**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 № 1, 2**

Виконав: Перевірив:

студент групи ІП-93 Порєв Віктор Миколайович

Домінський Валентин Олексійович

номер залікової книжки: 9311

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 1:**

Діалогове вікно з повзуном горизонтального скролінгу (Horizontal scroll Bar) та дві кнопки: [Так] і [Відміна]. Рухаючи повзунок скролінгу користувач вводить число у діапазоні від 1 до 100. Після натискування кнопки [Так] вибране число буде відображатися у головному вікні

**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.

**Вихідні тексти файлів:**

* Lab1.cpp (фрагменти, що ілюструють власний код)

// Lab1.cpp : Defines the input point for the application.

//

// First Part

#include "pch.h"

#include "framework.h"

#include "Lab1.h"

#include "module1.h"

#include "module2.h"

…

static void Work1**(**HWND hWnd**);** // Declaration of our function

static void Work2**(**HWND hWnd**);** // Declaration of our function

static void DrawTextOnScreen**(**HWND hWnd**);** // Declaration of our function

…

#pragma region ModifiedFuntions

// Third Part

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

//

// OBJECTIVE: Processes messages in the main window.

//

// WM\_COMMAND - Process the application menu

// WM\_PAINT - Drawing of the main window

// WM\_DESTROY - Send message about exit and return

//

//

/// <summary>

/// Processes messages in the main window.

/// </summary>

/// <param name="hWnd">The h WND.</param>

/// <param name="message">The message.</param>

/// <param name="wParam">The w parameter.</param>

/// <param name="lParam">The l parameter.</param>

/// <returns></returns>

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

{

switch (message)

{

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Disassemble the selection in the menu:

switch (wmId)

{

case IDM\_WORK\_MOD1:

// first menu

Work1(hWnd);

// The update region represents the portion of the window's

// client area that must be redrawn.

InvalidateRect(hWnd, 0, TRUE);

break;

case IDM\_WORK\_MOD2:

// second menu

Work2(hWnd);

// The update region represents the portion of the window's

// client area that must be redrawn.

InvalidateRect(hWnd, 0, TRUE);

break;

case IDM\_ABOUT:

// About Menu

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

break;

case IDM\_EXIT:

// Exit menu

DestroyWindow(hWnd);

break;

default:

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

}

}

break;

case WM\_PAINT:

{

DrawTextOnScreen(hWnd);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

…

/// <summary>

/// Function-handler of the menu item "Work1"

/// </summary>

/// <param name="hWnd">The h WND.</param>

void Work1(HWND hWnd)

{

// What we program here that will be done

Func\_MOD1(hInst,hWnd);

}

/// <summary>

/// Function-handler of the menu item "Work2"

/// </summary>

/// <param name="hWnd">The h WND.</param>

void Work2(HWND hWnd)

{

// What we program here that will be done

Func\_MOD2(hInst, hWnd);

}

/// <summary>

/// Draws the text on screen.

/// </summary>

/// <param name="hWnd">hWND</param>

void DrawTextOnScreen(HWND hWnd)

{

PAINTSTRUCT ps;

UpdateWindow(hWnd);

HDC hdc = BeginPaint(hWnd, &ps);

if (canWrite\_MOD1)

{

\_itoa\_s(pos\_MOD1, tempPlaceForText\_MOD1, sizeof(pos\_MOD1), 10);

canWrite\_MOD1 = FALSE;

}

TextOut(hdc, textHeightPosition, textWidthPosition, " ", 7);

TextOut(hdc, textHeightPosition, textWidthPosition, tempPlaceForText\_MOD1, numOfDig\_MOD1);

ZeroMemory(tempPlaceForText\_MOD1, pos\_MOD1);

EndPaint(hWnd, &ps);

}

#pragma endregion

* module1.cpp

#include "pch.h"

#include "framework.h"

#include "module1.h"

#pragma region VariablesAndFunctionsDeclarations

static int const maxSymbols\_MOD1 **=** 255**;**

char tempPlaceForText\_MOD1**[**maxSymbols\_MOD1**]** **=** **{** 0 **};**

int pos\_MOD1**;**

static int nMinPos\_MOD1 **=** 1**;**

static int nMaxPos\_MOD1 **=** 100**;**

static HWND hWndScrollBar\_MOD1**;**

BOOL canWrite\_MOD1 **=** FALSE**;**

int numOfDig\_MOD1**;**

static INT\_PTR CALLBACK Work1\_MOD1**(**HWND hDlg**,** UINT iMessage**,** WPARAM wParam**,** LPARAM lParam**);**

static void OnInit\_MOD1**(**HWND hDlg**);**

static void OnLineLeft\_MOD1**(**HWND hDlg**);**

static void OnLineRight\_MOD1**(**HWND hDlg**);**

static void OnOkMod2\_MOD1**(**HWND hDlg**);**

static void OnThumbPosAndTrack\_MOD1**(**HWND hDlg**,** WPARAM wParam**);**

static int Count\_MOD1**(**int pos\_MOD1**);**

#pragma endregion

#pragma region Functions

/// <summary>

/// dialog box creation function

/// </summary>

/// <param name="hInst">The hinst.</param>

/// <param name="hWnd">The hWND.</param>

/// <returns></returns>

int Func\_MOD1**(**HINSTANCE hInst**,** HWND hWnd**)**

**{**

**return** DialogBox**(**hInst**,** MAKEINTRESOURCE**(**IDD\_WORK\_MOD1**),** hWnd**,** Work1\_MOD1**);**

**}**

/// <summary>

/// Callback-function for hor.scrollbar

/// </summary>

/// <param name="hDlg"></param>

/// <param name="iMessage"></param>

/// <param name="wParam"></param>

/// <param name="lParam"></param>

/// <returns></returns>

INT\_PTR CALLBACK Work1\_MOD1(HWND hDlg, UINT iMessage, WPARAM wParam, LPARAM lParam)

{

switch (iMessage)

{

case WM\_INITDIALOG:

OnInit\_MOD1(hDlg);

break;

case WM\_HSCROLL:

pos\_MOD1 = GetScrollPos(GetDlgItem(hDlg, IDC\_SCROLLBAR1\_MOD1), SB\_CTL);

switch (LOWORD(wParam))

{

case SB\_LINELEFT:

OnLineLeft\_MOD1(hDlg);

break;

case SB\_LINERIGHT:

OnLineRight\_MOD1(hDlg);

break;

case SB\_THUMBPOSITION:

case SB\_THUMBTRACK:

OnThumbPosAndTrack\_MOD1(hDlg, wParam);

break;

default: break;

}

SetScrollPos(hWndScrollBar\_MOD1, SB\_CTL, pos\_MOD1, TRUE);

break;

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case IDOK:

OnOkMod2\_MOD1(hDlg);

return (INT\_PTR)TRUE;

break;

case IDCANCEL:

EndDialog(hDlg, 0);

return (INT\_PTR)TRUE;

break;

}

break;

case WM\_CLOSE:

{

EndDialog(hDlg, 0);

}

break;

default: break;

}

return FALSE;

}

/// <summary>

/// Called on initializing

/// </summary>

/// <param name="hDlg">The dialog.</param>

void OnInit\_MOD1(HWND hDlg)

{

hWndScrollBar\_MOD1 = GetDlgItem(hDlg, IDC\_SCROLLBAR1\_MOD1);

pos\_MOD1 = 1;

SetScrollRange(hWndScrollBar\_MOD1, SB\_CTL, nMinPos\_MOD1, nMaxPos\_MOD1, TRUE);

}

/// <summary>

/// Called when scroll pos goes to the left.

/// </summary>

/// <param name="hDlg">The dialog.</param>

void OnLineLeft\_MOD1(HWND hDlg)

{

if (pos\_MOD1 != nMinPos\_MOD1)

{

pos\_MOD1--;

}

SetDlgItemInt(hDlg, IDC\_STATIC\_MOD1, pos\_MOD1, TRUE);

}

/// <summary>

/// Called when scroll pos goes to the right.

/// </summary>

/// <param name="hDlg">The dialog.</param>

void OnLineRight\_MOD1(HWND hDlg)

{

if (pos\_MOD1 != nMaxPos\_MOD1)

{

pos\_MOD1++;

}

SetDlgItemInt(hDlg, IDC\_STATIC\_MOD1, pos\_MOD1, TRUE);

}

/// <summary>

/// Called when thumb position has changed

/// </summary>

/// <param name="hDlg">The h dialog.</param>

/// <param name="wParam">The w parameter.</param>

void OnThumbPosAndTrack\_MOD1(HWND hDlg, WPARAM wParam)

{

pos\_MOD1 = HIWORD(wParam);

SetDlgItemInt(hDlg, IDC\_STATIC\_MOD1, pos\_MOD1, TRUE);

}

/// <summary>

/// Called when IDOK clicked

/// </summary>

/// <param name="hDlg">The dialog.</param>

void OnOkMod2\_MOD1(HWND hDlg)

{

canWrite\_MOD1 = TRUE;

numOfDig\_MOD1 = Count\_MOD1(pos\_MOD1);

EndDialog(hDlg, 1);

}

/// <summary>

/// Function to Count how many digits are in int

/// </summary>

/// <param name="pos"></param>

/// <returns></returns>

int Count\_MOD1(int pos\_MOD1)

{

int count\_MOD1 = 0;

while (pos\_MOD1 != 0)

{

pos\_MOD1 = pos\_MOD1 / 10;

++count\_MOD1;

}

return count\_MOD1;

}

#pragma endregion

* module1.h

//{{NO\_DEPENDENCIES}}

// Included file created in Microsoft Visual C++.

// Using module1.rc

//

#ifndef MODULE\_1\_H

#define MODULE\_1\_H

#define IDD\_WORK\_MOD1 101

#define IDC\_SCROLLBAR1\_MOD1 1002

#define IDC\_STATIC\_MOD1 1003

#pragma region VariablesAndFuntions

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

extern char tempPlaceForText\_MOD1**[];**

extern int pos\_MOD1**;**

extern int numOfDig\_MOD1**;**

extern BOOL canWrite\_MOD1**;**

#pragma endregion

#endif // MODULE\_1\_H

// Next default values for new objects

//

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NEXT\_RESOURCE\_VALUE 103

#define \_APS\_NEXT\_COMMAND\_VALUE 40001

#define \_APS\_NEXT\_CONTROL\_VALUE 1005

#define \_APS\_NEXT\_SYMED\_VALUE 101

#endif

#endif

* module2.cpp

#include "pch.h"

#include "framework.h"

#include "module2.h"

#include "Resource.h"

#pragma region FunctionsDeclaration

static void OnNextMod2**(**HWND hDlg**);**

static void OnBackMod2**(**HWND hDlg**);**

static INT\_PTR CALLBACK Work1\_MOD2**(**HWND hDlg**,** UINT message**,** WPARAM wParam**,** LPARAM lParam**);**

static INT\_PTR CALLBACK Work2\_MOD2**(**HWND hDlg**,** UINT message**,** WPARAM wParam**,** LPARAM lParam**);**

#pragma endregion

#pragma region Functions

/// <summary>

/// dialog box creation function

/// </summary>

/// <param name="hInst">The hinst.</param>

/// <param name="hWnd">The hWND.</param>

/// <returns></returns>

int Func\_MOD2**(**HINSTANCE hInst**,** HWND hWnd**)**

**{**

**return** DialogBox**(**hInst**,** MAKEINTRESOURCE**(**IDD\_WORK1\_MOD2**),** hWnd**,** Work1\_MOD2**);**

**}**

/// <summary>

/// Callback-function for first dialog window

/// </summary>

/// <param name="hDlg">The h dialog.</param>

/// <param name="message">The message.</param>

/// <param name="wParam">The w parameter.</param>

/// <param name="lParam">The l parameter.</param>

/// <returns></returns>

INT\_PTR CALLBACK Work1\_MOD2**(**HWND hDlg**,** UINT message**,** WPARAM wParam**,** LPARAM lParam**)**

**{**

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case IDC\_CANCEL1\_MOD2:

EndDialog(hDlg, 0);

return (INT\_PTR)TRUE;

case IDC\_NEXT\_MOD2:

OnNextMod2(hDlg);

return (INT\_PTR)TRUE;

}

break;

case WM\_CLOSE:

{

EndDialog(hDlg, 0);

}

break;

}

return (INT\_PTR)FALSE;

}

/// <summary>

/// Called when IDC\_NEXT\_MOD2 clicked

/// </summary>

/// <param name="hDlg">The dialog.</param>

void OnNextMod2(HWND hDlg)

{

EndDialog(hDlg, 1);

DialogBox(hInst, MAKEINTRESOURCE(IDD\_WORK2\_MOD2), hDlg, Work2\_MOD2);

}

/// <summary>

/// Callback-function for second dialog window

/// </summary>

/// <param name="hDlg">The h dialog.</param>

/// <param name="message">The message.</param>

/// <param name="wParam">The w parameter.</param>

/// <param name="lParam">The l parameter.</param>

/// <returns></returns>

INT\_PTR CALLBACK Work2\_MOD2(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case IDC\_OK2\_MOD2: // Next button

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

case IDC\_CANCEL2\_MOD2: // Cancel button

EndDialog(hDlg, 0);

return (INT\_PTR)TRUE;

case IDC\_BACK\_MOD2: // Back button

OnBackMod2(hDlg);

return (INT\_PTR)TRUE;

}

break;

case WM\_CLOSE:

{

EndDialog(hDlg, 0);

}

break;

}

return (INT\_PTR)FALSE;

}

/// <summary>

/// Called when IDC\_BACK\_MOD2 clicked

/// </summary>

/// <param name="hDlg">The dialog.</param>

void OnBackMod2(HWND hDlg)

{

EndDialog(hDlg, 1);

DialogBox(hInst, MAKEINTRESOURCE(IDD\_WORK1\_MOD2), hDlg, Work1\_MOD2);

}

#pragma endregion

* module2.h

//{{NO\_DEPENDENCIES}}

// Included file created in Microsoft Visual C++.

// Using module2.rc

//

#ifndef MODULE\_2\_H

#define MODULE\_2\_H

#define IDC\_BACK\_MOD2 3

#define IDD\_WORK2\_MOD2 103

#define IDD\_WORK1\_MOD2 105

#define IDC\_CANCEL2\_MOD2 1002

#define IDC\_OK2\_MOD2 1003

#define IDC\_NEXT\_MOD2 1006

#define IDC\_CANCEL1\_MOD2 1007

#pragma region Functions

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

#pragma endregion

// Next default values for new objects

//

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NEXT\_RESOURCE\_VALUE 107

#define \_APS\_NEXT\_COMMAND\_VALUE 40001

#define \_APS\_NEXT\_CONTROL\_VALUE 1008

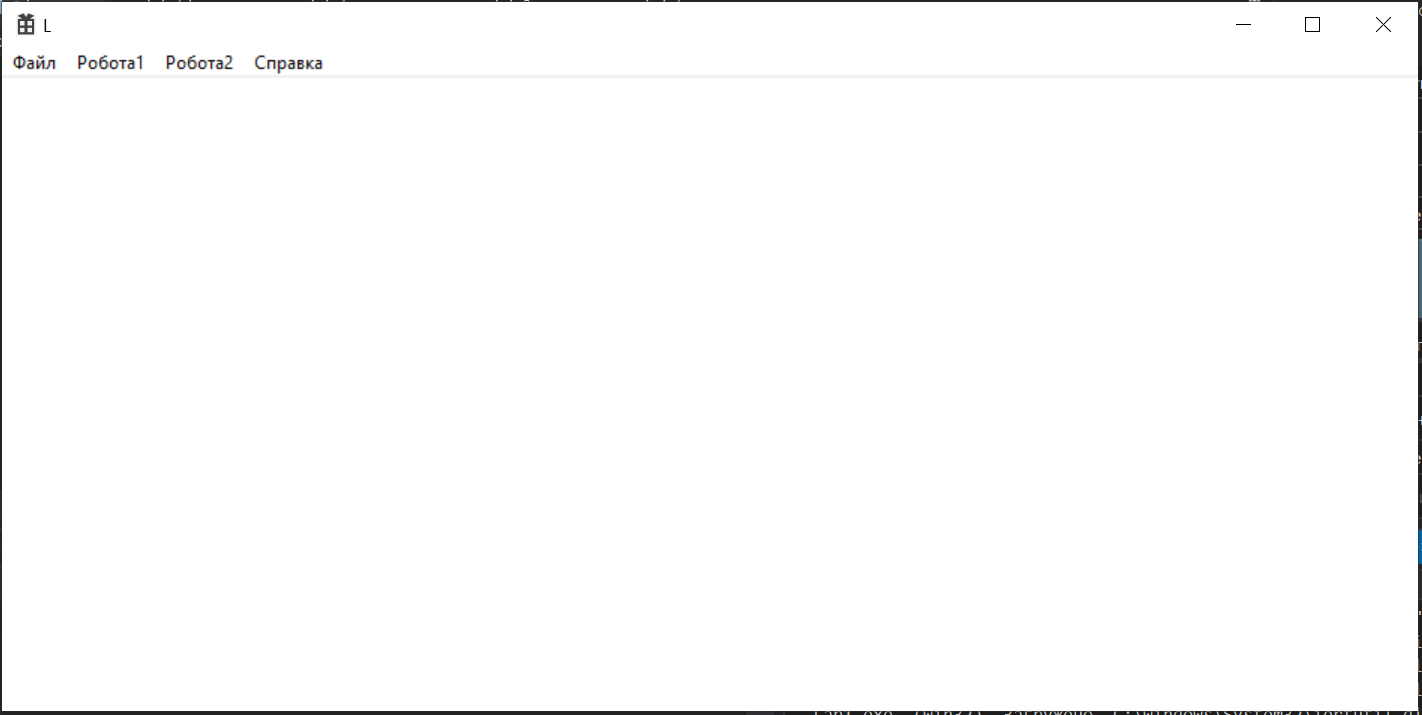
#define \_APS\_NEXT\_SYMED\_VALUE 101

#endif

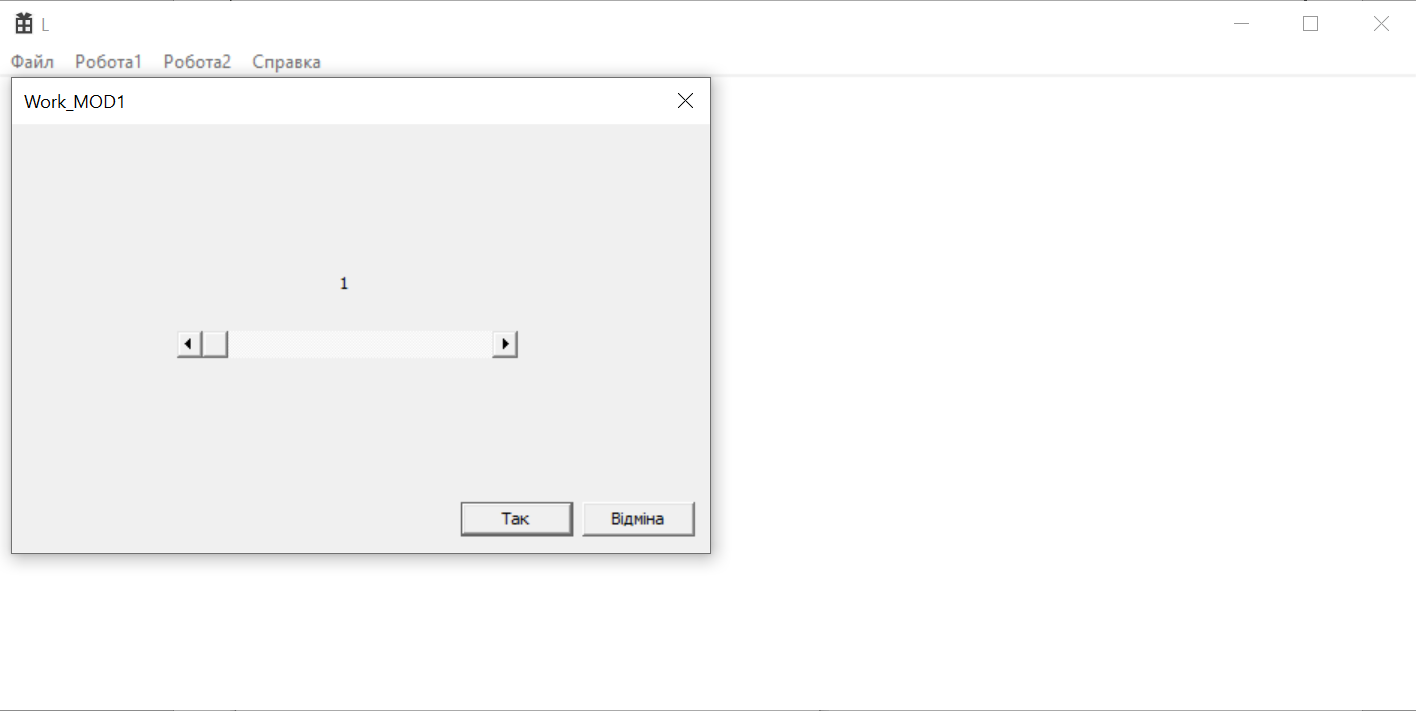
#endif

#endif

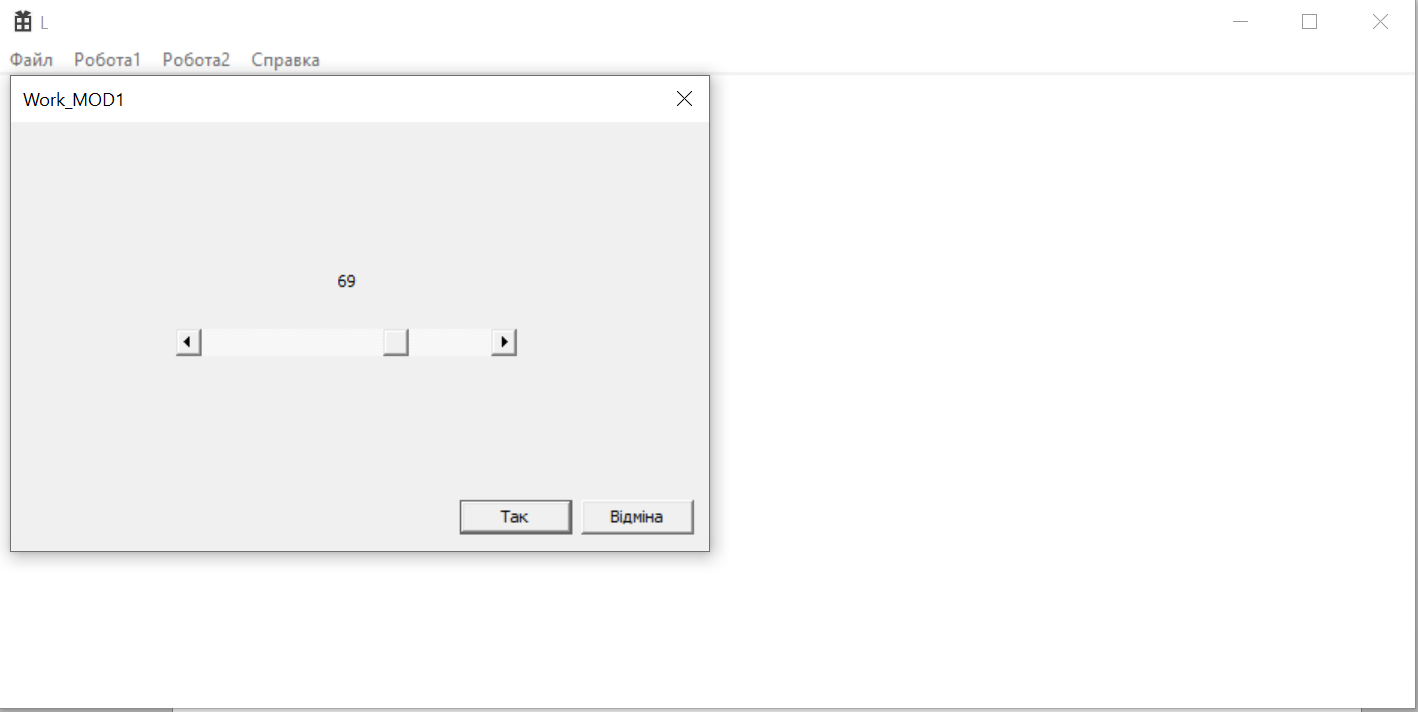
**Ілюстрації** **роботи:**

Головне вікно: 

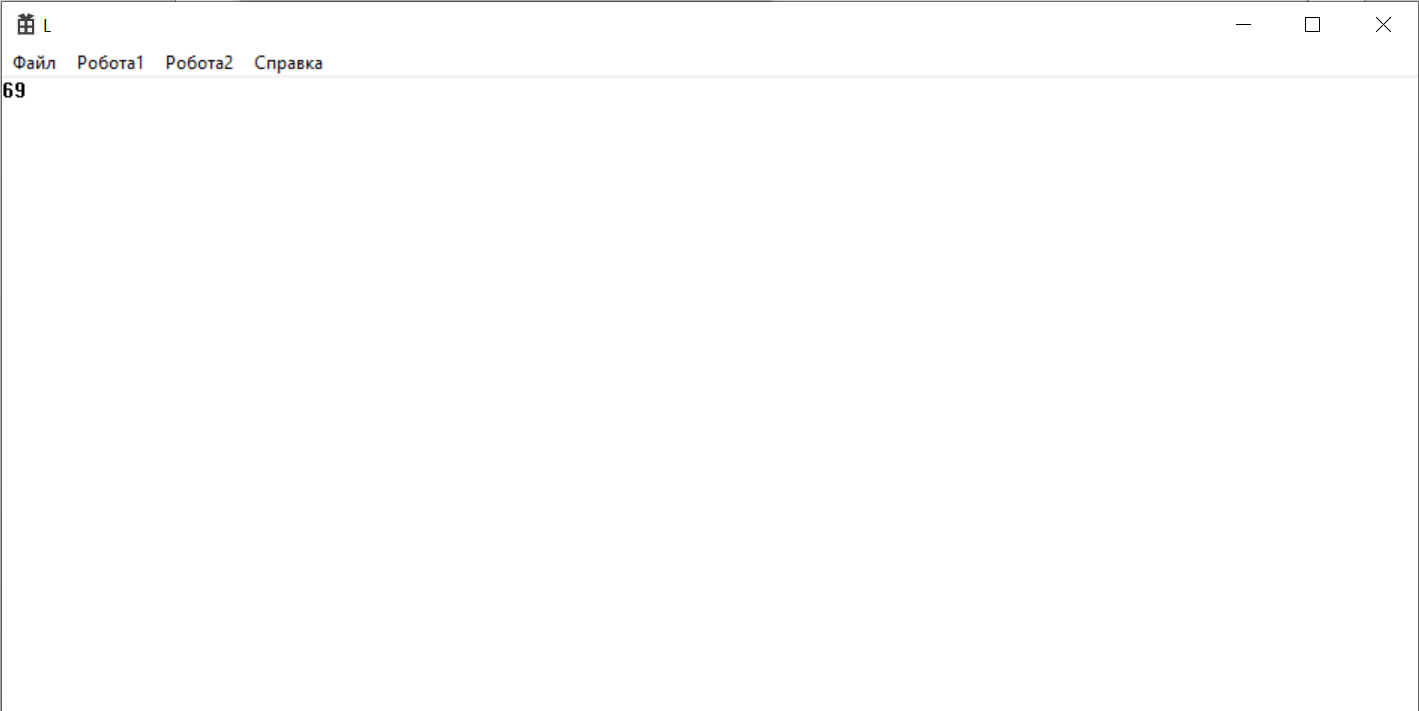
Перше завдання:



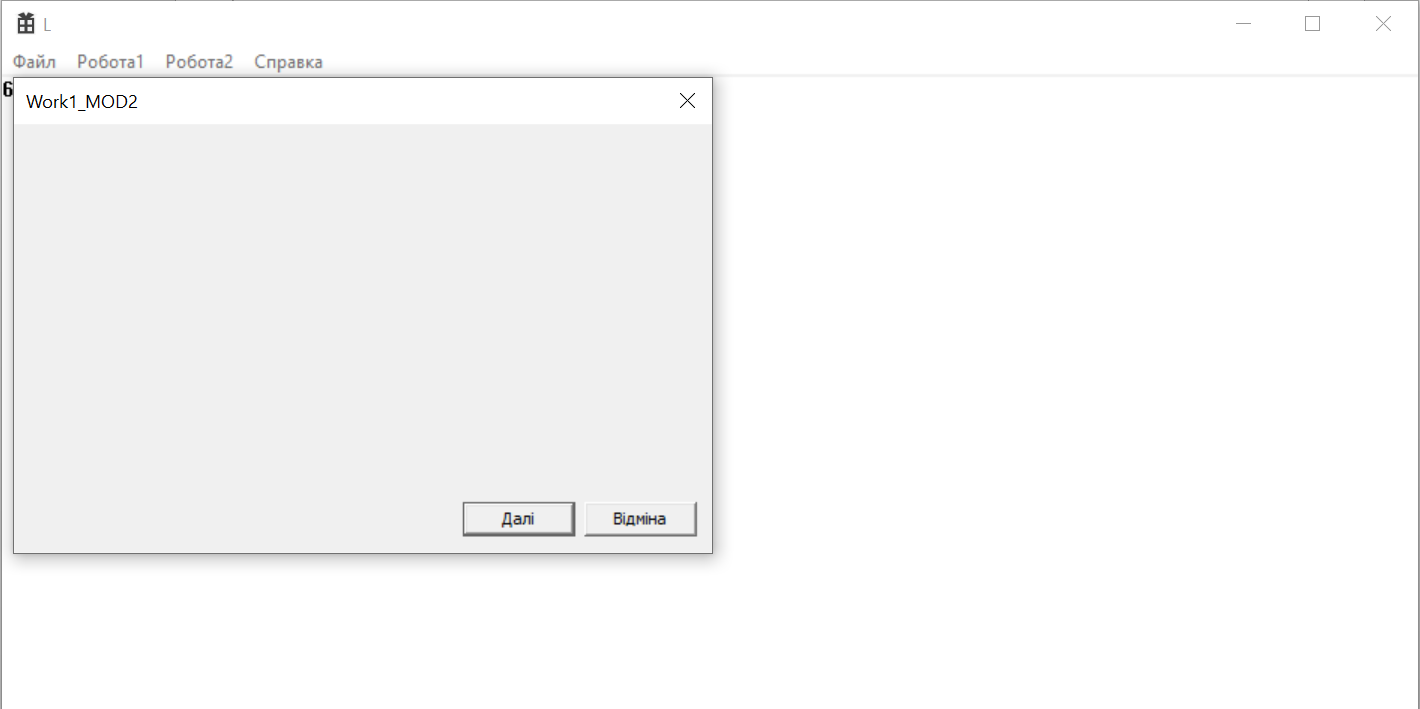
Зміна значення:



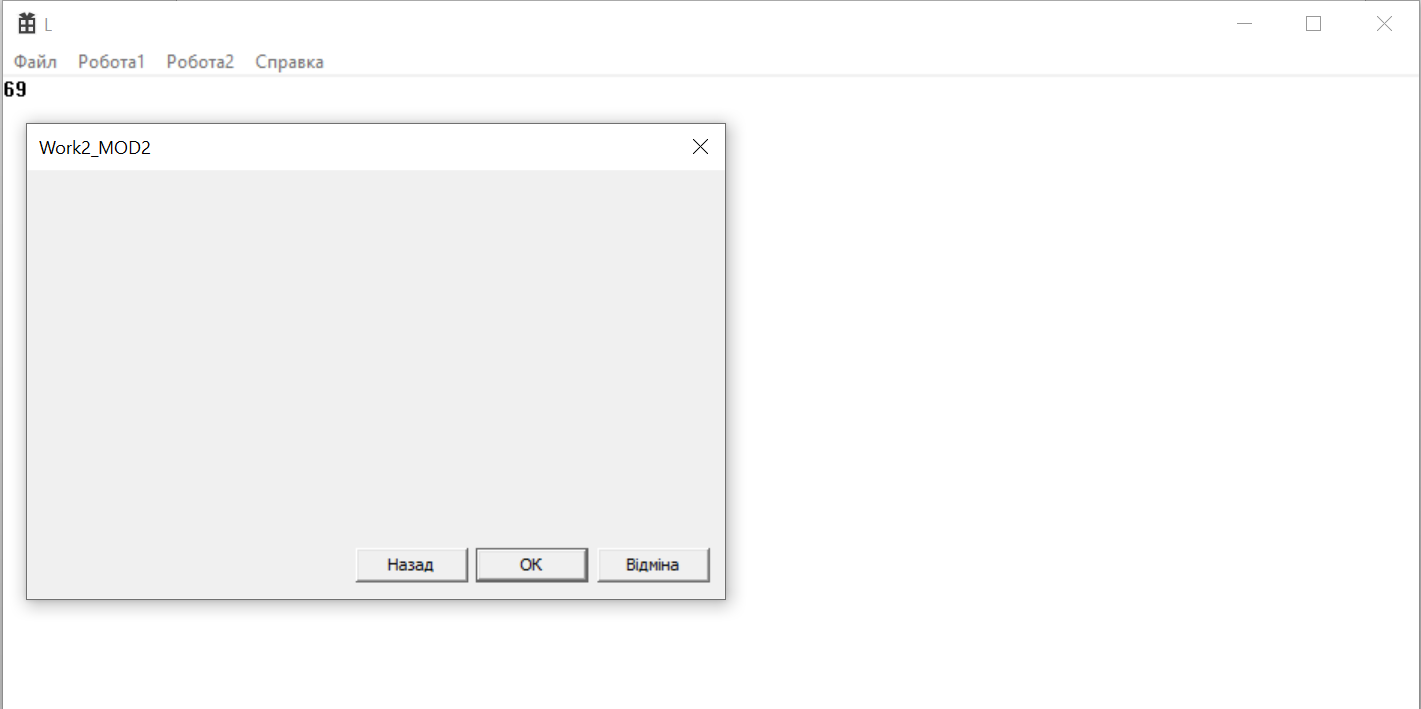
Виведення значення на екран:



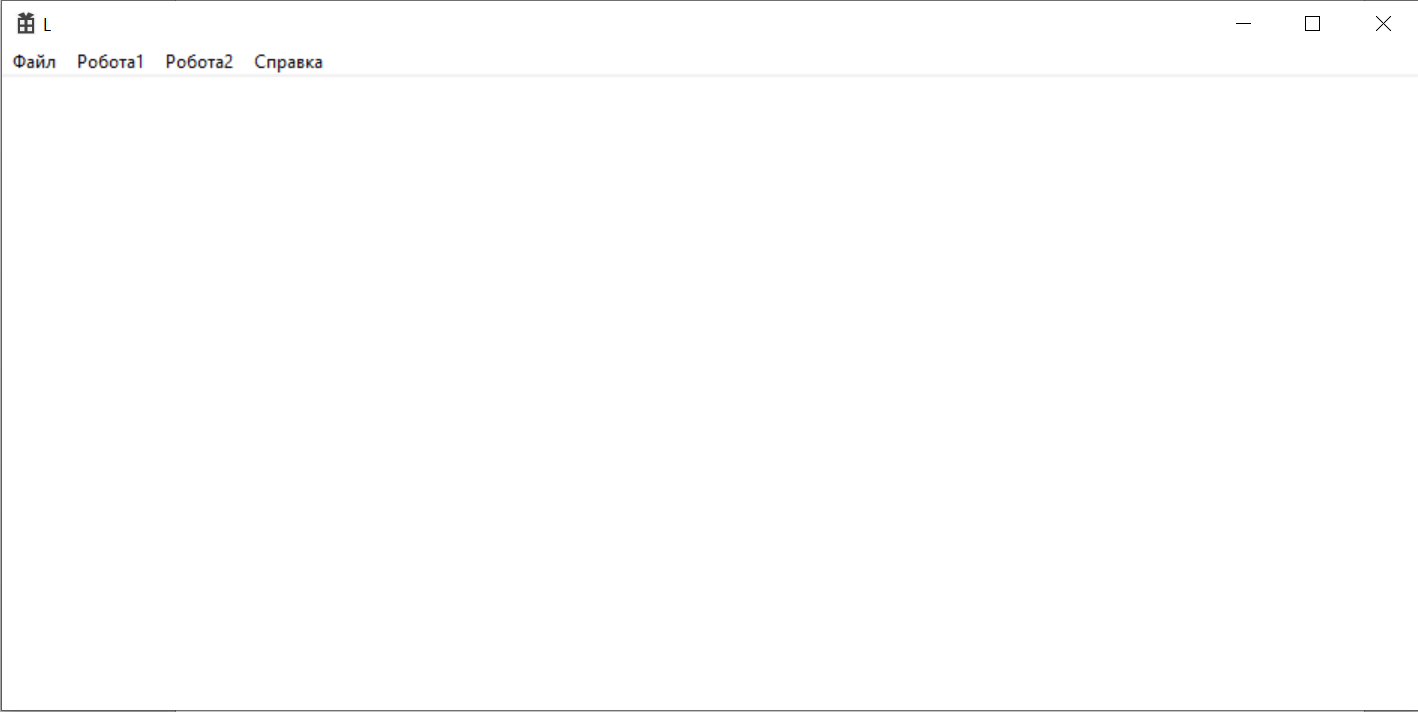
Друге завдання, перше вікно:



Друге завдання, друге вікно:



Натискання «ОК» або «Відміна»:



Також є анімація роботи (.gif), яка знаходиться в архіві.

**#include-ієрархія файлів:**

**Lab1.cpp**

#include "pch.h"

#include "framework.h"

#include "Lab1.h"

#include "module1.h"

#include "module2.h"

…

int APIENTRY wWinMain(…)

{

…

}

**Контрольні запитання:**

#include

#include

#include

Константи ідентифікаторів ресурсів module2

Константи ідентифікаторів ресурсів module1

Константи ідентифікаторів ресурсів module3

#include

#include "resource.h"

**Lab1.h**

#include

#include

**\*.h**

Інші заголовочні файли Windows API

#include

#include

#include

#include

#include

**Lab1.rc**

#include "resource.h"

…

#include "targetver.h"

…

#include <windows.h>

#include

**targetver.h**

**resource.h**

Константи ідентифікаторів ресурсів

Константи ідентифікаторів для опису версій Windows

**\*.h**

Заголовочні файли стандартних бібліотек C++

**framework.h**

Головний заголовочний файл Windows API

#include "targetver.h"

#define WIN32\_LEAN\_AND\_MEAN

#include <windows.h>

#include <stdlib.h>

#include <malloc.h>

#include <memory.h>

#include <tchar.h>

#include

**windows.h**

1. Що робить головна функція програми?

Головна функція – точка входу програми, початок роботи програми абсолютно повністю описується кодом цієї функції.Спочатку реєструється клас вікна, потім це вікно створюється. Далі програма входить у цикл, де буде очікувати на повідомлення. Подальша робота залежить 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 чітко вказуємо, де саме у коді повинно викликатися вікно, а при створенні - просто створюємо його.

3. Що робить у віконній функції оператор switch?

Дивиться на команди, що приходять у вікно та виконує певні дії

4. Як створюється новий пункт меню?

Меню – елемент графічного інтерфейсу користувача програми. Аби додати пункт меню, відкрити редактор ресурсів. Для цього двічі натискаємо у вікні Solution Explorer на файлі Lab1.rc. Після кліку на файлі Lab1.rc з'явиться редактор ресурсів. У списку ресурсів треба вибрати меню і двічі натиснути на нього. Тоді відкриється меню для редагування. Вводимо назву нового пункту меню, наприклад, "SomeWork1\_MOD1". За умовчанням редактор ресурсів присвоїть новому пункту меню ідентифікатор ID\_32771. Це погана назва ідентифікатора, отже краще його змінити.

5. Як викликається діалогове вікно?

За допомогою DialogBox();

6. Як створити нове діалогове вікно?

Спочатку у редакторі ресурсів додаємо новий Dialog у файл \*.rc. Після цього запрограмувати віконну функцію Нашого діалогового вікна. І в кінці запрограмувати виклик діалогового вікна у функції головного вікна, наприклад, MyWork1\_MOD1().

7. Де запрограмований цикл очікування повідомлень головного вікна?

Цикл очікування повідомлень головного вікна знаходиться у головній функції проекту (тобто у вході програми).

8. Що таке WM\_COMMAND?

Обробка меню програми.

9. Які файли потрібні для модулів проекту Win32?

Вихідні файли, де пишуться взаємодії з вікном (\*.cpp), файли ресурсів (\*.rc.) та файли заголовків (\*.h)

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?

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 компонувальник збирає отримані об'єктні модулі (з підключенням бібліотек) у виконуваний файл. Цей процес і називається роздільної компіляцією.

Її можна досягти за допомогою #include-ієрархії, яку описали вище. 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, зрозумів, як створювати вікна, як працювати з такими елементами, як editbox,buttons й подібні та на практиці задіяв роздільну компіляцію модулів.