**Міністерство освіти і науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра обчислювальної техніки**

**Лабораторна робота №1**

з дисципліни

«ООП»

на тему «Знайомство із середовищем розробки програм Microsoft Visual Studio та складання

модульних проектів програм на C++»

**Варіант № 2, 3**

Виконав: Перевірив:

студент групи ІП-93 Порєв Віктор Миколайович

Завальнюк Максим Євгенович

номер залікової книжки: 9312

Київ 2020

**Завдання на лабораторну роботу:**

**Мета:** отримати перші навички створення програм для Windows на основі проектів API

Win32 для Visual C++ і навчитися модульному програмуванню на C++.

**Завдання:**

1. Створити у середовищі MS Visual Studio C++ проект Win32 з ім’ям **Lab1**.

2. Написати вихідний текст програми згідно варіанту завдання.

3. Скомпілювати вихідний текст і отримати виконуваний файл програми.

4. Перевірити роботу програми. Налагодити програму.

5. Проаналізувати та прокоментувати результати та вихідний текст програми.

**Варіант 2:**

Два діалогових вікна. Спочатку з’являється перше, яке має дві кнопки: [Далі >] і [Відміна]. Якщо натиснути кнопку [Далі >], то з’явиться друге діалогове вікно, яке має три кнопки: [< Назад], [Так] і [Відміна]. Якщо натиснути кнопку [< Назад], то перехід до першого діалогово вікна.

**Варіант 3:**

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

**Вихідний текст файлів:**

* *Lab1.cpp*

// Lab1.cpp : main file of programm

//

#include "framework.h"

#include "Lab1.h"

#include "module1.h" // import module

#include "module2.h" // import module

#define MAX\_LOADSTRING 100

// Global variables:

HINSTANCE hInst; // current instance

WCHAR szTitle[MAX\_LOADSTRING]; // Title bar text

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

char buf[1024];

// Submit 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);

void paintOnWindow(HWND hWnd); // Paint function

// Main function

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// Init global strings

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

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

MyRegisterClass(hInstance);

// Execute initialization of program:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

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

MSG msg;

// Main message loop:

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

{

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

//

// FUNCTION: MyRegisterClass()

//

// PURPOSE: Registers a window class.

//

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);

}

//

// FUNCTION: InitInstance(HINSTANCE, int)

//

// PURPOSE: Retains the instance marker and creates the main window

//

// COMMENTS:

//

// In this function, the instance marker is stored in a global variable as well as

// the main program window is created and displayed.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Store instance marker in a global variable

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;

}

//

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

//

// PURPOSE: Processes messages in the main window.

//

// WM\_COMMAND - process application menu

// WM\_PAINT - Rendering the main window

// WM\_DESTROY - send an exit message and return

//

//

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

{

switch (message)

{

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Parse the selection in the menu:

switch (wmId)

{

case IDM\_WORK\_MOD1: // Our first button

Func\_MOD1(hInst, hWnd);

break;

case IDM\_WORK\_MOD2: // Our second button

Func\_MOD2(hInst, hWnd, buf);

break;

case IDM\_ABOUT:

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

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

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

}

}

break;

case WM\_PAINT:

paintOnWindow(hWnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

// Message handler for the "About" window.

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;

}

void paintOnWindow(HWND hWnd) {

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

TextOut(hdc, 1, 1, buf, 1024); // Output the text in buf

EndPaint(hWnd, &ps);

ZeroMemory(buf, 1024);

}

* *module1.cpp*

#include "framework.h"

#include "module1.h"

#include "module3.h"

HINSTANCE hinst1;

static BOOL CALLBACK Work\_MOD1(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);

int Func\_MOD1(HINSTANCE hInst, HWND hWnd)

{

return DialogBox(hInst, MAKEINTRESOURCE(IDD\_WORK\_MOD1), hWnd, Work\_MOD1);

}

BOOL onClick(HWND hDlg, WPARAM wParam)

{

return EndDialog(hDlg, LOWORD(wParam));

}

// First callback

BOOL CALLBACK Work\_MOD1(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_COMMAND:

switch(LOWORD(wParam))

{

case IDC\_CANCEL\_MOD1:

onClick(hDlg, wParam);

DefWindowProc(hDlg, message, wParam, lParam);

return TRUE;

case IDC\_NEXT\_MOD1:

onClick(hDlg, wParam);

Func\_MOD3(hinst1, hDlg);

return TRUE;

}

break;

case WM\_CLOSE:

EndDialog(hDlg, 0);

break;

}

return FALSE;

}

* *module1.h*

#pragma once

#include "resource1.h"

extern int Func\_MOD1(HINSTANCE hInst, HWND hWnd);

* *module2.cpp*

#include "framework.h"

#include "module2.h"

static BOOL CALLBACK Work\_MOD3(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);

int Func\_MOD2(HINSTANCE hInst, HWND hWnd, CHAR buf)

{

BOOL window = DialogBox(hInst, MAKEINTRESOURCE(IDD\_WORK\_MOD3), hWnd, Work\_MOD3);

InvalidateRect(hWnd, 0, TRUE);

return window;

}

// First callback

BOOL CALLBACK Work\_MOD3(HWND hDlg, UINT iMessage, WPARAM wParam, LPARAM lParam, CHAR buf)

{

switch (iMessage)

{

case WM\_INITDIALOG:

// Init list

SendDlgItemMessage(hDlg, IDC\_LIST\_MOD2, LB\_ADDSTRING, 0, (LPARAM)"IP-93");

SendDlgItemMessage(hDlg, IDC\_LIST\_MOD2, LB\_ADDSTRING, 0, (LPARAM)"IP-94");

SendDlgItemMessage(hDlg, IDC\_LIST\_MOD2, LB\_ADDSTRING, 0, (LPARAM)"IP-95");

SendDlgItemMessage(hDlg, IDC\_LIST\_MOD2, LB\_ADDSTRING, 0, (LPARAM)"IP-96");

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK)

{

INT indx = SendDlgItemMessage(hDlg, IDC\_LIST\_MOD2, LB\_GETCURSEL, 0, 0); // Get index

SendDlgItemMessage(hDlg, IDC\_LIST\_MOD2, LB\_GETTEXT, indx, (long)buf); // Get message by index

EndDialog(hDlg, 1);

break;

}

if (LOWORD(wParam) == IDCANCEL) EndDialog(hDlg, 0);

break;

case WM\_CLOSE:

EndDialog(hDlg, 0);

break;

return FALSE;

}

* *module2.h*

#pragma once

#include "resource2.h"

extern int Func\_MOD2(HINSTANCE hInst, HWND hWnd, CHAR \*buf);

* *module3.cpp*

#include "framework.h"

#include "module1.h"

#include "module3.h"

HINSTANCE hinst3;

static BOOL CALLBACK Work\_MOD3(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);

int Func\_MOD3(HINSTANCE hInst, HWND hWnd)

{

return DialogBox(hInst, MAKEINTRESOURCE(IDD\_WORK\_MOD2), hWnd, Work\_MOD3);

}

BOOL static onClick(HWND hDlg, WPARAM wParam)

{

return EndDialog(hDlg, LOWORD(wParam));

}

BOOL CALLBACK Work\_MOD3(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam) {

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case IDC\_OK\_MOD2: // Next button

onClick(hDlg, wParam);

return TRUE;

case IDC\_CANCEL\_MOD2: // Cancel button

onClick(hDlg, wParam);

DefWindowProc(hDlg, message, wParam, lParam);

return TRUE;

case IDC\_BACK\_MOD2: // Back button

onClick(hDlg, wParam);

Func\_MOD1(hinst3, hDlg);

return TRUE;

}

break;

case WM\_CLOSE:

EndDialog(hDlg, 0);

break;

}

return FALSE;

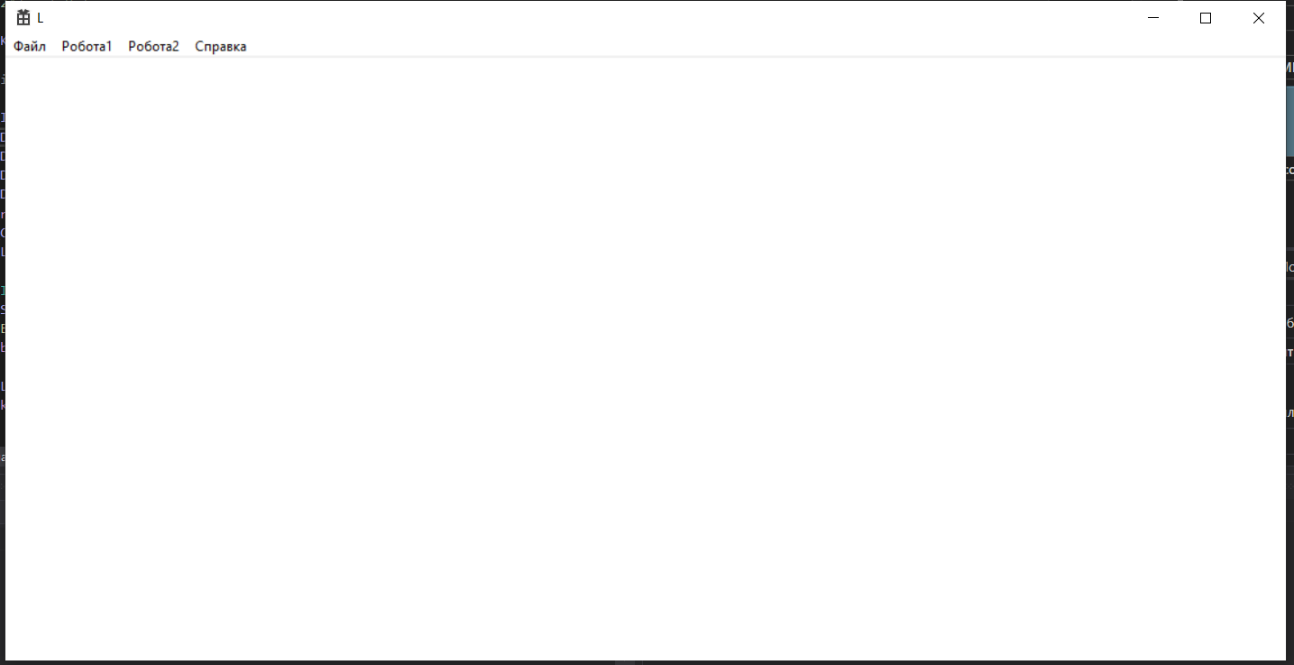
}

* *module3.h*

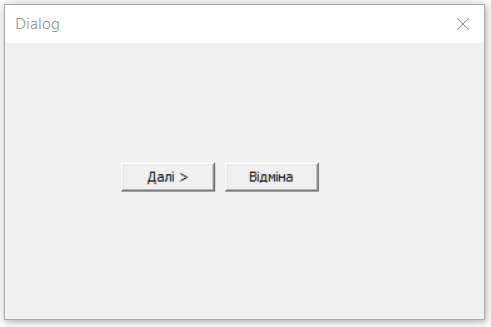
#pragma once

#include "resource1.h"

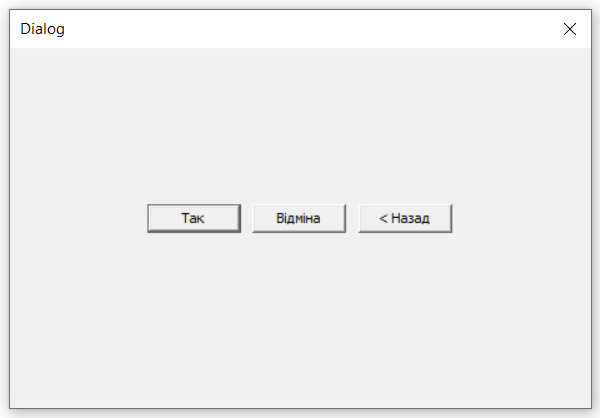
extern int Func\_MOD3(HINSTANCE hInst, HWND hWnd);

**Скріншоти роботи:**

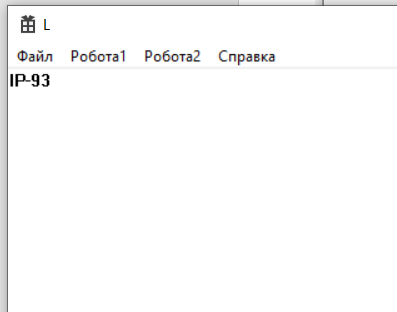
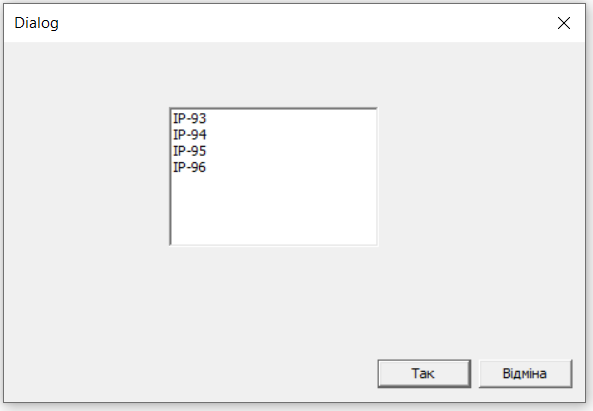
Головне вікно



Діалогове вікно для першого завдання



Наступне діалогове вікно для першого завдання



Діалогове вікно для другого завдання

Результат дії другого завдання

**#include-ієрархія файлів:**

**Lab1.h**

**Lab1.cpp**

Константи ідентифікаторів ресурсів module2

Константи ідентифікаторів ресурсів module1

**resource.h**

Константи ідентифікаторів ресурсів

**\*.h**

**Lab1.rc**

#include "resource.h"

. . .

#include "targetver.h"

. . .

#include <windows.h>

**targetver.h**

Головний заголовочний файл Windows API

Інші заголовочні файли Windows API

**\*.h**

**windows.h**

**framework.h**

#include "resource.h"

Заголовочні файли стандартних бібліотек C++

Константи ідентифікаторів для опису версій Windows

#include "targetver.h"

#include <windows.h>

// Файлы заголовков среды выполнения C

#include <stdlib.h>

// Lab1.cpp : main file of programm

//

#include "framework.h"

#include "Lab1.h"

#include "module1.h"

#include "module2.h"

. . .

int APIENTRY wWinMain(…)

{

. . .

}

**Контрольні запитання:**

1. Що робить головна функція програми?

Головна функція є точкою входу програми, початок роботи програми повністю описується кодом цієї функції. Загалом для будь-якої Windows-програми спочатку реєструється клас вікна, потім це вікно створюється і далі програма входить у цикл очікування повідомлень. Подальша робота залежіть вже від того, які повідомлення будуть надходити на адресу головного вікна програми.

2. Чим відрізняється реєстрація вікна від його створення?

На етапі реєстрації нового вікна ми вказуємо, де саме у коді повинно викликатися вікно, а при створенні - безпосередньо створюємо його.

3. Що робить у віконній функції оператор switch?

Відслідковує команди, які приходять у діалогове вікно

4. Як створюється новий пункт меню?

Меню є елементом графічного інтерфейсу користувача (GUI) програм. Щоб додати пункт меню, потрібно викликати редактор ресурсів. Для цього треба двічи клікнути у вікні "Обозреватель решения" (Solution Explorer) на файлі Lab1.rc Після кліку на файлі Lab1.rc з'являється редактор ресурсів. У вікні списку ресурсів треба вибрати меню і двічі клікнути – відкриється меню для редагування. Вводимо назву нового пункту меню, наприклад, "Work". За умовчанням редактор ресурсів присвоїть новому пункту меню ідентифікатор ID\_32771. Це не дуже зрозуміле імя константи, тому краще його змінити.

5. Як викликається діалогове вікно?

Фукнцією, яка описана у кейсі меню, а саме – DialogBox().

6. Як створити нове діалогове вікно?

Спочатку у редакторі ресурсів треба додати нове діалогове вікно (Dialog) у файл ресурсів \*.rc. Після цього запрограмувати віконну функцію даного діалогового вікна. Накінець запрограмувати виклик діалогового вікна у функції головного вікна, наприклад, MyWork().

7. Де запрограмований цикл очікування повідомлень головного вікна?

Цикл очікування повідомлень головного вікна запрограмований у головній функції проекту, яка є точкою входу програми, наприклад, \_tWinMain.

8. Що таке WM\_COMMAND?

Як зазначено в коментарях до проекту, WM\_COMMAND - це обробка меню програми.

9. Які файли потрібні для модулів проекту Win32?

Вихідні файли - \*.cpp, файли заголовків - \*.h, файли ресурсів - \*.rc.

10. Що таке #include – ієрархія?

Описує відношення включення файлів проекту директивами #include. У такій схемі вказуються файли, які містять директиви #include, а також файли, які включаються цими директивами. На найнижчому рівні ієрархії розташовуються файли, які не містять #include. Наступний рівень посідають файли, у текстах яких записані #include файлів найнижчого рівня. І так далі.

11. Що робить директива #include?

Директива #include пропонує компілятору включити інший вихідний файл, ім'я якого вказується після директиви. Файл лежить в подвійних лапках або в <>.

12. Чим відрізняється віконна функція головного вікна від діалогового?

Кардинальних відмінностей немає, кожна виконує функції для «своїх» вікон.

13. Що означає роздільна компіляція модулів і як її досягти?

Якщо вихідний код програми розділений на декілька файлів, то процес компіляції і збірки відбувається у такому порядку: спочатку всі одиниці компіляції окремо компілюються, а потім компоновщик збирає отримані об'єктні модулі (з підключенням бібліотек) у виконуваний файл. Власне, цей процес і називається роздільної компіляцією.

Її можна досягти за допомогою #include-ієрархії. Для цього функції модулів не використовуються безпосередньо у головному файлі програми, а виконуються імпортом через файли модулів .h.

**Висновок:** у ході виконання роботи я познайомився із технологією Win32, створив перший свій проект, ознайомився із створенням діалогових вікон, запрограмував їхню поведінку через функції, а також на практиці задіяв роздільну компіляцію модулів.