|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***«Информатика и управление»***

**КАФЕДРА**  ***«Информационные системы и сети»***

**ЛАБОРАТОРНАЯ РАБОТА № 3**

**«**Работа с элементами управления List Box и Combo Box**»**

**ДИСЦИПЛИНА: «Программирование в среде Windows »**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ЭВМ.Б-51 | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Лацин С. М. )  (Подпись) (Ф.И.О.) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Родионов А. В. )  (Подпись) (Ф.И.О.) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2019

**Цель работы:** изучить основные функции компонентов CListBox и CComboBox. Освоить приёмы работы с данными элементами

**Теоретическая часть.**

**Класс CListBox**

Класс ***CListBox*** производный от класса ***CWnd***:

class ***CListBox*** : public ***CWnd***

Элемент управления **List Box** выводит на экран список элементов, которые пользователь может просмотреть и выбрать.

В поле простого списка пользователь может выбрать только один элемент. В поле списка со множественным выбором может быть выбран диапазон элементов. Когда пользователь выбирает элемент, он выделен, и поле списка отправляет уведомление в родительское окно.

Возможно, создание поле списка или из диалогового шаблона или непосредственно в коде программы. Чтобы создать его непосредственно необходимо создать объект ***CListBox***, затем вызвать ***Create*** функцию класса, чтобы создать управление поля списка Windows и присоединить его к объекту ***CListBox***. Чтобы использовать поле списка в диалоговом шаблоне, необходимо объявить переменную поля списка в своем классе диалогового окна, затем используйте ***DDX\_Control*** в классе диалогового окна функцию ***DoDataExchange***, чтобы соединить задействованную переменную с управлением.

**Класс CComboBox**

Класс ***CComboBox*** производный от класса ***CWnd***:

class CComboBox : public CWnd

Поле Combo Box состоит из поля списка, объединенного или со статическим элементом управления или со средствами редактирования. Часть поля списка управления может быть выведена на экран в любом случае, или может только выпадающий, когда пользователь выбирает стрелку выпадающего списка рядом с управлением.

В настоящее время выбранный пункт (если таковые имеются) в поле списка выведен на экран в статическом элементе управления или средствах редактирования. Кроме того, если у поля комбинированного списка есть стиль выпадающего списка, пользователь может ввести начальный символ одного из элементов в списке, и поле списка, если видимый, выделит следующий элемент с тем начальным символом.

**Основные функции члены классов CListBox и CComboBox**

1. ***AddString*** – добавление записи в список

int AddString(LPCTSTR lpszItem );

lpszItem – строка, которая должна быть добавлена

Возвращаемое значение - индекс строки в поле списка (первый элемент списка имеет индекс 0). Если возвращаемое значение - LB\_ERR, то произошла ошибка.

1. DeleteString - eдаляет элемент в позиции nIndex списка.

int DeleteString(UINT nIndex );

nIndex - определяет индекс строки, которая будет удалена.

Возвращаемое значение - количество строк, остающихся в списке. Возвращаемое значение - LB\_ERR, если nIndex определяет индекс больше чем число элементов в списке.

1. GetCount - получает число элементов в поле списка.

int GetCount() const;

Возвращаемое значение - число элементов в поле списка или LB\_ERR, если ошибка.

1. GetCurSel - получает индекс выбранного элемента, если таковой имеется.

int GetCurSel( ) const;

Возвращаемое значение - индекс выбранного пункта, если это поле простого списка. Это - LB\_ERR, если никакой элемент в настоящее время не выбран.

1. GetItemData - получает значение двойного слова, связанное с указанным элементом поля списка.

DWORD\_PTR GetItemData(int nIndex ) const;

nIndex - определяет индекс элемента в поле списка.

Возвращаемое значение - 32-разрядное значение связанное с элементом или LB\_ERR, если ошибка.

1. GetText - получает строку от поля списка.

int GetText(int nIndex, LPTSTR lpszBuffer ) const;

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

nIndex - индекс строки, которая будет получена.

lpszBuffer – адрес буфера, который получает строку. У буфера должно быть достаточное пространство для строки и завершающегося нулевого символа. Размер строки может быть определен загодя, вызывая функцию членства GetTextLen.

rString - ссылка на объект CString.

Возвращаемое значение - длина (в байтах) строки, исключая завершающийся нулевой символ. Если nIndex не определяет допустимый индекс, возвращаемое значение - LB\_ERR.

1. GetTextLen - получает длину строки в элементе поля списка.

int GetTextLen( int nIndex ) const;

nIndex - индекс строки.

Возвращаемое значение - длина строки в символах, исключая завершающий нулевой символ. Если nIndex не определяет допустимый индекс, возвращаемое значение - LB\_ERR.

1. InsertString - вставляет строку в поле списка.

int InsertString(int nIndex, LPCTSTR lpszItem );

nIndex - индекс позиции перед которой будет вставлена строка. Если этот параметр –1, строка добавлена в конец списка.

lpszItem - адрес строки с завершающим нулем, которая должна быть вставлена.

Возвращаемое значение - индекс позиции, в которой была вставлена строка. Если возвращаемое значение - LB\_ERR, то произошла ошибка.

1. SetItemData - устанавливает 32-разрядное значение, связанное с указанным элементом в поле списка.

int SetItemData(int nIndex,DWORD\_PTR dwItemData );

nIndex - индекс элемента.

dwItemData - значение, которое будет связано с элементом.

**Практическая часть.**

**Листинг программы:**

// LABA3 MFCDlg.cpp: файл реализации

//

#include "pch.h"

#include "framework.h"

#include "LABA3 MFC.h"

#include "LABA3 MFCDlg.h"

#include "afxdialogex.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// Диалоговое окно CLABA3MFCDlg

CLABA3MFCDlg::CLABA3MFCDlg(CWnd\* pParent /\*=nullptr\*/)

: CDialog(IDD\_LABA3MFC\_DIALOG, pParent)

, m\_sinfo(\_T(""))

, m\_sChange(\_T(""))

{

m\_hIcon = AfxGetApp()->LoadIcon(IDR\_MAINFRAME);

}

void CLABA3MFCDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialog::DoDataExchange(pDX);

DDX\_Control(pDX, IDC\_LIST1, m\_List);

DDX\_Text(pDX, IDC\_INFO, m\_sinfo);

DDX\_Control(pDX, IDC\_COMBO1, m\_Combo);

DDX\_Text(pDX, IDC\_HZ, m\_sChange);

}

BEGIN\_MESSAGE\_MAP(CLABA3MFCDlg, CDialog)

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

ON\_LBN\_SELCHANGE(IDC\_LIST1, &CLABA3MFCDlg::OnLbnSelchangeList1)

ON\_CBN\_SELCHANGE(IDC\_COMBO1, &CLABA3MFCDlg::OnCbnSelchangeCombo1)

ON\_BN\_CLICKED(IDC\_BUTTON1, &CLABA3MFCDlg::OnBnClickedButton1)

ON\_BN\_CLICKED(IDC\_BUTTON2, &CLABA3MFCDlg::OnBnClickedButton2)

ON\_BN\_CLICKED(IDC\_BUTTON3, &CLABA3MFCDlg::OnBnClickedButton3)

ON\_BN\_CLICKED(IDC\_BUTTON4, &CLABA3MFCDlg::OnBnClickedButton4)

END\_MESSAGE\_MAP()

// Обработчики сообщений CLABA3MFCDlg

BOOL CLABA3MFCDlg::OnInitDialog()

{

CDialog::OnInitDialog();

// Задает значок для этого диалогового окна. Среда делает это автоматически,

// если главное окно приложения не является диалоговым

SetIcon(m\_hIcon, TRUE); // Крупный значок

SetIcon(m\_hIcon, FALSE); // Мелкий значок

// TODO: добавьте дополнительную инициализацию

m\_List.AddString("Red");

m\_List.AddString("Orange");

m\_List.AddString("Yellow");

m\_List.AddString("Green");

m\_List.AddString("Cyan");

m\_List.AddString("Blue");

m\_List.AddString("Purple");

m\_Combo.AddString("Red");

m\_Combo.AddString("Orange");

m\_Combo.AddString("Yellow");

m\_Combo.AddString("Green");

m\_Combo.AddString("Cyan");

m\_Combo.AddString("Blue");

m\_Combo.AddString("Purple");

return TRUE; // возврат значения TRUE, если фокус не передан элементу управления

}

// При добавлении кнопки свертывания в диалоговое окно нужно воспользоваться приведенным ниже кодом,

// чтобы нарисовать значок. Для приложений MFC, использующих модель документов или представлений,

// это автоматически выполняется рабочей областью.

void CLABA3MFCDlg::OnPaint()

{

if (IsIconic())

{

CPaintDC dc(this); // контекст устройства для рисования

SendMessage(WM\_ICONERASEBKGND, reinterpret\_cast<WPARAM>(dc.GetSafeHdc()), 0);

// Выравнивание значка по центру клиентского прямоугольника

int cxIcon = GetSystemMetrics(SM\_CXICON);

int cyIcon = GetSystemMetrics(SM\_CYICON);

CRect rect;

GetClientRect(&rect);

int x = (rect.Width() - cxIcon + 1) / 2;

int y = (rect.Height() - cyIcon + 1) / 2;

// Нарисуйте значок

dc.DrawIcon(x, y, m\_hIcon);

}

else

{

CDialog::OnPaint();

}

}

// Система вызывает эту функцию для получения отображения курсора при перемещении

// свернутого окна.

HCURSOR CLABA3MFCDlg::OnQueryDragIcon()

{

return static\_cast<HCURSOR>(m\_hIcon);

}

void CLABA3MFCDlg::OnLbnSelchangeList1()

{

int n = m\_List.GetCurSel();

m\_Combo.SetCurSel(n);

m\_List.GetText(n, m\_sinfo);

UpdateData(FALSE);

}

void CLABA3MFCDlg::OnCbnSelchangeCombo1()

{

int n = m\_List.GetCurSel();

m\_Combo.SetCurSel(n);

m\_List.GetText(n, m\_sinfo);

UpdateData(FALSE);

}

void CLABA3MFCDlg::OnBnClickedButton1()

{

// Delete color

UpdateData(TRUE);

int n = m\_List.GetCurSel();

if (n != LB\_ERR)

{

m\_List.DeleteString(n);

m\_Combo.DeleteString(n);

if (n >= m\_List.GetCount()) n = m\_List.GetCount() - 1;

if (n >= 0)

{

m\_Combo.SetCurSel(n);

m\_List.SetCurSel(n);

m\_List.GetText(n, m\_sinfo);

}

else

{

m\_sinfo.Empty();

m\_Combo.RedrawWindow();

}

UpdateData(FALSE);

}

}

void CLABA3MFCDlg::OnBnClickedButton2()

{

// Add color

UpdateData(TRUE);

if (m\_sChange.GetLength() == 0)

{

MessageBox("Enter colors, please!");

}

else

{

m\_List.AddString(m\_sChange);

m\_Combo.AddString(m\_sChange);

}

UpdateData(FALSE);

}

void CLABA3MFCDlg::OnBnClickedButton3()

{

// Insert element

UpdateData(TRUE);

if (m\_sChange.GetLength() == 0)

{

MessageBox("Enter color, please!");

}

else

{

m\_List.InsertString(m\_List.GetCurSel(), m\_sChange);

m\_Combo.InsertString(m\_Combo.GetCurSel(), m\_sChange);

}

UpdateData(FALSE);

}

void CLABA3MFCDlg::OnBnClickedButton4()

{

// Change color

UpdateData(TRUE);

if (m\_sChange.GetLength() == 0)

{

MessageBox("Enter color, please!");

}

else

{

m\_List.InsertString(m\_List.GetCurSel(), m\_sChange);

m\_List.DeleteString(m\_List.GetCurSel());

m\_Combo.InsertString(m\_Combo.GetCurSel(), m\_sChange);

m\_Combo.DeleteString(m\_Combo.GetCurSel());

int n = m\_List.GetCurSel();

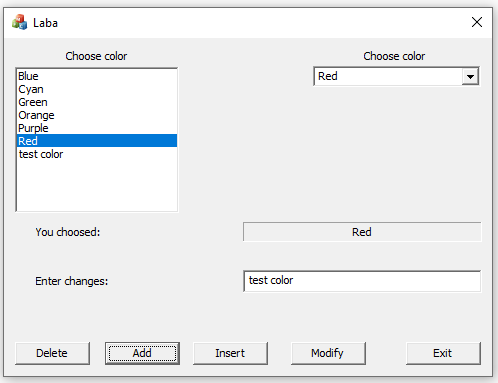
m\_Combo.SetCurSel(n + 1);

m\_List.GetText(n + 1, m\_sChange);

}

UpdateData(FALSE);

}

**Работа программы:**

**Вывод:** были изучены основные функции компонентов CListBox и CComboBox. Освоены приёмы работы с данными элементами.