НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Звіт з лабораторної роботи № 7**

Із курсу «Об’єктно орієнтоване програмування»

Виконав:

Студент групи ІП-64

Вінницький В’ячеслав

Варіант – 2

**Завдання**

Варіант завдання – 3:

Діалогове вікно з елементом списку (listbox) та двома кнопками: [Так] і [Відміна]. У список автоматично записуються назви груп нашого факультету. Якщо вибрати потрібний рядок списку і натиснути [Так], то у головному вікні повинен відображатися текст вибраного рядка списку.

**Текст коду вихідного файлу**

**Lab1.h**

// Отправить объявления функций, включенных в этот модуль кода:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

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

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

//INT\_PTR CALLBACK ID\_Dialog1(HWND, UINT, WPARAM, LPARAM);

//INT\_PTR CALLBACK ID\_Dialog2(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK ID\_Dialog3(HWND, UINT, WPARAM, LPARAM);

std::string buf;

**module2.h**

#pragma once

#include "lab1.h"

INT\_PTR CALLBACK ID\_Dialog3(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

SendDlgItemMessage(hDlg, IDC\_LIST3, LB\_ADDSTRING, 0, (LPARAM)"ІП-61");

SendDlgItemMessage(hDlg, IDC\_LIST3, LB\_ADDSTRING, 0, (LPARAM)"ІП-62");

SendDlgItemMessage(hDlg, IDC\_LIST3, LB\_ADDSTRING, 0, (LPARAM)"ІП-63");

SendDlgItemMessage(hDlg, IDC\_LIST3, LB\_ADDSTRING, 0, (LPARAM)"ІП-64");

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

if (LOWORD(wParam) == IDOK)

{

CHAR localpafch[100];

CHAR indx = SendDlgItemMessage(hDlg, IDC\_LIST3, LB\_GETCURSEL, 0, 0);

if (LB\_GETCURSEL == LB\_ERR)

EndDialog(hDlg, LOWORD(wParam));

SendDlgItemMessage(hDlg, IDC\_LIST3, LB\_GETTEXT, indx, (long)localpafch);

buf = localpafch;

EndDialog(hDlg, LOWORD(wParam));

}

break;

}

return (INT\_PTR)FALSE;

}

**Lab1.cpp**

// lab1.cpp: определяет точку входа для приложения.

//

#include "stdafx.h"

#include "lab1.h"

#include "string"

#include "module2.h"

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: разместите код здесь.

// Инициализация глобальных строк

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

LoadStringW(hInstance, IDC\_LAB1, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LAB1));

MSG msg;

// Цикл основного сообщения:

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

{

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

//

// ФУНКЦИЯ: MyRegisterClass()

//

// НАЗНАЧЕНИЕ: регистрирует класс окна.

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW 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\_LAB1));

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

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

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_LAB1);

wcex.lpszClassName = szWindowClass;

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

return RegisterClassExW(&wcex);

}

//

// ФУНКЦИЯ: InitInstance(HINSTANCE, int)

//

// НАЗНАЧЕНИЕ: сохраняет обработку экземпляра и создает главное окно.

//

// КОММЕНТАРИИ:

//

// В данной функции дескриптор экземпляра сохраняется в глобальной переменной, а также

// создается и выводится на экран главное окно программы.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Сохранить дескриптор экземпляра в глобальной переменной

HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

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

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

//

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

//

// НАЗНАЧЕНИЕ: обрабатывает сообщения в главном окне.

//

// WM\_COMMAND — обработать меню приложения

// WM\_PAINT — отрисовать главное окно

// WM\_DESTROY — отправить сообщение о выходе и вернуться

//

//

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

{

switch (message)

{

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Разобрать выбор в меню:

switch (wmId)

{

case IDM\_ABOUT:

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

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

/\* case ID\_lab1:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_DIALOG1), hWnd, ID\_Dialog1);

break;\*/

case ID\_lab2:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_DIALOG3), hWnd, ID\_Dialog3);

InvalidateRect(hWnd, nullptr, true);

break;

default:

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

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

TextOut(hdc, 40, 40, buf.c\_str(), buf.size());

// TODO: Добавьте сюда любой код прорисовки, использующий HDC...

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

// Обработчик сообщений для окна "О программе".

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

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

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

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

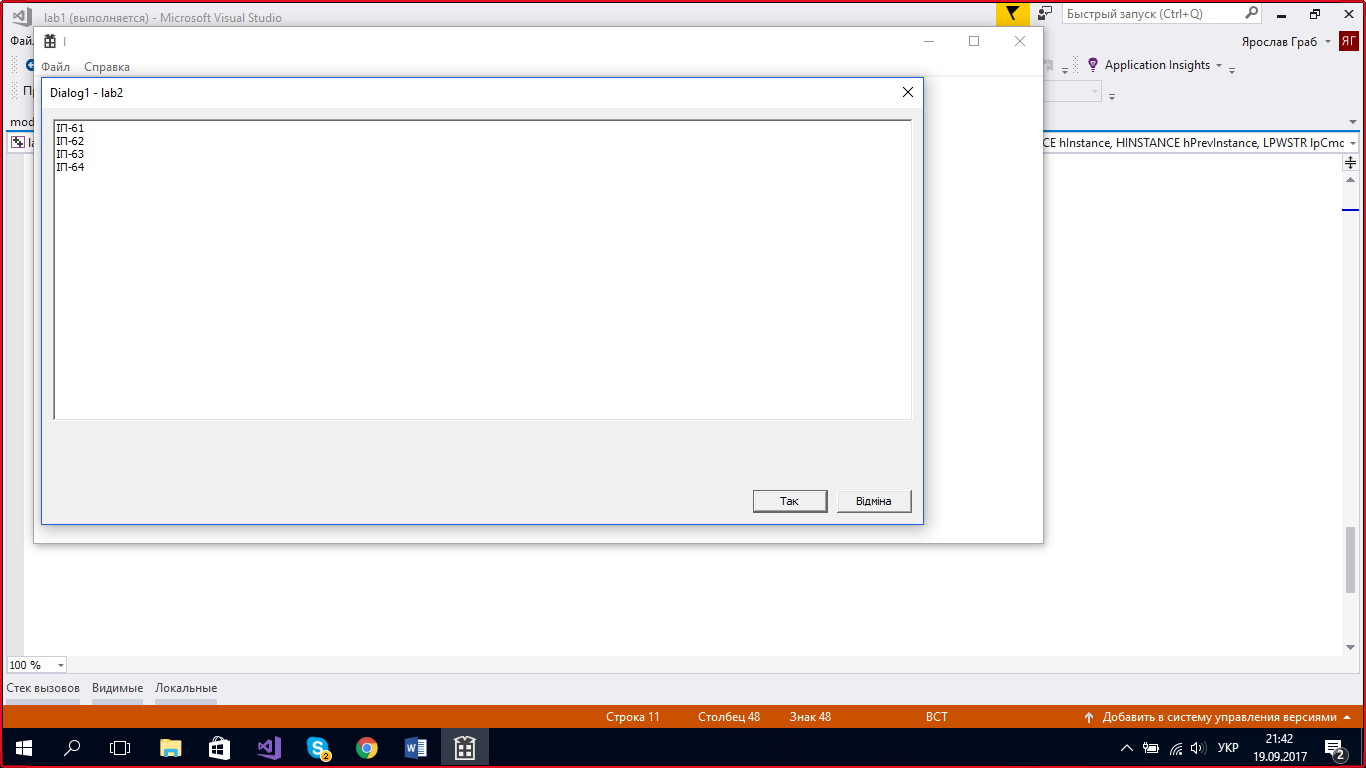
break;

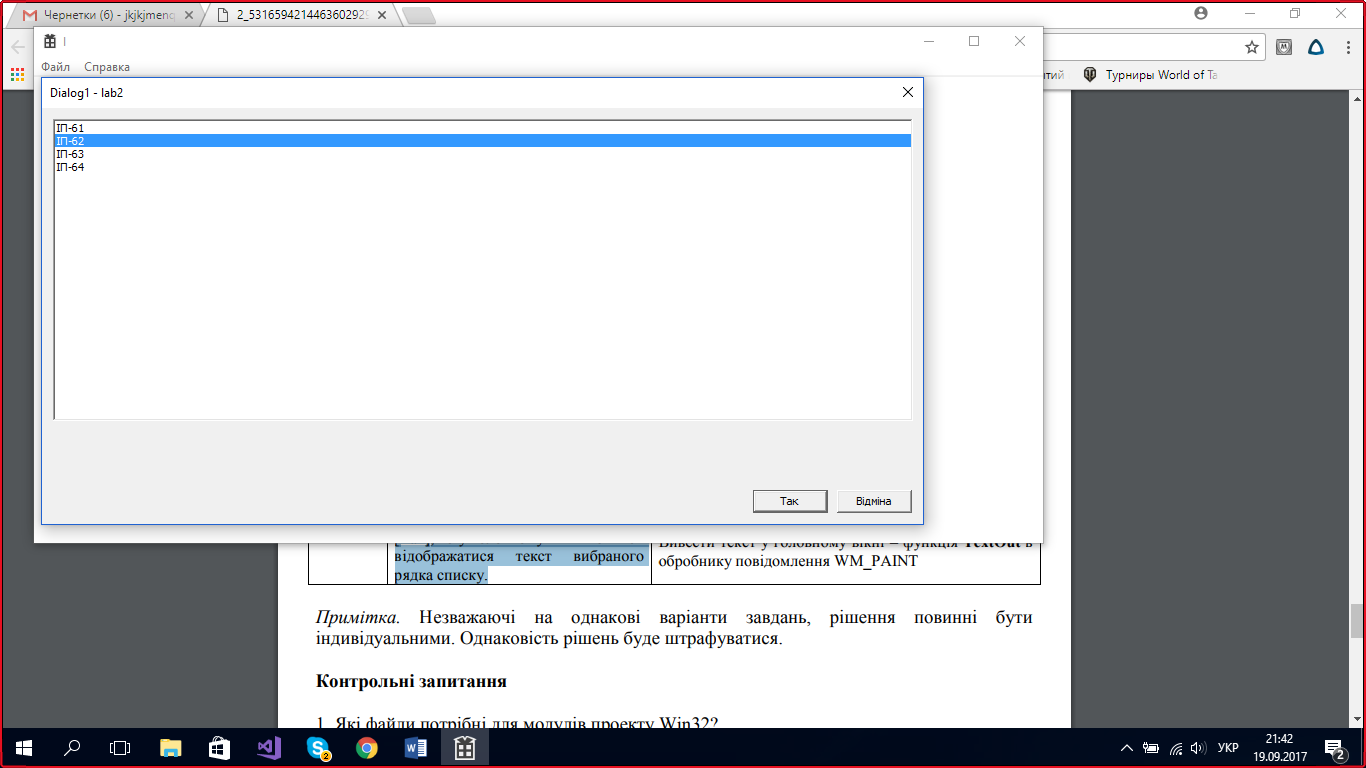
}

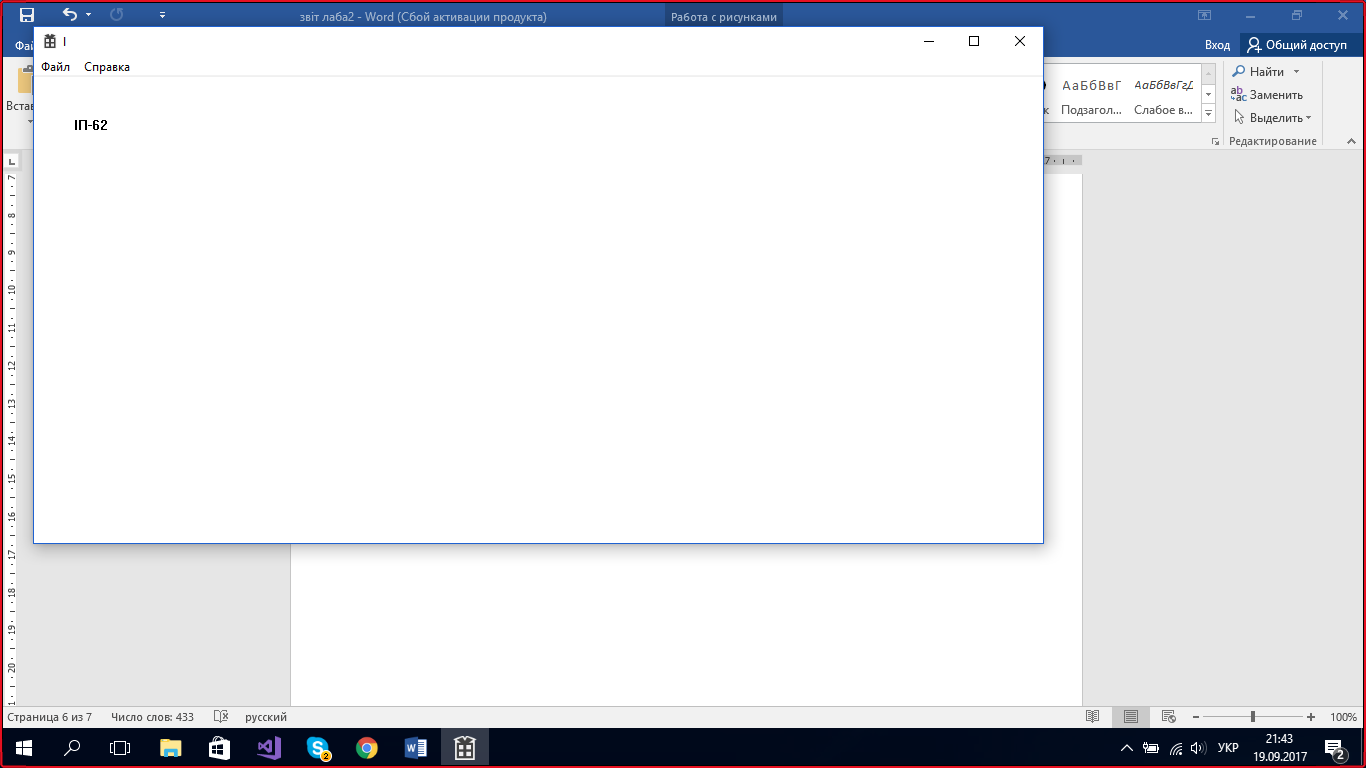
return (INT\_PTR)FALSE;

}

**Скріншоти**







**Висновки**

Я навчився розділяти текст програми на модулі, в залежності від того, яку функцію виконує той чи інший фрагмент коду.