Тема 3. Принципи програмування для Windows

План
1. Типи даних Windows
2. Принципи найменування змінних (Угорська нотація)
3. ASCII та Unicode
4. Домовленості для Windows-програмування

Програма користувача не має прямого доступу до внутрішніх даних ядра операційної системи. Якщо такі дані потрібні для виконання системного виклику, ОС створює для користувача об’єкт ядра (kernel object). До об'єктів відносяться файли, процеси, потоки, канали міжпроцесної взаємодії, проекції файлів, події, м'ютекси, семафори і багато іншого. Практично об’єкт – це блок пам’яті (структура даних) в адресному просторі ОС, в якому зберігаються дані про об’єкт. Внутрішня структура даних об’єкта прихована. Певні атрибути об’єкта притаманні кожному об’єкту (дескриптор, лічильник користувачів, маркер доступу), інші специфічні для конкретних об’єктів. Користувач не може прямо звертатися до даних об’єкта, для цього операційна система надає системні функції (методи об’єкта). Ідентифікація об’єктів і доступ до них здійснюється через дескриптор (handle) – своєрідний номер об’єкта. Всі маніпуляції з об'єктами ядра здійснюються виключно за допомогою функцій API. Подібна організація роботи нагадує принцип інкапсуляції даних в об'єктно-орієнтованому програмуванні, хоча сама Windows не є об'єктно-орієнтованою.

Windows – багатий і гнучкий інтерфейс. Це значить, що часто задачі можуть розв'язуватися за допомогою кількох функцій, є також допоміжні функції (convenience functions), одержані об'єднанням послідовності кількох простих (наприклад, функція CopyFile). Функції API мають численні параметри і прапорці, що дає велику кількість варіантів використання функцій.

Базовою одиницею виконання програмного коду є потік (thread). В кожному процесі (process) можуть виконуватися один або декілька потоків. Кожний процес має свій унікальний ідентифікатор, окремий адресний простір, набір системних ресурсів і мінімум один потік. Потоки поділяють адресний простір процесу. Потоки легші і ефективніші, ніж процеси. Windows багатозадачна операційна система, для ефективного керування потоками вона пропонує механізми синхронізації доступу до ресурсів і засоби взаємодії.

Для функцій Windows характерні довгі описові імена, які у більшості випадків дають можливість зробити припущення про призначення функції. Наведені нижче як приклад імена функцій ілюструють не лише домовленості про використання імен, але і багатоликість функцій Windows:

```
CreateFile
WaitForSingleObject
GetFileSize
GetLastError
GetStdHandle
```

Windows має численний набір власних типів даних. В іменах типів, що є вказівниками, операція * не використовується, наприклад, тип LPTSTR вже визначений як TCHAR *, а LPCTSTR як const TCHAR *. Типи даних Windows описуються великими літерами: BOOL, DWORD, LPDWORD.

1. Типи даних Windows

Windows API визначає істотне число типів даних для позначення параметрів функцій і значень, що повертаються. Ці специфічні типи Windows також поширюються на функції, які визначаються в MFC. Кожен з цих типів Windows відображатиметься на деякий тип C++, але відповідність між типами Windows і типами C++ може змінюватися, тому слід завжди використовувати типи Windows.

Наприклад, в минулому тип Windows **WORD** був визначений в одній версії Windows як unsigned short**,** а в іншій версії Windows як unsigned int. На 16-бітних машинах ці типи еквівалентні, але на 32-розрядних машинах вони рішуче відмінні, отже використовуючи тип C++ замість типу Windows можна нарватися на проблеми.

Повний перелік типів Windows можна знайти у документації, а тут подано лише ті, які зустрічаються найчастіше:

BOOLEAN		8-бітний логічний тип
BYTE	BYTE	8-бітний байт.
CHAR		8-бітний символ.
TCHAR		узагальнений (generic) символьний тип.
USHORT		16-бітний цілий тип
WCHAR	WORD	16-бітний символ.
WORD		16-бітове беззнакове ціле, яке відповідає типу unsigned short in C++.
BOOL		Логічна змінна (`TRUE` або `FALSE`) . Зауважте, що це не те саме, що тип C++ bool, який має значення true або false.
DWORD	DWORD	32-бітне беззнакове ціле, яке відповідає типу unsigned long у C++.
HANDLE		Дескриптор об'єкта – 32-бітне ціле число, яке ідентифікує об'єкт певного типу.
HBRUSH	DWORD	Дескриптор пензля для заливки області кольором.
HCURSOR		Дескриптор курсору.
HFILE		Дескриптор файлу.
HINSTANCE		Дескриптор примірника об'єкта.
LPARAM	DWORD	Параметр повідомлення.
WPARAM		Параметр повідомлення
LPCSTR		Вказівник на константний рядок 8-бітних символів із нулем у кінці.
LPHANDLE	DWORD	Вказівник на дескриптор.
LRESULT		Значення із знаком, яке одержується в результаті обробки повідомлення.
UINT	DWORD	Беззнакове ціле число
DOUBLE	QWORD	Число з плаваючою комою подвійної точності

Усі типи, які використовує Windows та прототипи функцій Windows API містяться в заголовковому файлі `windows.h`, отже необхідно включати цей файл у програми для Windows.

2. Принципи найменування змінних (Угорська нотація)

Windows використовує описові імена змінних, або т.зв. "угорську нотацію". Суть угорської нотації полягає в тому, що перед іменем змінної або функції ставиться префікс із однієї або кількох букв, який говорить про тип цієї змінної.

Так, префікс **n** позначає цілочисельну змінну, **sz** – символьний рядок, який закінчується двійковим нулем, **h** – дескриптор того чи іншого об'єкту. Звідси і беруться дещо незграбні на перший погляд імена змінних програми на кшталт **szClassName** або **szTitle**. Угорська система широко використовується в описах внутрішніх структур Windows. Так, в структурі типу WNDCLASS, із якою має справу функція RegisterClass(), є члени **lpszClassName** (**l**ong **p**ointer to a **z**ero terminated **s**tring with **Name** of **Class**, далекий вказівник на символьний рядок із нулем в кінці, що містить ім'я класу), **lpfnWndProc** (**l**ong **p**ointer to a **fu**n**c**tion **WndProc**, далекий вказівник на функцію вікна). Так само, **dwAccess** — подвійне слово (32 біта), що містить прапорці прав доступу до файлу, де "dw" означає "Double word" — "подвійне слово". Уяв-

лення про угорську нотацію дещо полегшує вивчення вмісту цих структур і використання їх елементів у програмах.

В таблиці наведено найбільш уживані префікси угорської нотації (для Win32).

Префікс	Розшифровка	Значення
b	**B**ool	Логічна змінна
by	**By**te	Байт (unsigned char)
c	**C**haracter	Символ
dw	**D**ouble**W**ord	Подвійне слово без знаку
f	**F**unction	Функція
h	**H**andle	Дескриптор об'єкта
l	**L**ong	Довге ціле із знаком
lp	**L**ong**P**ointer	Дальній вказівник
lpsz	**L**ong**P**ointer**S**tring**Z**ero	Дальній вказівник на символьний рядок, що закінчується двійковим нулем
n	i**N**t	Ціле із знаком
p	**P**ointer	Близький вказівник
pt	**P**oin**T**	X і Y координати точки, структура, 32+32 біт
rgb	**R**ed**G**reen**B**lue	Колір із червоної, зеленої та синьої складових, упакованих в 32 біти
sz	**S**tring**Z**ero	Символьний рядок, що закінчується двійковим нулем
u	**U**int	Ціле без знаку
w	**W**ord	Слово без знаку

Нагадаємо, що у Win32 всі вказівники є ближніми, хоча і займають 4 байти. Різниця між ближніми и дальніми вказівниками існувала лише у Win16.

Угорська нотація використовується з метою мінімізації неправильного використання змінних. Кожна змінна повинна трактуватись так, як це було визначено. Неправильне тлумачення легко було зробити у мові C. C++ має сильнішу перевірку відповідності типів, тому проблеми виникають рідше. Компілятор C++ завжди сигналізує помилку при несумісності типів у вашій програмі. З іншого боку, Угорська нотація допомагає зробити програми легшими для розуміння, особливо, коли ви маєте справу із багатьма змінними різних типів, які є параметрами для функцій Windows API.

Угорська нотація у жодному разі не є обов'язковою. Ви, можете не використовувати її взагалі, але у більшості випадків програмується простіше, якщо розумієш, якими є параметри функцій Windows API.

3. ASCII та Unicode

В зв'язку з необхідністю локалізації (англ. *localization*) програмного забезпечення постає питання відповідного кодування рядкових даних. Локалізація програмного продукту – приведення програмного продукту у відповідність із законами та іншими нормативно-правовими актами, стандартами, нормами і правилами, що діють у певній країні. Локалізація передбачає переклад і адаптацію елементів інтерфейсу, допоміжних файлів та документації.

Текстові елементи Win32 API зазвичай представляються у трьох форматах:

- ***Версія ANSI*** (типи CHAR або char);
- ***Версія Unicode*** (типи WCHAR або wchar_t);
- ***Узагальнена версія*** (тип TCHAR).

Версія ANSI представлена відомим символьним набором ASCII або Latin-1. Усі рядки в Unicode представлені 16-бітними значеннями, що дає можливість представити символи більшості мов світу. Починаючи із Windows 2000, усі системні виклики із текстовими параметрами очікують Unicode. Навіть якщо задати параметри в ASCII, вони будуть автоматично перетворені в Unicode при виконанні функції.

Якщо планується поставляти програму в різних кодуваннях, не обов'язково писати два різних коди. Можна створювати універсальні програми із використанням узагальнених типів рядкових даних, які компілюються в ANSI або Unicode. Узагальненим типом даних у windows виступає TCHAR (і відповідно LPTSTR, LPCTSTR).

Для підтримки розширеного кодування Unicode в усі модулі програми включають визначення:

#define UNICODE // для типів Windows
#define _UNICODE // для типів C
#include <windows.h>

Символьні буфери у програмі визначають, вказуючи кількість символів, а не байтів. Для цього підходить функція **sizeof()** (наприклад, sizeof(szBuffer) або sizeof(TCHAR)).

Якщо передбачається викликати функції для обробки узагальнених рядків, після windows.h включають заголовковий файл TChar.h замість String.h. Звичайні рядкові функції замінюють на відповідні узагальнені:

printf()	→	**_tprintf()**
sprintf()	→	**_stprintf()**
strlen()	→	**_tcslen()**
itoa()	→	**_itot()**

Рядкові константи також можуть бути трьох типів:

"This string uses 8-bit characters"	ASCII
L "This string uses 16-bit characters"	Unicode
_T ("This string uses generic characters")	Generic

4. Домовленості для Windows-програмування

У програму завжди включається заголовковий файл `windows.h`, який містить визначення констант, прототипи функцій, опис структур, які використовуються в програмах. SDK включає багато заголовкових файлів, значна їх частина автоматично підключаються до програми через windows.h.

Усі об'єкти ядра, створюються операційною системою за допомогою відповідних функцій, ідентифікуються через дескриптор – змінну типу HANDLE, який повертається при створенні об'єкта і вивільняється при його знищенні. Практично всі об'єкти закриваються за допомогою функції CloseHandle().

Для позначення числових констант і прапорців використовуються символічні імена, які розкривають їх значення:

INVALID_HANDLE_VALUE, GENERIC_READ, STD_INPUT_HANDLE, WM_MOUSEMOVE та ін.

Багато функцій Windows (ReadFile(), WriteFile() та ін.) повертають логічне значення типу BOOL, інші – числове значення. Деякі числові значення свідчать про помилку. Коди системних помилок доступні через функцію GetLastError().

В стандарті ANSI C головна функція має позначення **`main()`**. В Unicode головна функція – **`wmain()`**. Для побудови універсальної програми слід використовувати макрос **`_tmain()`**. Цей макрос розгортається до **`main()`** або до **`wmain()`**, в залежності від значення константи _UNICODE.

Програма з узагальненими типами рядкових даних буде мати вигляд:

```
#include <windows.h>
#include <tchar.h>
int _tmain (int argc, LPTSTR argv[])
{
…
}
```