МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп'ютерних систем та інфокомунікацій

Кафедра комп'ютерних систем, мереж і кібербезпеки

Лабораторна робота

з Системного програмування
(назва дисципліни)

на тему: «Створення віконних додатків за допомогою win32 API»

Виконала: студентка 3-го курсу групи №525ст2
напряму підготовки (спеціальності)
123-«Комп'ютерна інженерія»
(шифр і назва напряму підготовки (спеціальності))
Коваленко Я.О.
(прізвище й ініціали студента)

Прийняв: асистент каф.503
Мозговий М.В.
(посада, науковий ступінь, прізвище й ініціали)

Національна шкала: _____
Кількість балів: _____
Оцінка: ECTS _____

Харків – 2020

Цель работы:

Создание оконных приложений с помощью win32 API.

Постановка задачи:

Требуется разработать оконное приложение с SDI интерфейсом (еще бывает MDI, Dialog Based), которое будет реализовывать функцию поиска файла по имени в заданном каталоге и подкаталогах. Поиск должен выполняться в отдельном потоке, чтобы приложение продолжало отвечать на действия пользователя. Плюс должна идти визуализация процесса поиска: или в виде бегущего прогресс бара или в виде вывода имени с полным путем проверяемых каталогов. Так же можно реализовать приостановку процесса поиска путем перевода потока по запросу пользователя в приостановленное состояние.

Ход работы:

Код программы:

```
#ifndef UNICODE
#define UNICODE
#endif

#include "stdafx.h"
#include <windows.h>
#include <shobjidl.h>
#include "atlstr.h"

#define START_BUTTON 3001
#define FOLDER_BUTTON 3002

#define ID_EDIT 51
#define ID_FOLDER_EDIT 52

#define ID_RICH_TEXT 1001
#define ID_SEARH_TEXT 1002

#define WM_MY_DATA WM_USER+1

LPWSTR **shared_arr;
static int shared_array_index;
HANDLE ghMutex;

typedef struct Data {
HWND WindowHwnd;
LPWSTR file_name;
WCHAR directory_path[MAX_PATH];

} MYDATA, *PMYDATA;
```

```
	LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam);
	void FileDiag();
	void FindFile(PMYDATA pData);

	DWORD WINAPI MyThreadFunction(LPVOID lpParam)
	{
	PMYDATA pData = (PMYDATA)lpParam;
	WIN32_FIND_DATA found_file_data;
	WCHAR directory[MAX_PATH];

	wcsncpy_s(directory, pData->directory_path, MAX_PATH);
	wcsncat_s(directory, L"\\*", 3);

	HANDLE hFind = INVALID_HANDLE_VALUE;

	hFind = FindFirstFile(directory, &found_file_data);

	if (INVALID_HANDLE_VALUE != hFind)
	{
		do
		{
			if (found_file_data.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
			{
				if (wcscmp(found_file_data.cFileName, L".") != 0 && wcscmp(found_file_data.cFileName, L"..") != 0)
				{
					PMYDATA pDataCopy = (PMYDATA)HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, sizeof(MYDATA));

					pDataCopy->file_name = pData->file_name;
					pDataCopy->WindowHwnd = pData->WindowHwnd;
					wcsncpy_s(pDataCopy->directory_path, pData->directory_path, MAX_PATH);
					wcsncat_s(pDataCopy->directory_path, L"\\", 2);
					wcsncat_s(pDataCopy->directory_path, found_file_data.cFileName, MAX_PATH);

					FindFile(pDataCopy);
				}

			}

			DWORD dwWaitResult = WaitForSingleObject(ghMutex, INFINITE);
			switch (dwWaitResult)
			{
			case WAIT_OBJECT_0:
			{
				__try {
					shared_arr[shared_array_index] = new LPWSTR[MAX_PATH];
					shared_arr[shared_array_index][0] = new WCHAR[260];
```

```
                                        if (StrStrIW(found_file_data.cFileName, pData->file_name) != NULL) {
                                                wcsncat_s(found_file_data.cFileName, L"?", 1);
                                        }

                                        if (wcscmp(found_file_data.cFileName, L".") != 0 && wcscmp(found_file_data.cFileName, L"..") != 0) {
                                                WCHAR buf[MAX_PATH];
                                                wcsncpy_s(buf, pData->directory_path, MAX_PATH);
                                                wcsncat_s(buf, L"\\", 2);
                                                wcsncat_s(buf, found_file_data.cFileName, MAX_PATH);
                                                swprintf_s(shared_arr[shared_array_index][0], 260, buf);
                                                int index = shared_array_index;
                                                PostMessage(pData->WindowHwnd, WM_MY_DATA, 0, (LPARAM)index);
                                                Sleep(400);
                                                shared_array_index++;
                                        }
                                }
                                __finally {
                                        if (!ReleaseMutex(ghMutex))
                                        {
                                                MessageBox(NULL, L"ReleaseMutex failed", L"Error", MB_OK);
                                        }
                                }
                                        break;
                        }
                        case WAIT_ABANDONED:
                                return FALSE;
                        }
                } while (FindNextFile(hFind, &found_file_data) != 0);
                FindClose(hFind);
        }
        else {
                MessageBox(NULL, L"FindFirstFile failed", L"Error", MB_OK);
                return 0;
        }
        return 0;
        }
        void FindFile(PMYDATA pData) {

        DWORD IDThread;

        CreateThread(NULL,
                0,
                (LPTHREAD_START_ROUTINE)MyThreadFunction,
                pData,
                0,
                &IDThread);
```

```
}
//program entry point
int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE, PWSTR pCmdLine, int nCmdShow)
{
const wchar_t CLASS_NAME[] = L"Window Class";

WNDCLASS wc = { };

wc.lpfnWndProc = WindowProc;
wc.hInstance = hInstance;
wc.lpszClassName = CLASS_NAME;

RegisterClass(&wc);

HWND hwnd = CreateWindowEx(
	0,
	CLASS_NAME,
	L"Search Files",
	WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU | WS_MINIMIZEBOX | WS_MAXIMIZEBOX,
	CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,
	NULL,
	NULL,
	hInstance,
	NULL
);

if (hwnd == NULL)
{
	return 0;
}
SetWindowPos(hwnd, HWND_TOP, GetSystemMetrics(SM_CXSCREEN) / 2,
	GetSystemMetrics(SM_CYSCREEN) / 2, 600, 360, SWP_SHOWWINDOW);

MSG message = { };
while (GetMessage(&message, NULL, 0, 0) != 0)
{
	TranslateMessage(&message);
	DispatchMessage(&message);
}

return 0;
}
void appendTextToEdit(HWND hEdit, LPCWSTR newText)
{
int TextLen = SendMessage(hEdit, WM_GETTEXTLENGTH, 0, 0);
SendMessage(hEdit, EM_SETSEL, (WPARAM)TextLen, (LPARAM)TextLen);
SendMessage(hEdit, EM_REPLACESEL, FALSE, (LPARAM)newText);
}
```

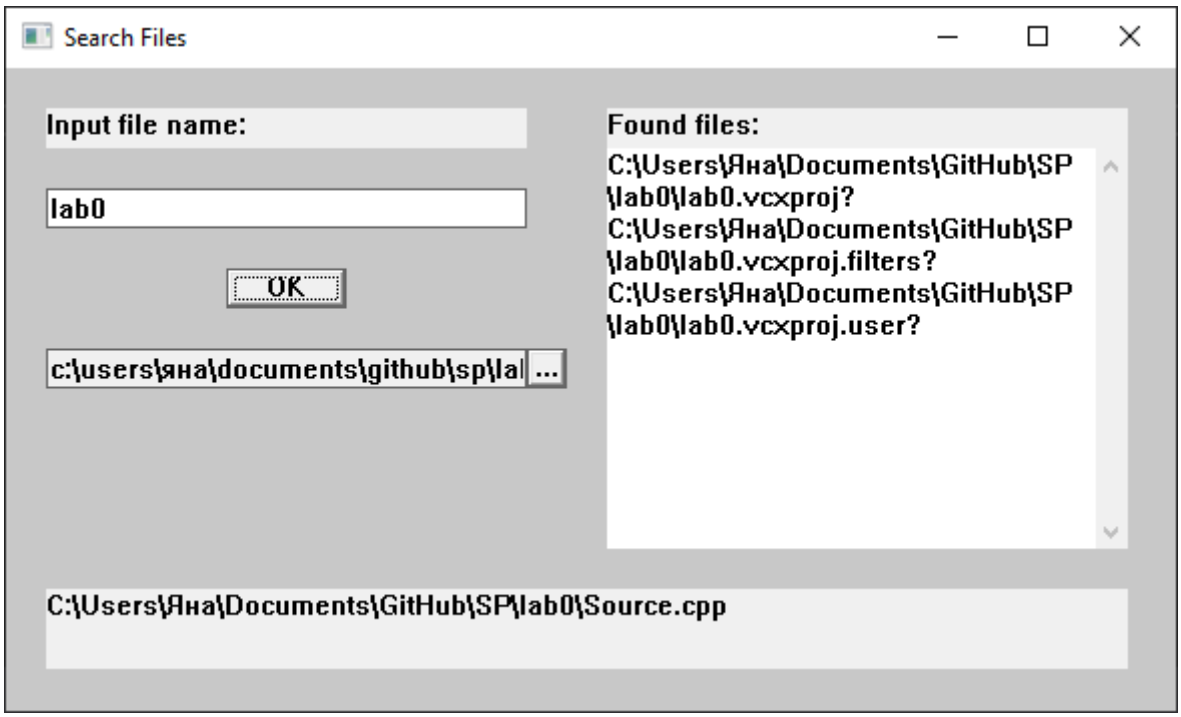
```
	LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
	{
	static HWND hwndEdit;
	static HWND hwndTextBottom;
	static HWND hwndFolder;
	static HWND hwndRichTextBox;
	static HWND hwndButton;
	switch (uMsg)
	{
	case WM_DESTROY: {
		PostQuitMessage(0);
		delete[] shared_arr;
		return 0;
	}

	case WM_CREATE: {

		HWND hwndTextTop = CreateWindowW(L"Static", L"Input file name:",
			WS_CHILD | WS_VISIBLE | SS_LEFT,
			20, 20, 240, 20,
			hwnd, (HMENU)1, NULL, NULL);

		HWND hwndTextTop1 = CreateWindowW(L"Static", L"Found files:",
			WS_CHILD | WS_VISIBLE | SS_LEFT,
			300, 20, 260, 220,
			hwnd, (HMENU)1, NULL, NULL);

		hwndEdit = CreateWindowW(L"Edit", NULL,
			WS_CHILD | WS_VISIBLE | WS_BORDER | ES_AUTOHSCROLL | ES_LEFT | ES_LOWERCASE,
			20, 60, 240, 20, hwnd, (HMENU)ID_EDIT,
			NULL, NULL);

		hwndButton = CreateWindow(
			L"BUTTON",
			L"OK",      // Button text
			WS_TABSTOP | WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON | BS_NOTIFY,  // Styles
			110,        // x position
			100,        // y position
			60,       // Button width
			20,       // Button height
			hwnd,     // Parent window
			(HMENU)START_BUTTON,
			(HINSTANCE)GetWindowLongPtr(hwnd, GWLP_HINSTANCE),
			NULL);

		hwndFolder = CreateWindowW(L"Edit", NULL,
			WS_CHILD | WS_VISIBLE | WS_BORDER | ES_AUTOHSCROLL | ES_LEFT | ES_LOWERCASE | ES_READONLY,
			20, 140, 240, 20, hwnd, (HMENU)ID_FOLDER_EDIT,
```

```
                NULL, NULL);

            HWND hwndButtonFolder = CreateWindow(
                L"BUTTON",
                L"...",
                WS_TABSTOP | WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON |
BS_NOTIFY,  // Styles
                260,
                140,
                20,
                20,
                hwnd,
                (HMENU)FOLDER_BUTTON,
                (HINSTANCE)GetWindowLongPtr(hwnd, GWLP_HINSTANCE),
                NULL);

            hwndTextBottom = CreateWindowW(L"Static", L"",
                WS_CHILD | WS_VISIBLE | SS_LEFT,
                20, 260, 540, 40,
                hwnd, (HMENU)ID_SEARH_TEXT, NULL, NULL);

            hwndRichTextBox =
                CreateWindowW(L"Edit", NULL,
                    WS_CHILD | WS_VISIBLE | WS_VSCROLL |
                    ES_LEFT | ES_MULTILINE | ES_AUTOVSCROLL,
                    300, 40, 260, 200, hwnd, (HMENU)ID_RICH_TEXT,
                    NULL, NULL);
        }
        case WM_PAINT:
        {
            PAINTSTRUCT ps;
            HDC hdc = BeginPaint(hwnd, &ps);

            FillRect(hdc, &ps.rcPaint, (HBRUSH)(1));

            EndPaint(hwnd, &ps);
        }
        case WM_COMMAND: {
            switch (wParam)
            {
            case START_BUTTON: {

                int text_length = GetWindowTextLengthW(hwndEdit) + 1;

                LPWSTR file_name = new WCHAR[text_length];
                GetWindowTextW(hwndEdit, file_name, text_length);

                SetDlgItemText(hwnd, ID_SEARH_TEXT, L"");
                SetWindowText(hwndRichTextBox, L"");
                PMYDATA pData = (PMYDATA)HeapAlloc(GetProcessHeap(),
                    HEAP_ZERO_MEMORY, sizeof(MYDATA));
                pData->WindowHwnd = hwnd;
```

```
                pData->file_name = file_name;
                GetCurrentDirectory(sizeof(pData->directory_path), pData->directory_path);

                shared_array_index = 0;
                shared_arr = new LPWSTR*[SHRT_MAX];
                ghMutex = CreateMutex(NULL, FALSE, NULL);

                FindFile(pData);

                return 0;
            }
            case FOLDER_BUTTON: {
                /*MSDN:For Windows Vista or later, it is
                recommended that you use IFileDialog with the
                FOS_PICKFOLDERS option rather than the SHBrowseForFolder function.*/
                FileDiag();

                WCHAR dir[MAX_PATH];

                GetCurrentDirectory(sizeof(dir), dir);
                SetWindowText(hwndFolder, dir);
                return 0;
            }
                                                return 1;
            }
            break;
        }
        case WM_MY_DATA: {

            int found_file_index = (int)lParam;

            SetDlgItemText(hwnd, ID_SEARH_TEXT, shared_arr[found_file_index][0]);

            if (StrStrIW(shared_arr[found_file_index][0], L"?") != NULL) {
                int outLength = GetWindowTextLength(hwndRichTextBox) +
lstrlen(shared_arr[found_file_index][0]) + 4;
                WCHAR * buf = (WCHAR *)GlobalAlloc(GPTR, outLength *
sizeof(WCHAR));

                _tcscat_s(buf, outLength, shared_arr[found_file_index][0]);
                _tcscat_s(buf, outLength, L"\r\n");
                appendTextToEdit(hwndRichTextBox, buf);

                GlobalFree(buf);
            }


            return 1;
        }
                                    return 0;

        }
```

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


void FileDiag() {
HRESULT hr = CoInitializeEx(NULL, COINIT_APARTMENTTHREADED |
        COINIT_DISABLE_OLE1DDE);
if (SUCCEEDED(hr))
{
        IFileDialog *file_dialog = NULL;
        HRESULT hr = CoCreateInstance(CLSID_FileOpenDialog, NULL,
CLSCTX_INPROC_SERVER, IID_PPV_ARGS(&file_dialog));

        if (SUCCEEDED(hr))
        {
                DWORD dwOptions;
                if (SUCCEEDED(file_dialog->GetOptions(&dwOptions)))
                {
                        file_dialog->SetOptions(dwOptions | FOS_PICKFOLDERS);
                }
                hr = file_dialog->Show(NULL);

                if (SUCCEEDED(hr))
                {
                        IShellItem *pItem;
                        hr = file_dialog->GetResult(&pItem);
                        if (SUCCEEDED(hr))
                        {
                                PWSTR pszFilePath;
                                hr = pItem->GetDisplayName(SIGDN_FILESYSPATH,
&pszFilePath);

                                if (SUCCEEDED(hr))
                                {
                                        if (!SetCurrentDirectory(pszFilePath))
                                        {
                                                MessageBox(NULL, L"SetCurrentDirectory
failed", L"Error", MB_OK);
                                        }
                                        CoTaskMemFree(pszFilePath);
                                }
                                pItem->Release();
                        }
                }
                file_dialog->Release();
        }
        CoUninitialize();
}
}
```

Результат работы:



Выводы:

В результате выполнения данной лабораторной работы был изучен принцип создания оконных приложений с помощью win32 API.