الآن.. سنذهب إليه ونستعير منه ملفين صغيرين ولا أعتقد أنه سيمانع لأنه معطاء وجود وإمكانك استخدام النسخة التجريبية منه مدى الحياة وبكافحة الإمكانيات. المهم، انتقل لمسار WinRAR في قرص نظام التشغيل واستعر منه الملفين Rar.exe و RarExt.dll، وضعهما في ذات مسار الملف RarFiles. أصبح لديك الملفات التالية:

Name	Type	Size
Files	File folder	
Decrypt	Windows Batch File	1 KB
Encrypt	Windows Batch File	1 KB
IMAGE	JPG File	147 KB
Rar	Application	584 KB
RarExt.dll	Application extens...	427 KB
RarFiles	Windows Batch File	1 KB

الملفات هي ملفان دفعيان للتشغير وفكه، صورة تخفي فيها ملفاتك، برنامج ومكتبة الضغط، ملف دفعي يدير عملية الضغط، ومجلد تضع فيه الملفات المطلوب ضغطها.

الآن ضع بعض الملفات في المجلد Files، ثم شغل الملف RarFiles:



Files	File folder	
Decrypt	Windows Batch File	1 KB
Encrypt	Windows Batch File	1 KB
IMAGE	JPG File	147 KB
Rar	Application	584 KB
RAR	WinRAR archive	11,374 KB
RarExt.dll	Application extens...	427 KB
RarFiles	Windows Batch File	1 KB

الآن أصبحت العملية كالتالي:

أولاً: يجب توفير الملفات Decrypt.bat وEncrypt.bat و Rar.exe و RarFiles.bat، وبطبيعة الأحوال المجلد الفارغ .Files.

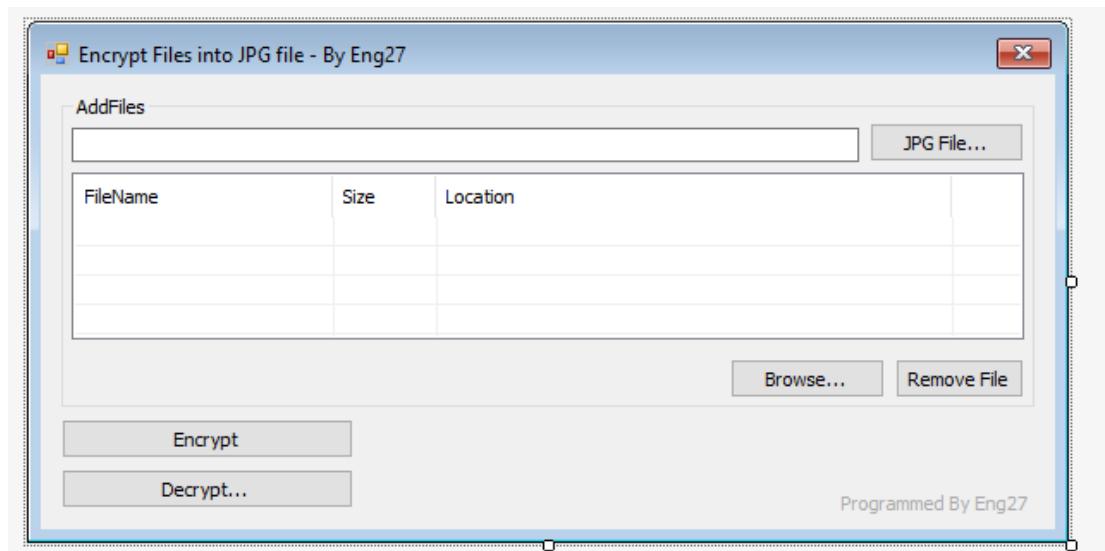
ثانياً: يجب توفير صورة باسم IMAGE.jpg

ثالثاً: يتم ضغط الملفات الموجودة داخل المجلد Files عبر تشغيل الملف RarFiles.bat والذي يتعامل مع ملفي الضغط.

رابعاً: يتم تشفير البيانات عبر تشغيل الملف Encrypt.bat

خامساً: يتم فك التشفير عبر تشغيل الملف Decrypt.bat

هذا كلّه سنقوم به عن طريق C# 😊، لنبدأ باسم الله، أنشئ مشروع جديداً وصمم النافذة التالية:



اضبط الخاصية Enabled على false، وأضف الأعمدة في ListView من الأمر Columns كما هو في الشكل



واضبط الخاصية GridLines على true لعرض شبكة من الأسطر والاعمدة ثم MultiSelect على false تُعد إمكانية التحديد المتعدد ثم View على Details لعرض العناصر على شكل تفاصيل.

ملاحظة

- عادةً ما توضع ثلات نقاط أمام أسماء الأزرار والقوائم التي ستفتح نوافذ جديدة، لهذا فإن بعض أزرار المشروع تم تطبيق هذه الفكرة عليها.

الزر JPG File سيفتح لك نافذة فتح الملفات لاختيار الصورة التي سنخفي فيها الملفات، حيث يتم وضع مسارها في مربع النص المجاور، ويمكن لصق مسار الصورة مباشرةً.

الزر Browse سيفتح لك نافذة فتح الملفات لاختيار الملفات التي سيتم تشفيرها لتضاف تفاصيلها إلى الأداة ListView، أما الزر Remove File فيحذف العنصر المحدد من ListView (لاحظ أنه لن يظهر نافذة جديدة لذلك لم نضع ثلاثة نقاط أمام اسمه)، هذا الزر لن يُفعّل مالم يتم تحديد عنصر ما من ListView.

الزر Encrypt سيقوم بتشذير الملفات، أي أنه سيستدعي الملف RarFiles لضغطها ثم Encrypt لتشذيرها (هنا لن يتم فتح نافذة جديدة).

الزر Decrypt سيفتح لك نافذة فتح الملفات والتي تتيح للمستخدم البحث عن الصورة التي فيها الملفات المشفرة – المخفية – ليقوم بفك تشذيرها.

استخدم الكود التالي:

```
using System.IO;
using System.Diagnostics;
namespace Encrypt3
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            DirectoryInfo Files = new DirectoryInfo(
Application.StartupPath.ToString() + @"\Files");
            foreach (FileInfo file in Files.GetFiles())
```



```

    {
        File.Delete(file.FullName); // حذف جميع الملفات القديمة
    }
    حذف الملفات القديمة الموجودة بجوار البرنامج //
    File.Delete(Application.StartupPath + "\\image.jpg");
    File.Delete(Application.StartupPath + "\\archive.rar");
    File.Delete(Application.StartupPath + "\\DecryptedFiles.rar");
}
التحقق من إمكانية التشفير //
{
    في حال كان هناك عناصر في أداة العرض //
    وفي نفس الوقت الصورة التي سيتم التشفير فيها موجودة في الكمبيوتر //
    فإنه من الممكن تشفير الملف //
    if (listView1.Items.Count > 0 & File.Exists(textBox1.Text))
        Encrypt.Enabled = true;
    else
        Encrypt.Enabled = false;
    Remove.Enabled = false; // إخفاء زر حذف العنصر
}
هذا الإجراء يقوم بضغط الملفات //
{
    var f = File.Create(Application.StartupPath + "\\RarFiles.bat");
    f.Close(); // إنشاء الملف الدفعي الذي سيضغط الملفات
    StreamWriter FileW = File.AppendText(Application.StartupPath +
"\\RarFiles.bat");
    FileW.WriteLine("Rar a Archive Files");
    FileW.Close();
    Process p = new Process();
    ProcessStartInfo pInfo = new ProcessStartInfo(f.Name);
    pInfo.StartInfo = pInfo;
    pInfo.Start();
    بدون هذا الكود سيتم الانتقال للتعليمية التالية مباشرة //
    دون انتظار الملفات ليكتمل ضغطها //
    كما تعلم فإن ضغط الملفات يحتاج وقت خصوصا الكبيرة منها //
}
void BatEncrypt()
{
    var f = File.Create(Application.StartupPath + "\\Encrypt.bat");
    f.Close();
    StreamWriter FileW = File.AppendText(Application.StartupPath +
"\\Encrypt.bat");
    FileW.WriteLine("copy /b image.jpg + archive.rar encrypted.jpg");
    FileW.Close();
    Process.Start(f.Name);
}
void BatDecrypt(string path)
{
}

```



```

var f = File.Create(Application.StartupPath + "\\Decrypt.bat");
f.Close();
StreamWriter FileW = File.AppendText(Application.StartupPath +
"\\Decrypt.bat");
FileW.WriteLine("ren \"{0}\" DecryptedFiles.rar",path);
FileW.Close();
Process.Start(f.Name);
}
private void Browse_Click(object sender, EventArgs e)
{
    OpenFileDialog open = new OpenFileDialog();
    open.Multiselect = true; //لنتتمكن من فتح أكثر من ملف معاً
    if (open.ShowDialog() == System.Windows.Forms.DialogResult.OK)
    {
        مسارات الملفات التي تم فتحها//foreach (string FileItem in Files)
        {
            طريقة إضافة تفاصيل لعناصر أداة العرض// FileInfo file = new FileInfo(FileItem);
            ListViewItem L;
            L = listView1.Items.Add(file.Name);
            long size = file.Length;
            string Unit = "Bytes";
            if (size < 1e6) //1e6 تعني 1x10^6
            {
                Unit = " KB";
                size /= 1024;
            }
            else if (size > 1e6 & size < 1e9)
            {
                Unit = " MB";
                size /= 1048576;
            }
            L.SubItems.Add(size.ToString() + Unit);
            L.SubItems.Add(file.FullName);
        }
    }
    CanEncrypt();
}
private void listView1_SelectedIndexChanged(object sender, EventArgs e)
{
    foreach (ListViewItem L in listView1.Items)
        if (L.Selected)
            Remove.Enabled = true;
        else
            Remove.Enabled = false;
}

```



```

}

private void Remove_Click(object sender, EventArgs e)
{
    foreach (ListViewItem L in listView1.Items)
        if (L.Selected)
            L.Remove();
    CanEncrypt();
}

private void Encrypt_Click(object sender, EventArgs e)
{
    string fullpath;
    foreach (ListViewItem Item in listView1.Items)
    {
        نسخ جميع الملفات إلى مجلد التشفير//
        fullpath = Application.StartupPath + @"\Files\" +
        Item.SubItems[0].Text;
        File.Copy(Item.SubItems[2].Text, fullpath);
    }
    نسخ الصورة التي سيتم التشفير فيها وتحويل اسمها//
    File.Copy(textBox1.Text, Application.StartupPath + @"\image.jpg");
    ضغط الملفات//;
    BatEncrypt();//
    تشفير الملفات//;
    MessageBox.Show("Files encrypted successfully!", "FilesToJPG By
Eng27");
    فتح الملف الحاوي على التطبيق وملفاته//;
    Process.Start(Application.StartupPath);
}

private void JPGfile_Click(object sender, EventArgs e)
{
    OpenFileDialog open = new OpenFileDialog();
    open.Filter = "JPG File | *.jpg";
    if (open.ShowDialog() == System.Windows.Forms.DialogResult.OK)
    {
        textBox1.Text = open.FileName;
    }
}

private void Decrypt_Click(object sender, EventArgs e)
{
    OpenFileDialog open = new OpenFileDialog();
    open.Filter = "JPG File | *.jpg";
    if (open.ShowDialog() == System.Windows.Forms.DialogResult.OK)
    {
        BatDecrypt(open.FileName);
        MessageBox.Show("Files decrypted successfully!", "FilesToJPG By
Eng27");
        Process.Start(open.FileName.Replace(open.SafeFileName,
"DecryptedFiles.rar"));
    }
}

```



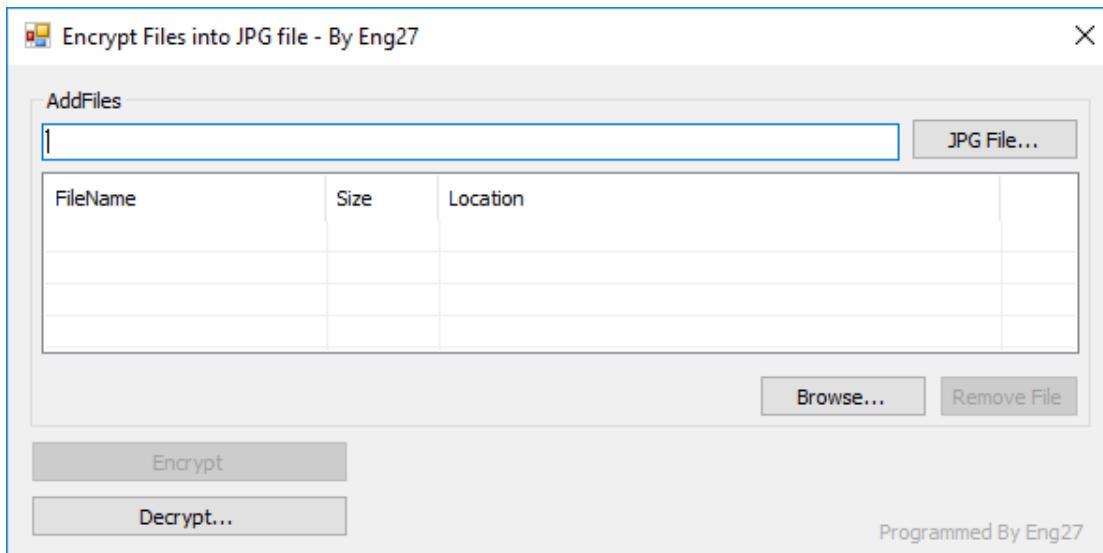
```

        }
    }

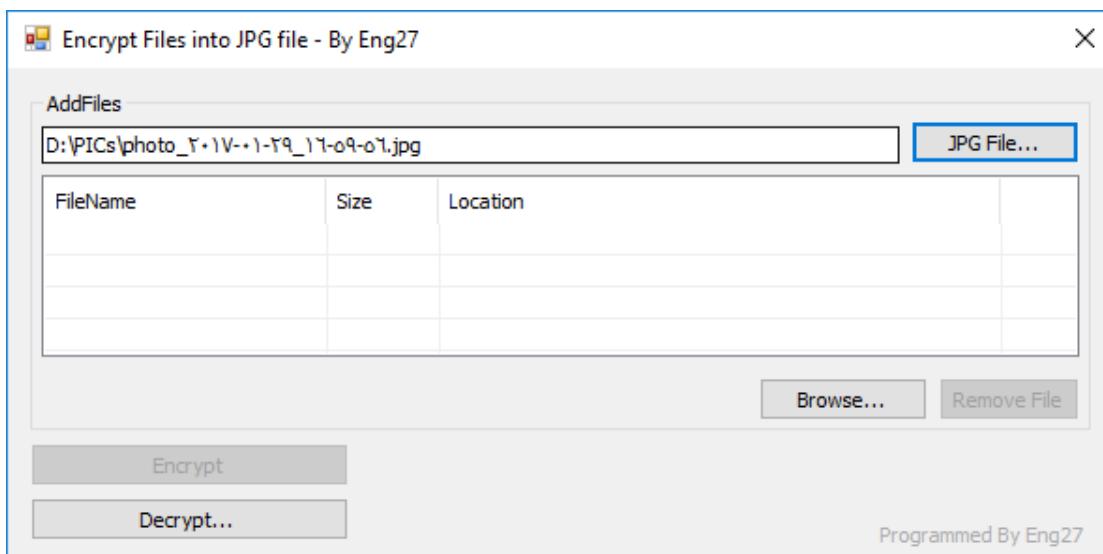
    private void textBox1_TextChanged(object sender, EventArgs e)
    {
        CanEncrypt();
    }
}

```

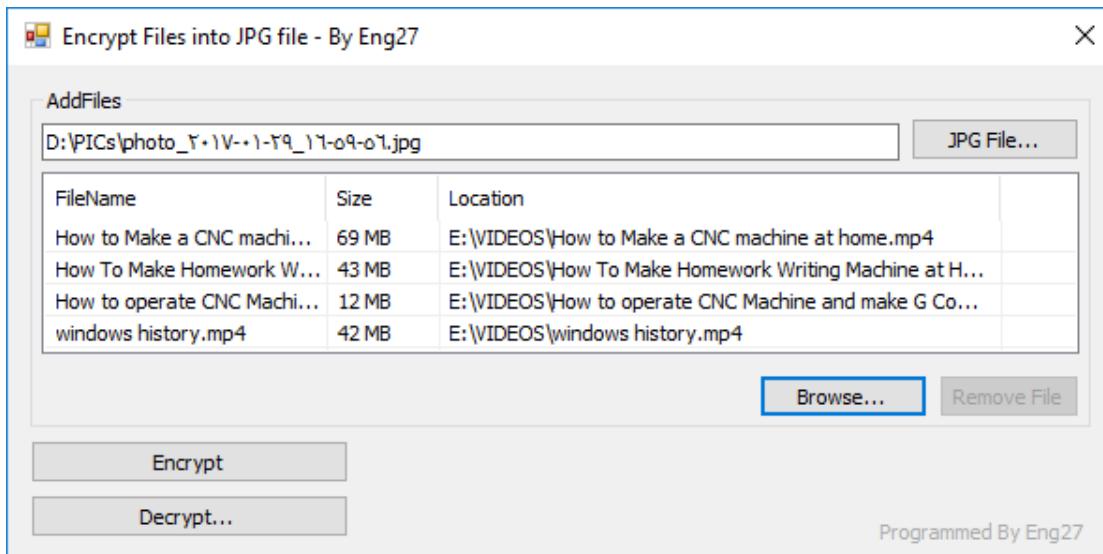
عند تشغيل التطبيق ستحصل على النافذة التالية:



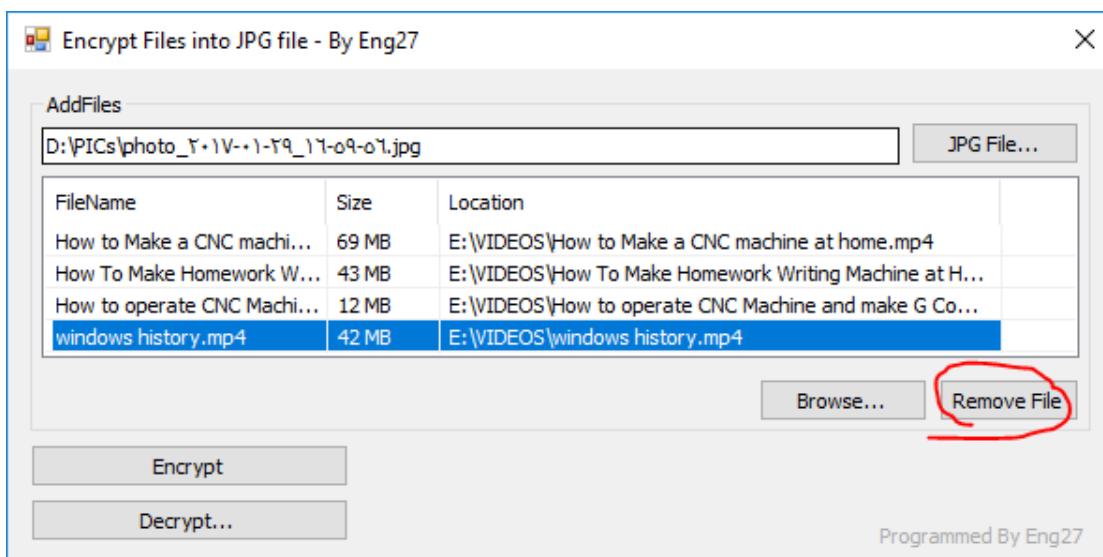
اضغط على JPG File لاختيار صورة تختفي بها ملفاتك:



اضغط Browse لاستعراض الملفات بحثا عن الملفات المراد إخفاءها داخل الصورة:



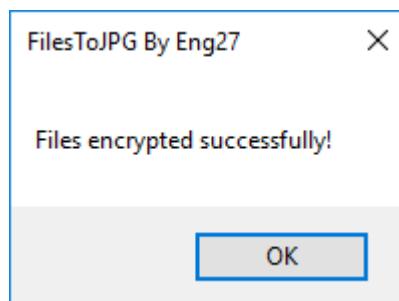
لاحظ تفعيل زر التشفير وذلك بسبب وجود عناصر في أداة العرض كما أن مسار الصورة صحيح موجود.
في حال وجود ملف لا ترغب بتشفيه مع الملفات أنقر عليه ثم اضغط إزالة ملف:



بعدها أنقر على Encrypt، وانتظر حتى تكتمل العملية:



```
C:\Windows\system32\cmd.exe
C:\Users\Hp\Documents\Visual Studio 2012\Projects\Encrypt3\Encrypt3\bin\Debug>Rar a Archive Files
RAR 5.40 x64 Copyright (c) 1998-2016 Alexander Roshal 15 Aug 2016
Trial version Type RAR -? for help
Evaluation copy. Please register.
Creating archive Archive.rar
Adding   Files\How to Make a CNC machine at home.mp4      OK
Adding   Files\How To Make Homework Writing Machine at Home.mp4    OK
Adding   Files\How to operate CNC Machine and make G Code.mp4      66%
Adding   Files\Microsoft history .mp4
```



ستفتح نافذة فيها تطبيقك وبجواره الصورة الملغومة، لاحظ:

	encrypted	٢٠١٨/١١/٢٢ ٣:١:٤	JPG File	195,204 KB
	image	٢٠١٨/٠٨/٢٧ ٤:٥:٦	JPG File	54 KB

لاحظ الفرق بين الصورتين (محتواهما ذات الصورة، على الثقة المحتوى نفسه ما في داعي أعرض لك إياهما). لاحظ الفرق في حجوم الصور!

لاستعادة الملفات اضغط Decrypt، لا داعي لأدلك على الطريق بتَّ تعرفه.

حماية الملفات داخل قواعد البيانات

بإمكانك حماية الملفات بتخزينها في قواعد البيانات بطريقتين، إما بتخزينها بالكامل بقاعدة البيانات، أو بتخزين مساراتها فقط.

تتميز الطريقة الأولى بالأمان، لكنها تتسبب بحجم كبير لقاعدة البيانات كما أنها قد تسبب بطء في عمل البرنامج. أما الثانية فإنها أسرع وأقل حجماً، لكنها أقل أماناً!

وعادةً ما تكون الملفات المخزنة هي صور سواءً بالطريقة الأولى أو الثانية. وهذا ما سنتحدث عنه في الفصول الأخيرة من هذا الكتاب.



حماية الملفات بتحويلها لملفات نظام مخفية

يعتمد هذا النوع من الحماية على الأمر attrib، فكما تعلم فإن هذا الأمر يقوم بضبط سمات الملفات والمجلدات. ومن السمات التي من الممكن ضبطها هي سمة ملفات النظام.

بإمكانك ضبط سمة "ملف نظام" من موجه الأوامر وفق الأمر:

```
Attrib +h +s File.txt
```

وتزال السمة بإبدال إشارة - بـ +.

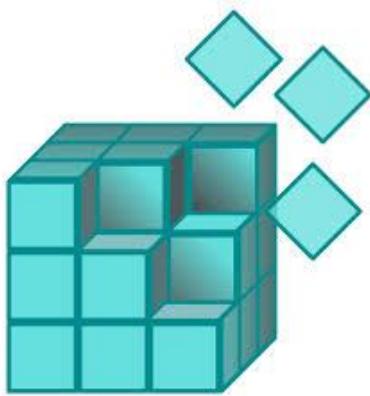
وكما تعلم فإن سمة ملفات النظام المتمثلة بالمتغير S وسمة الإخفاء المتمثلة بالمتغير h يجب أن يضطعا معًا.. سواءً للضبط أو الإزالة. (لم تكن تعلم؟؟ انتقل لجدول أوامر cmd فورًا ورأسًا ومبشرةً).





الفصل الحادي عشر - الرجسٰري

يعتبر محرر السجل أو الرجسٰري Registry إحدى أكثر أدوات نظام التشغيل شيوعاً بين المبرمجين، فهو يتيح لك الوصول لنظام التشغيل بشكل مباشر. يمكن تشبّيئه بالديوان في دوائر الدولة، فيه تحفظ كل شاردة وواردة في نظام التشغيل، بدءاً بالقطع المادية وانتهاءً بالمكونات البرمجية.



يمثل الرجسٰري قاعدة بيانات ضخمة يحتوي على أغلب المعلومات والإعدادات المتعلقة بنظام التشغيل والكثير من البرامج المثبتة عليه، وهذه الإعدادات تُعدّل تلقائياً من قبل نظام التشغيل. وعند حدوث أي فقد للمعلومات قد يتضرر نظام التشغيل لذلك ينصح بأخذ نسخة احتياطية عنه قبل التعديل عليه.

من المحتمل أن تكون هذه أول مرة تسمع بها بالرجسٰري، لذلك سأضع شرحاً موجزاً عنه، بما في ذلك التعامل معه والاستفادة منه وحتى التأثير على نظام التشغيل من خلاله، ثم سأنتقل لكيفية التعامل معه من خلال برمجك في C#. إذا لم تكن لك رغبة بتعلم الرجسٰري فانتقل مباشرةً لفقرة الرجسٰري من خلال C#.

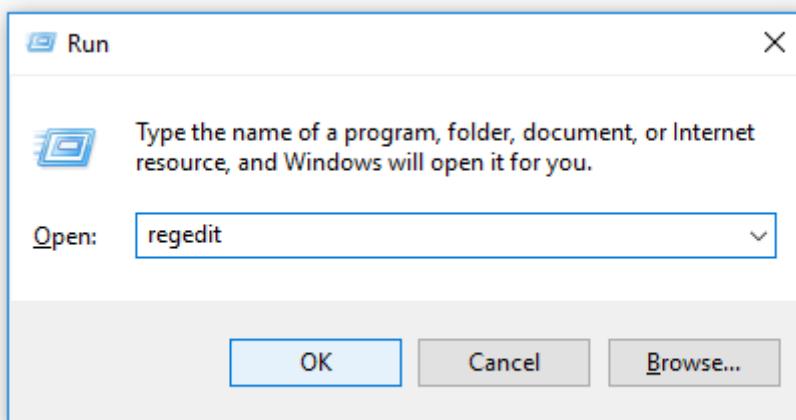
لتثبيت محرر التسجيل - للتعامل مع الرجسٰري - اكتب في مربع البحث في قائمة ابدأ الكلمة regedit أو من خلال الأمر RUN. على العموم فهو



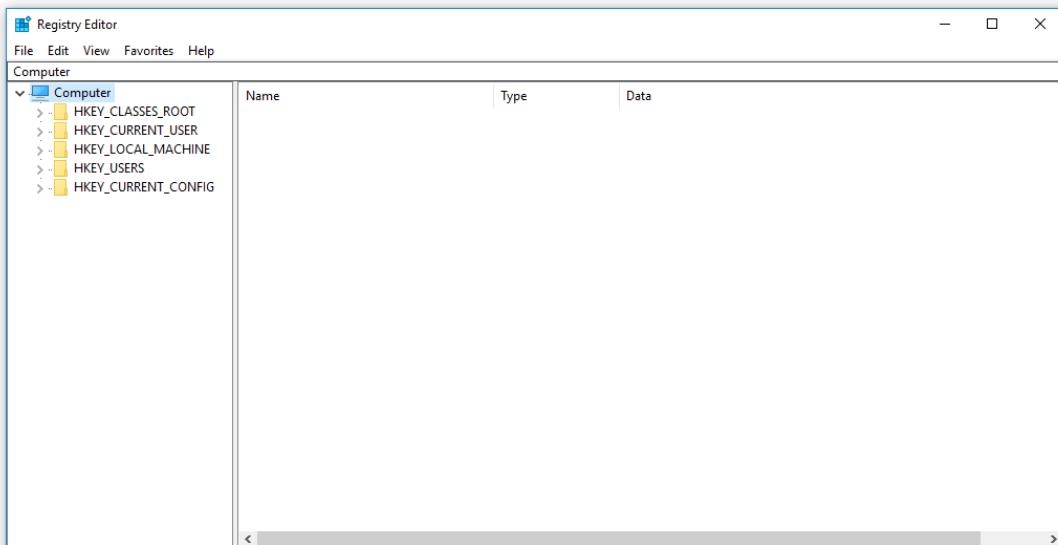
من الأدوات التي تأتي مع نظام التشغيل مثل الآلة الحاسبة والرسام .cmd وموجه الأوامر.

ملاحظة

- يمكن الوصول للأمر تشغيل عن طريق Win+R.



النافذة التالية هي نافذة محرر السجل:



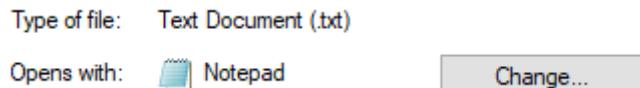
لا تغّرك المظاهر، مئات الآلاف من المعلومات موجودة في النافذة السابقة! وهي موزعة وفق ما يسمى بالخلايا كما يلي:

HKEY_CLASSES_ROOT •

وتحتوي على كل ما يخص الملفات من معلومات، مثل أنواع الملفات ومشغلاتها، وأيقوناتها وتفاصيلها. فمن خلال هذه الخلية يعرف نظام



التشغيل أن الملفات التي لاحقتها **txt** هي ملفات نصية وأيقونتها هي أيقونة ملف نصي، وعند فتح خصائص هذا الملف يقول لك أنه مستند نصي **Text Document**. لاحظ الصورة التالية من خصائص أحد الملفات النصية:



HKEY_CURRENT_USER •

وفيها يتم حفظ معلومات وإعدادات المستخدم الحالي، كشاشة التوقف الحالية وإعدادات الانترنت وغيرها من الإعدادات الشخصية.

HKEY_LOCAL_MACHINE •

وتحتوي هذه الخلية على معلومات عن المكونات المادية والبرمجية لحاسوبك.

HKEY_USERS •

وتحتوي على معلومات عن المستخدمين لحاسوبك.

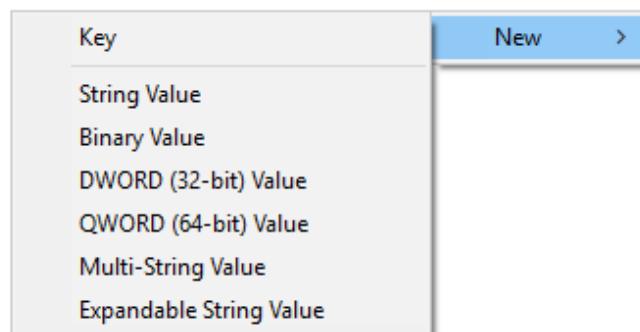
HKEY_CURRENT_CONFIG •

وهي عبارة عن اختصار ل **HKCU**، إلا أنها أكثر خصوصية. فالمعلومات الموجودة في **HKCC** هي نفسها المعلومات الموجودة في **HKCU**، إلا أن الأخيرة أشمل وهي الأساس. عند التعديل على المعلومات في إحدى الخلويتين فإن الخلية الأخرى ستتأثر.

ملاحظة

- تختصر الخلية الرئيسية عادة وفق أول حرف كلماتها، فال الخلية **.HKCR** هي **HKEY_CLASSES_ROOT**

يتكون الرجسستري من مفاتيح وقيم، يمكن تشبيه المفاتيح بالمجلدات، والقيم بالملفات. يوجد في الرجسستري الأنواع التالية من القيم:





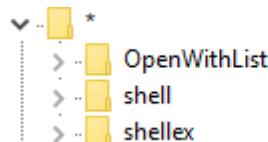
التعامل مع القيم والمفاتيح

يمكن التعامل مع الرجسستري من خلال محرر السجل regedit.exe أو من خلال موجه الأوامر cmd أو من خلال لغات البرمجة. وسنبدأ بمحرر السجل لنفهم المفاتيح والقيم والعلاقة بينها.. بحيث نتمكن لاحقاً من الرابط بين محرر السجل ولغة البرمجة لتسجيل برنامجنا ومعلوماته في الرجسستري.

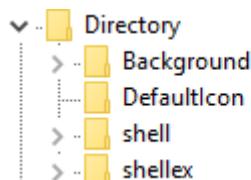
بالنسبة لمحرر التسجيل سنتعامل مع ملفات ذات امتداد reg، هذه الملفات ماهي إلا ملفات نصية تحوي مجموعة من الأوامر التي تخبر محرر السجل regedit بالقيم والمفاتيح التي عليه التعامل معها. ويمكن إنشاء هذه الملفات من خلال تصدير المفاتيح.

نظرة عامة

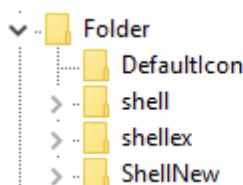
افتح محرر السجل ثم انتقل للخلية HKCR، سترى الكثير من المفاتيح، في رأس القائمة هناك مفتاح اسمه *، هذا المفتاح يمثل ما يجب على الكمبيوتر أن يفعله من أجل جميع أنواع الملفات. الصورة التالية توضح محتويات هذا المفتاح:



الآن انتقل إلى المفتاح Directory الموجود ضمن الخلية HKCR نفسها، هذا المفتاح يمثل ما يجب على الكمبيوتر أن يفعله من أجل المجلدات.
لاحظ:

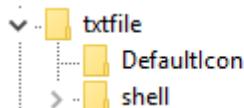


للمفتاح Folder صلة قرابة مع المفتاح Directory، وهذه محتوياته:

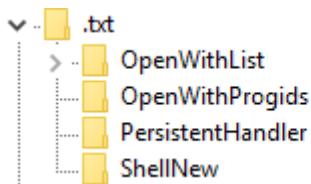




أما من أجل نوع محدد من الملفات، مثل الملفات النصية على سبيل المثال، فإنه يتم إنشاء مفتاحين هما txtfile و txtfile، والأمر مماثل بالنسبة لباقي أنواع الملفات، لاحظ إحدى القيم:



والقيمة الأخرى:



بعد هذه الجولة السريعة في محرر التسجيل سنقوم ببعض الأمور على القيم التي تصفحناها منذ قليل، لنفهم الرجسستري والمنطق الذي يعمل به، وبعدها بإذن الله سنتعامل معه من خلال C#.

ملاحظة

- كن حذرا عند تعاملك مع الرجسستري، أي خطأ من الممكن أن يكلفك الكثير، لذلك ينصح عادة بأخذ نسخة احتياطية قبل إجراء أي تعديل عليه.
- أنا لا أتحمل مسؤولية أي خطأ ناتج عن سوء استخدامك لمحرر التسجيل اتفقنا 😊؟

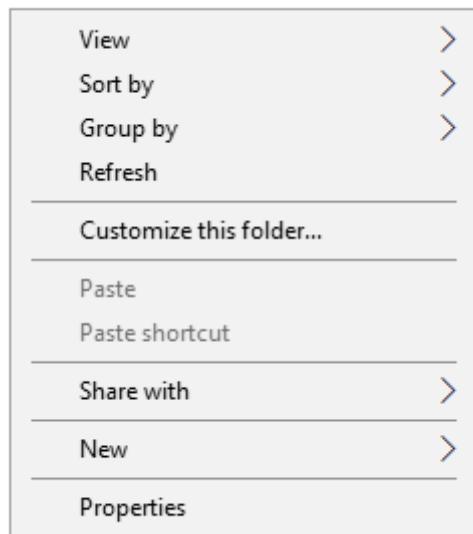
تطبيق 1 – نسخ الملف كمسار!

طالما عانى الكثيرون من معضلة مسارات الملفات، لديك ملف ما في مكان ما وتحتاج مساره، عليك الضغط على صندوق النص في أعلى متصفح الملفات ليتحدد مسار المجلد الحاوي على الملف المطلوب نسخ مساره، ثم تفتح المفكرة أو أي برنامج يستقبل النصوص وتضع إشارة "\\" في نهاية المسار، ثم اسم الملف ثم لاحقته، لتحصل على مسار الملف المطلوب بالآخر 😊

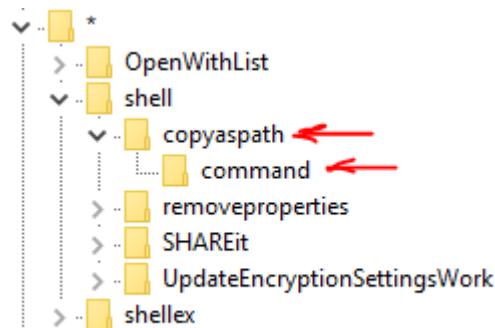
الآن تعال وفكّر معي، نحتاج لفتح القائمة المنبثقة التي تظهر عند الضغط على الملفات بالزر الأيمن، وبجوار الأوامر فتح وضغط وإرسال إلى وفتح باستخدام وبقية الأوامر الأخرى الشائعة نحتاج لوجود أمر يقوم بنسخ محتوى الملف المحدد، لا والأكثر من ذلك أن هذا الأمر له صورة واسم خاصان بك!



اذهب إلى الرجسستري وانتقل بعدها للمفتاح *، والذي اتفقنا أن الكمبيوتر يأتي إلى هنا عندما تتعامل مع أي نوع من الملفات. ثم انتقل للمفتاح shell والذي يمثل القائمة المنبثقة، وهي القائمة التي تظهر عند الضغط بالزر الأيمن.



أنشئ مفتاحاً باسم copyaspath ثم أنشئ بداخله مفتاحاً باسم command. ليصبح المفاتيح كما يلي:



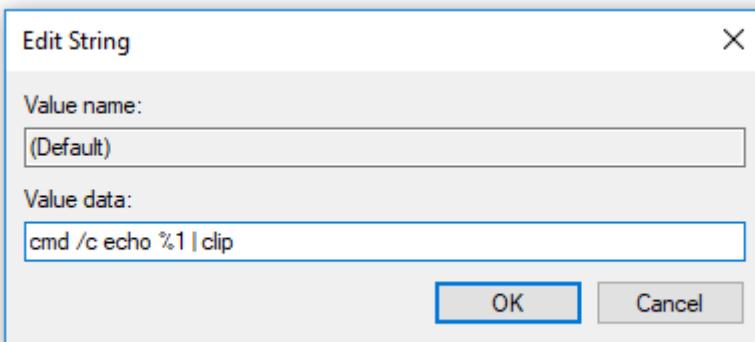
الآن عدل القيمة الافتراضية للمفتاح الأول كما هو موضح في الصورة أدناه¹، ثم قم بإضافة قيمة نصية وسمّها Icon واجعل قيمتها هو مسار أيقونة برنامجك أو شركتك إذا أردت ذلك، أو لا تضاف هذه القيمة إذا لم ترغب بوجود أيقونة بجانب الأمر.

Name	Type	Data
ab (Default)	REG_SZ	Copy as path - Eng27
Icon	REG_SZ	"E:\REG\icon.ico"

¹ على اعتبار أن هذه هي القيمة الافتراضية للمفتاح الرئيس، فستظهر للمستخدم..



ثم عدل القيمة الافتراضية للمفتاح الثاني واجعلها كما يلي:



الآن اذهب إلى أي ملف يخطر ببالك واضغط عليه بالزر الأيمن وتفاجأ بالنتيجة 😊، اضغط على الأمر الذي أضفناه ثم وبسرعة اذهب إلى أي برنامج يستقبل النصوص والصق بداخله محتوى الحافظة النصي وستجد مسار الملف. ولكن قبل ذلك، لا تنسَ أن تضع الأيقونة التي ستظهر بجانب الأمر في المسار الذي حددها في القيمة Icon! أو عدّل القيمة إلى المسار الذي وضعت به أيقونتك!!!

وكل شرح بسيط عما حدث، أضفنا مفتاحاً أسميناه وضبطنا اسمه وأيقونته، وجعلنا الأمر الخاص به هو تشغيل موجه الأوامر وكتابة المسار الذي تم تنفيذه منه، ثم نسخها للحافظة Clipboard.

ملاحظة

- إذا لم تضبط القيمة الافتراضية للمفتاح الرئيس، فإن اسمه سيظهر للمستخدم.
- من الممكن أن تكون قيمة أيقونة الأمر هي مسار ملف أيقونة أو مسار برنامج ما، حيث أن نظام التشغيل يأخذ أيقونة البرنامج.
- الأمر cmd هو في الواقع الافتراضية cmd.exe، وعلى اعتبار أنه من عائلة نظام التشغيل - أقصد أنه من البرامج الافتراضية والتي تجدها في مجلد System32 - فإذاً يمكنك كتابة اسمه دون لاحقة. وينطبق هذا على جميع ملحقات نظام التشغيل مثل cmd rung regedit (جرب افتح موجه الأوامر وكتب notepad.exe أو notepad، لافرق).

في موجه الأوامر echo يستخدم لطباعة عبارة نصية، وهو مناظر للأمر Console.WriteLine في C#. أما الخيار / ف يقوم بإيقاف موجه الأوامر عن العمل بعد تنفيذ الكود. وبالنسبة للمتغير¹ %1 فيحمل مسار الملف أو

¹ المتغيرات في بيئه DOS تحاط أو تسبق بإشارة %.



البرنامج أو المجلد والذي تم استدعاء موجه الأوامر من خلال قائمةه المنبثقة. والأمر Clip | يقوم بنسخ العبارة المطبوعة باستخدام echo.

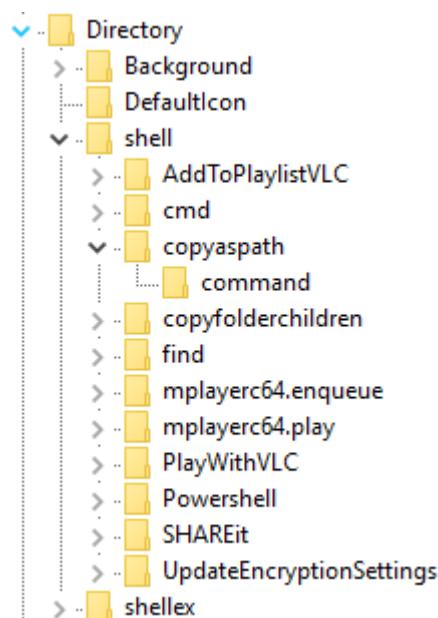
ملاحظة

- لطباعة مسار المجلد الأب echo %W
- لنسخ المسار مع تنسيص، ضع التنسيص في الأمر وذلك كما يلي: echo "%1" | clip

تطبيق 2 – نسخ المجلد كمسار

باتت الفكرة واضحة نوعاً ما ومكشوفة ولا داعي لإشارة تعجب في نهاية العنوان حتى 😊. في التطبيق السابق كنا نتعامل مع الملفات لذلك اتجهنا للمفتاح *، الآن سنتعامل مع المجلدات فما هو المفتاح الذي سنتعامل معه يا حذرك ؟؟؟🤔

انتقل للمفتاح Directory ثم shell، أنشئ مفتاحاً وسمه copyaspath، ثم أنشئ مفتاح بداخله باسم command.



أعد الخطوات التي قمنا بها في التطبيق الأول (القيمة الافتراضية، الأيقونة، الأمر).

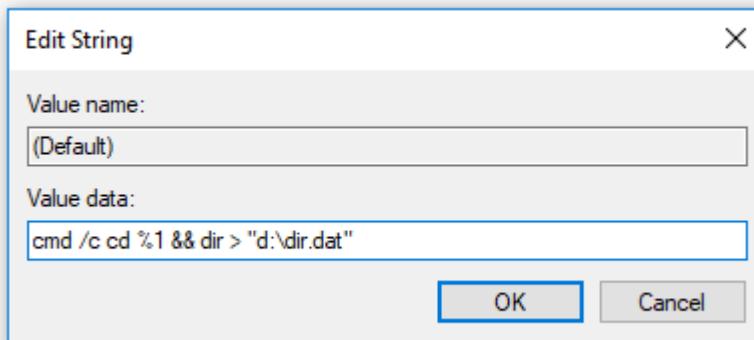
تطبيق 3 – نسخ تقرير عن محتويات المجلد

بنفس المبدأ وبالاعتماد على نظام الدوس المتمثل بموجه الأوامر يمكن معرفة محتويات مجلد ما بما فيها من ملفات ومجلدات، وبالإمكان نسخ



المحتوى إلى الحافظة باستخدام Clip | بعد تنفيذ الأمر أيضاً. ولكن ليس من المجد ولا من المستخدم ولا من العادة أن يرغب المستخدم بنسخ محتويات المجلدات ووضعها في الحافظة، بينما قد تحتاج لوضع هذه المحتويات في ملف ما بحيث يستفيد منها برنامجك، أو أنت لاحقاً.

أيا كانت غاياتك من محتوى المجلدات، قم بإضافة مفتاح باسم copyfolderchildren بجوار مفتاح التطبيق 2، ثم أجر عليه التغييرات اللازمة كما سبق (القيمة الافتراضية والأيقونة والأمر، على كل حال فالصورة السابقة فيها هذا الأمر وأنت لم تتبه له 😊، مابعرف وين عقلك 😂😂)، ثم استخدم الأمر:



حيث إن cd %1 هو أمر يقوم بنقل الدليل¹ الحالي في نظام дос إلى المسار %1 (وهذا المتغير كما تعلم يحوي مسار المجلد الذي تم تنفيذ الأمر منه، على اعتبار صار عندك خبرة)، أما الإشارة && فتفيد في تنفيذ أكثر من أمر في موجه الأوامر (ف cd أمر و dir أمر آخر، في حين أن العبارة echo %1 هي أمر واحد). وبالنسبة للأمر dir فهو يعرض محتويات الدليل الحالي، أما إشارة < والمسار الذي يليها فهو يعني أن ناتج تنفيذ الأمر لن يظهر على شاشة موجه الأوامر وإنما سيوضع في ملف ما، وفي حال قرأت الفقرات الأولى من الفصل السابق فإنك تعلم أن إشارة < تعني مسح الملف في حال وجوده وإنشاء آخر جديد، أما عندما تكون مزدوجة << فإنها تعني الكتابة إلى نهاية الملف في حال وجوده.

جرب الأمر على أحد المجلدات لديك ثم انتقل للمسار d:\dir.dat، افتح الملف dir.dat باستخدام المفكرة أو أي محرر نصوص، ثم لاحظ المحتويات والتي هي من الشكل:

¹ الدليل أو الفهرس هو مصطلح يعبر عن مسار المجلدات، إذا كنت في القرص D بداخل مجلد Games فهذا يعني أن فهرسك الحالي هو D:\Games



```
dir - Notepad
File Edit Format View Help
Volume in drive D is Files
Volume Serial Number is DEB1-DBA1

Directory of D:\GAMES

08/31/2018 11:20 AM <DIR> .
08/31/2018 11:20 AM <DIR> ..
08/17/2018 03:51 PM <DIR> Cheats
08/26/2018 07:25 PM 986 Cities XL 2012.lnk
1 File(s) 986 bytes
3 Dir(s) 235,948,896,256 bytes free
```

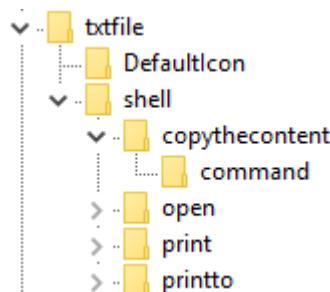
المجلدات تسبق بالرمز <DIR> أما الملفات فتسبق بحجمها بالبايتات.

تطبيق 4 – نسخ محتوى الملفات النصية!

قد لا تكون قد استوعبت الفكرة بعد أو تخيلت العملية ولا الحاجة من إشارة التعجب، تخيل المرور بجانب أحد الملفات النصية، وبدل فتح الملف النصي وتحديد كامل النص بداخله ثم نسخه وإغلاق الملف، بإمكانك إضافة أمر إلى القائمة المنبثقة ليقوم بنسخ محتوى أي ملف نصي تنفذه منه.

وعلى اعتبار أن العملية غير مجده مع غير الملفات النصية تقريباً، فلن نضيف هذا الأمر بالنسبة لجميع الملفات (أي بالمفتاح *)، وإنما سنضيفه للملفات النصية فقط (أي بالمفتاح txtfile). أعتقد الآن أنه بات لديك فكرة واضحة عن مبدأ عمل الرجسستري وكيف يتعامل معه نظام التشغيل.

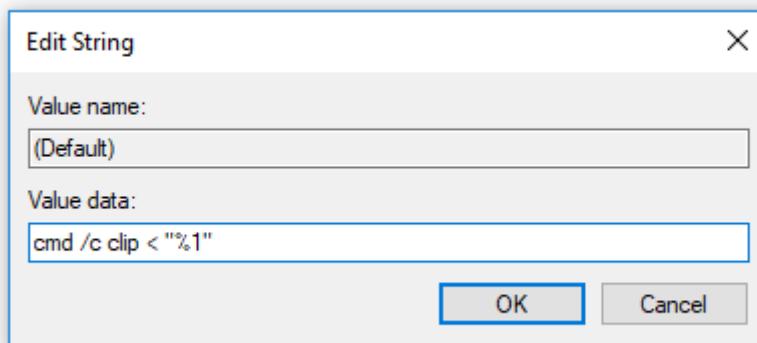
انتقل للمفتاح txtfile وأضف بداخله مفتاحاً باسم copythecontent، وأجرِ التغييرات والإضافات الالازمة كما سبق (لا داعي لأن أخبرك أن هذه الإضافات هي بين قوسين القيمة الافتراضية والأيقونة والأمر، فقد أصبحت خبيراً الآن ماشاء الله 🌟).





أعتقد أنه قد جذب انتباهك وجود المفتاح DefaultIcon، وحتى لو لم يجذب انتباهك فسأجذبه لك، في هذا المفتاح يوجد مسار أيقونة الملفات من نوع txtfile، لذلك ففي الأيام القادمة عندما تنشئ صيغة خاصة ببرامحك عليك إضافة هذا المفتاح لضبط أيقونات الملفات من هذا النوع، والكتاب سيناقش هذا الموضوع في الفقرات القادمة بإذن الله.

الآن لنعد لموضوعنا، أخذنا الحديث 😊، وصلنا للأمر الذي سنستخدمه في هذا التطبيق، لاحظ:



والأمر السابق يعني أنه على موجه الأوامر (cmd) وبعد أن يعلق نفسه (cmd /c) أن ينسخ (clip) محتوى المسار الذي تم تنفيذه منه ("%1"). حاول أنسنة جميع أعمالك حتى لو كانت برمجية بحثة.

لكن مشكلة نظام الدوس أنه لا يقبل المحارف العربية، لذلك فالمحظى العربي سيكون مشوّهاً عند نسخه لأنه سيمرر إلى شاشة موجه الأوامر تماماً كشاشة Console، بينما ستدعم المحارف الأجنبية بالكامل. ومن هنا أتت الحاجة لتطوير برنامج بلغة برمجة عالية المستوى لتعامل مع جميع أنواع المحارف. لذلك فالتطبيق التالي سيكون باستخدام C#.

تطبيق 5 – نسخ محتوى الملفات النصية 2

أنشئ مشروعًا جديداً في الفيجوال ستوديو (مشروع نوافذ، مع أنها لن تتعامل مع واجهة التطبيق، لكن بيئه Console لن تدعم المحارف العربية)، ثم اضبط خاصية Opacity للنافذة على 0، واستخدم الكود التالي:

```
string Path = Environment.GetCommandLineArgs()[1];
string content = File.ReadAllText(Path, Encoding.Default);
Clipboard.Clear(); Clipboard.SetText(content);
this.Close();
```

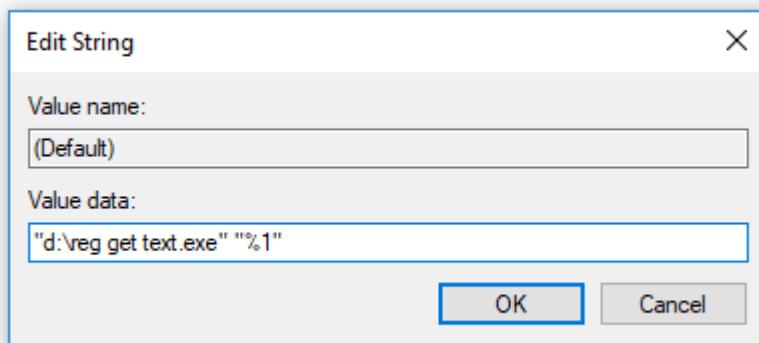


والكود رغم بساطته إلا أنه يعني الكثير، فالسطر الأول يصرّح عن متغير نصي سُيُسند إلى قيمة نصية هي مسار الملف الذي تم استدعاء برنامجنا من خلاله، وهي تكافئ 1% في نظام dos. أما السطر الثاني فيقوم بقراءة محتويات الملف الذي مساره Path بالترميز الافتراضي للنظام (وهذه أهم خطوة في كامل التطبيق هذا)، ثم يسند هذه المحتويات إلى متغير نصي. ثم يضع قيمة المتغير النصي الأخير في الذاكرة وذلك بعد حذف محتواها.. وأخيراً وعلى اعتبار أن النافذة لا تهمنا فيجب إغلاقها.

لا تنسِ إضافة System.IO إلى مراجع مشروعك!

كل ما سبق هو خطوة أولى، الآن نريد أن نربط برنامجنا بالأمر الموجود في القائمة المنبثقة، لأن الإجراء GetCommandLineArgs سيعيد مصفوفة نصية تحوي مسار البرنامج ومسار الملف الذي تم تنفيذ الأمر منه. الآن، كيف عرف نظام التشغيل أن عليه تنفيذ برنامجنا وبدوره برنامجنا وباستخدام هذا الأمر سيتعامل مع الملف الذي نفذه؟؟؟ ببساطة علينا الرجوع للرجسستري وبدل تنفيذ cmd سننفذ برنامجنا.

عدل كود التطبيق 4 واجعله كما يلي:



لكن لا تنسِ أن تضع البرنامج في القرص d مباشرةً أو أن تغير المسار في الكود ليتماشى مع موقع برنامجك! بإمكانك إخفاءه ليتواتي عن الأنظار، وبإمكانك جعله كملف نظام حتى لا يستطيع أحد إيجاده حتى لو فعل خيار إظهار الملفات والمجلدات المخفية، الفكرة هي كيف تستطيع أنت إظهاره إذا كان من غير الممكن لأي أحد إظهاره حتى ولو فعل خيار إظهار الملفات؟؟؟ الأيام ستعلّمك عزيزي.

طبعاً باستخدام فئات FileInfo مثل C# وغيرها بإمكانك تنفيذ التطبيقات الثلاثة الأولى دون استخدام cmd، ولكن أحببت أن أعطيك نظرة على



الجانب القديم من تفكير وعقلية الكمبيوتر، بالإضافة إلى تقليل البرامج التنفيذية المستخدمة للحصول على النتيجة المطلوبة. والتطبيق التالي سيمثل هذه الفكرة.

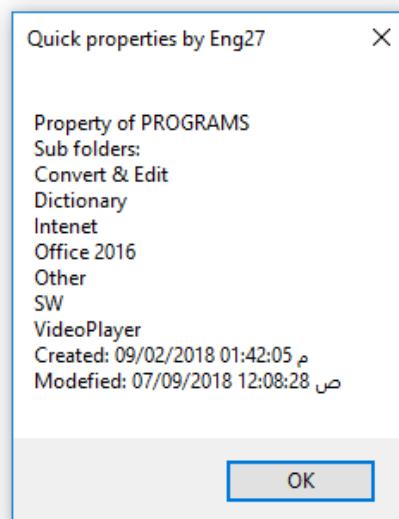
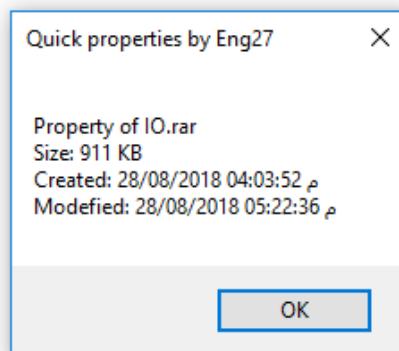
ملاحظة

- عند تنفيذ المشروع في الفيجوال ستوديو ستحصل على خطأ، لا مشكلة فقط أغلق المشروع وانسخ البرنامج التنفيذي من مشروعك إلى القرص D، واحرب التطبيق.

تطبيق 6 – خصائص الملفات باختصار

الفكرة من هذا التطبيق مشابهة لسابقه، إلا أنه سيقوم بالحصول على خصائص الملف وعرضها في صندوق رسالة بدلاً من نسخ محتويات الملف، كما أن هذا التطبيق يمكن تنفيذه على جميع أنواع الملفات وعلى المجلدات، لذلك فأتوقع أن ما يحول بخاطرك هو المفتاح * Directory .

التطبيق سيعطيك النتيجة التالية:





بنفس الأسلوب في التطبيق السابق، استخدم الكود:

```
string Path = Environment.GetCommandLineArgs()[1];
try
{
    FileInfo file = new FileInfo(Path);
    MessageBox.Show(string.Format("Property of {0}\nSize: {1} KB\nCreated:
{2}\nModefied: {3}",
        file.Name,
        file.Length / 1024,
        file.CreationTime, file.LastAccessTime), "Quick properties by Eng27");
}
catch
{
    DirectoryInfo folder = new DirectoryInfo(Path);
    string content = "";
    foreach (DirectoryInfo c in folder.GetDirectories())
        content += "\n" + c.Name ;
    if (content == "")
        content = "There is no folders inside!";
    MessageBox.Show(string.Format("Property of {0}\nSub folders:
{1}\nCreated: {2}\nModefied: {3}",
        folder.Name,
        content ,
        folder.CreationTime, folder.LastAccessTime),
        "Quick properties by Eng27");
}
this.Close();
```

يقوم الكود بمحاولة أخذ تفاصيل المسار كملف، فإذا ظهر خطأ فهذا يعني أن المسار هو مجلد.

وفي الرجسستري، أنشئ نفس المفاتيح بالأسماء الازمة واستخدم الكود:

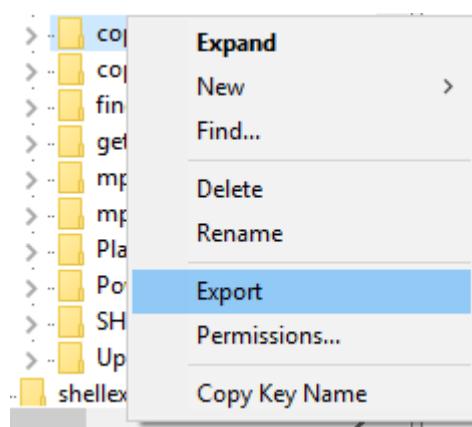




نشر المفاتيح، تصديرها واستيرادها

عندما تصمم برامج كبيرة تحتاج للتعامل مع الرجسبري، وهذه البرامج قمت بنشرها لتصبح على كمبيوترات المستخدمين، فإنك ستحتاج لطريقة لإضافة مفاتيح لكمبيوتر المستخدم عن بعد. ويمكن ذلك من خلال تصدر المفاتيح لتصبح ملفات تسجيل reg، بحيث يقوم المستخدم بفتحها والموافقة على التسجيل.

انتقل للمفتاح المطلوب نشره، اضغط عليه باليمن، ثم تصدر:



احفظ المفتاح في مكان ما ثم اذهب إليه، واضغط باليمن ثم تحرير لتحصل على كود يمثل مفاتيحة وقيمه:

```
Folders CopyAsPath - Notepad
File Edit Format View Help
Windows Registry Editor Version 5.00

[HKEY_CLASSES_ROOT\Directory\shell\copyaspath]
@="Copy as path - Eng27"
"Icon"="\"E:\\REG\\icon.ico\""

[HKEY_CLASSES_ROOT\Directory\shell\copyaspath\command]
@="cmd /c echo %1 | clip"
```

قم بتصدير جميع أوامرك واجمعها في ملف تسجيل واحد، كما في الصورة التالية:



```

Folders CopyAsPath - Notepad
File Edit Format View Help
Windows Registry Editor Version 5.00

[HKEY_CLASSES_ROOT\Directory\shell\copyaspath]
@="Copy as path - Eng27"
"Icon"="\"E:\\REG\\icon.ico\""

[HKEY_CLASSES_ROOT\Directory\shell\copyaspath\command]
@="cmd /c echo %1 | clip"

[HKEY_CLASSES_ROOT\Directory\shell\copyfolderchildren]
@="Copy folder children - Eng27"
"Icon"="\"e:\\reg\\icon.ico\""

[HKEY_CLASSES_ROOT\Directory\shell\copyfolderchildren\command]
@="cmd /c cd %1 && dir > \"d:\\dir.dat\""

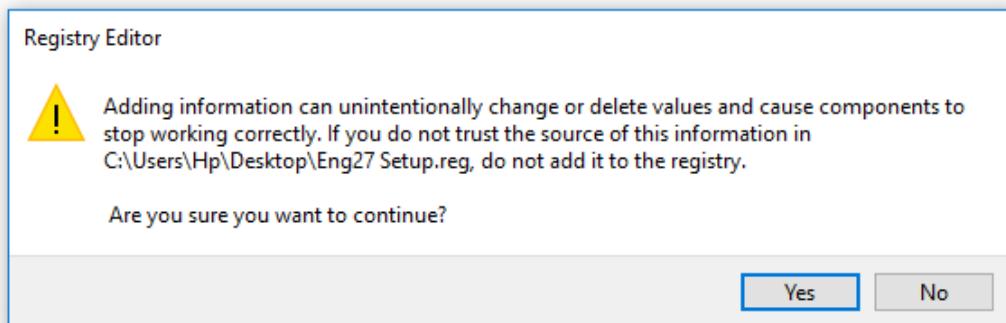
[HKEY_CLASSES_ROOT\Directory\shell\getproperties]
@="Quick properties by Eng27"
"Icon"="\"e:\\reg\\icon.ico\""

[HKEY_CLASSES_ROOT\Directory\shell\getproperties\command]
@="\"D:\\Reg get property.exe\" \"%1\""

```

امسح المفاتيح التي قمت بتصديرها ثم اذهب وافتح القائمة المنبثقة على أحد المجلدات لتأكد أن الأوامر غير موجودة.

الآن قم بتشغيل ملف التسجيل، ووافق عليه:



ادهّب لأحد المجلدات وحرّب القائمة المنبثقة عليه.. ثم قم بتسمية ملف التسجيل باسم مثل Eng27Setup، وهو جاهز للنشر والتوزيع الآن.

كرر الخطوة بالنسبة للملفات، وضعها مع كود المجلدات في ذات الملف لتصبح ملفاً واحداً.

لكن هناك موضوع لا تنساه، يجب على كل من الأيقونة والبرامج المستخدمة في الأوامر أن تتوارد في المسار المحدد في الأوامر



المستخدمه. ولأخفف عليك، سأعطيك كود ملف دفعي يقوم عنك وعن المستخدم بنسخ جميع الملفات المطلوبة إلى أماكنها المخصصة.

افتح مستند نصي جديد واتكتب داخله مايلي:

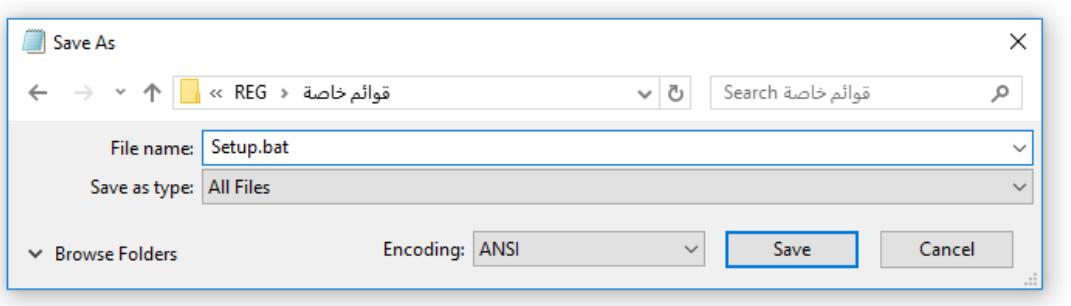
```

Untitled - Notepad
File Edit Format View Help
@echo off
xcopy /y /h *.exe d:\
md e:\REG
xcopy /y *.ico e:\REG
pause

```

يقوم الأمر الأول بعدم إظهار المسارات الحالية، والثاني يقوم بنسخ جميع الملفات التنفيذية exe حتى لو كانت مخفية أو ملفات نظام، مع عدم انتظار تأكيد النسخ من المستخدم، وذلك إلى القرص d. أما الثالث فيقوم بإنشاء مجلد جديد باسم REG في القرص e، ثم ينسخ جميع الأيقونات إليه.

احفظ الملف بصيغة bat باسم Setup وغير نوع الملفات إلى جميع الملفات:



ضع جميع الملفات بجانب بعضهم في مجلد واحد واضغطهم معًا:

Eng27 Setup	٢٠١٨/٠٩/١٤ م ١٢:٣٣	Registration Entries	3 KB
icon	٢٠١٨/٠٢/١٤ م ٠١:٥٥	ICO File	17 KB
README	٢٠١٨/٠٩/١٤ م ٠١:٤٤	Text Document	1 KB
Reg get property	٢٠١٨/٠٩/١٤ ص ١١:٤٩	Application	26 KB
Reg get text	٢٠١٨/٠٩/١٤ ص ١٠:٧٧	Application	8 KB
Setup	٢٠١٨/٠٩/١٤ م ٠١:٥١	Windows Batch File	1 KB
uninstall	٢٠١٨/٠٩/١٤ م ٠١:٤٨	Registration Entries	3 KB
قوائم خاصة	٢٠١٨/٠٩/١٤ م ٠١:٥٠	WinRAR archive	15 KB



مسح المفاتيح

يمكن مسح المفاتيح عن طريق ملفات التسجيل باستبدال إشارة @ بالإشارة - بالإضافة لوضعها في مسار المفتاح، لاحظ كود ملف uninstall الموجود في الفقرة السابقة:

```
uninstall - Notepad
File Edit Format View Help
Windows Registry Editor Version 5.00
[-HKEY_CLASSES_ROOT\Directory\shell\copyaspath]
-= "Copy as path - Eng27"
[-HKEY_CLASSES_ROOT\Directory\shell\copyaspath\command]
-= "cmd /c echo %1 | clip"
[-HKEY_CLASSES_ROOT\Directory\shell\copyfolderchildren]
-= "Copy folder children - Eng27"
[-HKEY_CLASSES_ROOT\Directory\shell\copyfolderchildren\command]
-= "cmd /c cd %1 && dir > \d:\dir.dat"
[-HKEY_CLASSES_ROOT\Directory\shell\getproperties]
-= "Quick properties by Eng27"
```

التسجيل الصامت!

يقصد بالتسجيل الصامت أو التنصيب الصامت بتسجيل المفاتيح برمجياً، دون الحاجة لطلب إذن المستخدم في تشغيل ملف التسجيل والموافقة على التسجيل..

بإمكانك إنشاء ملف دفعي يقوم بهذه المهمة، لاحظ الكود:

```
regedit /s "e:\Key.reg"
```

الرجسٌتري من خلال C#

أصبح لديك الآن كم جيد من المعلومات والأفكار عن الرجسٌتري وكيفية التعامل معه والروتينات التي تجري من خلاله عند كل ضغطة زر أو تشغيل برنامج أو حدث ما يمر على الكمبيوتر، لذلك سنتنقل لكيفية إضافة المفاتيح والقيم، التعديل عليها، إسناد قيم لها، وحتى حذفها برمجياً.

في بداياتك في البرمجة قيل لك - أو قال لك هذا الكتاب - أنه لا يكفي إنشاء البرامج للإدخال والإخراج وإنما يجب عليك استخدام مفهوم



المتغيرات، أن تجعل للبرنامج ذاكرة تخزن فيها ما يجول بخاطره ويقرر تصرفاته بناءً عليها. ثم قيل لك أنه لا يكفي وجود المتغيرات في برنامجك وإنما عليك اختيار قيمها في ملفات خارجية على القرص الصلب، وقراءة محتوياتها عند الحاجة، فالمتغيرات تفني عند الخروج من البرنامج!

قد تكون البيانات المختزنة ضمن المتغيرات حساسة أو هامة ولا يجب النفوذ إليها أو التعديل عليها، أو حتى حذفها، وقد لا تنفع وسائل التشفير والقفل، عندها بإمكانك استخدام الرجسستري لتخزين قيم متغيراتك بحيث لا أحد يعلم مكانها سواك..

القيم والمفاتيح برمجيًا

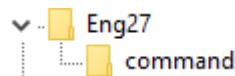
بإمكانك القيام بكل مasic من خلال C#, لا حاجة لفتح محرر التسجيل بالإضافة القيم والمفاتيح، بإمكانك التصريح عن كائنات من نوع مفاتيح الرجسستري لتحكمك بالمفاتيح بإمكانيات كثيرة.

للتعامل مع فئة الرجسستري عليك إضافة مجال أسماء Microsoft.Win32 إلى مشروعك أولاً. لإنشاء مفتاح ما استخدم الفئة Registry وفق الطريقة :CreateSubKey

```
Registry.CurrentUser.CreateSubKey("Software\\Eng27");
```

وللحكم بالمفتاح عليك إنشاء كائن من نوع RegistryKey وفتح المفتاح من خلال الطريقة OpenSubKey ثم إسناد المفتاح إلى الكائن:

```
RegistryKey key = default(RegistryKey);
key = Registry.CurrentUser.OpenSubKey("Software\\Eng27",
    RegistryKeyPermissionCheck.ReadWriteSubTree,
    System.Security.AccessControl.RegistryRights.FullControl);
key.CreateSubKey("command");
```



الآن أصبح الكائن key هو عبارة عن مفتاح وبإمكانك القيام من خلاله على كل العمليات الممكنة على المفاتيح والقيم.



لاحظ بعض الخصائص التي قد تلزmk:

```
int val = key.ValueCount; // عدد القيمة
int subkey = key.SubKeyCount; // عدد المفاتيح الفرعية
string keypath = key.Name; // مسار المفتاح
```

أما الطرق التي قد تتعامل معها:

```
مسار المفتاح // key.ToString()
ضبط بيانات القيمة // key.SetValue("Val", "Data");
قيمة ما // key.SetValue(null, "DefaultData!");
القيمة الافتراضية // key.SetValue(null, "DefaultData!");
فتح مفتاح فرعي // key.OpenSubKey("command");
الحصول على بيانات قيمة ما // string value = key.GetValue("Val").ToString();
```

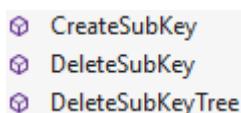
لاحظ ناتج تنفيذ الكود السابق فيما لو تم فتح أحد المفاتيح وإسناده لكاين:

(Default)	REG_SZ	DefaultData!
Val	REG_SZ	Data

ملاحظة

- للتعامل مع القيمة الافتراضية للمفتاح، نضع اسم القيمة null.

أما لحذف وإنشاء المفاتيح والقيم، فالإجراءات التالية هي وجهتك:



ولإغلاق المفتاح:

```
key.Close();
```

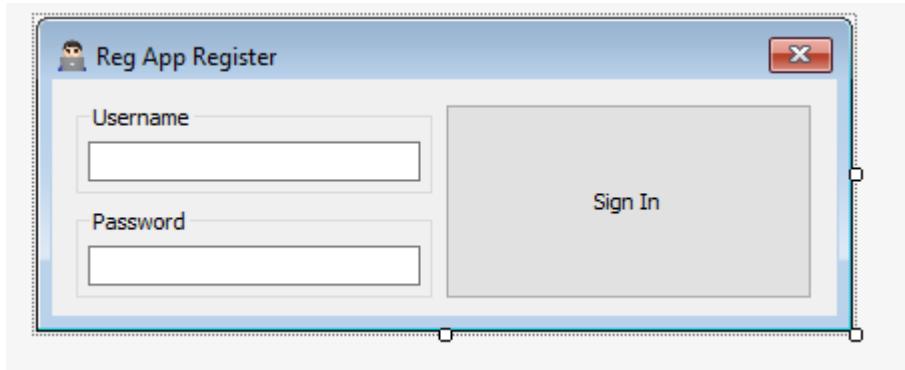
تطبيق 7 – التحقق من المستخدم!

من الأساليب المتّبعة في البرامج عند تعاملها مع الرجسٻتري هو حفظ مسار البرنامج ومسارات الملفات المستخدمة مؤخراً ومواعيد وعدد مرات تشغيل البرنامج وبعض البيانات الخاصة بالمستخدم وغيرها من التطبيقات البسيطة كمبدأ، والتي قد لا تخطر على بال المستخدم. وفي هذا

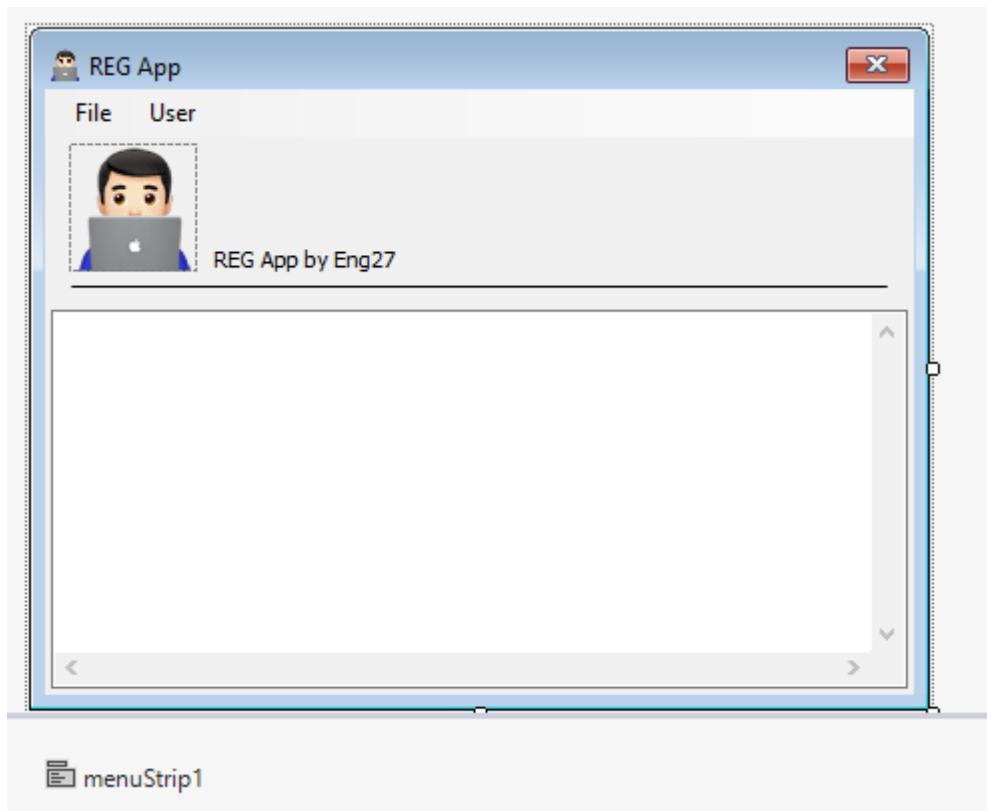


التطبيق سنسخدم هذا المبدأ في حفظ بيانات البرنامج، مثل اسم المستخدم وكلمة سره، وعدد مرات زيارته للبرنامج وتاريخ كل زيارة.

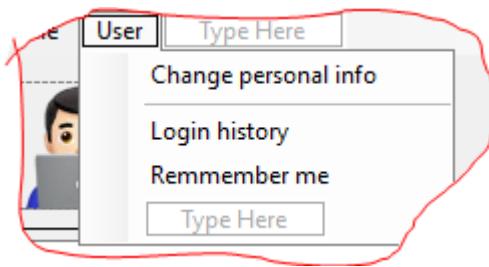
افتح الفيجوال ستوديو وأنشئ مشروعًا جديًّا، صمم الواجهة التالية بالأدوات الموضحة أمامك:



ثم الواجهة:

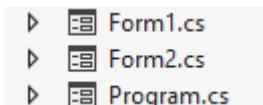


قم بإضافة قائمتين رئيسيتين File و User. وفي القائمة User أضف قوائم فرعية وفق الصورة التالية:



وسماها برمجيا PersonalM و HistoryM و RemmemberM، ليسهل التعامل معها. وفعل خاصية CheckOnClick لقائمة الأخيرة.

أصبح لديك في مستعرض المشروع مايلي:



افتح كود Program.cs ثم أضف إليه الفئة Forms المستنسخة من Form والتي سنضع فيها جميع المتغيرات التي سنستخدمها في مشروعنا. (توضع هذه الفئة بجانب الفئة Program، أي داخل مجال الأسماء الخاص بمشروعنا).

```
public class Forms : Form
{
    //Declare your global valubles here.
    public static RegistryKey key, visitkey;
    public static bool olduser;
    public static string username, password, currentuser, currentpass;
    public static int visits;
    public static DateTime registdate, visitdate;
}
```

الفئة Forms ترث من الفئة Form، وبدورهما Form1 و Form2 ترثان من Forms، لذلك في بداية كود كل من Form1 و Form2، غير الوراثة لتصبح بالشكل:

```
public partial class Form1 : Forms
```

```
public partial class Form2 : Forms
```

والخطوة هذه ضرورية لتضمن أن كل من Form1 و Form2 ستتمكن من الوصول للمتغيرات التي جعلناها في Forms. لاتنسى إدراج الفئة Microsoft.Win32 إلى بداية أكواد النوافذ..



كود النافذة الأولى:

```

public Form1()
{
    InitializeComponent();
}

private void Form1_Load(object sender, EventArgs e)
{
    التحقق من وجود مستخدم في الكمبيوتر // إذا لم يكن هناك مستخدم
    if (!olduser)
        أنشئ واحداً من خلال إجراء // وإن كان هناك مستخدم
    else
        سجل الدخول مباشرة اذا تم ضبط خيار تذكر الحساب // currentuser = key.GetValue("Username").ToString();
        currentpass = key.GetValue("Password").ToString();
        visits = int.Parse(key.GetValue("Visits").ToString());
        visits += 1;
        key.SetValue("Visits", visits.ToString());
        visitdate = DateTime.Today;
        visitkey.SetValue(visits.ToString(), visitdate.ToString());
        showForm();
    }
}

private bool.isUser()
{
    bool result;
    تم إنشاء المتغيرين النصيين التاليين للتنظيم فقط //
    يمكنك الاستغناء عنهما بتعويض قيمتهما بدلاً عنهما //
    string key1 = "Software\RegApp";
    string key2 = "Software\RegApp\VisitsDates";
    key = Registry.CurrentUser.OpenSubKey(key1, true);
    visitkey = Registry.CurrentUser.OpenSubKey(key2, true);
    try
    {
        إذا كان هناك قيمة بالرجسستري باسم البرنامج فهذا يعني أنه يوجد مستخدم //
        وبالتالي يمكن تعويض مسار مفتاح برنامجنا في متغير نصي //
        وهي خطوة تجريبية لمعرفة هل يوجد مفتاح أم لا //
        string temp = key.Name;
        result = true;
    }
    catch
    {
        إذا لم يكن هناك مفتاح سيحدث خطأ، وسنعالجه بهذه الحلقة..
        result = false;
    }
    return result;
}

```



```
}
private void newUser() // إجراء يقوم بتهيئة البرنامج لإنشاء حساب جديد
{
    button1.Text = "Sign Up";
    key = Registry.CurrentUser.CreateSubKey("Software\\RegApp");
    MessageBox.Show("New user?\nEnter a name and a password",
        "NEW USER!",
        MessageBoxButtons.OK,
        MessageBoxIcon.Information);
}
private void textBox1_TextChanged(object sender, EventArgs e)
{
    buttonCheck();
}
private void textBox2_TextChanged(object sender, EventArgs e)
{
    buttonCheck();
}

إجراء يقوم بتمكين زر الادخال أو إلغاءه //()
{
    if (textBox1.Text != "" && textBox2.Text != "")
        button1.Enabled = true;
    else
        button1.Enabled = false;
}
private void button1_Click(object sender, EventArgs e)
{
    إذا كان المستخدم موجوداً //(
    {
        // عند تسجيل الدخول إذا كانت كلمة السر واسم المستخدم صحيحين
        currentuser = key.GetValue("Username").ToString();
        currentpass = key.GetValue("Password").ToString();
        visits = int.Parse(key.GetValue("Visits").ToString());
        visitdate = DateTime.Today;
        string D = "RegistDate";
        registdate = DateTime.Parse(key.GetValue(D).ToString());
        if (currentpass == textBox2.Text && currentuser.ToLower() ==
            textBox1.Text.ToLower())
        {
            visits += 1;
            key.SetValue("Visits", visits.ToString());
            visitkey.SetValue(visits.ToString(), visitdate.ToString());
            showForm(); // إظهار النافذة الخاصة بالبرنامج من خلال إجراء
        }
    }
}
```



```

        MessageBox.Show("Username or Password is incorrect!",
            "Error on login",
            MessageBoxButtons.OK,
            MessageBoxIcon.Error);
        textBox2.SelectionStart = 0;
        textBox2.SelectionLength = textBox2.TextLength;
    }
}
// وإذا كان المستخدم جديداً
{
    // عندها يجب إنشاء مستخدم جديد بالبيانات المدخلة في صناديق النصوص
    currentuser = textBox1.Text;
    currentpass = textBox2.Text;
    key.SetValue("RegistDate", DateTime.Now.Date.ToString());
    key.SetValue("Username", currentuser);
    key.SetValue("Password", currentpass);
    MessageBox.Show("Registered for " + textBox1.Text,
        "Registered successfully!",
        MessageBoxButtons.OK,
        MessageBoxIcon.Information);
    // هذه أول زبارة للمستخدم ل البرنامج
    visits = 1;
    key.SetValue("Visits", visits.ToString());
    key.SetValue("RememberME", 0);
    visitdate = DateTime.Today;
    visitkey = key.CreateSubKey("VisitsDates");
    visitkey.SetValue(visits.ToString(), visitdate.ToString());
    // إظهار النافذة الخاصة بالبرنامج من خلال إجراء
}
}

// إجراء يقوم بإظهار النافذة الثانية
{
    Form2 form = new Form2();
    form.Show();
    this.Hide();
}

private void textBox2_KeyDown(object sender, KeyEventArgs e)
{
    // لتسهيل البرنامج، عند الضغط على مفتاح إنتر عند كتابة كلمة السر
    // يتم تنفيذ ما يحدث عند الضغط على الزر
    // لكن انتبه! يجب أن يكون الزر مفعلاً لتأكد أن المستخدم أدخل جميع البيانات
    if (e.KeyValue.ToString() == "13")
        if (button1.Enabled)
            button1_Click(sender, e);
}

```



كود النافذة الثانية:

```

public Form2()
{
    InitializeComponent();
}
private void Form2_Load(object sender, EventArgs e)
{
    this.Text = "REG App - " + currentuser;
    if (key.GetValue("RemmemberMe").ToString() == "1")
        RemmemberM.Checked = true;
    else
        RemmemberM.Checked = false;
    AddMenus();
}
private void Form2_FormClosed(object sender, FormClosedEventArgs e)
{
    if (e.CloseReason == CloseReason.UserClosing)
        Application.Exit(); //()
}
private void AddMenus()
{
    //إضافة قوائم تمثل تاريخ آخر عشر زيارات للبرنامج
    //يمكن إظهار أكثر أو أقل من ذلك ويمكن ترك إمكانية تحديدها للمستخدم
    int i = visitkey.ValueCount;
    while (HistoryM.DropDown.Items.Count < 10 && i > 0)
    {
        HistoryM.DropDown.Items.Add(i + " - " +
visitkey.GetValue(i.ToString()).ToString());
        --i;
    }
    if (HistoryM.DropDown.Items.Count < 1)
        HistoryM.Enabled = false; //()
}
private void RemmemberM_Click(object sender, EventArgs e)
{
    //حفظ اختيار المستخدم فيما إذا أراد طلب كلمة السر عند تسجيل الدخول أم لا
    if (RemmemberM.Checked)
        key.SetValue("RemmemberMe", 1);
    else
        key.SetValue("RemmemberMe", 0);
}

```



بعد استخدام البرنامج عدة مرات ستحصل على القيم التالية:

Name	Type	Data
ab (Default)	REG_SZ	(value not set)
ab Password	REG_SZ	1111
ab RegistDate	REG_SZ	16/09/2018 12:00:00
ab RememberME	REG_DWORD	0x00000001 (1)
ab Username	REG_SZ	Eng27
ab Visits	REG_SZ	39

كما أنه لدينا مجموعة من القيم داخل المفتاح .VisitsDates

تطبيق 8 – فكرة، نسخة محدودة..

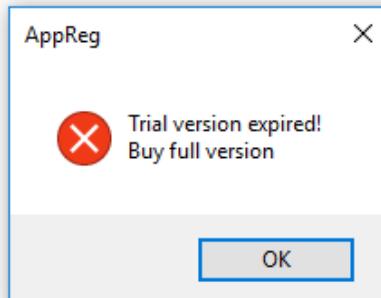
من الأفكار التي يمكن تطبيقها على المفتاح الخاص ببرنامجك في الرجسٻري هو تحديد عدد المرات التي يمكن للمستخدم استخدام البرنامج خلالها، وسنطور التطبيق 7 ليصبح مشتملاً على ذلك.

لدينا في البرنامج عدد رقمي visits يحوي عدد مرات تشغيل البرنامج، والذي سيقارن مع قيمة افتراضية..

الفكرة بسيطة ضع الكود التالي مع الأكواد الموجودة في :Form2_Load

```
if (visits > 5) عدد المرات الافتراضي هو 5، ضعه وفق رغبتك // 
{
    MessageBox.Show("Trial version expired!\nBuy full version",
        "AppReg",
        MessageBoxButtons.OK,
        MessageBoxIcon.Stop);
    Application.Exit();
}
```

وعند تخطي الفترة المسموحة سيحصل المستخدم على:



توقع بعد فترة أن تجد في الأسواق كراكًا لتطبيقك بقيامك بهذه الحركة

صيغة ملفات خاصة بك!

من المحتمل أنك لم تفهم معنى عنوان الفقرة، الموضوع كما قرأته "صيغة ملفات خاصة بك!".. أي أنه على غرار exe و txt و mp3 وغيرها من الصيغ بإمكانك إنشاء واحدة. ولهذه الصيغة على غرار هؤلاء أيقونة تمثل البرنامج الذي ستُفتح باستخدامه، ووصف خاص بها، وبرنامج افتراضي يفتحها بطبيعة الحال.. كما أن ملفات هذه الصيغة ستصبح من الملفات المسلمة بها بالنسبة لنظام التشغيل.

على اعتبار أننا نتعامل مع صيغ الملفات ومشغلاتها فإننا سنتنشر المفاتيح في HKCR. ويلزمنا مفاتيح رئيسين.

قبل أن نبدأ، انتقل إلى المفاتيح txt. وقارن txtfile في الرجسٰري وقارن محتواهُما.

أهم ما يميز المفتاح txt. هو وجود أن قيمته الافتراضية هي txtfile، وهذا يعني أن المشغل الافتراضي لهذا النوع من الملفات معرف في المفتاح txtfile. إذن المفتاح الذي يكون من الشكل "نقطة ثم صيغة" يمثل الصيغة بحد ذاتها، بمعنى أن الكمبيوتر عندما يفتح هذا النوع من الملفات يأتي إلى هنا، ويتجه للمفتاح المخزن في القيمة الافتراضية. جرب المفاتيح الخاصة بالصيغة exe. وmp3. وdla. مثلاً، لاحظ القيمة الافتراضية لكل منها.

 (Default)

REG_SZ

txtfile

أما بالنسبة لـ txtfile، ففيه يتم تخزين معلومات عن مشغل هذا النوع من الملفات. وقيمة الافتراضية تحدد وصف الملف، وداخل هذا المفتاح يتم وضع الايقونة الافتراضية في مفتاح DefaultIcon، بالإضافة إلى الأوامر التي تظهر للمستخدم في القائمة المنبثقة، مثل فتح أو تحرير أو أوامر خاصة مثل التطبيقات الأولى ضمن هذا الفصل وغيرها من الأوامر.

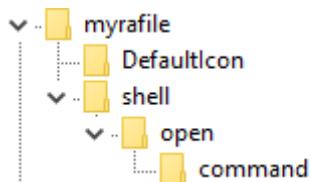
تطبيق 9 – ربط برنامجك بصيغة خاصة به

سنطور التطبيق 7 ليتمكن من تحرير وحفظ الملفات ذات صيغة ما ليحتوي على الأوامر الكلاسيكية في برامج تحرير الملفات.

لإنشاء صيغة خاصة بالتطبيق 7، ولتكن myra اختصاراً لـ MyRegApp لذلك قم بإنشاء مفتاح باسم myra.HKCR. ضمن القيمة الافتراضية له باسم المفتاح الذي سيمثل مشغل الملفات وهو myrafile، ثم أنشئ المفتاح الأخير.



أضف للمفتاح myrafile كل من المفاتيح التالية:



"RegApp Files" ثم اضبط القيمة الافتراضية للمفتاح الرئيس على القيمة "RegApp Files" وهو الوصف الذي سيظهر على ملفات هذا النوع.

واضبط القيمة الافتراضية للأيقونة الافتراضية على مسار الأيقونة التي ستظهر على الملفات أو مسار البرنامج.

أما بالنسبة لـ command فكما سبق ورأينا فإن قيمته هي مسار البرنامج التنفيذي ملحقاً بالمتغير `%1` وفي حالي `"e:\reg\regapp.exe %1"`.

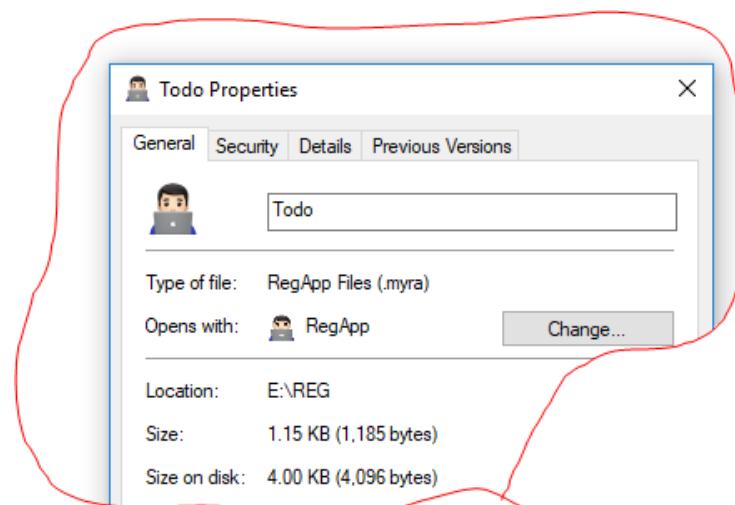
كما أنه بضبط القيمة الافتراضية لـ shell على اسم أحد مفاتيحيها الفرعية سيجعله المفتاح الافتراضي، لذلك اضبطها على القيمة open.

أما الأمر open فاضبطه كما يلي:

Name	Type	Data
ab (Default)	REG_SZ	
ab Icon	REG_SZ	"E:\REG\icon.ico"

بعد الانتهاء من ضبط مشغل الصيغة قد يتطلب إعادة تشغيل الكمبيوتر لتطبيق التغييرات، وهذا ما يفضل أن يفعله المستخدم بعد تسجيل الصيغة الخاصة بملفاتك.

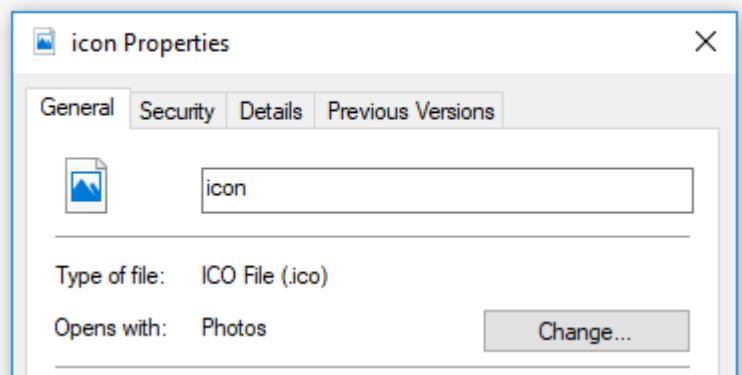
والخطوات السابقة تتيجتها ما يلي:





بفتح صندوق خصائص ملف ما من الملفات ذات الصيغة myra ستحمل أيقونة البرنامج المشغل لها أينما ذهبت، ولن تُكتب لاحقتها أمام اسمها لأنها من الملفات المعروفة لدى النظام. كما أن مشغلها الافتراضي هو Opens with وهذا واضح أمام العبارة RegApp.

لاحظ الفرق بين صيغتنا وصيغة موجودة في نظام التشغيل مثل icofile (طبعاً صيغتنا أحلى):



بقي لدينا فكرة أساسية وهي ربط برنامجنا بالصيغة الجديدة، عندما نفتح أي ملف يحمل الصيغة myra سيتم فتح البرنامج RegApp لأننا وضعنا مساره في المفتاح open، لكنه سيفتح وهو فارغ تماماً.

فكرة معي، عندما تفتح مستند وورد جديد، أو برنامج الوورد من موقعه الأصلي فإنه ستظهر لك نافذة بيضاء لا غبار عليها، صحيح؟؟ طيب عندما نفتح مستندًا قدِّيماً يحوي بيانات، كيف عرف نظام التشغيل أن هناك بيانات، وحتى كيف عرف مسارها؟؟؟

في الواقع فإننا تطرقنا إلى هذا الموضوع دون التنويه له وذلك لأن الأفكار والمعلومات فيما يخص التعامل مع الرجسٰتري كثيرة، وعندما تناولنا التطبيق 5 و 6 فإننا تعاملنا مع الملفات التي قامت بتنفيذ أمر التشغيل، ولكن بشكل سطحي جدًا..

الفكرة كلها في التابع Environment.GetCommandLineArgs()، وهوتابع يعيد مصفوفة نصية، مكوناتها مسار البرنامج التنفيذي ومسار الملف الذينفذ العملية.. والمساران مخزنان على الترتيب بالدليل 0 و 1.

أي أنه للحصول على متغير نصي وليس مصفوفة نصية تحوي المسار، يكفي كتابة [1]Environment.GetCommandLineArgs() مثلاً للحصول على مسار الملف الذي أرسل أمر تنفيذ البرنامج. وبحصولنا على مسار



الملف الذي نفذ البرنامج بإمكاننا التعامل معه كيما نشاء، بدءاً بالحصول على تفاصيله كما رأينا في التطبيق 5 و6، وانتهاءً بالتعديل والتحكم به.

الآن عد للتطبيق 7 وصرح متغيراً نصياً باسم path في المكان الذي صرحت فيه عن المتغيرات (ضمن الفئة Form : Form2 تذكرها؟).

ثم أضف الكود التالي إلى أكواد Form1_Load، (لا داعي لأن أخبرك أن هذا الإجراء موجود ضمن النافذة الأولى صحيح؟ أصبحنا كباراً الآن وذوي خبرة ماشاء الله):

```
try
{
    path = Environment.GetCommandLineArgs()[1];
}
catch
{
}
```

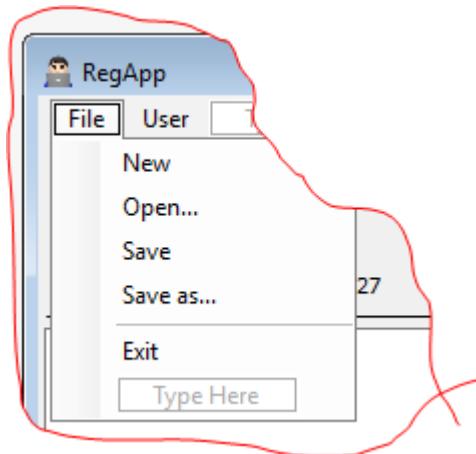
وبسبب احتمالية الأخطاء فإن حلقة try – catch لها نصيب من البرنامج.

ثم أضف الكود التالي إلى أكواد Form2_Load :

```
try
{
    string content = File.ReadAllText(path, Encoding.Default);
    textBox1.Text = content;
}
catch
{
}
```

ولا تنسى إضافة System.IO ها.

أضف مجموعة من القوائم الفرعية للقائمة File، وهي موضحة بالصورة التالية:



ثم غير الأسماء البرمجية لهذه القوائم لتصبح NewM وOpenM وهكذا.. ثم أضف الأكواد للقوائم كما يلي:

```
private void NewM_Click(object sender, EventArgs e)
{
    if (path != null)
    {
        string content = File.ReadAllText(path, Encoding.Default);
        if (textBox1.Text != string.Empty && textBox1.Text != content)
        {
            DialogResult result;
            result = MessageBox.Show("Do you want to save changes to\n" +
path,
            "Save changes?",
            MessageBoxButtons.YesNoCancel,
            MessageBoxIcon.Question);
            if (result == DialogResult.Yes)
            {
                SaveM_Click(sender, e);
                Application.Exit();
                Process.Start(Environment.GetCommandLineArgs()[0]);
            }
            else if (result == DialogResult.No)
            {
                Application.Exit();
                Process.Start(Environment.GetCommandLineArgs()[0]);
            }
        }
    }
}
```



```

else if (path == null)
{
    if (textBox1.Text != string.Empty)
    {
        DialogResult result;
        result = MessageBox.Show("Do you want to save changes to
untitled file?",
        "Save changes?",
        MessageBoxButtons.YesNoCancel,
        MessageBoxIcon.Question);
        if (result == DialogResult.Yes)
        {
            SaveAsM_Click(sender, e);
            Application.Restart();
        }
        else if (result == DialogResult.No)
        {
            Application.Restart();
        }
    }
    else
    {
        Application.Restart();
    }
}
}

```

```

private void OpenM_Click(object sender, EventArgs e)
{
    OpenFileDialog open = new OpenFileDialog();
    open.Filter = "RegApp File | *.myra";
    if (open.ShowDialog() == DialogResult.OK)
    {
        string content = File.ReadAllText(path, Encoding.Default);
        if (textBox1.Text != content)
        {
            DialogResult result;
            result = MessageBox.Show("Do you want to save changes to\n"
+ path,
            "Save changes?",
            MessageBoxButtons.YesNoCancel,
            MessageBoxIcon.Question);
            if (result == DialogResult.Yes)
            {

```



```

        SaveM_Click(sender, e);
    }
}
path = open.FileName;
textBox1.Text=File.ReadAllText (path, Encoding.Default);
}
}

```

```

private void SaveM_Click(object sender, EventArgs e)
{
    if (path == null)
    {
        SaveAsM_Click(sender, e);
    }
    else if (path != null)
    {
        File.WriteAllText(path, textBox1.Text, Encoding.Default );
    }
}

```

```

private void SaveAsM_Click(object sender, EventArgs e)
{
    SaveFileDialog save = new SaveFileDialog();
    save.Filter = "RegApp File | *.myra";
    if( save.ShowDialog() == DialogResult.OK)
    { path = save.FileName;
        File.WriteAllText(path, textBox1.Text, Encoding.Default ); } }

```

```

private void ExitM_Click(object sender, EventArgs e)
{
    Form2_FormClosed(sender,
        new FormClosedEventArgs(CloseReason.UserClosing));
}

```

ثم غير حدث إغلاق الفورم إلى:

```

private void Form2_FormClosed(object sender, FormClosedEventArgs e)
{
    if (e.CloseReason == CloseReason.UserClosing)
    {
        if (path == null)
        {
            if (textBox1.Text != string.Empty)

```



```
{  
    DialogResult result;  
    result = MessageBox.Show("Do you want to save changes to  
untitled file?",  
        "Save changes?",  
        MessageBoxButtons.YesNoCancel,  
        MessageBoxIcon.Question);  
    if (result == DialogResult.Yes)  
    {  
        SaveAsM_Click(sender, e);  
    }  
}  
}  
else if (path != null)  
{  
    string content = File.ReadAllText(path, Encoding.Default);  
    if (textBox1.Text != content )  
    {  
        DialogResult result;  
        result = MessageBox.Show("Do you want to save changes to\n"  
+ path,  
        "Save changes?",  
        MessageBoxButtons.YesNoCancel,  
        MessageBoxIcon.Question);  
        if (result == DialogResult.Yes)  
        { SaveM_Click(sender, e); }  
    }  
}  
Application.Exit();  
}  
}
```

أصبحت النافذة الثانية مكونة من الأكواد التالية:



```

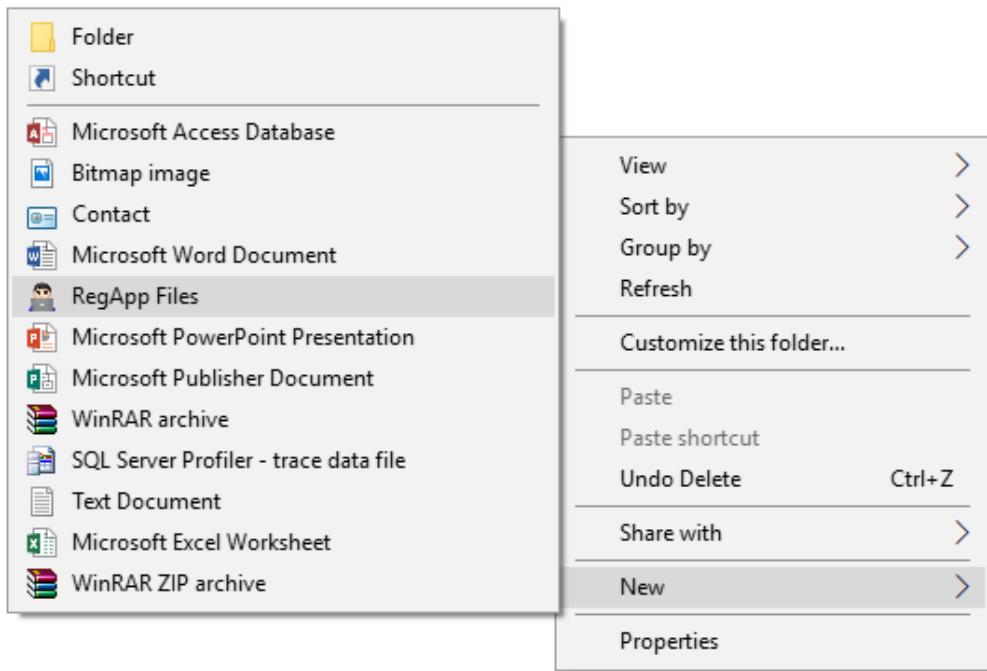
namespace RegApp
{
    public partial class Form2 : Forms
    {
        public Form2() ...
        private void Form2_Load(object sender, EventArgs e) ...

        private void Form2_FormClosed(object sender, FormClosedEventArgs e) ...
        private void AddMenus() ...

        private void RemmemberM_Click(object sender, EventArgs e) ...
        private void NewM_Click(object sender, EventArgs e) ...
        private void SaveM_Click(object sender, EventArgs e) ...
        private void SaveAsM_Click(object sender, EventArgs e) ...
        private void OpenM_Click(object sender, EventArgs e) ...
        private void ExitM_Click(object sender, EventArgs e) ...
    }
}

```

وإعطاء برنامجنا نفوذاً أكبر في ويندوز سنضيفه للقائمة "جديد" والتي تظهر عند الضغط على الزر الأيمن، لظهور القائمة التالية:



ولذلك انتقل للمفتاح `myra`. ثم أضف إليه مفتاحاً فرعياً باسم أي المفتاح "جديد" في القائمة المنبثقة، ثم أضف إليه قيمة باسم `NullFile` واتركها فارغة:

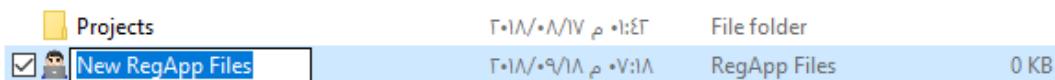


كما يمكنك إضافة البرنامج لقائمة إرسالة إلى، وذلك بوضع اختصار له في المسار التالي:

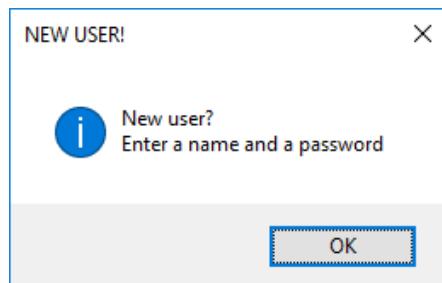
C:\Users\Hp\AppData\Roaming\Microsoft\Windows\SendTo

حيث Hp هو اسم المستخدم.

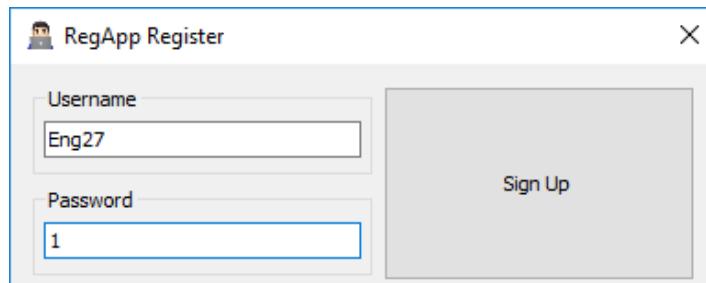
وهذه بعض الصور عن التطبيق عند استخدامه، إنشاء ملف من نوع الصيغة الخاصة بنا (من إنتاج شركتنا):



سم الملف على ذوقك، ثم افتحه:

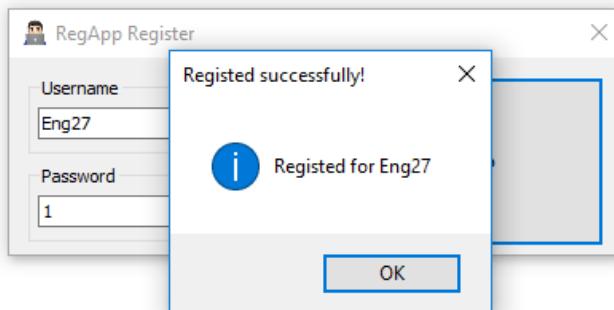


على اعتبار هذه أول مرة نفتح بها البرنامج (على أساس)¹، فإنه سيتم إنشاء مستخدم جديد:

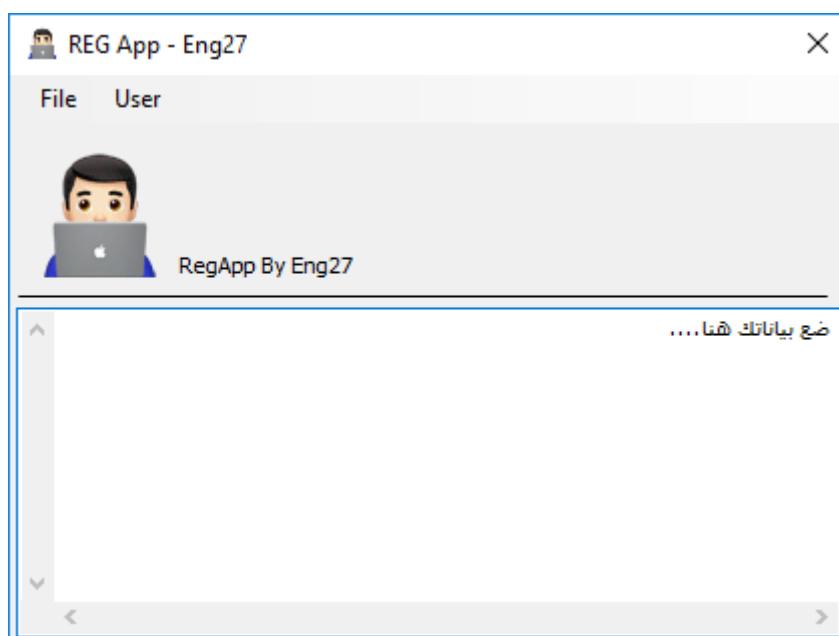


سمه بالاسم الذي ترغب به وضع له كلمة سر، ثم اضغط تسجيل.
ستحصل على رسالة تأكيد:

¹ على أساس: بفرض ذلك



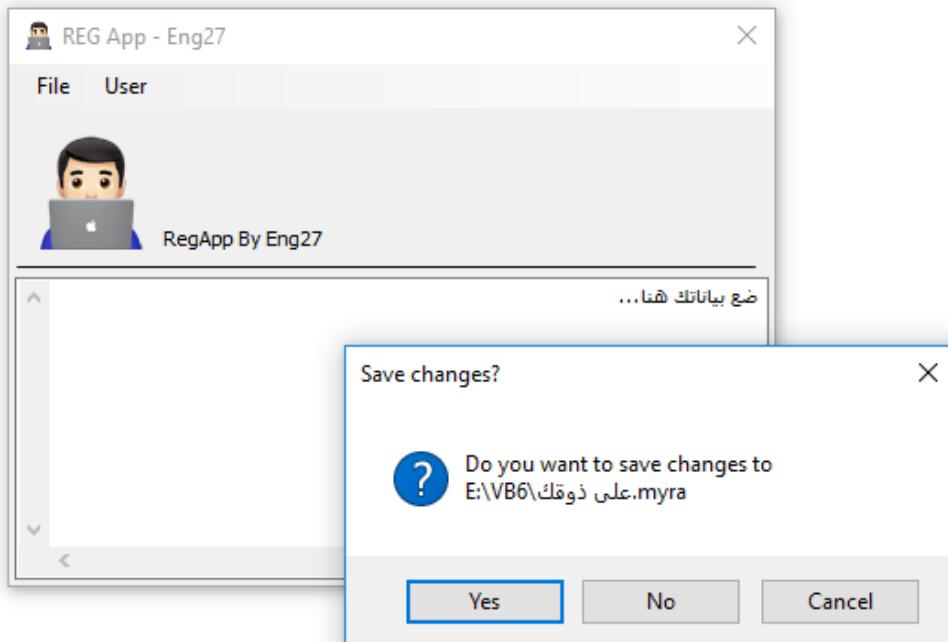
سيظهر البرنامج بواجهته الرئيسية..



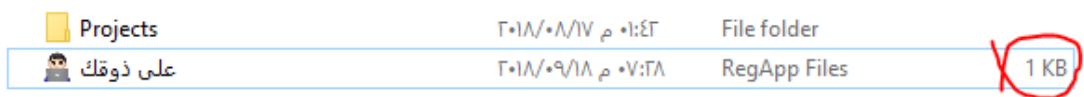
ملاحظة

- لمحاذاة النص من اليمين إلى اليسار اضغط RCtrl+RShift معا. وهو نفسه من أجل محررات النصوص العادية.

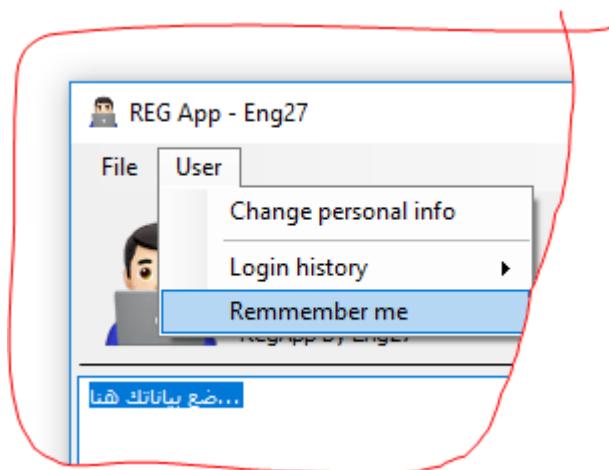
احفظ الملف أو اخرج من البرنامج، ستحصل على رسالة تفيد بأن الملف غير محفوظ إذا خرجم دون حفظ البيانات..



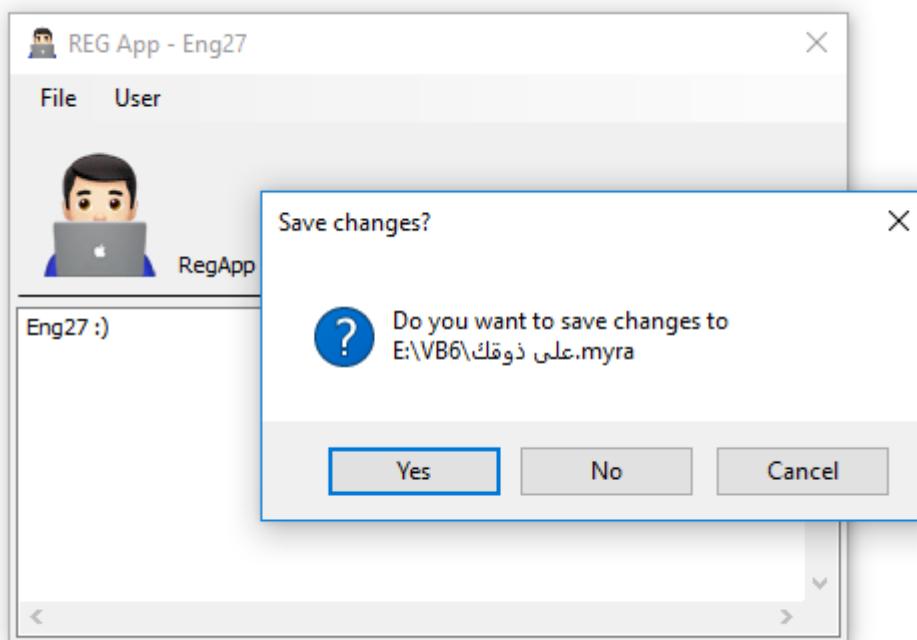
الآن الملف صار به بيانات:



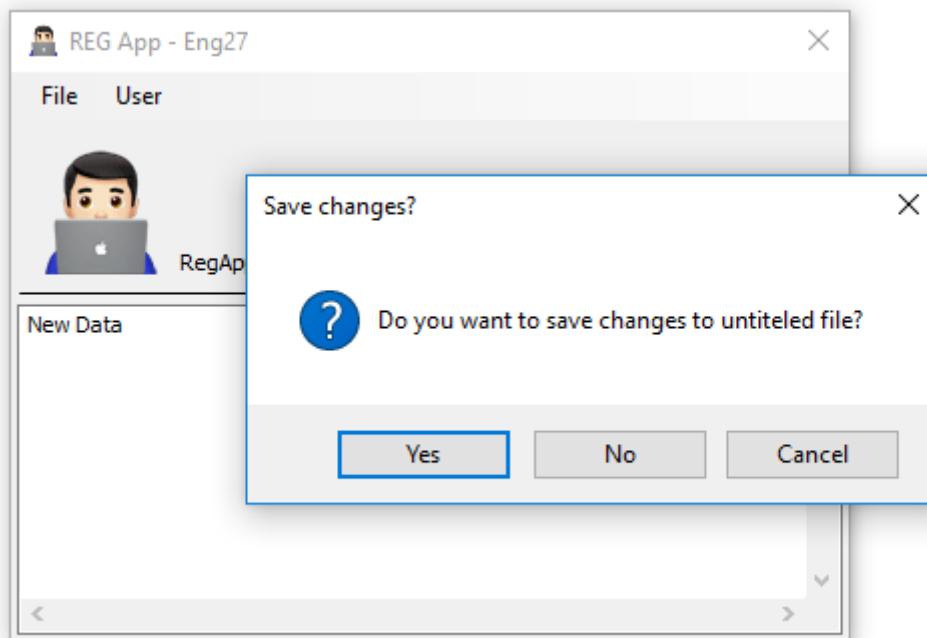
افتحه لنطمأن على بياناتنا، وفعل خيار Remember me إذا لم ترغب بعرض واجهة التحقق من المستخدم:



عدل على الملف ثم اختر جديد، لتحصل على رسالة تسألك فيما إذا أردت حفظ التغييرات أم لا:



في الملف الجديد أضف بعض البيانات ثم اضغط فتح، لتحصل على رسالة تطلب منك حفظ التغييرات إلى ملف غير محفوظ:



تهانينا ..!





الفصل الثاني عشر – قواعد بيانات SQL

تلعب قواعد البيانات دوراً هاماً وحيوياً في جميع البرامج التنظيمية والتجارية والإدارية، والتي تحوي عدداً كبيراً من البيانات ذات نوع محدد، وهذه البيانات موزعة وفق صفات وأعمدة لتشكل جداول، فتجمع البيانات وفق جداول أسهل بكثير من تجميعها وفق قيم منفصلة. تمثل الأسطر أو الصفات بياناتٍ شيء واحد لمختلف الصفات، بينما الأعمدة تمثل بياناتٍ أشياء كثيرة مشتركة بصفة واحدة.

تشابه الجداول المصفوفات إلى حد كبير، فقراءة وتنظيم وإدارة قيم المصفوفات أسهل بكثير من القيم المنفصلة والتي تنتمي لنوع واحد. ولهذا فإن لقواعد البيانات أهمية كبيرة في أي لغة برمجة.

خذ مثلاً بيانات عيادة ما. لديك مجموعة من المرضى، لكل مريض اسم وعمر، وحالة مرضية، وهل هو ذكر أم أنثى.. إلخ من البيانات. كما أنه لديك مجموعة من الموظفين في العيادة، لكل موظف اسم وعمر، ووظيفة، ومستوى تعليمي، وغيرها من البيانات الشخصية.

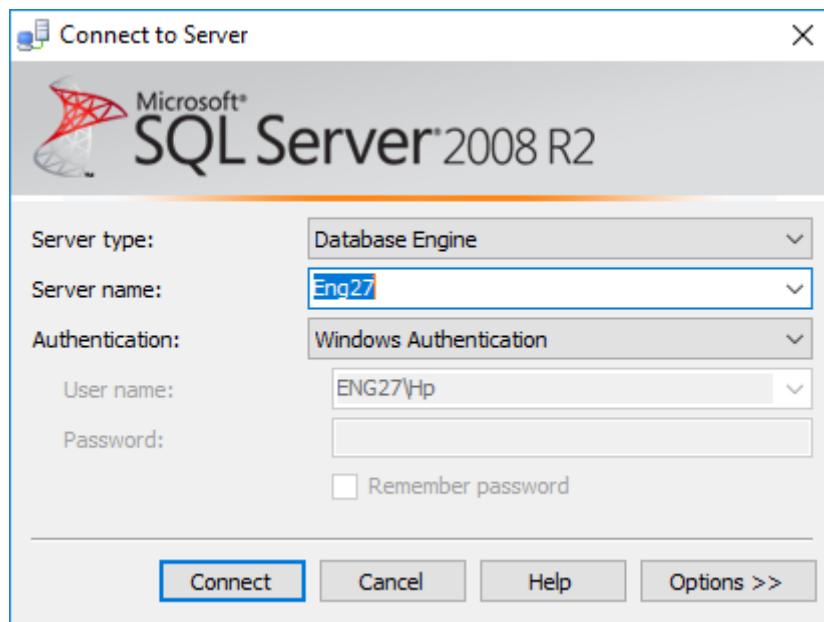
إن تنظيم هذه البيانات لكل مريض بمفرده أو لكل موظف بمفرده يجعل من الصعب إجراء الإحصائيات ومقارنة البيانات وتحليلها.. تخيل إجراء مقارنة بين مئات الملفات – الالكترونية أو الورقية – لمعرفة عدد المرضى الذكور مثلاً، أما لو كانت البيانات مرتبة وفق جداول معينة، وبالقاء نظرة على قيم عمود ما بإمكانك الحصول على الكثير من التفاصيل. عند حاجتك للحصول على ملف شامل لأحد المرضى أو الموظفين، بإمكانك جمع بيانات سطر واحد – يمثل هذا الموظف أو المريض – ووضعها في ملف منفصل.



إنشاء قاعدة بيانات

لإنشاء قاعدة بيانات فإنك تحتاج برنامجاً لإدارة قواعد البيانات. هناك الكثير من هذه البرامج مثل MySQL و Oracle وغيرها، كما أن بعض لغات البرمجة تمكنك من التعامل مع قواعد البيانات، ففي الفيجوال ستوديو بإمكانك إضافة مشروع قاعدة بيانات، وبإمكانك إضافة كائن قاعدة بيانات في C#، كما أن الفيجوال بيسبك تعطيك إمكانية إدارة قاعدة بيانات من أصل بيئة التطوير الخاصة بها. سنتعامل مع برنامج من إنتاج شركة Microsoft، وهو SQL Server Management Studio 2008 R2، وله إصدارات كثيرة وبإمكانك العمل بأي نسخة تريده تماماً كالفيجوال ستوديو. من الإصدارات المتوفرة إصدار 2005 و 2008 و 2012 و 2014 و 2017.

بعد تشغيل برنامج SQL2008 ستظهر لك نافذة الاتصال بالسرفر، وتتعلق مدخلاتها بالإعدادات التي قمت بضبطها أثناء تنصيب البرنامج.

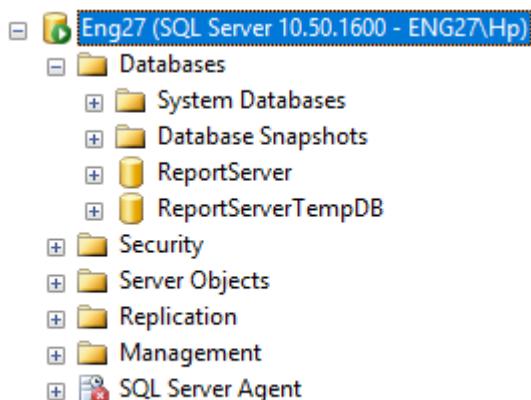
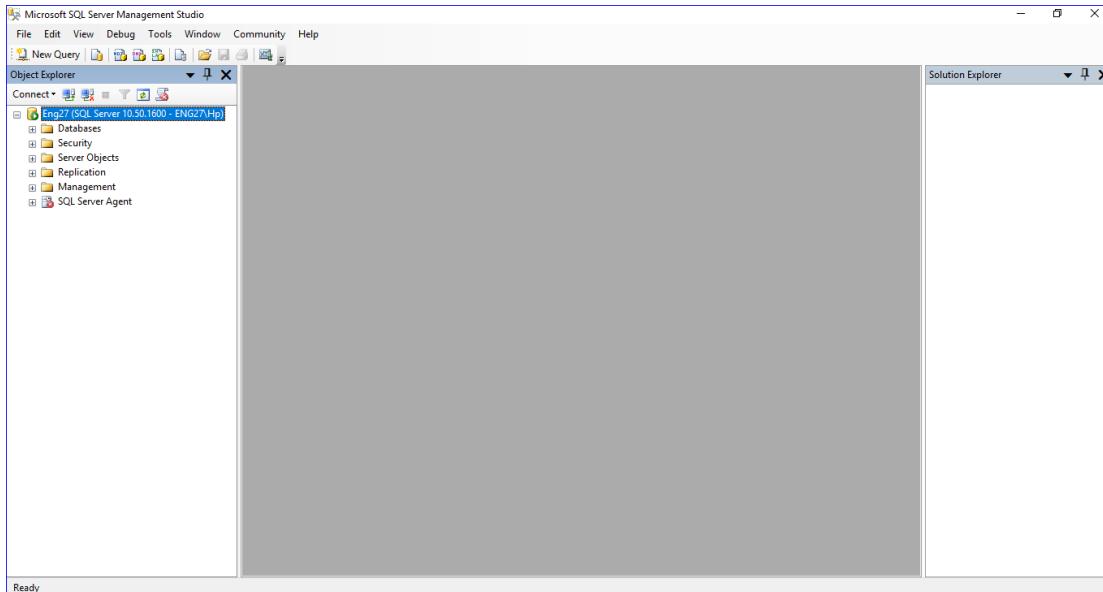


ويعتبر تنصيب برنامج إدارة قواعد البيانات حساساً للغاية وعليك اتباع الإرشادات بدقة، لأن أي خطأ أثناء ضبط إعدادات برنامج التثبيت قد يؤدي إلى مشاكل عديدة منها عدم إمكانية الاتصال بالسرفر أو أن البرنامج غير مثبت بشكل صحيح.

ومن المشاكل التي تظهر أيضاً وجود نسخة سابقة من برنامج إدارة قواعد البيانات حتى لو لم يكن موجوداً، وسبب هذه المشكلة هو وجود برنامج الفيجوال ستوديو والذي يتعامل مع قواعد البيانات، لذلك قم بحذفه وثبت برنامج SQL ثم ثبت الفيجوال ستوديو من جديد.

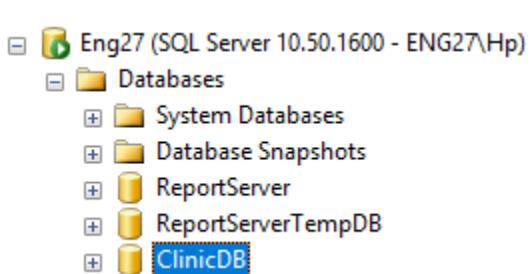
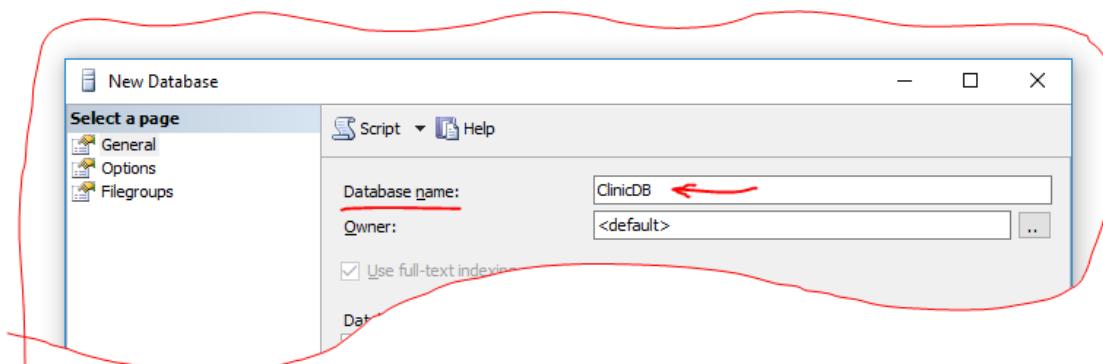


بعد الاتصال بال서فر ستحصل على النافذة التالية:



ما يهمنا منها هو العقدة **Databases** من شجرة العرض. والتي بفتحها ستحصل على:

بالنقر على **Databases** باليمين ثم **New Database**، نحصل على نافذة إضافة قاعدة بيانات جديدة. نختار اسمًا ما ثم موافق، لتضاف إلى الشجرة:



ولإنشاء جدول افتح فروع العقدة الخاصة بقاعدة البيانات التي أنشأتها ثم انقر على **Tables** باليمين ثم **New Table**:



ENG27.ClinicDB - dbo.Table_1		
Column Name	Data Type	Allow Nulls
		<input type="checkbox"/>

ومحتويات الجدول هي ثلاثة:

اسم العمود (أو الصنف)، نوع بيانته، وهل من الممكن أن يُترك فارغاً أم لا.

ثم قم بتعريف أعمدة الجدول الأول والممثل لبيانات مرضى العيادة كما يلي:

ENG27.ClinicDB - dbo.Table_1*		
Column Name	Data Type	Allow Nulls
id	int	<input type="checkbox"/>
Name	nvarchar(50)	<input type="checkbox"/>
Age	int	<input checked="" type="checkbox"/>
Sick	nvarchar(50)	<input checked="" type="checkbox"/>
Gender	nvarchar(50)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

ملاحظة

- لتسطيع التعامل مع قاعدة البيانات بالأحرف العربية نضبط نوع البيانات على nvarchar بدلاً من text.

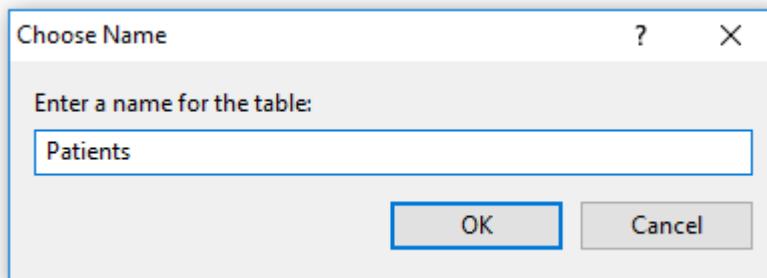
اجعل العمود الأول primary، وذلك حتى لا يمكن تكراره في بيانات الجدول، فمثلاً يمكن لأكثر من مريض أن يحمل ذات الاسم أو العمر أو المرض أو الجنس، لكن لا يمكن لأكثر من مريض أن يحمل ذات الرقم الشخصي، وهذا الرقم يميز مريضاً عن غيره، مثل الرقم الوطني في البطاقات الشخصية..



ولجعل العمود primary انقر باليمين على الحقل الصغير الموجود بجانبه ثم Set Primary Key، أو حدد العمود ثم من شريط الأدوات اختر الأمر كما في الشكل:



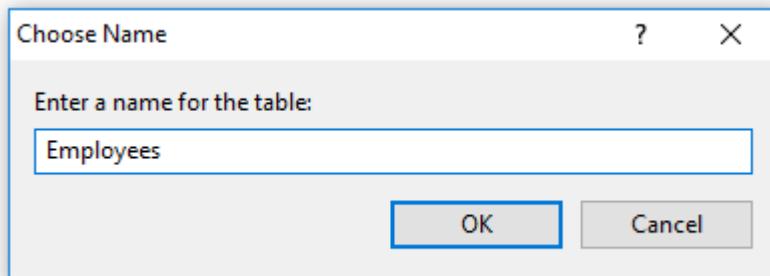
احفظ الجدول باسم ما وليكن :Patients



كرر الخطوة السابقة لإنشاء جدول الموظفين:

Column Name	Data Type	Allow Nulls
id	int	<input type="checkbox"/>
Name	nvarchar(50)	<input type="checkbox"/>
Age	int	<input checked="" type="checkbox"/>
Job	nvarchar(50)	<input checked="" type="checkbox"/>
Study	nvarchar(50)	<input checked="" type="checkbox"/>
Gender	nvarchar(50)	<input checked="" type="checkbox"/>
▶		<input type="checkbox"/>

اجعل العمود الأول primary ثم احفظ الجدول باسم Employees



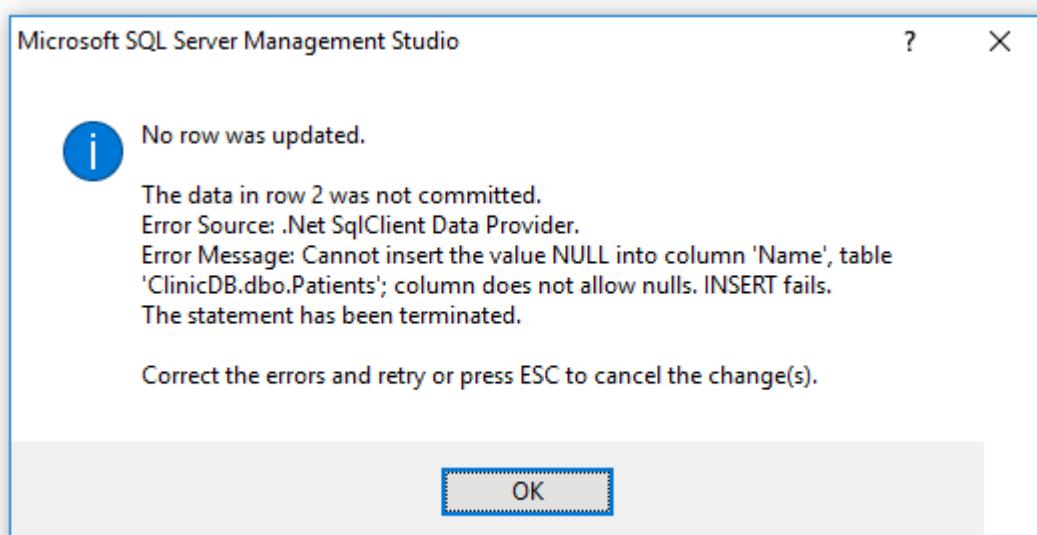


بعض أوامر قواعد البيانات

بإمكانك التعامل مع بياناتك من خلال بيئة SQL Server، وذلك بإنشاء استعلام Query جديد. وستلاحظ لاحقاً أن الأوامر التي سنقوم بها في C# هي نفسها التي سنقوم بها هنا في SQL Server. قم بإضافة بعض البيانات إلى قاعدة بياناتك، وذلك عبر الضغط باليمين على الجدول المطلوب من شجرة العرض، ثم Edit Top 200 Rows:

ENG27.ClinicDB - dbo.Patients					
	id	Name	Age	Sick	Gender
	1	أحمد	25	أسنان	ذكر
	2	خالد	18	أدن	ذكر
	3	محمد	33	جلدية	ذكر
	4	أسماء	31	أسنان	أنثى
►*	NULL	NULL	NULL	NULL	NULL

لا يمكن لمريضين أن يحملوا ذات الرقم id، وإلا فستحصل على الرسالة التالية:



أنشئ Query جديد عبر الأمر ، وتأكد أنك تنفذ الأوامر على قاعدة البيانات التي أنشأتها:





عرض البيانات select

- لعرض بيانات جدول ما كاملاً:

```
select * from Patients
```

- لعرض بيانات عمود ما ول يكن :Name

```
select name from Patients
```

- لعرض بيانات أكثر من عمود:

```
select name, age, gender from Patients
```

- لعرض البيانات وتغيير أسماء الأعمدة:

```
select name as [اسم المريض], age as [العمر] from Patients
```

ملاحظة

- يجب وضع إشارات [] عند العبارات التي تحوي فراغات، أو إحاطتها بإشارة تنسيق واحدة ك ' اسم المريض '

- لعرض أول خمسة بيانات مثلا نستخدم الكلمة :top

```
SELECT top 5 * from Patients
```

- لعرض السجل الثالث مثلا نختار أكبر 3 سجلات ونختار الأصغر منها:

```
SELECT min(top 3 max(age)) from Patients
```

أو باستخدام :limit

```
SELECT * from Patients  
Limit 3
```



- لعرض نسبة مئوية من السجلات نستخدم :select top percent

```
SELECT top 50 percent from patients
```

- لدمج البيانات نستخدم إشارة الجمع :

```
SELECT name + ' تفاصيل ' + STR(age) as 'عمره' from Patients
```

ملاحظة

- التابع STR يستخدم للتحويل للبيانات النصية، إذ لا يجوز دمج بيانات مختلفة الأنواع.
- سنرى التوابع بالتفصيل في فقرة لاحقة.

- لعرض البيانات دون تكرارها:

```
SELECT DISTINCT Name FROM Patients
```

- عرض عدد البيانات غير المكررة:

```
SELECT count(DISTINCT Name) FROM Patients
```

والصيغة السابقة لن تعمل مع قواعد بيانات Access كما سنرى لاحقا، ويستعاض عنها بالأمر التالي:

```
SELECT Count(*) FROM (SELECT DISTINCT name FROM Patients)
```

مدري ليش¹ تذكرت المثل العالمي القائل: "وين إدنك ياجحا"².

- يمكن إجراء العمليات الحسابية وذلك على جملة select كما يلي:

```
select name, age + 3 from Patients
```

ناتج العملية الحسابية لن يؤثر على البيانات ومحتها وإنما يقتصر على ما سيظهر للمستخدم فقط.

¹ مدري ليش: لا أعلم لماذا.

² وين إدنك ياجحا: أن تسلك طريقا طويلا على الرغم من وجود طريق قصير يوصلك لنفس النتيجة.



عرض البيانات وفق شرط where

لعرض بيانات وفق شرط ما نستخدم الكلمة **where** والتي تعني حيث.

- عرض بيانات معينة للمريض الذي رقمه 1:

```
select name, age, gender from Patients where id = '1'
```

- عرض بيانات معينة للمرضى الذين أعمارهم 18+ 😊:

```
select name, age, gender from Patients where age > 18
```

- عرض بيانات معينة لمرضى الأسنان:

```
select name, age, gender from Patients where Sick = 'أسنان'
```

- عرض جميع بيانات مرضى الأسنان:

```
select * from Patients where Sick = 'أسنان'
```

ملاحظة

- من الممكن التعامل مع كل من الروابط `>`, `<`, `=`, `>=`, `<=`, `<>`.
- من الممكن التعامل مع كل من الروابط المنطقية `.And`, `Or`, `Nor`.
- عند التعامل مع مجالات نستخدم الكلمة `.between`.
- عند اشتراط تشابه المعطيات بالسجلات نستخدم الكلمة `.like`.
- عند اشتراط وجود بيانات معينة (عدة خيارات) نستخدم الكلمة `in`,
- يعنى هي اختصار لاستخدام الكلمة `or`.

- عرض بيانات المرضى الذكور والذين أعمارهم فوق 20:

```
select * from Patients where Age>20 and Gender = 'ذكر'
```

- عرض بيانات المرضى الذين أمراضهم جلدية أو أسنان:

```
select * from Patients where sick = 'جلدية' or sick = 'أسنان'
```



- عرض بيانات جميع المرضى عدا الأسنان:

```
select * from Patients where not sick = 'أسنان'
```

- عرض بيانات المرضى الذكور الذين أُمراضهم الأسنان أو الجلدية:

```
select * from Patients
where gender = 'ذكر' and (sick = 'أسنان' or sick = 'جلدية')
```

- عرض بيانات المرضى لأكثر من حالة (باستخدام in):

```
select * from Patients where Sick in ('جلدية', 'أسنان')
```

وكما قلنا فإن in هي اختصار لـ or، تخيل كيف كانت ستكون حياتك لو لا in:

```
select * from Patients where Sick = 'أسنان' or Sick = 'جلدية'
```

- عرض بيانات المرضى الذي يعيشون في إحدى مدن الأطباء (بفرض أن لدينا حقل للمدن التي يقطنها المرضى وحقل للمدن التي يقطنها الأطباء):

```
select * from Patients
where country in (select country from employees)
```

- عرض بيانات المرضى الذين أعمارهم بين 30 و40:

```
select * from Patients where Age between 30 and 40
```

- عرض بيانات المرضى الذين أعمارهم بين 30 و40 وأمراضهم ليست أسنان أو جلدية:

```
select * from Patients where (Age between 30 and 40)
and not Sick = 'أسنان' or Sick = 'جلدية'
```

- عرض بيانات المرضى الذين أسماؤهم بين أحمد وخالد مثلاً:

```
select * from Patients where name between 'أحمد' and 'خالد'
```



كما يمكن استخدام between مع التواريخ، وذلك عبر إحاطة التاريخ ب # أو تنسيص أحدى مثل النصوص العاديّة.

- عرض بيانات المرضى بإدخال الحروف الأولى من بياناتهم:

```
select * from Patients where Name like 's%'
```

- عرض بيانات المرضى بإدخال الحروف الأخيرة من بياناتهم:

```
select * from Patients where Name like '%a'
```

- عرض بيانات المرضى بإدخال حروف من أي موقع من بياناتهم:

```
select * from Patients where Name like '%no%'
```

- عرض بيانات المرضى بإدخال الحروف التي تبدأ بالموقع الثاني من بياناتهم:

```
select * from Patients where Name like '_no%'
```

- عرض بيانات المرضى بإدخال الحروف الأولى من بياناتهم والتي هي - بياناتهم - على الأقل ثلاثة حروف:

```
select * from Patients where Name like 'n_%_%'
```

- عرض بيانات المرضى بإدخال الحروف الأولى والأخيرة من بياناتهم:

```
select * from Patients where Name like 's%a'
```

- عرض بيانات المرضى والتي لا تبدأ بحروف معينة:

```
select * from Patients where Name not like 'sa%'
```

- عرض بيانات المرضى بإدخال حروف معينة منها والباقي غير محدد:

```
select * from Patients where Name like 's_m_r'
```



أي أن الأسماء المطابقة لنتيجة الأمر هي مثلا samir أو samer أو مثلا.

- عرض بيانات المرضى والتي من الممكن أن تبدأ أسماؤهم بحروف معينة:

```
select * from Patients where Name like '[szt]%'
```

أي أن الأسماء المطابقة من الممكن أن تكون samer أو saif أو zaid أو turki أو tamim أو ... إلخ من أسماء الكائنات البشرية التي تبدأ بـ t أو z أو s.

أو من أجل مجموعة من الحروف المتتالية:

```
select * from Patients where Name like '[a-d]%'
```

أي أن الأسماء المطابقة ستبدأ بالحروف a أو b أو c أو d.

- عرض بيانات المرضى والتي لا تبدأ أسماؤهم بحروف معينة:

```
select * from Patients where Name like '[!szt]%'
```

أو

```
select * from Patients where Name not like '[szt]%'
```

ملاحظة

- في قواعد بيانات MS ACCESS – والتي سنتناولها في الفصل التالي
- فإننا نستخدم * بدلاً من %، و ? بدلاً من _¹.

عرض البيانات وفق ترتيب معين order by

- لعرض البيانات مرتبة تصاعديا نستخدم الكلمة order by:

```
select * from Patients order by Name
```

¹ كتاب خطوة بخطوة مع فيجوال ستوديو 2008 لأحمد جمال خليفة.



- لعرض البيانات مرتبة تنازليا نضع الكلمة desc في نهاية الأمر:

```
select * from Patients order by Name desc
```

- لعرض بيانات معينة وفق شرط ما وبترتيب ما يمكن استخدام order by where معاً:

```
select name, age from Patients where Age > 18 order by name
```

كما يمكن كتابة الأوامر بشكل مفصل وأعتقد أنك لاحظت ذلك، كل جملة في سطر:

```
select name  
from Patients  
order by age desc
```

- لعرض البيانات مرتبة بشكل عشوائي نستخدم التابع (NewId):

```
SELECT * FROM Patients ORDER BY newid()
```

الإضافة والحذف والتعديل insert, delete, update

- لإضافة بيانات نستخدم الكلمة insert into، بحيث تُسبق البيانات بالكلمة values، كما أنها نضع أسماء الأعمدة في الجملة الأولى، والقيم في الجملة الثانية:

```
insert into Patients (id, Name, Age, Sick, Gender)  
values('ذكر','أسنان',18,'محمد',6)
```

- لحذف بيانات نستخدم الكلمة delete، مع تحديد شرط:

```
delete from Patients where age > 10  
delete from Patients where id = 2  
delete from Patients where name = 'محمد'
```

طبعاً كل كود مستقل عن الآخر ويمثل مثال مستقل بحد ذاته.



- لتعديل البيانات نستخدم الكلمة `update`:

```
update Patients set Age = 35 where Name = 'حسان'
```

لو لم نضع شرطاً لتم تعديل جميع البيانات!.. كما يمكن تعديل أكثر من عمود بوضع فاصلة بين أسماء الأعمدة..

التابع الجاهزة

تعطيك التابع الجاهزة - كما في جميع لغات البرمجة - نتائج معينة بغض النظر عن الأكواد المستخدمة. كما يمكن استخدام الشروط والترتيب أيضاً عند التعامل مع التابع، الأمثلة التالية تعطيك مجموعة من التابع ومعناها:

- للحصول على أكبر قيمة لعمود نستخدم التابع `max`:

```
select max(age) from Patients
```

وبالمثل بإمكاننا الحصول على أصغر قيمة `Min` والمتوسط `Avg` والمجموع `Sum` والعدد `Count` وتسمى هذه التابع بـ `تتابع إحصائية`. بالإضافة لإمكانية استخدام أكثر من تابع معاً:

```
select max(age),min(age),avg(age) from Patients
```

ملاحظة

- بإمكانك تطبيق التابع الجاهزة على السجلات غير المكررة، قد لا يختلف ناتج بعض التابع مثل `Min Max`، إلا أن التابع التي تجري الحسابات تختلف نتيجتها بشكل كامل، لذلك استخدم الكلمة `select count(distinct age) from Patients` كالكود التالي:

كما يوجد دوال رياضية أخرى مثل `Sqrt` `Abs` والتي تستخدم للحصول على القيمة المطلقة والجذر، دوال نصية مثل `Left` `Right` `Upper` `Lower` `Len` وهي معروفة لمبرمجي `C#`.

أما بالنسبة لدوال التاريخ والوقت فهي كثيرة وهامة ويجب الإلمام بها، فالبرامج التي تتعامل مع قواعد البيانات غالباً ما يهمها التعامل مع بياناتها بحسب التاريخ والوقت.



- إضافة قيمة لمتغير من نوع التاريخ والوقت نستخدم (:DATEADD()

```
SELECT name,DATEADD(year,1,'8/24/1996')
```

حيث التابع dateadd يأخذ ثلاثة بارامترات أولها مقدار الزيادة ولها خيارات كثيرة مثل day وmonth وyear وweek وsecond، كما يمكن أن تكون minute وhour.. أما البارمتر الثاني فهو مقدار الزيادة.. والثالث هو التاريخ والوقت.

- للحصول على الفرق بين تاريخين نستخدم (:DATEDIFF())

```
SELECT DATEDIFF(year, '5/3/1996', '9/28/2018' )
```

- كما يمكن الحصول على الوقت الحالي لنظام التشغيل من خلال التابع (:GETDATE())

```
SELECT GetDate()
```

كما يوجد (:DATEPART) والذي يستخدم للحصول على جزء معين من التاريخ أو الوقت.

بالإضافة للدوال السابقة لدينا دوال تحويل الأنواع، فلتتحويل إلى العبارات النصية كما رأينا نستخدم التابع (:STR)، ولباقي الأنواع (:Convert).

تجميع البيانات group by

- لعرض البيانات وفق مجموعات نستخدم الكلمة :group by

```
select gender, max(age) from Patients group by gender
```

حيث سيقوم الأمر السابق بتجميع البيانات تبعاً للعمود gender، ويعرض بجانبه عموداً يحوي أكبر قيمة للعمر. أي سيعرض عمودين الأول فيه ذكر وأنثى أو الثاني فيه أكبر قيمة لعمر الذكور أو أكبر قيمة لعمر الإناث.

- لعرض البيانات وفق مجموعات وبشرط إحصائي معين نستخدم (:having)

```
select sick, avg(age) from Patients group by sick having count(*)>2
```



حيث يقوم الأمر الأخير بتجميع البيانات تبعاً للعمود sick ويعرض بجانبه متوسط أعمار المرضى وذلك إذا كان عدد المرضى أكثر من 2. أي سيعرض عمودين الأول فيه أسماء الأمراض والتي عدد مرضها أكثر من 2، والثاني فيه متوسط أعمار مرضى هذا المرض.

ملاحظة

- العمليات الإحصائية توضع في جملة having إذا استخدمت كشرط، غير ذلك توضع في جملة select

النسخ الاحتياطي

- لأخذ نسخة احتياطية نستخدم الكلمة backup database، وعادة ما يتم تصدير قاعدة البيانات إلى ملف من النوع bak، أو بإمكانك جعله أي نوع ترغب به:

```
backup database ClinicDB to disk = 'e:\Clinic.bak'
```

- لاستعادة النسخة الاحتياطية نستخدم الكلمة restore database، ولكن لا بد من حذفها أولاً لذلك استخدم drop database قبل الاستعادة.. كما أنه لاستعادة قاعدة بيانات ما يجب ألا تكون مستخدمة وهذا هام جداً.

```
drop database ClinicDB
restore database ClinicDB from disk = 'e:\Clinic.bak'
```

عند استعادة نسخة احتياطية يجب الاتصال بقاعدة بيانات معايرة للقاعدة التي تستعيدها، مثل ..MASTER

إنشاء اتصال مع قاعدة البيانات

للتعامل مع أوامر قواعد البيانات يجب إلتحق مجال الأسماء System.Data.SqlClient، ثم التصريح عن كائن يمثل كائن الاتصال بالسرفر (أو بقاعدة البيانات) :SqlConnection:

```
SqlConnection connection = new SqlConnection ();
connection = new SqlConnection ("Data Source=Eng27; Initial Catalog=New;
Integrated Security=true");
```



أو بالشكل:

```
SqlConnection connection = new SqlConnection ();
connection = new SqlConnection ("Server=Eng27; Database=New;
Integrated Security=true");
```

وهذا الكائن يقوم بالتصريح عنه خارج إجراءات الفورم، أي داخل فئة الفورم مباشرة، حيث إن Eng27 هو اسم السرفر الذي تعمل عليه - في SQL - New اسم قاعدة البيانات true تعني أنه ليس لقاعدة البيانات حماية. ثم نستدعي الإجراء Open للكائن connection ضمن الإجراء الذي سيحدث عنده الاتصال بالسرفر:

```
connection.Open();
```

بإمكانك الآن الحصول على الكثير من خصائص الاتصال، وذلك من خلال خصائص connection مثلConnectionString والتي تعطيك مواصفات السرفر والتي قمت بإدخالها عند التصريح عن كائن الاتصال connection، وDataSource والتي تعطيك اسم قاعدة البيانات، وDataSource والتي تعطيك اسم السرفر، وState للحصول على حالة الاتصال، مغلق أم مفتوح، وغيرها من الخصائص.

الاتصال باستخدام كائن بناء الاتصال

كما هو الحال في المتغيرات النصية string والتي يُنصح باستخدام كائن SqlConnectionStringBuilder بدلاً من استخدام متغيرات string حاف¹، فإنه مع أكواد الاتصال بقواعد البيانات بإمكانك استخدام كائن SqlConnectionStringBuilder والذي يقوم بتجميع معلومات الاتصال وبناء صيغة نصية للاتصال من خلالها.

```
SqlConnectionStringBuilder csb = new SqlConnectionStringBuilder();
csb.InitialCatalog = "New";
csb.DataSource = "Eng27";
csb.ConnectTimeout = 30;
connection = new SqlConnection();
connection.ConnectionString = csb.ConnectionString;
connection.Open();
```

¹ حاف: لوحدة.



القراءة من قاعدة البيانات

لقراءة البيانات نتعامل مع كائن `SqlCommand` والكائن `SqlDataReader`. فالأخير يقوم بتنفيذ أوامر SQL Server والتي تحدثنا عنها في فقرة سابقة مثل جمل `select` و `insert` وغيرها، أما الأول فهو يقوم بقراءة البيانات عند التعامل معها من خلال الكائن `SqlCommand`.

```
في التصريحات العامة//  
SqlConnection connection;  
SqlCommand command;  
SqlDataReader reader;  
//الاتصال بالسرفر  
// يتم عادة في بداية تحميل النموذج  
connection = new SqlConnection("server=Eng27; database=ClinicDB;  
Integrated Security=true");  
connection.Open();  
// في أحد إجراءات النموذج1  
List<string> Names = new List<string>();  
command = new SqlCommand("select name from patients",connection);  
reader = command.ExecuteReader();  
while (reader.Read())  
{  
    Names.Add(reader[0].ToString()); //إضافة العناصر للائحة  
}  
listBox1.Items.AddRange(Names.ToArray()); //إظهار العناصر للمستخدم  
reader.Close();  
connection.Close();
```

قمنا بالتصريح عن ثلاثة كائنات، الأول لإنشاء الاتصال والثاني للقيام بأوامر SQL Server والثالث لقراءة نتائج كائن الأوامر. ثم قمنا بالاتصال بالسرفر.

صرحنا بعدها عن لائحة لإضافة عناصر قاعدة البيانات إليها، فاللائحة تتميز بكونها ديناميكية ولا تتطلب عدد العناصر أثناء تعريفها. ثم أجرينا الأمر "اختر بيانات عمود الأسماء من جدول المرضى"، وذلك عبر الاتصال `connection` (هذا الكائن يحمل مواصفات السرفر وقاعدة البيانات ونوع الاتصال). ثم قرأتنا ناتج تنفيذ الأمر كما سيظهر في بيئه SQL Server، وأضفنا هذه البيانات إلى اللائحة التي صرحنا عنها في بداية الكود.

¹ عندما نقول أحد إجراءات النموذج أو الفورم فاننا نقصد الإجراءات مثل `button1_Click` أو `Form1_Load` أو `textBox1_ChangeText` أو غيرها من الإجراءات..



ولإظهار القيم المستخدمة أضفنا جميع عناصر اللائحة إلى `ListBox`. حيث إن هذه الخطوة يمكن اختصارها وإضافة العناصر من كائن القراءة `reader` مباشرة.

وأخيرا لا بد من إغلاق كائن القراءة `reader` وكائن الاتصال `connection`.

إضافة وحذف بيانات إلى قاعدة البيانات

لإضافة بيانات تحتاج للتعامل مع كائن من نوع `SqlCommand` فقط، وقد تم بيان كود إضافة بيانات في SQL Server وذلك في فقرة سابقة. وعند تنفيذ أمر من أوامر SQL Server التي لا تعيّد قيمة من خلال C# يجب أن نتعامل مع الإجراء `ExecuteNonQuery` وذلك باستدعاءه من كائن مستنسخ عن الفئة `SqlCommand`. وهذا الإجراء يعوضك عن كتابة الأوامر في استعلام `Query` في بيئة SQL Server. لاحظ:

```
int id = 8 ;
string name = "حسام";
int age = 41;
string sick = "أعصاب", gender = "ذكر";
reader.Close();
string insert = string.Format("insert into Patients (id,Name,Age,Sick,Gender)
values ({0},{1},{2},{3},{4})",
    id,
    name,
    age,
    sick,
    gender);
try
{
    لا تنسى إجراء الاتصال في إجراء ما قبل فتحه // connection.Open();
    command = new SqlCommand(insert, connection);
    command.ExecuteNonQuery();
    MessageBox.Show("تم إضافة البيانات بنجاح!");
}
catch (SqlException ex)
{
    MessageBox.Show(ex.Message);
}
Finally
{
    connection.Close();
}
```



حيث إن كل من المتغيرات `id`, `sick`, `age`, `name` و `gender` تدخل من خلال بعض الأدوات مثل صندوق الإدخال وغيرها، أو من خلال عمليات حسابية معينة، وتم إدخال البيانات برمجياً لإيصال الفكرة فقط (أي أسنداً لها المتغيرات قيمةً عوضاً عن استقبالها من أدوات الإدخال).. حتى أنك لا تحتاج أصلاً لتعريف متغير نصي لتضع فيه كود الإضافة، لكن لتنظيم الكود وسهولته اتبعت هذا الإسلوب.

وبطريقة مشابهة يتم حذف البيانات وفق الشكل التالي:

```
int id = 1;
string delete = "delete from Patients where id = " + id.ToString();
command = new SqlCommand(delete, connection);
command.ExecuteNonQuery();
connection.Close();
```

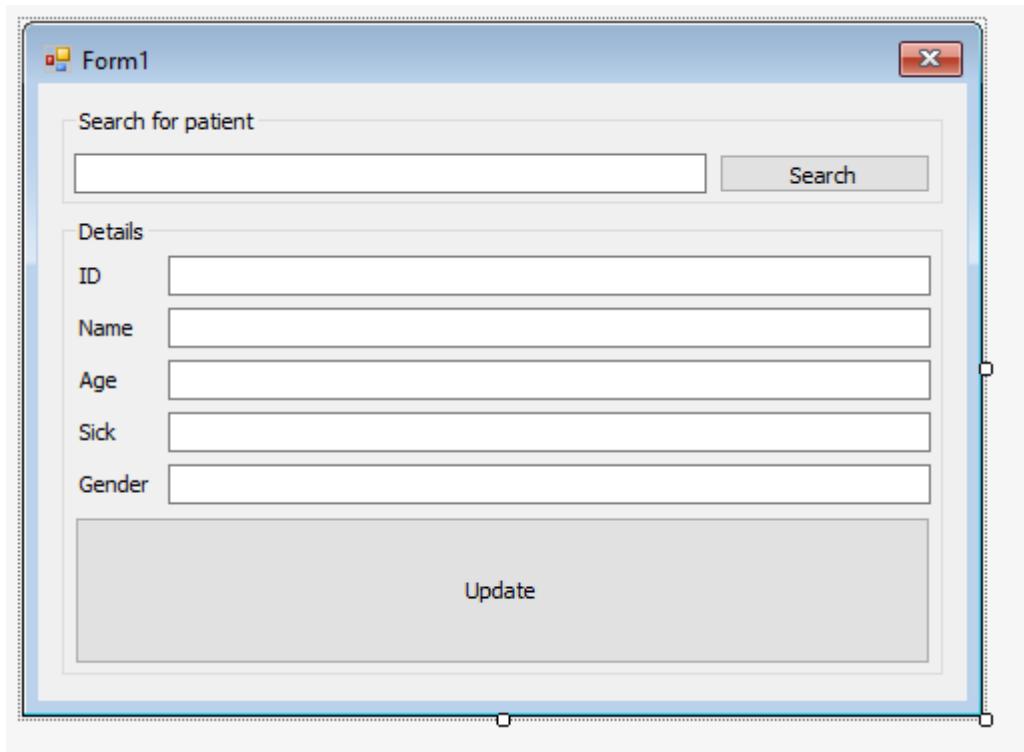
ملاحظة

- عندما لا نحتاج لإرجاع بيانات عند تنفيذ كود SQL مثل أوامر الحذف `.ExecuteNonQuery()` بالإضافة نستخدم الطريقة

تعديل بيانات قاعدة البيانات

كما في إضافة وحذف البيانات، فإننا عند تعديل البيانات ننفذ أمراً من أوامر SQL Server والتي ننفذها داخل استعلام Query، وللاستغناء عن الاستعلامات من أجل تنفيذ الكود من خارج SQL Server فإننا نتعامل مع الإجراء `..ExecuteNonQuery()`

صمم نافذة مشابهة للنافذة التالية، اضبط خاصية التفعيل لصندوق التجميع الثاني على `:false`



ثم استخدم الكود التالي:

```
SqlConnection connection;
SqlCommand command;
SqlDataReader reader;
public Form1()
{
    InitializeComponent();
    connection = new SqlConnection("server=Eng27; database=ClinicDB;
Integrated Security=true");
}
private void button1_Click(object sender, EventArgs e)
{
    connection.Open();
    command = new SqlCommand("select id,name,age,sick,gender from
patients where name = '" + textBox1.Text + "'", connection);
    reader = command.ExecuteReader();
    reader.Read();
    try
    {
        groupBox2.Enabled = true;
        groupBox2.Text = "Details";
        textBox2.Text = reader["id"].ToString();
        textBox3.Text = reader["name"].ToString();
        textBox4.Text = reader["age"].ToString();
        textBox5.Text = reader["sick"].ToString();
        textBox6.Text = reader["gender"].ToString();
    }
}
```



```

}
catch (Exception ex)
{
    groupBox2.Enabled = false;
    groupBox2.Text = ex.Message;
}
reader.Close();
connection.Close();
}
private void button2_Click(object sender, EventArgs e)
{
    connection.Open();
    string update = string.Format("update Patients set id = {0}, name = '{1}', 
age = {2}, sick = '{3}', gender = '{4}' where Name = '{1}'", 
    textBox2.Text,
    textBox3.Text,
    textBox4.Text,
    textBox5.Text,
    textBox6.Text);
try
{
    command = new SqlCommand(update, connection);
    command.ExecuteNonQuery();
    MessageBox.Show("Updated successfully!");
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
reader.Close();
connection.Close();
}

```

استخدام SqlParameter

كما هو الحال مع SqlConnectionStringBuilder يمكنك تنظيم الكود وحمايته من خلال ضبط متغيراته باستخدام كائن من نوع `:SqlParameter`

```

string show = string.Format("Select * From Patients where ID=@ID");
using (command = new SqlCommand(show, connection))
{
    SqlParameter param = new SqlParameter();
    param.ParameterName = "@ID";
    param.Value = 10;
    param.SqlDbType = SqlDbType.Int;
    command.Parameters.Add(param);
}

```



```
        command.ExecuteNonQuery();
    }
```

من المحتمل أنه قد تشكلت مجموعة من إشارات الاستفهام عن عبارة "حماية الكود" المذكورة قبل قليل.. في الواقع الحماية ليست للكود بحد ذاته - بغض النظر أن "هـ" في حمايته عائدۀ على كلمة الكود - وإنما هي حماية للبرنامج بشكل عام من السرقة والتعفيش¹، وستشعر بكلامي هذا في فقرة SQL Injection.

عرض البيانات في جدول عرض

من أهم طرق التفاعل مع بيانات قواعد البيانات هي عرضها في جدول عرض، فهي تعطيك رؤية واسعة وشاملة للبيانات وتعطيك إمكانيات عرض وتعديل كبيرة..

هناك أكثر من نوع للاتصال بقواعد البيانات، متصل ومنفصل. الأول يتم فيه التعامل مع قاعدة البيانات مباشرةً وذلك من خلال كائن مستنسخ عن الفئة SqlCommand، والثاني يتم من خلال كائن مستنسخ من الفئة SqlDataAdapter، وكلا الكائنين يؤمنان لك قراءة وعرض وتعديل وحذف البيانات، والفارق أن الكائن الذي يعمل في الوضع المتصل يقوم بتعديل البيانات مباشرةً ولا يحتاج لتخزينها في الذاكرة أما الثاني فيقوم بحجز أماكن في الذاكرة لوضع البيانات فيها.

عندما تعاملنا مع SqlCommand فإننا نرأت البيانات الناتجة عن تنفيذ الأمر من خلال SqlDataReader، أما عند التعامل مع SqlDataAdapter فإننا سنقرأ البيانات من خلال DataSet أو DataTable، فال الأول يعطيك إمكانية تخزين قاعدة بيانات كاملةً في الذاكرة أما الآخر فيسمح لك بحجز مكان لجدول واحد فقط..

أضف DataGridView إلى مشروعك وطبق الكود التالي لتحصل على البيانات في أداة جدول العرض:

```
في قسم التصريحات العامة//  
SqlConnection connection;  
SqlDataAdapter adapter;  
DataSet data;  
في أحد إجراءات البرنامج//
```

¹ التعفيش: النهب والسرقة في غياب الحكومة، وهي كلمة مشتقة من كلمة عَفْش وقد ظهرت أثناء الحرب السورية. والعفش: بفتح العين وسكون الفاء، هو الأثاث.



```
// ويفضل في إجراء يعمل تلقائيا مع بداية البرنامج //
// حتى يعرض الجدول تلقائيا مع بداية تشغيل البرنامج //
connection = new SqlConnection("server=Eng27; database=ClinicDB;
Integrated Security=true");
adapter = new SqlDataAdapter("select * from patients", connection);
data = new DataSet(); // لا تنسى استنساخ الكائن
adapter.Fill(data, "Table"); // اسم الجدول على ذوقك
dataGridView1.DataSource = data.Tables["Table"];
```

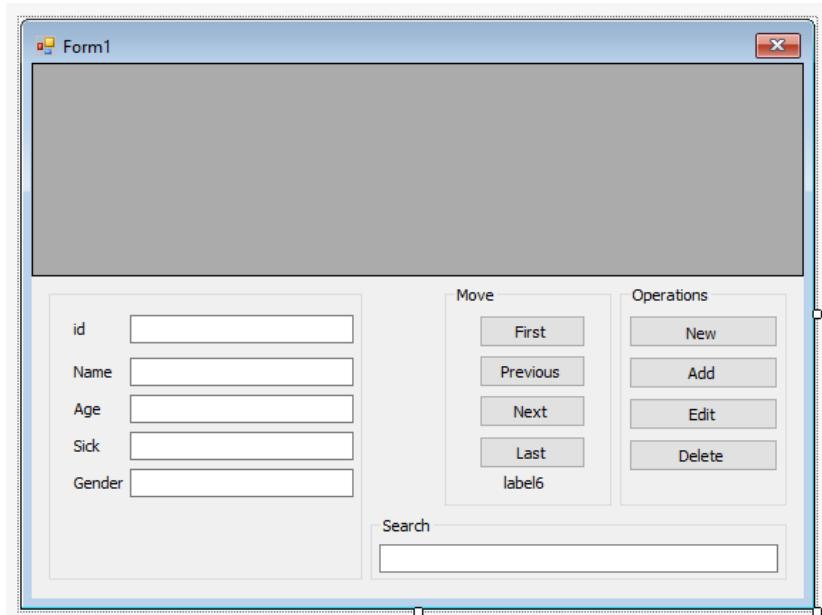
عند تكوين جدول في `data` بإمكانك تسميته بأي اسم ترغب به ولست ملزماً بالاسم الأصلي للجدول الذي تتعامل معه، لأنك أصلاً تقوم باستنساخ قاعدة البيانات أو جزء منها إلى الكائن `data`، وهذا الجزء المستنسخ لا يمت لقاعدة البيانات الأصلية بأي صلة وهو ملكية خاصة للكائن `.data`.

التنقل بين البيانات والعمليات عليها

تحدثنا عن بعض العمليات على البيانات مثل الحذف والتعديل والإضافة، كما تحدثنا عن كيفية عرضها في مربعات نصوص أو في جدول عرض.. والآن سندمج الفكرتين ونضيف عليهم أفكاراً أخرى..

سنعطي المستخدم إمكانية التنقل بين القيم من خلال أزرار تنقل، كما سيستطيع إضافة البيانات والتعديل عليها والبحث عنها.. في الواقع فمكونات هذه الفقرة توجز جميع ما سبقها في هذا الفصل.

صمم النافذة التالية:





ألغِ تفعيل زر Add، وغير التسمية البرمجية Name لجميع الأزرار إلى القيمة النصية Text الظاهرة عليها لتسهيل الأكواود.

```

 SqlConnection connection; //الاتصال بالسرفر
 SqlDataAdapter adapter; //المنفصل
 SqlDataReader reader; //الأوامر
 SqlCommand command; //البيانات
 DataTable data; //جداول البيانات
 CurrencyManager manager; //قواعد البيانات
 SqlCommandBuilder commandbuilder;
 public Form1()
 {
    InitializeComponent();
    إنشاء اتصال
    connection = new SqlConnection("server=Eng27; database=ClinicDB;
Integrated Security=true");
    adapter = new SqlDataAdapter("select * from patients", connection);
    استنساخ جدول
    data = new DataTable();
    adapter.Fill(data); //ملئ الجدول
    عرض الجدول في أداة جدول العرض //إعطاء أسماء الكائنات والأدوات التي سيديرها كائن الإدارة
    manager = (CurrencyManager)this.BindingContext[data];
    textBox2.DataBindings.Add("Text", data, "id");
    textBox3.DataBindings.Add("Text", data, "name");
    textBox4.DataBindings.Add("Text", data, "age");
    textBox5.DataBindings.Add("Text", data, "sick");
    textBox6.DataBindings.Add("Text", data, "gender");
}
private void New_Click(object sender, EventArgs e)
{
    manager.AddNew();
    Add.Enabled = true;
    New.Enabled = false;
    textBox2.Focus();
}
private void Add_Click(object sender, EventArgs e)
{
    manager.EndCurrentEdit();
    commandbuilder = new SqlCommandBuilder(adapter );
    adapter.Update(data);
    manager.Refresh();
    MessageBox.Show("Added successfully!");
    label6.Text = manager.Position + 1 + " of " + data.Rows.Count;
    Add.Enabled = false;
    New.Enabled = true;
}

```



```

private void First_Click(object sender, EventArgs e)
{
    manager.Position = 0;
    label6.Text = manager.Position + 1 + " of " + data.Rows.Count;
}
private void Previous_Click(object sender, EventArgs e)
{
    manager.Position -= 1;
    label6.Text = manager.Position + 1 + " of " + data.Rows.Count;
}
private void Next_Click(object sender, EventArgs e)
{
    manager.Position += 1;
    label6.Text = manager.Position + 1 + " of " + data.Rows.Count;
}
private void Last_Click(object sender, EventArgs e)
{
    manager.Position = data.Rows.Count - 1;
    label6.Text = manager.Position + 1 + " of " + data.Rows.Count;
}
private void Delete_Click(object sender, EventArgs e)
{
    manager.RemoveAt(manager.Position);
    manager.EndCurrentEdit();
    commandbuilder = new SqlCommandBuilder(adapter);
    adapter.Update(data);
    manager.Refresh();
    MessageBox.Show("Deleted successfully!");
    label6.Text = manager.Position + 1 + " of " + data.Rows.Count;
}
private void Edit_Click(object sender, EventArgs e)
{
    //Way1

    //manager.EndCurrentEdit();
    //commandbuilder = new SqlCommandBuilder(adapter);
    //adapter.Update(data);
    //manager.Refresh();
    //MessageBox.Show("Edited successfully!");
    //label6.Text = manager.Position + 1 + " of " + data.Rows.Count;

    //Way2

    connection.Open();
    string update = string.Format("update Patients set id = {0}, name = '{1}',  
age = {2}, sick = '{3}', gender = '{4}' where Name = '{1}'",  
        textBox2.Text,

```



```
    textBox3.Text,
    textBox4.Text,
    textBox5.Text,
    textBox6.Text);
try
{
    command = new SqlCommand(update, connection);
    command.ExecuteNonQuery();
    MessageBox.Show("Updated succesfully!");
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
connection.Close();
manager.Refresh();
}
private void textBox1_TextChanged(object sender, EventArgs e)
{
    DataView dv = new DataView(data);
    dv.RowFilter = string.Format("Name LIKE '%{0}%'", textBox1.Text);
    dataGridView1.DataSource = dv;
    manager.Refresh();
    connection.Open();
    command = new SqlCommand("select id,name,age,sick,gender from patients where name = '" + textBox1.Text + "'", connection);
    reader = command.ExecuteReader();
    reader.Read();
    try
    {
        textBox2.Text = reader["id"].ToString();
        textBox3.Text = reader["name"].ToString();
        textBox4.Text = reader["age"].ToString();
        textBox5.Text = reader["sick"].ToString();
        textBox6.Text = reader["gender"].ToString();
    }
    catch
    {
    }
    reader.Close();
    connection.Close();
}
```



الإجراءات المخزنة

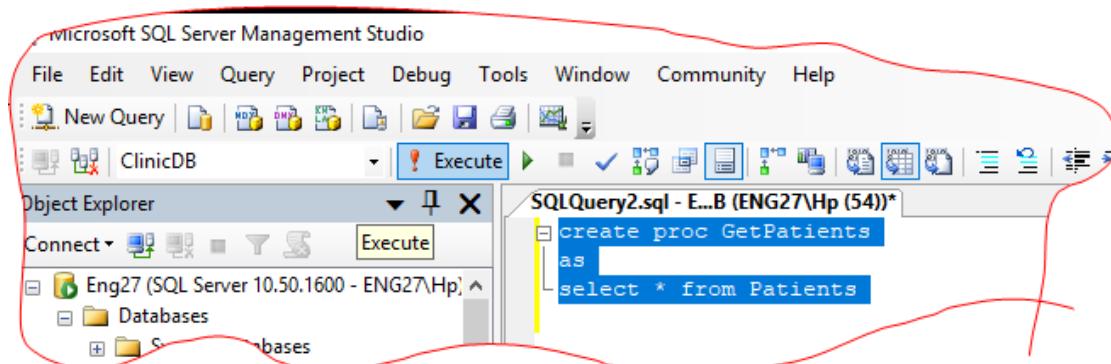
كما في مشاريع C#, بإمكانك إنشاء إجراءات خاصة بك في SQL Server، فبدلاً من تمرير أوامر SQL Server كoste في كائنات C#, بإمكانك كتابة هذه الأوامر في إجراءات معينة تخزنها في استعلام ما ثم تستدعيها عند الحاجة في مشروعك.

سنأخذ مثلاً بسيطاً فيه أداة عرض جدول بعض الأزرار مثل حذف وإضافة بيانات وغيرها..

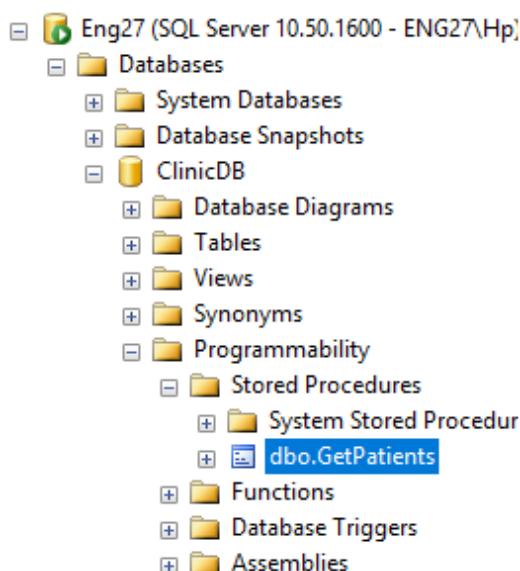
اده إلى الاستعلام الذي أنشأته واتكتب فيه:

```
create proc GetPatients
as
select * from Patients
```

ثم حدد الأسطر الثلاثة التي كتبتها واضغط تنفيذ كما هو موضح بالصورة:



ليتم إضافة هذا الإجراء إلى شجرة العرض:





ثم اذهب إلى الفيوجوال ستوديو واتكتب الكود التالي للاتصال بالقاعدة:

```
public Form1()
{
    InitializeComponent();
    connection = new SqlConnection("server=Eng27; database=ClinicDB;
Integrated Security=true");
    GetStart();
}
void GetStart()
{
    command = new SqlCommand("GetPatients", connection);
    // يجب ضبط كائن الأوامر على أنه يتعامل مع الإجراءات المخزنة
    command.CommandType = CommandType.StoredProcedure;
    adapter = new SqlDataAdapter(command);
    data = new DataTable(); adapter.Fill(data);
    // ملء الجدول
    dataGridView1.DataSource = data;
    // عرض الجدول في أداة جدول العرض
    manager = (CurrencyManager)this.BindingContext[data];
}
```

لا تنسى التصريح عن الكائنات المستخدمة مثل `command` و `adapter` وغيرها..

جرب الكود، قارن بين الطريقتين، وناقش الفوارق، وقارن النتائج فيما لو اتصلنا بقاعدة البيانات بالوضع المتصل (باستخدام `command` وليس `adapter`). ويرأيك لماذا قمنا باستنساخ الكائن `adapter` مع تمرير `command` ك وسيط له؟؟ (أعتقد أنه يخطر على بالك أننا نود الاتصال بالوضع المنفصل)..

كرر العملية من أجل إضافة مريض، في الاستعلام:

```
create proc AddPatient
@id int, @name nvarchar(50), @age int, @sick nvarchar(50),
@gender nvarchar(50)
as
insert into Patients (id,Name,Age,Sick,Gender)
values(@id,@name,@Age,@Sick,@Gender)
```

لا تنسى تحديد الإجراء بالكامل والضغط على تنفيذ.. وتأكد من أن الإجراء قد أضيف إلى شجرة العرض!



ملاحظة

- عند تعريف المتغيرات في استعلام من استعلامات SQL Server فإنه يجب إضافة رمز @ إلى بداية اسم المتغير.
- لا يفرق SQL Server بين الأحرف الكبير والأحرف الصغيرة فهو ليس حساساً لحالة الحروف.
- لترك ملاحظات في SQL Server نضع إشارة -- في بداية الجملة المراد جعلها ملاحظة.

وفي الفيديو ستوديو:

```
private void Add_Click(object sender, EventArgs e)
{
    command = new SqlCommand("AddPatient", connection);
    command.CommandType = CommandType.StoredProcedure;
    SqlParameter[] p = new SqlParameter[5];

    p[0] = new SqlParameter("@id", SqlDbType.Int);
    p[0].Value = textBox2.Text;

    p[1] = new SqlParameter("@name", SqlDbType.NVarChar, 50);
    p[1].Value = textBox3.Text;

    p[2] = new SqlParameter("@age", SqlDbType.Int);
    p[2].Value = textBox4.Text;

    p[3] = new SqlParameter("@sick", SqlDbType.NVarChar, 50);
    p[3].Value = textBox5.Text;

    p[4] = new SqlParameter("@gender", SqlDbType.NVarChar, 50);
    p[4].Value = textBox6.Text;

    command.Parameters.AddRange(p);
    connection.Open();
    command.ExecuteNonQuery();
    connection.Close();

    MessageBox.Show("Added successfully!");
    Add.Enabled = false;
    New.Enabled = true;
    data.Clear();
    GetStart();
}
```



والجديد في الكود هو الكائن المستنسخ عن الفئة SqlParameter، وهو كائن يمثل متغيرات SQL Server، استنسخنا مصفوفة من الكائنات مكونة من 5 عناصر، في كل عنصر أعطيته اسمًا أثناء التصريح نوعاً وحجماً إن طلب الأمر كما في SQL Server، ومن ثم أعطيته قيمةً.. ثم أضفنا هذه المتغيرات إلى الكائن command والذي بدوره يستدعي إجراءً مخزناً وفق ..connection

وبالمناسبة، فكود الزر New قد تغير أيضاً فلا وجود لـ manage ليدير أدواتك، فهذا الكود يقوم بتفعيل زر حفظ البيانات مع تهيئة أدوات الإدخال - مربعات النصوص - لاستقبال بيانات جديدة:

```
private void New_Click(object sender, EventArgs e)
{
    textBox2.Clear();
    textBox3.Clear();
    textBox4.Clear();
    textBox5.Clear();
    textBox6.Clear();
    Add.Enabled = true; New.Enabled = false;
    textBox2.Focus();
}
```

ومن أجل الحذف:

```
create proc DeletePatient
@id int
as
delete from Patients where id = @id
```

وفي الفيجوال ستوديو:

```
private void Delete_Click(object sender, EventArgs e)
{
    command = new SqlCommand("DeletePatient", connection);
    command.CommandType = CommandType.StoredProcedure;

    SqlParameter p = new SqlParameter();
    p = new SqlParameter("@id", SqlDbType.Int);
    p.Value = Convert.ToInt32(textBox2.Text);

    command.Parameters.Add(p);
    connection.Open(); command.ExecuteNonQuery(); connection.Close();
```



```
GetStart();  
MessageBox.Show("Deleted successfully!");  
}
```

ولتعديل البيانات:

```
create proc EditPatient  
@id int, @name nvarchar(50), @age int, @sick nvarchar(50),  
@gender nvarchar(50)  
as  
update Patients  
set id = @id,  
Name=@name,  
Age=@age,  
Sick=@sick,  
Gender=@gender  
where id=@id
```

وفي الفيجوال ستوديو:

```
private void Edit_Click(object sender, EventArgs e)  
{  
    command = new SqlCommand("EditPatient", connection);  
    command.CommandType = CommandType.StoredProcedure;  
    SqlParameter[] p = new SqlParameter[5];  
    p[0] = new SqlParameter("@id", SqlDbType.Int);  
    p[0].Value = textBox2.Text;  
  
    p[1] = new SqlParameter("@name", SqlDbType.NVarChar, 50);  
    p[1].Value = textBox3.Text;  
  
    p[2] = new SqlParameter("@age", SqlDbType.Int);  
    p[2].Value = textBox4.Text;  
  
    p[3] = new SqlParameter("@sick", SqlDbType.NVarChar, 50);  
    p[3].Value = textBox5.Text;  
  
    p[4] = new SqlParameter("@gender", SqlDbType.NVarChar, 50);  
    p[4].Value = textBox6.Text;  
  
    command.Parameters.AddRange(p);  
    connection.Open();  
    command.ExecuteNonQuery();  
    connection.Close();  
    MessageBox.Show("Edited successfully!");
```



```

        data.Clear();
        GetStart();
    }
private void dataGridView1_Click(object sender, EventArgs e)
{
    int i = dataGridView1.CurrentRow.Index;
    textBox2.Text = dataGridView1.Rows[i].Cells["id"].Value.ToString();
    textBox3.Text = dataGridView1.Rows[i].Cells["name"].Value.ToString();
    textBox4.Text = dataGridView1.Rows[i].Cells["age"].Value.ToString();
    textBox5.Text = dataGridView1.Rows[i].Cells["sick"].Value.ToString();
    textBox6.Text = dataGridView1.Rows[i].Cells["gender"].Value.ToString();
}

```

SQL Injection

إن أهم استخدام للإجراءات المخزنة هو الحماية من SQL Injection، طبعا له استخدامات أخرى. وحقن SQL أو حقن كود SQL هو عبارة عن كتابة أكواد SQL في أدوات الإدخال في البرنامج بحيث يتم تنفيذها في حين أنه يفترض عدم تنفيذها. كما يمكن تفادي حقن أكواد SQL من خلال ضبط بارامترات للكائن Command حتى لو لم يتم استخدام الإجراءات المخزنة.

والموضوع بهذا البساطة: لديك صندوقٌ نصوص أحدهما يستقبل اسم المستخدم والآخر يستقبل كلمة السر الخاصة به، وبدلًا من أن تدخل البيانات الصحيحة - الاسم وكلمة السر - فإنك تدخل كود SQL لتحقيق أغراض معينة.

صحيح أن البرنامج سيعطيك رسالة اعتذار مفادها أن كلمة السر واسم المستخدم خاطئان - أو أحدهما على الأقل - فإنك ستكون قد نفذت أغراضًا خاصة بك. تذكر: البرنامج ليس براماجك وأنت لست مبرمجه، وأنت مستخدم عادي في حالتنا!

الربط بين أكثر من جدول

الغاية الأساسية من قواعد البيانات هي إضفاء التنظيم وسهولة الوصول والقراءة والتعديل على البيانات، ولمزيد من التنظيم بإمكانك توزيع البيانات على أكثر من جدول..

قد لا تكون الجداول مترابطة مع بعضها مثل حالة جدول المرضى والموظفين في المستوى، لا يوجد رابط بينهما ولا علاقة بين المريض



والموظف. ولكن بإضافة جدول ثالث يحوي أسماء الأمراض مثلاً يصبح هناك رابط بين الموظفين والمريض وهو وجود الأمراض.. فالمريض القادم للستوصف معه مرض معين، وأحد موظفي المستوصف هو طبيب يعالج هذا المرض، لذلك يوجد رابط بين هذا الموظف وذاك المريض وهو وجود مرض ما يعاني منه المريض ويعالجه الموظف.

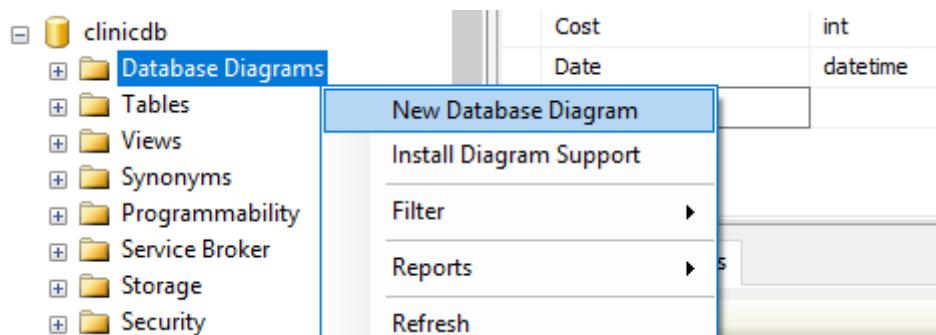
لذلك قم بإنشاء جدول جديد يمثل بيانات الأمراض، وعلى سبيل المثال سيحتوي هذا الجدول على رقم للمرض ووصف له فقط. ونحتاج أيضاً لجدول آخر يربط بين جدول الأمراض وجدول المرضى وجدول الموظفين، بالإضافة إلى احتواه على تاريخ زيارة المريض ومقدار فاتورته مثلاً. (هذا يعني أنه لا حاجة لعمود المرض sick في جدول المرضى لأننا سننشأ جدولًا خاصًا بالأمراض).

ENG27.clinicdb - dbo.Sick		
Column Name	Data Type	Allow Nulls
id	int	<input type="checkbox"/>
Sick	nvarchar(50)	<input checked="" type="checkbox"/>

ENG27.clinicdb - dbo.Visits		
Column Name	Data Type	Allow Nulls
id	int	<input type="checkbox"/>
PatientID	int	<input checked="" type="checkbox"/>
EmployeeID	int	<input checked="" type="checkbox"/>
SickID	int	<input checked="" type="checkbox"/>
Cost	int	<input checked="" type="checkbox"/>
Date	datetime	<input checked="" type="checkbox"/>

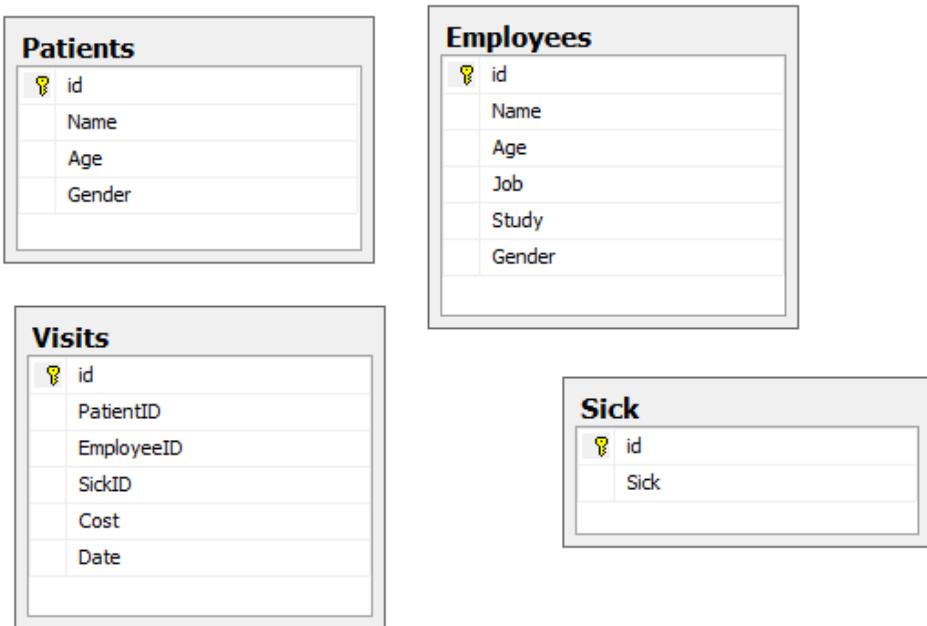
أصبح لدينا أربعة جداول، الأول للمرضى والثاني للموظفين والثالث للأمراض والرابع لزيارات المرضى.

أضف مخططًا بيانيًا Diagram لربط الجداول مع بعضها:

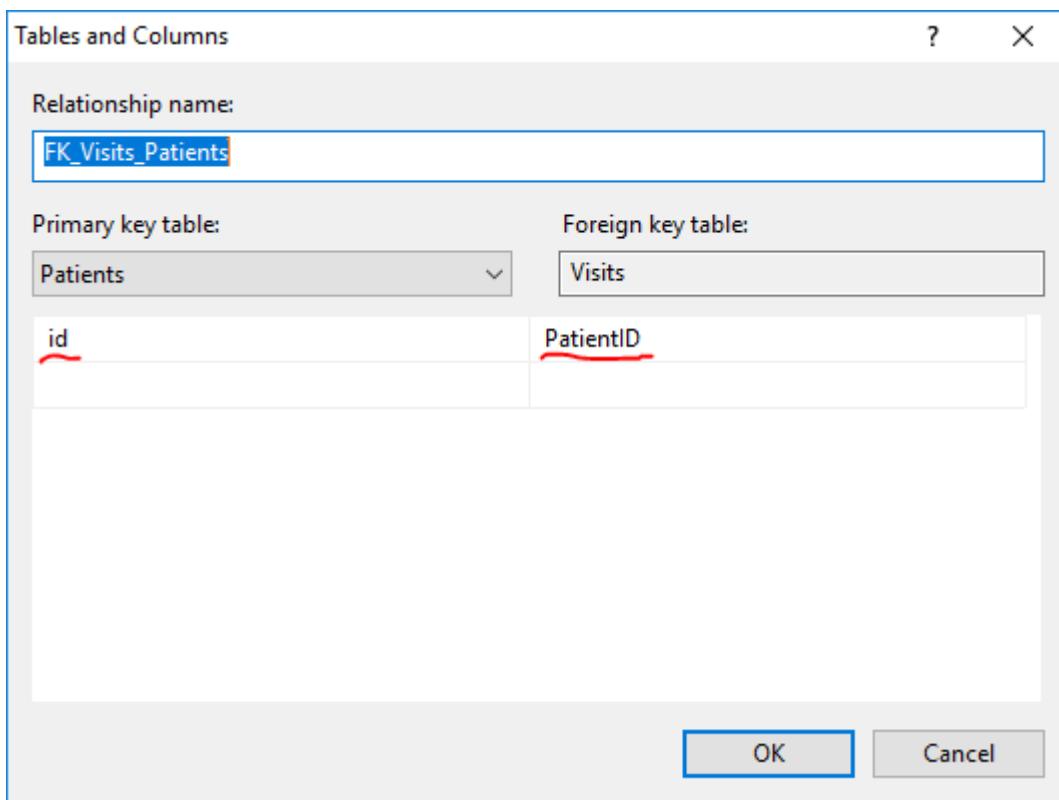




اضغط نعم ثم أضف الجداول المراد ربطها مع بعضها:

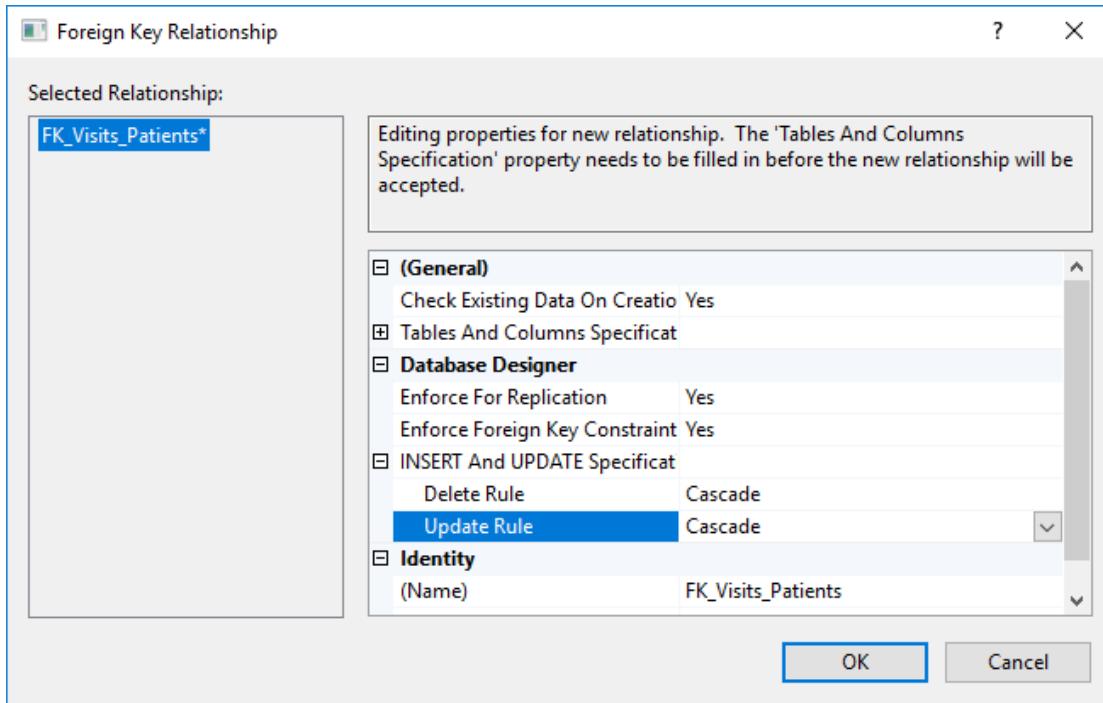


لا تنسَ ضبط الحقوق الرئيسية على ..PrimaryKey
اسحب الحقل id من الجدول Patients إلى PatientID في Visits وتأكد أن الحقوق المطلوبة تم ربطها معاً:



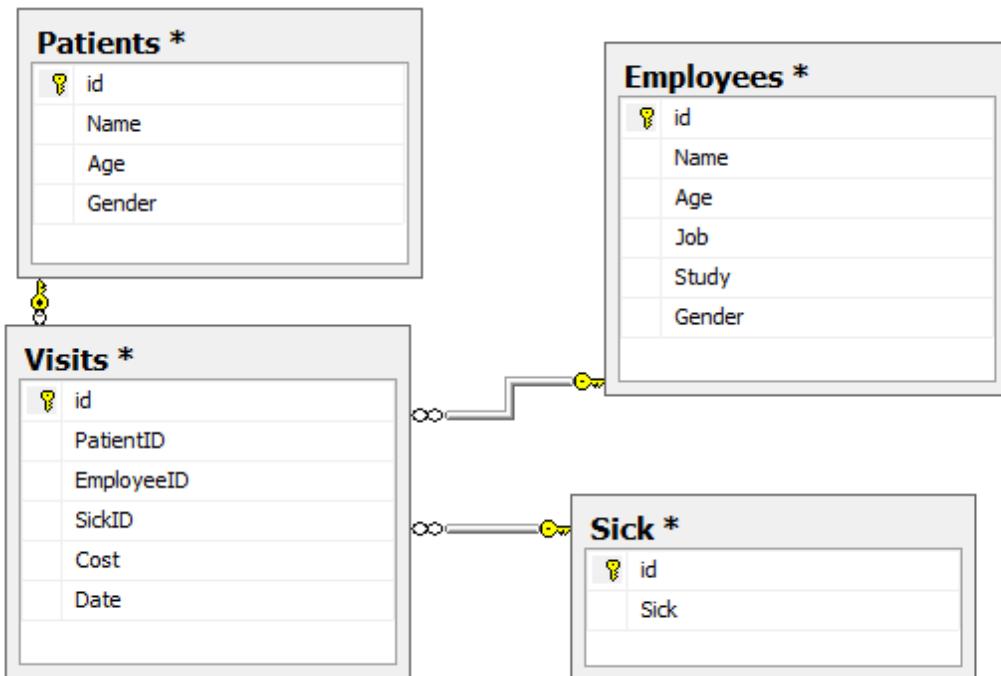


اضغط موافق لظهور لك النافذة التالية:



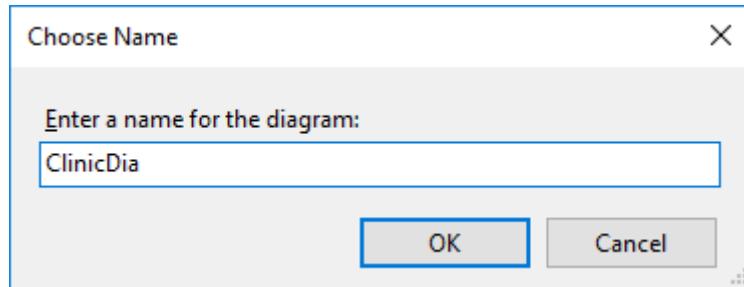
غير الخيار Delete و Update كما هو موضح بالصورة، حتى إذا ما تم حذف أحد الأمراض مثلاً، فإن جميع البيانات المتعلقة به ستتحذف.

كرر العملية للجدولين الآخرين ليصبح المخطط كما يلي:

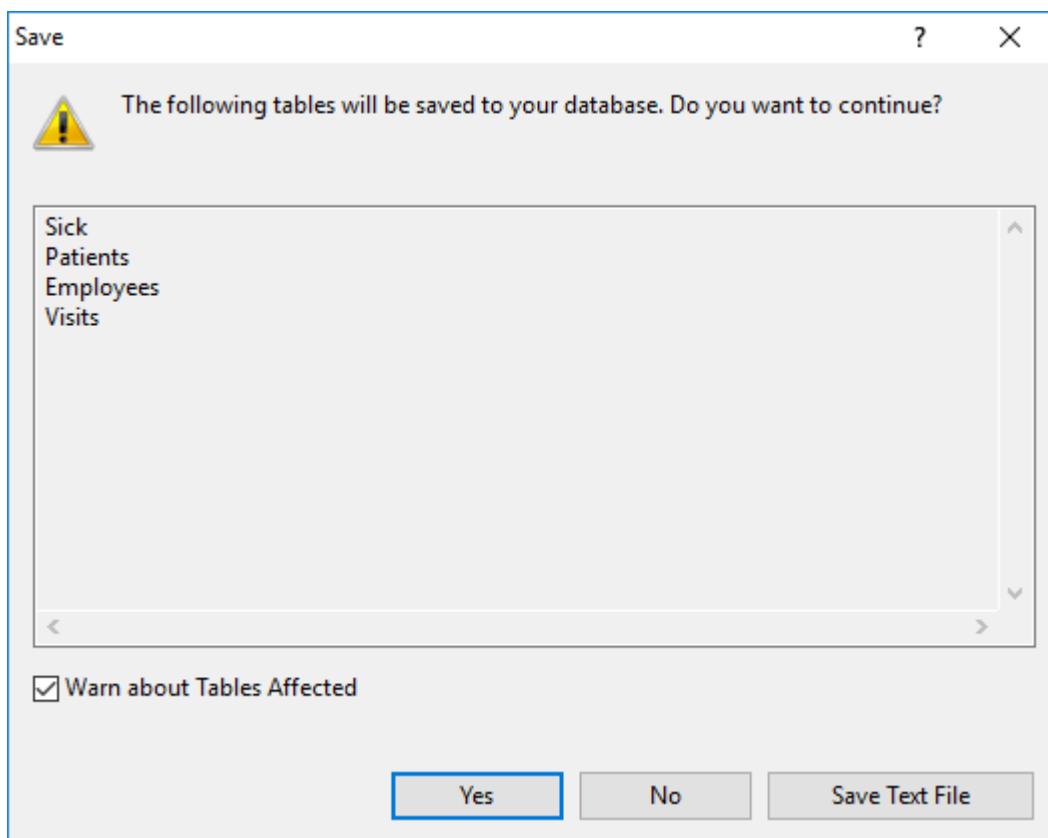




احفظ المخطط باسم ما ثم أضف بعض البيانات:



وافق على حفظ التغييرات على الجداول السابقة:



أصبح لديك المخطط التالي:





قم بتبينه البيانات على ذوقك:

	id	Name	Age	Job	Study	Gender
1	أحمد	37	طبيب أسنان	طب أسنان	ذكر	
2	محمد	49	طبيب أعصاب	طب بشرى	ذكر	
3	أسماء	43	طبيبة أطفال	طب بشرى	أنثى	
4	محمود	59	طبيب جلدية	طب بشرى	ذكر	
5	خالد	38	صيدلى	صيدلة	ذكر	
6	سمة	46	ممرضة	معهد تمريض	أنثى	
►*	NULL	NULL	NULL	NULL	NULL	NULL

	id	Name	Age	Gender
1	علي	37	ذكر	
2	حسان	61	ذكر	
3	نهى	58	أنثى	
4	مهند	31	ذكر	
►*	NULL	NULL	NULL	NULL

	id	Sick
1		أعصاب
2		جلدية
3		أسنان
4		أطفال
►*	NULL	NULL

أما بالنسبة للزيارات فهي كما يلي:

	id	PatientID	EmployeeID	SickID	Cost	Date
1	1	4	2	1500	2018-09-28 18:22:33.190	
►*	NULL	NULL	NULL	NULL	NULL	NULL

حيث أننا قد أضفنا زيارة واحدة فيها: رقم الزيارة هو 1، ورقم المريض هو 1 (يعني علي)، والموظف رقمه 4 (يعني محمود طبيب الجلدية)، ورقم المرض هو 2 (يعني مرض الجلدية)، والمبلغ المدفوع هو 1500، وتاريخ الزيارة هو 2018-09-28 18:22:33.190. ركز في المحتويات جيداً وافهمها..

بإمكانك إضافة الزيارة من خلال محرر الجدول كما في الجداول السابقة أو من خلال الأمر الذي ستضيف الزيارات والمرضى والموظفين والأمراض من خلاله:

```
insert into Visits (id,PatientID,EmployeeID,SickID,Cost,Date)
values(1,1,4,2,1500,GETDATE())
```



بنفس الطريقة قم بإضافة زيارات مختلفة (لا تنس إدخال الطبيب المناسب لمرض المريض):

SQLQuery2.sql - E...b (ENG27\Hp (56))*							ENG27.clinicdb - dbo.Visits	ENG27.clinicdb - dbo.Sick	ENG27.clinicdb - dbo.Patients
	id	PatientID	EmployeeID	SickID	Cost	Date			
	1	1	4	2	1500	2018-09-28 18:22:33.190			
	2	2	2	1	500	2018-09-28 18:30:07.627			
	3	4	2	1	500	2018-09-28 18:30:49.577			
	4	1	1	3	700	2018-09-28 18:31:32.390			
	5	3	4	2	700	2018-09-28 18:32:11.250			
►*	NULL	NULL	NULL	NULL	NULL	NULL			

الآن لنأت للأمثلة:

- للحصول على الموظفين الذين قابلوا المرضى وفق الزيارات المسجلة:

```
select *
from Employees
where id in (select EmployeeID from Visits )
```

والتي تعني قم بعرض جميع الحقول من جدول الموظفين والذين أرقامهم موجودة عند عرض حقل أرقام الموظفين من جدول الزيارات.

وبالمثل يمكن الحصول على أسماء المرضى الذين لهم أسماء ضمن جدول الزيارات. والتي سنستعرضها بصيغة أخرى:

```
select *
from Patients
where exists (select id from Visits where id = Visits.id)
```

- لعرض تفاصيل عن المرضى وأطبائهم وتاريخ الزيارات، مرتبة وفق آخر تاريخ زيارة:

```
select Patients.Name ,Patients.Age ,Employees.Name ,Visits.Cost ,Visits.Date
from Patients ,Employees ,Visits
where Visits.PatientID = Patients.id
and Visits.EmployeeID = Employees.id
order by visits.date desc
```



- لعرض المرضى الذين فاتورتهم أقل من 1000:

```
select Patients.Name ,Visits.Cost  
from Patients ,Employees ,Visits  
where Visits.PatientID = Patients.id  
and Visits.EmployeeID = Employees.id  
and Cost <1000
```

- لعرض مرضى مرض ما ول يكن الجلدية:

```
select Patients.Name, Patients.Age  
from Patients,Visits ,Sick  
where Visits.SickID = 2  
and patients.id = PatientID
```

- لعرض المثال السابق بتفاصيل أكثر:

```
select Patients.Name, Patients.Age, sick.Sick  
from Patients,Visits , Sick  
where Visits.SickID = 2  
and patients.id = PatientID  
and sick.id = SickID
```

لاحظ أنه عليك وضع شروط إضافية عند التعامل مع جداول جديدة، وذلك حتى لا يتم عرض جميع بيانات الجدول.

طبقة الوصول للبيانات DAL

طبقة الوصول للبيانات أو Data Access Layer هي طبقة من برنامج كمبيوتر توفر وصولاً مبسطاً إلى البيانات المخزنة في تخزين دائم من نوع ما، مثل قاعدة بيانات مرتبطة بكائنات معينة¹. بمعنى آخر فإن هذا المفهوم مشابه لفكرة مكتبات DLL أو الفئات CLASSES والتي يتم إنشاء طرقها وخصائصها، وإخفاءها عن المستثمرين - المبرمجين - بحيث تتركهم يصلون لخصائص وطرق معينة وتمنعهم من أخرى. هنا غايتك ليس منع الوصول - مع أنه من الممكن أن يكون كذلك - بقدر ما هي تنظيمًا للأ Kodaw.

من جهة أخرى، فإن استخدام الفئات - أو الطرق بشكل خاص - يجعل من صيانة البرنامج أمراً أسهل وأكثر كفاءة، فكما رأيت في أمثلة هذا الفصل

¹ المصدر: ويكيبيديا – Data access layer



قمنا بكتابه أكواد بشكل متكرر أكثر من مرة في الكود ذاته مثل كود الاتصال بقاعدة البيانات أو قطع الاتصال. أما عند استخدام الطرق فإن حدوث خطأ برمجي يصبح من السهل اقتناصه وإصلاحه فيما لو لم تستخدم الطرق فعليك تعديل جميع الأكواد التي من المحتمل أن يظهر فيها ذات الخطأ الذي ظهر في أحد الأكواد.

والأكواد التالية هي أكواد جاهزة يمكنك استخدامها في برنامجك للحصول على نتائج أفضل:

الاتصال وقطع الاتصال بقاعدة البيانات

ضع الإجراءين في برنامجك واستدعهما عندما ترغب بفتح أو إغلاق الاتصال، حيث يقومان بالتحقق من حالته أولاً تجنباً لرسائل الخطأ.

```
public void OpenConnection()
{
    connection.ConnectionString = connectionString;
    if (connection.State.ToString() != "Open")
        connection.Open();
}
// نص الاتصال بقاعدة البيانات هو متغير نصي عام على كافة المشروع
// ويحمل معلومات عن قاعدة البيانات المراد الاتصال بها
public void CloseConnection()
{
    if (connection.State.ToString() == "Open")
        connection.Close();
}
```

حذف البيانات

عملية الحذف من الممكن أن تكون من خلال حذف المريض بمعرفة رقم id الخاص به أو اسمه أو عمره مع تفاصيل أخرى مثل عمره و الجنسه أو عمره ومرضه أو جميع الأعمار الأكبر من أو الأصغر من أو التي تساوي عمر معين، أو معلومات أخرى بإمكانك إضافتها بنفس المبدأ.

```
حذف بحسب الرقم الشخصي//
public void DeletePatient(int id)
{
    string sql = string.Format("Delete from Patients where ID = {0}",id);
    using (SqlCommand command = new SqlCommand(sql, connection))
    {
        try{
            command.ExecuteNonQuery();
        }
```



```

        }
    catch (SqlException ex) {
        Exception error = new Exception("some error occurs: ", ex);
        throw error;
    }
}
// حذف بحسب الاسم
public void DeletePatient(string name)
{
    string sql = string.Format("Delete from Patients where [Name] =
'{0}'", name);
    using (command = new SqlCommand(sql, connection))
    {
        try{
            command.ExecuteNonQuery();
        }
        catch (SqlException ex) {
            Exception error = new Exception("some error occurs: ", ex);
            throw error;
        }
    }
}
// ضبط حالات العمر
enum deletecondition
{
    morethan,
    lessthan,
    equal
}
// حذف بحسب العمر
public void DeletePatient(int age, deletecondition delcondition)
{
    string sql = "";
    if (delcondition == deletecondition.morethan)
        sql = string.Format("Delete from Patients where age > {0}", age);
    else if (delcondition == deletecondition.lessthan)
        sql = string.Format("Delete from Patients where age < {0}", age);
    else
        sql = string.Format("Delete from Patients where age = {0}", age);
    using (command = new SqlCommand(sql, connection))
    {
        try
        {
            command.ExecuteNonQuery();
        }
        catch (SqlException ex)
    }
}

```



```
{  
    Exception error = new Exception("some error occurs: ", ex);  
    throw error;  
}  
}  
}  
}  
}  
  
// حذف بحسب العمر والمرض  
public void DeletePatient(int age, deletecondition delcondition, string sick)  
{  
    string sql = "";  
    if (delcondition == deletecondition.morethan)  
        sql = string.Format("Delete from Patients where age > {0}", age);  
    else if (delcondition == deletecondition.lessthan)  
        sql = string.Format("Delete from Patients where age < {0}", age);  
    else  
        sql = string.Format("Delete from Patients where age = {0}", age);  
    sql += string.Format(" and sick = '{0}'", sick);  
    using (command = new SqlCommand(sql, connection))  
{  
        try  
        {  
            command.ExecuteNonQuery();  
        }  
        catch (SqlException ex)  
        {  
            Exception error = new Exception("some error occurs: ", ex);  
            throw error;  
        }  
    }  
}  
}  
  
// حذف بحسب العمر والجنس  
public void DeletePatient(int age, deletecondition delcondition, string gender)  
{  
    string sql = "";  
    if (delcondition == deletecondition.morethan)  
        sql = string.Format("Delete from Patients where age > {0}", age);  
    else if (delcondition == deletecondition.lessthan)  
        sql = string.Format("Delete from Patients where age < {0}", age);  
    else  
        sql = string.Format("Delete from Patients where age = {0}", age);  
    sql += string.Format(" and gender = '{0}'", gender);  
    using (command = new SqlCommand(sql, connection))  
{  
        try  
        {  
            command.ExecuteNonQuery();  
        }  
    }  
}
```



```

        catch (SqlException ex)
    {
        Exception error = new Exception("some error occurs: ", ex);
        throw error;
    }
}
// حذف بحسب العمر والمرض والجنس
public void DeletePatient(int age, deletecondition delcondition, string sick,
string gender)
{
    string sql = "";
    if (delcondition == deletecondition.morethan)
        sql = string.Format("Delete from Patients where age > {0}", age);
    else if (delcondition == deletecondition.lessthan)
        sql = string.Format("Delete from Patients where age < {0}", age);
    else
        sql = string.Format("Delete from Patients where age = {0}", age);
    sql += string.Format(" and sick = '{0}'", sick);
    sql += string.Format(" and gender = '{0}'", gender);
    using (command = new SqlCommand(sql, connection))
    {
        try
        {
            command.ExecuteNonQuery();
        }
        catch (SqlException ex)
        {
            Exception error = new Exception("some error occurs: ", ex);
            throw error;
        }
    }
}

```

طبعاً يجب أن تفتح الاتصال قبل استدعاء إحدى هذه الإجراءات.
وهكذا بإمكانك تكوين توابع لكافة العمليات التي تحتاجها - الأساسية
وغيرها - وتضمينها ضمن فئات لحفظها بملفات DLL للاستفادة منها في
كافة برامجك.





الفصل الثالث عشر - قواعد بيانات ACCESS

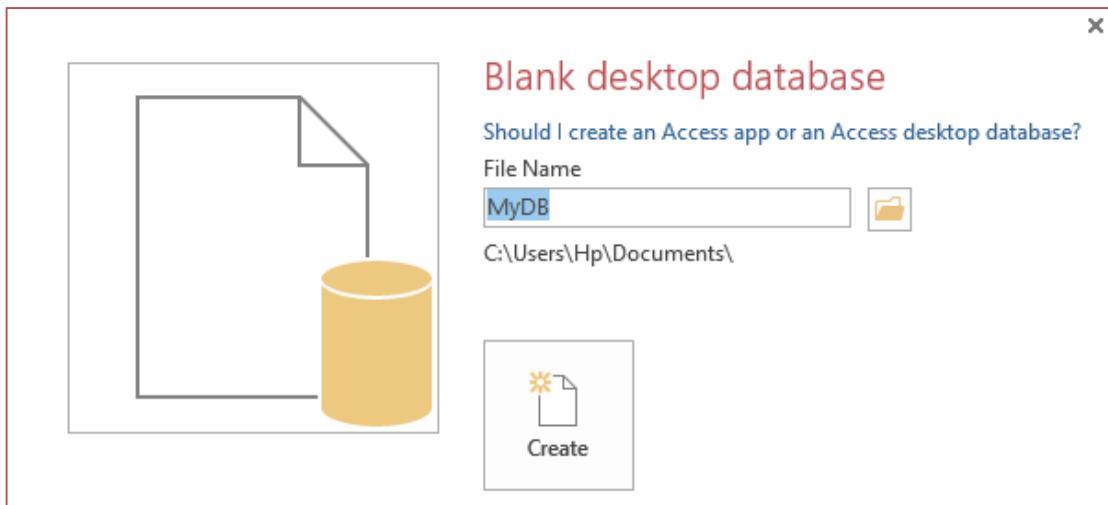
تحظى قواعد البيانات من نوع أكسس باستخدام واسع في عالم ويندوز، فهي لا تتطلب منك تثبيت برامج تشغيل لتتمكن من التعامل مع قواعد البيانات التي تعمل في بيئتها. ففي قواعد بيانات SQL مثلًا يجب أن يوجد في جهاز العميل نسخة من SQL Server لتسهيل استخدام قاعدة البيانات في البرنامج، وكما تعلم فإن تثبيت SQL Server يحتاج دقة في اختيار المتغيرات والغلطة بكفرة كما يقال. وبال مقابل فإن هذا لا يعني أن قواعد بيانات ACCESS لا تتطلب وجود نسخة أكسس في جهاز العميل، لكن فكر معي، من هنا لا يوجد بجهازه نسخة من Microsoft Office ؟؟؟ لذلك قلت "هي - قواعد بيانات أكسس - لا تتطلب منك تثبيت برنامج تشغيل" ولم أقل "لا تتطلب وجود برنامج تشغيل".

الخلاصة: لتعامل مع قاعدة بيانات من أي نوع كانت يجب أن يوجد لديك مشغلها، أو بالأحرى يجب أن يوجد لدى جهاز العميل مشغلها.



إنشاء قاعدة بيانات

خطوة أولى افتح MS Access، ثم أنشئ قاعدة بيانات جديدة:



وتحت الملف سيعمل عنوان الملف:

MyDB : Database- C:\Users\Hp\Documents\MyDB.accdb (Access 2007 - 2016 file format) - Access

وهذا يعني أنك تتعامل مع ملف أكسس ذو امتداد accdb وهو امتداد الملفات الحديثة لقواعد البيانات، حيث أن قواعد البيانات في ماضي كانت ذات امتداد mdb. وفي هذا الكتاب سنتعامل مع كلا النوعين، والفارق بينهما كما سنرى هو كائن الاتصال فقط.

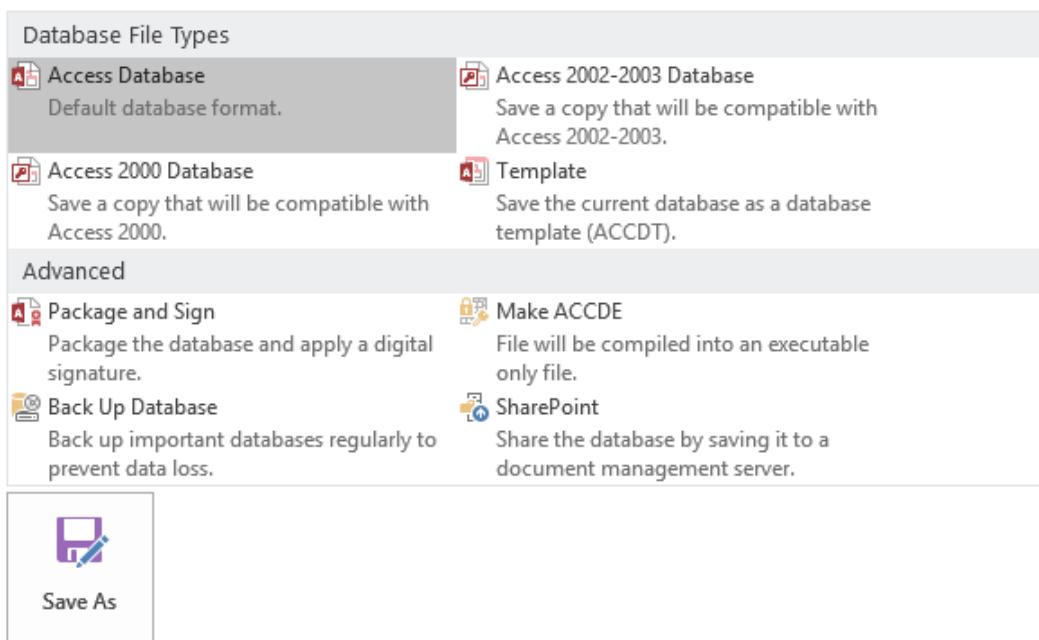
من التبويبة أنشئ جدولًا جديدًا من الأمر Table، ثم من قائمة الجداول انقر بالزر الأيمن على الجدول المنشأ ثم Design View، وسمه Students. ثم أنشئ الحقول التالية:

	Field Name	Data Type
ID	Number	
FirstName	Short Text	
LastName	Short Text	
Age	Number	
Class	Short Text	

احفظ الجدول ثم احفظ قاعدة البيانات. اضغط حفظ باسم لحفظ قاعدة البيانات بالامتداد القديم:



Save Database As



احفظ قاعدة البيانات بصيغة Access 2002-2003 Database وهي .mdb،
والآن أصبح لديك الملفين التاليين:

	٢٠١٨/١٠/٣٩ ٢٠:٤٣	Microsoft Access ...	416 KB
	٢٠١٨/١٠/٣٩ ٢٠:٤٤	Microsoft Access ...	240 KB

الأول ذا امتداد accdb وهو الامتداد الحديث أما الآخر فهو .mdb

إنشاء اتصال مع قاعدة البيانات

يتم الاتصال بقاعدة بيانات Access بواسطة الفئة OleDbConnection، لذلك استنسخ منها ما وليكن باسم connect (لتبسيط الأمور وتوضيح الاختلاف بين قواعد بيانات SQL وسنسمي Access SQL الكائنات في هذا الفصل بنفس أسماء الكائنات التي استخدمناها في الفصل السابق).

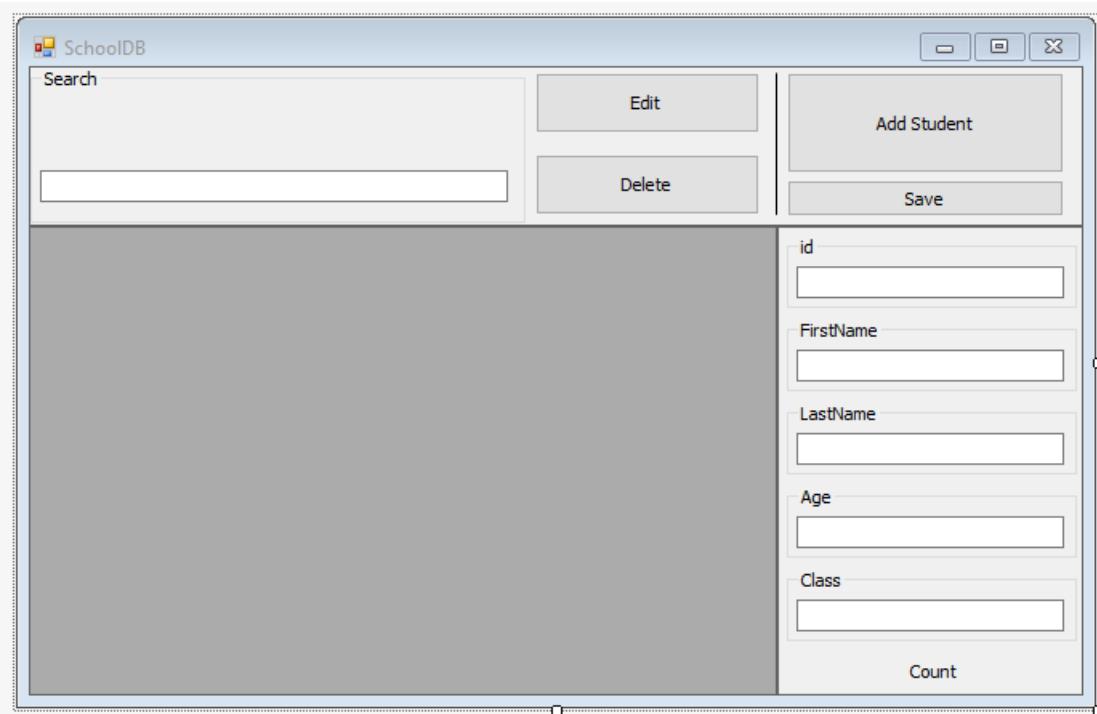
للاتصال بقاعدة بيانات mdb استخدم المزود jet.OleDb.4.0 كما يلي:

```
في مكان ما تصل إليه جميع الإجراءات//  
static string sql = @"Provider=Microsoft.jet.OLEDB.4.0;Data Source =  
C:\MyDB.mdb;Jet OLEDB:Database Password='';Persist Security Info=True";  
OleDbConnection connection;  
في أحد إجراءات النموذج وليكن إجراء تحميل النموذج//  
connection = new OleDbConnection(sql);
```



أما إذا كانت قاعدة بياناتك accdb فاستخدم ace.Oledb.12.0، ومن أجل كلمة سر في حال وجودها، فتوضع أمام Database Password في العبارة النصية مابين تنصيص أحادي '' (سيتم الحديث عن كيفية ضبط كلمة سر في الأكواد التالية).

سنقوم بإنشاء مثال شامل يحوي الأكواد الأساسية المستخدمة في أي قاعدة بيانات. أنشئ مشروعًا جديداً في الفيجوال ستوديو وصمم الواجهة التالية:



حيث أن مكونات النافذة هي ثلاثة أدوات panel وأربعة أزرار وستة صناديق تجميع وستة نصوص وأداة عرض بيانات واحدة.

كما أن كلا من الزر Save وصندوق التجميع المسمى id غير هي أدوات غير مفعولة (Enabled = False)، وصندوق التجميع المسمى بـ Search مضبوط على الخاصية Dock = Left.

ومن أجل ضمان بقاء الأدوات في أماكنها عند تغيير حجم النافذة اضبط الخاصية Anchor لجميع العناصر الواقعة في اللائحة العليا على Top, Right، عدا صندوق التجميع.



استخدم الكود التالي خارج الإجراءات (لتتمكن جميع الإجراءات من الوصول إليه):

```
static string accessDB = @"Provider=Microsoft.jet.OLEDB.4.0; Data Source =
" + Application.StartupPath + @"\MyDB.mdb; Jet OLEDB:Database
Password=''; Persist Security Info=True;"
```

كائن لإجراء الاتصال بقاعدة البيانات //
 كائن لإنشاء أوامر قواعد البيانات //
 كائن لربط كائن الأوامر مع كائنات أخرى //
 كائن لقراءة ناتج تنفيذ مائن إنشاء الأوامر //
 كائن يمثل جدول ضمن قاعدة بيانات //

تهيئة البرنامج: وفيه يتم تهيئة المكونات ضمن التابع البنّاء للنافذة ثم إجراء الاتصال مع قاعدة البيانات الموصفة بالمتغير النصي accessDB ثم تحميل بيانات قاعدة البيانات إلى أداة عرض البيانات.

```
public Form1()
{
    InitializeComponent();
    connection = new OleDbConnection(accessDB);
    dataGridView1.DataSource = LoadData();
}
```

تحميل البيانات إلى أداة عرض البيانات: وفيه يتم فتح الاتصال مع قاعدة البيانات ثم استنساخ كائن جديد يمثل جدولًا، وتنفيذ أمر تحميل جميع حقول قاعدة البيانات عبر الاتصال connection، ثم الربط بين كائن الأمر وكائن الجدول عبر الكائن adapter، وأخيراً إغلاق الاتصال وإرجاع الجدول كقيمة للطريقة LoadData.

```
public DataTable LoadData()
{
    تحميل البيانات إلى أداة عرض البيانات //
    connection.Open();
    data = new DataTable();
    string load = "select * from students";
    command = new OleDbCommand(load, connection);
    adapter = new OleDbDataAdapter(command);
    adapter.Fill(data); connection.Close(); return data;
}
```



ولضمان عدم تكرار الأرقام الشخصية id استخدم الكود التالي والذي يقوم بفتح الاتصال بقاعدة البيانات وإنشاء أمر يقوم باستدعاء جميع عناصر الحقل id، ثم قراءتها عبر الكائن reader، ثم إضافة جميع محتويات الحقل إلى لائحة بعد التصريح عنها. وأخيراً نغلق الاتصال والقارئ، ونتحقق من قيمة الرقم الممرر ك وسيط للتابع isID وذلك عبر بنية شرط، فإذا كان موجوداً أعاد القيمة true وإلا¹ false.

```
private bool isID(int n)
{
    connection.Open();
    command = new OleDbCommand("select id from students", connection);
    reader = command.ExecuteReader();
    List<string> id = new List<string>();
    while (reader.Read())
        id.Add(reader["id"].ToString());

    reader.Close();
    connection.Close();

    if (id.Contains(n.ToString()))
        return true;
    else
        return false;
}
```

تغيير حجم النافذة، وهو واضح ولا يحتاج شرح:

```
private void Form1_Resize(object sender, EventArgs e)
{
    groupBox6.Width = Edit.Left - 10;
    Search.Width = groupBox6.Width - 20;
}
```

إضافة سجل جديد

سنقوم بدايةًً عبر الضغط على زر Add بإفراغ صناديق النصوص من محتواها، ثم إضافة سجل جديد وذلك بعد التحقق من رقم السجل الجديد

¹ الغاية من التابع isID هو معرفة هل العدد صالح لأن يكون رقم id أم لا. فكما هو معرف فرقم id يجب ألا يتكرر على الإطلاق في قاعدة البيانات وذلك لأنه يميز عناصر السجلات عن بعضها في قاعدة البيانات. عند حذف أحد السجلات وفي حال لم يكن آخر سجل، سيبقى رقمه id حالياً لا يحمله أي سجل، وقد يسبب لك هذا بعض المشاكل، لذلك وعند إنشاء سجلات جديدة سنضعها مكان السجلات القديمة إن أمكن.



باستخدام التابع `isID`, ثم بعدها وبالضغط على `Save` يتم إضافة السجل إلى قاعدة البيانات.

```
private void Add_Click(object sender, EventArgs e)
{
    Add.Enabled = false;
    Save.Enabled = true;
    textBox2.Focus(); //بالإدخال
    int n = dataGridView1.Rows.Count;
    for (int i = 1; i <= dataGridView1.Rows.Count + 1; ++i)
        if (!isID(i))
    {
        n = i;
        break;
    }
    textBox1.Text = Convert.ToString(n);
    textBox2.Text = "";
    textBox3.Text = "";
    textBox4.Text = "";
    textBox5.Text = "";
}

private void Save_Click(object sender, EventArgs e)
{
    try
    {
        connection.Open();

        string insert = "insert into students (id,FirstName,LastName,Age,Class) values (@id,@FirstName,@LastName,@Age,@Class)";
        command = new OleDbCommand(insert, connection);
        command.Parameters.AddWithValue("@id", textBox1.Text);
        command.Parameters.AddWithValue("@FirstName", textBox2.Text);
        command.Parameters.AddWithValue("@LastName", textBox3.Text);
        command.Parameters.AddWithValue("@Age", textBox4.Text);
        command.Parameters.AddWithValue("@Class", textBox5.Text);

        command.ExecuteNonQuery();
        connection.Close();

        dataGridView1.DataSource = LoadData();

        Add.Enabled = true ;
        Save.Enabled = false ;
    }
}
```



```

catch
{
    // هنا صح شيئاً ما يستقبل الخطأ، رسالة أو غيرها
}
finally
{
    // عند تنفيذ الكود أو وجود خطأ يجب إغلاق الاتصال في النهاية
    if (connection.State.ToString() == "Open")
        connection.Close();
}
}

```

ملاحظة

- عند استخدام كلمات محفوظة في MS Access كأسماء حقول كالكلمة أو name count وغيرها، هذا إن استطعت استخدامها أصلاً، ضعها مابين قوسين متقطعين [] لتمكن من استخدامها بشكل طبيعي. مثلاً [count].
- القيم التي تبعها بعد الكلمة Values والمسبقة بإشارة @ ليس من الضروري أن تكون مماثلة لأسماء الحقول التي تمثلها ولكن يجب أن تكون مماثلة للكلمات التي ستوضع بالأسطر التالية.

حذف السجلات

ويكون بالكود التالي:

```

private void Delete_Click(object sender, EventArgs e)
{
    DialogResult M = MessageBox.Show(" حذف تريد هل " + textBox2.Text +
"?", "تأكيد الحذف", MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1, MessageBoxOptions.RightAlign);
    if (M == System.Windows.Forms.DialogResult.Yes)
        try
        {
            connection.Open();
            string delete = "delete from students where id = " +
dataGridView1.CurrentRow.Cells[0].Value.ToString() + "";
            command = new OleDbCommand(delete, connection);
            command.ExecuteNonQuery(); connection.Close();
            dataGridView1.DataSource = LoadData();
        }
        catch (OleDbException ex)
        { MessageBox.Show(ex.Message); }
}

```



```
finally
{
    if (connection.State.ToString() == "Open") connection.Close();
}
}
```

تعديل السجلات

لتعديل سجل ما يجب أن تتأكد أولاً أن عمر الطالب - في مثالنا - أكبر من الصفر، وهذا ما كان علينا إضافته إلى كود إضافة طالب جديد إلى سجلات قاعدة البيانات ولكنني نسيت ذلك، ولا أرغب بتعديل الصفحات السابقة وإعادة تنسيق الصفحات مع بعضها لذلك عدل الكود عندك ليشمل التحقق من أن عمر الطالب المدخل موجب تماماً.

ملاحظة

- في الواقع هناك الكثير من الأفكار الممكن تطبيقها على غرار التأكد من أن العمر موجب، فمثلاً بإمكانك تخمين السنة الدراسية للطالب اعتماداً على العمر (مثلاً من أجل عمر 6 سنوات يكون الطالب في الصف الأول، وهذا لبقية الأعمار).
- من غير المنطقي أن يكون الطالب عمره 30 سنة مثلاً -_- (لذلك من الممكن قبول أعمار محددة).

```
private void Edit_Click(object sender, EventArgs e)
{
    if (Convert.ToInt32(textBox4.Text) >= 0)
        try
        {
            connection.Open();
            string update = "update students set FirstName = @FirstName, LastName = @LastName, Age=@Age, Class=@Class where id = "
+ dataGridView1.CurrentRow.Cells[0].Value.ToString() + "";
            command = new OleDbCommand(update, connection);
            command.Parameters.AddWithValue("@FirstName", textBox2.Text);
            command.Parameters.AddWithValue("@LastName", textBox3.Text);
            command.Parameters.AddWithValue("@Age", textBox4.Text);
            command.Parameters.AddWithValue("@Class", textBox5.Text);

            command.ExecuteNonQuery();
            connection.Close();

            dataGridView1.DataSource = LoadData();
        }
}
```



```

catch (OleDbException ex)
{ MessageBox.Show(ex.Message); }
finally
{
    if (connection.State.ToString() == "Open")
        connection.Close();
}
else
{
    MessageBox.Show("يجب أن يكون العمر موجبا");
}
}

```

سهولة الوصول إلى البيانات

عند الضغط على إحدى خلايا الأداة dataGridView فإنه يتم عرض عناصر السجل المحدد في صناديق الإدخال لتوضيحها وسهولة تعديلها:

```

private void dataGridView1_SelectionChanged(object sender, EventArgs e)
{
    try
    {
        textBox1.Text = dataGridView1.CurrentRow.Cells[0].Value.ToString();
        textBox2.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();
        textBox3.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString();
        textBox4.Text = dataGridView1.CurrentRow.Cells[3].Value.ToString();
        textBox5.Text = dataGridView1.CurrentRow.Cells[4].Value.ToString();

        label1.Text = dataGridView1.CurrentCell.RowIndex + 1 + " من " +
data.Rows.Count;
    }
    catch { }
}

```

ولجعل التركيز على صندوق النص الممثل للخلية المحددة (الاسم الأول أو الأخير أو العمر أو الصف):

```

private void dataGridView1_CellEnter(object sender,
DataGridViewCellEventArgs e)
{
    switch (e.ColumnIndex )
    {
        case 0:
            textBox1.Focus();
    }
}

```



```

        break;
    case 1:
        textBox2.Focus();
        break;
    case 2:
        textBox3.Focus();
        break;
    case 3:
        textBox4.Focus();
        break;
    case 4:
        textBox5.Focus();
        break;
    }
}

```

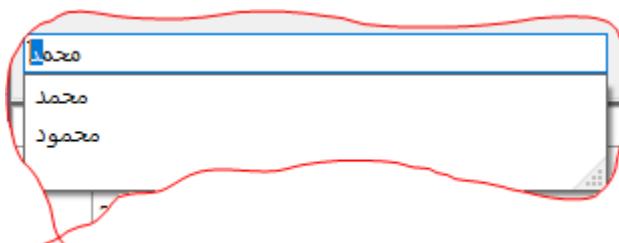
ولتسهيل الإدخال والبحث وللحصول على صناديق نصوص مثل صندوق بحث غوغل، استخدم الكود التالي:

```

يفضل وضع الأسطر الأربعية التالية في كود التمهئة//
Search.AutoCompleteMode = AutoCompleteMode.SuggestAppend;
Search.AutoCompleteSource = AutoCompleteSource.CustomSource;
textBox5.AutoCompleteMode = AutoCompleteMode.SuggestAppend;
textBox5.AutoCompleteSource = AutoCompleteSource.CustomSource;
//ضع الأسطر التالية في التابع
//LoadData
//وذلك قبل سطر الإرجاع
foreach (DataRow row in data.Rows)
{
    //comboBox1.Items.Add(row.ItemArray[1]);
    Search.AutoCompleteCustomSource.Add(row.ItemArray[1].ToString());
    Search.AutoCompleteCustomSource.Add(row.ItemArray[2].ToString());
    textBox5.AutoCompleteCustomSource.Add(row.ItemArray[4].ToString());
}

```

لاحظ صندوق البحث بعد إضافة الاقتراحات إليه:





بهذه الحلقة `foreach (DataRow row in data.Rows)` بإمكانك الحصول على أي سطر من أسطر قاعدة البيانات، وبالمثل لو تم استنساخ `column` من `data.Columns` بإمكانك الحصول على جميع عناصر عمود ما.

لذلك فمن الممكن الحصول على ذات نتيجة التابع `isID` لو استخدمنا الحلقات بدلاً من اللوائح والبحث بداخلها..

والإلغاء صناديق الإدخال وأزرار التعديل والحذف عندما لا يوجد سجلات،
استخدم الكود التالي:

```
private void dataGridView1_DataSourceChanged(object sender, EventArgs e)
{
    if (dataGridView1.Rows.Count > 0)
    {
        groupBox2.Enabled = true;
        groupBox3.Enabled = true;
        groupBox4.Enabled = true;
        groupBox5.Enabled = true;
        Edit.Enabled = true;
        Delete.Enabled = true;
    }
    else
    {
        groupBox2.Enabled = false ;
        groupBox3.Enabled = false;
        groupBox4.Enabled = false;
        groupBox5.Enabled = false;
        Edit.Enabled = false;
        Delete.Enabled = false;
    }
}
```

البحث عن السجلات

سنقوم بالبحث عن السجلات في صندوق الإدخال المخصص للبحث والمسمى برمجيا `Search`، وللحصول على جودة في الأداء سننفذ عملية البحث عند تغيير حدث `TextChanged`، بحيث يتم جلب القيم المشابهة.

```
private void Search_TextChanged(object sender, EventArgs e)
{
    connection.Open();
    DataTable data = new DataTable();
    string load = "select * from students where firstname+lastname+class like
```



```
'%" + textBox1.Text + "%'; //البحث
command = new OleDbCommand(load, connection);
adapter = new OleDbDataAdapter(command);
adapter.Fill(data);
connection.Close();
dataGridView1.DataSource = data;
}
```

ال코드 السابق سيبحث عن جميع السجلات التي تواافق فيها الحقول الاسم الأول أو الأخير أو الصف المحتوى المبحوث عنه، بمعنى أنه يمكن البحث عن الطالب باسمه الأول أو الأخير أو صفة.

تجميع البيانات

يتم عادة تجميع البيانات لإجراء مقارنات وإحصائيات معينة حيث يتم عرض السجلات منسوبة لبيانات حقل ما. ففي مثالنا يمكن عرض عدد الطلاب المسجلين بالصفوف حيث يتم عرض عدد الطلاب من أجل كل صف، أو عرض عدد الطلاب الذين لهم الاسم الفلاني، أو غيرها وغيرها من التطبيقات حسب الحاجة.

قم بإضافة زر لتجميع البيانات وزر لإعادة تحميل البيانات كاملة واستخدم الكود التالي:

```
private void Group_Click(object sender, EventArgs e)
{
    data = new DataTable();
    string load;
    load = "select count(age) as Age, class from students group by class";
    command = new OleDbCommand(load, connection);
    adapter = new OleDbDataAdapter(command);
    adapter.Fill(data);
    connection.Close();
    dataGridView1.DataSource = data;
    panel1.Enabled = false; panel2.Enabled = false;
    لا تنسى أنه علينا إلغاء تمكين المستخدم من تعديل السجلات أثناء التجميع //
}

private void Reload_Click(object sender, EventArgs e)
{
    dataGridView1.DataSource = null;
    connection.Open();
    data = new DataTable();
    string load = "select * from students";
```



```
command = new OleDbCommand(load, connection);
adapter = new OleDbDataAdapter(command);
adapter.Fill(data);
connection.Close();
dataGridView1.DataSource = data;
panel1.Enabled = true; panel2.Enabled = true;
}
```

فإذا كانت لدينا البيانات التالية:

	ID	FirstName	LastName	Age	Class
▶	2	معاذ	حمدى	10	خامس
	1	رامى	أحمد	12	خامس
	3	خالد	محمد	9	رابع
	4	خالد	أحمد	11	خامس
	5	محمد	سمير	9	رابع
	6	محمد	أحمد	12	سادس
	7	محمد	سعيد	8	تالت

فالنتيجة ستكون:

	Age	Class
▶	1	تالت
	3	خامس
	2	رابع
	1	سادس

كان يجب أن أسمّي العمود الأول [count] لكنني سيمته Age خطأً، لذلك تخيله بالاسم count من فضلك 😊. طبعاً لا تنسَ تغييره عندك.

أما إذا كان كود الاتصال:

```
string load = "select max(age) as Age, class from students group by class";
```

:-(悲 😢 😢 😢 😢) عندما سنحصل على أكبر عمر (هنا أيضًا لدي خطأ

	Age	Class
▶	8	تالت
	12	خامس
	9	رابع
	12	سادس



إظهار البيانات في قائمة عرض ListView

بإمكانك تقديم محتوى قاعدة بياناتك من خلال قائمة عرض، مما يعطيك تصميماً أكثر جودة من ذاك الذي تستخدم فيه أداة عرض البيانات.

أضف قائمة عرض ListView وغير طريقة عرض العناصر إلى تفاصيل MultiSelect، ثم فُعل الخيارين GridLines و FullRowSelect ثم ألغِ Details.

لاحظ النافذة التالية:

	id	First name	Last name	Age	Class
1	1	رامي	أحمد	12	خامس
2	2	معاذ	حمدي	10	خامس
3	3	خالد	محمود	9	رابع
4	4	خالد	أحمد	11	خامس
5	5	محمد	سمير	9	رابع
6	6	محمود	أحمد	12	سادس
7	7	محمد	سعيد	8	ثالث
8	8	خالد	حسين	11	رابع

لاحظ أنه باستخدام التصميم السابق باتت النافذة مرحلة أكثر للمستخدم، إذا رغب بإضافة طالب جديد ينتقل للتبوية New، وإلا فإنه يبقى في التبوية Operations لإجراء بعض العمليات الممكنة. بإمكانه البحث أيضاً ضمن صندوق النص الموضح بالشكل السابق. كما أن عملية ترتيب البيانات ممكنة بسهولة بالنسبة له. إليك بعض الأكواد التي ستحتاجها لإحياء النافذة السابقة:

```
using System;
using System.Windows.Forms;
using System.Data.OleDb;

namespace ListViewDB
{
    public partial class Form1 : Form
    {
        public partial class Form1 : Form
        {
            static string accessDB = @"Provider=Microsoft.jet.OLEDB.4.0;Data Source = " + Application.StartupPath + "\\MyDB.mdb;";

            static string id, OrderBy;
```



```
OleDbConnection connection;
OleDbCommand command;
OleDbDataReader reader;

static ListViewItem L;
public Form1()
{
    InitializeComponent();
    connection = new OleDbConnection(accessDB);
}

private void Form1_Load(object sender, EventArgs e)
{
    ShowData("id");
}

void ShowData(string Order)
{
    OrderBy = Order;
    listView1.Items.Clear();
    connection.Open();
    command = new OleDbCommand("select * from students order by "
+ OrderBy, connection);
    reader = command.ExecuteReader();

    while (reader.Read())
    {
        L = listView1.Items.Add(reader["id"].ToString());
        L.SubItems.Add(reader["firstname"].ToString());
        L.SubItems.Add(reader["lastname"].ToString());
        L.SubItems.Add(reader["age"].ToString());
        L.SubItems.Add(reader["class"].ToString());
    }
    connection.Close();
}

void ShowData()
{
    listView1.Items.Clear();
    connection.Open();
    command = new OleDbCommand("select * from students order by "
+ OrderBy, connection);
    reader = command.ExecuteReader();

    while (reader.Read())
    {
```



```
L = listView1.Items.Add(reader["id"].ToString());
L.SubItems.Add(reader["firstname"].ToString());
L.SubItems.Add(reader["lastname"].ToString());
L.SubItems.Add(reader["age"].ToString());
L.SubItems.Add(reader["class"].ToString());
}
connection.Close();
}

private void order_SelectedIndexChanged(object sender, EventArgs e)
{
    ShowData(order.Text);
}

private void delete_Click(object sender, EventArgs e)
{
    connection.Open();
    command = new OleDbCommand("delete from students where id =
" + id + "", connection);
    reader = command.ExecuteReader();
    connection.Close();
    ShowData();
}

private void listView1_SelectedIndexChanged(object sender, EventArgs e)
{
    foreach (ListViewItem L in listView1.Items)
        if (L.Selected)
        {
            id = L.SubItems[0].Text;
        }
    connection.Open();
    command = new OleDbCommand("select * from students where id =
" + id + "", connection);
    reader = command.ExecuteReader();
    reader.Read();
    textBox5.Text = reader[1].ToString();
    textBox4.Text = reader[2].ToString();
    numericUpDown2.Value = decimal.Parse(reader[3].ToString());
    comboBox1.Text = reader[4].ToString();
    connection.Close();
}
}
```

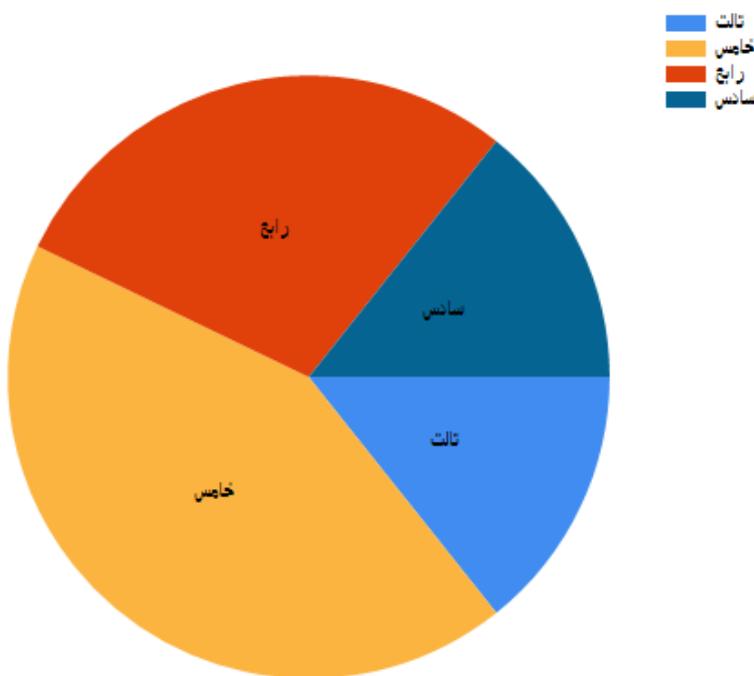


إظهار البيانات في مخططات

توفر لك C# إمكانية عرض بيانات وفق مخططات Charts وذلك عبر الأداة والتي توفر بإمكانات كبيرة تعطيك إمكانات إحصائية جيدة.

أضف أداة Chart إلى مشروعك ثم استخدم الكود التالي في أحد الأزرار:

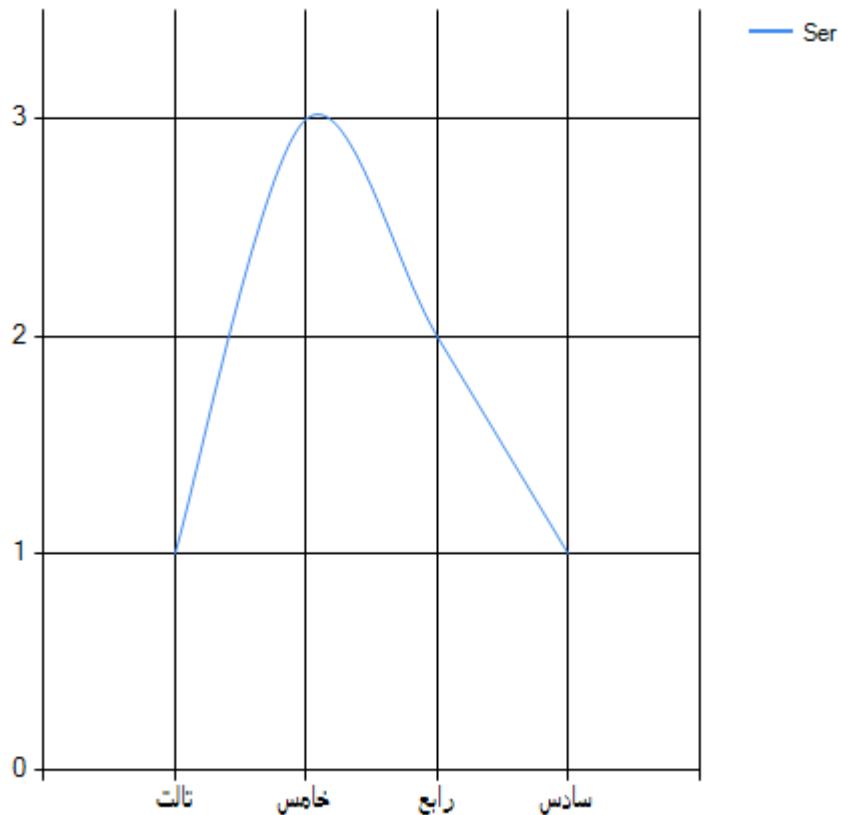
```
private void ShowChart_Click(object sender, EventArgs e)
{
    data = new DataTable();
    string load;
    load = "select count(age) as Age,class from students group by class";
    command = new OleDbCommand(load, connection);
    adapter = new OleDbDataAdapter(command);
    adapter.Fill(data);
    connection.Close();
    dataGridView1.DataSource = data;
    chart1.DataSource = data;
    chart1.Series.Clear();
    chart1.Series.Add("Ser");
    chart1.Series["Ser"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Pie;
    chart1.Series["Ser"].XValueMember = data.Columns[1].ToString();
    chart1.Series["Ser"].YValueMembers = data.Columns[0].ToString();
}
```





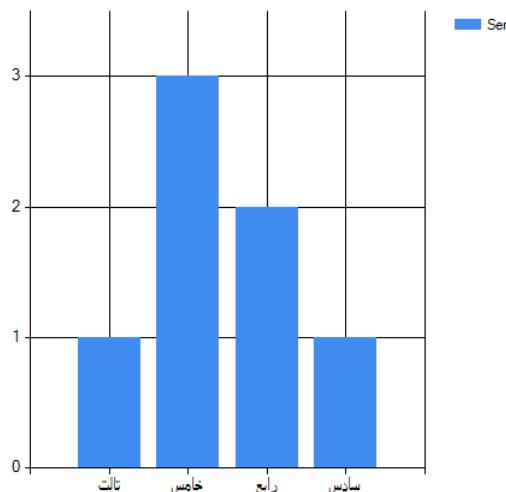
كما يمكن تغيير شكل المخطط بتغيير شكل السلسلة المدرسة:

```
chart1.Series["Ser"].ChartType =  
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;
```



أو بالشكل:

```
chart1.Series["Ser"].ChartType =  
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Column;
```





وغيرها الكثير..

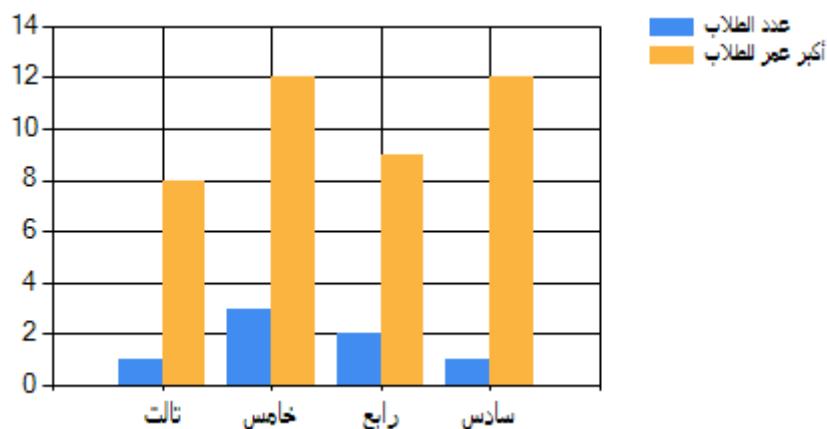
كما يمكنك إظهار سلسلتين معاً:

```
private void ShowChart_Click(object sender, EventArgs e)
{
    data = new DataTable();
    string load = "select count(age) as Age,max(age) as [Max],class from
    students group by class";
    command = new OleDbCommand(load, connection);
    adapter = new OleDbDataAdapter(command);
    adapter.Fill(data);
    connection.Close();
    dataGridView1.DataSource = data;
    chart1.DataSource = data;
    chart1.Series.Clear();

    chart1.Series.Add("عدد الطالب");
    chart1.Series["عدد الطالب"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Column;
    chart1.Series["عدد الطالب"].XValueMember = data.Columns[2].ToString();
    chart1.Series["عدد الطالب"].YValueMembers = data.Columns[0].ToString();

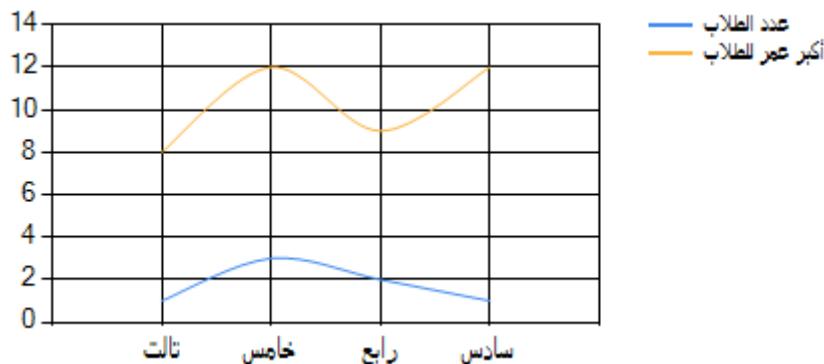
    chart1.Series.Add("أكبر عمر للطلاب");
    chart1.Series["أكبر عمر للطلاب"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Column;
    chart1.Series["أكبر عمر للطلاب"].XValueMember =
data.Columns[2].ToString();
    chart1.Series["أكبر عمر للطلاب"].YValueMembers =
data.Columns[1].ToString();
}
```

ويمكن أيضاً إظهار أكثر من سلسلة على ذات المخطط بالطريقة ذاتها.





وبتغيير شكل المخطط:

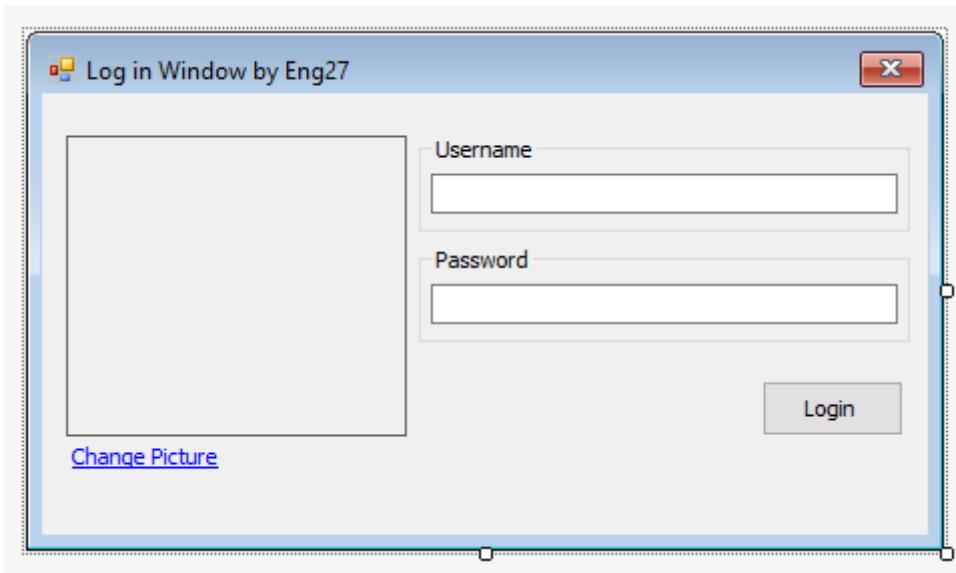


حفظ الصور في قاعدة البيانات

يمكن حفظ الصور في قاعدة البيانات بطريقتين:

- تخزين الصور في مكان ما من القرص الصلب أو ضمن ملفات المشروع، وحفظ مسارها في قاعدة البيانات، وهكذا فإن حجم قاعدة البيانات لن يكون كبيراً والبرنامج لن يحدث به بطء جراء الاستدعاء المتكرر لعناصر قاعدة البيانات، إذ إن القيم المخزنة في قاعدة البيانات والتي تمثل الصور هي قيم نصية. لكن مشكلة هذه الطريقة أن الصور عرضة للتلف أو الفيروس أو السرقة أو التعفيش¹ لأنها مكتوبة لل العامة.
 - حفظ الصور نفسها في قاعدة البيانات وذلك على شكل بيانات ثنائية Binary Data، وهكذا فإن الصورة ستكون محمية. لكن بالمقابل فحجم قاعدة البيانات سيزيد.
- وكمثال بسيط على الطريقة الأولى أنشأ مشروعًا جديداً وصمم النافذة التالية له:

¹ هل تذكر مصطلح التعفيش؟؟ السرقة في وضع النهار وخارج عن سيطرة الحكومة.



اضبط الخاصية `BorderStyle` لصندوق الصورة على `FixedSingle`، و `Size` على `170:150`، و `SizeMode` على `StretchImage`.
ثم أنشئ قاعدة بيانات باسم `LoginDB`، وأنشئ فيها جدولًا باسم `users` فيه الحقول التالية:

	Field Name	Data Type
key	ID	Number
	User	Long Text
	Pass	Long Text
	Image	Long Text

املأ الحقول ببعض البيانات.. ثم استخدم الكود التالي في الفيجوال ستوديو:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;

namespace SaveImagesDB
{
    public partial class Form1 : Form
```



```

{
    static string accessDB = @"Provider=Microsoft.jet.OLEDB.4.0;Data
Source = " + Application.StartupPath + "\\LoginDB.mdb;";

    static string id;
    static bool login = false;
    OleDbConnection connection;
    OleDbCommand command;
    OleDbDataReader reader;

    public Form1()
    {
        InitializeComponent();
        connection = new OleDbConnection(accessDB);
        textBox2.PasswordChar = '\u25cf'; //PassChar: ****
        linkLabel1.Enabled = false;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        connection.Open();
        command = new OleDbCommand("select * from users", connection);
        reader = command.ExecuteReader();
        while (reader.Read())
        {
            if (textBox1.Text == reader["User"].ToString())
                if (textBox2.Text == reader["Pass"].ToString())
                {
                    linkLabel1.Enabled = true;
                    pictureBox1.Image =
Image.FromFile(reader["Image"].ToString());
                    login = true;
                    id = reader["id"].ToString();
                }
        }
        if (!login)
            MessageBox.Show("Incorrect Password or Username!");
        connection.Close();
    }

    private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
{
    OpenFileDialog open = new OpenFileDialog();
    open.Filter = "JPG Images|*.jpg";
    if (open.ShowDialog() == System.Windows.Forms.DialogResult.OK)
    {
}
}

```



```
        connection.Open();
        command = new OleDbCommand("update users set [image] =
@image where id = " + id + "", connection);
        command.Parameters.AddWithValue("@image", open.FileName);
        command.ExecuteNonQuery();
        pictureBox1.Image = Image.FromFile(open.FileName);
        connection.Close();
    }
}
}
```

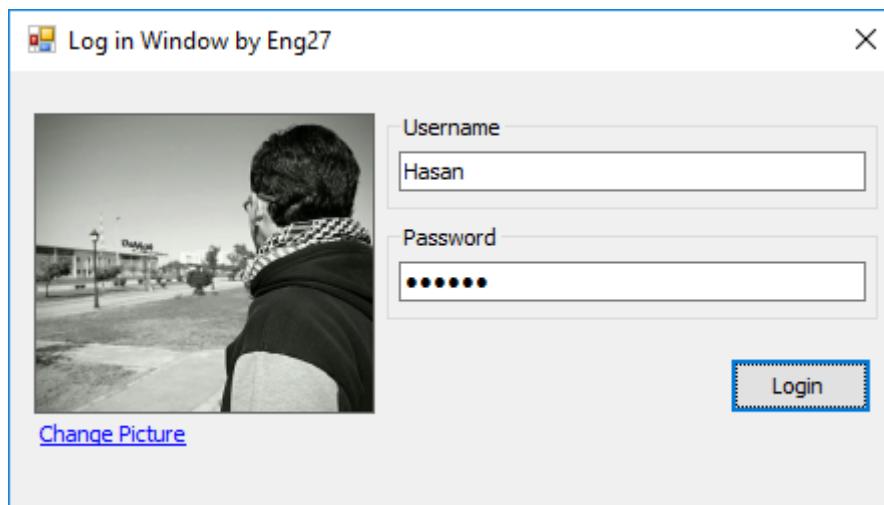
عند الضغط على زر تسجيل الدخول فإن جميع بيانات قاعدة البيانات سيتم معالجتها وقارنتها مع صناديق نصوص اسم المستخدم وكلمة مروره، في حال تطابق إحدى البيانات يتم تسجيل الدخول وتحميل الصورة وذلك بالاعتماد على مسارها في قاعدة البيانات ثم جعل قيمة متغير منطقي مساوية لـ 1، والتي تعني تم تسجيل الدخول.

وبعدها تتم مقارنة قيمة هذا المتغير فإذا كان 0 يظهر رسالة خطأ.

عند الضغط على رابط تغيير الصورة يتم إظهار كائن استعراض الملفات..

يمكن تطوير المثال ليتم نسخ الصورة إلى جوار البرنامج مثلاً أو لحفظها في مكان ما. في هذا المثال تم الإبقاء على الصورة في مكانها، أي أنه أي فقد في الصورة من الممكن أن يسبب خطأ في عمل البرنامج.

لاحظ البرنامج بعد تسجيل الدخول:



أما من أجا الطریقة الثانیة فالموضوع أعقد. تابع معي..



صمم نفس النافذة السابقة واستخدم الكود التالي:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.IO;
namespace SaveImagesDB2
{
    public partial class Form1 : Form
    {
        static string accessDB = @"Provider=Microsoft.jet.OLEDB.4.0;Data
Source = " + Application.StartupPath + "\\LoginDB.mdb;";
        MemoryStream ms; //لتخزين الصورة
        byte[] byteImage; //لتحويلها إلى بيانات
        static string id;
        static bool login = false;
        OleDbConnection connection;
        OleDbCommand command;
        OleDbDataReader reader;
        public Form1()
        {
            InitializeComponent();
            connection = new OleDbConnection(accessDB);
            textBox2.PasswordChar = '\u25cf'; //PassChar: ****
            linkLabel1.Enabled = false;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            connection.Open();
            command = new OleDbCommand("select * from users", connection);
            reader = command.ExecuteReader();
            while (reader.Read())
            {
                if (textBox1.Text == reader["User"].ToString())
                    if (textBox2.Text == reader["Pass"].ToString())
                {
                    linkLabel1.Enabled = true;
                    try
                    {

```



```
byte[] img = (byte[])reader["Image"];
ms = new MemoryStream(img);
pictureBox1.Image = Image.FromStream(ms);
}
catch
{
}
}
login = true;
id = reader["id"].ToString();
}
if (!login)
    MessageBox.Show("Incorrect Password or Username!");
connection.Close();
}
}
void JpgToBytes() // إجراء يحول الصورة لبيانات
{
    ms = new MemoryStream();
    pictureBox1.Image.Save(ms, pictureBox1.Image.RawFormat);
    byteImage = ms.ToArray();
}

private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
{
    OpenFileDialog open = new OpenFileDialog();
    open.Filter = "JPG Images|*.jpg";
    if (open.ShowDialog() == System.Windows.Forms.DialogResult.OK)
    {
        pictureBox1.Image = Image.FromFile(open.FileName);
    }
    JpgToBytes();
    connection.Open();
    command = new OleDbCommand("update users set [image] =
(@img) where id = " + id + "", connection);
    command.Parameters.Add("@img", OleDbType.Binary).Value =
byteImage;
    command.ExecuteNonQuery();
    connection.Close();
}
}
```

هل لاحظت كيف اختلفت الأكواد بشكل كبير بين هذه الطريقة وتلك؟ لا يوجد اختلاف بأكواد الاتصال والاستعلام والتعديل وغيرها، الفارق هو عند



حفظ الصورة كبيانات في قاعدة البيانات وعند تحميل الصورة من قاعدة البيانات كبيانات أيضاً.. حلقة التكرار كما في الطريقة السابقة لمعالجة جميع الحقول، أما حلقة try – catch فهي ستنقيك من حدوث خطأ مفاجئ لأن لا يمكنك تحويل قيمة فارغة من قاعدة البيانات لمصفوفة بيانات (أو أخطاء أخرى لا تتوقعها). أما إجراء تحويل الصورة فيقوم بتحويل صورة صندوق الصورة إلى كائن ذاكرة، والذي بدوره يتم تحويله إلى مصفوفة بيانات. وبالنسبة لاختيار صورة جديدة فهذا الكود يقوم بتحديث بيانات قاعدة البيانات بالإضافة الصورة المختارة إلى حقولها.

اذهب إلى قاعدة البيانات وتفقد حجمها..

النسخ الاحتياطي

لإجراء النسخ الاحتياطي هنا يكفي نسخ قاعدة البيانات واستعادتها في وقت لاحق وذلك عبر أكواد نسخ الملفات:

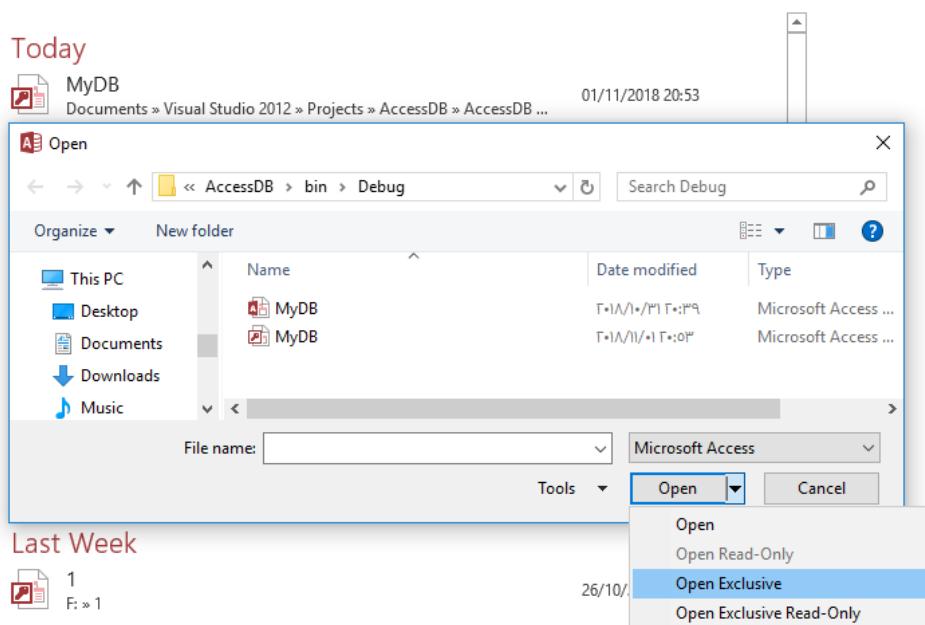
```
// أزرار النسخ الاحتياطي غير موجودة في المشروع، تكرّم وأضفها ^_^
private void Backup_Click(object sender, EventArgs e)
{
    SaveFileDialog save = new SaveFileDialog();
    save.Filter = "قاعدة بيانات (*.mdb)|*.mdb";
    if (save.ShowDialog() == DialogResult.OK)
    {
        try
        {
            File.Delete(save.FileName);
        }
        catch { }
        try
        {
            string path = Application.StartupPath + @"\MyDB.mdb";
            File.Copy(path, save.FileName, true);
            MessageBox.Show("تم تصدير نسخة احتياطية بنجاح!");
        }
        catch
        {
            MessageBox.Show("لا يمكن إجراء نسخة احتياطية!");
        }
    }
}
private void Restore_Click(object sender, EventArgs e)
{
    DialogResult M = MessageBox.Show("هل تود بالتأكيد استعادة نسخة احتياطية?", "سابقة؟ ستفقد جميع المعلومات التي لم يتم حفظها!",
```



```
MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1, MessageBoxOptions.RightAlign);
if (M == System.Windows.Forms.DialogResult.Yes)
{
    OpenFileDialog open = new OpenFileDialog();
    open.Filter = "بيانات قاعدة (*.mdb)|*.mdb";
    if (open.ShowDialog() == DialogResult.OK)
    {
        try
        {
            string path = Application.StartupPath + @"\MyDB.mdb";
            File.Copy(open.FileName, path, true);
            dataGridView1.DataSource = LoadData();
            MessageBox.Show("تم استعادة نسخة احتياطية بنجاح!");
        }
        catch
        {
            MessageBox.Show("لا يمكن استعادة نسخة احتياطية!");
        }
    }
}
```

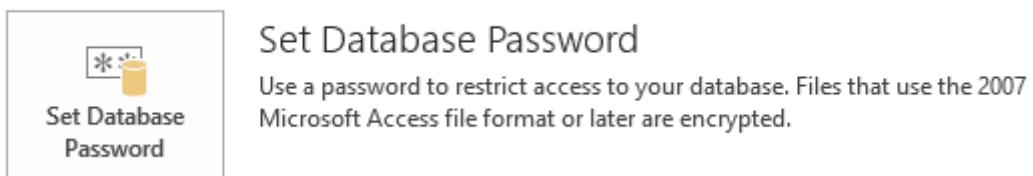
حماية قاعدة السنان

يمكن حماية قاعدة بيانات برنامجك بوضع كلمة سر لها، ويمكن ذلك من خلال برنامج Access. افتح قاعدة بياناتك بالوضع الخاص وذلك عبر القائمة :Browse Open ثم الأمر

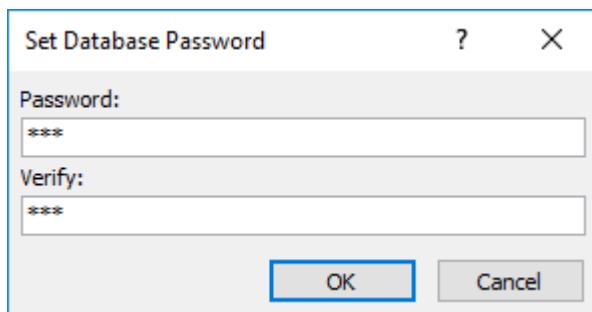




انتقل لتفاصيل قاعدة البيانات عبر القائمة File ثم اختر الأمر :Set Database Password الأخير ضمن النافذة



اضبط كلمة السر وأكدها:



وبطبيعة الحال فإن كود الاتصال بقاعدة البيانات سيكون بفرض كلمة السر هي 111

```
في مكان ما تصل إليه جميع الإجراءات//  
static string sql = @"Provider=Microsoft.jet.OLEDB.4.0; Data Source =  
C:\MyDB.mdb; Jet OLEDB:Database Password='111'; Persist Security  
Info=True;";  
OleDbConnection connection;  
في أحد إجراءات النموذج ول يكن إجراء تحميل النموذج//  
connection = new OleDbConnection(sql);
```

ويمكنك استخدام حماية قاعدة البيانات كحماية لبرنامجك:

```
في مكان ما تصل إليه جميع الإجراءات//  
static string accessDB = @"Provider=Microsoft.jet.OLEDB.4.0;Data Source = "  
+ Application.StartupPath + @"\MyDB.mdb;Jet OLEDB:Database Password=""  
+ password.Text + "";Persist Security Info=True;";  
OleDbConnection connection;  
في أحد إجراءات النموذج ول يكن إجراء تسجيل الدخول//  
try  
{ connection = new OleDbConnection(accessDB); }  
catch  
{ MessageBox.Show("اسم المستخدم أو كلمة السر خطأ");}
```



إعادة ضبط كلمة السر

لإعادة ضبط كلمة السر عليك إزالتها أولاً ثم ضبطها من جديد، ويمكن ذلك عن طريق فتح قاعدة البيانات في Access بالوضع الخاص.



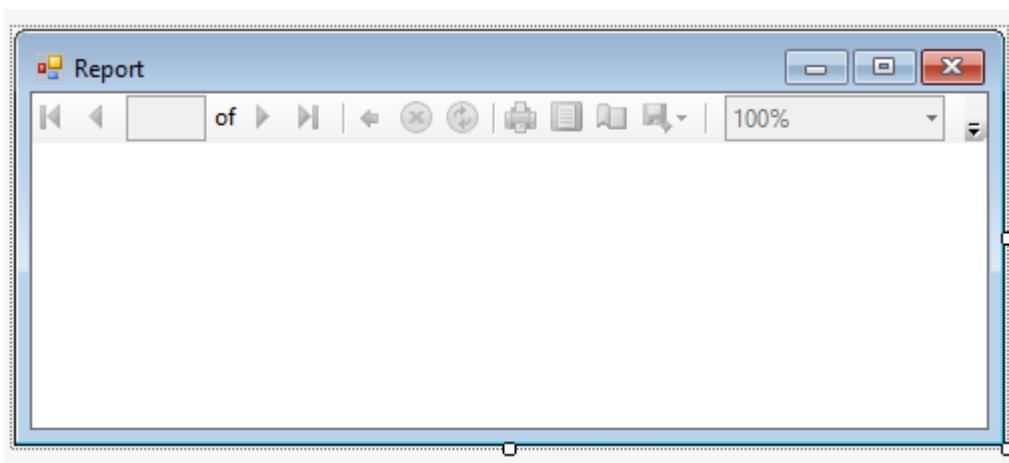


الفصل الرابع عشر – التقارير Reports

تنتشر التقارير بشكل واسع في البرامج التي تتعامل مع قواعد البيانات، مثل برامج الإدارة والمحاسبة وغيرها.. والفصل الحالي سيشرح لك كيفية إنشاء تقرير بأبسط شكل ممكن.

يكون التقرير عادة عبارة عن سرد أو جرد لمجموعة من البيانات، ومن الممكن أن تكون البيانات هذه ضمن قاعدة بيانات من أي نوع، على سبيل المثال سننثر تقريراً لقاعدة بيانات أكسس. (نفس قاعدة البيانات التي استخدمناها في الفصل السابق). والجدير بالذكر أن التقرير لا يحوي فقط على جدول بيانات معينة، وإنما قد يحوي على تفاصيل أخرى تضيفها أنت وفق رغبتك وحاجتك.

كبداية، صمم نافذة جديدةً لعرض التقرير عليها، أضف نافذة جديدةً لمشروعك وأضف إليها الأداة ReportViewer:

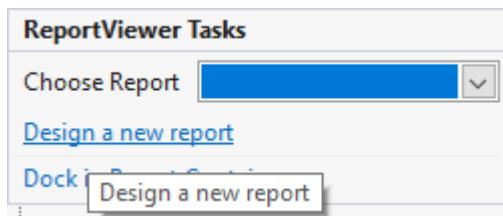




غير الخاصية Public Modifiers إلى لتتمكن من الوصول إلى الأداة من النوافذ الأخرى، وDock إلى Fill. اتبع الفقرات التالية خطوة خطوة:

تصميم التقرير

:Design a new report انقر على السهم أعلى أداة عارض التقارير ثم



سيظهر معالج تهيئة مصدر البيانات، اختر قاعدة بيانات، ثم اختر قاعدة البيانات من نوع Dataset. أنشئ بعدها اتصال جديد، ثم ابحث عن قاعدة البيانات المطلوبة.

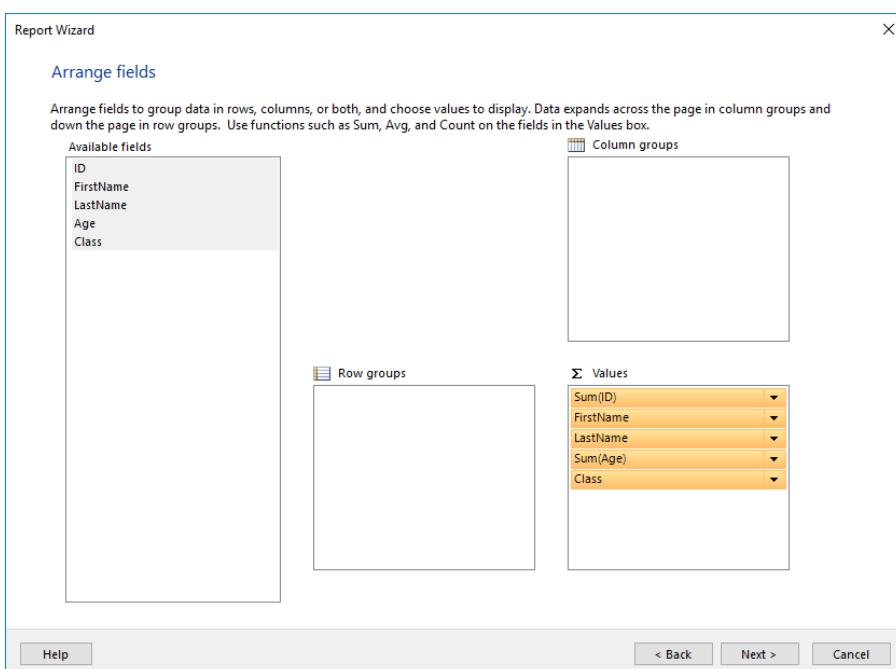
اضغط على زر تجربة الاتصال كما جرت العادات والتقاليد، ثم موافق.

اضغط التالي فتحصل على رسالة مفادها أن قاعدة البيانات ستنسخ إلى موقع المشروع، اضغط موافق.

ستظهر نافذة جديدة حدد ضمنها جميع الجداول المطلوبة، ثم إنتهاء.

ستظهر نافذة تخبرك بتفاصيل مصدر البيانات في مشروعك، اضغط التالي.

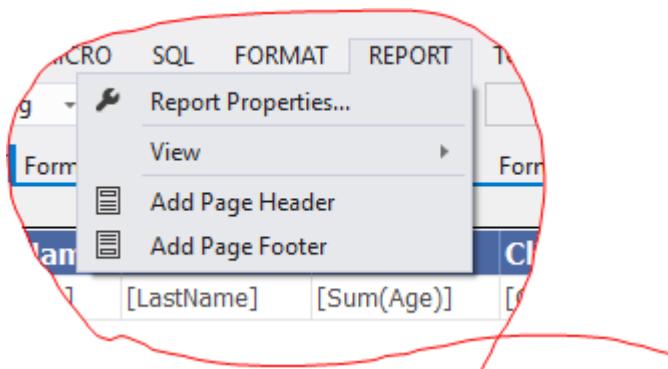
اسحب جميع الحقول المطلوب عرضها إلى Values:



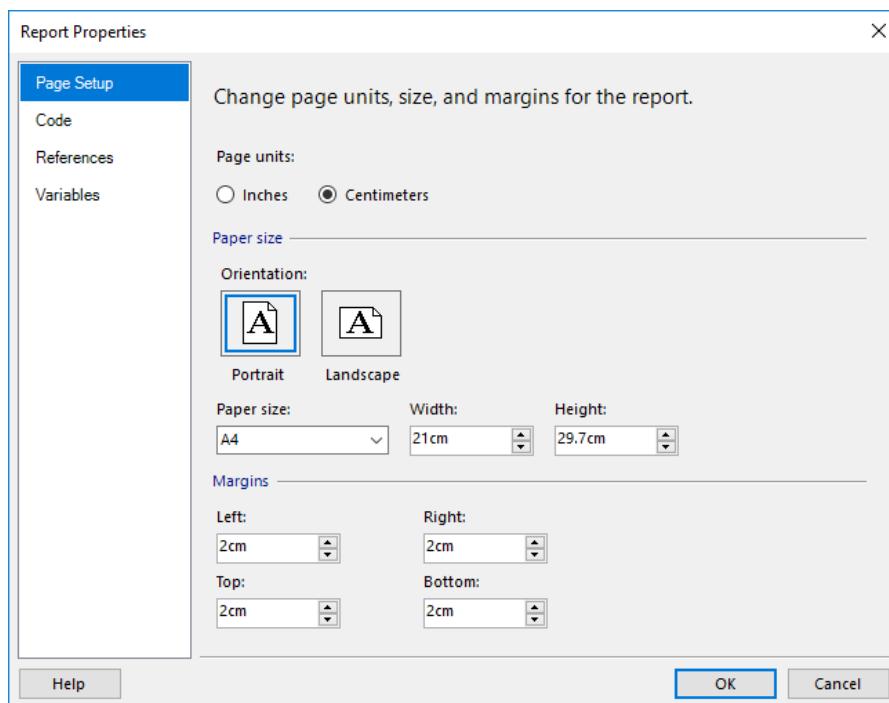


اضغط التالي، ثم إنتهاء. ليصبح المشروع بالشكل:

ضمن قوائم بيئة التطوير - بجانب Edit و File وغيرها من القوائم في الأعلى - هناك قائمة باسم REPORT، انقر عليها لتحصل على:



الأمر الأول، يعطيك إمكانية تنسيق حجم التقرير واتجاهه:



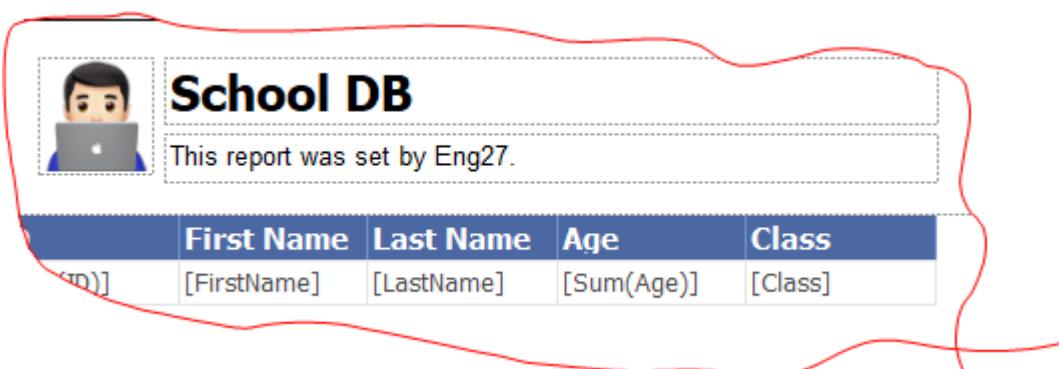


الأمر الثاني يعرض لك مسطرة كما لو أنك في برنامج وورد، ويعرض لك المجموعات لديك.

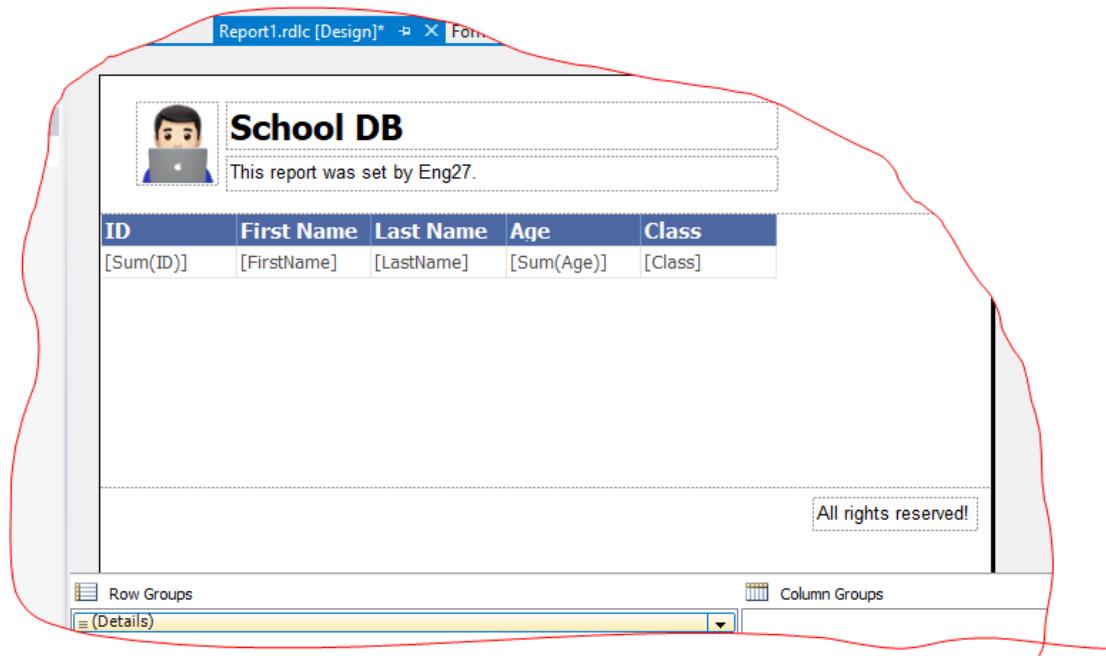
الأمر الثالث لإضافة رأس للتقرير، وهو نفسه المنطقة المكتوب عليها اسم كتابي وأسمى في أعلى أوراق كتابي هذا.

الرابع لإضافة تذيل للتقرير، ومشابه للرأس التقرير لكن في أسفل الصفحة.

لنعد للتقرير وتصميمه، أضف إلى رأس التقرير أداة صورة وأدوات نصوص أو ماتراه مناسباً، واضبط الصورة على الوضع :Stretch



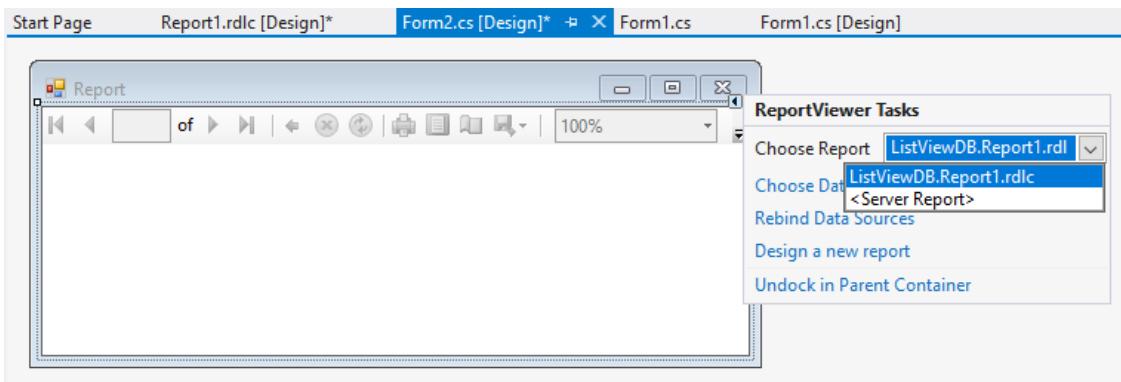
أما التذيل (في أسفل الصفحة)، فضع فيه أي عبارة أخرى على كيفك. ليصبح التقرير بالشكل:



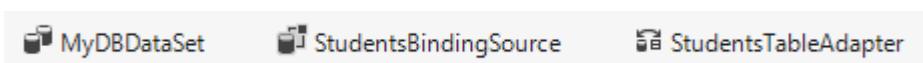


ربط أداة عرض التقارير بالتقرير

أيا كانت الطريقة التي أنشأت التقرير من خلالها - سواءً بالطريقة المذكورة بالفقرة السابقة أو يدوياً أو غيرها - فإنه عليك ربط التقرير بأداة عرض التقارير ReportViewer، والعملية بسيطة للغاية، عد إلى النافذة التي تحتوي على التقرير ومن السهم في أعلى الأداة:



وунدها ستلاحظ أنه تم إهداوك ثلاثة كائنات:



وما يهمك هو الكائن الأول،即 DataSet، والذي يحوي قاعدة بياناتك.

عرض تقرير عن كافة المحتويات

أضف زر إلى نافذة برنامجك لطباعة تقرير بالمحتويات، كوده ما يلي:

```
connection.Open(); Form2 f = new Form2(); // التقرير
adapter = new OleDbDataAdapter("select * from students", connection);
adapter.Fill(f.MyDBDataSet.Students);
f.reportViewer1.RefreshReport();
f.Show(); connection.Close();
```

لكن انتقل إلى كود نافذة التقرير واسمح للأكواد التالية أو حولها لملاحظة برمجية:

```
private void Form2_Load(object sender, EventArgs e)
{
    // TODO: This line of code loads data into the 'MyDBDataSet.Students' table. You can move, or remove it,
    this.StudentsTableAdapter.Fill(this.MyDBDataSet.Students);
    this.reportViewer1.RefreshReport();
}
```



ليظهر التقرير التالي بعد تكبيره:

ID	First Name	Last Name	Age	Class
2	معاذ	حمدى	10	خامس
8	خالد	حسين	11	رابع
1	رامي	أحمد	12	خامس
3	خالد	محمود	9	رابع
4	خالد	أحمد	11	خامس
5	محمد	سمير	9	رابع
6	محمود	أحمد	12	سادس
7	محمد	سعيد	8	ثالث

كما يمكنك عرض معاينة للصفحة التي ستظهر عند الطباعة، بالضغط على زر **معاينة الطباعة** Print Layout زر الطباعة:

ID	First Name	Last Name	Age	Class
2	معاذ	حمدى	10	خامس
8	خالد	حسين	11	رابع
1	رامي	أحمد	12	خامس
3	خالد	محمود	9	رابع
4	خالد	أحمد	11	خامس
5	محمد	سمير	9	رابع
6	محمود	أحمد	12	سادس
7	محمد	سعيد	8	ثالث

All rights reserved!



وبعدها، يمكنك طباعة التقرير أو حفظه كملف وورد أو pdf أو Excel.

عرض تقرير عن السجل المحدد

طبعاً هذه التفاصيل "المعينة" هي تفاصيل من قاعدة البيانات نفسها، أي أننا سنقوم بعرض نتائج بناءً على أوامر نجريها على قاعدة البيانات.

لذلك فعلينا الوصول لقاعدة البيانات .. ستقول لي ما الجديد في الموضوع؟ وسأقول لك: هل تذكر ال DataSet الذي قلنا أنه يتم إهداؤك إياه عند ربط أداة عرض التقارير بال்தقرير، في الواقع نحن نجري الأوامر على قاعدة البيانات ونضع النتائج التي نحصل عليها في ال DataSet. ما معنى هذا الكلام؟؟ لاحظ معي:

في البداية اذهب إلى خصائص كل من reportViewer وDatasert وغيرها إلى Public لتتمكن من إرسال الأوامر من جميع نقاط برامجك.

استخدم الكود التالي:

```
connection.Open();
Form2 f = new Form2();
adapter = new OleDbDataAdapter("select * from students where id = " + id
+ "", connection);
adapter.Fill(f.MyDBDataSet.Students);
f.reportViewer1.RefreshReport();
f.Show();
connection.Close();
```

طبعاً id يحوي قيمة السجل المحدد والذي يمكنك الحصول عليه من أجل ListView بالكود التالي:

```
private void listView1_SelectedIndexChanged(object sender, EventArgs e)
{
    foreach (ListViewItem L in listView1.Items)
        if (L.Selected)
    {
        id = L.SubItems[0].Text;
    }
}
```

ولتجنب حدوث خطأ، أنشئ حدثاً يراقب تعديلات id، فإذا لم تكن تحوي بيانات يجب عدم تفعيل الزر الذي سينشئ تقريراً بالسجل المحدد.



كما يمكنك عرض السجل المحدد بشكل عمودي وذلك بتصميم التقرير بالشكل التالي:

ID	First Name	Last Name	Age	Class
[Sum(ID)]	[FirstName]	[LastName]	[Sum(Age)]	[Class]

عرض تقرير عن نتائج البحث

يتم البحث عن السجلات من خلال صندوق نص وذلك عند تفجير¹ حدث تغيير الكتابة:

```
connection.Open();
Form2 f = new Form2();
string load = "select * from students where firstname+lastname+class like
'%" + textBox1.Text + "%'";
command = new OleDbCommand(load, connection);
listView1.Items.Clear();
reader = command.ExecuteReader();
while (reader.Read())
{
    L = listView1.Items.Add(reader["id"].ToString());
    L.SubItems.Add(reader["firstname"].ToString());
    L.SubItems.Add(reader["lastname"].ToString());
    L.SubItems.Add(reader["age"].ToString());
    L.SubItems.Add(reader["class"].ToString());
}
connection.Close();
```

حيث L متغير يمثل عناصر ومحطويات ListView.

¹ هل تذكر مصطلح تفجير؟؟ قلنا أن تفجير الحدث يعني حدوث الحدث.



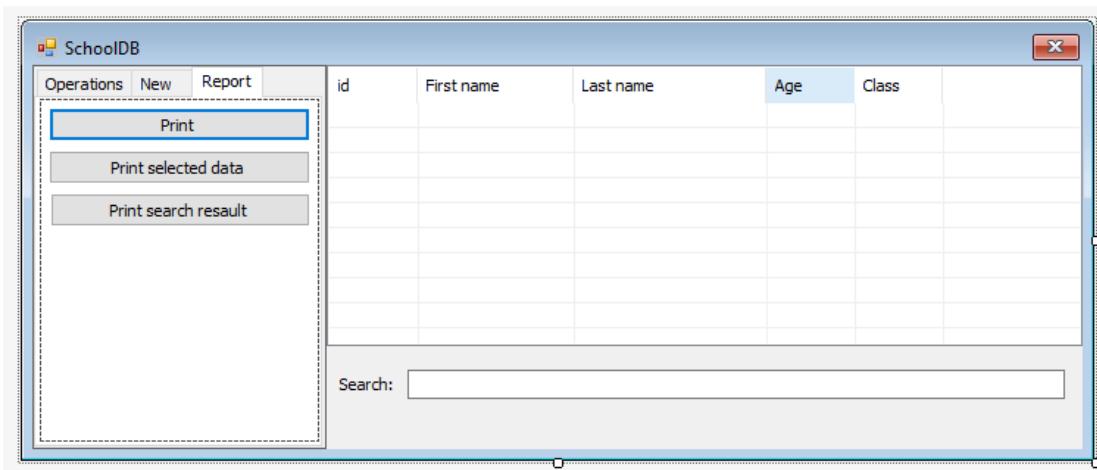
أما طباعة عناصر البحث فيكون بالإجراء التالي:

```
connection.Open();
Form2 f = new Form2();
string load = "select * from students where firstname+lastname+class like
'%" + textBox1.Text + "%'";
adapter = new OleDbDataAdapter(load, connection);
adapter.Fill(f.MyDBDataSet.Students);
f.reportViewer1.RefreshReport();
f.Show();
connection.Close();
```

طيب السؤال الذي قد يدور في خاطرك: لماذا مرة استخدمنا adapter ومرة command؟

الحقيقة أن الكائن الأول يقوم بربط الكائنات مع بعضها، مثل ربط ناتج أمر من أوامر قاعدة البيانات مع جداول عرض أو تقارير أو غيرها، أما الثاني فمن خلاله بإمكانك قراءة نتائج تنفيذ أوامر قاعدة البيانات مباشرة. لاحظ أننا استخدمنا الأول عند قراءة البيانات – لإظهارها على ListView – والثاني استخدمناه عند إرسال النتائج إلى التقرير (الرابط بين نتيجة الأمر والDataSet تبع التقرير).

ونافذة البرنامج الرئيسية أصبحت بالشكل:



لا تنس أن تعرف كل من المتغيرات اللاحمة في المشروع، command وreader وconnection وadapter وغيرها من الكائنات والمكتبات التي تلزمك لتنفيذ أوامر قواعد البيانات.



الفصل الخامس عشر - تقنيات مفيدة 😊

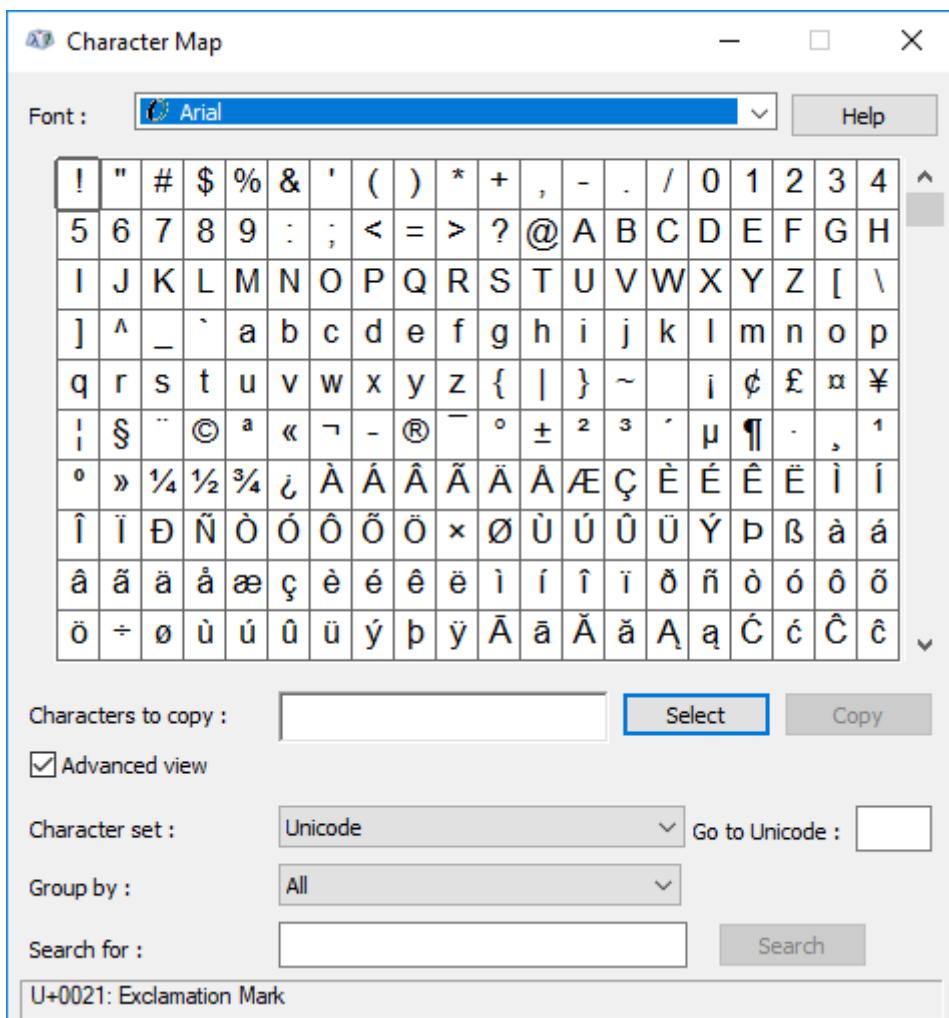
كخاتمة لهذا الكتاب، سيقدم لك هذا الفصل بعض المهارات والتقنيات التي قد تجعل من برامجك أفضل.

محارف غير موجودة في كيبوردك!

كما تعلم فلوحة المفاتيح لا تحتوي إلا على محارف محددة هي محارف لاتينية وأخرى عربية وبعض الرموز لإشباع حاجاتك، وعندما تصادف مثل هذه المحارف سيخطر ببالك أنها كُتبت بمحرر نصوص متقدم مثل برنامج وورد أو غيره من البرامج، وقد أصدرك عندما أقول لك بإمكانك كتابة أي رمز يتعرف عليه ويندوز باستخدام أي محرر نصوص أو أي صندوق نصوص حتى..

في البداية سأعرفك على الرموز التي يمكن لويندوز التعرف عليها. افتح قائمة ابدأ واكتب Character Map، لتحصل على برنامج يحوي العديد من الرموز والتي لأول مرة قد تراها.

بإمكانك من خلال هذا البرنامج البحث عن الرموز أيضا بمعرفة شيفرة الرمز بتشفير Unicode، وهو ما سنتعامل معه في برامجنا.



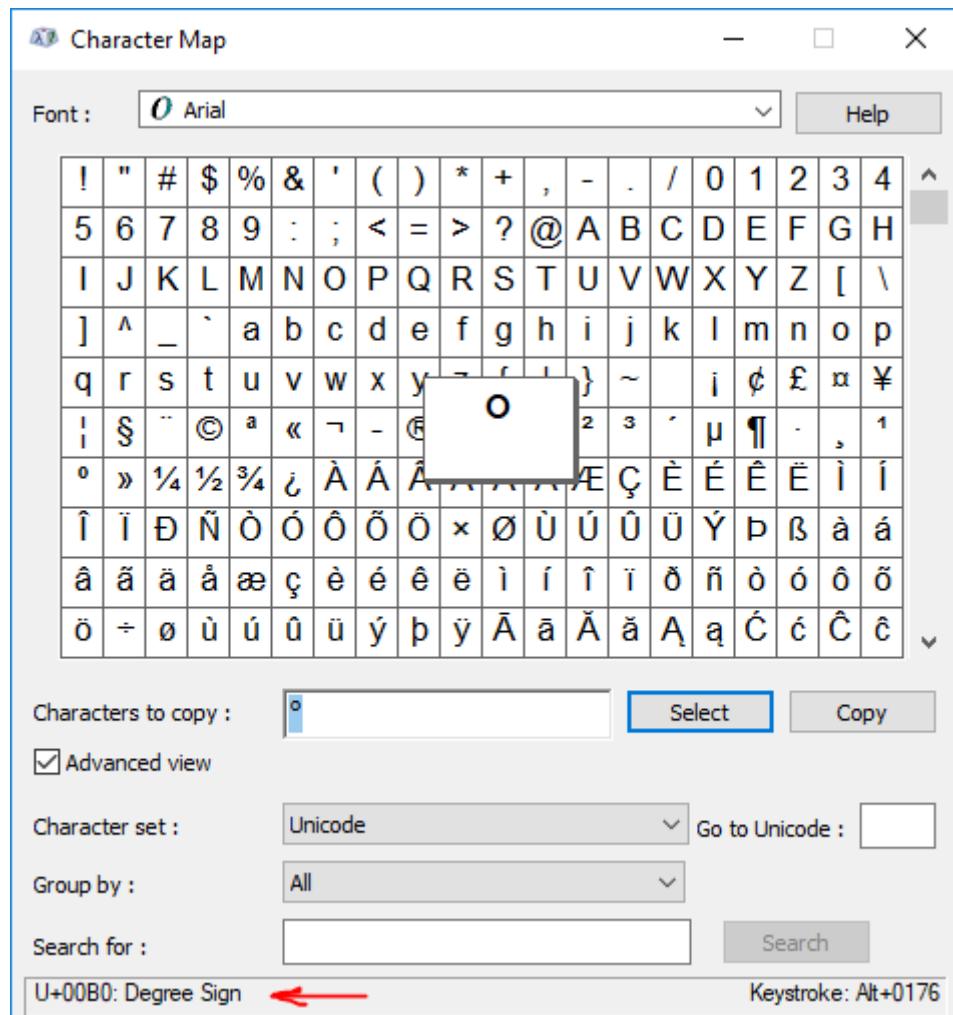
كمستثمر عادي لوبيندوز - حاشاك من هذا 😊 - بإمكانك تحديد الرمز ثم نسخ، أما كمبرمج فمثل هذه الرموز ستفيده في برامجك بشكل واسع، فعند تعاملك مع المسائل الرياضية مثلا تحتاج لرموز مثل ٤ رمز ابسيلون أو Δ رمز دلتا أو ° رمز الدرجة أو غيرها الكثير من الرموز.

لا شك أنك قد تابعت الكثير من الفيديوهات والدورس والتي تحوي صناديق نصوص تم ضبط خاصية PasswordChar فيها على القيمة " * "، ومما لا شك فيه أيضا أنك تساءلت أنه لماذا لا يتم استخدام " • "، ومما لا شك فيه أيضا أن المشكلة واجهتك ولم تعلم كيفية ضبط صناديق كلمات السر علىدائرة السوداء عوضا عن النجمة. وفي حال لا زلت تبحث عن الموضوع فهذه الفقرة هي ضالتك.

بإمكانك استخدام رمز برنامج خريطة المحارف وذلك بإسناد قيمة الرمز المطلوب عرضه إلى متغير نصي لاستخدامه عند الحاجة، أو استخدم كود المحرف، والذي تجده أسفل نافذة البرنامج.



فمثلاً لعرض عبارة نصية مع رمز درجة مئوية، ابحث أولاً عن المحرف، خذ كوده، ثم أسنده للمتغير:



خذ شيفرة المحرف المتمثلة برمز الـ Unicode الخاص به، ثم اكتبها داخل عبارة نصية بعد \u مباشرة.

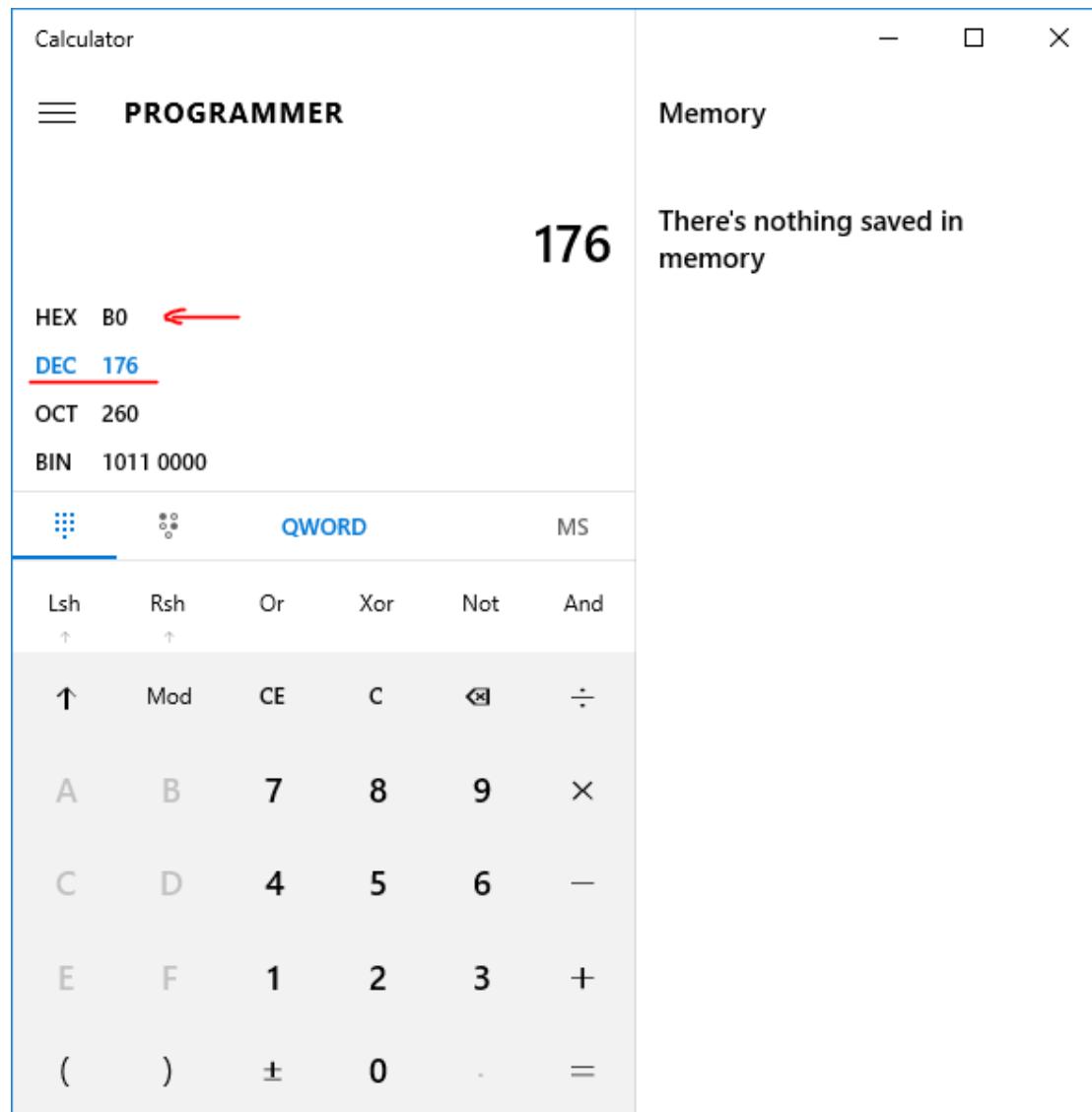
```
int i = 25;
string temp;
temp = i + " C\u00b0";
```

كما يمكنك كتابة المحرف باستخدام لوحة المفاتيح وذلك من خلال أكواد المفتاح Alt، والتي يمكنك إيجادها بنفس البرنامج السابق وذلك أمامك. فلكتابة ° استمر بالضغط على Alt واضغط بالتالي على Keystroke



المفاتيح 0 ثم 1 ثم 6 ثم 7 ثم 0176 (أي أن الكود هو Alt + 0176¹). بإمكانك تحرير المفتاح Alt الآن.

ما قد يثير اهتمامك هو أن الكود البرمجي - مع "\u" - ماهو إلا تحويل مابين النظامين العشري والست عشري:



أي أن شيفرات Unicode للمحارات تستخدم بالنظامين العشري والست عشري، فالأولى تستخدم برمجيا والثانية مع مفتاح Alt.

بعد هذه اللحظة سيكون من المخجل أن تنشئ برماج فيها صناديق الرسائل الخاصة بكلمات السر بالشكل "*****" ، وعوضا عنها استخدم

¹ راجع الملحق ب - أكواد Alt



"....." ، حيث أن المحرف الأخير شيفرة الـ Unicode الخاصة به لها القيمة u+0095 بالنظام السنت عشري والقيمة 0149 بالنظام العشري.

وكما أشرنا فإنه بإمكانك استخدام المحارف المعبرة عن الرموز والأحرف المستخدمة في الرياضيات والفيزياء مثل..

التحكم بمدخلات صناديق النصوص

عند تعاملك مع بيانات محددة فإنك ترغب بالحصول على معطيات محددة حتى لا تدخل بمتاهات احتمالات إدخال المستخدم والأخطاء التي قد تهدم مابنيته وتشوه سمعتك كمبرمج.. لذلك فعوضاً عن وضع خوارزميات معقدة وطويلة تتنبأ بما سيدخله المستخدم وما يجب على برنامجك أن يقوم به فيما لو أدخل المستخدم البيانات الفلانية، بإمكانك إجباره على معطيات محددة لا يمكنه الخروج عنها.

فمثلاً صناديق النصوص التي تستقبل أرقام الهواتف يجب أن تكون أرقاماً فقط، أو تلك التي تستقبل معطيات رقمية مثل الأعمار والعلامات القراءات الرقمية وغيرها.

ال코드 التالي يسمح لك بإدخال أرقام فقط:

```
ضع الكود في حدث //KeyPress
if (e.KeyChar != '0' &
    e.KeyChar != '1' &
    e.KeyChar != '2' &
    e.KeyChar != '3' &
    e.KeyChar != '4' &
    e.KeyChar != '5' &
    e.KeyChar != '6' &
    e.KeyChar != '7' &
    e.KeyChar != '8' &
    e.KeyChar != '9')
{
    e.Handled = true;
}
```

وفي حال أردت السماح بإدخال الفاصلة ". ، فكر أحد أجزاء الشرط ليحتوي الفاصلة، أو أرج رأسك واستخدم أداة UpDownTextBox.



وللسماح بأحرف معينة مثلاً Y:

```
ضع الكود في حدث//  
//KeyPress  
if (e.KeyChar != 'y' | e.KeyChar != 'Y')  
{  
    e.Handled = true;  
}
```

ولمنع إدخال المحارف:

```
ضع الكود في حدث//  
//KeyPress  
if (e.KeyChar == 'y' | e.KeyChar == 'Y')  
{  
    e.Handled = true;  
}
```

طبعاً بإمكانك التعامل مع شيفرة Ascii أو Unicode للمحارف فتحصل على تحكم أفضل.

تأكيد الخروج من البرنامج

بإمكانك عرض رسالة تأكيد تسؤال المستخدم فيما إذا كان متأكداً من نيته بالخروج من البرنامج أم لا، قد لا يعلم مثلاً أن الزر الأحمر الموجود في زاوية البرنامج العليا هو للخروج، فقد تكون هذه هي أول مرة يستخدم فيها الكمبيوتر، وبرنامجك هو من تشرف بكونه أول برنامج لمستخدمك.

قد لا يكون هذا الكود بهذه الأهمية لدرجة أن يوضع ضمن التقنيات المفيدة لكن لا بأس بسرده هنا، قد تحتاجه لإظهار رسالة تطلب من المستخدم حفظ التغييرات أو ما شابه ذلك.

```
ضع الكود في حدث//  
//Form_Closing  
DialogResult result = MessageBox.Show("هل أنت متأكد من أنك ترغب بالخروج?",  
    "تأكيد الخروج", MessageBoxButtons.YesNo, MessageBoxIcon.Question);  
if (result == DialogResult.No)  
{  
    e.Cancel = true;  
}
```



تغيير خط إحدى الأدوات

بإمكانك تغيير خط أي أداة بما فيها النموذج Form بذات نفسه أثناء التشغيل، وذلك بال코드 التالي:

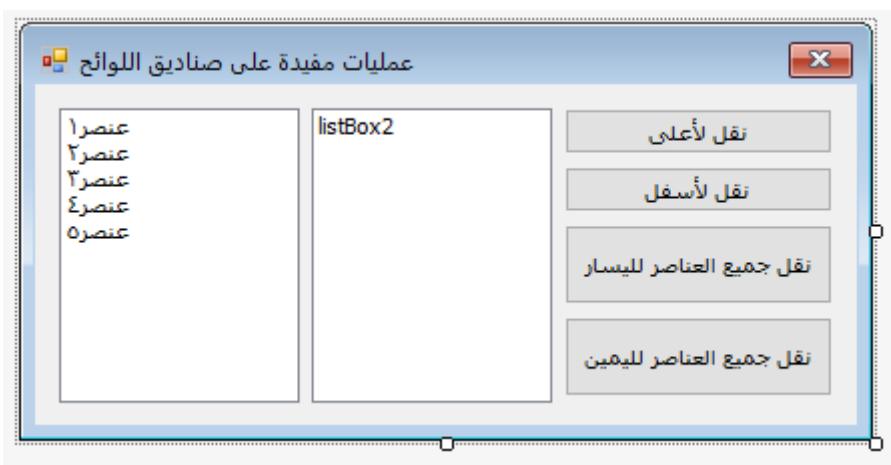
```
this.Font = new Font("Tahoma", 14, FontStyle.Bold);
```

طبعاً بإمكانك تغيير this لاسم الأداة المطلوبة.

عمليات مفيدة على ListBox

عند وجود أداتي ListBox أو أكثر بإمكانك نقل العناصر فيما بينها، بحيث تنقل عنصر واحد أو جميع العناصر. كما يمكنك إعادة ترتيب العناصر بحيث تنقل العنصر خطوة إلى الخلف أو إلى الأمام داخل أداة صندوق اللائحة نفسها.

صمم النافذة التالية:



أضف خمسة عناصر لصندوق اللائحة الأول، واترك الصندوق الآخر حالياً أو املأه إن شئت. ثم استخدم الكود التالي:

```
ListBox Selected; // سنخزن هنا صندوق اللائحة الفعال حالياً
نقل عنصر واحد // private void listBox1_MouseDoubleClick(object sender, MouseEventArgs e)
{
    if (listBox1.SelectedIndex >= 0)
    {
        listBox2.Items.Add(listBox1.Items[listBox1.SelectedIndex]);
        listBox1.Items.RemoveAt(listBox1.SelectedIndex);
    }
}
```



```

private void listBox2_MouseDoubleClick(object sender, MouseEventArgs e)
{
    if (listBox2.SelectedIndex >= 0)
    {
        listBox1.Items.Add(listBox2.Items[listBox2.SelectedIndex]);
        listBox2.Items.RemoveAt(listBox2.SelectedIndex);
    }
}

// نقل جميع العناصر
private void AllLeft_Click(object sender, EventArgs e)
{
    foreach (string s in listBox2.Items)
    {
        listBox1.Items.Add(s);
    }
    listBox2.Items.Clear();
}

private void AllRight_Click(object sender, EventArgs e)
{
    foreach (string s in listBox1.Items)
    {
        listBox2.Items.Add(s);
    }
    listBox1.Items.Clear();
}

// ضبط الأداة الفعالة حالياً
private void listBox1_MouseDown(object sender, MouseEventArgs e)
{
    Selected = listBox1;
}

private void listBox2_MouseDown(object sender, MouseEventArgs e)
{
    Selected = listBox2;
}

// نقل العنصر لأعلى
private void MoveUp_Click(object sender, EventArgs e)
{
    if (Selected.SelectedIndex >= 0)
    {
        string temp;
        if (Selected.SelectedIndex > 0)

```

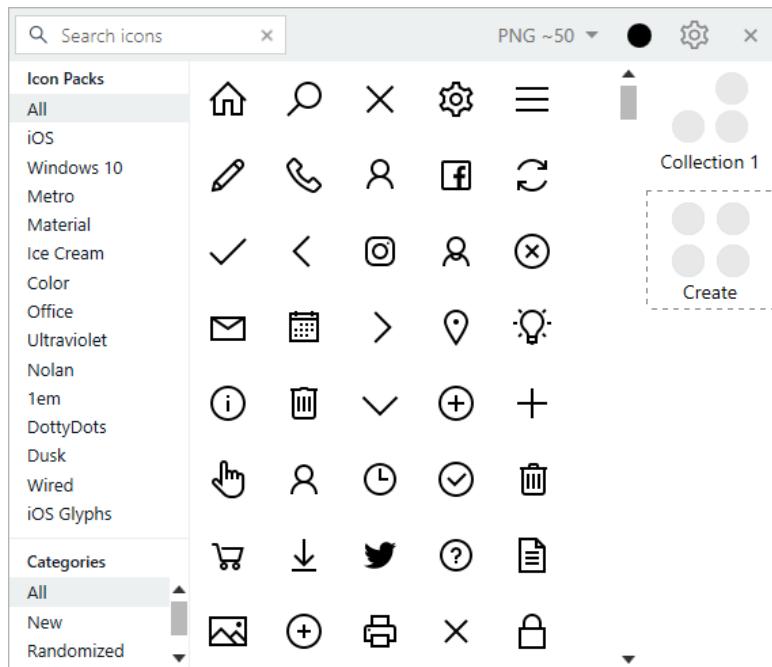


```
{  
    temp = Selected.Items[Selected.SelectedIndex].ToString();  
    Selected.Items[Selected.SelectedIndex] =  
Selected.Items[Selected.SelectedIndex - 1];  
    Selected.Items[Selected.SelectedIndex - 1] = temp;  
    Selected.SelectedIndex -= 1;  
}  
}  
}  
  
نقل العنصر للأسفل//  
private void MoveDown_Click(object sender, EventArgs e)  
{  
    if (Selected.SelectedIndex >= 0)  
    {  
        string temp;  
        if (Selected.SelectedIndex+1< Selected.Items.Count)  
        {  
            temp = Selected.Items[Selected.SelectedIndex].ToString();  
            Selected.Items[Selected.SelectedIndex] =  
Selected.Items[Selected.SelectedIndex + 1];  
            Selected.Items[Selected.SelectedIndex + 1] = temp;  
            Selected.SelectedIndex += 1;  
        }  
    }  
}
```

الأيقونات في التطبيقات مثل الألوان في اللوحات!

إن من أسوأ ما قد يواجهك خلال حياتك البرمجية هو شح الأيقونات لديك، خصوصاً إذا انتقلت لمرحلة متقدمة برمجياً بحيث تستطيع إسقاط جميع أفكارك على الأدوات المتوفرة لديك..

لذلك فسأعطيك تطبيقاً يحوي آلاف الأيقونات تشبع تعطشك للتصميم، وهو تطبيق :Icon8



* * *

وإلى هنا، نكون قد أنهينا جزءاً يسيراً مما يجب عليك معرفته لتتقن C#، وعلى أمل أن يحقق الكتاب الفائدة المرجوة منه أتمنى منك - عزيزي القارئ/المبرمج - نشر الكتاب ومعلوماته، وأن تدعوه لي وللمسلمين بظهور الغيب، عسى أن تكون ساعة استجابة أو أن تكون - أنت - من يحبهم الله، فإن الله خواص، في الأمكنة والأزمنة والأشخاص..

وختاماً، فإن هذا العمل لم يكن ليكتمل لولا توفيق من الله وحده، وإنني وإذ حاولت تنقيحه وتدقيقه إلا أن احتمال وجود أخطاء علمية أو نحوية وارد بلا شك، لذلك فيسعدني أن تقدم لي أخطائي أو أيّة نصائح أو اقتراحات لأأخذها بعين الاعتبار في النسخ القادمة من هذا الكتاب - إن قدر الله لهذا الكتاب أن يكون له نسخ أخرى - على وسائل التواصل الموجودة أول الكتاب. والسلام عليكم ورحمة الله وبركاته

حسن الفحل، 2019/2/13

الملحقات والمراجع

ملحق أ – جدول ASCII

The ASCII Table

Dec	Hex	Char									
00	00	NUL	32	20	SP	64	40	Ø	96	60	‘
01	01	SOH	33	21	!	65	41	A	97	61	ا
02	02	STX	34	22	”	66	42	B	98	62	ب
03	03	ETX	35	23	#	67	43	C	99	63	ـ
04	04	EOT	36	24	\$	68	44	D	100	64	ـ
05	05	ENQ	37	25	%	69	45	E	101	65	ـ
06	06	ACK	38	26	&	70	46	F	102	66	ـ
07	07	BEL	39	27	,	71	47	G	103	67	ـ
08	08	BS	40	28	(72	48	H	104	68	ـ
09	09	HT	41	29)	73	49	I	105	69	ـ
10	0A	LF	42	2A	*	74	4A	J	106	6A	ـ
11	0B	VT	43	2B	+	75	4B	K	107	6B	ـ
12	0C	FF	44	2C	,	76	4C	L	108	6C	ـ
13	0D	CR	45	2D	-	77	4D	M	109	6D	ـ
14	0E	SO	46	2E	.	78	4E	N	110	6E	ـ
15	0F	SI	47	2F	/	79	4F	O	111	6F	ـ
16	10	DLE	48	30	Ø	80	50	P	112	70	ـ
17	11	DC1	49	31	1	81	51	Q	113	71	ـ
18	12	DC2	50	32	2	82	52	R	114	72	ـ
19	13	DC3	51	33	3	83	53	S	115	73	ـ
20	14	DC4	52	34	4	84	54	T	116	74	ـ
21	15	NAK	53	35	5	85	55	U	117	75	ـ
22	16	SYN	54	36	6	86	56	V	118	76	ـ
23	17	ETB	55	37	7	87	57	W	119	77	ـ
24	18	CAN	56	38	8	88	58	X	120	78	ـ
25	19	EM	57	39	9	89	59	Y	121	79	ـ
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	ـ
27	1B	ESC	59	3B	;	91	5B	[123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	
29	1D	GS	61	3D	=	93	5D]	125	7D	}
30	1E	RS	62	3E	>	94	5E	-	126	7E	-
31	1F	US	63	3F	?	95	5F	-	127	7F	DEL

ملحق ب - أکواد Alt

Alt+		Alt+		Alt+	Alt+0		Alt+	Alt+0
0	@	64	Ç	€	128	Ł	À	192
1	A	65	ü	•	129	ł	Á	193
2	B	66	é	,	130	ł	Â	194
3	C	67	â	f	131	ł	Ã	195
4	D	68	ä	"	132	—	Ä	196
5	E	69	à	...	133	+	Å	197
6	F	70	å	†	134	ã	Æ	198
7	G	71	ç	‡	135	Ã	Ç	199
8	H	72	ê	~	136	Ł	È	200
9	I	73	ë	%o	137	Ƒ	É	201
10	J	74	è	Š	138	܂	Ê	202
11	K	75	ī	‘	139	܂	Ë	203
12	L	76	î	Œ	140	܂	Ì	204
13	M	77	ì	•	141	=	í	205
14	N	78	Ä	Ž	142	܂	Î	206
15	O	79	Å	•	143	܂	Ï	207
16	P	80	É	•	144	܂	Ð	208
17	Q	81	æ	‘	145	܂	Ñ	209
18	R	82	Æ	’	146	܂	Ò	210
19	S	83	ô	“	147	܂	Ó	211
20	T	84	ö	”	148	܂	Ô	212
21	U	85	ò	•	149	܂	Õ	213
22	V	86	û	—	150	܂	Ö	214
23	W	87	ù	—	151	܂	×	215
24	X	88	ÿ	~	152	܂	Ø	216
25	Y	89	Ö	™	153	܂	Ù	217
26	Z	90	Ü	š	154	܂	Ú	218
27	[91	ø	›	155	܂	Û	219
28	\	92	£	œ	156	܂	Ü	220
29]	93	Ø	•	157	܂	Ý	221
30	^	94	×	ž	158	܂	þ	222
31	_	95	f	Ŷ	159	܂	Þ	223
space	'	96	á		160	܂	à	224
!	a	97	í	í	161	܂	á	225
"	b	98	ó	¢	162	܂	â	226

Alt+		Alt+		Alt+	Alt+0	Alt+	Alt+0
#	35	c	99	ú	£	163	ò
\$	36	d	100	ñ	¤	164	ð
%	37	e	101	Ñ	¥	165	õ
&	38	f	102	ª	¦	166	µ
'	39	g	103	º	§	167	þ
(40	h	104	¸	“	168	è
)	41	i	105	®	©	169	ú
*	42	j	106	¬	ª	170	û
+	43	k	107	½	«	171	Ù
,	44	l	108	¼	¬	172	ý
-	45	m	109	¡		173	Ý
.	46	n	110	«	®	174	í
/	47	o	111	»	—	175	î
0	48	p	112	¤	°	176	ð
1	49	q	113	¤¤	±	177	ñ
2	50	r	114	¤¤¤	²	178	ò
3	51	s	115	¡	³	179	ó
4	52	t	116	¡¡	‘	180	¶
5	53	u	117	À	µ	181	§
6	54	v	118	Â	¶¶	182	õ
7	55	w	119	À	·	183	÷
8	56	x	120	©	‘	184	ø
9	57	y	121	¤	¹	185	ù
:	58	z	122	¤¤	º	186	ú
;	59	{	123	¤¤¤	»	187	û
<	60		124	¤¤¤¤	¼	188	ü
=	61	}	125	¢	½	189	ý
>	62	~	126	¥	¾	190	þ
?	63	¤	127	¤	¿	191	space
							ÿ
							255

أكواد العملات:

Name	Symbol	Code
Pound	£	0163
Euro	€	0128
Yen	¥	0165
Cent	¢	0162
Dutch Florin	ƒ	0402
Currency	¤	0164

أكواد علامات الترقيم:

Name	Symbol	Code
Copyright	©	0169
Registered	®	0174
Trademark	™	0153
Bullet	•	0149
Ellipsis	...	0133
Section	§	0167
Dagger	†	0134
Double Dagger	‡	0135
En dash	–	0150
Em dash	—	0151
Paragraph (pilcrow)	¶	0182
Left single quote	'	0145
Right single quote	'	0146
Left double quote	“	0147
Right double quote	”	0148
Left single angle quote	⟨	0139
Right single angle quote	⟩	0155
Left double angle quote	«	0171
Right double angle quote	»	0187
Inverted exclamation	¡	0161
Inverted question mark	¿	0191

الرياضيات:

Name	Symbol	Code
Multiply	\times	0215
Divide	\div	0247
Minus (not hyphen)	$-$	8722
Degree	$^\circ$	0176
Plus/Minus	\pm	0177
Not	\neg	0172
Micro	μ	0181
Infinity	∞	8734
Per mille	$\%$	0137
Quarter	$\frac{1}{4}$	0188
Half	$\frac{1}{2}$	0189
Three Quarters	$\frac{3}{4}$	0190
Prime (feet, minutes)	$'$	8242
Double prime (inches, seconds)	$"$	8243
Third	$\frac{1}{3}$	8531
Two Thirds	$\frac{2}{3}$	8532
Sum	Σ	8721
Integral	\int	8747
Approximately equal to	\cong	8773
Not equal to	\neq	8800

(الأكواود التي تنتهي بـ 8 قد لا تعمل على كل البرامج).

لاستخدام أحد المحارف الموضحة بالجداول السابقة اضغط مع الاستمرار على المفتاح Alt مع الضغط على الكود الخاص بالحرف (من اليسار إلى اليمين إذا كان أكثر من رقم).

الجدير بالذكر أن أكواود Alt هي نفسها شيفرات Unicode بعد تحويلها من النظام السادس عشر إلى النظام العشري، وقد تم إيضاح ذلك في هذا الكتاب.

ملحق ج - الكلمات الممحوزة في C#

الكلمات الممحوزة هي مكونات لغة البرمجة ولا يمكنك التصريح عن متغيرات باسم هذه الكلمات.

الجدول التالي يظهر كلمات C# الممحوزة ويفيد ما يلي:

- مكونات الفئات: تمثل الفئة أو مكوناتها البرمجية، أو مكونات الوحدة الأمر.
- مكونات أساسية في البرمجة: الكلمات التي تمثل النخاع الشوكبي للبرنامجه.
- مجالات الرؤية: تضبط إمكانية الوصول للكائنات البرمجية من الوحدات البرمجية المختلفة.
- أوصاف: هذه الكلمات تصف مكونات برمجية فتتغير من وظيفتها أو تعطيها معانٍ أخرى.
- بني تحكم: تحكم بالشرط وما يتعلق به، ويدخل بها بني اقتناص الأخطاء.
- بني التكرار: تنظم عمليات تنفيذ الكود بشكل متكرر.
- أنواع المتغيرات: تتعلق بأنواع التي يمكن للبيانات أن تكون عليها.
- أنواع بيانات خاصة: تتعلق بأنواع البيانات التي يمكن للمبرمج أن يضيفها للبرنامج.
- قيم منطقية: وهي true أو false.
- قيم لا نوع لها: وهي القيمة null.
- عاملات: أدوات ربط بين الكائنات المختلفة.
- عاملات تمرير: تحدد طريقة تمرير المتغيرات عند استدعاء الطرق.

(الجدول لا يتناول جميع الكلمات الممحوزة وإنما جميع ما تم ذكره بالكتاب، هناك بضعة كلمات ممحوزة أخرى لم يتم ذكرها هنا).

الكلمة	المعنى والاستخدام
abstract	مجرد، وهي وصف. الفئات المجردة هي فئات غير موجودة على أرض الواقع وإنما تمثل مفهوماً عاماً لشيء ما، وهي فئات لا يمكن استنساخ كائنات منها.
as	ـ، وهي عامل. تستعمل عند إجراء عمليات على الكائنات. ومن الممكن استخدامها لعمليات تحوي الأنواع.
base	أساس أو قاعدة، وهي من مكونات الفئات. تستعمل للوصول لطرق أو خصائص الفئة الأمر.

Boolean	منطقي، وهي نوع متغير، وهي اختصار لـ <code>.false</code> أو <code>true</code> يحمل قيمة <code>true</code> أو <code>false</code> .	bool
break	كسر أو إيقاف، وهي بنية تحكم. تستخدم لإنهاء أقرب حلقة تكرار أو بنية اختيار <code>switch</code> وذلك عند تحقق شرط ما.	
byte	بایت، وهي نوع متغير. تستخدم للتصریح عن متغير رقمي يحمل قيمة من 0 وحتى 255.	
case	حالة، وهي بنية تحكم. تستخدم في بنية الاختيار <code>switch</code> ، تمثل بنود البنية.	
catch	إمساك، وهي بنية تحكم. تستخدم في بنية رصد الأخطاء، وفيها يتم تنفيذ ما يُرغَب بتنفيذها عند وقوع خطأ ما.	
char	مِحرَف، وهي نوع متغير. تستخدم للتصریح عن متغير حرفی.	
class	فئة أو صنف، وهي مكون أساسي في البرمجة. الفئات هي تعبير عام يصف كينونة برمجية معينة، لها مزايا وصفات تسمى خصائص، وأفعال تسمى طُرُق، وتمر عليها ظروف تسمى أحداث. نشتق منها كائنات تمثل أجزاء البرنامج المختلفة. الفئة هي تمثيل عام مثل "كتاب" و"وسيلة نقل"، أما الكائن فهي تمثيل خاص مثل "كتاب الرياضيات" و"طائرة". (الفئات مفاهيم معنوية أما الكائنات فهي مفاهيم محسوسة).	
const	ثابت، وهي وصف. تستخدم عند التصریح عن الثوابت.	
continue	استمرار، وهي بنية تحكم. تستخدم لتمرير التنفيذ للقيمة التالية للحلقة المغلقة، أي تتجاهل تنفيذ محتوى الحلقة الحالي من أجل شرط معين.	
decimal	عشري، وهي نوع متغير. تستخدم للتصریح عن متغير عشري ذو دقة عالية. يستخدم هذا النوع من المتغيرات مع الأموال والمعاملات المصرفية.	
default	افتراضي، وهي بنية تحكم. تستخدم لضبط الحالة الافتراضية والتي سيتم تنفيذها في حال لم يتم تنفيذ أي حالة <code>case</code> من حالات بنية الاختيار <code>switch</code> .	
delegate	تفويض، وهي وصف. تستخدم عند إنشاء الأحداث Events. وهي إجراءات مرجعية reference type تقابل في C++ المؤشرات، إلا أنها آمنة (المؤشرات في C# هي أ��واد غير آمنة).	

افعل أو قم بـ أو نفذ، وهي بنية تكرار. تستخدم لإنشاء بنية تكرار تُنفذ مرة واحدة على الأقل.	do
مضاعف، وهي نوع متغير. تستخدم للتتصريح عن متغير عشري مضاعف المجال. يستخدم هذا النوع مع القيم من المتغيرات مع الحسابات العلمية.	double
آخر، وهي بنية تحكم. تستخدم في بنية الشرط، تمثل الحالات المختلفة من الشرط.	else
تعداد، وهي نوع بيانات خاصة. تستخدم للتتصريح عن مجموعة من الثوابت المتعلقة ببعضها، وتأخذ عناصرها قيمًا رقمية صحيحة بشكل افتراضي بحيث أول عنصر يأخذ القيمة 0، الثاني 1 والثالث 2 وهكذا دواليك. يمكن تعديل ترتيب العناصر أو ترقيم أكثر من العنصر بنفس الرقم.	enum
حدث، وهي وصف. تستخدم لإنشاء أحداث جديدة، والأحداث هي الظروف التي تمر على كائنات برنامحك.	event
خطأ، وهي قيمة منطقية. تستخدم مع المتغيرات المنطقية bool لجعلها تأخذ قيم النفي، ومعناها أن المتغير لم يتحقق، بمعنى لم يحصل.	false
أخيرًا، وهي بنية تحكم. تستخدم في بنية رصد الأخطاء، وفيها يتم تنفيذ ما يُرغب بتنفيذه سواءً أُوجد خطأ أم لم يوجد.	finally
عائم، وهي نوع متغير. تستخدم للتتصريح عن متغير عشري ذو دقة عالية، وهو متخصص للتعامل مع الأعداد العشرية. وسميت بهذا النوع "عائم" أو "float" لأنها مخصصة للتعامل مع الفواصل العائمة حيث يوجد في الحاسوب ما يسمى بوحدة الفاصلة العائمة. ووحدة الفاصلة العائمة (ويرمز لها بـ FPU اختصاراً لـ Floating Point Unit): هي وحدة في الحاسوب مصممة لتنفيذ العمليات الرياضية على أعداد الفاصلة العائمة مثل الجمع والطرح والضرب والقسمة والجذر التربيعي.	float
من أجل أو حتى، وهي بنية تكرار. تستخدم لإنشاء بنية تكرار تُنفذ التعليمات بدءاً من قيمة ابتدائية وانتهاءً بقيمة نهائية وذلك بفارق محدد بين كل قيمة وأخرى حتى الوصول للقيمة النهائية.	for
من أجل كل العناصر، وهي بنية تكرار. تستخدم للتعامل مع جميع عناصر مجموعات البيانات مثل المصفوفات واللواحق وغيرها.	foreach

اذهب إلى، وهي بنية تحكم. تستخدم لإنشاء بنية تحكم بسيطة بالشكل: اذهب إلى السطر الفلاني، عادةً ما توضع داخل بنية شرط لتصبح بالشكل: إذا تتحقق الشرط الفلاني اذهب إلى السطر الفلاني.	goto
إذا، وهي بنية تحكم. تستخدم لإنشاء بنية تحكم بالشكل: إذا تحقق شرط ما نفذ أكواود محددة. كما يمكن إضافة شروط أخرى للبنية عن طريق الكلمة else. تأخذ بنية التحكم if قيمةً من نوع bool لذلك فكل ما يدخل في شروطها يتم تحويله إلى bool وذلك عبر الروابط.	if
في، وهي معامل. تستخدم لربط متغير ما بعناصر مجموعات البيانات. وتستخدم مع foreach.	in
عدد صحيح، وهي نوع متغير، وهي اختصار ل Integer. تستخدم للتصریح عن متغير عددی صحيح.	int
واجهة، وهي مكون أساسی في البرمجة. تستخدم عند الوراثة المتعددة (أن ترث الفئة أكثر من فئة واحدة).	interface
داخلي، وهي مجال رؤية. تستخدم من أجل منع الوصول إلى الكائنات والمتغيرات إلا من خلال ملف التجميع assembly فقط.	internal
هو أو يكون، وهي معامل. تستخدم للتعامل مع الكائنات.	is
طويل، وهي نوع متغير. تستخدم للتصریح عن متغير عددی صحيح مجاله كبير.	long
مجال أسماء، وهي مكون أساسی في البرمجة. تستخدم لإنشاء مجال أسماء يحوي مجموعة من الفئات تمثل كینونة برمجية متكاملة.	namespace
مثلا: الفئة هي يد ورأس وقلب و...، مجال الأسماء هو إنسان والذي يتكون من مجموعة من الفئات (مجموعة من المكونات) والتي قلنا أنها فئات في بداية المثال.	
جديد، وهي معامل. تستخدم عند استنساخ كائنات جديدة أو إعادة استنساخها.	new
فارغ أو لا شيء، وهي قيمة لا نوع لها. تستخدم لعدم إعطاء المتغيرات قيمة محددة.	null
كائن، وهي نوع متغير. تستخدم للتصریح عن بيانات من نوع كائنات.	object

خارج، وهي معامل تمرين. تستخدم عندما يراد إرجاع قيم بالمرجع دون أن تحمل المتغيرات المُمَرّرة قيم.	out
تجاهل أو إلغاء، وهي وصف. تستخدم عند الحاجة وجود فئة في الفئة الأم يراد استخدامها.	override
متغيرات، وهي معامل. تستخدم عند تمرير مصفوفة بعد غير محدد من الوسطاء.	params
خاص، وهي مجال رؤية. تستخدم لجعل إمكانية الوصول للمتغيرات والكائنات من خلال الوحدة البرمجية الحاوية سواءً أكانت الوحدة البرمجية هي بنية أو حلقة أو طريقة أو فئة. وهي أقل أنواع مجالات الرؤية سماحيةً، وهي نوع التعريف الافتراضي.	private
محمي، وهي مجال رؤية. تستخدم لجعل إمكانية الوصول للمتغيرات والكائنات محصورة بين الفئة التي تم التصريح عن المتغيرات والكائنات فيها، ولل濂يات المشتقة من هذه الفئة.	protected
عام، وهي مجال رؤية. تستخدم لجعل إمكانية الوصول للمتغيرات والكائنات ممكنة من أي جزء من أجزاء برنامجك.	public
مرجع، وهي معامل تمرين، وهي اختصار ل reference. تستخدم عندما يراد إرجاع قيمة بالمرجع، أي أن قيمة المتغيرات الممررة ستتغير وفق ما سيتم تنفيذه في الطريقة الممرر إليها هذه المتغيرات. يجب أن تحوي المتغيرات الممررة بالمرجع على قيمة وإلا فاستخدم الكلمة out.	ref
إرجاع، وهي من مكونات الطرق. تستخدم لإرجاع قيمة في الطرق التي تعيد قيمة (التوابع).	return
مغلق، وهي وصف. تستخدم لجعل الفئات مغلقة، لا يمكن الوراثة منها (عقيمة).	sealed
قصير، وهي نوع متغير. تستخدم للتصريح عن متغير صحيح مجاله صغير.	short
ستاتيكي، وهي مجال رؤية. تستخدم لجعل إمكانية الوصول للمتغيرات والكائنات من خلال الفئات والطرق التي تحمل الكلمة static.	static
سلسلة، وهي نوع متغير. تستخدم للتصريح عن متغير نصي.	string

هيكل أو بناء، وهي مكون أساسي في البرمجة ونوع بيانات خاصة. تستخدم لإنشاء هيئات البيانات.	struct
اختيار، وهي بنية تحكم. تستخدم لإنشاء بنية اختيار مكونة من عدة حالات يتم إنشاؤها عبر الكلمة case.	switch
هذه، وهي من مكونات الفئات. تستخدم للوصول لمحتويات الوحدة البرمجية الحاوية سواءً أكانت الوحدة البرمجية بنية أو حلقة أو طريقة أو فئة.	this
رمي، وهي بنية تحكم. تستخدم لإطلاق الأخطاء.	throw
صحيح، وهي قيمة منطقية. تستخدم مع المتغيرات المنطقية bool لجعلها تأخذ قيم الإيجاب، ومعناها أن المتغير تحقق، بمعنى أنه حصل.	true
حاول، وهي بنية تحكم. تستخدم لمحاولة تنفيذ الكود، فإذا كان هناك خطأ تم الإمساك به في بنية catch.	try
باستخدام، وهي من مكونات الفئات. تستخدم لتضمين مكتبة ما ضمن برنامجك أو لتضمين إحدى الكائنات في طريقة ما.	using
وهمي، وهي وصف. تستخدم مع override لاستخدام طرق فئات أخرى.	virtual
إجراء، وهي مكون أساسي في البرمجة. تستخدم لإنشاء طرق لا تعيد قيمة.	void
بينما أو طالما، وهي بنية تكرار. تستخدم لتكرار تنفيذ كود معين طالما أن شرط ما متحقق.	while

ملحق د - رموز الكائنات في الفيجوال ستوديو

المعنى	الرمز
مجال أسماء namespace: وحدة برمجية متكاملة مكونة من مجموعة من المكونات البرمجية الجزئية هي فئات ومجالات أسماء أخرى.	{}
فئة class: وحدة برمجية - أصغر من مجال الأسماء - مكونة من مجموعة من الطرق والخصائص وتمر عليها مجموعة من الأحداث.	✚
واجهة interface: عنصر برمجي يستخدم لتوريث الفئات من أكثر من فئة واحدة، فالفئات وفق C# لا يمكن أن ترث من أكثر من فئة.	↔
طريقة أو وظيفة method: وهي التوابع والإجراءات، وهي الأفعال التي يمكن للفئات أن تقوم فيها.	◎
خاصية Property: وهي الصفات التي يمكن للفئات أن تتحلى بها.	PropertyParams
متغير Variable: وهو الخلايا الدماغية في برنامجك، فيه معلومات عن البرنامج ومعطياته وما إليه ونتائجها.	●
حدث Event: الظروف التي تمر على الفئات.	⚡
مفوض delegate: عنصر برمجي يقوم بإنشاء الأحداث، وهو عبارة عن طريقة method.	delegat
هيكل بيانات أو تراكيب struct: هو نوع بيانات خاص (يتكون من أنواع البيانات العاديّة والخاصّة).	---
معدّد enum: هو نوع بيانات خاص، رقمي، يتكون من عدة ثوابت ذات معانٍ معينة متعلقة ببعضها، يتم الإشارة إلى كل عنصر فيها بقيمة رقمية تبدأ من 0، ويمكن تغيير قيمة البدء. كما يمكن أن يكون لأكثر من العنصر القيمة الرقمية نفسها.	enum
ثابت Constant: قيمة ثابتة معرفة مسبقاً في خصائص الفئات، مثلاً $\pi = 3.14$ هو ثابت معرف مسبقاً في الفئة Math باسم Pi.	Constant
خطأ Error: أنت أدرى.	✖

راجع ملحقات الكتاب، الملحق ج - الكلمات المحجوزة في C# لمزيد من التفاصيل حصول بعض المفاهيم.

المراجع

المؤلف أو المصدر	المراجع
م. أحمد جمال خليفة	كتاب "C# Keywords"
خالد السعداني	كتاب "خطوة بخطوة مع فيجوال ستوديو 2008"
	كتاب "سييلك إلى تعلم لغة C#.Net"
	كتاب "سييلك المختصر إلى تعلم لغة C# برمجة الواجهات"
	كتاب "دليل المختصر - الهجرة من الفيجوال بسيك إلى الفيجوال سي شارب والعكس."
KSA	كتاب "الشرح الوافي لتعلم لغة SQL من نبعها الصافي"
زياد السعدي	كتاب غير معنون فيه أساسيات SQL
م. أمل منا	كتاب غير معنون فيه سلسلة أساسيات C#
د. محمد دباس الحميد	كتاب "برمجة (1)، لغة بيزيك المرئية للمهندسين"
د. م. عمر قطاطية	كتاب "لغات برمجة، C++"
طه سوافت	كتاب "البرمجة (1)، أساسيات البرمجة بلغة C++"
OMS	كتاب "Languages Programming with C#"
م. محمد كمال أحمد و م. شريف محمد رضا	كتاب " دوره كن خبيراً في لغة C#"
م. حسام الدين الرز	كتاب "الإبحار في لغة C#.Net - ج 2"
قناة خالد السعداني	قائمة تشغيل " دوره السي شارب كاملة" على يوتيوب
Foxlearn	قائمة تشغيل "C# Toturials" على يوتيوب
Sachin Paul	قائمة تشغيل "C# Tutorials From Lynda.com" على يوتيوب
Darren Lee	فيديوهات مختلفة
C# Examples	أمثلة من تطبيق أندرويد
	موقع https://www.w3schools.com/sql/
	موقع https://ss64.com/

بالإضافة إلى مواضيع مختلفة من موقع <https://www.c-sharpcorner.com/> و <https://www.google.com/>، طبعًا <https://stackoverflow.com/> هو الأكب الروحي للمواقع لا ننساه. تم كتابة أسماء الكتب كما وردت في النسخ الإلكترونية المطروحة على الانترنت. بإمكانك تحميل هذه الكتب - الإلكترونية فقط - من خلال التلغرام عبر قناتي.

رسالة الكتاب:

```
C#Book book = new C#Book();
book = this.Book;
If(book.IsHelpful)
{
    YouTube youtube = new Youtube("Hasan M. al-Fahl");
    FaceBook facebook = new FaceBook("Eng27 – Programming");
    Telegram telegram = new Telegram("@Eng27Channel");
    telegram.Join();
    facebook.Like();
    facebook.SeeFirst = true;
    youtube.Subscribe();
    youtube.GetNotifications = true;
    for (int i = 0; i < youtube.Videos.Count; i++)
    {
        youtube.Videos[i].Like();
        youtube.Videos[i].Comment();
        youtube.Videos[i].Share();
    }
}
//By Eng27 – Hasan M. al-Fahl
```

اعمل لآخرتك قبل أن يأتي هذا اليوم:



خُصّص دعوة يومية للناس، اترك بصمةَ خير في حياة بعضهم، ساعد من غير انتظار مقابل، كن عوناً للناس..

يقول أحد دكاتري في الجامعة: أنْ أتعلم كل يومٍ معلومة واحدة فقط يعني أنني سأتعلم 365 معلومة خلال عام!

عدّل المقوله السابقة لتجعلها: "لأنْ أساعد كل يوم شخصاً واحداً فقط يعني أنني سأساعد 365 شخص خلال عام!"