الخطوات الأولى

مع فيجوال سي++

First Steps With Visual C++ 6

تأليف
عمار ابراهيم

تقـــــــــديم

بسم الله الرحمن الرحيم والحمد لله رب العالمين وصل اللهم وسلم وبارك على سيدنا محمد وعلى آله وصحبه زنة عرشك وعدد معلوماتك ومداد كلماتك كلما ذكرك وذكره الذاكرون وغفل عن ذكرك وذكره الغافلون.

أما بعد:إخوتي الكرام في كل مكان من العالم أقدم لكم نفسي وأعرفكم بشخصيتي:
الاسم: ياسين
اللقب: عابر
السن: 31
الحالة العائلية: متزوج وأب لطفلين
العنوان: حي المنازل الجاهزة رقم 27 دائرة قورارية ولاية تيبازة الجمهورية الجزائرية

أحببت البرمجة وعشقتها حتى النخاع احترفت فيجوال بيسيك ثم انتقلت إلى فيجوال سي ++ أسمع الناس يتحدثون كثيرا عن لغة الدوت نت نعم بالفعل إنها آخر ما تطورت إليه البرمجة ولكن أقول لنفسي قبل أن أقول لغيري ماذا فعلتُ بفيجوال بيسك ماذا أفعل بفيجوال سي ++ أأترك كل جهودي تذهب سدى،هم أنشأوا آلاف البرامج بأبسط اللغات ثم اقتضتهم الضرورة حتى وصلوا إلى ما وصلوا إليه الآن.
أقول لكل مبرمج حارَ إلى أين يتجه تمسك بما تعرفه واحترفه مادام يلبي رغباتك ويفي بأغراضك فستأخذك الضرورة بلا شك إلى مستوى أعلى.
إخوتي الكرام أقدم كتابي هذا إلى كل من يعشق لغة الأصل اللغة التي تصنع المعجزات إنها لغة السي ++
أخي الكريم القارئ أهدي إليك هذا الكتاب من خالص قلبي كما أهديت لك من قيل كتاب " تعلم البرمجة بواسطة MFC " وأول شيء أطلبه منك الدعاء الخالص.
ثم أطلب منك أخي الكريم إذا قرأت كتابي هذا وانتفعت به وتأكدَ في نفسك أن صاحبه يستحق أن يُكرَم أن تعينني إن كنت قادرا بنصيب من المال بما تراه مناسبا والله يعلم انه لولا الضرورة الماسَّة ما قلت هذا الكلام والله لا يضيع أجر من أحسن عملا.

يمكنك إرسال الإعانة عن طريق رقم الحساب البنكي التالي:
442-051941-50-201-0-33
بنك التنمية الريفية
دائرة قورارية
ولاية تيبازة
الجزائر
أو إرسال حوالة بريدية إلى العنوان التالي
عابر ياسين
حي المنازل الجاهزة رقم 27
دائرة قورارية
ولاية تيبازة
الرمز البريدي:42135
الجزائر

ترقبوا الجزء الثاني من هذا الكتاب والذي هو في طور الإنجاز
لكافة الاستفسارات إليكم العنوان البريدي vsbelk@hotmail.fr

أخي الكريم قبل أن تبدأ بقراءة هذا الكتاب يستحسن أن يكون عندك إلمام قليل بالبرمجة عموما وبلغة السي خصوصا إن لم تكن كذلك يرجى قراءة الكتاب "لغة السى" المرفق مع هذا الكتاب وقراءة كتابي الأول "**تعلم البرمجة بواسطة MFC**"
هذا الكتاب مازال طور التصميم والتنقيح وهذه نسخة أولية للاطلاع على آراء الإخوة.
وكما يقال لكل جواد كبوة ولكل سيف نبوة ولكل عالم هفوة وأنا لي هفوات فضلا على أني لست عالما وإنما متطفلا وأرجو مِن كل من اطلع على هذا الكتاب ورأى خطأ أن ينظر إليه بعين الرضا ويعلمني به على بريدي وله مني جزيل الشكر.

إذا قرأت كتابي وانتفعت به * فاحذر وقيت الردي من أن تغيره
ودعه لي سالما إني شغفت به * لولا مخافة كتمَ العلم لم تره

هذا الكتاب موجه للمبرمجين المبتدئين أو ذوي خبرة في البرمجة بشكل عام يريدون استعمال **Visual C++** لإنشاء برامج متوافقة مع ويندوز ، يحتوي هذا الكتاب داخل كل فصل من فصوله على عدة أمثلة تطبيقية لتمرينك وترسيخ فهم الفصل في ذهنك وتحسين مهاراتك في **Visual C++** .

الفصل الأول:

Visual C++ 6

- **فهرست الفصل:**
- **مكونات Visual C++ 6**
- **تشغيل Developer Studio**
- **إنشاء برنامجك الأول في Visual C++**
- **إنشاء برنامج نوافذي باستعمال المعالج السحري AppWizard**

مكونات Visual C++ 6.

Visual C++ 6 هو نسخة جديدة من مترجم مايكروسوفت C++ ، ولكن لا يحتوي فقط على المترجم بل على كل المكتبات والأمثلة والتعليمات الضرورية لإنشاء تطبيقات فعالة متوافقة مع ويندوز.

محيط التطوير المتكامل:
أدوات البرمجة المرفقة مع بيئة التطوير:
المعالجات التي يوفرها Visual C++ لتوليد الشيفرة:

علاوة على أدوات إزالة العلل وأدوات تحرير وإنشاء الموارد resources فإن **Visual C++** يضع بين يديك عددا من المعالجات التي توفر للمبرمج جهدا كبيرا وتقوم بمهام كثيرة نيابة عنه لإنشاء برامج ويندوز.
إليك المعالجات الكثير الاستعمال لدى مبرمجين **Visual C++**.

- **AppWizard** أو (MFC AppWizard) . لإنشاء هيكل برنامج ويندوز فإن هذا المعالج السحرى يضع بين يدك ثلاثة أنواع من البرامج النوافذية ، التطبيقات ذات المستند واحد SDI والتطبيقات متعددة المستندات MDI وكلا هذين النوعين يرتكزان على الهندسية Document/View والنوع الثالث هو التطبيقات المرتكزة على النوافذ عادية Simples Dialogs ، سوف ترى في هذا الفصل كيفية استعمال هذا المعالج لإنشاء برنامج بسيط.
- **Class Wizard**. أهمية هذا المعالج تظهر بعد إنشاء هيكل التطبيق فهو يساعدك على إنشاء الخلايا class وتعريف الأحداث ، كما أنه يدير عملية ربط أدوات التحكم Controls بالمتغيرات وغير ذلك مما ستعرفه في الفصل الرابع.
- **ActiveX ControlWizard**. هذا المعالج تظهر أهميته عند إنشاء أدوات أكتيف إكس التي هي عبارة عن مجموعة من الوظائف وأدوات التحكم التي تتم برمجتها ليتم استعمالها كمكملات وأدوات مساعدة ضمن برامج أخرى سواء في إطار عمل **Visual C++** أو بيئة أخرى مثل Visual Basic أو Delphi وغيرها سترى هذا لاحقا ربما في نهاية الكتاب.

مكتبات MFC:

يتضمن **Visual C++** الإصدار السادس من مكتبات MFC اختصار Microsoft Foundation Classes هذه المكتبات تسهل البرمجة لويندوز فأنت عندما تستعمل هذه المكتبات تربح الكثير من الوقت والجهد أو قل أنك تقطع بها عقبة كئودا فهي توفر لك أوات التحكم والخطوط وقواعد البيانات وغيرها ، وعندما تستعمل هذه المكتبات فاعلم أنك تستعمل شيفرات منقحة قد صممت لتعمل بكفاءة من طرف مبرمجين ذوي كفاءة عالية وخبرة متقدمة ويجب عليك أن تضع في اعتبارك أنه عندما تحدث شركة مايكروسوفت هذه المكتبات وتصدر إصدارا جديدا فإن برامجك التي صممتها بواسطة هذه المكتبات سوف تكون متوافقة مع أي إصدار جديد.

تشغيل Visual C++ 6:

لتشغيل **Visual C++ 6** انقر على الأيقونة **Microsoft Visual C++ 6.0** سوف تجد هذا الاختصار ضمن المجلد Microsoft Visual Studio 6.0 وللوصول إلى هذا المجلد انقر على الزر ابد في شريط مهام ويندوز ثم أشر إلى البرامج ومن القائمة الفرعية أشر إلى المجلد المذكور ثم انقر على الاختصار يتم تشغيل برنامج **Visual C++ 6** .

عند تشغيل **Visual C++ 6** تظهر لنا نافذتان:

- نافذة المشروع على اليسار تسمى Workspace هذه النافذة تعرض معلومات خاصة بالمشروع الحالي.
- ونافذة على اليمين تعرض فيها ملفات البرمجة والموارد وغيرها.

كما أن **Visual C++ 6** يحتوي كبقية البرامج القياسية الأخرى على القوائم وأشرطة الأدوات وعناصر أخرى.

محرر شيفرة Visual C++ 6:

للعلم فإنه يمكنك بناء تطبيق بأبسط محرر مثل مفكرة ويندوز إلا أن لحد الان لم أر من فعل ذلك لأن لغات البرمجة تقدم محررات خاصة فيحتوي **Visual C++ 6** على محرر للشيفرة ذي ميزات عالية وقدرات فائقة تستطيع بواسطته كتابة وتنقيح واختبار الشفرة لإنشاء برامج ويندوز.

وفيما يلي أهم الميزات التي يقدمها هذا المحرر:

- تمييز كل نوع من الكلمات بلون مميز فمثلا الكلمات الأساسية مثل Public تكون بالأزرق والتعليقات بالأخضر ويمكنك عبر نافذة الخيارات وفي الشريحة Format اختيار اللون الذي يروق لك لكل نوع.
- تنظيم وإزاحة أسطر والكتل في المحرر تلقائيا لتسهل قراءة الكود .
- إتمام كتابة بعض الكلمات تلقائيا فمثلا عند كتابة اسم كائن أو خلية معرفة مسبقا وبعد كتابة النقطة فإن المحرر يعرض قائمة بجميع الخصائص والوسائل التي يملكها الكائن أو الخلية ومن ثم تستطيع اختيار الكلمة التي تريد كتابتها لتوفر على نفسك الوقت وتضمن سلامة الإملاء، ويمكنك الوصول إلى هذه القائمة عن طريق الضغط على مفتاح التحكم ومفتاح المسافة.
- المساعدة حول وسيطات الدوال فمثلا عند كتابة دالة ومباشرة بعد فتح القوس يظهر تلميح أصفر يبين نوع الوسائط الضرورية والاختيارية التي يمكن تمريرها للدالة.
- الوصول إلى المساعدة المباشرة أو على الخط حول الكلمة المحدد ة وذلك عن طريق الضغط على مفتاح التعليمات F1.
- إمكانية سحب وإلقاء الجمل والكلمات.
- إمكانية النسخ Ctrl+c والقص Ctrl+x واللصق Ctrl+v والتراجع Ctrl+z.
- إمكانية معرفة قيم المتغيرات وذلك بوضع المؤشر عليها والانتظار لثواني حيث يظهر تلميح يبين قيمة هذا المتغير وذلك أثناء التشغيل.

إنشاء برنامجك الأول في Visual C++ 6.

1- إنشاء المشروع:

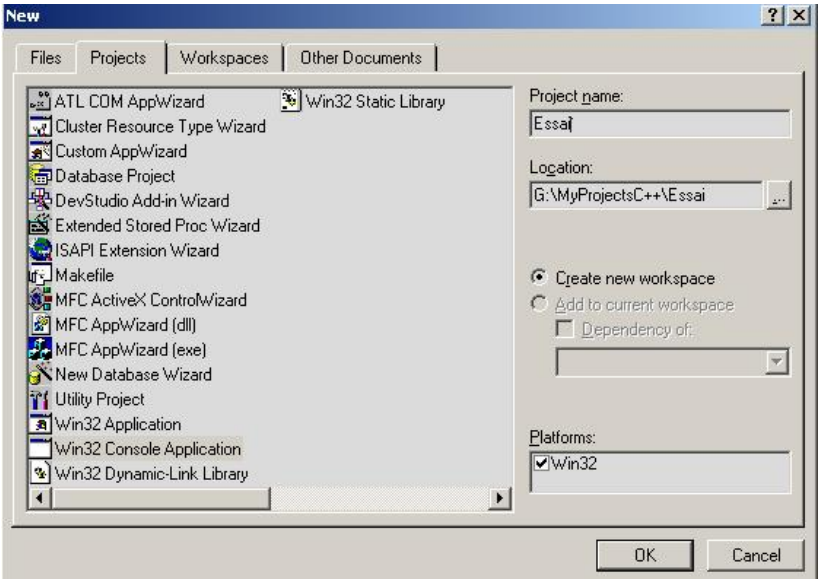
ستكتب برنامجك الأول في وضع MSDOS أو Console.

وظيفة هذا البرنامج عرض رسالة ترحيب على المستخدم "Bonjour".

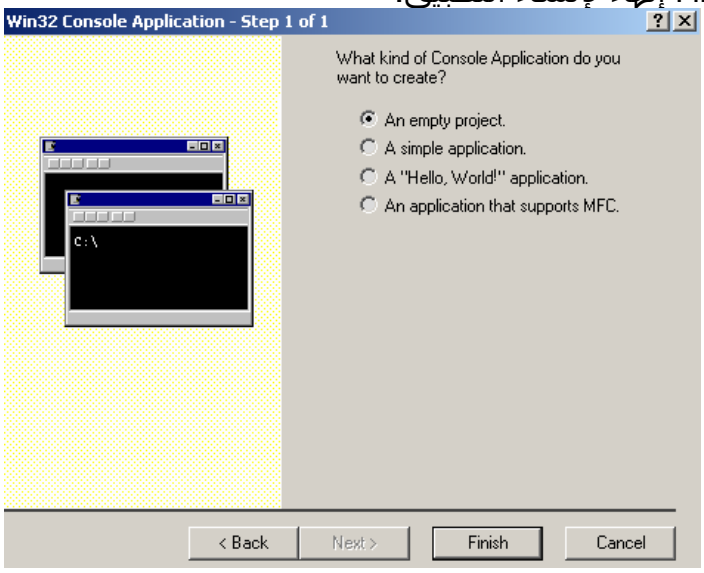
تصميم البرامج في وضع Console أسهل بكثير من تصميم البرامج النوافذية Windows.

ولإنشاء هيكل البرنامج اتبع الخطوات التالية:
ملاحظة: قبل إنشاء المشروع أنشئ المجلد C:\MyProjectsC++ لتحفظ فيه مشاريعك ويسهل الوصول إليه عوضا عن المجلد الافتراضي.

1- اختر أمر **File | New** من القائمة الرئيسية تظهر نافذة New.
2- حدد الشريحة **Projects** ثم انقر على الأيقونة **Win32 Console Application** في القائمة التي على اليسار.
3- أدخل كلمة Essai كاسم للمشروع، المجلد الذي يحفظ فيه المشروع سيتم إنشاؤه من قبل المعالج وفقا لاسم المشروع ومساره مبين في الخانة التي تحت اسم المشروع انقر على الزر الذي على اليمين وغير المسار إلى المجلد C:\MyProjectsC++ الذي أنشأته للتو انظر الشكل 1.2 .



4- انقر على الزر Ok للاستمرار.
5- يظهر معالج يطلب منك تحديد نوع مشروعك انظر شكل 1.3 حدد الخيار الأول An Empty Project مشروعا فارغا.
6- انقر على زر Finish إنهاء لإنشاء التطبيق.



كتابة كود المشروع:

المرحلة التالية هي كتابة الكود وهي أهم شيء في المشروع ، قائمة الكود 1.1 التالية قصيرة جدا ولكنها تحتوي على عناصر يكثر تداولها في المشاريع ويستلزم معرفتها.

قائمة 1.1 : برنامج بسيط في وضع Console

```
// برنامج الترحيب
#include <iostream>
using namespace std;
int main()
{
	cout << "Bonjour" << endl;
	return 0;
}
```

افتح مستندا جديدا من نوع File source ثم اكتب الكود تماما كما يظهر في القائمة 1.1 . ولفتح المستند المذكور أمامك طريقتان.

- انقر الزر New Text في شريط الأدوات .
- حدد الأمر File | New ثم اختر الأيقونة File Source c++ من القائمة ضمن الشريحة Files اكتب اسم الملف في الخانة File Name وليكن Essai تأكد من تحديد الخيار Add To Project ثم انقر Ok.

حفظ المستند:

إذا كنت قد فتحت المستند عن طريق الزر New Text في شريط الأدوات فيستحسن حفظه قبل كتابة الكود بالامتداد cpp لتستفيد من ميزات محرر الكود كالألوان والإتمام التلقائي وغيرها وعلى كل حال احفظ المستند باسم Essai.cpp.

إنشاء الملف التنفيذي exe:

الملف التنفيذي هو الملف النهائي الذي ينشئه مترجم c++ ليكون مستقلا ويعمل في أنظمة التشغيل المتعددة ويكون بالامتداد exe.

ولصنع الملف التنفيذي اختر الأمر Build | Build Essai.exe في شريط القوائم أو اضغط مفتاح الاختصار F7، إذا كنت قد كتبت القائمة 1.1 من دون أخطاء فسيتمكن المترجم من إنشاء الملف التنفيذي وسيكون آخر سطر في قائمة الإخراج :

```
Essai.exe - 0 error(s), 0 warning(s)
```

إذا لم يتمكن المترجم من إنشاء الملف التنفيذي لوجود أخطاء فإنه سيعرض عليك في نافذة الإخراج رسائل الخطأ اضغط F4 لتعرف الأسطر المتضمنة للأخطاء، تأكد من عدم وجود أخطاء إملائية ثم أعد المحاولة .

تنبيه : كن شديد الحذر من الأخطاء الإملائية واعلم أن المترجم يفرق بين الحرف الصغيرة والكبيرة فمثلا كلمة Main و main لا تعنيان كلمة واحدة لدى المترجم.

تشغيل البرنامج وتجربته:

لتشغيل البرنامج Essai.exe افتح نافذة موجه الأوامرMSDOS ثم أدخل مسار البرنامج:

```
C:\MyProjectsC++\Essai\Debug\Essai.exe
```

بعد تشغيل البرنامج تظهر كلمة الترحيب Bonjour أمامك.

للعلم فإن **Visual C++ 6** يضع الملف التنفيذي والملفات المتعلقة في مجلد ضمن المشروع اسمه Debug.

إنشاء برنامج نوافذي باستعمال المعالج السحري AppWizard

AppWizard معالج يدير عملية إنشاء مشاريع MFC ويقدم لك خيارات عديدة عن طبيعة المشروع ، وهو أيضا ينشئ الملفات اللازمة لمشروعك ملفات البرمجة ذات الامتداد pp وملفات الرأس ذات الامتداد h وملفات الموارد Resources وغيرها.

AppWizard يتيح لك إنشاء ثلاثة أنواع من المشاريع:

- مشاريع ذات مستند واحد SDI
- مشاريع متعددة المستندات MDI
- مشاريع مرتكزة على نوافذ عادية Dialogs

ويمكنك أيضا تحديد خيارات عديدة تتعلق بهذه الأنواع منها:

- إدراج شريط الأدوات

- شريط الحالة
- شريط أدوات مثل متصفح الإنترنت
- دعم مربعات الحوار الشائعة مثل فتح وحفظ وطباعة
- دعم قواعد البيانات و OLE

ولن يأخذ ذلك سوى دقيقة أو دقيقتان تقضيهما للإجابة على أسئلة المعالج.

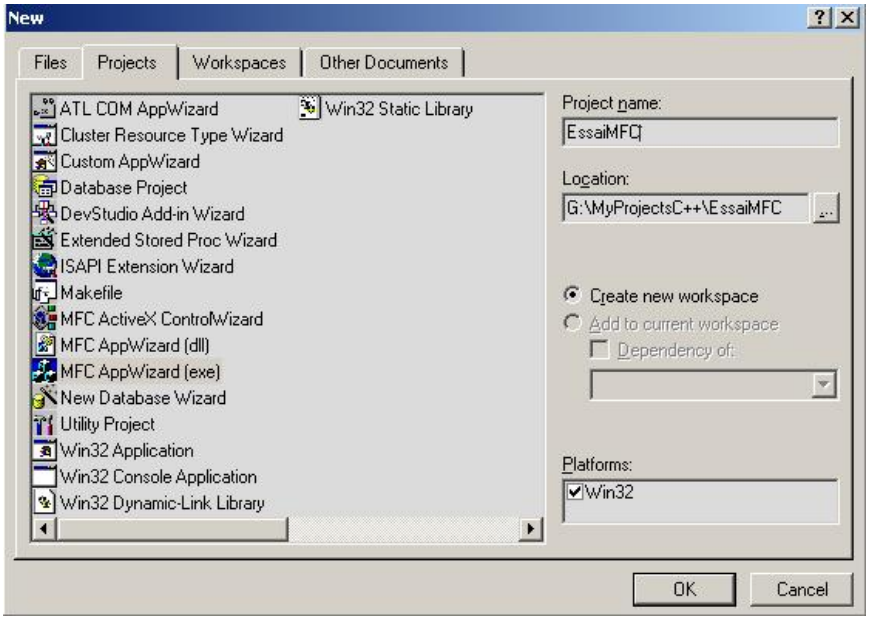
إنشاء تطبيق Windows بواسطة معالج AppWizard:

عموما الخطوات التي يجب اتباعها لإنشاء برنامج Windows هي كالتالي:

1. بناء الهيكل البرنامج بواسطة **AppWizard**
2. إنشاء الموارد الخاصة بالبرنامج النوافذ الصور القوائم
3. إضافة الخلايا والوظائف الخاصة بالأحداث أو الرسائل Messages
4. كتابة الوظائف والكود اللازم للبرنامج
5. تفسير البرنامج وتجربته بواسطة المنقح الداخلي ل **Visual C++ 6** لإزالة العلل.

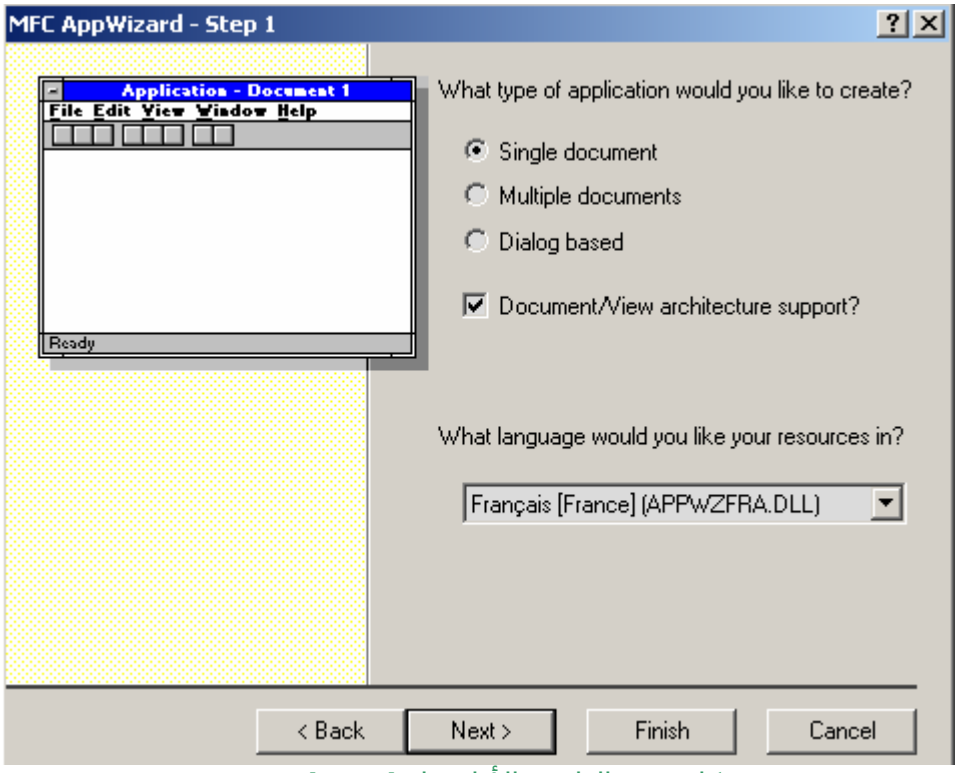
لتشغيل **AppWizard** ولإنشاء مشروع Windows الأول افعل ما يلي:

1. اختر الأمر File | New لإظهار نافذة جديد New
2. افتح الشريحة Projects لاستعراض قائمة بجميع أنواع المشاريع المتوفرة
3. لإنشاء مشروع يستخدم مكتبات MFC حدد الأيقونة MFC AppWizard (exe)
4. أدخل EssaiMFC كاسم للمشروع في خانة Project Name كما هو مبين في الشكل 1.4



شكل 1.4 إنشاء برنامج Windowsب AppWizard

5. تأكد من تحديد خانة الخيار Create New Workspace ثم انقر Ok
6. النافذة الأولى من معالج **AppWizard** تعرض عليك اختيار نوع المشروع كما ذكر سابقا واختيار اللغة التي تناسبك كما هو مبين في الشكل 1.5



شكل 1.5 النافذة الأولى ل **AppWizard**

7. استعرض النوافذ الستة التي يعرضها المعالج وذلك عن طريق الزرين Next و Back بالنسبة لمشروعنا حدد الخيار الأول في النافذة الأولى ودع الخيارات الأخرى كما هي ثم انقر الزر Finish.
8. النافذة الأخيرة التي يعرضها المعالج هي قائمة بجميع الخلايا والخيارات التي تم تحديدها والتي سيتم توليدها من طرف المعالج انقر على Ok ليتم إنشاء المشروع.

استكشاف المشروع EssaiMFC:

بمجرد ما يتم المعالج من إنشاء هيكل برنامجك فسيتم عرض معلومات المشروع في النافذة Workspace تحتوي هذه النافذة على ثلاثة شرائح :

- الشريحة الأولى Class View تعرض معلومات عن الخلايا المستعملة في المشروع.
- الشريحة الثانية Resource View تعرض الموارد المتعلقة بالمشروع مثل الصور والأيقونات والقوائم.
- أما الشريحة File View فإنها تعرض كافة الملفات التي يستخدمها المشروع.

إدارة عمليات الإخراج مع خلايا MFC:

AppWizard ينشئ وظيفة تدير عملية الإخراج اسم هذه الوظيفة OnDraw ستجدها في الخلية CEssaiMFCView حتى الآن فإن هذه الوظيفة لا تقوم بشيء فنحن سنقوم بإدخال الكود الذي يطبع تحية الإسلام .

للوصول إلى هذه الدالة وتحرير الخلية CEssaiMFCView افعل ما يلي:

1. انقر على الشريحة Class View في النافذة Workspace على اليسار ثم افتح اسم المشروع EssaiMFC Classes وذلك بالنقر على علامة الجمع + سنجد بعدها على يساره قائمة فرعية بجميع خلايا EssaiMFC سوف تلاحظ أن كل هذه الخلايا تبدأ بالحرف الكبير C وهي طريقة معروفة لجميع مبرمجي مايكروسوفت في تسمية الخلايا ،يحسن إتباعها عند إنشاء خلايا جديدة .
2. انقر علامة الجمع أمام الخلية CEssaiMFCView سترى قائمة فرعية لهذه الخلية بجميع وظائفها ومتغيراتها.
3. انقر نقرا مزدوجا على الوظيفة المسماة OnDraw سيفتح محرر الكود وسينتقل المؤشر مباشرة إلى الوظيفة CEssaiMFCView::OnDraw احذف الثلاثة أسطر الموجود بين الحاضنتين واكتب القائمة 1.2 :

قائمة 1.2: استعمال الدالة OnDraw في عملية الإخراج.

```
void CEssaiMFCView::OnDraw(CDC* pDC)
{
        pDC->TextOut(50,50,"السلام عليكم ",12);
}
```

اكبس المفتاح F7 لتفسير المشروع ، النافذة Build أسفل تعرض أثناء عملية التفسير المهام التي يقوم بها المترجم وعمليةَ التقدم و سيكون آخر سطر:

```
EssaiMFC.exe - 0 error(s), 0 warning(s)
```

تهنئة: لقد أنشأت للتو برنامج Windows و لتشغيل البرنامج اختر الأمر Build | Execute أو اضغط على Ctrl-F5.

أما الطريقة المثلى لتشغيل البرنامج فهي تشغيله باستعمال مزيل العلل Debug انقر على الزر Go في شريط الأدوات أو اضغط على F5.

بعد تشغيل البرنامج ستطبع الدالة OnDraw تحية الإسلام على المنظر View.

كيف تعمل الطريقة OnDraw:

تعمل الطريقة OnDraw باستعمال الوسائط التالية :

الوسيط الأول والثاني يحددان موقع الجملة عموديا وأفقيا بالنسبة لأعلى View ويساره.

والوسيط الثالث يحدد السلسلة المطبوعة

والوسيط الرابع يحدد طول السلسلة

للعلم فإن الدالة OnDraw تعرض السلسلة بشكل ثابت بحيث لا تتغير حدوده عند إعادة تحجيم النافذة في الفصل التالي سوف ترى كيف تطبع سلسلة في وسط النافذة وتبقى آخذة مكانها ولو أعيد تحجيم النافذة.

أسئلة:

1. هل تعلم c++ أمر صعب جدا .
2. هل يمكن أن أكتب برنامجا بواسطة المفكرة

الفصل الثـ ـ ـاني:

إنشاء برنامج C++ بسيط:

فهرس الفصل:

في هذا الفصل ستنشئ برنامجا بسيطا يطلب معلومات من المستخدم ثم يعرضها على الشاشة.

العناصر المتداولة في اللغة C++:

أنواع البيانات الأساسية :

توفر لغة C++ أنواعا مختلفة من البيانات الأساسية كما هو الحال بالنسبة للغات الأخرى تستخدم هذه الأنواع لتخزين المتغيرات وإجراء العمليات ، في الفصول القادمة ستستعمل الكثير من هذه الأنواع.

وإن اختلاف أنواع البيانات له أهمية كبيرة في البرمجة فأنت عندما تصرح متغيرا من نوع محدد تبقى متأكدا أن ذلك المتغير لن يتعامل إلا مع النوع الذي حددته له ويقلل أيضا من احتمالات الخطأ ، ومن هنا أيضا يسهل على المترجم اكتشاف عدم التوافقية.

وقبل التعامل مع أي متغير يجب تصريحه وتعريف نوعه كما في المثال التالي:

```
int      monAge;
```

هذا السطر يصرح ويعرف متغيرا اسمه monAge من نوع عدد صحيح int ، إن تصريح المتغيرات يطلب من المترجم حجز مساحة في الذاكرة لهذه المتغيرات.

ويمكن تصريح أكثر من متغير واحد في نفس السطر وهي عملية حسنة للتفريق بين أنواع المتغيرات كما في المثال التالي:

```
int      monAge, tonAge , ageMaxi;
```

فهذه المتغيرات الثلاثة كلها من نوع واحد وهو int .

أهمية تصريح المتغيرات:

بعض لغات البرمجة مثل Visual Basic تسمح استعمال المتغيرات دون تصريحها ولكون ذلك يؤدي في أغلب الأحيان إلى أخطاء فإن **Visual C++ 6** يجبرك على تصريح المتغيرات قبل استعمالها وبإمكانك أن تصرح المتغير ليكون عاما Public أو محليا Private .

أنواع المتغيرات المختلفة:

أهم أنواع المتغيرات التي يدعمها **Visual C++ 6** هي مايلي:

- bool المتغيرات المنطقية التي تقبل قيمتين فقط صحيح True أو خاطئ False
- char هذا النوع يستعمل عموما لتخزين عددا معتبرا من الحروف فيمكنه تخزين أي قيمة من -128 إلى 127، إذا صرح متغير من هذا النوع على شكلunsigned أي أن يكون تصريحه مسبوقا بهذه الكلمة وتعني عدم وجود الإشارة فالقيم التي يخزنها تبدأ من الصفر إلى غاية 255 أما القيم السلبية فهي غير مقبولة .
- المتغيرات short int أو المتغيرات short تشبه النوع int ولكن قيمها فيما بين 32768- إلى 32767 ،أما إذا كان مصرحا بشكل insigned فالمجال من 0 إلى 65535.
- المتغيرات int تخزن الأعداد الصحيحة أي التي لا تحتوي على الفاصلة العشرية يمكنها أن تحمل أي قيمة صحيحة في المجال 2147483648- إلى 2147483647 أما إذا كان مصرحا بشكل insigned فالمجال من 0 إلى 4294967295.
- المتغيرات long int أو long مثل النوع السابق إلى أن مجالها كبير جدا وقد يمكن أن تحمل قيما هامة.
- المتغيرات float النوع الصغير من المتغيرات الذي يمكنه تخزين أرقاما عشرية مثل 3,14 وتحد الأرقام العشرية في النوع ب 6
- المتغيرات Double مثل النوع السابق إلى أنها بإمكانها تخزين حتى 15 رقم عشري.
- المتغيرات long double مثل النوع السابق.

بعض المتغيرات السابقة يمكن تصريحها على شكل unsigned وفي هذه الحالة لا يمكنها تخزين سوى القيم الصحيحة وليكن في علمك أن تصريح المتغير بهذا الشكل يمكنه أن يخزن أكثر من التصريح العادي .

تسمية المتغيرات:

إن اختيار أسماء المتغيرات والعناصر الأخرى في برامجك شيء مهم فمن الأحسن اختيار الأسماء المعبر التي تبين وظيفة هذا المتغير وتصديرها بالحروف التي تدل على نوع المتغير حتى تسهل قراءتها و تذكر دورها بسهولة فمثلا إذا صرحت متغير من نوع int لتخزين العمر فأفضل اسم له nAge فالحرف n يدل على نوعه وكلمة Age تدل على وظيفته ويمكنك التفريق بين الكلمات إما بالحروف الكبير مثل nAge أو العلامة _ مثل n_age.

إسناد القيم للمتغيرات:

بعد أن تعلن عن المتغيرات يمكنك أن تحفظ فيها البيانات وأسهل طريقة لعمل ذلك هو أن تستخدم علامة = وإليك الطريقة الصحيحة لعمل ذلك:

```
int nAge = 31;
```

ويمكنك إسناد القيم إلى المتغيرات من نوع char على طريقتين الأولى جعل القيمة المسندة بين علامتي تنصيص فيعتبرها المترجم سلسلة نصية ويحوله إلى كود ASCII مثل:

```
char chNom= 'Yacine';
```

أو بدون علامة تنصيص وذلك في الأرقام مثل :

```
char chAge = 31;
```

قراءة المتغيرات:

يمكنك أيضا قراءة المتغيرات

إنشاء برنامج C++ بسيط:

في الفصل السابق أنشأت مشروعا c++ بسيطا يعرض رسالة بسيطة ، المشروع الذي ستنشئه في هذا الفصل يتقدم بك خطوة ستحصل على اسم المستخدم ثم يتم ترحيبه باسمه ولإنشائه اتبع الخطوات التالية:

1. اختر أمر **File | New** من القائمة الرئيسية تظهر نافذة New.
2. حدد الشريحة **Projects** ثم انقر على الأيقونة **Win32 Console Application** في القائمة التي على اليسار.
3. أدخل كلمة Bonjour كاسم للمشروع، وانقر Ok
4. انقر Finish ليتم إنشاء المشروع
5. اختر الأمر File | New وفي الشريحة Files أنشئ ملف برمجة جديد من نوع C++ Source File وأعطه اسم Bonjour.cpp ، تأكد من تحديد خانة الخيار Add To Project ثم انقر Ok.
6. اكتب القائمة التالية في ملف البرمجة.

القائمة 2.1: برنامج بسيط يطلب إدخالا من المستخدم.

```
#include <iostream>
#include <string>
using namespace std;

//طلب اسم المستخدم ثم تخزينه في متغير
// ثم عرضه على الشاشة مع الترحيب
int main()
{
 string nomUse;

 cout << "Comment vous appelez-vous ?";
 cin >> nomUse;
 cout << "Bonjour " << nomUse << " !" << endl;
```

```
    return 0;
}
```

7. ترجم المشروع بالضغط على F7.
8. شغل البرنامج Bonjour.exe افتح نافذة موجه الأوامر MSDOS ثم أدخل مسار البرنامج:
   C:\MyProjectsC++\Bonjour \Debug\Bonjour.exe
9. بعد تشغيل البرنامج تظهر رسالة تطلب منك إدخال اسمك ، بعد إدخال الاسم والضغط على زر الإدخال Enter يعرض البرنامج رسالة ترحب بك.

تحليل البرنامج Bonjour:

كود هذا البرنامج قصير ولكنه يحمل في طياته عناصر هامة يجب معرفتها والإلمام بها، ستجد هذه العناصر في فصول عدة من فصول هذا الكتاب وستجدها أيضا في كثير من المشاريع كالتي تحملها من الإنترنت أو تجدها في كتب أخرى إذن لنلقي نظرة على هذه العناصر:

العنصر *include* :

السطر الأول من البرنامج يطلب من المترجم ربط وتضمين الملف الموجود بين علامتي < > مع البرنامج عند تفسيره

```
#include <iostream>
```

المترجم يتلقى الأمر ويبحث عن ملف الرأس liostream ويربطه مع برنامجك وإنما ربطنا هذا الملف مع برنامجنا لأنه المسئول عن عملتي الإدخال والإخراج فالأمر cout و cun معرفان داخله.

والملفات التي نربطها مع برامجنا على نوعين:

- ملفات موجود داخل لغة c++ وتحاط بين العلامتين < > ولا يذكر معها الملحق h.
- ملفات خارجية تنشئها أنت مثلا وتحاط بالعلامتين "" ويذكر معها الملحق h.

السطر الثاني يدرج المكتبة string وتستعمل هذه المكتبة للتعامل مع السلاسل النصية:

التعليقات:

التعليقات جد هامة للمبرمج فهي تسهل عملية قراءة الكود وتنظيمه لأنه عندما يمر وقت على المشروع ويكثر حجمه يصبح من العسير قراءته وتنظيمه إن لم يكن فيه تعليقات والتعليقات تبتدأ بالعلامة //

الدالة main:

السطر التالي هو بداية الدالة وهي الدالة الرئيسية التي ينطلق منها البرنامج

السطر الأول داخل الدالة main

```
string nomUse;
```

يصرح متغير من نوع string سنستخدم هذ المتغير لتخزين اسم المستخدم

السطر التالي

```
cout << "Comment vous appelez-vous ?";
```

تعرض على المستخدم رسالة تطلب منه اسمه ، يستعمل الرمز << للإخراج على الشاشة.

السطر:

```
cin >> nomUse;
```

يفسح المجال للمستخدم لإدخال اسمه ومن ثم تحفظ في المتغير nomUse ويستعمل الرمز >> للإدخال

السطر:

```
cout << "Bonjour " << nomUse << " !" << endl;
```

تعرض رسالة ترحيب مع الاسم المدخل.

الفصل الثالث:

التعليمات البرمجية ،الدالات ،التركيبات ،والخلايا:

فهرس الفصل:

- **التعليمات البرمجية**
- **الدوال**
- **التركيبات**
- **الخلايـــا**

التعليمات البرمجية:

هي أوامر c++ التي تكتب في أي مشروع ويقرأها المترجم عند تفسير المشروع ويحولها إلى لغة الآلة لينشئ البرنامج التنفيذي.
وتنتهي جميع تعليمات **Visual C++ 6** بالنقطة الفاصلة (;) مثلا السطر الذي كتبته في الفصل الأول :

```
cout << "Bonjour"<< endl;
```

تعليمة تطلب من البرنامج إظهار كلمة الترحيب على الشاشة.

التصريحات:

التصريحات هي نوع آخر من التعليمات يتعلق بتصريح المتغيرات وهي أيضا أوامر للمترجم فالسطر:

```
int monAge;
```

يطلب من المترجم حجز مساحة في الذاكرة للمتغير monAge من نوع int

إسناد القيم إلى المتغيرات:

مثل

```
monAge=31;
```

فالإسنادات أيضا نوع من تعليمات **Visual C++ 6**.

المعاملات:

وهي المستعملة في العمليات الشائعة كالجمع والطرح كما في الجدول التالي:

المعامل	الوصف
+	الجمع
-	الطرح
*	الضرب
/	القسمة

ترتيب المعاملات: يؤدي **Visual C++ 6** العمليات الحسابية بترتيب محدد وصارم فهو يبدأ من اليسار إلى اليمين ويمكنك وضع الأقواس بين العمليات لتجبر **Visual C++ 6** على البدء بالعمليات المحاطة بالأقواس مثل:

```
(a+5)*3;
```

هذه العبارة تجمع قيمة a مع 5 ثم تضرب الناتج في 3.
وهناك حسنة يقدمها **Visual C++ 6** وهي عندما نريد إسناد قيمة لمتغير مع المحافظة على قيمته السابقة مثلا:

```
a=a+5;
```

تختصر ب

```
a+=5;
```

وما قيل في الجمع يقال فيما تبقى من المعاملات.

الدوال:

إن الدالات من أهم الأمور التي ينبني عليها أي مشروع **Visual C++ 6** إنها تسمح بتقسيم البرنامج إلى قطع منظمة وهي أيضا تغنيك عن تكرار كثير من الكود فمثلا إذا كنت تستعمل خمسة أوامر في أكثر من موضع في برنامجك فإنه يمكن جعل هذه الأوامر في دالة ويتم تنفيذها باستدعاء هذه الدالة فقط دون الحاجة إلى تكرار كتابة الأوامر الخمسة.
جميع برامج **Visual C++ 6** تحتوي على الأقل على دالة واحدة والتي تدعى main وهي نقطة انطلاق البرنامج وهذا في البرامج التي تعمل تحت بيئة الدوس ،أما بالنسبة للبرامج النوافذية فإنها تحتوي على دالة مشابهة لها وهي WinMain ولست بحاجة أخي المبرمج إلى إنشاء هذه الدالة لأن المعالج **AppWizard** يتولى ذلك نيابة عنك.

تصريح الدوال:

قبل أن تتمكن من استعمال دالة يجب عليك تصريحها ولتصريح دالة يجب ذكر اسمها ونوع القيمة التي تعود بها وقائمة الوسيطات التي تستعملها الدالة كما هو مبين في هذا المثال:

```
int CalculAge(int nAnneeNaissance);
```

هذا السطر هو تصريح للدالة CalculAge وهي تستعمل وسيطا واحدا وهو (nAnneeNaissance) وهي تعود بقيمة من نوع int عدد صحيح يعني أنه بعد تنفيذ هذه الدالة وتمرير عام الازدياد لها نحصل على عمر الشخص.
ولكن ما كل الدالات تعود بقيمة في هذه الحالة نبين في تصريح الدالة أنها لا تعود بشيء ونكتب void يعني لا شيء.

تلميح:

عموما يتم تصريح الدوال داخل ملف رأس file header والذي ينتهي بالامتداد h.

تعريف الدالة:

بعد تصريح الدالة يجب تعريفها وبناء جسمها وتعريف الدالة مشابه لتعريف الدالة القياسية main ، وهذه هي العناصر التي يتم بواسطتها تعريف الدوال:

- نوع القيمة التي تعود بها الدالة إن كانت تعود بقيمة مثل int وإلا كتبت الجملة void
- اسم الدالة ويراعى في ذلك نفس الشروط لكتابة المتغيرات.
- قائمة الوسيطات التي تستعملها الدالة ،وإن لم يكن للدالة وسيطات تحاط وجوبا بأقواس فارغة.
- وأخيرا جسم الدالة نفسه أي الأوامر والتعليمات التي تقوم الدالة بتنفيذها عند استدعائها ، وجسم الدالة يحاط بقوسي الاحتضان {} .

القائمة 3-1 إنشاء دالة لعرض غمر شخص ما، أنشئ مشروعا جديدا من نوع Console وسمه EssaiFonction ثم اكتب الدالة كما هي مبينة علي هذه القائمة:

```
#include <iostream>
using namespace std;
// تصريح الدالة
void AfficheAge(int nAge);

//الدالة الرئيسية
int main()
{
  //استدعاء الدالة
  AfficheAge(31);
  return 0;
}

//تعريف الدالة
void AfficheAge(int nAge)
```

```
{
 cout << "Bonjour ! J'ai " << nAge << " ans. " << endl;
}
```

لاحظ أن هذه الدالة لا تعود بقيمة لذلك صدَّرناها بالكلمة void.

استدعاء الدوال:

ونعني باستدعاء الدالة تنفيذ الأوامر الموجودة فيها وذلك يتم بكتابة اسمها وتمرير الوسيطات اللازمة لها إن كان لها وسيطات مثلا لاستدعاء الدالة السابقة كتبنا

```
AffricheAge(31);
```

حيث كتبنا اسم الدالة ومررنا لها الوسيط العددي من نفس النوع المصرح به وهو int .

التركيبات :

التركيب متغير عادي يضم أنواعا متعددة من البيانات معرّفة من قبل المستخدم يمكن التعامل معها كوحدة واحدة أو الوصول إلى أي متغير منها.

وفي أغلب الأحيان نستعمل التراكيب عندما نحتاج إلى دمج عدة متغيرات لتشير إلى شيء واحد، وغالبا ما نحتاج ذلك عند التعامل مع السجلات، فمثلا يمكننا إنشاء تركيب لتنظيم بيانات الموظفين هذا التركيب يشتمل على متغيرات تحمل اسم الموظف وعنوانه وراتبه وغير ذلك من المعلومات.

الإعلان عن التراكيب:

يتم الإعلان عن التركيب بواسطة الكلمة الأساسية struct ثم ذِكر اسم التركيب ويفضل أن يكون بالحروف الكبيرة ويُفْصَلُ بين الكلمات بالحرف _ وبعد ذلك عناصر التركيب وتكون محاطة بعلامتي الاحتضان {}وفي الأخير النقطة الفاصلة؛ .

استعمال التراكيب:

بعد تصريح التركيب والإعلان عنه يمكنك استعماله واشتقاق متغيرات منه كما هو الحال بالنسبة للمتغيرات الأخرى.

القائمة 3-2 مثال لتركيب يحتوي على اسم الموظف وعدد سنوات العمل وراتبه الشهري واسمه DATA_CLIENT .

```
#include <stdio.h>

// الإعلان عن التركيب
struct DATA_CLIENT
{
char   name[10] ; //اسم الموظف ونوعه حرفي
int      yearCount; //عدد سنوات العمل ونوعه عدد صحيح
double salary; // الراتب ونوعه عدد حقيقي طويل
};

//الدالة الرئيسية
void main()
{
//تصريح متغير من نوع DATA_CLIENT
DATA_CLIENT     dataClient;
//إسناد قيمة من طرف المستخدم
printf("Enter dataClient.name \n");
scanf(" %s",dataClient.name);
printf("\n dataClient.name= %s",dataClient.name);

//إسناد القيم من داخل البرنامج
```

```
dataClient.yearCount=10;
dataClient.salary=5.3;

printf("\n dataClient.yearCount= %d",dataClient.yearCount);
printf("\n dataClient.salary= %f",dataClient.salary);

//اضغط مفتاح الإدخال لغلق البرنامج
getchar();
getchar();

}
```

الخلايا Classes:

تعريف الخلية:

الخلايا قطع برمجية تسمح بتجميع وتنظيم وكبسلة (تغليف) مجموعة من البيانات والدوال تحت اسم واحد لتقوم بأعمال معينة ذات فائدة، وتسمى النسخة المأخوذة من الخلية كائن Object، ويمكن أن تحتوي الخلايا على متغيرات ودوال (وتسمى طرقا) وأحداثا وإجراءات، وفيما يلي بعض مزايا الخلايا:

- تنظيم مجموعة من الأكواد تحت اسم واحد مما يسهل صيانة البرنامج.
- إمكانية جعل أعضاء members الخلية داخل وخارج الخلية وذلك بتصريح هذه الأعضاء على شكل Public ، أو جعلها محلية فقط أي لا يمكن استعمالها إلا من داخل الخلية وذلك بتصريحها على شكل Private ، كما يمكن إخفاء بعض الأعضاء وجعلها غير مرئية للمستخدم.
- إمكانية كبسلة (تغليف)الخلية لتعمل في مشروع آخر أو حتى في لغة أخرى بأن تصبح ملفا تنفيذيا مستقلا على شكل DLL أو ActiveX.
- تأمين وحماية الخلية بحيث لا يمكن للمستخدم سوى الاستفادة من مزايا الخلية دون الاطلاع على الكود بداخلها.

وبرمجة الخلايا هي ما يسمى (البرمجة كائنية المنحنى) Programming Oriented Object والتي هي أهم ما يجب على المبرمج تعلمه والإلمام به، والخلايا ما هي إلا تركيبات ولكنها تميزت عنها بما ذكرناه سابقا.

تلميح: لو لاحظنا الخلايا القياسية لويندوز لوجدناها كلها تبدأ بالحرف الكبير C تلك هي الطريقة المتبعة لكثير من المبرمجين في تسمية الخلايا لذلك ينصح باتباعها، كما أن متغيرات الخلية الأعضاء تُستهل بالبادئة m_.

مكونات الخلية:

قد علمت أن الخلية عبارة عن كائن يقوم بعمل مفيد، كالإنسان كائن له خصائص مثل اللون والطول ويقوم بأعمال متعددة مثل المشي والكلام وتعتريه أحداث خارجية مثل المرض والفرح وغيرها، الخلايا كذلك لنأخذ مثلا الزر Button عندما تنشئ زرا فإنك أنشأت كائنا من الخلية CButton وهذا الكائن له:

- خصائص Properties مثل الطول والعرض واللون والتسمية.
- طرق Methods مثل فتح ملف أو تشغيل صوت أو تحريكه Move
- أحداث Events مثل حدث النقر والمرور بالفأرة فوقه.

تنبيه: الطرق والأحداث هي عبارة عن دوال عادية وكذلك الإجراءات إلا أنه في أغلب الأحيان الدوال تعود بقيمة والإجراءات لا تعود بشيء.

إنشاء الخلايا:

يتم إنشاء الخلايا عبر مرحلتين:

1. تصريح الخلية: وذلك بالإعلان عن جميع أعضاء الخلية وبيان أنواعها وقابلية الرؤية وغالبا ما يتم تصريح الخلايا في ملف رأس الذي ينتهي بالامتداد h..

2. تعريف وبناء الخلية: يتم في هذه المرحلة كتابة جسم الخلية والتعليمات والأوامر التي تقوم بها ويكون ذلك في ملف برمجة ينتهي بالامتداد cpp. .

تنبيه:بعد إنشاء الخلية يمكن استعمالها في ملف آخر في نفس المشروع أو في مشاريع أخرى ولكن قبل استعمالها يجب إعلام المترجم بإدراجها وذلك بوضع مرجع لها ويتم ذلك في كلمتين:

```
#include  "اسم الخلية.h"
```

ويوضع هذا السطر في أعلى ملف البرمجة.

وكما يمكن أيضا بناء خلية مشتقة من خلية أو خلايا اخرى.

بناء وهدم الكائن:

ويمكن لكل خلية أن تحتوي على حدثين:

1. حدث البناء : construction ويطلق هذا الحدث عند أخذ جلسة من الخلية ويمكن استعماله لتهيئة متغيرات أو إنشاء اتصالات خارجة كالاتصال مع قاعدة بيانات أو التحقق من كلمة المرور وغير ذلك، ويتم تعرف هذا الحدث في الخلية بنفس اسمها بأقواس فارغة ودون أي قيمة ترجع حي قيمة void.
2. حدث الهدم :destruction ويطلق هذا الحدث عند إنهاء استعمال الكائن وإرادة تحرير مكانه في الذاكرة ويمكن استعماله لغلق الاتصال بقاعدة البيانات أو حفظ بيانات في القرص أو سجل النظام وأغلب ما يستعمل في تحرير الموارد.

مثال إنشاء خلية:

فيما يلي مثال لخلية بسيطة لاحتساب العمر تحتوي هذه الخلية على متغيرين متغير عام الولادة ومتغير للسنة الحالية ودالة لاحتساب العمر.

```
#include <stdio.h>

// تصريح الدالة
class CcalculAge
{
public: //المتغيرات العامة
        void   AnneeNessance(int nAnne);// عام الولادة
        void   Anneeactuel(int nAnnAct);//العام الحالي
        int   CalculAge();//دالة تقوم بعملية الحساب
private: // المتغيرات الخاصة
        int m_nAnneeNessance;
        int m_nAnneeActuel;

};

// الدالة الرئيسية
int main ()
{
        CcalculAge myClass; //اشتقاق كائن من الخلية الجديدة
        myClass.AnneeNessance(1972);// إسناد قيمة لمتغير عام الولادة
        myClass.Anneeactuel(2004);//إسناد قيمة لمتغير العام الحالي
        //عرض العمر على الشاشة
         printf("mon age est :  \t %d ans ",myClass.CalculAge());
        // اضغط مفتاح الإدخال للخروج
        getchar();
        return 0;
}
```

```
مرحلة بناء جسم الخلية//

void CcalculAge::AnneeNessance(int nAnne)
{
   m_nAnneeNessance=nAnne;
}
void CcalculAge::Anneeactuel(int nAnnAct)
{
    m_nAnneeActuel=nAnnAct;
}
int CcalculAge::CalculAge()
{
   return m_nAnneeActuel - m_nAnneeNessance;
}
```

لاحظ أنه عندما تكتب اسم كائن الخلية ثم النقطة فإن **Visual C++ 6** يعرض قائمة بجميع أعضاء الخلية.

في هذا المثال البسيط قمنا بتصريح وبناء الخلية في ملف واحد ويمكنك إنشاء خلية بواسطة مربع الحوار New Class انقر على القائمة Insert ثم اختر الأمر New class تظهر النافذة New Class في الخانة Name اكتب اسم الخلية وتذكر أنه دائما يبتدأ بالحرف C ثم اضغط Ok، سينشئ المعالج ملف الرأس وملف البرمجة الخاصين بالخلية الجديدة وينشئ أيضا حدثي البناء وحدث الهدم هذا هو هيكل الخلية ما عليك أنت إلا ببناء الجسد وتعريف الأعضاء وفقك الله.

الفصل الرابع

مربعات الحوار

محتويات الفصل:

- **تعريف النوافذ.**
- **إدخال البيانات من المستخدم بواسطة مربعات الحوار**
- **إنشاء مربع حوار مع Visual C++ 6**
- **إنشاء مشروع معتمد على مربع حوار**

تعريف مربعات النوافذ.

مربع الحوار عبارة عن نافذة يمكنها ضم عدة أدوات تحكم مثل الأزرار ومربعات النص والصور وغير ذلك للتفاعل من المستخدم إلى البرنامج والعكس، فيمكن لمربعات الحوار أن تعرض على المستخدم مثلا حقوق البرنامج مثل النافذة About أو تطلب منه إدخال بيانات معينة كالتي تطلب منه إدخال كلمة المرور وغير ذلك.

والنوافذ تنقسم إلى نوافذ مشروطة وغير مشروطة.

النوافذ المشروطة هي النوع الأكثر استعمالا ،النوافذ المشروطة تمنع المستخدم من الوصول إلى أي نافذة أخرى من البرنامج ما دامت النافذة المشروطة مفتوحة مثال ذلك نافذة (حول) لبرنامج **Visual C++ 6**

والنوافذ غير المشروطة على العكس مثل نافذة البحث في وورد فيمكنك ونافذة البحث مفتوحة النقر على أمر في قائمة من القوائم أو فتح نافذة أخرى أو كتابة شيء في المستند.

إدخال البيانات من المستخدم بواسطة النوافذ

أكثر ما تستعمل مربعات الحوار للحصول على مدخلات من المستخدم فمربع الحوار قد يحتوي على أدوات قياسية مثل مربع النص الذي يطلب من المستخدم كتابة شيء أو أزرار الخيار وغير ذلك من الأدوات التي ستتعرف عليها ضمن هذا الكتاب.

علاوة على ذلك هناك مربعات حوار قياسية جاهزة تستعمل في جميع إصدارات ويندوز مثل مربع حوار فتح وحفظ واختيار الألوان والخطوط والطباعة وغيرها مما ستعرفه في الفصول اللاحقة، فهذه المربعات تغنيك عن إنشائها يدويا مما يوفر لك كثيرا من الوقت والجهد والسلامة من الأخطاء والتوافقية مع البرامج الأخرى القياسية.

إنشاء مربع حوار مع Visual C++ 6

يسهل **Visual C++ 6** استعمال مربعات الحوار في برامج ويندوز جميع الخطوات اللازمة لإنشاء مربع حوار سهلة كما ستراه.

إضافة مربع رسالة: أبسط مربع حوار في البرمجة هو مربع الرسالة الذي يستعمل لعرض معلومات على المستخدم، هذا النوع من مربعات الحوار سهل جدا فيمكن إظهار مربع رسالة عن طريق سطر واحد من الكود باستعمال مكتبات MFC ، ولإظهار هذا المربع بقيمه الافتراضية استعمل هذا السطر:

```
AfxMessageBox("مرحبا");
```

هذا السطر ينشئ مربع رسالة مع صورة علامة التعجب داخل مثلث أصفر مع زر واحد ، ينبغي تمرير وسيط واحد على الأقل وهو النص الذي ستعرضه الرسالة، ويمكنك تحديد الأيقونة والأزرار التي تريد إظهارها داخل الرسالة. الشكل 4-1 يبن الأيقونات المتاحة التي يمكن عرضها على الرسائل:

شكل 4-1 الأيقونات التي يمكن عرضها داخل مربع رسالة

كل أيقونة من الشكل 4-1 لها دلالة على نوع الرسالة مما يسهل على المستخدم معرفة نوع الرسالة بسهولة نظام ويندوز يستعمل هذه الأيقونات في جميع رسائله وكل أيقونة لها ثابت يخصها يمرر إلى الدالة AfxMessageBox لتعرضها ضمن الرسالة الجدول التالي يبين ذلك:

الأيقونة	الدلالة	اسم الثابت
علامة التعجب	تحذير	MB_ICONEXCLAMATION
حرف i داخل قوس	معلومات	MB_ICONINFORMATION
علامة استفهام	سؤال	MB_ICONQUESTION
علامة الضرب	خطأ	MB_ICONSTOP

من جانب آخر يمكنك تحديد نوع وعدد الأزرار التي تظهر على الرسالة، في الحالة الافتراضية تعرض الرسالة زرا واحدا يحمل العنوان Ok، ولكن عندما تطرح مثلا سؤالا على المستخدم أنت بحاجة إلى إظهار زري نعم ولا للحصول على جواب منه في هذه الحالة ينبغي عرض مربع رسالة يحمل أيقونة الاستفهام وزري نعم ولا الكود التالي يبين كيفية عرض سؤال على المستخدم عن استبدال ملف بآخر.

```
int nRepons = AfxMessageBox("هل تريد استبدال الملف ؟",
                            MB_YESNO |                  // تحديد الأزرار
                            MB_ICONQUESTION     // تحديد الأيقونة
                            );
if ( nRepons == IDYES )       // معرفة إذا كان المستخدم نقر على الزر نعم
{
        //استبدل الملف
}
```

العلامة العمودية | تستعمل للتفريق بين الثوابت المختلفة المسندة للدالة .
عند تنفيذ البرنامج ستعرض رسالة استفهام على المستخدم تحمل أيقونة استفهام وزرين نعم ولا .
الجدول التالي يبين ثوابت الأزرار التي يمكن استعمالها.

اسم الثابت	الأزرار المعروضة
MB_ABORTRETRYIGNORE	إحباط ، إعادة المحاولة ، تجاهل
MB_OK	موافق
MB_RETRYCANCEL	إعادة المحاولة ، إلغاء
MB_YESNO	نعم ، لا
MB_YESNOCANCEL	نعم ، لا ، إلغاء

ويمكن للدالة AfxMessageBox الرجوع بقيمة من نوع int تبين الزر الذي اختار المستخدم ، الجدول التالي يبين القيم التي تعود بها الدالة AfxMessageBox.

القيمة	الزر المختار
IDABORT	إحباط
IDCANCEL	إلغاء
IDIGNORE	تجاهل
IDNO	لا
IFOK	موافق
IDRETRY	إعادة المحاولة
IDYES	نعم

تنبيه: لا يمكن استعمال الرمز | لدمج أيقونتين أو نوعين مختلفين من ثوابت الأزرار، وإن لم يتم تمرير أي وسيط للدالة فإنها ستعرض افتراضيا أيقونة (معلومات) وزر (موافق).
إضافة مربع حوار: على العموم تمر عملية لإضافة مربع حوار إلى البرنامج بأربع خطوات:

1. إضافة ملف موارد Resource من نوع مربع حوار Dialog بواسطة محرر موارد **Visual C++ 6**.
2. استعمال المعالج Class Wizard لإنشاء خلية مشتقة من CDialog لإدارة مربع الحوار ووضع الكود اللازم له.
3. إضافة الكود والأوامر التي تريدها في الخلية ليقوم مربع الحوار بعمل مفيد.
4. كتابة الكود لعرض هذا المربع على المستخدم كالنقر مثلا على زر أو تحديد أمر من القوائم أو شريط الأدوات.

الموارد: الموارد أنواع من الملفات تحتاجها برامج ويندوز مثل الصور والأيقونات والمؤشرات والقوائم. مربعات الحوار هي أيضا نوع من هذه الموارد تحفظ في ملف الموارد على القرص ولا يحملها البرنامج عند تشغيله إلا عندما يحتاج إليها سترى في الفصول اللاحقة إن شاء الله الكلام على هذه الموارد.

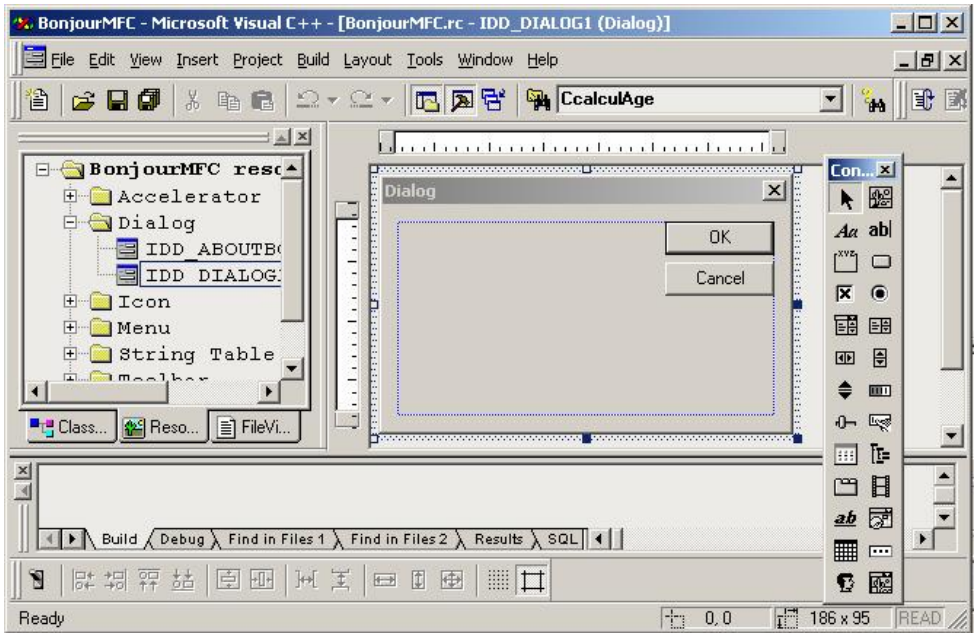
إنشاء مورد مربع حوار: يتيح **Visual C++ 6** إنشاء مربع حوار وتحريره يدويا بواسطة الفأرة ولوحة المفاتيح، كما يمكنك وضع بعض أدوات التحكم عليه مثل الأزرار ومربعات النص كما يمكنك تحجيمه وتغيير خصائصه وخصائص أدوات التحكم الموجودة عليه عن طريق نافذة الخصائص .

وقبل تنفيذ الخطوات التالية أنشئ مشروعا جديدا من نوع MFC AppWizard(exe) وسمه BonjourMFC حدد الخيار Single document أي من نوع SDi وانقر زر الإنهاء لإنشاء المشروع .

أضف مربع حوار جديد للمشروع ولفعل ذلك اتبع إحدى الطريقتين:

- افتح النافذة Insert Resource وذلك بالنقر على الأمر Resource من القائمة Insert ثم اختر Dialog من النافذة ثم انقر الزر New .
- انقر بالزر الأيمن على المجلد Dialog في الشريحة ResourceView من الإطار WorkSpace ثم اختر الأمر Insert من القائمة المنسدلة.

بعد تنفيذ إحدى الطريقتين سينشئ المحرر مربع حوار جديد مع زرين (موافق وإلغاء) يمكنك تعديل هذا المربع يدويا ووضع أدوات التحكم عليه كما سترى لاحقا، الشكل 4-2 يبين كيف يجب أن تبدو نافذتك:



خصائص مربعات الحوار: لمربعات الحوار خصائص ذات قيم متعددة يمكن تعديلها يدويا عن طريق نافذة الخصائص، لإظهار هذه النافذة انقر بالزر الأيمن على مربع الحوار ثم اختر خصائص من القائمة المنسدلة ، في الجدول التالي عرض لبعض خصائص مربعات الحوار الهامة:

الخاصية	الوصف
ID	الاسم البرمجي للمربع وتبتدأ أسماء مربعات الحوار بالكلمة IDD_ ثم كلمة معبرة في برنامجنا يمكن تسمية المربع ب IDD_BONJOR
Caption	عنوان النافذة الذي يظهر على الشريط الأزرق الأفقي في أعلى النافذة غيره إلى "ترحيب"
Menu	ربط قائمة معينة بالمربع (غالبا ما تكون هذا الخاصية معطلة في مربعات الحوار)

Font	تحديد اسم الخط المستعمل في المربع وحجمه، ولكن ينصح بترك هذه الخاصية بدون تعديل للسماح للمستخدم باختيار نوع الخط الذي يناسبه بواسطة نظام التشغيل فما كل المستخدمين لبرنامجك يفضلون نوع الخط الذي تحدده خاصة للذين يعانون من مشاكل بصرية.
Style	أكثر مربعات الحوار تستعمل القيمة Popup لهذه الخاصية.
Border	عموما Dialog Frame
Minimize Box	إضافة زر تصغير (معطل غالبا)
Maximize Box	إضافة زر تكبير (معطل غالبا)
Title Bar	إضافة شريط العنوان (محدد غالبا)
System Menu	إضافة قائمة النظام التي تظهر عند النقر على أيقونة النافذة في شريط العنوان.
Horizontal Scroll	إضافة شريط إزاحة أفقي (معطل غالبا)
Vertical Scroll	إضافة شريط إزاحة عمودي (معطل غالبا)
System Modal	إنشاء مربع حوار مشروط
Absolute Align	تحديد كيف يتم محاذاة المربع هل يحاذى نسبة للشاشة أم نسبة للنافذة الأم تحديد هذه الخاصية يضبط الخيار الأول.
Visible	تحديد ظهور المربع على الشاشة وعدمه
Disabled	تفعيل المربع بحيث يستجيب لعمل المستخدم أو لا
3D-Look	إحاطة المربع بحدود ثلاثية الأبعاد (معطل غالبا)
Set Foreground	إبقاء المربع دائما في المقدمة ولو كان فاقدا للتركيز (معطل غالبا)
No Fail Create	إعلام ويندوز بإنشاء المربع ولو كانت هناك أخطاء (معطل غالبا)
Control	جعل المربع كأداة تحكم يمكن أن تحتضنها نافذة أخرى (معطل غالبا)
Center	توسيط المربع على الشاشة عند ظهوره (معطل غالبا)
Center Mouse	توسيط مؤشر الفأرة على المربع عند ظهوره (معطل غالبا)
Context Help	إضافة زر التعليمات - الذي هو عبارة عن أداة استفهام – في شريط العنوان
Right-To-left Reading Order	محاذاة عنوان المربع على اليمين (خاصة بالمبرمج العربي)
Right To Left Layout	جعل مظهر المربع عربيا

ملاحظة: عند كتابة كلمة عربية في عنوان النافذة مثلا أو على زر فإنها ستظهر عند التنفيذ برموز غريبة لحل هذه المشكلة انقر بالزر الأيمن على اسم المربع في الشريحة ResourceView من الإطار WorkSpace ثم اختر الأمر خصائص غير الخاصية Language إلى Neutral.

إضافة أداة تحكم إلى المربع: أبسط أداة يمكن إضافتها إلى المربع هي أداة التسمية Static Text التي تستعمل لإظهار نصوص ولإضافة هذه الأداة اتبع الخطوات التالية:

1. حدد أيقونة Static Text في صندوق الأدوات Controls بعد تحديد هذه الأيقونة سيأخذ المؤشر شكل علامة الجمع + ، لمعرفة نوع الأيقونة ضع عليها الفأرة لفترة قصير سيظهر تلميح أصفر يبين اسم الأداة.
2. انقر بالزر الأيسر للفأرة وسط مربع الحوار سترى أداة التسمية وضعت على المربع مكتوب عليها Static.
3. انقر على هذه الأداة بالزر الأيسر واختر خصائص لاستعراض أهم الخصائص الخاصة بالأداة .

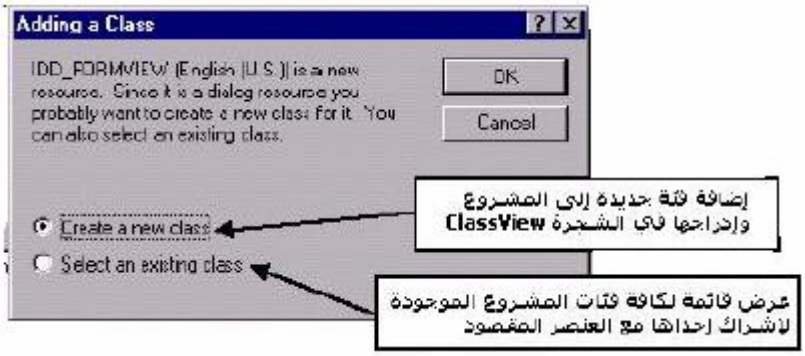
تلميح: تفيد التسميات لإظهار معلومات للمستخدم كأن توضع مثلا أمام مربع نص لحث المستخدم على كتابة اسمه.

إنشاء خلية للمربع الجديد: الخلية CDialog تتيح لك إدارة أكثر مهام مربعات الحوار برمجيا لذلك يجب اشتقاق خلية من هذا النوع للمربع الجديد.

ولفعل ذلك سنستعمل المعالج Class Wizard افتح هذا المعالج عن طريق:

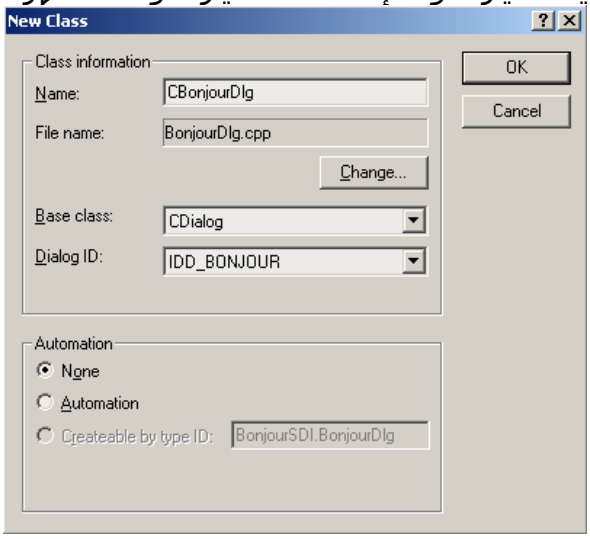
- الضغط على Ctrl – W
- اختيار أمر Class Wizard من القائمة View.
- النقر بالزر الأيمن في مكان خال من النموذج واختيار class Wizard من القائمة المنسدلة.

بعد تشغيل المعالج سيتعرف تلقائيا على المورد Resource الجديد لذلك يعرض هذه النافذة يقترح فيها هذين الخيارين انظر الشكل 4-3:



الخيار الأول يعني إنشاء خلية جديدة والثاني تحديد خلية موجودة.

غالبا ما ستحتاج إلى تحديد الخيار الأول، إذن حدد الخيار الأول ستظهر لك هذه النافذة:



شكل 4-4

سينشئ المعالج الخلية الجديدة وفقا للقيم التي تحددها في هذه النافذة ، أدخل القيم كما هي مبينة في هذا الجدول:

الأداة	القيمة
Name	CBonjourDlg
File Name	BonjourDlg.cpp
Base Class	CDialog
Diqlog ID	IDD_BOJOUR
Automation	Non

انقر Ok ستم إنشاء الخلية وسيضيف المعالج ملفين إلى المشروع:

- ملف الرأس BonjourDlg.h والذي يحتوي على تصريحات الخلية.
- ملف البرمجة BonjourDlg.cpp والذي يحتوي على جسم الخلية.

إضافة دالة تهيئة مربع الحوار: هناك دوال كثيرة يمكن توظيفها لاصطياد الرسائل المرسلة من قِبل نظام التشغيل ويندوز، ونعني بالرسائل الإعلام بوقوع حدث معين على كائن معين، فأنت عندما تنقر على زر مثلا فإن نظام التشغيل سيتعرف على هذا النقر ويرسل رسالة إلى برنامجك لإخباره بوقوع حدث النقر على الزر، والبرنامج ليس بحاجة إلى التعامل مع جميع رسائل ويندوز لذلك فالخيار يبقى لك في تحديد الرسائل التي تريد التعامل معها وتجاهل الباقي.
ومن بين الرسائل الهامة التي يرسلها نظام التشغيل الرسالة WM_INITDIALOG، يتم إرسال هذه الرسالة عند إنشاء مربع الحوار في الذاكرة وقبل عرضه على الشاشة فهذه الرسالة تصلح لتهيئة مربع الحوار وأدواته وإعطاء قيم لها قبل عرضها على الشاشة.
سننشئ دالة بواسطة Class Wizard لتلقي هذه الرسالة ولفعل ذلك اتبع الخطوات التالية:

- شغل المعالج Class Wizard بالضغط على Ctrl-W
- حدد الشريحة Message Maps ثم في القائمة المنسدلة Class Name اختر الخلية الخاصة بمربع الحوار وهي CBonjourDlg
- في القائمة Object ID اختر الكائن الذي سيتلقى الرسالة هنا CBonjourDlg بعد اختيارك للكائن ستمتلئ القائمة Messages بجميع الرسائل التي يدعمها الكائن.
- سنبحث عن رسالة التهيئة الخاصة بمربع الحوار ممر القائمة للأسفل وحدد الرسالة WM_INITDIALOG ثم انقر على الزر Add Function سيضيف المعالج تلقائيا الدالة OnInitDialog التي ستتولى اصطياد هذه الرسالة.
- انقر Ok لغلق المعالج ، يمكنك الانتقال مباشرة إلى الدالة بالنقر على الزر Edit Code .

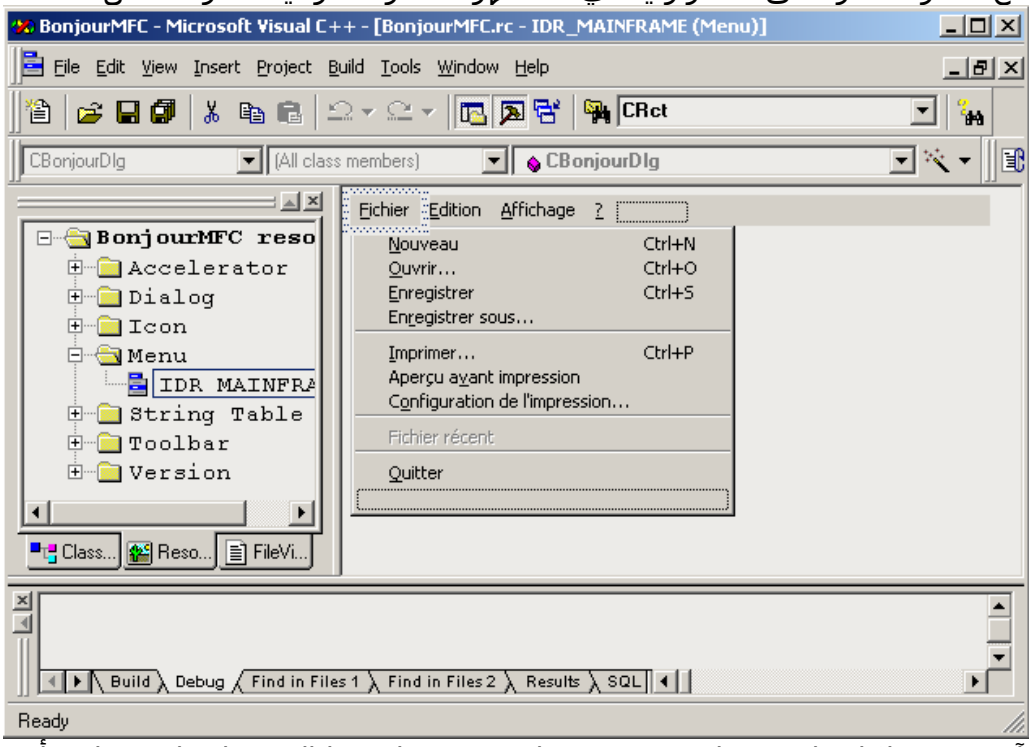
سنستعمل هذه الدالة لعرض رسالة على المستخدم تبين له أنه تم تلقي الحدث WM_INITDIALOG.
عدل الدالة OnInitDialog لتصبح كما في القائمة 4-1.

ملاحظة: للوصول إلى الدالة OnInitDialog في محرر الكود انقر على علامة الجمع + أمام اسم الخلية في الشريحة Class View من الإطار WorkSpace ابحث في الفروع عن اسم الدالة وانقر عليها نقرا مزدوجا.

```
BOOL CBonjourDlg::OnInitDialog()
{
CDialog::OnInitDialog();
AfxMessageBox("WM_INITDIALOG تم تلقي الرسالة  ");
return TRUE;  // return TRUE unless you set the focus to a control
}
```

إضافة عنصر إلى شريط القوائم: القوائم ستكون موضع اهتمامنا أكثر في الفصل العاشر من هذا الكتاب فيما يلي سنضيف عنصر قائمة لإظهار مربع الحوار الذي أنشأته سابقا بالنقر عليه.
القوائم محفوظة في المشروع ضمن ملف الموارد Resource، ولاستعراض هذه القوائم حدد الشريحة ResourceView من الإطار WorkSpace انقر علامة الجمع من العنصر BonjourMFC

Resources ستظهر عناصر فرعية بجميع أنواع الموارد المتوفرة في المشروع واحد من هذه العناصر اسمه Menu.
افتح المجلد Menu لعرض ملف القوائم IDR_MAINFRAM انقر عليه نقرا مزدوجا لفتحه بعد برهة سيفتح محرر الموارد ملف القوائم جاهزا لتحريره يمكنك استعراض عناصر القائمة تماما كما تفعل في البرامج الأخرى انقر على عنصر رئيسي ستظهر عناصره الفرعية انظر الشكل 4-5.



لاحظ أن آخر عنصر لكل قائمة عبارة عن مربع فارغ يستعمل هذا المربع لإضافة عناصر أخرى في البداية يمكنك كتابة اسم العنصر الجديد ثم وضعه في المكان المناسب له، إضافة عنصر قائمة جديد افعل ما يلي:

1. انقر نقرا مزدوجا على المربع الفارغ في القائمة ملف Fichier لإظهار نافذة الخصائص Menu Item Properties .
2. لإضافة عنصر قائمة اكتب اسمه ID وتسميته Caption في هذه النافذة، في مثالنا هذا أدخل ID_FILE_BONJOUR في الحقل ID و &Bonjour في الحقل Caption.
3. انقر خارج مربع الخصائص للعودة إلى المحرر.

بعد إضافتك للعنصر الجديد الخطوات التالية تبين كيفية إضافة دالة مرتبطة بحدث النقر على هذا العنصر سنضع في هذه الدالة الكود اللازم لعرض مربع الحوار على الشاشة.

1. افتح المعالج Class Wizard اضغط على Ctrl-W أو انقر بالزر الأيمن داخل نافذة البرمجة ثم اختر Class Wizard من القائمة الفرعية.
2. حدد الشريحة Message Maps وفي القائمة المنسدلة Class Name حدد الخلية التي ستعالج الرسالة وهي CMainFrame.
3. في القائمة Object ID حدد الكائن الذي ستربط به الرسالة هذا الكائن هو اسم عنصر القائمة الذي أنشأته للتو وهو ID_FILE_BONJOUR ستظهر لك في القائمة Messages رسالتان خاصتان هذا العنصر.
4. حدد الرسالة COMMAND في القائمة Messages ثم انقر على الزر Add Function أقبل اسم الدالة المقترح من طرف المعالج: OnFileBonjour.
5. انقر Ok لغلق المعالج ، ويمكنك الانتقال مباشرة إلى الدالة بالنقر على الزر Edit Code .

عدل الدالة CMainFrame::OnFileBonjour كما هو موضح في القائمة 4-2.

```
void CMainFrame::OnFileBonjour()
{
// TODO: Add your command handler code here
```

```
    CBonjourDlg     dlgBonjour;
    if( dlgBonjour.DoModal() == IDOK )
        AfxMessageBox("لقد نقرت على موافق");
     else //IDCANCEL
        AfxMessageBox("لقد نقر على إلغاء");
}
```

بعد ذلك أضف إلى الملف MainFrm.cpp سطر التوجيه الذي سيضيف تعريف الخلية CBonjourDlg الذي يوجد في الملف BonjourDlg.h أضف السطر التالي أسفل السطر #include "MainFrm.h"

```
#include "BonjourDlg.h";
```

فسر المشروع بالضغط على F7 ثم شغله بالضغط على F5 .

عند استدعاء الدالة DoModal سيعرض مربع الحوار IDD_BONJOUR لن تكمل الدالة مهمتها حتى تنقر على أحدي زري مربع الحوار موافق أو إلغاء إذا نقرت على موافق القيمة المعادة هي IDOK وإذا نقرت على إلغاء ستعاد القيمة IDCANCEl وستكمل الدالة مهمتها وتعرض مربع رسالة تبين فيها أي الزرين تم نقره اعتمادا على القيمة المعادة.

إنشاء مشروع معتمد على مربع حوار:

هذا هو النوع الثالث من أنواع المشاريع التي يوفرها المعالج AppWizard وهي المشاريع المعتمدة على مربعات الحوار وهناك برامج كثيرة تستعمل هذا النوع مثل البرامج المساعدة كالتي ندها في لوحة التحكم وغيرها.

البرامج من هذا النوع سهلة وسريعة الاستعمال لأنها لا تحتاج لكثير من الخلايا والموارد.

استعمال AppWizard لإنشاء مشروع معتمد على مربع حوار: يمكنك إنشاء مشروع معتمد على مربع حوار بواسطة المعالج **AppWizard** بنفس الطريقة التي تنشئ بها مشاريع SDI و MDI إلا أنه لكون المشاريع من هذا النوع سهلة وقليلة الموارد فإن المعالج يظهر لك أربع خطوات بدل من ستة .

لإنشاء مشروع معتمد على مربع حوار بواسطة المعالج **AppWizard** افعل كما يلي:

1. شغل المعالج MFC AppWizard باختيار الأمر File|New حدد الشريحة Projects أدخل "DialogBox" كاسم للمشروع وحدد العنصر MFC AppWizard(exe) ثم انقر Ok
2. عندما يعرض المعالج الخطوة الأولى حدد الخيار Dialog Based هذا هو الخيار الذي يحدد مشاريع من هذا النوع.
3. اقبل جميع الإعدادات الافتراضية في الخطوات اللاحقة ثم انقر Finish سينشئ المعالج هيكل المشروع بالمواصفات التي حددناها سابقا.

استكشاف المشروع DialogBox: بعد إنشاء المشروع يمكنك تفحص محتوياته عبر الإطار WorkSpace يعرض هذا الإطار ثلاثة شرائح هامة :

1. الشريحة ClassView : تعرض جميع الخلايا والمتغيرات الموجودة في مشروعك كما يمكنك عبر القائمة الموضعية التي تظهر عند النقر بالزر الأيمن على عنصر من عناصر هذه الشريحة :إنشاء خلية جديدة، تصريح متغير جديد، الذهاب إلى مكان تصريح أو تعريف العنصر المحدد.
2. الشريحة ResourceView: تعرض جميع ملفات الموارد الموجودة في مشروعك مثل الصور والقوائم والنماذج وأشرطة أدوات كما يمكنك من خلالها إضافة مورد جديد من أي نوع من الأنواع المتاحة والموارد تحفظ في ملف على القرص بالامتداد rc.
3. الشريحة FileView: تعرض جميع الملفات الموجودة على القرص التي يستخدمها مشروعك لأن الخلايا والموارد عبارة عن ملفات فكل خلية تعتمد على ملفين ملف رأس توضع فيه تصريحات الخلية والذي ينتهي بالامتداد h. وملف برمجة يوضع فيه جسم الخلية والذي ينتهي بالامتداد cpp. والموارد أيضا عبارة عن ملف ينتهي بالامتداد rc. إلى غير ذلك من الملفات التي تضيفها لمشروعك مثل صفحات الإنترنت والمستندات وغير ذلك.

وهناك اختلاف في عناصر هذه الشرائح بين المشاريع المعتمدة على مربعات الحوار والمشاريع ذات المستندات فنلاحظ في مشروعنا DialogBox :

- عدم وجود القوائم وأشرطة الأدوات إلى جانب وجود مربع الحوار الذي يعتمد عليه المشروع.

- عدم وجود خلايا من نوع Doc أو View.

محرر النماذج ومربعات الحوار:افتح محرر النماذج بالنقر المزدوج على العنصر IDD_DIALOGBOX_DIALOG في شريحة الموارد من الإطار WorkSpace وبالظبط تحت المجلد Dialog بعد فتح المحرر يمكنك تحرير هذا المربع ووضع أدوات التحكم عليه من خلال الشريط Dialog يوفر لك هذا الشريط مجموعة متميزة من أدوات التحكم مثل الأزرار ومربعات النص والشجرة وأداة العرض ومربع النص الغني وغيرها من الأدوات بعد أن تنتهي من وضع الأدوات يمكنك معاينة النموذج كيف يكون في وضع التشغيل دون الحاجة إلى تفسير المشروع وذلك من خلال الأمر Layout!Test أو المفتاح Ctrl-T بعدما تنتهي من التصميم المرئي يمكنك الانتقال إلى البرمجة وإضفاء الحيوية لهذا المربع ستجد الخلية الخاصة بهذا المربع في الشريحة ClassView باسم CDialogBoxDlg .

الفصل الخامس

الأزرار

محتويات الفصل:

- **تعريف الزر.**
- **إنشاء متغيرات أعضاء للتحكم في الأزرار**
- **التعبيرات الشرطية**
- **الأزرار المتاحة والمعطلة**
- **إخفاء الأزرار**
- **تحديد ترتيب الجدولة للأدوات**

تعريف الزر.

الأزرار نوع خاص من نوافذ ويندوز على شكل إطار تحمل نصا أو صورة ويمكن النقر عليه للقيام بعمل معين نجد الأزرار كثيرا في النماذج ومربعات الحوار وأشرطة الأدوات والنوافذ الأخرى الحاضنة لأدوات التحكم مثل متصفح الويب، يضع ويندوز بين أيدي المستعملين ثلاثة أنواع مختلفة من الأزرار:

- الأزرار القياسية وهي الأزرار المستعملة في معظم التطبيقات لها شكل ثلاثي الأبعاد وعندما يُنقر عليها تنضغط إلى أسفل وترتفع عندما يُحرر زر الفأرة غالبا ما تحمل هذه الأزرار نصا يدل على وظيفتها كطباعة أو حفظ.
- أزرار أو مربعات العلامة وهي عبارة عن مربعات تحمل نصا أمامها يمكن تحديدها أو إلغاء تحديدها ، مربع العلامة يعرض خيارا على المستخدم قد يظهر هذا الخيار منفردا أو ضمن مجموعة يختار المستخدم منها ما يشاء وعندما ينقر المستخدم مربع العلامة ستظهر علامة صح داخله مما يعني أن المستخدم قد اختاره وإذا كان المربع فيه علامة ثم نقره المستخدم فإن العلامة ستختفي مما يعني أن المستخدم قد ألغى اختياره وهناك قيمة أخرى لمربعات العلامة بين صح وخطأ ويظهر فيها مربع الخيار باللون البني.
- أزرار الخيار وهي عبارة عن دوائر بيضاء تحمل نصا أمامها وهي تتيح للمستخدم اختيارا واحدا من عدة اختيارات وعلى عكس مربعات العلامة فإن مربعات الخيار لن تتيح للمستخدم اختيار أكثر من خيار واحد فقط في نفس الوقت فإذا اختار المستخدم خيارا ثانيا فإن الخيار الأول سيلغى، هذه الأزرار تسمى أيضا أزرار الراديو لأنها تشبه أزرار الراديو الموجودة في السيارات القديمة.
- أزرار مخصصة التي ينشئها المستخدم وليست في ويندوز.

يستعمل الإطار لتجميع أزرار تشترك في وظيفة واحدة ونجد معظم البرامج تستعمل الإطارات لتجميع الأزرار وتصنيفها وتحسين المظهر الجمالي للنموذج.

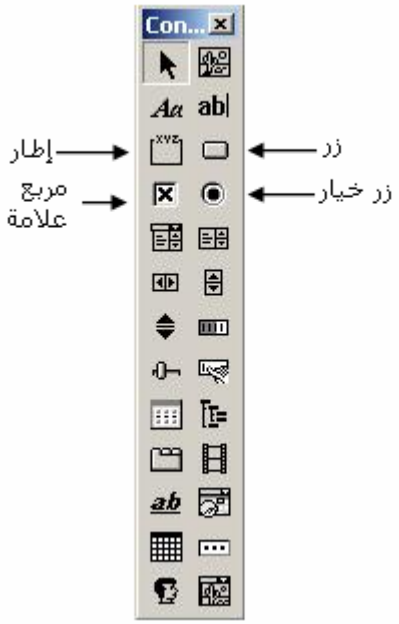
إدارة الأزرار بواسطة مكتبات MFC : بعد إضافة زر إلى نموذج يمكنك تشغيل المعالج AppWizard لإضافة الدالات التي تعالج الأحداث التي تطلق عندما ينقر المستخدم على الزر أو يحدد مربع خيار أو مربع علامة كما يمكنك أيضا استعمال هذا المعالج لإنشاء كائنات من الخلية CButton وربطها بالأزرار الموجودة على النموذج هذه الكائنات تتولى عملية إدارة الأزرار سيتضح لك هذا الكلام في أمثلة هذا الفصل.

يمكنك استعمال الخلية MFC CButton للتفاعل مع الأزرار المضافة إلى مربع الحوار في ملف الموارد أو مع الأزرار المنشأة ديناميكيا بواسطة الكود استعمل المعالج Class Wizard لربط زر مع كائن CButton .

مثال لمشروع يستعمل الأزرار: إذا أردت معرفة كيفية استعمال الأزرار مع مربعات الحوار أنشئ بواسطة المعالج AppWizard مشروعا معتمدا على مربع حوار وسمه Buttons لفعل ذلك اتبع الخطوات التي مرت بك في الفصل السابق سنستعمل هذا المشروع طوال هذا الفصل لنرى كيفية التعامل مع الأزرار.

بمجرد ما ينشئ المعالج المشروع ستظهر النافذة الرئيسية أمام عينيك جاهزة للتحرير إذا لم تظهر هذه النافذة يمكنك فتحها من الشريحة ResourceView بالنقر المزدوج على العنصر IDD_DIALOG في المجلد Dialog.

في نفس الوقت الذي يعرض فيه المحرر النموذج يعرض أيضا لوحة عائما تسمى Controls تحتوي هذه اللوحة على معظم أدوات التحكم التي يمكن إدراجها في النموذج انظر الشكل 5-1:



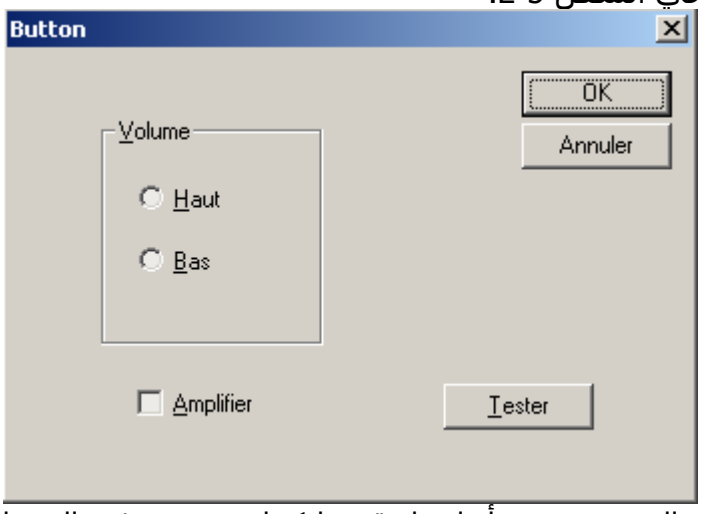
إن لم تكن هذا اللوحة معروضة يمكنك فعل بذلك عن طريق النقر بالزر الأيمن على شريط الأدوات ثم وضع علامة بجانب Controls.

ولإضافة زر إلى النموذج استعمل إحدى الطريقتين التاليتين:

- اسحب عنصر الزر من اللوحة بواسطة زر الفأرة الأيمن ثم أفلته فوق النموذج سيرتسم الزر على النموذج
- انقر على رمز الزر في اللوحة ثم أعد النقر فوق المكان الذي تريده فوق النموذج.

بعد وضعك الزر على النموذج يمكنك تغيير مكانه وأبعاده بالطرق الاعتيادية بواسطة زر الفأرة.

في مشروعنا هذا أضف إلى النموذج الأنواع المختلفة من الأزرار التي تم ذكرها سابقا عدل أبعادها كما هو مبين في الشكل 5-2.



أضف خمسة أزرار إلى النموذج وحدد أنواعها وقيمها كما هو مبين في الجدول التالي:

الاسم ID	نوع الزر	التسمية Caption
IDC_BTN_TESTER	زر عادي	&Tester
IDC_RADIO_HAUT	زر خيار	&Haut
IDC_RADIO_BAS	زر خيار	&Bas
IDC_GROUP_VOLUME	إطار	&Volume
IDC_CHECK_AMP	مربع علامة	&Amplifier

استعمال أسماء الأزرار القياسية:
هناك أنواع من الأزرار لها أسماء يكاد يستعملها كل المبرمجين في برامجهم ينصح بشدة باتباعها حتى لا تربك المستخدمين وجعلهم يتأقلمون بسهولة مع برامجهم وهي:

- موافق: يعني قبول جميع القيم الموجودة فلا يستحسن مثلا أن تبدلها بكلمة أخرى مثل مقبول .
- إلغاء: عدم قبول القيم
- إغلاق: غلق نافذة مع تنفيذ التغيرات التي أدخلها المستخدم.
- تعليمات: طلب المساعدة حول موضوع ما.
- تطبيق: تفعيل التغيرات وتطبيقها على البنامج.

وغالبا ما نجد هذه الأزرار في نوافذ الخيارات والمعالجات.

إنشاء متغيرات أعضاء للتحكم في الأزرار:

أسهل وسيلة للتعامل مع الأدوات وضبط وقراءة قيمها هي إشراك متغير عضو بالأداة عن طريق المعالج Class Wizard ، وعند إشراك متغير عضو بأداة معينة يمكنك تحديد نوع الربط ربط مع الأداة نفسها أو مع قيمتها Value ، والمتغيرات المربوطة مع الأزرار غالبا ما تربط مع قيمها إنما تربط بالزر نفسه أي مع الخلية CButton سترى فيما يلي كيفية ربط متغيرات أعضاء بأدوات مربع حوار.
لإضافة متغيرات أعضاء لخلية مشتقة من CDialog اتبع هذه الخطوات:

1. افتح Class Wizard
2. حدد الشريحة Member Variables
3. حدد الخلية المشتقة من CDialog التي تدير مربع الحوار هنا CButtonDlg
4. في القائمة Control ID's حدد ID_BTN_TESTER
5. انقر الزر Add Variable تظهر النافذة Add Member Variable أدخل البيانات كما هي مبينة في الجدول التالي ثم انقر OK
6. أغلق المعالج
7. كرر هذه العملية مع الأدوات الأخرى المبية في الجول.

Varible Type	Category	Member VariableName	اسم الأداة
CButton	Control	m_btnTester	ID_BTN_TESTER
CButton	Control	m_btnVolume	ID_GROUP_VOLUME
CButton	Control	m_btnAmp	ID_CHECK_AMP

سيضيف المعالج تلقائيا تصريحات هذه المتغيرات في الخلية CButtonDlg
لاحظ أن المعالج يقترح عليك البادئة m_ عند تصريح متغير عضو هذا يدل على أن المتغير عضو Member لذلك يجب عليك التقيد بها وعدم حذفها.

إضافة أحداث الأزرار إلى خلية مربع الحوار:
الأزرار التي وضعناها على النموذج تظهر عند تشغيل البرنامج ولكنها لا تتفاعل مع المستخدم إذا نقر عليها لأننا لم نضع أوامر في حدث النقر في الخطوات التالية نضع بعض الأوامر في حدث النقر على الزر ID_BTN_TESTER .

1. افتح Class Wizard
2. حدد الشريحة Message Maps
3. حدد CButtonDlg في Class name
4. حدد ID_BTN_TESTER في القائمة Object IDs
5. حدد BN_CLICKED في قائمة الرسائل
6. انقر الزر Add Function اقبل بالاسم الافتراضي للدالة التي تعالج هذا الحدث
7. انقر الزر Edit Code لغلق المعالج والانتقال مباشرة إلى ملف البرمجة لتحرير الكود

الرسالة BN_CLICKED تستعمل غالبا مع الأزرار القياسية أما مربعات العلامة والخيار فغالبا ما تستعملها، أضف السطر الموجود بين علامتي الاحتضان {} كما هو مبين في القائمة 5-1

```
void CButtonDlg::OnBtnTester()
```

```
{
        AfxMessageBox(" لقد نقرت الزر Tester");
}
```

تغيير نص زر:

كما هو الحال بالنسبة لبقية الأدوات الأزرار عبارة عن نوافذ خاصة وتستعمل الخلية CWnd كخلية قاعدية لجميع الأدوات، لتغيير عنوان زر يمكنك استعمال الدالة SetWindowText.

سنستعمل هذه الدالة لتغيير عنوان الزر ID_CHECK_AMP من Amplifier إلى Enregistrer لفعل ذلك غير الدالة السابقة CButtonDlg::OnBtnTester كما هو مبين في القائمة 2-5.

```
void CButtonDlg::OnBtnTester()
{
  static BOOL bNiveau = TRUE;
 if( bNiveau == TRUE )
        {
                m_btnVolume.SetWindowText( "&Niveau" );
                m_btnAmp.SetWindowText( "&Enregistrer" );
                bNiveau = FALSE;
        }
        else
        {
                m_btnVolume.SetWindowText( "&Volume" );
                m_btnAmp.SetWindowText( "&Amplifier" );
                bNiveau = TRUE;
        }
}
```

عند تشغيل البرنامج انقر الزر Tester سيستبدل النص "Volume" بالنص "Niveau" والنص "Amplifier" بالنص "Enregistrer" .

التعبيرات الشرطية:

القائمة السابقة 2-5 تستخدم عبارة شرطية لتحديد النص المطلوب وضعه على الأداتين من المناسب أن نسلط بعض الضوء على التعبيرات الشرطية.

التعبير الشرطي صيغة برمجية تعود بقيمة صح أو خطأ لا غير وأغلب البرامج لا تستغني عن استعمال هذا النوع من التعابير مما يسمح لها بتنفيذ تعليمات برمجية وفق معايير محددة من طرف المستخدم أو البرنامج نفسه.

التعليمة If:

التعليمة If تسمح بتنفيذ تعليمة أو أكثر وفق شرط معين موضوع بعدها بين قوسين إذا كان الشرط صحيحا فسيتم تنفيذ التعليمات المشروطة وإلا فسيقفز البرنامج للتعليمات الأخرى ولا ينفذ هذه الأوامر طالما كان الشرط خاطئا.

القائمة 3-5 مثال عن دالة تستعمل الشرط If إذا كان الوسيط الممرر إلى الدالة أكبر من فإن الدالة ستعود بقيمة صحيحة True وإلا فستبقى القيمة المعادة خاطئة False .

```
bool EstPositif( int nTestValeur )
{
 bool bValeurRenvoyee = false;
 if( nTestValeur > 0 )
        bValeurRenvoyee = true;
 return bValeurRenvoyee;
}
```

الكتل:

إذا كان هناك أكثر من تعليمة واحدة مشروطة ب If يجب إحاطة هذه التعليمات بقوسي الاحتضان {} الأوامر المحاطة بقوسي الاحتضان تدعى كتلة برمجية إما إذا كانت هناك تعليمة واحدة مشروطة ب If فليس من الضروري إحاطتها بقوسي الاحتضان مثل

```
if (nValue == 1)
nValue=+1
```

ولا توجد الكتل في التعليمات الشرطية فحسب بل نجدها في كثير من التركيبات مثل الخلايا والدوال والتراكيب والعدادات فكلها تحاط بقوسي الاحتضان

تنبيه : عندما تستعمل عامل المقارنة = في صيغة شرطية يجب أن تكتبه مرتين كما في المثال السابق لأن هذا العامل يستعل في حالة الإفراد لإسناد القيم فقط .

ويستحسن تنظيم الكتل باستخدام مفتاح الجدولة Tab لجعل الكود أكثر وضوحا للعلم فإن محرر كود vusial C++ يتولى هذه المهمة في اغلب الحيان لاحظ انك عندما تفتح قوس الاحتضان { ثم تضغط مفتاح الإدخال ينتقل المؤشر مسافة معينة نحو الأمام.

استعمال الكلمة else مع التعليمة if:

تستعمل الكلمة else بعد التعليمة if لتنفيذ أوامر معينة في حالة إذا ما كان الشرط خاطئا، عندما تختبر If الشرط فإذا

وجدته صحيحا تنفذ التعليمات التي بعدها مباشرة وإن وجدته خاطئا قفزت لتنفيذ التعليمات الموجودة بعد الكلمة else.

التعليمة switch:

ماذا لو كانت الشروط المختبرة كثيرة نوعا ما ،في الحالة ليس المناسب استعمال وتكرار الكلمة If لأنه يؤدي إلى تعقيد البرنامج واحتمال وقوع اخطاء، يوفر **Visual C++ 6** تعليمة أخرى بدل if تستعمل لمقارنة عدة قيم وهي switch تستعمل هذه التعليمة كما في القائمة 5-4

```
bool SelectionMenu(char chSelection)
{
        bool bInstructionValable = true;
        switch( chSelection )
        {
                case 'N':
                        NouveauFichier();
                        break;
                case 'P':
                        ImprimeDocument();
                        break;
                case 'S':
                        EnregistreFichier();
                        break;
                default:
                        bInstructionValable = false;
        }
        return bInstructionValable ;
}
```

كما ترى التعليمة switch مكونة من عدة أقسام إليك بيانها:

- التعليمة switch محاطة بالأقواس ،ويجعل داخل القوسين الكلمة المراد اختبار قيمتها.
- قائمة من القيم المتوقعة كل واحدة تبدأ بالكلمة case إذا طابقت إحدى هذه القيم قيمة المتغير الموجود بين قوسين بعد الكلمة switch فإن البرنامج ينفذ جميع الأوامر الموجودة بعدها.
- الكلمة break تجبر البرنامج على الخروج من الكتلة بعد تنفيذ التعليمات الموجودة بعد الخانة المطابقة للمتغير المختَبر .

- السطر default (اختياري) ينفذ عندما لا توجد قيمة مطابقة للمتغير المختبر ضمن قائمة القيم

الأزرار المتاحة والمعطلة:

أغلب الأدوات تكون متاحة للاستعمال افتراضيا ولكن قد يحتاج في بعض الأحيان إلى تعطيل بعض الأدوات مثل تعطيل زر طباعة عند عدم وجود طابعة وتظهر الأدوات المعطلة باللون الرمادي، ويمكنك تعطيل الأزرار يدويا من خلال وضع علامة على الخاصية Disabled في نافذة الخصائص.
ويمكن تعطيلها بأوامر البرمجة تحتوي الخلية CWnd على دالة EnableWindow التي تتولى مهمة تعطيل وتفعيل الأدوات وكما قلنا سابقا الأدوات - ومنها الأزرار - نوع من النوافذ ، وكلها مشتقة من هذه الخلية فيمكن استعمالها لهذا الغرض.
لتعطيل زر اكتب كما يلي:

```
pButton→EnableWindow( FALSE );
```

يجب أن يكون الوسيط إما صحيح TRUE لتفعيل الزر أو خاطئ FALSE لتعطيله وإذا لم يتم تمرير أي وسيط فإن الزر سيفعل افتراضيا.
في المثال التالي ستعدل الدالة CButtonDlg::OnBtnTester لتعطيل وتفعيل مربع الخيار Amplifier احذف الكود السابق وأضف الكود المبين في القائمة 5-5

```
void CButtonDlg::OnBtnTester()
{
    static BOOL bAccesControle = FALSE;
    m_btnAmp.EnableWindow( bAccesControle );
    if( bAccesControle == TRUE )
        bAccesControle = FALSE;
    else
        bAccesControle = TRUE;
}
```

عندما تنقر على الزر Tester سيعطل الزر Amplifier وإذا نقرت ثانيا سيفعل وهكذا.

إخفاء الأزرار:

ستحتاج في بعض الأوقات إخفاء بعض الأدوات لتعمل في خلفية البرنامج أو لعدم الحاجة إليها غالبا يمكنك استعمال الدالة CWnd::ShowWindow للتحكم في ظهور الأداة وخفائها كما يلي:

```
pButton->ShowWindow( SW_HIDE );//إخفاء الأداة
```

ولإظهار الأداة اجعل الوسيط SW_SHOW بدل SW_HIDE كما يلي:

```
pButton->ShowWindow( SW_SHOW );//إخفاء الأداة
```

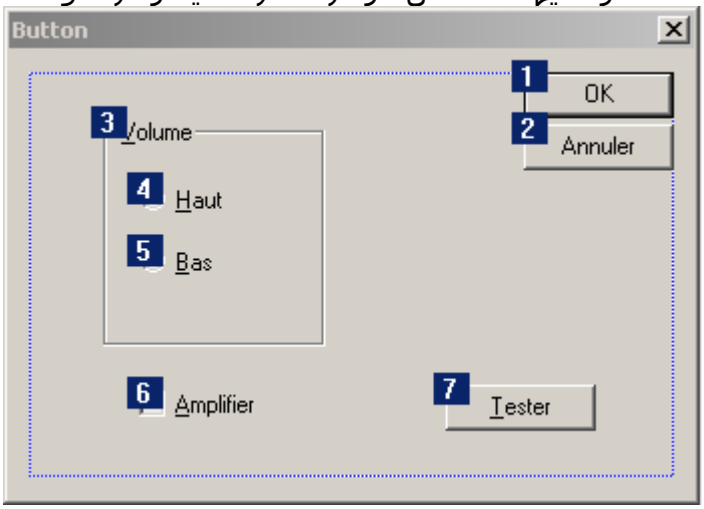
القائمة 5-6 عبارة عن مثال لإظهار وإخفاء الزر Amplifier بواسطة الدالة ShowWindow عدل الكود السابق كما يلي:

```
void CButtonDlg::OnBtnTester()
{
   static int nAfficheControle = SW_HIDE;
   m_btnAmp.ShowWindow( nAfficheControle );
   if( nAfficheControle == SW_SHOW )
       nAfficheControle = SW_HIDE;
   else
      nAfficheControle = SW_SHOW;
}
```

تحديد ترتيب الجدولة للأدوات:

عند عرض نموذج على الشاشة نرى أن أحد أدواته حصل على التركيز Focus وهو عبارة عن مربع منقط رقيق يحيط بالأداة هذا يعني أن الأداة تستجيب لإدخالات لوحة المفاتيح مثل مفتاح الإدخال Enter والهروب Esc.

ويمكن نقل التركيز من أداة إلى أخرى بواسطة المفتاح Tab دون استعمال الفأرة، كما يمكنك تحديد الترتيب الجدولي للأدوات الحاصلة على التركيز.
يرتب VC++ الأدوات حسب إنشائها على النموذج ولأعادت ترتيبها اضغط المفتاح Ctrl-D أو اختر الأمر Layout|Tab Order سيظهر أمام كل أداة رقمها الجدولي، لإعادة الترتيب انقر ببساطة على هذه الأدوات أو أداة تنقر عليها ستحمل الرقم 1 ثم الثانية رقم 2 وهكذا انظر الشكل 5-3:



كما يمكنك منع الأداة من الحصول على التركيز عند ضغط المفتاح Tab بجعل خاصيتها StopTabتساوي False من نافذة الخصائص.

الفصل السادس

مربعات النص

محتويات الفصل:

- حول مربعات النص.
- ربط كائن CEdit مع مربع نص
- التحكم في النصوص المُدخلة من المستخدم
- استعمال الروتينات DDV و DDX

حول مربعات النص.

مربعات النص: Edit Control نوافذ تسمح بعرض وتخزين النصوص المدخلة من طرف المستخدم وهي على قسمين مربع نص ذو سطر واحد ومربع نص متعدد الأسطر.
نجد مربعات النص داخل مربعات الحوار والنماذج لتمكين المستخدم من إدخال أو قراءة نصوص، مربعات النص ذات سطر واحد نجدها غالبا في النماذج التي تطلب إدخالات قصيرة من المستخدم مثل الاسم واللقب والهاتف.
أما مربعات النص ذات الأسطر المتعددة فإنها تستعمل في عرض وإدخال كمية كبيرة من النصوص مثل المقالات والدروس وتحتوي غالبا على شريط إزاحة Scroll Bar لتمكين المستخدم من معاينة كامل النص.
تدعم اغلب مربعات النص وظائف خاصة ذات فائدة كبيرة يوفرها نظام التشغيل دون الحاجة إلى برمجتها وهي مبينة في الجدول 6-1

الوظيفة	مفتاح الاختصار
قص	Ctrl-X
نسخ	Ctrl-C
لصق	Ctrl-V
تراجع	Ctrl-Z

مكتبات MFC ومربعات النص: لإضافة مربع نص إلى نموذجك اتبع نفس الطريقة التي تعلمتها مع الأزرار في الفصل الخامس، وبعد إضافة الأداة يمكنك المعالج Class Wizard ربط متغير عضو بمربع النص للتحكم فيه برمجيا ويمكن أن يكون هذا الربط بالأداة نفسها أي مع الخلية CEdit أو مع قيمة Value الأداة التي تدعمها الخلية CString.
إنشاء المشروع TestEdit: سننشئ في هذا الفصل مشروعا ذا مستند واحد SDI وسنضيف له مربع حوار لتجربة مربع نص لفعل ذلك اتبع الخطوات التالية:

1. أنشئ مشروعا MFC AppWizard(exe) ذا مستند واحد SDI وسمه TestEdit
2. أضف مربع حوار كما فعلت في الفصل الرابع، سم هذا المربع IDD_TEST واجعل عنوانه Test، ثم بواسطة المعالج Class Wizard أنشئ لهذا المربع خلية من نوع CDialog وسمها CTestDlg.
3. أضف عنصر قائمة إلى شريط القوائم تحت القائمة View وسمه ID_VIEW_TEST واجعل عنوانه Test ثم أضف دالة لمعالجة حدث النقر على هذا العنصر بواسطة Class Wizard (راجع الفصل الرابع) أضف الكود المبين في القائمة 6-1 للدالة OnViewTest التي أنشأتها للتو وظيفة هذا الكود إظهار النموذج عند النقر على عنصر القائمة.
4. أدرج تصريح الخلية CTestDlg في الملف MainFrm.cpp، أضف السطر التالي بعد ىخر عبارة #include في أعلى الملف MainFrm.cpp:

```
#include "Testdlg.h"
```

5. أضف زرا قياسيا إلى مربع الحوار وسمه IDC_TESTER واجعل عنوانه &تجريب ، ثم بواسطة المعالج Class Wizard أضف دالة تدير رسالة النقر BN_CLICKED الخاص بالزر سنستعمل هذه الدالة لاحقا.
6. ابن البرنامج بواسطة الأمر Build|Build TestEdit.exe أو بالضغط على المفتاح F7 تأكد من عدم وجود أخطاء، شغل البرنامج بالضغط على F5 وتأكد ظهور النموذج IDD_TEST عند النقر على عنصر القائمة Test

القائمة 6-1 لإظهار مربع حوار عند النقر على عنصر قائمة

```
void CMainFrame::OnViewTest()
{
    CTestDlg   dlg;
    dlg.DoModal();
}
```

إضافة مربع نص: أضف مربع نص إلى مربع الحوار كما فعلت مع الأزرار في الفصل السابق، إلا أنك تختار الأيقونة Edit Box من لوحة الأدوات، سم المربع IDC_EDIT_TEST واضبط مكان وحجم الأدوات كما هو مبين في الشكل 6-1

لا تنس أن تضيف أداة تسمية Static Text أمام المربع لإعلام المربع بنوع البيانات التي يدخلها.

خصائص مربعات النص: في الجدول 6-2 عرض ملخص لأهم خصائص مربعات النص الموجودة في نافذة الخصائص

الخاصية	الوصف
ID	الاسم البرمجي للزر ستجد **Visual C++ 6** قد اقترح اسما افتراضيا يمكنك تغييره ليعبر عن وظيفته إلا انه يستحب ترك البادئة IDC_ لبيان نوع الأداة .
Caption	عنوان النافذة الذي يظهر على الشريط الأزرق الأفقي في أعلى النافذة غيره إلى "ترحيب"
Visible	
Disabled	
Group	
Tab Stop	

ربط كائن CEdit مع مربع نص

يمكننا ربط كائن من الخلية CEdit مع مربع نص للتحكم في وظائفه، لفعل لك استخدم المعالج Class Wizard بنفس الطريقة التي فعلتها مع الأزرار في الفصل السابق.

1. شغل المعالج Class Wizard
2. حدد الشريحة Member Variables وفي القائمة Class Name حدد الخلية CTestDlg الخاصة بمربع الحوار الموجود عليه مربع النص
3. حدد اسم مربع النص في القائمة Control IDs وهو ID_EDIT_TEST
4. انقر الزر Add Variable تظهر النافذة Add Member Variable أدخل القيم الموجودة في الجدول 6-2 ثم انقر Ok يتم إضافة الكائن.

الجدول 6-2 القيم المستعملة لإضافة كائن من نوع CEdit

اسم الأداة	Member VariableName	Category	Varible Type
ID_EDIT_TEST	m_testEdit	Control	CEdit

لاحظ أن القيمة الافتراضية في القائمة Category هي Value سننشئ فيما بعد كائنا يستخدم هذا النوع.

التحكم في النصوص المُدخلة من المستخدم

تستعمل مربعات النص غالبا لحث المستخدم على إدخال نصوص معينة ثم عن البرنامج في حاجة إلى استرجاع هذه النصوص ، توفر الخلية CEdit كثيرا من الدالات التي تسهل التحكم في مربعات النصوص.
ولاسترجاع النص المدخل في مربع النص IDC_EDIT_TEST اكتب الكود المبين في القائمة 2-6 في الدالة OnTester التي تدير حدث النقر على الزر تجريب.

القائمة 2-6 استرجاع نص مربع نص بواسطة دوال الخلية CEdit

```
void CTestDlg::OnTester()
{
    CString   szEdit;     //متغير يخزن النص المدخل
    CString   szResultat; // متغير يوضع فيه النص وعدد حروفه
    int nLongueur = m_testEdit.LineLength();// تهيئة متغير رقمي بعدد حروف النص
    m_testEdit.GetWindowText( szEdit );// استرجاع النص ووضعه في المتغير szEdit
    szResultat.Format( "%s يحتوي على %d حروف",szEdit,nLongueur);
    AfxMessageBox( szResultat);

}
```

عندما تنقر على الزر تجريب فإن الدالة GetWindowText تسترجع النص وتضعه في المتغير szEdit كما تقوم الدالة LineLength باسترجاع عدد الحروف.
الدالة Format تهيئ المتغير szRsultat ليخزن النص وعدد حروفه، توضع الرموز %s و %d لبيان نوع المتغير الذي يحمل القيمة.
بعد تشغيل البرنامج ستحصل على نتيجة مشابهة للشكل 2-6

استعمال الروتينات DDV و DDX

الفصل السابع

مربعات القوائم والقوائم المنسدلة

محتويات الفصل:

- **حول القوائم.**
- **إضافة قائمة إلى مربع حوار**
- **الحلقات**

حول القوائم

1. مربعات القوائم أو القوائم البسيطة:

القوائم البسيطة Simple List Box هي مربعات تحتوي على قائمة من البنود يمكن تحديدها بواسطة الفأرة أو لوحة المفاتيح وهي على نوعين قوائم تسمح بتحديد عنصر واحد فقط وقوائم تسمح بتحديد مجموعة من العناصر ولكي تضيف المربع إلى النافذة استخدم أداة List Box من لوحة الأدوات، والعناصر التي تعرض على القائمة لن يتمكن المستخدم من إضافتها مباشرة وإنما تضاف بواسطة البرمجة.

إذا لم تكف مساحة مربع القائمة لعرض جميع البنود فإن VC++ يعرض شريط تمرير تلقائيا ليتمكن المستخدم من معاينة جميع البنود.

وبمجرد ما تنشئ مربع قائمة يمكنك إضافة بنود إليه عن طريق التعليمة التالية:

```
ListBox.AddString( "العنصر");
```

يمكن استرجاع عنصر من مربع القائمة عن طريق رقم فريد يسمى Index يبدأ التعداد في مربع القائمة من الرقم 0 فالعنصر الرابع رقمه 3 والأول 0.

مربعات القوائم ومكتبات MFC:

يتيح المعالج Class Wizard إضافة الدوال لإدارة أحداث مربع القائمة وكذا ربط متغير عضو بالخلية CListBox للتحكم في خصائصه، الخلية CListBox كبقية النوافذ مشتقة من الخلية الرئيسية CWnd التي تضم عشرات الدوال للتحكم في النوافذ ومنها الأداة List Box.

إضافة مربع قائمة:

أنشئ مشروعا MFC AppWizard(exe) جديدا من نوع Dialog Based أي مشروع معتمد على مربع حوار (راجع الفصل الرابع لمعرفة كيفية إنشاء مشروع من هذا النوع) .

بعد إنشاء المشروع افتح مربع الحوار IDD_LISTBOX_DIALOG وأضف له مربع قائمة من لوح الأدوات سم مربع القائمة IDC_LISTBOX .

افتح نافذة خصائص مربع القائمة بالضغط بالزر الأيمن على مربع القائمة ثم اختيار خصائص من القائمة المنسدلة.

خصائص مربع القائمة:

كما هو الحال مع بقية الأدوات يملك مربع القائمة خصائص يمكن ضبطها من نافذة الخصائص هناك خصائص مشتركة بين بقية الأدوات مثل الاسم والظهور لا داعي لتكرارها.

الخاصية	الوصف
ID	الاسم البرمجي للزر ستجد **Visual C++ 6** قد اقترح اسما افتراضيا يمكنك تغييره ليعبر عن وظيفته إلا انه يستحب ترك البادئة IDC_ لبيان نوع الأداة .
Caption	عنوان النافذة الذي يظهر على الشريط الأزرق الأفقي في أعلى النافذة غيره إلى "ترحيب"
Visible	
Disabled	
Group	
Tab Stop	

الخلية CListBox:

تستعمل الخلية CListBox لإدارة مربعات القوائم ويجب إنشاء متغير عضو من هذه الخلية وربطه بالأداة لفعل ذلك اتبع ما يلي:

1. شغل المعالج Class Wizard
2. حدد الشريحة Member Variables وفي القائمة Name حدد الخلية CListBoxDlg
3. في القائمة Control IDs حدد اسم مربع النص IDC_LISTBOX
4. انقر الزر Add Variable تظهر النافذة Add Member Variable أدخل القيم الموجودة في الجدول 6-2 ثم انقر Ok يتم إضافة الكائن.

الجدول 7-2 القيم المستعملة لإضافة كائن من نوع CListBox

Varible Type	Category	Member VariableName	اسم الأداة
CListBox	Control	m_listBox	IDC_LISTBOX

إضافة بنود إلى مربع القائمة:

هناك طريقتان إضافة سلسلة نصية كبند لمربع القائمة .

1. الدالة AddString مثل: m_listBox.AddString("Ahmed");
العناصر الجديدة تضاف إلى نهاية سلسلة العاصر إذا كانت قيمة الخاصية Sort تساوي False .
2. الدالة InsertString مثل: m_listBox.InsertString(0,"Ahmed");
هذه الدالة مثل الأولى إلا أنها تترك لك الخيار في تحديد المكان الذي يوضع فيه العنصر الجديد رقم العنصر يحدد في الوسيط الأول وعند إضافة عنصر بهذه الدالة ستتأثر باقي العناصر ويعاد ترتيبها حسب المكان الذي وضع فيه هذا العنصر.

وكلتا الدالتين تعودان بقيمة عددية تحمل رقم Index العنصر، إذا لم يتم إضافة العنصر لحدوث خطأ غير متوقع كنفاذ الذاكرة مثلا فإن القيمة المعادة هي الثابت LB_ERRSPACE أضف الكود المبين في القائمة 7-1 إلى دالة تهيئة النموذج CListBoxDlg::OnInitDialog لإضافة ستة عناصر إلى مربع القائمة IDC_LISTBOX.

ستجد بعض الكود أنشئ بواسطة المعالج AppWizard في الدالة CListBoxDlg::OnInitDialog لا تحذفه واكتب مباشرة بعد السطر //TODO .

القائمة 7-1 استعمال الدالة AddString لإضافة بنود لمربع القائمة

```
// TODO: Add extra initialization here
m_listBox.AddString("تمر");
m_listBox.AddString("تين");
m_listBox.AddString("زيتون");
m_listBox.AddString("قمح");
m_listBox.AddString("شعير");
m_listBox.AddString("موز");
```

ملاحظة :يمكنك وضع علامة على الخاصيتين Right Aligned Text و Right-To-Left Reading Order في الشريحة Extended Styles من نافذة الخصائص لعرض عناصر القائمة من اليمين إلى اليسار.

ولمعرفة عدد بنود مربع القائمة استعمل الدالة GetCount التابعة للخلية CListBox:

```
nItemCount = m_listBox.GetCount();
```

ملاحظة: هذه الدالة لا تعيد نص العنصر الأخير وإنما تعيد عدد العناصر الموجودة في مربع القائمة فلو كان هناك خمسة عناصر فستعيد الدالة رقم 4 لأن العد يبدأ كما علمت من الصفر.

حذف العناصر من مربع القائمة: لحذف عنصر من القائمة استعمل الدالة DeleteString ومرر لها رقم العنصر المراد حذفه:

```
ListBox.DeleteString(8);
```

هذا السطر يحذف العنصر التاسع من مربع القائمة (لا تنس أن العد يبدأ من الصفر)، والدالة DeleteString تعيد عدد العناصر الباقية بعد الحذف أو الثابت LB_ERR إن كانت هناك أخطاء عند الحذف ، يمكنك استعمال القيمة المعادة كما يلي:

```
int nEntrees = m_listBox.GetCount();
while( nEntrees > 3 && nEntrees != LB_ERR )
        nEntrees = m_listBox.DeleteString(nEntrees-1);
```

هذا الكود يحذف جميع عناصر مربع القائمة عدا العناصر الثلاثة الأولى.

ملحوظة: المعامل && يعني "و" And والمعامل &! يعني لا يساوي <>.

لمسح جميع محتويات القائمة استعمل الدالة RestContent
```
ListBox.ResetContent();
```
الدالة ResetContent لا تعيد شيئا Void.

أحداث مربع القائمة:

يرسل نظام التشغيل ويندوز عدة رسائل خاصة بمربع القائمة عند وقوع بعض الأحداث عليها، تسمى هذه الرسائل Notification وتصدر بالبادئة LBN_ أي List Box Notification ولن يتمكن برنامجك من التعامل مع هذه الرسائل حتى تكون الخاصية Notify التابعة لمربع القائمة تساوي True وهذه الرسائل هي:

- LBN_DBLCLK ويتم إرسالها عندما ينقر المستخدم نقرا مزدوجا على عنصر في مربع القائمة.
- LBN_ERRSPACE عندما تفشل عملية خاصة بمربع القائمة بسبب نقص في الذاكرة
- LBN_KILLFOCUS عندما يفقد مربع القائمة التركيز
- LBN_SELCANCEL عندما يلغي المستخدم تحديد عنصر في مربع القائمة
- LBN_SELCHANGE عندما يحدد المستخدم عنصرا في مربع القائمة

الرسالة الأكثر استعمالا من بين هذه الرسائل رسالة النقر المزدوج LBN_DBLCLK فالمستخدمون أغلب ينتظرون أن يفعل البرنامج شيئا ما عندما ينقرون نقرا مزدوجا على أحد عناصر مربع القائمة كفتح ملف أو تصفية بيانات على حساب عنصر محدد.

وتتم إضافة الدوال التي تعالج هذه الرسائل كبقية الرسائل الأخرى التي سبق لك معرفتها ، ولإنشاء دالة تعالج رسالة النقر المزدوج اتبع هذه الخطوات:

1. افتح Class Wizard وحدد الشريحة Message Maps
2. حدد الخلية CListBoxDlg والاسم ID_LISTBOX
3. في قائمة الرسائل حدد الرسالة LBN_DBLCLK ثم انقر على الزر Add Function
4. اقبل الاسم OnDblclkListbox المقترح من طرف المعالج
5. انقر على الزر Edit Code
6. أضف كود القائمة 7-2 إلى الدالة OnDblclkListbox

```
void CListBoxDlg::OnDblclkListbox()
{
    int nSelection = m_listBox.GetCurSel();
    if( nSelection != LB_ERR )
    {
       CString szSelection;
       m_listBox.GetText( nSelection,szSelection);
       AfxMessageBox( szSelection );
     }
}
```

فسر وشغل البرنامج ثم انقر نقرا مزدوجا على عنصر في مربع القائمة سترسل الرسالة LBN_DBLCLK إلى الدالة OnDblclkListbox ستنفذ الدالة الأوامر المطلوب منها وستعرض رسالة بنص العنصر المحدد.

الدالة GetCurSel تتيح معرفة رقم العنصر المحدد إذا لم يكن هناك عنصر محدد في القائمة ستعود الدالة بالقيمة LB_ERR.

ستحصل على نتيجة شبيهة بالشكل 7-1

2. القوائم المنسدلة أو القوائم المزدوجة:

القوائم المنسدلة Combo Box أدوات مكونة من أداتين مربع النص ومربع القائمة ولهذا يسمى المربع المزدوج إنها تتيح للمستخدم إما كتابة نص في مربع النص أو اختيار عناصر من مربع القائمة .

ولن يتمكن المستخدم من معاينة مربع القائمة إلا إذا نقر على السهم الموجود بجانب مربع النص .

وتنقسم القوائم المنسدلة إلى ثلاثة أقسام :

- مربع مزدوج منسدل على شكل قائمة *drop-down list combo box* هذا المربع يعمل مثل مربع القائمة، ولكنه يتميز عليه بأنه لا يحتل مساحة كبيرة من الشاشة مثل القائمة وإنما يظهر مغلقا ويحتل سطرا واحدا فقط حتى يقوم المستخدم بفتحه وعرض محتوياته سيتمكن المستخدم من اختيار أحد البنود في المربع ولكنه لن يتمكن من كتابة بنود جديدة.
- مربع مزدوج بسيط *simple combo box* يعمل مثل مربع القائمة العادي ولكنه يحتوي على مربع نص في أعلاه، مما يتح للمستخدم اختيار إحدى القيم الموجودة في القائمة أو كتابة قيمة جديدة في مربع النص الموجود أعلاه.
- مربع مزدوج منسدل *drop down combo box* هذا المربع يوفر المساحة على النافذة لأنه يظل مغلقا ويحتل سطرا واحدا فقط حتى يقوم المستخدم بفتح بعد ذلك يمكن للمستخدم اختيار أحد البنود الموجودة في المربع أو كتابة قيمة جديدة.

خصائص المربع المزدوج:

يضم المربع المزدوج عددا كبيرا من الخصائص ولكن أغلب هذه الخصائص مشتركة مع مربعات النص ومربعات القائمة هناك خاصيتين ينفرد بهما المربع المزدوج عنهما وهما:

الخاصية	الوصف
Enter list box item	تمكنك من إضافة بنود في وقت التصميم للتفريق بين البنود اضغط Ctrl-Enter
Type	تحدد نوع المربع من أحد الأنواع الثلاثة التي ذكرناها سابقh

المربع المزدوج ومكتبات MFC:

كما هو الحال بالنسبة لمربع القائمة يمكنك إضافة المربع المزدوج من لوح الأدوات من الأيقونة Combo Box وإنشاء متغير عضو من الخلية CComboBox للتحكم برمجيا في هذه الأداة

إضافة عناصر إلى المربع المزدوج:

عموما كل أنواع هذا المربع تتشابه مع مربع القائمة في طريقة إضافة البنود إليها واسترجاعها وإزالتها منها والاختلاف الأساسي هو في طريقة الظهور فقط هذا السطر لإضافة بند في المربع:

```
coboBox.AddString( "Ahmed" );
```

أو:

```
comboBox.InsertString( "Ahmed" );
```

وعناصر المربع المزدوج يبدأ ترقيمها أيضا بالصفر ، أما القيم المعادة في حالة حصول خطأ فإنها مثل مربع القائمة إلا أنها تصدر بالبادئة CB_ بدل LB_ مثلا حصول خطأ لإضافة بسبب نفاذ الذاكرة فإن القيمة المعادة هي CB_ERRSPACE.

السطر التالي يسترجع عدد البنود الموجودة في المربع المزدوج تماما مثل مربع القائمة:

```
nEntrees = comboBox.GetCount();
```

وتعاد القيمة CB_ERR إذا كانت هناك أخطاء.

إدارة النص المدخل بواسطة المستخدم:

يمكنك استرجاع النص المدخل في مربع نص من طرف المستخدم بواسطة الدالة GetWindowItem تماما مثل مربعات النص العادية ويمكنك بواسطة الدالة GetCurSel استرجاع رقم العنصر المحدد من طرف المستخدم مثل مربع القائمة.

ولاسترجاع نص عنصر ما في المربع المزدوج استعمل الدالة GetLBText ، تستعمل هذه الدالة وسيطين الأول متغير عددي يحمل رقم العنصر المراد استرجاع نصه ، والثاني متغير من نوع CString لتخزين النص المعاد فيه.

```
m_combo.GetLBText( 1,szItemSel );
```

البحث في المربع المزدوج:

يمكنك البحث عن سلسلة نصية خاصة في المربع المزدوج ، استعمل الدالة FindString والدالة FindStringExact

الدالة FindString تستعمل للبحث عن عناصر تبدأ بنص معين :

```
int index = m_combo.FindString( -1; szFind );
```

هذه الدالة تستعمل وسيطين الأول عددي لتحديد رقم العنصر الذي سيبدأ البحث منه يستعمل الرقم -1 للبحث من بداية المربع، والوسيط الثاني حرفي يحدد الكلمة المبحوث عن عناصر تبدأ بها.

أما الدالة FindStringExact مثل الأولى إلا أنها تستعمل للبحث عن عنصر يطابق السلسلة المبحوث عنها تماما.

وكلتا الدالتين تعيدان رقم العنصر الأول إن وجد وإلا فسيعيدان القيمة CB_ERR إن لم توجد أي نتيجة.

مشروع المربع المزدوج:

سننشئ مشروعا لنتدرب فيه على المربع المزدوج اتبع الخطوات التالية:

1- أنشئ مشروعا معتمدا على مربع حوار وسمه ComboBox
2- أضف إلى المربع الحوار IDD_COMBOBOX_DIALOG أداة مربع مزدوج
3- أعط الاسم IDC_COMBO للمربع المزوج واترك جميع الخصائص الأخرى على قيمها
4- أضف أداة من نوع Static Text إلى مربع الحوار وسمها IDC_RESULTAT سنستعمل هذه التسمية لعرض بعض النصوص المسترجعة من المربع.
5- شغل المعالج Class Wizard وحدد الخلية CComboBoxDlg وأضف متغير عضو خاص بالأداة باسم m_comboList واختر القيمة Control في الخاصية Category
6- بواسطة Class Wizard أيضا أضف دالتين إحداها لتلقي الرسالة CBN_CLOSEUP والأخرى لتلقي الرسالة CBN_EDITUPDATE واقبل اسمي الدالتين المقترحيْن من طرف المعالج.

إضافة عناصر إلى المربع المزدوج:

بعد عكمالك الخطوات السابقة أضف الكود المبين في القائمة 7-3 إلى دالة التهيئة OnInitDialog الخاصة بمربع الحوار الرئيسي هذا الكود يضيف ثلاثة عناصر إلى المربع المزدوج ، ستجد بعض الكود أنشئ بواسطة المعالج AppWizard في الدالةOnInitDialog لا تحذفه واكتب مباشرة بعد السطر //TODO .

القائمة7-3 إضافة بنود لمربع القائمة المنسدلة

```
// TODO: Add extra initialization here
m_comboList.AddString("محمد");
```

```
m_comboList.AddString("أحمد");
m_comboList.AddString("محمود");
```

التعرف على أحداث المربع المزدوج:

أضف كود القائمة 7-4 إلى الدالة CComboBoxDlg::OnCloseupCombo عندما تُستقبل الرسالة CBN_CLOSEUP ستعرض رسالة في التسمية IDC_RESULTAT:

```
void CComboBoxDlg::OnCloseupCombo()
{
	// TODO: Add your control notification handler code here
	CString		szChoix;
	CString		szResultat;
	int		nChoix;
	//استرجاع العنصر المحدد في المربع
	//أو المكتوب في مربع النص
	m_comboList.GetWindowText( szChoix );
	nChoix = m_comboList.GetCurSel();
	if( nChoix != CB_ERR )
	{
		//  إذا تم اختيار عنصر ولم تكن هناك أخطاء سيخزن في szChoix
		m_comboList.GetLBText( nChoix,szChoix );
		szResultat = "لقد اخترت " + szChoix;
	}
	else if(szChoix.IsEmpty() == TRUE )
	{
		//  إذا لم يتم اختيار أي عنصر ولم يكتب أي نص
		szResultat = "لا يوجد عنصر محدد ";
	}
	else if( m_comboList.FindStringExact(-1,szChoix)!= CB_ERR)
	{
		//لقد تم إدخال نص في المربع المزدوج
		szResultat =  "لقد اخترت " + szChoix;
	}
	else
	{
		//إضافة العنصر المدخل إلى القائمة
		m_comboList.AddString( szChoix );
		szResultat="إضافة العنصر" + szChoix + " إلى القائمة";
	}
	// إنشاء مؤشر نحو التسمية وعرض الرسائل فيها
	CWnd* pWnd = GetDlgItem( IDC_RESULTAT );
	ASSERT( pWnd );
	if( pWnd )
		pWnd->SetWindowText( szResultat );
}
```

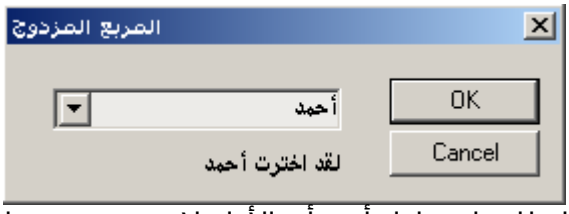
الدالة CComboBoxDlg::OnCloseupCombo تتأكد إن كان عنصر محدد في القائمة أو تم إدخال نص إلى مربع النص لتعرضه على التسمية

أضف كود القائمة 7-5 إلى الدالة CComboBoxDlg::OnEditupdateCombo سترسل الرسالة CBN_EDITUPDATE إلى الدالة عند تغيير محتوى المربع المزدوج بكتابة شيء أو حذفه بعده سيكتب النص في التسمية IDC_RESULTAT.

```
void CComboBoxDlg::OnEditupdateCombo()
{
	CString		szChoix;
	CString		szResultat;
	m_comboList.GetWindowText( szChoix );
	szResultat = "لقد كتبت الآن " + szChoix;
	CWnd* pWnd = GetDlgItem( IDC_RESULTAT );
	ASSERT( pWnd );
	if( pWnd )
		pWnd->SetWindowText( szResultat );
}
```

فسر البرنامج ثم شغله حاول أن تكتب نصا في خانة نص المربع المزدوج افتح القائمة المنسدلة وأغلقها اقرأ مختلف الرسائل التي تظهر في التسمية بعد كل عملة تقوم بها.

ملاحظة :يمكنك وضع علامة على الخاصيتين Right Aligned Text و Right-To-Left Reading Order في الشريحة Extended Styles من نافذة الخصائص لعرض عناصر المربع المزدوج من اليمين إلى اليسار انظر الشكل 2-7 :



ملحوظة أخرى: بعد تشغيلك للبرنامج إذا رأيت أن الأداة لا تعرض محتوياتها ارجع إلى طور التصميم وحدد الأداة ثم انقر على السهم يظهر إطار كبر هذا الإطار للأسفل بحسب ما يتسع للقائمة.

الحلقات:

لقد تعلمت في الفصل الخامس كيفية استعمال التعليمات الشرطية للتحكم في سير برنامجك وزيادة مرونته ، الحلقات تتيح لك تكرار مجموعة معينة من الأوامر وفق عدد معين من الدورات أو شرط معين يعني إيقاف التكرار عندما يتحقق شرط معين كان يقوم البرنامج بطباعة دروس معينة حتى تدخل له الكلمة إنهاء أو فتح ملف معين وقراءة محتواه حتى يصل إلى نهايته بأن تصبح الخاصية EOF (*End Of File*)صحيحة والحلقات في VC++ ثلاثة أنواع:

- الحلقة while
- الحلقة do-while
- الحلقة for

الحلقة while: الحلقة while تقوم بتنفيذ وتكرار أمر أو أكثر بناء وفق شرط معين أي تكرار هذه الأوامر مادام هذا الشرط صحيحا وإلا أوقف التكرار وأكمل بقية البرنامج كود القائمة 7-6 يوضح كيفية استعمال هذه الحلقة:

```
	CString	szMsg;
	int nCompteurBoucle = 0;
	while (nCompteurBoucle < 5 )
	{
		szMsg.Format(" رقم العداد %d",nCompteurBoucle);
		nCompteurBoucle++;
		AfxMessageBox(szMsg );
	}
```

هذه الحلقة تعرض رسالة بقيمة المتغير nCompteurBoucle مادامت أقل من خمسة بمجرد ما تصبح قيمة المتغير أكثر من خمسة فإن البرنامج سينهي الحلقة ويكمل بقية الأوامر .

الحلقة do-while: الحلقة do-while مثل الحلة while إلا أنها تنفذ الأوامر بعدها مرة واحدة على الأقل ثم إن وجدت الشرط صحيحا أكملت دورانه حتى يصبح الشرط خاطئا وإلا خرجت من الحلقة كود القائمة 7-7 مثال عن هذه الحلقة:

```
#include <iostream>
using namespace std;
int main()
{
	char ch;
	do{
		cout << "\n Prees in 'q' to exit ->";
		cin >> ch;
		//انتظار ضغط المستخدم على مفتاح الإدخال
		cin.ignore(120,'\n');
	}while( ch != 'q' );
	cout << "Bay Bay !" << endl;

	return 0;
}
```

الحلقة for : نوع من التكرار يتم تنفيذه عددا محددا من المرات وفق عداد معين يعرف ضمن الحلقة وليس هناك اختلاف كثير بين هذه الحلة والحلقة while يمكننا استبدال كود القائمة 6-7 بهذا الكود :

```
	CString		szMsg;
	//تعريف العداد وإعطاؤه قيمة أولية ثم الشرط ثم الكمية التي يزاد بها العداد
	for (int nCompteurBoucle =0;nCompteurBoucle <5;nCompteurBoucle++ )
	{
		szMsg.Format(" رقم العداد %d",nCompteurBoucle);
		AfxMessageBox(szMsg );
	}
```

الفصل الثامن

الرسائل والقوائم

محتويات الفصل:

- **ماذا تعني رسالة**
- **مشروع يراقب نقرات الفأرة**
- **إدارة الرسائل بواسطة Class Wizard**
- **خلايا MFC**

حول البرمجة لويندوز.

البرامج

مفهوم الرسائل.

ترسَل الرسائل إلى كل النوافذ التي يمكنها استقبال أحداث من المستخدم أو من جهة أخرى، فبمجرد تحريك بسيط لمؤشر الفأرة فوق نافذة برنامجٍ لويندوز يولد عددا كبيرا من الرسائل فالحدث هنا هو تحريك الفأرة ، والرسالة هي القيمة التي يبعثها ويندوز لبرنامجك يبين فيها نوع الحدث مثل WM_MOUSMOVE والبرنامج يرسل الرسالة إلى الدالة الخاصة بهذه الرسالة كالدوال التي تنشئها بالمعالج Class Wizard فتنفذ الدالة الأوامر المكتوبة فيها.

والرسائل المرسلة إلى نافذة ما يرتبها البرنامج بحسب ورودها واحدة بعد أخرى ثم يتفحصها مرتبة فيتعامل مع الرسائل المعلنة - أي التي وضعت لها دوال تعالجها - ويتجاهل الأخرى.

كيف تعالج الرسائل:

عندما يحرك المستخدم مؤشر الفأرة فوق النافذة الرئيسية للبرنامج تبعث رسالتان إلى البرنامج:

- الرسالة WN_NCMOUSEMOVE تبعث عند تحريك مؤشر الفأرة فوق القوائم أو شريط العنوان.
- الرسالة WN_MOISEMOVE عند تحريكه فوق منطقة المستخدم أي فوق النافذة.

هناك رسائل أخرى خاصة بالفأرة مثل الرسالة WM_LBUTTONDOWN تبعث عند الضغط على الزر الأيسر للفأرة والرسالة WM_RBUTTONDOWN تبعث عند الضغط على الزرالأيمن للفأرة وهناك رسائل أخرى ستعرفها في الفصول القادمة إن شاء الله.

إدارة الرسائل بواسطة Class Wizard

يتيح المعالج Class Wizard إنشاء الدوال والوظائف لمعالجة مختلف الرسائل . للعلم أنه يمكنك إنشاء هذه الوظائف يدويا ودون استعمال المعالج ولكن ينصح باستعمال المعالج لضمان سلامة الكود وتوفير الوقت ،

كود القائمة 8-1 مثال عن وظيفة منشأة بواسطة المعالج Class Wizard لمعالجة الرسالة LBUTTONDOWN:

```
void CTestView::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    CView::OnLButtonDown(nFlags, point);
}
```

التعريف بالقوائم:

القوائم جمع قائمة وهي عبارة عن نافذة تضم مجموعة من الأوامر التي يمكن تحديدها لتقوم بعمل محدد مثل إظهار مربع حوار أو طباعة أو حفظ ملف، وكل عنصر من القائمة يحتوي على نص قصير يبين وظيفته، كما أن لكل عنصر معرف ID يمكن التحكم برمجيا في هذا العنصر من خلال معرفه (اسمه) .

والقوائم نوع من الموارد التي يمكن إنشاؤها بواسطة محرر الموارد الذي يأتي مع VC++ والقوائم تنشئ مستقلة ثم تربط النوافذ بها من خلال الخاصية Menu .

ويمكن أيضا إنشاء القوائم الموضعية أو التي تسمى قوائم السياق والتي تظهر عند النقر بزر الفأرة الأيمن.

وهناك بعض الضوابط المتفق عليها بين المبرمجين في القوائم:

- كل عنصر يفتح مربع حوار يتبع في تسميته بثلاث نقاط ... مثل (&فتح...).
- يحتوي كل عنصر على مفتاح التسريع ويعرف بكونه مسطرا مفتاح التسريع يتيح لنا تحديد هذا العنصر عن طريق الضغط على المفتاح Alt و الحرف المسطر ويبغي في الأوامر القياسية مثل فتح إنهاء أن تجعل مفاتيح التسريع نفس المفاتيح التي في البرامج القياسية الأخرى مثل وورد وإكسيل حتى تختلط على المستخدمين كما يجب أن لا يكرر مفتاح التسريع لعنصري قائمة ويمكن إنشاء مفتاح التسريح بجعل الرمز & قبل الحرف الذي نريد جعلها مفتاح تسريع في الخاصية Caption للعنصر.
- معرف أو اسم ID العنصر كذلك ينبغي جعله وفق ما اتفق عليه المبرمجون كما سنذكره فيما بعد.
- لا يجعل VC++ عنصر القائمة متاحا للتحديد حتى تضع له دالة لمعالجة الرسائل.

إنشاء القوائم:

يتيح VC++ إنشاء القوائم بطريقة ديناميكية أي بواسطة أوامر البرمجة ، أو عن طريق محرر القوائم المشمول مع محرر موارد VC++.

توفر مكتبات MFC الخلية CMenu التي تتيح إنشاء وتعديل القوائم بطريقة ديناميكية.

إضافة عنصر قائمة جديد:

من أجل مثالنا هذا أنشئ مشروعا جديدا من نوع SDI وسمه Menu.

إن إضافة القوائم عملية سهلة ومرنة لا يلزمك في ذلك سوى خطوتان:

- إضافة العنصر عن طريق محرر القوائم.
- إضافة الدالة التي تعالج أحداث هذا العنصر عن طريق المعالج Class Wizard

للعلم فإن المعالج AppWizard ينشئ تلقائيا عند إنشاء المشاريع ذات المستندات مورد قائمة فيه بعض الأوامر الرئيسية للبرنامج مثل فتح وحفظ.

لفتح مورد القوائم هذا حدد الشريحة ResourceView في الإطار WorkSpace افتح المجلد الرئيسي للموارد تعرض مجلدات لأنواع مختلفة من الموارد افتح المجلد الذي يحمل اسم Menu لعرض موارد القوائم:

المشروعات من نوع MDI ينشئ لها المعالج AppWizard قائمتين:

- القائمة IDR_MAINFRAME تستعمل عندما يكون أي منظر View مفتوح في النافذة الرئيسية.
- القائمة الثانية تبدأ ب IDR_ ثم اسم المشروع ثم الكلمة TYPE فمثلا لو كان لمشروعنا هذه القائمة لكان اسمها IDR_MENUTYPE، تستعمل هذه القائمة عندما يكون منظر أو أكثر معروض في النافذة الرئيسية.

أما المشاريع ذات المستند واحد SDI كمشروعنا هذا فإن المعالج ينشئ لها قائمة واحدة وهي IDR_MAIFRAME.

تعديل قائمة المشروع:

انقر نقرا مزدوجا على القائمة ID_MAINFRAM لعرضها في محرر القوائم، ستظهر القائمة كما تظهر القوائم الأخرى العادية فيمكنك فتح عنصر قائمة رئيسي ومعاينة عناصره الفرعية.

لاحظ وجود مربع فارغ في آخر كل قائمة هذا المربع يتيح لك إضافة عناصر جديدة ولإضافة عنصر قائمة جديد افعل ما يلي:

1- انقر نقرا مزدوجا على المربع الفارغ في القائمة ملف File ستظهر نافذة الخصائص Menu Item Properties.
2- اكتب معرف العنصر في الخانة ID وكما ذكرت من قبل فإن أسماء ومعرفات الأدوات تخضع لاتفاقات عالمية ، معرفات عناصر القوائم تصدر بالبادئة ID_ ثم اسم القائمة الرئيسية التي يندرج تحتها هذا العنصر مثل ID_FILE ثم اسم العنصر والذي ينبغي أن يكون معبرا عن وظيفته. في مشروعنا هذا سم العنصر الجديد ID_FILE_HELLO.
3- اكتب نص العنصر Hello& في الخانة Caption واستخدم الرمز & لإنشاء مفتاح التسريع.
4- يمكنك إضافة وصف مختصر لمهمة العنصر، يظهر هذا العنصر في شريط الحالة Status Bar عند تمرير مؤشر الفأرة فوق هذا العنصر.
5- انقر خارج نافذة الخصائص للعودة إلى المحرر.

بعد الانتهاء من إنشاء العنصر الجديد يمكنك تحديد مكانه ضمن القائمة File وذلك بسحبه بواسطة الفأرة إلى المكان المقصود.

إضافة دالة معالجة أحداث القوائم:

الآن ننتقل إلى الخطوة الثانية وهي إنشاء دالة إلى العنصر الجديد نضع فيها الأوامر التي ستنفذ عند تحديد هذا العنصر.

1- افتح المعالج Class Wizard بالضغط على Ctrl+W
2- حدد الشريحة Message Maps ثم حدد في القائمة المنسدلة Class Name الخلية CMainFrame التي ستدير هذه الدالة.
3- حدد اسم عنصر القائمة ID_FILE_HELLO الذي أنشأناه للتو ستظهر رسالتان في قائمة الرسائل Messages:
4- حدد الرسالة COMMAND، ترسل هذه الرسالة عند تحديد عنصر القائمة.
5- انقر الزر Ok لغلق المعالج.

أضف السطر التالي إلى الدالة الجديدة CMainFrame::OnFileHello كما في القائمة 10-1.

```
void CMainFrame::OnFileHello()
{
        AfxMessageBox("العنصر الجديد يقول لكم السلام عليكم");
}
```

سيبدو برنامجك كما في الشكل 10-1:

إنشاء القوائم الموضعية:

القائمة الموضعية أو المختصرة أو قائمة السياق هي التي تظهر عند النقر بزر الفأرة الأيمن وتشكل عادة اختصارا لأهم الأوامر الموجودة في البرنامج.

يتم إنشاء القوائم الموضعية بتصميم مورد قوائم Resource Menu جديد لها. تدعم مكتبات MFC الرسالة WM_CONTEXTMENU التي ترسل عند النقر على الزر الأيمن للفأرة إن فهي أنسب مكان لعرض القوائم الموضعية.

إنشاء مورد القائمة:

استخدم محرر القوائم لإنشاء قائمة جديدة ويمكنك إضافة قائمة جديد بإحدى طريقتين:

- باختيار الأمر Insert | Resource... ومن مربع الحوار Insert Resource اختر العنصر Menu ثم انقر الزر New
- النقر بالزر الأيمن على المجلد Menu في ResourceView ثم اختيار Insert Menu من القائمة الموضعية.

أيا كانت الطريقة المستخدمة فإن VC++ سينشئ بعد ذلك قائمة جديدة فارغة لا تحمل سور مربعا فارغا.انقر بالزر الأيمن على اسم القائمة الجديدة في الشريحة ResourceView واختر خصائص غير اسم القائمة من IDR_MENU1 إلى IDR_POPUP.

بعد ذلك انقر نقرا مزدوجا على المربع الفارغ تظهر نافذة الخصائص Menu Item Properties أدخل مسافة في الحقل Caption أي سطر فارغ وذلك بقضيب المسافة. هذه المسافة تمكننا من إنشاء عنصر رأس للقائمة الجديدة.

أضف بعد ذلك ثلاثة عناصر إلى القائمة الجديدة كما هو مبين في الجدول 10-1:

المعرف ID	التسمية Caption
ID_LIONS	الأ&سود
ID_TIGRES	الن&مور
ID_PANTHERES	ال&فهود

إضافة وظيفة برمجية لإدارة الرسائل:

سنظهر هذه القائمة الموضعية عند النقر بالزر الأيمن على منظر View البرنامج وعندما نحدد عنصرا منها سيتم عرض رسالة داخل المنظر قرب القائمة.
أولا يجب تصريح متغيرين في الخلية CMenuView متغير من نوع CString يخزن سلسة الرسالة ومتغير من نوع CPoint الذي يخزن معلومات خاصة بموقع القائمة الموضعية على المنظر.
أضف كود القائمة 10-2 إلى الخلية CMenuView غي الملف MenuView.h قبل التعليق // Implementation:

```
protected:
    CPoint m_ptMsg;
    CString m_szMsg;
```

يجب علينا تهيئة المتغير m_ptMsg بقيمة افتراضية عن موقع المؤشر عند تشغيل البرنامج ،أفضل مكان لتهيئته هو حدث بناء الخلية ، حدد الدالة CMenuView::CMenuView ثم اكتب السطر التالي كما هو في القائمة 10-3:

```
CMenuView::CMenuView()
{
      m_ptMsg = CPoint(0,0);
}
```

ملاحظة:
كما قلنا سابقا حدث البناء والهدم يكون بنفس الاسم الخلية وبدون أي وسيطات أو قيمة معادة إلا حدث الهدم يكون مسبوقا بالرمز ~ ويمكنك الوصول إلى دالة حدث البناء وهي CMenuView::CMenuView كما يلي:

1. حدد الشريحة ClassView من الإطار WorkSpace
2. افتح علامة الجمع للخلية CMenuView
3. ابحث عن الدالة CMenuView وانقر عليها نقرا مزدوجا.

كما يمكنك من هنا تصريح المتغيرات بطريقة سهلة عبر مايلي:

1. انقر بالزر الأيمن على الخلية CMenuView
2. من القائمة الموضعية اختر Add Member Variable ثم صرح متغيرك في النافذة Add Member Variable مثل الشكل 10-2

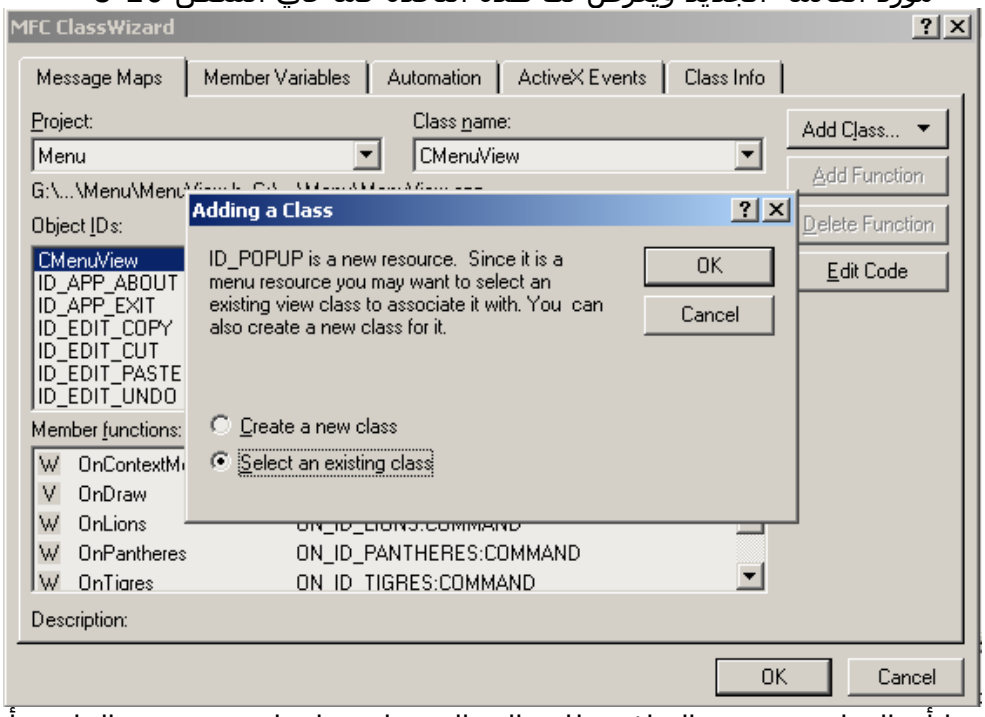
سنعدل الآن الدالة CMenuView::OnDraw التي تتولى عملية طبع الرسالة على المنظر اكتب الكود المبين في القائمة 10-4:

```
void CMenuView::OnDraw(CDC* pDC)
{
    pDC->TextOut( m_ptMsg.x,m_ptMsg.y,m_szMsg);
}
```

معالجة الرسائل:
كما رأينا سابقا تحديد عنصر قائمة يؤدي إلى إرسال الرسالة COMMAND إلى البرنامج وهو بدوره يمررها إلى الدالة الخاصة بالعنصر المحدد لتنفيذ أوامر معينة، سننشئ لكل عنصر من

العناصر الثلاثة وظيفة النقر الخاصة به كما سننشئ الوظيفة التي تتولى إظهار القائمة الموضعية عند النقر بالزر الأيمن على المنظر إذن سننشئ أربع وظائف.

1- حدد القائمة ID_POPUP وافتح المعالج Class Wizard بالضغط على Ctrl+W أو بالنقر بالزر الأيمن في المحرر واختيار Class Wizard من القائمة الموضعية سيتعرف المعالج على مورد القائمة الجديد ويعرض لك هذه النافذة كما في الشكل 10-3



بما أن القوائم نوع من النوافذ يطلب المعالج منك ربط خلية موجودة بالقائمة أو إنشاء خلية موجود حدد الخيار الافتراضي وهو الثاني واربط القائمة بالخلية CMainFrame في النافذة Select Class ثم انقر الزر Select للرجوع إلى المعالج.

2- حدد الشريحة Message Maps واختر من القائمة المنسدلة Class Name الخلية الخاصة بالمنظر هنا CMenuView

3- في القائمة Object ID حدد عنصر القائمة الذي يولد الرسالة واحدا بعد واحد كما في الجدول 10-2:

4- حدد الرسالة المعينة في القائمة Messages ثم انقر الزر Add Function اقبل الاسم المقترح للدالة من طرف المعالج Class Wizard.

5- أعد هذه الخطوات للعناصر الباقيةكما هو موضح في الجدول.

6- انقر Ok لغلق المعالج.

المعرف ID	الرسالة Message	اسم الوظيفة أو الدالة
CMenuView	WM_CONTEXTMENU	OnContextMenu
ID_LIONS	COMMAND	OnLions
ID_TIGRES	COMMAND	OnTigres
ID_PANTHERES	COMMAND	On_Pantheres

كود الدالة CMenuView::OnContextMenu مبين في القائمة 10-5

```
void CMenuView::OnContextMenu(CWnd* pWnd, CPoint point)
{
	CMenu zooMenu;
	m_ptMsg = point;
	ScreenToClient( &m_ptMsg );
	zooMenu.LoadMenu( ID_POPUP );
```

```
        CMenu* pPopup = zooMenu.GetSubMenu( 0 );
        pPopup->TrackPopupMenu( TPM_LEFTALIGN | TPM_RIGHTBUTTON,
                                point.x,
                                point.y,
                                this);
}
```

عند النقر بالزر الأيمن يتلقى البرنامج الرسالة WM_CONTEXTMENU ثم ينفذ الوظيفة الخاصة بها.

تنشئ الوظيفة OnContextMenu كائنا من CMenu ثم تحمل القائمة ID_POPUP بواسطة الدالة LoadMenu ثم تعرض القائمة بواسطة الوظيفتين GetSubMenu و TrackPopupMenu.

وأخيرا اكتب الوظائف المبينة في القائمة 10-6 الخاصة بكل عنصر من القائمة ID_POPUP

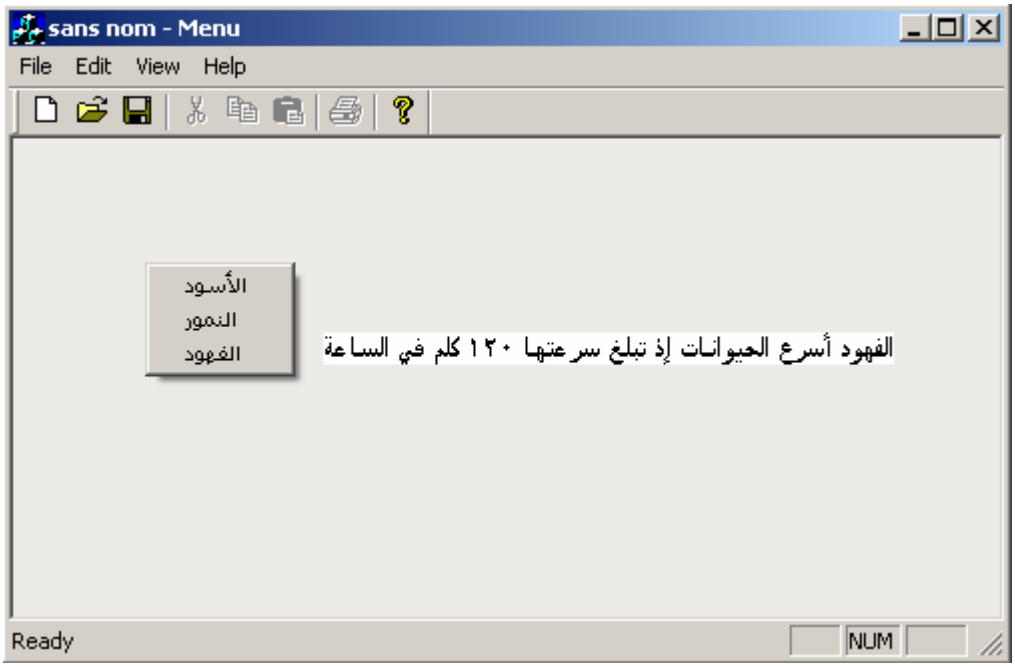
```
void CMenuView::OnLions()
{
        // TODO: Add your command handler code here
        // تخزين النص المعروض في المتغير
        m_szMsg = "الأسد ملك الحيوانات ولكنه لا يفترس حتى يجوع";
        // حث البرنامج على استدعاء دالة الرسم OnDraw
        InvalidateRect( NULL );
}

void CMenuView::OnTigres()
{
        // TODO: Add your command handler code here
        m_szMsg = "النمور جميلة لكنها شرسة";
        InvalidateRect( NULL );

}

void CMenuView::OnPantheres()
{
        // TODO: Add your command handler code here
        m_szMsg = "الفهود أسرع الحيوانات إذ تبلغ سرعتها 120 كلم في الساعة";
        InvalidateRect( NULL );
}
```

بعد تفسير البرنامج Menu.exe وتشغيله ستحصل على شكل مشابه للشكل 10-4:



مفاتيح الاختصار:

مفاتيح الاختصار هي مجموعة من اختصارات لوحة المفاتيح تتيح تنفيذ أوامر معينة بشكل أسهل وأسرع ونجدها غالبا مع القوائم.

المعالج AppWizard ينشئ تلقائيا بعض مفاتيح الاحتصار لبعض الوظائف مثلا:

- المفتاح Ctrl-N للأمر جديد
- المفتاح Ctrl-O للأمر فتح

ومفاتيح الاختصار نجدها مكتوبة على يسار العنصر فيمكنك مثلا في الأمر فتح ضغط المفتاحين Ctrl-O كما لو نقرت بزر الماوس على هذا الأمر.

عرض مورد مفاتيح الاختصار:

مفاتيح الاختصار نوع من الموارد كالقوائم والصور فيمكن لإضافة مورد جديد لمفاتيح الاختصار أو حذفه أو تعديله. لعرض مفاتيح الاختصار التي أنشأها المعالج AppWizard انقر علامة الجمع أمام المجلد Accelerator ثم انقر نقرا مزدوجا على العنصر IDR_MAINFRAME ستعرض على اليمين قائمة بجميع مفاتيح الاختصار الموجودة في برنامجك.

إضافة مفتاح جديد:

لإضافة مفتاح جديد افتح النافذة Accel Properties بالنقر المزدوج على السطر الفارغ في آخر قائمة مفاتيح الاختصار أو بالضغط على المفتاح Inser من لوحة المفاتيح.

والآن سنضيف مفتاح اختصار جديد إلى مشروعنا Menu هذا المفتاح يقوم بنفس العملية التي يقوم بها عنصر القائمة "الأسود" في القائمة الموضعية.

افتح المورد IDR_MAINFRAME واضغط المفتاح Inser ثم أضف مفتاحا جديدا واستعمل القيم الموجودة في هذا الجدول:

Type	Modifiers	Key	ID
VirtKey	Ctrl	L	ID_LIONS

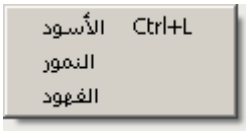
فسر البرنامج ثم شغله بدل أن تنقر عنصر القائمة الأسود اضغط مفتاح الاختصار Ctrl-L سيتم تنفيذ الأمر كما لو تم النقر على العنصر.

تلميح:

يمكنك بيان مفتاح الاختصار أمام عنصر القائمة بتعديل الخاصية Caption للعنصر فمثلا عدل تسمية الأسود إلى :

الأسود \t Ctrl+L

الرمز \t يضيف 8 مسافات فهو مثل الضغط على مفتاح الجدولة ستحصل على مثل الشكل 10-5:


الأسود Ctrl+L
النمور
الفهود

تنبيه : هناك فصول طور التنقيح والإعداد فلهذا تخطيتها وقدمت الكتاب للقارئ حتى لا يطول الانتظار

الفصل الرابع عشر

الأيقونات والمؤشرات

محتويات الفصل:

- **تعريف الأيقونات.**
- **إنشاء أيقونة بواسطة محرر الصور**
- **تعريف المؤشرات**
- **استعمال المؤشرات في برامج ويندوز**

تعريف الأيقونات.

الأيقونة Icon صورة صغيرة توضع على كائن ما من كائنات ويندوز ، ونجد الأيقونات تقريبا في كل برامج ويندوز تجدها على النماذج والأزرار وصفحات الإنترنت.

أنواع الأيقونات:

تستخدم الأيقونات على أربعة أنواع:

- أيقونات كبيرة ذات مقاس 32 * 32 بكسل وتستعمل 16 لونا.
- أيقونات صغيرة ذات مقاس 16 * 16 بكسل
- أيقونات ذات نمط عال من الألوان 256 لونا ومقياسها غالبا 48 * 48 بكسل.
- أيونات ثنائية اللون لا تستخدم سوى اللون الأبيض والأسود ومقياسها 32 * 32 بكسل.

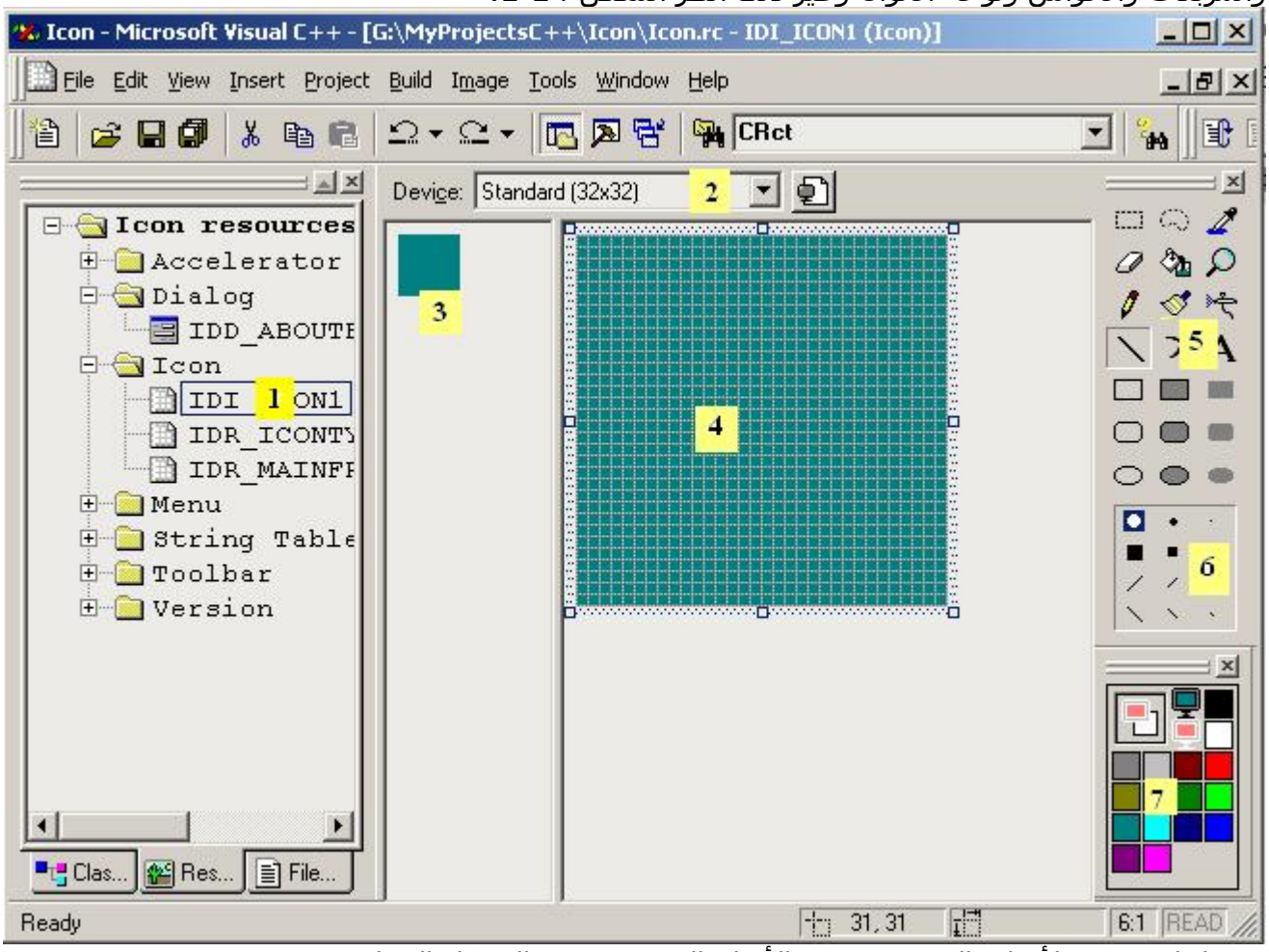
ليست هناك صعوبة في التعامل مع الأيقونات في مشروعاتك بحيث أنك لست بحاجة إلى التعامل مع خلية خاصة بالأيقونات وإنما تحتاج للتعامل مع مقبض Handel يسمى HICON.

إنشاء أيقونة بواسطة محرر الصور

الأيقونات نوع من الموارد مثل القوائم والنماذج تضيفها إلى مشروعك كما تضيف الموارد الأخرى ستجد بعض الأيقونات أضافها المعالج AppWizard تلقائيا إلى مشروعك.

ولإضافة أيقونة جديدة اختر الأمر ...Insert | Resource ومن مربع الحوار Insert Resource اختر العنصر Icon ثم انقر الزر New أو اختر الزر Imports لاستيراد أيقونة جاهزة من قرصك الصلب.

بعد النقر على الزر New سيعرض محرر الصور مع أيقونة فارغة جاهزة للتعديل وستجد أمامك جميع الأداوات الخاصة بالصور كالتي تجدها في برنامج الرسام لوحة أدوات الرسم كالفرشاة والمربعات والأقواس ولوحة الألوان وغير ذلك انظر الشكل 14-1:



وفيما يلي شرح لأدوات المحرر حسب الأرقام الموجودة في الشكل السابق:

1- اسم أو معرف الأيقونة يمكن تغييره من نافذة الخصائص والتي يمكن إظهارها بواسطة المفتاح Alt+Enter
2- مقاس الأيقونة بالبكسل وللعلم فإنه يمكن جعل للأيقونة الواحدة عدة مقاسات بالضغط على New Device Image الذي عن يمين المربع المنسدل كما أنه يمكنك جعل لكل مقاس صورة خاصة به.

لنقم بمثال:

أنشئ أيقونة جديدة وسمها IconTest واحفظها في المجلد :C
ارسم الرقم 32 في المقاس الأول 32 * 32
غير المقاس إلى 16* 16 وارسم الرقم 16
غير المقاس إلى 48 * 48 وارسم الرقم 48
أحفظ الأيقونة ثم استعرضها مستكشف ويندوز
عند تغيير نوع العرض في المستكشف ستلاحظ ما يلي:

- يظهر الرقم 48 في الوضع مصغرات
- يظهر الرقم 32 في الوضع أيقونات
- يظهر الرقم 16 في الوضع قائمة

من هنا تعرف أنك تستطيع أن تجعل لكل مقاس صورة خاصة ،و الصورة إذا كان لها مقاس واحد فإن ويندوز هو الذي سيتولى تغيير مقاساتها بما يتناسب مع نوع العرض في المستكشف.

3- نموذج عن الصورة الفعلية للأيقونة
4- هنا ترسم الأيقونة
5- أدوات الرسم
6- أشكال متنوعة لأدوات الرسم
7- الألوان ويمكنك من خلال هذه اللوحة جعل الأيقونة ذات خلفية معينة أو جعلها شفافة.

ربط أيقونة مع البرنامج:

لا يكلفك تحميل وإظهار أيقونة على البرنامج بعد إنشائها سوى ثلاثة أسطر من الكود .
لتحميل أيقونة في ذاكرة البرنامج وتجهيزها للعرض استعمل الدالة LoadIcon تستعمل هذه الدالة لاسترجاع مقبض الأيقونة.

```
HICON hIcon = AfxGetApp()->LoadIcon( IDI_ICONE1 );
```

الدالة LoadIcon عضو من الدالة AfxGetApp وتستعمل كمؤشر لاسترجاع مقبض الأيقونة.
بعد تحميل الصورة يمكنك عرضها مثلا في منظر يمكنك عرضها بواسطة الدالة OnDraw

```
pDC->DrawIcon( hIcon );
```

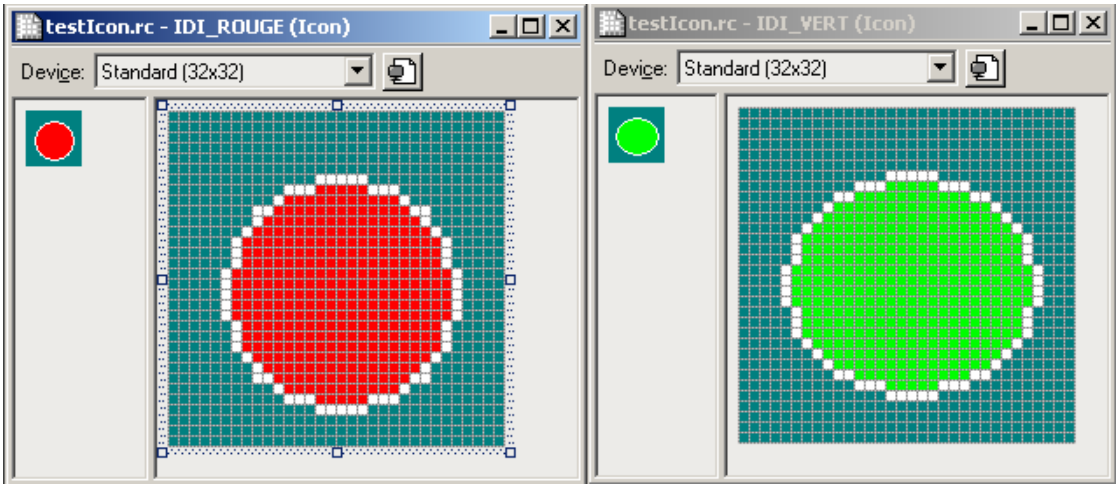
تنبيه: إذا نسيت استدعاء الدالة DestroyIcon فإن المساحة المحجوزة للأيقونة لا تحرر.

عرض أيقونة على زر أوامر:

أغلب ما تستعمل الأيقونات في إظهارها على الأزرار كأن توضع مثلا صورة طابعة على زر طباعة فالأيقونات بهذا الشكل تزيد من جمال البرنامج وتسهل معرفة اوامره.
الدالة SetIcon التابعة للخلية CButton تتيح عرض أيقونة الزر ولكن بعد تحميلها في الذاكرة.

المشروع testIcon:

أنشئ مشروعا معتمد على مربع حوار وسمه testIcon
سننشئ أيقونتين للزرين موافق وإلغاء أيقونة موافق عبارة عن دائرة خضراء واسمها IDI_ROUGE وأيقونة إلغاء عبارة عن دائرة حمراء واسمها IDI_VERT كما هو مبين في الشكل 2-14:



عدّل زري مربع الحوار IDD_TESTICON_DIALOG بوضع علامة صح أمام الخاصية Icon في الشريحة Style من نافذة الخصائص.
بواسطة المعالج Class Wizard أنشئ كائنين من الخلية CButton لكلا الزرين الكائن m_btnOk للزر موافق والكائن m_btnCancel للزر إلغاء كما في الجدول 14-1:

Varible Type	Category	Member VariableName	اسم الأداة
CButton	Control	m_btnOk	IDOK
CButton	Control	m_btnCancel	IDCANCEL

كتابة الكود:

أضف الكود المبين في القائمة 14-1 في الملف testIconDlg.h بعد التعليق:

```
// Implementation
```

هذا الكود يصرح حدث هدم الخلية الذي سنستعمله بعد غلق المربع لتحرير الذاكرة من المساحة التي تحتلها الأيقونتين، ويصرح متغيرين من HICON لتخزين مقبض الأيقونتين.

```
// Implementation
public:
        //تصريح حدث الهدم
        ~CTestIconDlg();
protected:
        HICON       m_hIconOk;
        HICON       m_hIconCancel;
```

أفضل مكان لرسم الأيقونتين على الزرين هو حدث التي التهيئة OnInitDiualog الذي يطلق عند استقبال الرسالة WM_INITDIALOG ابحث عن الملف testIconDlg.cpp ثم اكتب كود القائمة 14-2 مباشرة بعد التعليقTODO

```
        // TODO: Add extra initialization here
        CWinApp*  pApp = AfxGetApp();
        if( pApp != 0 )
        {
                m_hIcon = pApp->LoadIcon( IDI_VERT );
                m_hIconCancel = pApp->LoadIcon( IDI_ROUGE );
```

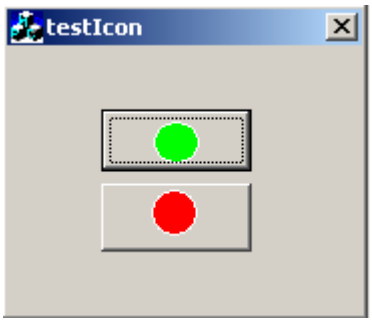
```
    ASSERT( m_hIcon );
        ASSERT( m_hIconCancel );

        m_btnOk.SetIcon( m_hIcon );
        m_btnCancel.SetIcon( m_hIconCancel );
}
```

يحمل هذا الكود الأيقونتين إلى الذاكرة ويخزن مقبضهما في متغيرين ثم يمرر المقبضين إلى الدالة SetIcon لتعرض الأيقونتين على الزرين.
عند إغلاق النموذج يجب تحرير الذاكرة سنفعل ذلك في حدث الهدم الذي صرحناه سابقا اكتب الكود المبين في القائمة 14-3 أسفل الملف testIconDlg.cpp

```
CTestIconDlg::~CTestIconDlg()
{
   DestroyIcon( m_hIcon );
   DestroyIcon( m_hIconCancel );
}
```

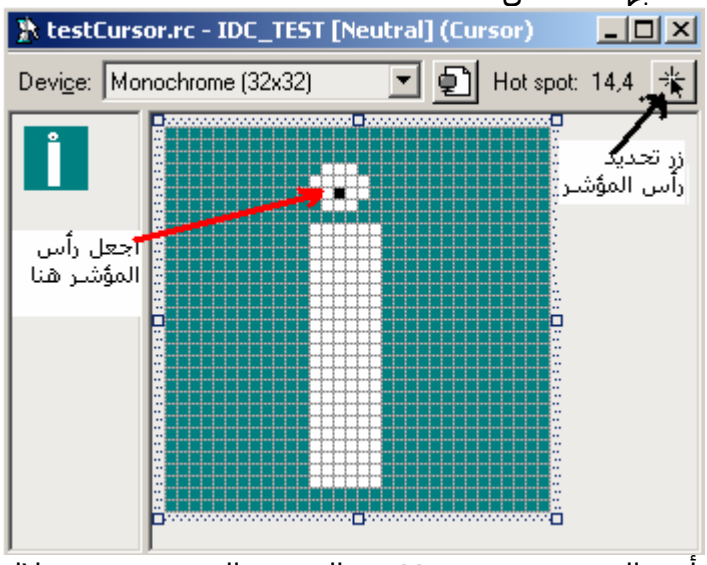
فسر البرنامج ثم شغله سوف يبدو برنامجك مثل الشكل 14-3



تعريف المؤشرات

ما هو المؤشر: المؤشر Cursor صورة صغيرة تتحرك على الشاشة تبين موقع مؤشر الفأرة أو هي صورة مؤشر الفأرة، وهناك مؤشرات عديدة يوفرها نظام التشغيل ويندوز، فأغلب النوافذ تستعمل صورة السهم وعندما تكون النوافذ غير متاحة لإدخالات المستخدم يتحول مؤشر الفأرة إلى ساعة رملية تبين للمستخدم أن عليه الانتظار ريثما ينهي البرنامج عمله، وعندما يحرك مؤشر الفأرة فوق حواف نافذة قابلة للتحجيم فإنه يتغير إلى شكل سهم مزدوج يبين للمستخدم أنه باستطاعته تحجيم النافذة ، وعندما يكون المؤشر فوق مربع نص قابل للكتابة نجد المؤشر يتحول إلى الحرف الكبير I كل هذه المؤشرات وغيرها يوفرها نظام ويندوز.
إن تحويل المؤشر إلى صورة معبرة يعتبر عملية فعالة لتسهيل فهم البرامج وتحسين مظهرها.
استعمال المؤشرات في برامج ويندوز:
أغلب البرامج تكتفي بالمؤشر الاعتيادي الموفر من قبل ويندوز وهو شكل السهم ولكن في حالة ما قد تحتاج إلى تغيير شكل المؤشر إلى صورة تنشئها بنفسك أو تحملها من الإنترنت سنشرح كيف يتم ذلك في بقية الفصل.
أنشئ مشروعا جديد معتمد على مربع حوار وسمه testCursor.
إنشاء مورد مؤشر: المؤشرات نوع من الموارد كالقوائم والصور لذا عليك إنشاؤها أو استادها بواسطة محرر الموارد.

أضف مؤشر جديدا إلى المشروع ولفعل ذلك انقر بالزر الأيمن على الشريحة ResourceView في الإطار WorkSpace ومن القائمة الموضعية اختر Inset تظهر النافذة Insert Resource حدد العنصر Cursor ثم انقر الزر New.
سيتم إنشاء مورد مؤشر جديد جاهز للتعديل غير اسمه IDC_TEST وبواسطة أدوات الرسم أرسم صورة للمؤشر مشابهة للشكل 14-4



تحديد رأس المؤشر: رأس المؤشر Hot spot هو الموضع الذي يتم من خلاله استرجاع إحداثيات المؤشر على الشاشة والموضع الذي يجب أن يكون مثلا فوق الزر ليتم انضغاطه عند النقر، وجميع المؤشرات يجب أن تحتوي على هذا الرأس فرأس المؤشر العادي مثلا هو نهاية السهم من فوق.
محرر الموارد يتيح لك إنشاء رأس المؤشر في أي مكان شئت من الصورة بواسطة الزر Hot spot انظر الشكل السابق، إن عدم تحديد رأس المؤشر يجعل المؤشر يعمل بصورة غير طبيعية،
ولتحديد رأس المؤشر انقر على الزر Hot spot ثم انقر في المكان الذي تريد جعله فيه.
بالنسبة لبرنامجنا اجعل رأس المؤشر في الدائرة العلوية كما هو مبين في الشكل السابق.
تحميل المؤشرات: يرسل نظام التشغيل الرسالة WM_SETCURSOR إلى النافذة عند تحريك المؤشر فوقها، لذلك تعتبر هذه الرسالة مناسبة لتغيير شكل المؤشر، أضف وظيفة تعالج هذه الرسالة إلى الخلية CTestCursorDlg بواسطة المعالج ClassWizard ثم عدل الوظيفة لتبدو كما في القائمة 14-4

```
BOOL CTestCursorDlg::OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT
message)
{
	// TODO: Add your message handler code here and/or call default

	CWinApp* pApp = AfxGetApp();
	HICON	hIconBang = pApp->LoadCursor( IDC_TEST );
	SetCursor( hIconBang );

	return TRUE;
}
```

فسر البرنامج وشغله لاحظ تغير شكل المؤشر إلى الصورة التي أنشأناها ولاحظ أنه لن تستطيع النقر على زر موافق مثلا حتى تكون الدائرة العلوية التي تحمل رأس المؤشر فوق الزر.
تغيير شكل المؤشر بطريقة شرطية: تحتاج أحيانا إلى تغيير شكل المؤشر عندما يكون فوق كائن ما زر مثلا ثم إعادته إلى شكله الطبيعي عندما لا يكون فوقه هنا يجب عليك استعمال التعابير

الشرطية سنعدل مشروعنا السابق لإرجاع المؤشر إلى شكله الأصلي عندما يكون فوق الزر موافق ، الكود اللازم موضح بالقائمة 14-5

```
BOOL CTestCursorDlg::OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT
message)
{
	// TODO: Add your message handler code here and/or call default
	BOOL bReturn;
	CRectrcBtn;
	CPoint	ptCursor;

	CWnd*	pBtn = GetDlgItem( IDOK );
	pBtn->GetWindowRect( rcBtn );
	GetCursorPos( &ptCursor );
	if( rcBtn.PtInRect( ptCursor ) == FALSE )
	{
		CWinApp* pApp = AfxGetApp();
		HICON	hIcon = pApp->LoadCursor( IDC_TEST );
		SetCursor( hIcon );
		bReturn = TRUE;
	}
	else
	{
		bReturn  = CDialog::OnSetCursor( pWnd, nHitTest,message);
	}
	return bReturn;
}
```

مؤشرات ويندوز القياسية:

يمكنك استعمال أحد هذه المؤشرات كما في كود القائمة 14-6:

```
BOOL CTestCursorDlg::OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT
message)
{
		// TODO: Add your message handler code here and/or call default

	CWinApp* pApp = AfxGetApp();
	HICON	hIconBang = pApp->LoadStandardCursor( IDC_HELP );
	SetCursor( hIconBang );

	return TRUE;

}
```

هذا الكود يحول شكل الماوس إلى علامة استفهام

جعل المؤشر على شكل ساعة رملية:

في أغلب الأحيان يتجاهل البرنامج إدخالات المستخدم عند يكون البرنامج طور التهيئة فنجده لا يستجيب لنقرات الماوس ولو كانت واجهته معروضة كما نرى تغير شكل المؤشر إلى ساعة رملية. لذلك ينصح عند تحميل برنامج ما بعرض المؤشر على شكل ساعة رملية لإعلام المستخدم بعدم إمكانية التحكم في البرنامج مادام لم يتم تحميله.

سنضيف في مشروعنا هذا وظيفتين إلى الخلية CAboutDlg:

وظيفة التهيئة OnInitDialog التي تنفذ جوابا للرسالة WM_INITDIALOG

الوظيفة OnTimer التي تنفذ جوابا للرسالة WM_TIMER

أضف هاتين الوظيفتين إلى الخلية CAboutDlg بواسطة المعالج Class Wizard

أضف الكود المبين في القائمة 14-7 إلى الوظيفتين:

```
BOOL CAboutDlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	SetCapture();
	BeginWaitCursor();
	SetTimer( 1,1500,NULL );
	return TRUE;
}


void CAboutDlg::OnTimer(UINT nIDEvent)
{
	ReleaseCapture();
	EndWaitCursor();
	KillTimer( 1 );
}
```

تنشئ الوظيفة OnInitDialog كائن مؤقت بقدر خمس ثواني الدالتان SetCapture و BeginWaitCursor تستعملان لتغيير شكل الفأرة إلى ساعة رملية ضمن الخمس ثواني يتجاهل البرنامج إدخالات المستخدم الوظيفة OnTimer تستعمل بعد انتهاء الوقت لإرجاع المؤشر إلى شكله الأصلي ومحو كائن المؤقت.

أدوات متقدمة

1. أداة الإدخال الرقمي، شريط التقدم، الشريط المنزلق
2. الصور النقطية وقائمة الصور
3. أداة العرض ListView
4. أداة الشجرة TreeView
5. أدوات أكتيف إكس

الفصل الخامس عشر

أداة الإدخال الرقمي، شريط التقدم، الشريط المنزلق

محتويات الفصل:

- أدوات ويندوز المشتركة
- أداة الإدخال الرقمي
- الشريط المنزلق
- شريط التقدم

أدوات ويندوز المشتركة

أدرجت شركة مايكروسوفت لدى ظهور ويندوز 95 ما يسمى بالأدوات المشتركة Common Controls ، هذه الأدوات زادت وظائف مهمة إلى واجهة نظام التشغيل، الجدول التالي يبين هذه الأدوات وخلاياها:

اسم الأداة	اسم الأداة بالإنجليزية	الخلية
أداة الإدخال الرقمي	Spin (Up-down)	CSpinButtonCtrl
شريط التقدم	Progress	CProgressCtrl
الشريط المنزلق	Slider (Trackbar)	CSliderCtrl
قائمة الصور	Image List	CImageList
أداة العرض	List	CListCtrl
الشجرة	Tree	CTreeCtrl

أداة الإدخال الرقمي

أداة الإدخال الرقمي مكونة من زرين متقاربين أحدهما متجه للأعلى والآخر للأسفل وهي تشبه في شكلها شريط التمرير العمودي، تتيح هذه الأداة تعديل قيمة رقمية لأداة أخرى مشتركة معها غالبا تكون مشتركة مع مربع نص Text Box والتي تسمى في هذه الحالة Buddy Control ، تستعمل أداة الإدخال الرقمي غالبا على شكل عمودي ولكن يمكن جعلها بشكل أفقي.
وتستعمل أداة الإدخال الرقمي غالبا عندما يُطلب من المستخدم إدخال قيمة معينة ضمن مجموعة قيم محددة كأن يطالب بإدخال قيمة ما بين 10 و 100 في غير هذه الحالة ليست هناك حاجة لاستعمال هذه الأداة.
ويمكنك إنشاء أداة الإدخال الرقمي وربطها بمربع النص بسوهلة ويسر ستحتاج لاستعمال المعالج Class Wizard.

المشروع Contols:

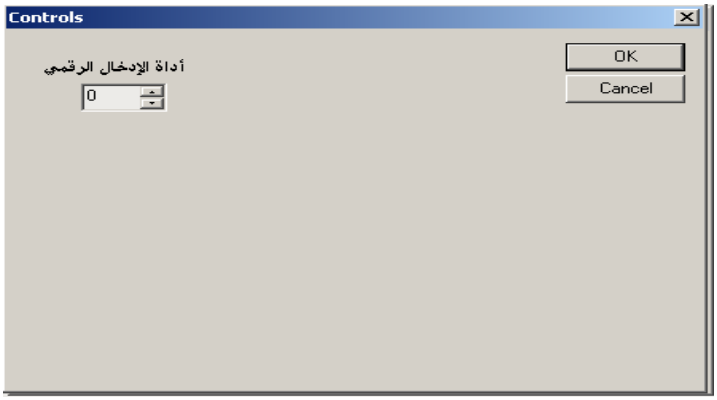
سنستعمل مشروعا واحدا لجميع الأدوات المشتركة لذا أنشئ مشروعا جديدا معتمد على مربع حوار وسمه Controls حدد الخيار Dialog Based من المعالج AppWizard ثم انقر Finish.
افتح مربع الحوار IDD_CONTROLS_DIALOG ثم أضف إليه أداة الإدخال الرقمي من لوح الأدوات Controls يمكنك معرفة أداة الإدخال الرقمي من لوح الأدوات بالاسم Spin ضع هذه الأداة في أعلى النموذج على الجهة اليسرى.

إضافة أداة مشتركة:

أسهل طريقة لإضافة أداة مشتركة مع أداة الإدخال الرقمي كما يلي:

1. أضف مربع نص إلى النافذة وضعه قرب أداة الإدخال الرقمي على اليسار
2. أعط معرف لمربع النص مثل IDC_EDIT
3. اضبط ترتيب الجدولة لمربع النص بحيث يكون قبل أداة الإدخال الرقمي.
4. تأكد من عدم اختيار الخاصة Tab Stop لأداة الإدخال الرقمي.
5. اختر قيمة من قيم المحاذاة Alignment لأداة الإدخال الرقمي إذا كنت قد جعلت الأداة عن يمين مربع النص اختر القيمة Right وإن كنت قد جعلتها على يسار مربع النص اختر القيمة Left.
6. ضع علامة صح للخاصيتين Auto Buddy و Set Buddy التابعتين لأداة الإدخال الرقمي في نافذة الخصائص.

الشكل 15- 1 يبين كيف يبدو البرنامج بعد التشغيل لاحظ أن أداة الإدخال الرقمي قد أخذت موضعا حسنا داخل مربع النص:



ولضبط أو قراءة قيمة أداة الإدخال الرقمي استعمل المعالج Class Wizard لربط كائن CEdit بمربع النص.

الخلية CSpinButtonCtrl:

تتيح الخلية CSpinButtonCtrl إدارة أداة الإدخال الرقمي استعمل المعالج Class Wizard لربط كائن من هذه الخلية مع أداة الإدخال الرقمي IDC_SPIN الموجودة على النافذة وأعط له القيم المبينة في الجدول 15-2:

اسم الأداة	Member VariableName	Category	Varible Type
IDC_SPIN	m_spin	Control	CSpinButtonCtrl

لعلك لاحظت عند تشغيل البرنامج السابق أن أداة الإدخال الرقمي ترفع قيمة مربع النص عند الضغط على الزر السفلي وتنقص من قيمته عند الضغط على الزر العلوي إنها تعمل مثل عرض التنقل بين الصفحات في مستند وورد مثلا إذا نقرت شريط التمرير من الزر العلوي انتقلت إلى الصفحات الأولى والعكس ، ولكن يمكنك عكس الزرين لأداة الإدخال الرقمي:

لفعل ذلك استعمل الدالة SetRange التابعة للخلية CSpinButtonCtrl تحتاج هذه الدالة لوسيطين القيمة الدنيا والقيمة العليا:

```
m_spin.SetRange( 0, 100 );
```

هذا الكود يحدد القيمة الدنيا والعليا لأداة الإدخال الرقمي يمكنك جعل هذا الكود في حدث تهيئة النموذج InInitDialog بعد تفسير وتشغيل البرنامج نجد أن الزرين يعملان بالعكس الزر الأعلى يزيد القيمة والزر الأسفل ينقصها

إليك في الجدول 15-3 أهم دوال أداة الإدخال الرقمي:

الدالة	الوصف
SetAccel	تظبط سرعة التمرير
GetAccel	تقرأ سرعة التمرير
SetBase	تظبط الرقم القاعدي بالنظام السداسي
GetBase	تقرأ الرقم القاعدي بالنظام السداسي
SetPos	تظبط القيمة الحالية
GetPos	تقرأ القيمة الحالية

الشريط المنزلق

ما هو الشريط المنزلق:إذا كانت الأشرطة الرسومية الموجودة في برنامجك تعجبك يمكنك استعمال الشريط المنزلق Slider لتلقي إدخال، وإظهار مدى تقدم مهمة ما، أداة الشريط المنزلق هي مكون واجهة يحتوي على وتد متحرك وعلامات تجزئة اختيارية.

يمكنك تحريك الوتد ضمن حدود الشريط المنزلق عن طريق سحبه أو نقر علامات التجزئة أو استعمال مفاتيح الأسهم في لوحة المفاتيح، استعمال الشريط المنزلق أداة مفيدة للحصول على إدخال بغية ضبط هوامش المستند في برنامج معالجة نصوص، أو للتحكم بخصائص الوسائط

المتعددة، أو للتمرير في إطار ما، أو لتنفيذ مهام أخرى تتطلب تحركا نسبيا، قد تجد هذه الأداة مفيدة أيضا لكتابة قيم رقمية أو لضبط الألوان.
إضافة الشرط المنزلق: لإضافة الشريط المنزلق إلى النموذج انقر الزر Slider في لوح الأدوات Control ثم ارسم الأداة على نموذجك.
أضف شريطا منزلقا إلى النموذج وسمه IDC_SLIDER وضع علامة صح على خصائصه التالية Tick Marks و Auto Ticks و Enable Selection .
قبل أن نتمرن على استعمال الشريط المنزلق سننتقل إلى أداة شريط التقدم ثم ننشئ إصدارا محسنا من مشروعنا Controls حيث سنضيف له شريطا منزلقا وشريط تقدم يعملان بطريقة مشتركة.

شريط التقدم

ما هو شريط التقدم:صبح دائما عند عملنا مع الكمبيوتر هو **شريط التقدم** ، وهو مؤشر حالة أفقي يبين لنا رسوميا الوقت الذي ستستغرقه العملية التي نقوم بها يمكنك رؤية نموذج منه عند عملية نسخ ملف في مستكشف ويندوز.
لا يشير شريط التقدم دائما إلى عدد الدقائق او الثواني التي سيستغرقها الكمبيوتر لتنفيذ مهمة ما، لكنه يعطي عرضا رسوميا كافيا لإقناع المستخدم أن معاملته ما زالت جارية وأن الكمبيوتر لا يزال يعمل. من وجهة نظر علم النفس، يخفف شريط التقدم من الوتر والشد العصبي الذي يرافق عادة انتظار ظهور نتائج عمليات،
إكمال المشروع Controls:
أضف شريط تقدم إلى النموذج لفعل ذلك انقر على الزر Progress في لوحة الأدوات Controls ثم ارسم الأداة على النموذج غير اسم الشريط إلى IDC_PROGRESS.
سنستعمل الشريط المنزلق للتحكم في قيمة شريط التقدم صعودا ونزولا لفعل ذلك اتبع ما يلي:

- بواسطة المعالج Class Wizard أضف إلى الخلية CControlsDlg كائنين واحد مشترك مع الشريط المنزلق والآخر مع شريط التقدم مستعملا في ذلك القيم المبينة في الجدول 14-7

اسم الأداة	Member VariableName	Category	Varible Type
IDC_SLIDER	m_slider	Control	CSliderCtrl
IDC_PROGRESS	m_progress	Control	CProgressCtrl

- في حدث تهيئة النموذج OnInitDialog هيئ القيمة الدنيا والقيمة العليا لكلتا الأداتين، كلتا الأداتين تملكان الدالة SetRang التي تتيح ضبط قيمتيهما الدنيا والعليا، لفعل ذلك أضف كود القائمة 15-2 إلى الدالة OnInitDialog التابعة للخلية CControlDlg تحت التعليق //TODO

```
// TODO: Add extra initialization here
m_spin.SetRange( 0,100 );
m_slider.SetRange( 0,100 );
m_slider.SetTicFreq( 10 );
m_progress.SetRange( 0,100 );
```

- بقي لنا الآن الكود الذي يغير قيمة شريط التقدم بحسب قيمة الشريط المنزلق بعد تحريكه.

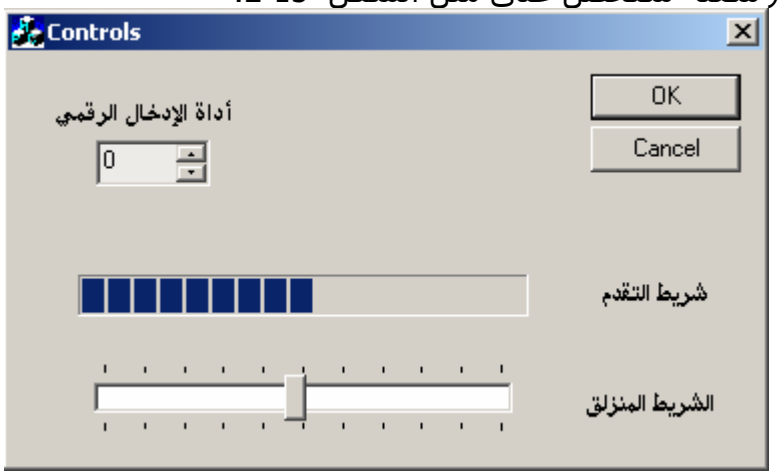
اعلم أنه عند تحريك أي شريط إزاحة على النموذج ترسل رسالة WM_HSCROLL إذا كان الشريط أفقيا أوالرسالة WM_SCROLL إذا كان الشريط عموديا.
وبما أن شريط الانزلاق الموجود على نموذجنا أفقيا فإن النموذج يستقبل الرسالة WM_HSCROLL عند تحريك وتده إلى اليمين أو إلى اليسار/
استعمل المعالج Class Wizard لإضافة وظيفة تعالج الرسالة WM_HSCROLL إلى الخلية CControlsDlg.

أضف الكود الموضح في القائمة 15-3 إلى هذه الوظيفة:

```
void CControlsDlg::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
	//قراءة قيمة شريط الانزلاق وتخزينها في متغير
	int nSliderPos = m_slider.GetPos();
	//ضبط قيمة شريط التقدم على قيمة شريط الانزلاق
	m_progress.SetPos( nSliderPos );
}
```

ينفذ هذا الكود كل مرة يتم فيها تحريك وتد شريط الانزلاق لتضبط قيمة شريط التقدم بنفس قيمة شريط الانزلاق.

فسر برنامجك ثم شغله ستحصل على مثل الشكل 15-2:



الفصل السادس عشر

الصور النقطية وقائمة الصور

محتويات الفصل:

- ماهي الصورة النقطية Bitmap
- ماهي قائمة الصور List Images
- استعمال قائمة الصور

ماهي الصورة النقطية Bitmap

الصورة النقطية كائن رسومي يمكن استعماله لتخزين وتحميل وعرض صور مختلفة في برامج ويندوز ، وتنتهي الصور النقطية غالبا بالامتداد bmp.وتتيح لك الخلية CBitmap التعامل والتلاعب بالصور النقطية في برامج ويندوز.

إنشاء صورة نقطية بواسطة VC++:

يوفر برنامج VC++ محررا للصور مشابه إلى حد كبير برنامج الرسام المشمول مع ويندوز، يمكنك بواسطة هذا المحرر إنشاء الصور النقطية وتعديلها وكذا حفظها في ملف خارجي يحتوي هذا المحرر على كل الأدوات والخصائص والألوان التي تحتاجها.

في هذا المقطع من الفصل سنفتح هذا المحرر وننشئ صورة نقطية ونعرضها على البرنامج.

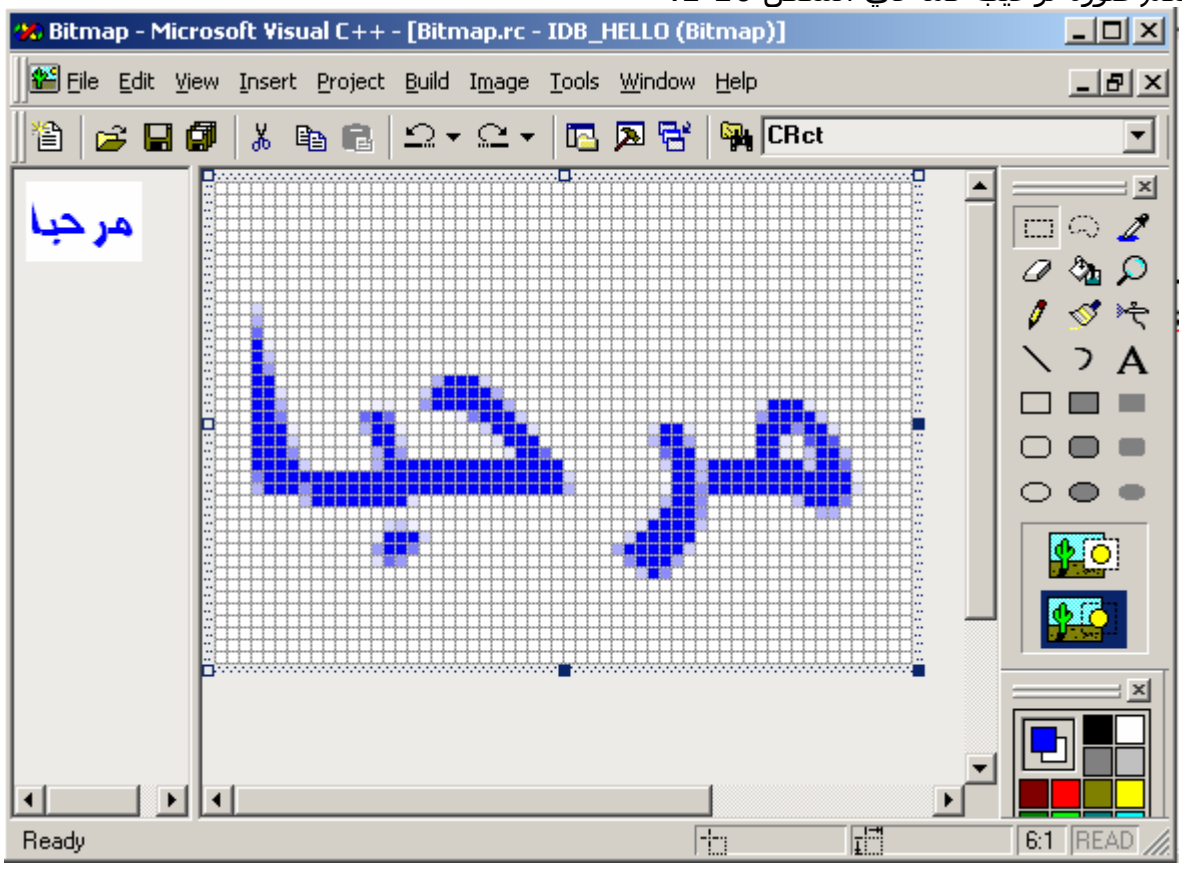
أنشئ مشروعا جديدا من نوع SDI وسمه Bitmap.

إدراج صورة نقطية جديدة اختر الأمر

حدد الأمر Insert | Resource... ومن مربع الحوار Insert Resource اختر العنصر Bitmap ثم انقر الزر New

غير اسم الصورة إلى IDB_HELLO

ارسم صورة ترحيب كما في الشكل 16-1:



يمكنك كتابة النص هلى الصورة محدد اسم وحجم الخط بواسطة الزر A الموجود على صندوق أدوات المحرر ، كما يمكنك تحجيم الصورة بسحب حدودها الجانبية.

تحميل وعرض الصورة النقطية:

افتح الملف BitmapView.cpp وجِدْ الوظيفة CBitmapView::OnDraw استبدل الكود الموجود في ها بكود القائمة 16-1:

```
void CBitmapView::OnDraw(CDC* pDC)
{
	CBitmap bmpHello;
	bmpHello.LoadBitmap( IDB_HELLO );
```

```
    BITMAP      bm;
    bmpHello.GetObject( sizeof(BITMAP), &bm );

    CDC         dcMem;
    dcMem.CreateCompatibleDC( pDC );

    CBitmap*    pbmpOld = dcMem.SelectObject( &bmpHello );
    pDC->BitBlt( 10,10,bm.bmWidth,bm.bmHeight,
                 &dcMem,0,0,SRCCOPY );

    dcMem.SelectObject( pbmpOld );
}
```

أهم دالة في هذا الكود الدالة LoadPitmap التي تحمل الصورة.

ماهي قائمة الصور List Images

عندما يحتاج برنامجك إلى التعامل مع عدد من الصور يستحسن استعمال هذه الأداة الرائعة، هذه الأداة يمكنها تجزين مجموعة من الصور يمكن أن تستعملها أدوات تجكم أخرى وغالبا ما تستعمل مع أداتي العرض ListView والشجرة TreeView ، يمكننا تشبيه هذه الأداة بالشريط الفوتوغرافي الذي يضم صورا عديدة مرتبة، هذه الأداة لا توجد ضمن أدوات التحكم في لوحة الأدوات Controls وإنما تنشئ برمجيا وهي أيضا لا تظهر في واجهة البرنامج، ومن مزايا هذه الأداة أنها تتيح عرض الصور بخلفية شفافة، وهي تخزن صورها مرتبة وتعطي لكل صورة رقما فريدا يمكن الوصول إلى هذه الصورة من خلاله.

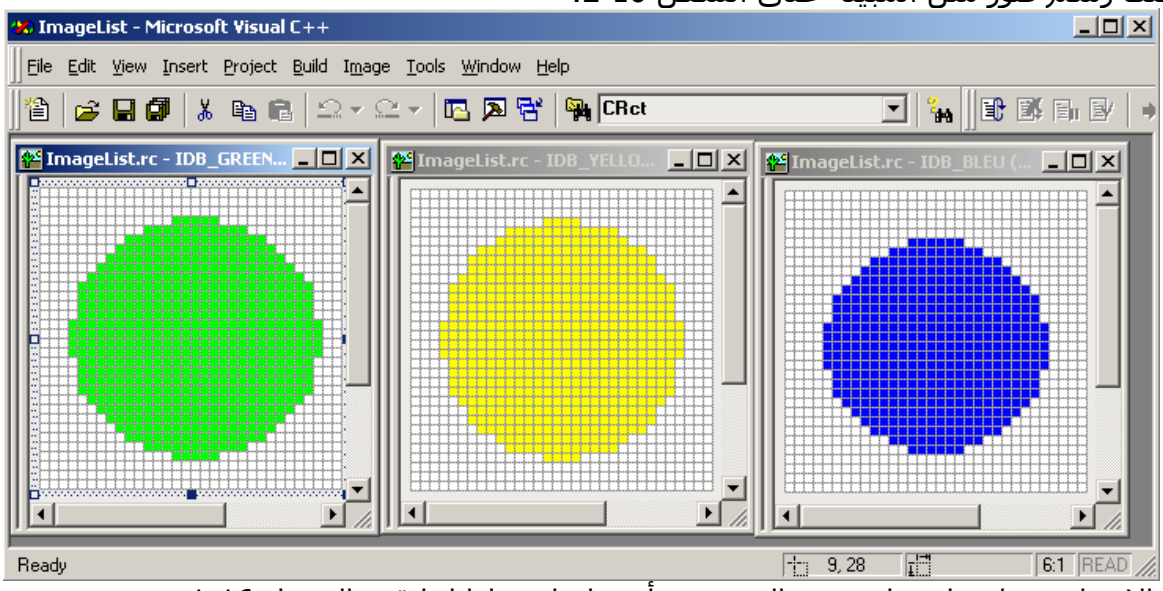
إنشاء قائمة الصور:

أنشئ مشروعا جديدا من نوع SDI وسمه ImageList ، سننشئ قائمة صور لتخزين ثلاث صور ثم نقوم بعرضها على منظر البرنامج عن طريق الوظيفة OnDraw .

الخطوة الأولى لإنشاء قائمة الصور هي إنشاء الصور نفسِها يتم بذلك بواسطة محرر الصور الذي استخدمناه قبل قليل.

افتح محرر الصور وأنشئ أو استرد ثلاث صور واجعل مقاسها 32 * 32 .

يمكنك رسم صور مثل المبينة على الشكل 16-2:



بعد الانتهاء من إنشاء موارد هذه الصور غير أسمائها معطيا لها قيم الجدول 16-1:

الاسم	الصورة

IDB_BLUE	الصورة ذات اللون الأزرق
IDB_YELLOW	الصورة ذات اللون الأصفر
IDB_GREEN	الصورة ذات اللون الأخضر

إن عملية تخزين صورة داخل قائمة الصور تتم على ثلاث خطوات:

1. تحميل الصورة
2. إنشاء فهرس جديد داخل قائمة الصور وإضافة الصورة لها
3. حذف الكائن Bitmap الذي حملت به الصورة لتحرير الذاكرة

بما أن الصور الثلاثة سيتم تحميلها وعرضها بصورة مشابهة يكفي أن ننشئ وظيفة واحدة نستدعيها لكل صورة دون أن نحتاج لتكرير كثير من الأسطر.
أضف الوظيفة الجديدة BitmapToList إلى الخلية CImageListView ، ضع جسم الوظيفة المبين في القائمة 16-2 في الملف ImageList.Cpp:

```
BOOL CImageListView::BitmapToList( UINT nIDResource )
{
  BOOL bReturn;
  CBitmap bmp;

  bReturn = bmp.LoadBitmap( nIDResource );
  if( bReturn != FALSE )
  {
        int nReturn = m_imageList.Add( &bmp,RGB(255,255,255) );
        bmp.DeleteObject();
  }
  return bReturn;
}
```

تقوم هذه الوظيفة بإنشاء كائن CBitmap لتحميل الصورة ثم تضيف الصورة بواسطة الطريقة Add التابعة لقائمة الصور وأخيرا تمحو الكائن CBitmap لتحرير الذاكرة.
الطريقة Add تأخذ وسيطين الأول عنوان الصورة المحملة في الذاكرة والثاني أنواع الوان المستعملة.
بعد ذلك أضف تصريح الوظيفة BitmapToList وتصريح المتغير العضو m_imageList الذي يمثل أداة قائمة الصور في الملف ImageListView.h بعد التعليق // Implementation كما هو موضح في القائمة 16-3

```
// Implementation
protected:
  BOOL  BitmapToList( UINT nResourceID );
  CImageList   m_imageList;
```

سنكتب الكود الذي ينشئ قائمة الصور ويستدعي الوظيفة BitmapToList أفضل مكان لهذا الكود هو حدث بناء الخلية CImageListView أكتب الكود 16- 4 في حدث البناء التابع للخلية CImageListView كما قلنا سابقا حدث البناء يكون اسمه بنفس اسم الخلية وبدون قيمة معادة أو وسيطات وكذلك حدث الهدم إلا انه يتبع بالرمز ~:

```
CImageListView::CImageListView()
{
  // TODO: add construction code here
  m_imageList.Create( 32,32,TRUE,3,1 );
```

```
    BitmapToList( IDB_YELLOW );
    BitmapToList( IDB_GREEN );
    BitmapToList( IDB_BLEU );
}
```

تنشئ قائمة الصور في هذا الكود بواسطة الدالة Create التابعة للخلية CImageList تأخذ هذه الدالة خمس وسيطات:

1- مقياس ارتفاع الصور هنا 32
2- مقياس عرض الصور هنا 32 أيضا
3- كون خلفية الصورة شفافة أم لا هنا نعم
4- عدد الصور المخزنة في القائمة هنا 3
5- عدد الصور الممكن إضافتها إلى القائمة بعد إنشائها هنا 1

عرض صور القائمة:

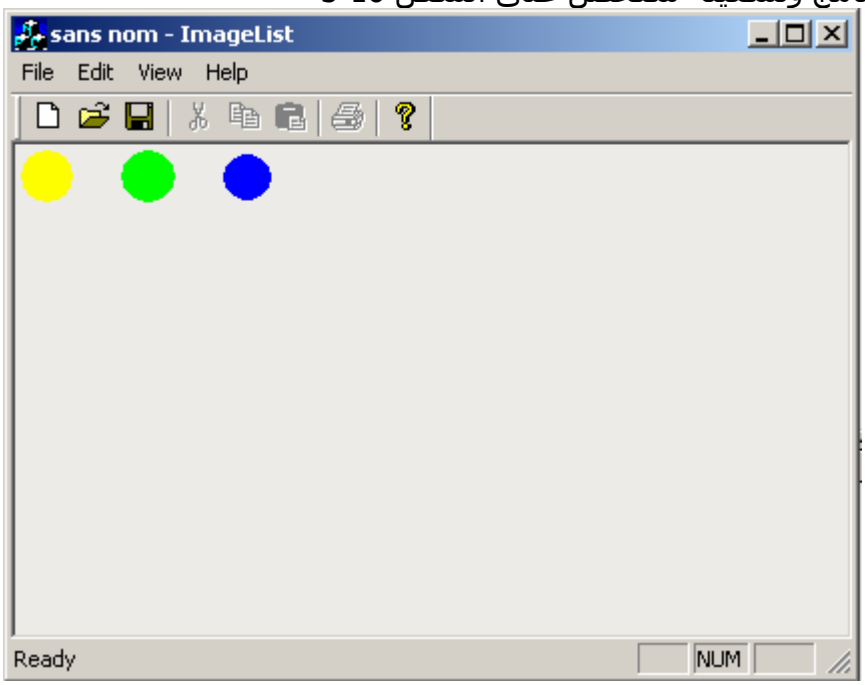
يمكن عرض صور القائمة بواسطة الدالة CListImage::Draw ولعرض صور القائمة أضف 5-16 إلى الدالة OnDraw في الملف ImageList.cpp:

```
void CImageListView::OnDraw(CDC* pDC)
{
  CPoint ptImage( 0,0 );
  for( int nImage = 0;nImage < 3 ; nImage++ )
  {
        m_imageList.Draw( pDC, nImage ,ptImage , ILD_NORMAL );
        ptImage.x += 50;
  }
}
```

تحتاج الدالة Draw إلى أربع وسيطات:

1- كائن من نوع CDC يمثل واجهة الرسم
2- رقم الصورة داخل قائمة الصور
3- المكان الذي سترسم في الصورة من المنظر استعملنا هنا إحداثيات كائن CPoint
4- هنا يمكن وضع إحدى ثمانية قيم تحدد أنواعا مختلفة لرسم الأداة اخترنا هنا الرسم العادي

بعد تفسير البرنامج وتشغيله ستحصل على الشكل 3-16



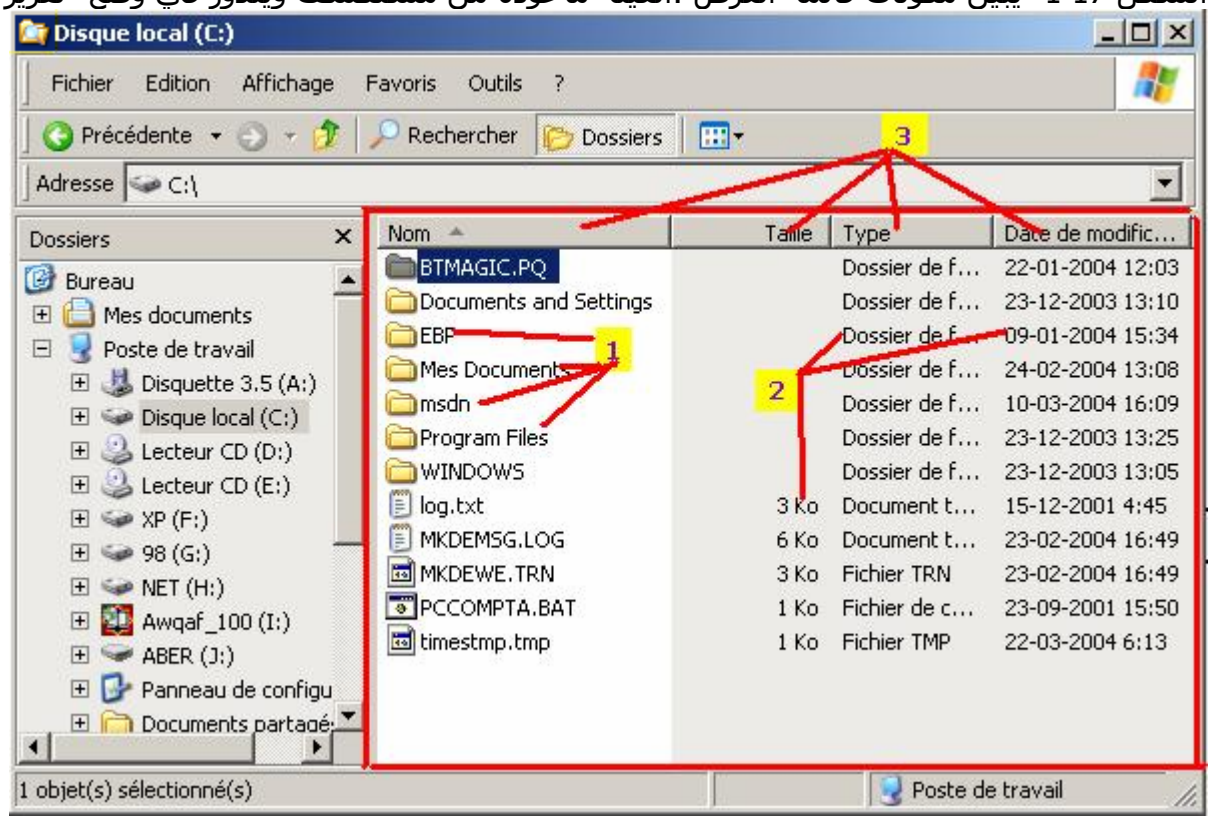
قائمة العرض مثل القائمة العادية ListBox إلا أنها تفوقها في قدرتها على عرض العناصر على شكل أيقونات وقدرتها على عرض البيانات باستعمال أنواع معاينة مختلفة تتمثل في ما يلي:

- رموز كبيرة بمقاس 32 * 32
- رموز صغيرة بمقاس 16 * 16
- لائحة مثل ListBox
- تقرير ، معاينة العناصر مع أيقوناتها مع عناصر إضافية لكل عنصر على شكل شبكة.

يمكنك رؤية عينة من هذه الأداة في مستكشف ويندوز فيمكنك عند معاينة محتوى قرص معاينة الملفات على شكل أيقونات صغيرة أو كبيرة أو تقرير وفي وضع التقرير نجد معلومات إضافية عن كل ملف نوعه وحجمه وتاريخ تعديله وغير ذلك.

مكونات قائمة العرض:

الشكل 17 1- يبين مكونات قائمة العرض ،العينة مأخوذة من مستكشف ويندوز في وضع "تقرير":



الأداة المحاطة بالمربع الأحمر هي قائمة العرض ،ويمكنك معاينة أنواع المعاينة الأخرى من خلال القائمة عرض ،وإليك تفصيل مكوناتها حسب الأرقم:

1. العناصر الرئيسية للأداة كل عنصر يحمل صورة ونصا هذه العناصر تظهر في جميع أنواع المعاينة للقائمة.
2. العناصر الفرعية للأداة وكل عنصر منها يعطي تفصيلا للعنصر الرئيسي الموجود في صفّه لا تظهر هذه العناصر إلا في وضع "تقرير".
3. أعمدة القائمة كل عمود يحمل نصا يبين نوع العناصر المدرجة فيه، الأعمدة أيضا لا تظهر إلا في وضع "تقرير".

تلميح:

لكي يمكن لقائمة العرض عرض صور مع العناصر يجب ربطها بقائمة صور ImageList هذه القائمة تخزن الصور المراد إظهارها في قائمة العرض ولمزيد من التفاصيل عن قائمة الصور راجع الفصل السابق:

نظرة عامة على هذه المكونات:

- **العناصر الرئيسية:** كل عنصر رئيسي في قائمة العرض يتم إدراجه عن طريق إنشاء متغير مشتق من التركيب **LV_ITEM** هذا التركيب يضم مجموعة من الأعضاء تحدد قيم العنصر المراد إضافته كاسمه ورقمه وصورته وغير ذلك، وبعد تحديد المعلومات الخاصة بالعنصر يمكن إضافته إلى القائمة كما سنوضح لك في ما بعد.
- **الأعمدة:** اعلم أنك لا تحتاج إلى إضافة الأعمدة إلا عندما تنوي عرض العناصر في وضع تقرير ،وإضافة العمدة إلى قائمة العرض تتم عبر خطوتين الأولى تهيئة متغير مشتق من التركيب **LV_COLUMN** هذا التركيب يضم مجموعة من الأعضاء تحدد قيم العمود المراد إضافته كاسمه ورقمه وصورته وغير ذلك، والخطوة الثانية استعمال الدالة الخاصة بإنشاء الأعمدة كما سنوضح ذلك.
- **العناصر الفرعية:**هذه العاصر كالأعمدة لا تحتاج إلى إنشائها إلا في وضع التقرير .

المشروع ListEx:

أنشئ مشروعا معتمدا على مربع حوار Dialog Based وسمه ListEx سنستعمل في هذا المشروع قائمة العرض وأزرار خيار تتيح معاينة أنواع المعاينة المختلفة لأداة العرض.
أضف إلى النموذج قائمة عرض لفعل ذلك انقر الزر List Control في لوح الأدوات ثم ارسم الأداة على النموذج بعد ذلك أضف أربعة أزرار خيار واجعلهم في إطار تحت عنوان معاينة انظر الشكل 2-17

سمّ أداة العرض وأزرار الخيار بالأسماء المبينة في الجدول:

الأداة	الاسم	العنوان
زر خيار	ID_RADIO_LARGE	رموز &كبيرة
زر خيار	ID_RADIO_SMALL	رموز &صغيرة
زر خيار	ID_RADIO_LIST	&قائمة
زر خيار	ID_RADIO_REPORT	تقرير
قائمة عرض	IDC_LIST	لاشيء

استعمل المعالج Class Wizard لإضافة متغير عضو خاص بقائمة العرض استعمل القيم الموجودة في الجدول:

اسم الأداة	Member VariableName	Category	Varible Type

IDC_LIST	m_listCtrl	Control	CListCtrl

بعد ذلك أنشئ بواسطة المعالج Class Wizard الوظائف التي تعالج الرسالة BN_CLICKED الخاصة بأزرار الخيار اقبل أسماء الوظائف المقترحة من قبل المعالج الوظائف مبينة في الجدول

اسم الزر	اسم الخلية	الرسالة	
ID_RADIO_LARGE		رموز &كبيرة	
ID_RADIO_SMALL		رموز &صغيرة	
ID_RADIO_LIST		&قائمة	
ID_RADIO_REPORT		تقرير	
IDC_LIST		لاشيء	

التالي:

ربط قائمة الصور مع قائمة العرض :

الصور التي تعرض بجانب كل عنصر في قائمة العرض تكون مخزنة في قائمة صور ImageList يتم الربط بينهما كما سترى لاحقا، لإضافة قائمة صور إلى قائمة العرض استعمل الدالة SetImageList

```
m_listCtrl.SetImageList( &m_imageSmall, LVSIL_SMAL );
```

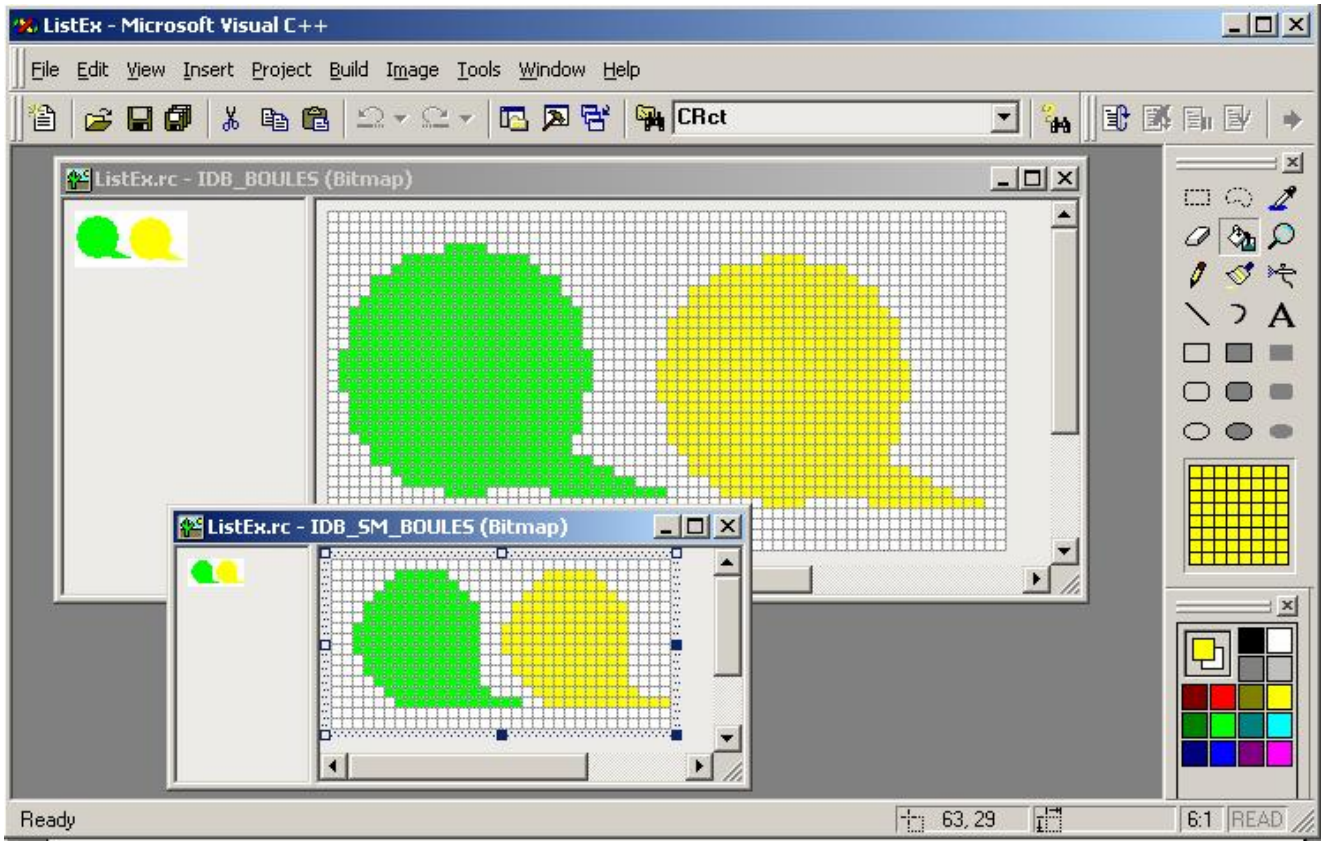
تستخدم هذه الدالة وسيطين عنوان قائمة الصور وثابت من ثلاثة ثوابت يحدد نوع الصور المخزنة في قائمة الصور،

إنشاء قائمة الصور:

سننشئ صورتين لمشروعنا الصورة الأولى مثل الأيقونات الصغيرة والأخرى الإيقونات الكبيرة الجدول التالي يبين اسم الصورتين وبعض القيم الأخرى:

اسم الصورة	العرض	الارتفاع	الخلفية
IDB_BOULES	64	32	بيضاء
IDB_SM_BOULES	32	16	بيضاء

شكل الصورتين:



يمكنك إظهار نافذة الخصائص للصورة عن طريق المفتاح Ctrl+Alt

إضافة عناصر:

يمكن إنشاء عناصر للقائمة بواسطة التركيب LV_ITEM هذا التركيب يمثل كل عنصر في الأداة فيمكن استعماله لإضافة عنصر أو حذفه أو قراءته.

وبعد تحديد تصريح متغير العنصر وإسناد القيم اللازمة له يمكن إضافته إلى القائمة عن طريق الدالة InsertItem هذه الدالة يمرر إليها عنوان المتغير الذي يمثل العنصر.

تحديد معلومات وضع التقرير:

وضع التقرير مصمم فمعاينة معلومات إضافية عن كل عنصر فهو عبارة عن عناصر فرعية للعناصر الرئيسية فمثلا في مستكشف الويندوز يمكنك معاينة الملفات في وضع التقرير لترى معلومات أخرى عن كل ملف حجمه ونوعه إلخ .

فالعنصر الرئيسي هو اسم الملف وهو يظهر في جميع أنواع المعاينة الأخرى والعناصر الأخرى عناصر فرعية ولا ترى إلا في وضع التقرير.

وقبل إضافة العناصر الفرعية يجب عليك إنشاء العمد التي تضمها كل عمود يحتوي على نص يوضح نوع المعلومات التي يحويها.

إضافة الأعمدة للقائمة تمر عبر مرحلتين:

- تهيئة متغير مشتق من التركيب LV_COLUMN هذا المتغير يمثل العمود المراد إضافته.
- إضافة العمود بواسطة الدالة InsertColumn

تعديل الخلية CListExDlg:

لكي يمكننا معاينة قائمة العرض في وضع رموز كبيرة وصغيرة يجب علينا إنشاء قائمتي صور واحدة للصور الكبيرة والأخرى للصغيرة.

سننشئ وأيضا وظيفة باسم SetListView تقوم بتغيير نمط العرض.

أضف الكود التالي في الملف ListExDlg.h بعد التعليق // Implementation هذا الكود يحتوي على تصريح الدالة SetListView وقائمتي الصور.

```
// Implementation
```

```
protected:
        void SetListView( DWORD dwView );
        CImageList    m_imageLarge;
        CImageList    m_imageSmall;
```

أضف الكود التالي إلى الدالة OnInitDialog التابعة الخلية CListExDlg في الملف ListExDlg.cpp
وبالظبط بعد التعليق //TODO

```
// TODO: Add extra initialization here
m_imageLarge.Create( IDB_BOULES, 32 , 1 ,RGB(255,255,255) );
m_imageSmall.Create( IDB_SM_BOULES, 16 , 1 ,RGB(255,255,255) );

m_listCtrl.SetImageList( &m_imageLarge, LVSIL_NORMAL );
m_listCtrl.SetImageList( &m_imageSmall, LVSIL_SMALL );

LV_COLUMN  listColumn;
LV_ITEM              listItem;

char*  arColonnes[3] = { "أسماك" ، "مشروبات" ، "حيوانات"};

listColumn.mask = LVCF_FMT | LVCF_WIDTH | LVCF_TEXT | LVCF_SUBITEM ;
listColumn.fmt = LVCFMT_LEFT;
listColumn.cx = 60;

for( int nColonne = 0;nColonne <3;nColonne++)
{
        listColumn.iSubItem = nColonne;
        listColumn.pszText = arColonnes[nColonne];
        m_listCtrl.InsertColumn( nColonne,&listColumn );
}
listItem.mask = LVIF_TEXT | LVIF_IMAGE;
listItem.iSubItem = 0;
char*  arAnimal[3]={"الفهد"،"الأسد"،"النمر"};
char*  arBoisson[3]={"عصير"،"لبن"،"ماء"};
char*  arNourriture[3]={"قرش"،"عنبر"،"سلمون"};

for( int nItem = 0; nItem <3;nItem++ )
{
        listItem.iItem = nItem;
        listItem.pszText = arAnimal[nItem];
        listItem.iImage = nItem % 2;
        m_listCtrl.InsertItem( &listItem );
        m_listCtrl.SetItemText( nItem,1,arBoisson[nItem]);
        m_listCtrl.SetItemText( nItem , 2,arNourriture[nItem] );
}
```

هذا الكود ينشئ قائمتي صور لأداة العرض ثم ينشئ أعمدة الأداة وبعد ذلك يدرج العناصر الثلاثة .
الدالة SetItemText تستعمل لإضافة عناصر فرعية

تغيير نوع العرض للقائمة:

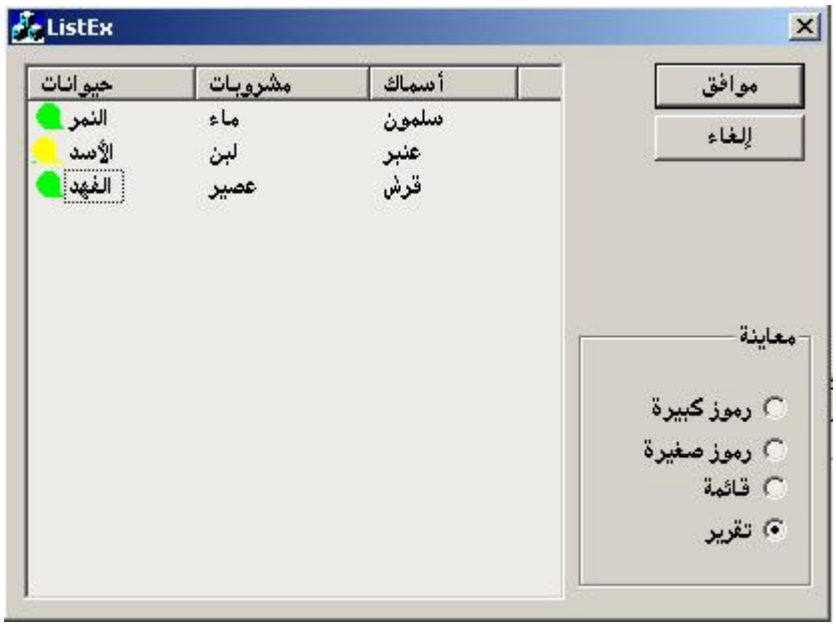
أضف تعريف الدالة التالي في آخر الملف ListExDlg.cpp

```
void CListExDlg::SetListView( DWORD dwNewStyle )
{
	DWORD	dwOldStyle;
	HWND hWndList = m_listCtrl.GetSafeHwnd();

	dwOldStyle = GetWindowLong( hWndList, GWL_STYLE );

	if( (dwOldStyle & LVS_TYPEMASK) != dwNewStyle )
	{
		dwOldStyle &= ~LVS_TYPEMASK;
		dwNewStyle |= dwOldStyle;
		SetWindowLong( hWndList, GWL_STYLE, dwNewStyle );
	}
}
```

بعد ذلك أضف الكود التالي للوظائف الخاصة بازرار الخيار ، النقر على زر من أزرار الخيار يؤدي إلى استدعاء الدالة SetListView الدالة تستقبل قيمة تبين نوع العرض المطلوب
بعد تفسيرك للبرنامج سيبدو مشابها لهذا الشكل.



فائدة:
يمكنك استعمال الكود التالي لاسترجاع العناصر المحددة في القائمة.

```
void CListExDlg::OnTest()
{
	int nTotalSelected = 0;
	int nSel = m_listCtrl.GetNextItem(-1,LVNI_SELECTED );
	while( nSel != -1 )
	{
		nTotalSelected ++;
		nSel = m_listCtrl.GetNextItem(nSel , LVNI_SELECTED );
	}
	TRACE( "%d item selected \n", nTotalSelected);
```

```
}
```

والسلام عليكم وإلى اللقاء مع الجزء الثاني إن وجدنا حماسا كافيا.
في الجزء الثاني ستدخلون في لب البرمجة بفيجوال سي ++ ،الشجرة TreeView ، قواعد البيانات ، تقنية ADO ، تعدد المهام ، الرسم والتلوين ، النماذج العائمة والمرتسية ، حفظ وتحميل المستندات ، مربع النص الغني RichTextBox، صفحات الخصائص ، بناء أدوات أكتيف إكس، إنشاء مشروع محترف في المحاسبة العامة والتحليلية بواسطة فيجوال سي ++،وغيرها إن شاء الله