

ЛЕКЦИЯ 12

СПИСОК	1
Свойства списка	1
Сообщения списка	1
Создание списка	2
Функции класса CListBox	2

СПИСОК

Элемент управления "список" представляет собой окно, где отображен список текстовых элементов, которые пользователь может просматривать и выбирать.

Свойства списка

Если **List Box** создается с помощью редактора диалоговых окон, то на странице свойств **Styles** доступны следующие стили:

Selection	определяет возможные способы выбора пунктов в списке;
Single	пользователь может выбрать только один пункт (по умолчанию);
Multiple	можно с помощью мыши и клавиш Shift и Ctrl выбирать несколько пунктов;
Extended	в дополнение к возможностям множественного выбора, у пользователя есть возможность расширять выделение путем перемещения курсора, что облегчает выбор стоящих подряд пунктов;
None	пользователь не может выбирать пункты;
Owner draw	используется, когда возникает необходимость формировать список с помощью функций родительского окна;
	No (по умолчанию), пункты списка могут быть только текстовыми;
	Fixed все пункты будут иметь одинаковую высоту;
	Variable высота пунктов может быть различна;
Has strings	определяет, что пунктами списка являются строки. Используется для списка, имеющего стиль Owner draw ;
Sort	элементы списка сортируются в алфавитном порядке (по умолчанию);
Notify	родительскому окну посылается сообщение, при изменении выбора элементов или при двойном щелчке мыши;
Multi-column	дает возможность создавать список, состоящий из нескольких столбцов. При этом возможность вертикальной прокрутки списка становится недоступной, а список отображается в виде столбцов, которые могут не поместиться в окне. Для их просмотра следует использовать горизонтальную прокрутку, которая устанавливается свойством Horizontal scroll ;
Vertical scroll	вертикальная прокрутка используется для больших списков, отображаемых в виде одного столбца;
No redraw	отключает автоматическое отображение изменений списка;
Use tabstops	позволяет отображать позиции табуляции;
Want key input	включает посылку сообщений родительскому окну при нажатии клавиш;
Disable no scroll	при свойстве Vertical scroll полоса прокрутки отображается как недоступная, даже если список полностью отображается в окне, иначе полоса отображается только по необходимости;
No integral height	отключает автоматическую подгонку высоты окна списка так, чтобы выводилось целое число элементов списка. Окно отображается точно таким, каким его создал разработчик;

Сообщения списка

Если при создании **List Box** назначено свойство **Notify**, то родительскому окну посылаются следующие сообщения:

LBN_SELCHANGE	посылается при изменении выделения, а для элементов со стилем Multiple данное сообщение посылается также и при нажатии клавиш перемещения курсора;
----------------------	---

LBN_DBLCLK	посылается при двойном щелчке мыши на строке списка;
LBN_ERRSPACE	посылается, когда элемент управления не может получить запрашиваемую память;
LBN_KILLFOCUS	при потере фокуса ввода;
LBN_JSELCANSEL	при отмене выбора;
LBN_SETFOCUS	при получении фокуса ввода.

Последние три из перечисленных сообщений посылаются также и при отсутствии свойства **Notify**.

Создание списка

Для создания элемента управления **List Box** программным путем нужно использовать конструктор класса **CListBox** и функцию **Create ()**, имеющую следующий прототип:

BOOL Create (DWORD dwStyle , const RECT& rect , CWnd* pParentWnd , UINT nID) ;

Параметр **dwStyle** может принимать значения, соответствующие свойствам окна списка, которые описаны выше. Эти значения следующие:

LBS_EXTENDEDSEL	LBS_HASSTRINGS	LBS_MULTICOLUMN
LBS_MULTIPLESEL	LBS_NOINTEGRALHEIGHT	LBS_NOREDRAW
LBS_NOTIFY	LBS_OWNERDRAWFIXED	LBS_STANDARD
LBS_SORT	LBS_OWNERDRAWVARIABLE	LBS_USETABSTOPS
LBS_DISABLENOSCROLL	LBS_WANTKEYBOARDINPUT	

Кроме того, могут использоваться стили окон Windows:

WS_CHILD	дочернее окно (используется всегда);
WS_VISIBLE	видимое окно (как правило);
WS_DISABLED	недоступное окно (иногда);
WS_VSCROLL	наличие вертикальной полосы прокрутки текста;
WS_HSCROLL	наличие горизонтальной полосы прокрутки;
WS_GROUP	включение управляющего элемента в группу;
WS_TABSTOP	возможность перехода между элементами управления с помощью клавиши Tab .

Функции класса **CListBox**

Не все параметры списка можно задать с помощью редактора ресурсов или параметров функции **Create ()**. Следующие функции дополняют возможности управления списком с помощью стилей. Функции

int GetHorizontalExtent () ;
void SetHorizontalExtent (int cxExtent) ;

получают и устанавливают ширину списка в пикселях, в пределах которой его можно прокручивать по горизонтали. Получить или установить индекс верхнего элемента списка можно при помощи функций

int GetTopIndex () const ;
int SetTopIndex (int nIndex) ;

Для многостолбцовых списков функция **SetColumnWidth ()** устанавливает ширину столбца. Ее аргументом является ширина столбца в пикселях. Для списков, созданных с использованием стиля **LBS_OWNERDRAWVARIABLE**, получить и установить высоту элемента списка позволяют функции

int SetItemHeight (int nIndex , UINT cyItemHeight) ;
int GetItemHeight (int nIndex) ;

Параметр **nIndex** является индексом элемента, **cyItemHeight** – его высотой в пикселях. Если указанный стиль не установлен, то индекс строки следует установить нулевым.

Установить позиции табуляции можно при помощи функций

void SetTabStops () ;
BOOL SetTabStops (const int & cxEachStop) ;
BOOL SetTabStops (int nTabStops , LPINT rgTabStops) ;

Первая устанавливает табуляцию по умолчанию, равной двум диалоговым единицам; вторая устанавливает эти позиции в соответствии с параметром **cxEachStop**, третья позволяет установить массив позиций табуляции из **nTabStops** элементов, определяемых массивом **rgTabStops**. Указанные функции применяются в сочетании со стилем **LBS_USETABSTOPS**.

После создания элемента управления **List Box**, его заполнение производится функцией

int AddString (LPCTSTR lpszItem) ;

которая добавляет строки, определяемые параметром **lpzItem** в список. Если установлен стиль **LBS_SORT**, то после добавления строки список сортируется, в противном случае строка добавляется в конец списка. Вставить строку в список в соответствии с индексом **nIndex** позволяет функция

```
int InsertString ( int nIndex , LPCTSTR lpzItem ) ;
```

Обе функции возвращают индекс новой строки в списке. Часто список используется для выбора имени файла в текущей директории. Для его инициализации служит функция

```
int Dir ( UINT attr , LPCTSTR lpzWildcard ) ;
```

Ее параметр **attr** определяет, файлы с какими атрибутами попадут в список. Значения параметра получаются комбинацией следующих значений:

0x0000	файлы доступные для чтения и записи;
0x0001	файлы, доступные только для чтения;
0x0002	скрытые файлы;
0x0004	системные файлы;
0x0010	подкаталоги;
0x0020	архивные файлы;
0x4000	имена дисков;
0x8000	флаг, который определяет, что в список должны попасть только специфицированные файлы, иначе специфицированные файлы будут дополнять список обычных файлов.

Второй параметр, **lpzWildcard**, задает маску для файлов (например **"*.cpp"**).

Удалить строку, определяемую индексом **nIndex**, из списка можно с помощью функции

```
int DeleteString ( UINT nIndex ) ;
```

Функция возвращает число оставшихся строк. Удалить все элементы списка можно функцией **ResetContent ()**.

Найти строку, содержащую в начале заданную подстроку, и выделить ее в списке позволяют функции

```
int FindString ( int nStartAfter , LPCTSTR lpzItem ) ;
```

```
int SelectString ( int nStartAfter , LPCTSTR lpzItem ) ;
```

Параметр **nStartAfter** – задает индекс строки, предшествующей началу поиска, по достижению конца списка поиск продолжается с его начала; выполнить поиск с начала можно, задав его значение **-1**; **lpzItem** – определяет задаваемую подстроку. Первая функция только возвращает индекс найденной строки, а вторая в дополнение к этому производит ее выделение.

Число элементов в списке определяет и возвращает функция **GetCount ()**. Определить, выбран ли пункт с заданным индексом, позволяет функция **GetSel ()**. Она возвращает положительное число, если пункт выбран, и нуль, если выбор не произведен. Индекс выбранной строки возвращает функция **GetCurSel ()**. Если выбора не было или список имеет множественный выбор, то возвращается значение **LB_ERR**. Программно выполнить выбор элемента списка с индексом **nIndex** можно с помощью функции

```
int SetSel ( int nIndex , BOOL bSelect = TRUE ) ;
```

Если второй параметр функции равен **TRUE** (по умолчанию), то производится выделение, а если – **FALSE**, то выделение снимается.

Для списка множественного выбора получить массив индексов выбранных элементов списка позволяет функция

```
int GetSelItems ( int nMaxItems , LPINT rgIndex ) ;
```

Она возвращает число выбранных элементов, параметр **nMaxItems** определяет размер буфера индексов, **rgIndex** – указатель на этот буфер.

Для получения текста выбранной строки служат следующие функции:

```
int GetText ( int nIndex , LPTSTR lpzBuffer ) const ;
```

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

Параметр **nIndex** является индексом требуемой строки; **lpzBuffer** – указатель на буфер – приемник строки (его размеры должны быть достаточны для размещения строки); **rString** – ссылка на объект **CString**.