## Лабораторная работа 3

Худяков Константин А-13-17

### *Управление окнами*

### Подготовка к лабораторной работе

Для выполнения лабораторной работы необходимо ознакомиться (по литературе или в сети) и сделать краткое описание следующих функций и параметров, а также используемых событий (также для работы будут необходимы функции и события, рассмотренные в работах № 1 и № 2):

1. Функции для работы с окнами
   * EnableWindow(handleW1,FALSE);

Функция **EnableWindow** включает или отключает мышь и ввод информации с клавиатуры заданного окна или органа управления. Когда ввод информации заблокирован, окно не принимает ввод информации типа щелчков клавишами мыши и нажатий клавиш клавиатуры. Когда ввод информации включен, окно принимает всю вводимую информацию.

Синтаксис

BOOL EnableWindow

(

HWND hWnd, // дескриптор окна

BOOL bEnable // флажок для включения или отключения ввода

);

* + IsWindowVisible

Функция **IsWindowVisible** находит данные о состоянии видимости заданного окна.

Синтаксис

BOOL IsWindowVisible

(

HWND hWnd // дескриптор окна

);

* + ShowWindow

Функция **ShowWindow** устанавливает состояние показа определяемого окна.

Синтаксис

BOOL ShowWindow

(

HWND hWnd, // дескриптор окна

int nCmdShow // состояние показа окна

);

**SW\_HIDE** - Скрывает окно и активизирует другое окно.

**SW\_SHOW** - Активизирует окно и отображает его текущие размеры и позицию.

1. Послать сообщение окну:
   * SendMessage

SendMessage( UINT message, WPARAM wParam = 0, LPARAM lParam = 0 );

Возвращаемое значение  
Результат обработки сообщения, значение зависит от посланного сообщения. Параметры  
**message**  
Определяет сообщение которое будет послано.  
**wParam**  
Определяет дополнительную зависимую от сообщения информацию.  
**pParam**  
Определяет дополнительную зависимую от сообщения информацию.

Замечания  
Посылает определенное сообщение окну. **SendMessage** функция вызывает процедуру окна непосредственно и не возвращается, пока та процедура окна не обработала сообщение. Это отличие от функции [PostMessage](http://www.firststeps.ru/mfc/steps/help/r.php?260) которая помещает сообщение в очередь сообщений окна и возвращается немедленно.

* + PostMessage

PostMessage( UINT message, WPARAM wParam = 0, LPARAM lParam = 0 );

Возвращаемое значение  
Отлично от нуля если сообщение зарегистрировано, иначе 0.

Параметры  
**message**  
Определяет сообщение которое будет послано.  
**wParam**  
Определяет дополнительную информацию сообщения. Содержание этого параметра зависит от посылаемого сообщения.  
**lParam**  
Определяет дополнительную информацию сообщения. Содержание этого параметра зависит от посылаемого сообщения.

Замечания  
Помещает сообщение в очередь сообщений окна и затем возвращается без того чтобы ждать соответствующее окно которое обрабатывает сообщение. Сообщения в очереди сообщений извлекаются обращениями к функциям**Windows PeekMessage** или **GetMessage**.  
Функция **Windows PostMessage** может использоваться чтобы обратиться к другой прикладной программе.

Например,

*Выполнить пункт меню:*

SendMessage(handleW1,WM\_COMMAND,MAKELONG(32771,0),0);

*Изменить заголовок окна*

SendMessage(handleW1,WM\_SETTEXT,0,LPARAM(LPCTSTR("I See you")));

*Послать сообщение о нажатии мыши:* SendMessage(hWnd,WM\_LBUTTONDOWN,MK\_LBUTTON,

MAKELONG(100,100));

1. Поиск окон:
   * FindWindow

*Пример:* handleWindow=FindWindow(0,"Мое окно");

Функция **FindWindow** разыскивает данные о дескрипторе окна верхнего уровня, чье имя класса и имя окна соответствуют определенным строкам. Эта функция не ищет дочерние окна.

Синтаксис

HWND FindWindow

(

LPCTSTR lpClassName, // указатель на имя класса

LPCTSTR lpWindowName // указатель на имя окна

);

* + GetWindow

*Пример:* handleW2=GetWindow(handleW1,GW\_CHILD);

Функция **GetWindow** отыскивает дескриптор окна, который имеет определенное отношение (**Z** - последовательность или владелец) к заданному окну.

Синтаксис

HWND GetWindow

(

HWND hWnd, // дескриптор первоначального окна

UINT uCmd // флажок отношения

);

* + EnumChildWindows

*Пример,* EnumChildWindows(handleW,&EnumCW,0);

Функция **EnumChildWindows** перечисляет дочерние окна, которые принадлежат определенному родительскому окну, в свою очередь, передавая дескриптор каждого дочернего окна в функцию повторного вызова, определяемую программой. Функция **EnumChildWindows** работает до тех пор, пока не будет перечислено последнее дочернее окно или функция повторного вызова не возвратит значение ЛОЖЬ (**FALSE**).

Синтаксис

BOOL EnumChildWindows

(

HWND hWndParent, // дескриптор родительского окна

WNDENUMPROC lpEnumFunc, // указатель на функцию обратного вызова

LPARAM lParam // значение, определяемое программой

);

*(функция обратного вызова должна быть реализована в программном коде и ее объявление добавлено в начало файла:* BOOL CALLBACK EnumCW(HWND, LPARAM);*)*

* + EnumWindows
* FindWindowEx

*Например,* hBut=FindWindowEx(handleW,0,"BUTTON",NULL);

## Задание

1. Для программных экспериментов необходимы три вспомогательных приложения:

* Первое: программа из лаб. работы № 2
* Второе: простейшая программа на Borland С++ (окно, поля ввода и вывода и две кнопки с обработчиками событий, можно также добавить другие элементы управления)
* Третье: простейшая программа на С# (также с кнопками и разнообразными элементами управления)

1. Создать основное приложение Win32 Project (оно будет управлять окнами других приложений)
2. Добавить в основное приложение следующие возможности:

Спрятать окно вспомогательного приложения, если оно видимо;

case HIDE\_WINDOWS\_BUTTON:

{

HWND lab2Window = FindWindow(NULL, \_T("Lab2"));

HWND borlandWindow = FindWindow(NULL, \_T("borland"));

HWND csWindow = FindWindow(NULL, \_T("cs"));

if (lab2Window)

changeWindowVisibleState(lab2Window);

if (borlandWindow)

changeWindowVisibleState(borlandWindow);

if (csWindow)

changeWindowVisibleState(csWindow);

}

break;

Сделать недоступным окно вспомогательного приложения;

case BLOCK\_LAB2\_BUTTON:

{

HWND lab2Window = FindWindow(NULL, \_T("Lab2"));

if (lab2Window) {

SetWindowText(staticBox, \_T("Захвачено окно Lab2"));

EnableWindow(lab2Window, false);

}

else

SetWindowText(staticBox, \_T("Не удалось захватить окно Lab2"));

}

break;

case BLOCK\_BORLAND\_BUTTON:

{

HWND borlandWindow = FindWindow(NULL, \_T("borland"));

if (borlandWindow) {

SetWindowText(staticBox, \_T("Захвачено окно borland"));

EnableWindow(borlandWindow, false);

}

else

SetWindowText(staticBox, \_T("Не удалось захватить окно borland"));

}

break;

case BLOCK\_CS\_BUTTON:

{

HWND cs = FindWindow(NULL, \_T("cs"));

if (cs) {

SetWindowText(staticBox, \_T("Захвачено окно cs"));

EnableWindow(cs, false);

}

else

SetWindowText(staticBox, \_T("Не удалось захватить окно cs"));

}

break;

case UNBLOCK\_BUTTON:

{

HWND lab2Window = FindWindow(NULL, \_T("Lab2"));

HWND borlandWindow = FindWindow(NULL, \_T("borland"));

HWND cs = FindWindow(NULL, \_T("cs"));

if (lab2Window)

EnableWindow(lab2Window, true);

if (borlandWindow)

EnableWindow(borlandWindow, true);

if (cs)

EnableWindow(cs, true);

SetWindowText(staticBox, \_T("Все окна разблокированы"));

}

break;

Закрыть приложение №3 (если оно работает и окно найдено)

case CLOSE\_CS\_BUTTON:

{

HWND cs = FindWindow(NULL, \_T("cs"));

if (cs)

SendMessage(cs, WM\_DESTROY, 0, 0);

}

break;

1. Управление приложением из лаб. работы № 2:

Нажать (программно) кнопку и выполнить команду;

case PRESS\_LAB2\_BUTTON:

{

HWND lab2Window = FindWindow(NULL, \_T("Lab2"));

if (lab2Window) {

HWND button1 = FindWindowEx(lab2Window, NULL, 0, \_T("Button1"));

if (button1) {

SendMessage(button1, WM\_LBUTTONDOWN, 0, 0);

Sleep(100);

SendMessage(button1, WM\_LBUTTONUP, 0, 0);

}

}

}

break;

Выполнить команду пункта меню;

case PRESS\_LAB2\_ABOUT\_BUTTON:

{

HWND lab2Window = FindWindow(NULL, \_T("Lab2"));

if (lab2Window) {

SendMessage(lab2Window, WM\_COMMAND, IDM\_ABOUT, 0);

}

}

break;

Заставить программу выполнить действия, соответствующие нажатию пользователем правой кнопки мыши;

case PRESS\_LAB2\_RBUTTON:

{

HWND lab2Window = FindWindow(NULL, \_T("Lab2"));

if (lab2Window) {

SendMessage(lab2Window, WM\_RBUTTONDOWN, 0, 0);

SendMessage(lab2Window, WM\_RBUTTONUP, 0, 0);

}

}

break;

Заставить окно переместиться по экрану (послав, соответствующие сообщения, так как это окно умеет перемещаться при движении мыши в области клиента).

case MOVE\_LAB2\_WINDOW\_BUTTON:

{

HWND lab2Window = FindWindow(NULL, \_T("Lab2"));

if (lab2Window) {

SendMessage(lab2Window, WM\_LBUTTONDOWN, 0, 0);

SetCursorPos(800, 800);

SendMessage(lab2Window, WM\_MOUSEMOVE, 0, 0);

SendMessage(lab2Window, WM\_LBUTTONUP, 0, 0);

}

}

break;

Найти и переименовать все дочерние окна вспомогательного приложения.

case RENAME\_LAB2\_WINDOWS\_BUTTON:

{

HWND lab2Window = FindWindow(NULL, \_T("Lab2"));

if (lab2Window) {

EnumChildWindows(lab2Window, &renameLab2Windows, 0);

}

}

break;

BOOL CALLBACK renameLab2Windows(HWND hwnd, LPARAM lParam) {

SendMessage(hwnd, WM\_SETTEXT, 0, LPARAM(LPCSTR(\_T("Renamed"))));

return TRUE;

}

В основное приложение добавить кнопку, при нажатии на которую программа найдет и пронумерует все запущенные в системе окна (а также и их дочерние). Окнам следует дать имена с номерами (например, Окно1, Окно2, Дочернее3, Дочернее4 и т.д.).

case RENAME\_ALL\_WINDOWS\_BUTTON:

{

EnumWindows(&renameWindow, 0);

}

break;

BOOL CALLBACK renameWindow(HWND hwnd, LPARAM lParam) {

TCHAR parentLabel[MAX\_LOADSTRING] = \_T("window");

TCHAR number[MAX\_LOADSTRING];

\_itot(windowNumber, number, 10);

\_tcscat(parentLabel, number);

windowNumber++;

SendMessage(hwnd, WM\_SETTEXT, 0, LPARAM(LPCSTR(parentLabel)));

EnumChildWindows(hwnd, &renameChildWindows, 0);

return TRUE;

}

BOOL CALLBACK renameChildWindows(HWND hwnd, LPARAM lParam) {

TCHAR parentLabel[MAX\_LOADSTRING] = \_T("child");

TCHAR number[MAX\_LOADSTRING];

\_itot(windowNumber, number, 10);

\_tcscat(parentLabel, number);

windowNumber++;

SendMessage(hwnd, WM\_SETTEXT, 0, LPARAM(LPCSTR(parentLabel)));

EnumChildWindows(hwnd, &renameChildWindows, 0);

return TRUE;

}

1. Запустить все приложения (четыре + плюс стандартные Windows и и.д. и пронумеровать окна (визуально определить примерное их количество, а также какие элементы являются окнами, а какие нет).