**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

**Мета:**

Отримати 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 варіантів завдання.

3. Скомпілювати вихідний текст і отримати виконуваний файл програми.

4. Перевірити роботу 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"

#define MAX\_LOADSTRING 100

#pragma region Variables

…

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

ATOM MyRegisterClass**(**HINSTANCE hInstance**);**

BOOL InitInstance**(**HINSTANCE**,** int**);**

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

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

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

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

static void About1**(**HWND hWnd**);** // Declaration of our function

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

static int textHeightPosition **=** 0**;**

static int textWidthPosition **=** 0**;**

#pragma endregion

…

#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);

break;

case IDM\_WORK\_MOD2:

// second menu

Work2(hWnd);

break;

case IDM\_ABOUT:

// About Menu

About1(hWnd);

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);

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

// client area that must be redrawn.

InvalidateRect(hWnd, 0, TRUE);

}

/// <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);

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

// client area that must be redrawn.

InvalidateRect(hWnd, 0, TRUE);

}

/// <summary>

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

/// </summary>

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

void About1(HWND hWnd)

{

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

}

/// <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**(**HWND hDlg**);**

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

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

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

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

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

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

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

static int Count**(**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(hDlg);

break;

case WM\_HSCROLL:

GetPos(hDlg);

switch (LOWORD(wParam))

{

case SB\_LINELEFT:

OnLineLeft(hDlg);

break;

case SB\_LINERIGHT:

OnLineRight(hDlg);

break;

case SB\_THUMBPOSITION:

case SB\_THUMBTRACK:

OnThumbPosAndTrack(hDlg, wParam);

break;

default: break;

}

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

break;

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case IDOK:

OnOk(hDlg);

return (INT\_PTR)TRUE;

break;

case IDCANCEL:

OnCancel(hDlg);

return (INT\_PTR)TRUE;

break;

}

break;

case WM\_CLOSE:

{

OnClose(hDlg);

}

break;

default: break;

}

return FALSE;

}

/// <summary>

/// Called on initializing

/// </summary>

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

void OnInit(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(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(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(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 OnOk(HWND hDlg)

{

canWrite\_MOD1 = TRUE;

numOfDig\_MOD1 = Count(pos\_MOD1);

EndDialog(hDlg, 1);

}

/// <summary>

/// Called when IDCANCEL clicked

/// </summary>

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

void OnCancel(HWND hDlg)

{

EndDialog(hDlg, 0);

}

/// <summary>

/// Called when window is closing

/// </summary>

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

void OnClose(HWND hDlg)

{

EndDialog(hDlg, 0);

}

/// <summary>

/// Get pos of scroll

/// </summary>

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

void GetPos(HWND hDlg)

{

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

}

/// <summary>

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

/// </summary>

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

/// <returns></returns>

int Count(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**

#pragma once

#include "resource1.h"

#pragma region Function

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

#pragma endregion

**module2.cpp**

#include "pch.h"

#include "framework.h"

#include "module3.h"

#include "module2.h"

#include "Resource.h"

#pragma region FunctionsDeclaration

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

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

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

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

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case IDC\_CANCEL1\_MOD2:

OnCancel(hDlg);

return (INT\_PTR)TRUE;

case IDC\_NEXT\_MOD2:

OnNext(hDlg);

return (INT\_PTR)TRUE;

}

break;

case WM\_CLOSE:

{

OnClose(hDlg);

}

break;

}

return (INT\_PTR)FALSE;

}

/// <summary>

/// Called when IDC\_NEXT\_MOD2 clicked

/// </summary>

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

void OnNext(HWND hDlg)

{

EndDialog(hDlg, 1);

Func\_MOD3(hInst, hDlg);

}

/// <summary>

/// Called when IDC\_CANCEL1\_MOD2 clicked

/// </summary>

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

void OnCancel(HWND hDlg)

{

EndDialog(hDlg, 0);

}

/// <summary>

/// Called when window is closing

/// </summary>

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

void OnClose(HWND hDlg)

{

EndDialog(hDlg, 0);

}

#pragma endregion

**module2.h**

#pragma once

#include "resource2.h"

#pragma region Function

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

#pragma endregion

**module3.cpp**

#include "pch.h"

#include "framework.h"

#include "module3.h"

#include "module2.h"

#include "Resource.h"

#pragma region FunctionsDeclaration

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

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

static void OnCancel**(**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\_MOD3**(**HINSTANCE hInst**,** HWND hWnd**)**

**{**

**return** DialogBox**(**hInst**,** MAKEINTRESOURCE**(**IDD\_WORK2\_MOD2**),** hWnd**,** 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

OnOk(hDlg);

return (INT\_PTR)TRUE;

case IDC\_CANCEL2\_MOD2: // Cancel button

OnCancel(hDlg, message, wParam, lParam);

return (INT\_PTR)TRUE;

case IDC\_BACK\_MOD2: // Back button

OnBack(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 OnBack(HWND hDlg)

{

EndDialog(hDlg, 1);

Func\_MOD2(hInst, hDlg);

}

/// <summary>

/// Called when IDOK clicked

/// </summary>

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

void OnOk(HWND hDlg)

{

EndDialog(hDlg, 1);

}

/// <summary>

/// Called when IDCANCEL clicked

/// </summary>

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

void OnCancel(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

EndDialog(hDlg, 0);

DefWindowProc(hDlg, message, wParam, lParam);

}

#pragma endregion

**module3.h**

#pragma once

#include "resource3.h"

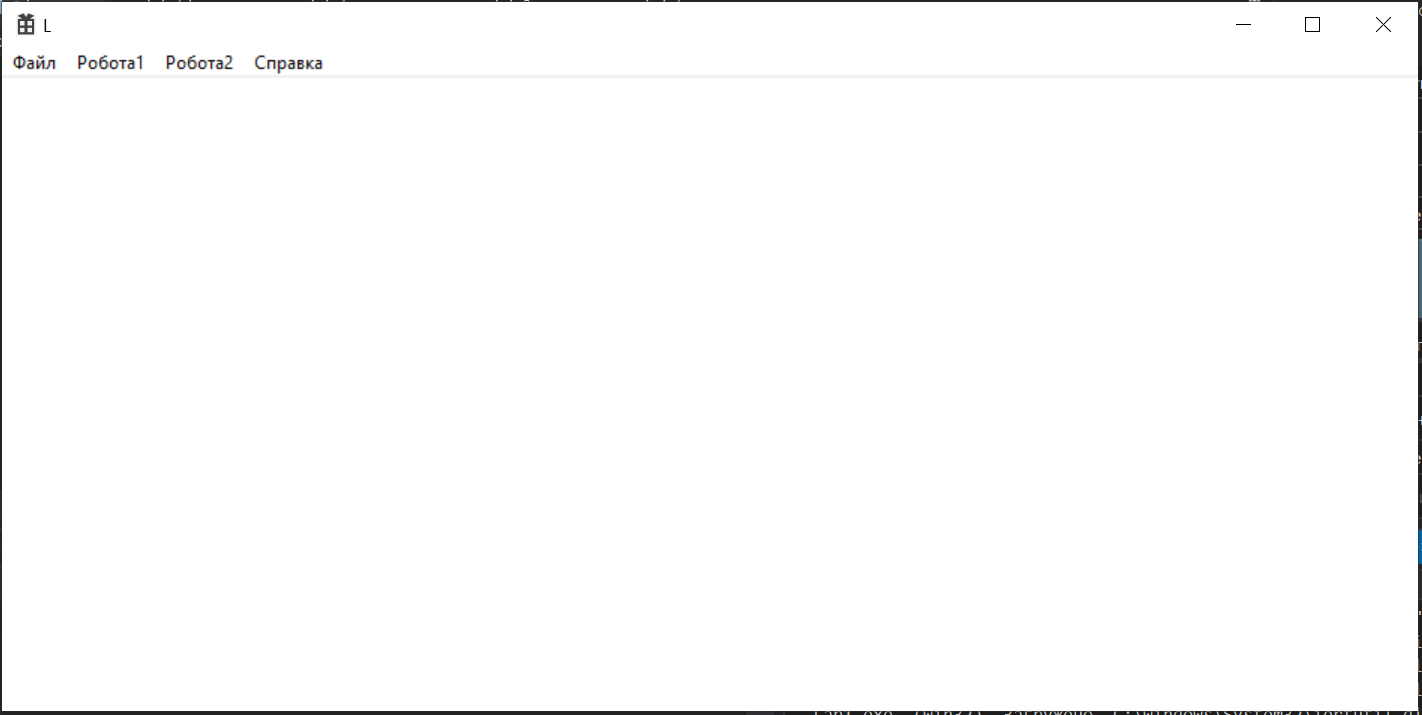
#pragma region Function

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

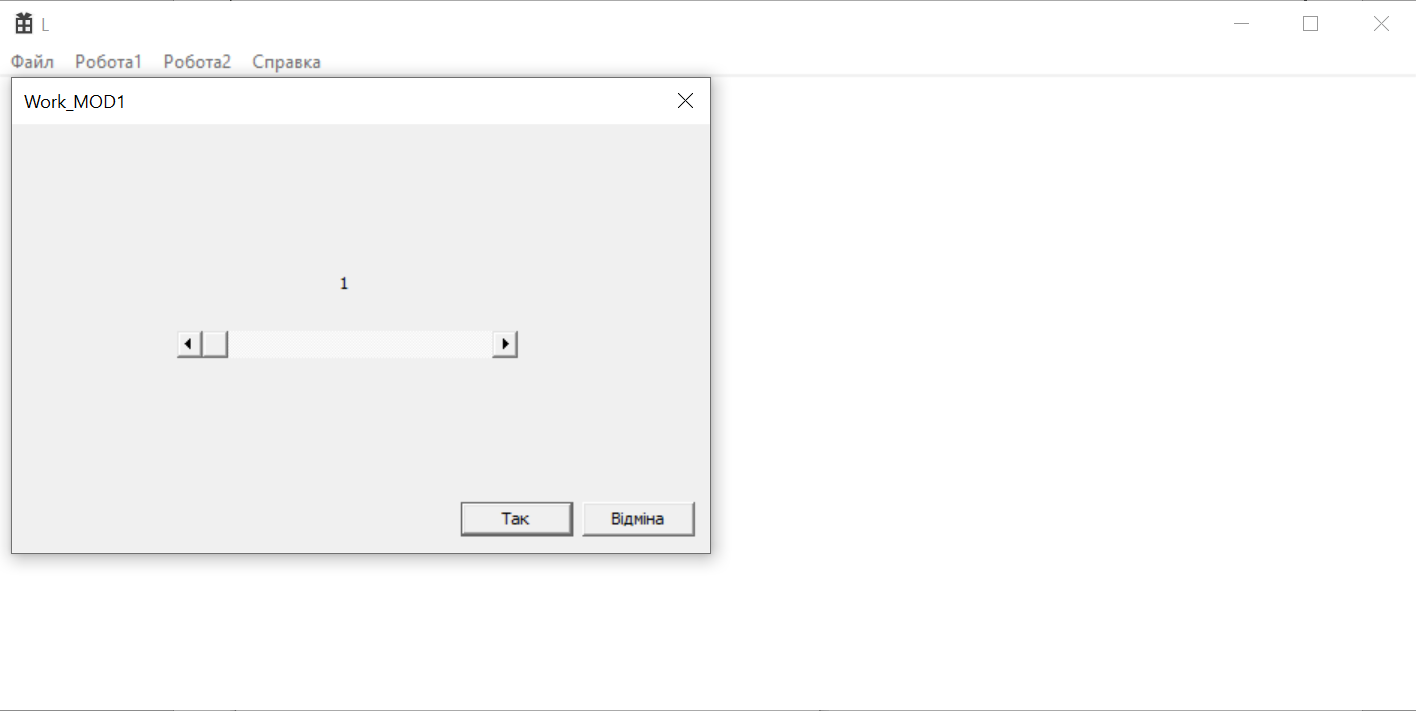
#pragma endregion

**Ілюстрації** **роботи:**

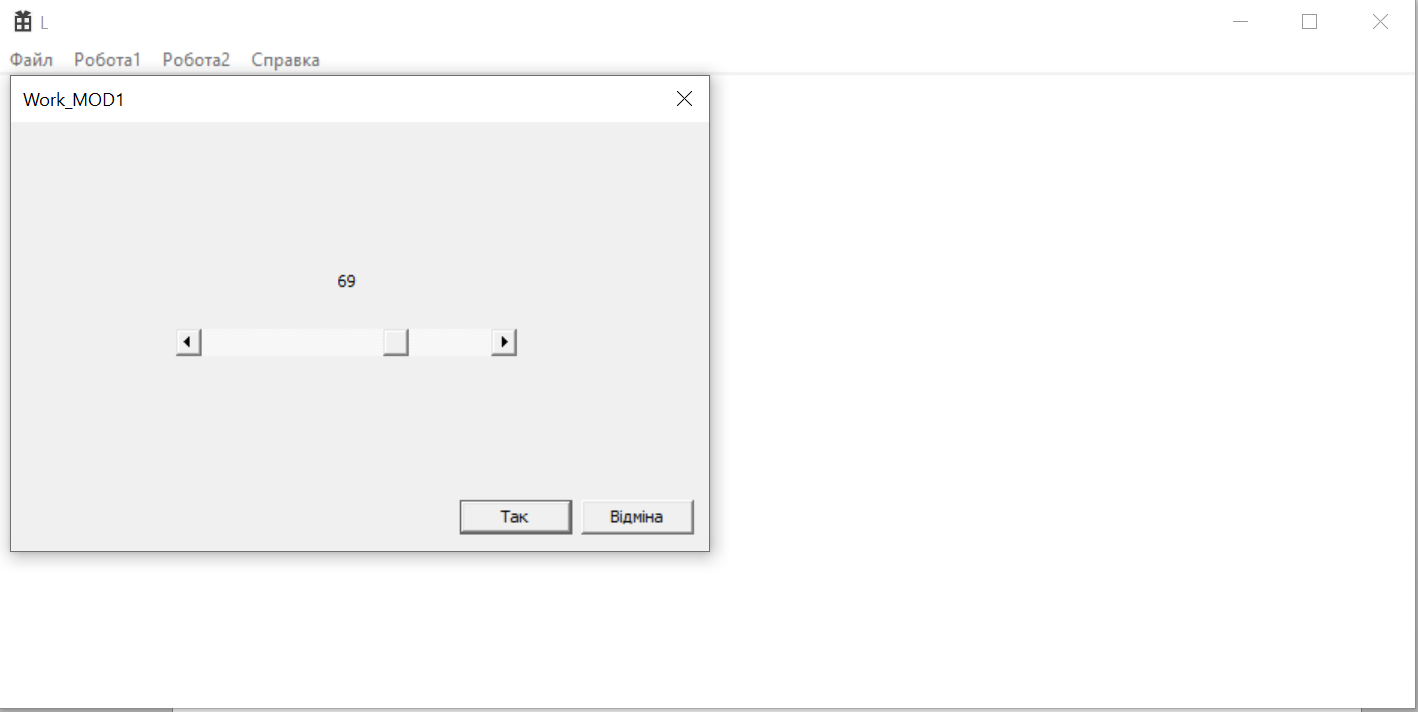
Головне вікно:



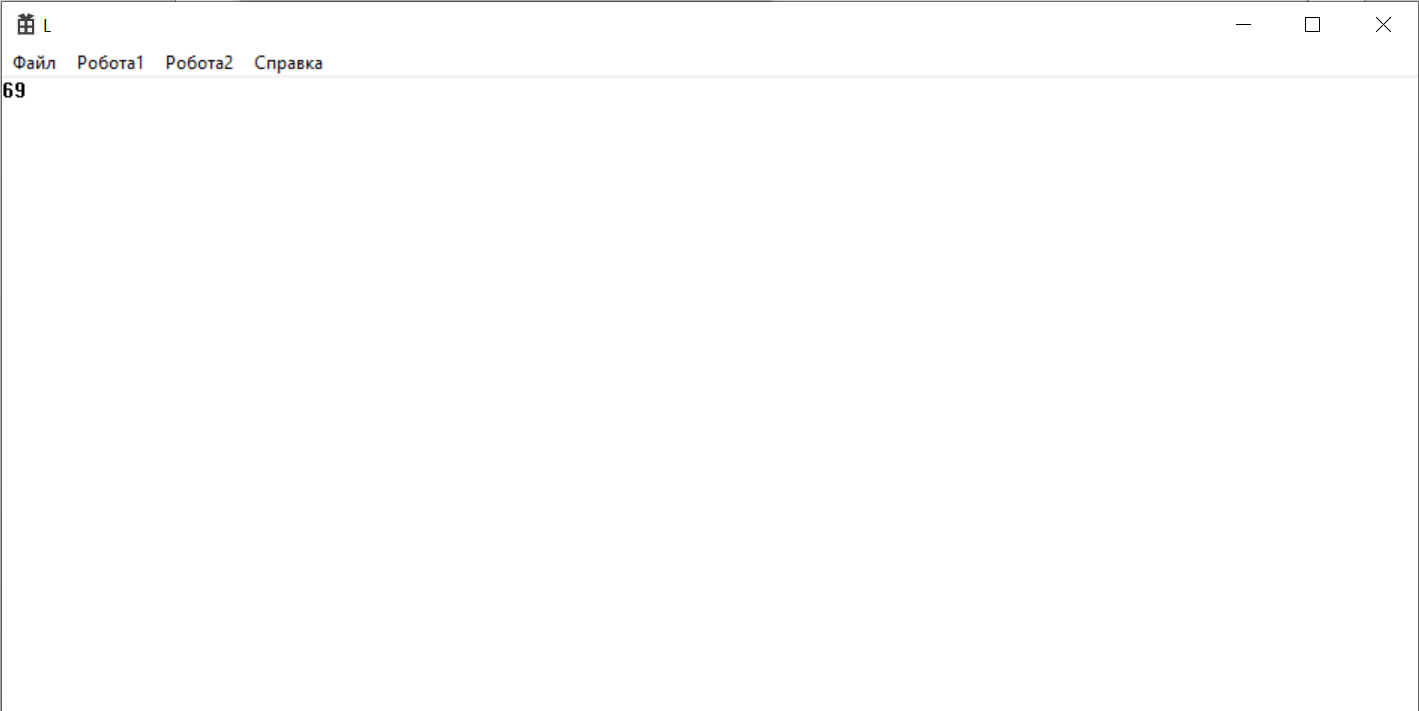
Перше завдання:



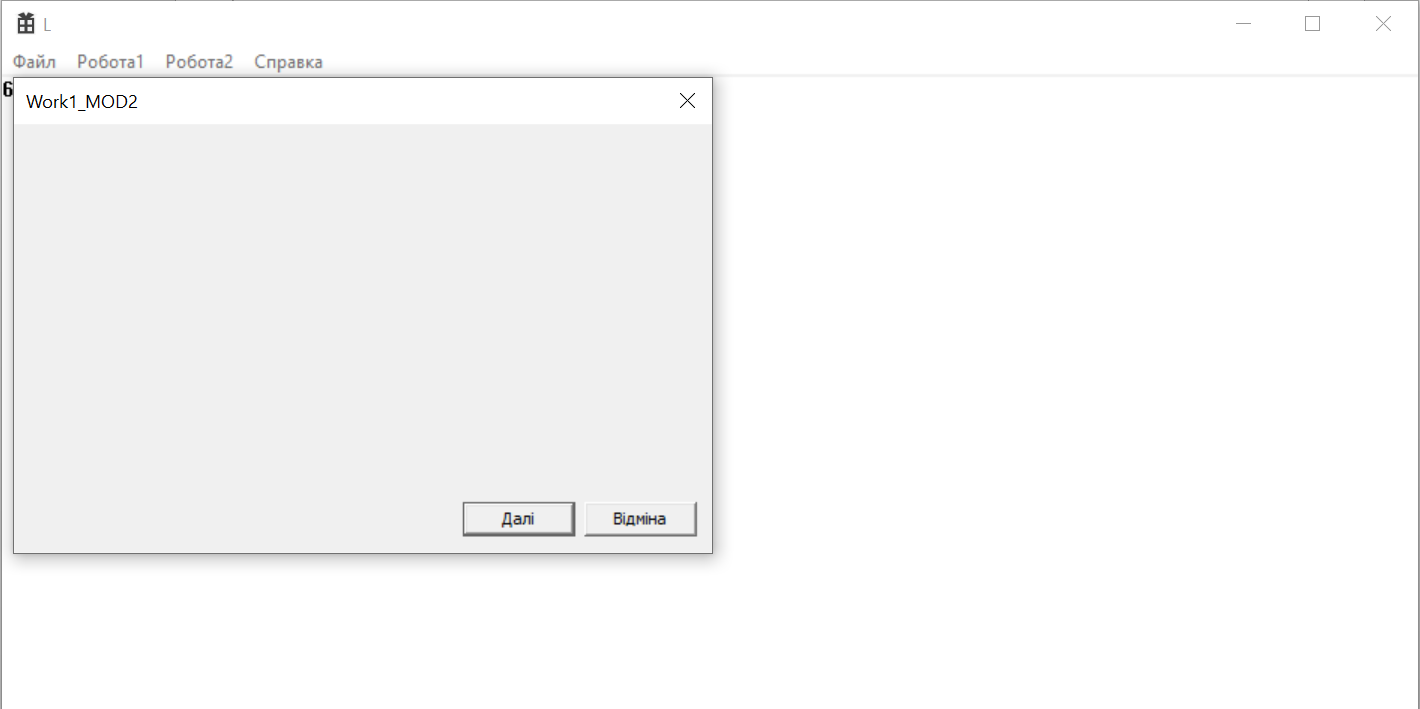
Зміна значення:



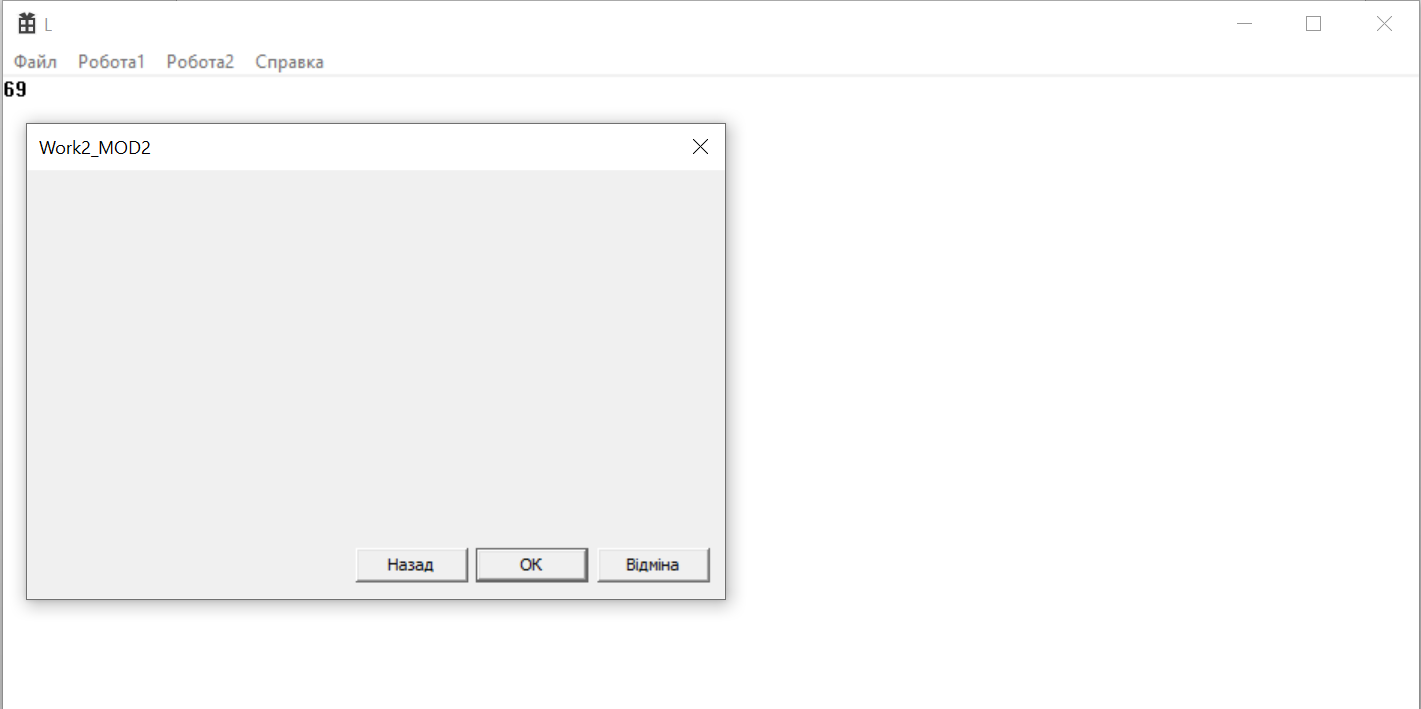
Виведення значення на екран:



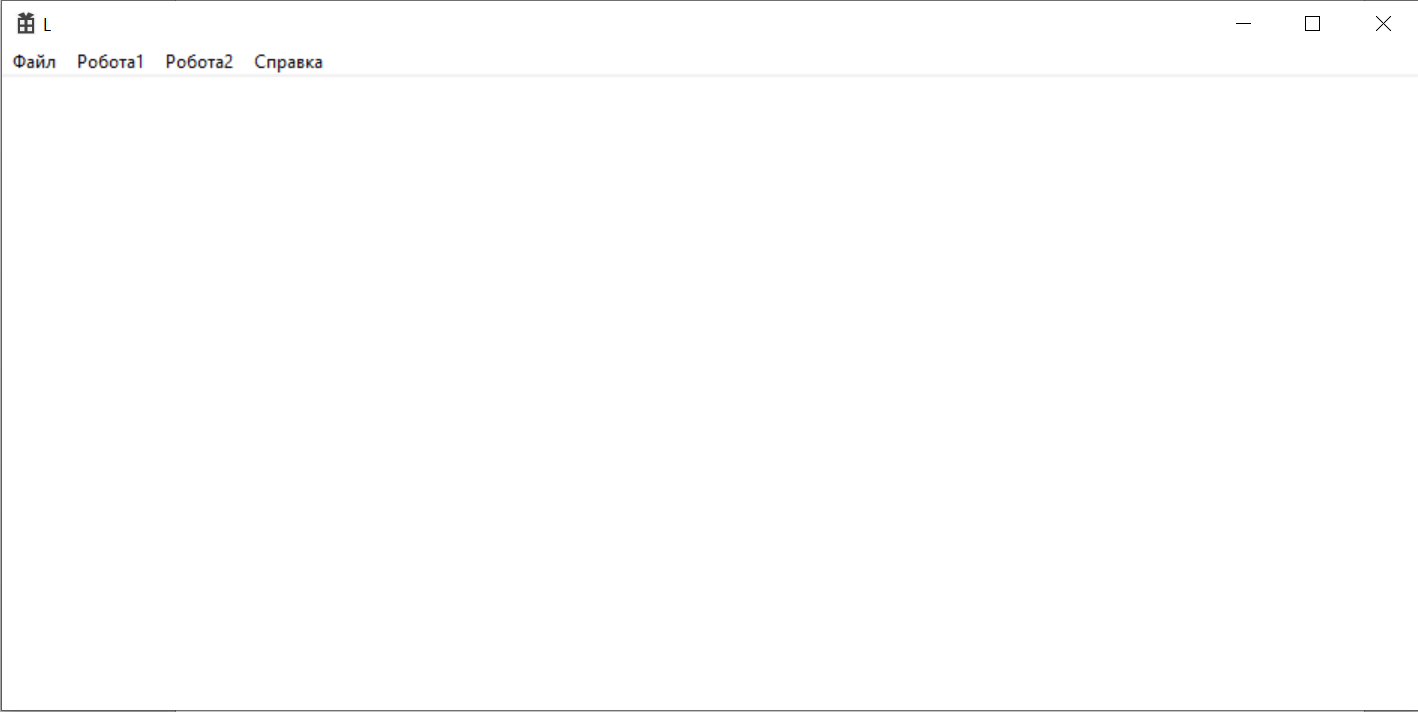
Друге завдання, перше вікно:



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?**

Це 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?**

При реєстрації Ми створюємо щось на кшталт шаблону вікна, а на етапі створення-створюємо вікно

**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?**

Треба відкрити редактор ресурсів. У списку ресурсів треба вибрати меню і двічі натиснути на нього. Відкривається меню для редагування. Треба ввести назву меню та змінити його ID

**5) Як викликається діалогове вікно?**

За допомогою DialogBox();

**6) Як створити нове діалогове вікно?**

Треба додати новий Dialog у файл .rc (через редактор ресурсів). Потім написати код для функції вікна. І нарешті написати код для виклику діалогового вікна у функції вікна.

**7) Де запрограмований цикл очікування повідомлень головного вікна?**

Він знаходиться у головній функції проекту (тобто у вході програми) (як було описано в пункті 1).

**8) Що таке WM\_COMMAND?**

Обробка меню програми.

**9) Які файли потрібні для модулів проекту Win32?**

Файли, де пишуться дії над вікном (.cpp), файли ресурсів (.rc) та файли заголовків (.h) разом з файлами (.rh)

**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 включає інший вихідний файл

**12) Чим відрізняється віконна функція головного вікна від діалогового?**

Кожна виконує функції лише для певних вікон.

**13) Що означає роздільна компіляція модулів і як її досягти?**

Спочатку усі модулі компілюються окремо, а потім збираються в один .ехе файл.

Це можна досягти використовуючи #include-ієрархію

**Висновок:**

Під час роботи Я познайомився з Win32, створив перший проект, навчився створювати вікна, static text , horizontal scroll bar і т.д. та почав працювати з роздільною компіляцією модулів.