Модуль №3

Занятие №3

Версия 1.0.1

План занятия:

1. Повторение пройденного материала.
2. Элемент управления «список» (List Box).
3. Сообщения списков.
4. Элемент управления «комбинированный список» (Combo Box).
5. Сообщения комбинированных списков.
6. Практическая часть.
7. Подведение итогов.
8. Домашнее задание.

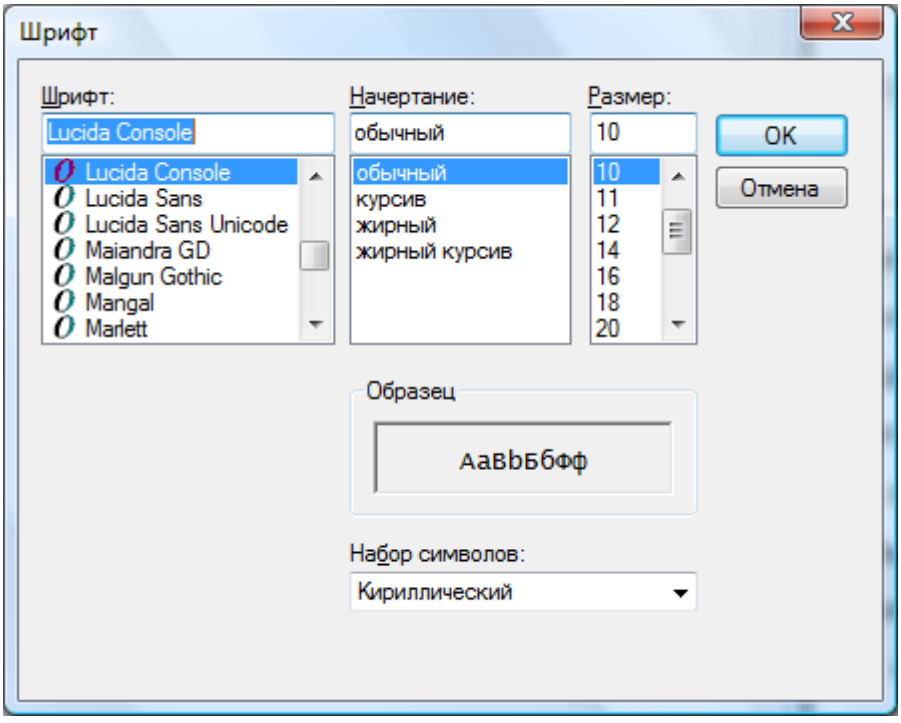
1. Повторение пройденного материала

Данное занятие необходимо начать с краткого повторения материала предыдущего занятия. При общении со слушателями можно использовать следующие контрольные вопросы:

1) Для чего обычно используют текстовые поля ввода?
2) Какими способами можно создать текстовое поле на диалоге?
3) Какое сообщение придёт в диалоговую процедуру при вводе текста в **Edit Control**?
4) Какое уведомление от текстового поля приходит родительскому окну (диалогу) при вводе текста?
5) Какие сообщения отправляются текстовому полю ввода при копировании, вырезании и удалении фрагмента текста?
6) Какое сообщение необходимо отправить текстовому полю, чтобы получить границы выделения текста?

7) Какое сообщение необходимо отправить текстовому полю, чтобы выделить фрагмент текста? Каким образом можно выделить весь текст?
8) Как определить длину текста, введённого в текстовое поле?
9) Какая функция позволяет установить, имеется ли текст в буфере обмена?
10) Какая функция позволяет изменить стиль элемента управления?
11) Какая функция позволяет переопределить стандартную оконную процедуру элемента управления?

2. Элемент управления «список» (List Box)



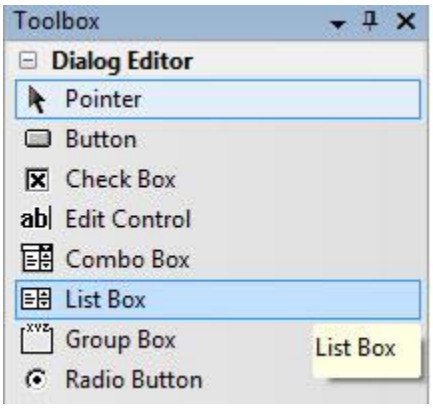
Повторив материал предыдущего занятия, ознакомить слушателей с элементом управления **List Box**. Отметить, что элемент управления **List Box** представляет собой список элементов, из которых пользователь может выбрать один или более. Элементы списка могут быть представлены строками, растровыми образами или комбинацией текста и изображения. Если размеры окна не

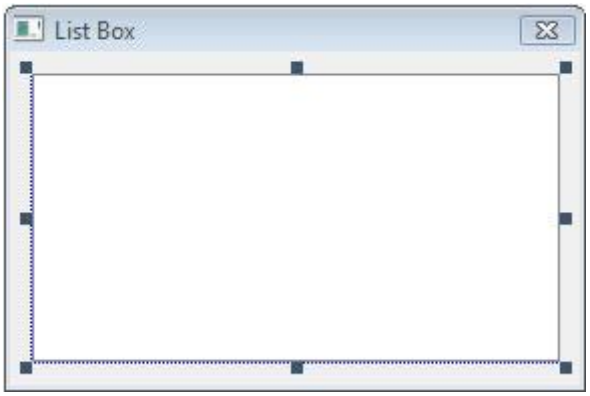
позволяют показать все элементы списка, то **List Box** создает полосу прокрутки. Типичный пример использования списка — список файлов и папок в диалоговом окне **«Открыть»** приложения «**Блокнот**».

Создать список на форме диалога можно двумя способами:

- с помощью средств интегрированной среды разработки **Microsoft Visual Studio**;
- посредством вызова функции **CreateWindowEx**.

При первом способе необходимо определить **List Box** в шаблоне диалогового окна на языке описания шаблона диалога. Это произойдёт автоматически, если активизировать окно **Toolbox** (<**Ctrl**><**Alt**><**X**>) и «перетащить» список на форму диалога.

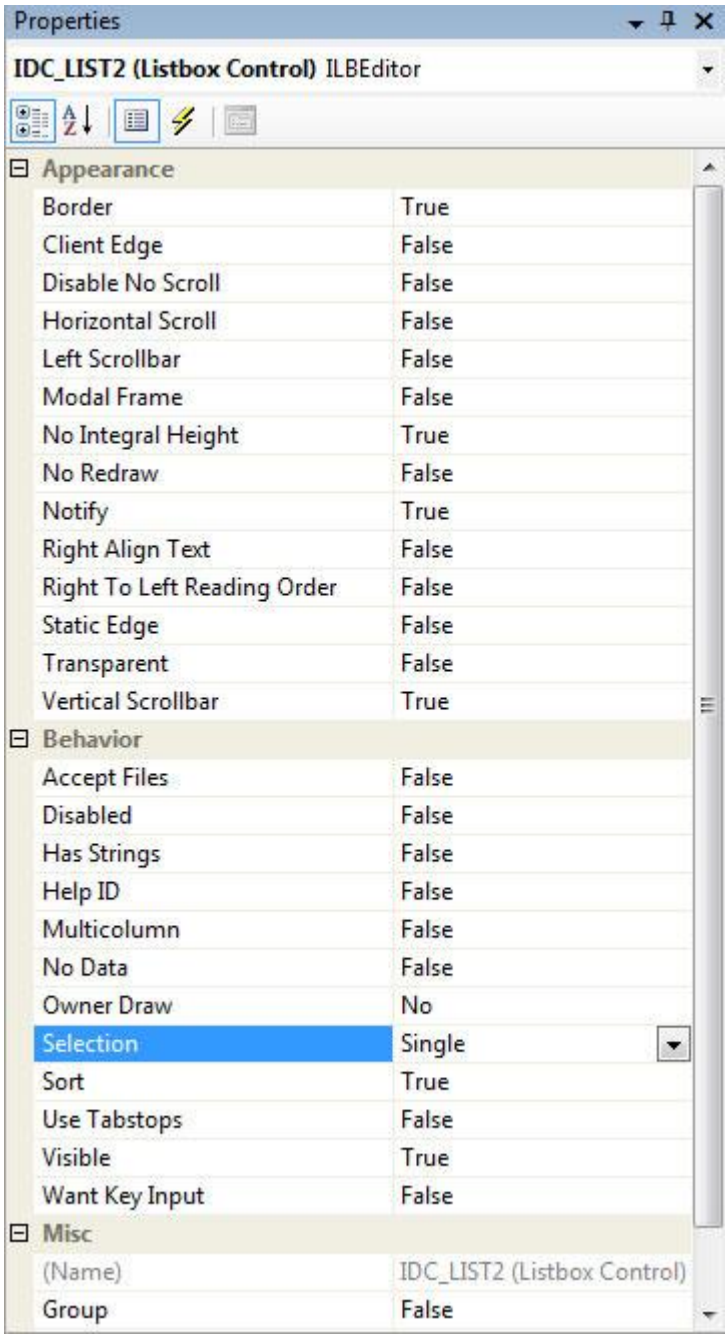




После размещения списка на форме диалога ему назначается идентификатор (например, **IDC_LIST1**), который впоследствии можно изменить на идентификатор, отражающий семантику ресурса.

Следует отметить, что различают списки с единичным выбором, в которых пользователь может выбрать только один пункт списка, и списки с множественным выбором, допускающие выбор более одного пункта списка. Система обычно отображает выбранный элемент в списке, инвертируя цвет текста и цвет фона для символов.

Необходимо ознакомить слушателей с некоторыми свойствами элемента управления **List Box**.



По умолчанию создается список с единичным выбором (свойство **Selection** со значением **Single**), имеющий рамку (свойство **Border** со значением **True**), выполняющий автоматическую сортировку элементов списка (истинное значение свойства **Sort**) и обеспечивающий появление вертикальной полосы прокрутки в случае необходимости (свойство **Vertical Scroll** со значением **True**).

Свойство **Multicolumn** позволяет располагать список в несколько столбцов. В этом случае **List Box** может прокручиваться по горизонтали. Но для этого свойство **Horizontal Scroll** должно иметь значение **True**.

3. Сообщения списков

Для выполнения различных операций со списком приложение может отправлять следующие сообщения элементу управления **List Box**:

Код сообщения	wParam	lParam	Описание
LB_ADDSTRING	0	szString	Добавить в список строку **szString**. Если свойство **Sort** у списка выключено, то строка будет добавлена в конец списка. Если это свойство включено, то после добавления строки производится сортировка списка
LB_FINDSTRING	iStart	szString	Найти строку, начинающуюся префиксом **szString**. Поиск начинается с элемента с индексом **iStart + 1**. Если **wParam** равен **-1**, то поиск начинается с начала списка. **SendMessage** возвращает индекс найденной строки или **LB_ERR**, если поиск завершился неуспешно
LB_GETCURSEL	0	0	Получить индекс текущего выбранного элемента или **LB_ERR**, если такого элемента нет (только для списков с единичным выбором)
LB_GETSELCOUNT	0	0	Получить количество выбранных элементов (для списка с множественным выбором)
LB_GETSELITEMS	nMax	pBuf	Заполнить буфер **pBuf** массивом индексов выделенных элементов для списка с множественным выбором. Параметр **nMax** задает максимальное количество таких элементов

Код сообщения	wParam	lParam	Описание
LB_INSERTSTRING	iIndex	szString	Вставить строку **szString** после строки с индексом **iIndex**. Применяется для списка с выключенным свойством **Sort**
LB_SELECTSTRING	iStart	szString	Аналогично сообщению **LB_FINDSTRING**, но дополнительно выделяется найденная строка
LB_SETSEL	wParam	iIndex	Выбрать элемент с индексом **iIndex** (в списке с множественным выбором). Если **wParam** равен **TRUE**, элемент выбирается и выделяется,если **FALSE** – выбор отменяется. Если **lParam** равен -1, то операция применяется ко всем элементам списка
LB_GETTEXT	iIndex	szString	Скопировать строку с индексом **iIndex** в буфер **szString**
LB_DELETESTRING	iIndex	0	Удалить строку с индексом **iIndex**.
LB_GETCOUNT	0	0	Получить количество элементов в списке
LB_GETTEXTLEN	iIndex	0	Получить длину строки с индексом iIndex
LB_RESETCONTENT	0	0	Удалить все элементы из списка
LB_SETCURSEL	iIndex	0	Выбрать элемент с индексом **iIndex** (в списке с единичным выбором)
LB_SETITEMDATA	iIndex	nValue	Установить целочисленное значение **nValue**, ассоциированное с указанным элементом списка
LB_GETITEMDATA	iIndex	0	Получить целочисленное значение, ассоциированное с элементом списка, имеющим индекс **iIndex**

Следует отметить, при воздействии на **List Box** (например, при выборе элемента списка), в диалоговую процедуру **DlgProc** поступает сообщение **WM_COMMAND,** в котором **LOWORD (wParam)** содержит идентификатор списка, **HIWORD (wParam)** содержит код уведомления (например, **LBN_SELCHANGE)**, а **lParam** – дескриптор списка.

Некоторые из кодов уведомления от списка приведены в таблице.

Код уведомления	Интерпретация
LBN_SETFOCUS	Окно получило фокус ввода
LBN_KILLFOCUS	Окно потеряло фокус ввода
LBN_SELCHANGE	Текущий выбор был изменен (свойство **Notify** должно иметь значение **True**)
LBN_DBLCLK	На данном пункте списка был двойной щелчок мыши (свойство **Notify** должно иметь значение **True**)
LBN_ERRSPACE	Превышен размер памяти, отведенный для списка

Далее следует привести пример, в котором рассматриваются сообщения, отправляемые списку для выполнения различных операций.

```
#include <windows.h>
#include <tchar.h>
#include "resource.h"

HWND hList1, hList2, hEdit1, hEdit2;

BOOL CALLBACK DlgProc(HWND, UINT, WPARAM, LPARAM);

int WINAPI _tWinMain(HINSTANCE hInstance, HINSTANCE hPrevInst,
                  LPTSTR lpszCmdLine, int nCmdShow)
{
      return DialogBox(hInstance, MAKEINTRESOURCE(IDD_DIALOG1), NULL,
                        DlgProc);
}

BOOL CALLBACK DlgProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
      switch(message)
      {
            case WM_CLOSE:
                  EndDialog(hWnd, 0);
                  return TRUE;

            case WM_INITDIALOG:
                  // получим дескрипторы элементов управления
                  hList1 = GetDlgItem(hWnd, IDC_LIST1);
                  hList2 = GetDlgItem(hWnd, IDC_LIST2);
                  hEdit1 = GetDlgItem(hWnd, IDC_EDIT1);
                  hEdit2 = GetDlgItem(hWnd, IDC_EDIT2);
                  // добавим 8 элементов в список с единичным выбором
                  SendMessage(hList1, LB_ADDSTRING, 0,
                              LPARAM(TEXT("Милан Италия")));
                  SendMessage(hList1, LB_ADDSTRING, 0,
                              LPARAM(TEXT("Ливерпуль Англия")));
                  SendMessage(hList1, LB_ADDSTRING, 0,
                              LPARAM(TEXT("Бавария Германия")));
```

```
            SendMessage(hList1, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Барселона Испания")));
            SendMessage(hList1, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Арсенал Англия")));
            SendMessage(hList1, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Лион Франция")));
            SendMessage(hList1, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Интер Италия")));
            SendMessage(hList1, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Реал Испания")));
          // установим ширину колонки для списка с множественным выбором
            SendMessage(hList2, LB_SETCOLUMNWIDTH, 170, 0);
            // добавим 8 элементов в список с множественным выбором
            SendMessage(hList2, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Рома Италия")));
            SendMessage(hList2, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Манчестер Юнайтед Англия")));
            SendMessage(hList2, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Вердер Германия")));
            SendMessage(hList2, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Валенсия Испания")));
            SendMessage(hList2, LB ADDSTRING, 0,
                        LPARAM(TEXT("Челси Англия")));
            SendMessage(hList2, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Марсель Франция")));
            SendMessage(hList2, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Ювентус Италия")));
            SendMessage(hList2, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Атлетико Испания")));
            return TRUE;

      case WM_COMMAND:
            switch(LOWORD(wParam))
            {
                  case IDC_ADD:
                  {
                        /* определим длину текста (названия
                        клуба), введённого в текстовое поле */
                        int length = SendMessage(hEdit1,
                                    WM_GETTEXTLENGTH, 0, 0);
                        // выделим память необходимого размера
                        TCHAR *pBuffer = new TCHAR[length + 1];
                        /* в выделенную память скопируем текст,
                        введённый в тестовое поле */
                        GetWindowText(hEdit1, pBuffer, length + 1);
                        if(lstrlen(pBuffer))
                        {
                              /* уточним, имеется ли уже в списке
                              введённое название клуба */
                              int index = SendMessage(hList1,
                              LB_FINDSTRINGEXACT, -1, LPARAM(pBuffer));
                              if(index == LB_ERR)
                              // добавим название клуба в список
                              SendMessage(hList1, LB_ADDSTRING, 0,
                                          LPARAM(pBuffer));
                              else
                                    MessageBox(hWnd,
                                    TEXT("Такой клуб уже существует!"),
```

```
                        TEXT("Добавление клуба"),
                        MB_OK | MB_ICONSTOP);
            }
            delete[] pBuffer;
      }
      break;
      case IDC_DEL:
      {
            // получим индекс выбранного элемента списка
            int index = SendMessage(hList1, LB_GETCURSEL,
                                    0, 0);
            if(index != LB_ERR) // выбран ли элемент списка?
            {
                // определим длину названия выбранного клуба
                  int length = SendMessage(hList1,
                              LB_GETTEXTLEN, index, 0);
                  // выделим память необходимого размера
                  TCHAR *pBuffer = new TCHAR[length + 1];
                 // скопируем название клуба в выделенную память
                  SendMessage(hList1, LB_GETTEXT, index,
                              LPARAM(pBuffer));
                  MessageBox(hWnd, pBuffer,
                  TEXT("Удаление клуба"),
                  MB_OK | MB_ICONINFORMATION);
                  // удалим строку из списка
                  SendMessage(hList1, LB_DELETESTRING,
                              index, 0);
                  delete[] pBuffer;
            }
            else
                  MessageBox(hWnd, TEXT("Клуб не выбран!"),
                  TEXT("Удаление клуба"),
                  MB_OK | MB_ICONSTOP);
      }
      break;
      case IDC_FIND:
      {
            /* определим длину искомого названия, введённого
            в текстовое поле */
            int length = SendMessage(hEdit2,
                        WM_GETTEXTLENGTH, 0, 0);
            // выделим память необходимого размера
            TCHAR *pBuffer = new TCHAR[length + 1];
            // скопируем название клуба в выделенную память
            GetWindowText(hEdit2, pBuffer, length + 1);
            if(lstrlen(pBuffer))
            {
                  /* определим, имеется ли в списке искомое
                  название клуба */
                  int index = SendMessage(hList1,
                  LB_SELECTSTRING, -1, LPARAM(pBuffer));
                  if(index == LB_ERR)
                   MessageBox(hWnd, TEXT("Клуб не найден!"),
                  TEXT("Поиск клуба"), MB_OK | MB_ICONSTOP);
            }
            delete[] pBuffer;
      }
      break;
```

```
		case IDC_DELALL:
			// очистим содержимое списка
			SendMessage(hList1, LB_RESETCONTENT, 0, 0);
			break;

		case IDC_GETSELITEMS:
		{
			/* определим количество выбранных элементов в
			списке с множественным выбором */
			int nCount = SendMessage(hList2, LB_GETSELCOUNT,
							0, 0);
			/* выделим память необходимого размера для
			хранения индексов выбранных элементов списка */
			int *p = new int[nCount];
			/* Заполним динамический массив индексами
			выделенных элементов списка */
			SendMessage(hList2, LB_GETSELITEMS, nCount,
					LPARAM(p));
			for(int i = 0; i < nCount; i++)
			{
				// определим размер текста элемента списка
				int length = SendMessage(hList2,
						LB_GETTEXTLEN, p[i], 0);
				// выделим память необходимого размера
				TCHAR *pBuffer = new TCHAR[length + 1];
				/* в выделенную память скопируем текст
				выбранного элемента списка */
				SendMessage(hList2, LB_GETTEXT, p[i],
				LPARAM(pBuffer));
				MessageBox(hWnd, pBuffer,
				TEXT("Список с множественным выбором"),
				MB_OK | MB_ICONINFORMATION);
				delete[] pBuffer;
			}
		}
	}
	// в списке с единичным выбором текущий выбор был изменён
	if(LOWORD(wParam) == IDC_LIST1 &&
		HIWORD(wParam) == LBN_SELCHANGE)
	{
		// получим индекс выбранного элемента списка
		int index = SendMessage(hList1, LB_GETCURSEL, 0, 0);
		if(index != LB_ERR) // выбран ли элемент списка?
		{
			// определим длину названия клуба
			int length = SendMessage(hList1, LB_GETTEXTLEN,
					index, 0);
			// выделим память необходимого размера
			TCHAR *pBuffer = new TCHAR[length + 1];
			/* в выделенную память скопируем текст
			выбранного элемента списка */
			SendMessage(hList1, LB_GETTEXT, index,
					LPARAM(pBuffer));
			// установим текст в заголовок главного окна
			SetWindowText(hWnd, pBuffer);
			delete[] pBuffer;
		}
	}
```

```
                return TRUE;
        }
        return FALSE;
}
```

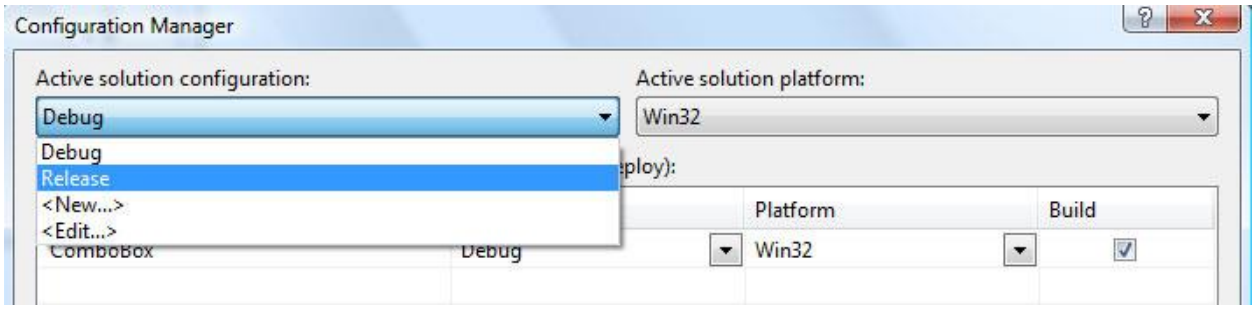
Альтернативный способ создания списка - использование функции **CreateWindowEx**, рассмотренной на одном из предыдущих занятий. В этом случае во втором параметре функции передается имя предопределенного оконного класса – **LISTBOX**.

В качестве примера, демонстрирующего программный способ создания списка, привести следующий фрагмент кода:

```
#define IDC_LIST  WM_USER
HWND hList = CreateWindowEx(WS_EX_CLIENTEDGE, TEXT("LISTBOX"), 0,
                        WS_CHILD | WS_VISIBLE | LBS_SORT | LBS_NOTIFY,
                        LEFT, TOP, WIDTH, HEIGHT, hWnd,
                        (HMENU)IDC_LIST /* идентификатор списка */,
                        hInstance, 0);
```

4. Элемент управления «комбинированный список» (Combo Box)

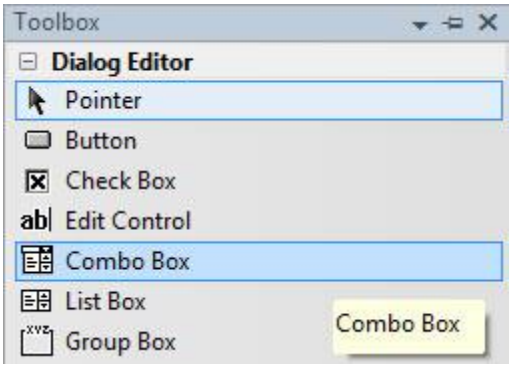
Комбинированный список (**Combo Box**) является комбинацией списка и текстового поля ввода. Типичный пример — окно «Configuration Manager» интегрированной среды разработки приложений **Microsoft Visual Studio**, обладающее комбинированным списком для выбора конфигурации проекта (**Debug** или **Release**).

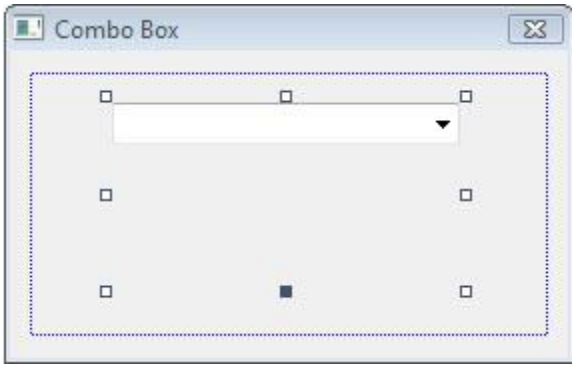


Создать **Combo Box** на форме диалога можно двумя способами:

- с помощью средств интегрированной среды разработки **Microsoft Visual Studio**;
- посредством вызова функции **CreateWindowEx**.

При первом способе необходимо определить **Combo Box** в шаблоне диалогового окна на языке описания шаблона диалога. Это произойдёт автоматически, если активизировать окно **Toolbox** (<**Ctrl**><**Alt**><**X**>) и «перетащить» комбинированный список на форму диалога.

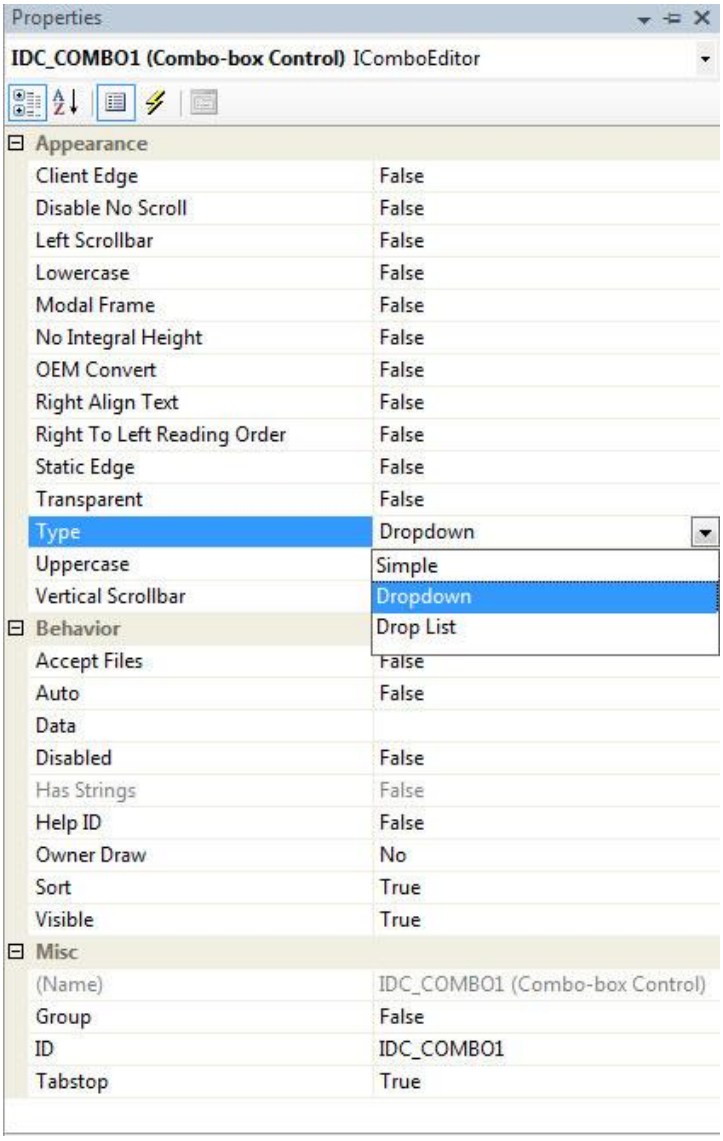




После размещения списка на форме диалога ему назначается идентификатор (например, **IDC_COMBO1**), который впоследствии можно изменить на идентификатор, отражающий семантику ресурса.

Следует отметить, что многие свойства **Combo Box** аналогичны свойствам элемента управления **List Box**. Но есть и некоторые различия. В частности, стиль комбинированного списка задаётся свойством **Type**, которое принимает одно из следующих значений:

Стиль	Описание
Simple	Комбинированный список отображается окном редактирования и раскрытым окном списка. Содержание окна редактирования можно изменять
Dropdown	Комбинированный список отображается в свернутом состоянии со стрелкой справа. Содержание окна редактирования можно изменять
Drop List	Комбинированный список отображается в свернутом состоянии со стрелкой справа. Содержание окна редактирования изменять нельзя



5. Сообщения комбинированных списков

Акцентировать внимание слушателей на том, что обмен сообщениями с комбинированным списком во многом похож на обмен сообщениями с элементами управления **Edit Box** и **List Box**. При этом идентификаторы сообщений, отправляемых элементу управления **Combo Box**, начинаются с префикса **CB_**, а идентификаторы уведомительных сообщений, посылаемых от **Combo Box** своему родительскому окну, имеют префикс **CBN_**.

Необходимо провести сравнительный анализ некоторых сообщений, посылаемых комбинированному списку, с аналогичными сообщениями, посылаемыми элементам управления **List Box** и **Edit Control**.

Код сообщения для Edit Control или List Box	**Код сообщения для Combo Box**
EM_GETSEL	CB_GETEDITSEL
EM_SETSEL	CB_SETEDITSEL
LB_ADDSTRING	CB_ADDSTRING
LB_DELETESTRING	CB_DELETESTRING
LB_GETCOUNT	CB_GETCOUNT
LB_GETCURSEL	CB_GETCURSEL
LB_GETTEXT	CB_GETLBTEXT
LB_SETCURSEL	CB_SETCURSEL

Далее следует провести аналогичный сравнительный анализ кодов уведомлений.

Код уведомления для Edit Control или List Box	**Код уведомления для Combo Box**
EN_SETFOCUS, LBN_SETFOCUS	CBN_SETFOCUS
EN_KILLFOCUS, LBN_KILLFOCUS	CBN_KILLFOCUS
EN_UPDATE	CBN_EDITUPDATE
EN_CHANGE	CBN_EDITCHANGE
EN_ERRSPACE, LBN_ERRSPACE	CBN_ERRSPACE
LBN_SELCHANGE	CBN_SELCHANGE

В качестве примера, демонстрирующего использование комбинированного списка, привести следующий код:

```
#include <windows.h>
#include <tchar.h>
#include "resource.h"

HWND hList, hCombo, hCheck, hEdit;

BOOL CALLBACK DlgProc(HWND, UINT, WPARAM, LPARAM);

int WINAPI _tWinMain(HINSTANCE hInstance, HINSTANCE hPrevInst,
                  LPTSTR lpszCmdLine, int nCmdShow)
{
      return DialogBox(hInstance, MAKEINTRESOURCE(IDD_DIALOG1), NULL,
                        DlgProc);
}

BOOL CALLBACK DlgProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
      switch(message)
      {
      case WM_CLOSE:
            EndDialog(hWnd, 0);
            return TRUE;

      case WM_INITDIALOG:
            // Получим дескрипторы элементов управления
            hCombo = GetDlgItem(hWnd, IDC_SEASONS);
            hList = GetDlgItem(hWnd, IDC_MONTH);
            hEdit = GetDlgItem(hWnd, IDC_DAYS);
            hCheck = GetDlgItem(hWnd, IDC_LEAF);

            // Добавляем времена года в ComboBox
            SendMessage(hCombo, CB_ADDSTRING, 0, LPARAM(TEXT("Зима")));
            SendMessage(hCombo, CB_ADDSTRING, 0, LPARAM(TEXT("Весна")));
            SendMessage(hCombo, CB_ADDSTRING, 0, LPARAM(TEXT("Лето")));
            SendMessage(hCombo, CB_ADDSTRING, 0, LPARAM(TEXT("Осень")));
            return TRUE;

      case WM_COMMAND:
            // событие CBN_SELCHANGE выпадающего списка происходит в тот
            // момент, когда выбирается новый элемент в списке
            if(LOWORD(wParam) == IDC_SEASONS &&
            HIWORD(wParam) == CBN_SELCHANGE)
            {
                  // Очистим ListBox
                  SendMessage(hList, LB_RESETCONTENT, 0, 0);
                  // Получим индекс выбранного элемента ComboBox
                  int index = SendMessage(hCombo, CB_GETCURSEL, 0, 0);
                  switch(index)
                  {
                        case 0:
                        // Добавляем в ListBox название месяца и запоминаем
                        // индекс добавленного элемента
```

```
                        index = SendMessage(hList, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Декабрь")));
                        // Ставим элементу списка в соответствие число,
                        // определяющее количество дней в месяце
                        SendMessage(hList, LB_SETITEMDATA, index, 31);
                        index = SendMessage(hList, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Январь")));
                        SendMessage(hList, LB_SETITEMDATA, index, 31);
                        index = SendMessage(hList, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Февраль")));
                        SendMessage(hList, LB_SETITEMDATA, index, 28);
                        break;

                  case 1:
                        index = SendMessage(hList, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Март")));
                        SendMessage(hList, LB_SETITEMDATA, index, 31);
                        index = SendMessage(hList, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Апрель")));
                        SendMessage(hList, LB_SETITEMDATA, index, 30);
                        index = SendMessage(hList, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Май")));
                        SendMessage(hList, LB_SETITEMDATA, index, 31);
                        break;

                  case 2:
                        index = SendMessage(hList, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Июнь")));
                        SendMessage(hList, LB_SETITEMDATA, index, 30);
                        index = SendMessage(hList, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Июль")));
                        SendMessage(hList, LB_SETITEMDATA, index, 31);
                        index = SendMessage(hList, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Август")));
                        SendMessage(hList, LB_SETITEMDATA, index, 31);
                        break;

                  case 3:
                        index = SendMessage(hList, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Сентябрь")));
                        SendMessage(hList, LB_SETITEMDATA, index, 30);
                        index = SendMessage(hList, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Октябрь")));
                        SendMessage(hList, LB_SETITEMDATA, index, 31);
                        index = SendMessage(hList, LB_ADDSTRING, 0,
                        LPARAM(TEXT("Ноябрь")));
                        SendMessage(hList, LB_SETITEMDATA, index, 30);
                        break;
            }
      }

      // событие LBN_SELCHANGE списка происходит в тот момент, когда
      // выбирается новый элемент в списке
      if(LOWORD(wParam) == IDC_MONTH && HIWORD(wParam) == LBN_SELCHANGE)
      {
            // Получим индекс выбранного элемента ListBox
            int index = SendMessage(hList, LB_GETCURSEL, 0, 0);
            // Получим значение, связанное с выбранным элементом списка
            int day = SendMessage(hList, LB_GETITEMDATA, index, 0);
```

```
                TCHAR buffer[10];
                // Снимем текст с выбранного элемента списка
                SendMessage(hList, LB_GETTEXT, index, LPARAM(buffer));
                // Выбранный месяц - Февраль?
                if(!lstrcmp(buffer, TEXT("Февраль")))
                {
                      // Проверяем, високосный год или нет
                      int status = SendMessage(hCheck, BM_GETCHECK, 0, 0);
                      if(status == BST_CHECKED)
                            day++;
                }
                wsprintf(buffer, TEXT("%d"), day);
                // Устанавливаем количество дней в EditControl
                SendMessage(hEdit, WM_SETTEXT, 0, LPARAM(buffer));
          }
          return TRUE;
     }
     return FALSE;
}
```

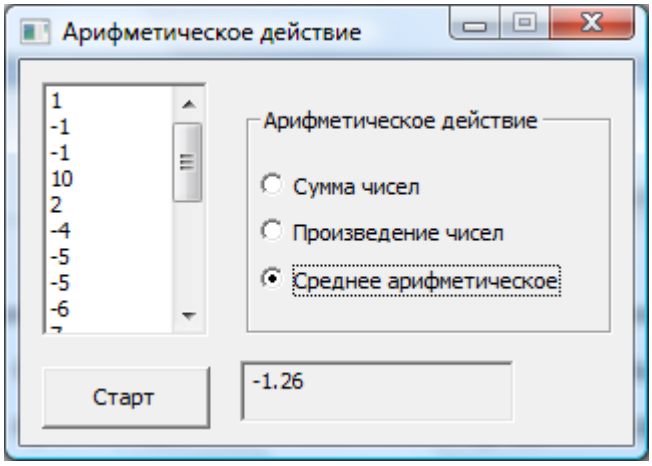
Альтернативный способ создания комбинированного списка - использование функции **CreateWindowEx**, рассмотренной на одном из предыдущих занятий. В этом случае во втором параметре функции передается имя предопределенного оконного класса – **COMBOBOX**.

В качестве примера, демонстрирующего программный способ создания комбинированного списка, привести следующий фрагмент кода:

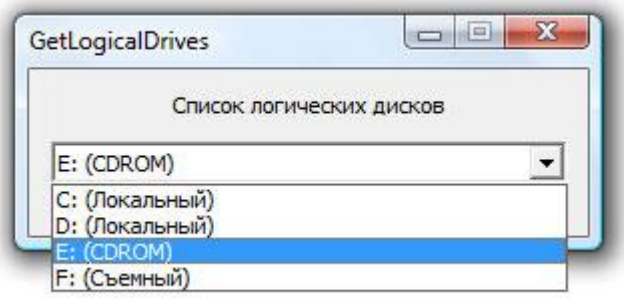
```
HWND hList = CreateWindowEx(WS_EX_DLGMODALFRAME, TEXT("COMBOBOX"), 0,
                  WS_CHILD | WS_VISIBLE | CBS_SORT | CBS_DROPDOWNLIST,
                  LEFT, TOP, WIDTH, HEIGHT, hWnd, 0, hInstance, 0);
```

6. Практическая часть

1. Создать диалоговое приложение, у которого на форме находится список, кнопка, текстовое поле и три переключателя. По кнопке «Старт» в список заносится от 10 до 20 случайных чисел в диапазоне от -10 до 10 (0 не включается). Переключатели обеспечивают отображение пользователю суммы чисел, произведения чисел либо среднего арифметического чисел.



2. Написать приложение, формирующее список логических дисков, которые доступны в данный момент.



При разработке приложения рекомендуется воспользоваться следующими функциями API:

```
DWORD WINAPI GetLogicalDrives(void);
```

Функция **GetLogicalDrives** возвращает битовую маску логических дисков, которые доступны в данный момент. При этом каждый бит маски представляет собой конкретный диск. Если бит установлен, то логический диск доступен, в противном случае бит будет сброшен. Бит 0 соответствует диску A, бит 1 – диску B, бит 2 – диску C и т.д.

```
UINT WINAPI GetDriveType( LPCTSTR lpRootPathName );
```

Функция **GetDriveType** возвращает тип накопителя по заданному имени корневого пути (например, "**C:**"). Перечень типов накопителей приведён в следующей таблице.

Возвращаемый код	Тип накопителя
DRIVE_UNKNOWN	Неизвестный тип накопителя
DRIVE_NO_ROOT_DIR	Корневой каталог не существует
DRIVE_REMOVABLE	Съемный диск
DRIVE_FIXED	Жесткий диск
DRIVE_REMOTE	Сетевой диск
DRIVE_CDROM	Компакт-диск
DRIVE_RAMDISK	Виртуальный диск

```
DWORD WINAPI GetLogicalDriveStrings(
  __in DWORD nBufferLength, // максимальный размер буфера, на который
// указывает второй параметр
  __out LPTSTR lpBuffer // указатель на буфер, который принимает набор строк,
// завершающихся нулевым символом, причем на каждый действительный логический
// диск в системе приходится по одной строке
);
```

Функция **GetLogicalDriveStrings** заполняет буфер строками, которые определяют действительные логические диски в системе.

7. Подведение итогов

Подвести общие итоги занятия. Отметить отличие между обычным списком и комбинированным. Подчеркнуть основные способы создания списков – с помощью средств интегрированной среды разработки приложений, а также посредством функции CreateWindowEx. Выделить сообщения списков, используемые наиболее часто. Акцентировать внимание слушателей на наиболее тонких моментах изложенной темы.

8. Домашнее задание

Создать диалоговое приложение, у которого на форме имеется выпадающий список с названиями элементов управления (BUTTON, EDIT, LISTBOX, COMBOBOX, STATIC), а также две кнопки – «Создать» и «Удалить». При нажатии на кнопку «Создать» на форме динамически создается элемент управления, выбранный из списка. При этом координаты расположения элемента управления генерируются случайным образом. При нажатии на кнопку «Удалить» уничтожается элемент управления, выбранный из списка. Причём удаляться элементы управления должны в порядке, обратном их созданию, т.е. элемент управления, созданный последним, удаляться должен первым. При написании приложения рекомендуется использовать библиотеку STL.