

الفصل الخامس عشر العمل مع أشرطة الأدوات وشريط المعلومات

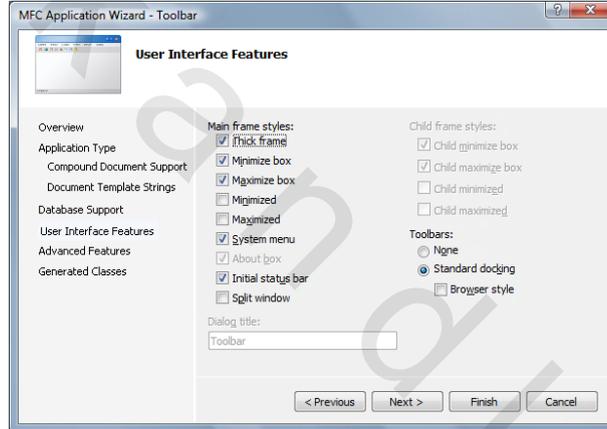
تستخدم معظم التطبيقات أشرطة الأدوات لمساعدة المستخدم على اختيار الأوامر المستخدمة بكثرة داخل التطبيق، حيث تعبر أزرار شريط الأدوات غالباً عن بعض الخيارات الموجودة داخل شريط القوائم. فإذا أراد المستخدم مثلاً طباعة الوثيقة الحالية فعليه نقر الزر المرسوم عليه رمز الطابعة من شريط الأدوات بدلاً من فتح قائمة **File** ثم اختيار الأمر **Print** مما يزيد من سرعة تنفيذ الأوامر داخل التطبيق.

بانتهاج هذا الفصل ستتعرف على:

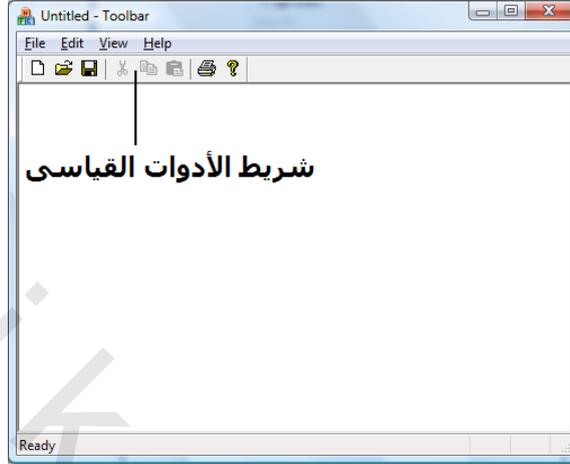
- ◆ إنشاء واستخدام أشرطة الأدوات
- ◆ استخدام شريط المعلومات لعرض مؤشرات عن عناصر التطبيق

استخدام شريط الأدوات القياسي

قم بإنشاء مشروع أحادي الوثيقة باسم **Toolbar** مع ترك الإعدادات الافتراضية لشريط الأدوات كما هي في التبويب **User Interface Features** داخل نافذة معالج إنشاء المشروع (انظر شكل ١٥-١)، تلاحظ إضافة شريط الأدوات القياسي إلى التطبيق الجديد، حيث يقوم المعالج بإنشاء الكود اللازم لإنشاء شريط الأدوات بما في ذلك صور الأزرار التي يحتويها هذا الشريط والتي تعبر بشكل كبير عن أوامر القوائم **File** و **Edit** و **Help** (انظر شكل ١٥-٢).



شكل ١٥-١ الإعدادات الافتراضية لشريط الأدوات القياسي



شكل ١٥-٢ شريط الأدوات القياسى المنشأ بواسطة المعالج

كيف يتم إنشاء شريط الأدوات القياسى؟

ذكرنا منذ قليل أن المعالج يقوم بكتابة الكود اللازم لإنشاء شريط الأدوات القياسى بما يحتويه من أزرار وصور، إلا أن التعرف على هذا الكود يساعدنا على فهم ما يحدث كى يُعرض هذا الشريط وبالتالي إمكانية التغيير في بعض خصائصه أو مكوناته. يتم إنشاء شريط الأدوات القياسى غالباً بتعريف عنصر عضو في التصنيف `CToolBar` باسم `m_wndToolBar` هكذا:

`CToolbar m_wndToolBar;`

وينبثق التصنيف `CToolbar` أساساً من التصنيف `CControlBar` الذى ينبثق بدوره من التصنيف الرئيسى `CWnd` وبالتالي فهو يحتوى على جميع الوظائف اللازمة لعرض شريط الأدوات في مكان معين داخل نافذة التطبيق أو جعله قابل للانتقال في أى مكان آخر كبقية النوافذ، كما يقوم أيضاً بإضافة الصور ومصفوفة الأزرار المستخدمة لتمثيل شريط الأدوات القياسى (راجع شكل ١٥-٢ السابق).

ذكرنا قبل ذلك أن التصنيف `CMainFrame` هو المسئول عن إنشاء الإطار الرئيسى لنافذة التطبيق أحادى الوثيقة أو متعدد الوثائق، وبمجرد إنشاءه يقوم المعالج باستدعاء الدالة `OnCreate()` لإنشاء كود نافذة شريط الأدوات القياسى كنافذة وليدة داخل إطار

```

IDR_MAINFRAME النافذة الأساسية، ثم يقوم بعد ذلك بتحميل شريط الأدوات
:MainFrm.cpp بسطور الكود التالية الموجودة بملف
if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD |
WS_VISIBLE | CBRS_TOP
| CBRS_GRIPPER | CBRS_TOOLTIPS |
CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ||
!m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
{
TRACE0("Failed to create toolbar\n");
return -1; // fail to create
}

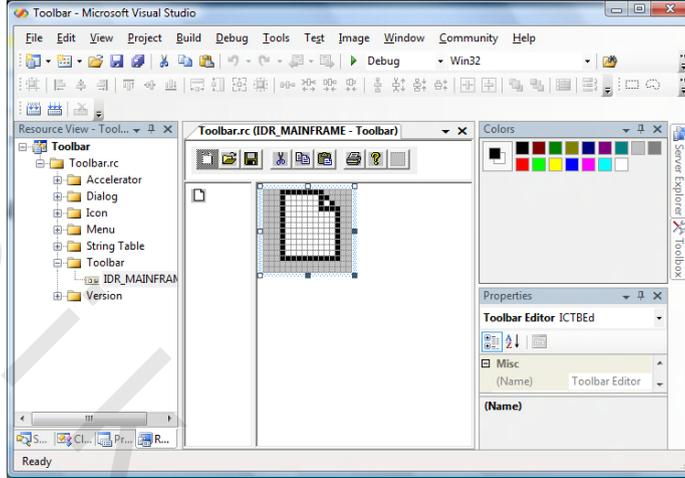
```

وعن هذا الكود، نوضح ما يلي:

- تم دمج إنشاء شريط الأدوات وتحميله داخل عبارة `if` بحيث تقوم الدالة `OnCreate()` بإرجاع القيمة `-1` في حالة فشل أى منهما.
- تم استدعاء الدالة `CreateEx()` والتي تستخدم لتمثيل أشرطة الأدوات وإنشاء حدودها، وتحتوى على ثلاثة معاملات يمكن بيانها كما يلي:
 - المعامل الأول عبارة عن مؤشر للنافذة الأم، وتم استخدام كلمة `this` دلالة على النافذة الحالية.
 - المعامل الثانى عبارة عن نمط الشريط وهو `TBSTYLE_FLAT` فى هذه الحالة، ويمكنك تغييره إلى `0` لإظهار الشريط فى شكله القديم.
 - يستخدم المعامل الثالث لتحديد خصائص شريط الأدوات باختيار التوليفة المناسبة من الجدول ١٥-١ التالى، لكن لا تنسى فى هذه الحالة تضمين كل من `WS_VISIBLE` و `WS_CHILD`.
- تم استدعاء الدالة `LoadToolBar()` لتحميل شريط الأدوات بتمرير اسمه `IDR_MAINFRAME`، حيث يحتوى هذا الاسم أيضاً على الصور المستخدمة مع

أزرار الشريط والتي يمكنك معاينتها داخل نافذة محرر الموارد أسفل المجلد **Toolbar** في تبويب عرض الموارد (انظر شكل ١٥-٣).
جدول ١٥-١ قيم خصائص شريط الأدوات

الوصف	القيمة
وضع شريط الأدوات أعلى نافذة الإطار ورسم حد أسفله	CBRS_TOP
وضع شريط الأدوات أسفل نافذة الإطار ورسم حد أعلاه	CBRS_BOTOM
وضع شريط الأدوات يسار نافذة الإطار ورسم حد يمينه	CBRS_LEFT
وضع شريط الأدوات يمين نافذة الإطار ورسم حد يسره	CBRS_RIGHT
وضع شريط الأدوات في أى مكان	CBRS_ALIGN_ANY
يمكن تحريك أكثر من شريط تحكّم في نفس الإطار	CBRS_FLOAT_MULTI
عرض تلميحات شريط الأدوات	CBRS_TOOLTIPS
تحديث نص الرسائل بمجرد تحديث التلميحات	CBRS_FLYBY
إضافة مقبض لتحريك شريط الأدوات	CBRS_GRIPPER
تمكين المستخدم من تغيير حجم شريط الأدوات	CBRS_SIZE_DYNAMIC
عدم تمكين المستخدم من تغيير حجم شريط الأدوات	CBRS_SIZE_FIXED



شكل ١٥-٣ شريط الأدوات القياسي داخل نافذة المحرر

تقييد شريط الأدوات القياسي

بعد إنشاء شريط الأدوات باستخدام الدالة `OnCreate()`، يمكنك تقييده بوضعه في مكان معين داخل النافذة الأم أو تركه حراً مثل باقي النوافذ العادية. قم بفتح الدالة `OnCreate()` داخل نافذة المحرر، تلاحظ استخدام سطر الكود التالي لتقييد شريط الأدوات:

```
m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
```

والذي يعني إمكانية وضع شريط الأدوات في أي جهة داخل إطار النافذة الأم. فإذا أردت أن يكون الشريط حراً، قم بحذف هذا السطر نهائياً. أما إذا أردت وضعه في مكان معين داخل النافذة، فعليك استخدام التوليفة المناسبة من القيم الموضحة بالجدول ١٥-٢ التالي.

جدول ١٥-٢ قيم تقييد شريط الأدوات

الوصف	القيمة
وضع شريط الأدوات في أي مكان داخل النافذة	CBRS_ALIGN_ANY
وضع شريط الأدوات أعلى النافذة	CBRS_ALIGN_TOP
وضع شريط الأدوات أسفل النافذة	CBRS_ALIGN_BOTTOM

الوصف	القيمة
وضع شريط الأدوات يسار النافذة	CBRS_ALIGN_LEFT
وضع شريط الأدوات يمين النافذة	CBRS_ALIGN_RIGHT

فمثلاً، إذا أردت تقييد شريط الأدوات ليكون عند الحافة اليسرى أو الحافة اليمنى، قم بتعديل الدالة `EnableDocking()` لتصبح كما يلي:

```
m_wndToolBar.EnableDocking(CBRS_ALIGN_LEFT |
CBRS_ALIGN_RIGHT);
```

السطر التالي داخل الدالة `OnCreate()` هو:

```
EnableDocking(CBRS_ALIGN_ANY);
```

ويستخدم لتمكين جميع حواف النافذة الأم لاحتواء شريط الأدوات، ويمكنك استخدام نفس التوليفة الموجودة بجدول ١٥-٢ السابق لتحديد حواف معينة.

آخر سطور الدالة `OnCreate()` يتم فيه استدعاء الدالة `DockControlBar()` لوضع شريط الأدوات داخل النافذة الأم كما يلي:

```
DockControlBar(&m_wndToolBar);
```

حيث تحتوى هذه الدالة أساساً على ثلاثة معاملات، الأول عبارة عن مؤشر لشريط التحكم المطلوب، والثاني يعبر عن مكان وضع الشريط ويحتوى على إحدى القيم الموضحة بجدول ١٥-٣ أو القيمة 0 إذا أردت وضع شريط الأدوات في أى جهة من النافذة الأم (وهي القيمة الافتراضية) وهذا المعامل اختياري، أما المعامل الثالث فيمكنك من خلاله تحديد مكان معين للشريط بتمرير مؤشر يحتوى على إحداثيات المكان المطلوب وهو اختياري أيضاً.

جدول ١٥-٣ قيم تقييد شريط الأدوات المستخدمة من قبل الدالة `DockControlBar()`

الوصف	القيمة
وضع شريط الأدوات أعلى النافذة	AFX_IDW_DOCKBAR_TOP
وضع شريط الأدوات أسفل النافذة	AFX_IDW_DOCKBAR_BOTTOM

الوصف	القيمة
وضع شريط الأدوات يسار النافذة	AFX_IDW_DOCKBAR_LEFT
وضع شريط الأدوات يمين النافذة	AFX_IDW_DOCKBAR_RIGHT

أما إذا أردت أن تترك شريط الأدوات حراً، فكل ما عليك أن تقوم باستبدال الدالة `DockControlBar()` بالدالة `FloatControlBar()` التي تحتوي أيضاً على ثلاثة معاملات. المعامل الأول عبارة عن مؤشر لشريط التحكم المطلوب، أما المعامل الثاني فيعبر عن إحداثي الركن الأيسر العلوي من شريط التحكم، بينما المعامل الثالث اختياري ويعبر عن اتجاه شريط التحكم ويكون `CBRS_ALIGN_TOP` (وهي القيمة الافتراضية) إذا أردت أن يكون الشريط رأسي أو `CBRS_ALIGN_LEFT` إذا أردت أن يكون الشريط أفقي.

فمثلاً، لجعل شريط الأدوات حراً ومكانه الابتدائي عند النقطة (50,50)، قم بتغيير سطر الدالة `DockControlBar()` ليصبح كما يلي:

```
FloatControlBar(&m_wndToolBar,CPoint(50,50));
```

وبذلك يتم إظهار شريط الأدوات بمجرد تشغيل البرنامج.

التحكم في شريط الأدوات القياسي

يتم إنشاء شريط الأدوات القياسي أثناء إنشاء التطبيق باستخدام معالج التطبيقات، إلا أنه يمكنك بسهولة تامة إضافة أو حذف أو تعديل أي من الأزرار الموجودة فيه أو حتى حذفه كاملاً إذا لم تكن في حاجة إليه وذلك كما يلي.

إضافة أزرار جديدة لشريط الأدوات

لإضافة زر جديد لشريط الأدوات القياسي، تابع معنا الخطوات الآتية:

1. نشط التبويب `Resource View` من نافذة عمل المشروع إذا لم يكن هو التبويب النشط.
2. قم بتوسيع شجرة الموارد حتى تصل لمجلد أشرطة الأدوات `Toolbar` ثم انقر شريط

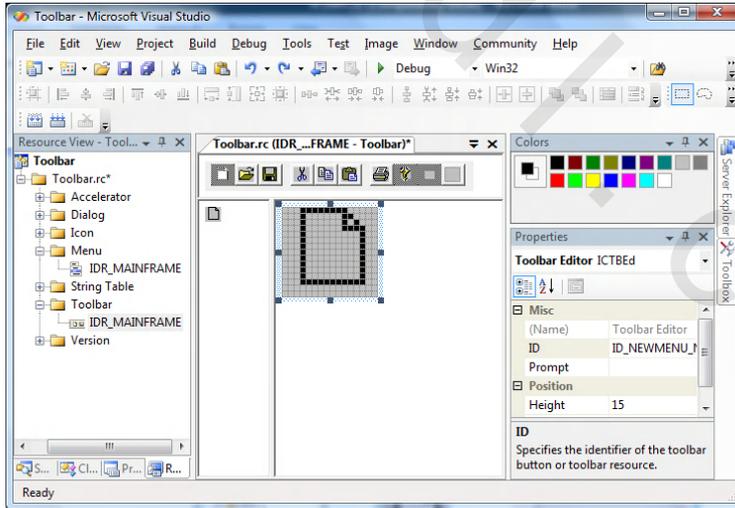
الأدوات المراد تحريره نقرأ مزدوجاً، يظهر قالب شريط الأدوات داخل نافذة المحرر (راجع شكل ١٥-٣).

٣. انقر الزر الخالي يمين شريط الأدوات ثم ابدأ في رسم الزر الجديد مستعيناً بشريط الأدوات وشريط الألوان (انظر شكل ١٥-٤).

٤. بمجرد أن تبدأ في إنشاء الزر، يظهر زر خالي يمين الزر الجديد. فإذا أردت إضافة زر آخر، قم بتكرار الخطوة السابقة.

٥. تأكد من ظهور مربع الخصائص وإلا اضغط الاختصار **Alt+Enter** من لوحة المفاتيح لإظهاره ثم قم بإدخال اسم الزر في الخاصية **ID** وليكن **ID_NEWMENU_NEW** (حيث يجب أن يكون هذا الاسم موجوداً بشريط القوائم) أو قم باختيار اسم موجود مسبقاً من القائمة المنسدلة.

٦. يمكنك كتابة النص الذي سيظهر في شريط المعلومات بمجرد نقر الزر داخل الخاصية **Prompt**. كما يمكنك كتابة التلميح الذي سيظهر عند مرور مؤشر الفأرة على الزر بعد نص شريط المعلومات وذلك بفصلهما بمعرف السطر الجديد **\n**.



شكل ١٥-٤ إضافة زر جديد إلى شريط الأدوات

نقل الأزرار وحذفها وإضافة الفواصل

يمكنك التحكم فى شكل شريط الأدوات بنقل أحد الأزرار إلى مكان آخر أو حذفه أو إضافة فاصل بين زرین وذلك كما يلي:

- لنقل زر فى شريط الأدوات إلى مكان ما، تابع معنا الخطوات الآتية:
 ١. قم باختيار الزر المراد نقله.
 ٢. قم بسحب الزر إلى المكان الذى تريد وضعه فيه.
 ٣. قم بتحرير زر الفأرة، تلاحظ نقل الزر إلى المكان الجديد.
- لإضافة فاصل بين زرین أو حذف الفاصل الموجود، تابع معنا الخطوات الآتية:
 ١. قم باختيار الزر.
 ٢. قم بسحب الزر بعيداً قليلاً يميناً أو يساراً عن الزر المجاور.
 ٣. قم بتحرير زر الفأرة.
- لحذف زر من شريط الأدوات، تابع معنا الخطوات الآتية:
 ١. قم باختيار الزر المراد حذفه.
 ٢. قم بسحب الزر إلى أسفل داخل النافذة.
 ٣. قم بتحرير زر الفأرة، تلاحظ حذف الزر.

احتواء أوامر شريط الأدوات

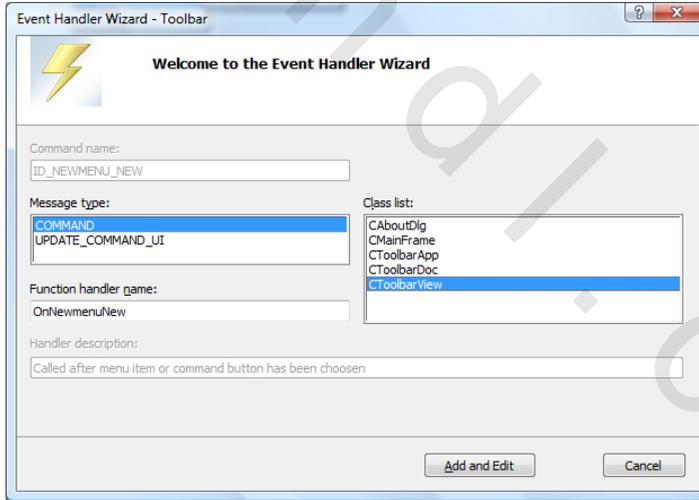
يمكنك التعامل مع عناصر شريط الأدوات بنفس الطريقة التى كنت تتعامل بها مع أزرار التحكم داخل المربعات الحوارية. فحينما يقوم المستخدم بنقر أحد أزرار شريط الأدوات، يقوم Windows بإرسال رسالة من النوع WM_COMMAND لنافذة التطبيق التى تحتوى على الشريط لاستدعاء دالة احتواء الزر إن وجدت. لذا للتعامل مع أحد الأزرار، استخدم مربع الخصائص لإنشاء دالة احتواء العنصر من النوع WM_COMMAND.

إضافة دالة احتواء الزر الجديد

بعد إجراء الخطوات السابقة، سيظهر الزر الجديد داخل شريط الأدوات بمجرد تشغيل التطبيق، إلا أنه سيكون معطلاً لأننا لم نقم حتى الآن بتعيين الكود الخاص به. ولأنه دائماً ما يعبر الزر الموجود بشريط الأدوات عن أحد الخيارات الموجودة بشريط القوائم، لذا قم أولاً بإضافة عنصر جديد إلى إحدى قوائم شريط قوائم التطبيق على أن يأخذ نفس اسم الزر الذي قمنا بإضافته (`ID_NEWMENU_NEW` في هذه الحالة) كما تعلمت في الفصل السابق.

ولإنشاء دالة احتواء الزر وعنصر القائمة المصاحب، تابع معنا الخطوات الآتية:

1. انقر عنصر القائمة المصاحب للزر الجديد بزر الفأرة الأيمن ثم اختر **Add Event Handler** من القائمة الموضعية، تظهر نافذة معالجة إضافة دالة احتواء حدث عنصر القائمة (انظر شكل ١٥-٥).



شكل ١٥-٥ إضافة دالة احتواء عنصر القائمة

2. من مربع السرد **Class List**، اختر اسم التصنيف **CToolbarView**.
3. تأكد من اختيار **COMMAND** من مربع السرد **Message Type**.

٤. من مربع النص **Function handler name**، قم بتعديل اسم الدالة أو اترك الاسم الافتراضى للدالة كما هو.
٥. انقر زر **Add and Edit** للبدء فى كتابة كود الدالة.
٦. قم بإضافة الكود الذى تريد تنفيذه بمجرد اختيار المستخدم لعنصر القائمة. فإذا أردت إظهار رسالة **New Button in my toolbar** على سبيل المثال، قم بتعديل كود الدالة كما يلى:

```
void CToolBarView::OnNew()
{
    AfxMessageBox("New Button in my toolbar");
}
```

إضافة دالة واجهة المستخدم

- تختص دالة واجهة المستخدم بشكل وغط وسلوك كل عنصر من عناصر شريط الأدوات. لإضافة دالة واجهة الزر الجديد، تابع معنا الخطوات الآتية:
١. انقر عنصر القائمة المصاحب للزر الجديد بشريط الأدوات بزر الفأرة الأيمن ثم اختر **Add Event Handler** من القائمة الموضعية، تظهر نافذة معالج إضافة دالة احتواء حدث عنصر القائمة (راجع شكل ١٥-٥).
 ٢. من مربع السرد **Class List**، اختر اسم التصنيف **CToolbarView**.
 ٣. اختر **UPDATE_COMMAND_UI** من مربع السرد **Message Type**.
 ٤. من مربع النص **Function handler name**، قم بتعديل اسم الدالة أو اترك الاسم الافتراضى للدالة كما هو ثم انقر زر **Ok** لإضافة الدالة الجديدة.
 ٥. انقر زر **Add and Edit** للبدء فى كتابة كود الدالة.
 ٦. قم بإضافة الكود الذى تريد تنفيذه لإضفاء سمات معينة على القائمة بمجرد إظهارها. فإذا أردت مثلاً أن يظهر الزر بارزاً، قم بتعديل كود الدالة كما يلى:

```
void CToolBarView::OnUpdateNew(CCmdUI *pCmdUI)
{
    pCmdUI->SetRadio(FALSE);
}
```

إنشاء شريط أدوات جديد

في التطبيقات الصغيرة، نقوم غالباً بتعديل شريط الأدوات القياسي بإضافة بعض الأزرار أو حذف البعض الآخر. أما في التطبيقات الكبيرة، فنحتاج غالباً لإضافة العديد من أشرطة الأدوات الجديدة لتلبية كافة احتياجاتك. كما تحتاج إلى تمكين المستخدم من إظهار أو إخفاء أي من هذه الأشرطة. لتوضيح ذلك، تابع معنا الأجزاء التالية.

إضافة شريط الأدوات الجديد

لإضافة شريط أدوات جديد إلى التطبيق الحالي، تابع معنا الخطوات الآتية:

١. نشط التبويب **Resource View** من نافذة عمل المشروع إذا لم يكن هو التبويب النشط.
٢. انقر عنصر المستوى الأعلى بزر الفأرة الأيمن ثم اختر **Add Resource** من القائمة الموضوعية، يظهر المربع الحوارى **Add Resource**.
٣. اختر **Toolbar** من قائمة أنواع الموارد الجديدة.
٤. انقر زر **New**، يظهر قالب شريط الأدوات الجديد داخل نافذة المحرر.
٥. بدلاً من ذلك، قم بتوسيع موارد المشروع ثم انقر المجلد **Toolbar** بزر الفأرة الأيمن واختر **Insert Toolbar** من القائمة الموضوعية.
٦. انقر قالب شريط الأدوات الجديد بزر الفأرة الأيمن ثم اختر **Properties** من القائمة الموضوعية لإظهار نافذة الخصائص وإلا اكتفى بنقر القالب إذا كان مربع الخصائص ظاهراً بالفعل.
٧. يمكنك الآن تغيير الاسم الافتراضى للمربع الحوارى من **IDD_TOOLBAR1** إلى اسم أكثر تعبيراً وليكن **IDD_NEWTOOLBAR** وذلك من خلال الخاصية **ID**.

٨. يمكنك الآن إضافة الأزرار إلى شريط الأدوات الجديد بنفس الطريقة السابقة. قم بإضافة زر واحد على الأقل حتى تتمكن من استخدام الشريط الجديد.

إضافة شريط الأدوات لنافذة التطبيق

بعد أن قمت بإضافة شريط الأدوات الجديد إلى التطبيق، يجب أن تقوم بإضافة الكود اللازم لتحميل هذا الشريط داخل إطار النافذة الأساسية للتطبيق بمجرد تشغيله. قم بفتح ملف **MainFrm.h** داخل نافذة المحرر ثم قم بإضافة تعريف شريط الأدوات الجديد كما يلي:

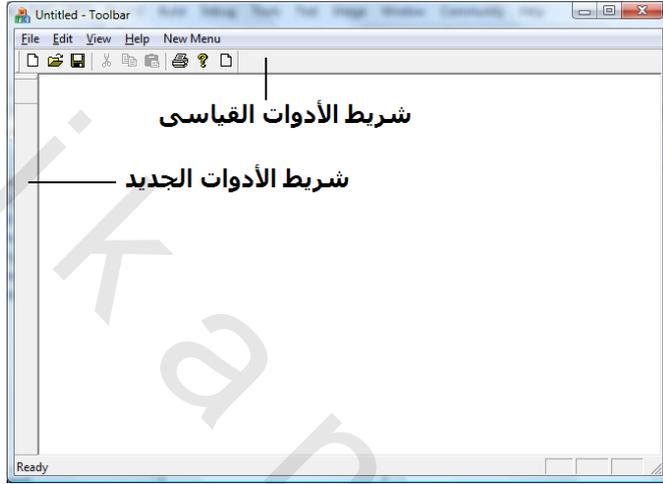
```
protected: // control bar embedded members
    CStatusBar m_wndStatusBar;
    CToolBar m_wndToolBar;
    CToolBar m_wndNewToolBar; // شريط الأدوات الجديد

بعد ذلك يجب أن تقوم بإضافة كود تحميل الشريط الجديد داخل الدالة OnCreate() بنفس الطريقة المتبعة مع شريط الأدوات القياسي كما يلي:
```

```
if (!m_wndNewToolBar.CreateEx(this, TBSTYLE_FLAT,
    WS_CHILD | WS_VISIBLE | CBRS_LEFT
    | CBRS_GRIPPER | CBRS_TOOLTIPS |
    CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ||
    !m_wndNewToolBar.LoadToolBar(IDR_NEWTOOLBAR))
{
    TRACE0("Failed to create toolbar\n");
    return -1; // fail to create
}

m_wndNewToolBar.EnableDocking(CBRS_ALIGN_ANY);
DockControlBar(&m_wndNewToolBar);
```

وبذلك سيتم إظهار شريط أدوات جديد في الجزء الأيسر من إطار النافذة، بينما يظهر شريط الأدوات القياسي المعدل أعلى إطار النافذة (انظر شكل ١٥-٦). كما يمكنك الآن إنشاء دالة احتواء ودالة واجهة شريط الأدوات الجديد كما سبق.



شكل ١٥-٦ شريط الأدوات الجديد داخل إطار النافذة.

إخفاء شريط الأدوات وإظهاره

يمكنك إخفاء شريط الأدوات أو إظهاره باستخدام الدالة `ShowControlBar()` التي تحتوى على ثلاثة معاملات. المعامل الأول عبارة عن مؤشر لشريط الأدوات الذي تريد إظهاره أو إخفاءه، والثاني يكون `TRUE` إذا أردت إظهار الشريط أو `FALSE` إذا أردت إخفاءه، أما المعامل الثالث فيكون `TRUE` إذا أردت إظهار الشريط أو إخفاءه مباشرةً أو `FALSE` إذا أردت وجود تأخير في عملية الإظهار أو الإخفاء. لتوضيح ذلك، قم بإضافة عنصر جديد لقائمة `Edit` باسم `ID_EDIT_SHOWNEW` وعنوانه `Show New` ثم قم بإنشاء دالة احتواء العنصر `OnShownew()` ودالة واجهة المستخدم `OnUpdateShownew()` (مع استخدام التصنيف `CMainFrame` في هذه الحالة). لأننا سنقوم باستخدام هذا الخيار للتبديل بين إظهار شريط الأدوات الجديد وإخفاءه ومن ثمَّ

وضع علامة اختيار بجوار عنصر القائمة في حالة ظهور الشريط وحذف هذه العلامة في حالة اختفائه، قم بتعديل كود كلٍ من الدالتين كما يلي:

```

1. void CMainFrame::OnShownew()
2. {
3.     m_bVisible =! m_bVisible;
4.     ShowControlBar(&m_wndNewToolBar,m_bVisible,FALSE);
5. }

6. void CMainFrame::OnUpdateShownew(CCmdUI* pCmdUI)
7. {
8.     pCmdUI->Enable();
9.     pCmdUI->SetCheck(m_bVisible);
10. }
    
```

وعن هذا الكود، نوضح ما يلي:

- في السطر رقم ٣ يتم عكس قيمة المتغير المنطقى `m_bVisible` بمجرد اختيار المستخدم للخيار `Show New` من قائمة `Edit` (لا تنسى تعريف المتغير المنطقى داخل التصنيف `CMainFrame`).
 - في السطر رقم ٤ يتم استدعاء الدالة `ShowControlBar()` لإظهار شريط الأدوات أو إخفائه تبعاً لقيمة المتغير المنطقى فى المعامل الثانى للدالة.
 - فى السطر رقم ٨ يتم استدعاء الدالة `Enable()` لتنشيط شريط الأدوات.
 - فى السطر رقم ٩ تم استدعاء الدالة `SetCheck()` لوضع علامة اختيار بجانب الخيار `Show New` أو حذفها تبعاً لقيمة المتغير المنطقى `m_bVisible`.
- قم ببناء التطبيق وتنفيذه، ثم انقر الخيار الجديد للتبديل بين إظهار شريط الأدوات الجديد وإخفائه.

تخزين وتحميل موضع شريط الأدوات

يمكنك تخزين وتحميل مواضع أشرطة الأدوات الموجودة بتطبيقك باستدعاء الدوال `SaveBarState()` و `LoadBarState()` مرة واحدة بتمرير معامل واحد يحتوى على

قيمة نصية فريدة. فمثلاً، يمكنك تخزين مواضع أشرطة الأدوات بمجرد إنهاء التطبيق باستدعاء الدالة `SaveBarState()` في دالة تدمير إطار النافذة `OnDestroy()` المشتقة من الرسالة `WM_DESTROY` هكذا:

```
void CMainFrame::OnDestroy()
{
    SaveBarState("MyToolbarPos");
    CFrameWnd::OnDestroy();
}
```

كما يمكنك استعادة هذه الإعدادات بمجرد بدء التطبيق مرة أخرى باستدعاء الدالة `LoadBarState()` في نهاية الدالة `OnCreate()` هكذا:

```
LoadBarState("MyToolbarPos");
```

العمل مع شريط المعلومات Status bar

يمكنك إضافة كافة المعلومات التي تقوم بتوضيح عمل أدوات التحكم الموجودة بالتطبيق إلى شريط المعلومات الذي يقع دائماً أسفل نافذة إطار التطبيق، ويحتوى أساساً على ثلاثة مربعات صغيرة لتوضيح حالة المفاتيح `CAPS Lock` و `NUM Lock` و `SCRL Lock`. يمكنك إنشاء مربعات أخرى داخل شريط المعلومات والوصول إليها باستخدام دوال التصنيف `CStatusBar` المشتق من التصنيف `CControlBar`.

كيف يتم إنشاء شريط المعلومات القياسي؟

يتشابه كود إنشاء شريط المعلومات القياسي مع الكود المنشأ من قبل المعالج لإنشاء شريط الأدوات، حيث يتم تعريف عنصر ينتمى للتصنيف `CStatusBar` داخل ملف `MainFrm.h` كما يلي:

```
CStatusBar m_wndStatusBar;
```

يتم بعد ذلك إنشاء نافذة شريط المعلومات نفسها باستدعاء الدالة `Create()` التي تنتمى للتصنيف `CStatusBar` داخل الدالة `OnCreate()`، حيث تحتوى الدالة `Create()` على ثلاثة معاملات. المعامل الأول إجبارى ويحتوى على مؤشر للنافذة الأم، بينما يتيح لك المعامل الثانى الاختيارى تحديد مكان وخصائص شريط المعلومات باختيار توليفة من القيم

الموضحة بمجدول ١٥-٤، والقيم الافتراضية لهذا المعامل هي **WS_CHILD**, **WS_VISIBLE**, **CBRS_BOTTOM**. أما المعامل الثالث والأخير؛ وهو اختياري أيضاً، فيتيح لك تعيين معرف لشريط المعلومات، والقيمة الافتراضية لهذا المعامل هي **.AFX_IDW_STATUS_BAR**.

جدول ١٥-٤ تعيين موضع شريط المعلومات

الوصف	القيمة
وضع شريط الأدوات الحوارى أعلى نافذة الإطار	CBRS_TOP
وضع شريط الأدوات الحوارى أسفل نافذة الإطار	CBRS_BOTOM
وضع شريط الأدوات الحوارى يسار نافذة الإطار	CBRS_LEFT
وضع شريط الأدوات الحوارى يمين نافذة الإطار	CBRS_RIGHT
عدم إعادة وضع شريط الأدوات الحوارى في حالة إعادة تحجيم النافذة	CBRS_NOALIGN

قم بفتح الدالة **OnCreate()**، تلاحظ ظهور كود إنشاء شريط المعلومات أسفل كود إنشاء شريط الأدوات كما يلي:

```
if (!m_wndStatusBar.Create(this) ||
    !m_wndStatusBar.SetIndicators(indicators,
    sizeof(indicators)/sizeof(UINT)))
{
    TRACE0("Failed to create status bar\n");
    return -1;    // fail to create
}

```

وعن هذا الكود، نوضح ما يلي:

- تم استدعاء الدالة **Create()** لإنشاء نافذة شريط المعلومات بتمرير معامل واحد فقط وهو كلمة **this** دلالة على النافذة الأم الرئيسية.
- تم استدعاء الدالة **SetIndicators()** لإنشاء مربعات التوضيح الموجودة بشريط المعلومات حيث تحتوى هذه الدالة على معاملين كما يلي:

- يحتوى الأول على مؤشر لمصفوفة من النوع UINT تحتوى على أسماء مربعات التوضيح التى ستظهر بشريط المعلومات، وقد تم تمرير المصفوفة `indicators` التى تتكون مما يلى (داخل الملف `MainFrm.cpp`):

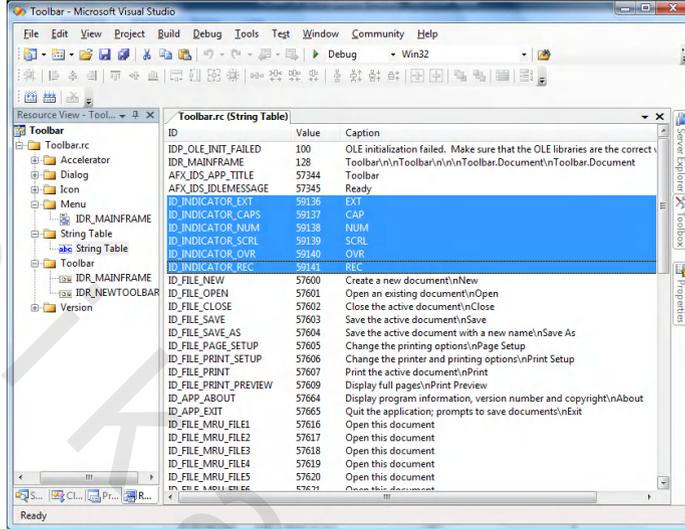
```
static UINT indicators[] =
{
    ID_SEPARATOR,          // status line indicator
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM,
    ID_INDICATOR_SCRL,
};
```

حيث تحتوى هذه المصفوفة على أربعة عناصر. يحتوى العنصر الأول منها على الجزء الأكبر من شريط المعلومات الذى تظهر به التنبيهات والتلميحات بمجرد مرور المؤشر على خيار من خيارات القوائم أو زر من أزرار أشرطة الأدوات. أما العناصر الأخرى فتحتوى على مربعات التوضيح الصغيرة التى تقوم بإظهار حالة المفاتيح `Caps Lock` و `Num Lock` و `Scroll Lock` على الترتيب.

- يحتوى المعامل الثانى للدالة `SetIndicators()` على عدد مربعات التوضيح الموجودة بشريط المعلومات بقسمة حجم المصفوفة على حجم كل عنصر من عناصرها.

إضافة المؤشرات والفواصل Indicator and separators

تظهر أسماء عناصر المصفوفة السابقة داخل مورد جدول النصوص كما فى شكل ١٥-٧ ومنه يمكنك إضافة مؤشرات (`Indicators`) أخرى أو حذف إحدى المؤشرات الموجودة أو تعديلها.



شكل ١٥-٧ مؤشرات شريط المعلومات داخل جدول النصوص

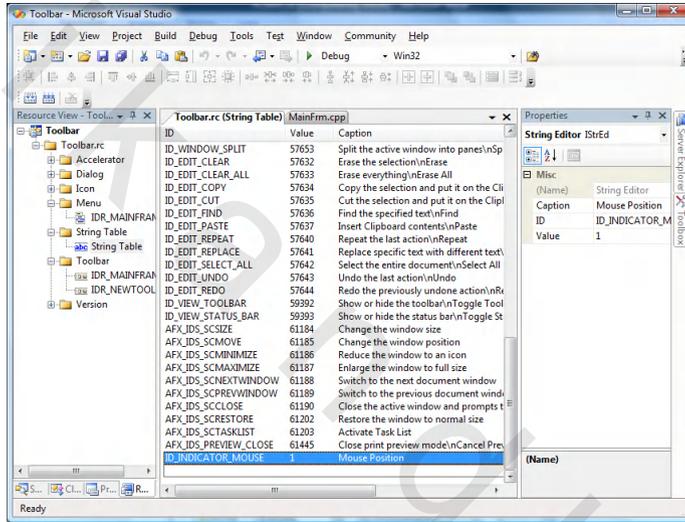
لإضافة مؤشرات Indicators أخرى لشريط المعلومات، يجب أن تقوم أولاً بإضافة معرف فريد للمؤشر الجديد داخل جدول النصوص ثم تقوم بعد ذلك بإنشاء دالة واجهة المستخدم لهذا المؤشر يدوياً كي تستطيع تمكين أو تعطيل هذا المؤشر أو تغيير نصه أثناء التشغيل، حيث لا يمكنك أداء ذلك باستخدام المعالج. للتعرف على طريقة إضافة مؤشر جديد لشريط المعلومات، استخدم المعالج لإنشاء تطبيق أحادي الوثيقة باسم StatusMouse. سنقوم بإضافة مؤشر جديد يحتوي على الإحداثيات الحالية للمؤشر الفأرة.

لإضافة معرف فريد للمؤشر الجديد داخل الجدول النصي، تابع معنا الخطوات الآتية:

١. نشط التبويب Resource View إذا لم يكن هو التبويب النشط.
٢. قم بفتح المجلد String Table ثم انقر مورد جدول النصوص String Table
٣. نقراً مزدوجاً، يظهر الجدول داخل نافذة المحرر.
٣. قم بالتنقل داخل الجدول حتى تصل إلى آخر عناصر جزء المؤشرات (راجع شكل ١٥-٧).

٤. انقر العنصر الأخير لاختياره ثم اضغط مفتاح **Insert** من لوحة المفاتيح، ومن مربع الخصائص قم بإدخال نص العنصر الجديد داخل الخاصية **Caption** وليكن **Mouse Position** (انظر شكل ١٥-٨).

٥. قم بتغيير معرف العنصر الجديد داخل الخاصية **ID** إلى اسم معبر وليكن **.ID_INDICATOR_MOUSE**.

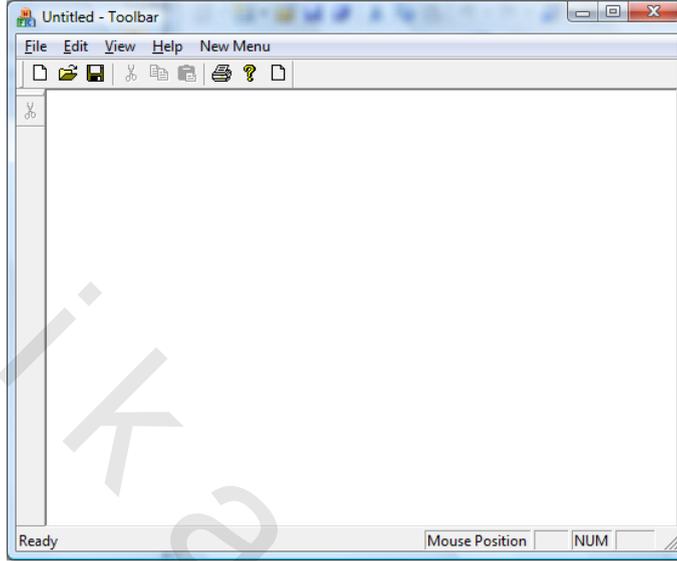


شكل ١٥-٨ إضافة المؤشر الجديد لجدول النصوص

الخطوة التالية هي إضافة المؤشر الجديد إلى مجموعة مبيّنات المصفوفة **indicators** بالملف **MainFRm.cpp** كما يلي:

```
static UINT indicators[] =
{
    ID_SEPARATOR, // status line indicator
    ID_INDICATOR_MOUSE,
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM,
    ID_INDICATOR_SCROLL,
};
```

إذا قمت الآن ببناء التطبيق وتنفيذه، ستلاحظ ظهور مربع جديد داخل شريط المعلومات ويحتوي على النص الذي قمت بتعيينه (انظر شكل ١٥-٩).



شكل ١٥-٩ مربع المؤشر الجديد داخل شريط المعلومات

يمكنك أيضاً إضافة فواصل جديدة بين المؤشرات بإدخال المعرف `ID_SEPARATOR` بين مربعي البيان اللذين تريد وضع الفاصل بينهما. فمثلاً إذا أردت إضافة فاصل بين المؤشر الجديد ومؤشر المفتاح `Caps Lock`، قم بتعديل كود المصفوفة `indicators` كما يلي:

```
static UINT indicators[] =
{
    ID_SEPARATOR,           // status line indicator
    ID_INDICATOR_MOUSE,
    ID_SEPARATOR, //          الفاصل الجديد
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM,
    ID_INDICATOR_SCRL,
};
```

قم ببناء التطبيق وتنفيذه، تلاحظ ظهور مؤشر جديد بين كل من `Mouse Position` والمؤشر `Cap` (انظر شكل ١٥-١٠).



شكل ١٥-١٠ إضافة فاصل جديد إلى شريط المعلومات

التحكم في خصائص المؤشر

يمكنك استخدام دوال التصنيف **CStatusBar** للتحكم في خصائص المؤشرات كتغيير الحجم أو الشكل أو نص المؤشر نفسه. وهذه الدوال هي **SetPanelInfo()** و **SetPaneStyle()** و **SetPaneText()** على الترتيب ويتم تمرير رقم المؤشر داخل المصفوفة في جميع هذه الدوال. ولا يجذب تمرير هذا الرقم صراحةً إلى الدالة، لأنك لو قمت بإضافة عنصر جديد إلى المصفوفة، فربما تكون في حاجة إلى تعديل الكثير من الكود. ومن حسن الحظ وجود الدالة **CommandToIndex()** التي تقوم بإرجاع رقم العنصر. فمثلاً الكود التالي يقوم بتخزين القيمة 1 داخل المتغير **nIndex** هكذا:

```
int nIndex = m_wndStatusBar.CommandToIndex  
(ID_INDICATOR_MOUSE);
```

تغيير نص المؤشر

يمكنك تغيير نص أحد المؤشرات باستخدام الدالة **SetPaneText()** التي تحتوى على ثلاثة معاملات. يحتوى المعامل الأول على رقم المؤشر داخل المصفوفة **indicators**، أما المعامل الثاني فيحتوى على النص الجديد، بينما يحتوى المعامل الثالث الاختياري على قيمة منطقية قد تكون **TRUE** إذا أردت تغيير النص بمجرد استدعاء الدالة (وهي القيمة

الافتراضية) أو FALSE إذا أردت تغيير النص مع أول عملية إعادة طلاء للنافذة. فمثلاً، لتغيير نص المؤشر المخزن داخل المتغير nIndex، قم بإدخال الكود التالى:

```
m_wndStatusBar.SetPaneText(nIndex,"My New Text");
```

يمكنك أيضاً استرجاع النص الحالى للمؤشر باستخدام الدالة GetPaneText() التى تقوم بإرجاع قيمة من النوع CString تحتوى على النص المطلوب.

تغيير نمط المؤشر

يمكنك تغيير نمط أحد المؤشرات باستدعاء الدالة SetPaneStyle() التى تحتوى على معاملين. يحتوى المعامل الأول على رقم المؤشر داخل المصفوفة Indicators، أما المعامل الثانى فيحتوى على النمط الجديد باختيار أحد الأنماط الموضحة بالجدول ١٥-٥ التالى.

جدول ١٥-٥ قيم أنماط المؤشر

الوصف	القيمة
النمط العادى للمؤشر	SBPS_NORMAL
يظهر المؤشر بارزاً للخارج	SBPS_POPOUT
عدم ظهور نص المؤشر	SBPS_DISABLED
تمدد المؤشر ليملأ الجزء الخالى من شريط المعلومات	SBPS_STRETCH
عدم رسم إطار ثلاثى الأبعاد حول المؤشر	SBPS_NOBORDERS

فإذا أردت تغيير نمط المؤشر Mouse Position من النمط الحالى إلى النمط البارز، قم باستخدام الكود التالى:

```
int nIndex = m_wndStatusBar.CommandToIndex
(ID_INDICATOR_MOUSE);
m_wndStatusBar.SetPaneStyle(nIndex,SBPS_POPOUT);
```

يمكنك أيضاً استخدام الدالة GetPaneStyle() للحصول على النمط الحالى لأحد المؤشرات الموجودة بشريط المعلومات حيث تقوم بإرجاع قيمة من النوع UINT.

تغيير حجم المؤشر

يمكنك تغيير حجم ونمط ومعرف المؤشر مرة واحدة باستخدام الدالة `SetPanelInfo()` التي تحتوى على أربعة معاملات. يحتوى المعامل الأول على رقم المؤشر داخل المصفوفة `Indicators`، أما المعامل الثانى فيحتوى على معرف المؤشر الجديد (أو المعرف القديم إذا أردت عدم تغييره)، بينما يحتوى المعامل الثالث على نمط المؤشر باختيار أحد الأنماط الموضحة بالجدول ١٥-٥ السابق (أو الإبقاء على النمط القديم باستدعاء الدالة `(GetPaneStyle())`، أما المعامل الرابع والأخير للدالة فيتيح لك تغيير العرض الافتراضى للمؤشر بتمرير قيمة صحيحة تعبر عن العرض الجديد (يتم إهمال هذه القيمة بالطبع فى حالة اختيار النمط `SBPS_STRETCH`). فمثلاً، إذا أردت تغيير عرض المؤشر `Mouse Position` إلى ١٠٠ نقطة، قم باستخدام الكود التالى:

```
int nIndex = m_wndStatusBar.CommandToIndex
                (ID_INDICATOR_MOUSE);
m_wndStatusBar.SetPanelInfo(nIndex, ID_INDICATOR_MOUSE,
GetPaneStyle,100);
```

سنقوم فيما يلى بتطوير المثال الذى بين أيدينا لإظهار إحداثى مؤشر الفأرة بدلاً من العبارة `Mouse Position`، لذا تابع معنا الخطوات الآتية:

١. قم بإضافة الدالة `SetMousePosText(CString strText)` من النوع `void`

إلى التصنيف `CMainFrame`.

٢. قم بتعديل كود الدالة كما يلى:

```
1. void CMainFrame::SetMousePosText(CString strText)
2. {
3.     int nIndex = m_wndStatusBar.CommandToIndex
                        (ID_INDICATOR_MOUSE);
4.     m_wndStatusBar.SetPaneText(nIndex,strText);
5.     CWindowDC dc(&m_wndStatusBar);
6.     CSize sizeText = dc.GetTextExtent(strText);
7.     m_wndStatusBar.SetPanelInfo(nIndex,
8.     ID_INDICATOR_MOUSE,SBPS_NORMAL,sizeText.cx);
9. }
```

وعن هذا الكود، نوضح ما يلي:

- فى السطر رقم ٣ تم تخزين ترتيب العنصر **Mouse Position** داخل المتغير **.nIndex**.
- فى السطر رقم ٤ تم استدعاء الدالة **SetPaneText()** لتغيير نص العنصر إلى القيمة المدخلة للدالة.
- فى السطور ٥ و ٦ تم استخدام بيئة العنصر والدالة **GetTextExtent()** للحصول على عرض نص العنصر.
- فى السطر رقم ٧ تم استدعاء الدالة **SetPanelInfo()** لتغيير حجم المؤشر تبعاً لعرض النص فى الخطوة السابقة.
- ٣. قم بإضافة الدالة **OnMouseMove()** لتصنيف **CToolBarView** من خلال رسائل مربع الخصائص.
- ٤. قم بتعديل كود الدالة كما يلي:

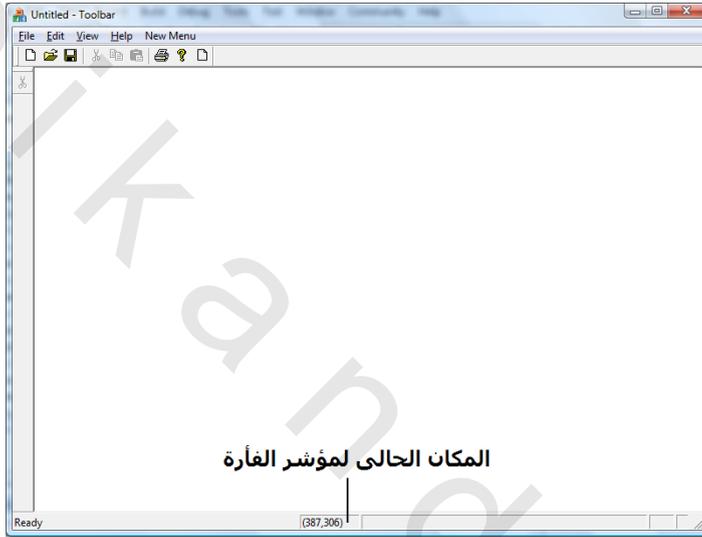
```

1. #include "MainFrm.h"
2. void CStatusMouseView::OnMouseMove(UINT nFlags,
   CPoint point)
3. {
4.     CString strMousePosition;
5.     strMousePosition.Format("(%d,%d)",point.x,point.y);
6.     CMainFrame* pMainFrame = (CMainFrame*)
7.     GetParentFrame();
8.     pMainFrame->SetMousePosText(strMousePosition);
9.     CView::OnMouseMove(nFlags, point);
10. }
    
```

وعن هذا الكود، نوضح ما يلي:

- يتم استدعاء هذه الدالة مع كل حركة لمؤشر الفأرة.
- فى السطر رقم ٤ تم إنشاء المتغير **strMousePosition** كى يحتوى على الإحداثى الحالى لمؤشر الفأرة.

- في السطر رقم ٨ يتم استدعاء الدالة `SetMousePosText()` بتمرير إحداثي مؤشر الفأرة المخزن داخل المتغير `strMousePosition`.
والآن قم ببناء التطبيق وتنفيذه، تلاحظ تغير نص وعرض المؤشر `Mouse Position` مع كل حركة لمؤشر الفأرة (انظر شكل ١٥-١١).



شكل ١٥-١١ عرض الإحداثي الحالي لمؤشر الفأرة داخل شريط المعلومات



obeykandi.com

الباب الخامس العمل مع الرسوم

١٦ . العمل مع الصور والرسوم.

١٧ . الرسم داخل بيئة العنصر *Device Context*.

١٨ . استخدام أقلام الرسم وفرش الألوان.

١٩ . العمل مع الخطوط.