***Министерство образования Республики Беларусь***

***Учреждение образования***

***«Брестский государственный технический университет»***

***Кафедра ИИТ***

**Лабораторная работа №3**

**По дисциплине ППвИС за IV семестр**

**Тема: «Каркасы. Создание интерфейсов. Диалоговые окна»**

**Выполнил:**

Студент группы ИИ-15 (1)

2-го курса

Волк И. А.

**Проверил:**

Муравьев Г.Л.

Брест 2019

Цель:

* Изучить создание Windows-приложений с использованием каркаса Hello.
* Изучить технологию работы с диалоговыми окнами и элементами управления (ЭУ). Познакомиться с применением диалогового окна в составе главного, в качестве главного окна.
* Изучить использование кнопок, окон редактирования.

Состав отчета:

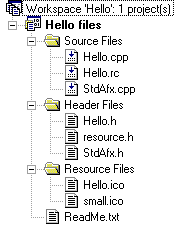
1. Описание каркаса Hello:
   1. Состав интерфейса
   2. Файловый состав
   3. Функциональный состав
   4. Диаграммы состояний, компонентов
2. Диаграммы состояний, листинги.
3. Выводы

Ход работы:

**Каркас Hello**

1. Изучим теоретический материал по каркасу Hello.
2. Создадим каркас Hello средствами мастера. Проект назовём Hello. Изучим его.

Файловый состав (дерево):



Листинг Hello.cpp:

// Hello.cpp : Defines the entry point for the application.

//

#include "stdafx.h"

#include "resource.h"

#define MAX\_LOADSTRING 100

// Global Variables:

HINSTANCE hInst; // current instance

TCHAR szTitle[MAX\_LOADSTRING]; // The title bar text

TCHAR szWindowClass[MAX\_LOADSTRING]; // The title bar text

// Foward declarations of functions included in this code module:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

LRESULT CALLBACK About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY WinMain(HINSTANCE hInstance,

HINSTANCE hPrevInstance,

LPSTR lpCmdLine,

int nCmdShow)

{

// TODO: Place code here.

MSG msg;

HACCEL hAccelTable;

// Initialize global strings

LoadString(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadString(hInstance, IDC\_HELLO, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Perform application initialization:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

hAccelTable = LoadAccelerators(hInstance, (LPCTSTR)IDC\_HELLO);

// Main message loop:

while (GetMessage(&msg, NULL, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return msg.wParam;

}

//

// FUNCTION: MyRegisterClass()

//

// PURPOSE: Registers the window class.

//

// COMMENTS:

//

// This function and its usage is only necessary if you want this code

// to be compatible with Win32 systems prior to the 'RegisterClassEx'

// function that was added to Windows 95. It is important to call this function

// so that the application will get 'well formed' small icons associated

// with it.

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEX wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = (WNDPROC)WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, (LPCTSTR)IDI\_HELLO);

wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);

wcex.lpszMenuName = (LPCSTR)IDC\_HELLO;

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, (LPCTSTR)IDI\_SMALL);

return RegisterClassEx(&wcex);

}

//

// FUNCTION: InitInstance(HANDLE, int)

//

// PURPOSE: Saves instance handle and creates main window

//

// COMMENTS:

//

// In this function, we save the instance handle in a global variable and

// create and display the main program window.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

HWND hWnd;

hInst = hInstance; // Store instance handle in our global variable

hWnd = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

//

// FUNCTION: WndProc(HWND, unsigned, WORD, LONG)

//

// PURPOSE: Processes messages for the main window.

//

// WM\_COMMAND - process the application menu

// WM\_PAINT - Paint the main window

// WM\_DESTROY - post a quit message and return

//

//

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

int wmId, wmEvent;

PAINTSTRUCT ps;

HDC hdc;

TCHAR szHello[MAX\_LOADSTRING];

LoadString(hInst, IDS\_HELLO, szHello, MAX\_LOADSTRING);

switch (message)

{

case WM\_COMMAND:

wmId = LOWORD(wParam);

wmEvent = HIWORD(wParam);

// Parse the menu selections:

switch (wmId)

{

case IDM\_ABOUT:

DialogBox(hInst,(LPCTSTR)IDD\_ABOUTBOX,hWnd, (DLGPROC)About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

break;

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

// TODO: Add any drawing code here...

RECT rt;

GetClientRect(hWnd, &rt);

DrawText(hdc, szHello, strlen(szHello), &rt, DT\_CENTER);

EndPaint(hWnd, &ps);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

// Mesage handler for about box.

LRESULT CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_INITDIALOG:

return TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return TRUE;

}

break;

}

return FALSE;

}

Структура Hello.cpp:



Рисунок 2.1 – Структура файла Hello.cpp

Насколько видно из листинга, основной код приложения включается в себя 5 функций, приведенных на рисунке 2.1. WinMain – главная функция Windows-приложения. Две из функций являются вспомогательными, а именно: MyRegisterClass и InitInstance. Первый служит для регистрации класса окна, второй создает и выводит окно, а также сохраняет hInstance в глобальную переменную. WndProc – функция обработчик сообщений главного окна. About – функция обработчик сообщений диалогового окна.

StdAfx.h содержит стандартные заголовки, включаемые в наше приложение. Листинг:

// stdafx.h : include file for standard system include files,

// or project specific include files that are used frequently, but

// are changed infrequently

//

#if !defined(AFX\_STDAFX\_H\_\_A9DB83DB\_A9FD\_11D0\_BFD1\_444553540000\_\_INCLUDED\_)

#define AFX\_STDAFX\_H\_\_A9DB83DB\_A9FD\_11D0\_BFD1\_444553540000\_\_INCLUDED\_

#if \_MSC\_VER > 1000

#pragma once

#endif // \_MSC\_VER > 1000

#define WIN32\_LEAN\_AND\_MEAN // Exclude rarely-used stuff from Windows headers

// Windows Header Files:

#include <windows.h>

// C RunTime Header Files

#include <stdlib.h>

#include <malloc.h>

#include <memory.h>

#include <tchar.h>

// Local Header Files

// TODO: reference additional headers your program requires here

//{{AFX\_INSERT\_LOCATION}}

// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX\_STDAFX\_H\_\_A9DB83DB\_A9FD\_11D0\_BFD1\_444553540000\_\_INCLUDED\_)

Файл resource.h содержит определения идентификаторов ресурсов, которые принадлежат нашем приложению:

//{{NO\_DEPENDENCIES}}

// Microsoft Visual C++ generated include file.

// Used by HELLO.RC

//

#define IDR\_MAINFRAME 128

#define IDD\_HELLO\_DIALOG 102

#define IDD\_ABOUTBOX 103

#define IDS\_APP\_TITLE 103

#define IDM\_ABOUT 104

#define IDM\_EXIT 105

#define IDS\_HELLO 106

#define IDI\_HELLO 107

#define IDI\_SMALL 108

#define IDC\_HELLO 109

#define IDC\_MYICON 2

#define IDC\_STATIC -1

// Next default values for new objects

//

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NEXT\_RESOURCE\_VALUE 129

#define \_APS\_NEXT\_COMMAND\_VALUE 32771

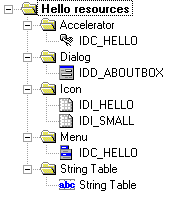
#define \_APS\_NEXT\_CONTROL\_VALUE 1000

#define \_APS\_NEXT\_SYMED\_VALUE 110

#endif

#endif

Файл Hello.rc содержит описания ресурсов. В среде разработки Visual C++ 6.0 содержание данного файла можно посмотреть в окне Resource View:

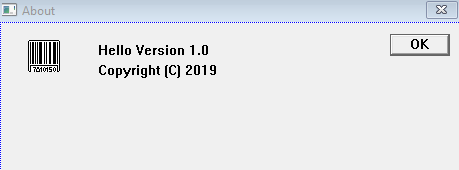


Проведем более детальный обзор ресурсов Windows-приложения:

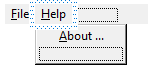
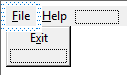
* Горячие клавиши. В нашем случае клавиши для вызова диалогового окна



* Диалоговое окно



* Меню



* Таблица строк. Тут храниться название класса, заглавие, а также выводимое сообщение



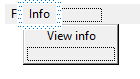
Запустим приложение.



Приложение представляет собой одно окно, состоящее из меню и сообщения Hello World! посреди клиентской области.

1. Изменим каркас:

* названия пунктов меню



* название главного окна



* текст в окне справке



* по нажатию OK в окне справки вывод MessageBox с запросом на подтверждение выхода:

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

if(LOWORD(wParam) == IDOK)

{

if(MessageBox(hDlg, "Do you really want to exit?", "Confirmation",

MB\_OKCANCEL) == IDCANCEL)

break;

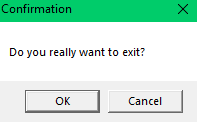
}

EndDialog(hDlg, LOWORD(wParam));

return TRUE;

}

break;



* по завершению работы приложения запрашивать подтверждение (аналогично с предыдущим пунктом). Рассматриваются две ситуации: выход через меню:

case WM\_COMMAND:

…

case IDM\_EXIT:

if(MessageBox(hWnd, "Do you really want to exit?", "Confirmation",

MB\_OKCANCEL) != IDCANCEL)

DestroyWindow(hWnd);

break;

И выход по нажатию на крестик:

case WM\_CLOSE:

if(MessageBox(hWnd, "Do you really want to exit?", "Confirmation",

MB\_OKCANCEL) != IDCANCEL)

DestroyWindow(hWnd);

break;

**Диалоговые окна**

1. Изучим теоретический материал по работе с графическими ресурсами приложений.
2. Использование диалогового окна в составе главного.
   1. Изучим теоретический материал и разработаем диаграмму состояний.



Д1: выделение ресурсов, создание главного окна, создание диалогового окна.

Д2: удаление диалогового окна, удаление главного окна.

* 1. Напишем программу

// maindlg.cpp : Defines the entry point for the application.

//

#include "stdafx.h"

LRESULT CALLBACK WndProc(HWND hWnd,

UINT msg,

WPARAM wParam,

LPARAM lParam);

LRESULT CALLBACK MainDlgProc(HWND hDlg,

UINT msg,

WPARAM wParam,

LPARAM lParam);

HINSTANCE hInst;

const int STR\_SIZE = 100;

char szWndClass[STR\_SIZE] = "WndClass";

char szWndTitle[STR\_SIZE] = "Title";

int APIENTRY WinMain(HINSTANCE hInstance,

HINSTANCE hPrevInstance,

LPSTR lpCmdLine,

int nCmdShow)

{

hInst = hInstance;

WNDCLASS wc;

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hbrBackground = (HBRUSH) GetStockObject(WHITE\_BRUSH);

wc.hCursor = LoadCursor(NULL, IDC\_ARROW);

wc.hIcon = 0;

wc.hInstance = hInstance;

wc.lpfnWndProc = WndProc;

wc.lpszClassName = szWndClass;

wc.lpszMenuName = NULL;

wc.style = CS\_HREDRAW | CS\_VREDRAW;

if(!RegisterClass(&wc))

return -1;

HWND hWnd = CreateWindow(szWndClass, szWndTitle,

WS\_OVERLAPPEDWINDOW,

0, 0, 600, 400,

NULL, NULL, hInstance, NULL);

if(!hWnd)

return -2;

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

MSG msg;

while(GetMessage(&msg, NULL, NULL, NULL))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

LRESULT CALLBACK WndProc(HWND hWnd,

UINT msg,

WPARAM wParam,

LPARAM lParam)

{

HDC hdc;

PAINTSTRUCT ps;

static bool fIsOpened = false;

switch(msg)

{

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

if(!fIsOpened)

{

DialogBox(NULL, (LPCTSTR)IDD\_MAINDLG, hWnd, (DLGPROC)MainDlgProc);

fIsOpened = true;

}

EndPaint(hWnd, &ps);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, msg, wParam, lParam);

}

return 0;

}

LRESULT CALLBACK MainDlgProc(HWND hDlg,

UINT msg,

WPARAM wParam,

LPARAM lParam)

{

switch(msg)

{

case WM\_INITDIALOG:

return TRUE;

case WM\_COMMAND:

switch(LOWORD(wParam))

{

case IDOK:

EndDialog(hDlg, FALSE);

break;

case IDCANCEL:

EndDialog(hDlg, FALSE);

break;

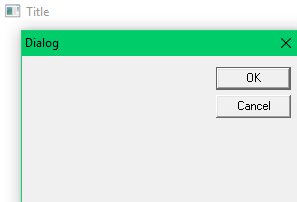
}

}

return 0;

}

Программа выполняется и закрывается при нажатии на любую кнопку диалогового окна.



* 1. Модифицируем приложение, добавив вывод подтверждения нажатия на кнопки окна.

LRESULT CALLBACK MainDlgProc( HWND hDlg,

UINT msg,

WPARAM wParam,

LPARAM lParam)

{

switch(msg)

{

case WM\_INITDIALOG:

return TRUE;

case WM\_COMMAND:

switch(LOWORD(wParam))

{

case IDOK:

if(MessageBox(hDlg, "Do you really want to close the application?",

"Confirmation", MB\_OKCANCEL) == IDOK)

EndDialog(hDlg, FALSE);

break;

case IDCANCEL:

if(MessageBox(hDlg, "Do you really want to close the application?",

"Confirmation", MB\_OKCANCEL) == IDOK)

EndDialog(hDlg, FALSE);

break;

default:

break;

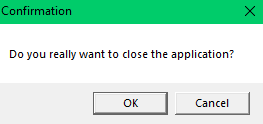
}

}

return 0;

}

Результат выполнения программы:



* 1. Чтобы разрушать окно только при нажатии CANCEL уберем обработчик IDOK.
  2. Сделаем так, чтобы вместе с диалоговым окном закрывалось и главное окно.

LRESULT CALLBACK MainDlgProc( HWND hDlg,

UINT msg,

WPARAM wParam,

LPARAM lParam)

{

switch(msg)

{

case WM\_INITDIALOG:

return TRUE;

case WM\_COMMAND:

switch(LOWORD(wParam))

{

case IDCANCEL:

if(MessageBox(hDlg, "Do you really want to close the application?",

"Confirmation", MB\_OKCANCEL) == IDOK)

PostQuitMessage(0);

break;

default:

break;

}

}

return 0;

}

**Элемент управления окно редактирования.**

1. Изучим теоретический материал по использованию элементов управления.
2. Создадим приложение с диалоговым окном в качестве главного. В данном окне будет 2 поля для ввода: одно будет активно, другое – нет. Если ввести текст в одно окно, а затем нажать ОК, то тест будет приведен к верхнему регистру и выведен в другое поле.

#include "stdafx.h"

#include <locale>

LRESULT CALLBACK MainDlgProc(HWND hDlg,

UINT msg,

WPARAM wParam,

LPARAM lParam);

const int STR\_SIZE = 100;

int APIENTRY WinMain(HINSTANCE hInstance,

HINSTANCE hPrevInstance,

LPSTR lpCmdLine,

int nCmdShow)

{

DialogBox(NULL, (LPCTSTR)IDD\_MAINDLG, NULL, (DLGPROC)MainDlgProc);

MSG msg;

while(GetMessage(&msg, NULL, NULL, NULL))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

LRESULT CALLBACK MainDlgProc(HWND hDlg,

UINT msg,

WPARAM wParam,

LPARAM lParam)

{

std::locale loc;

int i;

char szText[STR\_SIZE] = "";

switch(msg)

{

case WM\_INITDIALOG:

return TRUE;

case WM\_COMMAND:

switch(LOWORD(wParam))

{

case IDCANCEL:

if(MessageBox(hDlg, "Do you really want to close the application?",

"Confirmation", MB\_OKCANCEL) == IDOK)

PostQuitMessage(0);

break;

case IDOK:

GetDlgItemText(hDlg, IDC\_INPUT, szText, STR\_SIZE);

for(i = 0; szText[i]!='\0'; i++)

szText[i] = std::toupper(szText[i], loc);

SetDlgItemText(hDlg, IDC\_OUTPUT, szText);

break;

default:

break;

}

}

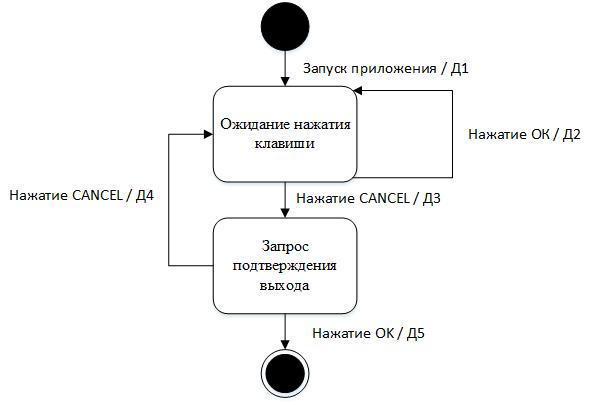
return 0;

}

Результат выполнения:



Диаграмма состояний:



Д1: выделение ресурсов, выведение диалогового окна

Д2: считывание содержимого поля для ввода, перевод содержимого в верхний регистр, вывод содержимого в поле для вывода

Д3: выведение окна сообщения с запросом на подтверждения

Д4: возврат к диалоговому окну

Д5: закрытие окна, высвобождение ресурсов

Вывод: По ходу лабораторной работы познакомились с каркасом Hello, изучили работу диалоговых окно и элемента управления окно редактирования.