# ЛАБОРАТОРНА РОБОТА №2.
# ПРОГРАМУВАННЯ СИСТЕМНИХ ЗАДАЧ З ВИКОРИСТАННЯМ API-ІНТЕРФЕЙСУ ОС WINDOWS (2 год.)

**Мета:** Ознайомитись з можливостями та набути навиків програмування в ОС Windows та засвоїти навики використання функцій API ОС Windows.

## ТЕОРЕТИЧНІ ВІДОМОСТІ

**API** (Application Programming Interface) інтерфейс операційної системи Windows — це набір функцій, класів, структур даних та інших програмних інтерфейсів, наданих Microsoft, які дозволяють програмістам взаємодіяти з операційною системою Windows. Ці інтерфейси надають програмам можливість використовувати послуги та функції, які пропонує операційна система.

**Основні поняття про API Windows**:

**Взаємодія з ОС**: API Windows дозволяють розробникам отримувати доступ до низькорівневих функцій операційної системи, таких як керування файлами, мережевими підключеннями, графікою, пам'яттю, пристроями вводу/виводу та іншими компонентами.

**Стандартний інтерфейс**: Всі функції Windows API стандартизовані, що означає, що вони мають визначені параметри, типи даних та спосіб повернення результатів. Це забезпечує стабільну та передбачувану взаємодію між програмами та операційною системою.

**Модульність**: API Windows організовані в кілька модулів або бібліотек, таких як kernel32.dll для операцій з ядром, user32.dll для роботи з користувацькими інтерфейсами та діалоговими вікнами, gdi32.dll для графічного виводу, advapi32.dll для роботи з реєстром та безпекою.

Основні категорії Windows API:

Windows Base API: Функції для управління процесами, потоками, пам'яттю, файлами, каталогами, пристроями, а також для обробки помилок.

Windows User API: Функції для роботи з графічним інтерфейсом користувача, включаючи вікна, кнопки, меню, діалогові вікна, клавіатуру та мишу.

Windows GDI (Graphics Device Interface): Функції для роботи з графікою, відображення тексту, малювання ліній, фігур, обробки зображень та шрифтів.

Windows Networking API: Функції для роботи з мережами, включаючи протоколи TCP/IP, управління мережею та віддаленими підключеннями.

Windows Security API: Функції для управління безпекою, включаючи аутентифікацію користувачів, доступ до файлів, шифрування та управління правами доступу.

Shell API: Функції для інтеграції додатків з оболонкою Windows (наприклад, робочим столом, файловим провідником), роботи з файлами, папками, ярликами.

Виклик API-функцій операційної системи Windows має свої особливості і вимоги. Ось деякі ключові моменти, на які слід звернути увагу:

**1. Використання заголовних файлів**

Щоб викликати API-функції Windows, потрібно включити відповідні заголовні файли. Наприклад, для більшості функцій Windows API вам потрібно підключити:

```
#include <windows.h>
```

**2. Використання бібліотек**

Бібліотеки (lib-файли) забезпечують реалізацію функцій, оголошених у заголовних файлах. Зазвичай, функції Windows API автоматично лінкуються, якщо ви використовуєте середовище розробки, таке як Microsoft Visual Studio. Але для ручного лінкування ви можете використовувати:

```
#pragma comment(lib, "User32.lib")
#pragma comment(lib, "Kernel32.lib")
```

**3. Використання параметрів**

Багато функцій API вимагають специфічних типів параметрів, і важливо надавати правильні значення. Наприклад, функції, що працюють з рядками, можуть використовувати LPCTSTR, LPCWSTR, або LPCSTR в залежності від того, чи ви використовуєте Unicode чи ANSI.

**4. Обробка помилок**

Більшість функцій Windows API повертають значення, яке слід перевіряти для визначення успішності операції. Якщо функція повертає FALSE або NULL, ви повинні викликати GetLastError() для отримання коду помилки, щоб дізнатися, що пішло не так:

```
DWORD error = GetLastError();
```

**5. Правильне управління ресурсами**

Деякі API-функції виділяють ресурси, такі як дескриптори, пам'ять або інші об'єкти, які потрібно явно закривати або звільняти після використання. Наприклад, якщо ви отримали дескриптор файлу через CreateFile(), його потрібно закрити через CloseHandle():

```
HANDLE hFile = CreateFile(...);
if (hFile != INVALID_HANDLE_VALUE) {
    // Виконання операцій з файлом
    CloseHandle(hFile);
```

**6. Розуміння конвенцій виклику**

Функції Windows API використовують конвенцію виклику __stdcall, яка відрізняється від __cdecl, що використовується у багатьох C++ функціях. Це впливає на те, як передаються параметри і як відбувається повернення значення.

Цю конвенцію, як правило, не потрібно враховувати, якщо ви використовуєте правильні заголовки і бібліотеки.

**7. Розуміння специфічних типів даних**

Багато функцій Windows API використовують специфічні типи даних, такі як HANDLE, HWND, HINSTANCE, які є дескрипторами різних системних об'єктів. Ці типи даних не мають явних значень у стандартному C++ і мають специфічний формат для операційної системи.

**8. Забезпечення сумісності**

Деякі функції Windows API можуть бути доступні тільки в певних версіях операційної системи. Щоб перевірити, чи підтримується певна функція на даній версії ОС, ви можете використовувати макроси, такі як WINVER або NTDDI_VERSION.

**9. Обробка Unicode та ANSI**

Windows API підтримує як Unicode, так і ANSI версії функцій. Функції з суфіксом W (наприклад, CreateFileW()) є Unicode версіями, а функції з суфіксом A (наприклад, CreateFileA()) є ANSI версіями. Макроси, такі як UNICODE або _UNICODE, можуть вплинути на те, яка версія функцій буде використовуватися.

**Дескриптор** — це абстрактний ідентифікатор або ключ, який використовуються для представлення ресурсів у системі операційного середовища. У контексті операційних систем і програмування, дескриптор часто використовується для доступу до різних системних об'єктів, таких як файли, сокети, процеси, потоки, дисплеї тощо.

**Основні типи дескрипторів:**

**Файлові дескриптори**:

Використовуються для представлення відкритих файлів.

У Windows це можуть бути дескриптори, отримані через функції, такі як CreateFile().

**Дескриптори процесів і потоків**:

Використовуються для представлення процесів і потоків.

У Windows це можуть бути дескриптори, отримані через функції, такі як CreateProcess() або CreateThread().

**Дескриптори сокетів**:

Використовуються для представлення мережевих з'єднань.

У Windows також використовуються дескриптори для роботи з мережевими сокетами.

**Дескриптори дисплеїв і графічних об'єктів**:

Використовуються для доступу до графічних контекстів і елементів інтерфейсу.

У Windows це можуть бути дескриптори контексту пристрою (HDC), об'єктів графіки (HRGN), шрифтів тощо.

**Як працюють дескриптори**:

**Створення**: дескриптор створюється через виклик системної функції, яка виділяє або ініціалізує ресурс і повертає дескриптор як результат. Наприклад, CreateFile() у Windows повертає дескриптор файлу.

**Використання**: програмісти використовують дескриптор для доступу до відповідного ресурсу, наприклад, для читання або запису в файл.

**Закриття**: коли робота з ресурсом завершена, дескриптор повинен бути закритий через відповідну функцію, наприклад, CloseHandle() у Windows, щоб звільнити ресурси системи.

## КОНТРОЛЬНІ ПИТАННЯ

1. Що таке API ОС Windows?
2. Які особливості виклику функцій API?
3. Які функції API ви знаєте?
4. Що таке дескриптор?
5. Яку конвенцію виклику використовують API ОС Windows?

## ЛІТЕРАТУРА

1. Yosifovich P. Windows Native API Programming / Pavel Yosifovich., 2024. – 404 с. – (Amazon Digital Services LLC).
2. Programming reference for the Win32 API [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/windows/win32/api/.

## ЗАВДАННЯ

1. Написати програму згідно свого варіанту завдань.
2. Відкомпілювати програму під 32-х розрядну і 64-х розрядну платформи відповідно та запустити їх на виконання.
3. Запустити програму під 32-х розрядну і 64-х розрядну платформи в покроковому виконанні, вибрати вкладу Disassembly (Debug\Windows\ Disassembly), знайти в коді рядки, які відповідають за передачу параметрів і виклик API функцій.
4. Підготувати та захистити звіт. В звіті **обов'язково** мають бути описані параметри виклику усіх використаних API функцій.
5. В звіті **обов'язково** мають бути код програми і скріншоти з результатами виконання програми!
6. Також в звіті мають бути рядки коду з вкладки Disassembly з позначенням відповідних параметрів і викликів усіх використаних API функцій.

*Таблиця 2.1.*

**Варіанти завдань**

| № | Завдання |
|---|---|
| 1. | Написати мовою програмування C/C++ програму, яка виведе на екран список усіх користувачів операційної системи Windows (скористатись функцією NetUserEnum()). Результати вивести на екран за допомогою функції MessageBox(). |
| 2. | Написати мовою програмування C/C++ програму, яка визначає інформацію про підключені логічні диски в ОС Windows (скористатись функцією GetLogicalDriveStrings()). Результати вивести на екран за допомогою функції MessageBox(). |
| 3. | Написати мовою програмування C/C++ програму, яка визначає кількість доступної оперативної пам'яті та розмір файлу підкачки в ОС Windows (скористатись функцією GlobalMemoryStatusEx()). Результати вивести на екран за допомогою функції MessageBox(). |
| 4. | Написати мовою програмування C/C++ програму, яка отримує інформацію про процесор (архітектура, кількість ядер, тощо) в ОС Windows (скористатись функцією GetSystemInfo()). Результати вивести на екран за допомогою функції MessageBox(). |
| 5. | Написати мовою програмування C/C++ програму, яка отримує інформацію про графічний адаптер (назву, тип адаптера, обсяг відеопам'яті) в ОС Windows (скористатись функцією EnumDisplayDevices()). Результати вивести на екран за допомогою функції MessageBox(). |
| 6. | Написати мовою програмування C/C++ програму, яка визначає час безперервної роботи системи (час з моменту завантаження ОС) в ОС Windows (скористатись функцією GetTickCount64()). Результати вивести на екран за допомогою функції MessageBox(). |

| № | Завдання |
|---|---|
| 7. | Написати мовою програмування C/C++ програму, яка отримує поточний системний час в ОС Windows (скористатись функцією GetSystemTime()). Результати вивести на екран за допомогою функції MessageBox(). |
| 8. | Написати мовою програмування C/C++ програму, яка визначає роздільну здатність екрана в ОС Windows (скористатись функцією EnumDisplaySettings()). Результати вивести на екран за допомогою функції MessageBox(). |
| 9. | Написати мовою програмування C/C++ програму, яка визначає IP-адреси мережевих адаптерів в ОС Windows (скористатись функцією GetAdaptersInfo()). Результати вивести на екран за допомогою функції MessageBox(). |
| 10. | Написати мовою програмування C/C++ програму, яка визначає MAC-адреси мережевих адаптерів в ОС Windows (скористатись функцією GetAdaptersInfo()). Результати вивести на екран за допомогою функції MessageBox(). |
| 11. | Написати мовою програмування C/C++ програму, яка визначає кількість вільного місця на заданому логічному диску в ОС Windows (скористатись функцією GetDiskFreeSpaceEx()). Результати вивести на екран за допомогою функції MessageBox(). |
| 12. | Написати мовою програмування C/C++ програму, яка визначає час створення заданого файлу в ОС Windows (скористатись функцією GetFileTime()). Результати вивести на екран за допомогою функції MessageBox(). |
| 13. | Написати мовою програмування C/C++ програму, яка визначає розмір заданого файлу в ОС Windows (скористатись функцією GetFileSizeEx()). Результати вивести на екран за допомогою функції MessageBox(). |
| 14. | Написати мовою програмування C/C++ програму, яка отримує поточний локальний час в ОС Windows (скористатись функцією GetLocalTime()). Результати вивести на екран за допомогою функції MessageBox(). |
| 15. | Написати мовою програмування C/C++ програму, яка визначає частоту оновлення екрану в ОС Windows (скористатись функцією EnumDisplaySettings()). Результати вивести на екран за допомогою функції MessageBox(). |
| 16. | Написати мовою програмування C/C++ програму, яка визначає кількість зайнятого місця на заданому логічному диску в ОС Windows (скористатись функцією GetDiskFreeSpaceEx()). Результати вивести на екран за допомогою функції MessageBox(). |
| 17. | Написати мовою програмування C/C++ програму, яка визначає глибину кольору (кількість кольорів, що відображаються на екрані) в ОС Windows (скористатись функцією GetDeviceCaps()). Результати вивести на екран за допомогою функції MessageBox(). |
| 18. | Написати мовою програмування C/C++ програму, яка отримує дескриптор і ідентифікатор для поточного процесу в ОС Windows (скористатись функціями GetCurrentProcess() і GetProcessId()). Результати вивести на екран за допомогою функції MessageBox(). |
| 19. | Написати мовою програмування C/C++ програму, яка визначить тип заданого логічного диску в ОС Windows (скористатись функцією GetDriveType()). Результати вивести на екран за допомогою функції MessageBox(). |

| № | Завдання |
|---|---|
| 20. | Написати мовою програмування C/C++ програму, яка визначить тип файлової системи заданого логічного диску в ОС Windows (скористатись функцією GetVolumeInformation()). Результати вивести на екран за допомогою функції MessageBox(). |
| 21. | Написати мовою програмування C/C++ програму, яка отримує інформацію про завантаження процесора в ОС Windows (скористатись функцією GetSystemTimes()). Результати вивести на екран за допомогою функції MessageBox(). |
| 22. | Написати мовою програмування C/C++ програму, яка отримує список модулів (DLL), завантажених у поточний процес в ОС Windows (скористатись функцією EnumProcessModules()). Результати вивести на екран за допомогою функції MessageBox(). |
| 23. | Написати мовою програмування C/C++ програму, яка отримує інформацію про підключені монітори в ОС Windows (скористатись функцією EnumDisplayMonitors()). Результати вивести на екран за допомогою функції MessageBox(). |
| 24. | Написати мовою програмування C/C++ програму, яка отримує інформацію про час початку виконання поточного процесу в ОС Windows (скористатись функцією GetProcessTimes()). Результати вивести на екран за допомогою функції MessageBox(). |
| 25. | Написати мовою програмування C/C++ програму, яка визначає тип клавіатури в ОС Windows (скористатись функцією GetKeyboardType()). Результати вивести на екран за допомогою функції MessageBox(). |
| 26. | Написати мовою програмування C/C++ програму, яка визначить дату останнього завантаження ОС Windows (скористатись функцією GetTickCount64()). Результати вивести на екран за допомогою функції MessageBox(). |
| 27. | Написати мовою програмування C/C++ програму, яка отримує інформацію з буфера обміну в ОС Windows (скористатись функцією GetClipboardData()). Результати вивести на екран за допомогою функції MessageBox(). |
| 28. | Написати мовою програмування C/C++ програму, яка отримує інформацію про колір фону стандартного вікна в ОС Windows (скористатись функцією GetSysColor()). Результати вивести на екран за допомогою функції MessageBox(). |
| 29. | Написати мовою програмування C/C++ програму, яка отримує інформацію про колір колір тексту в стандартному вікні в ОС Windows (скористатись функцією GetSysColor()). Результати вивести на екран за допомогою функції MessageBox(). |
| 30. | Написати мовою програмування C/C++ програму, яка отримує інформацію про відсоток використаної і вільної пам'яті в ОС Windows (скористатись функцією GlobalMemoryStatusEx()). Результати вивести на екран за допомогою функції MessageBox(). |

# ПРИКЛАД ВИКОНАННЯ

Для отримання імені комп'ютера використаємо функцію `GetComputerName()`, яка має два варіанти – `GetComputerNameA()` і `GetComputerNameW()`. Закінчення `A` позначає, що функція працює з символами у кодуванні ANSI (1 байт), а закінчення `W` позначає, що функція працює з символами у кодуванні UNICODE (2 байти).

У програмі використаємо другий варіант:

```
BOOL GetComputerNameW(
  [out]     LPWSTR  lpBuffer,
  [in, out] LPDWORD nSize
);
```

Функція має два параметри:

`[out] lpBuffer` – вказівник на буфер, який отримує ім'я комп'ютера. Розмір буфера має бути достатньо великим, щоб містити MAX_COMPUTERNAME_LENGTH + 1 символ.

`[in, out] nSize` – якщо використовується як вхідні дані, то вказується розмір буфера у TCHAR. У вихідних даних — кількість TCHAR , скопійованих у буфері призначення, без урахування символу завершення рядка. Якщо буфер занадто малий, функція завершується з помилкою і GetLastError() повертає ERROR_BUFFER_OVERFLOW.

Для виводу інформації на екран використаємо функцію `MessageBox()`, яка також має варіант для роботи з UNICODE – `MessageBoxW()`.

```
int MessageBoxW(
  [in, optional] HWND    hWnd,
  [in, optional] LPCWSTR lpText,
  [in, optional] LPCWSTR lpCaption,
  [in]           UINT    uType
);
```

Функція має такі параметри:

`[in, optional] hWnd` – дескриптор вікна власника вікна повідомлення, яке буде створено. Якщо цей параметр дорівнює NULL, вікно повідомлення не має вікна власника.

`[in, optional] lpText` – повідомлення, яке буде показано. Якщо рядок складається з кількох рядків, ви можете розділити рядки за допомогою символу переходу на новий рядкок.

`[in, optional] lpCaption` – заголовок діалогового вікна. Якщо цей параметр дорівнює NULL, типовим заголовком є Error.

`[in] uType -` вміст і поведінка діалогового вікна. Цей параметр може бути комбінацією прапорців. Щоб позначити кнопки, які відображаються у вікні повідомлення, укажіть одне з наведених нижче значень:

- MB_ABORTRETRYIGNORE 0x00000002L – Вікно повідомлення містить три кнопки: **Abort**, **Retry**, і **Ignore**..
- MB_CANCELTRYCONTINUE 0x00000006L – Вікно повідомлення містить три кнопки: **Cancel**, **Try Again**, **Continue**. Використовуйте цей тип вікна повідомлення замість MB_ABORTRETRYIGNORE.
- MB_HELP 0x00004000L – Додає кнопку **Help** у вікно повідомлень. Коли користувач натискає кнопку **Help** або натискає F1, система надсилає повідомлення WM_HELP власнику.
- MB_OK 0x00000000L – Вікно повідомлень містить одну кнопку: **OK**. Це значення за замовчуванням.

- MB_OKCANCEL 0x00000001L – Вікно повідомлення містить дві кнопки: **OK** і **Cancel**.
- MB_RETRYCANCEL 0x00000005L – Вікно повідомлення містить дві кнопки: **Retry** і **Cancel**.
- MB_YESNO 0x00000004L – Вікно повідомлення містить дві кнопки: **Yes** і **No**.
- MB_YESNOCANCEL 0x00000003L – Вікно повідомлення містить три кнопки: **Yes**, **No**, і **Cancel**.

**Код програми, яка виконує поставлене завдання:**

```
// Програма, яка отримує інформацію про ім'я комп'ютера в ОС Windows
// Результат виводиться на екран за допомогою функції MessageBox()

#include <windows.h>

int main() {
    // Буфер для зберігання імені комп'ютера
    TCHAR computerName[MAX_COMPUTERNAME_LENGTH + 1];
    DWORD size = sizeof(computerName) / sizeof(computerName[0]);

    // Отримання імені комп'ютера
    if (GetComputerName(computerName, &size)) {

        // Виведення ім'я комп'ютера за допомогою MessageBox
        MessageBox(NULL, computerName, TEXT("Ім'я комп'ютера"), MB_OK |
MB_ICONINFORMATION);
    }
    else {
        // Обробка помилки
        DWORD error = GetLastError();
        MessageBox(NULL, TEXT("Не вдалося отримати ім'я комп'ютера."),
TEXT("Помилка"), MB_OK | MB_ICONERROR);
    }

    return 0;
}
```

**Рядки коду з вкладки Disassembly для програми, скомпільованої під 32-х розрядну платформу:**

```
    // Отримання імені комп'ютера
    if (GetComputerName(computerName, &size)) {
001D1017  lea         eax,[size]
    // Занесення параметра nSize у стек
001D101A  push        eax
001D101B  lea         ecx,[computerName]
    // Занесення параметра lpBuffer у стек
001D101E  push        ecx
    // Виклик функції GetComputerName()
001D101F  call        dword ptr [__imp__GetComputerNameW@8 (01D2004h)]

        // Виведення ім'я комп'ютера за допомогою MessageBox
        MessageBox(NULL, computerName, TEXT("Ім'я комп'ютера"), MB_OK |
MB_ICONINFORMATION);
    // Занесення параметра uType у стек
001D1029  push        40h
    // Занесення параметра lpCaption у стек
001D102B  push        1D3000h
001D1030  lea         edx,[computerName]
    // Занесення параметра lpText у стек
001D1033  push        edx
    // Занесення параметра hWnd у стек
001D1034  push        0
    // Виклик функції MessageBox()
001D1036  call        dword ptr [__imp__MessageBoxW@16 (01D203Ch)]
```

**Рядки коду з вкладки Disassembly для програми, скомпільованої під 64-х розрядну платформу:**

```
    // Отримання імені комп'ютера
    if (GetComputerName(computerName, &size)) {
    // Занесення параметра nSize у регістр rdx
00007FF6E807101B  lea         rdx,[size]
    // Занесення параметра lpBuffer у регістр rcx
00007FF6E8071020  lea         rcx,[computerName]
    // Виклик функції GetComputerName()
00007FF6E8071025  call        qword ptr [__imp_GetComputerNameW (07FF6E8072008h)]

        // Виведення ім'я комп'ютера за допомогою MessageBox
        MessageBox(NULL, computerName, TEXT("Ім'я комп'ютера"), MB_OK |
MB_ICONINFORMATION);
    // Занесення параметра uType у r9d
00007FF6E807102F  mov         r9d,40h
    // Занесення у lpCaption у r8
00007FF6E8071035  lea         r8,[__xt_z+8h (07FF6E8072240h)]
    // Занесення параметра lpText у rdx
00007FF6E807103C  lea         rdx,[computerName]
    // Занесення параметра hWnd у ecx
00007FF6E8071041  xor         ecx,ecx
    // Виклик функції MessageBox()
00007FF6E8071043  call        qword ptr [__imp_MessageBoxW (07FF6E8072090h)]
```