

**الباب العاشر**  
**حفظ واسترجاع الوثيقة**  
**Save and Open**

## مفتنم

فى هذا الباب سوف نستخدم إمكانات برنامج الأستوديو ودوال المؤسسة MFC فى إنشاء الطراز الثالث من برنامج الرسم MyPaintBrush ، والطراز الثانى من برنامج المحرر MyEditor . وسوف نمح هذه البرامج الخصائص الآتية:

١. تغيير امتداد الملف.
٢. توظيف أوامر القائمة للحفظ والاسترجاع (Open/Save).
٣. استخدام راية تغيير الحالة التى تخبر البرنامج بأن التطبيق يحتاج إلى حفظ.
٤. منح البرنامج خاصية السحب والإسقاط (Drag and Drop) التى تتميز بها البرامج النوافذية.
٥. تسجيل البرنامج فى بيئة النوافذ ، حتى يمكن فتح التطبيقات من نافذة الكشاف.
٥. ربط الأيقونة بملفات البيانات الناتجة من التطبيق ، من خلال بيئة النوافذ.



تذكر:

١. للاطلاع على الصيغة التفصيلية لإحدى الدوال أو الفصائل ، ضع مؤشر الفأر فوقها ثم اضغط زر اللوحة F1.
٢. لعرض الخريطة الكاملة لشجرة فصائل المؤسسة MFC ابحث فى شاشة النجدة (باستخدام البطاقة Search) عن "Hierarchy Chart".
٣. للاطلاع على الكود الكامل لأحد المشروعات ، افتح المشروع من القرص المصاحب للكتاب.

## (١٠-١) برنامج الرسم – الطراز ٣

في هذا الباب سوف نستخدم نفس البرنامج الذي أنشأناه من قبل في الفصول السابقة MyPaintBrush لكي نمحه إمكانية حفظ الملفات على القرص واسترجاعها في وقت لاحق. ولتحقيق ذلك فإنك قد تستخدم أحد طريقتين:

- إما أن تستخدم نفس البرنامج الذي أنشأته في الباب التاسع وتستكمل الرحلة من حيث انتهيت.
- أو تنسخ الدوسيه `\Examples\VCPP\Ch09\MyPaintBrush` الموجود على القرص المصاحب للكتاب ، إلى القرص الصلب ، وتبدأ منه العمل (ويمكنك تغيير اسمه إذا شئت).

## (١٠-٢) استخدام الأوامر الجاهزة للقائمة

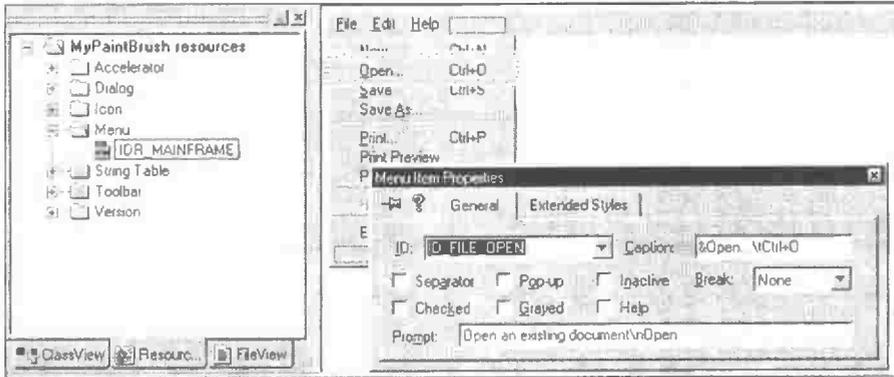
### **New/Open/Save/Print**

إن أوامر قائمة الملفات موجودة أصلاً في البرنامج بصورة سابقة التعريف ، وتستطيع استخدامها كما هي ، كما تستطيع أن تمسحها ثم تستخدم مشهد الموارد (Resource View) لخلقها من جديد. وفي هذا الباب سوف نكتفي بالأوضاع سابقة التعريف لأوامر قائمة الملفات وهي موضحة جميعاً بالجدول التالي (لاحظ أن استخدام دوال المقابض يستلزم إمدادها بالكود اللازم لتشغيلها – لقد عرضناها بالجدول من باب العلم بالشيء ، لكنك لا تحتاج إليها) :

رقم التعارف	اسم أمر القائمة	زر التفعيل	اسم المقبض
ID_FILE_NEW	&New	Ctrl+N	OnFileNew()
ID_FILE_OPEN	&Open	Ctrl+O	OnFileOpen()
ID_FILE_SAVE	&Save	Ctrl+S	OnFileSave()
ID_FILE_SAVE_AS	Save &As		OnFileSaveAs()
ID_FILE_MRU_FILE1	Recent File		

جدول (١٠-١) بيانات أوامر قائمة الملفات (Files)

ويمكنك مشاهدة هذه الأرقام والوظائف باستخدام دوسيه القائمة (Menu) بمشهد الموارد ، ثم الضغط ضغطة مزدوجة على الاختيار المطلوب كما هو موضح بالشكل التالي ، حيث نرى نافذة الأمر .File-Open



شكل (١٠-١) محرر أوامر القائمة

لاحظ أن الأمر Recent File يبدو معتماً في القائمة. وعند تشغيل البرنامج في الحفظ والاسترجاع ، سوف تلاحظ أن هذا الأمر يستبدل بأسماء الملفات التي تم فتحها مؤخراً.



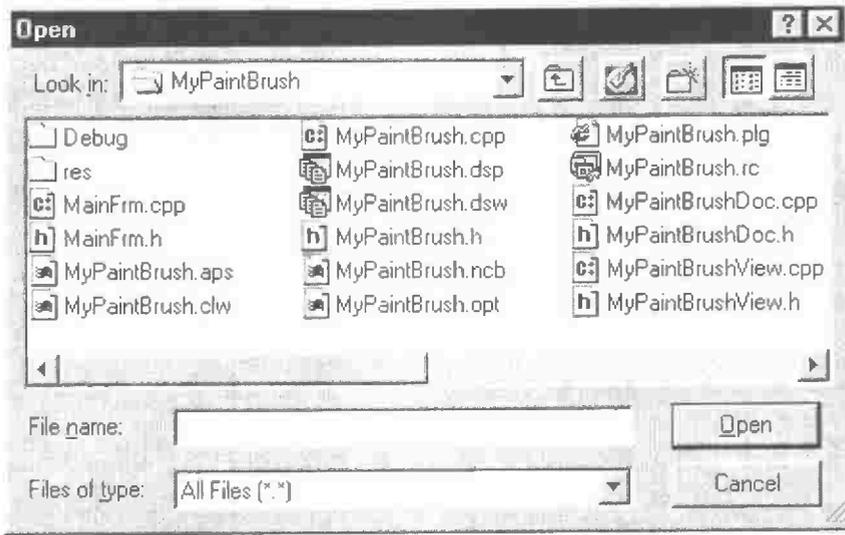
## وظائف أوامر القائمة

فيما يلي نعلق على كل أمر من هذه الأوامر ، ويمكنك تجربته قبل تعديل الكود ، حتى تشاهد الفرق بين الحالتين:

١. **File-New**: كما هو معروف في جميع البرامج النوافذية ، فإن هذا الأمر يؤدي إلى مسح محتويات الشاشة الحالية وبدء نافذة جديدة خالية من المحتويات (في حالة المشروع ذي الوثائق المتعددة MDI فإن هذا الأمر يؤدي إلى خلق وثيقة جديدة).  
ويمكنك التوصل إلى هذا الأمر بعدة طرق:

- باستخدام أمر قائمة الملفات File-New.
  - بالضغط على مجموعة أزرار التعجيل Ctrl+N.
  - بالضغط على أيقونة الصفحة البيضاء "□" بسطر الأدوات.
- جرب هذا الأمر بعد رسم بعض الخطوط على الصفحة فتحصل على صفحة بيضاء.

٢. **File-Open**: يؤدي هذا الأمر إلى فتح نافذة الملفات لاختيار الملف المطلوب تحميله ، كما بالشكل التالي. مع ملاحظة أن ملفات البرنامج MyPaintBrush ليس لها امتداد حتى الآن (وهذا هو موضوعنا في هذه الفقرة) ولذلك نرى كلمة All Files أمام نوع الملف (Files of type). كما أن هذا الأمر لا يعمل إلا بعد إضافة الكود اللازم لقراءة البيانات من القرص بالصورة المناسبة كما سنرى.



شكل (١٠-٢) نافذة الأمر Open

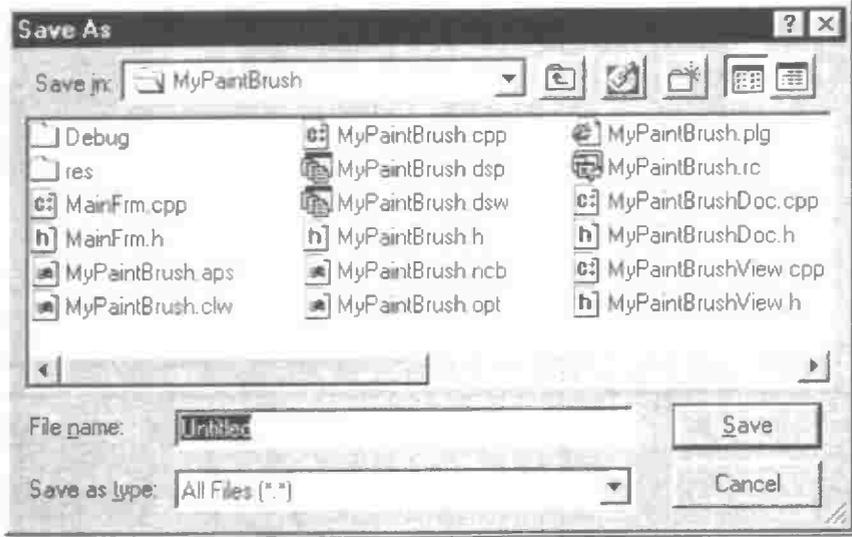
ويمكنك التوصل إلى هذا الأمر بإحدى الطرق الآتية:

- باستخدام أمر قائمة الملفات File-Open.
- بالضغط على مجموعة أزرار التعجيل Ctrl+O.
- بالضغط على أيقونة الدوسيه "تصانيف" بسطر الأدوات.

٣. **File-Save As** و **File-Save**: تؤدي هذه الأوامر إلى ظهور نافذة

الملفات كما بالشكل التالي ، ونلاحظ بها أيضاً أن الملفات ليس لها امتداد ما ولذلك نرى العبارة "All Files" بالنافذة (وهذا موضوع هذه الفقرة أيضاً). والفارق الرئيسي بين الأمرين أن الأمر save يؤدي إلى حفظ الملف مباشرة إذا كان له اسم ما وسبق حفظه. أما إذا كان الملف جديداً فإن كلاً من الأمرين يؤديان إلى ظهور النافذة الموضحة بالشكل التي تحمل العنوان

Save As. ويحتاج هذا الأمر أيضاً إلى تدعيمه بالكود اللازم لحفظ البيانات بالصورة المطلوبة كما سنرى.



شكل (٣-١٠) نافذة الأمر Save أو Save As

ويمكن التوصل إلى هذا الأمر بالطرق الآتية:

- باستخدام أمر قائمة الملفات File-Save أو File-Save As.
  - بالضغط على مجموعة أزرار التعجيل Ctrl+S.
  - بالضغط على أيقونة حفظ الملف "📁" بسطر الأدوات.
٤. الأمر Print: ولا يحتاج هذا الأمر إلى مزيد من المعالجة لأنه يؤدي إلى ظهور نافذة الطباعة المعتادة وطباعة محتويات الصفحة. ويمكنك تنفيذ هذا الأمر بعدة طرق:
- باستخدام أمر قائمة الملفات File-Print.
  - بالضغط على مجموعة أزرار التعجيل Ctrl+P.

• بالضغط على أيقونة الطابعة " " بستر الأدوات.

### (٣-١٠) تحديد امتداد الملفات

لكل برنامج تطبيقي امتداد معين لأسماء الملفات التي يحفظها على القرص مثل الامتداد bmp. لملفات البرنامج Paint. وفي هذا التطبيق سوف نختار الامتداد pnt. كامتداد مميز لبرنامجنا.

#### تحديد امتداد الملف باستخدام موارد البرنامج

لتحقيق ذلك اتبع الخطوات التالية:

١. افتح مشهد الموارد (Resource View).
٢. افتح الدوسيه String Table ، ثم اضغط ضغطة مزدوجة على الملف String Table ، فيظهر جدول الحرفيات في القسم الأيمن من نافذة الأستوديو ، كما هو موضح بالشكل التالي.

ID	Value	Caption
IDR_MAINFRAME	128	MyPaintBrush\MyPaintBrush.Document\MyPaintBrush
AFX_IDS_APP_TITLE	57344	MyPaintBrush
AFX_IDS_IDLEMESSAGE	57345	Ready
ID_FILE_NEW	57600	Create a new document\New
ID_FILE_OPEN	57601	Open an existing document\Open
ID_FILE_CLOSE	57602	Close the active document\Close
ID_FILE_SAVE	57603	Save the active document\Save
ID_FILE_SAVE_AS	57604	Save the active document with a new name\Save As
ID_FILE_PAGE_SETUP	57605	Change the printing options\Page Setup
ID_FILE_PRINT_SETUP	57606	Change the printer and printing options\Print Setup
ID_FILE_PRINT	57607	Print the active document\Print
ID_FILE_PRINT_PREVIEW	57609	Display full pages\Print Preview
ID_FILE_MRU_FILE1	57615	Open this document

شكل (٤-١٠) جدول الحرفيات (String Table)

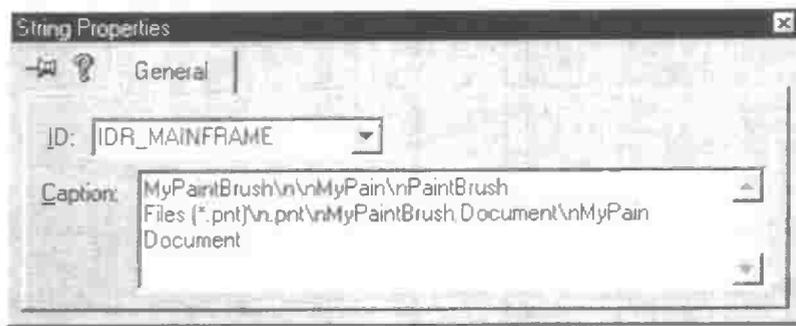
٣. إن السطر الأول الموجود بجدول الحرفيات يحتوى على النص الآتى كنص سابق التعريف:

```
MyPaintBrush\n\nMyPain\n\n\nMyPaintBrush.Document\nMyPain Document
```

٤. غير محتويات هذا النص لتصبح كالاتى (الأجزاء الجديدة موضحة بالبنط الثقيل):

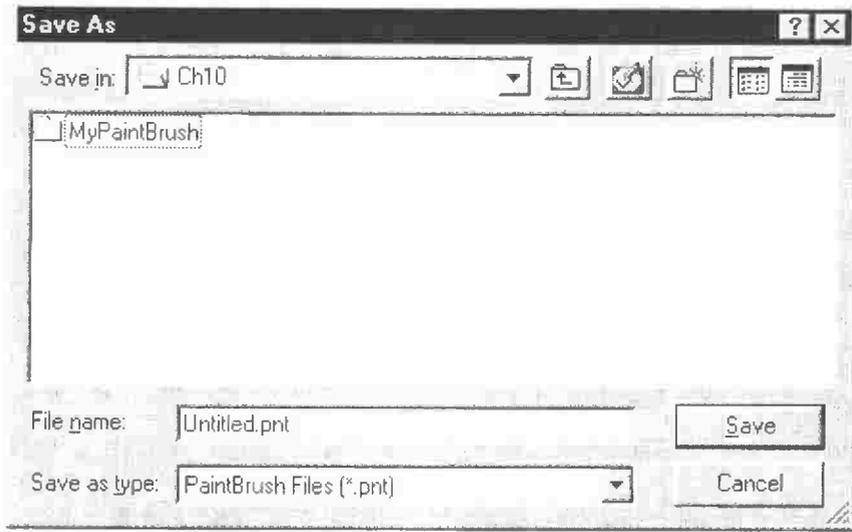
```
MyPaintBrush\n\nMyPain\nPaint Brush Files  
(* .pnt)\n.pnt\nMyPaintBrush.Document\nMyPain Document
```

ويتم التعديل بالضغط بضغط مزدوجة على السطر المطلوب تعديله (والذى يحمل رقم التعارف ID\_MAINFRAME) ، فتظهر النافذة القافزة الموضحة بعد. عدل النص الموجود فى الصندوق Caption بدون استخدام الزر Enter.



شكل (١٠-٥) نافذة تعديل امتداد الملف

٥. جرب الآن استخدام الأمر Open أو الأمر Save فتشاهد الشكل الموضح بعد ، ونلاحظ فيه أن العبارة "PaintBrush Files (\*.pnt)" قد ظهرت فى صندوق نوع الملف. كما أن النافذة ظهرت خالية تماماً من الملفات لأنها تعرض الملفات ذات الامتداد .pnt. فقط. وعندما نحفظ بعض الملفات سوف نشاهد أسماءها فى النافذة.



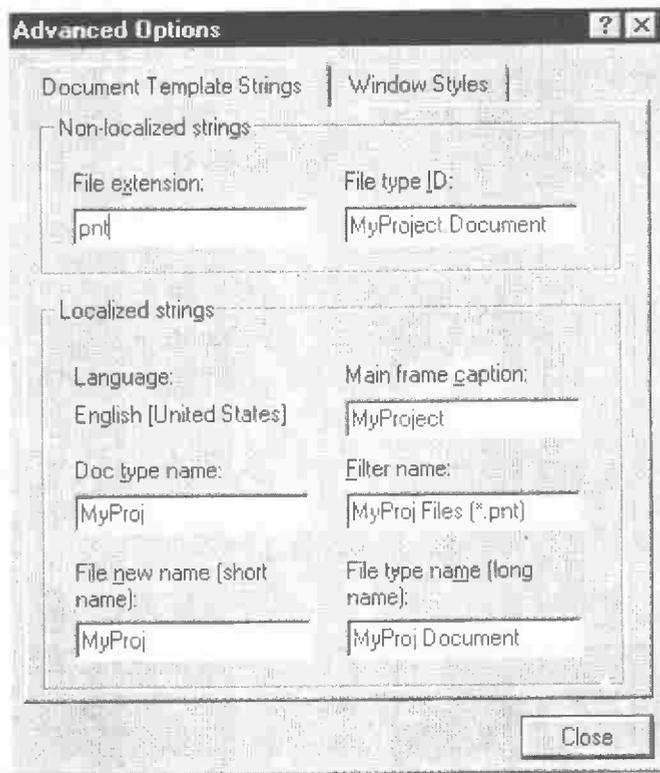
شكل (١٠-٦) نافذة الحفظ بعد تحديد امتداد الملفات ليكون .pnt.

### تحديد امتداد الملف عند إنشاء المشروع

يمكنك تحديد امتداد الملف أيضاً عند خلق المشروع لأول مرة ،  
وذلك باتباع الآتى:

١. بعد تحديد اسم ونوع المشروع ، استمر فى الضغط على الزر  
Next حتى تصل إلى النافذة رقم ٤ (Step 4 of 6). كما هو  
موضح بالشكل التالى.





شكل (١٠-٨) تحديد امتداد الملف في نافذة الساحر

## Save/Open

## (١٠-٤) الحفظ والاسترجاع

تتم عمليات القراءة والكتابة باستخدام الدالة `Object::Serialize()` وهي موجودة بداخل فصيلة الوثيقة التي ينشئها الساحر عند خلق المشروع. ولا تحتوى هذه الدالة إلا على الهيكل الخارجى لمنشأ شرطى (if condition) ، وعلينا أن نضيف إليه الكود اللازم لإتمام العملية. وهذا هو نص الدالة الموجود بالفصيلة `CMyPaintBrushDoc` قبل التعديل:

```

////////////////////////////////////
// CMyPaintBrushDoc serialization

void CMyPaintBrushDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        // TODO: add storing code here
    }
    else
    {
        // TODO: add loading code here
    }
}

```

وكما نلاحظ أن الدالة تحتوى على البارامتر `CArchive& ar` وهو عبارة عن مرجع إلى هدف من الفصيلة `CArchive` من فصائل المؤسسة MFC ، وهى الفصيلة المختصة بالكتابة على القرص أو القراءة منه. ونلاحظ فى بداية المنشأ الشرطى أن الشرط المستخدم يحتوى على تعبير بوليانى (منطقى) هو "`ar.IsStoring()`". إن هذا التعبير يستخدم القيمة المرتجعة من الدالة `IsStoring()` (من أعضاء الفصيلة `CArchive`). وتكون القيمة المرتجعة من هذه الدالة "صحيح" (`true`) فى حالة الكتابة ، كما تكون "غير صحيح" (`false`) فى حالة القراءة. وعلينا فى حالة الحفظ أن نستخدم الدالة `Serialize()` فى حفظ مصفوفة الخطوط `m_MyPaintArray` التى أنشأناها من قبل فى الباب السابق. وفى حالة القراءة فإننا نستخدم نفس الدالة لقراءة مصفوفة الخطوط. وفى الشكل التالى قد أضفنا الدالة `Serialize()` مقرونة باسم هدف المصفوفة `m_MyPaintArray` فى كل من بلوكات العبارة الشرطية باستخدام البنط الثقيل:

```

////////////////////////////////////
// CMyPaintBrushDoc serialization

void CMyPaintBrushDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        // TODO: add storing code here      (في حالة الكتابة (صحيح)
        m_MyPaintArray.Serialize(ar);
    }
    else
        (في حالة القراءة (غير صحيح)
    {
        // TODO: add loading code here
        m_MyPaintArray.Serialize(ar);
    }
}

```

وحتى يتم استخدام هاتين الدالتين يلزم إضافة أربعة أجزاء من الكود بالصورة الآتية:

١. الماكرو DECLARE\_SERIAL بملف عناوين الفصيلة CMyPaintClass

(الملف MyPaintBrushDoc.h)

٢. الماكرو IMPLEMENT\_SERIAL بملف تطبيق الفصيلة (الملف

(MyPaintBrushDoc.cpp)

٣. إعلان للدالة **Serialize()** بملف عناوين الفصيلة CMyPaintClass

(الملف MyPaintBrushDoc.h)

٤. تعريف الدالة **Serialize()** بملف تطبيق الفصيلة (الملف

(MyPaintBrushDoc.cpp)

معنى هذه الخطوات أننا سوف نركب الدالة الافتراضية **Object::Serialize()** من داخل فصيلة الهدف المراد قراءته أو

كتابته.

وهذه هي الخطوات التفصيلية:

1. أضف الماكرو DECLARE\_SERIAL إلى ملف إعلان الفصيلة (MyPaintBrushDoc.h). أضف أيضاً دالة بناء سابقة التعريف (سيلي شرح أهميتها) ، كما هو موضح بالشكل التالي حيث نرى الأجزاء الجديدة بالبنط الثقيل:

```
class CMyPaintClass: public COBJECT
{
protected:
    int m_x1, m_x2, m_y1, m_y2;
    CMyPaintClass() {}           دالة بناء سابقة التعريف
    DECLARE_SERIAL(CMyPaintClass)   الماكرو
public:
    CMyPaintClass(int x1, int y1, int x2, int y2) {
        m_x1 = x1;
        m_x2 = x2;
        m_y1 = y1;
        m_y2 = y2;
    }
    void DrawIt(CDC *PDC);
};
```

- ونلاحظ أن البارامتر المستخدم مع الماكرو DECLARE\_SERIAL عبارة عن اسم الفصيلة المطلوب التعامل مع أهدافها.
2. أضف الماكرو IMPLEMENT\_SERIAL إلى ملف الوثيقة MyPaintBrushDoc.cpp ، وليكن في نهاية الملف ، كالتالي:

```
void CMyPaintBrushDoc::OnUpdateEditUndo(CCmdUI* pCmdUI)
{
    // TODO: Add your command update UI handler code here
    pCmdUI -> Enable(m_MyPaintArray.GetSize());
}
```

## IMPLEMENT\_SERIAL (CMyPaintClass, CObject, 1)

والبارامترات الممّرة لهذا الماكرو هي:

- اسم الفصيلة CMyPaintClass
- اسم فصيلة الأساس CObject
- رقم طراز البرنامج (Schema Number) وهو الرقم 1 ، وهو يستخدم لمنع التداخل بين البيانات عند استخدام أكثر من طراز لنفس البرنامج.

٣. أضف إعلان الدالة الافتراضية **Serialize()** إلى ملف عناوين الفصيلة CMyPaintClass كالتالي:

```
public:
    CMyPaintClass(int x1, int y1, int x2, int y2) {
        m_x1 = x1;
        m_x2 = x2;
        m_y1 = y1;
        m_y2 = y2;
    }
    void DrawIt(CDC *PDC);
    virtual void Serialize(CArchive& ar);
};
```

٤. أضف تعريف الدالة الراكبة **Serialize()** إلى ملف الوثيقة كالتالي:

```
void CMyPaintClass::DrawIt(CDC *PDC)
{
    PDC -> MoveTo(m_x1, m_y1);
    PDC -> LineTo(m_x2, m_y2);
}

void CMyPaintClass::Serialize(CArchive &ar)
{
    if (ar.IsStoring())
```

```
ar << m_x1 << m_y1 << m_x2 << m_y2;
else
ar >> m_x1 >> m_y1 >> m_x2 >> m_y2;
}
```

في هذا التعريف ، تتم العملية الفعلية لتخزين البيانات الأعضاء أو قراءتها باستخدام مؤثرات الدخل والخرج << و >> (وهي مؤثرات محملة تحميلاً زائداً بواسطة الفصيلة CArchive). يمكنك تجربة البرنامج عند هذا الحد وذلك باستخدام الأمر Save لحفظ ملف الرسم ثم فتحه ثانية باستخدام الأمر New ، وسوف تجد أن عمليات الحفظ والاسترجاع عاملة.

### المناقشة

استخدمنا هنا الدالة **CObject::Serialize()** من فصيلة الأساس لتحقيق تسهيلات عامة في عمليات الحفظ والاسترجاع. كما استخدمنا الدالة الراكبة **CMyPaintClass::Serialize()** ، من فصيلة الهدف ، لإجراء عمليات القراءة والتخزين الفعلية. وتتم عملية الحفظ كالاتي:

١. تتولى الدالة **Serialize()** كتابة معلومات الهدف المراد حفظه على القرص.

٢. تستدعي الدالة الراكبة **Serialize()** لحفظ البيانات الفعلية الهدف.

كما تتم عملية القراءة كالاتي :

١. تتولى الدالة `Serialize()` قراءة بيانات فصيحة الهدف من القرص ، ثم خلق الهدف ديناميكياً باستخدام دالة البناء ، مع الاحتفاظ بمؤشر إلى الهدف.
٢. تستدعي الدالة الراكبة `Serialize()` التي تقرأ البيانات من القرص وتخزنها في الهدف نفسه.

## SetModifiedFlag

## (١٠-٥) راية تغيير الحالة

عندما تقوم بإجراء أى تعديل على ملف ما ، ثم تستخدم الأمر `New` فإن البرنامج عادة يرسل إليك رسالة تقول أى الملف الحالي قد تغيرت محتوياته ، وتسألك الرسالة إذا كنت ترغب فى حفظ هذه التغييرات. ومع ذلك فإن هذه الخاصية التى نراها فى جميع البرامج ليست خاصة ذاتية ، وإنما هى مسئولية المبرمج. وتتم برمجة هذه الخاصية بأسلوب بسيط. فعندما يتم إجراء أى تعديل على الملف المفتوح عليك أن ترفع راية ما (`Flag`) تخبرنا بأن محتويات هذا الملف تحتاج إلى حفظ ، فإذا تمت عملية الحفظ تخفض الـ `Flag`. والراية ليست موضوعاً جديداً ، فهى مجرد متغير بوليانى يأخذ القيمة `true` عندما تكون الـ `Flag` مرفوعة ، أو القيمة `false` عند خفض الـ `Flag`.

وتمدنا المؤسسة `MFC` بـ `Flag` مبنية فى اللغة تتحكم فيها الدالة `CDocument::SetModifiedFlag()`. ويؤدى استدعاء هذه الدالة إلى رفع الـ `Flag` أى أنها تأخذ القيمة `true`. أما خفض الـ `Flag` فيتم تلقائياً

عند حفظ الملف. وفي تطبيقنا الحالي علينا أن نستدعي هذه الدالة في عدة مواضع.

أضف هذه الدالة عند إضافة خط جديد بالدالة `PutLine()` بملف الوثيقة كالاتي:

```
void CMyPaintBrushDoc::PutLine(int x1, int y1, int x2, int y2)
{
    CMyPaintClass *ptrToLine = new CMyPaintClass(x1, y1, x2, y2);
    m_MyPaintArray.Add(ptrToLine);
    SetModifiedFlag();
}
```

أضف الدالة أيضاً عندما تسمح الرسم الحالي باستخدام أمر القائمة `Clear All` ، وذلك بداخل جسم الدالة `OnEditClearAll()` بنفس الملف:

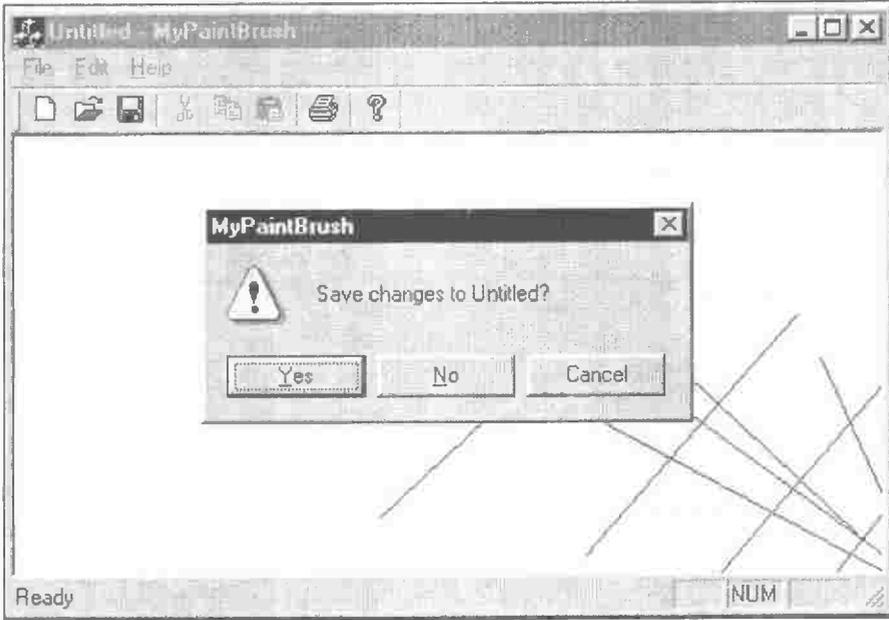
```
void CMyPaintBrushDoc::OnEditClearAll()
{
    // TODO: Add your command handler code here
    DeleteContents();
    UpdateAllViews(0);
    SetModifiedFlag();
}
```

أضف الدالة أيضاً عندما تستخدم الأمر `Undo` لتغيير الرسم. هذه هي الدالة `OnEditUndo()` بنفس الملف:

```
void CMyPaintBrushDoc::OnEditUndo()
{
    int i = m_MyPaintArray.GetUpperBound();
    if (i > -1) {
        delete m_MyPaintArray.GetAt(i);
        m_MyPaintArray.RemoveAt(i);
    }
    UpdateAllViews(0);
}
```

```
SetModifiedFlag();
```

جرب الآن تنفيذ البرنامج وسوف ترى أن استخدام أى أمر من أوامر القائمة التى تتضمن مسح الوثيقة الحالية مثل New أو Open سوف يودى إلى ظهور النافذة الموضحة بالشكل التالى لتذكيرك بحفظ الملف.



شكل (٩-١٠) رسالة: هل ترغب فى حفظ التغييرات؟

### ملاحظة:

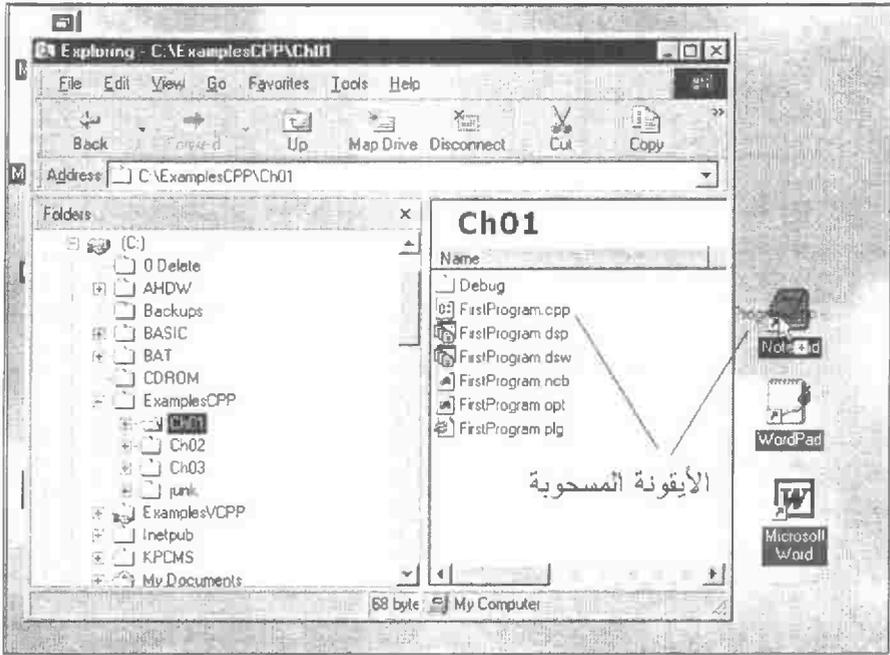
لا تحتاج إلى استخدام الدالة لخفض الراية لأن هذه العملية تتم آلياً. ومع ذلك فلو أنك أردت استخدام الدالة فى هذا الغرض فعليك بإمرار البارامتر FALSE صراحةً ، فصيغة الدالة تحتوى على بارامتر بولياني سابق التعريف كالاتى:  
CDocument::SetModifiedFlag(BOOL bModified = TRUE);

**Drag & Drop**

**(١٠-٦) استخدام السحب والإسقاط**

لو أنك فتحت نافذة الكشاف وسحبت أحد أيقونات ملفات النصوص (وليكن الملف FirstProgram.cpp بالباب الأول) ثم أسقطت هذه الأيقونة على أيقونة برنامج النوتة فإنه سوف يعرض فوراً على صفحة برنامج النوتة. ولو أنك أسقطت نفس الملف على أيقونة البرنامج "وورد باد" فسوف يعرض على شاشة هذا البرنامج ، ويحدث نفس الشيء مع أى برنامج قادر على عرض النصوص مثل "وورد". أى أن عملية السحب والإسقاط تقابل الأمر Open. ولكنها توفر الوقت وتغني عن الكثير من الخطوات. ولكي تجرب هذه الخاصية - إذا كنت لم تستخدمها من قبل - فلتفتح نافذة الكشاف بالصورة الموضحة بالشكل ، ولتضع على سطح المكتب بعض الطرق المختصرة إلى كل من البرامج وورد ، وورد باد ، والنوتة. ثم جرب سحب أحد ملفات لغة سي ++ وأسقطه على أى من أيقونات البرامج فيفتح البرنامج على الفور وترى على شاشته سطور الكود.

يمكنك أيضاً استخدام نافذة مفتوحة للبرنامج المستقبل بدلاً من الأيقونة. اسحب الملف من نافذة الكشاف إلى نافذة البرنامج التطبيقي ، فتحصل على نفس النتيجة.



شكل (١٠-١٠) السحب والإسقاط

ولكى تكسب برنامجك هذه الخاصية فإنك تحتاج إلى استدعاء الدالة `CWnd::DragAcceptFiles()` والمكان المناسب للاستخدام هذه الدالة هو فصيلة التطبيق ، بداخل الدالة `InitInstance()` المسئولة عن بناء نافذة التطبيق الرئيسية (بالملف `MyPaintBrush.cpp`). والشكل التالي يوضح مكان الدالة الجديدة.

```
{
...
...
// Dispatch commands specified on the command line
if (!ProcessShellCommand(cmdInfo))
return FALSE;

// The one and only window has been initialized, so show and update it.
m_pMainWnd->ShowWindow(SW_SHOW);
m_pMainWnd->UpdateWindow();
}
```

```

دالة السحب والإسقاط
m_pMainWnd->DragAcceptFiles();

return TRUE;
}

```

وتستخدم الدالة **DragAcceptFiles()** مع العضو **m\_pMainWnd** الذي يختزن مؤشراً إلى نافذة الإطار العام للتطبيق (وهو من أعضاء الفصيلة **CWinThread**).

ونلاحظ في الكود السابق أن الدالة الجديدة **DragAcceptFiles()** تأتي في نهاية الدالة **InitInstance()**. وبالرغم من أن هذا ليس بشرط لازم ، ولكنها لا بد أن تلي الدالة **ProcessShellCommand()** ، لأن هذه الدالة الأخيرة هي المسئولة عن إنشاء نافذة الإطار العلم وتخصيص قيمة العضو **m\_pMainWnd**.

قم ببناء البرنامج وجرب الآتي:

- ارسم بعض الخطوط ، ثم احفظ الملف تحت اسم ما وليكن **x.pnt**.
- افتح نافذة الكشاف واعرض الدوسيه المحتوى على الملف **x.pnt**.
- افتح نافذة الكشاف المحتوية على البرنامج **MyPaintBrush.exe** (وهو موجود بالدوسيه **\Examples\VCPP\Ch10\MyPaintBrush\Debug** على القرص المصاحب للكتاب).
- اسحب الأيقونة **x.pnt** وأسقطها على أيقونة البرنامج **MyPaintBrush.exe** وشاهد النتيجة.

## Application Registration (٧-١٠) تسجيل التطبيق

من خصائص البرامج النوافذية أن نظام التشغيل يتعرف علي ملفاتنا ، باستخدام امتداد الاسم ، ويفتحها بمجرد أن تضغط عليها في نافذة الكشاف. فلو أنك ضغطت ضغطة مزدوجة على ملف ذي امتداد bmp. فإنه يؤدي إلى فتح نافذة البرنامج Paint.exe وتحميل الملف. ومع ذلك فقد تصادف بعض الامتدادات التي لا يتعرف عليها نظام التشغيل وبالتالي فهي غير مرتبطة بأي برنامج تطبيقي. وتسمى خاصية ربط ملفات البيانات بالبرامج التطبيقية بخاصية التسجيل (Registration). ولو أنك حاولت تشغيل أحد ملفات الرسم التي تحمل الامتداد .pnt. بضغطة مزدوجة ، فلن تنجح حتى يتم تسجيل البرنامج. ومن المتوقع أن ترى النافذة الموضحة بعد التي تسألك أن تربط الملف test.pnt بأحد البرامج التطبيقية.



شكل (١٠-١١) نافذة ربط ملف ببرنامج تطبيقي

ولكى تسجل تطبيقك ، استخدم الدالتين :

**CWinApp::EnableShellOpen()**  
**CWinApp::RegisterShellFileTypes()**

بداخل جسم الدالة **InitInstance()** بفصيصة التطبيق (بالملف  
MyPaintBrush.cpp) كالاتى :

```
RUNTIME_CLASS(CMainFrame), // main SDI frame window
RUNTIME_CLASS(CMyPaintBrushView));
AddDocTemplate(pDocTemplate);
// Register the MyPaintBrush program : أضف هذه الدوال:
EnableShellOpen();
RegisterShellFileTypes();
// Parse command line for standard shell commands, DDE, file open
CCommandLineInfo cmdInfo;
ParseCommandLine(cmdInfo);
// Dispatch commands specified on the command line
if (!ProcessShellCommand(cmdInfo))
return FALSE;
```

لاحظ أن استدعاء هاتين الدالتين لابد وأن يلي استدعاء الدالة **AddDocTemplate()** المسئولة عن إضافة نموذج الوثيقة إلى هدف التطبيق ، بحيث يصبح كل من نوع الملف والامتداد متاحاً لهدف التطبيق.

نفذ البرنامج ، بعد هذا التعديل ، ثم جرب فتح أحد الملفات التى تحمل الامتداد .pnt من نافذة الكشاف.

هل لاحظت أن أيقونة الملفات ذات الامتداد .pnt. تأخذ شكل الأيقونة سابقة التعريف ، بمعنى أنها لا تأخذ شكل أيقونة البرنامج المميزة؟ حاول تغيير شكل الأيقونة إلى الشكل "39" كنوع من التدريب. وسوف نعاود هذه الكرة فى نهاية الباب.

## (١٠-٨) محرر النصوص \_ الطراز ٢

بدأنا في الباب الثامن برنامجاً لتحرير النصوص بالاسم MyEditor ، وقد تحققنا من أن عرض النص على الشاشة أسهل بكثير من عرض الرسومات ، حيث أن أغلب الخصائص المطلوبة لمعالجة النصوص مبنية في الفسييلة CEditView. وفي هذه الفقرة نستكمل بناء ملامح هذا البرنامج.

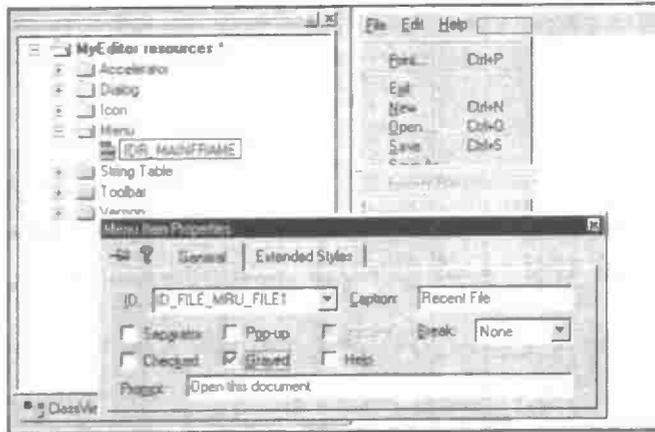
### اختيارات قائمة الملفات

لقد انصب اهتمامنا في هذا البرنامج على اختيارات قائمة التحرير (Edit) وعلى تصميم أيقونة البرنامج. أما قائمة الملفات (Files) فقد كانت تحتوى على اختيارين فقط ، هما اختيار الطباعة Print واختيار الخروج Exit ، حيث أننا مسحنا جميع الاختيارات الأخرى. لكي نضيف الاختيارات الباقية في قائمة الملفات فإنك تحتاج إلى أسماء ثوابت التعارف الموضحة بالجدول التالي. أما طريقة إضافة الأزرار فتتم باستخدام دوسيه القائمة بمشهد الموارد.

زر التعجيل	اسم أمر القائمة	رقم التعارف
Ctrl+N	&New	ID_FILE_NEW
Ctrl+O	&Open	ID_FILE_OPEN
Ctrl+S	&Save	ID_FILE_SAVE
	Save &As	ID_FILE_SAVE_AS
	Recent File	ID_FILE_MRU_FILE1

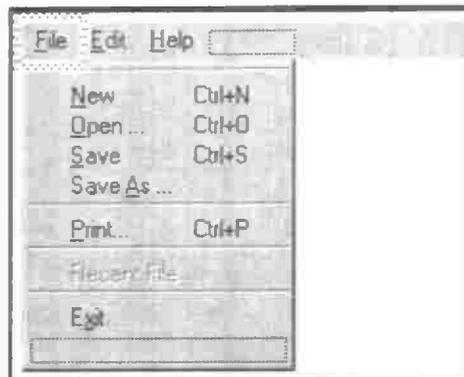
جدول (١٠-٢) بيانات أوامر قائمة الملفات (Files)

أضف الأوامر الموضحة بالجدول السابق وبالإستعانة بالشكل التالى الذى يوضح إضافة أمر القائمة Recent File. وفى نهاية العملية قم بترتيب الأوامر فى القائمة بسحبها وإسقاطها فى الأماكن المناسبة.



شكل (١٠-١٢) إضافة أوامر قائمة الملفات

وعندما تنتهى من تعديل أوامر القائمة والفواصل بينها ، فمن المتوقع أن تكون كما بالشكل التالى.



شكل (١٠-١٣) الصورة النهائية لأوامر القائمة

وبخلاف برامج الرسم ، فإن جميع أوامر قائمة الملفات ، بما فيها أوامر الحفظ والاسترجاع ، سوف تكون عاملة تلقائياً. وبصرف النظر عن امتداد الملفات المميزة لهذا البرنامج ففي إمكانك أن تحفظ ملفاً بأي امتداد مثل test.abc وتسترجعه بالأمر Open فيما بعد.

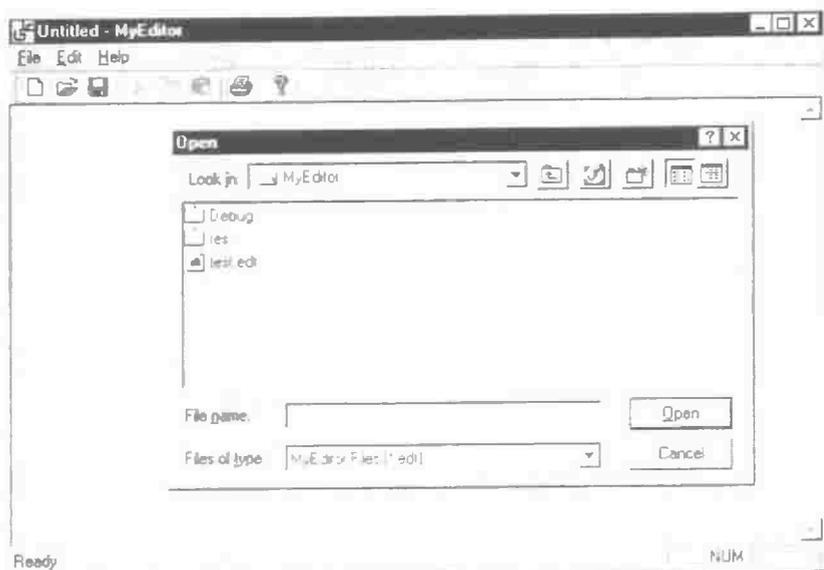
### تعديل امتداد اسم الملفات

---

من الأفضل حتى تحفظ للبرنامج هويته أن تمنح لملفاته امتداداً ما. وسنترك لك هذه المهمة كتدريب. وهذا هو التغيير المطلوب إدخاله في جدول الحرفيات للحصول على الامتداد ".edt":

MyEditor\n\nMyEdit\n**MyEditor Files**  
(\***.edt**)\n**.edt**\nMyEditor.Document\nMyEdit Document

والشكل التالي يوضح نافذة فتح الملفات بعد إجراء هذا التغيير ، ونرى بها الامتداد الجديد في صندوق نوع الملف.



شكل (١٠-١٤) نافذة فتح ملف بالامتداد \*.edt

### إضافة خاصية السحب والإسقاط (Drag and Drop)

لإكساب البرنامج خاصية السحب والإسقاط ، فإننا نتخذ نفس الخطوات التي اتبعناها مع برنامج الرسم. أضف الدالة **OnInitInstance()** إلى نهاية بلوك الدالة **CWnd::DragAcceptFiles()** ، وهى الدالة المسؤولة عن بناء النافذة الرئيسية للتطبيق بالملف **MyEditor.cpp** ، كما هو موضح بالشكل التالى.

```

BOOL CMyEditorApp::OnInitInstance()
{
    AfxEnableControlContainer();
    ...
    ...
    m_pMainWnd->ShowWindow(SW_SHOW);
    m_pMainWnd->UpdateWindow();
    m_pMainWnd->DragAcceptFiles();      أضف هذه الدالة
    return TRUE;
}

```

بإجراء هذا التغيير يصبح من الممكن أن تسحب أى ملف نصوص (بصرف النظر عن امتداد الاسم) من نافذة الكشف ، وتسقطه على أيقونة برنامج المحرر (أو طريق مختصر إليه) ، فترى النص بداخل نافذة البرنامج. وفي الشكل التالى نرى أحد أيقونات الكشف أثناء إسقاطها على أيقونة برنامج المحرر. ومن الجدير بالذكر أن ملفات النصوص (.txt) وملفات إتش - تى - إم - إل (.htm) وملفات لغة سى و سى++ كلها تعتبر من ملفات النصوص.



شكل (١٠-١٥) سحب أيقونة نصوص وإسقاطها فوق طريق مختصر إلى المحرر

## تسجيل برنامج المحرر

فى هذه الفقرة سوف نقوم بعملية تسجيل برنامج المحرر ، بنفس الطريقة التى سجلنا بها برنامج الرسم. ومن المتوقع بعد

التسجيل أن نستطيع فتح أى ملف بالامتداد .edt. مباشرة من شاشة الكشاف بمجرد الضغط على الأيقونة ضغطة مزدوجة. لتحقيق ذلك أضف الكود الموضح بالبنط الثقيل بالشكل التالى إلى الدالة المسئولة عن بناء النافذة الرئيسية للتطبيق (**InitInstance()**) بالملف MyEditor.cpp. ولاحظ أن الكود الجديد يلى الدالة **AddDocTemplate()**

```

BOOL CMyEditorApp::InitInstance()
{
    AfxEnableControlContainer();
    ...
    ...
    AddDocTemplate(pDocTemplate);

```

أضف هذا الكود:

```

EnableShellOpen();
RegisterShellFileTypes();
// Parse command line for standard shell commands, DDE, file open
CCommandLineInfo cmdInfo;
ParseCommandLine(cmdInfo);

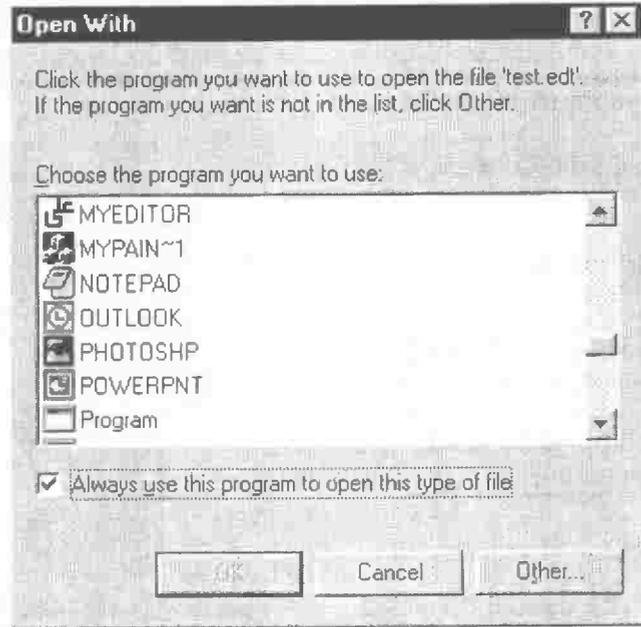
```

بعد إضافة هذه الدوال وتشغيل البرنامج سوف تلاحظ أنك تستطيع فتح الملفات ذات الامتداد .edt. من نافذة الكشاف مباشرة ، ومع ذلك فإن الملفات لا ترتبط بأيقونة البرنامج المميزة.

### ربط الملفات بأيقونة البرنامج

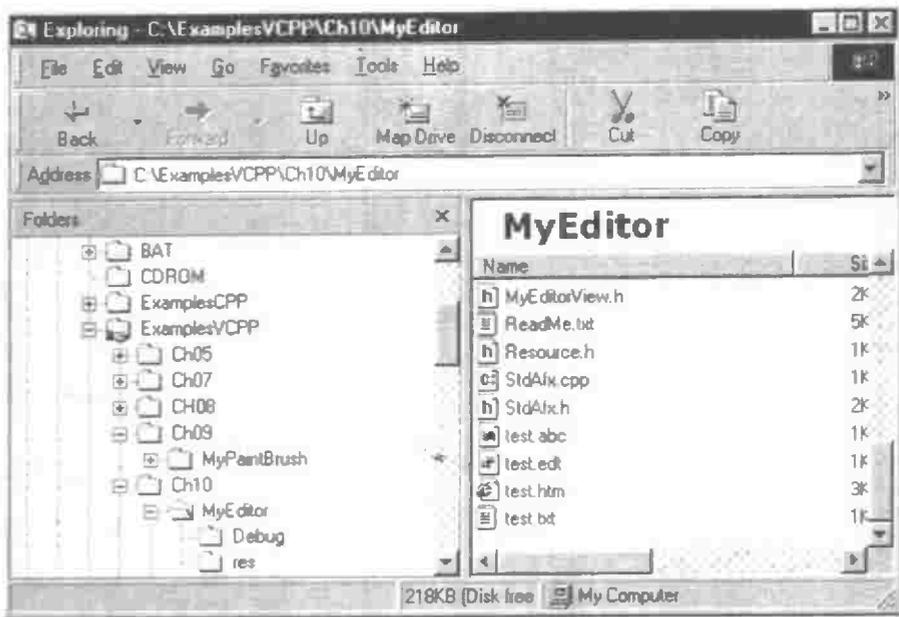
لكى تربط ملفا ما ببرنامج ما فإنك تستخدم الأمر Open with الموجود بالقائمة السريعة (النتيجة من الضغط على الزر الأيسر). ويظهر هذا الاختيار إذا كان الملف غير مسجل. أما إذا أردت

تحقيق ذلك بعد عملية التسجيل ، اضغط على الزر Shift ثم على الزر الأيمن للفأر فيظهر الاختيار Open with. وباستخدام هذا الأمر ينقلك إلى النافذة الموضحة بعد ، التي تختار منها التطبيق. وبالطبع ، فإن اسم التطبيق MyEditor سوف يظهر بهذه النافذة بعد عملية التسجيل (وليس قبل ذلك).



شكل (١٠-١٦) نافذة الأمر Open with وبها البرنامج MyEditor

اختر البرنامج MyEditor المميز بالأيقونة "على" ثم ضع علامة ؟ في صندوق الاختبار "Always use this program ...". وبمجرد إغلاق هذه النافذة سوف يتحقق الارتباط بين الملفات وبين أيقونة البرنامج. وفي الشكل التالي نرى بنافذة الكشاف أيقونة البرنامج test.edt وقد ظهرت بجوارها الأيقونة "على".



شكل (١٠-١٧) ملف البرنامج بنافذة الكشاف بأخذ شكل الأيقونة المميزة

## الموجز

١. قمنا في هذا الباب بتطوير برنامجين هما برنامج الرسم MyPaintBrush الذي بدأناه في الباب السابع ، وبرنامج المحرر MyEditor الذي بدأناه في الباب الثامن.
٢. استخدمنا جدول الحرفيات (String Table) بموارد البرنامج في تغيير امتداد الملف ، كما عرفنا كيفية تغيير الامتداد أثناء إنشء التطبيق.
٣. استخدمنا أوامر القائمة للحفظ والاسترجاع ، مع تعصيدها بالكود اللازم لقراءة وكتابة البيانات في صورة أهداف.
٤. كما استخدمنا راية تغيير الحالة التي تخبر البرنامج بأن التطبيق يحتاج إلى حفظ.

٥. منحنا البرنامج خاصية السحب والإسقاط التي تتميز بها البرامج النوافذية.

٦. كما مارسنا عملية تسجيل البرنامج في بيئة النوافذ ، حتى يمكن فتح التطبيقات من نافذة الكشاف.

٧. عرفنا كيف يتم ربط الأيقونة بملفات البيانات الناتجة من التطبيق ، من خلال بيئة النوافذ.

٨. في هذا الباب عرفنا الفصائل والدوال والثوابت الآتية:  
ثوابت تعارف:

```
ID_FILE_NEW  
ID_FILE_OPEN  
ID_FILE_SAVE  
ID_FILE_SAVE_AS  
ID_FILE_MRU_FILE1
```

دوال مقابض:

```
OnFileNew()  
OnFileOpen()  
OnFileSave()  
OnFileSaveAs()
```

فصائل:

```
CArchive  
CWnd  
CWinThread  
CWinApp
```

ماكرو:

```
DECLARE_SERIALIZE  
IMPLEMENT_SERIAL
```

## دوال أعضاء:

دالة الحفظ والاسترجاع	<b>CObject::Serialize()</b>
دالة الحالة (قراءة/كتابة)	<b>CArchive::IsStoring()</b>
دالة الراية	<b>CDocument::SetModifiedFlag()</b>
دالة السحب والإسقاط	<b>CWnd::DragAcceptFiles()</b>
دوال تسجيل التطبيق	<b>CWinApp::EnableShellOpen()</b> <b>CWinApp::RegisterShellFileTypes()</b>