**Работа со стандартными ресурсами**

**1. Методы класса CButton**

HBITMAP GetBitmap() const;

Возвращает дескриптор растрового изображения, сопоставленного кнопке. Если такового не существует, то возвращается NULL.

HBITMAP SetBitmap(HBITMAP hBitmap);

Сопоставляет кнопке растровое изображение. Значением параметра должен быть дескриптор растрового изображения. Правила размещения растрового изображения такие же, как и у значка.

HCURSOR GetCursor();

Возвращает дескриптор курсора, сопоставленного кнопке методом SetCursor. Если у кнопки нет сопоставленного курсора, то возвращается NULL.

HCURSOR SetCursor(HCURSOR hCursot);

Сопоставляет кнопке курсор, изображение которого будет помещено на поверхность кнопки аналогично значку и растровому изображению.

UINT GetState() const;

Возвращает описание набора текущих состояний кнопки. Чтобы выделить из этого описания значения конкретных типов состояния, можно использовать маски:

0х0003 - выделяет собственное состояние кнопки. Применимо только к флажку или переключателю. Если результат побитового умножения дает 0, значит кнопка находится в невыбранном состоянии, 1 - в выбранном, 2 - в неопределенном.

0х0004 - выделяет состояние первого типа. Ненулевой вариант означает, что кнопка "нажата", нулевой - кнопка свободна.

0х0008 - выделяет положение фокуса. Ненулевой вариант - кнопка в фокусе клавиатуры.

int GetCheck() const;

Возвращает собственное состояние флажка или переключателя. Возвращаемое значение может принимать одно из значений: 0 - кнопка не выбрана; 1 - кнопка выбрана; 2 - кнопка в неопределенном состоянии. Если кнопка не является ни переключателем, ни флажком, возвращается 0.

void SetCheck(int nCheck);

Устанавливает собственное состояние флажка или переключателя. Значения задаются из набора: 0 - невыбранное; 1 - выбранное; 2 - неопределенное. Значение 2 применимо только к флажку со свойством 3State.

UINT GetButtonStyle() const;

Возвращает стиль кнопки.

void SetButtonStyle(UINT nStyle, BOOL bRedraw=TRUE);

Устанавливает стиль кнопки. Если параметр bRedraw равен TRUE, кнопка перерисовывается.

HICON GetIcon() const;

Возвращает дескриптор пиктограммы, сопоставленной кнопке. Если у кнопки нет сопоставленной пиктограммы, возвращает NULL.

HICON SetIcon(HICON hIcon);

Сопоставляет кнопке пиктограмму. Значением параметра при вызове должен быть дескриптор пиктограммы.

Пиктограмма автоматически помешается на поверхность кнопки и сдвигается в ее центр. Если поверхность кнопки меньше пиктограммы, она обрезается со всех сторон до размеров кнопки. Положение пиктограммы может быть выровнено и не по центру. Для этого нужно, чтобы кнопка имела одно из следующих свойств: BS\_LEFT, BS\_RIGHT, BS\_CENTER, BS\_TOP, BS\_BOTTOM, BS\_VCENTER

Данный метод устанавливает для кнопки только одну пиктограмму, которая будет наравне с текстом присутствовать при любом ее состоянии. Не надо путать ее с растровым изображением у растровой кнопки.

**2. Методы класса CEdit**

Окна редактирования могут работать в режимах однострочного и многострочного редакторов. Приведем сначала методы, общие для обоих режимов, а затем методы для многострочного редактора.

Общие методы :

DWORD GetSel() const;

void GetSel(int& nStartChar, int& nEndChar) const;

Получает первую и последнюю позиции выделенного текста. Для значения типа DWORD младшее слово содержит позицию первого, старшее - последнего символа.

void SetSel(DWORD dwSelection, BOOL bNoScroll=FALSE);

void SetSel(int nStartChar, int nEndChar, BOOL bNoScroll=FALSE);

Устанавливает новое выделение текста, задавая первый и последний выделенный символ. Значение FALSE параметра bNoScroll должно отключать перемещение курсора в область видимости.

void ReplaceSel(LPCTSTR lpszNewText);

Заменяет выделенный текст на строку, передаваемую в параметре lpszNewText.

void Clear();

Удаляет выделенный текст.

void Copy();

Копирует выделенный текст в буфер.

void Cut();

Переносит (копирует и удаляет) выделенный текст в буфер обмена.

void Paste();

Вставляет текст из буфера обмена, начиная с позиции, в которой находится курсор.

BOOL Undo();

Отмена последней операции, выполненной редактором. Если редактор однострочный, возвращается всегда неотрицательное значение, иначе неотрицательное значение возвращается лишь в случае успешной замены.

BOOL CanUndo() const;

Определяет, можно ли отменить последнюю операцию редактора.

void EmptyUndoBuffer();

Сбрасывает флаг undo, сигнализирующий о возможности отмены последней операции редактора, и тем самым делает невозможным отмену. Этот флаг сбрасывается автоматически при выполнении методов SetWindowText и SetHandle.

BOOL GetModify() const;

Возвращает неотрицательное значение, если содержимое окна редактирования не модифицировалось. Информация о модификации поддерживается в специальном флаге, обнуляемом при создании окна редактирования и при вызове метода:

void SetModify(BOOL bModified=TRUE);

Устанавливает или сбрасывает флаг модификации (см. предыдущий метод). Флаг сбрасывается при вызове метода с параметром FALSE и устанавливается при модификации содержимого окна редактирования или при вызове SetModify с параметром TRUE.

BOOL SetReadOnly(BOOL bReadOnly=TRUE);

Устанавливает режим просмотра (bReadOnly=TRUE) или редактирования (bReadOnly=FALSE).

TCHAR GetPasswordChar() const;

Возвращает символ, который при выводе пароля будет появляться на экране вместо символов, набираемых пользователем. Если такой символ не определен, возвращается 0. Устанавливается этот символ методом (по умолчанию используется "\*"):

void SetPasswordChar(TCHAR ch);

void LimitText(int nChars=0);

Устанавливает максимальную длину в байтах текста, который может ввести пользователь. Если значение параметра равно 0, длина текста устанавливается равной UINT\_MAX.

Методы работы с многострочным редактором :

void LineScroll(int nLines, int nChars=0);

Прокручивает текст в области редактирования. Параметр nLimes задает число строк для вертикальной прокрутки. Окно редактирования не прокручивает текст дальше последней строки. При положительном значении параметра область редактирования сдвигается вдоль текста к последней строке, при отрицательной - к первой.

Параметр nChars задает число символов для горизонтальной прокрутки. Окно редактирования прокручивает текст вправо, даже если строки закончились. В этом случае в области редактирования появляются пробелы. При положительном значении параметра область редактирования сдвигается вдоль к концу строки, при отрицательном - к началу.

int GetFirstVisibleLine() const;

Возвращает номер первой видимой строки.

int GetLineCount() const;

Возвращает число строк текста, находящегося в буфере редактирования. Если текст не вводился, возвращает 1.

int GetLine(int nIndex, LPTSTR lpszBuffer) const;

int GetLine(int nIndex, LPTSTR lpszBuffer, int nMaxLength) const;

Копирует строку с номером, равным значению параметра nIndex, в буфер, заданный параметром lpszBuffer. Первое слово в буфере должно задавать его размер. При вызове второго варианта метода значение параметра nMaxLength копируется в первое слово буфера.

Метод возвращает число в действительности скопированных байтов. Если номер строки больше или равен числу строк в буфере окна редактирования, возвращает 0. Текст копируется без каких-либо изменений, нуль-символ не добавляется.

int LineIndex(int nLine=-1) const;

Возвращает номер первого символа в строке. Неотрицательное значение параметра принимается в качестве номера строки. Значение -1 задает текущую строку. Если номер строки больше или равен числу строк в буфере окна редактирования (строки нумеруются с 0), возвращается 0.

**3. Методы класса CListBox**

void ResetContent();

Очищает содержимое списка, делая его пустым.

int AddString( LPCSTR lpszItem);

Добавляет строку lpszItem в список и сортирует его, если при создании включено свойство Sort. В противном случае элемент добавляется в конец списка.

int DeleteString( UINT nIndex);

Удаляет из списка элемент с индексом nIndex. Индексация элементов начинается с 0.

int GetCurSel() const;

Получает индекс элемента, выбранного пользователем.

int SetCurSel( int nSelect);

Отмечает элемент с индексом nSelect как выбранный элемент списка. Если значение параметра равно -1, список не будет содержать отмеченных элементов.

int GetText( int nIndex, LPSTR lpszBuffer) const;

void GetText( int nIndex, CString& rString) const;

Копирует элемент с индексом nIndex в буфер.

int SetTopIndex( int nIndex);

Организует прокрутку списка в окне так, чтобы элемент с индексом nIndex был видимым.

int FindString( int nStartAfter, LPCSTR lpszItem) const;

Организует поиск в списке и возвращает в качестве результата индекс элемента списка, префикс которого совпадает со строкой lpszItem. Результат не зависит от регистра, в котором набирались символы сравниваемых строк. Параметр nStartAfter задает начало поиска, но поиск идет по всему списку. Он начинается от элемента, следующего за nStartAfter, до конца списка и затем продолжается от начала списка до элемента с индексом nStartAfter. В качестве результата выдается первый найденный элемент, удовлетворяющий условиям поиска. Если такого нет, результат получает значение LB\_ERR.

int FindStringExact( int nIndexStart, LPCSTR lpszFind) const;

Этот метод отличается от предыдущего тем, что теперь не префикс элемента должен совпадать со строкой lpszFind, а сам элемент. Поиск по-прежнему не чувствителен к регистру, в котором набираются символы.

4. Методы класса CComboBox

int GetCurSel() const;

Возвращает целочисленный указатель выбранной строчки.

int SetCurSel(int nSelect);;

Ставит указатель на строчку с номером nSelect.

int GetLBText(int nIndex, LPTSTR lpszText) const;

void GetLBText(int nIndex, CString& rString) const;

Записывает содержимое строчки с индексом nIndex в переменные LPTSTR lpszText или CString& rString.

int GetLBTextLen(int nIndex) const;

Возвращает длину строчки с индексом nIndex.

int AddString(LPCTSTR lpszString);

Добавляет строчку в список.

int DeleteString(UINT nIndex);

Удаление строчки с индексом nIndex.

int InsertString(int nIndex, LPCTSTR lpszString);

Заменяет строчку с индексом nIndex содержимым переменной LPCTSTR lpszString.

**5. Методы класса CProgressCtrl**

void SetRange(short nLower, short nUpper);

void SetRange32(int nLower, int nUpper);

Устанавливает минимальное ( nLower ) и максимальное значение ( nUpper ).

void GetRange(int& nLower, int& nUpper);

Записывает в переменные nLower и nUpper минимальное и максимальное значение.

int GetPos();

Возвращает текущее значение.

int SetPos(int nPos);

Устанавливает текущее значение в nPos.

int SetStep(int nStep);

Устанавливает шаг ( nStep ) вывода.

**6. Методы класса CSliderCtrl**

int GetRangeMax() const;

int GetRangeMin() const;

void GetRange(int& nMin, int& nMax) const;

Первые две функции возвращают максимальное и минимальное знанение, а третья - записывает эти значения в nMax и nMin соответственно.

void SetRangeMin(int nMin, BOOL bRedraw = FALSE);

void SetRangeMax(int nMax, BOOL bRedraw = FALSE);

void SetRange(int nMin, int nMax, BOOL bRedraw = FALSE);

Первые две функции устанавливают максимальное и минимальное знанение, а третья - устанавливает эти значения из переменных nMax и nMin соответственно. Аргумент bRedraw отвечает за перерисовку слайдера.

int GetPos() const;

Возвращает текущую позицию.

void SetPos(int nPos);

Устанавливает текущую позицию в nPos.

BOOL SetTic(int nTic);

Устанавливает шаг ( nTic ).

void SetTicFreq(int nFreq);

Устанавливает частоту засечек ( nFreq ).

**7. Методы класса CSpinButtonCtrl**

int SetPos(int nPos);

Устанавливает текущую позицию в nPos.

int GetPos() const;;

Возвращает текущую позицию.

void SetRange(int nLower, int nUpper);

void SetRange32(int nLower, int nUpper);

Устанавливает максимальное и минимальное знанение из переменных nMax и nMin соответственно.

void GetRange(int &lower, int& upper) const;

void GetRange32(int &lower, int& upper) const;;

Эти две функции записывают максимальное и минимальное знанение в upper и lower соответственно.