МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №5**

**з дисципліни «Операційні системи»**

**на тему: «Вивчення механізму повідомлень ОС Windows»**

Виконав:

Студентк гр. ПЗ2011

Кулик С. В.

Прийняв:

Андрющенко В. О.

Дніпро, 2023

**Тема**: Вивчення механізму повідомлень ОС Windows.  
**Мета:** Отримати практичні навики міжпроцесного обміну даними за допомогою апарату повідомлень.

**Короткі теоретичні відомості**

Всі операційні системи сімейства Windows відносяться до типа систем, керованих подіями. Це означає, що робота прикладних програм будується у вигляді реакції програми на певний набір подій, тобто змін в стані системи і зовнішніх пристроїв. У системі Windows ці події називаються повідомленнями, і з кожним повідомленням пов'язана структура даних, що містить інформацію, корисну для обробки повідомлення.

Windows використовує для передачі повідомлення окну чергу повідомлень. Кожен потік має свою чергу повідомлень. Потік повинен переглядати свою чергу повідомлень і направляти повідомлення на обробку відповідному вікну.

Є два принципово різних способу посилки повідомлень: синхронний (send) і асинхронний (post). Посилаючи повідомлення синхронно, відправник чекає закінчення його обробки, перш ніж продовжити роботу. Асинхронна посилка нагадує опускання листа в поштову скриньку: опустив і забув. Система синхронним способом посилає вікну повідомлення про створення, зміну стану і закриття (наприклад, WM\_CREATE, WM\_ACTIVATE, WM\_SETFOCUS, WM\_SETCURSOR, WM\_DESTROY) а також багато інших повідомлень. До асинхронних повідомлень відносяться повідомлення від клавіатури і миші, з деякою обмовкою до них можна віднести також WM\_PAINT і WM\_TIMER. Програма користувача може посилати будь-які повідомлення синхронним або асинхронним способом, як визнає потрібним розробник.

**Постановка завдання**

Скласти програму, що є простим застосуванням Win32. У програмі використовувати для створення інтерфейсу функції API Windows. У головному вікні додатка розмістити декілька (можна вибрати самостійно) інтерфейсних елементів, управління якими здійснюється за допомогою апарату повідомлень (обмін повідомленнями з інтерфейсними елементами двосторонній);

**Текст програми**

// WindowsProject1.cpp : Определяет точку входа для приложения.

//

#define \_CRT\_SECURE\_NO\_WARNINGS

#define MAX\_LOADSTRING 100

#include "framework.h"

#include "Resource.h"

#include <windows.h>

#include <stdlib.h>

#include <string>

#include <tchar.h>

#include <iostream>

#include <map>

#include <fstream>

#include <vector>

using namespace std;

HINSTANCE hInst;

WCHAR szTitle[MAX\_LOADSTRING] = L"Window 2";

WCHAR szWindowClass[MAX\_LOADSTRING];

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

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

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

std::map<string, HWND> windows;

vector<string> split(const string& str, const string& delimiter) {

vector<string> tokens;

size\_t start = 0, end = 0;

while ((end = str.find(delimiter, start)) != std::string::npos) {

tokens.push\_back(str.substr(start, end - start));

start = end + delimiter.length();

}

tokens.push\_back(str.substr(start));

return tokens;

}

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

LoadStringW(hInstance, IDC\_WINDOWSPROJECT1, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance(hInstance, nCmdShow))

return FALSE;

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_WINDOWSPROJECT1));

map<UINT, string> value;

ifstream infile("D:\\windows\_massages.txt"); // відкрити файл

if (infile.is\_open()) { // перевірка відкриття файлу

string line;

while (getline(infile, line)) { // читання рядку у файл

vector<string> strs = split(line, "\t");

value[stoi(strs[2])] = strs[4];

}

infile.close(); // закрити файл

}

else {

std::cerr << "Failed to open file with windows massages!" << '\n';

}

std::map<UINT, int> counter;

std::map<UINT, int> ::iterator it;

MSG msg;

// Цикл основного сообщения:

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

{

//підрахунок повідомлень

std::cout << &msg << std::endl;

it = counter.find(msg.message);

if (it == counter.end()) {

counter[msg.message] = 1;

}

else {

counter[msg.message]++;

}

TranslateMessage(&msg);

DispatchMessage(&msg);

}

std::ofstream out;

out.open("D:\\massages.txt");

std::map<UINT, std::string> ::iterator it\_val;

it = counter.begin();

it\_val = value.begin();

for (; it != counter.end(); it++, it\_val++) {

out << "Id: " << it->first << ", msg: " << value[it->first] << ", count: " << it->second << std::endl;

}

out.close();

return (int)msg.wParam;

}

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(IDC\_WINDOWSPROJECT1));

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

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

wcex.lpszMenuName = nullptr;

wcex.lpszClassName = szWindowClass;

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

return RegisterClassExW(&wcex);

}

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

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

windows["window"] = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, CW\_USEDEFAULT,

500, 150, NULL, NULL, hInstance, NULL);

windows["static\_text"] = CreateWindow(L"static", L"Повідомлення: ", WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON | ES\_CENTER,

5, 5, 150, 30, windows["window"], (HMENU)4, hInstance, NULL);

windows["edit"] = CreateWindow(L"Edit", L"Hello world!", WS\_CHILD | WS\_VISIBLE | WS\_BORDER | ES\_CENTER,

160, 5, 300, 30, windows["window"], (HMENU)3, hInstance, NULL);

windows["button\_clear"] = CreateWindow(L"button", L"Видалити", WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

5, 50, 150, 30, windows["window"], (HMENU)1, hInstance, NULL);

windows["button\_send"] = CreateWindow(L"button", L"Передати повідомлення", WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

160, 50, 300, 30, windows["window"], (HMENU)2, hInstance, NULL);

if (!windows["window"])

{

MessageBox(NULL, L"Помилка створення вікна!", L"Сповіщення", NULL);

return 1;

}

MoveWindow(windows["window"], 700, 100, 500, 120, TRUE);

ShowWindow(windows["window"], nCmdShow);

return TRUE;

}

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

{

PAINTSTRUCT ps;

HDC hdc;

switch (message)

{

case WM\_DESTROY:

PostQuitMessage(0);

break;

case WM\_PAINT:

break;

case WM\_COPYDATA: {

PCOPYDATASTRUCT s = (PCOPYDATASTRUCT)lParam;

WCHAR\* str = new WCHAR[50];

lstrcpy(str, (LPWSTR)s->lpData);

SetDlgItemText(hWnd, 3, LPWSTR(str));

break;

}

case WM\_COMMAND:

{

if (LOWORD(wParam) == 1)

{

hdc = BeginPaint(hWnd, &ps);

SetDlgItemText(hWnd, 3, L"");

EndPaint(hWnd, &ps);

}

if (LOWORD(wParam) == 2)

{

WCHAR str[50];

GetDlgItemText(hWnd, 3, LPWSTR(str), 50);

COPYDATASTRUCT cd;

cd.dwData = 0;

cd.cbData = lstrlen(str) \* 2 + 1;

cd.lpData = str;

HWND hRecieverWnd = FindWindow(NULL, L"Window 1");

if (hRecieverWnd != 0)

SendMessage(hRecieverWnd, WM\_COPYDATA, 0, (LPARAM)&cd);

else

MessageBox(NULL, L"Вікно приймання не створенно", L"Сповіщення", NULL);

}

break;

}

default:

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

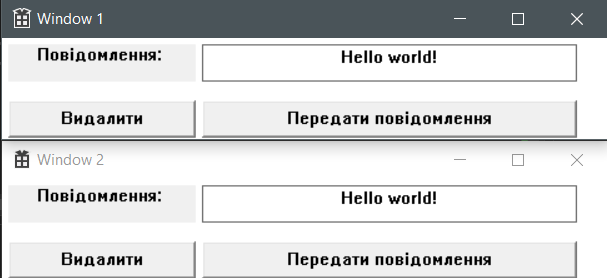
break;

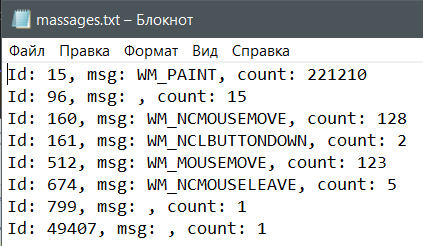
}

return 0;

}

**Результати програми та тестові прилади**





**Висновки**

У процесі виконання лабораторної роботи було вивчено механізм передачі повідомлень ОС Windows. За допомогою вікон процеси можуть взаємодіяти між собою. Крім того, вікна можуть бути призначені для обробки певних повідомлень.