Филиал «РКТ» МАИ  
Федеральное государственное бюджетное образовательное учреждение высшего образования  
**«Московский авиационный институт  
(национальный исследовательский университет)»**

Отчет по лабораторным работам

Вариант 14  
  
Системное программирование  
  
  
МП-31

ЛИСТОВ 27

Проверил Костромина Н.Е.

Разработал Коробков И.Д.

2020

Оглавление

[Часть 1 3](#_Toc57314913)

[Задание А 3](#_Toc57314914)

[Задание B 5](#_Toc57314915)

[Задание C 8](#_Toc57314916)

[Задание D 14](#_Toc57314917)

[Часть 2 17](#_Toc57314918)

[Часть 3 22](#_Toc57314919)

[Индивидуальное задание: 25 окон с крестиками 25](#_Toc57314920)

# Часть 1

## Задание А

Задание: Составьте программу, в которой в главное окно серого цвета выводятся три вложенных друг в друга квадрата с размерами сторон 300, 200 и 100 пикселов, соприкасающиеся левыми верхними углами. Внешний квадрат закрасьте фиолетовым цветом, средний – темно-синим, а внутренний – светло-синим.

В этой программе я не использовал собственные функции.

Листинг программы:

Lab1.cpp

#include<Windows.h>

#include<tchar.h>

#include <windowsx.h>

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

int cx, cy;

HBRUSH hPurpleBrush, hBlueBrush, hDBlueBrush;

//==================================================================================

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {

HWND hMainWnd;

TCHAR szClassName[] = \_T("MyClass");

MSG msg;

WNDCLASSEX wc;

//Заполняем структуру класса окна

wc.cbSize = sizeof(wc);

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

wc.lpfnWndProc = WndProc;

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hInstance = hInstance;

wc.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

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

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

wc.lpszMenuName = NULL;

wc.lpszClassName = szClassName;

wc.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

//Регистрируем класс окна

if (!RegisterClassExW(&wc)) {

MessageBoxW(NULL, \_T("Cannot registr class"), \_T("Error"), MB\_OK);

return 0;

}

//Создаем основное окно приложения

hMainWnd = CreateWindow(szClassName, \_T("A Hello1 Application"), WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, NULL, NULL,

hInstance, NULL);

if (!hMainWnd) {

MessageBox(NULL, \_T("Cannot create main window"), \_T("Error"), MB\_OK);

return 0;

}

//Показываем окно

ShowWindow(hMainWnd, nCmdShow);

//UpdateWindow(hMainWnd);

//Выполняем цикл обработки сообщений до закрытия окна

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

TranslateMessage(&msg);

DispatchMessageW(&msg);

}

return 0;

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT uMsg, WPARAM wParam,

LPARAM lParam) {

static POINT apt[4];

HDC hdc;

static int t;

PAINTSTRUCT ps;

switch (uMsg) {

case WM\_CREATE:

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

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

hDBlueBrush = CreateSolidBrush(RGB(0, 33, 55));

break;

case WM\_PAINT:

hdc = BeginPaint(hwnd, &ps);

SelectBrush(hdc, hPurpleBrush);

Rectangle(hdc, 0, 0, 300, 300);

SelectBrush(hdc, hBlueBrush);

Rectangle(hdc, 0, 0, 200, 200);

SelectBrush(hdc, hDBlueBrush);

Rectangle(hdc, 0, 0, 100, 100);

EndPaint(hwnd, &ps);

break;

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

}

return DefWindowProc(hwnd, uMsg, wParam, lParam);

}

Результат:



## Задание B

Задание: Составьте программу с меню **"Отражение"**, состоящим из пунктов с названиями **"Слева"** и **"Справа"**. Выбор каждого пункта меню должен приводить к выводу в главное окно серого цвета либо левой, либо правой половины синего круга. Полукруги рисуются с помощью функций **Pie()** или **Chord()**.

В этой программе я не использовал собственные функции.

Листинг программы:

Lab2.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<Windows.h>

#include<tchar.h>

#include<cmath>

#include "Lab2.h"

#include <windowsx.h>

HBRUSH hBlueBrush, hGreenBrush, hRedBrush, hYellowBrush;

HPEN hBluePen, hRedPen;

int sw;

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

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {

HWND hMainWnd;

TCHAR szClassName[] = \_T("menudemo");

MSG msg;

WNDCLASSEX wc;

ZeroMemory(&wc, sizeof(wc));

wc.cbSize = sizeof(wc);

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

wc.lpfnWndProc = WndProc;

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hInstance = hInstance;

wc.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

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

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

wc.lpszMenuName = \_T("menudemo");

wc.lpszClassName = szClassName;

wc.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

if (!RegisterClassExW(&wc)) {

MessageBoxW(NULL, \_T("Cannot registr class"), \_T("Error"), MB\_OK);

return 0;

}

hMainWnd = CreateWindow(szClassName, \_T("menudemo"), WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, NULL, NULL,

hInstance, NULL);

if (!hMainWnd) {

MessageBox(NULL, \_T("Cannot create main window"), \_T("Error"), MB\_OK);

return 0;

}

ShowWindow(hMainWnd, nCmdShow);

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

TranslateMessage(&msg);

DispatchMessageW(&msg);

}

return 0;

}

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

HDC hdc;

PAINTSTRUCT ps;

HMENU hMenu, hPopupMenu;

TCHAR str0[] = \_T("Отражение");

TCHAR str1[] = \_T("Справа");

TCHAR str2[] = \_T("Слева");

switch (uMsg) {

case WM\_COMMAND:

hMenu = GetMenu(hWnd);

switch (LOWORD(wParam))

{

case MI\_RIGHT:

sw = 1;

InvalidateRect(hWnd, NULL, TRUE);

break;

case MI\_LEFT:

sw = 2;

InvalidateRect(hWnd, NULL, TRUE);

break;

case MI\_EXIT:

DestroyWindow(hWnd);

}

break;

case WM\_CREATE:

hBlueBrush = CreateSolidBrush(RGB(200, 200, 200));

hGreenBrush = CreateSolidBrush(RGB(200, 200, 200));

hMenu = CreateMenu();

hPopupMenu = CreatePopupMenu();

AppendMenu(hPopupMenu, MF\_STRING, MI\_RIGHT, str1);

AppendMenu(hPopupMenu, MF\_STRING, MI\_LEFT, str2);

AppendMenu(hMenu, MF\_POPUP, (UINT)hPopupMenu, str0);

SetMenu(hWnd, hMenu);

break;

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

Ellipse(hdc, 250, 250, 450, 450);

if (sw == 2)

{

SelectBrush(hdc, hGreenBrush);

Pie(hdc, 250, 250, 450, 450, 350, 250, 350, 450);

}

if (sw == 1)

{

SelectBrush(hdc, hBlueBrush);

Pie(hdc, 250, 250, 450, 450, 350, 450, 350, 250);

}

EndPaint(hWnd, &ps);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

Lab2.h

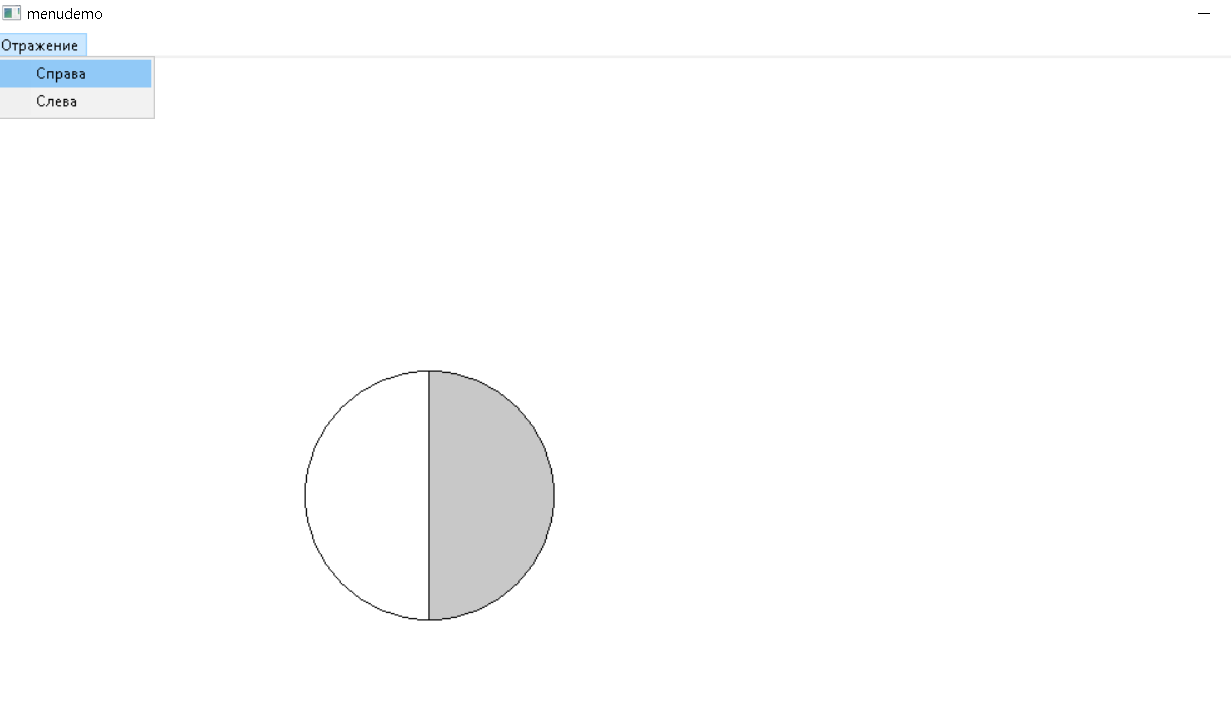
#define MI\_EXIT 102

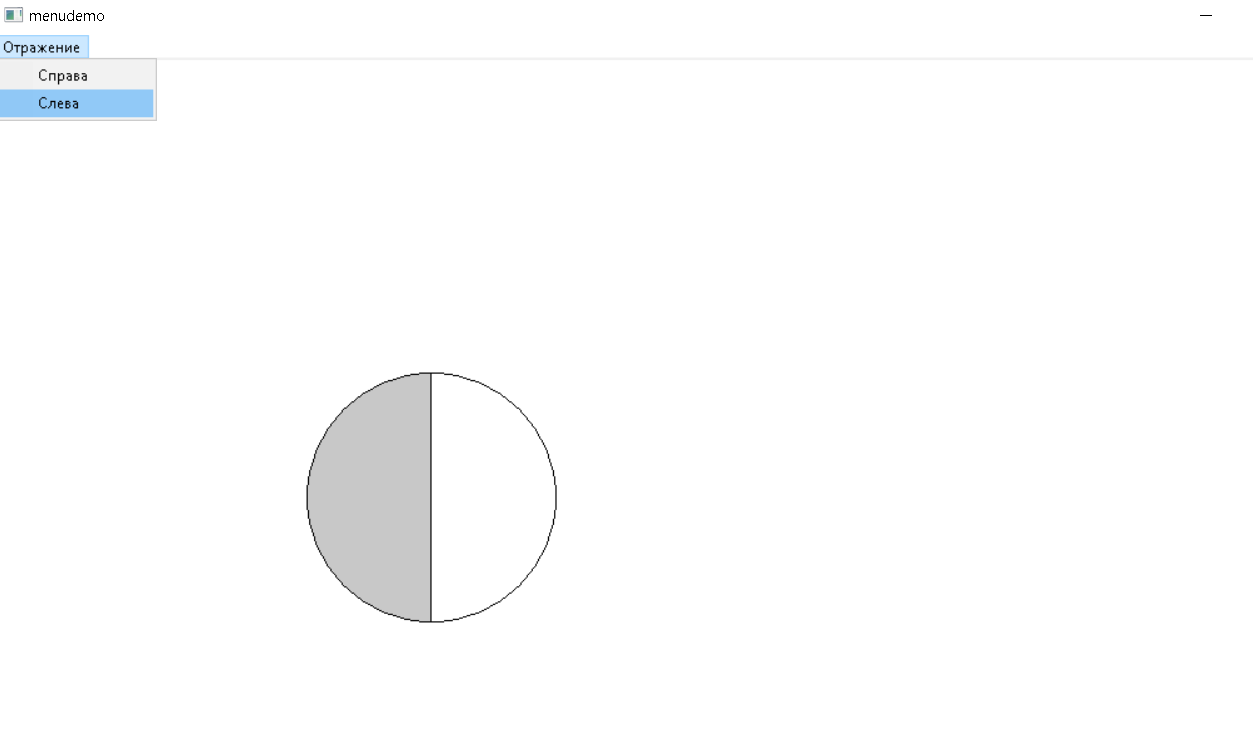
#define MI\_RIGHT 106

#define MI\_LEFT 107

Проблемы возникли с подбором чисел для последних четырёх параметров функции Pie(). Подбирать координаты для этих точек надо относительно окна.

Результат:





## Задание C

Задание Включите в состав диалога кнопку **"Закрыть"**, комбинированный список с линейкой вертикальной прокрутки, а также статический элемент, куда функция **DlgDirList­Combo­Box()** поместит имя выбранного файла. Введите в оконную процедуру диалога обработку сообщений **WM\_CTLCOLORDLG** и **WM\_­CTL­­COLORSTATIC**. Покрасьте окно диалога и статический элемент в бледно-фиолетовый цвет, а текст в статическом элементе – в темно-фиолетовый. Заполните комбинированный список упорядоченным содержимым текущего каталога диска, воспользовавшись для этого функцией **DlgDirListComboBox()**, которую следует вызвать при обработке сообщения **WM\_INITDIALOG**. Выведите в комбинированный список только файлы .EXE.

В этой программе я не использовал собственных функций.

Листинг программы:

Lab6.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<Windows.h>

#include<tchar.h>

#include "Resource.h"

#define ID\_STDDIALOG 11

//#define IDC\_HSCR -1

//#define IDC\_VSCR -2

INT\_PTR CALLBACK Dialog1(HWND, UINT, WPARAM, LPARAM);

//LPCWSTR combo[] = { \_T("aa"),\_T("bb"),\_T("cc") };

TCHAR WinName[] = \_T("MainFrame");

HINSTANCE hInst;

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

int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {

HWND hwnd;

MSG msg;

WNDCLASSEX wc;

hInst = hInstance;

wc.cbSize = sizeof(wc);

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

wc.lpfnWndProc = WndProc;

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hInstance = hInstance;

wc.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

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

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

wc.lpszMenuName = NULL;

wc.lpszClassName = WinName;

wc.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

//Регистрируем класс окна

if (!RegisterClassEx(&wc)) return 0;

hwnd = CreateWindow(WinName, \_T("Dialog"),

WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, CW\_USEDEFAULT,

CW\_USEDEFAULT, CW\_USEDEFAULT,

HWND\_DESKTOP, NULL, hInstance, NULL);

//Показываем окно

ShowWindow(hwnd, nCmdShow);

//UpdateWindow(hMainWnd);

//Выполняем цикл обработки сообщений до закрытия окна

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

TranslateMessage(&msg);

DispatchMessageW(&msg);

}

return 0;

}

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

PAINTSTRUCT ps;

HDC hdc;

TCHAR str[256];

int i;

HMENU hMenu, hPopupMenu;

TCHAR str0[] = \_T("Диалог");

TCHAR str1[] = \_T("Открыть");

switch (uMsg) {

case WM\_COMMAND:

switch (LOWORD(wParam)) {

case ID\_STDDIALOG:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, Dialog1);

return 0;

default: return DefWindowProc(hWnd, uMsg, wParam, lParam);

}

break;

case WM\_CREATE:

hMenu = CreateMenu();

hPopupMenu = CreatePopupMenu();

AppendMenu(hPopupMenu, MF\_STRING, ID\_STDDIALOG, str1);

AppendMenu(hMenu, MF\_POPUP, (UINT)hPopupMenu, str0);

SetMenu(hWnd, hMenu);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

}

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

}

INT\_PTR CALLBACK Dialog1(HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM lParam)

{

HBRUSH hBrushBk = CreateSolidBrush(RGB(149, 123, 141));

HBRUSH hBrushStatic = CreateSolidBrush(RGB(149, 123, 141));

static HWND hWndComboBox;

TCHAR arr[] = \_T("\*.exe");

HDC hdc;

switch (uMsg) {

case WM\_INITDIALOG:

DlgDirListComboBox(hDlg, arr, IDC\_COMBO1, IDC\_STATIC,

DDL\_READWRITE | DDL\_READONLY | DDL\_HIDDEN |

DDL\_SYSTEM | DDL\_DIRECTORY | DDL\_DRIVES |

DDL\_ARCHIVE);

return TRUE;

return TRUE;

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case IDC\_BUTTON1: return EndDialog(hDlg, 0);

case IDCANCEL: return EndDialog(hDlg, 0);

}

return TRUE;

case WM\_CTLCOLORDLG: return (LONG)hBrushBk;

case WM\_CTLCOLORSTATIC:

hdc = (HDC)wParam;

SetTextColor(hdc, RGB(148, 0, 211));

return (LONG)hBrushStatic;

case WM\_ERASEBKGND:

break;

default: return FALSE;

}

return FALSE;

}

Resource.h

//{{NO\_DEPENDENCIES}}

// Включаемый файл, созданный в Microsoft Visual C++.

// Используется Lab6.rc

//

#define IDC\_MYICON 2

#define IDD\_LAB6\_DIALOG 102

#define IDS\_APP\_TITLE 103

#define IDD\_ABOUTBOX 103

#define IDM\_ABOUT 104

#define IDM\_EXIT 105

#define IDI\_LAB6 107

#define IDI\_SMALL 108

#define IDC\_LAB6 109

#define IDD\_DIALOG1 110

#define IDR\_MAINFRAME 128

#define IDC\_COMBO1 1005

#define IDC\_SCROLLBAR2 1006

#define IDC\_BUTTON1 1007

#define IDC\_STATIC -1

// Next default values for new objects

//

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NO\_MFC 1

#define \_APS\_NEXT\_RESOURCE\_VALUE 129

#define \_APS\_NEXT\_COMMAND\_VALUE 32771

#define \_APS\_NEXT\_CONTROL\_VALUE 1008

#define \_APS\_NEXT\_SYMED\_VALUE 110

#endif

#endif

Lab6.rc

// Microsoft Visual C++ generated resource script.

//

#include "resource.h"

#define APSTUDIO\_READONLY\_SYMBOLS

/////////////////////////////////////////////////////////////////////////////

//

// Generated from the TEXTINCLUDE 2 resource.

//

#ifndef APSTUDIO\_INVOKED

#include "targetver.h"

#endif

#define APSTUDIO\_HIDDEN\_SYMBOLS

#include "windows.h"

#undef APSTUDIO\_HIDDEN\_SYMBOLS

/////////////////////////////////////////////////////////////////////////////

#undef APSTUDIO\_READONLY\_SYMBOLS

/////////////////////////////////////////////////////////////////////////////

// Русский (Россия) resources

#if !defined(AFX\_RESOURCE\_DLL) || defined(AFX\_TARG\_RUS)

LANGUAGE LANG\_RUSSIAN, SUBLANG\_DEFAULT

/////////////////////////////////////////////////////////////////////////////

//

// Icon

//

// Icon with lowest ID value placed first to ensure application icon

// remains consistent on all systems.

IDI\_LAB6 ICON "Lab6.ico"

IDI\_SMALL ICON "small.ico"

/////////////////////////////////////////////////////////////////////////////

//

// Menu

//

IDC\_LAB6 MENU

BEGIN

POPUP "&Файл"

BEGIN

MENUITEM "Вы&ход", IDM\_EXIT

END

POPUP "&Справка"

BEGIN

MENUITEM "&О программе...", IDM\_ABOUT

END

END

/////////////////////////////////////////////////////////////////////////////

//

// Accelerator

//

IDC\_LAB6 ACCELERATORS

BEGIN

"?", IDM\_ABOUT, ASCII, ALT

"/", IDM\_ABOUT, ASCII, ALT

END

/////////////////////////////////////////////////////////////////////////////

//

// Dialog

//

IDD\_ABOUTBOX DIALOGEX 0, 0, 234, 153

STYLE DS\_SETFONT | DS\_MODALFRAME | DS\_FIXEDSYS | WS\_POPUP | WS\_CAPTION | WS\_SYSMENU

CAPTION "О программе Lab6"

FONT 8, "MS Shell Dlg", 0, 0, 0x1

BEGIN

COMBOBOX IDC\_COMBO1,23,66,75,30,CBS\_DROPDOWN | CBS\_SORT | WS\_VSCROLL | WS\_TABSTOP

LTEXT "Статический",IDC\_STATIC,34,38,48,17

PUSHBUTTON "Закрыть",IDC\_BUTTON1,144,106,50,14

END

/////////////////////////////////////////////////////////////////////////////

//

// DESIGNINFO

//

#ifdef APSTUDIO\_INVOKED

GUIDELINES DESIGNINFO

BEGIN

IDD\_ABOUTBOX, DIALOG

BEGIN

LEFTMARGIN, 7

RIGHTMARGIN, 227

TOPMARGIN, 7

BOTTOMMARGIN, 146

END

END

#endif // APSTUDIO\_INVOKED

#ifdef APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// TEXTINCLUDE

//

1 TEXTINCLUDE

BEGIN

"resource.h\0"

END

2 TEXTINCLUDE

BEGIN

"#ifndef APSTUDIO\_INVOKED\r\n"

"#include ""targetver.h""\r\n"

"#endif\r\n"

"#define APSTUDIO\_HIDDEN\_SYMBOLS\r\n"

"#include ""windows.h""\r\n"

"#undef APSTUDIO\_HIDDEN\_SYMBOLS\r\n"

"\0"

END

3 TEXTINCLUDE

BEGIN

"\r\n"

"\0"

END

#endif // APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// String Table

//

STRINGTABLE

BEGIN

IDS\_APP\_TITLE "Lab6"

IDC\_LAB6 "LAB6"

END

#endif // Русский (Россия) resources

/////////////////////////////////////////////////////////////////////////////

#ifndef APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// Generated from the TEXTINCLUDE 3 resource.

//

/////////////////////////////////////////////////////////////////////////////

#endif // not APSTUDIO\_INVOKED

Картины в программе не используются.

Результаты:



Для того чтобы вывести файлы определенного расширения вторым параметром должен является указатель на строку шаблона имен файлов. Мы отображаем имена .exe файлов, поэтому в качестве шаблона используется строка "\*.exe".

## Задание D

Задание: Выведите график функции *f* = e–*x* cos (5*x)*, вычисляя координаты 200 точек для положительных значений *x* и 200 – для отрицательных. Для переменной *x* задайте шаг 1/100; ординаты графика умножайте на 50.

Я использовал свою функцию int function (double x), чтобы получить значение y при определенном значении x.

Листинг программы:

Lab5.cpp

#define \_USE\_MATH\_DEFINE

#include<Windows.h>

#include<tchar.h>

#include <cmath>

const int WIDTH = 100;

const int HEIGHT = 300;

int function(double x) {

return exp(-x)\*cos(5\*x);

}

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

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {

HWND hMainWnd;

TCHAR szClassName[] = \_T("MyClass");

MSG msg;

WNDCLASSEX wc;

//Заполняем структуру класса окна

wc.cbSize = sizeof(wc);

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

wc.lpfnWndProc = WndProc;

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hInstance = hInstance;

wc.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

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

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

wc.lpszMenuName = NULL;

wc.lpszClassName = szClassName;

wc.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

//Регистрируем класс окна

if (!RegisterClassExW(&wc)) {

MessageBoxW(NULL, \_T("Cannot registr class"), \_T("Error"), MB\_OK);

return 0;

}

//Создаем основное окно приложения

hMainWnd = CreateWindow(szClassName, \_T("A Hello1 Application"), WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, NULL, NULL,

hInstance, NULL);

if (!hMainWnd) {

MessageBox(NULL, \_T("Cannot create main window"), \_T("Error"), MB\_OK);

return 0;

}

//Показываем окно

ShowWindow(hMainWnd, nCmdShow);

//UpdateWindow(hMainWnd);

//Выполняем цикл обработки сообщений до закрытия окна

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

TranslateMessage(&msg);

DispatchMessageW(&msg);

}

return msg.wParam;

char path[100];

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT uMsg, WPARAM wParam,

LPARAM lParam) {

HDC hDC;

PAINTSTRUCT ps;

RECT rect;

static HPEN hpen1, hpen2;

static int sx, sy;

double x;

int y;

switch (uMsg) {

case WM\_CREATE:

hpen1 = CreatePen(PS\_SOLID, 2, RGB(0, 0, 255));

hpen2 = CreatePen(PS\_SOLID, 0.5, RGB(255, 0, 255));

break;

case WM\_SIZE:

sx = LOWORD(lParam);

sy = HIWORD(lParam);

//MoveWindow(hWnd, 0, 0, 500, 500, TRUE);

break;

case WM\_PAINT:

hDC = BeginPaint(hWnd, &ps);

SetMapMode(hDC, MM\_ANISOTROPIC);

SetWindowExtEx(hDC, 2 \* WIDTH, 2 \* HEIGHT, NULL);

SetViewportExtEx(hDC, sx / 2, sy / 2, NULL);

SetViewportOrgEx(hDC, sx / 2, sy / 2, NULL);

SelectObject(hDC, hpen1);

MoveToEx(hDC, -WIDTH, 0, NULL);

LineTo(hDC, WIDTH, 0);

MoveToEx(hDC, 0, HEIGHT, NULL);

LineTo(hDC, 0, -HEIGHT);

SelectObject(hDC, hpen2);

MoveToEx(hDC, -WIDTH, 0, NULL);

for (x = -200; x <= 200; x += 1.0f / 100)

{

y = function(x);

LineTo(hDC, x, y \* 50);

}

EndPaint(hWnd, &ps);

break;

case WM\_DESTROY:

DeleteObject(hpen1);

DeleteObject(hpen2);

PostQuitMessage(0);

break;

default:

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

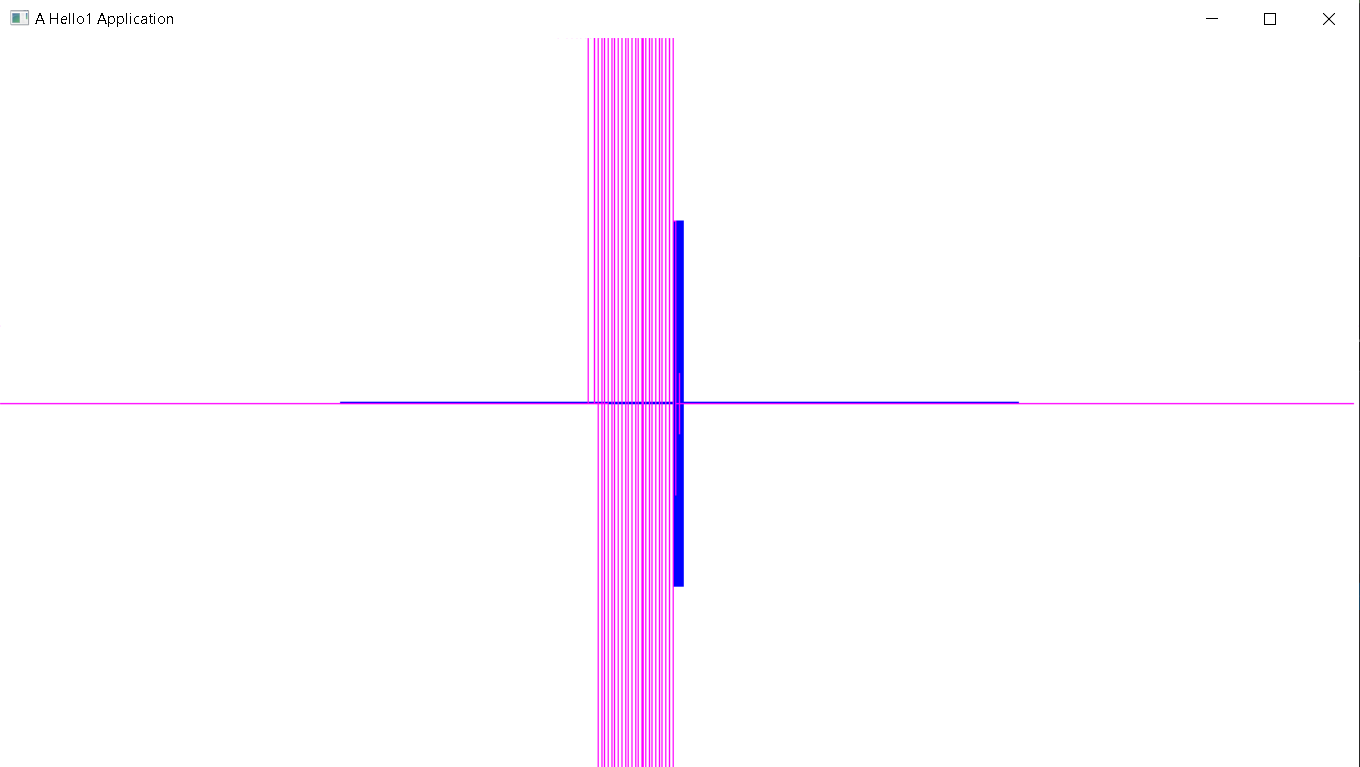
}

return 0;

}

Картины не использовал для этого приложения.

Результат:



Для наилучшего отображения нужно изменять значения переменных WIDTH и

HEIGHT.

# Часть 2

Задание: Анимация летящей бабочки.

Для анимации использовал свой класс ObjAnim, которая упрощает добавление и анимацию картинок. Имеет 2 метода:

* Create (LPCWSTR name, HDC hdc) – добавляет картинку в класс
* Draw (int x, int y, HDC hdc) – отображает картинку в окне

Листинг программы:

Anim.h

#include<Windows.h>

#include<tchar.h>

class ObjAnim {

public:

HBITMAP hBitmap;

HDC memBit;

BITMAP bm;

ObjAnim() {}

ObjAnim(LPCWSTR name, HDC hdc)

{

this->hBitmap = (HBITMAP)LoadImage(NULL, name, IMAGE\_BITMAP,

0, 0, LR\_LOADFROMFILE | LR\_CREATEDIBSECTION);

GetObject(hBitmap, sizeof(bm), &bm);

memBit = CreateCompatibleDC(hdc);

SelectObject(memBit, hBitmap);

}

void Create(LPCWSTR name, HDC hdc) {

this->hBitmap = (HBITMAP)LoadImage(NULL, name, IMAGE\_BITMAP,

0, 0, LR\_LOADFROMFILE | LR\_CREATEDIBSECTION);

GetObject(hBitmap, sizeof(bm), &bm);

memBit = CreateCompatibleDC(hdc);

SelectObject(memBit, hBitmap);

}

void Draw(int x, int y, HDC hdc) {

BitBlt(hdc, 0 + x, 0 + y, bm.bmWidth, bm.bmHeight, memBit, 0, 0, SRCCOPY);

}

};

Lab3.cpp

#include<Windows.h>

#include<tchar.h>

#include "resource.h"

#include "Anim.h"

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

TCHAR WinName[] = \_T("MainFrame");

HINSTANCE hInst;

int APIENTRY \_tWinMain(HINSTANCE This, //Дескриптор текущего приложения

HINSTANCE Prev, //В современных системах всегда 0

LPTSTR cmd, //Командная строка

int mode) //Режим отображения окна

{

HWND hWnd; //Дескриптор главного окна программы

MSG msg; //Структура для хранения сообщения

hInst = This;

WNDCLASS wc;//Класс окна

//Определение класса окна

wc.hInstance = This;

wc.lpszClassName = WinName; //Имя класса окна

wc.lpfnWndProc = WndProc; //Имя класса окна

wc.style = CS\_HREDRAW | CS\_VREDRAW; //Стиль окна

wc.hIcon = LoadIcon(NULL, IDI\_APPLICATION); //Стандартная иконка

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

wc.lpszMenuName = NULL; //Нет меню

wc.cbClsExtra = 0; //Нет дополнительных данных класса

wc.cbWndExtra = 0; //Нет дополнительных данных окна

//Заполнение окна белым цветом

wc.hbrBackground = (HBRUSH)CreateSolidBrush(RGB(255, 255, 255));

if (!RegisterClass(&wc)) return 0;//Регистрация класса окна

//Создание окна

hWnd = CreateWindow(WinName,//Имя класса окна

\_T("Каркас Windows-приложения"),//Заголовок окна

WS\_OVERLAPPEDWINDOW, //Стиль окна

CW\_USEDEFAULT,//x

CW\_USEDEFAULT,//y Размеры окна

CW\_USEDEFAULT,//Width

CW\_USEDEFAULT,//Height

HWND\_DESKTOP,//Дескриптор родительской информации нет

NULL,//Нет меню

This,//Дескриптор приложения

NULL);//Дополнительной информации нет

ShowWindow(hWnd, mode);//Показать окно

//Цикл обработки сообщений

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

{

TranslateMessage(&msg);//Функция трансляции кодов нажатой клавиши

DispatchMessage(&msg);//Посылает сообщение функции WndProc()

}

return 0;

}

//Оконная функция вызывается операционной системой

//и получает сообщения из очереди для данного приложения

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

{

PAINTSTRUCT ps;

HDC hdc;

static ObjAnim arr[31];

//static ObjAnim ship;

//static ObjAnim Sea;

static int caption, menu, border;

static int t = 0;

switch (message)

{

case WM\_CREATE:

SetTimer(hWnd, 10, 100, NULL);

caption = GetSystemMetrics(SM\_CYCAPTION);

menu = GetSystemMetrics(SM\_CYMENU);

border = GetSystemMetrics(SM\_CXFIXEDFRAME);

hdc = GetDC(hWnd);

arr[0] = ObjAnim(\_T("1.bmp"), hdc);

arr[1] = ObjAnim(\_T("2.bmp"), hdc);

arr[2] = ObjAnim(\_T("3.bmp"), hdc);

arr[3] = ObjAnim(\_T("4.bmp"), hdc);

arr[4] = ObjAnim(\_T("5.bmp"), hdc);

arr[5] = ObjAnim(\_T("6.bmp"), hdc);

arr[6] = ObjAnim(\_T("7.bmp"), hdc);

arr[7] = ObjAnim(\_T("8.bmp"), hdc);

arr[8] = ObjAnim(\_T("9.bmp"), hdc);

arr[9] = ObjAnim(\_T("10.bmp"), hdc);

arr[10] = ObjAnim(\_T("11.bmp"), hdc);

arr[11] = ObjAnim(\_T("12.bmp"), hdc);

arr[12] = ObjAnim(\_T("13.bmp"), hdc);

arr[13] = ObjAnim(\_T("14.bmp"), hdc);

arr[14] = ObjAnim(\_T("15.bmp"), hdc);

arr[15] = ObjAnim(\_T("16.bmp"), hdc);

arr[16] = ObjAnim(\_T("17.bmp"), hdc);

arr[17] = ObjAnim(\_T("18.bmp"), hdc);

arr[18] = ObjAnim(\_T("19.bmp"), hdc);

arr[19] = ObjAnim(\_T("20.bmp"), hdc);

arr[20] = ObjAnim(\_T("21.bmp"), hdc);

arr[21] = ObjAnim(\_T("22.bmp"), hdc);

arr[22] = ObjAnim(\_T("23.bmp"), hdc);

arr[23] = ObjAnim(\_T("24.bmp"), hdc);

arr[24] = ObjAnim(\_T("25.bmp"), hdc);

arr[25] = ObjAnim(\_T("26.bmp"), hdc);

arr[26] = ObjAnim(\_T("27.bmp"), hdc);

arr[27] = ObjAnim(\_T("28.bmp"), hdc);

arr[28] = ObjAnim(\_T("29.bmp"), hdc);

arr[29] = ObjAnim(\_T("30.bmp"), hdc);

arr[30] = ObjAnim(\_T("31.bmp"), hdc);

//ship = ObjAnim(\_T("ship.bmp"), hdc);

//Sea = ObjAnim(\_T("Sea.bmp"), hdc);

ReleaseDC(hWnd, hdc);

break;

case WM\_TIMER:

t++;

//if (t > arr[0].bm.bmWidth) t = -1\* arr[0].bm.bmWidth;

InvalidateRect(hWnd, NULL, TRUE);

break;

case WM\_SIZE:

MoveWindow(hWnd, 100, 50, arr[0].bm.bmWidth + 2 \* border, arr[0].bm.bmHeight + caption

+ menu + border, TRUE);

break;

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case IDM\_EXIT: DestroyWindow(hWnd); break;

default: DefWindowProc(hWnd, message, wParam, lParam);

}

break;

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

arr[t % 31].Draw(0, 0, hdc);

//Sea.Draw(0, Sea.bm.bmHeight + caption + menu + border, hdc);

//ship.Draw(t, 100, hdc);

EndPaint(hWnd, &ps);

break;

case WM\_DESTROY:PostQuitMessage(0); break;

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

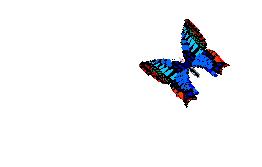
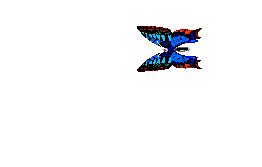
}

return 0;

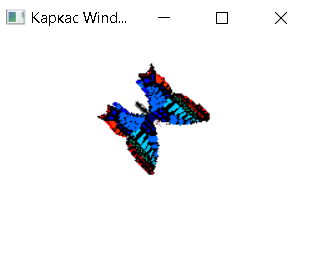
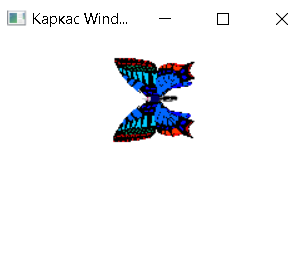
}

Картинки, используемые в программе:

1.bmp-31.bmp(в отчете только 5)



Результат:

Картинки должны быть одинакового размера для качественного отображения.

# Часть 3

Задание: В верхней части клиентской области окна находится надпись, показывающая номер текущей строки, выбранной в списке строк.

Своих функций я не использовал.

Листинг программы:

Lab4.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<Windows.h>

#include<tchar.h>

#include<string.h>

#include"Lab4.h"

HMENU hmenu, hpopup, hpopup2;

char szAppName[] = "calendar";

HWND hCB, hLB, hCY, hE, hT;

char Buf1[80], Buf[80];

HINSTANCE hInstance;

HBRUSH hbrush;

HDC hdc1, memdc;

PAINTSTRUCT ps;

LPSTR ReturnDate(int day, int month, int year);

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

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {

HWND hWnd;

WNDCLASSEX wc;

MSG msg;

ZeroMemory(&wc, sizeof(wc));

wc.cbSize = sizeof(wc);

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

wc.lpfnWndProc = WndProc;

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hInstance = hInstance;

wc.hIcon = LoadIcon(NULL, IDC\_ARROW);

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

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

wc.lpszMenuName = 0;

wc.lpszClassName = szAppName;

wc.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

if (!RegisterClassEx(&wc)) return FALSE;

hWnd = CreateWindow(szAppName, "CALENDAR", WS\_OVERLAPPEDWINDOW, 100, 100, 300, 530, (HWND)NULL, (HMENU)NULL,

hInstance, (LPSTR)NULL);

if (!hWnd) return FALSE;

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

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

TranslateMessage(&msg);

DispatchMessageW(&msg);

}

return ((int)msg.wParam);

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam) {

static HWND hwnded;

int i;

static int k = 0;

int maxx = 0, maxy = 0;

char day[10];

HBITMAP hbit;

switch (uMsg) {

case WM\_CREATE:

hdc1 = GetDC(hwnd);

maxy = GetSystemMetrics(SM\_CXSCREEN);

maxy = GetSystemMetrics(SM\_CXSCREEN);

hCB = CreateWindow("ComboBox", NULL, WS\_CHILD | WS\_VISIBLE | WS\_VSCROLL | CBS\_DROPDOWNLIST | CBS\_HASSTRINGS, 90, 80, 100, 200, hwnd, (HMENU)ID\_COMBODAY, hInstance, NULL);

//hT = CreateWindow("Button", "Выбрать", WS\_CHILD | WS\_VISIBLE, 90, 415, 100, 27, hwnd, (HMENU)ID\_TAKE, hInstance, NULL);

//hE = CreateWindow("Button", "Выход", WS\_CHILD | WS\_VISIBLE, 90, 450, 100, 27, hwnd, (HMENU)ID\_EXIT, hInstance, NULL);

for (i = 10; i < 54; i++)

{

\_itoa(i, day, 10);

SendMessage(hCB, CB\_ADDSTRING, 0, (LPARAM)day);

}

memdc = CreateCompatibleDC(hdc1);

hbit = CreateCompatibleBitmap(hdc1, maxx, maxy);

SelectObject(memdc, hbit);

break;

case WM\_SIZE:

InvalidateRect(hwnd, NULL, 1);

break;

case WM\_PAINT:

hdc1 = BeginPaint(hwnd, &ps);

SetTextColor(hdc1, RGB(255, 255, 67));

SetBkMode(hdc1, TRANSPARENT);

\_itoa(k, day, 10);

TextOut(hdc1, 85, 60, day, strlen(day));

memdc = CreateCompatibleDC(hdc1);

BitBlt(hdc1, 0, 0, maxx, maxy, memdc, 0, 0, SRCCOPY);

EndPaint(hwnd, &ps);

break;

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case ID\_COMBODAY:

if (HIWORD(wParam)== CBN\_SELENDOK) {

k = SendMessage(hCB, CB\_GETCURSEL, 0, 0L) + 1;

InvalidateRect(hwnd, NULL, TRUE);

}

break;

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

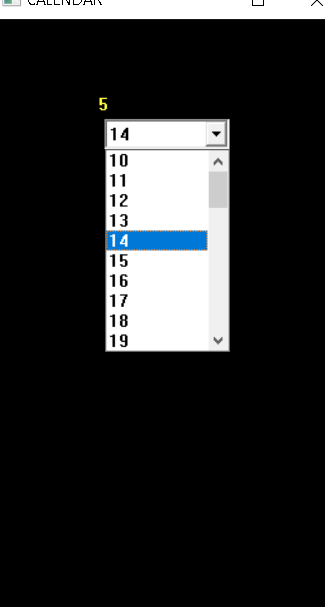
return DefWindowProc(hwnd, uMsg, wParam, lParam);

}

return 0;

} Картинки не использовал в программе.

Результат:



Чтобы в строке выводился номер строки, нужно преобразовать число в строку и сохранить в буфере(\_itoa(k, day, 10)) и обрабоать выбор в списке(CBN\_SELENDOK)

# Индивидуальное задание: 25 окон с крестиками

Задание: Вывести на экран 25 дочерних окон, в которых, при нажатии на них, отображается крест, при повторном нажатии исчезает

Листинг программы:

Lab7.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<Windows.h>

#include<Windowsx.h>

#include<tchar.h>

#define MoveTo(hdc, x, y) MoveToEx(hdc,x,y,NULL)

#define DIVISIONS 5

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

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

TCHAR szChildClass[] = \_T("Checker\_Child");

//==================================================================================

int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {

TCHAR szAppName[] = \_T("Checker");

HWND hMainWnd;

MSG msg;

WNDCLASSEX wc;

//Заполняем структуру класса окна

wc.cbSize = sizeof(wc);

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

wc.lpfnWndProc = WndProc;

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hInstance = hInstance;

wc.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

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

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

wc.lpszMenuName = NULL;

wc.lpszClassName = szAppName;

wc.hIconSm = LoadIcon(hInstance, szAppName);

//Регистрируем класс окна

RegisterClassEx(&wc);

wc.lpfnWndProc = ChildWndProc;

wc.cbWndExtra = sizeof(WORD);

wc.hIcon = NULL;

wc.lpszClassName = szChildClass;

wc.hIconSm = NULL;

if (!RegisterClassExW(&wc)) {

MessageBoxW(NULL, \_T("Cannot registr class"), \_T("Error"), MB\_OK);

return 0;

}

//Создаем основное окно приложения

hMainWnd = CreateWindow(szAppName, \_T("A Hello1 Application"), WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, NULL, NULL,

hInstance, NULL);

if (!hMainWnd) {

MessageBox(NULL, \_T("Cannot create main window"), \_T("Error"), MB\_OK);

return 0;

}

//Показываем окно

ShowWindow(hMainWnd, nCmdShow);

UpdateWindow(hMainWnd);

//Выполняем цикл обработки сообщений до закрытия окна

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

TranslateMessage(&msg);

DispatchMessageW(&msg);

}

return msg.wParam;

}

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

static HWND hwndChild[DIVISIONS][DIVISIONS];

int cxB, cyB, x, y;

switch (uMsg) {

case WM\_CREATE:

for (x = 0;x < DIVISIONS;x++)

for (y = 0; y < DIVISIONS; y++) {

hwndChild[x][y] = CreateWindow(szChildClass, NULL, WS\_CHILDWINDOW | WS\_VISIBLE, 0, 0, 0, 0, hWnd, (HMENU)(y << 8 | x), (HINSTANCE)GetWindowLong(hWnd, GWL\_HINSTANCE), NULL);

}

return 0;

case WM\_SIZE:

cxB = LOWORD(lParam) / DIVISIONS;

cyB = HIWORD(lParam) / DIVISIONS;

for (x = 0; x < DIVISIONS; x++)

for (y = 0; y < DIVISIONS; y++)

MoveWindow(hwndChild[x][y], x \* cxB, y \* cyB, cxB, cyB, TRUE);

return 0;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

LRESULT CALLBACK ChildWndProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam) {

HDC hdc;

PAINTSTRUCT ps;

RECT rect;

switch (uMsg)

{

//case WM\_CREATE:

//SetWindowWord(hwnd, 0, 0);

//return 0;

case WM\_LBUTTONDOWN:

SetWindowWord(hwnd, 0, 1 ^ GetWindowWord(hwnd, 0));

InvalidateRect(hwnd, NULL, FALSE);

case WM\_PAINT:

hdc = BeginPaint(hwnd, &ps);

GetClientRect(hwnd, &rect);

Rectangle(hdc, 0, 0, rect.right, rect.bottom);

if (GetWindowWord(hwnd, 0)) {

MoveTo(hdc, 0, 0);

LineTo(hdc, rect.right, rect.bottom);

MoveTo(hdc, 0, rect.bottom);

LineTo(hdc, rect.right, 0);

}

EndPaint(hwnd, &ps);

return 0;

}

return DefWindowProc(hwnd, uMsg, wParam, lParam);

}

Результат:

