الفصل التاسع
المزيد عن أدوات التحكم

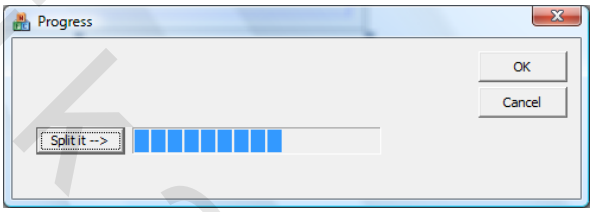
أثناء العمل مع تطبيقاتك، تحتاج غالباً إلى تعيين قيمة من مدى معين من القيم، أو تعيين بداية ونهاية مدى معين، أو إعطاء المستخدم خلفيه عن إعدادات المدى الحالى أو إظهار مؤشر لإنجاز الذى يتم أثناء إجراء عملية أو مهمة معينة طويلة نوعاً ما. يقوم المدى بتمثيل أنواع مختلفة من البيانات مثل البيانات الرقمية أو التاريخ أو الوقت أو الأمتار والأميال. لأداء هذه المهام، يوجد داخل بيئة تطوير **Visual C++ 2005** *بعض أدوات التحكم التى سنتعرض لها بالتفصيل من خلال هذا الفصل.*

بانتهاء هذا الفصل ستتعرف على:

- *استخدام أداة التقدم لعرض مدى إنجاز إحدى المهام*
- *استخدام المنزلق وشريط التمرير لتعيين مكان معين أو اختيار مدى معين .*

استخدام أداة التقدم

تستخدم أداه التقدم **Progress Control** أساساً لإخبار المستخدم بمدى تقدم تنفيذ عملية أو مهمة معينة طويلةً نسبياً ويتم تمثيلها بخط أزرق سميك مصمت أو منقَّط داخل مستطيل يزداد كلما زاد تقدم العملية (انظر شكل ٩-١). كما تستخدم أداة التقدم أيضاً لتمثيل قيمة داخل مدى معين من القيم (انظر شكل ٩-٢).



شكل ٩-١ استخدام أداة التقدم لتمثيل مدى تقدم عملية معينة

شكل ٩-٢ استخدام أداة التقدم لتمثيل قيمة داخل مدى معين من القيم

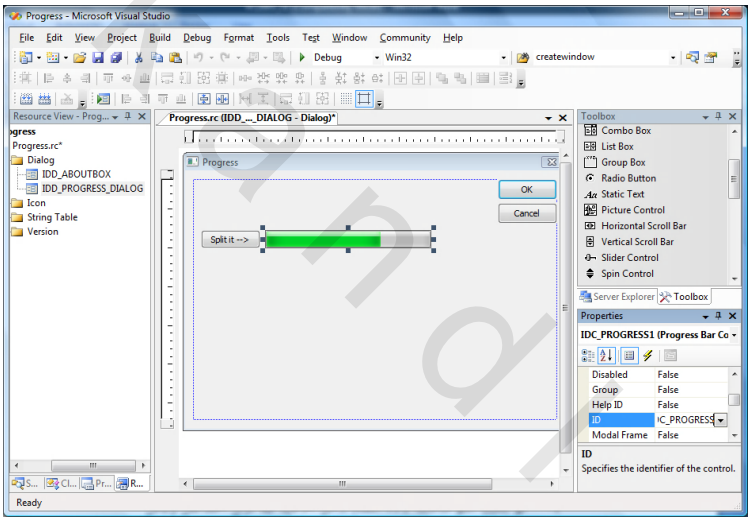
إضافة أداة التقدم إلى المربع الحوارى

لا تختلف طريقة إضافة أداة التقدم إلى المربع الحوارى كثيراً عن إضافة أدوات التحكم الأخرى التى تعاملنا معها فى الفصول السابقة. قم بإنشاء مشروع حوارى جديد باسم **Progress** ثم قم بإضافة زر جديد إلى المربع الحوارى

IDD_PROGRESS_DIALOG باسم IDC_STEPIT مع تغيير عنوانه إلى Split it-> (راجع شكل ٩-١). لإضافة أداة تقدم إلى المربع الحوارى، تابع معنا الخطوات الآتية:

١. انقر رمز أداة التقدم Progress Control من مربع الأدوات، يظهر مؤشر الفأرة مصحوباً بعلامة +.

٢. انقر بزر الفارة الأيسر فى المكان الذى تريد وضع أداة التقدم فيه داخل المربع الحوارى وليكن إلى اليمين من زر الأمر الذى قمت بإضافته منذ قليل، تظهر أداة التقدم داخل المربع الحوارى (انظر شكل ٩-٣).



شكل ٩-٣ إضافة أداة التقدم إلى المربع الحوارى

٣. انقر أداة التقدم بزر الفأرة الأيمن ثم اختر Properties من القائمة الموضعية لإظهار نافذة مربع الخصائص إذا لم تكن ظاهرة بالفعل.

٤. قم بتعيين اسم مناسب لأداة التقدم داخل الخاصية ID بمربع الخصائص وليكن IDC_PROGRESS.

٥. قم بتغير إعدادات شكل أداة التقدم كما يروق لك وذلك من خلال مربع الخصائص كما يلى:

- قم بتخصيص القيمة True للخاصية Border إذا أردت وضع إطار أسود رفيع

حول أداة التقدم.

- قم بتخصيص القيمة **True** للخاصية **Vertical** إذا أردت رسم أداة التقدم رأسياً وليس أفقياً (راجع شكل ٩-٢).
- قم بتخصيص القيمة **True** للخاصية **Smooth** إذا أردت رسم قضيب التقدم فى صورة خط مستمر وليس متقطع (راجع شكل ٩-٢).

ربط أداة التقدم بمتغير

كى تستطيع التعامل مع أداة التقدم، يجب أن تقوم بتعيين متغير عضو فى التصنيف **CProgressCtrl** الذى يحتوى على وظائف أداة التقدم. لأداء ذلك، تابع معنا الخطوات الآتية:

١. نشط التبويب **Class View** من نافذة عمل المشروع.
٢. انقر التصنيف **CProgressDlg** بزر الفأرة الأيمن ثم اختر **Add⮞Add Variable** من القائمة الموضعية، يظهر معالج إضافة متغير جديد.
٣. قم بتنشيط مربع الاختيار **Control variable** لأننا نرغب فى إنشاء متغير مرتبط بإحدى أدوات التحكم الموجودة بالمربع الحوارى.
٤. اختر أداة التقدم **IDC_PROGRESS** من مربع السرد **Control ID**.
٥. لاحظ اختيار **Control** فى مربع السرد والتحرير **Category** واختيار **CProgressCtrl** فى مربع **Variable Type** وهى الخيارات الوحيدة المتاحة مع أداة التقدم.
٦. اكتب اسم المتغير فى خانة **Variable name** وليكن **m_myProgress** واختر درجة المتغير من مربع السرد **Access** وهو **Public** فى هذه الحالة.
٧. انقر زر **Finish** لإغلاق نافذة المعالج.

وبذلك يمكنك باستخدام المتغير الجديد التحكم فى خصائص عرض أداة التقدم من خلال ثلاثة معاملات هى:

- مدى أداة التقدم، ويتم من خلاله تحديد القيمة الرقمية التى عندها تكون الأداة فارغة (0%full) والقيمة الرقمية التى تكون عندها الأداة ممتلئة (100%full).
- المكان الحالى لقضيب أداة التقدم والذى يعبر عن الجزء الذى تم إنجازه فى المدى المحدد مسبقاً.
- مقدار الخطوة، وهى مقدار زيادة قضيب الأداة كلما تم استدعاء الدالة StepIt().

وفيما يلى سنقوم بتوضيح كيفية التحكم فى تلك المعاملات.

تعيين مدى أداة التقدم

يتم تعيين مدى أداة التقدم باستخدام الدالة SetRange() التى تحتوى على معاملين يمثلان القيمة الصغرى والقيمة العظمى للمدى. بالطبع تختلف قيم المدى باختلاف المهمة المراد تنفيذها، فإذا أردت مثلاً حساب الأعداد المحصورة بين ٣٠٠٠ و ٧٠٠٠ تكون هذه الأرقام قيم المدى المطلوب. وغالباً يتم تعيين مدى أداة التقدم فى نهاية دالة الاستهلال الخاصة بالمربع الحوارى، كما يمكنك تغيير هذا المدى فى أى وقت أثناء تنفيذ البرنامج باستخدام نفس الدالة. لتعيين مدى أداة التقدم التى قمنا بإضافتها منذ قليل بحيث يكون من صفر إلى عشرة، تابع معنا الخطوات الآتية:

١. نشط التبويب Class View من نافذة عمل المشروع إذا لم يكن هو التبويب النشط.
٢. انقر التصنيف CProgressDlg بالجزء العلوى من التبويب لتنشيطه.
٣. من الجزء السفلى من التبويب، انقر دالة استهلال المربع الحوارى OninitDialog() نقراً مزدوجاً، يتم فتح الدالة داخل نافذة المحرر.
٤. قم بإضافة السطر التالى:

```
m_myProgress.SetRange(0,10);
```

إلى نهاية الدالة وقبل عبارة Return; (بعد السطر //TODO).

تعمل الدالة **SetRange()** على القيم ١٦بت، لذا فهى تصلح فى المدى من – ٣٢٧٦٨ إلى ٣٢٧٦٧. فإذا زاد المدى عن ذلك، يمكنك استخدام الدالة **SetRange32()** بدلاً منها، حيث تعمل هذه الدالة على القيم ٣٢ بت، لذا فهى تصلح فى المدى من –٢١٤٧٤٨٣٦٤٨ إلى ٢١٤٧٤٨٣٦٤٧.

تعيين المكان الحالى لقضيب أداة التقدم

بعد أن قمت بتعيين مدى الدالة، يمكنك تعيين أو تغيير الوضع الحالى لقضيب أداة التقدم باستخدام الدالة **SetPos()** التى تحتوى على معامل رقمى يوضح مكان انتهاء القضيب. ففى المثال الذى بين أيدينا، إذا أردت أن يكون قضيب أداة التقدم نصف ممتلئ، قم بتمرير رقم ٥ إلى الدالة هكذا:

```
m_myProgress.SetPos (5);
```

وحينما يقل هذا المعامل عن القيمة الصغرى للمدى، يظهر القضيب فارغاً تماماً، أما إذا زاد المعامل عن القيمة العظمى للمدى، فسيظهر القضيب ممتلئاً. عندما يفتح المربع الحوارى، يتم تعيين الوضع الحالى لأداة التقدم تلقائياً بالقيمة الصغرى للمدى، إلا أنك تستطيع أداء ذلك ذاتياً بإضافة السطر التالى للدالة **OnInitDialog()**:

```
m_myProgress.SetPos(0);
```

فضلاً عن تغيير وضع أداة التقدم بتمرير رقم يقع داخل المدى مباشرة، يمكنك تعيين مكان جديد بالنسبة للمكان الحالى وذلك باستخدام الدالة **OffSetPos()**، حيث يتم إضافة المعامل الجديد إلى المكان الحالى. ففى المثال الذى بين أيدينا، إذا أردت تغيير موضع أداة التقدم من الموضع الحالى **(5)** إلى الموضع **(8)**، قم بتمرير المعامل ٣ إلى الدالة هكذا:

```
m_myProgress.OffSetPos(3);
```

تعيين مقدار الخطوة

يمكنك تعيين قيمة تلقائية لزيادة تقدم القضيب كلما تم استقبال رسالة التحديث المناسبة وذلك باستخدام الدالة **SetStep()** التى تحتوى على معامل رقمى يوضح مقدار الخطوة. كما يمكنك تحديث أداة التقدم باستدعاء الدالة **StepIt()** دون أية معاملات. ففى المثال الذى بين أيدينا، يمكنك تعيين مقدار الخطوة بمقدار ١ بإضافة السطر التالى:

```
m_myProgress.SetStep(1);
```

لنهاية الدالة **OnInitDialog()** ثم استدعاء الدالة **StepIt()** داخل دالة نقر الزر **Step It-->** هكذا:

```
void CProgressDlg::OnBnClickedStepit()
{
        m_myProgress.StepIt();
}
```

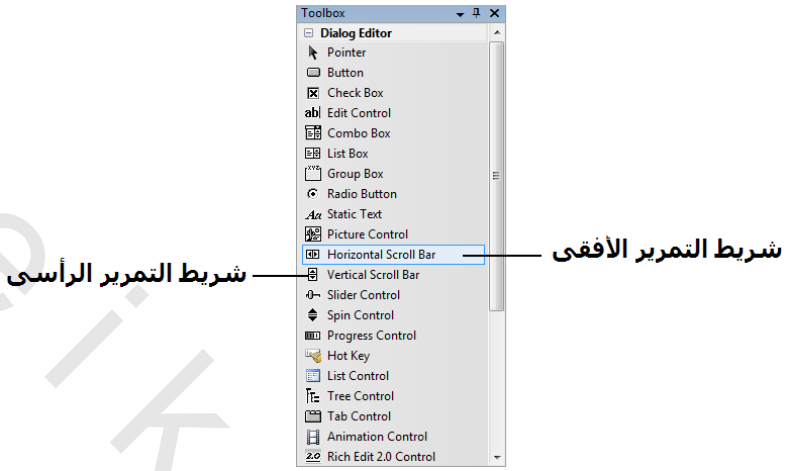
قم ببناء وتنفيذ البرنامج، تلاحظ أنك كلما نقرت زر **Step It-->**، كلما زاد طول قضيب أداة التقدم بمقدار ١ حتى تصل إلى النهاية بعد ١٠نقرات. فإذا قمت بنقر الزر مرةً أخرى، يصبح القضيب فارغاً، وهكذا تتكرر العملية باستمرار إلى أن تخرج من المربع الحوارى.

استخدام أشرطة التمرير

تظهر أشرطة التمرير **Scrollbars** غالباً مصاحبةً لإطار النافذة كى يتيح لك التنقل عبر محتوياتها بسهولة ويسر، إلا أنها تستخدم أحياناً لتعيين مكان داخل مدى معين، وهى الوظيفة الأساسية لأداة المنزلق **Slider Control** التى سنتعرض لها بالتفصيل بعد قليل.

إضافة شريط التمرير إلى المربع الحوارى

يوجد نوعان من أشرطة التمرير داخل مربع الأدوات، أحدهما شريط التمرير الأفقى **Horizontal Scrollbar** والآخر شريط التمرير الرأسى **Vertical Scrollbar** (انظر شكل ٩-٤).



شكل ٩-٤ شريط التمرير الأفقى والرأسى داخل مربع الأدوات

للتعرف على طريقة إضافة شريط التمرير إلى المربع الحوارى، قم بإنشاء مشروع حوارى جديد باسم **MyScroll** ثم تابع معنا الخطوات الآتية:

١. انقر رمز شريط التمرير الأفقى Horizontal Scroll Bar من مربع الأدوات، يظهر مؤشر الفأرة مصحوبا بعلامة +.

٢. انقر بزر الفأرة الأيسر فى المكان الذى تريد وضع شريط التمرير فيه داخل المربع الحوارى، يظهر شريط التمرير داخل المربع الحوارى.

٣. انقر شريط التمرير بزر الفأرة الأيمن ثم اختر **Properties** من القائمة الموضعية لإظهار نافذة مربع الخصائص إذا لم تكن ظاهرة. (يمكنك إظهار نافذة مربع الخصائص مباشرةً بضغط الاختصار **Alt+Enter** من لوحة المفاتيح).

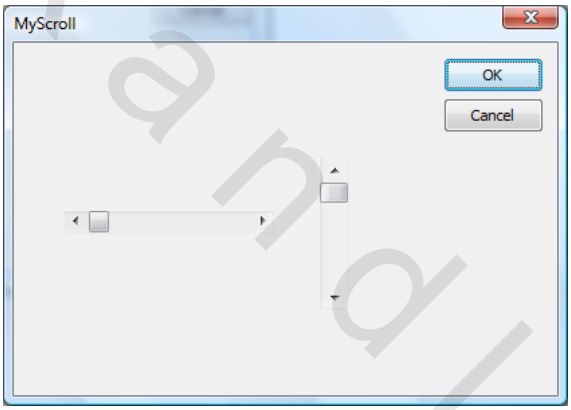
٤. قم بتعيين اسم مناسب لشريط التمرير الأفقى داخل الخاصية **ID** بمربع الخصائص وليكن **IDC_HSCROLL**.

٥. قم باختيار إحدى القيم التالية من مربع السرد والتحرير المجاور للخاصية **Align** بمربع الخصائص:

- الخيار **None** وهى الخيار الافتراضى ويتسبب فى رسم شريط التمرير بنفس حجم المستطيل الذى قمت بوضعه داخل قالب المربع الحوارى.
- الخيار **Top** وتتسبب فى رسم شريط التمرير بالعرض القياسى وتكون المحاذاة أعلى ويسار المستطيل الذى قمت بوضعه داخل قالب المربع الحوارى.
- الخيار **Bottom** ويتسبب فى رسم شريط التمرير بالعرض القياسى وتكون المحاذاة أعلى ويمين المستطيل الذى قمت بوضعه داخل المربع الحوارى.

قم باتباع نفس الخطوات السابقة لإضافة شريط تمرير رأسى باسم **IDC_VSCROLL**.

يوضح شكل ٩-٥ مظهر شريطى التمرير داخل المربع الحوارى.



شكل ٩-٥ مظهر أشرطة التمرير داخل المربع الحوارى

ربط شريط التمرير بمتغير

كى تستطيع التعامل مع شريط التمرير، يجب أن تقوم بتعيين متغير عضو فى التصنيف **CScrollDlg** الذى يحتوى على وظائف شريط التمرير وذلك كما يلى:

١. نشط التبويب **Class View** من نافذة عمل المشروع.

٢. انقر التصنيف **CScrollDlg** بزر الفأرة الأيمن ثم اختـــر **Add➢Add Variable** من القائمة الموضعية، يظهر معالج إضافة متغير جديد.

٣. قم بتنشيط مربع الاختيار **Control variable** لأننا نرغب فى إنشاء متغير مرتبط بإحدى أدوات التحكم الموجودة بالمربع الحوارى.
٤. اختر شريط التمرير الأفقى **IDC_HSCROLL** من مربع السرد **Control ID**.
٥. اختر **Control** من مربع السرد والتحرير **Category** وتأكد من اختيار **CScrollBar** فى مربع **Variable Type**.
٦. اكتب اسم المتغير فى خانة **Variable name** وليكن **m_hScroll** واختر درجة المتغير من مربع السرد **Access** وهو **Public** فى هذه الحالة.
٧. انقر زر **Finish** لإغلاق نافذة المعالج.
٨. قم بتكرار الخطوات السابقة لربط المتغير **m_vScroll** بشريط التمرير الرأسى **IDC_VSCROLL**.

تعيين قيمة ابتدائية لشريط التمرير

يمكنك تعيين قيمة ابتدائية لشريط التمرير عن طريق تعيين مدى شريط التمرير باستخدام الدالة **()SetScrolRange** التى تحتوى على ثلاثة معاملات، الأول يمثل القيمة الصغرى بينما يمثل الثانى القيمة العظمى أما المعامل الثالث فقيمته الافتراضية **TRUE**وهذا يعنى إعادة رسم القضيب لعكس التغيرات التى تتم عليه من خلال الكود ويمكنك تغيره إلى **FALSE**. لتعيين قيمة ابتدائية لمدى شريطى التمرير الأفقى والرأسى، قم بإضافة السطرين التاليين لنهاية الدالة **()OnInitDialog**:

```
m_vScroll.SetScrollRange (0,100);
m_hScroll.SetScrollRange (0,200);
```

لاحظ عدم تمرير المتغير الثالث وذلك لأن قيمته الافتراضية **TRUE**.

يمكنك أيضاً الحصول على قيم المدى الحالى لشريط التمرير باستخدام الدالة **()GetScrollRange** بتمرير مؤشرين صحيحين لاستقبال المدى الحالى وذلك كما يلى:

```
int nMin , nMax;
m_hScroll.GetScrollRange(&nMin,nMax);
TRACE("Range = (%d to %d)\n ", nMin , nMax);
```

كما يمكنك تعطيل أيٍ من أسهم التمرير الموجودة بنهاية شريط التمرير باستخدام الدالة **()EnableScrollBar** بتمرير قيمة من القيم الموضحة بالجدول ٩-١التالى.

جدول ٩-١القيم المستخدمة لتمكين/تعطيل أسهم التمرير

القيمة	*الاستخدام*
ESB_DISABLE_BOTH	تعطيل الزرين
ESB_DISABL_LTUP	تعطيل الزر الأيسر أو الأعلى تبعاً لنوع شريط التمرير
ESB_DISABLE_RTDN	تعطيل الزر الأيمن أو الأسفل تبعاً لنوع شريط التمرير
ESB_ENABLE_BOTH	تمكين الزرين

ومثلما فعلنا مع أداة التقدم، يمكنك تعيين المكان الحالى لمؤشر شريط التمرير الذى يتم تمثيله بمستطيل صغير متحرك باستخدام الدالة **()SetScrollPos** التى تحتوى على معامل رقمى يحدد مكان المستطيل. كما يمكنك معرفة المكان الحالى للمستطيل باستخدام الدالة **()GetScrollPos**.

الاستجابة لرسائل شريط التمرير

إذا قمت بتغيير مكان مستطيل شريط التمرير بنقر أسهم التمرير أو ضغط مفتاح من لوحة المفاتيح، يقوم شريط التمرير بإرسال رسالة إلى المربع الحوارى، وقد تكون هذه الرسالة **WM_HSCRLL** فى حالة شريط التمرير الأفقى أو **WM_VSCRLL** فى حالة شريط التمرير الرأسى. لإنشاء دالة احتواء الرسالة **Handler Function** لشريط التمرير الأفقى، تابع معنا الخطوات الآتية:

١. قم بتنشيط التصنيف **CScrollDlg** من نافذة عرض التصنيفات.
٢. انقر زر الرسائل  من نافذة مربع الخصائص، تظهر قائمة بالرسائل المختلفة.
٣. اختر الرسالة **WM_HSCROLL** ثم اختر **Add OnHScroll** من مربع السرد والتحرير المجاور، يظهر هيكل دالة الرسالة داخل نافذة الكود.

٤. قم بتكرار الخطوات السابقة ولكن مع اختيار **Add OnVScroll** من مربع السرد والتحرير المجاور للرسالة **WM_VSCROLL**.

٥. بعد أن تقوم بإنشاء الرسالتين، يقوم معالج التصنيفات بإنشاء الكود التالى لدالة احتواء رسالة شريط التمرير الرأسى:

```
void CScrollDlg::OnVScroll(UINT nSBCode, UINT nPos,
CScrollBar* pScrollBar)
{
        // TODO: Add your message handler code here
                and/or call default

        CDialog::OnVScroll(nSBCode, nPos, pScrollBar);
}
```

بينما يقوم بإنشاء الكود التالى لدالة احتواء رسالة شريط التمرير الأفقى:

```
void CScrollDlg::OnHScroll(UINT nSBCode, UINT nPos,
CScrollBar* pScrollBar)
{
        // TODO: Add your message handler code here
                and/or call default
        CDialog::OnHScroll(nSBCode, nPos, pScrollBar);
}
```

بالنظر إلى الدالة **()OnVScroll** أو الدالة **()OnHScroll**، تجد أن كلتاهما تحتوى على ثلاثة معاملات يمكن بيانها كما يلى:

- يستخدم المعامل **nSBCode** لتوضيح الطريقة التى قام بها المستخدم لتغيير مكان مستطيل شريط التمرير. فربما قام بسحب المستطيل أو نقر أحد أسهم التمرير أو نقر أى مكان داخل شريط التمرير أو ربما قام بضغط مفتاح **Page UP** أو **Page Down** أو أحد مفاتيح الأسهم الموجودة بلوحة المفاتيح. يوضح جدول ٩-٢ القيم المتاحة لهذا المعامل ودلالة كل منها. يمكنك استخدام هذه القيم لتحديد المكان الجديد للمستطيل باستخدام الدالة **()SetScrollPos**، وإلا سيعود المستطيل إلى بداية شريط التمرير بمجرد أن يقوم المستخدم بتحريره.

جدول ٩-٢ القيم المستخدمة من قِبل المعامل **nSBCode**

القيمة	*المدلول*
SB_THUMBTRACK	**قام المستخدم بسحب مربع التمرير إلى مكان معين**
SB_THUMBTPOSITION	**قام المستخدم بسحب مربع التمرير إلى مكان معين ثم قام بتحرير الزر**
SB_ENDSCROLL	**قام المستخدم بتحرير زر الفأرة بعد نقر إحدى نهايات أسهم التمرير أو جسم شريط التمرير**
SB_LINEUP	**يجب إنقاص مربع التمرير بمقدار واحد (فى حالة شريط التمرير الرأسى)**
SB_LINELEFT	**يجب إنقاص مربع التمرير بمقدار واحد (فى حالة شريط التمرير الأفقى)**
SB_LINEDOWN	**يجب زيادة مربع التمرير بمقدار واحد (فى حالة شريط التمرير الرأسى)**
SB_LINERIGHT	**يجب زيادة مربع التمرير بمقدار واحد (فى حالة شريط التمرير الأفقى)**
SB_PAGEUP	**يجب إنقاص مربع التمرير بمقدار صفحة واحدة(فى حالة شريط التمرير الرأسى)**
SB_PAGELEFT	**يجب إنقاص مربع التمرير بمقدار صفحة واحدة (فى حالة شريط التمرير الأفقى)**
SB_PAGEDOWN	**يجب زيادة مربع التمرير بمقدار صفحة واحدة (فى حالة شريط التمرير الرأسى)**
SB_PAGELEFT	**يجب زيادة مربع التمرير بمقدار صفحة واحدة (فى حالة شريط التمرير الأفقى)**

- يستخدم المعامل **npos** لتحديد المكان الحالى لمستطيل شريط التمرير وهو كما ترى من النوع **Unsigned Integer**. فإذا كان المدى الابتدائى لشريط التمرير يسمح بتمثيل الأرقام السالبة، قم بتغيير نوع المعامل من **Uint** إلى **int** وبالتالى ستقوم الأرقام الموجبة الكبيرة بتمثيل قيمة سالبة.

- يستخدم المعامل **pScrollBar** لمعرفة شريط التمرير الذى قام بإرسال الرسالة إلى المربع الحوارى حيث يحتوى على مؤشر لشريط التمرير الذى قام المستخدم بتغييره.

والآن قم بإدخال كود الدالة **OnVScroll()** كما يلى:

```
1.    void CScrollDlg::OnVScroll(UINT nSBCode, UINT nPos,
2.    CScrollBar* pScrollBar)
3.    {
4.      if (pScrollBar->GetDlgCtrlID() == IDC_VSCROLL)
5.      {
6.        int nCurrentPos = pScrollBar->GetScrollPos();
7.        switch(nSBCode)
8.        {
9.          case SB_THUMBTRACK:
10.         case SB_THUMBPOSITION:
11.                 pScrollBar->SetScrollPos(nPos);
12.                 break;
13.         case SB_LINEUP:
14.                 pScrollBar->SetScrollPos(nCurrentPos-1);
15.                 break;
16.         case SB_LINEDOWN:
17.                 pScrollBar->SetScrollPos(nCurrentPos+1);
18.                 break;
19.         case SB_PAGEUP:
20.                 pScrollBar->SetScrollPos(nCurrentPos-5);
21.                 break;
22.         case SB_PAGEDOWN:
23.                 pScrollBar->SetScrollPos(nCurrentPos+5);
24.                 break;
25.       }
26.     }

27.     CDialog::OnVScroll(nSBCode, nPos, pScrollBar);
```

```
28.  }
```

وعن هذا الكود نوضح ما يلى:

- فى السطر رقم ٤ يتم التأكد من أن شريط التمرير الرأسى هو الذى تم تغييره وذلك باختبار اسم القضيب باستخدام الدالة **GetDlgCtrlID()**. فإذا كان اسم القضيب هو **IDC _VSCROLL** يتم الحصول على المكان الحالى لمستطيل شريط التمرير باستخدام الدالة **GetScrollPos()** كما فى السطر رقم ٦ حيث يتم تخزينه فى المتغير **nCrrentPos**.
- فى السطر رقم ٧ يتم استخدام عبارة**switch** لتحديد الحدث الذى قام به المستخدم باستخدام المعامل **nSBCode** وذلك كما يلى:
  - إذا قام المستخدم بسحب مربع التمرير أو قام بسحب المربع وتحرير زر الفأرة، يكون المكان الحالى للمستطيل هو **nPos** (السطور من ٩ إلى ١٢).
  - إذا قام المستخدم بتحريك مربع التمرير لأعلى (أو لأسفل) بمقدار سطر بنقر أحد سهمى التمرير، يتم زيادة (أو إنقاص) المكان الحالى للمستطيل بمقدار ١(السطور من ١٣ إلى ١٨).
  - إذا قام المستخدم بضغط مفتاح**Page up** (أو مفتاح **Page Down**)، يتم زيادة (أو إنقاص) المكان الحالى للمربع بمقدار طول الصفحة وهو ٥ فى هذا المثال (السطور من ١٩ إلى ٢٤).

بنفس الطريقة قم بإضافة كود الدالة **OnHScroll()** مع كما يلى:

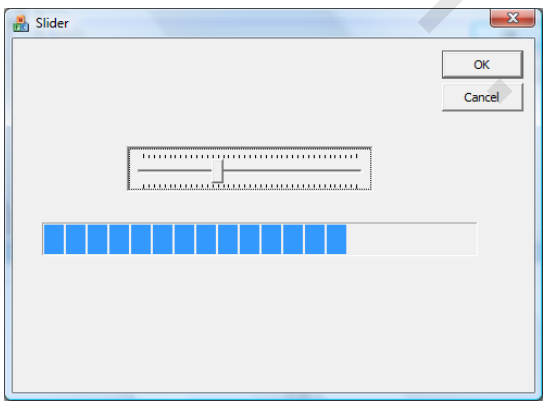
```
1.    void CScrollDlg::OnHScroll(UINT nSBCode, UINT nPos,
      CScrollBar* pScrollBar)
2.    {
3.      if (pScrollBar->GetDlgCtrlID() == IDC_HSCROLL)
4.      {
5.        int nCurrentPos = pScrollBar->GetScrollPos();
6.        switch(nSBCode)
7.        {
8.          case SB_THUMBTRACK:
9.          case SB_THUMBPOSITION:
10.               pScrollBar->SetScrollPos(nPos);
```

```
11.                 break;
12.         case SB_LINELEFT:
13.                 pScrollBar->SetScrollPos(nCurrentPos-1);
14.                 break;
15.         case SB_LINERIGHT:
16.                 pScrollBar->SetScrollPos(nCurrentPos+1);
17.                 break;
18.         case SB_PAGELEFT:
19.                 pScrollBar->SetScrollPos(nCurrentPos-5);
20.                 break;
21.         case SB_PAGERIGHT:
22.                 pScrollBar->SetScrollPos(nCurrentPos+5);
23.                 break;
24.       }
25.    }

26.     CDialog::OnHScroll(nSBCode, nPos, pScrollBar);
27.  }
```

استخدام المنزلق Slider

يستخدم المنزلق لتعيين قيمة داخل مدى معين بسحب المؤشر ثم تحريره (انظر شكل ٩-٦). ورغم أنه يشبه إلى حد كبير شريط التمرير فى احتوائه على مدى معين والمكان الحالى داخل هذا المدى، إلا أنه يتفوق عليه فى كثير من الصفات كما سنرى بعد قليل.



شكل ٩-٦ الشكل العام للمنزلق داخل المربع الحوارى

إضافة المنزلق إلى المربع الحوارى

للتعرف على طريقة إضافة المنزلق إلى المربع الحوارى، قم بإنشاء مشروع حوارى جديد باسم **Slider** ثم تابع معنا الخطوات التالية:

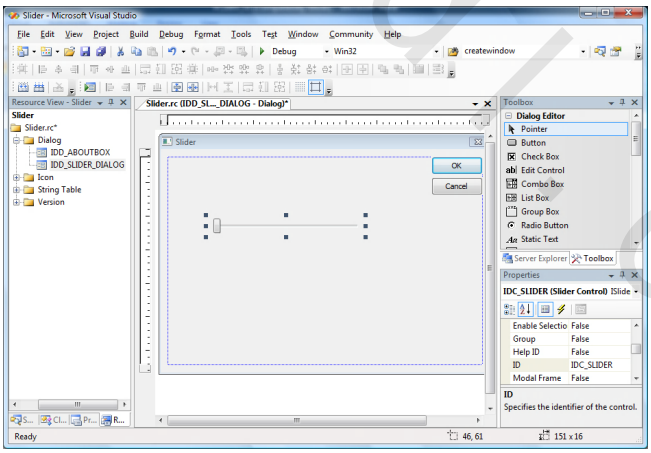
١. انقر رمز المنزلق Slider Control من مربع الأدوات، يظهر مؤشر الفأرة مصحوبا بعلامة +.

٢. انقر بزر الفأرة الأيسر فى المكان الذى تريد المنزلق فيه داخل المربع الحوارى، يظهر المنزلق داخل المربع الحوارى.

٣. انقر المنزلق بزر الفأرة الأيمن ثم اختر **Properties** من القائمة الموضعية لإظهار نافذة مربع الخصائص إذا لم تكن ظاهرة بالفعل.

٤. قم بتعيين اسم مناسب للمنزلق داخل الخاصية **ID** بمربع الخصائص وليكن **IDC_SLIDER**.

٥. من مربع الخصائص قم بتغير إعدادات شكل المنزلق كما يروق لك وذلك كما يلى (انظر شكل ٩-٧):



شكل ٩-٧ التحكم فى خصائص المنزلق

- من الخاصية **Orientation**، اختر اتجاه المنزلق، رأسى **Vertical** أم أفقى **Horizontal** (القيمة الافتراضية).

- من الخاصية **Point**، اختر شكل مؤشر المنزلق. اختر **Both** إذا أردت إظهار المؤشر على شكل مستطيل (القيمة الافتراضية) أو **Top / Left** إذا أردت أن يظهر المؤشر بنهاية حادة من أعلى فى حالة المنزلق الأفقى أو بنهاية حادة من اليسار فى حالة المنزلق الرأسى، أما الخيار الثالث فهو **Bottom /Right** وحينئذٍ سيظهر المؤشر بنهاية حادة وفى عكس اتجاه النوع السابق.
- قم بتخصيص القيمة **True** للخاصية **Tick Marks** إذا أردت أن يقوم المنزلق بإظهار خطوط صغيرة عمودية بعيداً عن اتجاه المؤشر كى تساعد المستخدم فى ضبط الوضع الحالى للمنزلق بدقة أعلى.
- قم بتخصيص القيمة **True** للخاصية **Auto Ticks** إذا أردت أن تظهر الخطوط الصغيرة تبعاً لمقدار الزيادة (مقدار الخطوة) داخل مدى المنزلق.
- قم بتخصيص القيمة **True** للخاصية **Enable Selection Range** إذا أردت إضافة شريط أبيض لزيادة التحكم فى وضع المؤشر داخل المدى باستخدام مثلثات صغيرة.
- قم بتخصيص القيمة **True** للخاصية **Border** إذا أردت رسم إطار أسود رفيع حول المنزلق.

ربط المنزلق بمتغير

كى تستطيع التعامل مع المنزلق، يجب أن تقوم بربط المنزلق بمتغير كغيره من أدوات التحكم الأخرى وذلك كما يلى:

١. نشط التبويب **Class View** من نافذة عمل المشروع.
٢. انقر التصنيف **CSliderDlg** بزر الفأرة الأيمن ثم اختـــر **Add➢Add Variable** من القائمة الموضعية، يظهر معالج إضافة متغير جديد.
٣. قم بتنشيط مربع الاختيار **Control variable** لأننا نرغب فى إنشاء متغير مرتبط بإحدى أدوات التحكم الموجودة بالمربع الحوارى.

٤. اختر المنزلق **IDC_SLIDER** من مربع السرد **Control ID**.

٥. اختر **Control** من مربع السرد والتحرير **Category** وتأكد من اختيار **CSliderBar** فى مربع **Variable Type**.

٦. اكتب اسم المتغير فى خانة **Variable name** وليكن **m_Slider** واختر درجة المتغير من مربع السرد **Access** وهو **Public** فى هذه الحالة.

٧. انقر زر **Finish** لإغلاق نافذة المعالج.

تعيين قيمة ابتدائية للمنزلق

كما فى شريط التمرير، يمكنك تعيين القيم الابتدائية للمنزلق باستخدام الدالة **()SetRange** والتى تحتوى على ثلاثة معاملات، الأول يمثل القيمة الصغرى للمدى ويمثل الثانى القيمة العظمى، أما الثالث فيكون افتراضياً **TRUE** وهذا يعنى إعادة رسم المنزلق تبعاً للتغييرات التى تتم عليه أثناء التشغيل ويمكنك تغييره إلى **FALSE**. كما يمكنك أيضاً تعيين القيمة الصغرى والكبرى كلٍ على حدة باستخدام الدالة **()SetRangeMin** لتعيين القيمة الصغرى أو الدالة **()SetRangeMax** لتعيين القيمة الكبرى. كما يمكنك معرفة المدى الحالى للمنزلق باستخدام الدوال **()GetRangeMin** أو **()GetRangeMax**. لتعيين مدى المنزلق، قم بإضافة السطر التالى لنهاية الدالة **()OnInitDialog**:

```
m_Slider.SetRange(0,200) ;
```

لاحظ عدم تمرير المتغير الثالث وذلك لأن قيمته الافتراضية هى **TRUE**.

يمكنك أيضاً أداء المهام التالية عند العمل مع المنزلق:

- لتعيين المكان الحالى لمؤشر المنزلق، قم باستخدام الدالة **()SetPos** التى تحتوى على معامل رقمى يحدد مكان المؤشر. ولمعرفة المكان الحالى للمؤشر، قم باستخدام الدالة **()GetPos**.
- يتاح للمستخدم تقديم أو تأخير المنزلق بمقدار سطر أو صفحة وذلك بضغط أحد الأسهم أو أحد زرى **Page Up** و **Page Down**، إلا أنك فى هذه الحالة لست فى حاجة إلى تعيين الرسائل الصادرة من المنزلق كما فى شريط التمرير حيث تتم

هذه العملية ذاتياً. أما إذا أردت التحكم فى مقدار الزيادة أو النقصان، فقم باستخدام الدوال SetlineSize() و SetPageSize() وذلك بتمرير القيم المناسبة لمقدار زيادة أو نقصان السطر أو الصفحة. كما يمكنك معرفة قيم زيادة السطر أو الصفحة باستخدام الدوال GetLineSize() و GetPageSize().

- يمكنك أيضاً برمجة المنزلق لتحديد كثافة الخطوط الصغيرة Tick Marks باستخدام الدالة SetTicFreq(). فإذا أردت رسم خط صغير كل خمس أماكن، تأكد من تخصيص القيمة True للخاصية AutoTicks ثم استخدم الكود التالى:

```
m_ Slider.SetTicFreq (5) ;
```

- يمكنك معرفة عدد الخطوط الصغيرة داخل المدى المحدد باستخدام الدالة GetNumTicks().
- إذا قمت بتخصيص القيمة True للخاصية Enable Selection Range داخل مربع الخصائص، يمكنك إظهار مدى الاختيار داخل المنزلق باستخدام الدالة SetSelection() وذلك بتمرير القيم الصغرى والعظمى للاختيار. كما يمكنك استرجاع القيمة الصغرى والعظمى للاختيار الحالى باستخدام الدالة GetSelection(). كما يمكنك أيضاً إلغاء الاختيار نهائياً باستخدام الدالة ClearSel().

الاستجابة لرسائل المنزلق

يقوم المنزلق بإشعار المربع الحوارى بنفس الرسائل WM_VSCROLL أو WM_HSCROLL المستخدمة مع شريط التمرير، إلا أنك لست فى حاجة لتعيين المكان الجديد لمؤشر المنزلق حيث يتم تحديثه ذاتياً. ورغم ذلك يمكنك استخدام هذه الرسائل لأداء مهام معينة حينما يقوم المستخدم بتغيير مكان المؤشر.

لتوضيح ذلك، قم بإضافة أداة تقدم إلى المربع الحوارى باسم IDC_PROGRESS أسفل المنزلق ثم قم بربطه بمتغير من النوع CProgressCtrl باسم m_Progress كما تعلمت من قبل (راجع شكل ٩-٦). نريد فى هذا المثال تحديث أداة التقدم كلما تم تغيير

موضع مؤشر المنزلق. لذا قم بإضافة الكود التالى لدالة المنزلق **()OnHscroll** بعد أن تقوم بإضافتها للمشروع مثلما فعلت مع شريط التمرير:

```
void CSliderDlg::OnHScroll(UINT nSBCode , UINT nPos , CScrollBar* pScrollBar)
{
        if (pScrollBar->GetDlgCtrlID() == IDC_SLIDER)
                m_progress.SetPos(m_slider.GetPos());

        CDialog::OnHScroll(nSBCode, nPos, pScrollBar);
}
```

حيث تم استخدام عبارة **If** للتأكد من أداة التحكم التى تم تغييرها، فإذا كانت أداة التحكم هى المنزلق، تم استدعاء الدالة **()SetPos** التى تحتوى على قيمة المكان الجديد للمنزلق لتحديث أداة التقدم.

والآن قم ببناء المشروع وتنفيذه ثم اسحب ذراع المنزلق يميناً ويساراً ولاحظ انعكاس ذلك على شريط التقدم (راجع شكل ٩-٦).

الباب الثالث

استخدام المربعات الحوارية

١٠ . إنشاء وتصميم المربعات الحوارية.

١١ . العمل مع المربعات الحوارية.

١٢ . الاستجابة للأحداث.