Белорусский государственный технологический университет

Кафедра Информационных Систем и Технологий

**Курс «Операционные системы и системное программирование»**

**Лабораторная работа 4. Создание оконных приложений**

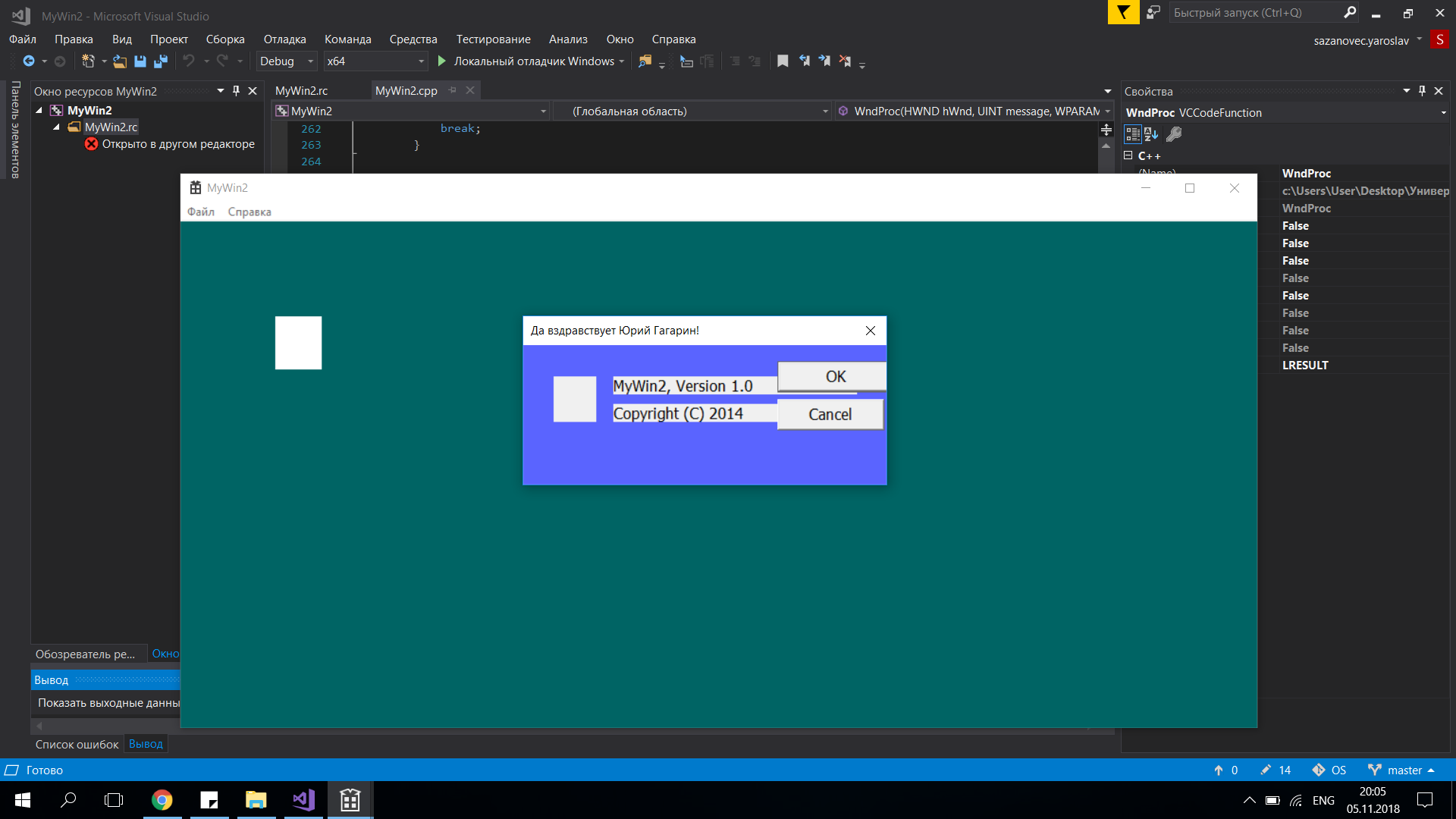
Выполнил: Сазановец Я.И.

ФИТ 3 курс 4 группа

Минск 2018

Цель: Научиться создавать и конфигурировать оконные приложения win32

**ЗАДАНИЕ.** Ознакомиться с материалом. Выполнить задания.



Код программы

#include "stdafx.h"

#include <string>

#include <stdio.h>

#include "Resource.h"

#define IDC\_MYICON 2

#define IDD\_WINCW\_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\_WINCW 107

#define IDI\_SMALL 108

#define IDC\_WINCW 109

#define IDC\_WMCHAR 110

#define IDR\_MAINFRAME 128

#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

#define MAX\_LOADSTRING 100

// Global Variables:

HINSTANCE hInst; // current instance

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

TCHAR szWindowClass[MAX\_LOADSTRING]; // the main window class name

int dx = 8;

int dy = 8;

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

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

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

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

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

int row = 1, column = 1;

int APIENTRY \_tWinMain(HINSTANCE hInstance,

HINSTANCE hPrevInstance,

LPTSTR lpCmdLine,

int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: Place code here.

MSG msg;

HACCEL hAccelTable;

// Initialize global strings

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

LoadString(hInstance, IDC\_MYWIN2, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Perform application initialization:

if (!InitInstance(hInstance, nCmdShow))

{

return FALSE;

}

hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_MYWIN2));

// Main message loop:

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

{

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int)msg.wParam;

}

//

// FUNCTION: MyRegisterClass()

//

// PURPOSE: Registers the window class.

//

// COMMENTS:

//

// This function and its usage are 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;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_MYWIN2));

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

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

wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_MYWIN2);

wcex.lpszClassName = szWindowClass;

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

return RegisterClassEx(&wcex);

}

//

// FUNCTION: InitInstance(HINSTANCE, 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;

}

HBRUSH brush = CreateSolidBrush(RGB(0, 100, 100));

SetClassLongPtr(hWnd, GCLP\_HBRBACKGROUND, (LONG)brush);

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

//

// FUNCTION: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// 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);

hdc = GetDC(hWnd);

TCHAR str[] = TEXT("BIG MAG");

TCHAR str1[] = TEXT(" ");

SetTextColor(hdc, RGB(255, 255, 255));

TextOut(hdc, 100, 100, str1, 9);

TextOut(hdc, 100, 110, str1, 9);

TextOut(hdc, 100, 120, str1, 9);

TextOut(hdc, 100, 130, str1, 9);

TextOut(hdc, 100, 140, str1, 9);

int responce = 0;

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\_RBUTTONDOWN:

DialogBox(hInst, (LPCTSTR)IDD\_DIALOG1, hWnd, (DLGPROC)myF);

break;

case WM\_KEYDOWN:

hdc = GetDC(hWnd);

InvalidateRect(hWnd, 0, TRUE);

UpdateWindow(hWnd);

SetBkColor(hdc, RGB(255, 0, 0));

SetTextColor(hdc, RGB(255, 255, 255));

switch (wParam)

{

case VK\_DOWN:

TextOut(hdc, column, row += dy, str, 7);

break;

case VK\_UP:

TextOut(hdc, column, row -= dy, str, 7);

break;

case VK\_RIGHT:

TextOut(hdc, column += dx, row, str, 7);

break;

case VK\_LEFT:

TextOut(hdc, column -= dx, row, str, 7);

break;

default:

TextOut(hdc, column, row, str, sizeof(str) / sizeof(TCHAR));

break;

}

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

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

EndPaint(hWnd, &ps);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

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

{

static HBRUSH testBrush;

static HDC dcOne = NULL;

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

testBrush = CreateSolidBrush(RGB(0, 0, 255));

return (INT\_PTR)TRUE;

case WM\_CTLCOLORDLG:

return (INT\_PTR)(testBrush);

case WM\_CTLCOLORBTN:

return (INT\_PTR)testBrush;

case WM\_COMMAND:

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

{

int z = MessageBoxEx(NULL, TEXT("HELLO"), TEXT("DialogWin"), MB\_OK, 0x0459);

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

// Message handler for about box.

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

static HBRUSH testBrush;

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

testBrush = CreateSolidBrush(RGB(90, 100, 255));

return (INT\_PTR)TRUE;

case WM\_CTLCOLORDLG:

return (INT\_PTR)testBrush;

case WM\_CTLCOLORBTN:

return (INT\_PTR)testBrush;

case WM\_COMMAND:

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

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}